This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3000641, IEEE Access

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Digital Object Identifier

A Kernel Density Estimation Based Loss Function and Its Application to ASV-Spoofing Detection

ALEJANDRO GOMEZ-ALANIS[1], JOSE A. GONZALEZ-LOPEZ[1], AND ANTONIO M. PEINADO.[1], (Senior Member, IEEE)

[1]Department of Signal Processing, Telematics and Communications, University of Granada, Granada, 18071 Spain (e-mail: agomezalanis@ugr.es; joseangl@ugr.es; amp@ugr.es)

Corresponding author: Alejandro Gomez-Alanis (e-mail: agomezalanis@ugr.es).

ABSTRACT Biometric systems are exposed to spoofing attacks which may compromise their security, and voice biometrics, also known as automatic speaker verification (ASV), is no exception. Replay, synthesis and voice conversion attacks cause false acceptances that can be detected by anti-spoofing systems. Recently, deep neural networks (DNNs) which extract embedding vectors have shown superior performance than conventional systems in both ASV and anti-spoofing tasks. In this work, we develop a new concept of loss function for training DNNs which is based on kernel density estimation (KDE) techniques. The proposed loss functions estimate the probability density function (pdf) of every training class in each mini-batch, and compute a log likelihood matrix between the embedding vectors and pdfs of all training classes within the mini-batch in order to obtain the KDE-based loss. To evaluate our proposal for spoofing detection, experiments were carried out on the recent ASVspoof 2019 corpus, including both logical and physical access scenarios. The experimental results show that training a DNN based anti-spoofing system with our proposed loss functions clearly outperforms the performance of the same system being trained with other well-known loss functions. Moreover, the results also show that the proposed loss functions are effective for different types of neural network architectures.

INDEX TERMS Spoofing detection, kernel density estimation, loss function, deep learning, automatic speaker verification.

I. INTRODUCTION

BIOMETRIC authentication [1] aims to authenticate the identity claimed by a given individual based on samples measured from biological processes and/or organs (e.g., voice, fingerprint, face, etc). Voice biometrics, in particular, is an emerging form of biometric authentication with potential advantages given its hands-free, liveliness and dynamic nature. Automatic speaker verification (ASV) [2] is the conventional way to put voice biometrics into practical usage. ASV techniques verify the claimed identity of a given speaker by recording her/his voice, extracting voiceprints from the voice recordings, and deciding whether the speaker is who s/he claims to be based on the extracted voiceprints and a set of pre-stored voiceprints from enrolled users.

However, the vulnerability of ASV systems to malicious attacks is a serious concern nowadays [3]. Our focus in this work is on spoofing detection for ASV, where an impostor could gain fraudulent bypass to the authentication system by presenting speech resembling the voice of a genuine user. Four types of spoofing attacks have been identified [4]: (i) impersonation (i.e., mimicking the voice of a target speaker), (ii) replay (i.e., using pre-recorded voice of a target user), and, also, either (iii) text-to-speech synthesis (TTS) or (iv) voice conversion (VC) systems to generate artificial speech resembling the voice of a legitimate user.

Spoofing detection or presentation attack detection (PAD in ISO/IEC 30107 nomenclature [5]) for ASV has become a hot research topic in recent years as evidenced by the organization of several evaluation campaigns (challenges) in this specific topic: (i) ASVspoof 2015 [6], which focused on logical access (LA) attacks (TTS and VC); (ii) ASVspoof 2017 [7], which focused on physical access (PA) attacks (replay attacks) under noisy environments; and (iii) ASVspoof 2019 [8], which addressed both the detection of LA attacks generated with the latest TTS and VC technologies, and simulated replay attacks under different reverberant acoustic conditions. One of the main conclusions withdrawn from these challenges is that the use of deep neural networks (DNNs) for the extraction of spoofing-aware embedding vectors outperforms other conventional approaches for ASV anti-spoofing [9]–[12].

DNN-based approaches have been widely explored for audio- [13] and video-based anti-spoofing [14], [15]. Three key points are important for building a DNN-based anti-spoofing system with generalization capabilities: (i) architecture, (ii) input features, and (iii) loss function. Multiple types of DNN architectures have been explored, such as feed-forward DNN [16], convolutional neural network (CNN) [16], [17], recurrent neural network (RNN) [17], [18], gated recurrent neural network (GRCNN) [19], light convolutional gated recurrent neural network (LC-GRNN) [9], light convolutional neural network (LCNN) [20], central difference convolutional network (CDCN) [14], etc. Also, a wide range of features have been proposed to train these models, such as spectrogram [12], linear frequency cepstral coefficients (LFCC) [21], constant Q cepstral coefficients (CQCC) [22], raw speech samples [23], local similar pattern (LSP) features [15], signal-to-noise mask (SNM) [19] features, etc. Normally, the architecture of the DNN is adapted to the dimension of the input features, and viceversa. However, the loss function employed to train the DNN is usually independent from the architecture and input features.

Within the DNN-based anti-spoofing framework, several recent studies have focused on designing new loss functions in order to make NNs more suitable for the specific tasks of anti-spoofing [24], ASV [25], [26] and/or their combination [27]. However, these studies do not usually address the following three issues. First, one particular characteristic of anti-spoofing applications, which is shared with ASV systems, is that embeddings extracted by DNNs should enable precise discrimination between *bona fide* speech and spoofed speech and, at the same time, they should be able to generalize well to unknown attacks that are not present in the training dataset. In other words, from a metric learning problem perspective [28]–[30], the goal is to learn a meaningful embedding representation that keeps similar training instances close to each other and the dissimilar instances far away on the embedding space. While specialized loss functions as the triplet network [28] specifically address this issue, conventional losses (e.g., softmax) fall short in achieving this goal. Second, in a supervised scenario, as is the case for DNN-based anti-spoofing detection, metric learning aims to learn a representation which keeps close the embeddings belonging to the same class. To represent each class, different representations have been investigated in the literature, such as representing each class by a centroid in the embedding space [25] or employing an anchor sample to represent the positive class [31]. In these representations, however, the training classes are not fully represented by all the samples in the mini-batch, but by a single embedding representation (i.e., either a centroid or an anchor sample), which may be suboptimal for distance learning. Third, recent loss functions, such as the siamese [24], generalized-end-to-end (GE2E) [25] and triplet loss [31] functions, are based on distance measures between embedding vectors. However, it is not straightforward to select the most appropiate distance measure as well as the embedding normalization technique. Moreover, these loss functions typically require the usage of an extra hyper-parameter called *margin* which is difficult to optimize.

To address all these issues, we propose a new probabilistic loss function for supervised metric learning, where every training class is represented with a probability density function (pdf) which is estimated through kernel density estimation (KDE) [32]–[34] in each mini-batch. The mini-batches are formed so that all training classes are present in the mini-batch and are represented with the same number of samples. Due to the fact that KDE techniques place a probability mass at every sample, we can argue that each class is more accurately represented than in previous approaches, since KDE estimates a pdf per class using all the samples of the mini-batch rather than representing each class with a sole point (centroid or anchor point). Thus, we replace the concept of distance between embeddings by the concept that an embedding belongs to a certain class with a given probability. This has the advantage of avoiding the selection of an appropiate distance measure as well as an embedding normalization technique. Although the experiments supporting these aforementioned advantages of the proposed loss functions are focused on ASV anti-spoofing, they could be applied to different classification tasks.

This paper is organized as follows. Section II outlines the most popular loss functions used to train DNNs for developing ASV and anti-spoofing systems. Then, in Section III, we describe the proposed loss functions based on KDE. Section IV describes the speech corpora, neural networks and loss functions which are then evaluated in Section V for spoofing detection. Finally, we summarize the conclusions derived from this research in Section VI.

II. RELATED WORK

This section describes several loss functions that can be used in the context of distance metric learning in order to learn a meaningful embedding representation for the data samples assuming that the target labels are available a priori (i.e., supervised scenario). Some of these functions have already been successfully applied to either ASV or anti-spoofing.

In this section we use the following notation: $\boldsymbol{e}_{ji}$ denotes the embedding (output of a hidden layer of the DNN) of the i-th utterance of the class j, M is the number of utterances per class in the mini-batch, and N is the number of classes of the training set. In addition, we consider that every mini-batch is composed of $N \times M$ utterances. In anti-spoofing, the number of classes N is usually the number of training spoofing attacks plus the genuine class.

A. CROSS ENTROPY LOSS FUNCTION

The cross entropy loss, also known as softmax loss, is widely used to train DNNs for classification tasks. Typically, when the softmax loss function is used in ASV and anti-spoofing systems, embeddings are extracted from a middle or the last hidden layer of the DNN. Assuming this latter case, where embeddings are extracted from the last hidden layer of the DNN, the softmax loss function can be expressed as,

$$\mathcal{L}_{\text{softmax}} = \sum_{j=1}^{N}\sum_{i=1}^{M} -\log \frac{\exp(\boldsymbol{w}_j^T \boldsymbol{e}_{ji} + b_j)}{\sum_{k=1}^{N} \exp(\boldsymbol{w}_k^T \boldsymbol{e}_{ji} + b_k)}, \quad (1)$$

where $\boldsymbol{w} = [\boldsymbol{w}_1, ..., \boldsymbol{w}_N]$ and $\boldsymbol{b} = [b_1, ..., b_N]$ are the weight matrix and bias vector of the output layer, respectively.

B. ADDITIVE MARGIN LOSS FUNCTION

The additive margin (AM) softmax loss function [35] was proposed to replace the inner product operation of the softmax loss function in Eq. (1) with the cosine similarity operation in order to widen the inter-class margin in the embedding space [36]. The AM softmax loss function can be expressed as,

$$\mathcal{L}_{\text{AM}} = \sum_{j=1}^{N}\sum_{i=1}^{M} -\log \frac{\exp(s \cdot (cos(\boldsymbol{w}_j, \boldsymbol{e}_{ji}) - m))}{\exp(s \cdot (cos(\boldsymbol{w}_j, \boldsymbol{e}_{ji}) - m)) + r_{ji}}, \quad (2)$$

$$r_{ji} = \sum_{\substack{k=1 \\ k \neq i}}^{N} \exp(s \cdot cos(\boldsymbol{w}_k, \boldsymbol{e}_{ji})), \quad (3)$$

where m is an additional margin and s is a scaling factor for stabilizing training. This loss function is a generalized version of the angular softmax loss [35]. Recently, this type of loss function has been successfully applied to anti-spoofing [37] and speaker verification systems [38], [39].

C. GENERALIZED END-TO-END LOSS FUNCTION

In the generalized end-to-end (GE2E) loss, which was originally proposed for ASV, each class (speaker) is represented by a centroid obtained averaging all the embeddings belonging to that class in the mini-batch. From those centroids, two loss functions were proposed in [25] which seek for minimizing the distance between the embeddings and their corresponding class centroids, while also maximizing the distance with the centroids from the other speakers. In anti-spoofing, the speakers are replaced by attacks. The distance between the embedding of the i-th utterance of the j-th attack ($\boldsymbol{e}_{ji}$) and the centroid of the k-th attack ($\hat{\boldsymbol{c}}_k$), is computed as:

$$\boldsymbol{S}_{ji,k} = \omega \cdot cos(\boldsymbol{e}_{ji}, \hat{\boldsymbol{c}}_k) + b, \quad (4)$$

where ω and b are learnable parameters for score scaling and shifting, $\boldsymbol{S}$ is the similarity matrix, and the centroid embedding is computed by averaging the embeddings of each attack:

$$\hat{\boldsymbol{c}}_k = \frac{1}{M}\sum_{i=1}^{M} \boldsymbol{e}_{ki}. \quad (5)$$

The GE2E loss function consists of two losses which are computed using the values of the similarity matrix $\boldsymbol{S}$: (i) softmax loss, and (ii) contrast loss. The softmax loss of the embedding $\boldsymbol{e}_{ji}$ is expressed as follows,

$$\mathcal{L}_{\text{GE2E-softmax}}(\boldsymbol{e}_{ji}) = -\boldsymbol{S}_{ji,j} + \log \sum_{k=1}^{N} \exp(\boldsymbol{S}_{ji,k}). \quad (6)$$

Likewise, the contrast loss of the embedding $\boldsymbol{e}_{ji}$ is computed as,

$$\mathcal{L}_{\text{GE2E-contrast}}(\boldsymbol{e}_{ji}) = 1 - \sigma(\boldsymbol{S}_{ji,j}) + \max_{\substack{1 \leq k \leq N \\ k \neq j}} \sigma(\boldsymbol{S}_{ji,k}), \quad (7)$$

where $\sigma(x)$ is the sigmoid function. This contrast loss function deserves some comments. For every utterance, exactly two components are added to the loss: (i) a positive component, which is associated with a positive match between the embedding $\boldsymbol{e}_{ji}$ and its true class centroid $\hat{\boldsymbol{c}}_j$; and (ii) a negative component, which is associated with a negative match between the embedding $\boldsymbol{e}_{ji}$ and the centroid $\hat{\boldsymbol{c}}_k$ with the highest similarity among all false class centroids.

Combining equations (6) and (7), the final GE2E loss function is the sum of the two losses over the similarity matrix:

$$\mathcal{L}_{\text{GE2E}} = \sum_{j=1}^{N}\sum_{i=1}^{M} \Big[\mathcal{L}_{\text{GE2E-softmax}}(\boldsymbol{e}_{ji}) + \mathcal{L}_{\text{GE2E-contrast}}(\boldsymbol{e}_{ji})\Big]. \quad (8)$$

D. SIAMESE LOSS FUNCTION

The siamese architecture processes two utterances at once using the same neural network, obtains two embeddings $\boldsymbol{e}_{ji}$ and $\boldsymbol{e}_{k\sim}$, and computes a loss based on the embedding distance:

$$\mathcal{L}_{\text{siamese}} = \sum_{j=1}^{N}\sum_{i=1}^{M} \delta_{jk} \cdot D(\boldsymbol{e}_{ji}, \boldsymbol{e}_{k\sim}) + (1 - \delta_{jk}) \cdot \max(m, D(\boldsymbol{e}_{ji}, \boldsymbol{e}_{k\sim})), \quad (9)$$

where $\mathbf{e}_{k\sim}$ denotes any embedding of the class k, $\delta_{jk} \in \{0, 1\}$ is a label which indicates whether the embeddings $\mathbf{e}_{ji}$ and $\mathbf{e}_{k\sim}$ belong to the same class (i.e., when $k = j$), $D(\mathbf{e}_{ji}, \mathbf{e}_{k\sim})$ is any distance measure between $\mathbf{e}_{ji}$ and $\mathbf{e}_{k\sim}$, and m is a hyper-parameter distance margin. There are many siamese network variants reported in the literature for different applications, such as face recognition [40], person identification [41], image recognition [42], etc.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3000641, IEEE Access

E. TRIPLET LOSS FUNCTION

The triplet network [28] is a neural network architecture which attempts to learn an embedding representation of a multi-class labeled dataset which favours a small distance between example pairs labeled as similar, and large distances for pairs labeled as dissimilar. However, unlike the siamese networks, this architecture works with triplets of embeddings. In particular, it defines a loss function which ensures that an *anchor* embedding ($\boldsymbol{e}_{ji}$) of class j is closer to other *positive* samples ($\boldsymbol{e}_{jp}$, $p \neq i$) than to any *negative* sample ($\boldsymbol{e}_{n\sim}, n \neq j$) [31]. Thus, if we consider a batch size of $N \times M$ utterances, the triplet loss which is minimized is:

$$\mathcal{L}_{\text{triplet}} = \sum_{j=1}^{N}\sum_{i=1}^{M} \max\left[\|\boldsymbol{e}_{ji}-\boldsymbol{e}_{jp}\|_2^2 - \|\boldsymbol{e}_{ji}-\boldsymbol{e}_{n\sim}\|_2^2 + \alpha, 0\right], \tag{10}$$

where α is a margin which is enforced between the positive and negative distances. Thus, given an anchor embedding $\boldsymbol{e}_{ji}$, its corresponding triplet $(\boldsymbol{e}_{ji}, \boldsymbol{e}_{jp}, \boldsymbol{e}_{n\sim})$ will be built with a *hard positive* embedding $\boldsymbol{e}_{jp}$ and a *hard negative* embedding $\boldsymbol{e}_{n\sim}$ such that indices p and n are selected according to the following criteria: $p = argmax_{r \neq i}\ \|\boldsymbol{e}_{ji} - \boldsymbol{e}_{jr}\|_2^2$, and $n = argmin_{s \neq j}\ \|\boldsymbol{e}_{ji} - \boldsymbol{e}_{s\sim}\|_2^2$.

Recently, the triplet loss function has been successfully applied to train face verification systems [31], ASV systems [43], [44], and joint ASV and PAD systems [27].

III. KERNEL DENSITY ESTIMATION LOSS FUNCTION

In this section we describe the proposed loss functions based on KDE for training DNN-based embedding extraction systems. Section III-A describes the computation of the log likelihood matrix employed by all the proposed losses. After that, the proposed KDE-based loss functions are described in Section III-B.

A. KDE-BASED LOG LIKELIHOOD MATRIX

Similarly to the GE2E loss method described in Section II-C, every mini-batch consists of $N \times M$ utterances from the N different training classes (genuine class and $N-1$ spoofing attacks), and each class is represented with M utterances. Thus, each utterance i $(1 \leq i \leq M)$ from the training class j $(1 \leq j \leq N)$, represented by its sequence of feature vectors $\boldsymbol{X}_{ji}$, is fed into a neural network in order to obtain the embedding vector $\boldsymbol{e}_{ji} = g(\boldsymbol{X}_{ji}; \boldsymbol{\Theta})$, where $\boldsymbol{\Theta}$ represents all the parameters of the neural network.

Let the embedding vectors from the k-th training class $\boldsymbol{e}_{k1}, ..., \boldsymbol{e}_{kM} \in \mathcal{R}^q$ be independent and identically distributed random samples from an unknown distribution $f_k(\boldsymbol{e})$. The estimation of its multivariate pdf using KDE [32], [45] is given by,

$$\begin{aligned}\hat{f}_k(\boldsymbol{e}) &= \frac{1}{M}\sum_{m=1}^{M}\frac{1}{\det(\boldsymbol{H}_k)}K\left(\boldsymbol{H}_k^{-1}(\boldsymbol{e}-\boldsymbol{e}_{km})\right)\\ &= \frac{1}{M}\sum_{m=1}^{M}K_{\boldsymbol{H}_k}(\boldsymbol{e}-\boldsymbol{e}_{km}),\end{aligned} \tag{11}$$

where $K(\cdot)$ is the kernel function, $\boldsymbol{H}_k$ is a nonsigular and symmetric bandwidth matrix [46], [47], and $K_{\boldsymbol{H}_k}(\boldsymbol{u}) = K(\boldsymbol{H}_k^{-1}\boldsymbol{u})/\det(\boldsymbol{H}_k)$. A range of kernel functions are commonly used, such as uniform, triangular, Gaussian and Epanechnikov [48]. For instance, the probability density function with a Gaussian kernel (that is, $K_{\boldsymbol{H}_k}(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{u}; \boldsymbol{0}, \boldsymbol{\Sigma}_k)$) can be computed as,

$$\hat{f}_k(\boldsymbol{e}) = \frac{1}{M}\sum_{m=1}^{M}\mathcal{N}(\boldsymbol{e}; \boldsymbol{\mu}_k = \boldsymbol{e}_{km}, \boldsymbol{\Sigma}_k = \sigma_k^2 \cdot \boldsymbol{I}), \tag{12}$$

where $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ are the mean vector and covariance matrix of the Gaussian distribution $\mathcal{N}(\cdot)$, $\boldsymbol{I}$ is the identity matrix, and σ_k^2 represents the bandwidth of the KDE model for class k. Every class has its corresponding bandwidth, which is a learnable parameter that is constrained to be positive ($\sigma_k^2 > 0$). In this way, the kernel density estimator $\hat{f}_k(\boldsymbol{e})$ places a probability mass at each observation embedding $\boldsymbol{e}_{km}$ according to a Gaussian probability model.

Once all the probability density functions of the considered mini-batch have been built, they are evaluated for every embedding belonging to that mini-batch. That is, all possible $\hat{f}_k(\boldsymbol{e}_{ji})$ $(k, j = 1, ...N; i = 1, ..., M)$ are computed. Then, these probabilities are arranged in the following log-likelihood matrix:

$$\mathbf{L}_{ji,k} = \begin{cases} \log\left(\frac{1}{M}\sum\limits_{m=1}^{M} K_{\boldsymbol{H}_k}\left(\boldsymbol{e}_{ji}-\boldsymbol{e}_{km}\right)\right) & k \neq j \\ \log\left(\frac{1}{M-1}\sum\limits_{\substack{m=1\\ m\neq i}}^{M} K_{\boldsymbol{H}_k}\left(\boldsymbol{e}_{ji}-\boldsymbol{e}_{km}\right)\right) & k = j \end{cases}. \tag{13}$$

To avoid trivial solutions and make training stable, the embedding vector $\boldsymbol{e}_{ji}$ is removed when estimating the density function of the true class (i.e., when $k = j$ in Eq. (13)). Fig. 1 illustrates the whole process for obtaining the log likelihood matrix with input features, embedding vectors and likelihoods from different training classes (genuine and spoofing attacks), represented by different colors.

From the log likelihood matrix $\mathbf{L}_{ji,k}$ in Eq. (13), we strive to achieve two goals simultaneously during the DNN training. First, we aim at maximizing the probability of each embedding vector $\mathbf{e}_{ji}$ belonging to its class j, that is,

$$\underset{\boldsymbol{\Theta}}{\text{maximize}} \quad \mathbf{L}_{ji,j} = \log \hat{f}_j(\boldsymbol{e}_{ji}), \tag{14}$$

where $\boldsymbol{\Theta}$ are the neural network parameters to be optimized in the training stage. At the same time, the probability of

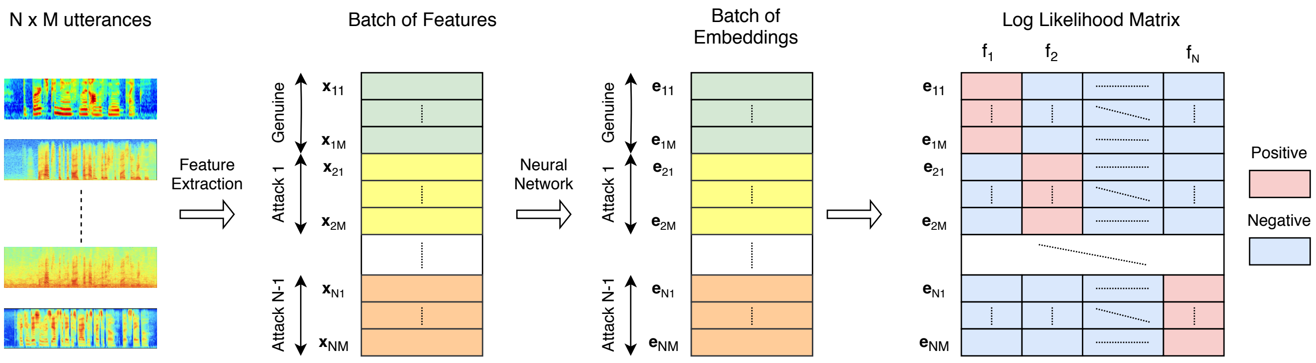

FIGURE 1. System overview for computing the log likelihood matrix of a mini-batch of $N \times M$ utterances.

each embedding vector $\boldsymbol{e}_{ji}$ belonging to the rest of the classes should be minimized:

$$\underset{\boldsymbol{\Theta}}{\text{minimize}} \quad \mathbf{L}_{ji,k} = \log \hat{f}_k(\boldsymbol{e}_{ji}) \; (k \neq j). \tag{15}$$

In other words, as depicted in Fig. 1, we strive to find the optimum set of weights $\boldsymbol{\Theta}$ that results in large log likelihood values for red cells in the figure and small values for the blue cells in the figure. We achieve these two simultaneous goals by means of three alternative loss functions, as described in the next section.

B. KDE-BASED LOSS FUNCTIONS

There are several ways to implement the requirements described above. In this section, we describe three alternative losses to achieve our goal during the training of the neural network: softmax, contrast and triplet KDE based losses.

1) KDE-Softmax Loss

As described in Section II-A, the softmax function is typically used in tandem with the negative log-likelihood (NLL), such that: $L(\mathbf{y}) = -\log(\text{softmax}(\mathbf{y}))$. The output of the softmax function can be interpreted as the probabilities that a certain set of features belong to a certain class, which is combined with the NLL in order to build the popular cross-entropy or softmax loss.

The softmax loss can be directly applied to KDE using the log likelihood matrix, such that:

$$\mathcal{L}_{\text{KDE-softmax}} = \sum_{j=1}^{N}\sum_{i=1}^{M}\left[-\mathbf{L}_{ji,j} + \log\sum_{k=1}^{N}\exp\big(\mathbf{L}_{ji,k}\big)\right]. \tag{16}$$

This loss function tries to increase the probability of each embedding belonging to its true class, while minimizing the probability of the embedding belonging to the rest of the classes.

2) KDE-Contrast Loss

The contrast loss is formed by two terms: (i) a positive term, which is the probability of the embedding $\boldsymbol{e}_{ji}$ belonging to the true class; and (ii) a hard negative term, which is the highest probability of that embedding belonging to any of the negative classes, that is,

$$\mathcal{L}_{\text{KDE-contrast}} = \sum_{j=1}^{N}\sum_{i=1}^{M}\max\left[\Big(-\mathbf{L}_{ji,j} + \max_{\substack{1\leq k\leq N\\ k\neq j}} \mathbf{L}_{ji,k}\Big), 0\right]. \tag{17}$$

3) KDE-Triplet Loss

In the following we describe the adaptation of the triplet loss to our KDE-based framework. Similarly to the triplet loss, we want to find an embedding representation that, for a given anchor embedding $\boldsymbol{e}_{ji}$, the probability of such embedding to the positive class j is large, whereas the probability of a negative exemplar of belonging to the same class is small. While this loss is motivated in [31] in the context of nearest-neighbour classification [49], here the quadratic distances are replaced by log likelihoods.

This loss tries to ensure that an embedding vector $\boldsymbol{e}_{ji}$ (*anchor*) of a specific class j (*positive class*) obtains a higher probability of belonging to that class than any other embedding vector $\boldsymbol{e}_{n\sim}$ (*negative*) from other class ($n \neq j$). In this way, the triplet is formed by: (i) an *anchor embedding* $\boldsymbol{e}_{ji}$, (ii) a *negative embedding* $\boldsymbol{e}_{n\sim}$, and (iii) a *positive* estimated density function $\hat{f}_j$.

Thus, this loss tries to ensure

$$\hat{f}_j(\boldsymbol{e}_{n\sim}) + \alpha < \hat{f}_j(\boldsymbol{e}_{ji}), \tag{18}$$

where α is a margin that is enforced between the true and false positive probabilities. Using the log likelihood matrix of Eq. (13), the loss which is minimized is

$$\mathcal{L}_{\text{KDE-triplet}} = \sum_{j=1}^{N}\sum_{i=1}^{M}\max\left[\mathbf{L}_{n\sim,j} - \mathbf{L}_{ji,j} + \alpha, 0\right], \tag{19}$$

where α is a hyper-parameter margin which is enforced between the positive and negative likelihoods.

Generating all possible triplets would result in many of them being easily satisfied (i.e., fulfill constraint (18)). Thus, not all of them would contribute to the training, which might result in a slower convergence. Therefore, it is crucial to select hard triplets which do not fulfill constraint (18), and can therefore contribute to improving the model. As suggested in [31], instead of picking the hardest positives, we use all anchor-positive pairs within the mini-batch. In addition, [31] shows that selecting the hardest negatives can in practice lead to a bad local minima in training. In order to mitigate this, we select *semi-hard* negative exemplars which lie inside the margin α [31]:

$$\hat{f}_j(\boldsymbol{e}_{n\sim}) < \hat{f}_j(\boldsymbol{e}_{ji}) < \hat{f}_j(\boldsymbol{e}_{n\sim}) + \alpha. \qquad (20)$$

4) Analysis and relation with other loss functions

From equations (16), (17) and (19), we can observe that the KDE softmax, contrast and triplet loss functions have in common the term $-\mathbf{L}_{ji,j}$, which aims at maximizing the probability of the embedding $\boldsymbol{e}_{ji}$ belonging to the estimated density function of the true class. The difference between these three loss functions lies in the penalization term, which tries to separate the positive class j from the rest of training classes. Specifically, the penalization term of these loss functions is:

- KDE-softmax loss: the sum of the likelihoods that the embedding vector $\boldsymbol{e}_{ji}$ belongs to all training classes.
- KDE-contrast loss: the highest log likelihood between the embedding vector $\boldsymbol{e}_{ji}$ and any negative class.
- KDE-triplet loss: the log likelihood that a negative embedding vector $\boldsymbol{e}_{n\sim}$ belongs to the anchor class j, plus a margin α.

If we combine the KDE softmax and contrast loss functions (Eqs. (16) and (17)), we can derive a probabilistic version of the GE2E loss described in Section II-C, which we call it as full kernel density estimation (FKDE) loss, that is,

$$\mathcal{L}_{\text{FKDE}} = \sum_{j=1}^{N}\sum_{i=1}^{M}\left[\mathcal{L}_{\text{KDE-softmax}}(\boldsymbol{e}_{ji}) + \mathcal{L}_{\text{KDE-contrast}}(\boldsymbol{e}_{ji})\right]. \qquad (21)$$

However, while the G2E2 technique computes a cosine similarity matrix, our proposed FKDE loss computes a log likelihood matrix. Furthermore, the GE2E technique represents each class by means of a centroid, while our technique estimates a pdf for each class. From a clustering point of view, we argue that the latter is a superior and more informative representation.

On the other hand, the KDE-triplet loss function in (19) can be shown to be a generalization of the classical triplet loss in (10) when KDE with Gaussian kernel (GKDE) and diagonal covariance matrix is employed. In fact, if we only consider an embedding $\boldsymbol{e}_{ji}$ for estimating the probability density function in (12), and we introduce a positive index p such that $1 \leq p \leq M, p \neq i$, the GKDE triplet loss in (19) would become:

$$\mathcal{L} = \sum_{j=1}^{N}\sum_{i=1}^{M}\max\left[\mathbf{L}_{n\sim,j(i)} - \mathbf{L}_{jp,j(i)} + \alpha, 0\right] =$$

$$\sum_{j=1}^{N}\sum_{i=1}^{M}\max\left[\log\hat{f}_j^{(i)}(\boldsymbol{e}_{n\sim}) - \log\hat{f}_j^{(i)}(\boldsymbol{e}_{jp}) + \alpha, 0\right], \qquad (22)$$

where,

$$\log\hat{f}_j^{(i)}(\boldsymbol{e}) = \log\frac{\exp\left(-\frac{1}{2}(\boldsymbol{e}-\boldsymbol{e}_{ji})^T\boldsymbol{\Sigma}_j^{-1}(\boldsymbol{e}-\boldsymbol{e}_{ji})\right)}{(2\pi)^{q/2}|\boldsymbol{\Sigma}_j|^2}, \qquad (23)$$

and q is the embedding size. Since we consider a diagonal covariance matrix $\boldsymbol{\Sigma}_j = \sigma_j^2 \cdot \boldsymbol{I}$, this log probability density function can be simplified to:

$$\log\hat{f}_j^{(i)}(\boldsymbol{e}) = -\frac{q}{2}\log(2\pi\sigma_j^2) - \frac{1}{2\sigma_j^2}\left\|\boldsymbol{e}-\boldsymbol{e}_{ji}\right\|_2^2. \qquad (24)$$

Finally, if we consider a constant bandwidth for the GKDE $\sigma_j^2 = 1$, and substitute (24) into (22), the modified version of the proposed GKDE triplet loss equals the classical triplet loss function:

$$\mathcal{L} = \sum_{j=1}^{N}\sum_{i=1}^{M}\max\left[\|\boldsymbol{e}_{ji}-\boldsymbol{e}_{jp}\|_2^2 - \|\boldsymbol{e}_{ji}-\boldsymbol{e}_{n\sim}\|_2^2 + \alpha, 0\right]. \qquad (25)$$

To sum up, the combination of the KDE softmax and contrast loss functions results in a probabilistic version of the GE2E loss. In addition, the GKDE triplet loss is a generalized version of the classical triplet loss.

IV. EXPERIMENTAL SETUP

This section is organized as follows. First, the speech corpora which was employed for the evaluation of the proposed techniques is described. Then, Section IV-B outlines the system configuration and network training. After that, Section IV-C provides the implementation details of the the loss functions that are evaluated, including our proposals and other well-known losses from the literature. Finally, the performance metrics employed to evaluate the performance of the anti-spoofing system are discussed.

A. SPEECH CORPORA

We conducted experiments on the recent ASVspoof 2019 database [8] which encompasses two partitions for the assessment of logical and physical access scenarios. A summary of their composition in terms of speakers and number of utterances is presented in Table 1.

TABLE 1. Structure of the ASVspoof2019 data corpus divided by the training, development and evaluation sets [8].

	#speakers		#utterances			
			Logical Access		Physical Access	
Subset	Male	Female	Bona fide	Spoof	Bona fide	Spoof
Training	8	12	2,580	22,800	5,400	22,800
Development	4	6	2,548	22,296	5,400	24,300
Evaluation	21	27	7,355	63,882	18,090	116,640

TABLE 2. LCNN architecture used in the experiments. MFM stands for Max Feature Map activation. FC stands for Fully Connected layer. "q" denotes the dimension of the embedding vectors extracted by the LCNN.

Type	Filter / Stride	Output size	# Parameters
Conv.	$5 \times 5 / 1 \times 1$	$256 \times 400 \times 16$	416
MFM	-	$256 \times 400 \times 8$	-
MaxPool	$2 \times 2 / 2 \times 2$	$128 \times 200 \times 8$	-
Batch Norm.	-	$128 \times 200 \times 8$	-
Conv.	$1 \times 1 / 1 \times 1$	$128 \times 200 \times 16$	144
MFM	-	$128 \times 200 \times 8$	-
Batch norm.	-	$128 \times 200 \times 8$	-
Conv.	$3 \times 3 / 1 \times 1$	$128 \times 200 \times 32$	2336
MFM	-	$128 \times 200 \times 16$	-
MaxPool	$2 \times 2 / 2 \times 2$	$64 \times 100 \times 16$	-
Batch norm.	-	$64 \times 100 \times 16$	-
Conv.	$1 \times 1 / 1 \times 1$	$64 \times 100 \times 32$	544
MFM	-	$64 \times 100 \times 16$	-
Batch norm.	-	$64 \times 100 \times 16$	-
Conv.	$3 \times 3 / 1 \times 1$	$64 \times 100 \times 32$	4640
MFM	-	$64 \times 100 \times 16$	-
MaxPool	$2 \times 2 / 2 \times 2$	$32 \times 50 \times 16$	-
Batch norm.	-	$32 \times 50 \times 16$	-
Conv.	$1 \times 1 / 1 \times 1$	$32 \times 50 \times 32$	544
MFM	-	$32 \times 50 \times 16$	-
Batch norm.	-	$32 \times 50 \times 16$	-
Conv.	$3 \times 3 / 1 \times 1$	$32 \times 50 \times 32$	4640
MFM	-	$32 \times 50 \times 16$	-
MaxPool	$2 \times 2 / 2 \times 2$	$16 \times 25 \times 16$	-
FC	-	$2 \times q$	$12800 \times q$
MFM	-	q	-

1) ASVspoof 2019 Logical Access Corpus

The LA database contains *bona fide* speech and spoofed speech data generated using 17 TTS and VC systems. Six of these systems are designated as known attacks, with the other 11 being designated as unknown attacks. The training and development sets only contain known attacks, whereas the evaluation set contains 2 known and 11 unknown spoofing attacks. Among the 6 known attacks there are 2 VC systems and 4 TTS systems. VC systems use a neural-network-based and spectral-filtering-based approaches [50]. TTS systems use either waveform concatenation or neural-network-based speech synthesis using a conventional source-filter vocoder [51] or a WaveNet based vocoder [52]. The 11 unknown systems comprise 2 VC, 6 TTS and 3 hybrid TTS-VC systems and were implemented with various waveform generation methods including classical vocoding, GriffinLim [53], generative adversarial networks [54], neural waveform models [55], waveform concatenation, waveform filtering [56], spectral filtering, and their combination.

2) ASVspoof 2019 Physical Access Corpus

The PA database contains *bona fide* speech and spoofed speech data generated according to a simulation of their presentation to the microphone of an ASV system within a reverberant acoustic environment. Training and development data is created by simulating 27 different acoustic and 9 different replay configurations. Acoustic configurations comprise an exhaustive combination of 3 categories of room sizes, 3 categories of reverberation and 3 categories of speaker-to-ASV microphone distances. Replay configurations comprise 3 categories of attacker-to-talker recording distances, and 3 categories of loudspeaker quality. Evaluation data is generated in the same manner as training and development data, albeit with different, random acoustic and replay configurations. Thus, the set of room sizes, levels of reverberation, speaker-to-ASV microphone distances, attacker-to-talker recording distances and loudspeaker qualities, are different from those of training and development.

B. SYSTEM DESCRIPTION

This section provides a detailed description of the implemented systems:

1) Spectral Analysis

Speech signals were analyzed using a Blackman analysis window of 25 ms length with 10 ms of frame shift. Log magnitude spectrogram features (STFT) with 256 frequency bins were obtained to feed the neural network. No normalization was applied to the input features.

We considered two techniques for obtaining an unified time-frequency (T-F) shape of features. First, we truncated the spectrum along the time axis with a fixed size of $T = 400$ frames in order to feed a convolutional neural network (CNN). During this procedure, short utterances were extended by repeating their contents if necessary to match the required length. Second, we used a sliding window approach of $W = 32$ frames with a shift of $\delta = 12$ frames in order to feed a RNN.

2) Light Convolutional Neural Network

A simplified version of the recently proposed Light Convolutional Neural Network (LCNN) [37] was employed in most of our experiments, which is an architecture that has demonstrated to be very effective to detect spoofed speech in the last two ASVspoof challenges [20], [37]. It was the best system of the ASVspoof 2017 challenge [20], and the best single system in the LA scenario of the ASVspoof 2019 challenge [37].

Table 2 details the architecture of the LCNN used in our experiments. In this model we truncated the spectrum of the utterances to a fixed size of $T = 400$ frames. As can be seen, the specific charasteristic of the LCNN architecture [10] is

TABLE 3. LC-GRNN architecture used in the experiments. MFM stands for Max Feature Map activation. FC stands for Fully Connected layer. "q" denotes the dimension of the embedding vectors extracted by the LC-GRNN.

RNN	Type	Filter / Stride	Output	# Parameters
Layer 1	Conv.	5×5 / 1×1	$256 \times 32 \times 16$	2496
	MFM	-	$256 \times 32 \times 8$	-
	MaxPool	2×1 / 2×1	$128 \times 32 \times 8$	-
	Batch Norm.	-	$128 \times 32 \times 8$	-
Layer 2	Conv.	1×1 / 1×1	$128 \times 32 \times 16$	864
	MFM	-	$128 \times 32 \times 8$	-
	Conv.	3×3 / 1×1	$128 \times 32 \times 32$	7008
	MFM	-	$128 \times 32 \times 16$	-
	MaxPool	2×1 / 2×1	$64 \times 32 \times 16$	-
	Batch Norm.	-	$64 \times 32 \times 16$	-
Layer 3	Conv.	1×1 / 1×1	$64 \times 32 \times 32$	3264
	MFM	-	$64 \times 32 \times 16$	-
	Conv.	3×3 / 1×1	$64 \times 32 \times 16$	6690
	MFM	-	$64 \times 32 \times 8$	-
	MaxPool	2×1 / 2×1	$32 \times 32 \times 8$	-
	Batch Norm.	-	$32 \times 32 \times 8$	-
-	FC	-	$2 \times q$	$16384 \times q$
	MFM	-	q	-

the usage of the Max-Feature-Map activation (MFM) which is based on the Maxout activation function [57]. Thus, the LCNN is composed of 7 convolutional layers with MFM activation, 4 max-pooling layers with kernel of size 2×2 and stride of size 2×2 in order to reduce both time and frequency dimension, 6 batch normalization layers in order to increase the stability and convergence speed during the training process, and one fully connected layer with MFM activation where the embedding vectors are extracted.

3) Light Convolutional Gated Recurrent Neural Network

We also used the Light Convolutional Gated Recurrent Neural Network (LC-GRNN) that we proposed in our previous works [9], [19]. It was one of the ten top performing single systems of the ASVspoof 2019 challenge [8]. This archictecture, in contrast to the LCNN described above, is based on a RNN, thus, having the potential advantage that there is no need to truncate the utterance to extract the embeddings.

Table 3 shows a summary of the LC-GRNN architecture. It processes context windows of $W = 32$ frames with a shift of $\delta = 12$ frames. It consists of 3 recurrent layers, where each one has different light convolutional layers followed by a max-pooling operation which reduces the frequency dimension. Also, batch normalization is applied in order to increase the stability and convergence speed of the training process. Once all the frame-level context windows are processed by the convolutional and recurrent layers, 8 feature maps of size 32×32 are flattened to make up a feature vector of 8192 components. Then, this vector is fed to a fully connected layer with MFM activation to obtain the embedding vector of the utterance.

4) Training setup

The neural networks were trained using the Adam optimizer [58] with a learning rate of $3 \cdot 10^{-4}$. Also, early stopping was applied when no improvement of the loss on the validation set was obtained after five epochs. To prevent the problem of overfitting, a 60% dropout was applied in the fully connected layer of the two models. All the specified hyperparameters of the systems were optimized using the validation set of the data corpora. The Pytorch toolkit [59] was employed to implement the deep learning framework.

5) Final classifier

The embeddings extracted from the utterances were finally processed by a classifier, which produces a score per utterance, indicating whether the utterance is genuine or spoofed. Based on the results from our previous works [9], [19], we used a probabilistic linear discriminant analysis (PLDA). We also applied a posterior normalization of the scores. Provided the prior of the different classes is uniform, the normalized score of the embedding vector $\boldsymbol{e}$ is

$$p(\text{genuine}|\boldsymbol{e}) = \log \frac{\exp(p(\boldsymbol{e}|\text{genuine}))}{\sum_{j=1}^{N} \exp(p(\boldsymbol{e}|j))}, \qquad (26)$$

where $p(\boldsymbol{e}|j)$ is the log posterior predictive probability of the embedding vector $\boldsymbol{e}$ given class j ($j = 1, ..., N$).

C. LOSS FUNCTIONS

This section details the usage and hyper-parameters of the different loss functions employed to train the LCNN and LC-GRNN models. We used $N = 7$ and $N = 2$ training classes in the LA and PA scenarios, respectively. In the LA scenario, we used the 6 known spoofing attacks and the genuine class. In the PA scenario, we only used 2 classes: genuine and spoofed speech.

1) Cross entropy or softmax loss

This loss processes the embedding vectors with an additional fully connected layer with softmax activation of N neurons to discriminate between the genuine and the $N - 1$ spoofing classes of the training set. After that, it applies the NLL to build the cross-entropy or softmax loss.

2) Additive margin loss

In our preliminary experiments we evaluated the *cosface* [60], *arcface* [61] and *sphereface* [35] versions of the additive margin loss. The difference between them lies in the additional margin $m = 30°, 64°, 64°$ and the scaling factor $s = 0.4, 0.5, 1.35$, respectively. The best performance in the preliminary experiments was obtained with the *cosface* version, so that we evaluated it in the rest of the experiments as angular softmax loss.

3) Generalized end-to-end (GE2E) loss

The number of training classes (N) is equal to the number of spoofing attacks of the training set plus the genuine class. We evaluated two versions of the GE2E loss: (i) GE2E with only the softmax loss, and (ii) GE2E with the softmax and contrast losses together, as it is indicated in Eq. (8).

4) Siamese loss

We evaluated a siamese variant called siamese-classification hybrid architecture [62], which has been successfully applied for replay spoofing detection [24]. This siamese network was trained by outputting a softmax layer over the two targets: *similar* and *dissimilar* input pairs. Thus, the network was trained to identify genuine-genuine or spoof-spoof speech as *similar* input pairs, and genuine-spoof pairs as *dissimilar* inputs.

5) Triplet loss

We evaluated the triplet loss using all anchor-positive pairs of the mini-batch and selecting the *semi-hard* negative utterances which lie inside the margin $\alpha = 1.0$, as shown in Eq. (10).

6) KDE-based loss functions

We computed the log likelihood matrix $\mathbf{L}_{ji,k}$ for every mini-batch and evaluated three KDE-based loss functions: (i) KDE softmax from Eq. (16), (ii) combination of KDE softmax and contrast from Eq. (21), and (iii) KDE triplet from Eq. (19). We evaluated them using different types of kernel functions, as it is discussed in Section V-A1. In the KDE triplet loss, we used all anchor-positive pairs of the mini-batch and selected the *semi-hard* negative utterances which lie inside the margin $\alpha = 1.0$.

D. PERFORMANCE METRICS

The evaluation of the anti-spoofing system is done in terms of the pooled equal error rate (EER) across all attacks, and the minimum normalized tandem detection cost function (min-tDCF) [63] for both the LA and PA scenarios, separately.

V. EXPERIMENTAL RESULTS

This section presents the results from the evaluation on the ASVspoof 2019 corpus. First, Section V-A evaluates the performance on the LA and PA evaluation sets of the anti-spoofing system based on a LCNN, which is trained using different embedding sizes, batch sizes and training techniques. Then, Section V-B is devoted to evaluate the performance of the anti-spoofing system based on a more complex neural network (LC-GRNN), which is trained with the proposed loss functions, and its performance is compared to other state-of-the-art systems.

A. LCNN RESULTS

1) Evaluation of the kernel function

The objective of this experiment is to analyze the performance of the proposed KDE loss functions when using different types of kernel functions. Table 4 reports the EERs obtained when training the LCNN with the proposed KDE based loss functions, with different kernels and using learnable bandwidths per training class (see next section for more details about the optimization of the bandwith). From our preliminary experiments, we chose an embedding size of 32 and a batch size of 140. As can be seen, the best performance is obtained with the Gaussian kernel, followed by the Epanechnikov [48], triangular [64] and uniform [64] kernels, respectively. The maximum difference of EER is 0.34 and 0.43%, which is achieved when comparing the uniform and Gaussian kernels in the KDE softmax and contrast loss function on the LA and PA scenarios, respectively. This means that there are no large differences of performance when employing different kernels. Since the Gaussian kernel obtains the best results, we will use it in the rest of the paper, and the resulting loss function will be referred to as Gaussian kernel density estimation (GKDE) based loss function.

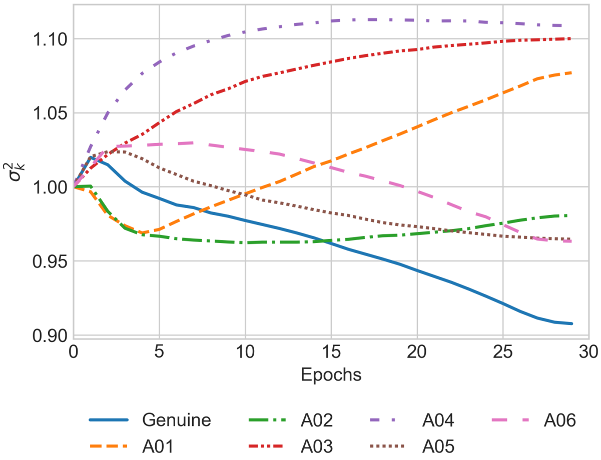

FIGURE 2. Class bandwidths optimization along the training process of the GKDE softmax loss function in the LA evaluation scenario.

2) Evaluation of the GKDE bandwidth

Next, we evaluate the performance achieved by the GKDE losses when using either fixed bandwiths σ_k^2 (0.5, 1.0 and 2.0) or learnable bandwidths, which are optimized along with the rest of parameters of the LCNN. As can be seen in Table 5, using a fixed bandwidth of $\sigma_k^2 = 1.0$ slightly achieves a better performance than using fixed bandwidths of $\sigma_k^2 = 0.5$ and $\sigma_k^2 = 2.0$. This can be due to the effect of under-smoothing and over-smoothing when using small and large bandwidths, respectively. However, the best performance is always obtained when the class bandwidths are optimized along with the rest of parameters of the neural network. For instance, optimizing the bandwidths with the rest of parameters overcomes the fixed bandwidth of $\sigma_k^2 = 1.0$ by an absolute EER of 0.68 and 0.63 % when evaluating the GKDE triplet loss function on the LA and PA scenarios, respectively.

Fig. 2 shows the optimization process of the class bandwidths through the different epochs when training the LCNN for the LA scenario. Despite the values for the different

TABLE 4. Results on ASVspoof 2019 logical and physical access test sets in terms of EER (%) of the LCNN based anti-spoofing system trained using KDE based loss functions (embeddings size of 32 and batch size of 140) with different kernel functions and optimizable bandwidths.

Loss Function	EER (%) (Logical Access / Physical Access)			
	Uniform	Triangular	Epanechnikov	Gaussian
KDE - Softmax	5.38 / 2.71	5.17 / 2.46	5.09 / 2.26	5.04 / 2.32
KDE - Softmax + Contrast	5.19 / 2.16	5.05 / 2.04	4.93 / 1.85	4.85 / 1.73
KDE - Triplet	4.97 / 1.97	4.76 / 1.82	4.65 / 1.74	**4.56 / 1.65**

TABLE 5. Results on ASVspoof 2019 logical and physical access test sets in terms of EER (%) of the LCNN based anti-spoofing system trained using GKDE based loss functions (embeddings size of 32 and batch size of 140) with fixed and optimizable bandwidths.

Loss Function	EER (%) (Logical Access / Physical Access)			
	$\sigma_k^2 = 0.5$	$\sigma_k^2 = 1.0$	$\sigma_k^2 = 2.0$	Learnable σ_k^2
GKDE - Softmax	5.91 / 3.37	5.57 / 2.92	6.06 / 3.59	5.04 / 2.32
GKDE - Softmax + Contrast	5.86 / 2.81	5.65 / 2.42	5.91 / 2.97	4.85 / 1.73
GKDE - Triplet	5.49 / 2.62	5.24 / 2.28	5.57 / 2.70	**4.56 / 1.65**

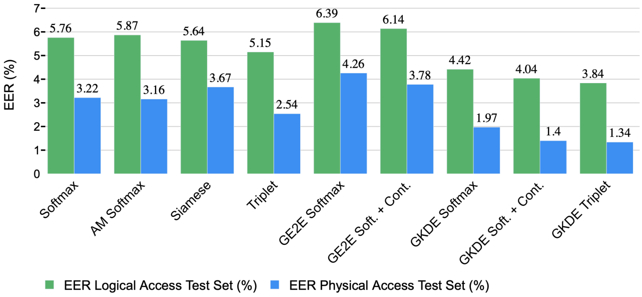

FIGURE 3. Bar plot of pooled EERs (%) evaluated in the logical and physical access test sets when the LCNN (embedding and batch size: 32 and 280, respectively) is trained with different techniques: (i) softmax; (ii) angular softmax; (iii) triplet loss; (iv) GE2E softmax; (v) GE2E softmax + contrast; (vi) GKDE softmax; (vii) GKDE softmax + contrast; (viii) GKDE triplet.

classes are not very different, the bandwidth of the genuine class is the one which achieves the smallest value, followed by the two types of VC attacks (A05 and A06). This result makes sense since genuine speech should be the most homogeneous class in the space of spoofing-aware embedding vectors. Furthermore, let us consider the three different types of speech data in the LA training set: (i) genuine speech, (ii) converted speech using two types of VC techniques (A05 and A06), and (iii) artificial speech using four types of TTS techniques (A01, A02, A03 and A04). As can be seen, the optimized bandwidths are similar within each group of speech nature, apart from the A02 attack which results to be more similar to VC attacks. This can be due to the fact that the waveform generator and acustic model employed to generate A02 attack are similar to the ones employed for generating the A05 attack [8].

According to the results of this study, we use learnable bandwidths in the rest of experiments of this work.

3) Evaluation of the embeddings size

Table 6 reports the EER and min-tDCF metrics achieved by the LCNN-based anti-spoofing system when trained using the maximum batch size which we can hold in our computational resources of 280 utterances ($N = 7$ classes and $M = 40$ utterances per class for the LA scenario, and $N = 2$ classes and $M = 140$ utterances per class for the PA scenario), different embedding sizes (16, 32 and 64) and the loss functions described in Sections II and III, namely: softmax, angular softmax, siamese, triplet, GE2E softmax, GE2E softmax and contrast, GKDE softmax, GKDE softmax and contrast, and GKDE triplet. It can be seen that the proposed GKDE based loss functions yield the best performance in terms of EER and min-tDCF, irrespective of the embedding size, on both the LA and PA evaluation scenarios. Regarding the loss functions described in Section II, the triplet loss achieves the best performance on both the LA and PA scenarios, followed by the softmax, angular softmax and siamese techniques. On the other hand, the GE2E based loss functions yield the worst performance. This could be due to the effect of smoothing caused by the use of a centroid for representing each class.

Moreover, the use of an embedding size of 32 is the best option for almost all the loss functions, and this size matches the embedding size selected in [20], which employs a similar LCNN based anti-spoofing system. To highlight the performance differences between the different techniques, Fig. 3 shows the pooled EERs achieved by each technique when using an embedding size of 32. As it can be seen, the proposed GKDE softmax loss function outperforms its counterpart softmax and GE2E softmax loss functions by an absolute pooled EER of 1.34 and 1.97% in the LA scenario, respectively, as well as by 1.25 and 2.29% in the PA scenario, respectively. Furthermore, when the softmax GE2E and GKDE loss functions are combined with a contrast loss, they yield a better performance due to the fact that the contrast loss helps to increase the inter-class variance. Related to this fact, the GKDE triplet loss, which is able to increase the inter-class variance while decreasing the intra-class variance at the same time, yields the best performance of all loss functions, outperforming its counterpart triplet loss by an absolute pooled EER of 1.31 and 1.20% in the LA and PA test sets, respectively.

TABLE 6. Results on ASVspoof 2019 logical and physical access test sets in terms of EER (%) and min-tDCF of the LCNN based anti-spoofing system trained using different loss functions and embedding sizes, and a batch size of 280 utterances.

Loss Function	Logical Access Results (EER (%) / min-tDCF)			Physical Access Results (EER (%) / min-tDCF)		
	Emb. Size: 16	Emb. Size: 32	Emb. Size: 64	Emb. Size: 16	Emb. Size: 32	Emb. Size: 64
Softmax	5.84 / 0.1106	5.76 / 0.1078	6.10 / 0.1124	3.46 / 0.0975	3.22 / 0.0885	3.38 / 0.0984
Angular Softmax	6.04 / 0.1118	5.87 / 0.1084	6.21 / 0.1158	3.54 / 0.1003	3.16 / 0.0878	3.32 / 0.0917
Siamese	5.88 / 0.1062	5.64 / 0.1044	6.01 / 0.1087	4.02 / 0.1063	3.67 / 0.1021	3.51 / 0.0976
Triplet	5.38 / 0.1025	5.15 / 0.1001	5.42 / 0.1034	2.70 / 0.0822	2.54 / 0.0806	2.81 / 0.0828
GE2E - Softmax	6.74 / 0.1212	6.39 / 0.1138	6.89 / 0.1296	4.39 / 0.1151	4.26 / 0.1102	4.52 / 0.1164
GE2E - Softmax + Contrast	6.34 / 0.1178	6.14 / 0.1127	6.44 / 0.1202	3.94 / 0.1041	3.78 / 0.1008	4.11 / 0.1086
GKDE - Softmax	4.76 / 0.1009	4.42 / 0.0935	4.64 / 0.0994	2.19 / 0.0735	1.97 / 0.0711	2.28 / 0.0756
GKDE - Softmax + Contrast	4.51 / 0.0961	4.04 / 0.0905	**4.32** / **0.0903**	1.56 / 0.0517	1.40 / 0.0465	1.74 / **0.0548**
GKDE - Triplet	**4.28** / **0.0911**	**3.84** / **0.0857**	4.35 / 0.0943	**1.51** / **0.0486**	**1.34** / **0.0452**	**1.65** / 0.0557

4) Evaluation of the batch size

Fig. 4 and 5 shows the pooled EERs evaluated in the LA and PA test sets, respectively, obtained by training the LCNN with different loss functions and using different batch sizes (70, 140, 210 and 280). The objective is to study the effect of the batch size on the anti-spoofing results. The softmax, angular softmax and siamese loss functions are not affected by the selection of the batch size, since they almost obtain the same EER in the four cases of batch size. However, the performance of the rest of loss functions does depend on the batch size. For instance, the triplet loss employs an online selection of the positive and negative samples within the batch, and it is more likely to find hard samples in a larger mini-batch. Likewise, the GE2E and GKDE based loss functions attain better performance when increasing the batch size, since a better representation of every class is obtained. Moreover, this performance difference is more noticeable in the LA scenario than in the PA scenario, due to the fact that $M = 40$ utterances per class are employed in the LA scenario, while $M = 140$ utterances per class are used for training the LCNN in the PA scenario. It is also quite remarkable that the proposed GKDE based loss functions are the ones which quantitatively improve more their performance when using a larger batch size. This is due to the fact that KDE estimates the pdf of each class in a 32-dimensional space (embedding size) by placing a probability mass at every embedding sample within the mini-batch, so the more samples per mini-batch are used the more accurate is the representation of the pdf for the classes. In contrast, the GE2E based techniques represent each class with a centroid, being this representation less affected by the changes in the batch size in comparison with the KDE-based representation of every class in GKDE.

5) t-SNE embeddings representation

For illustrative purposes, we represent the LA test embeddings (10,000 embeddings per class) in a two-dimensional space using t-SNE [65], which preserves distances in a two-dimension space. Fig. 6 shows the embeddings obtained by the following loss functions: (a) softmax loss, (b) GKDE softmax loss, (c) triplet loss, and (d) GKDE triplet loss. As we can see, the clusters of the different LA attacks and genuine class are more separated in the GKDE based loss functions than in the classical softmax and triplet losses, which explains the better performance of the proposed GKDE based loss functions. According to the results of the ASVspoof 2019 challenge [8], the VC attack A17, which is generated using waveform filtering and employing a variational autoencoder as acoustic model, is the most difficult to detect. This fact can also be seen in the t-SNE embeddings representations, where the cluster of the A17 attack is the one that overlaps the most with the genuine class cluster in the four cases.

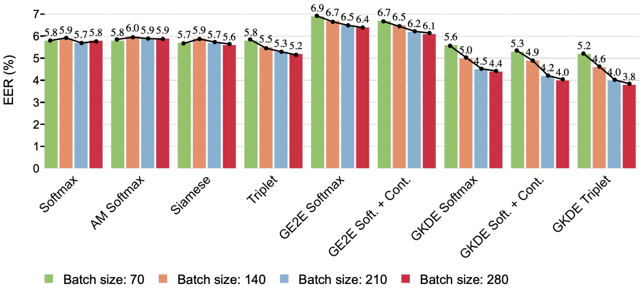

FIGURE 4. Bar plot of pooled EERs (%) evaluated in the logical access test set using an embedding size of 32 and training the LCNN with different batch sizes and techniques.

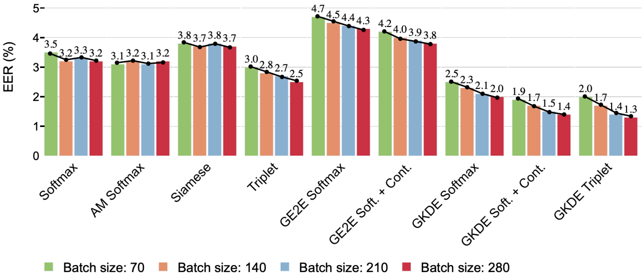

FIGURE 5. Bar plot of pooled EERs (%) evaluated in the physical access test set using an embedding size of 32 and training the LCNN with different batch sizes and techniques.

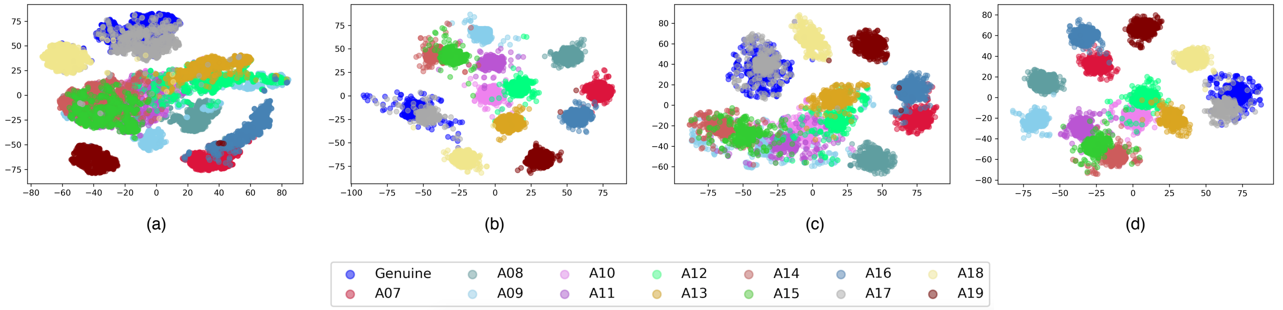

FIGURE 6. Representation of the logical access test embeddings using t-SNE: (a) softmax loss; (b) GKDE softmax loss; (c) triplet loss; (d) GKDE triplet loss.

TABLE 7. Comparison of single anti-spoofing systems performance on the ASVspoof 2019 logical and physical access test sets in terms of EER (%) and min-tDCF.

System	EER (%) / min-tDCF	
	Logical Access	Physical Access
Baseline: CQCC + GMM [8]	9.57 / 0.2366	11.04 / 0.2454
Baseline: LFCC + GMM [8]	8.09 / 0.2116	13.54 / 0.3016
STFT + LCNN + AM [37]	4.53 / 0.1028	-
CQT + LCNN + AM [37]	-	1.23 / 0.0295
TDNN + Softmax [66]	8.44 / 0.2251	-
SincNet + Softmax [67]	20.11 / 0.3563	2.11 / 0.0527
ResNet + Softmax [11]	7.69 / 0.2166	4.43 / 0.1070
LC-GRNN + Softmax [9]	6.28 / 0.1523	2.23 / 0.0614
GRCNN + Softmax [19]	3.85 / 0.0952	1.09 / 0.0234
LC-GRNN + GKDE - Softmax	3.77 / 0.0842	1.06 / 0.0222
LC-GRNN + GKDE - Soft. + Cont.	3.39 / 0.0805	0.97 / 0.0210
LC-GRNN + GKDE - Triplet	**3.03** / **0.0776**	**0.92** / **0.0198**

B. LC-GRNN RESULTS

To study the effect of employing a more complex neural network architecture, we also evaluated the effectiveness of the proposed GKDE losses on the LC-GRNN.

Table 7 compares the performance attained with the proposed GKDE based loss functions on the ASVspoof 2019 database using the LC-GRNN architecture and other other state-of-the-art single anti-spoofing systems from the literature. As can be seen, our proposed systems outperform the baseline anti-spoofing systems released with this database (CQCC + GMM and LFCC + GMM), as well as the other top performing single systems (presented to the ASVspoof 2019 Challenge [8]) and our previous GRCNN [19], in both the LA and PA scenarios. Specifically, the LC-GRNN trained with the GKDE based triplet loss yields a 5.06 % and 10.12 % lower pooled EER than the best baseline systems of the LA and PA scenarios, respectively. In addition, it achieves a 3.25 % and 1.31 % better pooled EER than the same system trained with the classical softmax loss proposed in our previous work [9] for both the LA and PA scenarios, respectively.

According to this evaluation, we can conclude that the proposed GKDE based loss functions are effective for different types of neural network architectures such as CNNs, RNNs and their combination. Moreover, the proposed single anti-spoofing systems are among the best state-of-the-art systems at detecting the recent attacks based on the latest technologies [8].

VI. CONCLUSION

In this paper we proposed various loss functions, based on kernel density estimation (KDE) techniques, which estimate the probability density function (pdf) of every training class in each mini-batch, and compute a log likelihood matrix by using the embedding vectors and pdfs of all training classes within the mini-batch. These loss functions address three main problems that have been detected in conventional loss functions: (i) the training samples which belong to the same class are kept close to each other and the dissimilar instances are kept far away on the embedding space by using hard negative mining, (ii) the training classes are fully represented by all the samples within the mini-batch, by estimating with KDE a pdf per class which places a probability mass at every embedding sample, and (iii) the concept of distance measure between embedding vectors is replaced by the concept of the probability that an embedding vector belongs to a certain class, which has the advantage of avoiding the selection of an appropiate distance measure and embedding normalization technique. Experimental results on the ASVspoof 2019 database have shown that the proposed losses outperform other conventional loss functions that have been used so far for training DNN-based antispoofing systems. Furthermore, it is shown that the performance gains are not restricted to a sole neural network architecture, but the proposed loss functions are effective for training different types of neural networks such as CNNs, RNNs and their combination.

We hope that this new concept of loss functions can be rather considered a general approach since it can be applied to any DNN-based embedding extraction system which comprises fully connected layers. As future work, we will evaluate the proposed loss functions in other speech related tasks such as ASV and integration of ASV and PAD systems.

ACKNOWLEDGMENT

This work has been supported by the Spanish MINECO/FEDER Project TEC2016-80141-P. Alejandro Gomez-Alanis holds a FPU fellowship from the Spanish Ministry of Education (FPU16/05490). Jose A. Gonzalez-Lopez holds a Juan de la Cierva-Incorporación fellowship from the Spanish Ministry of Science, Innovation and Universities (IJCI-2017-32926). We also acknowledge the support of NVIDIA Corporation with the donation of a Titan X GPU.

REFERENCES

[1] A. K. Jain, A. Ross, and S. Pankanti, "Biometrics: a tool for information security," IEEE Transactions on Information Forensics and Security, vol. 1, no. 2, pp. 125–143, 2006.

[2] R. Naika, "An overview of automatic speaker verification system," Advances in Intelligent Systems and Computing, vol. 673, 2018.

[3] Z. Wu, N. W. D. Evans, T. Kinnunen, J. Yamagishi, F. Alegre, and H. Li, "Spoofing and countermeasures for speaker verification: A survey," Speech Communication, vol. 66, pp. 130–153, 2014.

[4] Z. Wu, P. L. D. Leon, C. Demiroglu, A. Khodabakhsh, S. King, Z.-H. Ling, D. Saito, B. Stewart, T. Toda, M. Wester, and J. Yamagishi, "Anti-spoofing for text-independent speaker verification: An initial database, comparison of countermeasures, and human performance," IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 24, pp. 768–783, 2016.

[5] "Presentation attack detection." [Online]. Available: https://www.iso.org/standard/67381.html

[6] Z. Wu, T. Kinnunen, N. W. D. Evans, J. Yamagishi, C. Hanilçi, M. Sahidullah, and A. Sizov, "ASVspoof 2015: the first automatic speaker verification spoofing and countermeasures challenge," in Proc. Interspeech, 2015.

[7] T. Kinnunen, M. Sahidullah, H. Delgado, M. Todisco, N. W. D. Evans, J. Yamagishi, and K.-A. Lee, "The ASVspoof 2017 challenge: Assessing the limits of replay spoofing attack detection," in Proc. Interspeech, 2017.

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

[9] A. Gomez-Alanis, A. M. Peinado, J. A. Gonzalez, and A. M. Gomez, "A light convolutional GRU-RNN deep feature extractor for ASV spoofing detection," in Proc. Interspeech, 2019.

[10] 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, 2015.

[11] M. Alzantot, Z. Wang, and M. B. Srivastava, "Deep residual neural networks for audio spoofing detection," in Proc. Interspeech, 2019.

[12] A. Gomez-Alanis, A. M. Peinado, J. A. Gonzalez, and A. M. Gomez, "Performance evaluation of front- and back-end techniques for ASV spoofing detection systems based on deep features," in Proc. Iberspeech, 2018.

[13] 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, pp. 684–694, 2017.

[14] Z. Yu, C. Zhao, Z. Wang, Y. Qin, Z. Su, X. Li, F. Zhou, and G. Zhao, "Searching central difference convolutional networks for face anti-spoofing," in Proc. Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[15] L. Li, Z. Xia, A. Hadid, X. Jiang, H. Zhang, and X. Feng, "Replayed video attack detection based on motion blur analysis," IEEE Transactions on Information Forensics and Security, vol. 14, pp. 2246–2261, 2019.

[16] Y. Qian, N. Chen, H. Dinkel, and Z. Wu, "Deep feature engineering for noise robust spoofing detection," IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 25, no. 10, pp. 1942–1955, 2017.

[17] A. Gomez-Alanis, A. M. Peinado, J. A. Gonzalez, and A. M. Gomez, "A deep identity representation for noise robust spoofing detection," in Proc. Interspeech, 2018.

[18] Y. Qian, N. Chen, and K. Yu, "Deep features for automatic spoofing detection," Speech Communication, vol. 85, pp. 43–52, 2016.

[19] 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, vol. 27, no. 12, pp. 1985–1999, 2019.

[20] G. Lavrentyeva, S. Novoselov, E. Malykh, A. Kozlov, O. Kudashev, and V. Shchemelinin, "Audio replay attack detection with deep learning frameworks," in Proc. Interspeech, 2017.

[21] M. Sahidullah, T. Kinnunen, and C. Hanilçi, "A comparison of features for synthetic speech detection," in Proc. Interspeech, 2015.

[22] M. Todisco, H. Delgado, and N. W. D. Evans, "Constant Q cepstral coefficients: A spoofing countermeasure for automatic speaker verification," Computer Speech and Language, vol. 45, pp. 516–535, 2017.

[23] H. Muckenhirn, M. Magimai-Doss, and S. Marcel, "End-to-end convolutional neural network-based voice presentation attack detection," Proc. IEEE International Joint Conference on Biometrics (IJCB), pp. 335–341, 2017.

[24] K. Sriskandaraja, V. Sethu, and E. Ambikairajah, "Deep siamese architecture based replay detection for secure voice biometric," in Proc. Interspeech, 2018.

[25] L. Wan, Q. Wang, A. Papir, and I. Lopez-Moreno, "Generalized end-to-end loss for speaker verification," in Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2019.

[26] H.-S. Heo, J. weon Jung, I.-H. Yang, S.-H. Yoon, H. jin Shim, and H.-J. Yu, "End-to-end losses based on speaker basis vectors and all-speaker hard negative mining for speaker verification," in Proc. Interspeech, 2019.

[27] J. Li, M. Sun, X. Zhang, and Y. Wang, "Joint decision of anti-spoofing and automatic speaker verification by multi-task learning with contrastive loss," IEEE Access, vol. 8, pp. 7907–7915, 2020.

[28] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," in Proc. SIMBAD, 2015.

[29] D. Yi, Z. Lei, S. Liao, and S. Z. Li, "Deep metric learning for person re-identification," in Proc. International Conference on Pattern Recognition, 2014, pp. 34–39.

[30] K. Sohn, "Improved deep metric learning with multi-class n-pair loss objective," in Proc. Neural Information Processing Systems (NIPS), 2016.

[31] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in Proc. IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[32] E. Parzen, "On estimation of a probability density function and mode," The Annals of Mathematical Statistics, vol. 33, pp. 1065–1076, 1962.

[33] W. Hardle, A. Werwatz, M. Muller, and S. Sperlich, "Nonparametric and semiparametric models," Springer Series in Statistics, 2004.

[34] A. M. Peinado, J. Koloda, A. M. Gomez, and V. E. Sanchez, "A statistical analysis of the kernel-based mmse estimator with application to image reconstruction," Signal Processing: Image Communication, vol. 55, pp. 41–54, 2017.

[35] L. Weiyang, W. Yandong, Y. Zhiding, L. Ming, R. Bhiksha, and S. Le, "Sphereface: Deep hypersphere embedding for face recognition," in Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[36] W. Feng, C. Jian, L. Weiyang, and L. Haijun, "Additive margin softmax for face verification," IEEE Signal Processing Letters, vol. 25, no. 7, pp. 926–930, 2018.

[37] G. Lavrentyeva, S. Novoselov, A. Tseren, M. Volkova, A. Gorlanov, and A. Kozlov, "STC anti-spoofing systems for the asvspoof2019 challenge," in Proc. Interspeech, 2019.

[38] Y. Yu, L. Fan, and W. Li, "Ensemble additive margin softmax for speaker verification," in Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2019.

[39] L. Yutian, G. Feng, O. Zhijian, and S. Jiasong, "Angular softmax loss for end-to-end speaker verification," in Proc. International Symposium on Chinese Spoken Language Processing, 2018.

[40] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014, pp. 1701–1708.

[41] E. Ahmed, M. Jones, and T. K. Marks, "An improved deep learning architecture for person re-identification," in Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 3908–3916.

[42] K. Chen and A. Salman, "Extracting speaker-specific information with a regularized siamese deep network," in Proc. Neural Information Processing Systems (NIPS), 2011.

[43] C. Zhang, K. Koishida, and J. H. L. Hansen, "Text-independent speaker verification based on triplet convolutional neural network embeddings," IEEE/ACM Transactions on Audio, Speech and Language Processing, vol. 26, no. 9, pp. 1633–1644, 2018.

[44] C. Zhang and K. Koishida, "End-to-end text-independent speaker verification with triplet loss on short utterances," in Proc. Interspeech, 2017.

[45] J. Koloda, A. M. Peinado, and V. Sanchez, "Kernel-based MMSE multimedia signal reconstruction and its application to spatial error concealment," IEEE Transactions on Multimedia, vol. 16, no. 6, pp. 1729–1738, 2014.

[46] P. Green, A. H. Seheult, and B. Silverman, "Density estimation for statistics and data analysis," Monographs on Statistics and Applied Probability, vol. 26, 1988.

[47] B. Turlach, "Bandwidth selection in kernel density estimation: A review," CORE and Institut de Statistique, vol. 19, pp. 1–33, 1993.

[48] V. A. Epanechnikov, "Non-parametric estimation of a multivariate probability density," Theory of Probability and its Applications, vol. 14, no. 1, pp. 153–158, 1969.

[49] K. Weinberger, J. Blitzer, and L. Saul, "Distance metric learning for large margin nearest nearest neighbor classification," in Proc. Neural Information Processing Systems (NIPS), 2006.

[50] D. Matrouf, J.-F. Bonastre, and C. Fredouille, "Effect of speech transformation on impostor acceptance," in Proc. IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP), 2006, pp. 933–936.

[51] M. Morise, F. Yokomori, and K. Ozawa, "WORLD: A vocoder-based high-quality speech synthesis system for real-time applications," IECE Transactions on Information and Systems, vol. 99, no. 7, pp. 1877–1884, 2016.

[52] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," ArXiv, vol. abs/1609.03499, 2016.

[53] D. W. Griffin and J. S. Lim, "Signal estimation from modified short-time fourier transform," in Proc. IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP), 1983.

[54] K. Tanaka, T. Kaneko, N. Hojo, and H. Kameoka, "Synthetic-to-natural speech waveform conversion using cycle-consistent adversarial networks," in Proc. IEEE Spoken Language Technology Workshop (SLT), 2018, pp. 632–639.

[55] X. Wang, S. Takaki, and J. Yamagishi, "Neural source-filter-based waveform model for statistical parametric speech synthesis," in Proc. IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP), 2019, pp. 5916–5920.

[56] K. Kobayashi, T. Toda, G. Neubig, S. Sakti, and S. Nakamura, "Statistical singing voice conversion with direct waveform modification based on the spectrum differential," in Proc. Interspeech, 2014.

[57] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. C. Courville, and Y. Bengio, "Maxout networks," in Proc. International Conference on Machine Learning (ICML), 2013.

[58] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in Proc. International Conference on Learning Representations (ICLR), 2015.

[59] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. Devito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in Proc. Neural Information Processing Systems (NIPS), 2017.

[60] H. J. Wang, Y. Wang, Z.-F. Zhou, X. Ji, Z. Li, D. Gong, J. Zhou, and W. Liu, "Cosface: Large margin cosine loss for deep face recognition," in Proc. IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 5265–5274.

[61] J. Deng, J. Guo, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 4685–4694.

[62] N. N. Vo and J. Hays, "Localizing and orienting street views using overhead imagery," in Proc. European Conference on Computer Vision (ECCV), 2016.

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

[64] Y. Soh, Y. Hae, A. Mehmood, R. H. Ashraf, and I. Kim, "Performance evaluation of various functions for kernel density estimation," Open Journal of Applied Sciences, vol. 3, no. 1, pp. 58–64, 2013.

[65] L. van der Maaten and G. E. Hinton, "Visualizing data using t-SNE," Journal of Machine Learning Research, vol. 9, pp. 2579–2605, 2008.

[66] C. Su-Yu, W. Kai-Cheng, and C. Chia-Ping, "Transfer-representation learning for detecting spoofing attacks with converted and synthesized speech in automatic speaker verification system," in Proc. Interspeech, 2019.

[67] H. Zeinali, T. Stafylakis, G. Athanasopoulou, J. Rohdin, I. Gkinis, L. Burget, and J. H. Cernocký, "Detecting spoofing attacks using VGG and SincNet: BUT-Omilia submission to ASVspoof 2019 challenge," in Proc. Interspeech, 2019.

Alejandro Gomez-Alanis was born in Granada, Spain, in 1994. He received the B.Sc. and M.Sc. degrees in telecommunications engineering from the University of Granada, Spain, in 2016 and 2018, respectively. In 2016 and 2017 he worked as a Software Engineer at Seplin and Strivelabs for developing automatic statistical tools. Since late 2017 he holds an FPU fellowship for pursuing the Ph.D. degree with the Department of Signal Theory, Telematics and Communications at the University of Granada working on speech biometrics. His research activities focus on the processing, modelling, and classification of speech for human-oriented applications.

Jose A. Gonzalez received the B.Sc. and Ph.D. degrees in computer science, both from the University of Granada, Granada, Spain, in 2006 and 2013, respectively. He then spent four years as a Postdoctoral Research Associate at the University of Sheffield, Sheffield, U.K., working on silent speech technology with special focus on speech synthesis from speech-related biosignals. In late 2017 he took up a Lectureship at the Department of Languages and Computer Sciences, University of Malaga, Spain. Since 2019 he holds a Juan de la Cierva - Incorporacion fellowship at the University of Granada, working on silent speech interfaces and speech biometrics. His research activities focus on the processing, modelling, and classification of speech for human-centered applications. He has authored or co-authored more than 60 papers in these areas.

Antonio M. Peinado (M'95–SM'05) received the M.S. and Ph.D. degrees in physics (electronics specialty) from the University of Granada, Granada, Spain, in 1987 and 1994, respectively. In 1988, he worked with Inisel as a Quality Control Engineer. Since 1988, he has been with the University of Granada, where he has led several research projects related to signal processing and transmission. In 1989, he was a Consultant with the Speech Research Department, AT&T Bell Labs, Murray Hill, NJ, USA, and, in 2018, a Visiting Scholar with the Language Technologies Institute of CMU, Pittsburgh, PA, USA. He has held the positions of an Associate Professor from 1996 to 2010 and a Full Professor since 2010 with the Department of Signal Theory, Networking and Communications, University of Granada, where he is currently the Head of the research group on Signal Processing, Multimedia Transmission and Speech/Audio Technologies. He authored numerous publications in international journals and conferences, and has co-authored the book entitled Speech Recognition Over Digital Channels (New York, NY, USA: Wiley, 2006). His current research interests are focused on several speech technologies (anti-spoofing for automatic speaker verification, speech enhancement, and robust speech recognition and transmission), image processing and proteomic signal processing. Prof. Peinado has been a reviewer for a number of international journals and conferences, an evaluator for project and grant proposals, and a Member of the technical program committee of several international conferences.