IEICE TRANS. INF. & SYST., VOL.E106–D, NO.7 JULY 2023

PAPER

A Lightweight End-to-End Speech Recognition System on Embedded Devices

Yu WANG[†,††], ***Nonmember and*** **Hiromitsu NISHIZAKI**[††a)], ***Senior Member***

SUMMARY In industry, automatic speech recognition has come to be a competitive feature for embedded products with poor hardware resources. In this work, we propose a tiny end-to-end speech recognition model that is lightweight and easily deployable on edge platforms. First, instead of sophisticated network structures, such as recurrent neural networks, transformers, etc., the model we propose mainly uses convolutional neural networks as its backbone. This ensures that our model is supported by most software development kits for embedded devices. Second, we adopt the basic unit of MobileNet-v3, which performs well in computer vision tasks, and integrate the features of the hidden layer at different scales, thus compressing the number of parameters of the model to less than 1 M and achieving an accuracy greater than that of some traditional models. Third, in order to further reduce the CPU computation, we directly extract acoustic representations from 1-dimensional speech waveforms and use a self-supervised learning approach to encourage the convergence of the model. Finally, to solve some problems where hardware resources are relatively weak, we use a prefix beam search decoder to dynamically extend the search path with an optimized pruning strategy and an additional initialism language model to capture the probability of between-words in advance and thus avoid premature pruning of correct words. In our experiments, according to a number of evaluation categories, our end-to-end model outperformed several tiny speech recognition models used for embedded devices in related work.

key words: *automatic speech recognition, embedded and edge devices, end-to-end, prefix beam search, self-supervised learning*

1. Introduction

Automatic speech recognition (ASR) systems are widely used in various embedded equipment, such as smart home devices or in-vehicle infotainment (IVI) systems. ASR systems benefit from some cutting-edge technologies, including deep neural networks (DNN) [1], weighted finite state transducers (WFST) [2], and self-supervised learning [3]–[5].

When running ASR systems, large amounts of memory and computing power are normally required. Current ASR systems require thousands of hours of transcribed speech to achieve an acceptable level of performance [6]–[8]. Using Kaldi [9], which is one of the most popular open-source speech recognition toolkits, and the Mini-Librispeech dataset, which has only 5 hours of speech transcription, the memory occupied in the hard disk for building a WFST decoding graph is about 530 MB. On the other hand, most ASR systems for industrial products require low-latency real-time recognition of speech. That is, the corresponding time of the ASR algorithm needs to be completed within one second [10]. Therefore, an ASR system is typically deployed on a cloud server and communicates with edge devices through a wireless network. However, performing speech recognition on embedded devices locally is also an important requirement when the network is unavailable or when the user's personal data is not allowed to be uploaded.

When installing an ASR system on an embedded device, less computation and lower memory occupation are expected. Compared with the traditional modeling approach of the ASR system (which makes a decoding graph [2] from a hidden Markov model (HMM)-based acoustic model (AM), a pronunciation lexicon transducer and a statistical language model (LM)), end-to-end (E2E) model [11]–[13] (which only has a single end-to-end trained neural network model) is favored because of its smaller model size and state-of-the-art accuracy. Regarding the E2E modeling framework, there are a range of excellent papers on the topic of this framework and how it can be used to address the challenges of implementing speech recognition on embedded devices.

In [14], a combination of connectionist temporal classification (CTC)-based E2E AM and recurrent neural networks (RNN)-based [15] LM and beam-search decoding [16] was used for speech recognition in mobile and embedded devices. In this model, the number of parameters in the final E2E model was about 15 M. StreamE2E [17] also achieved faster computation and better recognition accuracy on mobile devices by optimizing the structure of the RNN transducer (RNN-T) model [18] and using text-to-speech (TTS)-based speech augmentation.

Thus, speech recognition has been achieved on edge devices, which are devices that have many available resources. However, two major problems must be solved in order to drive speech recognition on many general-purpose edge devices. First, on most embedded devices, model architecture is limited, depending on the software development kit (SDK) provided by the hardware manufacturers. The Android platform is widely used on edge platforms, and there are some excellent open-source frameworks like TensorRT [19] and NCNN [20] to help developers deploy their models. However, there are still many platforms and embedded devices active in industrial products that are not compat-

Manuscript received December 16, 2022.
Manuscript revised March 10, 2023.
Manuscript publicized April 13, 2023.

[†]The author is with the AI Engineering Research Center, Streamax Technology Co., Ltd., 16F, Zhiyuan B1, No.1001 Xueyuan Avenue, Nanshan, Shenzhen, Guangdong, China.
[††]The authors are with the Integrated Graduate School of Medicine, Engineering, and Agricultural Sciences, University of Yamanashi, Kofu-shi, 400–8511 Japan.
a) E-mail: hnishi@yamanashi.ac.jp

DOI: 10.1587/transinf.2022EDP7221

ible with these open-source frameworks. In addition, many edge devices do not utilize advanced neural network structures like LSTM and transformer [21]. Therefore, many of the network structures proposed in previous work may be hard to install. Second, differing from the decoding algorithms using WFST, the beam search family decoders do not build a large static decoding graph, but dynamically expand the search path during decoding. Therefore, they are also mainstream decoding algorithms for E2E models on mobile devices. However, generally speaking, the performance of beam search decoders is a matter of contention, and related research is less concerned with the optimization of beam search algorithms for ASR decoding. The problems discussed above are the focus of this paper.

Therefore, our goal is to propose a smaller, faster ASR system with stronger SDK compatibility and high accuracy. We hope that the number of parameters of the neural network model is within 1 M, the memory size of the whole ASR system is less than 10 M, the single core occupancy on an ARM-32 architecture chip, which is often used on a low-end embedded device, is not more than 20%, and the char error rate (CER) on the open source dataset Librispeech is less than 10%. Furthermore, we hope that the CER on a specific domain can reach less than 5%. This setting can ensure that our ASR system can be used on most mobile products for custom ASR tasks and provide a good user experience.

In this paper, we propose the following improvements to satisfy the above requirements in order to run highly accurate and memory-saving speech recognition on edge devices. First, convolutional neural networks (CNN) [22] have a natural computational advantage on graphics processing units (GPUs) or neural processing units (NPUs) and are compatible with most SDKs. Therefore, regarding the model structure, in this study, we propose to mainly use CNN layers in order for the model to be deployed on edge devices as easily as possible. To compensate for the shortcomings of the CNN layer in capturing information over a long distance, we fuse features of different scales. Moreover, instead of manual speech features such as mel-scale frequency cepstral coefficients (MFCC) or mel-filter banks (fBanks), we adopt a CNN feature extraction module to compute acoustic representations from audio waveforms directly. With this model, all inference processes are assigned to the dedicated GPUs or NPUs. Second, in the training stage of the model, we use a self-supervised learning approach based on wav2vec2.0 [5]. This strategy ensures that our tiny model can be converged and, in the case of the small amount of real data we can collect, plays an important role in improving accuracy. In our experiments, our E2E model performs competitively compared to some popular lightweight models, in terms of not just the model size but also the error rate. Finally, in terms of decoding, we propose a decoder using an improved prefix beam search to handle CTC probability output. We separate acoustic pruning for different prefixes, which prevents the right path from being excluded by mistake. In addition, instead of a word piece LM, which always takes up a large amount of memory, we design a joint method of sub-word LM and initialism LM to capture the context information within and between words, respectively.

Our proposed model had about 0.79 M parameters. Using the greedy decoder, the CER on Librispeech [6] *test-clean* was 13.57%. Although this accuracy rate is still far from our goal, it is better than that of some popular lightweight models, including a ResNet-18 [23] model and MobileNetv3 [24] model. Then the model was transferred to our specific task. Using the improved prefix beam search decoding algorithm, the error rate on our test dataset was 4.86%, slightly higher than the result 3.99% of the widely used WFST decoder, our ASR system only have four thousandths of the memory compared to said WFST decoder. Our improvement approaches yielded good results experimentally.

The contribution of this paper is to show that it is possible to realize an ASR model with high ASR performance that works in real-time and with little memory on edge devices. This ASR model could be realized without using a transformer model, which provides high ASR accuracy, but with a model mainly based on convolutional layers, utilizing pre-training based on wav2vec2.0, learning rate adjustment, and a prefix beam search algorithm with an initialism LM.

The remainder of the paper is organized as follows. In Sect. 2, we describe the E2E model architecture adopted in this work. In addition, the self-supervised training stages are derived, and we discuss how the decoding algorithm is improved and explain how to build an initialism LM. In Sect. 3, we describe the implementation of the proposed model. All experiment details and results are presented in Sect. 4. We conclude our work in Sect. 5.

2. Design of Lightweight End-to-End ASR Model

2.1 Outline of Model Architecture

In recent years, deep learning models have shown amazing potential in vision-related tasks and are widely deployed on embedded devices, such as ResNet [23] series, Yolo [25], [26] series, MobileNet [24] series, and so on. A CNN-based model has been proven to have better compatibility with most mobile devices and has also been introduced into ASR tasks [22], [27] because of its a small mount of parameters and fast inference. Figure 1 shows an example of the inference time and CPU occupation of an E2E ASR system on an embedded device with Android as its OS and a CPU of ARMv7 with four cores. The acoustic feature is 296×128-dimensional fBanks and the model is built using ResNet-18 backbone. When running with CPU only, the ASR algorithm takes up a non-negligible amount of computational resources, not only during model forwarding but also during the extraction of acoustic representations. As a result, we design our model architecture according to the following criteria:

- mainly using CNN layers in order to reduce parameters and ensure that the model can be deployed to most

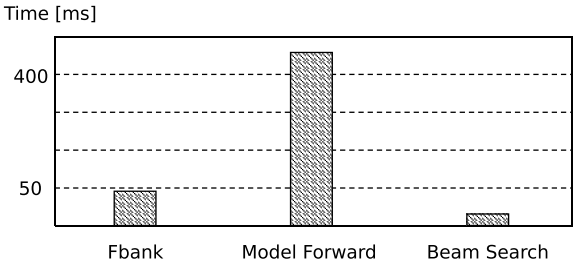

Fig. 1 The power consumption of an E2E ASR system on an embedded device. The device with the Android OS and ARMv7 CPU with four cores was developed by Streamax Technology Co., Ltd. The extraction of fBank feature takes about 40 ms. The model forwarding takes about 600 ms, and the beam search takes about 2 ms. The CPU occupation of the ASR algorithm is about 25% on single core.

embedded devices,

- being able to transfer the entire model to the GPU or NPU in order to speed up the inference process,

as well as reducing the number of parameters of the model.

Figure 2 illustrates the basic modules of our E2E model. The model consists of three components: the convolutional feature extractor "*Feature Extractor*," the feature reconstruction module "*Encoder*," and the projection head "*CTC Projector*." *Feature Extractor* module extracts raw acoustic representations from 1-dimensional audio waveforms. In *Encoder*, with reference to feature pyramid network (FPN) [28] and U-Net [29], we use multiple CNN layers to embed the raw acoustic representations to different scales and skip connections to fuse these features in order to capture local and long-distance linguistic relations in a context of a certain length. As shown in Fig. 2, there are three acoustic features generated in our model: *Raw Feature*, *Bottleneck Feature* and *Rebuild Feature*. These features can be used for different tasks. For the ASR task, we select *Rebuild Feature*, which contains more sophisticated semantic information, and finally input it to the *CTC projector*. In our experiment, we trained various *Feature Extractor*s and *Encoder*s, which varied in performance in terms of model size and accuracy. Section 3 details the proposed model structures.

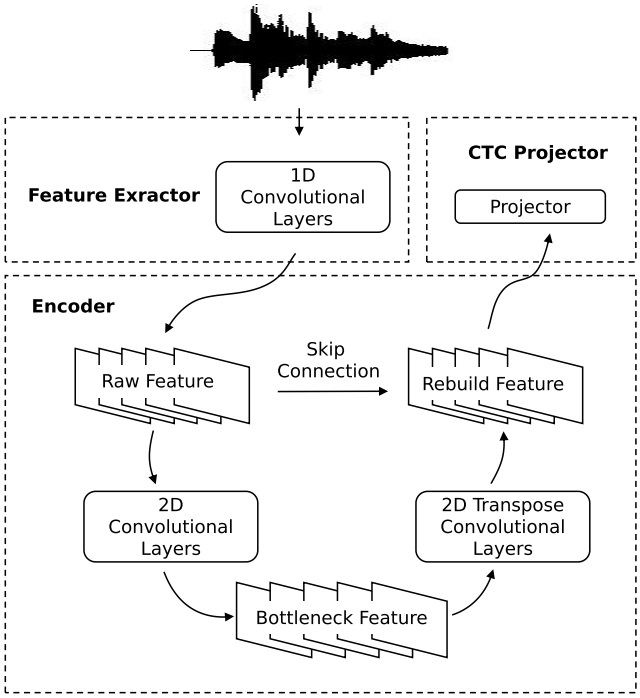

Fig. 2 The basic architecture of our E2E model. *Feature Extractor* and *Encoder* use multiple CNN layers with different kernel sizes. Skip connection connects only the hidden outputs of the same scale.

2.2 Self-Supervised Training

In the training stage of the E2E ASR model, it is necessary to train a feature extractor to extract acoustic representations from waveforms. For this purpose, we adopt the recently proposed concept of wav2vec2.0. The original wav2vec2.0 is an elaborate self-supervised pre-training framework that can perform powerful feature extraction for speech recognition. Therefore, we believe that using this framework will make the training of the feature extractor consisting of convolutional layers that we use in this study more robust. It trains an ASR model through two steps: *pre-training* and *fine-tuning*. In the *pre-training* stage, the fully convolutional networks are used to extract the acoustic representations from raw waveforms, and then the acoustic representations are sent to the multi-layer transformer network to reconstruct the feature map. At the same time, the acoustic representations are processed using masking and product quantization to generate learning targets. Then, by minimizing the contrastive loss and diversity loss between the reconstructed feature map and the targets, the feature extractor can obtain the ability to represent acoustic information, and the encoder can understand this information. Throughout the entire *pre-training* stage, speech transcripts are not required. After *pre-training*, the parameters of the feature extractor are fixed and a projection layer is appended behind the Transformer encoder, and then the parameters of these two modules are fine-tuned by calculating the CTC loss with the speech transcripts. This step requires labeled data.

For the E2E ASR model to work on edge devices, the original wav2vec2.0 is difficult to deploy on embedded devices because it uses Transformer, which is not supported by many SDKs, but we can apply its training approach to other layer structures such as our proposed model. We train our E2E speech recognition model in a similar way to wav2vec2.0 in order to achieve higher accuracy with the small amount of speech data collected on the device, as well as to help training convergence in a model with as small a number of parameters as possible. In the feature extractor, we use fewer layers than the feature extraction module of wav2vec2.0, as well as replace the traditional two-dimensional convolutional layer with the customized Mobilenet-v3 block where one-dimensional convolutional layer is used, thus reducing the number of parameters. As

mentioned in Sect. 2.1, because CNN is mainly selected, we have designed a completely different model structure from the transformer encoder of wav2vec2.0. Through these operations, our model structure has been greatly compressed.

Compared to the ASR system deployed in the cloud, the device-side ASR system is usually used for specific tasks. Therefore, developers pay more attention to maximize the accuracy in a specific domain under the limited model architecture than to its universality. However, the corpus of a specific domain is often very difficult to collect, for example due to commercial rights. With the help of wav2vec2.0, we can achieve high accuracy even with a small amount of domain data through self-supervised training. This is exactly what we need. However, compared to wav2vec2.0, the model structure we propose is smaller, so the upper limit of the accuracy it can achieve is theoretically lower. Therefore, instead of directly using a small amount of domain data to fine-tune the model, we first use labeled public data for fine-tuning, and then use the domain data for transfer learning. In summary, differing from original self-supervised learning strategy of wav2vec2.0, we train the model with an extra step: *transfer-learning*. We first use large-scale unlabeled data to pre-train *Feature Extractor* and *Encoder* and fine-tune the *Encoder*. Then, a *transfer-learning* process is performed using the fine-tuned model with a small amount of target domain speech data.

When conducting the *fine-tuning* and *transfer-learning* steps, we change the learning rate in order to obtain the ASR model with a better character error rate (CER). We use a classic cosine annealing scheduler, but once the parameters of the model have converged, we find the best model parameter that has been saved in advance and restart the training cycle with a motivated initial learning rate. Because the cosine annealing scheduler constantly scales the learning rate during the training process, the initial learning rate can be used again. After the model converges, values of the learning rate that are less than the current initial learning rate are likely to no longer get better results. Therefore, in the next cycle, we allow the learning rate to be magnified to a greater value than the current initial value. The mechanism used in our experiment is shown in Eq. (1). n is the number of cycles, and γ_0 is the initial learning rate of the first training loop.

$$\gamma_n = 2^{n-1} \times \gamma_0 \quad (n > 0) \tag{1}$$

Figure 3 shows the change in learning rate and CER after using this trick during the *fine-tuning* step in our experiment. We stimulated the learning rate to different initial values on two occasions (see the black dots), and each time, we got better results than before. Figure 3 indicates that the learning rate incentive method has a certain positive effect on model training.

In our experiment, we used an early stop strategy, which makes the training loop stop if the CER does not decrease in a certain time. This trick may lead to the early end of training before a new round of learning rate would be increased to the initial value. To avoid this situation, we amplified the learning rate to the same value as before and achieved a better result. Therefore, the early stop strategy may cause the model to miss the opportunity to get better in the new learning round.

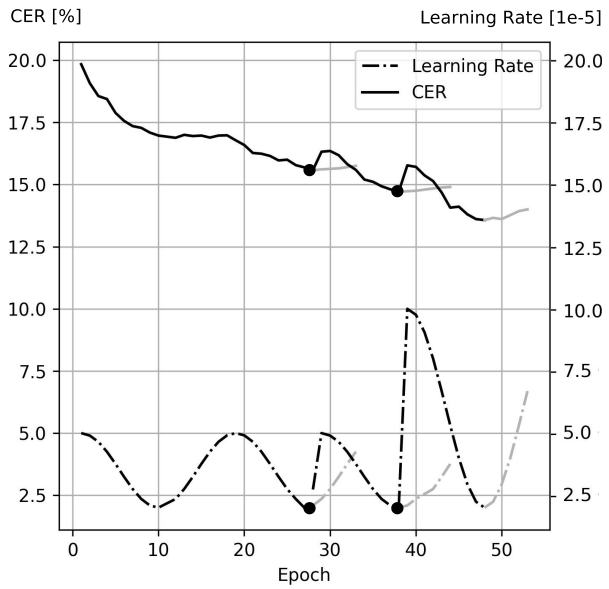

Fig. 3 Changes in learning rate and CER after using the learning rate incentive strategy. The solid line represents CER, and the dash-dot line represents the learning rate after magnification by 10000 times. The black dots mark the time when the learning rate incentive occurs.

2.3 Decoding with Prefix Beam Search

The WFST decoder is a popular algorithm in both traditional ASR systems and novel E2E systems. In our experiment, we used a token-lexicon-grammar (TLG) [30] decoding graph and obtained amazing accuracy. However, because it constructs from word piece LM, such a WFST decoding graph always has a large size, as interpreted in Sect. 1. Therefore, we prefer the beam search algorithm and propose several improvements that could be made to the algorithm. A prefix beam search algorithm is widely used to decode CTC probability on optical character recognition (OCR) [16] and ASR because of its flexibility and excellent precision, and we use this algorithm in our system. In the process of merging prefix beams, we dynamically search words in the lexicon trie tree. However, differing from most beam search algorithms, we do the acoustic pruning after a prefix is decided. We list the lexicon-based pruning strategy in Algorithm 1.

We define an object *beam* to carry a tracker responsible for searching forward in the lexicon trie and finding the path whose chars can be combined into a complete word. Therefore, instead of using a common pruning candidate list containing all char ids, we use a respective list for each beam by searching the next destinations of its tracker. This small change can reserve as many search paths as possible under the same beam size and avoid pruning the correct word, especially when the probability of a path is not high enough.

Another improvement in our work is the initialism of

Algorithm 1: Prefix beam search

Input: probability matrix M with T frames and D dimensions, blank index *bid*, blank probability threshold α, beam size β
Output: words

```
// beam is object whose properties contain at least last char id lastcid, probability of blank branch pb and non-bank branche pnb, and the tracker to a lexicon trie node ptr;
lastBeams = {};
put an initial beam into lastBeams;
for t to T do
    if M[t, bid] > α then
        skip frame t;
    end
    newBeams = {};
    for pbeam In lastBeams do
        topK = {};
        set variable cids to all char ids which ptr of pbeam can arrive to on trie;
        for cid In cids do
            put cid into topK;
        end
        put bid into topK;
        put lastcid of pbeam into topK if it is not appeared in topK;
        sort topK by the probability of each cid in topK at frame t, then keep top β candidates;
        for cid In topK do
            set variable nbeam to a new beam or an existed beam in newBeams according to what the cid and lastcid of beam are;
            update lastcid, pb, pnb and ptr of nbeam;
        end
    end
    sort newBeams by the sum of pb and pnb of each beam;
    clear lastBeams;
    select top β beams from newBeams and put them into lastBeams;
end
return words of best beam in lastBeams;
```

LM. The ASR system on embedded devices generally builds a grammar graph to recognize limited commands. However, for large vocabulary continuous speech recognition (LVCSR), we need to leverage the power of statistical LMs. If a word piece LM is used, the score is accumulated when and only when a word is outputted. The right path may have been incorrectly pruned before the word was synthesized. LM using sub-word units is one solution, but it is difficult to capture the contextual relationship between words. We use two methods to address this problem, as shown in Table 1. First, we mark the position for the initial sub-word with a specific tail symbol "_" in order that the LM can recognize the beginning of a new word, so that the implicit contextual information between words can be inferred during decoding. We call this method the position dependent (PD) sub-word LM. The second is called initialism LM. Just as the initial abbreviation of a phrase can represent the semantics of the phrase, we propose a new language modeling approach: modeling for the initialism. Because only the initial sub-words are used, this LM can be very compact in size

Table 1 Various language modeling units of an example phrase "Automatic Speech Recognition." "||" represents space. The sub-word is the character level. *PD* is short for "Position Dependent". We only mark the position of the initial letter of the word with "_" on the *PD sub-word* level.

word	AUTOMATIC||SPEECH||RECOGNITION
sub-word	A||U||T||O||M||A||T||I||C|| S||P||E||E||C||H|| R||E||C||O||G||N||I||T||I||O||N
PD sub-word	A_||U||T||O||M||A||T||I||C|| S_||P||E||E||C||H|| R_||E||C||O||G||N||I||T||I||O||N
initialism	A||S||R

and easily added to the search path as well as the sub-word model. The final score of a search path can be calculated as follows:

$$P_{beam} = a \cdot P_{CTC} + b \cdot P_{cLM} + c \cdot P_{iLM} \tag{2}$$

where a, b, c are the weights of CTC probability P_{CTC}, sub-word LM score P_{cLM} and initialism LM score P_{iLM}, respectively. Equation (3), (4) and (5) give the computing methods of these three probabilities. All probabilities are scaled to the sub-word level by length regularization. In these equations, P_b is the score of the blank path; P_{nb} is the score of the no-blank path; $P(cUNK)$ is the score of the unknown symbol of char LM; $p(iUNK)$ is the score of unknown symbol of initialism LM; N_{char} is the number of decoded chars; N_{word} is the number of decoded words; $P(S_c)$ is the score of n-gram char LM; and $P(S_i)$ is the score of n-gram initialism LM.

Parameter a can simply be set to 1.0, leaving only b and c to be control. These two values can be fine-tuned according to the decoding results. Generally, we can find the best values by parameter search. In our experiment, we used the search range of $b \in [0.05, 1.0]$, $c \in [0.05, 1.0]$.

$$P_{CTC} = (P_b + P_{nb})/(N_{frame}) \tag{3}$$

$$P_{cLM} = \begin{cases} P(cUNK) & \text{if } N_{char} = 0, \\ P(S_c)/(N_{char} + 1) & \text{if } N_{char} > 0. \end{cases} \tag{4}$$

$$P_{iLM} = \begin{cases} P(iUNK) & \text{if } N_{word} = 0, \\ P(S_i)/(N_{word} + 1) & \text{if } N_{word} > 0. \end{cases} \tag{5}$$

3. Implementation of the E2E ASR Model

Figure 4 illustrates the details of the implementation of our lightweight E2E ASR model. We name the feature extractor *LW-extractor* (lightweight feature extractor) and the encoder *LW-encoder* (lightweight encoder). In the *CDG* block in *Feature Extractor*, we use a large kernel-size CNN layer to filter out unimportant signals in the waveform. In the next*Feat block*, we use 3 layers of CNN with a small kernel size to refine the acoustic features. There are three skip

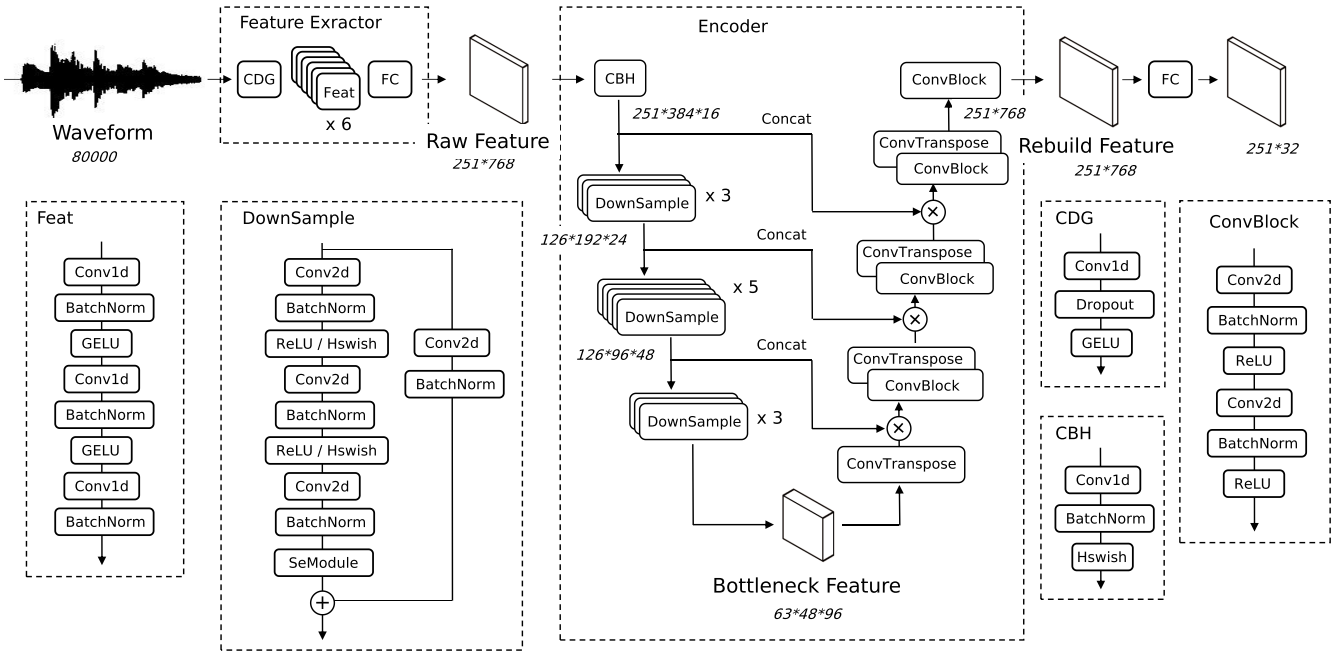

Fig. 4 The *LW-extractor + LW-encoder* model architecture. The *Hswish* is a non-linear activation function, and *SeModule* is an attention block. Both of these modules have been proposed in previous work on MobileNet-v3. The $\oplus$ is an element-wise "plus" operation. The $\otimes$ is a channel-wise "concat" operation. There is a residual connection in the *DownSample* block, which only works when the input and output dimensions are different. Whether the *ReLU* or *Hswish* and *SeModule* are used depends on the location of the *DownSample* layer.

connections in *Encoder*. The *DownSample* block is a duplicate of the MobileNet-v3 basic block but with different arguments. In FPN architecture, the encoder yields features of multiple scales. In our model, we only select the outermost output, which maintains the same size as the input feature, so it is more convenient to calculate wav2vec2.0 loss at the *pre-training* stage. Furthermore, more semantic information is available at this output.

4. Experiments

4.1 Dataset

As mentioned in Sect. 2.2, we need to use a relatively large amount (more than 100 hours) of public data for pre-training and fine-tuning our model, and for experimental comparison with previous work and other models. Then, we need to use a small amount (less than 100 hours) of domain data for transfer learning to simulate the deployment of our ASR system.

During the *pre-training* and *fine-tuning* training steps, we used the Librispeech speech dataset, which is an open-source corpus including 960 hours of training data of spoken English with a sampling rate of 16 kHz. In the *transfer-learning* step, we used a domain dataset containing about 32 hours of spoken English. This is a spoken English corpus collected for the purpose of developing an ASR system for police equipment. The speakers were adult North American police officers, and the recording environments were on the street and in the office, so the audio includes some street noise and office background noises. It was collected in the near field by Streamax Technology staff. The distance between the lips and the microphone was approximately 30 to 50 cm. The sampling rate was 16 kHz. This dataset consists of some control commands for the embedded equipment, such as:

Start recording

and some conversations during street patrols and rests in the office, such as:

Get out of the car and put your hands up now.

The Streamax dataset was split into two parts. The training dataset included 30 hours data (including 12,446 utterances). We used 2 hours data (including 844 utterances) named *Streamax* as the test dataset.

4.2 Evaluation Measures

Some evaluation measures were used to evaluate the performances of the ASR models in our experiments. We used CER to evaluate the accuracy of the ASR models, the number of parameters to evaluate the size of the model, and the memory occupation in the hard disk of model files to evaluate the size of the decoding graph. Finally, to evaluate the speech processing time, we used real-time factor (RTF) and CPU occupancy to evaluate the performance of the ASR models when the models were deployed in an embedded

device. Lower values of these factors indicate higher performance.

4.3 Details of Training Conditions

The *Adam* optimizer was used throughout our experiment. In the *pre-training* step, we adopted an initial learning rate of 0.0005 and trained the model with 64 NVIDIA TITAN RTX GPUs. The learning rate scheduler is a polynomial decay policy. Next, to obtain good initial parameters to speed up the convergence, we did a speculative job. We first conducted the supervised CTC training with a configuration that was provided by the wav2vec2.0 framework: all 960 hours of the labeled Librispeech training dataset, an initial learning rate 0.0001, and a tri-stage learning rate scheduler. This trick enabled us to get an initial model with a CER 20% on a *test-clean* dataset. Then, in the *fine-tuning* step, we used a single GPU with batch size 4. We cropped utterances whose duration was out of the range of 0.5 s to 30 s. As introduced in Sect. 2.2, we used an initial learning rate 0.00005, a cosine annealing scheduler, and the learning rate incentive policy mentioned in Eq. (1).

4.4 Evaluation of E2E Model

Our model takes 1-dimensional waveform as input. Both *pre-training* and *fine-tuning* are trained on all 960 hours of training data of the Librispeech corpus. For better comparison, we first trained the wav2vec2.0-small model in our training environment. The final model parameter was 94.4 M, and the CER on the Librispeech test-clean dataset was 2.89%. Then, we trained all models in this experiment using the same greedy search decoder used with the reference model.

Let us now compare the customized models of ResNet-18 and MobileNet-v3 backbone, both classical models used on embedded devices. Both models use 128-dimensional fBank features with Int8 quantization. All models compute CTC error with letter level. We tried two feature extractors: a multiple-layer full convolutional feature extractor named *CNN-extractor*, which has the same structure as wav2vec2.0 but without group normalization and layer normalization, and our proposed feature extractor, *LW-extractor*. The FPN feature encoding module tried a multiple-layer residual CNN named *Res-encoder* and our *LW-encoder*. Table 2 summarizes the number of parameters and the CERs of various models using the greedy search decoder. In this experiment, compared to the *ResNet-18* model, the *MobileNet-V3* model converged more easily and achieved better accuracy with only 0.54 M parameters. We have shown that the MobileNet-v3 network using only the CNN backbone has excellent potential for deployment on mobile devices.

Then, we tried a combination of *CNN-extractor* and *Res-encoder* and achieved the CER 6.64% with 39.87 M parameters, which is acceptable compared with previous work [14], [31]. We then tested the pair *CNN-extractor* + *LW-encoder*, which reduced the number of parameters to 5.19 M while CER increased by 4.75%. Finally, the proposed *LW-extractor* + *LW-encoder* model achieved a relatively good performance that best met our expectations in our experiment, with a number of only 0.79 M parameters and a CER of 13.57%. Note that the parameter includes the entire model: feature extractor, encoder, and projector. Finally, we used the domain dataset to implement the *transfer-learning* step of our proposed model and obtained a CER 8.52% on the *Streamax* test dataset. Although we did not reach our goal of reducing the CER below 5% on the domain dataset, we were able to reduce the CER by 11.52% compared to the previous model. This shows that transfer training can significantly improve the ASR system's ability to recognize speech in a specific domain and ensure its accuracy after deployment. The next section describes experiments in which the decoder was optimized to reduce the CER further.

Table 2 Number of parameters [M] and CERs [%] of various models using greedy search. *ResNet-18* and *MobileNet-v3* handle quantified fBank features. Other models take waveforms as an input source. Transfer learning is implemented based on our *LW-extractor* + *LW-encoder* mode.

	params	**Librispeech test-clean**	**Streamax** (conversation)
wav2v2c2.0-small	94.4 M	2.89	–
ResNet-18	3.61 M	57.3	–
MobileNet-V3	0.54 M	15.45	–
CNN-ext. + Res-enc.	39.87 M	6.64	–
CNN-ext. + LW-enc.	5.19 M	11.39	–
LW-ext. + LW-enc.	0.79 M	13.57	20.04
Transfer Learning	–	–	8.52

4.5 Comparison of Decoding Methods

In this experiment, we trained language models using all transcripts of our domain training dataset. Table3 lists the memory size of the graph file in the hard disk and the CER on *Streamax* test dataset using the various decoders.

We first decoded using a WFST decoder. A letter-level T, a word-to-letters L, and a 3-gram word-level G compose the TLG decoding graph by a sequence of operations of a finite state transducer. The process is shown in Eq. (6).

$$TLG = T \circ min(det(L \circ G)) \tag{6}$$

The symbol $\circ$ represents the "compose" operation, *det* is "determine," and *min* is "minimize." The second line of Table 3 shows. As far as accuracy is concerned, WFST decoder achieved good performance, with the best CER 3.99% in our experiments. However, as we had predicted, the graph size of 662 MB was too large to deploy on embedded devices.

We then evaluated the improved prefix beam search algorithm. Because the beam search decoder dynamically expands the search path, the lexicon and the original LM are the main parts of the decoding graph. Therefore, in this experiment, we have considered the file size of the N-gram LM and word-to-letter lexicon. All LM were trained with the KenLM [32] toolkit and compressed to binary format.

Table 3 Graph file size [MB] and CERs[%] on *Streamax* dataset using various decoders. The *Graph size* is the memory size in the hard disk of the LM file.

	Graph size	**CER**
greedy search	–	8.52
WFST decoding	662 MB	3.99
prefix beam search		
+ 4-gram PD character LM	1.59 MB	7.79
+ 4-gram character LM	1.47 MB	5.13
+ lexicon trie pruning	1.47 + 0.77 MB	5.02
+ 4-gram initialism LM	1.47 + 0.77 + 0.57 MB	4.86
flashlight lexicon decoder		
+ 3-gram word LM	143.95 + 0.77 MB	10.76

Table 4 CPU usage and RTF when our ASR system ran on the embedded device. *ResNet-18* is the performance when using the 128-dimensional fBank features, the ResNet18 encoder, and the beam search algorithm mentioned in Sects. 2.1 and 4.4. *LW-ext.* + *LW-enc.* is the proposed E2E ASR model.

	CPU	**RTF**
ResNet-18	25%	0.69
LW-ext. + LW-enc. (proposed)	16%	0.23

The lexicon used in our experiment contained 89, 123 words with a file size of 0.77 MB. We compared the results using 4-gram char-level LM and 4-gram PD char-level LM. Using PD LM achieves worse accuracy. This indicates that when lacking text data, the PD method is not enough to model the inter-word and between-words relationship at the same time. We then superimposed the prefix-dependent lexicon trie pruning trick on 4-gram char LM to further improve the accuracy; CER dropped by 0.11% when we did this. Finally, we added the 4-gram initialism LM. After combining the char-level LM, lexicon pruning, and initialism LM, we obtained the best result (CER 4.86%) of our experiment. Although it was 0.87% higher than the CER using the WFST decoder, the total size of the decoding graph was 2.81 MB and only four-thousandths of the former. Finally, we compared the lexicon decoder in the fairsq's library [33] used in wav2vec2.0. We built a 3-gram word LM. Regardless of the file size of the LM or CER, our decoder performed better than it in this task.

4.6 Evaluation on Embedded Device

We also deployed our ASR system on an embedded device developed by Streamax Technology Co., Ltd. This device, which has a configured Android OS and an ARMv7 CPU with four cores, is mentioned in Sect. 2.1. As shown in Table 4, our model achieved significantly reduced CPU usage and RTF. Note that the test was conducted when no other applications were running on the device, and sufficient resources were available to ensure the running condition of our ASR algorithm. The improvement in CPU utilization and RTF performance was due to the small size of the model and the delegation of all inference processing to the NPU, which allowed us to take full advantage of the hardware performance of the embedded device.

5. Conclusion

In this paper, we have presented a lightweight E2E ASR model that is easy to deploy for low-resource embedded devices. The two main contributions of our model are as follows:

- the ASR model mainly uses convolution layers, which enables it to be supported by most SDKs,
- the ASR model size is relatively small and consumes low levels of resources while still guaranteeing good accuracy and RTF.

Our model consists of three modules: the feature extractor, the encoder, and the projector. The feature extractor can extract acoustic representations from speech waveforms using multiple convolutional layers with a small kernel size. In the encoder, we adopt an FPN architecture to fuse hidden features to make up for the shortcomings of the CNN in capturing long-distance context information. In the training stage of the model, to achieve the best performance, we optimize the learning rate decay strategy to squeeze the convergence ability. In the decoding stage, we propose an improved method for the prefix beam search algorithm: prefix-based lexicon trie pruning and the initialism LM. This allowed us to build a decoder with competitive accuracy using only a few memory resources. The proposed system has demonstrated that ASR technologies could be effectively implemented in more practical developments.

We have some ideas for how the lightweight E2E ASR model could be further optimized in the future. First, we could combine our proposed *LW-extractor* and a wav2vec2.0-based transformer encoder to pre-train the feature extractor. Then, after the training, we could use just the *LW-extractor* and connect our proposed *LW-encoder* to pre-train the encoder. This could result in better performance of the feature extractor. Second, we could also use the pre-trained feature extractor and a transformer encoder provided by wav2vec2.0 as teacher models to train our modules with knowledge distillation [34]. With such an approach, we would benefit from the state-of-the-art performance of a self-supervised training strategy.

Acknowledgments

The part of this work was supported by JSPS KAKENHI Grant Number JP21H00901. We would like to thank Bojun Long and Huan Yuan for their helpful discussion and their help with data collection.

References

[1] G. Hinton, L. Deng, D. Yu, G.E. Dahl, A.r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T.N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," IEEE Signal Process. Mag., vol.29, no.6, pp.82–97, 2012.
[2] T. Hori and A. Nakamura, Speech Recognition Algorithms Using

Weighted Finite-State Transducers, Springer, Cham, 2013.
[3] S. Schneider, A. Baevski, R. Collobert, and M. Auli, "wav2vec: Unsupervised pre-training for speech recognition," Proc. INTERSPEECH 2019, pp.3465–3469, 2019.
[4] A. Baevski, S. Schneider, and M. Auli, "vq-wav2vec: Self-supervised learning of discrete speech representations," Proc. International Conference on Learning Representations (ICLR), pp.1–12, 2020.
[5] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," Proc. 34th Conference on Neural Information Processing Systems (NeurIPS 2020), pp.1–12, 2020.
[6] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An asr corpus based on public domain audio books," Proc. 2015 IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP), pp.5206–5210, 2015.
[7] K. Maekawa, "Corpus of Spontaneous Japanese: Its design and evaluation," Proc. ISCA & IEEE Workshop on Spontaneous Speech Processing and Recognition, 2003.
[8] B. Zhang, H. Lv, P. Guo, Q. Shao, C. Yang, L. Xie, X. Xu, H. Bu, X. Chen, C. Zeng, D. Wu, and Z. Peng, "WENETSPEECH: A 10000+ hours multi-domain mandarin corpus for speech recognition," Proc. 2022 IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP), pp.6182–6186, 2022.
[9] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlíšek, Y. Qian, P. Schwarz, J. Silovský, G. Stemmer, and K. Veselý, "The kaldi speech recognition toolkit," Proc. IEEE 2011 Workshop on Automatic Speech Recognition and Understanding, 2011.
[10] Y. Wang, C.S. Leow, A. Kobayashi, T. Utsuro, and H. Nishizaki, "ExKaldi-RT: A real-time automatic speech recognition extension toolkit of Kaldi," Proc. 2021 IEEE 10th Global Conference on Consumer Electronics (GCCE), pp.320–324, 2021.
[11] P. Guo, F. Boyer, X. Chang, T. Hayashi, Y. Higuchi, H. Inaguma, N. Kamo, C. Li, D. Garcia-Romero, J. Shi, J. Shi, S. Watanabe, K. Wei, W. Zhang, and Y. Zhang, "Recent developments on ESPnet toolkit boosted by conformer," Proc. 2021 IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP), pp.5874–5878, 2021.
[12] S. Kim, T. Hori, and S. Watanabe, "Joint CTC-attention based end-to-end speech recognition using multi-task learning," Proc. 2017 IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP), pp.4835–4839, 2017.
[13] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen, J. Chen, J. Chen, Z. Chen, M. Chrzanowski, A. Coates, G. Diamos, K. Ding, N. Du, E. Elsen, J. Engel, W. Fang, L. Fan, C. Fougner, L. Gao, C. Gong, A. Hannun, T. Han, L.V. Johannes, B. Jiang, C. Ju, B. Jun, P. LeGresley, L. Lin, J. Liu, Y. Liu, W. Li, X. Li, D. Ma, S. Narang, A. Ng, S. Ozair, Y. Peng, R. Prenger, S. Qian, Z. Quan, J. Raiman, V. Rao, S. Satheesh, D. Seetapun, S. Sengupta, K. Srinet, A. Sriram, H. Tang, L. Tang, C. Wang, J. Wang, K. Wang, Y. Wang, Z. Wang, Z. Wang, S. Wu, L. Wei, B. Xiao, W. Xie, Y. Xie, D. Yogatama, B. Yuan, J. Zhan, and Z. Zhu, "Deep speech 2: End-to-end speech recognition in english and mandarin," Proc. 33rd International Conference on Machine Learning, Proc. Mach. Learn. Res., vol.48, pp.173–182, June 2016.
[14] J. Park, Y. Boo, I. Choi, S. Shin, and W. Sung, "Fully neural network based speech recognition on mobile and embedded devices," Proc. 32nd Advances in Neural Information Processing Systems (NeurIPS 2018), pp.1–11, 2018.
[15] A. Graves, A.r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," Proc. 2013 IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP), pp.6645–6649, 2013.
[16] H. Scheidl, S. Fiel, and R. Sablatnig, "Word beam search: A connectionist temporal classification decoding algorithm," Proc. 2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR), pp.253–258, 2018.
[17] Y. He, T.N. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, R. Pang, Q. Liang, D. Bhatia, Y. Shangguan, B. Li, G. Pundak, K.C. Sim, T. Bagby, S.-y. Chang, K. Rao, and A. Gruenstein, "Streaming end-to-end speech recognition for mobile devices," Proc. 2019 IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP), pp.6381–6385, 2019.
[18] M. Ghodsi, X. Liu, J. Apfel, R. Cabrera, and E. Weinstein, "Rnn-transducer with stateless prediction network," Proc. 2020 IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP), pp.7049–7053, 2020.
[19] NVIDIA, "Tensorrt open source software." https://github.com/NVIDIA/TensorRT, referred on 4th/12/2022.
[20] Tencent, "ncnn." https://github.com/Tencent/ncnn, referred on 4th/12/2022.
[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L.u. Kaiser, and I. Polosukhin, "Attention is all you need," Proc. 31st Advances in Neural Information Processing Systems (NeurIPS 2017), pp.1–11, 2017.
[22] V. Liptchinsky, G. Synnaeve, and R. Collobert, "Letter-based speech recognition with gated convnets," arXiv preprint arXiv:1712.09444, pp.1–13, 2017.
[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," Proc. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.770–778, 2016.
[24] A. Howard, M. Sandler, B. Chen, W. Wang, L. Chen, M. Tan, G. Chu, V. Vasudevan, Y. Zhu, R. Pang, H. Adam, and Q. Le, "Searching for MobileNetV3," Proc. 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pp.1314–1324, 2019.
[25] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "YOLOX: Exceeding YOLO Series in 2021," arXiv preprint arXiv:2107.08430, pp.1–7, 2021.
[26] C.Y. Wang, A. Bochkovskiy, and H.Y.M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," arXiv preprint arXiv:2207.02696, pp.1–15, 2022.
[27] T.N. Sainath and C. Parada, "Convolutional neural networks for small-footprint keyword spotting," Proc. INTERSPEECH 2015, pp.1478–1482, 2015.
[28] T.Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," Proc. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.936–944, 2017.
[29] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," Proc. International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI 2015), pp.234–241, 2015.
[30] Y. Miao, M. Gowayyed, and F. Metze, "EESEN: End-to-end speech recognition using deep rnn models and wfst-based decoding," Proc. 2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), pp.167–174, 2015.
[31] K.C. Sim, F. Beaufays, A. Benard, D. Guliani, A. Kabel, N. Khare, T. Lucassen, P. Zadrazil, H. Zhang, L. Johnson, G. Motta, and L. Zhou, "Personalization of end-to-end speech recognition on mobile devices for named entities," Proc. 2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), pp.23–30, 2019.
[32] K. Heafield, "KenLM: Faster and smaller language model queries," Proc. 6th Workshop on Statistical Machine Translation, pp.187–197, July 2011.
[33] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli, "fairseq: A fast, extensible toolkit for sequence modeling," Proc. 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL), pp.48–53, June 2019.
[34] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," Proc. NIPS Deep Learning and Representation Learning Workshop, pp.1–9, 2015.

Yu Wang received his B.S. in College of Mechanical Engineering from Qinghai University in 2015. During 2019–2021, he conducted research on speech recognition systems and received the M.S. in Integrated Graduate School of Medicine, Engineering, and Agricultural Sciences from University of Yamanashi. He is now studying for a doctor's degree, and also works as an engineer in Streamax Technology Co., Ltd., engaged in the research and development of speech recognition on mobile devices.

Hiromitsu Nishizaki received his B.E., M.E., and Ph.D. degrees in engineering from Toyohashi University of Technology, Japan in 1998, 2020, and 2003, respectively. From August 2015 to March 2016, he was a visiting researcher at the National Taiwan University in the Republic of China. He has been a professor at Graduate School of Interdisciplinary Research, University of Yamahashi, since 2022. His research interests include spoken language processing and image processing using deep learning. He is also a senior member of the Institute of Electronics, Information, and Communication Engineers (IEICE) and IEEE.