2024

AdaStreamLite: Environment-adaptive Streaming Speech Recognition on Mobile Devices

Yuheng Wei
Xidian University

Jie Xiong
University of Massachusetts Amherst

Hui Liu
Xidian University

Yingtao Yu
Xidian University

Jiangtao Pan
Xidian University

See next page for additional authors

Recommended Citation

Wei, Yuheng; Xiong, Jie; Liu, Hui; Yu, Yingtao; Pan, Jiangtao; and Du, Junzhao, "AdaStreamLite: Environment-adaptive Streaming Speech Recognition on Mobile Devices" (2024). *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*. 1370.
https://doi.org/10.1145/3631460

AdaStreamLite: Environment-adaptive Streaming Speech Recognition on Mobile Devices

YUHENG WEI, Xidian University, China and Engineering Research Center of Blockchain Technology Application and Evaluation, Ministry of Education, China
JIE XIONG, Microsoft Research Asia, China and University of Massachusetts Amherst, United States
HUI LIU*, Xidian University, China and Engineering Research Center of Blockchain Technology Application and Evaluation, Ministry of Education, China
YINGTAO YU, Xidian University, China
JIANGTAO PAN, Xidian University, China
JUNZHAO DU*, Xidian University, China and Engineering Research Center of Blockchain Technology Application and Evaluation, Ministry of Education, China

Streaming speech recognition aims to transcribe speech to text in a streaming manner, providing real-time speech interaction for smartphone users. However, it is not trivial to develop a high-performance streaming speech recognition system purely running on mobile platforms, due to the complex real-world acoustic environments and the limited computational resources of smartphones. Most existing solutions lack the generalization to unseen environments and have difficulty to work with streaming speech. In this paper, we design AdaStreamLite, an environment-adaptive streaming speech recognition tool for smartphones. AdaStreamLite interacts with its surroundings to capture the characteristics of the current acoustic environment to improve the robustness against ambient noise in a lightweight manner. We design an environment representation extractor to model acoustic environments with compact feature vectors, and construct a representation lookup table to improve the generalization of AdaStreamLite to unseen environments. We train our system using large speech datasets publicly available covering different languages. We conduct experiments in a large range of real acoustic environments with different smartphones. The results show that AdaStreamLite outperforms the state-of-the-art methods in terms of recognition accuracy, computational resource consumption and robustness against unseen environments.

CCS Concepts: • **Human-centered computing → Ubiquitous and mobile computing systems and tools**.

Additional Key Words and Phrases: acoustic environment sensing, ambient noise adaptation, streaming speech recognition, on-device speech recognition

*corresponding author

Authors' addresses: Yuheng Wei, weiyuheng@stu.xidian.edu.cn, Xidian University, NO.2, South Taibai Road, Xi'an, Shaanxi, China, 710071 and Engineering Research Center of Blockchain Technology Application and Evaluation, Ministry of Education, Xi'an, Shaanxi, China; Jie Xiong, jxiong@cs.umass.edu, Microsoft Research Asia, Shanghai, China and University of Massachusetts Amherst, Amherst, United States; Hui Liu, liuhui@xidian.edu.cn, Xidian University, NO.2, South Taibai Road, Xi'an, Shaanxi, China, 710071 and Engineering Research Center of Blockchain Technology Application and Evaluation, Ministry of Education, Xi'an, Shaanxi, China; Yingtao Yu, yingtaoyu@stu.xidian.edu.cn, Xidian University, NO.2, South Taibai Road, Xi'an, Shaanxi, China, 710071; Jiangtao Pan, pjt@stu.xidian.edu.cn, Xidian University, NO.2, South Taibai Road, Xi'an, Shaanxi, China, 710071; Junzhao Du, dujz@xidian.edu.cn, Xidian University, NO.2, South Taibai Road, Xi'an, Shaanxi, China, 710071 and Engineering Research Center of Blockchain Technology Application and Evaluation, Ministry of Education, Xi'an, Shaanxi, China.

2474-9567/2023/12-ART187 $15.00
https://doi.org/10.1145/3631460

ACM Reference Format:
Yuheng Wei, Jie Xiong, Hui Liu, Yingtao Yu, Jiangtao Pan, and Junzhao Du. 2023. AdaStreamLite: Environment-adaptive Streaming Speech Recognition on Mobile Devices. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 7, 4, Article 187 (December 2023), 29 pages. https://doi.org/10.1145/3631460

1 INTRODUCTION

Speech interaction with smartphones [18] has been gradually assimilated into every aspect of our daily life. According to a recent survey [28], more than 1.24 billion adults use voice assistants (e.g., Amazon's Alexa, Apple's Siri, Google Assistant and Microsoft's Cortana) each month, and 91.0% of adult users enjoy the conversational artificial intelligence (CAI) technology on smartphones. For natural speech interaction, an essential part is to accurately recognize the content of utterances from users. Automatic speech recognition (ASR) technologies [47] transcribe speech to text and enable machines to understand what users say. Currently, most speech recognition applications on mobile devices adopt a cloud-edge architecture, due to the high complexity of deep learning based speech recognition models. More specifically, smartphones collect and upload speech data, and a cloud computing center receives and recognizes them. Several limitations of such an architecture have been identified. Firstly, users can not use speech recognition services when the network connection of smartphones is lost. Secondly, a lot of users are worried about privacy disclosure [50] when uploading their speech data to cloud service providers. Thirdly, transmitting speech data consumes much more network bandwidth than text data.

Recently, with the emergence of edge intelligence [71], ubiquitous computing and deep learning communities have been exploring how to transfer the training and inference processes of deep neural networks (DNNs) to edge devices, such as federated learning [33] and model compression [13][35]. In the field of speech signal processing, researchers aim to enable streaming speech recognition on mobile devices to provide real-time output when dealing with long speech input. Despite inspiring progress [26][8][56], real-world noisy environments still bring serious challenges to speech recognition systems, especially those on smartphones. Due to the portability and mobility of smartphones, users can turn on speech recognition services in diverse acoustic environments (e.g., train station, office and restaurant) with distinct types of ambient noises. Non-stationary (i.e., varying with time) ambient noise causes inaccurate recognition results, leading to bad user experience.

The mainstream solution to address noise in ASR is to use a speech enhancement (SE) [36] scheme as the front-end of ASR. The state-of-the-art (SOTA) SE methods [7][49] use a deep neural network (DNN) to estimate a real-valued or complex-valued ratio mask which can map noisy spectrograms to clean spectrograms. Though these SOTA methods can effectively improve the quality and intelligibility of speech signals, they lack the generalization to unseen environments with different noises, since the noise and speech data used to train DNNs are collected from a finite number of acoustic environments. Besides, they fail to cooperate with streaming speech recognition systems. There are two main reasons. The first one is that the separation of front and back ends increases the overall inference latency of a streaming ASR system, especially on mobile devices with limited computational resources. The other is that the streaming (real-time) nature can not provide sufficient context information of user speech required by these SOTA SE models. Developing noise-robust streaming ASR on smartphones remains a challenging problem.

In this paper, we propose a noise-robust speech recognition system deployed on smartphones to meet the needs in complicated noisy environments. We name it AdaStreamLite and it has the following advantages:

(1) **Environment-adaptive**: AdaStreamLite alleviates the impact of harmful ambient noise on recognition in a self-adaptive manner by automatically interacting with the current acoustic environment and leveraging the ambient sound information to improve recognition performance.
(2) **Privacy-protected**: The whole system of AdaStreamLite runs purely on smartphone platforms and generates speech recognition results without uploading any user speech data to cloud. This effectively protects users' privacy.

(3) **Streaming**: AdaStreamLite (including all its functional components) works in a streaming manner which is capable of providing its user with real-time recognition service, i.e., speech is transcribed to text when the user is still in the process of speaking one sentence.
(4) **Customized**: AdaStreamLite allows its users to flexibly determine whether to activate the environment self-adaptation function. This avoids unnecessary computation and saves battery consumption. For example, recognizing user speech in a quiet environment such as a library does not require activating the self-adaptation function.

Despite the above advantages, it is not trivial to realize AdaStreamLite. Several challenges need to be tackled before we can turn the idea into a functional system. The fist challenge is that the ambient noise is very different from white noise which can be modelled as a simple Gaussian distribution. Ambient noise is much more complicated and no simple statistical models can be used to fully characterize the property of the acoustic noise in one environment in either time domain or frequency domain. Addressing ambient noise to achieve high-accuracy streaming speech recognition is challenging. What makes it even more challenging is that acoustic noises in different environments (e.g., server room vs. gym) are dramatically different. To make AdaStreamLite a generic solution, it needs to work with unseen acoustic noises in new environments. The last challenge is that compared to high-performance cloud servers, smartphones are limited in terms of computational resources such as CPU, memory and battery capacity. Realizing real-time speech transcription purely on a smartphone with very limited resources is challenging. We tackle all these challenges in this paper and our main contributions are summarized as follows:

(1) We design an end-to-end speech recognition system named AdaStreamLite for smartphone users, which is capable of real-time speech recognition with all the processing happening purely on a smartphone. Considering the limited computational resources of a smartphone, we comprehensively reduce the complexity of AdaStreamLite through low-dimensional feature encoding, lightweight network structure design, and weight parameter quantization.
(2) We propose an environment self-adaptation mechanism to improve the system robustness against diverse acoustic noises in different environments, moving smartphone-based streaming speech recognition one critical step towards real-life adoption. The self-adaptation model and the streaming speech recognition model are jointly optimized to avoid performance degradation caused by the mismatch between frond and back ends.
(3) We develop a prototype of AdaStreamLite and evaluate it on different smartphones in diverse real-world acoustic environments. The results demonstrate the superior performance of AdaStreamLite over the state-of-the-art speech enhancement methods. Based on the volunteer survey, AdaStreamLite can provide users with satisfactory experience in real-world acoustic environments including those challenging noisy environments such as server room, laundry room and crowded CBD roadside. Our current version supports speeches in English and Chinese and it can be extended to support other languages.

The remaining parts of this paper are arranged as follows. In Section 2 and Section 3, we introduce the related work and background. In Section 4, we present the design details of AdaStreamLite. The system implementation and evaluation of AdaStreamLite are presented in Section 5 and Section 6, respectively, followed by a conclusion in Section 7.

2 RELATED WORK

In this section, we discuss the research closely related to AdaStreamLite.

2.1 Streaming Speech Recognition

Compared to touch, motion and gesture, speech is a more natural way of human-machine interaction. Automatic speech recognition (ASR) plays an important role in realizing this interaction mode. A typical ASR system takes a sequence of acoustic feature vectors as input and predicts a label sequence, where a label can represent a phoneme, a syllable, a letter, or a word. The traditional architecture of ASR [30] consists of three independent components: an acoustic model, a language model and a pronunciation dictionary. The classical combination of Hidden Markov Model (HMM) and Gaussian Mixture Model (GMM) has been a dominant model in the speech recognition field for a long time [45]. With the emergence of artificial intelligence (AI), deep neural network (DNN) has been used to replace GMM in the traditional framework [11]. Nowadays, end-to-end (E2E) models for ASR have demonstrated better performance than the HMM-GMM and HMM-DNN models on most large vocabulary continuous speech recognition (LVCSR) tasks, due to sufficiently large amounts of speech training data. The E2E ASR framework integrates all traditional components into a unit deep network, which directly outputs character sequences without any intermediate steps (such as force data alignment and table lookup). The mainstream ways to implement E2E ASR include Connectionist Temporal Classification (CTC) [20][21], Recurrent Neural Network-Transducer (RNN-T) [19][22], and Attention-based Encoder-Decoder (AED) [10][6][9]. The E2E ASR systems are more friendly to mobile devices, because they greatly simplify the training and inference processes.

With the rapid growth and adoption of smartphones, users urgently expect to interact with smartphones based on speech in a real-time manner. This motivates researchers to explore streaming E2E ASR methods. Different from non-streaming approaches that need to process the entire utterance at once before outputting a character sequence, streaming methods can predict a character during its pronunciation process by only accessing partial speech data. The RNN-T model is inherently suitable for streaming E2E ASR. Several variations of RNN-T are proposed to further improve the performance of streaming E2E ASR systems on smartphones, such as Uni-directional LSTM-Transducer [26], Transformer-Transducer [67] and Conformer-Transducer [8]. The hybrid CTC and AED based ASR framework [61][39][66] can leverage the advantages of both CTC and attention mechanism to achieve state-of-the-art recognition results, but can not be directly applied to streaming E2E ASR. Several methods are proposed to address this issue, such as monotonic chunk-wise attention [31] and monotonic truncated attention [39].

In this paper, we use a U2++ model [65] as the streaming E2E ASR component of AdaStreamLite. The U2++ model adopts the hybrid CTC and AED architecture, which is trained with a dynamic chunk-based attention strategy to support streaming speech recognition.

2.2 Noise Mitigation and Speech Enhancement

In the real world, ASR systems usually suffer from severe performance degradation caused by background noises from their surroundings. Noise-robust ASR methods [70] are proposed to alleviate the harmful effect of noise on speech recognition. They can be generally classified into two categories: model-based methods and preprocessing-based methods. Model-based methods such as multi-condition training [51] and data augmentation [27] aim to enhance the noise-robustness of the ASR models themselves. The preprocessing based methods adopt a speech enhancement (SE) scheme as the frond-end of ASR. SE aims to separate clean speech from background noises. Stationary noise (i.e., remains constant for a long time) can be adequately removed with traditional signal processing-based SE techniques, such as spectral subtraction [2], minimum mean-square error (MMSE) estimator [16], and Wiener filtering [34]. However, ambient noise usually continuously varies over time (i.e., non-stationary) in realistic acoustic environments, which leads to the failure of traditional signal processing methods.

Recently, deep learning-based SE methods have demonstrated their superiority over traditional signal processing based methods in term of tackling non-stationary ambient noise. These SE methods use a deep neural network to

directly predict clean speech [37][63][44] or to learn a mask which maps noisy speech to clean speech [59][40][4]. The mask learning-based SE methods have matured considerably over the last decade. To further improve speech quality and intelligibility, additional data modalities are introduced to guide the mask estimation process, such as ultrasound [68][15] and visual image [17]. However, it is not trivial to integrate these SE methods with a streaming ASR system. Though several state-of-the-art deep learning based SE methods have been proposed to enhance noisy speech recently, such as FullSubNet+ [7] and DeepFilterNet [49], they are still not streaming in nature. Besides, these deep learning-based SE methods ignore the diversity of acoustic noises in different environments. This limits their generalization to new environments with different acoustic noise types. Different from these methods, AdaStreamLite interacts with the surroundings and include the acoustic environment information to optimize the inference process of the streaming E2E ASR model. Therefore, AdaStreamLite can adapt to new environments with different noise types.

2.3 Speaker[1] Representation Learning

Speaker recognition (SR) [5] is to discriminate different speakers based on their utterances, which includes speaker verification, speaker identification and speaker diarization sub-tasks. Even if utterances are from the same speaker, they can still be significantly distinct from each other in both time and frequency domains, due to multiple factors such as verbal content, emotion, and health state. How to differentiate different speakers is one of the core problems in the SR field. Currently, the state-of-the-art SR methods [1] train a deep neural network to learn a representation space, where speech representations from the same speaker aggregate closely and those from different speakers are away from each other. The widely-used deep neural network architectures for SR include TDNN [52], ResNet [25], ECAPA-TDNN [14] and MFA-Conformer [69]. To enhance the discrimination of speaker representations, similarity-based loss functions (e.g., triplet loss [41], ge2e loss [58] and am-centroid loss [62]) and classification-based loss functions (e.g., aam-softmax loss [12] and real am-softmax loss [32]) are used to optimize the representation learning process. We are inspired by these SR works to learn acoustic environment representations w.r.t different environments with dramatic different noise types. AdaStreamLite can thus leverage environment representations to capture the characteristics of different types of non-stationary ambient noises.

3 BACKGROUND AND INITIAL INVESTIGATION

In this section, we explain the difference between streaming and non-streaming speech recognition. We also investigate the feasibility of modeling non-stationary acoustic environments.

3.1 Streaming Speech Recognition versus Non-streaming Speech Recognition

Generally, an end-to-end speech recognition system takes as input a sequence of acoustic feature vectors (FBanks, MFCCs, etc.) extracted from a speech signal, where an acoustic feature vector is also called a frame. The system predicts textual tokens (letters, words, Chinese characters, etc.) based on speech context information involved in the feature sequence. The biggest difference between streaming and non-streaming speech recognition lies in the length of the context window used to predict a textual token. The context window for streaming speech recognition is usually restricted to a limited number of chunks, where each chunk consists of 65 $\sim$ 265ms speech. In contrast, the context window for non-streaming speech recognition is the entire feature sequence.

Fig 1 shows an example, where a 0.74s speech signal contains the word *'hello'* and the textual token *'[]'* represents a blank symbol. Suppose 64 frames (acoustic feature vectors) are extracted from the speech signal. To predict the textual token *'o'*, the streaming speech recognition model only accesses two chunks (i.e., the current chunk and the previous one) consisting of 32 frames in total, while the non-streaming speech recognition model uses all 64 frames. This difference is hold for every predicted textual tokens. When a chunk is processed, the

[1]The speaker here refers to human speaker.

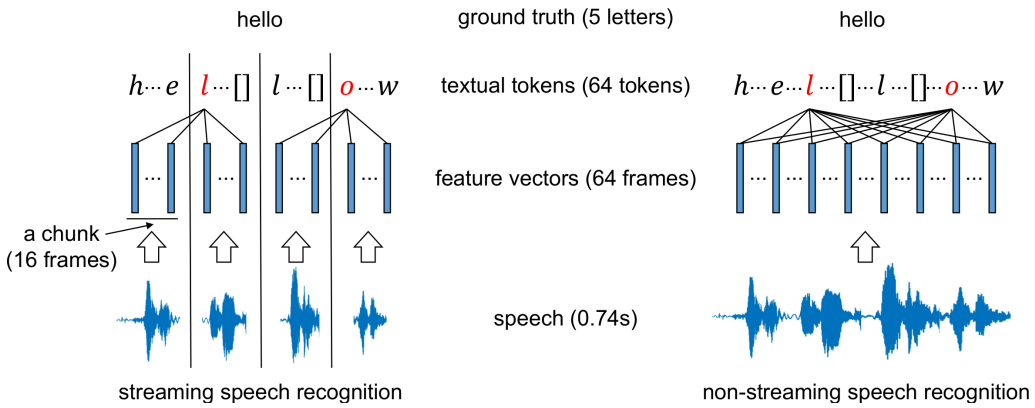


Fig. 1. The comparison between streaming and non-streaming speech recognition.

streaming speech recognition model immediately outputs corresponding textual tokens based on the current chunk and a few previous chunks. The states of these previous chunks are recorded in cache, so they do not need to be computed again. Therefore, the streaming speech recognition model has the ability to provide real-time recognition results for users, when transcribing long speech input (e.g., a voice message with a duration of 5 minutes). To ensure the streaming nature and real-time feedback, we must avoid introducing excessive computations into the inference process of a streaming speech recognition model when designing the environment adaptation mechanism.

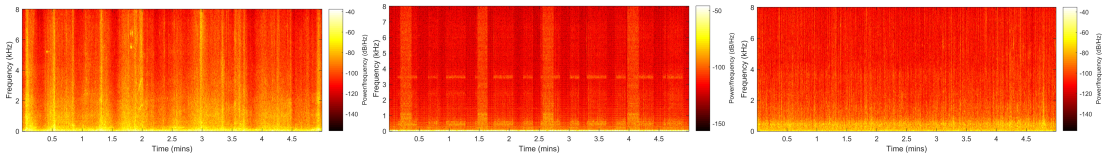
3.2 Non-stationary Ambient Noise

The speech signal processing community generally classifies real-world noises into two categories: stationary noise and non-stationary noise. The statistical properties of the stationary noise (e.g., white noise) remain constant over time. In contrast, the statistical properties of non-stationary noise vary over time. Ambient noise is the mixture of all noises in a specific environment and thus is non-stationary noise. In this paper, we focus on the impact of ambient noise on streaming speech recognition. It is extremely difficult to mathematically model the ambient noise based on statistical properties. Intuitively, different ambient noises lead to significantly distinct auditory effects. For example, a highway environment contains the sound of car horn, the sound of vehicle engine, etc., while the acoustic environment of a laundry room contains the sound of flowing water, the sound of washing machines, etc. We collect audios with the duration of 5 minutes from three different acoustic environments, i.e., a highway, a laundry room and a coffee house. We extract time-frequency spectrograms from these audios. Fig 2 shows their time-frequency spectrograms, where the x-axis and y-axis represent the time and frequency dimensions, respectively. We can observe that different acoustic environments have distinguishable time-frequency patterns. Also it is very difficult to use a mathematical model to characterize the properties of the acoustic noise in an environment.

4 ADASTREAMLITE DESIGN

4.1 Overview

AdaStreamLite exploits the built-in microphone of a smartphone to collect speech signals from users without requiring additional hardware. Besides, our system does not require users to change their smartphone usage habits. Therefore, AdaStreamLite is friendly to smartphone users. As shown in Fig 3, the AdaStreamLite system



(a) Highway. (b) Laundry room. (c) Coffee house.

Fig. 2. Time-frequency spectrograms of different ambient noises.

activates two light-weight threads to conduct two interactive tasks, respectively: encoding the current acoustic environment and recognizing the speech input from a user.

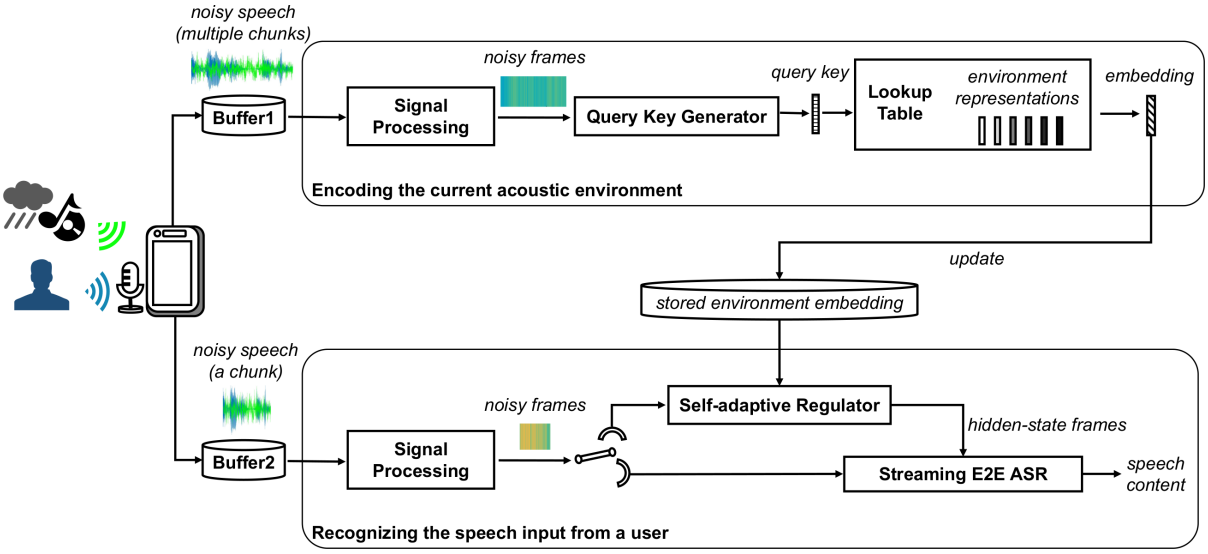


Fig. 3. The work flows and functional components of AdaStreamLite.

Encoding the current acoustic environment. We extract the time-frequency information of ambient noise from noisy speech to improve the inference process of the streaming ASR component. To enhance the generalization of AdaStreamLite to unseen ambient noises, we propose to represent new acoustic environments with enrolled ones. Therefore, we create a representation lookup table, which memorizes environment representations with respect to several known acoustic environments. AdaStreamLite maps the noisy speech from the user to a query key (i.e., a low-dimensional vector). The query key is used to aggregate the environment representations in the lookup table to represent the current acoustic environment. More specifically, the representation of the current acoustic environment (we name it as embedding) is the weighted average of the enrolled environment representations.

Recognizing the speech input from a user. AdaStreamLite provides both the general and environment-adaptive streaming E2E speech recognition services. The user of AdaStreamLite decides which running mode to use. When a user chooses the environment-adaptive mode, AdaStreamLite incorporates the ambient noise information into the inference of the streaming E2E ASR component to improve recognition performance. The

adaptive regulator receives the noisy speech input, reads the stored environment embedding, fuses them to create a mask used for calibration, and generates the calibrated hidden-state frames. The streaming E2E ASR model predicts textual tokens based on these hidden-state frames.

Each task reads noisy speech from an independent memory buffer in a streaming manner. The sizes of these two memory buffer are different. As discussed in Section 3.1, the streaming ASR component takes a chunk (65 ∼ 265ms) as input. We found that more sound data can help better depict the time-frequency pattern of ambient noise, so the environment encoding component accumulates multiple chunks (5 ∼ 10s in total) to create the environment embedding. The generated environment embedding is stored and periodically updated. Note that this encoding process does not affect the streaming nature of the system. When the environment encoding thread is first activated and before the embedding is ready, the system still adopts the general mode to ensure the streaming service is interrupted. Also, the acoustic environment encoding thread is activated only when the user needs it. This avoids unnecessary calculations under quiet environments.

4.2 Speech Signal Processing

AdaStreamLite uses the microphone to capture noisy speech and converts it to a discrete digital signal which is a mixture of user speech and ambient noise. Prior speech enhancement works [7][49] show that it is easier to separate speech from noise in the frequency domain than the time domain. Therefore, we use a signal processing module to extract frequency-domain acoustic features from noisy speech, as shown in Fig 4.

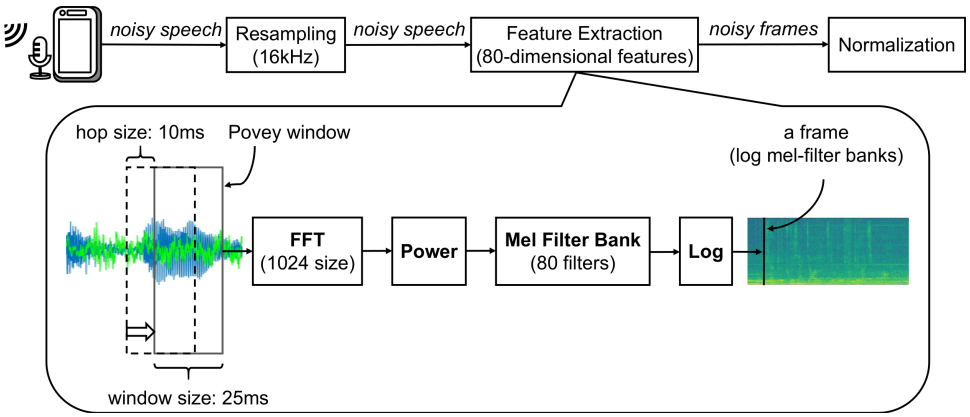


Fig. 4. The signal processing module.

Humans can hear sounds in the frequency range of 20Hz to 20kHz. Most acoustic applications on smartphones use 44.1kHz sampling rate as the default configuration to record sounds. This means one second of sound recordings contain 44, 100 acoustic samples. However, the daily speech produced by humans is mainly concentrated in the frequency range of 20Hz to 4kHz. Therefore, we only focus on the ambient noises in the same frequency range as speech, because ambient noises outside of this range can be easily removed by a low-pass or high-pass filter. We use a sampling frequency of 16kHz instead of 44.1kHz to collect speech from users, so that we can ensure the speech quality and meanwhile reduce the floating point operations (FLOPs) of discrete Fourier transform by approximately 2.76×. In the feature extraction process, the speech signal is firstly divided into short-time segments with a 25ms sliding window and a 10ms overlapping. We use the Povey window to alleviate spectrum leakage [38]. The Povey window is a set of coefficients $\{w[n]\}_{n=0}^{N-1}$, where $N = 0.025 \times 16,000 = 400$ is the number

of sampling points within the window. The coefficient $w[n]$ is calculated by

$$w[n] = \frac{1}{2}[1 - \cos(\frac{2\pi n}{N-1})]^{0.85}. \tag{1}$$

We perform Fast Fourier Transform (FFT) with the size of 1,024 on each window, which provides 512 frequency bins with sufficient resolution. We then calculate the power spectrum of the speech signal, which reflects the energy distribution over different frequency bins. To further reduce the computation of AdaStreamLite, we use a mel-scaled filter bank with 80 triangular filters to decrease the number of the frequency bins from 512 to 80. The logarithm operation is used to match the nonlinear perception property of human ears. We use the mean and variance normalization to mitigate the effect of stationary noise (e.g., white noise). The speech signal is converted to a sequence of 80-dimensional acoustic feature vectors (we name them as speech frames for convenience), which contains rich time-frequency information.

4.3 Lightweight Neural Network Design

Considering there are limited computational resources on a smartphone, we design two lightweight neural networks, i.e., FrameEncoder and SeqPooling. The FrameEncoder sub-network processes acoustic feature sequences at the frame level. The SeqPooling sub-network compresses all frames of a hidden-state feature sequence output by FrameEncoder into a compact feature vector.

4.3.1 FrameEncoder Sub-network. We design FrameEncoder to capture both short-term and long-term variations over an acoustic feature sequence. FrameEncoder is stacked by one down-sampling block and three Conformer encoders [24], which takes as input a sequence of 80-dimensional log Mel-filter banks and outputs a 32-dimensional hidden-state feature sequence. The down-sampling block is used to reduce the length of the input feature sequence, which effectively decreases the computational cost. As shown in Fig 5, a Conformer encoder is sequentially stacked by a first feed-forward (FFN) module, a multi-head self-attention (MHSA) module [57], a convolution (CONV) module, a second FFN module, and a final LayerNorm layer. Residual connections are also used for the Conformer encoder to avoid gradient vanish. The MHSA module is good at modeling global context, while the Conv module focuses on fine-grained local context.

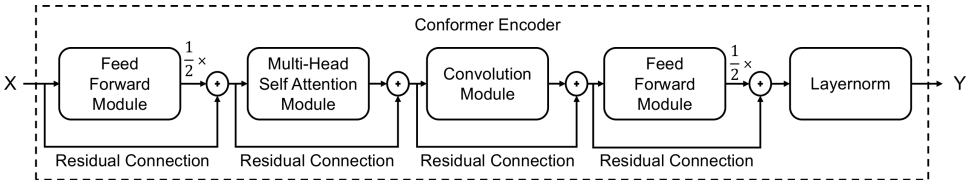


Fig. 5. The structure of a Conformer encoder.

Let $X \in \mathbb{R}^{D \times T}$ refer to a feature sequence with a varying length, where D indicates the feature dimension and T denotes the number of frames. For X input to a Conformer encoder, the output feature sequence $Y \in \mathbb{R}^{D \times T}$ is calculated as

$$\begin{aligned} X' &= X + \frac{1}{2}FFN(X); \\ X'' &= X' + MHSA(X'); \\ X''' &= X'' + CONV(X''); \\ Y &= LayerNorm(X''' + \frac{1}{2}FFN(X''')). \end{aligned} \tag{2}$$

Table 1. The blocks of the FrameEncoder sub-network.

	Down-sampling Block	Conformer Encoder × 3			
		FFN1	MHSA	CONV	FFN2
Configures	Conv2d: $3 \times 3, 32$ Conv2d: $3 \times 3, 32$ Flatten Linear: 608×32	Linear: 32×128 Linear: 128×32	Linear: 32×32 Linear: 32×32 Linear: 32×32 Linear: 32×32 Linear: 32×32	Conv1d: $1, 64$ Conv1d: $31, 32$ Conv1d: $1, 32$	Linear: 32×128 Linear: 128×32

Table 1 depicts the details of a FrameEncoder sub-network, where 'Linear' represents a fully connected layer, 'Conv1d' refers to a 1D convolution layer and 'Conv2d' denotes a 2D convolution layer. Additionally, Batch Normalization [29], Switch activation [46] and Dropout [53] are used for FrameEncoder to accelerate model convergence and avoid over-fitting.

4.3.2 SeqPooling Sub-network. We design a SeqPooling sub-network to aggregate the hidden-state feature sequence derived from the FrameEncoder sub-network. The frames in a hidden-state feature sequence are of unequal importance. Therefore, we use an Attentive Statistics Pooling (ASP) mechanism [42] with learnable parameters to emphasize those more important frames. Let $H \in \mathbb{R}^{D \times T}$ denote a hidden-state feature sequence, where D is the feature dimension and T is the number of frames. For the t-th frame $h_t \in \mathbb{R}^{D \times 1}$ in $H = \{h_1, h_2, \cdots, h_T\}$, the attention score $e_t \in \mathbb{R}$ is calculated by

$$e_t = \mathbf{v} f(\mathbf{W} h_t + \mathbf{b}) + k, \tag{3}$$

where $\mathbf{v} \in \mathbb{R}^{1 \times D}$, $\mathbf{W} \in \mathbb{R}^{D \times D}$, $\mathbf{b} \in \mathbb{R}^{D \times 1}$, $k \in \mathbb{R}$ are trainable parameters, and $f(\cdot)$ is the Tanh activation function. The attention score e_t is further normalized over all frames as follows

$$\alpha_t = \frac{\exp e_t}{\sum_{i=1}^{T} \exp(e_i)}. \tag{4}$$

The weighted mean vector $\widetilde{\mu} \in \mathbb{R}^{D \times 1}$ and the weighted standard deviation vector $\widetilde{\sigma} \in \mathbb{R}^{D \times 1}$ are calculated by

$$\begin{aligned} \widetilde{\mu} &= \frac{1}{T} \sum_{t=1}^{T} \alpha_t h_t, \\ \widetilde{\sigma} &= \sqrt{\sum_{t=1}^{T} \alpha_t h_t \otimes h_t - \widetilde{\mu} \otimes \widetilde{\mu}}, \end{aligned} \tag{5}$$

where $\otimes$ denotes the Hadamard product. The output of the ASP model is obtained by concatenating $\widetilde{\mu}$ and $\widetilde{\sigma}$. We adjust the dimension of the concatenation vector by a linear module. Table 2 shows the details of the SeqPooling sub-network. Through the above calculations, the SeqPooling sub-network maps a sequence of 32-dimensional hidden-state feature sequence to a 256-dimensional feature vector.

4.4 Encoding Acoustic Environment

As discussed in Section 3.2, it is not trivial to depict a non-stationary acoustic environment based on the statistical properties of ambient noise, but we observed that different ambient noises have distinguishable time-frequency patterns. To avoid imposing excessive computations on the streaming speech recognition component, the acoustic environment model should be lightweight. Inspired by prior works [52][14][69] in the speaker recognition

Table 2. The layers of the SeqPooling sub-network.

	ASP Module	Linear Module
Configures	Conv1d: 32×80 Stacking Conv1d: 240×128 Conv1d: 128×80	Linear: 160×256 Linear: 256×256

field, we encode the time-frequency pattern of the ambient noise as a compact feature vector and we name it environment representation.

4.4.1 Learning Representations for Different Acoustic Environments. We design an environment representation extractor and train it based on acoustic data collected from different acoustic environments, such as train station, office room, and restaurant. As shown in Fig 6, the environment representation extractor consists of a FrameEncoder module, a SeqPooling module, and a Linear output layer with 256 nodes, which maps an ambient sound signal to a 256-dimensional feature vector. To train the extractor, we use ambient sound signals collected from 11 different acoustic environments $D1-D11$ shown in Table 3 (Section 5.3) as training data. The ground truth of an ambient sound signal is a one-hot category label, which represents a specific acoustic environment. During the training process, a classifier receives environment representations as input and predicts their categories, which is parameterized by weight vectors $\{W_1, W_2, \cdots, W_M\}$, where $M = 11$ is the number of environment types involved in the training data. The smartphone microphone collects ambient noise and user speech at the same time, so AdaStreamLite practically extracts environment representations from noisy speech. To learn speech-independent environment representations, we perform data augmentation on training data by mixing ambient noise with speech at a possibility of 50%.

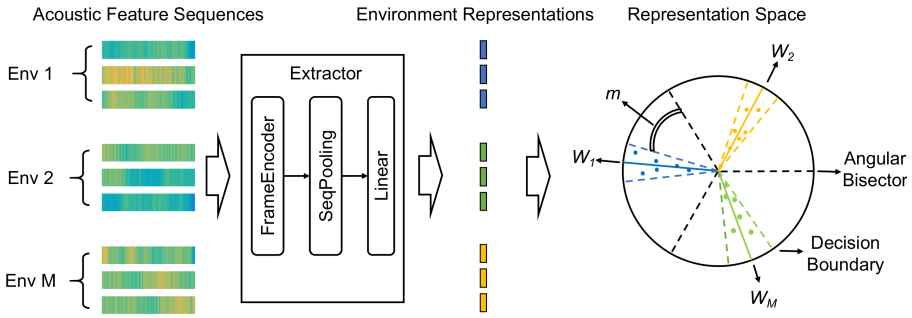


Fig. 6. Acoustic environment representation learning

We use the additive angular margin softmax (AAM-Softmax) loss [12] to enhance the discrimination of environment representations by increasing the inter-class distance and meanwhile decreasing the intra-class distance in the environment representation space, as shown in Fig.6. The AAM-Softmax loss for N training samples is calculated as

$$l_{AAM-Softmax} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s(\cos(\theta_{i,y_i}+m))}}{e^{s(\cos(\theta_{i,y_i}+m))}+\sum_{j=1,j\neq y_i}^{n}e^{s\cos\theta_{i,j}}}, \tag{6}$$

where s denotes a scale factor, m is the margin penalty, y_i refers to the ground truth (class label) w.r.t the i-th ambient sound, and $\theta_{i,j}$ is the angle between the i-th environment representation and the j-th class weight vector. We set $s = 32$ and $m = 0.2$.

To investigate whether the representation extractor has the ability to encode different acoustic environments, we collect ambient sounds from 6 acoustic environments in real world, including *basketball court*, *concert hall*, *server room*, *laundry room*, *subway station* and *school canteen*. Note that these acoustic environments do not appear in training data. We visualize their representations by the t-SNE algorithm [55]. Fig 7 shows the visualization results. We can see that these environment representations have the property of intra-class compactness and inter-class discrimination. This demonstrates that the representation extractor can capture the characteristics of acoustic environments.

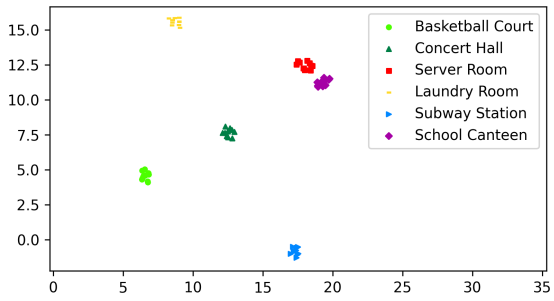


Fig. 7. The environment representations of 6 acoustic environments in the real world.

4.4.2 Constructing the Environment Representation Lookup Table. To further improve the generalization of AdaStreamLite to unseen acoustic environments, we propose to represent a new acoustic environment with known acoustic environments. This is consistent with the observations of realistic environments. For example, the acoustic environment of an open-air basketball court near a road can be regarded as the combination of the *basketball court* and *road* acoustic environments.

Neural Turing Machine (NTM) [23] is an artificial neural network model with external memory resources, which has been successfully used for various of speech processing tasks, such as speaker adaptation [48] and speech synthesis [60]. We construct a NTM based lookup table to memorize the representations of 11 acoustic environments obtained in Section 4.4.1. Since each acoustic environment has multiple ambient noise recordings, we average all representations belonging to the same acoustic environment.

4.4.3 Generating Embedding for the Current Acoustic Environment. We design a query key generator to map a noisy speech signal from the microphone to a query key, as shown in Fig 8. The structure of the query key generator is the same as the representation extractor. We use the query key to aggregate all environment representations in the lookup table based on the read operation of NTM.

There are 11 environment representations in the lookup table, $\{E_1, E_2, \cdots, E_{11}\}$, which correspond to 11 different types of acoustic environments. The read operation of NTM is based on attention mechanism. The query key is a low-dimensional vector $Q \in \mathbb{R}^{256\times 1}$. The attention score s_i for the i-th environment representation in the lookup table is calculated by

$$s_i = \frac{1}{\sqrt{256}} Q \odot E_i = \frac{1}{16} Q \odot E_i, \tag{7}$$

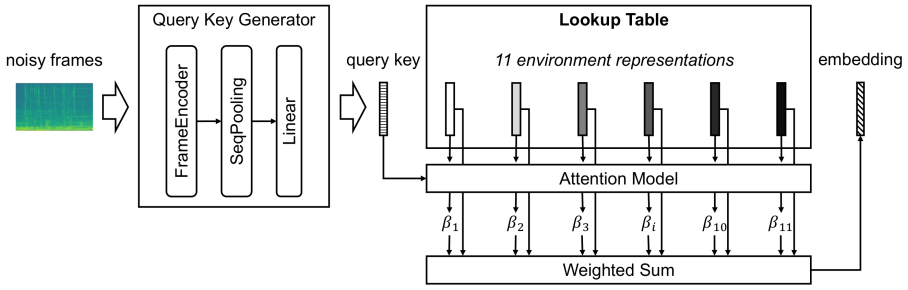


Fig. 8. Querying the lookup table.

where $\odot$ denotes the dot product between two vectors. These attention scores are normalized by

$$\beta_i = \frac{\exp(s_i)}{\sum_{j=1}^{11} \exp(s_j)}. \tag{8}$$

The environment embedding of the current acoustic environment $E_{curr} \in \mathbb{R}^{256\times 1}$ is calculated as

$$E_{curr} = \sum_{i=1}^{11} \beta_i E_i, \tag{9}$$

which is used for environment-adaptive end-to-end streaming speech recognition.

4.5 Recognizing User Speech

4.5.1 Streaming Speech Recognition Model. We use a U2++ model [65] as the streaming E2E ASR component of AdaStreamLite, which has a hybrid CTC/Attention architecture and adopts a dynamic chunk-based attention strategy. As shown in Fig 9, the U2++ model consists of a shared encoder, a CTC decoder, and an attention decoder. We stack 12 Conformer blocks [24] to act as the shared encoder. The CTC decoder is a simple linear layer, whose dimension is determined by the size of lexicon used for speech recognition. The attention decoder is comprised of 3 Transformer blocks [57].

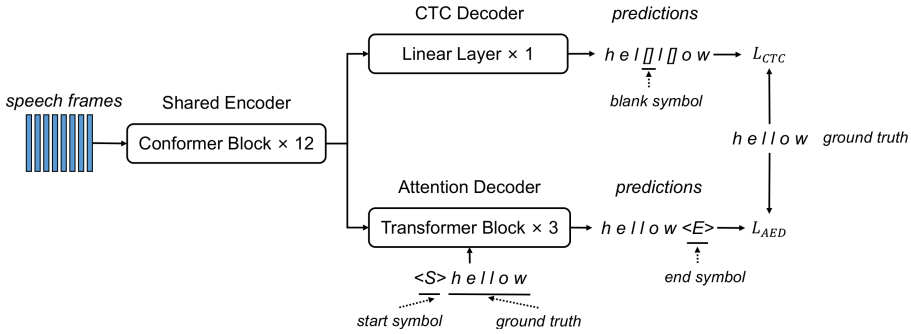


Fig. 9. The structure of the streaming speech recognition component of AdaStreamLite.

The CTC mechanism allows the speech recognition model to predict a blank symbol *[]* in a speech frame, which avoids the forced alignment between input speech frames and output textual tokens. For example, the

CTC decoder is allowed to predict '*hel[][]low*' or '*hel[]lo[]w*' for the ground truth '*hellow*'. The attention decoder leverages both speech frames and previous textual tokens to predict the current textual token, which introduces extra language information into the inference process. In the U2++ model, streaming speech recognition is provided by the shared encoder and the CTC decoder, while the attention decoder is used for calibration.

The loss functions L_{CTC} and L_{AED} are used to train the U2++ model. Let K be the size of lexicon, $\mathbf{x} = \{x_1, x_2, \cdots, x_T\}$ refers to an input frame sequence of Length T, and $\mathbf{z} = \{z_1, z_2, \cdots, z_i, \cdots, z_T; z_i \in \mathbb{R}^K\}$ represents the output sequence of the CTC decoder w.r.t $\mathbf{x}$. A textual token sequence $\pi = \{\pi_1, \pi_2, \cdots, \pi_T\}$ is a correct path, when the ground truth $\mathbf{y}$ of $\mathbf{x}$ can be obtained by removing the blank symbols and merging continuously repetitive textual tokens from π. The CTC loss L_{CTC} is defined as

$$L_{CTC} = -\log p(\mathbf{y}|\mathbf{x}) = -\log(\sum_{\pi \in \mathcal{B}^{-1}(\mathbf{y})} p(\pi|\mathbf{x})) = -\log(\sum_{\pi \in \mathcal{B}^{-1}(\mathbf{y})} \prod_{t=1}^{T} z_t^{\pi_t}), \tag{10}$$

where $\mathcal{B}^{-1}$ contains all correct paths to $\mathbf{y}$ and $z_t^{\pi_t}$ denotes the probability of the textual token π_t in z_t. The attention encoder-decoder loss L_{AED} is defined as the likelihood of the ground truth as follows

$$L_{AED} = -\log p(\mathbf{y}|\mathbf{x}) = -\log p(y_1|\mathbf{x}) \prod_{i=2}^{l} p(y_i|y_1, y_2, \cdots, y_{i-1}, \mathbf{x}), \tag{11}$$

where $\mathbf{x}$ is the input sequence and $\mathbf{y} = \{y_1, y_2, \cdots, y_l\}$ is the ground truth of length l.

4.5.2 Environment-adaptive Mechanism. We design a self-adaptive regulator to improve the robustness of AdaStreamLite to noisy environments in the real world. The self-adaptive regulator leverages the time-frequency information of the current acoustic environment to optimize the inference process of the streaming E2E ASR component. Fig 10 shows the workflow of the self-adaptive regulator. To reduce the computational complexity, the self-adaptive regulator enhances the frame output by a down-sampling block, instead of the original acoustic feature sequence extracted from the input speech. Let $S_{noi} \in \mathbb{R}^{F \times T}$ denote the frames, where T is the number of the frames (the length of the hidden-state feature sequence) and F is the feature dimension. The self-adaptive regulator fuses the speech and acoustic environment information by concatenating the recorded environment embedding with each frame of S_{noi}. The concatenation result is converted to a real-valued mask $\mathcal{M} \in \mathbb{R}^{F \times T}$ with a simple two-layer fully connected neural network ($768 \times 512, BatchNorm, LeakyReLu, 512 \times 512$) and a Sigmoid non-linear activation function, where $\mathcal{M}_{f,t} \in [0, 1]$. The outputs of the self-adaptive regulator are enhanced frames $S_{enh} = S_{noi} \otimes M$, where $\otimes$ is the Hadamard product. The enhanced frames are input into the ASR component to predict textual tokens. The query key generator, the self-adaptive regulator, and the streaming E2E ASR component are optimized jointly to minimize a hybrid speech recognition loss L_{ASR} in an end-to-end manner. Let $\mathbf{E}$, $\mathbf{D}_{ctc}$ and $\mathbf{D}_{aed}$ be the shared encoder, CTC decoder and attention decoder of the streaming E2E ASR model, respectively, the loss function L_{ASR} is then defined as

$$L_{ASR} = L_{CTC}(\mathbf{D}_{ctc}(\mathbf{E}(S_{enh})), y) + \lambda L_{AED}(\mathbf{D}_{aed}(\mathbf{E}(S_{enh})), y), \tag{12}$$

where y is the ground truth, L_{CTC} is the Connectionist Temporal Classification (CTC) loss [20][21], L_{AED} is the Attention-based Encoder-Decoder (AED) loss [10][6][9], and λ is a balancing coefficient.

5 SYSTEM IMPLEMENTATION

5.1 Software and Hardware

We use *PyTorch*, one of popular deep learning frameworks, to implement and train the deep neural networks involved in the functional components of AdaStreamLite. We process speech signals that are used as training data with the *librosa* python package. All of the training processes run on a high-performance server equipped with

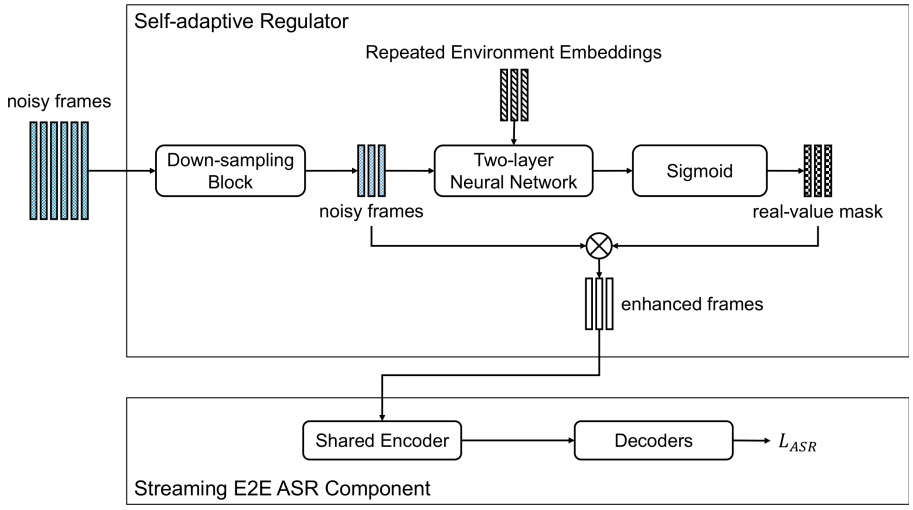


Fig. 10. The workflow of the self-adaptive regulator.

GTX Titan-V and RTX 2080-Ti GPUs. To further reduce the computational resource consumption of AdaStreamLite, we use the post training dynamic quantization tool provided by *PyTorch* to compress the computation-intensive fully connected layers involved in AdaStreamLite, which maps their weight parameters from 32-bit float point number to 8-bit integer number. We use Java and Android technologies to develop a prototype of AdaStreamLite. We deployed the prototype system on multiple mainstream smartphones.

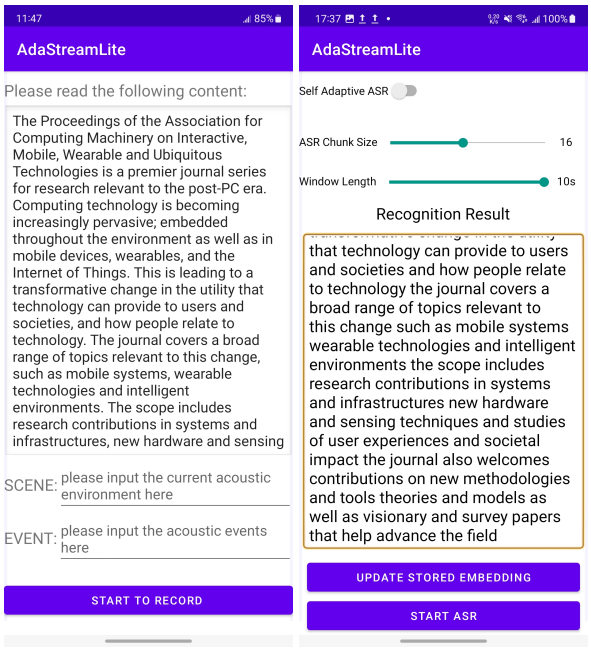
5.2 Graphical User Interface

We design an easy-to-use graphical interface for users, as shown in Fig 11. Fig 11(a) is a data collection tool based on Android. We invite volunteers to use this tool to collect speeches in real-world acoustic environments. Fig 11(b) shows the graphical user interface of AdaStreamLite. Users can decide whether to activate the self-adaptation function, set a desirable chunk size, and adjust the window size for ambient sound collection.

5.3 Data Preparation

Thousands of hours of speech data are essential to constructing a practical streaming ASR system. We use publicly available data resources from the speech signal processing community to develop our AdaStreamLite system, which are listed as follows:

- **LibriSpeech**: LibriSpeech [43] is a corpus derived from audio books, which consists of approximately 1000 hours of English speech. There are three training datasets in the LibriSpeech, i.e., *librispeech-100* (100 hours and 251 speakers), *librispeech-360* (360 hours and 921 speakers), and *librispeech-500* (500 hours and 1,166 speakers). The first two subsets are of high recording quality, while the third subset is collected under unconstrained conditions. AdaStreamLite uses these training sets to develop its streaming English speech recognition component. The test sets of LibriSpeech are used to evaluate AdaStreamLite in our experiments.
- **WenetSpeech and Aishell-1**: WenetSpeech [64] is a large-scale multi-domain Mandarin corpus, which contains 22,400+ hours speech data in total (10,000+ hours labeled speech, 2,400+ hours weakly labeled speech, and 10,000 hours unlabeled speech). These speech data are collected under multiple speaking



(a) Data Collection Tool. (b) AdaStreamLite.

Fig. 11. The graphical user interface.

Table 3. The acoustic environments in the DEMAND dataset.

Training	Office (D1), Hallway (D2), Meeting (D3), Station (D4), Restaurant (D5), Traffic (D6), Town Square (D7), Cafe (D8), Metro (D9), Bus (D10), Car (D11)
Test	Washing (D12), Kitchen (D13), Living (D14), Field (D15), River (D16), Park (D17)

styles, topics, real-world scenarios, and noisy conditions. Aishell-1 [3] is a high-quality Chinese Mandarin corpus, which contains 400 speakers with different accents and 170 hours of speech data. All the speech data are recorded by a high-fidelity microphone in a quiet indoor environment. AdaStreamLite uses the 10,000+ hours labeled speech of WenetSpeech and the training set of Aishell-1 to implement Chinese based streaming speech recognition function. The test set of Aishell-1 is used to evaluate AdaStreamLite.

- **DEMAND**: DEMAND [54] collects 17 types of ambient noises from daily life scenes and natural environments, such as meeting, station, park, and river. These noise data preserve the original noise power. We use 6 different types of acoustic noises in DEMAND for testing and the remainders for training, as listed in Table 3.

5.4 Model Training

Several components of AdaStreamLite are based on deep neural network, including the environment representation extractor, the query key generator, the self-adaptive regulator and the streaming E2E ASR model. It usually

requires several weeks or months to train a speech recognition model from scratch when the training speech data are thousands of hours. To reduce training load and meanwhile maintain high recognition performance, we initialized the weight parameters of the streaming E2E ASR component in AdaStreamLite with a pre-trained model. For English speech recognition, we use the U2++ model pre-trained with the collection of *librispeech-100*, *librispeech-360*, and *librispeech-500*. For Chinese speech recognition, we used the U2++ model pre-trained with the 10, 000+ hours labeled speech of WenetSpeech. Besides, we also initialize the query key generator with the weight parameters of the learned environment representation extractor. To generate noisy speech data used for training the components of AdaStreamLite, we mix the clean speech data (the training set of Aishell-1 or the collection of *librispeech-100* and *librispeech-360*) with the ambient noises from DEMAND (D1-D11) at different signal-to-noise ratio levels from 0dB to 30dB. We fine tune the query key generator and the streaming E2E ASR model with a learning rate of $2e-5$. The environment representation extractor and the self-adaptive regulator are trained with a learning rate of $1e-3$.

5.5 System Configures

AdaStreamLite receives speech in the presence of noise from a user chunk by chunk. The default chunk size is set to 16 frames (i.e., 185ms). In the inference process of a streaming ASR model, the history information of previous chunks can be leveraged to improve recognition performance of the current chunk. However, reserving all previous chunks for AdaStreamLite will lead to unacceptable system delay and memory consumption when dealing with long speech input (e.g., more than 5 minutes). To make a trade-off between recognition performance and inference latency, AdaSteamLite only memorizes the hidden states from the last 20 seconds. Also the ambient noise varies over time. AdaStreamLite therefore updates the stored environment embedding periodically (e.g., every 10s). In our experiments, we also investigate the impact of these configures on performance.

6 SYSTEM EVALUATION

In order to evaluate the proposed AdaStreamLite system under realistic acoustic environments, we recruited 10 volunteers aged from 20 to 32, which contain 5 males and 5 females, to participate in our experiments. We focus on the real experience of these volunteers.

6.1 Textual Materials

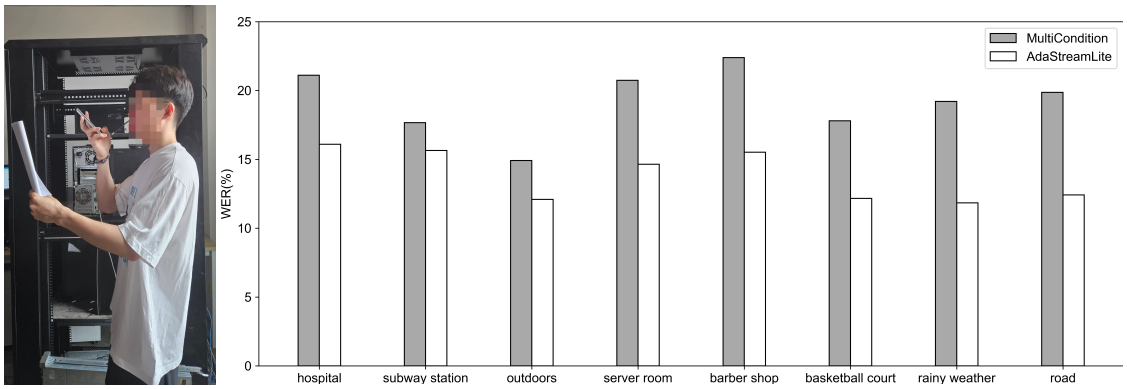
To evaluate AdaStreamLite in term of user experience, we prepared diverse textual materials for the volunteers to read. These textual materials include speech commands, newspapers, speeches of famous persons, and textbooks of language learning, which are on diverse topics including device control, information retrieval, news, philosophy of life, and daily conversations. There are 1, 448 different words in the textual materials.

6.2 Evaluation Metric

Following prior streaming end-to-end speech recognition studies, we use Word Error Rate (WER) to evaluate the recognition performance of AdaStreamLite. The WER metric measures the least number of operations, when a hypothesized word sequence is transformed to the ground truth by three types of basic operations: insertion, deletion and substitution. Let N refer to the number of words in a ground truth sentence. WER is defined as

$$WER = \frac{I + D + S}{N}, \tag{13}$$

where I, D and S denote the number of required insertions, deletions and substitutions, respectively.



(a) The experiment process in a server room.

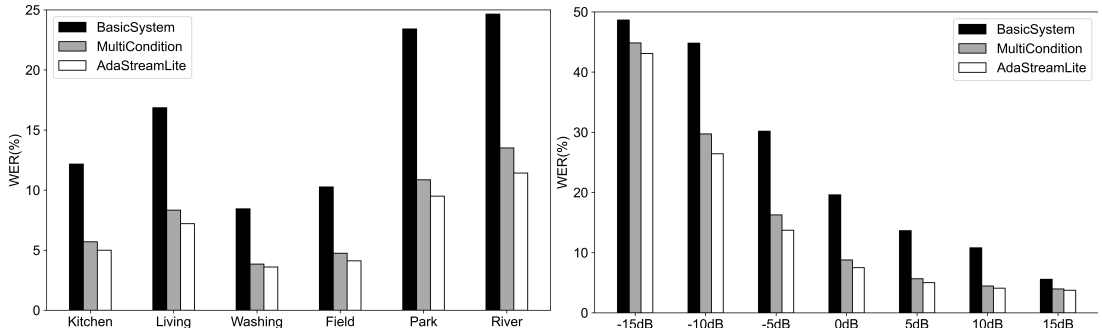
(b) WER (%).

Fig. 12. The WER (%) of AdaStreamLite under realistic noisy environments.

6.3 Overall Performance

6.3.1 Performance in Different Noisy Environments. To investigate the generalization of AdaStreamLite to different types of noisy environments, we asked all participants to use AdaStreamLite in 8 realistic acoustic environments, including hospital, subway station, outdoors (cicadas' chirping), server room, barber shop, basketball court, rainy weather, and road. The participants read the same textual materials for 10 minutes in each environment. The signal-to-noise ratio (SNR) of the speech signal ranges from 5.5dB to 7.5dB, which is quite low for most speech applications. We also developed a robust streaming ASR system (MultiCondition) on the test smartphone for comparison with AdaStreamLite, where a pre-trained U2++ model is fine-tuned by a multi-condition training method. Multi-condition training is the most popular noise-robustness enhancement method at model level in the speech recognition field, which augments training data by adding noises. The MultiCondition system uses the ambient noises collected in different environments ($D1$-$D11$) to perform data augmentation. Fig 12(b) shows the evaluation results. AdaStreamLite obtains an average WER of 13.805% across all environments. Compared to the MultiCondition system, the error rate of AdaStreamLite is decreased by 28.15%. This result demonstrates that dynamic environment adaptation performs better than static noise-robustness enhancement.

Besides, we also evaluated AdaStreamLite with the test sets mentioned in Section 5.3. To generate noisy speech for evaluation, we added ambient noises collected from the 6 acoustic environments $D12-D17$ of DEMAND to the clean test speech, respectively. Each noisy environment is configured with five SNR levels ($-5dB$, $0dB$, $5dB$, $10dB$, and $15dB$), which correspond to the most common conditions in daily life. We ensure that both human speakers and acoustic environments in the test data are not involved in the training data. Besides the MultiCondition system, we additionally developed a basic on-device streaming E2E ASR system (BasicSystem), which directly uses the pre-trained U2++ model without any enhancements. Fig 13(a) shows experiment results. AdaStreamLite achieves the best performance under all tested acoustic environments. Though the streaming ASR model of BasicSystem is trained with 10, 000+ hours speech data collected from most realistic environments, it still does not perform well in the experiment. Especially, the WERs of BasicSystem exceed 23.0% in the *Park* and *River* acoustic environments, which contain ambient noises from natural environments. Compared to the results in Fig 12(b), the difference between MultiCondition and AdaStreamLite is smaller in term of WER. This is because the ambient noises in $D1$-$D11$ are more similar to those in $D12$-$D17$ and the MultiCondition system has seen $D1$-$D11$ during its multi-condition training process. These two observations demonstrate that the diversity of acoustic environments



(a) Different acoustic environments. (b) Different SNR levels.

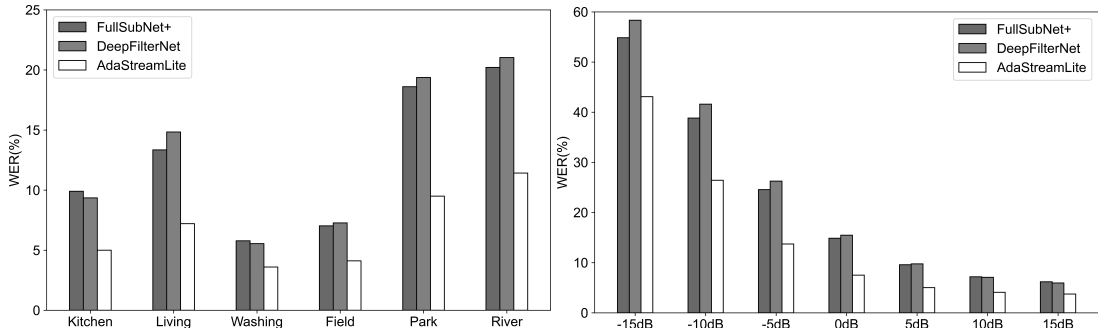
Fig. 13. The WER (%) metric of AdaStreamLite under noisy environments $D12 - D17$.

involved in training data is crucial to the noise-robustness of a streaming ASR system. However, it is impossible to include all real-world acoustic environments in the training data. AdaStreamLite, which dynamically interacts with its surroundings, shows its superiority over the baseline systems. AdaStreamLite achieves WERs lower than 5.0% in the *Kitchen*, *Washing* and *Field* acoustic environments. The WERs of AdaStreamLite are still below 12.0% in the *Park* and *River* acoustic environments.

6.3.2 Performance at Multiple SNR Levels. To investigate the impact of Signal-to-Noise Ratio (SNR) on AdaStreamLite, we set the SNR of test speech data to different levels. The SNR levels for test include $-15dB$, $-10dB$, $-5dB$, $0dB$, $5dB$, $10dB$, $15dB$. For each SNR level, we evaluated AdaStreamLite in 6 different acoustic environments $D12 - D17$ and calculated the averaged WER. Fig 13(b) shows the results. We can see that the WER value dramatically increases with the decrease of SNR. Compared to the two baselines BasicSystem and MultiCondition, AdaStreamLite shows better robustness against low SNR. At the common SNR levels in daily life ($-5dB$, $0dB$, $5dB$, $10dB$ and $15dB$), AdaStreamLite achieves an average WER of 6.8%, which is acceptable for most users. When the SNR level is too low, i.e., $-15dB$, the speech is completely buried in the ambient noise and the WER is the highest (about 50%).

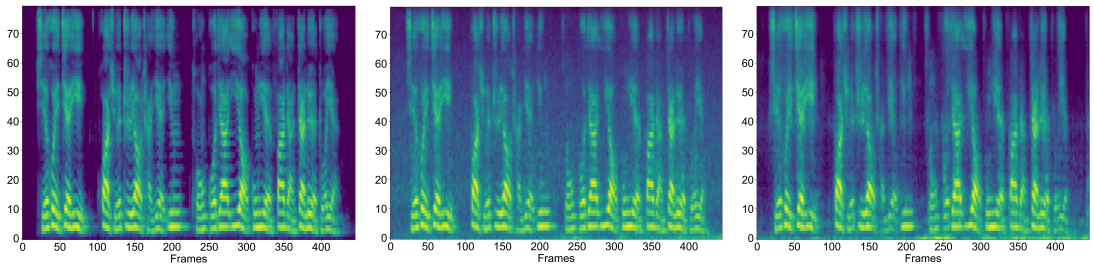
6.3.3 Performance Comparison with the State-of-the-art Systems. Most successful paradigms of noise-robust non-streaming speech recognition use speech enhancement schemes as the frond-end. We compared AdaStreamLite with two state-of-the-art speech enhancement systems, FullSubNet+ [7] and DeepFilterNet [49]. FullSubNet+ is a frequency-domain based SE method, which predicts a complex Ideal Ratio Mask (cIRM) to reconstruct both the amplitude and phase of enhanced speech. To reduce computational cost, FullSubNet+ uses lightweight multi-scale time sensitive channel attention modules and temporal convolutional network blocks as its computational components. DeepFilterNet is a two-stage speech enhancement system, which uses the combination of an equivalent rectangular bandwidth (ERB) filter bank and a deep filtering component, instead of a cIRM, to reconstruct enhanced speech. We used the official open-source codes provided by the authors to implement the FullSubNet+ and DeepFilterNet systems. For a fair comparison, we trained and tested them with the same speech data as those used for AdaStreamLite. The experiments include evaluations under different acoustic environments and SNR levels.

Fig 14(a) and Fig 14(b) show the evaluation results. We can see that AdaStreamLite outperforms the competing systems. These two speech enhancement systems can not reconstruct enhanced speech well based on limited speech context information (a chunk of 16 frames). Fig 15 shows the reconstructed speech spectrograms from



(a) Comparison under different acoustic environments. (b) Comparison at different SNR levels.

Fig. 14. The WER (%) comparison among AdaStreamLite and state-of-the-art systems.



(a) Clean speech. (b) FullSubNet+. (c) DeepFilterNet.

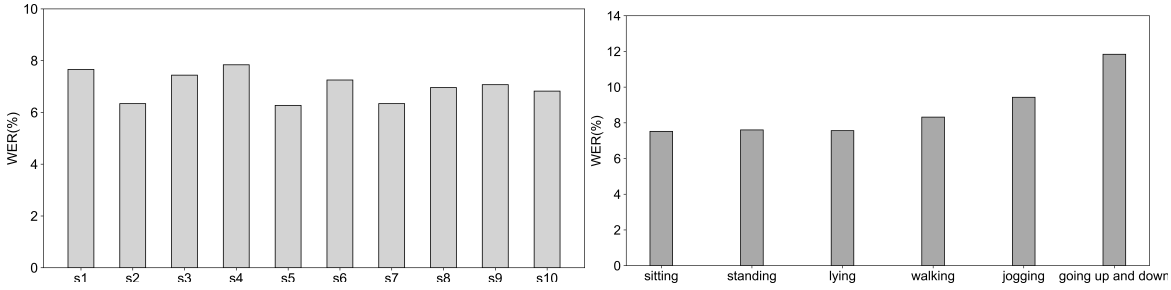
Fig. 15. Reconstructed speech from DeepFilterNet and FullSubNet+.

FullSubNet+ and DeepFilterNet. Compared to clean speech, the reconstructed speech spectrograms lost a large amount of details. We train all components of AdaStreamLite in an end-to-end fashion, so they are jointly optimized to maximize recognition performance based on limited speech context. Besides, different from these systems that do not interact with acoustic environments, AdaStreamLite leverages the information of ambient noise to achieve a better recognition performance.

6.3.4 Ablation Study. To better understand the contribution of each component to AdaStreamLite, we conducted the ablation study and constructed test speech data based on acoustic environments $D12 - D17$. Table 4 lists the experiment results. "No Self-adaptation" represents a noise-robust streaming E2E ASR model optimized by multi-condition training, which achieves the worst performance. This demonstrates that AdaStreamLite can benefit from interacting with its surroundings. "No Environment Embedding" refers to that the self-adaptive regulator only receives noisy speech as input to generate the mask used for calibration. Its performance is worse than AdaStreamLite, which demonstrates the effectiveness of leveraging ambient noise information to optimize the inference process of streaming E2E ASR. In "No Representation Table", we remove the representation lookup table from AdaStreamLite and directly use the environment representation derived from the query key generator to model the current acoustic noise. This leads to performance degradation. Therefore, representing new acoustic environments with known ones can improve the generalization of AdaStreamLite to unseen ambient noise. In summary, all proposed design components contribute to improving the performance of AdaStreamLite.

Table 4. Ablation study.

	Kitchen	Living	Washing	Filed	Park	River
AdaStreamLite	6.452	9.104	4.942	5.500	11.530	13.354
No Representation Table	6.956	9.636	5.100	5.756	12.382	14.288
No Environment Embedding	7.574	10.594	5.340	6.192	13.768	15.948
No Self-adaptation	7.762	10.694	5.425	6.286	13.916	15.168



(a) The WER (%) over different speakers.

(b) The impact of body movements on WER (%).

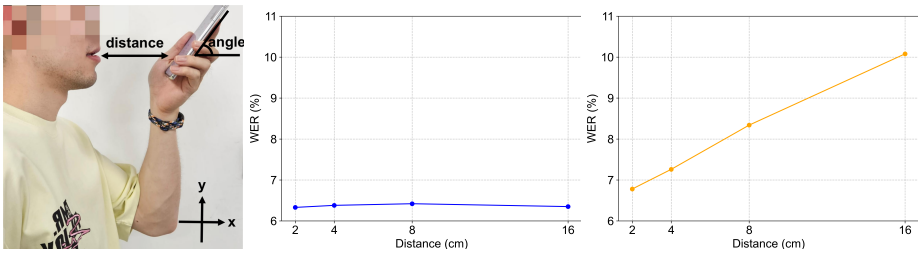
Fig. 16. Robustness of AdaStreamLite.

6.4 Robustness Analysis

6.4.1 Robustness Against Speaker Diversity. There are distinct differences among the physiological characteristics and speaking styles of users, which lead to different acoustic feature patterns even for the same speech content. To investigate the robustness of AdaStreamLite against different speakers, we invited 10 volunteers $s1 - s10$ to participate in our experiment. Each volunteer read the same textual materials following his/her natural speaking habit. We evaluated the WER of AdaStreamLite for these volunteers. As shown in Fig 16(a), the WERs range from 6% to 8%. We found that $s1$, $s3$, $s4$ and $s6$ have noticeable regional accents. The WER difference between any two volunteers is always below 1.5%, which demonstrates the robustness of AdaStreamLite against speaker diversity.

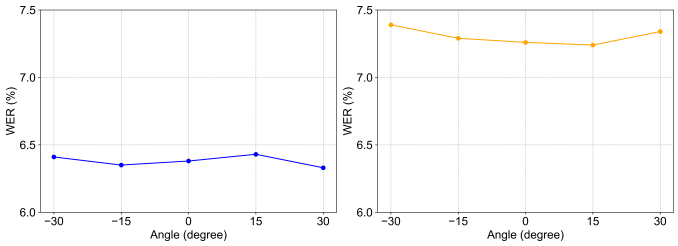
6.4.2 Robustness Against Body Movements. To investigate the impact of body movements on recognition performance, we asked the participants to use AdaStreamLite under different body movement states including sitting, standing, lying, walking, jogging, and going up and down. The participants read the same text material for a duration of 10 minutes. From Fig 16(b), we can see that walking and jogging slightly increase WER, compared to sitting, standing and lying. The participants were panting for breath when they spoke and walked (jogged) at the same time. Heavy breathing can slightly affect the performance. Besides heavy breathing, we found that the smartphones moved dramatically when the participants climbed the stairs. The movement of smartphone does affect speech collection. The WER for going up and down is the highest.

6.4.3 Robustness Against Distance Between the Mouth and the Smartphone. To investigate the impact of the mouth-to-smartphone distance on recognition performance, we invited 3 volunteers and deployed AdaStreamLite on their own smartphones. During the experiment process, each volunteer held the smartphone away from the mouth at a required distance and meanwhile read a content-fixed paragraph for a duration of 5 minutes. The distances used for test include 2cm, 4cm, 8cm, and 16cm. For each test distance, the above experiment process was repeated 5 times. We conducted this experiment in a quiet meeting room and a noisy server room, respectively.



(a) The holding gesture for a smartphone. (b) WER (%) in a quiet environment. (c) WER (%) in a noisy environment.

Fig. 17. The impact of distance on the WER metric.



(a) WER (%) in a quiet environment. (b) WER (%) in a noisy environment.

Fig. 18. The impact of angle on the WER metric.

Fig 17 shows the results. In a quiet environment, the mouth-to-smartphone distance has a small impact. However, we observed a significant performance variation w.r.t the distance when conducting this experiment in a noisy environment. This is because a larger distance leads to the decrease of SNR.

6.4.4 Robustness to Angle Between the Horizontal Plane and the Smartphone. We further investigated the impact of the angle between the smartphone and the horizontal plane. The experiment setup is the same as that in Section 6.4.3. Fig 18 shows the results. Different from distance, the impact of the angle is negligible in both quiet and noisy environments. We found that every test smartphone is equipped with a primary microphone at its bottom. Therefore, the mouth of the volunteer is always close to the microphone, no matter how the angle is changed.

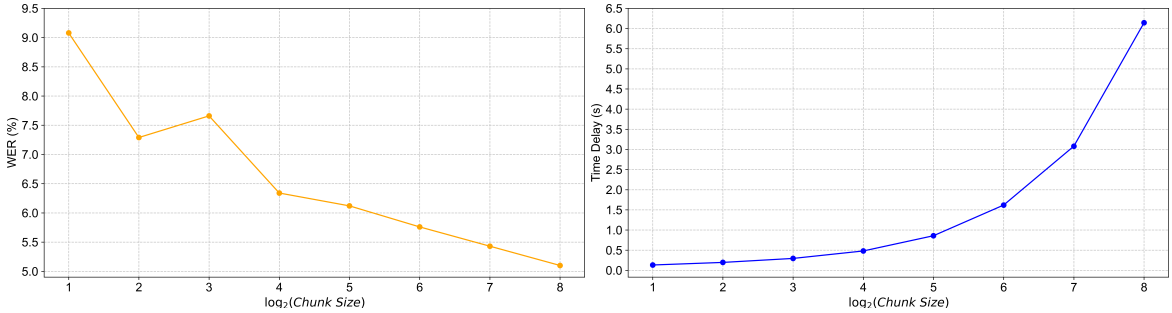
6.4.5 Robustness against Smartphone Diversity. We deployed AdaStreamLite on three different smartphones, including Samsung Galaxy S21, Xiaomi 13, and iQOO Neo6. We asked one participant to use each of them in a server room full of machine noises. The participant read the same textual materials. From Table 5, we can see that the AdaStreamLite system deployed on different smartphones achieves low and similar WERs, which demonstrates the generalibity of AdaStreamLite to different smartphones.

6.5 System Hyper-parameters

6.5.1 Chunk Size. The chunk size refers to the number of frames that are fed into a streaming E2E ASR model in the forward inference process. As introduced in the Subsection 4.2, a frame is an acoustic feature vector extracted from a small sliding window, and we use a 25ms sliding window with a 10ms overlapping for AdaStreamLite. To

Table 5. WER (%) on different smartphone platforms.

Smartphone Platform	Samsung Galaxy S21	Xiaomi 13	iQOO Neo6
WER (%)	7.54	7.62	7.58



(a) WER (%). (b) Time Delay (s).

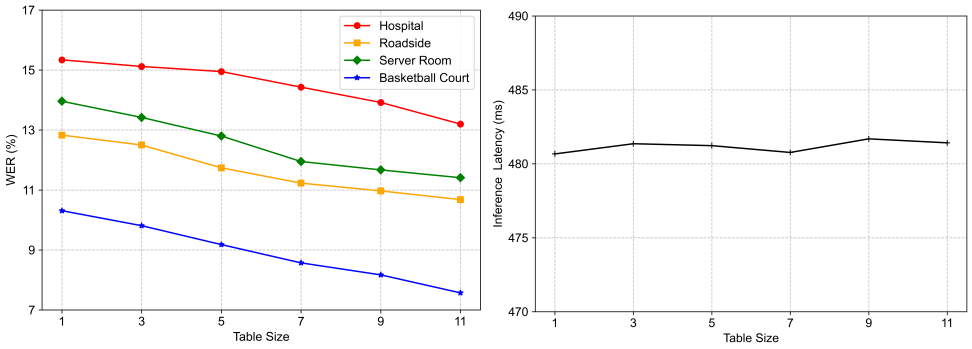
Fig. 19. The impact of chunk size.

investigate the impact of chunk size on recognition performance and inference latency, we asked a volunteer to perform the input of same speech. For each chunk size, the experiment process was repeated five times, and the averaged performance is reported.

Fig 19(a) and Fig 19(b) show the evaluation results of WER and inference latency, respectively. Here, the inference latency refers to the averaged time consumption to collect and process a chunk. The WER value decreases with the increase of the chunk size. This is due to that more speech context information can be used for prediction. However, the inference latency increases with the chunk size. Therefore, there is trade off between recognition performance and inference latency. Based on this observation, we set the chunk size to 16 for AdaStreamLite, which achieves a good balance.

6.5.2 Size of Lookup Table. We use acoustic representations of seen environments to represent unseen environments. We obtain one acoustic representation from each environment. The generalization of AdaStreamLite is related to the number of acoustic environment representations in the lookup table (i.e., the size of the lookup table). To investigate the effect of table size, we evaluated the recognition performance of AdaStreamLite in four unseen environments (i.e., hospital, roadside, server room and basketball court) by varying the size of the lookup table from 1 to 11. Besides, we also investigated the impact of table size on inference latency. Here, the inference latency refers to the time period required to collect and process a chunk of 16 frames. As shown in Fig 20, we can see that the achieved WERs in all four test environments decrease with the increase of table size. When there is only one environment representation in the lookup table, no matter what the new acoustic environment is, the environment embedding used for self-adaptation remains constant, which leads to poor performance. The new acoustic environment can be better represented when there are more environment representations in the lookup table. The computational complexity of the attention-based look-up operation is very low, and the size of the lookup table thus has a small impact on inference latency.

6.5.3 Window Size. When generating a new environment embedding, the query key generator needs to collect an ambient sound clip with a duration of Ts, that is, AdaStreamLite updates the stored environment embedding



(a) WER (%). (b) Inference Latency (ms).

Fig. 20. The impact of the lookup table size.

every Ts. We investigated the impact of the window size on recognition performance. To conduct this experiment, we asked a volunteer to read the same text in four different acoustic environments when setting the window size to different values. For each window size setting, the experiment process was repeated 5 times. We report the average recognition performance. Table 6 lists the WER results. We can see that the WER value decreases with the increase of window size. The reason is that a larger window contains more context information of the ambient noise.

Table 6. WER (%) vs. window size.

Window Size T (s)	hospital	roadside	server room	basketball court
1	17.86	13.27	14.47	12.58
2	16.33	13.01	14.00	10.57
4	15.31	12.50	12.32	9.59
8	14.04	11.27	11.56	8.57
10	13.20	10.68	11.41	7.57

6.6 System Runtime Performance

Besides recognition accuracy measured by WER, users are also concerned with the time delay, memory cost and energy consumption of AdaStreamLite. To investigate the runtime performance of AdaStreamLite, we deployed AdaStreamLite on a Galaxy smartphone (S21 SM–G9910) equipped with a Qualcomm Snapdragon 888 CPU, whose runtime memory and battery capacity are 8GB and 4, 000mAh, respectively. We asked a participant to read textual materials for a duration of 10 minutes after activating the environment-adaptation function of AdaStreamLite. Each experiment process was repeated 5 times and the averaged runtime performance is reported. We measured the time delay as the time interval between the timestamp the word/character shows up in the speech and the timestamp the screen displays the word/character. The time delay covers signal collection, signal pre-processing, speech recognition, and result presentation.

6.6.1 Comparison with Popular Smartphone Applications. To intuitively present the computing resource cost of AdaStreamLite, we selected four popular smartphone applications (TikTok, YouTube, Instagram, and Twitch) for

comparison and asked the participant to use each application 10 minutes on the Galaxy smartphone. Table 7 shows their averaged memory footprint and accumulative energy consumption over 10 minutes. We could see that the computational resource used by AdaStreamLite is quite low.

Table 7. Memory footprint and energy consumption.

	TikTok	YouTube	Instagram	Twitch	AdaStreamLite
Memory Footprint (MB)	635	387	427	566	435
Energy Consumption (mAh)	137	104	134	91	113

6.6.2 Runtime Performance on Different Smartphones. Besides Samsung Galaxy S21, we also evaluated AdaStreamLite on Xiaomi 13 and iQOO Neo6 smartphones. During the experiment process, a participant was asked to read the textual materials for ten minutes. Table 8 lists the results in term of time delay per word, averaged memory consumption, maximum memory consumption, and accumulated energy consumption.

Table 8. Runtime performance on different smartphones.

Smartphone	Galaxy S21	Xiaomi 13	iQOO Neo6
Processor (Qualcomm Snapdragon)	888	8GEN2	870
Runtime Memory (MB)	8192	12288	12288
Battery Capacity (mAh)	4000	4500	4700
Time Delay per Word	0.502	0.498	0.497
Averaged Memory Consumption (MB)	461.94	458.24	463.26
Maximum Memory Consumption (MB)	568.16	567.22	569.57
Energy Consumption (mAh)	90.56	89.32	92.84

7 LIMITATION AND FUTURE WORKS

More Lightweight Streaming E2E ASR Model: In this paper, we directly use the U2++ model as the streaming E2E ASR component of AdaStreamLite. Though the U2++ model can be deployed on a smartphone and its runtime performance is acceptable for smartphone users, the computational resource consumption of AdaStreamLite can be further reduced by using a more lightweight streaming E2E ASR model. In the future, we plan to use the state-of-the-art model compression technologies to optimize AdaStreamLite.

Supporting other Languages: The current version of AdaStreamLite supports speeches in English and Mandarin Chinese. AdaStreamLite can be easily extended to support other languages and we plan to extend AdaStreamLite for other languages.

Environment Switch Detection: AdaStreamLite continuously activates the table querying operation to generate environment embedding for current acoustic environment. Though the query key generator and the self-adaptive regulator are light-weight enough, it is possible to save more computational resource by automatically controlling the switch between two operational modes.

ACKNOWLEDGMENTS

This work is partially supported by a grant from the National Natural Science Foundation of China (No.62032017, No.62272368), the Innovation Capability Support Program of Shaanxi (No.2023-CX-TD-08), Shaanxi Qinchuangyuan

"scientists+engineers" team (No.2023KXJ-040), the Key Research and Development Program of Shaanxi (No. 2021ZDLGY03-09, No.2021ZDLGY07-02), Science and Technology Program of Xi'an, the Youth Innovation Team of Shaanxi Universities, the Engineering Research Center of Blockchain Technology Application and Evaluation of Ministry of Education, The Key Laboratory of Smart Human-Computer Interaction and Wearable Technology of Shaanxi Province.

REFERENCES

[1] Zhongxin Bai and Xiao-Lei Zhang. 2021. Speaker recognition based on deep learning: An overview. *Neural Networks* 140 (2021), 65–99.
[2] Steven Boll. 1979. Suppression of acoustic noise in speech using spectral subtraction. *IEEE Transactions on acoustics, speech, and signal processing* 27, 2 (1979), 113–120.
[3] Hui Bu, Jiayu Du, Xingyu Na, Bengu Wu, and Hao Zheng. 2017. AIShell-1: An Open-Source Mandarin Speech Corpus and A Speech Recognition Baseline. In *Oriental COCOSDA 2017*. Submitted.
[4] Suliang Bu, Yunxin Zhao, Tuo Zhao, Shaojun Wang, and Mei Han. 2022. Modeling Speech Structure to Improve T-F Masks for Speech Enhancement and Recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 30 (2022), 2705–2715. https://doi.org/10.1109/TASLP.2022.3196168
[5] Joseph P Campbell. 1997. Speaker recognition: A tutorial. *Proc. IEEE* 85, 9 (1997), 1437–1462.
[6] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. 2016. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 4960–4964. https://doi.org/10.1109/ICASSP.2016.7472621
[7] Jun Chen, Zilin Wang, Deyi Tuo, Zhiyong Wu, Shiyin Kang, and Helen Meng. 2022. FullSubNet+: Channel Attention Fullsubnet with Complex Spectrograms for Speech Enhancement. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 7857–7861. https://doi.org/10.1109/ICASSP43922.2022.9747888
[8] Xie Chen, Yu Wu, Zhenghao Wang, Shujie Liu, and Jinyu Li. 2021. Developing Real-Time Streaming Transformer Transducer for Speech Recognition on Large-Scale Dataset. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 5904–5908. https://doi.org/10.1109/ICASSP39728.2021.9413535
[9] Chung-Cheng Chiu, Tara N. Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J. Weiss, Kanishka Rao, Ekaterina Gonina, Navdeep Jaitly, Bo Li, Jan Chorowski, and Michiel Bacchiani. 2018. State-of-the-Art Speech Recognition with Sequence-to-Sequence Models. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 4774–4778. https://doi.org/10.1109/ICASSP.2018.8462105
[10] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-Based Models for Speech Recognition. In *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (Eds.), Vol. 28. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2015/file/1068c6e4c8051cfd4e9ea8072e3189e2-Paper.pdf
[11] George E. Dahl, Dong Yu, Li Deng, and Alex Acero. 2012. Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition. *IEEE Transactions on Audio, Speech, and Language Processing* 20, 1 (2012), 30–42. https://doi.org/10.1109/TASL.2011.2134090
[12] Jiankang Deng, Jia Guo, Jing Yang, Niannan Xue, Irene Kotsia, and Stefanos Zafeiriou. 2022. ArcFace: Additive Angular Margin Loss for Deep Face Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 10 (2022), 5962–5979. https://doi.org/10.1109/TPAMI.2021.3087709
[13] Lei Deng, Guoqi Li, Song Han, Luping Shi, and Yuan Xie. 2020. Model compression and hardware acceleration for neural networks: A comprehensive survey. *Proc. IEEE* 108, 4 (2020), 485–532.
[14] Brecht Desplanques, Jenthe Thienpondt, and Kris Demuynck. 2020. ECAPA-TDNN: Emphasized Channel Attention, Propagation and Aggregation in TDNN Based Speaker Verification. In *Proc. Interspeech 2020*. 3830–3834. https://doi.org/10.21437/Interspeech.2020-2650
[15] Han Ding, Yizhan Wang, Hao Li, Cui Zhao, Ge Wang, Wei Xi, and Jizhong Zhao. 2022. UltraSpeech: Speech Enhancement by Interaction between Ultrasound and Speech. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 6, 3, Article 111 (sep 2022), 25 pages. https://doi.org/10.1145/3550303
[16] Y. Ephraim and D. Malah. 1985. Speech enhancement using a minimum mean-square error log-spectral amplitude estimator. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 33, 2 (1985), 443–445. https://doi.org/10.1109/TASSP.1985.1164550
[17] Ariel Ephrat, Inbar Mosseri, Oran Lang, Tali Dekel, Kevin Wilson, Avinatan Hassidim, William T. Freeman, and Michael Rubinstein. 2018. Looking to Listen at the Cocktail Party: A Speaker-Independent Audio-Visual Model for Speech Separation. *ACM Trans. Graph.* 37, 4, Article 112 (jul 2018), 11 pages. https://doi.org/10.1145/3197517.3201357
[18] Petko Georgiev, Sourav Bhattacharya, Nicholas D. Lane, and Cecilia Mascolo. 2017. Low-Resource Multi-Task Audio Sensing for Mobile and Embedded Devices via Shared Deep Neural Network Representations. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 3, Article 50 (sep 2017), 19 pages. https://doi.org/10.1145/3131895

[19] Alex Graves. 2012. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711* (2012).
[20] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*. 369–376.
[21] Alex Graves and Navdeep Jaitly. 2014. Towards End-To-End Speech Recognition with Recurrent Neural Networks. In *Proceedings of the 31st International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 32)*, Eric P. Xing and Tony Jebara (Eds.). PMLR, Bejing, China, 1764–1772. https://proceedings.mlr.press/v32/graves14.html
[22] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. 6645–6649. https://doi.org/10.1109/ICASSP.2013.6638947
[23] Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401* (2014).
[24] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, et al. 2020. Conformer: Convolution-augmented transformer for speech recognition. *arXiv preprint arXiv:2005.08100* (2020).
[25] Mahdi Hajibabaei and Dengxin Dai. 2018. Unified hypersphere embedding for speaker recognition. *arXiv preprint arXiv:1807.08312* (2018).
[26] Yanzhang He, Tara N. Sainath, Rohit Prabhavalkar, Ian McGraw, Raziel Alvarez, Ding Zhao, David Rybach, Anjuli Kannan, Yonghui Wu, Ruoming Pang, Qiao Liang, Deepti Bhatia, Yuan Shangguan, Bo Li, Golan Pundak, Khe Chai Sim, Tom Bagby, Shuo-yiin Chang, Kanishka Rao, and Alexander Gruenstein. 2019. Streaming End-to-end Speech Recognition for Mobile Devices. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 6381–6385. https://doi.org/10.1109/ICASSP.2019.8682336
[27] Hu Hu, Tian Tan, and Yanmin Qian. 2018. Generative Adversarial Networks Based Data Augmentation for Noise Robust Speech Recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 5044–5048. https://doi.org/10.1109/ICASSP.2018.8462624
[28] Insider Intelligence. 2023. *Voice Assistants in 2023: Usage, growth, and future of the AI voice assistant market.* Retrieved July 08, 2023 from https://www.insiderintelligence.com/insights/voice-assistants/
[29] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*. pmlr, 448–456.
[30] F. Jelinek. 1976. Continuous speech recognition by statistical methods. *Proc. IEEE* 64, 4 (1976), 532–556. https://doi.org/10.1109/PROC.1976.10159
[31] Kwangyoun Kim, Kyungmin Lee, Dhananjaya Gowda, Junmo Park, Sungsoo Kim, Sichen Jin, Young-Yoon Lee, Jinsu Yeo, Daehyun Kim, Seokyeong Jung, et al. 2019. Attention based on-device streaming speech recognition with large speech corpus. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 956–963.
[32] Lantian Li, Ruiqian Nai, and Dong Wang. 2022. Real Additive Margin Softmax for Speaker Verification. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 7527–7531. https://doi.org/10.1109/ICASSP43922.2022.9747166
[33] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. 2020. Federated Learning: Challenges, Methods, and Future Directions. *IEEE Signal Processing Magazine* 37, 3 (2020), 50–60. https://doi.org/10.1109/MSP.2020.2975749
[34] J.S. Lim and A.V. Oppenheim. 1979. Enhancement and bandwidth compression of noisy speech. *Proc. IEEE* 67, 12 (1979), 1586–1604. https://doi.org/10.1109/PROC.1979.11540
[35] Sicong Liu, Bin Guo, Ke Ma, Zhiwen Yu, and Junzhao Du. 2021. AdaSpring: Context-Adaptive and Runtime-Evolutionary Deep Model Compression for Mobile Applications. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 5, 1, Article 24 (mar 2021), 22 pages. https://doi.org/10.1145/3448125
[36] Philipos C Loizou. 2013. *Speech enhancement: theory and practice.* CRC press.
[37] Xugang Lu, Yu Tsao, Shigeki Matsuda, and Chiori Hori. 2013. Speech enhancement based on deep denoising autoencoder. In *Proc. Interspeech 2013*. 436–440. https://doi.org/10.21437/Interspeech.2013-130
[38] Douglas A Lyon. 2009. The discrete fourier transform, part 4: spectral leakage. *Journal of object technology* 8, 7 (2009).
[39] Haoran Miao, Gaofeng Cheng, Pengyuan Zhang, and Yonghong Yan. 2020. Online Hybrid CTC/Attention End-to-End Automatic Speech Recognition Architecture. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 28 (2020), 1452–1465. https://doi.org/10.1109/TASLP.2020.2987752
[40] Zhaoxu Nian, Yan-Hui Tu, Jun Du, and Chin-Hui Lee. 2021. A Progressive Learning Approach to Adaptive Noise and Speech Estimation for Speech Enhancement and Noisy Speech Recognition. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 6913–6917. https://doi.org/10.1109/ICASSP39728.2021.9413395
[41] Sergey Novoselov, Vadim Shchemelinin, Andrey Shulipa, Alexandr Kozlov, and Ivan Kremnev. 2018. Triplet Loss Based Cosine Similarity Metric Learning for Text-independent Speaker Recognition. In *Proc. Interspeech 2018*. 2242–2246. https://doi.org/10.21437/Interspeech.2018-1209
[42] Koji Okabe, Takafumi Koshinaka, and Koichi Shinoda. 2018. Attentive Statistics Pooling for Deep Speaker Embedding. In *Proc. Interspeech 2018*. 2252–2256. https://doi.org/10.21437/Interspeech.2018-993

[43] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: An ASR corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 5206–5210. https://doi.org/10.1109/ICASSP.2015.7178964

[44] Santiago Pascual, Antonio Bonafonte, and Joan Serra. 2017. SEGAN: Speech enhancement generative adversarial network. *arXiv preprint arXiv:1703.09452* (2017).

[45] L.R. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE* 77, 2 (1989), 257–286. https://doi.org/10.1109/5.18626

[46] Prajit Ramachandran, Barret Zoph, and Quoc V Le. 2017. Searching for activation functions. *arXiv preprint arXiv:1710.05941* (2017).

[47] D Raj Reddy. 1976. Speech recognition by machine: A review. *Proc. IEEE* 64, 4 (1976), 501–531.

[48] Leda Sarı, Niko Moritz, Takaaki Hori, and Jonathan Le Roux. 2020. Unsupervised Speaker Adaptation Using Attention-Based Speaker Memory for End-to-End ASR. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 7384–7388. https://doi.org/10.1109/ICASSP40776.2020.9054249

[49] Hendrik Schroter, Alberto N. Escalante-B, Tobias Rosenkranz, and Andreas Maier. 2022. Deepfilternet: A Low Complexity Speech Enhancement Framework for Full-Band Audio Based On Deep Filtering. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 7407–7411. https://doi.org/10.1109/ICASSP43922.2022.9747055

[50] Eric Hal Schwartz. 2021. *EU Publishes Privacy Guidelines for Voice Assistants for Comment.* Retrieved April 27, 2023 from https://voicebot.ai/2021/03/23/eu-publishes-privacy-guidelines-for-voice-assistants-for-comment/

[51] Michael L. Seltzer, Dong Yu, and Yongqiang Wang. 2013. An investigation of deep neural networks for noise robust speech recognition. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. 7398–7402. https://doi.org/10.1109/ICASSP.2013.6639100

[52] David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur. 2018. X-Vectors: Robust DNN Embeddings for Speaker Recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 5329–5333. https://doi.org/10.1109/ICASSP.2018.8461375

[53] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.

[54] Joachim Thiemann, Nobutaka Ito, and Emmanuel Vincent. 2013. DEMAND: a collection of multi-channel recordings of acoustic noise in diverse environments. In *Proc. Meetings Acoust.* 1–6.

[55] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).

[56] Ehsan Variani, Ke Wu, Michael D Riley, David Rybach, Matt Shannon, and Cyril Allauzen. 2022. Global Normalization for Streaming Speech Recognition in a Modular Framework. In *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.), Vol. 35. Curran Associates, Inc., 4257–4269. https://proceedings.neurips.cc/paper_files/paper/2022/file/1b4839ff1f843b6be059bd0e8437e975-Paper-Conference.pdf

[57] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[58] Li Wan, Quan Wang, Alan Papir, and Ignacio Lopez Moreno. 2018. Generalized End-to-End Loss for Speaker Verification. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 4879–4883. https://doi.org/10.1109/ICASSP.2018.8462665

[59] Yuxuan Wang, Arun Narayanan, and DeLiang Wang. 2014. On Training Targets for Supervised Speech Separation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 22, 12 (2014), 1849–1858. https://doi.org/10.1109/TASLP.2014.2352935

[60] Yuxuan Wang, Daisy Stanton, Yu Zhang, RJ-Skerry Ryan, Eric Battenberg, Joel Shor, Ying Xiao, Ye Jia, Fei Ren, and Rif A Saurous. 2018. Style tokens: Unsupervised style modeling, control and transfer in end-to-end speech synthesis. In *International Conference on Machine Learning*. PMLR, 5180–5189.

[61] Shinji Watanabe, Takaaki Hori, Suyoun Kim, John R. Hershey, and Tomoki Hayashi. 2017. Hybrid CTC/Attention Architecture for End-to-End Speech Recognition. *IEEE Journal of Selected Topics in Signal Processing* 11, 8 (2017), 1240–1253. https://doi.org/10.1109/JSTSP.2017.2763455

[62] Yuheng Wei, Junzhao Du, and Hui Liu. 2020. Angular Margin Centroid Loss for Text-Independent Speaker Recognition. In *Proc. Interspeech 2020*. 3820–3824. https://doi.org/10.21437/Interspeech.2020-2538

[63] Yong Xu, Jun Du, Li-Rong Dai, and Chin-Hui Lee. 2015. A Regression Approach to Speech Enhancement Based on Deep Neural Networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 23, 1 (2015), 7–19. https://doi.org/10.1109/TASLP.2014.2364452

[64] Binbin Zhang, Hang Lv, Pengcheng Guo, Qijie Shao, Chao Yang, Lei Xie, Xin Xu, Hui Bu, Xiaoyu Chen, Chenchen Zeng, Di Wu, and Zhendong Peng. 2022. WENETSPEECH: A 10000+ Hours Multi-Domain Mandarin Corpus for Speech Recognition. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 6182–6186. https://doi.org/10.1109/ICASSP43922.2022.9746682

[65] Binbin Zhang, Di Wu, Zhendong Peng, Xingchen Song, Zhuoyuan Yao, Hang Lv, Lei Xie, Chao Yang, Fuping Pan, and Jianwei Niu. 2022. Wenet 2.0: More productive end-to-end speech recognition toolkit. *arXiv preprint arXiv:2203.15455* (2022).

[66] Binbin Zhang, Di Wu, Zhuoyuan Yao, Xiong Wang, Fan Yu, Chao Yang, Liyong Guo, Yaguang Hu, Lei Xie, and Xin Lei. 2020. Unified streaming and non-streaming two-pass end-to-end model for speech recognition. *arXiv preprint arXiv:2012.05481* (2020).

[67] Qian Zhang, Han Lu, Hasim Sak, Anshuman Tripathi, Erik McDermott, Stephen Koo, and Shankar Kumar. 2020. Transformer Transducer: A Streamable Speech Recognition Model with Transformer Encoders and RNN-T Loss. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 7829–7833. https://doi.org/10.1109/ICASSP40776.2020.9053896

[68] Qian Zhang, Dong Wang, Run Zhao, Yinggang Yu, and Junjie Shen. 2021. Sensing to Hear: Speech Enhancement for Mobile Devices Using Acoustic Signals. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 5, 3, Article 137 (sep 2021), 30 pages. https://doi.org/10.1145/3478093

[69] Yang Zhang, Zhiqiang Lv, Haibin Wu, Shanshan Zhang, Pengfei Hu, Zhiyong Wu, Hung yi Lee, and Helen Meng. 2022. MFA-Conformer: Multi-scale Feature Aggregation Conformer for Automatic Speaker Verification. In *Proc. Interspeech 2022*. 306–310. https://doi.org/10.21437/Interspeech.2022-563

[70] Zixing Zhang, Jürgen Geiger, Jouni Pohjalainen, Amr El-Desoky Mousa, Wenyu Jin, and Björn Schuller. 2018. Deep Learning for Environmentally Robust Speech Recognition: An Overview of Recent Developments. *ACM Trans. Intell. Syst. Technol.* 9, 5, Article 49 (apr 2018), 28 pages. https://doi.org/10.1145/3178115

[71] Zhi Zhou, Xu Chen, En Li, Liekang Zeng, Ke Luo, and Junshan Zhang. 2019. Edge Intelligence: Paving the Last Mile of Artificial Intelligence With Edge Computing. *Proc. IEEE* 107, 8 (2019), 1738–1762. https://doi.org/10.1109/JPROC.2019.2918951

A USER EXPERIENCE STUDY

To study the user experience of AdaStreamLite, we asked six volunteers to score the AdaStreamLite and MultiCondition systems on a scale of 1 to 5 (poor: 1, fair: 2, average: 3, good: 4, excellent: 5). More specially, we prepared the audio files of one hundred sentences and used both AdaStreamLite and MultiCondition to recognize them. These speech data are collected from 12 acoustic environments with different SNR levels (from −5dB to 30dB). Each audio file was broadcast to the volunteers, and the recognition results of the two systems were shown to the volunteers. We asked the volunteers to score the speech-text translation service provided by the two systems on a per-sentence basis and calculated the averaged score over all sentences for each system. Table 9 lists the evaluation results and we can see that AdaStreamLite achieves consistently higher scores than MultiCondition.

Table 9. User satisfaction scores.

	user 1	user 2	user 3	user 4	user 5	user 6
AdaStreamLite	4.70	4.56	4.72	4.68	4.36	4.52
MultiCondition	3.92	3.86	3.90	3.84	3.89	3.88