Vesper-IndicEAR: Multilingual Indic Speech Recognition from a Frozen Embedding-Model Audio Encoder

Rahul Sharma
Independent Researcher

***Abstract*—EmbeddingGemma 2, a multimodal embedding model, shares its 305M-parameter audio encoder with Gemma 4 E2B, whose language model reads the encoder's outputs to transcribe speech. We ask how far this encoder, left completely frozen and without the language model, can be turned into a multilingual CTC recogniser by small trained layers. Vesper-IndicEAR keeps the encoder unchanged and trains 11.8M parameters on top: residual adapters on its upper six layers, self-conditioned intermediate CTC over a shared phonetic inventory, a *script-factored* CTC output that predicts one of 11 writing systems and then a sub-word within it, and utterance- and frame-level language identification (LID) that report the language without routing the transcript. The model transcribes 11 Indian languages and English, including code-switched speech with each word in its own script. Trained in two stages (about 4 GPU-hours for a 4-language pilot and 6 for the full run, processing 8,356 hours of permissively licensed audio streamed from the Hugging Face Hub), it identifies the language of 97.8% of FLEURS development utterances and reaches 7.2/13.0% CER on LibriSpeech test-clean/test-other. On speaker-disjoint Indic test sets, scored on identical clips with a script-normalised CER, it reaches 21–32%, against 3–11% for two dedicated Indic recognisers, and leads only on the mean of the MUCS code-switched set. The same encoder read by Gemma 4's language model reports far lower error on Hindi, and linear probes show its states are less linearly decodable than an ASR-trained CTC encoder's, so the limit lies mainly in how much a small reader on frozen states can extract. A frozen embedding encoder therefore supports a working multilingual recogniser with language ID, but not a competitive one without a stronger reader or encoder adaptation. We release the model, the training framework and a benchmark of per-clip outputs for every system.**

***Index Terms*—speech recognition, frozen encoders, adapters, CTC, code-switching, Indian languages, language identification**

I. Introduction

Speech recognisers are usually built on encoders trained for recognition, either supervised on transcribed speech [1], [2] or self-supervised and then fine-tuned [3], [4], [5]. A different kind of audio encoder has recently become widely available: the audio towers of multimodal *embedding* models. EmbeddingGemma 2 [6], [7] embeds speech, text and images into one space so that a spoken query can retrieve a passage by meaning. Its audio tower is the 305M-parameter encoder of Gemma 4 E2B, a Conformer based on the Universal Speech Model whose continuous outputs Gemma 4's language model reads for speech tasks [8], [9]; in a sampled comparison of the released weights, the tower's tensors are identical in the two models except the output projection. The encoder therefore carries speech information. The question is whether it can be read without the language model, by small layers on top of an encoder that an application may already run for retrieval. If so, one frozen encoder could serve retrieval, language identification and transcription together, and adding a language would mean training a few tens of megabytes of output layers rather than a new model.

Indian languages are a demanding test of this idea. The eleven languages we target use ten writing systems (nine Brahmic scripts and Perso-Arabic), plus Latin for English. Speakers switch between their language and English within a sentence, and transcription conventions differ on whether those English words are written in Latin or native script. Hindi and Urdu are largely one spoken language written in two scripts. Most public transcribed speech comes from a few large collections of read, prompted and extempore speech [10], [11], [12].

We set three questions:

1) **Feasibility.** Can a frozen embedding-model audio tower, with only small trained layers, transcribe 11 Indian languages and English?
2) **Script and language.** Can a single output layer write each word in its correct script, and report the language, without a language picker in front of the recogniser?
3) **Gap.** How far is the result from dedicated Indic recognisers on the same clips, and where does the gap come from?

The answers are: yes; mostly, with Hindi–Urdu the main exception; and far: on the Indic test sets its error is 3 to 7 times that of the better dedicated model.

Contributions:

- **A recipe for ASR on a frozen embedding encoder** (Section III): staged execution of the unmodified tower with residual adapters, self-conditioned phonetic CTC, a script-factored CTC output over 11 scripts, LID heads that report but do not route, and decode-time script rules.
- **An account of what each part adds**: linear probes of the frozen tower layer by layer and against two other encoders (Section VI-A), an ablation of the trained components on a fixed 12-language set (Section VI-B), and the effect of the decode-time rules (Section VI-C).
- **A speaker-disjoint comparison on identical clips** with two dedicated Indic recognisers under one script-

normalised metric (Section VI-D), released with every system's per-clip output as a benchmark.

- **A streaming training framework** that trains on 53 Hub-hosted streams without downloading the corpora, and a record of the failure modes we met: input-pipeline starvation, side effects of speaker exclusion, Hindi–Urdu script competition, and a scoring artefact that briefly made an early model look competitive (Section VII, Appendix D).

We are not the first to train recognisers on this encoder (Section II). What this work adds is a single multilingual, multi-script model with language ID on the fully frozen tower and without its language model, an explicit factorisation of the output by script, and a careful measurement of how far that approach goes.

II. Related work

Frozen encoders and parameter-efficient adaptation: Self-supervised speech encoders are routinely evaluated frozen, with light task heads reading their layers [13], and their layers differ markedly in how much phonetic information they expose. Bottleneck adapters [14] and low-rank updates [15] adapt a frozen network with few parameters; in speech, per-language adapters were used for large-scale multilingual recognition [16], and adapters on a frozen Universal Speech Model come within a fraction of a point of full fine-tuning [9]. Our encoder is also USM-based [8], but its released form is part of an embedding model, and in Gemma 4 its outputs are read by a language model rather than a CTC layer; we measure how far a CTC reader gets without that model.

Recognition on the EmbeddingGemma 2 audio tower: In October 2026, community models fine-tuned or probed its audio tower for single-language recognition, among them Hindi, Malayalam and Bangla CTC models on the Hugging Face Hub [17], [18], [19], and a study on Saudi and Gulf Arabic compared frozen and adapted variants [20]. These works establish that the tower can be adapted to one language at a time. We study the fully frozen tower across 12 languages and 11 scripts in one model, with language identification and code-switching, and compare it on identical clips with dedicated Indic recognisers.

Multilingual and code-switched output layers: A single end-to-end model over nine Indian languages was shown to work, but on artificially concatenated Tamil–Hindi utterances it "picks one of the two scripts and sticks with it" [21]. For bilingual Mandarin–English code-switching, outputs have been factorised by language with language-aware encoders and conditional factorisation [22], [23], auxiliary CTC objectives conditioned on language identity have improved massively multilingual recognition [24], and bilingual streaming recognisers have been trained jointly with language identification [25]. For Indian languages, a common label set maps the Brahmic scripts to one phonetic inventory [26], [27], [28], and reduced, shared target sets have been studied for Hindi–English code-switching [29]. IndicConformer [10], [30] uses one output head per language, selected by a language identifier given at inference. Our output layer predicts $p(\text{script})\,p(\text{piece} \mid \text{script})$ over all 11 scripts at every frame, with no language picker, which is a class-factored softmax [31] applied to scripts. We use the shared phonetic inventory only as an intermediate target, with self-conditioning [32], [33] as in hierarchical multilingual CTC [34].

Indic speech recognition: Dedicated Indic recognisers have improved quickly with large transcribed collections: IndicWav2Vec [35], the IndicVoices corpus and IndicConformer models [10], and SraVaani [36], a 430M-parameter FastConformer model with a TDT decoder [2], [37]. Evaluation is complicated by script conventions for English words; script-normalised scoring has been proposed to separate script choice from recognition errors [38], and IndicContextEval evaluates how audio language models use context on speech in eight Indian languages [39]; we use its audio and references as one of our test sets. We adopt a script-normalised CER (Section V) after finding that strict scoring had inflated an early comparison in our favour.

TABLE I
Trained parameters. The 305M-parameter encoder is frozen.

Component	Parameters
Residual adapters, layers 7–12 (rank 128)	1.59M
Phonetic CTC + self-conditioning, layers 6, 9	0.34M
Script-factored output (11 scripts, 8,964 pieces)	9.20M
Utterance LID (12 languages)	0.40M
Frame LID (13 classes)	0.27M
Total	11.8M (45 MB)

III. Model

Figure 1 shows the model. The only trained parts are the 11.8M parameters in orange; Table I lists them.

A. Frozen encoder

The encoder is the audio tower of `google/embeddinggemma-2` at revision `914f7f89` [6], [7]: a USM-based Conformer [40], [9], [8] with two downsampling convolutions to 25 frames per second (40 ms), 12 layers of width 1,024 with 8 attention heads, and chunked attention with chunk size 12, left context 13 and *no right context*. It has 305M parameters. The tower's output projection to the 1,536-dimensional embedding space and the model's text and vision parts are not used.

We run the tower in two stages with its own modules and mask functions, so nothing in it is re-implemented: layers 1–6 without gradient, and layers 7–12 with adapters attached by forward hooks. Padded input frames are zeroed and padded attention keys are masked, so batched computation is exact per utterance; attention sees nothing beyond the current 12-frame chunk, so long audio can be transcribed in overlapping windows.

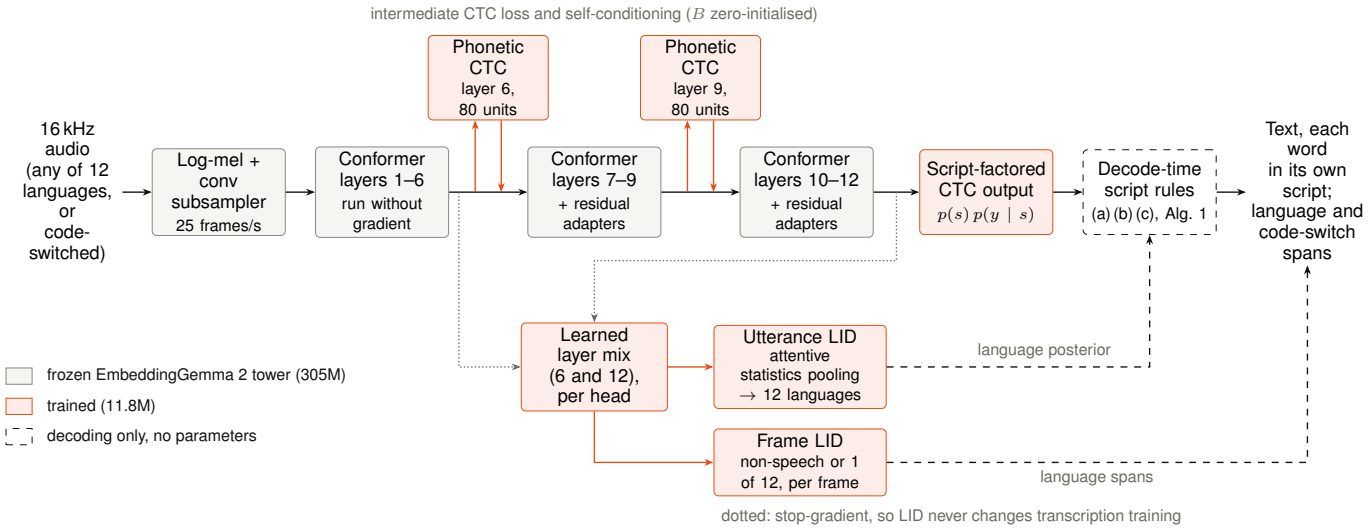


Fig. 1. Vesper-IndicEAR. The EmbeddingGemma 2 audio tower runs unchanged (grey). Trained parts (orange) are residual adapters inside layers 7–12, self-conditioned phonetic CTC heads after layers 6 and 9, the script-factored CTC output on layer 12, and two language-ID heads reading stop-gradient states of layers 6 and 12. Decode-time rules (dashed) use the utterance-LID posterior to restrict which scripts may appear; they have no parameters and can be switched off.

B. Trained components

Residual adapters: In each of layers 7–12, a bottleneck adapter [14] reads the block's second half-step feed-forward output z and adds

$$A_\ell(z) = W_\ell^{\uparrow}\,\mathrm{SiLU}\big(W_\ell^{\downarrow}\,\mathrm{LN}(z)\big) \tag{1}$$

to it, before the block's clamp and output normalisation (Fig. 2a), with $W_\ell^{\downarrow} \in \mathbb{R}^{128\times1024}$. $W_\ell^{\uparrow}$ is zero-initialised, so training starts from the exact frozen encoder. Adapters go only on the upper half because the probes of Section VI-A show the transcription-relevant information growing towards the top of the tower.

Self-conditioned phonetic CTC: After layers $k \in \{6, 9\}$, an intermediate CTC head [32] predicts a shared inventory of 80 phonetic units (82 outputs with blank and a word boundary), and its posterior is fed back into the residual stream [33]:

$$p_k = \mathrm{softmax}\big(W_k\,\mathrm{LN}(h^{(k)})\big), \qquad h^{(k)} \leftarrow h^{(k)} + B_k\,p_k, \tag{2}$$

with B_k zero-initialised. The units are ISO 15919-style labels shared by the Brahmic scripts through their common ISCII block layout [41], [42], Indian-accent spelling rules for English (retroflex t/d, w→v), and a lossy consonant-skeleton romanisation for Urdu. Sharing one inventory lets related languages share acoustic targets in the middle of the encoder, as in hierarchical multilingual CTC [34]. The phonetic units are never the output: a shared inventory can merge words that sound alike but are spelled differently, so the transcript always comes from the script-factored output.

Script-factored CTC output: Each script has its own SentencePiece unigram model [43], [44]: 1,024 pieces for Devanagari and Latin and 768 for each of the other nine scripts, trained with the script's full character inventory as required characters so that any word in it can be spelled. The output layer reads $h_t = \mathrm{LN}(h_t^{(12)})$ and factorises each piece y through its script $s(y)$ (Fig. 2b):

$$\log p(y \mid h_t) = \log p\big(s(y) \mid h_t\big) + \log p\big(y \mid s(y), h_t\big), \tag{3}$$

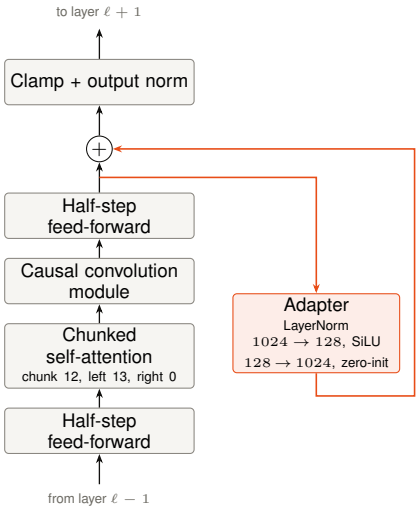
where $p(s \mid h_t)$ is a softmax over the CTC blank (script class 0) and the 11 scripts, and $p(y \mid s, h_t)$ is a softmax over the pieces of script s only. The result is one normalised distribution over blank and all 8,960 pieces (8,964 outputs including blank and three special tokens that are never emitted), trained with ordinary CTC [45]. Compared with a flat softmax, scripts do not compete for probability mass piece by piece, the script of every emitted piece is explicit (which the decode-time rules use), and a script can be added with its own rows without touching the others.

Language identification as an output, not a router: Two LID heads each read their own learned softmax-weighted mix of the layer-normalised states of layers 6 and 12, taken under stop-gradient, so they cannot change transcription training. The utterance head applies attentive statistics pooling [46] and a two-layer MLP (hidden size 256) to predict one of 12 languages; for code-mixed rows the target is the matrix language. The frame head predicts, per 40 ms frame, non-speech or one of the 12 languages. On monolingual rows every valid frame gets the utterance language; on code-mixed rows with word-level language labels, the head is trained with CTC over the sequence of language runs (blank = non-speech), so it learns where switches occur without forced alignment. The transcript never depends on these heads except through the optional script rules below.

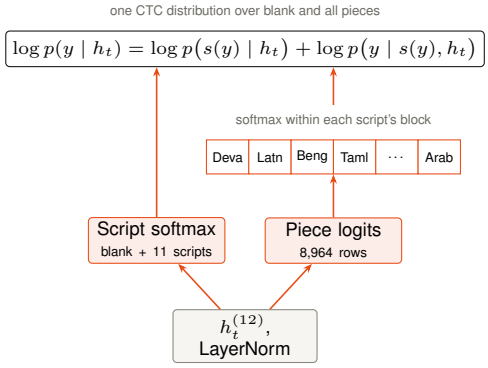
C. Training objective

The loss is

$$\mathcal{L} = 0.85\,\mathcal{L}_{\mathrm{CTC}} + 0.15\,\mathcal{L}_{\mathrm{phon}} + 0.1\,\mathcal{L}_{\mathrm{utt}} + 0.2\,\mathcal{L}_{\mathrm{frame}}, \tag{4}$$



(a) Adapter in a frozen Conformer block (layers 7–12)



(b) Script-factored CTC output

Fig. 2. (a) Adapter placement in a frozen Conformer block of layers 7–12: the adapter reads the second half-step feed-forward output and adds to it before the block's clamp and output norm. (b) The script-factored output: a script softmax and per-script piece softmaxes combine into one CTC distribution (Eq. (3)).

where $\mathcal{L}_{\text{phon}}$ is the mean of the two intermediate CTC losses. The utterance-LID loss uses label smoothing 0.1 [47] and class-balance weights proportional to $h_c^{-1/2}$ for h_c planned training hours of language c; the frame-LID loss is unweighted. The pilot stage (Section IV) used weights 0.3 for the phonetic and 0.1 for the frame-LID loss. Time and feature masks in the style of SpecAugment [48] are applied to the layer-6 states during training (masked values take the utterance mean), on top of the waveform augmentation of Section IV.

D. Decode-time script rules

Greedy and prefix-beam [49] CTC decoding can apply three rules without retraining (Algorithm 1). (a) Within a word, the script may change only from Latin into a piece that starts with a full Indic letter (an English stem with an Indic suffix), and a combining mark may never follow a piece of another script; a Latin piece glued to an Indic word is treated as a missed word boundary and starts a new word. (b) Only Latin, the script of the most likely utterance-LID language, and up to two further Indic scripts whose languages have posterior above 0.15 are allowed. (c) Devanagari and Perso-Arabic never mix in one utterance: the script whose languages have the larger total LID posterior is kept, with ties broken by the decoded evidence for each script. A caller who knows the language can pass it as a hint, which replaces the LID posterior in (b) and (c). Latin always stays allowed, so code-switching into English is never removed by the rules.

Algorithm 1 Decode-time script rules (greedy CTC)

Require: frame log-probs $\log p(y \mid h_t)$, utterance LID posterior $q(\ell)$, optional language hint $\hat{\ell}$

if $\hat{\ell}$ given **then** $q \leftarrow \text{onehot}(\hat{\ell})$
$\mathcal{S} \leftarrow \{\text{Latin}, \text{script}(\arg\max_\ell q)\} \cup \{\text{script}(\ell) : \ell \in \text{top}_2^{\text{Indic scripts}}(q),\ q(\ell) > 0.15\}$ ▷ (b)
if Devanagari $\in \mathcal{S}$ **and** Perso-Arabic $\in \mathcal{S}$ **then** ▷ rule (c)
    drop the one with smaller $\sum_{\ell:\,\text{script}(\ell)=\cdot}\, q(\ell)$
end if
mask every piece whose script $\notin \mathcal{S}$
for $t = 1, \ldots, T$ **do** ▷ rule (a)
    $y_t \leftarrow \arg\max_y \log p(y \mid h_t)$ over pieces allowed after the current word's script (rule (a))
end for
return collapse repeats, remove blanks, detokenise

TABLE II
TRAINING SOURCES AND PLANNED HOURS BEFORE SPEAKER EXCLUSION. ALL STREAM FROM THE HUB AT PINNED REVISIONS.

Source	Languages, licence	Hours
IndicVoices [10]	11 Indic, CC BY 4.0	3,300
Kathbath [11]	11 Indic, CC BY 4.0	1,345
Vaani, transcribed [12]	10 Indic + en, CC BY 4.0	599
LibriSpeech [51]	en, CC BY 4.0	120
People's Speech [52]	en, CC BY 2.0–4.0 subsets	80
VoxPopuli [53]	en, CC0	60
Total (53 streams)		5,504

IV. Data and Training

A. Training data

All training data is CC BY or CC0 and is streamed from the Hugging Face Hub at pinned revisions (Table II). Only development and evaluation sets are downloaded. MUCS 2021 [50] (CC BY-SA) is used for evaluation only, and no non-commercial data is used.

Transcript conventions: Indic words are written in their native script and English words in Latin script. IndicVoices `[gloss]` and Vaani `{gloss}` English annotations replace the native-script word only when the native word sounds like

the English gloss. Kathbath writes English loanwords in native script, so its transcripts go through a loanword normaliser that rewrites them in Latin.

Mixture: The mixture has 53 streams, one per source and language plus code-mixed-only streams. Rows of 0.5–20 s are drawn by temperature sampling [54] ($\alpha = 0.5$) within pinned per-language shares (Fig. 11); code-mixed rows are 15% of draws and non-speech negatives a fixed 2%.

Speaker-disjoint protocol: Every speaker of every development and evaluation set is removed from training. For Kathbath, whose "test-known" speakers are by design also training speakers, this removes 49% (Bengali) to 100% (Odia) of training rows. Our Kathbath scores are therefore speaker-disjoint, unlike those of models trained on Kathbath's training speakers.

B. Two-stage training

Pilot: Hindi, Tamil, Kannada and English, from the frozen encoder with newly initialised heads: 7,423 steps over 2,664 h of audio in about 4.3 A100-hours. Its CER on its own speaker-disjoint development set fell from 22.9 to 16.2.

Full run, warm-started: All 12 languages, starting from the pilot's final weights: 13,394 steps over 8,356 h of audio in about 6 GPU-hours on one RTX PRO 6000 (96 GB). The pilot's Devanagari, Latin, Tamil and Kannada tokenizers and their output rows are kept unchanged; the seven new scripts get their own tokenizers and new output rows initialised with bias -2 and trained at three times the base learning rate. Adapters, phonetic and LID heads continue from the pilot; the optimiser and schedule restart.

Optimisation: AdamW [55], weight decay 0.001, gradient clipping 1.0, bfloat16 autocast; a step whose gradient norm is not finite is skipped. The full run used peak learning rate 6×10^{-4} with 300 warm-up steps and cosine decay to 5×10^{-5}. Batches have a dynamic budget of 71,676 encoder frames (about 48 minutes of padded audio), chosen by an autotuner that targets 85% of GPU memory on probe batches. Half of the rows are augmented on the GPU before the log-mel: additive noise, babble, reverberation (RT60 0.3–0.9 s from 128 synthetic room impulse responses), speed perturbation (0.9/1.1) [56], 8 kHz band-limiting, and GSM, Opus or AMR codecs on 15% of clips. Appendix B lists every setting for both stages.

C. Streaming input pipeline

The mixture is 0.7–1.1 TB of audio against about 190 GB of free disk on the training machine, so nothing is downloaded in advance (Fig. 3). Each stream reads Parquet shards from the Hub through a bounded shard cache and parses, length-checks and filters rows on its own read-ahead thread. A planner draws from the streams by temperature into a shuffle buffer and duration-bucketed batches. Rows are decoded in parallel on free-threaded CPython 3.14 [57], without the global interpreter lock; training uses PyTorch [58] and the encoder implementation of Hugging Face Transformers [59]. The input state is part of every checkpoint, so a resumed run sees exactly the rows it would have seen. After the fixes described in Appendix D, the GPU was 93–99% busy at about 35,000 encoder frames per second, or about 1,390 hours of audio per GPU-hour.

V. Evaluation protocol

Held-out sets: Kathbath clean and noisy test (unseen-speaker and test-known splits) [11], IndicVoices validation [10], Vaani test [12], IndicContextEval [39], MUCS 2021 Hindi and Bengali test [50], LibriSpeech test-clean and test-other [51], VoxPopuli English test [53], Svarah (Indian-accented English) [60], and FLEURS development [61], with six acoustic corruptions of it. FLEURS test was never read during development. No speaker of these sets appears in training (Section IV).

Compared systems: IndicConformer 600M multilingual [30] (revision `2a77d305`, CTC decoder, given the language as it requires) and SraVaani-1.0 [36], [62] (revision `f5dd5358`, automatic language; it does not support Urdu). Each runs its repository's own inference code at the pinned revision. Vesper-IndicEAR is run in two modes: *automatic*, where its own LID drives the script rules, and *language given*, where the clip's language is passed as a hint, matching IndicConformer's input. Both use greedy decoding with the rules of Section III-D. MUCS results in Table IV are on the full test sets; all comparisons with the baselines are on the 300-clip subsets below. For the comparison, every system decodes the same first 300 clips per language of each set, and we release the clip list, the references where their licences allow, and every system's per-clip transcript.

Script-normalised CER: IndicVoices and Vaani references write English words in Latin script, whereas IndicConformer and SraVaani write them in native script. Strict CER counts that as an error. Our script-normalised CER (sn-CER) accepts an English word in either script, scoring each hypothesis against the closest whole-reference variant, in the spirit of SN-WER [38]. Before scoring, a fold removes spelling conventions that are not recognition errors: Malayalam chillus, Bengali khanda ta, the Marathi eyelash ra, Urdu heh variants, tatweel and short-vowel marks, NFKC presentation forms, and native digits. For English-only sets sn-CER equals CER.

VI. Results

A. What the frozen tower already encodes

Before training anything larger, we trained a linear CTC probe on each layer of the frozen tower, one language at a time, on 3–10 hours (7 on average) of FLEURS training speech per language (a LayerNorm and linear layer over a 256-piece SentencePiece unigram vocabulary per language, 6 epochs), and scored it on FLEURS development (Fig. 4). The best Indic probe reaches 16.1–27.1% CER (median 22.7) and English 29.4%. For all languages the best layer is 11 or 12: the information a CTC reader can use grows towards the top of the tower, which motivated adapters on the upper half. On English we ran the same probe on two other encoders: the best layer

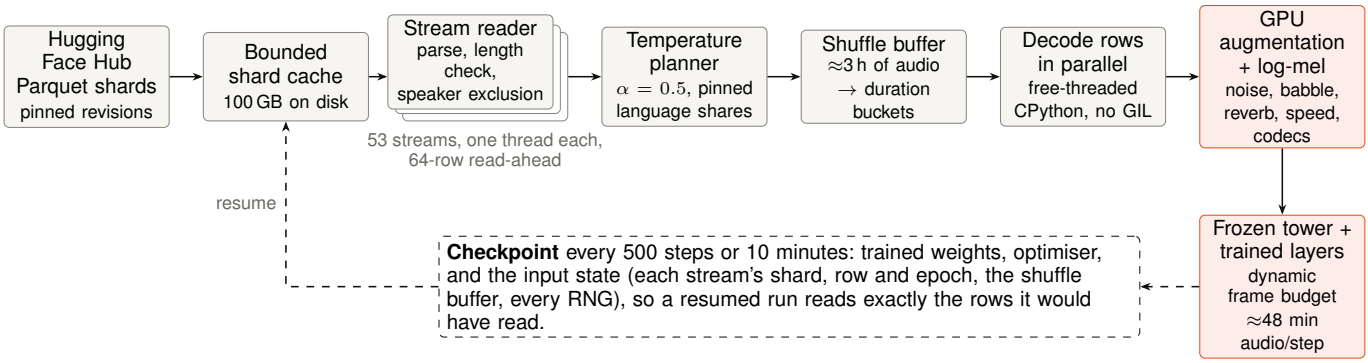


Fig. 3. Streaming training input. 53 (source, language) streams read Hub-hosted Parquet shards through a bounded cache; a temperature planner mixes them into a shuffle buffer and duration buckets, and rows are decoded on parallel threads and augmented on the GPU. The full input state is checkpointed with the weights.

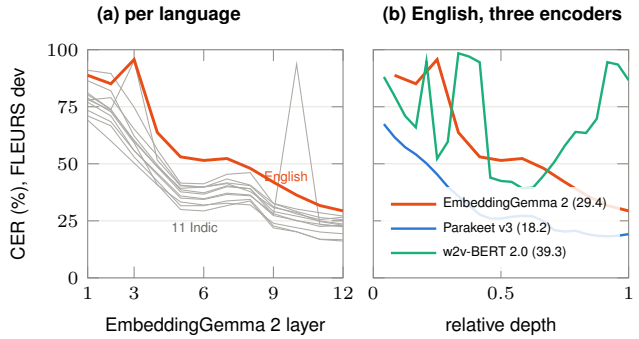


Fig. 4. Linear CTC probes on frozen encoders, FLEURS development. (a) EmbeddingGemma 2 layers 1–12 for 11 Indic languages (grey) and English. (b) English on three encoders by relative depth; the legend gives each encoder's best-layer CER.

TABLE III
ABLATION ON FLEURS (12 LANGUAGES, 88 H). MEAN CER AND SCRIPT-ERROR RATE (SHARE OF WORDS IN THE WRONG SCRIPT) OVER 11 LANGUAGES EXCLUDING URDU; SYNTHETIC CODE-SWITCH: CER OVER SIX CONCATENATED LANGUAGE PAIRS (URDU–ENGLISH EXCLUDED).

	Variant (cumulative)	CER	Script err.	Urdu CER	Code-switch
v1	linear CTC on layer 12	36.2	13.9	55.3	33.9
v2	+ adapters, layers 7–12	27.7	11.2	76.7	**27.3**
v3	+ phonetic self-cond.	28.2	10.5	75.7	28.5
v4	+ script-factored output	**27.6**	**10.0**	81.7	27.4

of the encoder of Parakeet-TDT-0.6B-v3 [63], [2], [37], an ASR-trained FastConformer, reaches 18.2% and that of w2v-BERT 2.0 [64] 39.3%. The EmbeddingGemma 2 tower thus sits between a self-supervised encoder and an ASR-trained CTC encoder: its states carry the content, but less of it is linearly decodable (11 CER points behind on English), which bears on the gap discussed in Section VII-B.

B. What each trained component adds

On a fixed 12-language set (FLEURS training, 88 h, 10 epochs; FLEURS development for scoring) we added the trained components one at a time to the frozen tower (Table III). Adapters bring the largest gain, from 36.2 to 27.7 mean CER over 11 languages. Self-conditioned phonetic CTC did not lower CER at this data scale but reduced script errors; the script-factored output had the lowest CER (27.6) and the fewest script errors (10.0% of words in the wrong script). Each variant ran with one seed and with its own autotuned batch and learning rate (from 111,806 frames, 2.4×10^{-3} and 880 steps for v1 to about 20,000–24,000 frames, 1.0–1.1×10^{-3} and 3,540–4,390 steps for v2–v4, all 10 epochs), so differences of under one point between v2 and v4 should not be read as significant; the script-error reduction guided the choice more than CER. We chose v4 and added the LID heads. Urdu is excluded from the means because, once adapters are added, it is mostly written in Devanagari: Hindi and Urdu share their acoustics and their scripts compete (Section VII-A). Two further variants, CR-CTC consistency regularisation [65] and a factored transducer head, are implemented in the released framework but were not evaluated in this study.

C. Decode-time script rules

The final model makes many script errors when decoded without constraints: on FLEURS development 20.2% of words over all 12 languages are in the wrong script, 64.8% for Urdu. The rules of Section III-D halve that and lower CER from 28.7 to 24.8 (Table IV), with the largest effect on Urdu, Hindi and Gujarati. English is unaffected. On a synthetic set of concatenated sentences in two languages, however, the rules raise CER (for Hindi+English from 16.7 to 30.8). A likely cause, which we have not analysed further, is that rule (b) is decided once per utterance from a single LID posterior, so on these artificial inter-sentential pairs it can remove the script of one of the two halves. Natural intra-sentential code-switching into English is not affected in the same way, because Latin is always allowed; on MUCS the rules lower CER for both Hindi (49.6→45.6) and Bengali (43.9→41.9). Per-frame use of the frame-LID head is the natural fix and is left to future work.

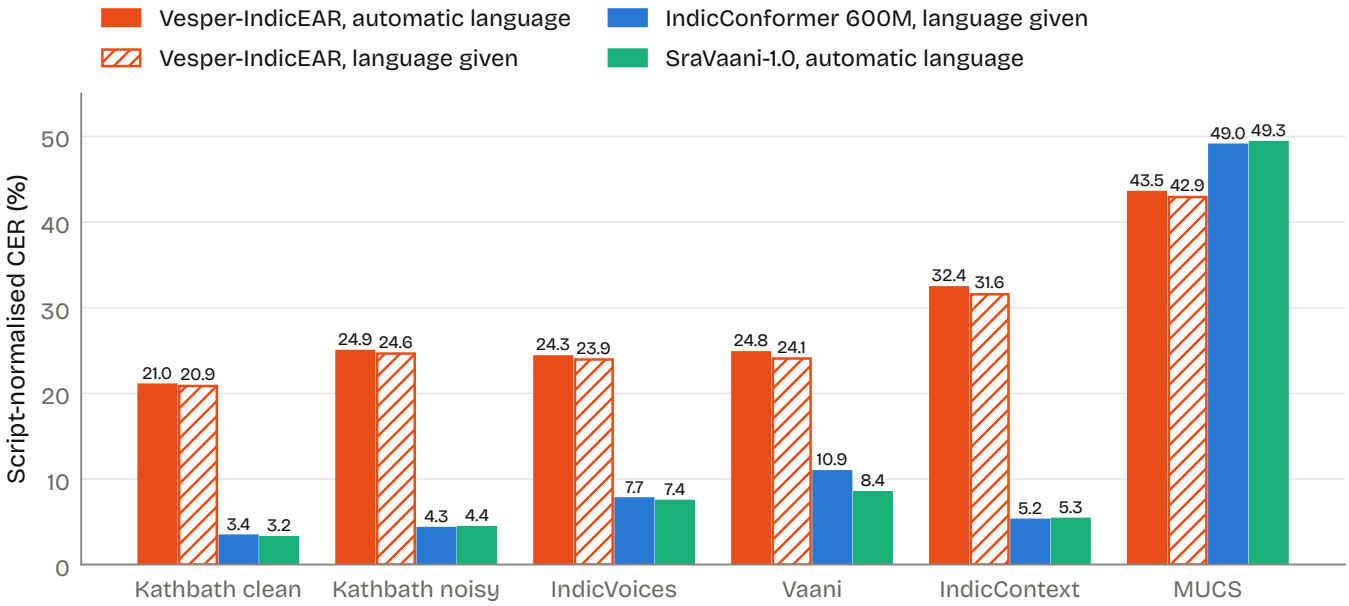


Fig. 5. Held-out sn-CER by evaluation family, as in Table V.

TABLE IV
EFFECT OF THE DECODE-TIME SCRIPT RULES ON THE FINAL MODEL (FLEURS DEVELOPMENT UNLESS NOTED; CER, AND SCRIPT-ERROR RATE IN PARENTHESES).

	Unconstrained	Rules (a)–(c)
All 12 languages	28.7 (20.2)	**24.8** (9.4)
Hindi	31.7 (21.0)	**26.2** (9.8)
Urdu	63.5 (64.8)	**44.4** (31.1)
Gujarati	36.4 (28.3)	**29.6** (11.6)
English	17.2 (0.4)	17.2 (0.0)
MUCS Hindi test	49.6	**45.6**
MUCS Bengali test	43.9	**41.9**
Synthetic Hindi+English pairs	**16.7**	30.8

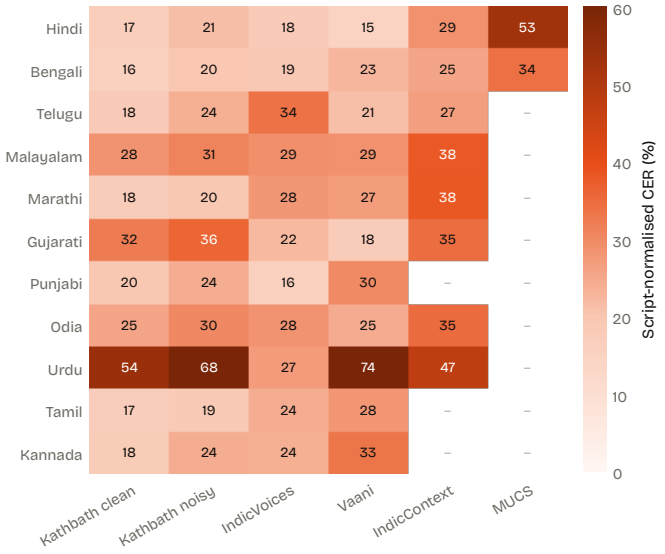


Fig. 6. Vesper-IndicEAR (automatic language) sn-CER per language and evaluation family.

D. Held-out comparison with dedicated recognisers

Table V and Fig. 5 compare the systems on identical clips. Vesper-IndicEAR reaches 21–32% sn-CER on the Indic sets, three to seven times the error of the better dedicated model, and leads only on the MUCS mean, a set of code-switched lecture and tutorial speech on which all systems score poorly; per language it leads on MUCS Bengali (34.2 against 46.4) but trails slightly on MUCS Hindi (52.9 against 51.7). Of 65 (set, language) pairs it is better than the better baseline in one, in both modes (MUCS Bengali). Passing the language helps mainly where the automatic LID is wrong: Urdu (Section VII-A), Hindi on IndicContext (28.7→23.9) and Punjabi on Vaani (29.8→24.1). Errors vary widely by language (Fig. 6). On Kathbath, Gujarati, Malayalam and Odia are clearly harder than Hindi, Bengali, Tamil and Telugu, but the ranking changes across sets (Gujarati is among the easiest on IndicVoices and Vaani), so it reflects the data as much as the language.

E. English, language identification and robustness

English is recognised at 7.2/13.0% CER on the full LibriSpeech test sets, but more weakly on accented and spontaneous speech (Table VI). The utterance LID is correct on 97.8% of FLEURS development utterances; its only material confusion is Urdu speech labelled Hindi (Fig. 7), which bounds the automatic script choice for Urdu. Acoustic corruption degrades the model steeply, and reverberation most of

TABLE V

HELD-OUT SCRIPT-NORMALISED CER (%), MEAN OVER THE LANGUAGES EVERY SYSTEM SUPPORTS (URDU EXCLUDED, AS SRAVAANI HAS NO URDU). EVERY SYSTEM DECODES THE SAME 300 CLIPS PER LANGUAGE PER SET; LOWER IS BETTER, BEST IN BOLD. VESPER-INDICEAR NEVER SAW THESE SETS' SPEAKERS; THE BASELINES MAY HAVE TRAINED ON SOME OF THESE SOURCES. PER-LANGUAGE RESULTS ARE IN APPENDIX A.

	Vesper-IndicEAR		IndicConformer	SraVaani	
Test set	automatic	language given	600M	1.0	Languages
Kathbath clean, unseen speakers	21.0	20.9	3.4	**3.2**	10
Kathbath clean, test-known speakers	20.5	20.4	3.3	**3.1**	10
Kathbath noisy, unseen speakers	24.9	24.6	**4.3**	4.4	10
IndicVoices validation	24.3	23.9	7.7	**7.4**	10
Vaani test	24.8	24.1	10.9	**8.4**	10
IndicContext	32.4	31.6	**5.2**	5.3	7
MUCS 2021 (Hindi, Bengali)	43.5	**42.9**	49.0	49.3	2

TABLE VI

ENGLISH RECOGNITION AND UTTERANCE LANGUAGE IDENTIFICATION.

English test set	CER (%)
LibriSpeech test-clean / test-other (full)	7.2 / 13.0
VoxPopuli test	12.8
FLEURS development	17.2
Vaani English test	16.9
Svarah (Indian-accented English)	25.6
Utterance LID, FLEURS development	
accuracy / macro-F1 (12 languages)	97.8 / 97.1
F1 Hindi / Urdu / all others	84.7 / 85.5 / ≥98.9

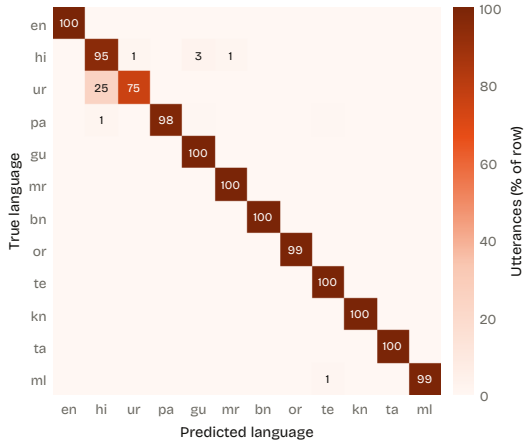


Fig. 7. Utterance-LID confusion on FLEURS development (row-normalised, %).

all (Fig. 8): with RT60 0.9 s CER rises from 24.8 to 46.2, and with RT60 0.6 s reverberation plus point noise at 10 dB to 53.2. Reverberation augmentation in training, which can only shape the adapters and heads, did not remove this weakness.

VII. ANALYSIS

A. Hindi and Urdu: one spoken language, two scripts

Hindi and Urdu speech differ mostly in vocabulary register, and our training data separates them only by the script its transcribers used. The full-run mixture initially gave Urdu

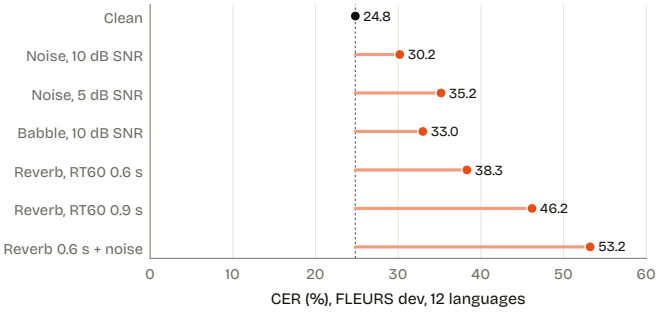


Fig. 8. CER under acoustic corruption of FLEURS development (all 12 languages, decode-time rules on).

7.5% of draws and Hindi 4.5%, since Hindi had already been trained in the pilot. After step 2,000 the model began writing Hindi speech in Urdu script: Hindi CER on the run's development set rose from 24.8 to 44.5 by step 3,000 while Urdu fell from 49.0 to 29.6 (Fig. 9); on 30 Hindi clips, 11 utterances mixed both scripts. Two fixes followed. Decode rules (a) and (b) lowered Hindi CER on those clips from 50.7 to 41.0, and rule (c), which forbids mixing the two scripts in one utterance, to 39.7. From step 3,427, Hindi's share of draws was raised to Urdu's; Hindi recovered to 23.3 by step 4,000 and the balance tipped back towards Devanagari. For the final model, the utterance LID recognises 95% of FLEURS Hindi utterances but labels 25% of Urdu ones as Hindi (Fig. 7), which bounds the automatic script choice: in automatic mode Urdu reaches 54–74% sn-CER on Kathbath and Vaani, and passing the language brings it to 32–50% (Appendix A). A model that is meant to serve both scripts needs either a script preference from the user or a stronger, register-aware language decision.

B. Why the gap to dedicated models

The gap is large and consistent across test families (Table V). In our judgement the causes are, roughly by weight:

1) **Reading the frozen encoder without its language model.** The encoder carries much more than our reader extracts: read by Gemma 4 E2B's language model, it reports 10.1% WER on FLEURS Hindi [8], whereas

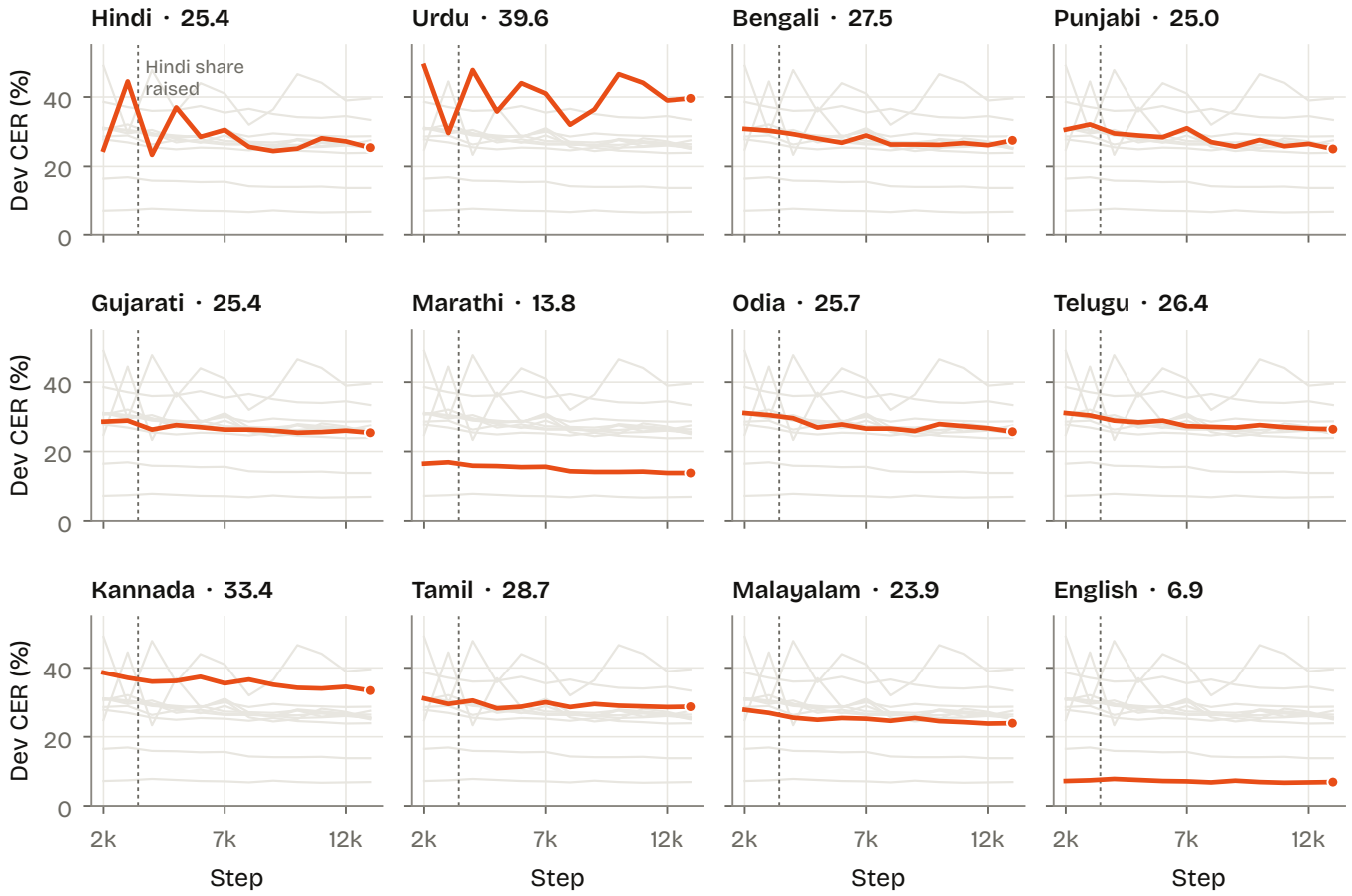


Fig. 9. Full-run CER per language on the run's own development set (unconstrained decoding) (orange) against the other languages (grey), from step 2,000; titles give the value at step 13,000. Hindi and Urdu move in opposite directions between steps 2,000 and 5,000; the dotted line marks the Hindi rebalance at step 3,427.

Vesper-IndicEAR reaches 40.8% WER with the language given on 84 FLEURS Hindi test clips (Appendix E; the text normalisation differs, so the comparison is indicative only). Its states are also less linearly decodable than those of an ASR-trained CTC encoder: the best linear probe trails Parakeet's by 11 CER points on English (Section VI-A). Six rank-128 adapters and CTC heads recover part of this; a deeper or autoregressive reader, or adapting the upper encoder layers, are the natural next steps.

2) **Training data.** About 8,400 hours of audio were processed, much of it seen more than once, whereas SraVaani was pre-trained on 31,255 hours of unlabelled Vaani audio, aligned with images, and fine-tuned on 31,263 hours of labelled speech from 24 datasets [36], and IndicConformer on the IndicVoices collection and other corpora [10].
3) **Speaker-disjoint Kathbath.** Our training saw few Kathbath speakers (Section IV); dedicated models typically train on Kathbath's training speakers, and its test-known split shares them by design. Kathbath is nevertheless where the gap is widest, so this explains only part of it.
4) **Tokenisation.** Marathi averages about 3.4 Devanagari pieces per word, against 1.4 for Hindi, because the Devanagari tokenizer was trained in the pilot on Hindi text only and kept for the warm start.

MUCS is the exception: it is code-switched lecture and tutorial speech on which all systems score poorly. Our training on code-mixed IndicVoices and Vaani rows may help there, but we have not isolated the cause.

C. A scoring artefact we nearly reported

An early comparison of the pilot model scored strict CER and showed it ahead of IndicConformer on Vaani and MUCS. Almost all of that lead came from IndicConformer writing English words in native script while our references write them in Latin; over the same ten languages, IndicConformer's IndicVoices and Vaani errors are 16.4 and 24.9 under strict CER against 7.7 and 10.9 under sn-CER. Script-normalised scoring removed the lead. Comparisons between Indic recognisers trained with different conventions for English words should report a script-normalised metric beside strict CER or WER.

TABLE VII
COMPUTE FOR THE TWO TRAINING STAGES.

	Pilot	Full run
GPU	A100 80 GB	RTX PRO 6000, 96 GB
Training time	≈4.3 GPU-h	≈6 GPU-h (6.9 h wall)
Compute units (Colab)	≈55	≈67 (incl. prep, eval)
Audio per GPU-hour	≈620 h	≈1,390 h
Peak GPU memory	≈76 GB	≈95 GiB

D. Training dynamics and cost

Each run was monitored on its own speaker-disjoint development set with unconstrained decoding (not FLEURS). The pilot's development CER fell from 22.9 to 16.2 over 7,423 steps. The full run added eight languages and seven scripts and reached a macro development CER of 25.1 at step 13,000, most of the gain coming in its first 4,000 steps (Appendix C). Most lost time came from the input pipeline, not the model: before per-stream read-ahead and parallel decoding, the GPU was about 15% busy; afterwards 93–99% (Appendix D). The whole project used about 55 Colab compute units for the pilot and about 67 for the full run including data preparation, restarts and evaluation (Table VII); per hour of audio processed, the RTX PRO 6000 was about 40% cheaper than the A100 once the pipeline kept it busy.

VIII. LIMITATIONS AND RESPONSIBLE USE

Vesper-IndicEAR has several times the error of dedicated Indic recognisers and is not suitable for production transcription, captions with legal or medical weight, or any decision about a person. Automatic LID labels much of the Urdu speech as Hindi, so it is written in Devanagari unless the language is given. Reverberant and far-field audio degrade it sharply. Its training data comes from a few collections of read, prompted and extempore speech, so other domains, accents and recording conditions will be worse than the numbers here. Output has no punctuation, casing or inverse text normalisation, and word times are approximate (40 ms frames, no forced alignment). Accuracy varies widely by language and likely by dialect, gender and recording conditions, which we did not measure. Our comparison uses 300 clips per language per set and the baselines' public checkpoints at pinned revisions; the baselines may have trained on some of the test sources, while Vesper-IndicEAR never saw their speakers. The encoder is used under its licence (Apache-2.0); Google's model card asks that deployments follow the Gemma Prohibited Use Policy.

IX. CONCLUSION

The audio encoder of a multimodal embedding model, left frozen and read without the language model it was built to feed, supports a working multilingual recogniser for 11 Indian languages and English with 11.8M trained parameters, a single script-factored output and language identification that reports without routing. It does not come close to dedicated Indic recognisers: its error is three to seven times theirs on speaker-disjoint test sets. The same encoder read by Gemma 4's language model does far better on Hindi, and its layers are less linearly decodable than an ASR-trained CTC encoder's, so the limit lies mainly in how much a small reader on frozen states can extract rather than in what the encoder holds. For an application that already runs such an embedding model, the result is a cheap multilingual transcription and language-ID capability from one shared encoder. For competitive accuracy, the next steps are a stronger reader on the same encoder or adaptation of its upper layers, measured at matched data.

RELEASE

The model package (frozen tower, trained weights, tokenizers and a checksummed manifest) is at https://huggingface.co/rahulsharmavishwakarma/vesper-indicEAR; the training framework, recipe and run records at https://github.com/rahulsharmavishwakarma/indicear; and the benchmark with every system's per-clip output at https://huggingface.co/datasets/rahulsharmavishwakarma/vesper-indic-bench. All are Apache-2.0, with dataset attributions in the repositories.

ACKNOWLEDGEMENTS

We thank Google for releasing EmbeddingGemma 2 under an open licence; AI4Bharat for IndicVoices, Kathbath, IndicContextEval, Svarah and IndicConformer; ARTPARK and IISc for Vaani and SraVaani; and the creators of LibriSpeech, The People's Speech, VoxPopuli, FLEURS and MUCS 2021.

REFERENCES

[1] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," in *Proceedings of the 40th International Conference on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research, vol. 202. PMLR, 2023, pp. 28 492–28 518.

[2] D. Rekesh, N. R. Koluguri, S. Kriman, S. Majumdar, V. Noroozi, H. Huang, O. Hrinchuk, K. Puvvada *et al.*, "Fast conformer with linearly scalable attention for efficient speech recognition," in *Proc. IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2023, pp. 1–8.

[3] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 12 449–12 460.

[4] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed, "HuBERT: Self-supervised speech representation learning by masked prediction of hidden units," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 3451–3460, 2021.

[5] Y.-A. Chung, Y. Zhang, W. Han, C.-C. Chiu, J. Qin, R. Pang, and Y. Wu, "w2v-BERT: Combining contrastive learning and masked language modeling for self-supervised speech pre-training," in *Proc. IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2021, pp. 244–250.

[6] H. Schechter Vera, S. Dua, J. Andonov, S. Smoot, S. Russo, I. Llanos, M. Matveev, J. H. Yoo *et al.*, "EmbeddingGemma 2: Unified and efficient multimodal representations," 2026.

[7] Google, "EmbeddingGemma 2 (google/embeddinggemma-2)," https://huggingface.co/google/embeddinggemma-2, 2026, hugging Face model; revision 914f7f89142e33e77833254d9c9b90c3cef7303b.

[8] Gemma Team, "Gemma 4 technical report," 2026.

[9] Y. Zhang, W. Han, J. Qin, Y. Wang, A. Bapna, Z. Chen, N. Chen, B. Li *et al.*, "Google USM: Scaling automatic speech recognition beyond 100 languages," 2023.

[10] T. Javed, J. Nawale, E. George, S. Joshi, K. Bhogale, D. Mehendale, I. Sethi, A. Ananthanarayanan *et al.*, "IndicVoices: Towards building an inclusive multilingual speech dataset for Indian languages," in *Findings of the Association for Computational Linguistics: ACL 2024*. Association for Computational Linguistics, 2024, pp. 10 740–10 782.

[11] T. Javed, K. Bhogale, A. Raman, P. Kumar, A. Kunchukuttan, and M. M. Khapra, "IndicSUPERB: A speech processing universal performance benchmark for Indian languages," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 11, 2023, pp. 12 942–12 950, introduces the Kathbath dataset.

[12] S. Pulikodan, A. Singh, A. Basu, N. Desai, P. K. J, P. D. Bhat, R. Dharmaraju, R. Gupta *et al.*, "VAANI: Capturing the language landscape for an inclusive digital India," 2026.

[13] S.-W. Yang, P.-H. Chi, Y.-S. Chuang, C.-I. J. Lai, K. Lakhotia, Y. Y. Lin, A. T. Liu, J. Shi *et al.*, "SUPERB: Speech processing universal PERformance benchmark," in *Proc. Interspeech*, 2021, pp. 1194–1198.

[14] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. de Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for NLP," in *Proceedings of the 36th International Conference on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research, vol. 97. PMLR, 2019, pp. 2790–2799.

[15] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," in *International Conference on Learning Representations (ICLR)*, 2022.

[16] A. Kannan, A. Datta, T. N. Sainath, E. Weinstein, B. Ramabhadran, Y. Wu, A. Bapna, Z. Chen, and S. Lee, "Large-scale multilingual speech recognition with a streaming end-to-end model," in *Proc. Interspeech*, 2019, pp. 2130–2134.

[17] epoch-discovery, "ep-hindi-asr," https://huggingface.co/epoch-discovery/ep-hindi-asr, 2026, hugging Face model; accessed 2026-10-29 at revision 1d0c3db1f9c3c3f03d9faf031a15d682785ab761.

[18] whiletruelearn, "dulu-malayalam-ctc-778h," https://huggingface.co/whiletruelearn/dulu-malayalam-ctc-778h, 2026, hugging Face model; accessed 2026-10-29 at revision d6a6b905c24696ec82f187ba89142f15355c6522.

[19] nehsan, "KothaGemma," https://huggingface.co/nehsan/KothaGemma, 2026, hugging Face model; accessed 2026-10-29 at revision c0e92637ce8ee78bf831610482bf4cebe500f5dc.

[20] A. Abdelkhalik, "How far can the Gemma 4 speech encoder go without changing its weights? Saudi and Gulf Arabic speech recognition with a frozen and an adapter-augmented encoder," 2026.

[21] S. Toshniwal, T. N. Sainath, R. J. Weiss, B. Li, P. Moreno, E. Weinstein, and K. Rao, "Multilingual speech recognition with a single end-to-end model," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 4904–4908.

[22] J. Tian, J. Yu, C. Zhang, Y. Zou, and D. Yu, "LAE: Language-aware encoder for monolingual and multilingual ASR," in *Proc. Interspeech*, 2022, pp. 3178–3182.

[23] B. Yan, C. Zhang, M. Yu, S.-X. Zhang, S. Dalmia, D. Berrebbi, C. Weng, S. Watanabe, and D. Yu, "Joint modeling of code-switched and monolingual ASR via conditional factorization," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 6412–6416.

[24] W. Chen, B. Yan, J. Shi, Y. Peng, S. Maiti, and S. Watanabe, "Improving massively multilingual ASR with auxiliary CTC objectives," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023, pp. 1–5.

[25] S. Punjabi, H. Arsikere, Z. Raeesy, C. Chandak, N. Bhave, A. Bansal, M. Müller, S. Murillo *et al.*, "Streaming end-to-end bilingual ASR systems with joint language identification," 2020.

[26] M. G. Kumar, J. Kuriakose, A. Thyagachandran, A. K. A, A. Seth, L. V. S. V. D. Prasad, S. Jaiswal, A. Prakash, and H. A. Murthy, "Dual script E2E framework for multilingual and code-switching ASR," in *Proc. Interspeech*, 2021, pp. 2441–2445.

[27] A. A, M. Batra, and U. S, "DuDe: Dual-decoder multilingual ASR for Indian languages using common label set," 2022.

[28] A. Gangwar, K. Jayakumar, and S. Umesh, "Building robust and scalable multilingual ASR for Indian languages," 2025, IIT Madras SPRING Lab system description for the ASRU MADASR 2.0 challenge.

[29] K. Dhawan, G. Sreeram, K. Priyadarshi, and R. Sinha, "Investigating target set reduction for end-to-end speech recognition of Hindi-English code-switching data," in *Proc. National Conference on Communications (NCC)*, 2020, pp. 1–5.

[30] AI4Bharat, "IndicConformer 600m multilingual (ai4bharat/indic-conformer-600m-multilingual)," https://huggingface.co/ai4bharat/indic-conformer-600m-multilingual, 2025, hugging Face model; revision 2a77d305038dfc502fd9464e7d028b90381976cb.

[31] J. Goodman, "Classes for fast maximum entropy training," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 1, 2001, pp. 561–564.

[32] J. Lee and S. Watanabe, "Intermediate loss regularization for CTC-based speech recognition," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 6224–6228.

[33] J. Nozaki and T. Komatsu, "Relaxing the conditional independence assumption of CTC-based ASR by conditioning on intermediate predictions," in *Proc. Interspeech*, 2021, pp. 3735–3739.

[34] W. Liu, J. Hou, D. Yang, M. Cao, and T. Lee, "LUPET: Incorporating hierarchical information path into multilingual ASR," in *Proc. Interspeech*, 2024, pp. 3979–3983.

[35] T. Javed, S. Doddapaneni, A. Raman, K. S. Bhogale, G. Ramesh, A. Kunchukuttan, P. Kumar, and M. M. Khapra, "Towards building ASR systems for the next billion users," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 10, 2022, pp. 10 813–10 821.

[36] S. Pulikodan, A. Basu, P. K. J, P. D. Bhat, S. Shukla, N. Desai, and P. K. Ghosh, "SraVaani 1.0: Scaling inclusive speech recognition for Indic languages," 2026.

[37] H. Xu, F. Jia, S. Majumdar, H. Huang, S. Watanabe, and B. Ginsburg, "Efficient sequence transduction by jointly predicting tokens and durations," in *Proceedings of the 40th International Conference on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research, vol. 202. PMLR, 2023, pp. 38 462–38 484.

[38] P. Pattnayak, "SN-WER: Script-normalized WER for multi-script Indic ASR evaluation," in *Proceedings of the 1st Workshop on Multilinguality in the Era of Large Language Models (MeLLM 2026)*. Association for Computational Linguistics, 2026, pp. 249–256.

[39] S. Joshi, D. S. Rathi, S. Singh, E. I. George, R. J. Hari, K. Bhogale, and M. M. Khapra, "IndicContextEval: A benchmark for evaluating context utilisation in audio large language models across 8 Indic languages," 2026, accepted at Interspeech 2026 according to the arXiv comment.

[40] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang *et al.*, "Conformer: Convolution-augmented transformer for speech recognition," in *Proc. Interspeech*, 2020, pp. 5036–5040.

[41] International Organization for Standardization, "ISO 15919:2001 information and documentation – transliteration of Devanagari and related Indic scripts into Latin characters," https://www.iso.org/standard/28333.html, 2001, international Standard, edition 1, ISO/TC 46.

[42] Bureau of Indian Standards, "IS 13194:1991 indian script code for information interchange – ISCII," Bureau of Indian Standards, New Delhi, 1991, indian Standard, December 1991.

[43] 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*. Association for Computational Linguistics, 2018, pp. 66–71.

[44] T. Kudo, "Subword regularization: Improving neural network translation models with multiple subword candidates," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2018, pp. 66–75.

[45] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd International Conference on Machine Learning (ICML)*. ACM, 2006, pp. 369–376.

[46] K. Okabe, T. Koshinaka, and K. Shinoda, "Attentive statistics pooling for deep speaker embedding," in *Proc. Interspeech*, 2018, pp. 2252–2256.

[47] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2818–2826.

[48] 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," in *Proc. Interspeech*, 2019, pp. 2613–2617.

[49] A. Y. Hannun, A. L. Maas, D. Jurafsky, and A. Y. Ng, "First-pass large vocabulary continuous speech recognition using bi-directional recurrent DNNs," 2014.

[50] A. Diwan, R. Vaideeswaran, S. Shah, A. Singh, S. Raghavan, S. Khare, V. Unni, S. Vyas *et al.*, "MUCS 2021: Multilingual and code-switching

ASR challenges for low resource Indian languages," in *Proc. Interspeech*, 2021, pp. 2446–2450.

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

[52] D. Galvez, G. Diamos, J. Torres, K. Achorn, J. Cerón, A. Gopi, D. Kanter, M. Lam, M. Mazumder, and V. Janapa Reddi, "The people's speech: A large-scale diverse English speech recognition dataset for commercial usage," in *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, vol. 1, 2021.

[53] C. Wang, M. Rivière, 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)*. Association for Computational Linguistics, 2021, pp. 993–1003.

[54] N. Arivazhagan, A. Bapna, O. Firat, D. Lepikhin, M. Johnson, M. Krikun, M. X. Chen, Y. Cao *et al.*, "Massively multilingual neural machine translation in the wild: Findings and challenges," 2019.

[55] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations (ICLR)*, 2019.

[56] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio augmentation for speech recognition," in *Proc. Interspeech*, 2015, pp. 3586–3589.

[57] S. Gross, "PEP 703 – making the global interpreter lock optional in CPython," https://peps.python.org/pep-0703/, 2023, python Enhancement Proposal, Standards Track; status Final.

[58] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[59] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault *et al.*, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, 2020, pp. 38–45.

[60] T. Javed, S. Joshi, V. Nagarajan, S. Sundaresan, J. Nawale, A. Raman, K. Bhogale, P. Kumar, and M. M. Khapra, "Svarah: Evaluating English ASR systems on Indian accents," in *Proc. Interspeech*, 2023, pp. 5087–5091.

[61] A. Conneau, M. Ma, S. Khanuja, Y. Zhang, V. Axelrod, S. Dalmia, J. Riesa, C. Rivera, and A. Bapna, "FLEURS: Few-shot learning evaluation of universal representations of speech," in *Proc. IEEE Spoken Language Technology Workshop (SLT)*, 2023, pp. 798–805, SLT 2022, held January 2023.

[62] ARTPARK-IISc, "SraVaani-1.0 (artpark-iisc/sravaani-1.0)," https://huggingface.co/ARTPARK-IISc/SraVaani-1.0, 2026, hugging Face model; revision f5dd5358325a5208775b91dad98918e079ea2b27.

[63] NVIDIA, "Parakeet TDT 0.6b v3 (nvidia/parakeet-tdt-0.6b-v3)," https://huggingface.co/nvidia/parakeet-tdt-0.6b-v3, 2025, hugging Face model; revision 541d1f99c6b0c3cd0b11a95167540bb8edefd82b.

[64] Seamless Communication, L. Barrault, Y.-A. Chung, M. C. Meglioli, D. Dale, N. Dong, M. Duppenthaler, P.-A. Duquenne *et al.*, "Seamless: Multilingual expressive and streaming speech translation," 2023.

[65] Z. Yao, W. Kang, X. Yang, F. Kuang, L. Guo, H. Zhu, Z. Jin, Z. Li, L. Lin, and D. Povey, "CR-CTC: Consistency regularization on CTC for improved speech recognition," in *International Conference on Learning Representations (ICLR)*, 2025. [Online]. Available: https://openreview.net/forum?id=CIs9x2ZRgh

[66] R. Sharma, "Vesper ASR benchmark," Hugging Face dataset, https://huggingface.co/datasets/rahulsharmavishwakarma/vesper-asr-benchmark, 2026.

[67] NVIDIA, "Nemotron 3.5 ASR streaming 0.6b (nvidia/nemotron-3.5-asr-streaming-0.6b)," https://huggingface.co/nvidia/nemotron-3.5-asr-streaming-0.6b, 2026, hugging Face model; accessed 2026-10-29 at revision ea30d66debe3740a08b573244286791d423d6b3e.

[68] X. Shi, X. Wang, Z. Guo, Y. Wang, P. Zhang, X. Zhang, Z. Guo, H. Hao *et al.*, "Qwen3-ASR technical report," 2026.

APPENDIX A
PER-LANGUAGE RESULTS

Table VIII gives every (set, language) pair behind Table V.

APPENDIX B
HYPERPARAMETERS

Table IX lists the settings of both stages; the exact command lines and per-step logs are in the released run records.

APPENDIX C
TRAINING CURVES AND INPUT DATA

Two data-steering changes were applied during the full run without changing its configuration, so it resumed rather than restarted: from step 1,079 the Kathbath Malayalam, Odia, Urdu and Gujarati streams were no longer drawn (after speaker exclusion, Malayalam would have been repeated about 310 times), and from step 3,427 Hindi's share was raised from 4.5% to about 7.5%, equal to Urdu's (Section VII-A).

APPENDIX D
ENGINEERING RECORD

Table X records every problem that cost time or compute, and Fig. 12 shows its effect on throughput. The main lesson is to load-test a streaming loader on the target machine, with every stream and the real speaker exclusion, before a long run.

APPENDIX E
ADDITIONAL COMPARISON ON PUBLIC BENCHMARK CLIPS

The public Vesper ASR benchmark [66] records transcripts of other speech models on public clips. We decoded the clips it shares with our languages and scored every system with one scorer (same normalisation, strict script), each on the clips all systems cover (Fig. 13). On FLEURS Hindi, Nemotron 3.5 ASR [67] and Qwen3-ASR [68] are two to three times more accurate; on English, Vesper-IndicEAR is far behind every model in the benchmark; on 16 MUCS Hindi–English clips it has the lowest CER, consistent with Section VI-D, but 16 clips are too few to rank systems.

TABLE VIII

PER-LANGUAGE HELD-OUT SN-CER (%), 300 CLIPS PER LANGUAGE PER SET. A: VESPER-INDICEAR AUTOMATIC; G: LANGUAGE GIVEN; IC: INDICCONFORMER 600M (LANGUAGE GIVEN); SV: SRAVAANI-1.0 (NO URDU). BEST PER ROW IN BOLD.

	Kathbath clean				Kathbath clean, known				Kathbath noisy				IndicVoices			
Language	A	G	IC	SV	A	G	IC	SV	A	G	IC	SV	A	G	IC	SV
Hindi	16.9	16.6	2.5	**2.2**	16.9	16.7	2.4	**2.2**	21.4	20.6	3.3	**3.2**	18.4	17.0	4.9	**4.7**
Bengali	16.2	16.2	2.2	**1.9**	19.9	19.7	2.2	**2.0**	20.1	20.0	**3.0**	3.1	19.2	18.9	**3.4**	3.5
Telugu	18.5	18.5	3.7	**3.5**	17.6	17.6	3.8	**3.6**	23.8	23.8	**4.4**	4.6	33.8	33.5	**10.7**	11.3
Malayalam	28.4	28.4	3.6	**3.2**	29.4	29.4	3.2	**2.8**	31.0	30.8	3.6	**3.4**	29.1	29.0	10.2	**9.6**
Marathi	17.7	17.7	4.3	**4.0**	17.0	17.0	4.8	**4.5**	19.7	19.4	**4.2**	4.3	27.7	27.3	6.9	**6.2**
Gujarati	32.2	31.9	**3.2**	**3.2**	28.0	27.9	2.9	**2.8**	36.1	35.5	**4.7**	5.3	22.1	21.8	**6.4**	7.3
Punjabi	19.6	18.9	**3.7**	**3.7**	17.8	17.3	**3.5**	**3.5**	24.4	23.8	**5.3**	5.9	16.3	16.2	5.5	**5.3**
Odia	25.3	25.3	3.6	**3.5**	25.2	25.2	**3.6**	**3.6**	30.0	30.0	**4.4**	4.5	28.5	28.1	9.7	**9.0**
Urdu	53.9	32.5	**2.6**	–	57.0	34.4	**2.6**	–	68.0	41.8	**2.4**	–	27.0	20.4	**2.7**	–
Tamil	17.3	17.2	4.2	**3.8**	16.2	16.1	3.8	**3.3**	18.6	18.4	4.5	**4.1**	24.1	24.1	9.1	**8.3**
Kannada	18.0	17.8	**2.7**	2.9	17.1	17.1	3.0	**2.9**	24.3	24.1	**5.2**	**5.2**	23.8	23.7	10.4	**9.1**

	Vaani				IndicContext				MUCS			
Language	A	G	IC	SV	A	G	IC	SV	A	G	IC	SV
Hindi	14.7	14.2	6.4	**3.8**	28.7	23.9	**5.9**	6.2	52.9	52.0	**51.7**	51.9
Bengali	22.9	22.5	8.7	**4.5**	25.5	25.4	3.4	**3.2**	34.2	**33.9**	46.4	46.8
Telugu	21.2	21.2	7.3	**6.6**	26.6	26.6	**3.6**	**3.6**	–	–	–	–
Malayalam	28.7	28.7	9.9	**9.0**	38.1	38.1	**7.1**	7.5	–	–	–	–
Marathi	26.7	26.5	11.1	**9.4**	38.4	38.0	**5.2**	5.5	–	–	–	–
Gujarati	17.8	17.7	5.1	**3.5**	34.7	34.3	2.4	**2.2**	–	–	–	–
Punjabi	29.8	24.1	**8.9**	10.6	–	–	–	–	–	–	–	–
Odia	24.5	24.5	12.9	**10.3**	34.7	34.7	**9.0**	9.2	–	–	–	–
Urdu	74.2	50.2	**19.2**	–	47.2	43.4	**13.1**	–	–	–	–	–
Tamil	28.0	27.9	18.4	**8.3**	–	–	–	–	–	–	–	–
Kannada	33.3	33.3	20.2	**18.7**	–	–	–	–	–	–	–	–

TABLE IX

TRAINING SETTINGS OF THE TWO STAGES. BATCHES ARE DYNAMIC FRAME BUDGETS AT 25 ENCODER FRAMES PER SECOND OF AUDIO.

	Pilot (`pilot-v4lid`)	Full run (`full_warm`)
Languages	hi, ta, kn, en	11 Indic + en
Initialisation	frozen encoder, new heads	pilot final (step 7,423)
Batch	67,045 frames (≈45 min padded audio)	71,676 frames (≈48 min), 2 micro-batches from step ≈350
Packing (real/padded frames)	0.48	0.784
Steps / audio processed	7,423 / 2,664 h	13,394 / 8,356 h
Learning rate	1.83×10^{-3}, 300 warm-up, cosine to 5%	6×10^{-4}, 300 warm-up, cosine to 5×10^{-5}
New output rows	–	learning rate $\times 3$, bias -2
Loss weights (phon., utt. LID, frame LID)	0.3, 0.1, 0.1	0.15, 0.1, 0.2
Optimiser	AdamW, weight decay 0.001, gradient clip 1.0, seed 0; non-finite gradient norm skips the step (full run)	
Precision	bfloat16 autocast (a 21-step resume on an Apple GPU used float16 with gradient scaling)	
Augmentation weights (50% of rows clean)	noise .25, babble .20, reverb .175, reverb + point noise .175, speed .15, 8 kHz .05	reverb .30, reverb + point noise .175, noise .20, babble .15, speed .125, 8 kHz .05
Augmentation detail	SNR 5–20 dB; RT60 0.3–0.9 s, 128 synthetic RIRs; speed 0.9/1.1; GSM/Opus/AMR on 15% of clips; on the GPU, seeded per row	
Hidden-state masking	2 time masks per 100 frames (width $\leq$10), 2 feature masks (width $\leq$64) on layer-6 states	
Checkpoints	every 500 steps or 10 minutes; development evaluation every 1,000 steps	
Hardware	A100-SXM4 80 GB	RTX PRO 6000 Blackwell Server Edition, 96 GB, 48 vCPU
Software	CPython 3.14 free-threaded, PyTorch 2.14.1 (CUDA 13.0), transformers 5.20.2	

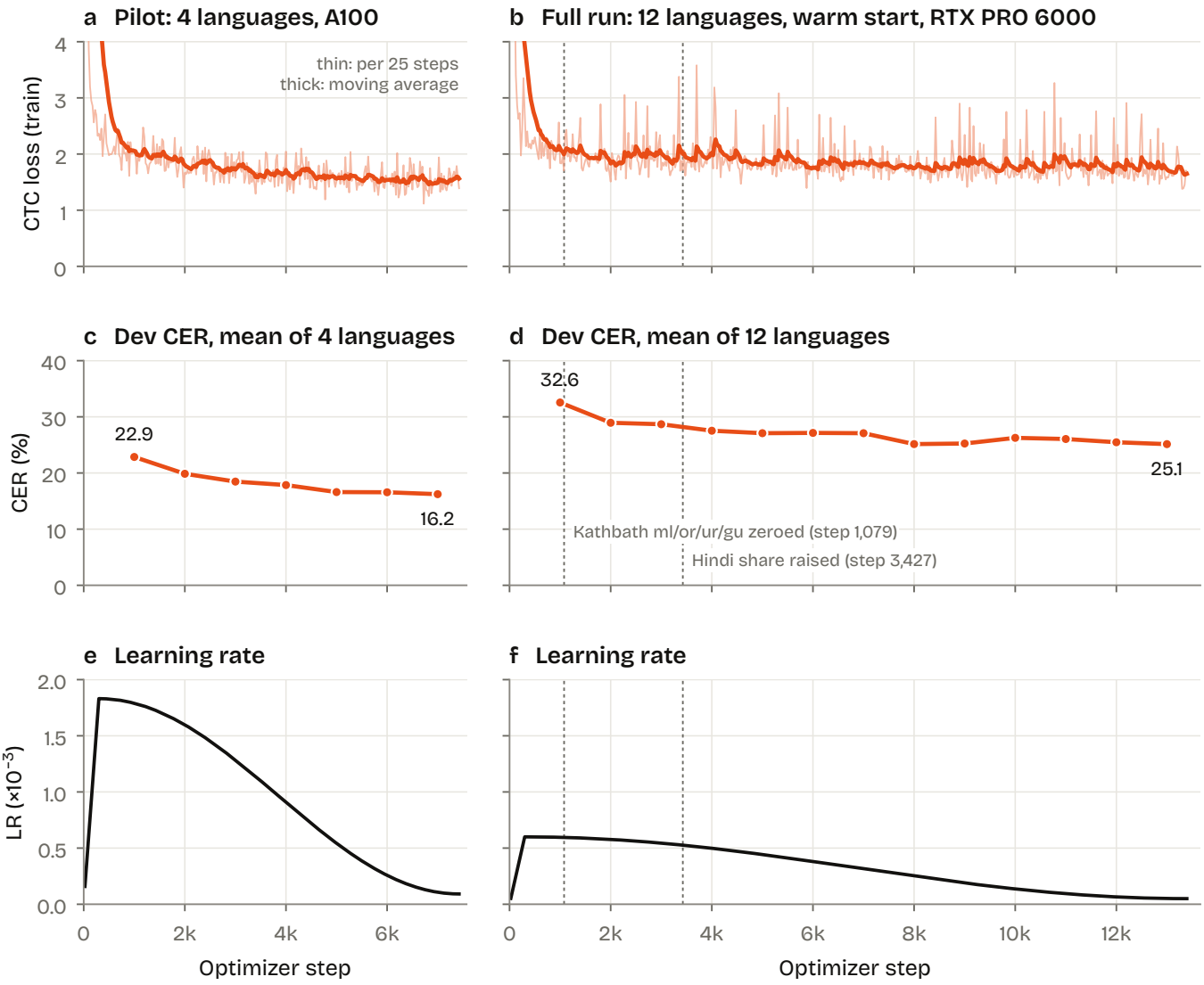


Fig. 10. Training curves. (a, b) CTC training loss (thin: per 25-step log; thick: moving average). (c, d) Macro development CER every 1,000 steps; the pilot's development set covers four languages and the full run's twelve, so the two are not comparable. (e, f) Learning rate. Dotted lines mark the two mid-run data changes of the full run.

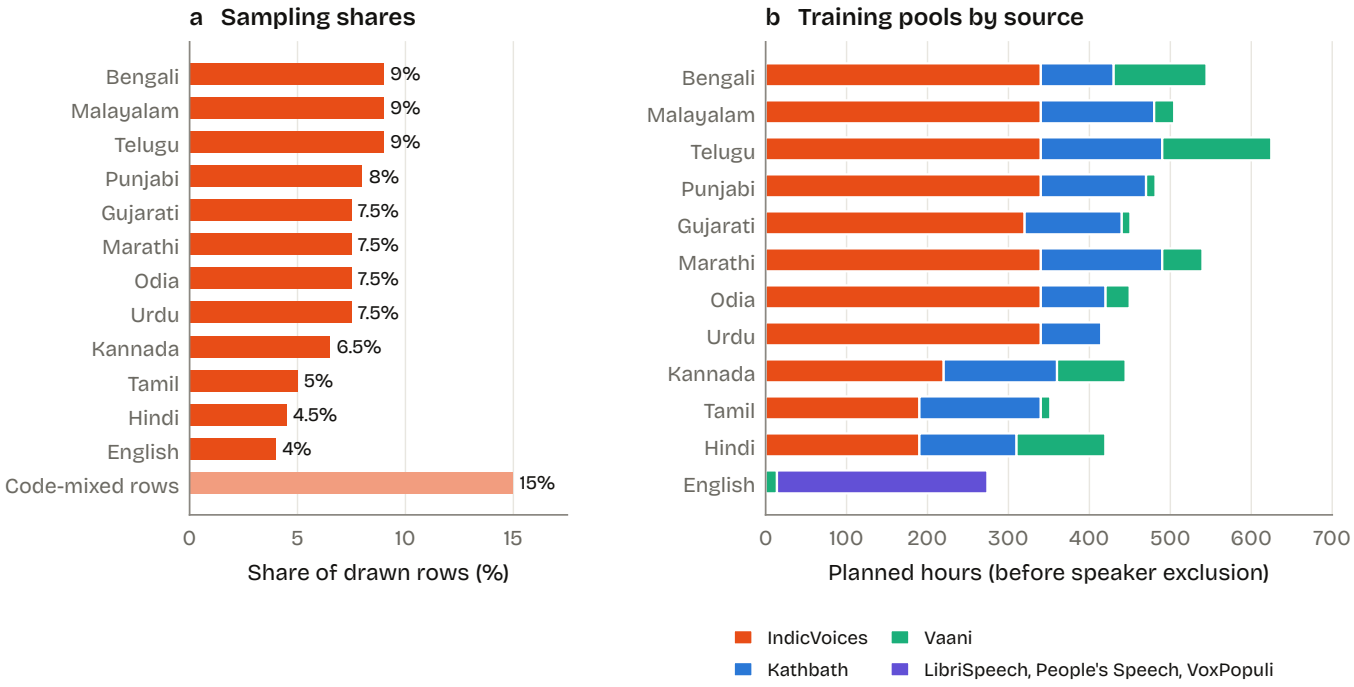


Fig. 11. Full-run mixture. (a) Pinned share of drawn rows per language; Hindi, Tamil and Kannada have a smaller share because the pilot already trained them. (b) Planned hours per language and source, before speaker exclusion.

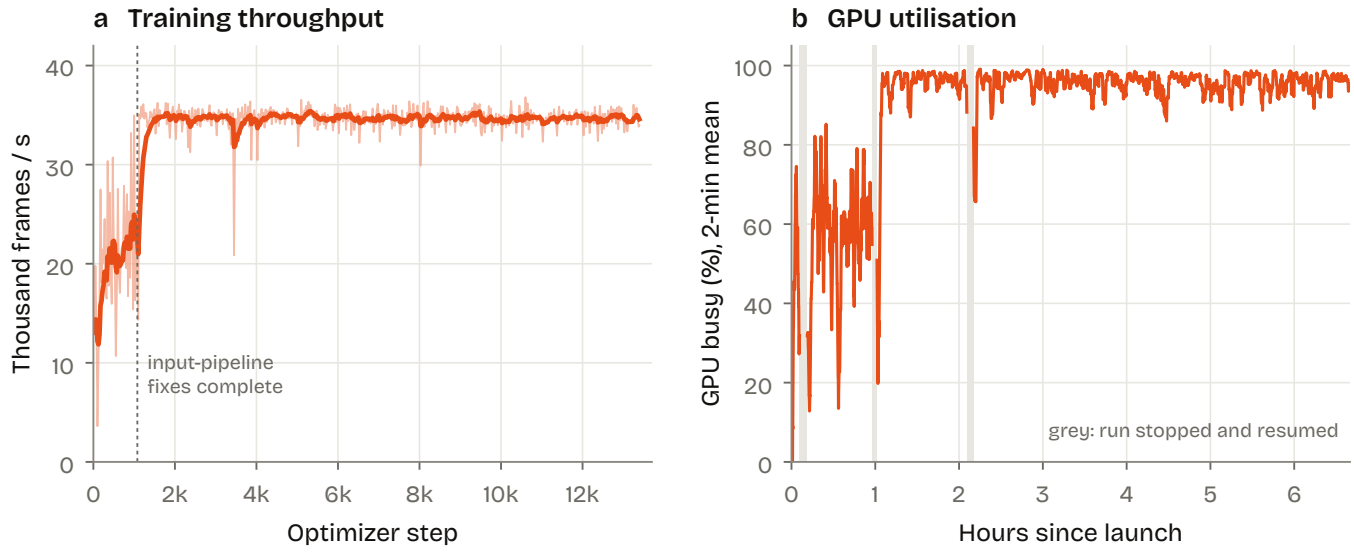


Fig. 12. (a) Full-run throughput in encoder frames per second, which rose from about 13,000 to 35,000 once the input-pipeline fixes landed. (b) GPU utilisation (2-minute mean); grey bands are stops for code deployment and resumes.

TABLE X

PROBLEMS MET DURING TRAINING, WITH CAUSE AND FIX. EACH COST TIME OR COMPUTE, AND MOST WOULD RECUR ON A NEW RUN.

When	Symptom	Cause	Fix
Pilot	cuFFT out of memory in GPU augmentation	band-limiting by large FFTs	power-of-two FIR convolution; retry on cuFFT memory errors
Pilot	compute balance ran out; machine and disk lost	billing lapse	checkpoints were already synced; the tokenizer was not, and was rebuilt and verified; the sync now includes it
Pilot	model reported ahead of IndicConformer	strict CER counted native-script English as errors	script-normalised CER (Section V)
Full, prep	load average 620 on 48 vCPUs	each ffmpeg call spawned 66 threads	one thread per ffmpeg call
Full, prep	a killed preparation stage was marked done	graceful SIGTERM exits with status 0	a stage is done only if its report is newer than its start
Full, prep	text pass took about 2 h	one serial pass over every stream	parallel text pass, sampled shards for the tokenizer: 13 min
Full, step 0	loss stuck at about 29.7	new-script bias -40; Adam moves a bias by about the learning rate per step	bias -2
Full, step 0	GPU idle	one thread read rows for all 53 streams at $\approx$266$\times$ real time, $\approx$740$\times$ needed	per-stream read-ahead threads
Full, steps 0–99	GPU about 15% busy	each batch decoded on one thread	rows decoded and augmented in parallel
Full, step 99	resume took 4.5 min	streams restored one by one	parallel restore, eager read-ahead
Full, steps 99–1,079	GPU 60–90% busy; Hub rate-limit errors	speaker exclusion left Kathbath Malayalam with 6% of its rows; the stream looped and re-downloaded shards	stop drawing those streams; a stream that accepts no row in a full pass stops
Full, step 3,000	Hindi development CER 24.8$\rightarrow$44.5	Urdu drew 7.5% of rows, Hindi 4.5%	decode rule (c); Hindi rebalanced from step 3,427
Full, any step	risk of a non-finite step in bfloat16	no guard without gradient scaling	a non-finite gradient norm skips the step

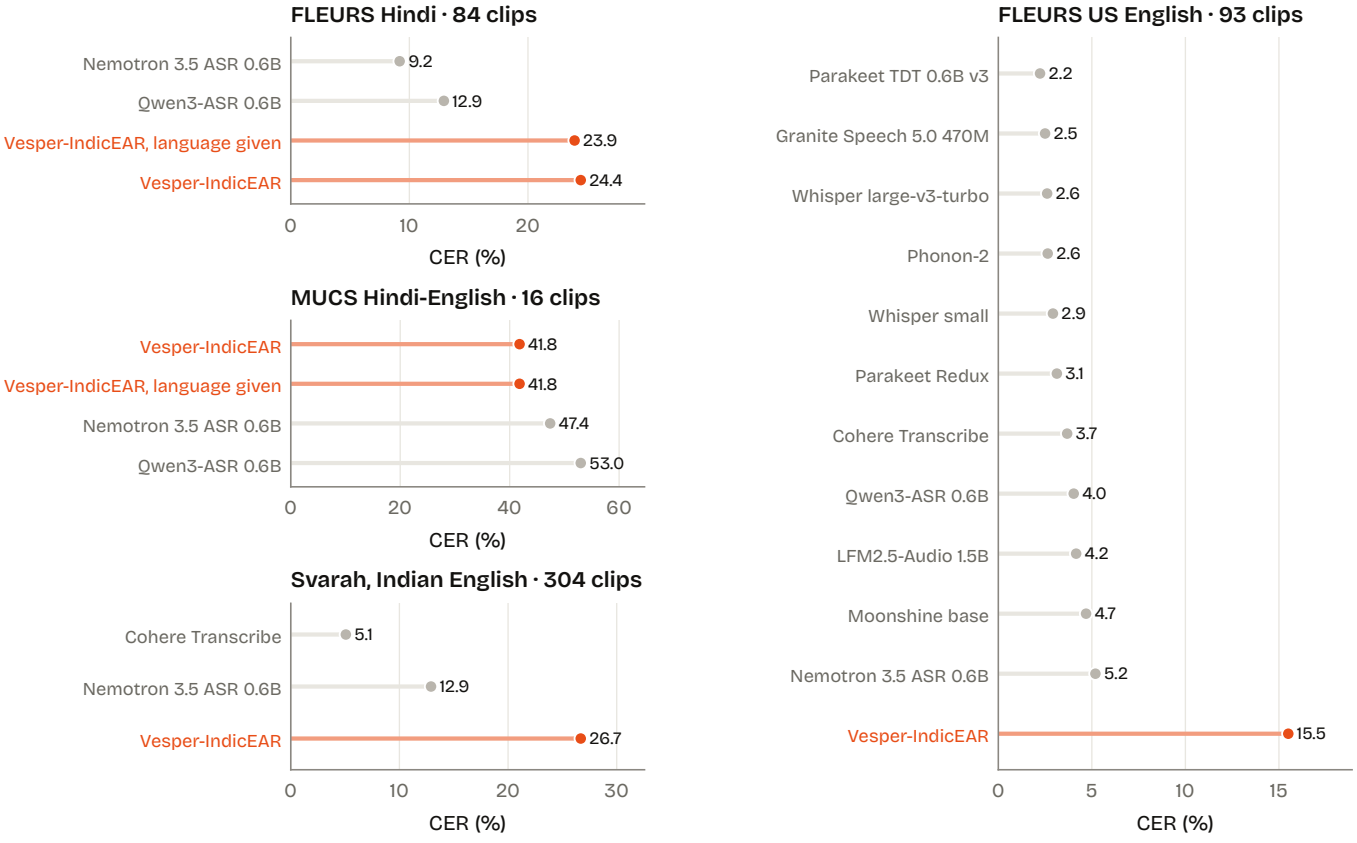


Fig. 13. CER on the Vesper ASR benchmark clips shared with our languages. Vesper-IndicEAR in orange; other models in grey, each in the runtime and precision recorded by the benchmark.