



Fully End-to-end Streaming Open-vocabulary Keyword Spotting with W-CTC Forced Alignment

Dohyun Kim, Jiwook Hwang

PuzzleAI, South Korea

dohyunkim1657@gmail.com, phypan11@gmail.com

Abstract

In open-vocabulary keyword spotting, an acoustic encoder pre-trained with Connectionist Temporal Classification (CTC) loss is typically used to train a text encoder by aligning audio embedding space with text embedding space. In previous work, word-aligned datasets were created by forced alignment algorithms such as the Montreal Forced Aligner (MFA) to train text encoder and verifier models. In this paper, we propose a new training pipeline for open-vocabulary keyword spotting using the W-CTC forced alignment algorithm, a simple modification of the practical CTC algorithm. Our approach eliminates the need for creating word-aligned datasets, operates in a fully end-to-end manner, and demonstrates superior performance on the Libriphrase hard dataset.

Index Terms: CTC, W-CTC, forced alignment, end-to-end training, keyword spotting, open-vocabulary keyword spotting

1. Introduction

Open-vocabulary keyword spotting models aim to recognize any keyword in a speech dataset. Solutions for this task include few-shot learning, where spoken keywords are enrolled before inference, known as the query-by-example method [1], and zero-shot learning, where only the text sequences of keywords are utilized. In zero-shot systems, text encoders are typically trained to match the distribution of a pre-trained audio encoder with contrastive learning [2]. This paper focuses on the more universal approach of zero-shot open-vocabulary keyword spotting.

Algorithms such as Dynamic Sequence Partitioning (DSP) [3, 4] and attention mechanisms [5, 6] have been applied to open-vocabulary tasks, where the alignment of audio and keywords is crucial. However, DSP is only applicable to forced-aligned datasets, and attention-based models trained with full utterances degrade performance because extraneous context interferes with accurate attention as demonstrated later in Sec. 3.4.1. A recent approach [7] utilizes CTC alignment by introducing pad tokens in CTC training to find alignments. However, it is unclear how to apply this approach directly to full-script audio and the approach is limited to short phrase datasets. Additionally, it does not leverage practical CTC so it requires effort to implement. Therefore, phrase datasets created with the MFA tool [8] are mainly used for training open-vocabulary keyword spotting models. However, creating these datasets is time-consuming and memory-intensive, making it challenging for large datasets. To overcome these problems, we use a W-CTC forced alignment [9] that closely resembles the practical CTC algorithm, directly incorporating it into the training process for end-to-end training on a full-script audio dataset. Furthermore, we explore various training pipelines to

assess the potential for fully end-to-end training. To the best of our knowledge, we are the first to train open-vocabulary keyword spotting on a full-script dataset.

Streaming keyword spotting is essential for verifying that keywords are spotted in noisy or continuous speech in real-time. Previous approaches have relied on Automatic Speech Recognition (ASR) models either to detect keywords in decoded text [10] or to verify scores by constraining sequences with keywords [11]. However, these methods do not address open-vocabulary scenarios and tend to perform poorly on challenging cases. The attention-based verifier model [5] offers a potential solution by using a sliding window algorithm. Attention should be confined to aligned audio regions for optimal performance, but the windowing process typically includes extraneous elements. Therefore, we introduce a streaming open-vocabulary keyword spotting system that is compatible with our proposed model utilizing streaming W-CTC forced alignment.

The contributions of this paper are as follows: **1.** Propose a new open-vocabulary keyword spotting model highly compatible with the W-CTC framework that achieves high performance. **2.** Explore a fully end-to-end training pipeline for the proposed open-vocabulary keyword spotting model. **3.** Introduce a streaming open-vocabulary keyword spotting algorithm compatible with our proposed model.

2. Proposed Method

The proposed method focused on obtaining the functionality of streaming open-vocabulary keyword spotting using an end-to-end training pipeline. To enable streaming, we use a blockwise conformer audio encoder [12] that extracts audio embeddings in continuous speech in real-time. In addition, we use W-CTC forced alignment to find keyword alignments in continuous speech at every training time step for end-to-end training. An overview of our model is shown in Fig. 1.

2.1. Audio encoder

The Contextual Blockwise Conformer (CBC) [12] adapts Conformer [13] for streaming applications by processing overlapping blocks with a constant hop size and concatenating output. To minimize the performance gap with Conformer, CBC incorporates a contextual vector from the previous block and look-ahead frames for future context. Its structure mirrors the tiny Conformer from [4] with a small block size for real-time processing. Phoneme sequences generated via G2P [14] serve as input text. The CBC, trained with CTC Loss, produces audio embeddings, $\mathbf{o} = (o_1, o_2, \dots, o_T)$, mapped by the CTC layer to a token probability distribution, $\mathbf{e} = (e_1, e_2, \dots, e_T)$, where T is the length of the encoder output.

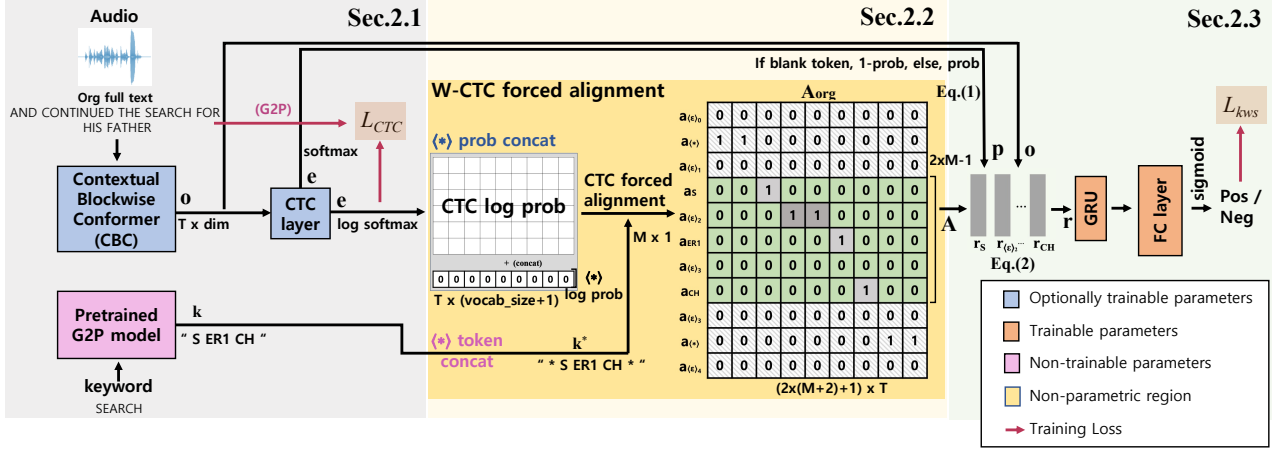


Figure 1: Overview of the proposed model training pipeline: The audio encoder(CBC) and CTC layer encode the audio, while G2P transforms the keyword into a phoneme sequence as described in Sec. 2.1. The W-CTC forced alignment uses the features and phoneme sequences to align the audio subset that best matches the phoneme sequence, leveraging the $\langle * \rangle$ token, as described in Sec. 2.2. The resulting alignment and audio features are used as input to the verifier (GRU + FC) to infer the keyword probability, as described in Sec. 2.3. L_{CTC} is used solely to train the audio encoder, whereas L_{kws} is mainly used to train the verifier model.

2.2. W-CTC forced alignment

While CTC loss collects all possible paths [15], CTC forced alignment finds the most probable path, which is implemented with Dynamic Programming [16]. Let us denote the input keyword as $\mathbf{k} = (k_1, k_2, \dots, k_M)$, where k_i is a token, and the result of the CTC forced alignment as $\mathbf{A}_{org} = (\mathbf{a}_{\langle \epsilon \rangle_0}, \mathbf{a}_{k_1}, \mathbf{a}_{\langle \epsilon \rangle_1}, \mathbf{a}_{k_2}, \dots, \mathbf{a}_{k_M}, \mathbf{a}_{\langle \epsilon \rangle_M})$, where $\mathbf{a}_n \in \{0, 1\}^T$. \mathbf{a}_n is a vector where the indices on the aligned path of token $n \in \{k_i, \langle \epsilon \rangle_i\}$ are set to 1, and 0 otherwise. The blank token, denoted as $\langle \epsilon \rangle$, is also aligned. \mathbf{a}_{k_i} can never be the zero vector because the phoneme token is aligned with at least one frame, but $\mathbf{a}_{\langle \epsilon \rangle_i}$ can be the zero vector when k_i and k_{i+1} are tightly aligned (See the yellow section in Fig. 1). Unlike standard CTC forced alignment, W-CTC forced alignment aligns partial sequences to spot keywords in continuous speech. In [9], the $\langle * \rangle$ token is inserted into unknown text sequences, masking alignment with $\langle * \rangle$ when there is no preference. The log probability of $\langle * \rangle$ is set to 0 and concatenated in every frame as in Fig. 1, ensuring only known sequences affect the score of the best path. In our W-CTC forced alignment, $\langle * \rangle$ are added at the start and end of \mathbf{k} , forming $\mathbf{k}^* = (\langle * \rangle, k_1, k_2, \dots, k_M, \langle * \rangle)$. The alignment process is identical to the practical CTC forced alignment algorithm, which is easy to implement. The algorithm is simple and efficient, with a time complexity of $\mathcal{O}(M \times T)$, outperforming the DSP algorithm's complexity of $\mathcal{O}(M \times T^2)$ [3]. Further optimizations can enhance its efficiency [16].

2.3. Verifier model

Our verifier model is composed of GRU and FC layers as described in [4], and its input is calculated with an alignment, \mathbf{A} , and token probabilities. $\mathbf{a}_{\langle \epsilon \rangle_i}$ are also included to effectively verify hard-negatives. This is because an abnormal $\langle \epsilon \rangle$ probability can signal a deletion error ($\mathbf{a}_{\langle \epsilon \rangle_i}$ includes indexes with non- $\langle \epsilon \rangle_i$ tokens). We solely use $\mathbf{A} = (\mathbf{a}_{k_1}, \mathbf{a}_{\langle \epsilon \rangle_1}, \mathbf{a}_{k_2}, \dots, \mathbf{a}_{k_M})$ as input with length $2M - 1$. We multiply the softmax score at each index according to the alignment. The scores of $\langle \epsilon \rangle$ are subtracted from 1 to maintain the distribution of the empty $\langle \epsilon \rangle$

alignment (a zero vector), as follows.

$$p_{(n,i)} = \begin{cases} 1 - \text{softmax}(e_i)_t & \text{if } n \text{ is } \langle \epsilon \rangle \\ \text{softmax}(e_i)_t & \text{otherwise} \end{cases} \quad (1)$$

\mathbf{a}_n is element-wise multiplied by $\mathbf{p}_n = (p_{(n,0)}, \dots, p_{(n,T)})$ along the time axis. Finally, audio embedding \mathbf{o} is multiplied by each frame index and averaged over the non-zero frames along the time axis, as follows:

$$r_n = ((\mathbf{a}_n \circ \mathbf{p}_n) \cdot \mathbf{o}) / \text{sum}(\mathbf{a}_n) \quad (2)$$

where \circ denotes element-wise multiplication and \cdot denotes matrix multiplication. The resulting matrix, $\mathbf{r} = (r_{k_1}, r_{\langle \epsilon \rangle_1}, r_{k_2}, \dots, r_{k_M})$, is fed to the verifier model.

2.4. Fully end-to-end training

Initially, for each utterance, positive phrases are randomly sampled N_{gen} times by choosing 1 to 4 consecutive words within the text label of the audio. Negative phrases are randomly sampled from the positive phrases of other utterances, excluding phrases that are part of the text label. In line with [4, 5], hard negative phrases are created from positive phrases by insertion, deletion and substitution of 1 to 3 successive phonemes. The substitution candidates are predefined with 5 acoustically similar phonemes, which is determined by the similarity of the CTC layer. As a result, a mini-batch contains $3 \times N_{gen}$ phrases per utterance.

As mentioned in Sec. 2.2, the resulting phrases undergo W-CTC forced alignment with full audio. The alignments are inferred on-the-fly and used for training with the verifier. To train the model in a fully end-to-end manner, we employ multitask learning.

$$L_{total} = L_{kws} + \lambda_{CTC} L_{CTC}, \quad (3)$$

where L_{CTC} denotes the CTC loss of full text input and L_{kws} refers to Binary Cross Entropy Loss calculated with keywords. λ_{CTC} is set to 1.0 for fully end-to-end learning and to 0.0 when training only the verifier with the pre-trained encoder.

The non-differentiability of CTC due to the gradient of the argmax function in alignment maps does not affect the model’s optimization or generalization. This is because the alignment is directly multiplied with the encoder outputs to ensure differentiability. Additionally, the impact of L_{kws} on the encoder is minimal, as the aligned path captures the features necessary for effective learning. Restricting L_{kws} gradients to the verifier model does not convey performance differences (AUC variation within 0.2), which confirms the robustness and effectiveness of end-to-end training with W-CTC alignment.

2.5. Streaming W-CTC keyword spotting inference

Algorithm. 1 outlines the streaming keyword spotting process, which utilizes a modified version of practical continuous CTC forced alignment. The first modification involves inserting the $\langle * \rangle$ token at the start of the W-CTC forced alignment to ensure that any arbitrary frame can serve as an alignment start point. The second involves backtracking from the index with the maximum score of the k_M token (ending of keyword, within window). This ensures that \mathbf{a}_{k_M} includes at least one frame containing the value 1 in the current window. Negative sample alignments often reveal abnormal probabilities for certain tokens, especially in the final sections of the alignment. They can also be completely misaligned, as shown in Fig. 2. These abnormal alignments, derived from nearby windows, with low scores, help verify the keyword within discrete audio windows. After the alignment is extracted, it is used in the verifier. The complexity of the streaming keyword spotting algorithm, $\mathcal{O}(M \times W)$ per W -sized window, matches that of practical CTC forced alignment. Thus, the algorithm is efficient because M and W are small enough for streaming.

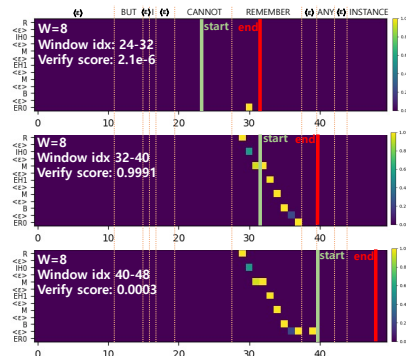


Figure 2: Alignment with window size 8 and target keyword “remember”. The first row is completely misaligned, the second row is well-aligned and the last row contains misaligned tokens with low probability. The verifier model assigns low scores to the misaligned cases.

3. Experimental Results

3.1. Datasets

Librispeech [17] train-clean-100 and 360 were used for training the CBC and end-to-end training. Our evaluation utilizes the Libriphrase dataset [6], derived from Librispeech train-other 500, comprising both the Libriphrase hard (LH) and Libriphrase easy (LE) datasets. In addition, a phrase dataset (PD) consisting of 1 to 4-words phrases from Librispeech train-clean-100 and 360 was employed for training the phrase model. The Speech

Algorithm 1 Streaming W-CTC Keyword Alignment

Input: $\mathbf{k} = (\langle * \rangle, k_1, \dots, k_M)$, $\mathbf{e} = (e_1, \dots, e_W)$, α, P
Output: $Align, \alpha, P$

- 1: **if** α and P not initialized **then**
- 2: $\alpha \in \mathbb{R}^{2(M+1)+1}$, $\alpha[0] = 0, \alpha[1] = 0, \alpha[2:] = -\inf$
- 3: \triangleright Forward score state (including $\langle * \rangle$ and $\langle \epsilon \rangle$)
- 4: $P \in \mathbb{R}^{t \times 2(M+1)}$, $P = []$ \triangleright Stores backtrack paths
- 5: **end if**
- 6: $s_{k_M} = []$ \triangleright Stores in current window
- 7: **for** $i \leftarrow 1$ **to** W **do**
- 8: $\alpha, P_{cur} \leftarrow CTC_{align}(\alpha, e_i, \mathbf{k})$ $\triangleright \mathcal{O}(M)$
- 9: $P.append(P_{cur})$
- 10: $s_{k_M}.append(\alpha[k_M])$ \triangleright Score of keyword ending
- 11: **end for**
- 12: $Align = backtrack(P, argmax(s_{k_M}))$ $\triangleright \mathcal{O}(M)$
- 13: **return** $Align, \alpha, P$

Commands V1 test (G) dataset [18] was also evaluated.

3.2. Training configurations

The acoustic features were 80-channel mel spectrograms (25ms window, 10ms hop, 16kHz sample rate). The CBC hyper-parameters were: layers: 6, enc-dim: 144, lin-dim: 576, conv-kernel: 3, att-heads: 4. For block processing, block/hop/look-ahead size were 20/8/8, respectively. The dimension of the GRU and FC layer in the verifier was 144. The total number of parameters was 3.61M (excluding the G2P model). For training, AdamW [19] was used with 5000 warm-up steps and a 0.0015 learning rate. The audio encoder was trained with CTC for 150 epochs on Librispeech and for 10 epochs of fine-tuning on the PD. With the pre-trained encoder, the verifier model was trained for 50 epochs on full utterances (Librispeech) and for 20 epochs on the PD. The fully end-to-end model was trained for 150 epochs. N_{gen} was set to 10 for keyword sampling. All source code was based on ESPNet [20].

3.3. Baseline Comparisons

Our evaluation was conducted using the AUC and EER metrics. For training and evaluation on the phrase datasets (PD, LE, LH), we padded a frame of low probability embedding, excluding the $\langle * \rangle$ token, at the start and end for W-CTC alignment. This prevented tightly aligned phrases from being adversely affected by the aligned index of $\langle * \rangle$. We utilized results from previous studies for the comparison study. For [5], we utilized the AdaKWS-Tiny model which has a similar model size. For [7], we used the phrase level cosine similarity value, the best result of the paper. Additionally, for fairness, we froze the CBC and trained the previous [4, 5, 7] and proposed models with the PD. The CBC model was initially trained with the Librispeech dataset and fine-tuned with the PD using CTC.

As shown in Table. 1, the proposed method achieves the best results on the LH dataset. On LE and G, high performance is achieved, similar to previous findings. Thus, using CTC alignment leads to more accurate alignment and our proposed verifier model improves performance on challenging cases.

3.4. Ablation Study

3.4.1. End-to-End Training

In this section, we explore the role of the verifier and the feasibility of end-to-end training. The base encoder model trained

Table 1: Baseline comparison. The first row shows results from previous studies. The second row refers to trained previous and proposed verifier models with the fixed CBC encoder.

Method	AUC			EER		
	LH	G	LE	LH	G	LE
[3]	73.58	81.06	96.7	32.9	27.25	8.42
[4]	92.7	93.94	99.84	14.4	13.45	1.7
[5]	93.75	-	99.80	13.47	-	1.61
[7]	77.10	-	98.32	29.63	-	6.06
CBC+[4]	88.26	92.79	99.69	19.58	14.92	2.48
CBC+[5]	92.42	98.94	99.73	14.10	4.23	1.27
CBC+[7]	75.16	91.82	95.32	31.27	16.67	8.99
CBC+Ours	95.93	98.89	99.95	10.21	4.64	0.91

Table 2: Results of an ablation study on end-to-end training. Detailed methodology is given in Sec. 3.4.1

Method	AUC		EER	
	LH	LE	LH	LE
(1)	66.11	91.36	38.70	15.03
(2)	78.88	99.12	28.19	4.18
(3)	94.71	99.93	11.19	0.97
(4)	95.93	99.95	10.21	0.91
(5)	94.66	99.92	11.47	1.04

with Librispeech is denoted as CBC_{base} . The other methods are described as follows: (1) CBC_{base} + verifying with the CTC score without using the verifier, (2) CBC_{base} + training AdaKWS [5] with Librispeech, (3) CBC_{base} + training the proposed verifier with Librispeech, (4) CBC_{base} fine-tuned with the PD + training the proposed verifier with the PD, (5) multi-task learning ($\lambda_{ctc} = 1.0$ in Eq. 3) without CBC_{base} (i.e. fully end-to-end training).

In previous studies [11, 21], only ASR models were used for verification (no verifier was used). However, open-vocabulary tasks have highlighted the essential role of verifier models. For the AdaKWS [5] model, training with full utterances was unsuccessful for difficult cases, necessitating the creation of the phrase dataset. This suggests that the model should solely consider aligned regions for optimal performance. However, our model trained with full utterances achieves comparable results to those trained with the phrase dataset. Additionally, the good performance of multitask learning demonstrates the feasibility of fully end-to-end learning.

3.4.2. W-CTC forced alignment

W-CTC’s forced alignment often compressed keyword boundaries due to the high log probability of the $\langle * \rangle$ token. Thus, we capped the log probability of $\langle * \rangle$ at -0.5 when the log probability of $\langle \epsilon \rangle$ in the same frame was below -0.5. This threshold was determined by examining the log probability distribution of forced alignment of CTC outputs, where approximately 95% of phoneme log probabilities were less than -0.5. This adjustment resulted in more realistic boundary predictions. We compared this method with the approach in [9] (where $\langle * \rangle$ was assigned a log probability of zero) by quantifying the differences between W-CTC (which include $\langle * \rangle$ and target labels only) and full-script CTC alignments. This method achieved an average frame difference of 0.17, which is substantially lower than that achieved by the original method, 0.57. However, the adjustment

Table 3: Results of ablation study on W-CTC modification in Sec. 3.4.2.

Method	AUC		EER	
	LH	LE	LH	LE
W-CTC	94.66	99.92	11.47	1.04
W-CTC _{mod}	94.83	99.92	11.05	1.06

Table 4: Results of ablation study on the various ways of processing $\langle \epsilon \rangle$ tokens stated in Sec. 3.4.3.

Method	AUC			EER		
	LH	LE	DEL	LH	LE	DEL
Non- $\langle \epsilon \rangle$	96.02	99.95	97.57	10.03	0.87	7.37
Avg	94.78	99.93	98.44	11.96	1.03	6.01
Concat	95.98	99.95	98.17	10.22	0.79	5.51
Proposed	95.93	99.95	98.87	10.21	0.91	4.58

did not result in a critical performance gap, as shown in Table. 3, leading us to conclude that compressed boundaries do not affect performance.

3.4.3. Dealing with $\langle \epsilon \rangle$ tokens

In previous studies, $\langle \epsilon \rangle$ frame embeddings were typically averaged with consecutive non- $\langle \epsilon \rangle$ token frame embeddings. To analyze the role of $\langle \epsilon \rangle$ probabilities in the verifier model, we explored three transformations of the alignment: **1. Non- $\langle \epsilon \rangle$:** Removed $\langle \epsilon \rangle$ token embeddings from the resulting matrix in Eq.(2), represented as $\mathbf{r}_{non-\langle \epsilon \rangle} = (r_{k_1}, \dots, r_{k_M})$. **2. Avg:** Averaged the non- $\langle \epsilon \rangle$ token embeddings with the following $\langle \epsilon \rangle$ token embeddings when calculating the resulting matrix, represented as $\mathbf{r}_{avg} = (r_{k_1+\langle \epsilon \rangle_1}, \dots, r_{k_M+\langle \epsilon \rangle_M})$. In this case, only the *softmax* value in Eq.(1) was used, not $(1 - softmax)$. **3. Concat:** Concatenated the non- $\langle \epsilon \rangle$ token embeddings with the following $\langle \epsilon \rangle$ alignments in the resulting matrix, represented as $\mathbf{r}_{con} = ((r_{k_1}, r_{\langle \epsilon \rangle_1}), \dots, (r_{k_M}, r_{\langle \epsilon \rangle_M}))$ (the dimension of the verifier model was doubled). The models were trained on the PD dataset and evaluated on the LE, LH and DEL datasets. In DEL, an intermediate word is removed from the 3 to 4-words Libriphrase dataset to simulate deletion errors.

The results in Table. 4 show that the averaging method, commonly used in previous studies underperformed on the LH dataset. On the other hand, on the DEL dataset, the proposed method showed the best performance, indicating that discrete $\langle \epsilon \rangle$ scoring assists in identifying deletion errors. Interestingly, the removal of entire $\langle \epsilon \rangle$ embeddings also resulted in high performance across all three datasets, suggesting that non- $\langle \epsilon \rangle$ token embeddings contain consecutive token probabilities that are robust to errors.

4. Conclusion

In this paper, we have proposed a fully end-to-end open-vocabulary keyword spotting model that uses W-CTC forced alignment. Our model structure not only achieves high performance on verifying hard keyword pairs but can also be trained in a fully end-to-end manner. A performance evaluation of diverse algorithms and showed that our proposed method is robust to hard negative cases. We have also proposed a streaming keyword spotting algorithm using W-CTC forced alignment that can continuously process an alignment, making it suitable for spotting keywords in continuous speech in real-time.

5. References

- [1] J. Jung, Y. Kim, J. Park, Y. Lim, B.-Y. Kim, Y. Jang, and J. S. Chung, "Metric learning for user-defined keyword spotting," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.
- [2] J. Zhu, F. Samir, C. Yang, and J. Islam, "Open-vocabulary keyword spotting in any language through multilingual contrastive speech-phoneme pretraining," *arXiv preprint arXiv:2311.08323*, 2023.
- [3] K. Nishu, M. Cho, and D. Naik, "Matching latent encoding for audio-text based keyword spotting," *arXiv preprint arXiv:2306.05245*, 2023.
- [4] K. Nishu, M. Cho, P. Dixon, and D. Naik, "Flexible keyword spotting based on homogeneous audio-text embedding," in *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024, pp. 5050–5054.
- [5] A. Navon, A. Shamsian, N. Glazer, G. Hetz, and J. Keshet, "Open-vocabulary keyword-spotting with adaptive instance normalization," in *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024, pp. 11 656–11 660.
- [6] H.-K. Shin, H. Han, D. Kim, S.-W. Chung, and H.-G. Kang, "Learning audio-text agreement for open-vocabulary keyword spotting," *arXiv preprint arXiv:2206.15400*, 2022.
- [7] S. Jin, Y. Jung, S. Lee, J. Roh, C. Han, and H. Cho, "Ctc-aligned audio-text embedding for streaming open-vocabulary keyword spotting," *arXiv preprint arXiv:2406.07923*, 2024.
- [8] M. McAuliffe, M. Socolof, S. Mihuc, M. Wagner, and M. Sonderegger, "Montreal forced aligner: Trainable text-speech alignment using kaldi." in *Interspeech*, vol. 2017, 2017, pp. 498–502.
- [9] X. Cai, J. Yuan, Y. Bian, G. Xun, J. Huang, and K. Church, "W-ctc: a connectionist temporal classification loss with wild cards," in *International Conference on Learning Representations*, 2021.
- [10] Y. He, R. Prabhavalkar, K. Rao, W. Li, A. Bakhtin, and I. McGraw, "Streaming small-footprint keyword spotting using sequence-to-sequence models," in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2017, pp. 474–481.
- [11] Y. Xi, H. Li, B. Yang, H. Li, H. Xu, and K. Yu, "Tdt-kws: Fast and accurate keyword spotting using token-and-duration transducer," in *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024, pp. 11 351–11 355.
- [12] E. Tsunoo, Y. Kashiwagi, T. Kumakura, and S. Watanabe, "Transformer asr with contextual block processing," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2019, pp. 427–433.
- [13] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu *et al.*, "Conformer: Convolution-augmented transformer for speech recognition," *arXiv preprint arXiv:2005.08100*, 2020.
- [14] J. Park, Kyubyong Kim, "g2pe," <https://github.com/Kyubyong/g2p>, 2019.
- [15] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 369–376.
- [16] V. Pratap, A. Tjandra, B. Shi, P. Tomasello, A. Babu, S. Kundu, A. Elkahky, Z. Ni, A. Vyas, M. Fazel-Zarandi *et al.*, "Scaling speech technology to 1,000+ languages," *Journal of Machine Learning Research*, vol. 25, no. 97, pp. 1–52, 2024.
- [17] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.
- [18] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *arXiv preprint arXiv:1804.03209*, 2018.
- [19] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.
- [20] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. E. Y. Soplin, J. Heymann, M. Wiesner, N. Chen *et al.*, "Espnet: End-to-end speech processing toolkit," *arXiv preprint arXiv:1804.00015*, 2018.
- [21] L. Lugosch, S. Myer, and V. S. Tomar, "Donut: Ctc-based query-by-example keyword spotting," *arXiv preprint arXiv:1811.10736*, 2018.