EPiC Series in Computing

Volume 97, 2024, Pages 30–37

Proceedings of 36th International Conference on
Computer Applications in Industry and Engineering

KWDOA: Adapted dataset for detection of the direction of arrival of the keyword

David Beneš and Luboš Šmídl

Department of Cybernetics, University of West Bohemia, Pilsen, Czech Republic
benesda@ntis.zcu.cz, smidl@kky.zcu.cz

Abstract

This paper describes a simulated audio dataset of spoken words which accommodate microphone array design for training and evaluating keywords spotting systems. With this dataset you could train a neural network for the detection direction of the speaker. Which is an advanced version of the original, with added noises during a speech in random locations and different rooms with different reverb. Hence it should be closer to real-world long-range applications. This task could be a new challenge for the direction of arrival activated by keyword spotting systems. Let's call this task KWDOA. This dataset could serve as the intro level for microphone array designs.

1 Introduction

Keyword spotting [11] (later only referred to as KWS) is a challenging task that can be used in many technology areas. KWS is often used as a wake-up system for home assistants or systems like Google, and Siri on phones. It requires high-efficiency computation and low memory utilization while maintaining high accuracy and low power consumption. However, there is little amount of datasets that use well-described multi-channel recordings. This dataset is simulated in a Python environment using module pyroomacoustics [1] for a specific real-world microphone array with the inner distance of microphones equal to 57mm. This microphone array can be used in the final prototype or as an education/testing/recording device.

The original dataset, Speech Commands Dataset [2], contains recordings of numerous speakers. It is very useful as a dataset for close-range KWS. The current state-of-the-art model by score on Google Speech Commands V1-12 with paper is BC-ResNet-8 [3] to the date of writing this paper. This dataset should create more diversity among different models due to more varied environments. It can also help produce better KWS for devices like home assistants or environment-aware robots. The new dataset is released under Creative Commons BY 4.0 license [4]. This means that anyone can reach this dataset and use it in research or development.

2 Motivations

Basically, every speech-based machine interaction depends on KWS to start interactions. Nice examples of starting interaction with machines are the phrases “Ok Google”, “Hey Alexa” or “Hey Siri”. Any of these phrases are detected on the device alone. After capturing and resulting in positive KWS, the rest of the audio can be sent to the server for further analysis. This approach reduces the amount of audio sent over the web to the server. In the case we didn't use KWS directly on the device we could risk violation of privacy, very high usage of data transfer over the web and higher power usage due to constant activity of wireless modules and/or CPU for compression/decompression.

The goal is the most power-efficient direction of arrival KWS model with the lowest memory requirement. In an ideal world, we would like to use a solution on a single chip with low memory usage and a low-power CPU. Usually, higher CPU usage results in higher power consumption, and higher memory usage results in the limitation of background processes.

When the model detects KWS on the device, it starts sending audio to the server, which will run a complex model for other tasks such as ASR, speech HMI, and so on. The main target of this dataset is deployment to the home assistants or robots with microphone arrays. Of course, it can be used for single microphone devices like cell phones by processing only one channel in the dataset.

The advantages of this dataset are numerous. The most notable is the usage of multi-microphones. It can be used for the direction of arrival [12] (DOA) estimations. For home assistants, it can be used as a limiting factor when a user set-up a location where it shouldn't react to KWS at all. Or if the home assistant is sitting between the kitchen and living room, it can detect the speaker's location for advanced analysis of requests. In the case of robots, it can simply rotate the head with sensors like a camera to the location where a person spoke a keyword/phrase. High-quality DOA can help with audio beamforming [10] as apriori information before calculation. Other advantages of this dataset are more complex sounds with random noises and a random environment. During training/evaluation, it is more challenging to get better results. However, the real world isn't perfect so training data shouldn't be as well.

3 Dataset Revision

The original Speech Commands Dataset has a couple of errors in audio recordings. First of all, there are several occurrences of exactly the same recordings (data duplicates) from the same speaker for the same class. These duplicates can be harmful during the training of neural networks because of the higher value for this occurrence. The other problem is the occurrence of only noise in the recording. This noise is often a simple sine wave, some white noise, or just environmental noise. This noise can be harmful to class differences during training. The last problem that occurs is a clipping of words in the middle of a speech. For example, during command “Five” there is recorded only “Fi” and “ve” is missing due to a clip of the word. The rest of the command is noise/silence.

Removing audio duplicates is simple. Calculation of differences of signals. Then find min/-max of the resulting signal. This is applied between every unique pair of audio files from the same class. If min/max is equal to 0, the signal is duplicated or there are zeroes in both signals. The next step is using simple VAD [5] with probability output of speech present. Every recording is evaluated using VAD, focusing on these points of interest:

- Maximum probability of recording being speech is less than 0.8 (not being speech just noise/silence).

- Speech is detected at the end of the recording (high probability of word clipping).
- Length of speech is longer or shorter than average for that class by 33

If any recording meets some of the criteria above, that recording is tagged. Every tagged recording was manually checked and removed from the dataset if considered as problematic. Recordings that are too noisy was deleted because of propagation during simulations, which can be bad for the final dataset. Almost 7000 recordings were removed from the original dataset.

4 Simulations

Every sound file was simulated using Python's pyroomacoustics [1] module. For a single file, there were multiple simulation runs. Every run with different room sizes, different reverb times, and different locations of the microphone array and source in the room. Up to that, multiple sources of noise are added randomly in the room. We can imagine it in a real room with running machines or background noises like a water sink.

Output simulated data are placed in separate folders as in the original dataset [2]. That means for example folders: bed, one, bird, ... All information used in the simulation is propagated into file names except locations of noises that are irrelevant to this task. The key for deciphering information is as follows: Every information is divided by char "-" and sub-information is divided by char "_". So the name of the file can be described: Num - DimRoom - LocSource - LocMic - PolLoc - ReverbTime - Length - Name.flac

Where:

- Num is the number of current simulations for a specific file from the Google speech dataset. Useful for backtracking the original signal. If someone would like to adapt some distributed final model.
- DimRoom is the dimension of the room. DimRoom stands for DRx_DRy_DRz obviously it contains 3 float numbers describing the room size in cartesian coordinates.
- LocSource is the location of the speaker in the room. Like DimRoom, LocSource contains three float numbers describing the speaker's position in cartesian coordinates.
- LocMic is the location of the microphone array in the room. Like DimRoom, LocMic contains three float numbers describing the position of the center of the microphone array in cartesian coordinates. The microphone array is circular with omnidirectional microphones. The diameter of the microphones is 79mm. The microphone array is always parallel to the floor.
- PolLoc is a polar description of the location in the room from LocMic to LocSource. There are three float numbers in the following order. The first number stands for the distance between the middle of the microphone array and the speaker; it should be in the range [0.5; 3.0] meters. The second number is Azimut in the range [0°; 359°]. The third number is the elevation from [30°; 150°]. For a well-known description in polar coordinates, we need to subtract 180° from azimuth and 90° from elevation.
- ReverbTime is the time used in the simulation, meaning the time period (known as RT60) of the reverberation through the room.

- Length is the estimation of the length of the spoken keyword in the original audio file. This number is in the samples. Estimation is based on VAD [5]. However, it serves just an informative purpose.
- Name is the file's original name in the Google speech dataset.

Example of file naming: 11-7.94_7.75_3.31-5.64_5.7_1.91-3.2_4.4_2.6-2.85_208.0_76.0-0.25-9376.0-bab36420_nohash_0.flac

- 11 - number of simulation run
- 7.94_7.75_3.31 - dimensions of the room: X=7.94[m]; Y=7.75[m]; Z=3.31[m]
- 5.64_5.7_1.91 - location of the source: X=5.64[m]; Y=5.70[m]; Z=1.91[m]
- 3.2_4.4_2.6 - location of the microphones: X=3.20[m]; Y=4.40[m]; Z=2.60[m]
- 2.85_208.0_76.0 - polar coordinates: r=2.85[m]; θ=208.0[°]; ϕ=76.0[°]
- 9376.0 - estimated length of the command: τ=9376.0[1]
- 0.25 - reverb time: RT60=0.25[s]
- bab36420_nohash_0 - name of the original audio file

The audio file format is in FLAC [9] format with a 16 kHz sampling rate. FLAC format is a lossless free audio codec that can accommodate 4-channel audio in a single file. Several current simulation is important. If the number is lower than 4, the audio is clean without any added noises. The output depends only on room size, source locations, microphones, and reverb time. On the other hand, if the number is higher than 4, the audio is simulated with random noises around the room. SNR is based on that number, too, as described in Table 1.

Table 1: Overview of different SNR

Current number of simulation [Num]	**Mean SNR**
8 > Num > 4	22dB
12 > Num > 8	16dB
Num >12	10dB

The source of added noises is background noise from the original dataset and ESC dataset [6]. From the ESC dataset, only 20 classes are used. There are several recordings for the same class and these recordings are converted into a single file, silence removed (clipped to 0.2s) and re-sampled to 16kHz. If noise is added to the simulation, there are three random noise sources in random locations. A segment of noise that is added to the simulation is randomly selected; the length of the segment is based on the length of the speech command.

The final KWDOA dataset [8] contains simulated non-speech recordings. It is simulated in the same manner as commands but for noises only. These simulations of noise can help during training and/or for other purposes, such as training VAD or DOA standalone. In the dataset are 500k of noise recordings. We encounter most of these noises on a daily basis. There are 20000 simulations for every noise entry with a fixed length of 1 second: brushing teeth, can opening, cat, clapping, coughing, crying baby, dog, doing the dishes, door knock, door wood

cracks, drinking sipping, miaowing, exercise bike, footsteps, keyboard typing, laughing, mouse click, pouring water, sneezing, snoring, toilet flush, vacuum cleaner, washing machine, white noise, wind.

5 Baseline Neural Network Model

The baseline model is an adaptation of 3x1x64 MatchBoxNet [7] referred to as MBN in the future. This model can be restructured for multiple inputs (of channels) and there is a possibility for a full or partly quantized version of the model. MatchBoxNet (MBN) achieved good results with 1-channel speech commands with 96,97 % accuracy for 35 commands. On the re-simulated dataset, the same neural network achieved only 89,19 % accuracy trained with every channel of all audio samples in training.

The first change in the model is the input, which is the energy of the frequency spectrum. This change provides better accuracy by at least 2 %. The next change is training by Adam with a learning rate of 0.002 and weight decay of 0.0001. The activation function was chosen SiLU, where possible, which achieved a slightly better score. If training wasn't achieved a lower loss of validation set in the next five epochs, the learning rate was decreased by 10% of the current value. The final and most important is parallel training of the same model on channels alone. This means a Neural network with 77.3k params is trained on every channel of the audio file at once. By doing this, the final mean accuracy achieved 91,52% and when added weights per class 93,60 % on the validation dataset.

The final structure of the neural network is described in Table 2. The input of the network is four channels audio with 16000 samples (which is 1s long audio padded with zeros if needed). From the audio is calculated energy of FFT with the length of window 512 and hop 160, resulting in a tensor of size 4x257x101. Followed by the main structure of MBN for every channel with shared variables. The output of the network is 4 separated tensors with 2 channels. From these tensors is calculated the mean value for channels and tensors separately. Resulting in the X channel tensor, where X is defined by number of classes.

Table 2: Detail of main structure 3x1x64 MBN design

Block	# Blocks	# Sub Blocks	Output shape	Kernel
Conv1	# 1	# 1	128 x 46	11, stride=2
B1	# 1	# 2	64 x 46	13
B2	# 1	# 2	64 x 46	15
B3	# 1	# 2	64 x 46	17
Conv2	# 1	# 1	128 x 2	23, dilation=2
Conv3	# 1	# 1	128 x 2	1, stride=1
Conv3	# 1	# 1	# classes x 2	1, stride=1
Mean			# classes	
Concat			4 x # classes	
Mean			# classes	
Soft-max				

Table 3 and Table 4 summarize the results for validation and test parts of the dataset. There is a split between SNR for a better overview. Adapted MBN weighted (Adapted MBN

with added weights per class) is my proposed baseline neural network. Ad. MBN 12 classes is a neural network trained to recognize classes: yes, no, up, down, left, right, on, off, stop, go, rest, and noise. Where noise is added randomly from a provided dataset with 50000 samples, the rest are every other class considered as one class. There is no used training data without added noise.

Table 3: Validation results of several Neural networks

Model	**Validation Acc**[%]		
	22dB SNR	16dB SNR	10dB SNR
Orig. MBN	90.77%	89.54%	87.27%
Adapted MBN	92.99%	91.88%	89.70%
Ad. MBN weighted	95.19%	94.05%	91.57%
Ad. MBN 12 classes	96.44%	95.66%	94.11%

Table 4: Test results of several Neural networks

Model	**Test Acc**[%]		
	22dB SNR	16dB SNR	10dB SNR
Orig. MBN	89.45%	88.15%	85.76%
Adapted MBN	91.93%	90.66%	88.24%
Ad. MBN weighted	94.00%	92.66%	90.11%
Ad. MBN 12 classes	96.10%	95.22%	93.59%

The direction of arrival uses the same format of the neural network as MatchBoxNet but with limited parameters with configurations 2x1x64 and 2x1x32. A detailed overview is in Table 5. Which resulted in 91k and 29k params, respectively. As inputs are used, the phase of FFT is multiplied by the energy of FFT and all four channels at once, which are decimated during inference. Output mean absolute error value and mean time for one inference are shown in Table 6, with some results of other well-known algorithms. Inference times are based and compared on a single core of intel 11400F. Achieved results of a mean absolute error are more than good for the sense of direction of home assistants or robots.

6 Conclusion

As we can see, the differences between results are large. Using an adaptation of Matchbox Net achieved good results. However, there is plenty of room for improvement. For KWS detection was achieved an accuracy of around 95.4% for 12 classes and 93,6% for 35 classes on validation data. And for test data, score was 94.97% for 12 classes and 92.26% for 35 classes. Part of the KWDOA problem is DOA itself, which resulted in worse mean absolute error compared to MUSIC or SRP-HAT but with much higher computational speeds and lower memory usage. With a quantized version of the networks, the speed and memory usage can be improved almost four times. With this dataset, we can develop KWS further and add more features to systems with microphone arrays.

Table 5: Detail of main structure 2x1xCH MBN design for DOA

Block	# Blocks	# Sub Blocks	Output shape	Kernel
Conv1	# 1	# 1	64 x 4 x 46	11, stride=2
B1	# 1	# 2	32 x 3 x 46	13
B2	# 1	# 2	32 x 2 x 46	15
Conv2	# 1	# 1	64 x 1 x 2	23, dilation=2
Conv3	# 1	# 1	64 x 1	1, stride=1
Conv3 Re-LU	# 1	# 1	# 1	1, stride=1

Table 6: Direction of arrival - precision and time of inference/calculation

Method	MBN-91k	MBN-29k	MUSIC	SRP-HAT	TOPS
MAE	10.91°	12.01°	8.53°	4.66°	6.55°
Time	5.86ms	5.16ms	11.49ms	18.9ms	343ms

7 Acknowledgements

This research was supported by the Ministry of the Interior of the Czech Republic, project No. VJ01010108. Computational resources were supplied by the project e–INFRA LM2018140.

References

[1] Robin , Scheibler, et al. PYROOMACOUSTICS: A PYTHON PACKAGE FOR AUDIO ROOM SIMULATION AND ARRAY PROCESSING ALGORITHMS. 2017, https://www.researchgate.net/publication/320344643_Pyroomacoustics_A_Python_Package_for_Audio_Room_Simulation_and_Array_Processing_Algorithms.

[2] Warden, Pete. Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. 2018, https://arxiv.org/pdf/1804.03209.pdf.

[3] Kim, Byeonggeun, et al. Broadcasted Residual Learning for Efficient Keyword Spotting. 2021, https://arxiv.org/pdf/2106.04140v2.pdf.

[4] Creative commons international attribution international 4.0 license. [Online]. Available: https://creativecommons.org/licenses/by/4.0/

[5] Team Silero. Silero VAD: Pre-Trained Enterprise-Grade Voice Activity Detector (VAD), Number Detector and Language Classifier. GitHub, 2021, https://github.com/snakers4/silero-vad.

[6] Piczak, Karol J. ESC: Dataset for Environmental Sound Classification. 2015, https://www.karolpiczak.com/papers/Piczak2015-ESC-Dataset.pdf.

[7] Majumdar, Somshubra, and Boris Ginsburg. MatchboxNet: 1D Time-Channel Separable Convolutional Neural Network Architecture for Speech Commands Recognition. 2020, https://arxiv.org/pdf/2004.08531v2.pdf.

[8] Speech Commands Dataset Enhanced for Direction-of-Arrival Estimation https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-5140

[9] Xiph.org. FLAC - format. 2014. https://xiph.org/flac/format.html.

[10] Barry D. Van Veen and Kevin M. Buckley. Beamforming: A Versatile Aproach to Spatial Filtering. http://users.umiacs.umd.edu/ ramani/cmsc828d_audio/Beamforming.pdf.

[11] Tara N. Sainath, Carolina Parada. Convolutional Neural Networks for Small-footprint Keyword Spotting.

[12] Pooja Gupta, S.P. Kar. MUSIC and improved MUSIC algorithm to estimate direction of arrival. https://ieeexplore.ieee.org/document/7322593.