Automatic speaker verification spoofing and deepfake detection using wav2vec 2.0 and data augmentation

Hemlata Tak\textsuperscript{1}, Massimiliano Todisco\textsuperscript{1}, Xin Wang\textsuperscript{2}, Jee-weon Jung\textsuperscript{3}
Junichi Yamagishi\textsuperscript{2} and Nicholas Evans\textsuperscript{1}

\textsuperscript{1}EURECOM, France, \textsuperscript{2}National Institute of Informatics, Japan
\textsuperscript{3}Naver Corporation, South Korea
\{tak, todisco, evans\}@eurecom.fr, \{wangxin, jyamagishi\}@nii.ac.jp, jeeweon.jung@navercorp.com

Abstract

The performance of spoofing countermeasure systems depends fundamentally upon the use of sufficiently representative training data. With this usually being limited, current solutions typically lack generalisation to attacks encountered in the wild. Strategies to improve reliability in the face of uncontrolled, unpredictable attacks are hence needed. We report in this paper our efforts to use self-supervised learning in the form of a wav2vec 2.0 front-end with fine tuning. Despite initial base representations being learned using only bona fide data and no spoofed data, we obtain the lowest equal error rates reported in the literature for both the ASVspoof 2021 Logical Access and Deepfake databases. When combined with data augmentation, these results correspond to an improvement of almost 90% relative to our baseline system.

1. Introduction

A persistent challenge in the design of spoofing countermeasures (CMs) for automatic speaker verification (ASV) is reliability in the face of diverse, unpredictable attacks [1]. ASV systems can be compromised by attacks belonging to a broad variety of different classes, e.g., converted voice, synthetic speech and replayed recordings. Even within each attack class, there is considerable potential variation, e.g., different algorithms or recording and replay device characteristics. An ideal spoofing detection solution should be robust to all such variation even if, in the wild, it is unpredictable. The acquisition of training data that is representative of spoofing attacks with near-boundless variability is obviously impracticable.

The ASVspoof initiative and challenge series have collected large databases of spoofed and bona fide utterances that are suitable for the training of spoofing countermeasures. To promote the development of generalisable countermeasures, namely detection solutions that cope well in the face of spoofing attacks not previously encountered, assessment is performed with experimental protocols and evaluation data that comprise spoofed utterances generated with a broad variety of different algorithms or techniques. The differences between training, development and evaluation data can lead to substantial differences in detection performance. For the most recent ASVspoof 2021 Logical Access (LA) evaluation [2], the equal error rate (EER) of the best performing baseline solution increased from 0.55% for the development set to 9.26% for the evaluation set [2]. Submission results show better performance [3–9], but the fundamental gap between performance for development and evaluation data remains, indicating a persisting lack of generalisation.

Given that the training data used for ASVspoof challenges comprises spoofed utterances generated with a modest number of different attack algorithms (six in the case of the ASVspoof 2019 LA database), the lack of generalisation may be difficult to improve upon unless challenge rules are relaxed to allow training using external data. A relaxed training data policy would complicate comparisons between different systems and technology trained using different data – the playing field would no longer be level – though potential improvements to generalisation may make it worthwhile.

The question then is what external training data to use and how to use it? With the drive toward reproducible research, a number of speech synthesis and voice conversion algorithms are now openly available as open source. Additional training data, generated with different attack algorithms, can hence be produced readily. The number of algorithms remains limited, however, and can fundamentally never be fully representative of what can reasonably be expected in the wild. We have hence explored a different approach.

Motivated by (i) its proven application to the learning of general neural representations for a range of different tasks [11–19], (ii) evidence that fine-tuning with modest quantities of labelled data leads to state-of-the-art results, (iii) encouraging, previously reported results for anti-spoofing [20,21] and (iv) the appeal of one-class classification approaches [22], we have explored the use of self-supervised learning to improve generalisation. Our hypothesis is that better representations trained on diverse speech data, even those learned for other tasks and initially using only bona fide data (hence one-class), may help to reduce over-fitting and hence improve reliability and domain-robustness, particularly in the face of previously unseen spoofing attacks. Additionally, we hope that better trained representations are complementary to data augmentation techniques which are already known to improve generalisation [7,23–27].

The principal contributions of this work are: (i) improved generalisation and domain robustness using a pre-trained, self-supervised speech model with fine-tuning; (ii) additional improvements using data augmentation showing complementary benefits to self-supervised learning; (iii) a new self-attention based aggregation layer which brings complementary improvements.

The first author is supported by the VoicePersonae project funded by the French Agence Nationale de la Recherche (ANR) and the Japan Science and Technology Agency (JST).
2. Related work

Self-supervised learning (SSL) has attracted considerable attention in recent times. Existing works show that pre-trained models derived using SSL generalise well across a multitude of different tasks when a relatively simple classifier is learned from the resulting representations using only a modest quantity of labelled data [28]. A growing number of self-supervised speech models have been proposed. Examples include contrastive predictive coding (CPC) [16, 29], auto-regressive predictive coding [30], wav2vec [31], HuBERT [32, 33], wav2vec 2.0 [12, 34] and Wavlm [35], with all showing promising results for a variety of different speech processing tasks.

Two particularly popular approaches, HuBERT and wav2vec 2.0, have been applied to automatic speech recognition [12, 33], mispronunciation detection [36, 37], speaker recognition [38, 39] and emotion recognition [40]. The same techniques have been explored in the context of spoofing detection [20, 21]. Xie et al. [20] showed the benefit of using SSL with a Siamese network for spoofing detection. With learned representations coming from their combination, and without comparative assessments using representations derived via alternative means, the specific benefits of SSL are difficult to judge.

Wang et al. [21] compared different SSL based front-ends and back-end architectures and showed the importance of fine-tuning SSL models for spoofing detection. By replacing a linear frequency cepstral coefficient front-end with a wav2vec 2.0 front-end and by fine-tuning, they achieved relative reductions in the equal error rate (EER) of 68% and 79% for the ASVspoof 2021 LA and DF databases respectively. Nonetheless, the EERs remain at 5% and the additional or complementary benefit of data augmentation, which is known to be beneficial in both cases [3, 4, 7], was not explored. Results showed that the wav2vec 2.0 [12] front-end gives better generalised spoofing detection performance than a HuBERT model.

We have hence explored the wav2vec 2.0 XLS-R (0.3B) model\(^1\) [13] in our work. wav2vec 2.0 XLS-R is a large-scale cross-lingually pre-trained model trained on diverse corpora including VoxPopuli data (VP-400K) [41], the multilingual Librispeech corpus (MLS) [42], CommonVoice (CV) [43], VoxLingua107 (VL) [44], and BABEL (BBL) [45] datasets. Together, they include speech data in 128 different languages from many different regions of the world. We explored the wav2vec 2.0 front-end with an integrated spectro-temporal graph attention network (AASIST) as a back-end [10]. The latter is described in the next section. Its coupling with a wav2vec 2.0 front-end is described in Section 4.

3. AASIST baseline system

The baseline system is an end-to-end, integrated spectro-temporal graph attention network named AASIST [10], illustrated in Fig. 1. It extracts representations directly from raw waveform inputs. As illustrated in Fig. 2-(a), AASIST uses a sinc convolutional layer based front-end [46]. It is initialised with 70 mel-scaled filters, each with a kernel size of 129 [47]. Through the addition of a channel dimension, the sinc layer output is fed to a post-processing layer and transformed to a spectro-temporal representation. These are fed to a RawNet2-based residual encoder, to learn a higher-level feature map \( \mathbf{S} \in \mathbb{R}^{C \times F \times T} \), where \( C, F \) and \( T \) refer to the number of channels, spectral bins and time frames respectively.

Separate spectral and temporal representations are learned from \( \mathbf{S} \) using a max-pooling operation which is applied to the absolute values across either temporal or spectral dimensions in order to construct either a spectral input graph \( \mathcal{G}_s \in \mathbb{R}^{N_s \times d_s} \) or a temporal input graph \( \mathcal{G}_t \in \mathbb{R}^{N_t \times d_t} \). \( N_s \) and \( N_t \) are the set of graph nodes in spectral and temporal graphs respectively whereas \( d \) is the feature dimensionality of each node. Spec-

\(^1\)https://github.com/pytorch/fairseq/tree/main/examples/wav2vec/xlsr
4. Self-supervised front-end

In this section we describe the replacement of the sinc-layer front-end with a wav2vec 2.0 model as illustrated in Fig. 2-(b). We describe both pre-training and fine-tuning to support downstream spoofing detection, both illustrated in Fig. 3.

4.1. Wav2vec 2.0 model

The wav2vec 2.0 pre-trained model is used to extract a sequence of feature representations \( o_{1:N} \) from the raw input waveform \( x_{1:L} \), where \( L \) is the number of samples. As shown in Fig. 3, the wav2vec 2.0 model consists of a convolutional neural network (CNN) and a transformer [28, 51] network. The former converts the input \( x_{1:L} \) to a hidden feature sequence \( z_{1:N} \) whereas the latter transforms \( z_{1:N} \) to output sequence \( o_{1:N} \). The ratio between \( L \) and \( N \) is dictated by the CNN stride of 20 ms (the default setting).

4.2. Pre-training

An illustration of the pre-training procedure following [12] is illustrated to the left in Fig. 3. Latent representations \( z_{1:N} \) are quantised to representations \( q_{1:N} \). Some portion of the latent representation \( z_{1:N} \) is then masked and fed to the transformer which builds new context representations \( o_{1:N} \). A contrastive loss for each masked time step \( n \) is then computed to measure how well the target \( q_n \) can be identified from among a set of distractors (i.e., \( q_n \) sampled from the other masked time steps where \( n' \neq n \)) given the corresponding context vector \( c_n \). All work reported in this paper was performed with the wav2vec 2.0 XLS-R (0.3B) model [13]. We followed the example in the Fairseq project toolkit [52] to extract feature representations from self-supervised wav2vec 2.0 pre-trained model.\(^2\)

4.3. Fine-tuning

Since pre-training is performed with only bona fide data (with no spoofed data), as per [21], spoofing detection performance is expected to improve with fine tuning using in-domain bona fide and spoofed training data. Our hypothesis is that fine tun-
ing will protect against over-fitting and hence promote better generalisation. For all experiments reported in this paper, including those related to the ASVspoof 2021 LA dataset and the ASVspoof 2021 DF dataset, fine-tuning is performed using the ASVspoof 2019 LA training partition only. Whereas the 2021 LA data contains codec and transmission variation and the 2021 DF data contains compression variation, the 2019 LA data used for fine-tuning contains neither. During fine-tuning, the pre-trained wav2vec 2.0 XLS-R model is optimised jointly with the AASIST CM via back-propagation using the ASVspoof 2019 LA training set. This process is described in section 6.3 whereas the fine-tuning procedure is illustrated to the right in Fig. 3. It is performed using a weighted cross entropy objective function to minimize the training loss. In contrast to pre-training, input masking is not applied to hidden features \( z_{1:N} \) during fine-tuning. Additionally, we add a fully connected layer on top of the wav2vec 2.0 transformer encoder output \( o_{1:N} \) in order to reduce the representation dimension (top-right of Fig. 3).

### 4.4. Use with AASIST CM

The sinc layer shown in Fig. 2-(a) is replaced with the wav2vec 2.0 front-end shown in Fig. 2-(b). As before, the output \( o_{1:N} \) is fed to a RawNet2-based residual encoder which is used to learn higher-level feature representations \( S \in \mathbb{R}^{C \times F \times T} \). Whereas the baseline system extracts temporal and spectral representations \( t \) and \( f \) from \( S \) using a max-pooling operation, a self-attentive aggregation layer (described in Section 5) was found to improve performance of both front-ends. As shown in Fig. 1, temporal and spectral representations are then fed to the AASIST CM model to obtain a two-class prediction (bona fide and spoofed) in the same manner as described in Section 3. A summary of the wav2vec 2.0 front-end and downstream AASIST model configurations is presented in Table 1.

### 5. Self-attention based aggregation layer

Attention based pooling layers such as self-attentive pooling (SAP) and attentive statistical pooling (ASP) [53] has shown to be beneficial to the aggregation of frame-level features and the extraction of embeddings [54–58] for speaker recognition and verification tasks. We have also found that the introduction of a 2D self-attention based aggregation layer between the front-end and back-end helps to improve spoofing detection performance.

The new self-attentive aggregation layer is used to extract more attentive/relevant spectral and temporal representations. It helps to aggregate and assign higher attention weights through weighted summation to the most discriminative temporal and spectral features. We generate 2-D attention maps (an attention weight matrix) using a 2-D convolutional (conv2d) layer with one kernel-size rather than conventional conv1d based attention applied to a single domain. Weights are derived from representations \( S \) processed by a conv2d layer followed by an activation & batch normalization (BN) layer, a 2-D convolutional layer and a softmax function to normalize the weights:

\[
W = \text{Softmax}(\text{conv2d}(\text{BN}(\text{SeLU}(\text{conv2d}(S))))),
\]

where \( \text{conv2d}() \) denotes the 2-D convolution operation with an scaled exponential linear unit SeLU() as the activation function [59], and BN is batch normalisation [60]. Temporal and spectral representations are then extracted from the self-attentive aggregation layer according to:

\[
t = \sum_{F} S \odot W,
\]

where \( \odot \) denotes element-wise multiplication. \( W \in \mathbb{R}^{C \times F \times T} \) is the 2-D attention normalised learnable weight matrix used in the self-attentive aggregation layer to calculate the weighted sum of the representation \( S \) across time and frequency.

### 6. Experimental setup

Described in the following are the databases and metrics used in all reported experimental work, our use of data augmentation and specific, reproducible implementation details.

#### 6.1. Databases and metrics

We used the training and development partitions of the ASVspoof 2019 LA database [61, 62] for training and validation. Evaluation was performed using the ASVspoof 2021 LA and domain mis-matched DF databases [2]. While both are generated from the same VCTK source database\(^3\), the LA database

<table>
<thead>
<tr>
<th>Layer</th>
<th>Input:64600 samples</th>
<th>Output shape</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data-aug</td>
<td>RawBoost</td>
<td>(64600)</td>
</tr>
<tr>
<td>SSL</td>
<td>wav2vec 2.0</td>
<td>(201,1024) (T,F)</td>
</tr>
<tr>
<td>front-end</td>
<td>FC (fine-tuning)</td>
<td>(201,128)</td>
</tr>
<tr>
<td>transpose</td>
<td>o=(128,201) (F,T)</td>
<td></td>
</tr>
<tr>
<td>post-processing</td>
<td>add channel</td>
<td>(1,128,201)</td>
</tr>
<tr>
<td>HS-GAL</td>
<td>Maxpool-2D(3)</td>
<td>(1,42,67)</td>
</tr>
<tr>
<td>BN &amp; SeLU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Res-block 2×</td>
<td>Conv-2D(2,3),1,32</td>
<td>(32,42,67)</td>
</tr>
<tr>
<td>BN &amp; SeLU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Res-block 4×</td>
<td>Conv-2D(2,3),1,64</td>
<td>(64,42,67)</td>
</tr>
<tr>
<td>BN &amp; SeLU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Res-block 4×</td>
<td>Conv-2D(2,3),1,64</td>
<td>(64,42,67)</td>
</tr>
<tr>
<td>BN &amp; SeLU</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 1: The wav2vec 2.0 and AASIST model architecture and configuration. Dimensions refer to (channels, frequency, time). Batch normalisation (BN) and scaled exponential linear unit (SeLU), beneath the dotted line, are applied to residual block outputs.**

The sinc layer shown in Fig. 2-(a) is replaced with the wav2vec 2.0 front-end shown in Fig. 2-(b). As before, the output \( o_{1:N} \) is fed to a RawNet2-based residual encoder which is used to learn higher-level feature representations \( S \in \mathbb{R}^{C \times F \times T} \). Whereas the baseline system extracts temporal and spectral representations \( t \) and \( f \) from \( S \) using a max-pooling operation, a self-attentive aggregation layer (described in Section 5) was found to improve performance of both front-ends. As shown in Fig. 1, temporal and spectral representations are then fed to the AASIST CM model to obtain a two-class prediction (bona fide and spoofed) in the same manner as described in Section 3. A summary of the wav2vec 2.0 front-end and downstream AASIST model configurations is presented in Table 1.

**5. Self-attention based aggregation layer**

Attention based pooling layers such as self-attentive pooling (SAP) and attentive statistical pooling (ASP) [53] has shown to be beneficial to the aggregation of frame-level features and the extraction of embeddings [54–58] for speaker recognition and verification tasks. We have also found that the introduction of a 2D self-attention based aggregation layer between the front-end and back-end helps to improve spoofing detection performance.

The new self-attentive aggregation layer is used to extract more attentive/relevant spectral and temporal representations. It helps to aggregate and assign higher attention weights through weighted summation to the most discriminative temporal and spectral features. We generate 2-D attention maps (an attention weight matrix) using a 2-D convolutional (conv2d) layer with one kernel-size rather than conventional conv1d based attention applied to a single domain. Weights are derived from representations \( S \) processed by a conv2d layer followed by an activation & batch normalization (BN) layer, a 2-D convolutional layer and a softmax function to normalize the weights:

\[
W = \text{Softmax}(\text{conv2d}(\text{BN}(\text{SeLU}(\text{conv2d}(S))))),
\]

where \( \text{conv2d}() \) denotes the 2-D convolution operation with an scaled exponential linear unit SeLU() as the activation function [59], and BN is batch normalisation [60]. Temporal and spectral representations are then extracted from the self-attentive aggregation layer according to:

\[
t = \sum_{F} S \odot W,
\]

where \( \odot \) denotes element-wise multiplication. \( W \in \mathbb{R}^{C \times F \times T} \) is the 2-D attention normalised learnable weight matrix used in the self-attentive aggregation layer to calculate the weighted sum of the representation \( S \) across time and frequency.

**6. Experimental setup**

Described in the following are the databases and metrics used in all reported experimental work, our use of data augmentation and specific, reproducible implementation details.

**6.1. Databases and metrics**

We used the training and development partitions of the ASVspoof 2019 LA database [61, 62] for training and validation. Evaluation was performed using the ASVspoof 2021 LA and domain mis-matched DF databases [2]. While both are generated from the same VCTK source database\(^3\), the LA database

\[^3\]http://dx.doi.org/10.7488/ds/1994
contains codec and transmission variability whereas the DF database contains compression variability in addition to data stemming from sources other than the VCTK database [63]. The 2019 data used for training and validation contains neither. We use two evaluation metrics: the Equal Error Rate (EER) [64] and the Minimum Tandem Detection Cost Function (min t-DCF) [65]. We focus on the first for ease of interpretation and include the second since it is the default metric for ASVspoof challenges.

6.2. Data augmentation

Data augmentation (DA) is already known to reduce overfitting and hence to improve generalisation [3, 4, 6, 7, 27] and is particularly effective in the case of LA scenarios in which there is substantial variability stemming from, e.g., encoding, transmission and acquisition devices [2]. We are interested to determine whether self-supervised learning is complementary to DA. Unlike traditional DA techniques which enlarge the training dataset using additional, artificially generated utterances, and using the RawBoost DA tool [27], we add nuisance variability on-the-fly to the existing training data. RawBoost adds variation in the form of: i) linear and non-linear convolutive noise; ii) impulsive signal-dependent additive noise; iii) stationary signal-independent additive noise. Full details are available in [27].

DA is applied using exactly the same configuration and parameters reported in the original work [27]. It shows that a combination of linear and non-linear convolutive noise and impulsive signal-dependent additive noise strategies work best for the LA database. These augmentation strategies suit the convolutive and device related noise sources that characterise telephony applications. In contrast, for the DF database, DA works best using stationary signal-independent additive, randomly coloured noise, which match better the effects of compression [66] applied in generating the DF database. DA experiments and configuration are discussed further in Section 7.3.

6.3. Implementation details

Audio data are cropped or concatenated giving segments of approximately 4 seconds duration (64,600 samples). Graph pooling is applied with an empirically chosen pooling ratio of $k = 0.5$ for spectral and temporal graphs. We used the standard Adam optimiser [67] with a fixed learning rate of 0.0001 for experiments without the wav2vec 2.0 front-end. Since fine-tuning demands high GPU computation, experiments with wav2vec 2.0 were performed with a smaller batch size of 14 and a lower learning rate of $10^{-6}$ to avoid model over-fitting. As illustrated to right in Fig. 3, the fully connected layer after the wav2vec 2.0 SSL front end used 128 output dimensions. All other hyperparameters are the same for both front-ends which are both jointly optimised with the back-end classifier using back-propagation [68]. As is now common in the related literature [69,70], we performed each experiments with three runs using different random seeds to initialize the network weights and report the results of the best performing seed and average results. All models were trained for 100 epochs on a single GeForce RTX 3090 GPU and all results are reproducible using open source code⁴ and with the same random seed and GPU environment.


<table>
<thead>
<tr>
<th>front-end</th>
<th>SA</th>
<th>DA</th>
<th>Pooled EER</th>
<th>Pooled min t-DCF</th>
</tr>
</thead>
<tbody>
<tr>
<td>sinc-layer</td>
<td>×</td>
<td>×</td>
<td>11.47 (11.95)</td>
<td>0.5081 (0.5139)</td>
</tr>
<tr>
<td>wav2vec 2.0</td>
<td>×</td>
<td>×</td>
<td>6.15 (6.46)</td>
<td>0.3577 (0.3587)</td>
</tr>
<tr>
<td>sinc-layer</td>
<td>✓</td>
<td>×</td>
<td>8.73 (11.61)</td>
<td>0.4285 (0.5203)</td>
</tr>
<tr>
<td>wav2vec 2.0</td>
<td>✓</td>
<td>×</td>
<td>4.48 (6.15)</td>
<td>0.3094 (0.3482)</td>
</tr>
<tr>
<td>sinc-layer</td>
<td>✓</td>
<td>✓</td>
<td>7.65 (7.87)</td>
<td>0.3894 (0.3960)</td>
</tr>
<tr>
<td>wav2vec 2.0</td>
<td>✓</td>
<td>✓</td>
<td>0.82 (1.00)</td>
<td>0.2066 (0.2120)</td>
</tr>
</tbody>
</table>

7. Results

We present five sets of experiments. The first is a comparison of each front-end in terms of performance for the ASVspoof 2021 LA database. The second and third assess the complementary benefits coming from the self-attention based aggregation layer and data augmentation. The fourth is an assessment performed on the ASVspoof 2021 DF database whereas the last is an assessment using a simplified CM solution.

7.1. Front-end comparison

Results for the AASIST baseline with the sinc-layer front-end (Section 3) and the same system with the wav2vec 2.0 front-end (Section 4) are presented in the first two rows of Table 2. These systems use neither the self-attention aggregation layer nor data augmentation. The baseline EER of 11.47% is high and shows that the system is not robust to the codec and transmission variability which characterises the ASVspoof 2021 LA dataset. The same system using the wav2vec 2.0 front-end delivers an EER of 6.15%. While the relative reduction is almost 46%, the EER is still unacceptably high.

7.2. Self-attention aggregation layer

Results for the same two front-end variants but using the self-attention aggregation layer (SA) introduced in Section 5 are presented in rows 3 and 4 of Table 2. In both cases the EER drops substantially; to 8.73% for the sinc-layer front-end and to 4.48% for the wav2vec 2.0 front-end. In this case the wav2vec 2.0 front-end is responsible for a relative improvement of almost 50%.

7.3. Data augmentation

Results for the same two systems, both with the self-attentive aggregation layer (SA), and now also with data augmentation (DA), are shown in rows 5 and 6 of Table 2. DA reduces the EER only marginally from 8.73% to 7.65% in case of the sinc-layer front-end. Since the EER is still unacceptably high, the system is not robust to the codec and transmission variability which characterises the ASVspoof 2021 LA dataset. The second and third assess the complementary benefits coming from the new self-attention based aggregation layer and data augmentation. The fourth is an assessment performed on the ASVspoof 2021 DF database whereas the last is an assessment using a simplified CM solution.
Table 3: As for Table 2 except for the ASVspoof DF database, evaluation set. Since there is no ASV in the DF scenario, there are no min t-DCF results. The last two lines show results for an LA-optimised DA configuration.

<table>
<thead>
<tr>
<th>front-end</th>
<th>SA</th>
<th>DA</th>
<th>Pooled EER</th>
</tr>
</thead>
<tbody>
<tr>
<td>sinc-layer</td>
<td>×</td>
<td>×</td>
<td>21.06 (22.11)</td>
</tr>
<tr>
<td>wav2vec 2.0</td>
<td>×</td>
<td>×</td>
<td>7.69 (9.48)</td>
</tr>
<tr>
<td>sinc-layer</td>
<td>✓</td>
<td>×</td>
<td>23.22 (25.08)</td>
</tr>
<tr>
<td>wav2vec 2.0</td>
<td>✓</td>
<td>×</td>
<td>4.57 (7.70)</td>
</tr>
<tr>
<td>wav2vec 2.0</td>
<td>✓</td>
<td>✓</td>
<td>24.42 (25.38)</td>
</tr>
<tr>
<td>sinc-layer</td>
<td>✓</td>
<td>✓</td>
<td>2.85 (3.69)</td>
</tr>
<tr>
<td>wav2vec 2.0</td>
<td>✓</td>
<td>✓</td>
<td>20.04 (20.50)</td>
</tr>
<tr>
<td>wav2vec 2.0</td>
<td>✓</td>
<td>✓</td>
<td>6.64 (7.32)</td>
</tr>
</tbody>
</table>

The last set of experiments were performed in order to gauge the relative importance of the AASIST and whether the improvements in generalisation are obtained for a simpler CM solution.

8. Conclusions and discussion

We report in this paper our attempts to harness the power of self-supervised learning in the form of the popular wav2vec 2.0 front-end to learn more reliable representations to improve spoofing detection performance. We show that a well-trained, fine-tuned front-end, even when learned initially using massive quantities of only bona fide utterances, can improve generalisation substantially. Compared to a sinc-layer front-end, when coupled with a new self-attentive aggregation layer and data augmentation, the wav2vec 2.0 front-end delivers up to a 90% relative reduction in the equal error rate for the logical access spoofing detection task in which spoofed utterances are generated with more than 100 different attack algorithms. Improvements stemming from the use of the self-supervised wav2vec 2.0 front-end are consistent for every experimental comparison and, to the best of the authors’ knowledge, results are the lowest reported EERs for both LA and DF databases to date.

We must nonetheless acknowledge that almost all results reported in the literature are derived using fixed training data, whereas those reported in this paper are derived from a model pre-trained using additional external data. Given that our results are obtained with different systems trained using different data, comparisons to the majority of results reported in the literature are obviously not fair. The scale of the improvements, however, up to a 90% relative reduction in EER, indicate the potential gain in performance that can be obtained with the use of additional, external training data and might suggest that the training data restrictions for ASVspoof evaluations might be relaxed. Nonetheless, the wav2vec 2.0 model is massively more complex when compared to the previous state-of-the-art solutions. Whether or not solutions with such footprints can be adapted to support practical applications remains to seen.

Future work should investigate which particular characteristics of the self-supervised front-end are most beneficial. One objective of such work is to use the results of such analysis as a starting point to scale down the model size and complexity so that it might be more easily adopted for practical scenarios with modest computational resources.

We used their respective optimised DA strategies for LA and DF as described in Section 6.2.

Table 4: Pooled EER and pooled min t-DCF (LA only) results for the ASVspoof 2021 LA and DF databases (DBs), evaluation sets, using DB-optimised DA and the simplified back-end.

<table>
<thead>
<tr>
<th>front-end</th>
<th>DA</th>
<th>DB</th>
<th>Pooled EER</th>
<th>Pooled min t-DCF</th>
</tr>
</thead>
<tbody>
<tr>
<td>wav2vec 2.0</td>
<td>✓</td>
<td>LA</td>
<td>1.19</td>
<td>0.2175</td>
</tr>
<tr>
<td>wav2vec 2.0</td>
<td>✓</td>
<td>DF</td>
<td>4.38</td>
<td>-</td>
</tr>
</tbody>
</table>

EER decreases from 4.48% to 0.82%. This result is also statistically significant. This result corresponds to a relative improvement of almost 90% when compared to the baseline EER of 7.65%. To the best of our knowledge, this is the lowest EER reported for the ASVspoof 2021 LA database.

4. DeepFake results

Results for exactly the same experiments, but for the ASVspoof 2021 DeepFake (DF) database, are shown in Table 3. While neither SA, nor DA improve upon the baseline EER of 21.06%, consistency improvements are obtained for the wav2vec 2.0 front-end for which the EER drops from 7.69% to 2.85% using both SA and DA. This result is also statistically significant. To the best of our knowledge, this is the lowest EER reported for the ASVspoof 2021 DF database.

These results, while determined with the same wav2vec 2.0 front-end used for LA experiments, relate to a DA strategy optimised for the DF database (stationary signal-independent additive randomly coloured noise – see Section 6.2). Results for exactly the same setup, using the DA strategy optimised for LA (linear and non-linear convolutive noise and impulsive signal-dependent additive noise) are shown in the last two rows of Table 3. While the EER increases to 6.64%, this is still a competitive result and is 67% lower relative to the result of 20.04% for the sinc-layer front-end. Whereas a component of the DF database originates from the same VCTK database as the entire LA database, other components are sourced from multiple different corpora (Voice Conversion Challenge 2018 and 2020 databases) [2] including spoofed utterances generated with more than 100 different algorithms. With the ASVspoof 2019 LA training data containing neither codec or transmission variability (LA evaluation data), nor compression variability (DF evaluation data), results show that the use of better pre-trained models leads to consistent improvements in generalisation, here being previously unseen spoofing attacks. Results for the DF database show that the benefit extends also to the case of domain mismatch.

5. Simplified CM solution

The last set of experiments were performed in order to gauge the relative importance of the AASIST and whether the improvements in generalisation are obtained for a simpler CM solution. We removed the RawNet2-based encoder and replaced AASIST with a simple back-end comprising a max-pooling layer, a single graph module layer and a linear layer. Results for both ASVspoof 2021 LA and DF databases using optimised DA strategies for each are shown in Table 4. LA and DF results of 1.19% and 4.38% show that competitive EERs can be obtained using the fine-tuned wav2vec 2.0 front-end even with relatively less complex networks and that the benefits to generalisation are still complementary to those of DA.
9. References


