Interspeech 2022
18-22 September 2022, Incheon, Korea

Multi-class AUC Optimization for Robust Small-footprint Keyword Spotting with Limited Training Data

Menglong Xu, Shengqiang Li, Chengdong Liang, Xiao-Lei Zhang

CIAIC, School of Marine Science and Technology, Northwestern Polytechnical University, China

{mlxu, shengqiangli, chengdongliang}@mail.nwpu.edu.cn, xiaolei.zhang@nwpu.edu.cn

Abstract

Deep neural networks provide effective solutions to small-footprint keyword spotting (KWS). However, most of the KWS methods take softmax with the minimum cross-entropy as the loss function, which focuses only on maximizing the classification accuracy on the training set, without taking unseen sounds that are out of the training data into account. If training data is limited, it remains challenging to achieve robust and highly accurate KWS in real-world scenarios where the unseen sounds are frequently encountered. In this paper, we propose a new KWS method, which consists of a novel loss function, named the maximization of the *area under the receiver-operating-characteristic curve* (AUC), and a confidence-based decision method. The proposed KWS method not only maintains high keywords classification accuracy, but is also robust to the unseen sounds. Experimental results on the Google Speech Commands dataset v1 and v2 show that our method achieves state-of-the-art performance in terms of most evaluation metrics.

Index Terms: keyword spotting, multi-class AUC optimization

1. Introduction

Keyword spotting (KWS), also known as spoken term detection (STD), is the task of detecting some predefined keywords from a stream of utterances. It is usually used as an intelligent agent in mobile phones or smart devices. Recently, deep neural network (DNN) based KWS has led to significant performance improvement over conventional methods. Deep KWS [1] first considers keyword spotting as an audio classification problem. It trains a DNN model to predict the posteriors of predefined keywords, in which each neuron in the softmax output layer of the DNN model corresponds to a keyword, with an additional "filler" neuron representing all other non-keyword segments. This classification-based method achieves significant improvement over the keyword/filter hidden markov models. Later on, a number of classification-based methods [2, 3, 4, 5, 6, 7, 8] were explored to minimize the memory footprint. However, due to the closed nature of the softmax cross-entropy loss [9], the aforementioned models need to collect various non-keyword segments as training samples to achieve robust performance in practice [1, 2, 4]. Moreover, using a single "filler" neuron to represent all non-keyword segments ignores the diversity among these sounds, which may hurt the performance of the model.

Recently, several works [10, 11, 12, 13, 14] introduced metric learning into KWS. Metric learning adopts a ranking loss to learn the relative distance between samples. It aims to enlarge the inter-class variance and reduce the intra-class variance in an embedded space of data. However, it will result in a significant performance drop if one directly applies metric learning to KWS without considering the prior knowledge that the target keywords are predefined and fixed. To address the problem, Huh *et al*. [13] proposed an angular prototypical network with fixed target classes (AP-FC) to enhance the robustness against non-keyword segments. However, they have to use an additional support vector machine (SVM) to make the final decision.

Motivated by the works on AUC optimization [15, 16, 17, 18, 19] and open-set recognition problem [9, 20], in this paper, we propose a new loss function, named the maximization of the *area under the receiver-operating-characteristic curve* (AUC), and a simple confidence-based decision method, which leads to a robust, small-footprint, and high accuracy KWS model. Specifically, the proposed method not only maximizes the classification accuracy of keywords, but also maximizes the AUC score for optimizing the performance of non-keyword segments detection. It gets rid of the constraint of the closed softmax cross-entropy loss, i.e. the requirement that the summation of the output probabilities over all classes should be 1. Therefore, it is easy to detect non-keyword segments by a predefined threshold. We compared the proposed multi-class AUC loss with softmax cross-entropy loss [3], prototypical loss [13], AP-FC loss [13], and triplet loss [14] on the Google Speech Commands dataset v1 [21] and v2 [22]. Experimental results demonstrate that our methods outperform the comparison methods in most evaluation metrics.

The remainder of the paper is organized as follows. Section 2 describes existing binary AUC optimization. Section 3 introduces the proposed multi-class AUC loss. Section 4 and 5 present the experimental setup and results respectively. Section 6 concludes the paper.

2. Background

The original AUC optimization is only designed for binary-class classification [15, 16]. Therefore, before describing the proposed multi-class AUC loss function, we first take a look at the existing binary AUC optimization. Given a binary-class dataset $\mathbf{X} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$ where $y_n \in \{0, 1\}$, and a binary-class neural network $f_\theta(\cdot)$ with θ being the parameters of the network, we define two new subsets: $\mathbf{S}^+ = \{f_\theta(\mathbf{x}_n),\ \forall \mathbf{x}_n \in \mathbf{X} \mid y_n = 1\}$ which is a set of neural network output scores for the samples with $y_n = 1$, and $\mathbf{S}^- = \{f_\theta(\mathbf{x}_n),\ \forall \mathbf{x}_n \in \mathbf{X} \mid y_n = 0\}$ which represents a set of neural network output scores for the samples with $y_n = 0$. Cardinalities of these two subsets are N^+ and N^- respectively. As described in [23], for the finite set of samples $\mathbf{X}$, the approximate estimate of the AUC metric is:

$$\text{AUC} = \frac{1}{N^+ N^-} \sum_{i=1}^{N^+} \sum_{j=1}^{N^-} \mathbb{I}\left(s_i^+ > s_j^-\right) \tag{1}$$

where $\mathbb{I}(\cdot)$ is an indicator function that returns 1 if the statement is true, and 0 otherwise, and s_i^+ and s_j^- are the elements of $\mathbf{S}^+$

10.21437/Interspeech.2022-11356

and $\mathbf{S}^-$ respectively. As [24] did, we relax (1) by replacing the indicator function by a modified hinge loss function:

$$\ell'_{\text{hinge}}(z) = \max(0, \delta - z)^2 \tag{2}$$

where $z = s_i^+ - s_j^-$, and $\delta > 0$ is a tunable hyperparameter controlling the distance margin between s_i^+ and s_j^-. Substituting (2) into (1) transforms the maximization problem of (1) into the following minimization problem:

$$\ell = \frac{1}{N^+N^-} \sum_{i=1}^{N^+} \sum_{j=1}^{N^-} \max\left(0, \delta - \left(s_i^+ - s_j^-\right)\right)^2 \tag{3}$$

which can be easily backpropagated throughout the network in a standard procedure.

3. Algorithm description

3.1. The proposed multi-class AUC optimization

In this paper, we decompose the KWS task into a non-keyword segments detection subtask and a closed-set classification subtask. Specifically, for a given input sample, we first determine whether it belongs to the predefined keywords set. If so, then we decide which keyword it is. Note that the two subtasks are performed simultaneously in practice.

To formalize the task, suppose there is a dataset $\mathbf{X} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$ where $\mathbf{x}_n \in \mathbb{R}^D$ is a high-dimensional acoustic feature of the n-th sample, and $y_n \in \{0, 1, 2, \ldots, C\}$ is the ground-truth label of $\mathbf{x}_n$. Note that, without loss of generality, we always assume that there are $C + 1$ categories with class 0 representing non-keyword segments, and the other classes $1, 2, \ldots, C$ representing C keywords respectively. We aim to train a neural network $f_\theta(\cdot) : \mathbb{R}^D \to \mathbb{R}^C$. It maps the D-dimensional input acoustic feature to a C-dimensional vector. Each dimension of the vector represents the confidence score of its corresponding keyword. In the test stage, we use $f_\theta(\cdot)$ to conduct KWS by the following criterion:

$$\hat{y}_n = \begin{cases} \arg\max_c \left([p_{n,c}]_{c=1}^C\right), & \text{if } \max_c \left([p_{n,c}]_{c=1}^C\right) \geq \eta \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

where $[p_{n,c}]_{c=1}^C = [p_{n,1}, p_{n,2}, \ldots, p_{n,C}]^T$ is the output scores of the neural network $f_\theta(\mathbf{x}_n)$, and η is the decision threshold. For simplicity, we denote $\mathbf{p}_n = [p_{n,c}]_{c=1}^C$ in the remaining of the paper.

Several studies have extended the binary AUC to a multi-class problem [18, 19, 25, 26, 27, 28], see [19] for comprehensive reviews of multi-class AUC. In this work, we propose a new extension suitable for most multi-class classification tasks and computationally straightforward. The key idea of this extension is to modify the two subsets $\mathbf{S}^+$ and $\mathbf{S}^-$ in the binary AUC optimization to new forms that satisfy the multi-class AUC optimization problem. Specifically, for the general KWS problem with more than one keyword, we define the subset of positive examples as

$$\mathbf{S}^+ = \{p_{n,y_n},\ \forall \mathbf{x}_n \in \mathbf{X} \mid y_n \in \{1, 2, \ldots, C\}\}$$

and the subset of negative samples $\mathbf{S}^- = \mathbf{S}_1^- \cup \mathbf{S}_2^-$ with

$$\mathbf{S}_1^- = \left\{\max_c \left([p_{n,c}]_{c \neq y_n}\right),\ \forall \mathbf{x}_n \in \mathbf{X} \mid y_n \in \{1, 2, \ldots, C\}\right\}$$

where p_{n,y_n} is the score corresponding to the ground-truth label, $\max_c \left([p_{n,c}]_{c \neq y_n}\right)$ is the maximum score in $\mathbf{p}_n$ except p_{n,y_n}, and

$$\mathbf{S}_2^- = \left\{\max_c \left([p_{n,c}]_{c=1}^C\right),\ \forall \mathbf{x}_n \in \mathbf{X} \mid y_n = 0\right\}$$

represents the set of the output scores of the neural network for the non-keyword segments in $\mathbf{X}$. Algorithm 1 presents the proposed multi-class AUC loss in detail.

Algorithm 1 Multi-class AUC loss for KWS

Input:
a batch of acoustic features, $\mathbf{x}$;
the corresponding labels, $\mathbf{y}$;
the number of samples in the mini-batch, N;
predefined hyperparameter, δ;
Output:
loss ℓ on the current mini-batch;

$N^+ \leftarrow \sum_{n=1}^N \mathbb{I}(y_n \neq 0)$;
$N^- \leftarrow N$;
Init the positive subset $\mathbf{S}^+$ which contains N^+ samples and the negative subset $\mathbf{S}^- = \mathbf{S}_1^- \cup \mathbf{S}_2^-$ which contains N^- samples;
$\mathbf{p} \leftarrow f_\theta(\mathbf{x})$;
for each $y_n \in \mathbf{y}$ **do**
  if $y_n \neq 0$ **then**
    add p_{n,y_n} to $\mathbf{S}^+$;
    add the maximum score in $\mathbf{p}$ except p_{n,y_n} to $\mathbf{S}_1^-$;
  else
    add the maximum score in $\mathbf{p}$ to $\mathbf{S}_2^-$;
  end if
end for
$\ell = \frac{1}{N^+N^-} \sum_{i=1}^{N^+} \sum_{j=1}^{N^-} \max\left[0, \delta - \left(s_i^+ - s_j^-\right)\right]$;
return ℓ;

In the test stage, the decision threshold η is calculated on a validation set by:

$$\eta = -\delta + \frac{1}{\sum_{n=1}^{N'} \mathbb{I}(y_n \neq 0)} \sum_{n=1}^{N'} \mathbb{I}(y_n \neq 0)\, p_{n,y_n} \tag{5}$$

where N' is the size of the validation set.

3.2. Connection to other loss functions

3.2.1. Connection to multi-class hinge loss

Under the same supposition in Section 3.1, the multi-class classification hinge loss is presented as:

$$\ell_{\text{hinge}} = \frac{1}{NC} \sum_{n=1}^N \sum_{\substack{\{c \mid c=0,\ldots,\\ C,\ c \neq y_n\}}} \max(0, \delta - p_{n,y_n} + p_{n,c}) \tag{6}$$

The connection between the proposed multi-class AUC loss and the multi-class hinge loss is as follows. The multi-class AUC loss calculates the loss on the whole training set. It essentially learns a rank of the training samples without resorting to a classification-based loss explicitly. In contrast, the multi-class hinge loss calculates the optimization objective on each sample respectively and then averages them on the entire dataset. It needs to assign all non-keyword segments to a single class.

3.2.2. Connection to AP-FC loss

The AP-FC loss first arranges the keywords in a predefined order. Then, for each mini-batch, it selects one sample from each

keyword, followed by N_0 non-keywords. Note that the first C samples should be arranged in the predefined order of the keywords. According to [13], we rewrite the AP-FC loss as:

$$\begin{aligned}\ell_{\text{AP-FC}} &= -\frac{1}{C}\sum_{c=1}^{C}\log\frac{e^{\mathbf{S}_{c,c}}}{\sum_{n=1}^{C}e^{\mathbf{S}_{n,c}}+\sum_{k=1}^{N_0}e^{\mathbf{S}_{C+k,c}}} \\ &= -\frac{1}{C}\sum_{c=1}^{C}\log(\text{softmax}(\mathbf{S}^T))_{c,c} \end{aligned} \quad (7)$$

with

$$\mathbf{S}_{n,c} = w\cos(\mathbf{e}_n, \mathbf{W}_c) + b,\ c \in \{1, 2, \ldots, C\} \quad (8)$$

where $\mathbf{e}_n$ is the extracted feature of the n-th sample by the neural network, $\mathbf{W}_c$ is the learnable class center of the c-th keyword, and w, b are learnable parameters with $w > 0$.

The proposed AUC loss and AF-FC loss are similar in that they do not assign widely distributed non-keyword segments to a single "filler" class. However, the implementation of the AP-FC loss has a strict constraint on the samples in each mini-batch. Moreover, the AP-FC loss-based model still needs an SVM back-end to make the final decision.

3.2.3. Connection to other multi-class AUC loss

The multi-class AUC optimization in [28] is a natural extension of the binary AUC optimization. Gimeno *et al*. extended the binary AUC optimization to the multi-class problem by the one-versus-one and one-versus-rest frameworks. The one-versus-one multi-class AUC loss is obtained by averaging the pairwise binary AUC losses. The one-versus-rest multi-class AUC loss decomposes the multi-class classification task to C binary tasks. For the c-th task, the c-th class is viewed as a positive class, and all other classes are merged into a negative class. However, the above two methods cannot be directly used for our open-set optimization problem, since that they need to assign non-keyword segments to a "filler" class. In addition, it is obvious that our proposed AUC loss is more computationally efficient than the above two methods.

3.3. Model implementation

We use `res15` [3] as the backbone network. It starts with a bias-free convolution layer with weight $\mathbf{W} \in \mathbb{R}^{m\times r\times n}$, where m and r are the height and width of the convolution kernel respectively, and n is the number of the output channels. Then, it takes the output of the first convolution layer as the input of a chain of residual blocks, followed by a separate non-residual convolution layer. Finally, the output of the network is obtained by an average-pooling layer. Additionally, a (d_w, d_h) convolution dilation is used to increase the receptive field of the network, and a batch normalization layer is added after each convolution layer to help train the deep network. The details of the backbone network can be found in [3].

Usually, the training samples in each mini-batch are randomly sampled from the whole training set, which results in the proportion of the keywords over non-keywords in each mini-batch vary greatly. We denote this sampling method as `random sampler`. However, the variable proportion will hinder the convergence of the model training using the proposed method. To overcome this problem, we use a `fixed proportion sampler`, which keeps the proportion of keywords and non-keywords consistent in each mini-batch.

4. Experimental setup

In our experiments, two popular keyword spotting datasets, Google Speech Commands v1 (GSC v1) [21] and v2 (GSC v2) [22], are used for evaluation. The dataset GSC v1 consists of 65K one-second-long recordings of 30 words from thousands of different speakers. GSC V2 is an augmented version of GSC v1, which contains 105K utterances of 35 words. In addition, both datasets contain several minute-long background noise files. Both GSC v1 and GSC v2 include a "validation_list" file and a "testing_list" file. We use audio files in the "validation_list" and "testing_list" as validation and testing data respectively, and the other audio files as training data. Following previous works [3], we apply random time-shift and noise injection to training data.

Tasks in previous works [3, 5, 7, 8] focus on discriminating the 11 keywords ("yes", "no", "up", "down", "left", "right", "on", "off", "stop", "go", "silence") and a non-keyword "unknown", where "silence" denotes silence segments and "unknown" represents all other words. In their settings, all unknown words used in the test set have been seen by the model in the training stage, which is not consistent with real-world KWS applications. To meet the real-world KWS applications, in our experiments, we consider the task in [13], where ten unknown words ("zero", "one", "two", "three", "four", "five", "six", "seven", "eight", "nine") are used for testing only. We use `Total acc` as our primary evaluation metric to reflect the performance of the KWS models in real world applications. `Total acc` is the classification accuracy on the test set that contains unseen unknown words. Note that unseen unknown words represent the ten words above that are used in testing only. We also use `Closed acc` and `F1 score` as supplement evaluation metrics. `Closed acc` is the classification accuracy on the test set that does not contain unseen unknown words. `F1 score` is extended to multi-class one by "macro" average. It is calculated on the test set that contains unseen unknown words. In addition, we plot the detection error tradeoff (DET) curve of the non-keyword segments detection subtask.

Each model in our experiments is trained for 60 epochs, using the Adam optimizer [29]. The initial learning rate is set to 0.001 and reduced to 0.0001 after 30 epochs. For the softmax cross-entropy loss, we use a mini-batch size of 128 and L_2 weight decay of 10^{-5}. We use the same hyperparameters in [13] and [14] for the prototypical loss, AP-FC loss and triplet loss. We use the validation set to select the best model among different epochs and evaluate the effect of the hyperparameter δ. We evaluate the proposed multi-class AUC loss with the `fixed proportion sampler` and the `random sampler`. For the `fixed proportion sampler`, the number of keywords and non-keywords in each mini-batch is set to 32 and 64, respectively; for the `random sampler`, the mini-batch size is set to 128, which is the same as the other comparison methods. The hyperparameter δ is set to 0.3 (see Section 5.2 for the effect of δ). Following the same training procedure, we evaluate all comparison methods for five independent times and report the average performance.

5. Results

5.1. Evaluation of the proposed methods

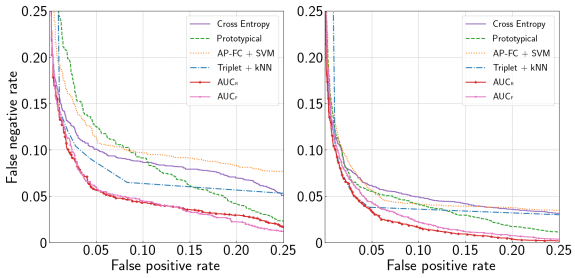
Table 1 lists the comparison result between the proposed methods and the four baselines. From the table, we see that both the two variants of the proposed multi-class AUC loss achieve significant improvement in terms of `Total acc` and

Table 1: *Comparison results between the proposed multi-class AUC and four referenced methods. The subscript R indicates the* `random sampler`*, and F the* `fixed proportion sampler`*.*

Loss	Back-end	GSC v1			GSC v2		
		Total acc	Closed acc	F1 score	Total acc	Closed acc	F1 score
Cross-entropy [3]	-	89.96%	97.14%	0.8805	92.74%	97.46%	0.9068
Prototypical [13]	-	87.89%	95.88%	0.8654	93.32%	96.55%	0.9149
AP-FC [13]	SVM	91.59%	96.72%	0.8962	93.77%	97.11%	0.9188
Triplet [14]	kNN	92.09%	**97.28%**	0.9019	94.01%	**97.78%**	0.9251
Multi-class AUC$_R$	-	92.16%	97.01%	0.9031	**94.87%**	97.39%	**0.9315**
Multi-class AUC$_F$	-	**92.97%**	97.22%	**0.9115**	94.71%	97.50%	0.9312

Table 2: *Effect of the hyperparameter δ on performance.*

	AUC							Cross Entropy
	sampler	0.1	0.2	0.25	0.3	0.35	0.4	0.5	
Closed acc	R	94.52%	96.29%	96.53%	96.85%	96.72%	96.49%	95.54%	96.20%
	F	94.04%	96.54%	96.71%	96.81%	96.44%	96.53%	96.20%	
F1 score	R	0.9426	0.9578	0.9581	0.9615	0.9577	0.9535	0.9429	0.9513
	F	0.9321	0.9599	0.9613	0.9613	0.9553	0.9546	0.9508	



(a) Results on GSC v1. (b) Results on GSC v2.

Figure 1: *DET curves of the non-keyword segments detection subtask.*

`F1 score`, and achieve a competitive result with the best referenced method in terms of `Closed acc`. We take the result on GSC v1 as an example. Comparing to the softmax cross-entropy loss, the multi-class AUC loss with the `fixed proportion sampler` achieves 30.0% and 25.9% relative improvement in `Total acc` and `F1 score`, respectively. It also achieves a slightly higher `Closed acc` than the softmax cross-entropy loss. Even when compared with the triplet loss with a complex kNN backend, the proposed method still achieves a relative improvement of 11.1% in `Total acc` and 9.8% in `F1 score` while maintaining a similar `Closed acc`.

To further investigate the effectiveness of the proposed method, we conduct a comparison on GSC v2 using the same settings as that on GSC v1. Experimental results again demonstrate the superiority of our method. In addition, the result on GSC v2 indicates that the training data of GSC v2 is responsible for the substantial improvement in all evaluation metrics, which is consistent with the experimental phenomenon in [22]. However, although both variants of the proposed multi-class AUC loss achieve better results on GSC v2 than that on GSC v1, the improvement with `random sampler` is more evident than that with the `fixed proportion sampler`. This may be caused by that the training data of GSC v2 contains more non-keywords than the training data of GSC v1. In addition, we plot the DET curves of the non-keyword segments detection subtask in Figure 1. From the figure, we see that these curves are consistent with the results presented in Table 1.

5.2. Effect of the hyperparameter δ on performance

Because there are no unseen unknown words in the validation set, here we only use `Closed acc` and `F1 score` as the evaluation metrics. For simplicity, we show the experimental results on GSC v1 in Table 2 only. Note that the experimental phenomenon on the other evaluation dataset is consistent with that on GSC v1. From the table, we see that the performance of the two variants of the multi-class AUC loss first increases and then decreases along with the increase of δ, where the best performance is achieved at $\delta = 0.3$. It is also observed that both the two variants of the multi-class AUC loss outperform the cross entropy baseline in the two evaluation metrics when $0.2 \leq \delta \leq 0.4$.

6. Conclusions

In this study, we have proposed a robust and highly accurate KWS method based on a novel multi-class AUC loss function and a confidence-based decision method. Our KWS method not only significantly improves the robustness of the model against unseen unknown words by optimizing the proposed multi-class AUC loss, but also eliminates the complex back-end processing module by using the simple confidence-based decision method. We compared the proposed method with four representative methods on the two public available datasets. Experimental results show that the proposed method significantly outperforms the four representative methods in most evaluations with smaller model sizes and less computational complexity than the latter.

7. Acknowledgement

This work was supported in part by the National Science Foundation of China (NSFC) under Grant 62176211, and in part by the Project of the Science, Technology, and Innovation Commission of Shenzhen Municipality, China under Grant JCYJ20210324143006016 and JSGG20210802152546026.

8. References

[1] G. Chen, C. Parada, and G. Heigold, "Small-footprint keyword spotting using deep neural networks," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 4087–4091.

[2] S. Ö. Arık, M. Kliegl, R. Child, J. Hestness, A. Gibiansky, C. Fougner, R. Prenger, and A. Coates, "Convolutional recurrent neural networks for small-footprint keyword spotting," *Proc. Interspeech 2017*, pp. 1606–1610, 2017.

[3] R. Tang and J. Lin, "Deep residual learning for small-footprint keyword spotting," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5484–5488.

[4] C. Shan, J. Zhang, Y. Wang, and L. Xie, "Attention-based end-to-end models for small-footprint keyword spotting," *Proc. Interspeech 2018*, pp. 2037–2041, 2018.

[5] 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," *Proc. Interspeech 2019*, pp. 3372–3376, 2019.

[6] Y. Bai, J. Yi, J. Tao, Z. Wen, Z. Tian, C. Zhao, and C. Fan, "A time delay neural network with shared weight self-attention for small-footprint keyword spotting." in *INTERSPEECH*, 2019, pp. 2190–2194.

[7] M. Xu and X.-L. Zhang, "Depthwise separable convolutional resnet with squeeze-and-excitation blocks for small-footprint keyword spotting," *Proc. Interspeech 2020*, pp. 2547–2551, 2020.

[8] C. Yang, X. Wen, and L. Song, "Multi-scale convolution for robust keyword spotting," *Proc. Interspeech 2020*, pp. 2577–2581, 2020.

[9] A. Bendale and T. E. Boult, "Towards open set deep networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1563–1572.

[10] N. Sacchi, A. Nanchen, M. Jaggi, and M. Cernak, "Open-vocabulary keyword spotting with audio and text embeddings," in *INTERSPEECH 2019-IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2019.

[11] Y. Yuan, Z. Lv, S. Huang, and L. Xie, "Verifying deep keyword spotting detection with acoustic word embeddings," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2019, pp. 613–620.

[12] P. Zhang and X. Zhang, "Deep template matching for small-footprint and configurable keyword spotting," *Proc. Interspeech 2020*, pp. 2572–2576, 2020.

[13] J. Huh, M. Lee, H. Heo, S. Mun, and J. S. Chung, "Metric learning for keyword spotting," in *2021 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2021, pp. 133–140.

[14] R. Vygon and N. Mikhaylovskiy, "Learning efficient representations for keyword spotting with triplet loss," in *Speech and Computer*, 2021, pp. 773–785.

[15] C. Cortes and M. Mohri, "Auc optimization vs. error rate minimization," in *17th Annual Conference on Neural Information Processing Systems, NIPS 2003*. Neural information processing systems foundation, 2003.

[16] L. Yan, R. H. Dodier, M. Mozer, and R. H. Wolniewicz, "Optimizing classifier performance via an approximation to the wilcoxon-mann-whitney statistic," in *Proceedings of the 20th international conference on machine learning (icml-03)*, 2003, pp. 848–855.

[17] A. Herschtal and B. Raskutti, "Optimising area under the roc curve using gradient descent," in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 49.

[18] J. Keshet, D. Grangier, and S. Bengio, "Discriminative keyword spotting," *Speech Communication*, vol. 51, no. 4, pp. 317–329, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167639308001489

[19] Z. Yang, Q. Xu, S. Bao, X. Cao, and Q. Huang, "Learning with multiclass auc: Theory and algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[20] T. DeVries and G. W. Taylor, "Learning confidence for out-of-distribution detection in neural networks," *arXiv preprint arXiv:1802.04865*, 2018.

[21] P. Warden, "Speech commands: A public dataset for single-word speech recognition," *Dataset available from http://download. tensorflow. org/data/speech_commands_v0*, vol. 1, 2017.

[22] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *arXiv preprint arXiv:1804.03209*, 2018.

[23] Z.-C. Fan, Z. Bai, X.-L. Zhang, S. Rahardja, and J. Chen, "Auc optimization for deep learning based voice activity detection," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 6760–6764.

[24] Z. Bai, X.-L. Zhang, and J. Chen, "Partial auc optimization based deep speaker embeddings with class-center learning for text-independent speaker verification," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6819–6823.

[25] D. J. Hand and R. J. Till, "A simple generalisation of the area under the roc curve for multiple class classification problems," *Machine learning*, vol. 45, no. 2, pp. 171–186, 2001.

[26] B. Yang, "The extension of the area under the receiver operating characteristic curve to multi-class problems," in *2009 ISECS International Colloquium on Computing, Communication, Control, and Management*, vol. 2. IEEE, 2009, pp. 463–466.

[27] P. Honzík, P. Kučera, O. Hynčica, and V. Jirsík, "Novel method for evaluation of multi-class area under receiver operating characteristic," in *2009 Fifth International Conference on Soft Computing, Computing with Words and Perceptions in System Analysis, Decision and Control*. IEEE, 2009, pp. 1–4.

[28] P. Gimeno, V. Mingote, A. Ortega, A. Miguel, and E. Lleida, "Generalising auc optimisation to multiclass classification for audio segmentation with limited training data," *IEEE Signal Processing Letters*, 2021.

[29] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.