Sauti NeMo Technical Report

Offline and Streaming Transducers for Swahili and English Speech Recognition

Gilbert Korir Kevin Obote Moses Njau Cyprian Kiplangat
Irene Korir Caroline Heta Benard Otieno

MsingiAI, Nairobi, Kenya
research@msingiai.com

6 October 2026

Abstract

We report the adaptation and evaluation of two 0.6B-parameter speech recognizers for Swahili and English: a FastConformer token-and-duration transducer (TDT) for file transcription and a cache-aware FastConformer RNN transducer for streaming. Both are fine-tuned for 15,000 updates on 586,510 utterances totaling 1,518 hours. On 18,265 standard test clips and a speaker-disjoint holdout of 3,005 code-switched clips, the averaged TDT checkpoint obtains 13.83% and 10.31% official word error rate (WER). Relative to the streaming checkpoint with context [56,13], the paired WER differences are -3.80 percentage points (95% utterance-bootstrap interval $[-3.98, -3.62]$) and -1.42 points $[-1.57, -1.26]$. These intervals condition on the evaluated corpora; the standard split is not speaker-disjoint. For streaming inference, we implement per-stream prompt conditioning and state isolation within dynamic batches, and measure fixed-rate audio arrivals rather than throughput alone. At 32 scheduled streams on one A100-SXM-64GB, the 80 ms configuration records 62.8 ms 95th-percentile (p95) and 81.5 ms 99th-percentile (p99) chunk response over 132.3 seconds of full load, using 32-bit floating-point (FP32) precision. Of 52,681 eligible chunks, 680 exceed 80 ms. All 840 call replays across six load levels preserve the corresponding single-stream final transcript. The study identifies English words within code-switched speech, tail stalls and unmeasured endpointing costs as the main constraints on further deployment.

Keywords: speech recognition, Swahili, code-switching, transducer, streaming ASR, FastConformer, RNN-T, TDT, bilingual, latency

1 Introduction

A bilingual recognizer must preserve English lexical content while adapting its acoustic and language representations to Swahili. Code-switched speech makes this requirement measurable: a corpus-level score can improve while a small English component remains poorly recognized. A live recognizer adds a second constraint. It must process new audio without access to future frames and retain sufficient state to continue an utterance as other requests enter and leave a batch.

We study these constraints with two pretrained NVIDIA transducer checkpoints. Sauti NeMo TDT uses Parakeet TDT 0.6B v3 [1]; Sauti NeMo Stream uses Nemotron 3.5 ASR Streaming 0.6B [2]. Both models are fully fine-tuned on the same accepted corpus mixture. The released artifacts average the three checkpoints with the lowest validation WER. The architecture, transducer objectives and pretrained representations are inherited; the work reported here concerns adaptation, evaluation and streaming-system implementation.

The empirical results are organized around three questions. First, how do the two adapted models compare on identical file-transcription examples? Second, how does changing the

streaming attention context affect accuracy and computation for one fixed checkpoint? Third, what response-time distribution is observed when independently scheduled audio streams share a GPU? We answer the first with complete test sets and paired resampling, the second with a four-context streaming study, and the third with an open-loop replay benchmark.

The contributions are (i) an audited 1,518-hour training recipe and two reproducible fine-tuned checkpoints; (ii) aligned bilingual and code-switch evaluations with explicit normalization, source composition and uncertainty; and (iii) a cache-aware batching implementation whose final transcripts are checked against single-stream inference and whose latency includes queueing. The current evidence does not isolate the effect of corpus weighting, tokenizer retention or checkpoint averaging, since those choices were not separately ablated.

2 Model Architecture

2.1 Shared Acoustic Encoder

Both saved configurations use a 24-layer FastConformer encoder with model dimension 1,024, eight attention heads, feed-forward expansion factor four and convolution kernel size nine. The acoustic front end produces 128 log-mel features from 16 kHz mono audio with a 25 ms Hann window, 10 ms frame shift and 512-point FFT. Depthwise striding reduces the frame rate by a factor of eight. Both encoders belong to the FastConformer family described by Rekesh et al. [3].

Table 1: Architectural differences from the saved fine-tuned configurations. Inactive beam-search configuration entries are omitted; evaluated decoding is greedy.

Component	TDT	Stream
Encoder attention	Full context in file evaluation	Chunk-limited; contexts [56,r]
Subsampling / conv.	Noncausal / batch norm	Causal / layer norm
Acoustic normalization	Per feature	None across the utterance
Vocabulary size	8,192	13,087
Prediction network	2 recurrent layers, width 640	2 recurrent layers, width 640
Joint network width	640	640
Output objective	TDT; durations {0,1,2,3,4}	RNN-T; FastEmit coefficient 0.005
Language conditioning	None	128-dim one-hot prompt
Greedy symbol limit	10 per decoding step	10 per decoding step

2.2 Transducer Objectives and Language Conditioning

Let h_t denote an encoder state and g_u a prediction-network state after u output tokens. An RNN transducer assigns probabilities to label and blank transitions through a joint network. Training marginalizes over alignments compatible with the reference sequence. The TDT formulation additionally predicts a duration d and permits the decoder to advance multiple encoder frames; token and duration outputs are separately normalized [4]. The evaluated TDT configuration uses durations 0 through 4, sigma 0.02 and omega 0.1 in the saved NeMo loss configuration.

$$P(y, d \mid x) = \prod_{u} P_{\text{token}}(y_u \mid h_{t_u}, g_u)\, P_{\text{dur}}(d_u \mid h_{t_u}, g_u) \tag{1}$$

For Stream, the language vector is concatenated with each encoder output and projected before RNN-T decoding. The prompt has 128 entries, and the encoder state has dimension 1,024.

Our mixed-batch path constructs a separate vector for each stream rather than applying one batch-wide language setting. The following equation describes this implemented conditioning, with P denoting the checkpoint's prompt projection:

$$\tilde{h}_t^{(i)} = P\big([h_t^{(i)}; \ell^{(i)}]\big) \tag{2}$$

File evaluation groups examples by the manifest language and restores their original order. English clips use `en-US`; Swahili and mixed Swahili-English clips use `sw-KE`. Automatic language detection is not evaluated for the fine-tuned checkpoint. This distinction matters when interpreting the earlier zero-shot Stream runs, which used the auto prompt (Appendix A).

2.3 Streaming Context and Retained State

Cache-aware inference retains intermediate attention and convolution activations instead of recomputing overlapping windows [5]. Stream uses causal subsampling and convolution and accepts right contexts $r \in \{0, 3, 6, 13\}$ with left context 56. In the measured runtime, the corresponding chunk durations are 80, 320, 560 and 1,120 ms. These configurations share one set of weights. An 80 ms chunk period specifies the arrival and processing unit; it does not specify when the recognizer emits its first token.

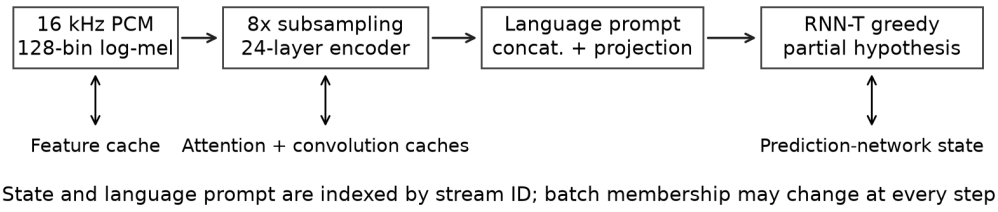

Figure 1: Implemented Stream inference path. Each stream carries acoustic and decoder state independently. The language projection follows the upstream prompted model; the scheduler may combine different languages in one batch.

3 Data and Training

3.1 Corpus Construction

After filtering and evaluation exclusion, the training mix contains 586,510 utterances and 1,518.0 hours. Table 2 reports the accepted source counts and configured mixing weights. The sum of source hours rounded to two decimals is 1,518.02 hours. The mixture includes Common Voice [9], FLEURS [10] and the clean-100 and clean-360 training splits of LibriSpeech [11]. Keystats was included following project-owner confirmation of permission to use the corpus; no license identifier is recorded in the build metadata.

The source sampler uses probabilities proportional to $H_s^{0.6}$, where H_s is accepted duration in hours. Relative to sampling in proportion to duration, this increases exposure to smaller sources. We use this recipe for both models and do not estimate its causal contribution to WER.

$$p_s \propto H_s^{0.6} \tag{3}$$

Training accepts clips lasting 0.5 to 40 seconds and text rates from 2 to 28 characters per second. Targets are lower-case, stripped of noise, music and unknown-token tags, and converted to digit form using the project's Swahili inverse text normalization. The build rejects 468 rows by

Table 2: Accepted training mixture. Weights sum to one before display rounding. They parameterize source selection and should not be interpreted as measured fractions of audio consumed.

Source	Utterances	Hours	Weight
AfriVoice	85,089	478.43	25.74%
Common Voice SW	242,217	365.92	21.91%
Common Voice EN	57,751	91.17	9.52%
FLEURS	2,996	12.73	2.92%
KenSpeech	3,969	16.98	3.47%
Keystats	52,591	79.51	8.77%
LibriSpeech	132,541	464.19	25.28%
OpenSLR	9,356	9.09	2.39%

text rate and 370 for empty or invalid-script text. Validation contains 22,374 accepted utterances. Punctuation and capitalization restoration are not training objectives in this run.

3.2 Evaluation Isolation and Speaker Overlap

Validation and test audio paths are excluded from training. The exclusion manifest records 43,724 clip identities and 256 code-switch holdout speakers. The code-switch set was drawn from AfriVoice health-domain material, and its speakers are removed from training. Thus its result tests held-out speakers within that data source; it is not an external conversational code-switch benchmark.

The standard test split is not speaker-disjoint. The preparation audit found 1,700 source-qualified speaker IDs shared by the input training metadata and standard test, and 1,656 shared with validation. It found no train-test path overlap and no code-switch holdout speaker overlap. These checks operate on metadata identities, not audio fingerprints. Standard-set WER therefore measures held-out utterances under the inherited split, not generalization to entirely unseen speakers. Missing or inconsistent speaker IDs and cross-corpus acoustic duplicates are not ruled out.

3.3 Tokenization and Optimization

Training ran on the CINECA Leonardo system. Each production run used four nodes with four A100-SXM-64GB GPUs per node.

Both runs retain their upstream tokenizers and pretrained prediction networks. A TDT tokenizer check over up to 20,000 AfriVoice training rows and 20,000 English training rows recorded 2.728 subword tokens per Swahili word and 1.716 per English word, with no observed unknown tokens. The word counts were 646,755 and 696,596. This diagnostic motivated retention; it is not evidence that the retained tokenizer is optimal. Stream retains its 13,087-entry vocabulary without a comparable tokenizer ablation.

The Stream Noam schedule has effective learning rate $\eta(t) = 0.15 \times 1024^{-1/2} \times \min(t^{-1/2}, t \times 2000^{-3/2})$, before applying the configured minimum. Its peak is approximately 1.05×10^{-4}. The scale 0.15 is not an optimizer step size comparable directly with TDT's 10^{-4}. Duration-based batching also prevents interpreting 15,000 steps as a fixed number of data epochs.

Both configurations use two frequency masks of width 27 and ten time masks with width parameter 0.05. Checkpoints are ranked by globally aggregated validation word errors divided by reference words, not a mean of rank-specific WERs. The average uses floating-point weights from the best three saved checkpoints; non-floating state is retained from the first selected checkpoint.

Table 3: Fine-tuning configuration. Both models use 16 A100-SXM-64GB GPUs, BF16 mixed precision, AdamW with beta values (0.9, 0.98), weight decay 0.001 and gradient clipping at 1.0.

Setting	TDT	Stream
Updates / validation interval	15,000 / 1,000	15,000 / 1,000
Learning-rate schedule	Cosine; peak 10^{-4}	Noam; scale 0.15; dim 1,024
Warmup / minimum LR	2,500 / 10^{-6}	2,000 / 10^{-6}
Duration budget per GPU	600 s	400 s
Fused-loss microbatch	4	1
Duration bucketing	30 buckets; quadratic 15	Same
Training elapsed time	4 h 31 m 13 s	4 h 34 m 36 s
Best validation WER	12.3593% at 14,000 updates	16.5709% at 15,000 updates
Averaged checkpoints	13,000; 14,000; 15,000	13,000; 14,000; 15,000

Each production job completed successfully. Their combined allocation time is 145.6 GPU-hours, computed from elapsed time multiplied by 16; setup, failed preflights and evaluation are excluded.

4 Experimental Protocol

4.1 Evaluation Sets and Text Normalization

The standard set contains 18,265 clips (29.75 hours); the code-switch set contains 3,005 clips (17.50 hours). All predictions were checked for clip identity, reference equality, output count and order. Empty hypotheses remain in the denominator. File transcription uses one GPU and a configured batch size of 32. Model loading is outside the decode timer. File results are distinct from the cache-aware raw-PCM path used in Sections 6 and 7.

$$\mathrm{WER} = \frac{\sum_i (S_i + D_i + I_i)}{\sum_i N_i} \tag{4}$$

Here S_i, D_i and I_i are the substitution, deletion and insertion counts from minimum-edit alignment for utterance i; N_i is its normalized reference length in words.

We report two normalization views. *Official* applies Unicode normalization, lowercasing, punctuation removal with internal apostrophes retained, whitespace normalization and digit expansion to Swahili words. *Digit-canonical* first applies inverse text normalization and disables number expansion. Official WER preserves comparability with the saved evaluation; digit-canonical WER reduces the effect of Swahili number expansion on English material. Neither is described as a language-neutral universal metric.

Table 4: Reference populations after normalization. Code-switch language attribution uses the original tagged reference tokens, not independently normalized word lists.

Set	Clips	Official words	Digit-canonical words
Standard	18,265	176,692	173,509
Code-switch holdout	3,005	106,614	106,126

4.2 Paired Statistical Analysis

For this report, we recompute word-error totals from the saved prediction files and obtain exact agreement with the experiment records. All comparisons align on unique audio path and require

identical references and source labels. We draw 10,000 bootstrap samples of n utterances with replacement from each n-utterance test set. Each draw is shared across the three configurations. We then recompute corpus WER from summed errors and words, rather than averaging utterance WERs.

$$\Delta_{A-B}^{(b)} = \frac{\sum_{i \in I_b} E_{A,i} - \sum_{i \in I_b} E_{B,i}}{\sum_{i \in I_b} N_i} \tag{5}$$

I_b is the multiset of utterance indices in draw b, and $E_{A,i}$ is the total word-error count for configuration A on utterance i. Thus the paired difference preserves the shared word denominator and accounts for the correlation between predictions on each example.

Intervals are the 2.5th and 97.5th percentiles of the resampled statistic, with linear percentile interpolation. The generator is NumPy PCG64, using base seed 20261006 and fixed offsets per set and normalization. These are utterance-level intervals conditional on the recorded corpora. They do not account for shared speakers, training randomness or distribution shift, and the intervals are not adjusted for multiple comparisons. We use them to quantify the precision of the observed within-corpus differences.

5 File Transcription Results

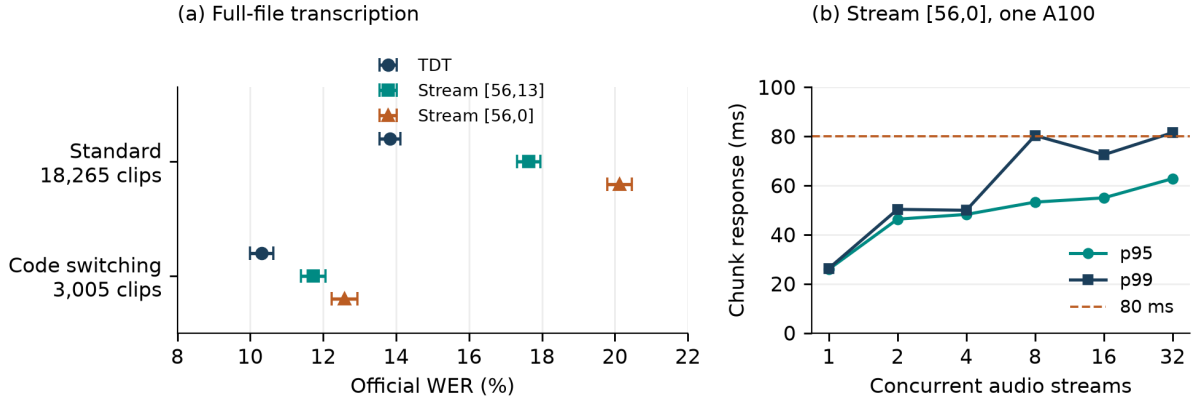

Figure 2: File-transcription accuracy and paced concurrency are separate experiments. Left: official WER with 95% utterance-bootstrap intervals on the complete test sets. Right: non-final chunk response under full scheduled load, including queueing; one GPU, FP32, 80 ms chunks. The highest tested concurrency is 32, not a measured capacity limit.

Table 5: Complete file-transcription results. WER and intervals are percentages. Empty outputs are counted as recognition errors. Each Stream row uses the same fine-tuned checkpoint.

Set	Configuration	Official WER	95% interval	Digit WER	Empty
Standard	TDT	13.83	[13.54, 14.12]	13.63	108
Standard	Stream [56,13]	17.63	[17.32, 17.95]	17.40	143
Standard	Stream [56,0]	20.13	[19.80, 20.46]	19.95	145
Code switch	TDT	10.31	[10.00, 10.63]	10.31	0
Code switch	Stream [56,13]	11.73	[11.39, 12.06]	11.71	0
Code switch	Stream [56,0]	12.58	[12.23, 12.94]	12.56	0

TDT has the lowest WER on both complete sets under both normalization views (Figure 2). Its official WER is 13.83% on standard speech and 10.31% on code-switching. The wider-context

Stream result is 17.63% and 11.73%; the 80 ms setting gives 20.13% and 12.58%. The standard-set official error totals are 24,438, 31,154 and 35,567, respectively. TDT's gain over Stream [56,13] is 3.80 points on standard speech and 1.42 points on code-switching (Table 6).

Table 6: Paired official WER differences in percentage points. Negative values favor the first configuration. Intervals describe utterance resampling, not variation across training runs.

Set	Comparison	Difference	95% interval
Standard	TDT − Stream [56,13]	−3.80	[−3.98, −3.62]
Standard	TDT − Stream [56,0]	−6.30	[−6.50, −6.10]
Standard	Stream [56,13] − Stream [56,0]	−2.50	[−2.65, −2.35]
Code switch	TDT − Stream [56,13]	−1.42	[−1.57, −1.26]
Code switch	TDT − Stream [56,0]	−2.27	[−2.44, −2.10]
Code switch	Stream [56,13] − Stream [56,0]	−0.85	[−0.98, −0.73]

The comparison between Stream [56,13] and [56,0] isolates an inference-context change for one trained model. The reduction is 2.50 points on standard speech and 0.85 points on code-switching, with both intervals below zero. The comparison between TDT and Stream changes the pretrained checkpoint, objective, prompt conditioning and optimization recipe together. It therefore does not identify a causal benefit of the TDT objective alone.

5.1 Source and Code-Switch Error Structure

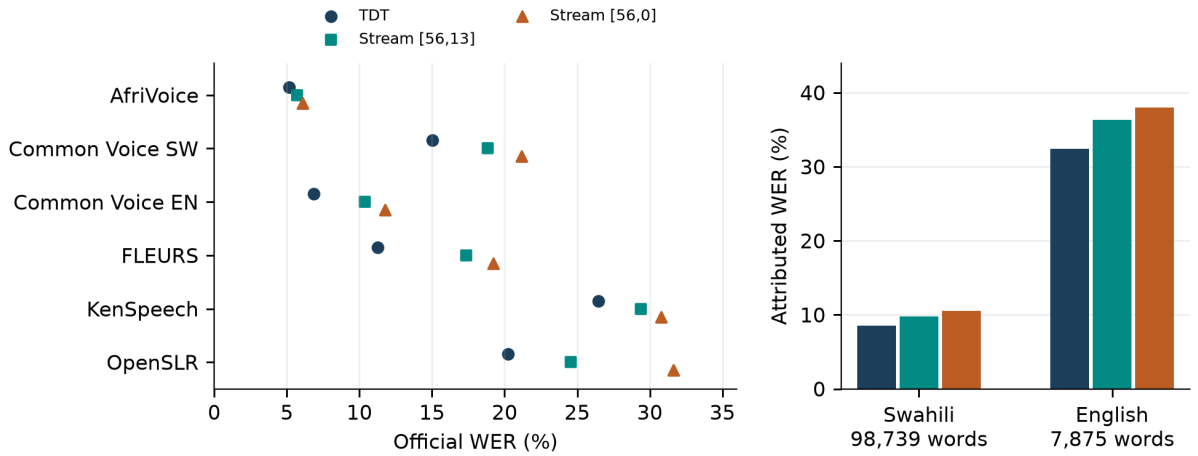

Figure 3: Error structure. Left: newly recomputed official WER by source on the complete standard set. Right: errors attributed to tagged reference languages on the complete code-switch holdout. English comprises only 7.39% of the tagged words. Source-specific digit-canonical scores appear in Appendix B.

The source distribution is highly uneven: Common Voice Swahili accounts for 12,253 of 18,265 standard clips, while AfriVoice and KenSpeech contribute 177 and 213. Source-level differences should therefore accompany the aggregate. Appendix B gives both normalizations and counts. The previously saved `by_source` field uses digit-canonical normalization; Figure 3 is recomputed from predictions using the official normalizer and is labeled accordingly.

The holdout contains 98,739 Swahili words, 7,875 English words and 7,744 switch boundaries. TDT's English-word WER is 32.41%, compared with 8.55% on Swahili. Consequently, 10.31% aggregate code-switch WER does not imply uniformly reliable bilingual output. The language-attributed metric uses original tagged references and a specific insertion rule; it should not be

Table 7: Language-attributed errors within code-switched speech. All values are percentages. A switch is correct only when both adjacent reference words are recognized exactly. Language WER uses insertion attribution to the preceding reference word, or the first word for leading insertions.

Configuration	Swahili WER	English WER	Switch accuracy
TDT	8.55	32.41	54.16
Stream [56,13]	9.77	36.29	49.64
Stream [56,0]	10.56	38.02	47.97

recombined algebraically with independently normalized corpus WER. Its role here is to identify the minority-language error burden.

5.2 Matched Zero-Shot Comparisons

Saved zero-shot predictions permit a matched analysis on the first 200 entries of each test manifest. On those exact examples, TDT changes from 87.95% to 22.49% official WER on the standard prefix, and from 77.56% to 9.46% on the code-switch prefix. All 200 standard-prefix clips are Common Voice Swahili. The sample is an ordered manifest prefix rather than a representative random sample; these numbers cannot replace the full-set results.

The corresponding Stream comparisons also improve substantially, but the base used automatic prompting and the fine-tuned model uses manifest language labels. Thus those observations combine adaptation and prompt-policy changes. Appendix A reports the complete matched table and intervals. No matched contemporary external-system evaluation is available, so this report makes no state-of-the-art claim.

6 Streaming Implementation and Context Study

6.1 Causal Audio Processing and Batched State

The streaming front end consumes raw PCM chunks and retains only the feature history required by the configured cache. Dither is set to zero, padding is controlled explicitly, and utterance-wide normalization is rejected. Encoder attention and convolution caches and the RNN-T hypothesis state are keyed by stream identity. Cache gather and scatter follow current batch membership; finalized streams release their decoder state. The language projection in Equation 2 is applied independently to every batch member.

Before timing, the custom batch-one path is required to match every native partial transcript for representative Swahili, English and code-switch clips. Larger batches must preserve final text and duplicate-stream histories. One English word ending is emitted one chunk earlier in some larger-batch preflight traces. This difference is retained; final-text equivalence does not establish identical partial timing. In the single-stream study, truncated-prefix checks confirm unchanged partial text before the altered end marker. These are behavioral causality checks on tested prefixes, not a formal proof for all inputs.

6.2 Sample and Timing Protocol

The diagnostic pool contains 20 standard Swahili, 20 standard English and 20 code-switch clips, selected with seed 20261005. Swahili selection cycles across five source groups, giving four clips per source. The total duration is 748.54 seconds; the median is 10.52 seconds and the longest clip is 28.92 seconds. For each attention context, all 60 clips are processed without pacing; three

clips per language group are then replayed at audio speed. Two short clips warm up each context before timing.

Unpaced RTFx is audio duration divided by measured inference wall time. Paced first-partial latency is measured from recording start to the first alphanumeric partial, which may later change. Paced response is measured after the required chunk audio has arrived. Model loading, file reads, network transport and endpoint detection are excluded; end-of-file supplies the end-of-stream signal. Inference uses FP32 and high matrix-multiplication precision on one A100 with eight CPU threads.

Table 8: Single-stream context study. All accuracy groups contain 20 unique clips. First-partial medians use only nine paced clips per context and are exploratory. WER uses official normalization.

Context	Chunk ms	Unpaced RTFx	First partial s	SW WER	EN WER	CS WER
[56, 0]	80	3.21	1.47	24.54	11.36	12.57
[56, 3]	320	11.56	1.63	23.01	11.82	13.58
[56, 6]	560	19.94	1.71	23.31	9.55	13.58
[56, 13]	1120	37.69	2.27	22.09	9.55	13.58

Increasing chunk duration from 80 to 1,120 ms raises unpaced RTFx from 3.21 to 37.69 because each invocation processes more audio. The median time to first partial in the nine paced examples rises from 1.47 to 2.27 seconds. The small WER groups do not support a precise ranking among contexts: for example, the 80 ms setting has lower observed code-switch WER than the larger chunks on this subset, despite the opposite ordering in full-file evaluation. These results measure different input paths and sample sizes.

Across all contexts, the verification record covers 240 unpaced and 36 paced streams, with complete audio consumption and matching paced/unpaced final text. This establishes consistency for the selected pool. It does not demonstrate production endpointing behavior, which was not part of the timed path.

7 Concurrent Streaming Evaluation

7.1 Workload Generation and Scheduler

We fix context [56,0] and schedule 1, 2, 4, 8, 16 and 32 audio lanes. Each lane is phase-shifted within one 80 ms interval. Before inference, the sample order is shuffled deterministically and greedily assigned to the lane with the smallest scheduled end time. A new call begins at that lane's scheduled audio end, not when the previous inference finishes. Arrivals therefore continue during a backlog.

$$K(C) = \left\lfloor \frac{120 \cdot C}{D} \right\rfloor + C \tag{6}$$

The schedule targets at least 120 seconds of audio per lane on average; D is the duration of one 60-clip pool. The final term ensures at least C scheduled calls. Equation 6 gives the number of complete pool replays at concurrency C. The six stages contain 60, 60, 60, 120, 180 and 360 calls. The scheduler chooses the oldest ready chunk, then admits at most one chunk per stream with the same end-of-stream flag, up to C members. It inserts no deliberate batch-formation wait. Encoder and feature caches are allocated for all planned calls so that a new arrival can proceed even when an earlier call remains backlogged. This choice is included in reported memory usage.

Four seconds from each representative speech group warm up the native path and each requested batch size. Timing begins after audio loading, warmup and bulk cache allocation. It

includes per-call resets, cache movement, prompt projection, decoding and GPU synchronization. Python scheduling, completion trace writes and progress logging execute within the trial. Measurements therefore describe this implementation rather than isolated GPU kernels.

7.2 Metric Definitions and Analysis Window

For non-final chunk j of stream i, let a_{ij} be scheduled audio readiness, b_{ij} the start of its GPU batch and c_{ij} the batch completion time. We record queue delay Q and complete-batch service time G, with response $R = Q + G$. A batch's full wall time is attributed to each member when reporting chunk response; batch service is counted only once when computing aggregate service RTF.

$$Q_{ij} = b_{ij} - a_{ij}, \quad G_{ij} = c_{ij} - b_{ij}, \quad R_{ij} = c_{ij} - a_{ij} \tag{7}$$

The full-load interval starts after all lane phases have begun and ends when the first lane finishes its final call. The reported chunk population additionally excludes final chunks and retains readiness times from one chunk after the full-load start through one chunk before its end. This is the same population for the response quantiles and full-load deadline counts in Table 9.

$$\text{miss}_{ij} = \mathbf{1}[R_{ij} > 80\,\text{ms}] \tag{8}$$

We distinguish Equation 8 from the stricter next-arrival deadline $c_{ij} > a_{i,j+1}$. The latter can be shorter than 80 ms before a partial terminal chunk. Both are retained in the traces. Quantiles use linear interpolation over ordered observations. Chunk observations share streams and batches; a large chunk count does not imply equally many independent latency samples.

7.3 Response Time and Tail Behavior

Table 9: Full-load non-final chunk results. Response includes queueing. Miss count is the numerator for the final column. All load levels use the same 60 unique clips; repeats increase duration only.

Streams	Full load s	Chunks	p95 ms	p99 ms	Misses	Miss %
1	748.5	9,323	25.9	26.3	0	0.00
2	374.0	9,315	46.3	50.3	20	0.21
4	180.7	8,996	48.2	49.9	3	0.03
8	177.4	17,662	53.2	80.3	178	1.01
16	131.4	26,152	55.0	72.5	75	0.29
32	132.3	52,681	62.8	81.5	680	1.29

At 32 scheduled audio lanes, p95 response is 62.81 ms and p99 is 81.51 ms. Of 52,681 non-final chunks in the full-load window, 680 exceed 80 ms (1.29%). The strict next-arrival rule instead counts 862 misses (1.64%). The full-load interval lasts 132.29 seconds. These observations establish the measured operating point; no search for a saturation boundary or repeated-run capacity estimate was performed.

Tail behavior is not monotonic in load. The largest two-stream response is 533.50 ms, composed of 0.08 ms queue wait and 533.42 ms measured batch service. The largest eight-stream response is 407.89 ms, with 48.20 ms queueing and 359.69 ms service. The service interval includes host work and synchronization, so these observations locate the delay within the measured path but do not identify a GPU, runtime or operating-system cause. None of these events is removed from the analysis.

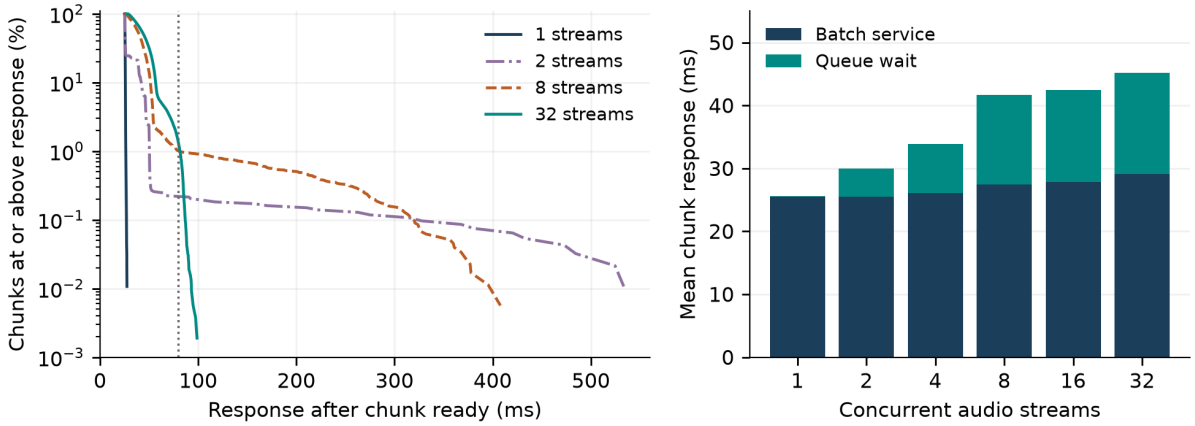

Figure 4: Tail and service analysis on the same full-load non-final chunk population as Table 9. Left: empirical response survival curves, retaining all observed stalls. Right: mean response decomposed into queue wait and attributed batch-service time. Means add exactly; separately computed quantiles would not.

Batching improves aggregate efficiency: the observed mean batch sizes across complete trials are approximately 1.00, 1.00, 1.34, 2.59, 5.21 and 10.76. At 32 lanes the largest batch contains 27 streams, despite a configured maximum of 32. The scheduler processes available work without waiting to fill a batch; concurrency is therefore not synonymous with batch size. At that load, the full-window service-time p95 per member is 30.67 ms and queue-delay p95 is 32.16 ms. These two p95 values must not be added to reconstruct response p95.

7.4 First Text, Memory and Transcript Consistency

Table 10: Per-load first-partial latency and resource use. First-partial columns use the first pass of unique clips only. FP: recording-start to first partial (59 valid clips). SO: estimated speech-onset to first partial (58 valid clips). Memory is peak PyTorch allocation, not total device usage.

Streams	FP median s	FP p95 s	SO median s	SO p95 s	Allocated GiB	Wall RTFx
1	1.31	1.95	0.74	1.58	2.88	1.00
2	1.31	1.96	0.74	1.58	2.90	2.00
4	1.31	1.98	0.75	1.59	2.94	3.80
8	1.32	1.98	0.74	1.60	3.36	7.64
16	1.31	1.98	0.77	1.60	3.81	14.80
32	1.31	1.96	0.76	1.60	5.09	29.42

At 32 lanes, recording-start first-partial latency is 1.31 seconds median and 1.96 seconds p95. The onset-adjusted values are 0.76 and 1.60 seconds. Onset is an offline WebRTC VAD estimate using mode 2, 20 ms frames and the start of the first run of three voiced frames. One unique clip produces no nonempty ASR partial; another has no estimated onset despite a transcript. Missing measurements are retained, and the implementation does not clamp negative onset-adjusted latencies.

Peak allocated memory is 5.09 GiB at 32 lanes, with 7.09 GiB reserved by PyTorch. Cache allocation scales with the planned call pool in this harness, so dividing this number by 32 would not yield a general per-session requirement. Aggregate wall RTFx is 29.42, including the drain interval with fewer active lanes. The sum of batch service times divided by aggregate audio

duration is 0.0333. The final backlog clears 27.2 ms after the last scheduled audio end.

All 840 final call transcripts across the six trials match their corresponding single-stream baseline. Repeated clips show no final-text variation. First-pass official WER is unchanged across concurrency: 24.54% Swahili, 11.36% English and 12.57% code-switching, on 20 unique clips per group. This verifies output consistency on the pool; it does not turn the 840 replays into 840 independent accuracy examples. The 32-lane p95 finalization tail after known file end is 80.4 ms and excludes endpoint detection.

8 Comparison with Sauti Whisper v2.3

This section summarizes a local evaluation conducted between 2026-10-06 and 2026-10-12 on one laptop (RTX 5060 8 GB, i9-14900HX, 30 GB RAM) comparing the two adapted NeMo checkpoints against Sauti Whisper v2.3, a fine-tuned Whisper large-v3-turbo (32 encoder / 4 decoder layers, approximately 0.8 B parameters). The NeMo models use PyTorch FP32; Whisper uses BF16 on GPU with model-card decode settings.

8.1 Evaluation Sets

Two local sets were used. Set A contains 500 clips (143.3 minutes, 71 speakers): 300 plain Swahili and 100 code-switched AfriVoice clips sampled across five domains, plus 100 LibriSpeech test-clean clips. Set B contains 200 randomly sampled FLEURS `sw_ke` test clips (46.6 minutes). Both sets are distinct from the official 18,265-clip standard test. Set A is in-domain for the NeMo models (fp32 TDT: 6.3% here versus 13.8% officially), so absolute figures should not be compared directly with Section 5; relative differences between systems on the same set are the intended use.

8.2 Accuracy by Domain

Table 11: Digit-canonical WER (%) on local evaluation sets. Card figures are from published model cards and were not re-measured. Stream uses right context 13 unless noted.

Set	Source	TDT	Stream [56,13]	Whisper v2.3
Official standard (card)	—	13.63	17.40	10.52
Set A (AfriVoice-style)	local	6.3	7.6–9.6	9.6–10.9
Set B (FLEURS)	local	11.99	20.58	13.50

On the official broad test set, Whisper v2.3 leads by 2.9 points over TDT and 6.9 points over Stream [56,13]. On AfriVoice-style speech (Set A), the NeMo models lead by 3.3 to 4.6 points. On FLEURS (Set B), TDT is level with or slightly ahead of Whisper (11.99% versus 13.50%), while Stream is substantially behind (20.58%). TDT outperforms Stream on every evaluated set, by 1.3 points on Set A to 8.6 points on Set B.

8.3 Stream Deletion Behaviour

On FLEURS, 41% of Stream's errors are deletions, compared with 13% for TDT and 12% for Whisper. Stream skipped whole phrases and returned empty transcripts for 4 of 200 clips. The same behaviour was observed with the Q8 GGUF build, ruling out a PyTorch-specific cause. Changing the language prompt from `sw-KE` to `auto` did not resolve the deletions on Set B, though it substantially improved output on mixed-language live speech (Section 8.4).

8.4 Language Prompt Effect on Mixed Speech

Live conversation sessions (approximately half English) were recorded and replayed offline. Most sessions had been decoded with the `sw-KE` prompt. Replaying the same audio with `auto` produced 1,781 words instead of 1,446 (+23%; per session +0% to +50%) and visibly better English segments. On the benchmark sets, where each clip carries the correct language label, switching to `auto` changed WER by −0.14 (−0.31, +0.03) points on Set A and −0.33 (−1.09, +0.37) on Set B, neither significant. The prompt therefore explains part of the live impression but not the FLEURS deletion behaviour.

8.5 Quantization

Table 12: WER (%) under GGUF quantization on Set A. fp32 is the PyTorch reference. Differences below ±0.2 points are within the resolution of the 500-clip set.

Model	fp32	Q8_0	Q4_K	Δ(fp32 → Q4_K)
TDT	6.3	≈6.3	≈6.3	≈0
Stream [56,13]	7.6	+0.26	≈ +0.5	≈ +0.5
Stream [56,0]	9.6	≈0	≈ +0.5	≈ +0.5

TDT is statistically indistinguishable from FP32 down to Q4_K (421.5 MB). Stream incurs +0.26 points at Q8_0 with context [56,13] and approximately zero at [56,0]; Q4_K costs about +0.5 points. Q8_0 is the recommended default for both models. GGUF Q8_0 runs in 0.9 to 1.0 GB of RAM on CPU (PyTorch FP32: 5.7 to 6.0 GB) and approximately 1.0 GB of VRAM (versus 3.0 GB). Stream on CPU at 80 ms chunks takes approximately 26 ms per chunk with Q8_0, well within the chunk budget.

8.6 Live Pipeline Consistency

Across 10 recorded sessions, live Stream text matched a GPU replay of the same audio with the same model and language prompt at 0.42% word disagreement. Stream on GPU recorded 46 to 64 ms p50 step latency per 1,120 ms chunk; Stream on CPU (PyTorch FP32) recorded approximately 240 to 256 ms, below the chunk duration so no backlog accumulated. Whisper LocalAgreement on GPU recorded 235 to 284 ms p50 step latency (p95: 345 to 605 ms). Over 22.5 minutes of mixed-language conversation, TDT emitted 102 words per minute, Whisper offline 74, Stream `sw-KE` 64 and Stream `auto` 79.

8.7 Engineering Notes

Two implementation issues affected early Stream results. First, NeMo's `transcribe()` leaves 702 of 711 submodules in training mode after the first call (including all dropout layers), making cache-aware replay non-deterministic. Calling `model.eval()` at stream start and after each offline call resolves this; any Stream replay produced before this fix should be treated as suspect. Second, passing `--device cpu` did not prevent NeMo from allocating CUDA when a GPU was visible, skewing CPU timings; the worker now sets `CUDA_VISIBLE_DEVICES=""` when CPU is selected.

9 Limitations

9.1 Comparative Scope

The principal results compare two adapted checkpoints and two contexts of one checkpoint. They do not establish superiority over contemporary external ASR systems. The available zero-shot prefix study is small and ordered, with a prompt-policy confound for Stream. A matched external baseline and a larger, source-stratified base-model comparison remain necessary for broader competitive claims.

9.2 Data and Uncertainty

The standard split shares speakers with training metadata. The code-switch split holds out speakers but remains within one source and domain. Utterance bootstrap intervals do not model speaker clusters or training-seed variation. No full audio-fingerprint audit, independent out-of-domain evaluation or repeated training run is included. English contributes a small fraction of code-switch reference words and remains substantially less accurate.

9.3 System Scope

The concurrency experiment runs once per load on one GPU and one runtime. Its longest full-load interval at 32 streams is about 2.2 minutes, with clips no longer than 28.92 seconds. It does not measure hour-long sessions, network jitter, real microphones, reconnects, overload admission, endpointing or stable-prefix latency. FP32 A100 measurements cannot be projected directly onto a smaller GPU, quantized model or different inference engine.

9.4 Attribution of Gains

Source temperature, retained tokenizers, checkpoint averaging, prompt selection and optimization settings were selected as one recipe. Their effects are not separately ablated. The context comparison is an inference-time experiment; comparing TDT with Stream changes multiple factors. Training allocation time excludes unsuccessful attempts and does not measure GPU utilization, FLOPs or financial cost.

10 Conclusion

The completed adaptation yields two identifiable checkpoints with different operating characteristics. TDT achieves lower full-file WER on both evaluated corpora. Stream provides causal incremental inference with preserved final transcripts under dynamic batching through the highest tested load of 32 audio lanes. The measurements support those statements while exposing three unresolved issues: English-word errors within code switches, service-path stalls, and the absence of end-to-end endpointing and long-session measurements. The next experimental priorities are a speaker-clustered external evaluation, controlled recipe ablations and repeated live-serving load trials.

References

[1] NVIDIA. Parakeet TDT 0.6B v3 model card. https://huggingface.co/nvidia/parakeet-tdt-0.6b-v3

[2] NVIDIA. Nemotron 3.5 ASR Streaming 0.6B model card. https://huggingface.co/nvidia/nemotron-3.5-asr-streaming-0.6b

[3] Dima Rekesh et al. Fast Conformer with Linearly Scalable Attention for Efficient Speech Recognition. *ASRU*, 2023. https://arxiv.org/abs/2305.05084

[4] Hainan Xu et al. Efficient Sequence Transduction by Jointly Predicting Tokens and Durations. *ICML*, 2023. https://proceedings.mlr.press/v202/xu23g.html

[5] Vahid Noroozi et al. Stateful Conformer with Cache-based Inference for Streaming Automatic Speech Recognition. *ICASSP*, 2024. https://arxiv.org/abs/2312.17279

[6] MsingiAI. Sauti NeMo code and model release, 5 October 2026. Private repository. https://github.com/Msingi-AI/sauti-nemo

[7] OpenAI. Whisper large-v3-turbo model card. https://huggingface.co/openai/whisper-large-v3-turbo

[8] NVIDIA. NeMo-Speech.cpp v0.2.0 (Apache-2.0). https://github.com/NVIDIA/NeMo-Speech.cpp

[9] Rosana Ardila et al. Common Voice: A Massively-Multilingual Speech Corpus. *LREC*, 2020. https://aclanthology.org/2020.lrec-1.520/

[10] Alexis Conneau et al. FLEURS: Few-shot Learning Evaluation of Universal Representations of Speech. *arXiv:2205.12446*, 2022. https://arxiv.org/abs/2205.12446

[11] OpenSLR. LibriSpeech ASR corpus. Resource SLR12. https://www.openslr.org/12

Appendices

Appendix A: Matched Zero-Shot Prefix Evaluation

This appendix rescoring uses the first 200 manifest entries evaluated before fine-tuning and the matching rows from the completed fine-tuned evaluation. Audio identities and reference strings match exactly. WER is official; differences are fine-tuned minus base. The confidence intervals use 10,000 paired utterance bootstrap draws, with seeds 20261206, 20261207, 20261216 and 20261217 for the displayed rows.

Table 13: Matched prefix results. Stream base uses auto prompting; Stream fine-tuned uses manifest language. All standard-prefix examples are Common Voice Swahili. These rows do not estimate full-test zero-shot performance.

Set / model	Base WER	Fine WER	Difference pp	95% interval
Standard / TDT	87.95	22.49	−65.46	[−69.31, −61.25]
Standard / Stream [56,13]	96.59	25.85	−70.74	[−75.82, −65.86]
Code switch / TDT	77.56	9.46	−68.10	[−70.03, −66.16]
Code switch / Stream [56,13]	95.81	10.60	−85.21	[−86.87, −83.57]

The TDT rows provide evidence of adaptation on the measured examples under an unchanged no-prompt policy. They do not isolate a particular data source or training intervention. The Stream rows provide descriptive before/after system comparisons; the change in language prompting prevents interpreting their full difference as a fine-tuning-only effect.

Table 14: Complete standard-set source breakdown. Every cell reports official / digit-canonical WER in percent. The saved `score_run by_source` field contains the second quantity. Values below were recomputed from aligned predictions.

Source	Clips	TDT	Stream [56,13]	Stream [56,0]
AfriVoice	177	5.16 / 5.06	5.73 / 5.64	6.08 / 5.97
Common Voice SW	12,253	15.02 / 14.95	18.85 / 18.73	21.18 / 21.11
Common Voice EN	3,144	6.87 / 6.09	10.38 / 9.63	11.76 / 11.03
FLEURS	487	11.26 / 11.45	17.37 / 17.35	19.20 / 19.38
KenSpeech	213	26.46 / 26.37	29.39 / 29.39	30.76 / 30.81
OpenSLR	1,991	20.23 / 20.32	24.54 / 24.73	31.60 / 31.81

Table 15: Paired differences under digit-canonical normalization. All quantities are percentage points. The ordering is unchanged from the official view.

Set	Comparison	Difference	95% interval
Standard	TDT − Stream [56,13]	−3.77	[−3.95, −3.59]
Standard	TDT − Stream [56,0]	−6.32	[−6.52, −6.11]
Standard	Stream [56,13] − Stream [56,0]	−2.54	[−2.69, −2.39]
Code switch	TDT − Stream [56,13]	−1.40	[−1.55, −1.25]
Code switch	TDT − Stream [56,0]	−2.25	[−2.41, −2.08]
Code switch	Stream [56,13] − Stream [56,0]	−0.84	[−0.96, −0.72]

Appendix B: Source-Level Results and Normalization

Appendix C: Error Counts and Benchmark Details

Table 16: Official error decomposition and character error rate (CER). Counts are corpus totals. Empty predictions are included.

Set	Configuration	Substitutions	Deletions	Insertions	CER%
Standard	TDT	17,686	4,663	2,089	5.16
Standard	Stream [56,13]	21,908	6,947	2,299	6.86
Standard	Stream [56,0]	24,603	8,330	2,634	8.22
Code switch	TDT	8,583	1,093	1,321	3.60
Code switch	Stream [56,13]	9,586	1,646	1,276	4.28
Code switch	Stream [56,0]	10,211	1,845	1,360	4.64

The 80 ms diagnostic accuracy groups contain 326, 220 and 692 official reference words for standard Swahili, standard English and code-switching. The corresponding total word errors are 80, 25 and 87. Small word denominators explain why a few changed words can alter the context-study percentages materially. No latency confidence interval is inferred from independent-chunk assumptions.

Appendix D: Runtime and Implementation Records

Training used four nodes with four A100-SXM-64GB GPUs per node. Evaluation and streaming trials used one such GPU with eight CPU threads. The runtime was Python 3.12.15, NeMo 3.0.0

Table 17: Scheduler detail. Batch statistics cover complete trials. Queue and service quantiles use the full-load non-final population. Strict misses use the next-arrival deadline, which may be shorter than 80 ms.

Streams	Calls	Mean batch	Max batch	Queue p95 ms	Service p95 ms	Strict misses
1	60	1.00	1	0.08	25.86	17
2	60	1.00	2	20.93	25.86	38
4	60	1.34	4	21.54	27.33	29
8	120	2.59	8	25.95	27.88	234
16	180	5.21	14	27.12	28.60	154
32	360	10.76	27	32.16	30.67	862

from NVIDIA-NeMo/Speech revision `fd6a877539710e2b98f28c43272ff81312f83417`, PyTorch 2.12.0+cu126, NumPy 2.3.5, Lightning 2.4.0, numba 0.68.0, numba-cuda 0.30.4, cuda-bindings 12.9.9 and cuda-core 1.2.1. The GPU driver was 535.274.02. Conditional CUDA graph decoding was disabled by the runtime on this driver.

The distributed fine-tuning entry point aggregates validation errors and words at epoch end to avoid unequal fused-microbatch collectives. Audio-loader workers are disabled after observed validation-worker hangs. Stream's duration budget was reduced from a failed 600-second preflight to 400 seconds with a fused microbatch of one. On successful completion, checkpoints are synchronously saved, process groups are destroyed and loggers flushed before a success-only direct process exit that avoids a runtime finalization hang. Failed training or saving still exits unsuccessfully.

Table 18: Experiment identities. Scheduler IDs identify original runs; the statistical analyses in this revision do not rerun model inference.

Experiment	Job IDs
TDT / Stream training	59393809 / 59395342
TDT file evaluation	59427501
Stream file evaluation	59427744 at [56,13]; 59428712 at [56,0]
Single-stream context study	59431687
Concurrency study	59437316

The [56,13] score files completed before job 59427744 was intentionally cancelled because the next requested context [56,1] was unsupported. The subsequent job evaluated [56,0]. Score-file completeness and identity checks are therefore the relevant evidence for [56,13], rather than a claim that every evaluation allocation completed normally.

The code release is revision `91e7894905a41193136f19d97cca8c909f03c85b` [6]. It records 113 passing tests. The experiment source snapshots remain authoritative for the measured runs because later release packaging includes path and documentation changes. This report's bootstrap analysis runs locally on the saved prediction files; the exact analysis script and sufficient statistics are retained alongside `analysis.json`.

Appendix E: Artifact Identity and Evidence Index

Upstream references establish architectural provenance. Their benchmark results are not imported as matched Sauti baselines. Project experiment records and analysis outputs are indexed below.

Both model repositories are private. The release verification checked anonymous-read denial and exact agreement of uploaded file size and SHA256 with the averaged checkpoints. The TDT

package preserves upstream CC-BY-4.0 attribution; Stream preserves OpenMDW-1.1 attribution. Dataset permissions remain separate from model access. No training audio is redistributed in the code release.

msingiai/sauti-nemo-tdt-0.6b
https://huggingface.co/msingiai/sauti-nemo-tdt-0.6b
`sauti-nemo-tdt-0.6b.nemo`; 2,509,332,480 bytes
SHA256: `115706eac907cdd2b3efd3cf8ea7ba18bf48efeec3a431b8d4749fd9decb2c80`
Revision: `20c9f2e149ff93d1b8fe201ac11af4bb8d32a4a5`

msingiai/sauti-nemo-stream-0.6b
https://huggingface.co/msingiai/sauti-nemo-stream-0.6b
`sauti-nemo-stream-0.6b.nemo`; 2,553,098,240 bytes
SHA256: `11a0e34dd397334755fadd7419b4b419bf7f9c74376fadcd6db4bfdc061c129f`
Revision: `682df3e88c835e3d63d74a85b5a996ff2f5c92a4`

The following records are relative to the original `sauti-asr` experiment workspace. Their JSON metrics drive the report tables and figures. Raw output and source hashes are retained in those directories; report-side rescoring verifies original error totals before bootstrap analysis.

- **Data audit and zero-shot outputs.** `readiness.json`, `audit_summary.json`, `tokenizer_report.json` and zeroshot prediction files.
  `outputs/nemo_readiness_20261005/`
- **Training.** `build_tdt.json`, `build_stream.json`, `launch.json` and `completion.json`.
  `outputs/nemo_training_20261005/`
- **Complete evaluation.** `results.json`, all six prediction files and evaluation logs containing averaged-checkpoint identities.
  `outputs/nemo_evaluation_20261005/`
- **Streaming study.** `results.json`, 60-clip manifest and per-context traces.
  `outputs/nemo_streaming_benchmark_20261005/`
- **Concurrency study.** `results.json`, arrival plans, chunk traces, batch traces and exact source snapshots.
  `outputs/nemo_concurrency_benchmark_20261005/`
- **New statistical analysis.** `analysis.json`, per-utterance error/word sufficient statistics and full-load response arrays.
  `outputs/nemo_report_analysis_20261006/`
- **Publication record.** `results.json` with repository privacy, revisions, sizes and checkpoint hashes.
  `outputs/nemo_private_release_20261005/`

Shared streaming sample SHA256:
`c2764b55533de872f19de860420c2a56fa3cbb9ce8109556d145ea92d1e0cdf7`
Concurrency raw-archive SHA256:
`ca2441395cd50f566c3788c73ad5502269657a73543dc9a67940cb05017ae5de`