On the Application of Conformers to Logical Access Voice Spoofing Attack Detection

Eros Rosello$^1$, Alejandro Gomez-Alanis$^1$, Manuel Chica$^1$, Angel M. Gomez$^1$, José A. Gonzalez-Lopez$^1$, Antonio M. Peinado$^1$

$^1$Department of Signal Processing, Telematics and Communications, University of Granada, Spain

erosrosello@ugr.es, agomezalanis@ugr.es, manuelc@ugr.es, amgg@ugr.es, joseangl@ugr.es, amp@ugr.es

Abstract

Biometric systems are exposed to spoofing attacks which may compromise their security, and automatic speaker verification (ASV) is no exception. To increase the robustness against such attacks, anti-spoofing systems have been proposed for the detection of spoofed audio attacks. However, most of these systems cannot capture long-term feature dependencies and can only extract local features. While transformers are an excellent solution for the exploitation of these long-distance correlations, they may degrade local details. On the contrary, convolutional neural networks (CNNs) are a powerful tool for extracting local features but not so much for capturing global representations. The conformer is a model that combines the best of both techniques, CNNs and transformers, to model both local and global dependencies and has been used for speech recognition achieving state-of-the-art performance. While conformers have been mainly applied to sequence-to-sequence problems, in this work we make a preliminary study of their adaptation to a binary classification task such as anti-spoofing, with focus on synthesis and voice-conversion-based attacks. To evaluate our proposals, experiments were carried out on the ASVspoof 2019 logical access database. The experimental results show that the proposed system can obtain encouraging results, although more research will be required in order to outperform other state-of-the-art systems.

Index Terms: Spoofing detection, deep learning, conformers.

1. Introduction

Automatic speaker verification systems (ASV) are designed to determine whether a given speech signal proceeds from a given individual [1]. As the interest in this technology grows, due to its commercial application [2], the same happens to the concerns about its security. ASV systems can be fooled by presenting speech samples resembling the voice of a genuine user. We can distinguish four types of spoofing attacks [3]: (i) replay (i.e. replay of a pre-recorded voice of a genuine user), (ii) impersonation (i.e. mimicking a genuine voice), and (iii) text-to-speech systems (TTS) or (iv) voice conversion (VC) systems to generate artificial speech resembling the voice of a legitimate user. In this paper, we will focus on the detection of logical access (LA) attacks, i.e., TTS- and VC-based spoofing attacks.

Spoofing detection for ASV has become an important topic for researchers in recent years as evidenced by the organization of several evaluation campaigns (challenges) on this specific topic: (i) ASVspoof 2015 [4], which focused on LA attacks (TTS and VC); (ii) ASVspoof 2017 [5], which focused on physical access (PA) attacks (replay attacks) under noisy environments; (iii) ASVspoof 2019 [6], which addressed both the detection of LA attacks generated with the latest TTS and VC technologies and simulated replay attacks under different reverberant acoustic conditions; and the most recent one (iv) ASVspoof 2021 [7], which extended both evaluation datasets, LA and PA, the first with a focus on robustness to channel variation and the second with recordings made in real physical environments and adding a speech deepfake detection sub-challenge. One of the main conclusions extracted from these challenges is that the use of deep neural networks (DNNs) outperforms other conventional approaches [8]–[20].

Recent works have shown that combining transformers and convolutions may yield better performance than using them individually [21]. Thus, CNNs can efficiently extract local features and the attention mechanisms can learn content-based global interactions. In particular, conformers have shown excellent performance in automatic speech recognition (ASR) tasks [22]. In this preliminary work, we attempt to study how to adapt the conformer to a classification and detection problem such as anti-spoofing for ASV. We hypothesize that global and local correlations are relevant for this classification task, thus bringing out conformers as a powerful modeling tool to be reckoned with.

The rest of this paper is organized as follows. Section 2 summarizes related work and describes the proposed adaptation that allowed us to use the conformer for classification. Section 3 describes the experimental setup used in our experiments. Section 4 describes the results achieved with the different adaptations that we propose to adapt the conformer to classification tasks. Finally, we summarize the conclusions derived from this research in Section 5.

2. Conformer-based spoofing detection

This section starts by describing the conformer encoder architecture that allows the modeling of dependencies without regard to their distance. Transformers employ a self-attention mechanism to draw global dependencies between input and output [23]. Conformers add convolutional layers into the transformer architecture, thus improving their robustness and accuracy [22].

The conformer encoder architecture is composed of four modules stacked together: a feed forward module, a multi-headed self-attention block that makes use of a sinusoidal po-
sional encoding (a technique borrowed from Transformer-XL [24]), then the convolutional module, and finally another feed forward module and followed by a layer normalization. This structure is illustrated in Figure 1.

![Diagram](image)

Figure 1: Diagram depicting the main processing blocks of the Conformer encoder architecture.

2.2. Conformer adaptation to classification tasks

We propose two alternative approaches to adapt the conformer architecture to classification tasks. The first approach, depicted in Figure 2 and described in subsection 2.2.1, is inspired by the visual transformer (ViT) [25], which uses an extra learnable class embedding that is prepended to the transformer encoder input. The output of that embedding is the one used for classification.

The second approach, depicted in Figure 3 and described in subsection 2.2.2, makes use of the transformer decoder [23] with some adjustments to make it work for classification. This adaptation allows us to use all the outputs of the classification-adapted conformer encoder. To the best of our knowledge, this is the first time that the transformer decoder structure is adapted to a classification task.

2.2.1. Classifier with conformer encoder and classification token

A block diagram of our first approach for classification using the conformer is illustrated in Figure 2. The first layer is a linear layer that receives the log magnitude spectrogram features with shape $400 \times 256$ and reduces them to $400 \times d$, where $d$ is the dimension of the conformer and dimension 400 corresponds to the number of frames selected (details in Section 3.2). This layer allows us to reduce the dimension of the input sequence. Then we feed the conformer encoder with the output of this previous layer. Thus, the conformer encoder receives a sequence of vectors $x \in \mathbb{R}^d$.

Similar to ViT [25], we prepend a learnable embedding to the conformer input sequence. We refer to this embedding as classification token. The conformer encoder processes it along with the output of the previous linear layer. Finally, the transformed classification token is forwarded to the final linear layer

![Figure 2](image)

Figure 2: Overview of the first proposed classifier. We feed the log magnitude spectrogram features to a linear layer and feed the resulting sequence of vectors to a conformer encoder. To perform classification we add an extra learnable classification token to the sequences and use its output to classify.

in order to classify the input speech signal as either bonafide or spoofing.

2.2.2. Classifier with conformer encoder and decoder

In our second approach (Fig 3), we have placed an adapted transformer decoder at the output of the conformer encoder so that we can use an additional attention mechanism to process all the outputs of the conformer encoder. This decoder has two inputs: the memory input, which is used as the key (K) and value (V) for the attention blocks, and the target input, which is fed to the attention blocks as the query (Q). As previously, the first layer is a linear layer that compresses the log magnitude spectrogram features and feeds the conformer encoder. In turn, its outputs feed the first block of the decoder, a multi-head attention block, as memory input. Regarding the target input we have considered two options:

- In the first option, we take the classification token as target input, identical to the one employed in the previous approach. In this option, the target input will be the learnable classification token that in Figure 3 is labeled as the Classification Token.
- In the second option, the attention block (input Q) is fed with the classification token after going through the encoder block. That is, we feed the adapted transformer decoder with the vector that in the first approach is used for classification. In this option, the target input will be the state of the classification token at the output of the conformer encoder. This is depicted in Figure 3 as the Processed Classification Token.

The idea behind these two approaches, establishing the classification token as the network target, is to train the decoder so that it is able to predict a new refined classification token which is finally processed by the classification linear layer, the same as in the approach of the previous subsection.

Then, the outputs of the multi-head attention block feed a layer normalization and a feed-forward layer. This structure is very similar to the transformer decoder [23], although we do not use the first self-attention block since our target input is not
a sequence. The output of the decoder can be processed by another decoder block or by a linear layer, as depicted in Figure 3, before performing the final classification.

![Diagram](image.png)

Figure 3: Overview of our second proposed classifier with one decoder block.

3. Experimental Setup

In this section, we describe the datasets and evaluation metrics employed for the experiments as well as the spectral analysis performed and the training details.

3.1. Dataset and evaluation metrics

We conducted experiments on the ASVspoof 2019 logical access subset [6]. A summary of their composition in terms of speakers and number of utterances is presented in Table 1.

The database contains bonafide speech and spoofed speech systems. Six TTS, and 3 hybrid systems and were implemented with waveform generation methods including classical vocoding, GriffinLim [29], generative adversarial networks [30], neural waveform models [31], waveform concatenation, waveform filtering [32], spectral filtering, and their combination.

We used the pooled equal error rate (EER) [33] as the primary metric and also report results in terms of the minimum normalized tandem detection cost function (t-DCF) [34].

3.2. Spectral Analysis

Speech signals were analyzed using a Blackman analysis window of 25 ms length with 10 ms of frame shift. Log magnitude spectrogram features (STFT) with 256 frequency bins (512-point FFT) were obtained to feed the neural network.

We truncated the spectrum along the time axis with a fixed size of $T = 400$ frames in order to feed the first layer of the neural network. During this procedure, short utterances were extended by repeating their contents if necessary to match the required length.

3.3. Training Details

The network is trained using the ASVspoof 2019 LA training partition to minimize a cross-entropy loss function. We used the Adam optimizer with a fixed learning rate of $6 \cdot 10^{-5}$. The batch size is set to 128.

According to some preliminary experiments, training computational burden and overfitting can be reduced by selecting 2100 utterances of the development set for validation, for which 0% EER is achieved sooner than for the whole set. Thus, the results in the next section will be given directly on the evaluation set. To further prevent overfitting we use dropout in the decoder and each conformer module, with a 0.35 dropout for the convolutional and attention modules, and a 0.3 dropout for the feed-forward module and the decoder. Also, early stopping was applied when no improvement of the EER of the validation sub-set was obtained after nine epochs.

We also trained the best-performing models with another loss function for comparison. We used the one-class softmax (OC-Softmax) [35] loss function with hyper-parameters: $\alpha = 20$ and $m = 0.9$. For this loss function, we use the Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ to update the weights in the model. We used stochastic gradient descent (SGD) optimizer for parameters in the loss functions. The learning rate is initially set to $3 \cdot 10^{-4}$ with a 50% decay every 10 epochs.

3.4. Results

This section starts with the results obtained with our first adaptation of the conformer to classification tasks (W/O-DECODER), which uses a class embedding similar to the one used in vision transformers [25]. After that, Section 4.2 describes the results of the second proposed approach (W-DECODER) with different systems, 6 TTS, and 3 hybrid systems.
options for the target input. Finally, Section 4.3 describes the results achieved with the OC-softmax loss function [35]. Also, it must be pointed out that the goal of this section is not so much to achieve competitive results as to study what performance can be achieved.

4.3. Results with OC-Softmax loss function

All the previous results are the ones obtained using the classical softmax loss function. We also trained our best two models using the OC-Sofmax loss function [35]. The results, which are shown in Table 5, are better than the ones obtained with the models which have the same number of parameters but trained with only 2 classes. However, they do not outperform the ones trained with 7 classes using the classical softmax loss.

5. Conclusions and future work

In this paper, we have explored the use of conformers for the classification task involved by ASV anti-spoofing. We have shown that the conformer encoder can obtain encouraging results but not better than those of other state-of-the-art anti-spoofing systems, like the ones obtained by the gated recurrent convolutional neural network (GRCNN) in [10] which achieved a 3.85% in EER and 0.0952 in t-DCF (notice however that GRCNN includes phase features and time modeling). We also introduced a new approach for adapting the seq2seq structure of encoder and decoder to classification, showing that this new approach can provide a slight improvement over the previous that only uses the encoder structure with an extra learnable embedding. We theorize this is due to the use of all the outputs of the encoder for classification and not only the output of the classification token. In future work, it would be worthwhile to investigate a sliding-window encoder-decoder approach in order to fully exploit the sequential nature of speech. Under this new approach, the decoder could lead to further reductions of EER since the decoder will be able to process its previous outputs similarly to a sequence-to-sequence model.

6. Acknowledgements

This work was supported by the FEDER/Junta de Andalucía-Consejería de Transformación Económica, Industria, Conocimiento y Universidades Proyecto PY20_00902 and by the project PID2019-104206GB-I00 funded by MCIN/ AEI /10.13039/ 501100011033.

---

Table 2: Results of the W/O-DECODER approach and model size.

<table>
<thead>
<tr>
<th>Dim</th>
<th>Attention Heads</th>
<th>Classes</th>
<th>Param</th>
<th>EER (%)</th>
<th>t-DCF</th>
</tr>
</thead>
<tbody>
<tr>
<td>256</td>
<td>8</td>
<td>7</td>
<td>2.6M</td>
<td>10.09</td>
<td>0.2735</td>
</tr>
<tr>
<td>128</td>
<td>4</td>
<td>7</td>
<td>0.8M</td>
<td>7.80</td>
<td>0.1710</td>
</tr>
<tr>
<td>100</td>
<td>8</td>
<td>7</td>
<td>0.79M</td>
<td>9.14</td>
<td>0.1734</td>
</tr>
<tr>
<td>100</td>
<td>8</td>
<td>2</td>
<td>0.79M</td>
<td>8.25</td>
<td>0.1929</td>
</tr>
<tr>
<td>100</td>
<td>4</td>
<td>7</td>
<td>0.59M</td>
<td>7.56</td>
<td>0.1715</td>
</tr>
<tr>
<td>100</td>
<td>4</td>
<td>2</td>
<td>0.59M</td>
<td>8.93</td>
<td>0.1960</td>
</tr>
<tr>
<td>64</td>
<td>4</td>
<td>7</td>
<td>0.37M</td>
<td>8.58</td>
<td>0.1735</td>
</tr>
</tbody>
</table>

---

Table 3: Results of the W-DECODER1 approach.

<table>
<thead>
<tr>
<th>Decoder Dimension</th>
<th>Attention Decoder Heads</th>
<th>Decoder Blocks</th>
<th>EER (%)</th>
<th>t-DCF</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>10</td>
<td>2</td>
<td>8.835</td>
<td>0.1653</td>
</tr>
<tr>
<td>100</td>
<td>10</td>
<td>1</td>
<td>10.005</td>
<td>0.1974</td>
</tr>
</tbody>
</table>

---

Table 4: Results when the target input is the state of the classification token after going through the conformer and the dimension size of the decoder is 100 (W-DECODER2).

<table>
<thead>
<tr>
<th>Attention Decoder Heads</th>
<th>Decoder Blocks</th>
<th>#Classes</th>
<th>EER (%)</th>
<th>t-DCF</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>2</td>
<td>7</td>
<td>7.517</td>
<td>0.1561</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>2</td>
<td>8.197</td>
<td>0.1853</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>7</td>
<td>8.835</td>
<td>0.1653</td>
</tr>
</tbody>
</table>

---

Table 5: Results of the two approaches trained using OC-Softmax. The parameters selected are the ones that yielded the best performance with the previous loss function.

<table>
<thead>
<tr>
<th>Model</th>
<th>EER (%)</th>
<th>t-DCF</th>
</tr>
</thead>
<tbody>
<tr>
<td>W/O-DECODER</td>
<td>7.695</td>
<td>0.1754</td>
</tr>
<tr>
<td>W-DECODER2</td>
<td>8.867</td>
<td>0.1865</td>
</tr>
</tbody>
</table>
7. References


