IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 10 July 2023, accepted 28 September 2023, date of publication 4 October 2023, date of current version 11 October 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3321919

RESEARCH ARTICLE

RawSpectrogram: On the Way to Effective Streaming Speech Anti-Spoofing

PETR GRINBERG AND VLADISLAV SHIKHOV
Samsung R&D Institute Russia (SRR), 127018 Moscow, Russia
Corresponding author: Vladislav Shikhov (v.shikhov@samsung.com)

ABSTRACT Traditional anti-spoofing systems cannot be used straightforwardly with streaming audio because they are designed for finite utterances. Such offline models can be applied in streaming with the help of buffering; however, they are not effective in terms of memory and computational consumption. We propose a novel approach called RawSpectrogram that makes offline models streaming-friendly without a significant drop in quality. The method was tested on RawNet2 and AASIST, resulting in new architectures called RawRNN (RawLSTM and RawGRU), RS-AASIST, and TAASIST. The RawRNN-type models are much smaller and achieve a better Equal Error Rate than their base architecture, RawNet2. RS-AASIST and TAASIST have fewer parameters than AASIST and achieve similar quality. We also proved our concept for models with time-frequency transform front-ends and automatic speaker verification systems by proposing RECAPA-TDNN based on ECAPA-TDNN. RS-AASIST and RECAPA-TDNN were combined into the first streaming-friendly spoofing-aware speaker verification system reported in the literature. This joint system achieves significantly better quality than the corresponding offline solutions. All our models require far fewer floating-point operations for score updates. RawSpectrogram usage significantly reduces the latency of the prediction and allows the system to update the probability with each new chunk from the stream, preserving all information from the past. To the best of our knowledge, TAASIST is the most successful voice anti-spoofing system that employs a vanilla Transformer trained using supervised learning.

INDEX TERMS Anti-spoofing, ASVspoof challenge, automatic speaker verification system, countermeasure, SASV challenge, spoofing-aware speaker verification system, streaming.

I. INTRODUCTION

Technological advancement increases the diversity of methods for falsification and gaining unauthorized access. Playbacks of recorded speech and modern algorithms for synthesized speech can be roughly detected by a casual user, and criminals can use them effectively. The fraud calls from a person who pretends to be a bank or government employee allow offenders to steal money from accounts and sensitive information from documents. The integration of voice assistants in smart home systems leads to burglars gaining full access to apartment devices. Audio spoofing attacks are classified into four groups [1]: impersonation, text-to-speech (TTS), voice conversion (VC), and replay. Security improvements must be implemented to effectively counter each type of attack.

Voice biometric systems consist of an automatic speaker verification system (ASV) and countermeasure system (CM). The ASV protects against impersonation attacks by distinguishing the bona-fide speech of the information owner from an impostor's voice. The CM is added to make the ASV defensible against other types of attacks. A combination of these two systems is called a spoofing-aware speaker verification (SASV) system. It is a complete biometric algorithm that covers all attacks. This research area is highly supported by the ASVspoof initiative, which created data and comparison standards and organized several competitions [2], [3], [4], [5], [6].

Existing CMs and ASVs mostly operate with full-length or very long (2-4 seconds) utterances. In fraud call and voice assistant scenarios, the final length of the audio is

The associate editor coordinating the review of this manuscript and approving it for publication was Cong Pu.

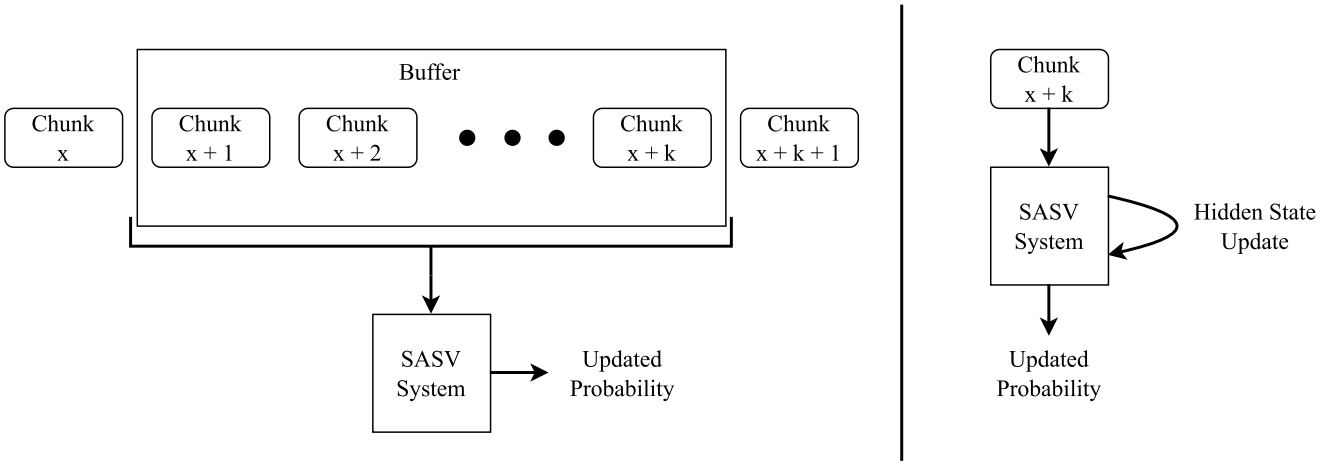

FIGURE 1. Spoofing probability update for offline solution with buffering (left) and native-streaming (right).

unknown and can be infinite. Therefore, a system should be able to continuously update its predicted probability of audio spoofing. Moreover, the latency before a model decision should be as low as possible, and two or four seconds may be too long for business requirements. Thus, the traditional way of applying biometric systems differs from that needed for streaming audio, and conventional ASVs and CMs are not designed for such use.

We classify approaches for the creation of streaming-friendly SASV systems into two general categories:

1) **Offline solution with audio buffering**. The model prediction is based on a buffer that is updated with every chunk from the stream by removing old data from the head and adding new data to the tail. See Fig. 1 (left) for visualization.
2) **Native streaming**. The system processes only the newest chunk, and the output probability depends only on this new segment and the hidden state from the past. See Fig. 1 (right) for visualization.

The audio-buffering approach consumes considerable computational and memory resources because it requires maintaining big data and reprocessing it fully every time. Several optimization techniques can reduce the complexity of the solutions. For example, for the convolution calculation, an input is split into several subparts, some of which will not change after the buffer update; therefore, there is no need to count the results on these unchanged parts again. Whereas the first category uses traditional systems with buffers, the second applies models specifically designed for audio stream applications. Native streaming solutions aim to work with as small input as possible (much shorter than an ordinary buffer) to quickly and frequently update the output score. Moreover, these systems do not have an audio buffer; the information from the past is preserved in a single object that consumes less memory than a usual buffer and knows about all previous chunks and not just the latest ones. Native streaming models can have significantly faster updates of the predicted probability. Indeed, regardless of the length of the chunk w, offline models are applied to the buffer of length b for each score update. Native streaming solutions process chunks in a straightforward manner; therefore, the length of their input is always equal to w. If $w \ll b$, which is usually true, offline solutions will require substantially more floating-point operations for an update.

An offline model does not have any structural restrictions and is therefore likely to be more robust than a native streaming model. However, we argue that the latter is more intuitive and, with a proper design, can be much more lightweight while having similar anti-spoofing capabilities. Our paper focuses on the development of a streaming-friendly and resource-effective SASV system. We propose a novel approach for audio processing, referred to as RawSpectrogram. It allows the creation of representations that adjust to a given task and enables effective native streaming. For the CMs, there are two corresponding deep neural network (DNN) models that use RawSpectrogram: RawRNN (RawLSTM and RawGRU) and RS-AASIST based on RawNet2 [7] and AASIST [8], respectively. We also propose a model named TAASIST in which the recurrent neural network from RawSpectrogram (see Section III) is replaced with Transformer [9]. To the best of our knowledge, this is the most successful usage of a vanilla Transformer trained via supervised learning in voice anti-spoofing. This model is not a native streaming model and is designed to show how RawSpectrogram enables the easy optimization of computations for offline solutions with buffering. All our proposed CMs have fewer parameters than their base offline architectures and update probabilities significantly faster, preserving robustness. We prove that our concept also works for ASV systems and models with non-raw front-ends by proposing a RECAPA-TDNN model based on ECAPA-TDNN [10] with 1024 channels. The resulting solution is smaller than the base architecture, updates the output score more than two times faster, and has a quality that is better

than that of ECAPA-TDNN with 512 channels. Finally, we combine RS-AASIST and RECAPA-TDNN into the first-ever native-streaming SASV system.

The remainder of this paper is organized as follows. Section II describes the current state of the considered research area. RawSpectrogram with RawRNN, RS-AASIST, TAASIST, and RECAPA-TDNN are introduced in Section III. The setup of the corresponding experiments is described in Section IV. The results and conclusion are in Sections V and VI, respectively.

II. RELATED WORK

In this section, we describe RawNet2, AASIST, ECAPA-TDNN, and streaming-friendly models from literature.

A. BASE CMS

RawNet2 [7] is the standard baseline architecture for CMs that work with raw input. The main idea behind it is to use a Sinc-layer [11] instead of a conventional convolution. In RawNet2, a Sinc-layer is followed by several ResBlocks [12] with Feature-Map-Scaling (FMS, [13]), Gated Recurrent Unit (GRU, [14]), and two Fully-Connected layers (FC-layers).

The state-of-the-art CM is AASIST, which was proposed in [8]. It is a graph neural network (GNN) consisting of several graph attention (GAT), heterogeneous graph attention (HS-GAL) and graph pooling layers. The model uses a RawNet2-based encoder without FMS and GRU to produce temporal and spectral features. These features are separated and judged as nodes of a complete heterogeneous graph with three types of relations: between time nodes, between frequency nodes, or mixed.

B. BASE ASV

ECAPA-TDNN [10] is a well-known ASV system. It uses Mel-frequency Cepstral Coefficients (MFCCs) as input and applies a 1D-convolution block with several Squeeze-Excitation (SE) blocks and Attentive Statistics Pooling, followed by an FC-layer. The modules are designed to capture a wider temporal context, which is beneficial for extracting speaker properties. In our experiments, we used a publicly available ECAPA-TDNN variant from [15].[1] It has a slightly different training scheme and replaces the MFCCs with a Mel spectrogram.

C. NATIVE STREAMING

There are three statements regarding native streaming models: (i) they are applied successively to each new chunk from the stream; (ii) the complexity of processing one segment depends only on its length; and (iii) the model output relies on the hidden state that preserves information from the past. Recurrent neural networks (RNN) exactly fit these criteria. Thus, the CM and ASV systems that properly use RNN can be considered streaming-friendly, even though they were originally developed as offline models.

The authors of [16] suggested replacing linear layers in GRU with LightCNN [17] (LCNN) and proposed the LC-GRU architecture. The latter system can be used for native streaming. Indeed, LC-GRU works with a log-magnitude spectrogram created by overlapping windowing over utterance. We can create spectrogram frames from each window independently and process them sequentially, sharing the hidden state of the GRU architecture. A similar approach can be applied to 3-layer LSTM (Long Short-Term Memory, [18]) and GRU from [19]. Unlike these studies that develop one architecture, we propose an approach that creates a native streaming model from any offline one.

RawNet2 has convolutions before GRU, which are applied to a full-length utterance. It could be possible to optimize convolution and calculate it only on a new chunk, thereby making RawNet2 streaming-friendly. However, FMS creates a dependency between time segments, meaning that the new chunk affects previous computations. AASIST does not have RNN-type modules, and its complete inner graph creates dependencies between all temporal nodes. Thus, both CMs do not correspond to the definition of native streaming. Therefore, it is reasonable to test our method on them.

For the ASV, RawNet [20] with optimized convolutions can be considered streaming-friendly because it does not have FMS as its advanced version, RawNet2 [13]. The SE-blocks and attentive statistic pooling layer in ECAPA-TDNN create the same problem as FMS for RawNet2, meaning that ECAPA-TDNN cannot be used as a native-streaming solution.

D. SASV

The existing solutions for SASV architectures can be divided into three types:

1) Fusion over scores or embeddings of independent ASV and CM systems.
2) A cascade approach, in which the two subsystems are stuck after each other and the output of the second model is constrained by the first one.
3) End-to-End systems.

There are several papers describing the second [21], [22] and the third [23], [24] methods; however, the first method usually outperforms others, although it is much simpler in implementation. The fusion approach was described in [21], [22], [25], [26], [27], [28], and [29]. In this paper, we consider the method from [25], as it does not require extra DNNs and thus preserves the effective resource consumption of our proposed systems.

Having a pretrained CM and ASV system, the authors of [25] suggest using a probabilistic framework based on the product rule. The SASV score is computed according to the following (1):

$$S_{\text{SASV}} = \sigma(S_{\text{CM}}) \cdot f(S_{\text{ASV}}), \tag{1}$$

[1] Implementation is available at https://github.com/TaoRuijie/ECAPA-TDNN.

where S_{CM} is the output score from the CM system, S_{ASV} is the cosine similarity score from the ASV system, $\sigma(x)$ is a sigmoid function, and $f(x)$ can be any function that monotonically maps the cosine similarity score to $[0, 1]$. We consider two possible f functions from [25]. The first is a linear mapping: $f(x) = \frac{x+1}{2}$. The second function is sigmoid. The authors of [25] also proposed the use of a calibration function; however, its performance was poor owing to overfitting. There is also a version of (1) in which S_{CM} is conditioned on the ASV output. This is done via fine-tuning the final FC-layer in the CM system by optimizing the joint score.

III. RAWSPECTROGRAM

In this section, we define RawSpectrogram, RawRNN, RS-AASIST, TAASIST, and RECAPA-TDNN.

A. METHOD DESCRIPTION

A straightforward approach for working with streaming audio is to process it chunk-by-chunk. However, each speech segment should have an informative representation before passing to the network, and we should be able to capture global and local contexts in the frequency and time domains to achieve superior spoof determination capabilities. We propose the following method to comply with the listed requirements (see Fig. 2 for visualization).

First, we obtain overlapping windows with a fixed window length W and hop size H from the audio stream. Each segment is fed to an Embedding Network (EmbNet) to obtain discriminative features. This idea is similar to the Short-Time Fourier Transform (STFT) front-end: windowing is required for the determination of artifacts in the time domain, and EmbNet procures frequency information. EmbNet can be any trainable DNN that is compatible with raw audio input. If we compose the frequency-describing embeddings of each window into one matrix, we will obtain a spectrogram-like object from the raw chunks: rows and columns depict changes in the time and frequency domains, respectively. This interpretation leads to the "RawSpectrogram" name of the method. Apart from other learnable STFT-like representations [30], [31], [32], [33], we have more degrees of freedom by choosing the architecture for EmbNet and learning an end-to-end embedding transform. The temporal convolutions in Wav2Vec [34] and Wav2Vec 2.0 [35] are applied with the given stride and, therefore, make windowing, but the concept and learning strategy of these models completely differ from our approach.

Second, the embeddings from EmbNet serve as input for the RNN model to enable native streaming and capture the global temporal context. The use of a unidirectional LSTM or GRU is recommended to avoid vanishing gradients.

Finally, the output from the last RNN cell is sent to a binary classifier that predicts the probability of audio being spoofed. The choice of architecture for this part is unlimited.

In the inference phase, we obtain a new window from the stream, calculate its embedding via EmbNet, feed it to the RNN cell, conserving the hidden state from the previous segment, and make a probability prediction, pushing the RNN output to the classifier.

TABLE 1. EmbNet for the RawRNN architecture. It is based on the RawNet2 model. For the convolution layers, *k* is the kernel size and *f* is the number of filters.

Layer	Description
Fixed Mel-Scaled Sinc filters	$\text{Conv}(k = 129, f = 16)$ MaxPool(3) BN & SELU
ResBlock-0	BN & LeakyReLU(0.3) $\text{Conv}(k = 3, f = 16)$ BN & LeakyReLU(0.3) $\text{Conv}(k = 3, f = 16)$ MaxPool(3) FMS
ResBlock-1	$\text{Conv}(k = 3, f = 64)$ BN & LeakyReLU(0.3) $\text{Conv}(k = 3, f = 64)$ MaxPool(3) FMS
GRU	BN & GRU(256)
FC	FC(256)

EmbNet allows the preservation of the effectiveness of the original offline model. The task of capturing the temporal context is transmitted to a powerful RNN that can be relatively small, thereby reducing memory and computational costs. Thus, the RawSpectrogram approach leads to native streaming models with the quality of the original offline solutions and less resource consumption. W controls the amount of temporal context in one window and the delay before the first prediction. H regulates how frequently we can update the probability and amount of shared temporal context between adjacent segments. Both parameters define the overall latency of chunk processing.

It should be noted that when native streaming is not the objective, EmbNet embeddings can be stacked in a matrix and work as a learnable front-end for offline models.

B. RAWRNN AND RS-AASIST

For a feasible study, we chose the two most impactful models applicable to raw input: RawNet2 and AASIST. These models can be converted into a native streaming format by setting them as EmbNet in RawSpectrogram. We used a 2-layer MLP with BatchNorm, LeakyReLU, and the hidden size equal to 256 as the classifier.

For RawRNN (RawLSTM or RawGRU), EmbNet is a smaller version of RawNet2 with mel-scaled sinc filters. We changed the hyperparameters, removed the last FC-layer from the original RawNet2 architecture, and left only the first two residual blocks. A complete description of the system is presented in Table 1. The W and H are set to 512 and 256, respectively. RawLSTM has two layers of LSTM with the 0.3 dropout probability and the hidden size equal to 256.

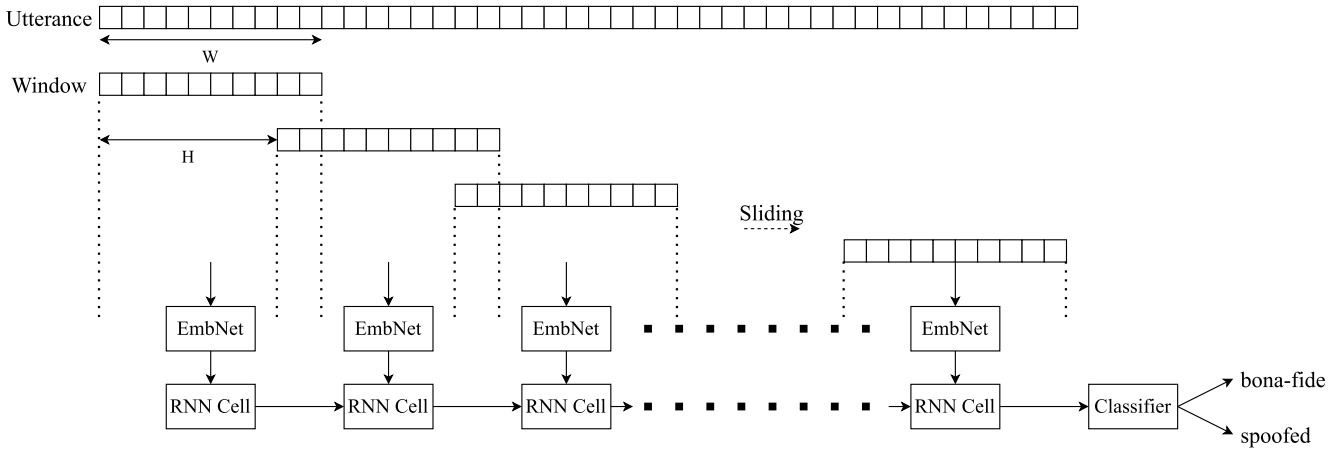

FIGURE 2. RawSpectrogram Pipeline.

RawGRU has only one layer of GRU with the hidden size equal to 100.

Setting AASIST as EmbNet and pushing each embedding to RNN makes the AASIST architecture recurrent (RS-AASIST, RawSpectrogram-AASIST). For this model, we also reduced the size of the system by removing the last two residual blocks and the last classification FC-layer. The only hyperparameters we have changed were the two last graph pool ratios and temperatures, which were decreased from 0.5 to 0.3 and from 100 to 80, respectively. The composition of the node-wise maximum and average, together with the stack node from the EmbNet part, is fed to a 1-layer GRU with the hidden size equal to 32. We tested several W and H for RS-AASIST and found that $W = 4096$, $H = 2048$ worked the best. Such large windowing parameters are caused by the structure of the original AASIST, which requires a big temporal input length. A long window allows the model to preserve the context and informative size of temporal graphs in the GAT and HS-GAL parts of AASIST.

C. TAASIST

The TAASIST (Transformer-AASIST) architecture is similar to RS-AASIST; however, the GRU layer is replaced with a vanilla Transformer Encoder [9]. The output of EmbNet is considered as token embeddings. We add an extra classification token to the start of the sequence, which representation after Encoder is passed to the classifier. Transformer has two heads, 80-dimensional embeddings, the feedforward size equal to 80, the dropout set to 0.2, and one inner layer. We also used pre-normalization for faster convergence [36], GELU activation [37], and a learnable codebook for positional encoding. The EmbNet architecture is the same as that in RS-AASIST; however, the number of output channels in the last two residual blocks is reduced from 64 to 48. The 160-dimensional outputs of EmbNet are projected to 80-dimensional Transformer inputs using an FC-layer. W and H are identical to those of RS-AASIST.

There has been research on the combination of Transformer and AASIST. In [38], the authors replaced the RawNet2 encoder in AASIST with Wav2Vec 2.0 [35], which is based on the Transformer architecture. However, this solution uses self-supervised learning and pre-training on external data, whereas we consider only supervised learning and the datasets allowed in the ASVspoof Challenge.

The streaming application of TAASIST is done with audio buffering. However, computations can be easily optimized because there is no need to recompute representations from EmbNet for chunks that have already been seen. This is also more efficient in terms of memory because we conserve N 80-dimensional EmbNet embeddings instead of a raw audio buffer of size $N \cdot H + W$ where N denotes the length of the token sequence for Transformer.

D. RECAPA-TDNN

Our native streaming ASV system is Recurrent ECAPA-TDNN (RECAPA-TDNN), which is a conversion of ECAPA-TDNN [10] via RawSpectrogram. It differs from the CM systems proposed above. If all EmbNet networks in our CM systems operate on raw chunks, this system uses a Mel Spectrogram as front-end. The application of the time-frequency transform requires a significant increase in the window length W. Indeed, the transform itself is STFT-like and needs a large input time dimension to produce useful features for DNN. We used W equal to 24120 and H set to 12060.

We noticed that RECAPA-TDNN tends to overfit if we only change the hyperparameters for EmbNet. Therefore, we decided to use another approach to make EmbNet smaller. First, EmbNet was set as the pretrained ECAPA-TDNN with 1024 channels. Then we pruned 20% of the output channels for each convolution layer and input features for the fully-connected layers. To avoid feature dimension mismatches between successive layers, we added padding and pooling. Each convolution is followed by a padding layer that adds fictitious zero channels to the output, replacing

those that were pruned. The pooling layer is located before the FC-layer and removes the activations in the input that correspond to the pruned features. RECAPA-TDNN has one GRU layer with 800 hidden units. The classifier consists of BatchNorm, an FC-layer with 192 output features, and another BatchNorm.

IV. EXPERIMENTAL SETUP

In this section, we describe the training scheme and experiments for analyzing the anti-spoofing capabilities of RawRNN, RS-AASIST, TAASIST, and RECAPA-TDNN for streaming speech.

A. MODEL TRAINING

The training scheme for RS-AASIST was identical to that of the original AASIST paper [8], except that we used gradient accumulation. The batch size was set to five, and the optimization step was done once in five sequential steps. TAASIST was trained in a similar way but used 1Cycle learning rate scheduler [39] with 5000 steps per epoch and the maximum learning rate set to 10^{-4}. The learning rate increased for the first 10% of the cycle. We also replaced the Adam [40] optimizer with AdamW [41].

For RawRNN, we used the Adam optimizer with a weight decay and learning rate both equal to 10^{-4}. The learning rate was divided by a factor of 2 every 2540 steps. The model was trained for 60 epochs, with 127 steps in each. The batch size was set to 200. The loss function was Weighted Cross Entropy with weights equal to 0.1 and 0.9 for spoof and bona-fide classes, respectively.

We used several data augmentation methods from PyRubberBand[2] that were sequentially applied in the following order:

- Pitch shift by random semitones between -3 and 3. The probability of augmentation was set to 0.3.
- Time stretch by a random factor between 0.5 and 2. The probability of augmentation was set to 0.3.

All CMs were trained on the training partition of the LA ASVspoof 2019 dataset [4], [42]. For each audio, we defined a fixed number of windows (128 for RawRNN, 40 for RS-AASIST and TAASIST) created using the RawSpectrogram method. We did this by padding with duplication or randomly cropping each utterance. However, audios are not pre-processed in the evaluation stage, and all our models operate with varying numbers of windows (for TAASIST in the offline mode, utterances are cut to have 40 chunks or less).

The training scheme for RECAPA-TDNN was similar to that in [15]. We used a pretrained ECAPA-TDNN for EmbNet (see Section III-D) and pretrained parameters for AAM-Softmax [43], [44], which were also taken from the pretrained ECAPA-TDNN. The learning rate was set to 10^{-4} for the pretrained parameters and to 10^{-3} for the other parameters, and it was multiplied by 0.97 each epoch. To increase the convergence speed and reduce overfitting, a knowledge distillation [45], [46] technique with Mean-Squared-Error (MSE) over embeddings from teacher and student was used. The pretrained ECAPA-TDNN was a teacher. The final loss function is expressed as follows:

$$\mathcal{L}_{\text{Final}} = (1-\alpha)\cdot\mathcal{L}_{\text{AAM-Softmax}} + \alpha\cdot\mathcal{L}_{\text{MSE}}. \tag{2}$$

The α parameter in (2) controls the amount of distillation. RECAPA-TDNN was trained on the development part of the VoxCeleb2 dataset [47] for seven epochs with the batch size equal to 400 and α set to 0.99. The learning rate was then reduced to 10^{-5} for pretrained parameters and to 10^{-4} for other parameters, the batch size was increased to 500, α was set to zero, and RECAPA-TDNN was trained for eight more epochs. We also trained ECAPA-TDNN with 512 channels for comparison purposes, following the training scheme from [15]. All the considered ASV systems used the augmentation techniques described in [48] and SpecAugment [49].

Our SASV system was constructed identically to [25] using the publicly available implementation.[3] The only difference is that we fine-tuned the entire classifier and not only the last FC-layer. We consider four different models, choosing whether f is a linear or sigmoid function and whether scores from CM and ASV are independent. The proposed SASV systems are named in the following form: RawSpectrogram-X-Y, where X denotes the f function type (L for linear and S for sigmoid) and Y whether to do fine-tuning or not (I – direct inference, F – fine-tune).

B. COMPLEXITY OF A PROBABILITY UPDATE

For the application of systems in streaming, we measured the complexity and speed of a single score update by counting the number of floating-point operations (FLOPs).

Native streaming models straightforwardly process chunks; therefore, we simply pass an input of size W, which is the chunk size, and calculate the FLOPs.

For offline models, we first have to define the size of the buffer b. The most reasonable way is to examine the input length in the evaluation scheme for the considered offline solution. Indeed, if the model is tested on utterances of length b_1, we cannot assume that its quality will be the same for audio of length b_2, where $b_1 \neq b_2$. For example, AASIST was evaluated on 4-second-long utterances (64600 samples, to be exact) in the original paper [8]; therefore, we set b to 64600 for the experiments with AASIST. Likewise, for RawNet2, b is equal to 64000 according to the original paper [7]. When the length of the buffer is defined, we only need to pass a buffer of length b as the input for the model and calculate the FLOPs. It should be noted that b does not depend on the chunk size W.

In Section III-C, we explained how computations for TAASIST can be optimized. Hence, for this architecture, the cost of a single update is equal to the number of FLOPs for the inner Transformer and classifier forward step, summed with the FLOPs for processing one chunk by EmbNet.

[2]https://github.com/bmcfee/pyrubberband, v.0.3.0

[3]https://github.com/yzyouzhang/SASV_PR

C. PROBABILITY VS WINDOWS

The streaming audio produces successive chunks of information. In this experiment, we aimed to understand how probability changes with each new piece of utterance and how this is reflected in the Equal Error Rate (EER). Hence, we applied the classifier to each RNN cell and calculated the probability for each window.

To better capture the difference between offline and streaming-friendly models, we conducted the same experiment for TAASIST, AASIST, and RawNet2 using buffering. We split the audio into chunks, imitating stream, and successively updated buffers. TAASIST works with sequences of length $N = 40$, which means that it requires a raw audio buffer of length $N \cdot H + W$. If the number of chunks is less than N, Transformer uses a padding mask for the missing segments. The W and H for TAASIST were the same as those described in Section III-C. The buffer for AASIST is 4-second-long (64600 samples) because this was the size of the input in the original paper [8]. If the current audio length is shorter than that of the buffer, padding with repetition is used. For this experiment with AASIST, we updated the buffer by imitating chunks with W, H both equal to 1292 (non-overlapping windows). Similarly, W, H, and the buffer size were set to 1000, 1000, and 64000, respectively, for RawNet2.

The original RawNet2 weights are not published on the Internet; therefore, for this experiment, we reproduced the original model using the available implementation.[4]

V. RESULTS

In this section, we analyze the results of the experiments described in Section IV for the CM systems, RECAPA-TDNN, and our streaming-friendly SASV solution.

A. CM SYSTEMS

A comparison of the RawSpectrogram-based models with the original architectures is presented in Table 2 in terms of the EER and minimum tandem detection cost function (min t-DCF, [4], [50]) on the evaluation set of the LA partition of the ASVspoof 2019 dataset. We also show the number of model parameters, FLOPs for processing a 4-second-long utterance (offline mode), and FLOPs for updating the probability with a new chunk from the stream (streaming mode). FLOPs were counted using fvcore.[5] RawLSTM, RawGRU, and RS-AASIST were trained according to Section IV-A. For offline models, the complexity of updating the probability with a new window is equal to that required for processing the entire buffer (see Section IV-B). RawNet2 and AASIST are applied to padded or cropped waveforms of a 4-second duration in the original papers [7], [8]; hence, the buffer for these models is a 4-second-long raw audio. It must be noted that such a buffer length means that there is no difference in the FLOPs for a probability update in the streaming mode and for processing a single utterance in the offline mode. We also included the metrics for our reproduced version of RawNet2.

RawLSTM and RawGRU achieve similar quality in terms of EER and 12.5% relevant improvement from their base model, RawNet2. In addition, we can see that RawLSTM and RawGRU have more than 17 times fewer parameters and FLOPs, which are needed for processing 4-second-long audio. However, the min t-DCF is slightly worse. The metrics for our reproduced RawNet2 are close to the original ones.

In contrast, RS-AASIST and TAASIST require almost twice as many FLOPs for a 4-second-long utterance as AASIST, but they have 70K (23.6%) and 80K (26.9%) fewer parameters, respectively. Performance in terms of both metrics has decreased; however, they are close to the three-run average of 1.13 and 0.0346 for the EER and min t-DCF, respectively, which were reported in the original AASIST paper [8]. This means that this is a matter of a random initialization seed.

The column for the FLOPs for the probability update shows the effectiveness of the RawSpectrogram approach. RawNet2 and AASIST require large buffers and, thus, many computations. The optimization technique for TAASIST, described in Section III-C, leads to a considerable decrease in the computational costs. All our proposed models have significantly faster (more than ten times) output score updates than their base architectures.

An ablation study on the necessity of data augmentation is presented at the bottom of Table 3. It is clear that both RawLSTM and RawGRU have a substantial deterioration in terms of both metrics when they are trained without Pitch Shift and Time Stretch.

Fig. 3 shows the results of the experiment described in Section IV-C. We can see that although our models can start processing utterances with only a chunk of size W, they still require many windows to achieve satisfactory performance. This was expected because a large temporal context is required for robust spoofing detection. Offline models with buffering also suffer from this issue.

The metrics for AASIST, TAASIST, and reproduced RawNet2 in Fig. 3 all fall to their lowest levels at first and then show a clear rise, having large gaps between their best and latest EERs. Thus, we cannot rely on the assumption that offline solutions achieve the same performance when they are applied in streaming. It must be noted that AASIST and the reproduced RawNet2 show much worse performance over time than our TAASIST. The plots for the native streaming models are much more stable. The experiment shows why we need streaming-friendly anti-spoofing systems and cannot rely on offline systems.

Finally, we compare our native-streaming CMs with solutions from the literature, which are discussed in Section II and can be regarded as streaming-friendly. The authors of [19] did not evaluate their model on the LA task; therefore, their models were not considered in the comparison. Table 4 shows the results in terms of the EER and min t-DCF on the LA partition of the ASVspoof 2019 dataset.

[4] https://github.com/eurecom-asp/rawnet2-antispoofing
[5] https://github.com/facebookresearch/fvcore, v.0.1.5

TABLE 2. RawSpectrogram-based models compared to the original architectures. The offline mode in the FLOPs column is shown for a 4-second-long audio. The streaming mode is shown for one probability update.

Model	# Param	FLOPs (Offline Mode)	FLOPs (Streaming Mode)	EER (%)	min t-DCF
RS-AASIST (ours)	227K	17.677G	0.570G	1.12	0.0318
TAASIST (ours)	217K	15.276G	0.493G	1.03	0.0295
AASIST [8]	297K	9.618G	9.618G	0.83	0.0275
RawLSTM (ours)	1456K	0.439G	0.002G	4.95	0.1703
RawGRU (ours)	470K	0.439G	0.002G	4.89	0.1467
RawNet2 [7]	25433K	8.135G	8.135G	5.64	0.1301
RawNet2 (Reproduced)	25433K	8.135G	8.135G	5.54	0.1240

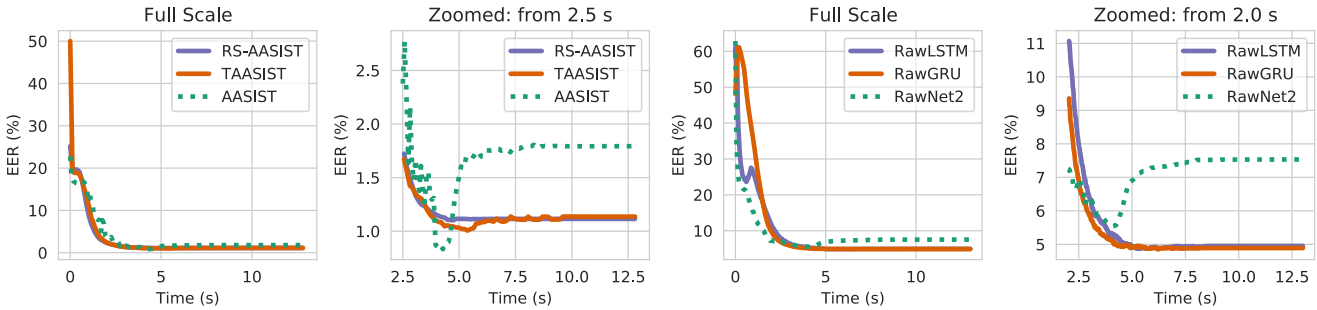

FIGURE 3. EER change with each new chunk from the stream for RS-AASIST, TAASIST, AASIST (the first and the second plots) and RawLSTM, RawGRU (the third and the fourth plots) on the evaluation set of the LA partition of the ASVspoof 2019 dataset. RawNet2 is reproduced.

TABLE 3. Ablation experiments considering the effectiveness of augmentation techniques for RawRNN-type models.

Model	EER (%)	min t-DCF
RawLSTM (ours)	4.95	0.1703
RawGRU (ours)	4.89	0.1467
RawNet2 [7]	5.64	0.1301
RawLSTM (ours) w/o data aug.	8.96	0.2570
RawGRU (ours) w/o data aug.	10.01	0.3029

TABLE 4. RawSpectrogram-based native streaming CMs compared with known streaming-friendly models.

Model	# Param	EER (%)	min t-DCF
RS-AASIST (ours)	227K	1.12	0.0318
RawLSTM (ours)	1456K	4.95	0.1703
RawGRU (ours)	470K	4.89	0.1467
LC-GRNN + LDA [16]	–	6.28	0.1523

LC-GRNN is much worse in terms of EER; however, its min t-DCF is close to that of RawGRU and is better than that of RawLSTM. Thus, our RawSpectrogram-based models are more lightweight or perform better than solutions from the literature.

B. ASV SYSTEM

A comparison of RECAPA-TDNN with different versions of ECAPA-TDNN is presented in Table 5 in terms of the EER and minimum Detection Cost Function (minDCF, [51]) computed according to the Vox1-O benchmark [52] on the VoxCeleb1 dataset [53]. Similarly to the CMs, we show the number of parameters and FLOPs in different modes. The EERs were calculated using Matrix Score Averaging (MFA, [54]) without AS-norm [55] with 3-second-long segments. We cannot assume that the quality of ECAPA-TDNN will be the same with an input of a different length; therefore, for streaming mode, we set the buffer size to 3 seconds and compared the FLOPs in offline mode on a 3-second-long utterance.

We can see that RECAPA-TDNN has 0.54M (3.50%) fewer parameters and processes chunks twice as fast as the original ECAPA-TDNN with 1024 channels. Our model is slightly slower in the offline mode and has worse quality in terms of both metrics. Nonetheless, RECAPA-TDNN significantly outperforms ECAPA-TDNN with 512 channels in terms of EER and minDCF while still being more optimal in the streaming mode. Hence, our ASV solution is sufficiently robust. The smaller version of ECAPA-TDNN surpasses our architecture only in terms of the number of parameters.

We also compared our system with RawNet [20], which can be regarded as native-streaming after optimization of convolutions. The results are presented in Table 6. The authors reported only the EER for their model. We used a publicly available implementation to calculate the number of parameters and FLOPs for offline mode.[6] Although RawNet is slightly smaller and requires fewer FLOPs in the offline mode, its quality dramatically degrades relative to that of RECAPA-TDNN. Optimization of convolutions is a difficult engineering task that we have not done in this paper, as it might not be worth it with such EER.

[6]https://github.com/Jungjee/RawNet

TABLE 5. Our RECAPA-TDNN compared with the 512 and 1024 channels versions of ECAPA-TDNN. The offline mode in the FLOPs column is shown for a 3-second-long audio. The streaming mode is shown for one probability update.

Model	# Param	FLOPs (Offline Mode)	FLOPs (Streaming Mode)	EER (%)	minDCF
RECAPA-TDNN (ours)	14.90M	5.10G	1.70G	1.11	0.0787
ECAPA-TDNN 512 ch., [10], [15]	6.98M	1.81G	1.81G	1.30	0.0909
ECAPA-TDNN 1024 ch., [10], [15]	15.44M	4.25G	4.25G	0.96	0.0717

TABLE 6. Our RECAPA-TDNN compared with RawNet, which could be regarded as streaming-friendly after the application of engineering techniques. The offline mode in the FLOPs column is shown for a 3-second-long audio.

Model	# Param	FLOPs (Offline Mode)	EER (%)	minDCF
RECAPA-TDNN (ours)	14.90M	5.10G	1.11	0.0787
RawNet [21]	12.84M	4.04G	4.0	–

TABLE 7. Our RawSpectrogram-based SASV system compared with the systems from [25]. The best values are in bold. Solutions are denoted in the Method-X-Y form, where X denotes the f function type (L for linear and S for sigmoid) and Y whether to do fine-tuning or not (I – direct inference, F – fine-tune).

System	SV-EER		SPF-EER		SASV-EER	
	Dev	Eval	Dev	Eval	Dev	Eval
RawSpectrogram-L-I	**1.51**	**0.98**	**0.07**	**0.32**	0.78	**0.69**
PR-L-I [26]	2.13	2.14	0.11	0.86	1.21	1.68
RawSpectrogram-S-I	1.58	1.12	**0.07**	0.34	0.81	0.81
PR-S-I [26]	2.43	2.57	**0.07**	0.78	1.34	1.94
RawSpectrogram-L-F	1.55	1.40	**0.07**	0.47	**0.67**	1.09
PR-L-F [26]	2.02	1.92	**0.07**	0.80	1.10	1.54
RawSpectrogram-S-F	1.62	1.47	**0.07**	0.52	**0.67**	1.19
PR-S-F [26]	2.02	1.94	**0.07**	0.80	1.10	1.53

TABLE 8. Ablation experiments considering different choices of layers to tune for a RawSpectrogram-based SASV system. The type of the solution is denoted in the X-Y form, where X denotes the f function type (L for linear and S for sigmoid) and Y whether to do fine-tuning or not (I – direct inference, F – fine-tune).

System Type	Layers to tune	SV-EER		SPF-EER		SASV-EER	
		Dev	Eval	Dev	Eval	Dev	Eval
L-F	Whole Classifier	1.55	1.40	0.07	0.47	0.67	1.09
L-F	Last FC-layer	1.56	1.44	0.07	0.45	0.94	1.12
S-F	Whole Classifier	1.62	1.47	0.07	0.52	0.67	1.19
S-F	Last FC-layer	2.02	1.89	0.07	0.56	1.13	1.51

C. SASV SYSTEM

We compared different setups of probabilistic fusion for our RawSpectrogram-based models with the corresponding combinations of ECAPA-TDNN and AASIST from [25]. The results are presented in Table 2 in terms of SASV-EER, SPF-EER, and SV-EER on the development and evaluation sets of the LA partition of the ASVspoof2019 dataset. The first metric is general and defines the capability of the model to distinguish both non-target attacks and synthesized speech. The second one is similar to the EER from Section V-A but does not consider impostor utterances. The last one checks how the solution discriminates between the target and non-target speech.

The PR-X-Y models from [25] performed the best with the fine-tuning approach. In contrast, the direct inference of our RawSpectrogram SASV system leads to lower EERs, surpassing the corresponding base offline solution by a large margin. Raw-Spectrogram-L-F significantly outperforms PR-L-F in terms of all metrics. Similarly, RawSpectrogram-S-F performs better than PR-S-F.

Whereas the best quality is achieved when the ASV and CM are considered independent, fine-tuned models have the least SASV-EER on the development set and the largest on the evaluation set. Hence, fine-tuning results in overfitting for RawSpectrogram-based models.

The best SASV-EER is 0.69% by RawSpectrogram-L-I, which is a 55% relevant improvement from the top system PR-S-F proposed in [25]. Thus, the results confirm that both RECAPA-TDNN and RS-AASIST can achieve the quality of the original offline models and even surpass them.

For our SASV system, we set the output of GRU as the embedding for our RS-AASIST and not the last hidden activation. Therefore, we fine-tuned the entire classifier in RawSpectrogram-S-F and RawSpectrogram-L-F, as discussed in Section IV-A. However, MLP with several layers can be too much for tuning, which may be the reason for overfitting. We did an ablation study in which we compared two approaches: tuning the entire classifier and tuning only the last FC-layer, as was done in [25]. The results are presented in Table 8. We can see that all metrics have dramatically worsened, except the SPF-EER, which slightly decreased from 0.47% to 0.45% for the linear function and from 0.56% to 0.50% for the sigmoid function. Thus, overfitting is a general problem for the fine-tuning approach for our models.

VI. CONCLUSION

In this paper, we proposed a novel approach for audio processing called RawSpectrogram. It allows the conversion of offline models into streaming-friendly models, preserving the quality of the original architectures. At the same time, our method reduces the number of model parameters and significantly speeds up the predicted probability update in the streaming mode. We proposed novel architectures: RawLSTM, RawGRU, RS-AASIST, TAASIST, and RECAPA-TDNN, which successfully employ the RawSpectrogram approach. However, RS-AASIST, TAASIST, and RECAPA-TDNN still require many FLOPs when they are applied to a finite utterance. Unlike the RawRNN-type models, we did not change AASIST a lot when we used it as EmbNet. A better choice of hyperparameters may enhance the RS-AASIST and TAASIST architectures. Nevertheless, TAASIST is the most successful CM that uses Transformer and is trained via supervised learning. For RECAPA-TDNN, a better investigation into the reasons for overfitting could lead to a faster ASV.

Our SASV system, RawSpectrogram-L-I, significantly outperforms the corresponding offline solutions and is the first native streaming system that appeared in the voice anti-spoofing literature.

CM and ASV tasks apply different architectures; however, we demonstrated that our RawSpectrogram approach works regardless of a task type and is compatible with both raw and time-frequency-transformed inputs. It makes us believe that the proposed EmbNet representation of an utterance well characterizes the audio before passing to RNN. Therefore, with a proper choice of EmbNet architecture, the RawSpectrogram method can be applied in other speech-related tasks.

REFERENCES

[1] Z. Wu, N. Evans, T. Kinnunen, J. Yamagishi, F. Alegre, and H. Li, "Spoofing and countermeasures for speaker verification: A survey," *Speech Commun.*, vol. 66, pp. 130–153, Feb. 2015.

[2] Z. Wu, T. Kinnunen, N. Evans, J. Yamagishi, C. Hanilçi, M. Sahidullah, and A. Sizov, "ASVspoof 2015: The first automatic speaker verification spoofing and countermeasures challenge," in *Proc. Interspeech*, Sep. 2015, pp. 2037–2041, doi: 10.21437/Interspeech.2015-462.

[3] T. Kinnunen, M. Sahidullah, H. Delgado, M. Todisco, N. Evans, J. Yamagishi, and K. A. Lee, "The ASVspoof 2017 challenge: Assessing the limits of replay spoofing attack detection," in *Proc. Interspeech*, Aug. 2017, pp. 2–6.

[4] M. Todisco, X. Wang, V. Vestman, M. Sahidullah, H. Delgado, A. Nautsch, J. Yamagishi, N. Evans, T. H. Kinnunen, and K. A. Lee, "ASVspoof 2019: Future horizons in spoofed and fake audio detection," in *Proc. Interspeech*, Sep. 2019, pp. 1008–1012.

[5] J. Yamagishi, X. Wang, M. Todisco, M. Sahidullah, J. Patino, A. Nautsch, X. Liu, K. A. Lee, T. Kinnunen, N. Evans, and H. Delgado, "ASVspoof 2021: Accelerating progress in spoofed and deepfake speech detection," in *Proc. Ed. Autom. Speaker Verification Spoofing Countermeasures Challenge*, Sep. 2021, pp. 47–54.

[6] J.-W. Jung, H. Tak, H.-J. Shim, H.-S. Heo, B.-J. Lee, S.-W. Chung, H.-J. Yu, N. Evans, and T. Kinnunen, "SASV 2022: The first spoofing-aware speaker verification challenge," in *Proc. Interspeech*, Sep. 2022, pp. 2893–2897.

[7] H. Tak, J. Patino, M. Todisco, A. Nautsch, N. Evans, and A. Larcher, "End-to-end anti-spoofing with RawNet2," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2021, pp. 6369–6373.

[8] J.-W. Jung, H.-S. Heo, H. Tak, H.-J. Shim, J. S. Chung, B.-J. Lee, H.-J. Yu, and N. Evans, "AASIST: Audio anti-spoofing using integrated spectro-temporal graph attention networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2022, pp. 6367–6371.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. U. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2017.

[10] B. Desplanques, J. Thienpondt, and K. Demuynck, "ECAPA-TDNN: Emphasized channel attention, propagation and aggregation in TDNN based speaker verification," in *Proc. Interspeech*, Oct. 2020, pp. 3830–3834.

[11] M. Ravanelli and Y. Bengio, "Speaker recognition from raw waveform with SincNet," in *Proc. IEEE Spoken Lang. Technol. Workshop (SLT)*, Dec. 2018, pp. 1021–1028.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[13] J.-W. Jung, S.-B. Kim, H.-J. Shim, J.-H. Kim, and H.-J. Yu, "Improved RawNet with feature map scaling for text-independent speaker verification using raw waveforms," in *Proc. Interspeech*, Oct. 2020, pp. 1496–1500.

[14] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," 2014, *arXiv:1406.1078*.

[15] R. Kumar Das, R. Tao, and H. Li, "HLT-NUS submission FOR 2020 NIST conversational telephone speech SRE," 2021, *arXiv:2111.06671*.

[16] A. Gomez-Alanis, A. M. Peinado, J. A. Gonzalez, and A. M. Gomez, "A light convolutional GRU-RNN deep feature extractor for ASV spoofing detection," in *Proc. Interspeech*, Sep. 2019, pp. 1068–1072.

[17] X. Wu, R. He, Z. Sun, and T. Tan, "A light CNN for deep face representation with noisy labels," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 11, pp. 2884–2896, Nov. 2018.

[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[19] Z. Chen, W. Zhang, Z. Xie, X. Xu, and D. Chen, "Recurrent neural networks for automatic replay spoofing attack detection," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2018, pp. 2052–2056.

[20] J.-W. Jung, H.-S. Heo, J.-H. Kim, H.-J. Shim, and H.-J. Yu, "RawNet: Advanced end-to-end deep neural network using raw waveforms for text-independent speaker verification," in *Proc. Interspeech*, Sep. 2019, pp. 1268–1272.
[21] X. Wang, X. Qin, Y. Wang, Y. Xu, and M. Li, "The DKU-OPPO system for the 2022 spoofing-aware speaker verification challenge," in *Proc. Interspeech*, Sep. 2022, pp. 4396–4400.
[22] J.-H. Choi, J.-Y. Yang, Y.-R. Jeoung, and J.-H. Chang, "HYU submission for the SASV challenge 2022: Reforming speaker embeddings with spoofing-aware conditioning," in *Proc. Interspeech*, Sep. 2022, pp. 2873–2877.
[23] Z. Teng, Q. Fu, J. White, M. Powell, and D. Schmidt, "SA-SASV: An end-to-end spoof-aggregated spoofing-aware speaker verification system," in *Proc. Interspeech*, Sep. 2022, pp. 4391–4395.
[24] W. Kang, M. J. Alam, and A. Fathan, "End-to-end framework for spoof-aware speaker verification," in *Proc. Interspeech*, Sep. 2022, pp. 4362–4366.
[25] Y. Zhang, G. Zhu, and Z. Duan, "A probabilistic fusion framework for spoofing aware speaker verification," in *Proc. Speaker Lang. Recognit. Workshop*, Jun. 2022, pp. 77–84.
[26] L. Zhang, Y. Li, H. Zhao, Q. Wang, and L. Xie, "Backend ensemble for speaker verification and spoofing countermeasure," in *Proc. Interspeech*, Sep. 2022, pp. 4381–4385.
[27] A. Alenin, N. Torgashov, A. Okhotnikov, R. Makarov, and I. Yakovlev, "A subnetwork approach for spoofing aware speaker verification," in *Proc. Interspeech*, Sep. 2022, pp. 2888–2892.
[28] P. Zhang, P. Hu, and X. Zhang, "Norm-constrained score-level ensemble for spoofing aware speaker verification," in *Proc. Interspeech*, Sep. 2022, pp. 4371–4375.
[29] H. Wu, L. Meng, J. Kang, J. Li, X. Li, X. Wu, H.-Y. Lee, and H. Meng, "Spoofing-aware speaker verification by multi-level fusion," in *Proc. Interspeech*, Sep. 2022, pp. 4357–4361.
[30] T. N. Sainath, B. Kingsbury, A.-R. Mohamed, and B. Ramabhadran, "Learning filter banks within a deep neural network framework," in *Proc. IEEE Workshop Autom. Speech Recognit. Understand.*, Dec. 2013, pp. 297–302.
[31] Y. Wang, P. Getreuer, T. Hughes, R. F. Lyon, and R. A. Saurous, "Trainable frontend for robust and far-field keyword spotting," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2017, pp. 5670–5674.
[32] K. W. Cheuk, H. Anderson, K. Agres, and D. Herremans, "nnAudio: An on-the-fly GPU audio to spectrogram conversion toolbox using 1D convolutional neural networks," *IEEE Access*, vol. 8, pp. 161981–162003, 2020.
[33] Q. Fu, Z. Teng, J. White, M. E. Powell, and D. C. Schmidt, "FastAudio: A learnable audio front-end for spoof speech detection," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2022, pp. 3693–3697.
[34] S. Schneider, A. Baevski, R. Collobert, and M. Auli, "wav2vec: Unsupervised pre-training for speech recognition," in *Proc. Interspeech*, Sep. 2019, pp. 3465–3469.
[35] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 12449–12460.
[36] R. Xiong, Y. Yang, D. He, K. Zheng, S. Zheng, C. Xing, H. Zhang, Y. Lan, L. Wang, and T. Liu, "On layer normalization in the transformer architecture," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 10524–10533.
[37] D. Hendrycks and K. Gimpel, "Gaussian error linear units (GELUs)," 2016, *arXiv:1606.08415*.
[38] H. Tak, M. Todisco, X. Wang, J.-W. Jung, J. Yamagishi, and N. Evans, "Automatic speaker verification spoofing and deepfake detection using wav2vec 2.0 and data augmentation," 2022, *arXiv:2202.12233*.
[39] L. N. Smith and N. Topin, "Super-convergence: Very fast training of neural networks using large learning rates," *Proc. SPIE*, vol. 11006, pp. 369–386, May 2019.
[40] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[41] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2017, *arXiv:1711.05101*.
[42] X. Wang et al., "ASVspoof 2019: A large-scale public database of synthesized, converted and replayed speech," *Comput. Speech Lang.*, vol. 64, Nov. 2020, Art. no. 101114.
[43] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "ArcFace: Additive angular margin loss for deep face recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4685–4694.
[44] X. Xiang, S. Wang, H. Huang, Y. Qian, and K. Yu, "Margin matters: Towards more discriminative deep neural network embeddings for speaker recognition," in *Proc. Asia–Pacific Signal Inf. Process. Assoc. Annu. Summit Conf. (APSIPA ASC)*, Nov. 2019, pp. 1652–1656.
[45] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.
[46] S. Wang, Y. Yang, T. Wang, Y. Qian, and K. Yu, "Knowledge distillation for small foot-print deep speaker embedding," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 6021–6025.
[47] J. S. Chung, A. Nagrani, and A. Zisserman, "VoxCeleb2: Deep speaker recognition," in *Proc. Interspeech*, Sep. 2018, pp. 1086–1090.
[48] Y. Kwon, H.-S. Heo, B.-J. Lee, and J. S. Chung, "The ins and outs of speaker recognition: Lessons from VoxSRC 2020," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2021, pp. 5809–5813.
[49] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "SpecAugment: A simple data augmentation method for automatic speech recognition," in *Proc. Interspeech*, Sep. 2019, pp. 2613–2617.
[50] T. Kinnunen, K. A. Lee, H. Delgado, N. Evans, M. Todisco, M. Sahidullah, J. Yamagishi, and D. A. Reynolds, "T-DCF: A detection cost function for the tandem assessment of spoofing countermeasures and automatic speaker verification," in *Proc. Speaker Lang. Recognit. Workshop*, Jun. 2018, pp. 312–319.
[51] A. Brown, J. Huh, J. Son Chung, A. Nagrani, D. Garcia-Romero, and A. Zisserman, "VoxSRC 2021: The third VoxCeleb speaker recognition challenge," 2022, *arXiv:2201.04583*.
[52] A. Nagrani, J. S. Chung, W. Xie, and A. Zisserman, "Voxceleb: Large-scale speaker verification in the wild," *Comput. Speech Lang.*, vol. 60, Mar. 2020, Art. no. 101027.
[53] A. Nagrani, J. S. Chung, and A. Zisserman, "VoxCeleb: A large-scale speaker identification dataset," in *Proc. Interspeech*, Aug. 2017, pp. 2616–2620.
[54] L. Zhang, H. Zhao, Q. Meng, Y. Chen, M. Liu, and L. Xie, "Beijing ZKJ-NPU speaker verification system for VoxCeleb speaker recognition challenge 2021," 2021, *arXiv:2109.03568*.
[55] P. Matějka, O. Novotný, O. Plchot, L. Burget, M. D. Sánchez, and J. Černocký, "Analysis of score normalization in multilingual speaker recognition," in *Proc. Interspeech*, Aug. 2017, pp. 1567–1571.

PETR GRINBERG received the B.S. degree in computer science from the National Research University Higher School of Economics, Moscow, Russia, in 2023. He is currently pursuing the M.S. degree in data science with École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland.

Since 2020, he has been a Research Assistant with the Samsung R&D Institute Russia. He participated in SASV Challenge 2022. His research interests include audio processing, voice anti-spoofing, and music generation.

VLADISLAV SHIKHOV received the Specialist degree in computer science from Vyatka State Pedagogical University, Russia, in 2002. He is currently a Research Engineer with the Recognition Technology Laboratory, Samsung R&D Institute Russia. His research interests include artificial intelligence, machine learning, and computer vision/signal processing algorithms in biometrics.