Head Pose Estimation Patterns as Deepfake Detectors

FEDERICO BECATTINI*, Università degli Studi di Siena, Italy
CARMEN BISOGNI, VINCENZO LOIA, and CHIARA PERO, Università degli Studi di Salerno, Italy
FEI HAO, School of Computer Science, Shaanxi Normal University, China

The capacity to create "fake" videos has recently raised concerns about the reliability of multimedia content. Identifying between true and false information is a critical step toward resolving this problem. On this issue, several algorithms utilizing deep learning and facial landmarks have yielded intriguing results. Facial landmarks are traits that are solely tied to the subject's head posture. Based on this observation, we study how Head Pose Estimation (HPE) patterns may be utilized to detect deepfakes in this work. The HPE patterns studied are based on FSA-Net, SynergyNet, and WSM, which are among the most performant approaches on the state of the art. Finally, using a machine learning technique based on K-Nearest Neighbor and Dynamic Time Warping, their temporal patterns are categorized as authentic or false. We also offer a set of experiments for examining the feasibility of using deep learning techniques on such patterns. The findings reveal that the ability to recognize a deepfake video utilizing an HPE pattern is dependent on the HPE methodology. On the contrary, performance is less dependent on the performance of the utilized HPE technique. Experiments are carried out on the FaceForensics++ dataset, that presents both identity swap and expression swap examples. The findings show that FSA-Net is an effective feature extraction method for determining whether a pattern belongs to a deepfake or not. The approach is also robust in comparison to deepfake videos created using various methods or for different goals. In mean the method obtain 86% of accuracy on the identity swap task and 86.5% of accuracy on the expression swap. These findings offer up various possibilities and future directions for solving the deepfake detection problem using specialized HPE approaches, which are also known to be fast and reliable.

CCS Concepts: • **Computing methodologies** → *Feature selection*; • **Applied computing** → **Investigation techniques**.

Additional Key Words and Phrases: DeepFake, Face Recognition, Head Pose Estimation, Machine Learning, Deep Learning

1 INTRODUCTION

Deepfake is a system that uses artificial intelligence to generate modified videos or images that appear real. Deepfake videos can be created using this technology to scam, mislead, or propagate disinformation. The inclusion of deepfake videos in forensics can pose a variety of issues. A deepfake video, for example, might be used as evidence in a judicial trial to indict an innocent person or absolve a criminal individual. A famous episode of the inital spreading of deepfake was in april 2018, when Jordan Peele e Jonah Peretti creted a deepfake by using Barack Obama as a spot on the dangers of deepfakes (Fig. 1). Deepfake video might potentially be used to deceive witnesses and influence jury decisions. Furthermore, the presence of deepfake videos may jeopardize the integrity of digital forensic evidence. For example, if a deepfake video is used to change digital evidence, investigators may

*Both authors contributed equally to this research.

Authors' addresses: Federico Becattini, federico.becattini@unisi.it, Università degli Studi di Siena, via Banchi di Sotto 55, Siena, Italy, 53100; Carmen Bisogni, cbisogni@unisa.it; Vincenzo Loia, loia@unisa.it; Chiara Pero, cpero@unisa.it, Università degli Studi di Salerno, Via Giovanni Paolo II, 132, Fisciano, Salerno, 84084, Italy; Fei Hao, feehao@gmail.com, School of Computer Science, Shaanxi Normal University, Xi'An, Shaanxi, 710062, China.

1551-6857/2023/11-ART $15.00
https://doi.org/10.1145/3612928

find it difficult to discern between what is genuine and what is not. Inevitably, deepfake videos may be exploited to violate people's privacy. A deepfake video, for example, may be constructed to share private information or to discredit an individual. This might irreparably harm the reputations and lives of those concerned. To summarize, the emergence of deepfake videos compromises the trustworthiness of digital forensic evidence as well as people's privacy. Forensic investigators and cybersecurity specialists must be aware of the risks involved with this technology and be prepared to recognize and refute deepfake videos as needed.

It is for this aim that deepfake detection methods were born. Deepfake detection methods use a combination of analytic and machine learning approaches to detect and distinguish deepfake videos from real ones. Deepfake detection employs a number of approaches, including facial and lip movement analysis, picture texture analysis, identification of compression artifacts, and evaluation of the video's temporal consistency. These strategies can be used in conjunction to increase the detection system's accuracy. Deepfake videos are frequently detected using machine learning methods such as convolutional neural networks [6]. These algorithms may be trained on a large number of deepfake and legitimate video examples to learn to recognize and identify the distinguishing elements of deepfake videos. Deepfake detection approaches, however, have certain drawbacks. Deepfake video developers, for example, may utilize more complex tactics to prevent detection, such as adversarial generation algorithms to disguise deepfake video trademarks. Also, image quality and the type of video studied might have an impact on detection. Therefore, it is important to always study new techniques based on different concepts and features. We propose research on the use of head posture estimation as characteristics for the detection of deepfakes in this work, driven by early concepts that have not yet been fully investigated [43].

Fig. 1. The deepfake of Obama created by Jordan Peele and Jonah Peretti. The image has been obtained by the BBC youtube channel.

Head Pose Estimation (HPE) is a technique for estimating a person's head position and orientation in an image or video. Several applications rely on head position estimation, including augmented reality, human-computer interface, and computer security. In head pose estimation, the angular notations pitch, yaw, and roll are often employed to indicate head orientation. Pitch is the angle of head tilt relative to the horizontal axis (forward and backward movement), yaw represents the angle of head rotation relative to the vertical axis (left and right movement), and roll represents the angle of head rotation relative to the longitudinal axis (lateral tilt movement). In recent times, the HPE algorithms have achieved very high performances in terms of angular errors. Some methods are based on both handcrafted features as statistical distribution [2], [7] or fractal encodings [9], [8] than neural networks [42], [38], [20] [37]. Those methods base their estimation technique on the aspect or key points of the face. The same features are used in several works operating in deepfake detection [32], [44].

As a result, the question arises: *can a video's representative sequence of head pose predictions give distinguishing information between a true pattern and a deepfake?* When a video is tampered with, an alignment on the face

needs to be performed. The discrepancy in this alignment can be more or less visible to a human. HPE methods are very accurate at detecting slight differences in rotations along the three axes of the head. In Figure 2 we can observe the variation in pitch, yaw, and roll during a video sequence for a real and a fake video generated by FaceSwap [1]. As we can notice, the discrepancy may assume different aspects. In pitch, it seems that the fake

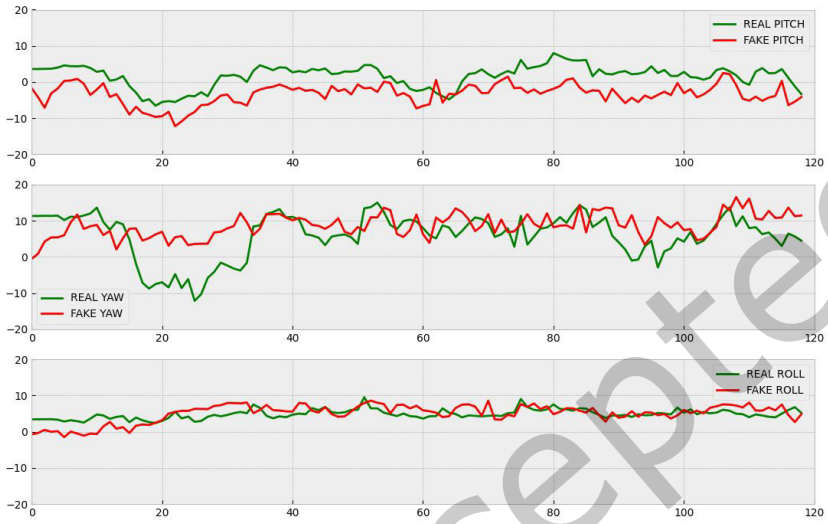


Fig. 2. Differences between the HPE path in pitch, yaw, and roll by using FSA-Net on a real video and a fake produced by FaceSwap.

video has mismatched the real one by a constant, since the real face roll rotation is always higher than the fake one. By observing the yaw rotation, we can notice that the fake face tends to rotate less than the real one. In particular, when the real one performs a significant rotation (between frames 15 and 40), the fake face is not able to reproduce this behavior. The roll rotation in this case, is the best hallucination created by the fake video. This could depend on which points were considered more relevant by the deepfake generator to align the images.

If we consider the mean and the standard deviation of the fake and real values on the axis, we obtain the results in Table 1

Axes	Real	Fake
Pitch	1.65±3.05	-3.34±2.79
Yaw	5.88±6.07	8.66±3.22
Roll	4.86±1.25	5.05±2.47

Table 1. The mean and the standard deviation along the three axes of the real and fake video from the sequence in Figure 2.

This suggests that the HPE pattern has relevance when seen from the point of view of the deepfake detector. In this paper, we conduct a horizontal and vertical analysis to attempt to answer the previous question, by analyzing three distinct HPE methods consistent with recent literature on the topic [3]. We can thus summarize the contribution of the paper as follows:

- An horizontal analysis is conducted on the popular FF++ deepfake dataset, examining the methods of HPE, FSA-Net, SynergyNet, and the Web-Shaped Model (WSM) and providing results in terms of accuracy and F1-score of the real and fake classes.
- The most performant classification method is identified by testing a machine learning approach based on KNN and Dynamic Time Warping (DTW) and a deep learning approach based on 1D convolutional networks and recurrent neural networks. The results on the KNN with DTW and FSA-Net shows an accuracy until 91%.
- A comparison with the state of the art and cross-dataset comparisons are provided to demonstrate how significantly HPE's best-performing approach differs from the performance of contemporary, purpose-built publications on the deepfake detection challenge.

The remainder of the paper is organized as follows: In Section 2 we present the methods at the state of the art that are focused on the deepfake detection problems. In Section 3 we discuss in detail the steps of the selected HPE techniques, together with the motivations why we prefer those techniques over other methods at the state of the art. In Section 4 we present the experimental protocol. In particular, here we discuss the dataset used, the details of the classification methods, and their characteristics. In Section 5 the results obtained are shown. Firstly, an ablation study has been conducted to find the optimal parameters and the optimal HPE method by taking a subset of the dataset into account. Then, the optimal method is compared with the state of the art and in a cross-validation context between identity swap and expression swap. Finally, in Section 6 we draw our conclusions and give consideration to the effectiveness of the HPE methods to detect deepfake videos. Here, we also suggest some future directions to improve the performance of HPE methods to solve this task.

2 RELATED WORKS

Various recent state-of-the-art studies may be found in the domain of deepfake detection. Deepfakes' implications are of great interest not just to the scientific community, but also to the media and ordinary people who have access to social and media channels. As a result, various authors have investigated methods to recognize these artifacts, frequently using the same techniques that created them, specifically deep learning. Because the presence of a temporal variable limits the research category to biometrics generated from videos, we will encounter various studies based on graphs and LSTMs. Therefore, we may divide algorithms into two main types: those that regard deepfakes as anomalies and those that treat deepfakes as a distinct pattern from real videos. Rossler et al. [33] proposed firstly the FaceForensics++ Dataset, composed by 4 deepfakes generated videos. Together with the dataset, they also presented some benchmarks on the latter, such as XceptionNet [11], MesoNet [4], etc. The technology utilized by Rossler et al. to create these videos, such as the Face2Face app or FaceSwap, are widely available to the average user. Furthermore, they produce artifacts of varied complexity, allowing one to analyze the degree to which a deepfake detection algorithm is capable of detecting artifacts. Deepfake detection has grown since this time, and academic research and artifacts that can be made by the general consumer have aligned. Demir et al. [14] developed a three-step method. They begin by identifying certain eye and gaze traits that are common in deep fakes and display distinct patterns in actual videos. Second, they combined these characteristics into signatures and compare them across actual and fake videos, taking into account geometric, visual, metric, temporal, and spectral variances. Finally, they created a deep neural network to extend this formulation and categorize any videos as false or real. Also, Xu et al. [41] proposed an approach based on different sets of features. They describe MCX-API, which combines pairwise learning and information from different color space representations in a fine-grained manner to detect deepfakes. To leverage information from various color spaces, a multi-channel network is used as the base. This is followed by enforcing pairwise learning, utilizing the architecture of AattentivePpairwiseLlearning Khalid et al. [24] proposed a new architecture based on graph neural networks for detecting hyper-realistic deepfakes. The method involves dividing the image into smaller

patches, or nodes, which are then connected to form a graph by linking the nearest neighbors. To facilitate the exchange of information between nodes, the model consists of two core modules: GraphNet, which utilizes graph convolution layers to gather and update graph information, and FFN, which employs linear layers to transform node features. This architecture is particularly popular for detecting deepfakes. Xie et al. [40] focused on the computational performances. They proposed a modified version of AlexNet. While the classical architecture has 5 convolutional layers, 3 of which are fully connected, and a softmax activation function, the author proposed using a lighter version with only 3 convolutional layers. Ilyas et al. [21] also use a previously existing architecture to propose a hybrid deep learning framework called InceptionResNet-BiLSTM. It comprises two components: a customized InceptionResNetV2 and a Bidirectional Long-Short Term Memory (BiLSTM). The framework operates by extracting faces from videos and feeding them to the customized InceptionResNetV2, which extracts learnable features at the frame level. The sequence of features obtained is then used to train a temporally aware BiLSTM, which classifies videos as either real or fake.

Then, we have methods that see the deepfake as an anomaly, and as a consequence, the problem is solved with anomaly detection techniques. Liu et al. [28] introduced a new detector called TCSD, which consists of three complementary streams based on several semantic discrepancies. Firstly, they developed a new depth estimator to capture depth information (DI). To further enhance the detection of forgeries, they also took into account the discrepancy between the foreground and background information (FBI) and the inconsistency between local and global information (LGI). These measures are aimed at providing more comprehensive forgery clues. An attention-based multi-scale feature extraction (MsFE) module captures more complementary features from the depth information, foreground information, and background information. And finally, two attention-based feature fusion modules dynamically combine the extracted information. Khalid et al. [25] proposed the OCFakeDect, an approach based on variational autoencoder block to detect Deepfakes by effectively learning the features of real images. The problem is thus structured as a one-class anomaly detection problem. To this aim, they defined an anomaly score by the loss reconstruction score. The thresholding is statistically based, obtained by calculating the inter-quartile range considering 80% of the distribution. Heo et al. [18] proposed a detection method for fake videos using a Vision Transformer model and a distillation methodology. They designed a CNN feature and patch-based positioning model that interacts with all positions to identify the artifact region, which helps solve the false negative problem. To detect discrepancies, Nirkin et al. [31] focused on the generic features of the face as a biometric trait. To accomplish this, they employ two networks: a face identification network that focuses on the facial region delimited by a precise semantic segmentation and a context recognition network that takes the context of the face into account. The recognition signals obtained from these two networks are then used to detect these discrepancies. Finally, Chugh et al. [12] proposed deepfake video identification based on the dissimilarity between audio and visual domains, known as the Modality Dissonance Score (MDS). They anticipated that manipulating one modality will result in disharmony between the two, such as loss of lip-sync, abnormal face and lip motions, and so on. The mean aggregate of dissimilarity scores between audio and visual segments in a video is used to calculate MDS. Discriminative characteristics are gained chunk-wise for the audio and visual channels, using the cross-entropy loss for individual modalities and a contrastive loss to describe inter-modality similarity.

3 HPE METHODS

One of the most prevalent strategies for estimating head posture is to employ a mix of face detection and landmark analysis tools. The system, in particular, can recognize the individual's face in the image and then evaluate markers such as the eyes, nose, and mouth to establish the position and orientation of the head. Other methods may employ machine learning algorithms to understand the correlations between facial landmarks and head posture, or they may employ information about the shape and structure of the human head to estimate its location

and orientation. Many HPE approaches have recently produced extremely good results for this problem. In controlled environments, those methods may reach a mean error of around 3 °[2], [17], [37]). The literature of the last five years has on this kind of data a mean error of 4.64 °. In the wild, the accuracies are very various depending on the database used to perform the training and on how much the model has been adapted to the latter by preprocessing. Only one algorithm obtained an error lower than 2°in mean [39]. The majority has a mean error over 3°[26], [20], [29]. The literature of the last five years has on this kind of data a mean error of 5.04 °.

The HPE methods used in our analysis are FSA-Net [42], SynergyNet [38] and Web-Shaped Model (WSM) [2]. Those methods have been chosen on the basis of the following criteria:

- **Availability of the code.** In order to reproduce the methods correctly, we preferred the papers that have a well commented and reproducible code.
- **Novelty and differentiation of the methods.** We preferred three methods that use completely different architectures. The methods are from the years 2019, 2020, and 2021; for this reason, they are in line with the SOTA.
- **Performances.** The preferred methods present high performances on the HPE task. FSA-Net has a mean 4°of error in controlled environments and 5.07°of error in the wild. SynergyNet has, on average, 2.55°of error in the wild. WSM has in mean 2.43°of error on controlled environments and 4.09°of error in the wild. All three methods have an error near or lower than the mean of the SOTA for both environments.

In Figure 3 it is possible to appreciate the angles estimated in pitch, yaw, and roll by the three methods on a frame of a deepfake video from the FaceForensics++ Dataset.

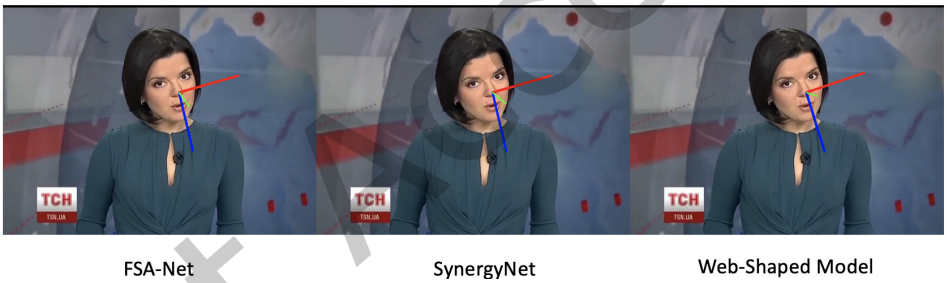


Fig. 3. The three degrees of freedom in head orientation - Pitch (in red), Yaw (in green), and Roll (in blue) - estimated from the methods used in the experimental phase.

3.1 FSA-Net

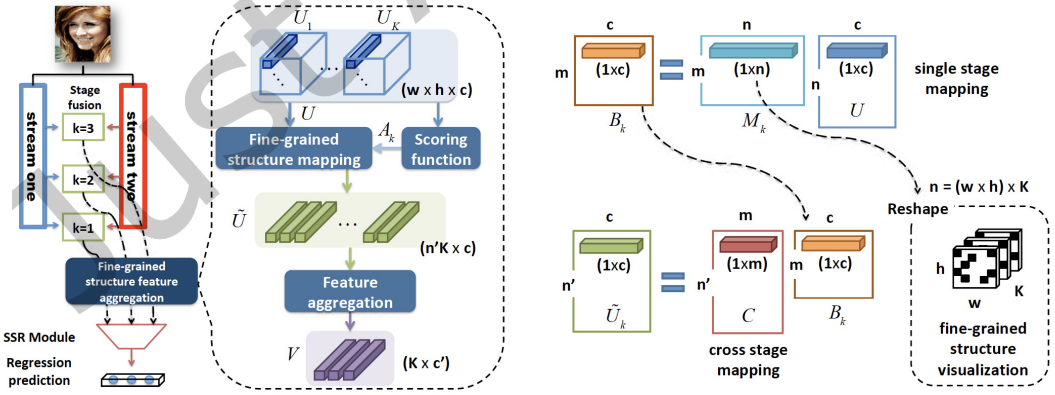
The authors in [42] presented a model called FSA-Net that was designed for the estimation of head pose from a single image without the use of facial landmarks. FSA-Net is built on the foundation of the *soft stagewise regression* (SSR) scheme, and to capture multi-scale information, it combines feature maps obtained from different layers. The fundamental concept underlying FSA-Net is the spatial grouping of pixel-level features within the feature map to generate a collection of characteristics encoded with spatial information. These features are then selected as candidates for aggregation. The crux of FSA-Net lies in its capability to learn a fine-grained structure mapping that groups pixel-level features into region-level features with greater potency. This mapping can be viewed as a highly adaptable and versatile tool for pooling. To extract a broader range of spatial information, the authors have incorporated both learnable and non-learnable importance measures. This approach facilitates the creation of complementary model variants that work together synergistically to form a powerful and robust ensemble.

A comprehensive description of the FSA-Net architecture is provided below, focusing on its main features and relevant aspects.

Given an input image, the architecture is based on a process that involves two streams, each of which is responsible for extracting feature maps at various stages. Specifically, there are K stages, and for the k-th stage, the feature maps extracted from both streams are fused together. This results in a combined feature map that is subsequently transformed into c channels. To reduce the size of the feature map, average pooling is applied, yielding a $w \times h \times c$ feature map U_k for the k-th stage. U_k is a grid that represents the spatial layout of the input image, with each individual cell containing a c-dimensional feature representation that corresponds to a specific location within the image. After that, the K feature maps are inserted into the mapping module, which generates K c'-d vectors. These vectors are utilized to compute the stage outputs for SSR function. With K feature maps of dimensions $w \times h \times c$, the fundamental purpose of the aggregation module is to compress a group of features into a limited set of more relevant characteristics (K c'-d features, one for each stage). However, several existing feature aggregation methods entirely neglect the spatial information. To address this issue, the authors put forth the suggestion of performing spatial grouping of features before introducing them into the aggregation process. For every feature map U_k, a scoring function is employed to calculate its corresponding attention map A_k. Afterward, the fine-grained structure mapping module, designed to learn the process of extracting n' c-d meaningful features through the spatial weighting of pixel-level characteristics, receives the feature maps and attention maps, i.e., U_k and A_k, respectively, as input. These resultant vectors are input into a feature aggregation technique, which produces the final set for regression, V, consisting of K c'-d features. Utilizing the vector V_k, a fully connected layer generates the stage outputs for the k-th stage, which are then substituted into the SSR function to derive the head pose. The resulting blocks representing the core of the network can be represented as:

$$\begin{aligned} B_R(c) &= \{SepConv2D(3x3, c) - BN - ReLu\} \\ B_t(c) &= \{SepConv2D(3x3, c) - BN - Tanh\} \end{aligned} \tag{1}$$

SepConv2D represent the separable convolution, BN is the Batch normalization of parameter c and ReLu and Tanh represent the two activation functions of the blocks. For more details about the scoring function and fine-grained structure mapping, please refer to [42]. Figure 4 shows the architecture of FSA-Net.



(a) FSA-Net (b) Fine-grained structure mapping

Fig. 4. Overview of FSA-Net architecture [42].

3.2 Web-Shaped Model

The method employed in [2] to estimate head pose consists of two major steps:

- *Facial landmark detection*: it detects 68 facial keypoints from a given input image of a face.
- *Web-Shaped Model* (WSM): the WSM is employed to draw a virtual spider's web on the input face image. It is then utilized to derive a descriptive array of the pose, which is determined by the position of the aforementioned landmarks.

Initially, the approach introduced in [23] is employed to predict 68 facial landmarks. These landmarks are denoted as coordinate points (x, y) on the input image. In particular, they provide a clear definition of the face contours, including the nose, lips, eyes, and eyebrows. The second method involves utilizing the Web-Shaped Model. This model is constructed by drawing a virtual spider web on the input image using the previously predicted landmark coordinates. The center of the spider web is placed at the tip of the nose, which corresponds to landmark number 33, and the radius is set to the distance from the center to the furthest landmark. The WSM categorizes each of the 68 facial landmarks (excluding landmark number 33) into a specific "*sector*" on the spider web based on its location. The radius r of the spider's web is determined by computing the Euclidean distance d between the center point $O = (x_{33}, y_{33})$ and the farthest landmark P_j, $j = 1, ..., 68$. The maximum value of $d(O, P_j)$ is selected as the radius r. Once the method assigns each facial keypoint to its respective sector on the spider web, it proceeds to extract a feature vector that characterizes the pose, taking into account the distribution of landmarks within each sector. This feature vector effectively captures the head's orientation and position in the input image. In WSM, a specifically created dataset of reference prototypes, called *Lara*, is utilized. The 3D model of a synthetic head is systematically rotated along the three axes, as illustrated in Figure 5. Different combinations of pitch, yaw, and roll angles are employed to generate representative exemplars, from which pose reference vectors are extracted.

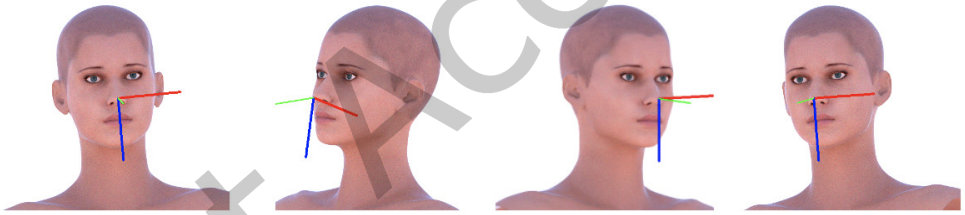


Fig. 5. Samples exhibiting pitch, yaw, and roll axes variations in Lara dataset.

According to the authors, the term "*circles*" describes the series of concentric circular shapes that comprise the spider's web. The term "*slices*" refers to the individual sections of the web that are formed by the intersection of two consecutive rays. In addition, a "*quarter*" is defined as one of the four quadrants of a Cartesian plane that is centered at the center of the spider-web. Finally, the term "*sectors*" is used to describe the various sections of the spider web that are delimited by both circles and slices. The main formulas to compute the position of the landmark in the spiderweb is:

$$\begin{aligned} a &= \widehat{YOP} \\ \theta &= 90/m \\ s &= \lceil a/\theta \rceil \end{aligned} \tag{2}$$

where θ is the width of the m slices, a is the relative coordinates of the point in the Cartesian plane centered at O are used to identify the quarter of the point. Finally, s identify the slice. The chosen spider-web configuration,

among those tested, is denoted as "*4C_4S_var4*", where *C* represents "*circles*" and *S* represents "*slices*". This particular configuration consists of four concentric circles and four slices for each quarter. The "*var4*" suffix denotes that this is the fourth variant tested, which pertains to different ray lengths. An array of values with a size equal to the number of sectors within the spider's web is extracted to represent the pose. This is determined by multiplying the number of slices by four quarters (which is always four) and the number of circles, resulting in a size of $m \times 4 \times n$. Thus, in the case of "*4C_4S_var4*", the size of the array is 64. The spider-web sectors are numbered in a clockwise order, starting from the outermost circle and moving towards the innermost one. Each array component contains information about the number of landmarks that fall within a particular sector. Further details on the relationship between landmarks and sectors can be found in [7]. To determine the pose in the extracted array, three regression models were created, each corresponding to the pitch, yaw, and roll axes, respectively. Figure 6 illustrates the framework of Web-Shaped Model.

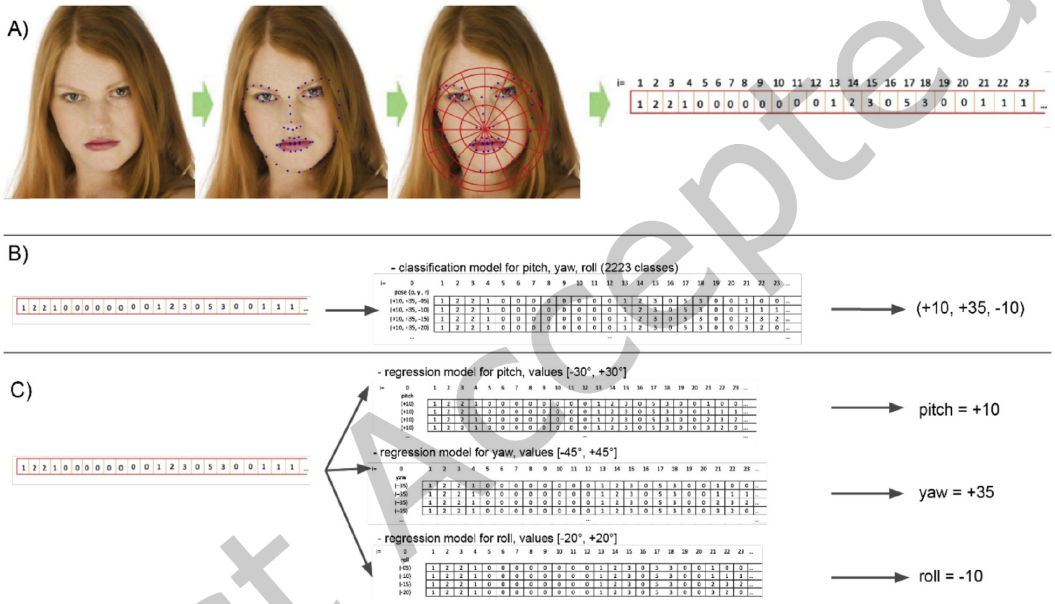


Fig. 6. Framework of Web-Shaped Model [2].

3.3 SynergyNet

The methodology introduced in [38], namely SynergyNet, endeavors to attain a superior level of precision and accuracy to predict complete 3D facial geometry. The framework comprises a two-stage pipeline that synergistically combines 3D landmarks and 3D Morphable Models (3DMM) parameters to optimize the learning of 3D facial geometry. In the first stage, the method performs a preliminary 3DMM regression using images and utilizes the multi-attribute feature aggregation technique to refine facial landmarks. In the second stage, the approach employs a landmark-to-3DMM regressor to uncover the embedded facial geometry in sparse keypoints. The 3DMM regression includes the estimation of parameters for pose, shape, and expression from a single frame utilizing a backbone network. To create 3D faces, 3DMM base models are used, followed by obtaining 3D landmarks from facial meshes. The multi-attribute feature aggregation technique is utilized to produce landmark structures that are more accurate. The method of regressing 3DMM parameters from 3D landmarks in the reverse

direction is used, presuming that the 3D landmarks provide rough facial geometry. This technique can get rid of the ambiguity that traditional 3DMM-based techniques frequently have because they only make assumptions about facial geometry from images.

Below is a concise overview of the primary modules that comprise the SynergyNet framework. The initial phase involves the backbone network (i.e., MobileNetV2), which is trained to perform two tasks: first, to regress the 3DMM parameters (α_p, α_s, and α_e) for pose, shape, and expression, respectively, and second, to reconstruct 3D face meshes using monocular face images. To accomplish this, the authors employ an existing regression framework called 3DDFA-V2 [16], which predicts 62-dimensional 3DMM parameters α. After regressing the *alpha* parameters, the 3D face mesh corresponding to the input image can be generated and aligned with the input face; the 3D landmarks L^c are extracted by utilizing landmark indices. To further refine the facial keypoints, the multi-attribute feature aggregation technique is utilized, which involves analyzing a range of characteristics, including landmark features, image features, and the shape and expression of the 3DMM semantics. Following that, the reverse direction module is developed to regress 3DMM parameters from the refined landmarks L^r, utilizing the holistic facial keypoint characteristics. SynergyNet enhances its representation strategy by utilizing a two-fold approach, incorporating both a forward progression from 3DMM parameters to accurately refined 3D landmarks and a backward regression from 3D landmarks to 3DMM parameters. The total loss of the method is composed as

$$\mathcal{L} = \lambda_1 \mathcal{L}_{3DMM} + \lambda_2 \mathcal{L}_{lmk} + \lambda_3 \mathcal{L}_{3DMM_{lmk}} + \lambda_4 \mathcal{L}_g \tag{3}$$

where the first term is the loss of the 3DMM module, the second term is the loss of the aligned loss of the landmark module, the third is the refined landmark 3DMM module and the forth module is a self-supervised loss. This methodology significantly enhances overall performance. The overall workflow of SynergyNet is in Figure 7.

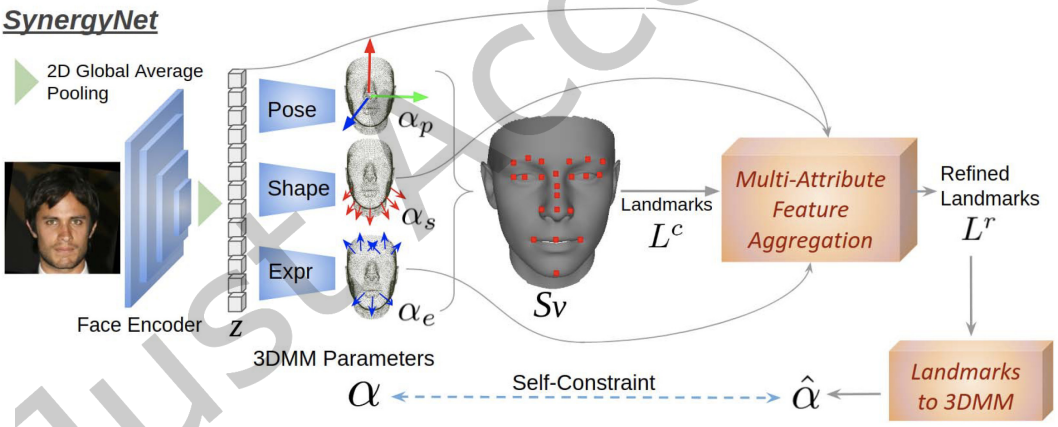


Fig. 7. Workflow of SynergyNet [38].

4 EXPERIMENTAL PROTOCOL

4.1 Datasets

Several datasets were presented as the state of the art to solve the deepfake detection problem. However, the majority of them, as the popular CelebDF [27] presents a high imbalance between real and deepfake videos number. In addition, they only perform one of the swaps (identity or expression). For this reason, to conduct a depth ablation study, we prefer to use the FaceForensics++ dataset [34]. This dataset has also been used in a very

popular deepfake detection challenge [15] of 2022, where an EfficientNet [13] based method reached the best results on creating a Deepfake detector capable of working in an "in the wild" scenario. The task of reconstructing the original video from the deepfake is still open. In the following, we will provide more details about this dataset.

The FaceForensics++ dataset [34] is a comprehensive repository of facial manipulation data that encompasses both real and fake portrait videos. The real videos were lawfully obtained from YouTube with the subjects' explicit consent, whereas the fake videos were produced utilizing four distinct manipulation techniques: Deepfakes, FaceSwap, Face2Face, and NeuralTextures. In the following paragraphs, a brief description of these methods is provided. The dataset contains 4,000 manipulated videos in total, with each manipulation method comprising 1,000 videos. Figure 8 illustrates some samples from the FF+ database.

Deepfakes *Deepfakes* have gained widespread recognition as a method for face replacement through the use of deep learning. This approach entails substituting a face in a target sequence with one derived from a source video or image collection. This technique involves training two autoencoders, which share an encoder, to reconstruct the images of both the source and target faces. Furthermore, the images are cropped and aligned using a face detector; finally, the encoder and decoder of the source face are employed on the target face to generate a fake image.

FaceSwap *FaceSwap* is a graphical approach used for transferring facial features from a source image to a destination image. The method involves using a facial landmark detection algorithm to extract the face region and then fitting a 3D template model with blendshapes. After being adjusted to match the localized keypoints and textures of the input image, the model is projected onto the target image. To produce a realistic result, the model is blended with the target image, and color correction methods are utilized.

Face2Face *Face2Face* represents a revolutionary technology that makes it possible to transfer facial expressions from a source image to another while simultaneously preserving the identity of the target person. In essence, this system enables only the facial expressions of different characters to be exchanged with one another. Face2Face implementation entails working with two distinct video input streams and manually selecting keyframes to start the process. Once the keyframes have been identified, a facial reconstruction is generated, which can then be used to recreate the face with different expressions and lighting conditions.

NeuralTextures *NeuralTextures* is a technology that allows for precise manipulation of facial features by utilizing the neural texture rendering method. This involves extracting the neural texture information of a target face and training a rendering network corresponding to the target face. The training phase entails fine-tuning the system with a photometric reconstruction loss combined with an adversarial loss, resulting in highly accurate facial reproductions.

4.2 Classification Methods

4.2.1 KNN with Dynamic Time Warping. The K-Nearest Neighbor algorithm is a well-known simple and effective classification algorithm of supervised learning [35]. The core of the method is represented by the proximity of the data, which is a discriminant in the prediction process. The K parameter represents the number of neighbors to take into consideration during the classification stage. KNN has three main steps:

- The learning data positions and characteristics are used to partition the space into regions, which can serve as the algorithm's learning set, despite not being explicitly mandated by the initial conditions.
- To calculate distance, information is depicted through position vectors within a space that has multiple dimensions. Although Euclidean distance is the norm, there are other types of distance that can be just as relevant.

Fig. 8. Example images from the FaceForensics++ dataset.

- The class is allocated to a point that represents an object, provided it's the most common class among the k nearest examples to the given object, where distance between points determines proximity. The known set of objects with accurate classifications is used to select the nearest neighbors.

In our case, the distance used is obtained by the Dynamic Time Warping (DTW). This algorithm proves to be incredibly advantageous for dealing with sequences that exhibit variations in characteristics over time, where conventional linear expansion or compression methods of such sequences fall short of delivering satisfying outcomes. Its application has been widespread, spanning across different domains such as speech recognition [36] and activity recognition [30].

The ideal match between two sequences refers to a match that satisfies all the conditions and regulations while incurring the lowest cost. The cost is calculated by summing up the absolute differences between the values of each matched index pair.

In our cases, the sequences are represented by the values of pitch, yaw, and roll extracted by the HPE method throughout the video.

Defined as $X = (x_{p1}, x_{y1}, x_{r1}, ..., x_{pN}, x_{yN}, x_{rN})$ and $Y = (y_{p1}, y_{y1}, y_{r1}, ..., y_{pM}, y_{yM}, y_{rM})$ the HPE sequences of two videos of length N and M, respectively, a warping path is an (N, M) path of length L

$$W = (w_1, ..., w_L) \quad with \quad w_l = (n_l, m_l) \in [1:N]x[1:M] \quad for \;\; l \in [1:L] \tag{4}$$

The path must satisfy the following conditions:

- ***Boundary*** The first and last indices of the sequences must be matched: $w_1 = (1, 1)$ and $w_L = (N, M)$.
- ***Monotonicity*** The mapping of indices between the two sequences must be in a strictly increasing order, and vice versa: if the indices are $j > i$ in the first sequence, none of the indices of the second sequence can be $l > k$ with k matched with j and l matched with i.
- ***Full matching*** There should be a one-to-one or one-to-many mapping between the indices in the two sequences, without any index left unpaired.

In order to evaluate the quality of the warping path, we need a way to quantitatively compare the elements in the feature sequences X and Y. Let be the Head Pose Patterns (HPP) our features space of which x_{pi}, x_{yi}, x_{ri} and y_{pj}, x_{yj}, x_{rj} belong. We will need a local cost measure that is a function c from $HPPxHPP$ to $\mathbb{R}$. By evaluating the cost for each pair of the sequence, we obtain a cost matrix, C. If $c(x_i, y_j)$ is the generic element of this matrix,

we can define the total cost of a warping path as:

$$c_p := \sum_{l=1}^{L} c(x_{il}, y_{jl}) = \sum_{l=1}^{L} C(i_l, j_l) \tag{5}$$

The definition of the warping path involves accumulating the cost of all cells it passes through. A warping path is considered "good" if its total cost is low and "bad" if its total cost is high. The goal is to find an optimal warping path between sequences X and Y, which is defined as the warping path that has the lowest total cost among all possible warping paths. The optimum warping path's cells encode the greatest feasible alignment between the elements of the two sequences, guaranteeing that each element of X is allocated to at least one element of Y and vice versa. This leads to the definition of the DTW distance, abbreviated as DWT, as the lowest cost of the best warping path between sequences X and Y:

$$DTW(X, Y) = min\{c_p(X, Y)|P \text{ is an } (N, M) - \text{warping path} \tag{6}$$

Since the number of possible warping paths is exponential, dynamic programming is necessary to obtain a feasible computation when N and M are large. The initial problem can be divided in subproblems and their solutions combined to obtain an overall solution. The concept of DTW is to deduce an optimal warping path for the original sequences from optimal warping pathways for truncated subsequences. This concept may then be applied recursively. Given $X(1:i) = (x_1, ..., x_i)$ for $i \in [1:N]$ and $Y(1:j) = (x_1, ..., x_j)$ for $j \in [1:M]$, we define their distance as $DTW(X(1:i), Y(1,j))$. This is a matrix D that can be defined as the *accumulated cost matrix*. Notice that $D(N, M) = DTW(X, Y)$. The cumulative cost matrix D is generated repeatedly using a nested loop in the first phase. The ideal warping path is determined using a backtracking approach in the second half. The detailed pseudocode to compute the DTW is shown in Algorithm 1.

In our case, DTW is particularly effective at aligning sequences of different lengths. We have different lengths because the video may contain a different number of frames or because the HPE method selected is not able to find the face when extreme rotations occur. For this reason, we have that also in the same video, but in its real and its fake versions, we may obtain sequences of different lengths.

In Figure 9, we depicted the differences in the comparisons between two sequences with Euclidean distance and with DTW. As can be seen, the DTW does not compare frames based on their order. On the contrary, it is able to handle disaligned sequences.

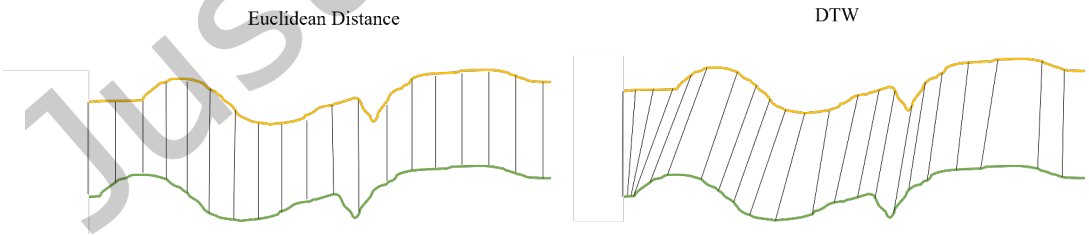


Fig. 9. An example of the differences between the use of Euclidean distance and DTW to align two sequences.

4.2.2 Deep Learning Models. We also study the effectiveness of modeling head poses with deep learning approaches. In particular, we propose two methods that process the sequences of head yaw, pitch, and roll using two orthogonal approaches: 1D convolutions (Fig. 10) and recurrent neural networks (Fig. 11).

Algorithm 1 The DTW algorithm

Require: C
Ensure: D, The optimal path P'
  $D(i,1) = \sum_{h=1}^{i} C(k,1)$ with $i \in [1:N]$
  $D(1,j) = \sum_{h=1}^{j} C(1,k)$ with $j \in [1:M]$
  for $i = 2, ..., N$ and $j = 2, ..., M$ **do**
    $D(i,j) = C(i,j) + min\{D(i-1,j-1), D(i-1,j), D(i,j-1)\}$
  end for
  $l = 1$
  $t_l = (N, M)$
  while $t_l \neq (1,1)$ **do**
    l=l+1
    $(i,j) = t_{l-1}$
    if i=1 **then**
      $t_l = (1, j-1)$
    else if m=1 **then**
      $t_l = (i-1, j)$
    else$t_l = armin\{D(i-1,j-1), D(i-1,j), D(i,j-1)\}$
    end if
  end while
  $L = l$
  $P' = (t_L, t_{L-1}, ..., t_1)$
  return P', D

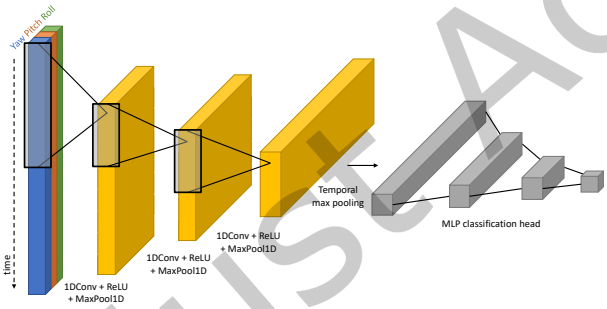


Fig. 10. Convolutional model architecture.

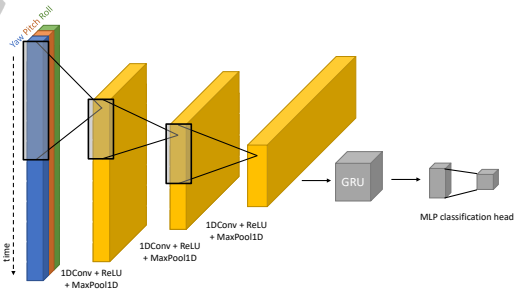


Fig. 11. Recurrent model architecture.

1D Convolution Model. To model time series, 1D convolutions have shown remarkable results [22]. We adapt this idea to our framework by considering each video as a sequence of head pose patterns, as done in Sec. 4.2.1. Therefore, a sample is a sequence of $\{x_{pi}, x_{yi}\}, x_{ri}$, where the subscripts p, y, and r respectively denote pitch, yaw, and roll, and i indicates the temporal index in the time series. In short, a sample can be denoted as $\mathbf{x} \in \mathbb{R}^{N\times 3}$, where N is the number of frames in the video.

The rationale behind the usage of 1D convolutions is that a kernel of size k can aggregate a neighborhood of samples in a sliding window fashion across the time sequence and a richer description can be provided by increasingly adding information along the channel dimension while pooling the temporal one.

We build a model by stacking three 1D convolutional layers with kernel sizes 6, 3, and 3, respectively, mapping the number of channels from 3 to 6, 8, and 12. For every layer, we use stride one and padding zero, and we apply a ReLU activation followed by a 1D max pooling with size 3 and stride 2. The 1D max pooling helps in creating a more compact representation, which reduces the extent of the temporal pattern. To make the model free to process arbitrary length sequences, we then perform a max reduction over the temporal dimension, yielding a 12-dimensional vector that is processed by a final classification module made of a multilayer perceptron with two layers and a hidden dimension of 4. A sigmoid activation is used to generate classification scores.

Recurrent Model. Recurrent Neural Networks (RNN) are the most common deep learning based architecture to process temporal data. These networks update an internal hidden layer while processing the inputs sequentially, one at a time. We exploit a particular instance of an RNN named Gated Recurrent Unit (GRU) [10], which, similarly to LSTMS [19], leverages an update gate and a reset gate to retain or discard information from past timesteps. We therefore modify the 1D convolution network by inserting a cascade of two GRU layers to perform temporal pooling after the signal has been pre-processed by the 1D convolutional layers. In this way, we replace the *max* operation along the temporal axis with a trainable set of weights that learns to condense the history of the whole sequence into a hidden state. The two GRU layers that we introduce have a number of hidden states equal to 24 and 36 and are followed by a linear layer with ReLU activation and a hidden dimension of 8, and a final classification layer with a single output followed by a sigmoid.

5 RESULTS

In this section, we will present our ablation studies and the final accuracy for the different configurations of the proposed architectures.

5.1 Results of KNN with DTW

The KNN method has as a parameter the number of neighbors to consider. The DTW method needs to know the warping parameter. The warping parameter indicates every how many values to align the sequences. The smaller this value is, the higher the computational cost will be. This is because a small value indicates sequences that need to be aligned more often. We started with $k = 1$ to examine the ideal warping window. In Table 2 we show this first study using FSA-Net as the HPE method. Even if the number of iterations increases, the results do not show better results when warping windows are smaller. This happens not only when deepfakes are obtained by FaceSwap but also with Face2Face. Since the length of the sequences is 300, we can claim that the optimal configuration in terms of computational cost and performance is represented by sequences aligned three times.

The time required to perform the estimation is directly proportional to the number of iterations. This because the iterations depend inversely on the warping windows dimension. The higher the number of warping windows, the lower the frequency of sequence alignment. Less alignment correspond to less iterations to perform and, as a consequence, a lower computational time required.

When we test other values of k we can notice that performances does not increase. In some cases, they also decrease. This indicates that when the correct warping window is selected, the best match to perform classification is represented by the closest point.

FSA-Net presented better performances on FaceSwap compared to Face2Face. This is easily explainable by the fact that FaceSwap has a "bad" construction of deepfake. In some cases, they are also easily detected by the human eye. In general, the behavior of this method is the same with respect to the parameters k and w when we compare FaceSwap and Face2Face.

As we can see in Table 3, the results are considerably worse than FSA-Net. The behavior of the parameters k and w is the same in FSA-Net and also in FaceSwap and Face2Face. However, overall performance results decreased significantly.This indicates that SynergyNet provides real and fake HP sequences with very similar

Table 2. Performances of the FSA-Net HPE method on the FF++ Dataset.

FF++ with FSA-Net	Original vs FaceSwap				
k value	Warping window	# of iterations	Accuracy	f1-score Real	f1-score Fake
1	100	6368	0.9	0.9	0.9
1	50	12746	0.9	0.9	0.9
1	25	25472	0.87	0.88	0.87
3	100	6368	0.9	0.9	0.9
5	100	6368	0.85	0.87	0.83

FF++ with FSA-Net	Original vs Face2Face				
k value	Warping window	# of iterations	Accuracy	f1-score Real	f1-score Fake
1	100	6368	0.82	0.82	0.81
1	50	12746	0.76	0.76	0.77
1	25	25472	0.78	0.79	0.77
3	100	6368	0.8	0.82	0.78
5	100	6368	0.63	0.7	0.51

values. Another important difference between the behavior of SynergyNet and FSA-Net is represented by the F1-score. SynergyNet is less balanced between the real and fake classes. In the majority of cases, it prefers to classify sequences as real, and the fake sequences are more often misclassified.

Table 3. Performances of the SynergyNet HPE method on the FF++ Dataset.

FF++ with SynergyNet	Original vs FaceSwap				
k value	Warping window	# of iterations	Accuracy	f1-score Real	f1-score Fake
1	100	6368	0.54	0.59	0.48
1	50	12736	0.54	0.52	0.55
1	25	25472	0.53	0.56	0.51
3	100	6368	0.52	0.54	0.5
5	100	6368	0.53	0.53	0.53

FF++ with SynergyNet	Original vs Face2Face				
k value	Warping window	# of iterations	Accuracy	f1-score Real	f1-score Fake
1	100		0.48	0.58	0.32
1	50		0.5	0.52	0.47
1	25		0.5	0.55	0.44
3	100		0.47	0.53	0.39
5	100		0.49	0.55	0.42

Also in the case of WSM, the accuracies are not satisfactory. However, if we compare the F1-score of real and fake samples, the situation is more balanced. It means WSM is not good to classify real and fake sequences, but it is able to estimate real end fake in the same way. Only in two cases is the difference higher, when the k value is 3. The use of a synthetic model in an HPE method (as WSM) may improve the latter's ability to generalize and its overall performance. However, it does not affect its performance on a deepfake detection task, demonstrating that the effectiveness of one method over another does not depend on the dataset on which the method has been trained.

Table 4. Performances of the WSM HPE method on the FF++ Dataset.

FF++ with WSM	**Original vs FaceSwap**				
k value	**Warping window**	**# of iterations**	**Accuracy**	**f1-score Real**	**f1-score Fake**
1	100	6368	0.58	0.55	0.6
1	50	12736	0.54	0.54	0.54
1	25	25472	0.54	0.54	0.53
3	100	6368	0.57	0.45	0.65
5	100	6368	0.59	0.51	0.65

FF++ with WSM	**Original vs Face2Face**				
k value	**Warping window**	**# of iterations**	**Accuracy**	**f1-score Real**	**f1-score Fake**
1	100	0.48	0.5	0.46	0.32
1	50	0.48	0.51	0.45	0.47
1	25	0.46	0.48	0.44	0.44
3	100	0.49	0.45	0.52	0.39
5	100	0.45	0.47	0.43	0.42

Based on those results, we can claim that FSA-Net is a good HPE method to distinguish between the real and fake videos. On the contrary, WSM and SynergyNet, independently of the configuration used, do not produce sufficiently high results.

This suggests that when we want to perform Deepfake detection by using HPE methods, the *kind* of method is more important than the classification method parameter. We can also observe that a good HPE method for this purpose is also stable respect to the different deepfake method used.

In Figure 12 we analyze the first video sequence of FF++ to examine the discrepancy between real and fake videos from the point of view of the HPE method. As can be noticed, SynergyNet shows very few differences between a real and fake sequence; this can be the cause of the bad performances on the deepfake detection problem. Synergynet has flattened patterns, this can be the result of their use of a 3DMM that limits extreme variations between two poses. Even if the resulting is a HPE method with a lower mean error, the same is not suitable to perform deepfake detection. On the contrary, WSM shows a higher level of freedom in the variations of the poses. However, real and deepfake video pattern have several point in common. The optimal configuration of the web-shape for the HPE task is, in our opinion, not sufficient to highlight the difference between a real and a deepfake video. The effect of a denser spider web could be studied. As we expect, FSA-Net is the HPE method that maximize the differences between real and deepfake videos. Peak are not correspondent, and fake videos seem to be more flattened compared to the real ones. The optimal results obtained by FSA-Net are probably the consequence of the multi-view features that they adopt. In particular, the authors focused on spatial feature aggregation prior to adopting regression.

5.2 Results with Deep Learning Methods

Here we report the results obtained by our two deep learning approaches that process sequences of head poses. Both the 1D convolutional model and the recurrent model were trained using the Adam optimizer with a learning rate of 0.0005. To train the models using batches of samples, we zero-padded all the samples to match the longest sample in every batch.

In Tab. 5 we show the results obtained by the 1D convolutional model on the FF++ dataset for the FSA-Net, SynergyNet, and WSM head pose predictors and for two different deepfake approaches: faceswap and face2face. We train a different model for every setting, treating it as a binary classification problem, i.e. feeding to the

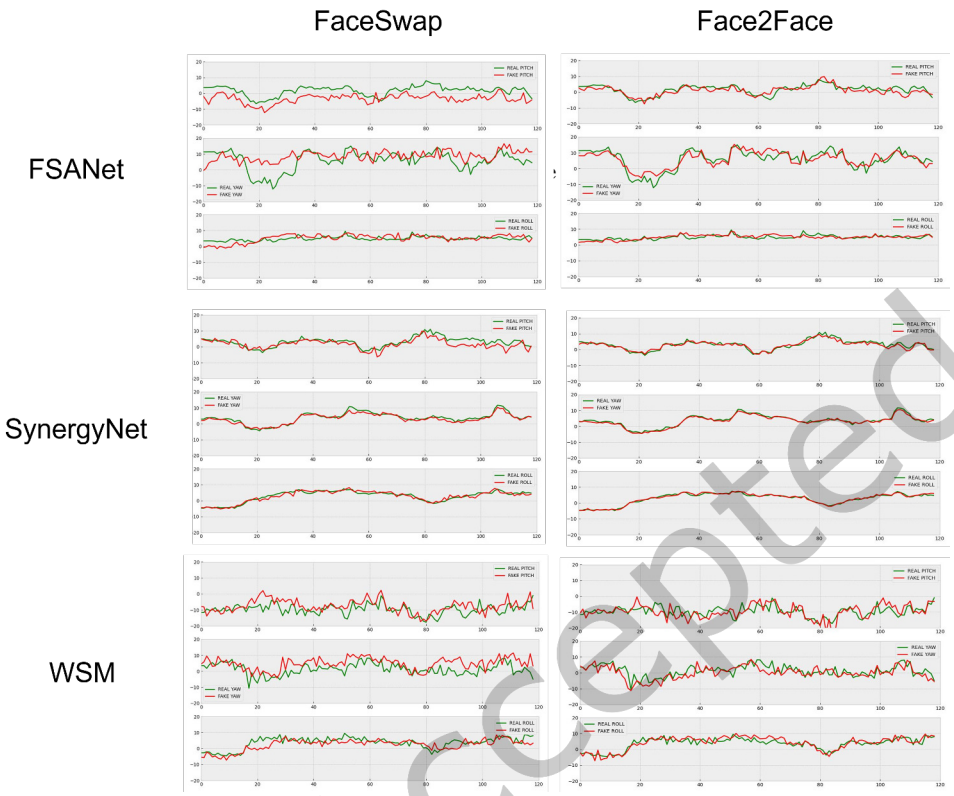


Fig. 12. The comparisons of the discrepancy between real and fake head poses extracted by FSA-Net, SynergyNet and WSM. Data were extracted by the first video of the first subject of FF++.

network the sequences head poses estimated on the original videos and on the attacked videos. As for the DTW approach, for every setting we use a single split of the dataset by leveraging 80% of the data for training the model and the remaining 20% for testing. From the results, it can be seen that the model struggles to deal with Faceswap, obtaining slightly above random results. More accurate results are instead obtained by the model against Face2Face, without however noticing big differences when using a head pose descriptor rather than another.

Interestingly, looking at Tab. 6 the recurrent model instead shows an opposite trend. In this case, in fact, we obtain better results against Faceswap, with poor performances against Face2Face. We attribute the higher accuracy on Faceswap with the recurrent model to its improved capacity to model temporal dynamics with respect to the 1D convolutional model. In fact the latter makes a temporal reduction of the temporal dimension via max pooling, thus temporal patterns can only be captured locally within the receptive field of the convolutions. On the other hand, the recurrent model can observe the whole sequence and identify longer-term patterns. At the same time, the higher complexity of the recurrent model leads to higher overfitting in Face2Face, where the convolutional model manages to find local inconsistencies more effectively. Overall, however, we notice that both deep learning methods obtain results that are better or on par compared to KNN with DTW when using SynergyNet and WSM (see Tab. 3 and Tab. 4). Yet much lower results are obtained when compared to KNN with

DTW using FSA-Net. We attribute this gap to a training issue due to overfitting and a low amount of training data, which only affects learning-based methods. In addition, RNN models are known to suffer from vanishing and exploding gradient issues when long sequences are used for training.

Table 5. Performances of the 1D Convolution Model on the FF++ Dataset.

Training configuration (FF++)	Accuracy	f1-score real	f1-score fake
FsaNet_Faceswap	0.55	0.57	0.48
FsaNet_Face2Face	0.61	0.38	0.56
SynergyNet_Faceswap	0.57	0.47	0.50
SynergyNet_Face2Face	0.61	0.50	0.65
WSM_Faceswap	0.58	0.59	0.49
WSM_Face2Face	0.60	0.51	0.55

Table 6. Performances of the recurrent model on the FF++ Dataset.

Training configuration (FF++)	Accuracy	f1-score real	f1-score fake
FsaNet_Faceswap	0.62	0.56	0.68
FsaNet_Face2Face	0.51	0.01	0.67
SynergyNet_Faceswap	0.62	0.52	0.65
SynergyNet_Face2Face	0.57	0.55	0.55
WSM_Faceswap	0.61	0.51	0.67
WSM_Face2Face	0.51	0.66	0.06

5.3 Cross-dataset validation and comparisons with the SOTA

From the results previously obtained, it is clear that the best configuration for HPE-based systems uses FSA-Net as the head pose extractor and KNN with DTW as the classifier. The best parameters in terms of computational cost and performance are $k = 1$ and $w = 100$. For this reason, we will use the latter to perform the following experiments on the remaining part of FF++: the cross-valdation experiments.

In Table 7 we show the results of the method on all of the subsets of FF++. On the main diagonal, we can find the results for each subset; on the remaining parts of the table are the cross-dataset evaluations.

Table 7. The Cross-Dataset evaluation performed on the subset of FF++. The HPE method is FSA-Net with k=1, w=100.

FF++ Cross-Dataset Evaluation				
Train/Test	**DeepFake**	**Face2Face**	**FaceSwap**	**NeuralTexture**
DeepFake	0.82	0.83	0.88	0.89
Face2Face	0.8	0.82	0.88	0.88
FaceSwap	0.8	0.84	0.9	0.91
NeuralTexture	0.81	0.84	0.91	0.91

As we can observe, the HPE-based method is very stable with respect to cross-validation. All of the results present an accuracy ≥ 0.8. In particular, we can notice that all of the training sets provide the best transferability on neural texture (an expression swap approach). The worse transferability is on the deepfake subset (an identity swap approach).

In order to evaluate if the HPE methods proposed here to detect deepfakes are in line with the state of the art, we perform the comparisons in this section.

In Table 8 are shown the results of different methods introduced in section 2. The best results in mean are obtained by Liu et al. [28], with a method based on semantic discrepancy. The mean accuracy of the SOTA is 0.831. The HPE-based method reach 0.86 in accuracy, higher than the SOTA mean, and surpass 4 out of 10 of them. Considering that the HPE-based method does not use specific deepfake detector features, but instead uses the output of existing methods, this is a considerable result. We can also notice that, when compared with others, the HPE-based system is very stable with respect to the different deepfake generators. The mean difference between the accuracy on the different sets is 0.081 in the SOTA, and only 0.042 in the HPE-based technique.

Table 8. Comparisons with the state of the art on Face2Face and FaceSwap from FF++. The HPE-based method used is FSA-Net with k=1, w=100.

Method	Year	DeepFake	Face2Face	FaceSwap	NeuralTexture	Mean (FF++)
Khalid et al. [24]	2023	0.98	0.62	0.98	0.75	0.83
Xu et al. [41]	2023	0.92	0.92	0.91	0.91	0.92
Liu et al. [28]	2023	0.96	0.98	0.99	0.92	0.96
Nirkin et al. [31]	2022	0.94	0.8	0.84	0.74	0.83
Ilyas et al. [21]	2022	0.93	0.92	0.93	0.78	0.89
Heo et al. [18]	2021	0.96	0.93	0.95	-	-
Demir et al. [14]	2021	0.93	0.59	0.91	0.57	0.75
Chugh et al. [12]	2020	0.94	0.93	0.95	-	-
Xie et al. [40]	2020	0.93	0.91	0.74	0.65	0.81
Khalid et al. [25]	2020	0.88	0.71	0.86	0.87	0.83
Rossler et al. [33]	2019	0.96	0.86	0.9	0.8	0.88
Afchar et al. [5]	2018	0.87	0.56	0.61	0.4	0.61
HPE-based		0.82	0.82	0.9	0.91	0.86

In Table 9 we present the cross-dataset evaluation with the authors Khalid et al. [24] who also performed the same experiments. In particular, they focused on the transferability between indentity swap and expression swap. They trained the method on both the identity and expression swaps, respectively, while we prefer to train with the two subsets separately to ensure the data remain balanced. As can be noticed, in the identity swap, we obtained far higher results than Khalid et al. [24], outperforming the test accuracy on NeuralTextures of 0.3 and 0.28 using FaceSwap and DeepFake respectively. Less evident are the discrepancy between our and their performances by testing on Face2Face and on the Expression Swap as train set. We can claim that the FSA-Net based method is more stable to cross-validation than the method at the SOTA.

In addition, it is essential to underline that the method by Khalid et al. [24] trained on FS and DF considered them as a single dataset. This means they have the advantage of a more rich dataset to extract deepfake characteristics. On the other side, we only trained on one set at a time. This allow us to observe the performances obtained by learning the characteristics of each dataset and lighten the training process.

Furthermore, it has to be noted that transferability from Face2Face and NeuralTextures to DeepFake and FaceSwap is better than the other way around. In fact, we can group the approaches into two distinct categories: DeepFake and FaceSwap perform an identity swap, rendering faces that have the same pose and expression of the source, yet changing the identity to match a given target; on the contrary, Face2Face and NeuralTextures apply a target expression maintaining the source identity unaltered. This means that Face2Face and NeuralTextures result in more realistic rendering since the task is less invasive and fewer alterations are required (see. Fig. 8). As a direct consequence, detecting expression swaps is in general harder. Thus, a model trained on Face2Face and

Table 9. Cross-datasets and cross-domain comparisons with the SOTA. The HPE-based method used is FSA-Net with k=1, w=100. DF=DeepFake, FS=FaceSwap, F2F=Face2Face, NT=NeuralTextures.

Identity Swap as Train - Expression Swap as Test			
Comparisons	**Train**	**NT**	**F2F**
Khalid et al. [24]	FS+DF	0.61	0.73
HPE-based	FS	0.91	0.84
HPE-based	DF	0.89	0.83
Expression Swap as Train - Identity Swap as Test			
Comparisons	**Train**	**DF**	**FS**
Khalid et al. [24]	F2F+NT	0.7	0.73
HPE-based	F2F	0.8	0.88
HPE-based	NT	0.81	0.91

NeuralTextures needs to discover more subtle attacks in the image, which leads to improved transfer capability towards more macroscopic attacks such as FS and DF.

6 CONCLUSIONS

Deepfake videos put cybersecurity, privacy, and democracy at risk. They may be used to make fake and edited videos of individuals that seem real yet have been created artificially using artificial intelligence algorithms. In this paper, we have presented a study regarding the usage of head pose estimation approaches to detect deep fake tampered videos. We used three different head pose estimation approaches, showing that real and fake videos tend to exhibit different distributions. We offered two different lines of investigation: a distance-based approach using KNN with Dynamic Time Warping and a learning-based approach, training both a 1D convolutional model and a recurrent model for deepfake detection. From our experiments it appeared that head pose estimation data can indeed be used effectively to identify tampered videos, however, training neural network methods for this task is not trivial. On the other hand, the KNN with DTW approach proved to be extremely effective in addressing this task. The findings of employing HPE-based approaches on the FF++ dataset reveal that the performance difference may be linked to the specific head pose extraction method utilized. In particular, when utilized as a features extractor in conjunction with KNN and DTW classification, FSA-Net displayed performance similar to the state of the art in our study. The same yielded consistent results in cross-dataset validation, even when the datasets evaluated were used for various objectives in making the deepfake videos (identity swap and expression swap). In particular, we obtained a mean accuracy of 0.86 on the overall FF++. The mean accuracy on the identity swap is 0.86 and the mean accuracy on the expression swap is 0.865, demonstrating the method with FSANet+DTW is stable compared to different kind of deepfakes. The experiment carried out sets the path for a more extensive evaluation of the usage of HPE approaches to detect deepfake patterns. In the future, more HPE approaches, particularly those using a pose estimation methodology similar to FSA-Net, may be researched. It would also be interesting to see whether various extractor settings that did not yield ideal results for HPE would actually perform better for the task at hand.

ACKNOWLEDGEMENTS

This work was partially supported by the project IDA included in the Spoke 2 - Misinformation and Fakes of the Research and Innovation Program PE00000014, “SEcurity and RIghts in the CyberSpace (SERICS)”, under the National Recovery and Resilience Plan, Mission 4 "Education and Research" - Component 2 "From Research to Enterprise" - Investment 1.3, funded by the European Union - NextGenerationEU.

REFERENCES

[1] 2021. Face Swap algorithm. https://faceswap.dev/

[2] Andrea F Abate, Paola Barra, Chiara Pero, and Maurizio Tucci. 2020. Head pose estimation by regression algorithm. *Pattern Recognition Letters* 140 (2020), 179–185.

[3] Andrea F. Abate, Carmen Bisogni, Aniello Castiglione, and Michele Nappi. 2022. Head pose estimation: An extensive survey on recent techniques and applications. *Pattern Recognition* 127 (2022), 108591. https://doi.org/10.1016/j.patcog.2022.108591

[4] Darius Afchar, Vincent Nozick, Junichi Yamagishi, and Isao Echizen. 2018. MesoNet: A compact facial video forgery detection network. (Sept. 2018). arXiv:1809.00888 [cs.CV]

[5] Darius Afchar, Vincent Nozick, Junichi Yamagishi, and I. Echizen. 2018. MesoNet: a Compact Facial Video Forgery Detection Network. 1–7. https://doi.org/10.1109/WIFS.2018.8630761

[6] Saadaldeen Rashid Ahmed, Emrullah Sonuç, Mohammed Rashid Ahmed, and Adil Deniz Duru. 2022. Analysis Survey on Deepfake detection and Recognition with Convolutional Neural Networks. In *2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*. 1–7. https://doi.org/10.1109/HORA55278.2022.9799858

[7] Paola Barra, Silvio Barra, Carmen Bisogni, Maria De Marsico, and Michele Nappi. 2020. Web-shaped model for head pose estimation: An approach for best exemplar selection. *IEEE Transactions on Image Processing* 29 (2020), 5457–5468.

[8] Carmen Bisogni, Michele Nappi, Chiara Pero, and Stefano Ricciardi. 2021. FASHE: A FrActal Based Strategy for Head Pose Estimation. *IEEE Transactions on Image Processing* 30 (2021), 3192–3203. https://doi.org/10.1109/TIP.2021.3059409

[9] Carmen Bisogni, Michele Nappi, Chiara Pero, and Stefano Ricciardi. 2021. HP2IFS: Head Pose estimation exploiting Partitioned Iterated Function Systems. In *2020 25th International Conference on Pattern Recognition (ICPR)*. 1725–1730. https://doi.org/10.1109/ICPR48806.2021.9413227

[10] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[11] Francois Chollet. 2017. Xception: Deep learning with depthwise separable convolutions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Honolulu, HI). IEEE.

[12] Komal Chugh, Parul Gupta, Abhinav Dhall, and Ramanathan Subramanian. 2020. Not Made for Each Other- Audio-Visual Dissonance-Based Deepfake Detection and Localization. In *Proceedings of the 28th ACM International Conference on Multimedia* (Seattle, WA, USA) *(MM '20)*. Association for Computing Machinery, New York, NY, USA, 439–447. https://doi.org/10.1145/3394171.3413700

[13] Davide Alessandro Coccomini, Nicola Messina, Claudio Gennaro, and Fabrizio Falchi. 2022. Combining EfficientNet and vision transformers for video deepfake detection. *Image Analysis and Processing – ICIAP 2022* (2022), 219–229. https://doi.org/10.1007/978-3-031-06433-3_19

[14] Ilke Demir and Umur Aybars Ciftci. 2021. Where Do Deep Fakes Look? Synthetic Face Detection via Gaze Tracking. *ACM Symposium on Eye Tracking Research and Applications* (2021).

[15] Luca Guarnera, Oliver Giudice, Francesco Guarnera, Alessandro Ortis, Giovanni Puglisi, Antonino Paratore, Linh M. Bui, Marco Fontani, Davide Alessandro Coccomini, Roberto Caldelli, and et al. 2022. The Face Deepfake Detection Challenge. *Journal of Imaging* 8, 10 (2022), 263. https://doi.org/10.3390/jimaging8100263

[16] Jianzhu Guo, Xiangyu Zhu, Yang Yang, Fan Yang, Zhen Lei, and Stan Z Li. 2020. Towards fast, accurate and stable 3d dense face alignment. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIX*. Springer, 152–168.

[17] Aryaman Gupta, Kalpit Thakkar, Vineet Gandhi, and P J Narayanan. 2019. Nose, Eyes and Ears: Head Pose Estimation by Locating Facial Keypoints. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 1977–1981. https://doi.org/10.1109/ICASSP.2019.8683503

[18] Young Jin Heo, Young Ju Choi, Young-Woon Lee, and Byung-Gyu Kim. 2021. Deepfake Detection Scheme Based on Vision Transformer and Distillation. *CoRR* abs/2104.01353 (2021). arXiv:2104.01353 https://arxiv.org/abs/2104.01353

[19] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[20] Heng-Wei Hsu, Tung-Yu Wu, Sheng Wan, Wing Hung Wong, and Chen-Yi Lee. 2019. QuatNet: Quaternion-Based Head Pose Estimation With Multiregression Loss. *IEEE Transactions on Multimedia* 21, 4 (2019), 1035–1046. https://doi.org/10.1109/TMM.2018.2866770

[21] Hafsa Ilyas, Aun Irtaza, Ali Javed, and Khalid Mahmood Malik. 2022. Deepfakes Examiner: An End-to-End Deep Learning Model for Deepfakes Videos Detection. In *2022 16th International Conference on Open Source Systems and Technologies (ICOSST)*. 1–6. https://doi.org/10.1109/ICOSST57195.2022.10016871

[22] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. 2019. Deep learning for time series classification: a review. *Data mining and knowledge discovery* 33, 4 (2019), 917–963.

[23] Vahid Kazemi and Josephine Sullivan. 2014. One millisecond face alignment with an ensemble of regression trees. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1867–1874.

[24] Fatima Khalid, Ali Javed, Qurat ul ain, Hafsa Ilyas, and Aun Irtaza. 2023. DFGNN: An interpretable and generalized graph neural network for deepfakes detection. *Expert Systems with Applications* 222 (2023), 119843. https://doi.org/10.1016/j.eswa.2023.119843

[25] Hasam Khalid and Simon S. Woo. 2020. OC-FakeDect: Classifying Deepfakes Using One-class Variational Autoencoder. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2794–2803. https://doi.org/10.1109/CVPRW50498.2020.00336

[26] Jing Li, Jiang Wang, and Farhan Ullah. 2020. An End-to-End Task-Simplified and Anchor-Guided Deep Learning Framework for Image-Based Head Pose Estimation. *IEEE Access* 8 (2020), 42458–42468. https://doi.org/10.1109/ACCESS.2020.2977346

[27] Yuezun Li, Xin Yang, Pu Sun, Honggang Qi, and Siwei Lyu. 2020. Celeb-df: A large-scale challenging dataset for deepfake forensics. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 3207–3216.

[28] Xiaolong Liu, Yang Yu, Xiaolong Li, Yao Zhao, and Guodong Guo. 2022. TCSD: Triple Complementary Streams Detector for Comprehensive Deepfake Detection. *ACM Trans. Multimedia Comput. Commun. Appl.* (aug 2022). https://doi.org/10.1145/3558004 Just Accepted.

[29] Zhaoxiang Liu, Zezhou Chen, Jinqiang Bai, Shaohua Li, and Shiguo Lian. 2019. Facial Pose Estimation by Deep Learning from Label Distributions. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. 1232–1240. https://doi.org/10.1109/ICCVW.2019.00156

[30] Hoda Mohammadzade, Soheil Hosseini, Mohammad Reza Rezaei-Dastjerdehei, and Mohsen Tabejamaat. 2021. Dynamic Time Warping-Based Features With Class-Specific Joint Importance Maps for Action Recognition Using Kinect Depth Sensor. *IEEE Sensors Journal* 21, 7 (2021), 9300–9313. https://doi.org/10.1109/JSEN.2021.3051497

[31] Yuval Nirkin, Lior Wolf, Yosi Keller, and Tal Hassner. 2022. DeepFake Detection Based on Discrepancies Between Faces and Their Context. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 10 (2022), 6111–6121. https://doi.org/10.1109/TPAMI.2021.3093446

[32] Md Shohel Rana, Mohammad Nur Nobi, Beddhu Murali, and Andrew H. Sung. 2022. Deepfake Detection: A Systematic Literature Review. *IEEE Access* 10 (2022), 25494–25513. https://doi.org/10.1109/ACCESS.2022.3154404

[33] Andreas Rossler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Niessner. 2019. FaceForensics++: Learning to Detect Manipulated Facial Images. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 1–11. https://doi.org/10.1109/ICCV.2019.00009

[34] Andreas Rossler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. 2019. Faceforensics++: Learning to detect manipulated facial images. In *Proceedings of the IEEE/CVF international conference on computer vision*. 1–11.

[35] Gregory Shakhnarovich, Trevor Darrell, and Piotr Indyk. 2005. *Nearest-neighbor Methods in Learning and Vision*. Mit Press.

[36] Meenakshi Sood and Shruti Jain. 2021. Speech recognition employing MFCC and dynamic time warping algorithm. In *Innovations in Information and Communication Technologies (IICT-2020)*. Springer International Publishing, Cham, 235–242.

[37] Roberto Valle, José M. Buenaposada, and Luis Baumela. 2021. Multi-Task Head Pose Estimation in-the-Wild. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43, 8 (2021), 2874–2881. https://doi.org/10.1109/TPAMI.2020.3046323

[38] Cho-Ying Wu, Qiangeng Xu, and Ulrich Neumann. 2021. Synergy between 3dmm and 3d landmarks for accurate 3d facial geometry. In *2021 International Conference on 3D Vision (3DV)*. IEEE, 453–463.

[39] Jiahao Xia, Libo Cao, Guanjun Zhang, and Jiacai Liao. 2019. Head Pose Estimation in the Wild Assisted by Facial Landmarks Based on Convolutional Neural Networks. *IEEE Access* 7 (2019), 48470–48483. https://doi.org/10.1109/ACCESS.2019.2909327

[40] Daniel Xie, Prosenjit Chatterjee, Zhipeng Liu, Kaushik Roy, and Edoh Kossi. 2020. DeepFake Detection on Publicly Available Datasets using Modified AlexNet. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. 1866–1871. https://doi.org/10.1109/SSCI47803.2020.9308428

[41] Ying Xu, Kiran Raja, Luisa Verdoliva, and Marius Pedersen. 2023. Learning Pairwise Interaction for Generalizable DeepFake Detection. In *2023 IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW)*. 1–11. https://doi.org/10.1109/WACVW58289.2023.00074

[42] Tsun-Yi Yang, Yi-Ting Chen, Yen-Yu Lin, and Yung-Yu Chuang. 2019. FSA-Net: Learning Fine-Grained Structure Aggregation for Head Pose Estimation From a Single Image. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 1087–1096.

[43] Xin Yang, Yuezun Li, and Siwei Lyu. 2019. Exposing Deep Fakes Using Inconsistent Head Poses. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 8261–8265. https://doi.org/10.1109/ICASSP.2019.8683164

[44] Peipeng Yu, Zhihua Xia, Jianwei Fei, and Yujiang Lu. 2021. A Survey on Deepfake Video Detection. *IET Biometrics* 10, 6 (2021), 607–624. https://doi.org/10.1049/bme2.12031