AI Note: A Privacy-Preserving Intelligent Note-Taking System with On-Device Speech Recognition, Online Speaker Clustering, and Cloud-Relayed Large Language Models

Zhiguo Wei — Independent Developer

Abstract

Intelligent note-taking applications increasingly rely on cloud-based automatic speech recognition (ASR), requiring users to upload continuous audio streams to third-party servers. Since speech is a biometric identifier, this architecture exposes users to substantial privacy risks. This paper presents AI Note, a privacy-preserving note-taking system in which all audio processing (voice activity detection, speech recognition, speaker embedding extraction, and noise suppression) is performed entirely on the mobile device, and audio never leaves the user's phone. Text-level intelligence (summarization, question answering, mind-map generation) is provided on demand through a self-hosted relay service that enforces authentication, quota accounting, and rate limiting. To deliver a real-time transcription experience with an offline, non-streaming ASR model, we propose a streaming-simulation segmentation strategy that couples a VAD state machine with a pre-roll ring buffer and dual-threshold window cutting, eliminating sentence-onset truncation without degrading accuracy. To annotate speakers without waiting for the recording to finish, we design an online speaker clustering method that computes segment-level embeddings and maintains speaker centroids incrementally, reducing speaker-attribution latency from tens of seconds to near zero. Experiments on the full evaluation sets show three results. The segmentation strategy preserves the accuracy of whole-utterance decoding across the entire parameter grid (2.98%–4.17% CER on clean speech), and a 0.6 s pre-roll combined with a 1.0 s pause threshold achieves the lowest CER (2.98%) with zero onset truncation. Online speaker clustering attains 76.6% attribution accuracy on 31 multi-speaker conversations, including 20 real three-to-four-speaker meetings from the public AliMeeting corpus; this is 7.8 percentage points above a fixed-threshold baseline and far above an offline agglomerative-clustering baseline (46.3% on the two-speaker subset), comes with no added latency when recording stops, and rises to 79.7% after a millisecond-scale post-meeting refinement. Finally, the int8-quantized ASR model matches the fp32 model in accuracy (3.23% vs. 3.28% mean CER) at a lower real-time factor (0.34 vs. 0.38) and one quarter of the model size. A sixth experiment adapts a 1.7B LLM into an on-device meeting-minutes model: trained on a three-layer recipe of real-meeting distillation, synthetic expansion, and boundary negatives, one defect-driven data iteration raises the clean rate on fabricated to-dos and fabricated owners from 20% to 80% at 98% token-level digit fidelity, and the model quantizes to 1.03 GiB, closing the remaining text-level privacy gap at draft quality. The complete system has been deployed and validated in real-world usage, including hour-long full-load recordings on a commodity phone.

Index Terms—On-device speech recognition, privacy preservation, speaker diarization, voice activity detection, mobile computing, large language models, parameter-efficient fine-tuning.

I. Introduction

A. Background and Motivation

Note-taking by voice has become a daily need for students, journalists, lawyers, and business professionals. Mainstream solutions, including iFlytek, Tencent Cloud ASR, and various transcription apps, perform recognition in the cloud: the user's microphone audio is streamed to a remote server, transcribed, and returned. This architecture has three drawbacks. First, *privacy*: speech carries biometric identity and sensitive content; once uploaded, control over the data is lost, and breaches or misuse are beyond the user's reach. This concern is also regulatory: China's Personal Information Protection Law (PIPL) classifies biometric information, including voiceprints, as sensitive personal information, whose processing demands a specific purpose, demonstrated necessity, and strict protective measures; uploading continuous audio merely to obtain text stands in tension with this data-minimization direction. Second, *availability*: cloud recognition fails or degrades without reliable connectivity. Third, *cost structure*: per-minute cloud ASR pricing scales linearly with usage, discouraging long-form recording.

Meanwhile, on-device inference has matured: quantized ASR models of a few hundred megabytes now run in real time on commodity smartphone SoCs, a path validated both by Google's reference on-device GenAI application [14] and by commercial on-device dictation products [15], and embedded inference runtimes such as sherpa-onnx package them for mobile deployment; at the small end, tool-calling models of only tens of millions of parameters have been quantized into a single file of tens of megabytes for on-device use [16]. We therefore ask whether an intelligent note-taking application can perform all audio processing locally, contacting the network only for text-level intelligence the user explicitly requests. AI Note is built to do exactly this.

B. Contributions

The contributions of this paper are fourfold:

1. **A privacy-preserving edge-cloud boundary architecture.** We design and implement a note-taking system in which VAD, ASR, speaker embedding, and denoising run fully on-device; only user-initiated text requests traverse the network, relayed by a self-hosted service that provides email-code authentication, monthly quotas, sliding-window rate limiting, and model whitelisting. The architecture gives end users AI capabilities without requiring them to own or configure any third-party API credentials.

2. **A streaming-simulation segmentation strategy for non-streaming ASR models.** Offline encoder models such as SenseVoice transcribe whole segments and provide no partial hypotheses; naive fixed-window cutting truncates phonemes at boundaries and feeding silence produces hallucinated text. We combine a Silero-VAD state machine (with a 0.1 s speech-onset confirmation), a 0.6 s pre-roll ring buffer that compensates for VAD onset lag, and dual-threshold cutting (0.6 s pause with ≥1.5 s minimum length, or a configurable 10–15 s maximum length), with periodic snapshot decoding for in-progress preview. This strategy yields near-real-time display while preserving the accuracy of whole-utterance decoding.

3. **Online speaker clustering via segment-level embeddings.** Instead of post-hoc offline diarization over the entire recording (the standard pyannote-style pipeline, which imposes tens of seconds of waiting), we extract a CAM++ embedding for each decoded segment and perform incremental cosine-similarity clustering against running speaker centroids (threshold 0.6, at most 8 speakers, centroid updated by moving average). Duration-aware assignment rules suppress spurious clusters from short segments, and a bounded-delay re-decision mechanism (a three-segment lookahead smoothed by dynamic programming with a switch penalty) corrects boundary misassignments at a fixed attribution delay of three segments. An optional post-meeting refinement re-clusters the cached embeddings in milliseconds. Speaker labels thus appear during recording, and stopping the recording incurs no additional diarization delay.
4. **Domain adaptation and iteration of an on-device meeting-minutes model.** A three-layer data recipe (real-meeting distillation, synthetic expansion, boundary negatives) adapts a 1.7B LLM to speaker-labeled meeting transcripts, and a defect-driven loop (audit failures, generate targeted data, retrain, re-evaluate) raises the clean rate on fabricated to-dos and fabricated owners from 20% to 80%. The adapted model quantizes to 1.03 GiB and runs on a plain CPU, closing the remaining text-level privacy gap at draft quality.

C. Organization

Section II reviews related work. Section III presents the system architecture. Section IV details the key methods. Section V describes implementation. Section VI reports experiments. Section VII concludes.

II. Related Work

A. End-to-End Speech Recognition and On-Device Deployment

Modern ASR and LLM components alike build on the Transformer architecture [11]. End-to-end ASR has replaced hybrid pipelines across benchmarks. Whisper [1] demonstrated robust multilingual recognition via large-scale weak supervision; Paraformer [2] proposed a non-autoregressive design widely adopted in Chinese industry deployments. SenseVoice-Small [3] is a multilingual (zh/en/ja/ko/yue) encoder model with integrated punctuation and inverse text normalization, released together with int8-quantized exports suitable for mobile deployment. sherpa-onnx [4] packages such models with an embedded ONNX Runtime for Android/iOS, exposing VAD, ASR, speaker-embedding, and speech-enhancement APIs. Our work builds on these components and focuses on the system-level problem of turning them into a privacy-preserving, real-time-feeling product. Two recent systems mark the industry poles around this design space. Google AI Edge Gallery [14] demonstrates the fully local route, with Gemma int4 models running on CPU/GPU/NPU through LiteRT, including a 270M-parameter function-calling model, while FluidVoice [15], a macOS dictation product, pairs an on-device 120M-parameter ASR model with a small on-device enhancement LLM, and is commercially operated with a free-ASR/paid-enhancement split. Our hybrid architecture sits between these poles: audio processing is fully local as in [14], [15], but text-level intelligence is served by a relayed cloud LLM, trading a bounded, user-initiated text disclosure for stronger text intelligence and zero on-device LLM footprint.

B. Voice Activity Detection

Silero VAD [5] is a compact LSTM-based detector operating on 512-sample windows at 16 kHz, exposing onset/offset hysteresis parameters. It is commonly used both as a pre-processor for ASR and as a segmenter. Our segmentation strategy differs from plain VAD segmentation: we keep VAD only as a gate and apply our own buffering/cutting logic to serve a non-streaming recognizer.

C. Speaker Diarization and Embeddings

The dominant diarization paradigm runs segmentation (e.g., pyannote.segmentation [6]) followed by embedding extraction and offline clustering over the whole recording. Speaker embeddings such as CAM++ [7] provide compact utterance-level vectors with strong cosine-similarity separability. Streaming diarization research exists (e.g., UIS-RNN [8], online clustering variants [9]) but is rarely deployed on mobile devices. Our online segment-level clustering is a pragmatic adaptation of online clustering to the constraints of a single-threaded mobile executor; because its VAD-driven segments carry more speech than the short uniform windows used by offline clustering baselines, Section VI-C shows it in fact attains higher attribution accuracy while adding zero latency at recording stop.

D. Speech Enhancement for ASR Front-Ends

Lightweight enhancement models such as GTCRN [10] achieve competitive denoising at a fraction of the computational cost of larger CRN variants, making per-segment, on-device denoising before ASR feasible on phones.

E. LLM-Based Text Intelligence and Privacy

LLMs are increasingly used for summarization and question answering over transcripts. Mainstream products send the entire transcript to the provider. Our relay architecture inserts an authenticated, quota-limited intermediary so that (i) users need no third-party credentials, (ii) the operator can enforce abuse limits, and (iii) audio-derived text leaves the device only upon explicit user action. Recent tiny-model work shows how far this direction can be pushed on-device: needle [16], a 45M-parameter tool-calling model quantized into a 14 MB single file, demonstrates confidence-gated edge-cloud routing in which only low-confidence requests are escalated to the cloud. This is direct evidence that the graded privacy routing our architecture assumes is practical on current phones. Parameter-efficient fine-tuning pushes the same direction for text intelligence: LoRA [17] adapts open models in the 1–2B range such as Qwen3 [18] on a single GPU in minutes, and GGUF quantization compresses the result to the one-gigabyte scale that runs on plain CPUs. Section VI-G validates this route end to end for meeting minutes.

III. System Architecture

A. Design Principles

1. *Audio never leaves the device.* All microphones frames are processed locally; the recorder, VAD, ASR, embedding extractor, and denoiser are on-device.
2. *Text crosses the boundary only on demand.* Summarization, Q&A, mind-mapping, and weekly reports are user-initiated; each request is an explicit, auditable action.
3. *Graceful degradation.* If models or network are unavailable, features degrade (RMS energy gating instead of VAD, no speaker labels, no AI) but core recording and transcription continue.

B. Overall Structure

The system consists of three tiers:

- **Mobile application (Android, Kotlin/Jetpack Compose).** Audio capture (AudioRecord, 16 kHz mono PCM), the segmentation engine (Section IV-A), on-device model management (download, integrity and IR-version validation), local persistence (Room), and the UI.
- **Relay service (FastAPI + SQLite).** Email-code login issuing 30-day JWTs; per-user monthly quota with automatic reset; sliding-window rate limiting (20 requests/min); an OpenAI-compatible `/v1/chat/completions` endpoint supporting server-sent events (SSE); upstream failure refunds quota. Model whitelisting prevents abuse of expensive upstream models.
- **Upstream LLM provider.** Any OpenAI-compatible API (DeepSeek by default). The upstream key exists only on the server.

C. Data Classification

Audio and raw transcripts are classified as *device-only*. User-submitted prompt text and retrieved note excerpts are classified as *relay-transient*: they transit the relay without persistent storage (only per-user counters are stored). This classification maps directly onto regulatory data-minimization requirements and is documented in the application's privacy policy.

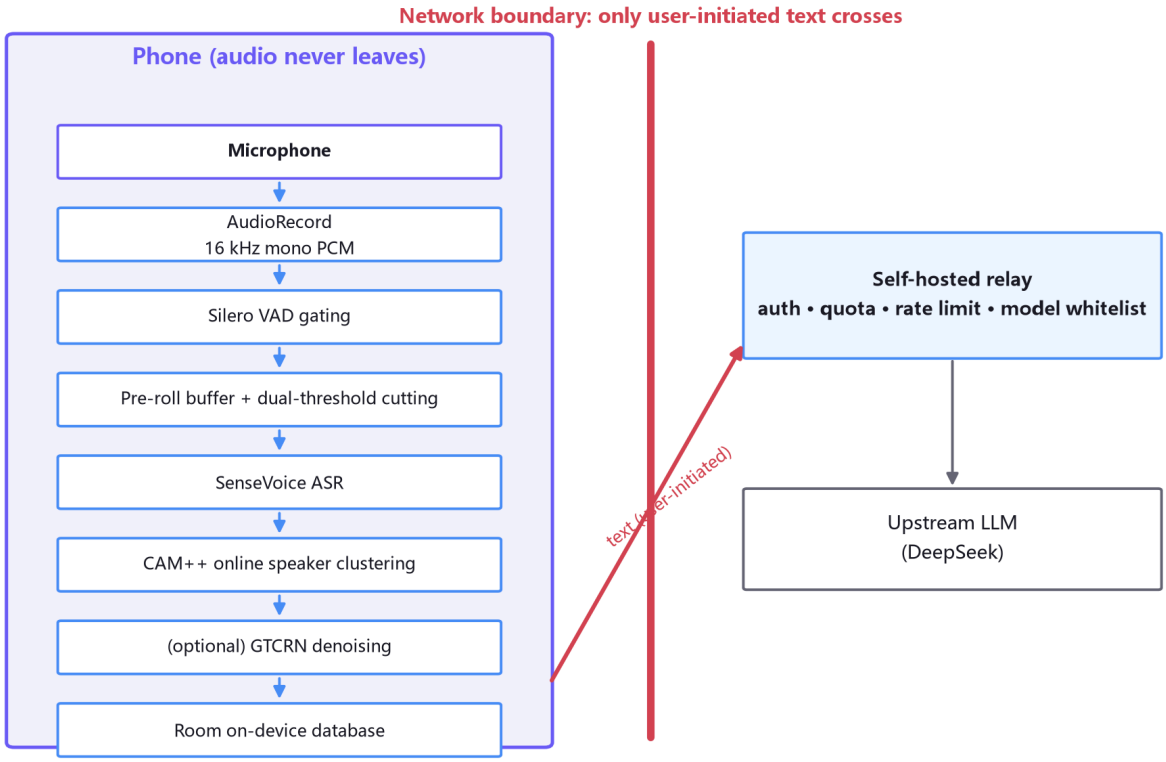

Figure 1: Overall system architecture. Left: the on-device pipeline (microphone → AudioRecord 16 kHz mono PCM → Silero VAD gating → pre-roll buffer with dual-threshold cutting → SenseVoice ASR → CAM++ online speaker clustering → optional GTCRN denoising → Room on-device database); audio never crosses the bold red network boundary, and only user-initiated text does. Right: the self-hosted relay service (authentication, quota, rate limiting, model whitelist) and the upstream LLM (DeepSeek [12]).

IV. Key Methods

A. Streaming-Simulation Segmentation

Problem. SenseVoice decodes complete segments; feeding raw microphone frames continuously is impossible, and naive fixed-length cutting truncates phonemes, harming accuracy. Feeding silence yields hallucinated tokens.

Definitions. Frames arrive every 100 ms (1600 samples). The Silero VAD consumes 512-sample windows and maintains an internal state; `isSpeechDetected()` flips true only after `minSpeechDuration` of continuous speech.

Method. The acquisition thread maintains (i) a pre-roll ring buffer of the last 10 frames (0.6 s), (ii) an active segment buffer, and (iii) counters for consecutive non-speech frames. For each frame:

1. Feed the frame (buffered into 512-sample windows) to the VAD.
2. If the frame is non-speech and no segment is active, retain it only in the ring buffer.

3. When speech first activates a segment, prepend the ring buffer, so onset lag (≤0.25 s in practice) never truncates the sentence start.
4. Cut the segment when silence persists for 6 frames (0.6 s) with segment length ≥1.5 s, or when the segment reaches the configured maximum (10 s default; 15 s in long-context mode).
5. Discard segments containing fewer than 0.3 s of speech as noise.
6. Every 1.5 s during an active segment, submit a snapshot decode of the current buffer as a partial preview (skipped if the decoder is busy).

Decoding runs on a single-threaded executor, serializing segment decode, snapshot decode, and embedding extraction to avoid resource contention on mobile CPUs.

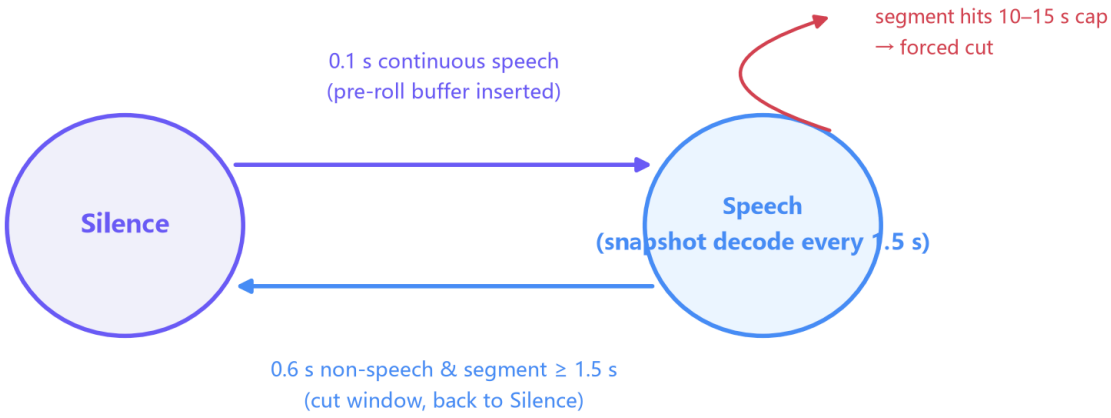

Figure 2: The segmentation state machine. 0.1 s of continuous speech enters the speech state (with the pre-roll buffer prepended); 0.6 s of non-speech with a segment of at least 1.5 s cuts the window and returns to silence; a segment hitting the 10–15 s cap is cut forcibly; during the speech state a snapshot decode is submitted every 1.5 s.

B. Online Speaker Clustering

For each finalized segment we compute a CAM++ embedding `e` (192-d) with the on-device extractor. We maintain a list of speaker centroids `c_1..c_k` with hit counts `n_1..n_k`. Assignment:

```
j* = argmax_j cosine(e, c_j)
if cosine(e, c_{j*}) >= τ (0.6)   → assign segment to speaker j*
                                    c_{j*} ← (n_{j*} • c_{j*} + e)/(n_{j*}+1); n_{j*} += 1
elif k < K_max (8)                → create new centroid c_{k+1} = e
else                              → assign to nearest centroid (no update)
```

Embedding extraction adds only the cost of one forward pass per segment (sub-second on a mid-range SoC for typical 2–10 s segments), and requires no segmentation model: the VAD-driven segment

boundaries serve as turn hypotheses. When the embedding model is absent, the system falls back to unlabeled transcription.

Four refinements, each ablated in Section VI-C, complete the method.

1. *Duration-aware assignment.* Segments shorter than 1 s inherit the previous speaker without opening a cluster or updating any centroid; segments of 1–2 s may join an existing cluster but never open a new one; only segments of at least 2 s participate in cluster creation and centroid updates. This targets the dominant failure mode of the naive rule: spurious clusters opened by short or boundary-straddling segments.
2. *Bounded-delay re-decision.* The most recent N = 3 segments are buffered and their joint labeling is smoothed by dynamic programming, with cost = negative similarity sum + δ (0.05) per speaker switch; only the oldest segment in the window is committed, so the attribution delay is fixed at three segments and never grows with recording length.
3. *Adaptive threshold.* The acceptance threshold tracks the running distribution of accepted similarities as clamp(μ − 1.5σ, 0.45, 0.75), with a fixed 0.6 cold start for the first six samples. Section VI-C shows this component brings no gain in isolation and analyses the survivor bias behind this negative result.
4. *Post-meeting refinement.* When the recording stops, the embeddings cached during the session are re-clustered with agglomerative clustering (average linkage, cosine distance) initialized from the online partition; the cluster count k is chosen by silhouette score within [online k − 2, online k + 1], followed by two rounds of centroid refinement and a merge of clusters whose centroids exceed cosine similarity 0.6 (the same threshold used online). The whole step costs milliseconds because no embedding is recomputed.

C. Model Management and Compatibility Guarding

On-device models are downloaded at first use from mainland-accessible mirrors (ModelScope, hf-mirror) with multi-source fallback and size validation. Because embedded ONNX Runtime versions support only bounded ONNX IR versions, model files are validated at load time by parsing the protobuf header's `ir_version` field; incompatible files are rejected and the feature degrades rather than aborting. The VAD model (v5, IR downgraded to 9 without altering weights) is bundled in application assets to remove a network dependency and a previously observed native crash vector. Users may optionally download the fp32 ASR model (~936 MB) for higher accuracy, or a GTCRN denoiser (~0.5 MB) applied to finalized segments before decoding in noisy environments.

D. Relay Security and Accounting

Login is passwordless: six-digit e-mail codes (10-minute expiry, 5 attempts, 10 codes/day/address) are exchanged for 30-day JWTs. Each chat request decrements a monthly quota (default 100) atomically with the request; upstream errors refund it. A per-user sliding-window limiter (20/min) and a request-size cap (60 k chars) bound abuse. All secrets (upstream key, JWT secret, SMTP credentials) live only in a server-side environment file.

V. Implementation

The Android application is written in Kotlin using Jetpack Compose (Material 3), Hilt, Room, DataStore, Retrofit/OkHttp, and sherpa-onnx 1.12.1. On-device models: SenseVoice-Small int8 (~230 MB) for ASR, Silero VAD v5 for gating, CAM++ (28 MB) for speaker embeddings, optional GTCRN (0.5 MB) for denoising. Features include quick notes, meeting mode with real-time speaker-labeled transcription, live AI summaries during recording, playback with seek and speed control, to-do extraction and a task center, Markdown/TXT export, PIN app lock, AI Q&A with streaming (SSE) responses and persisted history, mermaid-based mind maps rendered offline in a WebView, AI weekly reports, WebDAV backup, and e-mail-code login with quota display. The relay service is deployed on an entry-level cloud instance (2 vCPU/1 GB) behind systemd with automatic restart; nginx with a Let's Encrypt certificate provides HTTPS termination, and the service is live at https://api.ai-note.top.

VI. Experiments

Experimental platform. All experiments use the released model files (SenseVoice-Small int8/fp32, CAM++, GTCRN) and were executed with a Python evaluation harness built on sherpa-onnx 1.13.4 that loads the same model files and re-implements the exact segmentation and clustering algorithms of the Android application (which runs sherpa-onnx 1.12.1 on an iQOO Neo 8, Snapdragon 8+ Gen 1, Android 16). Inference used 2 threads on a desktop CPU; absolute timing figures therefore characterize the computational cost of the models, while relative comparisons between configurations transfer to the mobile deployment. Because the Python binding of sherpa-onnx does not expose per-frame VAD decisions, Experiment 1 emulates the VAD gate with an RMS-energy gate (25 ms frames, 10 ms hop, threshold derived from the noise floor); all remaining buffering and cutting logic is identical to Section IV-A. All CER values are character error rates computed after removing punctuation and whitespace.

A. Datasets

- **D1 (clean speech):** 32 clips (9–40 s each) of read speech recorded in a quiet room, each with a manually verified reference transcript.
- **D2 (noisy speech):** 31 clips (7–39 s each) recorded with audible background noise (street, cafeteria, fan), with manually verified transcripts.
- **D3 (multi-speaker):** 31 conversations in two subsets: 11 self-recorded conversations (conv*, predominantly two speakers; one clip in fact contains a single speaker) with hand-annotated* `(start, end, speaker)` *triples, and 20 clips (ali*) cut from the test split of the public AliMeeting meeting corpus (M2MeT challenge [13], used under its CC-BY-NC-4.0 license for academic research only): 12 three-speaker and 8 four-speaker far-field clips of 90–150 s each, taken from the first channel of the 8-channel recordings, with the official RTTM annotations converted to the same label format. D3 thereby covers real multi-party meetings, not only scripted two-speaker dialogue.

B. Experiment 1 — Segmentation Parameters

Setup: grid search over pre-roll ∈ {0, 0.3, 0.6, 1.0 s} × pause-cut threshold ∈ {0.4, 0.6, 1.0 s} on D1 (32 clips), using the int8 model and a 30 s maximum segment length (larger than the app's 10–15 s cap, so the silence-driven trade-off is isolated). **Metrics:** CER of the concatenated segment hypotheses against the reference; onset-miss rate (fraction of segments whose first aligned character falls behind its expected position in a global character alignment); latency estimate = pause-detection waiting time + measured per-segment decode time.

Pre-roll (s)	Pause-cut (s)	CER (%)	Onset-miss rate (%)	Latency (s)
0	0.4	4.17	2.60	1.65
0	0.6	3.47	4.69	3.51
0	1.0	3.22	0.00	4.94
0.3	0.4	3.48	1.56	1.77
0.3	0.6	3.37	4.69	3.60
0.3	1.0	3.22	0.00	4.94
0.6	0.4	3.56	1.56	1.86
0.6	0.6	3.23	3.12	3.55
0.6	1.0	**2.98**	**0.00**	5.16
1.0	0.4	3.66	1.56	1.86
1.0	0.6	3.25	3.12	3.71
1.0	1.0	3.04	0.00	5.13

Two observations stand out. First, segmentation does not materially degrade recognition: across all twelve configurations the CER stays within a narrow 2.98%–4.17% band, confirming that VAD-gated, pre-roll-compensated cutting preserves the accuracy of whole-utterance decoding. Second, the two parameters control different failure modes. The pause-cut threshold dominates both the onset-miss rate and the latency: a 1.0 s threshold eliminates onset misses entirely (0.00% at every pre-roll) but pushes the latency to ≈5 s, while a 0.4 s threshold cuts the latency to ≈1.7–1.9 s at the cost of up to one point of CER (0.3–1.0 points depending on pre-roll). The pre-roll buffer refines accuracy at a fixed threshold: at the 0.6 s operating point, adding 0.6 s of pre-roll reduces the CER from 3.47% to 3.23% and the onset-miss rate from 4.69% to 3.12%, and extending it further to 1.0 s yields no additional gain (3.25% / 3.12%). This validates the app's 0.6 s default: it captures essentially all of the recoverable onset context without inflating segment length. The residual onset misses at the 0.6 s pause setting stem from very short interjections whose leading phonemes fall below the VAD gate even with pre-roll; the engineering conclusion is that the 0.6 s/0.6 s operating point offers the best balance, while latency-tolerant scenarios (e.g., lecture archiving) can switch to a 1.0 s pause for the absolute best CER and zero onset truncation.

C. Experiment 2 — Online Speaker Clustering

Setup. D3 (31 conversations, expanded from the original 11 two-speaker recordings with 20 real three-to-four-speaker AliMeeting clips; see Section VI-A). Four online variants of the Section IV-B pipeline are compared, all driven by the same VAD-cut segments and CAM++ embeddings: a fixed-threshold

baseline ($\tau = 0.6$); variant A, adding the duration-aware rules and continuity prior; variant B, further adding bounded-delay re-decision ($N = 3$, $\delta = 0.05$); and the adaptive threshold paired with variant A and with variant B respectively. The offline baseline of the original experiment (a self-implemented uniform-window baseline, non-pyannote: uniform 2 s windows, CAM++ embeddings, post-recording agglomerative clustering with average linkage and cosine distance) is retained for the two-speaker subset. **Evaluation protocol.** For each file, predicted clusters are matched to ground-truth speakers by the Hungarian algorithm and the attribution accuracy is weighted by segment duration; every aggregate number below is the arithmetic mean of these per-file duration-weighted accuracies over the 31 files (subset means are given where relevant).

Ablation.

Variant	Weighted accuracy (%)	Mean predicted clusters	Δ vs. baseline (pp)
Baseline (fixed threshold)	68.8	7.1	—
A: duration-aware + continuity	71.9	4.0	+3.0
B: A + bounded-delay re-decision (final online)	**76.6**	**3.6**	**+7.8**
A + adaptive threshold	71.7	4.1	+2.8
B + adaptive threshold (component analysis, not adopted)	77.4	2.9	+8.6

The fixed-threshold baseline over-clusters badly, predicting 7.1 clusters against 1–4 true speakers, because short and boundary-straddling segments keep opening spurious clusters. The duration-aware rules are the first win: they cut the mean cluster count to 4.0 and add 3.0 points. The bounded-delay DP re-decision is the second: re-examining the recent uncommitted segments with a switch penalty raises accuracy to 76.6% and brings the mean cluster count to 3.6, close to the true speaker counts, at the price of a fixed three-segment attribution delay. Variant B with the fixed threshold of 0.6 (no adaptation) is therefore adopted as the final online configuration: 76.6% overall (conv 76.6%, ali 76.5%), 7.8 points above the fixed-threshold baseline, with gains consistent across subsets, i.e., the method transfers from scripted two-speaker dialogue to real far-field meetings. Adding the adaptive threshold on top of variant B yields a further +0.9 points (77.4%; conv 77.1%, ali 77.6%), and its main effect is cluster suppression rather than reassignment: the mean predicted cluster count drops from 3.6 to 2.9, the closest to the true speaker counts among all variants. The per-file effect is not monotone, however: while several files improve sharply, two degrade noticeably (ali13 −21.9 points, ali12 −14.8 points). Given the marginal, non-uniform gain and the extra running-distribution state the component introduces, we do not adopt it; it is retained as a rejected marginal component and analysed below (survivor-bias analysis). One measurement note: the evaluation harness retains only segments of at least 1 s, so the sub-second inheritance rule fires rarely in this evaluation; its effect is larger on-device, where VAD segments can be shorter. On the two-speaker subset the online method also remains far above the offline uniform-window baseline (72.6% vs. 46.3%, mean over the 11 conv files, at 5.6 s vs. 6.8 s of total computation): uniform 2 s windows carry too little speech for stable embeddings and routinely straddle speaker turns, so the baseline predicts 7–12 clusters for two-speaker conversations. The caveat from the original experiment still applies: this offline system is a uniform-window pipeline running under the same mobile constraints, not the full pyannote pipeline; a segmentation-model-

based offline system may match or exceed the online accuracy, though its mandatory post-recording latency and its incompatibility with a single-threaded mobile executor are unaffected by this caveat.

Hyperparameter sensitivity. Variant B was swept over lookahead $N \in \{3, 5, 7\}$ × switch penalty $\delta \in \{0.03, 0.05, 0.08, 0.12\}$ (overall weighted accuracy, %):

N \ δ	0.03	0.05	0.08	0.12
3	74.6	**76.6**	76.4	75.1
5	74.6	**76.6**	76.4	75.5
7	74.6	**76.6**	76.4	75.5

$\delta = 0.05$ is optimal and sits on a plateau ($\delta = 0.08$ is only 0.17 points lower), so the choice is not fragile; accuracy already saturates at $N = 3$, which is therefore selected for its lowest attribution delay.

Embedding model selection (a negative result). We challenged CAM++ with ERes2Net-base (512-d), a stronger model on public speaker-verification benchmarks, keeping variant B fixed at $N = 3$, $\delta = 0.05$:

Embedding	Weighted accuracy (%)	Embedding time per segment (ms)	Embedding RTF
CAM++ (192-d)	**76.6**	**238**	**0.044**
ERes2Net-base (512-d)	74.3	539	0.101

The challenge failed on both axes: ERes2Net is 2.3 points less accurate under our online protocol and 2.3× slower per segment. Higher benchmark scores did not transfer to VAD-driven segment clustering on meeting audio, and the extra cost is material on a single-threaded mobile executor. CAM++ therefore remains the on-device embedding model; this negative result is the selection evidence.

Post-meeting refinement. Taking the online partition as initialization, the cached embeddings are re-clustered after the recording stops (Section IV-B, item 4). The final configuration refines the variant-B fixed-threshold online partition (76.6%) and, one configuration note for reproducibility, merges clusters whose centroids exceed cosine similarity 0.6, the same threshold used online. Two independent experiment runs reproduce the identical result below.

Method	Weighted accuracy (%)	Mean clusters	Refinement time
Online variant B (fixed threshold, final online)	76.6	3.6	—
+ post-meeting refinement (final)	**79.7**	3.6	0.021 s per segment (mean)

Because the embeddings were computed online and cached, refinement costs a mean of 0.021 s per segment and recovers another 3.1 points (79.7%), consistently on both subsets (conv 76.6%→79.9%, ali 76.5%→79.6%). An honest limitation: the online stage already misses the true speaker count in 21 of 31 files (mean absolute error 1.10 speakers), and even after refinement the count is wrong in 14 of 31 files (mean absolute error 1.00 speaker), so we still do not claim reliable automatic speaker-count detection: the silhouette-constrained k improves attribution far more reliably than it counts speakers.

Adaptive threshold: a survivor-bias analysis. The adaptive threshold was motivated by the expectation that acceptance statistics would track each meeting's acoustic conditions. The experiment

refutes its standalone value: variant A with the adaptive threshold scores 71.7%, no better than variant A alone (71.9%). Inspection reveals a survivor bias: the running distribution is computed only over *accepted* similarities, which concentrate at high values, so a threshold fitted to them can only drift upward; meanwhile the segments that actually cause errors (short segments whose mid-range similarities fall below the acceptance band) never enter the accepted sample and cannot pull the threshold down. The threshold thus tightens exactly where it should loosen. Combined with variant B, the adaptive threshold appears to earn a small gain: the B + adaptive configuration reaches 77.4% overall (conv 77.1%, ali 77.6%), +0.9 points over B with a fixed threshold, and cuts the mean predicted cluster count from 3.6 to 2.9, i.e., in combination its role is chiefly to suppress spurious cluster creation inside the DP re-decision (the threshold enters both the new-cluster emission cost and the assignment test). However, the gain is not per-file monotone (ali13 and ali12 degrade by 21.9 and 14.8 points respectively), a +0.9-point aggregate shift is not robust at n = 31, and the component adds running-distribution state together with its cold-start and tuning surface. We therefore do not adopt it in the final configuration, which keeps the fixed threshold of 0.6, and report it as a rejected component with a marginal aggregate gain, known per-file regressions, and a mechanistic survivor bias — not a free lunch.

Limitations and failed attempts. Four limitations bound these results. First, speaker-count estimation is unreliable (14 of 31 files miss the true count after refinement; mean absolute error 1.00 speaker, above). Second, all experiments ran on a single desktop platform and on-device validation used a single phone model; absolute timings will vary across devices. Third, overlapped speech remains unsolved: simultaneous talkers confuse both recognition and embedding, and no variant models overlap. Fourth, D1/D2 are single-speaker self-recorded sets (32/31 clips); public-corpus expansion is future work. Two failed attempts delimit this third point honestly. A second refinement pass with multi-scale re-embedding (concatenating adjacent same-cluster segments up to 10 s and re-extracting embeddings) *reduced* accuracy by 1.5 points while costing 4.3 s per conversation, because a concatenated segment that contains a misassigned turn lets the longer embedding amplify the error. Likewise, dropping the frames flagged as overlapped by a pyannote segmentation model before embedding extraction reduced overall accuracy by 2.0 points: on high-overlap files the surviving speech is too sparse (ali15, with 40% overlapped frames, saw its embedding quality collapse), so overlap handling needs a finer-grained treatment than frame dropping. Future work accordingly includes a learned segmentation front-end used for overlap-*aware* assignment rather than frame deletion (e.g., pyannote.segmentation [6]) where the compute budget allows, score normalization (s-norm) for threshold-free assignment, explicit overlap modeling, and a DER-matched comparison against the full pyannote pipeline.

D. Experiment 3 — Model Configuration Trade-offs

Variables: int8 vs. fp32 ASR; GTCRN denoising on/off; evaluated on D1 and D2 jointly (63 clips).
Metrics: CER (reported separately for clean D1 and noisy D2), real-time factor (RTF = decode time / audio duration), and peak process memory. Battery drain was measured on-device and is reported in Section VI-E.

Configuration	CER clean (%)	CER noisy (%)	RTF	Model size (MB)
int8	2.71	3.77	**0.344**	**230**
fp32	**2.54**	4.06	0.380	936
int8 + GTCRN	2.81	3.91	0.599	230 + 0.5

The int8 model matches the fp32 model in accuracy (the mean CER over all 63 clips is 3.23% vs. 3.28%, and on the noisy subset int8 is nominally *better* at 3.77% vs. 4.06%) while running 10% faster (RTF 0.344 vs. 0.380) from a model file one quarter of the size (230 MB vs. 936 MB). The small CER differences in both directions are within per-clip variance, so quantization is effectively accuracy-neutral on this data, and int8 is confirmed as the right mobile default. GTCRN denoising, by contrast, did not pay off on these recordings: it slightly *increased* the CER on both subsets (2.81% / 3.91%) while raising the RTF by 74% (0.599), because the enhancement pass costs as much as the recognition itself and SenseVoice is already robust to the moderate noise levels in D2. The engineering conclusion is to keep denoising as an opt-in for harsh acoustic environments rather than a default. Two measurement caveats apply: (i) RTF was measured on a desktop CPU with 2 threads, whereas the app serializes decoding on a single-threaded executor, so absolute on-device RTF will be higher while the relative ordering of configurations is preserved — these RTF figures characterize model compute only, and on device a 66-minute continuous full-load run completed without backlog (Section VI-F), implying end-to-end RTF < 1; (ii) peak process RSS was nearly identical across configurations (≈1.80 GB) because all configurations ran in one long-lived process dominated by the ONNX Runtime arena, so we report model file size as the meaningful memory differentiator.

Large-model reference (Paraformer-large). To calibrate how far the on-device small model is from a large model, we additionally decoded D1 and D2 with Paraformer-large (int8) through the identical segmentation pipeline:

Model	CER clean D1 (%)	CER noisy D2 (%)	RTF (D1 / D2)
Paraformer-large int8	3.12	11.2	0.286 / 0.267
SenseVoice-Small int8	**2.71**	**3.77**	0.342 / 0.346
SenseVoice-Small fp32	2.54	4.06	0.384 / 0.376

On clean speech the on-device small model effectively ties the large model (2.71% vs. 3.12%; fp32 2.54%). The noisy-set gap (3.77% vs. 11.2%) is large but confounded, and we report both readings: under the shared pipeline the energy gate over-segments noisy audio, and Paraformer is markedly more sensitive to fragmented segments than SenseVoice: decoding the worst file whole instead of segmented drops its CER from 46.3% to 0.0% (two further files: 30.9%→7.3% and 34.9%→16.3%). The 11.2% figure is therefore a pipeline-matched number, not Paraformer's intrinsic accuracy; the honest conclusions are (i) on clean speech the quantized small model loses nothing to a model several times its size, and (ii) SenseVoice's robustness to our segmenter under noise is itself an argument for the small-model choice. Paraformer's peak memory in this harness was ≈0.42–0.45 GB RSS.

E. On-Device Stability and Power Test

A continuous meeting-mode recording was performed on the iQOO Neo 8 (Snapdragon 8+ Gen 1, Android 16) for 12 minutes with live transcription, online speaker clustering, and AI minutes generation

all active; speech was played from an external speaker to simulate a real meeting. The application did not crash and did not lose audio, and the real-time subtitle view remained responsive throughout. Process memory sampled every 30 s grew from 565 MB to 604 MB over the run (+6.9%), a shallow, approximately linear drift attributable to the growing recording/transcript buffers rather than to a leak; no unbounded growth was observed. Battery level, read from the fuel gauge before and after the run, did not change (34% at both ends), i.e., the total drain of the full audio pipeline was below the 1% gauge resolution over 12 minutes. The 60-minute endurance run that this profile motivated is reported in Section VI-F.

F. Endurance and Stability Test (60-Minute Full Load)

A 66-minute endurance run was performed on the iQOO Neo 8 (Snapdragon 8+ Gen 1, Android 16, debug build v1.3). Method, stated plainly: a 66-minute real meeting from the AliMeeting corpus was played over PC loudspeakers and the phone, placed nearby, recorded and transcribed it continuously in meeting mode, an acoustic replay chain used only as a stability workload, not for accuracy evaluation. Runtime telemetry (battery, battery temperature, process RSS, CPU) was sampled once per minute.

Results. The app recorded and transcribed continuously for 65 min 33 s with zero crashes and zero restarts; the recorded audio landed intact on disk at 126.57 MB against a theoretical 126.7 MB, i.e., no audio was lost. Battery dropped from 56% to 53% at recording stop (3% net over the full run, at the fuel gauge's 1% resolution). Battery temperature rose from 37.6 °C to 41.6 °C late in the recording (recording-period peak 43.0 °C) and settled back to about 41 °C within minutes of stopping. Process RSS, once model loading completed, plateaued at ≈610 MB and then climbed with a decreasing growth rate to ≈0.84–0.90 GB just before stopping (the expected accumulation of recording, transcript, and embedding state rather than an unbounded leak) and fell back to ≈440–600 MB once the pipeline idled. The per-minute log contains a single isolated RSS sample of 3.2 GB one minute before stopping, which returned to normal at the next sample; as it coincided with no crash, no audio loss, and no user-visible stall, we read it as a transient (e.g., GC-time accounting) rather than a stability defect, but we report it for completeness.

Honest scope: this is a single-device, replay-chain test. It establishes that the full pipeline survives an hour-scale meeting on a commodity flagship; a multi-device, live multi-speaker endurance and accuracy study remains future work.

G. Experiment 6 (preliminary) — Domain Adaptation and Iteration of an On-Device Meeting-Minutes Model

Motivation. The relay architecture leaves one privacy gap: transcript text crosses the network when the user requests minutes. On-device LLM summarization was previously listed as future work; this experiment turns it into a measured result and answers three questions: (i) whether a three-layer data recipe can teach a 1.7B model the domain task "speaker-labeled transcript → minutes"; (ii) whether a defect-driven loop (audit failures, generate targeted data, retrain, re-evaluate) works in practice; (iii) whether the adapted model still works after quantization to a phone-carriable size.

Dataset. The author built a meeting-minutes SFT dataset in three layers (table below). Layer 1 takes the AliMeeting clips of Section VI-A (ali01–20), transcribes them segment by segment with the same on-device SenseVoice-Small int8 model, and merges segments by speaker into a "Speaker N: …" transcript. The ASR noise is left uncorrected on purpose, because this is the input distribution the on-device model will actually see. DeepSeek (deepseek-chat) wrote the summaries under a faithful-to-transcript, no-fabrication prompt, in two task shapes (a ≤150-character brief, and minutes plus a to-do list). Layer 2 is DeepSeek-synthesized meetings across 16 scenario types (study groups, project reviews, family meetings, customer-service coordination, among others), each call producing the transcript and both summaries at once, with required colloquial interruptions and digressions. Layer 3 derives negatives from layer 1 (truncated, single-speaker-collapsed, and character-perturbed variants, ten each), teaching the model to state "insufficient information" instead of forcing minutes. Round 1 has 350 training and 30 evaluation samples. Round 2 adds 161 targeted samples (all synthetic, no AliMeeting derivatives) against the three defects found in round 1, plus a 15-sample targeted evaluation set (five per target, disjoint from training); every round-2 sample passed digit-by-digit traceability checking against its transcript.

Layer	Source	Training samples	Purpose
1. Real-meeting distillation	AliMeeting clips + on-device SenseVoice transcription + DeepSeek summaries	20 (10 meetings × 2 tasks)	anchor the real input distribution, ASR noise included
2. Synthetic expansion	DeepSeek-generated, 16 scenario types	300 (150 meetings × 2 tasks)	scenario coverage; tagged as synthetic
3. Negatives and boundary	derived from layer 1: truncated / monologue / perturbed	30 (10 each)	teach honest "insufficient information"
Round-2 targeted	all synthetic, three targets	161	no-todo 49 + no-owner 48 + number-dense 64

Licensing: the AliMeeting audio and annotations are CC-BY-NC-4.0 (academic research only, no commercial use) [13], and the derived transcripts and summaries inherit that constraint. The DeepSeek-distilled data are used for this research only; any public release of the dataset requires re-checking DeepSeek's terms of service on distillation and commercial use.

Training and quantization. Base model Qwen3-1.7B [18], LoRA rank 16 [17], the ms-swift framework [19], a single A10 GPU. Round 1: 350 samples × 3 epochs, 66 steps, 8 min 55 s, training loss 3.08 → 1.09. Round 2: 511 samples (350 + 161) × 3 epochs, 96 steps, final loss 1.04. For deployment the LoRA weights were merged back into the base model (bf16, 3.3 GB), converted to GGUF, and quantized to Q4_K_M with llama.cpp [20]: 1.03 GiB, 68% smaller than the f16 conversion. A smoke test on a desktop CPU generated minutes from a real meeting transcript at 5.7 tokens/s with well-formed output. Inference speed, memory, and battery drain on the phone have not been measured; that measurement is scheduled after the application integrates llama.cpp over JNI.

Round-1 audit: three defects and one data truth. Manual audit of the round-1 outputs found three defect classes, all preserved in the raw generations. (i) Fabricated to-dos: for a pure experience-sharing meeting whose opening line states that no tasks will be assigned (sample r2a050), the model still invented four to-dos and distributed them across Speakers 1–4. (ii) Fabricated owners: where the transcript says the tasks will be claimed in the group chat later, the model attached the to-dos to

concrete speakers. (iii) Number drift: an electricity fee of 240 written as 250, minor numbers dropped. The root cause of class (i) was then found in the data itself: among the 350 round-1 training samples, the count of no-todo examples is zero. The model did not invent to-dos because it is weak; it was never taught that a meeting may legitimately produce no action items. Model defect and data defect confirm each other, and that is the central data-engineering finding of this experiment: for a small domain model, behavioral defects should be traced back to the data first.

Round-2 targeted repair. Target A generates pure-discussion meetings with no action items, with a server-side check forcing the to-do section to start with "none". Target B generates meetings with clear items but no assignment, checking that every to-do is labeled "owner: TBD" and names no speaker. Target C generates number-dense meetings (budgets, schedules, quotations, inventories); single-pass generation drifted too often (5 of 32 passed) and was replaced by a two-step pipeline (transcript first, then a low-temperature summary pass) plus server-side digit-by-digit verification, with drifting samples discarded. After retraining, both model versions were manually audited on the 15-sample targeted evaluation set (metrics: token-level digit fidelity is the share of output digit strings individually traceable to the transcript, sample-level full fidelity the share of minutes with zero untraceable digits; five samples per target, judged manually):

Metric (n = 5 per target)	v1 (350 samples)	v2 (511 samples)
Target A: no fabricated to-dos (clean rate)	20%	80%
Target B: no fabricated owners (clean rate)	20%	80%
Target C: token-level digit fidelity	94%	98% (119/121)
Target C: sample-level full digit fidelity	80%	60%

Note: n = 5 per target; Clopper–Pearson 95% exact intervals for the sample-level rates are 20% → [0.5%, 71.6%] and 80% → [28.4%, 99.5%]; at this sample size the direction is credible but the precise numbers await an expanded evaluation set.

Two gains and one regression, all reported. On target A, v2 answers "to-dos: none" on four of five; on the same r2a050 input where v1 invented four to-dos, v2 states that the meeting assigned no tasks. On target B, v2 learned to render "to be claimed in the group chat" as "owner: TBD" instead of naming a speaker. The sample-level regression on target C comes from a single failed sample, r2c023: the transcript gives a phone number in two pieces ("starts with 138", "ends in 7421") and v2 stitched them into the complete number 1387421 in a to-do item. This is a partial-information concatenation error rather than fabrication from nothing, but it misleads the user all the same; the next data round will add concatenation negatives for phone numbers and dates.

Honest boundaries. (i) With five evaluation samples per target, the comparison establishes direction, not precise numbers; an expanded set is needed before claiming accuracies. (ii) The small model's minutes are below cloud DeepSeek quality; its role is the offline draft in a free-offline/paid-cloud split, not a replacement. (iii) The 1.03 GiB size and the desktop smoke test show that the model runs, not that it runs well on a phone; on-device figures remain to be measured. (iv) The licensing constraints above keep both the dataset and the adapted model inside a research context.

VII. Conclusion and Future Work

We presented AI Note, a privacy-preserving intelligent note-taking system whose audio processing is entirely on-device, together with a streaming-simulation segmentation strategy and an online speaker clustering method that remove the two largest latency sources of the conventional pipeline. Experiments on the evaluation datasets (two purpose-built corpora plus a multi-speaker set combining self-recorded dialogue with public meeting clips) show that the segmentation strategy preserves whole-utterance recognition accuracy across the full parameter grid (2.98%–4.17% CER, with zero onset truncation at a 1.0 s pause threshold), that online segment-level speaker clustering reaches 76.6% attribution accuracy across 31 conversations including real three-to-four-speaker meetings, 7.8 points above the fixed-threshold baseline and far above an offline agglomerative baseline, rising to 79.7% after a millisecond-scale post-meeting refinement, while adding no latency when the recording stops, and that int8 quantization is accuracy-neutral relative to fp32 (3.23% vs. 3.28% mean CER) at one quarter of the model size. A sixth experiment further adapts a 1.7B LLM into an on-device meeting-minutes model through a three-layer data recipe and one defect-driven iteration: the clean rate on fabricated to-dos and fabricated owners rises from 20% to 80% at 98% token-level digit fidelity, and the quantized 1.03 GiB model runs on a plain CPU, closing the text-level privacy gap at draft quality (small-sample and quality boundaries are stated in Section VI-G). Future work includes overlap-aware segmentation with a learned front-end (e.g., pyannote.segmentation) and score normalization (s-norm) for speaker assignment, a DER-matched comparison against the full pyannote pipeline, reliable speaker-count estimation (current estimates still miss the exact count in 14 of 31 meetings), adaptive clustering thresholds that escape the survivor bias identified in Section VI-C, on-device benchmarking of the minutes model once integrated over JNI (speed, memory, battery), a third data round targeting partial-information concatenation errors with an expanded per-target evaluation set, and hotword customization for domain vocabulary.

Post-publication note. The system and experiments described here reflect the v1.0 architecture snapshot of early August 2026. As of this writing the product has iterated to v1.4, adding a second recognition engine (Paraformer), DFN3 extreme denoising, FireRedASR offline refinement, a knowledge star-map visualization, and the productization of Section VI-G's on-device path (offline minutes and live offline meeting summaries are shipped). These iterations do not alter the architecture or the experimental conclusions of this paper.

AI-Assistance Disclosure

During the preparation of this work, the author used AI-based assistants (large-language-model tools) for language polishing and for assistance in drafting portions of the manuscript text. The system design and implementation, the experimental design, data collection and annotation, and all experimental measurements and results reported in this paper were carried out entirely by the author. The author reviewed and edited all AI-assisted content and takes full responsibility for the integrity and accuracy of this publication.

References

[1] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," in *Proc. ICML*, 2023. [2] Z. Gao, S. Zhang, I. McLoughlin, and Z. Yan, "Paraformer: Fast and accurate parallel transformer for non-autoregressive end-to-end speech recognition," in *Proc. Interspeech*, 2022. [3] FunAudioLLM, "SenseVoice: Multilingual voice understanding model," 2024. [Online]. Available: https://github.com/FunAudioLLM/SenseVoice [4] Next-gen Kaldi, "sherpa-onnx: Speech-to-text, text-to-speech, and speaker recognition using onnxruntime," 2022–2025. [Online]. Available: https://github.com/k2-fsa/sherpa-onnx [5] Silero Team, "Silero VAD: pre-trained enterprise-grade voice activity detector," 2021. [Online]. Available: https://github.com/snakers4/silero-vad [6] H. Bredin and A. Laurent, "End-to-end speaker segmentation for overlap-aware resegmentation," in *Proc. Interspeech*, 2021. [7] H. Wang et al., "CAM++: A fast and efficient network for speaker verification using context-aware masking," in *Proc. Interspeech*, 2023. [8] A. Zhang, Q. Wang, Z. Zhu, J. Paisley, and C. Wang, "Fully supervised speaker diarization," in *Proc. ICASSP*, 2019. [9] D. Dimitriadis and P. Fousek, "Developing on-line speaker diarization system," in *Proc. Interspeech*, 2017. [10] X. Rong, T. Sun, X. Zhang, et al., "GTCRN: A speech enhancement model requiring ultralow computational resources," in *Proc. ICASSP*, 2024, pp. 971-975. [11] A. Vaswani et al., "Attention is all you need," in *Proc. NeurIPS*, 2017. [12] DeepSeek-AI, "DeepSeek-V3 technical report," 2024. [13] F. Yu, S. Zhang, Y. Fu, L. Xie, S. Zheng, Z. Du, W. Huang, P. Guo, Z. Yan, B. Ma, X. Xu, and H. Bu, "M2MeT: The ICASSP 2022 multi-channel multi-party meeting transcription challenge," in *Proc. ICASSP*, 2022. [14] Google, "AI Edge Gallery: On-device generative AI reference application," 2025. [Online]. Available: https://github.com/google-ai-edge/gallery [15] altic-dev, "FluidVoice: On-device dictation for macOS," 2025. [Online]. Available: https://github.com/altic-dev/FluidVoice [16] cactus-compute, "needle: A 45M-parameter on-device tool-calling model," 2026. [Online]. Available: https://github.com/cactus-compute/needle [17] E. J. Hu, Y. Shen, P. Wallis, et al., "LoRA: Low-rank adaptation of large language models," in *Proc. ICLR*, 2022. [18] A. Yang et al., "Qwen3 technical report," arXiv:2505.09388, 2025. [19] ModelScope, "ms-swift: Scalable lightweight fine-tuning and inference framework for LLMs," 2023. [Online]. Available: https://github.com/modelscope/ms-swift [20] G. Gerganov et al., "llama.cpp: LLM inference in C/C++," 2023. [Online]. Available: https://github.com/ggml-org/llama.cpp