The SHADOW Team Submission to the ASVSpoof 2024 Challenge

Jesús Villalba[1,2], *Tiantian Feng*[3], *Thomas Thebaud*[1,2],
Jihwan Lee[3], *Shrikanth Narayanan*[3], *Najim Dehak*[1,2]

[1]Center for Language and Speech Processing, Johns Hopkins University, Baltimore, MD, USA
[2]Human Language Technology Center of Excellence, Johns Hopkins University, Baltimore, MD, USA
[3]Signal Analysis and Interpretation Laboratory, University of Southern California,
Los Angeles, CA, USA

jvilla17@jhu.edu

Abstract

This paper presents the SHADOW team's submission to the ASVSpoof 2024 challenge. We evaluated various models, including ECAPA-TDNN, ResNet34, ConvNeXt, and S4 Structured-State-Space Models. 2D convolution-based models outperformed other types, with the best Progress set results achieved using FwSE-ResNet34 with codec augmentations. In the Track 1 Eval set, this system achieved minDCF=0.44, a 47% improvement over the challenge baseline.

For Track2, we contribute a straightforward method for combining well-calibrated speaker and spoofing detection scores into a single system. This involves calculating the posterior probability for a trial being both same-speaker and bonafide. However, the significant mismatch between the Dev and Progress/Eval sets not only complicated the selection of the best systems and codecs but also impacted the Eval set calibration and score combination. Nevertheless, we achieved a-DCF=0.397 in Track 2, a 42% improvement over the baseline.

1. Introduction

In recent years, the rapid advancement of speech synthesis and voice conversion technologies has led to significant improvements in the quality and accessibility of synthetic speech. While these technologies offer numerous benefits, they also pose a growing threat in the form of spoofing attacks, where synthetic or manipulated speech is used to deceive automatic speaker verification (ASV) systems. The ASVSpoof 2024 challenge [1] aims to address these challenges by fostering the development of robust anti-spoofing techniques capable of distinguishing between genuine and spoofed speech.

In this paper, we present our submission to the ASVSpoof 2024 challenge. Various architectures have been proposed in the literature for the spoofing detection task (Track 1), including squeeze-excitation ResNets like ASSERT [2], Spectro-Temporal Graph Attention Networks (AASIST) [3], and RawNet2 [4]. For our submission, we focused on evaluating well-known neural network architectures that have demonstrated strong performance in speaker detection tasks, such as ECAPA-TDNN [5], Thin-ResNet34 [6], FwSE-ResNet34 [7] (similar to ASSERT), besides of ConvNeXt [8], and Structured State-Space Models [9].

In Track 2, we integrated speaker and spoofing detection models into a unified system that outputs a unique score, representing the log-likelihood ratio between the target bonafide hypothesis and the non-target or spoofing hypothesis. Various methods for this integration are proposed in [10], including both linear and non-linear fusion of the speaker and spoofing log-likelihood ratios. Instead, we propose a straightforward method that involves computing the posterior probability that the enrollment and test speakers are the same and that the test sample is bonafide, given the trial's data, i.e., $P\left(\text{same}, \text{bonafide}|\mathbf{x}_e, \mathbf{x}_t\right)$. This can be accomplished using basic probability rules, assuming that we have well-calibrated speaker and spoofing detection scores.

The proposed systems achieved competitive performance in the challenge, demonstrating significant improvements of over 40% relative to the baseline systems provided by the organizers. The paper is organized as follows: Section 2 details the spoofing detection systems, while Section 3 outlines the mathematical framework for combining speaker and spoofing systems. Section 4 describes the experimental setup and the speaker recognition system used in Track 2. Finally, Section 5 presents a discussion of the results, and Section 6 concludes the paper with a summary of our findings.

2. Spoofing Detection Models

2.1. Feature Extraction

We utilized an 80-dimensional log-Mel filter bank computed at 16 kHz sampling frequency with 25 ms. window length and 10 ms. shift. The features were short-time mean normalized with a 3-second window. Silence frames were retained in the processing.

2.2. ECAPA-TDNN

ECAPA-TDNN, which stands for Emphasized Channel Attention, Propagation, and Aggregation Time Delay Neural Network [5], enhances traditional Time Delay Neural Networks (TDNNs) [11] by incorporating several advanced features. It utilizes Res2Net blocks [12] based on 1-dimensional convolutions. Each Res2Net block integrates a Squeeze-and-Excitation (SE) mechanism [13], which dynamically recalibrates channel-wise feature responses, emphasizing the most important channels. Additionally, ECAPA-TDNN employs a multi-layer feature aggregation strategy by concatenating outputs from various network layers, followed by a linear projection. The frame-level features are pooled into a single vector using a channel-wise attentive pooling mechanism. This pooled vector is then linearly projected into a bottleneck feature of 192 dimensions. For the output layer, we used the subcenter additive angular margin softmax (Subcenter-AAM-Softmax) [14], with two outputs representing bonafide and spoof categories. To enhance class sep-

Table 1: *Detailed architecture specifications for Thin-ResNet34, FwSE-ResNet34 and ConvNext. The numbers inside the parentheses denote the filter sizes and the output channel sizes. The multiplication factor indicates the number of residual blocks at each stage. The inner brackets following by FwSE indicates the output dimension of the two fully connected layers in an SE module. GRN stands for Global Response Normalization.*

Layer Name	Thin-ResNet34	FwSE-ResNet34	ConvNext2d-Atto	ConvNext2d-Pico
Stem	3×3, 16, stride=1	3×3, 64, stride=1	7×7, 40, stride=1	7×7, 64, stride=1
Res1	$\begin{bmatrix} 3\times 3, 16 \\ 3\times 3, 16 \end{bmatrix} \times 3$	$\begin{bmatrix} 3\times 3, 64 \\ 3\times 3, 64 \\ \text{FwSE}[20, 80] \end{bmatrix} \times 3$	$\begin{bmatrix} d7\times 7, 40 \\ 1\times 1, 160 \\ 1\times 1, 40 \\ \text{GRN} \end{bmatrix} \times 2$	$\begin{bmatrix} d7\times 7, 64 \\ 1\times 1, 256 \\ 1\times 1, 64 \\ \text{GRN} \end{bmatrix} \times 2$
Res2	$\begin{bmatrix} 3\times 3, 32, s=2 \\ 3\times 3, 32 \end{bmatrix} \times 1$	$\begin{bmatrix} 3\times 3, 128, s=2 \\ 3\times 3, 128 \\ \text{FwSE}[10, 40] \end{bmatrix} \times 1$	$\begin{bmatrix} 2\times 2, 80, s=2 \end{bmatrix} \times 1$	$\begin{bmatrix} 2\times 2, 128, s=2 \end{bmatrix} \times 1$
	$\begin{bmatrix} 3\times 3, 32 \\ 3\times 3, 32 \end{bmatrix} \times 3$	$\begin{bmatrix} 3\times 3, 128 \\ 3\times 3, 128 \\ \text{FwSE}[10, 40] \end{bmatrix} \times 3$	$\begin{bmatrix} d7\times 7, 80 \\ 1\times 1, 320 \\ 1\times 1, 80 \\ \text{GRN} \end{bmatrix} \times 2$	$\begin{bmatrix} d7\times 7, 128 \\ 1\times 1, 512 \\ 1\times 1, 128 \\ \text{GRN} \end{bmatrix} \times 2$
Res3	$\begin{bmatrix} 3\times 3, 64, s=2 \\ 3\times 3, 64 \end{bmatrix} \times 1$	$\begin{bmatrix} 3\times 3, 256, s=2 \\ 3\times 3, 256 \\ \text{FwSE}[5, 20] \end{bmatrix} \times 1$	$\begin{bmatrix} 2\times 2, 160, s=2 \end{bmatrix} \times 1$	$\begin{bmatrix} 2\times 2, 256, s=2 \end{bmatrix} \times 1$
	$\begin{bmatrix} 3\times 3, 64 \\ 3\times 3, 64 \end{bmatrix} \times 5$	$\begin{bmatrix} 3\times 3, 256 \\ 3\times 3, 256 \\ \text{FwSE}[5, 20] \end{bmatrix} \times 5$	$\begin{bmatrix} d7\times 7, 160 \\ 1\times 1, 640 \\ 1\times 1, 160 \\ \text{GRN} \end{bmatrix} \times 6$	$\begin{bmatrix} d7\times 7, 256 \\ 1\times 1, 1024 \\ 1\times 1, 256 \\ \text{GRN} \end{bmatrix} \times 6$
Res4	$\begin{bmatrix} 3\times 3, 128, s=2 \\ 3\times 3, 128 \end{bmatrix} \times 1$	$\begin{bmatrix} 3\times 3, 512, s=2 \\ 3\times 3, 512 \\ \text{FwSE}[2, 10] \end{bmatrix} \times 1$	$\begin{bmatrix} 2\times 2, 320, s=2 \end{bmatrix} \times 1$	$\begin{bmatrix} 2\times 2, 512, s=2 \end{bmatrix} \times 1$
	$\begin{bmatrix} 3\times 3, 128 \\ 3\times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times 3, 512 \\ 3\times 3, 512 \\ \text{FwSE}[2, 10] \end{bmatrix} \times 2$	$\begin{bmatrix} d7\times 7, 320 \\ 1\times 1, 1280 \\ 1\times 1, 320 \\ \text{GRN} \end{bmatrix} \times 2$	$\begin{bmatrix} d7\times 7, 512 \\ 1\times 1, 2048 \\ 1\times 1, 512 \\ \text{GRN} \end{bmatrix} \times 2$
Pooling	Channel-wise Attentive Pooling			
Projection	Fully Connected, dim=192			
Output	Subcenter-AAM-Softmax, dim=2, subcenters=2, margin=0.2, scale=30			

aration, we applied an angular margin of 0.2. Each class was assigned two subcenters to mitigate the impact of mislabeling in the training data.

2.3. ResNet34

We experimented with two variants of the residual network [6], namely, ThinResNet34 and FwSE-ResNet34, as detailed in Table 1. The ResNet34 architecture consists of an input stem layer followed by 16 Basic residual blocks utilizing 2D convolutions. This design downscales the feature maps through stride-2 convolutions (resulting in an overall downsampling of 8×) while simultaneously doubling the number of channels. The output of this network is a four-dimensional tensor $(B, C, F/8, T/8)$, where B denotes the batch size, C is the number of channels, F is the number of Mel filters, and T is the temporal dimension. Before passing the features to the pooling layer, the channel and frequency dimensions are flattened to $(B, C \times F/8, T/8)$. Given the limited training data, we explored Thin-ResNet34, which reduces the number of convolution channels by a factor of 4 compared to regular ResNet34, thereby reducing the model's parameters by 16×. However, we achieved better performance with the larger FwSE-ResNet34. FwSE-ResNet34 enhances ResNet34 by incorporating frequency-wise squeeze excitation in the residual blocks. The FwSE mechanism, as proposed in [7], performs a squeeze operation by pooling feature maps across the time and channel dimensions to produce an excitation scalar for each frequency dimension. These models also utilized channel-wise attentive pooling and a Subcenter-AAM-Softmax output layer.

2.4. ConvNeXt2d

ConvNeXt [15] is a modern deep learning architecture that revisits traditional convolutional neural network (CNN) designs with insights from recent advancements in vision transformers. As a pure CNN model, ConvNeXt integrates key features such as large kernel sizes, depthwise convolutions, and inverted bottlenecks, drawing inspiration from transformer models. Unlike the traditional bottleneck blocks used in many ResNet models, which first reduce the number of channels before expanding them, the inverted bottleneck block in ConvNeXt expands the number of channels first and then reduces them. This block begins with a depthwise convolution with a large kernel (7x7), capturing spatial information for each channel independently without significantly increasing the number of parameters. This is followed by a pointwise convolution that expands $\times 4$ the feature map dimensions, and another pointwise convolution that reduces the number of channels back to the bottleneck size. Additionally, ConvNeXt replaces BatchNorm with LayerNorm and ReLU activations with GELU. ConvNeXt layers mirror the structure of transformers where depthwise convolutions function similarly to self-attention layers, and the pointwise convolutions resemble the transformer feedforward block. This design enables the block to capture rich spatial information while maintaining computational efficiency and a manageable parameter count. Following the ConvNeXt-v2 scheme [8], we also incorporated Global Response Normalization (GRN) as a replacement for squeeze-excitation. GRN normalizes the feature map based on the L2 norm of each channel computed separately across the spatial dimensions., ensuring balanced response magnitudes and preventing any single channel from dominating the network's output. Like the ECAPA-TDNN and ResNet models, our ConvNeXt utilizes channel-wise attentive pooling and a Subcenter-AAM-Softmax output.

2.5. S4

We also experimented with the State-space models (SSM) [9] for spoofing detection. SSM is a recent popular deep neural network architecture that is specifically designed to model long-term sequences, such as those with long duration or extremely high sampling frequency. In practice, we can regard the SSMs as special cases of convolutional and recurrent layers. Notably, we leverage the *Structured State Space for Sequence Modeling* (S4) that was introduced in [16]. Compared to normal SSMs, S4 was initialized using an approach called Hippo [17], which empirically yields a more stable training process. In our experiments, we adopt a modified version of the original S4. We experiment with the S4 model consisting of S4 blocks in $\{3, 6, 12\}$, each with a Layer Normalization module, the kernel $\bar{K}$ estimation module, a GELU [18] activation, Dropout, and a linear projection. We empirically experiment with the input and output dimension in $\{128, 256\}$ and a dropout rate of 0.1. The final output from the S4 layers is fed through a classifier layer with two linear layers and a GELU layer in between. We adopt the implementation from the GitHub link [1] described in our previous work [19].

3. Combination of Speaker and Spoofing Detection

We aim to combine the speaker verification and spoofing detection models into a single system that produces a unique likelihood ratio that can be used to decide between the target and non-target hypothesis in applications where spoofing is a risk. To do this, we propose to use the rules of probability. In essence, we are interested in obtaining the posterior probability for the case that enrollment and test speakers are the same and the test is bonafide given the trial's data $(\mathbf{x}_e, \mathbf{x}_t)$, i.e., $P\left(\text{same}, \text{bonafide}|\mathbf{x}_e, \mathbf{x}_t\right)$. Using the chain-rule, and assuming independence between the enrollment and the test bonafide/spoof status, we write the posterior as

$$\begin{aligned} &P\left(\text{same}, \text{bonafide}|\mathbf{x}_e, \mathbf{x}_t\right) = \\ &P\left(\text{same}|\mathbf{x}_e, \mathbf{x}_t, \text{bonafide}\right) P\left(\text{bonafide}|\mathbf{x}_t\right) \ . \end{aligned} \tag{1}$$

We note now that $P\left(\text{same}|\mathbf{x}_e, \mathbf{x}_t, \text{bonafide}\right)$ is the regular posterior of an unprotected speaker verification system since it assumes that $\mathbf{x}_t$ is bonafide and can be written as a function of the regular ASV log-likelihood ratio $\log R_{\text{ASV}}$,

$$\begin{aligned} &P\left(\text{same}|\mathbf{x}_e, \mathbf{x}_t, \text{bonafide}\right) = \\ &= \sigma\left(\log \frac{P\left(\mathbf{x}_e, \mathbf{x}_t|\text{same}, \text{bonafide}\right) P\left(\text{same}|\text{bonafide}\right)}{P\left(\mathbf{x}_e, \mathbf{x}_t|\text{diff}, \text{bonafide}\right) P\left(\text{diff}|\text{bonafide}\right)}\right) \qquad (2) \\ &= \sigma\left(\log R_{\text{ASV}} + \log \frac{P\left(\text{same}|\text{bonafide}\right)}{P\left(\text{diff}|\text{bonafide}\right)}\right) \qquad (3) \end{aligned}$$

where σ is the sigmoid function. Note that we are given $P\left(\text{target}\right) = 0.9405$, $P\left(\text{nontarget}\right) = 0.0095$, $P\left(\text{spoof}\right) = 0.0095$, and $C_{\text{miss}} = 1$, $C_{\text{fa}} = C_{\text{fa-spoof}} = 10$. From these priors, we derive,

$$P\left(\text{same}|\text{bonafide}\right) = \frac{P\left(\text{target}\right)}{P\left(\text{target}\right) + P\left(\text{nontarget}\right)} = 0.99 \tag{4}$$

$$P\left(\text{diff}|\text{bonafide}\right) = 1 - P\left(\text{same}|\text{bonafide}\right) = 0.01 \ . \tag{5}$$

Nevertheless, we need to take into account the costs of the different alternatives and derivate an equivalent effective prior as,

$$\frac{P_{\text{eff}}(\text{same}|\text{bonafide})}{1 - P_{\text{eff}}(\text{same}|\text{bonafide})} = \frac{C_{\text{miss}} P\left(\text{same}|\text{bonafide}\right)}{C_{\text{fa}} P\left(\text{diff}|\text{bonafide}\right)} \tag{6}$$

obtaining $P_{\text{eff}}(\text{same}|\text{bonafide}) = 0.90$. $P_{\text{eff}}(\text{same}|\text{bonafide})$ is the prior we need to plug into (3). Also, we need to use this effective prior to train the logistic regression calibrator that transforms the speaker verification score into a well-calibrated likelihood ratio $\log R_{\text{ASV}}$. This calibrator has to be trained on just bonafide trials since all the probabilities involved are *given the bonafide case*.

Equivalently, the bonafide posterior can be written as

$$P\left(\text{bonafide}|\mathbf{x}_t\right) = \sigma\left(\log R_{\text{bona/spoof}} + \log \frac{P\left(\text{bonafide}\right)}{P\left(\text{spoof}\right)}\right) \tag{7}$$

where $R_{\text{bona/spoof}}$ is the likelihood ratio between the bonafide vs spoof hypothesizes, and $P\left(\text{bonafide}\right) = 1 - P\left(\text{spoof}\right)$. Again, we need to consider the decision costs to calculate the effective prior

$$\frac{P_{\text{eff}}(\text{bonafide})}{1 - P_{\text{eff}}(\text{bonafide})} = \frac{C_{\text{miss}} P\left(\text{bonafide}\right)}{C_{\text{fa-spoof}} P\left(\text{spoof}\right)} \tag{8}$$

and use this effective prior to calibrating the score of the spoofing detection system into $\log R_{\text{bona/spoof}}$ in Eq. (7).

Thus, the product of (3) and (7) give us the desired posterior $P\left(\text{same}, \text{bonafide}|\mathbf{x}_e, \mathbf{x}_t\right)$. If we want the combined output in the form of a log-likelihood ratio instead of a posterior, we can transform the posterior into a combined log-likelihood ratio $\log R_{\text{ASV-spoof}}$, by inverting the logistic function,

$$\begin{aligned} &P\left(\text{same}, \text{bonafide}|\mathbf{x}_e, \mathbf{x}_t\right) = \\ &= \sigma\left(\log R_{\text{ASV-spoof}} + \log \frac{P_{\text{eff}}(\text{target})}{1 - P_{\text{eff}}(\text{target})}\right) \end{aligned} \tag{9}$$

with

$$\frac{P_{\text{eff}}(\text{target})}{1 - P_{\text{eff}}(\text{target})} = \frac{C_{\text{miss}} P(\text{target})}{C_{\text{fa}} P(\text{nontarget}) + C_{\text{fa-spoof}} P(\text{spoof})} \ . \tag{10}$$

Note that the speaker verification and spoofing detection likelihood ratios are application-independent, i.e., if they are well calibrated across a wide range of operating points, we can use any prior to transforming them into posterior probabilities. However, the combined likelihood ratio $R_{\text{ASV-spoof}}$ is no longer application-independent since we need to plug the priors into Eq. (3) and (7) to calculate it. If we change the priors, we need to repeat all the calculations. Therefore, for the combined system, producing a likelihood ratio rather than a posterior is not as advantageous.

[1] https://github.com/klean2050/tiles_ecg_model

Table 2: *Codecs used in the final submission, probability of using each codec and codec options*

Codec	Prob.	Options
Telephone		
A-law	0.17	Highpass: 100-300 Hz
μ-law	0.17	
G723.1	0.03	Lowpass: 3400-3700 Hz
G726	0.13	
Media		
G722	0.10	
MP3	0.25	CBR (50%): 128-320 kbps, VBR (50%): QScale=0-3, Compr=0-3
AC3	0.08	
Vorbis	0.02	Compression=6-10
Opus	0.05	Compression=6-10

4. Experimental Setup

4.1. ECAPA-TDNN, ResNet, ConvNext

ECAPA-TDNN, ResNets, and ConvNeXts were trained on the fixed condition data using 2-second chunks and an effective batch size of 256. The actual batch size varied depending on GPU memory and network size, with gradient accumulation employed to reach the desired effective batch size. We utilized the AdamW optimizer with a weight decay of 0.1. The learning rate was warmed up over 1.5k steps to a maximum of 0.001, and after 3k steps, it was halved every 3k steps. The AAM-Softmax margin was gradually increased from 0 to 0.2 over five epochs. Early stopping was based on performance on the development and progress datasets. The final model submitted for the Eval phase was trained for five epochs.

The data was augmented using noise (N) from Musan music and environmental noises [2], as well as synthetic and real reverberation (R) from RIR [3]. We also experimented with media codecs (MC), telephone codecs (TC), and a combination of telephone and media codecs (TMC), which was used in our final submission. The codecs were applied on-the-fly to 20% of the utterances using torchaudio[4]. We utilized all ffmpeg codecs compatible with torchaudio. Each codec type was applied with a different probability: codecs with longer processing times, such as G723.1 and Vorbis, were used less frequently, while MP3, which offers more configuration options, was applied more often. Telephone codecs also included resampling to 8 kHz and back to 16 kHz, high-pass filtering with a cut-off frequency between 100-300 Hz, and low-pass filtering with a cut-off frequency between 3400-3700 Hz. Table 2 lists the codec types, their application probabilities, and the specific options used for each codec.

4.2. S4

Similar to the prior experiments, we trained the S4 with 2-second speech chunks. We used a learning rate and a weight decay of 0.0002 and 0.001, respectively. We select the best-performing model based on the performance from the dev set. We did not submit the model in the Eval phase due to the relatively low performance on the progress set compared to other model architectures. We performed speech augmentation using Audiomentations [5]. Noise augmentation used noises from ESC-50 [20], excluding human-related sounds. We also applied other augmentation techniques, such as time masking, time stretching, and pitch shifting, with a probability of 0.5 and RIR from [6].

4.3. Speaker Verification System

The speaker verification system used in Track 2 was a FwSE-ResNet100 with the configuration described in [21]. Like the Spoofing Detection systems, it uses channel-wise attentive pooling, 192-dimensional embedding, and a Subcenter-AAM-Softmax output layer with margin=0.2. The network was trained on VoxCeleb2 using 2-second chunks and an effective batch size of 256. We used Adam optimizer with a maximum learning rate of 0.01 warmed-up for 15k steps. After 65k steps, the learning rate was halved every 40k steps. Following, the network was fine-tuned on 4-second chunks using hard-prototype mining [22] and increasing the margin to 0.3 and Inter-Top margin=0.1 with K=5 [23]. Finetuning used SGD optimizer with lr=0.001 and momentum=0.9. The learning rate was warmed up for 8k steps, held 8k steps, and halved every 32k steps.

Trials were evaluated just by doing cosine scoring between enrollment and test segments.

4.4. Calibration and Fusion

Spoofing detection scores were calibrated using logistic regression on the ASVSpoof2024 Dev data. Speaker verification scores were calibrated on the ASVSpoof2024 Dev bonafide trial list, excluding spoofing trials. We used effective priors for calibration as outlined in Section 3. We also explored fusing multiple spoofing detection systems, but did not observe significant improvements in either the Development or Progress sets compared to the best single system, FwSE-ResNet34.

5. Results

5.1. Track 1

Table 3 presents the results of different systems on ASVSpoof Dev, Progress, and Eval sets. We tested three sizes of ECAPA-TDNN, ranging from three 256-dim. layers to four 1024-dim layers. No advantage was observed with larger network sizes, likely due to the limited training data. Attempts to mitigate overfitting by increasing dropout were unsuccessful. The S4 model performed comparably to the ECAPA-TDNN with four 1024-dimensional layers. ConvNeXt outperformed ECAPA-TDNN, with ConvNeXt Pico showing slight improvements over ConvNeXt Atto, particularly in terms of Equal Error Rate (EER). However, the EER on the Development set did not reliably predict performance on the Progress set, making minDCF a more accurate indicator. Thin-ResNet34 performed better than ConvNeXT in terms of DCF and worse in terms of EER. Apparently, ResNet34 was less impacted by overfitting than previous architectures, allowing us to scale up the size $\times 4$ from Thin-ResNet34 to FwSE-ResNet34, significantly outperforming all previous networks on Dev and Progress sets. In general, it seems that 2d convolution networks outperform other types of models in this task.

Our experiments with codec augmentation were unsuccessful on the dev set, always degrading performance w.r.t. systems

[2] http://www.openslr.org/resources/17
[3] http://www.openslr.org/resources/28
[4] https://pytorch.org/audio/stable/index.html
[5] https://github.com/iver56/audiomentations
[6] https://github.com/RoyJames/room-impulse-responses

Table 3: *Results of individual systems on ASVSpoof 2024 Track 1 Development and Progress/Evaluation. Augmentations are Noise (N), Reverberation (R), Telephone Codecs (TC), Media Codecs (MC), Telephone+Medica Codecs (TMC).*

System		ASVSpoof2024 Dev.				ASVSpoof2024 Progress/Eval			
NNet	Augment.	minDCF	actDCF	Cllr	EER	minDCF	actDCF	Cllr	EER
Progress Phase									
S4	R+N	0.213			12.96	0.469	0.500	0.725	18.40
ECAPA-TDNN 256x3	R+N	0.169	0.169	0.345	11.91				
ECAPA-TDNN 512x3	R+N	0.194	0.194	0.355	11.51				
ECAPA-TDNN 1024x4	R+N	0.218	0.219	0.383	11.94				
ConvNext2d Atto	R+N	0.165	0.171	0.359	11.56				
ConvNext2d Pico	R+N	0.160	0.161	0.300	9.94				
ThinResNet34	R+N	0.155	0.156	0.337	12.09	0.288	0.329	0.470	12.77
FwSE-ResNet34	R+N	0.139	0.139	0.295	10.625	0.203	0.386	0.524	8.68
FwSE-ResNet34	R+N+TC	0.148	0.148	0.299	10.64	0.181	0.493	0.673	7.92
FwSE-ResNet34	R+N+MC	0.151	0.151	0.311	11.17	0.174	0.304	0.438	6.89
Eval Phase									
RawNet2 (B01)						0.827	0.992	4.094	36.04
AASIST (B02)						0.711	0930	4.001	29.12
FwSE-ResNet34	R+N+TMC	0.146	0.147	0.301	10.768	0.439	0.633	0.853	17.09

Table 4: *minDCF on ASVSpoof 2024 Track 1 Eval break down: attacks vs codecs*

minDCF	pooled	-	codec-1	codec-2	codec-3	codec-4	codec-5	codec-6	codec-7	codec-8	codec-9	codec-10	codec-11
pooled	0.43902	0.11552	0.36221	0.30136	0.36489	0.58699	0.17203	0.34829	0.61012	0.45044	0.55084	0.58544	0.20854
A17	0.12073	0.00238	0.05582	0.04579	0.10364	0.18785	0.05728	0.09298	0.21407	0.15664	0.08707	0.13375	0.01135
A18	0.26011	0.00462	0.12911	0.11319	0.23931	0.35899	0.04687	0.17427	0.39729	0.29418	0.46023	0.49395	0.04842
A19	1.00000	0.99983	1.00000	1.00000	1.00000	1.00000	0.99953	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
A20	0.88272	0.20024	0.71056	0.62973	0.53405	1.00000	0.21344	0.54258	1.00000	0.72430	0.93500	0.82739	0.33669
A21	0.12632	0.00011	0.03685	0.02754	0.12320	0.24757	0.03483	0.08065	0.25752	0.17913	0.06956	0.10375	0.00589
A22	0.26349	0.01162	0.16125	0.14406	0.30857	0.46063	0.18418	0.21456	0.42658	0.20466	0.23884	0.29684	0.03871
A23	0.23259	0.00470	0.10802	0.07223	0.18266	0.37189	0.04312	0.14191	0.41397	0.22025	0.20003	0.24253	0.03202
A24	0.21775	0.00493	0.14949	0.09330	0.31428	0.36716	0.06417	0.06178	0.38493	0.27066	0.34166	0.47328	0.04524
A25	0.28885	0.00012	0.18628	0.22180	0.29273	0.78529	0.10493	0.12989	0.80810	0.30049	0.34760	0.27354	0.00505
A26	0.39640	0.01582	0.30882	0.28902	0.39947	0.62565	0.16715	0.25557	0.64512	0.52960	0.74689	0.76463	0.13869
A27	0.49597	0.03058	0.44808	0.29657	0.29546	0.57074	0.05037	0.36644	0.64002	0.67656	0.89168	0.82937	0.23279
A28	0.46131	0.06560	0.35920	0.36657	0.46074	0.62820	0.30518	0.39011	0.63350	0.50328	0.58438	0.73438	0.24889
A29	0.03620	0.00102	0.01043	0.00349	0.01665	0.06393	0.00269	0.00290	0.09057	0.09508	0.01231	0.03153	0.00421
A30	0.50422	0.03153	0.40225	0.23806	0.35832	0.64631	0.08674	0.40136	0.72361	0.53946	0.75415	0.77992	0.17138
A31	0.55579	0.05705	0.40128	0.23085	0.40288	0.65010	0.12079	0.43174	0.68561	0.44386	0.64385	0.72412	0.24023
A32	0.53126	0.03275	0.48909	0.28248	0.30981	0.74528	0.05326	0.38236	0.80361	0.56364	0.71241	0.65610	0.17139

without codecs. Eventually, we submitted systems with telephone (TC) and media (MC) codecs to the Progress set, observing significant gains for both. Media codecs achieved the best performance. Unfortunately, we did not have time to submit the systems trained on a combination of telephone and media codecs (TMC) to the Progress set. Nevertheless, we thought that TMC would be the best option and submitted this system to the Eval phase, achieving minDCF 47% better than the organizer baseline AASIST.

We detected a significant mismatch between Dev and Progress/Eval sets. This rendered the Dev set useless once we achieved minDCF≤ 0.15. This mismatch also resulted in badly calibrated systems. We can observe in the table that actDCF and Cllr become worse as we improve minDCF and EER.

Table 4 breaks down minDCF per attack methods, and codec type in the Eval set for the FwSE-ResNet34. The tables for actDCF, Cllr, and EER were similar. Attack and Codecs are described in the challenge summary paper [1]. Attacks A01-8 were included in the Training set, A09-16 in the Dev set, and A17-32 in the Eval set. Therefore, the Eval attacks were all unseen at training. This causes a large disparity in the performance for different attack types. A17(ZMM-TTS), A21(BigVGAN), A29(XTTS) have pooled minDCF< 0.12. These systems all have a vocoder based on HiFi-GAN, the same as A08 (VITS) included in the training, which could explain this good performance. Meanwhile, A20 (in-house unit selection) has minDCF=0.88 and A19 (MaryTTS) renders the system useless with minDCF=1. A19-20 are both classical concatenative unit selection systems. Meanwhile, the training attacks were all deep-learning-based TTS models. Another set of attacks with high minDCF~ 0.5, are A28,30,31,32 that use the Malacopula adversarial attack (AT) on top of TTS or VC. The equivalent TTS and VC methods without AT are A26,18,22,25, with minDCFs of 0.26-0.39. Thus, the AT increased minDCF by 17-115% relative.

Regarding codec types, the training and development sets did not include any codecs other than those we simulated. Our system performed exceptionally well on non-codec trials, achieving a pooled minDCF of 0.11, with only failing in A19. It also showed decent performance with codec 5 (MP3), yielding a minDCF of 0.187, and codec 11 (varied 8 kHz telephone), with a minDCF of 0.20. This performance can be attributed to augmenting with MP3 and varied 8 kHz telephone codecs during training. However, the MP3 codecs used during training had higher bit rates compared to those in the Eval set, as low bit rate codecs negatively impacted performance in the Dev

Table 5: *Results of individual systems on ASVSpoof 2024 Track 2 Development and Evaluation*

System			ASVSpoof2024 Dev.			ASVSpoof2024 Eval		
Spk-NNet	Spoof-NNet	Spoof-Augment.	a-DCF	t-DCF	t-EER	a-DCF	t-DCF	t-EER
(B03) ECAPA-TDNN	AASIST					0.681	0.929	28.78
FwSE-ResNet100	ConvNeXt Pico	R+N	0.143	0.242	6.24			
FwSE-ResNet100	FwSE-ResNet34	R+N	0.132	0.122	6.39			
FwSE-ResNet100	FwSE-ResNet34	R+N+TC	0.134	0.220	6.47			
FwSE-ResNet100	FwSE-ResNet34	R+N+MC	0.135	0.223	6.74			
FwSE-ResNet100	FwSE-ResNet34	R+N+TMC	0.134	0.219	6.53	0.397	0.700	15.09

set. Codecs 1 (Opus), 2 (ARM-WB), 3 (Speex), and 6 (MP4) exhibited intermediate performance, with minDCF values ranging from 0.30 to 0.36. Among these, only Opus was included in training. However, AMR-WB (G722.2) has higher quality than G722, seen in training, so that could have helped to avoid further degradation. The codecs that degraded performance more were 4 (Encodec), 7 (MP3+Encodec), 9 (AMR 8 kHz), and 10 (Speex 8 kHz), each with a minDCF greater than 0.55. Encodec was not included in training, highlighting that new deep-learning codecs can significantly impair spoofing detection systems. Additionally, for 8 kHz codecs, we only considered typical telephone codecs like A-law, μ-law, G723, and G726, without including narrow-band versions of wide-band codecs (AMR, Speex), which might explain some of the observed degradation.

We also observe that there are complicated interactions between attack methods and codecs. While A19 performed badly for any codec, A20 had decent performance for no-codec and codec 5 (MP3) but had very bad performance when paired with codecs 1, 7, 8, 9, and 10. Also, A25 goes from minDCF=0.00012 without codec to 0.78 with codec 4 (Encodec).

5.2. Track 2

Table 5 presents the results for Track 2, where speaker and spoofing detection systems are combined into a single system. We observed a significant performance degradation in the Eval set compared to the Dev set, likely due to mismatches in attack types and codecs between the two sets. As mentioned in the previous section, these mismatches lead to miscalibration in the spoofing detection scores. Since our combination method, described in Section 3, relies on calibrated speaker and spoofing detection scores, such miscalibration can notably impact the a-DCF. Nevertheless, our system achieved a 44% improvement over the baseline B03.

6. Conclusion

This paper describes the SHADOW team's submission to ASVSpoof 2024. Our systems were built on networks commonly used for speaker verification, such as ECAPA-TDNN and ResNet34. We found that 2D convolution-based models outperformed 1D convolution models and state-space models (SSMs). FwSE-ResNet34 emerged as our best-performing system, and we did not achieve any gains from combining it with other systems. To integrate speaker and spoofing detection systems, we propose a straightforward method that calculates the posterior probability of a trial being both target and bonafide, using probability rules and well-calibrated scores. The substantial mismatch between the Dev and Eval sets complicated model tuning, particularly in selecting the optimal model and codec combination. We observed a strong dependency between codecs and performance; trials without codecs or codecs seen during training performed significantly better than unseen codecs. We also noted that unit selection TTS become undetected by spoofing detectors training on current deep-learning TTS models. Despite these challenges, our models achieved a notable relative improvement of 44-47% over the challenge baselines.

6.1. Acknowledgments

Tiantian Feng, Jihwan Lee and Shrikanth Narayanan are partially supported by NSF.

7. References

[1] X. Wang et al., "ASVspoof 5: Crowdsourced data, deepfakes and adversarial attacks at scale," in *ASVspoof 2024 workshop (submitted)*, 2024.

[2] Cheng-I Lai, Nanxin Chen, Jesús Villalba, and Najim Dehak, "ASSERT: Anti-Spoofing with Squeeze-Excitation and Residual Networks," in *Proc. Interspeech 2019*, 2019, pp. 1013–1017.

[3] 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)*, 2022, pp. 6367–6371.

[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)*, 2021, pp. 6369–6373.

[5] B. Desplanques, J. Thienpondt, and K. Demuynck, "ECAPA-TDNN: Emphasized Channel Attention, Propagation and Aggregation in TDNN Based Speaker Verification," in *Interspeech 2020*, 2020.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[7] Jenthe Thienpondt, Brecht Desplanques, and Kris Demuynck, "Integrating Frequency Translational Invariance in TDNNs and Frequency Positional Information in 2D ResNets to Enhance Speaker Verification," in *Proc. Interspeech 2021*, 2021, pp. 2302–2306.

[8] Sanghyun Woo, Shoubhik Debnath, Ronghang Hu, Xinlei Chen, Zhuang Liu, In So Kweon, and Saining Xie, "Convnext v2: Co-designing and scaling convnets with masked autoencoders," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 16133–16142.

[9] Albert Gu, Isys Johnson, Karan Goel, Khaled Saab, Tri Dao, Atri Rudra, and Christopher Ré, "Combining recurrent, convolutional, and continuous-time models with linear state space layers," *Advances in neural information processing systems*, vol. 34, pp. 572–585, 2021.

[10] Xin Wang, Tomi Kinnunen, Lee Kong Aik, Paul-Gauthier Noe, and Junichi Yamagishi, "Revisiting and Improving Scoring Fusion for Spoofing-aware Speaker Verification Using Compositional Data Analysis," in *Proc. Interspeech*, 2024, p. (accepted).

[11] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-Vectors : Robust DNN Embeddings for Speaker Recognition," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018*, Alberta, Canada, apr 2018, pp. 5329–5333, IEEE.

[12] Shang-Hua Gao, Ming-Ming Cheng, Kai Zhao, Xin-Yu Zhang, Ming-Hsuan Yang, and Philip Torr, "Res2Net: A New Multi-Scale Backbone Architecture," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 2, pp. 652–662, Feb 2021.

[13] J. Hu, L. Shen, and G. Sun, "Squeeze-and-Excitation Networks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7132–7141.

[14] J. Deng, J. Guo, T. Liu, M. Gong, and S. Zafeiriou, "Sub-center ArcFace: Boosting Face Recognition by Large-Scale Noisy Web Faces," 10 2020, pp. 741–757.

[15] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie, "A convnet for the 2020s," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 11966–11976.

[16] Albert Gu, Karan Goel, and Christopher Ré, "Efficiently modeling long sequences with structured state spaces," *arXiv preprint arXiv:2111.00396*, 2022.

[17] Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré, "Hippo: Recurrent memory with optimal polynomial projections," *Advances in NeurIPS*, vol. 33, pp. 1474–1487, 2020.

[18] Dan Hendrycks and Kevin Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

[19] Kleanthis Avramidis, Dominika Kunc, Bartosz Perz, Kranti Adsul, Tiantian Feng, Przemysław Kazienko, Stanisław Saganowski, and Shrikanth Narayanan, "Scaling representation learning from ubiquitous ecg with state-space models," *IEEE Journal of Biomedical and Health Informatics*, 2024.

[20] Karol J Piczak, "Esc: Dataset for environmental sound classification," in *Proceedings of the 23rd ACM international conference on Multimedia*, 2015, pp. 1015–1018.

[21] Nikita Torgashov, Rostislav Makarov, Ivan Yakovlev, Pavel Malov, Andrei Balykin, and Anton Okhotnikov, "The id rd voxceleb speaker recognition challenge 2023 system description," 2023.

[22] J. Thienpondt, B. Desplanques, and K. Demuynck, "Cross-Lingual Speaker Verification with Domain-Balanced Hard Prototype Mining and Language-Dependent Score Normalization," in *Proc. Interspeech 2020*, 2020, pp. 756–760.

[23] Miao Zhao, Yufeng Ma, Yiwei Ding, Yu Zheng, Min Liu, and Minqiang Xu, "Multi-query multi-head attention pooling and inter-topk penalty for speaker verification," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 6737–6741.