POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

DeepMetric: Enhancing Synthetic Speech Detection through Support Tracks Generation

Laurea Magistrale in Music and Acoustic Engineering

Author: Alessandro Orsatti

Advisor: Prof. Paolo Bestagini

Co-advisors: Davide Salvi, Daniele Ugo Leonzio

Academic year: 2022-2023

1. Introduction

The extensive uptake of Artificial Intelligence (AI)-powered tools has opened up a new era of creative possibilities, offering advantages to both prominent tech corporations and ordinary individuals, largely due to their enhanced accessibility.

These aforementioned tools enable the generation of synthetic media with minimal effort. Textual inputs can be transformed to mimic someone's voice or produce digital content that closely resembles authentic media. While these technologies have numerous positive applications such as dubbing and medical purposes, they also pose risks to public information and privacy.

Deepfakes, a form of synthetic content created through Deep Learning (DL) techniques, are particularly concerning. They involve appropriating the physical characteristics of a real person to generate new, often misleading content. This aggravates the issues surrounding fake news, as deepfakes can be used to manipulate public perception and credibility, especially when attributed to public figures like politicians. The ease of creating deepfakes has led to a surge in fake news problems, worsened by the rapid consumption of information in today's society. The problems related to these new public issues are addressed by the MultiMedia Forensic (MMF) community, which guides the research towards detecting deepfakes for public and private security purposes. Despite the results and methods utilized for deepfake spoofing have led to a more comfortable situation concerning the specific field, several difficulties, such as predictability of the detection on unseen data, are still faced by the MMF community. This is the inspiring factor behind the decision and development of this thesis's main topic: Deepfake Audio Detection [4].

Old architectures for spoofing were based on signal processing techniques. Nowadays modern detectors are based on DL, varying from simple architectures composed only by Convolutional Neural Networks (CNNs) to more complex systems which can take as input raw waveform (RawNet [2]). DL-based architectures are way more precise in a detection scenario because of their ability to capture patterns in the input and recognize them as deepfakes. Although present and functioning, those architectures are far from being perfect, given the constant introduction of new synthetic speech generators. These new

generators were not seen during training by the network, so they provide, in the resulting deepfake, unseen artifacts capable of tricking the network. With new generators rapidly becoming available, each producing distinct artifacts in the audio, maintaining consistency and robustness against unfamiliar models necessitates experimentation with varied solutions to address unpredictability. To address this problem, the system proposed in this thesis, named DeepMetric, aims to make a detector reliable and consistent by utilizing speech deepfakes synthesized with new generative models, among which several Text-to-Speech (TTS) and Voice Conversion (VC), never seen during training. The main idea is that, instead of considering the output of the detector, we generate *support tracks* with the same textual and semantic content of the original track and calculate the distance in the latent space, which is used as a discriminant to make detection and improve performance. Distance between feature vectors in the latent space is highly descriptive for finding similarities between the inputs of the model. By comparing the embedding derived from the original input track with those of the deepfakes, we can discern whether the input has been artificially generated. All the deepfake copies contain the same textual component of the original track and the embeddings used to compute the distance are extracted for each track from the penultimate layer of the neural network and represent high-level, non-descriptive features of its input. This method is thought not only to contrast unseen architectures by directly comparing them to the original audio in the latent space but also to put emphasis on a modular setup, allowing for the incorporation of future architectures. Importantly, this approach is specifically implemented during the test phase, ensuring its compatibility with any DL-based detector and any TTS or VC model without being bound to a specific architecture.

2. Proposed method

The problem we address is formally defined as follows. Let us consider a discrete-time input speech signal $\mathbf{x}$ sampled with a sampling frequency f_s. The signal belongs to a class $y \in \{0, 1\}$, where 0 means that the signal is authentic while 1 indicates that has been generated with any speech synthesis technique.
Our goal is to develop a speech deepfake detector $\mathcal{D}$ trained to estimate the class of the signal $\mathbf{x}$ as $\hat{y} = \mathcal{D}(\mathbf{x})$, where $\hat{y} \in [0, 1]$ is a real number indicating the likelihood that the signal $\mathbf{x}$ is fake.
In this thesis we propose a method to address the problem of lack of accuracy over data generated with unseen architectures. Given a speech deepfake detector $\mathcal{D}$, during its inference phase, for each track $\mathbf{x}$ under analysis, we generate a set of *support tracks* $\mathbf{x}_i$ with $\mathbf{i} \in \{1, N\}$ that contain the same textual and speaker information of the original signal $\mathbf{x}$. This is achieved by considering a set of N synthetic speech generators that leverage TTS and VC techniques. The TTS component of the generator is used to synthesize a signal $\mathbf{x}_i$ with the same textual content as $\mathbf{x}$, while the TTS is used to modify the speaker of $\mathbf{x}_i$ and make it equal to that of $\mathbf{x}$. The N support tracks are used to make the detection process of $\mathcal{D}$ more robust. To do so, we consider the detector $\mathcal{D}$ as embedding extractor and use it to compute a feature vector $\mathbf{e}$ for both the original track and the generated ones. Finally, we consider as detector score, no more the output $\hat{y}$ of the detector $\mathcal{D}$ but the thresholded average of the distances $\mathbf{d}_i$ with $\mathbf{i} \in \{1, N\}$ between the embeddings $\mathbf{e}_i$ of the analyzed *support tracks* and $\mathbf{e}$ of the original signal.
The rationale behind this method is that the distance $\mathbf{d}_i$ in the latent space between the feature vectors of the original signal $\mathbf{x}$ and the *support tracks* $\mathbf{x}_i$ is more robust than the simple output $\hat{y}$ of the detector.
We highlight that the proposed system involves only the inference phase of the model and does not affect training. Furthermore, it is independent from both the synthetic speech generation and detection models utilized and the number N of *support tracks*. These aspects make the method applicable to any detector $\mathcal{D}$ and capable of being improved when new technologies appear on the market.
In Figure 1 is presented the flow diagram of the system on which all the method is based. To summarize, here are the three main phases of the proposed methodology outlined in this thesis:

1. **Generation Phase:** This involves creating N deepfakes copies (namely *support tracks* $\mathbf{x}_i$), with the same textual informa-

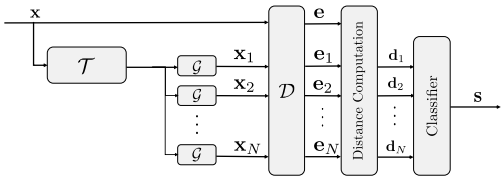

Figure 1: Pipeline of the proposed method.

tion, of the audio track $\mathbf{x}$ to be detected.

2. **Embeddings extraction Phase:** Embeddings $\mathbf{e}_i$ with $\mathbf{i} \in \{1, N\}$ for all *support tracks*, including $\mathbf{e}$ for the original $\mathbf{x}$, are extracted from the penultimate layer of a neural detector $\mathcal{D}$.
3. **Metric Comparison Phase:** Each embedding $\mathbf{e}_i$ from the *support tracks* is utilized to calculate its distance from the embedding $\mathbf{e}$ of the original track $\mathbf{x}$ using various metric functions. The resulting metric scores $\mathbf{d}_i$ are then averaged and thresholded to generate the final detection score $\mathbf{s}$.

3. Experimental setup

3.1. Generation pipeline

3.1.1 Automatic Speaker Recognition

Automatic Speaker Recognition (ASR), in a synthetic generation pipeline, takes the job of extracting the textual content of the original track to be fed to a Text-to-Speech model. For what concerns our work, the ASR model assigned to text extraction from the original track is Whisper. Whisper is a robust speech recognition model trained on an extensive collection of varied audio data. Several models for as many languages are available. Finally, we chose the base model as it guaranteed the perfect balance between speed and accuracy.

3.1.2 Text-to-Speech

A complete TTS models can be either composed of a combination of Spectrogram Generator and Vocoder trained on the same specific set of data, or an end-to-end model. All the models utilized in the generation pipeline were taken from the coqui.ai API and are all trained on a single language (in our case we took the ones trained on English data). For the generation pipeline we utilized 5 different TTS models, varying existing Spectrogram Generator and Vocoder combinations.

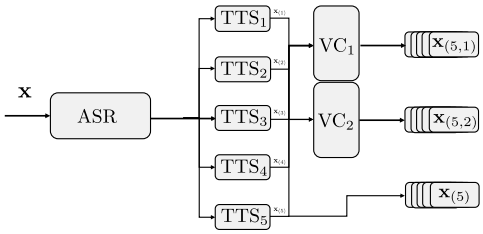

Figure 2: Generation pipeline utilized for the experiments conducted in this thesis work.

3.1.3 Voice Conversion

In this work we decided to opt for two different VC models to add variety to the *support tracks* and expand the types of generated artifacts present in them. The first model $\mathbf{VC}_1$ is provided, as for the TTS models, from the coqui.ai API in the form of a pre-trained model called FreeVC. FreeVC is a one-shot model that only needs one reference track in order to perform Voice Conversion. The second model $\mathbf{VC}_2$ takes a different approach over VC and is called kNN-VC, an any-to-any VC model. Any-to-any VC aims to transform source speech into a target voice with just a few examples of the target speaker as a reference.

3.1.4 Flow Diagram

As a summary, the whole generation pipeline utilized for the experiments in this thesis work is the one depicted in Figure 2. In the experiments carried out for this thesis work we implemented a pipeline composed of an ASR model extracts the text from the original audio, 5 TTS models that receive the text as an input and generate 5, and 2 VC models which each receive all TTS output tracks. The 5 tracks from the TTS output are also retained in order to have, as a result, 15 *support tracks*.

3.2. Dataset

The dataset chosen to develop and test the idea presented in this thesis is ASVspoof2019 [3]. The dataset is composed of both real tracks and deepfakes generated with various algorithms.

For the development of this thesis, we chose to use only the logical access (LA) section of the dataset. A second dataset, namely Fake of Real (FoR) [1], has been used for a further experiment with the purpose of testing the proposed method in an open-set scenario.

3.3. Detector

In this work we employed a single DL architecture for the detector utilized to extract embeddings and predictions for the proposed techniques. RawNet2 [2] processes raw audio and possesses the capability to learn cues that conventional countermeasure solutions may not detect. With respect to its original implementation, the input size has changed from 64000 to 32000, that is because we maintained a consistent sampling rate of 16kHz across all experiments and datasets and, after experimenting with various length, we chose a window length of 2 seconds.

3.4. Training procedure

The proposed detector has been trained with the ASVspoof2019 *train* and validated with the *dev* partition. As stated above 3, each audio input has been cut with a window of 2 seconds in a random position along the usable length. In the event of an audio shorter than 2 seconds the input is padded to the correct input length for the model. To maintain the training procedure unbiased, every batch uses the same amount of real and fake tracks. The parameters of the training are **Learning Rate** equal to 0.0001, **Epochs** equal to 50, **Batch Size** equal to 128, **Window Length** equal to 2 seconds. **Loss Function** is Cross-Entropy and **Optimizer** is Adam. It is important to underline that for the Cross-Entropy loss function, has been taken the possibility to take into account uncertainties in the dataset through the use of label-smoothing. Label smoothing is a regularization technique that addresses issues related to both overfitting and overconfidence. In our case label-smoothing has been set equal to 0.3.

3.5. Metric operations

The embeddings $\mathbf{e}_i$ with $\mathbf{i} \in \{1, 15\}$ relative to the 15 generated *support tracks* and the one for original track $\mathbf{e}$ are extracted from the penultimate layer of the RawNet2, a dense fully-connected layer composed of 1024 neurons. We then need to assess the distance $\mathbf{d}_i$ with $\mathbf{i} \in \{1, 15\}$ in the latent space between each of the *support tracks* embeddings $\mathbf{e}_i$ and those of the original track $\mathbf{e}$. This distance is evaluated using 3 different metric functions, which are Cosine Distance, Euclidean Distance and Manhattan Distance. As an added experiment the resulting feature vectors will be separately fed into the Machine Learning (ML) classifier to predict the final score.

3.6. Nearest Centroid Classifier

For what concerns the ML algorithm for classification, we decided to go with *Nearest Centroid*. It works by representing data in the latent space and assigning to the input the label of the class of training samples whose mean (centroid) is closest to it. In our experimental scenario, the *Nearest Centroid* algorithm takes as input a feature vector composed of the 15 distances extracted, always in the same order with respect to the generation models.

4. Results

For all the Tables present in this section results in bold underline the best performance of the metrics with regards to the predictions.

4.1. Support Tracks

As a preliminary experiment, it is interesting to state how our generated *support tracks* behave in a detection scenario with our detector $\mathcal{D}$ trained on ASVspoof2019. Figure 3 shows the output of the last layer of our implementation of the RawNet2 fed with the *support tracks*. As we can clearly see, the majority of the tracks are classified correctly, 1 being fake and 0 real, although several tracks were classified as real, tricking the network. This is interesting because in this way we are adding variety to our generated latent space, and so the possibility of taking advantage of the distances as a measure of authenticity.

4.2. Distance Metrics

Here we report our result on the synthetic speech detection task based on the outputs of the metric computation phase. We proceeded in the following way: we took all the tracks from the ASVspoof2019 *eval* partition (71237 original tracks), we tested them with our detector to

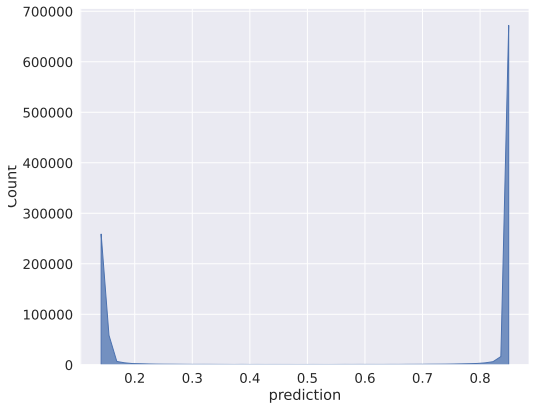

Figure 3: Histogram representing the score distribution for the *support tracks* tested with our detector $\mathcal{D}$.

	AUC	EER
Prediction	0.827	0.228
Cosine Distance	0.802	0.275
Euclidean Distance	**0.856**	**0.226**
Manhattan Distance	0.842	0.234

Table 1: Evaluation metrics for the experiment conducted on the ASVspoof2019 dataset.

extract the predictions, generated the *support tracks* and extracted the distances, with each metric function, between them and the original in the latent space. Then we computed the average between the 15 values of the 3 resulting feature vectors, thresholded them and assigned the pseudo-score for the individual metric function, which represents the final result of our pipeline. As we can see, the graph in Figure 4 shows promising results in terms of performance improvements of the proposed method compared to a simple detector. The score that performs the best is the Euclidean Distance, soon followed by the Manhattan Distance. Cosine Distance, in this form of experiment, performs slightly worse than the predictions of the detector. In Table 1 there are the results of the Area Under Curve (AUC) for the experiment, along with the Equal Error Rate (EER).

4.3. Classifier

For the added experiment of our evaluation of the proposed method we tested the performances of *Nearest Centroid* algorithm in classi-

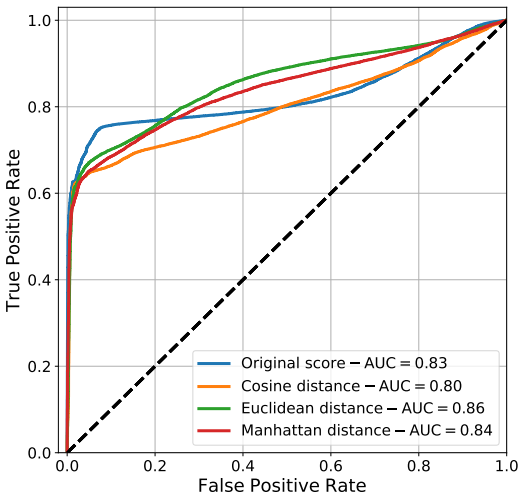

Figure 4: Receiver Operating Characteristic (ROC) AUC for the experiment conducted on the ASVspoof2019 dataset.

	AUC	EER	Bal. Acc.
Prediction	0.827	0.228	0.805
Cosine Distance	0.805	0.240	0.805
Euclidean Dist.	0.810	0.280	**0.810**
Manhattan Dist.	0.809	0.270	0.809

Table 2: Evaluation metrics for the added experiment conducted on ASVspoof2019 with the ML Classifier.

fying the original signals. The input for the ML algorithm is a feature vector of size 15 composed of the distances $\mathbf{d}_i$ per original track. The output is directly the predicted class. Half of the input feature vectors relative to the original tracks were used for fitting the model, the other half to test it. In Table 2 are presented the evaluation metrics for each metric function, along with the one for the simple predictions. The results underline a different behavior with respect to the previous experiments. Here we can see that the AUC evaluation metric in Table 2 is not as high as the one in Table 1 and is similar across all metric functions.

4.4. Generalization Test

In this test we evaluate our proposed system with a different dataset to extend the results to a more generalized scenario. The test has been conducted with both the original method

	AUC	EER
Prediction	0.489	0.511
Cosine Distance	**0.540**	**0.449**
Euclidean Distance	0.495	0.502
Manhattan Distance	0.463	0.528

Table 3: Evaluation metrics for the added experiment conducted on a different dataset.

	AUC	EER
Prediction	0.489	0.511
Cosine Distance	0.543	**0.173**
Euclidean Distance	0.541	0.211
Manhattan Distance	**0.563**	0.257

Table 4: Evaluation metrics for the added experiment conducted on a different dataset with the ML Classifier.

and the one with the added classification step. Table 3 and 4 show the results for the test performed on the FoR dataset with respectively the original pipeline and with the *Nearest Centroid* Classifier. It is interesting to underline that the trend with the results is opposite to the one of the previous experiments on the ASVspoof2019 dataset. Here, indeed, we found out that, besides performing better than the original predictions in any scenario, the method relying on the ML classifier performs better than the one that does not employ it. Furthermore, in both experiments we can see an improvement of the method proposed in almost all results, with respect to the prediction of the classifier.

5. Conclusions

In this work we presented a new audio forensics approach suitable for audio deepfake detection, named DeepMetric, capable of detecting signals synthesized with the speech generation techniques of TTS and VC. The three main phases of the proposed method are: Generation Phase, Embedding extraction Phase and Metric Comparison Phase. In the generation phase we synthesize *support tracks* by taking advantage of existing TTS and VC models. During the second phase a detector is utilized as a feature extractor by taking as input the *support tracks* together with the original signal and giving as output the embeddings from its penultimate layer. For the last phase, the distances between each generated track and the original signal is computed in the latent space constructed with the aforementioned embeddings, the distances are employed to compute a final score for the prediction. The main benefit of our method is that it does not rely on specific detection or generation models, but it is, instead, modular and capable of being easily upgradable with time. To perform a complete evaluation, we have tested the system with both the dataset used to train the detector, which is ASVspoof2019, and a different dataset named FoR. We can still apply many improvements to the proposed methodology. The first one focuses on improving the Embedding extraction Phase by employing different architectures for the detector. The second one requires finding an optimal classifier to exploit completely the distance feature vector obtained through embedding comparison with each *support tracks*. The third one is to find a strategy to optimize the TTS and VC model selection, therefore avoiding slowdowns due to unnecessary *support tracks*.

References

[1] Ricardo Reimao and Vassilios Tzerpos. For: A dataset for synthetic speech detection. In *2019 International Conference on Speech Technology and Human-Computer Dialogue (SpeD)*, 2019.

[2] Hemlata Tak, Jose Patino, Massimiliano Todisco, Andreas Nautsch, Nicholas Evans, and Anthony Larcher. End-to-end anti-spoofing with rawnet2. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.

[3] Xin Wang, Junichi Yamagishi, Massimiliano Todisco, Héctor Delgado, Andreas Nautsch, Nicholas Evans, Md Sahidullah, Ville Vestman, Tomi Kinnunen, Kong Aik Lee, et al. Asvspoof 2019: A large-scale public database of synthesized, converted and replayed speech. *Computer Speech & Language*, 64:101114, 2020.

[4] Jiangyan Yi, Chenglong Wang, Jianhua Tao, Xiaohui Zhang, Chu Yuan Zhang, and Yan Zhao. Audio Deepfake Detection: A Survey. *arXiv preprint arXiv:2308.14970*, 2023.