Device playback augmentation with echo cancellation for keyword spotting

Kuba Łopatka, Katarzyna Kaszuba-Miotke, Piotr Klinke, Pawel Trella

Intel Corporation, Gda´nsk, Poland
kuba.lopatka, katarzyna.kaszuba, piotr.klinke, pawel.trella@intel.com

Abstract

Keyword spotting (KWS) is required to operate in device playback conditions in which the device itself plays interfering signals. We propose a new method to augment the training set and adapt the acoustic model to the playback environment. It is based on acoustic simulation which models the coupling between the device’s loudspeakers and microphones. The employed model involves frequency response of the device, as well as room impulse response and nonlinear distortions introduced in the playback path. Finally, we pass the simulated signals through Acoustic Echo Cancellation (AEC) to model the artifacts introduced by AEC algorithm. The proposed method reduces False Rejection Rate in device playback by 25-60% for a Time-Delay Neural Network-based KWS engine. It is shown that the introduction of device characteristics and nonlinear filtration is necessary to achieve improvement in playback conditions. The augmentation scheme is highly independent of the architecture of the KWS system.

Index Terms: keyword spotting, acoustic echo cancellation, data augmentation

1. Introduction

Keyword spotting (KWS) is an algorithm which detects a pre-defined word in the audio input stream, typically to trigger personal assistant software. The task is related to Automatic Speech Recognition (ASR) but has some unique challenges. KWS often runs locally on a battery-operated device, so power is a key factor [1]. The compute and memory footprint [2], as well false detection rate have to be kept to a minimum [3]. A practical system has two stages, where the first stage model works with limited resources and second stage verifies the detection with more resources available [4].

Several model architectures have been tried for KWS, including multi-layer perceptron [5], Convolutional Neural Networks (CNN) [3, 6], Time-Delay Neural Networks (TDNN) [1] or recurrent networks [7, 8]. Latest research on KWS includes attention pooling [9] or max pooling [7, 10] as well as spiking neural networks (SNN) [11]. Some researchers introduce end-to-end architectures with max-pooling loss [12, 13] or raw audio input [14].

Keyword spotting is required to work in challenging conditions, in the presence of room reflections and additive noise. For both KWS and ASR, the state-of-the-art approach to make the model robust against challenging environments consists of augmenting the training set. Most widely used augmentations include adding noise and reverberation to clean speech [15, 16, 17]. Some approaches involve transformations of source speech such as changing tempo, pitch or vocal tract characteristics [18]. Researchers also report improvement after adding synthesized speech samples to the training set [19]. Finally, methods like SpecAugment transform the features directly before they are fed into the model [20].

Arguably, the most difficult condition is device playback. In this condition the user utters the keyword while the device is playing sounds (e.g. music) through built-in loudspeakers. As it is shown in Fig. 1, the playback sound leaks to the microphone through direct coupling, internal coupling and room reflections. Due to physical proximity of playback and capture transducers, the leakage is often louder than speech, thus making it difficult to recognize the keyword. Typically, Acoustic Echo Cancellation (AEC) is employed to filter out the leakage [21]. It is achieved by adaptive filtration with loopback signal used as reference. After AEC, playback is highly attenuated and speech is more intelligible. However, due to nonlinear distortions present in the audio path, considerable artifacts remain which significantly impair the accuracy of KWS [22]. There is little research focusing on improving KWS performance under playback conditions. Raju et al. study the impact of adding music signals to speech in the training set and report improvement compared to a model trained on clean speech [23].

Our solution is to include simulation of device playback in the process of training the acoustic model. We simulate the capture path of the device and process the training signals with AEC to incorporate specific echo cancellation artifacts. In order to achieve high accuracy the frequency response of loudspeakers and microphones, as well nonlinear distortions are factored into the simulation. By augmenting the training set with playback simulation we achieve increased robustness of the acoustic model which leads to consistent improvement in KWS performance for signals after echo cancellation. To the best of our knowledge, simulation of device playback with AEC has not been included in training ASR or KWS models before.

2. Acoustic simulation

2.1. Noise and reverberation

The most common approach to prepare a multi-condition training set is based on applying a room response filter $h_{RIR}$ to clean speech $s(t)$ and adding a noise signal $n(t)$.

$$x(t) = s(t) * h_{RIR}(t) + G_s * n(t) * h_{RIR}(t)$$

(1)

Additional noise gain $G_s$ is used to control the resulting Signal-to-Noise Ratio (SNR). Room impulse response $h_{RIR}$
can originate from a database of impulse responses [24] or it can be generated using software such as pyroomacoustics [25] given the geometric setup of the room, sources and receivers.

This methodology is very widely adopted [23, 15, 16, 17]. It yields good results for noise and far field scenarios but falls short when dealing with device playback.

2.2. Device playback

To properly simulate device playback specific transformations have to be introduced in the capture path, which resemble distortions observed on real devices. The simulation diagram is featured in Fig. 2. In this subsection we introduce the formulas to simulate the respective components in the audio path.

The goal is to model the capture signal $c(t)$ which is registered by the built-in microphone. It is a sum of 4 components: speech signal $s'(t)$, self noise $w_N(t)$, playback leakage $m_{C}(t)$ and playback room-related echo $m_{R}(t)$.

$$c(t) = s'(t) + w_N(t) + m_{C}(t) + m_{R}(t) \quad (2)$$

These components are simulated by convolution of a source signal with an impulse response and scaling by gain.

$$c(t) = G_S \cdot s(t) \ast h_S(t) + G_N \cdot w(t) \ast h_N(t) + G_C \cdot m(t) \ast h_{C}(t) + G_R \cdot m(t) \ast h_R(t) \quad (3)$$

Source signals for speech $s(t)$ and playback $m(t)$ are sampled from the training and augmentation set described in Sec. 4. For device self noise we use white noise $w(t)$. Gain is introduced to control the resulting SNR, or to model additional acoustic properties. For instance, speech gain $G_S$ and room echo gain $G_R$ are related to microphone sensitivity, whereas coupling gain $G_C$ also depends on weighted terminal coupling loss $TCL_w$.

Impulse responses $h_S(t)$, $h_C(t)$ and $h_R(t)$ are decomposed into the following factors.

$$h_S(t) = h_{RIR_S}(t) \ast h_{mics}(t) \quad (4)$$

$$h_C(t) = h_{C}(t) \ast h_{C}(t) \quad (5)$$

$$h_R(t) = h_{RIR_R}(t) \ast h_{spkr}(t) \ast h_{micR}(t) \quad (6)$$

where $h_{RIR_S}$ and $h_{RIR_R}$ are room impulse responses applied to speech and room echo respectively, $h_{C}$ and $h_{C}$ model internal and direct coupling, $h_{spkr}$ is the diffuse field loudspeaker impulse response, $h_{micR}$ is the diffuse field microphone impulse response and $h_{mics}$ is the free field microphone impulse response.

So far, the model incorporates only linear distortions of the audio path. In reality, there are non-linear distortions present as well, which are the main cause of artifacts observed after echo cancellation [26]. To factor in the non-linearities, the resulting capture signal $c'(t)$ is processed by a non-linear filter $NLF$. The employed non-linear filter is discussed in Sec. 2.4.

2.3. Parametric equalization

The impulse responses $(h_S, h_C, h_{C}, h_{spkr}, h_{micR}, h_{micC})$ are modeled by parametric equalization. Parametric filters are designed to resemble the frequency responses of real devices, based on multiple measurements. We provide an example of how the microphone impulse response is modeled.

The two most important factors which shape the frequency response of a digital microphone are low frequency ventilation hole effect and Helmholtz resonance in high frequencies [27, 28]. Low frequency filtration $H_{bpf}(f)$ is characterized by the cutoff frequency $f_c$ and attenuation in decibels per octave $k$, while Helmholtz resonance $H_{peak}(f)$ depends on resonance frequency $f_{peak}$, resonance gain $g_{peak}$ and resonance Q-factor $Q_{peak}$. Both factors are modeled in frequency domain $H_{mic}(f)$ and approximating filter $h_{mic}(t)$ is calculated using frequency sampling FIR filter design method [29].

$$H_{bpf}(f) = -20 \log \frac{\sqrt{1 + 10^{-4 k \log(f)/\log(2) + \log(f_c)}}}{10} \quad (7)$$

$$H_{peak}(f) = g_{peak} \cdot e^{-\frac{1}{2} (f - f_{peak}) Q_{peak} \frac{1}{f_{peak}}} \quad (8)$$

$$H_{mic}(f) = H_{bpf}(f) + H_{peak}(f) \quad (9)$$

By altering the parameters of these filters, an arbitrary number of device characteristics can be generated. This mechanism prevents overfitting to specific devices which is an advantage over using frequency responses originating from measurement. An analogous process is employed to model the remaining impulse responses featured in the simulation.

2.4. Nonlinear filtering

Nonlinear behavior of the acoustic path is mostly related to physical properties of loudspeakers, such as suspension stiffness and voice coil inductance [30], as well as internal coupling through non-ideal common chassis supporting both speakers and microphones [31]. We model the nonlinearities with waveshaping method using $N^{th}$ order Chebyshev Polynomials with $N = 5$. Model produces both intermodulation and harmonic distortion components. Each polynomial has a single scaling parameter $\alpha_n$. First order linear component uses
\( \alpha_1 = 1. \) Scales for higher order polynomials are smaller. DC component is compensated afterwards.

\[
\begin{align*}
T_0(x) &= 1 \\
T_1(x) &= 2x - 1 \\
T_{n+1}(x) &= 2xT_n - T_{n-1} \quad (10) \\
NLF(x) &= \sum_{n=1}^{N} T_n(x) \cdot \frac{\alpha_n}{N} \quad (11)
\end{align*}
\]

### 2.5. Acoustic echo cancellation

Finally, the capture and loopback signals are processed with AEC. The employed algorithm is Frequency-domain adaptive Kalman Filter (FDKF) [32]. The filter requires input (capture) and reference (loopback). The loopback signal \( m'_c(t) \) is the raw device playback signal \( m(t) \) scaled with gain relative to the device loopback sensitivity. Due to the distortions introduced in the simulation path the resulting sample \( x(t) \) incorporates artifacts very similar to the ones observed in real signals passed through AEC. Hence, the keyword spotting engine can be adapted to classify such signals with higher accuracy.

### 2.6. Simulation parameters

Selected parameters which can be adjusted in simulation along with the assumed range of values are listed in Tab. 1. The values can be sampled at random but some supervision is beneficial to make sure that the resulting characteristics are realistic. Thus, we define a set of virtual devices which we employ to generate the training samples.

<table>
<thead>
<tr>
<th>parametric model</th>
<th>parameter</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gain SNR sensitivity</td>
<td>(-7dB_s, +7dB_s)</td>
<td>(-26dB_{FS}/Pa)</td>
</tr>
<tr>
<td>Frequency response</td>
<td>(f_c)</td>
<td>((100Hz, 250Hz))</td>
</tr>
<tr>
<td></td>
<td>(k)</td>
<td>((3dB/oct, 6dB/oct))</td>
</tr>
<tr>
<td></td>
<td>(f_{peak})</td>
<td>((6kHz, 20kHz))</td>
</tr>
<tr>
<td></td>
<td>(g_{peak})</td>
<td>((1, 15))</td>
</tr>
<tr>
<td>Device coupling (G_D)</td>
<td>TCF</td>
<td>((-10dB, +10dB))</td>
</tr>
<tr>
<td>Self noise (h_{SN})</td>
<td>SNR</td>
<td>((60dB, 70dB))</td>
</tr>
<tr>
<td>Nonlinear filter</td>
<td>(\alpha_n)</td>
<td>((0, 0.1))</td>
</tr>
</tbody>
</table>

### 3. Keyword spotting engine

Our engine, which is presented in Fig. 3, is composed of three main blocks: frontend, acoustic model and keyword/rejection model. The details of our algorithms are discussed in previous work [33]. The effect of augmenting the training set with device playback simulation is largely independent from the employed KWS algorithm.

#### 3.1. Acoustic model

The input of the acoustic model are 40 log-filterbank features extracted from 25 ms overlapping frames with 10 ms shift. The network is prepended with a scaling layer which standardizes the input features. We use 3 TDNN layers with 128 units and bottleneck layers in between, all with rectified linear unit (ReLU) activation. The TDNN layers connect to activations from previous layers from 3 frames back \((t - 3)\) and 3 frames ahead \((t + 3)\). This enables skipping every 2nd and 3rd frame during inference to save computations, as proposed in [34]. The final layer has 3897 outputs for all Large Vocabulary Continuous Speech Recognition (LVCSR) posteriors. Only 86 of these posteriors are used in the keyword/rejection model (36 for keyword, 50 for rejection). In total, the neural network has 135K parameters which makes it suitable for deployment as a first pass model in embedded environment.

### 3.2. Keyword model

The phrase used in experiments is Hello computer. It comprises 12 triphones, which yields \(N = 36\) states \((3 \text{ per triphone})\).

For each time step \(t\) the acoustic model estimates a vector of framewise posteriors \([p_1, p_2, ..., p_N]\) which relate to triphone states of the keyword \(S_1\) to \(S_N\). The first state \(S_0\) is a rejection state. It is updated based on a set \(R\) of 50 rejection posteriors chosen with a heuristics based on a priori phonetic knowledge and triphone statistics of the English language.

\[
\begin{align*}
S_0(t) &= 0 \\
S_n(t) &= S_0(t - 1) + \max_{m \in R}(p_m(t)) \quad (12)
\end{align*}
\]

For \(n = 1..N\) the state scores are updated in each time step:

\[
\begin{align*}
S_n(t) &= -\infty \\
S_n(t) &= \max_{t \in \{1, 2, ..., T - 1\}}(S_n(t - 1) + p_n(t)) + \max_{m \in R}(p_m(t)) \quad (13)
\end{align*}
\]

Finally, the final score \(S_{final}\) is computed.

\[
S_{final} = \max_{t \in \{1, 2, ..., T - 1\}}(S_N(t) - S_0(t)) \quad (14)
\]

### 3.3. Model training

The model is trained with state sequence pooling method [33]. It combines standard framewise cross entropy with a dedicated loss function tied to the keyword score. We first pretrain the model for 50 iterations employing just cross entropy loss with LVCSR targets and then we train for 20 more iterations with state sequence pooling loss. We use Adam optimizer with learning rate of 0.001.

### 4. Experiment

#### 4.1. Training set

| Table 2: Training sets employed for specific models |
|-------------------------------|---|---|---|---|---|
| **model** | **A** | **B** | **C** | **D** | **E** |
| non keywords | clean | 47/7k | 47/7k | | | |
| | noise/reverb | 206k | 165k | | | |
| | playback | 0 | 52k | | | |
| keywords | clean | 1.5k | 1.5k | | | |
| | noise/reverb | 10k | 7k | | | |
| | playback | 0 | 5k | | | |
| TOTAL | 706k | 720k | | | | |
| playback simulation factors | device | | | | | |
| | nonlinearity | | | | | |
| | AEC | | | | | |

The structure of the training set is outlined in Tab. 2. For non-keywords, we use US English Speecron database [35]. As keyword samples we use 1500 internal and crowd-sourced
1. Augmentation signals

For noise signal $n(t)$ we utilize Speecoon [35] and MUSAN [36] open corpora. We select 2k samples for noise and 700 samples for music. For room simulation ($h_{RIR}(t)$) we employ impulse responses published in [24]. Device playback simulation utilizes the same 700 samples of music signals for $m(t)$. Target SNR for the noise/reverb case is set in the range (-10,10) dB.

4.3. Evaluation set

For evaluation, we use 389 internal recordings of keyword samples. Instead of simulated characteristic, real impulse responses and real music leakage signals recorded on the target device are used to generate signals with high fidelity. The generated signal was shown to be very close to real-life signals through extensive acoustic tests. The signals are passed through AEC with FDKF algorithm and then fed into the keyword recognizer. For non-keywords, we use snippets from US English podcasts mined from sources with open license. We extract 17280 random 5-second snippets which totals of 24 hours.

4.4. Results

The plot in Fig. 4 presents the tradeoff between false rejection rate (FRR) and false accept rate (FAR). The models defined in Tab. 2 are compared. The models are quantized in 8-bit fixed point format. We present average results for two music recordings. The model trained on just noise/reverb augmentations (A) serves as baseline. With the full simulation path, including device characteristics and non-linear filtration (model B), there is visible improvement over the baseline. Model C, including device characteristics but no nonlinearities, is better than base-line for some threshold values, but the improvement is inconsistent. We also observe that the model trained without device characteristics (D) and the model incorporating only the AEC algorithm (E) yield no improvement over the baseline. Comparing the best model (B) to the baseline (A) we observe an FRR reduction ranging from 25.2% at 1 false wake per 8 hours (from 27.98% to 20.92%) to 60.4% at 1 false wake in 10 minutes (from 6.16% to 2.44%). On average the FRR is reduced by 45.6%.

5. Conclusions

We introduce a simulation framework to augment the training set for keyword spotting under device playback conditions. We achieve visible improvement in keyword detection accuracy. Interestingly, consistent improvement is observed only when all simulation factors are enabled, which enables close reproduction of the acoustic path of the device.

This work can be extended in the following ways. Firstly, we aim to examine the relation between the number of simulated devices and the ranges of simulation parameters to the accuracy of the trained model. Secondly, instead of fixed impulse responses, room geometry can be defined and the impulse responses can be generated on the fly. Thirdly, it is interesting to examine the performance of the trained model in various conditions, i.e. recorded on different devices, with different playback signals or noises. Finally, the proposed augmentation method can be applied to other tasks than keyword spotting, including automatic speech recognition or acoustic event detection.
6. References


