INTERSPEECH 2020
October 25–29, 2020, Shanghai, China

Multi-Task Siamese Neural Network for Improving Replay Attack Detection

Patrick von Platen[1,2], *Fei Tao*[1], *Gokhan Tur*[1]

Uber AI[1]
RWTH Aachen University[2]

patrick.platen@rwth-aachen.de, feit@uber.com, gokhan@uber.com

Abstract

Automatic speaker verification systems are vulnerable to audio replay attacks which bypass security by replaying recordings of authorized speakers. Replay attack detection (RA) systems built upon Residual Neural Networks (ResNet)s have yielded astonishing results on the public benchmark ASVspoof 2019 Physical Access challenge. With most teams using fine-tuned feature extraction pipelines and model architectures, the generalizability of such systems remains questionable though. In this work, we analyse the effect of discriminative feature learning in a multi-task learning (MTL) setting can have on the generalizability and discriminability of RA detection systems. We use a popular ResNet architecture optimized by the cross-entropy criterion as our baseline and compare it to the same architecture optimized by MTL using Siamese Neural Networks (SNN). It can be shown that 26.8% relative improvement on Equal Error Rate (EER) is obtained by leveraging SNN. We further enhance the model's architecture and demonstrate that SNN with additional reconstruction loss yield another significant improvement of relative 13.8 % EER.

Index Terms: replay attack detection, siamese neural networks, multi-task learning, discriminative feature learning

1. Introduction

Automatic speaker verification (ASV) systems are nowadays increasingly used for various applications. However, ASV systems are vulnerable to *audio spoofing attacks*, which attempt to gain unauthorized access by manipulating the audio input. One of the most popular and effective audio spoofing attacks are *replay attacks* (RA)s. In an RA the attacker fools the ASV system by replaying a recording of an authorized speaker. Considering how effective and cheap RAs are, it is necessary to augment an ASV system with an RA detection system in practical applications.

The public benchmark ASVspoof initiative started with the ASVspoof 2015 challenge which dealt with text-to-speech and voice conversion spoofing attacks [1]. ASVspoof 2017 [2] was the first challenge concerned with RA detection and thus created a benchmark data set consisting of voice command recordings. ASVspoof 2019 [3], then introduced a much larger corpus of longer and text-independent recordings for RA detection.

The performance of RA detection systems has been thought highly dependent on their input feature processing [4]. Correspondingly, earlier work has largely dealt with handcrafted feature processing and it has been found that high frequency and phase information can be helpful for RA detection (*e.g.* in [5, 6]). Popular input features that emerged include linear frequency cepstral coefficients (LFCC) [7] and group delay (GD) grams [8]. In recent years, input features derived from shorter handcrafted feature processing pipelines, such as the log power magnitude spectra (*LOGSPEC*) [9], attracted more interest. In contrast to LFCC, LOGSPEC preserves much more of the information present in the original raw signal and thus relies on deep neural netwokrs (DNN)s as powerful feature extractors [9, 10, 11, 12, 13]. Overall, there is currently no conclusive consensus about the best input feature for RA detection.

As the quality of recording and replaying devices is getting better, detecting the difference between genuine and spoofed audios is becoming more difficult. Thus, it becomes necessary to improve the discriminability and generalizability of RA detection systems. Besides common regularization techniques, like data augmentation and Dropout (*cf.* with [10, 13]), multiple teams have used discriminative loss functions and multi-task learning (MTL) [14] for better feature discrimination and generalization (*cf.* with [9, 12, 15]).

Siamese Neural Networks (SNN) [16] have shown to significantly improve the discriminability and generalizability of models [17]. In this paper, we propose to use SNN in an MTL setting for RA detection. More generally, we investigate to what extent adding discriminative loss functions in a MTL setting can improve the performance of RA detection systems on the ASVspoof 2019 challenge Physical Access (PA) data. The analysis is conducted on multiple input features. It is made sure that none of the systems rely on additional data and labels and that all of our settings follow the real-world application implementation. Summarized, our main contributions include:

1. Proposal of SNN in MTL setting for improved discriminability and generalizability of RA detection systems.
2. Extensive analysis of discriminative loss functions on multiple input features.
3. Enhancement of a popular architecture for RA detection with second-order statistics pooling.
4. Combination of reconstruction loss (ReL) with SNN in an MTL setting.

2. Related Work

Convolutional neural networks (CNN)s and especially deep residual neural networks (ResNet)s [18] have yielded the state-of-the-art performance on the ASVspoof 2019 PA data set [9, 10]. To deal with the much smaller data set than the one ResNet was originally designed for [9, 10, 15] significantly reduce the size of their models by scaling down the number of kernels employed in each of the CNN layers. A key component in the architecture of their models is the projection of ResNet's three dimensional tensor output to a one dimensional vector for further binary classification:

- In [15], a recurrent layer processes the tensor along the time dimension and outputs the last hidden state.
- In [9] and [10], it was found that a simpler and more effective approach consist of using a global average pooling (GAP) layer instead.

http://dx.doi.org/10.21437/Interspeech.2020-86

Given the success of ResNet with GAP, we use this architecture as our baseline in this study. In other fields of research it has been shown that using second-order statistics in addition to first-order statistics yields better feature embeddings for utterance level classification tasks, *e.g.* in [19]. This led us to extent the GAP layer to additionally perform variance pooling.

In [15], MTL has been applied for RA detection in the form of center loss (CL), which has been shown to greatly improve the discriminability of a model [20]. CL is comprised of the cross-entropy (CE) loss and the *intra-class* variance loss of the feature embeddings weighted by a hyper parameter to control the intra-class compactness [20]. SNN are known to significantly improve the discriminability and generalizability of a model [17] and have found to be effective in similarity assessment in computer vision [21]. By using a pair of input features during training, SNN simultaneously increase the *inter-class* variance of the embedded input features while decreasing the intra-class variance of the embedded input features. Since CL can be seen as a special case of SNN [1], SNN are expected to better improve the discriminability of the model. This inspired us to propose SNN in a MTL setting for RA detection.

Another loss function, which is easily applicable in the MTL setting, is ReL. ReL is an unsupervised loss function and is usually employed in autoencoders to improve the network's ability to maintain the most distinctive information about the input features in compressed form. When added to a standard CE loss function, ReL can act as an effective regularizer by encouraging the network to learn robust feature embeddings [22].

3. Proposed Approach

3.1. Audio Preprocessing

In a real-world application, the speakers input can be considered as a continuous buffer of audio input, which is then used to decide between genuine and spoofed input. The buffer size is in the following defined as l_b. In all experiments l_b is set to reasonable values to keep the audio processing step simple and easy to deploy. Therefore, all utterances are cut or zero-padded to have a maximum length of l_b seconds and all models **only** process whole utterances, which makes such an RA system easy to deploy.

3.2. Feature Extraction

The models are tested on the three input features: linear frequency filterbank features (*LFBANK*), *LOGSPEC* and *GD grams*. LFBANK correspond to the conventionally used LFCC features without the discrete cosine decorrelation step. We chose to leave out this decorrelation step because neural networks are known to act as excellent decorrelators. LOGSPEC and GD grams are extensively used in the RA research community and preserve more information of the raw audio signal than LFBANK because the short time Fourier transformed (STFT) input signal is **not** further filtered.

For all input features, the STFT employs a window size of 50ms and a window shift of 15ms. LOGSPEC features are retrieved by first squaring the STFT features to obtain their spectrum and subsequently applying a *log* compression. LFBANK are extracted by further applying 80 evenly spaced triangular filters (*cf.* with [23]) to the LOGSPEC features. No delta coefficients are added. We used modified GD grams as defined in Eqs. (28) and (29) in [8] with $\alpha = 0.4$ and $\gamma = 0.9$ because the GD grams as formalized in [6] and [10] did not yield any reasonable results in our experiments.

The resulting feature shape (with *feature dim* $\times$ *time dim*) for GD gram and LOGSPEC is 401×566 and for LFBANK amounts to 80×566.

Confirming the observations made in [9] and [23], the input features are **not** normalized, but simply scaled down to be in the range from -1 to 1.

[1]The centroid used in CL can be seen as one of the inputs in the input pair used for SNN.

Table 1: *Architecture of "thin" 34-layer ResNet. GD/LOGSPEC and LFBANK employ slightly different stride kernels as can be seen on the right. The first ResNet block (Res1) e.g., consists of 3 ResNet layers each having 2 CNN layers. All CNN layer kernels employ the kernel* $[3 \times 3]$.

Layer	Kernel	Filters	Input feature	Stride
Conv1	$[3 \times 3]$	16	GD/LOGSPEC LFBANK	$[2 \times 2]$ $[2 \times 2]$
Res1	$3 \times \begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$	16	GD/LOGSPEC LFBANK	$[2 \times 2]$ $[1 \times 1]$
Res2	$4 \times \begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$	32	GD/LOGSPEC LFBANK	$[2 \times 2]$ $[1 \times 2]$
Res3	$6 \times \begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$	64	GD/LOGSPEC LFBANK	$[1 \times 1]$ $[2 \times 2]$
Res4	$3 \times \begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$	128	GD/LOGSPEC LFBANK	$[1 \times 1]$ $[2 \times 2]$

3.3. ResNet for Replay Attack Detection

Similar to [10], the RA detection system is built upon a "thin" 34-layer ResNet, which is presented in detail in Table 1. The ResNet blocks (*i.e. Res1 - Res4*) employ the "full pre-activation" residual unit proposed in [24]. Due to differences in the feature dimension between LOGSPEC/GD gram and LFBANK, slightly different stride kernels are used in our architecture (*cf.* with Table 1).

The ResNet network is followed by a GAP layer as explained in Eq. (1) in [10]. Extending GAP to the retrieval of second-order statistics, we define a global average and variance pooling (GAVP) layer that extracts both the mean and variance from all feature maps of ResNet's last CNN layer. To keep the number of parameters constant, the pooling layer is followed by a 128×64 dense layer if GAP is employed and a 256×32 dense layer if GAVP is employed. The final dense layer (called "Out" in Fig. 1) following the GAP or GAVP layer has a **single output neuron** with Sigmoid activation function yielding the probability of the input feature to be classified as being *spoofed*. All layers except the pooling and final layer make use of the Rectified Linear Unit (ReLU).

The model has about 1.34 million trainable parameters.

3.4. Multi-Task Learning with Siamese Neural Networks

SNN are made of two sub-networks which share the same set of trainable parameters so that a pair of input features $\mathbf{X}_1, \mathbf{X}_2$ is used as an input during training. Besides computing the conventional CE loss for each sub-network individually (*i.e.* $\mathcal{L}_{\text{CE}_1}$, $\mathcal{L}_{\text{CE}_2}$), a distance loss $\mathcal{L}_{\text{SNN}}$ between the feature embedding (*i.e.*

$\mathbf{e}_1, \mathbf{e}_2$) of each sub-network is calculated (*cf.* with Fig. 1). A common choice for $\mathcal{L}_{\text{SNN}}$ is the hinge loss (*cf.* with [25]):

$$\mathcal{L}_{\text{SNN}} = \mathop{\mathbb{E}}_{(\mathbf{X}_1, y_1), (\mathbf{X}_2, y_2) \sim \mathcal{D} \times \mathcal{D}} [\max(0, m - l_d d(\mathbf{e}_1, \mathbf{e}_2))], \quad (1)$$

wheres m represents the margin. The boolean l_d equals 1 if the input feature labels y_1, y_2 are equal or else -1 and $d(\mathbf{e}_1, \mathbf{e}_2)$ is a distance metric of choice for which we empirically found the cosine similarity:

$$d(\mathbf{e}_1, \mathbf{e}_2) = \frac{\mathbf{e}_1 \mathbf{e}_2}{|\mathbf{e}_1|_2 |\mathbf{e}_2|_2}, \quad (2)$$

with $|\cdot|_2$ being the euclidean norm, to work best. During training, SNN aims at minimizing the sum of $\mathcal{L}_{\text{CE}_1}$, $\mathcal{L}_{\text{CE}_2}$ and $\mathcal{L}_{\text{SNN}}$, whereas each loss contributes with equal weight.

Optionally, two ReLs ($\mathcal{L}_{\text{ReL}_1}, \mathcal{L}_{\text{ReL}_2}$) - one for each sub-network - can be added to the overall loss. In this case a shared decoder (with a negligible amount of parameters) is used to reconstruct the pair of input features $\mathbf{X}_1, \mathbf{X}_2$ from the outputs $\mathbf{O}_1, \mathbf{O}_2$ of the last convolutional layer:

$$\mathcal{L}_{\text{ReL}_i} = |\mathbf{X}_i - \text{Decoder}(\mathbf{O}_i)|_F^2, \forall i \in \{1, 2\}, \quad (3)$$

with $|\cdot|_F$ being the Frobenius norm. The decoder consists of three consecutive Deconvolution, each of which upsamples the input using a stride kernel of size $[2 \times 2]$. The decoder has three such layers which employ in decreasing order $32, 16$ and 8 kernels each of which is of size $[3 \times 3]$. The outputs of Decoder($\mathbf{O}_1$) and Decoder($\mathbf{O}_2$) are "mean-pooled" over their 8 output feature maps. To make sure the "mean-pooled" feature maps output has exactly the same shape as the input feature matrix ($\mathbf{X}_i$), it is zero-padded on the sides. The complete architecture of SNN is illustrated in Fig. 1.

As can be noted from Eq. (1), the space of possible training samples for SNN includes all pair-wise combinations of $\mathcal{D}$ with itself, which is prohibitively large. A simple remedy taken in this study is to control the dataset's size by a hyper parameter *numSamples*. Before every epoch, a dataset $\mathcal{D}_{SNN}$ is created by the following simple, but effective sampling procedure:

function CREATESNNDATASET($numSamples, \mathcal{D}$)
  SHUFFLE($\mathcal{D}$)
  // decompose data set into pos and neg class data sets
  $\mathcal{D}_{pos}, \mathcal{D}_{neg} \leftarrow$ SPLITBYLABEL($\mathcal{D}$)
  $c_{pos}, c_{neg} \leftarrow 0$
  for i **in** 1 **to** $numSamples$ **do**
    for j **in** 1 **to** 2 **do**
      $\mathcal{D}_r \leftarrow$ RAND($\mathcal{D}_{pos}, \mathcal{D}_{neg}$)
      if $\mathcal{D}_r == \mathcal{D}_{pos}$ **then**
        $\mathbf{X}_j \leftarrow \mathcal{D}_{pos}[c_{pos}]$
        $y_j \leftarrow 1$
        $c_{pos} \leftarrow (c_{pos} + 1) \bmod$ LEN($\mathcal{D}_{pos}$)
      else
        $\mathbf{X}_j \leftarrow \mathcal{D}_{neg}[c_{neg}]$
        $y_j \leftarrow 0$
        $c_{neg} \leftarrow (c_{neg} + 1) \bmod$ LEN($\mathcal{D}_{neg}$)
    $\mathcal{D}_{SNN}[i] \leftarrow ((\mathbf{X}_1, y_1), (\mathbf{X}_2, y_2))$
  return $\mathcal{D}_{SNN}$

Hereby, $\mathcal{D}_{pos}$ only consists of *spoofed* utterances and $\mathcal{D}_{neg}$ represents the data of *genuine* utterances with $y_j = 1$ being the label for a *spoofed* utterance and $y_j = 0$ for a *genuine* one $\forall j \in \{1, 2\}$. First, no data sample in $\mathcal{D}_{pos}$ (or $\mathcal{D}_{neg}$, respectively) is used twice *before* every other data sample has been sampled at least once, which ensures almost certainly that all data samples are used per epoch by setting *numSamples* accordingly. Second, it is ensured that the space of possible sample pairs $\mathcal{D} \times \mathcal{D}$ is vastly explored by shuffling the order of $\mathcal{D}_{pos}$ and $\mathcal{D}_{neg}$ before every epoch. Third, by choosing to sample from $\mathcal{D}_{pos}$ or $\mathcal{D}_{neg}$ with even probability, the smaller of the two data sets is upsampled so that $\mathcal{D}_{SNN}$ is balanced.

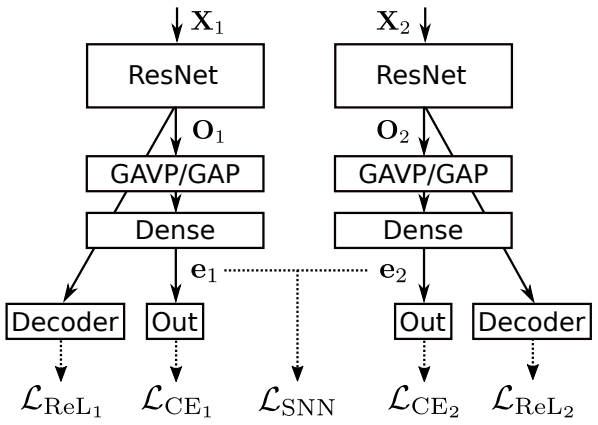

Figure 1: *A sketch map of SNN for MTL showing the loss functions $L_{CE_1}, L_{CE_2}, L_{SNN}$ and L_{ReL_1}, L_{ReL_2} for RA detection. Please note that the filled arrows represent connections between layers, whereas the dotted arrows show schematically from which outputs the losses are derived.*

4. Experimental Setup

In all experiments, the models were evaluated on the PA subset of the ASVspoof 2019 corpus [3]. PA consists of 48600 *spoofed* and 5400 *genuine* utterances in the training (*train*) data set, 24300 *spoofed* and 5400 *genuine* utterances in the development (*dev*) data set and 116640 *spoofed* and 18090 *genuine* utterances in the evaluation (*eval*) data set. The utterance lengths range from *ca.* 3 seconds to *ca.* 11 seconds and consists of more than 100 (both male and female) different speakers. It is important to notice that in contrast to the ASVspoof 2017 PA corpus, the PA subset of the ASVspoof 2019 corpus is *text-independent*, which makes it a much more realistic data set for RA detection.

All models were optimized by Adam with $\beta_1 = 0.9$, $\beta_2 = 0.999$, learning rate 3.95×10^{-4}. To regularize the network, dropout was applied to all CNN layers with a probability of $p = 0.1$ during training. Additionally, we found it to be effective to tune weight decay for each experiment separately. Training was stopped if the equal error rate (EER) on the dev data set did not improve over 15 consecutive epochs or at a maximal epoch of 75. The models were implemented with the Keras framework [26]. The Bosaris toolkit was used for system fusion [27].

5. Results

First, we analysed the effect of audio input length on the performance of a smaller 18-layer ResNet model using LFBANK on the eval set. In this experiment, we simply cut or padded the end of the audio to the specific length. We noticed that increasing the input length l_b from 5.0s to 6.5s and eventually to 8.5s, improved the EER from 9.31 % to 6.75 % and finally to 6.22 %.

Table 2: *Comparison of M_{CE}^{GAP}, M_{CL}^{GAP}, M_{SNN}^{GAP} for different input features. Results are reported in % EER on both Dev and Eval data. LOGSPEC clearly outperforms all other input features. Also there is a quite obvious ranking of model architectures being $M_{CE}^{GAP} < M_{CL}^{GAP} < M_{SNN}^{GAP}$.*

Model	Loss	Input Feature	Dev	Eval
$M_{\text{CE}}^{\text{GAP}}$	$\mathcal{L}_{\text{CE}}$	LFBANK	3.70	5.17
		GD Gram	6.20	8.63
		LOGSPEC	1.98	2.79
		Fused	-	2.22
$M_{\text{CL}}^{\text{GAP}}$	$\mathcal{L}_{\text{CE}}+$ $0.001\mathcal{L}_{\text{CL}}$	LFBANK	2.76	4.06
		GD Gram	4.44	7.13
		LOGSPEC	1.37	2.33
		Fused	-	1.70
$M_{\text{SNN}}^{\text{GAP}}$	$\mathcal{L}_{\text{CE}_1}+$ $\mathcal{L}_{\text{CE}_2}+\mathcal{L}_{\text{SNN}}$	LFBANK	2.73	3.66
		GD Gram	3.53	5.89
		LOGSPEC	**1.15**	**2.25**
		Fused	-	1.52

Based on these findings and existing literature (*cf.* with [23]), we concluded that the end of an utterance includes important discriminative information, thus leads to better performance. Considering this and our practical application, we decided to set the input length l_b to 8.5s and to do cutting and padding at the end of the audio from now on (so that we do not rely on voice activity detection in practical applications).

We then analysed the proposed model architecture with GAP [10]. As one baseline, the model was trained using simple CE loss ($\mathcal{L}_{\text{CE}}$) [15], which is abbreviated as $M_{\text{CE}}^{\text{GAP}}$. As another baseline, the model was trained using CL loss ($\mathcal{L}_{\text{CE}} + \gamma\mathcal{L}_{\text{CL}}$), which is abbreviated as $M_{\text{CL}}^{\text{GAP}}$. For $M_{\text{CL}}^{\text{GAP}}$, we found that $\gamma = 0.001$ yields the best results. The baselines are compared to SNN as described in Section 3.4, which we abbreviate as $M_{\text{SNN}}^{\text{GAP}}$. For $M_{\text{SNN}}^{\text{GAP}}$ the *numSamples* was set to 1×10^6 and the margin m was set to 0.5. In all training setups, we used a batch size of 32. $M_{\text{CE}}^{\text{GAP}}$, $M_{\text{CL}}^{\text{GAP}}$ and $M_{\text{SNN}}^{\text{GAP}}$ were all evaluated using LFBANK, LOGSPEC and GD gram as input. In a final step, the systems were systematically fused by means of logistic regression using the *dev* data set for calibration.

Due to the data imbalance of 9 to 1 in the training set, we adopted the weighted CE loss for $M_{\text{CE}}^{\text{GAP}}$ and $M_{\text{CL}}^{\text{GAP}}$ with the CE weight for *spoofed* utterance input set to $\frac{1}{9}$. To improve training stability, the bias of the output neuron was initialized to $\log(9)$ (*cf.* with [28]) if weighted CE was used. The results can be seen in Table 2. It can be seen that both MTL models $M_{\text{CL}}^{\text{GAP}}$ and $M_{\text{SNN}}^{\text{GAP}}$ outperform the single task learning model $M_{\text{CE}}^{\text{GAP}}$ by relative 23.4 % EER and 31.5 % EER averaged over all input features. $M_{\text{SNN}}^{\text{GAP}}$ further outperforms $M_{\text{CL}}^{\text{GAP}}$ by a relative margin of 10.6 % EER. We could observe that during training, the MTL setups $M_{\text{CL}}^{\text{GAP}}$ and $M_{\text{SNN}}^{\text{GAP}}$ converged faster and also seemed to generalize better as the EER on the dev data set decreased much smoother during training.

In the second experiment, we took the best performing model $M_{\text{SNN}}^{\text{GAP}}$ for LOGSPEC input as our new baseline. First, we analysed the effect of extracting second-order statistics in addition to first-order statistics from the CNN feature maps by replacing the GAP layer with a GAVP layer. This setup is abbreviated as $M_{\text{SNN}}^{\text{GAVP}}$. Second, we extended SNN with two additional reconstruction loss functions $\mathcal{L}_{\text{ReL}_1}$, $\mathcal{L}_{\text{ReL}_2}$ according to Eq. (3) for both GAP ($M_{\text{SNN,ReL}}^{\text{GAP}}$) and GAVP ($M_{\text{SNN,ReL}}^{\text{GAVP}}$). Empirically, it was found that $\mathcal{L}_{\text{ReL}_1}$ and $\mathcal{L}_{\text{ReL}_2}$ is much smaller than $\mathcal{L}_{\text{CE}_1}$, $\mathcal{L}_{\text{CE}_2}$ and $\mathcal{L}_{\text{SNN}}$, so that the loss is scaled by a weighting factor of 50. Because, we experienced RAM memory overflow issues with $M_{\text{SNN,ReL}}^{\text{GAP}}$ and $M_{\text{SNN,ReL}}^{\text{GAVP}}$, the batch size used in training was reduced to 16 and *numSamples* was set to 5×10^5 to have the same number of steps per epoch as before [2]. The results are shown in Table 3.

Table 3: *Comparison of M_{SNN}^{GAVP}, $M_{SNN,ReL}^{GAP}$ and $M_{SNN,ReL}^{GAVP}$ for LOGSPEC input feature on both* Dev *and* Eval *data. Results are reported in % EER. Using a GAVP layer and adding reconstruction loss gives a clear improvement in EER.*

Model	Loss	Dev	Eval
$M_{\text{SNN}}^{\text{GAVP}}$	$\mathcal{L}_{\text{CE}_1}+$ $\mathcal{L}_{\text{CE}_2}+\mathcal{L}_{\text{SNN}}$	0.83	2.01
$M_{\text{SNN,ReL}}^{\text{GAP}}$	$\mathcal{L}_{\text{CE}_1}+\mathcal{L}_{\text{CE}_2}+\mathcal{L}_{\text{SNN}}$ $+50\mathcal{L}_{\text{ReL}_1}+50\mathcal{L}_{\text{ReL}_2}$	**0.66**	2.08
$M_{\text{SNN,ReL}}^{\text{GAVP}}$	$\mathcal{L}_{\text{CE}_1}+\mathcal{L}_{\text{CE}_2}+\mathcal{L}_{\text{SNN}}$ $+50\mathcal{L}_{\text{ReL}_1}+50\mathcal{L}_{\text{ReL}_2}$	0.76	**1.94**

It can be seen that both using the GAVP layer and adding ReL gives a significant performance boost compared to $M_{\text{SNN}}^{\text{GAP}}$. Consequently the best single system performance of 1.94 % EER on the *eval* data set is achieved by $M_{\text{SNN,ReL}}^{\text{GAVP}}$ which outperforms $M_{M_{CE}^{GAP}}$ by a relative margin of 30.5 % EER while having the same number of parameters.

6. Conclusion

We have thoroughly analysed discriminate feature learning in an MTL setting for RA detection and found that SNN significantly outperform the baseline on multiple input features. We explain this improvement by the following. First, SNN greatly enhance the discriminability of the model by *explicitly* increasing the inter-class variance of the model. Second, SNN sample from a very large pool of possible sample pairs. During training, each sampled pair results in a gradient signal to discriminate between utterances of a previously (most likely) unseen pair. This effectively reduces overfitting, thus allowing the model to regularize much better during training.

We then further improve upon SNN by replacing GAP with GAVP and adding ReL. Given our results, we hypothesize that second-order moments of the learned ResNet outputs can contain additional discriminative information which allows for better generalization. ReL forces the model to retain the most characteristic information of every input utterance, which also gives a slight performance boost on the eval set. Our best single sytem $M_{\text{SNN,ReL}}^{\text{GAVP}}$ attains an EER of 1.94% on the eval set, which is an improvement of more than 30% relative in EER to the best single system baseline $M_{\text{CE}}^{\text{GAP}}$. This is achieved without adding a significant amount of parameters, but only by extending the CE training objective by 4 additional training objectives.

[2] More details can be found at `https://www.comet.ml/patrickvonplaten/anti-spoof`.

7. References

[1] Z. Wu, T. Kinnunen, N. Evans, J. Yamagishi, C. Hanilçi, M. Sahidullah, and A. Sizov, "Asvspoof 2015: the first automatic speaker verification spoofing and countermeasures challenge," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[2] T. Kinnunen, M. Sahidullah, H. Delgado, M. Todisco, N. Evans, J. Yamagishi, and K. A. Lee, "The asvspoof 2017 challenge: Assessing the limits of replay spoofing attack detection," *ISCA (the International Speech Communication Association)*, 2017.

[3] M. Todisco, X. Wang, V. Vestman, M. Sahidullah, H. Delgado, A. Nautsch, J. Yamagishi, N. Evans, T. Kinnunen, and K. A. Lee, "Asvspoof 2019: Future horizons in spoofed and fake audio detection," *arXiv preprint arXiv:1904.05441*, 2019.

[4] H. A. Patil and M. R. Kamble, "A survey on replay attack detection for automatic speaker verification (asv) system," in *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE, 2018, pp. 1047–1053.

[5] P. Nagarsheth, E. Khoury, K. Patil, and M. Garland, "Replay attack detection using dnn for channel discrimination." in *Interspeech*, 2017, pp. 97–101.

[6] F. Tom, M. Jain, and P. Dey, "End-to-end audio replay attack detection using deep convolutional networks with attention." in *Interspeech*, 2018, pp. 681–685.

[7] M. Sahidullah, T. Kinnunen, and C. Hanilçi, "A comparison of features for synthetic speech detection," *ISCA (the International Speech Communication Association)*, 2015.

[8] R. M. Hegde, H. A. Murthy, and V. R. R. Gadde, "Significance of the modified group delay feature in speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 1, pp. 190–202, 2006.

[9] C.-I. Lai, N. Chen, J. Villalba, and N. Dehak, "Assert: Anti-spoofing with squeeze-excitation and residual networks," *arXiv preprint arXiv:1904.01120*, 2019.

[10] 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," *arXiv preprint arXiv:1907.02663*, 2019.

[11] A. Gomez-Alanis, A. M. Peinado, J. A. Gonzalez, and A. M. Gomez, "A gated recurrent convolutional neural network for robust spoofing detection," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2019.

[12] G. Lavrentyeva, S. Novoselov, A. Tseren, M. Volkova, A. Gorlanov, and A. Kozlov, "Stc antispoofing systems for the asvspoof2019 challenge," *arXiv preprint arXiv:1904.05576*, 2019.

[13] H. Zeinali, T. Stafylakis, G. Athanasopoulou, J. Rohdin, I. Gkinis, L. Burget, J. Černockỳ *et al.*, "Detecting spoofing attacks using vgg and sincnet: but-omilia submission to asvspoof 2019 challenge," *arXiv preprint arXiv:1907.12908*, 2019.

[14] R. Caruana, "Multitask learning," *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.

[15] J.-w. Jung, H.-j. Shim, H.-S. Heo, and H.-J. Yu, "Replay attack detection with complementary high-resolution information using end-to-end dnn for the asvspoof 2019 challenge," *arXiv preprint arXiv:1904.10134*, 2019.

[16] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a" siamese" time delay neural network," in *Advances in neural information processing systems*, 1994, pp. 737–744.

[17] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *ICML deep learning workshop*, vol. 2, 2015.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[19] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu, "Semantic segmentation with second-order pooling," in *European Conference on Computer Vision*. Springer, 2012, pp. 430–443.

[20] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," in *European conference on computer vision*. Springer, 2016, pp. 499–515.

[21] S. Bell and K. Bala, "Learning visual similarity for product design with convolutional neural networks," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, p. 98, 2015.

[22] F. Zhuang, D. Luo, X. Jin, H. Xiong, P. Luo, and Q. He, "Representation learning via semi-supervised autoencoder for multi-task learning," in *2015 IEEE International Conference on Data Mining*, 2015.

[23] 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," *arXiv preprint arXiv:1904.04589*, 2019.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European conference on computer vision*. Springer, 2016, pp. 630–645.

[25] L. Rosasco, E. D. Vito, A. Caponnetto, M. Piana, and A. Verri, "Are loss functions all the same?" *Neural Computation*, vol. 16, no. 5, pp. 1063–1076, 2004.

[26] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[27] N. Brümmer and E. De Villiers, "The bosaris toolkit: Theory, algorithms and code for surviving the new dcf," *arXiv preprint arXiv:1304.2865*, 2013.

[28] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.