AUDIO CROSS VERIFICATION USING DUAL ALIGNMENT LIKELIHOOD RATIO TEST

Heidi Lei[†⋆] *Arm Wonghirundacha*[‡⋆] *Irmak Bukey*[‡⋆] *TJ Tsai*[§]

[†]Massachusetts Institute of Technology
[‡]Pomona College
[§]Harvey Mudd College

ABSTRACT

This paper explores a way to verify that audio has not been maliciously tampered in a specific context: short viral videos taken from news recordings. Rather than trying to detect artifacts of tampering (internal inconsistency), we focus on positively verifying a query against a trusted source such as a news recording (external consistency). We propose a method for cross verifying a short audio query against a reference recording from which it was taken. Our approach is to define two hypotheses (non-tampered vs tampered), calculate the most likely alignment between query and reference for each hypothesis, and then perform a likelihood ratio test on the two alignments. We show that this method is fast to compute, much more robust than using MFCC features with Euclidean distance, and has the key benefit of explainability. Our cross verification approach provides an alternative perspective and complementary tool to existing tampering detection methods.

Index Terms— Cross verification, audio authentication, tampering detection, forensics

1. INTRODUCTION

One significant issue in our society today is the reliability of audiovisual information. The availability of deep fake technology and audio/video digital editing software has made it easy for non-experts to generate or modify audiovisual content in a way that seems realistic. These technologies have been used for nefarious purposes such as nonconsensual pornography and political defamation [1], and their use casts doubt on the authenticity of legitimate audiovisual data [2].

This paper focuses on a very specific subset of this general problem: verifying the authenticity of viral videos posted on social media whose source content was directly recorded by major news agencies. This includes speeches by political leaders or other high profile individuals, which tend to be on matters of national interest and thus are especially important to consider. One could imagine a company like Twitter providing a green check mark next to a viral video that says "Verified against NBC" (or some other major news agency), which would provide the viewer confidence that what they are watching can be trusted and has not been tampered in a malicious way. On a technical level, this problem has two inputs: a short query (e.g. a low-res 10 second video clip of a campaign speech that is posted on social media) and a long reference recording from which the query is taken or adapted (e.g. a high-res 30 minute recording of the entire campaign speech as recorded by a major news agency). The goal is then to compare the query and reference, and to determine if the query matches the reference (verified matching) or if it has been tampered (verified tampered). In this work, we focus exclusively on verifying audio content. This is the audio cross verification problem.

There is a large body of previous work on detecting audio tampering. Audio tampering involves modifying a genuine audio recording through insertion, deletion, or replacement with foreign material. Many audio forensics methods have been proposed to detect artifacts of tampering, such as double compression [3][4], broken watermarks [5][6], discontinuities in the embedded electrical network frequency (ENF) [7][8], inconsistencies in the acoustic environment signature [9][10], or singularities in the spectrogram [11][12][13]. See [14] for a survey of previous work. The timeliness of this topic is shown by the recent introduction of new datasets for speaker verification [15] and Deepfake video detection [16] to study short duration temporal forgeries. While many of the above methods work well in detecting tampering, they can also be extremely vulnerable to anti-forensic countermeasures, as recent studies have shown [17][18][19][20].

While similar in application, the audio cross verification problem has a different objective than audio tampering detection. Rather than trying to detect artifacts of tampering (internal inconsistency), it instead focuses on positively verifying a query against a trusted source (external consistency).[1] This change in paradigm has two significant benefits and one significant drawback. The first benefit is that the efficacy of a cross verification method does not depend on how seamlessly a tampering operation is done (since it is not looking for artifacts of tampering). In this way, it provides a permanent and stable solution rather than one that depends on the state of deepfake technology or anti-forensic countermeasures. The second benefit is that cross verification methods can make a stronger claim than audio tampering methods. An audio tampering method can at best say, "There is no evidence of tampering (though it's possible that the tampering was done seamlessly or the whole recording is a deepfake)". A cross verification method can say, "This clip matches a trusted source, so it can be trusted." The drawback of cross verification methods is that they can only be used when a trusted reference is available, which limits the contexts in which they can be used.

An ideal audio cross verification method would have several desirable characteristics. First, it should be extremely accurate in predicting whether a query is matching or tampered. Second, it should be robust to real-world conditions. In particular, we would want the method to be insensitive to harmless differences like audio encoding formats, volume differences, and audio compression artifacts, while at the same time being very sensitive to changes due to tampering operations like insertion, deletion, and replacement. We would also

[⋆]Equal contribution

[1]Reference-based audio authentication is not new – it has been explored in the audio forensics community, primarily through extracting the ENF signal from the query and comparing against an ENF database from the electric grid. This technique can be used to ascertain the timestamp of recording and assess if any tampering has been done (e.g. [21][22][23][24]). In this paper, we adopt a similar approach but focus on directly verifying the audio content.

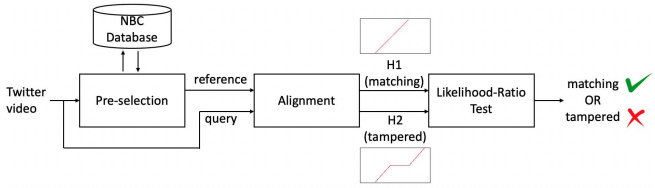

Fig. 1. Overview of the proposed Dual Alignment Likelihood Ratio Test (DA-LRT) method for audio cross verification.

want the method to be robust to overfitting, since complex models can be vulnerable to distribution shift. Third, an ideal method would be fast in terms of runtime. Fourth, the method should be explainable. Since this tool would be public facing and have ramifications in the public sphere, it is necessary for predictions to be justifiable.

We propose an approach called Dual Alignment Likelihood Ratio Test (DA-LRT) that is designed around the desirable characteristics above. DA-LRT consists of three stages. The first stage is pre-selection, in which we use a binary feature representation to quickly determine the rough offset where the query occurs in the reference. The second stage is to perform two different alignments between the query and the reference. One alignment is performed under the hypothesis that there is no tampering, and the other alignment is performed under the hypothesis that tampering is present in the form of insertions, deletions, and replacements. The third stage is to identify differences between the two alignments, and then to perform a statistical test to determine which hypothesis is more likely given the observed features. We describe this in detail in the next section.

2. SYSTEM DESCRIPTION

The DA-LRT method has three main stages (see Figure 1), which are described in detail in the next three subsections.[2] Our goal is to compare a short query recording to a reference recording (from which it was taken), and to determine if the query matches the reference or if it has been tampered.

2.1. Pre-Selection

The first stage is pre-selection, in which the goal is to find the original source material that matches the query. File-level pre-selection can be done using an audio fingerprinting method (e.g. Shazam [25], Google [26]) or metadata search (e.g. speaker and date). To keep our discussion focused, we will simply assume in this paper that a matching reference recording has already been identified.[3]

Given a matching reference recording, we would also like to perform temporal pre-selection in order to identify the part of the reference recording that matches the query. Because stage 2 requires dynamic programming, this temporal pre-selection can significantly reduce total runtime if the reference is much longer than the query. We use a simplified version of audio hashprints [27] to accomplish this quickly and efficiently. This approach consists of three steps. First, we compute standard mel frequency cepstral coefficients (MFCCs) with 25 ms analysis frames and 10 ms hop size. We include delta and delta-delta features, which results in a total of 39 features per frame. Second, we represent each frame with a 26-bit binary feature representation by thresholding the delta and delta-delta features at 0. (The MFCC features are stored for later use in stages 2 and 3.) Third, we identify the offset in the reference recording that results in the lowest total Hamming distance between the corresponding binary feature sequences (assuming a 1-to-1 correspondence). This can be computed efficiently by encoding each frame in memory as a single 32-bit integer and performing bit operations. The optimal offset specifies the approximate matching region of the reference recording. Since the query may be tampered from insertion or deletion, we include a short buffer before and after the matching region to provide a conservative estimate. This specifies our pre-selected reference material.

2.2. Dual Alignment

The second stage is to compute two different alignments between the query and the pre-selected reference material. These two alignments are estimated under two different hypotheses (tampered vs. non-tampered). The procedure for the two alignments is described in detail in the remainder of this subsection. We will refer to the query features as $q_0, q_1, \ldots, q_{N-1}$ and the (pre-selected) reference features as $r_0, r_1, \ldots, r_{M-1}$, where $q_i \in \mathbb{R}^{39}$ and $r_i \in \mathbb{R}^{39}$ are the MFCC features for the i^{th} audio frame.

The first alignment is estimated under the assumption that the query is non-tampered. In this case, the alignment path is assumed to be a diagonal line, and the only unknown is the offset at which the matching region begins. We re-estimate[4] the alignment path by (1) computing a pairwise cost matrix $C \in \mathbb{R}^{N \times M}$ between the query and reference using the MFCC features and a Euclidean distance cost metric, (2) calculating the sum of pairwise costs along each complete diagonal path of the cost matrix, and (3) selecting the diagonal alignment path with the minimum total cost. The optimal path has (query, reference) coordinates $(0,\Delta), (1, \Delta+1), \ldots, (N-1, \Delta+N-1)$.

The second alignment is estimated under the assumption that the query is tampered. In this case, the shape of the alignment path is not known in advance, since the query may have been tampered with insertions, deletions, and replacements. We estimate this alignment path using an adaptation of Hidden State Time Warping (HSTW) [28], a previously proposed dynamic programming algorithm that allows for state-based time warping. HSTW allows the alignment path at any location in the cost matrix to be in one of two states, where each state has its own unique time warping characteristics. In our scenario, we define one state to be a "matching" state in which the only allowable transition is (1,1), and we define the other state to be a "tampered" state in which the allowable transitions are (0,1) and (1,0). For completeness, we describe the algorithm below.

Our adaptation of HSTW has four steps. First, we calculate a pairwise cost matrix $C \in \mathbb{R}^{N \times M}$ using MFCCs and Euclidean distance. Second, we initialize the cumulative cost tensor $D \in \mathbb{R}^{2 \times N \times M}$ and corresponding backtrace tensor $B \in \mathbb{N}^{2 \times N \times M}$. Note that the two "planes" of the tensor correspond to the two different states. We will refer to the tampered plane as $D_t \in \mathbb{R}^{N \times M}$ and the matching plane as $D_m \in \mathbb{R}^{N \times M}$. Since the query occurs at an unknown offset in the reference, we initialize $D_m[0,j] = C[0,j]$, $j = 0, 1, \ldots, M-1$ and $D_t[0,j] = \frac{\alpha+\gamma}{2}$, $j = 0, 1, \ldots, M-1$, where α and γ are hyperparameters (described in more detail below). This initialization allows the alignment path to begin at any offset in the reference in either state without penalty. Third, we compute the remaining values in $D[i,j]$, $i = 1, \ldots, N-1$,

[2] Code at https://github.com/HMC-MIR/AudioCrossVerification.

[3] Note that, if the query is mostly tampered material or is a completely synthetic deep fake, a match will not be found in the database. In our Twitter application scenario, one could affix a label that says "No match found with NBC" to provide useful information to the viewer.

[4] We re-estimate the offset using MFCCs since they are more informative than coarsely quantized binary hashprints in the pre-selection stage.

$j = 0, \ldots, M-1$ using the following dynamic programming rules: $D_t[i,j] = min(D_t[i,j-1]+\gamma, D_t[i-1,j]+\alpha, D_m[i-1,j-1]+\gamma+\alpha)$ and $D_m[i,j] = min(D_m[i-1,j-1]+C[i,j], D_t[i,j]+\beta)$. Here, α, γ, and β are hyperparameters specifying the insertion, deletion, and plane transition penalties, respectively. As we compute each element in D, we also update the corresponding element of B to record the optimal transition type. Fourth, we identify the element in the last row of D_t or D_m that has the lowest cumulative cost, and then follow the backpointers in B to determine the optimal alignment path.

2.3. Likelihood-Ratio Test

The third stage is to treat each alignment path as a hypothesis, and then to determine which hypothesis has a higher likelihood based on the observed features. This stage consists of four steps, which are described below. For brevity, we will refer to the matching hypothesis as H1 and the non-matching hypothesis as H2.

The first step is to identify differences between the alignment paths under H1 and H2. Specifically, we partition the query frames q_0, q_1, …, q_{N-1} into two groups: those whose alignments are the same between H1 and H2 (partition A), and those whose alignments are different (partition B). To account for analysis frame offsets, we define "different" to mean that the alignments differ by 2 or more frames. Query frames whose HSTW alignment lies entirely in the tampering plane are placed in partition B. If the two alignment paths are identical, we simply declare the query to be matching.

The second step is to model the observed differences between query and reference features in matching regions (i.e. where the query is non-tampered). We do not know in advance which frames are matching, but we can interpret regions where the two alignment paths agree as circumstantial evidence that such regions are matching. Accordingly, we use the observed features for query frames in partition A to model what a "match" looks like. Let (n_1, m_1), (n_2, m_2), …, (n_L, m_L) be the alignment path coordinates where the two alignment paths are in agreement, $q_{n_i} \in \mathbb{R}^{39}$ and $r_{m_i} \in \mathbb{R}^{39}, i = 1, \ldots, L$ denote the corresponding MFCC feature vectors, and $q_{n_i,f} \in \mathbb{R}$ and $r_{m_i,f} \in \mathbb{R}$ indicate the f^{th} MFCC feature in q_{n_i} and r_{m_i}, respectively. For each of the 39 features, we calculate the mean and variance of the observed feature differences in matching regions as $\mu_f = \frac{1}{L}\sum_{i=1}^{L}(q_{n_i,f} - r_{m_i,f})$ and $\sigma_f^2 = \frac{1}{L-1}\sum_{i=1}^{L}\Big((q_{n_i,f} - r_{m_i,f}) - \mu_f\Big)^2$. The end result is a set of 39 scalar Gaussian distributions $\mathcal{N}(\mu_f, \sigma_f^2)$, $f = 1, \ldots, 39$ that model the feature differences when the query and reference frames are matching. We will use these to compute the likelihood of H1.

The third step is to model the differences between query and reference features in non-matching regions (i.e. where the query is tampered). In tampered regions, we assume that the query and reference features are independent. If we treat each feature as an independent random variable drawn from a scalar Gaussian distribution $\mathcal{N}(\mu_f, \sigma_f^2)$ (as estimated above), then the mean and variance of feature differences in non-matching regions can be estimated as $\tilde{\mu}_f = \frac{1}{L}\sum_{i=1}^{L}(q_{n_i,f} - r_{m_i,f})$ and $\tilde{\sigma}_f^2 = \frac{1}{L-1}\sum_{i=1}^{L}\Big[(q_{n_i,f} - \mu_f^{query})^2 + (r_{m_i,f} - \mu_f^{ref})^2\Big]$, $f = 1, \ldots, 39$. The end result is again a set of 39 scalar Gaussian distributions $\mathcal{N}(\tilde{\mu}_f, \tilde{\sigma}_f^2)$, $f = 1, \ldots, 39$ that model the feature differences in non-matching regions. We will use these to compute the likelihood of H2.

The fourth step is to perform a likelihood-ratio test on the two hypotheses using the query frames in partition B as observations. In other words, we would like to know which hypothesis is more likely given the observed feature differences. To make the solution tractable, we assume that features are independent across time and across feature type. Clearly, these assumptions are faulty (e.g. the features are highly correlated over time), but we apply these simplifying assumptions to both hypotheses, so we don't anticipate that they will cause a systematic bias in one direction. We describe the details of the likelihood-ratio test in the next two paragraphs.

Definitions. Let the query frame indices in partition B be given by $\tilde{n}_1$, $\tilde{n}_2$, …, $\tilde{n}_{N-L}$. (There are L query frames in partition A, so partition B will have $N - L$ elements.) Let the alignment path coordinates for these query frames under H1 be given by $(\tilde{n}_1, \Delta + \tilde{n}_1)$, $(\tilde{n}_2, \Delta + \tilde{n}_2)$, …, $(\tilde{n}_{N-L}, \Delta + \tilde{n}_{N-L})$ and the alignment path coordinates under H2 be given by $(\tilde{n}_1, \tilde{m}_1)$, $(\tilde{n}_2, \tilde{m}_2)$, …, $(\tilde{n}_{N-L}, \tilde{m}_{N-L})$. The feature differences are modeled by $\mathcal{N}(\mu_f, \sigma_f^2)$, $f = 1, \ldots, 39$ under H1 and by $\mathcal{N}(\tilde{\mu}_f, \tilde{\sigma}_f^2)$, $f = 1, \ldots, 39$ under H2.

Likelihood Calculation. Given the independence assumptions, the maximum likelihood criterion is given by:

$$\prod_{i=1}^{N-L}\prod_{f=1}^{39}\frac{1}{\sqrt{2\pi\sigma_f^2}}e^{-\frac{1}{2\sigma_f^2}\left((q_{\tilde{n}_i,f} - r_{\Delta+\tilde{n}_i,f}) - \mu_f\right)^2} \underset{H2}{\overset{H1}{\gtrless}} \prod_{i=1}^{N-L}\prod_{f=1}^{39}\frac{1}{\sqrt{2\pi\tilde{\sigma}_f^2}}e^{-\frac{1}{2\tilde{\sigma}_f^2}\left((q_{\tilde{n}_i,f} - r_{\tilde{m}_i,f}) - \tilde{\mu}_f\right)^2} \quad (1)$$

Note that the two sides of equation 1 use different observations (i.e. $(q_{\tilde{n}_i,f} - r_{\Delta+\tilde{n}_i,f})$ vs $(q_{\tilde{n}_i,f} - r_{\tilde{m}_i,f})$), whereas a typical maximum likelihood formulation has a shared set of observations. Taking the log of both sides, equation 1 simplifies to:

$$\sum_{i=1}^{N-L}\sum_{f=1}^{39}\left(ln(\frac{\tilde{\sigma}_f^2}{\sigma_f^2}) - z_{i,f}^2 + \tilde{z}_{i,f}^2\right) \underset{H2}{\overset{H1}{\gtrless}} 0 \quad (2)$$

where $z_{i,f} \triangleq \frac{(q_{\tilde{n}_i,f} - r_{\Delta+\tilde{n}_i,f}) - \mu_f}{\sigma_f}$ and $\tilde{z}_{i,f} \triangleq \frac{(q_{\tilde{n}_i,f} - r_{\tilde{m}_i,f}) - \tilde{\mu}_f}{\tilde{\sigma}_f}$ are the Z-scores of the observed feature differences under H1 and H2, respectively. The left side of equation 2 is a log likelihood-ratio test statistic, and we use this quantity as the final "tampering score" for the query. This allows us to characterize performance more broadly with a receiver operation characteristic curve.

3. EXPERIMENTAL SETUP

We used the DAPS dataset [29] for our experiments. This dataset contains high-quality audio recordings of 20 different speakers each reading 5 scripts, where scripts are 2–4 minutes long. The dataset also includes recordings in multiple acoustic conditions, which will allow us to study the effect of acoustic environment in future work. For this study, we focus only on the 100 high-quality, low-noise recordings since these most closely match our intended application.

The queries are generated by processing the DAPS data in the following way. First, each of the 100 high quality original recordings is compressed to a lower bitrate at R kbps. This step reflects the fact that videos shared on social media are typically compressed. Second, we randomly sample 10 different 10-second segments from each compressed recording. Third, we tamper each 10-second segment in three different ways to simulate insertion, deletion, and replacement tampering operations. For insertions, we randomly select an L second filler segment from within the same compressed recording (but outside of the selected 10-second segment). This L second filler is then inserted into the segment at a random location in the segment. This arrangement means that the inserted foreign material

is perfectly matched in terms of speaker, speaking style, and acoustic environment. For deletions, we randomly select an L second interval from within the 10-second segment and delete it. For replacements, we randomly select an L second interval from within the 10-second segment, and then replace it with another L second filler randomly selected from within the same compressed recording. In addition to the tampered versions, we also include an untampered version.

Our benchmarks are constructed in the following manner. Given the above method for generating queries, we have a total of 100 x 10 x 4 = 4000 queries for a given bitrate R and tampering duration L. Each of these queries is associated with a reference recording, which is the original DAPS recording from which the query was taken. Half of the speakers are set aside for training, and the other half are set aside for testing. We consider three different bitrates R = 256, 128, and 64 kbps and five different tampering durations L = 4, 2, 1, 0.5, 0.25 seconds. In total, our evaluation includes 15 benchmarks, each containing 2000 training queries and 2000 test queries.

Our evaluation metric is equal error rate (EER). EER is a useful metric because it summarizes performance in a single number and is invariant to priors. Note that we chose to evaluate classification performance at the recording (rather than frame) level since this most closely aligns with the user's experience of our intended application.

4. RESULTS

Table 1 shows the results of our proposed method on the audio cross verification task. The left half of Table 1 shows the performance of the proposed system, and the right half shows the performance of a baseline system ("MFCC-Euclidean") that uses the Euclidean distance between MFCC features as a tampering score. The table shows the EER of both systems across a range of different conditions. (Note that numbers are expressed in percentages, so 0.20 corresponds to a 0.20% EER.) Each row shows the performance with a fixed tampering duration L and query bitrate R, and each column shows the performance in detecting tampering of specific types (i.e. the benchmark only includes one tampering type plus untampered queries). The rightmost column ("all") in each panel shows performance when all tampering types are present, and the bottommost row ("all") in each panel shows performance when all tampering durations are present.

There are three things to notice about the results in Table 1. First, queries tampered with replacement have much higher error rates than queries tampered with insertion or deletion. This is because insertion and deletion operations will cause any audio frames after the tampering point to become unsynchronized with the reference, making it much easier to detect differences. In contrast, tampering through replacement does not cause this global shift, so it can only be detected by observing feature differences within the actual tampered region. It should be noted that for short tampering operations (e.g. L=0.25), our method of randomly selecting intervals may result in replacing silence with silence, so these results may be overly pessimistic. Second, the proposed method substantially reduces the duration of tampering that can be detected reliably. We see that the MFCC-Euclidean baseline has low (but not perfect) error rates for tampering durations of 2 seconds or more, whereas the proposed method has reliable performance for insertion and deletion queries down to L=0.25 sec and for replacement queries down to 0.5-1.0 sec. Third, query bitrate does not affect performance with long tampering durations, but has a moderate effect with short tampering durations. For example, the EER for replacement queries with L=0.5 sec is 3.1% for R=256 and R=128, but worsens to 4.0% for R=64. Because 64 kbps is considered a very low bitrate, we can interpret these results as a kind of worst case scenario.

	Tamp	DA-LRT				MFCC-Euclidean			
	Len	ins	del	rep	all	ins	del	rep	all
256 kbps	4.0s	.00	.00	.00	.00	.00	0.4	.00	.13
	2.0s	.00	.00	.00	.00	.20	1.6	1.2	1.0
	1.0s	.00	.00	.00	.00	1.6	2.0	12.0	6.2
	0.5s	.20	.20	3.1	1.2	4.6	2.2	29	15
	0.25s	.20	.00	33	14	5.4	3.2	40	20
	all	.08	.04	9.5	3.4	2.7	2.0	20	9.2
128 kbps	4.0s	.00	.00	.00	.00	.00	.40	.00	.13
	2.0s	.00	.00	.00	.00	.20	1.6	1.0	.93
	1.0s	.00	.00	.00	.00	1.4	1.8	12	5.9
	0.5s	.20	.40	3.1	1.3	4.6	2.0	30	15
	0.25s	.20	.00	34	15	5.4	3.0	40	20
	all	.08	.08	10	3.7	2.6	1.9	20	9.0
64 kbps	4.0s	.00	.00	.00	.00	.00	.20	.00	.07
	2.0s	.00	.00	.00	.00	.20	1.0	0.4	.53
	1.0s	.00	.00	.20	.07	1.0	1.8	10	5.1
	0.5s	.20	.40	4.0	1.6	3.8	1.8	28	14
	0.25s	.20	.00	38	17	4.8	2.6	39	19
	all	.08	.08	12	4.3	2.2	1.5	19	8.8

Table 1. Comparing the performance of DA-LRT and MFCC-Euclidean distance baseline on the audio cross verification task. Numbers indicate equal error rate in percentages, so 0.20 corresponds to 0.20% EER. Rows show performance for a fixed tampering duration and query bitrate, and columns show performance in detecting specific types of tampering (insertions, deletions, replacements). The rightmost column and bottommost row in each panel show aggregate performance when multiple tampering types and multiple tampering durations are present, respectively.

It is useful to consider the strengths and weaknesses of our proposed method. Recall the four characteristics that we identified in an ideal solution to the cross verification problem: accurate, robust, fast, and explainable. With regards to accuracy and robustness to real-world conditions, our proposed method reliably detects insertions and deletions, since these cause a global shift that results in feature mismatches for all audio frames after the tampering point. The biggest weakness of the system is in detecting short duration (< 0.5 sec) replacements, which must be detected solely on local feature mismatches. With regards to having a fast runtime, the system is fast enough to be useful in an automated system. In our experiments, it took an average of 433 ms to compute the MFCC features on the query, 5.82 sec to compute the MFCC features on the reference, and 123 ms to perform the remaining computations for offset estimation, alignments, and maximum likelihood test.[5] With regards to explainability, our method compares the likelihood of two interpretable hypotheses (tampered vs non-tampered) and makes clear, explicit assumptions about the nature of the alignment and likelihood calculations.

5. ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 1948531.

[5] We used the python_speech_features library for MFCC feature computation, implemented the alignment in cython, and ran our experiments on a 2.4 GHz Intel Xeon CPU.

6. REFERENCES

[1] Mika Westerlund, "The emergence of deepfake technology: A review," *Technology Innovation Management Review*, vol. 9, no. 11, 2019.

[2] Cristian Vaccari and Andrew Chadwick, "Deepfakes and disinformation: Exploring the impact of synthetic political video on deception, uncertainty, and trust in news," *Social Media+ Society*, vol. 6, no. 1, pp. 2056305120903408, 2020.

[3] Aykut Büker and Cemal Hanilçi, "Angular margin softmax loss and its variants for double compressed amr audio detection," in *Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security*, 2021, pp. 45–50.

[4] Da Luo, Rui Yang, Bin Li, and Jiwu Huang, "Detection of double compressed amr audio using stacked autoencoder," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 2, pp. 432–444, 2016.

[5] Valentin A Nita and Amelia Ciobanu, "Tic-tac, forgery time has run-up! live acoustic watermarking for integrity check in forensic applications," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 1977–1981.

[6] RA Dobre, RO Preda, and AE Marcu, "TIC-TAC based live acoustic watermarking with improved forgery detection performances," in *IEEE 25th International Symposium for Design and Technology in Electronic Packaging (SIITME)*, 2019, pp. 408–412.

[7] Yufei Wang, Yongjian Hu, Alan Wee-Chung Liew, and Chang-Tsun Li, "ENF based video forgery detection algorithm," *International Journal of Digital Crime and Forensics (IJDCF)*, vol. 12, no. 1, pp. 131–156, 2020.

[8] Xiaodan Lin and Xiangui Kang, "Supervised audio tampering detection using an autoregressive model," in *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, 2017, pp. 2142–2146.

[9] Tejas Bhangale and Rashmika Patole, "Tampering detection in digital audio recording based on statistical reverberation features," in *Soft Computing and Signal Processing*, pp. 583–591. 2019.

[10] Xuebo Meng, Chen Li, and Lihua Tian, "Detecting audio splicing forgery algorithm based on local noise level estimation," in *2018 5th international conference on systems and informatics (ICSAI)*, 2018, pp. 861–865.

[11] Kanghao Zhang, Shan Liang, Shuai Nie, Shulin He, Jiahui Pan, Xueliang Zhang, Haoxin Ma, and Jiangyan Yi, "A robust deep audio splicing detection method via singularity detection feature," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 2919–2923.

[12] Akanksha Chuchra, Mandeep Kaur, and Savita Gupta, "A deep learning approach for splicing detection in digital audios," in *Congress on Intelligent Systems*, 2022, pp. 543–558.

[13] Chenyu Liu, Jia Li, Junxian Duan, Haifeng Shen, and Huaibo Huang, "LightCvT: Audio forgery detection via fusion of light cnn and transformer," in *2021 10th International Conference on Computing and Pattern Recognition*, 2021, pp. 99–105.

[14] Prabhu R Bevinamarad and MS Shirldonkar, "Audio forgery detection techniques: present and past review," in *2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)*, 2020, pp. 613–618.

[15] Lin Zhang, Xin Wang, Erica Cooper, Nicholas Evans, and Junichi Yamagishi, "The partialspoof database and countermeasures for the detection of short generated audio segments embedded in a speech utterance," *arXiv preprint arXiv:2204.05177*, 2022.

[16] Zhixi Cai, Kalin Stefanov, Abhinav Dhall, and Munawar Hayat, "Do you really mean that? content driven audio-visual deepfake dataset and multimodal method for temporal forgery localization," *arXiv preprint arXiv:2204.06228*, 2022.

[17] Tianyun Liu, Diqun Yan, Nan Yan, and Gang Chen, "Anti-forensics of fake stereo audio using generative adversarial network," *Multimedia Tools and Applications*, vol. 81, no. 12, pp. 17155–17167, 2022.

[18] Biaoli Tao, Rangding Wang, Diqun Yan, and Chao Jin, "Anti-forensics of double compressed mp3 audio," *International Journal of Digital Crime and Forensics (IJDCF)*, vol. 12, no. 3, pp. 45–57, 2020.

[19] Qi Yan, Rui Yang, and Jiwu Huang, "Detection of speech smoothing on very short clips," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 9, pp. 2441–2453, 2019.

[20] Xiaowen Li, Diqun Yan, Li Dong, and Rangding Wang, "Anti-forensics of audio source identification using generative adversarial network," *IEEE Access*, vol. 7, pp. 184332–184339, 2019.

[21] Guang Hua, "Error analysis of forensic enf matching," in *IEEE International Workshop on Information Forensics and Security (WIFS)*, 2018, pp. 1–7.

[22] Guang Hua, Ying Zhang, Jonathan Goh, and Vrizlynn LL Thing, "Audio authentication by exploring the absolute-error-map of enf signals," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 5, pp. 1003–1016, 2016.

[23] Zhisheng Lv, Yongjian Hu, Chang-Tsun Li, and Bei-bei Liu, "Audio forensic authentication based on mocc between enf and reference signals," in *IEEE China Summit and International Conference on Signal and Information Processing*, 2013, pp. 427–431.

[24] Catalin Grigoras, "Applications of enf criterion in forensic audio, video, computer and telecommunication analysis," *Forensic science international*, vol. 167, no. 2-3, pp. 136–145, 2007.

[25] Avery Wang, "An industrial strength audio search algorithm," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2003, pp. 7–13.

[26] Shumeet Baluja and Michele Covell, "Audio fingerprinting: Combining computer vision & data stream processing," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2007, vol. 2, pp. 213–216.

[27] TJ Tsai, Thomas Prätzlich, and Meinard Müller, "Known-artist live song identification using audio hashprints," *IEEE Transactions on Multimedia*, vol. 19, no. 7, pp. 1569–1582, 2017.

[28] Claire Chang, Thaxter Shaw, Arya Goutam, Christina Lau, Mengyi Shan, and TJ Tsai, "Parameter-free ordered partial match alignment with hidden state time warping," *Applied Sciences*, vol. 12, no. 8, pp. 3783, 2022.

[29] Gautham J Mysore, "Can we automatically transform speech recorded on common consumer devices in real-world environments into professional production quality speech?—a dataset, insights, and challenges," *IEEE Signal Processing Letters*, vol. 22, no. 8, pp. 1006–1010, 2014.