Audio Splicing Detection and Localization Based on Acquisition Device Traces

Daniele Ugo Leonzio, *Student Member, IEEE,* Luca Cuccovillo, Paolo Bestagini, *Member, IEEE,* Marco Marcon, *Member, IEEE,* Patrick Aichroth, and Stefano Tubaro, *Senior Member, IEEE*

***Abstract*—In recent years, the multimedia forensic community has put a great effort in developing solutions to assess the integrity and authenticity of multimedia objects, focusing especially on manipulations applied by means of advanced deep learning techniques. However, in addition to complex forgeries as the deepfakes, very simple yet effective manipulation techniques not involving any use of state-of-the-art editing tools still exist and prove dangerous. This is the case of audio splicing for speech signals, i.e., to concatenate and combine multiple speech segments obtained from different recordings of a person in order to cast a new fake speech. Indeed, by simply adding a few words to an existing speech we can completely alter its meaning. In this work, we address the overlooked problem of detection and localization of audio splicing from different models of acquisition devices. Our goal is to determine whether an audio track under analysis is pristine, or it has been manipulated by splicing one or multiple segments obtained from different device models. Moreover, if a recording is detected as spliced, we identify where the modification has been introduced in the temporal dimension. The proposed method is based on a Convolutional Neural Network (CNN) that extracts model-specific features from the audio recording. After extracting the features, we determine whether there has been a manipulation through a clustering algorithm. Finally, we identify the point where the modification has been introduced through a distance-measuring technique. The proposed method allows to detect and localize multiple splicing points within a recording.**

***Index Terms*—Audio forensics, audio authentication, microphone fingerprints, splicing detection, splicing localization**

I. Introduction

The rapid developments in technology and the increasingly widespread availability of advanced processing techniques have made the creation and distribution of multimedia content more accessible to everyone. It has also become much easier to create synthetic content or modify existing one making indistinguishable to everybody the difference between originals and copies anymore. This is especially true in the field of audio recordings, where the latest developed technologies can synthesize very realistic voices [1], clone voices with few training samples [2], or fill in small audio gaps controlling the synthesized words [3].

Despite these techniques offer a great potential in many fields, they can also be used for malicious purpose (e.g., creating fake speech for fake news spreading, defamation, etc.). If such a forged audio recording is shared online or is presented as evidence in a court of law, it becomes crucial to determine if it presents traces of manipulation or not. For this reason, the forensic community is pushing toward the development of forensic detectors tailored to identify advanced audio editing techniques [4]–[10].

Despite these new possibilities in audio forgery, classical and simple tampering approaches still prove valid and dangerous. This is the case of audio splicing, i.e., to create an audio recording as a composition of multiple concatenated audio pieces obtained from one or more audio tracks. Creating a spliced audio track is as easy as to perform some copy-paste operations. However, if applied to speech, the results can be worrisome and detrimental. For instance, think at how easy it is to twist the meaning of a speech by simply adding a negation (e.g., a spoken “not”) at some point of the recording. For this reason, we believe that it is necessary not to overlook the development of forensic detectors for more classical forgery operations.

In the light of this, in this paper we propose a novel technique for detection and localization of speech audio splicing. This means, given a speech audio recording under analysis, to determine if it has been manipulated through splicing (i.e., detection) and identify all time instants in which a splicing has possibly been operated (i.e., localization). The method we propose is based on the assumption that each different audio piece that is concatenated with the others contains specific characteristics that differentiate it from the other pieces. It is therefore possible to estimate if these characteristics of the audio recording change over time in order to detect and localize splicings.

This idea has been exploited in the literature by techniques focusing on estimating changes in specific parameters like noise level [11], bit depths [12], Electric Network Frequency (ENF) [13] or reverberation times [5]. In our work, we follow

This material is based on research sponsored 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. This work has also received funding by the European Union under the Horizon 2020 AI4Media project, Grant Agreement number 951911. This work was partially supported by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on ”Telecommunications of the Future” (PE00000001 - program ”RESTART”).

D. U. Leonzio, P. Bestagini, M. Marcon and S. Tubaro are with the Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, 20133 Milan, Italy (e-mail: danieleugo.leonzio@polimi.it, paolo.bestagini@polimi.it, marco.marcon@polimi.it, stefano.tubaro@polimi.it).

L. Cuccovillo and P. Aichroth are with the Fraunhofer Institute for Digital Media Technology, 98693 Ilmenau, Germany (e-mail: luca.cuccovillo@idmt.fraunhofer.de, patrick.aichroth@idmt.fraunhofer.de)

Manuscript received Month XX, 20XX; revised Month XX, 20XX.

the same principle as [5], but we exploit traces left by the recording device (i.e., smartphone model in our case). This means that we detect and localize splicing whenever the spliced audio pieces have been recorded with different acquisition pipelines belonging to different models of acquisition device. The reason behind this choice is twofold:

- Due to the widespread use of social platforms and messaging apps, it is nowadays easy to collect several speech tracks from the same person obtained from multiple recording sessions characterized by different devices (e.g., interviews from different news channels, personal social network pages, etc.). This makes multi-device splicing more and more popular, hence the need to deal with it from a forensic point of view.
- The literature on detecting and characterizing audio recording devices is rich and deep. This provides us with well-established methods to extract device traces, which increase the robustness of our splicing detection and localization technique.

Concerning the last statement, several methods that investigate the task of microphone identification have been proposed in the literature. There are methods based on some audio features such as [14]. This uses descriptors extracted from the audio recording and then cluster them with K-Means. Alternatively, in [15] the authors use Mel Frequency Cepstrum coefficients (MFCCs) and a Support Vector Machine (SVM) to identify different microphones. More advanced methods are based on Convolutional Neural Networks (CNNs) and extract some features directly from the audio recording. It is worth noting that CNN-based techniques are nowadays considered the most accurate methods in the literature to solve audio device classification problems. An example in this direction is [16] in which the authors adopt a CNN to extract the microphone characteristics from the Short-Time Fourier Transform (STFT) of the recorded signal.

Despite microphone identification is a topic deeply studied and different methodologies have been proposed, the task of determining the location of an editing point by exploiting microphone characteristics has not been thoroughly investigated. To the best of our knowledge, the only work that investigates the use of microphone cues to detect spliced traces is [17]. However, this method works under strong assumptions, i.e., the splicing point position must be known a-priori and the technique verifies it.

The method we propose for speech splicing detection and localization works as follows. Given a recording under analysis, we first adapt a well-established CNN-based method [16] to extract audio embeddings relative to different models of acquisition device from different time windows. We then apply a clustering step to possibly detect inconsistencies among the embeddings, which is sign that traces of multiple devices are present. If splicing is detected, we localize the splicing point by measuring embedding distances, and refine detection results. Finally, we perform additional iterations of the technique until all splicing points have been detected. The method is tested on a dataset built on purpose starting from the MOBIPHONE database [18]. Results show a 96% balanced accuracy on detection, and an average error on splicing localization of 0.012 second.

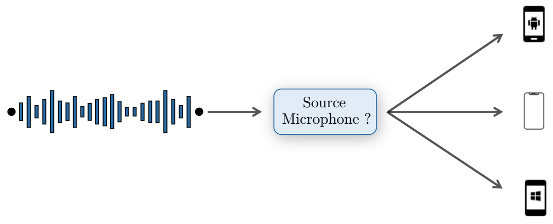


Fig. 1. Schematic interpretation of the device attribution problem. Given an audio recording, the goal is to detect which device has been used to acquire it.

To summarize, the contributions of this paper are the following:

- We propose a splicing detection and localization algorithm for audio speech recordings, which is a task overlooked in the literature.
- This work is the first to explore the problem of splicing localization making use of traces left by different models of acquisition devices.
- This is one of the first attempts in detecting and localizing multiple splicing points, rather than stopping at just one splicing.
- We consider the challenging situation in which our analysis window may not be aligned with the actual splicing point, as in a realistic situation.

The paper is organized as follow. Section II reviews the state of the art on audio device attribution and splicing detection and localization techniques. Section III reports the formal description of the problem addressed in this work. Section IV describes the main steps of the proposed technique when applied to detect and localize a single splicing. This provides the reader information about the proposed pipeline. Section V highlights the advanced aspects of the proposed method in order to better deal with audio windowing and extend the technique to the case of multiple splicings. Section VI presents the used datasets. Section VII describes how we select the feature extractor CNN and the parameters of our method. Section VIII presents the performance results on splicing detection and localization. Section IX shows a test of the presented method on a real case. Section X reports an experiment conducted to broaden the concept of traces left by an acquisition device. Finally, Section XI concludes the paper.

II. State of the Art and Background

Audio forensics is a branch of the broad field of forensic science. It refers to the acquisition, analysis, and interpretation of audio recordings as part of an official investigation [19]. Several audio forensic methods have been proposed in the literature for many different problems.

In this section we provide an overview of audio forensic methods related to the audio device attribution and splicing detection and localization problems faced in this work.

A. Audio Device Attribution

The goal of audio device attribution is to associate a given audio track to the device used for its acquisition (see Figure 1). Some works refer to this problem as microphone classification task, even if the classification is done on the complete acquisition pipeline and not only on the microphone. The initial works on audio source attribution used microphones as source devices. However, due to the increasing amount of available mobile devices, most recordings are nowadays obtained with smartphones. Therefore, also the source attribution domain evolved, and the most recent studies focus on associating an audio recording to the mobile phone used to record it [20].

In the literature, there are several methods to solve the problem of audio source attribution. One of the first method has been proposed by Kraetzer et al. in [14]. In this work, the authors suggest a set of audio steganalysis-based features to cluster, with K-means, or to predict, with Naive Bayes classifiers, both the microphone and the environment. Buchholz et al. [21] instead propose to use features extracted from Fourier coefficients from the near-silence segments to solve the microphone classification task. Both works have been improved by Kraetzer et al. in [22], where it is demonstrated that with the combination of statistical features and unweighted information fusion, it is possible to improve the classification accuracy. Another method has been proposed by Garcia-Romero and Wilson in [23]. In this paper, the classification is performed using Gaussian Support Vectors (GSVs), derived from the means of a trained Gaussian Mixture Model (GMM). A new study by Jiang and Leung [24] added a kernel-based transformation from the original GSV feature vector to another projected space, resulting in a better performance for the microphone recognition task. In [25] Panagakis and Kotropoulos propose random spectral features and classifiers based on the sparse representation. The same authors improved this study in [26] with labeled spectral features and SVM as a classifier.

Given that the most commonly used devices nowadays are mobile devices, many novel solutions have been proposed to classify the acquisition smartphone. The first study focusing on mobile phones is [15], in which Hanilci et al. use MFCC and a SVM to classify 14 different mobiles. In another study [27], they make a comparison among different acoustic features for mobile phone classification. The study concludes that in general, MFCCs perform better compared to other cepstral based features such as Linear Frequency Cepstrum coefficients (LFCCs), Bark Frequency Cepstrum coefficients (BFCCs) and Linear Prediction Cepstrum coefficients (LPCCs). Nevertheless, with mean and variance normalization, LPCCs provide the best classification results. Moreover, the authors observed that adding corresponding delta features (i.e., derivatives of order one and two) to the original cepstral features results in better performances.

Non-speech regions of the recorded audio were used in [28] for the cell-phone recognition task. Pandey et al. [20] use the estimate of power spectral density of the speech-free regions of the audio recording for source cell-phone classification tasks. Noisy part of the speech is utilized in [29] to extract MFCC feature vectors. In [18], intrinsic traces of cell-phone left on the recorded audio are captured by first extracting the MFCC feature vector at the frame level, then training a GMM. Finally, GSVs are taken as a template for each of the devices. Maximum classification accuracy of 97.6% is achieved on 21 cell-phones of seven different brands. GSVs have also been used for cell-phone verification [30].

Li et al. in [31] proposed an unsupervised method for cell-phone clustering. Deep auto-encoder networks are used to extract intrinsic signatures of a recording device. Spectral clustering is used to form a single cluster for the audio recordings of the same cell-phone. Concatenation of MFCC and inverted MFCC (IMFCC) feature vectors are used in [32] to capture device-specific traces. Additionally, Luo et al. [33] proved that the frequency response curve computed from the recorded audio could represent a robust device signature. A feature vector named BED (Band Energy Difference) has been derived to capture device-specific signatures. Qin et al. in [34] have explored the problem of cell-phone classification in the presence of five different types of noises. Cell-phone attribution in the presence of AWGN noise has been faced in [16], and [35]. However, these two systems use synthetic tones compared to real-world recorded audio signals. The method in [35] requires the suspected cell-phone to be available during the identification or authentication phase.

A channel attention mechanism based on subband awareness was used in [36] to recognise different cell-phone models. This approach utilizes a multi-stream network that leverages the distinctions between frequency bands, which contain crucial information for identifying the unique characteristics of built-in microphones. As a result, this method is capable of accurately recognizing cell phones from various manufacturers, as well as different models from the same manufacturer. Baldini et al. in [37] propose microphone identification method based on the combination of a CNN with spectral entropy features. A dimensionality reduction technique is proposed for spectral representation of audio signals captured from distinct microphones. The proposed method uses spectral features derived from Shannon entropy and Renyi entropy, in conjunction with the ReliefF algorithm. However, also this work as [16] use synthetic tones.

Despite multiple techniques have been proposed in the years, the use of Deep Learning (DL)-based methods seems to be the *de facto* standard in the modern literature. For this reason, in our work we start from the CNN proposed in [16] to extract device-related features.

B. Splicing Detection and Localization

Another problem of interest in audio forensics is the detection and localization of splicing forgeries. Detecting a splicing means to be able to tell if an audio recording is a composition or not. Localizing a splicing means to identify the point where a manipulated track has been altered.

To our knowledge, localization has not been still profoundly explored in the literature. Furthermore, the existing works on tampering localization are not based on microphone analysis.

An example is the study in [13] where the authors propose an audio forensic tool to assess audio authenticity. Their tool is based on the ENF. This is a signal embedded in audio files when the recording is taken with the device connected to an electrical outlet or when specific microphones are affected by a magnetic field. The authors analyze discontinuity in the phase of the power grid signal and then localize the splicing point at the discontinuities.

Another approach was used by Grigoras and Smith in [12], based on quantization level analysis to detect tampering in audio signals. The authors underline that PCM recording with low bit-depth, whenever processed by classic editing software, exhibit 16-bit of quantization instead of the original 8-bit or 12-bit only in correspondence of the tampering borders, or whenever external 16-bit content is spliced in the file.

In [38] Gartner et al., the method proposed to detect manipulation inside an audio signal is based on the analysis of discontinuities in the framing grid. Also here, when a discontinuity is found, tampering is detected and localized at the corresponding time instant.

Another different approach by Alan J. Cooper can be found in [39]. In this work, the author proposes a method based on the analysis of butt-spliced edits. Forgeries are often created using simple editing techniques such as butt-splicing. This can leave traces as discontinuity in the audio waveform. The method is time domain-based, uses a high pass filter on the audio data, and models the discontinuity at higher frequencies. The method then adopts a template to discover potential edits in the filtered signal.

Some other approaches detect splicing by analyzing noise levels in the audio signal. For instance, Pan et al. compute global and local noise levels of audio data and identify abnormal changes in the noise level [11]. Meng et al. compute local noise levels with respect to each syllable in the speech signal [40]. Yan et al. [41] address the problem of composite audio with signals from different sources but with similar or equal Signal to Noise Ratio (SNR). This work locates splicing points from the variances of MFCCs.

Zhang et al. introduced a splicing detection technique in their study [42], which relies on detecting high-frequency singularities. Splicing operations often create singularities in the waveform, and the authors took advantage of this characteristic to develop a method that identifies high-frequency singularities in the wavelet transform. The method is based on the combination of a CNN plus a Long Short-Term Memory (LSTM), that has the objective to learn the sequence information.

Jadhav et al. [43] use the short-term Fourier transform of spliced audio signals directly as input to a CNN. They propose a solution based on a DL technique, and the method they propose is robust under added white Gaussian noise and dynamic range compression. However, this work does not consider the highly relevant case of splices from recordings of the same speaker.

A different process was used by Capoferri et al. [5]. They explore in this paper the use of reverberation cues as a feature to detect and localize the point of splicing in an audio recording. In fact, distinct recordings may be recorded in different environments that are typically characterized by different reverberation cues. This method can be explained in three steps. They first apply a time-frequency transform on the time domain signal, then estimate the reverberation time in the free decay region, and finally, analyze the estimated reverberation times to determine the splicing location.

The authors got successful results, but the approach presented some intrinsic problems that affected the localization results. The main problem is that the reverberation time can be estimated only in the free decay region, so they could not localize the correct time instant of the splicing.

A new approach to solve both the detection and the localization was proposed in [9]. In this work the authors used a transformer architecture to solve this task. They simulated various types of post processing operations that can hide splicing, and trained a seq2seq architecture for detect and localize the splicing points. The results achieved are promising but also in this case there are intrinsic problems connected to the model proposed. The first one is on the vocabulary used to train the transformer that limits the localization precision. The second one is that this kind of architecture needs a large number of samples to be trained properly and achieve good results.

To the best of our knowledge, the possibility of exploiting device traces to assess the integrity of an audio recording and detect splicing has been seldom studied in the literature. A study in this direction can be found in [17]. In this work, the authors proposed a method of audio tampering based on the microphone classification but without providing any algorithm for localization. The authors assumed that the location of the splicing borders was known beforehand to simulate a setup similar to the one often asked in court cases. This setup is, of course, not applicable if the splicing location is unknown, which is often the case and that we are going to address in the following pages. The microphone classification algorithm is based on the work [44], but in this case, the channel models the frequency response of the microphone instead of the environment. On top of that, they build the tampering algorithm that relies on a SVM with a Radial Basis Function (RBF) kernel. They test the method on different encoding types and bitrates, reaching an overall accuracy of 95% on detection.

III. Problem Formulation

In this work we focus on the problem of speech audio splicing detection and localization. This is a problem of interest because it is reasonably easy to create fake speech audio recordings and completely change their original meaning by simply cutting and pasting a few words. This can lead to serious problem if we think, e.g., to a politician speech.

For a splicing operation to be effective from a semantic point of view, it is necessary that the attacker has a large enough library of speeches from the person of interest, in order to properly select words and sentences to be combined. In this context, it is possible that the audio tracks used for the splicing were recorded using different models of acquisition devices. Therefore, we can expect that a spliced recording exhibits traces from multiple recording devices. Our technique exploits these traces to detect and localize splicing.

Formally, let us take into account M audio pieces (i.e., portions of audio recordings) $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_M$ acquired with the same sampling frequency. Each piece $\mathbf{x}_m$ is a vector of N_m samples, which depends on the length of the single recording. Moreover, each piece $\mathbf{x}_m$ has an associated label l_m, which indicates the model of the used recording device. A spliced recording $\mathbf{x}_{\text{spliced}}$ can be built by concatenating $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_M$ as

$$\mathbf{x}_{\text{spliced}} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_M]. \tag{1}$$

If all M pieces have been acquired with the same model (i.e., $l_m = l_{m+1}$ for $m \in [1, ..., M-1]$), we cannot expect any change in device footprints over time. However, if at least two of the M slices belong to different classes (i.e., $\exists\ m : l_m \neq l_{m+1}$), then the device characteristics of $\mathbf{x}_{\text{spliced}}$ should change at the different devices concatenation samples.

Given a recording $\mathbf{x}$ with a length of N samples, we can associate to it a label $c \in [0, 1]$, where 0 means that the recording is pristine, whereas 1 means that the recording results from a splicing operation. In our device-based context this translates to

$$c = \begin{cases} 1, & \text{if } \exists\ m : l_m \neq l_{m+1}, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Detecting a splicing means to be able to compute a label $\hat{c}$ that is an estimate of c.

If we detect tampering in the audio recording (i.e., $\hat{c} = 1$), we also estimate the splicing points, i.e., an approximation of the sample indexes where the concatenations took place. This means to compute all sample positions $\hat{n}_m$, estimates of the samples n_m corresponding to a change in device model.

Formally, the m-th splicing point is defined as

$$n_m = \sum_{i \leq m} N_i. \tag{3}$$

Localizing the splicing points means estimating the set $\mathcal{N}$ of samples that represent the change point from one model to another defined as

$$\mathcal{N} = \{n_m : l_m \neq l_{m+1},\ m \in [1, ..., M-1]\}. \tag{4}$$

Figure 2 shows a visual representation of a spliced audio waveform. The recording $\mathbf{x}$ is composed by four pieces $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$ and $\mathbf{x}_4$. The color of each $\mathbf{x}_m$ piece denotes its class in terms of device model (i.e., l_m). In red we highlight the n_m belonging to the set $\mathcal{N}$. For instance, we do consider n_2 as a splicing point, as the device used to acquire $\mathbf{x}_2$ and $\mathbf{x}_3$ is the same. However, we can localize n_1 and n_3.

IV. Proposed Method (Main Pipeline)

In this section we provide all the details behind the main pipeline of the proposed method for audio splicing detection and localization based on traces left by the acquisition device model. In order to better focus on the overall pipeline and clearly explain the role of each block, in this section we consider the case of a single splicing. We leave to the next section the discussion on how to improve the method to deal with multiple splicing and to exploit a better windowing approach.

The idea behind our method is to use an audio device classification algorithm to extract significant audio descriptors in time, and explore them to build an identification and localization algorithm that searches for possible inconsistencies.

With reference to Figure 3, given an audio recording under analysis, we first pre-process it to obtain a time-frequency representation. Then, we use a modified version of a CNN architecture originally devised for microphone identification [16] to extract a time-series of feature vectors. These vectors, which supposedly contain the intrinsic characteristics of the input microphone, are used as input to a clustering technique to decide if the analyzed recording is a composition of different device models or not. If we detect a tampering in the recording, we apply a localization procedure to localize a candidate splicing cut point. Localization information is used to further refine detection results.

In the following we better detail each block of the pipeline.

A. Time-Frequency Transform

The first block of our pipeline is the time-frequency transformation, which is necessary to prepare the input data for further feature extraction.

Formally, let us consider we have an audio recording $\mathbf{x}$. The first operation to apply to the signal is windowing. The $\mathbf{x}$ signal is divided into W non-overlapping windows of L samples, with the w-th window defined as $\mathbf{x}^w$. The output of the pre-processing step is a matrix $\mathbf{X}_i$, which contains the log magnitude of the STFT of each w-th window defined as

$$\mathbf{X}^w = \log(|\text{STFT}(\mathbf{x}^w)|), \tag{5}$$

with $w \in [1, ..., W]$ denoting the window index.

B. Features Extraction

The second step of the pipeline consists in extracting a series of feature vectors by means of a CNN. This CNN is based on the work presented in [16], in which the authors illustrate a possible approach to solve a smartphone classification problem. The results they achieved were very successful, thus motivating us in using this network as a feature extractor for our algorithm. In particular, we train this CNN to solve a smartphone classification task as shown in [16], but we use the features extracted at an intermediate layer of the network as an embedding characterizing the acquisition device. Note that different feature extractors than [16] can in principle be used. The better the feature extractor, the better the splicing detection and localization capabilities.

The network architecture is shown in Figure 4. It is composed by a series of convolutional layers with ReLu activation interleaved by MaxPooling layers and batch normalization. After these blocks, two dense layers reduce the dimensionality

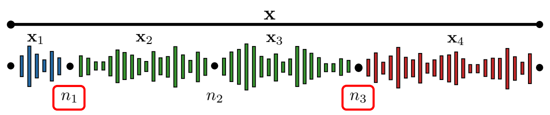


Fig. 2. Splicing concatenation example with four audio pieces and three different classes. Each color represents a recording device model. Splicing points that we consider for localization are highlighted in red.

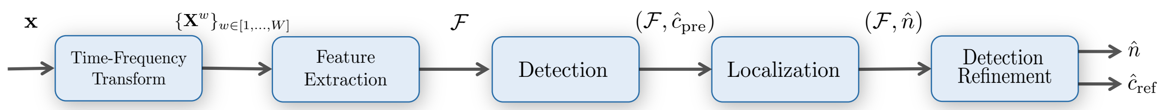


Fig. 3. Proposed method pipeline for solving one splicing point detection and localization.

of the data up to the number of considered device classes. The first convolutional layer has a kernel size of $(24, 24)$, stride equal to $(2, 2)$ and the padding option is set to "same". The second convolutional layer has a kernel size of $(4, 4)$ with same stride and padding used before. The last convolutional layer has a kernel size of $(1, 1)$. In the first Dense layer we add also a kernel regularizer, in particular an L2 regularizer with a penalty of 0.01.

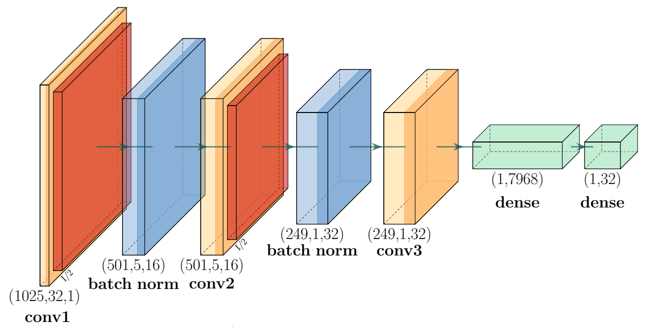


Fig. 4. CNN proposed in [16] used for feature extraction with an input of size 1025×32. The numbers are referred to the input size of each layer.

We train the CNN as multi-class classifier for microphone identification using sparse categorical cross-entropy as loss measure as shown in [16]. In doing so, the last layer of the network can be considered a one-hot encoding of the microphone labels. After training, we extract model-related features from the output of the first dense layer as commonly done in the literature. Indeed, this provides us with a higher dimensional feature vector with respect to the number of classes used in training, and does not depend on the number of classes.

Formally, we can say that when the CNN is fed with the input $\mathbf{X}^w$, it returns as output the model label estimate

$$\hat{l}^w = \arg\max \text{CNN}_{\text{out}}(\mathbf{X}^w), \tag{6}$$

and the feature vector of interest as

$$\mathbf{f}^w = \text{CNN}_{\text{feat}}(\mathbf{X}^w). \tag{7}$$

The feature vector $\mathbf{f}^w$ is composed by F elements. So for each spectrogram $\mathbf{X}^w$ we extract a feature vector $\mathbf{f}^w$ having a fixed length F. The set of feature vectors extracted for all the W windows is defined as

$$\mathcal{F} = \{\mathbf{f}^w\}_{w\in[1,\dots,W]} \tag{8}$$

This is used in the next step.

C. Detection

The next step is to detect if the audio recording has been tampered with by means of splicing operations. To do this, we apply a clustering technique to the set of features vectors $\mathbf{f}^w$ extracted by the CNN from each window of the input audio recording under analysis.

The idea is that a pristine recording is composed of windows that all share very similar device model information, hence feature vectors. Conversely, if an audio recording is a composition from multiple device models, the feature vectors extracted in time should change at some point.

In our method, we propose to use a procedure based on the K-Means algorithm [45]. We use the K-Means to divide the extracted feature vectors into two groups. If the two groups are far apart enough we can say that the initial recording was manipulated, otherwise we can conclude that no manipulation is detected. The detection pipeline can be summarized in the following steps:

- Apply the K-Means clustering algorithm.
- Compute the centroids of the corresponding clusters.
- Compute the distance between cluster centroids.
- Label the input recording as being tampered or not according to this distance.

In the first step, we apply a K-Means algorithm to the feature vectors $\mathbf{f}^w$. From the K-Means we obtain as many clusters as we asked for (two in our case). For each cluster we compute the centroid. Let us consider $\mathcal{F}_k$ as the set of all features belonging to the cluster number k. The k-th cluster centroid is computed as

$$\mu_k = \frac{1}{|\mathcal{F}_k|} \sum_{i:\mathbf{f}^w \in \mathcal{F}_k} \mathbf{f}^w, \tag{9}$$

where $|\cdot|$ indicates the cardinality of a set, and the sum is performed element-wise on each element of the vectors $\mathbf{f}^w$ so that μ_k shares the same dimensionality of a feature vector $\mathbf{f}^w$.

In the second step we compute the distance between the found centroids. In our study, we empirically decided from preliminary experiments to use the Euclidean distance as a distance measure. The distance between two centroids is defined as

$$d = \|\mu_1 - \mu_2\|_2, \tag{10}$$

with the $\|\cdot\|_2$ operator computing the ℓ^2 norm.

The final decision is taken based on a threshold mechanism. We set a threshold, and if the distance computed between the two centroids is higher than the threshold, we state that the recording has been modified, as we can see features belonging to more than one model.

Formally, we assign the preliminary tampering class label as

$$\hat{c}_{\text{pre}} = \begin{cases} 0 \text{ if } d \leq \gamma_{\text{pre}}, \\ 1 \text{ if } d > \gamma_{\text{pre}}, \end{cases} \tag{11}$$

where γ_{pre} is tuned based on a validation set of data.

D. Localization

If the analyzed signal is labeled as fake, i.e., formed by different model recordings as in Equation (1), the localization algorithm is applied to find the samples in which there is the change of model. Now we illustrate the proposed localization algorithm, in the basic scenario of one splicing point.

The main idea is that we can compute the distance between feature vectors belonging to neighboring windows from the audio recording. Distances should be small as long as we analyze windows recorded with the same model. Conversely, if the model change at some point, we should observe a high distance. The time instant in which we observe a high feature distance indicates the splicing point location.

Formally, let us consider an input recording $\mathbf{x}_{\text{spliced}}$ defined as in Equation (1) (i.e., by concatenating M sources of different lengths). We can compute the cosine distance $\mathbf{y}[w]$ between consecutive features extracted from the w-th and $w+1$-th window of the signals $\mathbf{x}_{\text{spliced}}$ under analysis as

$$\mathbf{y}[w] = 1 - \frac{\mathbf{f}^w \cdot \mathbf{f}^{w+1}}{||\mathbf{f}^w|| \cdot ||\mathbf{f}^{w+1}||}. \tag{12}$$

The splicing point is identified in the position in which $\mathbf{y}[w]$ shows its maximum. Formally,

$$\hat{w} = \arg\max_{w} \mathbf{y}[w] \tag{13}$$

corresponds to the detected splicing window index, that can be mapped into an audio sample index as

$$\hat{n} = \hat{w}L + 1, \tag{14}$$

where L is the length of a window in sample.

Figure 5 shows an example of $\mathbf{y}[w]$ where a clear maximum can be detected.

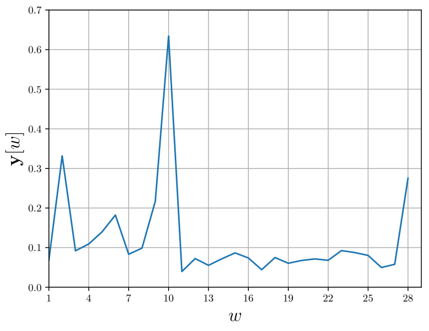


Fig. 5. Example of $\mathbf{y}[w]$ computed with Equation (12). The maximum obtained for $w = 10$ indicates a splicing point.

E. Detection Refinement

After estimating the splicing point, we perform a refinement on the detection step based on the value of the maximum associated to the splicing point. This prediction is based on a threshold mechanism. We set a threshold and if the maximum value is below the threshold we conclude that the audio recording is pristine and was wrongly detected as spliced by K-means. Formally,

$$\hat{c}_{\text{ref}} = \begin{cases} 0 \text{ if } \max \mathbf{y}[w] \leq \gamma_{\text{ref}}, \\ 1 \text{ if } \max \mathbf{y}[w] > \gamma_{\text{ref}}, \end{cases} \tag{15}$$

where γ_{ref} is a selected threshold tuned empirically on a set of validation data.

In Section III we defined the goal of the detection phase as the computation of $\hat{c}$. After refinement, we set $\hat{c} = \hat{c}_{\text{ref}}$.

V. Proposed Method (Enhanced)

In the previous section, we described our general proposed pipeline in a simplified case. We considered the presence of a single splicing, and we did not take into account the fact that the real splicing point may fall within a window of signal, rather than between two windows.

In a realistic scenario, we have to assume that:

- More than one splicing point in an audio recording can be present.
- The splicing point is not always in between two windows.

For these reasons, we created an advanced pipeline, adding some blocks to the basic one. This section explains which blocks we added and how they make the method applicable to realistic scenarios.

A. Multi-shift

The localization pipeline described in Section IV-D works with the assumption that the splicing point occurs in the correspondence of the window change point. This means that if we use the localization pipeline described in Section IV-D we suffer from two major issues:

- We feed the feature extractor a window containing samples coming from two device models, which is a situation for which the CNN has not been trained.
- We introduce a localization error since we estimate the splicing point in the window change point.

In Figure 6 we show a visual representation of the first major issue described above. The figure depicts a splicing between two recordings $\mathbf{x}_1$ and $\mathbf{x}_2$. It also reports two scenarios (case (a) and case (b)) of signal windowing. In case (a), the splicing point falls within the analyzed window $\mathbf{x}^w$, so the window $\mathbf{x}^w$ contains samples from two different models. Instead, case (b) represents the ideal case in which the splicing point falls between two consecutive windows. To solve these problems, we propose to enhance the proposed method with an analysis technique called multi-shift. This procedure analyzes the audio recording multiple times, each time starting from a shifted starting point. We consider each shift as a single audio recording and follow the pipeline explained in Section IV.

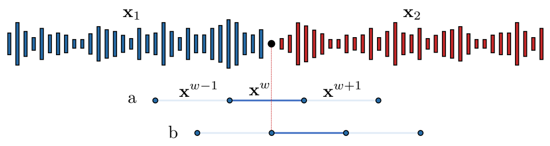


Fig. 6. Visual representation of the issue present in the localization pipeline described in Section IV-D. In case (a), the window $\mathbf{x}^w$ contains samples from multiple models. In case (b), the splicing point falls between the windows $\mathbf{x}^w$ and $\mathbf{x}^{w-1}$.

The main idea is that we should be able to find one good shift that ensures that the splicing point is in a good position with respect to the window borders. At the end, we merge the results from the different runs to achieve a more precise splicing point localization prediction.

Let us assume that the feature extraction CNN works correctly given that the input $\mathbf{X}^w$ is obtained from a L samples window containing the vast majority of samples from a single model. Only a fraction of ρ samples (i.e., ρL samples) can belong to a different model. We run the proposed pipeline Q times. Each time, we run it on a version of the input $\mathbf{x}$ to which we remove an amount of ρL samples at the beginning with respect to the previous run (see Figure 7). Formally, the q-th run is performed on the signal $\mathbf{x}_q$ corresponding to $\mathbf{x}$ deprived by its first $(q-1)\rho L$ samples. Q is set to a value so that we skip at most one window of L samples in the last run (i.e., $(Q-1)\rho L < L \leq Q\rho L$).

According to this procedure, there exist one $\mathbf{x}_q$ for which the splicing point falls within a window with a margin of at most ρL samples from the window border. The proposed detection and localization method should then correctly work on this selected run, while showing more uncertain results on the other runs. In practice, on the correct run, $\mathbf{y}[w]$ will exhibit the highest maximum.

With this idea in mind, we can estimate the index of the best run by computing

$$\hat{q} = \arg\max_{q} \mathbf{y}_q, \tag{16}$$

where $\mathbf{y}_q$ corresponds to $\mathbf{y}$ computed on the q-th run. We then compute estimate the index of the window containing the splicing as

$$\hat{w} = \arg\max_{w} \mathbf{y}_{\hat{q}}[w]. \tag{17}$$

Finally, we convert the window index into a sample index by correcting for the shift as

$$\hat{n} = (\hat{w} + \hat{q}\rho)L + 1. \tag{18}$$

This is the estimated splicing point in samples.

In Figure 8 there is an example of the distances $\mathbf{y}[w]$, after applying the multi-shift operation with $Q = 5$.

B. Multiple Splicings

The other problem in the basic pipeline is detecting more than one splicing point. In fact, if we use the pipeline described in Section IV we can detect only one splicing point, but we want to detect and localize all the splicing points inside the analyzed audio recording.

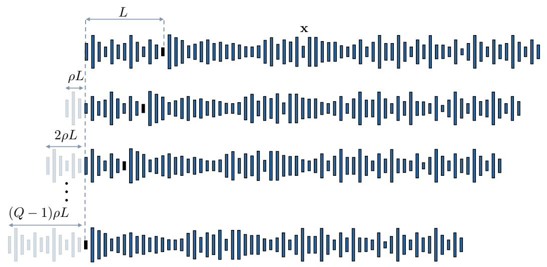


Fig. 7. Visual description of how the multi-shift operation is performed. Each row represent one of the Q runs. For each run we remove the first ρL samples before starting the windowing process with L sample long windows.

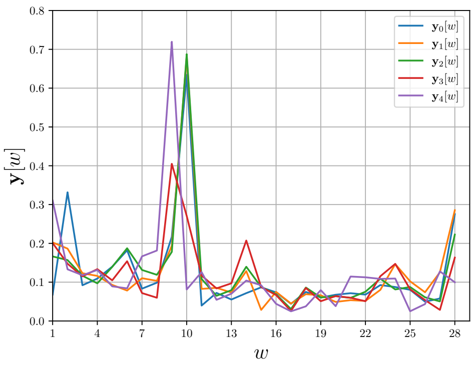


Fig. 8. Example of $\mathbf{y}_q[w]$ computed Q times using the multi-shift technique. Each $\mathbf{y}_q[w]$ shows a maximum around $w = 9$ or $w = 10$. We estimate $\hat{w} = 9$ as $\mathbf{y}_4[w]$ has the highest peak.

We solve this issue by making the algorithm iterative. After detecting and localizing the first splicing point, we split the audio recording into two slices by cutting the recording at the predicted splicing point. We then analyze each slice as a single audio recording, repeating the basic pipeline.

The algorithm keeps running iteratively until one of the following stopping conditions is reached:

- No more splicing are detected (i.e., $\hat{c}_{\text{ref}} = 0$)
- The minimum length of the audio file for proper analysis is reached (the length must be greater than $2L$ to have at least two windows).

Figure 9 reports a visual example of the iterative process.

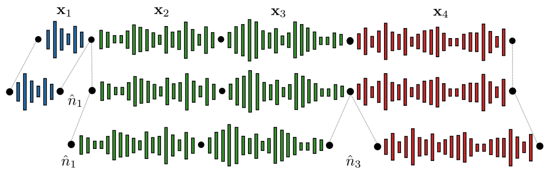


Fig. 9. Example of the iterative process with three classes and two splicing points.

VI. Dataset

In this section we explain the dataset we used in our study.

The dataset we decided to use is the MOBIPHONE dataset [18]. This is a collection of 4800 utterances recorded with 21 mobile phone models from seven different brands. For each phone there are tracks obtained recording 24 speakers from TIMIT [46], 12 males and 12 females. For each speaker we have 10 sentences. The TIMIT is a corpus designed to provide speech data for acoustic-phonetic studies and for the development and evaluation of automatic speech recognition systems. We followed the same approach explained in [47] and [48] in which they excluded the model "Samsung s5830i" due to the short duration of its recording.

In Table I, we report the complete list of brands and cell phone models used in this study. There is also the class label tag associated to each mobile phone class.

To be more realistic, we decided to adopt an open-set approach, so not all the models of the dataset have been used for network training. This is essential to check if we can detect and localize splicing also when models different from the ones seen by the feature extractor are used.

We divided the dataset in the following way:

- Training: models from class 0 to 15, speakers from 1 to 16.
- Test: models from class 0 to 15, speakers from 17 to 24.
- Open-set: models from class 16 to 19, speakers from 17 to 24.

From the test set and open set audio recordings we have built the splicing dataset. The splicing dataset has been obtained by creating all possible pairs and triplets of recordings (with and without smartphone model change), always considering the same speaker within the pair or triplet. This is important so to ensure that we only detect device model changes, and we do not detect splicing due to different speakers. In this way we were able to get 2560 pairs and 51200 triplets, that we used to test the method as explained above. We will refer to this dataset as the Clean version.

In addition to this, to check the robustness of our method we generated additional versions of the dataset by adding noise or compressing it, as often done in literature [16] [49]. For the noisy datasets we decided to use an Additive White Gaussian Noise (AWGN) with different SNRs. As SNRs we decided to use 15dB, 20dB, 30dB, 40dB. On the other hand for the compression datasets we decided to use the Adavnced Audio Coding (AAC) compression pipeline (a standard for smartphones), with different bitrates. The bitrates adopted are 96 kbit/s and 128 kbit/s.

TABLE I
LIST OF MOBIPHONE MODELS

Class	Brand and Model	Class	Brand and Model
0	Apple iPhone 5	10	Samsung Galaxy GT-I9100 s2
1	HTC desire c	11	Samsung Galaxy Nexus S
2	LG GS290	12	Samsung GT-N7100 (galaxy note2)
3	LG L3	13	Sony Ericson c510i
4	LG Optimus L9	14	Sony Ericson c902
5	Nokia 5530	15	Vodafone joy 845
6	Nokia C5	16	HTC Sensation xe
7	Samsung e1230	17	Nokia N70
8	Samsung E2121B	18	LG Optimus L5
9	Samsung E2600	19	Samsung GT-I8190 mini

VII. Parameters selection

In this section we report details about the parameters we chose in our pipeline, from the CNN feature extractor to the different thresholds.

A. Feature Extraction CNN

In order to choose the best smartphone classification networks, we did some experiment on different architectures and combinations. We compared two networks, the first one taken by Baldini et al. [16] and the second one by Zeighidour et al. [50]. The second CNN is slightly different from the first one as it has more layers and a higher number of hyper-parameters, which we set as explained in [50]. The first one is the network we decided to use as it showed the best results for our task.

The training was done on the training set, composed as explained before by 16 speakers per class. The optimizer used is the RMSPprop with a learning rate set to 0.0001 [51]. The number of epochs was set to 100, unless the training was automatically stopped in case of overfitting on small validation set extracted from the training one. We set a Dropout equal to 0.3. The loss function to minimize is the sparse categorical cross-entropy function. As explained in Section IV-B the network has been trained to solve a smartphone classification task.

Notice that this is the only training step of our entire pipeline. In this step we do not need to use spliced data. We only work with original recordings from different models.

Figure 10 shows the confusion matrix that captures the attribution performance of the network proposed in [16] on the test set. These results show that with this network we are able to get successful results in terms of classification, with a final accuracy value of 95%.

This motivates us in using this network as feature extractor.

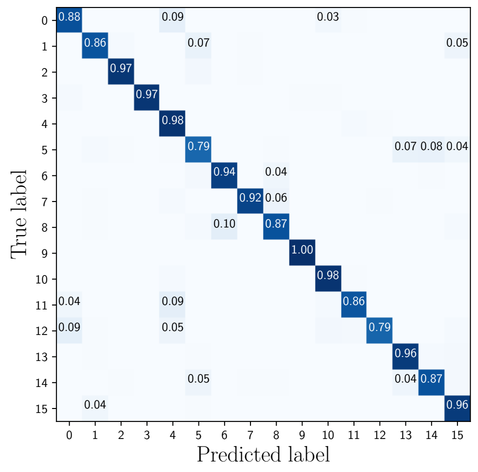


Fig. 10. Confusion matrix obtained with the CNN proposed in [16] on the test set reported in Table I (i.e., classes from 0 to 15).

B. ρ selection

The ρ parameter is used to select the Q value, so how many time we scan the audio signal starting each time ρL samples after the previous one, as described in Section V-A. The ρ value represents the portion of a different class that can be present in a window in order to get the desired accuracy in prediction of the original class.

We tested the CNN on the classification task for 10 different values of ρ, and then we analyzed how the accuracy changed with respect to ρ values. In other words, after training the CNN, we built 10 different test sets (one per ρ value). Given a certain ρ, we tested the CNN in classifying a window of samples composed by ρL samples from one device and $(\rho - 1)L$ samples from another device, with the goal of recognizing the second device.

Figure 11 shows the achieved results, using random combination of models in the test set reported in Table I. The curve shows that for $\rho = 0$ (i.e., all samples in the analysis window come from a single device), the CNN works with an accuracy higher than 90% (confirming the results from Figure 10). As ρ increases (i.e., more samples come from a different device), the CNN performance worsen as expected (as the CNN analyses a mix of samples from multiple devices). When $\rho = 1$, all samples in the window come from a different device, and the CNN always fails as expected.

From these results we decided to use a value of ρ equal to 0.2. In this way, we are able to have an overlap of 80%, an accuracy above 70% and in the same time limiting the iteration to 5.

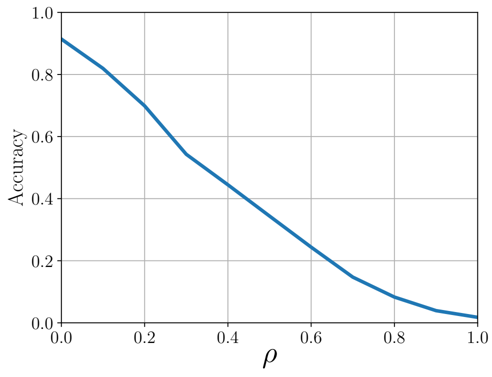


Fig. 11. Mean accuracy change with different ρ values.

C. Threshold for K-Means

In order to detect the presence of splicing, we threshold the quantity d computed as in Equation (10) representing the distance between features from different models. To do so, we computed the value d for multiple slices pairs (belonging or not to the same model), taken from the test and open set, and we evaluated the results by means of Receiver Operating Characteristic (ROC) curves and Area Under the Curve (AUC).

To perform this evaluation, we had to work with pairs of classes, since the ROC measures a binary classifier. So we built all the possible pairs among the various classes. Then we built the distance signal $\mathbf{y}[w]$, as explained in Equation (12).

Figure 12 reports the ROC curve we used to select the threshold γ_{pre}. From the ROC curve we extracted the threshold as the one that maximized the accuracy. The value we obtained for the threshold γ_{pre} used in Equation (11) is 0.2.

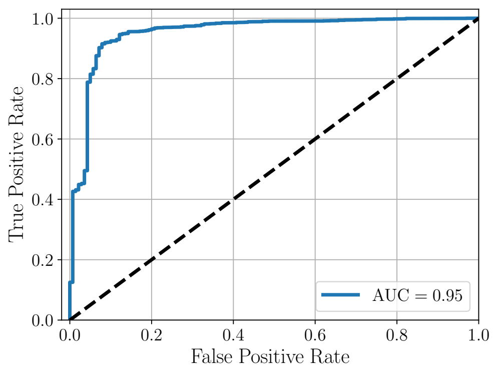


Fig. 12. ROC for K-Means threshold selection.

D. Threshold for detection refinement

We adopted the same approach explained in Section VII-C, to determine the right threshold for the detection refinement. We build the ROC curve based on the maximum value of each predicted splicing point (i.e., $\max \mathbf{y}[w]$), and the we found the best threshold that separates the true splicing point from the wrong ones.

In Figure 13 there is the final ROC curve we used to select the threshold γ_{ref}. From this ROC curve we obtained a value of γ_{ref} equals to 0.3.

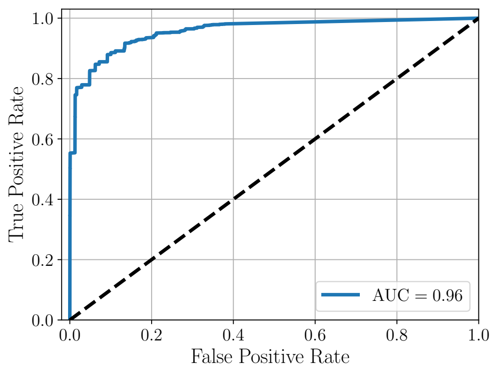


Fig. 13. ROC for max values threshold selection.

VIII. Performance

This section presents the results achieved by our splicing detection and localization method. The results are divided in three subsections: Detection, Localization and Iterative results.

In Section VIII-A and Section VIII-B we report the results for detection and localization, respectively. In both sections the results have been obtained testing the method on one splicing point, so two different models concatenated together (i.e., the dataset composed by track pairs). The results are organized in tables and the test were done on the clean pairs dataset, the noise dataset and the compressed one.

In Section VIII-C we report the results achieved with the iterative algorithm, so testing the method on multiple splicing points as in a realistic condition. These results are obtained considering the dataset composed by triplets of recording, in its clean form, the noisy ones, and the compressed ones.

A. Detection

After determining the optimal threshold and verifying that it was adequate for all the classes as explained in the previous section, we tested the clustering algorithm reported in Section IV-C.

1) Comparison Against Baselines: The first set of experiments is aimed to test our proposed method against some baselines.

As first baseline (i.e., Baseline I), we considered the approach explained in [5]. Instead of using the K-Means, we used the max value of the distance sequence to obtained the prediction label. The second baseline (i.e., Baseline II) is obtained using the method explained in [52]. This approach is based on computing the correlation between features and detect the splicing using a threshold on the minimum correlation value obtained by and audio recording.

Concerning our method, we show the performance in two situations: considering the preliminary estimation $\hat{c}_{\text{pre}}$ (i.e., Proposed); considering the refined result $\hat{c}_{\text{ref}}$ as reported in Section IV-E (i.e., Proposed Refinement). This could be interpreted as a sort of ablation study on the use of the multi-shift refinement.

Figure 14 shows the ROC curves obtained for the two baselines (Baseline I and Baseline II) and our method (Proposed) tested on the clean dataset (i.e., tracks not affected by noise or compression). The ROC curves have produced by thresholding the distance values d, i.e., the output of the detection before computing the value $\hat{c}_{pre}$.

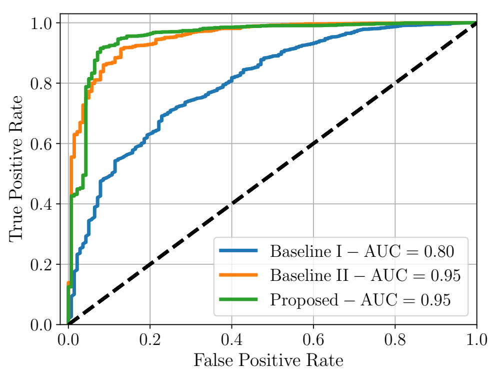


Fig. 14. ROC curves comparison among baselines and proposed method.

These results show that the proposed method achieves the same AUC of the best baseline. However, this is not the only metric of interest.

To better capture the behaviour of the method, Table II reports the outcome of the clustering algorithm on the clean dataset in terms of balanced accuracy when the best working point is selected. In practice, we consider as true positives all the spliced tracks that are correctly detected, and as true negatives all the tracks that are correctly detected as not containing any change.

These results show that the proposed preliminary method outperforms both baselines. The refinement step improves the accuracy even more, up to 96%.

In addition to these results we have tested also the method proposed by Jadhav et al. in [43]. This work is based on a CNN trained to detect the presence of a splicing point inside an analyzed window. With this pipeline tested on our dataset, we were able to achieve an overall accuracy of about 87%. So also in this case our method is the best performing one for the detection problem.

TABLE II
DETECTION RESULTS ON THE CLEAN DATASET OF TRACK PAIRS.

Method	Balanced Accuracy
Baseline I [5]	0.765
Baseline II [52]	0.859
Proposed	0.906
Proposed Refinement	0.960

2) Robustness: As additional experiment, we tested the behaviour of the proposed method on tracks corrupted by noise or compression.

The choice of additive noise is to test the method under more realistic working conditions in which attackers may decide to apply some additional operations after splicing to mask possible forensic traces. Indeed, adding noise to multimedia data can be seen as a very simple but often effective anti-forensic technique.

The choice of testing the methods against compressed audio recordings also goes in the direction of working in a more realistic case. It is indeed expected that audio recordings are seldom uncompressed. They are rather almost always compressed to either limit the space on the device onboard memory, or save bandwidth during transmission. It is also well known that compression may hinder forensic traces in multiple scenarios.

Table III reports the results obtained with the augmented datasets, created as explained in Section VI. In this case we do not report the baselines anymore, as they are always outperformed.

These results show that the proposed method is robust to both compression and additive noise, as only a few percentage points of accuracy are lost with respect to the clean dataset scenario. Moreover, the boost in performance obtained by using the refined technique are evident.

B. Localization

Once the tampering is detected we need to find the splicing point. To do this we used the localization algorithm explained in the Section IV-D.

To evaluate how well the localization works we set an error measure as the difference between the predicted point and the

TABLE III
DETECTION RESULTS ON THE NOISY AND COMPRESSED DATASET OF TRACK PAIRS.

Method	Dataset	Balanced Accuracy
Proposed	SNR 40dB	0.884
	SNR 30dB	0.879
	SNR 20dB	0.805
	SNR 15dB	0.742
	AAC 128 Kbit/s	0.886
	AAC 96 Kbit/s	0.872
Proposed Refined	SNR 40dB	0.952
	SNR 30dB	0.942
	SNR 20dB	0.879
	SNR 15dB	0.808
	AAC 128 Kbit/s	0.954
	AAC 96 Kbit/s	0.950

true splicing cut point. The error is expressed in seconds and is defined as:

$$e = \frac{(n - \hat{n})}{S}, \tag{19}$$

where n is the true splicing point, $\hat{n}$ is the predicted one and S is the sampling rate.

1) Comparison Against Baselines: As first experiment, we compare our method against a baseline on the clean dataset. Also in this case we used as baseline the method explained in [5]. Our method, in fact, can be seen as an advanced version of this baseline because they in the work [5] considered only one version of the signal, whereas we analyzed the signal Q-th times. In addition we compared our method with the pipeline proposed by Moussa et al. in [9]. In this work they propose a transformer architecture for audio splicing localization. The transformer is trained to detect splicing points with a minimum error of $0.5s$, so just from how the architecture is trained we can say that our method outperforms their pipeline. Moreover we tested their method on our dataset, and the average error is about $4.5s$. This is due, in our opinion, because the two architectures are trained to solve a similar problem but in two different cases. In fact the transformer architecture is not trained to detect the splicing points if the manipulation is only a change of the device used for the recording.

The results of the localization method are reported in Table IV, and for each method we reported both the mean and the standard deviation of the obtained error e. Results show that the proposed enhancement on the Q runs of the base localization step strongly improve the results.

TABLE IV
LOCALIZATION RESULTS ON THE CLEAN DATASET OF TRACK PAIRS.

Method	Error	
	Mean	Std
Baseline I [5]	0.540	± 4.556
Proposed	0.012	± 3.863

To better visualize the performance of the proposed method we show the histogram of this error measure in Figure 15. The horizontal axis represents the localization error e in seconds. The vertical axis represents the occurrence over the test set of clean pairs. Negative values show that the detected splicing point precedes the actual splicing.

From these results we can see that the proposed method hardly makes an error larger than 1 second in localizing a splicing when it is detected.

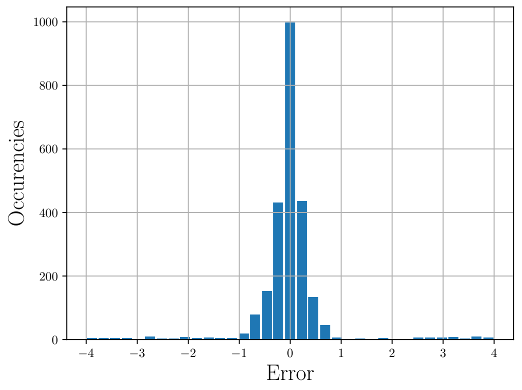


Fig. 15. Histogram of the localization error measure for the clean dataset.

2) Robustness: As we have done for the detection results, we report in Table V the results obtained with the augmented datasets to test the robustness of our method to editing operations like noise addition and compression.

TABLE V
LOCALIZATION RESULTS ON THE NOISY OR COMPRESSED DATASETS OF TRACK PAIRS.

Method	Dataset	Error	
		Mean	Std
Proposed	SNR 40dB	0.046	± 4.489
	SNR 30dB	0.196	± 5.650
	SNR 20dB	0.218	± 6.853
	SNR 15dB	0.416	± 7.515
	AAC 128 Kbit/s	0.048	± 3.876
	AAC 96 Kbit/s	0.006	± 3.789

C. Iterative results

This section presents the results obtained with the complete iterative pipeline.

We built the test data with triplets of models from Table I. As explained in Section VI we have created the dataset with all the possible triplets for each speaker, considering both test and open set models. This means that in an audio recording there can be zero, one or two splicing points, depending on the models used in the triplet.

As for the previous section the detection results are shown in terms of balanced accuracy, consider as true positive the true model changes and as true negative the true no changes. In this case, we achieved 92.4% for the preliminary detection, and 95% for the refined one.

For the localization we use the mean and the standard deviation of the error, computed as explained in Section VIII-B, considering only the correct detection. For the localization the mean error takes into account the average error on all splicing

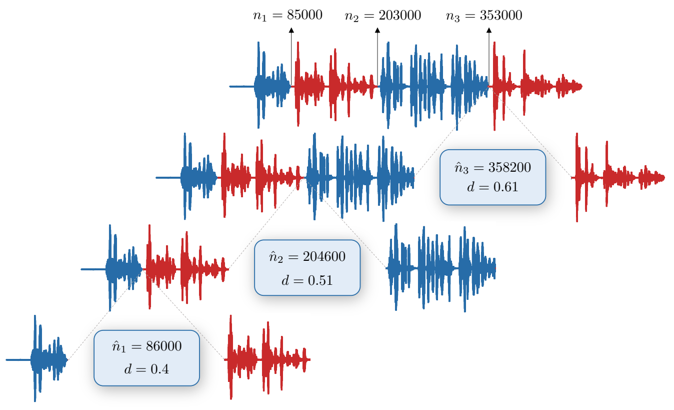


Fig. 16. Case study with 2 models (classes 0 and 19), and 3 splicing points. The colour of each waveform corresponds to the class model.

points per test track. We achieved an average localization error of 0.581 seconds, with a standard deviation of 2.769.

IX. Case study

In this section we show how the proposed pipeline performs on a realistic case. To do so, we consider a longer speech track with more than two splicing points to evaluate how the iteration process performs in a more challenging scenario. The considered case is more similar to real conditions, where we have to deal with more complex audio tracks that contain an unknown number of forgeries.

In this example we created a spliced audio recording using two models from the MOBIPHONE dataset, namely classes 0 and 19. We considered the same speaker for both classes, i.e., number 11. We merged the audio recordings of the selected smartphones to create a new recording with 3 splicing points.

To consider the most challenging scenario possible, we selected one device among the training set models, class 0, and the other among the open-set ones, class 19.

To better understand the example, we have reported all the iterative steps in the Figure 16. The first row of the Figure 16 shows the spliced audio recording we built. Each color of the audio waveform represents a model class and the splicing points correspond to the color changes.

Each row of the figure represents a step of the iteration process. The blue boxes report the summary values of each iteration. In particular, we report the true splicing point n in samples, the predicted one $\hat{n}$ in samples, and the distance value d at the predicted splicing point.

As described in Section V-B, we have split the audio recording into two parts at the predicted splicing point for each step after the splicing localization. Then we repeated the proposed method for each part until we reached one of the stop conditions explained in Section V-B.

As can be deduced from the reported results, the proposed method performs well also in a more complex scenario. We were able to detect all the splicing points present in the audio recording, with an average error about 0.1 seconds.

X. Beyond Acquisition Devices

In this paper, we have addressed the problem of splicing detection and localization considering the analysis of traces left by the acquisition device. However, the concept of *acquisition device traces* is very broad. These traces may derive from the microphone. Alternatively, they may be due to some specific onboard processing and equalization. In other cases, these traces may be left by compression.

In this section, we perform a preliminary experiment to investigate if the proposed method is capable of working on other kinds of traces that can be compared to those of the acquisition device. We take into account audio tracks obtained through synthetic speech generators. We consider tracks synthesized by different generation algorithms as if they were acquired with different devices. Indeed, we know from the literature that different synthetic speech algorithms leave different traces [53]. We aim to see whether our feature extractor is capable of capturing traces that distinguish real speech from synthetic one, as well as to tell different synthetic generation algorithms apart. Indeed, if this happens, it means that we could use the proposed splicing detection and localization algorithms also to spot if a speech is a composition of real and synthetic tracks, or tracks from multiple generators.

To do so, we present an evaluation of the proposed method on two distinct tasks: deepfake detection and attribution. To assess the generalization ability of our CNN, we conducted experiments on the ASVspoof 2019 dataset [53] using the model trained on the microphone classification task. This dataset includes various synthesis algorithms that can be treated as different microphones. Our evaluation involved two separate sub-tasks: first, detecting whether a track is bonafide or spoofed, i.e., deepfake detection; and second, identifying the specific algorithm used to generate the synthetic track, i.e., deepfake attribution. To perform these tasks, we utilized the CNN architecture shown in Figure 4, which was originally trained on the MOBIPHONE dataset. To analyze whether the extracted features can be used for both problems, detection and attribution, we used a Random Forest classifier trained on top of the features extracted with the MOBIPHONE pre-trained extractor CNN.

Figure 17 shows the confusion matrix for the detection problem, where class 0 stands for bonafide and class 1 represents the spoof class. It is possible to see that the pre-trained feature extractor works nicely in distinguishing real from synthetic speech.

Figure 18 reports the confusion matrix for the attribution problem, where class numbers are referred to the ASVspoof 2019 notation used to differentiate the different generators, where we consider ASVspoof 2019 voice conversion and text-to-speech algorithms. It is possible to see that the selected four synthesizers can be clearly separated. This is surprising given that the feature extractor was only trained on MOBIPHONE data, and a very simple Random Forest was used for classification purpose.

These preliminary experiments show the effectiveness of our proposed method in detecting and attributing deepfake audio,

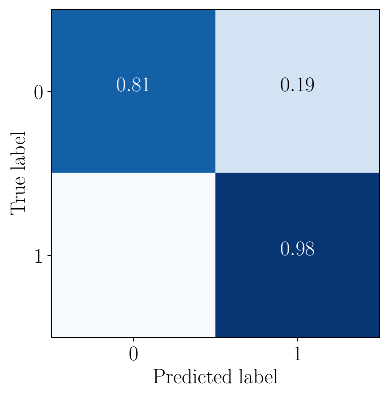


Fig. 17. Confusion matrix obtained for the deepfake detection problem with the CNN in Figure 4. Values below 4% are not reported.

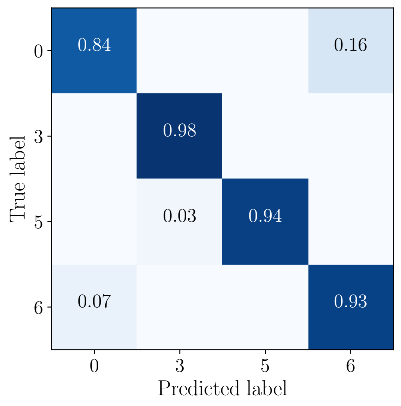


Fig. 18. Confusion matrix obtained for the deepfake attribution problem with the CNN in Figure 4. Values below 4% are not reported.

and showcase its ability to generalize across different tasks and datasets.

XI. Conclusion

This work proposes a new methodology for audio forensic analysis exploiting traces left on a recording by acquisition devices. Specifically, we have addressed two main tasks:

- Detecting if an audio recording comes from a single recording or it is a splicing generated concatenating slices from different models.
- Localizing the splicing points in order to separate the concatenated slices from the original one.

The devised methodology is based on a CNN able to extract suitable features from the audio recording, K-Means clustering algorithm to recognize the presence of traces from multiple models, and a distance measure to localize the splicing points. We also enhance the basic pipeline with a multi-shift operation to achieve a better precision in localization and an iterative process to find multiple splicing points.

The method has been evaluated on a dataset built on top of the MOBIPHONE one, that we used both for training the CNN and to test the entire algorithm.

The proposed approach has shown promising results both in detection and localization. In particular the pipeline described in our work achieves outperforming accuracy. The detection stage shows 96% accuracy, while the maximum localization error is about 0.012 second on clean recordings. This results highlights that the proposed method is a valid approach to solve both detection and localization tasks.

Given the achieved promising results, future work will be devoted to extend the method in two direction. On one hand, we would like to deepen the study on the applicability of the proposed technique to synthetically generated audio tracks given the preliminary promising results. Moreover, we would like to make the feature vector extractor capable of capturing traces left by audio editing techniques in addition to device ones. Finally, it could be interesting to investigate the scenario in which the recordings have been transmitted between smartphones or obtained through wiretapping.

References

[1] Y. W. et al., “Tacotron: Towards end-to-end speech synthesis,” in *INTERSPEECH*, 2017, pp. 4006–4010.
[2] S. Arik, J. Chen, K. Peng, W. Ping, and Y. Zhou, “Neural voice cloning with a few samples,” in *Advances in Neural Information Processing Systems (ANIPS)*, 2018, pp. 1–11.
[3] Z. Borsos, M. Sharifi, and M. Tagliasacchi, “Speechpainter: Text-conditioned speech inpainting,” *ArXiv*, 2022.
[4] E. A. AlBadawy, S. Lyu, and H. Farid, “Detecting ai-synthesized speech using bispectral analysis,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2019.
[5] D. Capoferri, C. Borrelli, P. Bestagini, F. Antonacci, A. Sarti, and S. Tubaro, “Speech audio splicing detection and localization exploiting reverberation cues,” in *IEEE International Workshop on Information Forensics and Security (WIFS)*, 2020.
[6] D. Cozzolino, M. Nießner, and L. Verdoliva, “Audio-visual person-of-interest deepfake detection,” *arXiv preprint arXiv:2204.03083*, 2022.
[7] D. Castán, M. H. Rahman, S. Bakst, C. Cobo-Kroenke, M. McLaren, M. Graciarena, and A. Lawson, “Speaker-targeted synthetic speech detection,” in *Odyssey 2022: The Speaker and Language Recognition Workshop*, 2022, pp. 62–69.
[8] B. Hosler, D. Salvi, A. Murray, F. Antonacci, P. Bestagini, S. Tubaro, and M. C. Stamm, “Do deepfakes feel emotions? A semantic approach to detecting deepfakes via emotional inconsistencies,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.
[9] D. Moussa, G. Hirsch, and C. Riess, “Towards Unconstrained Audio Splicing Detection and Localization with Neural Networks,” in *International Conference on Pattern Recognition*, 2022.
[10] S. Cui, E. Li, and X. Kang, “Autoregressive model based smoothing forensics of very short speech clips,” in *IEEE International Conference on Multimedia and Expo (ICME)*, 2020.
[11] X. Pan, X. Zhang, and S. Lyu, “Detecting splicing in digital audios using local noise level estimation,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012.
[12] C. Grigoras and J. M. Smith, “Quantization level analysis for forensic media authentication,” in *AES International Conference on Audio Forensics (ICAF)*, 2014, pp. 71–76.
[13] D. P. N. Rodríguez, J. A. Apolinário, and L. W. P. Biscainho, “Audio authenticity: Detecting enf discontinuity with high precision phase analysis,” *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 5, no. 3, pp. 534–543, 2010.
[14] C. Krätzer, A. Oermann, J. Dittmann, and A. Lang, “Digital audio forensics: a first practical evaluation on microphone and environment classification,” in *Workshop on Multimedia & Security (MM&Sec)*, 2007, pp. 63–74.
[15] C. Hanilçi, F. Ertas, T. Ertas, and Ö. Eskidere, “Recognition of brand and models of cell-phones from recorded speech signals,” *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 7, pp. 625–634, 2012.
[16] G. Baldini, I. Amerini, and C. Gentile, “Microphone identification using convolutional neural networks,” *IEEE Sensors Letters*, vol. 3, pp. 1–4, 2019.
[17] L. Cuccovillo, S. Mann, M. Tagliasacchi, and P. Aichroth, “Audio tampering detection via microphone classification,” in *IEEE International Workshop on Multimedia Signal Processing (MMSP)*, 2013.

[18] C. Kotropoulos and S. Samaras, "Mobile phone identification using recorded speech signals," in *International Conference on Digital Signal Processing (ICDSP)*, 2014.

[19] R. C. Maher, *Principles of Forensic Audio Analysis*, 1st ed. Springer, 2018.

[20] V. Verma and N. Khanna, "Speaker-independent source cell-phone identification for re-compressed and noisy audio recordings," *Multimedia Tools and Applications*, vol. 80, pp. 25 581–23 603, 2021.

[21] R. Buchholz, C. Krätzer, and J. Dittmann, "Microphone classification using fourier coefficients," in *Information Hiding (IH)*, 2009.

[22] C. Krätzer, M. Schott, and J. Dittmann, "Unweighted fusion in microphone forensics using a decision tree and linear logistic regression models," in *Multimedia and Security Workshop (MM&Sec)*, 2009.

[23] D. Garcia-Romero and C. Espy-Wilson, "Automatic acquisition device identification from speech recordings," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2010.

[24] Y. Jiang and F. H. F. Leung, "Source microphone recognition aided by a kernel-based projection method," *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 14, pp. 2875–2886, 2019.

[25] Y. Panagakis and C. Kotropoulos, "Automatic telephone handset identification by sparse representation of random spectral features," in *Multimedia and Security Workshop (MM&Sec)*, 2012.

[26] ——, "Telephone handset identification by feature selection and sparse representations," in *IEEE International Workshop on Information Forensics and Security (WIFS)*, 2012.

[27] C. Hanilçi and F. Ertas, "Optimizing acoustic features for source cell-phone recognition using speech signals," in *ACM Information Hiding and Multimedia Security Workshop (IH&MMSec)*, 2013.

[28] C. Hanilçi and T. Kinnunen, "Source cell-phone recognition from recorded speech using non-speech segments," *Digital Signal Processing*, vol. 35, p. 75–85, 2014.

[29] R. Aggarwal, S. Singh, A. K. Roul, and N. Khanna, "Cellphone identification using noise estimates from recorded audio," in *International Conference on Communication and Signal Processing (ICCSP)*, 2014.

[30] L. Zou, Q. He, and X. Feng, "Cell phone verification from speech recordings using sparse representation," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015.

[31] Y. Li, X. Zhang, X. Li, Y. Zhang, J. Yang, and Q. He, "Mobile phone clustering from speech recordings using deep representation and spectral clustering," *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 13, pp. 965–977, 2018.

[32] V. Verma, P. Khaturia, and N. Khanna, "Cell-phone identification from recompressed audio recordings," in *National Conference on Communications (NCC)*, 2018.

[33] D. Luo, P. Korus, and J. Huang, "Band energy difference for source attribution in audio forensics," *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 13, pp. 2179–2189, 2018.

[34] T. Qin, R. ding Wang, D. Yan, and L. Lin, "Source cell-phone identification in the presence of additive noise from CQT domain," *Information*, vol. 9, p. 205, 2018.

[35] G. Baldini and I. Amerini, "Smartphones identification through the built-in microphones with convolutional neural network," *IEEE Access*, vol. 7, pp. 158 685–158 696, 2019.

[36] X. Lin, J. Zhu, and D. Chen, "Subband aware cnn for cell-phone recognition," *IEEE Signal Processing Letters*, vol. 27, pp. 605–609, 2020.

[37] G. Baldini and I. Amerini, "Microphone identification based on spectral entropy with convolutional neural network," in *IEEE International Workshop on Information Forensics and Security (WIFS)*, 2022.

[38] D. Gärtner, C. Dittmar, P. Aichroth, L. Cuccovillo, S. Mann, and G. Schuller, "Efficient cross-codec framing grid analysis for audio tampering detection," in *AES International Convention*, 2014.

[39] A. J. Cooper, "Detecting butt-spliced edits in forensic digital audio recordings," in *AES International Conference on Audio Forensics (ICAF)*, 2010.

[40] X. Meng, C. Li, and L. Tian, "Detecting audio splicing forgery algorithm based on local noise level estimation," in *International Conference on Systems and Informatics (ICSAI)*, 2018.

[41] Y. D, M. Li, M, and G. J, "Exposing speech transsplicing forgery with noise level inconsistency," *Application-Aware Multimedia Security Techniques*, vol. 2021, pp. 1–6, 2021.

[42] K. Zhang, S. Liang, S. Nie, S. He, J. Pan, X. Zhang, H. Ma, and J. Yi, "A robust deep audio splicing detection method via singularity detection feature," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022.

[43] S. Jadhav, R. Patole, and P. Rege, "Audio splicing detection using convolutional neural network," in *International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, 2019.

[44] N. D. Gaubitch, M. Brookes, P. A. Naylor, and D. Sharma, "Single-microphone blind channel identification in speech using spectrum classification," in *European Signal Processing Conference (ESPC)*, 2011.

[45] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Berkeley Symposium on Mathematical Statistics and Probability*, 1967.

[46] J. Garofolo, L. Lamel, W. Fisher, J. Fiscus, D. Pallett, N. Dahlgren, and V. Zue, "TIMIT Acoustic-Phonetic Continuous Speech Corpus," 1993.

[47] V. Verma and N. Khanna, "CNN-based system for speaker independent cell-phone identification from recorded audio," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[48] Z. Borsos, Y. Li, B. Gfeller, and M. Tagliasacchi, "Micaugment: One-shot microphone style transfer," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.

[49] A. Giganti, L. Cuccovillo, P. Bestagini, P. Aichroth, and S. Tubaro, "Speaker-independent microphone identification in noisy conditions," in *European Signal Processing Conference (EUSIPCO)*, 2022.

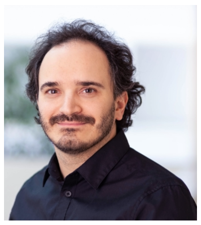
[50] N. Zeghidour, O. Teboul, F. de Chaumont Quitry, and M. Tagliasacchi, "Leaf: A learnable frontend for audio classification," in *International Conference on Learning Representations (ICLR)*, 2021.

[51] T. Tieleman and G. Hinton, "Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude," COURSERA: Neural Networks for Machine Learning, 2012.

[52] L. Cuccovillo and P. Aichroth, "Open-set microphone classification via blind channel analysis," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016.

[53] A. Nautsch, X. Wang, N. Evans, T. Kinnunen, V. Vestman, M. Todisco, H. Delgado, M. Sahidullah, J. Yamagishi, and K. A. Lee, "ASVspoof 2019: Spoofing Countermeasures for the Detection of Synthesized, Converted and Replayed Speech," *IEEE Transactions on Biometrics, Behavior, and Identity Science*, vol. 3, pp. 252–265, 2021.

Daniele Ugo Leonzio was born in Foggia, Italy, in 1997. He received the M.Sc degree in Music and Acoustic Engineering from the Politecnico di Milano, Italy, in 2021. He is currently a Ph.D. student with the Image and Sound Processing Lab (ISPL) at the Department of Electronics, Information and Bioengineering (DEIB) of the Politecnico di Milano. His research activity focuses on the study of multimedia signal processing techniques.



Luca Cuccovillo has received his M.Sc. in Computer Science and Engineering at the Politecnico di Miano (Italy). He is currently pursuing his PhD at Ilmenau University of Technology (Germany), on the topic of audio signal analysis for multimodal tampering detection. He is working as a researcher at Fraunhofer IDMT since 2013, in the research group of media distribution and security. His research activity focuses on signal processing and trustworthy AI, and on their applications to audio forensics and fight against disinformation.

Paolo Bestagini (M'11) was born in Novara, Italy, in 1986. He received the M.Sc. degree in Telecommunications Engineering and the Ph.D. degree in Information Technology from the Politecnico di Milano, Italy, in 2010 and 2014, respectively. He is currently Assistant Professor with the Image and Sound Processing Lab (ISPL) at the Department of Electronics, Information and Bioengineering (DEIB) of the Politecnico di Milano. His research interests focus on multimedia forensics and acoustic signal processing for microphone arrays. He has been scientific investigator for the European projects SCENIC and REWIND coordinated by the Politecnico di Milano. He has been co-principal investigator for the DARPA-funded MediFor project. He is co-principal investigator for the DARPA-funded SemaFor project. He is an elected member of the IEEE Information Forensics and Security Technical Committee for the second time. He serves as an Associate Editor for the IEEE Transactions on Circuits and Systems for Video Technology (TCSVT) and the Elsevier Journal of Visual Communication and Image Representation (JVCI). He is Co-Organizer of the IEEE Signal Processing Cup 2018 and 2022.

Marco Marcon was born in Bollate (Milan) in 1972. He completed his studies in Electronic Engineering at the Politecnico di Milano, Italy in 1998. He received the Ph.D. in Applied Physics in 2003 from the Department of Physics of the Politecnico di Milano. In 2003 he joined the Department of Electronics, Information and Bioengineering (DEIB) of the Politecnico di Milano, where he is actually working as a research scientist in the Image and Sound Processing Lab (ISPL). His current research interests are in the area of Digital Image Processing and Computer Vision for Biometry and 3D object reconstruct from multiple uncalibrated views. He participated to the Origami European project and to the Visnet Network of Excellence and he is participating to its continuation: Visnet II NoE.

Patrick Aichroth worked as a freelance software developer and IT trainer before becoming a research associate at Fraunhofer IDMT in 2003. Since 2006, he has been head of the Media Distribution and Security Group at Fraunhofer IDMT, which focuses on the development of technologies for audio manipulation detection and provenance analysis, media security, privacy enhancing technologies and trustworthy AI.

Stefano Tubaro (SM'01) was born in Novara, Italy, in 1957. He received the M.Sc. degree in Electronic Engineering from the Politecnico di Milano, Milan, Italy, in 1982. He then joined the Department of Electronics, Information and Bioengineering (DEIB), Politecnico di Milano, first as a Researcher of the National Research Council, and then in 1991 as an Associate Professor. Since 2004, he has been appointed as a Full Professor of telecommunication with the Politecnico di Milano. His current research interests include advanced algorithms for video and sound processing. He coordinates the research activities of the Image and Sound Processing Lab (ISPL) at the Department of Electronics, Information and Bioengineering (DEIB), Politecnico di Milano. He had the role of a Project Coordinator of the European Project ORIGAMI: A new paradigm for high-quality mixing of real and virtual and of the research project ICT-FET-OPEN REWIND: REVerse engineering of audio-VIsual coNtent Data. This last project was aimed at synergistically combining principles of signal processing, machine learning, and information theory to answer relevant questions on the past history of such objects. He has authored more than 180 scientific publications on international journals and congresses and has coauthored more than 15 patents. In the past few years, he has focused his interest on the development of innovative techniques for image and video tampering detection and, in general, for the blind recovery of the processing history of multimedia objects. He is a member the IEEE Multimedia Signal Processing Technical Committee and of the IEEE SPS Image Video and Multidimensional Signal Technical Committee. He was in the organization committee of a number of international conferences, including the IEEE MMSP 2004/2013, the IEEE ICIP 2005, the IEEE AVSS 2005/2009, the IEEE ICDSC 2009, the IEEE MMSP 2013, and the IEEE ICME 2015. From 2012 to 2015, he was an Associate Editor of the IEEE Transactions on Image Processing (TIP), and he is currently an Associate Editor of the IEEE Transactions on Information Forensics and Security (TIFS).