EdgeTalk: Agentic Speech-Text Models for Edge Devices

1st Megha Sharma
Amazon AGI
Bangalore, India
meghshar@amazon.com

2nd Tushar Vatsal *
Amazon AGI
Bangalore, India
tvvatsal@amazon.com

2nd Prakhar Swarup * †
Dolby Laboratories
Bangalore, India
prkhrswrp@gmail.com

3rd Raghavendra Bilgi
Amazon AGI
Bangalore, India
rrbilgi@amazon.com

***Abstract*—We present EdgeTalk, a unified streaming speech-text model for on-device voice assistants, running on CPU-only edge hardware. Unlike chained automatic speech recognition (ASR) and natural language understanding (NLU) pipelines, EdgeTalk handles ASR, API invocation, and response generation within a unified 1.2B-parameter model. Joint speech-text-API training recovers errors that cascade in pipelines, improving turn-level API accuracy by 8.2% over a same-size chained variant (85.1% vs 76.9%). The model approaches the end-to-end task success of a 2.7x larger Whisper-1.5B + Qwen3-1.7B baseline (73.1% vs 77.3%) while achieving 1.5s per-turn latency through stacked inference optimizations (127% improvement in tokens per second with minimal accuracy loss). We find 1.2B (1B SLM + 200M encoder) is the maximum viable size for our target hardware and share deployment learnings including failure modes and latency-accuracy tradeoffs for edge agentic systems.**

***Index Terms*—speech-text models, streaming ASR, API calling, CPU-only edge inference, voice assistant**

I. Introduction

Conversational interfaces between users and AI systems have evolved from rigid command-style interactions to fluid, multi-turn agentic systems powered by large language models (LLMs) [1], as shown in Figure 1. These systems resolve user queries using API calls and are widely deployed as chat bots [2], customer service agents [3], and search agents [4]. However, deploying such systems involves high inference costs and latency, motivating edge-based alternatives that provide comparable experience at lower cost. Applications such as automotive and smart home assistants [5], [6] further benefit from edge deployment as it reduces data sharing with cloud services, improving user privacy.

To this end, we present **EdgeTalk**: a streaming speech-text model with up to 1.2B parameters (200M encoder + 1B small language model), equipped with conversational and API calling [7], [8] capability, optimized for CPU-only inference on edge hardware. EdgeTalk targets voice-enabled applications such as automotive assistants, entertainment systems, and Internet of Things (IoT) devices [9], constrained by memory, compute, and latency but without significant battery constraints. Extending EdgeTalk for mobile applications will require further work to reduce power usage.

*Equal contribution
†Work done while at Amazon

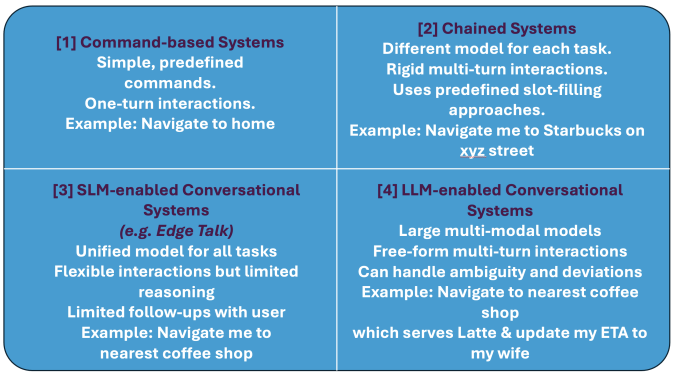


Fig. 1: Evolution of conversational interface in agentic systems with advancements in technology.

Challenges: Building an edge agentic system is non-trivial due to: (a) no or shared access to hardware accelerators; (b) model size constraints from limited memory; (c) offline operation without internet; (d) user privacy limiting data sharing [10].

Gaps: Existing speech-text agentic LLMs have 7B+ parameters [11], [12], making them unsuitable for edge. Smaller systems chain task-specific models (e.g., separate ASR + NLU + text-to-speech (TTS) [13]), compromising conversational flow and adding orchestration overhead. Recent work has demonstrated text-only function calling at edge scale [14], [15], but these do not handle speech as input. To our knowledge, no unified agentic (API-calling) speech-text model exists at sub-2B for CPU-only edge.

Contributions: We address the question: *How to build and deploy speech-text agentic models on edge devices?* Our jointly-trained 1.2B unified model improves turn-level API accuracy by 8.2% absolute over a same-size chained variant (85.1% vs 76.9%), and approaches the end-to-end (E2E) task success of a 2.7x larger baseline (73.1% vs 77.3%) at 1.5s turn latency on CPU-only hardware. We present: (1) evidence that joint training recovers cascaded errors that chaining propagates; (2) a multi-stage training pipeline achieving agentic adaptation in only 170M tokens (vs 3T for foundation training); (3) CPU-only optimizations yielding 127% tokens per second (TPS) increase with minimal accuracy loss; and (4) deployment-oriented learnings including failure mode analysis.

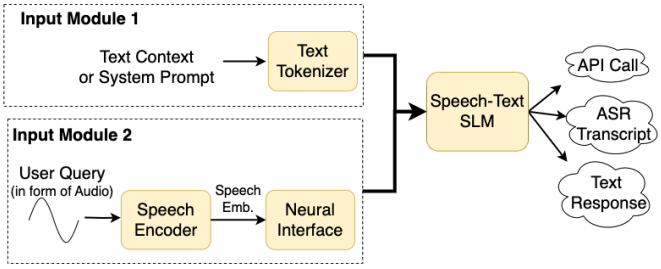


Fig. 2: High-level design diagram of EdgeTalk models

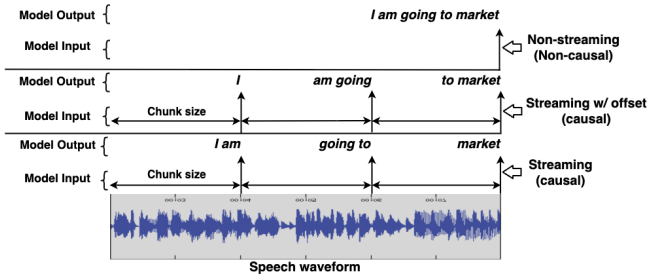


Fig. 3: Depiction of differences between streaming, streaming with offset and non-streaming speech processing. Both accuracy and latency increase in the same order due to delayed transcript prediction by model.

II. Related Work

Speech-text models use encoder-decoder architectures with varied training strategies: freezing components [16], co-training [17]–[19], or adapter-based updates. Most require 7B+ parameters, making them infeasible for edge. Smaller variants like Whisper [20] lack streaming behavior. In this work, we build streaming speech-text models at 1.2B and below.

Model optimizations: Post-training quantization methods such as GPTQ [21], SqueezeLLM [22], and SmoothQuant [23] reduce LLM inference cost to 3-8 bits with minimal accuracy loss. We use SmoothQuant combined with speculative decoding [24] and constrained decoding [25] to reduce latency.

API calling: Supervised fine-tuning (SFT) [8] and few-shot learning are common methods for enabling domain-specific API calling in LLMs. Error accumulation in multi-turn tasks is addressed by interleaving reasoning with execution results [26], [27]. We take a similar approach, feeding API return values back to the model to inform subsequent predictions.

III. Architecture & Design Choices

A. Components:

Figure 2 illustrates the 3 major components of EdgeTalk: (a) a speech encoder to process input speech signal, (b) a neural interface to project encoder output to text space, and (c) a small language model (SLM) to produce textual output.

Speech Encoder: Before passing through the encoder, speech input is resampled to 16kHz and converted to 128-dimensional mel-spectrogram features using a window size of 25ms and a hop size of 10ms. This gives us a feature frame rate of 100Hz which is reduced to 25Hz using stacking and down-sampling by a factor of 4. Our 200M encoder is based on the Conformer [28] architecture with 1 convolutional front-end filter and 18 blocks having d_model = 512 and d_ff = 2048. Training details are covered in Section V-A.

Neural Interface: It projects speech encoder output to SLM's input embedding dimension. It is a linear layer with input dimension = d_model for speech encoder and output dimension = SLM's token embedding dimension.

SLM: We use an internal decoder-only transformer [29] in 3 sizes (160M, 350M, 1B) with grouped query attention [30] (16 heads, group size 4) and a sentencepiece [31] tokenizer with vocabulary of 50K. With the 200M encoder, total model sizes are 360M, 550M, 1.2B respectively. We refer to these as S2A-360M/550M/1.2B (Speech-to-API) in later sections.

B. Design Choices

Streaming speech processing: Figure 3 contrasts non-streaming (wherein model has full right context) and streaming (no right context) which is an important consideration in encoder design. To minimize latency, we cannot wait for the entire user utterance before we start processing. Hence, we fix all convolution and attention operations as streaming.

Continuous neural interface: We project speech embeddings directly to the SLM's input space via a linear layer, avoiding information loss from discretizing speech into tokens.

Model sizes: To determine maximum feasible model size for target hardware (Section VII), we benchmark randomly initialized models of varying sizes on the target hardware and find that SLMs up to 1B parameters execute reliably. We then train EdgeTalk models with different parameter budgets to study how model size influences accuracy and latency.

Context window: EdgeTalk uses an 8K context window (fits up to ∼5 multi-turn exchanges), sufficient for task-oriented interactions. Longer conversations fall back to cloud models.

IV. Training Setup

We start from internal text-pretrained SLMs (trained on 3T tokens) in 3 sizes (160M, 350M, 1B) and train in two stages: (1) foundation model development (speech encoder pretraining and speech-text alignment), and (2) agentic task adaptation (streaming ASR SFT followed by agentic SFT). Total Token Budget: Speech-text pretraining - 2.1T tokens, streaming ASR SFT - 93B, and agentic SFT - 170M. Agentic adaptation requires orders of magnitude fewer tokens than foundation training, making domain adaptation cheap.

A. Datasets

Foundation Model: For speech pretraining, we use 8.3M hours of labeled and unlabeled speech from Librispeech [32], Multilingual Librispeech [33], CommonVoice [34], VoxPopuli [35], and in-house data. Speech-text pretraining re-uses these along with the internal text pretraining corpora.

Agentic Task: Training data used for streaming ASR SFT is a subset of labeled data used in speech pretraining for which word alignment information is available, totaling 1M hours of speech. For agentic SFT, we design 9 APIs specific

to an automotive AI assistant, capable of reading or altering the state of an automotive vehicle. Due to lack of real multi-turn conversations for these APIs, we create synthetic training data comprising 16K conversations generated using Mixtral-8x7B [36] model. Corresponding audios are generated using in-house TTS system, augmented with noise injection and SpecAugment [37], standard augmentations for acoustic variability. We create two test sets, A & B, both involving multi-turn conversations and multi-API interactions specific to those 9 APIs. Testset A contains 365 conversations each recorded and curated from a pool of 48 participants, while testset B contains 1965 synthetic conversations and corresponding audios using Claude 3 Sonnet [38] and in-house TTS system respectively.TTS speaker pools are mutually exclusive between train and test. We did not use Claude for training data generation to avoid train-test bias. While our foundation models are multilingual, our agentic task is English-only.

B. Metrics

We use word error rate (WER) for speech and speech-text pretraining stages. For agentic task, we evaluate at two levels:
Turn-level API accuracy evaluates correctness of API name, parameter names, and parameter values per user-agent exchange (turn). We use Mixtral-8x7B [36] as an LLM judge for semantic equivalence rather than strict AST matching [39], since parameter values often have multiple valid surface forms (e.g., "321 Main St" vs "321 main st, chicago") that AST would incorrectly penalize. We validated this judge against 100 human-annotated API turns, achieving 99% agreement (Cohen's κ=0.98).
Turn-level response accuracy evaluates semantic equivalence of agent responses using the same LLM judge setup, validated at 90% agreement (κ=0.80) on 100 human-annotated responses. We prefer this over BLEU [40] as it is less sensitive to minor lexical variations.
E2E task accuracy measures whether all APIs in a multi-turn conversation are invoked correctly without teacher forcing. This is the strictest metric, reflecting real deployment where errors accumulate across turns.
We do not report WER for the agentic task, as intermediate transcriptions serve as latent reasoning steps and are inconsequential as long as downstream API calls succeed.

V. Foundation Model Development

This section covers stage 1 of training: building speech and text understanding on top of the pretrained SLMs. Agentic task adaptation (stage 2) is covered in Section VI.

A. Speech Pretraining

We adopt a dual-mode framework [41] for encoder pretraining, allowing streaming/non-streaming switching at inference via attention masks without re-training. We train a 2B non-streaming teacher encoder and distill it into our 200M student. On CommonVoice [34] English test, the student achieves 17.97% WER (non-streaming) and 22.71% (streaming), vs 14.84% for Whisper-large-v2 (637M encoder, 1.5B total; non-streaming only). The gap is due to Whisper's 3x larger encoder.

B. Speech-Text Pretraining

TABLE I: Non-streaming ASR WER (↓) of speech-text pretrained models vs. Whisper-large-v2. Enc.: encoder size (M). t-o: Librispeech test-other [32], mls: MLS [33], cv: CommonVoice [34], v: Voxpopuli [35]. *-en: English partition.

	Enc.	**t-o**	**mls-en**	**cv-en**	**v-en**
Whisper-1.5B	637	**5.2**	**6.2**	**9.0**	**7.3**
1.2B	200	6.0	6.8	10.1	8.7
550M	200	6.7	8.1	12.3	11.0
360M	200	8.5	10.8	17.3	11.5

This stage aligns speech encoder and SLM representations using paired speech-text data. Neural interface and SLM are kept fully trainable, speech encoder is frozen and used in non-streaming mode. We retain the text continuation task at 50% weight to avoid catastrophic forgetting. We introduce streaming ASR task in adaptation stage (Section VI).
Table I shows our 1.2B model has 15% relatively worse ASR WER than Whisper, largely due to its 3x larger encoder. However, Whisper-like encoders are unsuitable for EdgeTalk due to size and non-streaming nature. We fine-tune our SLM for streaming task in the subsequent stage while encoder is frozen and used in streaming mode.

VI. Building Agentic System

A. Task Description

EdgeTalk supports multi-turn agentic conversations where user query is spoken. Figure 4 depicts a sample conversation. We achieve this by augmenting foundation models (Section V) with agentic capabilities by fine-tuning the SLM and neural interface on streaming ASR, API-calling and text response generation tasks. In each turn, the model first generates a transcription in streaming fashion. It then either generates an API call or asks a clarifying question. In case of an API call, return values are fed back to the model, which generates a text response rendered to the user via a separate TTS system.

B. Special Tokens

Since EdgeTalk models supports two modalities (speech and text) and multiple tasks, we use special tokens, shown in Figure 4, to disambiguate content from different sources present in SLM's context window. *User* and *Agent* symbols indicate start of user input and that of agent output respectively. *Transcript*, *API* and *Response* mark the beginning of task-specific output. API return values are preceded by *OBS*. Lastly, *EOS* marks end of generation by SLM.

C. Methodology

For streaming ASR, we process audio incrementally in chunks (sampled from 40ms to 2s during training) rather than waiting for the full audio. During training, each chunk's speech embeddings are interleaved with corresponding text and aligned using word-level timestamps from an in-house model. To reduce

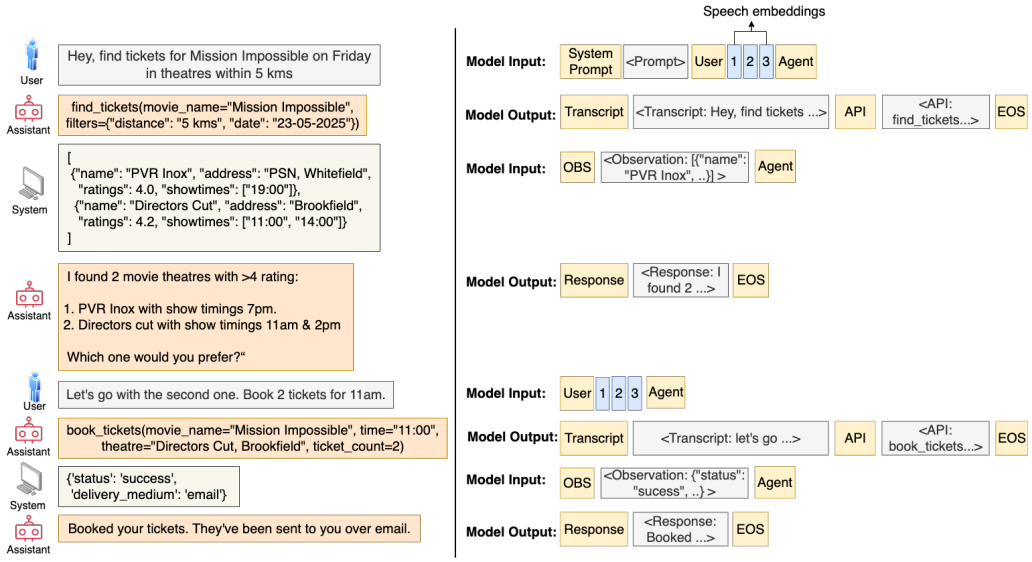


Fig. 4: **Left:** Example conversation with EdgeTalk. **Right:** SLM's corresponding context window including special tokens (marked in yellow).

errors, we delay text prediction by 240ms relative to audio (Figure 3), allowing the model to accumulate more context before prediction, improving WER at slight latency cost.

We adopt curriculum learning [42] by first training on single-turn streaming ASR, then gradually introducing API-calling and response generation (agentic SFT). For multi-turn training, N-1 turns are teacher-forced while training the N-th turn. Encoder is kept frozen; SLM and neural interface are trainable.

D. Results

Streaming ASR Evaluation: We evaluate stage 1 of curriculum learning using greedy decoding, computing WER by concatenating chunk-wise text output against reference. Table II shows results at fixed chunk sizes of 1 and 2 sec. As expected, WER improves with increased chunk and model size. Note that no open-source baseline exists at this scale for streaming speech-text models.

TABLE II: Streaming ASR WER (↓) for chunk sizes 1s and 2s. t-c/t-o: Librispeech clean/other, mls: MLS, cv: CommonVoice. *-en: English partition.

	Chunk	t-c	t-o	mls-en	cv-en
1.2B	1	2.9	7.8	9.7	16.9
	2	**2.7**	**7.2**	**9.1**	**16**
550M	1	5.4	11.6	13.9	22
	2	4.1	9.9	13	20.8
360M	1	6.8	13.4	15.3	23.9
	2	5	11.3	13.6	21.9

Turn-level API & Response Evaluation: We perform turn-wise evaluation on both multi-turn datasets and aggregate results across conversations. Table III presents turn-level results for EdgeTalk models compared against a chained baseline: Whisper-1.5B for ASR [20] + Qwen3-1.7B [43] for Text2API (T2A), fine-tuned till convergence on the same training set as EdgeTalk for fair comparison.

S2A-1.2B outperforms the 2.7x larger baseline on turn-level API accuracy on both testset A (85.1% vs 81.4%) and B (81.6% vs 79.6%). Although our 1.2B model has higher WER than Whisper on speech-text benchmarks (Table I), this degradation does not propagate to downstream API accuracy, showing benefit of effective joint training. Despite performing agentic SFT on augmented TTS audios (Section IV), S2A-1.2B achieves 85.1% API accuracy on human-recorded testset A, showing transfer to real speech within our task scope. The baseline achieves higher response accuracy than S2A-1.2B, due to stronger text generation capability of Qwen3 LLM.

Joint training is critical: To isolate the effect of joint training, we fine-tune the 1B SLM separately on Text2API and ASR tasks (ASR+T2A-1.2B row). This chained variant achieves only 76.9% API accuracy on testset A (8.2% drop vs S2A-1.2B), confirming that joint training recovers errors that cascade in a pipeline architecture. However, joint training yields diminishing returns with sub-500M models.

Size scaling: S2A 550M and 360M show 10% and 17% absolute drops in API accuracy on testset A compared to 1.2B (Table III), with similar trends on testset B. This shows only 1.2B model is viable for our task. Both smaller models trail the baseline significantly, indicating that joint training alone cannot compensate for insufficient model capacity.

E2E Evaluation: We report E2E task accuracy on testset A, which evaluates whether all APIs in a multi-turn conversation are invoked correctly without teacher forcing (i.e., each turn conditions on the model's prior outputs). Table IV shows that the baseline outperforms S2A (77.3% vs 73.1%). This reversal from turn-level results reflects error accumulation as turn-level evaluation uses teacher-forced preceding turns, while E2E lets errors propagate across turns. The baseline's advantage stems from Qwen3-1.7B being a reasoning model capable of self-correction, whereas our 1B SLM compounds errors across turns. The dominant failure modes (Figure 5) are missing API calls (48.4%) and incorrect parameter values (32.8%), while syntax errors are eliminated by jump-forward decoding

TABLE III: Turn-level API and Response accuracy (%) on testsets A (human-recorded) and B (synthetic). All models fine-tuned on same data. S2A = Speech2API; size includes encoder + SLM.

	Testset A		Testset B	
	API	**Resp.**	**API**	**Resp.**
Whisper+Qwen3-1.7B	81.4	**83.2**	79.6	**92.0**
ASR+T2A-1.2B	76.9	73.1	69.3	89.0
S2A-1.2B	**85.1**	78.2	**81.6**	88.9
S2A-550M	74.8	75.0	71.1	87.2
S2A-360M	67.8	63.0	66.1	83.5

TABLE IV: E2E task accuracy (%) on testset A (human-recorded). A conversation is successful when all reference APIs are invoked correctly without teacher forcing.

Model	**Task Accuracy (%)**
Whisper+Qwen3-1.7B	77.3
S2A-1.2B	73.1

(Section VII-B). Missing API calls occur when the model generates a response where an API was expected.

Statistical Significance: We assess accuracy gaps using bootstrap confidence intervals (10,000 resamples, n=242 conversations containing at least one API call). All API accuracy differences in Table III are statistically significant (95% confidence interval (CI) excludes zero) except S2A-1.2B vs the 2.7x larger baseline. The turn-level gap of 3.7% has 95% CI $[-0.8, 8.3]$. For E2E, since task success is binary per conversation (pass/fail), we use McNemar's test; the 4.2% gap yields p=0.13. Both are not significant, supporting our framing that S2A-1.2B approaches but does not match a system 2.7x its size. All other gaps, including the 8.2% joint training advantage and size scaling results, are statistically significant.

VII. Optimizations for Edge Hardware

A. Hardware Constraints

We use an automotive system-on-chip (SoC), comparable to SA8295P-class platforms [44]: 4 CPU cores, 68.2 GB/s memory bandwidth, and 600 INT8 GOPS, with hardware accelerators unavailable to our workload. Memory bandwidth bounds tokens per second (TPS), while computational throughput bounds context encoding (CE). These metrics are on par with mobile/automotive chips but 10 to 100$\times$ lower than high-end GPUs. Each optimization in Section VII-B targets a specific bottleneck: quantization reduces memory bandwidth pressure during decoding, speculative decoding amortizes autoregressive cost, and jump-forward decoding eliminates redundant computation for deterministic structured output.

B. Optimization Techniques

We present techniques to reduce SLM inference latency. We exclude speech encoder as it operates in streaming mode with real-time factor (RTF) < 0.05. Table V reports absolute and relative latency metrics, all relative to the FP16 1.2B baseline.

Size Variations: Smaller models (550M, 360M) offer large latency gains but at significant accuracy cost (>10%). We therefore apply all optimizations below to the 1.2B model only.

Quantization: We use INT8 precision for weights and activations to reduce memory bandwidth and leverage higher computational throughput of INT8 matrix multiplication. To handle activation outliers, we combine SmoothQuant [23] with dynamic quantization, achieving 73% increase in TPS and 42% decrease in CE relative to FP16 with minimal accuracy drop. We exclude the output vocabulary projection (`lm_head`) from quantization for accuracy reasons.

TABLE V: Latency and accuracy on target SoC (4 CPUs) at 1024-token context for benchmarking; actual per-turn context encoding (CE) is lower as prior turns are KV-cached. TPS: tokens per second. API Acc.: turn-level accuracy on testset A.

	CE (s)	**TPS**	**API Acc.(%)**
FP16 1.2B	7.20	13.7	85.1
Model size variations			
FP16 550M	2.64 (-63%)	35.3 (+158%)	74.8 (-10%)
FP16 360M	0.47 (-93%)	146.4 (+968%)	67.8 (-17%)
Stacked optimizations on 1.2B			
INT8 Quant	4.21 (-42%)	23.8 (+73%)	84.5 (-1%)
+ Spec. Decoding	4.21	30.3 (+121%)	84.5
+ Jump Forward	4.21	31.1 (+127%)	84.5

Speculative Decoding (SD): We use SD [24] with a quantized 160M draft and 1B target SLM, performing speculative sampling [45] with draft size 4 and rejection-based verification for lossless generation. This achieves 121% TPS increase over FP16 1.2B model. Acceptance rates vary by task: ASR 80%, API 83%, response 63%; structured outputs (APIs) benefit most due to predictable token patterns.

Jump-forward Decoding (JF): Since API is structured text, we create a finite state machine (FSM) of our API set and autocomplete API and parameter names once initial subwords are generated [25]. This yields 127% TPS gain over FP16.

VIII. Deployment Considerations

We evaluate EdgeTalk on the target automotive SoC, using only 4 of 8 available CPUs to reflect realistic deployment where remaining compute is reserved for infotainment rendering and other real-time tasks.

Baseline Deployability: The Whisper-1.5B + Qwen3-1.7B baseline (3.2B total) is impractical for our target deployment. First, Whisper is non-streaming, adding full decode latency to the post-speech wait. Second, loading baseline combination requires 2.7x more memory footprint than EdgeTalk, leaving less headroom for concurrent infotainment workloads. EdgeTalk's unified architecture avoids both limitations by design.

A. E2E Latency

While Table V reports CE and TPS at 1024-token context for benchmarking (actual per-turn CE is lower as prior turns are KV-cached), practitioners need conversation-level latency to assess UX. We derive E2E latency for a representative one-turn interaction below using average token counts observed in our evaluation data. The system prompt is pre-cached and the encoder (RTF < 0.05) processes audio faster than real-time, so neither is a bottleneck.

We decompose per-turn latency into sequential stages. Let t_{CE} = per-token context encoding time (CE@1024/1024) and R_{dec} = TPS, both measured in Table V:

Time to first token: $T_{\text{first}} = 1\text{s} + 25 \cdot t_{\text{CE}} + \frac{1}{R_{\text{dec}}}$

The model waits for one 1s audio chunk (producing 25 speech

embeddings at 25Hz after 4× downsampling), encodes them, and decodes the first transcript token.

End-of-speech to API: $T_{\text{API}} = \frac{5+N_{\text{API}}}{R_{\text{dec}}}$ where N_{API}=30 (FP16/INT8) or 10 (with JF).

After the user stops speaking, the model generates ∼5 residual transcript tokens (due to 240ms lookahead delay) and API tokens. Without JF, all 30 API tokens are generated; with JF, only ∼10 are generated as the FSM auto-completes the rest.

Total turn time: $T_{\text{turn}} = T_{\text{API}} + 20 \cdot t_{\text{CE}} + \frac{30}{R_{\text{dec}}}$

The API return (∼20 tokens) is encoded into context, then the model generates a ∼30 token response.

Table VI reports T_{first}, T_{API}, and T_{turn} for each optimization configuration, derived from Table V and validated against on-device inference. T_{first} overlaps with user speech and does not add to perceived latency. The user-perceived latency (T_{turn}) with INT8+SD+JF is ∼1.5s for a single turn accounting for streaming ASR, API calling, and response generation.

TABLE VI: E2E latency of S2A-1.2B on target SoC for one turn.

Metric	FP16	INT8	INT8+SD+JF
Time to first token (T_{first})	1.25s	1.14s	1.14s
End-of-speech to API (T_{API})	2.55s	1.47s	0.48s
End-of-speech to response (T_{turn})	4.88s	2.85s	1.51s

B. Failure Mode Analysis

We analyze EdgeTalk's failure modes in Figure 5 to understand where the system fails and inform deployment strategy.

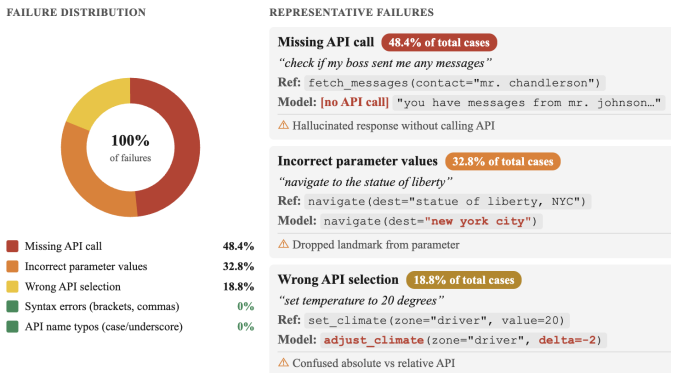


Fig. 5: Failure mode distribution and representative examples of S2A-1.2B on testset A (E2E, no teacher forcing).

JF decoding eliminates all API-related syntax and naming errors via FSM constrained generation. The remaining failures (missing calls, wrong parameters, wrong API choice) are semantic errors requiring reasoning beyond 1.2B scale, motivating hybrid edge-cloud system for low-confidence predictions.

IX. Learnings

Edge agentic models are feasible: Despite reduced general capabilities at small scale, edge agentic models are feasible when task scope is constrained and targeted optimizations are applied. Our S2A-1.2B approaches a 2.7x larger baseline on E2E task success (73.1% vs 77.3%) while getting 1.5s turn latency on CPU-only hardware. We find 1.2B (1B SLM + 200M encoder), the largest our hardware fits, meets our task-quality bar; smaller variants degrade significantly (Table III).

Joint training recovers errors that chaining propagates: S2A-1.2B improves turn-level API accuracy by 8.2% absolute over a same-size chained variant (85.1% vs 76.9%). In a pipeline, streaming ASR errors propagate irreversibly to the downstream model. Joint training avoids this by letting the SLM condition API predictions directly on speech context, not just noisy intermediate transcriptions.

Target task-specific optimizations: Generic INT8 quantization gives TPS gain, but leveraging structured nature of API enables further acceleration. Our SD draft model predicts API tokens more accurately (83% acceptance) than free-form responses (63%), and JF decoding eliminates generation entirely for predictable API substrings via FSM. Stacking all three optimizations yields 127% TPS over FP16 1.2B model with minimal accuracy loss.

Opt for streaming in latency-sensitive applications: Processing speech incrementally in chunks rather than waiting for the full utterance enabled us to achieve 1.5s turn latency. Without streaming, the model idles during user speech, adding full utterance duration to perceived latency.

X. Concluding Remarks

Edge models are crucial for applications constrained by data sharing, inference cost, and latency. In this work, we present EdgeTalk, a 1.2B unified streaming speech-text model that processes speech, calls APIs, and responds within 1.5s per turn on CPU-only hardware. A key finding is that joint speech-text-API training recovers 8.2% accuracy lost to cascaded ASR errors in chained pipelines. With quantization, speculative decoding, and jump-forward decoding, EdgeTalk approaches the task success of a 2.7x larger baseline while meeting real-time latency constraints that the larger system cannot. We intend this work as a practical blueprint for practitioners building constrained agentic systems on edge devices.

XI. Limitations & Future Work

Complexity: Due to limited reasoning capacity, EdgeTalk targets bounded, medium-complexity tasks. For complex queries exceeding on-device capability, we will fallback to a cloud-based model. Future work will explore confidence-based escalation to route only uncertain predictions to cloud.

Safety: Smaller models are more prone to hallucinations and biased outputs. Guardrails and preference alignment [46], [47] will be added before production deployment.

Speech generation: EdgeTalk produces text output only; a separate lightweight TTS renders responses to the user. Integrating speech generation within the model is future work.

Scope: The system is trained on only automotive APIs via SFT and does not generalize to unseen APIs at runtime. Scaling to new domains requires synthetic data regeneration and retraining. While our foundation models are multilingual, agentic evaluation is currently English-only; extending to other languages is a natural next step.

REFERENCES

[1] S. Chen, Y. Liu, W. Han, W. Zhang, and T. Liu, "A survey on llm-based multi-agent system: Recent advances and new frontiers in application," 2025. [Online]. Available: https://arxiv.org/abs/2412.17481

[2] ChatGPT, "OpenAI: Introducing ChatGPT," 2022. [Online]. Available: https://openai.com/blog/chatgpt

[3] Addepto, "LLM-Based assistance bot to enhance airport operations," 2025. [Online]. Available: https://addepto.com/case-studies/llm-based-assistance-bot-to-enhance-airport-operations/

[4] PerplexityAI, "Perplexity AI," 2022, AI tool. [Online]. Available: https://perplexity.ai/

[5] Rhasspy, "Rhasspy: Fully offline voice assistant," 2018. [Online]. Available: https://rhasspy.readthedocs.io/

[6] OpenVoiceOS, "Openvoiceos: Community-driven voice ai," 2024. [Online]. Available: https://www.openvoiceos.org/

[7] J. Zhang, T. Lan, M. Zhu, Z. Liu, T. Hoang, S. Kokane, W. Yao, J. Tan, A. Prabhakar, H. Chen, Z. Liu, Y. Feng, T. Awalgaonkar, R. Murthy, E. Hu, Z. Chen, R. Xu, J. Niebles, S. Heinecke, H. Wang, S. Savarese, and C. Xiong, "xlam: A family of large action models to empower ai agent systems," *arXiv preprint arXiv:2409.03215*, 2024. [Online]. Available: https://arxiv.org/abs/2409.03215

[8] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, L. Zettlemoyer, N. Cancedda, and T. Scialom, "Toolformer: Language models can teach themselves to use tools," 2023. [Online]. Available: https://arxiv.org/abs/2302.04761

[9] T. Miller, M. Shafique, and Y. Chen, "Voice assistants in the internet of things: Architecture, applications, and challenges," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9728–9742, 2020.

[10] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017, pp. 1273–1282.

[11] A. Plaat, M. van Duijn, N. van Stein, M. Preuss, P. van der Putten, and K. J. Batenburg, "Agentic large language models, a survey," 2025. [Online]. Available: https://arxiv.org/abs/2503.23037

[12] E. C. Acikgoz, J. Greer, A. Datta, Z. Yang, W. Zeng, O. Elachqar, E. Koukoumidis, D. Hakkani-Tür, and G. Tur, "Coalm: A unified conversational agentic language model," arXiv preprint arXiv:2502.08820, 2025.

[13] ABexit, "Asr-llm-tts: Modular speech interaction using sencevoice, qwen2.5 (0.5b/1.5b), and cosyvoice/edge-tts," GitHub repository, 2025, https://github.com/ABexit/ASR-LLM-TTS.

[14] Y. S. Khiabani, F. Atif, C. Hsu, S. Stahlmann, T. Michels, S. Kramer, B. Heidrich, M. S. Sarfraz, J. Merten, and F. Tafazzoli, "Optimizing small language models for in-vehicle function-calling," 2025. [Online]. Available: https://arxiv.org/abs/2501.02342

[15] Z. Liu, C. Zhao, F. Iandola, C. Lai, Y. Tian, I. Fedorov, Y. Xiong, E. Chang, Y. Shi, R. Krishnamoorthi, L. Lai, and V. Chandra, "Mobilellm: optimizing sub-billion parameter language models for on-device use cases," in *Proceedings of the 41st International Conference on Machine Learning*, ser. ICML'24. JMLR.org, 2024.

[16] X. Wang, Y. Li, C. Fu, L. Xie, K. Li, X. Sun, and L. Ma, "Freeze-omni: A smart and low latency speech-to-speech dialogue model with frozen llm," in *arXiv preprint arXiv:2411.00774*, 2024.

[17] S. Hu, L. Zhou, S. Liu, S. Chen, L. Meng, H. Hao, J. Pan, X. Liu, J. Li, S. Sivasankaran, L. Liu, and F. Wei, "Wavllm: Towards robust and adaptive speech large language model," 2024. [Online]. Available: https://arxiv.org/abs/2404.00656

[18] J. Ao, R. Wang, L. Zhou, C. Wang, S. Ren, Y. Wu, S. Liu, T. Ko, Q. Li, Y. Zhang, Z. Wei, Y. Qian, J. Li, and F. Wei, "Speecht5: Unified-modal encoder-decoder pre-training for spoken language processing," *ArXiv*, 2021.

[19] C. Tang, W. Yu, G. Sun, X. Chen, T. Tan, W. Li, L. Lu, Z. Ma, and C. Zhang, "Salmonn: Towards generic hearing abilities for large language models," 2024. [Online]. Available: https://arxiv.org/abs/2310.13289

[20] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," 2022. [Online]. Available: https://arxiv.org/abs/2212.04356

[21] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, "Gptq: Accurate post-training quantization for generative pre-trained transformers," 2023. [Online]. Available: https://arxiv.org/abs/2210.17323

[22] S. Kim, C. Hooper, A. Gholami, Z. Dong, X. Li, S. Shen, M. W. Mahoney, and K. Keutzer, "Squeezellm: Dense-and-sparse quantization," 2024. [Online]. Available: https://arxiv.org/abs/2306.07629

[23] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, and S. Han, "SmoothQuant: Accurate and efficient post-training quantization for large language models," in *Proceedings of the 40th International Conference on Machine Learning*, 2023.

[24] Y. Leviathan, M. Kalman, and Y. Matias, "Fast inference from transformers via speculative decoding," 2023. [Online]. Available: https://arxiv.org/abs/2211.17192

[25] B. T. Willard and R. Louf, "Efficient guided generation for large language models," 2023. [Online]. Available: https://arxiv.org/abs/2307.09702

[26] Y. Zhuang, X. Chen, T. Yu, S. Mitra, V. Bursztyn, R. A. Rossi, S. Sarkhel, and C. Zhang, "Toolchain*: Efficient action space navigation in large language models with a* search," 2023. [Online]. Available: https://arxiv.org/abs/2310.13227

[27] L. Alazraki and M. Rei, "Meta-reasoning improves tool use in large language models," 2025. [Online]. Available: https://arxiv.org/abs/2411.04535

[28] A. Gulati, J. Qin, C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, and R. Pang, "Conformer: Convolution-augmented transformer for speech recognition," in *21st Annual Conference of the International Speech Communication Association, Interspeech 2020, Virtual Event, Shanghai, China, October 25-29, 2020*, H. Meng, B. Xu, and T. F. Zheng, Eds. ISCA, 2020, pp. 5036–5040. [Online]. Available: https://doi.org/10.21437/Interspeech.2020-3015

[29] S. Black, S. Biderman, E. Hallahan, Q. Anthony, L. Gao, L. Golding, H. He, C. Leahy, K. McDonell, J. Phang, M. Pieler, U. S. Prashanth, S. Purohit, L. Reynolds, J. Tow, B. Wang, and S. Weinbach, "Gpt-neox-20b: An open-source autoregressive language model," 2022. [Online]. Available: https://arxiv.org/abs/2204.06745

[30] J. Ainslie, J. Lee-Thorp, M. de Jong, Y. Zemlyanskiy, F. Lebron, and S. Sanghai, "GQA: Training generalized multi-query transformer models from multi-head checkpoints," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, H. Bouamor, J. Pino, and K. Bali, Eds. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 4895–4901. [Online]. Available: https://aclanthology.org/2023.emnlp-main.298/

[31] T. Kudo and J. Richardson, "SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, E. Blanco and W. Lu, Eds. Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 66–71. [Online]. Available: https://aclanthology.org/D18-2012/

[32] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An asr corpus based on public domain audio books," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 5206–5210.

[33] V. Pratap, Q. Xu, A. Sriram, G. Synnaeve, and R. Collobert, "Mls: A large-scale multilingual dataset for speech research," in *Interspeech 2020*. ISCA, October 2020. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2020-2826

[34] R. Ardila, M. Branson, K. Davis, M. Kohler, J. Meyer, M. Henretty, R. Morais, L. Saunders, F. Tyers, and G. Weber, "Common voice: A massively-multilingual speech corpus," in *Proceedings of the Twelfth Language Resources and Evaluation Conference*, N. Calzolari, F. Béchet, P. Blache, K. Choukri, C. Cieri, T. Declerck, S. Goggi, H. Isahara, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, and S. Piperidis, Eds. Marseille, France: European Language Resources Association, May 2020, pp. 4218–4222. [Online]. Available: https://aclanthology.org/2020.lrec-1.520/

[35] C. Wang, M. Riviere, A. Lee, A. Wu, C. Talnikar, D. Haziza, M. Williamson, J. Pino, and E. Dupoux, "VoxPopuli: A large-scale multilingual speech corpus for representation learning, semi-supervised learning and interpretation," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, C. Zong, F. Xia, W. Li, and R. Navigli, Eds. Online: Association for Computational Linguistics, Aug. 2021, pp. 993–1003. [Online]. Available: https://aclanthology.org/2021.acl-long.80/

[36] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. de las Casas, E. B. Hanna, F. Bressand,

G. Lengyel, G. Bour, G. Lample, L. R. Lavaud, L. Saulnier, M.-A. Lachaux, P. Stock, S. Subramanian, S. Yang, S. Antoniak, T. L. Scao, T. Gervet, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed, "Mixtral of experts," 2024. [Online]. Available: https://arxiv.org/abs/2401.04088

[37] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[38] Anthropic, "Claude sonnet," https://www.anthropic.com/claude/sonnet, 2024.

[39] Python-AST, *Abstract Syntax Trees, Python Software Foundation*, Online documentation, 2025. [Online]. Available: https://docs.python.org/3/library/ast.html

[40] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: A method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, P. Isabelle, E. Charniak, and D. Lin, Eds. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, Jul. 2002, pp. 311–318. [Online]. Available: https://aclanthology.org/P02-1040/

[41] P. R. Male, S. N. Ray, H. Arsikere, A. Jaiswal, P. Swarup, P. Sen, D. Chakrabarty, K. V. V. Girish, N. Bhave, F. Weber, S. Bhattacharya, and S. Garimella, "Durep: Dual-mode speech representation learning via asr-aware distillation," 2025. [Online]. Available: https://arxiv.org/abs/2505.19774

[42] P. Soviany, R. T. Ionescu, P. Rota, and N. Sebe, "Curriculum learning: A survey," 2022. [Online]. Available: https://arxiv.org/abs/2101.10382

[43] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv *et al.*, "Qwen3 technical report," *arXiv preprint arXiv:2505.09388*, 2025.

[44] Lantronix, Inc., "SA8295P automotive development platform," https://www.lantronix.com/products/sa8295p-automotive-development-platform/, 2024.

[45] C. Chen, S. Borgeaud, G. Irving, J.-B. Lespiau, L. Sifre, and J. Jumper, "Accelerating large language model decoding with speculative sampling," 2023. [Online]. Available: https://arxiv.org/abs/2302.01318

[46] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe, "Training language models to follow instructions with human feedback," 2022. [Online]. Available: https://arxiv.org/abs/2203.02155

[47] R. Rafailov, A. Sharma, E. Mitchell, S. Ermon, C. D. Manning, and C. Finn, "Direct preference optimization: Your language model is secretly a reward model," 2024. [Online]. Available: https://arxiv.org/abs/2305.18290