Università degli studi di Padova

Department of Information Engineering

Master Thesis in

ICT for Internet and Multimedia

Gait analysis from encrypted video surveillance traffic

Supervisor
Simone Milani

Co-supervisors
Sebastiano Verde
Umberto Michieli

Master Candidate
Sara Bordin

16 December

Academic Year 2018/2019

Non c'è piacere nel successo,
se non lo dividi con qualcuno.
I fantastici 4

Abstract

In recent times, gait analysis has emerged as a new area of forensic practice. Differing from other biometric identification technologies, such as face recognition, gait recognition is widely known as one of the most important non-contactable, non-invasive biometric identification technologies since it is very hard to counterfeit.

Thanks to these advantages, gait recognition is expected to be applied in scenarios like criminal investigation and access control. Although there are some critical issues regarding the identification accuracy, the introduction of new enabling technologies has significantly advanced the study of gait since it could be the only available tool when other biometric features are concealed.

In the academic literature, several approaches have been proposed for analysing gait in a forensic context, but research is still ongoing to increase accuracy, validity, reliability and robustness of these methods.

This thesis proposes an original video-based gait analysis technique, different from others already existing in the literature. In particular, we leverage deep learning techniques to analyze packet sizes from streamed video sequences both in a virtual and real environment.

As further examinations, we address the case in which encryption mechanisms are adopted and we conclude the study proposing an incremental learning framework to render the system more suitable to real life applications where training data becomes progressively available over time.

Sommario

Negli ultimi tempi, l'analisi della camminata è emersa come una nuova area della pratica forense. Diversamente da altre tecnologie di identificazione biometrica, come il riconoscimento facciale, l'analisi dell'andatura è ampiamente conosciuta come una delle più importanti tecniche di identificazione biometrica non invasive e che non richiedono contatto poichè è molto difficile contraffarla.

Grazie a questi vantaggi, si dovrebbe applicare il riconoscimento della camminata in scenari quali investigazioni criminali e controllo degli accessi. Sebbene esistano alcuni problemi critici riguardanti l'accuratezza dell'identificazione, l'introduzione di nuove tecnologie abilitanti ha significativamente avanzato lo studio dell'andatura dal momento che questa potrebbe essere l'unico strumento disponibile quando le altre le caratteristiche biometriche vengono nascoste.

Nella letteratura accademica, diversi approcci sono stati proposti per analizzare la camminata in un contesto forense, ma la ricerca è ancora in corso per aumentare l'accuratezza, la validità, l'affidabilità e la robustezza di questi metodi.

Questa tesi propone una tecnica originale basata su video per analizzare l'andatura, diversa da altre già presenti nella letteratura.

In particolare, sfruttiamo tecniche basate sul deep learning per analizzare le dimensioni dei pacchetti di sequenze video sia in un ambiente virtuale che reale.

Inoltre, affrontiamo il caso in cui vengono adottati meccanismi di criptazione e concludiamo lo studio proponendo un framework di apprendimento incrementale per rendere il sistema più adatto ad applicazioni di vita reale dove i dati di training vengono resi disponibili progressivamente nel tempo.

Contents

Listing of figures

Listing of tables

High'st queen of state, Great Juno, comes; I know her by her gait.

The Tempest (Act 4, Scene 1)

1 Introduction

People have been reasoning about how they walk since the earliest times. Aristotle (382-322 BCE) can be credited as the pioneer of gait analysis. He left the first known written observations which are reported in the treaty *On the gait of Animals* (*De Motu Animalium*):

> *If a man were to walk on the ground alongside a wall with a reed dipped in ink attached to his head the line traced by the reed would not be straight but zig-zag, because it goes lower when he bends and higher when he stands upright and raises himself.*

Aristotle, in his works, discussed the difference between human and animal gait; moreover, he was interested in classifying organisms according to their motion. Unfortunately, none of his propositions were ever tested by experiments: he lived in a society where it was assumed that reasoning about a problem was sufficient to solve it. At a later time, whilst this particular observation is true, many of his conclusions proved to be wrong [11].

Only durint the Renaissance in Europe, science and mathematics started to develop coherently. It was at this time that the basis of gait analysis were established.

The biggest debate in the 19th century concerned the horse's gaits, rather than human's ones, since it was the primary means of transportation.

Particularly, the main interest was aimed at finding an answer whether during the trot all four hooves are off the ground at some instants.

Some centuries later, Leland Standford, american senator from California, chose to settle the question with the help of images. To this purpose, he engaged a photographer, Eadweard

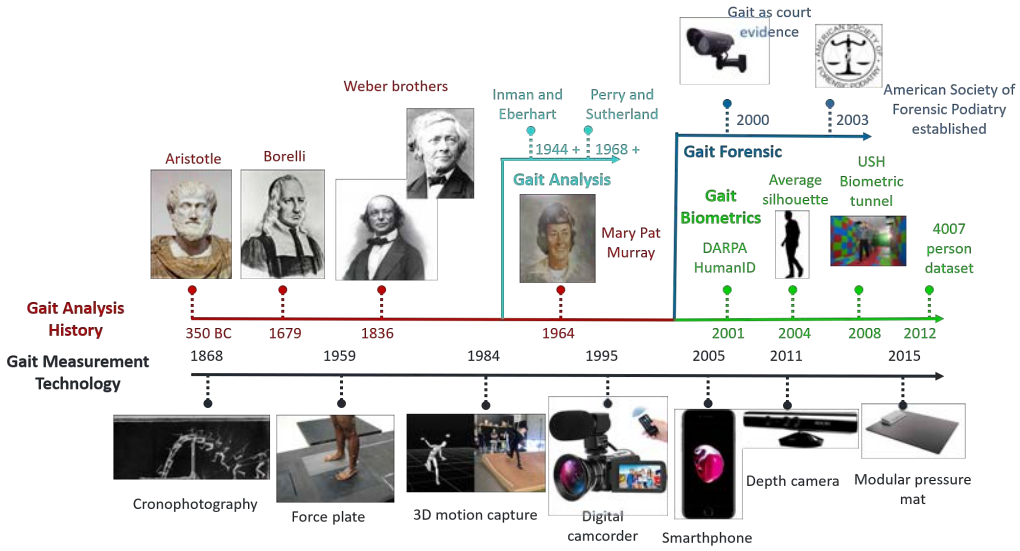


Figure 1.1: Gait recognition timeline. On the upper side: gait recognition and its advancements over time. Three main areas are highlighted: gait analysis, gait forensics and gait biometrics. On the bottom side: a list of enabling technologies.

Muybridge (1830–1904), and succeeded in proving his theory: there are some instants in which all four feet are, indeed, simultaneously off the ground.

The idea of getting data about gait from a series of images that can capture motion details otherwise imperceptible to the naked eye, marked the beginning of a new technique in the gait analysis area. The development of photography made all this possible.

The studies carried out nowadays are the consequence of collaborations of individuals with different fields of expertise.

In [11] it is possible to find a list of major contributors to this field while, Figure 1.1, reports the timeline history of gait analysis and the main enabling technologies that have driven the development of this field [12].

In the following, we will present the problem of gait acquisition and analysis from a biometric perspective.

Biometric systems follow the following processing steps:

- capture biometric samples;
- extract relevant features from acquired information;
- compare the extracted features with the available dataset.

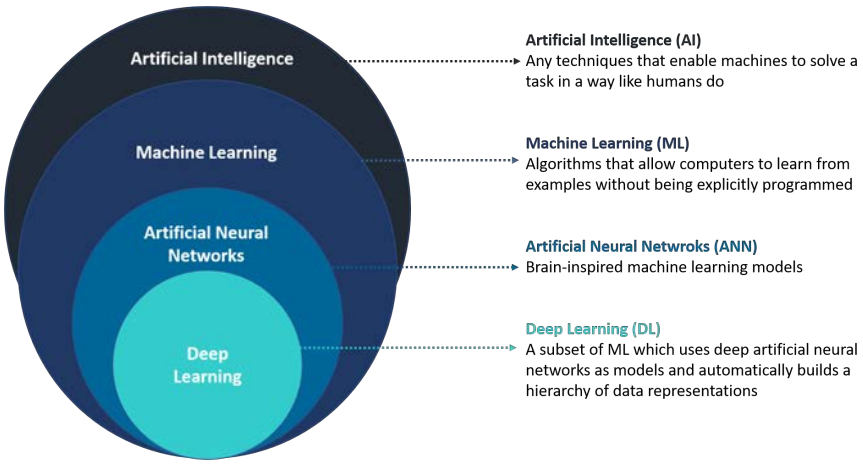


Figure 1.2: The relations between Artificial Intelligence (AI), Machine Learning (ML), Artifcial Neural Networks (ANN) and Deep Learning (DL).

According to how data are gathered, biometric gait recognition methods can be categorized into three main groups:

- *computer vision based:* gait patterns are acquired through video cameras. Image and video processing techniques are adopted to extract features for classification purposes;
- *floor sensor based:* floors are embedded with sensors that enable to measure gait related features;
- *weareable sensor based:* gaits are collected using wearable devices, such as smart watches and wristbands, which can be worn on different points of the human body [13].

One of the main advantages of the last two approaches is their privacy-preserving data collection, while, the first, allows gathering rich information without any close interaction with the monitored subject.

Application areas for computer based gait recognition are usually surveillance and forensics; floor sensing is exploited to control access on buildings (sensors are usually placed in front of doors) while, wearable sensor based strategies are often employed for security and user authentication in mobile and portable electronic devices.

The work carried out in this thesis falls within the first category in an original way. The majority of the investigations are based on hand crafted features and images preprocessing techniques: images are processed to retrieve pose estimations or silhouettes. The novelty of

this thesis resides in the fact that, instead processing images, we analyzed packet size series from compressed video sequences. Moreover, automatic feature extraction is implemented through convolutional neural networks that are part of deep learning techniques. As shown in Figure 1.2 , deep learning is a subset of machine learning but it differs from standard algorithms thanks to its impressive performances [14].

Furthermore, the system developed in this work is desired to be appropriate for real life applications where training data becomes gradually available over time or their size is out of system memory capability. The tool is, therefore, required to dynamically extend its expertise adapting to new data without forgetting its existing knowledge as human beings do. To implement this ever increasing learning, an incremental learning technique is proposed.

In the end, it is shown that the designed procedure is effective also when encryption is enacted. Samuele Piazzetta, a collaborator of this project, investigated the protocols involved in secure video surveillance transmissions and designed a tool that simulates real time streaming.

The tool used in this thesis are: Unity 2019.1.1.11f1, FFmpeg and FFprobe libraries, Python 3.7, Anaconda2 2019.3 and Jupyter notebook, Matlab 2019a, Keras 2.2.4 with Tensorflow-gpu 1.14.0 backend. The exploited hardware resources are: a very performing computer (Intel Core i7-8700K CPU and 32 GB RAM) equipped with NVIDIA GeForce GTX 1070 GPU to implement and train the classifiers, and a Microsoft®LifeCam HD-3000 webcam, to acquire real data.

The rest of the thesis is organized as follows:

- in Chapter 2, an overview concerning gait analysis background and information leakage from encrypted systems is given;
- in Chapter 3, Unity game engine is presented;
- in Chapter 4, video coding concepts exploited in this thesis are described;
- in Chapter 5, secure protocols behind video surveillance systems are analyzed;
- in Chapter 6, some machine learning strategies are introduced, especially focusing on convolutional neural networks and incremental learning;
- in Chapter 7, the classifier designed within the thesis research work is described;
- in Chapter 8, the main results are discussed;
- in the last Chapter, the conclusions are drawn.

Imagine a bank robber who has covered his face but can be identified by the way he walks out of the bank.

2 State-of-the-Art

This Chapter is intended to present the background concerning gait analysis investigations. Additionally, works related to side channel attacks are examinated.

At the end of the Chapter, it is reported the work from which the eigenwalks strategy is inspired. The study addresses a forensics investigation through an eigenalgorithms approach.

2.1 Gait analysis background

2.1.1 Biometric systems

In the modern society where we live, a reliable identification of individuals becomes a fundamental necessity in many real-time applications (such as forensics, international border crossing, financial transactions, and computer security).

Human body characteristics (such as face, iris, voice, and gait) play a vital role in characterizing person. Such biological characteristics that allow to uniquely identify a human being are termed as biometric features.

Biometric traits are mainly classified into two categories, as shown in Figure 2.1:

- *physiological*: such as iris, face, ear, fingerprint, DNA or vein patterns;
- *behavioral*: such as voice, gait or signature.

Any human physiological and behavioral characteristics can be used as a biometric parameter for recognition if it satisfies the following properties:

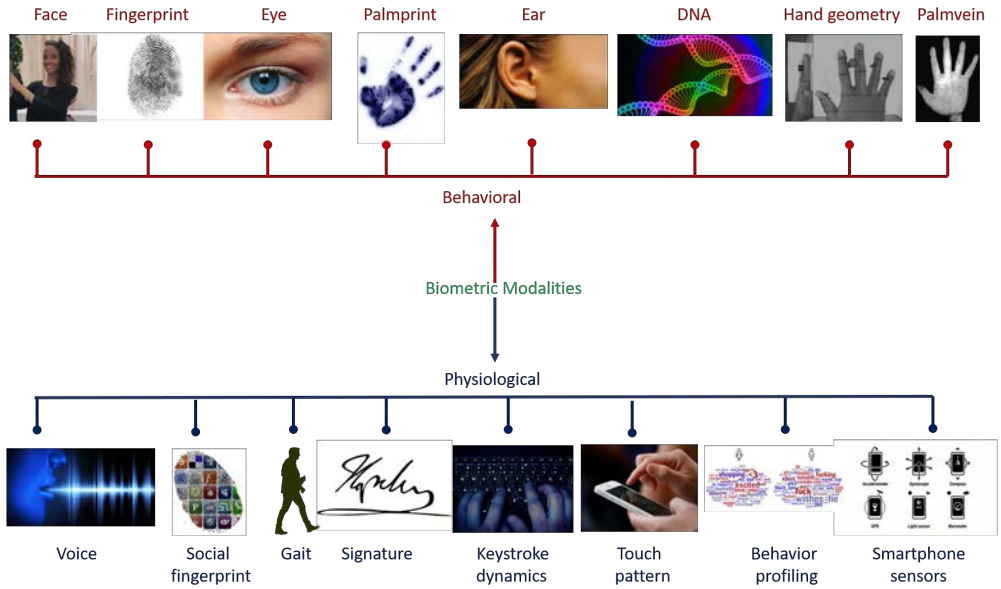


Figure 2.1: Major types of biometric modalities.

- *universatility*;
- *distinctiveness*;
- *stability*;
- *measurability*;
- *performance*;
- *utility*;
- *acceptability*;
- *resistance to attacks* [15].

Biometric gait recognition means recognizing people from the way they walk. Specifically, gait analysis is *the systematic study of human walking, using the eye and brain of experienced observers, augmented by instrumentation for measuring body movements, body mechanics and the activity of the muscles* [16].

This topic is relatively recent, compared to the traditional approaches such as fingerprint recognition, but it has drawn more attention to researchers since:

- each of us have a sufficiently distinctive uniqueness in walking style;

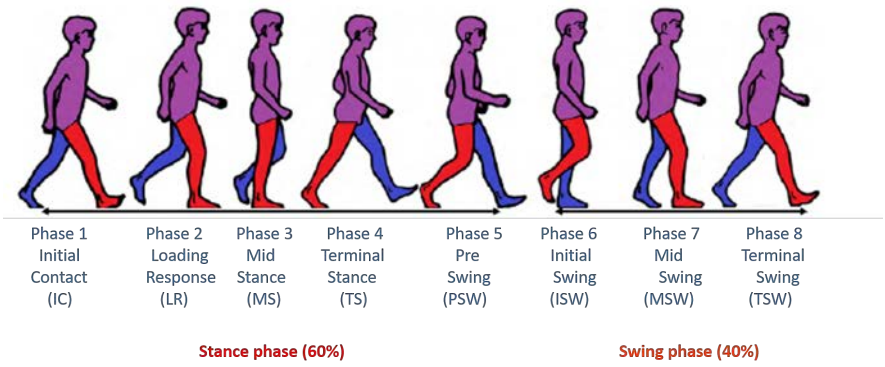


Figure 2.2: Gait cycle phases with the right leg (red color) considered as a reference leg [1].

- gait features can be easily extracted without cooperation and from a distance of more than 10m;
- gait characteristics can even be analyzed from low-resolution video sequences;
- gait characteristics are less prone to circumvention as compared to signature, face and voice [15].

Researches associated with gait analysis aim at identifying human walking patterns and found their analysis on human gait cycle, which will be briefly discussed in the following subsection.

2.1.2 Human gait cycle

A gait cycle is the time period (or sequence of events) included between the instant when foot contacts the ground and the instant when the same foot contacts the ground again. A single gait cycle is also known as a *stride* and, when analyzing a gait cycle, one foot is taken as reference: its motion is going to define the gait pattern.

Each stride has two phases:

- *Stance Phase:* in this period the foot remains in contact with the ground. It is, in turn, subdivided into 5 periods and it covers the 60% of the overall gait cycle;
- *Swing Phase:* in this period the foot is not in contact with the ground. It has 3 periods and it constitutes the 40% of the stride.

Figure 2.2 shows the gait cycle phases with the right leg (red color) considered as a reference leg.

2.1.3 Challenges

Despite the advantages of gait traits compared to other biometric features, many factors in gait-based systems pose several challenges to gait-based systems affecting, consequently, their accuracy:

- *Internal factors*: these cause changes to the natural walk due to sickness, aging, pregnancy, drunkenness or other physiological body's changes;
- *External factors*: they do not depend on intrinsic properties of the gait but rather on the environment where an individual walks or his/her clothing. Some examples can be: lighting conditions, viewing angles, outdoor/indoor environment, clothes, shoe types, object carrying and so on;
- *Occlusion:* it is one of the crucial challenges that frequently occurs in gait recognition, especially in real-world surveillance and control access. Occlusion can occur because of multiple factors. For instance, whenever a person walks amid a group of other people it is very difficult to acquire using a computer vision method its walking pattern.

Many works in the literature address this issues. To tackle more robustly this issues, multimodal biometric systems can be implemented: in these systems, gait is combined with other biometrics. As an example, in [17] gait is combined with face traits: the frontal face was captured by one camera and the side-view of the person was captured by another camera.

2.1.4 Applications

Forensic gait examinations have contributed to several criminal trials in Europe in the past 15 years (the first known case occurred perhaps in the 2000), but the admission of gait evidence differs between courts [12].

In a bank robbery case occurred in Denmark in 2004, a court found video-based gait analysis to be a valuable tool [18]. The Institute of Forensic Medicine in Copenhagen (IFMC) was asked to confirm the perpetrator via gait examinations and the suspect was convicted of robbery. Video-based gait investigations are particularly useful in robbery cases where the perpetrator used to wears mask and gloves making therefore impossible to capture his face or fingerprints.

Gait recognition is useful also for identification purposes where it is of the utmost importance to adopt a suitable user authentication strategy in order to prevent unauthorized access. User authentication is the process of verifying claimed identity and it can be grouped into three classes [13]:

- *Knowledge based authentication (KBA):* it relies on something that one secretly knows such as password or PIN codes;
- *Object (or token) based:* it is based on the possession of an object or device, and it is usually combined with the knowledge based approach. A common example is the combination of a bank card with its PIN code;
- *Biometric based:* it finds its foundation on some physical and behavioral traits of individuals..

The first two classes have some inconveniences since passwords and tokens can be forgotten, lost or stolen. They also pose several usability limitations since managing multiple or strong passwords is not an easy task. Biometric based person identification overcomes this issues since they use peculiar measurable traits of human beings.
As a matter of fact, nowadays the demand for biometrics-based systems is increasing.

Furthermore, biometric systems operate in two modes:

- *verification*: the system performs a one to one comparison in order to establish whether to accept or not the claimed identity. The verification mode tries to answer to the question *"Is he/she who he/she claims he/she is?"*;
- *identification*: the system performs a one to many comparison in order to determine the user's identity. The identification searches an answer to the question *"Who is he/she?"*.

Besides the forensic and security worlds, gait analysis is widely used for medical purposes: gait investigations provide valuable information in medical diagnosis and rehabilitation. For example, assessment of gait abnormalities allows to detect Parkinson's disease [19].

Moreover, gait studies can be used to perform gender recognition and age estimation.

2.2 Side channel attacks

Recent years have seen the ubiquitous presence of smart interconetted devices touching almost every corner of the world: we are living in the so called Internet of Things (IoT) era.

Together with convenience and efficiency, privacy and security issues can be associated with those fast developing technologies [20]. Cryptography tries to overcome this problems making use of mathematical tools and offering services such as data confidentiality, integrity

and authenticity. Nevertheless, the discovery of side channel attacks (SCA) in late 1990s [21] alerts the world that several physical information source can also become a secret teller that impairs security and confidentiality in many practical scenarios.
Basically, SCA are not interested on decrypting the message itself but they rather exploit side effects to gather sensitive information. Example of extra source of information are: timing information, power consumption and electromagnetic leaks. Many works in literature are interested on detecting such sources of information.

In this section, works related to information inference from packet traffic analysis are addressed: in the first subsection, investigations related to general fields are reviewed, while, in the second one, the ones concerning video surveillance systems are illustrated.

2.2.1 Information inference from secure channels

An interesting investigation is reported in [22] where it is proved the feasibility of an accurate identification of Netflix videos through passive traffic analysis. Netflix first encodes their videos as variable bitrate (VBR) MPEG-4 streams and transmit them using Dynamic Adaptive Streaming over HTTP (DASH) via Microsoft Silverlight [23]. This combination of DASH and VBR produces sequences of video segment sizes that are unique for each video (i.e., they define a "fingerprint" for each video sequence). The authors exploit this fact to create a dataset consisting on 42,027 fingerprints. Specifically, they leverage the findings asserted in [24], where it is demonstrated that fingerprints for a video can be created by capturing the metadata transmitted during the first few seconds of a stream without necessarily watching the entire video. The approach adopted uses only the information provided by TCP/IP headers. The authors propose a fully automated fingerprints creation process to assess that a robust attack can be created with a very small effort. Moreover, differently from the implementation carried out in this project, they do not exploit machine learning techniques.

The outcomes show that 199 of 200 video streams are identified (corresponding to 99.5%) with the earliest identification occurring at 2:00 and the latest at 12:04.

Additionally, they prove that HTTPS adds a negligible amount of overhead to each video segment. Therefore, traffic analysis attack also works against HTTPS-protected Netflix videos. This study proves that Netflix's recent upgrade to HTTPS is not effective in protecting the privacy of their users against a passive traffic analysis attack. In the same work, the authors propose some possible solutions to mitigate this issue.

In [25] it is proposed a framework to infer user activities performed while using mobile

phones by eavesdropping encrypted network traffic. The authors' aim is to highlight the gap between theory an practice: albeit Secure Sockets Layer (SSL) and its successor Transport Layer Security (TLS) are used to protect the content of a packet, they do not prevent the detection of networks packets patterns that may reveal some sensitive information about the user behavior. In particular, the authors run a set of experiments considering three popular apps: Gmail v4.7.2 , Facebook v3.8, and Twitter v4.1.10. The investigation is carried out assuming the traffic is encrypted and the adversary eavesdrops the messages exchanged between the user's device and the web services that he uses without modifying them. At first, packet sequences are aligned using Dynamic Time Warping; then, a hierarchical clustering is applied, and in the end, a Random Forest classifier identifies the adopted app and the users' actions. Also in this work, Wireshark tool is used to eavesdrop network packets. The authors proved that a high accuracy on the identification of the adopted application and users' actions for all the three apps. As for Facebook, the average *F-measure* is equal to 99%; for Gmail, the *open chats* action is the most challenging one while, the authors observe that Twitter actions prove to be more difficult to classify than those of Gmail and Facebook. Despite this, good results can be obtained also for this app with an average *F-measure* equal to 97%.

The study is concluded mentioning possible countermeasures to this kind of attacks and pointing out the respective weaknesses. Therefore, authors urge, future researchers to work on these issues.

2.2.2 Side channel attacks in video surveillance systems

A first work concerning side-channel leakage for IP video surveillance traffic is presented in [26]. Through analysis of network traffic metadata such as packet size, inter-arrival time and overall stream bandwidth, the authors prove that it is possible to detect motion (such as person standing up or walking past a camera) and scene changes even when SSL or AES encryption techniques are embraced. The study is carried out testing multiple codecs (such as MJPEG, H.264 and VP8) and cameras (including Skype and Hangouts).

A second solution analyzing an encrypted video surveillance stream can be found in [27], where it is demonstrated that users' basic activities of daily living can be recongized with high accuracy. The monitored activities are: dressing, styling hair, moving and eating and the method adopted allows to infer activities based only on the size of the encrypted traffic of a video stream. Moreover, the authors claim that, whilst difference coding allows for efficient video compression, it is also the cause of side channel information leakage. In par-

ticular, they note that the traffic size of an encrypted video stream with no motion is much smaller than one where some people of object moves. As a consequence, traffic patterns differ significantly when a user performs different activities within the field of view of the camera. In this paper, manual feature extraction are implemented and both supervised and unsupervised learning techniques are applied adopting k-NN classification and DBSCAN-based clustering, respectively. The hardware used includes two commercial cameras with AES encryption. The experiments are conducted by 4 volunteers (2 females and 2 males), with each performing the 4 activities in front of the cameras for 400 times in a typical living room with natural lighting conditions. The authors evaluate impact of: the traingin set size, the distance from the camera and of the illumination. All the results are motivated by difference coding and in accordance with what intuition suggests: the best performance are achieved when activity are performed close to the camera and in the afternoon.

2.2.3 Eigenalgorithms for device identification

The work of this project is inspired by [28] where the authors adopt eigenalgorithms strategy to perform device identification. This multimedia forensic challenge is usually addressed by leveraging footprints left by the the acquiring device in the processing steps. Many works in the literature exploit traces left by the imaging sensor, the approach in [28] modified this usual strategy, and the traces left by lossy coding are used to reveal the acquiring device. In particular, the authors want to identify the motion estimation algorithm used by a video encoder since it represents one of the non-normative tools that can be customized in the design of the encoder. They assume the forenisc analyst does not perfectly know all the available algorithms but a subset of them. Specifically, the *closed and known set* available to the analyst consists on 6 algorithms while the *unknown set* comprises 2 algorithms. From the closed set, the analyst selects 3 eignealgorithms and investigate different values of QP. The authors prove that they achieve nearly the same outcomes reached in [29] where it is assumed that the analyst has full knowledge of the available algorithms. Furtermore, they highlight that their results are of particular interest to identify a device whose software implementation is not known (e.g. because it is proprietary).

In the future, they propose to extend the study to further processing units and to examinate a possible adversary counterattack that might fool this method.

3 Unity

Unity is the starting point for the development of the entire project.

In this Chapter, this game engine is described starting from the basic building blocks till the general structure of a generic application.

3.1 Unity Game Engine

A game engine is a software environment devised to build high quality video games easily and efficiently. Developers use game engines to create games for consoles, mobile devices, and personal computers [30].

The term *game engine* arose in the mid-1990s in reference to first-person shooter (FPS) games like the popular *Doom* and *Quake* by id Software [31], [32]. From the 90s onwards, game engine technology has expericenced an ever increasing evolution.

Typically, a game engine provides a set of core components including: the rendering engine, the collision and physics engine, the animation system, the audio system, networking, scripting, memory management and so on.

Currently, no specific standard architecture has been developed in the literature but some components are essential to all games while others are fundamental only for some of them [33].

Nowadays, more than 100 engines are available in the market [33], for both commercial and educational purposes. These differ for their features and characteristics; as a mattter of

fact, selecting an appropriate game engine for a specific purpose is challenging.

In [33], some existing game engine solutions are discussed and 20 of them are evaluated according to the *MULER* (modularity, usability, library resources, efficiency, rendering effects and picture quality) criterion.

In this thesis, the Unity game engine [34], also known as Unity3D, is used. Unity is famous for its ease of programming and versatility and, for this reason, it is particularly suitable for independent game developers and small teams [35], [36].

The first version of Unity (1.0.0) was launched by the three colleagues David Helgason, Joachim Ante and Nicholas Francis in Denmark on June 6, 2005 [35]. The aim of the creators was to *democratize game development* providing a professional tool for amateur game developers.

At the beginning, Unity was available solely for Mac OS X. The current version (2019.2.0) is supported on both Windows and Mac OS X, with a version available for the Linux platform in an experimental stage [37]. Furthermore, it offers more than 25 different target platforms [38].

In the following sections, the basic Unity concepts are explained focusing on the ones exploited in this work.

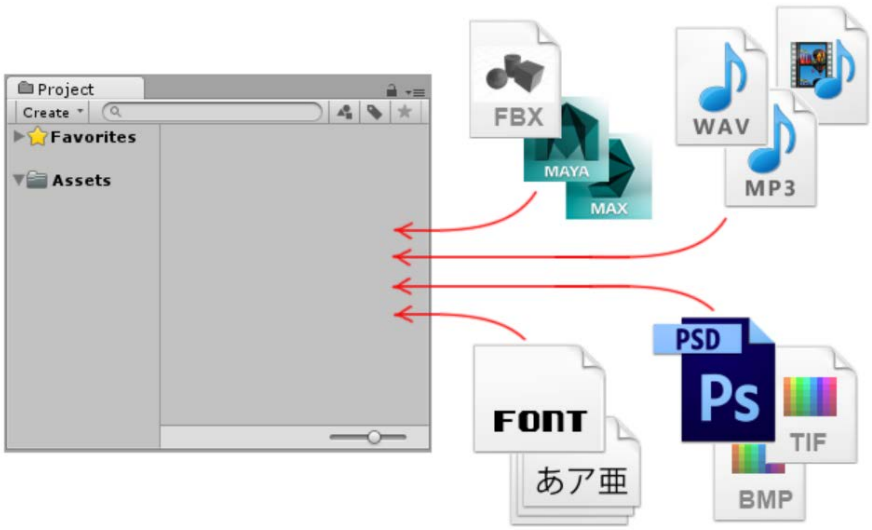
3.2 Unity Components

Before presenting the main components of Unity, some essential terms are recalled that anyone dealing with Unity environment should know.

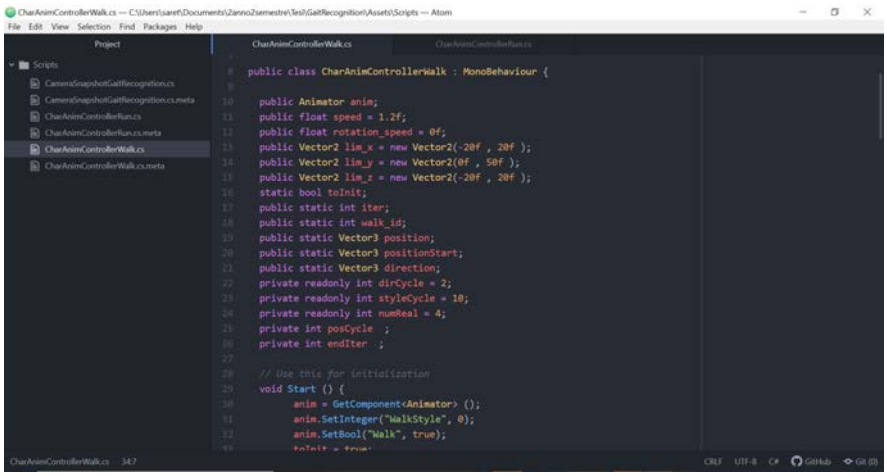
3.2.1 Technical terms

- *Assets*: these are the building blocks of all Unity projects. Unity refers to every file used to create the game as asset: from image files to 3D models and sound files or any file types that Unity supports. This is the reason why, in any project folder, all files used are stored in a sub folder named Assets. Moreover, an Asset may come from a file created both inside or outside Unity [39], [40]. Samples of assets used in this project coming outside Unity are the FBX files [41] imported from Mixamo [42], while, samples of assets created within Unity are the *Animation Controllers*. Both of them will be presented in chapter 7;

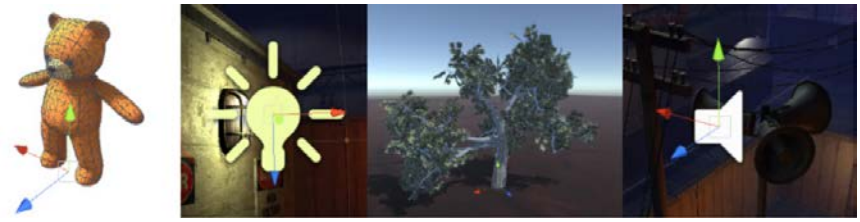
- *Scenes*: each scene file can be thought as a individual level or area of a game content where environments, objects and physical effects are placed. By constructing the game



(a) Asset types.



(b) Script.



(c) Four GameObject types: a character, a light, a tree, an audio source.

Figure 3.1: Unity components [2]

with many scenes, hence designing and building the game in pieces, it is possible to test different parts of the game individually.

A default new scene is usually empty except for a Camera (called *Main Camera*) and a Light (called *Directional Light*). [43], [40];

- *Game Objects*: a Game Object is the most important concept in Unity. An asset within a game scene, is defined as a new Game Object. Every element in the game is a GameObject, from characters to lights or cameras. However, a GameObject can not do anything on its own; its properties need to be dynamically changed by adding *components*. A GameObject can be thought as an empty cooking pot, and components as different ingredients that make up the recipe of the game [44], [40].

- *Components*: these defines the properties of the Game Object. All Game Objects contain at least one initial component: the *Transform component*, which specifies the position, rotation, and scale of the object. Unity has lots of different built-in component types but components can be also created or modified through scripting. All the components of a given object are displayed in the Inspector window [45], [40].

  The Transform Component also enables a concept called *parenting* which is a crucial concept of the Unity environment;

- *Scripts*: these are considered by Unity as components and they are an essential part of game development. Scripting (or creating scripts) is writing in code personal additions to the Unity functionalities. When a script is saved and attached to the GameObject, the script appears in the GameObject Inspector just like a built-in component. Unity natively supports the C# programming language but many others can be used [46], [47], [40].

- *World Space*: the absolute XYZ coordinates of all objects;

- *Local Space*: it defines object positions in relation to one another. By creating parent-child relationships between objects, their positions can be compared in relation to one another;

- *Parenting*: it is among the most important concepts to understand when dealing with Unity. When a GameObject is a parent of another GameObject, the Child GameObject will move, rotate, and scale exactly as its parent does.

  Parenting can be view as the relationship between the arms and the body: whenever the body moves, the arms also move along with it.

  Child objects can also have children of their own and so on. Any object can have multiple children, but only one parent. These multiple levels of parent-child relationships form a *Transform hierarchy*. The object at the very top of a hierarchy is known as

Figure 3.2: Unity interface.

the *root*. A Parent-Child relationship between two GameObjects can be created by dragging any GameObject in the *Hierarchy View* onto another.

Note that the Transform values in the Inspector for any child GameObject are displayed relative to the Parent's Transform values. These values are referred to as local coordinates [48].

- *Prefabs*: these allow to store the object, including its components, property values, and child GameObjects as a reusable Asset. Prefabs can be thought as templates that can be recycled. Prefab system is better than simply copying and pasting the GameObject since it allows to automatically keep all the copies in sync. However, it is possible to override settings on individual prefab or create variants of Prefabs [49], [40].

To recap, the hierarchy for organizing Unity3D game projects: a game is composed of one or more scenes, each scene includes one or more GameObjects and every GameObject is composed of some components or child GameObjects.

3.2.2 Unity Interface

In this section the Unity interface is described.

After that the program is launched and the project is created, the Unity editor appears on the screen. The Unity interface, like many other working environments, has a customizable layout.

As shown in Figure 3.2, a typical Unity layout comprises:

- *Scene View*: this window is where the game is built. It offers a perspective (full 3D) view, which is switchable to orthographic (top down, side on, and front on) views. Dragging an asset to this window will make it an active game object [50]. This window is also accompanied by six useful control buttons that are described in [51].

  The Scene view is tied to the Hierarchy and to the Inspector windows;

- *Hierarchy Window*: this window lists all active objects in the currently open scene by order of creation [52];

- *Inspector Window*: this window can be thought as a personal toolkit to adjust every element of any game object or asset in the project. It shows the main properties of the selected object and allows its modification. Having selected objects in either the Scene or Hierarchy, implies the selection in the other window. In this way it is possible to verify their properties in the Inspector [53], [40];

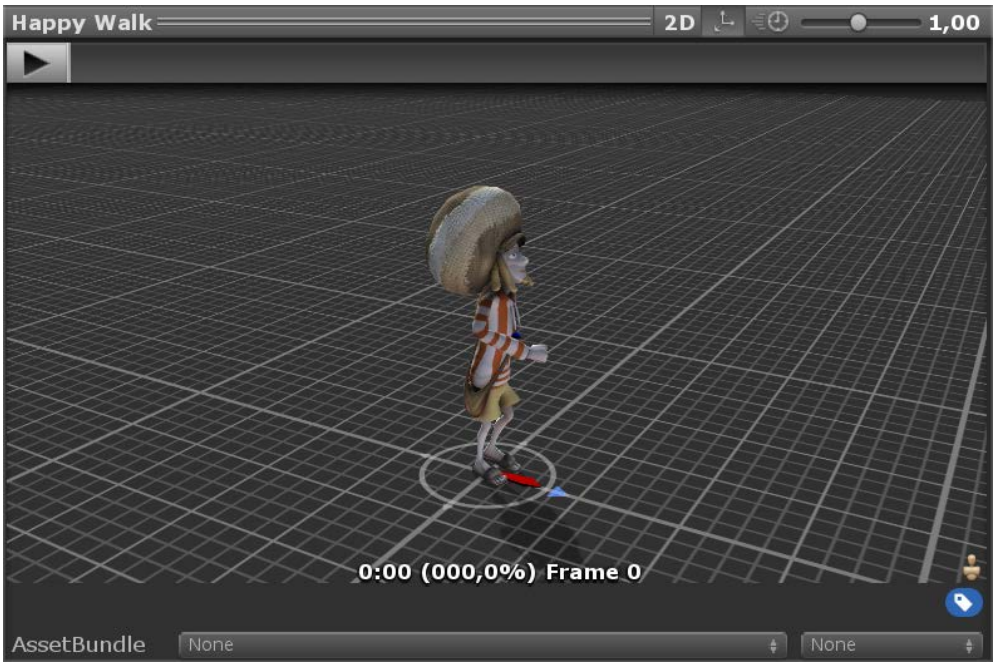
- *Game View*: this window is invoked automatically by pressing the Play button and acts as a realistic preview of the game rendered from the camera of the scene. All changes made during play mode are resetted when the Play mode ends;

- *Asset Store*: it permits downloading free or affordably priced assets. This store is a growing library of assets created and published both by Unity Technologies and also members of the community. The variety of files available covers everything from Textures, Models and animations to whole Project examples, tutorials and Editor extensions. The Asset Store window allows to download and import Assets directly into the Project [54], [55];

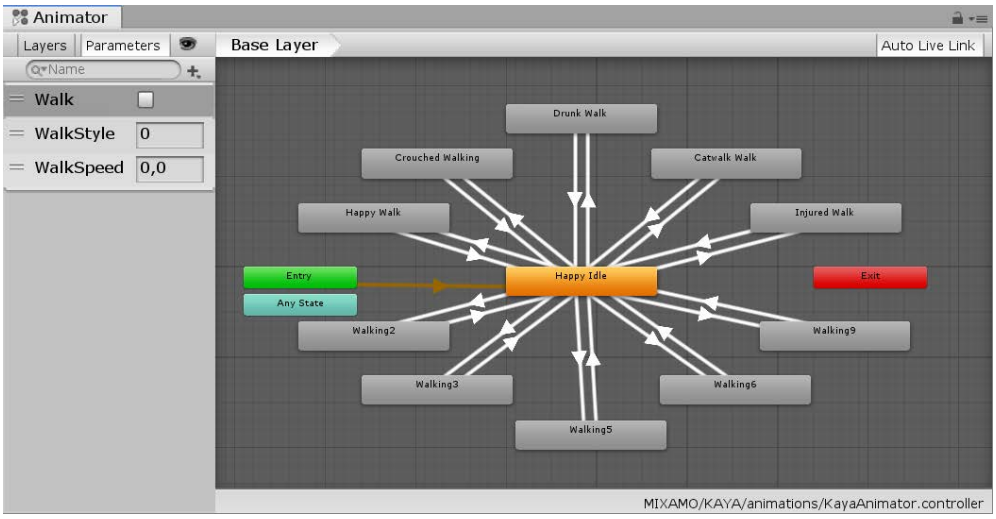
- *Project Window*: it provides a direct view of the Asset folder of the project. From this window it is possible to manage the assets.

  The left panel of the window shows the hierarchical folder structure of the project. In the right hand panel, the individual assets are shown as icons that indicate their type. Above the project structure list is a *Favorites* section where frequently-used items can be kept for easy access. Located at the left side of the toolbar, the *Create* menu allows adding new assets and sub-folders to the current folder. On its right, a set of tools enable searching different items in the project [56],.

- *Console Window*: it shows logs of messages, warnings, and errors. Messages in the Console can be displayed using also the *Debug.Log*, *Debug.LogWarning* and *Debug.LogError* functions [57].



(a) Animation Clip.



(b) Animator Controller.

Figure 3.3: Unity animation.

```
using UnityEngine;
using System.Collections;

public class MainPlayer : MonoBehaviour {

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {

    }
}
```

Figure 3.4: Anatomy of a script file.

3.2.3 Unity Animation

In this subsection it is illustrated two of the core elements of the Unity animation system.

The first one is the *Animation Clip* that exports the information on position, rotation, or the values of other properties over time for different objects.

Anymation Clips can be imported from external sources such as Autodesk 3ds Max ®[58], Autodesk Maya [59] or Unity Asset Store [54]. Otherwise, they can be created or edited from scratch within the editor using the Animation window [60]. A sample of animation clip taken from Mixamo can be seen in Figure 3.3a.

The second one is the *Animation Controller*. The controller has references to the animation clips used within it, and manages the various animation states and the transitions between them using a so-called State Machine, which could be thought of as a kind of flowchart, or a simple program written in a visual programming language within Unity [61],[62]. A sample of Animation Controller created exploiting Mixamo animation clips can be found in Figure 3.3b.

3.2.4 Unity Scripting

In 3.2.1, scripts are presented and their role are defined.

In this subsection it is depicted the anatomy of a script file since it is the one used in this thesis. Unlike most other assets, scripts are usually created within Unity directly. Once created, it will be opened in a text editor by double-clicking on it. A sample of the initial contents

(a) Adam [65].

(b) Baymax Dreams [66].

Figure 3.5: Cinematographic applications.

(a) 3D auto rendered with Unity and Autodesk [67].

(b) Volkswagen VR Training by Innoactive [68].

Figure 3.6: Automotive applications.

of the file is shown in Figure 3.4. *MonoBehaviour* is the base class from which every Unity script derives. When using C#, all classes must explicitly derive from MonoBehaviour [46]. Inside the class, there are two main functions to note. The *Start* function will be called by Unity before gameplay begins (hence, before the Update function is called for the first time) and it is the place to do any initialization. Start is called exactly once in the lifetime of the script [47].

The *Update* function is the place to put code that will handle the frame update for the GameObject including movements, actions, response to user input or whatever needs to be managed over time during gameplay. Not every MonoBehaviour script needs Update [63].

These functions are called *Event Functions* [64]. Several others exist that can be consulted in [46] but they are outside the scope of this work.

3.3 Unity Applications

This chapter ends by mentioning some of the most interesting uses involving Unity game engine.

Besides being widely adopted by industries to build 3D, 2D, virtual reality, and augmented reality games, Unity finds application also in filmaking and simulation purposes.

The first film experimented with Unity is *Adam*, a short film about a robot escaping from prison, built to showcase and test out the graphical quality achievable with Unity in 2016 [65].

Also Disney Television Animation launched three shorts, called *Baymax Dreams*, created with Unity in 2018 [66].

Additionally, automakers use Unity technology to build full-scale models of new cars in virtual reality for design purposes: designers and engineers can physically walk around a virtual car to see how it looks, which saves the cost and time required to build a physical model.

Volkswagen and Toyota use this tech to train workers and Unity-based programs allow customers to virtually inspect a car on a screen before heading to the dealership [69], [70].

Moreover, Unity established a partnership with DeepMind [71], a world leader in artificial intelligence research. DeepMind uses Unity engine to train artificial intelligence following the philshophy pointed out by Danny Lange, vice-president of Artificial intelligence and Machine Learning at Unity Technologies [72]:

> *Games are in many, many ways much closer to nature than people think... You get the visual, the physics, the cognitive, and the social aspect–the interaction... These all put evolutionary pressures on algorithms, just as nature does on living things.*

To conclude the whole disquisition it is worth to cite the talk given by Hannah Luxenberg at University of Padua on April 24, 2019 [73]. Hannah is a filmmaker and the founder and creative director of *DreamShip* [74]. She is deeply interested to create stories that matter and on psychological health research. Indeed, the purpose of Hannah and of the *DreamShip* foundation is:

> *To bring dreams alive for children, families and those toward end-of-life through virtual reality by means of distraction therapy.*

Hannah works closely with patients, doctors, and research teams to develop the next therapeutic technology programs.

Kodak asserted:

(a) Hannah giving a talk at University of Padua on April 24, 2019 [73].

(b) Hannah working with children [75].

Figure 3.7: Hannah Luxenberg working with Unity.

> *Hannah effectively envelops patients in calming and engaging therapeutic scenery, providing an alternative means of therapy and a method for coping with emotional and physical pain through the power of virtual reality [76].*

Figures 3.5, 3.6 and 3.7 display the mentioned applications.

In this project, it is used *3D Characters* and *Animation Clips* provided by Mixamo [42] in FBX format, while, animations are driven by personally created *Animation Controllers* and *Scripts*.

4

Video Coding

The last twenty years have seen a significant technological revolution that changed the way moving images are created, shared and watched: the world shifted to digital video.

The key leading to the widespread adoption of digital video technology is video compression that will be the subject of this Chapter.

Both spatial and temporal compression will be addressed, particularly focusing on the latter one from which the whole study originates.

Additionally, VBR and CBR encoding are explained and the standard used in this project, the H.264 codec, will be presented.

4.1 Introduction to video coding

Before entering the core of video coding, let's give some basic definitions:

- *Frame*: it is an image representing the complete visual scene at a specific point in time;

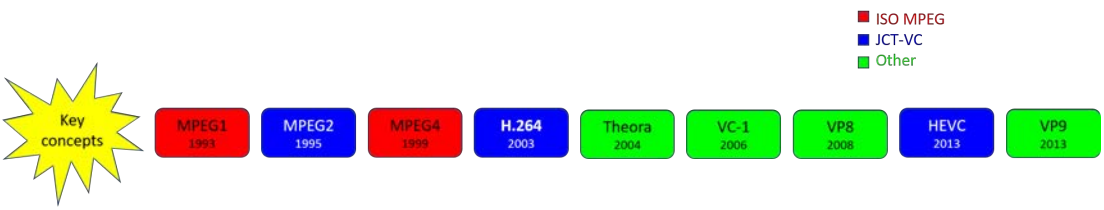


Figure 4.1: Timeline of major video coding standards and formats.

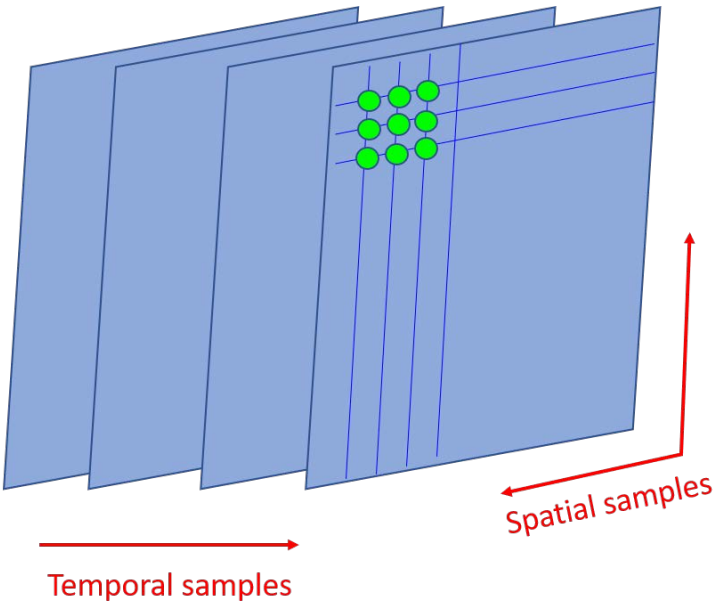


Figure 4.2: Spatial and temporal sampling of a video sequence.

- *Field*: it consists of either the odd-numbered or even-numbered lines within a video frame. A field corresponds to half frame;
- *Digital video*: it is a representation of a natural or real-world visual scene, sampled spatially, usually on a rectangular grid, and temporally.

  The temporal samples are the frames or the fields. Hence, a digital video is a sequence of frames (or fields).

Sampling is repeated at intervals (e.g. 1/25 or 1/30 second intervals) to produce the appearance of motion. When a video signal is sampled as a series of complete frames we talk about *progressive sampling*, when it is a sequence of fields, we call it *interlaced sampling*. The advantage of the second method is that it is possible to send twice as many fields per second as the number of frames in an equivalent progressive sequence with the same data rate, giving, therefore, the appearance of smoother motion.

The temporal sampling rate influence the motion perceived: the higher the rate, the smoother the motion appears. When the rate is too small, the motion is jerky and unnatural but the video is more suitable for low bit rate communications.

Raw, or uncompressed digital video, typically requires a large bitrate; whence, video compression (or video coding) is essential to save transmission bandwidth and storage space.

Figure 4.3: Video coding scenario.

Video coding will be the subject of the following sections.

4.1.1 H.264 and HEVC standards

Many techniques for video coding have been proposed and researched. Hundreds of research papers are published each year describing new and innovative compression techniques [77].

In contrast to this wide range of innovations, commercial video coding applications tend to use a limited number of standardized techniques for video compression and H.264 is considered to be the state-of-the-art standard.

H264, also referred to as MPEG-4 Part 10 or Advanced Video Coding (MPEG-4 AVC), is a video compression standard jointly published by the International Telecommunications Union (ITU-T) and the International Standards Organisation (ISO/IEC)in 2003.

It builds on the concepts of earlier standards such as MPEG-2 and MPGE-4 Visual, offering better compression efficiency (i.e. better-quality compressed video) and greater flexibility in compressing, transmitting and storing video.

Better video compression is the key to satisfy the ever increasing demand for higher quality video.

The biggest advantage of H.264 over previous standards is its compression performance: it allows to achieve better image quality at the same compressed bitrate.

The benefits of H.264/AVC come at a price of a greater computational cost: it can take significantly more processing power to compress and decompress videos.

Together with its improved compression performance, H.264 introduces higher flexibility in terms of compression options and transmission support. This make the H.264 format suitable for a wide range of applications ranging from High Definition DVDs to videoconferencing or mobile video services [78].

High Efficiency Video Coding (HEVC), also known as H.265, is the standard designed to be the successor of H.264.

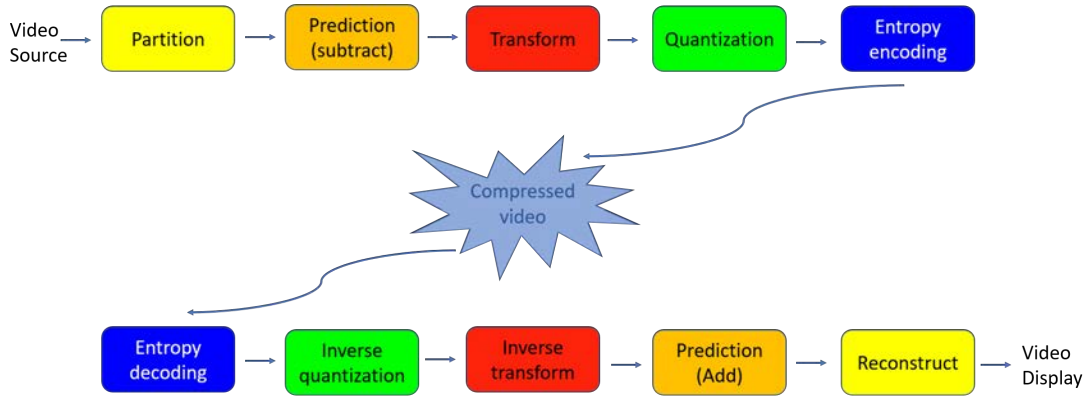


Figure 4.4: Video coding pipeline.

HEVC standard was first published in January 2013 by the Joint Collaborative Team on Video Coding (JCT-VC), a collaboration between Moving Picture Experts Group (MPEG) and Video Coding Experts Group (VCEG) [79].

Although HEVC offers significantly higher compression than earlier standards, it is still not popular and has not replaced its predecessor, yet, due to the required computing power.

This new technology is on standby until the hardware market adapts to it, as it happened with H.264, launched in 2003 but only gaining popularity a few years later [80].

In Figure 4.1 it is shown the timeline of the major video coding standards and formats [81], while, in Figure 4.2 it is illustrated the spatial and temporal sampling of a video sequence.

4.2 Video coding concepts

4.2.1 The CODEC model

Video compression (or *video encoding*) is the process of reducing the amount of data required to represent a digital video signal, prior to transmission or storage. The complementary operation, decompression (or *decoding*), recovers a digital video signal from a compressed representation, prior to display to the final user [77].

Video coding involves a complementary pair of systems, a compressor (*encoder*) and a decompressor (*decoder*), that are usually built into a device such as a video camera or a DVD player. The encoder/decoder pair is often described as a *CODEC* (enCOder/DECoder).

Data compression is achieved by removing spatial and temporal redundancy. Spatial redundancy removal exploits correlation between adjacent pixels (*intra-prediction*) while, temporal reduncancy removal leverages correlation between adjacent frames (*inter or motion*

prediction).

Furthermore, compression techniques can be categorized into:

- *lossless compression*: with this type of compression, reconstructed data are perfect copies of the original ones. The best compression ration achievable is around 3-4 times;
- *lossy compression*: with this type of compression, the decompressed data are not identical to the source ones. Lossy compression is necessary to achieve higher compression at the expense of a loss of visual quality.

Lossy video compression systems are based on the principle of removing *subjective redundancy*: the elements removed are the ones that do not significantly affect the viewer's perception of visual quality.

A video encoder consists of three main functional units:

1. *a prediction model*: it attempts to reduce both spatial and temporal redundancy taking as input the raw video sequence.

   In H.264/AVC, the prediction is formed from data in the current frame or in one, or more, previous and/or future frames.

   The output of the prediction model is a *residual frame*, created by subtracting the prediction from the actual current frame and a set of model parameters, indicating the intra-prediction type or describing how the motion was compensated;

2. *a spatial model*: the residual frame forms the input to the spatial model whose aim is to reduce spatial redundancy.

   In H.264/AVC this is carried out by applying a transform to the residual samples and quantizing the results.

   The transform converts the samples into another domain in which they are represented by transform coefficients. The coefficients are quantized to remove insignificant values providing a more compact representation of the residual frame.

   The output of the spatial model is a set of quantized transform coefficients;

3. *an entropy encoder*: the parameters of the prediction/spatial model are compressed by the entropy encoder that removes the statistical redundancy in the data.

   The entropy encoder produces a compressed bit stream or file that may be transmitted and/or stored. A compressed sequence consists of coded prediction parameters, coded residual coefficients and header information.

The purpose of transform coding is to further reduce redundancy.

Many transforms have been proposed. These are categorized into *block-based* and *image-based*. Samples of the former are: Karhunen-Loeve Transform (KLT), Singular Value Decomposition (SVD) and Discrete Cosine Transform (DCT) while, the most popular image transform, is the DiscreteWavelet Transform (DWT), or just wavelet.

The video decoder reconstructs the video frame from the compressed bit stream exploiting, in turn, a system symmetrical to the one just mentioned.

Note that not only spatial and temporal redundancy can be considered, but also color space transformations can be adopted: after being captured, an RGB image may be converted to YCbCr space in order to reduce storage and/or transmission requirements.

4.2.2 Spatial compression

In the spatial domain, there is usually a high correlation between pixels that are close to each other.

Spatial compression is achieved through successive steps:

1. Conversion from RGB image to YCbCr format;
2. Macroblock partitioning;
3. Transform coding;
4. Quantization;
5. Entropy coding.

For further details, the reader is referred to [77].

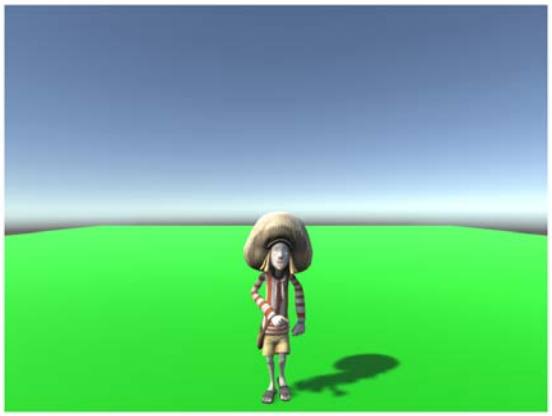
4.2.3 Temporal compression

Temporally adjacent frames are often highly correlated, especially if the temporal sampling rate, or frame rate, is high. Therefore, temporal prediction is enacted in order to transmit only additional information.

Temporal prediction is created from one or more, past or future frames, known as *reference frames*.

The simplest method of temporal prediction is to use the previous frame as the predictor for the current one. In Figures 4.5a and 4.5b, two successive frames of a video sequence are

(a) Predecessor frame.



(b) Current frame.

(c) Residual frame.

Figure 4.5: Temporal prediction without motion compensation.

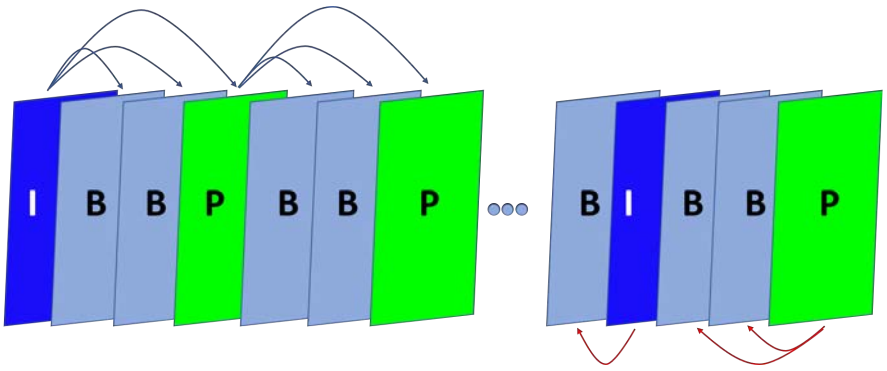


Figure 4.6: Classical Group of Pictures structure.

shown. The former is the predecessor of the latter frame. Figure 4.5c displays the *residual frame*, formed by subtracting the predictor from the current frame.

From the last figure, we not that there is still a lot of redundant information. For this reason, *motion estimation* and *compensation* techniques are implemented to improve the accuracy of the prediction.

Several motion estimation techniques can be implemented. Basically, these work by partioning the frame in smaller blocks and computing a *motion vector* for each of them. Whence, the residual frame is no longer the previous one but the previous frame with each block shifted according to the motion vectors.

4.2.4 Group of Pictures

A video stream is a sequence of frames, as mentioned in section 4.2.1. These frames are thoroughly organized as a succession of Group of Pictures (GOPs).

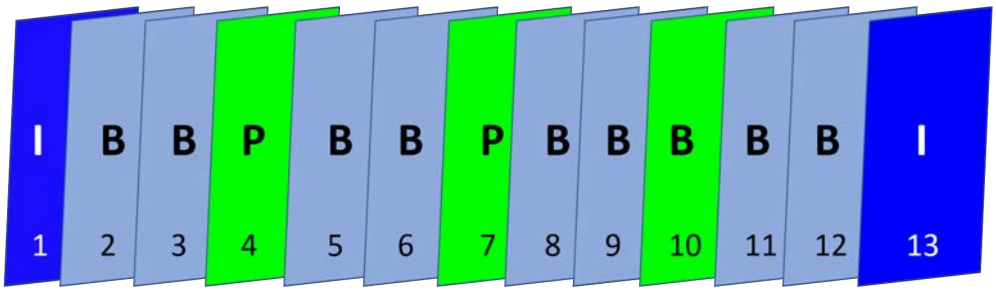
A GOP can contain the following frames types:

- *I frame (intra coded picture)*: it is coded independently of all other pictures. It contains the full image and does not require any additional information to reconstruct it.

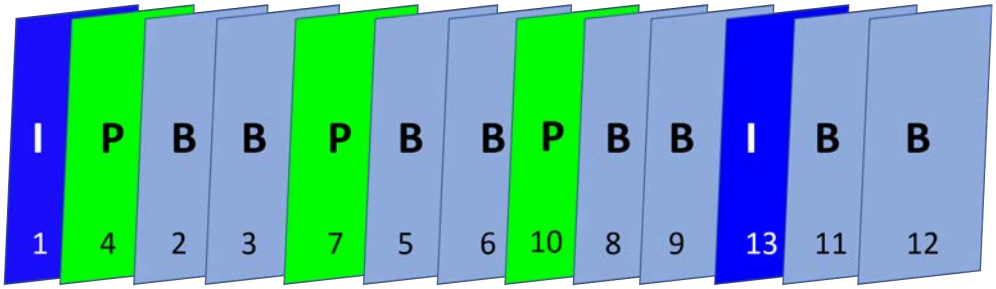
  An I frame indicated the beginning of a GOP;

- *P frame (predictive coded picture)*: it is predicted with motion compensation from the preceding I or P slices. Hence, only from past frames. In older designs, P pictures can reference only one picture: the I or P frame that preceeds it;

- *B frame (bipredictive coded picture)*: it is predicted with motion compensation from the I and/or P slices on either side of it. Therefore, both from past and future frames.



(a) Display ordering.



(b) Bitstream ordering.

Figure 4.7: The two corresponding sequence ordering types.

B frames are inserted between I and P slices and are the ones that achieve the highest level of compression. In older designs, B images can reference only two pictures: the reference frame that preceeds and the one that follows it.

In older formats, the ordering and referencing structure were rather constrained. In newer techniques, such as H.264 and HEVC, there is much more flexibility and also B frames can serve as references when coding other B or P pictures. Moreover, multiple referencing and hierarchical referencing structure are allowed [77]. This extra flexibility can improve compression efficiency.

The GOP arrangement is referred by two numbers defining:

- *the GOP structure*: it indicates the number of B frames between two reference frames (either I or P). In recent designs, a variable GOP structure is allowed;
- *the GOP size*: it denotes the number of frames within a GOP. Whence, the distance between two I frames.

This two parameters involves a trade-off between compression efficiency and quality of prediction.

To conclude, the GOP organization allows to distinguish two different sequence orders:

- *display order*: the order in which the video sequence is displayed to the user;
- *bitstream order*: the order in which the video sequence is coded. This ordering change in the coding phase is necessary because of the reliance of the B frames on future anchor frames: future frames must be known before predicting B ones.

In Figure 4.7a, it is shown a sample of display ordering, while, in Figure 4.7b, it is shown the corrispondent bitstream ordering. The frame being predicted and the frame upon which the prediction is based are not necessarily adjacent [82].

4.3 VBR vs CBR

Video data can be encoded in several modes.

The two considered in this thesis are:

- *Variable bitrate (VBR)*: with this mode, the bitrate changes according to the image content.

  Higher bitrate is assigned to the more complex frames, lower bitrate to the simpler ones.

  VBR is usually achieved setting a fixed quality or a quality range. However, a bitrate range or an average bitrate can also be imposed [83], [84];

- *Constant bitrate (CBR)*: with this mode, the bitrate is kept the same throughout the encoding process, regardless the frame complexity [85].

Generally, VBR can retain the best image quality during video streaming. However, CBR is preferable when dealing with constrained environments or when quality is not required.

Additionally, VBR may take more time to encode, as the process is more complex, and may arise problems when bitrate exceeds the communication bandwidth [86], [87].

Nevertheless, intermediate approaches can be adopted according to the needing.

In this work, video encoding is performed exploiting the *FFmpeg* tool [88] that will be presented in Chapter 7.

Basically, a protocol is an agreement between the communicating parties on how communication is to proceed.

TANENBAUM, WETHERALL

5 Protocols for video transmission

With the rapid improvement of smart technologies, Internet Protocol (IP) cameras are advancing over *Closed-Circuit Television* (CCTV) ones [89]. Since video sequences are transmitted over public networks, it is necessary to verify whether the characteristics of compressed video packet streams are kept by encrypted traffic in order to prove the robustness of the designed algorithms.

For this reason, Samuele Piazzetta, a collaborator of this project, analyzed encryption protocols employed in video surveillance transmissions and created a tool to simulate secure streaming.

This Chapter is intended to provide some notions of a data communication system. A short overview of transmission protocols is presented and some state-of-the-art encrypted video surveillance systems are presented.

Additionally, possible approaches to retrieve orginal data from encrypted ones are proposed.

5.1 Introduction to protocols

5.1.1 Data communication system

A data communications system has five components [3], as it can be seen from Figure 5.1:

- *the message*: it is the information (data) to be communicated. Popular forms of information include text, numbers, pictures, audio, and video;

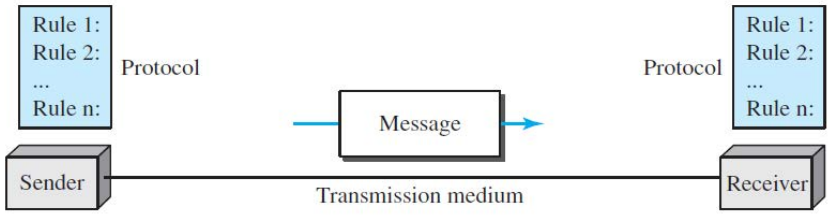


Figure 5.1: Five components of data communication systems [3].

- *the sender*: it is the device that sends the data message;
- *the receiver*: it is the device that receives the message. Both sender and receiver can be a computer, workstation, telephone handset, television, and so on;
- *the transmission medium*: it is the physical path that allows the message travelling from sender to receiver. Some examples of transmission media include twisted-pair wire, coaxial cable, fiber-optic cable, and radio waves;
- *the protocol*: it is a set of rules governing data communications. It can be seen as an agreement between the communicating devices: without it, two devices may be connected but not communicating.

Hence, a protocol defines the rules that both the sender and receiver and all intermediate devices need to follow to communicate effectively . When communication is simple, one simple protocol is sufficient; when the communication is complex, we may need to divide the task between different layers, therefore, we need a protocol at each layer, or *protocol layering*. This is the reason for which we talk about *layered architecture* [3], [4].

In the following section, layered architectures are briefly discussed.

5.1.2 Layered Architecture: TCP/IP protocol suite

Protocol layering enables to divide a complex task into several smaller and simpler tasks [3], [4]. The purpose of each layer is to offer certain *services* to the higher layers while hiding implementations details. Hence, each layer of the architecture receives a set of services from the lower layer and gives services to the upper layer.

TCP/IP is a *protocol suite* (a set of protocols organized in different layers) used in the Internet today. Whilst everyone talks about the TCP/IP protocol suite when speaking of the Internet, this not the only suite of protocols defined.

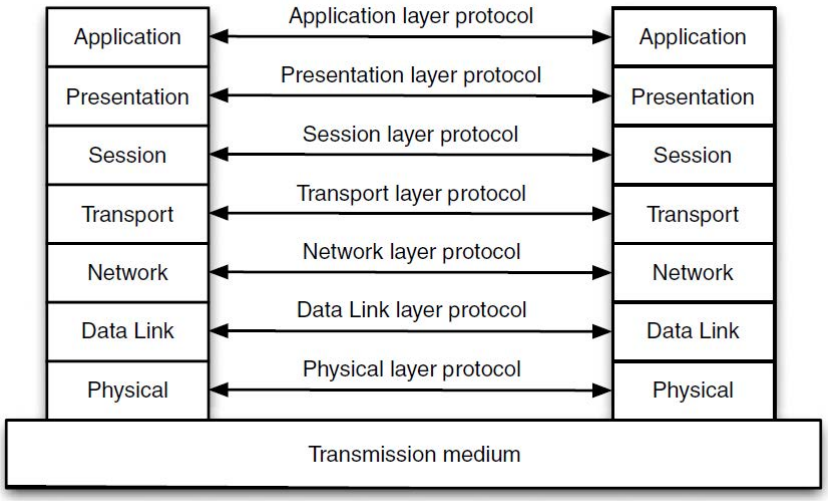


Figure 5.2: The ISO/OSI model [4].

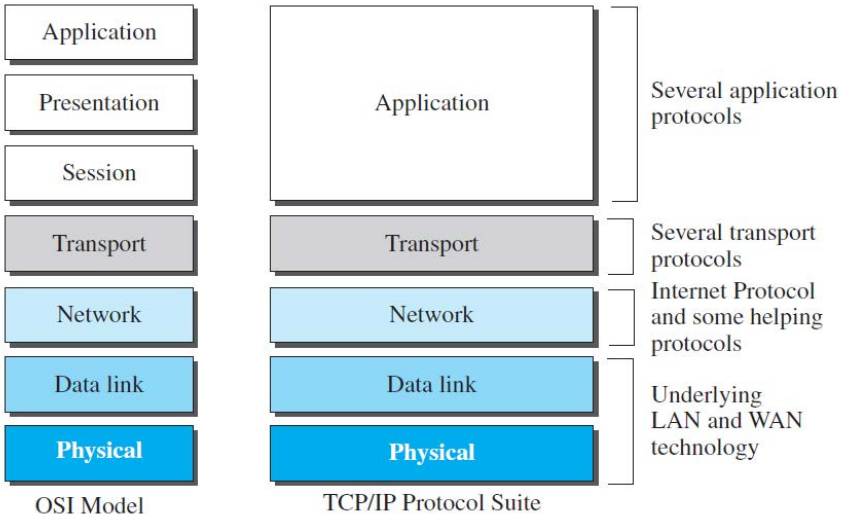


Figure 5.3: TCP/IP and OSI model [3].

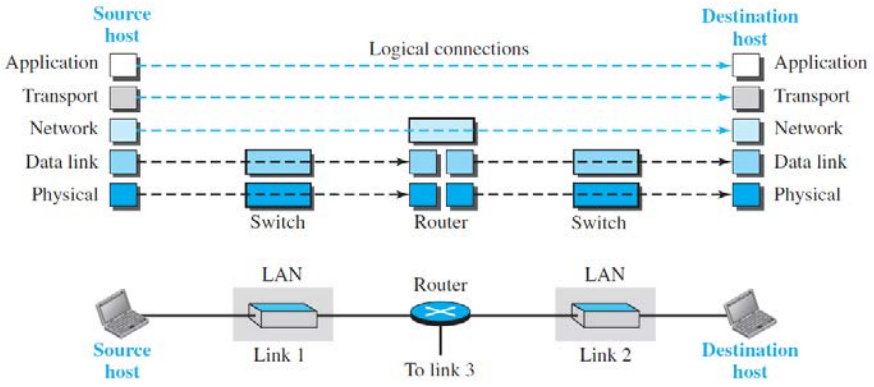


Figure 5.4: Logical connections between layers of the TCP/IP protocol suite [3].

In the late 1970s, the Open Systems Interconnection (OSI) model was developed by the International Organization for Standardization (ISO).

The subdivision in layers for the TCP/IP protocol suite and for the OSI model can be seen in Figure 5.3. Differently from the OSI model, the TCP/IP suite does not have the *session* and *presentation* layers: these are included on the *application* one.

The full description of the role of each layer can be found in [3].

The OSI model appeared after the TCP/IP protocol suite but it failed to replace its predecessor.

The entities comprising the corresponding layers on different machines are called *peers* [4]. The peers may be software processes, hardware devices, or even human beings. The peers that communicate among each other using some specific protocols, as shown in Figure 5.2. For this reason, we talk about *logical connections*: we can think that a layer-to-layer communication exists. In reality, no data are directly transferred from corresponding layers of differente machines. Instead, each layer passes data and control information to the layer immediately below it, until the lowest layer is reached.

In Figure 5.4, it is reported a sample of communication flow through Internet. From this Figure, it is evident the meaning of logical connections. Moreover, it can be noticed that not all the devices need to implement the full layered structure [3].

5.1.3 Encapsulation and Decapsulation

Encapsulation and *decapsulation* are two essential elements in protocol layering. To understand what they stand for, it is necessary to introduce two terms:

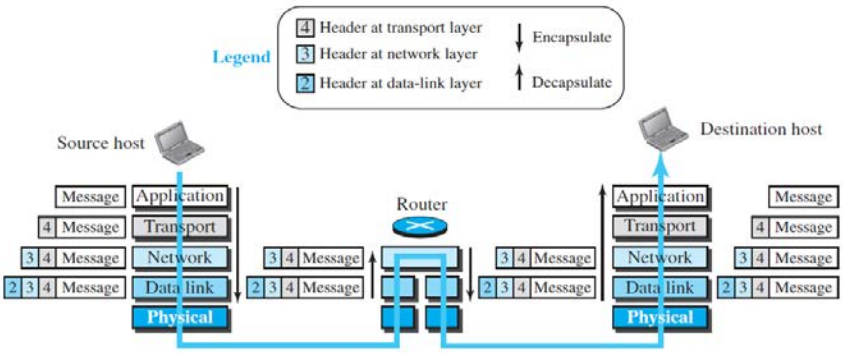


Figure 5.5: Encapsulation/Decapsulation [3].

- *payload*: the actual data to be exchanged;
- *header*: supplemental data placed before the payload.

Sometimes, besides headers, we talk about *trailers* whose scope is the same but they are placed after the payload.

Headers and trailers contains the identifiers of the source and destination plus some more information needed for the end-to-end delivery of the message. Hence, they serve to control the data exchange (such as error control or congestion control). Each packet consists of payload and control information.

During encapsulation, each layer builds its packet adding its own header to the payload received from the upper layer while, during the decapsulation, the inverse process happens.

In Figure 5.5 it is graphically reported this concept. It is evident that encapsulation resides on the source while decapsulation on the destination host. Furthermore, intermediate devices implements both the processes [3].

5.1.4 Standard and Administration

When dealing with Internet and protocols, we often see references to *standard* and *administration*. Therefore, these two concepts are worthy of mention.

An *Internet standard* is a formalized regulation that must be followed. It is a thoroughly tested specification that is useful to and adhered to by those who work with the Internet. A specification attains Internet standard status after passing a strict procedure.

Internet administration, instead, is the set of groups that coordinate Internet issues and that guide its growth and development. Figure 5.6 shows the general organization of Internet administration.

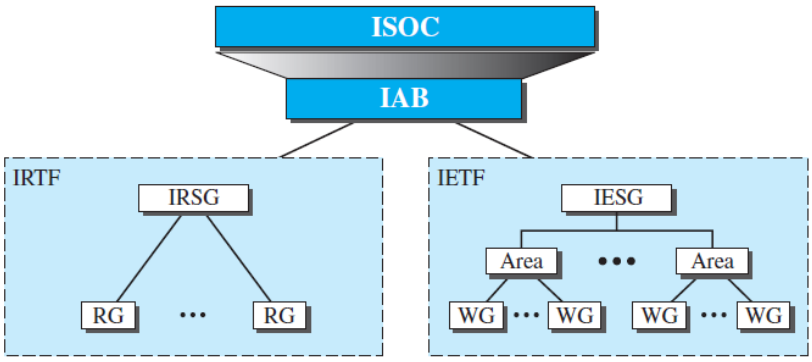


Figure 5.6: Internet Administration [3].

The protocols presented in the following sections can be thought as application layer protocols [3].

5.2 Application protocols established by ONVIF

ONVIF (Open Network Video Interface Forum) [90] is an organization started in 2008 by Axis Communications, Bosch Security Systems and Sony that creates standards to govern the communication of IP products within video surveillance and other physical security areas.

Nowadays it inlcudes 480 member companies categorized in five levels of membership: *observer*, *user*, *contributing*, *affiliate* and *full member*.

The *ONVIF Streaming Specification* [91], indicates the RTP and RTCP protocols as standards for real-time streaming.

Whence, these protocols will be the subject of the following subsection.

Physical Security Interoperability Alliance (PSIA) [92] is the second group formed in 2008 to address IP video surveillance standardization issues. PSIA is quite smaller in number than ONVIF but both pursue the same goal of bringing interoperability to IP-based security systems.

5.2.1 RTP and RTCP

Real-time Transport Protocol (RTP) [93] is the protocol designed to handle real-time traffic on the Internet. It was developed by the Audio-Video Transport Working Group of the Internet Engineering Task Force (IETF) and first published in 1996.

RTP is used in conjunction with the RTP Control Protocol (RTCP):

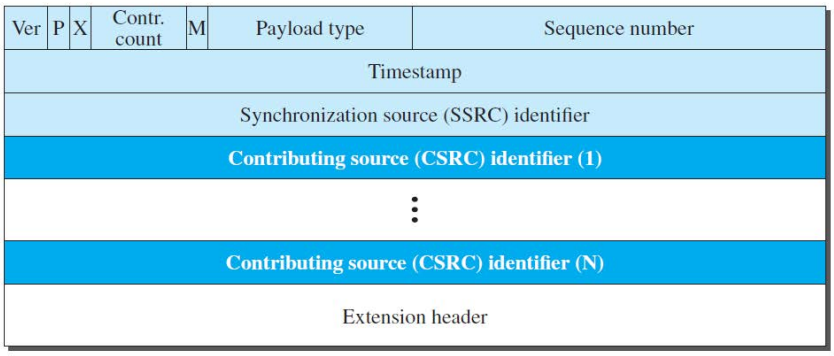


Figure 5.7: RTP packet header format [3].

- *RTP*: it carries the real time media streams (e.g., audio and video);
- *RTCP*: it handles the communication control.

The primary function of RTCP is to provide feedback on the quality of service by periodically sending statistics information such as packet count, packet loss, packet delay and round trip time. Hence, it partners with RTP in the delivery and packaging of multimedia data, but does not transport any media data itself.

An RTP packet includes:

- an *RTP payload*: the actual data to be transmitted. For example audio samples or compressed video data;
- an *RTP header*: it includes a fixed part plus optional header extensions that may be present. In Figure 5.7 it is represented the RTP header format. The first twelve bytes are present in every RTP packet, while the list of CSRC identifiers is the variable part.

RTP typically runs over User Datagram Protocol (UDP) [94], whence, RTP packets are successively encapsulated into UDP ones. UDP is preferable to Transmission Control Protocol (TCP) [95] since it generates a smaller overhead traffic that enables a faster transmission [91].

RTCP is the sister protocol of RTP. This means that the UDP sometimes carries RTP payloads while other times the RTCP ones.

RTCP is based on the periodic transmission of control packets and its five common packet types are: *Sender Report Packet*, *Receiver Report Packet*, *Source Description Packet*, *Bye Packet* and *Application-Specific Packet* [3].

The Real Time Streaming Protocol (RTSP) [96] is another protocol designed to be used in entertainment and communications systems. It allows to control the playing of audio/video providing clients with a set of VHS-style remote control commands, such as play, record and pause. However, the transmission of streaming data itself is not a task of RTSP. At this purpose, most RTSP servers use the RTP in conjunction with RTCP for media stream delivery.

RTP and RTCP do not provide any flow encryption or authentication methods. Security measures are implemented adopting approaches discussed in the following session.

5.3 Protocols for secure data transfer

According to ONVIF, data protection can be achieved following two options:

1. *Adopting the secure profile of RTP*, called Secure Realtime Trasport Protocol (SRTP);
2. *Tunneling RTP, RTCP and RTSP protocols over HTTPS*;

Both the procedures are described in this section.

5.3.1 SRTP ans SRTCP

The Secure Real-time Transport Protocol (SRTP) [97] is intended to provide encryption, message authentication and integrity, and replay attack protection to RTP packets.

Secure RTCP (SRTCP) is its syster protocol that corresponds to the secure version of RTCP.

The *Encrypted Portion* of SRTP/SRTCP packets consists in the encryption of the RTP/RTCP payloads (including RTP padding when present) of the equivalent RTP/RTCP packets.

5.3.2 HTTPS

A *tunneling protocol* is a communication protocol that allows data movements in different networks through encapsulation processes.

Usually, it is used to transfer packets from a private network to a public one. The encapsulation process allows data packets to be transmitted across a public network while keeping a certain privacy level on them.

Tunneling is also used to allow a protocol to operate over a network that does not support that particular protocol, e.g., running IPv6 over IPv4; tunneling also permits increasing the

security of unencrypted data since, thanks to the encapsualtion system, they hide the whole packet in the payload [98].

HTTPS tunneling is, therefore, the process by which messages are encapsulated using HTTPS protocol (the secure version of the Hypertext Transfer Protocol [99], also called HTTP over TLS [3]).

Tunneling RTP, RTCP and RTSP protocols over HTTPS [100] allows the transfer of packets on which the encryption is applied. In this way, header information of the three protocols is completely lost.

Encryption in HTTPS packets is implemented by the TLS protocol [101].

Both SRTP and most of the cypher suites used by TLS follow the Advanced Encryption Standard (AES) for data encryption. The AES algorithm is briefly depicted in the following section where it is shown that the padding added by it is negligible.

5.3.3 AES

When Data Encryption Standard (DES) started to manifest some limitations, the U.S. National Institute of Standards and Technology (NIST) started an open process to select a new encryption algorithm that would be called Advanced Encryption Standard.

In January 1997, researchers from all over the world were invited to submit proposals for this new standard and, in October 2000, NIST selected the Rijndael algorithm [102]. The name Rijndael is derived from the last names of its inventors, two young Belgian cryptographers: Vincent Rijmen and Joan Daemen. In November 2001, Rijndael became the AES U.S. Government standard, published as FIPS (Federal Information Processing Standard) 197 [103].

This algorithm has become the world's dominant cryptographic cipher thanks to its security, performance, efficiency, implementability, and flexibility.

AES is a *symmetric-key* algorithm, i.e., the same key is used for both encrypting and decrypting the data. De facto, it has two variants: a 128-bit block with a 128-bit key and a 128-bit block with a 256-bit key.

Rijndael uses substitution, permutations and multiple rounds (from 10 to 14).

It is based on Galois field theory that has three parameters:

1. *plaintext*: an array of 16 bytes containing the input data;

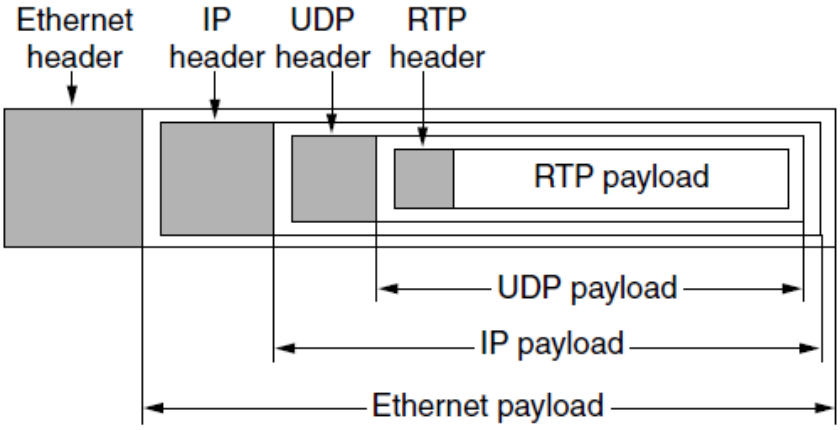


Figure 5.8: RTP packet encapsulation [4].

2. *ciphertext*: an array of 16 bytes where the enciphered output will be returned;
3. *key*: the 16-byte key.

Without getting into mathematics, only the interesting aspects relevant for this project are pointed out:

1. *being a block cipher algorithm*, it operates on block of fixed length. Whence the maximum number of padding bits added to the original video are 128;
2. *same plaintext blocks are mapped into distinct cyphertext blocks* to enhance security. This mechanism is achieved thanks to different *mode of operations* [104];
3. *the algorithm has been designed not only for great security, but also for great speed*. A good software implementation on a 2-GHz machine should be able to achieve an encryption rate of 700 Mbps, which is fast enough to encrypt over 100 MPEG-2 videos in real time.

Hence, AES algorithm is suited also for the less performing hardware devices on the market such as the camera used in this work. An high level description of the whole AES algorithm can be found in [4].

5.4 Lesson Learned

To summarize the conclusions of [105], we can deduce that:

1. *in SRTP streaming*: only payloads are encrypted, together with any padding.

   To retrieve the actual frame size, we can:

   - use the *mark flag* (M): this flag is usually used in the RTP header to mark the end of the data stream;
   - if RTP packet does not use the mark flag to signal the final packet of a given frame, it is possible to use the *timestamp* field of the RTP header. The timestamp reflects the sampling instant of the first byte in the RTP data packet. Packets belonging to the same video frame carry the same timestamp value.

2. *when transmission is carried out through HTTPS tunneling*: headers information is completely lost. As a consequence, we can not retrieve mark flag or timestamp fields.

   Nevertheless, two options are available to deal with this issue:

   - if a TLS packet is used for each RTP one,the arrival time could be use to extract the original data removing TLS and RTP headers;
   - if a TLS packet is used for more RTP packets, we are no longer able to understand the boundaries of each frame, hence, we are no longer able to recover the actual frame size. In this case, the only solution that can be addressed is to count the number of bytes transferred in each interval t, where t stands for the time elapsed between the reception of two TLS packets.

3. *the AES algorithm works on blocks of 128 bits*: thus, the padding added by it can be considered negligible since the packets dimensions we are working with are much higher.

   Let's consider, for instance, a packet of 2000 bytes, introducing an error caused by added padding of 15 bytes, the percentage error that we would get is less than 1%.

Finally, to perform the ultimate payload extraction, the headers of all the layers involved must be considered.

Let's exainate the layered architecture shown in Figure 5.8. There are:

- header RTP: 12 bytes;
- header UDP: 8 bytes;
- header IPv4: 20 bytes;

- header Ethernet: 14 bytes [4].

The outcomes provided by the tool designed by Samuele will be reported in Chapter 8.

Learning is the process of converting experience into expertise or knowledge

6 Machine Learning

Machine Learning is one of the most influential and powerful technologies in today's world.

The ever increasing amount of data becomes useless unless we are able to analyse it and find the patterns hidden within. Machine learning is the tool that helps human beings in doing that efficiently turning information into knowledge.

In this Chapter, the machine learning environment is introduced presenting the basic concepts behind the whole development of the project.

These are essentially four:

- *Convolutional neural networks;*
- *Incremental learning;*
- *PCA and eigendecomposition;*
- *Decision trees.*

6.1 Introduction to Machine Learning

Machine learning (ML) is one of the fastest growing areas of computer science, with far-reaching applications: from Internet fraud detection and customer predictions till precision agricolture [106].

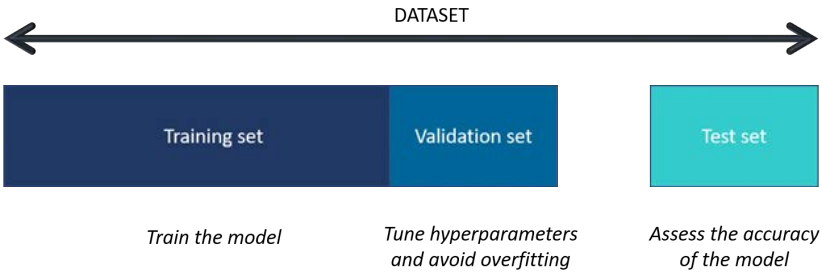


Figure 6.1: Dataset splitting: training, validation and test sets.

Machine learning is a set of methods that give computer systems the ability to *learn* from data to make predictions about novel data samples. As intelligent beings acquire or refine many of their skills through *learning from experience*, ML tools are concerned with endowing programs with the ability to learn and adapt. Therefore, roughly speaking, learning is *the process of converting experience into expertise or knowledge* [107].

In the ML terminology, three types of data can be mainly discerned:

- *training data*: these are the input that the learner has access to; i.e. the data from which it acquires its knowledge. The model *sees* and *learns* from these data and it is *fitted* according to them;
- *test data*: these are the data used by the learner to assess the performance of its estimations; i.e. how good it is to make predictions. A test set is never used in the training process and it is only exploited once a model is completely trained;
- *validation data*: these data are used to tune the hyperparameters of a classifier. For example, when to stop the training of a neural network. The validation set is often used to avoid overfitting (i.e. to stop the training before the model fits too well the input data). Hence, it can be seen as an hybrid between a training and a test set: it is not properly used to the train the learner but it is leveraged to select the best parameters by estimating the model accuracy. The model *occasionally sees* these data, but *never learns* from them.

The split of the whole dataset into train, validation and test sets is performed according to the needs. Usually the first is the biggest one while the last is the smallest as shown in Figure 6.1. A common split could be 70/20/10 %.

Learning is a very wide domain. Consequently, the field of machine learning has branched into several subfields dealing with different types of learning tasks. As shown in Figure 6.2, three main categories can be distinguished:

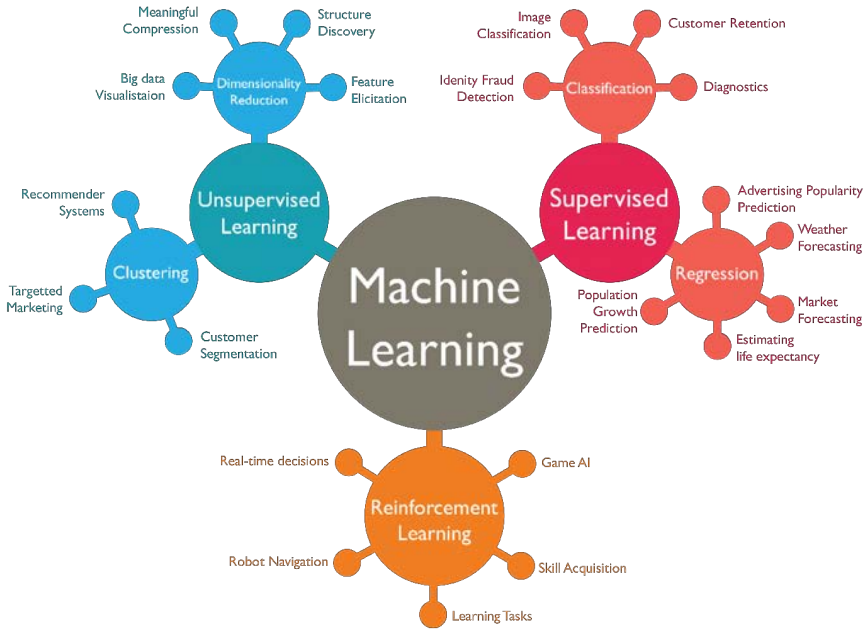


Figure 6.2: Machine learning branched into several subfields [5].

- *supervised learning*: the training set is provided with labels. Hence, the ground truth of the training data is known. The model is trained to learn the mapping between the inputs and the outputs in order to make good predictions on newer samples.
- *unupervised learning*: the training set is not labelled. The aim of unsupervised algorithms is to infer the inner structure of those data, finding common patterns and trying to cluster them into categories.
- *reinforcement learning*: there are not training data at all but the agent (or learner) interacts with the surrounding environment learning through a system of *reward and penalty*. The agent learns without intervention from a human by maximizing its reward and minimizing its penalty. Therefore, the agent acquires knowledge in the same way as a child learns to perform a new task.

In this thesis, both supervised and unsupervised learning are implemented.

The former is conceived to solve classification task through convolutional neural networks (CNNs), the latter is used to cluster new data by means of decison trees. With this clustering technique, labels can be assigned to new data and, when a sufficient amount of new samples are collected, incremental learning is performed to make CNN capable to classify the new classes.

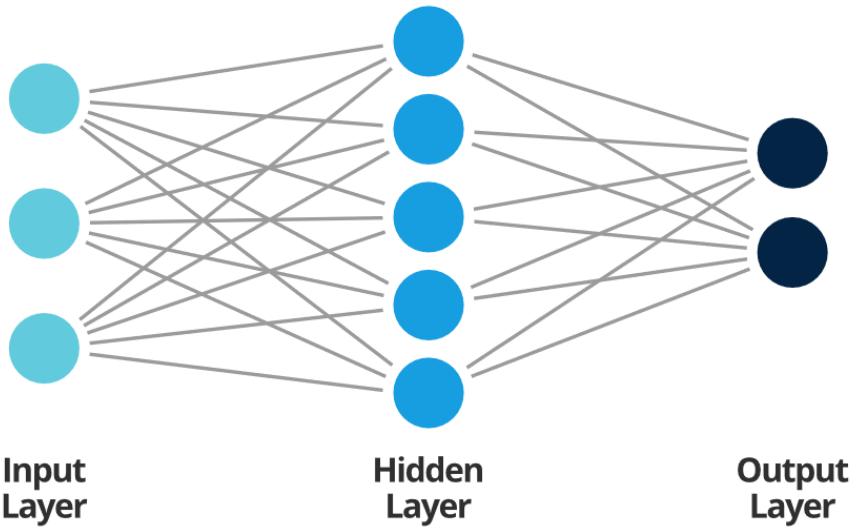


Figure 6.3: Sample of a FFNN architecture [6].

CNNs belong to the bigger Neural Networks (NN) category that is the subject of the following section.

6.2 Neural networks

Artificial neural networks (ANN) are models of computation inspired by the structure of neural networks in the brain.

The brain is responsible for all processing and memory and it consists of a large number of basic computing units or atoms (*neurons*) connected to each other in a complex communication network.

ANN were first proposed in 1940-50 but practical results were lower than SVM and other techniques. From 2010 on, deep architectures took over, thanks to their impressive performances, and ANN are, nowadays, the workhorses of deep learning (DL) [108]. DL is a subfield of machine learning that exploits multiple layers to progressively extract higher level features from raw inputs.

A neural network is represented as a directed graph organized into layers where each neuron takes as inputs only the outputs of neurons of the previous layer. In particular, the simplest form of NN is the feedforward one (FFNN) that is described by a directed acyclic graph $G = (V, E)$ and a weighted function over the edges, $w: E \rightarrow \mathbb{R}$, where:

- V: are the nodes of the graph; i.e. the neurons. $|V|$ is the network size;

- E: are the edges of the graph; i.e. connections between neurons;

Each neuron:

- takes in input the sum of the outputs of the connected neurons from the previous layer weighted by the edge weights;
- applies to it a simple scalar function (*activation function* σ).

Figure 6.3 shows an example of a simple FFNN architecture.

The ultimate scope of a NN is to learn the weights in order to autonomously solve a particular task. A neural network learns by optimizing a particular *cost function* that depends on the predicted outputs and on the given targets.

Many parameters influence the neural networks performance such as:

- *the learning rate*: it is an hyperparameter that controls how quickly or slowly a neural network model learns a problem (i.e. weights are updated). Generally, a large learning rate allows the model to learn faster but the final set of weights is a suboptimal one. A smaller learning rate may allow the model to reach a optimal set of weights but may take significantly longer to train;
- *the cost function*: it is also called *loss function* or *objective function* and it quantifies the error between predicted values and expected ones. Many loss functions exist and [109] reports the one made available by keras; despite these, custom losses can also be defined;
- *the optimizer*: it is the algorithm used to optimize the objective function.

The zoo of neural network types grows exponentially; Figure 6.4 reports a chart showing many emerging architectures and approaches [7].

CNNs will be further illustrated in the following subsection since they are used in this project.

6.2.1 Convolutional Neural Networks

Convolutional neural neetworks are a type of feed-forward deep neural networks that have been tremendously successful in practical applications [110].

CNNs are more efficient than standard fully connected multilayer networks since they:

- *reduce the number of edges*: in fully connected FFNN, all nodes at layer t are connected to all nodes at layer t_{-1}. In CNN, only neighboring neurons are connected;

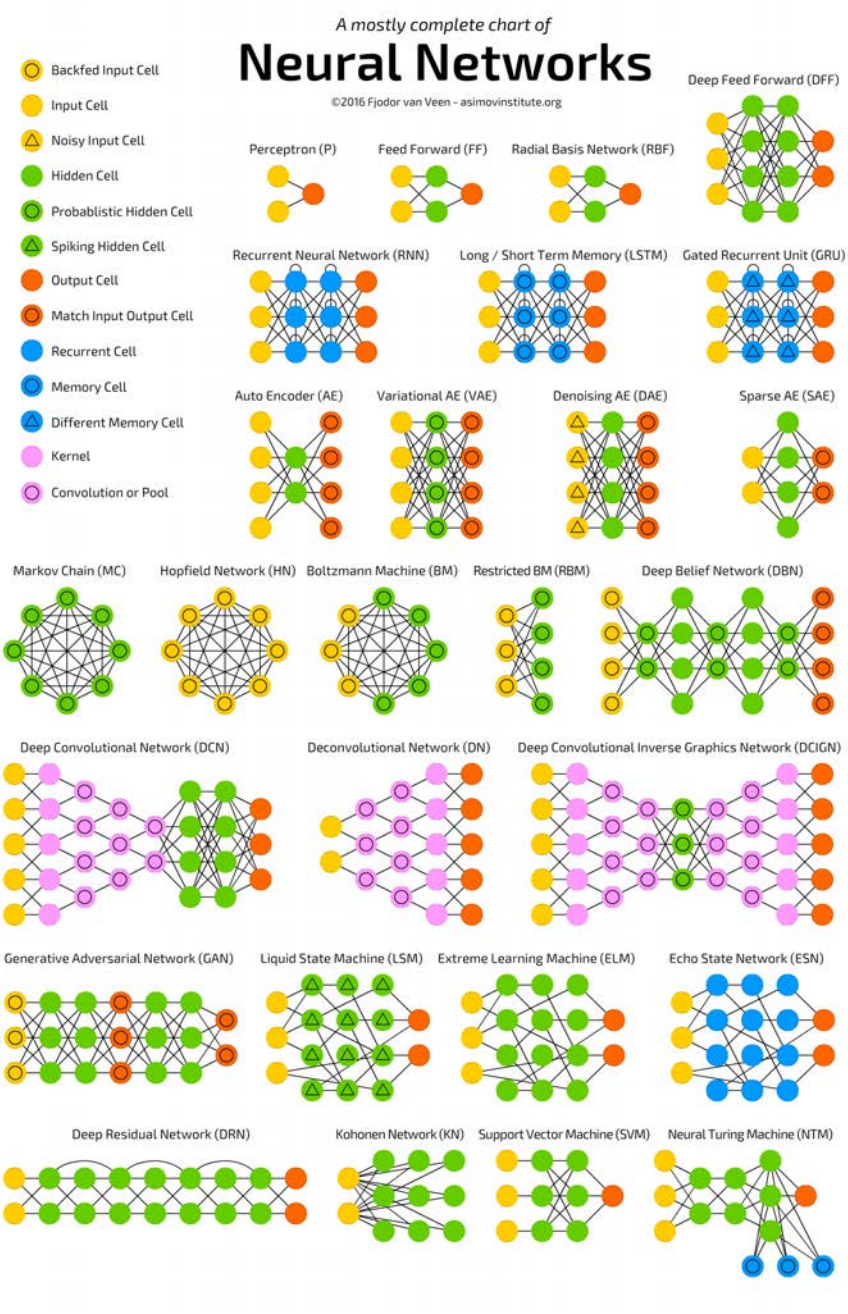


Figure 6.4: A mostly complete chart of neural networks [7].

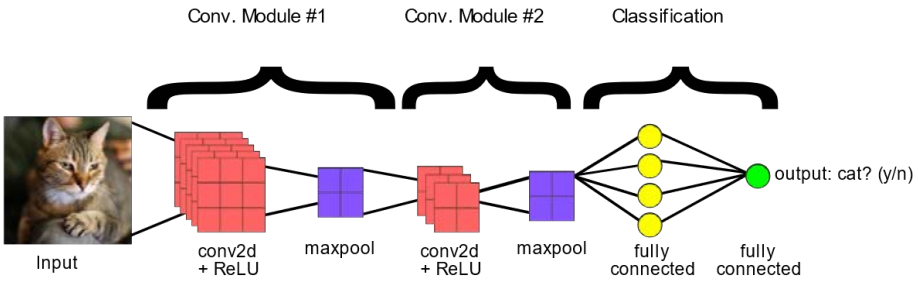


Figure 6.5: Sample of a CNN architecture [8].

- *exploit local connections*: a neuron can be *closer* to some neurons and less to others;
- *shared edge weights*: the same kernel is applied to the whole signal with a sliding window approach;
- *exploit pooling stages*: these stages progressively reduce the resolution allowing to extract higher level features from layer to layer.

A typical CNN layer consists of three stages:

- *convolutional stage*: in this stage, a *kernel* (or *filter*) is convolved with the input with a sliding window approach. The shift is controlled by the *stride* parameter. The height and the width of the kernel are design parameters, while the depth must be equal to that of the input data. Another parameter involved in this stage is the *padding* one that influence the output dimensions. Each convolutional stage can use multiple kernels;
- *detector stage*: a non linear activation function is applied, such as the rectified linear activation function (ReLU);
- *pooling stage*: a pooling function is used to further reduce the data dimensions. Originally, the pooling layer was introduced to reduce the computational burden and the memory requirements, but it turned out to be crucial to obtain state-of-the-art performances in many applications. Usually, the *max pooling* function is used since it works very well, it is efficient and it can be efficiently implemented in hardware [111].

Each CNN can use multiple concatenated layers each one including several filters. The output of each CNN layer is referred to as *feature map* whose depth is equal to the filters' number.

Usually, one or more *fully connected layers* are placed at the end of the network and the *softmax* function is exploited to fulfill the final classification outputing a probabilities vector.

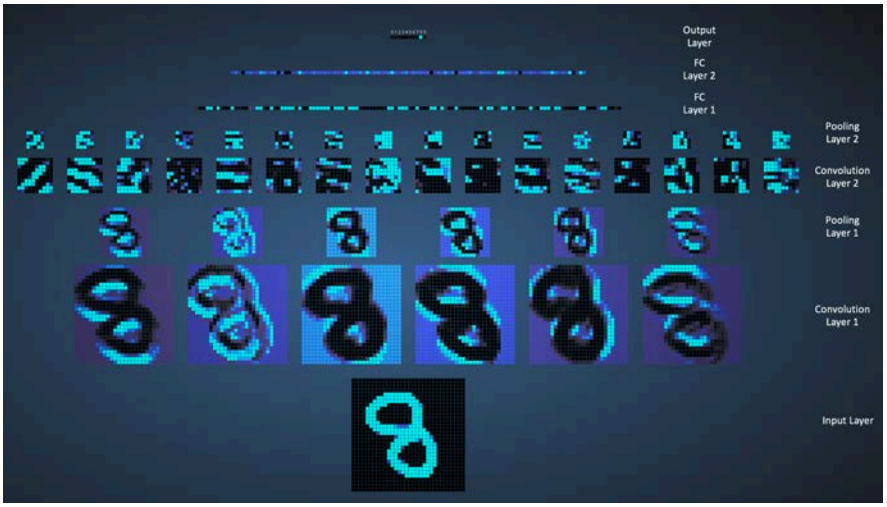


Figure 6.6: 2D visualization of a convolutional neural network trained on the MNIST Database of handwritten digits: from the input layer till the output one [9].

In Figure 6.5 it is shown a sample of a CNN architecture, while, in Figure 6.6 it is reported a visual representation of the weights learned by a CNN trained on the MNIST database of handwritten digits [9]. From this figure, it is evident that the first layers learn the features at higher resolution while, the last ones, work at higher level.

In this section, it is also worth providing some comments on the problem of *overfitting* that is a common issue when working with CNN. Overfitting means that the model fits too well the training set; this implies a poor perfomances when the classifier is applied on new data. Overfitting can be detected by checking the accuracy (or the loss) on validation data and comparing it to the training set one. If the accuracy on the training set is much higher than the validation one, the model clearly suffers from overfitting. Several strategies can be adopted in this case, such as:

- *data augmentation*;
- *add more data*;
- *reduce the architecture complexity*;
- *add regularization*: the most adoted techniques are *dropout* and *L1/L2 regularization* [112].

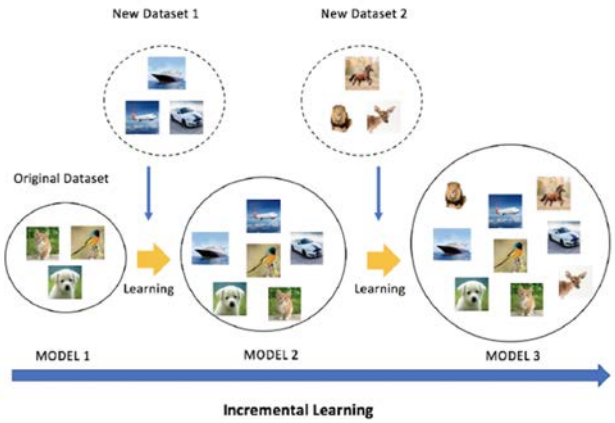


Figure 6.7: The incremental learning framework [10].

All the discussions done until now deal with a static knowledge: the models learn from the available data. Incremental learning framework comes into play when dynamic learning is required.

Incremental learning is the subject of the following section and it is leveraged in this project to conceive a more realistic and intriguing study.

6.3 Incremental learning

Classic machine learning methods are powerful when dealing with static data.

In this scenario it is assumed that:

- data do not change over time;
- all possible classes are known from the beginning and samples of each one are provided in the training set: new classes will not appear in the future.

Obviously, this is not a real scenario where a constantly arriving data stream is present. Further, dynamic learning becomes necessary in interactive scenarios where training examples are provided based on human feedback over time.

Incremental learning refers to learning from streaming data, as shown in Figure 6.7.

More precisely, the incremental learning problem is defined as the capability of machine learning architectures to continuously improve the learned model by feeding new data without loosing previously learned knowledge [113].

Catastrophic forgetting represents one of the main limitations of the incremental framework: a dramatic decrease in the overall performance is perceived when new classes are added incrementally. This is because current neural network architectures require the entire dataset, consisting of all old samples as well as the new classes, to update the model. This requirement is clearly unaffordable in many applications where privacy issues or limited storage budgets are present.

Several strategies have been investigated to tackle this issue. Some studies exploit growing architectures [114], [115].

Some other methods freeze some layers in the learning process [116], [117].

A different way of preserve high performance on old tasks is *knowledge distillation*. It was firstly pioneered by [118], generalized by [119] and then adapted in recent studies [120], [113].

Specifically, knowledge distillation is a model compression method in which a small model is trained to mimic a pre-trained, larger model. For this reason, this training setting is also referred to as *teacher-student* model: the larger model is the *teacher* while the smaller one is the *student*. In distillation, knowledge is transferred from the teacher to the student by minimizing a loss function whose targets are the prediction forecasted by the teacher.

Furthermore, some studies keep a small portion of data belonging to previous tasks to preserve the accuracy on old tasks when dealing with new problems. This *representative memory* is used to perform the training jointly with the new data.

The exemplar set to store can be chosen at random or according to some specific metrics.

In this thesis, incremental learning is designed implementing a knowledge distillation approach in a similar way as done in [113] where knowledge is distilled directly from the last trained model.

Differently from [113], experiments with representative memory are also investigated assuming that the storing of previously seen data is not a limitation.

6.4 Other algorithms

It is worth to mention other two algorithms, without going deeper into technical details, since they are part of the building blocks of this thesis.

6.4. OTHER ALGORITHMS

6.4.1 PCA and Eigendecomposition

Principal component analysis (PCA) allows to perform dimensionality reduction: vectors of size *m* are mapped into vectors of size *k* (with k<m) through a linear transformation. This is useful because high-dimensional data may have nearly all their variation in a small number of dimensions.

The goal of PCA is, therefore, to find a new basis to re-express the dataset in order to filter out noise and reveal interesting structure. The vectors forming the new bases are called *principal components* (hence the name of the algorithm).

Their two main properties are:

- they are othogonal to each other;
- they capture the highest possible variance in decreasing order.

There are several reasons for which PCA technique is often implemented, including:

- *perform lossy data compression:* high dimensional data impose computational challenges;
- *perform feature extraction:* extract relevant information in big and confusing datasets;
- *implement clustering and data visualization:* dimensionality reduction can be used for interpretability of the data and for finding meaningful structure of the data
- *perform noise removal.*

PCA can be done by means of eigendecomposition on the data covariance matrix and it is widely used on the face domain. Eigenface technique was firstly introduced in [121] and the idea behind the algorithm is to represent every face image as a linear combination of some faces (the *eigenfaces*). These eigenfaces serve as a basis for the entire set of faces.

In other words, every image can be represented in a smaller subspace using the best subset of faces that represent the entire set.

To conclude, dimensionality reduction find applications in a wide range of domains such as the eigenalgorithm investigation shown in [28] from which this project is inspired.

6.4.2 Decision trees

Decision trees are one of the most popular and simple algorithms used in machine learning.

Our brain works like a decision tree every time we ask ourselves a question: we split the question in multiple subquestions taking a decision from time to time until reaching the final one.

A decision tree predicts the label associated with an instance x by traveling from a root node of the tree till one of its leaves. Every leaf contains a specific label. At each node on the root-to-leaf path, the successor child is chosen according to a specific splitting of the input space. Usually, the splitting is based on one of the features of x or on a chosen set of splitting rules. A popular splitting rule at internal nodes of the tree is based on thresholding the value of a single feature as it is done in this thesis [107].

7 Implementation

In this Chapter, all the implementation details are described introducing the developed project and the processing pipelines first.

7.1 Project presentation

In this project, we want to identify people, in virtual and real environments, from their gait intercepting encrypted video surveillance traffic and exploiting deep learning techniques.

The peculiarity of this project relies on the processed data: instead of examining images, as usually done in the literature, we investigate packet size patterns from encrypted video streams proposing a very simple approach.

Figure 7.1 shows an high level description of the developed project.

The first phase involves the generation of synthetic/real videos. At this scope, in the virtual case, we create some simple game-like virtual walking characters with the Unity tool. In the real scenario, instead, we film different walking people in a common testing place. Exploiting the camera attached to virtual/real scenes, we capture images and encode them through the H.264 compression standard. These videos are loaded into a server and transmitted to a second host, the client, when ever the latter asks to access to one of these streams.

Once established the connection, a third actor, the sniffer, intercepts transmitted data, retrieves packet size information, processes them and enacts a classifier that estimates the identity of the walking person.

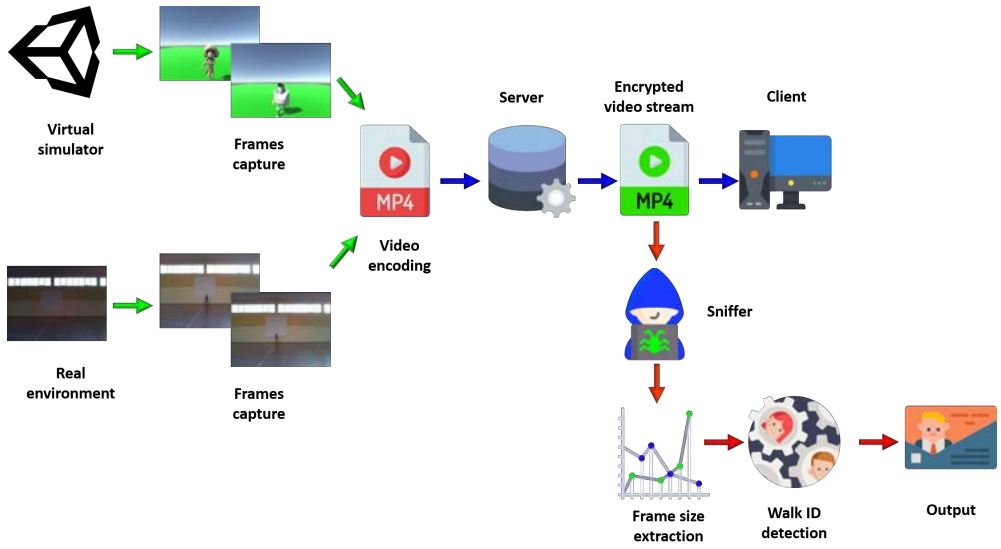


Figure 7.1: Scenario developed with this project.

The designed identification system is a bit more sophisticated than a standard neural network implementation. Instead of simply training a network to classify the whole set of people, we take inspiration from [28], where an eigenalgorithm strategy is implemented to perform device identification, and we propose an eigenwalk technique that ideally aims at recognizing the walk patterns of all the people from a subset of them only.

Additionally, we extend the identification system with an incremental framework that continually learn new data. We develop this last procedure to make the system suitable to real applications where data are progressively available over time.

Actually, the whole work is splitted into 2 sub works. The first, shown in Figure 7.2, is the actual task I developed. It does not include the encryption mechanism: I implemented all the phases from the dataset generation till the classification output. The second, is the one developed by Samuele Piazzetta. He is in charge of designing a tool that simulate secure real time streaming in order to prove the robustness under encryption mechanisms. Figure 7.3 shows the virtual/real scenario reconstructed by Samuele: a sniffer intercepts transmitted data and retrieves packet size information.

The whole work is basically splitted into 4 processing pipelines that will be described in the following sections.

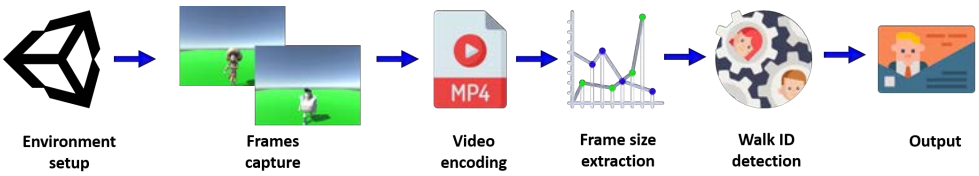


Figure 7.2: High level description of the steps implemented in the first task.

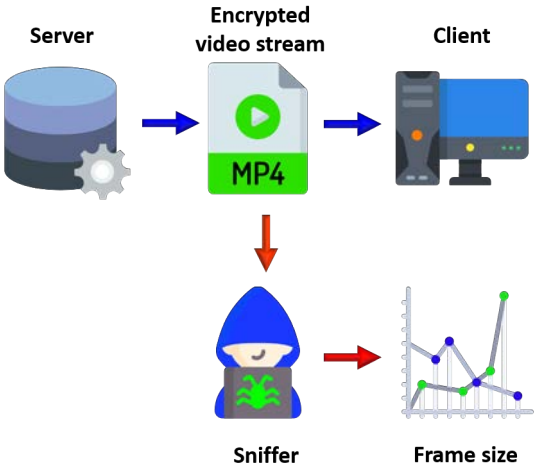


Figure 7.3: Virtual/real scenario reconstructed by Samuele: the sniffer intercepts the transmitted packets and retrieve packet size information.

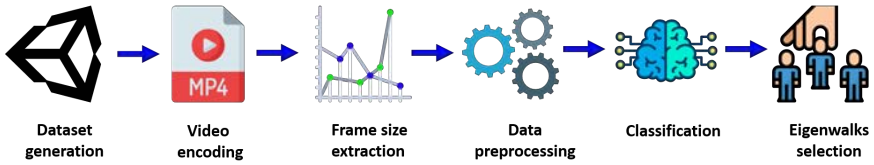


Figure 7.4: Processing pipeline for eigenwalks selection.

7.2 Processing Pipelines

In this work, we assume that the forensic analyst does not perfectly know all the walking classes but only a subset of them.

In particular:

- the *closed and known set* comprises 3 walking types. From it, 2 of them are selected as eigenwalks;
- the *open and unknown set* consists on 1 walking type.

The 4 processing pipelines implemented, both in the virtual and real environments, are the follow :

- *the first* is designed to perform the eigenwalks selection;
- *the second* enacts the classification;
- *the third* implements the incremental framework.
- *the last* simulate the secure transmission and the corresponding packet sizes reconstruction.

As shown in Figure 7.4, the first processing entails:

1. *Dataset generation*: virtual and real frames are generated. Virtual walks are simulated with the Unity tool; real walks are acquired with a Microsoft®LifeCam HD-3000 camera. In this phase, all the walks that we want to classify are generated: both the ones belonging to the closed set and the ones of the open set;
2. *Video encoding*: the frames generated in the previous step are encoded with H.264 standard exploiting the *FFmpeg* tool;
3. *Frame size extraction*: during the encoding process, *FFprobe* tool is leveraged to retrieve packet sizes;

4. *Preprocessing*: in this phase, intra frames are removed, noise reduction is performed and dataset splitting and segmentation are enacted;

5. *Classification*: a convolutional neural network is designed to classify all the walking types belonging to the closed set;

6. *Eigenwalks selection*: eigenwalks are selected from the whole closed and known set according to the classification outputs of the previous step. In particular, the walks maximizing the respective distance are selected following the same idea behind PCA.

The second pipeline is implemented with a *thresholds-based* decision tree algorithm where:

1. *in the 1^{st} level*: the eigenwalk closest to the walk to be classified is selected;

2. *in the 2^{nd} level*: the final class prediction is fulfilled discerning a true eigenwalk from the one similar to it.

The incremental procedure includes:

- the training of the updated model starting from the previous one: the new model includes an additional class;

- the reclassification of all the walks with an additional eigenwalk available.

To conclude, the encryption mechanisms are implemented with the tool developed by Samuele.

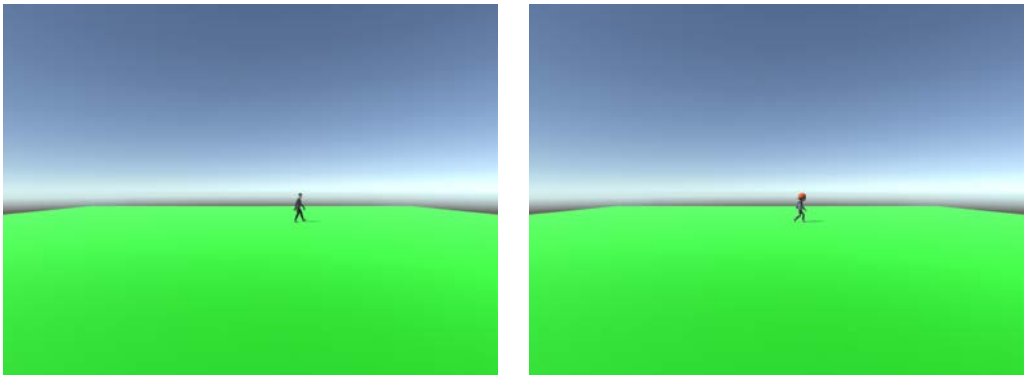
The next sections are organized as follow:

- the first and the second pipelines are described introducing the virtual environment first and, successively, the real one;

- the incremental learning procedure is described in Section 7.5;

- the tool designed by Samuele Piazzetta to perform video encryption is reported in Section 7.6;

- at the end of the Chapter, the chosen perfomance metric is discussed.



(a) Passive character. (b) Aj character.

Figure 7.5: Two samples of virtual frames generated by Unity.

7.3 Virtual environment

7.3.1 Dataset Generation

Synthetic video streams are generated simulating virtual walking characters exploiting the Unity tool.

To this end, a virtual environment is created consisting on:

- a 50 × 50 square with uniform texture;
- a camera placed in one side, 2 meters off the ground.

The virtual environment can be seen in Figure 7.5.

The walking characters are taken from Mixamo [42] and the animations are driven by an Animator Controller and a Script attached to them. Moreover, a further script connected to the camera is used to generate video frames, corresponding to the designed game, at 30 fps.

More precisely:

- 4 Mixamo characters walking at constant speed in the rightward direction are used;
- 5 Animation Clips are considered in the Animator Controller: the first corresponds to the idle state while the last are walking/running styles. These 5 animations are shown in Figure 7.6.
- 384 videos are simulated with different configurations of walking style, speed and starting position.

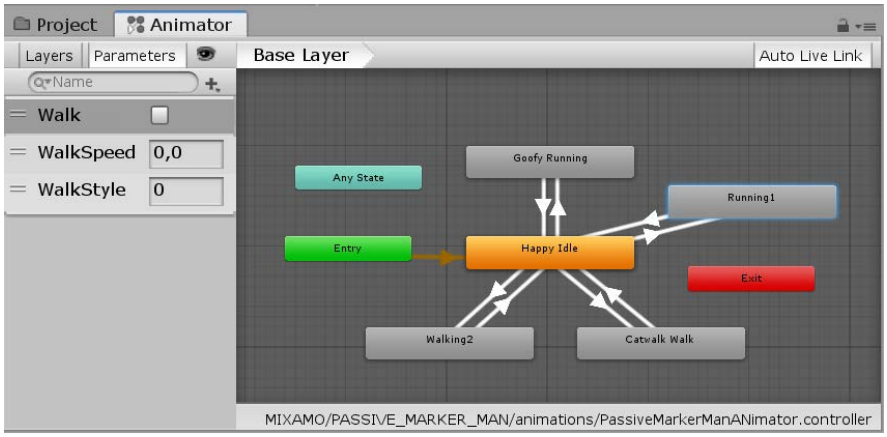


Figure 7.6: Animator Controller used to simulate the virtual walk. 5 Animation Clips are present: the 1^{st} represents the idle state, the 2^{nd} and the 3^{rd} are walking types, the last two are running types.

The configurations considered are the follow (24 realizations per walking/running style):

- 2 walking styles and 2 running styles.
- walking speed = 1;1.2;1.4 m/s;
- running speed = 2.2;2.4;2.6 m/s;
- starting from center or random position. The random position is in the range -1.5–1.5 from the starting point.

7.3.2 Video encoding and Frame size extraction

Video encoding is implemented exploiting the *FFmpeg* library.

More precisely, the frames generated with Unity tool are saved in i folders, named $iter_i$ (with i in the range 1–384), and then encoded both in VBR and CBR mode using a Matlab script.

In the VBR case, the following FFmpeg parameters are set:

- *-vcodec libx264:* uses the x264 library to encode video streams with the H.264 format;
- *-r 30:* sets the input frame rate to 30 fps;
- *-i folder/frame%4d.png:* indicates the path of the input files;
- *-r 30:* sets the output frame rate to 30 fps;

- *-pix_fmt yuv420p:* defines the pixel format. In this case, the *YUV* color space is used with 4:2:0 sampling format. This means that *Cr* (red chroma component) and *Cb* (blue chroma component) each have half the horizontal and vertical resolution of Y (luma component). The numbers are rather confusing since they do not actually have a logical interpretation but the term has solely historical reasons. 4:2:0 sampling is widely used for consumer applications such as video conferencing or digital television [77];
- *-g 30:* set the GOP size to 30;
- *-qmin 2:* set the lowest quality to 2. Note that, for libx264 the quality range is 0–51 with 0 corresponding to the best quality reachable while 51 to the lowest. The default quality is equal to 23 [122];
- *-qmax 2:* set the highest quality to 2. This combination of qmin and qmax coincides with imposing constant quality;
- *-coder ac:* sets the arithmetic entropy encoder. This is actually the default one, also referred to as CABAC (Context-Adaptive Binary Arithmetic Coding);
- *-y:* overwrites output files without asking.

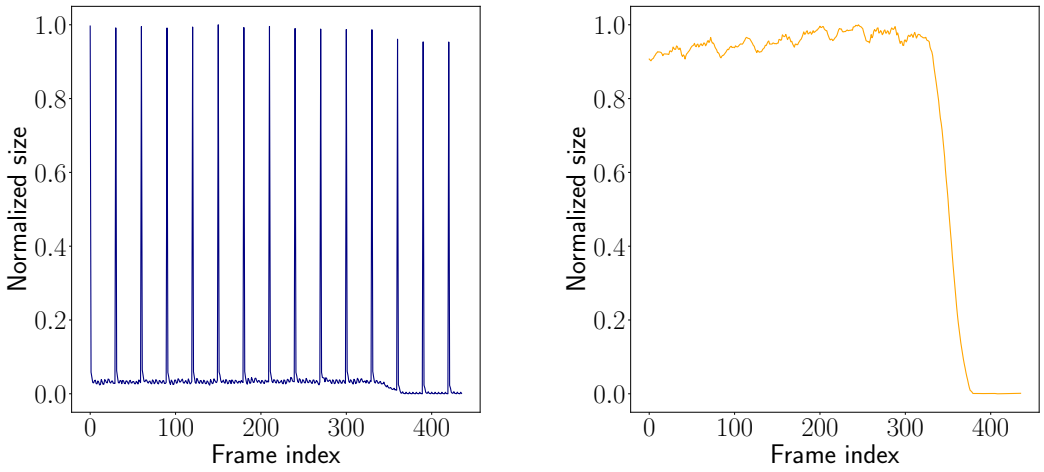
In the CBR mode, instead of setting the quality range, the bitrate is defined:

- *-x264 -params nal-hrd=cbr:* forces the CBR mode;
- *-b:v 2M:* specifies the target (average) bit rate;
- *-maxrate 2M:* specifies the maximum tolerance;
- *-bufsize 2M:* specifies the buffer rate control. In other words, it tells the encoder how often to calculate the average bit rate and check to see if it conforms to the average bit rate specified by the b:v command. Whence, this parameter determines the output bitrate variability.

For each iter, the script outputs:

1. *the mp4 file;*
2. *the pxt.txt file:* this file is obtained exploiting the FFprobe stream analyzer [123] and it contains informations regarding the coding process including the size for each frame.

The whole analysis developed in this project starts from these *pxt.txt* files, from which all size informations can be retrieved.



(a) Signal before noise reduction.

(b) Signal after noise reduction.

Figure 7.7: Signals comparison: before and after intra frame removal and noise reduction.

7.3.3 Preprocessing

The preprocessing step involves:

1. *intra frame removal and noise reduction:* they are both implemented with a moving average filter with length equal to the GOP size (i.e equal to 30). With this approach, each sample is replaced with the average of the 30 nearest samples. This type of filter is widely used to smooth noisy data while preserving the shapes. Note that I frames are filtered out since they do not provide extra information and they are only responsible for the high periodic spikes: only P and B frames are useful for gait analysis purposes;

2. *useful content selection:* the extreme frames of each signal are removed since they correspond to instants where no walking or running activities are present;

3. *dataset splitting:* the whole dataset is splitted into: 70 % training, 20 % validation and 10 % test sets.

4. *normalization:* a simple per-iter *min-max* normalization is implemented:

$$iter_i(x) = \frac{x - min_i}{max_i - min_i}. \tag{7.1}$$

where x is any value of the *i-th* iter, while min_i and max_i are, respectively, its minimum and maximum values. Note that feature scaling implementations (i.e normalization or

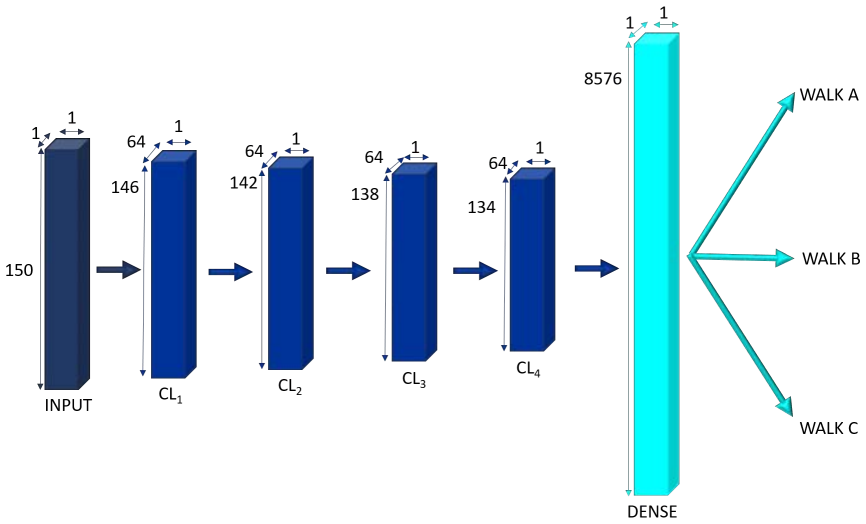


Figure 7.8: CNN architecture used in this thesis.

standardization) are quite standard when dealing with machine learning algorithms. Feature scaling helps in improving the stability of the algorithms changing the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values;

5. *segmentation:* every signal is segmented with a sliding window approach setting the window length to 150 and the sliding stride to 7;

6. *labelling:* every segmented signal is labelled with one of the four corresponding labels: A, B, C or D.

Figure 7.7 shows the difference of a virtual signal before and after noise reduction in the VBR case.

7.3.4 Eigenwalk selection

A former classification is designed with the purpose of investigating the classes at maximum average distance among each other. These classes will be identified as eigenwalks and will be used to perform the final clustering.

In this phase, only the first two sets of the closed set are used. In particular:

- *training set*: it is used to train the network;
- *validation set*: it is used to stop the training through early stopping technique and to select the two eigenwalks by estimating the network predictions.

As shown in Figure 7.8, the network architecture consists on:

- *CL_1, CL_2, CL_3 and CL_4:* the first four layers are convolutional ones. Each convolutional kernel performs a 1D convolution over the time dimension. Each convolutional layer has: 64 kernels and filter size equal to 5. These layers employ ReLU activation function. After each convolutional layer a batch normalization layer is used to increase the stability of the network [124];
- *Dense:* the last layer is the dense one. It yields the classification outcome through the softmax activation function.

The training options are:

- *optimizer:* the Adam optimizer is used with the default parameters and learning rate equal to 0.00001;
- *batch size:* 128. This parameter sets the batch size to use for each training iteration. A batch is a subset of the training set that is used to evaluate the gradient of the loss function and update the weights;
- *max epochs:* 500. This parameter sets the maximum number of epochs to perform the training. An epoch is a full pass through the entire data set;
- *loss:* the categorical cross entropy loss is used. The cross entropy loss is widely used when dealing with classification tasks;
- *early stopping:* it is implemented to perform an automatic stop of the training by checking the validation loss. The patience is set equal to 50. This means that 50 is the number of epochs that the loss on the validation set can be larger than or equal to the previously smallest loss before network training stops.

To perform the eigenwalks selection step, the following notion of distance is defined:

$$d_c(n) = 1 - y_{pred}[n, c] \tag{7.2}$$

where:

- *n*: is the element;
- *c*: is the class;

- $y_{pred}[n,:]$: is the network output for the n element. In other words, it is a 1×2 vector of probabilities where element c represents the probability for element n to belong to class c;
- $d_c(n)$: is the distance for element n from class c.

The validation set is used to estimate the network predictions. From these outputs, a distance matrix DM is built where:

- *row i*: corresponds to true class i;
- *column c*: corresponds to the average distance from class c.

Whence, the element $DM(i,c)$ reports the average distance of true class i from class c according to the network predictions.

The two eigenwalks selected e_1 and e_2 are such that:

$$DM(e_1, e_2) + DM(e_2, e_1) = max_{i \neq j}(DM(i, j) + DM(j, i)) \tag{7.3}$$

For the rest of the work:

$$d^{max}_{c_1,c_2} = DM(e_1, e_2) + DM(e_2, e_1) \tag{7.4}$$

i.e. $d^{max}_{c_1,c_2}$ is the maximum average distance between two classes c_1,c_2.

Note that the two eigenwalks resulting from this procedure are not perfectly orthogonal to each other, as in standard PCA algorithm, but they satisfy the second property mentioned in 6.4.1: they aim to capture the highest possible variance.

7.3.5 Final classification

The ultimate classification is enacted as follow:

- a CNN is trained to recognize the two selected eigewalks. The architecture and the training options are the same mentioned in Section 7.3.5;
- a decision tree DT algorithm is designed to perform the final clustering. The algorithm works on a global classification averaging the output predictions over the whole video.

As shown in Figure 7.9, in the 1^{st} level any walk (i.e. eigenwalk or not) is associated to the most similar eigenwalk; in the 2^{nd} level the walk is labelled according to a threshold metric. These thresholds will be discussed in Chapter 8.

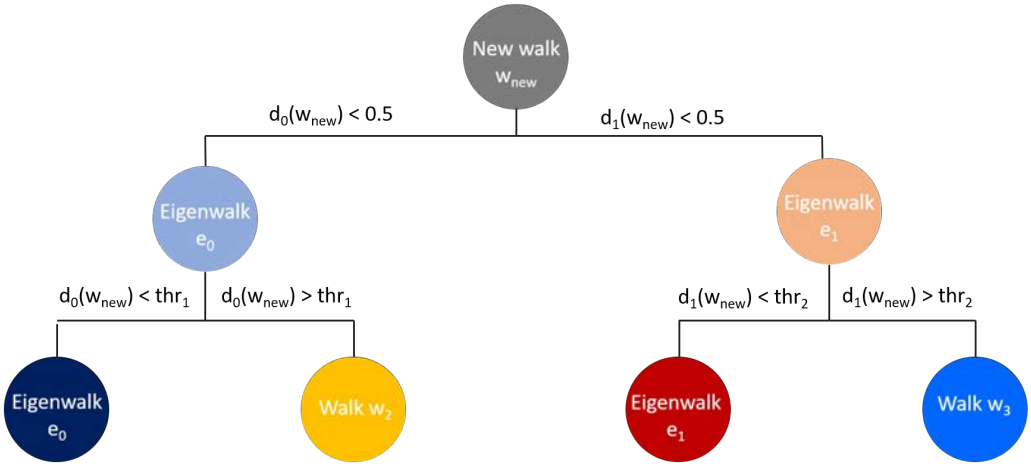


Figure 7.9: Decision tree algorithm implemented to classify all the 4 virtual walks.

7.4 Real environment

In the real environment, the same processing steps are implemented with some small changes that will be highlighted. These are mainly due to the class imbalance and the small dataset.

7.4.1 Dataset generation

Real video streams are generated with Microsoft®LifeCam HD-3000 and FFmpeg library.

The camera is used to film people walking sideward while, FFmpeg is used to create video frames equivalent to those generated by the Unity tool.

Specifically, experiments were conducted recording 4 volounteers for the following number of times:

- 59 iterations for Volunteer 1 (V_1);
- 49 iterations for Volunteer 2 (V_2);
- 26 iterations for Volunteer 3 (V_3);
- 10 iterations for Volunteer 4 (V_4);

The environmetal setup is shown in Figure 7.10: it is a Palazzetto located in Ronchi di Casalserugo. The shooting took place one afternoon with natural lighting conditions. The following FFmpeg parameters are used in a matlab script to produce the needed frames:

- *-f dshow*: it forces FFmpeg to acquire screens from a Windows device;

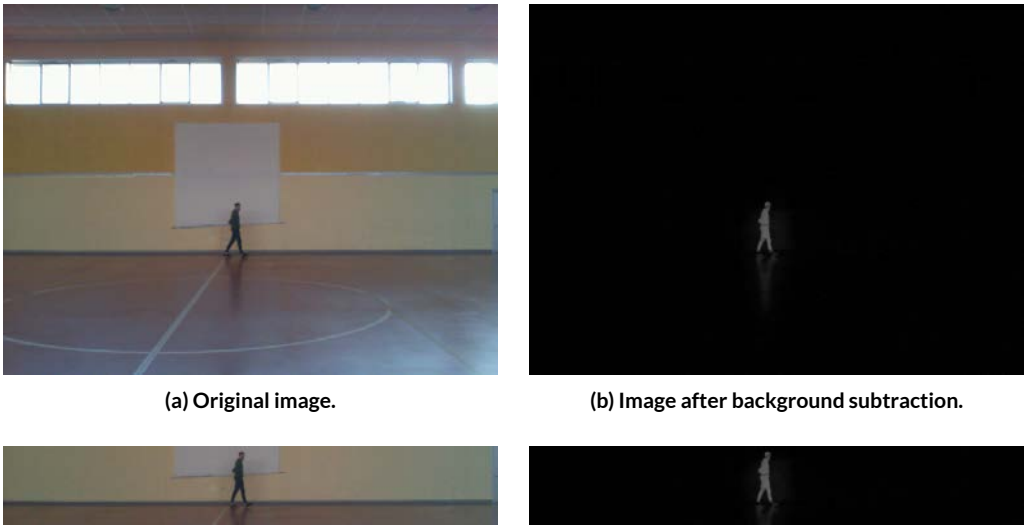
Figure 7.10: Palazzetto located in Ronchi di Casalserugo where regordings for real investigations took place.

- *-r 30*: it sets the input acquisition frame rate;
- *-i video = "Microsoft LifeCam HD-3000"*: it indicate the name of the acquiring device;
- *-r 30*: it sets the output acquisition frame rate
- *-vframes*: it sets the number of frames to be acquired.

Analisys on the real environment are carried out not only on the original noisy videos but also on black and white (B&W) ones in order to derive some useful comparisons. This second set of videos is created by subtracting from each acquired image the background image shown in Figure 7.10.

More specifically, 4 datasets are considered:

1. *original dataset:* videos are generated starting from original images;
2. *cropped original dataset:* videos are generated starting from cropped original images. These are constructed from original images selecting the middle portion corresponding to the walking man;
3. *B&W dataset:* videos are generated starting from B&W images. These are created by subtracting from each original image the background one;



(a) Original image.

(b) Image after background subtraction.

(c) Cropped original image.

(d) Cropped image after background subtraction.

Figure 7.11: Volunteer 3 walking during recordings taken at Palazzetto di Ronchi di Casalserugo.

4. *cropped B&W dataset:* videos are generated starting from cropped B&W images. These are created by cropping B&W images.

The two cropped versions are created with the following FFmpeg parameter:

-filter:v crop=640:100:0:250

Samples of the 4 datasets can be seen in Figures 7.11 and 7.12.

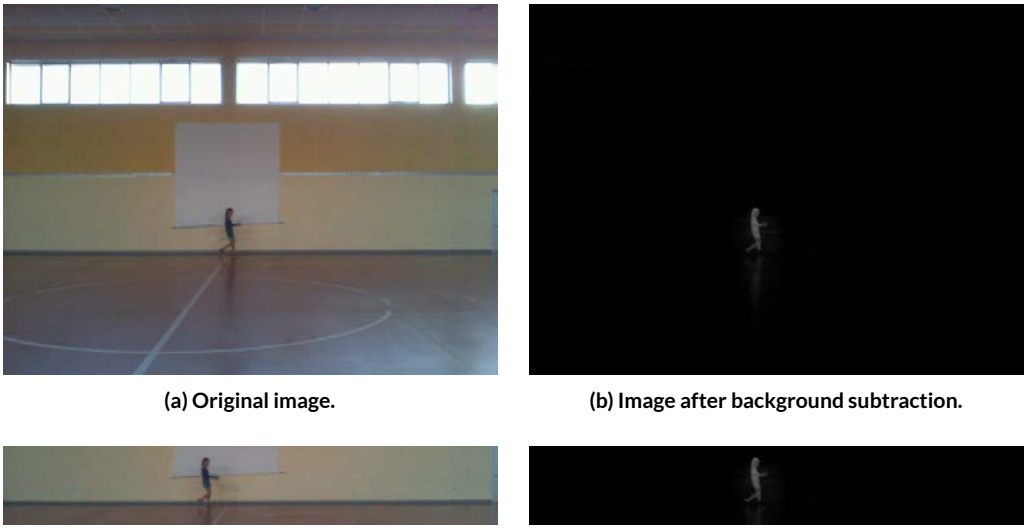
Note that all the four aforementioned datasets correspond to four different strategies to implement background subtraction and noise reduction. Additionally, these are all methods that a true forensic analyst could easily implement.

The only difference between these dataset relies on how images are processed before being encoded: the rest of the processing pipelines are identical.

7.4.2 Video encoding and Preprocessing

To encode real videos, the same procedure described in Section 7.3.2 is implemented. The only difference relies on the VBR case where the quality is set to 19 in order to keep the avergae bitrate equal to 2 Mbps.

The preprocessing steps are identical to those implemented in 7.3.3 with the following exceptions:



(a) Original image.

(b) Image after background subtraction.

(c) Cropped original image.

(d) Cropped image after background subtraction.

Figure 7.12: Volunteer 2 walking during recordings taken at Palazzetto di Ronchi di Casalserugo.

1. for the segmentation phase, the sliding stride is set equal to 1 to increase the number of segments;

2. for the labelling step, the labels correspond to the first letter of the volounteers' names: V_1, V_2, V_3 and V_4.

7.4.3 Classification and eigenwalks selection

The two eigenwalks are chosen in advance selecting the two volounteers with the highest number of acquisitions (V_1 and V_2). This procedure is almost equivalent to the virtual eigenwalk selection: these two real eigenwalks are among those whose pairwise distance is maximum, as will be shown in Chapter 8.

The network architecture and the training options are the same described in Section 7.3.4

7.5 Incremental learning

The incremental learning procedure starts from the model M^{-1} trained in Section 7.3.5. The third class belonging to the closed and known set is added to it training the new model M^0.

The M^0 architecture is identical to that of the M^{-1} model. The only difference relies on the loss to be minimized:

- M^{-1} minimizes the cross entropy loss $\mathcal{L}_{CE}$, which learns how label the classes;
- M^0 minimizes a linear combination $\mathcal{L}$ of two losses: the cross entropy loss $\mathcal{L}_{CE}$ and the distillation loss $\mathcal{L}_D$, which helps to retain the knowledge of previously seen classes;

More specifically, the total loss $\mathcal{L}$ is defined as:

$$\mathcal{L} = \mathcal{L}_{CE} + \lambda_D \mathcal{L}_D \tag{7.5}$$

where the cross entropy loss is:

$$\mathcal{L}_{CE} = -\frac{1}{|\mathcal{D}_0^{tr}|} \sum_{X_n \in \mathcal{D}_0^{tr}} \sum_{c=1,2,3} Y_n[c] \cdot \log(M^0(X_n[c]) \tag{7.6}$$

and the distillation loss is:

$$\mathcal{L}_D = -\frac{1}{|\mathcal{D}_0^{tr}|} \sum_{X_n \in \mathcal{D}_0^{tr}} \sum_{c=1,2} M^{-1}(X_n[c]) \log(M^0(X_n[c]) \tag{7.7}$$

In the formulas:

- $\mathcal{D}_0^{tr}$: is the training set available at the incremental step;
- X_n: is the sample n belonging to $\mathcal{D}_0^{tr}$;
- Y_n: is the one-hot encoded ground truth corresponding to X_n. Each Y_n is a binary vector with dimension 1×3 where each element is either 0 or 1 (where 1 indicates the true class). Therefore, in $Y_n[c]$, c indicates the class;
- $M^0(X_n)$: is the M^0 prediction for the sample X_n; as before, in $M^0(X_n[c])$, c indicates the class;
- $M^{-1}(X_n)$: is the M^{-1} prediction for the sample X_n.

The distillation loss has a regularizing effect controlled by the λ_D parameter. In Chapter 8, the outcomes corresponding to $\lambda_D = 0$, $\lambda_D = 1$ and $\lambda_D = 10$ will be reported.

From a pratical point of view the combination of the two losses is implemented building the same model described in Section 7.3.4 with two outputs and two losses.

Specifically:

- *the first output targets:* is the ground truth;
- *the second output targets:* is the output of M^{-1} for all the samples belonging to $\mathcal{D}_0^{tr}$. In other words, the predictions of the previous model for samples in $\mathcal{D}_0^{tr}$ are used as new ground truth ;
- both the loss are the categorical cross entropy loss.
- the two losses are balanced setting the *loss_weights* training option. This parameter is a vector 1×2 whose values corresponds to the loss contributions. The first element is kept equal to 1, while, the second corresponds to the λ_D value.

To conclude, we compare the scenario where no previous training samples are available and the scenario where a small random portion of previous training set $\mathcal{D}_{-1}^{tr}$ can be stored.

The incremental procedure is implemented comparing the following cases:

- $M_{0\%}^0$: none of the previous samples are used;
- $M_{10\%}^0$: 10% of the previous training samples are exploited;
- $M_{25\%}^0$: 25% of the previous training samples are employed;
- $M_{25\%}^0$: 50% of the previous training samples are used;

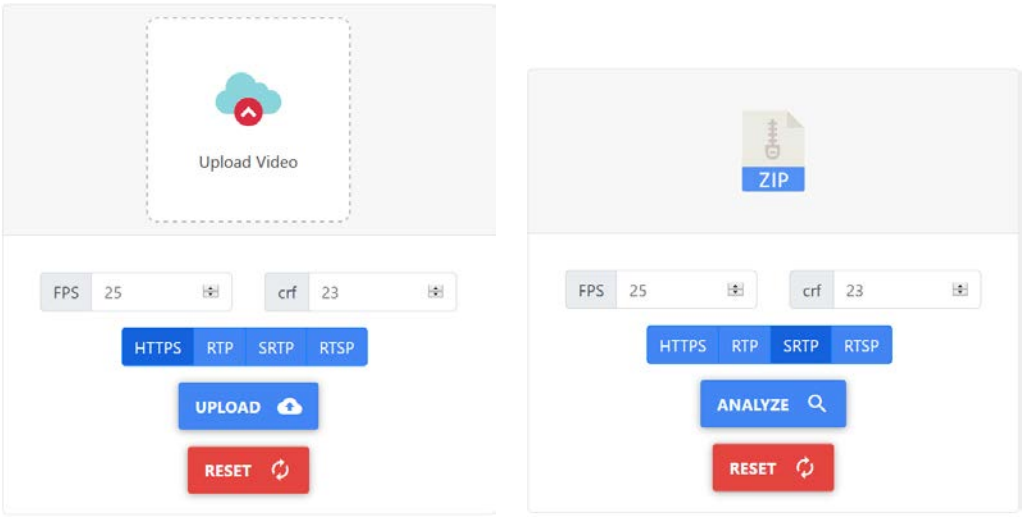
7.6 Encrypted video

Samuele Piazzetta, after investigating the protocols behind protected transmissions, designed a tool that:

- simulate real time streaming of encrypted videos;
- capture transmitted packets;
- retrieve packets size of sniffed videos.

The tool consists on a client-server architecture that can be summarized as follows:

- *the client* includes the interface that allows the upload of a set of videos to the server and, after selecting the desired protocol, requests the real time streaming and receives the transmitted stream. At the end, the client can ask for a report of the reconstructed packets size of transmitted videos and save a local copy;



(a) Starting tool interface.

(b) Ending tool interface.

Figure 7.13: Video encryption tool designed by Samuele: client side.

- *the server* implements two components: the *sniffer* and the *extractor*. The first sniffs the traffic exchanged from the server to the client exploiting the *Wireshark* tool [125], the second, after the sniffing session, analyzes packets and reconstructs the frames size.

Figure 7.13 shows the designed client interface.

Note that the tool also allows to set the Constant Rate Factor parameter (CRF) to establish the quality of the encoded video compared to the original one.

7.7 Performance measures

It is worth to spent some words on the metric used to assess the performances on the entire investigation since there are several ways to evaluate it. The choice of an appropriate measure is not trivial as they may reflect some specific qualities of the system while hiding or misrepresenting others. This is particularly important when dealing with real-life data where labels used as ground truth might be loosely defined or ambiguous. Furthermore, the available dataset may be highly unbalanced with one of the classes being overrepresented with respect to the others.

The metric chosen is the weighted F_1 score (also called *F-score* or *F-measure*). The F-score is defined as the harmonic average of the *precision* and *recall*.

Table 7.1: Confusion matrix for binary classification. TP, FP, TN, FN are all intended for class A.

		Predicted Class	
		A	B
Actual Class	A	TP	FN
	B	FP	TN

Mathematically speaking, precision P and recall R are the following:

$$\begin{aligned} P &= \frac{TP}{TP + FP} \\ R &= \frac{TP}{TP + FN} \end{aligned} \tag{7.8}$$

where TP stands for true positives, FP stands for false positives and FN for false negatives. A visual representation of this terminology can be found in Table 7.1.

The weighted F_1 *measure* counts for the class imbalance and is defined as:

$$F_1 = \sum_i 2 \times w_i \frac{P_i \times R_i}{P_i + R_i}. \tag{7.9}$$

where i is the class index and $w_i = \frac{n_i}{N}$ is the proportion of samples of class i, with n_i being the number of samples of the *i-th* class and N being the total number of samples.

This metric is more suitable than accuracy one:

$$Accuracy = \frac{(TP + TN)}{(TP + FP + TN + FN)} \tag{7.10}$$

since it is highly affected by the sample distribution across classes [126], [127].

8 Results

8.1 Virtual environment

8.1.1 Dataset comparison

In Figure 8.1 samples of signals resulting from VBR and CBR encoding are reported. It can be seen that the walks have a distinguishable shapes in the VBR case while, in the CBR scenario, all the shapes are more similar each other presenting a constant trend with small peaks up and down.

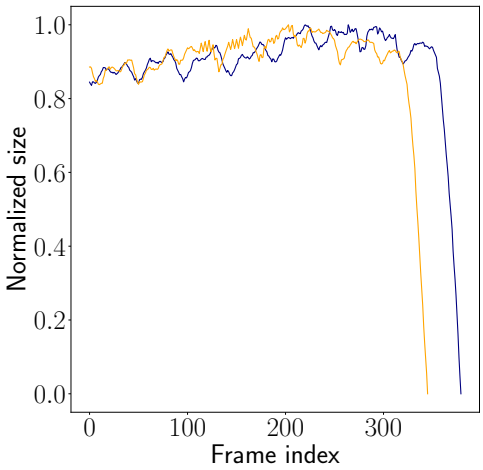
8.1.2 Eigenwalks selection

Figure 8.2 shows the confusion and distance matrices resulting from the first classification of the 3 walks belonging to the known set in the VBR case. To be fair, these are computed on the validation set rather than on the test one.

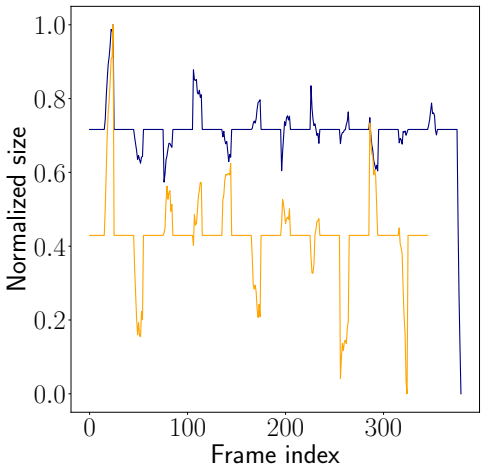
In Figure 8.3 it is reported the equivalent in the CBR scenario.

Note that, an evident gap between the two configurations are present: VBR encoding allows to achieve higher accuracy rather than CBR one. This could be expected from the comparison made in the previous section.

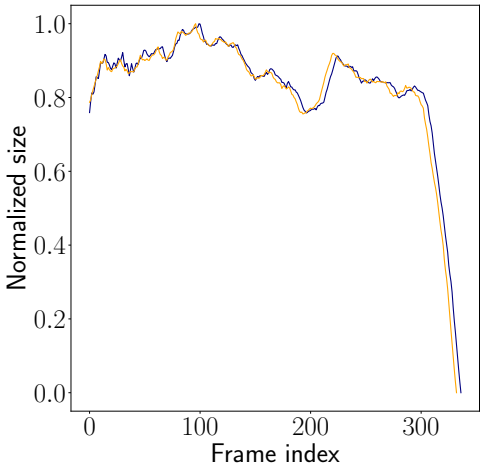
Nevertheless, the resulting two eigenwalks that maximize the average distance among each other are the last two in both the cases: their average distance is equal to 0.99 in the VBR mode, while it is 0.85 in the CBR case.



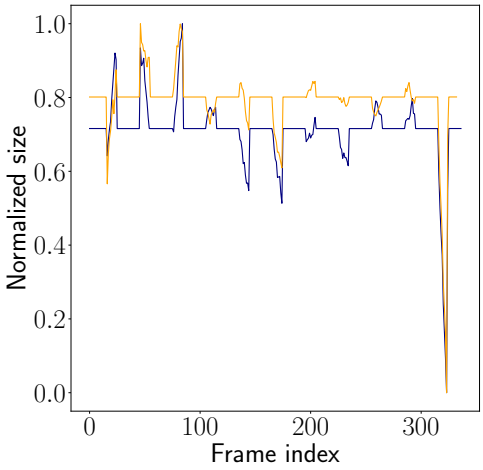
(a) Two samples of walk A, VBR.



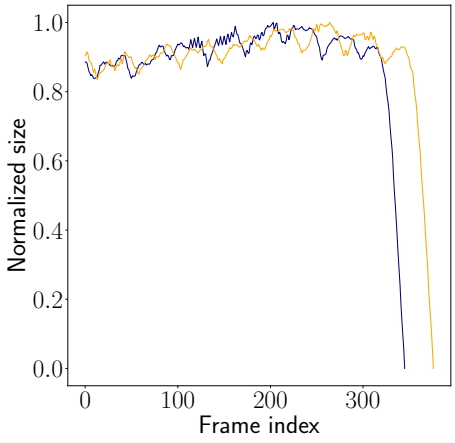
(b) Two samples of walk A, CBR.



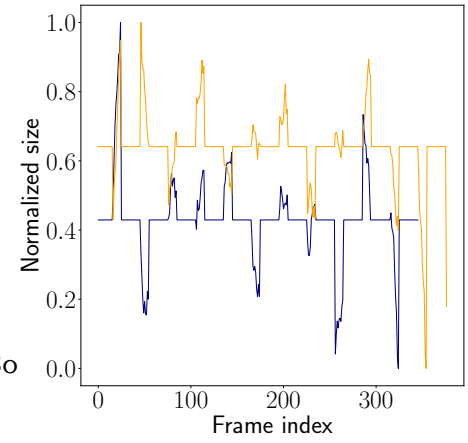
(c) Two samples of walk B, VBR.



(d) Two samples of walk B, CBR.

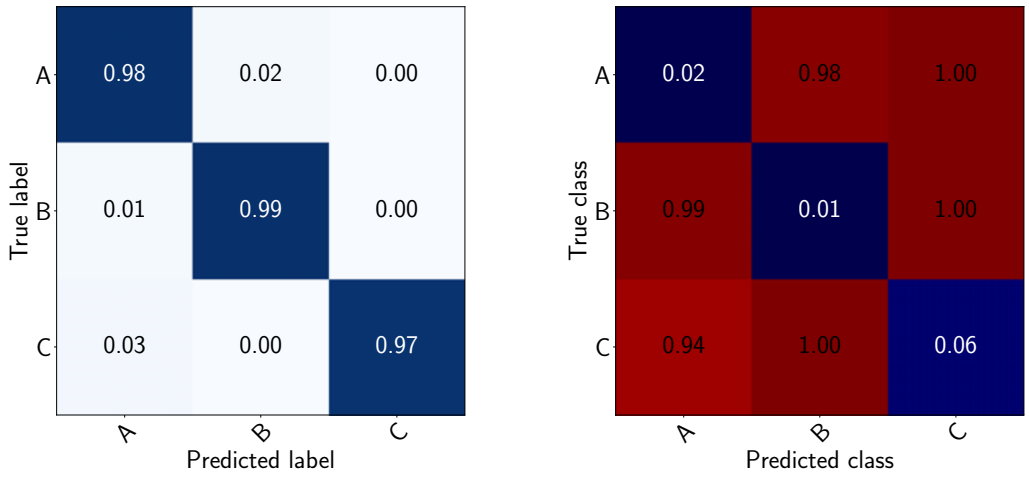


(e) Two samples of walk C, VBR.



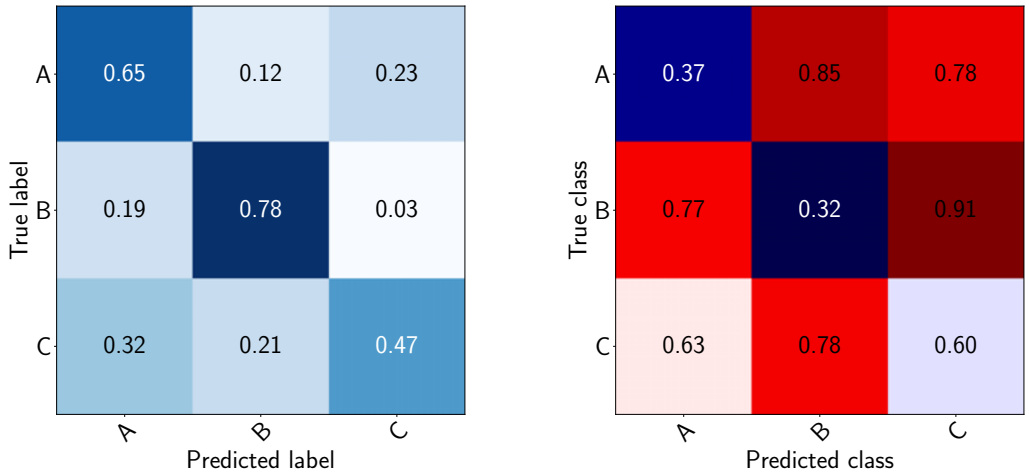
(f) Two samples of walk C, CBR.

Figure 8.1: Frame size series for the 3 walks types belonging to the known set: comparison between VBR and CBR signals.



(a) Confusion matrix. **(b) Distance matrix *DM*.**

Figure 8.2: Outcomes resulting from the first classification for the eignewalks selection in the VBR case. The results are computed on the validation set.



(a) Confusion matrix. **(b) Distance matrix *DM*.**

Figure 8.3: Outcomes resulting from the first classification for the eigenwalks selection in the CBR case. The results are computed on the validation set.

Table 8.1: F_1 scores for walk A, B, C and D after applying the *DT* algorithm in the test set.

			Walk A	Walk B	Walk C	Walk D	Weighted F_1 score
F_1 score	VBR	1^{st} level	0.95	1	1	0.95	0.97
		2^{nd} level	0.89	1	0.94	0.89	0.86
	CBR	1^{st} level	0.78	0.82	0.8	0.8	0.8

Therefore, walks B and C are the two selected eigenwalks.

8.1.3 VBR vs CBR

Figures 8.4 and 8.5 show the distances for the 4 walks from the two selected eigenwalks in the VBR and CBR modes respectively. The outcomes are the results of the second CNN training described in 7.3.5. In this case, the results are computed on the test set.

Note that, the 4 clusters are more distinguishable in the VBR mode. Indeed, Figures 8.4c and 8.4d report the thresholds applied in the 2^{nd} level of the DT algorithm, while, in Figures 8.5c and 8.5d no thresholds are shown since no value led to achieve satisfying results.

Thresholds thr_1 and thr_2, in the VBR case, are both set equal to 0.001 after observing the average distances on the validation set of e_1 and e_2 from themselves (they are both equal to 0.00).

Table 8.1 quantifies the F_1 scores resulting from the DT algorithms applied in the two case. For the CBR encoding, only the scores for the 1^{st} level are shown.

It is evident that the VBR scenario achieves the highest accuracy, as expected. The results in the CBR case are passing only in the 1^{st} level.

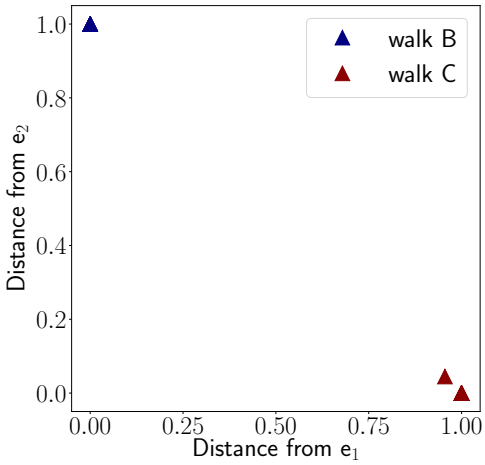
8.1.4 Incremental learning

Tables 8.2 and 8.3 exhibit the F_1 scores resulting from the incremental learning implementation. For each representative set, the best results are highlighted: the best and the worst cases are shown in blue and red color, the others in green.

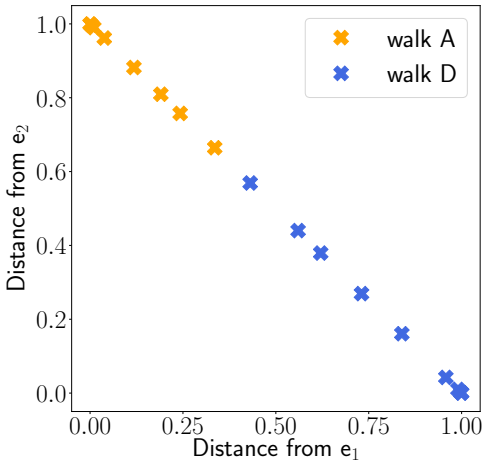
For the sake of comparison, also the case $\mathcal{M}^{-1}_{A\text{-}C}$ in which all the 3 classes are learned at once are reported.

From these tables, the following can be inferred:

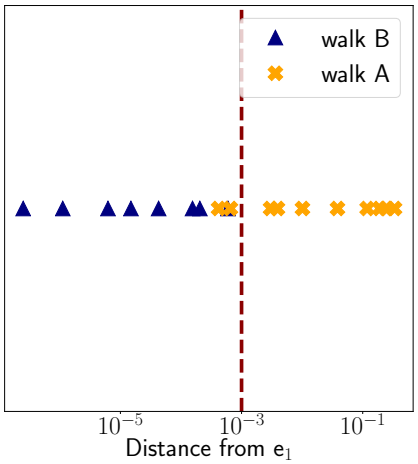
- the quality reachable by the VBR encoding is confirmed;
- in the CBR case all the scores are below 55 %;



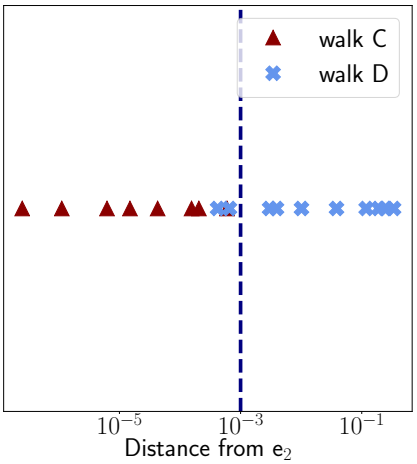
(a) Distance for B and C from e_1 and e_2.



(b) Distance for A and D from e_1 and e_2.

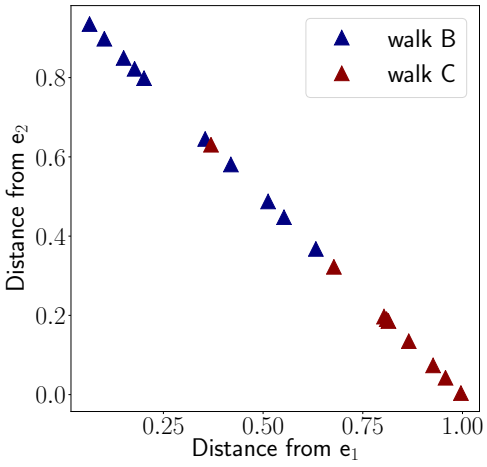


(c) Distance for B and A from e_1, logaritmic scale.

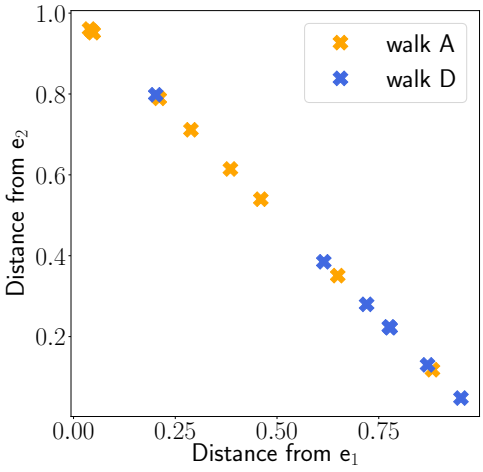


(d) Distance for C and D from e_2, logaritmic scale.

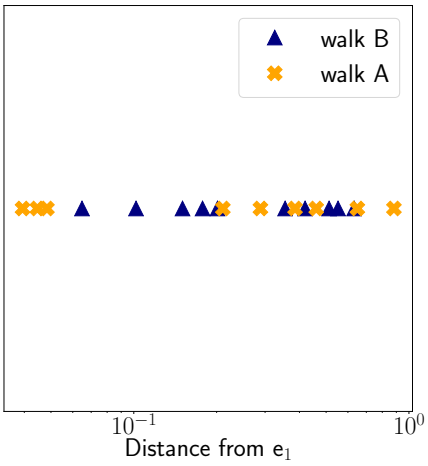
Figure 8.4: Distances of the 4 walks to be classied from the 2 selected eigenwalks, VBR case.



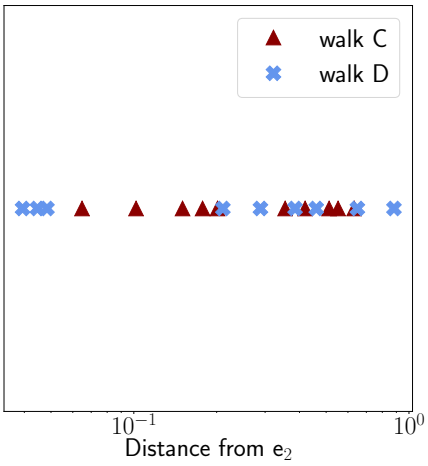
(a) Distance for B and C from e_1 and e_2.



(b) Distance for A and D from e_1 and e_2.



(c) Distance for B and A from e_1, logaritmic scale.



(d) Distance for C and D from e_2, logaritmic scale.

Figure 8.5: Distances of the 4 walks to be classied from the 2 selected eigenwalks, CBR case.

- the availability of a representative memory makes the difference: it suffices to have a memory of 10% to achieve significantly better results. This is mainly due to the *catastrophic forgetting* phenomenon mentioned in Chapter 7;
- as expected, the higher the memory, the better;
- the best λ_D values are entirely empirical and they depend on the designed model. For instance, $\lambda_D = 1$ is preferable in most of the cases but it is not always true. Note that, also $\lambda_D = 0$ is sometimes the best but, in these cases, the reached scores aren't remarkably higher. However, in future studies, the λ_D influence could be deeper investigated;
- when the distillation loss is considered (i.e.$\lambda_D \neq 0$) reaching the highest scores, the quality of predictions improve also for the new added class;
- thanks to the incremental procedure, results comparable to non-incremental models can be obtained.

Figures 8.6 and 8.7 shown the worst and the best confusion matrices attained with incremental framework in the VBR and CBR case, respectively.

The reported results highlight the catastrophic phenomenon: in the worst case all the classes tend to be confused with the last added one.

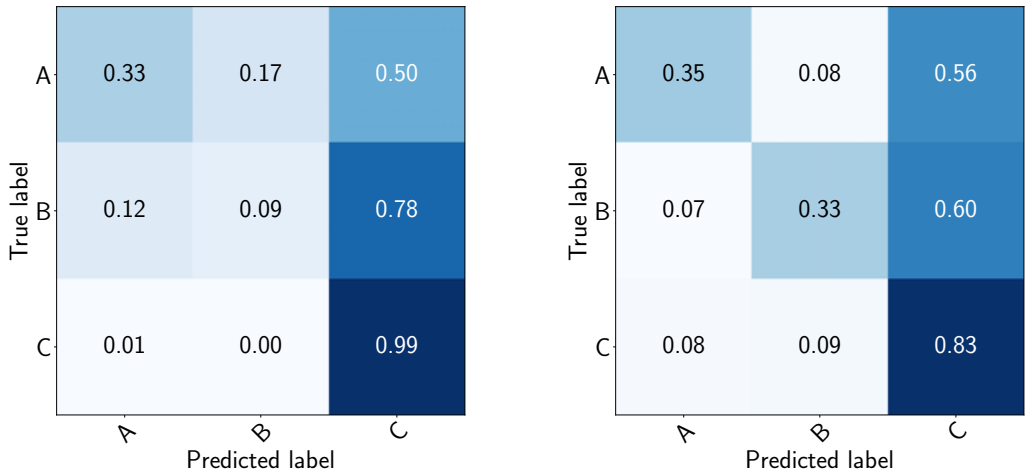
To conclude the analysis, the 4^{th} walk is classified in the new 3D space. In the 1^{st} level of the DT algorithm it is always associated with the 3^{rd} walk e_3. Figure 8.8 shows the distances of A and D from e_3 with VBR and CBR encodings. In the CBR case, clusters aren't easily discernible but, in the VBR mode, a double threshold technique could work. This double threshold mechanisms could be furhter investigated in future analysis.

Table 8.2: F_1 scores resulting from the incremental procedure, VBR case.

			Walk A	Walk B	Walk C	Weighted F_1 score
F_1 score	$\mathcal{M}^0_{0\%}$	$\lambda_D = 0$	0.5	0.02	0.72	0.50
		$\lambda_D = 1$	0.31	0.10	0.73	0.46
		$\lambda_D = 10$	0.46	0.14	0.74	0.52
	$\mathcal{M}^0_{10\%}$	$\lambda_D = 0$	0.65	0.81	0.82	0.77
		$\lambda_D = 1$	0.89	0.90	0.92	0.91
		$\lambda_D = 10$	0.79	0.77	0.86	0.82
	$\mathcal{M}^0_{25\%}$	$\lambda_D = 0$	0.95	0.97	0.97	*0.97*
		$\lambda_D = 1$	0.94	0.96	0.97	0.96
		$\lambda_D = 10$	0.89	0.77	0.88	0.86
	$\mathcal{M}^0_{50\%}$	$\lambda_D = 0$	0.97	0.96	0.98	0.97
		$\lambda_D = 1$	0.94	0.97	0.97	0.96
		$\lambda_D = 10$	0.92	0.86	0.92	0.91
	$\mathcal{M}^{-1}_{A\text{-}C}$		0.97	0.96	0.96	0.97

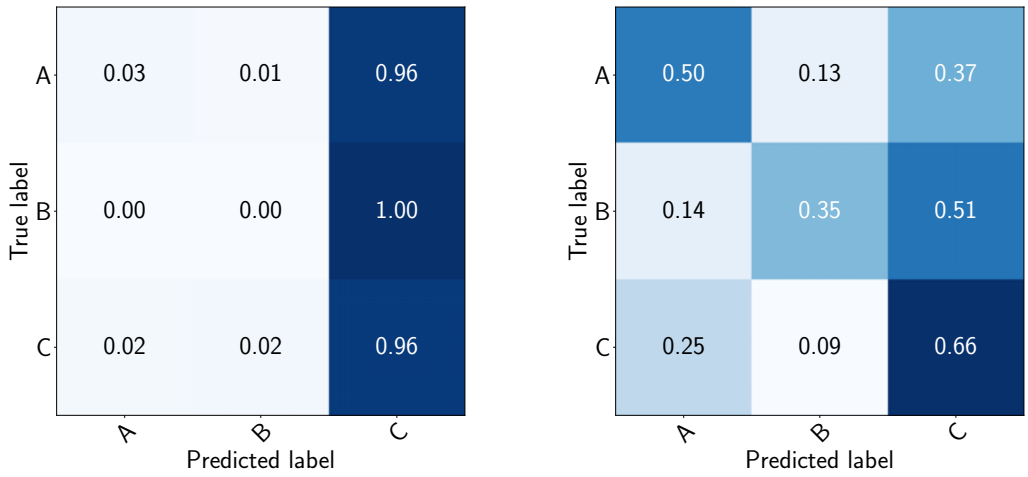
Table 8.3: F_1 scores resulting from the incremental procedure, CBR case.

			Walk A	Walk B	Walk C	Weighted F_1 score
F_1 score	$\mathcal{M}^0_{0\%}$	$\lambda_D = 0$	0.04	0.01	0.50	0.20
		$\lambda_D = 1$	0.05	0	0.52	0.21
		$\lambda_D = 10$	0.03	0.008	0.89	0.20
	$\mathcal{M}^0_{10\%}$	$\lambda_D = 0$	0.34	0.17	0.55	0.38
		$\lambda_D = 1$	0.29	0.18	0.57	0.37
		$\lambda_D = 10$	0.32	0.05	0.53	0.35
	$\mathcal{M}^0_{25\%}$	$\lambda_D = 0$	0.45	0.23	0.59	0.46
		$\lambda_D = 1$	0.47	025	0.58	0.47
		$\lambda_D = 10$	0.44	0.13	0.58	0.43
	$\mathcal{M}^0_{50\%}$	$\lambda_D = 0$	0.57	0.37	0.55	0.53
		$\lambda_D = 1$	0.53	0.38	0.54	0.51
		$\lambda_D = 10$	0.46	0.37	0.57	0.53
	$\mathcal{M}^{-1}_{A\text{-}C}$		0.56	0.65	0.43	0.57



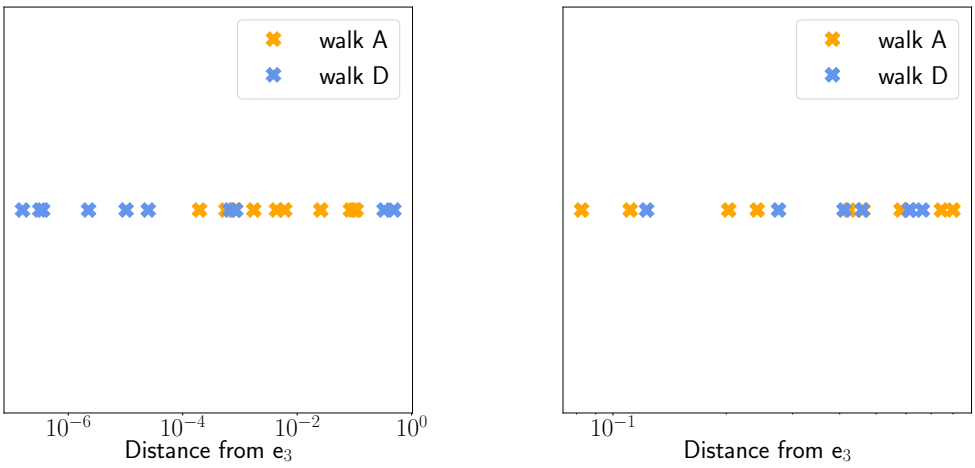
(a) Worst confusion matrix. **(b) Best confusion matrix.**

Figure 8.6: Incremental framework: worst and best results, VBR case.



(a) Worst confusion matrix. **(b) Bestt confusion matrix.**

Figure 8.7: Incremental framework: worst and best results, CBR case.



(a) VBR scenario

(b) CBR scenario.

Figure 8.8: Incremental framework: distance for C and D from e_3, logaritmic scale.

8.2 Real environment

8.2.1 Datasets comparison

Figures 8.9 and 8.10 show real signals, corresponding to Volunteer 1 and Volunteer 4 walks, comparing the 4 datasets examinated. First of all, note that real signals are noisier than the ones in the virtual environment. Moreover, from the mentioned Figures, a progressive noise reduction can be appreciated. This predicts that if the forensic analyst is able to perform, somehow, background reduction, some quality improvement can be obtained.

Tables 8.4 and 8.5, where the validation set scores are reported after the first classification in the VBR and CBR cases, provide evidence for these statements.

Note that, in the VBR scenario, the 4^{th} dataset is the best while, in the CBR mode, the weighted F_1 scores between the first two datasets and the last two do not differ.

For the reason stated above, the rest of the investigation is carried out exploiting this last dataset.

8.2.2 Eigenwalks selection

As stated in 7.4.3, V_1 and V_2 are nominated in advance to be the 2 eigenwalks e_1, e_2 in the real environment. As can be seen in Figure 8.13 these are, indeed, the 2 walks that maximize the average distance in the VBR case. In the CBR mode, their distance are not too far from $d^{max}_{c_1,c_2}$.

8.2.3 VBR vs CBR

Exploiting the outcomes of the second CNN training, it can be noticed that both the walks V_3 and V_4 are associated to the 1^{st} eigenwalk. This can be motivated by the fact that walk V_1 is the one with the highest number of acquisitions. Figure 8.14 shows the distances of V_1, V_2 and V_3 from e_1.

In the VBR case the threshold thr_1 is set to 0.001 while, in CBR mode, it is equal to 0.01, since the distances estimated through the validation set are 0.00 and 0.01 respectively. Note that, also in the real environment the CBR encoding leads to the worst results: in Figure 8.14b one thresold is not sufficient to distinguish walk V_1 from the other two but a double threshold strategy could be more suitable.

Table 8.6 quantifies the F_1 scores resulting from the two DT algorithms.

Table 8.4: F_1 scores for walk V_1, V_2 and V_3 after applying the first classification in the VBR scenario. Results are computed on the validation set.

		Walk V_1	Walk V_2	Walk V_3	Weighted F_1 score
F_1 score	Original dataset	0.75	0.36	0.45	0.63
	Cropped original dataset	0.87	0.68	0.46	0.80
	B&W dataset	0.76	0.40	0.74	0.67
	Cropped B&W dataset	0.96	0.83	0.76	*0.91*

Table 8.5: F_1 scores for walk V_1, V_2 and V_3 after applying the first classification in the CBR scenario. Results are computed on the validation set.

		Walk V_1	Walk V_2	Walk V_3	Weighted F_1 score
F_1 score	Original dataset	0.67	0.2	0.42	0.54
	Cropped original dataset	0.82	0.45	0.36	*0.70*
	B&W dataset	0.63	0.33	0.41	0.54
	Cropped B&W dataset	0.80	0.41	0.55	*0.70*

Note that, the results for the last two walks indicate the accuracy with which they are distinguished from walk V_1. In future studies, with more walks available, it could also be quantified with how much accuracy V_3 and V_4 are distinguishalbe from each other.

8.2.4 Incremental Learning

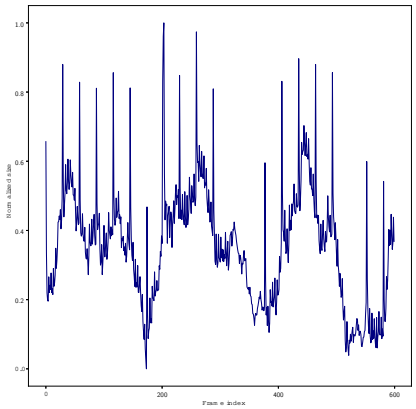
Tables 8.7 and 8.8 show the outcomes achieved with the incremental procedure in the real environment. The same conclusions presented in Section 8.1.4 can be derived.

Furthermore, the interesting aspect to highlight once again is the gap between virtual and real scenarios: in the former case, the highest reached score is 0.97, in the latter one, the maximum value is 0.85.

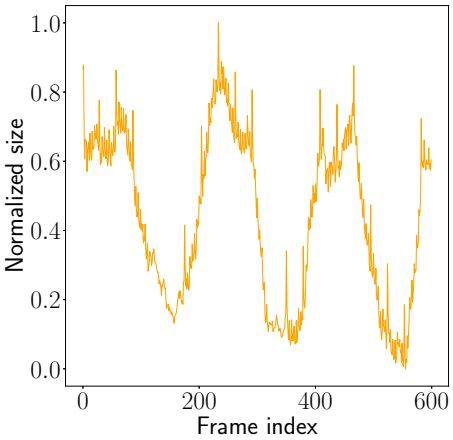
To conclude, exploiting the *DT* algorithm with this incremental procedure, the 4^{th} walk

Table 8.6: F_1 scores for walk V_1, V_2, V_3 and V_4 after applying the *DT* algorithm in the test set.

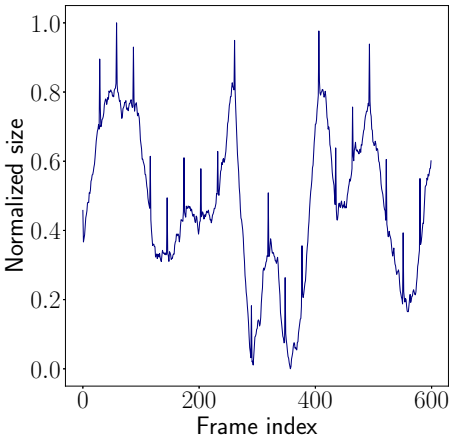
			Walk V_1	Walk V_2	Walk V_3	Walk V_4	Weighted F_1 score
F_1 score	VBR	1^{st} level	0.92	0.88	0.67	0.92	0.85
		2^{nd} level	0.92	0.88	1	1	0.95
	CBR	1^{st} level	0.86	0.75	1	1	0.89



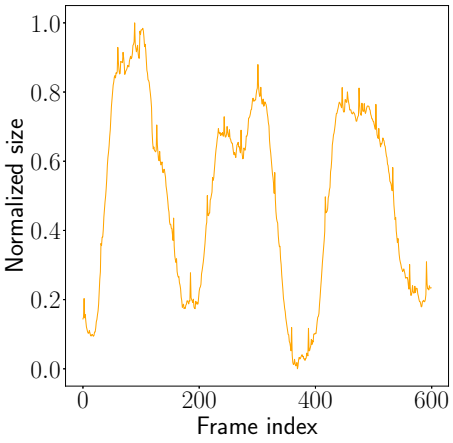
(a) Signal derived from the original image.



(b) Signal derived from the B&W image.

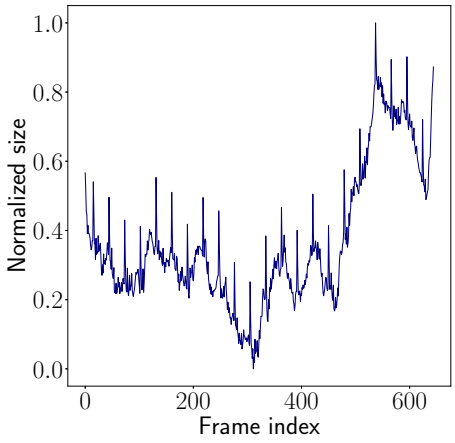


(c) Signal derived from the cropped original image.

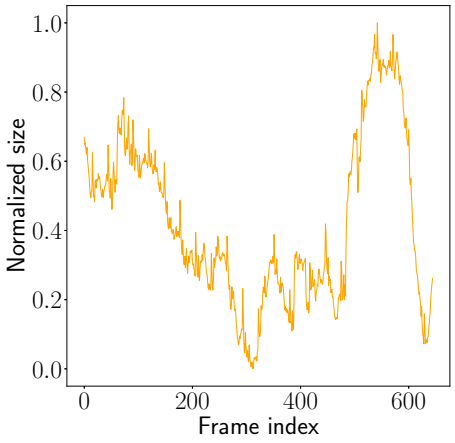


(d) Signal derived from the cropped B&W image.

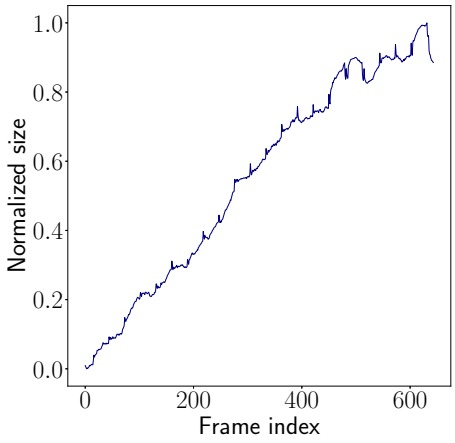
Figure 8.9: Signals corresponding to Volunteer 1 walks, VBR case.



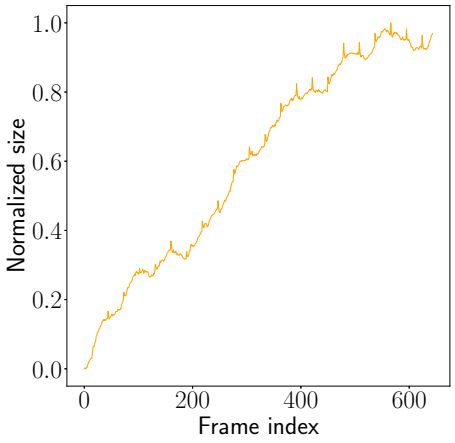
(a) Signal derived from the original image.



(b) Signal derived from the B&W image.

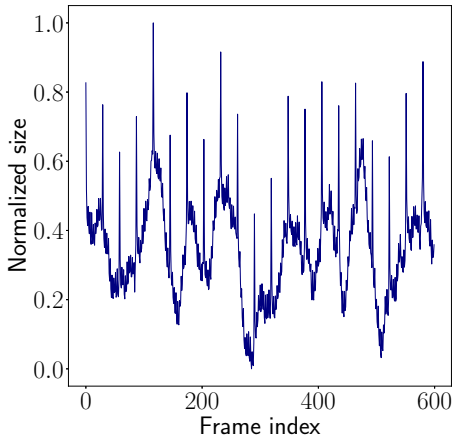


(c) Signal derived from the cropped original image.

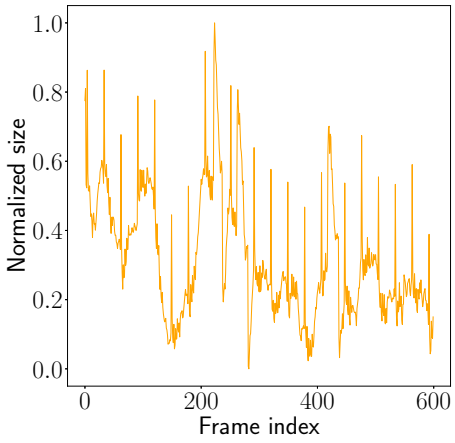


(d) Signal derived from the cropped B&W image.

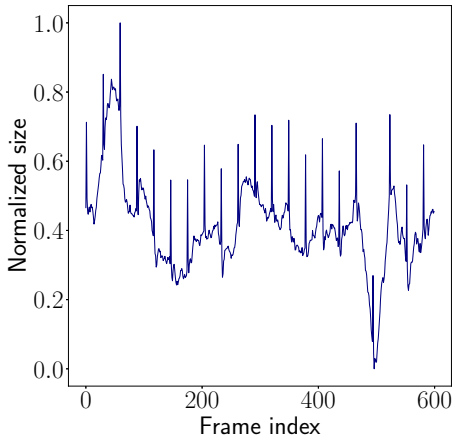
Figure 8.10: Signals corresponding to Volunteer 4 walks, VBR case.



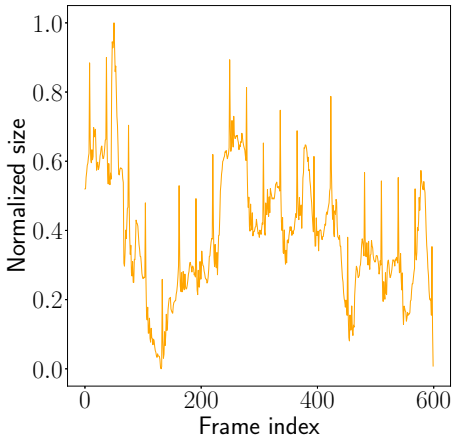
(a) Signal derived from the original image.



(b) Signal derived from the cropped original image.

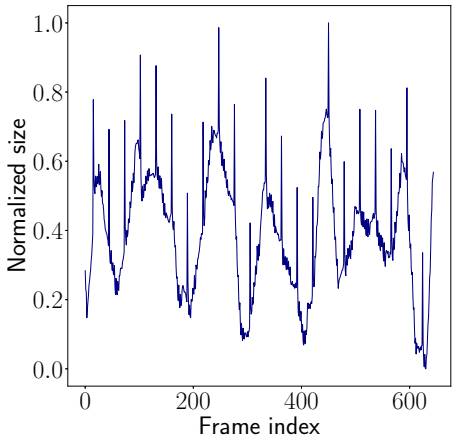


(c) Signal derived from the B&W image.

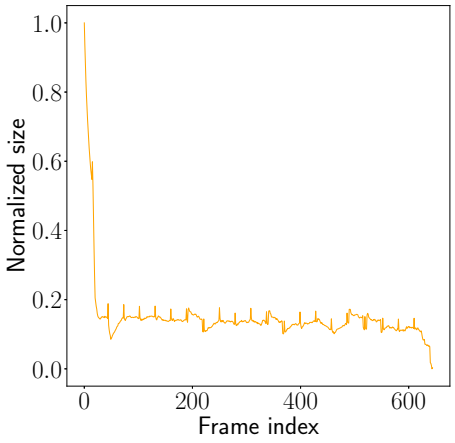


(d) Signal derived from the cropped B&W image.

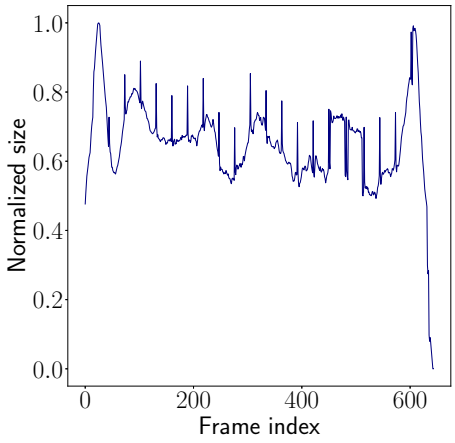
Figure 8.11: Signals corresponding to Volunteer 1 walks, CBR case.



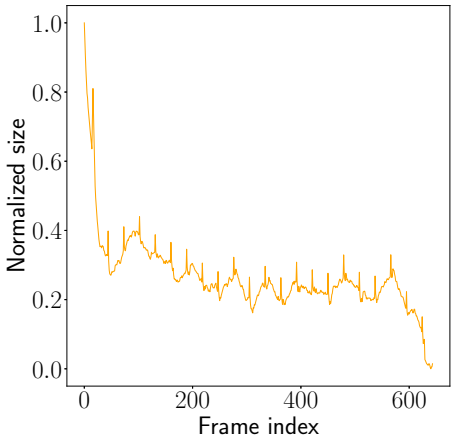
(a) Signal derived from the original image.



(b) Signal derived from the cropped original image.

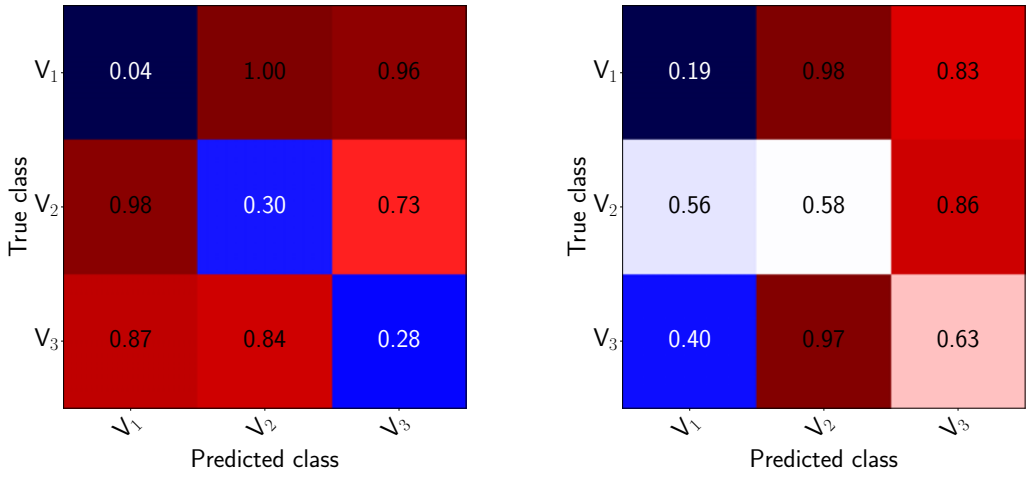


(c) Signal derived from the B&W image.



(d) Signal derived from the cropped B&W image.

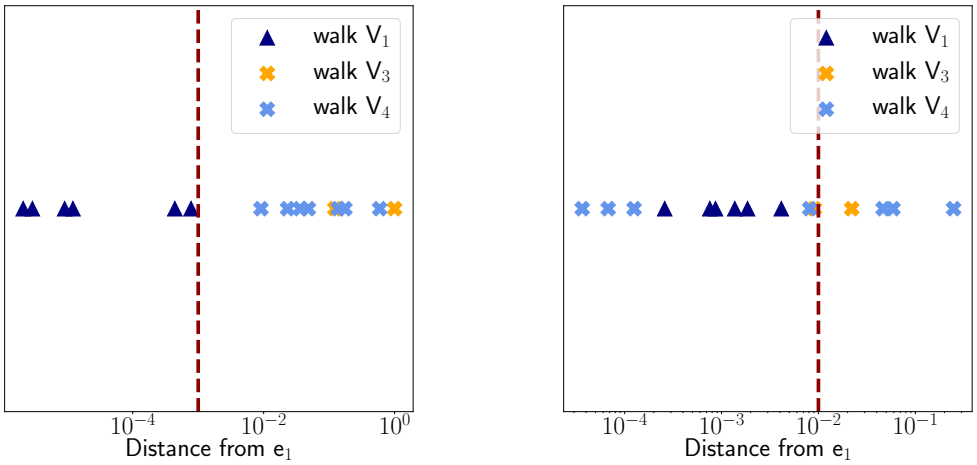
Figure 8.12: Signals corresponding to Volunteer 4 walks, CBR case.



(a) Distance matrix *DM*, VBR case.

(b) Distance matrix *DM*, CBR case.

Figure 8.13: Outcomes resulting from the first classification in the real case. The results are computed on the validation set.



(a) VBR case.

(b) CBR case.

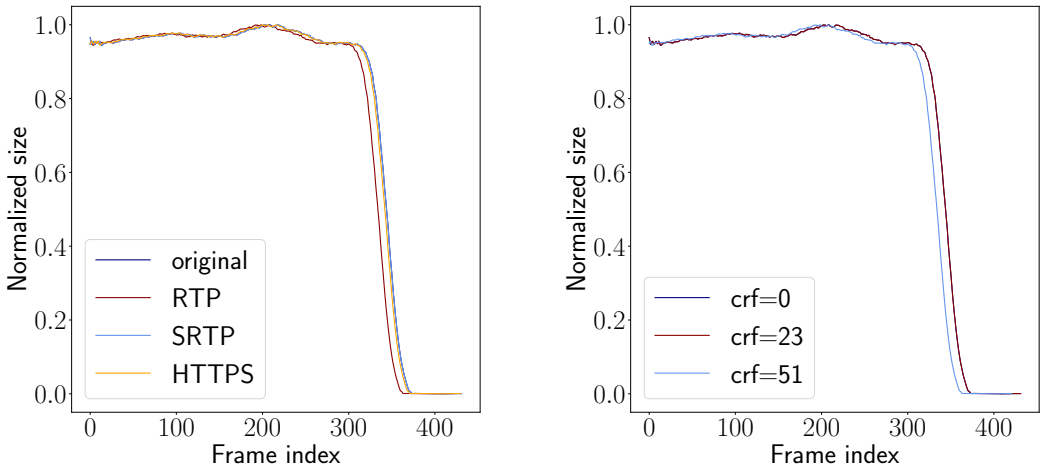
Figure 8.14: Distances of walks V_1, V_3 and V_4 from e_1.

Table 8.7: F_1 scores resulting from the incremental procedure, real scenario, VBR case.

			Walk V_1	Walk V_2	Walk V_3	Weighted F_1 score
F_1 score	$\mathcal{M}^0_{0\%}$	$\lambda_D = 0$	0	0.02	0.32	0.06
		$\lambda_D = 1$	0	0.4	0.33	0.09
		$\lambda_D = 10$	0	0.44	0.33	0.09
	$\mathcal{M}^0_{10\%}$	$\lambda_D = 0$	0	0.63	0.95	0.82
		$\lambda_D = 1$	0	0.56	0.93	0.80
		$\lambda_D = 10$	0.01	0.56	0.93	0.80
	$\mathcal{M}^0_{25\%}$	$\lambda_D = 0$	0.02	0.63	0.95	0.83
		$\lambda_D = 1$	0	0.56	0.93	0.80
		$\lambda_D = 10$	0	0.70	0.96	0.85
	$\mathcal{M}^0_{50\%}$	$\lambda_D = 0$	0	0.64	0.94	0.82
		$\lambda_D = 1$	0	0.63	0.95	0.82
		$\lambda_D = 10$	0.03	0.56	0.922	0.80
		$\mathcal{M}^{-1}_{D\text{-}A}$	0.98	0.90	0.91	0.96

Table 8.8: F_1 scores resulting from the incremental procedure, real scenario, CBR case.

			Walk V_1	Walk V_2	Walk V_3	Weighted F_1 score
F_1 score	$\mathcal{M}^0_{0\%}$	$\lambda_D = 0$	0	0.21	0.31	0.07
		$\lambda_D = 1$	0	0.16	0.31	0.06
		$\lambda_D = 10$	0	0.21	0.30	0.07
	$\mathcal{M}^0_{10\%}$	$\lambda_D = 0$	0.04	0	0.88	0.66
		$\lambda_D = 1$	0	0	0.87	0.65
		$\lambda_D = 10$	=.02	0	0.87	066
	$\mathcal{M}^0_{25\%}$	$\lambda_D = 0$	0.06	0	0.88	0.66
		$\lambda_D = 1$	0.01	0	0.88	0.66
		$\lambda_D = 10$	0.05	0	0.88	0.66
	$\mathcal{M}^0_{50\%}$	$\lambda_D = 0$	0.09	0	0.87	0.66
		$\lambda_D = 1$	0.05	0	0.86	0.65
		$\lambda_D = 10$	0.05	0	0.86	0.65
		$\mathcal{M}^{-1}_{D\text{-}A}$	0.91	0.75	0.81	0.84



(a) Protocols comparison.

(b) Constant rate factor comparison.

Figure 8.15: Outcomes resulting from Samuele's encryption tool.

is associated to the 3^{rd} one both in the VBR and CBR case but the data are too few to quantify the precision with which these can be distinguished from one another.

8.2.5 Encrypted videos

Figure 8.15 show the outcomes resulting from Samuele's encryption tool (i.e. the reconstructed packet sizes). In particular, Figure 8.15a compares the protocols while, Figure 8.15b, compares different values of CRF. It is evident that neither of the two parameters influences the data obtained: the shapes are exactly the same. In the first Figure, the original signal is superimposed on the RTP one while, in the second Figure, the first two signals overlap.

9 Conclusion

This dissertation addressed gait analysis for identification purposes in the forensic field. The novelty of this work relies on the fact that we processed time series signals rather than developing more complex image-based processing techniques.

We examinated both virtual and real characters applying some slightly different analysis. In both environments, we inspected VBR and CBR encodings but, given that the real scenario is noisier than the virtual one, we proposed 3 different types of noise reduction approaches.

Furthermore, we leveraged eigenwalks strategy to classify 4 walks exploiting only two of them. This procedure could be very useful in real applications where it is asked to classify several people knowing the data of only a subset of them.

The study proved that VBR encoding is more vulnerable to information leakage rather than CBR scenario, both in the virtual and real environment.

The real scenario confirmed to be the most challenging one: it was necessary to adopt image processing techniques, before working with signals, to achieve satisfying results. Specifically, it was noticed that the background influences the outcomes. For this reason, 3 different types of background removal was proposed based on image cropping or silhouette extraction.

Thanks to these operations, the people classification system implemented with this project achieves high accuracy also in the real environment, even if still not comparable with the virtual case where the reached quality is almost maximum.

The study was further extended designing an incremental framework, to make the analysis suitable to real applications where data must be continually learned.

We want to highlight that the combination of the eigenwalk strategy with the incremental procedure is perfectly suited to real needs where new data can be identified and clustered through the 1^{st} approach until a sufficient amount of them are collected to take advantage of the 2^{nd} technique.

We conclude this project asserting the robustness under encryption mechanisms: video surveillance systems exploit SRTP or HTTPS protocols to protect transmitted data but both of them are not effective in protecting individuals privacy from passive traffic analysis.

Since we tackled several aspects in this work, the open research fields are many.

Two of the main investigation trends will be:

- *extension of the eigenwalk technique:* we could be verify if 2 eigenwalks are enough to classify more than 4 walks or if it is necessary to extend the eigenwalks space considering 3 or more eignewalks;
- *extension of the incremental procedure:* we could adopt other incremental approaches, such as layers' freezing, or we could address the simultaneous addition of multiple classes. Moreover, we could explore deeper the λ_D influence.

These further examinations are actually the next plans for the future.

Additionally, we could extend the analysis started in this thesis in the real environment exploiting a larger dataset with a triple purpose:

- *increase the accuracy;*
- *classify more people;*
- *prove the generalization capability.* More precisely, we could verify the robustness to: light conditions, walking speed or clothes change. We could also show the ability to identify the same person on different days where the external factors subject to change could be many.

There is one last aspect worthy of being investigated and even more intriguing: given the strong correlation between reality and noise, instead of processing images, we could work with signals to implement background removal. Specifically, we could leverage the UNET convolutional autoencoder to perform signal denoising. This investigation is worth to be pursued for a dual objective:

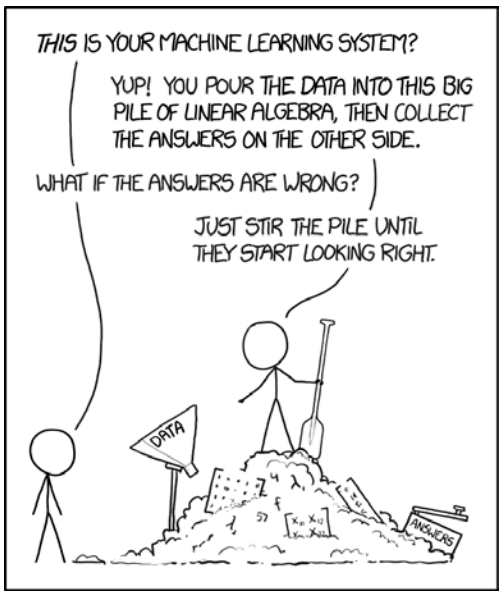
- prove that people identification achieves very high accuracy in real environments leveraging only signals informations (i.e none image preprocessing is required at all);
- show the effectiveness of deep learning architectures to perform not only classification tasks but also for denoising purposes.

To conclude, the whole project gives its own contribution to prove that: *Human gait is a promising biometrics resource.*

9.1 Personal considerations

Besides technical aspects, the major learned lesson in the development of this project is the human one: the importance and beauty of working in a group whose members have different backgrounds but, above all, share the same passion for their work.

With this project I experienced: *The true team working.*


THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS

References

[1] J. Singh, S. Jain, S. Arora, and D. U. Singh, "Vision-based gait recognition: A survey," *IEEE Access*, vol. PP, 11 2018.

[2] U. technologies, "Manual." [Online]. Available: https://docs.unity3d.com/Manual/index.html

[3] B. A. Forouzan, *Data Communication and Networking*, 5th ed. McGraw-Hill Education, 2013.

[4] A. S. Tanenbaum, D. Wetherall *et al.*, *Computer networks*. Prentice hall, 1996.

[5] "Machine learning types." [Online]. Available: https://towardsdatascience.com/machine-learning-types-2-c1291d4f04b1

[6] "Real-life and business applications of neural networks." [Online]. Available: https://www.smartsheet.com/neural-network-applications

[7] "The mostly complete chart of neural networks, explained." [Online]. Available: https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464

[8] "Introducing convolutional neural networks." [Online]. Available: https://developers.google.com/machine-learning/practica/image-classification/convolutional-neural-networks

[9] A. W. Harley, "An interactive node-link visualization of convolutional neural networks," in *International Symposium on Visual Computing*. Springer, 2015, pp. 867–877.

[10] "Enhancing cnn incremental learning capability with an expanded network." [Online]. Available: https://www.semanticscholar.org/paper/Enhancing-CNN-Incremental-Learning-Capability-with-Cai-Xu/dbc8142fbb952f721d88eccd12c61017c81f92e4

[11] R. Baker, "The history of gait analysis before the advent of modern computers," *Gait & posture*, vol. 26, no. 3, pp. 331–342, 2007.

[12] P. Connor and A. Ross, "Biometric recognition by gait: A survey of modalities and features," *Computer Vision and Image Understanding*, vol. 167, pp. 1–27, 2018.

[13] D. Gafurov, "A survey of biometric gait recognition: Approaches, security and challenges," in *Annual Norwegian computer science conference*. Annual Norwegian Computer Science Conference Norway, 2007, pp. 19–21.

[14] "Deep learning vs classical machine learning." [Online]. Available: https://towardsdatascience.com/deep-learning-vs-classical-machine-learning-9a42c6d48aa

[15] A. K. Jain, A. Ross, S. Prabhakar *et al.*, "An introduction to biometric recognition," *IEEE Transactions on circuits and systems for video technology*, vol. 14, no. 1, 2004.

[16] "Forensic gait analysis principles and practice." [Online]. Available: https://www.routledge.com/Forensic-Gait-Analysis-Principles-and-Practice/Birch-Nirenberg-Vernon-Birch/p/book/9781138386846

[17] G. Shakhnarovich, L. Lee, and T. Darrell, "Integrated face and gait recognition from multiple views," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1. IEEE, 2001, pp. I–I.

[18] P. K. Larsen, E. B. Simonsen, and N. Lynnerup, "Gait analysis in forensic medicine," *Journal of forensic sciences*, vol. 53, no. 5, pp. 1149–1153, 2008.

[19] F. Corona, M. Pau, M. Guicciardi, M. Murgia, R. Pili, and C. Casula, "Quantitative assessment of gait in elderly people affected by parkinson's disease," in *2016 IEEE International Symposium on Medical Measurements and Applications (MeMeA)*. IEEE, 2016, pp. 1–6.

[20] X. Ye, "Side channel leakage analysis-detection, exploitation and quantification," 2015.

[21] P. C. Kocher, "Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems," in *Annual International Cryptology Conference*. Springer, 1996, pp. 104–113.

[22] A. Reed and M. Kranch, "Identifying https-protected netflix videos in real-time," in *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*. ACM, 2017, pp. 361–368.

[23] "Microsoft silverlight." [Online]. Available: https://www.microsoft.com/silverlight

[24] A. Reed and B. Klimkowski, "Leaky streams: Identifying variable bitrate dash videos streamed over encrypted 802.11 n connections," in *2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2016, pp. 1107–1112.

[25] M. Conti, L. V. Mancini, R. Spolaor, and N. V. Verde, "Can't you hear me knocking: Identification of user actions on android apps via traffic analysis," in *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*. ACM, 2015, pp. 297–304.

[26] C. Wampler, S. Uluagac, and R. Beyah, "Information leakage in encrypted ip video traffic," in *2015 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2015, pp. 1–7.

[27] H. Li, Y. He, L. Sun, X. Cheng, and J. Yu, "Side-channel information leakage of encrypted video stream in video surveillance systems," in *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE, 2016, pp. 1–9.

[28] S. Milani, M. Tagliasacchi, and S. Tubaro, "Identification of the motion estimation strategy using eigenalgorithms," in *2013 IEEE International Conference on Image Processing*. IEEE, 2013, pp. 4477–4481.

[29] M. Sorell, "Video provenance by motion vector analysis: A feasibility study," in *2012 5th International Symposium on Communications, Control and Signal Processing*. IEEE, 2012, pp. 1–4.

[30] "Game engines—how do they work?" [Online]. Available: https://unity3d.com/what-is-a-game-engine

[31] "Innovation, immersion, intensity." [Online]. Available: https://www.idsoftware.com/en-us

[32] J. Gregory, *Game engine architecture*. AK Peters/CRC Press, 2015.

[33] A. Zarrad, "Game engine solutions," in *Simulation and Gaming*. BoD–Books on Demand, 2018, pp. 75–87.

[34] "Unity for all." [Online]. Available: https://unity.com/

[35] J. K. Haas, "A history of the unity game engine," 2014.

[36] J. Xie, "Research on key technologies base unity3d game engine," in *2012 7th international conference on computer science & education (ICCSE)*. IEEE, 2012, pp. 695–699.

[37] M. Best, "Announcing the unity editor for linux," May 2019. [Online]. Available: https://blogs.unity3d.com/2019/05/30/announcing-the-unity-editor-for-linux/

[38] U. technologies, "Build once, deploy anywhere." [Online]. Available: https://unity3d.com/unity/features/multiplatform

[39] ——, "Asset workflow." [Online]. Available: https://docs.unity3d.com/Manual/AssetWorkflow.html

[40] W. Goldstone, *Unity game development essentials*. Packt Publishing Ltd, 2009.

[41] U. technologies, "Asset types." [Online]. Available: https://docs.unity3d.com/Manual/AssetTypes.html

[42] "Animated 3d characters. no 3d knowledge required." [Online]. Available: https://www.mixamo.com/#/

[43] U. technologies, "Creatingscenes." [Online]. Available: https://docs.unity3d.com/Manual/CreatingScenes.html

[44] ——, "Gameobject." [Online]. Available: https://docs.unity3d.com/Manual/class-GameObject.html

[45] ——, "Components." [Online]. Available: https://docs.unity3d.com/Manual/Components.html

[46] ——, "Monobehaviour." [Online]. Available: https://docs.unity3d.com/ScriptReference/MonoBehaviour.html

[47] ——, "Monobehaviour.start()." [Online]. Available: https://docs.unity3d.com/ScriptReference/MonoBehaviour.Start.html

[48] ——, "Transforms." [Online]. Available: https://docs.unity3d.com/Manual/Transforms.html

[49] ——, "Prefabs." [Online]. Available: https://docs.unity3d.com/Manual/Prefabs.html

[50] ——, "The scene view." [Online]. Available: https://docs.unity3d.com/Manual/UsingTheSceneView.html

[51] ——, "Scene view navigation." [Online]. Available: https://docs.unity3d.com/Manual/SceneViewNavigation.html

[52] ——, "The hierarchy window." [Online]. Available: https://docs.unity3d.com/Manual/Hierarchy.html

[53] ——, "The inspector window." [Online]. Available: https://docs.unity3d.com/Manual/UsingTheInspector.html

[54] ——, "Quick guide to the unity asset store." [Online]. Available: https://unity3d.com/quick-guide-to-unity-asset-store

[55] ——, "Using the asset store." [Online]. Available: https://docs.unity3d.com/Manual/AssetStore.html

[56] ——, "The project window." [Online]. Available: https://docs.unity3d.com/Manual/ProjectView.html

[57] ——, "Console window." [Online]. Available: https://docs.unity3d.com/Manual/Console.html

[58] "Autodesk 3ds max." [Online]. Available: https://www.autodesk.it/products/3ds-max/overview

[59] "Autodesk maya." [Online]. Available: https://www.autodesk.it/products/maya/overview

[60] U. technologies, "Animation clips." [Online]. Available: https://docs.unity3d.com/Manual/AnimationClips.html

[61] ——, "Animation overview." [Online]. Available: https://docs.unity3d.com/Manual/AnimationOverview.html

[62] ——, "Animator controllers." [Online]. Available: https://docs.unity3d.com/Manual/AnimatorControllers.html

[63] ——, "Monobehaviour.update()." [Online]. Available: https://docs.unity3d.com/ScriptReference/MonoBehaviour.Update.html

[64] ——, "Event functions." [Online]. Available: https://docs.unity3d.com/Manual/EventFunctions.html

[65] ——, "Adam." [Online]. Available: https://unity3d.com/pages/adam

[66] ——, "Baymax-dreams." [Online]. Available: https://unity.com/madewith/baymax-dreams

[67] ——, "Unity and autodesk: Powering immersive experiences." [Online]. Available: https://unity.com/partners/autodesk

[68] "Innoactive hub. deploy xr across your organization." [Online]. Available: https://innoactive.de/hub/

[69] U. technologies, "Speed up knowledge transfer." [Online]. Available: https://unity.com/solutions/automotive-transportation/training-and-guidance

[70] S. Edelstein, "How gaming company unity is driving automakers toward virtual reality," may 2018. [Online]. Available: https://www.digitaltrends.com/cars/unity-automotive-virtual-reality-and-hmi/

[71] "Deepmmind." [Online]. Available: https://deepmind.com/

[72] "Dannny lange." [Online]. Available: http://www.dannylange.ai/bio

[73] M. team, "Dreamship: Immersive technology in the silicon valley @mime." [Online]. Available: http://mime.dei.unipd.it/life-mime/dreamship-immersive-technology-in-the-silicon-valley

[74] "Health and education in immersive technology." [Online]. Available: http://hannahluxenberg.com/health

[75] "Hannah luxenberg. film director | xr manager | artificial intelligence." [Online]. Available: http://hannahluxenberg.com/

[76] "Dreamship studios and kodak pixpro. bringing dreams alive through 360° therapy." [Online]. Available: https://www.kodak.com/uploadedFiles/Consumer/Products/Cameras/Action/SP3604K/Testimonials/SP360-4K-VR-CaseStudy-Dreamship.pdf

[77] I. E. Richardson, *The H. 264 advanced video compression standard*. John Wiley & Sons, 2011.

[78] "An overview of h.264 advanced video coding." [Online]. Available: AnOverviewofH.264AdvancedVideoCoding

[79] "Hevc: An introduction to high efficiency coding." [Online]. Available: https://www.vcodex.com/hevc-an-introduction-to-high-efficiency-coding/

[80] "H.264 vs h.265 — a technical comparison. when will h.265 dominate the market?" [Online]. Available: https://medium.com/advanced-computer-vision/h-264-vs-h-265-a-technical-comparison-when-will-h-265-dominate-the-market-266593031710

[81] "Historical timeline of video coding standards and formats." [Online]. Available: https://www.vcodex.com/historical-timeline-of-video-coding-standards-and-formats/

[82] K. Sayood, *Introduction to data compression*. Morgan Kaufmann, 2017.

[83] "Variable bit rate (vbr)." [Online]. Available: https://www.techopedia.com/definition/4759/variable-bit-rate-vbr

[84] "Ffmpeg vbr settings." [Online]. Available: https://slhck.info/video/2017/02/24/vbr-settings.html

[85] "Ffmpeg vbr settings." [Online]. Available: https://www.lifewire.com/what-is-cbr-2438539

[86] "Cbr vs vbr vs mbr - surveillance streaming." [Online]. Available: https://ipvm.com/reports/vbr-vs-cbr-surveillance-streaming

[87] "Cbr vs vbr encoding." [Online]. Available: https://www.lifewire.com/difference-between-cbr-and-vbr-encoding-2438423

[88] "Ffmpeg." [Online]. Available: https://ffmpeg.org/

[89] V. Kumar and J. Svensson, *Promoting social change and democracy through information technology*. IGI Global, 2015.

[90] "Onvif." [Online]. Available: https://www.onvif.org/

[91] "Onvif streaming specification." [Online]. Available: https://www.onvif.org/specs/stream/ONVIF-Streaming-Spec-v210.pdf

[92] "Psia." [Online]. Available: https://psialliance.org/

[93] V. Jacobson, R. Frederick, S. Casner, and H. Schulzrinne, "Rtp: A transport protocol for real-time applications," 2003.

[94] J. Postel, "User datagram protocol," *Isi*, 1980.

[95] ——, "Transmission control protocol," 1981.

[96] H. Schulzrinne, "Real time streaming protocol (rtsp)," 1998.

[97] D. A. McGrew and K. Norrman, "The secure real-time transport protocol (srtp)," 2004.

[98] "What does tunneling mean?" [Online]. Available: https://www.techopedia.com/definition/5402/tunneling

[99] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext transfer protocol–http/1.1," 1999.

[100] E. Rescorla, "Http over tls," 2000.

[101] T. Dierks, "The transport layer security (tls) protocol version 1.2," 2008.

[102] J. Daemen and V. Rijmen, "Aes proposal: Rijndael," 1999.

[103] N.-F. Standard, "Announcing the advanced encryption standard (aes)," *Federal Information Processing Standards Publication*, vol. 197, no. 1-51, pp. 3–3, 2001.

[104] M. Dworkin, "Recommendation for block cipher modes of operation. methods and techniques," National Inst of Standards and Technology Gaithersburg MD Computer security Div, Tech. Rep., 2001.

[105] S. Piazzetta, "Studio ed analisi dei protocolli e degli algoritmi utilizzati nei dispositivi di video sorveglianza."

[106] "7 reasons why machine learning is a game changer for agriculture." [Online]. Available: https://towardsdatascience.com/7-reasons-why-machine-learning-is-a-game-changer-for-agriculture-1753dc56e310

[107] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[108] "The complete beginners guide to deep learning." [Online]. Available: https://towardsdatascience.com/intro-to-deep-learning-c025efd92535

[109] "Usage of loss functions." [Online]. Available: https://keras.io/losses/

[110] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[111] "A comprehensive guide to convolutional neural networks." [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[112] "More on cnns handling overfitting." [Online]. Available: https://towardsdatascience.com/deep-learning-3-more-on-cnns-handling-overfitting-2bd5d99abe5d

[113] U. Michieli and P. Zanuttigh, "Incremental learning techniques for semantic segmentation," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[114] R. Istrate, A. C. I. Malossi, C. Bekas, and D. Nikolopoulos, "Incremental training of deep convolutional neural networks," *arXiv preprint arXiv:1803.10232*, 2018.

[115] D. Roy, P. Panda, and K. Roy, "Tree-cnn: A hierarchical deep convolutional neural network for incremental learning," *arXiv preprint arXiv:1802.05800*, 2018.

[116] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 12, pp. 2935–2947, 2017.

[117] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[118] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2006, pp. 535–541.

[119] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[120] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari, "End-to-end incremental learning," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 233–248.

[121] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of cognitive neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.

[122] "Ffmpeg vbr settings." [Online]. Available: https://slhck.info/video/2017/02/24/vbr-settings.html

[123] "Ffprobe documentation." [Online]. Available: https://ffmpeg.org/ffprobe.html

[124] "Batch normalization in neural networks." [Online]. Available: https://towardsdatascience.com/batch-normalization-in-neural-networks-1ac91516821c

[125] "Wireshark." [Online]. Available: https://www.wireshark.org/

[126] "Accuracy vs. f1-score." [Online]. Available: https://medium.com/analytics-vidhya/accuracy-vs-f1-score-6258237beca2

[127] H. Sagha, S. T. Digumarti, J. d. R. Millán, R. Chavarriaga, A. Calatroni, D. Roggen, and G. Tröster, "Benchmarking classification techniques using the opportunity human activity dataset," in *2011 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, 2011, pp. 36–40.

Acknowledgments

Come conclusione dell'intero percorso, vorrei ringraziare alcune persone.

Ringrazio, in primis, il professor Milani, per avermi *regalato* l'esperienza più bella nell'ambito universitario e per la sua *infinita disponibilità*.

L'onnipresente professor Badia, per aver sempre creduto in me.

Tutti quei professori, a partire dalla professoressa Sgargetta, che rendono del loro lavoro una passione, per avermela trasmessa.

Un sentito grazie va anche a tutti i camminatori, del DEI e non, per aver investito diverse giornate a *Camminare per la tesi di Sara*.

In particolare, grazie a: Sebastiano, Fabio, Umberto Michieli, Umbi, Albi, Elena, Marco, Samuele, Daniele Mari, Daniele Bagatella e Leo.

Infine, gli ultimi ringraziamenti, ma non meno importanti, vanno a don Giorgio, per avermi permesso di usare la location che ha reso possibile lo sviluppo dell'intero progetto nei tempi previsti.

Ma soprattutto:

Grazie a chi quotidianamente mi SUPPORTA ed è PRESENTE,
anche con un semplice messaggio.

.


GRAZIE