Custom Mandarin Keyword Spotting with Extended Long Short-Term Memory

Haitao Cao, Xi Liu, Zhiguo Tan, Zhenlun Yang, and Xin Qin

***Abstract*—In real-world scenarios, Deep Neural Network (DNN)-powered Keyword Spotting (KWS) systems are typically engineered as lightweight architectures, optimizing for superior performance and low computational complexity in resource-limited devices. However, such lightweight designs often encounter limitations in generalization, particularly when it comes to customizing keywords. This paper presents a two-stage method to customize a Mandarin KWS system rapidly. First, we propose an embedding model to learn the embedding representations of general Mandarin keywords. Subsequently, we facilitate keyword customization with the generalization capability of embedding models through few-shot transfer learning. To improve performance further, in the embedding model, we introduce two scale blocks to fuse acoustic features and employ an Enhanced Extended Long Short-Term Memory (ExLSTM) as the backbone. Experimental results on both English and Mandarin keyword datasets highlight the advantages of the proposed embedding model. In addition, we conduct keyword customization on a self-recorded dataset containing 10 Mandarin keywords. The impressive average accuracy of 97.45% with merely five target samples demonstrates the effectiveness of our method.**

***Index Terms*—keyword spotting, extended long short-term memory, feature fusion, embedding representations, transfer learning**

I. Introduction

KEYWORD Spotting (KWS) currently stands as a focus within speech signal processing research. One of its notable applications is in voice wake-up systems, which have been widely integrated into various products in human-machine interaction scenarios. In recent years, the voice-control functions in mobile phones and smart speakers have sparked significant interest among Chinese consumers. The huge market propels extensive research efforts in the domain of Mandarin KWS [1], [2].

Thanks to the rapid advancements in Deep Learning (DL) technologies, the Deep Neural Network (DNN) has made remarkable improvements in enhancing the performance of KWS systems [3], [4], [5]. Consequently, numerous lightweight KWS systems, characterized by high accuracy, low false alarm rate [6], and low latency [7], have been successfully deployed in resource-limited devices.

Despite these significant advancements, a common limitation of many KWS systems lies in their reliance on predefined keywords, thereby preventing users from defining personalized keywords of their preference. This constraint not only raises privacy concerns, as unauthorized users may inadvertently activate personal devices [5], [8], but also brings challenges for users with accents, potentially diminishing their overall user experience due to the difficulty of a standard pronunciation.

Even though keyword customization has garnered growing attention in the KWS field [4], [9], the generalization capability of DNN is often limited to its lightweight architectures [10], [11], which will compromise the performance of customization. To address this challenge, this paper introduces a two-stage method to rapidly customize a Mandarin KWS system. In the first stage, a novel embedding model is proposed to effectively learn embedding representations from a large-vocabulary dataset. Subsequently, the second stage leverages few-shot transfer learning to efficiently perform the keyword customization process. Our main contributions are listed as follows:

1) To enhance the robustness of the embedding model, we design two simple scale blocks based on Convolutional Neural Networks (CNN) to fuse acoustic features.
2) An Enhanced Extended Long Short-Term Memory (ExLSTM) is proposed as the backbone of the embedding model for efficient feature extraction.
3) We compile a real-life dataset to evaluate the performance of our customized KWS systems. The promising outcomes demonstrate the effectiveness of our approach.

The rest of this paper is organized as follows: Section II discusses related work, Section III details our proposed framework, Section IV presents experimental details and results, and Section V concludes the paper.

II. Related Work

In the field of DL, KWS is conventionally formulated as an audio classification task. This process involves extracting acoustic features, which represent the intrinsic characteristics of the audio signal, from raw data for subsequent analysis by DNN models. The traditional acoustic features are often derived from Short-Time Fourier Transform (STFT) methods, including Mel-spectrogram and Mel-Frequency Cepstral Coefficients (MFCC). These handcrafted features effectively capture both the temporal and spectral content

Manuscript received March 25, 2024; revised October 10, 2024.

This work was supported in part by the Education Department of Guangdong Province of China (Research Grant no. 2022ZDZX1065, 2023ZDZX1072), the Guangzhou Intelligent Education Technology Collaborative Innovation Center Construction Project (Research Grant no. 2023B04J0004), and the Guangdong Innovative Projects with Characteristics in Colleges and Universities (Research Grant no. 2022KTSCX292).

Haitao Cao is a lecturer of the School of Information Engineering, Guangzhou Panyu Polytechnic, Guangzhou 511483, China. (e-mail: hcaogm@gmail.com)

Xi Liu is an associate professor of the School of Information Engineering, Guangzhou Panyu Polytechnic, Guangzhou 511483, China. (e-mail: liux239@mail2.sysu.edu.cn)

Zhiguo Tan is an associate professor of the School of Information Engineering, Guangzhou Panyu Polytechnic, Guangzhou 511483, China. (e-mail: tanzhiguo136@163.com)

Zhenlun Yang is an associate professor of the School of Information Engineering, Guangzhou Panyu Polytechnic, Guangzhou 511483, China. (corresponding author to provide e-mail: 54635097@qq.com)

Xin Qin is an undergraduate student of the Institute of Big Data and Internet Innovation, Hunan University of Technology and Business, Changsha 410205, China. (e-mail: 1027015887@qq.com)

of audio signals, exhibiting robustness to background noise [12]. Consequently, they have been widely adopted in KWS applications [7], [13].

However, it has been observed that STFT-based features inherently suffer from the loss of large-scale information owing to their fixed resolution. Furthermore, these features exhibit a lack of translation invariance, thereby reducing their robustness in real-world environments [14]. To address these shortcomings, wavelet scattering was proposed as a more effective transformation for audio signals, demonstrating promising outcomes in audio classification tasks [15], [16]. This advancement offers a solution to the limitations of traditional acoustic features, enhancing the performance of KWS systems in diverse and challenging conditions.

Meanwhile, significant efforts have been directed towards designing novel DNN architectures to improve KWS performance. For instance, the lightweight models like DFSMN [3], MatchboxNet [17], and AdderNet [18] were developed specifically for real-life applications, catering to the demands of efficiency and practicality. Additionally, attention mechanisms and Transformer-based structures have shown success in handling the challenges of KWS, as evidenced by studies like [7] and [19]. Nevertheless, their quadratic complexity concerning the length of input sequences limits their scalability and applicability in resource-limited scenarios.

Several studies have delved into exploring efficient model optimization techniques to advance the field [20]. One promising approach is the utilization of low-rank approximation, which has demonstrated advantages in significantly reducing model size, as seen in [21]. Furthermore, techniques such as broadcasted residual learning [22] and advanced convolutional methods (e.g., 1-dimensional convolution, depthwise convolution, dilated convolution) were employed to enhance computational efficiency [23], [24].

As previously noted, keyword customization has emerged as a hot topic in KWS applications. When it comes to traditional DL approaches, the development of a user-defined KWS system requires a substantial collection of target samples and entails retraining an end-to-end DNN model, which will be time-consuming. Recently, the Query-by-Example (QbyE) method has garnered significant attention due to its ability to facilitate keyword customization on resource-limited devices [4], [8]. However, ensuring the robustness of QbyE in real-world applications remains a challenge [9].

Researchers also approached KWS as a subset of Large Vocabulary Continuous Speech Recognition (LVCSR), where keyword customization revolves around identifying target phonemes [2]. Nevertheless, these LVCSR-based models often consume significant computing resources. Alternatively, transfer learning has emerged as a practical solution for customized KWS, offering advantages in customization across different languages with minimal effort and resources [25].

III. Proposed Method

Our proposed method is illustrated in Fig. 1. It consists of two stages, namely, the training of the embedding model and few-shot transfer learning, which will be described in detail next.

A. Embedding Model

The embedding model is trained to learn embedding representations of diverse keywords from a large-vocabulary dataset encompassing general keyword patterns. The objective of this training is to enhance the model's generalization capability for its effectiveness in keyword customization. The model architecture consists of three main components:

1) **Feature Extractor**: The feature extractor computes a 128-dimensional Mel-spectrogram and a 32-dimensional MFCC feature, leveraging a Hann window of 32 milliseconds with a 16-millisecond overlap. This integrated functionality within the embedding model streamlines the process, facilitating an end-to-end architecture for future deployment.
2) **Scale Blocks**: The scale blocks comprise two convolutional blocks designed to scale the frequency dimensions and merge the channel dimensions of the extracted features. Each block uses a CNN structure to enhance the model's ability to capture higher-level abstractions of the acoustic features.
   Specifically, the Mel-spectrogram and MFCC features are scaled to 32-dimensional feature maps and subsequently concatenated, forming an 8-channel input for the backbone network. Within the two scale blocks, the convolutional layers use a kernel size of (5,5) and a stride of (1,1), ensuring that the input size is maintained through the implementation of same-padding. The number of output channels for these convolutional layers is set to 4, which is then followed by batch normalization, ReLU activation function, and a dropout layer with a dropping rate of 0.5. Notably, "Scale Block 1" incorporates an additional max-pooling layer with a pooling size of (1,2) to downsample the Mel-spectrogram.
3) **Backbone**: The backbone network aims to capture more comprehensive patterns from the fused acoustic features. For comparison, various DNN structures were deployed as backbone candidates in the experiment, with their specific details outlined in Appendix A. Within this study, our primary focus lies in refining and enhancing the Extended Long Short-Term Memory (xLSTM) model, as described in the subsequent section. The output layer of the backbone is a dense layer equipped with a softmax activation function for a multi-category audio classification task.

B. Extended Long Short-Term Memory

1) Vanilla LSTM: The Long Short-Term Memory (LSTM) network was introduced to mitigate the training challenges commonly associated with Recurrent Neural Networks (RNNs). LSTM incorporates the idea of the constant error carousel and gating mechanisms that allow for better retention of long-term dependencies, leading to numerous successful applications across the domain of DL [26], [27]. Given input features $\boldsymbol{x}_t$ at the current frame t and the hidden state $\boldsymbol{h}_{t-1}$ from the previous frame $t-1$, the update rules

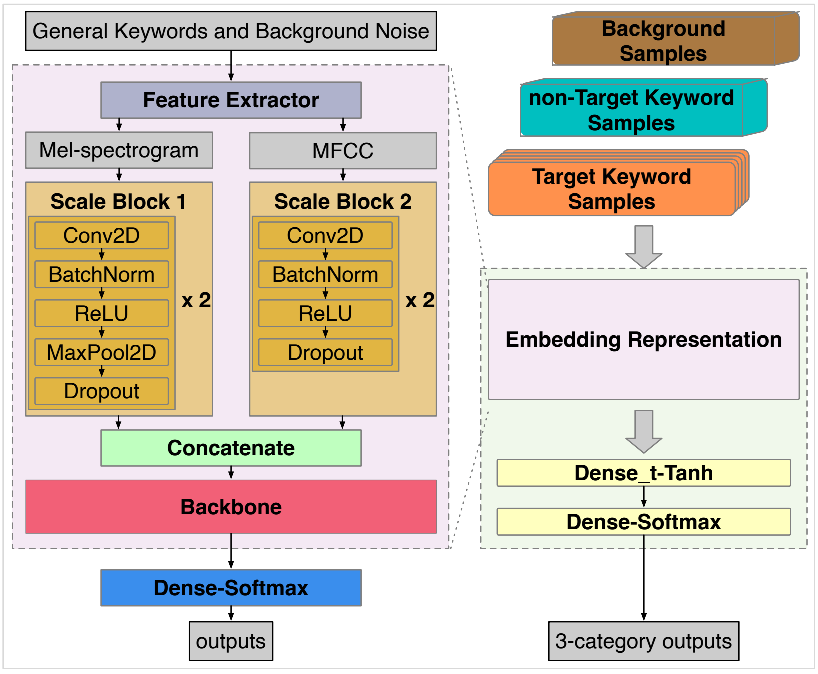

Fig. 1. The diagram of the proposed method. The left panel illustrates the embedding model with scale blocks. In the ablation experiments, the Mel-spectrogram or MFCC features are directly fed into the backbone, bypassing "Scale Block 1" and "Scale Block 2". The right panel depicts the few-shot transfer learning process for customizing KWS models.

of a vanilla LSTM memory cell can be formulated as [28]:

$$\text{input gate}: \boldsymbol{i}_t = \sigma\left(\boldsymbol{W}_i \boldsymbol{x}_t + \boldsymbol{R}_i \boldsymbol{h}_{t-1} + \boldsymbol{b}_i\right) \tag{1}$$

$$\text{forget gate}: \boldsymbol{f}_t = \sigma\left(\boldsymbol{W}_f \boldsymbol{x}_t + \boldsymbol{R}_f \boldsymbol{h}_{t-1} + \boldsymbol{b}_f\right) \tag{2}$$

$$\text{output gate}: \boldsymbol{o}_t = \sigma\left(\boldsymbol{W}_o \boldsymbol{x}_t + \boldsymbol{R}_o \boldsymbol{h}_{t-1} + \boldsymbol{b}_o\right) \tag{3}$$

$$\text{cell input}: \boldsymbol{z}_t = \varphi\left(\boldsymbol{W}_z \boldsymbol{x}_t + \boldsymbol{R}_z \boldsymbol{h}_{t-1} + \boldsymbol{b}_z\right) \tag{4}$$

$$\text{cell state}: \boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \boldsymbol{z}_t \tag{5}$$

$$\text{hidden state}: \boldsymbol{h}_t = \boldsymbol{o}_t \odot \psi\left(\boldsymbol{c}_t\right) \tag{6}$$

For the input gate, forget gate, output gate, and cell input in Eqs. (1)-(4), there are respective input weight matrices $\boldsymbol{W}$ applied to the inputs $\boldsymbol{x}_t$, and recurrent weight matrices $\boldsymbol{R}$ applied to the hidden state $\boldsymbol{h}_{t-1}$, along with corresponding bias vectors $\boldsymbol{b}$. σ represents the Sigmoid function. The symbol $\odot$ in Eqs. (5) and (6) denotes the point-wise multiplication of two vectors. φ and ψ are the activation functions (typically Tanh).

However, when handling very long sequences or requiring higher-dimensional features, LSTM exhibits several limitations, which can be categorized into three main aspects [29]:

- *Inability to revise storage decisions*. LSTM encounters difficulties in efficiently updating stored values when confronted with similar information, potentially compromising its performance in tasks that require dynamic updating of stored information.
- *Limited storage capacity*. LSTM compresses information into scalar values, inherently limiting its capacity to effectively store and retrieve complex data patterns, especially in scenarios involving rare labels or long-range dependencies.
- *Lack of parallelizability due to memory mixing*. The memory mixing mechanism in LSTM involves dependencies between hidden states across temporal frames, thereby enforcing sequential processing and hindering computation parallelization.

To address these limitations, the xLSTM was proposed to enhance the capabilities of the vanilla LSTM by introducing new gating mechanisms and storage structures [29]. It includes two types of modules: Scalar LSTM (sLSTM) and Matrix LSTM (mLSTM).

2) Scalar LSTM: sLSTM innovatively adopts exponential activation functions for both its input and forget gates, deviating from the conventional Sigmoid functions utilized in vanilla LSTM architectures, as shown in Eqs. (7)-(8). This exponential gating empowers the model to dynamically adapt to changes, facilitating rapid integration of new information and more efficient updating of stored memories. Additionally, sLSTM introduces a normalizer state to enhance model stability during the processing of long sequences, as illustrated in Eqs. (9)-(10). The combination of exponential gating and the normalizer state enables the model to make more adaptive storage decisions, enhancing its overall performance.

$$\text{input gate}: \boldsymbol{i}_t = \exp\left(\boldsymbol{W}_i \boldsymbol{x}_t + \boldsymbol{R}_i \boldsymbol{h}_{t-1} + \boldsymbol{b}_i\right) \tag{7}$$

$$\text{forget gate}: \boldsymbol{f}_t = \exp\left(\boldsymbol{W}_f \boldsymbol{x}_t + \boldsymbol{R}_f \boldsymbol{h}_{t-1} + \boldsymbol{b}_f\right) \tag{8}$$

$$\text{normalizer state}: \boldsymbol{n}_t = \boldsymbol{f}_t \odot \boldsymbol{n}_{t-1} + \boldsymbol{i}_t \tag{9}$$

$$\text{hidden state}: \boldsymbol{h}_t = \boldsymbol{o}_t \odot \left(\boldsymbol{c}_t \odot \boldsymbol{n}_t^{-1}\right) \tag{10}$$

3) Matrix LSTM: In line with sLSTM, mLSTM utilizes the same activation functions for its input, forget, and output gates, with the exclusion of the recurrent weight matrix $\boldsymbol{R}$ to facilitate parallelization, as shown in Eqs. (11)-(13). Drawing from Transformer terminology, mLSTM introduces query, key, and value vectors through linear projections of the input $\boldsymbol{x}_t$, as specified in Eqs. (14)-(16), where d denotes the dimensionality of these vectors. In mLSTM, the forget gate $\boldsymbol{f}_t$ and the input gate $\boldsymbol{i}_t$ corresponds to the decay rate and the learning rate of the covariance update rule respectively

[29], as formulated in Eq. (17).

Notably, the cell state of mLSTM transitions from a vector $\boldsymbol{c} \in \mathbb{R}^d$, as seen in the vanilla LSTM (Eq. (5)), to a matrix $\boldsymbol{C} \in \mathbb{R}^{d \times d}$, significantly enhancing the storage capacity of models. Furthermore, the normalizer state $\boldsymbol{n}_t$ is computed as a weighted sum of the key vectors (Eq. (18)), while the output state scales the retrieved vector to yield the hidden state $\boldsymbol{h}_t$ (Eq. (19)).

$$\text{input gate}: \boldsymbol{i}_t = \exp\left(\boldsymbol{W}_i \boldsymbol{x}_t + \boldsymbol{b}_i\right) \tag{11}$$
$$\text{forget gate}: \boldsymbol{f}_t = \exp\left(\boldsymbol{W}_f \boldsymbol{x}_t + \boldsymbol{b}_f\right) \tag{12}$$
$$\text{output gate}: \boldsymbol{o}_t = \sigma\left(\boldsymbol{W}_o \boldsymbol{x}_t + \boldsymbol{b}_o\right) \tag{13}$$
$$\text{query input}: \boldsymbol{q}_t = \boldsymbol{W}_q \boldsymbol{x}_t + \boldsymbol{b}_q \tag{14}$$
$$\text{key input}: \boldsymbol{k}_t = \frac{1}{\sqrt{d}} \boldsymbol{W}_k \boldsymbol{x}_t + \boldsymbol{b}_k \tag{15}$$
$$\text{value input}: \boldsymbol{v}_t = \boldsymbol{W}_v \boldsymbol{x}_t + \boldsymbol{b}_v \tag{16}$$
$$\text{cell state}: \boldsymbol{C}_t = \boldsymbol{f}_t \odot \boldsymbol{C}_{t-1} + \boldsymbol{i}_t \left(\boldsymbol{v}_t \boldsymbol{k}_t^\top\right) \tag{17}$$
$$\text{normalizer state}: \boldsymbol{n}_t = \boldsymbol{f}_t \odot \boldsymbol{n}_{t-1} + \boldsymbol{i}_t \odot \boldsymbol{k}_t \tag{18}$$
$$\text{hidden state}: \boldsymbol{h}_t = \boldsymbol{o}_t \odot \frac{\boldsymbol{q}_t \boldsymbol{C}_t}{\max\left(\left|\boldsymbol{n}_t \boldsymbol{q}_t^\top\right|, 1\right)} \tag{19}$$

In summary, the exponential gating mechanism and normalization significantly enhance LSTM's capacity to dynamically update stored information. The introduction of a high-dimensional cell state and parallel computing further boosts LSTM's storage capability and accelerates the training process. Moreover, xLSTM incorporates multi-head mechanisms and stabilization techniques, resulting in improved performance across a wide range of applications [30]. The architectures of sLSTM and mLSTM are depicted in Fig. 2. For further details, readers are encouraged to refer to the original paper [29].

4) Enhanced xLSTM: The original xLSTM, depicted in grey in Fig. 2, leverages historical information and ensures low latency in real-world applications. Convinced that incorporating future information has the potential to significantly boost performance, we introduce a memory block to both sLSTM and mLSTM, as highlighted in yellow in Fig. 2. This integration of both historical and future information significantly enhances the models' capabilities, as demonstrated in numerous sequential tasks [3], [31]. Consequently, we call the proposed structure Enhanced xLSTM, or ExLSTM for short.

The design of the memory block is inspired by DFSMN, a framework renowned for its low computational complexity and high efficacy in the field of speech recognition [31]. The computational process within this memory block is formulated as follows:

$$\boldsymbol{p}_t = \boldsymbol{V}\boldsymbol{x}_t + \boldsymbol{b}_v \tag{20}$$
$$\tilde{\boldsymbol{p}}_t = \boldsymbol{p}_t + \sum_{i=0}^{N_1} \boldsymbol{a}_i \odot \boldsymbol{p}_{t-i} + \sum_{j=1}^{N_2} \boldsymbol{c}_j \odot \boldsymbol{p}_{t+j} \tag{21}$$
$$\boldsymbol{m}_t = \boldsymbol{U}\tilde{\boldsymbol{p}}_t + \boldsymbol{b}_u. \tag{22}$$

Here, Eq. (20) represents a linear projection that transforms the input $\boldsymbol{x}_t$ into a higher-dimensional space, denoted as z. Subsequently, the encoding of both historical and future information is carried out using Eq. (21), where the symbol $\odot$ means element-wise multiplication. Specifically, N_1 represents the look-back order, referring to the number of historical items considered from the past, and N_2 is the look-ahead order, representing the size of the look-ahead window into the future. Finally, the encoded result $\tilde{\boldsymbol{p}}_t$ is projected back to the original space using Eq. (22).

For clarity, a comparison between the causal convolution utilized in xLSTM and the memory blocks integrated into ExLSTM is presented in Fig. 3. Notably, the overall latency τ of ExLSTM is dependent on the look-ahead order N_2 within each memory block, which can be computed using Eq. (23). Here, L denotes the total number of memory blocks, and N_2^l represents the look-ahead order of the l-th memory block. In real-time applications, the look-ahead order can be flexibly adjusted to meet specific latency requirements. As shown in Appendix A, our experimental setup featured $N_2 = 4$ and

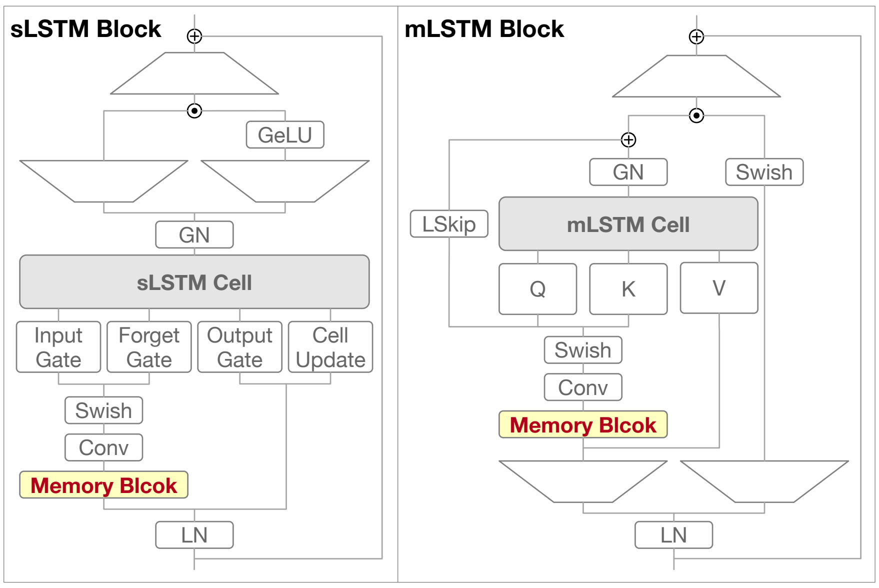

Fig. 2. The blocks of sLSTM and mLSTM, as shown in the left panel and right panel, respectively. The original design is colored in grey. The integration of memory blocks that are colored in yellow results in the proposed Enhanced xLSTM.

$N_2 = 2$ for mLSTM and sLSTM respectively, resulting in a latency of 6 frames.

$$\tau = \sum_{l=1}^{L} N_2^l \tag{23}$$

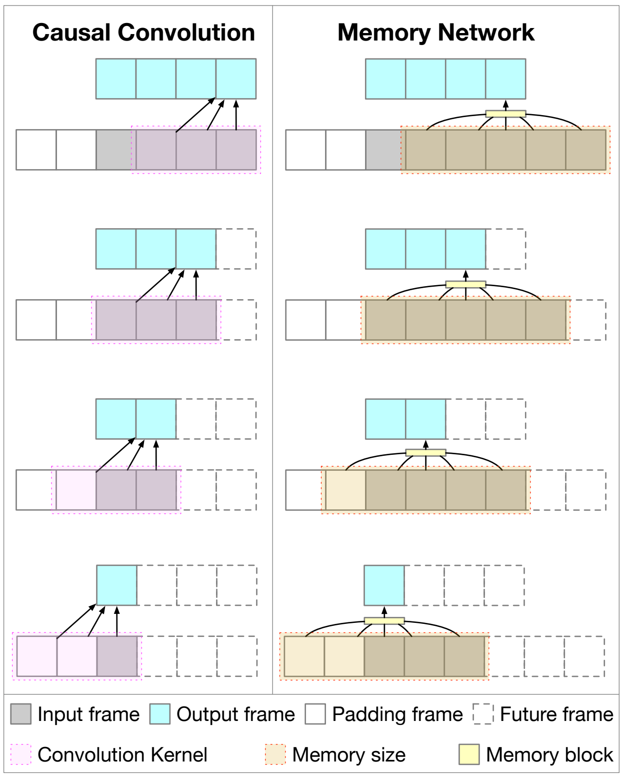

Fig. 3. Left panel: the causal convolution with a kernel size of three frames. Right panel: the memory block with $N_1 = N_2 = 2$.

C. Few-Shot Transfer Learning

Upon completion of training, the parameters of the embedding model are frozen to extract embedding representations that serve as the foundation for customizing a KWS system using few-shot transfer learning. To this end, we append a trainable dense layer (”Dense_t” in the right panel of Fig. 1), consisting of 128 units with a Tanh activation function, to the head of the embedding model to compile a KWS model. The output layer utilizes a softmax activation function and is optimized using the cross-entropy loss function, which is widely employed for multi-category audio classification tasks.

To facilitate the training of KWS models through few-shot transfer learning, we structure three categories of input data: samples of the target keyword, samples of non-target keywords, and background noise. Consequently, this setup translates into a 3-category audio classification task. The preparation and utilization of the datasets are critical for this stage, and further details are provided in the experimental section IV-C.

IV. Experiments

Our experiment involves two types of datasets. The first type comprises open-source keyword datasets used to train embedding models, including the public Google Speech Commands (GSC) dataset and a collected Mandarin keyword dataset for investigating English and Mandarin keywords, respectively. The second type is a compact self-recorded Mandarin dataset designed specifically for few-shot transfer learning.

All audio samples within the datasets have been converted to WAV format, ensuring a standardized sampling rate of 16,000 Hz and a mono-channel configuration. We conducted all experiments using TensorFlow 2.9, except for the xLSTM and ExLSTM backbones, which utilized PyTorch 2.3. The training efficiency was significantly enhanced by an NVIDIA GeForce RTX 3060 GPU.

The structures of the backbones are detailed in Appendix A. In the Appendix section, the parameters not listed were configured to the default values as specified in the code references for Tensorflow 2.9 or PyTorch 2.3. The “-” denotes that the parameters were set to be the same as those for the GSC dataset. The code references are partly borrowed from GitHub: kws_streaming (https://github.com/google-research/google-research/tree/master/kws_streaming), DFSMN (https://github.com/yangxueruivs/DFSMN), MatchboxNet (https://github.com/dominickrei/MatchboxNet), NX-AI (https://github.com/NX-AI/xlstm).

A. Datasets

1) Google Speech Commands Dataset: The English speech commands dataset, provided by Google and widely adopted in the KWS field, was utilized in our experiments in its V2 version [32]. This dataset consists of 105,829 audio recordings, each containing one of 35 keywords and lasting one second. To ensure consistency and comparability with prior work, we followed the same data split ratio of 8:1:1 as outlined in [33], resulting in training, validation, and testing sets comprising 84,843, 9,981, and 11,005 samples, respectively.

2) Mandarin Keyword Dataset: We extensively reviewed several open-source Mandarin keyword datasets, including the Multilingual Spoken Words Corpus [34], aidatatang_200zh (http://www.openslr.org/62/), AISHELL-3 [35], and SHALCAS22A (http://www.openslr.org/138/). To enhance the model’s generalization capability while considering the characteristics of Mandarin keywords, we selected the keywords consisting of 2 to 4 Chinese characters, ensuring that each keyword has at least 10 samples.

Furthermore, we selected samples with durations ranging from 0.9 to 2.5 seconds and volume levels between -40 dB and -10 dB to facilitate effective data augmentation. This approach allowed us to compile a comprehensive Mandarin dataset of 191 keywords, with a total of 7,690 samples. The distribution of these samples across individual keywords is depicted in Fig. 4.

Another important consideration was the duration constraint imposed on the Mandarin keyword samples, which was set to 2.5 seconds. Samples shorter than 2.5 seconds were padded with zeros to meet this duration requirement. Following this pre-processing step, the dataset was then randomly split into three sets with a ratio of 7:2:1 for training, validation, and testing, respectively.

3) Background Noise Dataset: We employed the TUT Acoustic Scenes 2017 dataset (TAS) [36] as a source of

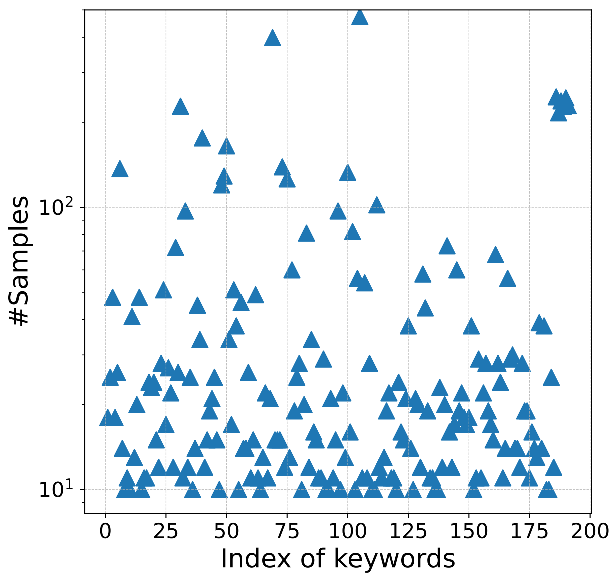

Fig. 4. The distribution of samples in the Mandarin keyword dataset.

background noise, leveraging its diversity of various acoustic environments. Specifically, we selected noise samples with volume levels ranging from -50 dB to -20 dB, ensuring a comprehensive representation of environmental disturbances. This selection process resulted in a collection of 4,266 background noise samples, each with a duration of 10 seconds.

In the classification task for English keywords, the TAS samples were segmented into 1-second clips, and 75% of these clips were randomly assigned to training, validation, and testing sets in an 8:1:1 ratio. These TAS clips were combined with the GSC dataset to create a 36-category classification task (35 English keywords and 1 background noise category) on the embedding models. The remaining 25% of TAS clips were reserved for data augmentation purposes. This splitting strategy is illustrated in the top panel of Fig. 5.

Regarding another task, the TAS samples were segmented into 2.5-second clips, and 70% of these clips were randomly divided for classification and data augmentation in a 3:1 ratio. Subsequently, the clips designated for classification were split randomly into training, validation, and testing sets with a ratio of 7:2:1, as illustrated in the middle panel of Fig. 5. These clips were then combined with the Mandarin keyword dataset to form a 192-category classification task (191 Mandarin keywords and 1 background noise category) on the embedding models. Meanwhile, the remaining 30% of TAS clips were reserved for transfer learning, following a similar splitting strategy as for the embedding models.

B. Training Embedding Model

For each classification task, the training dataset served as the foundation for training the embedding models over 25,000 batch steps, using a linearly decaying learning rate and the Adam optimizer. The initial learning rate was set to 0.01, and the batch size was fixed at 128. To enhance the robustness of the models, data augmentation techniques, including SpecAugment [37] and Cutout [38], were strategically employed.

Throughout the training process, we evaluated the performance of the embedding models by utilizing the validation dataset at every 200 steps. Upon completing training, the model that demonstrated the highest validation accuracy was selected for further evaluation on the testing dataset. The test accuracy scores achieved by different acoustic features and various backbones are compared in Fig. 6. More details are recorded in Tab. I. In the table, the number of model parameters is recorded in thousands. “Fusion” means that we used the proposed scale blocks, i.e., fusing the Mel-spectrogram and MFCC features.

Upon comparing the acoustic features, it can be found that the utilization of scale blocks contributes to improving accuracy while maintaining a relatively small model size in most cases. Specifically, for the GSC dataset, the “Inception-ResNet” and “xLSTM” backbones performed better when fed with MFCC features. Similarly, this trend was also observed with the “CNN” and “Xception” backbones when working with the Mandarin keyword dataset. However, it is noteworthy that the Mel-spectrogram feature consistently led to the largest volumes of model size.

Regarding the backbones, the original “xLSTM” has achieved high accuracy scores in some cases, with a model size of fewer than 25,000 parameters when using MFCC or fused features. Our proposed “ExLSTM” enhanced the performance further and showed the best results in all cases. An interesting observation is that, despite the increased duration of samples in the Mandarin keyword dataset, the number of “ExLSTM” parameters did not increase significantly. This advantage can be largely attributed to its Maxpool layer, as detailed in Appendix A. More potentialities of ExLSTM are to be discovered in the future.

C. Transfer Learning for Customized Keywords

To establish an evaluation dataset for keyword customization, we recruited 15 participants (9 males and 6 females) aged between 18 and 32 years to record 10 famous Mandarin wake-up words. These words served as the keywords for our study, each consisting of 4 Chinese characters in pinyin format, as depicted in Tab. II. To ensure the authenticity of the recordings, we developed a dedicated Android application, allowing us to gather the data under realistic conditions. Each participant was tasked with reading all keywords and repeating each keyword at least 20 times for sample diversity.

For the set of 10 wake-up words, we successively designated one as the target keyword and carried out 10 trials for each. In each trial, we randomly selected K target samples and applied the same augmentation strategies outlined in section IV-B to generate 2,000 augmented target samples. Here, K varied from 1 to 5 for comparative purposes. Simultaneously, we randomly chose 2,000 non-target samples from the remaining 9 keywords and 2,000 background noise clips from the specified TAS described in section IV-A3. These samples were combined with the 2,000 augmented target samples to construct the training dataset. The remaining samples were randomly divided into validation and testing sets using a 7:3 ratio. The preparation of the self-recorded dataset for transfer learning is illustrated in the bottom panel of Fig. 5.

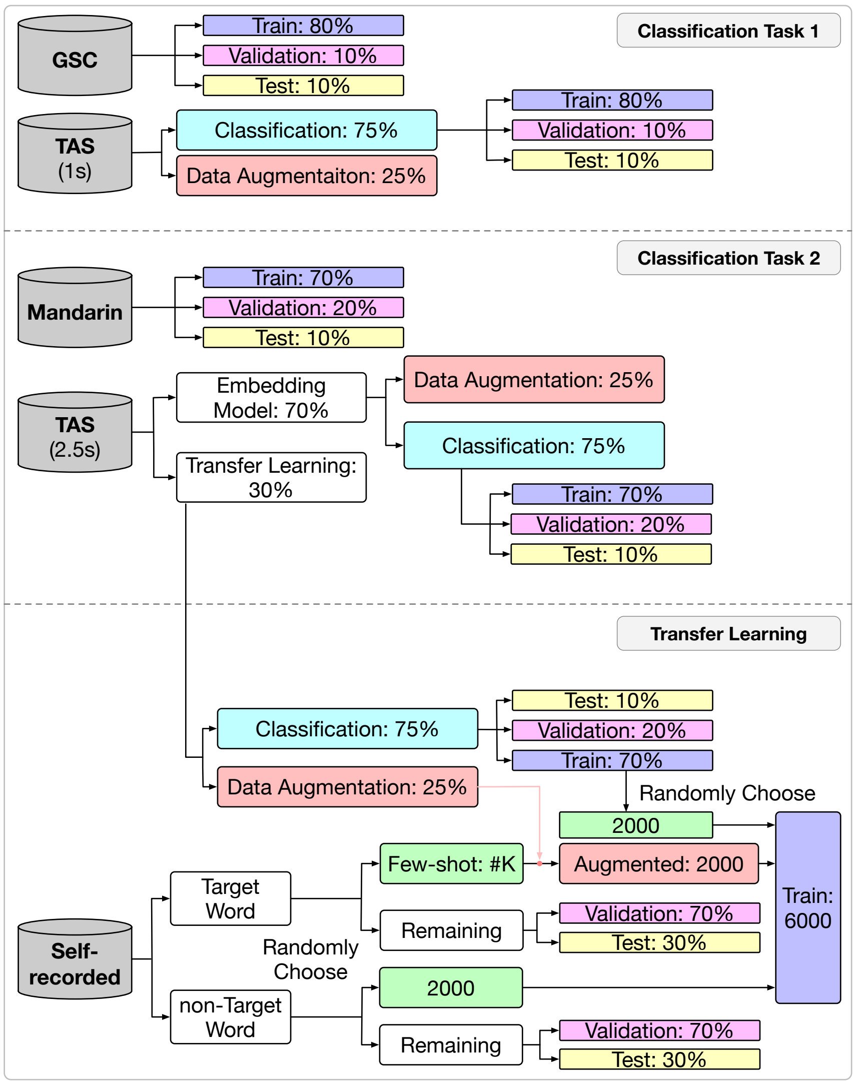

Fig. 5. The dataset splitting strategies in each experimental step.

TABLE I
RESULTS OF THE EMBEDDING MODELS FOR TWO DATASETS, BASED ON DIFFERENT ACOUSTIC FEATURES AND BACKBONES.

Backbones	Google Speech Commands						Mandarin Keyword Dataset					
	Mel-spectrogram		MFCC		Fusion		Mel-spectrogram		MFCC		Fusion	
	Acc.	#Param.	Acc.	#Param.	Acc.	#Param.	Acc.	#Param.	Acc.	#Param.	Acc.	#Param.
CNN [39]	15.72%	205.3	72.68%	45.5	78.16%	47.1	71.44%	823.2	76.76%	165.8	76.56%	167.3
Xception [40]	92.49%	67.1	89.54%	51.7	92.26%	52.8	88.72%	194.2	89.60%	163.5	89.21%	164.6
Inception-ResNet [41]	93.75%	65.6	93.21%	50.2	92.23%	51.3	63.48%	168.3	81.49%	152.9	91.26%	154.0
GRU [39]	75.43%	82.0	80.13%	53.2	92.46%	54.3	72.27%	198.0	77.05%	161.1	86.57%	162.2
DFSMN [31]	86.08%	73.1	85.73%	48.5	93.44%	51.6	86.23%	213.8	84.67%	164.7	88.13%	167.8
MatchboxNet [17]	81.69%	88.7	81.33%	54.9	84.31%	56.0	71.92%	292.5	79.79%	157.3	86.52%	158.4
xLSTM [29]	91.74%	316.3	93.31%	23.8	92.23%	24.9	88.64%	231.5	91.28%	21.2	91.96%	22.2
ExLSTM (proposed)	**94.2**%	416.2	**94.54**%	51.4	**95.31**%	52.4	**88.87**%	436.3	**92.36**%	56.5	**94.48**%	57.6

In this step, we leveraged the embedding models equipped with scale blocks. Different backbones were compared to extract embedding representations. As mentioned in section III-C, a dense layer was appended to the head of the embedding model to construct the KWS model. The hyper-parameters for training the KWS models were configured identically to those employed for the embedding models, except for reducing the training steps to 1,000 and setting an initial learning rate of 0.005 to stabilize the training process.

As depicted in Fig. 7, we averaged the test accuracy scores of all customized KWS models and conducted a comparative analysis across varying K values. It can be found that the performance of all backbones presented a roughly upward trend as K increased. Specifically, the ExLSTM backbone achieved the best performance with an overall average accuracy of 97.45% when $K = 5$. For this specific scenario, the detailed average accuracy scores corresponding to each target keyword are presented in Tab. II.

V. CONCLUSIONS

KWS applications require high performance and low computational complexity. We introduce a two-stage method for

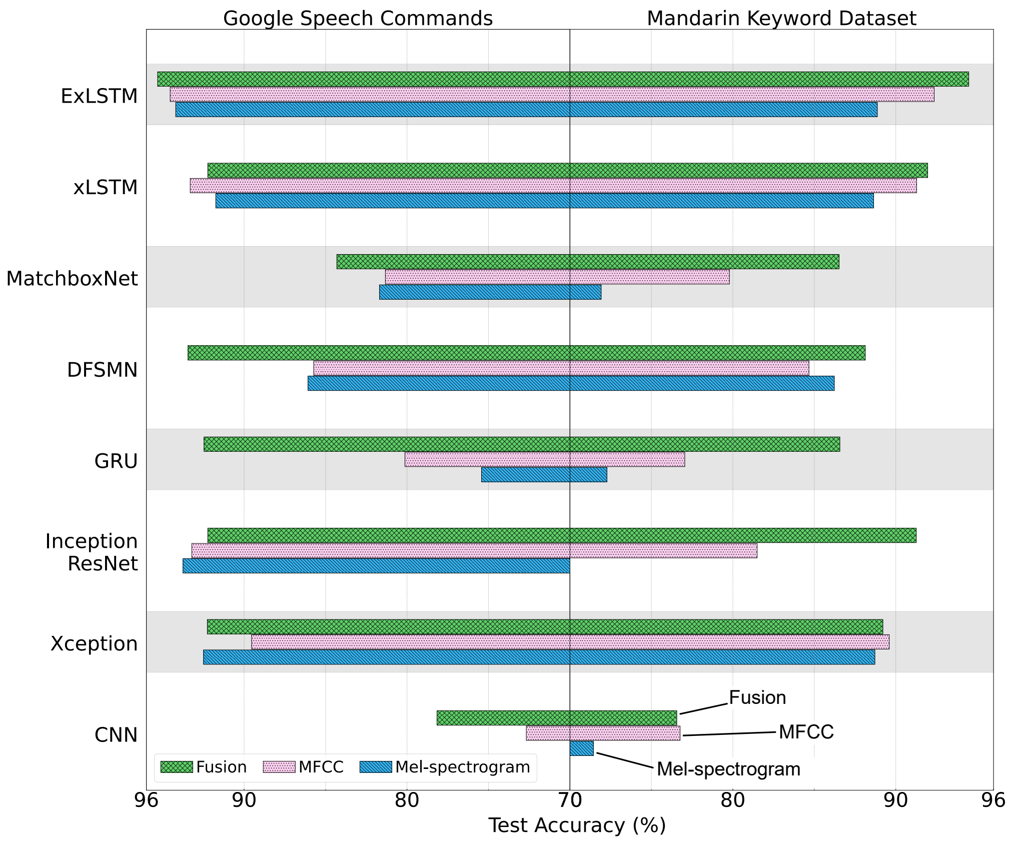

Fig. 6. The test accuracy scores and model sizes of the embedding models with various backbones for two datasets.

TABLE II
THE MANDARIN WAKE-UP WORDS DEFINED BY CHINESE COMPANIES. THE THIRD COLUMN DISPLAYS THE TEST ACCURACY SCORE OF THE KWS MODEL AVERAGED OVER 10 TRIALS. EACH KWS MODEL WAS CUSTOMIZED BY THE CORRESPONDING WAKE-UP WORD, WITH EXLSTM AS THE BACKBONE, UNDER THE SETTING OF $K = 5$.

Wake-up Word	Company	Acc.	#Samples
ding1 dong1 ding1 dong1	JD	98.57%	346
ni3 hao3 wen4 wen4	Mobvoi	99.84%	377
tian1 mao1 jing1 ling2	Alibaba	97.03%	355
xiao3 ai4 tong2 xue2	Xiao mi	98.22%	342
xiao3 bu4 xiao3 bu4	OPPO	96.62%	351
xiao3 di2 xiao3 di2	BYD	96.08%	346
xiao3 du4 xiao3 du4	Baidu	96.29%	341
xiao3 mei3 xiao3 mei3	Midea	98.21%	335
xiao3 ya3 xiao3 ya3	Ximalaya	97.16%	371
xiao3 yi4 xiao3 yi4	Huawei	96.46%	333
		Avg. 97.45%	**Total 3497**

rapid customization of KWS systems. In the first stage, we demonstrate the efficacy of scale blocks and lightweight ExLSTM backbone in learning embedding representations from large-vocabulary datasets. In the second stage, we leverage few-shot transfer learning to effectively transfer these embedding representations to target keywords. The encouraging outcomes from our self-recorded Mandarin dataset indicate the practical significance of our approach in this field. Moving forward, our future work will concentrate on developing keyword customization applications for micro-controllers like STM32.

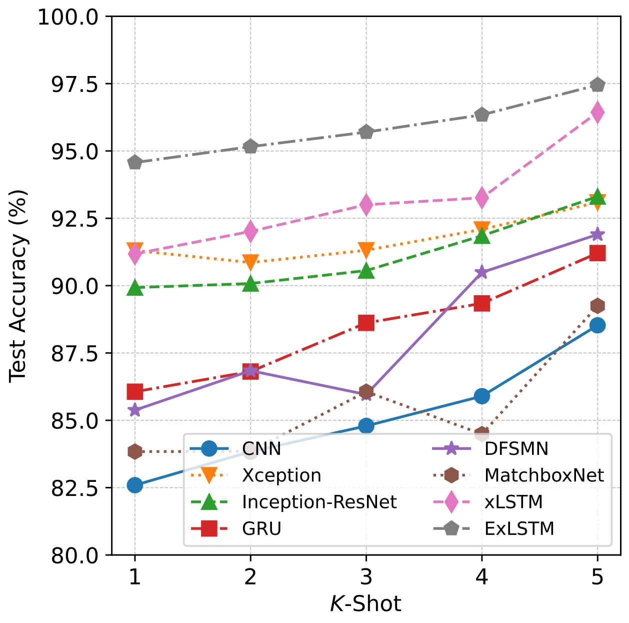

Fig. 7. Average test accuracy scores of the customized KWS models with different K values.

REFERENCES

[1] Z. Liu, T. Li, and P. Zhang, "Rnn-t based open-vocabulary keyword spotting in mandarin with multi-level detection," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 5649–5653.

[2] S. Li and H. Zhang, "Keyword spotting based on ctc and similarity matching for chinese speech," in *2023 IEEE/ACIS 23rd International Conference on Computer and Information Science (ICIS)*, 2023, pp. 79–84.

[3] H. Qin, X. Ma, Y. Ding, X. Li, Y. Zhang, Z. Ma, J. Wang, J. Luo, and X. Liu, "Bifsmnv2: Pushing binary neural networks for keyword spotting to real-network performance," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 8, pp. 10 674–10 686, 2024.

[4] L. Lei, G. Yuan, H. Yu, D. Kong, and Y. He, "Multilingual customized keyword spotting using similar-pair contrastive learning," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 31, pp. 2437–2447, 2023.

[5] J. Jung, Y. Kim, J. Park, Y. Lim, B.-Y. Kim, Y. Jang, and J. S. Chung, "Metric learning for user-defined keyword spotting," in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023, pp. 1–5.

[6] Y. M. Saidutta, R. S. Srinivasa, C.-H. Lee, C. Yang, Y. Shen, and H. Jin, "To wake-up or not to wake-up: Reducing keyword false alarm by successive refinement," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.

[7] A. Berg, M. O'Connor, and M. T. Cruz, "Keyword transformer: A self-attention model for keyword spotting," *arXiv preprint arXiv:2104.00769*, 2021.

[8] M. Li, "A lightweight architecture for query-by-example keyword spotting on low-power iot devices," *IEEE Transactions on Consumer Electronics*, vol. 69, no. 1, pp. 65–75, 2023.

[9] H.-K. Shin, H. Han, D. Kim, S.-W. Chung, and H.-G. Kang, "Learning audio-text agreement for open-vocabulary keyword spotting," *arXiv preprint arXiv:2206.15400*, 2022.

[10] R. Shan, X. Zhang, and S. Li, "A method of pneumonia detection based on an improved yolov5s," *Engineering Letters*, vol. 32, no. 6, pp. 1243–1254, 2024. [Online]. Available: https://www.engineeringletters.com/issues_v32/issue_6/EL_32_6_17.pdf

[11] X. Wen, Y. Yao, Y. Cai, Z. Zhao, T. Chen, Z. Zeng, Z. Tang, and F. Gao, "A lightweight st-yolo based model for detection of tea bud in unstructured natural environments," *IAENG International Journal of Applied Mathematics*, vol. 54, no. 3, pp. 342–349, 2024.

[12] Q. Shao, J. Hou, Y. Hu, Q. Wang, L. Xie, and X. Lei, "Target speaker extraction for customizable query-by-example keyword spotting," in *2021 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 2021, pp. 672–678.

[13] Z. Yang, S. Sun, J. Li, X. Zhang, X. Wang, L. Ma, and L. Xie, "Catt-kws: a multi-stage customized keyword spotting framework based on cascaded transducer-transformer," *arXiv preprint arXiv:2207.01267*, 2022.

[14] S. Mallat, "Group invariant scattering," *Communications on Pure and Applied Mathematics*, vol. 65, no. 10, pp. 1331–1398, 2012. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpa.21413

[15] J. Andén and S. Mallat, "Deep scattering spectrum," *IEEE Transactions on Signal Processing*, vol. 62, no. 16, pp. 4114–4128, 2014.

[16] H. Cao, H. Chen, and J. Yuan, "Infant cry detection with lightweight wavelet scattering networks," *IEEE Access*, vol. 11, pp. 135 905–135 914, 2023.

[17] S. Majumdar and B. Ginsburg, "Matchboxnet: 1d time-channel separable convolutional neural network architecture for speech commands recognition," *arXiv preprint arXiv:2004.08531*, 2020.

[18] H. Zhou, W. Hu, Y. T. Yeung, and X. Chen, "Energy-friendly keyword spotting system using add-based convolution," in *Interspeech 2021*, 2021, pp. 4234–4238.

[19] K. Ding, M. Zong, J. Li, and B. Li, "Letr: A lightweight and efficient transformer for keyword spotting," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 7987–7991.

[20] Z. Q. Lin, A. G. Chung, and A. Wong, "Edgespeechnets: Highly efficient deep neural networks for speech recognition on the edge," *arXiv preprint arXiv:1810.08559*, 2018.

[21] T. Higuchi, M. Ghasemzadeh, K. You, and C. Dhir, "Stacked 1d convolutional networks for end-to-end small footprint voice trigger detection," *arXiv preprint arXiv:2008.03405*, 2020.

[22] B. Kim, S. Chang, J. Lee, and D. Sung, "Broadcasted residual learning for efficient keyword spotting," *arXiv preprint arXiv:2106.04140*, 2021.

[23] 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.

[24] M. Xu and X.-L. Zhang, "Depthwise separable convolutional resnet with squeeze-and-excitation blocks for small-footprint keyword spotting," *arXiv preprint arXiv:2004.12200*, 2020.

[25] 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.

[26] Y. Wang, C. Zhu, Q. Wang, and J. Fang, "Research on fault detection of rolling bearing based on cwtdccnn-lstm," *Engineering Letters*, vol. 31, no. 3, pp. 987–1000, 2023.

[27] H. Sen, "Time series prediction based on improved deep learning," *IAENG International Journal of Computer Science*, vol. 49, no. 4, pp. 1133–1138, 2022.

[28] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222–2232, 2017.

[29] M. Beck, K. Pöppel, M. Spanring, A. Auer, O. Prudnikova, M. Kopp, G. Klambauer, J. Brandstetter, and S. Hochreiter, "xlstm: Extended long short-term memory," *arXiv preprint arXiv:2405.04517*, 2024.

[30] B. Alkin, M. Beck, K. Pöppel, S. Hochreiter, and J. Brandstetter, "Vision-lstm: xlstm as generic vision backbone," *arXiv preprint arXiv:2406.04303*, 2024.

[31] S. Zhang, M. Lei, Z. Yan, and L. Dai, "Deep-fsmn for large vocabulary continuous speech recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 5869–5873.

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

[33] O. Rybakov, N. Kononenko, N. Subrahmanya, M. Visontai, and S. Laurenzo, "Streaming keyword spotting on mobile devices," *arXiv preprint arXiv:2005.06720*, 2020.

[34] M. Mazumder, S. Chitlangia, C. Banbury, Y. Kang, J. M. Ciro, K. Achorn, D. Galvez, M. Sabini, P. Mattson, D. Kanter, G. Diamos, P. Warden, J. Meyer, and V. J. Reddi, "Multilingual spoken words corpus," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. [Online]. Available: https://openreview.net/forum?id=c20jiJ5K2H

[35] Y. Shi, H. Bu, X. Xu, S. Zhang, and M. Li, "Aishell-3: A multi-speaker mandarin tts corpus and the baselines," *arXiv preprint arXiv:2010.11567*, 2020.

[36] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, "DCASE 2017 challenge setup: Tasks, datasets and baseline system," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, November 2017, pp. 85–92.

[37] 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," in *Interspeech 2019*, 2019, pp. 2613–2617.

[38] T. Devries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *arXiv preprint arXiv:1708.04552*, 2017.

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

[40] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1800–1807.

[41] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2818–2826.

APPENDIX A
THE STRUCTURES OF THE BACKBONES FOR TWO DATASETS.

Backbones	**Parameters**	**Google Speech Commands**	**Mandarin Keyword Dataset**
CNN [39] (Code Ref.: kws_streaming)	Conv filters	$[8, 8, 8, 8, 16, 8, 16]$	$[8] \times 7$
	Conv activation functions	$[\mathrm{ReLU}] \times 7$	-
	Flatten drop rate	0.5	-
	Dense units	$[8, 16]$	-
	Dense activation functions	$[\mathrm{Linear}, \mathrm{ReLU}]$	-
Xception [40] (Code Ref.: kws_streaming)	Conv filters	$[32, 32, 64, 128]$	$[64, 64, 128, 256]$
	Conv kernels	$[(5, 5)] \times 4$	-
	Conv activation functions	$[\mathrm{ReLU}] \times 4$	-
	MaxPooling strides	$[(2, 1)] \times 4$	-
	Dense units	$[128]$	-
	Dense activation functions	$[\mathrm{ReLU}]$	-
Inception-ResNet [41] (Code Ref.: kws_streaming)	Conv filters 1	$[32]$	-
	Conv kernels 1	$[(5, 1)]$	-
	Conv activation functions 1	$[\mathrm{ReLU}]$	-
	Conv filters 2 - branch 0	$[32, 32, 32]$	-
	Conv filters 2 - branch 1	$[32, 32, 32]$	$[64, 64, 64]$
	Conv filters 2 - branch 2	$[32, 32, 32]$	$[64, 64, 128]$
	Conv kernels 2	$[(3, 1), (5, 1), (5, 1)]$	-
	Conv activation functions 2	$[\mathrm{ReLU}] \times 3$	-
	Conv residual scale 2	$[0.2, 0.5, 1.0]$	-
	MaxPooling strides 2	$[(2, 1), (2, 1), (1, 1)]$	-
GRU [39] (Code Ref.: kws_streaming)	GRU units	$[100]$	$[128]$
	Flatten drop rate	0.1	-
	Dense units	$[64, 64]$	$[128, 256]$
	Dense activation functions	$[\mathrm{Linear}, \mathrm{ReLU}]$	-
DFSMN [31] (Code Ref.: DFSMN)	Conv filters	$[32, 32, 64]$	$[32, 64, 128]$
	Conv kernels	$[(3, 3)] \times 3$	-
	Conv activation functions	$[\mathrm{ReLU}] \times 3$	-
	Conv drop rate	0.5	-
	AveragePooling size	$[(1, 2)] \times 3$	-
	DFSMN input size	$[32, 32]$	$[32, 32, 32]$
	DFSMN hidden size	$[64, 64]$	$[128, 128, 128]$
	DFSMN output size	$[32, 32]$	$[32, 32, 32]$
	DFSMN left memory	$[15, 15]$	$[15, 15, 15]$
	DFSMN right memory	$[5, 5]$	$[5, 5, 5]$
	DFSMN strides	$[1, 1]$	$[1, 1, 1]$
	DFSMN activation functions	$[\mathrm{ReLU}] \times 2$	$[\mathrm{ReLU}] \times 3$
	DFSMN drop rate	0.5	-
MatchboxNet [17] (Code Ref.: MatchboxNet)	Prologue conv filters	32	128
	Prologue conv kernels	11	-
	Prologue conv strides	2	-
	Prologue conv activation functions	ReLU	-
	Intermediate block channels (C)	32	-
	Intermediate block kernels	$[13, 15, 17, 19, 21, 23]$	-
	Repeat sub-blocks (R)	2	-
	Epilogue conv filters	$[16, 16]$	$[64, 64]$
	Epilogue conv kernels	$[29, 1]$	-
	Epilogue conv dilations	$[2, 1]$	-
	Epilogue conv activation functions	$[\mathrm{ReLU}] \times 2$	-
xLSTM [29] (Code Ref.: NX-AI)	Structure	mLSTM→sLSTM→AvgPool	mLSTM→mLSTM→MaxPool
	mLSTM: Conv1D kernel	11	7
	sLSTM: Conv1D kernel	11	✗
	Number of heads	1	-
ExLSTM (proposed) (Code Ref.: DFSMN,NX-AI)	Structure	mLSTM→sLSTM→MaxPool	-
	mLSTM: Conv1D kernel	4	-
	mLSTM: Look-back order N_1	3	-
	mLSTM: Look-ahead order N_2	4	-
	sLSTM: Conv1D kernel	7	-
	sLSTM: Look-back order N_1	10	-
	sLSTM: Look-ahead order N_2	2	-
	sLSTM: Up-projection activation	ReLU	-
	Number of heads	1	-
	Dimension of memory block: z	128	-