

# In-Place Indic Tokenizer Expansion and Continual Pretraining of Nemotron-3-Nano-30B-A3B

Avinash Benki\* Prasanjith Pasunuti\* Sutanshu Saksena Raj\*  
 Parijat Chatterjee Nayanjit Sarkar Daksh Maan  
 Gnani.AI  
 {avinash, prasanjith.p, sutanshu.raj} @ gnani.ai

## Abstract

We adapted Nemotron-3-Nano-30B-A3B for eight Indian languages without changing its backbone architecture. We expanded the vocabulary in place from 131,072 to 163,072 entries using continued byte-pair encoding and AdaptMix rebalancing, reducing tokens per word by 38.6%. We initialized new input and output rows by averaging their original constituent rows, then trained them on 90% Indic and 10% English text before updating the backbone. Continual pretraining then progressed through broad training, high-quality annealing, and long-context training over approximately 1.27 trillion tokens. The annealed checkpoint gained 12.9 points on the Indic MILU benchmark but lost 3.4 on English MMLU. We also report practical negative results. Direct token addition underperformed continued byte-pair encoding. AdamW weight decay changed rows intended to stay fixed. Perplexity on an Indic-heavy holdout did not select the learning rate that best preserved English performance.

## 1 Introduction

Tokenizers can split words in Indian languages into many small pieces, increasing the number of tokens needed to represent a document. We investigate whether expanding an existing tokenizer, followed by continual pretraining (CPT), can improve tokenization efficiency and benchmark performance across eight Indian languages.

We use Nemotron-3-Nano-30B-A3B Base BF16 [2], which has 31.75B parameters and activates 3.31B per token. We retain its hybrid backbone of Mamba-2 [6], mixture-of-experts, and attention layers. Adding 32,000 vocabulary entries increases the untied input and output matrices by  $2 \times 32,000 \times 2,688 = 172,032,000$  parameters. Table 9 gives the architecture details.

**Relation to prior work.** We follow established methods for continued byte-pair encoding (BPE), initialization from existing token embeddings, embedding-only warmup, and subsequent CPT [3]. We use AdaptMix-style rebalancing to adjust each language’s share of the tokenizer training data [4]. Our contribution is an account of applying these methods to eight Indian languages on a hybrid model, including the implementation problems and measured trade-offs.

### 1.1 Contributions

1. We document tokenizer expansion and embedding warmup for eight Indian languages, including tokenization comparisons and checks that keep inherited vocabulary rows fixed (Sections 2 and 3).

---

\*Equal contribution.

2. We report continual pretraining over approximately 1.27T tokens across broad training, high-quality annealing, and long-context training, with evaluations of Indic gains, English retention, and long-context performance (Sections 4 and 5).

## 2 Tokenizer expansion

We first expanded the vocabulary from 131,072 to 163,072 entries using continued BPE [3]. This preserves existing token identities and adds merges for Hindi, Bengali, Gujarati, Kannada, Malayalam, Marathi, Tamil, and Telugu. The construction corpus also includes romanized text.

For a corpus  $\mathcal{D}$ , tokenizer fertility is  $F(\mathcal{D}) = \sum_{x \in \mathcal{D}} N_{\text{tok}}(x) / \sum_{x \in \mathcal{D}} N_{\text{word}}(x)$ , where  $N_{\text{tok}}(x)$  and  $N_{\text{word}}(x)$  count tokens and whitespace-delimited words in text  $x$ , respectively. Lower fertility means less fragmentation. Native-script comparisons use 2,000 Samanantar [7] samples per language with seed 42. Fertility is computed separately for each of the eight target languages, and every mean we report across them is unweighted.

**Expansion method.** In an initial comparison, continued BPE gave lower fertility than direct token addition with `add_tokens` in all eight languages. Mean fertility was 2.17 for continued BPE and 2.48 for direct addition. We used continued BPE in all later experiments.

**Normalization and rebalancing.** Normalization variants produced similar fertility scores. We selected NFKC with newline cleanup and Indic-aware preprocessing, although the boundary-matching check did not provide a controlled comparison of these variants. We then ran nine rounds of AdaptMix-style rebalancing [4], increasing the data share of languages with higher fertility. The loop converged by round five. Appendix A gives the normalization results, quota updates, and final tokenizer preparation.

**Final tokenizer.** The selected tokenizer, TOK-2, reduced mean fertility from 3.55 to 2.18 tokens per word, a 38.6% reduction. Its score differs slightly from the initial continued-BPE result because it uses the selected normalization and rebalanced data. Gujarati improved most at 47.1%, followed by Malayalam at 44.3% (Table 10, Figure 2). These are tokenization results, not measured latency gains. The larger output vocabulary also changes per-token compute.

**Romanized text.** Despite its inclusion during construction, romanized text remained more fragmented than native-script text. On paired transliterated samples, the mean romanized-to-native fertility ratio was 1.84. Tamil had the largest ratio at 2.58, with 5.96 tokens per romanized word compared with 2.31 per native-script word.

## 3 Embedding warmup

We initialized and trained only the new input and output rows. Each new token  $t$  was decomposed into a sequence of original subtokens  $S(t)$ . We used constituent-mean initialization,

$$e_{\text{new}}(t) = \frac{1}{|S(t)|} \sum_{s \in S(t)} e_{\text{old}}(s),$$

where  $e_{\text{old}}(s)$  is the original row for subtoken  $s$ . We computed the mean separately for the input and output matrices. This follows source-subtoken averaging [3], rather than the weighted-overlap method used by FOCUS [1]. The final warmup used approximately 3B tokens, with 90% Indic and 10% English text, while keeping the backbone and inherited vocabulary rows fixed.

**Fixing the frozen rows.** Inherited rows changed during an early warmup run despite gradient masking. We traced the drift to AdamW [8] weight decay, whose decay-only update is

$$\theta_{t+1} = \theta_t - \eta \lambda \theta_t,$$

where  $\eta$  is the learning rate and  $\lambda$  is the weight-decay coefficient. This update changes inherited rows even when their gradients are masked to zero. Their relative drift reached approximately 4% over 8,000 iterations. Training loss appeared normal, but English scores fell to near-random levels. We fixed this by saving the inherited rows before each optimizer step and restoring them afterwards. After each step, we verified bit-exact equality on every tensor-parallel rank.

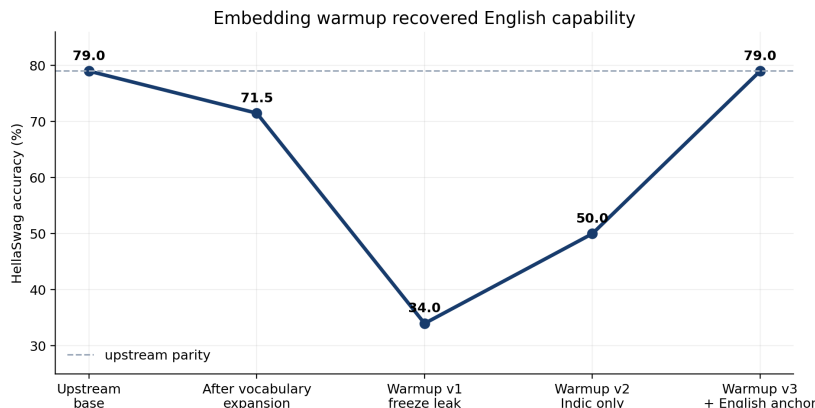


Figure 1: English scores during initialization and embedding warmup, shown in run order. These runs form a debugging sequence, not a controlled ablation. These scores are not directly comparable with the CPT results in Table 1.

**Including English data.** English scores recovered only partially after Indic-only warmup. A later run that included 10% English text restored parity on the warmup evaluation (Figure 1). These runs were part of a sequential debugging process, not a controlled ablation. Without a matched comparison under the corrected freeze, we cannot isolate the effect of the English data.

## 4 Continual pretraining

After embedding warmup, we continued pretraining with all model parameters trainable. Training had three stages: broad training at 8K context, high-quality annealing at 8K, and long-context training at 128K (Figure 5). Together, they consumed approximately 1.27T tokens on 256 H200 GPUs.

**Data preparation.** We combined English code, mathematics, reasoning, and high-quality web text with Indic web, academic, reference, and translated material from a pool of approximately 691B Indic tokens. Processing used exact and then inter-document fuzzy deduplication [5]. Indic documents received quality scores and high, medium, or low tiers based on length, script purity, repetition, boilerplate, KenLM perplexity [9], and safety signals. Synthetic translations and rewrites were down-weighted during broad training and excluded from high-quality annealing.

**Learning-rate selection.** We compared four peak learning rates from  $10^{-4}$  to  $5 \times 10^{-6}$ , training each run for 1B tokens. The lowest validation perplexity came from  $10^{-4}$ , while  $10^{-5}$  best preserved English benchmark performance. We selected  $10^{-5}$  with a Warmup-Stable-Decay schedule, a global batch size of 2,048, and a sequence length of 8,192 (Figure 4). The validation set was Indic-heavy, so its aggregate perplexity did not capture the English retention trade-off.

**Broad training and annealing.** Broad training consumed approximately 974.5B tokens at sequence length 8,192. During mixture planning, a separate 120B-token ablation with 55% English and 45% Indic data showed that gains varied across languages. Kannada gained 14.4 MILU points, Tamil 12.4, and Gujarati 12.2, while Hindi and English declined slightly. Increasing the Indic share to 60% did not improve aggregate Indic performance. These results informed later allocation alongside source quality and repetition constraints (Appendix D, Figure 6). High-quality annealing then continued from the selected broad checkpoint for 194.9B tokens at the same context length.

**Long-context training.** The final stage used approximately 99.86B tokens of direct long-document training at sequence length 131,072. We evaluated this checkpoint on RULER [10] and question-answering tasks separately from the broad and annealed checkpoints. Model FLOP utilization was 24.95% for broad training, 25.87% for annealing, and 27.15% at long context. Appendices H and J record the compute costs and the training traces for all three stages.

Table 1: Results under the same evaluation setup. Both adapted checkpoints precede long-context training. Appendix F lists the full benchmark results for these checkpoints.

| Benchmark                               | Base   | Broad  | Anneal |
|-----------------------------------------|--------|--------|--------|
| MILU [12]                               | 40.63  | 53.54  | 53.52  |
| ARC-Challenge-Indic [30]                | 35.23  | 48.20  | 48.20  |
| IndicGLUE-selected [14]                 | 52.12  | 57.86  | 57.22  |
| IndicMMLU-Pro [15]                      | 14.44  | 20.03  | 20.02  |
| BharatMath                              | 29.99  | 41.06  | 39.70  |
| Indic BPB overall ( $\downarrow$ ) [13] | 0.8453 | 0.6189 | 0.6132 |
| MMLU [16]                               | 78.50  | 75.20  | 75.11  |
| MMLU-Pro [17]                           | 64.83  | 54.78  | 55.43  |
| HellaSwag [19]                          | 85.57  | 83.52  | 83.69  |

## 5 Results and English retention

Table 1 compares all three checkpoints using the same evaluation setup.

Relative to the base model, the annealed checkpoint gained 12.9 points on MILU and 13.0 on ARC-Challenge-Indic. Indic bits per byte fell from 0.8453 to 0.6132. MMLU declined by 3.4 points and MMLU-Pro by 9.4. Annealing recovered part of the English cost of broad training: the nine-benchmark English mean rose from 67.98 to 68.40, with MMLU-Pro regaining 0.7 points while MMLU held flat.

We selected the annealed checkpoint for downstream post-training because it maintained the main Indic gains, further reduced Indic bits per byte, and recovered part of the English retention cost. A different balance between Indic and English performance could still favor the broad checkpoint. The warmup scores in Figure 1 use a different setup and cannot be compared directly with the CPT evaluations.

**Long-context evaluation.** The long-context checkpoint achieved mean RULER recall of 76.7% across 55 evaluation slices. Mean exact match was 45.7% across 10 question-answering slices. Recall decreased from 96.6% at 8,192 tokens to 46.9% at 131,072 tokens. Appendix G gives recall at each of the five evaluated context lengths.

A companion HELMET [11] evaluation shows that long-context performance varies by task. In-context learning improves from 59.2 at 8K to 81.7 at 128K and is the only evaluated category whose 128K score exceeds its 8K score. Retrieval-augmented generation and long-form question answering decline to 37.2 and 26.8 at 128K, respectively; long-form question answering peaks at 40.6 at 64K. Exact retrieval deteriorates markedly beyond 32K, falling from 95.9 at 8K to 14.0 at 128K.

## 6 Limitations and conclusion

We did not run a matched CPT control with the original tokenizer, so we cannot separate the contribution of tokenizer expansion from additional training. The warmup sequence was not a controlled ablation. The pretraining corpus includes twelve Indic languages, but only eight received dedicated vocabulary expansion. Odia, Punjabi, Urdu, and Nepali received a small share of the corpus without expansion. Romanized text remained more fragmented, and we did not measure end-to-end latency. The long-context results report one checkpoint without a matched base-model comparison.

Tokenizer expansion and CPT improved native-script tokenization efficiency and Indic benchmark performance, but reduced accuracy on several English benchmarks. Our experience also shows why embedding warmup needs explicit checks that inherited rows remain unchanged, and why model selection should measure English retention alongside Indic performance.

**Model release.** The downstream model is available at <https://huggingface.co/gnani/gnani-evon-v3.3-30B-A3B>. It includes supervised fine-tuning and reinforcement learning beyond the CPT stages evaluated here.

## References

- [1] K. Dobler and G. de Melo. FOCUS: Effective Embedding Initialization for Monolingual Specialization of Multilingual Models. In *Proceedings of EMNLP 2023*, pages 13440–13454, 2023. arXiv:2305.14481.
- [2] NVIDIA. Nemotron 3 Nano: Open, Efficient Mixture-of-Experts Hybrid Mamba-Transformer Model for Agentic Reasoning. arXiv:2512.20848, 2025.
- [3] J. T. H. Smith et al. In-Place Tokenizer Expansion for Pre-trained LLMs. arXiv:2607.15232, 2026.
- [4] A. Thakur et al. The Art of Breaking Words: Rethinking Multilingual Tokenizer Design. arXiv:2508.06533, 2025.
- [5] J. Li et al. DataComp-LM: In Search of the Next Generation of Training Sets for Language Models. arXiv:2406.11794, 2024.
- [6] T. Dao and A. Gu. Transformers are SSMS: Generalized Models and Efficient Algorithms Through Structured State Space Duality. arXiv:2405.21060, 2024.
- [7] G. Ramesh et al. Samanantar: The Largest Publicly Available Parallel Corpora Collection for 11 Indic Languages. *Transactions of the Association for Computational Linguistics*, 10:145–162, 2022. arXiv:2104.05596.
- [8] I. Loshchilov and F. Hutter. Decoupled Weight Decay Regularization. In *ICLR*, 2019. arXiv:1711.05101.
- [9] K. Heafield. KenLM: Faster and Smaller Language Model Queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, 2011. ACL Anthology: W11-2123.
- [10] C.-P. Hsieh et al. RULER: What’s the Real Context Size of Your Long-Context Language Models? In *COLM*, 2024. arXiv:2404.06654.
- [11] H. Yen et al. HELMET: How to Evaluate Long-Context Language Models Effectively and Thoroughly. In *ICLR*, 2025. arXiv:2410.02694.
- [12] S. Verma et al. MILU: A Multi-task Indic Language Understanding Benchmark. arXiv:2411.02538, 2024.
- [13] H. Singh et al. IndicGenBench: A Multilingual Benchmark to Evaluate Generation Capabilities of LLMs on Indic Languages. In *Proceedings of ACL*, pages 11047–11073, 2024. arXiv:2404.16816.
- [14] D. Kakwani et al. IndicNLP Suite: Monolingual Corpora, Evaluation Benchmarks and Pre-trained Multilingual Language Models for Indian Languages. In *Findings of EMNLP*, pages 4948–4961, 2020. ACL Anthology: 2020.findings-emnlp.445.
- [15] Sankalp KJ et al. IndicMMLU-Pro: Benchmarking Indic Large Language Models on Multi-Task Language Understanding. arXiv:2501.15747, 2025.
- [16] D. Hendrycks et al. Measuring Massive Multitask Language Understanding. In *ICLR*, 2021. arXiv:2009.03300.
- [17] Y. Wang et al. MMLU-Pro: A More Robust and Challenging Multi-Task Language Understanding Benchmark. arXiv:2406.01574, 2024.
- [18] P. Clark et al. Think You Have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge. arXiv:1803.05457, 2018.
- [19] R. Zellers et al. HellaSwag: Can a Machine Really Finish Your Sentence? arXiv:1905.07830, 2019.
- [20] L. Bandarkar et al. The Belebele Benchmark: A Parallel Reading Comprehension Dataset in 122 Language Variants. In *Proceedings of ACL*, pages 749–775, 2024. arXiv:2308.16884.

- [21] S. Singh et al. Global MMLU: Understanding and Addressing Cultural and Linguistic Biases in Multilingual Evaluation. arXiv:2412.03304, 2024.
- [22] A. Maheshwari et al. IndicParam: Benchmark to Evaluate LLMs on Low-resource Indic Languages. arXiv:2512.00333, 2025.
- [23] A. Maji et al. SANSKRITI: A Comprehensive Benchmark for Evaluating Language Models’ Knowledge of Indian Culture. In *Findings of ACL*, pages 4434–4451, 2025. arXiv:2506.15355.
- [24] K. Cobbe et al. Training Verifiers to Solve Math Word Problems. arXiv:2110.14168, 2021.
- [25] T. Mihaylov et al. Can a Suit of Armor Conduct Electricity? A New Dataset for Open Book Question Answering. arXiv:1809.02789, 2018.
- [26] Y. Bisk et al. PIQA: Reasoning about Physical Commonsense in Natural Language. arXiv:1911.11641, 2019.
- [27] G. Lai et al. RACE: Large-scale ReAding Comprehension Dataset From Examinations. arXiv:1704.04683, 2017.
- [28] K. Sakaguchi et al. WinoGrande: An Adversarial Winograd Schema Challenge at Scale. arXiv:1907.10641, 2019.
- [29] M. Chen et al. Evaluating Large Language Models Trained on Code. arXiv:2107.03374, 2021.
- [30] Sarvam AI. ARC-Challenge-Indic. Hugging Face dataset, n.d.  
<https://huggingface.co/datasets/sarvamai/arc-challenge-indic>. Accessed September 8, 2026.

## A Tokenizer experiments

Table 2 summarizes the tokenizer comparisons recorded in the report compiled on July 13, 2026. We first compared expansion methods, then normalization and language rebalancing. TOK-2 was selected for training. The larger TOK-3 tokenizer was tested later and was not used for CPT.

Table 2: Tokenizer fertility on Samanantar, using seed 42 and 2,000 samples per language.

| Run                     | Extension | Mean fertility | Notes                             |
|-------------------------|-----------|----------------|-----------------------------------|
| Base Nemotron           | None      | 3.5472         | Original tokenizer                |
| Direct add_tokens       | 32K       | 2.4815         | 14.3% above continued BPE         |
| Continued BPE           | 32K       | 2.1715         | Used in subsequent experiments    |
| Raw / NFKC / NFKC+regex | 32K       | 2.1716–2.1823  | Similar fertility across variants |
| TOK-2                   | 32K       | 2.1773         | Selected for CPT                  |
| TOK-3                   | 64K       | 2.0996         | Lowest mean, not used for CPT     |

**Normalization comparison.** We compared raw text, Unicode NFKC, and NFKC followed by newline cleanup and an Indic-aware regex pass on the same 7.58M-row mixture. Mean fertility was 2.1716 for raw text and 2.1823 for both normalized variants. Fertility did not distinguish the two normalized variants, and we selected the regex variant. A boundary-agreement check against the base pretokenizer was run only for the selected variant. Variants without a regex stage match the base pretokenizer by construction, so this check cannot rank the three variants. We therefore have no measured criterion that distinguishes them.

**Rebalancing rule.** We rebalanced language shares using these fertility-based updates:

$$\delta_l = \frac{f_l - f_{\text{best}}}{f_{\text{range}}}, \quad w_l = \max(\delta_l + \epsilon, \epsilon), \quad m_l = (1 - \mu)m_{l,\text{prev}} + \mu t_l.$$

Here  $f_l$  is the fertility of language  $l$ ,  $f_{\text{best}}$  is the lowest fertility, and  $f_{\text{range}}$  is the range across languages. The update moves the previous data share  $m_{l,\text{prev}}$  toward the target share  $t_l$ . We used  $\mu = 0.4$  and  $\epsilon = 0.001$  in all nine rebalancing rounds.

**Changes in language shares.** From initialization to round nine, Malayalam’s share rose from 12.5% to 18.7%, Kannada’s from 12.5% to 14.3%, and Telugu’s from 12.5% to 14.5%. Hindi’s share fell from 12.5% to 7.0% as more data was allocated to languages with higher fertility.

**Rebalancing result.** The nine-round loop converged by round five. Malayalam had the highest fertility before rebalancing. Its data share increased from 12.5% to 18.7%, and its fertility improved by 5.4% relative to the initial tokenizer.

**Final tokenizer preparation.** Of the 32,000 appended entries, 31,999 were byte-identical between the rebalanced artifact and the tokenizer staged for training. A space-plus-newline merge was made unreachable and its entry was later reused as an atomic separator token. This kept the vocabulary size and tensor alignment unchanged.

**Later 64K expansion.** The exploratory TOK-3 tokenizer reduced fertility relative to the original tokenizer by 27.2% for Hindi, 51.8% for Gujarati, 43.5% for Kannada, and 42.5% for Malayalam. Compared with TOK-2, its mean fertility was 3.6% lower, but Malayalam fertility was 3.1% higher (Figure 3). The CPT configuration was fixed before TOK-3 was developed, and all runs used TOK-2.

## B CPT configuration

Table 3: Configuration used for broad continual pretraining.

| Field                    | Value                                            |
|--------------------------|--------------------------------------------------|
| Initialization           | Checkpoint after warmup, vocabulary 163,072      |
| Precision                | BF16                                             |
| Sequence length          | 8,192 (broad and anneal), 131,072 (long context) |
| Global / micro batch     | 2,048 / 1                                        |
| Parallelism              | TP=1, EP=8, PP=1, remaining ranks use DP         |
| Optimizer                | Distributed AdamW                                |
| Peak / min learning rate | $1.0 \times 10^{-5}$ / $1.0 \times 10^{-6}$      |
| Schedule                 | Warmup-Stable-Decay with square-root decay       |
| Warmup iterations        | 200                                              |
| MoE dispatcher           | flex + HybridEP, capacity factor 1.25            |
| Kernels                  | Grouped GEMM, selective recompute, CUDA graphs   |
| Tracking                 | Weights & Biases                                 |

## C Launch configuration and validation

The CPT setup uses Nemotron-Bridge/Megatron, with training scripts, data-mixture manifests, and pipelines for building the caches used during model training.

**Full training setup.** Training used 32 nodes with eight H200 GPUs per node, for 256 GPUs in total. The parallelism setting was `REPLICA_MP = (1, 8, 1)`, with expert-parallel groups kept within each node. Dense data parallelism was 256 and expert data parallelism was 32. We used eight gradient-accumulation steps with a microbatch size of one.

**Mid-training data plan.** The mid-training manifest specifies 285B tokens over 16,987 iterations, with global batch size 2,048 and sequence length 8,192. Approximately 34% of tokens come from long-document sources. The remaining tokens cover shorter examples in reasoning, code, knowledge retention, and instruction data. Table 11 lists the planned rather than measured token allocation.

**Validation on 128 GPUs.** We tested the launch setup on 16 nodes using the checkpoint after embedding warmup and the tokenizer-construction data mixture described above. The run completed 100 iterations with a peak learning rate of  $10^{-5}$  and a Warmup-Stable-Decay stable fraction of 0.80. During iterations 50–100, step time was 11.4–11.6 s and compute throughput was 254–261 TFLOP/s/GPU. Language-model loss fell from 4.78 at iteration 10 to 4.05 during iterations 50–100. Gradient norm was approximately 0.9–1.6. The run had no skipped or NaN iterations. We saved the checkpoint after completing all 100 iterations.

**Full-run settings.** The 32-node configuration sets the vocabulary size to 163,072 and disables forced router load balancing. It also uses NCCL settings tuned for the InfiniBand interconnect.

## D English and Indic data-mixture ablation

Table 4: MILU scores before and after 120B tokens of CPT with 55% English and 45% Indic data.

| Language  | Base | 120B CPT | $\Delta$ |
|-----------|------|----------|----------|
| Kannada   | 32.8 | 47.2     | +14.4    |
| Tamil     | 34.2 | 46.6     | +12.4    |
| Gujarati  | 34.4 | 46.5     | +12.2    |
| Telugu    | 35.8 | 47.9     | +12.1    |
| Malayalam | 35.5 | 45.7     | +10.1    |
| Bengali   | 39.4 | 47.7     | +8.4     |
| Marathi   | 40.1 | 46.8     | +6.7     |
| Hindi     | 53.1 | 51.4     | -1.8     |
| English   | 65.6 | 63.9     | -1.7     |

The results informed later language allocation. The planned shares were 16% Telugu, 15% Tamil, 13% Kannada, 12% each for Gujarati, Malayalam, Bengali, and Hindi, and 8% Marathi. These are planning weights, not measured shares of the completed training run.

## E Long-context CPT mixture

Table 5: Composition of the 100B-token long-context data pool. The selected run consumed approximately 99.86B tokens from this pool.

| Bucket                       | Weight | Shards | Pool tokens |
|------------------------------|--------|--------|-------------|
| Indic long documents         | 0.22   | 9      | 22.0B       |
| Indic high-quality           | 0.15   | 132    | 15.0B       |
| English long documents       | 0.20   | 7      | 20.0B       |
| Mathematics and STEM         | 0.14   | 1      | 14.0B       |
| Long-context supervised data | 0.10   | 3      | 10.0B       |
| Code                         | 0.08   | 1      | 8.0B        |
| Base-retention data          | 0.07   | 2      | 7.0B        |
| Other multilingual           | 0.04   | 2      | 4.0B        |

## F Full checkpoint evaluation results

Table 6: Results for the base, broad, and annealed checkpoints. Two benchmarks are omitted because they lack results for one or more checkpoints.

| Benchmark                               | Base   | Broad  | Anneal |
|-----------------------------------------|--------|--------|--------|
| MILU [12]                               | 40.63  | 53.54  | 53.52  |
| ARC-Challenge-Indic [30]                | 35.23  | 48.20  | 48.20  |
| Belebele [20]                           | 37.49  | 42.17  | 42.27  |
| BharatMath                              | 29.99  | 41.06  | 39.70  |
| Global-MMLU-Lite [21]                   | 32.50  | 36.17  | 36.00  |
| IndicGLUE-selected [14]                 | 52.12  | 57.86  | 57.22  |
| Indic BPB overall ( $\downarrow$ ) [13] | 0.8453 | 0.6189 | 0.6132 |
| IndicMMLU-Pro [15]                      | 14.44  | 20.03  | 20.02  |
| IndicParam [22]                         | 28.08  | 31.59  | 31.55  |
| Sanskriti [23]                          | 84.15  | 80.60  | 80.82  |
| ARC-Challenge [18]                      | 67.66  | 67.49  | 67.92  |
| GSM8K [24]                              | 86.13  | 84.46  | 84.46  |
| HellaSwag [19]                          | 85.57  | 83.52  | 83.69  |
| MMLU [16]                               | 78.50  | 75.20  | 75.11  |
| MMLU-Pro [17]                           | 64.83  | 54.78  | 55.43  |
| OpenBookQA [25]                         | 46.60  | 43.60  | 44.80  |
| PIQA [26]                               | 84.44  | 83.03  | 83.46  |
| RACE-filtered [27]                      | 42.39  | 41.72  | 42.39  |
| WinoGrande [28]                         | 79.48  | 78.06  | 78.37  |

The broad checkpoint generated an unintended code separator during HumanEval [29]. Its HumanEval result therefore does not provide a reliable measure of code-generation retention.

## G Long-context checkpoint evaluation

We evaluated the long-context checkpoint on RULER [10] and question-answering (QA) tasks. A companion HELMET [11] evaluation measured additional long-context capabilities. Table 7 reports the mean recall over 55 RULER slices, the mean exact match over 10 question-answering slices, and recall at five context lengths. The percentages are scores for this checkpoint rather than improvements over a reference model.

Table 7: RULER recall and QA exact-match scores for the long-context checkpoint. These tasks differ from those used for the broad and annealed checkpoint comparisons.

| Metric or context length        | Score (%) |
|---------------------------------|-----------|
| Mean RULER recall (55 slices)   | 76.7      |
| Mean QA exact match (10 slices) | 45.7      |
| Recall at 8,192 tokens          | 96.6      |
| Recall at 16,384 tokens         | 90.6      |
| Recall at 32,768 tokens         | 82.4      |
| Recall at 65,536 tokens         | 67.3      |
| Recall at 131,072 tokens        | 46.9      |

HELMET’s Recall category scores are 95.9, 81.4, 57.0, 37.3, and 14.0 at the same five context lengths. The gap relative to RULER grows from 0.7 points at 8K to 32.9 points at 128K. The evaluations use different task subsets and aggregation schemes, so their recall scores should not be compared directly.

## H CPT compute usage

Table 8: GPU-hours for CPT, excluding embedding warmup and ablations.

| Stage                                   | GPU-hours     |
|-----------------------------------------|---------------|
| CPT broad ( $\sim 974\text{B}$ )        | $\sim 25,147$ |
| CPT annealing (195B)                    | $\sim 4,988$  |
| CPT long context ( $\sim 100\text{B}$ ) | 4,977         |
| <b>CPT total</b>                        | $\sim 35,112$ |

Broad training and annealing ran at a median 5.85 s per step on 256 GPUs, processing approximately 2.87M tokens per second. At sequence length 131,072, the median step time was 159.89 s.

## I Supporting tables and figures

This section provides the architecture summary, per-language fertility results, planned mid-training mixture, and additional plots for the tokenizer and CPT experiments.

Table 9: Nemotron-3-Nano-30B-A3B architecture and expanded vocabulary. Parameter counts refer to the base model. Vocabulary expansion adds approximately 172 million parameters.

| Field                                  | Value                          |
|----------------------------------------|--------------------------------|
| Total / active parameters              | 31.75B / 3.31B                 |
| Layers                                 | 52 (23 Mamba-2, 23 MoE, 6 GQA) |
| Model dim / heads / KV heads           | 2,688 / 32 / 2                 |
| MoE experts (routed / active / shared) | 128 / 6 / 1                    |
| Vocabulary / max sequence              | 163,072 / 131,072              |
| Position encoding                      | none (NoPE)                    |

Table 10: Fertility of the original tokenizer and TOK-2 (163,072 entries).

| Language  | Upstream | Expanded | Change |
|-----------|----------|----------|--------|
| Hindi     | 2.05     | 1.66     | −19.2% |
| Bengali   | 3.06     | 2.14     | −30.0% |
| Gujarati  | 3.61     | 1.91     | −47.1% |
| Kannada   | 3.86     | 2.34     | −39.5% |
| Malayalam | 4.97     | 2.77     | −44.3% |
| Marathi   | 3.10     | 1.94     | −37.4% |
| Tamil     | 3.77     | 2.29     | −39.1% |
| Telugu    | 3.96     | 2.37     | −40.1% |
| Mean      | 3.55     | 2.18     | −38.6% |

Table 11: Planned data allocation for the 285B mid-training mixture.

| Bucket                 | Weight | Target tokens |
|------------------------|--------|---------------|
| Reasoning              | 22%    | 62.7B         |
| Indic long documents   | 15%    | 42.8B         |
| Math & STEM            | 14%    | 39.9B         |
| Knowledge retention    | 12%    | 34.2B         |
| English long documents | 11%    | 31.4B         |
| Code                   | 9%     | 25.7B         |
| Instruction            | 7%     | 20.0B         |
| Indic high-quality     | 6%     | 17.1B         |
| Web / multilingual     | 4%     | 11.4B         |

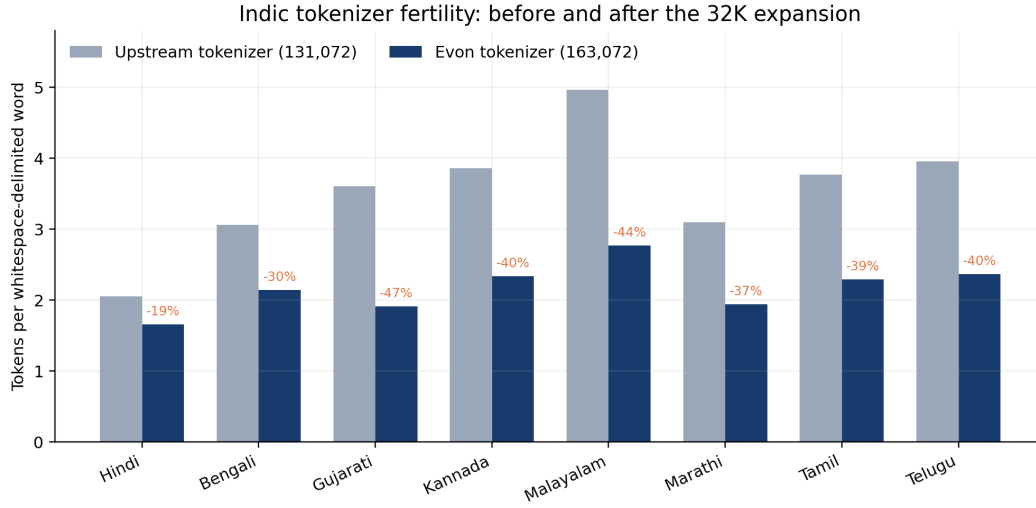


Figure 2: Tokens per word before and after the 32K expansion. Lower is better.

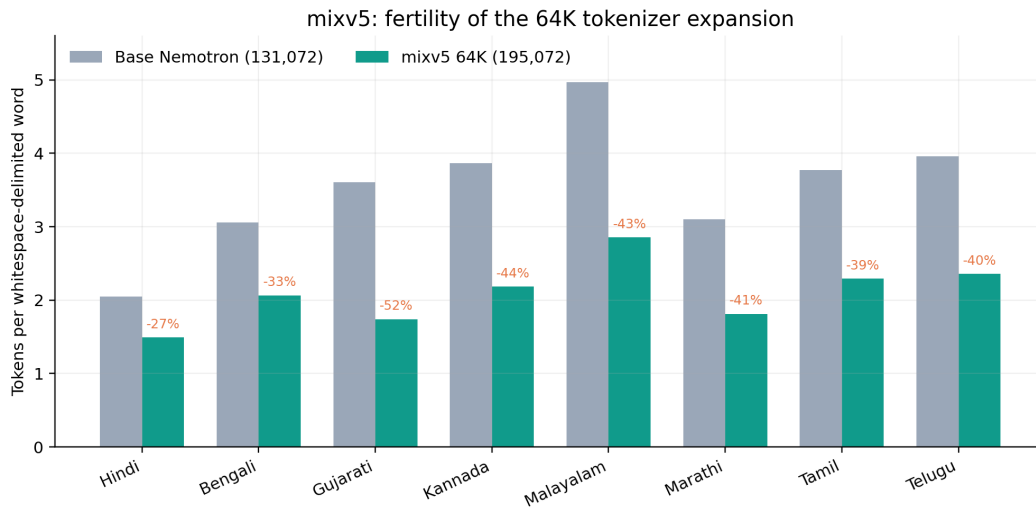


Figure 3: Fertility of the original tokenizer and the exploratory TOK-3 tokenizer with 64,000 added entries. TOK-3 was not used in any CPT run.

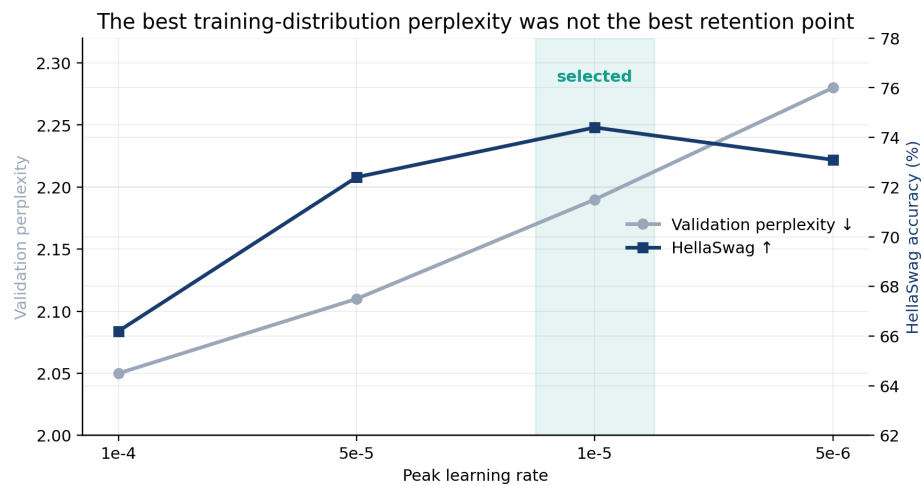


Figure 4: Learning-rate comparison. Validation perplexity and English benchmark performance favored different learning rates in the four-run sweep.

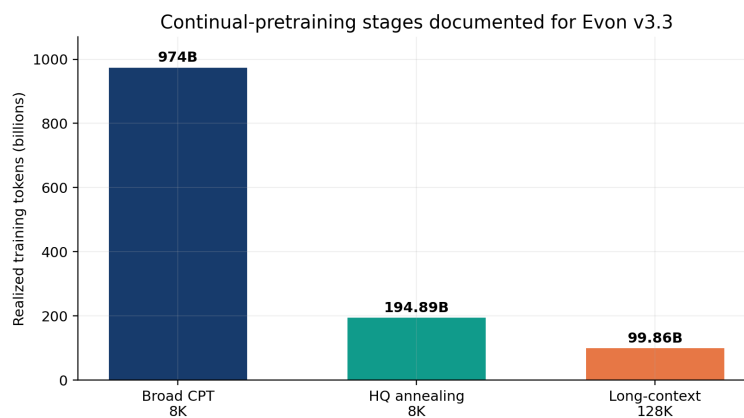


Figure 5: Tokens consumed during broad training, high-quality annealing, and long-context training.

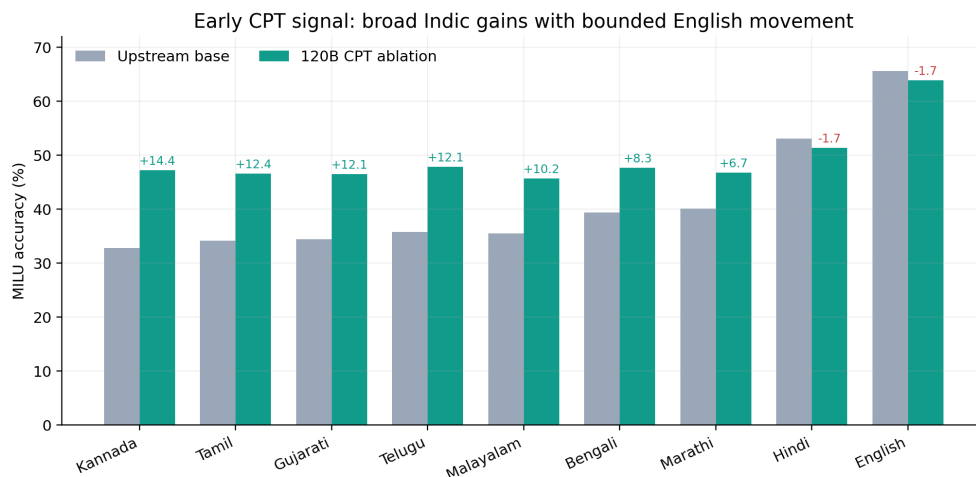


Figure 6: MILU accuracy before and after the 120B-token ablation with 55% English and 45% Indic data. This run informed mixture planning and was not released as a checkpoint.

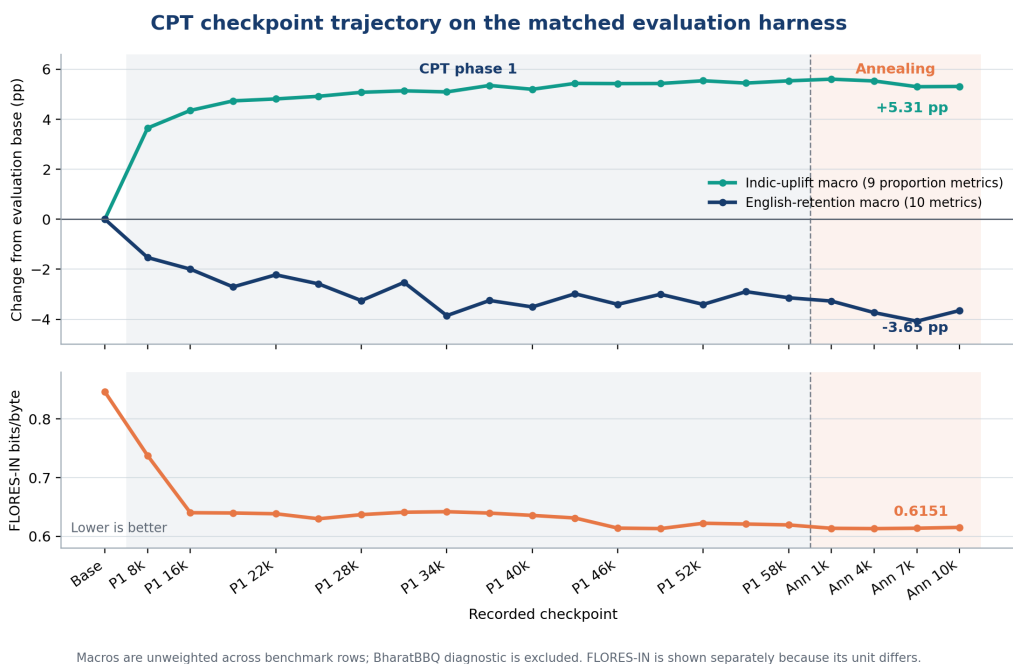


Figure 7: Indic and English benchmark results across CPT checkpoints under the same evaluation setup. Lower FLORES-IN bits per byte indicate better performance on that metric.

## J CPT training traces

The following plots were exported from the recorded CPT runs and are reproduced from the technical report. Each row shows MoE sequence load-balancing loss, gradient norm, learning rate, and validation language-model loss, from left to right. The load-balancing loss is an auxiliary objective and is separate from the language-model loss.

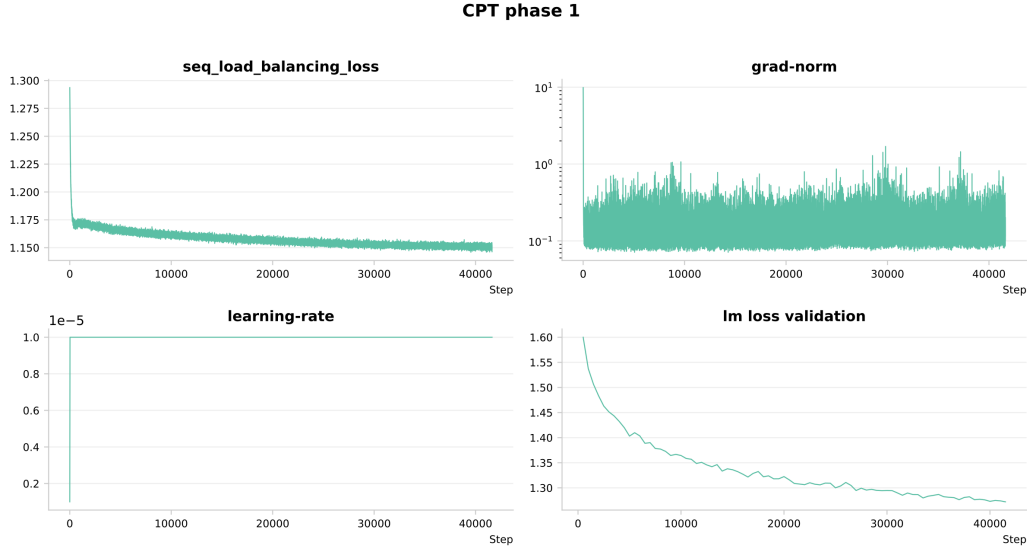


Figure 8: Weights & Biases traces for broad training at 8K context. These plots end before the broad checkpoint reported in the benchmark evaluation.

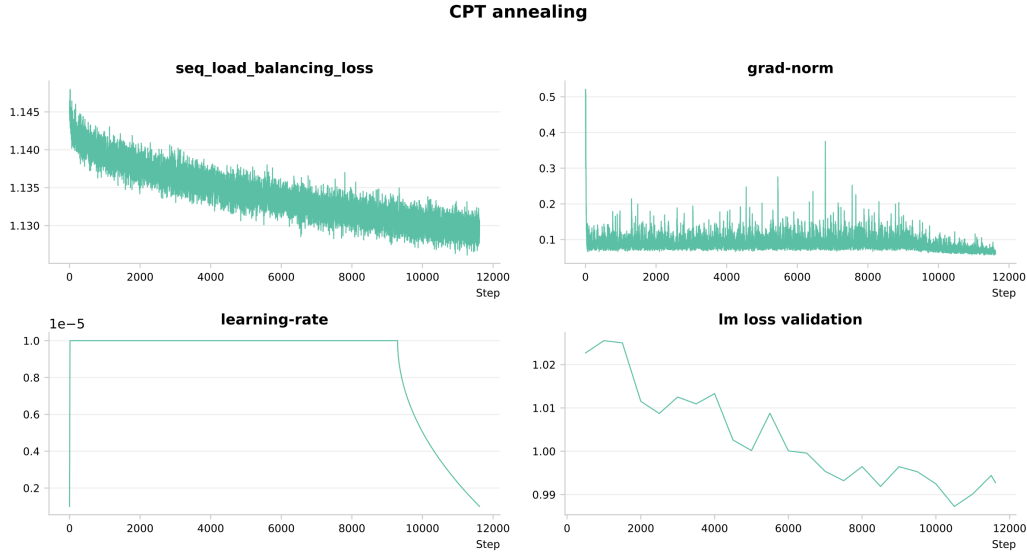


Figure 9: Weights & Biases traces for high-quality annealing. Learning-rate decay begins late in this stage. Validation loss fluctuates around 1.0 over the recorded interval.

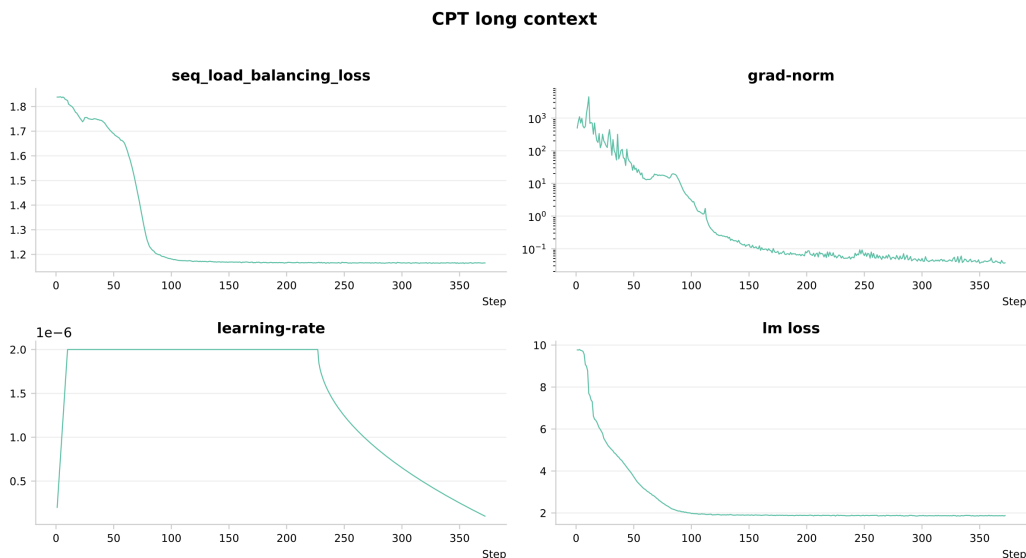


Figure 10: Weights & Biases traces for long-context training at sequence length 131,072. Gradient norm and language-model loss fall early in training and then stabilize. Sequence load-balancing loss settles near 1.17 during the recorded run.

## K Broader impacts

The expanded tokenizer represents native-script text in eight Indian languages with fewer tokens. This allows more text to fit within a fixed token budget, although we have not measured end-to-end latency or serving costs. The public model weights allow other groups to build on the adaptation without repeating the full training process.

The adaptation reduced accuracy on several English benchmarks (Section 5). Applications that require strong English performance should consider these losses when selecting a checkpoint. Romanized text remained more fragmented than native-script text, so users typing Indic languages in Latin script may not receive the same tokenization benefits.

The model could also be misused to generate misinformation in the supported languages. This is a concern where detection and moderation tools are limited. Odia, Punjabi, Urdu, and Nepali received a small share of the pretraining corpus but no dedicated vocabulary expansion. Benefits for languages outside the eight expanded here remain untested.

## L Reproducibility

Repeating the full adaptation requires substantial compute. Our CPT runs used 256 H200 GPUs. The reusable components include the tokenizer and its construction procedure, initialization from existing token rows, checks that inherited rows remain unchanged, data-mixture specifications, and evaluation settings. Tokenizer and row-freezing tests do not require a full CPT run.