Speech Augmentation Based Unsupervised Learning for Keyword Spotting

Jian Luo¹, Jianzong Wang¹*, Ning Cheng¹, Haobin Tang¹,², Jing Xiao¹

¹Ping An Technology (Shenzhen) Co., Ltd.
²University of Science and Technology of China

Abstract—In this paper, we investigated a speech augmentation based unsupervised learning approach for keyword spotting (KWS) task. KWS is a useful speech application, yet also heavily depends on the labeled data. We designed a CNN-Attention architecture to conduct the KWS task. CNN layers focus on the local acoustic features, and attention layers model the long-time dependency. To improve the robustness of KWS model, we also proposed an unsupervised learning method. The unsupervised loss is based on the similarity between the original and augmented speech features, as well as the audio reconstructing information. Two speech augmentation methods are explored in the unsupervised learning: speed and intensity. The experiments on Google Speech Commands V2 Dataset demonstrated that our CNN-Attention model has competitive results. Moreover, the augmentation based unsupervised learning could further improve the classification accuracy of KWS task. In our experiments, with augmentation based unsupervised learning, our KWS model achieves better performance than other unsupervised methods, such as CPC, APC, and MPC.

Index Terms—Speech Augmentation, Unsupervised Learning, Keyword Spotting

I. INTRODUCTION

Keyword Spotting (KWS) is a useful speech application in real-world scenarios. KWS aims at detecting a relatively small set of pre-defined keywords in an audio stream, which usually exists on the interactive agents. The KWS systems usually have two kinds of applications: Firstly, it can detect the startup commands, such as “hey Siri” or “OK, Google”, providing explicit cues for interactions. Secondly, KWS can help to detect some sensitive words to protect the privacy of the speaker. Therefore, highly accurate and robust KWS systems can be of great significance to real speech applications [1]–[3].

Recently, extensive literature research on KWS has been published [4]–[6]. As a traditional solution, keyword/filler Hidden Markov Model (HMM) has been widely applied to KWS tasks, and remains competitive results [7]. In this generative approach, an HMM model is trained for each keyword, while another HMM model is trained from not-keyword speech segments. At inference, the Viterbi decoding is required, which might be computationally expensive depending on the HMM topology. In recent years, deep learning models have gained popularity on the KWS task, which show better performance than traditional approaches. Google proposed to use Deep Neural Networks (DNN) to predict sub-keyword targets. It uses the posterior processing method to generate the final confidence score, and outperforms the HMM-based system [8].

In contrast, Convolutional Neural Networks (CNN) is more attractive, because DNN ignores the input topology, but audio features could have a strong dependency in time or frequency domains [9]–[11]. However, there is a potential drawback that CNN might not model much contextual information. Also, Recurrent Neural Networks (RNN) with Connectionist Temporal Classification (CTC) loss was also investigated for KWS. However, the limitation of RNN is that it directly models the speech features without learning local structure between successive time series and frequency steps [12]. There are also some works that combined CNN and RNN to improve the accuracy of KWS. For example, Convolutional Recurrent Neural Networks (CRNN) and Gated Convolutional Long Short-Term Memory (LSTM), achieved better performance than that of only using CNN or RNN [13]. In recent years, many researchers focus on the transformer-based models with self-attention mechanism. As a typical model, Bidirectional Encoder Representations from Transformer (BERT) has been proven to be an effective model in many Natural Language Processing (NLP) tasks [14]–[16]. The transformer-based models have also obtained much application in Automatic Speech Recognition (ASR) tasks [17], [18]. In this work, we introduced transformer to the network architecture of KWS. We think that transformer encoder has great advantage on the speech representation, and established a CNN-Attention based network to deal with the KWS task. The CNN helps network to learn the local feature, and the self-attention mechanism of transformer focuses on the long-time information.

The above supervised approaches have acquired good performance, but these models require a lot of labeled datasets. Obviously, for KWS task, the negative samples could be more procurable than positive samples, meaning that the positive samples might not be obtained easily. Especially when the keyword changes, it requires much time to collect the positive target samples, and the existing models might not easily transfer to other KWS models. In this paper, we focus on the unsupervised learning approach to alleviate this problem. The unsupervised learning mechanism allows the neural network to be trained on unlabeled datasets. With unsupervised learning, the performance of downstream task could be improved with limited labeled datasets. Unsupervised learning has made great success in the audio, image and text tasks [19]. In speech area, researchers also proposed some unsupervised pre-training algorithms [20]–[22]. Contrastive Predictive Coding

*Corresponding author: Jianzong Wang, jzwang@188.com
Propose a CNN-Attention architecture for keyword spotting task, having competitive results on Google Speech Commands V2 Dataset.

Design an unsupervised loss based on the Mean Square Error (MSE) to measure the distance between the original and augmented speech.

Define a speech augmentation based unsupervised learning approach, utilizing the similarity between the bottleneck layer feature, as well as the audio reconstructing information for auxiliary training.

The rest of the paper is organized as follows. Sec. II highlights the related prior works about data augmentation, unsupervised learning, and other methodologies of KWS tasks. Sec. III describes the proposed model architecture and augmentation based unsupervised learning loss. Sec. IV reports the experimental results compared with other pre-training methods. We also discuss relationship between pre-training steps and performance of downstream KWS tasks. In Sec. V, we conclude with the summary of the paper and future works.
III. PROPOSED METHOD

A. KWS Model Architecture

The keyword spotting task could be described as a sequence classification task. The keyword spotting network maps an input audio sequence $X = (x_0, x_1, \ldots, x_T)$ to a limited set of keyword classes $Y \in \{1, \ldots, S\}$. In which, $T$ is the number of audio frames and $S$ is the number of classes. Our proposed model architecture for keyword spotting is shown in Fig 1. The network contains five parts: (1) CNN Block, (2) Transformer Block, (3) Feature Selecting Layer, (4) Bottleneck Layer, and (5) Project Layer.

The CNN block consists of several 2D-convolutional layers, handling the local variance on time and spectrum axes.

$$E_{cnn} = 2D\text{Conv}_{xN}(X)$$  
(1)

In which, $N$ is the number of convolutional layers. Then, the CNN output $E_{cnn}$ is inputted to the transformer block, to capture long-time information with self-attention mechanism.

$$E_{tran} = \text{SelfAttention}_{xM}(E_{cnn})$$  
(2)

In which, $M$ is the number of self-attention layers. After transformer block, we designed a feature selecting layer to extract keyword information from sequence $E_{tran}$.

$$E_{feat} = \text{Concat}(E_{tran}[T-r, T])$$  
(3)

In feature selecting layer, we firstly collect last $r$ frames of $E_{tran}$. And then, we concatenate all the collected frames together, into one feature vector $E_{feat}$. After feature selecting layer, we use a bottleneck layer and a project layer, projecting the hidden states to the predicted classification classes $\hat{Y}$.

$$E_{bn} = \text{FC}_{bn}(E_{feat})$$  
(4)

Finally, the the cross-entropy (CE) loss for supervised learning and model fine-tuning is calculated via predicted classes $\hat{Y}$ and ground truth classes $Y$.

$$L_{ce} = \text{CE}(Y, \hat{Y})$$  
(5)

B. Augmentation Method

Data augmentation are the most common used methods to promote the robustness and performance of the model in speech tasks. In this work, speed and volume based augmentation are investigated in the unsupervised learning of keyword spotting. For a given audio sequence $X$, we denote it as the amplitude $A$ and time index $t$.

$$X = A(t)$$  
(7)

For speed augmentation, we set a speed ratio $\lambda_{\text{speed}}$ to adjust the speed of $X$.

$$X^{\text{aug}} = A(\lambda_{\text{speed}}t)$$  
(8)

For volume augmentation, we also set an intensity ratio $\lambda_{\text{volume}}$ to change the volume of $X$.

$$X^{\text{aug}} = \lambda_{\text{volume}}A(t)$$  
(9)

With different ratios $\lambda_{\text{speed}}$ and $\lambda_{\text{volume}}$, we could obtain multiple speech sequence pairs $(X, X^{\text{aug}})$, to train the audio representation network with unsupervised learning. We think that speech utterances at different speed or volume should have similar high-level feature representation for KWS tasks.

C. Unsupervised Learning Loss

The overall architecture of augmentation based unsupervised learning is shown in Fig 2. Similar to other unsupervised methods, the proposed approach also consists of two stages: (1) pre-training on unsupervised data, and (2) fine-tuning on supervised KWS data. In the pre-training stage, the bottleneck feature was obtained through training the unlabeled speech. In fine-tuning stage, the extracted bottleneck features are used for KWS prediction.

In the pre-training stage, the pair speech data $(X, X^{\text{aug}})$ are inputted into the CNN-Attention models respectively, but with the same model parameters. Because $X^{\text{aug}}$ comes from $X$, our designed unsupervised methods expect that $X$ and $X^{\text{aug}}$ will output similar high-level bottleneck features. It means that no matter how fast or how loud a speaker says, the content of the speech is the same. Thus, the optimization of network needs to reflect the similarity of $X$ and $X^{\text{aug}}$. We choose the Mean Square Error (MSE) $L_{sim}$ to measure the distance between the output of $X$ and $X^{\text{aug}}$.

$$L_{sim} = \frac{1}{U_{bn}} \sum_{u=0}^{U_{bn}} |E_{bn}(u) - E_{bn}^{\text{aug}}(u)|^2$$  
(10)

Where $U_{bn}$ represents the dimension of the bottleneck feature vector. $E_{bn}$ and $E_{bn}^{\text{aug}}$ are the output of bottleneck layer of original speech $X$ and augmented speech $X^{\text{aug}}$ respectively.
In addition, the designed network has another branch for auxiliary training, which predicts the average feature of the input speech segment. This branch guides the network to learn the intrinsic feature of the speech utterance. We firstly compute the average vector of the input Fbank vector $X$ alongside the time axis $t$. Then, we use another reconstructing layer attached to the bottleneck layer, to reconstruct the average Fbank vector $\tilde{X}$. We also use MSE loss $L_x$ to calculate the similarity between these two audio vectors alongside the feature dimension $U_x$.

$$\mathcal{X} = \frac{1}{T} \sum_t (X)$$

$$\tilde{X} = \text{FC}_{\text{reconstruct}}(E_{bn})$$

$$L_x = \frac{1}{U_x} \sum_{u=0}^{U_x} |\mathcal{X}(u) - \tilde{X}(u)|^2$$ (11)

In which, $U_x$ represents the dimension of Fbank feature vector, and $\mathcal{X}$ denotes the average vector of $X$. Similarly, the loss $L_x^{aug}$ between the augmented average audio $\tilde{X}^{aug}$ and the augmented feature $\tilde{x}^{aug}$ could be defined as follows:

$$L_x^{aug} = \frac{1}{U_x} \sum_{u=0}^{U_x} |\mathcal{X}^{aug}(u) - \tilde{x}^{aug}(u)|^2$$ (12)

Therefore, the final loss function $L_{ul}$ of the unsupervised learning (UL) consists of the above three losses $L_{sim}$, $L_x$, and $L_x^{aug}$.

$$L_{ul} = \lambda_1 L_{sim} + \lambda_2 L_x + \lambda_3 L_x^{aug}$$ (13)

Where $\lambda_1$, $\lambda_2$, $\lambda_3$ are factor ratio of each loss component.

In fine-tuning stage, the branch of average feature prediction is removed. A project layer and a softmax layer are added after the bottleneck layer to make the KWS prediction. In the fine-tuning, the parameters of original network could be fixed or updated. In our experiments, we found that updating all the parameters could help to improve the performance. Thus, we choose to update all parameters in the fine-tuning stage.

### IV. EXPERIMENTS

In this section, we evaluated the proposed method in keyword spotting tasks. We implemented our CNN-Attention model with supervised training and compared it with Google’s model. We also made an ablation study, to explore the effect of speed and volume augmentation on unsupervised learning. What’s more, other unsupervised learning methods are compared with our approach, including CPC, APC, MPC. When implementing these approaches, we used the network and hyperparameters in their publications, but all experimental tricks were not leveraged [23]–[25]. We also discuss the impact of different pre-training steps on the performance and convergence of downstream KWS task.

#### A. Datasets

We used Google’s Speech Commands V2 Dataset [41] for evaluating the proposed models. The dataset contains about 106000 one-second or more long utterances. Total 30 short words were recorded by thousands of different people, as well as background noise such as pink noise, white noise, and human-made sounds. The KWS task is to discriminate among 12 classes: "yes", "no", "up", "down", "left", "right", "on", "off", "stop", "go", unknown, or silence. The dataset was split into training, validation, and test sets, with 80% training, 10% validation, and 10% test. This results in about 37000 samples for training, and 4600 each for validation and testing. We

| TABLE I | MODEL CONFIGURATIONS |
|----------------|--------------------------|
| Unit Name | Hyperparameters |
| CNN Blocks | $M = 2$ layers, $3 \times 3$ kernel, $2 \times 2$ stride, 32 channels |
| Transformer Block | $N = 2$ layers, dimension $= 320$, 4 head, feedforward $= 1024$ |
| Feature Selecting Layer | Last $r = 2$ frames, $2 \times 320$ dimension |
| Bottleneck Layer | one FC layer, 800 dimension |
| Project Layer | one FC layer, 12 dimension softmax |
| Reconstruct Layer | one FC layer, 40 dimension softmax |
| Factor Ratio | $\lambda_1 = 0.9$, $\lambda_2 = 0.05$, $\lambda_3 = 0.05$ |
used the real noisy data HuNonspeech\(^1\) to corrupt the original speech. In the experiments, the Aurora4 tools were used to implement this strategy\(^2\). Each utterance will be randomly corrupted by public 100 kinds of noise in HuNonspeech. Each utterance has a level of 0-20 dB Signal Noise Ratio (SNR), and all datasets have an average 10 dB SNR.

Similar to other unsupervised methods, a large unlabeled corpus, 100 hours of Librispeech \[^42\] clean speech were also leveraged to pre-train the network by unsupervised learning. Firstly, the long utterances were split up into 1 second segments, keeping consistent with Speech Commands datasets. Nextly, the clean segments were also mixed with noisy HuNonspeech data by Aurora 4 tools, and the corrupted mechanism was as same as the Speech Commands.

\subsection*{B. Experimental Setups}

The acoustic features were 40-dimensional log-mel filter-bank with 30 ms frame length and 10 ms frame shift. The detailed hyperparameters of our proposed network were shown in Table I. For training the KWS model, all of the matrix weights are initialized with random uniform initialization, and the bias parameters are initialized with the constant value \(0.1\). In our experiments, we trained all the networks with Adam optimizer for 30k steps with a batchsize 200 until the loss becomes little change. In addition, the factor ratios of loss \(\lambda_1\), \(\lambda_2\), and \(\lambda_3\) are set to 0.9, 0.05, 0.05 respectively.

To demonstrate the effectiveness of our proposed model, we investigated several other approaches for comparison. For supervised learning, we used Sainath and Parada’s model by Google [43] as the baseline model. The Google blog post released the Sainath and Parada’s model implemented by TensorFlow. For unsupervised learning, we compared our method with other pre-training models:

- **Contrastive Predictive Coding (CPC)** [23]: Through an unsupervised mechanism by utilizing next step prediction, CPC learns representations from high-dimensional signal.

- **Autoregressive Predictive Coding (APC)** [24]: APC also belongs to the family of predictive models. APC directly optimizes L1 loss between input sequence and output sequence. APC has proved an effective method in recent language model pre-training task and speech representation.

- **Masked Predictive Coding (MPC)** [25]: Inspired by BERT, MPC uses Masked Language Model (MLM) structure to perform predictive coding on Transformer based models. Similar to BERT, 15% of feature frames in each utterance are chosen to be masked during the pre-training procedure. Among these chosen frames, 80% are replaced with zero vectors, 10% are replaced with random positions, and the rest remain unchanged. L1 loss is computed between masked input features and encoder output at corresponding position. Dynamic masking was also adopted where the masking pattern is generated when a sequence is fed into the model.

\subsection*{C. Results}

Table II lists the experimental results of supervised learning with Speech Commands dataset. We firstly implemented the Google’s Sainath and Parada model by the original TensorFlow recipes, achieving the accuracy of 84.7%. Secondly, our CNN-Attention model is implemented by supervised loss \(\mathcal{L}_{ce}\) without any augmented data and achieved 0.6% higher accuracy than Google’s model. It is proved that our designed CNN-Attention architecture is effective for KWS task. Finally, after adding speed and volume augmentation to speech, we got a higher accuracy. It corresponds with the existing research that augmented dataset is helpful for improving the performance.

\begin{table}[h]
\centering
\caption{Results Comparison of KWS Model, Classification Accuracy (\%)}
\label{tab:results}
\begin{tabular}{llll}
\hline
Model Name & Supervised Training Data & Dev & Eval \\
\hline
Sainath and Parada (Google) & Speech Commands & 86.4 & 85.3 \\
CNN-Attention (ours) & Speech Commands & 87.9 & 87.2 \\
CNN-Attention + volume & speed augment (ours) & Speech Commands & 88.2 & 88.1 \\
\hline
\end{tabular}
\end{table}

\begin{table}[h]
\centering
\caption{Ablation Study, the effect of speed and volume augmentation, Classification Accuracy (\%)}
\label{tab:ablation}
\begin{tabular}{llllll}
\hline
Model Name & Pre-training Data & Fine-tuning Data & Dev & Eval \\
\hline
CNN-Attention + volume pre-training & Speech Commands & Speech Commands & 86.1 & 85.9 \\
CNN-Attention + speed pre-training & Speech Commands & Speech Commands & 87.8 & 86.9 \\
CNN-Attention + volume & speed pre-training & Speech Commands & 87.9 & 87.2 \\
CNN-Attention + volume pre-training & Librispeech-100 & Speech Commands & 86.3 & 86.0 \\
CNN-Attention + speed pre-training & Librispeech-100 & Speech Commands & 87.9 & 87.9 \\
CNN-Attention + volume & speed pre-training & Librispeech-100 & 88.2 & 88.1 \\
\hline
\end{tabular}
\end{table}

\(^1\)http://web.cse.ohio-state.edu/pnl/corpus/HuNonspeech/
\(^2\)http://aurora.hsnr.de/index-2.html
Table IV: Compared with Other Unsupervised Learning Methods, Classification Accuracy (%)

| Model Name                          | Pre-training Data | Fine-tuning Data | Dev  | Eval  |
|-------------------------------------|-------------------|------------------|------|-------|
| Contrastive Predictive Coding (CPC) [23] | Speech Commands   | Speech Commands  | 87.6 | 86.9  |
| Autoregressive Predictive Coding (APC) [24] | Speech Commands   | Speech Commands  | 87.2 | 86.5  |
| Masked Predictive Coding (MPC) [25]  | Speech Commands   | Speech Commands  | 87.0 | 86.7  |
| CNN-Attention + volume & speed pre-training (ours) | Speech Commands | Speech Commands | 87.9 | 87.2  |
| Contrastive Predictive Coding (CPC) [23] | Librispeech-100   | Speech Commands  | 87.8 | 87.4  |
| Autoregressive Predictive Coding (APC) [24] | Librispeech-100   | Speech Commands  | 87.7 | 87.5  |
| Masked Predictive Coding (MPC) [25]  | Librispeech-100   | Speech Commands  | 87.9 | 87.0  |
| CNN-Attention + volume & speed pre-training (ours) | Librispeech-100   | Speech Commands  | 88.2 | 88.1  |

Fig. 3. The results comparison with different pre-training steps. Different pre-training steps of unsupervised learning result in different accuracy performance and fine-tuning convergence. In our experiments, pre-training 30K steps have the highest classification accuracy, and fastest convergence.

D. Pre-training Analysis

More pre-training steps usually help to improve the performance of downstream tasks. To get a better understanding of our unsupervised approach, we also conducted experiments with different pre-training steps. The 5K, 10K, 20K, 30K pre-training steps were used for making this comparison. The performance of different steps is plotted in Fig 3.

We show the model training of supervised learning with these different steps of pre-training. Our experiments demonstrated that more pre-training steps are not only helpful for achieving better performance but also making downstream KWS task converge faster. Unsupervised learning with 30K steps has the highest classification accuracy and the fastest convergence. It also should be noted that the difference between 20K and 30K was very close, meaning that the pre-training steps are enough to obtain the desired performance.

V. Conclusion

This paper investigated unsupervised learning method for keyword spotting task. We designed a CNN-Attention architecture and achieved competitive results on the Speech Commands dataset. In addition, we proposed a speech augmentation based unsupervised learning approach for KWS. Our method uses speed and intensity augmentation to establish training pairs, and pre-trains the network via the similarity loss between the speech pair and the speech reconstructed loss. In our experiments, the proposed unsupervised approach could further improve the model performance, and outperform other unsupervised methods, such as CPC, APC and MPC. We also found that more pre-training steps are not only helpful for better performance but also for faster convergence. In future works, we are interested in applying the augmentation based unsupervised learning approach to other speech tasks, such as speaker verification and speech recognition.

VI. Acknowledgement

This paper is supported by the Key Research and Development Program of Guangdong Province under grant No. 2021B0101400003. Corresponding author is Jianzong Wang from Ping An Technology (Shenzhen) Co., Ltd (jzwang@188.com).
