UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Unusual Event Detection in Surveillance Videos

Tiago Santana de Nazare

Tese de Doutorado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

ICMC USP SÃO CARLOS

Tiago Santana de Nazare

Unusual Event Detection in Surveillance Videos

Doctoral dissertation submitted to the Instituto de Ciências Matemáticas e de Computação – ICMC-USP, in partial fulfillment of the requirements for the degree of the Doctorate Program in Computer Science and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. Moacir Antonelli Ponti
Co-advisor: Prof. Dr. Rodrigo Fernandes de Mello

USP – São Carlos
February 2021

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados fornecidos pelo(a) autor(a)

A634m

Antonelli, Humberto Lidio
Modelo de teses e dissertações em LaTeX do ICMC / Humberto Lidio Antonelli; orientadora Renata Pontin de Mattos Fortes. -- São Carlos, 2017.
79 p.

Tese (Doutorado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2017.

1. Modelo. 2. Monografia de qualificação. 3. Dissertação. 4. Tese. 5. Latex. I. Fortes, Renata Pontin de Mattos, orient. II. Título.

Tiago Santana de Nazare

Detecção de anomalias em vídeos de segurança

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Doutor em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Moacir Antonelli Ponti
Coorientador: Prof. Dr. Rodrigo Fernandes de Mello

USP – São Carlos
Fevereiro de 2021

To everyone who helped me through this journey.

ACKNOWLEDGEMENTS

First of all, I would like to say my most sincere thanks to all the people that I forgot to mention in this acknowledgments. My lousy memory have most certainly made me mention less than half of all the people that helped me to conclude this thesis. So, from the bottom of my heart, my apologies and thanks to all of you.

I would like to thank all my family members for their patience and support during this entire journey. Thank you Ana Flávia for keep me going during the several times I thought about given up. I love you! Thank too my family (Solange, Maria, Rita and Gilberto) for all their encouragement and love during this 28 years. My thanks go also to relatives: Antônio, Aparecida, Jorge, Angelina, Mario, Polyana, Afonso, Beatriz, Marcos, Claudia, Antônio Ângelo, Neide, Taciana, Ciro, Marcia, Romildo, Ana Beatriz and João Henrique. None of this would be possible without you guys!

I would like to express my gratitude to all my advisers (in chronological order). Thanks mom and dad (Solange and Gilberto) for showing me the life-changing importance of studding. My most sincere gratitude to all my teachers for helping me build the foundation that allowed me to learn everything that I know about Computer Vision and Machine Learning. Thanks Moacir Antonelli for introducing me to the fascinating world of Computer Vision and for supervising/helping me for the last 10 years. Thank you very much Francisco Restivo for showing me that research is more about being part of a challenging/wonderful universe and less about publishing brilliant papers. Thank you Barbara Caputo for completely changing the way I approach research problems and for all the help during my stay in Rome. Thank you Eduardo Hruschka for all the technical discussions and advice during the two years that we worked together at Itaú. A huge thank you to Rodrigo Mello for helping me with several research ideas, carefully reviewing this text, believing that finishing this thesis was something doable and being a role model regarding academic career. Thank you all very much for making this thesis possible!

Thank you very much to all my friends from: VICG (Mineiro, Karina, Vô, Welitão, Gabriela, Vilson, Fabinho, Danilão and Tácito), Rep. Éden (Erick, Lemuel, Oliver, Caio, Giba and Tanaka), Itaú (Xandinho, Diandra, Fernando, Lucas, Vitão, Marcão, Manuel, Fernanda, Diego, Alceu, Edilson, Azeka, A.C., Frazato, ...), TFG (Fed, Saraiva, Gola, Temaki, Nog, Vini, ...), Santander (Rafa Gomes, Daniel, Lucas, Rapha Lucio and Catini), Comp. 010 (Jau, Julio and Batman) and FIA (Alessandra, the entire staff and students). You guys are awesome!

Last but not least, I would like to thank *Instituto de Ciências Matemáticas e de Com-*

putação (ICMC), *Fundação de Amparo à Pesquisa do Estado de São Paulo* (FAPESP) (FAPESP project number 2015/04883-0), *Coordenação de Aperfeiçoamento de Pessoal de Nível Superior* (CAPES) and *Brazil–Europe Erasmus Mundus* (BEMUNDUS) for funding this research. Additionally, I would like to point out that any opinions, findings, and conclusions expressed in this manuscript are those of the author and do not necessarily reflect the views, official policy or position of ICMC, FAPESP, CNPq and BEMUNDUS.

“The fact is, there aren’t just two sides to any issue, there’s almost always a range of responses, and ‘it depends’ is almost always the right answer in any big question.”

(Linus Torvalds)

ABSTRACT

NAZARE, T. S. **Unusual Event Detection in Surveillance Videos**. 2021. 108 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2021.

Presently, surveillance cameras have been massively employed to monitor public spaces such as malls, train stations and airports. The video feed generated by several of these security cameras is monitored, in real-time, by a small group of people in a control room to detect anomalous behaviors. Nonetheless, human monitoring is extremely ineffective when it comes to detecting anomalies in surveillance footage, mainly because such task is both tedious (most of the time nothing interesting/abnormal happens) and challenging (a single person is in responsible for keeping track of multiple cameras at the same time). The aforementioned problems motivated the machine vision community to develop automated methods for detecting unusual behaviors in security videos. Despite recent advances in this area, we have noticed that current video anomaly detection methods have some gaps regarding: i) the lack of noise removal/management techniques when modeling motion patterns using optical-flow estimates; and ii) the need for a more adaptive approach to tackle changes in viewing distances. Motivated by those issues, we proposed some methods/studies aiming at improving anomaly detection in surveillance videos, while maintaining (or reducing) computational cost. Our experiments show that employing lightweight filtering to optical-flow estimates and anomaly scores can significantly improve anomaly detection performance in surveillance scenarios, without increasing computational complexity. Furthermore, we presented an automatic method to estimate changes in object size caused by variations in viewing changes, which is capable of improving anomaly detection performance and reducing setup time. Based on those findings, we designed an anomaly detection method that is capable to achieve state-of-the-art anomaly detection performance in challenging surveillance scenarios employing only optical-flow information. Additionally, we showed that a domain-specific auto-encoder is capable of achieving comparable anomaly detection results to the ones of features from pre-trained CNN models, while having a significant lower computational complexity (smaller number of network parameters).

Keywords: Surveillance videos, Anomaly detection, Optical-flow, Computer vision.

RESUMO

NAZARE, T. S. **Detecção de anomalias em vídeos de segurança**. 2021. 108 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2021.

Atualmente, câmeras de segurança têm sido amplamente usadas para monitorar espaços públicos, como shoppings, estações de trem e aeroportos. O vídeo gerado por várias dessas câmeras de segurança é monitorado, em tempo real, por um pequeno grupo de pessoas em uma sala de controle para detectar comportamentos anômalos. No entanto, o monitoramento humano é extremamente ineficaz quando se trata de detectar anomalias em vídeos de segurança, principalmente porque tal tarefa é tediosa (na maioria das vezes nada de interessante/anormal acontece) e difícil (uma única pessoa é responsável por monitorar várias câmeras ao mesmo tempo). Tais problemas motivaram a comunidade de visão computacional a desenvolver métodos automatizados para detectar comportamentos incomuns em vídeos de segurança. Apesar dos recentes avanços nessa área, notamos que os atuais métodos de detecção de anomalias em vídeos de segurança têm algumas lacunas como: i) falta de uso de técnicas de remoção/tratamento de ruído ao modelar movimentos usando fluxo óptico; e ii) necessidade de uma abordagem mais adaptativa para lidar com as mudanças de tamanho dos objetos causadas por distorçcões de perspectiva. Motivados por essas questões, propusemos alguns métodos/estudos com o objetivo de melhorar a detecção de anomalias em vídeos de segurança, mantendo (ou reduzindo) o custo computacional. Nossos experimentos mostram que o uso de técnicas simples de filtragem das estimativas de fluxo óptico e *scores* de anomalias podem melhorar significativamente o desempenho da detecção de anomalias em cenários de vigilância, sem aumentar a complexidade computacional. Além disso, apresentamos um método que automaticamente estima alterações no tamanho do objeto causadas por distorções de perspectiva, o que ajuda a melhorar o desempenho de detecção de anomalias e reduzir o tempo de configuração do sistema de seguraça. Com base nessas descobertas, projetamos um método de detecção de anomalias, que usa somente informações de fluxo óptico, e é capaz de obter resultados de detecção de anomalias muito bons em cenários desafiadores. Além disso, mostramos que um *auto-encoder* treinado para um cenário específico de vigilância é capaz de alcançar resultados de detecção de anomalias comparáveis aos de *features* de CNNs pré-treinadas, mesmo tendo uma complexidade computacional significativamente menor (menor número de parâmetros de rede).

Palavras-chave: Vídeos de segurança, Detecção de anomalias, Fluxo ótico, Visão computacional.

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

`ReLU`	*Rectified Linear Unit*
AMDN	*Appearance and Motion DeepNet*
AUC	*Area Under the ROC Curve*
CCE	*Categorical Cross-Entropy*
CNN	*Convolutional Neural Network*
EER	*Equal Error Rate*
EMD	*Empirical Mode Decomposition*
Fast MMNB	*Fast Mixed-Membership Naive Bayes*
FPR	*False Positive Ratio*
HOFM	*Histograms of Optical Flow Orientation and Magnitude*
IMFs	*Intrinsic Mode Functions*
IPCA	*Incremental PCA*
ITQ	*Iterative Quantization Hashing*
LMH	*Local Motion Histogram*
LMP	*Local Motion Pattern*
LNND	*Local Nearest Neighbor Distance*
MLPs	*Multilayer Perceptrons*
MSE	*Mean-Squared Error*
OC-SVMs	*One-Class SVMs*
ROC	*Receiver Operating Characteristic*
SDAEs	*Stacked Denoising Auto-Encoders*
SSD	*Sum of Squared Differences*
STVV	*Spatio-Temporal Video Volume*
TCPs	*Temporal CNN Patterns*
TPR	*True Positive Ratio*
WEMD	*Wavelet Earth Mover's Distance*

CONTENTS

CHAPTER

1

INTRODUCTION

Our ever-increasing security concerns lead to the massive deployment of surveillance cameras in public spaces, such as airports, malls and subway stations (ASHBY, 2017). The images captured by those cameras have been used, in real-time, to monitor such environments and, eventually, to support the detection of unusual activities (PIZA *et al.*, 2019). Such monitoring is most typically performed from control rooms, where human operators continuously search for abnormal behaviors. Despite being helpful in detecting dangerous events, the human-based monitoring is susceptible to errors due to two main reasons: i) human operators cannot sustain an adequate level of attention for longer than 20 minutes while watching security footage (SEIDENARI; BERTINI, 2010), and ii) a single operator is usually responsible for keeping track of multiple cameras at the same time (LUVISON *et al.*, 2011) what hampers the surveillance quality. In order to address these issues, various automated anomaly detection systems have been proposed to assist operators in detecting anomalous events, thus improving the effectiveness of video surveillance.

Over the last few years, the machine vision community has proposed a great deal of automatic video surveillance methods (OLUWATOYIN; WANG, 2012), which have been achieving practical results in specific scenarios (LI; MAHADEVAN; VASCONCELOS, 2014). To build those models, several computer vision techniques have been employed, such as: dictionary learning (LU; SHI; JIA, 2013; LEE *et al.*, 2015), handcrafted spatio-temporal features (HU *et al.*, 2014), optical-flow features (COLQUE; CAETANO; SCHWARTZ, 2015; PONTI; NAZARE; KITTLER, 2017), pre-trained CNNs (RAVANBAKHSH *et al.*, 2018; NAZARE; MELLO; PONTI, 2018), auto-encoders (XU *et al.*, 2015) and GANs (RAVANBAKHSH *et al.*, 2017). Yet, the detection of unusual events in surveillance videos remains as an open question (NAPHADE *et al.*, 2019). Two of the main challenges composing this domain are: i) the production of highly discriminative motion features based on noisy optical-flow estimates, and ii) the adaptation to object sizes and velocities due to variations in terms of object-to-camera distances (even when we have static surveillance cameras).

Concerning noise, optical-flow methods estimate motion vectors in attempt to represent the movement of every pixel between consecutive frames. Such estimate is prone to be incorrect/noisy due to factors like occlusions and illumination variations (LIU *et al.*, 2019), which may lead to the poor motion modeling and, lastly, hamper the performance of anomaly detection methods. This problem has been reported in the literature, but most approaches try to reduce its effects without directly filtering out optical-flow estimates. For instance, Adam *et al.* (2008) apply a post-processing stage on anomaly scores to mitigate the effects of outlier scores in attempt to reduce the amount of false positives. We believe that it is possible to further improve the performance of optical-flow-based methods if we focus on improving the robustness of flow estimates instead of simply suppressing the effects of incorrect estimates.

Regarding object size/velocity variations, even in noise-free scenarios, optical-flow may fail to offer reliable estimates (GEORGE *et al.*, 2018). This is due to changes in the viewing distance of an object (object-to-camera distance) as it moves along the scene, what alters the relationship between the real world velocity (e.g. in mph) and pixel velocities. A common way of dealing with variations in viewing distance is to train a different anomaly detection model for each video region (COLQUE; CAETANO; SCHWARTZ, 2015), requiring the training footage to contain a sufficient representation of normal behavior patterns for every video region. As this kind of approach typically employs fixed-size regions, object and monitoring region sizes may become incompatible. To surpass this problem, a common approach is to process the video at multiple region sizes or resolutions (LU; SHI; JIA, 2013; XU *et al.*, 2015), increasing the computational demands. A more recent alternative is to manually construct non-uniform grids to account for the different object scales, while sustaining the overall computational complexity (LEYVA; SANCHEZ; LI, 2014; GEORGE *et al.*, 2018).

Inspired by the aforementioned challenges, this PhD thesis aims to improve anomaly detection in surveillance applications by investigating methods to filter out noise from optical-flow estimates while adapting to variations in object sizes/velocities. More precisely, this thesis was based on the following 2-fold **hypothesis**:

> *i) the removal of noise from optical-flow estimates improves video anomaly detection results;*
>
> *ii) the capability of automatically adapting to changes in viewing distances boosts anomaly detection results.*

This PhD thesis presents the following contributions, which are going to be presented in further details during this manuscript:

- A method to improve the detection of anomalous events by attenuating noise in optical-flow estimates;

- An adaptive approach to accommodate changing object velocities in light of their viewing distances;
- An analysis of a time series decomposition method as a way of removing noise from optical-flow estimates;
- A compact 8-dimensional motion pattern (a.k.a. descriptor) that achieves relevant anomaly detection results at a reduced computational cost.

Additionally, we bring two insights regarding the usage of transfer-learning to video anomaly detection scenarios. First, we present a broad comparison on the usage of pre-trained image and video-based CNN models in surveillance scenarios. Then, we show that a domain-specific autoencoder provides similar anomaly detection results in comparison to models that leverage features from pre-trained *Convolutional Neural Network* (CNN) models.

1.1 Outline

The remaining of this PhD thesis is organized as follows:

Chapter 2 reviews some technical background on anomaly detection, optical-flow estimation, signal decomposition, and deep convolutional neural networks;

Chapter 3 presents and discusses the related work;

Chapter 4 describes our proposed methods to improve anomaly detection in surveillance videos;

Chapter 5 reports our results, including a study of hyper-parameters and comparisons against several methods from the literature;

Chapter 6 draws concluding remarks and future research directions.

CHAPTER

2

TECHNICAL BACKGROUND

2.1 Opening remarks

This chapter presents an overview of the fundamental concepts used to address the problem of detecting anomalous events on security footage. Such concepts are employed on the following chapters to design our approach and the experimental evaluation.

2.2 Anomaly detection in surveillance videos

We start by briefly introducing the general concept of anomaly detection and discuss its similarities to noise removal and novelty detection (COSTA, 2014). Furthermore, we consider which requirements an anomaly detection method should meet in order to be more successful in practical video surveillance scenarios (ADAM *et al.*, 2008).

According to Hawkins (1980), normal observations are prevalent in some dataset, being the clear result of the expected probability distribution[1], while other observations may rise the suspicion of belonging to a different behavior thus being referred to as anomalies (see Figure 1 as an illustration). In attempt to detect anomalies, we must beforehand estimate the normal behavior to next infer the normalcy of new observations. An alert is issued whenever an anomaly is detected, indicating a problem may have happened (e.g. a network intrusion detection, banking fraud or mass panic in crowded environments). Besides being applied in the context of anomaly detection, normalcy modeling is also used in novelty and outlier detection, differing in the way they leverage unusual behaviors. Novelty detection algorithms look for changes in data distributions in order to adapt learning models to reflect new behaviors (ALBERTINI; MELLO, 2007), while outlier detection often considers anomalous samples as incorrect/noisy observations, removing or fixing them out to improve the dataset quality.

[1] In the context of this PhD thesis, the normal or expected behavior is defined from situations with no need for intervention or enforcement of security protocols.

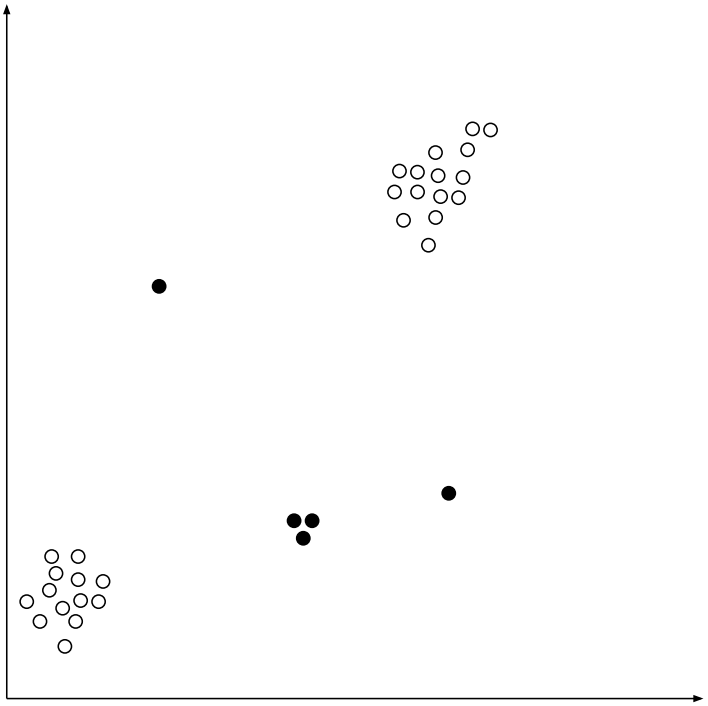


Figure 1 – An example of a 2D feature space containing few anomalies. Filled circles represent anomalies, while the remaining ones are associated to normal behavior (Adapted from Figure 2.1 of (COSTA, 2014)).

A useful anomaly detection method should be capable of obtaining a good detection performance, while having a low false positive rate. Nonetheless, when working in the video surveillance domain, there are other requirements to successfully tackle real-world scenarios, as compiled by Adam *et al.* (2008):

- Simple and fast tuning/setup process;
- Ability to adapt to environmental changes;
- Robustness to crowded and cluttered scenes;
- Automatic learning process, which (preferably) requires a small training set;
- Low computational cost.

In the context of this PhD thesis, a series of concepts is considered to address the anomaly detection in surveillance videos among which the optical-flow estimation is of great relevance, as discussed next.

2.3 Optical-flow estimation

The optical-flow estimation $h_t(.)$ takes two images as input and provides a difference vector $(u,v)_{x,y}$ for every pixel, by solving the following optimization problem:

$$\begin{aligned} {}_{(u,v)_{x,y}} ||I_{t-1}(x,y) - I_t(x+u,y+v)||_2^2 + \lambda ||(u,v)_{x,y}||_2^2 \\ (u,v) \in -M,\ldots,M \times -N,\ldots,N \\ \forall (x,y), \end{aligned}$$

given I_{t-1} and I_t are the two images under the same dimensions $M \times N$, the pixel position is expressed as (x,y), and λ is a regularization term multiplied by the norm of the difference of vector $(u,v)_{x,y}$, thus producing a set of vectors representing the displacements of all pixels from image $t-1$ to t in form:

$$h_t(I_{t-1},I_t) = \begin{bmatrix} (u,v)_{1,1} & (u,v)_{1,2} & \cdots & (u,v)_{1,N} \\ (u,v)_{2,1} & (u,v)_{2,2} & \cdots & (u,v)_{2,N} \\ \vdots & & \ddots & \vdots \\ (u,v)_{M,1} & (u,v)_{M,2} & \cdots & (u,v)_{M,N} \end{bmatrix}.$$

Ideally, one would expect to find perfect matches for each pixel, however it is unfeasible to obtain such ideal correspondence on real-world applications due to factors like noise, lighting variations and occlusions (LIU *et al.*, 2019). Therefore, by assuming that those variations are small, optical-flow estimation algorithms try to construct flow vectors $(u,v)_{x,y}$ so that $I_t(x+u,y+v)$ is as close as possible to $I_{t-1}(x,y)$, but not necessarily identical (i.e. pixels do not move too far from one image to its next).

The optical-flow approaches proposed by Horn and Schunck (1981) and Lucas and Kanade (1981) are among the two most relevant ones with regards to their applicability onto real-world problems and their influence on most modern approaches. Despite the fact that both minimize a pixel-correspondence error function to obtain a good optical flow estimate, the former operates globally on images while the latter does it locally. Thus, Horn and Schunck (1981) compute the difference vectors considering influences from the entire image such that $\forall (u,v) \in -M,\ldots,M \times -N,\ldots,N$, while Lucas and Kanade (1981) only consider spatially close pixels to perform such a task $\forall (u,v) \in -m,\ldots,m \times -n,\ldots,n$, in which $m < M$ and $n < M$. For a more complete overview on optical-flow approaches and their applications please refer to (RADKE, 2012).

Regarding surveillance applications, optical flow can be computed for temporally-close video frames and, then, be used to describe object-motion characteristics such as velocity and direction (ADAM *et al.*, 2008; COLQUE; CAETANO; SCHWARTZ, 2015; GEORGE *et al.*, 2018). However, the construction of motion estimates may lead to problems in the presence of variations in object-to-camera distances, i.e. when objects move along the scene (LEYVA; SANCHEZ; LI, 2014; GEORGE *et al.*, 2018), or when incorrect/noisy flow estimates are obtained (PONTI; NAZARE; KITTLER, 2017).

2.4 Signal decomposition using EMD and mutual information

One attempt to better analyze a time series is by decomposing it into deterministic and stochastic components, model each of those separately and, then combine their analysis to obtain better results. To this end, Rios and Mello (2016) combined the *Empirical Mode Decomposition* (EMD) (HUANG *et al.*, 1998) and Darbellay and Vajda's mutual information estimator (DARBELLAY; TICHAVSKY, 2000) to automatically perform this sort of decomposition.

2.4.1 Empirical Mode Decomposition

EMD (HUANG *et al.*, 1998) was designed to decompose a signal into *Intrinsic Mode Functions* (IMFs) without making any assumption regarding the linearity, stationarity, and stochasticity of the signal. EMD models a signal $y(t)$ as a sum of IMFs – each IMF is referred as $x_n(t)$ – plus a residue $r(t)$, as defined in the following equation:

$$y(t) = \sum_n x_n(t) + r(t). \tag{2.1}$$

IMFs are computed using an iterative process that begins by taking $x(t)$ and finding its local minima and maxima. Then, upper $u(t)$ and lower $l(t)$ envelopes are obtained applying a cubic spline to the local minima and maxima points, respectively, and used to compute a mean envelope $m(t)$. After, $m(t)$ is subtracted from $y(t)$ to calculate an IMF candidate. This candidate is considered an IMF if it meets one of the following requirements: i) $m(t)$ is zero for every point; ii) the number of extrema and zero-crossing points differ by at most one. Otherwise, the process is repeated – replacing the original data by the IMF candidate – until a candidate that meets the requirements is found. When an IMF is obtained, it is subtracted from the signal and this entire process is repeated. The process stops when the remaining signal becomes a monotonic function, being this last component considered as the residue $r(t)$.

2.4.2 IMF analysis for signal decomposition

Visually inspecting the phase spectra extracted from IMFs, Rios and Mello (2016) noticed that determinism increases along every consecutively extracted IMF. Based on such result, they used the method presented in (DARBELLAY; TICHAVSKY, 2000) to compute the mutual information v_n between the phase spectra of two consecutive IMFs ($\theta(x_n(t))$ and $\theta(x_{n+1}(t))$), as defined in the following equation:

$$v_n = \mathrm{MI}(\theta(x_n(t)), \theta(x_{n+1}(t))). \tag{2.2}$$

Leveraging this mutual information analysis, they decompose $y(t)$ into a stochastic and a deterministic component. To this end, they assumed that IMFs with higher frequencies have lower

mutual information values, while mutual information increases for low-frequency IMFs. Hence, the deterministic component is given by the sum of all IMFs with higher mutual information, while the stochastic components comprise the remaining ones.

2.5 Deep feedforward neural networks

Deep feedforward neural networks, such as *Multilayer Perceptrons* (MLPs), are supervised learning techniques that aim to approximate functions, being employed to address machine learning problems such as classification, regression and anomaly detection. Those networks are composed of several layers of neurons (or layers of processing units) arranged in a way that information always moves forward and, consequently, the connections between neurons do not form any cycle. There is always an input and an output layer, while the intermediary ones are referred to as hidden layers (for a visual representation, see Figure 2 which depicts a MLP with two hidden layers).

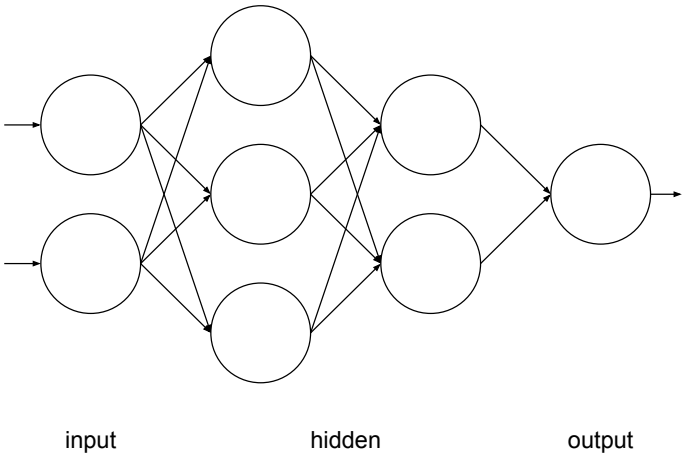


Figure 2 – An example of a MLP architecture with two hidden layers.

When data is fed into a neural network, every neuron computes its activation (output) a using the following equation:

$$a = f(\mathbf{w}\mathbf{x} + b),$$

in which $\mathbf{x}$ represents the input vector, $\mathbf{w}$ is the weight vector used to inhibiting or exciting each input, b is a bias term and f is called the activation function (see Figure 3 for a visual representation of the neuron activation). With regards to activation functions, several of them have been proposed in the literature mainly to make MLP training faster and allow the usage of deeper models. Figure 4 presents two of the most widely employed activation functions: sigmoid and *Rectified Linear Unit* (`ReLU`) (NAIR; HINTON, 2010).

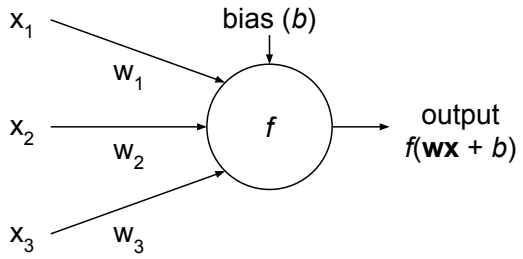


Figure 3 – Visual representation of the neuron activation.

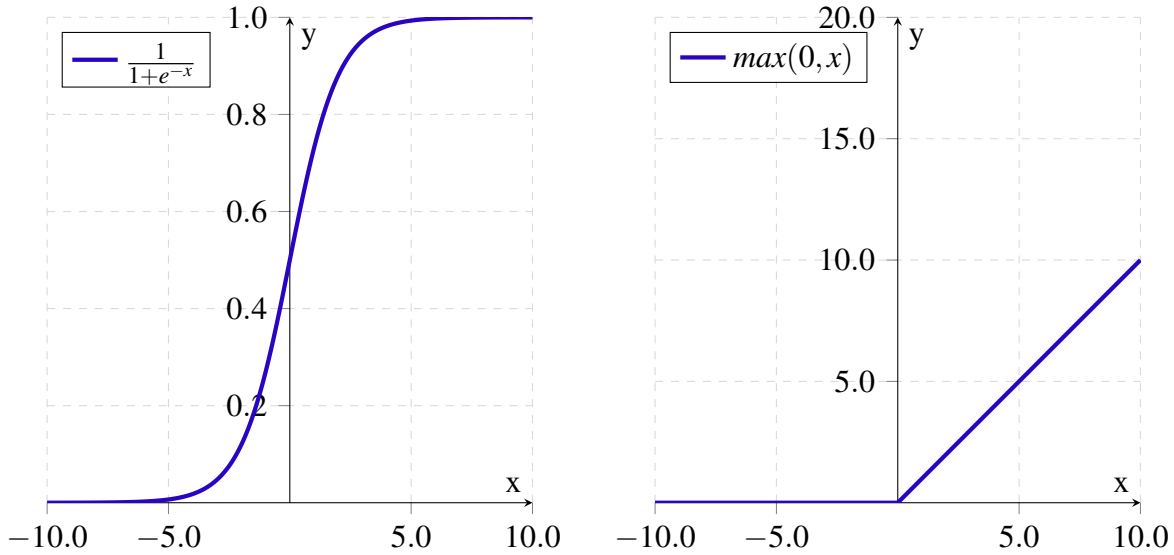


Figure 4 – Examples of activation functions: Sigmoid (left) and `ReLU` (right). Please note that the range of both plots are equal for the y axis, but differ in the x axis.

Now that we established that a MLP is formed by a set of connected neurons hierarchically organized along layers, we can start understanding how such model can be explored to approximate a function. Generally speaking, to use a MLP to learn the behavior of a certain function g^* we need to properly set its neuron parameters (weights and biases) in a way that, given an input vector $\mathbf{x}$, the model generates an output that is as close as possible to $g^*(\mathbf{x})$. To this end, we try to learn those parameters from a set of samples $(\mathbf{x}, y)$ – called training set – where $\mathbf{x}$ is an input vector and y is the mapping of $\mathbf{x}$ using g^* ($y = g^*(\mathbf{x})$).

In order to accomplish such task, we need a learning algorithm that estimates a good approximation of g^* from the training samples. Usually, when dealing with neural networks, the **gradient descent method** is applied to obtain this approximation. This method is widely adopted due to its ability in reducing errors, simplicity of use, and computational efficiency when simultaneously optimizing multiple variables (i.e. the weights and biases of a large network) in face of analytical approaches.

Gradient descent iteratively adapts the network parameters so the output of the induced model approximates the correct function value. Consequently, there is a need of quantifying

the difference between the approximation produced by the network and the actual function value. This is carried out by a loss function such as the *Mean-Squared Error* (MSE) and the *Categorical Cross-Entropy* (CCE), which are commonly used in regression and classification problems, respectively. These two functions are defined bellow:

$$\text{MSE}(\mathbf{W},\mathbf{b}) = \frac{1}{2n}\sum_{\mathbf{x}} ||y-\hat{y}||^2,$$

$$\text{CCE}(\mathbf{W},\mathbf{b}) = \frac{1}{n}\sum_{\mathbf{x}} \left(-\sum_{i=0}^{C} y_i log(\hat{y}_i) \right),$$

where $\mathbf{W}$ denotes all network weights, $\mathbf{b}$ represents all biases, n is the number of instances composing the training set, $\mathbf{x}$ is a training sample, y is the expected output for $\mathbf{x}$, $\hat{y}$ is the output in fact generated by the network, i indexes a particular class (on a classification problem) and C is the number of classes.

Using this loss function, the gradient descent method addresses the optimization problem by making small parameter changes towards the error reduction, i.e., in attempt to reduce the divergence between the network outputs and the expected values. Mathematically, consider some loss function $\mathbf{L}$ (e.g. MSE) and a set of variables $\mathbf{v}$ (e.g. weights and biases of a MLP), in order to reduce $\mathbf{L}$, we could change $\mathbf{v}$ as follows:

$$\Delta\mathbf{v} = -\eta\nabla\mathbf{L}, \tag{2.3}$$

in which $\eta > 0$ is a small positive value known as learning rate and $\nabla\mathbf{L}$ represents the gradient of the loss function $\mathbf{L}$.

Knowing that $\Delta\mathbf{L} \approx \nabla\mathbf{L}\cdot\Delta\mathbf{v}$ and using Equation 2.3, we obtain:

$$\begin{aligned}\Delta\mathbf{L} &\approx \nabla\mathbf{L}\cdot\Delta\mathbf{v}\\ &\approx \nabla\mathbf{L}\cdot(-\eta\nabla\mathbf{L})\\ &\approx -\eta\nabla\mathbf{L}\cdot\nabla\mathbf{L}\\ &\approx -\eta||\nabla\mathbf{L}||^2.\end{aligned}$$

Given that $||\nabla\mathbf{L}||^2 \geq 0$ and $\eta > 0$, we conclude $\Delta\mathbf{L} \leq 0$ and, therefore, $\mathbf{L}$ either decreases or stays the same. Nevertheless, notice this statement is based on an approximation ($\Delta\mathbf{L} \approx \nabla\mathbf{L}\cdot\Delta\mathbf{v}$), having greater chances of holding for smaller variations of $\mathbf{v}$. For that reason, we tend to choose small values of η when optimizing (training) some neural network.

At the last step, network parameters are updated based on the partial derivatives of the loss function with regards to weights and biases. Before doing so, we need to establish the following unambiguous notation:

- w^l_{jk}: weight at the connection between the j^{th} neuron at layer l, coming from the k^{th} neuron at layer $l-1$;

- b_j^l: bias associated with the j^{th} neuron at layer l;
- z_j^l: weighted sum of inputs and bias of the j^{th} neuron at layer l ($z_j^l = \sum_k w_{jk}^l a_k^{l-1} + b_j^l$);
- f: activation function;
- a_j^l: activation of the j^{th} neuron at layer l ($a_j^l = f(z_j^l)$);
- L: depth (number of layers) of the network which is used to index the output layer.

Once we have this notation, we define the derivative of the loss function with regards to the weighted sum of the inputs for any neuron j at any layer l as follows:

$$\delta_j^l = \frac{\partial \mathbf{L}}{\partial z_j^l}. \tag{2.4}$$

Based on this equation, we can find the partial derivatives for the output layer (L^{th} layer). To do so, we first use Equation 2.4 to express the partial derivative of the loss function with respect to the result of the j^{th} output neuron and – given that $\mathbf{L}$ does not directly depend on z_j^L, but rather on a_j^L – we apply the chain rule and rewrite the previous equation as follows:

$$\begin{aligned}
\delta_j^L &= \frac{\partial \mathbf{L}}{\partial z_j^L} \\
&= \frac{\partial \mathbf{L}}{\partial a_j^L} \frac{\partial a_j^L}{\partial z_j^L} \\
&= \frac{\partial \mathbf{L}}{\partial a_j^L} f'(z_j^L).
\end{aligned}$$

At this point, we can use δ_j^L to compute partial derivatives with regards to any weight w_{jk}^L and any bias b_j^L at the output layer. Regarding the weights, the derivative of δ_j^L is obtained as follows:

$$\begin{aligned}
\frac{\partial \mathbf{L}}{\partial w_{jk}^L} &= \frac{\partial \mathbf{L}}{\partial a_j^L} \frac{\partial a_j^L}{\partial z_j^L} \frac{\partial z_j^L}{\partial w_{jk}^L} \\
&= \frac{\partial \mathbf{L}}{\partial a_j^L} f'(z_j^L) \frac{\partial z_j^L}{\partial w_{jk}^L} \\
&= \delta_j^L \frac{\partial z_j^L}{\partial w_{jk}^L} \\
&= \delta_j^L \frac{\partial (\sum_k w_{jk}^L a_k^{L-1} + b_j^L)}{\partial w_{jk}^L} \\
&= \delta_j^L a_k^{L-1}.
\end{aligned}$$

Analogously it is possible to obtain $\frac{\partial \mathbf{L}}{\partial b_j^L} = \delta_j^L$. Having these two equations, we are able to leverage the gradient descent method to update weights and biases for any neuron at the output layer.

In a similar way, we use Equation 2.4 to find the partial derivatives for the weighted sum of any hidden neuron. To this end, we apply the chain rule to rewrite this equation in terms of δ_j^{l+1}, obtaining the following expression:

$$\begin{aligned}\delta_j^l &= \frac{\partial \mathbf{L}}{\partial z_j^l} \\ &= \sum_k \frac{\partial \mathbf{L}}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_j^l} \\ &= \sum_k \frac{\partial z_k^{l+1}}{\partial z_j^l} \delta_k^{l+1}.\end{aligned}$$

Then, we derive $\frac{\partial z_j^{l+1}}{\partial z_j^l}$ using $z_j^{l+1} = \sum_i w_{ki}^{l+1} a_i^l + b_k^{l+1}$ and $a_i^l = f(z_i^l)$ – which makes $f(z_i^l)$ vanish for $i \neq j$ – as shown in the following equation:

$$\begin{aligned}\frac{\partial z_j^{l+1}}{\partial z_j^l} &= \frac{\partial (\sum_i w_{ki}^{l+1} a_i^l + b_k^{l+1})}{\partial z_j^l} \\ &= \frac{\partial (\sum_i w_{ki}^{l+1} f(z_i^l) + b_k^{l+1})}{\partial z_j^l} \\ &= w_{kj}^{l+1} f'(z_j^l).\end{aligned}$$

Based on these last two results, we define δ_j^l as follows:

$$\delta_j^l = \sum_k w_{kj}^{l+1} f'(z_j^l) \delta_k^{l+1}.$$

To find the derivatives for weights of any hidden neuron, we do as follows:

$$\begin{aligned}
\frac{\partial \mathbf{L}}{\partial w_{jk}^l} &= \sum_i \frac{\partial \mathbf{L}}{\partial z_i^{l+1}} \frac{\partial z_i^{l+1}}{\partial a_j^l} \frac{\partial a_j^l}{\partial z_j^l} \frac{\partial z_j^l}{\partial w_{jk}^l} \\
&= \sum_i \delta_i^{l+1} \frac{\partial z_i^{l+1}}{\partial a_j^l} \frac{\partial a_j^l}{\partial z_j^l} \frac{\partial z_j^l}{\partial w_{jk}^l} \\
&= \sum_i \delta_i^{l+1} \frac{\partial (\sum_m w_{im}^{l+1} a_m^l + b_i^{l+1})}{\partial a_j^l} \frac{\partial a_j^l}{\partial z_j^l} \frac{\partial z_j^l}{\partial w_{jk}^l} \\
&= \sum_k \delta_i^{l+1} w_{ij}^{l+1} \frac{\partial a_j^l}{\partial z_j^l} \frac{\partial z_j^l}{\partial w_{jk}^l} \\
&= \sum_i \delta_i^{l+1} w_{ij}^{l+1} \frac{\partial f(z_j^l)}{\partial z_j^l} \frac{\partial z_j^l}{\partial w_{jk}^l} \\
&= \sum_i \delta_i^{l+1} w_{ij}^{l+1} f'(z_j^l) \frac{\partial z_j^l}{\partial w_{jk}^l} \\
&= \sum_i \delta_i^{l+1} w_{ij}^{l+1} f'(z_j^l) \frac{\partial (\sum_k w_{jk}^l a_k^{l-1} + b_j^l)}{\partial w_{jk}^l} \\
&= \sum_i \delta_i^{l+1} w_{ij}^{l+1} f'(z_j^l) a_k^{l-1} \\
&= \sum_i \delta_i^{l+1} w_{ij}^{l+1} f'(z_j^l) a_k^{l-1} \\
&= a_k^{l-1} \sum_i w_{ij}^{l+1} \delta_i^{l+1} f'(z_j^l) \\
&= a_k^{l-1} \delta_j^l.
\end{aligned}$$

Finding the derivatives for the biases ($\frac{\partial \mathbf{L}}{\partial b_j^l} = \delta_j^l$) is rather similar, therefore we decided to omit them in order to avoid repetition. Then, the network parameters (weights and biases) are updated using the following rules:

$$\begin{aligned}
w_{jk}^l &\leftarrow w_{jk}^l - \frac{\eta}{n} \sum_{\mathbf{x}} a_k^{l-1}(\mathbf{x}) \delta_j^l(\mathbf{x}) \\
b_j^l &\leftarrow b_j^l - \frac{\eta}{n} \sum_{\mathbf{x}} \delta_j^l(\mathbf{x}).
\end{aligned}$$

One problem with such approach is that it uses all n training samples at every parameter update, what may consume an excessive amount of computational resources, slowing down the training stage. In attempt to mitigate this issue, the mini-batch version of the gradient descent method is commonly used. In this technique, instead of using the entire training set to estimate the gradients, a set of m ($m \ll n$) training samples are randomly taken and used to perform such estimation, what significantly reduces the training time. Such approach relaxes training by providing a more rough estimate at each iteration using a mini-batch of examples. Other optimization and training techniques are relevant to induce good deep neural network models and can be found in (PONTI *et al.*, 2017) and (GOODFELLOW; BENGIO; COURVILLE, 2016).

2.6 Convolutional neural networks

If we try to use the networks presented in the last section to classify images using their pixels, we would have to turn every image into a vector, consequently, loosing all their natural spatial structure. Convolutional neural networks (CNNs) (LECUN *et al.*, 1998) were proposed to address such visual content by using local learners represented in terms of neurons with their respective convolution kernels (or filters) instead of globally adapting in response to dense combinations of the inputs. Based on the last section, we now address two fundamental building blocks in order to understand convolutional neural networks: convolutional and pooling layers (PONTI *et al.*, 2017). Then, mapping the other concepts from a regular network to a convolutional one is fairly straightforward.

A **convolutional layer** differs from a dense one mainly because each of its neurons learns a convolutional filter (or kernel) instead of learning a dense combination of all inputs. A convolutional neuron i from a layer l generates its output matrix Y_i^l (also known as feature map) as in the following equation:

$$Y_i^l = f(K_i^l * Y^{l-1} + b_i^l),$$

where K_i^l is the filter learned by a particular neuron, Y^{l-1} is the output tensor provided by the previous layer or the input image (in case of the first layer), $*$ refers to as the convolution operator, b_i^l is the bias learned by a particular convolutional neuron and f is the activation function. In the previous equation, the convolution is a matrix operation, while the addition of the bias term and the application of the activation function are pointwise operations. By doing this process, each neuron of a layer generates an output matrix, all being stacked up to generate the output tensor of such layer (as shown in Figure 5).

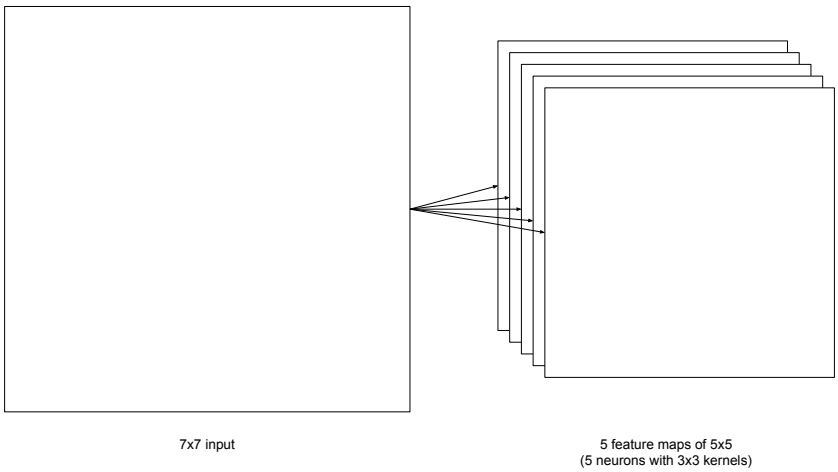


Figure 5 – Feature maps stacked up to compose the output tensor of a convolutional layer. In this case, the hidden layer contains 5 feature maps, each one of size 5×5. To obtain such feature map size, considering a 7×7 input, we used 3×3 filters and a stride equals to 1.

Another important characteristic of CNNs is the presence of **pooling layers** that are used

after a convolutional layer to reduce the amount of information of its feature maps. Pooling layers divide the feature map in equally-sized regions which are then summarized, typically by taking their maximal or average activation, as depicted in Figure 6. In more recent deep convolution networks, pooling is avoided in favor of using larger strides in the convolution process (HE *et al.*, 2016; CHOLLET, 2017; HUANG *et al.*, 2017).

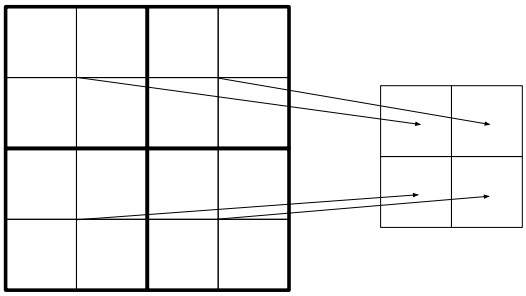


Figure 6 – Illustration of the pooling operation. A 4×4 region is summarized by a pooling layer that operates on 2×2 regions.

Lastly, despite the fact that all concepts on this sections were explained based on 2D networks, mapping them to 1D or 3D models – such as the C3D network (Section 2.8.5) and the proposed 3D auto-encoders (Section 5.4) – is straightforward. For a more in-depth view of such concepts, the reader may refer to (GOODFELLOW; BENGIO; COURVILLE, 2016).

2.7 Auto-encoders

An auto-encoder is a neural network typically used in unsupervised learning scenarios, in which the learning process is not guided by predicting labels but rather by copying the input data. In order to do so, an auto-encoder first uses an encoder to create a more restrict (or compact) representation of the input data (also called code), then, it uses a decoder to reconstruct the original data (the general structure of an auto-encoder is illustrated in Figure 7). By learning such restricted representation of the original data, an auto-encoder can discover useful data properties, instead of simply copying the input data.

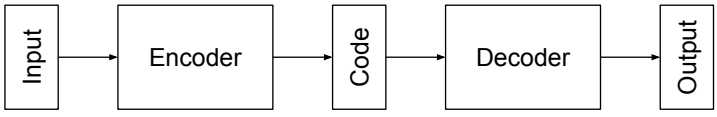


Figure 7 – Auto-encoder general structure.

When building an auto-encoder, there are two ways of obtaining a restricted representation: by undercompleteness and by regularization. Creating an **undercomplete** representation means that the code generated by the encoder has a smaller dimension than the one of the input data, which compels the auto-encoder to try to capture some intrinsic data properties. On the other hand, when working with **regularization**, instead of having shorter codes, we limit the

copying capabilities by adding a regularization term to the loss function. Sparse auto-encoders are one example of regularized auto-encoders that employ sparsity penalty on the generated code, what makes their reconstruction error to be defined as follows:

$$\mathbf{L}(\mathbf{x}, d(e(\mathbf{x}))) + \Omega(\mathbf{h}),$$

where $\mathbf{L}$ is a loss function that compares the original input $\mathbf{x}$ against its reconstruction $d(e(\mathbf{x}))$, e is the encoder part of the model, d is the decoder part, $\mathbf{c_x} = e(\mathbf{x})$ is the code generated by the encoder and Ω represents the sparsity penalty. Another example of models that use regularization are denoising auto-encoders. Instead of trying to reconstruct its original (uncorrupted) input, these models first apply some kind of noise to their input, then try to perform the reconstruction. Hence, their reconstruction error is given by:

$$\mathbf{L}(\mathbf{x}, d(e(\tilde{\mathbf{x}}))),$$

where $\tilde{\mathbf{x}}$ is a noisy version of the original input (e.g. if $\mathbf{x}$ is an image, $\tilde{\mathbf{x}}$ can be the same image corrupted by a Gaussian noise). As a consequence, in order to reduce the loss value, the model must be capable of removing noise, not simply copy the input.

2.8 CNN architectures

In this section, we discuss some widely used CNN architectures for video and image classification. These models produce rather generic features and, therefore, have been successfully employed in tasks other than the one they were trained for, thus comprising transfer-learning techniques. We start by presenting four widely used image classification models (VGG (SIMONYAN; ZISSERMAN, 2014), ResNet (HE *et al.*, 2016), Xception (CHOLLET, 2017) and DenseNet (HUANG *et al.*, 2017)), which were capable of obtaining remarkable results on the ImageNet (DENG *et al.*, 2009) classification challenge. Then, we present the C3D (TRAN *et al.*, 2015) architecture, that was proposed for action recognition in videos and uses 3D convolutions to process video segments. Later, these pre-trained CNN models are used in our experiments as feature extractors.

2.8.1 VGG networks

VGG networks take as input 224×224 RGB images and process them using five blocks of convolutional layers with 3×3 kernels (sometimes, 1×1 filters are also used) and convolutional stride equals to 1. Moreover, padding is carried out so it prevents convolutional layers to reduce spatial resolution; in other words, a padding of 1 is used for 3×3 filters, while there is no padding for 1×1 kernels. The number of neurons for the convolutional layers of each of the convolutional blocks is always: 64, 128, 256, 512 and 512, with the smaller layers as the ones closer to the input and the larger ones nearby the output. At the end of each block of convolutional layers,

there is a 2×2-windowed max-pooling layer using a stride equals to 2. After the convolutional part of the architecture, there are three fully connected layers: two `ReLU` layers with 4096 neurons followed by an output softmax layer with 1000 neurons. To illustrate the above mentioned VGG characteristics, we depict the architectures of one of its most successful variants, also used in some of our experiments, the VGG-16 network (see Figure 8).

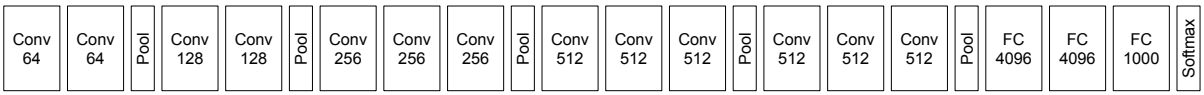


Figure 8 – VGG-16 architecture.

2.8.2 ResNet

One of the limitations of VGG-like models is their inability to converge when using more than 19 layers, what is caused by the vanishing/exploding gradient problem (BENGIO; SIMARD; FRASCONI, 1994). To overcome such issue, He *et al.* (2016) proposed the usage of residual units (see Figure 9) instead of sequential connections between convolutional layers. Such approach facilitates the propagation of error on deeper CNN models, consequently mitigating the vanishing/exploding gradient problem. Their approach then allows to train networks with more than 150 layers and outperformed the VGG on the ImageNet image classification challenge, while having less parameters.

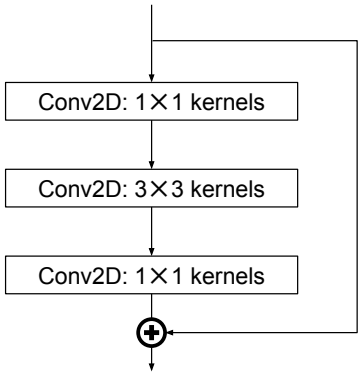


Figure 9 – Example of residual unit used in ResNet models (Adapted from Figure 2 of (NAZARE; MELLO; PONTI, 2018)).

2.8.3 Xception

The Xception architecture was proposed by Chollet (2017), inspired by Inception V3 (SZEGEDY *et al.*, 2016). In this CNN model, the standard convolution is replaced by a variation referred to as depthwise separable convolution, which is computed by applying the same convolutional filter (e.g. 3×3) to each tensor channel separately and, then, using a regular 1×1 convolution to combine all channels (as shown in Figure 10). By employing this methodology, Xception

models reduce the number of trainable parameters of each layer, consequently leveraging deeper architectures. This model outperformed Inception V3 on the ImageNet classification competition.

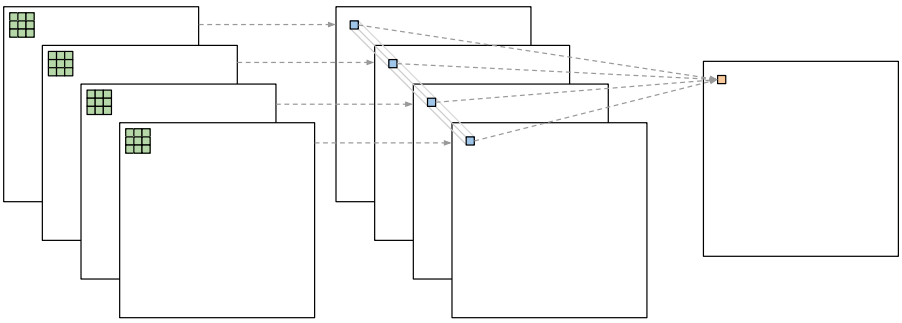


Figure 10 – Depthwise separable convolution layer (Adapted from Figure 3 of (NAZARE; MELLO; PONTI, 2018)).

2.8.4 DenseNet

Inspired by residual units, Huang *et al.* (2017) proposed DenseNets, which use shortcut/skipping connections to link every layer to all of its subsequent layers. By using more shortcut connections, the authors were able to obtain even deeper models (up to 201 layers), which led them to win the ImageNet classification challenge in 2017.

2.8.5 C3D networks

One of the seminal studies to understand the use of 3D convolutions to model videos was introduced in (TRAN *et al.*, 2015), which heavily explored 3D convolutions using CNN models and achieved state-of-the-art results in several action recognition benchmarks. Later, other researchers concluded that pre-trained networks based on the same principle as well as their features extracted from specific datasets could be used to tackle other tasks, inherently leading to transfer learning applications. Among all models, the 3D CNN trained on the Sports-1M dataset is the most successful with regards to producing transferable features. This is the reason we decided to review and analyze the features generated by such a network model which is, from now on, referred to as the C3D network.

The C3D network processes video segments with 16 224×224 frames, containing: 8 3D-convolutional, 2 fully connected, 5 max-pooling and one softmax layers, organized as illustrated in Figure 11. All eight convolutional layers use $3 \times 3 \times 3$ kernels and the `ReLU` activation function, having 64, 128, 256, 256, 512, 512, 512 and 512 kernels, respectively.

Of the five max-pooling layers, the first (`Pool1`) has a pooling kernel of $1 \times 2 \times 2$ – so that it only affects the two spatial dimensions, not the temporal one – while the other four use $2 \times 2 \times 2$ kernels, consequently their pooling operations occur in the spatial and temporal dimensions. Additionally, this network has three fully connected layers: two hidden ones (`fc6` and `fc7`) – which have 4096 neurons each, use `ReLU` as activation function and have a dropout

probability equals to 50% – and the output layer (`fc8`) that uses `softmax` activation and has 487 neurons (equal to the number of classes present in the Sports-1M dataset).

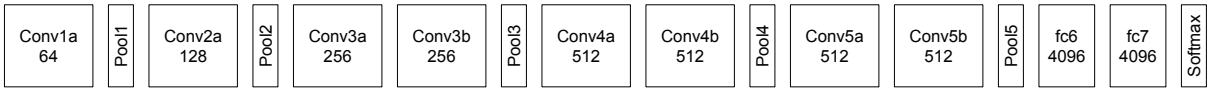


Figure 11 – C3D architecture employed on the Sports-1M dataset.

2.9 Concluding remarks

This chapter presented an overview on the fundamental concepts of: anomaly detection for surveillance applications, optical flow and CNNs. These concepts are necessary to understand the methods reviewed in Chapter 3, our proposals in Chapter 4, and the experimental results presented and discussed in Chapter 5.

CHAPTER

3

RELATED WORK

3.1 Opening remarks

In this chapter, we review several methods devoted to detect unusual events in surveillance videos by applying a myriad of strategies, including dictionary learning, optical-flow features, deep neural networks. After analyzing those methods, we listed gaps of existing video anomaly detection approaches, mainly: i) the lack of robustness to deal with noise, especially when working with optical flow data; and ii) the complexity of automatically adapting to object-size variations due to changes in viewing distances. The former gap is mostly related to the need of filtering noisy feature estimates, while the later is greatly associated to the usage of fixed-size monitoring video regions (referred to as monitors, windows, cuboids, spatial regions or video blocks). Those gaps comprise the main motivation for this PhD thesis whose contributions are discussed from Chapter 4 on.

3.2 Local motion histogram method

The *Local Motion Histogram* (LMH) method was introduced by Adam *et al.* (2008) to deal with real-time anomaly detection in security videos, thus adapting to environmental changes, while requiring small training samples. This method is primarily based on grids of low-level feature monitors to detect anomalies in different image regions (Figure 12 depicts the location of monitors).

Each monitor extracts low-level features (i.e. optical flow magnitude or optical flow orientation) only at its particular location in order to reduce the overall computational cost involved in this stage which is similar to the ones presented in (ROSENBERG; WERMAN, 1997; SHI; MALIK, 1998), being divided into the following steps:

I – Flow Probability Matrix: For two sequential frames, I_{t-1} and I_t, the method assumes that

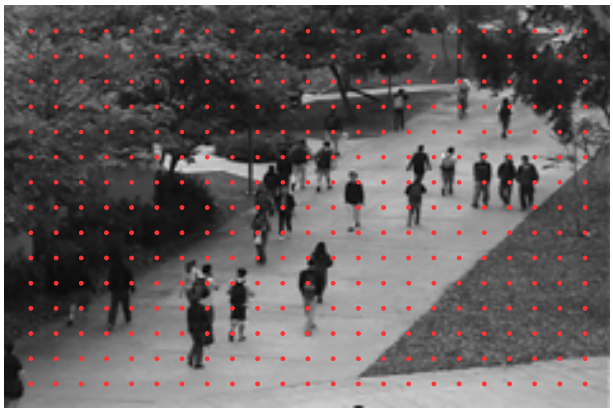


Figure 12 – Example of 23×15 grid of monitors, in which the red dots represent of the monitors.

all pixels within a template window W, centered at the monitor coordinates, have the same displacement. Then, it computes a *Sum of Squared Differences* (SSD) matrix, in which each element $SSD(u,v)$ represents the sum of squared differences between the elements of W in I_{t-1} and the elements of $W+(u,v)$ in I_t, as shown in Equation 3.1. Afterward, it transforms SSD into a probability distribution P using Equation 3.2, having c as the normalization factor, σ^2 corresponding to the SSD variance.

$$SSD(u,v) = \sum_{i,j \in W} (I_t(i+u, j+v) - I_{t-1}(i,j))^2 \tag{3.1}$$

$$P(u,v) = c \times e^{-\frac{SSD(u,v)}{\sigma^2}} \tag{3.2}$$

II – Binning: Using the flow probability matrix obtained at the last step, the algorithm calculates another probability distribution by aggregating the flow probabilities $P(u,v)$ by angle or magnitude (depending on the low-level feature). Thus, a new probability distribution $P(O_i)$ is generated, in which O_i represents the possible aggregated observations.

III – Ambiguity Test: As optical flow values may be imprecise, the method performs an ambiguity test to verify if the most likely observation O_{ML} can be considered valid. Given the distance between observations $d(O_i, O_j)$ and an ambiguity threshold A_T, O_{ML} is considered valid only if Equation 3.3 holds true. Otherwise, the monitor does not produce an observation for that frame.

$$\sum_i P(O_i) d(O_{ML}, O_i) < A_T \tag{3.3}$$

Each monitor keeps the last n valid observations in a buffer, which is used to compute the likelihood of new observations. In case such likelihood is below a given threshold, the monitor considers that there is an anomaly and produces an alert. Initially, the authors considered frames to be anomalous when at least one of the monitors produced an alert. Nonetheless, due to the presence of noise on the optical-flow values, they noticed that such approach led to a large

number of false alarms. In attempt to reduce the number of false alarms, LHM only issues an anomaly if at least n of the last y frames are anomalous.

3.3 Sparse combination method

Lu, Shi and Jia (2013) proposed the Sparse Combination method to detect anomalies in video footage in real-time with a low computational cost, achieving high detection rates while processing 150 FPS on a PC with a 3.4 GHz CPU with 8 GB RAM. Those results are the consequence of their sparsity-based dictionary learning method which leverages the redundancy of surveillance videos.

In such method, the video is firstly partitioned into non-overlapping spatial regions under the same size (e.g. 20×10 pixels). Next, those regions are divided into temporal segments of 5 consecutive frames described using the 3D gradient features by Kratz and Nishino (2009). The feature vector of each segment is then reconstructed by sparsely combining patterns from a learned normalcy dictionary, so that anomaly is characterized by large reconstruction errors.

Prior to such work, sparsity-based abnormality detection models were derived from training sets $\mathscr{X} = \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}\}$ only composed of normal data patterns in attempt to build some dictionary $\mathbf{D} \in \mathbb{R}^{p \times q}$ which was then used to sparsely reconstruct some new data sample $\mathbf{x}$ as follows:

$$\begin{aligned} \min_{\beta} \quad & ||\mathbf{x} - \mathbf{D}\beta||_2^2 \\ \text{s.t.} \quad & ||\beta||_0 \leq s, \end{aligned} \tag{3.4}$$

in which $\beta \in \mathbb{R}^{q \times 1}$ represents sparse coefficients, s is a sparsity parameter ($s \ll q$) and $||\mathbf{x} - \mathbf{D}\beta||_2^2$ is the fitting error (used to judge the normalcy of new observations). Such anomaly detection approach typically presents high computational demands, given that there are $\binom{q}{s}$ different ways of selecting s vectors from some dictionary with q patterns. Therefore, the computational cost of such sparse reconstruction does not allow real-time anomaly detection when employing a large dictionary.

When facing the real-time requirements of surveillance applications, the authors proposed a lightweight sparse combination learning that takes advantage of the highly redundant structure of surveillance videos. In their approach, given the same training set $\mathscr{X} = \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}\}$, instead of learning a dictionary, they learn a set of K combinations of basis vectors $\mathscr{S} = \{\mathbf{S_1}, \mathbf{S_2}, \ldots, \mathbf{S_K}\}$. Consequently, the reconstruction error for a new sample $\mathbf{x}$ is obtained by:

$$\min_{\beta^i} ||\mathbf{x} - \mathbf{S_i}\beta^{\mathbf{i}}||_2^2 \quad \forall\, i = 1, 2, \ldots, K, \tag{3.5}$$

in which $\mathbf{S_i} \in \mathbb{R}^{p \times s}$ is a set (combination) of basis vectors and $\beta^i \in \mathbb{R}^{s \times 1}$, which is obtained by a closed-form, contains the sparse coefficients. This means that the reconstruction error is

computed by simply calculating the least square error between $\mathbf{x}$ and each $S_i \in \mathscr{S}$ and, then, taking the lowest value which makes it possible to detect anomalies in real-time. Besides, due to this low computational cost, the authors were able to process each frame at 3 different scales (resolutions) to tackle object size variations while still being able to run their method in real-time.

3.4 Local nearest neighbor distance method

Local Nearest Neighbor Distance (LNND) (HU *et al.*, 2014) is a video anomaly detection technique that uses a compact description of video blocks. Such description is generated by employing the novel LNND descriptor, that considers, not only the spatio-temporal characteristics of a block, but also its spatio-temporal context.

3.4.1 LNDD descriptor

Initially, a video V is divided into a set $\{V_{s,t}\}$ of spatio-temporal cuboids, where s and t are the spatial and temporal coordinates of the cuboids, respectively. Those cuboids are partially overlapped and have the same spatio-temporal size of $h \times w \times \tau$, in which h is the spatial height, w is the spatial width and τ is the temporal duration (in number of frames) of the cuboid.

Once we have the spatio-temporal cuboids, the description process begins by computing the magnitude of the spatio-temporal gradient for each pixel within the cuboid, generating the gradient cuboids set $\{G_{s,t}\}$. Then, each gradient cuboid $G_{s,t}$ is described using the *Local Motion Pattern* (LMP) descriptor presented by Guha and Ward (2012)[1].

A cuboid description using LMP begins by calculating the variance, skewness and kurtosis – which are the second, third and fourth central moments – along the temporal dimension, for every spatial coordinate. These moments are stored, respectively, in three $h \times w$ matrices: M_2, M_3 and M_4. Therefore, each element of M_r ($r \in \{2,3,4\}$) is defined as:

$$M_r(i,j) = \frac{1}{\tau} \sum_{t=1}^{\tau} (G(i,j,t)^r), \tag{3.6}$$

where $G(i,j,t)$ is the spatio-temporal gradient magnitude at coordinate (i,j,t), τ is the temporal length of a cuboid and r is the moment being calculated. These matrices are transformed into vectors m_2, m_3 and m_4 and, then, concatenated to generate the final LMP descriptor with $3 \times h \times w$ dimensions (see Figure 13 for a visual representation of such procedure).

Next, for every cuboid $V_{s,t}$, a spatial neighborhood SN and a temporal neighborhood TN are defined. In SN, every cuboid is at the same time interval t as $V_{s,t}$ and, in TN, every cuboid is at the same spatial position s as $V_{s,t}$. The distance between $V_{s,t}$ and one of its spatial or temporal

[1] The original version of LMP applies a Gaussian blur on the raw pixel values, instead of computing the magnitude of the spatio-temporal gradient, prior to the calculation of the spatio-temporal descriptor.

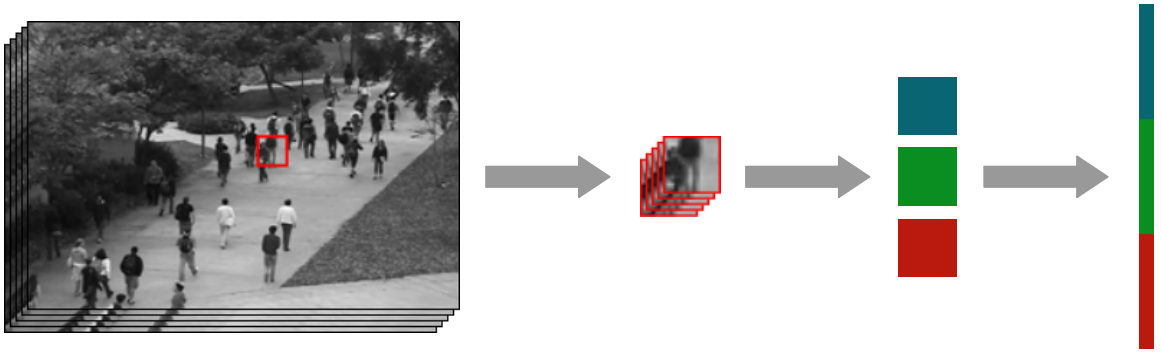


Figure 13 – Block description using LMP (based on Figure 2 of (GUHA; WARD, 2012)). First, a video segment is extracted. Then, the spatio-temporal gradient magnitudes and the three central moments (variance, skewness and kurtosis) are calculated, generating the matrices M_2, M_3 and M_4 (represented in blue, green and red, respectively). Finally, these matrices are transformed into vectors m_2, m_3 and m_4, which are concatenated to generate the final descriptor.

neighbors is defined as the *Wavelet Earth Mover's Distance* (WEMD) (SHIRDHONKAR; JACOBS, 2008) of their LMP descriptions.

The final LNND descriptor is constructed in three steps. Firstly, the smaller K distances between $V_{s,t}$ and its spatial neighbors are taken in ascending order. This generates a vector $X^{sd} = [d_1, d_2, \ldots, d_K]^T$, where d_k is the kth smallest distance between $V_{s,t}$ and one of its spatial neighbors. Secondly, by the same process, the smaller N distances between $V_{s,t}$ and its temporal neighbors are taken in ascending order. This generates another vector $X^{td} = [d_1, d_2, \ldots, d_N]^T$, where d_n is the nth smallest distance between $V_{s,t}$ and one of its temporal neighbors. Lastly, the final LNND is obtained by concatenating X^{sd} and X^{td} as follows:

$$LNND = \begin{bmatrix} X^{sd} \\ X^{td} \end{bmatrix}, \tag{3.7}$$

generating a descriptor with $K+N$ dimensions. In (HU *et al.*, 2014), the authors used $K = 8$ and $N = 1$, so the final LNND dimension was 9.

3.4.2 Training and abnormality detection

Using the LNND description of the training data – which does not contain any anomalies – a different *Fast Mixed-Membership Naive Bayes* (Fast MMNB) model is learned for each spatial region s. After the learning phase, given a new sample X_s that occurred at spatial region s, its log-likelihood l under the normalcy model of region s is calculated. If this likelihood is below a user-defined threshold, X_s is classified as an abnormality.

3.5 Motion influence map method

The Motion Influence Map method (LEE *et al.*, 2015) detects unusual events by embedding contextual motion information into the video description, as opposed to most methods that handle

context in a classification level. In order to deal with context at the feature level, a novel motion representation technique, named motion influence map, was proposed to quantify the "motion influence" between frame parts (blocks) thus helping to detect unusual motion patterns.

3.5.1 Motion influence map descriptor

The first step to construct a motion influence map of a frame is to estimate optical flow for every of its pixels. Then, the frame is divided into $M \times N$ non-overlapping blocks, as shown in Figure 14, and an optical flow value is assigned to each block by computing the average flow of its pixels.

Figure 14 – An example of a frame partitioned into uniform non-overlapping blocks.

Afterward, the weight w_{ij} is computed for all pairs of blocks i and j to quantify the influence of block i on the motion of block j by using the following equation:

$$w_{ij} = \delta_{ij}^{d} \delta_{ij}^{\phi} e^{-\frac{D(i,j)}{||\mathbf{b}_i||}}, \tag{3.8}$$

in which $D(i,j)$ is the Euclidean distance between blocks i and j, and $\mathbf{b}_i$ is the optical flow of block i. The variables δ_{ij}^{d} and δ_{ij}^{ϕ} indicate whether block i influences block j by considering their inner distance as well as the angle formed by their average flow vectors, respectively, as defined bellow:

$$\delta_{ij}^{d} = \begin{cases} 1, & \text{if } D(i,j) < T_d \\ 0, & \text{otherwise} \end{cases} \qquad \delta_{ij}^{\phi} = \begin{cases} 1, & \text{if } -\frac{F_i}{2} < \phi_{ij} < \frac{F_i}{2} \\ 0, & \text{otherwise} \end{cases} \tag{3.9}$$

in which T_d is a distance threshold, ϕ_{ij} is the angle between the vector ranging from the center of block i to the center of block j with the optical flow of block i ($\mathbf{b}_i$), and F_i is an angle threshold (the authors assumed $F_i = \pi$ in their experiments).

Using the influence weights, a motion influence map is constructed having each block j represented by an 8-bin histogram $H^j(k)$. Every bin k of $H^j(k)$ refers to an orientation range and stores the total influence (sum of influence weights) on block j in its orientation range. These

ranges are defined in Equation 3.10, in which $k \in \{1,2,3,4,5,6,7,8\}$, $\angle \mathbf{b}_i$ is the angle of the flow vector $\mathbf{b}_i$, and q is the function mapping the flow angle to one of the k orientations.

$$q(\angle \mathbf{b}_i) \equiv k \quad \text{s.t.} \quad \frac{(2k-3)\pi}{8} < \mathbf{b}_i \leq \frac{(2k-1)\pi}{8} \tag{3.10}$$

To summarize this entire process, Algorithm 1 depicts the main steps to construct a motion influence map of a frame, given the spatial position and the optical flow of each block.

Algorithm 1 – Motion influence map construction

Require: A distance threshold T_d, an angle threshold F_i, a set of blocks B.

```
for all block i do
    for all block j do
        if D(i, j) < T_d then
            if -F_i/2 < φ_ij < F_i/2 then
                H^j(q(∠b_i)) ← H^j(q(∠b_i)) + w_ij
            end if
        end if
    end for
end for
```

3.5.2 Feature extraction using motion influence maps

Feature extraction initiates by repartitioning the video, spatially and temporally, into mega blocks. A mega block is a spatio-temporal cuboid composed by $n \times n$ blocks over t sequential frames and is constructed using the motion influence representations, not the original frames. Next, the mega block descriptor is generated by summing up the motion influence vectors of all blocks within the same frame (generating an 8-dimensional vector for each of its t frames) and, then, concatenating such vectors. Hence, this entire process creates a final mega block descriptor of $8 \times t$ dimensions, as shown in Figure 15.

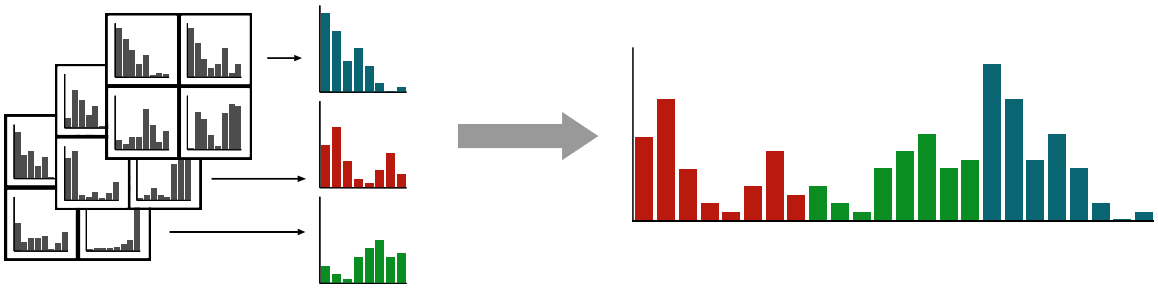


Figure 15 – Mega block description example (adapted from Figure 5 of (LEE *et al.*, 2015)). In this example a mega block is defined by 2×2 blocks over 3 consecutive frames. First, all 2×2 motion vectors are added for each frame, generating 3 motion vectors of size 8. The three vectors are concatenated to generate the final 8×3-dimensional descriptor.

3.5.3 Training and anomaly detection

Considering a training video composed solely by normal frames, codewords are learned separately for every mega block spatial position. This is done by taking all descriptions from a given spatial position, and by performing a k-means clustering to use the k centers as codewords. Thus, a region has its normal behavior represented by a dictionary with k codewords.

Lastly, in order to detect anomalies for new samples, the minimal distance between a new sample and the codewords in its correspondent spatial region is calculated. If this distance is above an user-defined threshold, the sample is considered as anomalous.

3.6 Composition pattern method

The Composition Pattern method (LI NANNAN, 2015) learns local contextual behavior in two main steps. First, local atomic patterns are learned globally, what means they do not take into account contextual information (they are atomic) neither consider spatial location (they are global). Second, the video is divided into cubes and a histogram of the distribution of atomic patterns is computed for each video cube. This process generates a new video description, called composition-pattern representation, which is used to learn normal behavior for different video spatial regions. In the two subsequent sections, we explain these steps further in details.

3.6.1 Learning atomic patterns

Before extracting features, frames are blurred using a Gaussian kernel (usually with $\sigma = 1$), then a *Spatio-Temporal Video Volume* (STVV) is built at each pixel. A STVV is a pixel cuboid of size $L_x \times L_y \times L_t$, in which L_x and L_y refer to the spatial dimensions and L_t is associated with the temporal duration in number of frames. Then, the polar representation of the spatio-temporal gradients is calculated for each STVV using the equations below:

$$M_{st} = \sqrt{G_x^2 + G_y^2 + G_t^2}, \tag{3.11}$$

$$\theta = \arctan\left(\frac{G_y}{G_x}\right), \tag{3.12}$$

$$\phi = \arctan\left(\frac{\sqrt{G_x + G_y}}{G_t}\right), \tag{3.13}$$

where M_{st} is the magnitude of the spatio-temporal gradient, $\theta \in (-\pi, \pi]$ and $\phi \in \left(\frac{-\pi}{2}, \frac{\pi}{2}\right]$ are the angles of the polar representation, $G_x = I(x+1,y,t) - I(x,y,t)$, $G_y = I(x,y+1,t) - I(x,y,t)$ and $G_t = I(x,y,t+1) - I(x,y,t)$.

In order to describe a STVV, θ and ϕ are quantized into eight and four values, respectively. Afterward, two histograms are computed: the first represents the distribution of M_{st} according to

the eight quantized values of θ, and the second represents the distribution of M_{st} according to the four quantized values of ϕ. Lastly, the atomic pattern descriptor is constructed by concatenating these two histograms and normalizing each bin, creating a final atomic descriptor h_i that has 12 dimensions.

To learn atomic patterns, the descriptions of all STVVs are clustered using the online fuzzy weighted c-means (HORE *et al.*, 2009) and a codebook is obtained by taking the cluster centers as codewords, what makes possible to represent each STVV as a set of membership degrees U_i quantifying the similarity between the STVV and each codeword.

3.6.2 Composition descriptor

As atomic patterns are obtained globally, so they do not take into account the spatial location within the frame, some activities referred as normal in some spatial locations and taken as abnormal in others, making difficult to classify them out. For instance, a person walking on the walkway is a normal event, whereas a person walking on the grass should be considered as an abnormal event. Hence, it is possible to notice that the spatio-temporal context needs to be assessed.

To address this problem, videos are described using compositions of the above-mentioned atomic patterns. This description initiates by redividing the video into overlapping cubes, which have a larger spatio-temporal size when compared against STVVs (i.e. a video cube contains many STVVs). Next, the description of a video cube is performed in 3 steps: i) the video cube is divided into 8 blocks under the same size; ii) a membership histogram is built for each block by accumulating membership degrees of all STVVs, so each block is represented by a histogram of size C which defines the number of codewords in the atomic pattern codebook; and iii) the membership histograms of all blocks are merged in a way that bins corresponding to the same codeword are grouped in, consequently the composition descriptor has $8 \times C$ dimensions. The entire composition process is illustrated in Figure 16.

3.6.3 Anomaly detection using a dictionary of composition patterns

After the phase of the composition-pattern description, normal behaviors are learned by constructing a different sparse dictionary for each spatial position. These dictionaries are built using the following model:

$$\begin{aligned} \min_{D,\alpha} \quad & \frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{2}||\mathbf{x_i}-\mathbf{D}\alpha_\mathbf{i}||_2^2+\lambda||\alpha_i||_1\right) \\ \text{s.t.} \quad & ||\mathbf{D_j}||_2 = 1, \forall j = 1,2,...,k, \end{aligned} \tag{3.14}$$

where $\mathbf{x_i} \in \mathbb{R}^d$ is a d-dimensional training sample, $\mathbf{D} \in \mathbb{R}^{d\times k}$ is the dictionary of size k to be learned, $\mathbf{D_j} \in \mathbb{R}^d$ is one of the basis, $\alpha_\mathbf{i} \in \mathbb{R}^{k\times n}$ are the reconstruction coefficients for each one of the n training samples ($\mathbf{x_i}$) and λ is a sparsity parameter. Therefore, the greater the parameter

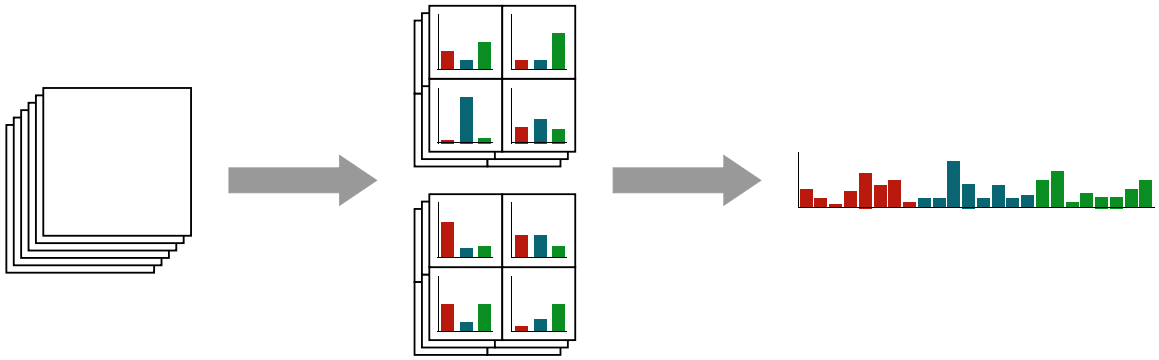


Figure 16 – Example of a composition-pattern representation with the number of codewords in the atomic pattern codebook equals to three ($C = 3$). First, the video cube is divided into 8 equally sized blocks and a membership histogram is computed for each of the 8 blocks. Then, the bins of all histograms are grouped according to the same codeword (Adapted from Figure 2 of Li Nannan (2015)).

λ is, the more sparse are the combinations. A stochastic approximation of the optimal solution for this model is obtained using the method from Mairal *et al.* (2009).

In the classification phase, given a new sample $\mathbf{x}$ and the learned dictionary $\mathbf{D}$, the reconstruction coefficients α^* are obtained from Equation 3.15. Next, the sparsity reconstruction cost of $\mathbf{x}$ is calculated using Equation 3.16 and, if such cost is greater than a given threshold, the sample is considered an anomaly.

$$\min_{\alpha} \left(\frac{1}{2} ||\mathbf{x} - \mathbf{D}\alpha||_2^2 + \lambda ||\alpha||_1 \right) \tag{3.15}$$

$$C_{\mathbf{x}} = \frac{1}{2} ||\mathbf{x} - \mathbf{D}\alpha^*||_2^2 + \lambda ||\alpha^*||_1 \tag{3.16}$$

Due to the fact that there is a different dictionary for each spatial location, the thresholds used to infer whether a sample is anomalous may vary depending on the spatial location (used dictionary) what makes hard to assign the thresholds manually. In order to tackle this problem, a Gaussian distribution of the reconstruction costs of the training samples is estimated for each spatial location. Then, thresholds are set in a way that a sample is considered to be an anomaly only if the difference between its reconstruction cost and the mean of the distribution is greater than twice the variance of the distribution.

3.7 Histograms of optical flow orientation and magnitude method

Based on the studies by Chaudhry *et al.* (2009), Colque, Caetano and Schwartz (2015) developed an anomaly detection method referred to as *Histograms of Optical Flow Orientation and Magnitude* (HOFM) to describe the optical-flow distribution on video cuboids. In their video anomaly detection approach, this descriptor is computed for non-overlapping video cuboids and

it is based on the optical flow estimated for every pair of consecutive frames. To compute the optical flow, the pyramidal implementation of the Lucas-Kanade-Tomasi algorithm (BOUGUET, 2000) was used. Additionally, to reduce processing time, they compute the difference between consecutive frames and only estimate optical flow for pixels whose change is greater than a given threshold.

3.7.1 Cuboid description using HOFM

In order to describe a cuboid they begin by obtaining the polar coordinate representation of the flow estimate. Then, they quantize the optical flow orientation, which ranges from 0 to 360 degrees, into S bins and map the optical flow magnitude from the interval $[0,\infty]$ to B bins. After that, a $S \times B$ matrix, each row corresponding to an orientation range and each column to a magnitude interval, is used to count the number of co-occurrences of quantized orientation and magnitude values. Lastly, such matrix is vectorized by concatenating its rows, generating the final HOFM descriptor. An example of a cuboid description is depicted in Figure 17.

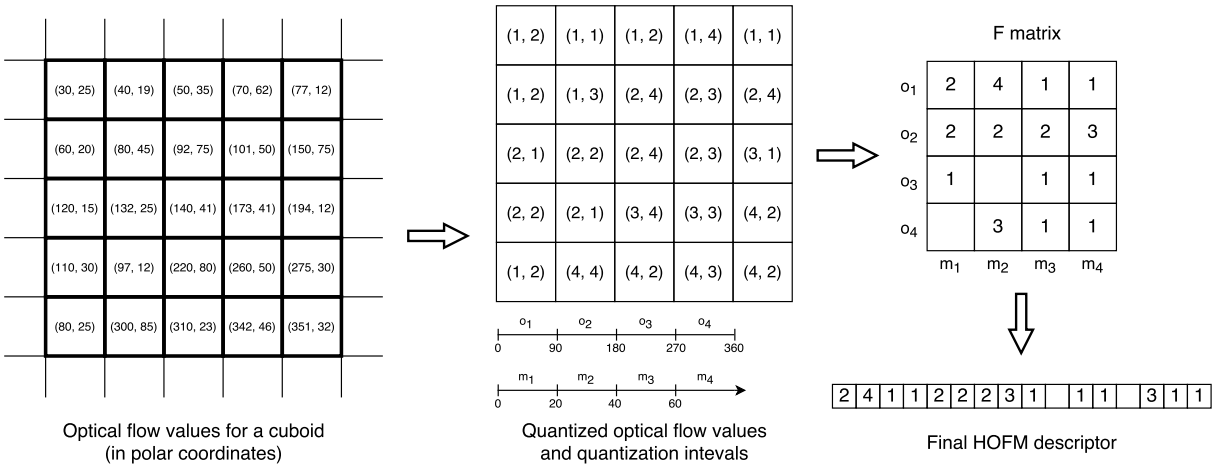


Figure 17 – An example of cuboid description using HOFM. First, an optical flow estimate is obtained (in polar coordinates) for every pixel within the cuboid. Secondly, both the optical flow orientation and magnitude are quantized. In this example, the optical flow orientation was quantized in four intervals ($\{[0,90);[90,180);[180,270),[270,360)\}$), and the magnitude was also quantized in four ranges ($\{[0,20);[20,40);[40,60),[60,\infty)\}$). Thirdly, a matrix – which is 4×4 because both orientation and magnitude were quantized in four intervals – counts co-occurrences of orientation and magnitude values. Finally, this matrix is vectorized, generating the final HOFM descriptor (Adapted from Figure 4 of (COLQUE; CAETANO; SCHWARTZ, 2015))).

3.7.2 Anomaly detection with HOFM descriptor

After computing the HOFM descriptions of all cuboids on the training data – which only contains non-anomalous observations – a new cuboid is classified by comparing its description against all training samples located at the same spatial position. If the distance of this new sample to the nearest point is greater than a given threshold, the cuboid is considered to be anomalous.

3.8 AMDN method

Xu *et al.* (2015) were the first to use an unsupervised deep learning framework – named *Appearance and Motion DeepNet* (AMDN) – to create a video representation to detect unusual behavior in security videos. Additionally, their framework uses both early and late fusion of motion and appearance information to improve performance and achieve state-of-the-art results.

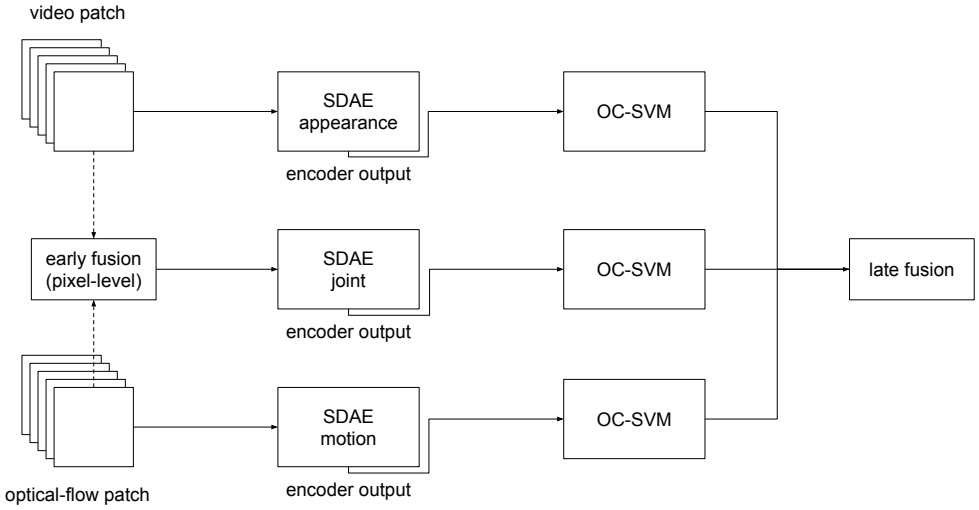


Figure 18 – AMDN framework overview. An important detail in this framework is that the OC-SVMs use the code (output of their encoder) generated by the SDAEs as feature vectors (Adapted from Figure 1 of (XU *et al.*, 2015)).

During the training phase, their method uses *Stacked Denoising Auto-Encoders* (SDAEs) to learn representations for three different sources: motion, appearance and both motion and appearance simultaneously (which provides an early fusion of the two sources of information). First the a layer-wise pre-training is carried out on the SDAE, then a fine tuning is performed on the entire model. Once the auto-encoder models are trained, the outputs of their encoder part are used to train three *One-Class SVMs* (OC-SVMs) to learn the normal behavior for each representation. Given the abnormality score of each of these three OC-SVM models can have different importance on the final anomaly detection, a weight is learned for each of them so detection become more accurate.

In the detection phase, these SDAEs are used to extract features from video patches, which are feed into the three previously trained OC-SVMs to compute an anomaly score for each of them. Then, these scores are combined, using the weights learned during the training time, and a final anomaly score – which allows a late fusion of motion and appearance information – is calculated. In this setup, anomalies are identified by thresholding this final score. An overview of the AMDN framework is depicted in Figure 18.

3.9 Plug-and-play CNN method

Ravanbakhsh *et al.* (2018) presented an abnormal event detection method for security videos that uses features extracted from an AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) CNN, pre-trained on the ImageNet dataset. Their method takes advantage of convolutional feature maps and optical flow to monitor changes in video regions and find unusual patterns. Their anomaly detection technique can be divided into three parts: generate CNN-based binary maps from a sequence of input frames, computing *Temporal CNN Patterns* (TCPs) and feature fusion (between the optical flow and TCPs). Such steps are illustrated by Figure 19.

The first step to compute the binary maps is to describe frame regions using the pre-trained AlexNet model, which is accomplished by taking the output (feature maps) of the last convolutional layer. By doing so, a n-dimensional region description is generated for each region having the size of the receptive field of such neurons. Considering the AlexNet model employed by the authors, each image region is described by a 256-dimensional feature vector, which is not convenient when trying to track temporal changes. Thus, to reduce the descriptor dimensionality, an *Iterative Quantization Hashing* (ITQ) (GONG; LAZEBNIK, 2011) is applied to convert the feature vectors into 24-bit binary codes.

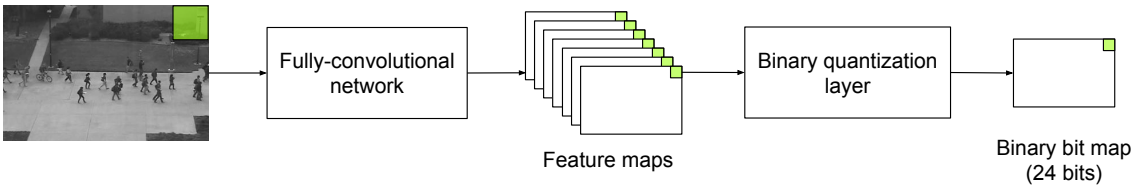


Figure 19 – The pipeline of binary maps generation.

Having the description of frame regions (also called patches), the video is then divided into temporally-overlapping video blocks. Such process is carried out by taking each patch description p_t^i – where t is the frame number and i the patch spatial position – and by creating a block b_t^i containing $L+1$ patches centered on p_t^i ($\{b_t^i\} = \{p_l^i\}_{l=t-L/2}^{t+L/2}$). Next, each block is described by a histogram h_t^i that represents the distribution of binary codes over its patches.

In attempt to infer when an appearance anomaly occurs using this histogram, the TCP measure of a block is computed as follows:

$$\mathrm{TCP}(b_t^i) = \sum_{j=1}^{|h_t^i|} ||h_t^i(j) - h_t^i(j_{\max})||_2^2,$$

where $|h_t^i|$ is the number of histogram bins, $||.||_2$ is the $\mathscr{L}_2$-norm, $h_t^i(j)$ is the j^{th} bin of h_t^i, and $j_{\max}$ is the index of the bin with the greatest count in h_t^i (i. e. $h_t^i(j_{\max})$ is the mode of h_t^i). This measure represents irregularity (deviation from the dominant pattern) in appearance, whose greater values increase the chance of having an anomaly.

As every pixel within a block has the same TCP value, TCP maps do not provide an accurate abnormality location as they still consider image regions instead of individual pixels. Aiming at improving anomaly localization, a structure named optical-flow maps is built based on the following steps: i) the optical flow magnitude is estimated for every pair of consecutive frames within the video, meaning that each pixel within a block has a corresponding optical flow magnitude value; ii) considering a block b_t^i of length $L+1$, the optical-flow map d_t^i is calculated by summing up the flow magnitude at each spatial position ($d_t^i = \sum_{l=1}^{L+1} \text{flow}_t^i(l)$) along time; and iii) the fusion between the TCP map and the optical flow map is carried out by taking their average and generating what the authors refer to the motion segment map, consequently leading regions with greater optical flow magnitudes to be more influenced by TCP values. Based on such process, anomaly detection is carried out by applying a threshold to the values on the motion segment map.

3.10 Parallelepiped spatio-temporal regions method

George *et al.* (2018) developed a technique that uses a non-uniform grid of regions to deal with distance variations between the camera and objects. Their approach uses parallelepiped spatio-temporal regions to account for object size variations on both horizontal and vertical axes, improving upon previous non-uniform grid methods (LEYVA; SANCHEZ; LI, 2017) that only consider size variations with regards to the vertical axis.

To build up the non-uniform grid, they start by defining the side of the row containing the smallest cells (top row) in the grid as y_0, and the growth rate of the cell rows as a. Next, they create cell rows, until the entire image is covered, using the following equation:

$$y_{k+1} = a y_k,$$

where k is the row id. By doing so, they are able to account for object-size variations along the vertical axis. To deal with horizontal variations, that can occur in scenes where objects are observed from an angle, they apply the following shear transform on the vertices of each cell:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & m \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix},$$

where (x', y') represents the shear transformed points and the parameter m determines the shear degree. At the end of such process, it is possible to obtain a non-uniform grid such as the one illustrated in Figure 20.

Once the cell regions are obtained, a different fully-connected auto-encoder model is learned for each region, which uses the HOFM features (COLQUE; CAETANO; SCHWARTZ, 2015) computed for that cell over the last 5 frames (for more details on the HOFM features, please refer to Section 3.7). During the anomaly detection phase, the reconstruction error of each model is used as an anomaly score for its corresponding frame region.

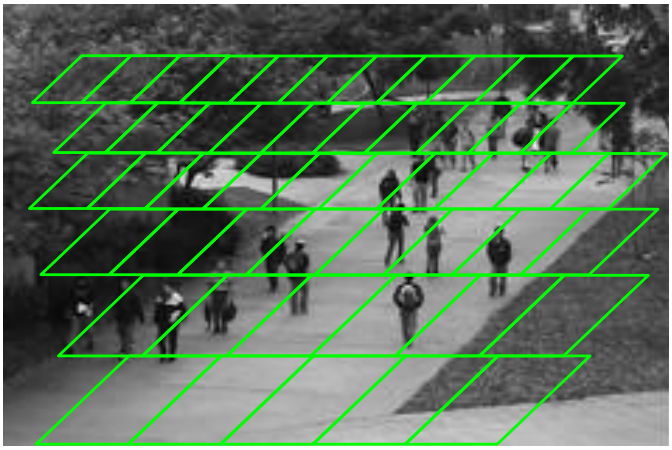


Figure 20 – An example of a non-uniform grid produced using a shear transform.

According to the experimental results, their method is capable of achieving good anomaly detection results and improving robustness with regards to changes of object sizes according to their distance to the camera. Nevertheless, their approach needs several parameters to be set in order to obtain a good non-uniform grid. Hence, this method can be rather time consuming to setup in a real-world situation, where several cameras must be monitored.

3.11 Comparison

Table 1 – Main characteristics of the video anomaly detection methods.

Method	Visual cues	Monitoring scheme	Adaptiveness	Filtering	Anomaly detection method	Feature type
LMH (ADAM *et al.*, 2008)	motion	fixed-size monitors	-	post-processing (anomaly alarms)	histogram-based likelihood	optical-flow magnitude
Sparse combination (LU; SHI; JIA, 2013)	both	fixed-size monitors	multi-scale processing	-	reconstruction error	dictionary patterns
LNND (HU *et al.*, 2014)	both	fixed-size monitors	-	-	Fast MMNB	LNDD features
Motion Influence Map (LEE *et al.*, 2015)	motion	fixed-size monitors	-	-	normalcy dictionary	dictionary patterns based on motion influence maps
Composition Pattern (LI NANNAN, 2015)	both	fixed-size monitors	muitiple models	Gaussian blur	reconstruction error	dictionary patterns
HOFM (COLQUE; CAETANO; SCHWARTZ, 2015)	motion	fixed-size monitors	muitiple models	-	nearest neighbor distance	HOFM descriptors
AMDN (XU *et al.*, 2015)	both	fixed-size monitors	multi-scale processing	denoising auto-encoders	OC-SVM scores	auto-encoder features (motion and appearance)
Plug-and-play CNN (RAVANBAKHSH *et al.*, 2018)	both	fixed-size monitors	-	-	TCP-based likelihood	CNN features and optical-flow
Parallelepiped regions (GEORGE *et al.*, 2018)	motion	handcrafted grid (non-uniform)	non-uniform grid/ multiple models	-	reconstruction error (using an auto-encoder)	HOFM descriptors

In this section, we have compiled Table 1 to summarize/compare the main characteristics of the reviewed methods, what allowed to notice the following patterns:

Visual cues: all reviewed methods employ motion cues in order to detect anomalous behavior, while only half of them also use appearance cues.

Monitoring scheme: most of the reviewed methods employ fixed-size monitoring regions, which tend to hamper their ability to adapt to viewing distance changes. To tackle this prob-

lem, George *et al.* (2018) use a non-uniform grid that accounts for object-size variations. Nonetheless, their grid needs to be manually created for each different camera, what makes their method rather timing consuming to be setup in applications with several cameras.

Adaptiveness: besides the usage of non-uniform grids of monitors – what is carried out only by (GEORGE *et al.*, 2018) – the reviewed methods that aim at adapting to changes in viewing distance either employ a different model for each monitoring location or process the video at multiple scales (spatial resolutions). The usage of multiple models tends to require more data for the training step, as all normal behaviors need to be well represented for every monitoring region. On the other hand, methods that process the same frame at multiple scales increase their computational complexity.

Filtering: for the most part, filtering the model inputs and outputs is a step that does not receive much attention in surveillance applications. Based on previous findings (ADAM *et al.*, 2008), we conclude that the filtering of surveillance video information (specially when working with optical-flow estimates) improves anomaly detection results.

Anomaly detection method: methods that employ dictionary learning and auto-encoder models tend to use the reconstruction error as an anomaly score, while other methods use some kind of likelihood (e.g. nearest neighbor, histogram-based probability, OC-SVM distances from the decision boundary) as such scores.

Feature type: about half of the reviewed methods employ feature learning by building up some auto-encoder or dictionary, while the remaining ones rely on handcrafted descriptors. Besides, the majority of reviewed methods use optical-flow estimates as part of their anomaly detection pipeline.

3.12 Concluding remarks

This chapter presented several anomaly detection methods for video surveillance applications. Even though these approaches have confirmed good performances on challenging scenarios, it is possible to notice that they still have relevant gaps: i) too many parameters must be optimized thus increasing the setup time; ii) issues are observed when dealing with noisy optical-flow estimates; and iii) videos must be analyzed multiple times at different scales and resolutions in order to account for viewing changes. These are some of the main gaps that motivated us to propose the methods detailed in Chapter 4.

CHAPTER

4

PROPOSED APPROACHES

4.1 Opening remarks

In the last chapter, we presented an overview of several methods designed to tackle the anomaly detection problem in video surveillance applications. Based on such study, we identified the following gaps in the literature: i) the lack of an in-depth understanding on the usage of CNNs on surveillance problems; ii) the need for robust methods to deal with subtle movements/changes occurring in such type of footage; and iii) automatic approaches to address different object-to-camera distances. By investigating those issues, security systems become less time consuming to be setup as well as more computationally efficient in production environments. From that, they become easier to be deployed particularly in scenarios with multiple cameras to be monitored.

Motivated by such gaps, we studied and developed a series of methods. First, we designed an auto-encoder, fully based on 3D convolutions, and employed its local reconstruction error to detect abnormal local events. Secondly, we proposed a framework to better leverage the motion information of surveillance videos. Finally, we proposed a novel method that adapts the local monitoring dimensions according to the expected average object size at each spatial location. The remaining of this chapter details all those approaches.

4.2 Anomaly detection with 3D-fully-convolutional auto-encoders

Motivated by the results obtained using auto-encoders in security videos (XU *et al.*, 2015) and by the performance of 3D-convolutional networks (TRAN *et al.*, 2015), we here introduce a fully 3D-convolutional auto-encoder architecture to detect anomalous events in surveillance videos, as shown in Figure 21. Our architecture uses only $3 \times 3 \times 3$ kernels, has 5 layers, employs $2 \times 2 \times 2$ kernels at the max-pooling and up-sampling operations, and it employs the mean-squared error

(MSE) as loss function, `ReLU` as the activation function of all layers, the Batch Normalization in every layer and Gaussian noise to corrupt input data.

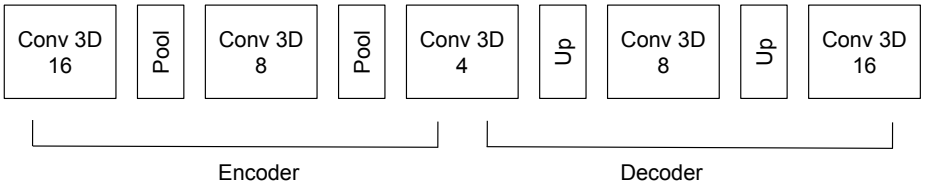


Figure 21 – 3D auto-encoder architecture. The number at each convolutional layer indicates how many filters are learned at that particular stage.

We trained our auto-encoder on $20 \times 20 \times 8$ cuboids extracted from training videos. After that, in order to detect anomalies, cuboids from the test set were reconstructed with the learned model and the MSE of such reconstruction was calculated. The MSE value is considered as an anomaly score, which means that higher values of such score indicate ha higher chance of having an anomaly. In order to illustrate the concept, Figure 22 shows a frame containing an anomaly and its reconstructed version. By comparing the two images, we notice a poor reconstruction result for the biker (anomaly) and a fairly good one for pedestrians and the footage background.

Figure 22 – Frame reconstruction using a 3D auto-encoder. The first image is the original frame and the second is its reconstructed version. Observe the reconstruction is very accurate, except for the biker (anomaly).

When compared to pre-trained CNN architectures, our 3D auto-encoders are more prone to learn specific features while using less parameters, what makes its inference more efficient. Nevertheless, our experimental results suggest that our approach struggles to detect anomalous behavior in the presence of subtle object movements and object-to-camera distance variations, what was addressed as discussed next.

4.3 Robust and adaptive motion descriptors with optical-flow filtering

Considering the gaps we have identified from the literature review and by analyzing the results of our auto-encoder model, we turn our attention to improve the robustness of optical-flow-based video anomaly detection methods. In particular, with respect to object-to-camera distance variations and subtle movements. We start tackling this issue by measuring the gains of reducing the noise on optical-flow estimates by leveraging the EMD decomposition. Despite being a tool that allowed us to demonstrate the importance of optical-flow filtering, the EMD decomposition is better suited for offline situations, such as a forensic scenarios, in which the entire time series/frames is assumed to be available beforehand. Then, we continue our study by investigating the usage of wider frame gap and a temporal median-filter to mitigate this issue, which also confirms to improve the detection performance, while being more convenient in terms of computational complexity. Lastly, we propose a novel method to adapt region monitoring size to the average object size/velocity expected for each image region.

4.3.1 Deterministic EMD components as filtered optical-flow estimate

As proposed by Rios and Mello (2016) and explained in Section 2.4, Empirical Mode Decomposition (EMD) can be used to decompose time series into two components: a deterministic and a stochastic one. One way of interpreting them is to consider the stochastic component as noise and the deterministic as a filtered time series.

Such a way of looking at EMD is particularly suitable when working with optical flow from surveillance videos, as the flow estimates between consecutive frames tend to be considerably noisy, while the flow field tends to be rather similar for temporally-close frames. Hence, the results of decomposing optical-flow estimation of a surveillance video can be seen in the following way: i) the deterministic component represents a filtered flow estimate obtained by leveraging the temporal smoothness of a surveillance video; and ii) the stochastic component can be seen as noise from the local optical-flow estimates. Based on that, we compare the detection performance of using raw and filtered local optical-flow magnitudes to detect anomalies in security videos. Using the pipeline presented in Figure 23, noisy optical-flow estimates can hamper the detection performance.

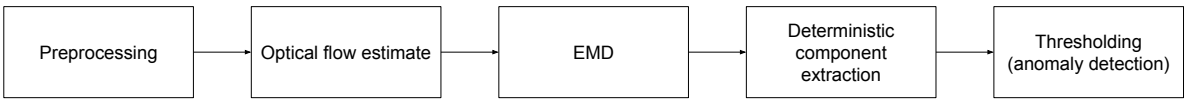


Figure 23 – The main steps involved in our pipeline.

To perform our comparison, we start by dividing the video in 30×30 overlapping

windows, then we compute a background image by taking the average of all training images (see Figure 24b). Next, we use such an image to speed-up the optical-flow estimation by identifying windows that do not contain movement, i.e. its absolute difference to our background image is lower than a user-defined threshold (illustrated in Figure 24a). Lastly, we compute the flow value for the foreground regions using the method introduced in (FARNEBÄCK, 2003) and set the flow as zero for the remaining ones. All this preprocessing approach is based on previous algorithm proposals, including (ADAM *et al.*, 2008) and (COLQUE; CAETANO; SCHWARTZ, 2015).

(a) (b)

Figure 24 – Preprocessing stage. In (a), we present the absolute difference between a frame and the background image. In (b), we have the background image obtained by computing the mean of all training images. By visually inspecting (a), one may observe that even a reasonably crowded image is mostly background, so this approach can indeed help reducing the flow estimation time (Adapted from Figure 2 of (PONTI; NAZARE; KITTLER, 2017).

Once we have the optical-flow estimates for all windows, we take the average flow magnitude of all pixels within each window and consider this value as a raw anomaly score for such a particular frame region. Next, we apply EMD on the features of each block, so the flow of each spatial position is temporally decomposed. As explained in Section 2.4, EMD iteratively generates a set of IMFs being the last ones more deterministic. Hence, it is possible to establish a cut-off point to divide IMFs into two groups: one with the more deterministic IMFs and another with the more stochastic ones. Then, by adding up IMFs belonging to each group, we obtain our two components. Roughly speaking, we perform this cut-off when the mutual information for consecutive IMFs changes abruptly. More specifically, we first obtain the mean mutual information value as follows:

$$\bar{\mathscr{V}} = \frac{1}{N-1} \sum_{n=1}^{N-1} v_n,$$

where N is the number of IMFs and v_n is the mutual information value computed between the IMFs at indices n and $n+1$. Next, we look for the greatest mutual information value v_z

that is smaller than $\bar{\mathscr{V}}$, which is then considered as the cut-off point. Hence, our deterministic component is defined as follows:

$$y_{det}(t) = \sum_{n=z+1}^{N} x_n(t) + r(t),$$

in which $x_n(t), \forall n \geq z+1$ are the deterministic IMFs and $r(t)$ is the residue. Then, we take the deterministic optical-flow component as a filtered version of the original anomaly score. Such approach generates much more reasonable motion/velocity estimates, as illustrated in Figure 25. Despite the improvements in anomaly detection when reducing noise in optical-flow estimates, such approach is limited to forensic (offline) applications as it requires the entire series to be available beforehand. Nonetheless, the results obtained by the EMD decomposition confirm the importance of reducing the amount of noise in optical flow estimates in surveillance scenarios.

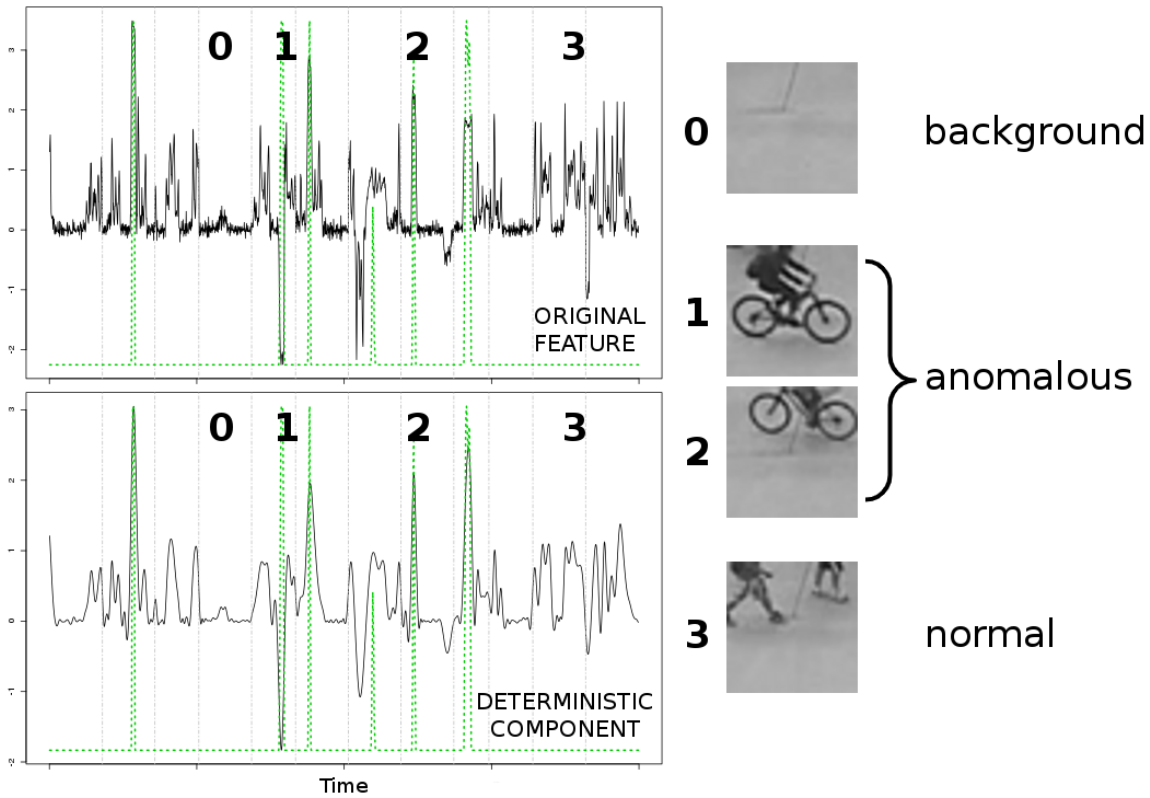


Figure 25 – Temporal filtering for an entire video with examples of normal and anomalous events. Observe it is easier to detect unusual events when using the deterministic component (Adapted from Figure 1 of (PONTI; NAZARE; KITTLER, 2017)).

4.4 Adaptive modeling of motion patterns

In order to make it possible to leverage optical-flow filtering in way that is more suitable for surveillance scenarios (without the need for the entire flow series), we present a novel approach (see Figure 26 in our workflow) that employs a lightweight filtering procedure and an adaptive

motion modeling framework to mitigate the effects of noisy optical-flow estimates and, also, viewing distance variations, leading to improvements in the abnormal event detection for security footage.

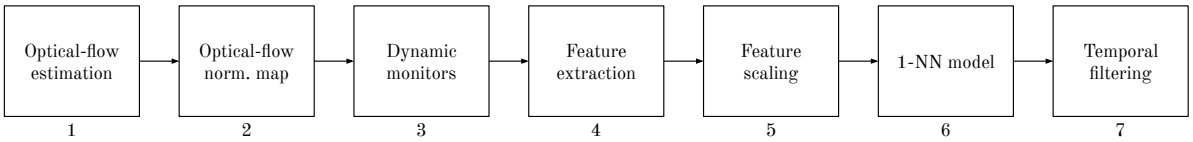


Figure 26 – Workflow of the proposed approach. Only steps 1 to 6 are involved in the training stage, having some type of learning from steps 2 to 6. During prediction, the temporal filtering (step 7) is used to reduce the number of false positives.

Having a video V composed of T instants, each instant $t = 1 \dots T$ is represented by a frame (image) I_t with resolution $M \times N$. The anomaly detection problem can be defined as finding a function $A : \mathbb{R}^D \rightarrow [0,1]$ which estimates an anomaly score based on D input features extracted from a frame region. Additionally, as this is an anomaly detection method, such function is inferred using features strictly extracted from normal data (frames with no abnormal behavior). In the remaining of this section, we introduce our novel motion-based anomaly detection framework, which was designed to be robust to noisy optical-flow estimates and adapt to object size changes. Also, to facilitate reading, Table 2 contains a description of all symbols used throughout this section.

Table 2 – Summary of the symbols used in this section.

Variable	Definition
V	video used by our method as training or test data
t	index of a certain frame in V
T	size (number of frames) in V
$M \times N$	number of rows and columns of each frame in V
I_t	t-th frame of V
A	anomaly detection method/function
(x,y)	coordinates of a pixel (row $\times$ column) within I_t
h_t	optical flow estimate between I_t and $I_{t-\Delta t}$
$\overline{h}$	optical flow map
w	base size of the dynamic window
$\vec{d}$	8-dimensional descriptor of a frame region

4.4.1 Training phase

At the **first step** of our approach, we perform a dense optical-flow estimate among pairs of frames. This means that we generate the flow field h_t that contains a vector estimating the movement of each pixel $h_t(x,y)$ from some previous frame $I_{t-\Delta t}$ to the current one I_t. As in security footage is common to capture footage in which the objects are fairly far from the camera, objects end-up being represented by small number of pixels. Hence, their movement tends

to be rather subtle from one frame to the next, making it hard to have a precise optical-flow estimate. Motivated by this issue, differently from most methods (ADAM *et al.*, 2008; COLQUE; CAETANO; SCHWARTZ, 2015; GEORGE *et al.*, 2018), we study the usage of optical flow estimated from longer frame intervals (i.e. $[t, t-\Delta t]$ for $\Delta t > 1$). The advantages of such simple change are noticeable from the comparison shown in Figure 27.

(a)

(b)

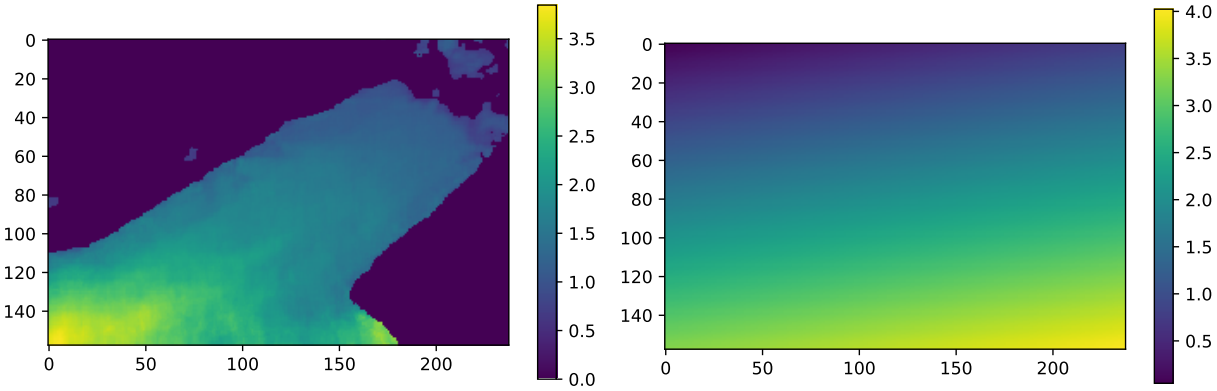
(c)

Figure 27 – Example of optical-flow estimation with different values for Δt. In (a), we have the original frame, (b) represents the flow extracted using $\Delta t = 1$, while (c) shows the results for $\Delta t = 3$. We can notice that the estimate of (c) has a lower degree of noise when compared to (b). For (b) and (c), colors represent the direction of the flow whose intensities indicate magnitudes (the greater the intensity is, the larger is the magnitude).

In scenarios where the size of objects change according to their position within the image (as the one shown in Figure 27), we are not able to use optical flow as a relative velocity measure among pixels due to the fact those methods estimate movement with regards to the number of pixels instead of physical distances. To tackle this issue, in the **second step** of our training pipeline, we use a linear regression model to infer the average velocity (in pixels) of an object for each image coordinate. This model allow us to estimate velocity ratios for each pair of image coordinates and, therefore, make motion patterns invariant to the distance between the camera

and the object.

To build such model, we start by analyzing the training set and counting the number of non-zero optical-flow magnitudes at each pixel location. Next, we take all locations with at least k (we set $k = 100$) non-zero magnitudes and learn a Huber linear regression model (HUBER, 2011) to estimate the average flow magnitude given the pixel coordinates (x, y) (see Figure 28a). This particular type of linear regression model was employed due to its robustness to outliers. Figure 28b allows to confirm that this model is capable of learning that objects at the top of the image move slower (with regards to the number of pixel) when compared to pixels at the bottom. It is important to notice that by using the Huber regression we assume that average velocities change linearly, which may be the case for several surveillance scenarios, but not for all of them.



(a) (b)

Figure 28 – Illustration of the optical flow mapping: (a) refers to the mean magnitudes for pixels with more than 100 non-zero observations; and (b) shows the final map. Please note that each plot has a different range of values.

Once our linear regression model (referred to as *optical-flow map*, for the remaining of this text) is trained, in the **third step** of our training pipeline, we use its results to infer the expected average object size at each image region. To do so, we start by locating the contour line of the optical-flow map in which the values are equal to one. Next, we place windows of size w along this particular contour line, so they have about 50% of horizontal superposition. For our experiments we use values for w that are approximately the expected object size for this contour line, while also testing the effects of using smaller window sizes. After creating monitors for this first contour line, we find two lines (one above and the other bellow the first one) that have roughly 50% of horizontal superposition with the monitors from the initial line and create windows of size $\overline{h}(x,y) \times w$, where $\overline{h}(x,y)$ is the mean optical-flow value for the contour line. This process is repeated until monitors cover the entire image up. By varying the window size according to the expected average pixel displacement and having vertical and horizontal window superposition, our method becomes more robust to changes in scale and avoids having anomalies occurring at the boundary between monitors, respectively. Therefore, our approach improves upon previous studies that – despite using window superposition – either use fixed

monitor sizes (COLQUE; CAETANO; SCHWARTZ, 2015; ADAM *et al.*, 2008), or multiple monitor sizes for the entire image (XU *et al.*, 2015). Figure 43 show a example of monitors for two different contour liner of a normalized optical-flow map.

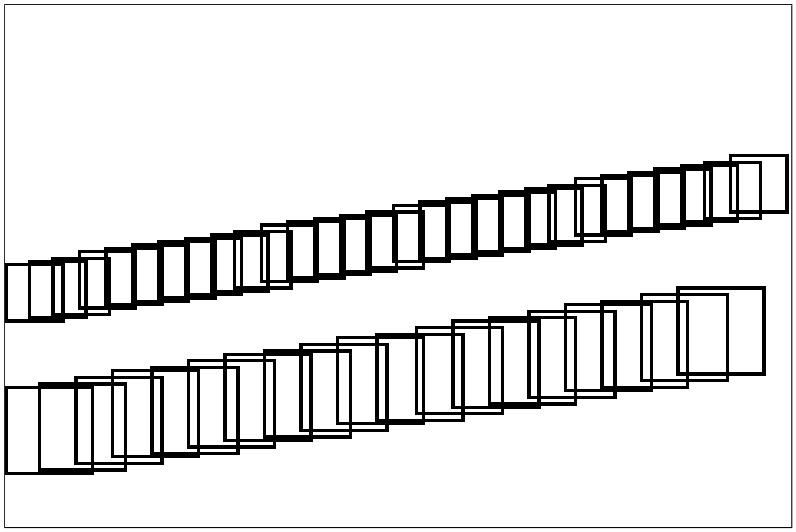


Figure 29 – Example of monitors generated for two different contour lines on the normalized optical-flow map.

At the **fourth step** of our training process, we normalize the optical-flow field of each frame by dividing flow vectors by the expected average flow magnitude (obtained from the *optical-flow map*). Then, for each window region (obtained at step three), we compute the following 8-dimensional descriptor:

$$\vec{d} = \{\mu_m, \sigma_m, \gamma_m, \kappa_m, \mu_o, \sigma_o, \gamma_o, \kappa_o\},$$

where μ, σ, γ, κ represent the mean, standard deviation, kurtosis and skewness, while m and o are related to the orientation and the magnitude of the optical-flow, respectively. This strategy makes our descriptor resilient to changes in image perspectives with regards to optical flow values and object scales. Moreover, we do not need to setup hyper-parameter like the bin sizes used in (COLQUE; CAETANO; SCHWARTZ, 2015) or manually create the adaptive grid of windows like (GEORGE *et al.*, 2018).

Given we employ a nearest neighbors approach to detect anomalies, we first need to normalize our features to ensure that they all have the same scale. To do that, in our **fifth step**, we normalize our data using a $0-1$ or a *z–score* normalization (we compare both approaches in our experiments).

With the normalized features, in our **sixth step**, we train a single 1-NN model to generate anomaly scores for all windows within the frame. This approach has two main advantages: i) it is able to learn normal behaviors from the entire image instead of building local models; and ii) it does not need to process the frame at multiple resolutions to improve resilience to viewing changes.

4.4.2 Prediction phase

After the training process, we obtain: the normalized optical-flow map, the location of all monitors, a feature normalization model and a trained 1-NN model. When generating predictions from new data, we do as follows:

1. Compute the optical flow between the new frame I_t and $t - \Delta$;
2. Normalize the obtained optical flow field using our *optical-flow map*;
3. Locate the region corresponding to each monitor within the frame;
4. Generate our 8-dimensional feature vector from every monitor region;
5. Normalize our feature vector;
6. Compute a first anomaly score for each pixel using our 1-NN model. Given our windows overlap, the same pixel may have more then one score, so the greatest value is taken as the pixel anomaly score;
7. Use a temporal median filter to remove noisy pixel scores.

Lastly, in the **seventh step** of our pipeline (that is only present in the prediction phase), we post-process (filter) our scores in attempt to mitigate the types of noise that are not addressed by increasing the frame gap. For instance, if we have some kind of interference at one of the two frames used to compute our flow, the results would be erroneous. To tackle problems that may emerge from the local nature of the optical flow, we enforce the fact that surveillance footage tends to smoothly change by filtering our anomaly scores.

Based on our pixel-level values, we also compute a frame-level anomaly score as the maximal anomaly score of its pixels. These scores are used in our experimental setup to compare our approach to related studies on both frame and pixel-level anomaly detection benchmarks.

In summary, our goal is to use the first and seventh steps of our framework to investigate if filtering out the noise from optical flow estimates can improve anomaly detection in surveillance situations (as presented in the first part of our hypothesis). Additionally, by employing our dynamic monitors strategy (second and third steps of our framework), we aim at verifying whether adapting to changes in viewing distances (second part of our hypothesis) improves anomaly detection results on security footage.

4.5 Concluding remarks

This chapter described our three anomaly detection methods for video surveillance applications: i) a 3D-convolutional auto-encoder that obtains comparable results to pre-trained CNN models;

ii) an EMD-based optical-flow decomposition that improves anomaly detection by providing more robust flow estimates; and iii) a motion modeling framework that tackles optical flow noise using lightweight filtering techniques and automatically adapting to viewing changes. The next chapter presents our experimental setup and shows an in-depth comparison of our proposed methods against the related work.

CHAPTER

5

EXPERIMENTAL SETUP AND RESULTS

5.1 Opening remarks

In this chapter, we report results on widely used benchmark datasets to analyze the performance of our three anomaly detection methods: i) dataset-specific fully-3D-convolutional auto-encoder; ii) EMD-based optical-flow filtering; and iii) adaptive modeling of motion patterns. Next, we establish some baselines to contrapose all three methods to pre-trained CNNs used as feature extractors, once: i) CNNs have achieved outstanding performance results on several transfer-learning scenarios; and ii) to the best of our knowledge, there are no studies comparing those models on video anomaly detection situations.

Starting with the first method, we confirmed that the dataset-specific fully-3D-convolutional auto-encoder is capable of achieving comparable results to the CNN baselines, while drastically reducing the number of network parameters. Following our study, we show a complete evaluation of our second and third methods to filter out and reduce the impacts of noisy optical-flow estimates on anomaly detection, which confirmed to obtain more robust optical-flow estimates without over-increasing the processing power demanded by the automated surveillance system.

At last, we compare the results of our three methods against the related studies, corroborating their abilities and robustness while outperforming strategies currently composing the state-of-the-art.

5.2 Anomaly detection datasets and evaluation metrics

This section presents some surveillance video datasets that are widely employed to assess the performance of anomaly detection systems. Furthermore, it reviews the main evaluation criteria used to compare models in this application domain.

5.2.1 UCSD anomaly datasets

The two UCSD datasets (MAHADEVAN *et al.*, 2010; LI; MAHADEVAN; VASCONCELOS, 2014), named *Ped1* and *Ped2*, were recorded at the UCSD (University of California, San Diego) campus using stationary cameras. Each dataset contains videos of a different pedestrian walkway on the UCSD campus and all videos within the same dataset are under the same specifications (i.e. camera position, viewing angle, resolution and fps).

Ped1 and *Ped2* are divided into train and test videos and only the test ones contain abnormal events, e.g. carts, skaters, bikers and people walking on the grass or across the walkway. Table 3 presents the main characteristics of *Ped1* and *Ped2* and Figure 30 shows some examples of normal and abnormal frames.

Table 3 – Video specifications for *Ped1* and *Ped2*.

Characteristic	**Dataset**	
	Ped1	**Ped2**
train videos	34	16
test videos	36	12
frame resolution	238×158	360×240
fps	10	10
frames per video	200	$120 - 180$

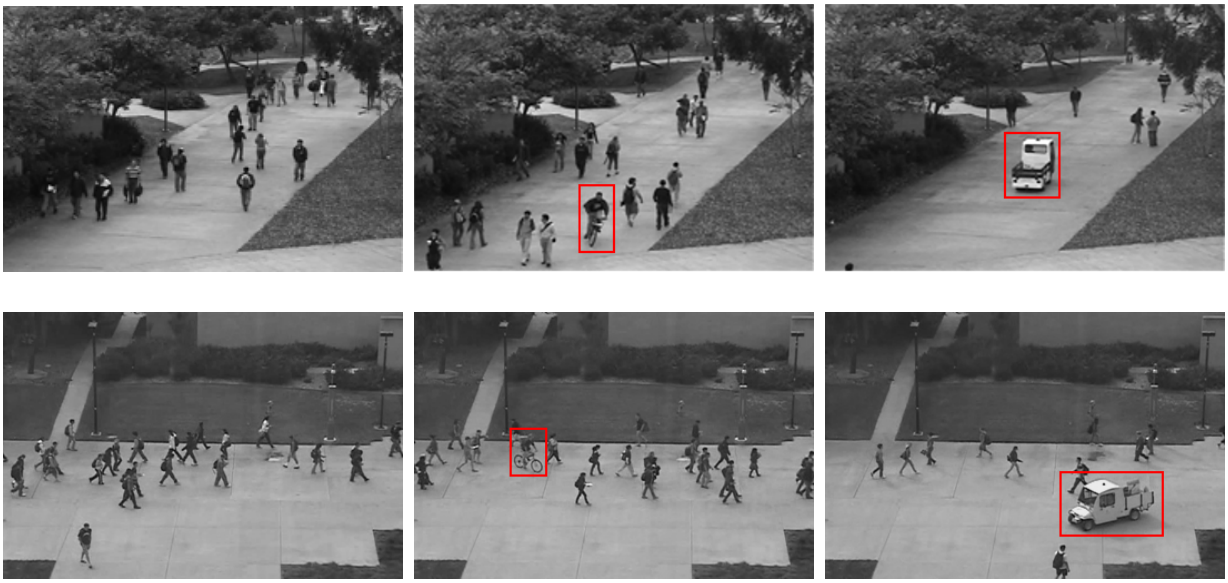


Figure 30 – Frames from *Ped1* (first row) and *Ped2* (second row) in which anomalies were manually highlighted by red boxes.

Among the main advantages, the videos in those datasets were not staged and the majority of test videos were manually annotated at the pixel-level. This means that every test frame has a corresponding binary mask indicating which pixels (if any) contain anomalies. By using this pixel-level annotation, it is possible to evaluate a method based on both its ability to detect abnormal frames and to locate an anomaly within a frame.

On the other hand, those datasets only contain short videos (none of the videos has more than 200 frames), therefore one may question if they do represent real surveillance scenarios. This fact may hamper the performance of algorithms that expect longer videos, like the ones that do not rely on a training phase.

5.2.2 UMN dataset

The UMN dataset[1] is composed of three different scenarios of several people walking (as shown in Figure 31). The anomalies of such dataset are represented by time intervals in which people are running, supposedly to escape from some kind of dangerous situation. Therefore, differently from the local anomalies of the UCSD datasets, the abnormal patterns are global. Regarding its labels, this dataset only has frame-level annotations. Some characteristics of each scenario are presented in Table 4.

Figure 31 – Sample frames from the UMN dataset. Each image shows a frame from one of the three different scenarios.

Table 4 – UMN dataset characteristics.

Characteristic	Scene 1	Scene 2	Scene 3
scene size	1450 frames	4415 frames	2145 frames
resolution	320×240 pixels	320×240 pixels	320×240 pixels
environment	outdoor	indoor	outdoor

5.2.3 Subway exit dataset

This dataset, introduced by Adam *et al.* (2008), contains a single 43-minute video with a 512×384 resolution and 25 fps. In their experiments, the first 5 video minutes were used to perform training and the remaining frames for testing. In this dataset, an unusual event occurs when a person is moving in the wrong direction, which means a person is trying to break in the subway station. In Figure 32, we show some frames extracted from this dataset.

[1] The UMN dataset can be downloaded at: <http://mha.cs.umn.edu/Movies/Crowd-Activity-All.avi>

Figure 32 – Frames extracted from the Subway Exit dataset. The first frame represents a normal event, while the second, which contains a person trying to break in the subway station, represents an unusual event.

Table 5 – Dataset comparison.

Dataset	Event type	Event source	Scene	Label type	Staged
UCSD	local	both	outdoor	frame, pixel	no
UMN	global	motion	both	frame	yes
Avenue	local	both	outdoor	frame, pixel	no
Subway exit	local	motion	indoor	frame	no

5.2.4 Avenue dataset

The Avenue dataset was presented in (LU; SHI; JIA, 2013) and contains a total of 30652 frames – being 15328 for training and 15324 for testing – filmed at the CUHK (The Chinese University of Hong Kong) campus avenue. Regarding the number of videos, it contains 16 for training and 21 for testing. This dataset has challenging scenarios like: i) there is a slight camera shake in one of the test videos; ii) the training set contains few outliers; iii) some of the normal patterns are rather rare in the training set. The evaluation process for this dataset focus on frame-level detection. Figure 33 shows some examples of anomalous frames.

5.2.5 Dataset comparison

Now, in Table 5, we summarize/compare some characteristics of the datasets described in the previous sections with regards to: types of events (local and global), source of anomalous events (appearance and motion), scene environment (indoor and outdoor), types of available labels (event, frame and pixel) and if the videos were staged.

We decided to use the UCSD and UMN datasets to evaluate our approaches because they can help us to measure the results of our models with regards to local and global anomaly detection, respectively. Another important reason for such a choice is that those datasets are widely used in the literature, thus facilitating comparisons of our methods to other studies.

(a) abnormal object (b) strange action

(c) wrong direction

Figure 33 – Frames extracted from the Avenue dataset showing different types of unusual events (highlighted by a red box). These frames with highlighted anomalies were obtained from <http://www.cse.cuhk.edu.hk/~leojia/projects/detectabnormal/dataset.html>.

5.2.6 Evaluation criteria

In the literature, the three most common strategies to assess the performance of a video anomaly detection algorithm are: the event-based, the frame-based and the pixel-based criteria. They are discussed in the next four sections.

Event-level criterion

In a event-based evaluation, the algorithm first detects frames, or frame intervals, supposedly containing unusual events. After that, these predictions are compared against a ground-truth list of unusual events to determine the percentage of correctly identified elements (detection rate) and the number of false alarms. An unusual event is considered to be correctly identified if the algorithm is capable of detecting at least a predetermined percentage of its anomalous frames in the event interval. On the other hand, a false alarm occurs when the algorithm detects a normal interval as anomalous.

Frame-level criterion

In an frame-based evaluation, the anomaly detection algorithm classifies each frame in the test set as normal or abnormal. After that, classifications are compared against a frame-level ground-truth

to determine the number of true-positive and false-positive frames. The *True Positive Ratio* (TPR) and *False Positive Ratio* (FPR) are then computed as follows:

$$TPR = \frac{\text{\# of true-positive frames}}{\text{\# of positive frames}}, \tag{5.1}$$

$$FPR = \frac{\text{\# of false-positive frames}}{\text{\# of negative frames}}. \tag{5.2}$$

Using TPR and FPR values, the *Receiver Operating Characteristic* (ROC) curve is constructed. Finally, this curve is summarized with the *Equal Error Rate* (EER), which is defined as the value of FPR at which $FPR = 1 - TPR$. This is the same as saying that the EER is the value of FPR at which the ROC curve intercepts the line connecting points $(0,1)$ and $(1,0)$, as shown in Figure 34. It is important to observe that the smaller is EER, the better it is.

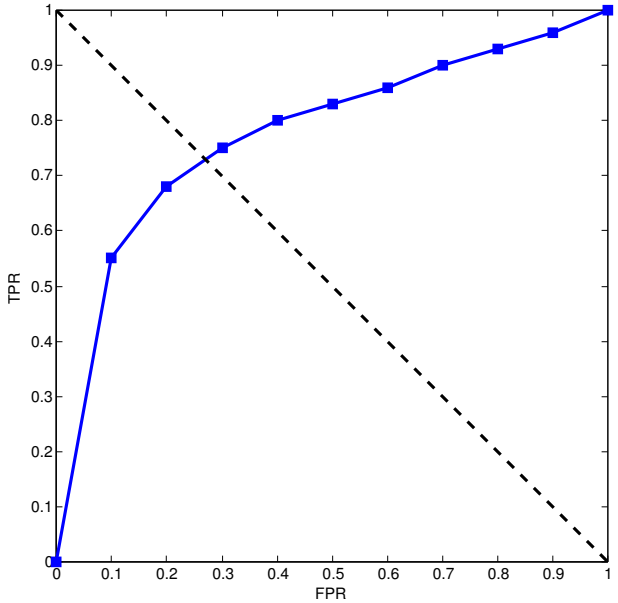


Figure 34 – ROC curve example. The EER value is obtained by taking the FPR value at the point at which the ROC curve (blue curve) intercepts the dashed line.

Pixel-level criterion

In a pixel-level evaluation, instead of detecting anomalous frames, the algorithm classifies each pixel as normal or abnormal. Then, these classifications are compared against a frame-level ground-truth (see Figure 35). If a frame contains an abnormality and the algorithm is capable of detecting at least a given percentage of its anomalous pixels, the frame is considered a true-positive[2]. On the other hand, if the algorithm considers one or more anomalous pixels in a frame with no abnormality, this frame is considered a false-positive. After computing a the number of

[2] Li, Mahadevan and Vasconcelos (2014) use 40% as the minimum pixel detection percentage for the UCSD datasets, which is adopted by most studies. We are going to consider the same threshold to make our results comparable against the ones from the literature.

true-positives and the number of false-positives, the ROC curve is calculated. To summarize this curve, the AUC (Area Under the ROC curve) and the EER are used.

Figure 35 – An image from the Ped1 dataset containing an anomaly and its corresponding pixel-level ground-truth.

5.2.7 Evaluation criteria comparison

Given that the prior evaluation criteria are widely used in the literature, Table 6 summarizes the main advantages and drawbacks of each criterion.

Table 6 – Evaluation criteria comparison.

Criterion	Advantages	Drawbacks
Event-level	It is the simplest among the three, but it can still point out if an algorithm is capable of detecting anomalous events, which is the main focus of surveillance applications.	It does not take into account the percentage of anomalous frames that the algorithm is capable of correctly detecting, neither considers if the algorithm is capable to correctly locating the anomalies within frames.
Frame-level	By computing the number of true-positive and false-positive frames, it better considers how precise are the anomalous event interval detection.	It does not take into consideration the effectiveness of the algorithm in correctly locating the anomalies within frames.
Pixel-level	It shows if the algorithm is capable of locating the anomaly within the frame or if the obtained frame-level detection was a “lucky guess”.	The manual pixel-level annotation process is time consuming.

5.3 Establishing baselines using pre-trained CNNs as feature extractors

For the most part of our approaches/experiments we use an 1-NN model to detect unusual behaviors in security footage. Along this section, we aim at establishing the effectiveness of such approach by using it in combination with features extracted from pre-trained image classification

CNNs, as shown in the experimental setup presented in Figure 36. Besides building baselines for our experimental setup we also seek to compare features generated by the convolutional part of the following state-of-the-art image classification models (trained on ImageNet): VGG-16 (SIMONYAN; ZISSERMAN, 2014), ResNet-50 (HE *et al.*, 2016), Xception (CHOLLET, 2017) and DenseNet-121 (HUANG *et al.*, 2017). To the best of our knowledge, this study is very relevant because – despite having some video anomaly detection that employ features from pre-trained CNNs (RAVANBAKHSH *et al.*, 2018) – no comparison has been made to establish the best model to be used. In Table 7, we present the number of features generated by the selected models, while Section 2.8 contains a detailed description of each architecture.

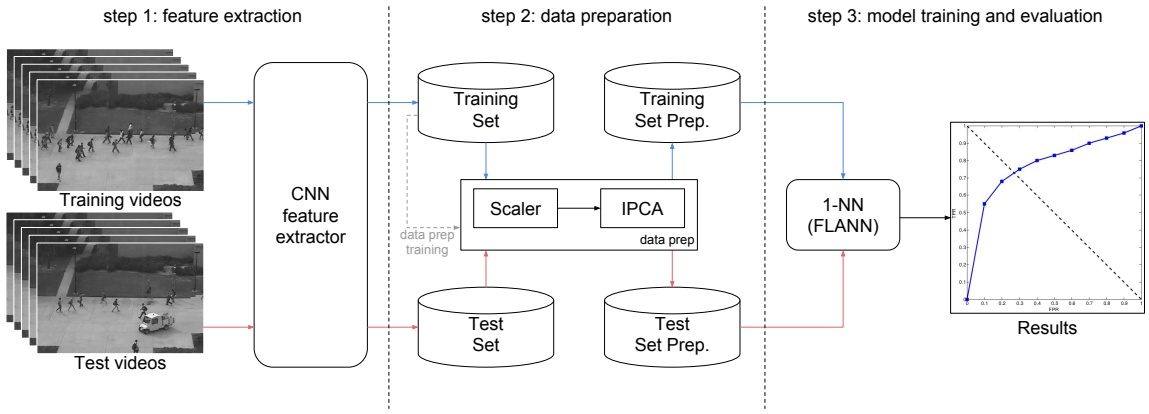


Figure 36 – Experimental setup diagram (Adapted from Figure 1 of (NAZARE; MELLO; PONTI, 2018)).

Table 7 – Number of parameters and output features for each one of the feature extractors (convolutional part of a pre-trained CNN) used in our experiments.

Feature extractor	Number of trainable parameters	Number of features
VGG-16	14,714,688	512
ResNet-50	23,534,592	2048
Xception	20,806,952	2048
DenseNet-121	6,953,856	1024

In the **first step** of our pipeline, we convert each video frame to 384×256 pixels, then we generate overlapping image regions of 32×32 pixels using a stride of 16 pixels. Each image region is transformed into d features by a forward pass through the convolutional part of a CNN pre-trained on the ImageNet dataset (DENG *et al.*, 2009), in which d depends on the network architecture – it is equal to the number of kernels (neurons) at the last convolutional layer of such architecture. For our setup, taking only the convolutional part of CNN model (rather than the entire model) is convenient, mainly because we can provide input images with different sizes other than used in the original training setup (e.g. 224×224 images). Also, is important to notice that we apply the same pre-processing procedure used by the pre-trained model.

In the **second step**, we preprocess the features so they can be used to detect anomalous events. We initiate this process by normalizing our feature vectors according to four different

criteria: z-score, 0-1, L1 and L2 normalization methods. This normalization step helps out the nearest neighbor algorithm, which could have its results hampered if features have a greater range of values. After having the normalized version of our feature vector, we reduce its dimensionality by applying the *Incremental PCA* (IPCA) algorithm (ROSS *et al.*, 2008) (in our experiments, we compare results using 50 and 100 dimensions). The two most important reasons for using IPCA are: i) the reduction in the number of dimensions (features) so we can speedup the anomaly detection process; and ii) as IPCA incrementally compute its components, we can deal with large datasets without demanding a huge amount of main memory.

Finally, in the **third step** of our setup, we train a single approximate 1-NN model (using the algorithm from (MUJA; LOWE, 2014)) on each dataset obtained after the step two (which has 50 or 100 features), thus the same nearest neighbor model is responsible for generating an anomaly score for every image region. This score is equal to the Euclidean distance from the new sample (feature vector extracted from a particular region of a frame) to its closest sample on the training set. Then, we compute the anomaly score for a frame simply by taking the maximum score among its regions. Using such frame scores, we can compute the *Area Under the ROC Curve* (AUC) and EER on a frame-level classification.

Based on the aforementioned experimental protocol, we learn 1-NN models using image region descriptions generated by the four pre-trained CNNs. On each of our tests, prior to training our anomaly detection models, we normalize descriptors and apply IPCA to generate lower dimensional representations of features. Seeking to achieve better results, we try several configurations for both the normalization technique and the number of IPCA dimensions. The frame-level results for both UCSD datasets are shown in Table 8.

Analyzing the results presented in Table 8, we can notice that different normalization methods have a significant impact on the anomaly detection performance. Furthermore, such influence varies according to the pre-trained CNN model. For instance, the z-score normalization seems to be more suitable for ResNet-50 and Xception features, while the $0-1$ normalization generated better results for DenseNet-121 features. Regarding the number of IPCA dimensions used to learn our 1-NN model, its is possible to notice that using 100 instead of 50 dimensions tends to improve results. Nonetheless, those improvements were fairly incremental. Thus, in practical applications, it is important to consider whether the performance gain of the anomaly detection offsets the increase in computational cost of running the nearest neighbor algorithm on a larger number of dimensions.

5.4 Anomaly detection with 3D-convolutions

In order to understand the effectiveness of using the simple 3D-auto-encoder presented in Section 4.2, we compare its results against the ones obtained from features generated by a pre-trained C3D model (TRAN *et al.*, 2015). Next, we compare its results against our auto-encoder.

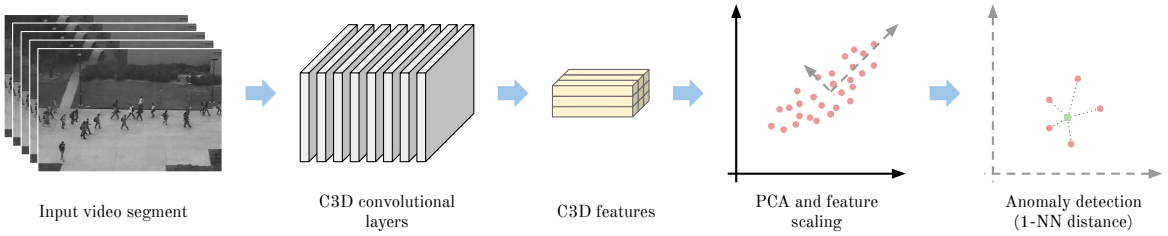
Table 8 – Frame level detection results on *Ped1* and *Ped2* datasets using several pre-trained CNNs as feature extractors, various feature normalization techniques. All results employed the approximate nearest neighbor algorithm. Please keep in mind that the smaller the EER is, the better it is; and the greater the AUC is, the better it is.

Feature extractor	**IPCA dimensions**	**Normalization method**	**Dataset**			
			Ped1		**Ped2**	
			AUC	**EER**	**AUC**	**EER**
VGG-16	50	0-1	63.98%	40.85%	63.81%	42.90%
		z-score	63.35%	42.02%	64.98%	40.05%
		L1	59.01%	43.28%	63.73%	39.95%
		L2	59.16%	43.15%	63.37%	40.13%
	100	0-1	**64.06%**	41.02%	63.42%	40.29%
		z-score	63.78%	41.13%	64.97%	38.53%
		L1	63.62%	**40.40%**	65.49%	38.40%
		L2	61.02%	42.72%	62.84%	40.70%
ResNet-50	50	0-1	59.10%	44.94%	62.05%	42.27%
		z-score	60.95%	43.17%	79.59%	28.88%
		L1	55.26%	45.73%	45.59%	51.93%
		L2	63.60%	41.25%	71.05%	34.53%
	100	0-1	59.65%	44.66%	69.02%	37.08%
		z-score	61.98%	42.21%	83.90%	23.33%
		L1	55.48%	45.93%	47.49%	51.46%
		L2	62.67%	42.19%	70.08%	34.89%
Xception	50	0-1	59.95%	44.18%	86.61%	21.82%
		z-score	59.16%	44.23%	87.33%	21.66%
		L1	51.59%	48.91%	63.72%	41.32%
		L2	60.02%	44.21%	80.78%	25.39%
	100	0-1	60.68%	43.75%	87.94%	**19.55%**
		z-score	59.61%	43.59%	**88.93%**	20.02%
		L1	51.99%	48.71%	65.21%	40.35%
		L2	58.83%	43.56%	82.03%	24.58%
DenseNet-121	50	0-1	63.65%	41.02%	82.91%	23.58%
		z-score	62.04%	41.83%	72.57%	34.81%
		L1	63.00%	40.60%	73.73%	31.98%
		L2	63.15%	41.34%	72.79%	32.87%
	100	0-1	63.16%	41.88%	84.61%	23.06%
		z-score	62.57%	41.44%	83.07%	24.31%
		L1	62.73%	41.40%	78.09%	28.16%
		L2	62.71%	42.13%	78.05%	27.07%

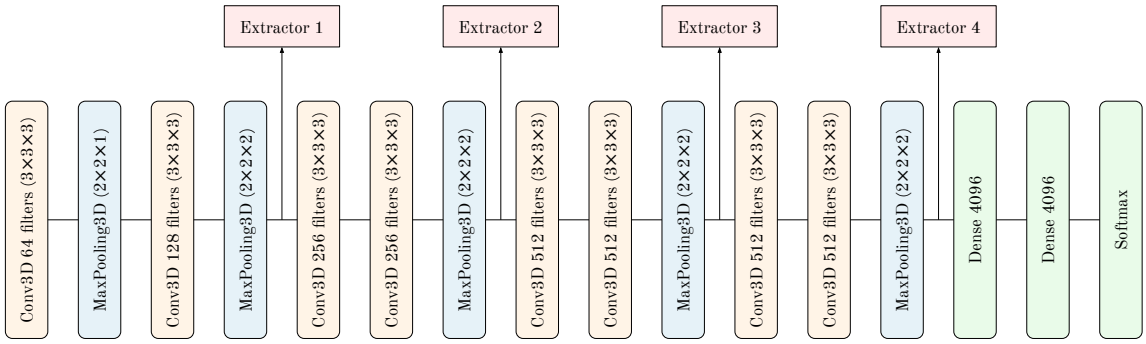
Furthermore, we compare the two approaches in terms of how their number of parameters affect their computational efficiencies.

5.4.1 Spatio-temporal features from C3D as descriptors for anomaly detection

To carry out this comparison, we start by employing the experimental setup shown in Figure 37 to try to optimize the results of C3D features. From a video anomaly detection perspective (see Figure 37a), our setup was based on the one presented in (NAZARE; MELLO; PONTI, 2018). With regards to the analysis of the features extracted from different convolutional layers of the C3D network, our study is strongly motivated by the findings of Yosinski *et al.* (2014).



(a)



(b)

Figure 37 – Experimental setup diagram for C3D features.

In the **first step** of our setup, we perform a forward pass of video segments (16 consecutive frames) through convolutional layers of a pre-trained C3D model. This generates an output tensor that describes $32 \times 32 \times 16$ regions of the video segment. Later, such description is employed to detect anomalies in video regions/frames.

Considering that we extract features from four different layers – in order to obtain a descriptor for $32 \times 32 \times 16$ video regions – the descriptor coming from each layer should be interpreted in a particular way. For instance, when we carry on with this process using the entire convolutional part of the C3D network (related to Extractor 4 in Figure 37b), we obtain an $M \times N \times 512$ tensor in which there are 512-dimensional feature vectors describing $M \times N$ video regions of $32 \times 32 \times 16$ pixels each (see Table 9). Nevertheless, if we are going to extract features using Extractor 3, each region is going to be described by a $2 \times 2 \times 2 \times 512$ tensor. It is possible to notice that, specially for Extractors 1 and 2, the number of features generated is rather great, making anomaly detection very slow. To tackle this problem, we perform a 3D pooling on the

spatial and temporal dimensions in order to reduce them out to one. This approach drastically reduces the number of features generated by the extractors, as seen in Table 9.

Table 9 – Number of features generated by each extractor when describing a $32 \times 32 \times 16$ video region (before and after pooling).

Extractor	Number of raw features	Number of features after pooling
1	$8 \times 8 \times 8 \times 128 = 65536$	128
2	$4 \times 4 \times 4 \times 256 = 16384$	256
3	$2 \times 2 \times 2 \times 512 = 4096$	512
4	$1 \times 1 \times 1 \times 512 = 512$	512

Before doing the forward pass, the pre-trained C3D model subtracts the average pixel value (calculated on the original training set) of its input video segment. To accommodate such pre-processing step – even when we use video segments with a frame resolution that is different from the one of the original C3D training dataset – we re-scale the pre-processing tensor to have the same spacial resolution of our input video segments and subtract these values from the input video segment.

Next, in the **second step** of our setup, we reduce the number of features generated by the extractor used in first step. In order to do that, we begin by applying a standard scaler on the raw features, then we reduce the number of features using an Incremental PCA (IPCA) model (ROSS *et al.*, 2008) and, lastly, we rescale the features again using another standard scaler. Reducing the number of features and having them following a mean equals to zero and a standard deviation equals to one is important, because, in the third and last step, we use an approximate nearest neighbor algorithm to detect anomalies (MUJA; LOWE, 2014). Such algorithm requires features in the same scale to provide the same importance to them, besides being faster when operating on less features.

Finally, in the **third step** of our setup, we detect local anomalies (video regions that are unusual) by computing the Euclidean distance of its descriptor – after having its dimensionality reduced in step two – to the most similar sample in the training data. In this case, the greater is the distance, the more likely is the chance of having an anomaly on that particular video region.

Using this setup, we compare the features generated by four different layers of a pre-trained C3D model with regards to their ability of enabling the detection of anomalies and their processing demands. To achieve that, we experiment with some variations of hyper-parameters (listed in Table 10) that were empirically chosen aiming at obtaining good anomaly detection results, while keeping the computational cost fairly low. Then, to evaluate the detection performance, for each experiment, we measure the AUC and EER for the frame-level anomaly detection on the *Ped1* and *Ped2* datasets.

We experimented with the values presented in Table 10 and collected the results shown in Table 11. By analyzing these results, one observes that the features extracted from Extractor 2 usually provided the best results, what is interesting for two reasons: i) it may indicate that

Table 10 – Values tested for each hyper-parameter.

Parameter	**Values**
Pooling type	average, max
Frame resolution	192×128, 384×256
IPCA output dimensions	8, 16, 32, 64, 128

Table 11 – Results obtained from a frame-level detection on the Ped1 and Ped2 datasets. These results were obtained using several different hyper-parameter settings (feature extractor, pooling method, number of IPCA features and frame resolution). Please notice that the greater the AUC is, the better it is; and the lower the EER is, the better it is.

Extractor	**Pooling ethod**	**IPCA dims.**	**Ped1**				**Ped2**			
			192×168		**384×256**		**192×168**		**384×256**	
			AUC	**EER**	**AUC**	**EER**	**AUC**	**EER**	**AUC**	**EER**
1	avg	8	**61.39%**	**42.21%**	54.88%	48.06%	79.53%	28.29%	88.17%	20.14%
		16	56.33%	44.08%	53.96%	47.03%	74.80%	31.80%	90.03%	17.18%
		32	56.83%	45.41%	55.03%	45.95%	70.20%	33.77%	88.96%	19.47%
		64	54.21%	47.58%	54.80%	45.28%	82.96%	25.57%	92.65%	15.41%
		128	49.96%	51.14%	49.70%	52.03%	84.11%	21.48%	94.63%	11.96%
	max	8	46.66%	52.88%	46.61%	51.99%	54.07%	42.95%	49.38%	50.43%
		16	43.04%	54.73%	47.61%	51.91%	51.27%	45.62%	50.48%	48.95%
		32	44.76%	54.23%	48.43%	52.22%	53.24%	46.52%	52.70%	49.51%
		64	44.41%	53.69%	48.43%	51.02%	59.25%	42.90%	57.66%	45.90%
		128	45.32%	52.34%	48.95%	52.60%	56.41%	43.82%	57.69%	44.35%
2	avg	8	57.94%	44.46%	**58.71%**	44.44%	68.58%	37.05%	77.18%	30.48%
		16	57.95%	43.54%	53.58%	47.56%	68.10%	36.07%	83.45%	25.53%
		32	59.44%	43.52%	55.00%	45.77%	70.52%	35.96%	90.77%	15.99%
		64	58.65%	43.68%	56.60%	46.33%	**89.26%**	20.22%	94.61%	11.80%
		128	54.33%	46.05%	57.02%	**44.42%**	88.44%	**19.40%**	**96.12%**	**11.48%**
	max	8	49.46%	50.47%	49.95%	49.41%	57.57%	45.08%	67.03%	37.40%
		16	48.44%	51.75%	49.72%	49.84%	53.27%	45.87%	79.46%	26.70%
		32	48.24%	50.87%	49.94%	49.28%	58.03%	41.84%	82.26%	24.52%
		64	48.10%	50.47%	49.86%	51.06%	77.71%	29.74%	85.66%	22.87%
		128	48.69%	50.51%	51.45%	48.91%	84.70%	23.28%	88.83%	19.07%
3	avg	8	53.35%	47.10%	50.45%	49.34%	61.18%	44.79%	75.97%	30.99%
		16	54.10%	46.34%	48.45%	51.16%	65.81%	40.80%	77.57%	30.93%
		32	52.09%	48.41%	50.37%	50.06%	80.25%	26.04%	85.00%	22.61%
		64	55.57%	46.89%	51.65%	48.90%	83.10%	23.27%	87.85%	18.36%
		128	55.68%	46.40%	51.94%	48.84%	85.04%	22.62%	88.22%	16.72%
	max	8	49.85%	49.47%	46.33%	52.78%	61.74%	43.23%	74.24%	29.02%
		16	49.16%	50.01%	40.64%	56.32%	61.29%	46.56%	72.60%	33.11%
		32	45.92%	53.10%	44.81%	53.72%	70.57%	32.56%	79.89%	25.05%
		64	49.83%	51.08%	48.12%	51.28%	76.87%	30.47%	86.95%	18.76%
		128	51.11%	48.90%	48.52%	50.36%	82.35%	28.20%	87.25%	17.18%
4	-	8	48.98%	50.78%	50.49%	50.25%	76.47%	30.07%	79.17%	24.32%
		16	49.85%	49.87%	46.04%	53.53%	77.41%	27.54%	84.98%	18.84%
		32	48.02%	50.95%	48.86%	51.92%	81.76%	23.20%	85.69%	17.43%
		64	48.53%	51.03%	46.55%	53.61%	72.93%	30.11%	87.93%	16.07%
		128	48.38%	51.40%	48.48%	51.66%	70.67%	32.39%	87.02%	16.11%

simpler (and more generic) features such as the ones of Extractor 2 can perform better, given the original dataset in which the C3D model was trained on and the target (USCD) datasets are very different; and ii) it is much faster to compute its descriptors, given Extractor 2 only uses some of the C3D layers.

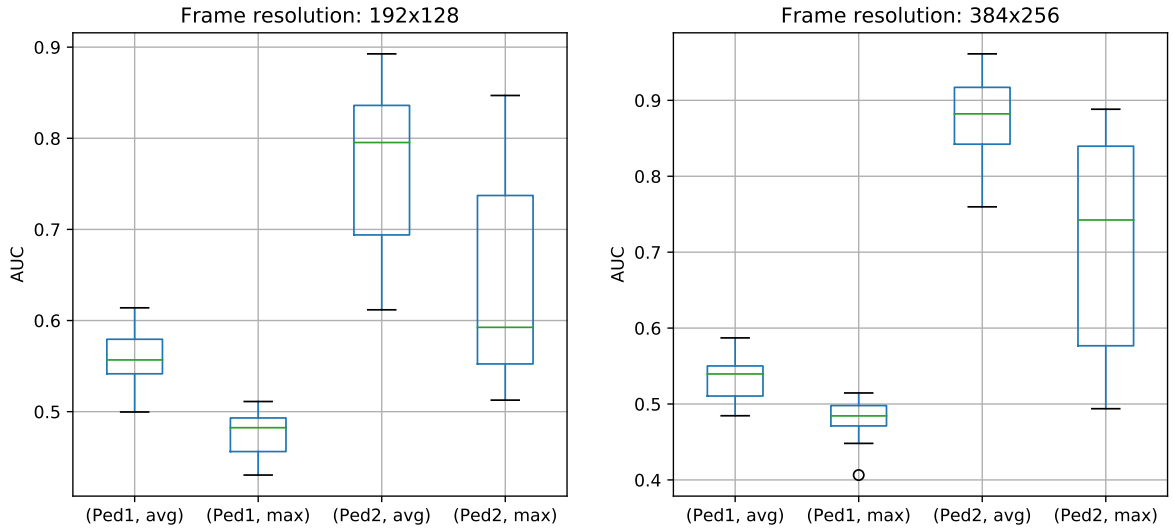


Figure 38 – Comparison of pooling methods. Results confirm average pooling at the last layer of the feature extractor, instead of max pooling, improves anomaly detection results.

Another interesting observable fact from Table 11 is that – for this particular method – the average pooling tends to lead to better anomaly detection results. In Figure 38, we present some boxplots that help to make evident such phenomenon as the AUCs are greater than for max pooling. Additionally, one notices that a greater number of IPCA dimensions (64 and 128) lead to better results.

When comparing the C3D results against the pre-trained image classification models, we observe that the features from the last convolutional layer of the C3D model are comparable to the ones of the best 2D model (Xception). Nonetheless, by using simpler features (from layers that are closer to the input video segment), we were able to surpass the Xception results for the *Ped2* dataset. Similarly to the image classification CNN, the C3D model obtained poor anomaly detection results when dealing with the *Ped1* dataset.

Our auto-encoder architecture was tested with and without background subtraction applied to the input frames, providing the results shown on Table 12. Regarding *Ped1*, our approach was capable of achieving better results when compared to all models learned with features from pre-trained models. Nevertheless, its results are still not comparable to the state-of-the-art. Considering *Ped2*, our results were comparable to the best ones presented in the literature. It is important to notice that our auto-encoder was capable of obtaining such a performance by using far less parameters than CNN architectures (as seen in Tables 7 and 13).

By analyzing these results, we believe that, despite being capable of modeling changes in spatio-temporal behavior, the studied methods cannot tackle variations in object sizes and subtle movements occurring in the *Ped1* dataset. Consequently, their results are hampered, despite their ability to generate good spatio-temporal feature representations (as seen in *Ped2* results). To mitigate such issue, we propose a more adaptive approach to model such scenes in the next section.

Table 12 – Frame-level anomaly detection results for the proposed auto-encoder architectures on the UCSD datasets.

Method	Ped1		Ped2	
	AUC	EER	AUC	EER
3D-auto-encoder (with bg subtraction)	64.8%	39.7%	91.0%	15.0%
3D-auto-encoder (without bg subtraction)	67.7%	38.7%	90.0%	19.0%

Table 13 – Number of trainable parameters for each 3D-convolutional models (C3D feature extractors and auto-encoder).

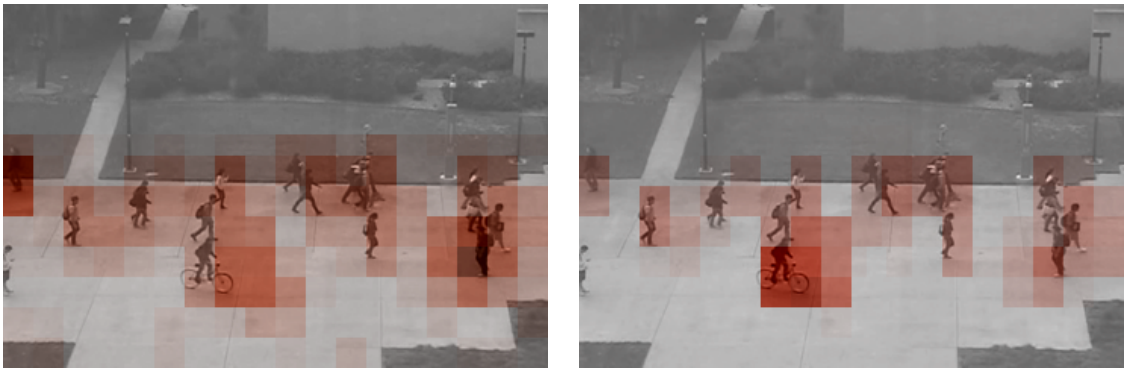
Model	Number of trainable parameters
AE-3D	9,364
C3D (Extractor 1)	226,560
C3D (Extractor 2)	2,881,280
C3D (Extractor 3)	13,499,136
C3D (Extractor 4)	27,655,936

5.5 Adaptive modeling of motion patterns

5.5.1 EMD as an optical-flow filtering method

In this section, we show how reducing the noise of optical-flow estimates can benefit the detection of unusual events. We first tackle this problem by using EMD to decompose the local optical flow – average flow on a frame window – into a deterministic component (that we consider to be a filtered optical-flow estimate) and a stochastic component (that we consider to be noise). Such local average is considered to be an anomaly score, which means that regions with objects moving faster are considered to be anomalous. By comparing the raw average flow against the filtered one, we are visually able to perceive that the filtered flow produces better anomaly scores. To illustrate the efficacy of our filtering technique, we show examples of comparisons between applying a threshold on filtered and original features (optical-flow magnitude) in Figures 39 and 40.

The aforementioned comparisons suggest that this type of filtering is beneficial. However, in attempt to quantify the performance gains of such technique, we compare it against the original features when detecting anomalies at the frame level on the *Ped2* dataset, using optical-flow magnitude as descriptor. When doing so, we observed that EER dropped from **40**.**2**% to **31**.**7**%, simply by doing the proposed filtering. This result indicates that our technique can be advantageous when working with noisy video descriptors (e.g. the ones based on optical flow). Furthermore, this result signals that leveraging the fact that surveillance videos are monotonous to enforce smoothness on the descriptions of temporally-close frames can improve detection rates. Despite having such gains, our method requires the entire video sequence to perform the decomposition, therefore its use in an online fashion can be compromised. Regardless of this



(a) (b)

Figure 39 – Optical-flow magnitude filtering. In (a) we have the original magnitude values, while in (b) we have the deterministic component obtained by the decomposition process described in this section (Adapted from (PONTI; NAZARE; KITTLER, 2017)).

(a) (b)

Figure 40 – Optical-flow magnitude thresholding example in a frame. In (a) we have the original features and in (b) the filtered ones. Blocks in red correspond to detected anomalies, while blocks in blue are borderline samples, within a 5% margin of the threshold (Adapted from (PONTI; NAZARE; KITTLER, 2017)).

drawback, the idea of temporally filtering features is a major contribution that can be considered to improve even online detection, as discussed in the next section.

5.5.2 Adaptive motion modeling framework

Prior to comparing our proposed methods against related work, we study the sensibility of our adaptive motion modeling framework (presented in Section 4.3.1) to changes to its hyper-parameters, listed in Table 14.

We start our hyper-parameter evaluation by inspecting the effectiveness of using the proposed dynamic monitors rather than fixed-size ones. We compare the frame-level of these two procedures on the *Ped1* dataset, due to its variations in object-camera distances. In order to do that, we compared the best results of each approach when varying all the remaining parameters according to the values presented in Table 14. By doing so, we obtained an AUC of

Table 14 – Search space of our hyper-parameters.

Parameter	Values
monitor sizes	fixed, dynamic
pixel scores	raw, filtered
optical-flow method	Farneback, Brox, Dual TV L1
optical-flow frame delta	1, 3, 5
dynamic monitor base size	16, 20
feature pre-processing	0-1 scaling, z-score

87.7% and an EER of 18.7% when employing dynamic monitors, while the results of fixed-size monitors achieved an AUC of 84.8% and an EER of 21.8%. Running the same experiments on *Ped2*, the two monitor types obtained similar performances. Thus, our results confirm that dynamic monitors improve performances when object-camera distances vary, while maintaining performance in situations where this phenomenon does not occur. For that reason, the dynamic monitors approach is employed on the remaining experiments.

Next, based on the findings of Adam *et al.* (2008), Ponti, Nazare and Kittler (2017), we turn our attention to understand the effects of considering filtered anomaly scores (obtained as the result of a temporal median filter) rather than the raw values. We employed the median filter due to both its anomaly detection improvements (which are going to be demonstrated by our experiments) and its low computational complexity. We start our investigation by comparing AUC and EER results at the frame level detection on both UCSD datasets. Our experiments consider all variations of the remaining hyper-parameters (presented in Table 14). The results of such experiments are summarized in Figure 41, from which one notices that filtered anomaly scores improve the detection performance in both datasets.

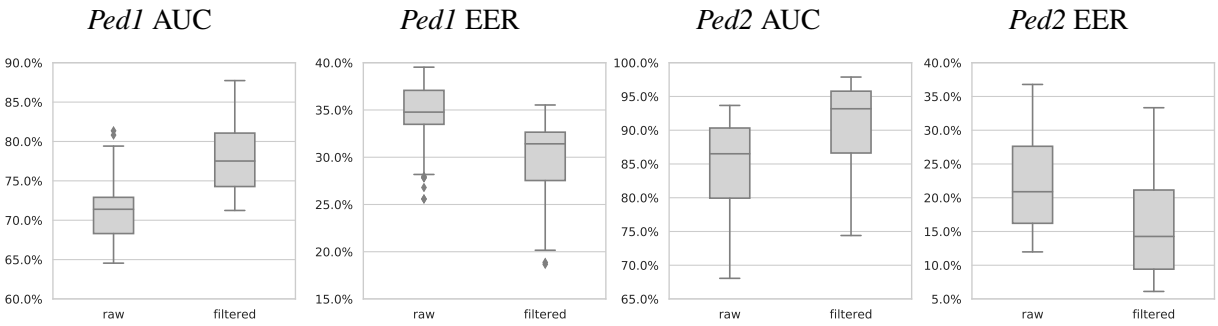


Figure 41 – Frame-level results in terms of AUC and EER with/without pixel score filtering. The two left-most images show the results for *Ped1*, while the two right-most ones show for *Ped2*. Please notice that each box plot has a different y-axis range.

To further understand the benefits of this kind of filtering, we take the best hyper-parameter combination for each dataset and compare their pixel-level detection results when using the two anomaly scores (raw and filtered). The results of such experiments are presented in Figure 42, confirming that filtered scores lead to a substantial improvement in anomaly detection

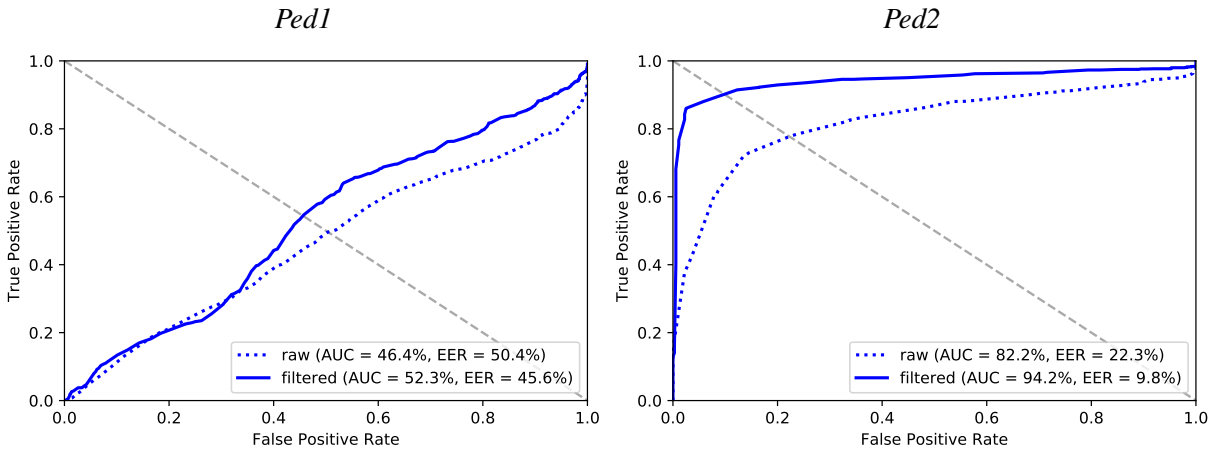


Figure 42 – Pixel-level ROC curves for the UCSD datasets. Please note the random performance with regards to pixel-level is not 50%, but rather close to zero.

performance. Specifically, our *Ped1* AUC results increased 5.9 pp[3] and our EER results improved 4.8, while for *Ped2* we obtained a 12 pp gain with regards to AUC and 12.5 improvement considering EER. Referring to these results, please keep in mind that the smaller the EER is, the better it is; while, the greater the AUC is, the better it is.

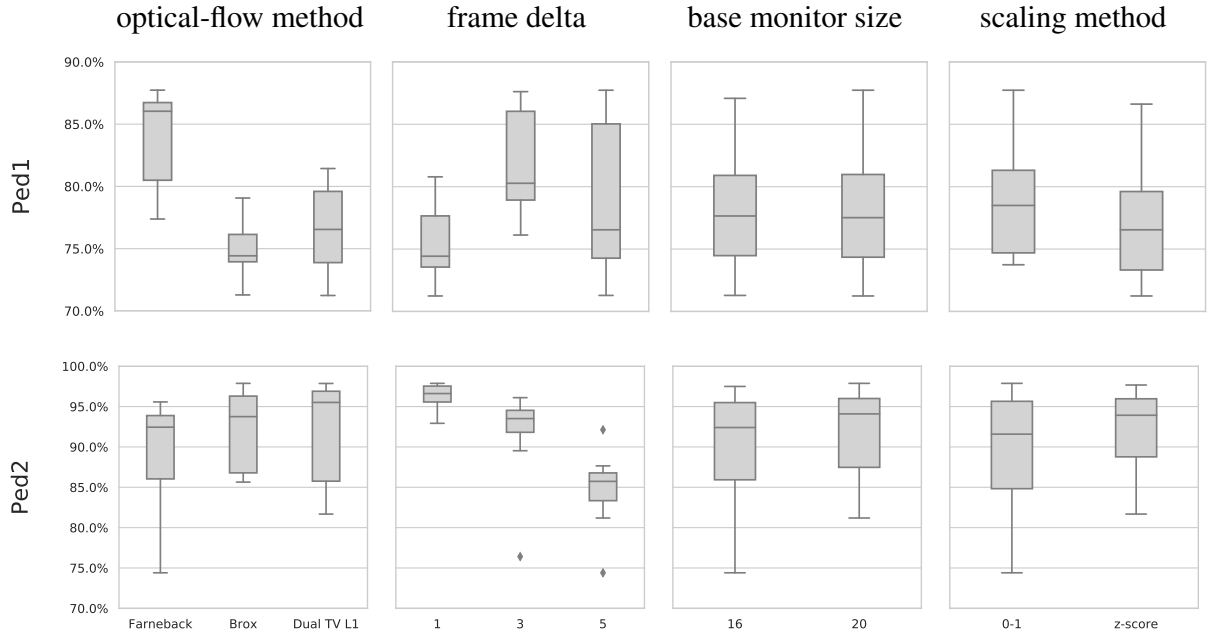


Figure 43 – Frame-level AUC results for a hyper-parameter sensitivity analysis. Rows correspond to results for *Ped1* and *Ped2*, respectively. Columns illustrate results while varying: the optical-flow estimation method, the frame delta parameter used by the optical-flow method, the base monitor size, and the feature scaling approach.

Supported by the prior experimental results, we study the remaining hyper-parameters by only considering their impacts on results generated by filtered pixel anomaly scores. Additionally, such comparison is carried out only using AUC results from frame-level experiments, as they

[3] pp – Percentage point.

allow us to have sufficient evidences to carry on with our comparison and a simpler experimental setup.

The results of this part of our study can be seen in Figure 43 from which we notice the following for each hyper-parameter:

Optical-flow estimation method: despite the fact that we do not focus on tuning the optical-flow algorithms, we compare the three most commonly used in our experiments according to the setup suggested by their authors (FARNEBÄCK, 2003; BROX *et al.*, 2004; ZACH; POCK; BISCHOF, 2007). The results depicted by the first column of Figure 43 indicate that the choice of such method can lead to substantial changes in anomaly detection results and that greater impacts seem to occur in footage with varying object-to-camera distance, like the ones from Ped1. First, regarding the results by Brox's method (BROX *et al.*, 2004), one possible reason for the lower performances on *Ped1* and on *Ped2* is the fact that such an approach tends to be hampered by changes in brightness, as the ones from *Ped1*. Second, while the Dual TV method (ZACH; POCK; BISCHOF, 2007) aims at increasing robustness to illumination changes, occlusions and noise, its parameters tend to be scene-dependent what may affect its results on *Ped1*. Last, Farneback's method (FARNEBÄCK, 2003) confirmed to be less sensitive to parameter tuning while obtaining the best overall performance, despite having worse AUC results on *Ped2*. In summary, our results suggest that Farneback's method can be used to obtain good results without deeply investigating this hyper-parameter, but scene-dependent studies on this matter can lead to additional performance gains.

Frame delta: we evaluated some gap sizes (specifically 1, 3 and 5) in order to determine if greater frame gaps could improve optical-flow estimates and, consequently, the anomaly detection performance. Our results are presented at the second column of Figure 43, confirming that an increasing in the frame gap helps improving results for *Ped1*, while reducing the performance for *Ped2*. We believe that the *Ped1* results were improved due to greater gaps (e.g. 3 and 5) make movements more noticeable and easier to estimate by using optical-flow techniques. This is particularly beneficial for this dataset, as it has a lower video resolution in which objects are represented by a reduced number of pixels, leading to very a subtle pixel movement that is difficult to estimate/notice. On the other hand, given *Ped2* has already more noticeable movements due to its greater frame resolution/object size, greater frame delta values can make the two frames used to estimate the optical flow so different that such flow becomes imprecise, negatively impacting on the anomaly detection performance. Therefore, as a rule of thumb, we recommend the use of greater frame gaps only on datasets with very subtle changes due to: low resolution, large object-to-camera distance, small objects.

Monitor size: as discussed in Section 4.4.1, we aim at setting this parameter as the average

object size on the scene, so a window can contain an entire object. Nonetheless, inferring this value for each scene can be time consuming, leading to the investigation of monitor sizes equal to 16 and 20 pixels. As cen be seen in Figure 44, such values are reasonable for *Ped1* and too small for *Ped2*, what helps us to understand possible drawbacks of having monitors that are too small when compared to the average object size. As it can be noticed from the third column of Figure 43, even with lower than ideal monitor sizes our method performs well on *Ped2*, confirming the robustness of our framework to variations on this hyper-parameter.

Feature scaling: in order to illustrate the robustness of our pipeline, we compared two types of feature scaling: the $0-1$ scaling and the z-score normalization. As we use an 1-NN approach to detect anomalous behavior, some type of normalization is mandatory so our method consider all feature equally (as explained in Section 4.4.1). As presented at the last column of Figure 43, our method achieve similar results with both normalization approaches.

Figure 44 – Illustrating the base monitor size for the UCSD datasets: *Ped1* (top), *Ped2* (bottom). The blue and red bounding boxes represent monitor sizes equal to 16 and 20 pixels, respectively.

5.5.3 Performance comparison

After having a better understanding of our proposed methods, we compare our results against the related work. In Table 15, we organize methods according to their type of visual cues (appearance, motion or both) and list their frame and pixel-level results on the UCSD datasets. As we compare our methods against the results reported in the literature, some values are missing. Moreover, we suppress the *Ped1* pixel-level results from papers that consider the extended label version[4].

Regarding the *Ped1* results, our method achieved comparable results to the ones of methods that use both appearance and motion cues. Still, some of such methods obtained considerably

[4] Originally, pixel-level annotations were created for 10 test videos, which was later extended to all 36 videos by Antić and Ommer (2011). However, to the best of our knowledge, those labels are no longer available online.

better results (i.e. more precisely Adversarial discriminator (RAVANBAKHSH *et al.*, 2017) and Plug-and-play CNN (RAVANBAKHSH *et al.*, 2018)). When comparing against motion-based methods, we obtained comparable results to the ones of Conv-WTA (TRAN; HOGG, 2017) and LSA (SALIGRAMA; CHEN, 2012), while being able to surpass the performance of AE-HOFM (GEORGE *et al.*, 2018) (which is the other method that applies a non-uniform grid of windows to mitigate the effect changes in viewing distance). The usage of an auto-encoder to learn dataset-specific motion features and the additional spatial-temporal neighbor analysis are, respectively, possible explanations for the superior performance of Conv-WTA and LSA on *Ped1*.

Concerning *Ped2*, which has more motion-related anomalies when compared to *Ped1*, our method was capable of obtaining the best results on both frame and pixel-level anomaly detection setups. We think that one of the main factors that contributed to our good performance is the dynamic window approach, which allows for the usage of a single anomaly, while most techniques employ a different model for each region. This approach is particularly beneficial when dealing with the *Ped2* dataset, because its videos have a poor representation of the different types of normal behaviors at the bottom part of the training frames, which hampers the performance of local anomaly detection at these regions.

Table 15 – Frame-level anomaly detection comparison on the UCSD datasets: *Ped1* and *Ped2*. Lower EER values are better, while greater AUC values are better.

Method	Type	Ped1				Ped2			
		Frame-level		Fixel-level		Frame-level		Pixel-level	
		AUC	EER	AUC	EER	AUC	EER	AUC	EER
CNN-2D features (NAZARE; MELLO; PONTI, 2018)	appearance	64.1%	40.4%	-	-	88.9%	19.6%	-	-
Sparse combination (LU; SHI; JIA, 2013)	both	91.8%	15.0%	63.8%	-	-	-	-	-
LNND (HU *et al.*, 2014)	both	-	27.9%	-	-	-	23.7%	-	-
MDT (LI; MAHADEVAN; VASCONCELOS, 2014)	both	81.8%	25.0%	44.0%	55.0%	82.9%	25.0%	-	55.0%
AMDN (XU *et al.*, 2015)	both	92.1%	16.0%	67.2%	40.1%	90.8%	17.0%	-	-
Composition pattern (LI NANNAN, 2015)	both	87.2%	21.0%	-	-	89.2%	20.0%	-	-
Adversarial discriminator (RAVANBAKHSH *et al.*, 2017)	both	**96.8%**	**7.0%**	-	-	95.5%	11.0%	-	-
Plug-and-play CNN (RAVANBAKHSH *et al.*, 2018)	both	95.7%	8.0%	64.5%	40.8%	88.4%	18.0%	-	-
LMH (ADAM *et al.*, 2008)	motion	63.4%	38.9%	24.0%	-	58.1%	45.8%	-	-
MPPCA (KIM; GRAUMAN, 2009)	motion	59.0%	40.0%	-	82.0%	69.3%	30.0%	-	-
Social force (MEHRAN; OYAMA; SHAH, 2009)	motion	67.5%	31.0%	-	79.0%	55.6%	42.0%	-	-
Sparse reconstruction (CONG; YUAN; LIU, 2011)	motion	86.0%	19.0%	46.1%	54.0%	-	-	-	-
LSA (SALIGRAMA; CHEN, 2012)	motion	92.7%	16.0%	-	-	-	-	-	-
WMD (LEYVA; SANCHEZ; LI, 2014)	motion	-	19.0%	-	-	-	16.0%	-	25.0%
HOFM (COLQUE; CAETANO; SCHWARTZ, 2015)	motion	71.5%	33.3%	-	-	89.9%	19.0%	-	-
Motion influence map (LEE *et al.*, 2015)	motion	-	24.1%	-	-	-	9.8%	81.5%	-
Flow decomposition (PONTI; NAZARE; KITTLER, 2017)	motion	-	-	-	-	-	31.7%	-	-
SL-HOF (WANG *et al.*, 2016)	motion	87.5%	18.0%	64.4%	**35.0%**	95.1%	9.0%	81.0%	19.0%
Conv-WTA (best) (TRAN; HOGG, 2017)	motion	91.9%	14.8%	**68.7%**	35.7%	96.6%	8.9%	89.3%	16.9%
AE-HOFM (GEORGE *et al.*, 2018)	motion	78.0%	29.5%	-	-	91.0%	15.8%	-	-
Ours (best)	motion	87.7%	18.7%	52.3%	45.6%	**97.9%**	**6.1%**	**94.2%**	**9.8%**

Finally, we carried out a comparison using the UMN dataset as seen in Table 16, which confirms that our method is also competitive with regards to detecting global anomalies. Nonetheless, it is important to point out that, differently from the experiments with the UCSD datasets, these results were not obtained using exactly the same experimental setup. The reason is that most

Table 16 – UMN dataset performance comparison.

Method	Frame-level AUC
Chaotic Invariants (WU; MOORE; SHAH, 2010)	99%
Social Force (MEHRAN; OYAMA; SHAH, 2009)	96%
optical-flow (MEHRAN; OYAMA; SHAH, 2009)	84%
Sparse (CONG; YUAN; LIU, 2011)	97%
LSA (SALIGRAMA; CHEN, 2012)	98%
AE-HOFM (GEORGE *et al.*, 2018)	88%
Ours	95%

methods use the first 500 to 600 frames from each UMN video for training and the remaining ones for testing, while we only use the first 300 frames for training and start our testing at frame 500. We choose to carry out our experiments in this way because, in some cases, the first 500 frames of a video contains anomalous events, which could influence our results. Regardless of those differences, our method achieves state-of-the-art results on the UMN dataset, indicating its suitability to detect global anomalies.

5.6 Concluding remarks

This chapter presented an in-depth comparison of features from pre-trained CNNs models versus learning dataset-specific features with a 3D-convolutional auto-encoder. The experiments indicated that such auto-encoder is capable of achieving comparable results, while increasing computational efficiency (having less parameters). Next, EMD was tested as way to obtain filtered optical-flow estimates and, consequently, better anomaly detection results, which proved to be very effective in surveillance footage. Despite such good results, EMD is not suitable to be used in surveillance scenarios as it requires the entire series to be available beforehand. In order to make it possible for this filtering results to be used in an online setup, a combination of pre (increasing frame gap when computing the optical flow) and post-processing (applying a median filter to the anomaly scores) was used. Such approach, in combination with the usage of dynamic windows, was capable of improving the motion anomaly detection with regards to both the robustness to noisy optical-flow estimates and changes in viewing distances.

CHAPTER

6

CONCLUSIONS

This PhD thesis is motivated by the challenges of detecting anomalies in surveillance videos, more specifically to improve the motion modeling aspect of methods devoted to tackle such problem. After studying and revisiting the literature, we discovered that a major gap in the way that anomaly detection methods deal with motion modeling regarding both noisy optical-flow estimates and changes in viewing distances, thus leading to this thesis hypothesis:

> *i) the removal of noise from optical-flow estimates improves video anomaly detection results;*
>
> *ii) the capability of automatically adapting to changes in viewing distances boosts anomaly detection results.*

Regarding the first item of our hypothesis, we initially used EMD to remove the noisy part (stochastic component) from the optical-flow series. Our results confirmed that it is possible to obtain significant improvements in the quality of the extracted features and, therefore, in the anomaly detection performance. Despite demonstrating the relevance of filtering, EMD tends to be more suitable to address offline applications, limiting its applicability to the online setup (e.g. security videos). Building upon this idea, we conclude that a lightweight filtering technique (temporal median filter) can also improve optical-flow quality, while being more adequate to surveillance applications.

Concerning the second item of our hypothesis, we have shown that many methods face problems when dealing with variations in viewing distances. To analyze such factor, we started by testing the features generated by pre-trained CNN models from the image and video classification domains. Our results corroborated that those methods can deal with surveillance applications, but they are not capable of tackling changes in object-to-camera distances. In attempt to further understand this issue, we trained an auto-encoder (using only 3D-convolutional layers) to verify whether such model would adapt to variation within frames. Despite being capable of achieving

comparable results to the ones obtained from pre-trained models while requiring far less parameters, our experiments confirmed that the auto-encoder could not properly manage variations depending on frame regions. Finally, we proposed an adaptive motion modeling framework that dynamically adapts windows according to expected object sizes. As also confirmed by our experiments, such method achieves state-of-the-art anomaly detection results with the usage of an 8-dimensional descriptor, even when varying viewing distances. Lastly, this approach obtained outstanding results when detecting global anomalies.

In a nutshell, the first part of our experiments indicate that domain-specific models can maintain the same level anomaly detection performance of pre-trained ones, while having a lower computational cost. Next, in the second part of our experiments, we noticed that employing lightweight pre and post-processing techniques can greatly improve the detection results of optical-flow-based methods, without significantly increasing their computational complexity. Lastly, in the third part of our experiments, we discovered that optical flow can be leveraged to estimate changes in viewing distance, improving model detection performance and reducing setup time. Based on such ideas, we have build a video anomaly detection model that is capable to achieve state-of-the-art anomaly detection employing a simple 8-dimensional motion descriptor based only on optical-flow information. We believe that our findings can be used by future research to further improve anomaly detection in surveillance videos with regards to: increasing model detection performance, lowering computational cost and reducing setup time.

6.1 Publications

Some of the results of this projected are presented in the following publications:

- **Nazare, Tiago**; Mello, Rodrigo; Ponti, Moacir. “Investigating 3D convolutional layers as feature extractors for anomaly detection systems applied to surveillance videos”, 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISAPP 2021);

- **Nazare, Tiago**; Mello, Rodrigo; Ponti, Moacir. “Are pre-trained CNNs good feature extractors for anomaly detection in surveillance videos?”, 14th Workshop de Visão Computacional (WVC 2018);

- Ponti, Moacir; **Nazare, Tiago**; Kittler, Josef. “Optical-flow features empirical mode decomposition for motion anomaly detection”, 42th International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2017).

The remaining of our findings is going to be presented in the paper listed bellow, which is under review:

- **Nazare, Tiago**; Mello, Rodrigo; Ponti, Moacir. "A novel motion modeling framework for anomaly detection in surveillance videos".

Lastly, during this project, we also collaborated on the publication of the following papers:

- Souza, Andrey; Lacerda, Wilian; Forster, Carlos Henrique; Lima, Danilo; **Nazare, Tiago**. "Aplicação de Redes Neurais Siamesas na Autenticação de Condutores", 14th Simpósio Brasileiro de Automação Inteligente (SBAI 2019);
- Kubo, Diandra; **Nazare, Tiago**; Aguirre, Priscila; Duarte, Felipe; Oliveira, Bruno. "On the usage of U-Net for pre-processing document images", Workshop of Works in Progress – 31th Conference on Graphics, Patterns and Images (SIBGRAPI 2018);
- **Nazare, Tiago**; Costa, Gabriel; Mello, Rodrigo; Ponti, Moacir. "Color quantization in transfer learning and noisy scenarios: an empirical analysis using convolutional networks", 31th Conference on Graphics, Patterns and Images (SIBGRAPI 2018);
- **Nazare Tiago**, Costa, Gabriel, Contato, Welinton, Ponti, Moacir. "Deep convolutional neural networks and noisy images", 22th Iberoamerican Congress on Pattern Recognition (CIARP 2017);
- Ponti, Moacir; Ribeiro, Leonardo; **Nazare, Tiago**; Bui, Tu; Collomosse, John. "Everything you wanted to know about Deep Learning for Computer Vision but were afraid to ask", Tutorials of the 30th Conference on Graphics, Patterns and Images (SIBGRAPI 2017);
- Costa, Gabriel; Contato, Welinton; **Nazare, Tiago**; Batista Neto, João; Ponti, Moacir. "An empirical study on the effects of different types of noise in image classification tasks", 12th Workshop de Visão Computacional (WVC 2016);
- Contato, Welinton; **Nazare, Tiago**; Costa, Gabriel; Ponti, Moacir; Batista Neto, João. "Improving Non-Local Video Denoising with Local Binary Patterns and Image Quantization", 29th Conference on Graphics, Patterns and Images (SIBGRAPI 2016);
- Ponti, Moacir; **Nazare, Tiago**; Thumé, Gabriela. "Image quantization as a dimensionality reduction procedure in color and texture feature extraction", Neurocomputing (Amsterdam), 2016.

6.2 Future work

A list of further possibilities is resultant of this PhD thesis. Among the most relevant items are:

- The combination of our approach on dynamic windows with appearance descriptors (e.g. features from pre-trained CNNs) to take advantage of such visual clues in our pipeline;

- The CNN models used in our experiments (e.g. VGG-16 and C3D) can be retrained using a gray-scale/noisier versions of the original datasets. This is specially motivated by some of our results (NAZARE *et al.*, 2018; COSTA *et al.*, 2016; NAZARE *et al.*, 2017), which have confirmed that such approach can be beneficial;
- Explore other pre-trained 3D-convolutional CNNs as features extractors for security footage (e.g. the models compared in (CARREIRA; ZISSERMAN, 2017));
- Employ the approach by Lee *et al.* (2015) to analyze neighboring regions in attempt to improve the way our method describes complex motion patterns;
- Better understand the behavior of different optical-flow methods when used in our framework and experiment with optical-flow estimation based on deep neural networks (e.g. the one presented by Ilg *et al.* (2017));
- Adapt our dynamic window approach to operate with moving cameras.

BIBLIOGRAPHY

ADAM, A.; RIVLIN, E.; SHIMSHONI, I.; REINITZ, D. Robust real-time unusual event detection using multiple fixed-location monitors. **IEEE Trans. Pattern Anal. Mach. Intell.**, v. 30, n. 3, p. 555–560, 2008. Citations on pages 30, 33, 34, 35, 49, 63, 64, 68, 71, 73, 79, 93, and 97.

ALBERTINI, M. K.; MELLO, R. F. de. A self-organizing neural network for detecting novelties. In: **Proceedings of the 2007 ACM Symposium on Applied Computing**. New York, NY, USA: ACM, 2007. p. 462–466. Citation on page 33.

ANTIć, B.; OMMER, B. Video parsing for abnormality detection. In: **2011 International Conference on Computer Vision**. [S.l.: s.n.], 2011. p. 2415–2422. Citation on page 96.

ASHBY, M. P. The value of cctv surveillance cameras as an investigative tool: An empirical analysis. **European Journal on Criminal Policy and Research**, Springer, v. 23, n. 3, p. 441–459, 2017. Citation on page 29.

BENGIO, Y.; SIMARD, P.; FRASCONI, P. Learning long-term dependencies with gradient descent is difficult. **IEEE transactions on neural networks**, IEEE, v. 5, n. 2, p. 157–166, 1994. Citation on page 46.

BOUGUET, J. yves. Pyramidal implementation of the lucas kanade feature tracker. **Intel Corporation, Microprocessor Research Labs**, 2000. Citation on page 59.

BROX, T.; BRUHN, A.; PAPENBERG, N.; WEICKERT, J. High accuracy optical flow estimation based on a theory for warping. In: SPRINGER. **European conference on computer vision**. [S.l.], 2004. p. 25–36. Citation on page 95.

CARREIRA, J.; ZISSERMAN, A. Quo vadis, action recognition? a new model and the kinetics dataset. In: **proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2017. p. 6299–6308. Citation on page 102.

CHAUDHRY, R.; VIDAL, R.; HAGER, G.; RAVICHANDRAN, A. Histograms of oriented optical flow and binet-cauchy kernels on nonlinear dynamical systems for the recognition of human actions. **2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshops)**, IEEE Computer Society, Los Alamitos, CA, USA, v. 00, n. undefined, p. 1932–1939, 2009. Citation on page 58.

CHOLLET, F. Xception: Deep learning with depthwise separable convolutions. In: **2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2017. p. 1251–1258. Citations on pages 44, 45, 46, and 84.

COLQUE, R. V. H. M.; CAETANO, C.; SCHWARTZ, W. R. Histograms of optical flow orientation and magnitude to detect anomalous events in videos. In: **Conference on Graphics, Patterns and Images (SIBGRAPI)**. [S.l.: s.n.], 2015. p. 1–8. Citations on pages 16, 29, 30, 35, 58, 59, 62, 63, 68, 71, 73, and 97.

CONG, Y.; YUAN, J.; LIU, J. Sparse reconstruction cost for abnormal event detection. In: **CVPR**. [S.l.]: IEEE Computer Society, 2011. p. 3449–3456. ISBN 978-1-4577-0394-2. Citations on pages 97 and 98.

COSTA, G. B. P. da; CONTATO, W. A.; NAZARE, T. S.; NETO, J. do E. S. B.; PONTI, M. An empirical study on the effects of different types of noise in image classification tasks. **CoRR**, abs/1609.02781, 2016. Citation on page 102.

COSTA, G. de Barros Paranhos da. **Detecção de anomalias utilizando métodos paramétricos e múltiplos classificadores**. Master's Thesis (Master's Thesis) — Universidade de São Paulo, 2014. Citations on pages 15, 33, and 34.

DARBELLAY, G. A.; TICHAVSKY, P. Independent component analysis through direct estimation of the mutual information. In: **ICA**. [S.l.: s.n.], 2000. v. 2000, p. 69–75. Citation on page 36.

DENG, J.; DONG, W.; SOCHER, R.; LI, L.-J.; LI, K.; FEI-FEI, L. ImageNet: A Large-Scale Hierarchical Image Database. In: **CVPR09**. [S.l.: s.n.], 2009. p. 248–255. Citations on pages 45 and 84.

FARNEBÄCK, G. Two-frame motion estimation based on polynomial expansion. In: **Proceedings of the 13th Scandinavian Conference on Image Analysis**. Berlin, Heidelberg: Springer-Verlag, 2003. (SCIA'03), p. 363–370. ISBN 3-540-40601-8. Citations on pages 68 and 95.

GEORGE, M.; JOSE, B. R.; MATHEW, J.; KOKARE, P. Autoencoder-based abnormal activity detection using parallelepiped spatio-temporal region. **IET Computer Vision**, IET, v. 13, n. 1, p. 23–30, 2018. Citations on pages 30, 35, 62, 63, 64, 71, 73, 97, and 98.

GONG, Y.; LAZEBNIK, S. Iterative quantization: A procrustean approach to learning binary codes. In: **Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition**. Washington, DC, USA: IEEE Computer Society, 2011. (CVPR '11), p. 817–824. ISBN 978-1-4577-0394-2. Citation on page 61.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>. Citations on pages 42 and 44.

GUHA, T.; WARD, R. K. Learning sparse representations for human action recognition. **Pattern Analysis and Machine Intelligence, IEEE Transactions on**, IEEE, v. 34, n. 8, p. 1576–1588, 2012. Citations on pages 15, 52, and 53.

HAWKINS, D. **Identification of Outliers**. [S.l.]: Chapman and Hall, 1980. (Monographs on applied probability and statistics). ISBN 9780412219009. Citation on page 33.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. In: **The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2016. p. 770–778. Citations on pages 44, 45, 46, and 84.

HORE, P.; HALL, L.; GOLDGOF, D.; GU, Y.; MAUDSLEY, A.; DARKAZANLI, A. A scalable framework for segmenting magnetic resonance images. **Journal of Signal Processing Systems**, Springer US, v. 54, n. 1-3, p. 183–203, 2009. ISSN 1939-8018. Citation on page 57.

HORN, B. K. P.; SCHUNCK, B. G. Determining optical flow. **Artificial Intelligence**, v. 17, p. 185–203, 1981. Citation on page 35.

HU, X.; HU, S.; ZHANG, X.; ZHANG, H.; LUO, L. Anomaly detection based on local nearest neighbor distance descriptor in crowded scenes. **The Scientific World Journal**, Hindawi Publishing Corporation, v. 2014, 2014. Citations on pages 29, 52, 53, 63, and 97.

HUANG, G.; LIU, Z.; MAATEN, L. van der; WEINBERGER, K. Q. Densely connected convolutional networks. In: **The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2017. p. 2261–2269. Citations on pages 44, 45, 47, and 84.

HUANG, N. E.; SHEN, Z.; LONG, S. R.; WU, M. C.; SHIH, H. H.; ZHENG, Q.; YEN, N.-C.; TUNG, C. C.; LIU, H. H. The empirical mode decomposition and the hilbert spectrum for nonlinear and non-stationary time series analysis. In: THE ROYAL SOCIETY. **Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences**. [S.l.], 1998. v. 454, n. 1971, p. 903–995. Citation on page 36.

HUBER, P. J. **Robust statistics**. [S.l.]: Springer, 2011. Citation on page 72.

ILG, E.; MAYER, N.; SAIKIA, T.; KEUPER, M.; DOSOVITSKIY, A.; BROX, T. Flownet 2.0: Evolution of optical flow estimation with deep networks. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2017. p. 2462–2470. Citation on page 102.

KIM, J.; GRAUMAN, K. Observe locally, infer globally: A space-time mrf for detecting abnormal activities with incremental updates. In: **CVPR**. [S.l.]: IEEE Computer Society, 2009. p. 2921–2928. ISBN 978-1-4244-3992-8. Citation on page 97.

KRATZ, L.; NISHINO, K. Anomaly detection in extremely crowded scenes using spatio-temporal motion pattern models. In: **CVPR**. [S.l.]: IEEE Computer Society, 2009. p. 1446–1453. ISBN 978-1-4244-3992-8. Citation on page 51.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: **Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems.** [S.l.: s.n.], 2012. p. 1106–1114. Citation on page 61.

LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. Gradient-based learning applied to document recognition. In: **Proceedings of the IEEE**. [S.l.: s.n.], 1998. p. 2278–2324. Citation on page 43.

LEE, D.-g.; MEMBER, S.; SUK, H.-i.; PARK, S.-k. Motion Influence Map for Unusual Human Activity Detection and Localization in Crowded Scenes. **Circuits and Systems for Video Technology, IEEE Transactions on**, IEEE, v. 8215, n. c, p. 1–12, 2015. Citations on pages 16, 29, 53, 55, 63, 97, and 102.

LEYVA, R.; SANCHEZ, V.; LI, C.-T. Video anomaly detection based on wake motion descriptors and perspective grids. In: IEEE. **2014 IEEE International Workshop on Information Forensics and Security (WIFS)**. [S.l.], 2014. p. 209–214. Citations on pages 30, 35, and 97.

LEYVA, R.; SANCHEZ, V.; LI, C. T. Video anomaly detection with compact feature sets for online performance. **IEEE Transactions on Image Processing**, IEEE, Institute of Electrical and Electronics Engineers, v. 26, n. 7, p. 3463–3478, 7 2017. ISSN 1057-7149. Citation on page 62.

LI NANNAN, W. X. X. D. G. H. F. W. Spatio-temporal context analysis within video volumes for anomalous-event detection and localization. **Neurocomputing**, v. 155, n. Complete, p. 309–319, 2015. Citations on pages 16, 56, 58, 63, and 97.

LI, W.; MAHADEVAN, V.; VASCONCELOS, N. Anomaly detection and localization in crowded scenes. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 36, n. 1, p. 18–32, 2014. Citations on pages 29, 78, 82, and 97.

LIU, P.; LYU, M.; KING, I.; XU, J. Selflow: Self-supervised learning of optical flow. In: **The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2019. p. 4571–4580. Citations on pages 30 and 35.

LU, C.; SHI, J.; JIA, J. Abnormal event detection at 150 FPS in MATLAB. In: **IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013**. [S.l.: s.n.], 2013. p. 2720–2727. Citations on pages 29, 30, 51, 63, 80, and 97.

LUCAS, B. D.; KANADE, T. An iterative image registration technique with an application to stereo vision. In: **Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1981. (IJCAI'81), p. 674–679. Citation on page 35.

LUVISON, B.; LAPRESTE, J.; SAYD, P.; PHAM, Q.; CHATEAU, T. **Automatic Detection of Unexpected Events in Dense Areas for Videosurveillance Applications**. [S.l.]: INTECH Open Access Publisher, 2011. ISBN 9789533074368. Citation on page 29.

MAHADEVAN, V.; LI, W.; BHALODIA, V.; VASCONCELOS, N. Anomaly detection in crowded scenes. In: **Proceedings of IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2010. p. 1975–1981. Citation on page 78.

MAIRAL, J.; BACH, F.; PONCE, J.; SAPIRO, G. Online dictionary learning for sparse coding. In: ACM. **Proceedings of the 26th Annual International Conference on Machine Learning**. [S.l.], 2009. p. 689–696. Citation on page 58.

MEHRAN, R.; OYAMA, A.; SHAH, M. Abnormal crowd behavior detection using social force model. In: **CVPR**. [S.l.]: IEEE Computer Society, 2009. p. 935–942. ISBN 978-1-4244-3992-8. Citations on pages 97 and 98.

MUJA, M.; LOWE, D. G. Scalable nearest neighbor algorithms for high dimensional data. **Pattern Analysis and Machine Intelligence, IEEE Transactions on**, IEEE, v. 36, p. 2227–2240, 2014. Citations on pages 85 and 88.

NAIR, V.; HINTON, G. E. Rectified linear units improve restricted boltzmann machines. In: FüRNKRANZ, J.; JOACHIMS, T. (Ed.). **Proceedings of the 27th International Conference on Machine Learning (ICML-10)**. [S.l.]: Omnipress, 2010. p. 807–814. Citation on page 37.

NAPHADE, M.; TANG, Z.; CHANG, M.-C.; ANASTASIU, D. C.; SHARMA, A.; CHELLAPPA, R.; WANG, S.; CHAKRABORTY, P.; HUANG, T.; HWANG, J.-N. *et al.* The 2019 ai city challenge. In: **CVPR Workshops**. [S.l.: s.n.], 2019. Citation on page 29.

NAZARE, T. S.; COSTA, G. B. P. da; CONTATO, W. A.; PONTI, M. Deep convolutional neural networks and noisy images. In: SPRINGER. **Iberoamerican Congress on Pattern Recognition**. [S.l.], 2017. p. 416–424. Citation on page 102.

NAZARE, T. S.; MELLO, R. F. de; PONTI, M. A. Are pre-trained cnns good feature extractors for anomaly detection in surveillance videos? **14th Workshop de Visão Computacional (WVC 2018)**, p. 50–55, 2018. Citations on pages 15, 17, 29, 46, 47, 84, 87, and 97.

NAZARE, T. S. de; COSTA, G. d. B. P. da; MELLO, R. F. de; PONTI, M. A. Color quantization in transfer learning and noisy scenarios: an empirical analysis using convolutional networks. In: **31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI 2018)**. [S.l.: s.n.], 2018. p. 377–383. Citation on page 102.

OLUWATOYIN, P. P.; WANG, K. Video-based abnormal human behavior recognition - a review. **IEEE Transactions on Systems, Man, and Cybernetics, Part C**, v. 42, n. 6, p. 865–878, 2012. Citation on page 29.

PIZA, E. L.; WELSH, B. C.; FARRINGTON, D. P.; THOMAS, A. L. Cctv surveillance for crime prevention: A 40-year systematic review with meta-analysis. **Criminology & Public Policy**, Wiley Online Library, v. 18, n. 1, p. 135–159, 2019. Citation on page 29.

PONTI, M.; NAZARE, T.; KITTLER, J. Optical-flow features empirical mode decomposition for motion anomaly detection. In: **IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2017)**. [S.l.: s.n.], 2017. p. 1403–1407. Citations on pages 16, 17, 18, 29, 35, 68, 69, 92, 93, and 97.

PONTI, M. A.; RIBEIRO, L. S. F.; NAZARE, T. S.; BUI, T.; COLLOMOSSE, J. Everything you wanted to know about deep learning for computer vision but were afraid to ask. In: **30th SIBGRAPI Conference on Graphics, Patterns and Images Tutorials (SIBGRAPI 2017)**. [S.l.: s.n.], 2017. p. 17–41. Citations on pages 42 and 43.

RADKE, R. J. **Computer Vision for Visual Effects**. New York, NY, USA: Cambridge University Press, 2012. ISBN 0521766877, 9780521766876. Citation on page 35.

RAVANBAKHSH, M.; NABI, M.; MOUSAVI, H.; SANGINETO, E.; SEBE, N. Plug-and-play cnn for crowd motion analysis: An application in abnormal event detection. In: **2018 IEEE Winter Conference on Applications of Computer Vision (WACV)**. [S.l.: s.n.], 2018. p. 1689–1698. Citations on pages 29, 61, 63, 84, and 97.

RAVANBAKHSH, M.; SANGINETO, E.; NABI, M.; SEBE, N. Training adversarial discriminators for cross-channel abnormal event detection in crowds. **CoRR**, abs/1706.07680, 2017. Citations on pages 29 and 97.

RIOS, R. A.; MELLO, R. F. d. Applying empirical mode decomposition and mutual information to separate stochastic and deterministic influences embedded in signals. **Signal Process.**, Elsevier North-Holland, Inc., Amsterdam, The Netherlands, The Netherlands, v. 118, n. C, p. 159–176, Jan. 2016. ISSN 0165-1684. Citations on pages 36 and 67.

ROSENBERG, Y.; WERMAN, M. Representing local motion as a probability distribution matrix and object tracking. In: **DARPA Image Undersading Workshop**. [S.l.: s.n.], 1997. p. 153–158. Citation on page 49.

ROSS, D. A.; LIM, J.; LIN, R.-S.; YANG, M.-H. Incremental learning for robust visual tracking. **Int. J. Comput. Vision**, v. 77, n. 1-3, p. 125–141, May 2008. ISSN 0920-5691. Citations on pages 85 and 88.

SALIGRAMA, V.; CHEN, Z. Video anomaly detection based on local statistical aggregates. In: **CVPR**. [S.l.]: IEEE Computer Society, 2012. p. 2112–2119. ISBN 978-1-4673-1226-4. Citations on pages 97 and 98.

SEIDENARI, L.; BERTINI, M. Non-parametric anomaly detection exploiting space-time features. In: BIMBO, A. D.; CHANG, S.-F.; SMEULDERS, A. W. M. (Ed.). **ACM Multimedia**. [S.l.]: ACM, 2010. p. 1139–1142. ISBN 978-1-60558-933-6. Citation on page 29.

SHI, J.; MALIK, J. Motion segmentation and tracking using normalized cuts. In: IEEE. **Computer Vision, 1998. Sixth International Conference on**. [S.l.], 1998. p. 1154–1160. Citation on page 49.

SHIRDHONKAR, S.; JACOBS, D. W. Approximate earth mover's distance in linear time. **2014 IEEE Conference on Computer Vision and Pattern Recognition**, IEEE Computer Society, Los Alamitos, CA, USA, v. 0, p. 1–8, 2008. Citation on page 53.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. **CoRR**, abs/1409.1556, 2014. Citations on pages 45 and 84.

SZEGEDY, C.; VANHOUCKE, V.; IOFFE, S.; SHLENS, J.; WOJNA, Z. Rethinking the inception architecture for computer vision. In: **IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2016. p. 2818–2826. Citation on page 46.

TRAN, D.; BOURDEV, L.; FERGUS, R.; TORRESANI, L.; PALURI, M. Learning spatiotemporal features with 3d convolutional networks. In: **IEEE International Conference on Computer Vision (ICCV)**. [S.l.: s.n.], 2015. p. 4489–4497. Citations on pages 45, 47, 65, and 85.

TRAN, H. T.; HOGG, D. Anomaly detection using a convolutional winner-take-all autoencoder. In: BRITISH MACHINE VISION ASSOCIATION. **Proceedings of the British Machine Vision Conference 2017**. [S.l.], 2017. p. 139.1–139.12. Citation on page 97.

WANG, S.; ZHU, E.; YIN, J.; PORIKLI, F. Anomaly detection in crowded scenes by sl-hof descriptor and foreground classification. In: **23rd International Conference on Pattern Recognition (ICPR 2016)**. [S.l.: s.n.], 2016. p. 3398–3403. Citation on page 97.

WU, S.; MOORE, B. E.; SHAH, M. Chaotic invariants of lagrangian particle trajectories for anomaly detection in crowded scenes. In: **CVPR**. [S.l.]: IEEE Computer Society, 2010. p. 2054–2060. ISBN 978-1-4244-6984-0. Citation on page 98.

XU, D.; RICCI, E.; YAN, Y.; SONG, J.; SEBE, N. Learning deep representations of appearance and motion for anomalous event detection. In: XIE, X.; JONES, M. W.; TAM, G. K. L. (Ed.). **BMVC**. [S.l.]: BMVA Press, 2015. p. 8.1–8.12. ISBN 1-901725-53-7. Citations on pages 16, 29, 30, 60, 63, 65, 73, and 97.

YOSINSKI, J.; CLUNE, J.; BENGIO, Y.; LIPSON, H. How transferable are features in deep neural networks? In: **Advances in neural information processing systems**. [S.l.: s.n.], 2014. p. 3320–3328. Citation on page 87.

ZACH, C.; POCK, T.; BISCHOF, H. A duality based approach for realtime tv-l 1 optical flow. In: SPRINGER. **Joint pattern recognition symposium**. [S.l.], 2007. p. 214–223. Citation on page 95.


ICMC
USP
SÃO CARLOS