METRIC LEARNING FOR USER-DEFINED KEYWORD SPOTTING

Jaemin Jung[1]*, *Youkyum Kim*[1]*, *Jihwan Park*[2,3], *Youshin Lim*[2,3], *Byeong-Yeol Kim*[2,3], *Youngjoon Jang*[1], *Joon Son Chung*[1]

[1]Korea Advanced Institute of Science and Technology, Daejeon, Republic of Korea
[2]Hyundai Motor Company, [3]42dot Inc., Seoul, Republic of Korea

https://mm.kaist.ac.kr/projects/kws

ABSTRACT

The goal of this work is to detect new spoken terms defined by users. While most previous works address Keyword Spotting (KWS) as a closed-set classification problem, this limits their transferability to unseen terms. The ability to define custom keywords has advantages in terms of user experience.

In this paper, we propose a metric learning-based training strategy for user-defined keyword spotting. In particular, we make the following contributions: (1) we construct a large-scale keyword dataset with an existing speech corpus and propose a filtering method to remove data that degrade model training; (2) we propose a metric learning-based two-stage training strategy, and demonstrate that the proposed method improves the performance on the user-defined keyword spotting task by enriching their representations; (3) to facilitate the fair comparison in the user-defined KWS field, we propose unified evaluation protocol and metrics.

Our proposed system does not require an incremental training on the user-defined keywords, and outperforms previous works by a significant margin on the Google Speech Commands dataset using the proposed as well as the existing metrics.

Index Terms— User-defined keyword spotting, Metric learning

1. INTRODUCTION

As an entry point to many speech-enabled systems, the performance of KWS systems is critical to providing a satisfactory user experience. While most existing KWS systems are based on a set of pre-defined keywords, the ability to define custom keywords can significantly improve user experience. For instance, the use of different keywords on different devices prevents accidental wake of nearby devices, and it also provides a layer of security by preventing access from strangers.

Most recent KWS methods [1, 2, 3, 4, 5] are based on classification networks that distinguish between target keywords and non-target noises. On the other hand, keyword spotting is closer to a detection task than a classification task, where the keywords are spotted from a range of unknown sounds. Therefore, user-defined keyword spotting can be naturally formulated as a metric learning problem, similar to face and speaker verification. To this end, we propose a metric learning-based training strategy to learn effective representations that can be used in query-by-keyword systems.

In many metric learning applications such as face [6, 7, 8] and speaker recognition [9, 10, 11], it has been demonstrated that the number of training classes strongly correlates with performance on the downstream task. However, the popular Google Speech Commands (GSC) dataset [12] only contains 35 classes, which is insufficient to facilitate good generalisation. To generate additional training data, previous works [13, 14, 15] extract keywords from Automatic Speech Recognition (ASR) datasets with a forced aligner [16] and then use them as training data. However, this approach does not guarantee the quality of the extracted keyword data. To mitigate this issue, we propose a new filtering method based on Character Error Rate (CER) to verify whether or not the extracted keywords are properly segmented using a pre-trained speech recognition model.

We conduct a wide variety of experiments using a range of objective functions borrowed from recent few-shot learning literature. Moreover, we propose a two-stage training strategy where we first pre-train the model on a large-scale out-of-domain corpus, then fine-tune the model on smaller in-domain data. We perform extensive ablations on the amount of training data, the two-stage training strategy, and various parameters to find their effects on KWS performance.

Finally, we list a number of metrics that are suitable for the user-defined keyword spotting as a detection task. While most existing works use the classification accuracy to evaluate their systems, the metrics of interest to the developers of KWS applications are the False Alarm Rates (FAR) and False Rejection Rates (FRR) at given operating points, represented on a Detection Error Tradeoff (DET) curve. Moreover, there is no standard evaluation protocol in the field that enables fair comparisons with different works. To this end, we propose an evaluation protocol that is relevant to the detection task. We evaluate the performance of the proposed methods on GSC dataset, using the existing metrics. Our model outperforms comparable works by a significant margin.

1.1. Related Works

In recent years, with the advances in deep learning technology, deep neural networks have been applied to the keyword spotting research. [17] utilises Convolutional Neural Network (CNN) to KWS, and [18] explores depth-wise separable convolution and point-wise convolution for the KWS task.

With the intimate involvement of technology into our daily lives, personalized services and privacy have become more important. As a result, there has been growing attention on *user-defined* keyword spotting technology. The user-defined KWS task can be viewed from two perspectives – as a *classification task* or a *detection task*.

[13, 14, 19] solve user-defined keyword spotting task as a classification task. [19] classifies non-target keywords into multiple classes in pre-training, and re-trains the model on the target keywords with data augmentation. [13] replaces the last linear layer with a randomly initialized linear layer during fine-tuning. [14] reinforces the model's representation capability by pre-training their model on the multilingual keyword dataset.

Some works [15, 20] approach the problem of KWS as a detection task. [15] designs a model architecture with multi-head attention layers and introduces soft-triple loss, which is a combination of triplet loss and softmax loss for learning feature representations. [20] proposes metric learning-based prototypical network that can effectively extract distinctive features to detect user-defined keywords. The method still requires an additional incremental training process to adapt the model to the target user-defined keywords. In contrast, our method does not require any incremental training during enrollment.

* These authors contributed equally.

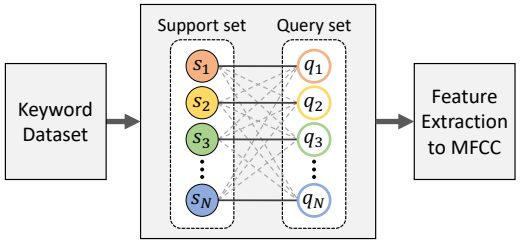


Fig. 1: Batch configuration for metric learning-based training. s_i and q_i stand for samples from i-th keyword class in the support set and the query set, respectively. N denotes size of a mini-batch. Solid lines connect positive pairs, while dotted lines represent negative pairs.

While there have been significant advances in keyword spotting algorithms, the field still suffers from the lack of diverse training data. To overcome this shortage problem, few recent studies try to extract keyword data from a large-scale speech corpus using forced aligners [16]. For example, [13, 15] and [14] extract keyword data from the LibriSpeech dataset [21] and the Common Voice dataset [22] respectively. [13, 14, 15] all use force-align transcripts to construct KWS datasets without verifying the aligned results. They also divide the datasets into training and test splits without considering the duplication of keywords between the sets.

In contrast, to validate the aligned keyword data, we evaluate CER on each of the keyword instances with a pre-trained speech recognition model and decide whether to include each data into the final training set. Moreover, we ensure that the keywords used in the pre-training or fine-tuning do not appear in the user-defined test data. Therefore, our experimental setup better addresses the *user-defined* keyword spotting problem in comparison to the previous literature.

2. METHODOLOGY

2.1. Large-scale Keyword Dataset

We construct a new large-scale keyword dataset, named LibriSpeech Keywords (LSK), consisting of 1,000 keyword classes extracted from the LibriSpeech corpus [21]. We utilise a pre-trained wav2vec 2.0 model [23] to force-align individual words from utterance-level labels. The wav2vec 2.0 model is pre-trained on 960 hours of unlabeled audio from LibriSpeech dataset and fine-tuned on the same audio with the corresponding transcript. The extracted keywords are truncated by 1 second to include noises or utterances that may occur before or after the keyword in a real-world scenario.

Unlike previous works [14, 15] which simply use the outputs of the forced aligner as their training dataset, we also validate the quality of the collected data. First, we compute CER score on each keyword in our dataset with the pre-trained wav2vec 2.0 model to filter misaligned examples which should not be used during the training step. Second, the 13 most frequent words and one-letter words are removed, because they consist mostly of articles and prepositions which are hard to recognise. Finally, 10 keywords in GSC dataset that are used as the user-defined keywords are removed. From this filtering process, we select the 1,000 most frequent keywords as our training data, followed by randomly sampling 1,000 instances per keyword. Note that our LSK dataset is only used in the *pre-training* stage.

2.2. Training Strategy

Our training strategy is divided into pre-training and fine-tuning stages. In the pre-training stage, our model is trained on the proposed LSK dataset containing out-of-domain data to represent spoken words in the discriminative embedding space. In the fine-tuning stage, our model is fine-tuned using only 25 keywords of the in-domain GSC dataset. During inference, we consider the remaining 10 keywords of GSC dataset to be user-defined keywords, simulating the real-world deployment scenario. Note that the GSC and LSK datasets exhibit different characteristics in terms of acoustics and word isolation, hence the need for fine-tuning.

2.3. Objective Functions

In this paper, we compare the baseline softmax loss and two metric learning-based objective functions. N denotes the number of utterances per mini-batch.

Softmax. Softmax loss consists of a softmax function followed by a multi-class cross-entropy loss. It is formulated as:

$$L_{\mathrm{S}} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{\mathbf{W}_{y_i}^T\mathbf{x}_i+b_{y_i}}}{\sum_{j=1}^{C}e^{\mathbf{W}_j^T\mathbf{x}_i+b_j}}, \tag{1}$$

where $\mathbf{W}$ and b are learnable parameters, $\mathbf{x}_i$ is the feature vector, $\mathbf{y}_i$ is the class label for corresponding $\mathbf{x}_i$. Here, C is the number of keyword classes. The loss function does not explicitly enforce intra-class compactness and inter-class separation. We use the softmax loss as the baseline objective function.

AM-Softmax. Additive Margin Softmax (AM-Softmax) loss [24, 25] introduces a margin to the original Softmax Loss. First, the weights and the input vectors are normalised in the softmax loss such that the posterior probability only relies on cosine of the angle between the weights and the input vectors:

$$L_{\mathrm{N}} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{\cos(\theta_{y_i,i})}}{\sum_j e^{\cos(\theta_{j,i})}}, \tag{2}$$

where $\cos(\theta_{j,i})$ is a dot product of normalised vector $\mathbf{W}_j$ and $\mathbf{x}_i$.

These embeddings are still not sufficiently discriminative because the equation only penalises the classification error. Cosine margin m is incorporated into the equation to mitigate this issue:

$$L_{\mathrm{C}} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s(\cos(\theta_{y_i,i})-m)}}{e^{s(\cos(\theta_{y_i,i})-m)}+\sum_{j\neq y_i}e^{s(\cos(\theta_{j,i}))}}, \tag{3}$$

where s is a fixed scale factor to prevent gradient from getting too small in training phase.

Angular Prototypical. The objective of prototypical loss [26] is to learn effective representations by explicitly optimising the distance between the query and the prototype (support set). In particular, we use the Angular Prototypical (AP) loss [27], which replaces the Squared Euclidean distance metric in the regular prototypical loss function with the cosine distance. For each keyword in a mini-batch, we consider one utterance out of M utterances as the query set, and the others as the support set. We will assume that the query is M-th utterance from every keyword for simplicity. Then the prototype (or centroid) for class k is:

$$\mathbf{c}_k = \frac{1}{M-1}\sum_{i=1}^{M-1}\mathbf{e}_{k,i}, \tag{4}$$

where $\mathbf{e}_{k,i}$ denotes an embedding feature. We use a cosine-based similarity metric with learnable scale w and bias b, as in the GE2E loss [28].

$$\mathbf{S}_{j,k} = w\cdot\cos(\mathbf{e}_{j,M},\mathbf{c}_k)+b \tag{5}$$

During training, each query example is classified against B classes in the mini-batch based on the softmax over distances to each keyword prototype:

$$L_{\mathrm{AP}} = -\frac{1}{B}\sum_{j=1}^{B}\log\frac{e^{\mathbf{S}_{j,j}}}{\sum_{k=1}^{B}e^{\mathbf{S}_{j,k}}}. \tag{6}$$

2.4. Batch Configuration

In each mini-batch, only one pair is positive and the rest are all considered negative pairs. As shown in Fig. 1, a positive pair consists of the same keyword but different audio data, whereas all the negative pairs consist of different keywords. For the prototypical-based networks, at least 2 samples per class per mini-batch are required.

3. EXPERIMENTS

3.1. Datasets

Google Speech Commands (GSC). To simulate a real-world deployment scenario, we select GSC dataset, which contains 35 different keywords, as our target domain keyword set. As shown in Table 1, we divide the GSC dataset into pre-defined, unknown, and user-defined splits. Note that only the pre-defined and unknown classes are used during the fine-tuning stage and we consider all classes in unknown split as one class.

Datasets	# Classes	Keywords
Pre-defined	10	'Yes', 'No', 'Up', 'Down', 'Left', 'Right', 'On', 'Off', 'Stop', 'Go'
Unknown	15	'Bed', 'Bird', 'Cat', 'Dog', 'Wow', 'House', 'Learn', 'Sheila', 'Tree', 'Happy', 'Marvin', 'Backward', 'Follow', 'Forward', 'Visual'
User-defined	10	'Zero', 'One', 'Two', 'Three', 'Four', 'Five', 'Six', 'Seven', 'Eight', 'Nine'

Table 1: Google Speech Commands dataset splits for user-defined keyword spotting.

LibriSpeech. LibriSpeech [21] dataset is a widely used speech corpus which contains 1,000-hour spoken sentences along with the corresponding transcripts. We construct our LibriSpeech Keywords (LSK) dataset using all of the training data in LibriSpeech.

Korean Speech Commands. Korean Speech Commands [29] contains a 4,000-hour Korean speech set. We construct our Korean Speech Keywords (KSK) dataset following the same pipeline used for the LSK dataset.

3.2. Evaluation Protocol

We reserve a specific number of samples from each target keyword for enrollment. We use the centroid of the embedding features extracted from the model as the prototype for each class. The model maps the input query signal to the embedding space and makes predictions based on the distance from the embedded features to the prototypes. We conduct experiments with 1, 5, and 10 samples for enrollment, referred to as 1-shot, 5-shot, and 10-shot enrollment, respectively.

The following metrics are employed to evaluate the performance of our system:

Equal Error Rate (EER). For a particular threshold value, two types of error rates are computed to evaluate how accurately the KWS model detects user-defined keywords: FRR indicates the proportion where the target keyword is not detected by the model, and FAR indicates the proportion where non-target keyword is detected as a target keyword. The trade-off between FRR and FAR can be visualised on a DET curve by changing the threshold (See our project page). The Equal Error Rate (EER) is determined where FAR and FRR are equal.

False Rejection Rate (FRR) at given False Alarm Rate (FAR). These metrics represent different operating points on the DET curve described above. In the industry, system designers are often interested in the detection rate for a fixed number of false alarms per hour (*e.g.* 70% detection rate at 5 false alarms). FRR at given FAR is measured to best simulate this requirement in the experiments.

F1-score. The F1-score combines the precision and recall of a classifier into a single metric by taking their harmonic mean. It is typically used to evaluate binary classification systems.

Accuracy. Accuracy represents the ratio of the number of correct predictions to the number of tests in total in a simple 10-way classification setup.

3.3. Implementation Details

Data preprocessing. We extract a 40 dimensional Mel-Frequency Cepstrum Coefficient (MFCC) with a 30ms window and 10ms frame shift. To augment the training data, we add diverse noises to the input data using the MUSAN dataset [30] and apply various Room Impulse Response (RIR) filters. The length of audio data is set to 1 second using truncation or zero padding operation.

Training Details. We select the `res15` architecture proposed in [31] as our baseline network. The network is optimised by the Adam optimizer [32]. For pre-training, the batch size is set to 256 and the initial learning rate is 10^{-3}. For fine-tuning, the batch size is set to 16 and the learning rate is initialised to 10^{-5}. During both pre-training and fine-tuning, we use a learning rate decay of 0.95 every epoch. Our framework is implemented with PyTorch [33].

4. RESULTS

In this section, we analyse the effects of various objective functions, pre-training methods, and data pre-processing. All models are pre-trained on the LSK dataset and fine-tuned on the GSC dataset unless otherwise stated.

Analysis on Objective Functions. The choice of objective function is paramount in learning effective representations. Table 2 reports the experimental results according to the various loss functions.

When one-stage training is applied without pre-training, the softmax and AM-Softmax show reasonable performance. The AP loss shows weak performance, since the loss requires sufficient number of classes to learn distinctive embeddings, but the limited number of classes in the GSC dataset hinders models' generalisation. On the other hand, when the pre-training and fine-tuning are performed, the model trained with AP loss in both stages shows the best performance on most metrics. Comparing this model to the baseline model trained with the softmax loss on the only GSC dataset, the performance gap between them stands out.

Effect of Pre-training. As reported in Table 2, the pre-trained models without the fine-tuning stage do not perform well because LSK dataset's characteristics are different from that of GSC in terms of acoustics and word isolation. When the model is fine-tuned with the softmax loss, the performance of the model deteriorates when compared to the model trained with only the GSC dataset. On the other hand, fine-tuning the model with metric learning-based objective functions enhances the performance of the model. In particular, the model pre-trained and fine-tuned with AP loss outperforms the other models.

In addition, when 1-shot enrollment is conducted, the performance of our best model is even better than that of the baseline [19] which uses 10-shot enrollment.

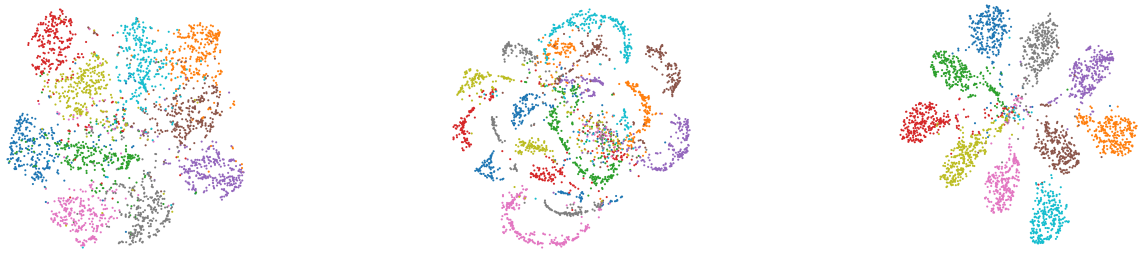
Ablation on the quantity of pre-training data. We perform ablations on the dataset configuration by changing (1) the number of the samples per keyword, and (2) the number of keywords while maintaining the number of samples per keyword. In both cases, the total number of samples is the same. As shown in Table 3, reducing the number of samples or the number

Training loss		EER ↓			Acc ↑			F1-score ↑			FRR@FAR=2.5 ↓			FRR@FAR=10 ↓		
Pre-train	Fine-tune	1-shot	5-shot	10-shot	1-shot	5-shot	10-shot	1-shot	5-shot	10-shot	1-shot	5-shot	10-shot	1-shot	5-shot	10-shot
[19] w/ Inc. Training		-	-	9.0†	-	-	-	-	-	-	-	-	17.0†	-	-	8.3†
-	Softmax	17.31	9.52	7.79	69.57	84.13	84.10	0.68	0.84	0.84	44.47	24.30	19.83	24.17	8.83	6.03
	AM-Soft	17.43	8.91	7.20	63.43	84.60	86.97	0.63	0.85	0.87	55.10	21.33	18.30	26.57	7.73	5.10
	AP	20.47	9.33	8.50	61.37	80.30	80.13	0.60	0.80	0.80	56.53	26.60	23.00	35.47	8.57	6.93
Softmax	-	30.77	20.64	19.01	47.07	62.23	67.23	0.47	0.63	0.68	66.10	59.57	44.10	51.20	36.87	27.77
	Softmax	16.91	11.00	9.20	69.47	83.23	85.47	0.68	0.83	0.85	48.33	26.67	21.67	25.10	11.67	8.67
	AM-Soft	10.47	4.75	4.01	85.43	94.80	95.33	0.85	0.95	0.95	24.20	6.90	5.97	10.87	3.07	2.03
	AP	10.10	5.20	3.77	83.53	94.23	95.00	0.83	0.94	0.95	23.00	7.67	5.47	10.23	3.40	2.20
AM-Soft	-	34.78	26.87	22.65	41.73	56.83	63.30	0.43	0.58	0.64	75.77	75.00	61.53	61.80	51.23	38.87
	Softmax	23.60	15.38	13.80	53.17	70.50	77.93	0.54	0.70	0.78	65.23	44.07	37.87	41.73	22.73	18.27
	AM-Soft	10.88	6.54	5.64	85.13	92.57	93.63	0.85	0.93	0.94	26.40	11.87	9.60	11.63	4.80	3.50
	AP	11.80	6.57	4.80	80.27	92.40	93.07	0.79	0.92	0.93	28.20	12.43	7.63	13.70	4.70	3.13
AP	-	32.70	24.81	21.01	41.23	60.03	69.37	0.45	0.61	0.70	80.50	74.03	57.67	59.60	50.77	36.23
	Softmax	15.81	10.97	8.87	70.07	80.77	83.33	0.71	0.81	0.84	55.10	29.60	20.77	25.83	12.07	7.57
	AM-Soft	8.08	5.31	4.27	88.53	**94.03**	95.53	0.88	**0.94**	**0.96**	17.10	7.50	5.67	6.90	3.37	2.60
	AP	**7.77**	**4.49**	**3.24**	**89.97**	93.93	**95.97**	**0.90**	**0.94**	**0.96**	**16.77**	**6.67**	**4.20**	**5.93**	**2.47**	**1.20**

Table 2: Experimental results using different loss functions. All numbers are in percent (%) except the F1-score. **FRR@FAR=10**: False Rejection Rate at False Alarm Rate of 10%; **AM-Soft**: Additive Margin Softmax loss; **AP**: Angular Prototypical loss; †: digitized from the DET curve on Figure 9(a) of [19].

Dataset	# Classes	# Samples	EER ↓	Acc ↑	F1-score ↑	FRR@FAR=2.5 ↓	FRR@FAR=10 ↓
LSK	500	500	4.13	94.50	0.95	6.07	2.13
	500	1,000	3.94	94.60	0.95	5.23	1.97
	1,000	500	3.63	95.07	0.95	5.13	1.47
	1,000	1,000	3.24	**95.97**	**0.96**	4.20	**1.20**
LSK+KSK	2,000	1,000	**3.07**	95.63	**0.96**	**3.73**	**1.20**

Table 3: The effect of pre-training data on final system performance. All numbers are in percent (%) except for F1-score. All experiments in this table utilise the Angular Prototypical (AP) loss.



(a) Trained only on the GSC dataset.

(b) Pre-trained only on the LSK dataset.

(c) Pre-trained on the LSK dataset, then fine-tuned on the GSC dataset.

Fig. 2: t-SNE visualisation of the embedding of unseen keywords. All different colours indicate different keywords. When pre-training and fine-tuning are conducted, the model better represents user-defined keywords in the distinctive embedding space.

of classes degrades the KWS performance. The results emphasise that the number of classes is a more important factor for the model to generalise well to unseen user-defined keywords. From this observation, we further enlarge the number of classes by adding the KSK dataset to the LSK dataset. When pre-trained on both datasets, the model gains performance even when the additional dataset is composed of another language.

Qualitative Results. We visualise the embeddings extracted from KWS models on the t-SNE plot [34] in Fig. 2. The plot shows how well our model separates the user-defined keywords for each of the different training strategies. As illustrated in Figs. 2a and 2b, the models trained on the GSC dataset or only pre-trained on the LSK dataset cannot extract distinctive embeddings from the unseen audios. On the other hand, Fig. 2c shows that the proposed training strategy, where the model is pre-trained and fine-tuned sequentially, improves both inter-class compactness and intra-class separation in the embedding space.

Effect of CER-based filtering. We verify the effect of the proposed CER-based filtering method in Table 4. Except for the use of data filtering, all other training details are constant. Comparing the performance, the model trained on the filtered data outperforms the model trained on the unfiltered data. Therefore, we confirm that removing the misaligned audio samples using the filtering process has a positive effect on performance.

Filtering	EER ↓	Acc ↑	F1-score ↑	FRR@FAR=2.5 ↓	FRR@FAR=10 ↓
✗	3.47	95.67	**0.96**	4.83	1.47
✓	**3.24**	**95.97**	**0.96**	**4.20**	**1.20**

Table 4: Results with and without the proposed CER-based filtering. All numbers are in percent (%) except for F1-score.

5. CONCLUSION

In this paper, we have proposed a novel metric learning-based training strategy for user-defined KWS task to represent spoken keywords in the discriminative embedding space. We collect large-scale out-of-domain keyword data, LSK dataset, to pre-train a model that can be utilised in various KWS-based downstream tasks. Furthermore, we fine-tune the model on in-domain but non-overlapping classes to improve generalisation to the target task.

We provide extensive ablations of the different factors that can affect the user-defined KWS performance. The experiments demonstrate that our best model achieves the state-of-the-art performance on the user-defined KWS task.

6. REFERENCES

[1] Seungwoo Choi, Seokjun Seo, Beomjun Shin, Hyeongmin Byun, Martin Kersner, Beomsu Kim, Dongyoung Kim, and Sungjoo Ha, "Temporal convolution for real-time keyword spotting on mobile devices," *arXiv preprint arXiv:1904.03814*, 2019. 1

[2] Peter Mølgaard Sørensen, Bastian Epp, and Tobias May, "A depthwise separable convolutional neural network for keyword spotting on an embedded system," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2020, no. 1, pp. 1–14, 2020. 1

[3] Byeonggeun Kim, Simyung Chang, Jinkyu Lee, and Dooyong Sung, "Broadcasted residual learning for efficient keyword spotting," *arXiv preprint arXiv:2106.04140*, 2021. 1

[4] Axel Berg, Mark O'Connor, and Miguel Tairum Cruz, "Keyword transformer: A self-attention model for keyword spotting," *arXiv preprint arXiv:2104.00769*, 2021. 1

[5] Jaesung Huh, Minjae Lee, Heesoo Heo, Seongkyu Mun, and Joon Son Chung, "Metric learning for keyword spotting," in *IEEE Spoken Language Technology Workshop*. IEEE, 2021, pp. 133–140. 1

[6] Florian Schroff, Dmitry Kalenichenko, and James Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proc. CVPR*, 2015, pp. 815–823. 1

[7] Kihyuk Sohn, "Improved deep metric learning with multi-class n-pair loss objective," in *NeurIPS*, 2016, vol. 29. 1

[8] Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, and Andrew Zisserman, "VGGFace2: A dataset for recognising faces across pose and age," in *IEEE International Conference on Automatic Face & Gesture recognition*. IEEE, 2018, pp. 67–74. 1

[9] Joon Son Chung, Arsha Nagrani, and Andrew Zisserman, "Voxceleb2: Deep speaker recognition," in *Proc. Interspeech*, 2018. 1

[10] Jixuan Wang, Kuan-Chieh Wang, Marc T Law, Frank Rudzicz, and Michael Brudno, "Centroid-based deep metric learning for speaker recognition," in *Proc. ICASSP*. IEEE, 2019, pp. 3652–3656. 1

[11] Yoohwan Kwon, Hee-Soo Heo, Bong-Jin Lee, and Joon Son Chung, "The ins and outs of speaker recognition: lessons from voxsrc 2020," in *Proc. ICASSP*. IEEE, 2021, pp. 5809–5813. 1

[12] Pete Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *arXiv preprint arXiv:1804.03209*, 2018. 1

[13] Abhijeet Awasthi, Kevin Kilgour, and Hassan Rom, "Teaching keyword spotters to spot new keywords with limited examples," *arXiv preprint arXiv:2106.02443*, 2021. 1, 2

[14] Mark Mazumder, Colby Banbury, Josh Meyer, Pete Warden, and Vijay Janapa Reddi, "Few-shot keyword spotting in any language," *arXiv preprint arXiv:2104.01454*, 2021. 1, 2

[15] Jinmiao Huang, Waseem Gharbieh, Han Suk Shim, and Eugene Kim, "Query-by-example keyword spotting system using multi-head attention and soft-triple loss," in *Proc. ICASSP*. IEEE, 2021, pp. 6858–6862. 1, 2

[16] Michael McAuliffe, Michaela Socolof, Sarah Mihuc, Michael Wagner, and Morgan Sonderegger, "Montreal forced aligner: Trainable text-speech alignment using kaldi.," in *Proc. Interspeech*, 2017, vol. 2017, pp. 498–502. 1, 2

[17] Tara Sainath and Carolina Parada, "Convolutional neural networks for small-footprint keyword spotting," 2015. 1

[18] Yundong Zhang, Naveen Suda, Liangzhen Lai, and Vikas Chandra, "Hello edge: Keyword spotting on microcontrollers," *arXiv preprint arXiv:1711.07128*, 2017. 1

[19] Li Liu, Mingxue Yang, Xinyi Gao, Qingsong Liu, Zhengxi Yuan, and Jun Zhou, "Keyword spotting techniques to improve the recognition accuracy of user-defined keywords," *Neural Networks*, vol. 139, pp. 237–245, 2021. 1, 3, 4

[20] Archit Parnami and Minwoo Lee, "Few-shot keyword spotting with prototypical networks," in *Proc. ICML*, 2022, pp. 277–283. 1

[21] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *Proc. ICASSP*. IEEE, 2015, pp. 5206–5210. 2, 3

[22] Rosana Ardila, Megan Branson, Kelly Davis, Michael Henretty, Michael Kohler, Josh Meyer, Reuben Morais, Lindsay Saunders, Francis M Tyers, and Gregor Weber, "Common voice: A massively-multilingual speech corpus," *arXiv preprint arXiv:1912.06670*, 2019. 2

[23] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *NeurIPS*, vol. 33, pp. 12449–12460, 2020. 2

[24] Feng Wang, Jian Cheng, Weiyang Liu, and Haijun Liu, "Additive margin softmax for face verification," *IEEE Signal Processing Letters*, vol. 25, no. 7, pp. 926–930, 2018. 2

[25] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu, "Cosface: Large margin cosine loss for deep face recognition," in *Proc. CVPR*, 2018, pp. 5265–5274. 2

[26] Jake Snell, Kevin Swersky, and Richard Zemel, "Prototypical networks for few-shot learning," in *NeurIPS*, 2017, vol. 30. 2

[27] Joon Son Chung, Jaesung Huh, Seongkyu Mun, Minjae Lee, Hee Soo Heo, Soyeon Choe, Chiheon Ham, Sunghwan Jung, Bong-Jin Lee, and Icksang Han, "In defence of metric learning for speaker recognition," in *Proc. Interspeech*, 2020. 2

[28] Li Wan, Quan Wang, Alan Papir, and Ignacio Lopez Moreno, "Generalized end-to-end loss for speaker verification," in *Proc. ICASSP*. IEEE, 2018, pp. 4879–4883. 2

[29] Yura Hwang, "Korean Speech Commands," https://aihub.or.kr/aihubdata/data/view.do?currMenu=115&topMenu=100&aihubDataSe=realm&dataSetSn=96, 2020. 3

[30] David Snyder, Guoguo Chen, and Daniel Povey, "Musan: A music, speech, and noise corpus," *arXiv preprint arXiv:1510.08484*, 2015. 3

[31] Raphael Tang and Jimmy Lin, "Deep residual learning for small-footprint keyword spotting," in *Proc. ICASSP*. IEEE, 2018, pp. 5484–5488. 3

[32] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014. 3

[33] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer, "Automatic differentiation in pytorch," 2017. 3

[34] Laurens Van der Maaten and Geoffrey Hinton, "Visualizing data using t-sne.," *Journal of machine learning research*, vol. 9, no. 11, 2008. 4