INTERSPEECH 2023
20-24 August 2023, Dublin, Ireland

Fast Enrollable Streaming Keyword Spotting System: Training and Inference using a Web Browser

Namhyun Cho, Sunmin Kim*, Yoseb Kang*, Heeman Kim*

Speech AI Lab., NCSOFT Corporation, South Korea

{cnh2769, essems79}@ncsoft.com, kjoe@sogang.ac.kr, pathos@ncsoft.com

Abstract

When a keyword spotting system is deployed on heavily personalized platforms such as digital humans, a few issues occur such as 1) a lack of training data when registering user-defined keywords, 2) a desire to reduce computation and minimize latency, and 3) the inability to immediately train and test the keyword-spotting model. We address the issues through 1) a keyword-spotting system based on a speech embedding model, 2) streamable system with duplicate computations removed, and 3) real-time inference in a web browser using WebAssembly.

Index Terms: Enrollable keyword spotting, Streamable model, Speech embedding model, WebAssembly, Low latency

1. Introduction

Digital human modeling is rapidly growing, and it can revolutionize many industries [1]. Personalized digital humans require the user-defined keywords as name-like calls for smooth interaction with humans. Therefore, an enrollable Keyword spotting (KWS) system is required. However, when such systems are operated on a personal device, the following issues occur: 1) Insufficient training data arises in creating a KWS model that can recognize each digital human by their unique name-like and user-defined keyword. 2) Generating a lightweight keyword-spotting model that can operate in real-time even under heavy load conditions, such as 3D rendering, speech recognition, and speech synthesis. 3) Difficulty in immediately training and testing the proper functionality of the KWS model.

To address these issues, this study proposes a fast enrollable streaming keyword spotting (FES-KWS) system utilizing speech embedding, which can be immediately tested via web browsers using WebAssembly (Wasm).

2. Related works

Keyword spotting has become widely known through voice-activated commands such as "Hey, Siri" and "OK, Google" [2]. To provide consistent performance for all users, these models use a large number of collected voice samples as training data [3]. This approach is not appropriate for implementing a user-defined keyword spotting system. Therefore, a few or zero-shot models have been developed to learn user-defined keywords by using high-level speech features through a pretrained embedding model [4, 5]. Despite the fact that high-performance pretrained models use transformers, building a streamable KWS model is challenging due to their fixed structure [6]. In this study, we introduced the process of converting a publicly available CNN-based speech embedding model into a streamable model and fast-lightweighted using TensorFlow Lite (TFLite). Additionally, we demonstrated a system that enables instant training and inference of the FES-KWS model in a web browser via Wasm.

* These authors contributed equally to this work.

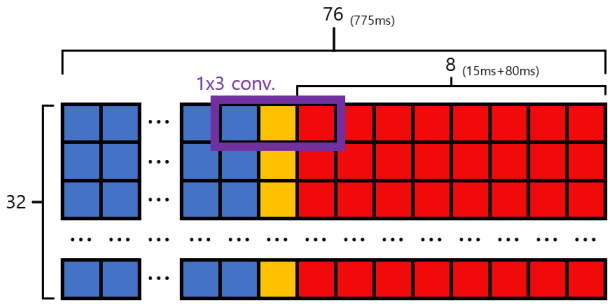

Figure 1: *Input layer of a streamable speech embedding model.*

3. Proposed Method

In this section, we describe the process of building an FES-KWS model with a speech embedding model. Additionally, we introduce the method of training and running the model using a web browser.

3.1. Building an FES-KWS model

Lin et al. proposed a one dimensional (1D) CNN-based speech embedding model,[1] which generates a speech embedding vector (SEV) from an initial 775 ms of speech, and subsequently generates additional vectors at 80 ms intervals with batches of input and output [4]. The disadvantage of obtaining a SEV every 80 ms for streaming audio input is that redundant calculations occur for approximately 695 ms (775 ms–80 ms). Therefore, we rebuild the model via Keras to store these redundant calculations into memory and reuse them for the next SEV calculation.

Figure 1 shows the structure of the input feature and provides information for removing redundant computations. The input feature of this model is a mel-filter bank with a window size of 400 ms, a shift window of 160 ms, and 32 mel bins. The blue area represents the part where the existing computation results can be reused when calculating the SEV for the next frame, which can be obtained after 80 ms. The yellow area represents the region that is previously calculated with zero padding in a previous frame, depending on the size of a convolution filter.

[1] Detailed information is in "https://tfhub.dev/google/speech embedding/1" using TensorFlow's "import pb to tensorboard.py."

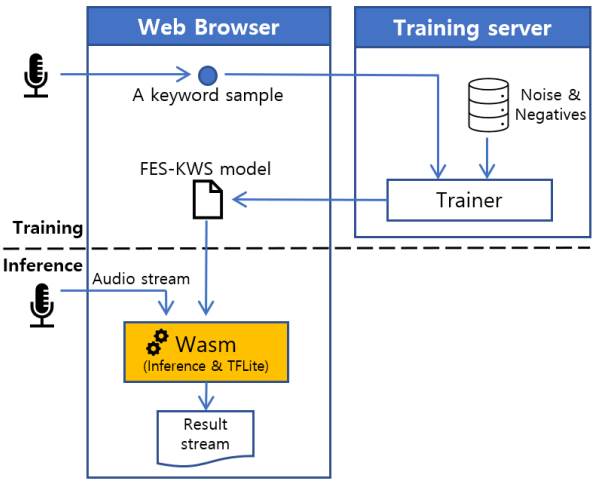

Figure 2: *Workflow of FES-KWS training and inference.*

Therefore, the calculation should be revised. Finally, the red area that represents the mel-feature obtained by concatenating the last 15 ms of the previous frame's audio with the newly inputted 80 ms, should be computed.

To eliminate redundant computations, the model has been modified to return blue regions for each layer. For the next frame, we generated SEV by concatenating the reusable regions returned after calculating only the yellow and red regions that require recomputation. This embedding model can reduce computational cost by employing only 1×3 or 3×1 convolution filters up to the final layer. By attaching a small head model to the restructured streamable speech embedding model as aforementioned, the FES-KWS model can be trained with only one speech sample of user-defined keywords [4].

3.2. Instant model training and inference via a web browser

In this section, we describe how to train and inference the FES-KWS model instantly in a web browser with the rebuild model.

By utilizing the AudioWorklet of the Web Audio API, an audio input received through a microphone in a web browser can be accessed. Using this, we can receive a speech sample of user-defined keywords through a microphone and start the learning process by sending it to a training server. The received positive samples, randomly extracts negative samples from speech corpora, and approximately 200 noise clips are used as training data. The model is trained with noise augmentation from SNR 20 to 0 dB, speed augmentation ranging 0.8–1.2x, and pitch shift ranging -2–2 semi-tones. The training time is within 10 s on CPU. The completed FES-KWS model is transmitted to a web browser and used as an inference model.

To perform inference on TFLite models in a web browser, a TFLite library built with Wasm is required. Wasm is a compact and efficient binary package that can be executed in web browsers. With the Emscripten toolchain to build FES-KWS inference code written in C/C++ and TFLite static library into Wasm, FES-KWS can be run in a web browser. By passing the audio obtained to the FES-KWS AudioWorklet, streaming inference can be performed on an audio stream.

4. Results

The method reduced redundant calculations for approximately 90 % of overlapping areas between adjacent frames, resulting in a 33.4% increase in processing speed on CPU. Additionally, the Keras model can be converted to a TFLite model, which was not possible with the original "pb" model from TensorFlow Hub, and it obtained an additional approximately 340x speed-up.

Figure 2 shows a flowchart of the aforementioned training and inference methods. During the training phase, a speech sample for a user-defined keyword was inputted through a web browser, and the FES-KWS model was generated on a training server using the method described in Chapter 3.1. During the inference phase, the presence of the user-defined keyword in an audio stream was determined using the model with the Wasm package, described in Chapter 3.2.

For performance evaluation, we used a self-made database consisting of recordings of "NC–ya" (hey, NC) keyword from 200 people, each six times. One of the audio file was used as a training phase, while the others were used as test sets. In cases where the speaker is consistent, the recognition success rate was measured at 99.3 % in an SNR 20 dB environment, while in the cases where the speaker was different, it was measured at 93.1 %. Although the speech embedding model was trained on English, it works reasonably well in Korean as well. The synthetic speech of 112 virtual speakers was used as training data to improve performance with different speakers. However, the performance slightly improved to 94.1 %.

5. Future works

To address false alarms, effective strategies are required by similar pronunciation speech to user-defined keywords. We explored speech clips with similar pronunciation in natural language and used them as negative samples for training data. Additionally, we reduced the number of speech samples required for the learning process. However, requiring input through a microphone is still inconvenient. Our goal is to research zero-shot KWS to address these issues.

6. References

[1] H. O. Demirel, S. Ahmed, and V. G. Duffy, "Digital human modeling: a review and reappraisal of origins, present, and expected future methods for representing humans computationally," *International Journal of Human–Computer Interaction*, vol. 38, no. 10, pp. 897–937, 2022.

[2] L. Burbach, P. Halbach, N. Plettenberg, J. Nakayama, M. Ziefle, and A. C. Valdez, "" hey, siri"," ok, google"," alexa". acceptance-relevant factors of virtual voice-assistants," in *2019 IEEE International Professional Communication Conference (ProComm)*. IEEE, 2019, pp. 101–111.

[3] B. Kim, M. Lee, J. Lee, Y. Kim, and K. Hwang, "Query-by-example on-device keyword spotting," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2019, pp. 532–538.

[4] J. Lin, K. Kilgour, D. Roblek, and M. Sharifi, "Training keyword spotters with limited and synthesized speech data," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7474–7478.

[5] H.-K. Shin, H. Han, D. Kim, S.-W. Chung, and H.-G. Kang, "Learning Audio-Text Agreement for Open-vocabulary Keyword Spotting," in *Proc. Interspeech 2022*, 2022, pp. 1871–1875.

[6] A. Berg, M. O'Connor, and M. T. Cruz, "Keyword Transformer: A Self-Attention Model for Keyword Spotting," in *Proc. Interspeech 2021*, 2021, pp. 4249–4253.