POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Forensic Detection of Deepfakes Generated Through Video-to-Video Translation

Laurea Magistrale in Music and Acoustic Engineering

Author: Carmelo Fascella

Advisor: Prof. Paolo Bestagini

Co-advisor: Dr. Sara Mandelli

Academic year: 2020-2021

1. Introduction

The widespread diffusion of manipulated media content has become a central problem in recent years, especially after the advent of so-called *deepfakes*. These consist of synthetic images, videos and audio generated by deep learning-based techniques where the identities, intentions and emotions of portrayed human characters are altered. The potential applications of this technology can provide interesting results and new opportunities, but at the same time significant security implications. Thus, it is of crucial importance to develop multimedia forensic techniques able to detect tampering and forgeries of various kinds and distinguish real from synthesized media.
In this work, we propose two forensic detectors able to infer if a video sequence is synthesized or not. The first detector, which is considered as a baseline reference, is based on the use of Convolutional Neural Networks (CNNs). The second detector is based on the extraction of temporal textural descriptors from the video sequences and on the classification of these features using a Multi-Layer Perceptron (MLP).

2. Problem Statement

We focus on videos representing a scene in which we have a single subject moving within it, performing any kind of moves. The person is filmed by a fixed camera in a way that all parts of his body are clearly visible. Given this kind of footage, our goal is to detect whether a certain video is authentic or it has been synthetically generated by the video-to-video translation algorithm *Everybody Dance Now* [3].
Formally, let us consider a video corresponding to a sequence of consecutive frames $\mathbf{V} = (\mathbf{F}_0, \mathbf{F}_1, ..., \mathbf{F}_N)$. The authenticity detection is a binary classification problem, which consists in defining the following function:

$$f : \mathbf{V} \rightarrow \hat{y} \tag{1}$$

The function f consists in a binary classifier, which receives as input the video sequence $\mathbf{V}$ and it is able to predict as output the discrete label $\hat{y} \in \{0, 1\}$, where 0 means "real", and 1 means "fake". Our goal is to introduce and compare different classification functions f.

$\mathbf{V}^{(g)}$ $\mathbf{V}^{(o)}$

Figure 1: Sample frames from the pre-processed videos.

3. Proposed Methodology

We propose two classification methods that find an estimate of the label of a video sequence. First, we introduce a classification approach based on a CNN that is considered as a comparison baseline. The second proposed method is based on the extraction of handcrafted textural features. In the following sections, we describe the pre-processing phase and the body parts area selection function, which is used to select the video parts we focus on, prior to feed them to the classifier. Finally, we provide details on the two proposed classification methods.

3.1. Video Pre-processing

Each video $\mathbf{V}$ is a 4D matrix of shape $N \times C \times H \times W$, where N is the number of frames, $C = 3$ is the number of color channels, and H and W are the height and width of the single frames. We propose to pre-process videos in two different ways, obtaining: (i) $\mathbf{V}^{(g)}$, which is the grayscale version of $\mathbf{V}$; (ii) $\mathbf{V}^{(o)}$, which is the gradient of the optical flow extracted from $\mathbf{V}$ with the *Flownet2* architecture [4]. In Figure 1, we show two sample frames of the pre-processed videos. We call the generic pre-processed video as $\mathbf{V}^{(\rho)}$, where the index $\rho = [g, o]$ indicates if we are considering $\mathbf{V}^{(g)}$ or $\mathbf{V}^{(o)}$.

3.2. Body Parts Area Selection Algorithm

Specific areas of each video corresponding to relevant body parts of the subjects are selected frame-by-frame. The idea is to analyze the texture of the subjects parts that contain important details useful for the recognition of the person, that are therefore more difficult to synthesize. The chosen body parts of the subjects are $\mathcal{B} = \{FC, LH, RH, LF, RF\}$: the face ($FC$), the left hand ($LH$), the right hand ($RH$), the left foot ($LF$) and the right foot ($RF$).

To select the different body parts for each frame, we first estimate the coordinates of 137 keypoints of the subject, by means of the *Openpose* detector [2]. Each keypoint is referred to a certain part of the skeleton. Then, we define an area selection function, which takes as input the keypoints and outputs the coordinates of the 4 points $\mathcal{M}_{\mathbf{F},b} = \{(w_1, h_1), (w_1, h_2), (w_2, h_1), (w_2, h_2)\}$ which delimits the area where the body part $b \in \mathcal{B}$ is located: w_1 and w_2 are the coordinates along the x-axis, h_1 and h_2 along the y-axis.
Each area selection function accurately estimates the position of a bounding box surrounding a body part by taking into account the keypoints of that body part. We use the face area selection function introduced in [3], and we introduce another selection function adapted by us to select the other body parts.

3.3. CNN-based classification

The baseline approach is based on the EfficientNet-B0 architecture [5]. The network processes our videos in a frame-by-frame mode, and works as feature extractor as well as classifier of the learned features. Inspired by state-of-the-art face manipulation detection, in this configuration we consider only the videos $\mathbf{V}^{(g)}$ as input, and we crop them on the subject face.

3.4. Feature-based classification

In this section we introduce the core method used for our classification task. It is based on the extraction of handcrafted textural features, namely Local Pattern Derivative on Three Ortogonal Planes (LDP-TOP) [1], and the use of MLP as classifier. In Figure 2, we show the general pipeline of the proposed method. Given a video $\mathbf{V}$, the input $\mathbf{V}^{(\rho)}$ of our classification block can be one of the two pre-processed video versions (i.e., $\mathbf{V}^{(g)}$ or $\mathbf{V}^{(o)}$). The proposed pipeline consists of six main steps:

1. *Temporal partition*: By definition, each LDP-TOP feature corresponds to a sequence of K consecutive frames. To do so, we need to isolate temporal sequences $\mathbf{S}^{(\rho)}$ from our input video $\mathbf{V}^{(\rho)}$.
2. *Body parts cropping*: Given a temporal sequence $\mathbf{S}^{(\rho)}$, we consider all the frames $\mathbf{F}^{(\rho)} \in \mathbf{S}^{(\rho)}$. Then, we get all the body parts area coordinates $\mathcal{M}_{\mathbf{F},b}$, $b \in \mathcal{B}$, obtained as

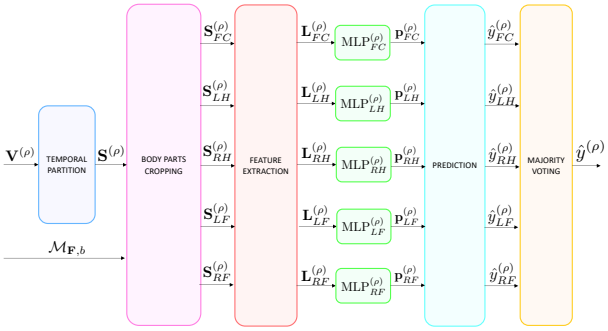

Figure 2: Feature-based classification outline.

reported in Section 3.2. For each body part, we crop the part inside the area bounded by the coordinates $\mathcal{M}_{\mathbf{F},b}$. In doing so, we get the sequence of the cropped body parts indicated with the notation $\mathbf{S}_b^{(\rho)} = \{\mathbf{S}_{FC}^{(\rho)}, \mathbf{S}_{LH}^{(\rho)}, \mathbf{S}_{RH}^{(\rho)}, \mathbf{S}_{LF}^{(\rho)}, \mathbf{S}_{RF}^{(\rho)}\}$.

3. *Feature extraction*: For each sequence obtained at the step before, we extract the 1D feature vector LDP-TOP of size 3072, indicated as $\mathbf{L}_b^{(\rho)}$.
4. *MLP classification*: the features $\mathbf{L}_b^{(\rho)}$ are fed into the classifiers $\text{MLP}_b^{(\rho)}$, in order to predict their class. The MLP architecture is the following: an input layer which takes as input the feature vector of size 3072; three hidden layers composed respectively by 100, 50 and 25 neurons; an output layer composed by 2 neurons. This gives as output $\mathbf{p}_b^{(\rho)} = [(\mathbf{p}_b^{(\rho)})_0, (\mathbf{p}_b^{(\rho)})_1]$ that indicates the probability of the sequence $\mathbf{S}_b^{(\rho)}$ to belong to one of the two classes "real" or "fake".
5. *Prediction*: The predicted label $\hat{y}_b^{(\rho)}$ of the input sequence $\mathbf{S}_b^{(\rho)}$ can be computed as:

$$\hat{y}_b^{(\rho)} = \arg\max_i (\mathbf{p}_b^{(\rho)})_i. \tag{2}$$

6. *Majority voting*: to evaluate if the sequence $\mathbf{S}^{(\rho)}$ is real or fake, we compute the majority voting between all the predictions $\hat{y}_b^{(\rho)}$ related to each body part sequence $\mathbf{S}_b^{(\rho)}$:

$$\hat{y}^{(\rho)} = \text{maj}\Big(\{\hat{y}_{FC}^{(\rho)}, \hat{y}_{LH}^{(\rho)}, \hat{y}_{RH}^{(\rho)}, \hat{y}_{LF}^{(\rho)}, \hat{y}_{RF}^{(\rho)}\}\Big). \tag{3}$$

The operator $\text{maj}(\cdot)$ returns the most recurring value between the input predictions.

3.5. Combined Prediction Technique

The combined prediction technique allows to predict the label of a generic temporal sequence $\mathbf{S}$ by considering at the same time the contribution of the two sequences $\mathbf{S}^{(g)}$ and $\mathbf{S}^{(o)}$.
We first follow the steps [1,2,3,4] explained in the previous section, for both $\mathbf{V}^{(g)}$ and $\mathbf{V}^{(o)}$.
For each $b \in \mathcal{B}$, we calculate the probability of the sequence $\mathbf{S}_b$ as the average between the probability calculated for the corresponding sequences $\mathbf{S}_b^{(g)}$ and $\mathbf{S}_b^{(o)}$:

$$\bar{\mathbf{p}}_b = \frac{\mathbf{p}_b^{(g)} + \mathbf{p}_b^{(o)}}{2}. \tag{4}$$

Then, we compute the predicted label $\hat{y}_b$ of each input sequence $\mathbf{S}_b$ by applying (2) to the probability scores $\bar{\mathbf{p}}_b$. Eventually, we apply majority voting between all the predictions $\hat{y}_b$ as suggested in (3).

4. Experiments and Results

In this chapter, we describe the used dataset and the experimental setup. Then, we report the achieved results and compare the two proposed detectors in different scenarios.

4.1. Dataset

The dataset is divided into three parts:

- *Source videos*: these videos are used as source for the video-to-video translation algorithm, to impose the motion of the source subjects onto the target videos. This part consists of 5 videos taken from YouTube.
- *Target videos*: these are the videos on which we want to transfer the movements of the subjects of the source videos. They are used to train the models needed for the video-to-video translation. This part of the dataset consists of 10 long videos: the longest one is composed by 30219 frames, and the average number of frames is 13574. Each video corresponds to a different human subject. Videos are taken from different sources: *Everybody Dance Now* Dataset, homemade videos and YouTube videos. Furthermore, these videos are used for our classification task and they are labeled as "real".
- *Synthesized videos*: we generate these videos by following the video synthesis algorithm described in [3]. Our dataset contains

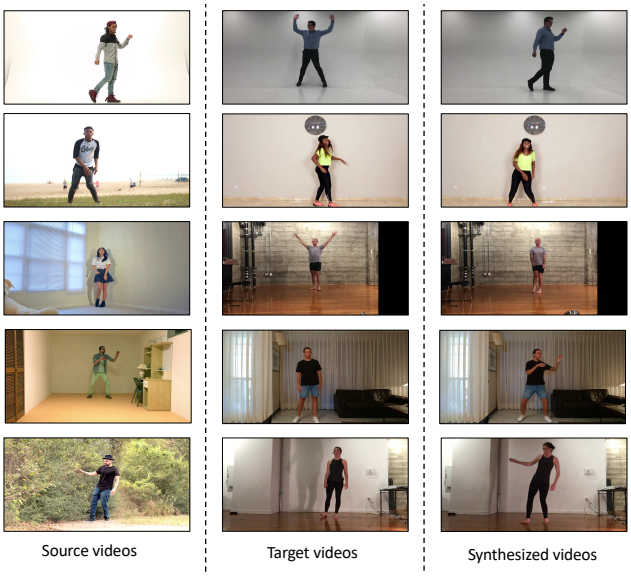

Figure 3: Sample of video frames taken from our dataset.

50 synthesized videos, where each of the 10 target subjects is synthesized with the appearance of each of the 5 source videos.

We generate the video dataset $\mathcal{V}$ by considering the original target videos and the synthesized videos. All videos are coded in H.264, with a frame resolution of 1024×512 pixel and Constant Rate Factor (CRF)=0. Furthermore, for each video, we generate different compressed and resized versions. For the compression, we use four different CRF values: 18, 23, 40, 51. Each video is resized to the following resolutions (in pixel): 512×256, 256×128, 128×64, 64×32. In Figure 3, we can see some frames taken from source videos, target videos and synthesized videos of our dataset.

4.2. Experimental Setup

In this section we discuss our experimental setup.

4.2.1 Temporal Sequences Partition

For every video, we extract all the possible temporal sequences, which are composed of $K = 50$ consecutive frames. The amount of frames between the beginning of one sequence and the beginning of the next one is $Q = 25$.

4.2.2 Leave-One-Out Cross Validation

We have a reduced number of subjects (only 10 persons) appearing in our videos. To counteract the limited number of subjects, we evaluate our investigations by using Leave-One-Out Cross-Validation (LOOCV), which is a particular case of k-fold Cross-Validation. The parameter k controls the number of subsets that the dataset is split into. Each subset is given the opportunity to be used as a test set while all other subsets together are used as a training dataset. In our case, we fit and evaluate $k = 10$ models, one for each subject in the dataset.

More specifically, given our dataset $\mathcal{V}$, let us consider the subset $\{\mathcal{N}_n\} \subset \mathcal{V}$, with $n = [0, ..., k-1]$, which is composed of all the real and fake videos where the n-th subject appears, and the subset $\{\mathcal{V} \setminus \mathcal{N}_n\} \subset \mathcal{V}$ which is composed of all the real and fake videos where the n-th subject does not appear. The idea is to do a train-test split of our dataset k times, one for each of the k available subject.

4.2.3 CNN and MLP setup

EfficientNet-B0 takes as input *cropped-on-face* frames from the pre-processed grayscale videos. The chosen resolution of the cropped frames is 70×70 pixels. Concerning the dataset split policy, first we apply the LOOCV technique. Then, we randomly split the training dataset in 80% of frames for training and 20% for validation. We train for at most 200 epochs, and the training phase is stopped if the validation loss does not decrease for more than 10 epochs. The model providing the best validation loss is selected.

On the other hand, MLP takes as input the features extracted from a certain body part area selected of a pre-processed videos. We train the network for at most 300 epochs. As the number of selected body parts $|\mathcal{B}| = 5$, we train 5 different MLPs for each of the two pre-processed video versions. Concerning the dataset split policy, we apply the LOOCV technique and we use all frames for training.

For both the two proposed methodologies, we use Cross-Entropy Loss (CEL) as a loss function, and Adam as optimizer.

4.2.4 Evaluation metrics

As we are dealing with a LOOCV dataset split policy, for each experiment we need to consider all the k dataset partitions. For each n-th partition, we evaluate the learned model over all

Body Parts → Technique ↓	*FC*	*LH*	*RH*	*LF*	*RF*	maj
with $\mathbf{V}^{(g)}$	98.83%	93.68%	94.30%	94.36%	95.22%	98.99%
with $\mathbf{V}^{(o)}$	88.70%	72.59%	75.98%	84.80%	86.95%	91.28%
Combined	98.87%	93.23%	93.69%	94.49%	96.08%	**99.30**%

Table 1: MBA comparison in the three feature-based classification methods.

the temporal sequences extracted from the test videos. We compute the results in terms of balanced accuracy in correctly estimating original and synthesized temporal sequences. To infer about the general reliability of our method, we introduce the Mean of Balanced Accuracies (MBA), which is the arithmetic mean among all the k balanced accuracies achieved on the sequences associated with the k subjects.

4.3. Results

In the CNN-based method, we predict the label of a temporal sequence of frames by averaging the predictions on the single frames of the sequence. The resulting MBA is 99.10%. For what concerns the feature-based method, in Table 1 we compare the results obtained by considering only the videos $\mathbf{V}^{(g)}$ as input, then only the videos $\mathbf{V}^{(o)}$ as input, and in the end the combined prediction technique.

The best results are obtained using the combined prediction technique. These results are very similar to the ones obtained considering only $\mathbf{V}^{(g)}$ videos, while the results obtained considering only $\mathbf{V}^{(o)}$ videos are worse than the previous ones. In each configuration, we achieve the best results by applying the majority voting between the body parts predictions.

We also compare the performances of the two methods as a function of the training dataset size. As we can see in Figure 4, by varying the number of frames in the training set, the accuracy of the feature-based method remains more or less constant, while the accuracy of the CNN-based method decreases significantly. The feature-based method proves more robust than the baseline, maintaining an accuracy of over 90% in all cases, while the baseline accuracy drops below 60% in the worst case.

Furthermore, we investigate what happens if we leave out more than one subject from the training dataset, and test on the corresponding videos. So, we test our methods using the Leave-p-Out Cross-Validation (LpOCV) technique, where the parameter p refers to the number of subjects whose videos are tested. We consider only 10 of the possible dataset split combinations for each LpOCV split, chosen in a random way. Figure 5 shows that the accuracies of both methods decreases as p increases. The predictions start being inaccurate when the number of subjects in the training set is too low.

We test the robustness of the LDP-TOP method on video compression and resizing and evaluate cross-test results, i.e., when training and testing data did not undergo the same processing operations. Figure 6 shows the results using the combined prediction technique.

For what concerns the tests on compressed videos, on the main diagonal we always get

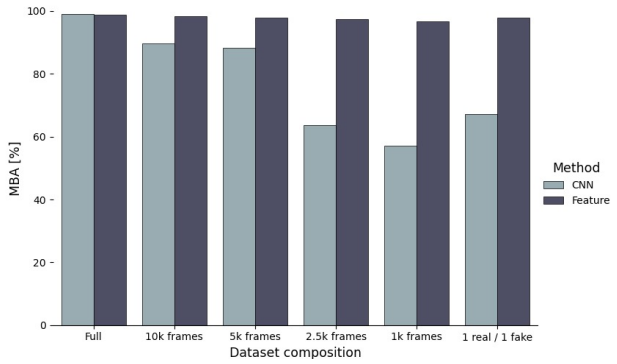

Figure 4: Comparison between CNN and LDP-TOP based methods by varying the number of frames per subject in the training dataset, considering as input only lossless videos $\mathbf{V}^{(g)}$ cropped on the face. "Full" refers to the starting point dataset composition; "10k, 5k, 2.5k, 1k frames" refers to the number of frames taken for each subject in the training dataset; in "1 real / 1 fake" we train the model by randomly selecting only one real video and one fake video for each subject, using all available frames.

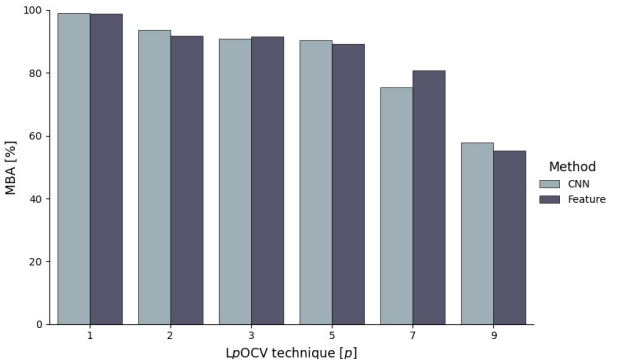

Figure 5: Comparison between CNN and LDP-TOP based methods using LpOCV with different values of p, considering only lossless videos $\mathbf{V}^{(g)}$ cropped on the face.

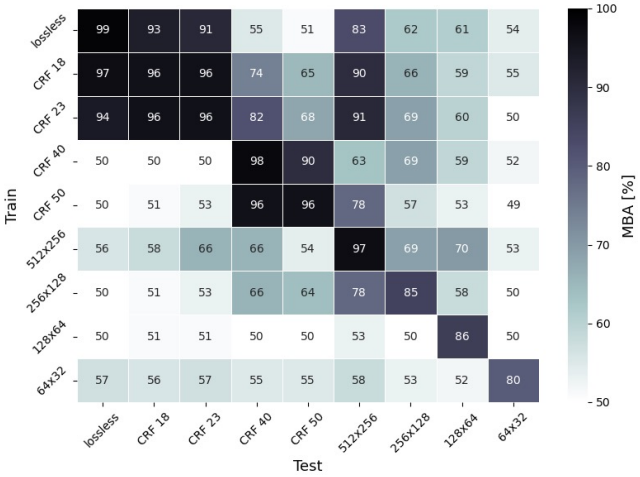

Figure 6: Cross-tests on compressed and resized videos, using the LDP-TOP combined prediction technique and majority voting between the body parts.

over 96% of MBA. Nonetheless, if we train on videos compressed with CRF $\leq$ 23, we achieve very good results in predicting videos with lower CRF, and we can still discriminate lower quality videos. By training on higher CRFs and testing on high quality videos, accuracy drops.
For what concerns the results on resized videos, we get good results on the main diagonal, but, as expected, the MBA decreases as the resolution decreases. Cross-test results drop when the training and/or testing video resolution becomes too small.

5. Conclusions and Future Works

The goal of this work was to analyze whether a video sequence is original or synthetic. We focused on deepfake videos which contain human characters moving inside the scene generated through the video-to-video translation algorithm *Everybody Dance Now* [3].
We created our own dataset because, to the best of our knowledge, there is not a comprehensive and publicly available dataset of synthesized human characters. In the absence of a true baseline for this kind of video classification, we first introduced a CNN-based approach. Then, we introduced different strategies based on the extraction of LDP-TOP features from video sequences and their classification with a MLP. We considered two pre-processing applied on videos and we extracted features from different body parts, doing the predictions for each of them.
Both methods achieved high classification accuracies. In the feature-based method we showed that, in general, the majority voting between the body part predictions achieved the best accuracy results. This method proved to be robust in case of limited training data conditions, unlike the CNN-based one.
Unfortunately, we have not been able to test our method on a large number of human subjects. Right now, the synthetic video generation process is very expensive in terms of computation and data collection. We firmly believe that, in the years to come, new technologies able to generate this kind of data at a lower cost will be released. Therefore, our method can be considered as a valid starting point for the classification of this type of videos.

References

[1] Mattia Bonomi, Cecilia Pasquini, and Giulia Boato. Dynamic texture analysis for detecting fake faces in video sequences. *Journal of Visual Communication and Image Representation*, 79:103239, 2021.

[2] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: realtime multi-person 2d pose estimation using part affinity fields. *IEEE transactions on pattern analysis and machine intelligence*, 43(1):172–186, 2019.

[3] Caroline Chan, Shiry Ginosar, Tinghui Zhou, and Alexei A Efros. Everybody dance now. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5933–5942, 2019.

[4] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2462–2470, 2017.

[5] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.

POLITECNICO
MILANO 1863

Dipartimento di Elettronica, Informazione e Bioingegneria

Master Degree in Computer Science and Engineering

Forensic Detection of Deepfakes Generated Through Video-to-Video Translation

by:
Carmelo Fascella

matr.:
913248

Supervisor:
Prof. Paolo Bestagini

Co-supervisor:
Dr. Sara Mandelli

Academic Year
2020-2021

Abstract

The creation of manipulated videos involving human characters has reached in recent years unprecedented realism. The diffusion of altered content may lead to severe consequences if it contains misleading pieces of information. For this reason, it is increasingly necessary to develop forensic methodologies that enable to verify if a given video has been digitally synthesized or not.

In this thesis, we focus on the problem of discerning whether a video sequence is original or not. The synthesized videos we are dealing with are generated by a video-to-video translation algorithm using pose as an intermediate representation, where the entire body of the involved human character is synthesized by a latest generation software.

We propose two methodologies to solve this task. The first detector, which is considered as a baseline reference, is based on the use of a Convolutional Neural Network (CNN). The second detector is based on the extraction of temporal textural descriptors from the video sequences and on the classification of these features using a Multi-Layer Perceptron (MLP). The features are extracted from five body parts of the human subjects, and we investigate two different pre-processed versions of the videos.

The proposed solutions are tested on a dataset we have generated and designed specifically for this task. The dataset involves 50 synthesized videos with a minimum length of 2 minutes of recordings, depicting 10 different persons.

Both proposed methodologies achieve high accuracy classification results. Moreover, we show that our proposed feature-based method is robust to different levels of video compression and resizing, and outperforms the baseline based on CNNs in different training conditions.

Sommario

La creazione di video che coinvolgono persone reali ha raggiunto negli ultimi anni un realismo senza precedenti. La diffusione di contenuti alterati può portare a gravi conseguenze se essi contengono informazioni fuorvianti. Per questo motivo, è sempre più necessario sviluppare tecniche forensi che consentono di verificare se un certo video è stato sintetizzato digitalmente o meno.

Il problema affrontato in questa tesi è capire se una sequenza video è originale o falsa. I video sintetizzati su cui focalizziamo la nostra attenzione sono generati da un algoritmo di traduzione da video a video che utilizza la posa come rappresentazione intermedia, dove l'intero corpo del soggetto umano coinvolto è sintetizzato da un software di ultima generazione.

Proponiamo due metodologie per affrontare questo problema. Il primo metodo, che è considerato come riferimento di base, si basa sull'uso di una Convolutional Neural Network (CNN). Il secondo metodo si basa sull'estrazione di descrittori spazio-temporali dalle sequenze video e sulla classificazione di tali caratteristiche utilizzando un Multi-Layer Perceptron (MLP). I descrittori sono estratti da cinque parti del corpo dei soggetti umani, considerando due diverse versioni pre-elaborate dei video.

Le soluzioni proposte sono testate su un insieme di dati che abbiamo generato e progettato appositamente per questo lavoro. L'insieme di dati comprende 50 video sintetizzati con una lunghezza minima di 2 minuti, raffiguranti 10 soggetti diversi. Entrambe le metodologie da noi proposte raggiungono un'alta accuratezza di classificazione. Inoltre, dimostriamo che il nostro metodo basato sull'estrazione dei descrittori è robusto rispetto alla compressione e al ridimensionamento, e supera il metodo basato su CNNs in diversi scenari.

Acknowledgments

This thesis is the result of almost a year of hard work and dedication.

First, I would like to thank Prof. Bestagini and Dr. Sara Mandelli for giving me the opportunity to work on topics in which I am passionate about, for their complete availability and constant support. It has been an honour and a pleasure working with you.

I would like to thank my friends and my classmates. I have spent many unforgettable moments with you that allowed me to make the most of these long years of study.

I would like to thank my bandmates for your true friendship and for sharing with me the passion for music and audio technology, which drove me on this path.

I would like to thank my parents: it is thanks to your support and encouragement that I have been able to reach this important goal.

Carmelo

Contents

List of Figures

List of Tables

Glossary

CEL Cross-Entropy Loss. 54, 55

CNN Convolutional Neural Network. i, ii, vi–viii, 2, 3, 8, 9, 15, 18, 28, 29, 31, 38, 47, 51, 53, 56, 57, 59–63, 67, 68

CRF Constant Rate Factor. 51, 56, 57, 64, 68

GAN Generative Adversarial Network. vi, 10, 11, 18, 19, 23–25, 27, 28

LpOCV Leave-p-Out Cross-Validation. vii, 61, 63

LBP Local Binary Pattern. vi, 12–14

LDP Local Pattern Derivative. 12, 13

LDP-TOP Local Pattern Derivative on Three Ortogonal Planes. vi, vii, 2, 12, 14, 41–43, 57, 59, 62, 63, 67, 69

LOOCV Leave-One-Out Cross-Validation. viii, 52–54, 56, 57, 61

LSTM Long Short-Term Memory. 29

MBA Mean of Balanced Accuracies. vii, viii, 56–60

MLP Multi-Layer Perceptron. i, ii, vi, vii, 2, 5, 6, 8, 18, 41, 43, 46, 51, 54, 57, 59, 61, 63, 67

MSE Mean Square Error. 7

NN Neural Network. 4–8, 15, 17, 18

PAF Part Affinity Field. 21

ReLU Rectified Linear Unit. 7, 9, 54

SVC Support Vector Classifier. vii, 59, 61

1 Introduction

The widespread diffusion of manipulated media content has become a central problem in recent years, especially after the advent of so-called *deepfakes* [10], which consist of synthetic images, videos and audio generated by deep learning-based techniques where the identities, intentions and emotions of portrayed human characters are altered. Rapid progress in the field of computer graphics and machine vision has enabled the development of easy-to-use software available on smartphones and laptops, enabling the manipulation and creation of highly realistic synthesized content.

The potential applications of this technology can provide interesting results and new opportunities in the field of photography, video games, virtual reality and film production. However, at the same time, such technology has significant security implications and can be used with malicious intent in different scenarios, like election manipulation, creation of warmongering situations, defaming any person and revenge porn. Furthermore, the impact of *deepfakes* is amplified by the action of social networks that deliver information quickly and worldwide.

Thus, it is of crucial importance to develop multimedia forensic techniques able to detect tampering and forgeries of various kinds and distinguish real from synthesized media. While the identification of computer-generated faces has been widely addressed for both images and videos due to the high level of visual quality achieved by the generators, the prob-

lem of identification of computer-generated human bodies still needs to be addressed. Only recently, the first software capable of fully synthesizing the entire body of a human subject has appeared, and we expect the quality of their synthesis to increase in the coming years.

The aim of this thesis is to determine if a video sequence is original or synthesized. The peculiarity of the synthesized videos considered in this work is that they contain human characters moving inside the scene and whose body is completely synthesized through a video-to-video translation algorithm. To the best of our knowledge, this is the first work which proposes a reliable detector on these kind of videos.

We propose two different video classification approaches. The first method, which we consider as a reference baseline, involves the use of a standard Convolutional Neural Network (CNN) [11] which classifies a video by leveraging only the face of human subjects. The second method aims at exploiting textural and temporal information of the video sequences by extracting hybrid descriptors operating in both spatial and temporal domain, and classify them with a simple Multi-Layer Perceptron (MLP) classifier [12].

For the second proposed method, we rely on the so-called Local Pattern Derivative on Three Ortogonal Planes (LDP-TOP) features [13], which is particularly effective in face anti-spoofing. In our approach, features are extracted from five different body parts: face, left hand, right hand, left foot and right foot. We propose to perform the analysis on video sequences by considering the prediction on single body parts, and then by combining these predictions. In this way, we focus on analyzing whether one or more body parts of the involved human subject have been synthesized or not, having at the same time a detailed and an overall view and of the veracity of the video sequence.

The features are extracted by two different pre-processed versions of the videos: (i) sequences obtained by simple grayscale conversion of the videos; (ii) sequences obtained by calculating the gradient of the optical flow extracted from the videos. The prediction on the sequences can be calculated by considering the two pre-processed versions separately; nonetheless, we also propose a combined technique where we jointly exploit the two pre-processed versions to compute the video prediction.

To cope with the lack of a comprehensive and publicly available dataset of videos with synthesized human bodies, we design our dataset. The original videos come from different sources (i.e. publicly available videos, videos taken from YouTube and our camera recordings), while the synthesized ones are generated using the video-to-video translation algo-

rithm *Everybody Dance Now* [6]. We compressed and resized each videos of the dataset using different level of compression and resolutions, in order to test the robustness of the proposed method on video compression and resizing.

The experiments we have conducted demonstrate the validity of both our proposed methods, achieving high levels of accuracy. The feature-based method proved to be robust to compression and resizing. In addition, unlike the CNN-based method, this method proved to be much more robust to the number of frames considered in the training dataset.

This thesis is organized as follows. In Chapter 2, we cover the background concepts which this work is based on, that will be necessary to understand the topics addressed in the thesis. In Chapter 3, we describe the video-to-video motion transfer algorithm used to synthesized videos, and we present the problem addressed in this thesis. In Chapter 4, we present the state of art in video *deepfakes* detection. In Chapter 5, we report our proposed methodology. In Chapter 6, we describe our dataset, the experimental settings and we report the results achieved in our experiments. Finally, conclusions and possible future directions are drawn in Chapter 7.

2

Theoretical Background

This chapter introduces the readers to the fundamental concepts that are necessary to understand the topics covered in this thesis. In particular, we focus on the concept of neural network, briefly discussing about the structure of the networks we use for our task; then, we take a look at the most famous textural features; in the end, we deal with the concept of optical flow and we summarize the functioning of a neural network that implements it.

2.1 Neural Networks

Neural Networks (NNs), also known as artificial neural networks, are a computational learning system that use a network of functions to understand and translate a data input of one form into a desired output, usually in another form [14, 15, 16]. NNs are a subset of machine learning algorithms and are at the heart of deep learning. The concept of NN was inspired by human biology and the way neurons of the human brain function together to understand inputs from human senses.

Machine learning and deep learning algorithms that use NNs generally do not need to be programmed with specific rules designed by humans that define what to expect from the input. The learning algorithm of the network, instead, learns from processing many examples that are

supplied during the training phase. Once a sufficient number of examples has been processed, the NN can begin to process new, unseen inputs and successfully return accurate results. The more examples and variety of inputs the program sees, the more accurate the results typically become because the program learns with experience. In the construction of a NN, a typical dataset is usually split in this way:

1. Training set: dataset from which the network learns a generalization for the assigned task;
2. Validation set: dataset used to give an estimate of model skill while tuning model's inner parameters;
3. Testing set: dataset used to test the network after it has been trained on an initial training set.

The typical approaches followed by NNs, and in general by all the machine learning algorithms, can be divided into two main groups:

1. Supervised learning;
2. Unsupervised learning.

The main distinction between the two approaches is that supervised learning uses labeled input data (i.e. we already know what the corresponding output is), while an unsupervised learning algorithm does not, so it works on its own to discover the inherent structure of unlabeled data. Supervised learning can be separated into two types of problems: classification and regression. Unsupervised learning models are used for three main tasks: clustering, association and dimensionality reduction. In this thesis, we focus on the supervised learning approach.

2.1.1 Multi-Layer Perceptron

In this section we restrict our attention to the specific class of NNs that have proven to be of greatest practical value, namely the MLP. But first, let us explain what a Perceptron is.

Perceptron, introduced by Rosenblatt in 1958 [17], is one of the first NN models introduced in the literature. This model relies on the basic unit of computation of the neuron, indicated as the $j-$node, and can be represented by the following mathematical formulation:

$$z_j = g_j(\sum_{i=1}^{n} w_{ij} s_j - b_j) \tag{2.1}$$

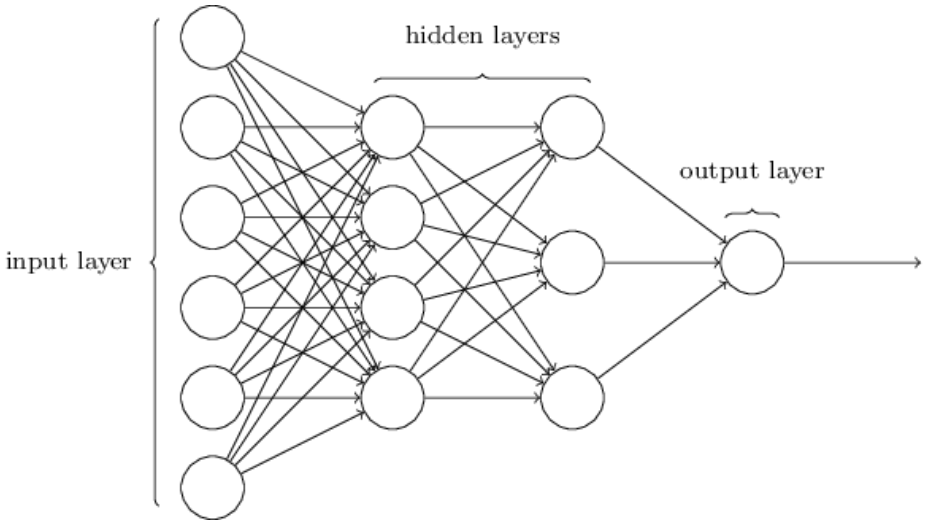

Figure 2.1: Representation of a generic Multi-Layer Perceptron (MLP) [1].

where s_j is the input data, b_j is the activation threshold (bias), w_{ij} are the weights, g_j is the activation function and z_j is the output data. Initially, weights and bias are assigned randomly or with a specific criterion. These weights help determine the importance of any given variable, with larger ones contributing more significantly to the output compared to the others. In literature we have a lot of different activation functions. As an example, we report the step function, which introduces a discontinuous nonlinearity between the input and the output:

$$g_j(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases} \tag{2.2}$$

The goal of the activation function is to allow information to pass from one neuron to another. This process of passing data from one layer to the next layer defines this NN as a feedforward network.

Stacking several perceptrons together, we obtain the MLP, which is a NN able to approximate any smooth and measurable function between an input and an output [18].

In Figure 2.1 we can see the general architecture of MLP: it is composed by an input and an output layer, and one or more hidden layers. Layers introduce continuous nonlinearities between input and output through a proper activation function. The most common activation

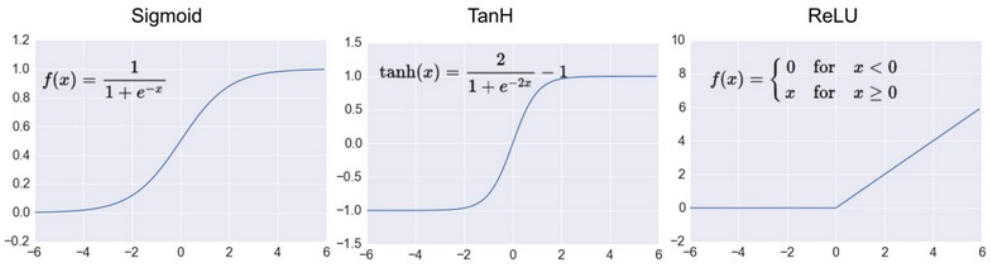

Figure 2.2: Activation functions.

functions are Rectified Linear Unit (ReLU), Sigmoid and Tanh, whose mathematical formulas and graphs are described in Figure 2.2.

2.1.2 Training and Loss function

In order to objectively evaluate the quality of the NN model, we need to define the loss function. If the model predictions are closer to the actual values, the loss will be minimum and if the predictions are totally away from the original values, the loss value will be high. A very common loss function is the Mean Square Error (MSE) defined as follows:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2.3}$$

where N is the number of samples; y_i is the outcome associated to a particular input data x_i that the network wants to predict; $\hat{y}_i$ is the predicted outcome from x_i.

The goal is to minimize the value of the loss function to ensure correctness of fit for any given observation. The output predicted by the network depends on its inner parameters (weights and biases), so we need to update them. The weights of a NN are initialized randomly or with a certain criterion. So, at the beginning, the network does not works accurately.

The problem of training is equivalent to the problem of minimizing the loss function by updating the parameters of the network. This process is called *backpropagation* [19] and it is defined as follows:

$$\mathbf{w}^{n+1} = \mathbf{w}^n + \Delta\mathbf{w} = \mathbf{w}^n - \eta \cdot \frac{\partial \mathcal{L}}{\partial \mathbf{w}}\bigg|_{\mathbf{w}=\mathbf{w}^n} \tag{2.4}$$

where $\mathbf{w}^n$ indicates a certain weight and bias parameter of the network at the time step n, $\mathbf{w}^{n+1}$ is the updated parameter at the step $n+1$, $\Delta\mathbf{w}$ is the learning step, η is the learning rate and $\mathcal{L}$ is the loss function.

The growth of the loss function is indicated by its derivative. When we update the parameter, we want that its growth goes in the opposite direction to the growth of the loss function. That is the reason why we add the minus sign in front of the derivative. The learning rate η controls the amount of weight increase: we want to proceed in small increments to avoid exceeding the minimum point and witness an increase in the loss function value. So, the learning step is composed of the derivative with opposite sign (to define the increase or decrease of the parameter) multiplied by a very small number, the learning rate (to proceed gradually in decreasing the value of the loss function).

Now, suppose we want to check how well our NN model learns and generalizes to the new data. The two main issues are: overfitting and underfitting. Overfitting occurs when the network learns a very complex function that works perfectly on the training data, but badly on test data. Underfitting occurs when the network is unable to capture the relationship between the input and output variables accurately because it learns a very simple function. In order to face these issues, it is not possible to define an universal method able to avoid them, but there are many techniques to reduce overfitting and underfitting, like regularization or crossvalidation, that let us build the optimal network for our task [12].

2.1.3 Convolutional Neural Networks

One of the most popular deep NN is the Convolutional Neural Network (CNN) [11]. This architecture has had ground breaking results over the past decade in many computer vision tasks and in particular in image classification.

An image is a matrix of pixel values. So, one can think of just flattening the image and feeding it to a MLP network for classification purposes. The input layer of MLP has one node for each pixel of the image, and so, the larger the image, the more complex the network and the greater the risk of overfitting. In cases of extremely basic binary images, the method might show an average precision score while performing prediction of classes, but would have little to no accuracy when it comes to complex images. The most beneficial aspect of CNNs is the low number of parameters involved in the network. Abstract features with lower dimensionality are extracted when input propagates toward the deeper layers. For example, in image classification, the edge might be detected in the first layers, and then the simpler shapes in the second layers, and then the higher level features such as faces in the next layers.

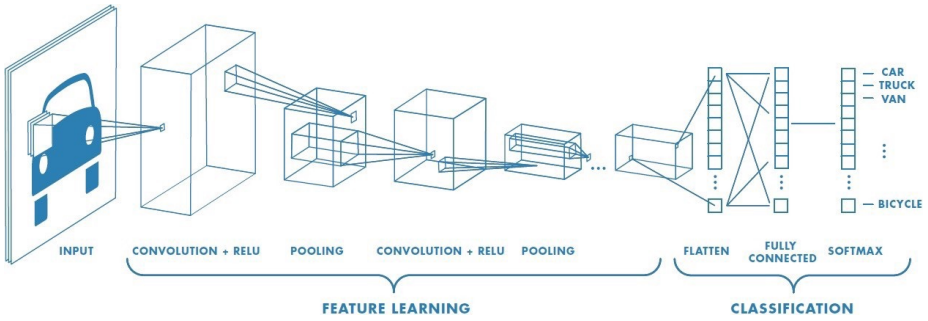

Figure 2.3: Representation of a CNN [2].

A CNN network is mainly made up of layers called convolutional layers; these layers are able to successfully capture the spatial and temporal dependencies in an image through the application of relevant convolutional filters, from which its name comes from. Convolution is a mathematical operation, that for a 2D input image is defined as follows:

$$\mathbf{Y}(w,h) = \mathbf{W} * \mathbf{X}(w,h) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} \mathbf{W}(s,t)\mathbf{X}(w-s,h-t) \qquad (2.5)$$

where $\mathbf{Y}$ is the output image obtained from convolution, $\mathbf{X}$ is the original input image and $\mathbf{W}$ is a weight matrix called kernel. In Figure 2.3 we can see the general architecture of a CNN. The main task of the first part of the network is feature extraction: this part is composed by several convolutional layers followed by ReLU activation functions, and pooling layers which are responsible for reducing the spatial size of the inner features. There are two types of pooling: Max Pooling and Average Pooling. Max pooling returns the maximum value from the portion of the image covered by the kernel. Average pooling returns the average of all the values from the portion of the image covered by the kernel. The main task of the second part of the network is classification. The output of the previous layer is flatten into a column vector which is fed into a fully-connected layer that learns non-linear combinations of the high-level features.

As explained in Section 2.1.2, in the training phase we need to compute a loss function and adjust weights and bias of each layer. In convolutional layers, weights are the elements inside kernels.

2.1.4 Generative Adversarial Networks

Generative Adversarial Network (GAN) is a deep learning architecture which works as a generative model for image synthesis [20]. Proposed in 2014 by Goodfellow et al [21], it is characterized by training a pair of networks in competition with each other: in literature, we call these networks as generator (G) and discriminator (D). Given a training set, this architecture learns to generate new data with the same statistics as the training set. For example, a GAN trained on photographs can generate new photographs that look at least superficially authentic to human observers, having many realistic characteristics. The networks that represent the generator and discriminator are typically implemented by multi-layer networks consisting of convolutional and fully-connected layers.

Formally, let us define the function implemented by the generator as:

$$G : G(\mathbf{z}) \rightarrow R^{|\mathbf{x}|} \tag{2.6}$$

where $\mathbf{z} \in R^{|\mathbf{z}|}$ is a sample taken from some representation space, called latent space: this sample can be considered a noise source because it can come from whatever distribution of data; $\mathbf{x} \in R^{|\mathbf{x}|}$ is the generated output and $|\cdot|$ denotes the number of dimensions.

The function implemented by the discriminator network can be formally defined as a function that maps from image data to a probability that the image is from the real data distribution, rather than the generator distribution:

$$D : D(\mathbf{x}) \rightarrow (0, 1) \tag{2.7}$$

where 0 means that the input image $\mathbf{x}$ comes from real data distribution, and 1 that it has been generated by G. If the generator manages to generate perfect images that are practically indistinguishable from the real ones, the discriminator will be maximally confused, predicting 0.5 for all inputs.

We can see a general overview of the GAN architecture in Figure 2.4. As we can see, the generator has no direct access to real images, while the discriminator has access to both the real and fake samples. The error signal to the discriminator is provided through the simple ground truth of knowing whether the image came from the real stack or from the generator. The same error signal, via the discriminator, can be used to train the generator, leading it towards being able to produce forgeries of better quality.

The training phase of GANs involves both finding the parameters of

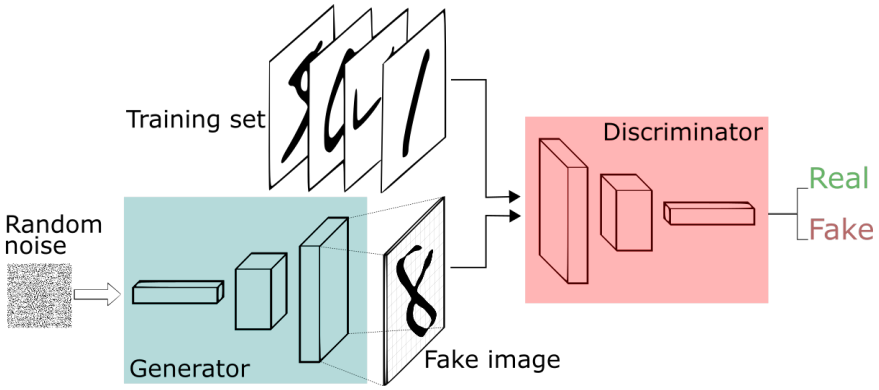

Figure 2.4: Overview of a GAN architecture [3].

a discriminator that maximize its classification accuracy, and finding the parameters of a generator which maximally confuse the discriminator, which means solving the following two-player min-max game:

$$\max_D \min_G \Big(\mathbb{E}_{\mathbf{x}} \log D(\mathbf{x}) + \mathbb{E}_{\mathbf{z}} \log(1 - D(\mathbf{z})) \Big) \tag{2.8}$$

where with the generic notation $\mathbb{E}_x f(x)$ we indicate the expected value of the function $f(x)$ assuming that the random variable x is distributed with respect to the probability distribution $p(x)$, and it is defined as follows.

$$\mathbb{E}_x f(x) = \int f(x) p(x) dx \tag{2.9}$$

During training, the parameters of one model are updated, while the parameters of the other are fixed. Goodfellow et al. [21] show that for a fixed generator there is a unique optimal discriminator. Ideally, the discriminator is trained until optimal with regard to the current generator; then, the generator is again updated. However in practice, the discriminator might not be trained until optimal, but rather may only be trained for a small number of iterations, and the generator is updated simultaneously with the discriminator. Despite the theoretical existence of unique solutions, GAN training is challenging and often unstable for several reasons [22] [23] [24].

In literature there are different variants of the GAN framework, each of them built to solve a certain type of task: classification, regression, image synthesis, image-to-image translation and super-resolution [20]. One of the most famous variant architecture is the Conditional GAN, introduced by Mirza et al. [25], where both the generator and discriminator

are conditioned on some sort of auxiliary information, such as class labels or data.

2.2 Textural Features

In this section we describe the Local Pattern Derivative on Three Orthogonal Planes (LDP-TOP), a handcrafted textural feature proved to be particularly effective in face anti-spoofing. This feature is a generalization of the widely used Local Binary Pattern (LBP). Let us start introducing the main concepts behind the idea of LBP in Section 2.2.1, and then we discuss LDP-TOP in Section 2.2.2, which is the feature we have used in our experiments.

2.2.1 Local Binary Pattern feature

LBP is defined as a grayscale invariant texture measure and is an useful tool to model texture images [26] [27] [28]. The LBP operator labels the pixels of an image $\mathbf{I}$, considered as a 2D array of pixel, by thresholding the 3×3 neighborhood of each pixel with the value of the central pixel and concatenating the results binomially to form a number. This operator can be formalized with the following threshold function:

$$f(\mathbf{I}(\mathbf{z}_0), \mathbf{I}(\mathbf{z}_i)) = \begin{cases} 0, & \text{if } \mathbf{I}(\mathbf{z}_i) - \mathbf{I}(\mathbf{z}_0) <= threshold \\ 1, & \text{if } \mathbf{I}(\mathbf{z}_i) - \mathbf{I}(\mathbf{z}_0) > threshold \end{cases} \tag{2.10}$$

with $i = 1, 2, ..., 8$, and $\mathbf{z}_i$ is the i-th neighborhood 2D coordinates point around $\mathbf{z}_0$ as shown in Figure 2.5. Then, the binary values are concatenated in a 8-bit vector called micropattern. The results of this binary function are stored in an 8-bit array called micropattern. Figure 2.6 shows an example of obtaining a LBP micropattern. The 2^8-bin histogram of these micropatterns extracted from an image is the feature that we consider: it contains information of the distribution of the edges, spots, and other local features in an image.

2.2.2 LDP-TOP feature

LDP-TOP is a spatio-temporal textural feature: it aims to exploit both spatial and temporal domains in the analysis of video sequences [13] [29]. It is an evolution of Local Pattern Derivative (LDP), which considers only the spatial domain [30].

$\mathbf{z}_1$	$\mathbf{z}_2$	$\mathbf{z}_3$
$\mathbf{z}_8$	$\mathbf{z}_0$	$\mathbf{z}_4$
$\mathbf{z}_7$	$\mathbf{z}_6$	$\mathbf{z}_5$

Figure 2.5: Example of 8-neighborhood around $\mathbf{z}_0$.

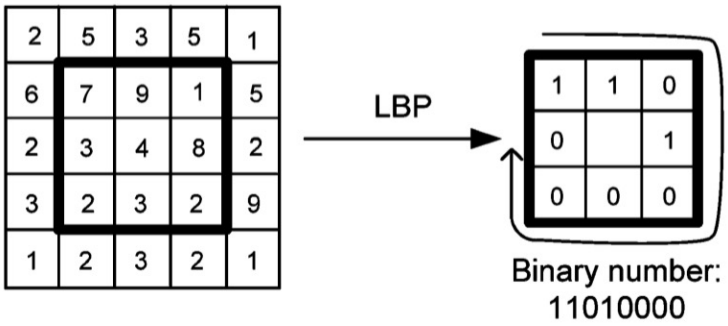

Figure 2.6: Example of how LBP micropattern is calculated for the region in the black square.

LDP is a point-wise operator applied to an image, considered as a 2D array of pixels, that encodes diverse local spatial relationships. As suggested in [31], we consider the second-order directional LDPs with direction α, indicated as LDP^2_α, with $\alpha \in \{0°, 45°, 90°, 135°\}$. Given a 2D array of pixels, the LDP^2_α at the location (h, w) is an 8-bit vector defined as:

$$\begin{aligned}\text{LDP}^2_\alpha(h,w) =[&f(\mathbf{I}'_\alpha(h,w),\mathbf{I}'_\alpha(h^-,w^-)), f(\mathbf{I}'_\alpha(h,w),\mathbf{I}'_\alpha(h^-,w)),\\ &f(\mathbf{I}'_\alpha(h,w),\mathbf{I}'_\alpha(h^-,w^+)), f(\mathbf{I}'_\alpha(h,w),\mathbf{I}'_\alpha(h,w^+)),\\ &f(\mathbf{I}'_\alpha(h,w),\mathbf{I}'_\alpha(h^+,w^+)), f(\mathbf{I}'_\alpha(h,w),\mathbf{I}'_\alpha(h^+,w)),\\ &f(\mathbf{I}'_\alpha(h,w),\mathbf{I}'_\alpha(h^+,w^-)), f(\mathbf{I}'_\alpha(h,w),\mathbf{I}'_\alpha(h,w^-))]\end{aligned} \tag{2.11}$$

with $h^+ := h+1, h^- := h-1$ and $w^+ := w+1, w^- := w-1$. The operator $\mathbf{I}'_\alpha(\cdot)$ is the first-order derivative in the direction α, and is defined pixel-

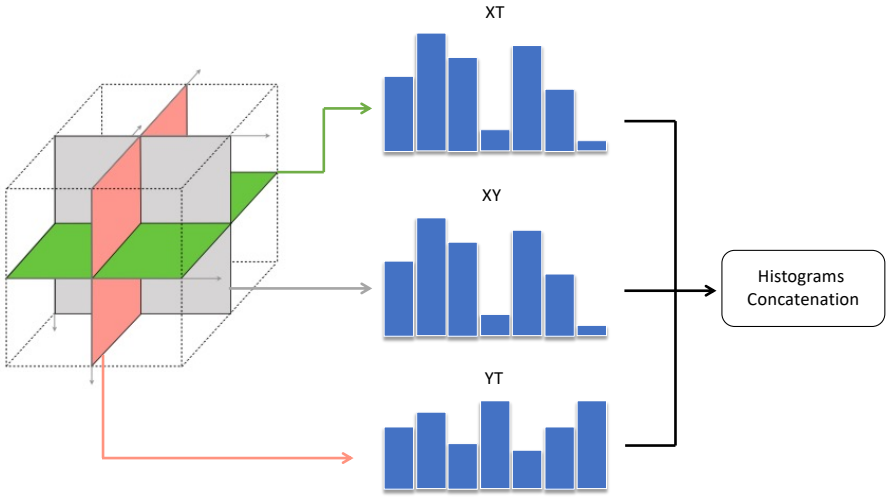

Figure 2.7: Overview of the LDP-TOP feature construction.

wise as:

$$\mathbf{I}'_{\alpha}(h,w) = \begin{cases} \mathbf{I}(h,w) - \mathbf{I}(h,w^{+}) & \text{if } \alpha = 0° \\ \mathbf{I}(h,w) - \mathbf{I}(h^{-},w^{+}) & \text{if } \alpha = 45° \\ \mathbf{I}(h,w) - \mathbf{I}(h^{-},w) & \text{if } \alpha = 90° \\ \mathbf{I}(h,w) - \mathbf{I}(h^{-},w^{-}) & \text{if } \alpha = 135° \end{cases} \tag{2.12}$$

where $\mathbf{I}(\cdot)$ indicates the intensity value of the pixel at the given coordinates, while the thresholding function $f(\cdot)$ is defined as:

$$f(a,b) = \begin{cases} 0 & \text{if } a \cdot b > 0 \\ 1 & \text{if } a \cdot b \leq 0 \end{cases} \tag{2.13}$$

So, differently from LBP, in this case we compare the first-order derivative in direction α of the pixel at the position (h,w) with all the other first-order derivatives in direction α of the pixels in its 3×3 neighborhood. We calculate LDP^2_{α} for each pixel in the image, and then their 2^8-bin histogram is computed: this is replicated for the four different directions indicated by α, and the histograms are concatenated.

LDP-TOP extends this computation by considering a temporal dimension, so we don't consider only a single image but a sequence of images. Given a 3D sequence of images, we consider the three central 2D arrays along each dimension that intersect orthogonally (that we call **XY**, **YT**, **XT**) and again concatenating the three obtained histograms. In Figure 2.7 we can see an overview of this procedure.

Considering 4 derivative directions and three 2D arrays, the feature vector length is equal to $2^8 \times 4 \times 3 = 3072$.

The sequence of images can be processed in three different temporal modes:

1. *Direct* mode: the sequence is processed forward along the temporal direction.

2. *Inverse* mode: the sequence is processed backward along the temporal direction starting from the last frame.

3. *Bidirectional* mode: the sequence is processed in both directions and histograms are concatenated, yielding a feature vector with doubled size.

2.3 Optical Flow

In this section we describe the concept of optical flow [32]. Optical flow is a task of per-pixel motion estimation between two consecutive frames in one video. The aim of this task is to retrieve the projection on the image plane of the 3D motion in the space, usually called motion field. There are two different conventions to visualize the pixel motion of an image, as shown in Figure 2.8. The first one uses one arrow per pixel, with varying length and direction, depending on the intensity and the direction of the movement. The second one instead, assigns a color to each pixel. Colors near the center of the square represent very small motions, while the ones at the boundaries represent large displacements. Furthermore, the position of the color in the square specifies the direction of the flow.

In literature, we have a lot of works in the field of optical flow estimations. In the following section we describe a state-of-the-art NN architecture, able to estimate the optical flow, that we used for our task.

2.3.1 Flownet 2.0 Neural Network

Flownet 2.0 is a CNN architecture able to directly learn the concept of optical flow from data [5]: given a dataset consisting of image pairs and ground truth flows, the network is trained in order to predict the optical flow fields directly from the images. It is the evolution of the Flownet architecture introduced by Dosovitskiy et al [33]. This model takes a pair of images as input and outputs the optical flow. As we can see in Figure 2.9, Flownet 2.0 is built by stacking multiple CNN networks:

- *FlownetC*: this network is an encoder-decoder architecture. In the contracting part we have two separate, yet identical streams for

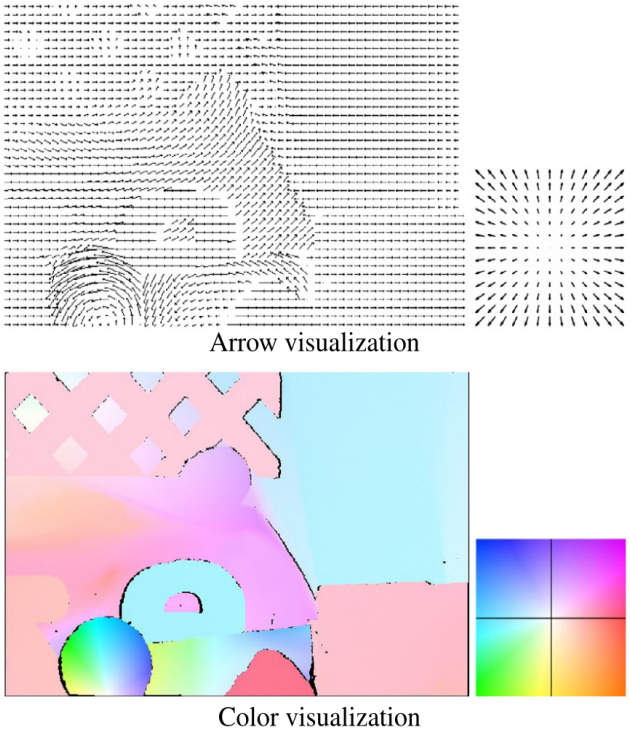

Figure 2.8: Two types of visualizations of the motion field between two consecutive frames [4].

the two input images. The two streams extract high level features from the input images, and then a correlation layer performs multiplicative patch comparisons between the two feature maps. In the expanding part, we have layers consisting of unpooling and upconvolution, able to increase the resolution of the obtained feature map. In the end we have a refinement part, where the feature maps are concatenated with the corresponding feature maps of the contractive part and an upsampled coarse flow prediction. In this way we preserve both the high-level information passed from coarser feature maps and fine local information provided in lower layer feature maps.

- *FlownetS*: is a straightforward encoder-decoder architecture composed by convolutional layers. This network works similarly to *FlownetC*, but in the contracting part we have only one input stream, which takes as input a 3D array built by stacking together two images.

- *Flownet-SD*: it is a network able to detect optical flow when there is a small displacement between consecutive frames. It is similar to the FlownetS architecture, but it has been trained by the authors

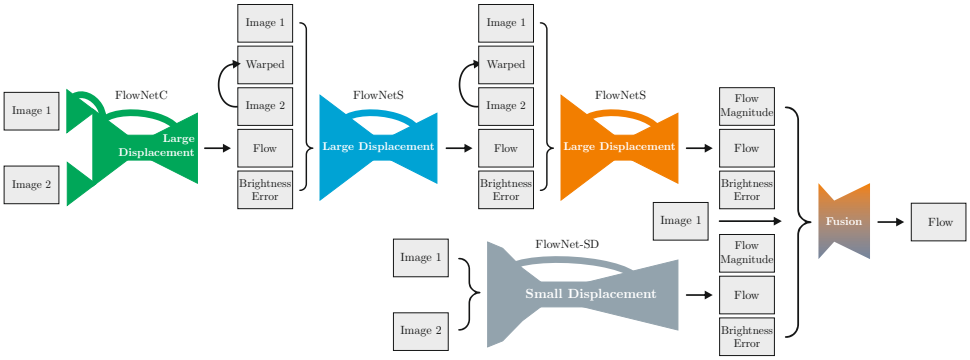

Figure 2.9: Schematic view of complete architecture of Flownet2 [5].

on a small dataset with very small displacements.

- *FlownetCSS*: it is obtained by stacking *FlownetC* and two *FlownetS*. The first network takes as input two images. The last two networks take as input the two images and the previous optical flow estimate. To make assessment of the previous error and computing an incremental update easier for the network, the second image is optionally warped via the optical flow with a bilinear interpolation. When the warping is computed, the network takes as input the warped image and the difference of the warped image and the first image (brightness error).

- *Fusion*: it is a small network that fuses *FlownetCSS* and *FlownetSD*. The network receives as input the estimated optical flows, the flow magnitudes and the brightness errors. It contracts the resolution two times by a factor of 2 and then expands to the full resolution.

The experiments on motion segmentation and action recognition done by Ilg et al. [5] show that the estimated optical flow with FlowNet 2.0 is reliable on a large variety of scenes and applications.

2.4 Conclusive Remarks

In this chapter we have introduced the basic concepts useful to understand the next part of the thesis. First, we have shown what a NN is and how it works, including the general concepts behind the training phase

and the loss function. We have described the structure of three NNs: MLPs, CNNs, and GANs. Then, we have described the handcrafted textural feature we use in the thesis, and how it is calculated. Finally, we have explained the main concepts behind the concept of optical flow, and a NN able to extract this kind of feature from the input video.

3
Problem Statement

In this chapter we describe and formalize the main purpose of this thesis, i.e. to provide a forensic tool able to discern between original and fake videos and how with deal with it from a high level point of view.

Before digging into the description of our problem, we describe the kind of videos we work on and we summarize the deepfake generator algorithm *Everybody Dance Now* [6], used to create our dataset of synthesized videos.

3.1 Everybody Dance Now Algorithm

In this section, we describe the motion transfer algorithm *Everybody Dance Now*, introduced by Chan et al. in [6], that we use to generate our synthesized videos. Given two videos, one of a target person whose appearance we wish to synthesize, and the other of a source subject whose motion we wish to impose onto our target person, the algorithm transfers motion between these subjects by learning a video-to-video translation. To accomplish this task, we need to follow a training and a transfer phase:

- *Training phase*: the model uses a pose detector P to create pose stick figures from the video frames of the target subject. Then, by using a conditional Generative Adversarial Network (GAN) model,

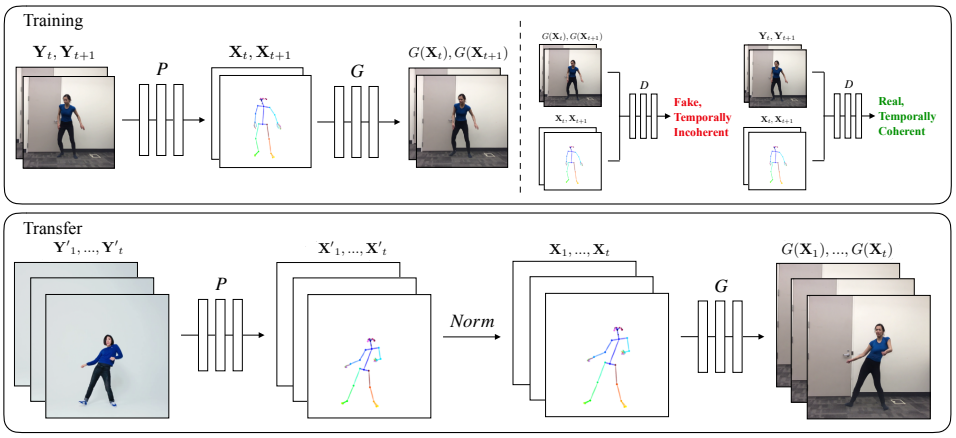

Figure 3.1: *Everybody Dance Now* algorithm overwiew [6].

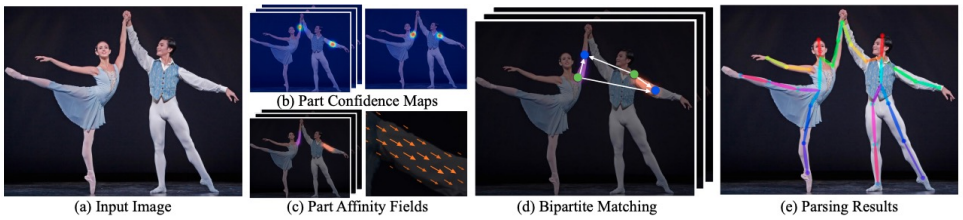

Figure 3.2: *Openpose* pipeline [7].

it learns a mapping G alongside an adversarial discriminator D which attempts to distinguish between real and fake sequences.

- *Transfer phase*: the model uses the pose detector P to extract the pose from the source person. Then, the source poses are normalized to match the target person appearance. In the end, we apply the trained mapping G to the normalized pose stick figures.

In Section 3.1.1 we describe the pose detector P used to extracts the pose; in Section 3.1.2, we explain how the pose normalization works; in Section 3.1.3, we describe the video synthesis model. In Figure 3.1, we show an overview of the *Everybody Dance Now* pipeline.

3.1.1 Pose Detection

To encode the body pose of a subject in a video, the authors of *Everybody Dance* Now use *Openpose*, which is an open-source library for multi-person pose detection, including body, foot, hand and facial keypoints [7]. *Openpose* is based on an algorithm which takes as input a color image

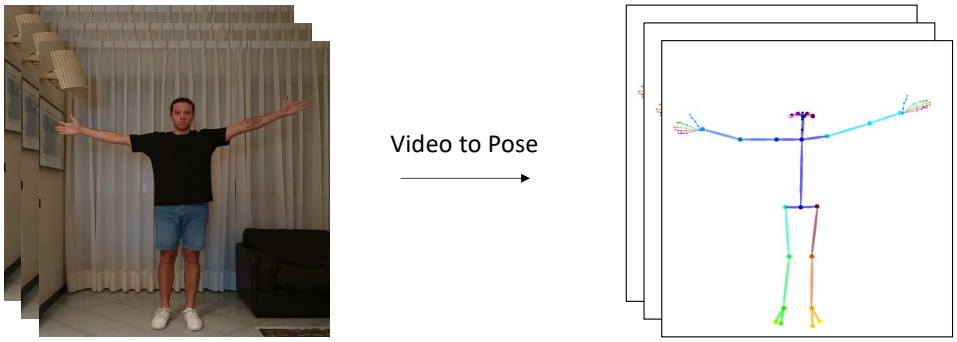

Figure 3.3: Correspondence between the subject appearing in the video frames and the detected poses.

and produces the 2D locations of anatomical keypoints for each person in the image. The algorithm follows a bottom-up approach, which means that it starts detecting single generic body parts and then associate them to a proper subject in an image. This kind of approach offers robustness to early commitment and save a lot of runtime complexity compared to top-down approach. Let us explain how this algorithm works.

First, we have a feedforward network which predicts a set of confidence maps and a set of Part Affinity Field (PAF). Confidence map is the 2D representation of the belief that a particular body part can be located. The number of confidence maps is the same as the total number of the body parts. PAFs encodes the degree of association between the parts. Each PAF is a 2D vector which encodes the direction that points from one part of the limb to the other. Then, confidence maps and PAFs are used to find the optimal association between the detected body parts in order to associate them with the subject they belong to: this reduces to a maximum weight bipartite graph problem [34]. In this way, a set of keypoints for each person in the image is obtained. In Figure 3.2 we show the general pipeline of the pose detector.

Then, a colored pose stick figure is created by plotting the keypoints and drawing lines between connected joints, as shown in Figure 3.3. This image is the corresponding label of the frame, used during the video synthesis process described in Section 3.1.3.

3.1.2 Pose Normalization

In this section we describe the pose normalization method introduced by the authors of *Everybody Dance Now* [6]. In different videos, subjects may have different limb proportions or stand in a different position in the scene captured by the camera. Therefore, it may be necessary to

transform the pose keypoints of the source person so that they appear in accordance with the target person's body shape and location. If we don't apply this transformation, we may generate images of the target person which are not congruent with the scene, and the overall quality of the synthesis is expected to decline. The transformation is parametrized in terms of a translation and a scale factor applied to all pose keypoints for a given frame. Let us explain how to get these parameters.

First, we indicate the closest and farthest distance in the 2D plane of the source subject from the camera as $\mathbf{s}_{close}$ and $\mathbf{s}_{far}$ respectively, and we indicate the same positions of the target subjects with $\mathbf{t}_{close}$ and $\mathbf{t}_{far}$, respectively. These distances are estimated by considering the positions of the subject's ankles in all frames of the video. Then, we need a translation factor that let us map the close and far range of the source to that of the target subject. The translation factor $\boldsymbol{\alpha}$ is defined as:

$$\boldsymbol{\alpha} = \mathbf{t}_{far} + \frac{\mathbf{p} - \mathbf{s}_{far}}{\mathbf{s}_{close} - \mathbf{s}_{far}}(\mathbf{t}_{close} - \mathbf{t}_{far}) \tag{3.1}$$

where $\mathbf{p}$ is the position of the source subject in the frame, calculated as the average of the subject's ankles coordinates.

Then, the heights of each subject are calculated at their closest and farthest positions in their video: $h_{\mathbf{s}_{close}}$ and $h_{\mathbf{s}_{far}}$ are the heights of the source, $h_{\mathbf{t}_{close}}$ and $h_{\mathbf{t}_{far}}$ are the heights of the target subject. The height of a subject in a certain frame is calculated as the euclidean distance between the average ankles positions and the nose keypoint. We determine separate scales for the close positions given by $c_{close} = \frac{h_{\mathbf{t}_{close}}}{h_{\mathbf{s}_{close}}}$, and similarly for the far position $c_{far} = \frac{h_{\mathbf{t}_{far}}}{h_{\mathbf{s}_{far}}}$. Now, let us define the scale factor $\boldsymbol{\beta}$ as:

$$\boldsymbol{\beta} = c_{far} + \frac{\mathbf{p} - \mathbf{s}_{far}}{\mathbf{s}_{close} - \mathbf{s}_{far}}(c_{close} - c_{far}) \tag{3.2}$$

After the translation and scale factors have been determined for a given source pose, we use these parameters to translate and scale the position of the target pose keypoints.

$$\mathbf{y} = (\mathbf{x} \cdot \boldsymbol{\beta}) + \boldsymbol{\alpha}. \tag{3.3}$$

where $\mathbf{x}$ indicates the 2D coordinates of a certain keypoint of the target subject before the transformation, and $\mathbf{y}$ indicates the 2D coordinates of the normalized keypoint.

3.1.3 Video Synthesis Model

The video synthesis method is based on the pix2pixHD model introduced by Wang et al. [8], which is an improvement of the conditional

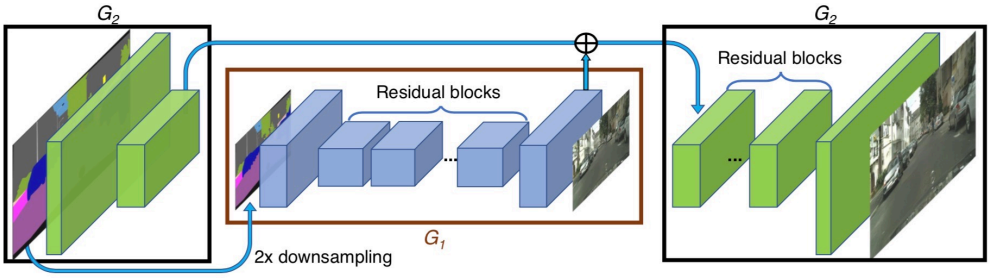

Figure 3.4: Network architecture of the generators of the pix2pixHD model [8].

GAN framework for image-to-image translation called pix2pix [35]. The model consists in a generator G and a discriminator D which are trained simultaneously and drive each other to improve: G learns to synthesize images of a person given a pose stick figure, while D learns differences between the generated outputs and the ground truth data.

Let us start describing the pix2pixHD framework, and then we discuss the modifications introduced by *Everybody Dance Now* model. The framework uses a coarse-to-fine generator, a multi-scale discriminator architecture, and a robust adversarial learning objective function. The generator G is decomposed in two sub-networks: a global generator G_1 and a local enhancer G_2, which are both based on the architecture proposed by Johnson et al. [36]. The local enhancer operates at a resolution which is 4 times the output size of the global generator. The multi-resolution pipeline for high resolution image synthesis adopted by this framework is a well-established practice in the computer vision field ([37], [38]): in the first stage, G_1 is trained on lower resolution images, while in the second stage G_1 and G_2 are trained jointly on high resolution images. In Figure 3.4, the network architecture of the two generators is shown.

The discriminator designed for this model is built on a multi-scale architecture. We have 3 discriminators D_1, D_2, D_3 that have an identical network structure, but operate at different image scales: D_1 operates with full resolution images, D_2 and D_3 operate, respectively, with images downsampled by a factor of 2 and 4. The idea is to train the three discriminators separately, to differentiate real and fake images at 3 different scales. The discriminator which operates at the coarsest scale has the largest receptive field, and guides the generator to generate globally consistent image; the discriminator at the finest scale encourages the generator to produce finer details.

Now, let us talk about the modifications introduced in the *Everybody Dance Now* model. The image generator is modified to enforce

temporal coherence between adjacent frames. So, instead of generating individual frames, the model predicts two consecutive frames in this way: given the 2D image of the rendered pose stick $\mathbf{X}_t$ corresponding to the t-th frame $\mathbf{Y}_t$ in a video, the generation of the synthesized image $G(\mathbf{X}_t)$ is conditioned on $\mathbf{X}_t$ and the output of $G(\mathbf{X}_{t-1})$. The task of the discriminator is to determine the difference in realism and temporal coherence between the "real" sequence $(\mathbf{X}_{t-1}, \mathbf{X}_t, \mathbf{Y}_{t-1}, \mathbf{Y}_t)$ and the fake sequence $(\mathbf{X}_{t-1}, \mathbf{X}_t, G(\mathbf{X}_{t-1}), G(\mathbf{X}_t))$. The temporal smoothing changes are reflected in the updated GAN objective:

$$\begin{aligned}\mathcal{L}_{\text{smooth}}(G, D) =& \mathbb{E}_{(\mathbf{X},\mathbf{Y})}[\log D(\mathbf{X}_{t-1}, \mathbf{X}_t, \mathbf{Y}_{t-1}, \mathbf{Y}_t)] \\ &+ \mathbb{E}_{\mathbf{X}}[\log(1 - D(\mathbf{X}_{t-1}, \mathbf{X}_t, G(\mathbf{X}_{t-1}), G(\mathbf{X}_t)))]\end{aligned} \tag{3.4}$$

Then, a specialized GAN is added to increase the realism of the generated face region. After generating the full image of the scene with G, we train another generator G_f, which takes as input a smaller section of the image centered around the face (i.e. 128×128 patch centered around the nose keypoint) that we call $G(\mathbf{X})_{FC}$, and the pose stick figure obtained in the same way $\mathbf{X}_{FC}$. The output of G_f is a residual $\mathbf{R} = G_f(\mathbf{X}_{FC}, G(\mathbf{X})_{FC})$. The final synthesized face region is obtained by summing the residual with the face region of the main generator: $\mathbf{R} + G(\mathbf{X})_{FC}$. A discriminator D_f attempts to discern the "real" face pairs $(\mathbf{X}_{FC}, \mathbf{Y}_{FC})$ from the "fake" face pairs $(\mathbf{X}_{FC}, \mathbf{R} + G(\mathbf{X})_{FC})$, where $\mathbf{Y}_{FC}$ is the face region of the original pose stick figure $\mathbf{X}$. Let us introduce the following loss function:

$$\begin{aligned}\mathcal{L}_{\text{face}}(G_f, D_f) =& \mathbb{E}_{(\mathbf{X}_{FC},\mathbf{Y}_{FC})}[\log D_f(\mathbf{X}_{FC}, \mathbf{Y}_{FC})] \\ &+ \mathbb{E}_{\mathbf{X}_{FC}}[\log(1 - D_f(\mathbf{X}_{FC}, G(\mathbf{X})_{FC} + \mathbf{R}))]\end{aligned} \tag{3.5}$$

In Figure 3.5, we show an overview of the Face GAN model.

The main generator and discriminator are trained with the following full objective:

$$\begin{aligned}&\min_G((\max_{D_i} \sum_{k_i} \mathcal{L}_{\text{smooth}}(G, D_k) + \lambda_{FM} \sum_{k_i} \mathcal{L}_{FM}(G, D_k) \\ &+ \lambda_P(\mathcal{L}_P(G(\mathbf{X}_{t-1}), \mathbf{Y}_{t-1}) + \mathcal{L}_P(G(\mathbf{X}_t), \mathbf{Y}_t)))\end{aligned} \tag{3.6}$$

where:

- $i = 1, 2, 3$ is the index of the multi-resolution discriminators
- $\mathcal{L}_{FM}(G, D)$ is the discriminator feature-matching loss. The idea is to extract features from multiple layers of the discriminator and learn to match these intermediate representation from real and synthesized images. It has been proved that this loss stabilizes the training [8].

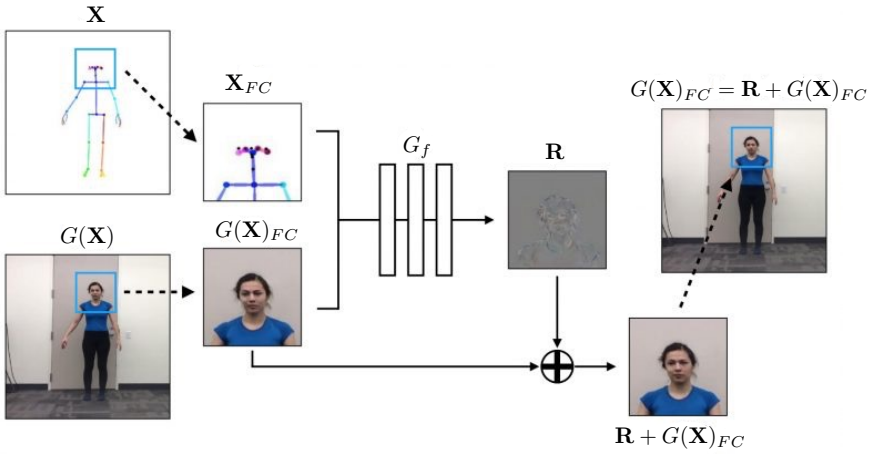

Figure 3.5: Face GAN pipeline overview.

- $\mathcal{L}_P(G(\mathbf{X}), D)$ is the perceptual reconstruction loss [36]
- λ_{FM} and λ_P are two hyperparameters.

As a last step, the weights of the full image GAN are optimized with the objective:

$$\min_{G_f}\left((\max_{D_f}\mathcal{L}_{\text{face}}(G_f, D_f)) + \lambda_P\mathcal{L}_P(\mathbf{R} + G(\mathbf{X})_{FC}, \mathbf{Y}_{FC})\right) \tag{3.7}$$

3.2 Problem Formulation

The problem analyzed in this thesis is the detection of authentic videos and tampered ones.

We focus on a certain kind of videos representing a scene in which we have a single subject moving within it, performing any kind of moves. The person is filmed by a fixed camera in a way that all parts of his body are clearly visible. Given this kind of footage, our goal is to detect whether a certain video is authentic or it has been synthetically generated by the *Everybody Dance Now* algorithm.

Formally, let us consider a video corresponding to a sequence of consecutive frames $\mathbf{V} = (\mathbf{F}_0, \mathbf{F}_1, ..., \mathbf{F}_N)$. The authenticity detection is a binary classification problem, which consists in defining the following function:

$$f : \mathbf{V} \rightarrow \hat{y} \tag{3.8}$$

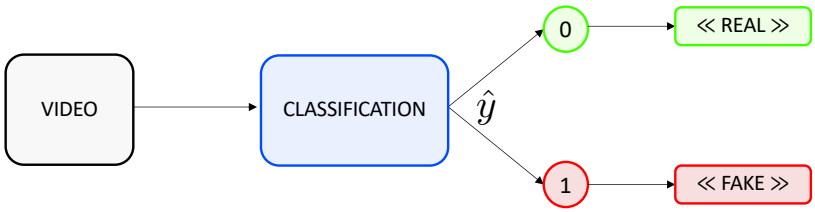

Figure 3.6: Formulation of the binary classification.

The function f is the binary classifier tool, which receives as input the video sequence $\mathbf{V}$ and it is able to predict as output the discrete label $\hat{y} \in \{0, 1\}$, where 0 means "real", and 1 means "fake". The goal of this thesis is to introduce and compare different classification functions f. In Figure 3.6 we report the pipeline of the entire process.

3.3 Conclusive Remarks

In this chapter we have described the video-to-video translation algorithm used to generate the videos analyzed in our thesis. Then, we have described how the video classification problem can be addressed.

4

State of the Art

Fake multimedia has become a central problem in the last few years, especially after the advent of the so called Deepfakes, i.e. fake media manipulated with the help of powerful and easy-to-use deep learning tools. These manipulated media may constitute a serious threat to attack the reputation of public subjects or to address the general opinion on a certain event. According to this, being able to individuate this kind of fake information becomes fundamental. In this section we will discuss some of the most recent deepfake detection methods able to discern between real and fake data [39].

4.1 Deepfake Detection Methods

Most deepfake generators rely on the creation of single images. To cope with this, several detection methods working at image-level have been introduced. Currently, deep learning-based attacks focus on two main targets: (i) the creation of fully synthetic images by means of Generative Adversarial Network (GAN)-based architectures; (ii) the manipulation of faces in videos aimed at changing identities or semantics. We have forensic methods for each of these approach.

Fully synthetic GAN-generated images can contain visible artifacts. For this reason some methods try to highlight specific failures in the gen-

eration process which does not reproduce perfectly all the details or lack to satisfy symmetry constraints. A method based on the construction of simple features to detect visual artifacts is exploited in [40]. A pixel-level detection of errors produced by the generation process is analysed in [41]. Some methods analyse inconsistencies in the way colours are synthesized [42] , [43]. Other works are based on the study of a specific fingerprint released by the GAN generator architecture [44] , [45], [46]. In another research study, a GAN simulator is proposed to reproduce common GAN-image artifacts, which manifest as spectral peaks in the Fourier domain [46]. Detection is done by training a classifier that takes as input this spectrum. Another work tries to attribute a generated image to a specific generator in a white-box scenario [47].

Several approaches adopt deep learning solutions in a supervised setting. In [48], different Convolutional Neural Network (CNN)-based architectures have been tested in order to understand whether an image is real or fake. In this work, classification accuracy drops with compressed images and when train and test set are not aligned in terms of compression level, showing lack of generalization ability. This problem is becoming a major issue, in particular considering the fast growing number of different manipulation methods. One of the first research that focuses on this problem propose an autoencoder-based architecture which adapts to new manipulations with just a few examples [49]. Other similar architectures that work in this direction are proposed in [50] [51]. In [52], a preprocessing step is introduced in order to reduce low level artifacts of GAN images and force the discriminator to learn more general forensic features. Another research proposes a solution that shows good generalization performance, introducing a CNN network that incorporates global image texture information in different semantic levels [53].

Manipulation of the faces is one of the most common deepfake techniques. For this reason, we have many forensic methods able to detect if a face is real or fake. In [54], detection relies on the spatial-temporal deformations of a 3D model that fits the face. In particular, natural faces follow more complex and various geometric deformations then synthetic ones, and cause higher perturbations of the 3D model.

We have also many methods devoted explicitly to detect face manipulations in videos: some of them are based on handcrafted features, other are based on deep learning techniques.

1. *Handcrafted solutions*: [55] studies the eye blinking of the subjects, which has a specific frequency and duration in humans difficult to replicate in synthesized videos. The solution is based on a long-

term recurrent network, working only on eye sequences, designed to catch such temporal inconsistencies. In [56], the detection is based on the analysis of biological signals of the subjects, analysed in both spatial and temporal directions. Other detection methods rely on face warping artifacts [57], face landmark locations [58] or head pose inconsistencies [59].

A clear advantage of all these methods is that visual artifacts are not affected by resizing and compression, while the disadvantage is that these manipulations can be recognized also by human viewers. In addition, it is very likely that the next-generation deepfakes will overcome such imperfections and synthesize more realistically.

The approach followed in [60] understands if a video has been manipulated or not, studying the correlation between the facial expression and the head movements. One of the latest researches presents a method that exploits both texture and temporal information of a video sequence, by extracting some hybrid face descriptors and classifying them through simple support vector machines [13].

2. *Deep Learning Solutions*: In [61], two simple CNN-based architectures are proposed with a small number of levels and parameters that exploits mesoscopic features. In [62] is shown that, in a supervised setting, very deep general-purpose neural networks outperforms forensics-oriented shallow CNNs, as well as methods based on handcrafted features, especially in the presence of the strong compression typical of video codecs. A lot of deep learning based methods work at the frame level [63], [64], [65], [66], [67].

   Methods that take into account also the temporal direction can achieve very good results. Long Short-Term Memory (LSTM) architectures are used to exploit such dependencies and improve upon single-frame analysis in [68] and [69]. In [70], it is proposed a solution based on the extraction of different kind of features which are fed into recurrent convolutional models. Another solution relies on the discrepancies of the estimated optical flow fields in original and synthesized videos [71]. In [72], it is presented a hierarchical neural memory network that exploits long-term dependencies, and achieve good generalization results. The approach followed in [73] is based on the observation that face boundaries exhibit traces of blending operation in case of forgery. In [74], generalization is gained by training the method only on pristine videos and by extracting the camera fingerprint information (noiseprint) gathered from multiple

frames.

4.2 Conclusive Remarks

As we can see, there are several video classification methods, but none of them focuses on analyzing if an entire body of a human subject is synthesized or not. This because only recently the first software capable of this kind of video synthesis have been released, and even if their synthesis quality is not perfect nowadays, we expect it to increase in the coming years. Anyway, our classification techniques are inspired by considering the most recent handcrafted solutions and deep learning solutions described in this chapter.

5 Proposed Methodology

In Chapter 3 we have explained what is the problem addressed in this thesis. Now let us see how to implement useful algorithms to solve this task, analyzing step by step our processing chain.

Before digging with the explanation of the proposed methodology, we describe the video pre-processing phase in Section 5.1, and the body parts area selection phase in Section 5.2 which selects the parts of the videos that we will consider in our method.

We propose two classification methods that find an estimate of the label of a video sequence ("real" or "fake"). First, we will describe a simple Convolutional Neural Network (CNN)-based classification approach in Section Section 5.3; this is considered as a comparison baseline for our proposed method in Section 5.4, where we introduce different strategies based on the extraction of handcrafted textural features. The general scheme of the proposed methodology is shown in Figure 5.1.

5.1 Video Pre-processing

First, let us consider a dataset of videos $\mathcal{V}$, composed of real videos and synthesized ones. We represent each video $\mathbf{V}$ in our dataset as a 4D matrix of shape $N \times C \times H \times W$, where N is the number of frames, C is the number of channels, and H and W are the height and width of

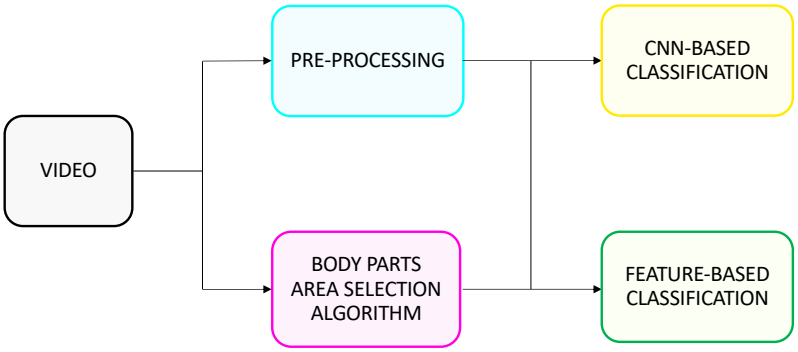

Figure 5.1: Method outline.

the single frame $\mathbf{F}$. Since we adopt the RGB representation, at this step, the value of C is equal to 3. As described in Figure 5.2, we propose to process the videos in two different ways:

- $\mathbf{V}^{(g)}$, which is the video $\mathbf{V}$ converted from RGB to grayscale colorspace, so $C = 1$.
- $\mathbf{V}^{(o)}$, which is the gradient of the optical flow extracted from $\mathbf{V}$.

Let us call the generic output video as $\mathbf{V}^{(\rho)}$, with the index $\rho = [g, o]$ indicates if we are considering $\mathbf{V}^{(g)}$ or $\mathbf{V}^{(o)}$.

Each frame of $\mathbf{V}^{(\rho)}$ is a 2D array of pixel of shape $H \times W$, where each pixel ranges in $[0, 255]$. In the Section 5.1.1 we will discuss about how to get $\mathbf{V}^{(o)}$.

5.1.1 Gradient of Optical Flow processing

We extract this kind of feature from the videos to enhance the edges of the movement of the subjects, information contained in the optical flow, and use them for our classification task.

Given $\mathbf{V} \in \mathcal{V}$, we estimate the optical flow with the Flownet 2.0 neural network (Section 2.3.1), and we apply a RGB to grayscale colorspace conversion, obtaining $\mathbf{V}_{flow}$. Now, we need to define the gradient function defined as:

$$\mathbf{V}^{(o)} = \Delta(\mathbf{V}_{flow}) = \frac{\partial^2(\mathbf{V}_{flow})}{\partial x^2} + \frac{\partial^2(\mathbf{V}_{flow})}{\partial y^2} \tag{5.1}$$

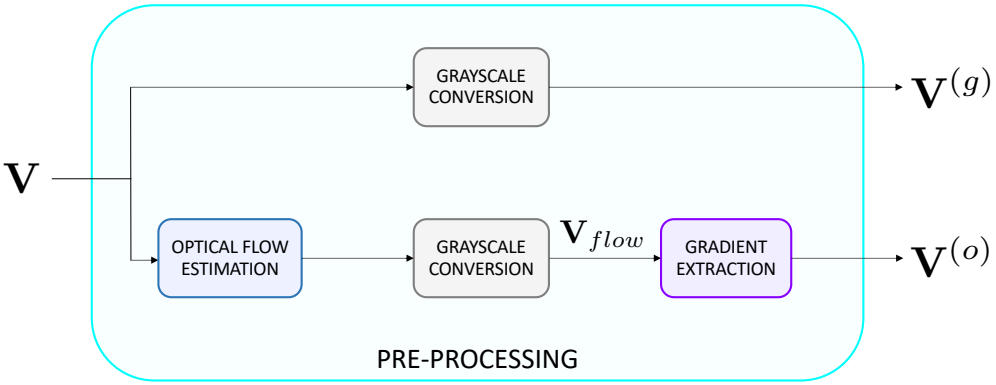

Figure 5.2: Pre-processing outline.

where $\Delta(\cdot)$ is the Laplacian derivative, with kernel size $k = 5$, applied to each frame of the input $\mathbf{V}_{flow}$. The function gives as output $\mathbf{V}^{(o)}$, which is the gradient of of the optical flow estimation of $\mathbf{V}$.

5.2 Body Parts Area Selection Algorithm

At this stage, specific areas of each video corresponding to relevant body parts of the subjects are selected frame-by-frame. The idea is to analyze the texture of the subjects parts that contain important details useful for the recognition of the person, that are therefore more difficult to synthesize. The chosen body parts of the subjects are $\mathcal{B} = \{FC, LH, RH, LF, RF\}$: the face ($FC$), the left hand ($LH$), the right hand ($RH$), the left foot ($LF$) and the right foot ($RF$). Let us explain the body parts selection algorithm step-by-step:

1. *Pose Detection*: We extract the coordinates of the subject's skeleton with the pose detector algorithm *Openpose* [7]. As described in Section 3.1.1, for each frame $\mathbf{F}$ contained in $\mathbf{V}$ of size $C \times H \times W$, *Openpose* estimates the position of 137 keypoints of the subject, that we call $\mathcal{C}_{\mathbf{F}} = \{\mathbf{c}_1, ..., \mathbf{c}_{137}\}$. Each keypoint is indicated as $\mathbf{c}_i = (\mathbf{c}_{i,x}, \mathbf{c}_{i,y})$, where i is the index of the keypoint, $\mathbf{c}_{i,x}$ is the coordinate along the x-axis, $\mathbf{c}_{i,y}$ is the coordinate along the y-axis. According to the annotation adopted by *Openpose*, each keypoint is referred to a certain part of the skeleton (for example $\mathbf{c}_0$ refers to the nose coordinates).

2. *Area Selection*: For each of the considered body parts $b \in \mathcal{B}$, we

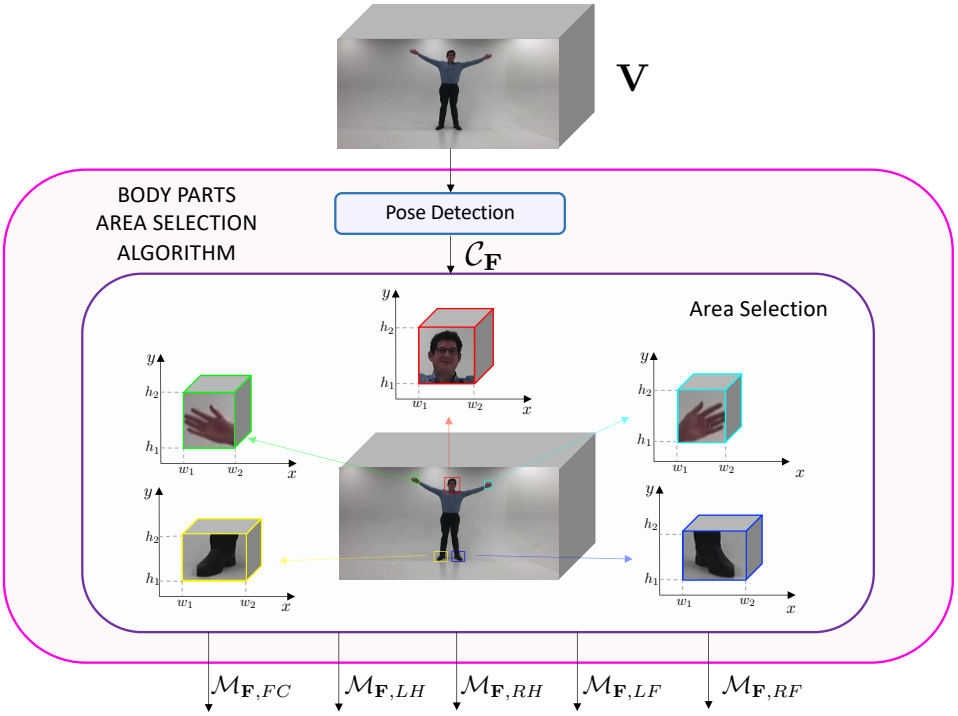

Figure 5.3: Scheme of the body parts area selection algorithm.

define:

$$\mathcal{M}_{\mathbf{F},b} = \phi_b(\mathcal{C}_{\mathbf{F}}) \tag{5.2}$$

where ϕ_b is the area selection function, which takes as input the set $\mathcal{C}_{\mathbf{F}}$ obtained in the previous step, and outputs the coordinates of the 4 points $\mathcal{M}_{\mathbf{F},b} = \{(w_1, h_1), (w_1, h_2), (w_2, h_1), (w_2, h_2)\}$ which delimit the area where b is located: w_1 and w_2 are the coordinates along the x-axis, h_1 and h_2 along the y-axis. The coordinates of the bounding box are different frame by frame, according to the location of the body part. Boxes surrounding a certain body part, within a certain video, keep the same dimensions. In Section 5.2.1 we explain the area selection function used for hands and feet; in Section 5.2.2 we deal with explanation of the face area selection function.

In Figure 5.3 we can see the general scheme of this algorithm.

5.2.1 Hands and Feet Area Selection

Let us solve the Equation (5.2) for all the body parts $b \in \mathcal{B}$, except for the face. For the face selection, we used an adapted version of the algorithm proposed in [6], explained in Section 5.2.2, that was not suitable for these

body parts selection for its construction. Now, let us explain the steps of the proposed algorithm:

1. According to the body part $b \in \mathcal{B}$ we are dealing with, let us select a subset of detected keypoints $\mathcal{C}_{\mathbf{F},b} \subset \mathcal{C}_{\mathbf{F}}$:

   - $\mathcal{C}_{\mathbf{F},LH} = \{\mathbf{c}_{26}, \mathbf{c}_{27}, ..., \mathbf{c}_{46}\}$ is the subset of the 21 keypoints referred to the left hand of the subject in frame $\mathbf{F}$;
   - $\mathcal{C}_{\mathbf{F},RH} = \{\mathbf{c}_{47}, \mathbf{c}_{48}, ..., \mathbf{c}_{67}\}$ is the subset of the 21 keypoints referred to the right hand of the subject in frame $\mathbf{F}$;
   - $\mathcal{C}_{\mathbf{F},LF} = \{\mathbf{c}_{15}, \mathbf{c}_{20}, \mathbf{c}_{21}, \mathbf{c}_{22}\}$ is the subset of the 4 keypoints referred to the left foot of the subject in frame $\mathbf{F}$;
   - $\mathcal{C}_{\mathbf{F},RF} = \{\mathbf{c}_{12}, \mathbf{c}_{23}, \mathbf{c}_{24}, \mathbf{c}_{25}\}$ is the subset of the 4 keypoints referred to the right foot of the subject in frame $\mathbf{F}$.

2. The detected keypoints of the skeleton $\mathcal{C}_{\mathbf{F}}$ are usually placed in the centre of the corresponding body part. Therefore, to ensure that the found box surrounds the entire body part, we also consider an offset q_b. So, let us calculate the box coordinates $\mathcal{M}_{\mathbf{F},b}$ for each $b \in \mathcal{B}$ as:

$$\begin{cases} w_1 = \max(0, \min((\mathbf{c}_{i,w}) - q_b) \\ h_1 = \max(0, \min((\mathbf{c}_{i,h}) - q_b) \\ w_2 = \min(W, \max((\mathbf{c}_{i,w}) + q_b) \\ h_2 = \min(H, \max((\mathbf{c}_{i,h}) + q_b) \end{cases} \tag{5.3}$$

   where the notation $\mathbf{c}_{i,w}$ indicates the coordinate along the x-axis of the i-th keypoint belonging to the set $\mathcal{C}_{\mathbf{F},b}$, and $\mathbf{c}_{i,h}$ indicates its coordinate along the y-axis. We used $q_b = 5$ with $b = \{LH, RH\}$, and $q_b = 12$ with $b = \{LF, RF\}$.

3. At this point, we calculate the centre of each of the two sides of the body part box:

$$\bar{w} = \frac{w_1 + w_2}{2} \qquad \bar{h} = \frac{h_1 + h_2}{2} \tag{5.4}$$

4. We repeat steps 1, 2 and 3 for every frame $\mathbf{F} \in \mathbf{V}$.

5. Let us calculate, among all the frames $\mathbf{F} \in \mathbf{V}$ and for each $b \in \mathcal{B}$, the average of the widths (Δ_w) and the average of heights (Δ_h) of all the obtained bounding boxes:

$$\Delta_w = \frac{1}{N} \sum_{\mathbf{F} \in \mathbf{V}} (w_{2_{(\mathbf{F})}} - w_{1_{(\mathbf{F})}}) \quad \Delta_h = \frac{1}{N} \sum_{\mathbf{F} \in \mathbf{V}} (h_{2_{(\mathbf{F})}} - h_{1_{(\mathbf{F})}}) \tag{5.5}$$

6. Then, for each $\mathbf{F} \in \mathbf{V}$, we update the coordinates $\mathcal{M}_{\mathbf{F},b}$ as indicated in Equation (5.6).

$$\begin{cases} w_1 &= \bar{w} - \frac{\Delta_w}{2} \\ h_1 &= \bar{h} - \frac{\Delta_h}{2} \\ w_2 &= \bar{w} + \frac{\Delta_w}{2} \\ h_2 &= \bar{h} + \frac{\Delta_h}{2} \end{cases} \tag{5.6}$$

   In this way we ensure that the areas referred to the body part b have the same resolution in all the frames of a certain video $\mathbf{V}$. This would be necessary for the calculation of the features in Section 5.4.

7. For each $\mathbf{F} \in \mathbf{V}$, to avoid inconsistencies of the just calculated box coordinates with the frame resolution $H \times W$, we update their values in this way.

$$\begin{cases} (w_1 = 0,\ w2 = \Delta_w) & \text{if } w1 < 0 \\ (h_1 = 0,\ h2 = \Delta_h) & \text{if } h1 < 0 \\ (w_1 = W - \Delta_w,\ w2 = W) & \text{if } w2 > W \\ (h_2 = H - \Delta_h,\ h2 = H) & \text{if } h2 > H \end{cases} \tag{5.7}$$

5.2.2 Face Area Selection

The face area selection function is the one used in [6] for the Face GAN training step. For each frame $\mathbf{F} \in \mathbf{V}$, as in Section 5.2.1, we consider the array of coordinates of the subject's skeleton $\mathcal{C}_{\mathbf{F}}$. To select the area surrounding b, we do the following steps:

1. Let us consider, for each $\mathbf{F} \in \mathbf{V}$, the keypoints $\mathbf{c}_{18} = (\mathbf{c}_{18,x}, \mathbf{c}_{18,y}) \in \mathcal{C}_{\mathbf{F}}$ and $\mathbf{c}_{19} = (\mathbf{c}_{19,x}, \mathbf{c}_{19,y}) \in \mathcal{C}_{\mathbf{F}}$, which are respectively the coordinates of the left and right ear of the subject.

2. Let us calculate, for each $\mathbf{F} \in \mathbf{V}$, the middle point between the coordinates of the two ears of the considered person, that is the center of the box containing the face.

$$\bar{w} = \frac{\mathbf{c}_{18,x} + \mathbf{c}_{19,x}}{2} \qquad \bar{h} = \frac{\mathbf{c}_{18,y} + \mathbf{c}_{19,y}}{2} \tag{5.8}$$

   where $\bar{w}$ is the middle-point coordinate along the x-axis, and $\bar{h}$ is the middle-point coordinate along the y-axis.

3. Now, for each $\mathbf{F} \in \mathbf{V}$, we calculate the face box coordinates, solving Equation (5.9).

$$\begin{cases} w_1 = max(\bar{w} - \frac{W_{FC}}{2}, 0) \\ h_1 = max(\bar{h} - \frac{H_{FC}}{2}, 0) \\ w_2 = min(\bar{w} + \frac{W_{FC}}{2}, W) \\ h_2 = min(\bar{h} + \frac{H_{FC}}{2}, H) \end{cases} \tag{5.9}$$

where $(H_{FC} \times W_{FC})$ is the pixel resolution of the face bounding box.

4. In the end, to avoid inconsistencies of the box coordinates just calculated with the frame resolution $(H \times W)$, we update their values in this way:

$$\begin{cases} (w_1 = 0,\ w2 = W_{FC}) & \text{if } w1 < 0 \\ (h_1 = 0,\ h2 = H_{FC}) & \text{if } h1 < 0 \\ (w_1 = W - W_{FC},\ w2 = W) & \text{if } w2 > W \\ (h_2 = H - H_{FC},\ h2 = H) & \text{if } h2 > H \end{cases} \tag{5.10}$$

5.3 CNN-based classification

In this section we show the first approach we have used to classify our videos. Since we are dealing with a problem of fake content recognition, we decided to use a method based on a simple neural network as a baseline. The network processes our videos in a frame-by-frame mode, and works as a feature extractor and as a classifier of the learned features. Even if the generic input of the network is a single frame, the classification is computed on isolated temporal sequences of frames belonging to a certain video. Inspired by state-of-the-art face manipulation detection works, we have decided to do this kind of classification considering only the faces of the subjects and only the pre-processed grayscale videos as input. The first step to do is face cropping, as described in Section 5.3.1. Then, in Section 5.3.2 we describe EfficientNet-B0, the neural network we use for this task, and in Section 5.3.4 we show our classification procedure.

5.3.1 Face Cropping

Let us consider only the pre-processed videos $\mathbf{V}^{(g)}$. After selecting the face areas of the subjects frame-by-frame as indicated in Section 5.2.2, we

crop these frames on these selected portions. In this way, for each $\mathbf{V}^{(g)}$, we get the *cropped-on-face* video $\mathbf{V}_{FC}^{(g)}$, which is a 3D array of shape $N \times H_{FC} \times W_{FC}$, with N the number of frames and $H_{FC} \times W_{FC}$ the resolution of each frame.

5.3.2 EfficientNet-B0

The neural network architecture we use for our binary classification task is called EfficientNet-B0. EfficientNet-B0 is the baseline architecture of EfficientNet's family of CNN models introduced by Tan et al. in [75]. These networks were designed to increase accuracy uniformly scaling their dimensions (depth, width and resolution) using a compound scaling factor. We use this network for the good accuracy results achieved in computer vision and multimedia forensics fields using fewer parameters than other famous networks, which allow us to have faster training phases.

As depicted in Table 5.1, the EfficientNet-B0 is composed of several convolutional layers and a final fully connected layer. For each i-th stage of the network, we indicate the type of operator ($\hat{\mathcal{F}}_i$), the input resolution ($\hat{H}_i \times \hat{W}_i$), the number of output channels ($\hat{C}_i$) and the number of occurrences of the layers ($\hat{L}_i$).

Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224 × 224	32	1
2	MBConv1, k3x3	112 × 112	16	1
3	MBConv6, k3x3	112 × 112	24	2
4	MBConv6, k5x5	56 × 56	40	2
5	MBConv6, k3x3	28 × 28	80	3
6	MBConv6, k5x5	14 × 14	112	3
7	MBConv6, k5x5	14 × 14	192	4
8	MBConv6, k3x3	7 × 7	320	1
9	Conv1x1 & Pooling & FC	7 × 7	1280	1

Table 5.1: Architecture details of EfficientNet-B0.

The main operators of the network are the Mobile inverted Bottleneck Convolution (MBConv), which is the fundamental block of the MobileNetV2 neural network introduced in [76]. MBConv belongs to a particular family of convolutional blocks called inverted residual blocks.

As we can see in figure Figure 5.4, general residual blocks connect the beginning and the end of a convolutional block with a skip connection.

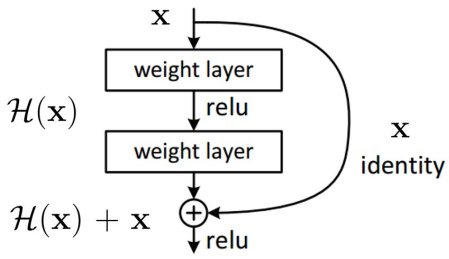

Figure 5.4: General residual block [9].

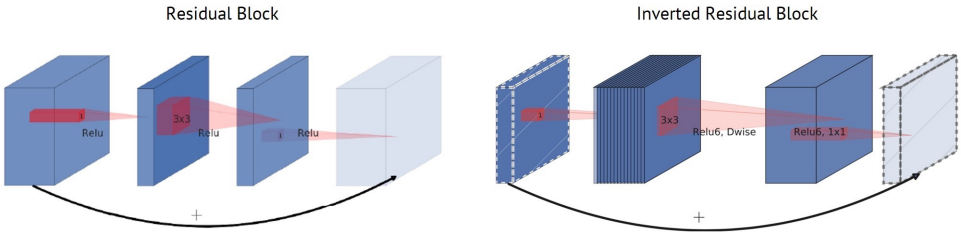

Figure 5.5: Representation of a residual block and an inverted residual block [9].

Let us explain the main idea behind this structure. We know that a neural network is a function approximator. In general, deep neural networks work well for complex functions and small networks do the same for simple ones. It turns out that for the degradation problem, sufficiently deep networks may have difficulties to learn simple functions, and a priori we cannot know the function's complexity. To solve this problem, the residual block introduces a skip connection that allows to skip some layers and learn a simpler function, if needed. In Figure 5.5, we can see the structure of the inverted residual block, which is a modification of the classic one. The inverted residual block differs from the classic residual one for the dimension of the inner blocks' channels: while the first one follows a wide-narrow-narrow approach, the inverted residual block follows a narrow-wide-narrow approach. The operation computed by the inverted residual block are: a point-wise (1×1) convolution + ReLU6 activation function, where the number 6 indicates that the activation is linear as long as it is between 0 and 6; a depth-wise separable convolution (Dwise) with a kernel filter of size (3×3); a point-wise convolution + a linear activation function. The names MBConv1 and MBConv6 contain the information of the so-called expansion factor (1 and 6), which affects the number of channels of the intermediate layers of the block.

5.3.3 CNN Data pre-processing

The network admits as input single frames taken from pre-processed grayscale *cropped-on-face* videos, indicated as $\mathbf{F}_{FC}^{(g)} \in \mathbf{V}_{FC}^{(g)}$ of size $H_{FC} \times W_{FC}$. It is a good practice to normalize data before providing them to neural networks; this is done because it can help the training phase [77]. There are several ways to normalize data; we use the z-score normalization:

$$\mathbf{Z}_l = \frac{(\mathbf{F}_{FC}^{(g)})_l - \mu}{\sigma} \tag{5.11}$$

where the notation $(\mathbf{F}_{FC}^{(g)})_l$ and $\mathbf{Z}_l$ indicates, respectively, the l-th pixel of the not normalized and the normalized frame; μ and σ respectively indicate the mean and the standard deviation of $\mathbf{F}_{FC}^{(g)}$.

5.3.4 Classification

As we are dealing with a binary classification problem, we need to replace the last fully connected layer of the EfficientNet-B0 with another one with a number of nodes equal to the number of output classes. The layer applies a linear transformation :

$$\mathbf{p}_f = \mathbf{W} \cdot \mathbf{x}_f + \mathbf{b} \tag{5.12}$$

where, f is the index of the f-th input frame of the network; $\mathbf{p}_f = [(\mathbf{p}_f)_0, (\mathbf{p}_f)_1]$ is the output of the network; $\mathbf{x}_f$ is the feature vector extracted from the EfficientNet-B0 of size 1280, $\mathbf{W}$ is the trainable weight matrix of size 2×1280 and $\mathbf{b}$ is a trainable bias parameter of size 2. The output vector $\mathbf{p}_f$ is passed into a nonlinear layer which calculates the *softmax* function. The *softmax* is an exponential function that maps the input into the range [0,1] and it is defined as:

$$\sigma(\mathbf{p}_f)_i = \frac{e^{(\mathbf{p}_f)_i}}{\sum_{j=0}^{1} e^{(\mathbf{p}_f)_j}} \tag{5.13}$$

where the function $\sigma(\mathbf{p}_f)_i$ scales the i-th value of $\mathbf{p}_f$ in the new range, and $j = [0, 1]$ is the index of the class. This normalization ensures the sum of the $\mathbf{p}_f$ components $(\mathbf{p}_f)_0 + (\mathbf{p}_f)_1$ to be 1. The output of Equation (5.13) is associated with the i-th node of the final fully connected layer. Each of these two values represents the probability of the input frame belonging to the class "real" or the class "fake".

In the testing phase we deal with predicting the label of a temporal sequence of frames, by aggregating the predictions on the single frames of

the sequence. This is done to compare the results obtained with this classification method with the ones obtained with the feature-based method, where each feature corresponds to a sequence of consecutive frames. In order to do it, we consider temporal sequences of grayscale *cropped-on-face* frames $\mathbf{S}_{FC}^{(g)}$ of size $K \times H \times W$. We want to calculate the probability of the entire sequence being "real" as the average of the probabilities of the single frames of the sequence being "real", and the probability of the entire sequence being "fake" as the average of the probabilities of the frames of the sequence being "fake". So, let us calculate:

$$\mathbf{p}_{FC}^{(g)} = \frac{1}{K} \sum_{f=1}^{K} \sigma(\mathbf{p}_f)_i \tag{5.14}$$

where $\mathbf{p}_{FC}^{(g)} = [(\mathbf{p}_{FC}^{(g)})_0, (\mathbf{p}_{FC}^{(g)})_1]$ indicates the probability of $\mathbf{S}_{FC}^{(g)}$ belonging to one of the classes with index $i = [0, 1]$, f is the index of a frame in the sequence, with $f = [0, ..., K-1]$.

The predicted label of $\mathbf{S}_{FC}^{(g)}$ is calculated as follows:

$$\hat{y}_{FC}^{(g)} = \arg\max_i (\mathbf{p}_{FC}^{(g)})_i \tag{5.15}$$

where:

- if $\hat{y}_{FC}^{(g)} = 0$, the sequence $\mathbf{S}_{FC}^{(g)}$ is labeled as "real".
- if $\hat{y}_{FC}^{(g)} = 1$, the sequence $\mathbf{S}_{FC}^{(g)}$ is labeled as "fake".

5.4 Feature-based classification

In this section we introduce the core method used for our classification task. The idea is to find a tool that infers if a video sequence is synthesized or not, focusing on the body parts of the subjects in the video scenes. The method is based on the extraction of the handcrafted textural features Local Pattern Derivative on Three Ortogonal Planes (LDP-TOP) (Section 2.2.2) from our video sequences, and the use of a Multi-Layer Perceptron (MLP) as a feature classifier (Section 2.1.1). In Figure 5.6 we show the general pipeline of the proposed method. Let us introduce the notation and explain the method step-by-step:

1. *Input video*: given $\mathbf{V} \in \mathcal{V}$, the input of our classification block are the pre-processed videos $\mathbf{V}^{(\rho)} = [\mathbf{V}^{(g)}, \mathbf{V}^{(o)}]$. Each of them will follow a certain configuration of classification. Each video can be "real" (i.e. labeled as 0) or "fake" (i.e. labeled as 1).

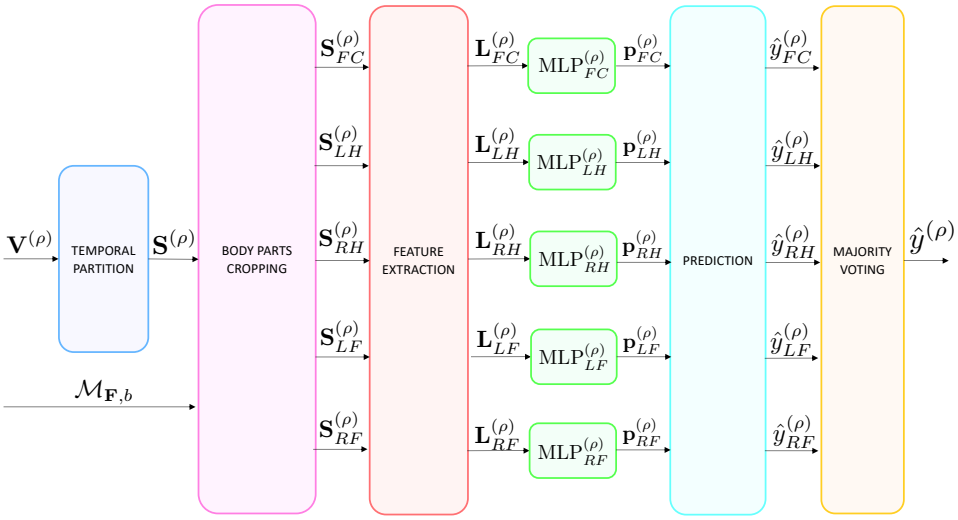

Figure 5.6: Feature-based classification method outline.

2. *Temporal partition*: By definition, each LDP-TOP feature corresponds to a sequence of consecutive frames. The idea is to test our model on a certain feature, in order to predict the class of its corresponding temporal sequence of frames. So, before extracting the features, we need to isolate temporal sequences $\mathbf{S}^{(\rho)}$ from our input video $\mathbf{V}^{(\rho)}$, where $\rho = [g, o]$ refers to the considered pre-processed video. Each sequence is composed of K consecutive frames and the amount of frames between the beginning of one sequence and the beginning of the following one is Q, with $Q < K$.

3. *Body parts cropping*: Given a temporal sequence $\mathbf{S}^{(\rho)}$, with $\rho = [g, o]$, we consider all the frames $\mathbf{F}^{(\rho)} \in \mathbf{S}^{(\rho)}$. For each frame in the sequence, we get all the body parts area coordinates $\mathcal{M}_{\mathbf{F},b}$, with $b \in \mathcal{B}$, obtained applying the algorithm explained in Section 5.2. For each frame in the sequence and for each body part, we crop the part inside the area bounded by the coordinates $\mathcal{M}_{\mathbf{F},b}$. In this way, for each $b \in \mathcal{B}$, we get the sequence of the cropped body part indicated with the notation $\mathbf{S}^{(\rho)}_b$, where the index b refers to the body part associated with the sequence:

   - $\mathbf{S}^{(\rho)}_{FC}$ is the sequence cropped on subject's face;
   - $\mathbf{S}^{(\rho)}_{LH}$ is the sequence cropped on subject's left hand;
   - $\mathbf{S}^{(\rho)}_{RH}$ is the sequence cropped on subject's right hand;

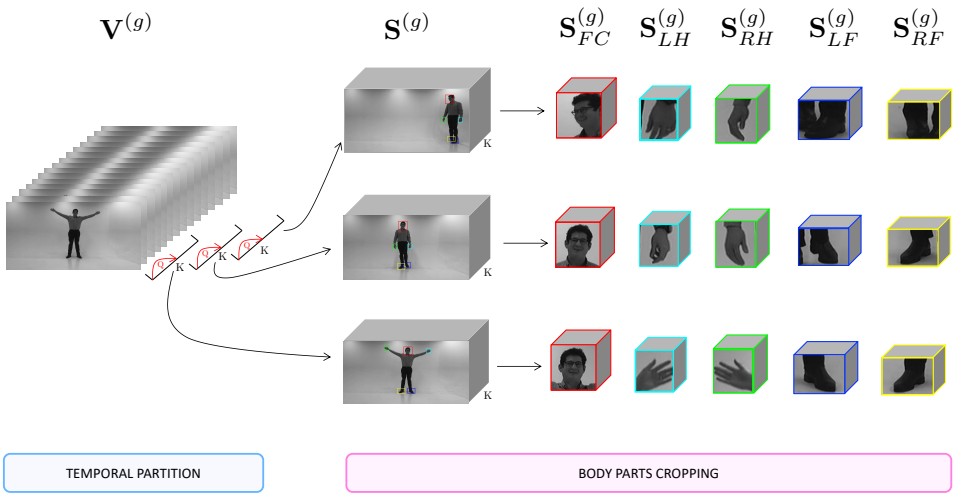

Figure 5.7: Zoom on temporal partition and body parts area selection given as input $\mathbf{V}^{(g)}$.

- $\mathbf{S}^{(\rho)}_{LF}$ is the sequence cropped on the subject's left foot;
- $\mathbf{S}^{(\rho)}_{RF}$ is the sequence cropped on the subject's right foot.

In Figure 5.7 we focus on the temporal partition and body parts area selection steps when the input is $\mathbf{V}^{(g)}$, and in Figure 5.8 we focus on the temporal partition and body parts area selection steps when the input is $\mathbf{V}^{(o)}$.

4. *Feature extraction*: For each sequence obtained at the step before, we extract the 1D feature vector LDP-TOP of size 3072, indicated as $\mathbf{L}^{(\rho)}_b$, following the feature extraction method explained in Section 2.2.2. The feature vectors computed from each temporal sequences inherit the label of the video they belong.

5. *Training phase*: we train one MLP classifier (Section 2.1.1) for each group of features $\mathbf{L}^{(\rho)}_b$, indicated as $\text{MLP}^{(\rho)}_b$. The architecture of the MLP is the following: an input layer which takes as input the feature vector of size 3072; three hidden layers composed respectively by 100, 50 and 25 neurons; an output layer composed by 2 neurons. So, each MLP classifier gives as output a vector of size 2, where the first value is associated with the probability of the input sequence to belong to the "real" class and the second value is associated with the probability of the input sequence to belong to the "fake" class.

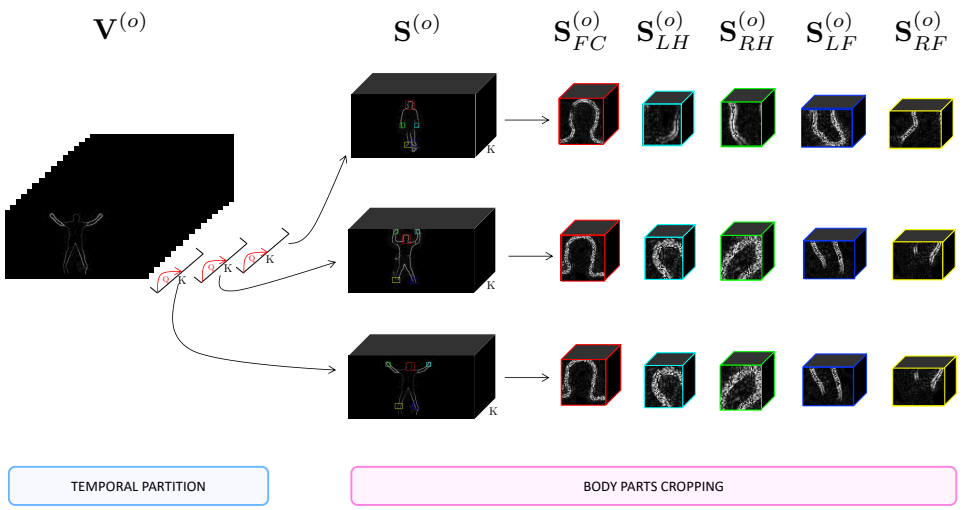

Figure 5.8: Zoom on temporal partition and body parts area selection given as input $\mathbf{V}^{(o)}$.

6. *Testing phase*: During the testing phase, the features extracted from the videos of the testing dataset $\mathbf{L}_b^{(\rho)}$ are fed into the trained classifiers $\text{MLP}_b^{(\rho)}$, in order to predict their class. We need to introduce the following equation:

$$\mathbf{p}_b^{(\rho)} = \text{MLP}_b^{(\rho)}\left(\mathbf{L}_b^{(\rho)}\right) \tag{5.16}$$

where the notation $\mathbf{p}_b^{(\rho)} = [(\mathbf{p}_b^{(\rho)})_0, (\mathbf{p}_b^{(\rho)})_1]$ indicates the probability of the sequence $\mathbf{S}_b^{(\rho)}$ to belong to one of the two classes "real" or "fake" indicated with index $i = [0, 1]$.

7. *Prediction*: Now, let us calculate the predicted label of the input sequence $\mathbf{S}_b^{(\rho)}$:

$$\hat{y}_b^{(\rho)} = \arg\max_i \, (\mathbf{p}_b^{(\rho)})_i \tag{5.17}$$

where $(\mathbf{p}_b^{(\rho)})_i$ are the probabilities calculated at the step before, and $\hat{y}_b^{(\rho)}$ is the predicted label.

8. *Majority voting*: to evaluate if the *not-cropped* video sequence $\mathbf{S}^{(\rho)}$ is real or fake, we calculate the majority voting between all the predictions $\hat{y}_b^{(\rho)}$ calculated for each cropped body parts sequences $\mathbf{S}_b^{(\rho)}$ at the previous step:

$$\hat{y}^{(\rho)} = \text{maj}\left(\{\hat{y}_{FC}^{(\rho)},\ \hat{y}_{LH}^{(\rho)},\ \hat{y}_{RH}^{(\rho)},\ \hat{y}_{LF}^{(\rho)},\ \hat{y}_{RF}^{(\rho)}\}\right) \tag{5.18}$$

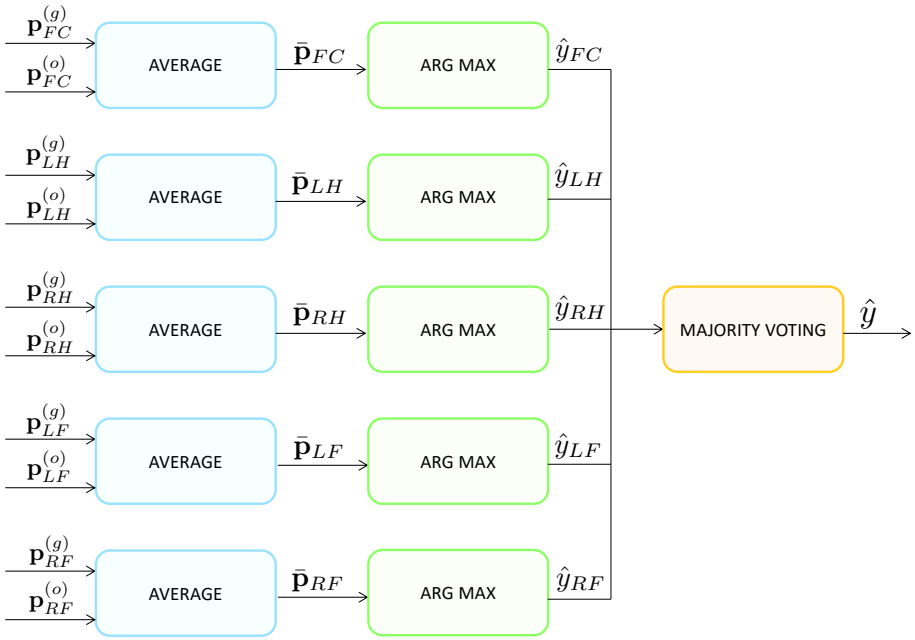

Figure 5.9: Scheme of the combined prediction technique.

where the operator maj$(\cdot)$ outputs the value recurring most frequently between the input predictions.

In Section 5.4.1 we introduce a combined prediction technique: the idea is that, for each temporal sequence, the prediction is done considering, at the same time, the contribution of the sequences obtained by the two processed videos $\mathbf{V}^{(g)}$ and $\mathbf{V}^{(o)}$.

5.4.1 Combined Prediction Technique

The combined prediction technique introduced in this section allows to predict the label of a generic temporal sequence of frames, that we indicate as $\mathbf{S}$, isolated from a $\mathbf{V} \in \mathcal{V}$, by considering at the same time the contribution of the two temporal sequences $\mathbf{S}^{(g)}$ and $\mathbf{S}^{(o)}$ isolated from the two pre-processed video $\mathbf{V}^{(g)}$ and $\mathbf{V}^{(o)}$. The information content of the two sequences $\mathbf{S}^{(g)}$ and $\mathbf{S}^{(o)}$ is the same, and it is referred to the frames contained in $\mathbf{S} \in \mathbf{V}$.

Now, let us explain the proposed method step-by-step:

1. First, we follow the steps [1,2,3,4,5,6] explained in the previous section, for both the pre-processed videos $\mathbf{V}^{(g)}$ and $\mathbf{V}^{(o)}$. For both videos, we isolate the temporal sequences, we crop the sequences on

the areas referred to each body parts, we extract the features from these cropped sequences and then we train one MLP classifier for each group of features. In the testing phase, we calculate the class probability of the test features. So, for each generic body cropped sequence indicated as $\mathbf{S}_b$, we have the probability $\mathbf{p}_b^{(g)}$ calculated for the corresponding sequences $\mathbf{S}_b^{(g)}$ and the probability $\mathbf{p}_b^{(o)}$ calculated for the corresponding sequences $\mathbf{S}_b^{(o)}$.

2. *Averaging Probabilities*: For each $b \in \mathcal{B}$, we calculate the probability of the temporal sequence $\mathbf{S}_b$ as the average between the probability calculated for the corresponding sequences $\mathbf{S}_b^{(g)}$ and $\mathbf{S}_b^{(o)}$. The probability is calculated as follows:

$$\bar{\mathbf{p}}_b = \frac{\mathbf{p}_b^{(g)} + \mathbf{p}_b^{(o)}}{2} \tag{5.19}$$

where the notation of $\bar{\mathbf{p}}_b = [(\bar{\mathbf{p}}_b)_0, (\bar{\mathbf{p}}_b)_1]$ indicates the probability of $\mathbf{S}_b$ belonging to one of the classes "real" or "fake" indicated with index with $i = [0, 1]$, while $\mathbf{p}_b^{(g)} = [(\mathbf{p}_b^{(g)})_0,\ (\mathbf{p}_b^{(g)})_1]$ and $\mathbf{p}_b^{(o)} = [(\mathbf{p}_b^{(o)})_0, (\mathbf{p}_b^{(o)})_1]$ are the probabilities calculated, respectively, for the corresponding sequences $\mathbf{S}_b^{(g)}$ and $\mathbf{S}_b^{(o)}$. In this way, the probability of each sequence $\mathbf{S}_b$ belonging to a certain class is calculated by considering at the same time the contribution of both the temporal sequences deriving from the two pre-processed videos.

3. *Prediction*: Now, let us calculate the predicted label of the input sequence $\mathbf{S}_b$:

$$\hat{y}_b = \arg\max_i \, (\bar{\mathbf{p}}_b)_i \tag{5.20}$$

where $(\mathbf{p}_b^{(\rho)})_i$ are the probabilities calculated at the step before, and $\hat{y}_b^{(\rho)}$ is the predicted label.

4. *Majority voting*: to evaluate if the whole *not-cropped* video sequence $\mathbf{S}$ is real or fake, we calculate the majority voting between all the predictions $\hat{y}_b$:

$$\hat{y} = \text{maj}\Big(\{\hat{y}_{FC},\ \hat{y}_{LH},\ \hat{y}_{RH},\ \hat{y}_{LF},\ \hat{y}_{RF}\}\Big) \tag{5.21}$$

where the operator $\text{maj}(\cdot)$ outputs the value recurring most frequently between the input predictions.

In Figure 5.9 we show a representation of the combined prediction technique.

5.5 Conclusive Remarks

In this chapter we have described the two methodologies proposed to solve the video classification task. First, we started by describing the video pre-processing phase and the body parts area selection algorithm. Then, we have described the CNN-based classification method, which is our working baseline, and then the feature-based classification method, which is our proposed method.

6
Experiments and Results

In this chapter we explain in detail the experiments carried out and the practical implementation choices of the proposed methodologies.

6.1 Dataset

In this section we describe the dataset involved in the video-to-video translation algorithm and used for our classification task.

The videos of the dataset come from different sources: videos generated and recorded by us, a collection of videos taken from the online video sharing platform YouTube, and the open-source dataset released from the authors of the algorithm *Everybody Dance Now* [6].

The dataset is divided into three parts. Let us explain in detail each of these parts:

- *Source videos*: these videos are used as source for the video-to-video translation algorithm, in order to impose the motion of the source subjects onto the target videos, obtaining the synthesized videos as results. The videos are chosen from the list of YouTube clips reported in the *Everybody Dance Now*'s repository. The only requirement of the chosen clips is that the scene presents an in-the-wild single-dancer, filmed by a static camera. This part of the dataset consists of 5 short videos downloaded from YouTube,

corresponding to a total of 20.169 frames. The average number of frames in these videos is 4034 frames; the longest video is composed by 6026 frames and the shortest one by 1999 frames. The original resolution of each frame of the videos is 1920×1080 pixel, but we have resized them at the resolution of 1024×512 pixel to fulfil the requirements of the video synthesis algorithm.

- *Target videos*: these are the videos on which we want to transfer the movements of the subjects of the source videos. The videos are used to train the models used in the video-to-video translation algorithm. To learn the appearance of the target subject in many poses, it is important that the target video captures a sufficient range of motion and sharp frames with minimum blur. All the videos have been recorded by a static camera, and they have a static background. This part of the dataset consists of 10 long videos, each one corresponding to a different human subject. Each video is taken from the following sources:

  - *Everybody Dance Now* dataset: this is the open-source dataset released by the authors of *Everybody Dance Now* [6]. It is composed by 5 videos, corresponding to a total of 97.708 frames. We have 4 videos at 1920×1080 resolution, and 1 at 1280×720, that have been resized to a resolution of 1024×512 pixel for our purposes. The target subjects are normal people and professional dancers. As reported in [6], the subjects have been filmed from 8 to 17 minutes with a static modern cellphone camera with a real time footage at 120 frames per second. The videos correspond to a total of 97708 frames and an average number of 19542 frames; the longest video is composed by 30212 frames and the shortest one by 11752 frames.
  - *Homemade videos*: this part of the dataset is composed by 3 videos recorded by us. The videos correspond to a total of 47242 frames and an average number of 15747 frames; the longest video is composed by 19321 frames and the shortest one by 9100 frames. The videos have been recorded with a resolution of 1920×1080 pixel, and then resized to 1024×512 pixel. The target subjects are the author himself, a friend and his father. All the subjects have been filmed from 5 to 10 minutes with a static modern camera with a real time footage at 60 frames per second.

  - *YouTube videos*: this part of the dataset is composed of 2 videos downloaded from YouTube: the first video contains 30219 frames and the second one 22280 frames. The videos consist of workout tutorials given by professional personal trainers. The initial video resolution is 1920×1080 pixel, then resized to 1024×512 for our purpose.

  Furthermore, these videos are used for our classification task and they are labeled as "real".

- *Synthesized videos*: we generated these videos by applying the video synthesis algorithm described in Section 3.1. In total we have generated 50 videos, corresponding to 200.590 frames of resolution 1024×512 pixel. The average number of frames in these videos is 4012 frames; the longest video is composed by 5973 frames and the shortest one by 1993 frames. In particular, each of the 10 target subject is synthesized with the appearance of each of the 5 source videos.

  For the generation of these videos, we use the same settings reported by the authors of *Everybody Dance Now* [6]. So, we train one model for each of the 10 target subjects. As reported in the paper, we follow the progressive learning schedule from pix2pixHD [8]: the global generator takes as input frames with resolution of 512×256 pixel, and the local enhancer takes as input frames with resolution of 1024×512 pixel. For predicting face residual, we use the global generator of pix2pixHD and a single 70×70 Patch-GAN discriminator [35]. For each model we train the global stage for 5 epochs, the local stage for 30 epochs, and the face GAN for 5 epochs. The time needed to train all the stages of each models is from 1 to 2 weeks, and it varies according to the number of frames of the target video: the more is the number of frames, the more is the time needed for the training phase.

  Furthermore, these are the videos labeled as "fake" used for our classification task.

In Figure 6.1 we can see some frames taken from source videos, target videos and synthesized videos of our dataset.

6.1.1 Compression and resizing

The videos used for our classification task, which are *target videos* and *synthesized videos*, have been coded using the H.264 codec [78] with a

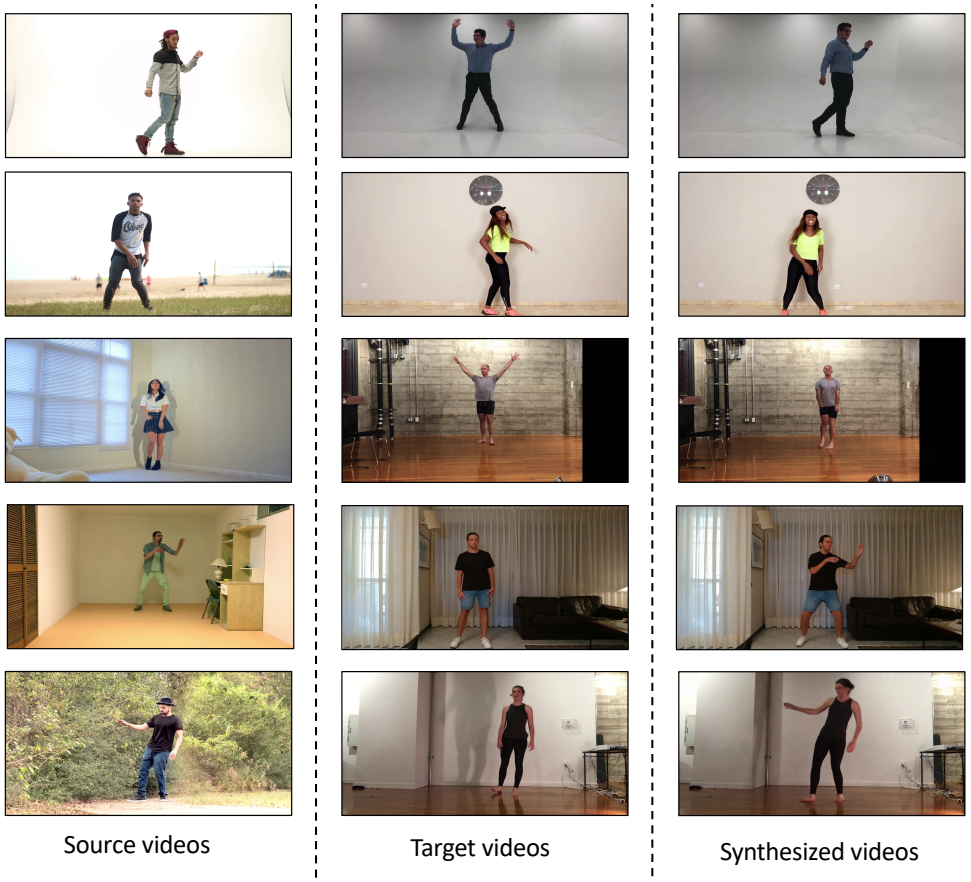

Figure 6.1: Sample of video frames taken from our dataset.

Constant Rate Factor (CRF) equal to 0, which let us create lossless videos, with a resolution of 1024×512 pixel. In our analysis, we evaluate how compression and resizing of the videos impacts our method.

For the compression, we use FFmpeg [79] at four different CRF values: 18, 23, 40, 51; smaller values correspond to better quality, larger values to lower quality.

Each videos is resized to the following resolutions (in pixel): 512×256, 256×128, 128×64, 64×32.

6.2 Experimental Setup

In this section we discuss our experimental setup. In Section 6.2.1 we describe how we get temporal sequences from our videos; in Section 6.2.3 we show the setup of the network used for the Convolutional Neural Network (CNN)-based classification; in Section 6.2.4 we show the setup of the Multi-Layer Perceptron (MLP) classifier used for the feature-based classification; in Section 6.2.2 we describe the dataset split policy we used.

6.2.1 Temporal Sequences Partition

Each temporal sequence $\mathbf{S}$, isolated from a certain video $\mathbf{V}$, is composed of K consecutive frames. The amount of frames between the beginning of one sequence and the beginning of the following one is Q, with $Q < K$. So, the n-th sequence isolated from our input video is defined as:

$$(\mathbf{S})_n = \{(\mathbf{F})_{n \cdot Q}\ ,\ \ldots\ ,\ (\mathbf{F})_{n \cdot Q + (K-1)}\} \tag{6.1}$$

where the notation $(\mathbf{S})_n$ indicates the n-th temporal sequence isolated from the input video, $\mathbf{F}$ indicates a generic frame, $n \cdot Q$ is the index of the first frame of the sequence, $n \cdot Q + (K-1)$ is the index of the last frame of the sequence. The amount of isolated sequences depends on the number of frames contained in the input video. Each temporal sequence is a 3D array with size $K \times C \times H \times W$. We fix the windows parameters with the following values: $K = 50, Q = 25$, referring to the experiments results shown in [29].

6.2.2 Leave-One-Out Cross Validation

In many applications, the supply of data for training and testing is limited, and in order to build good models, we wish to use as much of the available data as possible for training. One solution to this dilemma is Cross-Validation [80], which is a statistical method of evaluating and comparing learning algorithms by dividing the dataset into two segments: one used to learn or train a model and the other used to validate the model. In typical cross-validation, the training and validation sets must cross over in successive rounds such that each data point has a chance of being validated against. The basic form of cross-validation is k-fold Cross-Validation, and we focus on a particular type called Leave-One-Out Cross-Validation (LOOCV). This procedure is used in the case we have a small dataset or when an accurate estimate of model performance is more important than the computational cost of the method. An accuracy estimate obtained using LOOCV is known to be almost unbiased, but it has high variance [81].

For what concerns the number of subjects that appear in our videos, which are only 10, we can say that we are in a limited data condition and so, we need to use this evaluation technique. The k-fold Cross-Validation has a single hyperparameter k that controls the number of subsets that a dataset is split into. Once split, each subset is given the opportunity to be used as a test set while all other subsets together are used as a

training dataset. LOOCV, in our case, involves fitting and evaluating $k = 10$ models, that means one model for each subject in the dataset.

Given our dataset $\mathcal{V}$, let us consider the subset $\{\mathcal{N}_n\} \subset \mathcal{V}$, with $n = [0, ..., k-1]$, which is composed of all the real and fake videos where the n-th subject appears, and the subset $\{\mathcal{V} \backslash \mathcal{N}_n\} \subset \mathcal{V}$ which is composed of all the real and fake videos where the n-th subject doesn't appear. The idea is to do a train-test split of our dataset k times, one for each of the n-th available subject, as follows:

- Training dataset: $\{\mathcal{V} \setminus \mathcal{N}_n\}$
- Testing dataset: $\{\mathcal{N}_n\}$

Each training dataset $\{\mathcal{V} \setminus \mathcal{N}_n\}$ is balanced to increase the accuracy of the classification [82]. The balancing is done in this way:

1. Given a subject which appears in the dataset, we calculate the amount of frames contained in the video labeled as "real" referred to this subject, and the amount of frames contained in the videos labeled as "fake" referred to this subject.
2. Then, we calculate the minimum between the two quantities calculated at step 1, and we indicate it as k.
3. The frames in the dataset are ordered by two indices: the first one is the temporal index in the video timeline, and the second one is the identity index of the video to which it belongs. We keep the first k frames contained in the video labeled as "real" and the first k frames contained in the video labeled as "fake".
4. We repeat this balancing procedure for each subject which appears in the videos of the training dataset.

In the end, the dataset is shuffled randomly to reduce variance and overfit less.

6.2.3 CNN setup

The CNN network EfficientNet-b0 takes as input *cropped-on-face* frames taken from the pre-processed grayscale videos, described in Section 5.3. The chosen resolution of the cropped frames is 70×70 pixel.

Concerning the dataset split policy, first we apply the LOOCV technique described in Section 6.2.2: in each experimental configuration, the training dataset contains the *cropped-on-face* frames of the videos where

all but one of the subjects appears, while the testing dataset contains the *cropped-on-face* frames of videos where that subject appears. The next step is to use the 20 % of the frames contained in the just created training dataset for the validation phase, chosen in a random way. The training and validation datasets are divided in several smaller portions, called batches. In our case, we use a batch-size of 128 samples. The optimization algorithm used for our network is the Adam optimizer [83]. The learning rate is initialized to 10^{-2} and it is decreased exponentially with a decay constant equal to 0.9, in each epoch where the validation loss does not improve. The EfficientNetB0 weights are initialized randomly. As a loss function, we use the Cross-Entropy Loss (CEL), defined as follows:

$$\text{CEL} = -\frac{1}{M}\sum_{i=1}^{M}\sum_{j=0}^{1} y_{ij}\, log(\mathbf{p}_{ij}) \tag{6.2}$$

where M is the batch size; i is the index of the i-th frame in the batch, $j = [0, 1]$ is the index of the class; y_{ij} is the label of the frame (0 if "real", 1 if "fake"); $\mathbf{p}_{ij}$ is the probability of the i-th frame of the batch to belong to j-th class.

We train the network for at most 200 epochs, and the training phase is stopped if the validation loss does not decrease for more than 10 epochs. The model providing the best validation loss is selected and saved.

6.2.4 MLP setup

To perform our Feature-based classification, we need to consider both the pre-processed videos described in Section 5.1. MLP takes as input the features extracted from a certain body part area selected of a pre-processed videos, as described in Section 5.4. As the number of selected body parts $|\mathcal{B}| = 5$, we train 5 different classifiers for each of the two pre-processed videos. In the combined technique (Section 5.4.1), where we consider at the same time both pre-processed videos $\mathbf{V}^{(\rho)} = [\mathbf{V}^{(g)}, \mathbf{V}^{(o)}]$, we train 10 classifiers for each experiment.

Concerning the dataset split policy, first we apply the LOOCV technique described in Section 6.2.2: in each experimental configuration, the training dataset contains the features extracted from the videos where all but one of the subjects appears, while the testing dataset contains the features extracted from the videos where that subject appears. We have not foreseen a validation phase. The used batch size is 200. The chosen activation function for each neuron in the network is Rectified Linear Unit (ReLU). The optimization algorithm used for our network

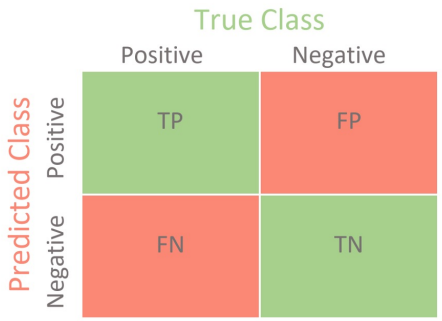

Figure 6.2: Representation of a confusion matrix.

is the Adam optimizer. The learning rate is initialized to 10^{-3} and it is kept constant. We train the network for at most 300 epochs, and the training phase is stopped when the loss is not improving by at least a value of 10^{-4} for 10 consecutive epochs. As a loss function, we use the CEL, defined as in Equation (6.2). In this case each sample of the batch is a feature vector and not a frame.

6.3 Evaluation Metrics

The evaluation metric used to properly evaluate the results of our binary classification methods is the Balanced Accuracy [84]. This metric is useful when the classes are unbalanced, i.e. one of the two classes appears a lot more often than the other. As an example, we consider a dataset composed of samples belonging to two classes: "positive" and "negative". Let us start introducing the confusion matrix shown in Figure 6.2, where:

- TP (True Positive) is the number of samples belonging to the "positive" class which are correctly predicted
- FP (False Positive) is the number of samples belonging to the "positive" class, but they predicted as "negative"
- FP (False Negative) is the number of samples belonging to the "negative" class, but they predicted as "positive"
- TN (True Negative) is the number of samples belonging to the "negative" class which are correctly predicted

Now, let us introduce the following metrics:

- Sensitivity (also known as True Positive Rate or Recall) is the number of "positive" samples detected and it is calculated as follows:

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{6.3}$$

- Specificity (also known as True Negative Rate or False Positive Rate) is the number of "negative" samples detected and it is calculated as follows:

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{6.4}$$

The balanced accuracy is the arithmetic mean of Recall and Specificity:

$$\Lambda = \frac{\text{Sensitivity} + \text{Specificity}}{2} \tag{6.5}$$

where the notation Λ indicates the balanced accuracy.

As we are dealing with a LOOCV dataset split policy, explained in Section 6.2.2, for each experiment we need to consider all the k dataset partitions. For each n-th partition, with $n = [0, ..., k-1]$, we evaluate the learned model with a proper balanced accuracy metric. So, to infer about the reliability and generalisability of our method, we introduce the Mean of Balanced Accuracies (MBA), which is the arithmetic mean among all the k balanced accuracies. It is defined as:

$$\bar{\Lambda} = \frac{1}{k} \sum_{n=1}^{k} \Lambda_n \tag{6.6}$$

where the notation $\bar{\Lambda}$ is the MBA, and Λ_n indicates the balanced accuracies of the n-th model, with $n = [0, ..., k-1]$.

6.4 Results with CNN-based method

In this section we show the results obtained with the CNN-based classification method. To evaluate the reliability of our classifier, we use the LOOCV technique, described in Section 6.2.2.

Let us consider only the lossless videos (CRF=0, resolution: 1024×512 pixel). So, for each of the $k = 10$ dataset partition, we train and test the CNN model on the *cropped-on-face* frames taken from the pre-processed grayscale videos. We remind that the prediction is computed on a temporal sequence of consecutive frames, as indicated in Section 5.3.4.

Let us evaluate each n-th model trained on the dataset $\{\mathcal{V} \setminus \mathcal{N}_n\}$ and evaluated on dataset $\{\mathcal{N}_n\}$ through the balanced accuracy metric Λ_n. In

Table 6.1 we show the balanced accuracies obtained in each configuration.

Λ_0	Λ_1	Λ_2	Λ_3	Λ_4	Λ_5	Λ_6	Λ_7	Λ_8	Λ_9
97.43%	100%	96.92%	99.25%	100%	99.25%	99.62%	100%	98.76%	99.67%

Table 6.1: Balanced accuracies calculated for each dataset split using LOOCV, in CNN-based classification.

The MBA calculated among all the balanced accuracies is 99.10%. We can notice that this model works pretty well. The main problem of the CNN-based classification method is the amount of data needed to achieve this accuracy results, as well as the high computational time and the number of physical resources required to train all these models.

Just to give an idea, the required computational time to train the model on dataset $\{\mathcal{V} \setminus \mathcal{N}_0\}$ is 2600 seconds, while the required computational time to train the feature-based model on the same dataset is less than 30 seconds. For the sake of clarity, in Section 6.5.1 we provide more details about the amount of data required by CNNs to achieve accurate results.

6.5 Results with Feature-based method

In this section we show the results obtained with the Feature-based classification method. To evaluate the reliability of our classifiers, we use the LOOCV technique, described in Section 6.2.2. So, for each of the $k = 10$ dataset partition, we train and test the MLP models on the Local Pattern Derivative on Three Ortogonal Planes (LDP-TOP) features extracted by a certain body part temporal sequence $\mathbf{S}_b^{(\rho)}$, as described in Section 5.4. In this thesis we consider only LDP-TOP extracted in *Direct* mode, because we have seen that the results obtained considering the other temporal modes are very similar. From now on, for the sake of brevity, we discuss our results only through the MBA metric. The videos considered for these configurations are the lossless videos (CRF=0) with resolution of 1024×512 pixel.

In Figure 6.3, we show the MBA with the pre-processed $\mathbf{V}^{(g)}$ as input videos. In Figure 6.4, we show the MBA with the pre-processed $\mathbf{V}^{(o)}$. as input videos. In Figure 6.5, we show the MBA considering the combined prediction technique. For each of these configurations, we report the results obtained considering the features extracted by each body parts separately and the majority voting between these predictions.

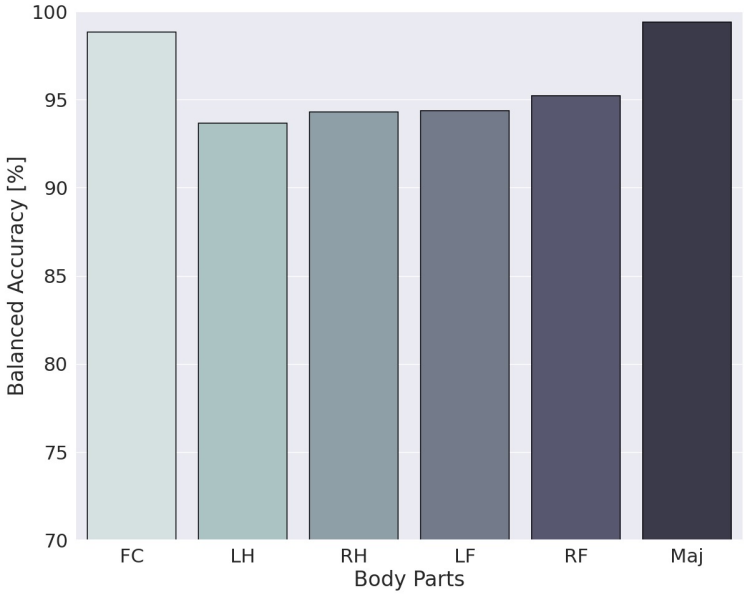

Figure 6.3: MBA in feature-based classification, considering only the pre-processed video $\mathbf{V}^{(g)}$.

Body Parts → **Technique ↓**	*FC*	*LH*	*RH*	*LF*	*RF*	maj
with $\mathbf{V}^{(g)}$	98.83%	93.68%	94.30%	94.36%	95.22%	98.99%
with $\mathbf{V}^{(o)}$	88.70%	72.59%	75.98%	84.80%	86.95%	91.28%
Combined	98.87%	93.23%	93.69%	94.49%	96.08%	**99.30**%

Table 6.2: MBA comparison in the three feature-based classification methods.

In Table 6.2 we compare the results obtained in the three different classification configurations. As we can see, the best accuracy results is obtained using the combined prediction technique. The results obtained by considering only $\mathbf{V}^{(g)}$ are very similar to the results obtained by the combined prediction technique and the ones obtained by considering only $\mathbf{V}^{(o)}$ are worse than the combined ones. For each of these configurations, we achieve the best results applying the majority voting between the body parts predictions. All the majority voting predictions achieve more than 90% of accuracy, reaching the maximum value of 99.30% with the combined prediction technique. We noticed that, between the single body parts, the classification on the face is the one which achieves the best result. This result can be explained with the greater complexity of the face textures compared to the other body parts, which means greater complexity in synthesizing all the face details and so a greater ease in

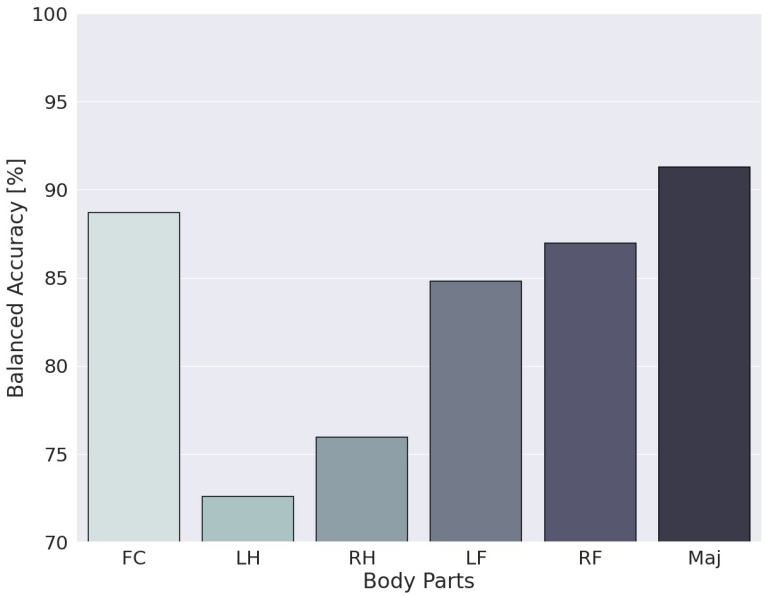

Figure 6.4: MBA in feature-based classification, considering only the pre-processed video $\mathbf{V}^{(o)}$.

separating real and fake faces.

In order to compare our results with the ones reported in [13], we replicated the same feature-based classification experiment using a linear Support Vector Classifier (SVC) [85], considering only the pre-processed videos $\mathbf{V}^{(g)}$. As we can see in Figure 6.6, the accuracy results obtained with MLP are better in case we consider only the face, the left foot, the right foot and the majority voting of the predictions. In this way, we motivate our decision to classify our LDP-TOP features with MLP instead of SVC.

6.5.1 Results varying dataset composition

One of the main issue of the methods based on neural network is that they require a big amount of data to achieve good results. So, let us see how the accuracy results change varying the number of frames where each subject in the dataset appears, for both CNN-based method and feature-based method. For the sake of brevity, we consider only the accuracy results on the faces, taking as input the pre-processed video $\mathbf{V}^{(g)}$. Let us describe the three different dataset compositions considered:

- The starting point dataset composition, discussed in Section 6.2.2.
- We consider different dataset composition where the number of considered frames is less than the one discussed at the previous

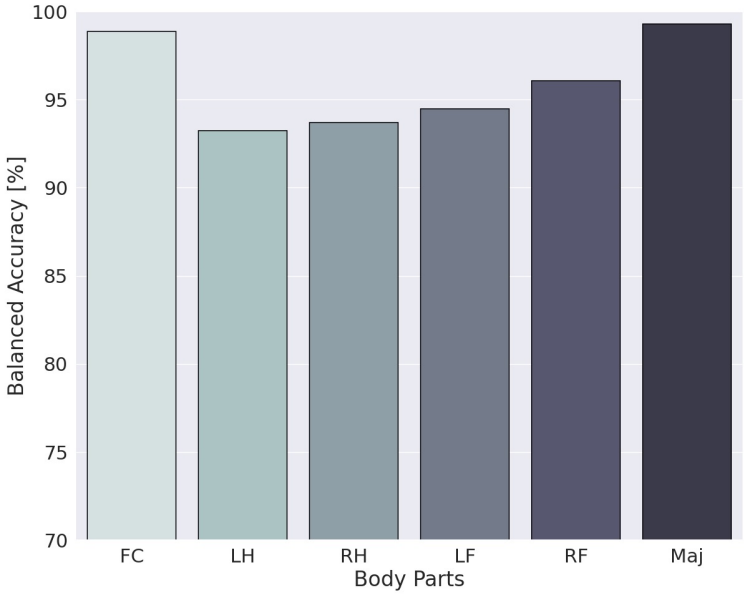

Figure 6.5: MBA in feature-based classification, with the combined prediction technique.

step. For each subject which appears in the training dataset videos, we decide to keep the first j frames referred to this subject: we take the first $\frac{j}{2}$ frames from the video labeled as "real" referred to this subject, and the first $\frac{j}{2}$ frames from the videos labeled as "fake" referred to this subject. The values chosen for j are $j = [10.000, 5.000, 2.500, 1.000]$.

- For each video in the training dataset labeled as "real" referred to a certain subject, we consider only one video labeled as "fake" where this subject appears, chosen randomly between all the synthesized videos referred to that subject. Then, the balancing step involves keeping the minimum amount of frames between the two videos. The amount of frames in each synthesized videos is always lower than the amount of frames in the corresponding original video that contains the same subject. As described in Section 6.1, the average number of frames in videos labeled as "fake" is 4012: that means that, in this dataset composition, each videos labeled as "fake" will contain an average number of 4012 frames.

As we can see in Figure 6.7, varying the number of frames in the dataset, the accuracy of the feature-based classification method remains more or less constant, while the accuracy of the CNN-based classification method decreases significantly. Our method proves to be more robust

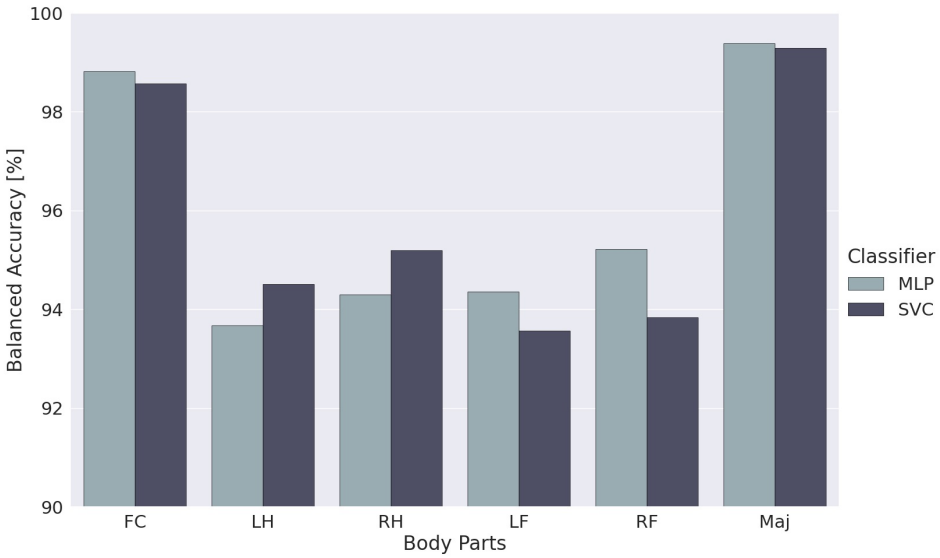

Figure 6.6: Comparison between MLP and SVC, considering only lossless videos and grayscale video as input.

than the baseline in case of limited training dataset maintaining an accuracy of over 90% in all cases, while the accuracy of CNN-based method drops below 60% in the worst case.

Then, we compare the results of the two classification methods splitting our dataset in a different way. Let us introduce the Leave-p-Out Cross-Validation (LpOCV) technique, where the parameter p refers to the number of subjects whose videos are tested, while all the other ones are used to train the model. To compare it with the LOOCV technique, we decide to consider only 10 of the possible dataset splits combinations for each LpOCV splits, chosen in a random way between all the possible combinations. Following our previous considerations, in this analysis we consider only lossless video $\mathbf{V}^{(g)}$ and the prediction done on the faces.

In Figure 6.8, we show how the accuracy of the classifiers varies as the parameter p varies. Increasing the parameter p, which means reducing the number of subjects that appear in the training dataset, the accuracy of both methods decreases in a very similar manner. For this reason, we can say that LOOCV and LpOCV techniques are comparable. As expected, the lower the number of subjects appearing in the training dataset, the lower the ability of the methods to classify videos where new subjects appear. The highest accuracy is achieved in the case $p = 1$, which corresponds to the LOOCV technique; the accuracy value remains constant around 90% up to the value $p = 5$, and then starts dropping.

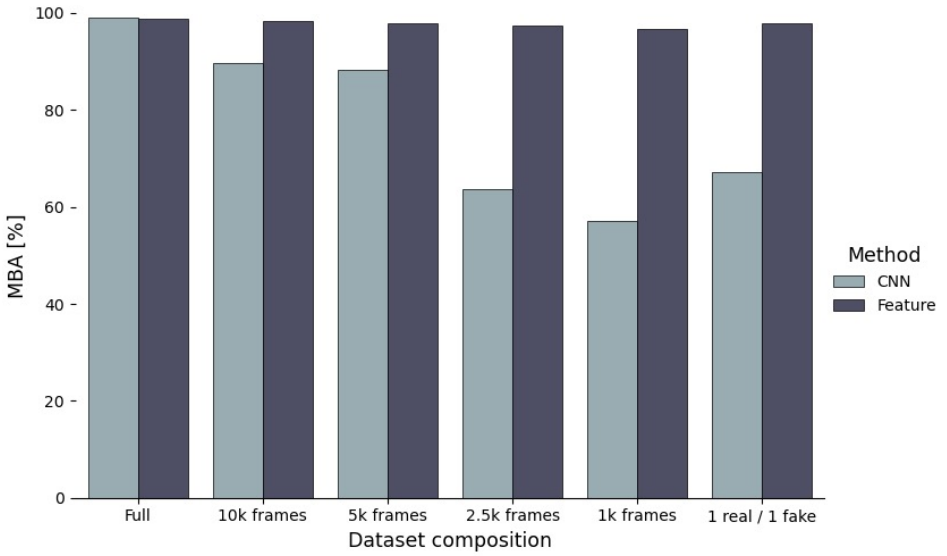

Figure 6.7: Comparison between CNN and LDP-TOP varying the number of frames per subject in the training dataset, considering only lossless videos and grayscale video as input and FC as body part. "Full" refers to the starting point dataset composition; "10k, 5k, 2.5k, 1k frames" refers to the number of frames taken for each subject in the training dataset; "1 real / 1 fake" refers to the third dataset composition.

6.5.2 Results with compressed and resized videos

In this section, we want to test the robustness of the proposed method on video compression and resizing.

Our dataset offers different compressed and resized versions of the videos, as described in Section 6.1.1. In this test we consider only the feature-based classification method. The idea is to replicate the same experiment conducted on the lossless videos, considering all the other different compressed and resized versions of the videos, and then compare these results.

For what concerns the coordinates of the bounding boxes in compressed videos, we consider the ones calculated for the corresponding lossless videos in the body parts area selection step, without recalculating them. This is because compression does not affect the position of the subject frame-by-frame, and lowering the video quality reduces the accuracy of pose detection. For what concerns the coordinates of the bounding boxes in resized videos, we consider the ones calculated for the corresponding lossless videos at full resolution and we scale them according to the following scale factors: $(\tau_w, \tau_h) = (\frac{W_{res}}{W}, \frac{H_{res}}{H})$, where τ_w is

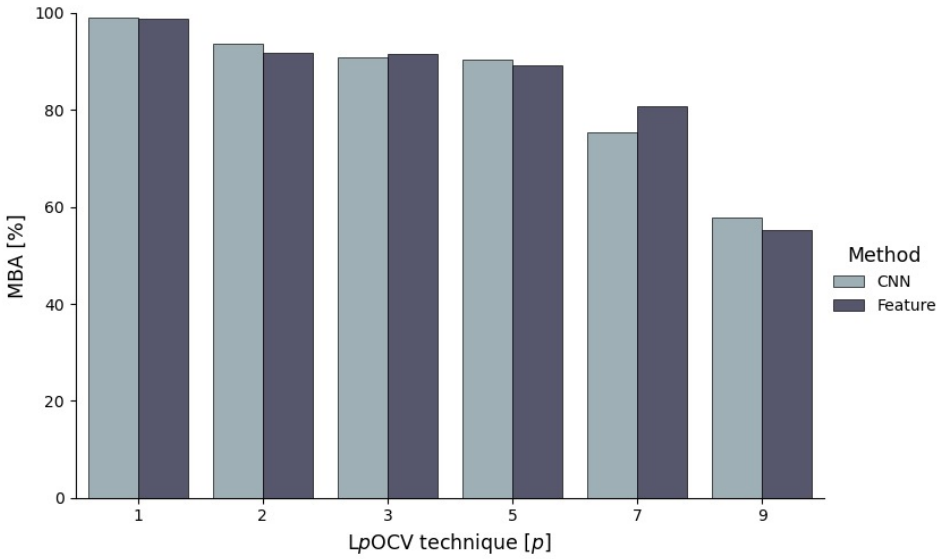

Figure 6.8: Comparison between CNN and LDP-TOP using LpOCV technique with different values of p, considering only lossless videos, grayscale videos as input and FC as body part.

the scale factor along the x-axis; τ_h is the scale factor along the y-axis; $W \times H = 1024 \times 512$ pixel is the starting point full resolution; $W_{res} \times H_{res}$ is the resolution of the resized videos.

For each pre-processed videos and for each different version of the videos, we train a MLP classifiers on each set of features extracted from a certain body part area. After the training phase, we conduct a serie of cross-tests, where the models trained on the features extracted from a certain version of videos are tested on the features extracted from all the other compressed and resized versions. For the sake of brevity, we consider only the majority voting predictions. In Figure 6.9 and Figure 6.10, we show the accuracy achieved in the cross-tests considering the pre-processed videos $\mathbf{V}^{(g)}$ and $\mathbf{V}^{(o)}$, respectively. In Figure 6.11, we show the accuracy achieved in the cross-tests using the combined prediction technique.

As we can see, the matrices are quite comparable. The accuracy results on the main diagonal give the best results. That is because the models trained on features extracted from a certain video version, works better on features extracted from the same video versions. The accuracy drops as the compression factor increases and as the pixel resolution decreases. As expected, the higher the quality of the considered videos in the dataset, the higher the accuracy results in the cross-tests.

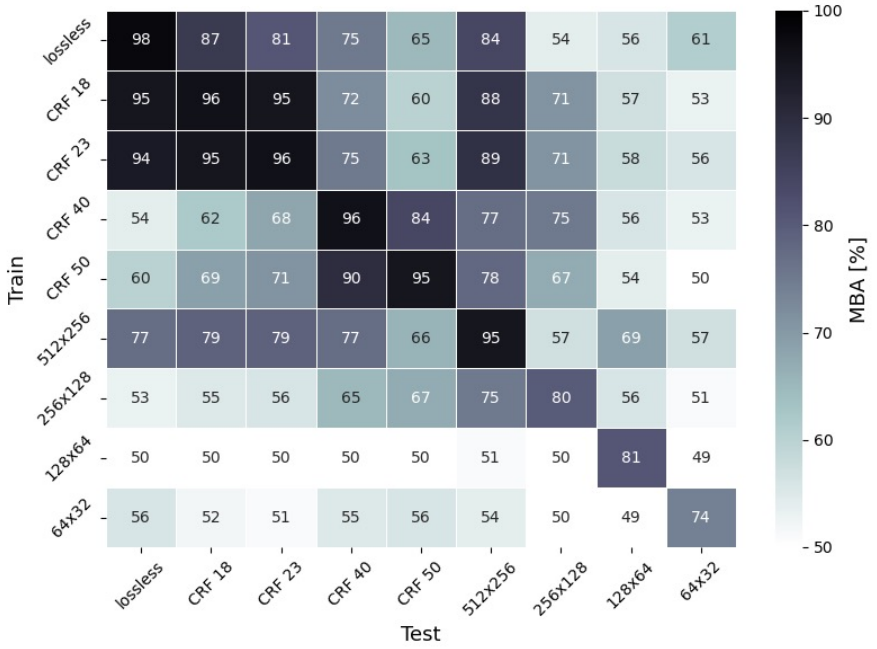

Figure 6.9: Cross-tests on different compressed and resized video versions, considering only the pre-processed video $\mathbf{V}^{(g)}$ and the majority voting between the body parts.

For what concerns the tests on compressed videos, on the main diagonal we always get over 90% of accuracy. In cross-tests, when we train on videos compressed up to CRF 23, we achieve good results in predicting the features extracted from lossless videos and higher compressions. Considering videos compressed with higher CRF, we have good results in the cross-tests involving them, but if we train on them and test on features extracted from videos with lower compression, accuracy drops. For what concerns the tests on resized videos, we get good accuracy results on the main diagonal, but the accuracy drops as the resolution decreases. In cross-tests our method is not very robust: if we train on low resolution videos, we get bad results in predicting high resolution videos, and vice-versa. The only case in which we can achieve acceptable accuracies is when we train on videos of resolution 256×128 pixel and we test on videos of resolution 512×256 pixel.

If we train on resized videos and we test on lossless or compressed videos, in general accuracy drops. The only exception is when we consider only $\mathbf{V}^{(g)}$: if we train on videos of resolution 512×256 pixel, we achieve acceptable results testing on lossless and compressed videos.

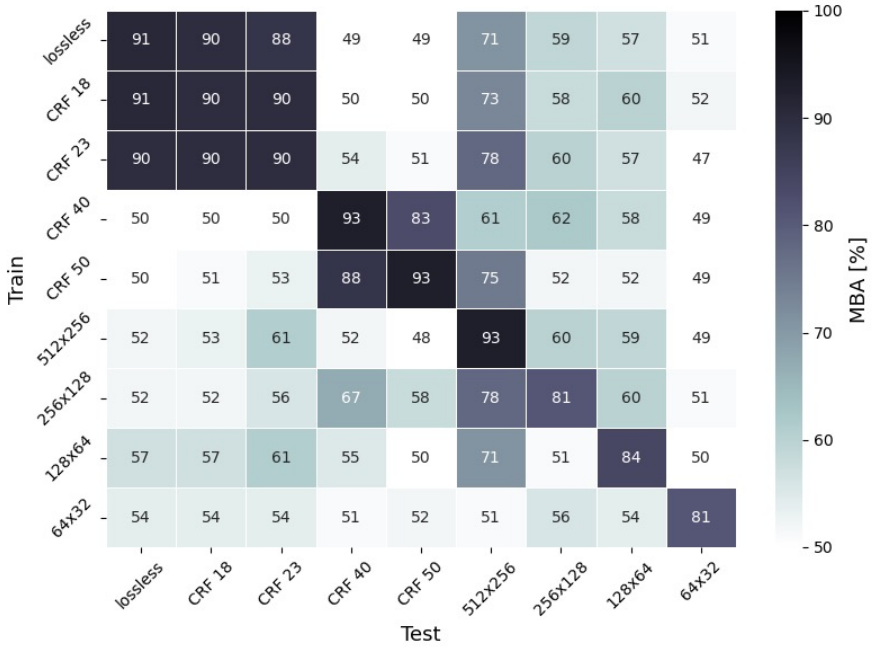

Figure 6.10: Cross-tests on different compressed and resized video versions, considering only the pre-processed video $\mathbf{V}^{(o)}$ and the majority voting between the body parts.

6.6 Conclusive Remarks

In this chapter we have evaluated the proposed methodology through simulations and experiments. We started by showing the details of the dataset we have designed for our experiments. We showed our experimental setup, focusing on temporal sequences partition useful for feature extraction, the performance evaluation of the methods, and the neural networks setup. Then we discussed our evaluation metrics. For each method, we showed the achieved accuracy results. Then, we have highlighted the strengths and weaknesses of both methods, deducing which could be the best based on a proposed scenario.

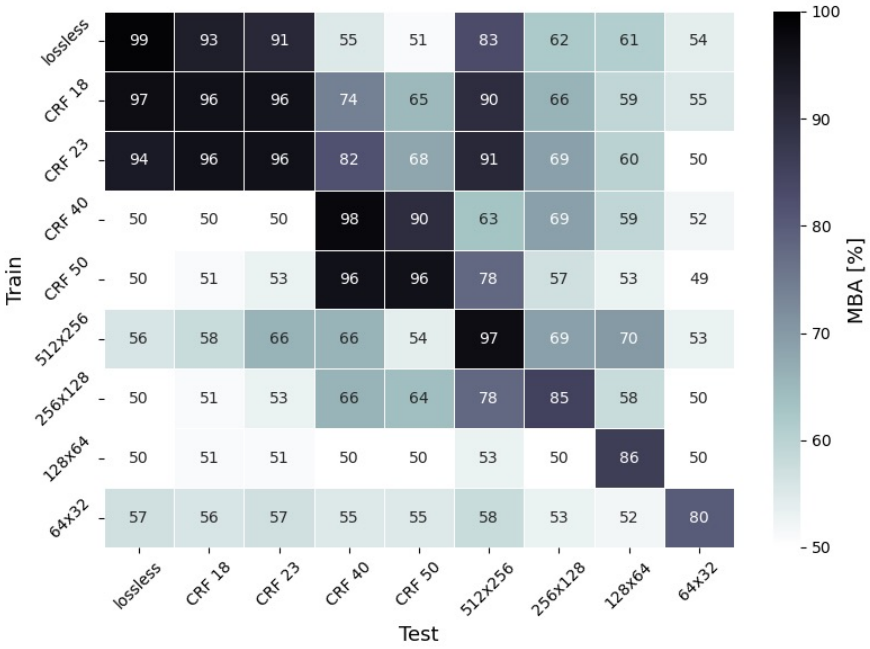

Figure 6.11: Cross-tests on different compressed and resized video versions, using combined the combined prediction technique and majority voting between the body parts.

7

Conclusions and Future Works

This thesis proposed a methodology to analyze whether a video sequence is original or fake. We have focused on synthetic videos which contain human characters moving inside the scene and whose bodies are completely synthesized through a video-to-video translation algorithm.

In the absence of a true baseline for this kind of video classification, we first introduced a CNN-based approach, based on the well-known EfficientNet-B0, which analyzes the videos in a frame-by-frame manner considering only the faces of the human subjects. Then, we proposed a method based on the extraction of temporal textural features (namely, LDP-TOP features) from video sequences and on the classification of these features with a MLP classifier.

The input videos are pre-processed in two different ways: (i) by converting them to grayscale color space; (ii) by computing the gradient of the optical flow extracted from them. For each frame, we select the regions related to five body parts (i.e. the face, the left and right hands, the left and right feet). In doing so, we can classify the video sequences by considering the predictions of the body parts features separately and by calculating the majority voting of the body parts predictions.

We created our own dataset because, to the best of our knowledge, there is not a comprehensive and publicly available dataset of synthesized human characters. The synthesized videos have been generated using the video-to-video translation algorithm *Everybody Dance Now* [6], while the

original ones come from different sources (i.e. publicly available videos, videos taken from YouTube and our camera recordings). Since we have a limited number of human subjects appearing in the dataset, we decided to evaluate the performance of our methods using Leave One Out cross validation, where the videos referred to one subject are tested, while all the others correspond to the training dataset.

High classification accuracies are achieved with both methods. With the CNN-based method, we achieved an average accuracy of 99.10%. In the feature-based method, we showed that, in general, the majority voting between the body parts predictions achieved the best accuracy results. First, we applied this method by considering the two pre-processed videos separately. We obtained very good results by considering the grayscale pre-processed videos, achieving an accuracy of 98.99% in the majority voting prediction, while we achieved an accuracy of 91.28% by considering the gradient of optical flow pre-processed videos. The best results were obtained in the combined prediction technique, where we achieved an accuracy of 99.30% in the majority voting prediction.

We compared the performances of CNN-based method and feature-based method by varying the number of samples in the training dataset. First, we showed that the performance of the first method drops when decreasing the number of training frames, while the performance of the feature-based method remains constant. Then, we evaluated the performances of the methods by reducing the number of training subjects, showing how the accuracy of both methods drops as the number of subjects appearing in the training dataset decreases.

We tested the robustness of the proposed feature-based method on video compression and resizing. We showed how our method is robust to compression, achieving accuracies of over 90% at maximum compression in every classification configuration, while with resizing performances can drop until 80%. Then, we conducted cross-tests in order to test how a model trained on features extracted from certain compressed/resized videos worked on the features extracted from other video versions. In each configuration, cross-test achieved good results when in train and test we considered lossless videos, compressed videos with CRF 18, 23, and resized videos with resolution 512×128. Surprisingly, we got good accuracy results when in train and test we considered compressed videos with CRF 40 and 51. In the other cross-tests, generally, accuracy drops.

7.1 Future Works

The results achieved in our experiments highlight a series of future challenges and improvements that can be tackled as future work.

Generalization We have not been able to test our method on a large number of human subjects or different dataset. Right now, the video generation process is very expensive in terms of computation and collecting data. We firmly believe that, in the years to come, new technologies able to generate this kind of data at a lower cost will be released, and so it will be easier to test the generalization of the model. Therefore, our method can be considered as a good starting point for the classification of this type of video.

Challenging scenarios A possible different scenario would be the one where more than a single human subject appears in the video. Moreover, one might consider the analysis of videos where the background is not homogeneous, where the camera is not static or where the subject's body is not always completely visible.

Different approaches A possible different approach to detect synthetic videos concerns the type of hybrid descriptors and the chosen selected areas for feature extraction in video sequences. It would be interesting to extract LDP-TOP or another feature from the whole frame and not from single cropped areas.

Bibliography

[1] M. Nielsen, "Neural Networks and Deep Learning." http://neuralnetworksanddeeplearning.com.

[2] S. Saha, "A Comprehensive Guide to Convolutional Neural Networks." https://tinyurl.com/ypz3henb.

[3] T. Silva, "An intuitive introduction to Generative Adversarial Networks (GANs)." https://tinyurl.com/5yme9cf5.

[4] M. FARÈ, "Deep models for unsupervised and weakly-supervised optical flow estimation," *Politecnico di Milano*, 2018.

[5] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2462–2470, 2017.

[6] C. Chan, S. Ginosar, T. Zhou, and A. A. Efros, "Everybody dance now," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5933–5942, 2019.

[7] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, "Openpose: realtime multi-person 2d pose estimation using part affinity fields," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 1, pp. 172–186, 2019.

[8] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional gans," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8798–8807, 2018.

[9] S. Sahoo, "Residual blocks—building blocks of resnet, 2018." https://towardsdatascience.com/residual-blocks-building-blocks-of-resnet-fd90ca15d6ec.

[10] T. T. Nguyen, C. M. Nguyen, D. T. Nguyen, D. T. Nguyen, and S. Nahavandi, "Deep learning for deepfakes creation and detection: A survey," *arXiv preprint arXiv:1909.11573*, 2019.

[11] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 International Conference on Engineering and Technology (ICET)*, pp. 1–6, Ieee, 2017.

[12] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[13] M. Bonomi, C. Pasquini, and G. Boato, "Dynamic texture analysis for detecting fake faces in video sequences," *Journal of Visual Communication and Image Representation*, vol. 79, p. 103239, 2021.

[14] DeepAI, "Neural Network." https://deepai.org/machine-learning-glossary-and-terms/neural-network.

[15] I. C. Education, "Neural Networks." https://www.ibm.com/cloud/learn/neural-networks.

[16] K. Gurney, *An introduction to neural networks*. CRC press, 1997.

[17] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[18] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[19] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in *Neural networks for perception*, pp. 65–93, Elsevier, 1992.

[20] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 53–65, 2018.

[21] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[22] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[23] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," *Advances in neural information processing systems*, vol. 29, pp. 2234–2242, 2016.

[24] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," *arXiv preprint arXiv:1701.04862*, 2017.

[25] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[26] T. Ojala, M. Pietikäinen, and D. Harwood, "A comparative study of texture measures with classification based on featured distributions," *Pattern recognition*, vol. 29, no. 1, pp. 51–59, 1996.

[27] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 7, pp. 971–987, 2002.

[28] M. Pietikäinen, T. Ojala, and Z. Xu, "Rotation-invariant texture classification using feature distributions," *Pattern recognition*, vol. 33, no. 1, pp. 43–52, 2000.

[29] Q.-T. Phan, D.-T. Dang-Nguyen, G. Boato, and F. G. De Natale, "Face spoofing detection using ldp-top," in *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 404–408, IEEE, 2016.

[30] T. Jabid, M. H. Kabir, and O. Chae, "Local directional pattern (ldp) for face recognition," in *2010 digest of technical papers international conference on consumer electronics (ICCE)*, pp. 329–330, IEEE, 2010.

[31] B. Zhang, Y. Gao, S. Zhao, and J. Liu, "Local derivative pattern versus local binary pattern: face recognition with high-order local pattern descriptor," *IEEE transactions on image processing*, vol. 19, no. 2, pp. 533–544, 2009.

[32] N. Paragios, Y. Chen, and O. D. Faugeras, *Handbook of mathematical models in computer vision*. Springer Science & Business Media, 2006.

[33] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *Proceedings*

of the IEEE international conference on computer vision, pp. 2758–2766, 2015.

[34] D. B. West *et al.*, *Introduction to graph theory*, vol. 2. Prentice hall Upper Saddle River, 2001.

[35] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125–1134, 2017.

[36] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European conference on computer vision*, pp. 694–711, Springer, 2016.

[37] P. J. Burt and E. H. Adelson, "The laplacian pyramid as a compact image code," in *Readings in computer vision*, pp. 671–679, Elsevier, 1987.

[38] M. Brown, D. G. Lowe, *et al.*, "Recognising panoramas.," in *ICCV*, vol. 3, p. 1218, 2003.

[39] L. Verdoliva, "Media forensics and deepfakes: an overview," *IEEE Journal of Selected Topics in Signal Processing*, vol. 14, no. 5, pp. 910–932, 2020.

[40] F. Matern, C. Riess, and M. Stamminger, "Exploiting visual artifacts to expose deepfakes and face manipulations," in *2019 IEEE Winter Applications of Computer Vision Workshops (WACVW)*, pp. 83–92, IEEE, 2019.

[41] X. Zhu, F. Che, T. Yang, T. Yu, D. Meger, and G. Dudek, "Detecting gan generated errors," *arXiv preprint arXiv:1912.00527*, 2019.

[42] S. McCloskey and M. Albright, "Detecting gan-generated imagery using saturation cues," in *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 4584–4588, IEEE, 2019.

[43] H. Li, B. Li, S. Tan, and J. Huang, "Detection of deep network generated images using disparities in color components. arxiv 2018," *arXiv preprint arXiv:1808.07276*, 2018.

[44] F. Marra, D. Gragnaniello, L. Verdoliva, and G. Poggi, "Do gans leave artificial fingerprints?," in *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pp. 506–511, IEEE, 2019.

[45] N. Yu, L. S. Davis, and M. Fritz, "Attributing fake images to gans: Learning and analyzing gan fingerprints," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7556–7566, 2019.

[46] X. Zhang, S. Karaman, and S.-F. Chang, "Detecting and simulating artifacts in gan fake images," in *2019 IEEE International Workshop on Information Forensics and Security (WIFS)*, pp. 1–6, IEEE, 2019.

[47] M. Albright and S. McCloskey, "Source generator attribution via inversion.," in *CVPR Workshops*, vol. 8, 2019.

[48] F. Marra, D. Gragnaniello, D. Cozzolino, and L. Verdoliva, "Detection of gan-generated fake images over social networks," in *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pp. 384–389, IEEE, 2018.

[49] D. Cozzolino, J. Thies, A. Rössler, C. Riess, M. Nießner, and L. Verdoliva, "Forensictransfer: Weakly-supervised domain adaptation for forgery detection," *arXiv preprint arXiv:1812.02510*, 2018.

[50] M. Du, S. Pentyala, Y. Li, and X. Hu, "Towards generalizable forgery detection with locality-aware autoencoder," *arXiv e-prints*, pp. arXiv–1909, 2019.

[51] F. Marra, C. Saltori, G. Boato, and L. Verdoliva, "Incremental learning for the detection and classification of gan-generated images," in *2019 IEEE International Workshop on Information Forensics and Security (WIFS)*, pp. 1–6, IEEE, 2019.

[52] S.-Y. Wang, O. Wang, R. Zhang, A. Owens, and A. A. Efros, "Cnn-generated images are surprisingly easy to spot... for now," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8695–8704, 2020.

[53] Z. Liu, X. Qi, and P. H. Torr, "Global texture enhancement for fake face detection in the wild," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8060–8069, 2020.

[54] D.-T. Dang-Nguyen, G. Boato, and F. G. De Natale, "3d-model-based video analysis for computer generated faces identification," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 8, pp. 1752–1763, 2015.

[55] Y. Li, M.-C. Chang, and S. Lyu, "In ictu oculi: Exposing ai created fake videos by detecting eye blinking," in *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*, pp. 1–7, IEEE, 2018.

[56] U. A. Ciftci, I. Demir, and L. Yin, "Fakecatcher: Detection of synthetic portrait videos using biological signals," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[57] Y. Li and S. Lyu, "Exposing deepfake videos by detecting face warping artifacts," *arXiv preprint arXiv:1811.00656*, 2018.

[58] X. Yang, Y. Li, H. Qi, and S. Lyu, "Exposing gan-synthesized faces using landmark locations," in *Proceedings of the ACM Workshop on Information Hiding and Multimedia Security*, pp. 113–118, 2019.

[59] X. Yang, Y. Li, and S. Lyu, "Exposing deep fakes using inconsistent head poses," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8261–8265, IEEE, 2019.

[60] S. Agarwal, H. Farid, Y. Gu, M. He, K. Nagano, and H. Li, "Protecting world leaders against deep fakes.," in *CVPR workshops*, vol. 1, 2019.

[61] D. Afchar, V. Nozick, J. Yamagishi, and I. Echizen, "Mesonet: a compact facial video forgery detection network," in *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*, pp. 1–7, IEEE, 2018.

[62] A. Rossler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner, "Faceforensics++: Learning to detect manipulated facial images," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1–11, 2019.

[63] J. Li, T. Shen, W. Zhang, H. Ren, D. Zeng, and T. Mei, "Zooming into face forensics: A pixel-level analysis," *arXiv preprint arXiv:1912.05790*, 2019.

[64] H. Dang, F. Liu, J. Stehouwer, X. Liu, and A. K. Jain, "On the detection of digital face manipulation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern recognition*, pp. 5781–5790, 2020.

[65] P. He, H. Li, and H. Wang, "Detection of fake images via the ensemble of deep representations from multi color spaces," in *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 2299–2303, IEEE, 2019.

[66] H. H. Nguyen, J. Yamagishi, and I. Echizen, "Capsule-forensics: Using capsule networks to detect forged images and videos," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2307–2311, IEEE, 2019.

[67] N. Bonettini, E. D. Cannas, S. Mandelli, L. Bondi, P. Bestagini, and S. Tubaro, "Video face manipulation detection through ensemble of cnns," in *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 5012–5019, IEEE, 2021.

[68] D. Güera and E. J. Delp, "Deepfake video detection using recurrent neural networks," in *2018 15th IEEE international conference on advanced video and signal based surveillance (AVSS)*, pp. 1–6, IEEE, 2018.

[69] S. J. Sohrawardi, A. Chintha, B. Thai, S. Seng, A. Hickerson, R. Ptucha, and M. Wright, "Poster: Towards robust open-world detection of deepfakes," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pp. 2613–2615, 2019.

[70] E. Sabir, J. Cheng, A. Jaiswal, W. AbdAlmageed, I. Masi, and P. Natarajan, "Recurrent convolutional strategies for face manipulation detection in videos," *Interfaces (GUI)*, vol. 3, no. 1, pp. 80–87, 2019.

[71] I. Amerini, L. Galteri, R. Caldelli, and A. Del Bimbo, "Deepfake video detection through optical flow based cnn," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pp. 0–0, 2019.

[72] T. Fernando, C. Fookes, S. Denman, and S. Sridharan, "Exploiting human social cognition for the detection of fake and fraudulent faces via memory networks," *arXiv preprint arXiv:1911.07844*, 2019.

[73] L. Li, J. Bao, T. Zhang, H. Yang, D. Chen, F. Wen, and B. Guo, "Face x-ray for more general face forgery detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5001–5010, 2020.

[74] D. C. G. P. L. Verdoliva, "Extracting camera-based fingerprints for video forensics," *Proc. of CVPRW*, 2019.

[75] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning*, pp. 6105–6114, PMLR, 2019.

[76] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

[77] D. Kim, "Normalization methods for input and output vectors in backpropagation neural networks," *International journal of computer mathematics*, vol. 71, no. 2, pp. 161–171, 1999.

[78] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the h. 264/avc video coding standard," *IEEE Transactions on circuits and systems for video technology*, vol. 13, no. 7, pp. 560–576, 2003.

[79] S. Tomar, "Converting video formats with ffmpeg," *Linux Journal*, vol. 2006, no. 146, p. 10, 2006.

[80] P. Refaeilzadeh, L. Tang, and H. Liu, "Cross-validation.," *Encyclopedia of database systems*, vol. 5, pp. 532–538, 2009.

[81] B. Efron, "Estimating the error rate of a prediction rule: improvement on cross-validation," *Journal of the American statistical association*, vol. 78, no. 382, pp. 316–331, 1983.

[82] S. Kotsiantis, D. Kanellopoulos, P. Pintelas, *et al.*, "Handling imbalanced datasets: A review," *GESTS International Transactions on Computer Science and Engineering*, vol. 30, no. 1, pp. 25–36, 2006.

[83] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[84] K. H. Brodersen, C. S. Ong, K. E. Stephan, and J. M. Buhmann, "The balanced accuracy and its posterior distribution," in *2010 20th international conference on pattern recognition*, pp. 3121–3124, IEEE, 2010.

[85] J. Platt *et al.*, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Advances in large margin classifiers*, vol. 10, no. 3, pp. 61–74, 1999.