INTERSPEECH 2021
30 August – 3 September, 2021, Brno, Czechia

A Comparative Study on Recent Neural Spoofing Countermeasures for Synthetic Speech Detection

Xin Wang, Junichi Yamagishi

National Institute of Informatics, Chiyoda-ku, Tokyo, Japan
wangxin@nii.ac.jp, jyamagis@nii.ac.jp

Abstract

A great deal of recent research effort on speech spoofing countermeasures has been invested into back-end neural networks and training criteria. We contribute to this effort with a comparative perspective in this study. Our comparison of countermeasure models on the ASVspoof 2019 logical access scenario takes into account common strategies to deal with input trials of varied length, recently proposed margin-based training criteria, and widely used front ends. We also measured intra-model differences through multiple training-evaluation rounds with random initialization. Our statistical analysis demonstrates that the performance of the same model may be statistically significantly different when just changing the random initial seed. We thus recommend similar statistical analysis or reporting results of multiple runs for further research on the database. Despite the intra-model differences, we observed a few promising techniques, including average pooling, to efficiently process varied-length inputs and a new hyper-parameter-free loss function. The two techniques led to the best single model in our experiment, which achieved an equal error rate of 1.92% and was significantly different in statistical sense from most of the other experimental models.

Index Terms: anti-spoofing, countermeasure, ASVspoof 2019, logical access, deep learning, significance test

1. Introduction

Neural-network-based countermeasures (CMs) are a hot topic in speech anti-spoofing research [1]. Many studies have used neural networks and loss functions [2, 3, 4] proposed for face verification and image classification tasks, for example, light convolution network (LCNN) [5] and margin-based softmax [6, 7]. There is, however, a potential glitch of using an image-oriented neural network for speech because speech trials have varied length, while images do not. No consensus has been reached: some studies trim or pad input speech trials while others squeeze them through pooling or attention. On loss functions, although the margin-based softmax is well supported by empirical studies in the image field, comparative studies for speech anti-spoofing are lacking. Whether or not the margin-based softmax is well worth the efforts to search for a good hyper-parameter set for the margin is unknown.

We have two purposes in this study. First, we summarize and compare the strategies to deal with varied-length input and a few loss functions reported in the recent speech anti-spoofing literatures. Second, we introduce a simple hyper-parameter-free loss function. We built a few CMs by combining the aforementioned components and compared their performance on the ASVspoof2019 logical access (LA) database across front ends based on linear frequency cepstral coefficients (LFCCs), linear filter bank coefficients (LFBs), and spectrograms.

Through training and evaluating each CM multiple times with random initialization, we observed that the performance variation through multiple runs can be more statistically discernible than inter-model differences. This intra-model variation calls for more caution when reporting and interpreting the CM performance on the database. We suggest conducting statistical analysis or simply multiple runs for future work.

Despite the intra-model variation, our results suggest that CMs can efficiently process the varied-length speech trials using attention or simply average pooling. Results on loss functions demonstrate the decent performance of a simple sigmoid function, which makes the gain of the margin-based softmax trivial. The new and simple loss function based on P2SGrad [8] is a competitive alternative. By combining the new loss function and an LCNN with average pooling, one of the CM achieved an equal error rate (EER) of 1.92% in the best training-evaluation round. This is one of the lowest EERs among existing single CMs on the database without data augmentation [3, 4, 9, 10, 11], even though the differences may not be statistically significant.

This paper briefly describes recent neural CMs in Sec. 2 and introduces the P2SGrad-based loss function. It then details the experimental design and results in Sec. 3. Note that this work focuses on the CM back end, a better CM also requires joint efforts from the front end [12, 13] and model ensemble [1, 14].

2. Brief Overview of Neural CMs

We focus on CMs that convert a sequence of acoustic features (e.g., LFCC) into a single score for one input trial, but the description is generalizable to those using a waveform input [11, 15]. We use $\boldsymbol{x}_{1:N^{(j)}} \equiv (\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_{N^{(j)}}) \in \mathbb{R}^{N^{(j)} \times D}$ to denote the input feature sequence of the j-th trial, where $\boldsymbol{x}_n \in \mathbb{R}^D$ is the feature vector at the n-th frame, and where $N^{(j)}$ is the total number of frames. A CM converts $\boldsymbol{x}_{1:N^{(j)}}$ into a score $s_j \in \mathbb{R}$ that shows how likely the input trial is bonafide.

2.1. Neural-network-based back end

A practical issue is that $\boldsymbol{x}_{1:N^{(j)}}$ has varied length $N^{(j)}$. The strategy to handle the varied-length input depends on the back end, and Fig. 1 illustrates three typical cases.

2.1.1. *From fixed-length input to score*

The first strategy assumes a fixed-size input. It pads or trims $\boldsymbol{x}_{1:N^{(j)}}$ into $\tilde{\boldsymbol{x}}_{1:K} \in \mathbb{R}^{K \times D}$ and uses a CNN to transform $\tilde{\boldsymbol{x}}_{1:K}$ into $\boldsymbol{h}_{1:K/L}$, where L is determined by convolution stride. It then flattens $\boldsymbol{h}_{1:K/L}$ and transforms it into the score s_j. The mapping can be summarized as $f : \mathbb{R}^{K \times D} \to \mathbb{R}^{K/L \times D_h} \to \mathbb{R}$. This strategy is used in LCNN, ResNet, or other CNNs-based CMs [2, 3, 16, 17, 18]. The CM may pad short trials with random noises or replicated frames [3, 17]. For trials with $N^{(j)} > K$, the CM may set $\tilde{\boldsymbol{x}}_{1:K} = \boldsymbol{x}_{n:n+K}$, where n is a

http://dx.doi.org/10.21437/Interspeech.2021-702

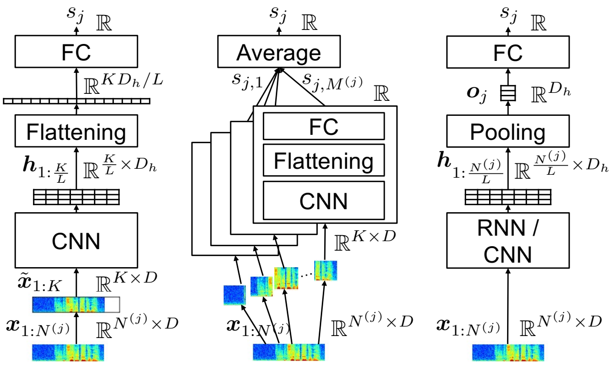

Figure 1: *Typical neural CM backends during inference stage. FC denotes a fully-connected layer. Letter j is a trial index.*

random integer [3] or is equal to one [2].

Based on the above strategy, the second approach usually sets a small K and frames $\boldsymbol{x}_{1:N^{(j)}}$ into fixed-size chunks $(\boldsymbol{X}_1, \boldsymbol{X}_2, \cdots, \boldsymbol{X}_{M^{(j)}})$, where $\boldsymbol{X}_m \in \mathbb{R}^{K\times D}$. It then uses CNNs to convert each chunk $\boldsymbol{X}_m$ into a score $s_{j,m}$ and computes a final score $s_j = \frac{1}{M^{(j)}}\sum_m s_{j,m}$. Some CMs may also pad $\boldsymbol{x}_{1:N^{(j)}}$ into a long sequence before chunking [19].

2.1.2. *From varied-length input to score*

Using a fixed-size input has side-effects: trimming discards information in $\boldsymbol{x}_{K+1:N^{(j)}}$, and padding propagates artifacts [20]. While chunking keeps all the information, the chunks are independently scored by the neural network.

For speech input with varied length, an alternative strategy is $f : \mathbb{R}^{N^{(j)}\times D} \rightarrow \mathbb{R}^{N^{(j)}/L\times D_h} \rightarrow \mathbb{R}^{D_h} \rightarrow \mathbb{R}$. The first step to convert $\boldsymbol{x}_{1:N^{(j)}}$ to $\boldsymbol{h}_{1:N^{(j)}/L}$ is supported by common neural networks, where $L > 1$ if it is a CNN with a large stride or $L = 1$ for a recurrent neural network (RNN). The hidden $\boldsymbol{h}_{1:N^{(j)}/L}$ can be further pooled in an utterance-level vector $\boldsymbol{o}_j = \sum_{m=1}^{N^{(j)}/L} w_m \boldsymbol{h}_m$, and this $\boldsymbol{o}_j \in \mathbb{R}^{D_h}$ can be easily transformed into the score s. The pooling weight w_m can be either uniform or computed by attention [21] [1]. This strategy has been used in a few RNN [24] and CNN-based CMs [25, 26].

2.2. Loss functions

2.2.1. *Cross entropy with vanilla and margin-based softmax*

The CM can be trained by minimizing cross entropy (CE). If the j-th trial has a class label $y_j \in \{1, \cdots, C\}$, the loss function over a dataset $\mathcal{D}$ is defined as

$$\mathcal{L}^{(\text{ce})} = -\frac{1}{|\mathcal{D}|}\sum_{j=1}^{|\mathcal{D}|}\sum_{k=1}^{C}\mathbb{1}(y_j = k)\log P_{j,k}, \quad (1)$$

where $P_{j,k}$ is the probability that the j-th trial is from class k and can be computed through a softmax function. With the rightmost CM in Fig. 1 as an example, we have $P_{j,k} = \frac{\exp(\boldsymbol{c}_k^\top \boldsymbol{o}_j)}{\sum_{i=1}^{C}\exp(\boldsymbol{c}_i^\top \boldsymbol{o}_j)}$, where $\boldsymbol{c}_k \in \mathbb{R}^{D_h}$ is a weight vector for class k. When $C = 2$, we can re-write $P_{j,k}$ as a sigmoid function, e.g., $P_{j,1} = \frac{1}{1+\exp(-(\boldsymbol{c}_1-\boldsymbol{c}_2)^\top \boldsymbol{o}_j))}$. This is widely used in CMs that assign either *bonafide* or *spoof* to the input trial.

[1] We may use RNN to process $\boldsymbol{h}_{1:N^{(j)}/L}$ and use the output from the last RNN step as $\boldsymbol{o}_k$ [15, 22, 23]. However, this approach was found to be inferior to pooling-based ones [24].

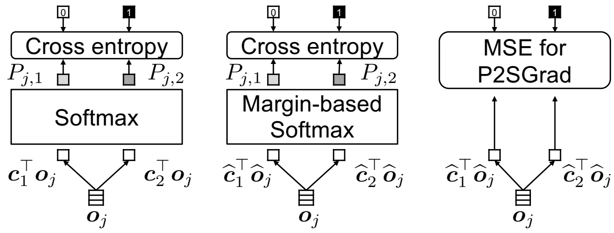

Figure 2: *Loss functions for neural CM back ends and binary classification. Vector o_j is from rightmost network in Fig. 1.*

Advanced CMs enhance the softmax with margins, which can be written in general as [27]:

$$P_{j,k} = \frac{e^{\alpha[\cos(m_1\theta_{j,k}+m_2)-m_3]}}{e^{\alpha[\cos(m_1\theta_{j,k}+m_2)-m_3]} + \sum_{i=1,i\neq k}^{C} e^{\alpha\cos\theta_{j,i}}}, \quad (2)$$

where $\cos\theta_{j,k} = \widehat{\boldsymbol{c}}_k^\top \widehat{\boldsymbol{o}}_j$ is the cosine distance between length-normalized vector $\widehat{\boldsymbol{c}}_k = \boldsymbol{c}_k/||\boldsymbol{c}_k||$ and $\widehat{\boldsymbol{o}}_j = \boldsymbol{o}_j/||\boldsymbol{c}_j||$, and (α, m_1, m_2, m_3) is a hyper-parameter set that defines the type of the margin. For example, an angular-softmax [28] defined by $(m_1 > 1, m_2 = 0, m_3 = 0)$ is used in an LCNN-based CM [2], while an additive-margin (AM) softmax [6, 7] defined by $(m_1 = 1, m_2 = 0, m_3 > 0)$ is used in recent CMs [3, 4]. Note that Eq. (2) may be further extended, for example, by defining a $m_{3,k}$ for each class k. This function is referred to as a one-class (OC) softmax and has been used for CMs [3].

The score in the inference stage is set by either $s = P_{j,1}$ or $s = \widehat{\boldsymbol{c}}_1^\top \widehat{\boldsymbol{o}}_j$ if class label $y_j = 1$ denotes being bonafide.

2.2.2. *New mean-square-error loss function with P2SGrad*

The discriminative power of the margin-based softmax has been testified in a few speech [29] and image processing tasks [3], but its performance is sensitive to the hyper-parameter setting [8]. Here, we propose a hyper-parameter-free loss function:

$$\mathcal{L}^{(\text{p2s})} = \frac{1}{|\mathcal{D}|}\sum_{j=1}^{|\mathcal{D}|}\sum_{k=1}^{C}(\cos\theta_{j,k} - \mathbb{1}(y_j = k))^2, \quad (3)$$

where $\cos\theta_{j,k} = \widehat{\boldsymbol{c}}_k^\top \widehat{\boldsymbol{o}}_j$. This loss is a mean-square error (MSE) between a network's output and a scalar target value, but the network's output here is a cosine distance $\cos\theta_{j,k}$ rather than an unconstrained value. The gradient $\partial\mathcal{L}^{(\text{p2s})}/\partial\boldsymbol{o}_j$ can be shown to be identical to the probability-to-similarity gradient (P2SGrad) [8], and this points to the same optimal direction as the margin-based softmax. We refer to the new loss function as *MSE for P2SGrad*.

Note that the P2SGrad paper defines only the gradient but not the loss function [8]. During inference, we may set $s = \cos\theta_{j,1}$ as the output score if $k = 1$ indicates the bonafide class.

3. Experiments

We conducted the experiments on the ASVspoof 2019 LA database [30] and followed the official protocol without data augmentation. We released the code and all trained models under BSD-3-Clause License [2]. Results that could not fit into this paper are included in the appendix on Arxiv[3].

[2] https://github.com/nii-yamagishilab/project-NN-Pytorch-scripts
[3] Arxiv link: https://arxiv.org/abs/2103.11326

3.1. Experimental CMs

Experimental CMs were from a combination of the network structures and loss functions described in Sec. 2, but not all of them were included due to the limited computation resources:

- `LCNN-trim-pad`: the same LCNN as [2], but the fixed-size input $\tilde{\boldsymbol{x}}_{1:K}$ used $K = 750$ [3]. A shorter trial was padded with zero, and a longer trial was trimmed by $\tilde{\boldsymbol{x}}_{1:K} = \boldsymbol{x}_{n:n+K}$, where n is a random integer;
- `LCNN-attention`: the same LCNN as [2], but layers after the CNN part[4] were replaced with a single-head-attention pooling layer [21] and an FC layer;
- `LCNN-LSTM-sum`: the same LCNN as [2], but layers after the CNN part were replaced with two Bi-LSTM layers, an average pooling layer, and an FC layer. A skip connection was added over the two Bi-LSTM layers, and the size of the Bi-LSTM layers was equal to the dimensions of the CNN part's output.

Both `LCNN-attention` and `LCNN-LSTM-sum` received varied-length input during training and inference, while `LCNN-trim-pad` processed only $\tilde{\boldsymbol{x}}_{1:K}$ for each trial. Note that $K = 750$ is long enough to cover the input features of 98% of the trials in the database when using front ends explained later. Stacking attention to `LCNN-LSTM-sum` was found to be unhelpful in a pilot experiment, while removing LSTMs from `LCNN-LSTM-sum` degraded performance. These two configurations were thus not included in any further experiment. The network in the middle of Fig. 1 was excluded because of the cost of searching for good settings to frame the input feature.

All networks were trained with CE for binary classification, and we assumed $y = 1$ and $y = 2$ denote *bonafide* and *spoof*, respectively. The activation function can be either sigmoid, AM-softmax with $(\alpha = 20, m_3 = 0.9)$ [3], OC-softmax with $(\alpha = 20, m_{3,1} = 0.9, m_{3,2} = 0.2)$ [3], or MSE for P2SGrad. The latter three activation functions had an additional FC layer of size 64 in the front. Accordingly, if we use the rightmost CM in Fig 1 as example, the mapping from input to output score becomes $\mathbb{R}^{N^{(j)} \times D} \rightarrow \mathbb{R}^{N^{(j)}/L \times D} \rightarrow \mathbb{R}^{D_h} \rightarrow \mathbb{R}^{64} \rightarrow \mathbb{R}$.

Each network was tested with three front ends: LFCC, LFB, and spectrogram. The 60-dimensional LFCC followed the ASVspoof 2019 baseline recipe [31]: a frame length of 20ms, a frame shift of 10ms, a 512-point FFT, a linearly spaced triangle filter bank of 20 channels, and delta plus delta-delta coefficients. However, the first dimension was replaced with log spectral energy. The LFB had a similar configuration but contained only static coefficients from 60 linear filter-bank channels. The spectrogram was configured similarly and had 257 dimensions.

When using the spectrogram, a trainable FC layer initialized with coefficients of the linear filter bank for LFB was added before the LCNN. This layer compressed the spectrogram to 60-dimensional hidden features, and it improved the CMs performance by a large margin. Other detailed implementations that cannot be presented here can be found in the released Pytorch codes.

3.2. Training recipe

All the CMs were trained using an Adam optimizer with $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$ [32]. The initial learning rate of 3×10^{-4} was multiplied by 0.5 for every ten epochs [3]. The mini-batch size was 64 or 8, and each mini-batch contained randomly selected trials with similar duration. No voice activity detection or feature normalization was used.

It is known that CM performance varies when the training starts with a different random seed. To fairly compare the CMs with limited computation resources, we trained and evaluated each CM for six times with the random seed for the k-th run set to 10^{k-1}. Note that no randomness occurred during the evaluation in our implementation except the random trimming in `LCNN-trim-pad` , but the randomness was trivial because only 2% of the evaluation trials needed to be trimmed. All the experiments were done using Nvidia Tesla P100. The results are reproducible with the same random seed and GPU environment.

3.3. Evaluation metrics and statistical analysis

We evaluated the CMs by calculating the EERs and tandem detection cost function (t-DCF) [33] using routines from the ASVspoof official website[5].

We also conducted pair-wise statistical analysis using the methodology in [34]. For a pair of models (A, B) in the comparison, a z value was computed using

$$z = \frac{2|\mathrm{EER}_A - \mathrm{EER}_B|}{\sqrt{\Big[\mathrm{EER}_A(1-\mathrm{EER}_A) + \mathrm{EER}_B(1-\mathrm{EER}_B)\Big]\frac{N_{\text{bona}}+N_{\text{spoof}}}{N_{\text{bona}}N_{\text{spoof}}}}}, \tag{4}$$

where N_{bona} and N_{spoof} denote the number of evaluated bonafide and spoof trials, respectively. z was compared with a value $Z_{\alpha/2}$ decided by a significance level $\alpha = 0.05$ and Holm-Bonferroni correction [35], and $z \geq Z_{\alpha/2}$ suggests a statistically significant difference between A and B.

3.4. Experimental results

Table 1 lists the EERs and min t-DCFs on the evaluation set. The outcomes of the pair-wise statistical analysis are plotted in Fig. 3, where white and dark grey indicate insignificant and significant statistical differences, respectively.

We observed that the same model may perform quite differently when simply changing the random seed, and this intra-model difference can be statistically significant, as the blocks on the anti-diagonal line of Fig. 3 demonstrate. Although such variation was expected for neural-network-based CMs, the difference between the best and worst runs of the same model may be larger than the inter-model differences. This finding calls for caution when comparing and interpreting the EERs of different models. Particularly, the commonly used baseline model LFCC `LCNN-trim-pad` achieved a low EER of 2.54% when it was well initialized, which is a stronger baseline than those in many other studies.

Despite the variation, `LCNN-LSTM-sum` with the LFCC and P2S achieved EERs in the range of (1.92%, 3.10%), which is in the same tier as the lowest EERs reported in other studies without data augmentation, e.g., 2.19% in [3], 3.49% in [4], and 4.04% in [1]. The best single run with 1.92% EER is currently the lowest EER value on the database without data augmentation as far as we know, and this EER is statistically significantly different from most of the others, as the bottom row of Fig. 3 illustrates. Some techniques used by the model may be promising for this task.

One of the techniques is the new P2SGrad-based loss function. With the commonly used LFCC front end, the min and max EERs of the P2SGrad-based models were lower than

[4] Specifically layers after 'MaxPool_28' in [2]

[5] https://www.asvspoof.org/asvspoof2019/tDCF_python_v1.zip

Table 1: *EER and min t-DCF on ASVspoof2019 LA evaluation set. For visualization, the results of six training-evaluation rounds were sorted in accord with EER from low (I) to high (VI). A darker cell color indicates a higher EER or min t-DCF value.*

EERs

		AM-softmax						OC-softmax						Sigmoid						MSE for P2SGrad					
		I	II	III	IV	V	VI	I	II	III	IV	V	VI	I	II	III	IV	V	VI	I	II	III	IV	V	VI
LFB	`LCNN-trim-pad`	5.59	6.39	7.12	7.79	9.33	10.72	5.98	6.76	7.02	7.14	9.56	11.34	7.00	7.40	8.69	8.89	9.86	10.13	6.81	7.25	8.10	8.28	8.41	10.21
	`LCNN-attention`	4.26	4.32	4.69	5.23	8.55	14.86	4.01	4.54	4.57	5.27	5.33	5.76	3.34	5.07	5.70	5.91	5.93	5.93	3.99	4.87	5.75	5.85	6.20	7.36
	`LCNN-LSTM-sum`	4.24	5.27	5.71	6.23	7.03	7.10	5.81	6.51	6.89	7.64	9.15	10.24	7.04	7.93	8.52	9.16	9.53	9.84	5.06	5.17	5.69	5.75	6.17	6.50
SPEC	`LCNN-trim-pad`	4.84	5.02	5.23	5.87	5.90	6.25	4.41	4.60	4.84	5.04	5.37	6.35	3.09	3.58	4.72	4.82	5.22	5.56	2.94	3.22	3.59	3.73	4.49	4.74
	`LCNN-attention`	4.02	4.08	4.99	5.22	5.57	8.20	4.05	4.55	4.94	5.67	6.31	8.47	3.92	4.04	4.42	4.91	4.95	6.62	4.70	5.03	5.60	5.88	6.35	6.50
	`LCNN-LSTM-sum`	3.96	4.04	4.38	4.52	5.13	5.97	2.81	3.41	4.49	4.50	4.65	4.91	3.29	3.56	3.82	4.45	4.61	5.44	2.37	2.91	3.00	3.94	4.26	4.37
LFCC	`LCNN-trim-pad`	3.04	3.52	4.73	4.89	5.85	7.06	2.93	2.93	2.95	2.99	3.84	4.00	2.54	2.73	2.77	2.91	3.08	3.47	2.31	2.46	2.64	2.65	3.09	3.11
	`LCNN-attention`	2.99	3.48	3.60	3.61	3.62	3.92	2.91	2.96	3.36	3.40	3.61	3.71	3.18	3.19	3.24	3.29	3.39	4.12	2.72	2.81	2.91	3.03	3.22	3.30
	`LCNN-LSTM-sum`	2.46	3.02	3.23	3.34	3.41	3.86	2.23	2.42	2.96	2.96	2.96	4.64	2.67	2.94	3.20	3.40	3.45	3.53	1.92	2.09	2.43	2.50	2.62	3.10

min t-DCF (legacy version used in ASVspoof 2019 challenge)

		AM-softmax						OC-softmax						Sigmoid						MSE for P2SGrad					
		I	II	III	IV	V	VI	I	II	III	IV	V	VI	I	II	III	IV	V	VI	I	II	III	IV	V	VI
LFB	`LCNN-trim-pad`	0.129	0.168	0.204	0.212	0.235	0.236	0.156	0.179	0.173	0.171	0.200	0.246	0.166	0.175	0.195	0.222	0.229	0.227	0.171	0.212	0.200	0.190	0.217	0.215
	`LCNN-attention`	0.118	0.117	0.093	0.101	0.203	0.331	0.111	0.121	0.122	0.148	0.155	0.160	0.064	0.087	0.126	0.088	0.108	0.092	0.089	0.110	0.124	0.119	0.123	0.131
	`LCNN-LSTM-sum`	0.091	0.141	0.146	0.197	0.195	0.197	0.160	0.184	0.165	0.198	0.210	0.285	0.182	0.201	0.197	0.197	0.231	0.224	0.148	0.147	0.156	0.163	0.172	0.183
SPEC	`LCNN-trim-pad`	0.129	0.132	0.130	0.133	0.141	0.158	0.123	0.122	0.122	0.130	0.134	0.158	0.085	0.099	0.123	0.132	0.137	0.148	0.085	0.088	0.095	0.106	0.126	0.127
	`LCNN-attention`	0.119	0.117	0.131	0.153	0.139	0.156	0.105	0.126	0.141	0.153	0.162	0.194	0.109	0.109	0.115	0.140	0.141	0.173	0.135	0.125	0.143	0.129	0.153	0.183
	`LCNN-LSTM-sum`	0.106	0.102	0.116	0.118	0.136	0.134	0.077	0.088	0.118	0.130	0.121	0.126	0.087	0.105	0.102	0.122	0.122	0.146	0.060	0.077	0.079	0.097	0.097	0.117
LFCC	`LCNN-trim-pad`	0.068	0.081	0.083	0.092	0.090	0.118	0.068	0.068	0.071	0.087	0.103	0.099	0.069	0.069	0.071	0.074	0.074	0.099	0.056	0.061	0.068	0.065	0.074	0.075
	`LCNN-attention`	0.074	0.073	0.071	0.077	0.072	0.074	0.066	0.074	0.074	0.076	0.084	0.084	0.068	0.081	0.076	0.076	0.086	0.101	0.079	0.080	0.080	0.079	0.082	0.091
	`LCNN-LSTM-sum`	0.057	0.068	0.078	0.077	0.079	0.068	0.064	0.064	0.073	0.076	0.076	0.127	0.064	0.063	0.066	0.083	0.073	0.075	0.052	0.058	0.067	0.055	0.067	0.075

those using other loss functions for the three tested network structures. Furthermore, the P2SGrad-based loss had no hyper-parameter. Note that the AM and OC-softmax used the best hyper-parameter configuration reported in the literature [3, 4].

Among the three network structures, `LCNN-trim-pad`'s EER was comparable to the other two when using the LFCC and sigmoid or P2SGrad. However, while `LCNN-attention` and `LCNN-LSTM-sum` had roughly 190 ± 30k and 290 ± 30k parameters, respectively[6], `LCNN-trim-pad` had more than 860k. Furthermore, the single FC layer after a flattening operation in `LCNN-trim-pad` took 710k parameters, which was above 80% of the network size. This monolithic layer conducted $\mathbb{R}^{\frac{K}{L}D_h} \rightarrow \mathbb{R}$, and its parameter size was proportional to the fixed length K of the input features. Such a configuration is inefficient for speech anti-spoofing.

Finally, we observed that the LFCC leads to better performance than the other two frond ends on the database, and the difference was statistically significant as Fig. 3 shows. Except `LCNN-trim-pad` with the AM-softmax, all models with the LFCC had an EER below 4.64%, which is comparable to the best LFCC-based single system in ASVspoof2019 [1]. Although LFB has demonstrated good performance on ResNet in another study [4], it may not be the best choice for LCNN back ends. This possibility will be explored in future work.

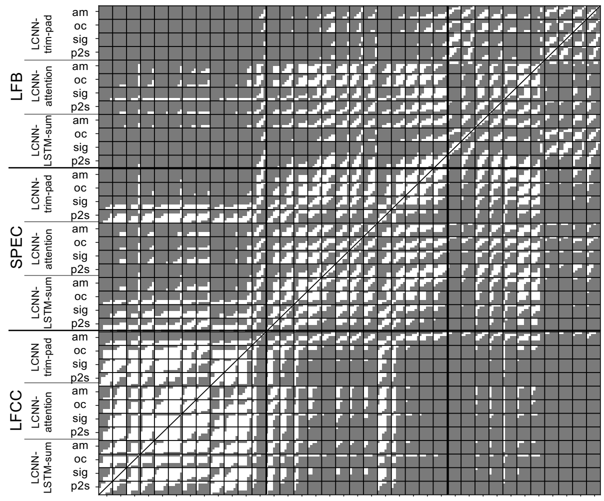

Figure 3: *Statistical significance test on EERs [34], given $\alpha = 0.05$ with Holm-Bonferroni correction. Significant difference is indicated by dark grey, otherwise by white. Each square in the black frames contains 6×6 entries and denotes pair-wise tests between six training-evaluation rounds of two models. The six rounds of each model were in the same order as that in Table 1.*

4. Conclusion

We compared commonly used components for neural-network-based CMs on the ASVspoof2019 LA database. By training and evaluating each CM for multiple runs, we observed that the intra-model differences due to random initial seeds may be statistically significant. We tentatively recommend similar analysis or multiple training-evaluation rounds on this database.

Despite the variation, our results suggest a few promising components for CMs. On neural networks, although CNNs with fixed-size input are widely used, our results suggest that networks with attention or even average-based pooling are potentially more efficient to process varied-length input trials for speech anti-spoofing. On loss functions, the simple sigmoid function is comparable to margin-based softmax for LCNN-based CMs, and a newly proposed loss function with P2SGrad also performed decently through six rounds of training and evaluation using the LFCC front end. With the best combination of LFCC front end, LCNN-LSTM back end, and P2SGrad-based loss function, the lowest EER on the ASVspoof2019 LA evaluation set reached 1.92% in this study. We released the code and trained models for reproducible research.

[6]Parameter size varies due to additional FC layers for different loss functions and spectrogram compression (Sec. 3.1).

5. Acknowledgements

This study is supported by JST CREST Grants (JPMJCR18A6 and JPMJCR20D3), MEXT KAKENHI Grants (16H06302, 18H04120, 18H04112, and 18KT0051), Japan, and Google AI for Japan program. Some of the experiments were conducted on TSUBAME 3.0 of Tokyo Institute of Technology. We thank other ASVspoof2019 organizers and the reviewers for the comments. Some replies are added to the appendix on Arxiv.

6. References

[1] A. Nautsch, X. Wang, N. Evans, T. Kinnunen, V. Vestman, M. Todisco, H. Delgado, M. Sahidullah, J. Yamagishi, and K. A. Lee, "ASVspoof 2019: spoofing countermeasures for the detection of synthesized, converted and replayed speech," *IEEE Transactions on Biometrics, Behavior, and Identity Science*, 2021.

[2] G. Lavrentyeva, S. Novoselov, A. Tseren, M. Volkova, A. Gorlanov, and A. Kozlov, "STC Antispoofing Systems for the ASVspoof2019 Challenge," in *Proc. Interspeech*, 2019, pp. 1033–1037.

[3] Y. Zhang, F. Jiang, and Z. Duan, "One-Class Learning Towards Synthetic Voice Spoofing Detection," *IEEE Signal Processing Letters*, vol. 28, pp. 937–941, 2021.

[4] T. Chen, A. Kumar, P. Nagarsheth, G. Sivaraman, and E. Khoury, "Generalization of Audio Deepfake Detection," in *Proc. Odyssey*, 2020, pp. 132–137.

[5] X. Wu, R. He, Z. Sun, and T. Tan, "A light cnn for deep face representation with noisy labels," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 11, pp. 2884–2896, 2018.

[6] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, and W. Liu, "Cosface: Large margin cosine loss for deep face recognition," in *Proc. CVPR*, 2018, pp. 5265–5274.

[7] F. Wang, J. Cheng, W. Liu, and H. Liu, "Additive margin softmax for face verification," *IEEE Signal Processing Letters*, vol. 25, no. 7, pp. 926–930, 2018.

[8] X. Zhang, R. Zhao, J. Yan, M. Gao, Y. Qiao, X. Wang, and H. Li, "P2sgrad: Refined gradients for optimizing deep face models," in *Proc. CVPR*, 2019, pp. 9906–9914.

[9] A. Luo, E. Li, Y. Liu, X. Kang, and Z. J. Wang, "A Capsule Network Based Approach for Detection of Audio Spoofing Attacks," in *Proc. ICASSP*. IEEE, 2021, pp. 6359–6363.

[10] X. Li, N. Li, C. Weng, X. Liu, D. Su, D. Yu, and H. Meng, "Replay and synthetic speech detection with res2net architecture," in *Proc. ICASSP*. IEEE, 2021, pp. 6354–6358.

[11] H. Tak, J. Patino, M. Todisco, A. Nautsch, N. Evans, and A. Larcher, "End-to-end anti-spoofing with RawNet2," *Proc. ICASSP*, pp. 6369—-6373, 2020.

[12] H. Tak, J. Patino, A. Nautsch, N. Evans, and M. Todisco, "An Explainability Study of the Constant Q Cepstral Coefficient Spoofing Countermeasure for Automatic Speaker Verification," in *Proc. Odyssey*, 2020, pp. 333–340.

[13] B. Chettri, T. Kinnunen, and E. Benetos, "Subband Modeling for Spoofing Detection in Automatic Speaker Verification," in *Proc. Odyssey*, 2020, pp. 341–348.

[14] H. Tak, J. Patino, A. Nautsch, N. Evans, and M. Todisco, "Spoofing Attack Detection Using the Non-Linear Fusion of Sub-Band Classifiers," in *Proc. Interspeech*, 2020, pp. 1106–1110.

[15] H. Dinkel, N. Chen, Y. Qian, and K. Yu, "End-to-end spoofing detection with raw waveform CLDNNS," in *Proc. ICASSP*. IEEE, 2017, pp. 4860–4864.

[16] B. Chettri, D. Stoller, V. Morfi, M. A. M. Ramírez, E. Benetos, and B. L. Sturm, "Ensemble Models for Spoofing Detection in Automatic Speaker Verification," in *Proc. Interspeech*, 2019, pp. 1018–1022.

[17] C.-I. Lai, A. Abad, K. Richmond, J. Yamagishi, N. Dehak, and S. King, "Attentive filtering networks for audio replay attack detection," in *Proc. ICASSP*. IEEE, 2019, pp. 6316–6320.

[18] Z. Wu, R. K. Das, J. Yang, and H. Li, "Light Convolutional Neural Network with Feature Genuinization for Detection of Synthetic Speech Attacks," in *Proc. Interspeech 2020*, 2020, pp. 1101–1105.

[19] C.-I. Lai, N. Chen, J. Villalba, and N. Dehak, "ASSERT: Anti-Spoofing with Squeeze-Excitation and Residual Networks," in *Proc. Interspeech*, 2019, pp. 1013–1017.

[20] B. Chettri, E. Benetos, and B. L. T. Sturm, "Dataset artefacts in anti-spoofing systems: a case study on the ASVspoof 2017 benchmark," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 3018–3028, 2020.

[21] Y. Zhu, T. Ko, D. Snyder, B. Mak, and D. Povey, "Self-Attentive Speaker Embeddings for Text-Independent Speaker Verification," in *Proc. Interspeech 2018*, 2018, pp. 3573–3577.

[22] C. Zhang, C. Yu, and J. H. L. Hansen, "An investigation of deep-learning frameworks for speaker verification antispoofing," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 4, pp. 684–694, 2017.

[23] K. R. Alluri, S. Achanta, S. R. Kadiri, S. V. Gangashetty, and A. K. Vuppala, "SFF Anti-Spoofer: IIIT-H Submission for Automatic Speaker Verification Spoofing and Countermeasures Challenge 2017," in *Proc. Interspeech 2017*, 2017, pp. 107–111.

[24] H. Dinkel, Y. Qian, and K. Yu, "Investigating raw wave deep neural networks for end-to-end speaker spoofing detection," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 11, pp. 2002–2014, 2018.

[25] W. Cai, H. Wu, D. Cai, and M. Li, "The DKU Replay Detection System for the ASVspoof 2019 Challenge: On Data Augmentation, Feature Representation, Classification, and Fusion," in *Proc. Interspeech*, 2019, pp. 1023–1027.

[26] Y. Yang, H. Wang, H. Dinkel, Z. Chen, S. Wang, Y. Qian, and K. Yu, "The SJTU Robust Anti-Spoofing System for the ASVspoof 2019 Challenge," in *Proc. Interspeech 2019*, 2019, pp. 1038–1042.

[27] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *Proc. CVPR*, 2019, pp. 4690–4699.

[28] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, "Sphereface: Deep hypersphere embedding for face recognition," in *Proc. CVPR*, 2017, pp. 212–220.

[29] Y. Liu, L. He, and J. Liu, "Large Margin Softmax Loss for Speaker Verification," in *Proc. Interspeech 2019*, 2019, pp. 2873–2877.

[30] X. Wang, J. Yamagishi, M. Todisco, H. Delgado, A. Nautsch, N. Evans, M. Sahidullah, V. Vestman, T. Kinnunen, K. A. Lee, L. Juvela, P. Alku, Y.-H. Peng, H.-T. Hwang, Y. Tsao, H.-M. Wang, S. L. Maguer, M. Becker, F. Henderson, R. Clark, Y. Zhang, Q. Wang, Y. Jia, K. Onuma, K. Mushika, T. Kaneda, Y. Jiang, L.-J. Liu, Y.-C. Wu, W.-C. Huang, T. Toda, K. Tanaka, H. Kameoka, I. Steiner, D. Matrouf, J.-F. Bonastre, A. Govender, S. Ronanki, J.-X. Zhang, and Z.-H. Ling, "ASVspoof 2019: A large-scale public database of synthesized, converted and replayed speech," *Computer Speech & Language*, vol. 64, p. 101114, nov 2020.

[31] M. Todisco, X. Wang, V. Vestman, M. Sahidullah, H. Delgado, A. Nautsch, J. Yamagishi, N. Evans, T. H. Kinnunen, and K. A. Lee, "ASVspoof 2019: future horizons in spoofed and fake audio detection," in *Proc. Interspeech*, 2019, pp. 1008–1012.

[32] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2014.

[33] T. Kinnunen, K. A. Lee, H. Delgado, N. Evans, M. Todisco, M. Sahidullah, J. Yamagishi, and D. A. Reynolds, "t-DCF: a detection cost function for the tandem assessment of spoofing countermeasures and automatic speaker verification," in *Proc. Odyssey*, 2018, pp. 312–319.

[34] S. Bengio and J. Mariéthoz, "A statistical significance test for person authentication," in *Proc. Odyssey*, 2004.

[35] S. Holm, "A simple sequentially rejective multiple test procedure," *Scandinavian journal of statistics*, pp. 65–70, 1979.