IMPACT OF CHANNEL VARIATION ON ONE-CLASS LEARNING FOR SPOOF DETECTION

Rohit Arora¹, Aanchan Mohan², Saket Anand¹

¹Indraprastha Institute of Information Technology, Delhi (IIIT-D)  
²Synaptitude Brain Health, Vancouver, BC, Canada

ABSTRACT

The value of Spoofing detection in increasing the reliability of the ASV system is unparalleled. In reality, however, the performance of countermeasure systems (CMs) degrades significantly due to channel variation. Multi-conditional training (MCT) is a well-established technique to handle such scenarios. However, which data-feeding strategy is optimal for MCT? is not known in the case of spoof detection. In this paper, various codec simulations were used to modify ASVspoof 2019 dataset, and assessments were done using data-feeding and mini-batching strategies to help address this question. Our experiments aim to test the efficacy of the various margin-based losses for training Resnet based models with LFCC front-end feature extractor to correctly classify the spoofed and bonafide samples degraded using codec simulations. Contrastingly to most of the works that focus mainly on architectures, this study highlights the relevance of the deemed-of-low-importance process of data-feeding and mini-batching to raise awareness of the need to refine it for better performance.

Index Terms—spoof detection, codec simulation, channel variability, one-class learning, mini-batch creation.

1. INTRODUCTION

Spoofing detection in Automatic Speaker Verification (ASV) systems is a well-established problem as evident from numerous research and competitions held to tackle it [1, 2, 3]. Spoofing attacks which are synthetically generated fall in the Logical Access (LA) category [4]. Perennially, numerous CMs have been trained and evaluated to give promising results on datasets consisting of speech samples synthesized under controlled and constrained conditions. This makes the model good at spoof detection, but fails to be robust against factors like channel and condition variability which are quite commonly experienced in real-world scenarios.

MCT is shown to be an effective technique to increase the robustness of the model in such conditions [5, 6, 7, 8]. The CM model is trained on degraded speech samples for it to learn features that are invariant to the channel and conditional variations. This leads us to consider the following questions: What is the optimal data-feeding and mini-batching strategy for handling the added variability introduced due to external conditions? To investigate this question, we used random and various custom batching strategies to train our model on two datasets with different level of variability, both derived from ASVspoof 2019 dataset [4, 9]. The above mentioned experiments were carried out using two data-feeding strategy which will be explained later in the paper.

Many high-performing CM systems, for the ASVspoof 2019 dataset, used various margin-based losses and observed significant performance improvements [10, 11, 12, 13]. These approaches have shown promising results. However, with the data being degraded by codec simulations, the margin values used previously set are not optimal and could lead to the model overfitting on spoofed and bonafide samples. Along with the experiments mentioned above, we also thought it would be interesting to compare Softmax, AM-Softmax, and OC-Softmax loss on this new and relatively more practical codec simulated dataset.

2. EMPIRICAL STUDIES

This section investigates the impact of two different data-feeding strategies and random and various custom mini-batching strategies. The experiments were carried out on two degraded datasets that were derived from ASVspoof 2019 dataset. All the experiments in this section were done with Resnet-OC, as it showed comparable performance with state-of-the-art ensemble systems [14, 15] being a single system.

2.1. Dataset

The following datasets were used for carrying out the experiments:

ASVspoof 2019 LA subset The LA track for ASVspoof 2019 contains bonafide and spoofed speech data generated using 17 different TTS and VC systems. Six of these systems are designated as known attacks, with the other 11 being designated as unknown attacks. For more details about the dataset and rules, refer to [4, 9].

Degraded Dataset The train and development set of the original dataset is joined, and randomly 10,000 samples are chosen for the development set, and the rest 40,224 samples...
are used as a train set. The models are trained on two datasets. Both the train sets are constructed to test the performance of our model under different settings of Bit-rate, Loss, DTX, and µ/a-law parameters for various codec simulations. A list of 16 and 45 codec simulations was applied on the original dataset in a cyclic manner for the first and second datasets, respectively. This implies that there are 2,514 and 893 samples per degradation for the first and second dataset, respectively. This setup provides the chance to test the effect of random and custom batching to handle different levels of variability introduced in the two datasets through codec simulations. More information about the dataset is given in section 6.

The randomly sampled development set and ASVspoof 2019 LA eval set are also degraded in a cyclic manner and are used for validating our model. Whereas, ASVspoof 2021 eval set is used as test set. Ver 1, Ver 2, Deg dev, sim19 and eval21 are names for first, second, simulated development and evaluation set respectively.

| length | Mini-Batch | Degraded | Original |
|-------|------------|----------|----------|
|       |            | Dev | Eval | Dev | Eval |
| 1sec  | Random     | 24.12 | 26.65 | 11.62 | 13.54 |
|       | Custom class | 25.84 | 26.52 | 12.40 | 13.31 |
|       | Custom speak | 24.50 | 27.14 | 12.28 | 12.45 |
| mean  | Random     | 11.04 | 12.16 | 7.24  | 10.30 |
|       | Custom class | 13.50 | 14.20 | 7.90  | 9.25  |
|       | Custom speak | 11.98 | 12.18 | 8.02  | 9.33  |
| max   | Random     | 3.37  | 4.34  | 0.46  | 1.23  |
|       | Custom class | 4.30  | 4.78  | 1.03  | 1.49  |
|       | Custom speak | 4.88  | 5.06  | 1.27  | 1.50  |

**2.2.2. Margin based Softmax loss**

The Equation (2) and (3) formulates the AM-softmax and OC-softmax loss respectively.

\[
\mathcal{L}_{AMS} = \frac{1}{N} \sum_{i=1}^{N} \log \left( 1 + e^{\alpha \left( m - \left( w_{y_i} - w_{\hat{y}_i} \right)^T x_i \right)} \right) \quad (2)
\]

\[
\mathcal{L}_{OCS} = \frac{1}{N} \sum_{i=1}^{N} \log \left( 1 + e^{\alpha \left( m_{\hat{y}_i} - w_{\hat{y}_i} \hat{x}_i \right) (1)_{y_i}} \right) \quad (3)
\]

In the equations (3) and (2), \( \hat{x}_i \) stands for normalised input vector containing LFCC speech features and \( y_i \) stands for the output labels of the \( i \)-th sample. The \( \hat{w}_0 \) is the normalised weight vector which optimizes direction of the target class embedding. This loss function uses two margins \( (m_0, m_1 \in [-1, 1], m_0 > m_1) \) to bound the compact space for the target class in the direction of \( \hat{w}_0 \) and have a wider angular margin for non-target class. Refer to [11] for further details.

Both these loss improve upon the softmax loss by introducing a margin. The AM-softmax makes the embedding distribution compact for both classes. At the same time, OC-softmax compacts it only for the bonafide class. This strategy avoids over-fitting on known spoofed classes and makes the latter more suitable for the task of spoof detection. However, due to channel variation, additive noise increases the angular domain of all samples, even genuine speech, in the embedding space. This makes it essential to compare the performance of basic Softmax loss with OC-softmax. As there are chances, the smaller angles for embedding space of genuine speech might lead to a decrease in performance. Thus, we will test the Softmax loss and variations of angle with OC-softmax loss functions in this paper.

**2.3. Implementation details**

The model architecture is adapted straight away from the One-class classification model OC model [1]. All the parameters were kept the same, with the Adam optimizer being used to update the weights. For pre-processing, we extract 60-dimensional LFCCs (including delta and double deltas) from the audio samples. The frame size was set to approximately 200 ms, and the hop size was 100 ms (50% overlap). PyTorch based LFCC layer was embedded into the original model [2].

1. [https://github.com/yzyouzhang/AIR-ASVspoof](https://github.com/yzyouzhang/AIR-ASVspoof)
2. [https://github.com/rohit18115/ASVspoof2021_OC_model](https://github.com/rohit18115/ASVspoof2021_OC_model)
Table 2. Logical access results of LFCC-Resnet with OC-Softmax loss trained on 3 versions of datasets (Original, Ver1 and Ver2) tested on our degraded development (degdev) set, degraded ASVspoof 2019 (sim19) and organizers evaluation set (eval21).

| Dataset    | Mini-Batch      | Data-feeding Strategies |
|------------|----------------|-------------------------|
|            |                | 1sec mean max           | deg dev| sim19| eval21| deg dev| sim19| eval21| deg dev| sim19| eval21| deg dev| sim19| eval21|
| Original   | Random         | 24.12 26.65 33.36      | 11.04 12.16 34.38 | 3.27  4.34  38.47 |
|           | Custom class   | 25.84 26.52 29.91      | 13.50 14.20 32.54 | 4.30  4.78  34.56 |
|           | Custom speak   | 24.50 27.14 25.80      | 11.98 12.18 32.14 | 4.88  5.06  35.70 |
| Ver 1      | Random         | 18.58 20.32 22.74      | 4.28  7.11  30.90 | 0.61  0.88  37.56 |
|           | Custom sim     | 23.52 24.02 24.40      | 8.02  23.03 26.20 | 2.55  27.56 28.78 |
| Ver 2      | Random         | 19.11 21.70 23.86      | 5.76  9.54  31.35 | 0.79  1.08  38.35 |
|           | Custom sim     | 24.28 24.95 24.66      | 8.91  16.44 26.91 | 3.69  20.65 27.21 |
| Fusion     |                | 18.05 19.68 21.50      |         |       |       |         |       |       |         |       |       |         |       |       |
| Softmax    |                | 20.06 22.88 26.67      |         |       |       |         |       |       |         |       |       |         |       |       |
| AM-Softmax | ($m=0.9$)     | 25.24 27.55 28.68      |         |       |       |         |       |       |         |       |       |         |       |       |
| AM-Softmax | ($m=0.3$)     | 19.85 21.90 23.40      |         |       |       |         |       |       |         |       |       |         |       |       |
| OC-Softmax | ($m_0=0.5$, $m_1=0.2$) | 19.46 19.88 22.75 |         |       |       |         |       |       |         |       |       |         |       |       |

3. RESULTS AND DISCUSSION

All the results are presented in terms of Equal Error Rate (EER [\%]), and the minimum tandem detection cost function (min-tDCF) is omitted. The result obtained on our degraded development set and evaluation set, presented in the Table 1 states that the One-Class classification model incurs a significant loss in performance when trained on the Original ASVspoof 2019 dataset. And this observation is consistent for all data-feeding strategies.

The first 8 rows in Table 2 shows the results of our investigations on the impact of multi-conditional training and various data-feeding and mini-batching strategies as mentioned in section 2.3.1. From the experiments carried out, it can be confirmed that the length of the speech sample and the mini-batching strategy used to decide how well the model generalizes on different datasets. If we compare the performance of the models with respect to the data-feeding strategies, it is evident that the model loses its generalizing capabilities with an increase in the sample size that is being fed to it. The model trained using max training strategy performs exceptionally well on deg dev and sim19 and much better than the models trained using 1sec data-feeding strategy. Nevertheless, when the models are evaluated on eval21, the model trained on shorter speech samples performs better.

It can be observed that the multi-conditional training with random mini-batching done on Ver 1 and Ver 2 does improve performance as compared to when the model is trained on the original dataset. It is also worth mentioning that even though random-batching showed more performance gain as compared to custom batching, the latter helped the model ex-

3We have not used the pre-trained weights given by the author to avoid any inconsistencies with other models we have trained.
tract features that helped the model generalize well on the evaluation set. Furthermore, this is consistent along with all the data-feeding strategies. Although models trained using $max$ length do not generalize well on new datasets, from table 1 it can be observed that they still show remarkable performance on degraded and original, Development and Evaluation sets.

The Figure 1 illustrates the performance of the systems mentioned in the Table 2. We only used the systems trained on 1sec chunks as it gave the most generalized performance. The following observations are made:

- All the systems performed well on wide-band codecs as compared to the narrow-band ones.
- The EER increases in proportion to the increase in loss parameter. This can be because the valuable artifacts that help distinguish spoofed and genuine samples with loss in packets tend to diminish.
- With similar logic if $DTX$ parameter is not used should increase the performance of the systems, but this is true only for codec with high bit-rates but not for codec with the low bit-rate setting.
- Codecs with high bit-rate setting makes spoof detection easier as compared to low bit-rate ones.
- Satellite-based codec simulations showed an exceptionally high increase in EER, for $C^2$ codec the reason can be its low bit-rate.

From Figure 1 it is evident that a weighted score fusion between systems trained on Ver 1 with random batching and Ver 2 with random batch would increase the performance. After few experiments, we found out that the weight of 10 and 90 respectively gave the best performance.

The EER[\%] values of the various settings of loss functions are reported in the last 4 rows of table 2. Note, model were trained using 1sec chunks and random batching for these comparisons. The Figure 2 gives analysis of False Rejection Rate i.e., the number of miss-classifications of genuine samples when LFCC-Resnet model is trained with OC-Softmax,

\[ \text{OC-Softmax-wide}(m_0=0.5, \ m_1=0.2), \] which is the loss with less restricted embedding space for genuine samples, and Softmax loss respectively. From both the EER values and the increase in FRR of the OC-Softmax as compared to the latter confirms that the strict restrain over the embedding space of genuine samples has adverse effect and reducing the restrain might be a better option for the new data setting.

4. CONCLUSIONS

In this paper, we constructed two databases, adding various codec simulations to facilitate our experiments. Results show that the performance of the one-class classification system suffers in the new setting. Encouragingly, multi-conditional training improves performance by 35.55\%. It is observed that random mini-batching gives lower EER as compared to custom mini-batching, whereas the latter generalizes well then the former for the evaluation set. Moreover, it can be confirmed that the length of the speech sample and the mini-batching strategy used to decide how well the model generalizes on different datasets. And a strict restrain on the embedding space over the genuine samples leads to sub-optimal performance and reducing the restrain would be a good option to deal with the added variability due to codec simulations.
5. REFERENCES

[1] Massimiliano Todisco, Xin Wang, Ville Vestman, Md. Sahidullah, Héctor Delgado, Andreas Nautsch, Junichi Yamagishi, Nicholas Evans, Tomi H. Kinnunen, and Kong Aik Lee, “ASVspoof 2019: Future horizons in spoofed and fake audio detection,” in Interspeech 2019. Sept. 2019, ISCA.

[2] Tomi Kinnunen, Md. Sahidullah, Héctor Delgado, Massimiliano Todisco, Nicholas Evans, Junichi Yamagishi, and Kong Aik Lee, “The ASVspoof 2017 challenge: Assessing the limits of replay spoofing attack detection,” in Interspeech 2017. Aug. 2017, ISCA.

[3] Zhizheng Wu, Tomi Kinnunen, Nicholas Evans, Junichi Yamagishi, Cemal Hanılıçlı, and Md Sahidullah, “Asvspoof 2015: the first automatic speaker verification spoofing and countermeasures challenge,” in Interspeech 2015. 09 2015, ISCA.

[4] Junichi Yamagishi, Massimiliano Todisco, Md Sahidullah, Héctor Delgado, Xin Wang, Nicholas Evans, Tomi Kinnunen, Kong Aik Lee, Ville Vestman, and Andreas Nautsch, “Asvspoof 2019: Automatic speaker verification spoofing and countermeasures challenge evaluation plan,” 2019.

[5] Hong Yu, Achintya Sarkar, Dennis Alexander Lehmann Thomsen, Zheng-Hua Tan, Zhiyu Ma, and Jun Guo, “Effect of multi-condition training and speech enhancement methods on spoofing detection,” in 2016 First International Workshop on Sensing, Processing and Learning for Intelligent Machines (SPLINE). July 2016, IEEE.

[6] Xiaohai Tian, Zhizheng Wu, Xiong Xiao, Eng Siong Chng, and Haizhou Li, “An investigation of spoofing speech detection under additive noise and reverberant conditions,” in Interspeech 2016. Sept. 2016, ISCA.

[7] Xiaohai Tian, Zhizheng Wu, Xiong Xiao, Eng Siong Chng, and Haizhou Li, “Spoofing detection under noisy conditions: a preliminary investigation and an initial database,” 2016.

[8] Cemal Hanılıçlı, Tomi Kinnunen, Md Sahidullah, and Aleksandr Sizov, “Spoofing detection goes noisy: An analysis of synthetic speech detection in the presence of additive noise,” Speech Communication, vol. 85, pp. 83–97, Dec. 2016.

[9] Héctor Delgado, Nicholas Evans, Tomi Kinnunen, Kong Aik Lee, Xuechen Liu, Andreas Nautsch, Jose Patino, Md Sahidullah, Massimiliano Todisco, Xin Wang, and Junichi Yamagishi, “Asvspoof 2021: Automatic speaker verification spoofing and countermeasures challenge evaluation plan,” 2021.

[10] Galina Lavrentyeva, Sergey Novoselov, Andzhukaev Tseren, Marina Volkova, Artem Gorlanov, and Alexander Kozlov, “Stc antispoofing systems for the asvspoof2019 challenge,” 2019.

[11] You Zhang, Fei Jiang, and Zhiyao Duan, “One-class learning towards synthetic voice spoofing detection,” IEEE Signal Processing Letters, vol. 28, pp. 937–941, 2021.

[12] You Zhang, Ge Zhu, Fei Jiang, and Zhiyao Duan, “An empirical study on channel effects for synthetic voice spoofing countermeasure systems,” 2021.

[13] Xin Wang and Junich Yamagishi, “A comparative study on recent neural spoofing countermeasures for synthetic speech detection,” 2021.

[14] Bhusan Chhetri, Daniel Stoller, Veronica Morri, Marco A. Martinez Ramírez, Emmanouil Benetos, and Bob L. Sturm, “Ensemble models for spoofing detection in automatic speaker verification,” 2019.

[15] Yexin Yang, Hongji Wang, Heinrich Dinkel, Zhengyang Chen, Shuai Wang, Yannin Qian, and Kai Yu, “The SJTU Robust Anti-Spoofing System for the ASVspoof 2019 Challenge,” in Proc. Interspeech 2019, 2019, pp. 1038–1042.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, “Deep residual learning for image recognition,” 2015.

[17] João Monteiro, Jahangir Alam, and Tiago H. Falk, “Generalized end-to-end detection of spoofing attacks to automatic speaker recognizers,” Computer Speech & Language, vol. 63, pp. 101096, Sept. 2020.

[18] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun, “An empirical evaluation of generic convolutional and recurrent networks for sequence modeling,” 2018.

[19] João Monteiro, Jahangir Alam, and Tiago H. Falk, “Residual convolutional neural network with attentive feature pooling for end-to-end language identification from short-duration speech,” Computer Speech & Language, vol. 58, pp. 364–376, Nov. 2019.

6. APPENDIX

6.1. Dataset

6.1.1. ASVspoof 2019 LA subset

The LA database for ASVspoof 2019 contains bonafide and spoofed speech data generated using 17 different TTS and VC systems. Data used for the training of TTS and VC systems also comes from the VCTK database but there is no overlap.
with the data contained in the 2019 database. Six of these systems are designated as known attacks, with the other 11 being designated as unknown attacks. For more details about the dataset and rules refer to [4] [9]

6.1.2. Codec Simulations and Degraded Dataset

In order to tackle this newly added complexity in a systematic manner we need to be aware of the types of codec simulations and their corresponding parameters that can affect the performance of the countermeasure system. Here’s a brief description of the parameters available for codec simulation:

- **Bit-rate**: It alludes to the quantity of bits that are passed on or processed in a given unit of time. The higher the bit-rate the better the audio quality of the output.

- **Loss**: Gives control over the amount of packets lost during the transmission.

- **mu/a-law**: Common companding algorithms used in telephony system. Both have fairly minimal difference and the advantages of one over the other are insignificant, with $\mu$ – law having higher dynamic range but also has higher distortion for small signals when compared to $a$ – law.

- **DTX**: DTX stands for Discontinuous Transmission, it diminishes the transmission rate during inactive discourse periods while maintaining a respectable level of yield quality.

The above discussed parameters are not common to all the codec simulations, hence we have listed the codecs along with their parameters available for tuning, categorized according to usage, used to create the modified dataset in table 1

| Usage   | Codes | Parameters |
|---------|-------|------------|
| Landline | G.711 | ✓          |
|          | G.726 | ✓          |
| Cellular | AMR-NB | ✓ | ✓ |
|          | AMR-WB | ✓ | ✓ |
|          | GSM-FR | ✓ | ✓ |
| Satellite | G.728 | ✓ | ✓ |
|          | CVSD | ✓ | ✓ |
|          | Codec2 | ✓ | ✓ |
| VoIP     | SILK | ✓ | ✓ | ✓ |
|          | SILK-WB | ✓ | ✓ | ✓ |
|          | G.729a | ✓ | ✓ | ✓ |
|          | G.722 | ✓ | ✓ | ✓ |
| Playback | MP3 | ✓ | ✓ |
|          | AAC | ✓ | ✓ |
| BP filter | G.712 | ✓ | ✓ | ✓ |
|          | P341 | ✓ | ✓ | ✓ |
|          | IRS | ✓ | ✓ | ✓ |
|          | MIRS | ✓ | ✓ | ✓ |

6.2. One-Class Classifier

6.2.1. Overview

Most of the deep learning methods which showed significant performance on the ASVspoof 2019 Challenge dataset were ensemble pipelines. We use One-Class classifier [11] as our baseline model and improvised upon it, as it showed comparable performance with ensemble systems being a single system.

The network architecture of our spoofing detection classifier has following important parts that must be understood: [6.2.2] which talks about the CNN architecture used and the changes made to its layers to make it suitable for spoofing detection; [6.2.1] which talks about the how the Softmax loss is modified for One-class learning.

### Table 3. List of codecs and their corresponding parameters, used to degrade the dataset, categorized according to their usage. The checkmark in the corresponding cell indicates the availability of the parameter for tuning

| Usage | Codes | Parameters |
|-------|-------|------------|
|       | Bit-rate | Loss | mu/a-law | DTX |
| Landline | G.711 | ✓ | ✓ | |
|          | G.726 | ✓ | ✓ | |
| Cellular | AMR-NB | ✓ | ✓ | ✓ |
|          | AMR-WB | ✓ | ✓ | ✓ |
|          | GSM-FR | ✓ | ✓ | ✓ |
| Satellite | G.728 | ✓ | ✓ | ✓ |
|          | CVSD | ✓ | ✓ | ✓ |
|          | Codec2 | ✓ | ✓ | ✓ |
| VoIP | SILK | ✓ | ✓ | ✓ |
|          | SILK-WB | ✓ | ✓ | ✓ |
|          | G.729a | ✓ | ✓ | ✓ |
|          | G.722 | ✓ | ✓ | ✓ |
| Playback | MP3 | ✓ | ✓ | ✓ |
|          | AAC | ✓ | ✓ | ✓ |
| BP filter | G.712 | ✓ | ✓ | ✓ |
|          | P341 | ✓ | ✓ | ✓ |
|          | IRS | ✓ | ✓ | ✓ |
|          | MIRS | ✓ | ✓ | ✓ |

6.2.2. CNN model

We follow the ResNet [16] setup given in [17] [11]. Convolutional neural networks are used in our system, as recent studies show that CNNs are well-suited for time dependency modelling [18]. The use of ResNet helps us deal with issues of vanishing gradients and early convergence, due to its skip connections. The main purpose for the use of convolution function is to shrink dimension of coefficients and create mappings of speech utterances into ‘vector descriptors’ [17].

Inspired from [19], we use self-attentive pooling across time, also called as temporal pooling, which helps us to deal with inputs of varying lengths and assign higher importance to particular parts of inputs.