POLITECNICO DI TORINO

Master's Degree in Computer Engineering

Automation and Intelligent Cyber-Physical Systems

Politecnico di Torino

Master's Degree Thesis

Anomaly and Violence Detection in Surveillance Videos

Supervisor

Prof. Maurizio MORISIO

Candidate

Mattia TORRE

April 2023

Summary

With the technological progress in multiple fields of research, there are ever more application cases where technology has proven to both improve human work conditions and serve the public interest; in particular, the development of Artificial Intelligence and Machine Learning methods has gained impressive momentum in the later years, especially with regards to visual and data problems.

Among others, a topic of trending interest in the computer science world is the automatic detection of suspicious events from surveillance videos. The challenges of this task are several, from the mere definition of an abnormal act to the problem description, which could be tackled from different perspectives, to the generalization complexity due to the variety of human behavior and the trade-off between accuracy and available resources.

In this thesis, some of the prevalent methods available in the literature for the problem of anomaly and fight detection have been analyzed, with particular attention on the most exhaustive available collections of surveillance videos, namely UCF-Crime and RWF-2000. These two datasets have been thereby used in the construction of a novel light architecture, posing the problem as a binary classification task and employing a dual representation of the input, leveraging the informative aspect of the optical flow.

The novelties that this work carries are threefold: the introduction of a novel convolutional block that aims to improve the Inflated 3D architecture, at present a de-facto standard in the field; an adaptive cropping strategy to minimize the information loss in the resize processing; the fine-tuning of a lightweight CNN-model for the optical flow estimation to reduce the pre-processing and inference times. Moreover, a basic interactive system has been developed for helping the operator experience, which produces both a visual output and a textual log by applying the trained model on the prompted files.

The achieved results improve the state of the art with an accuracy on the validation set of 90.45%, an upgrade of +0.7%. Nevertheless, this work still showcases some limitations and the problem remains definitely open; further improvements stemming from this project are therefore envisaged, following the path taken by the author.

"Qualche volta il destino assomiglia a una tempesta di sabbia che muta incessantemente la direzione del percorso. Per evitarlo cambi l'andatura. E il vento cambia andatura, per seguirti meglio. Tu allora cambi di nuovo, e subito di nuovo il vento cambia per adattarsi al tuo passo. Questo si ripete infinite volte, come una danza sinistra col dio della morte prima dell'alba. Perché quel vento non è qualcosa che è arrivato da lontano, indipendente da te. È qualcosa che hai dentro. Quel vento sei tu.

Perciò l'unica cosa che puoi fare è entrarci, in quel vento, camminando dritto, e chiudendo forte gli occhi per non far entrare la sabbia. Attraversarlo, un passo dopo l'altro.

Poi, quando la tempesta sarà finita, probabilmente non saprai neanche tu come hai fatto ad attraversarla e a uscirne vivo. Anzi, non sarai neanche sicuro se sia finita per davvero. Ma su un punto non c'è dubbio. Ed è che tu, uscito da quel vento, non sarai lo stesso che vi era entrato."

Haruki Murakami

Acknowledgements

In queste poche righe vorrei ringraziare chi ha reso possibile il raggiungimento di questo importante traguardo della mia carriera: i miei genitori, i miei fratelli e gli amici più stretti. Questo lavoro, e ciò che ne conseguirà, assumono significato solo nella misura in cui le gioie, le soddisfazioni e le difficoltà sono stati condivisi con voi. Grazie.

Un sentito ringraziamento anche a chi mi ha accompagnato in questa fase finale di questo percorso, il professor Maurizio Morisio per la disponibilità mostrata nei miei confronti e i miei tutor aziendali, Carla Federica Melia e Dario Sammaruga, e a tutti i colleghi e professori incotrati nel corso di questi anni.

Table of Contents

List of Tables

List of Figures

Acronyms

AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
BCE	Binary Cross Entropy
C3D	3D Convolutional Neural Network
CCTV	Closed-Circuit TeleVision
CMY	Cyan, Magenta, Yellow
CMYK	Cyan, Magenta, Yellow, blacK
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CTU	Coding Tree Unit
CV	Computer Vision
DCT	Discrete Cosine Transform
DL	Deep Learning
DN	Digital Number
DNN	Deep Neural Network
DT	Dense Trajectories
ETL	Extract, Transform, Load
FC	Fully Connected
FPS	Frames Per Second
FT	Fine-Tuned
GD	Gradient Descent
GMM	Gaussian Mixture Models
GPU	Graphics Processing Unit
HECV	High-Efficiency Video Coding
HMM	Hidden Markov Models
HSL	Hue, Saturation, Lightness
HSV	Hue, Saturation, Value
LSTM	Long Short-Term Memory

MC	Motion Compensation
MD	Mean absolute Differences
MIL	Multiple Instance Learning
ML	Machine Learning
MPEG	Moving Picture Experts Group
MV	Motion Vector
NN	Neural Network
OF	Optical Flow
PCM	Pulse-Code Modulation
ReLU	Rectified Linear Unit
ResNet	Residual Network
RGB	Red, Green, Blue
RNN	Recurrent Neural Network
ROI	Region Of Interest
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine
TSN	Temporal Segment Network
UGC	User Generated Content

Chapter 1

Introduction

The history of mankind has been marked by technological advances to such an extent that, nowadays, every aspect of human life is affected by and blended with technology. The increase in computational and parallel power in processing units has accompanied research progress, allowing the computer science world to better cope with the more and more large and complex collections of digital data available, extracting knowledge and insights and utilizing them to provide new and efficient services.

Long considered one of the greatest challenges for Artificial Intelligence methods, videos are now becoming ever-increasing relevant as one of the most effective and pervasive information mediums. It is estimated that, in 2022, 82% of the whole consumer Internet traffic would be made up of video traffic, up from 73% in 2017, with a compound annual growth rate of 34% [1]. A great factor in the present rise is the enormous quantity of User Generated Content available online: to provide an overview, each day over 720,000 hours of new video content are uploaded second the esteem [2] to YouTube, which is only one of the many social platforms widespread today.

A separate analysis should be made on the rising of autonomous cameras, which today are essentially mounted and spread in each corner of our cities. People's actions are silently recorded every day by webcams, surveillance systems, cars, door phones, and even drones in the sky and robots in industrial estates; an average person in the United Kingdom is estimated to be filmed by a Closed-Circuit Television (CCTV) camera at least 70 times per day [3].

The act of processing and extracting useful knowledge from videos presents several criticalities due to the peculiar characteristics of such a medium which merges together spatial and temporal information. Nevertheless, the applied fields of use are extremely varied, indicating its potential, ranging from context awareness (e.g., for autonomous driving systems) to value extraction (for social media and market analysis) to visual searching, and several other complex tasks.

The specific research field of the present work is the automatic detection of "abnormal activities" in videos recorded by surveillance cameras. This problem, depending on the precise formulation and implementation, can be considered pertaining to the video classification, crowd action detection, or human activity recognition area. Notwithstanding the specifics, the need for algorithms of this kind is dependent on the great deficiency between the burgeoning amount of surveillance camera videos to be analyzed and the available human labor force. The most recent statistics estimate a 15% decrease in crime rate in case of active monitoring (i.e., CCTV cameras actively watched by trained staff) with respect to CCTV systems with no active monitoring [4], and while fully automatic mechanisms for threats detection are still not considered acceptable, various supporting methods have proven to aid the operators' work, making it less exhausting [5].

The project's final aim is to produce a simple but effective interactive interface that allows the extraction of video portions considered to be "suspicious" from customized video surveillance videos uploaded by the user. In this way, the goal is to rationalize the workload of exploring documentary evidence, in a mostly off-line methodology, by providing the operator with a large but not definitive projection of video fragments.

In this prefatory chapter, a brief theoretical overview of the project background scope is given, starting from the mathematical model of digital images and videos in Sect. 1.1, with an explicit deepening on data compression. Sect. 1.2 contains a concise historical survey on Computer Vision techniques and tasks; an introduction to Machine Learning and Deep Neural Networks is provided in Sect. 1.3, whereas some of the Computer Vision descriptors and methods applied to digital videos are examined in Sect. 1.4. This discussion has been regarded as fundamental by the author in order to fully comprehend the techniques and architectures implemented in the subsequent work, yielding a coherent and extensive theoretical framework and preventing any possible misapprehensions. Nevertheless, the following sections have not to be considered a thorough dissertation since the descriptions deemed unnecessary have been omitted.

1.1 Theoretical Framework

Vision is widely considered to be the most advanced of the human senses: using the eyes, the optic nerve, the optic tract, and the visual cortex, humans are able to see and interpret the surrounding environment. The world, the natural phenomena, and the inputs perceivable from it are analog, that is continuous in time, and with continuous values.

Conversely, computers are digital: the information is represented as discrete

in time (i.e., *sampled*) and in amplitude (i.e., *quantized*) [6], in a process called "digitization". Finally, the discrete signals are coded into the binary number system, encoding the data in bit strings according to leading standards like ASCII for characters in electronic communication or PCM for digital audio.

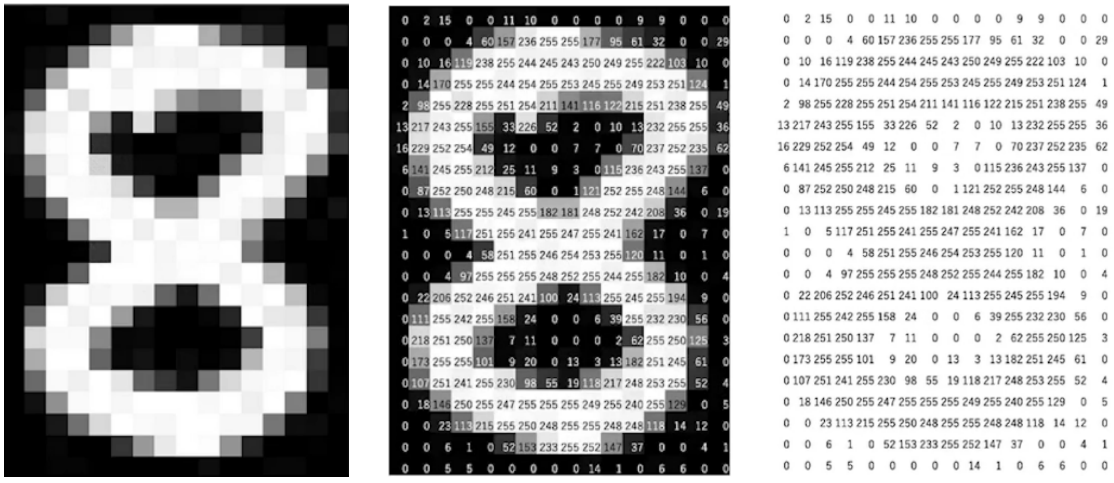
1.1.1 Digital Images

As defined in [7],

> A digital image is a representation of a real image as a set of numbers that can be stored and handled by a digital computer.

There exist two types of digital images: raster images and vector graphics. The subsequent treatise will focus exclusively on raster images.

A raster image is a matrix data structure composed of **m** rows and **n** columns having *pixels* (picture elements) as its smallest addressable element. Therefore, each pixel is characterized by a row index i and a column index j, and they are arranged in a regular - usually rectangular - grid on a Cartesian coordinate system. Every pixel is also distinguished by an integer number called "Digital Number" (DN), which usually refers to the brightness level measured for the ground resolution cell represented by that specific pixel [8]. In Fig. 1.1, a raster image of the number 8 is visualized, together with its matrix representation.



(a) **(b)** **(c)**

Figure 1.1: A digital image of the number 8 and its matrix representation: a DN of 0 corresponds to total black, whereas the maximum brightness level achievable in this example, representing pure white, is 255. Images from [9].

As every color perceived by the cone cells in the human eye can be defined by a linear combination of the three primary colors, red, green, and blue [10], likewise a digital color image is formed by a combination of individual 2-D images, often following the same trichromacy principle. The two most common color models, respectively used in digital media visualization and ink printing, are RGB (red, green, and blue), which is an additive model, and CMYK (cyan, magenta, yellow, and black), a subtractive model[1]. In the following dissertation and work, the RGB schema, unless otherwise stated, is mainly employed.

In this system, a color image consists of three individual matricial images, one for each channel: each pixel's DN of the conjunct image is in fact a vector of three numbers (i.e., the brightness level for each color). The number of bits destined to represent each color channel is referred to as "color depth": assuming a standard color depth of 8 bits, a total number of 256 intensity levels for each of the three channels for a count of ≈ 16.777 millions of different colors can be represented[2]. Therefore, a single pixel can be defined as a vector function:

$$p(i,j) = [r(i,j), g(i,j), b(i,j)] \tag{1.1}$$

with $r, g, b : \{0, ..., h\} \times \{0, ..., w\} \rightarrow \{0, ...,255\}$ the distinct scalar functions associating a single-channel brightness value to the bi-dimensional position (i, j). In Fig. 1.2, a digital image decomposed in its distinct color channels is represented as an illustration.

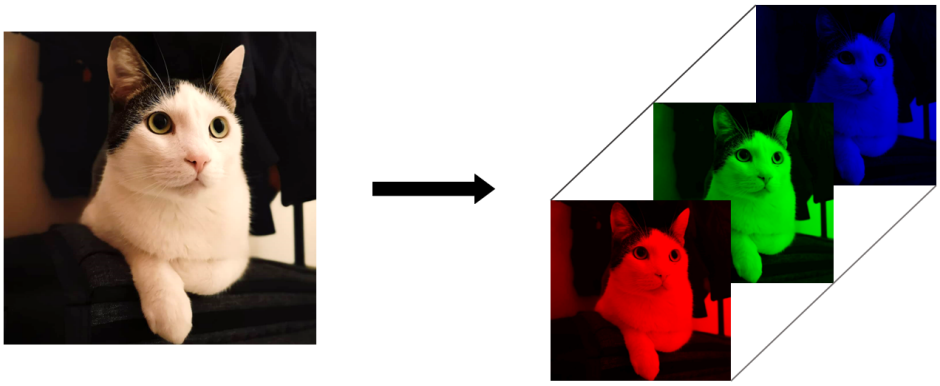


Figure 1.2: A digital image of a cat, decomposed in the RGB channels.

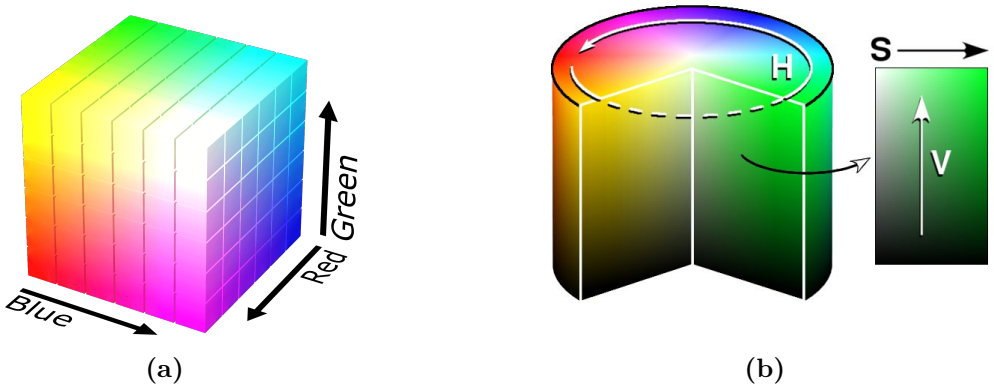
Altogether, a raster digital image can be defined as a tensor $\mathcal{I}^{h \times w \times c}$, where h

[1]Additive models function by summing the numeric representations of the single component colors to form the final color, while a subtractive color mixing works in the opposite way leveraging the absorbing power of materials on a reflecting or transparent, thus white, surface.

[2]This standard is known as True Color model.

stands for height (i.e. the number of pixels for each row of the grid), w for width (i.e., columns), with the product $h \times w$ known as resolution, and $c = 3$ the number of color channels of the image.

An important contribution made by computer graphics researchers in the 1970s has been the introduction of HSL (hue, saturation, and lightness) and HSV (hue, saturation, and value) as an alternative color representation models with respect to the RGB and CMYK: in particular, their hallmark is to treat the color shades of each hue via coordinates of cylindrical slices, as shown in Fig. 1.3. They are classified as user-oriented models as the aim is to better approximate the human perception of colors [11].



(a) (b)

Figure 1.3: The comparison between the RGB cube (a) and the HSV cylinder (b), via [12].

1.1.2 Videos

As defined in [13],

> A digital video is an electronic representation of moving visual images (video) in the form of encoded digital data.

Without loss of generality, a digital video can be described as the sequence of digital images (called "frames"), shown in succession at a rate termed *fps*, which stands for "frames per second". Typical fps values are in the range of $20 - 30$, although, depending on the field of usage, values from as few as 5 to as many as 60 and 120 are widely used. To date, the fastest digital camera available on the market is the Phantom v2512, which achieves up to 1 million frames per second at a limited resolution.

A video signal contains information in three dimensions, modeled as spatial (x, y) and temporal (t) domains. It is also frequent to have digital videos encapsulating

audio information (e.g., a digital audio track), extending the domain to a fourth informative time-dependent dimension in the form of a digitized sound wave. However, this possibility is left unexamined for the purpose of the present thesis since uncommon in the applicative field of use.

In consistency with the treatise on digital images, a single frame is characterized dimensionally by the width w and the height h of the frame and by the color depth: therefore, as far as memory usage is concerned, a raw digital video occupies fps times the number of bytes required to represent a still image of the same characteristics. It stands to reason that the compression of digital videos is an essential process when there exists a limit on the computational resources at disposal.

1.1.3 Compression

Data compression is the process of encoding information using fewer bits with respect to the original representation. In the context of digital videos, the operation of video compression is of paramount importance due to the substantial storage occupation and bandwidth needed to record, process, and transfer raw video. The term *codec* has been coined as a portmanteau of both words "coder" and "encoder", and it is used to define the software methods and standards for converting analog signals into digital signals and, in the context of digital images and videos, implementing data compression (or *source coding*, often used as a synonym).

There exist two types of data compression codecs: lossless methods, which preserve all the starting information, and lossy compression algorithms, which tend to maximize the storage and bandwidth savings at the cost of quality (thus, information) loss while still achieving good fidelity with respect to the original video[3]. A benchmark study performed by Dmitriy Vatolin et al. [15] in 2007 has estimated the lossless codecs compression rate to be in the 5 to 12 interval, while lossy methods usually have a compression factor of 20-200.

The rationale for preserving information while performing video compression is the high temporal correlation among consecutive frames. For instance, a standard rate of 25 frames per second signifies having one still image every 40 milliseconds: therefore, there exists a non-negligible redundancy since few objects can significantly move in a real-world scenario in such a short time.

The main technique that follows this principle is Motion Compensation, which has as its first step the computing of the differences between consecutive frames, that is, the brightness variation in each pixel for each color channel. With this coding mechanism, a video can be transformed into a stream of a limited number of full frames (which are called "reference frames"): all the frames in between can be

[3]The constraint to comply with, depending on the specific application, usually can be a quality index of the compressed video or a bandwidth/storage threshold [14].

obtained using the encoded MC information, transforming a single object and/or the entire scene depending on the object and camera movement.

In the following figures, it is reported an example of the Motion Compensation method applied on a frame from the 2006 Dutch computer-animated movie "Elephants Dream", taken from [16]: the Mean Absolute Differences (MD) between two consecutive frames are computed in Fig. 1.5, and a right-shift by 2 pixels is then applied in Fig. 1.6 in order to take into account the camera movement. The significantly less amount of codified information in the latter frame with respect to the original frame in Fig. 1.4 is self-evident.

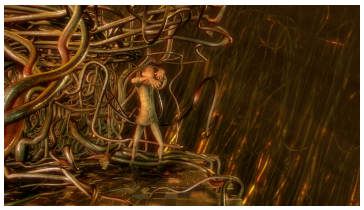


Figure 1.4: The original frame.

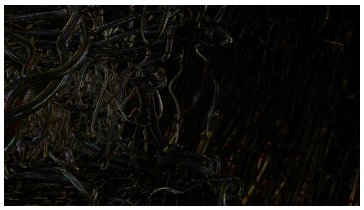


Figure 1.5: Differences between the original frame and the successive frame.

Figure 1.6: Camera-compensated differences.

The most used video codecs, like MPEG [17] and H.26x [18] standards, encapsulates Motion Compensation into Discrete Cosine Transform (DCT) coding, which is a transformation technique similar to the Discrete Fourier Transform that formulates a finite sequence of data points as a sum of cosine functions at different frequencies [19]. Moreover, in this representation each frame is usually segmented into macroblocks, typically consisting of 8×8 or 16×16 pixels, and individual predictions and specific handling can occur sub-partitioning by this means the picture as opposed to treating each frame as a whole.

The most recent extension has been the HEVC standard (High-Efficiency Video Coding) [20] which achieves the best performance in several quality metrics according to De Cock et al. in [21]. As shown in Fig. 1.7, which represents a schema of the HEVC codec standard functioning, the amount of signal processing implemented is very significant and requires higher-than-ever compressing capabilities with respect to previous standards. Nevertheless, by additionally adopting larger Coding Tree Unit (CTU) sizes[4] up to 64×64 pixels, overall higher coding efficiency and a reduced decoding time have been reached compared to previous standards, including H.264/MPEG-4 Advanced Video Coding [22].

[4]The HEVC equivalents of the macroblock units used in the antecedent codecs.

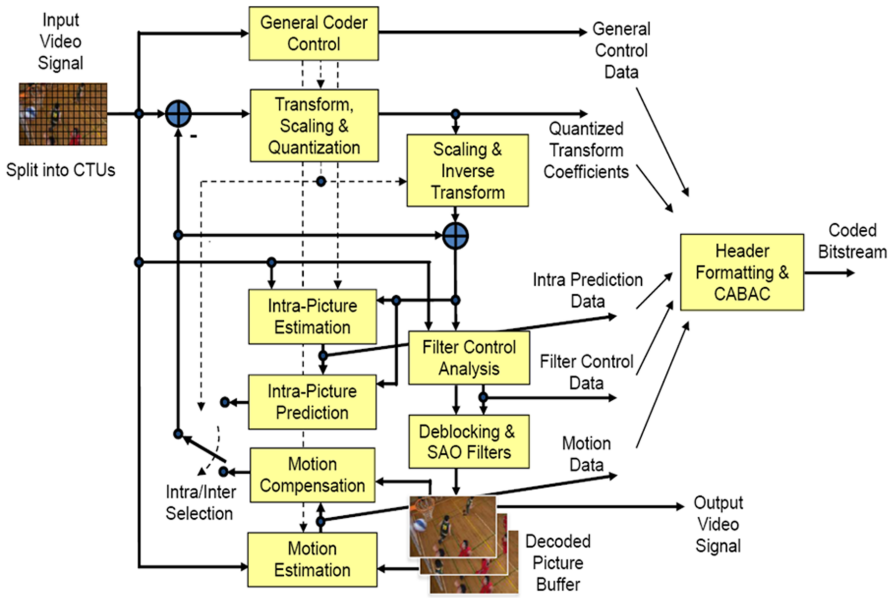


Figure 1.7: The block diagram functioning of HEVC, as illustrated in [23].

1.2 Computer Vision

Starting from the early 70s, there has been an increasing interest in the computer science world in mimicking the visual perception ability of human beings, with digital image processing aptly considered to be the cornerstone for endowing robots with the necessary 3D scene understanding for more ambitious higher-level reasoning starting from a 2-D picture [24].

A primal development has been inventing automated feature extraction methods, which seek the intrinsic and invariant characteristics of the relevant objects in the image: the attributes they have to possess are to be ideally informative, distinct, and accurate. These methods could be used stand-alone or as a preliminary step to reduce the input data, filtering out unproductive binary information while preserving the global structure of the image for successive tasks.

The first typology of features is represented by the so-called *key-point features* or *interest points*, which are a set of points or small patches of pixels usually sparse in the entire image yet robust for correspondences matching in the case of orientation changes, occlusions, and scale variance. It is useful to note that in literature, the term *corner* is used alike to express such features if identified as the intersection of two edges, and redundancy has to be promoted to cope with the possibility of noisy errors affecting the recognition task.

Some algorithms that perform such operations are Moravec's corner detection algorithm [25], Harris and Stephens' [26], and FAST (Features from Accelerated

Segment Test) [27]. In particular, the latter is of particular significance since it has been traditionally used for real-time video processing applications due to its high-speed performance, hence the name: it applies the Bresenham's Midpoint Circle Algorithm [28] with a fixed radius of 3 to compare the brightness level of the midpoint i (the center of the circle) with the circumference points, thus classifying the former as a corner if a certain amount of contiguous pixels in the circle are all brighter or all darker than $|\mathrm{DN}_i \pm threshold|$.

A more semantic typology comprises the full extent of edges and curves, which are usually characterized by abrupt brightness and/or color changes. As a matter of fact, it is proven that, under general conditions [29], discontinuities in brightness values correspond to one or a combination of variations of illumination, depth, surface orientation, and material properties changes. Nevertheless, the complexity of the real world and the challenging working conditions under which digital images are often taken, render this problem non-trivial. Single edgels[5] could be linked together into chains, forming lines and contours, with low-strength contour segments possibly filtered out depending on a defined threshold. The prevalent state-of-art edge detector, since its introduction in 1986, is John F. Canny's algorithm [30], reported with an example in Fig. 1.8. It consists of four main steps: the noise reduction via a Gaussian filter which smooths the image; the computation via the Sobel–Feldman operator [31] of the edge gradient and edge direction for each pixel in the blurred image; a non-maximum suppression step to remove any erroneous pixel classified as edge and finally a hysteresis threshold with two tunable filtering values to discard "weak edges" and noise pixels.

Other feature extraction techniques that adopt differing distinct approaches include the Hough transform, Blob Detectors, Affine Region Detectors such as Harris or Hessian [32], and Ridge Detectors.

At the turn of the 20^{th} and 21^{st} centuries, some of these techniques have been implemented in computer vision algorithms able to not only detect relevant features from an image but also of extracting the feature descriptors, namely vectors encoding a distinguishable numerical fingerprint of that feature. Finally, the act of matching two or more feature descriptors mined from distinct images can be performed in order to achieve multifaceted tasks like object recognition, robot localization, 3D scene modeling, and image stitching. Two notable algorithms of such class are SIFT (Scale-invariant Feature Transform) and SURF (Speeded Up Robust Features) [33][34], whose computing is parallelizable and thereby suitable for modern multi-core CPUs and GPUs systems. A relevant application of the latter in the field of surveillance video monitoring, among the others, has been [35] by Jiang et al. which employs the SURF algorithm for annotating the color and

[5] i.e., pixels in an image recognized as an edge.

(a) Original image **(b)** Grayscale **(c)** Canny's Algorithm

Figure 1.8: An application of the Canny Edge Detector in a highly complex real-world scenario, such as a public park: the hysteresis threshold parameters can be tuned to filter out less or more information; in this case, the `Python OpenCV` library with sample filtering values has been used.

texture features of the fire in video footage for the construction of a real-time fire detection method.

The more recent rise of Machine Learning first, and then Deep Learning, has transformed to a great extent and stretched the boundaries of the Computer Vision research field towards new frontiers.

1.3 Machine Learning

Machine Learning is a branch of Computer Science and Artificial Intelligence (AI), with the latter being the study of techniques that enables computers, machines, or agents to mimic, reproduce or outperform human behavior and decision-making to solve tasks independently or with minimal human intervention [36].

The term Machine Learning was coined by Arthur Samuel, and the first formal definition, credited to him[6], is the following:

[6]The reported definition is generally [37][38] cited as from his 1959 paper "Some studies in Machine Learning using the game of checkers" [39], although it is not explicitly written in the aforecited work. The closest quotes are considered to be *"A computer can be programmed so that it will learn to play a better game of checkers than can be played by the person who wrote the program"* and *"Programming computers to learn from experience should eventually eliminate the*

> Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed.

A more rigorous definition is due to Tom Mitchell in [40]:

> A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

The vast range of applications and the diverse techniques to implement Machine Learning, accounting for the most diversified tasks, hinder the possibility of giving a thorough overview of the topic in these pages; nevertheless, a common stylized pattern of functioning can be detected and drawn from all of them. The core way most of these methods and algorithms learn is via sample data referred to as *training data*: in general, given a specific assignment, the objective is to have the computer (machine) extract automatically the algorithm to solve it. An intelligent system, capable of learning, adapting to changing conditions, and nonetheless being able to perform the prefixed task, divests the need for the system designer to foresee and provide solutions for all possible situations [41].

Formally [42], three are considered to be the main approaches of Machine Learning:

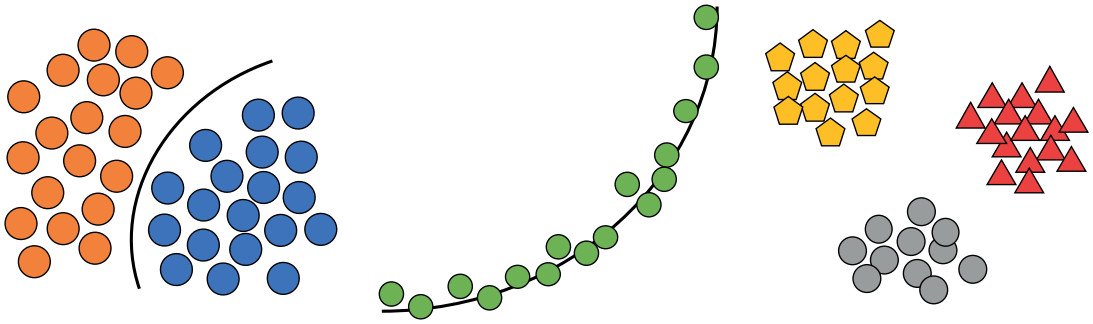
- Supervised Learning: The training data is labeled, i.e., all the samples from the data source have already been assigned to them the correct classification; the model learns having the ground truth as a reference to compare the predicted output with, to the point of generalizing the acquired knowledge.
- Unsupervised Learning: The training data is not labeled; therefore, the model tries to learn hidden patterns in the input data, clustering together datasets with shared attributes or discovering connections based on association rules.
- Reinforcement Learning: The agent learns via trial-and-error the policy, potentially optimal or sub-optimal, that maximizes the total rewards assigned via a suitable policy function designed by the programmer to encourage correct decisions; the main challenge is to find the best trade-off between the *exploitation* of effective strategies and the *exploration* of new actions.

A simple example of a supervised Machine Learning model is a *predictor*, which is a function that produces an output when given a vector of features as inputs: the objective of the learning phase of the model, in this case, is to have a good

---

need for much of this detailed programming effort". Nevertheless, Dr. Arthur Samuel is widely regarded as the father of ML.

performance according to suitable metrics on unseen data. This training phase is also referred to as *parameters estimation* phase; nevertheless, it is often the case that some characteristics such as the number of components or some training configuration variables have to be manually fixed by the programmer at the beginning of the training. These are called the "hyper-parameters" of the model and are usually set by the rule of thumb, via trial and error or determined by an optimization routine (*hyper-parameter tuning*). In Fig. 1.9, three main ML applications are reported.



(a) Binary Classification **(b)** Regression **(c)** Clustering

Figure 1.9: Graphical representations of three ML problems: a) and b) are two forms of supervised learning, whereas c) is an example of unsupervised learning.

An important additional technique that aims to alleviate the hand-labeling process is the Weakly Supervised Learning or Semi-Supervised Learning [43], which is a hybrid of the aforementioned ones: according to it, noisy sources and low-quality annotations are used in the training sessions, which resembles a Supervised setting. As a matter of fact, in various applications, a strong and robust predictive model can still be obtained, notwithstanding the use of imperfect labels. By this means, it can be avoided having as a bottleneck the need for precisely annotated data points, which could depend on either insufficient domain experts' availability, time constraints for a proper dataset collection, or as a trade-off choice between larger quantities of data of lower quality [44].

1.3.1 Artificial Neural Networks

The family of Artificial Neural Networks, ANNs for short, groups several variants of ML algorithms of particular interest for many applications.

The development foundation of ANNs is the functioning, at a conceptual level, of the human brain (hence the term *neural*). Avoiding excessive specifics, unnecessary

for the purpose of this discussion, the processing abilities of the brain mainly lie in the connecting pattern of neurons, which are its small computational units. A brain is composed of a very large ($\approx 10^{11}$) number of them, approximately the same number of stars in the Milky Way, all operating in parallel: each one is connected via *synapses*, branched out from its unique *axon*, to roughly 10^4 other neurons.

The artificial analog of a biological neuron is a summing junction $\sum$ of several synaptical inputs x_i: the values are modeled by individual weights w_i and the result of the addition supplies an activation node, in its simplest and earliest implementation [45] a threshold logic function with two possible outputs (0,1).

As in the real neurons the synaptic strengths may be modified in order to adapt the behavior of each neuron in response to a particular stimulus input, in an artificial neuron (or *perceptron*) the experience, or training, could modify the weight values [46].

A supplementary term that is generally present is the bias b, which is a value added to the node and acts as a shift for the activation function, which more commonly is a non-linear function such as a sigmoid or a hyperbolic tangent: the non-linearity is needed in order to access to a vaster set of possible functional solutions, referred to as *hypothesis space*; otherwise, the neurons could only learn linear (or *affine*) transformations of the input data. In Fig. 1.10 a standard visualization of both neurons is provided, whereas in Fig. 1.11 some of the most widespread activation functions are represented.

In an Artificial Neural Network, a number ranging from a few to thousands of these nodes are arranged in a layered structure, forming a directed acyclic graph: the layer (or layers, if in greater numbers) other than the input and the output is called *hidden layer*, since it produces intermediate results which are not directly visible as an output of the model. These networks could extend in depth as the number of input transformations (thus, the number i of hidden layers) grows: since the information flows from the input x to the output y through the intermediate computations $f^{(i)}$, with no feedback connections (i.e., no cycles or loops are present), they are referred to as *Feedforward Neural Networks*.

The training phase is modeled as an optimization problem, namely finding the set of parameters for every connection that minimizes a defined loss function on the input data, or *training set*: one algorithm that is commonly adopted is the stochastic gradient descent or SGD [48]. As a matter of fact, although the inputs are propagated from the first to the output layer flowing in a *forward pass*, as the network error signal L or the scalar cost $J(\theta)$ is computed, the information flows backward in order to iteratively compute the gradient (i.e., the derivative of the L-function with respect to the weights and biases of the network) with the aim of finding a local minimum of the function. Most Neural Networks use the back-propagation algorithm [49], which relies on the chain rule already introduced by Leibniz in 1676, to efficiently calculate the gradient since numerically evaluating

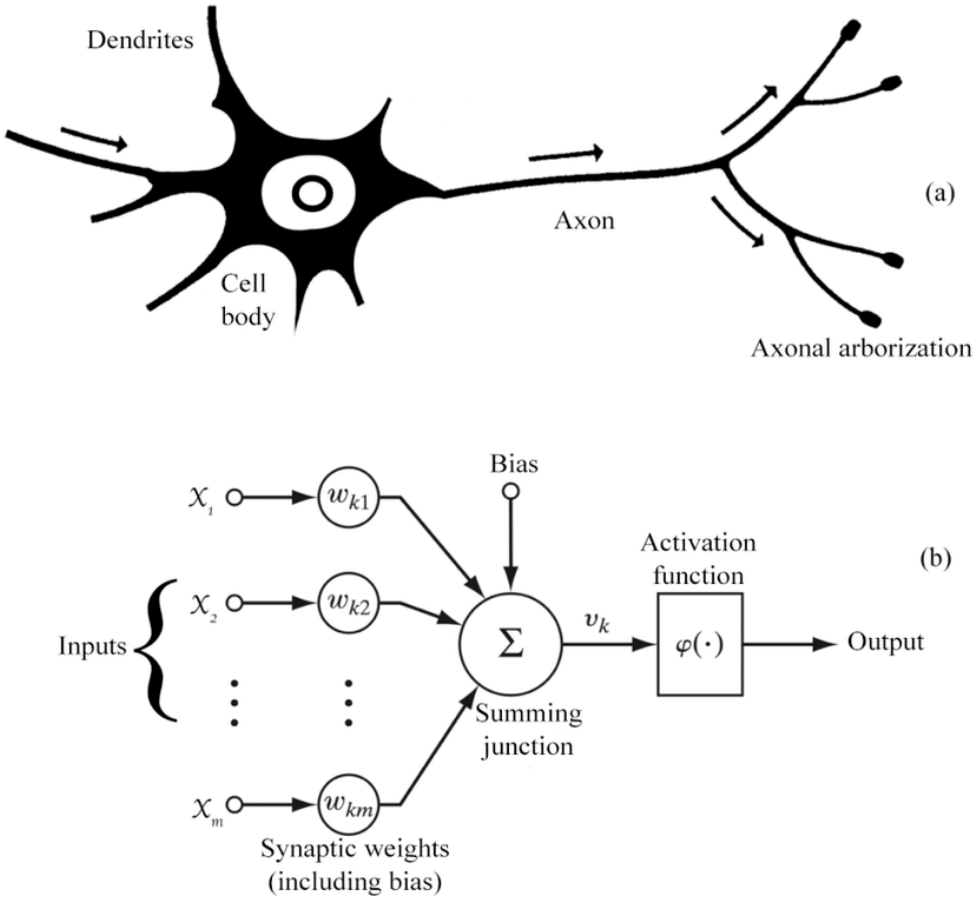
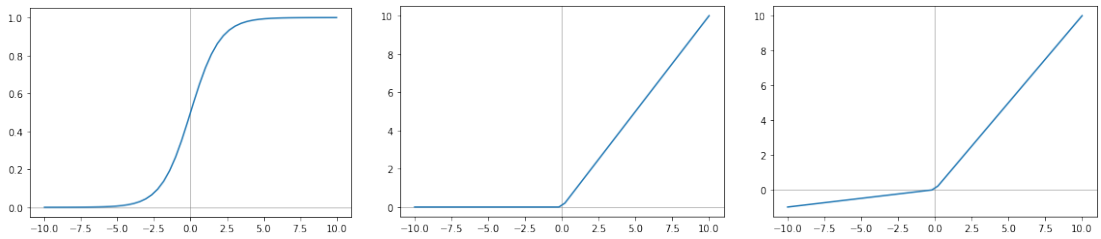


Figure 1.10: A representation of a biological neuron (a) and an artificial neuron (b), as illustrated in [47].

its analytical expression could be computationally very expensive.

For the assessment of a model's performance, the most well-established guidelines especially in the field of supervised learning provide to evaluate the network on out-of-sample data, that is, input similar to the training data yet left excluded from the training samples: some common practices include building an ad-hoc *test set* on which to evaluate the selected accuracy metric, and the k-fold cross-validation technique [50]. In fact, the challenging goal of the training phase ultimately is making the model able to generalize the acquired knowledge to new data, and the best performing model is the one having the lowest *generalization error* (i.e., the lowest expected value of the error on a new input, with the expectation usually taken across the samples in the test set).

Other quality measures may also be extremely relevant, depending on the specific task and on the application usability, like computational resources, the average



(a) Sigmoid

$$S_{(x)} = \frac{1}{1 + e^{-x}}$$

(b) Rectified Linear Unit

$$ReLU_{(x)} = max(0, x)$$

(c) Leaky ReLU

$$LReLU_{(x)} = max(0.1x, x)$$

Figure 1.11: Graphics and formulas of three sample activation functions.

cost for prediction, memory usage, and the interpretability of the results and of the model, with a suitable trade-off among those insights chosen by the business.

1.3.2 Deep Neural Networks

A Deep Neural Network is a particular type of ANNs characterized by many hidden layers so that it progressively extracts higher-level features from the raw input in an automatic fashion called *feature learning* [51]; the word "deep" emphasizes the more extensive depth of the architecture, in contrast with the *shallowness* of preceding ANNs, and it is applied as a descriptive adjective for the entire set of techniques and methods, referred to as *Deep Learning*, which follows this paradigm.

Furthermore, neurons inside a DNN layer are often more complex or use more advanced operations (e.g., convolutions, multiple activations, etc.) with respect to the schematization in Fig. 1.10; therefore, these models are usually employed with success in high-dimensional data input problems, such as those involving speech, images, audio data, and videos, where they have proven to outperform previous ML algorithms [48], or those in which the handcrafted feature engineering process results too much time-consuming.

The peculiarities of Machine Learning and Deep Learning are schematized and compared with the classic explicit programming in Fig. 1.12 with particular emphasis on the feature extraction phase, which in Regular Machine Learning is handcrafted via feature extraction techniques within an application-specific engineering process.

Albeit a deeper and more elaborate architecture renders the model flexible enough to the point of capturing non-linear regularities and relevant patterns for the learning task, which could be of extreme intricacy, an adverse effect of having

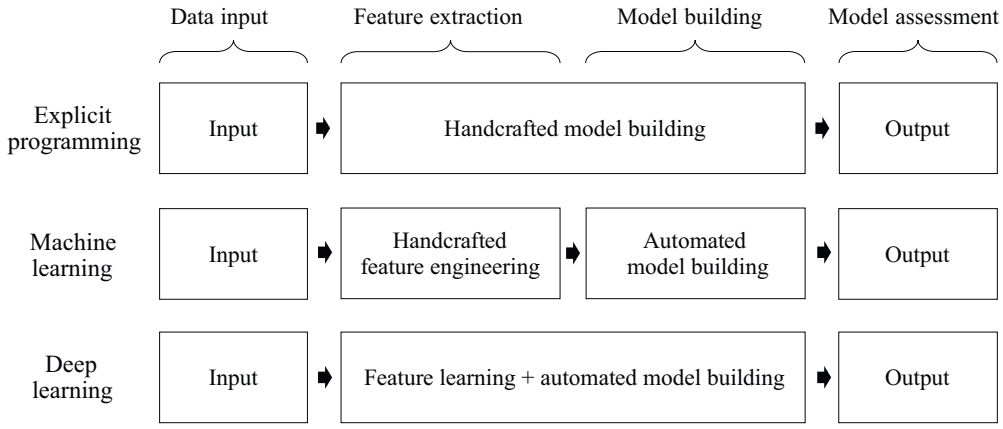
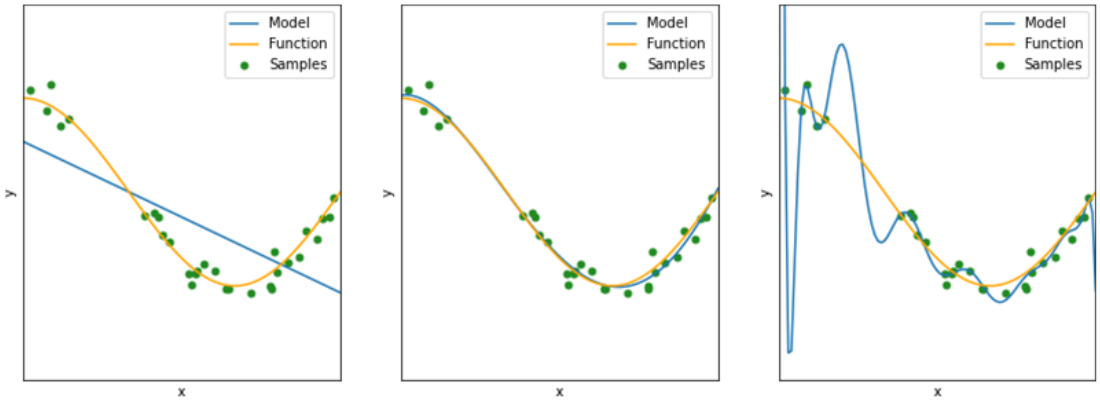


Figure 1.12: The process of a model building according to three different programming methods, as shown in [40]; here, the term Machine Learning has to be interpreted as pre-Deep Learning ML, sometimes also referred to as Classic, Regular, or Shallow ML.

more complex models is the increased risk of overshooting the model's capacity, that is, providing it with a vaster than needed hypothesis space. This may lead to the model adhering too precisely to the training samples, and on the noise and particular representations they have, rather than being able to properly generalize on unseen data, a flawed behavior referred to as *overfitting*, obtainable and to be avoided in general in all Machine Learning applications. In Fig. 1.13 a visualization of this problem is provided, together with its opposite, *underfitting*, that is a model too simple to capture the data structure: without loss of generality, a simple polynomial function has been chosen as the underlying function of a regression problem in order to have a convenient interpretation of the matter.

While acting directly on some training parameters (e.g., training time or full dataset passes) or reducing the data samples may help to avoid overfitting, another possibility, often more in tune, is to act on the hypothesis space of the model by forcibly reducing it during the training phase: this process is called "regularization", and encompasses any modification made on the algorithm that is intended to reduce its test error even at the cost of slightly worsening the training error [48]. A simple but effective method is dropping some nodes for one or more layers, randomly selected at every training iteration with an adjustable probability p, together with their connections, thus creating a child network with fewer parameters in order to prevent co-adaptation and improve the model's generalization. In Fig. 1.14 an example of a simple Deep Neural Network adopting this *dropout* technique is represented.



(a) Underfitting **(b)** Fitting **(c)** Overfitting

Figure 1.13: In blue, the fit of three regression models to a training set (green) obtained by a polynomial function (orange) with added random statistical noise: while a) represents a simple linear function unable to grasp its curvature, c) adheres exactly to the data points and has a smaller mean absolute error with respect to b), which however reproduces far better the real polynomial. The code used for these graphs has been readapted from the `Python Scikit-learn` library.

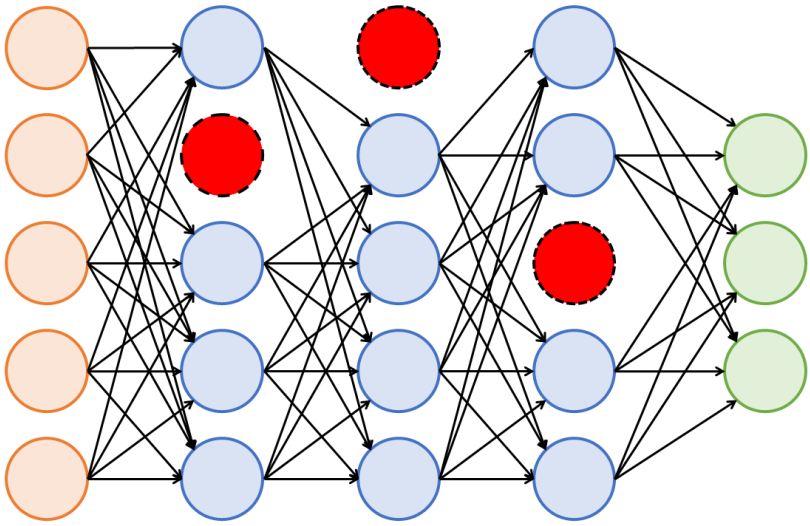


Figure 1.14: A Deep Neural Network with three hidden layers (blue): the unconnected nodes in red with dashed contour indicate a dropout rate of 20% in each layer. No dropout is applied on the input layer (orange) and on the output (green).

1.3.3 Convolutional Neural Networks

Neural Networks are proven to perform well in several fields: however, high-dimensional data inputs still represent a significant problem when accounting for the computational resources available. According to the notation in Sec. 1.1, an image of size $200 \times 200 \times 3$ would lead to neurons each having 120.000 weights to estimate, with the total number of model parameters to train quickly adding up as the number of hidden layers grows. Moreover, the fully-connectivity pattern of the standard hidden layers (i.e., each neuron in one layer is connected to all neurons in the subsequent layer) may often result problematic even if regularization is applied. The advent of Deep and Feature Learning has greatly affected this research field, with the triumph of the newly proposed AlexNet architecture in the 2012 ILSVRC[7] being usually considered the fuel of this revolution [52].

AlexNet is a Convolutional Neural Network, i.e. a typology of an ANN that presents in at least one of its hidden layers the mathematical operation of convolution in place of general matrix multiplication: in particular, AlexNet is characterized by having 5 of these layers.

The convolution, from a mathematical point of view, is an integral that blends together two functions, by expressing the amount of overlap of one function g as it is shifted over a function f, and it is indicated with the expression $f * g$ or $f \circledast g$. In the context of a Convolutional Neural Network, a convolution consists of the multiplication of a relatively small two-dimensional array of weights, called *kernel*[8], with the input, that is for the input layer the image tensor.

The result of this sliding sequence of dot-products is referred to as *activation map* or *feature map* and it is itself a two-dimensional array due to the fact that the filter is systematically applied multiple times to the input array. The size of the activation map is given, in the simplified case of one-channel square matrix input, by the following dimensional formula:

$$[(W+P)\times(W+P)]*[K\times K] = [\frac{(W-K+2P)}{S}+1]\times[\frac{(W-K+2P)}{S}+1] \quad (1.2)$$

where W is the input width and height, K is the kernel dimension, n_c the number of channels, P is the padding (i.e., the number of zero-pixels added at the contours of the image) and S is the stride (i.e., the step-size shift of the filter over the input matrix). An example of this computation is illustrated in Fig. 1.15.

In a more precise and realistic application, the input image has a number n_c of color channels, and, accordingly, the filter must have the same depth as the input;

[7] ImageNet Large Scale Visual Recognition Challenge

[8] Or *filter*.

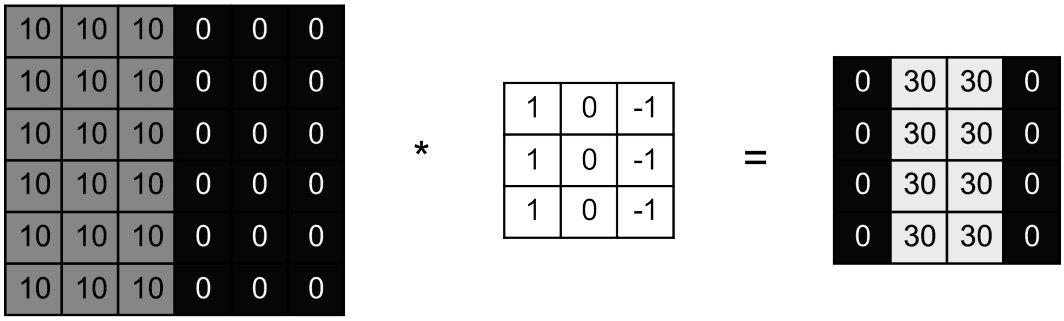


Figure 1.15: A trivial example of edge detection in a $6 \times 6 \times 1$ image using a 3×3 handcrafted kernel for vertical lines detection with no padding: the brightness difference (e.g., a contour of an object) in the left image is magnified in the resulting $4 \times 4 \times 1$ feature map.

furthermore, it has to be noted that, once the first convolutional layer is applied to the input image, the subsequent layers would have as input an intermediate activation map. This process leads to a hierarchical decomposition of the input image, in full compliance with the feature learning paradigm: the convolutional layer n°1 acts on raw pixel values, identifying low-level features such as edges and corners, whereas the successive layers learn the more and more complex high-level features of the image to the point of entire meaningful parts, relevant textures, shapes, and finally objects [53].

To date, the most commonly used activation function in CNN models is the ReLU [54] since it is simple to implement, less susceptible to the problem of vanishing gradients (which prevents deep models from properly learning), and has proven effective in a variety of applications.

The learning phase of the network consists in finding the values for every filter that activate the detection of some features at some spatial position in the input to the point of minimizing the defined loss function of the specific task to perform, and it is usually performed by processing the extensive input data parted in "batches" (i.e., the number of samples processed before the update of the model weights) over several "epochs" (i.e., the number of passes over the whole dataset), with each epoch consisting of a number of iterations equal to the ratio between the number of the data samples and the batch size.

The building blocks of a CNN include two other types of layers:

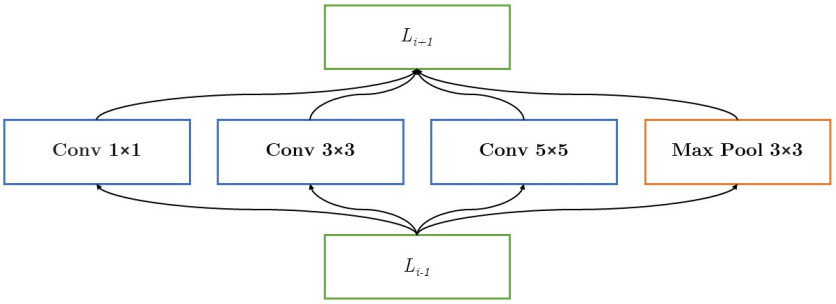
- Pooling layers: they perform a down-sampling of the activation map by summarizing and combining the outputs of clusters of neurons into a single neuron in the subsequent layer; the two common types are *average pooling*, which uses the average value of each cluster, and *max pooling*, which takes the maximum value. No learnable parameters are associated with this layer typology since it has solely a dimensionality reduction purpose.

- Fully Connected layers: in the context of a CNN, they are also referred to as Dense, and they usually are inserted at the end of the network, taking the activations as inputs, flattened in a 1-D vector, and generating a n elements-long vector as output. With a proper design of a sequence of FC layers and using a *softmax* activation function, the final output of the network is a set of class probabilities for solving a classification problem.

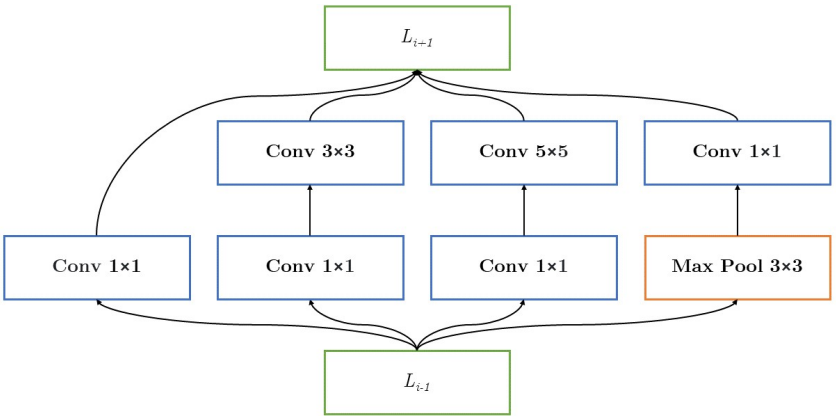
In this respect, a fundamental contribution has been made in 2014 by Christian Szegedy et al. [55], with the introduction of the Inception block, named after the famous Christopher Nolan movie: since the information in an activation map at a certain layer i may be clustered in informative patches of various sizes, they proposed a new architectural module which consists of several convolutional layers characterized by filters of various dimensions (namely 1×1, 3×3, and 5×5) in order to cover all the clusters, with their outputs concatenated together with the output of a pooling layer in order to form a single output vector, which would be the input map of the layer $i + 1$. This *naive* implementation (Fig. 1.16a), which allows the increasing of the number of units at each stage (in *width* instead of in-depth), has been improved afterwards by applying the 1×1 convolution before the larger ones, and also after the pooling operation (see Fig. 1.16b): this greatly reduces the dimensionality of the problem, resulting in improved performance, memory usage, and speed.

Finally, the same authors in a successive work [56] presented several improvements on this architecture, including the factorization of computationally complex convolutions ($n \times n$) into the combination of several simpler ones ($1 \times n$ followed by a $n \times 1$), as shown in 1.16c, with a computational cost saving proved to increase almost exponentially to n.

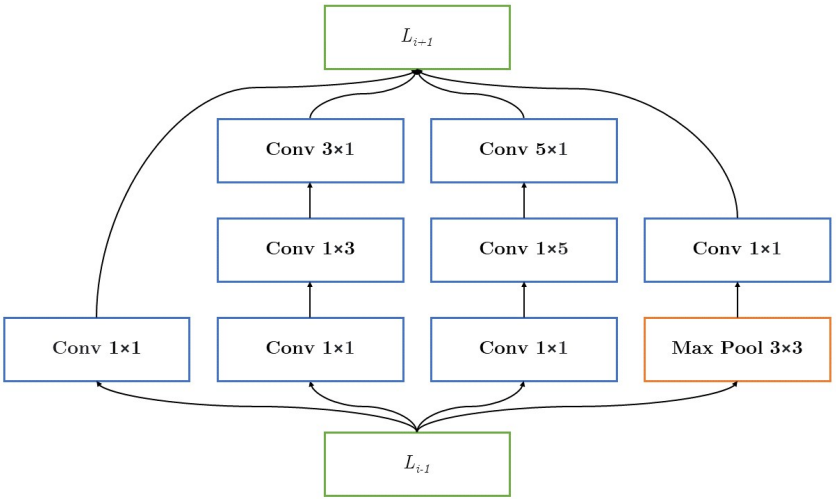
To date, CNN models are considered to be the de-facto standard for addressing several computer vision problems, especially those whose starting inputs are digital images, such as object classification, facial recognition, object detection, image captioning, and object segmentation; in particular, among the best performing architectures on benchmark datasets in the object classification field, several adopt the Inception module or some variants of it, like Xception and InceptionResNetV2 [57][58].



(a) Naive Inception Block



(b) Inception Block



(c) InceptionV3 Block with 3×3 and 5×5 one-time factorization

Figure 1.16: Graphical visualization of three main implementations of the Inception module, in ascending order of performance and computational savings.

1.3.4 Model Explainability and Ablation Studies

Due to the intrinsic complexity of Deep Neural Networks, an emerging challenge in this research field is how to explain the model functioning in a deeper and more understandable fashion than the mere re-proposition of the functional mechanisms of the network, like the units and the activation functions. Several tools and methods have been over time introduced to aid the unveiling of the *black box* that is the model itself.

A worth-mentioning technique that permits to investigate the causality of a model in its modularity is the so-called "ablation", from the Latin *ablatus* which means "carried away" and has lexical application in the medical field, indicating the surgical removal of a biological structure or functionality from the body. The core idea is to remove one or more component(s) of a Deep Learning network at a time, therefore analyzing the contribution of an individual module to the overall model by studying the performance worsening in the case of its removal [59].

Additionally, in the case of a Convolutional Neural Network, the study of the filters of the convolutional layers is possible. Nevertheless, while the first layer kernels can be visualized immediately and provide direct legibility, due to their function of capturing edges and opponent colors, for the subsequent layers more complex approaches are needed. Among the others, an important approach that delivers a visually attractive understanding of the higher-level feature maps is the deconvolutional network, first proposed by Zeiler et al. in [60].

1.4 Computer Vision Techniques for Videos

There are numerous challenges in computer vision techniques applied to digital videos: a video, first and foremost, is a collection of frames in which the temporal order is an essential informational component, thus a serious source of difficulties is finding how to capture the temporal relationship among the frames and codify it in utilizable features; also the time interval choice for the observation is non-trivial and severely task-dependent. Another main challenge is due to the sheer storage and bandwidth occupation of such a medium, which has made it difficult to operate on many input data and also to produce large-scale benchmarks datasets, commensurable in the number of instances and variety with the available image datasets, in comparison with which for a long time video datasets have appeared dwarfed [61].

In order to cope with the latter problem, several dimensionality reduction techniques could be applied: data compression (Sect. 1.1.3) is one of the most popular methods. Another approach is the smart resolution reduction, which stems from how the human fovea and peripheral vision work: each frame could be represented by an inferior bits amount with a non-uniform quality between the

central patches, where most of the important information is supposed to be present, and the edges of the image, at low resolution. Finally, fps rate reduction may also be considered.

A technique extensively researched has been the keyframe[9] extraction: as a book is divided into chapters, a video can be partitioned into distinct scenes, i.e., semantically coherent parts as far as subjects and environment are concerned; each of these scenes or stories could be summed up by one shot to represent them altogether [62]. This technique, which can be applied with both a supervised and an unsupervised learning approach, has been applied in a variety of applications, including video browsing, that is the summarizing of a video as a sequence of keyframes, video indexing for retrieval systems, video analysis or segmentation [63], and many others.

Furthermore, in the pre-Machine Learning era, many of the standard classifiers have been adopted in the literature to solve problems such as video classification. Among the others, there can be found visual-based approaches with Bayesian Networks in works like [64], Support Vector Machines (SVMs) for genre classification [65], with a feature extraction process of color, shape, and motion, Hidden Markov Models (HMMs) for gesture recognition [66] with key-point frames extraction and modeling of the problem as a Markov process, and Gaussian Mixture Models (GMMs) [67], expressing the unknown probability distribution function as a linear combination of k Gaussian distributions.

1.4.1 Features and Video Descriptors

Over the past few decades, several handcrafted or automatic visual descriptors have been introduced to describe and represent visual characteristics of a video in a similar manner to the digital images feature extraction methods, explicating elementary characteristics such as color and shape, or more complex, and identifying ones like location and motion. Hereunder, a brief illustration of some of them is presented, with some associated examples in the figures from 1.17 to 1.20.

Motion Vector (MV) is mainly used in compression standards and has partially been introduced in Sect. 1.1.3 in the context of Motion Compensation: the differences between consecutive frames can be encoded in vectors that provide an offset that represents the movement of the objects, usually with a granularity of 8×8 or 16×16 pixels corresponding to the macroblocks in which the single frame is partitioned; the process of matching the macroblocks in a video sequence is called Block Matching [68].

Optical Flow (OF) captures the apparent velocities of movement of brightness

[9]Shortcut name for key-point frames.

patterns between two successive frames, caused by the movement of the object or the camera: the underlying assumption is that the brightness of a moving pixel has to remain constant over time [69]. Similar in concept to MV, and sometimes used interchangeably[10], is in point of fact a more complex and denser descriptor, a motion field describing the deformation of all pixels from a shot to the corresponding locations in the second image.

Whilst MV and OF are representations of *where* the motion happened, Motion History Images (MHIs), introduced by Davis and Bobick in [71], represent the *how* of the motion: they are gray-scale images, where dominant motion information is represented in a compact way expressing pixels brightness intensity as a function of the temporal history of the motion. The result is extremely expressive in the case of a still camera, like in surveillance applications, engraving the moving parts of a video sequence (longer with respect to the MV and OF 1-frame window) in a single image; due to its sliding window approach, the history of temporal changes at each pixel location then decays over time, to the point of having a black image in case of no succeeding movement [72].

Feature Trajectories are represented as sequences of log-polar quantized velocities, and they are usually obtained by means of the Kanade–Lucas–Tomasi (KLT) feature tracker or by matching SIFT descriptors between consecutive frames. Wang et al. [73] in 2011 proposed Dense Trajectories (DT), obtained by tracking densely sampled points[11] and using a particular optical flow fields solution called MBH (Motion Boundary Histogram), which represents the gradient of the optical flow, to remove camera motion. Improved DT [74] compensate for camera motion by determining the homography between consecutive images, before computing the optical flow, thus avoiding to codify non relevant information.

1.4.2 Deep Learning Architectures

The rise of Deep Learning and the state of art performances achieved in most of the image analysis tasks have prompted the research for adapting the existing networks or creating new architectures more suitable for image sequences, i.e., videos.

A first approach by Karpathy et al. in [61] has been utilizing a Convolutional Neural Network as a backbone network for extracting features and then fusing the information extracted from consecutive frames to effectively exploit the temporal dimension for an action recognition problem. Several connectivity patterns have

[10]More precisely, it is possible to distinguish "Sparse Optical Flow" for methods as the Lucas-Kanade algorithm, where motion is encoded in pixels blobs like in MV, and "Dense Optical Flow", which is computed on all the points, with the Gunnar Farnebäck's algorithm as the de-facto standard in literature since its introduction [70].

[11]Video blocks sampled at regular positions and scales in space and time.

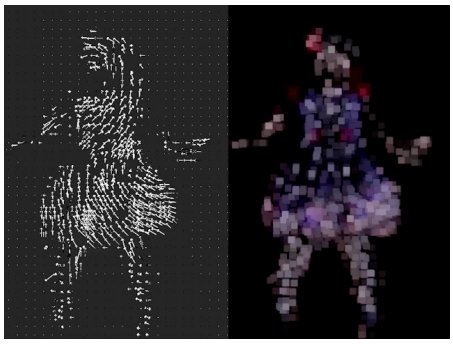


Figure 1.17: Motion Vector of a dancing girl with macro-block visualization, using the `FFmpeg` suite. Images from [75].

Figure 1.18: Dense Optical Flow visualized by the `OpenCV` library with the HSV color scheme: the displacement coordinates are converted into polar coordinates as magnitude (Value) and angle (Hue) for every pixel, while Saturation remains constant. Images from [76].

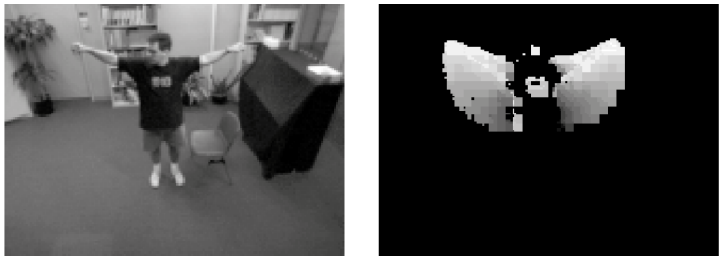


Figure 1.19: Motion History Image of a man waving his arms, as shown in [71].

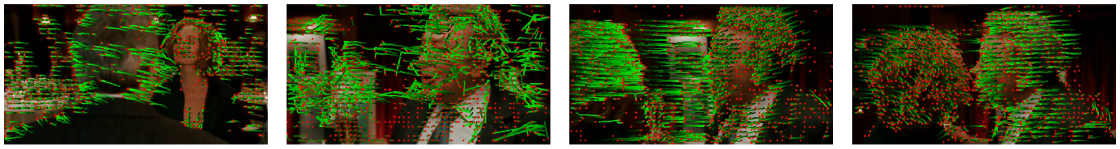


Figure 1.20: Dense Trajectories of frames from the Hollywood2 dataset, reported from [73].

been studied in the aforementioned work: an early fusion, which consists in stacking together a number T of frames, accordingly modifying the filter of the first level to operate with an input of dimension $W \times H \times 3 \times T$; a late fusion, placing two identical single-frame branches in parallel with shared parameters at a distance of N frames apart, merging them in the fist Fully Connected layer of the classifier; a slow fusion, a balanced mix of the two previous patterns, which obtained the best results of the group on the benchmark dataset UCF101 [77] and on the at the time newly proposed video dataset Sports.1M (Fig. 1.21). Nevertheless, the ever-so-slight improvement in the performance metric over the single-frame CNN denoted a sub-optimal and improvable approach.

Figure 1.21: An overview of the Sports.1M Dataset, a dataset of 1 million videos mined by YouTube belonging to a taxonomy of almost 500 classes of sports, introduced by [61].

Another proposal [78] has been an Encoder+Decoder end-to-end-trainable framework, combining a CNN for extracting visual features (Encoder) with a Long Short-Term Memory network (LSTM) for capturing temporal features (Decoder). An LSTM is a special type of a Recurrent Neural Network (RNN), that is an ANN with connections between nodes that form an undirected graph along a temporal dimension in order to process input data sequences, usually text or speech; in an LSTM, each node of the chain has three typologies of *gates* that regulate the flow of information and avoid to incur into the problem of losing long-term dependencies.

An important contribution has been the introduction of 3D ConvNets (C3D), which are an extension in which convolutions and pooling involve all spatio-temporal dimensions and the processing of a video volume produces a volume [79], whereas a standard 2D convolution on a stack of frames produces a 2D output. The main characteristic of the C3D features is that they model appearance and motion simultaneously, and this approach has been applied extensively, also in combination with an RNN [80].

The inspiration for another framework came, yet again, from the human brain, namely the visual cortex: a model of the neural processing of vision distinguishes it in a ventral stream (also known as the "what pathway") for object recognition and visual identification, and a dorsal stream ("where pathway") for spatial processing and visually guided actions [81]. The Two-Stream models implement a similar strategy, naturally decomposing the video into a spatial part, in the form of a single frame appearance, and a temporal component, modeled as a multi-frame OF [82], processing them separately via a two-path Convolutional Neural Network. In Fig. 1.22 a visualization of this architecture is reported.

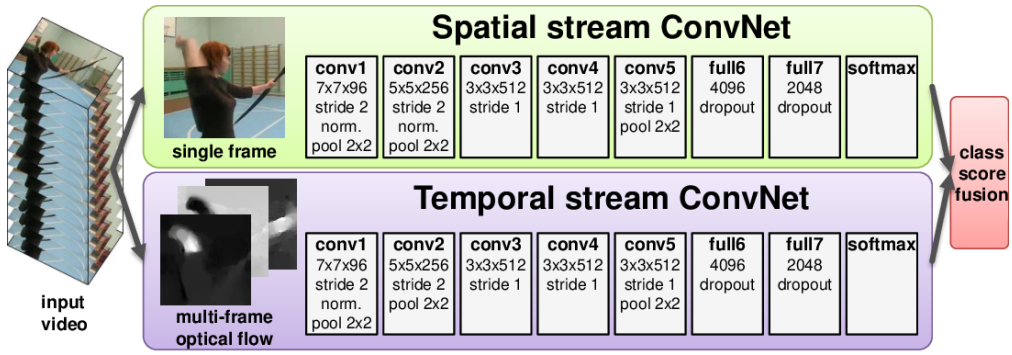


Figure 1.22: Two-Stream model for video classification, as proposed in [82].

Other recent proposals include Temporal Segment Network (TSN) with sparse sampling [83], feature aggregation models such as ActionVLAD [84]; notably, among the others, a Two-Stream architecture with two parallel 3D ConvNet called Inflated 3D, stemming from the aforementioned C3D, has also been proposed in [85].

Altogether, this non-extensive survey demonstrates the burgeoning interest and activity in the research field of Deep Learning applications for videos, and the variety of useful tasks to accomplish has prompted the following project work on a particular, yet of extreme significance, utilization.

Chapter 2

Methods and Techniques

In the following section, a survey on the topic of Anomaly Detection in the context of surveillance videos is presented in order to provide an in-depth guide on the subject.

A rigorous description of the thesis project is given in Sect. 2.2, highlighting the causes and the goals to accomplish, whereas Sect. 2.3 is dedicated to a thoughtful description of the two benchmark datasets used in the work, which are UCF-Crime [86] and RWF-2000 [87]. Due to the different visual descriptors used in the pre-processing of the two datasets, a more meticulous dissertation upon them with respect to the general introduction paragraph is rendered in Sect. 2.4.

The baseline architectures from which this work stems are reported in Sect. 2.5, in order to enlighten the reader with the associated mathematical descriptions of the frameworks and of the adopted loss functions; the discussion on the framework to adopt together with the final problem definition is reserved in Sect. 2.6. Finally, Sect. 2.7 together with the quality metrics for evaluating the achieved performance.

2.1 Literature review on Anomaly Detection

Anomaly Detection in real-life videos is a trending topic in the Computer Science world, due to the increasing interest and demand for public security issues, backed by the more widespread presence of CCTV cameras in the public and private spaces of our cities to the point of being referred to as "surveillance cameras ubiquity" [88]. In Fig. 2.1 the number of publications on this issue is reported for the last decade: the graph shows a distinct and more sloping than linear trend as a consequence of the advent of Deep Learning, with an overall decennial growth rate of 383.9%.

The primal challenge of this task is defining what anomaly is: in contrast with other computer vision tasks, such as object recognition, wherein staple formulations

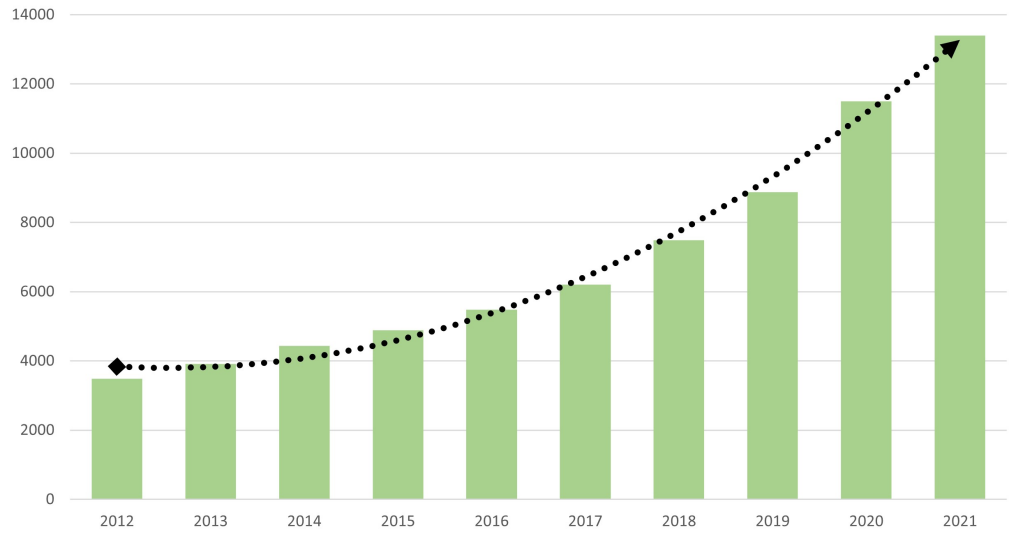


Figure 2.1: Number of publications on the "Anomaly Detection in video" topic published in the Google Scholar database per year in the last decade; in black, a 2^{nd} degree polynomial trend curve is represented, highlighting the steepest increase after the advent of DL.

apply, and no ambiguity is present[1], the boundaries of the definition of "anomalous event" are far more unclear, as well as dependent, in some cases, on the ethical and specific cultural environment. It should be enforced that the just outlined point is not a mere linguistic issue since the modeling of the problem is of paramount importance in the investigation for a framework to address it.

It should be no surprise that an initial method of tackling the issue has been to provide a statistical description, juxtaposing the meanings of *anomalous* and *abnormal*, thus treating the rare events or the outliers [88] as the ones to detect.

Kratz et al. in 2009 [89] proposed, in the context of extremely crowded scenes, a statistical model to compactly represent a video volume by detecting the relationships between local spatio-temporal motion patterns[2], and constructing a distribution-based HMM to capture the temporal relationship between local spatio-temporal motion patterns and a coupled HMM to capture its stationary structure: "standard" motion patterns would model usual events within the scene like people normally walking, so atypical events, like individuals obstructing traffic, reversing

[1]e.g., an object of the class "apple" is definitely an apple and clearly distinguishable from an object of the class "bottle"

[2]i.e., describing motion in terms of local dominant directions of movement using cuboids of predetermined size

walking direction or a sudden absence of people in a part of a frame, were identified as (statistical) anomalies.

Another work in this sense has been [90] by Li et al., who in 2012 proposed a statistical scene modeler founded on Bayesian theory and GMMs for anomaly detection in uncrowded scenes with background subtraction using feature clustering.

The second trod approach is to explicitly represent the events, rendering the model able to distinguish different activities and therefore identify the abnormal event. Chan et al. in 2004 proposed an HMM of this kind [91]: after having defined the objects in the scene, a multiple-states Markov chain was built for describing each and every event of interest based on prior knowledge (in the specific case, cargo load and unload cycles in an airport), enabling the detection of rare (i.e., abnormal) events like, for example, the utilization of an unscheduled route by a trailer cart.

On a similar note, in 2010, an abnormal traffic detector was presented by Sultani et al. [92], which explicitly extracted 3-dimensional patterns of driver behaviors, classifying as abnormal the series of patterns (called *document*) with very low likelihood with respect to a baseline "intelligent behavior", also granting an abnormality localization feature.

With the advent of Deep Learning, a similar twofold method could be identified. Several works proceed by taking an unsupervised (or weakly supervised) learning approach, pointing to making the model able to generalize to a variety of anomalous events with little to minimum supervision [86]. On the other hand, other projects rely on the supervised learning technique, creating more specialized models like violence detectors [87] on par with classification frameworks. A more in-depth analysis of two frameworks representative of both concepts is presented in the following Sec. 2.4.

In parallel with model deployment, a great development has concurrently happened in the production of footage datasets in the anomaly detection field of research, increasingly tackling the specific challenges of such tasks, which include clutter, complicated occlusions, blur due to movement or resolution, and more.

One of the first works in this regard has been the Hockey Fight Detection Dataset by Nieval et al. [93] in 2011, which helped to start the shift of the action recognition community's focus from detecting simple actions like clapping, walking, or jogging, to the detection of aggressive behaviors. It is a set of 1000 sequences divided into two even-numbered groups, Fights and Non-Fights, taken from real-world professional hockey matches in which brawls and altercations occurred: each video clip consists of 50 video frames with resolution $360 \times 288 \times 3$ and a frame rate of 25 fps. In concomitance with the same work, they also produced the Movies Fight Dataset, which accounts for 200 clips with a length of 1.6 to 2 seconds and frame dimension of 720×480, extracted from Hollywood-distributed action movies

containing trimmed acted violent scenes.

Moreover, several Human Activity Recognition datasets were developed and used also for violence detector training. One example has been the SBU Kinect Interaction Dataset [94], introduced in 2012 and containing eight types of two-person interactions, both pacific (like a hug or a handshake) or violent (e.g, a staged karate kick, a pushing or punch): the data was collected using both a stationary camera with a depth map and the Microsoft Kinect sensor. Worth mentioning is also the BEHAVE Video Dataset [95], a multi-person action classification database with bounding box level annotations, which included also the "Fight" label among other behavior types: still, the portion of frames containing (staged) violence was only a small fraction of the total (1751 frames out of 29.196), hindering the training feasibility for a specific violence detection task due to the skewed class representation.

An important contribution has been the BOSS dataset [96], originally developed as part of Eureka's Celtic Initiative "BOSS : On Board Wireless Secured Video Surveillance BOSS" for October 2006 to June 2009. It consists of video and audio recordings by ten surveillance cameras mounted on-board inside a moving train and, in 2017, it was manually annotated by Velastin et al. to distinguish the overall poses of the people (e.g., "standing" or "sitting"), individual actions (e.g., "walking", "laying down") and interactions, like fighting, harassment or helping. Formed with the general aim of improving the security inside public trains, it had as its main limitation the total length of only 27 minutes.

Capitalizing on the ever-extending diffusion of videos online, particularly on YouTube, in the same year Tal Hassner et al. [97] assembled the Crowd Violence Database, for use in both violence classification and violence detection tasks. The distinctive traits of this dataset with respect to the previously available databases are twofold. Firstly, the scenario of violence definition is more challenging and relevant to reality: the "violent activity" is no more an acted scene of violence in the spotlight, or a distinct brawl in a controlled environment like a hockey field, but rather outbreaks of violence by one or more people in crowded scenes which also comprehend spatially unconstrained human motion. Secondly, it captures a wide range of challenging real-world viewing conditions, including uncontrolled, in-the-wild footage, therefore presenting a diversified variety of scene types and video qualities. All the 246 collected videos were resized to 320×240 pixels, with the average length of a video clip being 3.60 seconds. Fig. 2.2 contains some graphical representations of the Crowd Violence Database, together with other aforementioned datasets, to highlight its peculiarities.

Finally, the first baseline CCTV dataset for fight detection, the CCTV-Fights dataset, was presented by Mauricio Perez et al. [98] in 2019: it contains 280 CCTV untrimmed videos characterized by different types of fights, ranging in length from 5 seconds to 12 minutes, for a total of over 8 hours of footage. All the videos were

(a) Hockey Fight Dataset

(b) SBU Kinect Interaction Dataset

(c) Crowd Violence Database

Figure 2.2: Examples of the different violence depictions in three of the described datasets: a) presents brawls and altercations in a controlled environment; b) depicts staged actions in a static setting; c) displays diverse and in-the-wild footage with uncontrolled violent actions usually by more than one person.

annotated at a frame level, i.e., each fight instance in a video was labeled with its specific start and end points: since the exact beginning and end-points of a fight could be to some extent subjective, some good practices and a strict annotation method were adopted in the labeling phase, and a suitable flexibility degree was accounted for in the prediction of the temporal localization.

2.2 Project Description and Goal

The project's scope is to contribute to creating an interface with an associated and automated workflow in order to rationalize the workload of legal personnel exploring documentary evidence in the form of visual footage. The envisaged scenario of the application is a mostly offline environment, without the strict timing constraint of real-time processing: the goal is to provide the user with a large but not definitive projection of video fragments, confidently containing the "abnormal" or "violent" portions in them, starting from a long sequence of untrimmed videos, possibly prompted by the user or automatically extracted from a previous ETL process.

The path followed in the model building has been to replicate two reference studies, one constituting an anomaly detector with an unsupervised learning approach and the second one a supervised violence binary detector. The two have been handpicked among the alternatives to show the differences in the adopted techniques, but also for the important and concomitant contribution of reference large-scale benchmark datasets. In this way, different architectures have been tried and several models have been trained.

Secondly, the model which has been considered as the most promising according to the obtained accuracy on the training set and to the intrinsic functioning with respect to the application has been extensively explored, introducing architecture novelties and exploring the hyperparameters space, in order to better the performance in the study case.

Finally, it has been produced a basic interactive interface that allows the extraction of video portions considered to be "suspicious" from customized video surveillance videos uploaded by the user by implementing the best-performing model trained.

2.3 Datasets

Stemming from the outline presented in Sect. 2.1, in this Section the two datasets deepened in the following work are presented: both can be considered to be among the most advanced and prevalent available databases to date.

2.3.1 UCF-Crime

The UCF-Crime dataset [86] is a large-scale dataset of 128 hours of videos, published in 2019 following a collaboration between the Information Technology University of Pakistan and the University of Central Florida (UCF). It is composed of 1900 untrimmed real-world surveillance videos of varying lengths, representing 13 realistic anomalies being labeled as follows: Abuse, Arrest, Arson, Assault, Burglary, Explosion, Fighting, Road Accident, Robbery, Shooting, Stealing, Shoplifting, and Vandalism. These categories were selected considering the significant impact on public safety, according to the authors, with a number ranging from 50 to 150 videos for each abnormal category.

All the videos have been retrieved via web scraping using text search queries in different languages, and only unedited, real, and fixed-camera videos have been selected, leaving out pranks, edited videos, and also those filmed by a hand-held camera. In Fig. 2.3 an overview of the dataset is presented, with sample frames for each of the labeled anomalies.

The peculiarities of this contribution are several: firstly, the sheer size of the dataset is vastly unparalleled with respect to previous works, with a total length of 128 hours and an average of 7267 frames per video, for a duration circa 8 times greater than the most extensive video dataset for anomaly detection presented beforehand.

Secondly, the variety of the depicted events is of great interest, due to the distributed representation of diverse abnormal activities pertaining to more than a dozen of macro categories. Although an unmitigated presentation of every possible abnormal behavior by one or more people remains still today hardly practicable, this publication proposed a first categorization of various anomalous behaviors: therefore, the UCF-Crime dataset has the twofold intent of allowing the training of an anomaly detector model and of an abnormal activity classificator.

Finally, the labeling procedure has been performed following the principle of weak labeling, enabling a semi-supervised approach in addition to a supervised one. In this direction, the videos constituting the dataset are relatively long and most importantly untrimmed: that is, each video containing one or more abnormal activity depictions is also characterized by normal activity before it and in the aftermath, hence the unprecedented average length of the clips. This has been a deliberate choice by the authors, not only to economize the available time needed for the dataset building thus collecting more videos, but also with the aim of creating a more robust model capable of better generalizing on unseen violent outbreaks. Since in a real-world scenario, the depicted episodes could greatly vary in settings and unfolding, it is of paramount importance to aid the model during the training phase in capturing the intrinsic characteristics of the abnormal activities with respect to normal behavior.

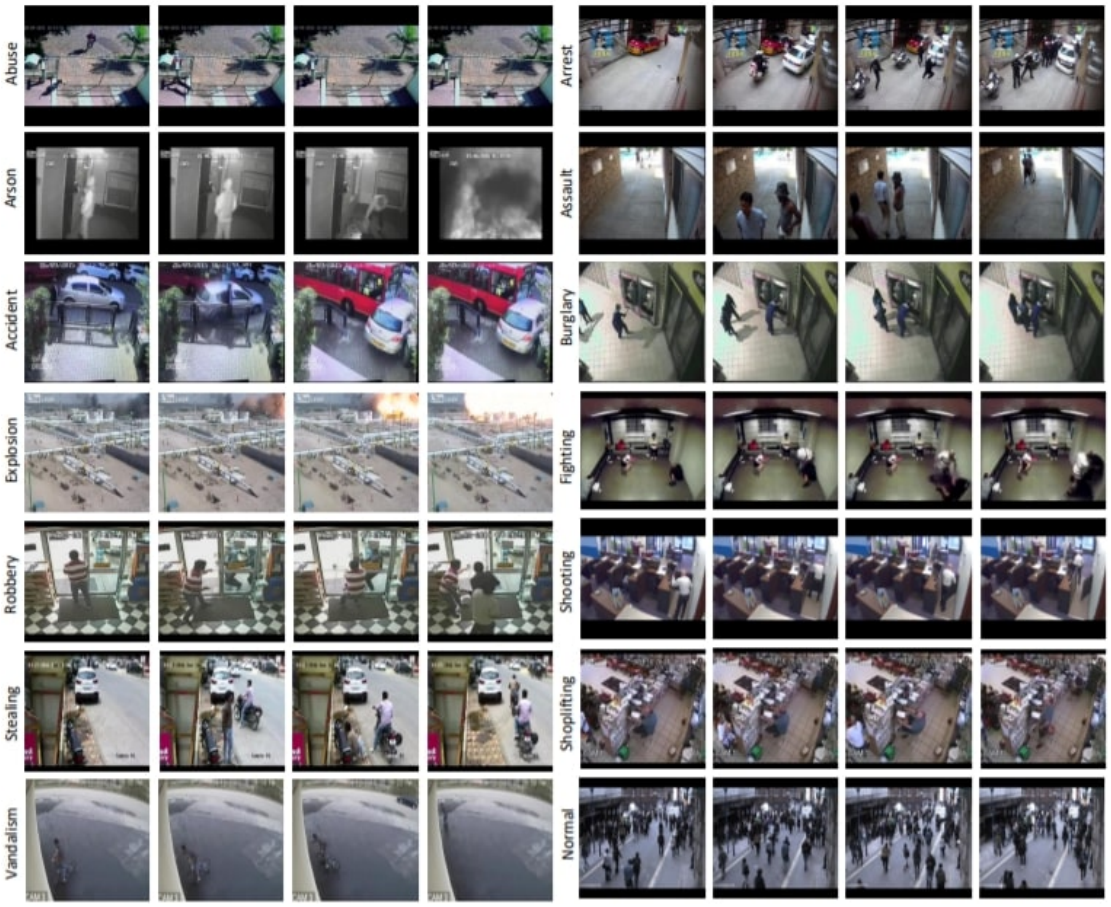


Figure 2.3: Screenshots from the UCF-Crime Dataset.

2.3.2 RWF-2000

RWF-2000 is a dataset containing 2,000 videos captured by surveillance cameras in real-world scenarios, published by [87] for performing violence detection; RWF is an acronym for Real-World Fighting. Similarly to the previously introduced UCF-Crime, it contains only real and unmodified CCTV footage collected from the YouTube platform, using scraping techniques and prompt keywords in several languages.

The only additional edit, according to the authors, has been the trimming of the clips into a constant duration of 5 seconds at 30 FPS: therefore, the total counting is 300.000 frames. Fig. 2.4 shows sample frames from it, whereas Fig. 2.5 represents the resolution distribution of the videos.

Each clip is characterized by a binary label:

- Fight (1): if containing violent abnormal activity; several events, including

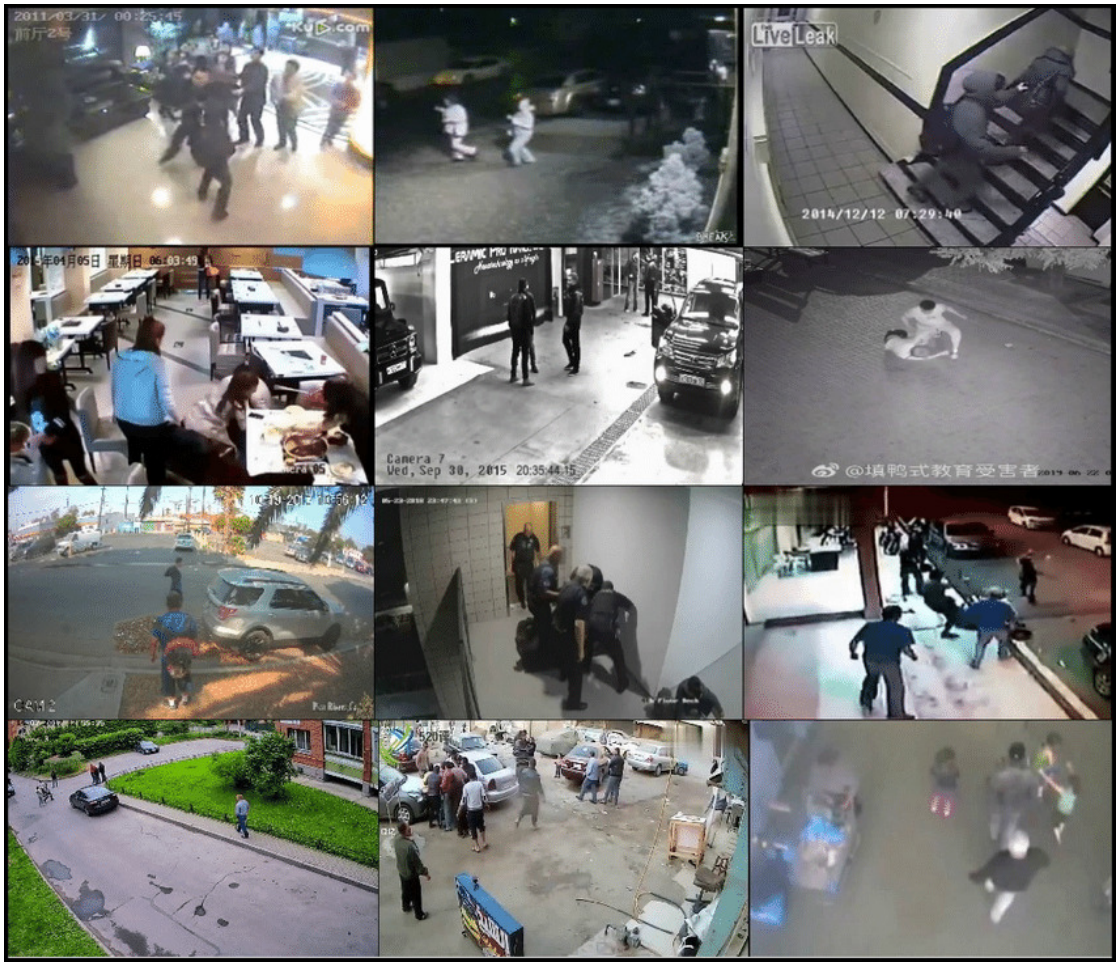


Figure 2.4: Sample overview of the RWF-2000 Dataset.

shooting, robbery, fistfight, explosion, and assault, are depicted, all under the same class.

- Non-Fight (0): otherwise; it can be either a pre-fight scene or a CCTV filming of normal human activity, like people walking on the street or talking in a bar.

The main purpose of this separation is therefore to train a unique anomaly detector with the ability to bundle the various depictions of anomalous activity.

As a matter of fact, a handcrafted distinction in different classes could in some cases prove to be problematic for the labeling crew and for the generalization error of the model, since ambiguity and unsubstantial definitions severely affect this research field. On the other hand, also this precise binary demarcation, apparently simple, could be far from self-evident given the fact that it characterizes altogether

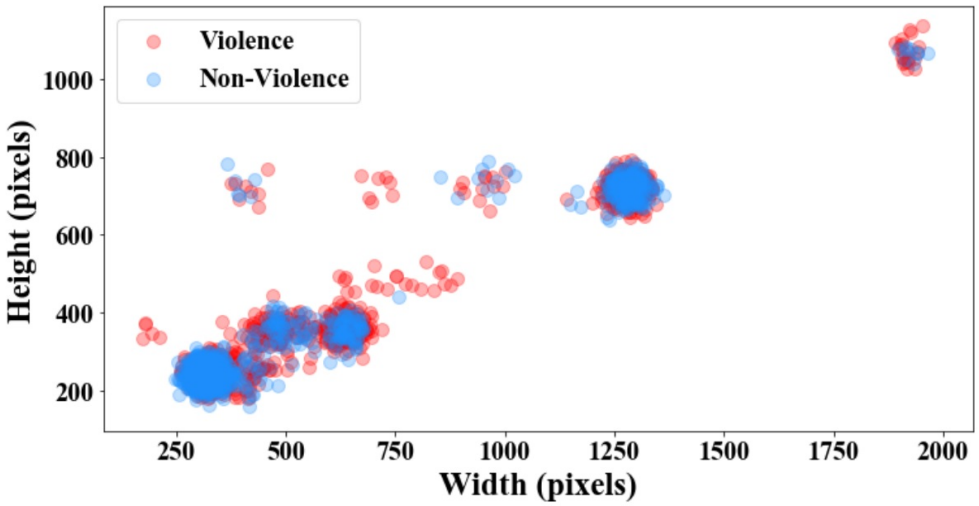


Figure 2.5: Resolution distribution of the videos from the RWF-2000 Dataset: the proportion is balanced, with significant clusters surrounding the standard definition formats of 240P, 320P, 720P, and 1080P; small random noise is added to the original data points in order to obtain a clearer visualization. Image from the original paper.

5 full seconds of actions: a thorough refinement has been necessary for the different phases of the dataset collecting to achieve a cohesive collection.

2.4 Feature Descriptors

2.4.1 Gunnar Farnebäck's Optical Flow

In the context of Gunnar Farnebäck's algorithm for computing the dense optical flow [69], a polynomial expansion[3] refers to the approximation of the neighborhood of a certain pixel with a quadratic polynomial expression of the following type:

$$f(\mathbf{x}) \approx \mathbf{x}^T \mathbf{A} \mathbf{x} + \mathbf{b}^T \mathbf{x} + c \tag{2.1}$$

where $\mathbf{A}$ is a symmetric matrix, $\mathbf{b}$ a vector, and c a scalar: the values are estimated via weighted least squares fit to the signal values in the neighborhood of the pixel. The weights follow two principles: *certainty*, which refers to the exclusion

[3]This dissertation strictly follows the algorithm's description included in the original publication.

of the points in the neighborhood but outside of the image, and *applicability*, which usually determines a Gaussian weighting where the center of the neighborhood has the highest weight.

The modeling of the movement of an object in a video is as an ensemble of affine transformations: assuming that the pixel intensities[4] are constant, an elementary object occupying 1 pixel in a frame characterized by $I(x, y, t)$, that is the pixel intensity of the pixel in position (x, y) at the time t, in the next frame is characterized by $I(x+dx, y+dy, t+dt)$: this hypothesis is referred to as brightness constancy.

Therefore, under an ideal affine translation $\mathbf{d}$ of the signal f_1:

$$f_1(\mathbf{x}) = \mathbf{x}^T \mathbf{A}_1 \mathbf{x} + \mathbf{b}_1^T \mathbf{x} + c_1 \tag{2.2}$$

a new signal f_2 is determined:

$$\begin{aligned} f_2(\mathbf{x}) &= f_1(\mathbf{x} - \mathbf{d}) = (\mathbf{x} - \mathbf{d})^T \mathbf{A}_1 (\mathbf{x} - \mathbf{d}) + \mathbf{b}_1^T (\mathbf{x} - \mathbf{d}) + c_1 \\ &= \mathbf{x}^T \mathbf{A}_1 \mathbf{x} + (\mathbf{b}_1 - 2\mathbf{A}_1 \mathbf{d})^T \mathbf{x} + \mathbf{d}^T \mathbf{A}_1 \mathbf{d} - \mathbf{b}_1^T \mathbf{d} + c_1 \\ &= \mathbf{x}^T \mathbf{A}_2 \mathbf{x} + \mathbf{b}_2^T \mathbf{x} + c_2 \end{aligned} \tag{2.3}$$

In an ideal scenario, the coefficients should equate; however, in a realistic environment with relaxed and less strict assumptions, by defining $\mathbf{A}(x)$ as the average of the two $\mathbf{A}(i)$ matrices and generalizing the displacement $\mathbf{d}$ with the variable field $\mathbf{d}(x)$, we obtain the right-hand side of the following expression to solve:

$$\mathbf{b}_2 = \mathbf{b}_1 - 2\mathbf{A}_1 \mathbf{d} \Longrightarrow \begin{cases} \Delta\mathbf{b}(\mathbf{x}) = -\frac{1}{2} \left(\mathbf{b}_2(\mathbf{x}) - \mathbf{b}_1(\mathbf{x}) \right) \\ \mathbf{A}(\mathrm{x})\mathbf{d}(\mathrm{x}) = \Delta\mathbf{b}(\mathrm{x}) \end{cases} \tag{2.4}$$

Instead of simply solving the second equation point-wise, with the additional assumption that $\mathbf{b}(x)$ is slowly varying, we can include information on the neighborhood I of the pixel weighting its contribution with a suitable vector w; therefore, to the equation of (2.4) we add the following minimization problem:

$$\sum_{\Delta\mathbf{x} \in I} w(\Delta\mathbf{x}) \| \mathbf{A}(\mathbf{x} + \Delta\mathbf{x}) \mathbf{d}(\mathbf{x}) - \Delta\mathbf{b}(\mathbf{x} + \Delta\mathbf{x}) \|^2 \tag{2.5}$$

which has its minimum for the following expression of the displacement field:

$$\mathbf{d}(\mathbf{x}) = \left(\sum w \mathbf{A}^T \mathbf{A} \right)^{-1} \sum w \mathbf{A}^T \boldsymbol{\Delta}\mathbf{b} \tag{2.6}$$

[4]That is, the vector of the intensity for the three color channels

The very assumption of this algorithm is that the polynomial expansions in the two signals are identical except for the displacement field $\mathbf{d}(x)$; however, small errors in the expressions would appear, especially for larger displacements. Therefore, it is useful to incorporate a priori knowledge of the displacement field to have a first estimate of it, and then compute the relative displacement between the estimate and the real value, which should be small. With this additional component, the loop of the algorithm is closed and it can be iterated: at step i, the estimated displacements from the step $i-1$ are used as its a priori displacement knowledge, with $\mathbf{d}(x)$ equal to zero in the first step, and each successive iteration being a better estimate of the displacement vector. The stopping criteria could either be a measurement threshold or a specific number of iterations [99].

Since its introduction, this method has been the reference standard for the computation of the dense optical flow in a digital video. By implementing this procedure in video footage, the final output for every successive frame is a 2-dimensional array with optical flow vectors (u, v) for each pixel of coordinates (x, y) with $u = \frac{dx}{dt}$ and $v = \frac{dy}{dt}$, and each component representing the magnitude of the flow in that direction.

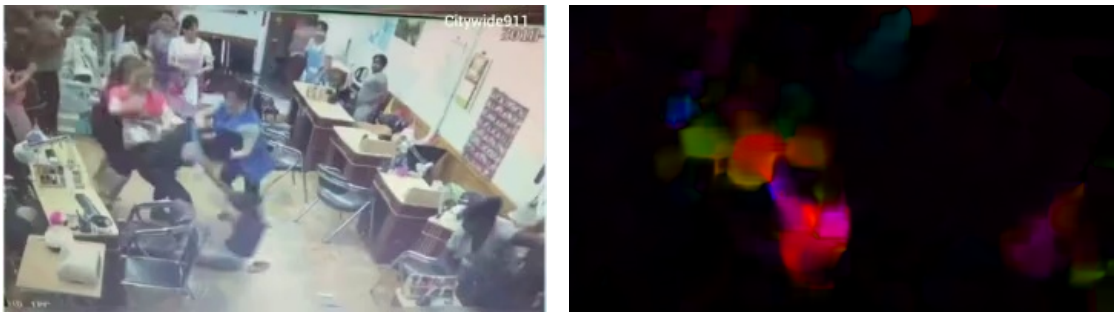
A supplementary step in this algorithm could be the subtraction of the mean optical flow between pairs of frames, in order to cancel out the camera movement: this becomes discretionary in most anomaly detection cases, due to the fixed stance of the majority of surveillance cameras, although IP cams equipped with the PTZ functionality are increasingly diffused[5].

Usually, an HSV rendering is delivered to provide a better visual understanding, as shown in Fig. 2.6 with an example.

2.4.2 CNN-Based Optical Flow Estimation

The main difficulty of the optical flow computation is intrinsic in its definition and it is referred to as the "aperture problem": the a priori knowledge on the relative displacement is a limited source of information when, for example, a moving object is seen through a small aperture or, in the case of real-world security video footage, occlusions and off-screen movements occur [100]. Therefore, it is possible to locally encounter ambiguous motion signals and, consequently, problems in the outlined algorithm due to the restricted nature of the local evidence used to estimate the optical flow. In addition, Gunnar Farnebäck's algorithm could be in some real-world

[5]Internet Protocol cameras are digital video cameras which communicate via an IP network; PTZ is the acronym for Panning (horizontal movement), Tilting (vertical motion) and Zooming (movement on the z axis), with some commercial IP cams having the internal motors to carry out these to focus on specific areas of the total field of view automatically or remotely controlled by the user.



(a) Original Frame **(b)** Dense OF

Figure 2.6: Gunnar Farnebäck's Optical Flow of a sample frame of a video from the RWF-2000 dataset presenting a scuffle, computed and visualized in the HSV space using the `Python cv2` library, with a neighborhood size of 5 pixels: the Hue indicates the angle (direction) of the flow, the Value the distance (magnitude) of the movement whereas the Saturation is fixed to 255; the background areas and the parts of the frame without movement are therefore black.

applications, with low latency required, too computationally expensive.

A recent field of research has been the use of Convolutional Neural Networks for optical flow estimation, with FlowNet by Philipp Fischer et al. [101] often cited as one of the pioneering architectures [102].

The main concept is to construct a suitable architecture that takes two images, naturally being two consecutive frames, as input, and train it in order to produce an estimate of the movement of every pixel from the frame f_i to the frame f_{i+1}. This process could logically be iterated over each successive pair of frames of a videotape, yielding as output its estimated dense optical flow. It should be noted that the network could well have no understanding of what the optical flow actually is: its objective is simply to emulate as well as possible the ground truth information, which is the actual optical flow of the clip traditionally computed via Gunnar Farnebäck's algorithm or already available, hence the usage of the word "estimate".

It is important to add that also unsupervised networks like UFlow [103] have been lately submitted, which leverage the burgeoning amount of videos available online, incomparably greater with respect to the lesser number of clips associated with their dense optical flow at disposal, imperative for a supervised learning approach. However, due to model availability and performance reasons, these models are left undiscussed since unexploited in the present work.

The fundamental ideas in most end-to-end trainable CNNs for this task are, firstly, to exploit the natural ability of convolutional layers to learn features at multiple levels of representation, and secondly, to render the network able to match them at the different locations of appearance in the two images. The correlation is

mathematically defined, for two informative patches of center $\mathbf{x}_1$ and $\mathbf{x}_2$, respectively for the first and second image, and square size $K := 2k + 1$, as follows:

$$c\left(\mathbf{x}_1, \mathbf{x}_2\right) = \sum_{\mathbf{o} \in [-k,k] \times [-k,k]} \left\langle \mathbf{f}_1\left(\mathbf{x}_1 + \mathbf{o}\right), \mathbf{f}_2\left(\mathbf{x}_2 + \mathbf{o}\right) \right\rangle \tag{2.7}$$

Notably, the expression is identical to the convolution operation, as treated in Sect. 1.3.3, with the only difference in having data convolving with other data instead of with a trainable filter. Therefore, no learnable weights are associated with this type of layer; however, due to a large amount of computation needed, that equals to $c \cdot K^2 \cdot w^2 \cdot h^2$ multiplications for comparing all patch combinations, a maximum displacement $d \ll k$ is introduced to limit the neighborhood of search to $D := 2d + 1$.

Since the correlation and convolution of two inputs have the effect of spatially shrinking the feature maps, aggregating information over large areas of the input images, which is indeed enforced by pooling layers in order to make network training computationally feasible, the problem of enlarging the final representation, to produce an output of spatial size $w \times h$, arises.

Consequently, *upconvolutional layers* are introduced and utilized in the latter part of the architecture: they are made of the sequence of a convolutional layer with an unpooling layer, which has the goal of extending the feature maps by upsampling, as opposed to pooling, with each step final step increasing each spatial dimension twice. Thus, the comprehensive structure resembles a species of an encoder-decoder model, as shown in Fig. 2.7.

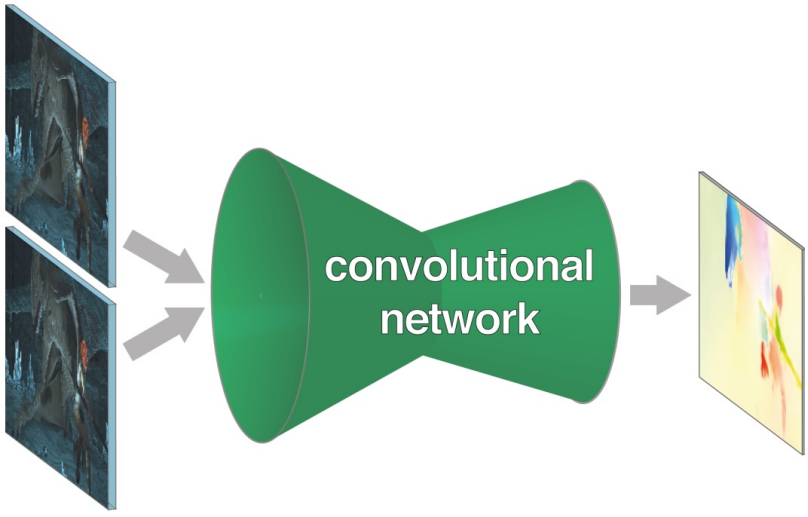


Figure 2.7: The encoding (compressing) and decoding (expanding) structure of an Optical Flow CNN, as shown in [101].

The reference dataset for model training and to benchmark the performances, are: KITTI Vision Benchmark Suite [104], which is obtained via a system of color and gray-scale video cameras together with a laser scanner and a GPS localization system mounted on a moving station wagon, thus capturing pedestrians, cars, and background movements; MPI Sintel Dataset [105], which is a naturalistic open source movie with ground truth optical flow (albeit rendered), and Flying Chairs [106], a synthetic dataset made of random Flickr photos with renderings of 3D chairs moving over the background. Fig. 2.8 shows the results of applying Flownet on a pair of frames from the Sintel movie.

(a) Original frames **(b)** Flownet OF Estimate **(c)** Real OF

Figure 2.8: The estimated (b) and the ground truth optical flow (c) of a pair of Sintel's frames - shown overlapped in (a).

2.4.3 C3D Features

As briefly recounted in Sect. 1.4.2, C3D Features refer to the output vector representation of a particular CNN architecture, named 3D ConvNet, whose main hallmark is the spatio-temporal performing of convolution and pooling operation as opposed to its standard 2D counterpart [79]. This model, as its introduction, reached a state-of-art performance on the Sports-1M dataset among other short-clips competitors: as outlined in its architecture in Tab. 2.1, the starting video clips have to be pre-processed, randomly extracting from them 16 representative frames, in order to be fed to the input layer.

To be more precise, the C3D features stem from the activation maps of the FullyConnected_6 layer, extracted by feeding to the network 16-frames long clips, trimmed from the original video (from which to extract the descriptor) with an overlapping factor of 50%: therefore, a video of length n frames is characterized by $\texttt{floor}(\frac{n}{16}) + (\texttt{floor}(\frac{n}{16}) - 1)$ activation maps. These are successively averaged to form a unique video descriptor vector of dimension 1×4096 and, finally, by applying an L2-normalization to this vector, we have as a result the representation formalized as C3D features of the input video.

By using the deconvolution method described by Zeiler and Fergus in [60] it is possible to comprehend the informative aspect encoded into the C3D features:

Layer	**Size ($n_c\times$ frames $\times$ height $\times$ width)**
Input	$3 \times 16 \times 112 \times 112$
Conv_1a (64)	$64 \times 16 \times 112 \times 112$
Pool_1	$64 \times 16 \times 56 \times 56$
Conv_2a (128)	$128 \times 16 \times 56 \times 56$
Pool_2	$128 \times 8 \times 27 \times 27$
Conv_3a (256)	$256 \times 8 \times 27 \times 27$
Conv_3b (256)	$256 \times 8 \times 27 \times 27$
Pool_3	$64 \times 4 \times 14 \times 14$
Conv_4a (512)	$512 \times 4 \times 14 \times 14$
Conv_4b (512)	$512 \times 4 \times 14 \times 14$
Pool_4	$512 \times 2 \times 7 \times 7$
Conv_5a (512)	$512 \times 2 \times 7 \times 7$
Conv_5b (512)	$512 \times 2 \times 7 \times 7$
Pool_5	$512 \times 1 \times 4 \times 4$
FullyConnected_6	1×4096
FullyConnected_7	1×4096
Softmax	

Table 2.1: 3D ConvNet architecture: all 3D convolution kernels are $3 \times 3 \times 3$ with stride 1 in both spatial and temporal dimensions, whereas all pooling kernels are $2 \times 2 \times 2$ except for Pool_1 which is $1 \times 2 \times 2$. In parentheses, the number of kernels is indicated for the convolutional layers; n_c in this notation stands also for the number of color channels for the input layer (3 as in the RGB standard).

as shown in Fig. 2.9, the last convolutional layer's feature maps have varying high-activation patches for the different frames.

In particular, the feature begins by capturing the subject of the video in the first frame, and as the motion occurs, it tracks it over the rest of the frames: therefore, C3D features are able to retain and focus on appearance and motion at different instants of a video segment, codifying it in an informative descriptor. On the contrary, the filters of standard 2D ConvNets applied to video footage could only focus on appearance, with the temporal information and peculiarities of each frame simply averaged out across all of them.

Moreover, it has been demonstrated, by projecting the features into lower dimensions, that C3D features are also a more compact and discriminative descriptor with respect to previous ones: in fact, it is possible to achieve a better performance using C3D features even if there exists a dimensionality cap to be respected [107]. The better semantic separability of the different action classes observed on the UCF-101 dataset is also worth mentioning, as shown in Fig. 2.10.

For further deepening, the reader is referred to the original publication [79].

2.5 Baseline Networks

2.5.1 Supervised Learning Approach

The combined use of RGB frames and Dense Optical Flow as dual input for a neural network architecture has achieved considerable success in recent publications [82][85] in the most diverse domains. As a consequence, it should be no surprise that this framework has been applied also to the anomaly detection task.

The approach is to separately work on the two input media, via 3D-convolutional blocks[6] that progressively extract higher-level features and informative patches, and then merge together the two obtained representations. This structure is illustrated in Fig. 2.11.

Although rather straightforward, this path is in fact non-trivial, in the sense that several architectural choices must be addressed and made.

Firstly, the structure of the 3D convolutional blocks has to be chosen: among the several solutions proposed over the years, this framework has been applied relying on the so-called *pseudo-3D blocks* introduced by Zhaofan Qiu et al. in [108], which in their turn stem from the 2D Residual blocks, ushered by Kaiming He et al. for the task of image recognition in [109].

A Residual block, or unit, could be described by the expression:

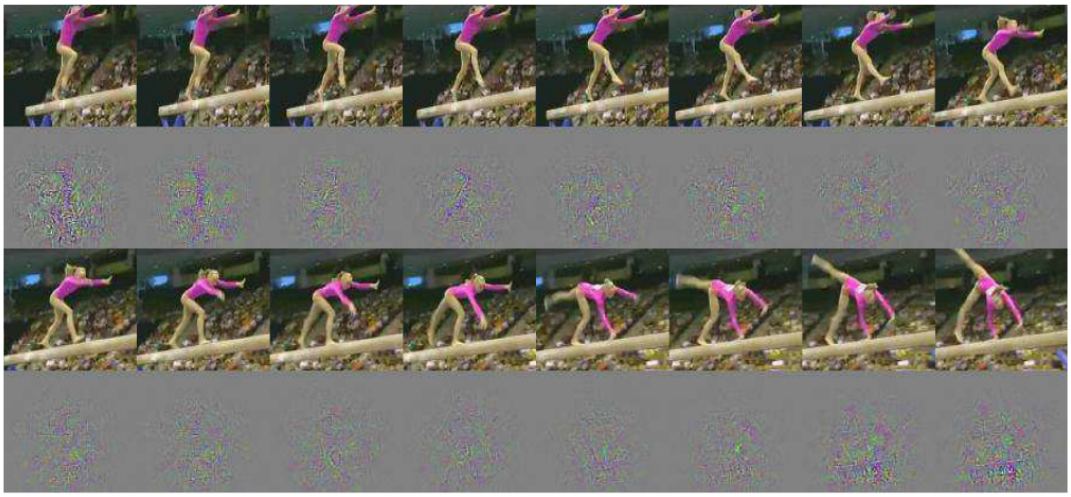
$$\mathbf{x}_{t+1} = \mathbf{H}(\mathbf{x}_t) + \mathbf{F}(\mathbf{x}_t), \tag{2.8}$$

with $\mathbf{x}_t$ denoting the input of the t-Residual block, $\mathbf{x}_{t+1}$ its output, $\mathbf{H}(\mathbf{x}_t) = \mathbf{x}_t$ being an identity mapping, and $\mathbf{F}$ being a non-linear additive residual function. The expression can be rewritten, using $\mathbf{I}$ as the identity operator, as follows:

$$(\mathbf{I} + \mathbf{F}) \cdot \mathrm{x}_t = \mathrm{x}_t + \mathbf{F} \cdot \mathrm{x}_t := \mathrm{x}_t + \mathbf{F}(\mathrm{x}_t) = \mathrm{x}_{t+1}, \tag{2.9}$$

which better highlights that the input $\mathbf{x}_t$ essentially follows a forked path: a *shortcut connection*, that is, a connection skipping one or more layers, towards an additive node, and the normal sequence of weight layers $\mathbf{F}(\mathrm{x}_t)$, usually two convolutional layers, linearly followed by the same additive node where the two paths are joined back together. It is important to note that the identity shortcut connections add no extra parameter to the model, thus no additional computational complexity, and that the end-to-end trainability of the architecture according to the stochastic gradient descent and back-propagation methods remains unaltered.

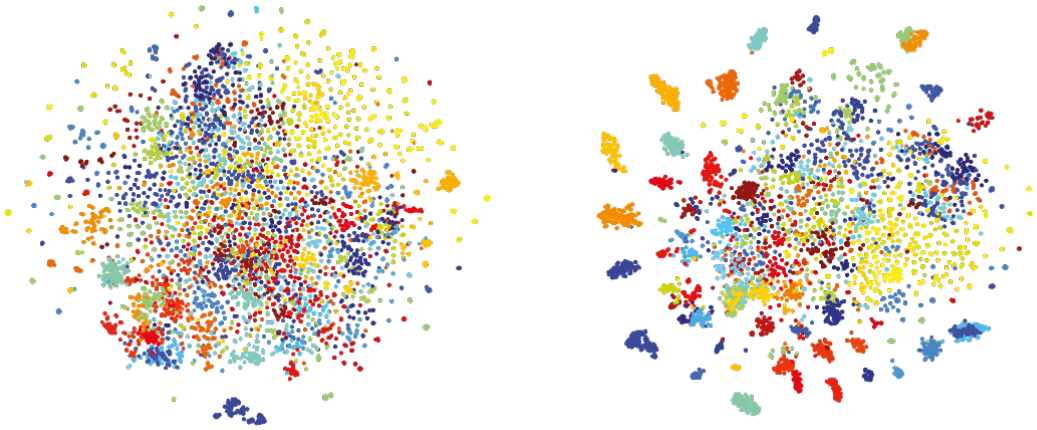
[6] That is, combinations of 3D-convolutional layers and 3D-pooling layers,



(a) Pole Vault Action

(b) Apply Makeup

Figure 2.9: Feature maps of the Conv_5b layer for two sample videos from the UCF-101 dataset: while in the first frame of both videos, the focus of the features is the global subject (the gymnast and the human face, respectively), in the successive frames the features focus more on the salient motion occurring in the footage, encoding it. Images from [79].



(a) Imagenet **(b)** C3D

Figure 2.10: T-distributed Stochastic Neighbor Embedding (tSNE) visualization of the feature embedding of Imagenet features (a) and C3D features (b): the better separability of videos pertaining to the same classes, indicated via dots of identical color, is self-evident in the latter, whereas Imagenet features are more sparse therefore less suitable for video descriptor extraction. Images from [79].

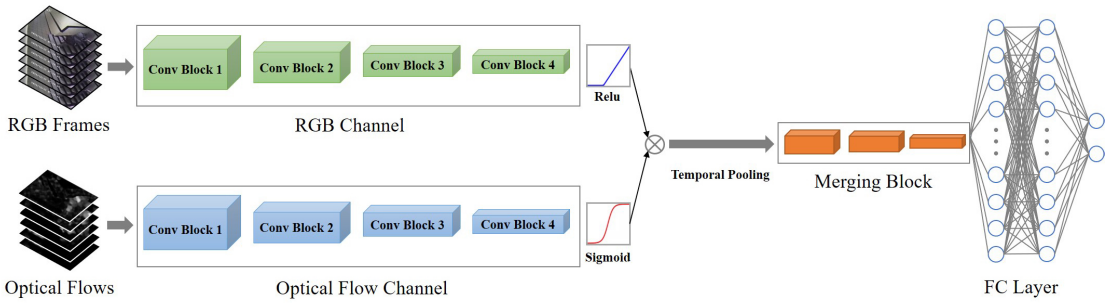


Figure 2.11: The two-stream architecture used in [87], from which this work stems.

At this point, it is convenient to recall the principle that 3D convolutions are able to simultaneously model the spatial information and construct temporal connections among successive frames.

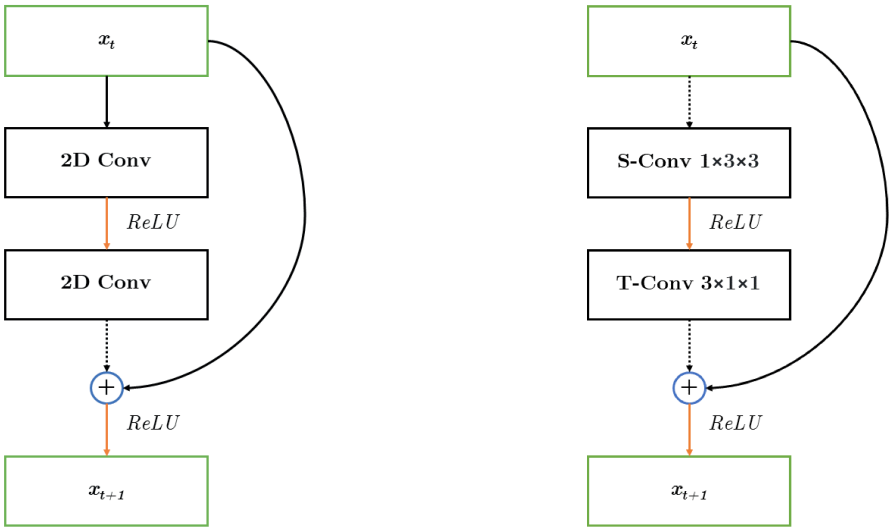
Denoting the size of a 3D convolutional filter as $d \times k \times k$ with d being the temporal depth of the kernel of spatial size $k \times k$, it could be consequently decoupled into a *spatial* convolution $\mathbf{S}$ with a filter of dimension $1 \times k \times k$, equivalent to a standard 2D convolution, and a *temporal* convolution $\mathbf{T}$ with a kernel of size

$d \times 1 \times 1$, with an ensuing benefit in the computational complexity of the operation.

Hence, the two aforementioned tenets could be united in the following expression, which represents the mathematical formulation of a pseudo-3D block[7] according to the previously introduced notation:

$$(\mathbf{I} + \mathbf{T} \cdot \mathbf{S}) \cdot \mathbf{x}_t := \mathbf{x}_t + \mathbf{T}(\mathbf{S}(\mathbf{x}_t)) = \mathbf{x}_{t+1} \tag{2.10}$$

A visualization of both a standard Residual block and a pseudo-3D block is provided in Fig. 2.12.



(a) Residual block **(b)** Pseudo-3D block

Figure 2.12: A standard Residual block (a) is characterized by two (or more, as suggested by the dotted line) weight layers, followed by the summing point where the path of the identity function is combined; its 2D convolutional filters in the original implementation [109] are of size 3×3. A pseudo-3D block (b) decomposes a single 3D convolution in its temporal and spatial components, with the simple succession of the two - hence, the application of the $d \times 1 \times 1$ convolution on the output of the spatial $1 \times k \times k$ one - heuristically [108] being the best performing fusion method; the dotted lines suggest the expansibility of this block to a $1 \times 1 \times 1$ factorization for computational purposes and dimensionality restoration. In both cases, the ReLU function applies after the additive joint.

[7] More specifically, using the nomenclature provided by Zhaofan Qiu et al. in [108], of a Pseudo-3D Block of type A.

Secondly, a paramount decision to be taken is the fusion method of the two informative flows. The simplest and main approach, used in several aforementioned solutions like the two-stream ConvNet in [82] and the Two-Stream Inflated 3D ConvNet (I3D) [85] is to perform a mere average of the outputs of the two paths, with them being usually a class function score for an action recognition task.

In this case, a more thoughtful method has been applied, referred to by the authors as a *temporal pooling gate*, by leveraging two different nonlinear activation functions and a combination of two layers, one of which is a pseudo-layer due to the absence of learnable parameters. In particular, the RGB channel has as its final activation function the Rectified Linear Unit, hence it produces an output representation ranging in the interval $y = [0, +\infty[$. On the other hand, at the end of the Optical Flow channel, the sigmoid function is placed, which scales the x in input to the interval $y = [0, +1]$.

Therefore, by multiplying the output of the RGB channel with the output of the Optical Flow channel, and following it with a 3D-Max Pooling layer of kernel size $= 8 \times 1 \times 1$, the obtained representation is the result of a self-learned pooling strategy which utilizes the Optical Flow information as a scaling factor to better retain and drop the informative features of the RGB channel. This procedure is similar to the gate strategies adopted in several architectural models to elaborate a sequential or temporal input, such as an LSTM [87][78].

Finally, three additional sequences of pseudo-3D Convolutional layers and 3D-Max Pooling layers have been appended after the aforesaid merging node, in order to have hundreds and thousands of further weights and parameters to learn the highest-level feature of the whole representation. The comprehensive architecture is schematized in Tab. 2.2 and envisages also the presence of two Fully-Connected layers.

Overall, the general framework is the following:

- Video pre-processing: each video is resized and cropped to a fixed resolution of 224×224;
- Gunnar Farnebäck's Optical Flow computation between each pair of frames for every video via the `Python cv2` library, as described in 2.4.1, obtaining for each video a 2-dimensional vector of size 2×300, with 300 being the total number of frames and 2 being the two - vertical and horizontal - components of the Optical Flow;
- RGB frames extraction via the `Python cv2` library, obtaining for each video a 3-dimensional vector of size 3×300, with 3 being the number of color channels;
- Creation of 5-dimensional `NumPy` arrays by concatenating the two representations;

Block	**Layer**	**Filter**
RGB and Optical Flow Channels (×2)	Conv3d	$1 \times 3 \times 3$@16
	Conv3d	$3 \times 1 \times 1$@16
	MaxPool3d	$1 \times 2 \times 2$
	Conv3d	$1 \times 3 \times 3$@32
	Conv3d	3×1×1@32
	MaxPool3d	$1 \times 2 \times 2$
Fusion	Multiply	
Pooling	MaxPool3d	$8 \times 1 \times 1$
Merging Block	Conv3d	$1 \times 3 \times 3$@64
	Conv3d	$3 \times 1 \times 1$@64
	MaxPool3d	$2 \times 2 \times 2$
	Conv3d	1×3×3@64
	Conv3d	3×1×1@64
	MaxPool3d	$2 \times 2 \times 2$
	Conv3d	$1 \times 3 \times 3$@128
	Conv3d	$3 \times 1 \times 1$@128
	MaxPool3d	$2 \times 2 \times 2$
Fully Connected Layers	FC_Layer	128
	FC_Layer	128
	Softmax	2

Table 2.2: The architecture used in [87]; the number after the at symbol @ corresponds to the number of filters for that specific convolutional layer; the number 2 in line with the Softmax layer indicates the 2 possible outputs (0,1) of the binary classification task of violence detection.

- Sparse sampling of 64 frames out of the 300 for dimensionality reduction;
- Passing of the $64 \times 224 \times 224 \times 5$ input through the architecture as visualized in Fig. 2.11, with the first three components following the RGB channel and the last two the Optical Flow channel;
- Computing of the binary class for the analyzed video, with 1 representing a Fight instance whereas 0 stands for Non-Fight footage.

This architecture is end-to-end trainable using the techniques described in Sect. 1.3.

2.5.2 Semi-Supervised Learning Approach

The method proposed by Sultani et al. in [86] falls into the specific category of Multiple Instance Learning (MIL), which relies on weakly labeled data and deals with bags of instances instead of single instances [110].

In this context, a bag is defined as an invariant set X of data points:

$$X = \{\mathbf{x}_1, \ldots, \mathbf{x}_K\}, \tag{2.11}$$

that is as a whole characterized by a single label Y, as follows:

$$Y = \begin{cases} 0, & \text{iff } \sum_k y_k = 0 \\ 1, & \textit{otherwise} \end{cases} \Longrightarrow Y = \max_k \{y_k\}, \tag{2.12}$$

with y_k being the individual binary label $y_k \in \{0,1\}$ for a given data point x_k that is assumed to be inaccessible at training time and, possibly, completely unknown, thus relaxing the assumption of having accurate high-level annotations.

According to this MIL formulation, the precise temporal locations of the abnormal activities in the videos are not needed: only video-level labels on the entire video are required, simply specifying if the untrimmed footages present some anomalies are present or not. Therefore, a positive video is a bag B_p of instances, i.e., temporal segments $(p^1, p^2, \ldots, p^m)$, where m is the total number of snippets in which the original video is split, altogether characterized by the label $Y = 1$. On the other hand, a negative bag B_n is a video whose n fragments $(n^1, n^2, \ldots, n^n)$ are all negative instances, i.e., $Y = 0$.

Bearing in mind the expression of the hinge loss, which is used especially in Support Vector Machines or in general in classification problems that incorporate the distance (margin) from the classification boundary, penalizing even correct prediction but with small margins [111], and for an intended output $y = \pm 1$ and a classifier score $\hat{y}$ is:

$$L(y) = \max(0,1 - y \cdot \hat{y}), \tag{2.13}$$

the objective function of the described problem to be minimized, for the classifier $\mathbf{w}$ to be learned, is:

$$\min_{\mathbf{w}} \left[\frac{1}{z} \sum_{j=1}^{z} \overbrace{\max\left(0,1 - Y_{\mathcal{B}_j}\left(\max_{i \in \mathcal{B}_j}(\mathbf{w} \cdot \phi(x_i)) - b\right)\right)}^{①} \right] + \|\mathbf{w}\|^2, \tag{2.14}$$

with ① being the hinge loss, $Y_{\mathcal{B}_j}$ the bag-level label, y_i the label of each instance, $\phi(x)$ the feature representation of the video segment, b a bias, and k the total number of bags.

The innovative formulation proposed is to pose the problem of anomaly detection as a regression one, using a modification of the ranking loss due to the absence of segment-level annotations: in particular, the objective function has to compare the segment featuring the highest anomaly score from the positive bag $\mathcal{B}_p$ with the corresponding snippet from the negative bag $\mathcal{B}_n$:

$$\max_{i \in \mathcal{B}_p} f\left(\mathcal{V}_p^i\right) > \max_{i \in \mathcal{B}_n} f\left(\mathcal{V}_n^i\right), \tag{2.15}$$

where $\mathcal{V}_p$ and $\mathcal{V}_n$ represent anomalous and normal video snippets. It is of paramount importance to note that, since the positive bag also contains negative videos, it is not guaranteed that the $\mathcal{V}_p^i$ selected actually is an anomalous segment, although it remains very likely [112].

The following is the formulation of the adopted hinge loss that implements the ranking loss:

$$L_r(\mathcal{B}_p, \mathcal{B}_n) = \max\left(0,1 - \max_{i \in \mathcal{B}_p} f\left(\mathcal{V}_p^i\right) + \max_{i \in \mathcal{B}_n} f\left(\mathcal{V}_n^i\right)\right), \tag{2.16}$$

to which two extra constraints for the positive videos $\mathcal{V}_p^i$ are added, to form the final formulation:

$$L(\mathcal{B}_p, \mathcal{B}_n) = L_r(\mathcal{B}_p, \mathcal{B}_n) + \lambda_1 \overbrace{\sum_{i}^{(n-1)} \left(f\left(\mathcal{V}_p^i\right) - f\left(\mathcal{V}_p^{i+1}\right)\right)^2}^{①} + \lambda_2 \overbrace{\sum_{i}^{n} f\left(\mathcal{V}_p^i\right)}^{②}, \tag{2.17}$$

with ① being a smoothness term which minimizes the difference scores of successive segments, to reinforce the insight that the anomaly score should vary gracefully during the video, and ② being a sparsity constraint, due to the fact that

typically abnormal events last for a short amount of time with respect to the total length of the untrimmed videos, and λ_i with $i = 0{,}1$ two tunable weights.

Therefore, the total objective function inclusive of the model's weights $\mathcal{W}$ is:

$$\mathcal{L}(\mathcal{W}) = L\left(\mathcal{B}_p, \mathcal{B}_n\right) + \|\mathcal{W}\|_F \tag{2.18}$$

Overall, the comprehensive framework is the following:

- Video pre-processing: each video is resized to a fixed resolution of 320×240 and 30 fps;
- Segmentation of each video into a fixed number of not-overlapping clips (32 is the heuristic number used), with the subsequent formation of two bags, one positive and one negative, each with 32 segment instances, as shown in Fig. 2.13;
- Extraction of the C3D features, as described in Sect. 2.4.3, for each bag instance;
- Computing of the anomaly score for each instance using the Fully Connected Feedforward Neural Network outlined in Tab. 2.3;
- MIL Ranking Loss computation between the highest-score video segments from each bag, as visualized in Fig. 2.14;

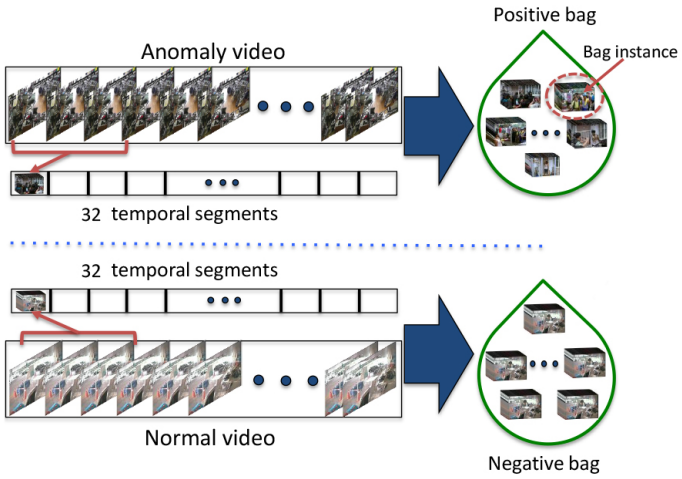


Figure 2.13: The second step of the video processing according to the Semi-Supervised approach proposed by Sultani et al.: the cutting of a negative (normal) and of a positive (anomaly) video into a fixed number of temporal segments, and the bags' construction. Image from the original publication.

Layer	Dimension
Input	1×4096
FullyConnected_1 (ReLU)	1×4096
Dropout_1	0.6
FullyConnected_2	1×512
Dropout_2	0.6
FullyConnected_3 (Sigmoid)	1×32
Dropout_3	0.6
Output	1

Table 2.3: The simple architecture of the Fully Connected Feedforward Neural Network which predicts the anomaly score of a video starting from its pre-computed C3D features. Each segment of the two bags is processed first by the standard C3D illustrated in Tab. 2.1 and then by this architecture: the comprehensive output is a score for each segment.

Figure 2.14: The final step of the framework: the highest-score segments from each bag (in this visualization, the positive is in red, whereas the negative thus normal is in red dots) are matched, maximizing their separation using the Loss defined in Eq. 2.17.

2.6 Technical Considerations

Both the presented approaches have their merits, which have been extensively expanded by researchers, and, being the problem open, there is not a perfect solution. In general, the methods following a semi-supervised or an unsupervised learning path usually provide more generalization potential at the cost of major added complexity and higher computational needs.

More in detail, as deepened by [112], a problem of most MIL-based methods in this application is the high sensibility to label noise: the possibility of the mistaken selection of a normal snippet as the top abnormal event in an anomaly video is a present risk in the positive bag construction. Moreover, the normal snippets may on the contrary be relatively easy to fit, undermining the training convergence.

Additionally, and more importantly, the assumption of each positive video containing only one abnormal snippet could in fact prove to be incorrect. For example, it should be reminded that the positive videos of the UCF-Crime dataset have median duration in the order of several minutes, though they present widely sparse lengths, from the 4 seconds of `Assault_038` to the over 35 minutes of `Arrest_047`: nevertheless, according to this framework, all of the videos in the pre-processing phase are indistinctly divided into a fixed amount of 32 clips.

This entails two main drawbacks: the chance of having a more effective training process containing more abnormal snippets per video is missed, making the more lengthy videos only burdensome from a computational and storage point of view and not exploiting their informative specificity; the selection by the model of the least relevant of the abnormal snippets among the various possible abnormal snippets in a positive bag.

Therefore, for the combination of these technical reasons and the limitedness of the available resources, the problem has been tackled following the same approach as the one in Sect. 2.5.1, although it may result more simplistic.

2.7 Metrics

Having posed the problem of anomaly and violence detection as a binary classification task, the metric of reference is accuracy.

Definition 2.7.1 (Accuracy). The accuracy of a classification algorithm is defined as the number of true positives and true negatives divided by the sum of the number of true positives, true negatives, false positives, and false negatives, and represents the number of correctly predicted data points:

$$accuracy = \frac{(\mathrm{TP} + \mathrm{TN})}{(\mathrm{TP} + \mathrm{TP} + \mathrm{FP} + \mathrm{FN})} \tag{2.19}$$

Another relevant metric used in this work, in particular in the optical flow computation discussion, is the Average Endpoint Error.

Definition 2.7.2 (Endpoint Error). The End-to-end point error, or Endpoint Error in brief, between two optical flow vectors of the same dimensions, V_{truth} and V_{est}, is a vector of equal dimensions as the original, in which each point is defined as the Euclidean distance between the two:

$$EPE = ||V_{truth} - V_{est}|| \tag{2.20}$$

Chapter 3

A Novel Architecture

In this chapter, the proposed architecture for solving the problem of anomaly and violence detection on the RWF-2000 dataset is illustrated, with each section specifically dedicated to its threefold main innovative peculiarities.

Finally, Sect. 3.4 is reserved for its whole structure: the adopted framework, following the previous works on this task, has been the construction of a two-stream architecture with RGB frames and estimated optical flow vectors as the dual inputs. It is important to note that all the code implementation of the said model has been done leveraging the Keras API [113], which is built on top of TensorFlow 2 [114].

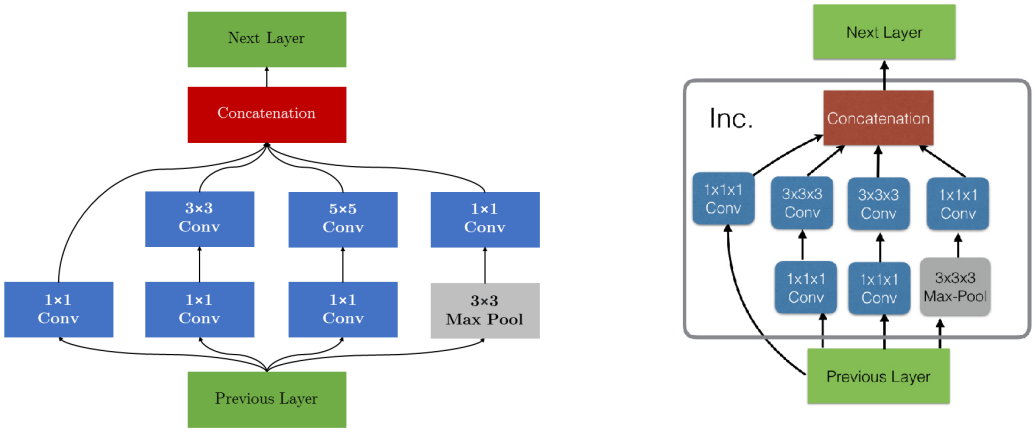
3.1 Improved 3D-Inflated Inception Block

The first contribution of this novel architecture is the proposal of an innovative convolutional block, which stems from the aforecited Inception Convolutional 2D block (extensively described in Sect. 1.3.3 and visualized in Fig. 1.16).

This architecture has been already expanded in [85] to being applicable to video inputs, therefore starting with a 2D block and *inflating* the filters of the convolutional and pooling layers which is composed of an additional temporal dimension. Due to the fact that filters are usually square, the result is the usage of cubic filters of size $N \times N \times N$.

In Fig. 3.1 a representation of such a block compared with the original 2D implementation is offered: the main difference between the two implementations, apart from the obvious one less axis in the 2D version, is the absence of the 5×5 convolution in its 3D counterpart, deemed unnecessary by the authors with respect to the dimensionality of the inputs, and replaced by an additional parallel $3 \times 3 \times 3$ convolution.

This *inflating* process could thereupon be applied also on the more recent and optimized InceptionV3 block [56], which integrates a 1×1 convolutional block before



(a) Original Inception block **(b)** Inflated Inception block

Figure 3.1: The comparison between the 2D Incpetion block (left) and of the Inflatet 3D Inception block as presented in [85]: note the replacement of the 5×5 convolution with an extra $3 \times 3 \times 3$ convolution.

the so-called *one-time factorization* of the convolutions, that is the decomposition of the computationally more complex convolutions of kernel size $N \times N$ into the sequence of two simpler ones, i.e., a convolution with a $1 \times N$ filter followed by a $N \times 1$ one. In the 2D application, this has proven to cause a computational cost saving almost exponential to N.

Moreover, this factorization and optimization principle follows the core tenet of the pseudo-3D block illustrated in Sect. 2.5.1, that is, to decouple the temporal convolution and the spatial convolution. As demonstrated in [108], the mere sequence of the two decoupled convolutions offers also the best performance with respect to the other more complex proposals, which include the concatenation of the output of the S-convolution and T-convolution applied to the same input and the cascade of the two convolutions with an additional shortcut connection in order to enable a direct connection between **S** and the final output

Finally, the presence of the 5×5 convolution has been restored: in particular, it has been re-instituted but replaced by factorizing it into two consecutive 3×3 convolutions, as suggested by the principles in [56] which ensures a relative gain of $\approx 28\%$ in computational costs. Self-evidently, these 2D convolutions have been in turn inflated and decoupled, for a total sequence of five convolutions with respective filters of size $1 \times 1 \times 1$ (for dimensionality reduction), $1 \times 3 \times 3$, $3 \times 1 \times 1$, $1 \times 3 \times 3$, and $3 \times 1 \times 1$, in accordance with the other branches of the block.

Overall, the block proposed in this work is the concatenation of four branches

presenting the following characteristics:

- A convolution filter of size $1 \times 1 \times 1$;
- The sequence of three convolutions with respective kernels of dimensions $1 \times 1 \times 1$, $1 \times 3 \times 3$, and $3 \times 1 \times 1$;
- The series of five convolutions with respective filters of size $1 \times 1 \times 1$, $1 \times 3 \times 3$, $3 \times 1 \times 1$, $1 \times 3 \times 3$, and $3 \times 1 \times 1$;
- The sequence of a Max Pooling layer with the kernel of size $1 \times 3 \times 3$ and of a convolution $1 \times 1 \times 1$.

All the convolution blocks have as kernel initializer the "He Normal" [115] with the Rectified Linear Unit as the activation function; the other features of these layers are a stride factor of $1 \times 1 \times 1$ and the padding option set to "same", therefore having the output size equal to the input size, except for the convolutions used to reduce the dimensionality (the ones with filter $= 1 \times 1 \times 1$) which have a zero-padding.

Having used the Keras framework, by concatenation of the four branches it is meant the use of the Concatenate layer, which is a class of layers that takes as input a list of tensors, in this case, the four outputs of the respective four branches, and returns a single tensor that is the chain of all inputs. The block is illustrated in Fig. 3.2

3.2 Optical Flow-based Intelligent Video Cropping

As mentioned throughout the present thesis, the primal issue in working with video footage as data input is the ensuing memory occupation and consumption, which clashes with the resource constraints and with some low-latency and/or real-time scenarios. Notwithstanding the used compression formats and strategies illustrated in Sect. 1.1.3, the approach followed in nearly every work on the subject is the sparse sampling of the video clips, leveraging once again the great correlation among frames close in time.

However, other dimensionality reduction strategies could be still needed, one among all the reduction of width and length of the videos to a uniform size: this also addresses the problem of having video footage of different dimensions, whereas a Deep Learning model, whatever it might be, needs a staple and precise input dimension. However, the act of resizing the whole frame could entail a non-negligent loss of information which could severely affect the model's performance.

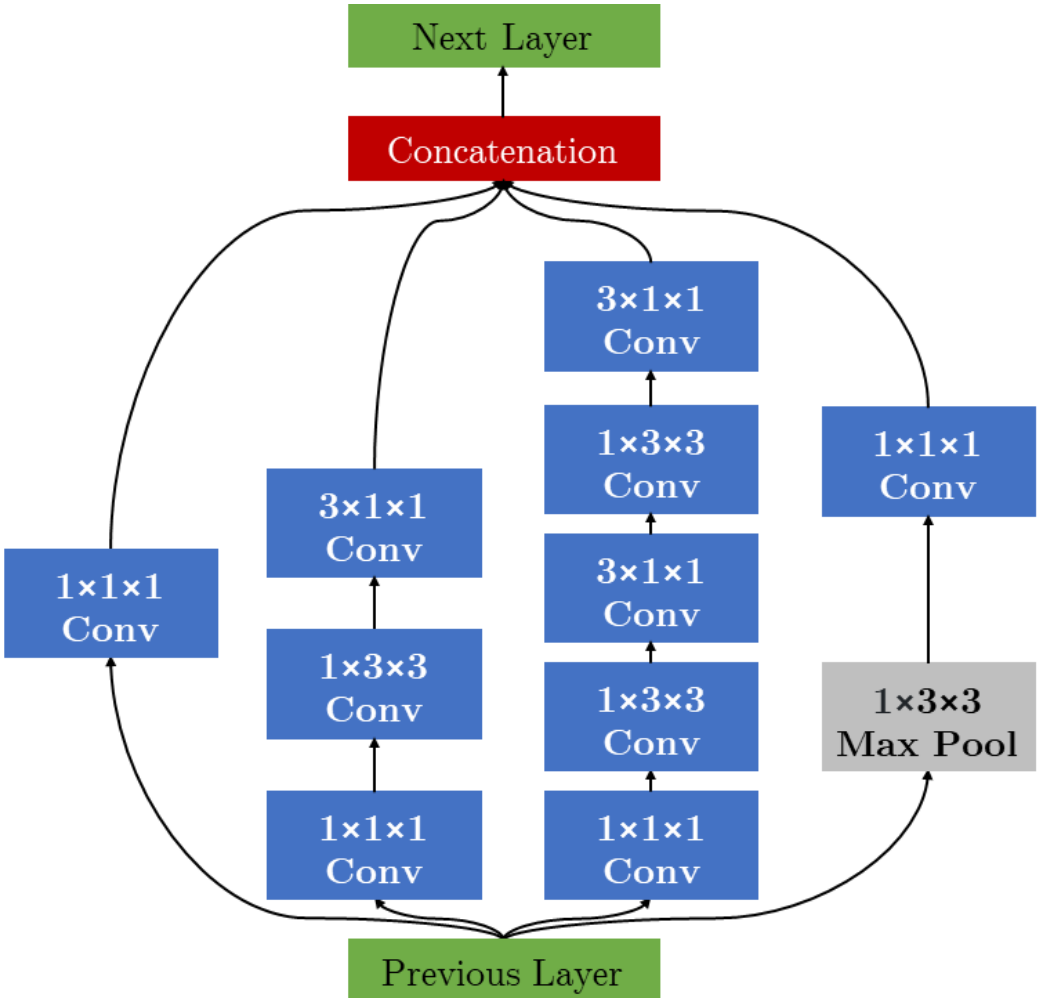


Figure 3.2: The structure of the proposed block, here named Iv3D block.

Therefore, in this work, a novel method for a sensible dimensionality reduction of the input data, named *Intelligent Video Cropping*, is outlined. The intuition behind this proposal is the fact that not all the region captured by a CCTV camera, which usually covers a large area due to its strategic location, may be relevant for the anomaly detection task: this tenet is ever more enforced when the clips are of short temporal length since it becomes more implausible, although not impossible, that the significant actions happen all over the entire frame.

Hence, the core idea is to use the computation of the optical flow over a sparse sample of the video frames to construct an Intensity Map by stacking together the features extracted by the OF calculation (or estimate). This is done by computing the norm of each vector indicating the motion displacement intensity, therefore forming a map, and then by summing all the intensity maps to form a final

representation.

Subsequently, a Region of Interest (RoI) is determined if the final intensity map encloses an area occupying a fraction of the whole frame. In this case, the cropping of the video is conducted, pairing it with a suitable additional resizing procedure if the obtained resolution is nonetheless greater than the required input dimensions; therefore, all the procedure is *intelligent* in the sense that it is entirely adaptive and dependent on the specific clip case[1]. On the other hand, if the movement features of the samples are more diffused, a standard resizing of the whole clip is performed.

Altogether, the complete algorithm of this procedure, visualized in Fig. 3.3, is the following:

- Sparse sampling of 10 pairs of frames from the video, each one at a time distance of 0.5 second - thereby 15 frames - from the previous couple, with a random coefficient determining the initial frame in the frame interval $[0 - 14]$;
- Computation of the optical flow between each pair of frames, hence obtaining 5 bi-dimensional motion maps;
- Production of 5 intensity maps by computing the norm of each vector;
- Sum of the 5 intensity maps to obtain the final intensity map and calculation of its centroid;
- A three-way forked path is then envisaged:
  - If the Region of Interest is localized in an area equal or smaller than 220×220 pixels, cropping of the entire video centering in the centroid of the RoI, as in Fig. 3.3a with a padding of 2 pixels on each extremity to reach 224×224 pixels; an exception to this centering staple is foreseen with the purpose to avoid to chop beyond the frame boundaries when the movement is localized near the edges of the scene, in which case the center of the cropped frames would not correspond to the centroid of the RoI;
  - If the Region of Interest is localized in an area smaller than the 60% of the whole frame with empty vertical and horizontal bands of at least 15%, cropping of the clip centering in the centroid of the RoI to fully contain

[1]For the sake of exhaustiveness, it should be noted that a similar approach was theorized by Ming Cheng et al. in [87], although it was not applied in their final implementation publicly available on GitHub [116]; moreover, the cropping strategy proposed by them had fixed the cropping region dimensions, therefore encountering problems in the case of diffuse movement.

it with a 2 pixels padding per side, and subsequent adaptive resizing to obtain a square of height and width each equal to 224 pixels;

– Otherwise, standard resizing of the whole clip to final dimensions of 224×224 pixels, as in Fig. 3.3b.

The final result of this strategy is having clips of the desired width and length 224×224 encapsulating more relevant information, which means a less compressed representation of the region of interest of the task, than clips with standard data pre-processing, at the cost of a slight additional computation.

3.3 Fine-Tuned LiteFlowNet2 for Optical Flow Estimation

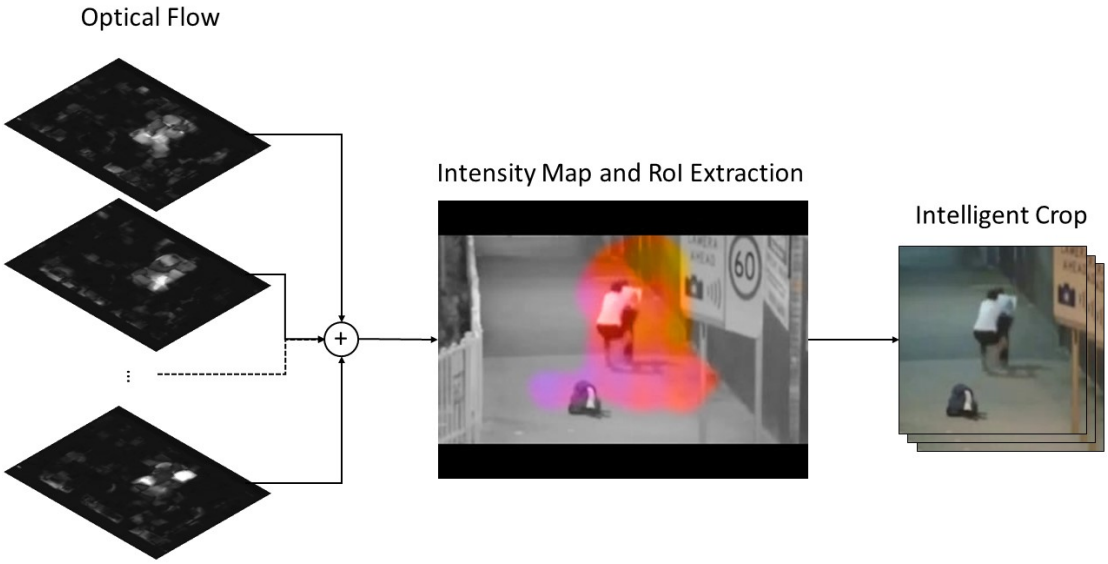
The Gunnar Farnebäck's algorithm for the Optical Flow extraction, outlined in Sect. 2.4.1, is computationally very expensive. Moreover, the just outlined method calls for additional computation of the optical flow, although to a lesser scale and less impactful to the overall performance. Among the various approaches available, relevance has been bestowed in this work to the CNN-based methods for the estimation of the optical flow.

In particular, the model named LiteFlowNet2, proposed by Tak-Wai Hui et al. in [102] and visualized in Fig. 3.4, has been picked among the alternatives: while for a more deep understanding of its working flow the reader is referred to the original implementation [117], it is important to underline the main characteristics that render this choice suitable for the task.

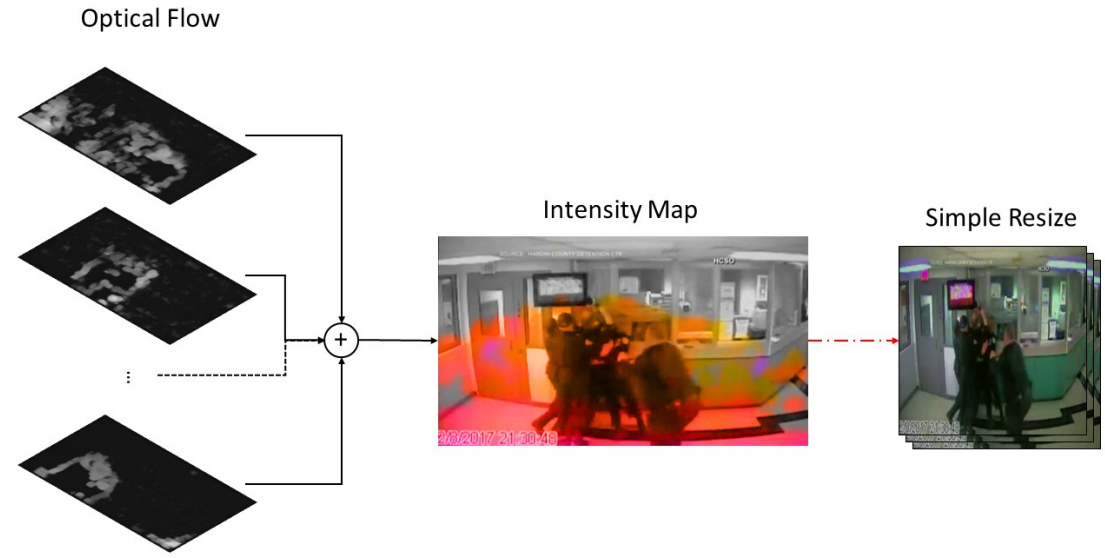
Firstly, compared to other architectures, it is extremely fast: it requires 6.42 M parameters, while for comparison the original FlowNet2 has 162.49 M learnable weights; therefore, it reaches an impressive real-time inference performance on full-sized 1024×436 images when an NVIDIA GPU as the GTX 1080 is adopted, i.e., an fps processing superior to 24fps with less than 40ms of runtime for each pair of frames [102].

Secondly, and equally relevant, it has a first-rate trade-off between being lightweight and performing well: after being trained on the Flying Chairs and Things3D [118] datasets, the measured Average end-point error of the model on the two reference benchmark datasets on which it has been tested, i.e., Sintel and KITTI 2012, are respectively of 3.85 and 3.77, at the time the state of art performances and still to date among the best results regardless of the computational heaviness.

Instead of using the provided model with its off-the-shelf weights, a global fine-tuning procedure of the LiteFlowNet2 pre-trained model has been performed, to further strengthen its performance in the application case.



(a)



(b)

Figure 3.3: Two different application cases from the RWF-2000 dataset: in the first one, the intensity map obtained by the stacking of the computed optical flows encloses a localized and relatively small area of the clip, which is therefore adaptively cropped; in (b), the movement is diffused, hence it is applied only the resize of the video footage.

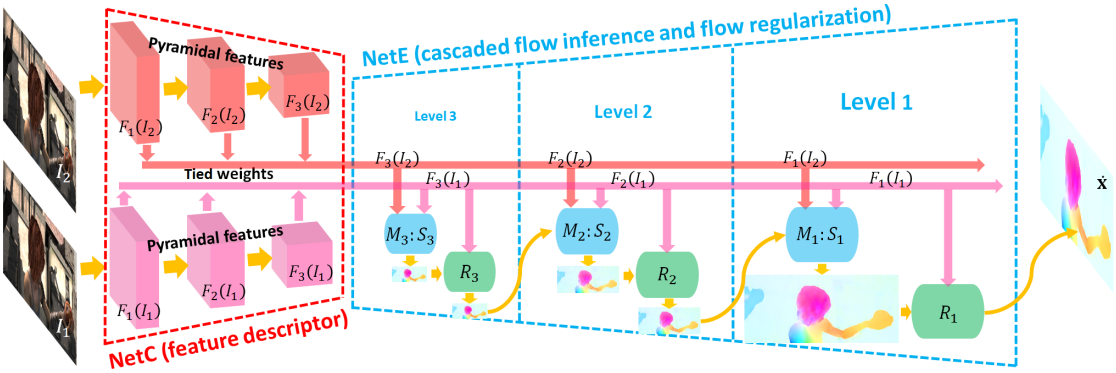


Figure 3.4: A broad representation of the architecture of LiteFlowNet2 from [102]: the main takeaway is that the network possesses a dual pyramidal feature descriptor, which is the encoder, here with a design of only a 3-level pyramid for easing the understanding, and a cascaded module, which is the decoder, for inferring the optical flow starting from a pair of feature maps from the encoder; several additional peculiarities, regularizers, and optimizers are here not displayed.

A random subset of the UCF-Crime videos for a total of 10000 frame pairs has been used in this respect, with the optical flows computed via Gunnar Farnebäck's algorithm employed as ground truth for the optical flow estimation task. This choice has been done due to the similarities in appearance and portrayed movement between the UCF-Crime and the RWF-2000 datasets; the latter could not have been used for this purpose otherwise it would have caused data leakage [119]. It is important to add also that the randomly selected frames respected the resolution distribution of the RWF-2000 dataset as visualized in Fig. 2.5.

In order to ease the work, the TensorFlow-based re-implementation of LiteFlowNet2 provided by [120], instead of the original Caffe one in [117], has been used as the pre-trained model. This fine-tuning procedure has been accomplished by setting all the layers weights to `learnable`, with initial learning rate = 6e-5 as done by the same authors in their fine-tuning trials and reducing it by half for every increment of 10K iterations, for a total of 50K iterations.

The results of this model, hereinafter referred to as FT-LiteFlowNet2, are reported in Tab. 3.1: the improvement of the performance on the UCF-Crime subset is small but non-negligible.

Model	Sintel Final	KITTI 2012	UCF-Crime*
LiteFlowNet2	**3.85**	**3.77**	11.31
FT-LiteFlowNet2	3.88	4.11	**9.88**

Table 3.1: Comparison of the AEE of the original model and of the fine-tuned one on the two benchmark datasets and on the reduced subset, hence the asterisk, of the UCF-Crime dataset used in the fine-tuning procedure; the best values are reported in bold.

3.4 Architecture

The architecture of the proposed model is visualized in Fig. 3.5 and extensively deepened in Tab. 3.3, 3.4, and 3.2.

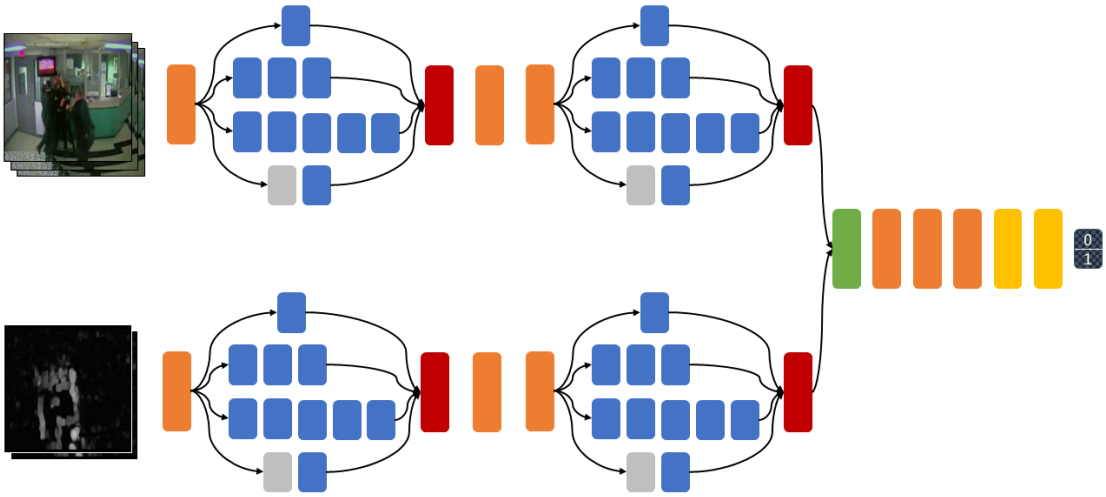


Figure 3.5: Graphical representation of the proposed architecture: the orange blocks represent a pseudo-3D block, the red blocks exemplify the Concatenate layers, the green block is the Multiply layer, the yellows blocks are the two final Fully Connected layers, whereas the illustration of the four employed Improved Inceptionv3 3D blocks is consistent with the visualization in Fig.3.2.

Comprehensively, it is a two-stream model with input size $64 \times 224 \times 224 \times 5$, that is, $64 \times 224 \times 224 \times 3$ for the RGB channel and $64 \times 224 \times 224 \times 2$ for the OF

channel. The number 64 indicates the sparse sample of 64 frames from each video for dimensionality reduction, done in coherence with [87].

Each channel is composed of a pseudo-3D block, i.e., the sequence of a S-convolution, a T-convolution, and of a Max Pooling 3D layer, to which follows the first of the two novel Improved Inceptionv3 3D blocks adopted for each channel. After two successive pseudo-3D blocks and the second Improved Inceptionv3 3D block, the two branches are then multiplied, by leveraging the use of the sigmoid activation function in the last convolutional layers of the OF channel, as indicated in Tab. 3.4.

It is important to add that, for dimensionality compatibility, the Max Pooling 3D layer of the last pseudo-3D block of the optical flow channel, named OF Block_3, has filter of size $1 \times 3 \times 3$ instead of $1 \times 2 \times 2$ as in all the other pseudo-3D blocks for both the optical flow and the RGB branches. In this way, the outputs of the last Concatenate layers of the second Improved Inceptionv3 3D block of each channel are equal, $64 \times 18 \times 18 \times 112$, thus enabling the product of these two feature maps using the optical flow branch as a temporal pooling gate by leveraging the sigmoid activation function in the last OF convolutional layers in place of the Rectified Linear Unit.

After the fusion block, which is also composed of a Max Pooling 3D layer, three successive pseudo-3D blocks follow, with the main features being the presence of 64 filters for each convolutional layer and the cubic nature of the three Max Pooling layers. This choice has been made in order to reach a proper dimensionality reduction of the feature maps, with the output dimension of the last layer of the Full_Block_3 being $1 \times 2 \times 2 \times 128$.

Finally, a Flatten layer has been introduced to unroll the map into a flat shape of 512 elements, with $512 = 2 \times 2 \times 128$. Three Fully Connected layers, or, as defined in Keras, Dense layers, conclude the architectures, with the first one being characterized by a Dropout regulator of 0.25 at training time, the second one with no regularization due to the smaller dimensionality, and the third one being the prediction layer.

Since the problem is issued as a binary problem, the sigmoid activation function should have naturally been used to obtain as the final output the binary label. However, by implementing the softmax activation function, it is possible to act on the output of the form (P(Fight),P(Non-Fight)) with P(Fight)+P(Non-Fight)= 1 with a different threshold operator: this possibility has been therefore left open in this work.

Comprehensively, the model is composed of a total of 536,370 learnable parameters for a file size in the .h5 format equal to just 2.9 MB.

Block	Layer	Filter Size or Output Dimension	Parameters
Input	Lambda	$64 \times 224 \times 224 \times 5$	
RGB Channel	*see Tab. 3.3*		108656
OF Channel	*see Tab. 3.4*		108512
Fusion	Multiply	$64 \times 18 \times 18 \times 112$	
Pooling	MaxPool3d	$8 \times 1 \times 1$	
Full_Block_1	Conv3d	$1 \times 3 \times 3$@64	64576
	Conv3d	$3 \times 1 \times 1$@64	12352
	MaxPool3d	$2 \times 2 \times 2$	
Full_Block_2	Conv3d	$1 \times 3 \times 3$@64	36928
	Conv3d	$3 \times 1 \times 1$@64	12352
	MaxPool3d	$2 \times 2 \times 2$	
Full_Block_3	Conv3d	$1 \times 3 \times 3$@64	73856
	Conv3d	$3 \times 1 \times 1$@64	49280
	MaxPool3d	$2 \times 2 \times 2$	
Final_Block	Flatten	1×512	
	Dense	128 with Dropout=0.25	65664
	Dense	32	4128
Prediction	Dense	2	66

Table 3.2: Full model's architecture.

3.4.1 Ablation Study

In order to analyze the contribution of each of the presented components to the final model's performance, an ablation study has been carried out, involving a total of 4 models, as follows:

- Full_Model (or FT-LFN-Full_Model), that is the model as described beforehand and in the tables below;
- GF-Full_Model, that is the same architectural network as the previous one, with the exception of using Gunnar Farnebäck's algorithm for the optical flow calculation in place of the Fine-Tuned LiteFlowNet2;
- FT-LFN-RGB_Model, which is a reduced version of the Full_Model involving only the RGB channel;
- FT-LFN-OF_Model, which is a reduced version of the Full_Model involving only the OF branch.

In conclusion, in all the experiments the FT-LiteFlowNet2 model has been used for the sparse optical flow estimation necessary for the newly introduced intelligent cropping strategy described in this chapter since little relevance would have had a more fine-grained ROI construction via a point-wise elaborate computation.

Block		Layer	Filter Size or Output Dimension	Parameters
Input		Lambda	$64 \times 224 \times 224 \times 3$	
RGB Block_1		Conv3d	$1 \times 3 \times 3$@16	448
		Conv3d	$3 \times 1 \times 1$@16	784
		MaxPool3d	$1 \times 2 \times 2$	
Improved Inceptionv3 3D Block_1	*Branch_1*	Conv3d	$1 \times 1 \times 1$@16	272
	Branch_2	MaxPool3d	$1 \times 3 \times 3$	
		Conv3d	$1 \times 1 \times 1$@32	544
	Branch_3	Conv3d	$1 \times 1 \times 1$@16	272
		Conv3d	$1 \times 3 \times 3$@32	4640
		Conv3d	$3 \times 1 \times 1$@32	3104
	Branch_4	Conv3d	$1 \times 1 \times 1$@16	272
		Conv3d	$1 \times 3 \times 3$@32	4640
		Conv3d	$3 \times 1 \times 1$@32	3104
		Conv3d	$1 \times 3 \times 3$@32	9248
		Conv3d	$3 \times 1 \times 1$@32	3104
	Branches_Fusion	Concatenate	$64 \times 74 \times 74 \times 112$	
RGB Block_2		Conv3d	$1 \times 3 \times 3$@32	32288
		Conv3d	$3 \times 1 \times 1$@32	3104
		MaxPool3d	$1 \times 2 \times 2$	
RGB Block_3		Conv3d	$1 \times 3 \times 3$@32	9248
		Conv3d	$3 \times 1 \times 1$@32	3104
		MaxPool3d	$1 \times 2 \times 2$	
Improved Inceptionv3 3D Block_2	*Branch_1*	Conv3d	$1 \times 1 \times 1$@16	528
	Branch_2	MaxPool3d	$1 \times 3 \times 3$	
		Conv3d	$1 \times 1 \times 1$@32	1056
	Branch_3	Conv3d	$1 \times 1 \times 1$@16	528
		Conv3d	$1 \times 3 \times 3$@32	4640
		Conv3d	$3 \times 1 \times 1$@32	3104
	Branch_4	Conv3d	$1 \times 1 \times 1$@16	528
		Conv3d	$1 \times 3 \times 3$@32	4640
		Conv3d	$3 \times 1 \times 1$@32	3104
		Conv3d	$1 \times 3 \times 3$@32	9248
		Conv3d	$3 \times 1 \times 1$@32	3104
	Branches_Fusion	Concatenate	$64 \times 18 \times 18 \times 112$	

Table 3.3: Structure of the RGB Channel.

Block		Layer	Filter Size or Output Dimension	Parameters
Input		Lambda	$64 \times 224 \times 224 \times 2$	
OF Block_1		Conv3d	$1 \times 3 \times 3$@16	304
		Conv3d	$3 \times 1 \times 1$@16	784
		MaxPool3d	$1 \times 2 \times 2$	
Improved Inceptionv3 3D Block_1	*Branch_1*	Conv3d	$1 \times 1 \times 1$@16	272
	Branch_2	MaxPool3d	$1 \times 3 \times 3$	
		Conv3d	$1 \times 1 \times 1$@32	544
	Branch_3	Conv3d	$1 \times 1 \times 1$@16	272
		Conv3d	$1 \times 3 \times 3$@32	4640
		Conv3d	$3 \times 1 \times 1$@32	3104
	Branch_4	Conv3d	$1 \times 1 \times 1$@16	272
		Conv3d	$1 \times 3 \times 3$@32	4640
		Conv3d	$3 \times 1 \times 1$@32	3104
		Conv3d	$1 \times 3 \times 3$@32	9248
		Conv3d	$3 \times 1 \times 1$@32	3104
	Branches_Fusion	Concatenate	$64 \times 112 \times 112 \times 112$	
OF Block_2		Conv3d	$1 \times 3 \times 3$@32	32288
		Conv3d	$3 \times 1 \times 1$@32	3104
		MaxPool3d	$1 \times 2 \times 2$	
OF Block_3		Conv3d	$1 \times 3 \times 3$@32	9248
		Conv3d	$3 \times 1 \times 1$@32	3104
		MaxPool3d	$1 \times 3 \times 3$	
Improved Inceptionv3 3D Block_2	*Branch_1*	Conv3d (sigmoid)	$1 \times 1 \times 1$@16	528
	Branch_2	MaxPool3d	$1 \times 3 \times 3$	
		Conv3d (sigmoid)	$1 \times 1 \times 1$@32	1056
	Branch_3	Conv3d	$1 \times 1 \times 1$@16	528
		Conv3d (sigmoid)	$1 \times 3 \times 3$@32	4640
		Conv3d (sigmoid)	$3 \times 1 \times 1$@32	3104
	Branch_4	Conv3d	$1 \times 1 \times 1$@16	528
		Conv3d	$1 \times 3 \times 3$@32	4640
		Conv3d	$3 \times 1 \times 1$@32	3104
		Conv3d (sigmoid)	$1 \times 3 \times 3$@32	9248
		Conv3d (sigmoid)	$3 \times 1 \times 1$@32	3104
	Branches_Fusion	Concatenate	$64 \times 18 \times 18 \times 112$	

Table 3.4: Optical Flow Channel: the use of the sigmoid activation function in the last convolutional layers is emphasized.

Chapter 4

Experimental Results

4.1 Data Preparation

As previously mentioned, the RWF-2000 dataset, outlined in Sect. 2.3.2, has been used in its integrity for all the experiments carried out in the present work. Since at the time, it was not directly and publicly available, access to its usage has been granted for this purpose by the authors, Ming Cheng et al., via a suitable request documentation to the Speech and Multimodal Intelligent Information Processing (SMIIP) lab at the Duke Kunshan University, China.

The original $80\% - 20\%$ split into Training and Validation set has been kept, to render the performance evaluation comparable with the state of the art, precisely due to being delivered under the same conditions. Moreover, since the 2000 clips of which it is composed are extracted from about 1000 unique videos, a considered approach has been carried out by the authors of the dataset in the partition, which this work has not the intention to alter.

As a matter of fact, having clips from the same video present in both the training and the validation sets would have caused data leakage, an undesirable problem to avoid at all costs in a model training insofar as the network would have had already available information extremely similar to what it is trying to predict, undermining both the fairness of the obtained results and the model's generalization capability.

The first pre-processing operation performed has been the conversion of all the 2000 clips from the original `.avi` format into a more common and web-friendly MPEG-4 codec, coherently with the assumption of the model being used on clips at the disposal.

Secondly, the intelligent video cropping technique described in Sect. 3.2 has been operated, using the estimated optical flow information via the fine-tuned LiteFlowNet2 model. After this step, all the 2000 videos had a spatial resolution of 224×224 pixels, as required by the architecture, either by the cropping procedure

or by simply resizing the whole frame.

The selection of 64 frames from the 150 of which each clip is composed has been performed via uniform sampling, that is sampling under a uniform probability distribution. After the selection of their indices, in the RGB channel, the frames were simply extracted via the `cv2` library and the `VideoCapture` functionality.

On the other hand, for the OF branch also the frame at interval $i + 5$ was extracted for each f_i with i being the original index of the frame in the global clip; afterward, the pairs of frames were converted into an 8-bit grey-scale color method and finally, the optical flows were computed via the fine-tuned LiteFlowNet2 or via Gunnar Farnebäck's algorithm, obtaining 128 vector maps (64 for each axis), stored in the same `.npy` vector as the RGB frames.

The constitution of a Data Generator was made necessary in order to parallelize the work; moreover, all the data augmentation techniques have been processed at runtime via the aforementioned Data Generator class.

The augmentation methods implemented with their respective ranges were the following:

- Random horizontal flips of the whole video with a probability of 0.5;
- Random color jitters of the whole video in the HSV space, that is the variation of the saturation of the whole clip with uniform distribution in the interval [-0.2,0.2] and of the hue value in the interval of [-30°,30°]; additional suitable controls of the final value of saturation and hue have been added to avoid going out of scale, as follows: $s|_{s<0} = 0$, $s|_{s>1} = 1$; $h|_{h<0^\circ} = 0^\circ$, and $h|_{h>360^\circ} = 360^\circ$;
- Random video horizontal shears of the whole video, that is the appliance of a transformation matrix with a probability of 0.5 which transforms each pixel (x, y) into $(x + \alpha \cdot y)$, hence forming a "parallelogrammed" frame video; the filling of the blank portion of the frame has been made via the "nearest" method, that is, the horizontal application for each row of the same pixel value as the last pixel on that edge in the deformed image;
- Random pixels shifts of the whole video, with the range in the interval of [0,4] pixels to comply with the padding envisaged by the cropping strategy;
- Gaussian blur of the whole video with $\sigma = [0,3]$;
- Random noise of the type "salt and pepper" [121] via the `random_noise` function with argument value = 0.5 which means an equal ratio of salt and pepper noise;
- Random brightness shifts in the range of [-30,30] with a probability of 0.5 for each frame, thus not illuminating or darkening the whole video clip but simulating flash effects and lights switching on/off; this augmentation procedure

has been carried out only in the RGB channel to refrain from interfering with the optical flow estimation.

4.2 Model Training

Before discussing the training parameters, it is important to note that, for time-saving reasons, the deeper hyper-parameter tuning has been performed relying only on the Full Model's performance, adapting the obtained observations on the other tested models. This could have resulted in slightly under-optimal performance results of the latter, yet fully maintaining the validity of the experiments.

As far as the hyper-parameters are concerned, the tested optimizers have been:

- Adam optimizer, which stands for Adaptive Moment Estimation and calculates an exponential moving average of the gradient and the squared gradient, controlled in their decay rates by two parameters β_1 and β_2, usually chosen close to 1.0 and specifically put equal to 0.9 and 0.999, respectively;
- SGD optimizer with momentum, which is a component that accelerates the gradient descent in the relevant direction and dampens fluctuations, equal to 0.9 and weight decay rate equal to $1 \cdot 10^{-6}$;

In both cases, the reported values have been the result of a random search technique to explore the hyper-parameters space: several starting values for each of the aforementioned hyper-parameters have been handpicked leveraging the empirical best practice available in the literature, which also suggested fixing $\epsilon = 1 \cdot 10^{-8}$ for the Adam optimizer, and the results of the arising random combinations have been studied to choose the best combination in each case.

The starting learning rate has been then put equal to $1 \cdot 10^{-2}$ with two possible callbacks for its adaptation:

- A fixed scheduler, which reduced the learning rate to $\frac{1}{10}$ of the original every 10 epochs;
- An adaptive scheduler via the `ReduceLROnPlateau` callback supplied by the Tensorflow API, with factor =0.5 and "patience" equal to 3 epochs, thus halving the learning rate if the observed metric (i.e., validation loss) has stopped improving over the last observation window parameterized with the patience value.

No particular improvement has been noted with the use of the latter, therefore the fixed scheduler has been chosen for its computational lightweight.

Finally, four batch size dimensions have been tested, namely 32, 16, 8, and 4, with a consequent number of processes to spin up due to the process-based

threading procedure, in Keras, `workers`, equal to half of the respective batch size. Moreover, since a Data Generator has been used, the maximum queue of the generator had to be picked: despite the default value being 10, it was found that setting `maxqueuesize=8` proved to be more performing.

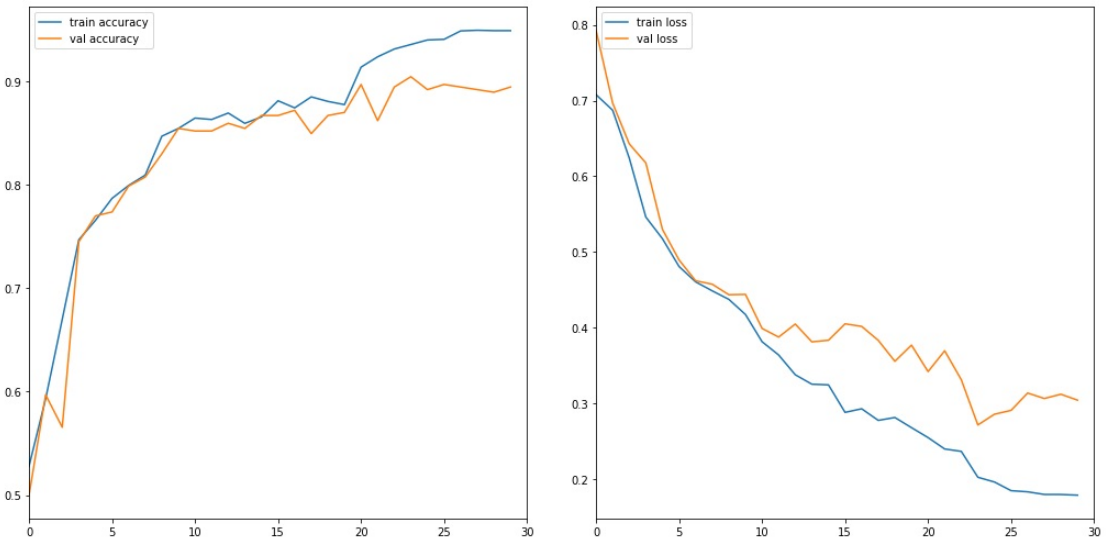
In conclusion, the final choice has been the use of the Adam optimizer and of a batch size equal to 8, therefore a number of concurrent processes equal to 4.

The number of epochs has been set to 30, with 200 iterations per epoch for a total of 320000 steps, with the best model on the validation loss being saved before the occurrence of the eventual overfitting phenomenon.

The selected loss function has been the categorical cross-entropy loss, which is constituted of a softmax activation function on top of a softmax loss, and it is perfectly suitable for one-hot encoded label vectors as in this case. Needless to say, the only relevant measured metric has been the accuracy of the prediction.

All the model training has been performed on the Google Colab platform, mainly leveraging the NVIDIA® T4 and NVIDIA® V100 GPUs, with 52GB of RAM and 2 virtual CPUs, that are physical CPUs assigned to a virtual machine, with processors Intel® Xeon® @2.20 GHz. On these conditions, the training time took a median of $\approx$ 449 seconds per epoch.

Fig. 4.1 shows the full model training results for the training and validation sets, whereas in Fig. 4.2 the same representation is delivered for the two single-branch models.



(a) **(b)**

Figure 4.1: The performance of the Full model.

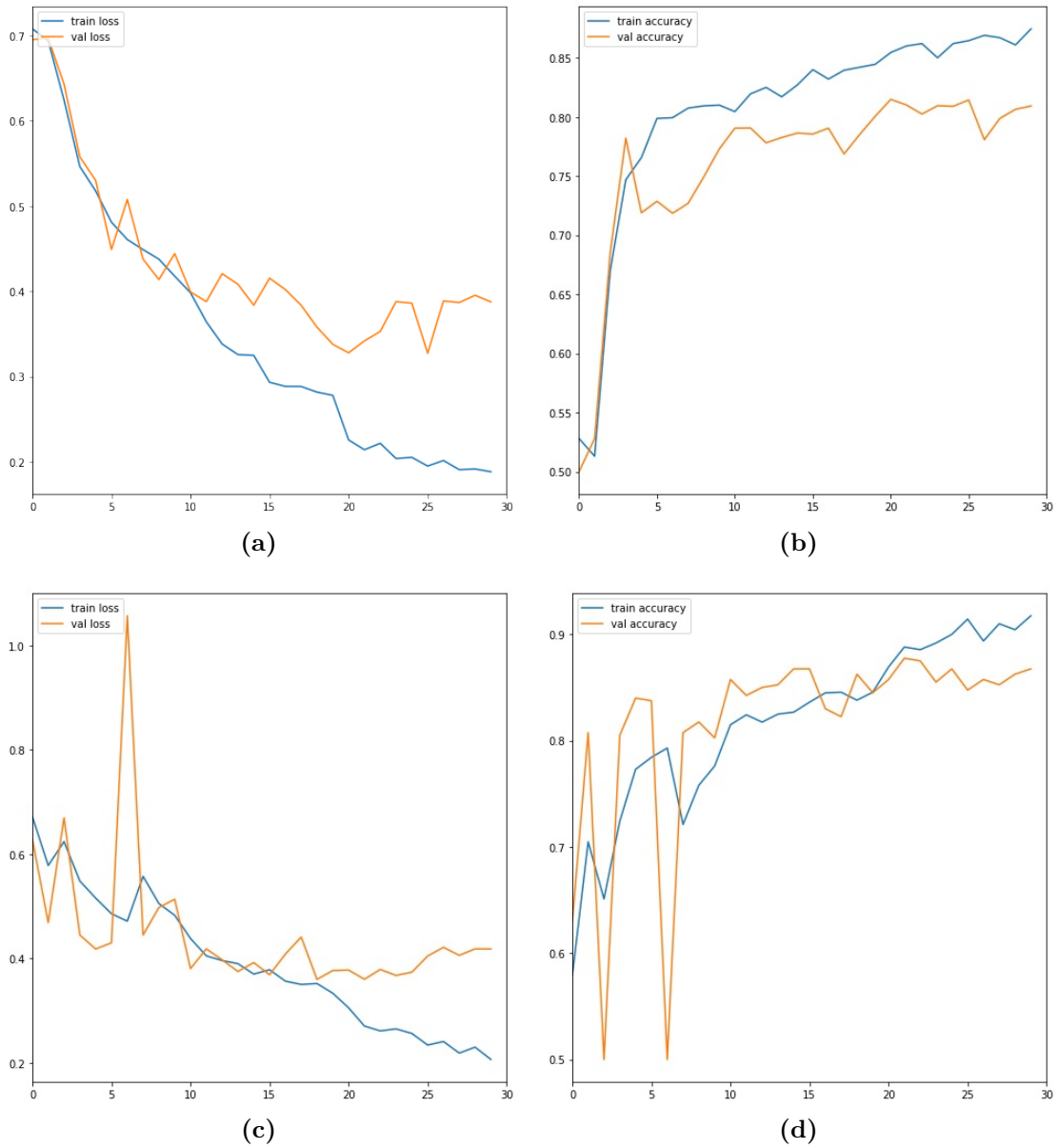


Figure 4.2: The performance of the two single branch models over the training and validation set: the slight sub-optimal hyper-parameter tuning is evident in the more noisy graphs with respect to Fig. 4.1.

4.3 Performance Evaluation

The results are reported in Tab. 4.1 with a comparison with the state of art models available and tested on the same dataset. The performance values of these reference models and their specifications have been obtained from the dedicated publications reported inside the table, or via the PapersWithCode benchmark table for the dataset [122] excluding the ones trained with additional data[1].

Model	**Accuracy (%)**	**Parameters (M)**
C3D [79]	82.75	94.8
I3D (RGB Only)	85.75	12.3
I3D (OF Only)	75.50	12.3
I3D [85]	81.50	24.6
Flow Gated Network (RGB Only)	84.50	0.248
Flow Gated Network (OF Only)	75.50	0.248
Flow Gated Network (C3D Fusion)	85.75	0.507
Flow Gated Network [87]	87.25	0.273
SepConvLSTM-C (RGB Only)	83.75	0.186
SepConvLSTM-M [124]	89.75	0.333
SPIL ConvNet [125]	89.30	-
FT-LFN-Model (RGB Only)	87.75	0.428
FT-LFN-Model (OF Only)	81.40	0.427
FT-LFN-Full Model	**90.45**	0.536

Table 4.1: Results compared with the state of the art; the best result overall is reported in bold; regarding the SPIL ConvNet model, no detailed information on the weights count or on the FLOPS complexity was possible to extract from the reference paper, nor the code implementation was made available by the authors, hence the absent number.

As shown, the Full Model has outperformed the state of the art best performing networks on the RWF-2000 dataset, reaching an impressive 90.45% accuracy on the validation set, for an increase of +0.7% with respect to the second best model, SepConvLSTM-M, in the performance.

Just as important, it is to note that the complexity of the model is in the same order of magnitude with two of the best-performing networks, SepConvLSTM-M and Flow Gated Network, and way more lightweight than the de-facto standard architectures C3D and I3D (respectively, ≈ 177 and ≈ 46 times bigger).

[1]It is the case of the two-stream SSHA network by Hamid Mohammadi and Ehsan Nazerfard in [123], which reaches a 90.3% accuracy by training on another dataset together with RWF-2000.

The results of the single branch models are equally as impressive, with the RGB-Only model being the best performing model among those using only the extracted frames as input, with an out-performance of +2% in accuracy. At the same time, the OF-Only model has crushed the competition, reaching a +5.9% in the evaluated metric, proving the quality of the proposed architecture and of the fine-tuned LiteFlowNet2 for optical flow estimation.

For good measure, the fusion procedure is effective to the extent that the Full Model is clearly the most precise of the three, therefore gaining more insight with the optical flow branch acting as a temporal pooling gate, and further processing the obtained feature maps. For thoroughness, compare this behavior with the I3D case, where the Fusion model formed of the simple average of the output of the two branches experiences a performance degradation with respect to the RGB-Only model.

The visualization of the application of the FT-LFN-Full Model on a first-time video footage is presented in Fig. 4.3, with both cases of correct and incorrect predictions, to demonstrate its generalization capabilities and straits.

Finally, Tab. 4.2 shows the pre-processing speed and the inference time (on an already pre-processed clip) of the two Full Model versions, the one using the fine-tuned LiteFlowNet2 for optical flow estimation and the one adopting Gunnar Farnebäck's algorithm. This data has been collected under the same conditions, i.e., making the model run on Google Colab with NVIDIA® T4 GPUs, and averaged over several runs on the validation set.

Model	**Accuracy (%)**	**Pre-processing (s)**	**Inference (s)**
FT-LFN-Full Model	90.45	**0.63**	1.25
GF-Full Model	90.46	4.35	1.26

Table 4.2: Speed comparison of the two models: by the pre-processing time it is referred to the transformation of a 5-second clip into the suitable $64 \times 224 \times 224 \times 5$ input format, while the inference time is the time for predicting the output label of an already pre-processed input clip.

While no significant difference can be encountered in the accuracy percentage and in the inference time, the pre-processing speed of the two models greatly differs, to the benefit of the FT-LFN-Full Model, which shows potential for being used in a real-time scenario.

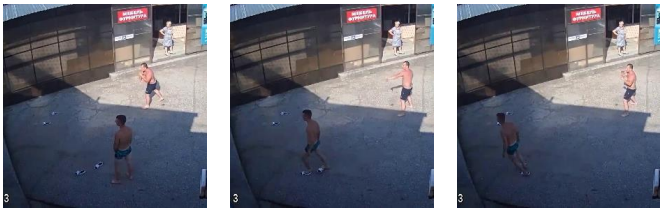
Clip Frames	Ground Truth	Predicted Label
	No Fight	No Fight
	Fight	Fight
	No Fight	Fight

Figure 4.3: Qualitative results of the proposed FT-LFN-Full Model for violence detection on a new, unseen video, retrieved from the YouTube platform: the first two rows contain examples of correctly predicted clips; the last row presents a failure case, where probably the compound movement of the subjects, in particular the shadow-boxing of one of them, may have led the model towards an incorrect prediction despite the absence of violence in the clip.

4.4 Interactive Utility

A basic interface to aid the usability of the trained model has been produced: it consists of a utility by which a video clip can be uploaded and it is automatically pre-processed and evaluated. It has been built on the Google Colab platform mainly leveraging the same pre-processing as the dataset preparation, with the additional use of the `ffmpeg` and `cv2` library to perform the necessary supplemental transformations (namely, the measurement of the length of the video and the split into 5-second clips).

After each clip is passed through the model and the predicted output label has been obtained, the user is presented with two types of output:

- A log file having name structure as `"video_name_log.txt"` containing the moments of time of the whole video where some violent action has been detected (e.g., in the form of `"Detected violence starting at 0:15:10 and ending at 0:15:15`);
- The clips labeled as containing abnormal activities for further manual investigation and skimming; the "normal" clips, on the other hand, are discarded.

Obviously, in the case of continuative violence, that is, consecutive clips all labeled as "1-Fight", their single outputs are merged into a unique video file and aggregate starting and ending points in the textual log. This procedure is as a whole visualized in Fig. 4.4.

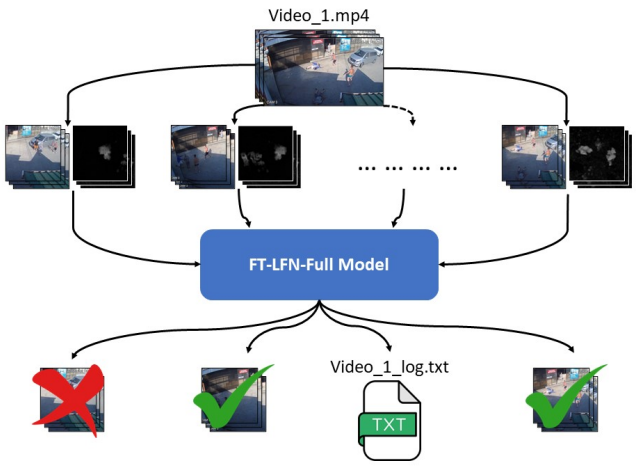


Figure 4.4: The application framework of the trained model on a new unseen video.

Chapter 5

Conclusions

This work had the aim of creating an interface with an automated workflow to help rationalize the workload of legal personnel exploring documentary evidence in the form of visual footage. The envisaged scenario of the application was a mostly offline environment, forecasting a subsequent human manual intervention after the model provided the user with a large but not definitive projection of video fragments, confidently containing the "abnormal" or "violent" portions in them.

The proposed framework reflects the expectations that prompted this project, and it is coherent with respect to the methods and techniques of reference in the case study. Moreover, the simple envisaged ETL architecture only requests the manual prompt of the video file or folder from the user, with the remainder of operations being all automatized and the final production of both a visual and a textual output log.

The availability of two extensive benchmark datasets, UCF-Crime and RWF-2000, has been key in pursuing this objective, relieving the need for the construction of a novel one via techniques such as web scraping and extraction from online public video-sharing platforms. Although mostly automatic, this process would have still required attention, programming expertise, and significant manual intervention in the cleaning and preparation phase, which altogether would have shortened the time available for the other phases of the work.

In-depth, the proposed final model stemmed from the intuition that, among the endless architectural alternatives, two-stream networks could still be relevant in the anomaly detection task if the problem is modeled in a supervised learning scenario and could outperform the other solutions. Additionally, the fusion strategy of the two informative branches has proven to be effective, in coherence with the latest findings in literature: the consequent re-elaboration of the feature maps via successive additional pseudo-3D blocks has enabled the achievement of satisfying results which improve the current state-of-the-art performance among the publicly

available models on the same dataset.

In this regard, it is important to underline that the "recurrent" informative aspect, which is core in a frame sequence and is explicitly enforced in other proposals such as RNNs and LSTMs, in this architectural schema is provided by the optical flow itself, since it actually performs, mathematically speaking, an iterative optimization for the flow fields over time. The utilization of a CNN-based estimation in place of the proper algorithm computation turned out to be equally effective and with great time and computational savings.

Additionally, it is possible to claim that the feature maps representation coming from the last layers of the model are a very compact and informative descriptor in the manner of the C3D features, albeit being definitely more space-saving and, according to the results, of higher expression quality, summoning up the characteristics of the moving elements of a short clip in a tensor and even possibly exploitable as an input for other processing procedures.

Finally, the innovative intelligent optical flow-based cropping strategy and the improved Inception 3D block both had their merits in the achievement of the satisfactory results of the present project.

However, some critical issues have still emerged and have to be properly addressed.

First of all, from a global point of view, some ethical issues in performing automatic anomaly detection arise, in particular for the legal applications of deep learning models and how these may affect humans' well-being. It is demonstrated in literature by several researchers, such as Kate Crawford in [126], that networks trained on biased datasets encode and reproduce the bias in their predictions, therefore procuring damage to the already more exposed social groups.

In the specific case of violence detection, the social implications of an inappropriate use of a such model, for example leveraging the model by itself to automatically persecute the depicted subject dismissing the human approval part of the matter, could be of extreme interest and to avoid at all costs.

Secondly, the framework is not as it is directly applicable for run-time monitor aiding: although the starting objective was an off-line application, and from this point of view the work has succeeded, as of now the need for humans intervention in the setting phase and file selection could be of great limitation, which collides with a real-time scenario. Even in the projected study case, which does not require the automatizing of the prompt command, it could still be a sub-optimal solution.

Finally, the generalization capabilities of the proposed model are dependent on the representations provided at training time, which, although made as extensive as possible, could not be really fully exhaustive in the depiction of abnormal phenomena due to the intrinsic extremely diversified nature of them: therefore, in the reality, this problem remains open.

5.1 Future Works

Several future expansions to this work could be envisaged: hereinafter, a few of them are reported.

Firstly, a more user-friendly interface, with the aid of some powerful solutions available to date such as PowerBI Desktop or Microsoft Azure, could be produced. Additionally, without the need for third-party software, several Python libraries exist for the construction of more complete dashboards or web applications whose usage could facilitate the user experience in this application case. The textual log could be likewise enhanced.

On another note, the model itself could be improved, especially in the fine-tuning of the LiteFlowNet2 architecture utilized for the optical flow estimation; although the results suggest that the improvement that could be brought by such a more extensive procedure are marginal and not really relevant for the significant improvement of the accuracy of the model, it could still be performed since even a slight model performance improvement could be in a specific setting important.

However, more significantly, the UCF-Crime dataset could be transformed, that is, elaborated in order to obtain 5-sec clips, with the binary label applied on the basis of the ground truth timing of the anomaly action in the "positive" videos. This operation, which the author recognizes as really intensive and time-consuming, could nevertheless be the most practical idea to further improve this model, by supplying a gargantuan comprehensive dataset resulting from the combination of the two. This process would improve also the generalization capabilities of the model, leveraging the most varied actions depicted in the UCF-Crime dataset with respect to the RWF-2000 dataset. Of course, the need for suitable computational resources should be addressed, which was a key down-factor limitation in this work's boundaries.

Additionally, the framework could be extended in order to create a pipeline to function at almost-real-time, that is with a small delay in the range of a few seconds, to aid the active monitoring of security footage: this delay corresponds to the sum of the pre-processing phase and of the inference time, and it is for a single clip in the reported results acceptable, compliant with this reasonable range. However, at present, it is realistically under-reported in this work, since the presented timings do not take into account the output presentation, which occurs only at the end of the last clip: therefore, although the prediction for the first clip would be made after 1 or 2 seconds, its output would be made available only after the processing of the entire length of the video. It is possible to imagine that, by pipelining the entire framework and possibly leveraging different computational units for each assignment, this issue could be significantly lessened. Likewise importantly, with this procedure, the final delay could remain in this order of magnitude even with a large number of CCTV cameras concomitant transmissions. It is important to

note that, in order to address this issue, the structure of the model itself needs to be slightly modified, possibly avoiding leveraging the Generator class, which puts an input strain on the evaluation phase.

Lastly, this work has not approached the industrial solutions in the private-security field, notably intelligent automated CCTV cameras, possibly integrated with PTZ capabilities. Important improvements could be done in this sector, by adapting the solution to the technology in the field of use: a possible application could be, as an example, to dictate the movement of the camera via real-time violence estimation, for example enlarging (i.e., zooming out) the scene or automatically tilting the camera if suspicious actions occur in the proximity of the edges of the frame, making possible to track them and the possible perpetrators and not making them go off-screen.

Bibliography

[1] Cisco. *Consumer Internet Video Traffic*. 2018. URL: https://www.cisco.com/c/dam/m/en_us/solutions/service-provider/vni-forecast-highlights/pdf/Global_Device_Growth_Traffic_Profiles.pdf (cit. on p. 1).

[2] *More than 500 hours of content are now being uploaded to Youtube every minute*. May 2019. URL: https://www.tubefilter.com/2019/05/07/number-hours-video-uploaded-to-youtube-per-minute/ (cit. on p. 1).

[3] Evening Standard. *We're watching you: 'britons caught on CCTV 70 times a day'*. Apr. 2012. URL: https://www.standard.co.uk/hp/front/we-re-watching-you-britons-caught-on-cctv-70-times-a-day-6573202.html (cit. on p. 1).

[4] Eric L. Piza, Brandon C. Welsh, David P. Farrington, and Amanda L. Thomas. *Closed-Circuit Television (CCTV)* -. Oct. 2021. URL: https://www.college.police.uk/research/crime-reduction-toolkit/cctv (cit. on p. 2).

[5] Pawel Pawlowski, Adam Dabrowski, Julian Balcerek, Adam Konieczka, and Karol Piniarski. «Visualization techniques to support CCTV operators of smart city services». In: *Multimedia Tools and Applications* 79 (Aug. 2020). DOI: 10.1007/s11042-020-08895-6 (cit. on p. 2).

[6] Peter Kinget. *The World is Analog*. Jan. 2021. URL: https://circuitcellar.com/insights/tech-the-future/kinget-the-world-is-analog/ (cit. on p. 3).

[7] *Digital Images*. URL: https://www.encyclopedia.com/computing/news-wires-white-papers-and-books/digital-images (cit. on p. 3).

[8] W.H. Freeman Sabins F. F. Jr. and Co. «Remote Sensing. Principles and Interpretation, 2nd ed. xi 449 pp. New York». In: *Geological Magazine* 124.3 (1987). DOI: 10.1017/S0016756800016575 (cit. on p. 3).

[9] *How are images stored on a computer: Grayscale & RGB image formats*. Mar. 2021. URL: https://www.analyticsvidhya.com/blog/2021/03/grayscale-and-rgb-format-for-storing-images/ (cit. on p. 3).

[10] James Clerk Maxwell. «XVIII.—Experiments on Colour, as perceived by the Eye, with Remarks on Colour-Blindness». In: *Transactions of the Royal Society of Edinburgh* 21.2 (1857), pp. 275–298. DOI: 10.1017/S0080456800032117 (cit. on p. 4).

[11] Noor Ibraheem, Mokhtar Hasan, Rafiqul Zaman Khan, and Pramod Mishra. «Understanding Color Models: A Review». In: *ARPN Journal of Science and Technology* 2 (Jan. 2012) (cit. on p. 5).

[12] Wikimedia Commons. *RGB, HSL and HSV*. Feb. 2023. URL: https://commons.wikimedia.org/ (cit. on p. 5).

[13] Wikipedia. *Digital Video — Wikipedia, the Free Encyclopedia*. [Online; accessed on August 2022]. 2011. URL: https://en.wikipedia.org/wiki/Digital_video (cit. on p. 5).

[14] «Introduction to Video Compression». In: *Handbook of Image and Video Processing (Second Edition)*. Ed. by AL BOVIK. Second Edition. Communications, Networking and Multimedia. Burlington: Academic Press, 2005, p. 775. ISBN: 978-0-12-119792-6. DOI: https://doi.org/10.1016/B978-0-12-119792-6.50145-5. URL: https://www.sciencedirect.com/science/article/pii/B9780121197926501455 (cit. on p. 6).

[15] Dmitriy Vatolin, Iwan Seleznev, and Maxim Smirnov. «Lossless Video Codecs Comparison '2007». In: *CS MSU Graphicsamp;Media Lab* (Mar. 2007). DOI: http://compression.ru/video/codec_comparison/pdf/msu_lossless_codecs_comparison_2007_eng.pdf (cit. on p. 6).

[16] Wikimedia Commons. *Screenshot from "Elephants Dream" to illustrate motion compensation in video compression*. (c) copyright 2006, Blender Foundation / Netherlands Media Art Institute / www.elephantsdream.org. 2009. URL: https://commons.wikimedia.org/wiki/File:Motion_compensation_example-original.jpg (cit. on p. 7).

[17] J. Valentim, P. Nunes, and F. Pereira. «IST MPEG-4 Video Compliant Framework». In: *Conf. on Telecommunications - ConfTele*. Vol. I. Apr. 2001, pp. 442–446 (cit. on p. 7).

[18] T. Silva and A. Navarro. «H.264 Video Transmission over 2x2 OFDM-MIMO». In: *IEEE International Conf. on Multimedia and Expo - ICME*. Vol. 1. July 2011, pp. 1–8 (cit. on p. 7).

[19] Thazni Aziz, D. Raveena, and Judie Dolly. *Motion Estimation and Motion Compensated Video Compression Using DCT And DWT*. 2012 (cit. on p. 7).

[20] G.J. Sullivan, J.R. Ohm, Woo-Jin Han, and T. Wiegand. «Overview of the high efficiency video coding standard». In: 22 (Dec. 2012), pp. 1648–1667 (cit. on p. 7).

[21] Jan De Cock, Aditya Mavlankar, Anush Moorthy, and Anne Aaron. «A large-scale video codec comparison of x264, x265 and libvpx for practical VOD applications». In: *Applications of Digital Image Processing XXXIX*. Ed. by Andrew G. Tescher. Vol. 9971. Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series. Sept. 2016, 997116, p. 997116. DOI: 10.1117/12.2238495 (cit. on p. 7).

[22] Jens-Rainer Ohm, Gary J. Sullivan, Heiko Schwarz, Thiow Keng Tan, and Thomas Wiegand. «Comparison of the Coding Efficiency of Video Coding Standards—Including High Efficiency Video Coding (HEVC)». In: *IEEE Transactions on Circuits and Systems for Video Technology* 22.12 (2012), pp. 1669–1684. DOI: 10.1109/TCSVT.2012.2221192 (cit. on p. 7).

[23] Salahuddin Swati, Khizar Hayat, and Zafar Shahid. *A watermarking scheme for High Efficiency Video Coding (HEVC)*. URL: https://doi.org/10.1371/journal.pone.0105613 (cit. on p. 8).

[24] Richard Szeliski. *Computer vision algorithms and applications*. 2011. URL: http://dx.doi.org/10.1007/978-1-84882-935-0 (cit. on p. 8).

[25] Hans Moravec. *Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover*. Tech. rep. Pittsburgh, PA: Carnegie Mellon University, Sept. 1980 (cit. on p. 8).

[26] Christopher G. Harris and M. J. Stephens. «A Combined Corner and Edge Detector». In: *Alvey Vision Conference*. 1988 (cit. on p. 8).

[27] Edward Rosten and Tom Drummond. «Machine Learning for High-Speed Corner Detection». In: *Computer Vision – ECCV 2006*. Ed. by Aleš Leonardis, Horst Bischof, and Axel Pinz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 430–443. ISBN: 978-3-540-33833-8 (cit. on p. 9).

[28] J. E. Bresenham. «Algorithm for computer control of a digital plotter». In: *IBM Systems Journal* 4.1 (1965), pp. 25–30. DOI: 10.1147/sj.41.0025 (cit. on p. 9).

[29] H.G. Barrow and J.M. Tenenbaum. «Interpreting line drawings as three-dimensional surfaces». In: *Artificial Intelligence* 17.1 (1981), pp. 75–116. ISSN: 0004-3702. DOI: https://doi.org/10.1016/0004-3702(81)90021-7. URL: https://www.sciencedirect.com/science/article/pii/0004370281900217 (cit. on p. 9).

[30] John Canny. «A computational approach to edge detection». In: *IEEE Transactions on pattern analysis and machine intelligence* 6 (1986), pp. 679–698 (cit. on p. 9).

[31] Irwin Sobel. «An Isotropic 3x3 Image Gradient Operator». In: *Presentation at Stanford A.I. Project 1968* (Feb. 2014) (cit. on p. 9).

[32] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. «A comparison of affine region detectors». In: *International Journal of Computer Vision* 65.1-2 (2005), pp. 43–72. DOI: 10.1007/s11263-005-3848-x (cit. on p. 9).

[33] D.G. Lowe. «Object recognition from local scale-invariant features». In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Vol. 2. 1999, 1150–1157 vol.2. DOI: 10.1109/ICCV.1999.790410 (cit. on p. 9).

[34] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. «SURF: Speeded Up Robust Features». In: *Computer Vision – ECCV 2006*. Ed. by Aleš Leonardis, Horst Bischof, and Axel Pinz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 404–417. ISBN: 978-3-540-33833-8 (cit. on p. 9).

[35] Bo Jiang, Yongyi Lu, Xiying Li, and Liang Lin. «Towards a solid solution of real-time fire and flame detection». In: *CoRR* abs/1502.00416 (2015). arXiv: 1502.00416. URL: http://arxiv.org/abs/1502.00416 (cit. on p. 9).

[36] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (4th Edition)*. Pearson, 2020. ISBN: 9780134610993. URL: http://aima.cs.berkeley.edu/ (cit. on p. 10).

[37] Guy Hindley, Olav B. Smeland, Oleksandr Frei, and Ole A. Andreassen. «3 - Big data and the goal of personalized health interventions». In: *Mental Health in a Digital World*. Ed. by Dan J. Stein, Naomi A. Fineberg, and Samuel R. Chamberlain. Global Mental Health in Practice. Academic Press, 2022, pp. 41–61. ISBN: 978-0-12-822201-0. DOI: https://doi.org/10.1016/B978-0-12-822201-0.00021-6. URL: https://www.sciencedirect.com/science/article/pii/B9780128222010000216 (cit. on p. 10).

[38] Mariette Awad and Rahul Khanna. «Machine Learning». In: *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers*. Berkeley, CA: Apress, 2015, pp. 1–18. ISBN: 978-1-4302-5990-9. DOI: 10.1007/978-1-4302-5990-9_1. URL: https://doi.org/10.1007/978-1-4302-5990-9_1 (cit. on p. 10).

[39] A. L. Samuel. «Some Studies in Machine Learning Using the Game of Checkers». In: *IBM J. Res. Dev.* 3.3 (July 1959), pp. 210–229. ISSN: 0018-8646. DOI: 10.1147/rd.33.0210. URL: https://doi.org/10.1147/rd.33.0210 (cit. on p. 10).

[40] Tom M. Mitchell. *Machine Learning*. New York: McGraw-Hill, 1997. ISBN: 978-0-07-042807-2 (cit. on pp. 11, 16).

[41] Ethem Alpaydin. *Introduction to Machine Learning*. 3rd ed. Adaptive Computation and Machine Learning. Cambridge, MA: MIT Press, 2014. ISBN: 978-0-262-02818-9 (cit. on p. 11).

[42] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018 (cit. on p. 11).

[43] Ved Vasu Sharma. *Weekly supervised learning-getting started with unstructured data*. Oct. 2020. URL: https://towardsdatascience.com/weekly-supervised-learning-getting-started-with-unstructured-data-123354dad7c1 (cit. on p. 12).

[44] Yuji Roh, Geon Heo, and Steven Euijong Whang. «A Survey on Data Collection for Machine Learning: a Big Data - AI Integration Perspective». In: *CoRR* abs/1811.03402 (2018). arXiv: 1811.03402. URL: http://arxiv.org/abs/1811.03402 (cit. on p. 12).

[45] Warren S. McCulloch and Walter Pitts. «A logical calculus of the ideas immanent in nervous activity». In: *The Bulletin of Mathematical Biophysics* 5.4 (1943), pp. 115–133. DOI: 10.1007/bf02478259 (cit. on p. 13).

[46] Kevin Gurney. «An introduction to neural networks». In: (2018). DOI: 10.1201/9781315273570 (cit. on p. 13).

[47] Ergün Akgün and Metin Demir. «Modeling Course Achievements of Elementary Education Teacher Candidates with Artificial Neural Networks». In: *International Journal of Assessment Tools in Education* 5 (Jan. 2018). DOI: 10.21449/ijate.444073 (cit. on p. 14).

[48] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016 (cit. on pp. 13, 15, 16).

[49] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. «Learning representations by back-propagating errors». In: *Nature* 323 (1986), pp. 533–536 (cit. on p. 13).

[50] Christian Janiesch, Patrick Zschech, and Kai Heinrich. «Machine learning and deep learning». In: *CoRR* abs/2104.05314 (2021). arXiv: 2104.05314. URL: https://arxiv.org/abs/2104.05314 (cit. on p. 14).

[51] Li Deng and Dong Yu. «Deep Learning: Methods and Applications». In: *Foundations and Trends® in Signal Processing* 7.3–4 (2014), pp. 197–387. ISSN: 1932-8346. DOI: 10.1561/2000000039. URL: http://dx.doi.org/10.1561/2000000039 (cit. on p. 15).

[52] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. «ImageNet Classification with Deep Convolutional Neural Networks». In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf (cit. on p. 18).

[53] Jason Brownlee. *How do convolutional layers work in Deep Learning Neural Networks?* Apr. 2020. URL: https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/ (cit. on p. 19).

[54] F. Chollet. *Deep Learning with Python*. Manning Publications Company, 2017. ISBN: 9781617294433. URL: https://books.google.de/books?id=Yo3CAQAACAAJ (cit. on p. 19).

[55] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. «Going Deeper with Convolutions». In: *CoRR* abs/1409.4842 (2014). arXiv: 1409.4842. URL: http://arxiv.org/abs/1409.4842 (cit. on p. 20).

[56] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. «Rethinking the Inception Architecture for Computer Vision». In: *CoRR* abs/1512.00567 (2015). arXiv: 1512.00567. URL: http://arxiv.org/abs/1512.00567 (cit. on pp. 20, 56, 57).

[57] François Chollet. «Xception: Deep Learning with Depthwise Separable Convolutions». In: *CoRR* abs/1610.02357 (2016). arXiv: 1610.02357. URL: http://arxiv.org/abs/1610.02357 (cit. on p. 20).

[58] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. «Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning». In: *CoRR* abs/1602.07261 (2016). arXiv: 1602.07261. URL: http://arxiv.org/abs/1602.07261 (cit. on p. 20).

[59] Richard Meyes, Melanie Lu, Constantin Waubert de Puiseau, and Tobias Meisen. «Ablation Studies in Artificial Neural Networks». In: *CoRR* abs/1901.08644 (2019). arXiv: 1901.08644. URL: http://arxiv.org/abs/1901.08644 (cit. on p. 22).

[60] Matthew D. Zeiler and Rob Fergus. «Visualizing and Understanding Convolutional Networks». In: *CoRR* abs/1311.2901 (2013). arXiv: 1311.2901. URL: http://arxiv.org/abs/1311.2901 (cit. on pp. 22, 42).

[61] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. «Large-Scale Video Classification with Convolutional Neural Networks». In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 1725–1732. DOI: 10.1109/CVPR.2014.223 (cit. on pp. 22, 24, 26).

[62] Francesco Camastra and Alessandro Vinciarelli. «Machine learning for audio, image and video analysis». In: *Springer* (2008). DOI: 10.1007/978-1-84800-007-0 (cit. on p. 23).

[63] Sarah Porter, Majid Mirmehdi, and Barry Thomas. «Temporal video segmentation and classification of edit effects». In: *Image and Vision Computing* 21 (Dec. 2003), pp. 1097–1106. DOI: 10.1016/j.imavis.2003.08.014 (cit. on p. 23).

[64] A. Mittal and L.-F. Cheong. *Addressing the Problems of Bayesian Network Classification of Video Using High-Dimensional Features*. 2004 (cit. on p. 23).

[65] Vakkalanka Suresh, C. Krishna Mohan, R. Kumara Swamy, and B. Yegnanarayana. «Content-based video classification using support Vector Machines». In: *Neural Information Processing* (2004), pp. 726–731. DOI: 10.1007/978-3-540-30499-9_111 (cit. on p. 23).

[66] T. Starner and A. Pentland. «Real-time American Sign Language recognition from video using hidden Markov models». In: *Proceedings of International Symposium on Computer Vision - ISCV*. 1995, pp. 265–270. DOI: 10.1109/ISCV.1995.477012 (cit. on p. 23).

[67] Rui Tan, Hong Huo, Jin Qian, and Tao Fang. «Traffic video segmentation using adaptive-K gaussian mixture model». In: *Advances in Machine Vision, Image Processing, and Pattern Analysis* (2006), pp. 125–134. DOI: 10.1007/11821045_13 (cit. on p. 23).

[68] Changsoo Je and Hyung-Min Park. «Optimized hierarchical block matching for fast and accurate image registration». In: *Signal Processing: Image Communication* 28.7 (2013), pp. 779–791. ISSN: 0923-5965. DOI: https://doi.org/10.1016/j.image.2013.04.002. URL: https://www.sciencedirect.com/science/article/pii/S0923596513000581 (cit. on p. 23).

[69] Zhigang Tu, Wei Xie, Dejun Zhang, Ronald Poppe, Remco Veltkamp, Baoxin Li, and Junsong Yuan. «A survey of variational and CNN-based optical flow techniques». In: 72 (Mar. 2019). DOI: 10.1016/j.image.2018.12.002 (cit. on pp. 24, 37).

[70] Gunnar Farnebäck. «Polynomial Expansion for Orientation and Motion Estimation». In: (Dec. 2002) (cit. on p. 24).

[71] Aaron Bobick and J.W. Davis. «The recognition of human movement using temporal templates». In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 23 (Apr. 2001), pp. 257–267. DOI: 10.1109/34.910878 (cit. on pp. 24, 25).

[72] Md. Atiqur Ahad. «Motion history images for action recognition and understanding». In: *SpringerBriefs in Computer Science* (2013). DOI: 10.1007/978-1-4471-4730-5 (cit. on p. 24).

[73] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. «Action Recognition by Dense Trajectories». In: *IEEE Conference on Computer Vision Pattern Recognition* (June 2011). DOI: 10.1109/CVPR.2011.5995407 (cit. on pp. 24, 25).

[74] Heng Wang and Cordelia Schmid. «Action Recognition with Improved Trajectories». In: *2013 IEEE International Conference on Computer Vision*. 2013, pp. 3551–3558. DOI: 10.1109/ICCV.2013.441 (cit. on p. 24).

[75] LukasBommes. *Extract frames and motion vectors from H.264 and MPEG-4 encoded video*. URL: https://github.com/LukasBommes/mv-extractor (cit. on p. 25).

[76] Maxim Kuklin (Xperience.AI). *Optical flow in opencv (c++/python)*. July 2021. URL: https://learnopencv.com/optical-flow-in-opencv/#dense-optical-flow (cit. on p. 25).

[77] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. *UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild*. 2012. DOI: 10.48550/ARXIV.1212.0402. URL: https://arxiv.org/abs/1212.0402 (cit. on p. 26).

[78] Jeff Donahue, Lisa Anne Hendricks, Marcus Rohrbach, Subhashini Venugopalan, Sergio Guadarrama, Kate Saenko, and Trevor Darrell. *Long-term Recurrent Convolutional Networks for Visual Recognition and Description*. 2014. DOI: 10.48550/ARXIV.1411.4389. URL: https://arxiv.org/abs/1411.4389 (cit. on pp. 26, 48).

[79] Du Tran, Lubomir D. Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. «C3D: Generic Features for Video Analysis». In: *CoRR* abs/1412.0767 (2014). arXiv: 1412.0767. URL: http://arxiv.org/abs/1412.0767 (cit. on pp. 27, 42, 44–46, 74).

[80] Alberto Montes, Amaia Salvador, and Xavier Giró-i-Nieto. «Temporal Activity Detection in Untrimmed Videos with Recurrent Neural Networks». In: *CoRR* abs/1608.08128 (2016). arXiv: 1608.08128. URL: http://arxiv.org/abs/1608.08128 (cit. on p. 27).

[81] Melvyn A. Goodale and A.David Milner. «Separate visual pathways for perception and action». In: *Trends in Neurosciences* 15.1 (1992), pp. 20–25. ISSN: 0166-2236. DOI: https://doi.org/10.1016/0166-2236(92)90344-8. URL: https://www.sciencedirect.com/science/article/pii/0166223692903448 (cit. on p. 27).

[82] Karen Simonyan and Andrew Zisserman. «Two-Stream Convolutional Networks for Action Recognition in Videos». In: *CoRR* abs/1406.2199 (2014). arXiv: 1406.2199. URL: http://arxiv.org/abs/1406.2199 (cit. on pp. 27, 44, 48).

[83] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. «Temporal Segment Networks: Towards Good Practices for Deep Action Recognition». In: *CoRR* abs/1608.00859 (2016). arXiv: 1608.00859. URL: http://arxiv.org/abs/1608.00859 (cit. on p. 27).

[84] Rohit Girdhar, Deva Ramanan, Abhinav Gupta, Josef Sivic, and Bryan C. Russell. «ActionVLAD: Learning spatio-temporal aggregation for action classification». In: *CoRR* abs/1704.02895 (2017). arXiv: 1704.02895. URL: http://arxiv.org/abs/1704.02895 (cit. on p. 27).

[85] João Carreira and Andrew Zisserman. «Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset». In: *CoRR* abs/1705.07750 (2017). arXiv: 1705.07750. URL: http://arxiv.org/abs/1705.07750 (cit. on pp. 27, 44, 48, 56, 57, 74).

[86] Waqas Sultani, Chen Chen, and Mubarak Shah. «Real-world Anomaly Detection in Surveillance Videos». In: *CoRR* abs/1801.04264 (2018). arXiv: 1801.04264. URL: http://arxiv.org/abs/1801.04264 (cit. on pp. 28, 30, 34, 50).

[87] Ming Cheng, Kunjing Cai, and Ming Li. «RWF-2000: An Open Large Scale Video Database for Violence Detection». In: *CoRR* abs/1911.05913 (2019). arXiv: 1911.05913. URL: http://arxiv.org/abs/1911.05913 (cit. on pp. 28, 30, 35, 46, 48, 49, 60, 65, 74).

[88] Vijay Mahadevan, Wei-Xin LI, Viral Bhalodia, and Nuno Vasconcelos. «Anomaly Detection in Crowded Scenes». In: June 2010, pp. 1975–1981. DOI: 10.1109/CVPR.2010.5539872 (cit. on pp. 28, 29).

[89] Louis Kratz and Ko Nishino. «Anomaly detection in extremely crowded scenes using spatio-temporal motion pattern models». In: *2009 IEEE Conference on Computer Vision and Pattern Recognition* (2009). DOI: 10.1109/cvpr.2009.5206771 (cit. on p. 29).

[90] H. Li, A. Achim, and D. Bull. «Unsupervised video anomaly detection using feature clustering». In: *IET Signal Processing* 6.5 (2012), p. 521. DOI: 10.1049/iet-spr.2011.0074 (cit. on p. 30).

[91] M.T. Chan, Anthony Hoogs, J. Schmiederer, and M. Petersen. «Detecting rare events using semantic primitives with HMM». In: Sept. 2004, 150–154 Vol.4. ISBN: 0-7695-2128-2. DOI: 10.1109/ICPR.2004.1333726 (cit. on p. 30).

[92] Waqas Sultani and Jin Young Choi. «Abnormal Traffic Detection Using Intelligent Driver Model». In: *2010 20th International Conference on Pattern Recognition*. 2010, pp. 324–327. DOI: 10.1109/ICPR.2010.88 (cit. on p. 30).

[93] Enrique Bermejo Nievas, Oscar Deniz Suarez, Gloria Bueno Garcia, and Rahul Sukthankar. «Hockey Fight Detection Dataset». In: *Computer Analysis of Images and Patterns*. Springer. 2011, pp. 332–339. URL: http://visilab.etsii.uclm.es/personas/oscar/FightDetection/ (cit. on p. 30).

[94] Kiwon Yun, Jean Honorio, Debaleena Chattopadhyay, Tamara L. Berg, and Dimitris Samaras. «Two-person interaction detection using body-pose features and multiple instance learning». In: *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. 2012, pp. 28–35. DOI: 10.1109/CVPRW.2012.6239234 (cit. on p. 31).

[95] S. Blunsden and Robert Fisher. «The BEHAVE video dataset: ground truthed video for multi-person». In: *Ann. BMVA* 4 (Apr. 2009) (cit. on p. 31).

[96] Sergio A Velastin and Diego A Gómez-Lira. «People Detection and Pose Classification Inside a Moving Train Using Computer Vision». In: *International Visual Informatics Conference*. Springer. 2017, pp. 319–330 (cit. on p. 31).

[97] Tal Hassner, Yossi Itcher, and Orit Kliper-Gross. «Violent flows: Real-time detection of violent crowd behavior». In: *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. 2012, pp. 1–6. DOI: 10.1109/CVPRW.2012.6239348 (cit. on p. 31).

[98] Mauricio Perez, Alex C. Kot, and Anderson Rocha. «Detection of Real-world Fights in Surveillance Videos». In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019, pp. 2662–2666. DOI: 10.1109/ICASSP.2019.8683676 (cit. on p. 31).

[99] Raghak Radhakrishnan, Dheeraj Sharma, and Varshita Murthy. «A Review On Particle Image Velocimetry And Optical Flow Methods In Riverine Environment.» In: Feb. 2017 (cit. on p. 39).

[100] Tianfan Xue, Hossein Mobahi, Frédo Durand, and William T. Freeman. «The aperture problem for refractive motion». In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 3386–3394. DOI: 10.1109/CVPR.2015.7298960 (cit. on p. 39).

[101] Philipp Fischer, Alexey Dosovitskiy, Eddy Ilg, Philip Häusser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. «FlowNet: Learning Optical Flow with Convolutional Networks». In: *CoRR* abs/1504.06852 (2015). arXiv: 1504.06852. URL: http://arxiv.org/abs/1504.06852 (cit. on pp. 40, 41).

[102] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. «A Lightweight Optical Flow CNN - Revisiting Data Fidelity and Regularization». In: *CoRR* abs/1903.07414 (2019). arXiv: 1903.07414. URL: http://arxiv.org/abs/1903.07414 (cit. on pp. 40, 61, 63).

[103] Rico Jonschkowski, Austin Stone, Jonathan T. Barron, Ariel Gordon, Kurt Konolige, and Anelia Angelova. «What Matters in Unsupervised Optical Flow». In: *CoRR* abs/2006.04902 (2020). arXiv: 2006.04902. URL: https://arxiv.org/abs/2006.04902 (cit. on p. 40).

[104] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. «Vision meets Robotics: The KITTI Dataset». In: *International Journal of Robotics Research (IJRR)* (2013) (cit. on p. 42).

[105] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. «A naturalistic open source movie for optical flow evaluation». In: *European Conf. on Computer Vision (ECCV)*. Ed. by A. Fitzgibbon et al. (Eds.) Part IV, LNCS 7577. Springer-Verlag, Oct. 2012, pp. 611–625 (cit. on p. 42).

[106] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. v.d. Smagt, D. Cremers, and T. Brox. «FlowNet: Learning Optical Flow with Convolutional Networks». In: *IEEE International Conference on Computer Vision (ICCV)*. 2015. URL: http://lmb.informatik.uni-freiburg.de/Publications/2015/DFIB15 (cit. on p. 42).

[107] Binu M. Nair. *Deep dive into convolutional 3D features for action and activity recognition (C3D)*. July 2018. URL: https://medium.com/@nair.binum/quick-overview-of-convolutional-3d-features-for-action-and-activity-recognition-c3d-138f96d58d8f (cit. on p. 43).

[108] Zhaofan Qiu, Ting Yao, and Tao Mei. «Learning Spatio-Temporal Representation with Pseudo-3D Residual Networks». In: *CoRR* abs/1711.10305 (2017). arXiv: 1711.10305. URL: http://arxiv.org/abs/1711.10305 (cit. on pp. 44, 47, 57).

[109] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. «Deep Residual Learning for Image Recognition». In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385 (cit. on pp. 44, 47).

[110] Maximilian Ilse, Jakub M. Tomczak, and Max Welling. «Attention-based Deep Multiple Instance Learning». In: *CoRR* abs/1802.04712 (2018). arXiv: 1802.04712. URL: http://arxiv.org/abs/1802.04712 (cit. on p. 50).

[111] Author Seb. *Understanding hinge loss and the SVM cost function*. June 2022. URL: https://programmathically.com/understanding-hinge-loss-and-the-svm-cost-function/ (cit. on p. 50).

[112] Yu Tian, Guansong Pang, Yuanhong Chen, Rajvinder Singh, Johan W. Verjans, and Gustavo Carneiro. «Weakly-supervised Video Anomaly Detection with Contrastive Learning of Long and Short-range Temporal Features». In: *CoRR* abs/2101.10030 (2021). arXiv: 2101.10030. URL: https://arxiv.org/abs/2101.10030 (cit. on pp. 51, 54).

[113] Keras Team. *Keras Documentation: Keras API reference*. URL: https://keras.io/api/ (cit. on p. 56).

[114] Google Brain Team. *Module: TF tensorflow V2.12.0*. URL: https://www.tensorflow.org/api_docs/python/tf (cit. on p. 56).

[115] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. «Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification». In: *CoRR* abs/1502.01852 (2015). arXiv: 1502.01852. URL: http://arxiv.org/abs/1502.01852 (cit. on p. 58).

[116] Mchengny. *Mchengny/RWF2000-video-database-for-violence-detection*. URL: https://github.com/mchengny/RWF2000-Video-Database-for-Violence-Detection (cit. on p. 60).

[117] Twhui. *Liteflownet2: A Lightweight Optical flow CNN*. URL: https://github.com/twhui/LiteFlowNet2 (cit. on pp. 61, 63).

[118] Nikolaus Mayer, Eddy Ilg, Philip Häusser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. «A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation». In: *CoRR* abs/1512.02134 (2015). arXiv: 1512.02134. URL: http://arxiv.org/abs/1512.02134 (cit. on p. 61).

[119] Jason Brownlee. *Data leakage in machine learning*. Aug. 2020. URL: https://machinelearningmastery.com/data-leakage-machine-learning/ (cit. on p. 63).

[120] Rogerhcheng. *LiteFlowNet2 implementation with tensorflow 2*. URL: https://github.com/rogerhcheng/LiteFlowNet2-TF2 (cit. on p. 63).

[121] *Salt-and-pepper noise*. Apr. 2022. URL: https://en.wikipedia.org/wiki/Salt-and-pepper_noise (cit. on p. 70).

[122] *Papers with code - RWF-2000 benchmark (activity recognition)*. URL: https://paperswithcode.com/sota/activity-recognition-on-rwf-2000 (cit. on p. 74).

[123] Hamid Mohammadi and Ehsan Nazerfard. «SSHA: Video Violence Recognition and Localization Using a Semi-Supervised Hard Attention Model». In: *CoRR* abs/2202.02212 (2022). arXiv: 2202.02212. URL: https://arxiv.org/abs/2202.02212 (cit. on p. 74).

[124] Zahidul Islam, Mohammad Rukonuzzaman, Raiyan Ahmed, Md. Hasanul Kabir, and Moshiur Farazi. «Efficient Two-Stream Network for Violence Detection Using Separable Convolutional LSTM». In: *CoRR* abs/2102.10590 (2021). arXiv: 2102.10590. URL: https://arxiv.org/abs/2102.10590 (cit. on p. 74).

[125] Yukun Su, Guosheng Lin, Jinhui Zhu, and Qingyao Wu. «Human Interaction Learning on 3D Skeleton Point Clouds for Video Violence Recognition». In: *Computer Vision – ECCV 2020*. Ed. by Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm. Cham: Springer International Publishing, 2020, pp. 74–90. ISBN: 978-3-030-58548-8 (cit. on p. 74).

[126] Kate Crawford. *Power, Politics, and the Planetary Costs of Artificial Intelligence*. New Haven: Yale University Press, 2021. ISBN: 9780300252392. DOI: doi:10.12987/9780300252392. URL: https://doi.org/10.12987/9780300252392 (cit. on p. 79).