Auto-KWS 2021 Challenge: Task, Datasets, and Baselines

Jingsong Wang, Yuxuan He, Chunyu Zhao, Qijie Shao, Wei-Wei Tu, Tom Ko, Hung-yi Lee

14Paradigm Inc., China
2ASLP@NPU, Northwestern Polytechnical University, China
3ChaLearn, USA
4Department of Computer Science and Engineering
Southern University of Science and Technology, China
5College of Electrical Engineering and Computer Science, National Taiwan University

Abstract

Auto-KWS 2021 challenge calls for automated machine learning (AutoML) solutions to automate the process of applying machine learning to a customized keyword spotting task. Compared with other keyword spotting tasks, Auto-KWS challenge has the following three characteristics: 1) The challenge focuses on the problem of customized keyword spotting, where the target device can only be awakened by an enrolled speaker with his/her specified keyword. The user can use any language and accent to define his keyword. 2) All data of the challenge is recorded in realistic environment to simulate different user scenarios. 3) Auto-KWS is a “code competition”, where participants need to submit AutoML solutions, then the platform automatically runs the enrollment and prediction steps with the submitted code. This challenge aims at promoting the development of a more personalized and flexible keyword spotting system. Two baseline systems are provided to all participants as references.

Index Terms: keyword spotting, query by example, automated machine learning, automated deep learning, meta-learning, Auto-KWS, AutoSpeech

1. Introduction

Recently, the keyword spotting (KWS) technology [1, 2, 3, 4] has started emerging in people’s daily life through voice assistants, smart speakers and in-vehicle devices. Well-known examples of KWS applications include trigger word detection such as “OK Google” in Google Home and “Hey Siri” in Apple’s Siri. People can wake up the device by speaking the predefined keyword. Meanwhile, the KWS task is extended to consider the security. Personalized wake-up mode, including customized wake-up word detection and specific voiceprint verification, has created more application scenarios. The Auto-KWS challenge, as a new chapter of the AutoSpeech challenge series, is an automated machine learning challenge for customized keyword spotting, specifically designed to address this difficult problem. In this challenge, participants need to develop a computer program to solve the customized keyword spotting problem autonomously (without any human intervention).

The Auto-KWS challenge simulates the scenario where a device can be customized for its wake-up condition. For a new enrollment, the user selects a keyword and reads it repeatedly. The wake-up device records the content and speaker voiceprint of these sound clips. In the later stage, the device determines whether it is woken up by comparing the input voice with the enrollment information. Some conventional wake-up devices also support multiple keywords or voiceprint authentication. However, wake-up word detection and speaker verification (SV) are carried out separately in the pipeline, where a wake-up word detection system is used to generate a successful trigger, followed by a speaker verification system used to perform identity authentication [5]. Besides that, both the keyword spotting and voiceprint verification are text-dependent, limiting the flexibility to customise wake-up words.

Machine learning, especially deep learning, has brought great improvements to the field of speech technology related to this challenge. Deep neural networks (DNN) based KWS have gradually replaced conventional approaches. More complex network structures have been adopted to promote the performance of KWS systems [6, 3, 7]. Moreover, query by example (QbE) methods, which compare the test audio segment against the templates to make detection decisions, can improve the KWS performance [8, 9, 10]. Besides, DNNs have also shown great performance on speaker verification tasks [11, 12].

There are some prior works on combining keyword detection and speaker verification [5, 13]. Their approaches combine two independently trained sub-systems and present a joint end-to-end neural network system. In spite of the core techniques of the two individual tasks, a joint system can be benefited from the following techniques. Audio Word2vec [14] extracted fixed-dimension segmental representation for query-by-example. [15, 16, 17] demonstrated a sequence of extracted representations including phonetic, speaker and emotion characteristics. [18, 19, 20, 21] showed that semi-supervised learning can establish new principles to solve ASR problem.

The solution to this challenge may contain a complex pipeline. Automatic machine learning (AutoML) related technologies or methods can help build a solution system which has good performance and high computational efficiency [22], such as proposing more effective representation for enrollment utterances of each speaker, or a more robust and effective feature-based method than the conventional DTW method [23]. In low resource situations, meta-learning or few shot learning have the ability to fully utilize limited data and improve the performance of the KWS or SV model under difficult conditions [24, 25, 26, 12].

The Auto-KWS Challenge is the third in a series of au-

*corresponding author

http://dx.doi.org/10.21437/Interspeech.2021-817
tomated speech challenges\textsuperscript{12}, which applies AutoML to the tasks in speech processing [27]. Unlike many other challenges [5, 28], we require code submission instead of prediction or score submission. Participants’ code will be automatically run on multiple datasets on competition servers provided by the organizer with the same hardwares (CPU, GPU, RAM, Etc.) in order to have truly fair comparisons. As the test datasets on the platform are unseen by the participants, we provide the practice data in advance, in order to facilitate the participants’ offline debugging and model tuning. The submitted code should complete the enrollment, prediction and other processes within a specified time budget. The platform will calculate a score for ranking according to the predictions on the test datasets.

The paper is organized as follows: Section 2 describes the design of the competition, including competition protocol, metrics, datasets and starting kit. Section 3 describes two baselines we use and the results of the experiments. Section 4 presents the conclusions.

2. Competition Design

2.1. Competition protocol

The Auto-KWS 2021 competition has 3 phases: feedback phase, check phase, and final phase. Before the feedback phase, participants are provided with the training and practice datasets to develop their solutions offline. During the feedback phase, participants can upload their solutions to the platform to receive immediate performance feedback for validation. Then in the check phase, the participants can submit their code only once to make sure their code works properly on the platform. The platform will indicate success or failure to the participants, but detailed logs will be hidden. Lastly, in the final phase, participants’ solutions will be evaluated on the private dataset. Once the participants submit their code, the platform will run their algorithms automatically to test on the private data with time budget. The final ranking will be generated based on the scores calculated in this phase.

The platform exploits the same evaluation logic in all phases, shown in Figure 1. The task is defined by:

\[ T = (D_e, D_t, L, B_T, B_S) \]

where \( D_e \) and \( D_t \) represent examples in the enrollment data and test data without labels respectively, \( L : Y \times Y \rightarrow \mathbb{R} \) is a loss function measuring the losses of predictions \( y \) with respect to the true labels \( y \). \( B_S \) is the space budget and \( B_T \) contains the time budget for initialization, enrollment and prediction programs.

The initialization time budget is 30 minutes, the enrollment time budget is 5 minutes for each speaker, and the test time budget is Real Time Factor (RTF) \( F_r \) times the total duration of the test audio. \( F_r \) is calculated as,

\[ F_r = T_{\text{process}} / T_{\text{data}} \]

where \( T_{\text{process}} \) is the total processing time of all test data, and \( T_{\text{data}} \) is the total duration of the test audio. Notice that \( T_{\text{process}} \) only includes the inference time for each test audio since the enrollment and initialization processes have already been completed. In each task, after initialization, the platform will call the ”enrollment.sh” script, which runs for 5 minutes and then call the

Figure 1: Auto-KWS evaluation process for one task. The ingestion program will look for “initialize.sh”, “enrollment.sh” and “predict.sh” from participant’s submitted code folder and generate prediction results against the hidden test data. The prediction output will be passed into the scoring program to compare against the true label. The ingestion program contains the time and space budget \( B_T, B_S \) for initialization, enrollment and prediction processes respectively.

\textsuperscript{12}https://www.4paradigm.com/competition/autospeech2021

\textsuperscript{13}https://www.4paradigm.com/competition/autospeech2020

Table 1: Language/dialect summary. The contributors to voices in the dataset are randomly selected. They come from many different regions of China. Therefore, our dataset’s accents are diverse, and one quarter of them are dialects, including Cantonese, Szechwanese and many others.

<table>
<thead>
<tr>
<th>Language</th>
<th>Dialect</th>
<th>Number of Contributors</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>-</td>
<td>39</td>
</tr>
<tr>
<td>Chinese</td>
<td>Mandarin</td>
<td>121</td>
</tr>
<tr>
<td></td>
<td>Others</td>
<td>46</td>
</tr>
</tbody>
</table>
Table 2: The summary of datasets illustrates the data distribution in each subset. Each subset corresponds to a specific competition phase. Over half of the data are released to allow the participants to fine-tune their models better.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Speaker Number</th>
<th>Phase</th>
<th>Enrollment Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>100</td>
<td>Before feedback</td>
<td>10</td>
</tr>
<tr>
<td>Practice</td>
<td>20</td>
<td>Feedback</td>
<td>5</td>
</tr>
<tr>
<td>Feedback</td>
<td>40</td>
<td>Final</td>
<td>5</td>
</tr>
</tbody>
</table>

Test data in practice, feedback and private phases have the same format. For each speaker, only samples containing the wake-up word from the speaker are positive (wake-up word only at the end of audio). All other samples, such as other utterances from the speaker, the wake-up word recorded by other speakers, and any other audio that does not contain the wake-up word recorded by the speaker, are negative. In order to increase the number and diversity of test data, we have expanded the test data from the following directions:

- Splicing two pieces of audio
- Adding the MUSAN noises [29], the signal-to-noiseratio(SNR) was set between 5dB to 25dB
- Adding the RIRs-NOISES [30], the mixture weight was 0.5
- Perturbing the volume, the perturbation scale was set between 0.5 to 2

2.3. Starting kit

We provide the participants with a starting kit1, which contains practice datasets, submission sample code, two baseline, and the ingestion and scoring code that has the similar call logic with the online challenge platform. Participants can create their own code submission by just modifying the dir “code_submission”, which contains “initialize.sh”, “enrollment.sh”, and “predict.sh”, or adding other dependency code files including pretrained models. Then they can upload the zip-package of the submission folder.

It is very convenient to test and debug locally with the same handing programs and Docker image2 of the Challenge platform. Besides, evaluation with practice datasets is also easy.

Starting kit can be run in both CPU and GPU environment, but two baselines need GPU environment and the version of cuda cannot be lower than 10. Participants can check the python version and install python packages in the docker.

3. Baseline and Experiments

3.1. Baseline method 1

As shown in Fig. 2, baseline method 1 consists of a query by example (QbE) module and a speaker verification (SV) module. The alarm utterances will be sent to SV module to re-verify.

![System framework for baseline method 1](image)

Starting kit can be run in both CPU and GPU environment, but two baselines need GPU environment and the version of cuda cannot be lower than 10. Participants can check the python version and install python packages in the docker.

Table 3: The network config of the BNF extractor

<table>
<thead>
<tr>
<th>Layer</th>
<th>Input Context Frame</th>
<th>Output Dim</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDA</td>
<td>-2, -1, 0, 1, 2</td>
<td>5 * 40</td>
</tr>
<tr>
<td>TDNN1</td>
<td>0</td>
<td>850</td>
</tr>
<tr>
<td>TDNN2</td>
<td>-1, 0, 2</td>
<td>850</td>
</tr>
<tr>
<td>TDNN3</td>
<td>-3, 0, 3</td>
<td>850</td>
</tr>
<tr>
<td>TDNN4</td>
<td>-7, 0, 2</td>
<td>850</td>
</tr>
<tr>
<td>TDNN5</td>
<td>-3, 0, 3</td>
<td>850</td>
</tr>
<tr>
<td>BNF</td>
<td>0</td>
<td>40</td>
</tr>
<tr>
<td>output</td>
<td>0</td>
<td>targets number</td>
</tr>
</tbody>
</table>

We use Kaldi tools [31] to train the bottleneck feature (BNF) extractor model in the QbE module. The BNF extractor is a special time-delay neural network (TDNN) automatic speech recognition (ASR) acoustic model. We use 40-dim MFCC features as input. The model structure is shown in Table 3. It refers to `kaldi/egs/aishell2/s5/local/nnet3/rnn.jdkn.sh`, and the difference is that we add a 40-dim affine-layer (we call

1https://github.com/janson9192/Auto-KWS2021
2https://hub.docker.com/r/janson91/Auto-KWS2021
Table 4: Baseline Results on feedback and private datasets in Auto-KWS 2021. Datasets in feedback phase and final phase contains 20 speakers and 40 speakers respectively. Average Score, Average Miss Rate and Average FA Rate are calculated based on the baselines’ predictions of all utterances. Compute time represents the running time of the whole program, which contains initialization, enrollment and prediction processes.

<table>
<thead>
<tr>
<th>Phase</th>
<th>Baseline</th>
<th>Average Score</th>
<th>Average Miss Rate</th>
<th>Average FA Rate</th>
<th>Compute Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feedback</td>
<td>Baseline 1</td>
<td>0.859</td>
<td>0.443</td>
<td>0.046</td>
<td>0:54:07</td>
</tr>
<tr>
<td></td>
<td>Baseline 2</td>
<td>1.695</td>
<td>0.481</td>
<td>0.135</td>
<td>1:24:07</td>
</tr>
<tr>
<td>Final</td>
<td>Baseline 1</td>
<td>0.742</td>
<td>0.531</td>
<td>0.023</td>
<td>1:57:15</td>
</tr>
<tr>
<td></td>
<td>Baseline 2</td>
<td>1.086</td>
<td>0.691</td>
<td>0.044</td>
<td>3:01:32</td>
</tr>
</tbody>
</table>

Table 5: The data augment config of the SV model

<table>
<thead>
<tr>
<th>Augment Type</th>
<th>Detail</th>
<th>Times</th>
</tr>
</thead>
<tbody>
<tr>
<td>reverberation</td>
<td>small room (1m to 10m)</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>medium room (10m to 30m)</td>
<td>0.5</td>
</tr>
<tr>
<td>noise</td>
<td>MUSAN noise SNR: 15, 10, 5</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>MUSAN music SNR: 20, 15, 10</td>
<td>0.5</td>
</tr>
<tr>
<td>origin</td>
<td>—</td>
<td>1</td>
</tr>
<tr>
<td>total</td>
<td>—</td>
<td>3</td>
</tr>
</tbody>
</table>

3.2. Baseline method 2

Baseline method 2 uses a very simple code pipeline, which contains a pretrained model [33] for feature extraction, and a DTW [23] discriminator. The pretrained model is trained by Librispeech dataset without finetuning, and we use the quantization module to extract speech representation, with 256 dimensions.

After feature extraction, DTW is used for feature comparison. A long test audio will be compared with the enrollment audio segment by segment, and the minimum score will be taken as the score of the test audio. If the score is bigger than the threshold, the audio is judged to be awake. The awake threshold is calculated by DTW scores from the enrollment audio, and constrained between 0.45 and 0.6.

3.3. Experiments

We submitted two baselines mentioned above to the challenge platform, according to the competition process. Each baseline was submitted twice and run on feedback and private datasets automatically. The experiments are carried out on Google Cloud virtual machine instances under Ubuntu 18.04, with one single GPU (Nvidia Tesla P100) running CUDA 10 with drivers cuDNN 7.5, 100 GB Disk and 26 GB Memory. The experiments are constrained to be completed within the same limited time as the requirement to participants. The results on feedback and private datasets are presented in Table 4, and details can be viewed on the platform.

As shown in Table 4, the performance of Baseline method 1 is better than that of Baseline method 2 in terms of Miss Rate, False Alarm Rate and final Average Score. Moreover, the running efficiency of Baseline method 1 is higher than that of Baseline method 2, although the pipeline of Baseline method 2 is much simpler. This challenge is a particularly difficult problem, so the performance of the two baselines is relatively poor, leaving a lot of room for participants to improve.

4. Conclusions

Auto-KWS 2021 focuses on Automated Machine Learning for customized Keyword Spotting tasks. In this paper, we introduce the setup of the Auto-KWS 2021 and describe the protocol, metrics, datasets, starting kit and two baseline systems of the challenge. Personalized and customizable KWS is a very interesting and practical scene. According to this baseline experiment, however, it is still a field that has not been conquered. It is expected that researchers from both academia and industry can advance problem solving through this challenge.

5. Acknowledgements

This project was supported in part by 4Paradigm Inc., ASLP@NPU and ChaLearn. The authors were grateful to Isabelle Guyon and Zhen Xu for computational resources. The platform, automl.ai\(^\text{9}\), is built based on Codalab\(^\text{9}\), an web-based platform for machine learning competitions. Thanks to Xiawei Guo, Shousiang Liu, Zhen Xu, Feng Bin and Bojian Hou for their support on challenge platform construction.
6. References


