Speech Replay Detection with \(x\)-Vector Attack Embeddings and Spectral Features

Jennifer Williams and Joanna Rownicka

The Centre for Speech Technology Research, University of Edinburgh, United Kingdom

j.williams@ed.ac.uk and j.m.rownicka@sms.ed.ac.uk

Abstract

We present our system submission to the ASVspoof 2019 Challenge Physical Access (PA) task. The objective for this challenge was to develop a countermeasure that identifies speech audio as either bona fide or intercepted and replayed. The target prediction was a value indicating that a speech segment was bona fide (positive values) or “spoofed” (negative values). Our system used convolutional neural networks (CNNs) and a representation of the speech audio that combined \(x\)-vector attack embeddings with signal processing features. The \(x\)-vector attack embeddings were created from mel-frequency cepstral coefficients (MFCCs) using a time-delay neural network (TDNN). These embeddings jointly modeled 27 different environments and 9 types of attacks from the labeled data. We also used sub-band spectral centroid magnitude coefficients (SCMCs) as features. We included an additive Gaussian noise layer during training as a way to augment the data to make our system more robust to previously unseen attack examples. We report system performance using the tandem detection cost function (tDCF) and equal error rate (EER). Our approach performed better than both of the challenge baselines. Our technique suggests that our \(x\)-vector attack embeddings can help regularize the CNN predictions even when environments or attacks are more challenging.

Index Terms: automatic speaker verification, spoofing countermeasures, speech replay detection

1. Introduction

Detecting replayed speech is a notoriously difficult task in speech processing. In particular, an adversary might obtain a snippet of recorded speech wherein the human target has used their voice authentically, such as when speaking a passphrase. When an automatic speaker verification (ASV) system is presented with fake speech intending to impersonate a live talker their voice authentically, such as when speaking a passphrase. When an automatic speaker verification (ASV) system is presented with fake speech intending to impersonate a live talker, the target prediction was a value indicating that a speech segment was bona fide (positive values) or “spoofed” (negative values). Our system used convolutional neural networks (CNNs) and a representation of the speech audio that combined \(x\)-vector attack embeddings with signal processing features. The \(x\)-vector attack embeddings were created from mel-frequency cepstral coefficients (MFCCs) using a time-delay neural network (TDNN). These embeddings jointly modeled 27 different environments and 9 types of attacks from the labeled data. We also used sub-band spectral centroid magnitude coefficients (SCMCs) as features. We included an additive Gaussian noise layer during training as a way to augment the data to make our system more robust to previously unseen attack examples. We report system performance using the tandem detection cost function (tDCF) and equal error rate (EER). Our approach performed better than both of the challenge baselines. Our technique suggests that our \(x\)-vector attack embeddings can help regularize the CNN predictions even when environments or attacks are more challenging.

2. Feature Development

2.1. Speech Signal Features

Following from the features analysis for replay attack detection that was presented in [2] for the ASVspoof 2017 challenge, we extracted the following features: Mel Frequency Cepstral Coefficients (MFCCs), Inverted Mel Frequency Cepstral Coefficients (IMFCCs), Rectangular Filter Cepstral Coefficients (RFCCs), Linear Frequency Cepstral Coefficients (LFCCs), Sub-band Spectral Centroid Magnitude Coefficients (SCMCs), and Constant Q Cepstral Coefficients (CQCCs). A description of the features is provided in Table 1. We used static features because preliminary experiments indicated that the dynamic features were not as useful in the spoofing task, especially the second-order features.

We used the IDIAP Bob.ap signal processing library to extract this set of signal processing features from speech audio files [19, 20]. In the case of CQCCs, we used the code provided by the challenge organizers and also described in [11]. For each audio file, the feature extractor output is an \(N \times M\)-dimensional matrix with \(N\) as the number of coefficients and \(M\) as the duration of the file in frames.

Each audio file in the ASVspoof 2019 dataset was of a different length. We pre-processed the extracted features to handle this length variability and to create same-sized feature vectors as input to our CNN classifier. We used a down-sampling
technique in the feature space. This means that for a given coefficient in a given audio file, we down-sampled the number of frames to a constant. The technique preserved the original per-coefficient distributions in a file while also setting the number of frames to a constant. For re-sampling we used the Fast Fourier Transform (FFT) so that the spacing between the original frames, \( s = \delta x \), became: \( s = \delta x \times M / M' \). By doing this, we set the number of down-sampled frames \( M' \) to a constant where \( M' = 10 \). This effectively shortened the audio file duration to 10 frames while preserving the mean and standard deviation of each coefficient in a given file [21].

After this, our signal processing features were represented as a \( N \times 10 \)-dim matrix. We selected \( M' = 10 \) frames based on two motivations: 1) to reduce the overall size and overhead of the dataset for later processing, and 2) to allow our countermeasures to operate on very short audio examples. We then stacked the coefficients on a per-frame basis, in an effort to preserve some temporal nature of the original signal. Finally, each of the signal feature values were re-scaled to be between \(-1\) and \(+1\) using per-feature max values from the training set, applied later to development and evaluation sets. We re-scaled the values in order to align with our selection of activation function for the regression task described in Section 3, which outputs a value between \(-1\) (spoofed) and \(+1\) (authentic).

### 2.2. X-vector Embedding Creation

In this work we also used x-vectors [5] as auxiliary features for the CNN model described in Sec. 3. Our aim was to extract meaningful fixed-size utterance-level vectors representing the factors of variation, namely environment and attack conditions, in the spoofing task. We used our extracted representations to account for these factors in the final spoofing detection task. This effort was to improve the system robustness to unseen conditions by leveraging information about the environment and attack classes from the labels provided for each training example.

The Kaldi Toolkit [22] was used to extract x-vectors representing a joint environment+attack class (which we refer to as env+attack). The input features for the x-vector extractor were 40-dim MFCCs, with 80 filters in a filter bank. The x-vector extractor was a time-delay neural network (TDNN) with the same architecture as in [5], i.e. seven layers with batch normalisation and ReLU activations. The x-vectors were extracted from the sixth layer before the nonlinearity. Differently to the model in [5] though, ours was not trained to classify speakers. The extractor was trained to differentiate between classes jointly representing types of acoustic environments and types of attacks.

The joint env+attack classes were created by combining each category label of variation for the simulated acoustic environments and attack types, i.e. room size, T60 reverberation time, talker-to-ASV distance, attacker-to-talker distance, and replay device quality. From 10 attack type configurations, and 27 acoustic environment configurations (9 attacks plus authentic speech), we created 270 env+attack classes. Training an x-vector extractor to differentiate between env+attack classes enabled us to learn fixed-size representations, capturing both the type of attack and the type of acoustic environment. The classification accuracy of our joint env+attack x-vectors was around 85% for 270 unique classes on a held-out validation set (10% of training), hence these representations were meaningful.

After the x-vector extraction, we reduced the dimensionality of our x-vectors from 512-dim to 10-dim using Linear Discriminant Analysis (LDA). The LDA model also used env+attack classes for training. We selected 10-dim based on the EER from the development set. For 59,400 trials with non-target proportion of 50%, the EER in the env+attack verification task was 23.96% with the LDA backend.

### 2.3. X-vector Embedding Analysis

In this section we show an analysis of how the x-vector embeddings could differentiate between different types of attacks and different types of environments. There are 27 environment classes and 10 attack classes. It was easier to show the analysis for environments and attacks separately, compared to modeling all 270 classes for the jointly trained env+attack embeddings, which are the ones ultimately used in our system.

Environment classification was an easier task (accuracy 86% on the validation set) than attack type classification (accuracy 60% on the validation set), even though the number of classes for classifying attacks was smaller than for classifying environment. We hypothesize that this may be caused partly by data imbalance in the case of attack recognition, compared to the evenly distributed examples for the environment classes.

<table>
<thead>
<tr>
<th>Features</th>
<th>Coeff. num. (N)</th>
<th>( f_{\text{min}} - f_{\text{max}} ) (Hz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MFCC</td>
<td>70</td>
<td>300-8000</td>
</tr>
<tr>
<td>IMFCC</td>
<td>60</td>
<td>200-8000</td>
</tr>
<tr>
<td>RFCC</td>
<td>30</td>
<td>200-8000</td>
</tr>
<tr>
<td>LFCC</td>
<td>70</td>
<td>100-7800</td>
</tr>
<tr>
<td>SCMC</td>
<td>40</td>
<td>100-8000</td>
</tr>
<tr>
<td>CQCC</td>
<td>50</td>
<td>15.62-8000</td>
</tr>
</tbody>
</table>

Table 1: Description of signal features extracted from the raw speech audio. Note CQCC used a specific \( f_{\text{min}} \) from [11].

![Figure 1: Confusion matrix for attack class predictions. The scoring is based on the per attack mean development x-vectors against per attack mean training x-vectors. Class 00 is the bona fide class. For other labels, letters in the first position corresponds to the attacker-to-talker distance (A - lowest, C - highest), letters in the second position corresponds to the replay device quality (A - the best, C - the worst).](image)
the scores for mean x-vectors per attack. First of all, it can be observed that the replay device quality is well captured by the x-vectors (labels with the same letter in the second position). However, the attack-to-talker distance does not seem to be modeled well with the attack x-vectors (e.g. scores for classes AC, BC, and CC are very close to each other, but different than for any other classes). The most evenly distributed scores can be observed for the medium replay device quality classes (AB, BB, CB). Our x-vector embeddings can detect when the replay device is very poor. However, if the replay device quality is near perfect, it is much more difficult to develop the spoofing countermeasures.

![Confusion matrix for environment class predictions](image)

Figure 2: Confusion matrix for environment class predictions using mean development x-vectors against mean training x-vectors. Letters in the first position of the label ID correspond to the room size (a - smallest, c - biggest). Letters in the second position correspond to T60 reverberation time (a - shortest, c - longest). Letters in the third position correspond to the talkerto-ASV distance (a - shortest, c - longest).

We hypothesize that even though the x-vectors do not differentiate very well between every attack and every environment class, they do differentiate between some of them. Furthermore, x-vector embeddings for similar acoustic conditions are close to each other in the x-vector space. Therefore, these can be useful representations complimenting our signal processing features.

2.4. Feature Combination

In our experiments, we explored several combinations of our features while evaluating on the development set. For the first case, we evaluated signal processing features individually. Next, we evaluated x-vector embeddings individually. Finally, we combined signal processing features with x-vector embeddings. Before concatenating the LDA x-vectors to the signal processing features, the LDA x-vectors were scaled with $c = 0.1$ constant. We empirically found the scaling to have a good effect on the final EER and DCF metrics. Scaling x-vectors before concatenation is conceptually similar to applying a fixed LDA-like transform in Kaldi (usually used when i-vectors are concatenated to the input features for normalisation in ASR), which is scaling down the dimensions that are “non-informative”. This has the effect of encouraging stochastic gradient descent (SGD) to ignore non-informative values. Scaled and transformed env+attack x-vectors are denoted as xEAs in our paper. They were concatenated to the signal processing features at the input of the CNN model. We hypothesize that it enabled us to normalize out some factors of variation, subsequently enabling the CNN model to learn more robust representations for the final countermeasure task.

Our system submission to the ASVspoof 2019 Challenge was based on two features: SCMC signal features concatenated with the xEAs vectors (scaled and transformed as described above). For the submission, our dataset consisted of 54,000 training instances and 29,700 development instances. Our training data was therefore of size (54000, 410). These dimensions were based on 40 SCMC coefficients by 10 frames per coefficient, plus the additional 10-dimension x-vectors. The two features were combined via concatenation. In each of the training and development sets, we found 5,400 instances had been labeled as bona fide while the remaining had been labeled as spoof, thus the dataset was very imbalanced for the two targets.

Figure 3: Overview of our machine learning system architecture. Scaled LDA x-vectors (xEAs) were concatenated to the signal processing features (SCMCs). The DNN was a 3-layer convolutional neural network.

3. System Architecture

Our system was implemented with the Keras library with TensorFlow backend. We designed our system to perform regression. To do this we converted categorical target labels to numerical values as follows: “spoof” became $-1$ and “bona fide” became $+1$. Since we used the hyperbolic tangent activation function, the output of our system was therefore a value between $-1$ and $+1$. The challenge evaluation plan called for values with greater negative magnitude to correspond to the “spoof” class while values with greater positive value to correspond to the “bona fide” class. Thus we intentionally set up the problem as a regression task. Our system output could be interpreted to represent the degree of authenticity of the audio especially considering that the countermeasures output by our system were later evaluated in tandem with an ASV system.
3.1. Convolutional Neural Networks (CNN)

Figure 3 shows an overview of our system architecture. The first layer of our CNN was an additive Gaussian noise layer [25] [26][27]. We used this as a form of data augmentation to help the model generalize to unseen conditions. We determined the placement of this layer experimentally and also tried different values for the standard deviation of the noise distribution, finally deciding on $\sigma = 0.001$. The next layer was a batch normalization layer. The CNN consisted of 3 Conv1D layers. The kernel size was set to 3 and we used 32 convolutional filters. Each Conv1D layer included a L2 regularizer [28]. Each Conv1D layer was followed by a max pooling layer with pool size and stride set to 2. Finally, we used a fully connected layer with a single output and the hyperbolic tangent activation function (tanh) [29]. The activation function had the effect of restricting the output between $-1$ and 1.

3.2. System Training

We trained our model using a 10% subset of the training set as a validation set. During training we swept several different parameters. We explored values for standard deviation in the Gaussian additive noise layer, in the range of: [0.000001, 0.000005, 0.00001, 0.0001, 0.001]. We also explored incremental values for L2 regularization between [0.00001, 0.001]. For training, we measured loss using mean-squared-error (MSE). We used early-stopping [30], and monitored validation loss with change $\delta = 0$ and patience $p = 5$ epochs. We used the Adam optimizer [31] with learning rate $lr = 0.001$ and remaining parameters set as default.

4. Results and Discussion

Our system was evaluated using two related metrics. The primary metric was the tandem detection cost function (tDCF) computed in conjunction with an ASV system that was kept hidden from participants [32]. This allowed the organizers to vary the ASV system to evaluate robustness of systems. The secondary metric was equal-error-rate (EER) based on the quality of the countermeasure alone to predict bona fide versus spoofed. We selected our best system for submission to the challenge based on performance with the tDCF metric.

In Table 2, we report our system performance on the development set using the signal features, the $x$-vector features, and our top features combined. We also report the official results for our submission on the evaluation set. For reference, we provide the evaluation performance for both baselines - which used a Gaussian Mixture Model (GMM) and signal features. They were: LFCC-GMM [23] and CQCC-GMM [4]. Our system performed better than the baselines for both metrics on development and evaluation sets (at 3-significant digits). Further experiments on the use of the Gaussian noise layer indicated that using the noise layer (shown by default in Table 2) always performed better than without it (shown as SCMC+xEAs-N in Table 2).

In our earlier analysis, we had found that differences in the replay device quality was captured by the $x$-vectors. When the replay device quality is very good, or perfect, then it is much more difficult to develop spoofing countermeasures. While not reported in this paper, the official detailed performance results from the ASVspoof 2019 challenge organizers also indicate that certain attack types are much more difficult, specifically with the high-quality replay device. That is an important finding for ASV research and our analysis in this paper also supports it.

A potential limitation of our approach is due to our use of the tanh activation function for the CNN regression output. Typically, this activation forces output near the boundaries of -1 and +1, making it more difficult to obtain output scores near the centerline. An interesting analysis of our system might include how the output values are situated within the range of -1 and +1 for various attack types and environmental conditions.

As future work, we would also like to experiment with different input features for the $x$-vector extraction. In this paper, we used frame-level 40- and 60- $x$-vectors, without restricting the frequency range. We encourage future work to look more closely at the best frequency range and the best type of frame-level features to capture differences in the acoustic conditions of an utterance with these $x$-vector embeddings. Different normalization techniques could also be investigated.

5. Acknowledgements

This work was supported in part by the EPSRC Centre for Doctoral Training in Data Science, funded by the UK Engineering and Physical Sciences Research Council (grant EP/L016427/1) and the University of Edinburgh. It was also supported by a PhD studentship from the DataLab Innovation Centre, Ericsson Media Services, and Quorate Technology. The authors thank: Erfan Loweimi, Ondrej Klejch, and Joachim Fainberg (CSTR), and Michael Camilleri (IANC) for their helpful discussions.
6. References


