The Automatic Speaker Verification Spoofing Countermeasures Workshop (ASVspoof 2024)
31 August 2024, Kos, Greece

Robust Audio Deepfake Detection: Exploring Front-/Back-End Combinations and Data Augmentation Strategies for the ASVspoof5 Challenge

Karla Schäfer⋆, *Matthias Neu*†, *Jeong-Eun Choi*⋆

⋆ Fraunhofer Institute for Secure Information Technology, ATHENE
† Federal Office for Information Security (BSI)
{karla.schaefer, jeong-eun.choi}@sit.fraunhofer.de, matthias.neu@bsi.bund.de

Abstract

The robustness and generalizability of audio deepfake detectors are becoming more important due to the technical advances in generation methods and the widespread usage of audio deepfakes. The ASVspoof5 challenge addresses this by providing a new dataset. This paper presents Fraunhofer SIT's anti-spoofing detectors submitted to the ASVspoof5 challenge. AASIST(-L), RawGAT-ST and data augmentation was used in the closed condition. In the open condition, we evaluated different SSL-based front-ends using diverse training data. The results indicate that the utilisation of extensive data augmentation improve the results when using a non SSL-based front-end, whereas its incorporation with an SSL-based front-end led to a decline in performance. The implementation of a large SSL front-end improved the result. Our best detector in the closed setting attained a min DCF of 0.589 and the best in the open condition (using SSL) a min DCF of 0.174 on the ASVspoof5 evaluation set.

1. Introduction

As the amount of AI-generated audio content on social media rapidly increases, there is a growing necessity to develop detectors that can distinguish between AI-generated and genuine audio. Tools like ElevenLabs or RVC (Retrieval based Voice Conversion) allow people with minimal background knowledge to generate audio deepfakes, which decrease the barrier for generating audio deepfakes and consequently increase the potential misuse of speech generation technologies. An important factor of these detectors is their ability to generalize on new generation methods and to be robust against adaptive or adversarial attacks, i.e. in real-world scenarios. One commonly used test set for evaluating detectors in their ability to detect real-world data is the in-the-wild dataset [1]. It contains real and spoofed audio recordings of English-speaking celebrities and politicians. Using the in-the-wild dataset as test set, mainly self-supervised learning (SSL)-based detectors proved to be the best performing methods on such real-world data. For example, [2] developed the SOTA detector on the in-the-wild dataset achieving an EER of 4.07%. Their model consists of an SSL-based front-end and neuronal network back-end as classifier.

In the ASVspoof5 challenge, a new dataset is introduced with the goal of developing a detector with greater generalizability and robustness. As source data of the ASVspoof5 dataset, the Multilingual Librispeech (MLS) dataset [3] in its English-language subset was used. It contains speech recordings collected from numerous speakers in various recording conditions. As generation methods, different text-to-speech (TTS) and voice conversion (VC) methods were used. The generation methods are, at the time of writing this paper, unknown. The participants are asked to develop a detector under open and closed condition. For creating our detectors, prior to the challenge, we conducted some preparatory experiments on RawNet2 [4], RawGAT-ST [5], AASIST(-L) [6] and PC-DARTS [7] to evaluate which detector structure performs best. For this, we trained different front-end and back-end combinations on two different datasets. The best combination was then used in the closed condition of the ASVspoof5 challenge. As SSL models currently achieve the best results when faced with generalizability, in the open condition, we tested different SSL models as feature extraction method. As back-end we used the best classifier from the closed condition, being AASIST. In both conditions, we applied data augmentation with the aim to increase the training set and to improve the robustness of the detectors against adaptive attacks.

In the following, first, the task of the challenge is described shortly (Section 2). In Section 3, our methods consisting of feature extraction, classifier and data augmentation methods are presented. In Section 4 the experiments are presented, followed by the results (5) and the conclusion (Section 6).

2. Task Description

The ASVspoof5 challenge aims to generate generalizable and robust countermeasures, following the previously conducted ASVspoof2021 challenge. Therefore, the detectors are expected to perform reliably even on utterances generated with new or unseen spoofing attacks, recordings with non-studio-quality and post-processed data. The challenge was conducted in two conditions: the closed and open condition.

In the closed condition only the training partition of the provided ASVspoof5 was allowed to be used. Additionally, it was not allowed to use pre-trained models. In the open condition, participants were allowed to use external data and pre-trained models. Hereby, any data contained within LibriLight [8], Multilingual Librispeech (MLS) English datasets [3], and the MUSAN corpus speech subset [9] were restricted. The restriction also applied to any other dataset, pre-trained model, or any source derived from these datasets.

3. Methods

We analysed different combinations of front-end feature extraction methods and back-end classifier. Furthermore, we applied different data augmentation techniques. For an overview of the process see Figure 1.

10.21437/ASVspoof.2024-9

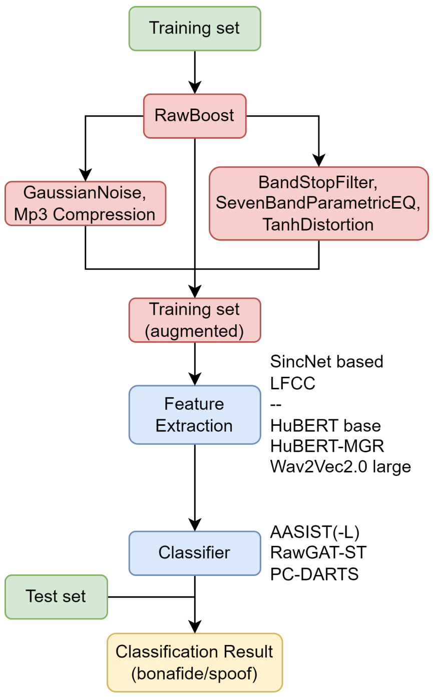

Figure 1: *Overview of our experiments conducted (Red: Data Augmentation).*

3.1. Feature Extraction

Depending on the condition of the challenge, we used different feature extraction methods. In the open challenge, we were able to use SSL-based front-ends. As only front-ends not pre-trained on MLS or other databases sourced from LibriVox audiobooks [10] were allowed, we choose to use HuBERT base [11], HuBERT-MGR [12] (hubert_base_robust_mgr) and Wav2Vec2.0 large [13] (wav2vec2_large_960), all pre-trained on the LibriSpeech [14] corpora.

In the closed condition, no pre-trained models were allowed. In the preparatory tests we examined RawNet2, AASIST(-L), RawGAT-ST and PC-DARTS. For RawNet2, AASIST(-L) and RawGAT-ST, the feature extraction methods are based on SincNet. SincNet [15] is a CNN which can be applied on the raw waveform. It is based on parametrized sinc functions, which implement band-pass filters. In contrast to standard CNNs, that learn all elements of each filter, only low and high cutoff frequencies are directly learned from data with the proposed method. In this way, SincNet encourages the first convolutional layer to discover more meaningful features. Additionally, in our preparatory tests we used linear-frequency cepstrum coefficients (LFCC), based on the implementation provided by the authors of PC-DARTS.

3.2. Classifier

As back-ends, in our preparatory analysis, we applied RawNet2, RawGAT-ST, AASIST(-L) and PC-DARTS. In the challenge, we applied the best performing models of our preparatory analysis, being AASIST(-L) and RawGAT-ST. In the following, a short explanation of the models used are given.

RawNet2 is based on the RawNet2 architecture, modified to be applied on anti-spoofing [4]. New to RawNet2 is the application of filter-wise feature map scaling (FMS) using a sigmoid function applied to residual block outputs. The embedding dimension for RawNet2 is also greatly increased, from 128 for RawNet1 to 1024 for RawNet2.

RawGAT-ST is based on the idea that artefacts for distinguishing bona fide from spoofed speech reside in specific subbands and temporal segments [5]. Their detector uses a spectro-temporal graph attention network (GAT) to learn the relationships between cues spanning different sub-bands (spectral and temporal) and temporal intervals.

AASIST is based on RawGAT-ST and focuses also on the idea to build one system for detecting artefacts in both, spectral and temporal intervals [6]. Its lightweight variant with only 85k parameters is AASIST-L, AASIST uses the RawNet2-based encoder (based on SincNet) for extracting high-level feature maps from raw input waveform.

PC-DARTS applies differentiable architecture search (DARTS) on the deepfake detection task [7]. PC-DARTS consist of two steps: 1) the architecture search, were given a training set the optimal architecture is determined by optimising the structure of a stack of two normal cells and one reduction cell, and 2) the training of the found architecture. The cell architecture is fixed in the first step, in the second step only the network parameters are optimized. With PC-DARTS, the authors were able to achieve substantial savings in terms of computing complexity and memory.

3.3. Data Augmentation

For data augmentation, two different method types were alternated and combined applied on the training data. First, RawBoost [16] was applied using a series of linear and non-linear convolutive noise and impulsive signal dependent noise (algo: 5 in all settings), see also the work of [17]. Secondly, using the audiomentations library[1], the data augmentation techniques adding gaussian noise (p=0.1) and mp3 compression (p=0.5) were applied.

Furthermore, in each condition, on our best performing detector structure, we applied BandStopFilter, SevenBandParametricEQ and TanhDistortion from the audiomentations library on the training set. Hereby, we concatenated the existing training set with augmented (p=0.6) versions of the same dataset using the three augmentation types. We hoped to increase and diversify the training set with this approach. With BandStopFilter we aimed to prevent the model from overfitting to specific frequency relationships, helping to make the model robust to diverse audio environments and scenarios where frequency losses can occur. With SevenBandParametricEQ we tried to make the model more robust to various frequency spectrums and TanhDistortion was included for adding harmonics.

4. Experiments

Previously to the challenge we performed some preparatory tests using the ASVspoof 2019 LA [18] dataset and WaveFake [19] with its genuine samples of the LJSpeech corpora [20]. The ASVspoof 2021 LA and DF splits [21] were developed to evaluate the robustness and generalizability of detectors. In our preparatory analysis, we used them as test sets. Furthermore, we used the in-the-wild dataset [1] to evaluate the generalizability of the models on unseen real-world attacks. The best model combinations of these preparatory tests were then used during the challenge in the closed setting using the ASVspoof5 dataset as training set.

[1]https://iver56.github.io/audiomentations/

4.1. Datasets

ASVspoof 2019 is based upon the VCTK corpus, with a set of 107 speakers. In the Logical Access (LA) scenario, for the training set, spoofed utterances were generated using four TTS and two VC methods. With this, the training set consists of 2,580 bona fide and 22,800 spoofed utterances. As second training set, we used **WaveFake** and its genuine counterpart LJ Speech [20]. The WaveFake [19] dataset contains 117,985 spoofed samples. Mostly, the dataset is based on the English LJSpeech data set [20]. Additionally, the JSUT data set was used, containing Japanese speech samples. The LJSpeech dataset consists of 13,100 short audio clips with an average of 6 second recordings each (24 hours in total) read by a female speaker. The LJSpeech dataset was used here to represent genuine samples. In the closed setting of the ASVpoof5 challenge, we only used the **ASVspoof5** training set. In the open condition, we used a combination of the WaveFake (+LJ Speech), LibriSeVoc [22], ASVspoof 2019 LA, Fake-or-Real (FoR) [23] and ASVspoof5 datasets. The **Fake-or-Real (FoR)** dataset [23] contains genuine and spoofed audio samples from seven SOTA TTS models. The genuine data was sourced from YouTube and open-source datasets like Arctic Dataset, LJSpeech and VoxForge. Four different versions of the FoR dataset exist: FoR-original, FoR-norm, FoR-2seconds and FoR-rerecorded. We used the FoR-original version. FoR-original contains the files as collected from the speech sources, without any modification or class/gender balancing. The **LibriSeVoc** Dataset [22] was created for vocoder artefact detection and contains 13,201 genuine audio samples and 79,205 spoof audio samples generated through self-vocoding by 6 vocoders, each represented with 13,201 samples. The dataset contains 34.92 hours of genuine audio with samples of audio lengths from 5 seconds to 34 seconds at 24kHz.

4.2. Implementation Details

In the preparatory tests and the experiments in the closed condition, the original implementations of the models RawNet2, RawGAT-ST, AASIST(-L)[2] and PC-DARTS[3] were used. In the open condition we used SSL-based feature extraction. For this, we modified the code of [17][4], using an AASIST classifier together with SSL-models as front-end. For the SSL-based front-end implementation, we used the S3PRL library[5].

For all experiments, we applied a weighted sampler. Therefore, we made sure that each dataset contains 50% of bona fide samples. The loss was calculated using cross-entropy loss for all back-ends. All recordings used for training were trimmed and padded to approx. 4 seconds of audio clips. The non-SSL-based detectors were trained for 70 epochs, with a batch size of 16. PC-DARTS in the architecture search phase was trained for 50 epochs, as proposed in the paper. The architecture with the highest development and training accuracy was taken for training the final classifier. The SSL-based detectors were trained for a number of epochs depending on the training set used. The numbers of epochs were calculated based on the length of the train loader (100/(len(train_loader)/4000)). Depending on the training set, this resulted in 15 (ASVspoof5, ASVspoof 2019 LA, WaveFake+LJ, LibriSeVoc, FoR) to 35 (ASVspoof5) epochs. The batch size was set to 16.

[2] `https://github.com/clovaai/aasist/tree/main?tab=readme-ov-file`

[3] `https://github.com/eurecom-asp/pc-darts-anti-spoofing/tree/main`

[4] `https://github.com/TakHemlata/SSL_Anti-spoofing`

[5] `https://s3prl.github.io/s3prl/tutorial/upstream_collection.html`

4.3. Evaluation Metrics

In our preparatory analysis we used the commonly used Equal Error Rate (EER) as evaluation metric. In the challenge, we used the provided CodaLab platform for calculating the minimum/actual detection cost function (min/act DCF), the cost of log-likelihood ratio (Cllr) and the EER.

5. Results

In our analysis we tested different front-end and back-end combinations, whereby the front-end consisted of either an SSL-based feature extraction method or a LFCC/SincNet based feature extraction. All results were obtained giving the whole sample as test data (no trimming of the test data). Data augmentation with RawBoost was used in all the experiments. In a preparatory analysis, we tested different non-SSL-based detectors. The best method was used in the closed condition of the ASVspoof5 challenge. For the open setting, we applied AASIST, being the best model in the open condition, with different SSL-based front-ends.

5.1. Preparatory Tests

In the preparatory tests, for non-SSL based detection, we evaluated two different front-ends, SincNet and LFCC using two different training sets (ASVspoof 2019 LA and WaveFake+LJ). As classifier, we tested AASIST, AASIST-L, RawNet2, RawGAT-ST and PC-DARTS. With these experiments, we wanted to find out which combination of (non-SSL-based) front-end and back-end lead to the best results, using different training data. As test sets, we used the in-the-wild dataset and the ASVspoof2021 LA and DF test splits. Additionally, we applied data augmentation for enhancing our training set, with the aim to make the detector more robust against data processing such as mp3 compression. In Table 1 the results of the non-SSL-based detectors trained with the ASVspoof 2019 LA or the WaveFake dataset and tested on the in-the-wild dataset and the ASVspoof 2021 LA/DF split are given.

As one can see in Table 1, when viewing the results on the in-the-wild dataset, the results when using WaveFake were superior of them using ASVspoof 2019 LA as training data. When viewing the ASVspoof 2021 LA and DF test sets it was the training set ASVspoof 2019 LA. For example, the best EER using the ASVspoof 2019 LA partition as training data was RawGAT-ST with a SincNet front-end, with an EER of 18.08%. When using the WaveFake dataset as training data, the RawGAT-ST with SincNet front-end achieved an EER of 13.07% on the in-the-wild dataset. But, using WaveFake as training data, AASIST with the SincNet front-end performed even better with an EER of 11.83%, being the best detector on the in-the-wild dataset in this table. Using the training set ASVspoof 2019 LA provided the superior results on the ASVspoof 2021 LA and DF splits, which is not surprising as the ASVspoof 2021 LA dataset is based on the ASVspoof 2019 LA partition and the ASVspoof 2021 DF partially on ASVspoof 2019 LA. The best EERs of 20.72% (ASVspoof 2021 LA -all) and 19.64% (ASVspoof 2021 DF-all) were here achieved with AASIST and the SincNet front-end. Using the WaveFake dataset as training data only scores of up to 38.10% (ASVspoof 2021 LA -all) and 28.74% (ASVspoof 2021 DF-

Table 1: *Results in the preparatory analysis on non-SSL-based detectors using the ASVspoof 2019 LA and WaveFake as training data (data augmentation: GaussianNoise, Mp3 compression). The best EERs per training set are marked in grey.*

Front-end	**Back-end**	**Training Data**	**EER (%) in-the-wild**	**EER (%) ASVspoof 2021 LA all/eval/progress/hidden**	**EER (%) ASVspoof 2021 DF all/eval/progress/hidden**
SincNet	AASIST	ASVspoof 2019 LA	23.55	20.72/ 20.81/ 19.75/ 20.57	19.64/ 20.07/ 19.00/ 18.99
SincNet	AASIST-L	ASVspoof 2019 LA	26.86	26.07/ 26.12/ 24.82/ 25.81	25.17/ 25.32/ 24.29/ 24.39
SincNet	RawGAT-ST	ASVspoof 2019 LA	18.08	21.39/ 21.39/ 21.06/ 21.90	21.43/ 21.85/ 21.56/ 21.37
SincNet	RawNet2	ASVspoof 2019 LA	34.72	23.71/ 23.14/ 22.85/ 26.23	26.73/ 27.06/ 22.53/ 25.90
LFCC	PC-DARTS	ASVspoof 2019 LA	38.95	33.80/ 33.07/ 31.62/ 40.36	35.26/ 36.27/ 29.94/ 36.92
SincNet	AASIST-L	WaveFake + LJ	14.51	38.10/ 38.09/ 37.41/ 39.03	28.74/ 27.25/ 36.46/ 38.29
SincNet	AASIST	WaveFake + LJ	11.83	41.49/ 41.14/ 41.04/ 44.90	32.14/ 30.92/ 39.79/ 43.49
LFCC	AASIST	WaveFake + LJ	54.94	50.95/ 50.85/ 50.84/ 51.38	50.17/ 48.23/ 50.28/ 52.02
SincNet	RawGAT-ST	WaveFake + LJ	13.07	40.97/ 40.75/ 40.57/ 42.80	31.86/ 30.40/ 39.58/ 41.62
LFCC	RawGAT-ST	WaveFake + LJ	56.10	48.83/ 48.78/ 48.39/ 49.64	47.91/ 45.71/ 46.24/ 47.98
SincNet	RawNet2	WaveFake + LJ	23.14	41.45/ 41.37/ 40.81/ 42.86	37.10/ 36.67/ 40.07/ 41.12
LFCC	PC-DARTS	WaveFake + LJ	43.11	45.71/ 45.98/ 45.65/ 44.70	33.80/ 33.07/ 31.62/ 40.36

all) were achieved, using AASIST-L (a light weighted version of AASIST) and a SincNet front-end. Regarding the front-ends, LFCC performed rather poor compared to SincNet. For example, with the WaveFake training data using AASIST and comparing the performance of it with a LFCC and a SincNet front-end, with LFCC an EER of 54.94% on the in-the-wild dataset was reached, with SincNet it was 11.83%. The same happened when evaluating RawGAT-ST on these two different front-ends. Compared to the other classifiers, PC-DARTS performed worse with an EER of 38.95% on the in-the-wild dataset when trained on the ASVspoof 2019 LA dataset and 43.11% when trained on WaveFake. However, since LFCC was used as the front-end here, the results are not as bad when compared to the other detectors using the LFCC front-end. One idea for future work would be to test PC-DARTS with a SincNet front-end.

Overall, the models performed rather differently, depending on the training sets used. But, the SincNet based front-end provided better results when trained on both training sets. Consequently, we employed SincNet based front-ends in the closed condition of the ASVspoof5 challenge. Furthermore, we evaluated AASIST(-L) and RawGAT-ST as these back-ends provided the best results.

5.2. Challenge Results

In the challenge, we evaluated different combinations and data augmentation techniques in the two conditions: closed and open.

5.2.1. Closed Condition

Because of the results in our preparatory analysis (see Table 1) we tested AASIST(-L) and RawGAT-ST with their already implemented SincNet based front-end on the ASVspoof5 progress set. All models were trained on the ASVspoof5 training set only. The data augmentation types were varied, see Table 2 for the results of the different combinations on the ASVspoof5 progress set. Without data augmentation (only using RawBoost), RawGAT-ST and AASIST performed nearly the same with a min DCF of 0.423 and 0.422. AASIST-L performed worse, with a min-DCF of 0.518, which is why AASIST-L is not considered further. Using gaussian noise and mp3 compression, AASIST performed better then RawGAT-ST with a min DCF of 0.416 (RawGAT-ST: 0.436). Using AASIST, the use of these data augmentation methods improved the results. Using RawGAT-ST the results worsen. Therefore, we also tested AASIST when increasing the trainings set with data augmentations using BandStopFilter, SevenBandParametric and TanhDistortion. This improved the results to a min DCF of 0.354, being also our best score in this setting. In a last step, we combined both data augmentation techniques, adding gaussian noise, mp3 compression and BandStopFilter, SevenBandParametric and TanhDistortion. This deteriorated the results slightly, leading to a min DCF of 0.358.

5.2.2. Open Condition

In the open condition we were allowed to use additional datasets and pre-trained models. Therefore, we applied three different SSL-based front-ends, using AASIST as previous best model as back-end. See Table 3 for the results on the ASVspoof5 progress set. For data augmentation we first applied gaussian noise and mp3 compression, as it improved the results in the first setting. Additionally, we tested different training sets using the ASVspoof5, WaveFake(+LJ Speech), LibriSeVoc, ASVspoof 2019 LA and the FoR dataset, in the hope to improve the results of the detector when it is trained on diverse data. Testing three different front-ends, the results when using all these training data deteriorated, compared to when only using the ASVspoof5 training set. For example, with a HuBERT base front-end, a min DCF of 0.102 was calculated when trained on ASVspoof5, when trained on all the training data described a min DCF of 0.155 was reached. We assume, it may be the case that the use of a substantial quantity of training data does not necessarily lead to enhanced outcomes. Alternatively, it is possible that the model dimension is insufficiently sized.

Overall, the front-end Wav2Vec2.0 (large 960), being also the one with the largest embedding dimension (1024) and most numbers of parameters (317M) [13], performed better than the other two front-ends tested (Wav2Vec2.0 trained on ASVspoof5: 0.066; HuBERT base trained on ASVspoof5: 0.102; HuBERT base robust mgr trained on ASVspoof5: 0.130). HuBERT base robust performed worst. As Wav2Vec2.0 being the best front-end, we tested its performance using the ASVspoof5 training set and combining it with the ASVspoof 2019 LA set, as this training set is from a previous challenge, we hoped to improve the results with this possibly similar structured but additional data. Using this additional dataset, the min DCF slightly improved to 0.063 (previously, without ASVspoof 2019 LA: 0.066). This was also our best result in this setting.

Additionally, we again extended the different training set combinations by using BandStopFilter, SevenBandParametric and TanhDistortion. In contrast to the closed condition, in all settings, the results worsen with the use of these data augmentation techniques (ASVspoof5: 0.081/0.066; ASVspoof5+ASVspoof 2019 LA: 0.074/0.063; all data: 0.144/0.097). Interestingly, the results when using these three data augmentation types on ASVspoof5 only and on a com-

Table 2: *Results in the closed condition on the ASVspoof5 progress set. The best results in terms of min-DCF are marked in grey.*

Back-end	Data Augmentation		minDCF	actDCF	Cllr	EER
	GaussianNoise, Mp3 compression	BandStopFilter, SevenBandParametric, TanhDistortion				
RawGAT-ST	-	-	0.423	0.573	1.066	17.66
RawGAT-ST	✓	-	0.436	0.659	1.011	18.09
AASIST-L	✓	-	0.518	0.667	1.333	21.44
AASIST	-	-	0.422	0.55	0.844	16.89
AASIST	✓	-	0.416	0.592	0.983	17.16
AASIST	-	✓	0.354	0.479	0.731	13.71
AASIST	✓	✓	0.358	0.590	0.880	14.19

Table 3: *Results in the open condition on the ASVspoof5 progress set. As classifier, we used AASIST in all settings. The best results in terms of min-DCF are marked in grey. Abbreviation: all data: ASVspoof5, WaveFake(+LJ), LibriSeVoc, ASVspoof 2019 LA, FoR.*

Front-end	Data Augmentation		Training Data	minDCF	actDCF	Cllr	EER
	GaussianNoise, Mp3 compression	BandStopFilter, SevenBandParametric, TanhDistortion					
hubert_base	✓	-	ASVspoof5	0.102	0.248	0.393	3.90
hubert_base	✓	-	all data	0.155	0.156	0.222	5.43
hubert_base_robust_mgr	✓	-	ASVspoof5	0.130	0.223	0.376	5.14
hubert_base_robust_mgr	✓	-	all data	0.151	0.156	0.226	5.35
wav2vec2_large_960	✓	-	ASVspoof5	0.066	0.291	0.475	2.58
wav2vec2_large_960	✓	-	ASVspoof5, ASVspoof 2019 LA	0.063	0.113	0.164	2.27
wav2vec2_large_960	✓	-	all data	0.097	0.101	0.136	3.37
wav2vec2_large_960	-	✓	ASVspoof5	0.081	0.282	0.516	3.07
wav2vec2_large_960	-	✓	ASVspoof5, ASVspoof 2019 LA	0.074	0.178	0.224	2.66
wav2vec2_large_960	-	✓	all data	0.144	0.155	0.208	5.04

bination of ASVspoof5 and ASVspoof 2019 LA, the use of ASVspoof 2019 LA improved the min DCF from 0.081 to 0.074. Without these data augmentation, the additional use of ASVspoof 2019 LA only improved the min DCF slightly (0.066 to 0.063).

In the open and closed setting, we tested AASIST when trained on the ASVspoof5 training set only and applying gaussian noise and mp3 compression for data augmentation. The only difference was the front-end used, being SincNet based in the closed condition and SSL-based in the open condition. This setting allows us to see the effect of the SSL-based front-end. With the SincNet based front-end a min DCF of 0.416 was calculated. As SSL-based front-ends, we tested three different models in the same setting. With the Wav2Vec2.0 front-end a min DCF of 0.066 was calculated, with HuBERT base 0.102 and with HuBERT base robust mgr a min DCF of 0.130. Through the SSL-based feature extraction the results have improved heavily.

5.2.3. ASVspoof5 Evaluation Set

Lastly, the best detectors on the ASVspoof5 progress set were tested on the ASVspoof5 evaluation set. See Table 4 for the results. Both detectors performed worse on the evaluation set than on the progress set. AASIST trained on ASVspoof5 with data augmentation using BandStopFilter, SevenBandParametric and TanhDistortion achieved a min DCF of 0.589 on the evaluation set (progress set: 0.354). AASIST with a Wav2Vec2.0 front-end and trained on ASVspoof5 and ASVspoof 2019 LA with data augmentation using gaussian noise and mp3 compression scored a min DCF of 0.174 on the evaluation set (progress set: 0.063).

For the creation of the evaluation set, 16 attacks and 11 codecs were used [24]. Here, besides the overall pooled min DCF, we also evaluated the performance of the two best detectors dependent on the different attack types and codecs used. See Table 5 for the evaluation in the closed condition and Table 6 in the open condition using an SSL-based front-end. In Table 5 the most challenging attack codec combinations are marked in grey. As most challenging combinations, we describe combinations with a min DCF higher than 0.9. The non-SSL based detector had the most problems identifying the attacks A19, A20 and A30, on all three attacks a min DCF of 1 was reached, even when using no codecs. A19 is an attack using MaryTTS [25], A20 and A30 are adversarial attacks [24]. The most challenging codecs for our non-SSL based detectors were codec-4 (Encodec [26]), 7 (mp3+Encodec) and 10 (speex) with pooled min DCF of 0.675, 0.683 and 0.701. All three codecs score higher than 0.9, combined with six different attacks. Additionally, we marked all best identified attack-codec combinations, being the one with a min DCF smaller 0.1. Overall, attack A29 (pre-trained XTTS [27]) was the best recognized attack with a pooled score of 0.075 and having scores smaller 0.1 in combination with eight of the 11 codecs. The second best recognized attack was A21 (ToucanTTS [28] +BigVGAN [29]) with a pooled min DCF of 0.148. As codecs, no codecs and codec-5 (mp3) was recognized best, having a min DCF smaller 0.1 on five and four of the 16 attacks.

The SSL-based detector performed overall better (see Table 6). Therefore, in this setting, we defined the best attack-codec combinations with a score smaller 0.01 and the most challenging combinations having a min DCF greater 0.5. With this defi-

Table 4: *Results of the best detectors on the ASVSpoof5 evaluation set.*

System Description	minDCF	actDCF	Cllr	EER
Closed: AASIST trained on ASVspoof5 with BandStop-Filter, SevenBandParametric, TanhDistortion	0.589	0.688	1.328	24.59
Open: AASIST with Wav2Vec2 trained on ASVspoof5 and ASVspoof 2019 LA with GaussianNoise, Mp3 Compression	0.174	0.309	0.476	6.06

Table 5: *Min DCF results in the closed condition on the ASVspoof5 evaluation set by attack type and codec. The most challenging attack-codec combinations in terms of min-DCF (here defined as >0.9) are marked in dark grey. The best combinations (here defined as <0.1) are marked in light grey (pld: pooled).*

	pld.	**-**	**codec-1**	**codec-2**	**codec-3**	**codec-4**	**codec-5**	**codec-6**	**codec-7**	**codec-8**	**codec-9**	**codec-10**	**codec-11**
pld.	0.589	0.506	0.550	0.578	0.609	**0.675**	0.521	0.528	**0.683**	0.667	0.629	**0.701**	0.5539
A17	0.183	0.047	0.174	0.123	0.196	0.252	0.083	0.111	0.280	0.336	0.302	0.289	0.104
A18	0.797	0.644	0.767	0.846	0.888	0.844	0.701	0.670	0.873	0.864	0.904	1	0.760
A19	1	1	1	1	1	1	1	1	1	1	1	1	1
A20	1	1	1	1	1	1	1	0.996	1	1	0.902	0.946	1
A21	0.148	0.037	0.130	0.090	0.152	0.224	0.052	0.097	0.228	0.302	0.173	0.193	0.085
A22	0.193	0.091	0.152	0.153	0.181	0.290	0.099	0.139	0.299	0.319	0.259	0.267	0.136
A23	0.456	0.368	0.394	0.405	0.524	0.571	0.358	0.370	0.575	0.535	0.447	0.572	0.385
A24	0.245	0.098	0.207	0.167	0.285	0.290	0.154	0.206	0.322	0.374	0.381	0.453	0.191
A25	0.478	0.212	0.502	0.493	0.520	0.681	0.285	0.330	0.673	0.723	0.692	0.672	0.415
A26	0.723	0.509	0.703	0.723	0.789	0.919	0.530	0.601	0.913	0.867	0.933	0.945	0.715
A27	0.849	0.769	0.726	0.901	0.891	0.965	0.781	0.761	0.969	0.865	0.863	0.984	0.760
A28	0.465	0.247	0.407	0.474	0.480	0.564	0.287	0.376	0.599	0.641	0.704	0.779	0.454
A29	0.075	0.006	0.084	0.022	0.044	0.077	0.012	0.032	0.110	0.244	0.105	0.099	0.023
A30	1	1	0.974	1	1	1	1	1	1	0.999	0.996	1	1
A31	0.457	0.408	0.339	0.414	0.447	0.578	0.409	0.426	0.606	0.492	0.408	0.525	0.420
A32	0.778	0.744	0.714	0.800	0.803	0.946	0.771	0.715	0.936	0.757	0.633	0.788	0.613

Table 6: *Min DCF results in the open condition on the ASVspoof5 evaluation set by attack type and codec. The most challenging attack-codec combinations in terms of min-DCF (here defined as >0.5) are marked in dark grey. The best combinations (here defined as <0.01) are marked in light grey (pld: pooled).*

	pld.	**-**	**codec-1**	**codec-2**	**codec-3**	**codec-4**	**codec-5**	**codec-6**	**codec-7**	**codec-8**	**codec-9**	**codec-10**	**codec-11**
pld.	0.174	0.056	0.128	0.119	0.157	0.334	0.065	0.082	0.414	0.291	0.166	0.209	0.078
A17	0.033	0.002	0.010	0.005	0.020	0.083	0.001	0.003	0.111	0.088	0.015	0.031	0.006
A18	0.102	0.013	0.052	0.053	0.090	0.237	0.016	0.030	0.311	0.207	0.106	0.139	0.026
A19	0.025	0.002	0.008	0.005	0.016	0.063	0.001	0.004	0.086	0.047	0.009	0.018	0.002
A20	0.093	0.018	0.049	0.052	0.067	0.215	0.022	0.031	0.285	0.140	0.066	0.084	0.021
A21	0.041	0.004	0.019	0.010	0.033	0.091	0.003	0.007	0.126	0.094	0.015	0.031	0.008
A22	0.131	0.030	0.084	0.075	0.120	0.287	0.032	0.045	0.375	0.252	0.108	0.163	0.041
A23	0.106	0.019	0.063	0.057	0.095	0.225	0.022	0.034	0.283	0.205	0.100	0.135	0.028
A24	0.207	0.078	0.155	0.135	0.198	0.382	0.084	0.108	0.442	0.330	0.177	0.241	0.080
A25	0.077	0.010	0.037	0.032	0.054	0.188	0.014	0.018	0.252	0.159	0.070	0.095	0.017
A26	0.053	0.004	0.019	0.017	0.047	0.123	0.005	0.010	0.176	0.115	0.041	0.063	0.009
A27	0.161	0.033	0.106	0.113	0.145	0.370	0.045	0.065	0.464	0.300	0.158	0.205	0.046
A28	0.470	0.242	0.418	0.361	0.446	0.669	0.262	0.331	0.761	0.719	0.536	0.660	0.423
A29	0.024	0.004	0.008	0.005	0.014	0.043	0.004	0.004	0.056	0.085	0.008	0.018	0.007
A30	0.174	0.043	0.123	0.123	0.161	0.397	0.052	0.076	0.484	0.312	0.176	0.220	0.050
A31	0.269	0.088	0.239	0.256	0.280	0.505	0.114	0.135	0.615	0.444	0.309	0.356	0.100
A32	0.164	0.032	0.117	0.122	0.160	0.411	0.042	0.058	0.503	0.298	0.171	0.207	0.037

nition, we identified the attacks A28 (pre-trained YourTTS [30]) and A31 (malacopula+[31]) as the most challenging, having a pooled min DCF of 0.470 and 0.269 and having scores greater 0.5 in combination with five and two codecs. Again, codec-4 and 7 were the most challenging codecs. The attacks A19 (MaryTTS) and A29 (pre-trained XTTS) were identified as the ones being the most well recognized attacks by this SSL-based detector. Attack A19 has a pooled min DCF of 0.025 and A29 a pooled min DCF of 0.024. Both attacks have a min DCF smaller 0.01 when combined with six of the 11 codecs. Interestingly, A29 was identified as well recognisable by both detectors. The attack A19 was one of the most challenging attacks in the first setting using a non-SSL based detector and one of the best recognized in the second setting using an SSL-based detector. Furthermore, attack A28 (pre-trained YourTTS) stands out, which was better recognised in the closed condition (pooled min DCF: 0.465) than in the open (pooled min DCF: 0.470).

6. Conclusion

We examined detectors with different front-/ back-end combinations and found, that AASIST and RawGAT-ST with a SincNet based front-end performs better on different trainings sets than RawNet2, PC-DARTS and using a LFCC front-end. Moreover, our findings indicate that an abundance of training data may not always be beneficial and could potentially lead to suboptimal outcomes. Also, the detectors' architecture in combination with the training data effects the results. When using an SSL-based front-end heavy data augmentation was not necessary. Contrarily, without an SSL-based front-end the results improved with heavy data augmentation. Overall, an SSL-based front-end highly improved the results, at least on the ASVspoof5 progress split. It appears that the embedding dimension and the number of parameters of the SSL model have an impact on the results, with the largest model demonstrating the most favourable outcomes. This is evidenced by the superior performance of Wav2vec2.0 (large 960) as the optimal front-end among the three tested. When viewing the results on the evaluation set, for both detectors the codec-4 and 7, both based on Encodec, were the most challenging. The attack using pre-trained XTTS (A29) were well recognized in both settings. MaryTTS (A19) proved to be difficult to detect for the non SSL-based detector in the closed setting (pooled min DCF of 1) but were well recognized in the open setting (pooled min DCF of 0.025). The inverse is true for the attack using pre-trained YourTTS (A28).

7. Acknowledgement

This research work was supported by the National Research Center for Applied Cybersecurity ATHENE and within the joint project SecMedID with the BSI.

8. References

[1] Nicolas Michael Müller, Pavel Czempin, Franziska Dieckmann, Adam Froghyar, and Konstantin Böttinger, "Does audio deepfake detection generalize?," in *Interspeech*, 2022.

[2] Xin Wang and Junichi Yamagishi, "Can large-scale vocoded spoofed data improve speech spoofing countermeasure with a self-supervised front end?," *ArXiv*, vol. abs/2309.06014, 2023.

[3] Vineel Pratap, Qiantong Xu, Anuroop Sriram, Gabriel Synnaeve, and Ronan Collobert, "Mls: A large-scale multilingual dataset for speech research," *arXiv preprint arXiv:2012.03411*, 2020.

[4] Hemlata Tak, Jose Patino, Massimiliano Todisco, Andreas Nautsch, Nicholas Evans, and Anthony Larcher, "End-to-end anti-spoofing with rawnet2," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 6369–6373.

[5] Hemlata Tak, Jee-weon Jung, Jose Patino, Madhu Kamble, Massimiliano Todisco, and Nicholas Evans, "End-to-end spectro-temporal graph attention networks for speaker verification anti-spoofing and speech deepfake detection," *arXiv preprint arXiv:2107.12710*, 2021.

[6] Jee-weon Jung, Hee-Soo Heo, Hemlata Tak, Hye-jin Shim, Joon Son Chung, Bong-Jin Lee, Ha-Jin Yu, and Nicholas Evans, "Aasist: Audio anti-spoofing using integrated spectro-temporal graph attention networks," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 6367–6371.

[7] Wanying Ge, Michele Panariello, Jose Patino, Massimiliano Todisco, and Nicholas Evans, "Partially-connected differentiable architecture search for deepfake and spoofing detection," *arXiv preprint arXiv:2104.03123*, 2021.

[8] Jacob Kahn, Morgane Riviere, Weiyi Zheng, Evgeny Kharitonov, Qiantong Xu, Pierre-Emmanuel Mazaré, Julien Karadayi, Vitaliy Liptchinsky, Ronan Collobert, Christian Fuegen, et al., "Libri-light: A benchmark for asr with limited or no supervision," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7669–7673.

[9] David Snyder, Guoguo Chen, and Daniel Povey, "Musan: A music, speech, and noise corpus," *arXiv preprint arXiv:1510.08484*, 2015.

[10] Jodi Kearns, "Librivox: Free public domain audiobooks," *Reference Reviews*, vol. 28, no. 1, pp. 7–8, 2014.

[11] Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed, "Hubert: Self-supervised speech representation learning by masked prediction of hidden units," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 3451–3460, 2021.

[12] Kuan Po Huang, Yu-Kuan Fu, Yu Zhang, and Hung-yi Lee, "Improving distortion robustness of self-supervised speech processing tasks with domain adaptation," *arXiv preprint arXiv:2203.16104*, 2022.

[13] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *Advances in neural information processing systems*, vol. 33, pp. 12449–12460, 2020.

[14] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.

[15] Mirco Ravanelli and Yoshua Bengio, "Speaker recognition from raw waveform with sincnet," in *2018 IEEE spoken language technology workshop (SLT)*. IEEE, 2018, pp. 1021–1028.

[16] Hemlata Tak, Madhu Kamble, Jose Patino, Massimiliano Todisco, and Nicholas Evans, "Rawboost: A raw data boosting and augmentation method applied to automatic speaker verification anti-spoofing," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 6382–6386.

[17] Hemlata Tak, Massimiliano Todisco, Xin Wang, Jee-weon Jung, Junichi Yamagishi, and Nicholas Evans, "Automatic speaker verification spoofing and deepfake detection using wav2vec 2.0 and data augmentation," in *The Speaker and Language Recognition Workshop*, 2022.

[18] 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*, vol. 64, pp. 101114, 2020.

[19] Joel Frank and Lea Schönherr, "Wavefake: A data set to facilitate audio deepfake detection," 2021.

[20] Keith Ito and Linda Johnson, "The lj speech dataset," `https://keithito.com/LJ-Speech-Dataset/`, 2017.

[21] Xuechen Liu, Xin Wang, Md Sahidullah, Jose Patino, Héctor Delgado, Tomi Kinnunen, Massimiliano Todisco, Junichi Yamagishi, Nicholas Evans, Andreas Nautsch, et al., "Asvspoof 2021: Towards spoofed and deepfake speech detection in the wild," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2023.

[22] Chengzhe Sun, Shan Jia, Shuwei Hou, and Siwei Lyu, "Ai-synthesized voice detection using neural vocoder artifacts," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2023, pp. 904–912.

[23] Ricardo Reimao and Vassilios Tzerpos, "For: A dataset for synthetic speech detection," in *2019 International Conference on Speech Technology and Human-Computer Dialogue (SpeD)*. IEEE, 2019, pp. 1–10.

[24] Xin Wang, Héctor Delgado, Hemlata Tak, Jee-weon Jung, Hye-jin Shim, Massimiliano Todisco, Ivan Kukanov, Xuechen Liu, Md Sahidullah, Tomi Kinnunen, Nicholas Evans, Kong Aik Lee, and Junichi Yamagishi, "ASVspoof 5: Crowdsourced speech data, deepfakes, and adversarial attacks at scale," in *ASVspoof Workshop 2024 (accepted)*, 2024.

[25] Ingmar Steiner and Sébastien Le Maguer, "Creating new language and voice components for the updated marytts text-to-speech synthesis platform," *arXiv preprint arXiv:1712.04787*, 2017.

[26] Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi, "High fidelity neural audio compression," *arXiv preprint arXiv:2210.13438*, 2022.

[27] Edresson Casanova, Kelly Davis, Eren Gölge, Görkem Göknar, Iulian Gulea, Logan Hart, Aya Aljafari, Joshua Meyer, Reuben Morais, Samuel Olayemi, et al., "Xtts: a massively multilingual zero-shot text-to-speech model," *arXiv preprint arXiv:2406.04904*, 2024.

[28] Florian Lux, Julia Koch, and Ngoc Thang Vu, "Low-resource multilingual and zero-shot multispeaker tts," *arXiv preprint arXiv:2210.12223*, 2022.

[29] Sang-gil Lee, Wei Ping, Boris Ginsburg, Bryan Catanzaro, and Sungroh Yoon, "Bigvgan: A universal neural vocoder with large-scale training," *arXiv preprint arXiv:2206.04658*, 2022.

[30] Edresson Casanova, Julian Weber, Christopher D Shulby, Arnaldo Candido Junior, Eren Gölge, and Moacir A Ponti, "Yourtts: Towards zero-shot multi-speaker tts and zero-shot voice conversion for everyone," in *International Conference on Machine Learning*. PMLR, 2022, pp. 2709–2720.

[31] Florian Lux, Julia Koch, and Ngoc Thang Vu, "Exact prosody cloning in zero-shot multispeaker text-to-speech," in *2022 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2023, pp. 962–969.