A Cascaded Speaker Diarization and Transcription System for Japanese Conversational Speech

Bagus Tris Atmaja* and Sakriani Sakti*
* Nara Institute of Science and Technology, Ikoma, Japan
E-mail: bagus.tris@naist.ac.jp

***Abstract*—We present a cascaded system for joint speaker diarization and automatic speech recognition (ASR) targeting Japanese conversational speech. It integrates NVIDIA NeMo's SortFormer streaming diarizer with eight interchangeable Japanese ASR backends in a session-based pipeline that processes audio chunk-by-chunk. Diarization timestamps act as authoritative boundaries, and ASR output is aligned into speaker segments using word-level timestamps where available. Our primary contribution is a benchmark of these eight backends on the CALLHOME Japanese corpus (TalkBank) reporting accuracy and efficiency jointly: Diarization Error Rate (DER), Character Error Rate (CER), the speaker-attributed concatenated minimum-permutation CER (cpCER), real-time factor (RTF), loading time, and per-recording latency. We additionally fine-tune the diarizer on CALLHOME Japanese and Sakura. Fine-tuning gives a modest DER reduction from 7.35% to 7.02% (4.5% relative) at collar = 0.25 s but is neutral under strict no-collar scoring; the larger gains come from post-processing and from the ASR backend, which moves CER between 36.3% and 83.8%. In offline pipeline mode, seven of the eight backends reach RTF < 1 (faster-than-real-time throughput).**

I. Introduction

Speaker diarization determines "who spoke when" in a multi-speaker recording and underpins meeting transcription, conversational analytics, and spoken dialogue systems [1], [2]. Combined with automatic speech recognition (ASR), it yields a speaker-attributed transcript directly usable downstream.

Japanese conversational speech presents several challenges. Japanese is written without whitespace word boundaries and mixes three scripts (kanji, hiragana, katakana), so error rates are conventionally measured over characters rather than words. The most widely used conversational corpus [3] consists of telephone-channel recordings (8 kHz, narrowband) that lack the acoustic richness of the wideband speech on which current diarization and ASR models are pre-trained. Japanese conversation is also characterized by frequent backchannels (*aizuchi*) such as *un*, *aa*, and *hai* that are acoustically short but diarization-relevant. In addition, overlapping speech is common in informal Japanese dialogue, requiring robust diarization of concurrent speakers.

Recent end-to-end neural diarization systems such as EEND [4] and SortFormer [5] substantially outperform clustering-based pipelines, and the streaming variant of SortFormer [6] enables real-time inference over a sliding context window. On the ASR side, the NVIDIA Parakeet family, built on Fast Conformer [7] with Token-and-Duration Transducer (TDT) [8] decoding, achieves state-of-the-art character error rates on Japanese benchmarks.

In this paper we describe a cascaded diarization-and-transcription system that combines these two model families within a session-based pipeline designed for real-time deployment. The work is deliberately an integration and measurement study: rather than proposing a new architecture, it quantifies what is achievable on Japanese conversational speech with currently available components, and at what cost. Our main contributions are:

- A benchmark of eight Japanese ASR backends behind a common diarizer, reporting accuracy (CER, cpCER) and efficiency (loading time, RTF, latency) under an identical protocol, so the accuracy–efficiency trade-off is directly readable;
- An evaluation framework for speaker-attributed character-based ASR, including a cpCER whose handling of speaker-count mismatch is specified explicitly;
- A fine-tuning recipe for the SortFormer diarizer on CALLHOME and Sakura, with an ablation reporting how small and how collar-dependent the resulting gain is;
- A modular streaming pipeline whose alignment stage attributes words to speaker segments without requiring forced alignment.

II. System Description

A. Overview

Fig. 1 shows the overall architecture. Audio is received as a continuous PCM stream at 16 kHz mono, segmented into fixed-length chunks (default: 2 s) and processed in order. Each chunk is appended to a per-session rolling buffer retaining up to 600 s of audio, and three sequential stages are applied to every chunk: (1) streaming speaker diarization, (2) per-segment ASR, and (3) alignment of ASR output into speaker-labelled segments.

All models used here expect 16 kHz input, whereas CALLHOME Japanese is 8 kHz narrowband telephone audio; such input is upsampled with a polyphase resampler before entering the buffer. Upsampling restores the sampling rate but not the missing 4–8 kHz band, so the models see a spectrum unlike their wideband pre-training data. This mismatch plausibly contributes to the absolute error rates of Section VI, and it applies equally to every backend.

Part of this work was supported by JSPS KAKENHI JP24K0296

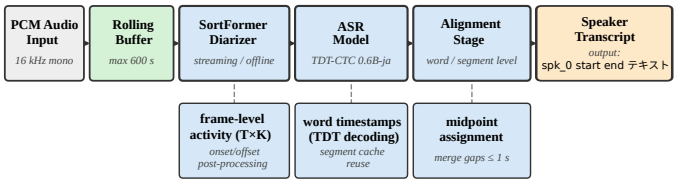


Fig. 1. Cascaded diarization-and-transcription pipeline: audio chunks pass through streaming diarization, per-segment ASR, and alignment to produce a speaker-attributed transcript.

B. Streaming Speaker Diarization

Diarization is performed by open weight speaker diarization model[1], which is a pre-trained NeMo SortFormer model supporting up to four simultaneous speakers [5], [6]. The model operates in two modes depending on the call context:
Streaming (chunk) mode: Each chunk, with the session buffer as context, is processed via `forward_streaming`, SortFormer's incremental inference path, so speaker identities stay consistent across chunks. Raw output is a $(T \times K)$ matrix of frame-level speaker activity probabilities, where T is the number of prediction frames and $K = 4$ the maximum speaker count; a threshold of 0.5 binarises it, and contiguous active regions become diarization segments with absolute timestamps.
Final-pass (offline) mode: At the end of a session the complete buffer is diarized offline via NeMo's `diarize()` API, running SortFormer sliding-window over the full recording to produce the globally consistent hypothesis used for evaluation. Post-processing parameters (onset/offset thresholds, minimum segment duration) control the false-alarm rate.

Consecutive same-speaker segments separated by less than 0.5 s are merged *before* being passed to ASR, reducing fragmentation and giving the ASR model more contextual audio. This pre-ASR merge is distinct from the 1 s post-ASR merge described in Section II-D, which only affects the readability of the final transcript.

C. Japanese ASR

Transcription is performed by eight interchangeable ASR backends. The default is the Parakeet TDT-CTC Japanese model[2] [7], [8], a 0.6-billion-parameter Fast Conformer with TDT-CTC decoding that emits character sequences directly in Japanese script without a language model, making inference fast and deterministic.

The remaining seven backends span a range of architectures and training regimes. **ReazonSpeech-NeMo** [9] is a NeMo Conformer-CTC model trained on the large-scale ReazonSpeech corpus of Japanese broadcast audio. **Whisper** [10] is OpenAI's encoder–decoder transformer (`whisper-large-v3`), trained on 680K hours of weakly supervised multilingual audio; we set the language to Japanese and request word-level timestamps. **Kotoba-Whisper** (`distil-whisper-ja`) [11] is a knowledge-distilled Japanese-specialised Whisper with a much shallower

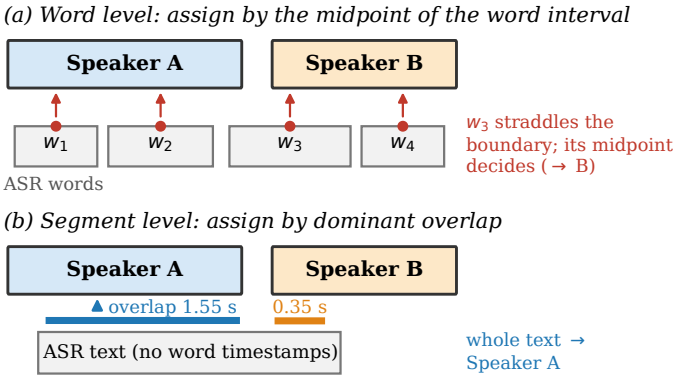


Fig. 2. Diarization–ASR alignment. (a) With word timestamps, each word is assigned to the segment containing the midpoint of its start–end interval; w_3 straddles the boundary and follows its midpoint. (b) Without word timestamps, the whole ASR text of a window is assigned to the segment with the largest overlap.

decoder. **Qwen3-ASR** [12] is Alibaba's 1.7B-parameter autoregressive multilingual ASR model. **IBM Granite 4.0 1B Speech**[3] is a compact multilingual model for constrained hardware; it supports but is not specialized for Japanese, a useful contrast with the Japanese-focused models. **Kushinada-HuBERT** [13], [14] is an ESPnet2 model applying a large pre-trained HuBERT encoder fine-tuned on LABOROTV2 Japanese broadcast speech. **Cohere Transcribe**[4] is Cohere-Labs' multilingual transcription model, based on an undisclosed encoder–decoder architecture.

For each diarized segment an audio slice is extracted from the buffer and passed to the ASR model, with segments batched for GPU efficiency; segments whose boundaries are unchanged since the previous chunk are cached by speaker label and time range and their transcripts reused. A *whole-audio* mode also computes $\mathrm{CER_w}$, transcribing the full recording in fixed 60 s chunks without diarization. This matches the official Parakeet benchmark conditions and isolates recognition from diarization quality.

D. Diarization–ASR Alignment

Diarization timestamps are treated as authoritative: they define the speaker boundaries of the output transcript. The alignment stage maps transcribed text onto the diarized segments using the two-level strategy illustrated in Fig. 2.

Word-level alignment. When the ASR model emits per-word timestamps (available from TDT decoding), every recognized word w carries a start–end interval $[t_w^{\mathrm{s}}, t_w^{\mathrm{e}}]$. We compute the midpoint of that interval, $m_w = (t_w^{\mathrm{s}} + t_w^{\mathrm{e}})/2$, and assign w to the diarized segment whose span contains m_w (Fig. 2a); a word straddling a segment boundary is thus attributed to whichever speaker holds the larger share of its duration. If m_w falls in a gap between segments, w goes to the segment whose own midpoint is closest to m_w.

Segment-level alignment. When per-word timestamps are not available (e.g. CTC decoding without forced alignment),

[1]https://huggingface.co/nvidia/diar_streaming_sortformer_4spk-v2.1
[2]https://huggingface.co/nvidia/parakeet-tdt_ctc-0.6b-ja
[3]https://huggingface.co/ibm-granite/granite-4.0-1b-speech
[4]https://huggingface.co/CohereLabs/cohere-transcribe-03-2026

the full ASR text for a given audio slice is assigned to the diarized segment with the greatest time overlap with that slice (dominant-speaker assignment, Fig. 2b). If no overlap exists, the segment containing the midpoint of the ASR window is used as a fallback.

We chose midpoint containment over forced alignment or joint decoding because it is causal and $O(1)$ per word, so it adds no latency and works unchanged in streaming mode; it needs no acoustic model beyond the ASR decoder, keeping backends that expose only segment-level timings interchangeable; and the midpoint is robust to the boundary jitter typical of TDT and CTC timestamps, since an error below half the word duration leaves the assignment unchanged. The cost is that alignment cannot repair a diarization error: missed or confused turns propagate into the transcript, which is what cpCER measures.

The aligned segments are sorted by start time and returned as the pipeline output. A post-processing step optionally merges consecutive same-speaker segments separated by a configurable gap (default: 1 s) for readability; this merge runs after transcription and does not alter the audio sent to the ASR model.

III. Datasets

We use two Japanese conversational corpora from the TalkBank repository [15]: CALLHOME Japanese[5] and Sakura[6]. They play different roles: CALLHOME is used for both fine-tuning and evaluation, while Sakura serves *only* as additional fine-tuning material and is never scored, for the reason given below. CALLHOME Japanese [3] consists of 120 telephone conversations between native Japanese speakers in the United States. It is predominantly, but not exclusively, two-party: 105 of the 120 recordings have two annotated speakers, 10 have three, 2 have four, and 3 have five. All evaluation uses a fixed 12-file HF test split (80/10/10 partition, seed=0) for reproducibility and direct comparison with prior work. Sakura [16] is an 18-recording corpus of video-recorded, face-to-face group discussions among university students. It is *not* dyadic and not telephone speech: every recording has four or more annotated speakers and 17 of the 18 exceed the four-speaker capacity of the diarization model (Section II-B), so scoring it would largely measure that ceiling rather than diarization quality. We therefore restrict Sakura to the fine-tuning set, where it adds acoustic and conversational variety absent from telephone speech. Both corpora are in CHAT format [15] with orthographic transcripts and time-aligned speaker labels. Table I summarizes their characteristics.

Effect of the four-speaker cap. Our SortFormer checkpoint supports at most $K = 4$ concurrent speakers, so in a recording with more reference speakers at least one cannot receive a distinct output track. This places a floor on the attainable error: in DER that speaker's speech is scored as missed detection or speaker confusion, and in cpCER the permutation search must match at least one reference string against an empty hypothesis, a full deletion of that speaker's characters. In CALLHOME the cap affects 3 of 120 recordings (2.5%) and is thus a marginal factor for the results reported here; in Sakura it affects 17 of 18, which is why that corpus is excluded from scoring. No recording is dropped and no reference speaker removed at evaluation time.

TABLE I
Dataset characteristics: CALLHOME Japanese and Sakura. All audio is resampled to 16 kHz before processing.

Property	CALLHOME	Sakura
Corpus size (total)	120 recordings	18 recordings
Role in this work	fine-tune + test	fine-tune only
# Test samples	12 (HF test split)	–
Recording condition	telephone, 8 kHz	face-to-face
Speakers per recording	2–5 (105 are 2)	4–6 (none are 2)
Recordings with >4 spk	3 (2.5%)	17 (94%)
Mean duration	557 s (9.3 min)	1528 s (25.5 min)

IV. Domain Adaptation by Fine-Tuning

A. Data Preparation

We fine-tune the SortFormer diarizer on the two corpora of Section III: CALLHOME Japanese [3] and Sakura [16], the latter placed entirely in the training set. Each recording is converted to NeMo RTTM format and, for training only, recordings with more than four speakers are reduced to the four with the longest total speech, matching the model's output capacity; evaluation applies no such reduction. The 120 CALLHOME recordings are randomly split into training (90%), validation (5%) and test (5%) subsets with a fixed seed, and NeMo JSON manifests are generated for each. All results below are reported on the independent 12-recording *HF test split* derived from the HuggingFace `talkbank/callhome` dataset with an 80/10/10 partition (seed 0).

B. Fine-Tuning Configuration

We initialise from the pre-trained `Streaming Sortformer Diarizer 4spk v2.1` checkpoint and fine-tune with the NeMo toolkit [17], which builds on PyTorch Lightning. Only the SortFormer attention head is fine-tuned; the Fast Conformer encoder and mel-spectrogram preprocessor are frozen to preserve the acoustic representations learned during large-scale pre-training and to reduce the risk of overfitting on the small Japanese corpus.

Training uses AdamW with cosine annealing (initial learning rate 10^{-5}, minimum 10^{-6}, 100 warm-up steps), batch size 4, and a session length of 90 s, for at most 20 epochs on a single NVIDIA RTX 3090 GPU; the checkpoint with the lowest validation loss is retained (epoch 14, validation loss 0.1319). The model is saved as a `.nemo` archive and loaded directly into the pipeline, making fine-tuned and base models interchangeable without code changes.

V. Evaluation Protocol

A. Diarization Error Rate

DER is computed using the standard formulation [18]:

$$\mathrm{DER} = \frac{E_{\mathrm{MS}} + E_{\mathrm{FA}} + E_{\mathrm{SC}}}{T_{\mathrm{ref}}}, \tag{1}$$

[5]https://talkbank.org/ca/access/CallHome/jpn.html
[6]https://talkbank.org/ca/access/Sakura.html

where E_{MS}, E_{FA}, and E_{SC} are the durations of missed speech, false alarm, and speaker confusion errors, respectively, and T_{ref} is the total reference speech duration. The collar is a tolerance window around reference boundaries within which errors are not penalized, accounting for annotation uncertainty. We report both the standard 0.25 s collar and the strict no-collar condition, and treat inclusion of overlapping reference regions as an explicit ablation factor rather than a fixed convention; unless stated otherwise, overlap is *included*. Scoring uses `pyannote.metrics` [19].

B. Character Error Rate Metrics

Character Error Rate (CER): CER is the primary ASR metric for Japanese, as the language has no whitespace word boundaries in native script. We compute CER as the Levenshtein distance between the reference and hypothesis character sequences (spaces removed):

$$\text{CER} = \frac{S + D + I}{N}, \tag{2}$$

where S, D, I are substitutions, deletions, and insertions, and N is the number of reference characters. Reference transcripts come from the CHAT-format `.cha` files distributed with the TalkBank corpora [15], with CHAT annotations (timestamps, retrace markers, non-verbal events, unintelligible tokens) stripped before scoring. Each recording is matched to its `.cha` file by aligning the HuggingFace audio duration with the transcript duration derived from the embedded CHAT timestamps.

Concatenated Minimum-Permutation CER (cpCER): The CER above compares one time-ordered concatenation of the hypothesis against one time-ordered concatenation of the reference, and so is blind to *which* speaker produced a character: a hypothesis that transcribes every word correctly but attributes it to the wrong speaker attains the same CER as a fully correct output. To make attribution visible we also report cpCER, the character-level analogue of the concatenated minimum-permutation WER used in multi-speaker benchmarks [20].

Given N reference speakers, reference text is concatenated per speaker in order of first appearance, forming strings $R_1, \ldots, R_N$. Hypothesis text is grouped by diarization label to form $H_1, \ldots, H_M$, where M is the number of distinct speaker labels the diarizer emitted for that recording (M need not equal N). The metric minimises over injective maps $\pi : \{1, \ldots, N\} \to \{1, \ldots, \max(M, N)\}$, where the hypothesis side is padded with empty strings when $M < N$ (see below) so that such a map always exists; that is, $\pi(i)$ is the single hypothesis index assigned to reference speaker i and no hypothesis speaker is used twice:

$$\text{cpCER} = \min_{\pi} \frac{\sum_{i=1}^{N} \text{edit}(R_i,\, H_{\pi(i)})}{\sum_{i=1}^{N} |R_i|}, \tag{3}$$

where $\text{edit}(\cdot, \cdot)$ is the character-level Levenshtein distance and $|R_i|$ is the number of characters in reference speaker i. Speaker-count mismatch is handled as follows. If $M < N$, the $N - M$ unmatched reference speakers are paired with the empty string, so all their characters count as deletions; this penalises a diarizer for merging speakers and is also the floor imposed on recordings with more than four speakers (Section III). If $M > N$, the $M - N$ surplus hypothesis clusters are left unmatched and their characters discarded rather than charged as insertions. This deviates from the standard cpWER formulation, which pads the reference side with empty strings and so penalises over-segmentation; our variant is therefore optimistic about speaker over-splitting. Since all backends share one diarizer and hence one M per recording, the comparison in Table III is unaffected.

VI. Results and Discussion

A. Benchmark

As a reference point, Table II also lists the publicly available fine-tuned pyannote segmentation model trained on the same Japanese CALLHOME corpus with the `diarizers` library [21], which reports DER 23.11% against 15.19% for our base model and 15.23% for our fine-tuned model under no-collar scoring.

We deliberately refrain from claiming a like-for-like improvement over this baseline, because its published figure is not directly comparable to ours in three respects. It is computed with the frame-level DER of `pyannote.audio.torchmetrics`, which exposes a detection threshold but no collar parameter, whereas our numbers come from `pyannote.metrics` with an explicit collar; it is measured on the validation split rather than the 12-file test split used throughout this paper; and it reflects the released checkpoint's default inference and post-processing settings, whereas our post-processing thresholds were tuned on our own model. The two systems also differ architecturally: SortFormer emits up to four sorted speaker tracks end-to-end, while the pyannote model performs local segmentation and relies on a downstream clustering stage. A strictly comparable number would require re-scoring the baseline under our protocol, which we leave to future work.

B. Diarization Ablation

Table II shows the effect of Japanese fine-tuning, optional post-processing (PP), and skipping overlapping reference regions (Skip) on diarization quality, under both standard (collar = 0.25 s) and strict (no collar) evaluation conditions.

The effect of fine-tuning is small and collar-dependent, and we state it as such. At collar = 0.25 s without post-processing, fine-tuning lowers DER from 7.35% to 7.02%: 0.33 points, or 4.5% relative. It redistributes error more than it removes it — missed detection falls (5.17% → 3.47%) and speaker confusion falls slightly (0.82% → 0.63%), while false alarm more than doubles (1.36% → 2.93%). Under strict no-collar scoring the gain disappears (15.19% base vs. 15.23% fine-tuned, i.e. neutral to marginally worse). The consistent part of the improvement is therefore boundary-insensitive speaker assignment, not sharper boundary placement, and on a 12-file test set we do not claim the 0.33-point difference is statistically meaningful.

The other two factors matter more. Skipping overlap has the largest single impact, removing the hardest regions from scoring and dropping DER from 7.35% to 4.93% for the base model (collar = 0.25 s) — a reminder that DER is only comparable across papers when the overlap convention is stated. Applying PP to the base model *increases* DER (7.35% → 8.47%) because the PP thresholds were optimized on the fine-tuned model outputs and do not generalize to the base model. For the fine-tuned model, PP lowers false alarm (2.93% → 1.42%) at the cost of increased missed detections, leaving DER roughly unchanged. The best overall DER (4.83%) is achieved by the fine-tuned model with both PP and overlap skipping. For the full-overlap evaluation (Skip = −), fine-tuned without PP is the best configuration (DER = 7.02%) and is therefore used as the pipeline default in the ASR experiments that follow.

TABLE II
DIARIZATION BENCHMARK AND ABLATION ON THE CALLHOME JAPANESE HF TEST SPLIT (IN %). PP = POST-PROCESSING (SEGMENT MERGING); SKIP = OVERLAPPING REFERENCE REGIONS EXCLUDED FROM SCORING; C = COLLAR SIZE; DER/FA/MD/SC = DIARIZATION ERROR, FALSE ALARM, MISSED DETECTION, SPEAKER CONFUSION.

C	Model	PP	Skip	DER	FA	MD	SC
0.00 s	pyannote-ft-jpn[a]	–	–	23.11	4.77	13.52	4.82
	Base	–	–	15.19	4.12	9.60	1.47
		–	✓	**12.18**	5.79	4.80	1.59
		✓	–	16.39	2.84	12.22	1.33
		✓	✓	12.24	4.00	6.81	1.42
	Fine-tuned	–	–	15.23	7.73	6.27	1.23
		–	✓	15.46	10.95	2.95	1.56
		✓	–	15.37	4.58	9.52	1.27
		✓	✓	12.48	6.42	4.43	1.63
0.25 s	Base	–	–	7.35	1.36	5.17	0.82
		–	✓	4.93	1.73	2.49	0.71
		✓	–	8.47	0.91	6.76	0.80
		✓	✓	5.51	1.15	3.65	0.70
	Fine-tuned	–	–	7.02	2.93	3.47	0.63
		–	✓	6.12	3.78	1.63	0.71
		✓	–	7.24	1.42	5.14	0.68
		✓	✓	**4.83**	1.82	2.23	0.78

[a] https://huggingface.co/diarizers-community/speaker-segmentation-fine-tuned-callhome-jpn

C. ASR Ablation

Table III benchmarks eight ASR backends on the CALLHOME Japanese HF test split using the fine-tuned SortFormer diarizer in offline pipeline mode, reporting accuracy and efficiency side by side. `kushinada-hubert` achieves the best CER (36.25%) and cpCER (34.41%), showing the advantage of a model trained directly on Japanese broadcast speech. Compact models such as `parakeet` (39.00%) and `distil-whisper-ja` (39.74%) stay competitive, within 2.8 and 3.5 CER points, while requiring less memory. `granite` (67.76%) and `qwen3-asr` (83.83%) are not usable for Japanese conversational speech without further adaptation.

Why cpCER falls below CER. For most backends cpCER is slightly lower than CER (34.41% vs. 36.25% for `kushinada-hubert`), seemingly contradicting the claim that cpCER *adds* an attribution penalty. The two are not nested, because they concatenate text differently. CER compares one time-ordered concatenation of the recording against another, so a turn placed early or late changes the interleaving and charges the same characters twice, as a deletion and an insertion. cpCER concatenates per speaker into N blocks, removing cross-speaker interleaving, and its minimum-permutation search removes the arbitrary label ordering; both relaxations lower the score and here outweigh the added penalty. Precisely, cpCER penalizes attribution errors that a speaker-agnostic comparison *over the same per-speaker grouping* would not, but is not an upper bound on CER.

Why the absolute error rates are high. A best CER of 36% is far above the single-digit rates reported for Japanese read or broadcast speech, because several properties of this task compound: narrowband telephone audio upsampled to 16 kHz (Section II), spontaneous speech whose disfluencies and self-repairs the CHAT references transcribe in detail, frequent backchannels of which our filter catches only canonical forms, scored rather than excluded overlap, and residual orthographic variation between kanji, hiragana and katakana. These numbers are therefore relative rankings under a common protocol, not absolute transcription quality.

Whole-audio versus pipeline CER. $\mathrm{CER}_{\mathrm{w}}$ and CER diverge in both directions and by large margins: `kushinada-hubert` is far worse without diarization (77.25% vs. 36.25%) while `whisper` is far better (39.13% vs. 58.56%). The conditions differ in input length, and the families respond oppositely: encoder-only CTC models trained on short single-speaker utterances degrade on the 60 s whole-audio chunks, whereas autoregressive encoder–decoder models exploit long context and are known to hallucinate and loop on the short, sometimes sub-second segments the pipeline produces. We did not run the segment-length sweep that would isolate this, so we offer it as the likeliest explanation rather than a demonstrated one. Either way, the best backend depends on the operating mode, so choosing an ASR model on whole-audio numbers can mislead when the deployed system is a diarize-then-transcribe pipeline.

Inspection of the normalized transcripts shows two distinct failure modes behind these numbers. The stronger backends recover the content words and err by substitution (e.g. らいとがくる for らいうとかがくるの), and `granite` likewise produces fluent but inaccurate Japanese, which is how a seemingly plausible output reaches CER 68%. `qwen3-asr` instead collapses outright, emitting repetitive affirmative tokens (はいはいはい… うん…) with no propositional content.

D. RTF and Latency Comparison

Real Time Factor (RTF) is the ratio of processing time to audio duration. Timings are measured on a single NVIDIA RTX 3090 GPU with wall-clock `perf_counter` in offline pipeline mode (diarize once → batch ASR); model load time is measured separately and excluded from RTF. Diarization RTF is nearly identical across backends (≈ 0.004) because all runs share the same fine-tuned SortFormer model, so ASR dominates total processing time. Seven of the eight backends run faster than real time ($\mathrm{RTF}_{\mathrm{t}} < 1.0$), with `parakeet` the fastest (0.0118, latency 6.7 s), followed by `cohere` (0.0283, 16.7 s) and `reazonspeech-nemo` (0.0464, 26.0 s).

TABLE III

ACCURACY AND EFFICIENCY OF EIGHT ASR BACKENDS ON THE CALLHOME JAPANESE HF TEST SPLIT (12 FILES, OFFLINE PIPELINE MODE, FINE-TUNED DIARIZER, COLLAR = 0.25 S, BACKCHANNELS SKIPPED, CHARACTERS NORMALIZED). CER = PIPELINE CER; CER_W = WHOLE-AUDIO CER (NO DIARIZATION, 60 S CHUNKS); CPCER = CONCATENATED MINIMUM-PERMUTATION CER; LOAD = ONE-TIME DIARIZER + ASR LOADING TIME (S); $\text{RTF}_\text{D}/\text{RTF}_\text{A}/\text{RTF}_\text{T}$ = REAL-TIME FACTOR OF DIARIZATION/ASR/TOTAL; $\overline{\text{LAT}}$ = MEAN WALL-CLOCK TIME PER RECORDING (S).

		Accuracy			Efficiency				
Backend	**Model**	**CER_w %**	**CER %**	**cpCER %**	**Load**	**RTF_d**	**RTF_a**	**RTF_t**	**$\overline{\text{Lat}}$**
kushinada-hubert	imprt/kushinada-hubert-large-laborotv2-asr	77.25	**36.25**	**34.41**	11.0	**0.0037**	0.2442	0.2480	138.2
parakeet	nvidia/parakeet-tdt_ctc-0.6b-ja	43.93	39.00	37.37	16.0	0.0039	**0.0079**	**0.0118**	**6.7**
distil-whisper-ja	kotoba-tech/kotoba-whisper-v2.0	56.86	39.74	38.57	**8.6**	**0.0037**	0.0644	0.0681	37.8
cohere	CohereLabs/cohere-transcribe-03-2026	40.85	41.11	40.20	13.7	0.0040	0.0243	0.0283	16.7
reazonspeech-nemo	reazon-research/reazonspeech-nemo-v2	56.25	44.31	45.11	13.9	0.0042	0.0423	0.0464	26.0
whisper	openai/whisper-large-v3	**39.13**	58.56	57.47	**8.6**	**0.0037**	0.8647	0.8684	477.6
granite	ibm-granite/granite-4.0-1b-speech	75.73	67.76	67.49	19.2	0.0038	0.0537	0.0575	34.0
qwen3-asr	Qwen/Qwen3-ASR-1.7B	94.52	83.83	84.03	9.7	**0.0037**	2.7834	2.7872	1550.9

`kushinada-hubert`, the CER leader, incurs substantial latency (138.2 s, $\text{RTF}_\text{a} = 0.244$) from computationally intensive ESPnet2 inference, and only `qwen3-asr` ($\text{RTF}_\text{t} = 2.7872$) cannot keep up with a live stream at all.

RTF and latency are not interchangeable, and `whisper` illustrates why. Its RTF_t of 0.8684 is below 1.0, so it consumes audio faster than it arrives and would not fall progressively behind a live stream. But these recordings average 557 s and the offline transcript appears only after the whole recording is processed, giving a mean latency of 477.6 s: an RTF just under 1.0 means the listener waits nearly the duration of the conversation. Interactive use needs RTF well below 1.0 — a margin, not merely a value under one. `parakeet`, `cohere` and `reazonspeech-nemo` have two orders of magnitude of headroom; `whisper` has essentially none. All timings are for offline pipeline mode; streaming mode processes 2 s chunks, trading throughput for incremental output.

VII. CONCLUSIONS

We built a Japanese diarization and transcription pipeline and used it to benchmark eight ASR backends behind a common fine-tuned SortFormer diarizer, measuring accuracy and efficiency under one protocol. The backend choice dominates every other factor we varied: CER ranges from 36.25% to 83.83% and latency spans more than two orders of magnitude (6.7 s to 1550.9 s). The most accurate backend is not the most deployable: `kushinada-hubert` leads on CER and cpCER but is 20× slower than `parakeet`, which trails it by 2.8 CER points. The best backend also depends on the operating mode, since whole-audio and diarized-segment CER rank the models differently. Fine-tuning the diarizer gave a smaller and less robust benefit than the pipeline configuration did: 7.35% → 7.02% DER at collar = 0.25 s, and no gain under strict no-collar scoring. Absolute error rates remain high on this narrowband, spontaneous material, so these numbers are best read comparatively. Future work should address the four-speaker capacity limit, which prevents Sakura from being scored at all, and evaluate alignment strategies beyond the midpoint rule.

REFERENCES

[1] X. Anguera *et al.*, "Speaker diarization: A review of recent research," *IEEE Trans. Audio, Speech, Language Process.*, vol. 20, no. 2, pp. 356–370, 2012.

[2] T. J. Park *et al.*, "A review of speaker diarization: Recent advances with deep learning," *Comput. Speech Lang.*, vol. 72, p. 101 317, 2022.

[3] A. Canavan *et al.*, *CALLHOME japanese speech*, LDC Catalog No. LDC96S37, 1997.

[4] H. Bredin *et al.*, "End-to-end speaker segmentation for overlap-aware resegmentation," in *Proc. Interspeech*, 2021, pp. 3111–3115.

[5] T. Shim *et al.*, "SortFormer: Seamless integration of speaker diarization and ASR by bridging timestamps and tokens," *arXiv preprint arXiv:2409.06656*, 2024.

[6] N. R. Koluguri *et al.*, "Streaming speaker diarization with SortFormer," in *Proc. Interspeech*, 2024.

[7] D. Rekesh *et al.*, "Fast Conformer with linearly scalable attention for efficient speech recognition," *Proc. ASRU*, 2023.

[8] H. Xu *et al.*, "Efficient sequence transduction by jointly predicting tokens and durations," *Proc. Mach. Learn. Res.*, pp. 38 462–38 484, 2023.

[9] Y. Aoki *et al.*, "Towards building a large-scale, high-quality Japanese ASR model," in *Proc. Interspeech*, 2024.

[10] A. Radford *et al.*, "Robust speech recognition via large-scale weak supervision," in *Proc. ICML*, 2023, pp. 28 492–28 518.

[11] S. Gandhi *et al.*, *Distil-Whisper: Robust knowledge distillation via large-scale pseudo labelling*, arXiv preprint 2311.00430, 2023.

[12] X. Shi *et al.*, "Qwen3-ASR Technical Report," pp. 1–16, 2026. arXiv: 2601.21337. [Online]. Available: http://arxiv.org/abs/2601.21337.

[13] W.-N. Hsu *et al.*, "HuBERT: Self-supervised speech representation learning by masked prediction of hidden units," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 29, pp. 3451–3460, 2021.

[14] S. Watanabe *et al.*, "ESPnet: End-to-end speech processing toolkit," in *Proc. Interspeech*, 2018, pp. 2207–2211.

[15] B. MacWhinney, *The CHILDES Project: Tools for Analyzing Talk*, 3rd. Mahwah, NJ: Lawrence Erlbaum Associates, 2000.

[16] S. Miyata *et al.*, *CABank Japanese Sakura Corpus*, 2009. DOI: 10.21415/T5M90R.

[17] O. Kuchaiev *et al.*, "NeMo: A toolkit for building AI applications using neural modules," in *Proc. SysML*, 2019.

[18] J. Fiscus *et al.*, "Multiple dimension Levenshtein edit distance calculations for hypothesis comparison," in *Proc. LREC*, 2006.

[19] H. Bredin, "Pyannote.audio 2.1 speaker diarization pipeline: Principle, benchmark, and recipe," in *Proc. Interspeech*, 2023, pp. 1983–1987.

[20] S. Watanabe *et al.*, "CHiME-6 challenge: Tackling multi-speaker speech recognition for unsegmented recordings," in *Proc. CHiME-6 Workshop*, 2020.

[21] K. Akesbi *et al.*, *Diarizers: A repository for fine-tuning speaker diarization models*, https://github.com/huggingface/diarizers, 2024.