Spoofing Detection with DNN and One-class SVM for the ASVspoof 2015 Challenge

Jesús Villalba, Antonio Miguel, Alfonso Ortega, Eduardo Lleida

ViVoLab, Aragon Institute for Engineering Research (I3A), University of Zaragoza, Spain
{villalba,amiguel,ortega,lleida}@unizar.es

Abstract

Speaker verification systems have achieved great performance in recent times. However, we usually measure performance on an ideal scenario of speaker conversion that does not modify their voices by different impostors. The fact of impersonating a legitimate user is known as spoofing attack. Recent works show the vulnerability of current speaker verification technology to several types of attacks. Most of these works use non-public databases and different performance measures, which makes difficult to compare approaches. The spoofing challenge (ASVspoof 2015) [7] tries to overcome this problem by proposing a common evaluation framework. This paper describes our submission to the challenge. We proposed to use spectral log-filter-bank and relative phase shift features as input to classifiers based on deep neural networks (DNN). The first of our classifiers used DNN posteriors to decide if the trial is spoof or non-spoof. The second used a bottleneck feature from the DNN as input to a one-class SVM. The one-class SVM models the distribution of legitimate speech, not needing spoofing data for training. We fused the score of the different classifiers to produce our final submission. Our system attained very competitive results with EER<0.05% in 9 out of 10 spoofing types.

1. Introduction

Speaker verification systems have achieved great performance in recent times, due to approaches like joint factor analysis and i-vectors. However, we usually measure performance in an ideal scenario of speaker conversion that does not modify their voices to impersonate the target speakers. The fact of impersonating a legitimate user is known as spoofing attack. With the improvement of the technology, the interest for including it in commercial products has grown. However, recent works show the vulnerability of current speaker verification technology to several types of attacks: replay attack, imitation, voice conversion and synthesis [1, 2, 3, 4].

There are some works that propose countermeasures for the different types of attacks. An extensive summary can be found in [5]. Most works assume the existence of a unique spoof, the one that is the object of its study. That makes the countermeasures to be very specific and they may not generalize well for unknown attacks. For example, countermeasures for voice conversion [3] may not detect speech synthesis. Also, works about speech synthesis [4] usually focus only on one or two types of TTS among all the possibilities available in the market. This is mainly due to the fact that most research groups do not have the means to build a diverse database. The lack of public databases and protocols also complicates comparison of approaches and collaboration between institutions. To overcome these problems, the recently introduced SAS dataset [6] and the spoofing challenge (ASVspoof 2015) [7] based on it propose a common evaluation framework. This challenge focused on detecting 10 types of voice conversion and synthesis attacks.

This paper describes our submission to the challenge. Our systems used spectral log-filter-bank and relative phase shift features (RPS) [8], described in Section 2. The results in [9] indicate that RPS provides robust spoof detection across vocoders. We proposed two classifiers based on deep neural networks (DNN), described in Section 3. The first one uses the DNN posteriors as output. The second one is a novel classifier using a one-class SVM to model the distribution of genuine speech. The SVM works on bottleneck features from a DNN trained to discriminate spoofed from genuine speech. In Section 4, we include a short description of the challenge protocol and detailed discussion of our results on the development and evaluation data. Finally, Section 5 presents our conclusions.

2. Features

2.1. Magnitude Spectrum

Our first features were based on the magnitude value of the fundamental frequency in that frame, the one that is the object of its study. That makes the countermeasures to be very specific and they may not generalize well for unknown attacks. For example, countermeasures for voice conversion [3] may not detect speech synthesis. Also, works about speech synthesis [4] usually focus only on one or two types of TTS among all the possibilities available in the market. This is mainly due to the fact that most research groups do not have the

\[ x(t) = \sum_{k=1}^{K} A_k \cos(\phi_k(t)), \quad \phi_k(t) = 2\pi f_0 k t + \theta_k, \quad (1) \]

where \( f_0 \) is the fundamental frequency in that frame, \( A_k \) are the amplitudes, \( \phi_k(t) \) are the instantaneous phases, and \( \theta_k \) is the initial phase shift of the \( k^{th} \) component. \( K \) is the number of components.
of harmonics that fit into the interval $[0, f_s/2]$ Hz, being $f_s$ the sampling frequency. The RPS is the phase shift between every harmonic and the first harmonic at the point $t_0$ of the fundamental period where $\phi_k(t_0) = 0$. However, we can compute the RPS at any point $t$ as:
$$\psi_k = \phi_k(t) - k\phi_1(t) = \theta_k - k\theta_1 .$$

Then, RPS are wrapped to values in $[-\pi, \pi]$ interval. The RPS reveals a structured pattern of the phase information that we can not see looking at the instantaneous phases [8, 11, 9]. The RPS only makes sense on voiced frames, so unvoiced frames were removed.

The RPS values are not suitable for statistical modeling. First, discontinuities appear due to the wrapping of the parameters, so we unwrapped the phase to avoid discontinuities in the RPS envelope. Second, the unwrapping can create very different envelopes even for very similar signals. We differentiated the RPS in the frequency axis (not time) to normalize the envelopes. Another problem is that, depending of the pitch frequency, each frame has different number of bands. We interpolated the differentiated RPS to obtain a value for each point of the FFT. Finally, we reduced dimensionality by applying a 40 filters filter bank. We tried linear and Mel filter banks, the linear one attained better results. As for the magnitude spectrum, when using GMM classifiers, we decorrelated the filter bank output with a DCT and appended first and second derivatives. We computed the RPS with the help of the COVAREP toolkit [12].

3. Classifiers

3.1. Gaussian mixture models

Our baseline classifier was similar to the one used in [9]. We trained a GMM on genuine speech ($M_{human}$) and another on spoofed speech ($M_{spoof}$) by EM iterations. Then, in the evaluation phase, we computed the log-likelihood ratio (LLR) of the test segment feature frames $D$ given both hypothesis:
$$LLR = \log P(D|M_{human}) - \log P(D|M_{spoof}) .$$

3.2. Deep neural networks

We trained a deep neural network (DNN) for each type of feature (Lin-filtered Spectrum and RPS). The input layer consisted of a sliding window with the current frame in the center and a context of several previous and posterior frames. The output layer was a softmax of dimension 5, one output for the human hypothesis, and one output for each of the four types of spoof in the training set—Actually there were 5 types of attack but, as we point out in Section 4, we considered spoofs 3 and 4 as the case of one-class SVMs [16]. One-class SVM are usually used to find abnormal data. This SVM basically separates all the data points from the origin in the high dimensional space defined by the kernel function. In this manner, we obtain a binary function that describes the probability density function where the normal data lives. This function returns +1 in the small region corresponding to the training data and -1 elsewhere. To train the SVM, we used the libsvm toolkit [17].

As input to the SVM, we used a feature obtained from DNNs. We put a bottleneck layer in the last hidden layer of the DNN. This layer has discriminative information that allows to distinguish between spoofed and normal speech. As the DNN were trained with the training spoofs the method was not totally spoof-independent but we wanted to compare both methods on the evaluation data.

We trained one SVM for each type of feature and the outputs were fused again by logistic regression. This was our contrastive 1 system. The fusion of the four systems, DNN and SVM with both features, was our contrastive 2 system.

4. Experiments

4.1. ASV spoofing challenge

The ASV spoofing challenge provides a standard corpora and evaluation metric to compare different spoofing detection approaches [7]. This challenge tries to stimulate the development of generalized countermeasures able to detect unobserved spoofing attacks. The challenge is based on a database containing genuine and spoofed speech. The spoofed speech is obtained from the original data by applying several voice conversion (VC) and speech synthesis (SS) methods as described in [6]. These spoofing techniques are:

- S1: Frame selection based VC [18].
- S2: VC based on modifying the C1 of the MFCC.
- S3: SS using the hidden Markov model toolkit (HTS), adapting models to the target speaker with 20 utterances [19].
- S4: S3 adapting with 40 utterances.
- S5: VC with Festvox1.
- S6: VC using joint density GMM and maximum likelihood parameter estimation [20].
- S7: Similar to S6 but using line spectrum pair (SLP) instead of MFCC.
- S8: Tensor based VC [21].
- S9: VC using kernel partial least square (KPLS) to implement a non-linear transformation [22].
- S10: SS with MaryTTS2 training models with 40 utterances per target speaker.

Finally, we fused the scores of both networks (Spectrum and RPS) by linear logistic regression with Bosaris toolkit [15]. The DNNs were trained on the training data and the fusion on the development data. This was our primary system.

3.3. One-class SVM

We thought that DNNs could over-fit for the training spoofs. To create a spoof-independent system, we decided to try a classifier that can be trained on non-spoof data only. That is the case of one-class SVMs [16]. One-class SVM are usually used to find abnormal data. This SVM basically separates all the data points from the origin in the high dimensional space defined by the kernel function. In this manner, we obtain a binary function that describes the probability density function where the normal data lives. This function returns +1 in the small region corresponding to the training data and -1 elsewhere. To train the SVM, we used the libsvm toolkit [17].

As input to the SVM, we used a feature obtained from DNNs. We put a bottleneck layer in the last hidden layer of the DNN. This layer has discriminative information that allows to distinguish between spoofed and normal speech. As the DNN were trained with the training spoofs the method was not totally spoof-independent but we wanted to compare both methods on the evaluation data.

We trained one SVM for each type of feature and the outputs were fused again by logistic regression. This was our contrastive 1 system. The fusion of the four systems, DNN and SVM with both features, was our contrastive 2 system.
Table 1: EER(%) for different systems and attack types in development set. The term LF stands for linear filtered and BN Layer indicates the hidden layer where we computed the bottleneck features.

<table>
<thead>
<tr>
<th>System</th>
<th>All attacks</th>
<th>S1</th>
<th>S2</th>
<th>S3-4</th>
<th>S5</th>
</tr>
</thead>
<tbody>
<tr>
<td>GMM LFCC+Δ+ΔΔ</td>
<td>2.852</td>
<td>0.258</td>
<td>8.307</td>
<td>0.066</td>
<td>1.966</td>
</tr>
<tr>
<td>GMM MFCC+Δ+ΔΔ</td>
<td>7.539</td>
<td>1.001</td>
<td>19.65</td>
<td>0.057</td>
<td>6.564</td>
</tr>
<tr>
<td>GMM LF RPS+DCT+Δ+ΔΔ</td>
<td>0.042</td>
<td>0.039</td>
<td>0.029</td>
<td>0.014</td>
<td>0.105</td>
</tr>
<tr>
<td>DNN LF Spectrum</td>
<td>0.039</td>
<td>0.000</td>
<td>0.035</td>
<td>0.062</td>
<td>0.015</td>
</tr>
<tr>
<td>DNN LF RPS</td>
<td>0.161</td>
<td>0.128</td>
<td>0.177</td>
<td>0.047</td>
<td>0.279</td>
</tr>
<tr>
<td>Fusion DNN (Spectrum+RPS) (primary)</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>SVM LF Spectrum BN Layer 2</td>
<td>1.817</td>
<td>0.706</td>
<td>3.194</td>
<td>1.084</td>
<td>0.834</td>
</tr>
<tr>
<td>SVM LF Spectrum BN Layer 3</td>
<td>0.227</td>
<td>0.024</td>
<td>0.449</td>
<td>0.000</td>
<td>0.030</td>
</tr>
<tr>
<td>SVM LF RPS BN Layer 2</td>
<td>3.625</td>
<td>2.744</td>
<td>2.942</td>
<td>0.393</td>
<td>6.702</td>
</tr>
<tr>
<td>SVM LF RPS BN Layer 3</td>
<td>1.349</td>
<td>1.077</td>
<td>1.226</td>
<td>0.084</td>
<td>2.232</td>
</tr>
<tr>
<td>Fusion SVM (Spectrum+RPS) (contrastive1)</td>
<td>0.019</td>
<td>0.000</td>
<td>0.067</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Fusion (DNN+SVM)×(Spectrum+RPS) (contrastive2)</td>
<td>0.002</td>
<td>0.000</td>
<td>0.007</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Fusion DNN Spectrum + GMM RPS</td>
<td>0.002</td>
<td>0.009</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Fusion DNN (Spectrum+RPS) + GMM RPS</td>
<td>0.003</td>
<td>0.009</td>
<td>0.000</td>
<td>0.000</td>
<td>0.001</td>
</tr>
</tbody>
</table>

All methods, except S4 and S10, were trained with 20 utterances of the target speaker.

The challenge database was split into three parts: training, development and evaluation. The training set is intended to train spoofing and genuine speech models (3750 genuine, 12625 spoofed). The development set is used to tune the hyperparameters and train fusion of classifiers (3497 genuine, 49875 spoofed). Finally, spoofing detection performance is measured on the evaluation set (9404 genuine, 184000 spoofed). Spoofing detection performance is measured in terms of spoofing detection equal error rate.

4.2. Results

4.2.1. Development

Table 1 shows EER on the development data for different spoofing detection systems. We present results for each type of attack. We fused S3 and S4, given that both synthesize the speech with the same algorithm. We can see very low error rates in the table. First of all, we want to point out that, in cases like this, we must take into account the significance of the results. According to the “rule of 30” [23], to be 90% confident that the true error rate is ±30% of the measured error rate there must be at least 30 errors. By applying this rule to the full development dataset, we obtain that a system needs to reduce EER by around 0.06% absolute to be significantly better than another. That also means that we cannot measure EER under 0.06% with confidence. If we consider each attack individually the minimum EER raises to 0.22%. Most of our systems have EER under this threshold so, to be able to continue improving performance, we should increase the number of trials.

The first block of the table shows results with our baseline classifier, the GMM. The terms LF and MF indicate that the features are linear or Mel filtered. The linear filter bank clearly outperformed the Mel filter bank in this task. We observed the same trend for RPS and for DNN classifiers. Nevertheless, RPS features yielded EER almost 20 times better than LFCC. The second block presents results with the DNN classifier. The DNN on the log-linear-filtered spectrum improved the GMM with LFCC by 99%. It also improved the GMM with RPS but not significantly. The EER for RPS with DNN was 4 times higher than for RPS with GMM. However, it seems to include information complementary to the Spectrum so the fusion of both DNNs achieved EER=0%—this was our primary system. The third block displays results with bottleneck features on one-class SVM classifiers. The term BN Layer N in the table indicates that we computed the bottleneck features in a DNN with N hidden layers. The BN features were computed in the last hidden layer. We tried several sizes for the BN layer obtaining the best results for a dimension of 10. Using 3 hidden layers was significantly better than using 2 layers. Increasing the number of layers even more did not help. The results of the individual SVM were significantly worse than the results with DNN. However, the fusion of the SVM of Spectrum and RPS–BN 3 hidden layers—we achieved very good performance, not significantly worse than the fusion of DNNs. This was our contrastive 1. In the fourth block, we show some fusions. The fusion of DNN and SVM with both features was our contrastive 2 system. All the fusions attained competitive results with no significant differences.

We wanted to obtain an estimation of how our DNN system would perform on unknown attacks. For that, we experimented training the DNN on only one attack type and evaluating on the others. Table 2 shows the results of this experiment, which we called leave-three-out training. The EER degraded significantly with regard to training with all the spoofs. Attacks S1 and S2 are the most similar, training on S1 produced low EER on S2 and vice versa. On the other hand, S3 and S4 are very different from the others. EER on S3–4 was high when training on the other attacks, and reciprocally, training on S3–4 produced high errors on S1, S2 and S5. The system trained on S5 is the one that generalized the best. For RPS features, training on S5 produced good EER on S1 and S2. However, this was not reciprocal, training on S1 or S2 produced very high error rates on S5. We
also experimented training the DNN with all the spoofs but one (leave-one-out) observing similar patterns. To obtain EER we needed to train with at least three attacks, including S3-4.

4.2.2. Evaluation

Table 3 shows our results on the evaluation data. These error rates were provided by the challenge organizers. For reference, the last line shows the result of the best overall system of the challenge. After the evaluation, the organizers allowed to submit additional systems for post-evaluation analysis. Again, we must take into consideration the significance of the results. For all the trials, 30 errors correspond to an error rate of 0.015%; and for known or unknown attacks to 0.03%.

For known attacks, our best system was the fusion of DNN and SVM with both features (contrastive 2). The rest of fusions were also competitive, not significantly worse than our best. Also the GMM with RPS features performed very well. The fusions of systems with spectral and RPS features significantly improved the results of the individual systems. With regard to unknown attacks, the best system was the GMM with LFCC, which was the worst for known attacks. We also note that spectral features performed better than RPS on unknown attacks. The SVM systems performed worst than the DNN in both types of attacks. The fact that the SVM was trained only on genuine speech did not help to generalize better. When averaging all the attacks the best overall system was our primary.

The organizers also provided EER per attack type for the three systems submitted to the evaluation. Table 4 shows the results, the last line shows the best competing system for reference. For individual spoofs, the error rate corresponding to 30 errors is 0.108%. Our primary is under this error rate for all the attacks but S10. Our systems were significantly better than the best system of the challenge in 6 out of 10 attacks (S1,S2,S5,S6,S7,S9). However, the best system was around 5 times better than ours on S10. That made its average EER lower than all the rest. Our systems did not generalize well for S10. MaryTTS uses MBROLA3 to generate the speech waveform while the rest of spoofs used the STRAIGHT [24] or the MLSA [25] vocoders. This fact seems to indicate that our countermeasures are vocoder dependent.

5. Conclusions

This paper presents our submission to the automatic speaker verification spoofing challenge (ASVspoof 2015). We proposed systems based on magnitude Spectrum and relative phase shift (RPS) features and two DNN based classifiers. The first classifier used the DNN posterior probability for genuine speech as final output. In the second one, we extracted a bottleneck feature from a DNN and fed a one-class SVM with it. The one-class SVM modeled the distribution of genuine speech. We thought that the SVM might work better than the DNN on unknown attacks, given that it was trained only on non-spoof trials. However in the evaluation, the DNN performed better. The fact that the bottleneck DNN was trained only known spoofs limited the generalization capability of the SVM.

In the case of spectral features, DNN improved significantly with respect to the GMM baseline. This is interesting for certain applications. For example, mobile telephony vocoders do not take into account the phase, so we could not use a system based on it. The fusion of systems based on spectrum and RPS significantly improved the results with respect to the individual systems. Looking at our results for each type of attack, our primary have EER< 0.1%—which is the minimum EER that we can measure significantly for a dataset of this size—, for 9 out of 10 attacks. Our system failed on only one spoof with EER=40%. Most of the systems submitted to the evaluation also failed on this attack. This last attack used a different method to generate the speech signal, which indicates that our methods are vocoder dependent and that there is still work to do to create a countermeasure robust to all kind of attacks.

3http://tcts.fpms.ac.be/synthesis/mbrola.html
6. References


