Wake Word Detection with Alignment-Free Lattice-Free MMI

Yiming Wang[1], *Hang Lv*[4,1], *Daniel Povey*[3], *Lei Xie*[4], *Sanjeev Khudanpur*[1,2]

[1]Center for Language and Speech Processing, [2]Human Language Technology Center of Excellence, Johns Hopkins University, Baltimore, MD, USA
[3]Xiaomi Inc., Beijing, China
[4]ASLP@NPU, School of Computer Science, Northwestern Polytechnical University, Xi'an, China
{yiming.wang,khudanpur}@jhu.edu, {hanglv,lxie}@nwpu-aslp.org, dpovey@gmail.com

Abstract

Always-on spoken language interfaces, e.g. personal digital assistants, rely on a *wake word* to start processing spoken input. We present novel methods to train a hybrid DNN/HMM wake word detection system from partially labeled training data, and to use it in on-line applications: (i) we remove the prerequisite of frame-level alignments in the LF-MMI training algorithm, permitting the use of un-transcribed training examples that are annotated only for the presence/absence of the wake word; (ii) we show that the classical keyword/filler model must be supplemented with an explicit non-speech (silence) model for good performance; (iii) we present an FST-based decoder to perform online detection. We evaluate our methods on two real data sets, showing 80%–90% reduction in false rejection rates at pre-specified false alarm rates over the best previously published figures, and re-validate them on a third (large) data set.

Index Terms: wake word detection, lattice-free MMI, alignment free

1. Introduction

Wake word detection is the task of detecting a predefined keyword from a continuous stream of audio. It has become an important component in today's voice-controlled digital assistants and smart phones. Voice-controlled devices, with wake word detection system running in the background, require a low power solution. When people wish to to interact with such devices by voice, they *wake up* the device by saying a predefined word like "Alexa" for Amazon Echo or "Okay Google" for Google Home. If the word is identified and accepted, the device turns on, i.e. goes into a state with higher power consumption to recognize and understand more complex spoken instructions [1].

HMM-based keyword-filler models are used to represent both the keyword and filler (background) models [2, 3, 4]. The keyword model consists of all valid phone sequences from the keyword, and the filler model includes all other speech and non-speech. During the decoding phase, usually the ratio of the scores with keyword graph and to the filler graph is computed for determining the presence of the wake word. With recent advances in deep learning, HMM-DNN hybrid wake word systems replace GMM-based acoustic models with a neural network to classify individual frames [5, 6, 7]. While the filler model for background speech is specified as an ergodic topology between speech and non-speech in [5, 6], it is represented as an all-phones loop in [7], increasing both the neural network model size and decoding graph size due to the increased number of modeling units. Finally, some methods add automatic speech recognition (ASR) as an auxiliary task during training.

Pure neural models abandon HMMs and completely rely on neural networks for acoustic modeling, where the subwords or even the whole word of the wake word phrase (wake phrase, for short) is directly used as modeling units. The first successful wake word detection systems of this type are proposed in [8, 9]. They use individual words in the wake phrase as modeling units to reduce the network size. However, they still need a forced alignment obtained from an existing HMM-based ASR system, to obtain training labels, which limits their applications if an ASR system is unavailable. For decoding, they adopt a fast posterior handling approach where the posterior of words is smoothed within a sliding window over the audio frames. [10, 11] use the whole wake phrase as the training target, but it still needs phone-level alignments to pretrain a small network being part of a larger one. There are also several proposals that do not require frame-level alignment for training, including max-pooling [12, 13], the attention mechanism [14, 15], and global mean-pooling [16].

It has been shown that a *sequence-level training criteria* perform better than frame-level criteria for ASR. The output in a wake word detection task, by contrast, is relatively simple. However, if the modeling units are subwords (e.g., phonemes or HMM states), wake word detection may still be considered as a sequence prediction task. Sequence-level discriminative training such as CTC loss [17] has been explored for the wake word detection task with graphemes or phonemes as subword units [18, 19, 20, 21]. Lattice-free maximum mutual information (LF-MMI) is an HMM-based sequence-level loss first proposed in [22] for ASR. In the context of wake word detection, it is recently investigated in [23], where it still requires alignments from a prior model like an HMM-GMM system to generate numerator graphs.

In this paper we propose a wake word detection system with alignment-free LF-MMI as training criterion, while not requiring any forced alignments for training.[1] Alignment-free LF-MMI was initially proposed for ASR [25]. In order to make it work for our task, we made several necessary adaptations/changes to the lexicon, HMM topology, data preprocessing for both efficiency and performance reasons. A fast online decoder is also proposed for our task. The experiments on three real wake word data sets all show its superior performance compared to other systems recently reported on the same data sets.

This work was partially supported by unrestricted gifts from Mobvoi Information Technology Company Limited, and Applications Technology (AppTek). The authors thank Hainan Xu and Tongfei Chen for valuable comments.

[1]The code and recipes are available in Kaldi [24]: https://github.com/kaldi-asr/kaldi/tree/master/egs/{snips,mobvoi,mobvoihotwords}.

2. The Proposed System

2.1. HMM Topology

Different from other traditional HMM-based keyword-filler models (e.g., those proposed in [5, 6, 7, 23] where each phoneme of the whole wake phrase corresponds to an HMM or HMM state, we propose to model the whole wake phrase (in positive recordings) with a single HMM (referred to as *word* HMM), and the number of distinct states within that HMM is a predefined value which is not necessarily proportional to the number of phonemes in its pronunciation. We argue that using a fixed number of HMM states, which is usually less than the number of the actual phonemes, has enough modeling power for the wake word task. Similarly, we use another HMM of the same topology (referred to as *freetext* HMM) to model all non-silence speech (in negative recordings). From our preliminary experiments we also found that having an additional HMM dedicated to non-speech sounds, denoted *SIL* and called the *"silence" phone*, is crucial for good performance. The *SIL* phone is added as *optional* silence [26] to the beginning and end of each positive/negative recording so that it can learn the actual silence properly. The resulting topologies are shown in Fig. 1.

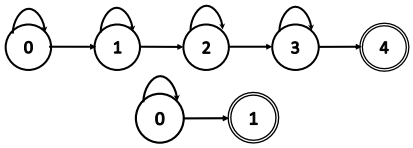

Figure 1: *The HMM topologies used for the* wake word *and* freetext *(top), and* SIL *(bottom).*

2.2. Alignment-Free Lattice-Free MMI

Lattice-free MMI (LF-MMI) loss [22] is a sequence-level criterion and can be formulated as:

$$\mathcal{F}_{\text{LF-MMI}} = \sum_{n=1}^{N} \log P(L_n|\mathbf{O}_n) = \sum_{n=1}^{N} \log \frac{P(\mathbf{O}_n|L_n)P(L_n)}{\sum_L P(\mathbf{O}_n|L)P(L)}$$

where L_n and L are the subword truth sequence and a competing hypothesis sequence respectively, and $\mathbf{O}_n$ is the input audio. In the regular LF-MMI the numerator graph used to compute the truth sequence is an acyclic graph generated from an existing GMM model. In alignment-free LF-MMI [25], the numerator graph is an unexpanded FST directly generated from training transcripts, giving more freedom to learn the alignments during the forward-backward pass in training.

For ASR, the competing hypotheses in the denominator graph are constructed from a phone LM trained from the training transcripts. On the contrary, for our wake word detection task, we manually specify the topology of the phone LM FST as in Fig. 2. One path containing the *word* HMM corresponds to positive recordings[2], and the other two correspond to negative recordings (other speech/non-speech and silence). We assign final weights in a way such that they reflect the ratio of the number of positive/negative examples in the training set.

2.3. Acoustic Modeling

Owing to efficiency and latency concerns specific to our task, the family of recurrent [27, 28] or self-attention-based [29] neural networks is not within our consideration. Instead, we use factorized TDNN (TDNN-F) with skip connections [30] for acoustic modeling. In a TDNN-F layer, the number of parameters is reduced by factorizing the weight matrix in TDNN layers [31] into the product of two low-rank matrices, the first of which is constrained to be semi-orthogonal.

As in other architectures like ResNet [32], we incorporate skip connections: each TDNN-F layer receives its immediate prior layers output as the skip connection, which is added to the input of the current layer after being scaled down by 0.66.

We use a narrow (hidden dimension is 80) but deep (20 layers) network with each output frame covering a receptive field of size 80. The output is evaluated every 3 frames for LF-MMI loss to reduce the the computation cost both in training and test time. We also find a cross-entropy regularization together with the main LF-MMI loss helpful. As a result, the total number of parameters is about 150k, with the number of targets being only 18 using the HMM topologies described in Sec. 2.1.

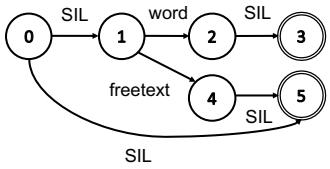

Figure 2: *Topology of the phone language model FST for the denominator graph. Labels on arcs represent phones.*

2.4. Data Preprocessing and Augmentation

Compared with the positive recordings, the negative recordings usually have a longer duration and have more variability as they can include all possible speech except the wake phrase. However we only use a single *freetext* HMM for it, making it difficult for the model to learn smoothly if batching them with positives directly (see Sec. 3.2). To tackle this problem, before training we segment all negative recordings into several chunks so that the chunk lengths are draw from the empirical distribution of the positive recordings. As any segments from the original negative recordings can still be considered as negative, we just assign a negative label to each of them.

Although all our training data is recorded in real environments with background noise, we still found that data augmentation is helpful. Therefore we apply the same type of data augmentation techniques as used in [33], making use of noise, music, background speech from the MUSAN corpus [34], simulated reverberation [35] and speed perturbation [36]. This effectively increases the amount of training data 7 times.

2.5. Decoding

We propose online decoding without lattice generation for wake word detection in the following way.

First we construct our decoding graph with a word-level FST specifying the prior probabilities of all possible word paths, in a similar way as we specify the phone language model FST in Fig. 2, except that the start state and final states are merged to form a loop. The loop allows decoding with an audio interleaving with wake words and other possible speech.

During online decoding, every time after processing a fixed-length chunk from a recording, we backtrack along the frames delimited by two most recent "immortal tokens" by calling the routine UPDATEIMMORTALTOKEN in Algorithm 1, checking if there is a wake word detected from this partial backtracking. If

[2]If we have more than one wake word as those in our Mobvoi (SLR87) data set, each wake word would correspond to one such path.

Algorithm 1 Update the Immortal Token for Backtracking

Input: activeTokList ▷ represents all current hypotheses
Output: immortalTok ▷ is global, storing the latest one

```
1:  procedure UPDATEIMMORTALTOKEN(activeTokList)
2:      emitting ← ∅
3:      for tok in activeTokList do
4:          while isNonEmittingToken(tok) do tok ← tok.prev
5:          if tok ≠ NULL then emitting.insert(tok)
6:      tokenOne ← NULL
7:      while True do
8:          if |emitting| = 1 then
9:              tokenOne ← emitting[0]; break
10:         if emitting ≠ ∅ then break
11:         prevEmitting ← ∅
12:         for tok in emitting do
13:             prevTok ← tok
14:             while isNonEmittingToken(tok) do
15:                 prevTok ← tok.prev
16:             if prevTok = NULL then continue
17:             prevEmitting.insert(prevTok)
18:         emitting ← prevEmitting
19:     if tokenOne ≠ NULL then
20:         immortalTok ← tokenOne
```

a wake word is found, we just stop decoding and trigger the system; otherwise continue the decoding process. The "immortal token" is the common ancestor of all active tokens. Line 3-5 obtain the last emitting token from each active token. Line 7-18 are trying to find the common ancestor of all active tokens. Line 19-20 update the immortal token if a newer one is found; otherwise keep the old one from the previous decoding step.

The intuition is that, if all currently active partial hypotheses are from the same token at a previous time-step, all hypotheses before that token had already collapsed to one hypothesis (due to beam search and pruning), from which we would check whether it contains the wake word in a chunk-by-chunk fashion.

3. Experiments

3.1. Data Sets

There are three real wake word data sets to be evaluated: SNIPS data set[3] [11] with the wake word "Hey Snips", Mobvoi single wake word data set[4] [15] with the wake word "Hi Xiaowen", and Mobvoi (SLR87) data set[5] [37] with two wake words "Hi Xiaowen" and "Nihao Wenwen". The statistics for each data set are summarized in Table 1. We will use the first two data sets to demonstrate the effects of several design choices in our system, and give the final results on all these three data sets when comparing our system with others. If not otherwise specified, we show our experimental results in an incremental way, meaning that later experiments would be conducted on top of the one that is better from the previous experiment. The operating points in DET curves are obtained by varying the cost corresponding to the positive path in the decoding graph while keeping the cost corresponding to negative path at 0. 40-dimensional MFCC features are extracted in all the experiments.

[3] https://github.com/snipsco/keyword-spotting-research-datasets

[4] This data set is not publicly available.

[5] https://www.openslr.org/87

Table 1: *Statistics for the three wake word data sets.*

Name	Train		Dev		Eval	
	#Hrs	#Utts (#Positive)	#Hrs	#Utts (#Positive)	#Hrs	#Utts (#Positive)
SNIPS	54	50,658 (5,799)	24	22,663 (2,484)	25	23,072 (2,529)
Mobvoi	67	74,134 (19,684)	7	7,849 (2,343)	7	7,841 (1,942)
Mobvoi (SLR87)	144	174,592 (43,625[6])	44	38,530 (7,357)	74	73,459 (21,282)

3.2. Effect of Negative Recordings Sub-segmentation

We first show the effect of sub-segmenting negative recordings on training. We start from the training data with only speed-perturbation applied. To keep consistent with the numbers reported in others' work on the same data sets, false rejection rate (FRR) is reported in Table 2 at 0.5 false alarms per hour (FAH) on the SNIPS data set, and at 1.5 FAH for Mobvoi. Apparently without sub-segmentation the performance is far from satisfactory, indicating the alignments learned with the LF-MMI system is poor. We also inspected the training/validation loss in both cases (not shown here), and found that there is severe overfitting when training without sub-segmentation.

Table 2: *Effect of sub-segmentation of negative recordings.*

FRR(%)	SNIPS (FAH=0.5)	Mobvoi (FAH=1.5)
w/o sub-segmentation	67	47
w/ sub-segmentation	**0.6**	**5.6**

3.3. Effect of Data Augmentation

We augment the training data using the MUSAN corpus in the following way: we randomly apply additive noise from the "babble", "music" and "noise" data sets separately on each copy of the original training data once. For each training example, "babble" is added as background noises 3 to 7 times with SNRs ranging from 13 to 20; music is added as background noises once with SNRs ranging from 5 to 15; noise is added as foreground noises at the interval of 1 second with SNRs ranging from 0 to 15. Then reverberation is also separately applied to the training data using the simulated RIRs with room sizes uniformly sampled from 1 meter to 30 meters. The above procedure increase the training data by a factor of 4. We then apply 3 way speed-perturbation on top of the original training set. The augmentation strategy together results in about 7× more training data. The results before and after augmentation are shown in Table 3. It can be seen the augmentation strategy is highly effective, where FRR with SNIPS is even 0 at FAH=0.5.

Table 3: *Effect of data augmentation.*

FRR(%)	SNIPS (FAH=0.5)	Mobvoi (FAH=1.5)
w/o data augmentation	0.6	5.6
w/ data augmentation	**0**	**0.4**

3.4. Effect of Alignment-Free LF-MMI Loss

To compare our proposed alignment-free LF-MMI loss with regular LF-MMI and conventional cross-entropy loss for our task, we train a phoneme-based HMM-GMM system to generate the numerator lattice (for regular LF-MMI loss) or the forced alignments (for conventional cross-entropy loss) for the same sub-segmented and augmented training data. the network architectures are the same as what is used for alignment-free

[6] The statistics include two wake words.

LF-MMI training except the final layer (depending on the loss being used). The results in Table 4 validate that LF-MMI loss is generally advantageous to the frame-level cross-entropy loss in the wake word detection task, and alignment-free LF-MMI loss achieves better performance than regular LF-MMI on SNIPS, but worse on Mobvoi. It is worth noting that we believe SNIPS results are more indicative of performance, as after manually listening to the false alarms in Mobvoi at this specific operating point, we found that all the false alarms (9 in total) are actually intentionally pronounced with a different tone on the last character "wen", which is extremely difficult for the model to learn given the limited amount of data; some would even argue that those examples should be labeled as positive cases in order for the model to accommodate Chinese speakers with accents. The better performance of the system with alignment-free LF-MMI loss is possibly due to the capability of learning flexible alignments than GMM models.

Table 4: *Effect of alignment-free LF-MMI loss.*

FRR(%)	SNIPS (FAH=0.5)	Mobvoi (FAH=1.5)
cross-entropy	0.6	3.5
regular LF-MMI	0.1	**0.2**
alignment-free LF-MMI	**0**	0.4

3.5. Regular LF-MMI Refinement

The experiment from the previous section motivates us to do an additional experiment investigating how the regular LF-MMI performs when it gets alignments from the alignment-free LF-MMI system instead of from a GMM model, and whether the regular LF-MMI training could further improve the performance as a refinement of our existing system. To this end, we compare the three systems in Table 5. Note that as we already achieve FRR=0 at FAH=0.5 with SNIPS using our alignment-free LF-MMI system, we set the operating point at a smaller FAH (0.04) for it. Table 5 demonstrates that further improvement can be obtained by running an additional regular LF-MMI training on top of alignment-free LF-MMI, suggesting an optional refinement stage for better performance.

Table 5: *Effect of using alignments from Alignment-free LF-MMI for regular LF-MMI.*

FRR(%)	SNIPS (FAH=0.04)	Mobvoi (FAH=1.5)
regular LF-MMI	0.2	**0.2**
alignment-free LF-MMI	0.2	0.4
+regular LF-MMI refinement	**0.1**	0.3

3.6. Comparison with Other Baseline Systems

We compare our proposed system with other systems recently proposed on the same data sets. We use our alignment-free LF-MMI system without refinement as the refinement is optional. The results are shown in Table 6. DET curves of our system on all the three data sets are plotted in Fig. 3.

For SNIPS data set we compare against their original paper [11], where a voice activity detection system is used to obtain frame-level wake word labels for training. While their system is already very good in term of FRR, our system even achieves FRR=0 at FAH=0.5.

For Mobvoi data set we compare our system with [15] where an attention mechanism is adopted for pooling across frames to make a prediction. They also propose an adversarial examples generation algorithm for robust training. The modeling units are wake words. Our system achieves significant better results, improving FRR by around 90% relative compared to their number at FAH=1.5.

For Mobvoi (SLR87) data set, both [37] and [13] are compared. [37] is trying to simultaneously localize the wake word and make predictions, while [13] tackles the label imbalance issue commonly found in frame-wise training of word labels. Again, both use wake words as modeling units. Note that this data set contains two wake words ("Hi Xiaowen" and "Nihao Wenwen"), and when we are evaluating for a specific one, the other one is considered as negative. We achieve 80-90% reduction in FRR than the baselines with the same FAH=1.0.

Table 6: *Comparison with other baselines.*

SNIPS	FRR(%) at FAH=0.5	
Coucke et al. [11]	0.12	
alignment-free LF-MMI (Ours)	**0**	
Mobvoi	FRR(%) at FAH=1.5	
Wang et al. [15]	∼3.6	
alignment-free LF-MMI (Ours)	**0.4**	
Mobvoi (SLR87)	FRR(%) at FAH=1.0	
	Hi Xiaowen	Nihao Wenwen
Hou et al. [37]	∼7.5	∼4.8
Hou et al. [13]	3.1	2.7
alignment-free LF-MMI (Ours)	**0.3**	**0.4**

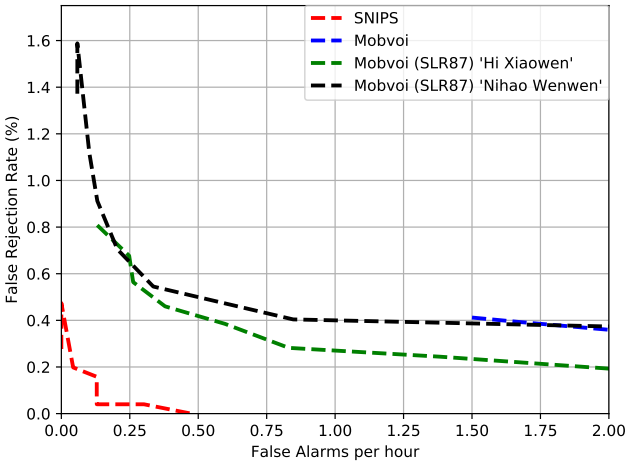

Figure 3: *DET curves for the three data sets.*

4. Conclusions and Future Work

We describe a suite of methods to build a hybrid HMM-DNN system for wake word detection, including sequence-discriminative training based on *alignment-free* LF-MMI loss, removing the need for frame-level training alignments, and whole-word HMMs for the wake word and filler speech, removing the need for training transcripts or pronunciation lexicons. These features significantly reduce model sizes and greatly simplify the training process. An online decoder tailored to wake word detection is proposed to complete the suite. The system widely outperforms other wake word detection systems on three different real-world wake word data sets. We have open-sourced our system in Kaldi, and to make it accessible to other deep learning frameworks, we are implementing a PyTorch-based version based on ESPRESSO [38] and PYCHAIN [39].

5. References

[1] Y. Wang, X. Fan, I. Chen, Y. Liu, T. Chen, and B. Hoffmeister, "End-to-end anchored speech recognition," in *Proc. ICASSP*, 2019, pp. 7090–7094.

[2] J. R. Rohlicek, W. Russell, S. Roukos, and H. Gish, "Continuous hidden markov modeling for speaker-independent word spotting," in *Proc. ICASSP*, 1989, pp. 627–630 vol.1.

[3] R. C. Rose and D. B. Paul, "A hidden markov model based keyword recognition system," in *Proc. ICASSP*, 1990, pp. 129–132 vol.1.

[4] I. Szöke, P. Schwarz, P. Matejka, L. Burget, M. Karafiát, and J. Cernocký, "Phoneme based acoustics keyword spotting in informal continuous speech," in *Proc. Text, Speech and Dialogue, 8th International Conference*, 2005, pp. 302–309.

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

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

[7] M. Wu, S. Panchapagesan, M. Sun, J. Gu, R. Thomas, S. N. P. Vitaladevuni, B. Hoffmeister, and A. Mandal, "Monophone-based background modeling for two-stage on-device wake word detection," in *Proc. ICASSP*, 2018, pp. 5494–5498.

[8] G. Chen, C. Parada, and G. Heigold, "Small-footprint keyword spotting using deep neural networks," in *Proc. ICASSP*, 2014, pp. 4087–4091.

[9] T. N. Sainath and C. Parada, "Convolutional neural networks for small-footprint keyword spotting," in *Proc. INTERSPEECH*, 2015, pp. 1478–1482.

[10] S. Myer and V. S. Tomar, "Efficient keyword spotting using time delay neural networks," in *Proc. INTERSPEECH*, 2018, pp. 1264–1268.

[11] A. Coucke, M. Chlieh, T. Gisselbrecht, D. Leroy, M. Poumeyrol, and T. Lavril, "Efficient keyword spotting using dilated convolutions and gating," in *Proc. ICASSP*, 2019, pp. 6351–6355.

[12] M. Sun, A. Raju, G. Tucker, S. Panchapagesan, G. Fu, A. Mandal, S. Matsoukas, N. Strom, and S. Vitaladevuni, "Max-pooling loss training of long short-term memory networks for small-footprint keyword spotting," in *Proc. SLT*. IEEE, 2016, pp. 474–480.

[13] J. Hou, Y. Shi, M. Ostendorf, M.-Y. Hwang, and L. Xie, "Mining effective negative training samples for keyword spotting," in *Proc. ICASSP*, 2020, pp. 7444–7448.

[14] C. Shan, J. Zhang, Y. Wang, and L. Xie, "Attention-based end-to-end models for small-footprint keyword spotting," in *Proc. INTERSPEECH*, 2018, pp. 2037–2041.

[15] X. Wang, S. Sun, C. Shan, J. Hou, L. Xie, S. Li, and X. Lei, "Adversarial examples for improving end-to-end attention-based small-footprint keyword spotting," in *Proc. ICASSP*, 2019, pp. 6366–6370.

[16] Y. Bai, J. Yi, J. Tao, Z. Wen, Z. Tian, C. Zhao, and C. Fan, "A Time Delay Neural Network with Shared Weight Self-Attention for Small-Footprint Keyword Spotting," in *Proc. INTERSPEECH*, 2019, pp. 2190–2194.

[17] A. Graves, S. Fernández, F. J. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proc. ICML*, vol. 148, 2006, pp. 369–376.

[18] S. Fernández, A. Graves, and J. Schmidhuber, "An application of recurrent neural networks to discriminative keyword spotting," in *Proc. ICANN*, 2007, pp. 220–229.

[19] C. T. Lengerich and A. Y. Hannun, "An end-to-end architecture for keyword spotting and voice activity detection," *CoRR*, vol. abs/1611.09405, 2016.

[20] Z. Wang, X. Li, and J. Zhou, "Small-footprint keyword spotting using deep neural network and connectionist temporal classifier," *CoRR*, vol. abs/1709.03665, 2017.

[21] Y. Zhuang, X. Chang, Y. Qian, and K. Yu, "Unrestricted vocabulary keyword spotting using LSTM-CTC," in *Proc. INTERSPEECH*, 2016, pp. 938–942.

[22] D. Povey, V. Peddinti, D. Galvez, P. Ghahremani, V. Manohar, X. Na, Y. Wang, and S. Khudanpur, "Purely sequence-trained neural networks for asr based on lattice-free mmi," in *Proc. INTERSPEECH*, 2016, pp. 2751–2755.

[23] Z. Chen, Y. Qian, and K. Yu, "Sequence discriminative training for deep learning based acoustic keyword spotting," *Speech Communication*, vol. 102, pp. 100–111, 2018.

[24] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The kaldi speech recognition toolkit," in *Proc. ASRU*, 2011.

[25] H. Hadian, H. Sameti, D. Povey, and S. Khudanpur, "End-to-end speech recognition using lattice-free MMI," in *Proc. INTERSPEECH*, 2018, pp. 12–16.

[26] G. Chen, H. Xu, M. Wu, D. Povey, and S. Khudanpur, "Pronunciation and silence probability modeling for ASR," in *Proc. INTERSPEECH*, 2015, pp. 533–537.

[27] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[28] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *EMNLP*, 2014, pp. 1724–1734.

[29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. NeurIPS*, 2017, pp. 5998–6008.

[30] D. Povey, G. Cheng, Y. Wang, K. Li, H. Xu, M. Yarmohammadi, and S. Khudanpur, "Semi-orthogonal low-rank matrix factorization for deep neural networks," in *Proc. INTERSPEECH*, 2018, pp. 3743–3747.

[31] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *Proc. INTERSPEECH*, 2015, pp. 3214–3218.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. ICCV*, 2016, pp. 770–778.

[33] Y. Wang, D. Snyder, H. Xu, V. Manohar, P. S. Nidadavolu, D. Povey, and S. Khudanpur, "The JHU ASR System for VOiCES from a Distance Challenge 2019," in *Proc. INTERSPEECH*, 2019, pp. 2488–2492.

[34] D. Snyder, G. Chen, and D. Povey, "MUSAN: A music, speech, and noise corpus," *CoRR*, vol. abs/1510.08484, 2015.

[35] T. Ko, V. Peddinti, D. Povey, M. L. Seltzer, and S. Khudanpur, "A study on data augmentation of reverberant speech for robust speech recognition," in *Proc. ICASSP*. IEEE, 2017, pp. 5220–5224.

[36] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio augmentation for speech recognition," in *Proc. INTERSPEECH*, 2015, pp. 3586–3589.

[37] J. Hou, Y. Shi, M. Ostendorf, M. Hwang, and L. Xie, "Region proposal network based small-footprint keyword spotting," *IEEE Signal Process. Lett.*, vol. 26, no. 10, pp. 1471–1475, 2019.

[38] Y. Wang, T. Chen, H. Xu, S. Ding, H. Lv, Y. Shao, N. Peng, L. Xie, S. Watanabe, and S. Khudanpur, "Espresso: A fast end-to-end neural speech recognition toolkit," in *Proc. ASRU*, 2019.

[39] Y. Shao, Y. Wang, D. Povey, and S. Khudanpur, "PyChain: A fully parallized PyTorch implementation of LF-MMI for end-to-end ASR," *submitted to INTERSPEECH*, 2020.