Alamayreh *et al.*
EURASIP Journal on Image and Video Processing (2024) 2024:21
https://doi.org/10.1186/s13640-024-00635-2

EURASIP Journal on Image
and Video Processing

RESEARCH **Open Access**

Just Dance: detection of human body reenactment fake videos

Omran Alamayreh[1*], Carmelo Fascella[2], Sara Mandelli[2], Benedetta Tondi[1], Paolo Bestagini[2] and Mauro Barni[1]

*Correspondence:
omran@diism.unisi.it

[1] Department of Information Engineering and Mathematics, University of Siena, Siena, Italy
[2] Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milan, Italy

Abstract
In the last few years, research on the detection of AI-generated videos has focused exclusively on detecting facial manipulations known as deepfakes. Much less attention has been paid to the detection of artificial non-facial fake videos. In this paper, we address a new forensic task, namely, the detection of fake videos of human body reenactment. To this purpose, we consider videos generated by the "Everybody Dance Now" framework. To accomplish our task, we have constructed and released a novel dataset of fake videos of this kind, referred to as FakeDance dataset. Additionally, we propose two forgery detectors to study the detectability of FakeDance kind of videos. The first one exploits spatial–temporal clues of a given video by means of hand-crafted descriptors, whereas the second detector is an end-to-end detector based on Convolutional Neural Networks (CNNs) trained on purpose. Both detectors have their peculiarities and strengths, working well in different operative scenarios. We believe that our proposed dataset together with the two detectors will contribute to the research on the detection of non-facial fake videos generated by means of AI.

Keywords: DeepFake, FakeDance, GANs, Multimedia forensics

1 Introduction

The widespread diffusion of manipulated media content has become a central problem in recent years, especially after the advent of the so-called *deepfakes*. These are videos (but in principle could be images or audio recordings) generated through (DL) techniques where the identities, intentions and emotions of portrayed human characters are altered and swapped with those of other targets [1, 2]. The ability to create these media contents with minimum user intervention, together with the widespread use of social media raises enormous concerns about the authenticity and the credibility of what we see. The consequences of these fake media on our society could be devastating, changing virtually the meaning of evidence in all domains ranging from everyday life, through journalism, criminal justice, and national security.

The necessity of distinguishing between fake and original media has attracted the interest of researchers all around the world. However, most of the attention has been devoted to the detection of *deepfake* videos [1]. The detection of non-facial fake videos generated by means of DL has received much less attention [3].

To move a first step to fill the gap, in this paper we focus on the forensic analysis of a new class of fake human motion transfer videos, hereinafter referred to as FakeDance videos [4]. These videos represent the body of a person (target person or subject), moving with the motion inferred from another person (source person or subject). Even though creating these videos with friends looks like just a fun and entertaining activity, we expect that in the near future they could be used in a harmful way [5]. For example, to alter crime scenes videos or to create fake surveillance videos. In the light of this, the contribution of our paper is twofold: the generation of the first dataset of fake videos of human body reenactment to train and test forensic methodologies; the proposal of two forensic detectors aiming at studying the detectability of FakeDance videos. Concerning the data generation contribution, we generate four sub-datasets. In each of the sub-datasets, we mitigate one or more of the limitations presented in *Everybody Dance Now* framework [4]. We generate 594 videos using different generation and compression settings. The dataset contains the fake videos together with the original real videos used to train the *Everybody Dance Now* technique. Regarding the detectability of FakeDance videos, we propose two complementary detectors following different approaches. One detector starts from a hand-crafted feature extraction process, limiting the data-driven part to the final classifier, while the other detector is purely data-driven and consists of a Convolutional Neural Network (CNN) trained on purpose.

To evaluate the generalization capability of both detectors and their ability to work in non-ideal conditions, we considered both raw sequences and videos compressed at different quality levels. Additionally, we have carried out the training in data scarcity conditions, in which only a few training examples are available. The experiments prove the validity of the proposed detectors in different settings. Results exhibited an extremely high accuracy, demonstrating that fake videos of the FakeDance dataset can be reliably distinguished from real videos.

The rest of the paper is organized as follows. In Sect. 2, we briefly review the state of the art of deepfake generation and detection. In Sect. 3, we present our FakeDance dataset. In Sect. 4, we describe the proposed forgery detectors. The results of the experiments we carried out are reported and discussed in Sect. 5. Finally, we conclude the paper with our remarks and some clues for future research lines in Sect. 6.

2 Related work

In this section, we review the most common approaches for synthetic fake video generation including human reenactment videos. Then, we discuss state-of-the-art techniques to detect this kind of videos, including deepfakes.

2.1 Generation of fake videos

Most of the methods for synthetic fake video generation focus on the synthesis of human faces and facial movements. This is the case of well-known deepfake videos as well. In this context, one of the first attempts of deepfake creation is FaceSwap, developed using autoencoder–decoder pairing structure [6]. In this method, the autoencoder extracts latent features from face images, while the decoder is used to reconstruct face images. This approach is applied in other works such as DeepFaceLab [7] and DFaker [8]. High quality deepfakes based on Generative Adversarial Network (GAN) called

Faceswap-GAN [9] have been obtained by adding adversarial loss and perceptual loss implemented in VGGFace [10] to the encoder–decoder architecture. The VGGFace perceptual loss is added to make eye movements to be more realistic and consistent with input faces. Face2Face [11] is a real-time facial reenactment method for monocular target video sequences, able to animate the facial expressions of the target video by a source actor and re-render the manipulated output video in a photo-realistic fashion. Another interesting work is Neural Voice Puppetry [12], which is a method for audio-driven facial video synthesis. It is able to synthesize videos of a talking head from an audio sequence of another person using 3D face representation.

In addition to video generation techniques to manipulate human faces, recent works also focus on other aspects and applications. This is the case of [13], in which the authors propose a general-purpose video-to-video synthesis technique. This method can be used not just for faces, but also to generate fake video acquisitions from car dash cameras. Another notable example is the work proposed by Chan et al. in [4]. The authors propose a video synthesis generator which can transfer the motion from one subject to another one, thus making a person's body move at will. Considering the social impact that videos generated with this technique may have, we decided to focus on this specific video generator, which will be further detailed in 3.

2.2 Fake videos detection

As far as the detection of fake videos generated by GANs [14] is concerned, the forensic community has put considerable effort in the last few years. However, most of the attention has been devoted to the detection of fake facial videos as deepfakes [1]. Commonly, the developed techniques could be classified under two main categories: handcrafted features methods and DL-based methods. Handcrafted methods exploit a set of predefined traces that result from the production pipeline of deepfake videos. For instance, Matern et al. [15] designed the detector using several visual features that focus on the eyes, teeth, facial contours visual artifacts. Yang et al. [16] exploited the semantic inconsistencies of head pose to detect fake videos. Li et al. [17] exposed deepfake videos through unnatural eye blinking signals. Modern DL techniques based on CNN permit to get powerful solutions. Methods based on MesoInception [18], Capsule [19], XceptionNet [20], EfficientNet [21, 22], RNN [23] and LSTM [24] provide superior performance compared to the traditional handcrafted ones.

Concerning the detection of non-facial manipulations, only a few works have been proposed to the best of our knowledge. The authors of [3] have proposed a method to detect GAN-synthesized street videos. Considering human body reenactment fake videos generated in *Everybody Dance Now* [4], a forgery detector based on a CNN is also considered by the same authors. Such detector corresponds to the discriminator used in the GAN framework to create the fake videos, which has been trained in parallel with the generator. However, we can argue that in a real-world scenario the forgery detector does not know the details of the generation process of the fake media, and then any practical detector has to be trained separately from the generator, only exploiting the availability of fake videos generated by the model. The lack of methods specialized for non-facial manipulation detection highlights that some efforts must be put in this direction.

3 Human body reenactment video dataset

To properly develop and test forensic detectors for fake human body reenacted videos, a well-defined dataset is needed. For this reason, we release a new dataset, namely the FakeDance dataset, including original and synthetic videos in which the scene presents an in-the-wild single dancer filmed by a static camera. In particular, the synthetic videos depict human subjects whose dance motion has been automatically transferred from a source subject. We generate the synthetic videos by leveraging the video-synthesis algorithm introduced by Chan et al. [4], defined as the *Everybody Dance Now* generation model. Given a source video with a person dancing, *Everybody Dance Now* can generate a new synthetic video in which the source motion is transferred to a target person.

The video synthesis model is based on GANs, in particular on the image-to-image translation pix2pixHD model introduced by Wang et al. [25]. The model consists of a generator and a discriminator which are trained simultaneously and drive each other to improve. The generator learns to synthesize single frames of a person given a pose stick figure, while the discriminator learns the differences between the generated frames and the ground truth ones. With respect to the original pix2pixHD, the image generator of [4] is modified to enforce temporal coherence between adjacent frames. Moreover, a specialized GAN is exploited to increase the realism of the generated facial region.

In the following, we provide additional details about the necessary steps to synthesize a video starting from a source video and a target one. Then, we describe the FakeDance dataset, providing information about the videos included and their characteristics.

3.1 Video synthesis pipeline

As depicted in Fig. 1, the video synthesis pipeline proposed in [4] is divided into two main stages: a *training phase* and a *transfer phase*. Given a target person, in the training phase, a model has to be trained that learns to transfer the motion from a pose stick figure, obtained through a pose detection step, to the target person. In the transfer phase, given a source video and a target video, the motion is transferred from the source subject to the target subject. The steps required for synthesizing new videos are as follows:

Pose detection: The body poses of both source and target person must be extracted in order to generate the synthesized video. As suggested in [4], we exploit the OpenPose [26] detector to accurately estimate the 2D coordinates of the body skeletons. Given these coordinates, we can create a pose stick figure for each video frame.

Model training: Given a target video, we have to train a motion transfer model related to the target subject, providing as input the pose stick figures extracted from the video frames, together with the target video frames, that are used as reference by pix2pixHD [25]. The trained model learns to map a pose into a video frame that looks like a frame from the target subject video.

Pose normalization: Different humans might have different body shapes with different body parts proportions. Additionally, the distance between the video camera and

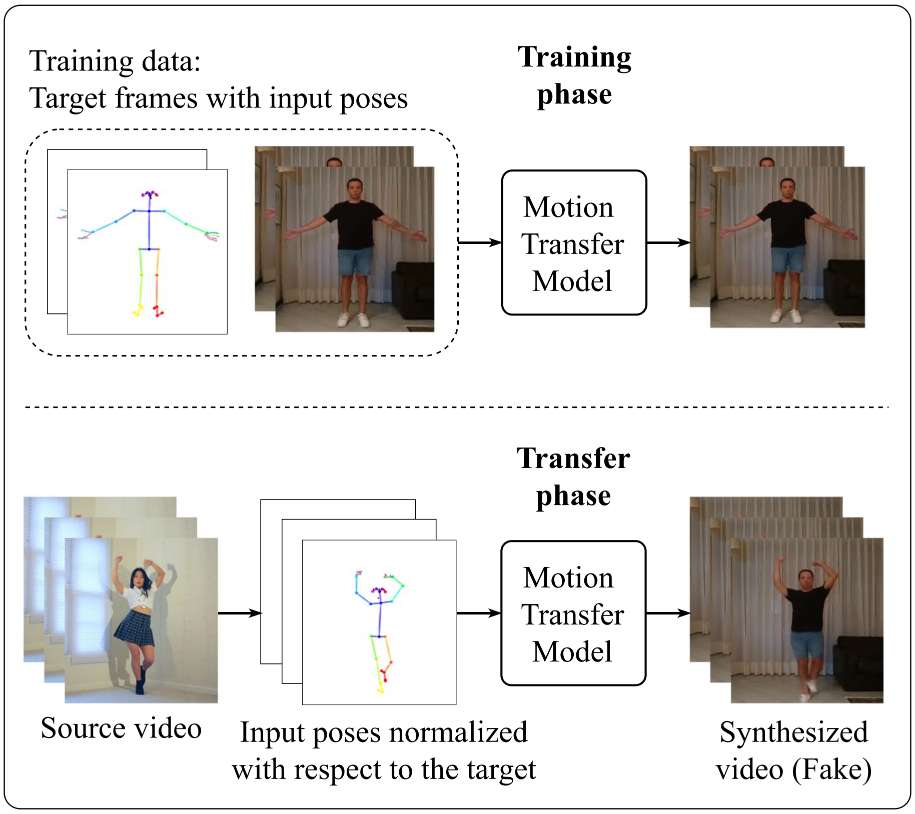

Fig. 1 Video synthesis pipeline. In the training phase, a motion transfer model learns to transfer the motion to the target subject from pose stick figures and video frames. In the transfer phase, the motion is transferred from the source to the target subject by feeding the model with normalized pose stick figures of the source person

the subject might differ from video to video. To accurately transfer the motion from the source to the target subject, we should take these considerations into account. Therefore, it might be necessary to modify the skeleton points of the source subject, so that they are "consistent" with the target subject. As suggested in [4], we apply a linear transformation to all pose key-points of the source subject for every video frame.

Reenactment of body motion: Once the source pose is normalized as detailed above, we can feed the normalized pose to the trained model. The network output consists of a sequence of synthesized video frames, in which the motion of the source subject is transferred to the target person. We generate the final synthesized video by converting the synthesized video frames into a H.264 video sequence through FFmpeg [27].

For the sake of brevity, we do not report the implementation details on the four main steps above. The interested reader may refer to the original work [4].

3.2 FakeDance dataset

The proposed FakeDance dataset is composed of four sub-datasets having different characteristics, that are described in the following.

Swap_dance: This sub-dataset has a total of 20 videos, 10 real and 10 fake ones. The real videos depict 10 different dancers and include 5 original video sequences provided by [4], 2 videos selected from YouTube and 3 videos that have been filmed by ourselves, all longer than 6 min. The synthetic videos have been generated by following the steps reported in Sect. 3.1.

We considered 10 target subjects, corresponding to the persons appearing in the real videos. Concerning the source subjects, we collected 5 additional video sequences from YouTube (average length $\sim$ 2 min) depicting different dancing persons from the target subjects. We transferred the motion of each source subject into the target videos, ending up with 50 fake videos. To obtain an equal number of real and fake videos for every target, we selected only the synthetic video with the highest visual quality out of the 5 available ones. As the fake videos present significantly shorter length than original target videos, we crop every original video to have the same temporal duration of its related fake version.

Puppet_dance: In this sub-dataset, the source and the target subjects used to synthesize the videos are the same. To this purpose, we exploit the 5 original sequences provided by [4]. For every original video, we generate one fake video with the same subject and the same motion.

To get the synthesized videos, we temporarily crop each original video by separating the first 70% of its frames from the remaining 30%. We exploit the 70% of the frames (with the corresponding skeleton poses) for training the transfer model. Then, we use the other 30% for the transfer phase, i.e., we use them as source into which transferring the motion. In doing so, we end up with 5 fake videos with length equal to 30% that of the original sequences. To obtain a balanced dataset, i.e., in which the temporal duration of real and fake videos is the same, we include in the "real" class only the 30% portion of the original sequences (the same used for the transfer phase).

The *Puppet_dance* sub-dataset simulates a potentially worrying scenario, in which the original motion of a target person is maliciously modified by exploiting other sequences depicting the same subject while she/he is moving differently. The threat is represented by out of context new motions transferred to the target video.

It is worth noticing that, even if target and source subjects are the same, *Puppet_dance* considers different input skeleton poses at the training and transfer phases. Thus, the transfer model is not completely overfitted on the input data seen at the transfer phase.

Differently from *Swap_dance*, *Puppet_dance* does not need the pose normalization step to generate synthetic videos, as the source and target subject is the same. As a consequence, no normalization errors are introduced and the synthesized videos have better visual quality.

Fit_dance: As done for *Puppet_dance*, the *Fit_dance* sub-dataset also considers pairs of the same source and target subjects to synthesize new fake videos. However, contrarily to *Puppet_dance*, we use *all* the frames of the original videos to train the transfer model for each subject. In the transfer phase, we feed the generative model with the same data already seen during training. Notice that the *Fit_dance* sub-dataset includes synthesized videos with high visual quality, reasonably better with respect to the fake videos in *Swap_dance* and *Puppet_dance*. Indeed, the transfer model is completely overfitted on the input data seen at transfer phase. Moreover, in this scenario as well, the pose normalization step is not required, thus we do not introduce further errors in the video synthesis.

The original data used for generating *Fit_dance* consist in 74 video sequences downloaded from YouTube, depicting different dancing subjects.

Splice_dance: This sub-dataset includes 20 videos, 10 real and 10 fake, randomly selected from the test set portion of the *Fit_dance* sub-dataset, in which we modify the background of the fake videos. More specifically, for each fake video frame, we select the pixel area of the synthesized human subject and paste it into the background scene of the corresponding original video. The pixel region around the subject is selected by dilating its corresponding pose stick figure.

The *Splice_dance* dataset enables to investigate the challenging scenario in which the video under analysis might not be completely synthetic, but could have been doctored with the insertion of a small synthetic portion, e.g., a human subject which was not depicted in the original scene.

All the video sequences composing the four sub-datasets present common video codec parameters and pixel dimensions. Since we generate H.264 fake sequences from the synthesized video frames, we re-encode all the original videos using FFmpeg [27] with the same parameters used for the fake videos. We end up with a dataset including video sequences with common resolution of $1024 \times 512 \times 3$ with a frame rate of 25 frames per second. Whenever the original videos have different dimensions, we process them to satisfy these conditions.

The four sub-datasets are always balanced for what concerns the temporal duration of the real and fake videos, i.e., any pairs of real-fake sequences present the same number of video frames. A summary of all the sub-datasets of the FakeDance dataset is provided in Table 1, while Fig. 2 shows a few selected video frames. The FakeDance dataset has been made publicly available and can be found at https://bit.ly/3J2IENp.

It is worth stressing that building our FakeDance dataset required a noticeable computational effort. Indeed, a model has to be trained for each target person. Different training times have been observed to be necessary to reach a good visual quality of the synthetized videos, also depending on the video length. As an example, it took 3 days to train a model to generate a 7 min video for a target person using a 12GB Quadro M6000 GPU, while for our longest input video—that is 20 min long—it took

Table 1 Summary of the four sub-datasets composing the FakeDance dataset

	Swap_dance	***Puppet_dance***	***Fit_dance***	***Splice_dance***
Video subjects	Source ≠ Target	Source = Target	Source = Target	Source = Target
Video motion	Real ≠ Fake	Real = Fake	Real = Fake	Real = Fake
Pose normalization	Required	Not required	Not required	Not required
GAN input pose	Training phase ≠ Transfer phase	Training phase ≠ Transfer phase	Training phase = Transfer phase	Training phase = Transfer phase
Fake videos background	Synthetic	Synthetic	Synthetic	Original
No. real videos	10	5	74	10
No. fake videos	10	5	74	10
Avg. duration [min:s]	02:31	02:21	02:32	03:22

All the videos included have a common resolution of $1024 \times 512 \times 3$ pixels. In all the scenarios, every real video and its corresponding fake version have the same temporal length

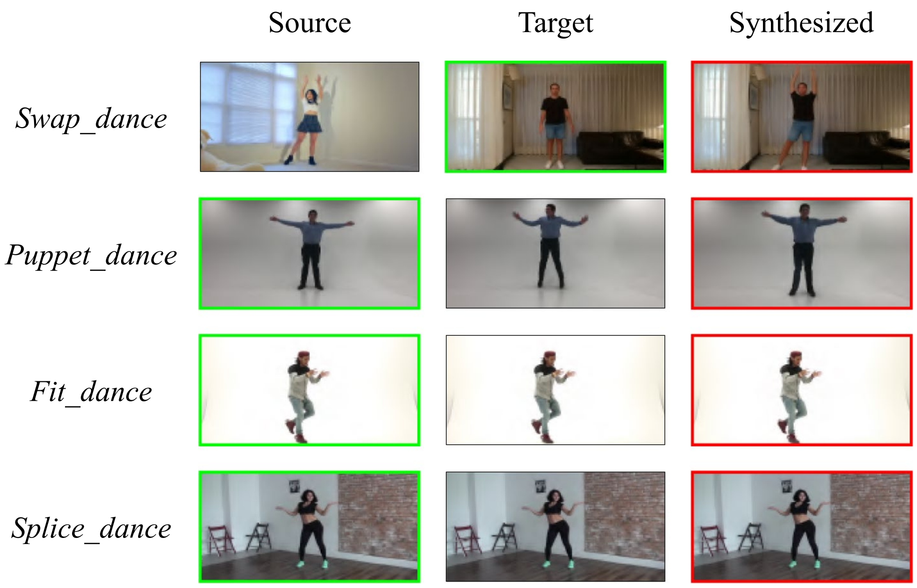

Fig. 2 Video frames selected from the FakeDance dataset. In green and red, the frames related to the "real" class and the "fake" class, respectively

over 2 weeks to train a model for a single target person using a 24GB Quadro P6000 GPU.

3.3 Fake video quality assessment

In this section, we assess the quality of the generated fake videos by computing the similarity between the generated videos and their corresponding real counterpart. In particular, referring to Fig. 2, notice that we can evaluate the similarity between real and fake content for *Fit_dance*, *Puppet_dance*, and *Splice_dance* sub-datasets, where the motion of the subject in the real and fake sequences is matched. In the case of *Swap_dance* sub-dataset, the real and fake subjects perform completely different movements, thus it is more challenging to evaluate the quality of the synthesized videos.

For what concerns *Fit_dance* and *Puppet_dance* sub-datasets, which consist of fully generated sequences, we compute the Structural Similarity Index Measure (SSIM) [28] between the real and fake video frames. The SSIM score ranges from 0 to 1, where higher values indicate better generation quality.

In the case of the *Splice_dance* sub-dataset, given that videos are only partially synthetic (i.e., the background is real), we compute a modified SSIM version inspired by the Mask-SSIM metric proposed in [29]. This metric was originally developed for facial deepfake evaluation; to evaluate the quality of the generated video frames, a mask is drawn to select the facial area and the SSIM is computed between the real and fake facial pixels. We adopt this metric to work on *Splice_dance* videos, considering a mask that matches the subject motion in every video frame. Then, we compute the SSIM only between real and fake video frames by confining the computation to the pixels in the mask region. In doing so, we avoid being biased by the background pixels, which are exactly the same between real and fake sequences.

Table 2 Average similarity scores of different FakeDance sub-datasets

Sub-dataset	*Fit_dance*	Puppet_dance	Splice_dance (Mask-SSIM)
SSIM	0.92	0.80	0.93

In case of *Fit_dance* and *Puppet_dance*, we computed the SSIM full-frame; in case of *Splice_dance*, we computed the Mask-SSIM, considering only the subject area

In all the considered scenarios, we calculate the similarity by sampling one video frame per second. Then, we average the achieved values for each frame pair to obtain a global similarity value, measuring the quality of each fake video. The average similarity values obtained for *Fit_dance*, *Puppet_dance* and *Splice_dance* are reported in Table 2.

We see that the obtained results reflect the differences in the production pipeline for each sub-dataset. Specifically, *Fit_dance* and *Splice_dance* videos have superior quality with respect to *Puppet_dance* videos. We attribute this to the consistency between the input pose during the training phase and the transferred pose, that characterizes *Fit_dance* and *Splice_dance* videos. In contrast, in *Puppet_dance*, the input poses in the training and transfer phase differ, yielding lower quality results.

3.4 Limitations of the FakeDance dataset

Overall, we can highlight a few potential limitations of our proposed FakeDance dataset. Indeed, the quality of the generated fake videos can be influenced by several factors. One of these is the background of the target video. If the target video lacks a static background, it may lead to challenges in maintaining a consistent and high-quality background in the generated fake videos. Additionally, the Everybody Dance Now framework [4] may face difficulties in accurately portraying individuals wearing loose clothing or with intricate hairstyles. The current framework might struggle to effectively convey such details, potentially resulting in discrepancies or distortions in the appearance of clothing and hair in the generated fake videos.

Moreover, several other elements can influence the quality of the generated fake videos. These include the length of the target video, the variations in human body shapes, and the pose normalization procedure that does not accommodate varying camera positions or extreme poses, such as handstands, within the source or the target video. Such variations could lead to artifacts or inconsistencies in the generated fake videos, affecting overall quality.

4 Proposed methodology

In this section, we introduce the two detectors built to distinguish pristine from fake videos of motion transfer. The first detector is based on handcrafted features and exploits the spatial–temporal information of the video sequences, while the second one is based on a CNN architecture and works on a frame basis. As we show in Sect. 5, both detectors work well, each of them having strengths and weaknesses that make them suitable in different testing conditions.

4.1 Features-based detector

The first proposed method is based on the extraction of handcrafted features known in the literature as Local Derivative Pattern on Three Orthogonal Planes (LDP-TOP) [30].

In a nutshell, given a query video sequence, the proposed detection method is based on two main steps: (i) pre-processing and body parts selection, in which the video is pre-processed in two different ways and five body parts are selected; (ii) feature extraction and classification, in which spatial–temporal features are extracted from the selected video and fed to a classifier. We propose to fuse the contributions of the pre-processing applied and to aggregate the predictions related to the extracted body parts to obtain a final classification score related to the entire video sequence.

4.1.1 Pre-processing and data selection

We propose to pre-process each video $\mathbf{V}$ in two different ways to capture as many visual cues as possible: (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 [31]. The information contained in $\mathbf{V}^{(o)}$ enhances the edges of the movement of the subjects. We call the generic pre-processed video as $\mathbf{V}^{(\rho)}$, where $\rho \in [g, o]$ indicates if we are considering $\mathbf{V}^{(g)}$ or $\mathbf{V}^{(o)}$.

We select frame-by-frame specific areas of each video $\mathbf{V}^{(\rho)}$ corresponding to relevant body parts of the subjects. The idea is to analyze the texture of subjects' parts that contain important details which are useful for the recognition of the person and might be more difficult to synthesize. The chosen body parts 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 body parts for each frame, we first estimate the skeleton coordinates of the subject by means of the *Openpose* detector [26]. Then, following the approach suggested in [4], we exploit the skeleton coordinates to estimate the position of the bounding box surrounding each body part. For every body part, we spatially crop each pre-processed video sequence around the selected body part, defining a new sequence $\mathbf{V}_b^{(\rho)}$, where $b \in \mathcal{B}$.

4.1.2 Feature extraction and classification

Once selected the body parts, for each of them and for both pre-processed video versions (i.e., for every sequence $\mathbf{V}_b^{(\rho)}$), we extract some handcrafted features, namely the LDP-TOP [30], recently introduced as a pattern descriptor for face recognition.

The LDP-TOP is the evolution of the Local Derivative Pattern (LDP) feature [32] already proposed for natural 2D images. To extract LDP features from a 2D image, we have to consider the 3×3 pixel neighborhood surrounding each pixel. We must compute the difference between each pixel and its adjacent at 0° (i.e., the closest on the right) and replace its value with this difference. Then, a sign comparison is done between the values of the surrounding pixels and the central one, replacing these values with 0 if the central pixel is smaller, 1 otherwise. For each pixel, an 8-bit binary vector is obtained. The LDP feature extracted from the image is the histogram of the binary vectors related to the pixels, consisting of 2^8 elements.

The LDP-TOP feature exploits both spatial and temporal domains in the analysis of gray-scale video sequences. Given a video with dimensions $H \times W \times T$ (H, W being the spatial dimensions and T the temporal one), we have to consider the three central 2D arrays along each dimension, i.e., the 2D arrays with dimensions $H \times W$, $W \times T$ and $H \times T$ found by temporally and spatially cutting the entire video in $T/2$, $H/2$ and

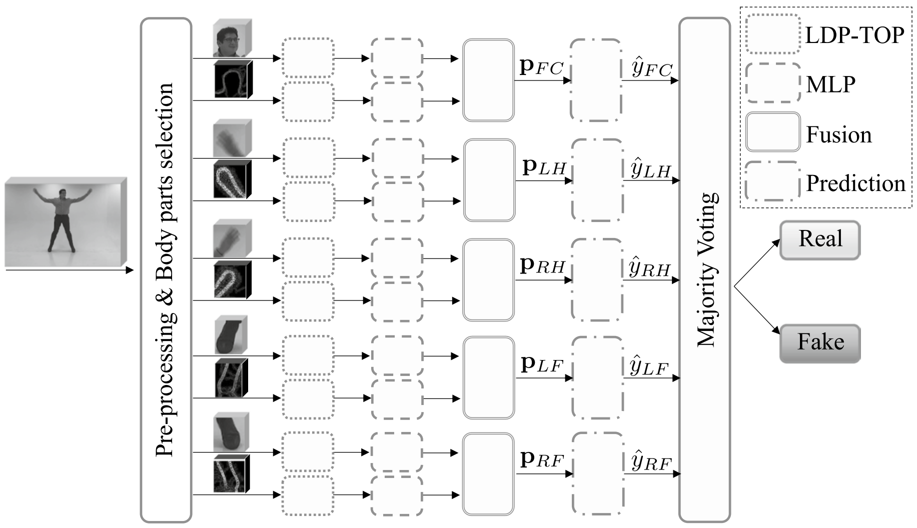

Fig. 3 Feature-based classification pipeline. Given a video sequence, we pre-process it in two ways and we extract sequences related to five different body parts. Then, we extract LDP-TOP feature from each sequence, we pass it through MLP classifier and finally perform prediction through fusion and majority voting

$W/2$, respectively. For every 2D array, we must extract four LDPs considering four adjacency angles in the calculation of pixel differences (0°, 45°, 90°, 135°) and concatenate them. By concatenating the three resulting features of the 2D arrays, we obtain the LDP-TOP feature of the video, consisting of $2^8 \times 4 \times 3 = 3072$ elements.

After the extraction of the LDP-TOP from each video $\mathbf{V}_b^{(\rho)}$, we propose a classification method to predict the class of the query video $\mathbf{V}$ by combining all the contributions of the two pre-processing and the five body parts. The proposed pipeline consists of five main steps (see Fig. 3):

1. *Feature extraction*: Given a video sequence $\mathbf{V}_b^{(\rho)}$, we extract the LDP-TOP feature vector defined as $\mathbf{L}_b^{(\rho)}$.
2. *Classification*: The features $\mathbf{L}_b^{(\rho)}$ are fed to one multi-layer perceptron (MLP) classifier, in order to predict their class. The output vector $\mathbf{p}_b^{(\rho)} = [(\mathbf{p}_b^{(\rho)})_0, (\mathbf{p}_b^{(\rho)})_1]$ indicates the probability of the sequence $\mathbf{V}_b^{(\rho)}$ to belong to the classes “real” and “fake”.
3. *Fusion*: For each $b \in \mathcal{B}$, we calculate the probability associated with the sequence $\mathbf{V}_b$ as the average between the probability calculated for the corresponding sequences $\mathbf{V}_b^{(g)}$ and $\mathbf{V}_b^{(o)}$:

$$\mathbf{p}_b = \frac{\mathbf{p}_b^{(g)} + \mathbf{p}_b^{(o)}}{2}. \tag{1}$$

4. *Prediction*: we compute the predicted label $\hat{y}_b$ of the sequence $\mathbf{V}_b$ as:

$$\hat{y}_b = \arg\max_{i \in [0,1]} (\mathbf{p}_b)_i. \tag{2}$$

5. *Majority voting*: the final video prediction $\hat{y}$ is computed as the majority voting between all the predictions $\hat{y}_b$ related to each body part:

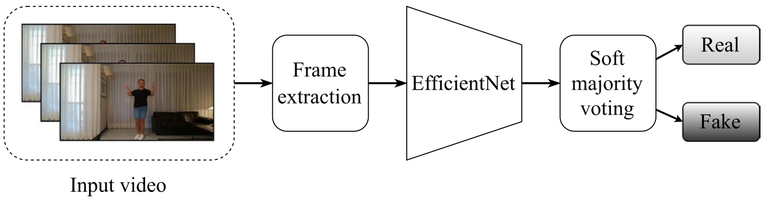

Fig. 4 Detection pipeline of the CNN-based detector

Table 3 Number of videos in each sub-dataset

	Training	Validation	Testing
Swap_dance	16	0	4
Puppet_dance	8	0	2
Fit_dance	108	14	26
Splice_dance	0	0	20

Every set contains an equal number of real and fake videos

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

The operator maj(·) returns the most recurring value between the input predictions. Furthermore, we also provide a probability score p associated with the synthetic class for the analyzed video sequence $\mathbf{V}$. This is computed by soft majority voting on the probability scores related to each body part (see (1)). In practice, the score p is the arithmetic mean of the body part probabilities associated with the synthetic class.

4.2 CNN-based detector

Following a common approach in the state-of-the-art [21, 22], we propose a completely data-driven synthetic video detector based on CNNs working on a frame level. In particular, the best detection results have been obtained using an EfficientNet-B3 architecture [33] as backbone network for the proposed detector, pre-trained on Imagenet [34]. EfficientNet relies on a scaling method that uniformly scales depth, width, and resolution of the network, adapting them to the input image, exploiting the intuition that if the image is bigger than the network needs more layers to get a similar receptive fields and more channels to capture patterns having the same graininess. It has been shown that EfficientNet achieves better accuracy and efficiency and requires less parameters with respect to other well-known CNNs [33].

The EfficientNet-B3 is fed with the frames extracted from the videos. During testing, given an input query video, we first extract the video frames. Then, we aggregate the scores associated with the frames to achieve a final score for the prediction. The decision on the whole video (sequence of frames) is taken by performing soft majority voting, that is considering the score obtained by averaging the frame predictions, the detected class being determined accordingly. The detection pipeline is illustrated in Fig. 4.

5 Experimental analysis

In this section, we present the performed experimental campaign. We start providing details on the experimental setup required to train the proposed detectors, then discuss the comparison with the state-of-the-art and report the evaluation results.

5.1 Experimental setup

We train and test the proposed detectors exploiting the dataset split policy reported in Table 3, randomly splitting the available videos into different sets. All the sub-datasets are used both in the training and evaluation phases except for *Splice_dance*, which is employed only at the test stage. In fact, as the background of fake videos corresponds to the original one, the CNN-based detector would reasonably fail if trained on this dataset. On the contrary, the feature-based detector, working only on the subject body parts, would likely report similar results to training on the *Fit_dance* sub-dataset.

Since *Swap_dance* and *Puppet_dance* have few examples and we do not have enough data for the validation process, we perform validation only when we train on *Fit_dance*, which is the largest sub-dataset. We are aware of the risk of overfitting in case no validation set is present in the training phase. However, in Sect. 5.3, we show that the detectors trained on *Swap_dance* and *Puppet_dance* demonstrate no signs of overfitting to the training data and a high test accuracy is obtained (aligned with the training one).

Features-based detector (LDP-TOP extraction): As suggested in [30], we extracted the LDP-TOP feature considering video sequences of 50 consecutive video frames, i.e., $T = 50$. In light of this, we temporally cropped each video of the FakeDance dataset such that its temporal dimension counts 50 frames. To augment the dataset for the experimental evaluations, we extracted all the possible sequences of 50 frames from each video, considering an overlap of 25 frames between adjacent sequences. We defined each final video sequence as $\mathbf{V}$.

Features-based detector (MLP classification): We propose a straightforward MLP architecture, considering an input layer which is fed with the LDP-TOP feature and returns an output vector of 100 elements. Then, three hidden layers follow, composed, respectively, by 100, 50 and 25 neurons. The output layer is composed of 2 neurons, related to the probability vector $\mathbf{p}_b^{(\rho)}$ defined in Sect. 4.1. The activation function selected for each neuron is the Rectified Linear Unit (ReLU).

As the number of selected body parts $|\mathcal{B}| = 5$, we trained 5 different classifiers for each of the 2 pre-processed videos, for a total of 10 MLP classifiers. We used cross-entropy loss and Adam optimizer with a constant learning rate of 10^{-3}. We trained the MLP for at most 300 epochs, stopping training when the investigated loss (the training loss in case validation is absent, the validation loss otherwise) does not decrease for 10 consecutive epochs. All the experiments have been performed on a workstation equipped with one Intel Core i7–6700K 8-Core CPU (4 GHz).

CNN-based detector: We extracted frames from each video with a frame rate of 2 frames per second. All the frames are resized to a common size of 300×300 pixels prior feeding them to the network. To be completely aligned with the feature-based detector setup and to fairly compare the two detectors, we always consider sequences of 50 consecutive frames extracted as we described for the LDP-TOP setup.

The CNN is trained using cross-entropy loss and Adam optimizer with a constant learning rate of $2 \cdot 10^{-4}$.

We stop training when the investigated loss does not decrease for 5 consecutive epochs. All the experiments have been performed using Pytorch [35] as a Deep Learning framework on a workstation equipped with one AMD Ryzen 9 3900X 12-Core CPU and one NVIDIA TITAN RTX 24GB.

5.2 Comparisons with state-of-the-art

To contextualize our proposed detection methodologies in the wide literature of multimedia forensics, we conduct a comparison with some state-of-the-art deepfake detection methods.

Given the nature of our fake sequences which principally consist of fully generated video frames (i.e., all sub-datasets except *Splice_dance* are fully synthetic), we compare our detectors with state-of-the-art works that deal with the detection of fully generated images [3, 36–38]. We do not retrain these techniques on our dataset, but we exploit the trained models released by the authors. Notice that all these methods are designed for synthetic image detection (i.e., they are developed for natural photographs) rather than for video frame analysis. To assess their performance on our dataset, we extract one video frame per second and we calculate the outcome for the entire video sequence by averaging the frames' results.

We obtain an average ACC of 50% for all the compared techniques, demonstrating that none of the methods can work on our dataset and confirming the need for dedicated techniques for our task. These results highlight a well-known challenge encountered with deepfake detectors, that is, the generalization to different semantic content and generation models. Another known issue is the poor generalization to deepfake video frames for methods trained on synthetic images only.

Notice that we could in principle compare our proposed methodology also with state-of-the-art detectors developed for deepfake videos. However, the common forgery detection methods designed to work on "standard" deepfake facial videos (i.e., including human subjects whose facial characteristics have been manipulated) cannot be directly applied to our case. The main reason for such a difficulty is that deepfake detectors strongly rely on the facial features of the videos, all the more that they usually crop the face region before inputting the video frames to the detectors. Furthermore, some detectors are based on the analysis of a set of geometric and semantic features that are directly related to human faces; like eye color [15] or facial pose [16]. No such traces can reasonably be exposed on FakeDance videos.

5.3 Results

We conduct several experiments to assess the performance of the proposed forgery detectors on the FakeDance dataset. In particular, we consider matched and mismatched dataset conditions, i.e., when the detectors are trained and tested on the same and different sub-datasets of the FakeDance dataset. We also consider compressed versions of the videos in addition to their RAW versions. Specifically, we compress the videos with two quality levels, using H.264 codec with a constant rate parameter equal to 23 (HQ) and 40 (LQ), respectively. We evaluate our method

Table 4 Performance of the detectors on FakeDance dataset in cross-dataset scenarios, when we train and test on RAW videos

	LDP-TOP						**CNN**					
	Fit		***Puppet***		***Swap***		***Fit***		***Puppet***		***Swap***	
	ACC	**AUC**	**ACC**	**AUC**	**ACC**	**AUC**	**ACC**	**AUC**	**ACC**	**AUC**	**ACC**	**AUC**
Fit	1.00	1.00	0.87	0.97	0.97	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Puppet	0.93	0.98	0.87	1.00	0.97	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Swap	0.89	0.96	0.72	0.98	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00

The first column represents the training set, while the other columns represent the test sets

Table 5 Performance of the detectors on FakeDance dataset in cross-dataset scenarios, when we train and test on HQ videos

	LDP-TOP						**CNN**					
	Fit		***Puppet***		***Swap***		***Fit***		***Puppet***		***Swap***	
	ACC	**AUC**	**ACC**	**AUC**	**ACC**	**AUC**	**ACC**	**AUC**	**ACC**	**AUC**	**ACC**	**AUC**
Fit	0.94	0.99	0.78	0.93	0.80	0.92	1.00	1.00	1.00	1.00	1.00	1.00
Puppet	0.64	0.69	1.00	1.00	0.82	0.91	0.73	0.82	1.00	1.00	0.87	0.96
Swap	0.62	0.67	0.52	0.84	0.87	0.99	0.76	0.85	0.90	0.96	1.00	1.00

The first column represents the training set, while the other columns represent the test sets

Table 6 Performance of the detectors on FakeDance dataset in cross-dataset scenarios, when we train and test on LQ videos

	LDP-TOP						**CNN**					
	Fit		***Puppet***		***Swap***		***Fit***		***Puppet***		***Swap***	
	ACC	**AUC**	**ACC**	**AUC**	**ACC**	**AUC**	**ACC**	**AUC**	**ACC**	**AUC**	**ACC**	**AUC**
Fit	0.87	0.98	0.52	0.74	0.60	0.74	0.80	0.94	0.60	0.76	0.50	0.56
Puppet	0.55	0.59	1.00	1.00	0.71	0.87	0.52	0.53	0.90	1.00	0.81	0.89
Swap	0.52	0.52	0.51	0.74	0.86	0.94	0.50	0.51	0.80	0.88	0.75	1.00

The first column represents the training set, while the other columns represent the test sets

performance by means of two metrics: the area under the curve (AUC) of the receiver operating characteristic (ROC), which is computed from the continuous scores returned by the two detectors, and the balanced accuracy (ACC) of the binary classification problem.

Due to space limitations, we were not able to include all the results in this paper. However, all additional results are included in supplementary material accessible through the following link: https://bit.ly/3J2IENp

Performance under matched compression: Tables 4, 5, 6 show the ACC and AUC of the proposed detectors trained and tested on the first three main subsets of FakeDance, namely, *Fit_dance*, *Puppet_dance* and *Swap_dance*. Table 4 reports the results for the RAW videos, Table 5 for the HQ videos and Table 6 for the LQ videos, when training and testing is done in matched compression conditions, that is, for a same compression quality of the videos. For each table, the results in the case of matched and mismatched datasets are reported.

When training and testing on RAW videos (see Table 4), the performance of both detectors are extremely good for all sub-datasets, with the CNN-based approach getting the best accuracies in all the cross-dataset scenarios. Not surprisingly, performance decrease if we analyze compressed videos. Nonetheless, if we train and test on the same dataset, performance remain accurate even when the compression is strong (LQ). In this scenario (see Table 6), we always achieve AUCs greater than 0.94.

In cross-dataset scenarios, the *Fit_dance* sub-dataset is the one showing the best generalization to the other sub-datasets. For instance, when working with HQ videos (see Table 5), perfect classification can be achieved with the CNN-based method and a small performance drop is observed for the LDP-TOP-based detector (AUC always above 0.92). This behavior confirms the expectation that training the detector on the most difficult cases, i.e., on the high-quality videos (i.e., *Fit_dance* videos) for which the discrimination is reasonably harder, permits to generalize to the other categories of fake videos.

In general, the CNN-based detector outperforms the LDP-TOP-based one whenever the video quality remains high, i.e., for RAW and HQ videos. However, when working on LQ videos, the LDP-TOP-based detector achieves better or similar performance to the CNN, showing very good robustness to data compression.

Moreover, focusing on Table 6, we observe that there are cases in which high AUCs do not correspond to high ACCs, and this is especially true for the CNN detector. For instance, when training and testing on *Puppet_dance* with the CNN-based method, we achieve perfect AUC but the ACC is 50%. This result indicates that the real and fake score distributions are perfectly separated, but the separating threshold is different from 0. As argued in many works, e.g., [39], when testing conditions are very different with respect to those considered for training (and validation), using the same fixed threshold typically does not work and may cause a wrong decision, with the consequent drop in ACC. To realign the ACC to the AUC metric, threshold calibration needs to be performed on data that are representative for the testing conditions considered.

Performance under mismatched compression: In this scenario, we train and test on videos that underwent compression with different compression levels. Motivated by

Table 7 Detection performance in cross-dataset scenarios and mismatched compression conditions

		Train on RAW				**Train on HQ**				**Train on LQ**			
Test	**Method**	**Test HQ**		**Test LQ**		**Test RAW**		**Test LQ**		**Test RAW**		**Test HQ**	
		ACC	**AUC**	**ACC**	**AUC**	**ACC**	**AUC**	**ACC**	**AUC**	**ACC**	**AUC**	**ACC**	**AUC**
Fit	LDP-TOP	0.61	0.85	0.51	0.63	0.91	0.99	0.58	0.71	0.50	0.91	0.50	0.78
	CNN	0.88	0.99	0.53	0.73	1.00	1.00	0.65	0.71	0.61	0.88	0.50	0.77
Puppet	LDP-TOP	0.52	0.65	0.50	0.50	0.90	0.99	0.50	0.53	0.50	0.94	0.50	0.93
	CNN	0.70	1.00	0.50	0.68	1.00	1.00	0.60	0.80	0.60	0.96	0.60	0.68
Swap	LDP-TOP	0.52	0.69	0.50	0.40	0.96	0.99	0.50	0.40	0.50	0.91	0.50	0.73
	CNN	0.70	0.93	0.50	0.78	1.00	1.00	0.50	0.72	0.55	0.89	0.55	0.89

Detectors are trained on *Fit_dance*

Table 8 Detection performance in cross-dataset scenarios and mismatched compression conditions

		Train on RAW				Train on HQ				Train on LQ			
Test	**Method**	**Test HQ**		**Test LQ**		**Test RAW**		**Test LQ**		**Test RAW**		**Test HQ**	
		ACC	**AUC**	**ACC**	**AUC**	**ACC**	**AUC**	**ACC**	**AUC**	**ACC**	**AUC**	**ACC**	**AUC**
Fit	LDP-TOP	0.76	0.85	0.51	0.63	0.94	0.99	0.60	0.71	0.79	0.91	0.70	0.78
	CNN	0.92	0.99	0.65	0.73	0.92	1.00	0.65	0.71	0.76	0.88	0.73	0.77
Puppet	LDP-TOP	0.59	0.65	0.52	0.50	0.96	0.99	0.50	0.53	0.61	0.94	0.75	0.93
	CNN	0.90	1.00	0.50	0.68	0.90	1.00	0.70	0.80	0.70	0.96	0.70	0.68
Swap	LDP-TOP	0.63	0.69	0.50	0.40	0.93	0.99	0.43	0.40	0.78	0.91	0.67	0.73
	CNN	0.85	0.93	0.70	0.78	0.80	1.00	0.60	0.72	0.80	0.89	0.80	0.89

Detectors are trained on *Fit_dance* and we set the threshold on *Fit_dance* validation set considering a compression quality matched with testing

Table 9 Relative accuracy change (Δ-ACC) between the results shown in Table 8 and those shown in Table 7

		Train on RAW		Train on HQ		Train on LQ	
Test	**Method**	**Test HQ**	**Test LQ**	**Test RAW**	**Test LQ**	**Test RAW**	**Test HQ**
		Δ-ACC	**Δ-ACC**	**Δ-ACC**	**Δ-ACC**	**Δ-ACC**	**Δ-ACC**
Fit	LDP-TOP	+24%	–	+3.2%	+3.4%	+58%	+40%
	CNN	+4.5%	+22%	−8.0%	–	+24%	+46%
Puppet	LDP-TOP	+13%	+4.0%	+6.6%	–	+22%	+50%
	CNN	+28%	–	−10%	+16%	+16%	+16%
Swap	LDP-TOP	+21%	–	−3.1%	−14%	+56%	+34%
	CNN	+21%	+40%	−20%	+20%	+45%	+31%

"–" refers to no changes

the previous considerations on the generalization capability of *Fit_dance*, we considered the detectors trained on this sub-dataset. Table 7 depicts the results under mismatched compression (the results in the matched compression case are not reported since they are the same as before).

We notice that training on videos compressed with a certain quality level helps achieving a good level of robustness against similar or higher quality videos, that is, for a similar or weaker compression level. For instance, by training on HQ videos, we achieve excellent performance on RAW videos but poor on LQ ones; by training on LQ videos, we achieve acceptable performance on RAW videos and only slightly worse ones on HQ videos. The ACC results are not very good. As observed before, this is due to the mismatch between training and testing conditions.

To address the threshold calibration issue, we re-compute the threshold in such a way to maximize the accuracy on the *Fit_dance* validation set, for a same quality of the compression. For example, when testing HQ images, we consider the threshold set on the *Fit_dance* HQ validation set, that is, under matched compression conditions. From Table 8, we can see that the ACC increases in most of the cases where AUC and ACC were not aligned before.

Table 10 Performance of the detectors on *Splice_dance*, when training is performed on *Fit_dance* videos, RAW quality

Method	Test RAW		Test HQ		Test LQ	
	ACC	AUC	ACC	AUC	ACC	AUC
LDP-TOP	1.00	1.00	0.60	0.90	0.50	0.58
CNN	0.70	1.00	0.60	0.72	0.55	0.65

To highlight the change, Table 9 reports the relative accuracy change (defined as *Delta*-ACC) between the results in Table 8 and Table 7. When we train on RAW or HQ videos, a mismatch could remain between testing data and the data used for setting the threshold, i.e., when testing *Puppet_dance* and *Swap_dance*. Not surprisingly, then, in these scenarios, the ACC could improve further by calibrating the threshold on data taken from the same dataset. On the contrary, when training on LQ videos, the proposed threshold calibration step always improves the results, with improvements above +15%.

Performance with locally manipulated videos: Results achieved on *Splice_dance* for various compression qualities are shown in Table 10, when training is performed on *Fit_dance* RAW videos. In this scenario, the CNN-based strategy achieves worse results with respect to the test performed on the *Fit_dance* videos (see Table 7, second row, train on RAW). We conjecture the performance deterioration of the CNN-based method is due to the fact that, since this detector works full frame, it relies on artifacts introduced by the motion transfer generative model in both the foreground and the background. However, the background artifacts are not present in *Splice_dance*, due to the real background replacement. On the contrary, the performances of LDP-TOP are similar to those achieved on *Fit_dance* (see Table 7, first row, train on RAW case). In fact, by looking only at the body parts and their movements, the LDP-TOP-based detector is by construction more robust against background manipulations and splicing.

Performance in data-scarcity conditions: We investigate the performance of the detectors in data scarcity conditions, that is, when only very few examples are available at training time. These tests are particularly relevant since, in addition of relieving from the burden of collecting large amount of data, in some scenarios it might be the case that a large number of samples representative for the task are not available. Results are reported in Table 11, for different numbers of training videos selected from *Fit_dance*, RAW quality.

As in the splicing case, in this scenario as well the LDP-TOP-based detector can achieve superior performance with respect to the CNN-based method, that, being completely data driven, requires a larger number of training videos to get good discrimination capabilities. In particular, for the LDP-TOP detector, 4 videos are already enough in most of the cases to get a balanced accuracy larger than 88%, while the CNN-based method is not able at all to discriminate with so few videos. The capability to work in data-scarcity conditions is a noticeable strength of the LDP-TOP-based approach.

Table 11 Performance of the detectors in data scarcity conditions

	Swap_dance				*Puppet_dance*				*Fit_dance*				*Splice_dance*			
No. training	**LDP-TOP**		**CNN**		**LDP-TOP**		**CNN**		**LDP-TOP**		**CNN**		**LDP-TOP**		**CNN**	
videos	**ACC**	**AUC**	**ACC**	**AUC**	**ACC**	**AUC**	**ACC**	**AUC**	**ACC**	**AUC**	**ACC**	**AUC**	**ACC**	**AUC**	**ACC**	**AUC**
2	0.63	0.96	0.55	0.65	0.63	0.90	0.40	0.60	0.94	0.99	0.73	0.69	0.88	0.94	0.55	0.58
4	0.89	1.00	0.55	0.66	0.71	0.96	0.70	0.80	0.98	1.00	0.61	0.73	0.95	1.00	0.55	0.57
8	0.85	0.98	0.80	0.91	0.94	0.98	0.80	0.96	0.98	1.00	0.96	0.99	1.00	1.00	0.50	0.68
16	0.90	0.99	0.90	1.00	0.87	0.99	1.00	1.00	0.99	1.00	1.00	1.00	1.00	1.00	0.60	0.79
32	0.96	1.00	1.00	1.00	0.89	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.76	0.80
54	0.97	1.00	1.00	1.00	0.87	0.97	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.70	1.00

We train on RAW *Fit_dance* videos and we test on RAW videos of all four sub-datasets

Table 12 Performance of the detectors after fusion on RAW videos

	LDP-TOP			CNN			Fusion		
	Fit	*Puppet*	*Swap*	*Fit*	*Puppet*	*Swap*	*Fit*	*Puppet*	*Swap*
	Δ-ACC	**Δ-ACC**	**Δ-ACC**	**Δ-ACC**	**Δ-ACC**	**Δ-ACC**	**ACC**	**ACC**	**ACC**
Fit	–	+14.94%	+3.09%	–	–	–	1.00	1.00	1.00
Puppet	+7.52%	+14.94%	+3.09%	–	–	–	1.00	1.00	1.00
Swap	+12.36%	+38.89%	+1.01%	–	–	–	1.00	1.00	1.00

LDP-TOP and CNN refer to the relative accuracy change for both detectors after fusion, defined as Δ-ACC. The symbol "–" indicates no changes. The first column represents the training set, while the other columns represent the test sets

Table 13 Performance of the detectors after fusion on HQ videos

	LDP-TOP			CNN			Fusion		
	Fit	*Puppet*	*Swap*	*Fit*	*Puppet*	*Swap*	*Fit*	*Puppet*	*Swap*
	Δ-ACC	**Δ-ACC**	**Δ-ACC**	**Δ-ACC**	**Δ-ACC**	**Δ-ACC**	**ACC**	**ACC**	**ACC**
Fit	+6.38%	+28.20%	+25.00%	–	–	–	1.00	1.00	1.00
Puppet	+18.75%	–	−8.53%	+4.10%	–	−13.79%	0.76	1.00	0.75
Swap	+29.03%	+92.30%	+14.94%	+5.26%	+11.11%	–	0.80	1.00	1.00

LDP-TOP and CNN refer to the relative accuracy change for both detectors after fusion, defined as Δ-ACC. The symbol "–" indicates no changes. The first column represents the training set, while the other columns represent the test sets

Table 14 Performance of the detectors after fusion on LQ videos

	LDP-TOP			CNN			Fusion		
	Fit	*Puppet*	*Swap*	*Fit*	*Puppet*	*Swap*	*Fit*	*Puppet*	*Swap*
	Δ-ACC	**Δ-ACC**	**Δ-ACC**	**Δ-ACC**	**Δ-ACC**	**Δ-ACC**	**ACC**	**ACC**	**ACC**
Fit	+10.34%	−3.84%	−16.66%	+20.00%	−16.66%	–	0.96	0.50	0.50
Puppet	−3.63%	–	+40.84%	+1.92%	+11.11%	+23.45%	0.53	1.00	1.00
Swap	−3.84%	+96.07%	+16.27%	–	+25.00%	+33.33%	0.50	1.00	1.00

LDP-TOP and CNN refer to the relative accuracy change for both detectors after fusion, defined as Δ-ACC. The symbol "–" indicates no changes. The first column represents the training set, while the other columns represent the test sets

5.4 Fusion of LDP-TOP and CNN detectors

As a final investigation, we conduct an experiment involving the fusion of two detectors. We employ a straightforward fusion, considering the average of the output probabilities generated by both detectors.

The fusion results are presented in Tables 12, 13, and 14 for RAW videos, HQ videos and LQ videos, respectively, and compared with those achieved by the single techniques shown in Tables 4, 5, 6, i.e., in the matched compression scenario. In the columns labeled as "LDP-TOP" and "CNN", we highlight the relative accuracy change (defined as Δ-ACC) of the fusion process with respect to the single detector results shown in Tables 4, 5, and 6.

On average, the achieved results demonstrate an improvement in performance with respect to the single detectors, which in some cases is very relevant. This consideration is valid primarily for the LDP-TOP technique which, if considered as a single approach, indeed returned worse results than the CNN, especially on RAW and HQ videos (see Tables 4, 5). The CNN approach mainly benefits from the fusion strategy on LQ video sequences (see Table 14). As a matter of fact, the LDP-TOP method in this case achieved comparable results to the CNN approach (see Table 6), therefore fusion between the two methods reasonably helps to improve the CNN performances as well.

6 Conclusions

In this paper, we addressed the problem of detection of human motion transfer videos. We considered two detection approaches: a feature-based approach that relies on hand-crafted features, namely the LDP-TOP, extracted from the body parts of the human subject, and a completely data-driven method, based on a CNN trained directly on video frames.

We also contributed with the construction and the release of a dataset, the FakeDance dataset, with almost 600 synthetic videos, obtained by following different generation pipelines and different compression settings.

Several experiments have been carried out to test the effectiveness of the proposed detectors and generalization performance. In particular, the performance of the detectors is assessed and discussed in the cross-dataset scenario, that is, when different subsets of the FakeDance dataset are considered for training and testing, for both matched and mismatched compression levels of the videos.

When working with high visual quality videos, the CNN-based outperforms the LDP-TOP-based method in all the conditions. However, the LDP-TOP-based method is more robust to strong data compression and achieves better results on low visual quality videos. Furthermore, while the CNN-based method achieves superior performance when many video samples are available for training the detector, the LDP-TOP-based method gets better performance in data-scarcity conditions, when very few videos are available to the analyst. Finally, being focused on the analysis of body parts and their movements, the LDP-TOP-based detector achieves better performance in the challenging splicing scenario, in which a synthesized human subject is copied and pasted over a real background.

Future research will be devoted to the investigation of an advanced fusion strategy of the two detectors, in the attempt to get a detector that works well in all the operating conditions. Then, it would be also interesting to investigate the robustness of the proposed detectors against adversarial attacks, aimed at detection failure, under different degrees of knowledge about the defence approach.

Abbreviations

CNN	Convolutional Neural Network
LDP	Local Derivative Pattern
MLP	Multi-layer perceptron
GAN	Generative Adversarial Network
LDP-TOP	Local Derivative Pattern on Three Orthogonal
ReLU	Rectified Linear Unit
LSTM	Long short-term memory
AUC	Area under the curve
ACC	Accuracy
DL	Deep Learning

Acknowledgements
We gratefully acknowledge the financial support from the Italian Ministry of University and Research, the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL).

Author contributions
All authors have contributed toward this work as well as in compilation of this manuscript. All authors read and approved the final manuscript.

Funding
This work has been supported by the PREMIER project under contract PRIN 2017 2017Z595XS-001, funded by the Italian Ministry of University and Research and by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL) under agreement number FA8750-20-2-1004. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA and AFRL or the U.S. Government.

Availability of data and materials
Our data are available in a public repository. The address is informed in the paper (https://bit.ly/3J2lENp). Moreover, additional results, not included in the paper due to space limitations, can also be found at the same link.

Declarations

Competing interests
The authors declare that they have no conflict of interest.

Received: 15 July 2022 Accepted: 31 July 2024
Published online: 14 August 2024

References
1. L. Verdoliva, Media forensics and deepfakes: an overview. IEEE J. Select. Topics Signal Process. **14**(5), 910–932 (2020)
2. T.T. Nguyen, C.M. Nguyen, D.T. Nguyen, D.T. Nguyen, S. Nahavandi, Deep learning for deepfakes creation and detection: A survey. arXiv preprint arXiv:1909.11573 (2019)
3. O. Alamayreh, M. Barni, Detection of gan-synthesized street videos. In: 2021 29th European Signal Processing Conference (EUSIPCO), pp. 811–815 (2021). IEEE
4. C. Chan, S. Ginosar, T. Zhou, A.A. Efros, Everybody dance now. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 5933–5942 (2019)
5. H. Farid, Seeing is not believing. IEEE Spectr. **46**(8), 44–51 (2009)
6. FaceSwap https://github.com/deepfakes/faceswap
7. I. Perov, D. Gao, N. Chervoniy, K. Liu, S. Marangonda, C. Umé, M. Dpfks, C.S. Facenheim, RP, L., J. Jiang, S. Zhang, P. Wu, B. Zhou, W. Zhang, Deepfacelab: A simple, flexible and extensible face swapping framework. CoRR (2020). arXiv:2005.05535
8. DFaker https://github.com/dfaker/df
9. GitHub: Faceswap-GAN. GitHub
10. VGGFace https://github.com/rcmalli/kerasvggface
11. J. Thies, M. Zollhöfer, M. Stamminger, C. Theobalt, M. Nießner, Face2face: Real-time face capture and reenactment of RGB videos. CoRR (2020). arXiv:2007.14808
12. J. Thies, M. Elgharib, A. Tewari, C. Theobalt, M. Nießner, Neural voice puppetry: Audio-driven facial reenactment. CoRR (2019). arXiv:1912.05566
13. T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, G. Liu, A. Tao, J. Kautz, B. Catanzaro, Video-to-video synthesis. (2018) arXiv preprint arXiv:1808.06601
14. I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, (2014) Generative adversarial networks. arXiv preprint arXiv:1406.2661
15. F. Matern, C. Riess, M. Stamminger, Exploiting visual artifacts to expose deepfakes and face manipulations. In: 2019 IEEE Winter Applications of Computer Vision Workshops (WACVW), pp. 83–92 (2019). IEEE
16. X. Yang, Y. Li, 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 (2019). IEEE
17. Y. Li, M.-C. Chang, 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 (2018). IEEE
18. D. Afchar, V. Nozick, J. Yamagishi, I. Echizen, Mesonet: a compact facial video forgery detection network. In: 2018 IEEE International Workshop on Information Forensics and Security (WIFS), pp. 1–7 (2018). IEEE
19. H.H. Nguyen, J. Yamagishi, I. Echizen, Use of a capsule network to detect fake images and videos. (2019) arXiv:1910.12467
20. Rossler, A., Cozzolino, D., Verdoliva, L., Riess, C., Thies, J., Nießner, M.: Faceforensics++: Learning to detect manipulated facial images. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 1–11 (2019)
21. N. Bonettini, E.D. Cannas, S. Mandelli, L. Bondi, P. Bestagini, S. Tubaro, Video face manipulation detection through ensemble of CNNs. In: International Conference on Pattern Recognition (ICPR) (2021)
22. S. Seferbekov, DeepFake Detection (DFDC) Solution. https://github.com/selimsef/dfdc_deepfake_challenge (2021)

23. D. Güera, E.J. Delp, Deepfake video detection using recurrent neural networks. In: 2018 15th IEEE International Conference on Advanced Video and Dignal Based Surveillance (AVSS), pp. 1–6 (2018). IEEE
24. I. Amerini, R. Caldelli, Exploiting prediction error inconsistencies through LSTM-based classifiers to detect deepfake videos. In: Proceedings of the 2020 ACM Workshop on Information Hiding and Multimedia Security, pp. 97–102 (2020)
25. T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, 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)
26. Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, Y. Sheikh, Openpose: realtime multi-person 2d pose estimation using part affinity fields. IEEE Trans. Pattern Anal. Mach. Intell. **43**(1), 172–186 (2019)
27. S. Tomar, Converting video formats with ffmpeg. Linux J. **2006**(146), 10 (2006)
28. W. Zhou, C. B. Alan, L. Ligang, Why is image quality assessment so difficult? In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (2002). https://doi.org/10.1109/ICASSP.2014.6854082
29. Y. Li, X. Yang, P. Sun, H. Qi, S. Lyu, Celeb-df: A large-scale challenging dataset for deepfake forensics. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 3207–3216 (2020)
30. Q.-T. Phan, D.-T. Dang-Nguyen, G. Boato, F.G. De Natale, Face spoofing detection using ldp-top. In: 2016 IEEE International Conference on Image Processing (ICIP), pp. 404–408 (2016). IEEE
31. E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, 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)
32. T. Jabid, M.H. Kabir, O. Chae, Local directional pattern (ldp) for face recognition. In: 2010 Digest of Technical Papers International Conference on Consumer Electronics (ICCE), pp. 329–330 (2010). IEEE
33. M. Tan, Q. Le, Efficientnet: Rethinking model scaling for convolutional neural networks. In: International Conference on Machine Learning, pp. 6105–6114 (2019). PMLR
34. J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255 (2009). IEEE
35. A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems (2019)
36. S.-Y. Wang, O. Wang, R. Zhang, A. Owens, 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)
37. S. Mandelli, N. Bonettini, P. Bestagini, S. Tubaro, Detecting gan-generated images by orthogonal training of multiple cnns. In: 2022 IEEE International Conference on Image Processing (ICIP), pp. 3091–3095 (2022). IEEE
38. B. Liu, F. Yang, X. Bi, B. Xiao, W. Li, X. Gao, Detecting generated images by real images. In: European Conference on Computer Vision, pp. 95–110 (2022). Springer
39. S. Mandelli, D. Cozzolino, E.D. Cannas, J.P. Cardenuto, D. Moreira, P. Bestagini, W.J. Scheirer, A. Rocha, L. Verdoliva, S. Tubaro, E.J. Delp, Forensic analysis of synthetically generated western blot images. IEEE Access, 1–1 (2022). https://doi.org/10.1109/ACCESS.2022.3179116

Publisher's Note