APSIPA Transactions on Signal and Information Processing, 2024, 13, e

Original Paper

Detection of Arbitrary Wake Words by Coupling a Phoneme Predictor and a Phoneme Sequence Detector

Ryota Nishimura[1*], Takaaki Uno[2], Taiki Yamamoto[1], Kengo Ohta[3] and Norihide Kitaoka[2]

[1] *Tokushima University, Tokushima 770-8506, Japan*
[2] *Toyohashi University of Technology, Toyohashi 441-8122, Japan*
[3] *National Institute of Technology, Anan College, Anan 774-0017, Japan*

ABSTRACT

Most wake word (WW) detection systems used in smartphones and smart speakers only detect specific, predefined WWs such as "Hey, Siri" or "OK, Google". To build such a system, a large speech corpus consisting of many examples of the selected WWs must be collected to train the model. If we want the device to detect a different WW, collection of a new speech corpus and re-training of the model are required.

In this study, we propose a system which is capable of detecting any chosen WW without additional model training or a corpus of WW utterances, allowing users to select and use their preferred WW. Our system consists of a phoneme predictor (PP) and a phoneme sequence detector (PSD). The PP predicts phoneme sequences using acoustic features of the input speech, and outputs phoneme probability distributions. The acoustic models in the PP are trained using the Connectionist Temporal Classification (CTC) loss criterion. The PSD takes the output of the PP as input, and predicts the probability of whether or not the WW has been input. In our evaluation experiments, we performed six-phoneme WW

*Corresponding author: Ryota Nishimura, nishimura@is.tokushima-u.ac.jp

Received 29 February 2024; revised 11 June 2024; accepted 11 June 2024
ISSN 2048-7703; DOI 10.1561/116.20240014

detection. Our results showed that the proposed method achieved 90% WW detection accuracy.

Keywords: wake word, CTC, end-to-end modeling, phoneme sequence detector

1 Introduction

Smartphones and smart speakers are equipped with voice assistants that can be controlled using speech. Apple's "Siri"[1] [2], Google's "Google Assistant"[2] [21], and Amazon's "Alexa"[3] [30] are examples of these voice assistants, which give users the ability to perform hands-free operations, even at a distance, by vocalizing specific instructions to the device. The term "Wake Word" (WW) refers to the words spoken by users to activate these voice assistant applications.

Most current speech recognition systems employ a large vocabulary continuous speech recognition (LVCSR) model [12]. These models require massive computational resources however, so it is difficult to run them constantly on small devices such as smartphones and smart speakers, which have limited computational resources and battery power. Therefore, a lightweight WW detection system that only detects specific, predefined WWs, such as "Hey, Siri," "OK, Google," or "Alexa", is always running on the smart device in order to activate the speech recognition system only when the voice assistant is needed.

Training a WW detection system requires a large amount of speech data for the target WW, and the system can only detect the learned WW. To change a WW, a large amount of speech data for the new WW must be collected and the model retrained. The purpose of this study is to construct a system that can detect any WW without requiring an additional corpus of WW speech or retraining of the model.

This paper consists of five sections. Section 2 provides an overview of related research. In Section 3, we describe our proposed wake word detection system. In Section 4, we explain our evaluation experiments and provide the results of these experiments when using the proposed method. In Section 5, we discuss the performance of our proposed method. Finally, in Section 6 we present the conclusions of this study.

[1] https://www.apple.com/siri/
[2] https://assistant.google.com/
[3] https://developer.amazon.com/alexa

2 Related Work

2.1 Wake Word Detection

WW Detection has also been called Magic Word Detection (MWD) or Keyword Spotting (KWS) in previous studies. The main task of WW Detection is to detect only specific words at a small computational cost, which trigger activation of speech recognition or other systems.

MWD systems using LVCSRs have long been studied [20, 26, 17, 23]. In recent years, systems based on neural networks (NN) have also become mainstream [2, 21, 30, 6, 1, 16]. Models that can (or aim to) perform sequential processing in real-time, on devices such as smartphones and smart speakers, have also been developed [27, 4, 19, 5, 11]. MWD systems can be divided into two types; closed systems that can detect only specific words such as "Hey, Siri" or "OK, Google", for example, and open systems that allows users to freely set their own wake words. In previous studies on closed MWD systems, spectrograms have been used as the system input, and detection methods have generally been based on convolutional neural networks (CNNs) combined with recurrent neural networks (RNNs) [14]. However, the features propagated to the RNN are one-dimensional vectors, thus the time-frequency relationship is lost, so RNNs tend to lose temporal history information. Therefore, the use of Long Short-Term Memory (LSTM), to propagate temporal history information efficiently and selectively, is becoming more popular.

In contrast, open MWD systems use speech as the system input to a speech recognition system, and use the output, such as text or phoneme sequences, and their related probability distributions, to detect the WW. This allows any word to be set and detected as a WW without the need for a WW corpus or additional model training. However, most previously proposed open MWD systems are less accurate than closed systems, because they use large-scale speech recognition systems which can encounter problems such as recognition errors or unknown words.

In our proposed method, which is an open MWD system, the input speech is first converted into a phoneme probability distribution using a phoneme predictor, then the probability values of the phonemes included in the WW are extracted from the phoneme probability distribution. These probability values are then input, in a time series, to a phoneme sequence detector, which is a neural network (NN) used to detect the WW. This method allows us to construct a Wake Word detection system without a speech corpus for a specific wake word. In addition, changing a Wake Word is easy because the user only needs to supply the phonemes of the new wake word. Moreover, the proposed system can be used at low computational cost because it does not use an LVCSR model.

2.2 Connectionist Temporal Classification (CTC)

Connectionist Temporal Classification (CTC) [8] is a loss function that estimates phoneme sequences without using a Hidden Markov Model (HMM) [18] or labels. The advantage of using CTC is that the temporal alignment between the speech and labels is automatically estimated, therefore, there is no need to assign detailed temporal labels to the training data. CTC models are End-to-End (E2E) models with Recurrent Neural Network structures, allowing them to directly obtain the output from the input without the multi-stage processing required by conventional machine learning models.

The CTC outputs phonemes and <blank> tokens. The model takes acoustic features as input and outputs a phoneme posterior probability distribution. In the time series of phonemes estimated for each speech frame, these continuous phonemes are combined into a single phoneme, and the <blank> tokens are removed to obtain the final output of the phoneme sequence. For the word "cat", when the acoustic features are input to the model and "- - c - a a - t - -" is output as the phoneme prediction. After combining identical consecutive phonemes in the sequence into a single phoneme, we obtain the results "- c - a - t -". Then, by removing the <blank> tokens, the sequence "cat" is obtained.

CTC learns to maximize the posterior probability of the correct phoneme sequence by minimizing the CTC loss.

2.3 End-to-End Speech Processing Toolkit (ESPnet)

In this study, we used the End-to-End Speech Processing Toolkit (ESPnet) [29] to build our phoneme prediction model. ESPnet is an open source toolkit that supports various spoken language processing tasks, such as speech recognition and speech synthesis. It consists of a Python library of end-to-end models and recipes written in shell script. The End-to-End model is created using PyTorch, which allows flexible model extension. The recipes are in a format that is based on the one used by the Kaldi speech processing toolkit, and allow batch execution during experiments. These recipes include all of the steps necessary to reproduce experiments, such as data download, pre-processing, feature extraction, model construction, and model evaluation. The recipes are written in script format, which makes it easy to reproduce experiments. In this study, we ran an automatic speech recognition recipe on ESPnet to build a phoneme predictor. However, since the experiments in this study do not involve language models, the recipe skips the stages related to language model construction, so only the commands and programs corresponding to the processes necessary for model construction and the recognition experiments are executed.

3 Wake Word Detection Method

In this study, we propose a WW detection system which predicts phoneme sequences in order to detect the WW selected by the user. Figure 1 shows a schematic diagram of the proposed system. The system consists of two stages; a phoneme predictor and a phoneme sequence detector. First, the input WW speech is converted into acoustic features, such as Mel-frequency cepstral coefficients (MFCCs), which are then input into the phoneme predictor. The phoneme predictor calculates the phoneme probability distribution for the input acoustic features over time, and outputs a time series of the phoneme probability distribution. Next, the probabilities of the phonemes in the target WW and of <blank> are extracted from the input time series of the phoneme probability distribution. A vector of the time series of these probabilities is then input to the phoneme sequence detector, and the probability that the target frame is at the end of the WW is output as a probability value. When the output probability value is greater than a pre-determined threshold value, the user's WW is detected.

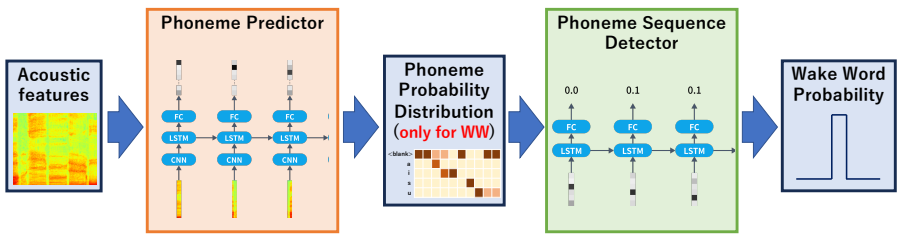

Figure 1: Proposed wake word detection method.

Our model has been released as open source software (MIT license) on GitHub.[4]

3.1 Phoneme Predictor

Figure 2 shows the structure of the phoneme prediction model. The phoneme predictor (PP) uses a general acoustic model trained using the CTC loss criterion. The model's input consists of the acoustic features of the user's WW, and the output consists of 49 different phoneme probability distributions, consisting of phonemes, unknown phoneme tokens <unk> and <blank> tokens. Normally, CTC outputs a phoneme sequence that combines continuously repeated phonemes excluding blanks, but our proposed system uses the probability distribution itself.

[4] https://github.com/kitaoka-lab/WakeWord_uno

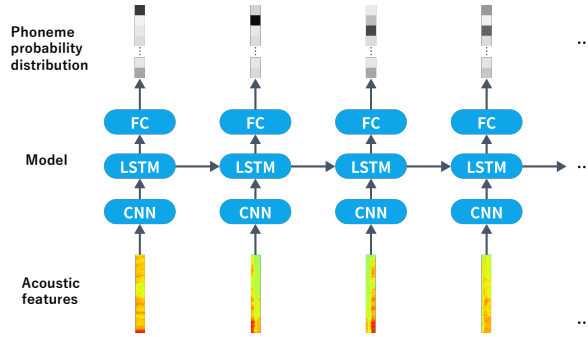

Figure 2: Phoneme prediction model.

Instead of determining a single phoneme for recognition by the phoneme predictor, the latter stage of the phoneme sequence detector looks at the temporal distribution of probability, which enables WW recognition that is robust to phoneme recognition errors. Initially, the PP consists of a large speech recognition model, but once the WW is determined knowledge distillation is automatically performed, and the model is reconfigured into a smaller, phoneme sequence detector-based model, which can operate at a lower computational cost.

The proposed method requires a fairly generic phoneme predictor that has been trained on a large amount of data, so we used CTC since it can be trained without the need for detailed labels.

Details of our phoneme predictor model are provided in Section 4.2.1.

3.2 Phoneme Sequence Detector

3.2.1 Picking out Phoneme Probability Distribution for User's WW

To prepare the input for the phoneme sequence detector, our proposed system picks out the probabilities for the phonemes of the user's chosen WW from among all of the phoneme probabilities computed by the phoneme predictor. As shown in Figure 3, the probabilities of the component phonemes of the example WW "aisu" and the probability of <`blank`> are extracted from the phoneme probability distribution (the 49-dimensional output of the phoneme predictor). When a series of phonemes is detected with the same phonemes that appear in the target WW, the probability of each phoneme is entered in the order in which they are encountered. The WW detection model is trained to judge whether the phonemes in the WW appear in order from beginning to end, as shown in Figure 4. Therefore, the operation of the detector is independent of the content of the WW. The probabilities of phonemes are arranged in the order in which the phonemes occur in user's WW, in order

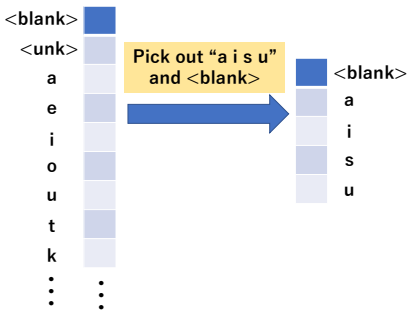

Figure 3: Picking out probabilities of phonemes for user's wake word “aisu”.

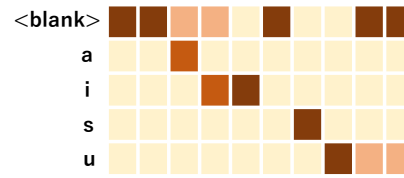

Figure 4: Phoneme probability distribution for user's wake word “aisu” (4 phonemes + blank).

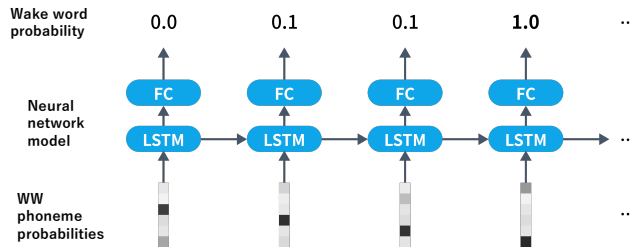

Figure 5: Phoneme Sequence Detection model.

to detect the target WW. Note that although the phoneme sequence detector (PSD) is independent of phonemes in the WW, its architecture is based on the number of phonemes in the WW, therefore it is necessary to construct a PSD model that corresponds to the length of the user's WW.

3.2.2 Phoneme Sequence Detection Model

Figure 5 shows the structure of the phoneme sequence detector model. The model is constructed using an LSTM [9] and a fully connected layer (FC). As described above, the probabilities of the component phonemes of the WW and the probability of <blank> are extracted from the phoneme probability distribution, arranged in order, and converted into a vector that is input to the model. The output of the model is the probability that the input frame is the end of the WW. The three frames from the end of the targeted speech (i.e., the WWs) are then assigned the label “1”, while the other frames are assigned the label “0”.

Our model utilizes LSTM instead of the more recent Transformer or Conformer architectures because our goal is to create a model that can more efficiently process sequential data, a task at which both Transformer and

Conformer are less effective. Additional details of the PSD model are provided in Section 4.2.2.

4 Experiment

4.1 Experimental Conditions

The training data used to train the phoneme predictor and phoneme sequence detector was obtained from the Corpus of Spontaneous Japanese (CSJ)[5] [13], a database for spoken language research that contains a large amount of spontaneous Japanese speech, as well as additional information useful for speech research. The CSJ corpus is available to the public for a fee, and can be licensed commercially, but when used for research the cost is nominal. The CSJ is one of the world's best speech research databases in terms of both quality and quantity. For this study, only core data whose phonemes are transcribed into eXtensible Markup Language (XML) were used. Details of the data set used in our experiments are shown in Table 1. A total of about 129 hours of audio data, consisting of 10,865 sentences, was used. This data was divided into a training dataset containing 6,826 sentences, and an evaluation dataset containing 4,039 sentences. In addition to the original speech selected from the CSJ corpus, the same audio data was expanded by converting it to 0.9 times and 1.1 times its original speed.

Table 1: Speech dataset

Duration	129 h 12 m 2 s
Total sentences	10,865
Training data	6,826
Test data	4,039

4.2 Model Training

4.2.1 Phoneme Prediction Model

ESPnet2 was used as the training framework for the phoneme predictor, and the model was built by running automatic speech recognition (ASR) recipes. The CSJ data described above was used as training data, and the PhonemeEntity attributes were extracted from CSJ BaseXML to generate the correct phoneme sequences for the training data.

[5] https://clrd.ninjal.ac.jp/csj/en/

Mel-Frequency Cepstral Coefficients (MFCCs), which are acoustic features, were used as the input to the model, and were computed using the following settings:

1. Short-time Fourier Transform
   (a) Sampling rate = 16 kHz
   (b) Window function = Hanning window, Window of 32 ms
   (c) Frame shift = 8 ms (overlap = 24 ms)
2. Mel filter bank = 80
3. Discrete Cosine Transform
4. Cepstral Mean Normalization (CMN)
   (a) For each frame value, subtract the mean value of the cepstral time series until the target frame.

Thus, 80-dimensional MFCCs were obtained and used as input to the phoneme prediction model.

Regarding the Convolutional Neural Network (CNN), VGGnet [25], which is often used for object/image recognition, was used. The VGGnet was configured as follows:

1. Input dimensions = 80
2. Output dimensions = 2,560
3. Frame length = 32 ms (because there are two layers of MaxPooling with a size of 2×2)

The encoder was an LSTM model with 2,560 input dimensions, 4 layers, and a hidden layer with 1,024 dimensions. The output also contains 1,024 dimensions. The CTC decoder had 1,024 input dimensions and 49 output dimensions.

The AdaDelta [31] optimization algorithm was used for model training, with a batch size of 24.

4.2.2 Phoneme Sequence Detection Model

Tensorflow was used as the framework for the phoneme sequence detector (PSD). Training data for the PSD model was generated using phoneme probability distributions, which are the output of the phoneme predictor. The PSD model operates as follows:

1. Picking out the phoneme probability distribution for the WW
   (a) Input: Phoneme probability distribution in 49 dimensions.

(b) Target: Probability distribution of phonemes in the WW (six phonemes in this experiment).
(c) The extracted probability distributions for the WW are ranked, and transformed into vectors that are input to the model.

The number of phonemes in the WWs used in our experiment was set to 6, since the average length of a Japanese word is approximately 3 or 4 syllables, which corresponds to about 6 phonemes. (Note that Figures 3 and 4 show examples of WWs with 4 phonemes.)

We picked several 6 phoneme words and used their phoneme probability distributions as training data for the PSD model. As described in Section 3.2.1, the probabilities of the component phonemes of the WW and the probability of <blank> are extracted from the phoneme probability distribution.

2. Assigning model output labels
   (a) The three frames from the end of the targeted speech (i.e., the WWs) are then assigned the label "1", while the other frames are assigned the label "0".

The PSD model was constructed using LSTM and FC layers, as shown in Figure 5 (Section 3.2.2). Details of the LSTMs and FCs that make up the PSD model are as follows:

1. LSTM:
   (a) Input dimensions = 7 (6 phonemes + <blank>)
   (b) Output dimensions = 128
   (c) Number of layers = 1
2. FC:
   (a) Input dimensions = 128
   (b) Output dimensions = 64
3. Output:
   (a) Input dimensions = 64
   (b) Output dimensions = 2

The activation function for the FC layer is ReLU [15], and Adam [10] was used as the optimization algorithm, with an initial learning rate of 0.001 and a batch size of 4. The learning rate was reduced by a factor of 0.2 if there was no improvement after three epochs. Sparse Categorical Cross entropy was used as the loss function.

4.3 Results

4.3.1 Phoneme Predictor

In our proposed system, the time series of phoneme probability vectors is transformed into a series of phonemes with the largest probability value at each time frame, and continuous identical phonemes are combined into one representation, as described in Section 2.2, in order to evaluate the phoneme prediction results after removing the <blank> tokens. Phoneme Error Rate (PER) is used as the evaluation method, which is calculated as shown in Equation (1).

$$PER = \frac{Substitution + Deletion + Insertion}{Number\ of\ Correct\ Phonemes} \tag{1}$$

Inference results for the evaluation data are shown in Table 2. The PER for the evaluated data was averaged 7.8%.

Table 2: Phoneme Error Rate (PER) for Test Data

Data set	Correct	Substitution	Deletion	Insertion	PER
eval1	93.3%	3.7%	3.1%	1.4%	8.2%
eval2	93.8%	3.7%	2.5%	1.2%	7.4%
eval3	93.6%	3.8%	2.6%	1.5%	7.9%

4.3.2 Phoneme Sequence Detector

A Detection Error Tradeoff (DET) curve was used as a method of evaluating the accuracy of Wake Word detection. The DET curve for the evaluation data is shown in Figure 6. The horizontal axis represents the False Rejection Rate (FRR) and the vertical axis represents the False Alarm Rate (FAR). The FRR represents how often the WW went undetected, while the FAR represents how often the a word that was not the WW was flagged as the WW. The FRR and FAR can be adjusted by changing the threshold for determining the end of the WW, based on the probability value used to determine whether or not it is the end of the Wake Word when output by the phoneme sequence detector. The formulas for calculating FRR and FAR are shown in Eqs. (2) and (3), respectively.

$$\text{False Rejection Rate (FRR)} = \frac{\text{Number of 1 labels wrongly set to 0}}{\text{Number of 1 labels}} \tag{2}$$

$$\text{False Alarm Rate (FAR)} = \frac{\text{Number of 0 labels wrongly set to 1}}{\text{Number of 0 labels}} \tag{3}$$

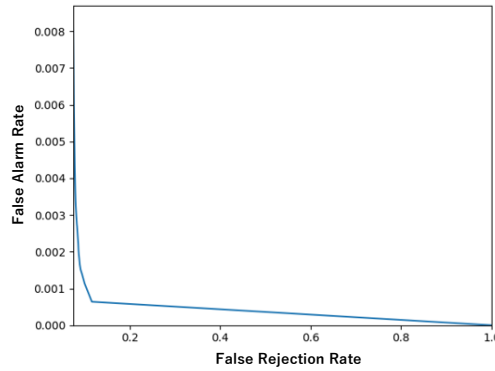

Figure 6: DET curve for test data

Table 3: Comparison of the prediction accuracy of the proposed model with that of models proposed in previous studies. The 'FS' column shows 'frame shift'.

Model	FAR (%)	FRR (%)	Acc. (%)	FA/hr	FS (ms)
Amazon [11]	0.0	2.8	-	0	-
[19]	0.002	1.7	-	1	100
KWT-1 [4, 22]	2.5	-	94.9	9,000	10
LF-MMI [27]	-	0.1	-	0.04	-
[5]	0.2	5.2	94.8	30	240
Ours	0.064	11.6	93.4	288	8

A comparison of the prediction accuracy of the proposed model with those of models proposed in previous studies are shown in Table 3, and detailed for each model are shown in Table 4. The proposed model generated 288 false alarms per hour when using CSJ speech data, with a FAR of 0.064% and FRR of 11.6%. FA/hr for the proposed model is higher than for the other models because the number of outputs per hour[6] is higher. The number of model runs (frame shift) for the proposed model falls when it is matched with the output rates of some of the other models, e.g., to $1/30^{th}$ of the reported value when compared to method [5], which means that the FA/hr of the proposed model in this case is equivalent to 9.6 FA/hr. Compared to other models, the proposed model has a higher FRR (11.6%), but its FAR and Accuracy are comparable to those of the other models, indicating that the proposed method has potential for use.

[6] $60min \times 60sec \times 1,000ms \div 8fs = 450,000\ outputs$

Table 4: Number of parameters, training dataset and language for each model.

Model	#Parameters	Dataset	Language
Amazon [11]	76M+3K+3K	SLURP(TTS) [3]	English
[19]	63.5M	Common Voice, SNIPS [7]	Multilingual
KWT-1 [4, 22]	551K	GSC	English
LF-MMI [27]	150K	SNIPS	English
[5]	93K	GSC [28]	English
Ours	103M+78K	CSJ	Japanese

[DATA ID] S00M0117

Japanese Text :えー朝の 7 時えー9 分えー我孫子という駅始発の千代田線の一両目に
English Translation :Uh... 7 o'clock in the morning uh... 9 minutes uh...
Abiko station is on the first car of the first train of *Chiyoda* line...

Correct Phonemes : e H a s a n o **sj** i cj i zj i e **H** k y u H F u N e H **h** a b i **Q** k o **H** t o y u H
e kj i sj i h a c u n o **cy** o **H** d a s e N n o i cj i **zy** o H m e n j i

Recognition Result : e H a s a n o **hj** i cj i zj i e k y u H F u N e H a b i k o t o y u H
e kj i sj i h a c u n o **cj i y** o d a s e N n o i cj i **ry** o H m e n j i

Figure 7: Examples of phoneme recognition errors.

5 Discussion

5.1 Phoneme Predictor

Examples of phoneme recognition errors are shown in Figure 7. Comparing the correct phonemes and the recognition results in this example, we can see that "`sj`" changed to "`hj`", "`cy`" to "`cj i y`", "`zy`" to "`ry`", and some phonemes were missed, such as "`H`" and "`Q`". However, since the input to the phoneme predictor only contains the probability of the phonemes that make up the WW and <`blank`> , these errors are not a problem as long as relatively high probability values are assigned to the targeted phonemes. We checked the phoneme probability distributions of the incorrectly recognized phonemes such as "`hj`", "`cj i y`", and "`H`", then assigned probability values to the correct phonemes, so these recognition errors can generally be ignored without problems. However, it was often observed that when a <`blank`> token appeared in a word, the probability of a correct phoneme was also applied to the <`blank`> token, which had a major negative impact on WW recognition. Therefore, removing <`blank`> tokens from the CTC output will allow the model to more accurately detect correct phonemes.

The proposed system was trained to recognize 46 phonemes and 3 tokens (i.e., unknown <`unk`>, start/end of sequence <`sos/eos`>, and <`blank`>) that represent phonemes not present in the training data. However, some of these phonemes and tokens are unnecessary for WW detection, because the system

only detects phoneme sequences of the WW. Removing these unnecessary phonemes from the training data, e.g., the start/end of sequence symbols and CSJ phoneme auxiliary labels (such as vowel indeterminate "FV" and consonant indeterminate "?"), may improve the performance of the phoneme predictor. We can also combine similar phonemes, such as "t", "cj", and "c" into a single phoneme, which may also improve detection accuracy. These techniques will be tested in future research. Furthermore, the proposed phoneme predictor has a large number of parameters, since it has the same architecture as a CTC-based speech recognizer. However, our WW detection method only requires the use of a small number of phoneme probabilities (those related to the WW), therefore it would be possible to construct a much smaller phoneme predictor through knowledge distillation. This is also a task for future work.

5.2 Phoneme Sequence Detector

In the example shown in Figure 7, the sequence "sj i cj i zj i (7 o'clock)" is set as the WW. The target sequence and their corresponding phoneme sequences are underlined. If the label "1" is output at the position of the final "i" phoneme, then detection of the WW is successful. When the output of the PSD model was checked, the recognition result was "hj i cj i zj i" as shown in the figure, but "1" was still output as the 'end of WW' label, indicating that the WW was still recognized. The output probability of this "1" label was only 0.57 however, which is fairly low. This low probability may increase the FRR, which is the percentage of failures to detect the Wake Word. However, since we set a low probability threshold for the PSD model, the WW was still detected despite the recognition error.

When labeling the dataset, the label "1" was only placed on the last three frames of the WW, therefore only three frames in each set of WW phonemes are positive examples. Thus, when we compare the number of positive and negative examples in each WW occurrence, the number of negative examples is relatively larger, resulting in unbalanced data. This results in most of the negative cases being recognized as negative cases, but the positive cases are also sometimes recognized as negative cases. Since the negative cases can be correctly recognized, the model's accuracy seems to be higher, however the WW cannot be reliably detected. This 'unbalanced WW data' problem can be avoided by changing the weight of the loss function or by changing the type of loss function.

5.3 Advantages of Proposed Model

In this section, the advantages of the proposed model are described.

5.3.1 WakeWord can be Customized by Users

In most WW detection systems, the WW is fixed and cannot be reset by the user. Although companies would like use the names of their products in their WWs, this is not always possible because it requires retraining of the model and a large amount of relevant data. It is also not currently possible for users to name each of their own devices. If multiple devices are present, each must have a different name, or else they will all respond to the same WW. When changing the WW of a conventional system, it is necessary to prepare at least tens of thousands of speech samples for training data, which is not feasible for individual users. Our proposed model allows WWs to be set freely, without retraining the model.

5.3.2 Real-time Wake Word Detection

In the PP, acoustic features are fed into the LSTM framewise using a CNN (VGGnet), as shown in Figure 2. In the PSD, the input values are also processed frame-wise in the LSTM. Therefore, the maximum delay before starting the computation is only one frame (window shift width of 128/16,000 Hz = 8ms). In addition, although the proposed model uses a large ASR model in its first stage (i.e., the PP stage), we plan to make it smaller through the appropriate use of knowledge distillation. The second part of the model (i.e., PSD) is very simple, consisting only of an LSTM. These features allow the proposed model to achieve less detection latency than other models, and our experimental results confirm that there is no latency problem.

Another feature of the proposed model is that its configuration makes it suitable for processing streaming data. Current mainstream deep learning models such as Transformer and Conformer require the entire history of the speech segment being processed to be input each time. In other words, if the model is to return a result for every frame, it must input audio that includes the entire speech segment possibly containing the WW target (about 2 seconds in this paper), and every frame (8ms in this case). This computation process must be applied to all of the data. In contrast, our proposed model only requires the input of the last frame of data for each output (historical data is stored in the model). This makes the proposed model suitable for real-time and streaming processing. However, even when Transformer or Conformer are used there seems to be no problem in terms of operating speed, so further investigation from the standpoint of improving accuracy is warranted.

Our investigation of the proposed model confirms that "arbitrary phoneme sequences can be recognized in combination with the phoneme predictor", and in this respect the effectiveness of the proposed method has been demonstrated. It should also be possible to further modify and improve the PSD in the future.

5.3.3 Computational Complexity

Current mainstream Transformer-based models require the input of information about the length of a word for each frame, as well as the computation of all networks, while the proposed model uses a recurrent network (LSTM), which requires only the input and computation of the newly targeted frames, thus reducing computational complexity and making it suitable for processing real-time, streaming audio data.

The number of parameters for each model are also listed in Table 4. Systems based on an ASR model have a vocabulary of several tens of millions of words. However, in the case of models that only recognize WWs, the vocabulary ranges from several hundred thousand to several tens of thousands of words. Since keyword spotting requires only a limited number of phonemes, such models are small, so we wanted our proposed model to be similar. However, the model proposed in this paper uses a configuration similar to LVCSR, and thus its vocabulary is large in size, but the size can be reduced through the use of 'knowledge distillation'. This is an issue we will address in future research. In actual operation, the scope of the ASR model will be automatically reduced through knowledge distillation once the WW is determined by the user.

5.4 Limitations of Proposed Method

The arbitrary wake word detection system proposed in this paper has some issues and limitations which will be addressed in future research. They include the following:

Future issues to be addressed include the following:

1. **Low detection performance:** It is necessary to investigate exactly how streamlining the model using knowledge distillation affects WW recognition performance.
2. **Efficiency:** After streamlining the model, the relationship between FAR and FRR, as well as optimal frame shift, should be investigated.
3. **Latency:** Since the size of the model is still fairly large, it is necessary to verify the computational efficiency of the streamlined model after completion of knowledge distillation.
4. **Comparison with previous studies:** Experiments should be performed using other corpora in order to allow direct comparisons with the results of previous studies. These corpora include the Google speech commands data sets V2,[7] [28] LibriPhrase hard dataset,[8] [24] SNIPS.[9] [7]

[7] https://storage.googleapis.com/download.tensorflow.org/data/speech_commands_v0.02.tar.gz

[8] https://github.com/gusrud1103/LibriPhrase

[9] https://github.com/sonos/keyword-spotting-research-datasets

6 Conclusion

In this study, we have proposed an arbitrary Wake Word detection system that can detect user-selected WWs using phoneme sequences. The proposed WW detector was constructed by connecting a phoneme predictor and a phoneme sequence detector. The system is characterized by its ability to accept an arbitrary WW without model retraining, use of an extremely lightweight phoneme sequence detection model in the latter stages of the model, and frame-wise sequential computation, allowing real-time operation. Based on our evaluation experiments, the proposed WW detection system can detect approximately 90% of WW utterances when the phoneme error rate (PER) of the phoneme predictor is 7.899%.

In future work, we will first investigate the proposed system's limitations, which are described in Section 5.4 , and then try to make the model lighter, more accurate, and more efficient, while also reducing latency.

Financial Support

This work was supported by JSPS KAKENHI Grant Numbers 22K19793, JP23H00493.

Biographies

Ryota Nishimura received his B.S., M.S. and Ph.D. degrees from the Toyohashi University of Technology (TUT), Japan, and joined TUT as a researcher in 2011. He was also a researcher at Nagoya University, Japan, from 2011 to 2012. He was an assistant professor at the Nagoya Institute of Technology, Japan, from 2012 to 2015, and then an assistant professor at Keio University, Japan, from 2015 to 2017. He was a researcher at Tokushima University, Japan, from 2017 to 2018, and became an associate professor there in 2018. His research interests include spoken dialog systems and spoken language information processing. He is a senior member of IEEE, and a member of International Speech Communication Association (ISCA), Asia Pacific Signal and Information Processing Association (APSIPA), the Information Processing Society of Japan (IPSJ), the Acoustical Society of Japan (ASJ), the Japanese Society for Artificial Intelligence (JSAI), the Institute of Electronics, Information and Communication Engineers (IEICE), the Association for Natural Language Processing (NLP), and the Institute of Electrical Engineers of Japan (IEEJ).

Kengo Ohta received his B.S., M.S., and Ph.D. degrees from the Toyohashi University of Technology (TUT), Japan. From 2011 to 2013, he was a Research Fellow of the Japan Society for the Promotion of Science (JSPS Fellow, DC2). He joined the National Institute of Technology, Anan College, as an assistant professor in 2013, and was a lecturer there from 2018 to 2020. He was also a lecturer at TUT from 2020 to 2021. He has been an associate professor at the National Institute of Technology, Anan College since 2021. His research interests include spoken language processing, speech recognition, speech synthesis, and spoken dialog systems. He is a member of the Institute of Electronics, Information and Communication Engineers (IEICE), the Information Processing Society of Japan (IPSJ), the Acoustical Society of Japan (ASJ), and the Japanese Society for Artificial Intelligence (JSAI).

Norihide Kitaoka received his B.S. and M.S. degrees from Kyoto University, Japan. In 1994, he joined DENSO CORPORATION. In 2000, he received his Ph.D. degree from the Toyohashi University of Technology (TUT), Japan. He joined TUT as a research associate in 2001 and was a lecturer from 2003 to 2006. He was an associate professor at Nagoya University, Japan, from 2006 to 2014, and joined Tokushima University, Japan, as a professor in 2014. He has been a professor at TUT since 2018. His research interests include speech processing, speech recognition, and spoken dialog systems. He is a member of the IEEE, the International Speech Communication Association (ISCA), the Asia Pacific Signal and Information Processing Association (APSIPA), the Information Processing Society of Japan (IPSJ), the Acoustical Society of Japan (ASJ), the Japanese Society for Artificial Intelligence (JSAI), and the Association for Natural Language Processing.

References

[1] S. Ö. Arık, M. Kliegl, R. Child, J. Hestness, A. Gibiansky, C. Fougner, R. Prenger, and A. Coates, "Convolutional Recurrent Neural Networks for Small-Footprint Keyword Spotting", in *Interspeech 2017*, 2017, 1606–10, DOI: 10.21437/Interspeech.2017-1737.

[2] Arnav Kundu and Mohammad Samragh Razlighi and Minsik Cho and Priyanka Padmanabhan and Devang Naik, "HEiMDaL: Highly Efficient Method for Detection and Localization of wake-words", in *ICASSP 2023*, 2023.

[3] E. Bastianelli, A. Vanzo, P. Swietojanski, and V. Rieser, "SLURP: A Spoken Language Understanding Resource Package", in *EMNLP 2020*, 2020, 7252–62, DOI: 10.18653/v1/2020.emnlp-main.588.

[4] A. Berg, M. O'Connor, and M. T. Cruz, "Keyword Transformer: A Self-Attention Model for Keyword Spotting", in *INTERSPEECH 2021*, 2021, 4249–53, DOI: 10.21437/Interspeech.2021-1286.

[5] A. Bittar, P. Dixon, M. Samragh, K. Nishu, and D. Naik, "Improving Vision-inspired Keyword Spotting Using a Streaming Conformer Encoder With Input-dependent Dynamic Depth", in *ICASSP 2024*, 2024.

[6] G. Chen, C. Parada, and G. Heigold, "Small-footprint keyword spotting using deep neural networks", in *ICASSP 2014*, Vol. 14, Citeseer, 2014, 4087–91.

[7] A. Coucke, M. Chlieh, T. Gisselbrecht, D. Leroy, M. Poumeyrol, and T. Lavril, "Efficient keyword spotting using dilated convolutions and gating", in *ICASSP 2019*, 2019, 6351–5.

[8] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks", in *International Conference on Machine Learning (ICML 2006)*, Association for Computing Machinery, 2006, 369–76, DOI: 10.1145/1143844.1143891.

[9] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural Computation*, 9(8), 1997, 1735–80.

[10] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization", 2017, arXiv: 1412.6980 [cs.LG].

[11] P.-J. Ku, I.-F. Chen, H. Yang, A. Raju, P. Dheram, P. Ghahremani, B. King, J. Liu, R. Ren, and P. Nidadavolu, "Hot-fixing wake word recognition for end-to-end ASR via neural model reprogramming", in *ICASSP 2024*, 2024.

[12] A. Lee and T. Kawahara, "Recent development of open-source speech recognition engine Julius", in *APSIPA ASC 2009*, 2009, 131–7.

[13] K. Maekawa, "Corpus of Spontaneous Japanese: Its design and evaluation", in *ISCA & IEEE Workshop on Spontaneous Speech Processing and Recognition (SSPR 2003)*, 2003.

[14] T. Mikolov, M. Karafiát, L. Burget, J. Černocký, and S. Khudanpur, "Recurrent neural network based language model", in *INTERSPEECH 2010*, 2010, 1045–8, DOI: 10.21437/Interspeech.2010-343.

[15] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines", in *International Conference on Machine Learning (ICML 2010)*, 2010.

[16] D. Ng, Y. Xiao, J. Q. Yip, Z. Yang, B. Tian, Q. Fu, E. S. Chng, and B. Ma, "Small Footprint Multi-channel Network for Keyword Spotting with Centroid Based Awareness", in *INTERSPEECH 2023*, 2023, 296–300, DOI: 10.21437/Interspeech.2023-1210.

[17] T. Nitta, S. Iseji, T. Fukuda, H. Yamada, and K. Katsurada, "Key-word spotting using phonetic distinctive features extracted from output of an LVCSR engine", in *ISCA & IEEE Workshop on Spontaneous Speech Processing and Recognition (SSPR 2003)*, 2003.

[18] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition", *Proceedings of the IEEE*, 77(2), 1989, 257–86, DOI: 10.1109/5.18626.

[19] P. M. Reuter, C. Rollwage, and B. T. Meyer, "Multilingual query-by-example keyword spotting with metric learning and phoneme-to-embedding mapping", in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2023, 1–5.

[20] R. C. Rose and D. B. Paul, "A hidden Markov model based keyword recognition system", in *ICASSP 1990*, IEEE, 1990, 129–32.

[21] O. Rybakov, N. Kononenko, N. Subrahmanya, M. Visontai, and S. M. Laurenzo, "Streaming keyword spotting on mobile devices", in *INTERSPEECH 2020*, 2020, 2277–81, DOI: 10.21437/Interspeech.2020-1003.

[22] 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.

[23] L. Sarı, B. Gündoğdu, and M. Saraçlar, "Fusion of LVCSR and posteriorgram-based keyword search", in *INTERSPEECH 2015*, 2015, 824–8, DOI: 10.21437/Interspeech.2015-259.

[24] H.-K. Shin, H. Han, D. Kim, S.-W. Chung, and H.-G. Kang, "Learning Audio-Text Agreement for Open-vocabulary Keyword Spotting", in *INTERSPEECH 2022*, 2022, 1871–5, DOI: 10.21437/Interspeech.2022-580.

[25] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition", in *3rd International Conference on Learning Representations (ICLR 2015)*, Computational and Biological Learning Society, 2015.

[26] R. A. Sukkar and J. G. Wilpon, "A two-pass classifier for utterance rejection in keyword spotting", in *ICASSP 1993*, Vol. 2, IEEE, 1993, 451–4.

[27] Y. Wang, H. Lv, D. Povey, L. Xie, and S. Khudanpur, "Wake Word Detection with Alignment-Free Lattice-Free MMI", in *INTERSPEECH 2020*, 2020, 4258–62, DOI: 10.21437/Interspeech.2020-1811.

[28] P. Warden, "Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition", 2018, arXiv: 1804.03209 [cs.CL].

[29] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. Enrique Yalta Soplin, J. Heymann, M. Wiesner, N. Chen, A. Renduchintala, and T. Ochiai, "ESPnet: End-to-End Speech Processing Toolkit", in *INTERSPEECH 2018*, 2018, 2207–11, DOI: 10.21437/Interspeech.2018-1456.

[30] G.-P. Yang, Y. Gu, S. Macha, Q. Tang, and Y. Liu, "On-device constrained self-supervised learning for keyword spotting via quantization aware pre-training and fine-tuning", in *ICASSP 2024*, 2024.
[31] M. D. Zeiler, "ADADELTA: An Adaptive Learning Rate Method", 2012, arXiv: 1212.5701 [cs.LG].