Interspeech 2022
18-22 September 2022, Incheon, Korea

EXPLORING REPRESENTATION LEARNING FOR SMALL-FOOTPRINT KEYWORD SPOTTING

Fan Cui, Liyong Guo, Quandong Wang, Peng Gao, Yujun Wang

Xiaomi, Beijing, China
cuifan@xiaomi.com

Abstract

In this paper, we investigate representation learning for low-resource keyword spotting (KWS). The main challenges of KWS are limited labeled data and limited available device resources. To address those challenges, we explore representation learning for KWS by self-supervised contrastive learning and self-training with pretrained model. First, local-global contrastive siamese networks (LGCSiam) are designed to learn similar utterance-level representations for similar audio samplers by proposed local-global contrastive loss without requiring ground-truth. Second, a self-supervised pretrained Wav2Vec 2.0 model is applied as a constraint module (WVC) to force the KWS model to learn frame-level acoustic representations. By the LGCSiam and WVC modules, the proposed small-footprint KWS model can be pretrained with unlabeled data. Experiments on speech commands dataset show that the self-training WVC module and the self-supervised LGCSiam module significantly improve accuracy, especially in the case of training on a small labeled dataset.

Index Terms: keyword spotting, representation learning, self-supervised learning, self-training

1. Introduction

Keyword spotting (KWS) aims to identify the target keyword in audio segments, mainly used as the activation of voice assistants. With the development of deep learning based speech technology, it is possible to deploy high-performance small-footprint keyword spotting system in commercial products (e.g., "Hey Siri" [1], "Alexa" [2], and "Okay Google" [3]).

Recent keyword spotting methods focus on improving performance and reducing computation complexity to deploy on low-resource devices [4, 5, 6, 7, 8, 9]. Especially, convolutional neural networks (CNN) based KWS methods show remarkable accuracy and efficiency [1, 2, 7, 10]. 2D CNN-based methods need very deep model to capture the dependency of different frequency bins and different frames. TC-ResNet [11] is a fast and accurate KWS model, which applies temporal convolution network. However, those methods require large volumes of labeled data, which is time-consuming and expensive. Additionally, data distribution shifting between different recording environments leads to model mismatch.

Self-supervised pretrained models have been investigated and shown significant improvements in different machine learning topics. BERT [12] is a pretrained model that serves as a foundation for improving the accuracy of machine learning in natural language processing. Self-supervised speech recognition has been proposed with many approaches [13, 14, 15], which provide pretrained models that can be fine-tuned on downstream tasks for both high-resource and low-resource conditions. Those self-supervised pretrained models are trained on large amounts of unlabeled data and can be used in two ways. First, they work as feature extractors and fine-tuned on a small amount of labeled data [13]. Second, they serve as teacher models to generate pseudo-label for training student models [16, 17]. Depending on the downstream tasks, the student model can be smaller. When the labeled data is enough to train the student model, pseudo-label can remove low-quality ground truth and avoid the student model learning implausible data. Additionally, pseudo-label can be generated for unlabeled data when the initial labeled data is not large enough to train a reliable model. In [18], pretrained embedding representation learning is proposed for few-shot KWS. However, the pretrained model is too large to deploy on low-resource devices.

In this study, we aim at exploring representation learning to improve the small-footprint KWS model from two aspects. First, self-supervised pretrained Wav2Vec 2.0 [15] acoustic model is applied for teacher-student training, which uses unlabeled data to learn meaningful acoustic embedding representations. Instead of fine-tuning the Wav2Vec 2.0 for the downstream KWS model, we use it to extract hidden representations and force the encoder of the KWS model to learn similar speech representations. Second, we design a self-supervised contrastive siamese networks (LGCSiam) for the proposed KWS model to learn similar utterance-level representations for similar audio samplers. The siamese networks are built upon recent progresses of contrastive learning in computer vision [19, 20]. Siamese networks are weight-sharing neural networks with similar but different inputs. In this task, the random augments of audio sampler are used as the inputs of siamese networks.

Our contributions are as follows:

- We propose a local-global contrastive siamese structure for KWS model to learning utterance representation from unlabeled data. An undesired result to siamese networks is all the outputs "collapsing" to a constant [20]. To prevent from collapsing, the proposed structure generates positive pairs and negative pairs for contrastive learning.
- We investigate teacher-student learning with self-supervised pretrained Wav2Vec 2.0 model for KWS model to learn acoustic representation.
- A light-weight KWS model TCANet is proposed, which uses temporal convolution networks in the encoder part, and adopt multi-head self-attention network as the decoder part.
- An evaluation of the representation learning for low-resource KWS.

The remainder of this paper is organized as follows: Section 2 introduces our proposed low-resource keyword spotting method. The experimental setup is presented in Section 3. The experimental results and analyses are shown in Section 4. Section 5 is the conclusion part.

Copyright © 2022 ISCA

10.21437/Interspeech.2022-10558

2. Methodology

In this section, we present our system summarized in Figure 1, contains self-training WVC module, self-supervised LGCSiam module and the light-weight TCANet model architecture.

2.1. TCANet Model Architecture

As shown in Figure 1, the input waveforms are transformed to log-melspectrogram $x \in \mathbb{R}^{B\times T\times 40}$, where B represents batch size, T is the maximum frame index, and the number of mel filterbanks is 40. The proposed TCANet model mainly contains the encoder $E(\cdot)$ and the decoder $D(\cdot)$. The encoder structure is similar to TC-ResNet, containing seven convolutional layers. The kernel size is 3 for the first layer and that of the other layers are 9. To reduce the complexity, the stride of the first layer is 2, which reduces time domain from T to $\frac{T}{2}$. The output channel of each conv layer is 64. Rather than using normal conv1d layer, we use separable conv1d to reduce parameters except the first layer. Compared with normal conv1d, separable conv1d can reduce parameter size from $9*64*64$ to $9*64*1+1*64*64$. Each of the convolutional layers followed by batch normalization and ReLU. After encoder, input $x \in \mathbb{R}^{B\times T\times 40}$ is transformed to $e \in \mathbb{R}^{B\times T/2\times 64}$.

The output embeddings of the encoder are fed into the decoder, which is a multi-head attention block [21]. The query, key and value are obtained by linear projections: $Q = e\cdot W_Q$, $K = e\cdot W_K$, $V = e\cdot W_V$, where $W_Q, W_K, W_V \in \mathbb{R}^{64\times 64}$. For each attention head, the output is calculated as $head_i = Softmax(\frac{Q_i K_i^T}{h_n})V_i$, where h_n is $\frac{64}{num_heads}$. The result of the decoder part is:

$$D(e) = Concat(head_1, ...head_n)W_o \tag{1}$$

The output of each attention head are concatenated, and projected by matrix $W_o \in \mathbb{R}^{64\times 64}$. For supervised training, a global average pooling layer is applied to merge time domain to one frame. Finally, dense and softmax layer are adopted to get classification probability $p \in \mathbb{R}^{B\times C}$, C is the number of class. The ground-truth is $y \in \mathbb{R}^{B\times C}$, where $y_{ij} = \{0, 1\}$. The cross entropy (CE) loss is adopted to optimize the classification model:

$$Loss_{CE} = -\sum_{i=1}^{B}\sum_{j=1}^{C} y_{ij} log(p_{ij}) \tag{2}$$

2.2. WVC Module

The Wav2Vec 2.0 model is a transformer-based neural network, which is pretrained on a large volume of unlabeled data. It is effective for semi-supervised downstream tasks training with small labeled dataset. It mainly contains four elements: feature encoder, context network, quantization module and pretraining contrastive loss [13].

In the paper, we use Wav2Vec 2.0 base model[1], which processes the latent feature vectors obtained by feature encoder through 12 transformer blocks in context network to increase the dimension from 512 to 768. In our WVC module, input waveforms are fed into Wav2Vec 2.0 base model to get a context representation $e_0 \in \mathbb{R}^{B\times T/2\times 768}$, where T is the maximum frame index with frame-shift 10ms. The KWS encoder is forced to learn frame-level acoustic representation, its output $e_1 \in \mathbb{R}^{B\times T/2\times 64}$ should be similar to the output of Wav2Vec 2.0 model e_0. Multi-layer perceptron (MLP) block is adopted as a projection head to transform e_1 to $e_1^{'}$:

$$e_1^{'} = W_2\sigma(W_1\cdot e_1) \tag{3}$$

Where $W_1 \in \mathbb{R}^{64\times 128}, W_2 \in \mathbb{R}^{128\times 768}$, σ is a ReLU nonlinearity. MSE loss is applied to measure the similarity of $e_1^{'}$ and e_0:

$$Loss_{WVC} = \left\| e_1^{'} - e_0 \right\|_2 \tag{4}$$

2.3. LGCSiam Module

Two randomly augmented waveforms are transformed to log-melspectrogram $x_1 \in \mathbb{R}^{B\times T\times 40}$, $x_2 \in \mathbb{R}^{B\times T\times 40}$ token as the inputs of LGCSiam module. As shown in Figure 1, the two branches of LGCSiam module share same weights in encoder and decoder parts. The two augmented samplers are processed by encoder $E(\cdot)$ and decoder $D(\cdot)$ networks to get representations $d_1 \in \mathbb{R}^{B\times \frac{T}{2}\times 64}$,$d_2 \in \mathbb{R}^{B\times \frac{T}{2}\times 64}$:

$$d_1 = D(E(x_1)), d_2 = D(E(x_2)) \tag{5}$$

A MLP projection head transforms the decoder representations as:

$$d_1^{'} = W_4\sigma(W_3\cdot d_1), d_2^{'} = W_4\sigma(W_3\cdot d_2) \tag{6}$$

Where $W_3 \in \mathbb{R}^{64\times 128}, W_4 \in \mathbb{R}^{128\times 128}$, σ is a ReLU.

Our proposed Local-Global contrastive loss can force the KWS model to learning similar representations $d_1^{'}, d_2^{'}$. For calculating contrastive loss, we prepare positive and negative pairs, shown in Figure 2. From local aspect, the corresponding frame in decoder results $d_1^{'}, d_2^{'}$ should servers as positive pairs $[d_1^{'}(0,0,:), d_2^{'}(0,0,:)]$, $[d_1^{'}(0,1,:), d_2^{'}(0,1,:)]$...$[d_1^{'}(B,T,:), d_2^{'}(B,T,:)]$. From a global aspect, the time-domain averaging of $d_1^{'}, d_2^{'}$ should be a positive pair $[\frac{2}{T}\sum_{i=0}^{\frac{T}{2}} d_1^{'}(:,i,:), \frac{2}{T}\sum_{i=0}^{\frac{T}{2}} d_2^{'}(:,i,:)]$. To avoiding the network learning useless same representations for all the inputs, the negative pairs should be designed. In this paper, the negative pairs are built by the decoder results of different timestamps $[d_1^{'}(j_1,i_1,:), d_2^{'}(j_2,i_2,:)]$, where $i_1 \neq i_2$ or $j_1 \neq j_2$. The global consistency loss is calculated as:

$$L_{global} = \frac{1}{B}\sum_{j=1}^{B}\{[\frac{2}{T}\sum_{i=0}^{\frac{T}{2}} d_1^{'}(j,i,:) - \frac{2}{T}\sum_{i=0}^{\frac{T}{2}} d_2^{'}(j,i,:)]^2\} \tag{7}$$

The local contrastive loss is calculated as:

$$L_{local} = -log\frac{exp(sim(d_1^{'}(i_1,j_1), d_2^{'}(i_1,j_1))/\tau)}{\sum_{i_2=0,j_2=0}^{i_2=B,j_2=\frac{T}{2}} exp(sim(d_1^{'}(i_1,j_1), d_2^{'}(i_2,j_2))/\tau)} \tag{8}$$

where $i_1 \neq i_2 || j_1 \neq j_2$, temperature parameter τ is 0.5, function sim is cosine similarity. The local-global contrastive loss is:

$$Loss_{LGCSiam} = L_{global} + L_{local} \tag{9}$$

The pretraining stage contains two steps. First, the encoder with WVC is optimized by $Loss_{WVC}$ with unlabeled data. Second, the encoder and decoder modules are trained with unlabeled data by loss function:

$$L = \lambda_1 LossLGCSiam + \lambda_2 LossWVC \tag{10}$$

[1] https://dl.fbaipublicfiles.com/fairseq/wav2vec/wav2vec_small.pt

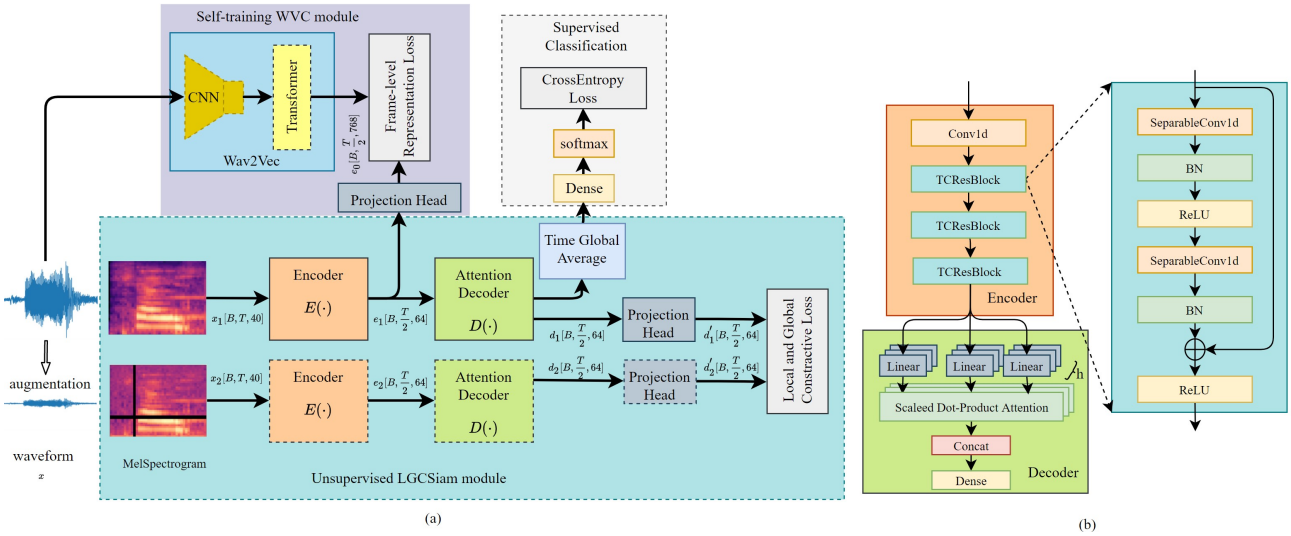

Figure 1: *The proposed architecture. (a) the training workflow contains three parts: self-training WVC module, unsupervised LGCSiam module and supervised classification module. The WVC and LGCSiam modules are used for representation learning. The two branches of unsupervised LGCSiam module share the same weights. (b) shows the detail of TCANet model structure.*

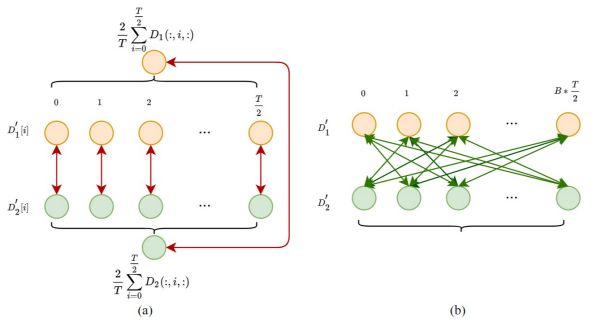

Figure 2: *Positive pairs and negative pairs for local-global contrastive loss. (a) shows the global and local positive pairs of two corresponding audio samplers, (b) shows the negative pairs generated by different samplers or different timestamps.*

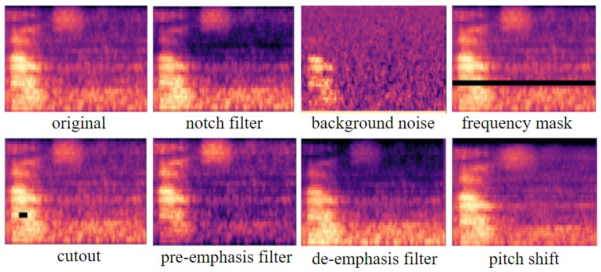

Figure 3: *Different data augmentation operations.*

In the supervised fine-tuning stage, the classification module is added, the CE loss defined in equation 2 is combined with contrastive loss. The total loss function is:

$$L = \gamma_1 LossCE + \gamma_2 LossLGCSiam + \gamma_3 LossWVC \quad (11)$$

where λ_1, λ_2, γ_1, γ_2, γ_3 are weights of the corresponding losses in different stages.

3. Experimental setup

3.1. Dataset

The proposed models are trained and evaluated with speech commands dataset [22], which contains 65000 one-second long audio utterances. Following the implementation of [22], we split the dataset into training, validation, and test sets. Speech commands dataset contains 12 classes: "yes", "no", "up", "down", "left", "right", "on", "off", "stop", "go", unknown, or silence. We randomly select and clip background noise to generate background noise. To compare the representation learning on different domain datasets, unlabeled AISHELL dataset [23] is used to pretrain the WVC module and LGCSiam module.

3.2. Implementation Details

All the models are trained and evaluated using Pytorch [24] toolkit with a mini-batch of 128 samples. We use the stochastic gradient descent optimizer with momentum of 0.9. To avoid overfitting, we use a weight decay of 0.0001. The initial learning rate is set to be 0.1 and decayed by a factor of 3 when the validation accuracy does not increase for 3 epochs. For equation 10, λ_1, λ_2 are 0.1, 0.9. For equation 11, γ_1, γ_2, γ_3 are 0.9, 0.05, 0.05.

3.3. Data Augmentation

Figure 3 shows the data augmentation operations randomly applied in our experiments:

- Pre-emphasize: pre-emphasize an audio signal with a first-order autoregressive filter, $(y[n] = y[n] - coef * y[n-1]), coef \in \{0.95 \sim 0.99\}$
- De-emphasize: de-emphasize an audio signal with the inverse operation of Pre-emphasize.
- Pitch shift: shift the pitch of a waveform by $n_{steps} \in \{-5 \sim 5\}$ steps.

- Notch filter & Peak filter: suppress or enhance the signal at a random frequency range.
- Adding background noise: the signal-to-noise ratio is between -5dB to 15dB.
- Frequency masking [25]: the maximal number of masking frequency is 10.
- Cutout: the maximal number of cutout frequency is 10 and the maximal number of time step is 10.

3.4. Model setup

3.4.1. Baseline system

- TC-ResNet [11]: TC-ResNet8 has one convolution layer and three residual blocks and 16, 24, 32, 48 channels for each layer. TC-ResNet14 expands the network by adding three residual blocks compared to TC-ResNet8. TC-ResNet8-1.5 and TC-ResNet14-1.5 are based on TC-ResNet8 and TC-ResNet14 with a channel multiplier of 1.5 expands the model to have 24, 36, 48, 72 number of channels respectively.
- DS-CNN [8]: DS-CNN utilizes depthwise convolutional layer, which achieve high accuracy when memory and computation resources are constrained. DS-CNN-S, DS-CNN-M, and DS-CNN-L represent small-, medium-, and large-size model, respectively.

3.4.2. Proposed system

- TCANet: train the proposed TCANet model with labeled data.
- TCANet+WVC: pretrain TCANet by the WVC module with unlabeled data and fine-tuned with labeled data.
- TCANet+LGCSiam: pretrain TCANet by the LGCSiam module with unlabeled data and fine-tuned with labeled data.
- TCANet+WVC+LGCSiam: firstly, pretrain TCANet by the WVC module. Then, the LGCSiam module is trained with unlabeled data. Finally, fine-tune the whole model with labeled data.

4. Results and Analysis

Impact of WVC and LGCSiam module pretrained with speech commands dataset. In table 1, we compare the original TCANet model trained with 100% and 5% speech commands dataset and the TCANet+LGCSiam, TCANet+WVC, TCANet+WVC+LGCSiam pretrained with unlabeled speech commands dataset and fine-tuned with 100% and 5% labeled data. Compared with supervised TCANet, pretrained models have a significant accuracy gain. The results show that the unsupervised pretraining helps the encoder and decoder to learning useful representation.

Impact of WVC and LGCSiam module pretrained with AISHELL dataset. In table 2, we compare the original TCANet trained with 100% and 5% speech commands dataset and the TCANet+LGCSiam, TCANet+WVC, TCANet+WVC+LGCSiam pretrained with unlabeled AISHELL dataset and fine-tuned with 100% and 5% labeled speech commands dataset. From the results, we can see the models pretrained with AISHELL dataset also help to get a performance gain. The results show that the self-supervised pretraining on different domain dataset is useful for fine-tuning the downstream KWS task.

In table 3, the proposed TCANet, TCANet+LGCSiam, TCANet+WVC, TCANet+WVC+LGCSiam are compared with the baseline systems. The results demonstrate the proposed system have a better performance with a smaller model size.

Table 1: *Comparing the influence of LGCSiam and WVC module for representation learning on Speech Commands (SC) dataset. We show the accuracies of TCANet+LGCSiam, TCANet+WVC and TCANet+WVC+LGCSiam pretrained with 100% unlabeled SC dataset and fine-tuned with 100% and 5% labeled data .*

pretrained on 100% unlabeled SC	Acc. (100% SC)	Acc. (5% SC)
TCANet(Without pretrain)	96.85	88.24
TCANet+LGCSiam	96.97	89.92
TCANet+WVC	97.45	92.3
TCANet+WVC+LGCSiam	97.5	92.52

Table 2: *Comparing the influence of LGCSiam and WVC module for cross-domain representation learning. We show the accuracies of TCANet+LGCSiam, TCANet+WVC and TCANet+WVC+LGCSiam pretrained with 100% unlabeled AISHELL dataset and fine-tuned with 100% and 5% labeled SC data.*

pretrained on 100% unlabeled AISHELL	Acc. (100% SC)	Acc. (5% SC)
TCANet(without pretrain)	96.85	88.24
TCANet+LGCSiam	96.96	89.09
TCANet+WVC	97.32	92.13
TCANet+WVC+LGCSiam	97.45	92.33

Table 3: *Comparison of various KWS models on the Speech Commands dataset.*

Model	Acc.	Params
DS-CNN-S	94.4	24K
DS-CNN-M	94.9	140K
DS-CNN-L	95.4	420K
TC-ResNet8	96.1	66K
TC-ResNet8-1.5	96.2	145K
TC-ResNet14	96.2	137K
TC-ResNet14-1.5	96.6	305K
TCANet	96.85	65K
TCANet+WVC+LGCSiam	97.5	65K

5. Conclusion

In this paper, we investigate representation learning for keyword spotting with limited labeled data and limited device resources. We utilize self-training with pretrained acoustic model, and design a contrastive learning structure for keyword spotting. The experiments show the proposed representation learning modules are useful to improve accuracy with a small labeled dataset.

6. References

[1] S. Sigtia, R. Haynes, H. Richards, E. Marchi, and J. Bridle, "Efficient voice trigger detection for low resource hardware." in *Interspeech*, 2018, pp. 2092–2096.

[2] M. Sun, D. Snyder, Y. Gao, V. K. Nagaraja, M. Rodehorst, S. Panchapagesan, N. Strom, S. Matsoukas, and S. Vitaladevuni, "Compressed time delay neural network for small-footprint keyword spotting." in *Interspeech*, 2017, pp. 3607–3611.

[3] 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.

[4] S. Panchapagesan, M. Sun, A. Khare, S. Matsoukas, A. Mandal, B. Hoffmeister, and S. Vitaladevuni, "Multi-task learning and weighted cross-entropy for dnn-based keyword spotting." in *Interspeech*, vol. 9, 2016, pp. 760–764.

[5] 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.

[6] Y. Zhuang, X. Chang, Y. Qian, and K. Yu, "Unrestricted vocabulary keyword spotting using lstm-ctc." in *Interspeech*, 2016, pp. 938–942.

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

[8] Y. Zhang, N. Suda, L. Lai, and V. Chandra, "Hello edge: Keyword spotting on microcontrollers," *arXiv preprint arXiv:1711.07128*, 2017.

[9] G. Tucker, M. Wu, M. Sun, S. Panchapagesan, G. Fu, and S. Vitaladevuni, "Model compression applied to small-footprint keyword spotting." in *Interspeech*, 2016, pp. 1878–1882.

[10] A. Coucke, M. Chlieh, T. Gisselbrecht, D. Leroy, M. Poumeyrol, and T. Lavril, "Efficient keyword spotting using dilated convolutions and gating," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6351–6355.

[11] 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," *arXiv preprint arXiv:1904.03814*, 2019.

[12] "Bert: Pre-training of deep bidirectional transformers for language understanding."

[13] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *Advances in Neural Information Processing Systems*, vol. 33, pp. 12 449–12 460, 2020.

[14] A. Van den Oord, Y. Li, O. Vinyals *et al.*, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, vol. 2, no. 3, p. 4, 2018.

[15] S. Schneider, A. Baevski, R. Collobert, and M. Auli, "wav2vec: Unsupervised pre-training for speech recognition," *arXiv preprint arXiv:1904.05862*, 2019.

[16] W.-N. Hsu, Y.-H. H. Tsai, B. Bolte, R. Salakhutdinov, and A. Mohamed, "Hubert: How much can a bad teacher benefit asr pre-training?" in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 6533–6537.

[17] M. Huang, Y. You, Z. Chen, Y. Qian, and K. Yu, "Knowledge distillation for sequence model." in *Interspeech*, 2018, pp. 3703–3707.

[18] M. Mazumder, C. Banbury, J. Meyer, P. Warden, and V. J. Reddi, "Few-shot keyword spotting in any language," *arXiv preprint arXiv:2104.01454*, 2021.

[19] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[20] X. Chen and K. He, "Exploring simple siamese representation learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 15 750–15 758.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

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

[23] H. Bu, J. Du, X. Na, B. Wu, and H. Zheng, "Aishell-1: An open-source mandarin speech corpus and a speech recognition baseline," in *2017 20th Conference of the Oriental Chapter of the International Coordinating Committee on Speech Databases and Speech I/O Systems and Assessment (O-COCOSDA)*. IEEE, 2017, pp. 1–5.

[24] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[25] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.