A Novel Loss Function and Training Strategy for Noise-Robust Keyword Spotting

Espejo, Iván López; Tan, Zheng-Hua; Jensen, Jesper

Published in:
IEEE/ACM Transactions on Audio, Speech, and Language Processing

DOI (link to publication from Publisher):
10.1109/TASLP.2021.3092567

Publication date:
2021

Document Version
Accepted author manuscript, peer reviewed version

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/TASLP.2021.3092567, IEEE/ACM Transactions on Audio, Speech, and Language Processing

A Novel Loss Function and Training Strategy for Noise-Robust Keyword Spotting

Iván López-Espejo, Zheng-Hua Tan, *Senior Member, IEEE*, and Jesper Jensen

***Abstract*—The development of keyword spotting (KWS) systems that are accurate in noisy conditions remains a challenge. Towards this goal, in this paper we propose a novel training strategy relying on multi-condition training for noise-robust KWS. By this strategy, we think of the state-of-the-art KWS models as the composition of a keyword embedding extractor and a linear classifier that are successively trained. To train the keyword embedding extractor, we also propose a new $(C_{N,2}+1)$-pair loss function extending the concept behind related loss functions like triplet and N-pair losses to reach larger inter-class and smaller intra-class variation. Experimental results on a noisy version of the Google Speech Commands Dataset show that our proposal achieves around 12% KWS accuracy relative improvement with respect to standard end-to-end multi-condition training when speech is distorted by unseen noises. This performance improvement is achieved without increasing the computational complexity of the KWS model.**

***Index Terms*—Keyword spotting, noise robustness, multi-condition training, deep metric learning, loss function, keyword embedding.**

I. Introduction

VOICE-activated assistants such as Amazon's Alexa and Apple's Siri are revolutionizing our daily lives by enabling comfortable interaction between users and electronic devices [1]. Devices with voice assistants, e.g., smart speakers, are commonly activated by means of a wake-up-word or keyword. Hence, for a good user experience, accurate keyword spotting (KWS) systems are desirable. For example, to avoid our privacy to be compromised, low false acceptance rates are required, as, otherwise, unintended speech might be picked up by voice assistants. On the other hand, a low miss rate is desired for comfortable usage.

Despite all the research progress, acoustic noise is still one of the most notable factors that negatively affect KWS performance and that of automatic speech recognition (ASR) in general [2]. Recent work encompasses both single- [3] and multi-channel [4]–[6] noise-robust KWS. In [3], Yu *et al.* develop a text-dependent single-channel speech enhancement front-end based on a long short-term memory (LSTM) recurrent neural network (RNN). This front-end computes a mask that is applied to the input spectrogram before feeding it to a small-footprint (that is, low memory and low computational complexity) keyword spotter. A multi-channel approach is adopted in [4], where a neural network-based keyword spotter with a feedback loop to a noise reduction front-end inspired by the human auditory system is proposed. Huang *et al.* [6] propose a short-time Fourier transform (STFT)-based adaptive noise cancellation method modified to employ deferred filter coefficients in order to retrieve keywords from noisy signals. Furthermore, in [5], beamformed signals are combined on the basis of an attention mechanism for end-to-end noise-robust KWS. While all of these techniques improve KWS performance in noisy acoustic conditions, this is achieved at the expense of a significant increase of the computational complexity.

Multi-condition training remains one of the most successful approaches for noise-robust ASR [7]. It consists of training the acoustic model with speech data either collected from different (noisy) acoustic conditions or artificially distorted with a variety of noises through a procedure called data augmentation. For instance, it has been reported that the performance of a multi-condition trained acoustic model is highly competitive with that of a state-of-the-art method like the application of teacher-student learning using a parallel clean and noisy speech corpus [8]. Furthermore, in general, combining speech enhancement approaches with multi-condition trained acoustic models has not yielded the expected improvements with respect to just exploiting effective multi-condition training [9].

Indeed, KWS has also benefited from multi-condition training based on data augmentation [10]. Very recently, Gao *et al.* [11] explored multi-condition training with stratified data augmentation for KWS robust to different acoustic conditions, achieving a comparable performance to that of a production-grade model. Moreover, it is worth to remark the following positive feature of multi-condition training: it improves robustness to noise while not increasing the computational complexity of the model.

In this paper, we propose a novel training strategy relying on multi-condition training for noise-robust KWS. Many state-of-the-art KWS systems consist of the extraction of speech features that are fed into a deep learning model comprising a final fully-connected layer with softmax activation for classification [12]–[18], as in Figure 1. In this work, we think of the input vectors to those final fully-connected layers as linearly classifiable *keyword embeddings*. Thus, instead of training such models in an end-to-end manner using cross-entropy loss, which is commonly done [12]–[15], we propose to divide their training into two successive stages:

Manuscript received month day, year; revised month day, year; accepted month day, year. Date of publication month day, year; date of current version month day, year. This work was supported, in part, by the Demant Foundation. The associate editor coordinating the review of this manuscript and approving it for publication was xxyyzz xxyyzz.

I. López-Espejo, and Z.-H. Tan are with the Department of Electronic Systems, Aalborg University, Aalborg 9220, Denmark (e-mail: ivl@es.aau.dk; zt@es.aau.dk).

J. Jensen is with the Department of Electronic Systems, Aalborg University, Aalborg 9220, Denmark, and also with Oticon A/S, Smørum 2765, Denmark (e-mail: jje@es.aau.dk; jesj@oticon.com).

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/TASLP.2021.3092567, IEEE/ACM Transactions on Audio, Speech, and Language Processing

1) The final fully-connected layer with softmax activation (linear classifier) is removed and the remaining model is multi-condition trained to be a discriminative and noise-robust keyword embedding extractor;
2) Then, the linear classifier is trained using cross-entropy loss and taking as input keyword embeddings extracted by means of the model resulting from the first stage.

As a second contribution of this paper, and to train the keyword embedding extractor, we suggest a new $(C_{N,2}+1)$-pair loss function, which is related to other well-known tuple-based loss functions like triplet loss [19] and N-pair loss [20]. In fact, recent work on KWS has explored the use of triplet loss —either standalone [21] or in combination with the reversed triplet and hinge loss functions [22]— and related losses, like an angular variant of the prototypical loss [23] (sharing many similarities with the N-pair loss), for word embedding learning. However, unlike these loss functions, our proposed $(C_{N,2}+1)$-pair loss imposes some restrictions on the way the negative training samples relate to each other in terms of embedding distance, as will be carefully developed in a subsequent section. This allows us to achieve larger inter-class and smaller intra-class variation for improved KWS performance in noisy acoustic conditions.

In order to support the multi-condition training paradigm, we create a noisy speech corpus with several acoustic conditions —i.e., noise types and signal-to-noise ratio (SNR) levels— from the Google Speech Commands Dataset (GSCD) [24]. Experimental results show that our proposal is able to enhance the generalization ability of a multi-condition trained state-of-the-art KWS system. This means that our proposal obtains around 12% KWS accuracy relative improvement with respect to standard end-to-end multi-condition training when speech is contaminated with noises not seen during training. Besides KWS performance gains, it is also relevant to highlight the following key features of our method: *1)* it can be applied to most of the latest (single- and multi-channel) KWS models, *2)* it does increase neither the number of parameters nor the number of multiplications of the model[1], *3)* it can potentially be useful to mitigate the effect of other types of distortions in addition to acoustic noise, and *4)* it might be exported to other application areas (e.g., image classification).

The rest of this paper is structured into five more sections. The suggested KWS training strategy is explained in Section II. In Section III, the proposed training loss function is presented along with related tuple-based loss functions. The experimental setting is described in Section IV. Experimental results and discussion are presented in Section V. Finally, Section VI concludes the paper.

II. Keyword Spotting Training Strategy

Most of the state-of-the-art KWS systems [3]–[6], [12]–[18] follow the general approach illustrated by Figure 1. First, a speech feature matrix (e.g., a normalized log-Mel feature matrix) $\mathbf{X} \in \mathbb{R}^{K\times T}$ (where K and T represent the number of features —e.g., frequency bins— and time frames, respectively) is typically computed from an input speech segment. Then, $\mathbf{X}$ is the input to a deep learning architecture modeling a function $\mathbf{f}(\cdot|\theta) : \mathbb{R}^{K\times T} \rightarrow I^N$, where θ denotes the parameters of the model and $I = [0,\ 1]$ represents the unit interval. The final layer of this architecture is a fully-connected (FC) layer with softmax activation. Hence, the output of this function, $\mathbf{y} = \mathbf{f}(\mathbf{X}|\theta)$, $\mathbf{y} \in I^N$, can be interpreted as a posterior probability distribution on the N different recognizable types of words (classes) given the input speech feature matrix $\mathbf{X}$, that is,

$$\mathbf{y}_i = P(i|\mathbf{X},\theta), \qquad i = 1, ..., N, \tag{1}$$

where subscript i denotes the i-th element of a vector and $\sum_{i=1}^{N}\mathbf{y}_i = 1$. In other words, the deep learning model solves an N-class classification problem, where, normally, $N-1$ classes correspond to $N-1$ different keywords and the remaining class is the filler (non-keyword) class.

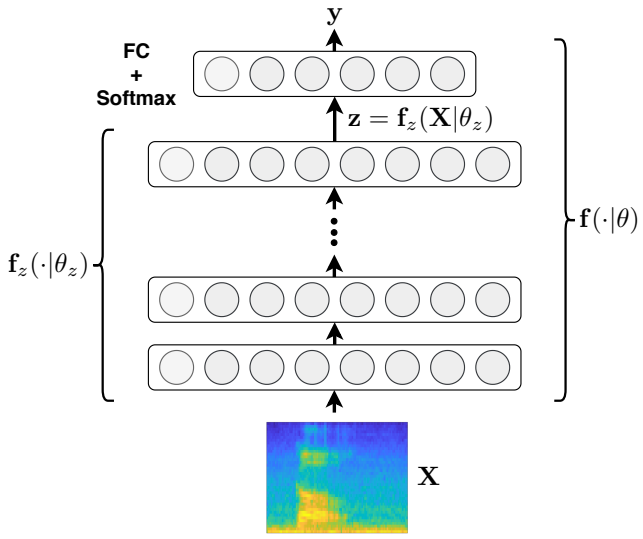


Fig. 1: General state-of-the-art KWS approach. "FC + Softmax" stands for fully-connected layer with softmax activation. See the text for further details.

In this work, we think of the input vectors to the final fully-connected layer with softmax activation (i.e., *linear classifier*), $\mathbf{z} \in \mathbb{R}^D$ (see Figure 1), as linearly classifiable keyword embeddings. Let $\mathbf{f}_z(\cdot|\theta_z) : \mathbb{R}^{K\times T} \rightarrow \mathbb{R}^D$, $\theta_z \subset \theta$, be the function modeled by the architecture of Figure 1 when removing the contribution of its final linear classifier. Then, $\mathbf{z} = \mathbf{f}_z(\mathbf{X}|\theta_z)$ can be understood as a D-dimensional keyword embedding, and $\mathbf{f}_z(\cdot|\theta_z)$, as a *keyword embedding extractor*. The keyword embedding $\mathbf{z}$, which is a more compact (i.e., lower-dimensional) representation of $\mathbf{X}$, is further processed by the final fully-connected layer with softmax activation as

$$\mathbf{y}_i = \frac{e^{(\mathbf{W}\mathbf{z}+\mathbf{b})_i}}{\sum_{j=1}^{N} e^{(\mathbf{W}\mathbf{z}+\mathbf{b})_j}}, \qquad i = 1, ..., N, \tag{2}$$

where $\mathbf{W} \in \mathbb{R}^{N\times D}$ and $\mathbf{b} \in \mathbb{R}^N$ are the weight matrix and bias vector, respectively, of the fully-connected layer. A

[1]This is of utmost importance, since KWS systems are frequently intended for relatively low-resource devices.

keyword might be spotted by simply picking the most likely class $\hat{i}$ from $\mathbf{y}$, namely,

$$\hat{i} = \underset{1 \leq i \leq N}{\operatorname{argmax}} \mathbf{y}_i. \tag{3}$$

A common practice is to end-to-end train the deep learning model $\mathbf{f}(\cdot|\theta)$ by means of cross-entropy loss (e.g., [12]–[15]). Alternatively, to obtain more discriminative and noise-robust keyword embeddings for improved KWS performance in noisy conditions, we propose the following two-stage KWS training strategy to estimate the set of parameters $\theta = \{\theta_z, \mathbf{W}, \mathbf{b}\}$:

1) *Training of the keyword embedding extractor:* First, the keyword embedding extractor $\mathbf{f}_z(\cdot|\theta_z)$ is multi-condition trained by considering a new $(C_{N,2}+1)$-pair loss function, which is formulated in Section III.
2) *Training of the linear classifier:* Second, the linear classifier of Eq. (2) is trained using multi-condition keyword embeddings extracted by means of $\mathbf{f}_z(\cdot|\theta_z)$. To this goal, cross-entropy loss [25], $\mathcal{L}_{\text{CE}}$, is used as in

$$\mathcal{L}_{\text{CE}} = -\sum_{i=1}^{N} l_i \log(\mathbf{y}_i), \tag{4}$$

where $\{l_i;\ i = 1, ..., N\}$, $l_i \in \{0, 1\}$, are the corresponding KWS true labels.

The benefits of this two-stage KWS training strategy will be clearly demonstrated throughout the following sections.

III. A New $(C_{N,2}+1)$-Pair Loss Function

The objective of the keyword embedding extractor $\mathbf{f}_z(\cdot|\theta_z)$ introduced above is to compute discriminative and noise-robust keyword embeddings to improve KWS performance in noisy acoustic conditions. To this end, $\mathbf{f}_z(\cdot|\theta_z)$ is multi-condition trained using a new $(C_{N,2}+1)$-pair loss function that is developed in the present section.

A. Related Work

Deep metric learning is widely used to obtain compact and discriminative representations —that is, embeddings— of different kind of data for different tasks like, for example, visual understanding tasks [26]. In essence, deep metric learning, which is based on the principle of data similarity, is about learning a set of hierarchical non-linear transformations to project data samples into another (embedding) space for comparison or matching [26]. The resulting set of hierarchical non-linear transformations serves as an embedding extractor.

A central aspect of deep metric learning is the tuple-based loss function. This function is, in turn, defined from a distance function measuring the similarity between pairs of embeddings computed from tuples comprising two or more data samples. Indeed, the training of the embedding extractor is targeted to minimize such a tuple-based loss function. This is equivalent to the embedding extractor producing close (distant) embeddings in case of, according to the task-related criteria, similar (dissimilar) data samples.

Without loss of generality, from now on we will assume that we want to obtain discriminative embeddings to perform N-class classification. One of the first proposed tuple-based loss functions was contrastive loss [27]. Let $\mathbf{z}_1^{(i)}$ and $\mathbf{z}_2^{(j)}$ be different D-dimensional embeddings belonging to classes i and j $(i, j = 1, ..., N)$, respectively. The 2-tuple contrastive loss, $\mathcal{L}_{\text{Contrastive}}$, is defined as

$$\begin{aligned}\mathcal{L}_{\text{Contrastive}} = {} & \delta_{ij} \mathcal{D}\left(\mathbf{z}_1^{(i)}, \mathbf{z}_2^{(j)}\right) \\ & + (1 - \delta_{ij}) \max\left(0,\ m - \mathcal{D}\left(\mathbf{z}_1^{(i)}, \mathbf{z}_2^{(j)}\right)\right), \end{aligned} \tag{5}$$

where $\mathcal{D}\left(\mathbf{z}_1^{(i)}, \mathbf{z}_2^{(j)}\right) : \mathbb{R}^D \times \mathbb{R}^D \rightarrow \mathbb{R}^+$ is a (symmetric) distance function between the embedding pair $\mathbf{z}_1^{(i)}$ and $\mathbf{z}_2^{(j)}$, $m > 0$ is a margin and δ_{ij} is the Kronecker delta,

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } i \neq j. \end{cases} \tag{6}$$

The goal of (5) is to learn embeddings $\mathbf{z}_1^{(i)}$ and $\mathbf{z}_2^{(j)}$ with a short (large) distance between them when $i = j$ $(i \neq j)$. Notice that when the distance between negative embedding pairs (i.e., when $i \neq j$) is equal to or larger than m, $\mathcal{L}_{\text{Contrastive}} = 0$ and training can be focused on more difficult negative pairs, i.e., those with short distances.

Schroff *et al.* [19] took the concept behind contrastive loss further by proposing triplet loss. Let $\mathbf{z}_a^{(i)}$, $\mathbf{z}_p^{(i)}$ and $\mathbf{z}_n^{(j)}$, $j \neq i$, denote the so-called anchor, positive and negative embeddings in such a manner that $\left\{\mathbf{z}_a^{(i)}, \mathbf{z}_p^{(i)}\right\}$ and $\left\{\mathbf{z}_a^{(i)}, \mathbf{z}_n^{(j)}\right\}$ constitute positive and negative (i.e., same- and different-class) embedding pairs, respectively. Then, triplet loss is a 3-tuple loss function that can be written as

$$\begin{aligned}\mathcal{L}_{\text{Triplet}} = \max\Big(0,\ & \mathcal{D}\left(\mathbf{z}_a^{(i)}, \mathbf{z}_p^{(i)}\right) \\ & - \mathcal{D}\left(\mathbf{z}_a^{(i)}, \mathbf{z}_n^{(j)}\right) + m\Big). \end{aligned} \tag{7}$$

The objective of (7) is to ensure that the distance between negative and anchor embeddings (belonging to different classes j and i) is equal to or larger than the distance between positive and anchor embeddings (belonging to the same class i) plus a margin m.

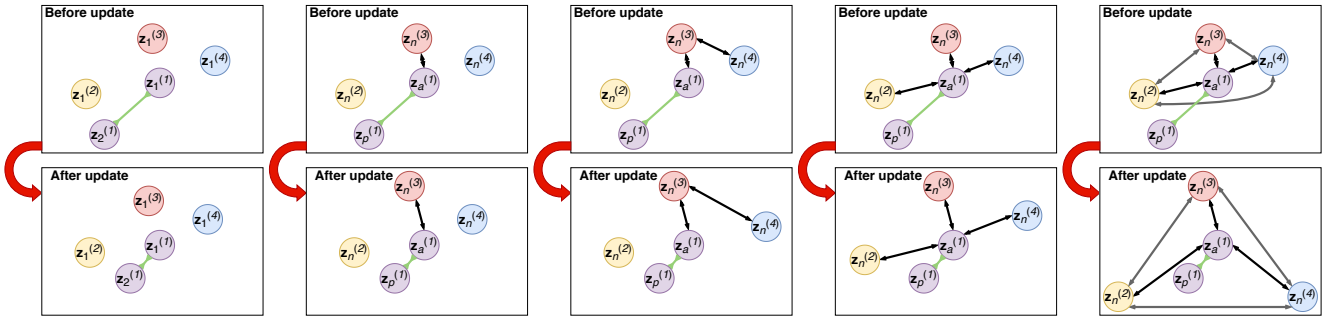
In practice, setting a margin m involves some heuristics, and setting a bad one might yield poor local optima [28]. To avoid this issue, in [28], it is proposed to use the softplus function, $\log(1+\exp\{\cdot\})$, as an approximation to the function $\max(0,\ \cdot)$. Thus, triplet loss can be simply rewritten as

$$\begin{aligned}\mathcal{L}_{\text{Triplet}} \cong \log\Big(1 + \exp\Big\{ & \mathcal{D}\left(\mathbf{z}_a^{(i)}, \mathbf{z}_p^{(i)}\right) \\ & - \mathcal{D}\left(\mathbf{z}_a^{(i)}, \mathbf{z}_n^{(j)}\right)\Big\}\Big). \end{aligned} \tag{8}$$

This variant is shown to outperform (7) for person re-identification [28].

Furthermore, to improve the generalization ability of triplet loss, quadruplet loss was proposed in [29]. Let $\mathbf{z}_n^{(k)}$ be another negative embedding from a third class k, where $k \neq i$ and

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/TASLP.2021.3092567, IEEE/ACM Transactions on Audio, Speech, and Language Processing



(a) Contrastive loss (b) Triplet loss (c) Quadruplet loss (d) N-pair loss (e) $(C_{N,2}+1)$-pair loss

Fig. 2: An illustrative comparison among different tuple-based loss functions when $N = 4$.

$k \neq j$. Using the softplus function, this 4-tuple loss function, $\mathcal{L}_{\text{Quadruplet}}$, might be expressed as

$$\mathcal{L}_{\text{Quadruplet}} = \log\left(1 + \exp\left\{\mathcal{D}\left(\mathbf{z}_a^{(i)}, \mathbf{z}_p^{(i)}\right) - \mathcal{D}\left(\mathbf{z}_a^{(i)}, \mathbf{z}_n^{(j)}\right)\right\} + \exp\left\{\mathcal{D}\left(\mathbf{z}_a^{(i)}, \mathbf{z}_p^{(i)}\right) - \mathcal{D}\left(\mathbf{z}_n^{(k)}, \mathbf{z}_n^{(j)}\right)\right\}\right). \quad (9)$$

The second exponential term of (9) helps to achieve larger inter-class and smaller intra-class variation compared to triplet loss, thus improving the generalization ability of the embedding extractor [29].

A major problem with all of the tuple-based loss functions presented above is that, depending on the value of N, there might be no interaction among all the N classes in each update of the network weights. Hence, frameworks for training of deep models using these loss functions might experience slow convergence and poor local optima [20]. Also, these loss functions may require expensive sample mining to provide non-trivial tuples in order to accelerate the training. To address these limitations, the N-pair loss was proposed in [20]:

$$\mathcal{L}_{N\text{-pair}} = \log\left(1 + \sum_{\substack{j=1 \\ j\neq i}}^{N} \exp\left\{\mathcal{D}\left(\mathbf{z}_a^{(i)}, \mathbf{z}_p^{(i)}\right) - \mathcal{D}\left(\mathbf{z}_a^{(i)}, \mathbf{z}_n^{(j)}\right)\right\}\right). \quad (10)$$

As can be observed from (10), given an anchor example, N-pair loss is intended to identify a positive example represented by $\mathbf{z}_p^{(i)}$ out of $N-1$ negative examples, each from a different class, represented by $\left\{\mathbf{z}_n^{(j)};\ 1 \leq j \leq N;\ j \neq i\right\}$. Notice that when $N = 2$, Eq. (10) is equivalent to triplet loss as in (8). This N-pair loss function has been successfully applied to tasks like image recognition and verification [20], and context-aware recommendation [30].

B. Proposed Training Loss Function

In N-pair loss, $N-1$ negative examples, each from a different class, are pushed away from the anchor example in the embedding space. However, the way in which these negative examples relate to each other in terms of distance is not under control. Evidently, we would like them to be as distant each other as possible. Therefore, we propose a $(C_{N,2}+1)$-pair loss that also looks for maximizing the distance among the $N-1$ negative examples as

$$\mathcal{L}'_{(C_{N,2}+1)\text{-pair}} = \log\left(1 + \exp\left\{\mathcal{D}\left(\mathbf{z}_a^{(i)}, \mathbf{z}_p^{(i)}\right) - \lambda \sum_{\substack{j=1 \\ j\neq i}}^{N}\left[\mathcal{D}\left(\mathbf{z}_a^{(i)}, \mathbf{z}_n^{(j)}\right) + \sum_{\substack{k>j \\ k\neq i}}^{N} \mathcal{D}\left(\mathbf{z}_n^{(j)}, \mathbf{z}_n^{(k)}\right)\right]\right\}\right), \quad (11)$$

where $C_{N,2} = N(N-1)/2$ is a binomial coefficient accounting for the total number of different negative pairs, and $\lambda > 0$ is a hyperparameter balancing the importance of the negative pairs distances with respect to the positive pairs distances.

Figure 2 shows an illustrative comparison between our proposal and the tuple-based loss functions briefly reviewed in Subsection III-A when $N = 4$. As in the case of quadruplet loss in relation to triplet loss, we hypothesize that our proposal might achieve larger inter-class and smaller intra-class variation compared to N-pair loss in Figure 2d. This, in turn, could improve the generalization ability of the embedding extractor.

It is important to set a proper value for the hyperparameter λ in (11). For example, if the value of λ is too small, embeddings will tend to collapse into a single point during training, thus yielding a totally useless solution. We hypothesize that this is because learning gets just focused on the minimization of the positive pairs distances. We found it effective (and, to some extent, fair for comparison purposes) to set λ in such a manner that the importance of $\mathcal{D}\left(\mathbf{z}_a^{(i)}, \mathbf{z}_p^{(i)}\right)$ is equal to that of the set of distances $\left\{\mathcal{D}\left(\mathbf{z}_a^{(i)}, \mathbf{z}_n^{(j)}\right);\ 1 \leq j \leq N;\ j \neq i\right\}$, as in N-pair loss (see Eq. (10)). In other words,

$$\lambda = \frac{1}{N-1}. \quad (12)$$

From (11) and (12), it is easy to show that the importance (i.e., weight) of the set of distances among negative embeddings $\left\{\mathcal{D}\left(\mathbf{z}_n^{(j)}, \mathbf{z}_n^{(k)}\right);\ 1 \leq j \leq N-1;\ j+1 \leq k \leq N;\ j,k \neq i\right\}$ with respect to $\mathcal{D}\left(\mathbf{z}_a^{(i)}, \mathbf{z}_p^{(i)}\right)$ increases linearly with N at a rate of $\lambda C_{N-1,2} = (N-2)/2$, where

$C_{N-1,2} = (N-1)(N-2)/2$ accounts for the number of distances in the above set.

A common practice in deep metric learning is to apply ℓ^2-norm normalization to embeddings so they are constrained to be on a hypersphere of radius 1, e.g., [19], [31]. Preliminary experiments using the N-pair loss function with and without embedding normalization revealed no statistically significant differences in terms of KWS performance. Nevertheless, the application of ℓ^2-norm normalization to embeddings will facilitate further calibration of (11) as explained below. Hence, in this work, instead of $\mathbf{z}$, we will always employ its normalized version

$$\bar{\mathbf{z}} = \frac{\mathbf{z}}{\|\mathbf{z}\|_2}, \tag{13}$$

where $\|\cdot\|_2$ is the ℓ^2-norm operator.

Finally, let us analyze the range of the argument of the exponential term in (11). Because embeddings are now constrained to be on a hypersphere of radius 1, it is clear that the distance between any two embeddings $\bar{\mathbf{z}}_1$ and $\bar{\mathbf{z}}_2$ is within certain narrower limits. More in particular, it can be stated that, regardless of the distance function, $\mathcal{D}(\bar{\mathbf{z}}_1, \bar{\mathbf{z}}_2) \in [0,\ d_{\max}]$, where $d_{\max}$ is the maximum distance possible. For example, $d_{\max}$ is 2 for the Euclidean distance, and π, for the great-circle distance [32]. With this in mind, and taking into account (12), it is straightforward to realize that a theoretical range of the argument of the exponential term in (11) is

$$[-\lambda C_{N,2} d_{\max},\ d_{\max}] = \left[-\frac{N d_{\max}}{2},\ d_{\max}\right]. \tag{14}$$

A problem with (14) is that, as N increases, the argument of the exponential term in (11) can be more negative, causing the value of the exponential term to approach 0. This, in turn, would cause the optimization of the embedding extractor to take place along a rather flat section of the loss curve close to the minimum loss value possible, i.e., 0. This might yield poor local optima. Therefore, to avoid this problem, we shift the optimization section along the loss curve by simply adding a constant $\mathcal{K}$ to the argument of the exponential term in (11). This constant is designed in such a manner that the lower bound of the new argument of the exponential term in (11) is equal to the lower bound of the argument of the exponential term in N-pair loss, that is, $-d_{\max}$ (see Eq. (10)). Thus,

$$\mathcal{K} = \frac{(N-2) d_{\max}}{2}. \tag{15}$$

With all of these elaborations, namely, considering Eqs. (12), (13) and (15), our proposed tuple-based loss, $\mathcal{L}_{(C_{N,2}+1)\text{-pair}}$, can be finally expressed as in Eq. (16). Despite its apparent higher complexity in relation to the N-pair loss, Eq. (16), just like (10), also needs an $(N+1)$-tuple as input. Moreover, while our proposal requires the computation of $C_{N-1,2} = (N-1)(N-2)/2$ and $(N^2-5N+8)/2$ more distances and additions, respectively, with respect to the N-pair loss, the latter involves the calculation of $N-2$ more exponentiations than our tuple-based loss.

In Section V, $\mathcal{L}_{(C_{N,2}+1)\text{-pair}}$ is shown to be superior, in terms of both KWS performance and speed of convergence, to the other related state-of-the-art tuple-based loss functions reviewed in this paper.

C. Distance Function

Let $\bar{\mathbf{z}}_1$ and $\bar{\mathbf{z}}_2$ be any two normalized embeddings. In principle, any (symmetric) distance function $\mathcal{D}(\bar{\mathbf{z}}_1, \bar{\mathbf{z}}_2) : \mathbb{R}^D \times \mathbb{R}^D \rightarrow [0,\ d_{\max}]$ could be used by the different tuple-based loss functions described above.

Preliminary experiments using the N-pair loss were conducted to select a distance function for further experimentation. First, because embeddings are constrained to be on a hypersphere of radius 1, we found it reasonable from a geometric point of view to assess the performance provided by the great-circle distance [32]:

$$\mathcal{D}_{\text{GC}}(\bar{\mathbf{z}}_1, \bar{\mathbf{z}}_2) = \arccos\left(\bar{\mathbf{z}}_1^\top \bar{\mathbf{z}}_2\right), \tag{17}$$

which can be interpreted as the inverse of the cosine similarity $S_c(\mathbf{z}_1, \mathbf{z}_2)$ [33]:

$$\begin{aligned} S_c(\mathbf{z}_1, \mathbf{z}_2) &= \frac{\mathbf{z}_1^\top \mathbf{z}_2}{\|\mathbf{z}_1\|_2 \|\mathbf{z}_2\|_2} = \bar{\mathbf{z}}_1^\top \bar{\mathbf{z}}_2 \\ &= \cos\left(\mathcal{D}_{\text{GC}}(\bar{\mathbf{z}}_1, \bar{\mathbf{z}}_2)\right). \end{aligned} \tag{18}$$

In spite of making sense to use the great-circle distance from a geometric perspective, employing (17) in preliminary experiments yielded embeddings collapsing into a single point during training.

We also compared the use of both the Euclidean distance and its squared version, commonly considered in the deep metric learning literature, e.g., [19], [28], [31]. In this case, no embedding collapsing phenomena were noticed, as well as better KWS performance was observed when utilizing the Euclidean distance

$$\mathcal{D}_E(\bar{\mathbf{z}}_1, \bar{\mathbf{z}}_2) = \|\bar{\mathbf{z}}_1 - \bar{\mathbf{z}}_2\|_2 \tag{19}$$

instead of its squared version. Taking into account that $\mathcal{D}_E(\bar{\mathbf{z}}_1, \bar{\mathbf{z}}_2) \in [0,\ d_{\max} = 2]$, that might be related to the fact that $\mathcal{D}_E^2(\bar{\mathbf{z}}_1, \bar{\mathbf{z}}_2)$ overestimates actual (non-squared) Euclidean distances above 1 while underestimating them when $\mathcal{D}_E(\bar{\mathbf{z}}_1, \bar{\mathbf{z}}_2) < 1$. Thus, the use of $\mathcal{D}_E^2(\bar{\mathbf{z}}_1, \bar{\mathbf{z}}_2)$ instead of $\mathcal{D}_E(\bar{\mathbf{z}}_1, \bar{\mathbf{z}}_2)$ might produce less inter-class separation and higher intra-class dispersion (see Subsection V-B for an experimental validation of this). As a result, all the tuple-based loss functions evaluated in Section V integrate the Euclidean distance as in (19).

D. Mining Strategy

It is well-known that, for training, mining non-trivial tuples —that is, informative tuples that make the loss function "sufficiently" greater than zero— can be of importance to accelerate convergence and improve performance [34]. For instance, in [28], the online mining strategy so-called *Batch Hard*, already outlined in [19], provides the best person re-identification performance among several relevant mining approaches. In short, *Batch Hard* consists of the selection, on a mini-batch basis, of the hardest positive and negative samples in terms of embedding distance with respect to each of the training samples in the mini-batch in order to form (hard) tuples for training. Our preliminary experiments using N-pair loss along with *Batch Hard* also yielded embeddings collapsing into a

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/TASLP.2021.3092567, IEEE/ACM Transactions on Audio, Speech, and Language Processing

$$\mathcal{L}_{(C_{N,2}+1)\text{-pair}} = \log\left(1+\exp\left\{\mathcal{D}\left(\bar{\mathbf{z}}_a^{(i)},\bar{\mathbf{z}}_p^{(i)}\right) - \frac{1}{N-1}\sum_{\substack{j=1\\ j\neq i}}^{N}\left[\mathcal{D}\left(\bar{\mathbf{z}}_a^{(i)},\bar{\mathbf{z}}_n^{(j)}\right) + \sum_{\substack{k>j\\ k\neq i}}^{N}\mathcal{D}\left(\bar{\mathbf{z}}_n^{(j)},\bar{\mathbf{z}}_n^{(k)}\right)\right] + \frac{(N-2)d_{\max}}{2}\right\}\right) \tag{16}$$

TABLE I: Noisy conditions (i.e., combinations of noise type and SNR level) covered by the training and validation sets (*mid gray*), and by the test set (*light gray*). Intersection (*only in terms of noisy conditions, and being the noise realizations different across sets*) between the training and validation sets and the test set is indicated in *dark gray*.

Noise type	**SNR (dB)**							
	-10	-5	0	5	10	15	20	Clean
White noise								
Babble								
Machine gun								
F-16 cockpit								
Vehicle interior								
Factory_1								
Bus								
Pedestrian street								
Factory_2								
Buccaneer jet cockpit								
Café								
Street junction								

TRAINING AND VALIDATION SETS

TEST SET

single point during an early stage of the training. This fact was already observed in previous work when using hard tuples for training [19], [28], and it may be related to hard tuples producing large attractive (small repelling) gradients from hard positive (negative) pairs [35].

To address the above issue, we relaxed *Batch Hard* as follows. For each of the training samples in the mini-batch, a hard or a randomly-arranged tuple is formed with a probability of σ and $1-\sigma$, respectively, where the value of σ is increased as the training loss decreases. While in this way we are able to bridge the critical point in relation to the aforementioned embedding collapsing during an early stage of the training, we also noticed that, as a trend, the lower the value of σ, the better the KWS performance. As a consequence, for all the tuple-based loss functions tested in Section V, we adopt a straightforward, yet very effective mining strategy consisting of randomly arranging tuples, on an epoch basis, for each of the samples in the training set (i.e., avoiding *Batch Hard* altogether).

IV. Experimental Setting

A. Experimental Database

For experimental purposes, a noisy speech corpus was created from the Google Speech Commands Dataset (GSCD) [24], which is a speech database comprising one-second long utterances, each containing one word out of 35 different possible words. Words can be anywhere within the one-second long audio segments, namely, words are not evenly segmented. The GSCD was recorded by 2,618 speakers.

To create our experimental corpus, first, clean speech utterances of the GSCD were identified from the inherently noisy GSCD recordings. On an utterance basis, noise power was estimated from the first 100 ms (1,600 samples at a sampling rate of 16 kHz) of the signal. Then, the remaining signal samples were employed to calculate the noisy speech power. From the ratio of these two power values, the *a posteriori* SNR was obtained, and the corresponding utterance was considered to be clean if and only if the *a posteriori* SNR was greater than 40 dB. This was done for the training, validation and test sets originally defined in the GSCD to define equivalent clean training, validation and test sets.

Furthermore, the clean training, validation and test sets were split into eight different partitions each. As in the classical AURORA-2 database [36], each of these partitions was assigned a type of noise from either NOISEX-92 [37] or CHiME-3 [38], [39]. Training and validation sets noises are vehicle interior, factory_1, bus, pedestrian street, white noise, babble, machine gun and F-16 cockpit. Test set noises are vehicle interior, factory_1, bus, pedestrian street, factory_2, Buccaneer jet cockpit, café and street junction[2].

Each of the above clean speech partitions was contaminated with a particular noise type at different SNR levels[3] by means of Filtering and Noise-adding Tool (FaNT) [40]. For both the training and validation sets, the considered SNR values were, apart from the clean case, $\{0, 5, 10, 15, 20\}$ dB. For the test set, $\{-10, -5, 0, 5, 10, 15, 20\}$ dB were the chosen SNR values apart from the clean case as well. Thus, each noisy condition, i.e., each combination of noise type and SNR level, is *evenly* represented by 3,699, 427 and 497 utterances in the training, validation and test sets, respectively. As a summary, Table I allows us to see at a glance the different noisy conditions covered by the training, validation and test sets.

The necessary tools to generate this noisy speech corpus are publicly available[4].

B. Keyword Spotting Model

To test our proposal and comparison techniques, we use the state-of-the-art deep residual learning model with dilated convolutions (architecture `res15`) reported in [14]. This model is fed with log-Mel spectrograms computed from one-second long speech segments using a 30 ms Hann window with a 10 ms shift. Log-Mel spectrograms are normalized to have zero mean and unit standard deviation by employing global sample mean and standard deviation values calculated from the training set. These log-Mel spectrograms are comprised of $K = 40$ frequency bins and $T = 101$ time frames. In addition,

[2]Specifically, bus, pedestrian street, café and street junction noises are from CHiME-3.

[3]In other words, the corpus comprises as many distorted copies of every clean speech utterance as considered SNR levels. Indeed, noise realizations are different across both utterances and SNR levels.

[4]`http://ilopez.es.mialias.net/misc/NoisyGSCD.zip`

the output of the global average pooling layer of `res15` prior to the final fully-connected layer with softmax activation is interpreted as a $D = 45$-dimensional keyword embedding $\mathbf{z}$.

This model is trained, as in [16]–[18], to spot the following 10 keywords: "yes", "no", "up", "down", "left", "right", "on", "off", "stop" and "go". Non-keywords are expected to be classified into the filler class, which is comprised of the remaining 25 words of the GSCD. Thus, $N = 11$ in this paper. All the $N = 11$ classes are approximately balanced across the training, validation and test sets.

To train the deep residual learning model either end-to-end or piecewise —that is, as proposed, first, the keyword embedding extractor, and then, the linear classifier—, the following setup is considered. As an optimizer, Adam [41] with default parameters (i.e., the learning rate is 0.001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$) is used. As a regularization method, early-stopping [42] with a patience of 5 epochs is employed. The size of the mini-batch is 16 samples/tuples. Practical implementation was carried out with Keras [43] running on top of TensorFlow [44].

V. Results and Discussion

In this section, our proposal and comparison techniques are evaluated in terms of KWS performance and training convergence speed. Accuracy, which is the ratio between the number of correct predictions and the total number of them, is employed as the primary evaluation metric in this work. For accuracy computation, a prediction is made by simply picking the most likely class as indicated by Eq. (3). Accuracy results are presented along with 95% confidence intervals from the Student's t-distribution [45]. To obtain confidence intervals, $\mathsf{n} = 5$ different neural network models considering different random weight initialization are trained for each of the reported experiments. Let μ_{acc} and σ_{acc} be, respectively, the sample mean and standard deviation values calculated from the resulting $\mathsf{n} = 5$ accuracy values from a particular experiment, we can define confidence intervals as in [45]

$$\left[\mu_{\text{acc}} - t_{0.025,\mathsf{n}-1}\frac{\sigma_{\text{acc}}}{\sqrt{\mathsf{n}}},\ \mu_{\text{acc}} + t_{0.025,\mathsf{n}-1}\frac{\sigma_{\text{acc}}}{\sqrt{\mathsf{n}}}\right], \tag{20}$$

where $t_{0.025,\mathsf{n}-1} \approx 2.78$ is the 97.5$^{\text{th}}$ percentile of the Student's t-distribution with $\nu = \mathsf{n} - 1$ degrees of freedom.

Apart from accuracy measurements, detection error trade-off (DET) curves are also presented. These curves represent pairs of false alarm and false reject rates as a function of the sensitivity threshold (swept from 0 to 1) setting the minimum posterior probability value from which a keyword is detected. DET curves are calculated for each of the keywords and noisy conditions, and, then, they are averaged. The smaller the area under a DET curve, the better a system is.

Moreover, Subsection V-C shows streaming KWS results comprising the processing of a continuous stream of audio data in which the different word classes are rather unbalanced. Due to the latter, accuracy is not a good metric in such circumstances, and, therefore, F-score curves as a function of the sensitivity threshold are presented instead. F-score, F_1, is defined as the harmonic mean of precision and recall [46], namely,

$$F_1 = \frac{2}{\text{Recall}^{-1} + \text{Precision}^{-1}} = \frac{2\text{TP}}{2\text{TP} + \text{FN} + \text{FP}}, \tag{21}$$

where $0 \leq F_1 \leq 1$, and TP, FN and FP denote the number of true positives, false negatives and false positives, respectively. Similarly to the case of the DET curves, F-score curves are computed for each of the keywords and noisy conditions, and, then, they are averaged. The larger the area under an F-score curve, the better a system is.

A. Evaluated Techniques

We define our baseline system as the state-of-the-art deep residual learning KWS system of [14] (architecture `res15`), which is end-to-end, multi-condition trained by means of cross-entropy loss. We compare the baseline to systems based on the `res15` architecture, trained using the different tuple-based loss functions described in Section III: contrastive loss, triplet loss, quadruplet loss, N-pair loss and $(C_{N,2}+1)$-pair loss. It should be noticed that, for experimental purposes, all of these tuple-based loss functions incorporate the softplus function. Therefore, while triplet loss is as in (8), contrastive loss needs to be approximated as follows:

$$\begin{aligned}\mathcal{L}_{\text{Contrastive}} \cong\ & \delta_{ij}\log\left(1+\exp\left\{\mathcal{D}\left(\mathbf{z}_1^{(i)},\mathbf{z}_2^{(j)}\right)\right\}\right) \\ & + (1-\delta_{ij})\log\left(1+\exp\left\{-\mathcal{D}\left(\mathbf{z}_1^{(i)},\mathbf{z}_2^{(j)}\right)\right\}\right).\end{aligned} \tag{22}$$

Let us also notice that some training trials using triplet and quadruplet loss functions yielded keyword embedding extractors getting stuck in poor local minima. Those triplet and quadruplet loss trials were replaced by other trials with "good convergence" to obtain the KWS performance results reported in this section.

B. Keyword Spotting Results

Keyword spotting accuracy results with 95% confidence intervals are reported in Table II. These results are broken down by SNR level, and *seen* (vehicle interior, factory$_1$, bus and pedestrian street) and *unseen* (factory$_2$, Buccaneer jet cockpit, café and street junction) *noises* during the training phase. First, accuracy results prove that the proposed training strategy integrating a tuple-based loss function is able to improve (except for Contrastive loss on *seen noises*) Baseline, the latter involving traditional end-to-end training exploiting cross-entropy loss. This is due to our training strategy allowing for more discriminative and noise-robust keyword embedding computation. In fact, we should remark that a series of experiments (not reported in this manuscript) suggests that our two-stage training strategy might be superior to a single-stage training approach considering a joint loss function as in, e.g., $\mathcal{L} = \omega_1\mathcal{L}_{\text{CE}} + \omega_2\mathcal{L}_{(C_{N,2}+1)\text{-pair}}$, where ω_1 and ω_2 are loss combination weights. Second, the general trend of these results reveals that the more examples from different classes a tuple-based loss function exploits in each update, the better the KWS performance is.

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/TASLP.2021.3092567, IEEE/ACM Transactions on Audio, Speech, and Language Processing

TABLE II: Keyword spotting accuracy results, in percentages, with 95% confidence intervals. Results are broken down by SNR level, and *seen* (vehicle interior, factory$_1$, bus and pedestrian street) and *unseen* (factory$_2$, Buccaneer jet cockpit, café and street junction) *noises* during the training phase.

		SNR (dB)								
		-10	-5	0	5	10	15	20	Clean	Average
Seen noises	Baseline	57.51 ± 0.96	75.94 ± 1.42	88.36 ± 1.30	92.90 ± 0.43	94.59 ± 0.75	95.17 ± 0.74	96.09 ± 0.72	96.45 ± 0.87	87.13 ± 0.60
	Contrastive loss	59.73 ± 1.46	75.07 ± 1.74	86.81 ± 0.83	92.56 ± 0.64	94.37 ± 0.29	95.14 ± 0.31	95.85 ± 0.80	96.72 ± 0.36	87.03 ± 0.65
	Triplet loss	60.12 ± 2.68	75.70 ± 2.07	86.96 ± 0.78	93.04 ± 0.72	93.86 ± 0.78	95.14 ± 0.87	96.01 ± 0.76	96.67 ± 0.42	87.19 ± 0.91
	Quadruplet loss	60.36 ± 1.47	78.00 ± 1.17	87.85 ± 1.17	92.80 ± 0.66	94.40 ± 0.90	95.58 ± 0.59	96.21 ± 0.48	96.88 ± 0.94	87.76 ± 0.68
	N-pair loss	60.07 ± 1.73	77.29 ± 0.95	88.72 ± 0.61	93.41 ± 0.50	95.00 ± 0.53	95.92 ± 0.22	96.62 ± 0.64	97.00 ± 0.44	88.00 ± 0.16
	$(C_{N,2}+1)$-pair loss	61.84 ± 1.73	78.00 ± 1.97	88.82 ± 0.41	93.48 ± 0.49	95.17 ± 0.82	95.94 ± 0.57	96.76 ± 0.56	96.91 ± 0.57	88.36 ± 0.72
Unseen noises	Baseline	35.11 ± 1.87	62.06 ± 1.40	82.20 ± 1.91	89.75 ± 1.18	92.87 ± 0.92	95.11 ± 1.14	96.11 ± 0.43	96.52 ± 0.64	81.22 ± 0.69
	Contrastive loss	38.98 ± 1.74	62.61 ± 2.71	81.19 ± 1.76	89.02 ± 0.87	92.50 ± 1.22	94.29 ± 0.43	95.38 ± 0.84	96.61 ± 0.32	81.32 ± 0.86
	Triplet loss	37.78 ± 2.44	62.44 ± 1.14	81.84 ± 1.57	90.13 ± 0.57	93.45 ± 0.47	95.28 ± 0.37	96.01 ± 0.58	96.74 ± 0.59	81.71 ± 0.52
	Quadruplet loss	38.26 ± 0.95	63.94 ± 1.37	83.02 ± 1.03	90.11 ± 0.70	93.88 ± 0.61	95.26 ± 0.44	95.94 ± 0.59	96.78 ± 0.48	82.15 ± 0.27
	N-pair loss	39.93 ± 2.54	64.86 ± 1.89	83.92 ± 0.68	90.50 ± 0.27	93.81 ± 0.64	95.28 ± 0.50	96.30 ± 0.64	96.98 ± 0.54	82.70 ± 0.52
	$(C_{N,2}+1)$-pair loss	41.21 ± 2.63	65.27 ± 1.10	84.67 ± 0.99	92.26 ± 0.15	94.70 ± 0.48	96.20 ± 0.87	96.71 ± 0.49	97.19 ± 0.59	83.53 ± 0.30

TABLE III: Mean intra- and inter-class Euclidean distances with their associated variances on the training and test sets. Distances on the test set are broken down by *seen* and *unseen noises* during the training phase.

	Intra-class distance			*Inter-class distance*		
	Training	Test: *Seen noises*	Test: *Unseen noises*	Training	Test: *Seen noises*	Test: *Unseen noises*
Baseline	0.327 ± 0.033	0.423 ± 0.087	0.488 ± 0.103	1.272 ± 0.031	1.234 ± 0.032	1.197 ± 0.032
Contrastive loss	0.084 ± 0.059	0.235 ± 0.208	0.324 ± 0.273	1.482 ± 0.001	1.480 ± 0.001	1.477 ± 0.001
Triplet loss	0.031 ± 0.014	0.199 ± 0.191	0.280 ± 0.255	1.471 ± 0.036	1.470 ± 0.036	1.469 ± 0.036
Quadruplet loss	0.030 ± 0.014	0.196 ± 0.186	0.278 ± 0.253	1.473 ± 0.028	1.472 ± 0.028	1.471 ± 0.028
N-pair loss ($\mathcal{D}_E^2(\bar{\mathbf{z}}_1, \bar{\mathbf{z}}_2)$)	0.052 ± 0.015	0.236 ± 0.195	0.325 ± 0.260	1.482 ± 0.000	1.478 ± 0.000	1.474 ± 0.000
N-pair loss	0.023 ± 0.015	0.196 ± 0.199	0.278 ± 0.269	1.481 ± 0.005	1.481 ± 0.005	1.480 ± 0.005
$(C_{N,2}+1)$-pair loss	0.018 ± 0.012	0.191 ± 0.204	0.268 ± 0.272	1.483 ± 0.000	1.483 ± 0.000	1.482 ± 0.000

According to Table II, while, in general, there are no statistically significant differences among the average accuracy results for *seen noises*, the best performance for *unseen noises* is clearly obtained by the proposed method (∼83.53% acc.) in a statistically significant manner. In particular, on average, our training strategy using the $(C_{N,2}+1)$-pair loss function yields around 12% KWS accuracy relative improvement with respect to Baseline.

Also from Table II, we can see that, overall, $(C_{N,2}+1)$-pair loss provides better performance than N-pair loss, especially for *unseen noises*. As already discussed in Subsection III-B, we hypothesize that our proposal might obtain larger inter-class and smaller intra-class variation compared to N-pair loss thanks to that the former also looks for maximizing the distance among the $N-1$ negative examples of each tuple. In this way, the generalization ability of the keyword embedding extractor may be improved, and, as a result, the KWS performance.

The above hypothesis is supported by Table III. This table reports mean intra- and inter-class Euclidean distances with their associated variances on the training and test (broken down by *seen* and *unseen noises*) sets. Distance values are derived with respect to median-estimated class centroids. In addition, Baseline distances are calculated from (ℓ^2-norm) normalized keyword embeddings for the sake of comparison. Numbers in Tables II and III are highly correlated, which comes as no surprise. Thus, distance values in Table III reinforce the utility of the proposed training strategy incorporating a tuple-based loss function to achieve more discriminative and noise-robust keyword embeddings in comparison with Baseline. In particular, the $(C_{N,2}+1)$-pair loss function helps for obtaining, on all of the reported sets (i.e., training, test-*seen noises* and test-*unseen noises*) and among all of the tested techniques, the largest inter-class distances and the shortest intra-class distances, being particularly remarkable the low training set intra-class value.

Besides, as mentioned in Subsection III-C, preliminary experiments using N-pair loss revealed that better KWS performance could be achieved by utilizing the actual (non-squared) Euclidean distance $\mathcal{D}_E(\bar{\mathbf{z}}_1, \bar{\mathbf{z}}_2)$ instead of its squared version $\mathcal{D}_E^2(\bar{\mathbf{z}}_1, \bar{\mathbf{z}}_2)$. We hypothesize that this fact might be related to $\mathcal{D}_E^2(\bar{\mathbf{z}}_1, \bar{\mathbf{z}}_2)$ overestimating actual Euclidean distances above 1 while underestimating them when these are less than 1 (recall that $\mathcal{D}_E(\bar{\mathbf{z}}_1, \bar{\mathbf{z}}_2) \in [0,\ d_{\max}=2]$). Hence, using the squared Euclidean distance instead of the non-squared one would yield less inter-class separation and higher intra-class dispersion. This hypothesis is very much endorsed by Table III, where, as can be seen, using $\mathcal{D}_E^2(\bar{\mathbf{z}}_1, \bar{\mathbf{z}}_2)$ instead of $\mathcal{D}_E(\bar{\mathbf{z}}_1, \bar{\mathbf{z}}_2)$ clearly yields larger mean intra-class distance on the different sets[5].

For illustrative purposes, Figure 3 shows a test set (*unseen noises* only) keyword embedding representation obtained by means of t-distributed Stochastic Neighbor Embedding (t-SNE) [47]. This figure suggests, in agreement with results of Tables II and III, better class definition and separability for our training strategy using the $(C_{N,2}+1)$-pair loss function than for the comparison techniques. This fact is also reflected by the KWS detection error trade-off curves plotted in Figure 4, in which Baseline shows a more competitive performance than the indicated by Tables II and III.

Finally, Table IV presents a comparison, in terms of both

[5]In line with this, N-pair loss ($\mathcal{D}_E^2(\bar{\mathbf{z}}_1, \bar{\mathbf{z}}_2)$) average KWS accuracy for *seen* and *unseen noises* is 87.71% ± 0.49 and 82.11% ± 0.54, respectively.

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/TASLP.2021.3092567, IEEE/ACM Transactions on Audio, Speech, and Language Processing

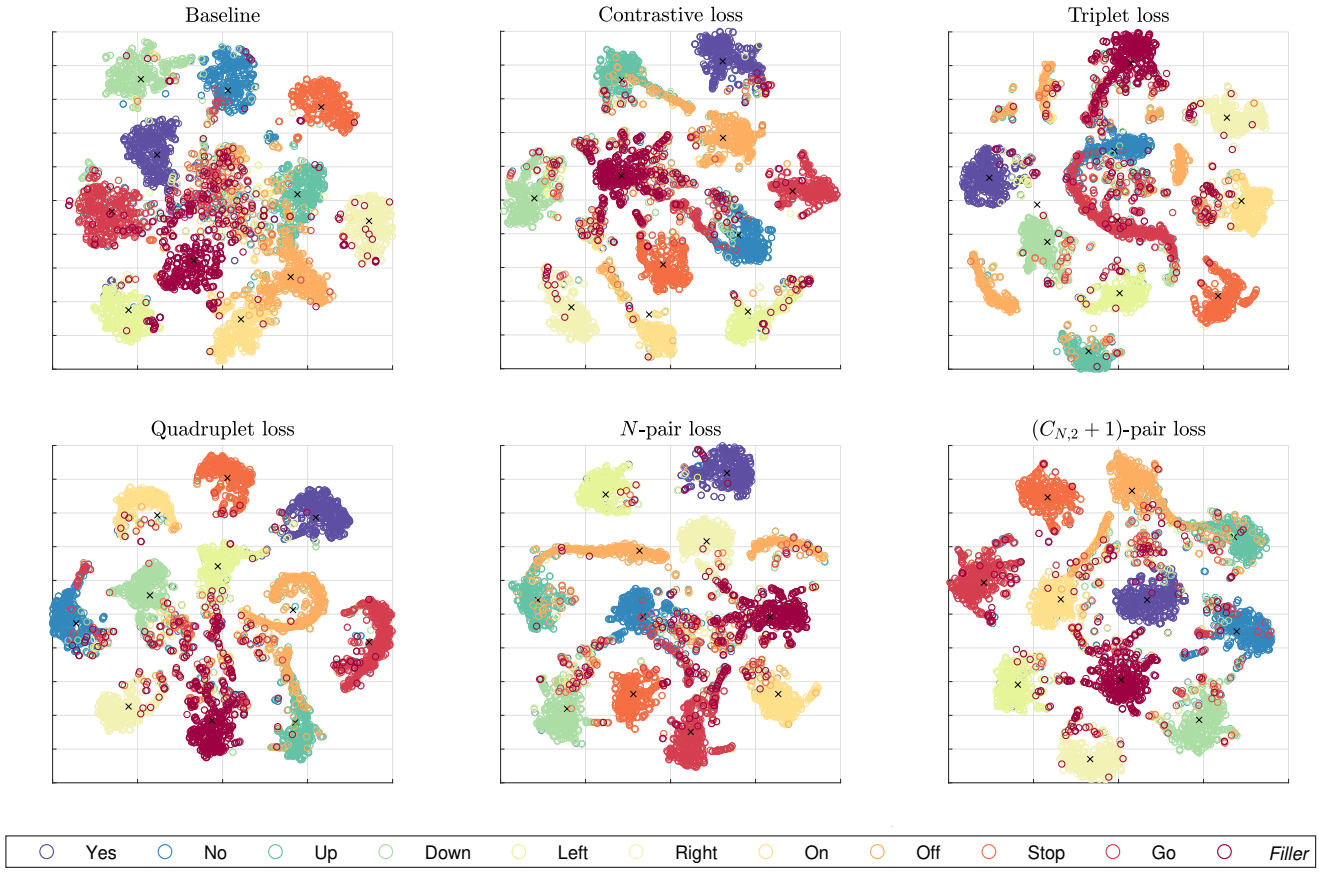


Fig. 3: Test set (*unseen noises* only) keyword embedding representation obtained by means of t-SNE [47]. Black crosses mark the location of the class centroids.

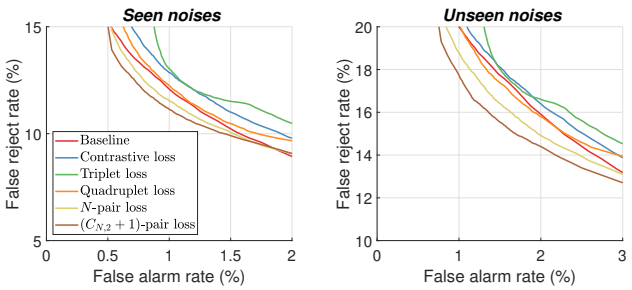


Fig. 4: Keyword spotting detection error trade-off curves for *seen* and *unseen noises* during the training phase.

accuracy and number of model parameters, between Baseline, our proposal (i.e., $(C_{N,2}+1)$-pair loss), and some of the most prominent KWS models/techniques when they are evaluated on the original GSCD. Such most prominent KWS models/techniques consist of *a)* a fully-connected, feedforward neural network (DNN [48]), *b)* a convolutional neural network with striding equal to 2 (CNN+strd [48]), *c)* a recurrent neural network with an attention mechanism [49] (Att-RNN [50]), *d)* a depthwise separable CNN with striding equal to 2 (DSCNN-strd [51]), *e)* a variant of TC-ResNet [15] (where TC stands for "temporal convolution"), and *f)* an extension of Att-RNN with multi-head attention (MHAtt-RNN [48]). Numbers reported for these models/techniques are from [48, Table 2], except for Att-RNN, which are from [50, Table 2]. From Table IV, we see that the application of our proposal to the Baseline deep residual learning model results in a very competitive position in the ranking taking also into account the number of model parameters.

TABLE IV: State-of-the-art KWS model/technique comparison, in terms of accuracy (%) and number of model parameters, on the original Google Speech Commands Dataset. Reported numbers of models/techniques with superscript * are from [48, Table 2]. Reported numbers of Att-RNN† are from [50, Table 2].

Model/technique	**Accuracy (%)**	**No. of parameters**
DNN*	90.6	447k
CNN+strd*	95.6	529k
Baseline	95.93 $\pm$ 0.74	238k
Att-RNN†	96.9	202k
DSCNN+strd*	97.0	485k
$(C_{N,2}+1)$-pair loss	97.18 $\pm$ 0.14	238k
TC-ResNet*	97.4	365k
MHAtt-RNN*	98.0	743k

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/TASLP.2021.3092567, IEEE/ACM Transactions on Audio, Speech, and Language Processing

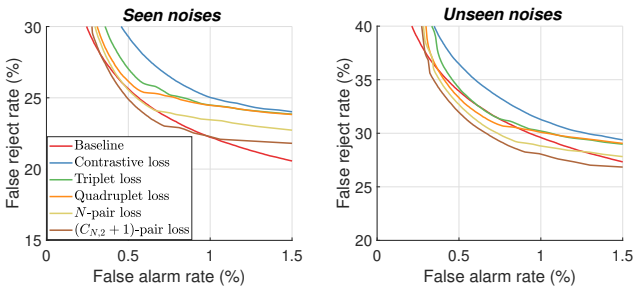


Fig. 5: Streaming KWS detection error trade-off curves for *seen* and *unseen noises* during the training phase.

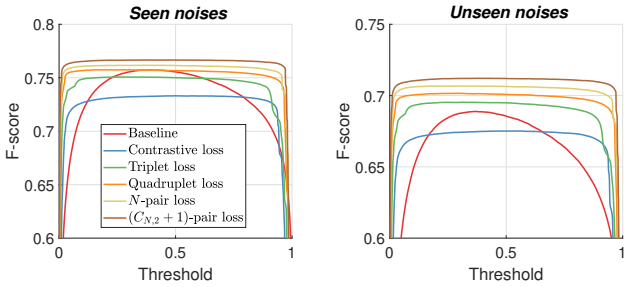


Fig. 6: Streaming KWS F-score curves for *seen* and *unseen noises* during the training phase.

C. Streaming Keyword Spotting

Better capturing the use of KWS in the real world, streaming KWS comprises the processing of a continuous stream of audio data in which the different word classes are rather unbalanced. To perform streaming KWS evaluations, we follow a procedure similar to the one described in [17]. First, we re-train the different KWS models to spot an additional silence/background noise class, so now $N = 12$. As before, all of the classes —including the silence/background noise class— are rather balanced for training. Second, a basis test audio stream is generated by concatenation of test clean speech and silence fragments. This basis audio stream contains around 150 utterances of each keyword type and near 2,000 non-keywords, which are randomly intermingled along the audio sequence. This great word class unbalancing better reflects what can be expected to observe in real-life applications. The duration of the basis test audio stream is, approximately, 1 hour and 23 minutes.

Then, again by means of FaNT, this audio stream is successively contaminated with the test set noises (i.e., vehicle interior, factory$_1$, bus, pedestrian street, factory$_2$, Buccaneer jet cockpit, café and street junction) at the test SNR levels $\{-10, -5, 0, 5, 10, 15, 20\}$ dB to generate a total of 8 noise types $\times$ 7 SNR levels $=$ 56 noisy audio sequences for testing, in addition to the basis clean stream.

At test time, the sequence of posterior probabilities $P(i|\mathbf{X}, \theta)$ $(i = 1, ..., N)$, resulting from employing a one-second long sliding window with a hop of 250 ms on the input stream, is processed as in [52].

Figure 5 depicts streaming KWS detection error trade-off curves. While Baseline again shows a very competitive performance in terms of DET, our training strategy exploiting the $(C_{N,2}+1)$-pair loss function stands out for *unseen noises* as a result of its greater generalization ability. Moreover, streaming KWS F-score curves are plotted by Figure 6. From this figure, we can confirm the usefulness of our training strategy to improve KWS performance, especially in noisy conditions not seen during the training phase. In addition, we can clearly note the aforementioned trend whereby the more examples from different classes a tuple-based loss function considers in each update, the better the KWS performance is. As can be observed, $(C_{N,2}+1)$-pair loss outperforms all of the related tuple-based loss functions.

TABLE V: Training times per epoch, in seconds, with 95% confidence intervals when using different loss functions to train the keyword embedding extractor.

Loss function	**Training time per epoch (s)**
Contrastive loss	396.8 $\pm$ 0.6
Triplet loss	578.1 $\pm$ 0.4
Quadruplet loss	774.3 $\pm$ 1.3
N-pair loss	2,247.0 $\pm$ 4.0
$(C_{N,2}+1)$-pair loss	2,268.8 $\pm$ 5.7

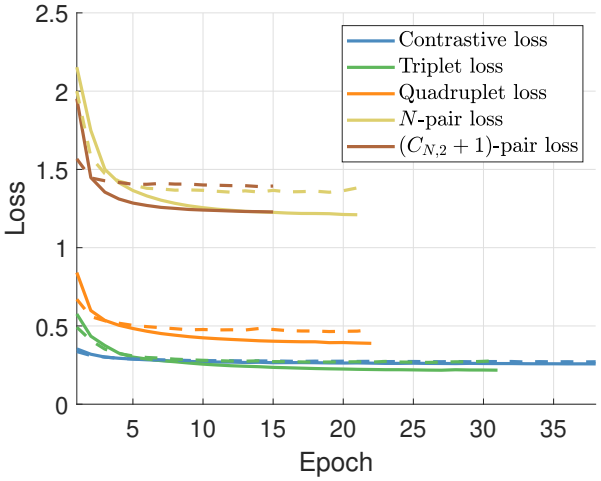


Fig. 7: Keyword embedding extractor training (solid lines) and validation (dashed lines) loss curves as a function of the epoch number. Loss curves were obtained by averaging across five trials.

D. Training Convergence Speed

Generally, fast training convergence of neural networks is something desirable. Hence, in this subsection we compare the $(C_{N,2}+1)$-pair loss function with the other evaluated tuple-based loss functions in terms of training convergence speed. Notice that all of the following results were obtained by averaging across five training trials that were run on a GPU NVIDIA GeForce GTX 1080 Ti.

First, Table V reports training time per epoch (in seconds) as a function of the different losses. This table shows an approximately linear relationship between the size of the tuple used by a loss function and its training time per epoch. In

TABLE VI: Keyword spotting accuracy results, in percentages, with 95% confidence intervals, from using `res15-narrow` [14] instead of `res15` on a noisy version of the Hey Snapdragon Keyword Dataset [53]. Results, averaged across SNR levels, are broken down by *seen* and *unseen noises* during the training phase.

Technique	*Seen noises*	*Unseen noises*
Baseline	89.90 ± 2.22	85.94 ± 3.22
Contrastive loss	93.06 ± 0.81	89.35 ± 0.81
Triplet loss	90.36 ± 1.19	85.98 ± 1.87
Quadruplet loss	92.98 ± 1.23	89.39 ± 1.44
N-pair loss	93.09 ± 0.92	90.24 ± 1.25
$(C_{N,2}+1)$-pair loss	93.37 ± 0.88	90.74 ± 0.78

addition, Figure 7 plots training and validation loss curves as a function of the epoch number. According to this figure, the more examples from different classes a loss function exploits in each update, the faster the training convergence is. This partially compensates for the larger training times per epoch associated to loss functions employing bigger tuples. Finally, it is worth mentioning that, since $\mathcal{L}_{(C_{N,2}+1)\text{-pair}}$ also aims at maximizing the distance among the $N-1$ negative examples of each tuple, $(C_{N,2}+1)$-pair loss provides faster convergence than N-pair loss. Since training times per epoch for N-pair and $(C_{N,2}+1)$-pair losses are very similar (see Table V), we can conclude that $(C_{N,2}+1)$-pair loss is able to yield superior KWS performance than N-pair loss with much less total training time.

E. On the Proposal Generalizability

To explore the generalizability potential of the proposed two-stage training strategy and $(C_{N,2}+1)$-pair loss, we also performed KWS evaluations on a different KWS corpus: Hey Snapdragon Keyword Dataset (HSKD) [53]. This corpus, recorded by 50 speakers, is comprised of 4,270 utterances, each covering one keyword out of 4 different possible keywords: "Hey Android", "Hey Snapdragon", "Hi Galaxy" and "Hi Lumina". Because this corpus is composed by just clean speech, we followed an equivalent procedure to that of Subsection IV-A to generate a noisy version of the HSKD. Furthermore, speech data from the Hey Snips database [54] were used to populate the filler (non-keyword) class. Notice that, this time, $N=5$, and, like for the (noisy) GSCD, all the $N=5$ classes are rather balanced in the training, validation and test sets. Speakers do not overlap across sets.

Due to the relatively small size of the noisy HSKD, we decided to use a lighter deep residual learning model, `res15-narrow` [14], instead of `res15` for experimental purposes. Model `res15-narrow` has nearly 6 times less parameters than `res15`, leading to $D=19$-dimensional keyword embeddings $\mathbf{z}$. On this occasion, `res15-narrow` is fed with log-Mel spectrograms computed from 1.5-second long speech segments (i.e., $T=151$ time frames). At training time, the size of the mini-batch is 32 samples/tuples. The rest of the setup is as indicated in Subsection IV-B.

Table VI shows the KWS accuracy results, in percentages and with 95% confidence intervals, corresponding to the experimental framework here described. These results, which are averaged across SNR levels, are broken down by *seen* and *unseen noises* during the training phase. With relatively narrow confidence intervals, our training strategy exploiting the $(C_{N,2}+1)$-pair loss function again yields the highest average accuracy for both *seen* and *unseen noises*, being remarkable its improvement over Baseline as well as the competitive performance of Contrastive loss (see Table VI).

VI. Concluding Remarks

In this work, we have developed a novel tuple-based loss function along with a training strategy for noise-robust keyword spotting. The proposed $(C_{N,2}+1)$-pair loss function extends the concept behind well-known tuple-based loss functions like triplet loss and N-pair loss to obtain larger inter-class and smaller intra-class variation, as we have experimentally demonstrated. As a result, we have been able to significantly improve the generalization ability of a modern KWS model compared to when the model is end-to-end, multi-condition trained using cross-entropy loss (which is a standard approach in the literature). It is worth emphasizing that this has been achieved by increasing neither the number of parameters nor the number of multiplications of the KWS model, which is of importance, since KWS systems are often intended for low-resource devices.

As future work, we will study the application of this training strategy exploiting $(C_{N,2}+1)$-pair loss to the mitigation of other types of distortions as well as to other tasks such as, for example, speaker recognition and image classification.

References

[1] M. Hoy, "Alexa, Siri, Cortana, and more: An introduction to voice assistants," *Medical Reference Services Quarterly*, vol. 37, pp. 81–88, 01 2018.

[2] Z. Zhang, J. Geiger, J. Pohjalainen, A. E.-D. Mousa, W. Jin, and B. Schuller, "Deep learning for environmentally robust speech recognition: An overview of recent developments," *ACM Transactions on Intelligent Systems and Technology*, vol. 9, pp. 1–28, 2018.

[3] M. Yu, X. Ji, Y. Gao, L. Chen, J. Chen, J. Zheng, D. Su, and D. Yu, "Text-dependent speech enhancement for small-footprint robust keyword detection," in *Proceedings of INTERSPEECH 2018 – 19th Annual Conference of the International Speech Communication Association, September 2-6, Hyderabad, India*, pp. 2613–2617, 2018.

[4] Y. Huang, T. Hughes, T. Z. Shabestary, and T. Applebaum, "Supervised noise reduction for multichannel keyword spotting," in *Proceedings of ICASSP 2018 – 43rd IEEE International Conference on Acoustics, Speech and Signal Processing, April 15-20, Calgary, Canada*, pp. 5474–5478, 2018.

[5] X. Ji, M. Yu, J. Chen, J. Zheng, D. Su, and D. Yu, "Integration of multi-look beamformers for multi-channel keyword spotting," in *Proceedings of ICASSP 2020 – 45th IEEE International Conference on Acoustics, Speech and Signal Processing, May 4-8, Barcelona, Spain*, pp. 7464–7468, 2020.

[6] Y. A. Huang, T. Z. Shabestary, and A. Gruenstein, "Hotword Cleaner: Dual-microphone adaptive noise cancellation with deferred filter coefficients for robust keyword spotting," in *Proceedings of ICASSP 2019 – 44th IEEE International Conference on Acoustics, Speech and Signal Processing, May 12-17, Brighton, UK*, pp. 6346–6350, 2019.

[7] J. Malek and J. Zdansky, "On practical aspects of multi-condition training based on augmentation for reverberation-/noise-robust speech recognition," *Lecture Notes in Computer Science*, vol. 11697, pp. 251–263, 2019.

[8] L. Mošner, M. Wu, A. Raju, S. H. K. Parthasarathi, K. Kumatani, S. Sundaram, R. Maas, and B. Hoffmeister, "Improving noise robustness of automatic speech recognition via parallel data and teacher-student learning," in *Proceedings of ICASSP 2019 – 44th IEEE International Conference on Acoustics, Speech and Signal Processing, May 12-17, Brighton, UK*, pp. 6475–6479, 2019.

[9] P. Wang, K. Tan, and D. Wang, "Bridging the gap between monaural speech enhancement and recognition with distortion-independent acoustic modeling," in *Proceedings of INTERSPEECH 2019 – 20th Annual Conference of the International Speech Communication Association, September 15-19, Graz, Austria*, pp. 471–475, 2019.

[10] Y. Wang, P. Getreuer, T. Hughes, R. F. Lyon, and R. A. Saurous, "Trainable frontend for robust and far-field keyword spotting," in *Proceedings of ICASSP 2017 – 42nd IEEE International Conference on Acoustics, Speech and Signal Processing, March 5-9, New Orleans, USA*, pp. 5670–5674, 2017.

[11] Y. Gao, Y. Mishchenko, A. Shah, S. Matsoukas, and S. Vitaladevuni, "Towards data-efficient modeling for wake word spotting," in *Proceedings of ICASSP 2020 – 45th IEEE International Conference on Acoustics, Speech and Signal Processing, May 4-8, Barcelona, Spain*, pp. 7479–7483, 2020.

[12] S. O. Arik, M. Kliegl, R. Child, J. Hestness, A. Gibiansky, C. Fougner, R. Prenger, and A. Coates, "Convolutional recurrent neural networks for small-footprint keyword spotting," in *Proceedings of INTERSPEECH 2017 – 18th Annual Conference of the International Speech Communication Association, August 20-24, Stockholm, Sweden*, pp. 1606–1610, 2017.

[13] C. Shan, J. Zhang, Y. Wang, and L. Xie, "Attention-based end-to-end models for small-footprint keyword spotting," in *Proceedings of INTERSPEECH 2018 – 19th Annual Conference of the International Speech Communication Association, September 2-6, Hyderabad, India*, pp. 2037–2041, 2018.

[14] R. Tang and J. Lin, "Deep residual learning for small-footprint keyword spotting," in *Proceedings of ICASSP 2018 – 43rd IEEE International Conference on Acoustics, Speech and Signal Processing, April 15-20, Calgary, Canada*, pp. 5484–5488, 2018.

[15] S. Choi, S. Seo, B. Shin, H. Byun, M. Kersner, B. Kim, D. Kim, and S. Ha, "Temporal convolution for real-time keyword spotting on mobile devices," in *Proceedings of INTERSPEECH 2019 – 20th Annual Conference of the International Speech Communication Association, September 15-19, Graz, Austria*, pp. 3372–3376, 2019.

[16] I. López-Espejo, Z.-H. Tan, and J. Jensen, "Keyword spotting for hearing assistive devices robust to external speakers," in *Proceedings of INTERSPEECH 2019 – 20th Annual Conference of the International Speech Communication Association, September 15-19, Graz, Austria*, pp. 3223–3227, 2019.

[17] I. López-Espejo, Z.-H. Tan, and J. Jensen, "Improved external speaker-robust keyword spotting for hearing assistive devices," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 1233–1247, 2020.

[18] I. López-Espejo, Z.-H. Tan, and J. Jensen, "Exploring filterbank learning for keyword spotting," in *Proceedings of EUSIPCO 2020 – 28th European Signal Processing Conference, January 18-21, Amsterdam, Netherlands*, 2021.

[19] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *Proceedings of CVPR 2015 – Conference on Computer Vision and Pattern Recognition, June 7-12, Boston, USA*, pp. 815–823, 2015.

[20] K. Sohn, "Improved deep metric learning with multi-class N-pair loss objective," in *Proceedings of NIPS 2016 – 30th International Conference on Neural Information Processing Systems, December 5-10, Barcelona, Spain*, pp. 1857–1865, 2016.

[21] N. Sacchi, A. Nanchen, M. Jaggi, and M. Cernak, "Open-vocabulary keyword spotting with audio and text embeddings," in *Proceedings of INTERSPEECH 2019 – 20th Annual Conference of the International Speech Communication Association, September 15-19, Graz, Austria*, pp. 3362–3366, 2019.

[22] Y. Yuan, Z. Lv, S. Huang, and L. Xie, "Verifying deep keyword spotting detection with acoustic word embeddings," in *Proceedings of ASRU 2019 – IEEE Automatic Speech Recognition and Understanding Workshop, December 14-18, Singapore, Singapore*, pp. 613–620, 2019.

[23] J. Huh, M. Lee, H. Heo, S. Mun, and J. S. Chung, "Metric learning for keyword spotting," in *Proceedings of SLT 2021 – IEEE Spoken Language Technology Workshop, January 19-22, Shenzhen, China*, pp. 133–140, 2021.

[24] P. Warden, "Speech Commands: A dataset for limited-vocabulary speech recognition," *arXiv:1804.03209v1*, 2018.

[25] J. S. Bridle, "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition," *Neurocomputing*, pp. 227–236, 1990.

[26] J. Lu, J. Hu, and J. Zhou, "Deep metric learning for visual understanding: An overview of recent advances," *IEEE Signal Processing Magazine*, vol. 34, pp. 76–84, 2017.

[27] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *Proceedings of CVPR 2005 – Conference on Computer Vision and Pattern Recognition, June 20-25, San Diego, USA*, pp. 539–546, 2005.

[28] A. Hermans, L. Beyer, and B. Leibe, "In defense of the triplet loss for person re-identification," *arXiv:1703.07737v4*, 2017.

[29] W. Chen, X. Chen, and J. Zhang, "Beyond triplet loss: A deep quadruplet network for person re-identification," in *Proceedings of CVPR 2017 – Conference on Computer Vision and Pattern Recognition, July 21-26, Honolulu, USA*, pp. 1320–1329, 2017.

[30] M. S. Kristoffersen, S. E. Shepstone, and Z.-H. Tan, "Context-aware recommendations for televisions using deep embeddings with relaxed N-pairs loss objective," *arXiv:2002.01554v1*, 2020.

[31] C. Zhang and K. Koishida, "End-to-end text-independent speaker verification with flexibility in utterance duration," in *Proceedings of ASRU 2017 – IEEE Automatic Speech Recognition and Understanding Workshop, December 16-20, Okinawa, Japan*, pp. 584–590, 2017.

[32] K. Gade, "A non-singular horizontal position representation," *The Journal of Navigation*, vol. 63, pp. 395–417, 2010.

[33] A. Singhal, "Modern information retrieval: A brief overview," *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, vol. 24, pp. 35–43, 2001.

[34] X. Wang, Y. Hua, E. Kodirov, G. Hu, and N. Robertson, "Deep metric learning by online soft mining and class-aware attention," in *Proceedings of AAAI 2019 – Conference on Artificial Intelligence, January 27-February 1, Honolulu, USA*, pp. 5361–5368, 2019.

[35] C.-Y. Wu, R. Manmatha, A. J. Smola, and P. Krahenbuhl, "Sampling matters in deep embedding learning," in *Proceedings of ICCV 2017 – International Conference on Computer Vision, October 22-29, Venice, Italy*, pp. 2840–2848, 2017.

[36] D. Pearce and H.-G. Hirsch, "The AURORA experimental framework for the performance evaluation of speech recognition systems under noisy conditions," in *Proceedings of ICSLP 2000 – 6th International Conference on Spoken Language Processing, October 16-20, Beijing, China*, 2000.

[37] A. Varga and H. J. Steeneken, "Assessment for automatic speech recognition: II. NOISEX-92: A database and an experiment to study the effect of additive noise on speech recognition systems," *Speech Communication*, vol. 12, pp. 247–251, 1993.

[38] J. Barker, R. Marxer, E. Vincent, and S. Watanabe, "The third 'CHiME' speech separation and recognition challenge: Dataset, task and baselines," in *Proceedings of ASRU 2015 – IEEE Automatic Speech Recognition and Understanding Workshop, December 13-17, Scottsdale, USA*, pp. 504–511, 2015.

[39] J. Barker, R. Marxer, E. Vincent, and S. Watanabe, "The third 'CHiME' speech separation and recognition challenge: Analysis and outcomes," *Computer Speech & Language*, vol. 46, pp. 605–626, 2017.

[40] H.-G. Hirsch, *FaNT - Filtering and noise adding tool*, 2005. https://github.com/i3thuan5/FaNT.

[41] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of ICLR 2015 – 3rd International Conference on Learning Representations, May 7-9, San Diego, USA*, 2015.

[42] N. Gershenfeld, "An experimentalist's introduction to the observation of dynamical systems," in *Directions in Chaos — Volume 2*, pp. 310–353, World Scientific, 1988.

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

[44] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[45] N. Blachman and R. Machol, "Confidence intervals based on one or more observations," *IEEE Transactions on Information Theory*, vol. 33, pp. 373–382, 1987.

[46] C. J. van Rijsbergen, *Information Retrieval*. Butterworth-Heinemann, 1979.

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

[48] O. Rybakov, N. Kononenko, N. Subrahmanya, M. Visontai, and S. Laurenzo, "Streaming keyword spotting on mobile devices," in *Proceedings of INTERSPEECH 2020 – 21st Annual Conference of the International Speech Communication Association, October 25-29, Shanghai, China*, pp. 2277–2281, 2020.

[49] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Łukasz Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of NIPS 2017 – 31st International Conference on Neural Information Processing Systems, December 4-9, Long Beach, USA*, pp. 5998–6008, 2017.

[50] D. C. de Andrade, S. Leo, M. L. D. S. Viana, and C. Bernkopf, "A neural attention model for speech command recognition," *arXiv:1808.08929v1*, 2018.

[51] Y. Zhang, N. Suda, L. Lai, and V. Chandra, "Hello edge: Keyword spotting on microcontrollers," *arXiv:1711.07128v3*, 2018.

[52] S. Fernández, A. Graves, and J. Schmidhuber, "An application of recurrent neural networks to discriminative keyword spotting," in *Proceedings of ICANN 2007 – 17th International Conference on Artificial Neural Networks, September 9-13, Porto, Portugal*, pp. 220–229, 2007.

[53] B. Kim, M. Lee, J. Lee, Y. Kim, and K. Hwang, "Query-by-example on-device keyword spotting," in *Proceedings of ASRU 2019 – IEEE Automatic Speech Recognition and Understanding Workshop, December 14-18, Singapore, Singapore*, pp. 532–538, 2019.

[54] A. Coucke, M. Chlieh, T. Gisselbrecht, D. Leroy, M. Poumeyrol, and T. Lavril, "Efficient keyword spotting using dilated convolutions and gating," in *Proceedings of ICASSP 2019 – 44th IEEE International Conference on Acoustics, Speech and Signal Processing, May 12-17, Brighton, UK*, pp. 6351–6355, 2019.