POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

A metric learning approach for splicing localization based on synthetic speech detection

Laurea Magistrale in Music and Acoustic Engineering

Author: Francesco Castelli

Advisor: Prof. Paolo Bestagini

Co-advisor: Clara Borrelli, Davide Salvi

Academic year: 2020-2021

1. Introduction

Nowadays, the generation of realistic synthetic media, as images, videos and audio tracks, is a relatively easy task thanks to the recent progress in Artificial Intelligence (AI). On one side, these technologies can open the doors to new exciting scenarios, for instance, developing speech-driven human-computer interfaces. On the other hand, when misused, they can produce unpleasant situations. This is the case of Deepfakes (DFs), data produced using Deep Learning (DL) that realistically represent people in deceptive behaviors. Audio DFs have already been used for fraud and fake news spreading, making it crucial to be able to discriminate between real and fake speech automatically. DF generation techniques can be used not only to forge fully synthetic signals but also to replace portions of real speech signals. We refer to this manipulation technique as audio splicing, i.e., the track is generated by concatenating two (or more) different signals, whether real or fake. This technique can produce incredibly realistic tracks and fool the state-of-the-art detectors when used for malicious purposes since the audio still contains sections of real signals. In this work, we consider the problem of splicing detection and localization based on synthetic voice production. We are interested only in spliced audio created by substituting parts of a pristine speech with synthetically generated speech. The proposed method extracts embeddings from the input signal through a sliding window and it employs the novelty function [1] to analyze the similarity between the embeddings. When the novelty function shows peaks of significant prominence, a splicing point is detected and localized.

2. Proposed method

In this work, we aim to perform splicing detection and localization of synthetically generated signals. Given an input speech signal $\boldsymbol{y}$, we say that it contains a splicing if it has been generated by the concatenation of multiple audio tracks. Establishing the presence or absence of a splicing point corresponds to the detection task while estimating the time instant when it occurs is the localization task. In particular, we focus on the concatenations between real and fake signals, looking for the transitions between speeches of these two classes. We consider all the authentic signals as real, while as fake all those generated with any DF technique. As an example, given two speech signals $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ belonging to two

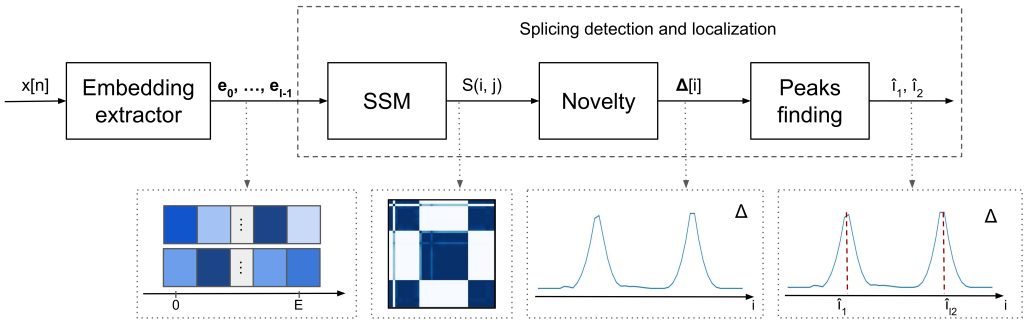


Figure 1: Pipeline of the proposed method.

different classes l, where $l \in \{\text{REAL}, \text{FAKE}\}$, we define

$$\boldsymbol{y} = [\boldsymbol{s}_1, \boldsymbol{s}_2] \tag{1}$$

as a spliced signal and we aim at localizing the sample at which the concatenation occurs. We also address multiple splicings in the same audio track. Finally, since our method focuses on real-fake transitions, non-spliced signals are made by concatenating signals of the same class.

Our proposed method is composed of three main stages. First, we use a sliding window to split the input signal into frames and an embedding extractor to map them into the corresponding embeddings in a high-dimensional space. Then we calculate the self-similarity matrix (SSM) $\boldsymbol{S}$ of the embeddings using the squared euclidean distance as a measure of dissimilarity. Finally, we compute the novelty function from the matrix $\boldsymbol{S}$ and look for peaks of sufficient prominence. Those correspond to the detected splicing points. Figure 1 shows the complete pipeline.

2.1. Embedding extractor

At this stage we extract embeddings from different time windows of the input signal. Starting from the signal under analysis $\boldsymbol{y}$, we divide it into I frames $\boldsymbol{x}_i$, $i = 0, 1, ..., I-1$, using a rectangular window of length W and overlap of H seconds. Then, we give each frame $\boldsymbol{x}_i$ as input to an embedding extractor $\mathcal{E}$ that turns it into an embedding $\boldsymbol{e}_i \in \mathbb{R}^E$. In this work we consider RawNet2 [3] as extractor $\mathcal{E}$, thus we feed the frames $\boldsymbol{x}_i$ to the network without any feature extraction. At the end of this step, we extract exactly I embeddings of dimension E.

It is worth noticing that our method is not strictly dependent on RawNet2, and other Deep Neural Networks (DNNs) can be used as a drop-in replacement. Specifically, we also tested a modified version of RawNet2 using a deep metric learning approach in the training phase.

2.2. Splicing detection and localization

At this step we detect and localize the splicing points in the audio track $\boldsymbol{y}$, if present. Given the I embeddings $\boldsymbol{e}_0, \boldsymbol{e}_1, ..., \boldsymbol{e}_{I-1}$ extracted from the speech signal under analysis, we compute the distance matrix $\boldsymbol{D}$ between them. $\boldsymbol{D}$ is a $I \times I$ symmetric matrix, such that each element $\boldsymbol{D}(i,j)$ is computed using the squared euclidean distance

$$\boldsymbol{D}(i,j) = \|\boldsymbol{e}_i\|^2 - 2\langle \boldsymbol{e}_i, \boldsymbol{e}_j \rangle + \|\boldsymbol{e}_j\|^2, \tag{2}$$

where $\boldsymbol{e}_i$ and $\boldsymbol{e}_j$ are the considered embeddings. The matrix $\boldsymbol{D}$ is then turned into a SSM $\boldsymbol{S}$ such that each element $\boldsymbol{S}(i,j)$ is computed as

$$\boldsymbol{S}(i,j) = exp((\boldsymbol{D}(i,j))/\sigma_D), \tag{3}$$

where σ_D is the standard deviation of the matrix $\boldsymbol{D}$. Then, a novelty value $\boldsymbol{\Delta}[i]$ is computed for each time window i by correlating a checkerboard-like kernel $\boldsymbol{K}$ of size L along the main diagonal of the $\boldsymbol{S}$ matrix:

$$\boldsymbol{\Delta}[i] = \sum_{k,l \in [-L,L]} \boldsymbol{K}(k,l) * \boldsymbol{S}(i+k, i+l). \tag{4}$$

The novelty function $\boldsymbol{\Delta}$ exhibits noticeable peaks if any splicing is present in the input audio. We select peak $\hat{i}$ as candidate splicing point based on its prominence, which defines how much it stands out from the surrounding baseline of the signal, and height $\boldsymbol{\Delta}[\hat{i}]$. All the peaks with a prominence value greater than an absolute prominence P and height $\boldsymbol{\Delta}[\hat{i}]$ greater than a threshold T are detected as splicing and

localized at the corresponding peak position in the signal.

3. Experimental setup

3.1. Splicing dataset

To evaluate the proposed method, we built two datasets of spliced tracks called SplicingSingle and SplicingDouble. All the tracks of the two sets have been generated by concatenating different speech signals, both real and fake. The number of concatenated signals is equal to two for SplicingSingle and three for SplicingDouble, resulting in a number of splicing points up to 1 and 2 in the two datasets, respectively. According to our approach, we highlight that two concatenated tracks belonging to the same class (real/fake) do not constitute splicing.

The considered audio chunks used to generate the data are all taken from the Logical Access (LA) partition of the ASVSpoof 2019 dataset [4], which contains both real and fake data. In particular, fake samples are generated considering 19 different systems, including text-to-speech (TTS), voice conversion (VC) or TTS/VC hybrid methods. The first 6 generation methods (named A01, A02, ..., A06) are included in the *train* and *dev* partitions of the dataset. The following ones (A07, ..., A19) are present in the *eval* set only.

The generation pipeline of each audio signal of the two datasets is divided in the following steps:

1. *Selection*: We select speech signals from the ASVspoof 2019 dataset following two constraints. All the selected tracks must contain speech from the same person and all the chosen fake signals in the same track must be generated using the same deepfake technique.
2. *Silence trim*: We trim the leading and trailing silences from the selected tracks, in order to remove the uneven distribution of silences duration of the ASVspoof dataset [2].
3. *Concatenation*: We generate the final audio track by concatenating all the selected speech signals.

In the *Selection* step, the number of chosen signals equals $n = 2$ for the SplicingSingle dataset and $n = 3$ for the SplicingDouble. We created a closed and an open set version for each of the two datasets based on the generation methods of the fake tracks. In particular, we used the *dev* partition of ASVspoof 2019 to create the closed set version and the *eval* partition for the open set. Table 1 shows the structure of the two datasets. The audio signals that do not contain any splicing point are those generated by the concatenation of chunks of the same class. These are added to the dataset to prove that the simple concatenation of audio does not introduce artifacts that our method can use to find splicing points. Indeed, if the latter were true, the model would also find splicing points in signals like those.

Table 1: Structure of the splicing datasets.

SplicingSingle			
Set	**0 splicing**	**1 splicing**	**Tot**
Closed	9877	10052	19929
Open	19955	19991	39946
SplicingDouble			
Set	**0 splicing**	**2 splicing**	**Tot**
Closed	10023	9977	20000
Open	20086	19962	40048

3.2. Embedding extractor

In the experimental phase we considered two different types of embedding extractors, called $\mathcal{E}_1$ and $\mathcal{E}_2$, both based on RawNet2 [3].

The first embedding extractor $\mathcal{E}_1$ corresponds to the exact implementation of RawNet2 proposed in [3], with 129 filters in the SincNet layer and a Mel-distributed filterbank initialization. We only changed the duration of the input audio signal from 4 seconds (64000 samples) to 0.5 seconds (8000 samples) and either padding short audio or selecting a random crop of 0.5 seconds in long signals. We re-trained the network from scratch on the *train* partition of the ASVspoof 2019 LA dataset with Binary Cross Entropy (BCE) loss function and mini-batch size $B = 256$, trimming all the leading and trailing silences of the signals before feeding them to the network.

The second embedding extractor $\mathcal{E}_2$ we consider is a modified version of RawNet2. Specifically, we applied a two-heads (classification and embedding) network structure. The classification head corresponds to the Fully Connected (FC) layers of RawNet2, while the embedding head is

composed of a single FC layer with 512 neurons attached to the Gated Recurrent Unit (GRU) layer. During training we used the loss function

$$\mathcal{L} = \mathcal{L}_{soft} + \lambda \mathcal{L}_{triplet}, \tag{5}$$

where $\mathcal{L}_{soft}$ is BCE loss on the classification head and $\mathcal{L}_{triplet}$ is triplet loss used on L2 normalized embeddings at the output of the embedding head. The balancing hyper-parameter λ was set to $\lambda = 1.2$. Online triplet mining is used to produce triplets on the fly, starting from a mini-batch of $B = 512$ embeddings. In particular, we used hard-sampling as triplet's sampling strategy, imposing two additional constraints on selecting positive/negative. These must present the same speaker of the anchor when the audio is real/fake and the same deepfake generation technique when the audio is fake.
When $\mathcal{E}_1$ is used, embeddings are taken from the output of the GRU layer with a dimension $E_1 = 1024$. In the case of $\mathcal{E}_2$, embeddings come from the embedding head, with a dimension $E_2 = 512$. The parameters for the sliding window for both $\mathcal{E}_1$ and $\mathcal{E}_2$ are $W = 0.5s$ and $H = 0.125s$.

3.3. Splicing detection and localization

For the novelty computation we use a Gaussian kernel $\boldsymbol{K}_{Gauss}$ of size $L = 6$:

$$\boldsymbol{K}_{Gauss}(k,l) = \phi(k,l) \cdot \boldsymbol{K}_{box}(k,l), \tag{6}$$

for $k, l \in [-L, L]$. The kernel $\boldsymbol{K}_{box}$ is defined as

$$\boldsymbol{K}_{box}(k,l) = \text{sgn}(k) \cdot \text{sgn}(l),\ k, l \in [-L, L], \tag{7}$$

and $\phi : \mathbb{R}^2 \to \mathbb{R}$ is a radially-symmetric Gaussian function computed as

$$\phi(s,t) = exp(-\varepsilon^2 (s^2 + t^2)), \tag{8}$$

with $\varepsilon = 0.11$ used to control the degree of tapering toward 0 at the edges of the kernel. We set the prominence to $P = 0.2$ and threshold to $T =$ for $\mathcal{E}_1$, while values $P =$ and $T =$ are used to $\mathcal{E}_2$. These values are found by analyzing the detection accuracy at different threshold and prominence on SplicingSingle and SplicingDouble. As baseline system $\mathcal{E}_{2bsl}$ we use the squared Euclidean distance among consecutive embeddings to create the function $\boldsymbol{\Delta}_{dist}$:

$$\Delta_{dist}[i] = |\boldsymbol{e}_{\boldsymbol{i}}^2 - \boldsymbol{e}_{\boldsymbol{i+1}}^2|, \tag{9}$$

where $\boldsymbol{e}_i$ and $\boldsymbol{e}_{i+1}$ are embeddings extracted with $\mathcal{E}_2$. $\boldsymbol{\Delta}_{dist}$ should still present peaks if any splicing point is present and replace $\boldsymbol{\Delta}$.

4. Results

4.1. Anti-spoofing

We evaluate the two embedding extractors $\mathcal{E}_1$ and $\mathcal{E}_2$ on the *eval* partition of the ASVSpoof 2019 dataset. The overall balanced accuracy for extractor $\mathcal{E}_1$ is BA $= 0.75$ and EER $= 0.26$. Embedding extractor $\mathcal{E}_2$, instead, achieved an overall balanced accuracy of BA $= 0.74$ and EER $= 0.29$. Table 2 shows a more extensive comparison of the True Negative Rates (TNRs) on the real signals and True Positive Rates (TPRs) broken down for each fake generation system. The two models shows the same trend among the generation systems, probably because they are variations of the same architecture. The two methods do not reflect the performances indicated in the literature since they do not have access to silences [2]. All the worst-performing systems (A10, A12 and A15) are TTS systems that use a state-of-the-art waveform generator as WaveNet and WaveRNN.

4.2. Splicing detection

We now evaluate our method on the splicing detection task. We highlight that performances on SplicingDouble dataset may be higher concerning SplicingSingle, since the presence of more splicing points increase the likelihood of founding at least one of them. On the closed set versions $\mathcal{E}_1$ and $\mathcal{E}_2$ respectively reach a balanced accuracy of $\text{BA}_{det} = 0.87$ and $\text{BA}_{det} = 0.94$ on SplicingSingle, while $\text{BA}_{det} = 0.88$ and $\text{BA}_{det} = 0.93$ on SplicingDouble. The baseline reach instead $\text{BA}_{det} = 0.9$ on SplicingSingle and $\text{BA}_{det} = 0.92$ on SplicingDouble. On the open set versions, since spliced data of both datasets are created by concatenating real and fake signals generated with systems A07-A19, it plausible to think that errors of the extractor $\mathcal{E}$ directly influence the splicing detection performances. In Table 3 we observe the same TPRs trend across fake generation systems A07-A19 that we saw in Table 2 when evaluating $\mathcal{E}_1$ and $\mathcal{E}_2$ on the anti-spoofing task, confirming our hy-

Table 2: TPRs and TNRs per fake generation systems of the *eval* partition of ASVspoof 2019.

	TNR	TPR												
$\mathcal{E}$	Real	A07	A08	A09	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19
$\mathcal{E}_1$	0.850	1.000	1.000	0.604	0.167	0.836	0.085	0.428	1.000	0.160	1.000	0.688	0.460	1.000
$\mathcal{E}_2$	0.872	1.000	0.999	0.351	0.125	0.840	0.058	0.433	1.000	0.094	1.000	0.578	0.363	1.000

Table 3: TPRs on the open set versions of the SplicingSingle (**1spl**) and SplicingDouble (**2spl**) datasets.

	$\mathcal{E}$	A07	A08	A09	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19
1 spl	$\mathcal{E}_1$	0.921	0.901	0.304	0.116	0.474	0.103	0.345	0.909	0.139	0.910	0.343	0.366	0.922
	$\mathcal{E}_2$	0.983	0.940	0.065	0.041	0.406	0.045	0.380	0.990	0.050	0.976	0.230	0.270	0.988
	$\mathcal{E}_{2bsl}$	0.899	0.915	0.322	0.094	0.725	0.059	0.533	0.939	0.089	0.901	0.531	0.240	0.856
2 spl	$\mathcal{E}_1$	0.971	0.965	0.548	0.397	0.721	0.343	0.567	0.970	0.372	0.974	0.617	0.582	0.977
	$\mathcal{E}_2$	0.998	0.981	0.306	0.303	0.615	0.289	0.556	0.997	0.282	0.995	0.479	0.464	1.000
	$\mathcal{E}_{2bsl}$	0.972	0.977	0.540	0.362	0.818	0.333	0.667	0.986	0.331	0.957	0.723	0.451	0.958

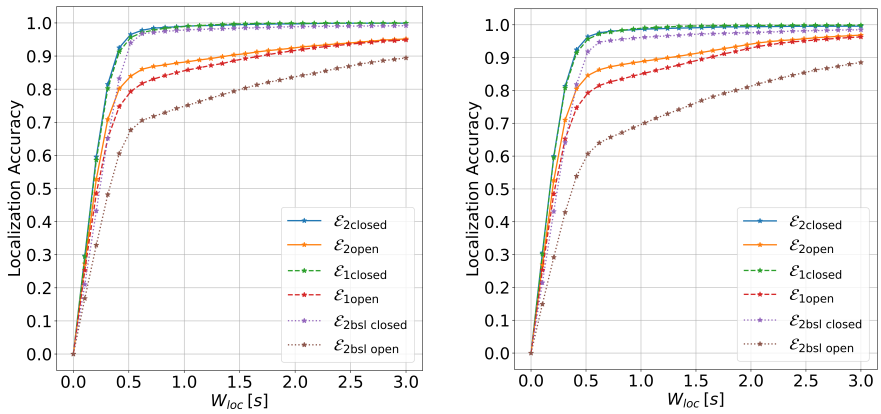
Table 4: Number of detected splicing points per spliced audio of SplicingSingle.

Closed set			
$\mathcal{E}$	0 spl	1 spl	> 1 spl
$\mathcal{E}_1$	956	9092	4
$\mathcal{E}_2$	268	9727	57
$\mathcal{E}_{2bsl}$	1011	8647	394
A07, A08, A14, A16, A19			
$\mathcal{E}$	0 spl	1 spl	> 1 spl
$\mathcal{E}_1$	681	7034	59
$\mathcal{E}_2$	193	7526	55
$\mathcal{E}_{2bsl}$	761	6387	626

pothesis. Specifically, $\mathcal{E}_1$ and $\mathcal{E}_2$ show very good performances on systems A07, A08, A14, A16 and A19. By restricting our view on those systems, we can still see that $\mathcal{E}_2$ consistently outperforms $\mathcal{E}_1$ and the baseline on both datasets. Moreover, on non-spliced signals (TNRs) of both datasets extractor $\mathcal{E}_2$ (TNR = 0.87 on SplicingSingle and TNR = 0.81 on SplicingDouble) significantly outperforms $\mathcal{E}_1$ (TNR = 0.81 on SplicingSingle and TNR = 0.72 on SplicingDouble). With a TNR = 0.8 on SplicingSingle and TNR = 0.73 on SplicingDouble the baseline system is not able to reach the results of the novelty function.

4.3. Splicing localization

We introduce the localization accuracy metric ACC_{loc} used for splicing localization. Let us consider a spliced audio signal y in which the correct number of splicing points is detected by the system. We consider a ground truth splicing point as localized if it is inside a window of length W_{loc}, centered around the predicted splicing point. The audio y is then correctly localized if all the ground truth splicing points in it are localized. We define ACC_{loc} as the ratio among the correctly localized audio and the total number of audio in which at least the correct number of splicing points is predicted. Is thus interesting to look at the number of splicing points detected by our method in each **spliced** audio of SplicingSingle and SplicingDouble. Thus, we restrict the two datasets at only the spliced signals. As explained in Section 4.2, in the open set versions we only consider spliced audio of fake generation systems A07, A08, A14, A16 and A19. Table 4 shows that extractor $\mathcal{E}_1$ do not detect any splicing point in almost 10% of the spliced audio of SplicingSingle, which confirms the lower detection results regarding $\mathcal{E}_2$. Table 5 shows that on SplicingDouble closed $\mathcal{E}_2$ outperforms $\mathcal{E}_1$, which only predict one of two splicing points in almost 15% of the spliced audio. On the open set both extractor are missing one of the two splicing points in a fair amount of cases. The baseline system is instead predicting more splicing points than the ones truly present in both datasets: $\mathbf{\Delta}_{dist}$ is vulnerable to small error of the extractor on top, causing abrupt changes which are detected as splicing points. Finally, we present the localization accuracy of the two ex-



(a) SplicingSingle dataset (b) SplicingDouble dataset

Figure 2: Splicing localization accuracy on SplicingSingle and SplicingDouble datasets.

Table 5: Number of detected splicing points per spliced audio of SplicingDouble.

Closed set				
$\mathcal{E}$	0 spl	1 spl	2 spl	> 2 spl
$\mathcal{E}_1$	91	1444	8430	12
$\mathcal{E}_2$	12	336	9552	77
$\mathcal{E}_{2bsl}$	104	1742	7606	525
A07, A08, A14, A16, A19				
$\mathcal{E}$	0 spl	1 spl	2 spl	> 2 spl
$\mathcal{E}_1$	216	2693	4553	105
$\mathcal{E}_2$	41	2180	5257	89
$\mathcal{E}_{2bsl}$	226	2243	4167	931

tractor in Figure 2 at variable localization window length W_{loc}. We should interpret ACC_{loc} as a measure of the effectiveness of the system in localize splicing points correctly with a certain degree of error (W_{loc}), but we must always contextualize it by considering the system ability to correctly detect the right number of splicing points. Even if $\mathcal{E}_2$ is slightly better than $\mathcal{E}_1$ on the closed set versions, both extractors have very good localization ability. On the open set version of both datasets we observe a deterioration of performances. Also in this case $\mathcal{E}_2$ exceed $\mathcal{E}_1$, with a ACC_{loc} 4% higher at $W_{loc} = 0.5s$ in both datasets. The novelty function clearly outperforms the baseline on localization.

5. Conclusion

We presented a system for detecting and localizing the spliced regions in a tampered speech signal and built two datasets for its evaluation. It is based on a DNN trained on the anti-spoofing task, used as embedding extractor, and the novelty function computed on the embeddings to find the splicing points. The presented system outperformed the considered baseline and proved robust to multiple splicings. Furthermore, we have observed the importance of a metric learning approach used in the training phase for the overall performance of splicing detection and localization. Indeed, even if $\mathcal{E}_2$ shows lower performances on anti-spoofing, its ability to better position the embeddings in the space is of crucial importance in both splicing detection and localization.

References

[1] J. Foote. Automatic audio segmentation using a measure of audio novelty. In *IEEE International Conference on Multimedia and Expo (ICME)*, 2000.

[2] Nicolas M., Franziska D., Pavel C., Roman C., Konstantin B., and Jennifer W. Speech is Silver, Silence is Golden: What do ASVspoof-trained Models Really Learn?, 2021.

[3] H. Tak, J. Patino, M. Todisco, A. Nautsch, N. Evans, and A. Larcher. End-to-end anti-spoofing with RawNet2. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.

[4] M. Todisco, X. Wang, V. Vestman, M. Sahidullah, H. Delgado, A. Nautsch, J. Yamagishi, N. Evans, T. Kinnunen, and K.g A. Lee. ASVspoof 2019: Future horizons in spoofed and fake audio detection. *arXiv preprint arXiv:1904.05441*, 2019.