



A Sovereign, Indic-First, Reasoning Model

Tokenizer expansion, continual pretraining, supervised fine-tuning, and reinforcement learning.

Core Team: Sutanshu Saksena Raj, Prasanjith Pasunuti, Parijat Chatterjee, Daksh Maan, Purusharth Malik, Nayanjit Sarkar

Technical Lead: Avinash Benki

Gnani.AI

Release family Released RL checkpoint

License Apache License 2.0

Report status Technical Report

Model artifacts: <https://huggingface.co/gnani/gnani-evon-v3.3-30B-A3B>

Abstract

Gnani-Evon-v3.3-30B-A3B is a long-context hybrid Mamba–Transformer mixture-of-experts model for reasoning, instruction following, and tool use in English and eight Indian languages. Starting from NVIDIA’s Nemotron 3 Nano 30B-A3B checkpoint, it retains the 52-layer backbone and changes the vocabulary, training data, and post-training objectives. An in-place expansion from 131,072 to 163,072 tokens reduces mean Indian-language fertility by 38.6%. The new embedding rows are initialized from their upstream decompositions and warmed before the backbone is updated. Continual pretraining then adds broad English–Indic data, high-quality annealing, and direct training at a 131,072-token sequence length.

The central result appears after post-training. Reinforcement learning across four verifiable task environments improves mathematics, graduate-level science, and instruction following, but degrades Indic extractive question answering because the rewards do not require an answer in the requested language. The share of Indic extraction items answered in English rises from 8.7% to 91.7%. Accuracy conditional on answering in the target script roughly doubles, which is consistent with preserved reading ability but is measured on a self-selected subset.

Two unreleased studies test repairs. A scoped Indic reward restores the language convention. A 26B-A4B generative judge outperforms a 235B-parameter model on RM-Bench, and distillation restores 81% of the capabilities displaced by optimizing against it.

Relative to the architecture-identical starting checkpoint, the released model gains 5.2 points on an Indic aggregate while giving up 1.7 points of English retention. The unreleased Indic-aware branch gains 14.6 for 5.4. A planned FP8 release reduces the artifact to 32GB with near-zero mean change across 32 paired benchmarks.

Keywords: Indic language models, continual pretraining, tokenizer expansion, mixture of experts, Mamba, long context, supervised fine-tuning, GRPO, multilingual reinforcement learning.

Contents

1	Introduction	1
1.1	Contributions	1
2	Starting Point: An Efficient Reference Architecture	2
2.1	Why reuse Nemotron 3 Nano?	2
2.2	NoPE and the meaning of 128K context	2
3	Tokenizer Expansion	4
3.1	The language tax	4
3.2	Continued BPE and normalization	4
3.3	Fertility results	5
3.4	An exploratory 64K vocabulary study	6
3.5	Native versus romanized text	6
3.6	Initializing the new matrices	6
3.7	A tokenizer pathology worth reporting	7
4	Embedding Warmup	7
4.1	Why a separate stage is needed	7
4.2	Decoupled weight decay can move frozen rows	7
4.3	Recovery trajectory	8
5	Continual Pretraining	8
5.1	Data preparation and quality control	8
5.2	Choosing the learning rate	9
5.3	Measured CPT performance ceiling	9
5.4	The realized training ladder	11
5.5	Checkpoint evaluation across CPT phase 1 and annealing	12
5.6	Ablation-guided language allocation	12
5.7	Long-context mixture	12
6	Supervised Fine-Tuning	14

6.1	Role in the model story	14
6.2	Selected Evon-Instruct checkpoint	15
6.3	Data construction	15
6.4	Masking and long-sequence packing	15
6.5	What SFT establishes	16
7	Reinforcement Learning	16
7.1	GRPO in one equation	16
7.2	Released Evon-RL-Gen	17
7.3	Capability versus convention	19
8	Release Evaluation	19
8.1	Comparison with open models	19
8.2	Safety evaluation	19
9	Experimental Research Branches (Not Released)	21
9.1	A sibling Indic-aware branch	21
9.2	Learning from a judge	21
9.3	Recovering a convention by distillation	24
9.4	Weight-space alternatives to distillation	25
10	FP8 Quantization and Serving	26
10.1	Selective E4M3 W8A8 recipe	26
10.2	Calibration and validation	27
10.3	Serving performance	27
10.4	Matched BF16 versus FP8 quality	27
11	Compute and Infrastructure	29
12	Responsible Deployment	29
12.1	Intended use	29
12.2	Risks	29
12.3	Recommended deployment controls	30

13 Release and Reproducibility	30
14 Conclusion	31
A Architecture and Artifact Accounting	31
B Exploratory 64K Vocabulary Study	32
B.1 Production tokenizer chain of custody	33
C Training-Time and Throughput Diagnostics	33
D CPT Ablations and Non-lineage Validation Runs	34
D.1 Operational validation runs	34
D.2 The 120B mixing ablation	34
D.3 Uplift-weighted planning table	35
E Detailed CPT Configuration	36
E.1 Production CPT training traces	36
E.2 CPT data composition	38
F CPT Checkpoint Evaluation Series	39
F.1 HELMET long-context evaluation	41
G SFT Corpus Composition Across Retained Stages	41
G.1 Selected checkpoint accounting	41
G.2 Source-family evolution	43
H SFT Training Traces	43
I Matched Evaluation of the Released Checkpoint	46
J Indic Extractive-QA Failure Taxonomy	48
K Sibling Indic-Aware RL Study (Not Released)	49
K.1 Branch definition and reward	49
K.1.1 Why the experimental mix retains an English guardrail pool	49
K.1.2 Experimental mixture and configuration	50

K.1.3	Experimental matched results	52
K.1.4	Direct comparison with the released checkpoint	54
K.1.5	Experimental tool-use decomposition	55
L	Reinforcement-Learning Configuration, Side by Side	55
M	Reward Model and Distillation Study (Not Released)	55
M.1	Preference data	55
M.2	The judge’s reward	57
M.3	Reinforcement learning against the judge	58
M.4	Length control	58
M.5	The 70:30 mixture that failed	58
M.6	Distillation configuration	59
N	Matched BF16 and FP8 Benchmark Results	59
O	Glossary	61

1 Introduction

Large language models do not process words directly. They first split text into tokens, and the quality of that split depends on which languages were well represented when the tokenizer was built. For many Indian scripts, a tokenizer optimized primarily for English and code can turn one word into several fragments. Training, context capacity, latency, and serving cost are all paid per token, so this fragmentation creates a practical language tax.

Evon v3.3 was built around a simple question: how much Indic capability can be added to an efficient, already capable base model without discarding its English reasoning, code, or deployment advantages? The answer required more than adding translated data [11]. Four linked interventions were needed, following the pattern of other sovereign-language adaptation programmes [2]:

1. expand the vocabulary so that Indian scripts are represented efficiently;
2. adapt the new embedding and output rows without disturbing the inherited model;
3. continue pretraining on a quality-controlled English–Indic curriculum and teach the model directly at 128K context;
4. post-train for instruction following and reasoning, measure native-script retention, and study corrective language rewards in a separate experimental branch.

Table 1 gives the development path. In brief, tokenizer adaptation and continual pretraining produce Evon-Base; supervised fine-tuning produces Evon-Instruct; and the released RL stage produces Evon-RL-Gen. The sibling Indic-aware branch is experimental, not an ancestor of the release.

1.1 Contributions

This report makes six main contributions. The lineage they describe runs Evon-Base → Evon-Instruct → Evon-RL-Gen, with Evon-RL-Indic branching from Evon-Instruct as a research arm.

- **An Indic-aware tokenizer upgrade.** The vocabulary is expanded in place rather than replaced, improving eight-language fertility by 38.6% while preserving inherited token identities.
- **A guarded embedding-adaptation recipe.** Decoupled weight decay can move nominally frozen rows, while an Indic-only warmup can disturb English output calibration even with a frozen backbone.
- **A measured continual-pretraining curriculum.** Per-language uplift informs the data mixture, and direct training establishes the model’s 128K context length.
- **A reward-mixture diagnosis.** The released RL run improves reasoning but weakens unrewarded native-script conventions. A separate Indic-aware experiment tests a scoped repair.
- **A reward model beyond verifiable tasks.** An Indic-capable generative judge beats a nine-times-larger model on RM-Bench. Distillation then restores 81% of the capabilities displaced by optimizing against that judge while retaining most of its gains.
- **Negative results with operational value.** RL can erase output conventions without erasing capability; perplexity can select the wrong retention rate; and tool metrics can hide call policies.

Table 1: Production ancestry of the released RL checkpoint and its planned FP8 derivative.

Stage	Artifact	What changed
0	NVIDIA base checkpoint	Nemotron 3 Nano 30B-A3B Base BF16, upstream revision 97ab8012, with a 131,072-token vocabulary.
1	Tokenizer and initialization	A 32,000-entry continued-BPE extension, portable tokenizer backend, and constituent-mean initialization produce the 163,072-row vocabulary matrices.
2	Embedding warmup	New vocabulary rows are adapted on a 90% Indic / 10% English anchor mixture while inherited parameters remain fixed.
3	CPT phase 1	Broad 8,192-token English–Indic continual pretraining.
4	CPT annealing	High-quality 8,192-token continuation from the selected phase-1 checkpoint.
5	CPT long context	Direct 131,072-token NoPE training through iteration 372.
6	SFT-phase-1	First retained supervised curriculum stage.
7	SFT-phase-2	Second retained supervised stage.
8	SFT-rebuild-3 epoch 1	First pass over the rebuilt SFT-rebuild-3 corpus. Iteration 2,000 is the retained branch point.
9	SFT-rebuild-3 epoch 2 / Evon-Instruct	Epoch-2 continuation from SFT-rebuild-3 epoch-1 iteration 2,000; its iteration-2,400 export initializes both RL studies.
10	Evon-RL-Gen	Released four-environment GRPO checkpoint.
11	Planned Evon-FP8	Selectively quantized derivative of Evon-RL-Gen, planned for release.

2 Starting Point: An Efficient Reference Architecture

2.1 Why reuse Nemotron 3 Nano?

The upstream model is NVIDIA-Nemotron-3-Nano-30B-A3B-Base-BF16 [1]. It combines Mamba-2 sequence mixers [7], sparse attention, and granular mixture-of-experts layers. This matters for two reasons. First, only a small fraction of the total parameters is active for each token. Second, only six attention layers maintain a growing key–value cache, while the Mamba layers carry sequence state compactly.

Sparse experts provide capacity, while routing activates only a small subset for each token.

The released model contains 31.75B parameters, of which 3.31B are active per token under NVIDIA’s convention. Expanding both untied vocabulary matrices adds 172.03M parameters without changing the backbone. Table 2 summarizes the design, and Appendix A provides the exact accounting.

2.2 NoPE and the meaning of 128K context

Evon v3.3 uses no explicit positional embedding, so its 128K support comes from direct training on 131,072-token sequences rather than positional scaling. That establishes a trained sequence length, not uniform capability across it. At the CPT checkpoint, HELMET in-context learning rises from 59.2 at 8K to 81.7 at 128K, while exact retrieval falls from 95.9 to 14.0; retrieval-augmented generation and long-form question answering remain between those extremes.

Table 2: Architecture inherited from the Nemotron 3 Nano reference configuration.

Hyperparameter	Value
Total parameters	31,749,975,232 (31.75B)
Active parameters	3,313,773,760 (3.31B; NVIDIA convention)
Layer count	52 total: 23 Mamba-2, 23 MoE, 6 GQA
Model dimension	2,688
Attention heads / KV heads	32 / 2
Attention head dimension	128
Mamba-2 state dimension / groups	128 / 8
Mamba-2 heads / head dimension	64 / 64
Routed / activated / shared expert modules	128 / 6 / 1
Routed / shared expert hidden dimension	1,856 / 3,712 (2× width)
MoE activation	squared ReLU (<code>relu2</code>)
Router	learned MLP with sigmoid gating
Position encoding	none (NoPE)
Normalization	RMSNorm
Input embedding / output projection	untied
Evon vocabulary size	163,072
Maximum supported sequence length	131,072 tokens

Under the same completion prompting, the released checkpoint raises average recall from 57.1 to 84.0. Table 3 gives the remaining post-training results. Long-context ability therefore depends strongly on the task and prompt format, even within the supported length.

Table 3: Long-context results under matched completion prompting. Evon-RL-Indic is an unreleased research branch.

Measure	CPT checkpoint	Evon-RL-Gen	Evon-RL-Indic
Recall average	57.1	84.0	81.2
Abstention on unanswerable questions	98.1	99.8	99.7
IndicQA F1, HELMET reader slice	54.3	60.5	60.1

3 Tokenizer Expansion

3.1 The language tax

We measure tokenizer fertility as the number of tokens produced per whitespace-delimited word:

$$F(\mathcal{D}) = \frac{\sum_{x \in \mathcal{D}} N_{\text{tok}}(x)}{\sum_{x \in \mathcal{D}} N_{\text{word}}(x)}. \quad (1)$$

Lower fertility is better: the same sentence occupies fewer model tokens. This increases usable context and reduces the amount of token-level work needed to process the sentence.

The upstream vocabulary contains 131,072 entries. We continue its BPE merge process on data from Hindi, Bengali, Gujarati, Kannada, Malayalam, Marathi, Tamil, and Telugu, including romanized text during construction. The selected tokenizer adds 32,000 entries, producing a vocabulary of 163,072.

The released vocabulary contains 162,071 ordinary BPE entries and 1,001 added tokens. Of the added tokens, 19 have named functions and 982 are reserved placeholders. English and existing special tokens retain their upstream identities.

3.2 Continued BPE and normalization

Training a completely new tokenizer would break the mapping learned by the base model. Instead, continued BPE preserves inherited pieces and appends merges for sequences that remain fragmented. This follows the in-place expansion pattern studied by Smith et al. [3] and uses adaptive mixture ideas related to AdaptMix [4].

The production Round-09 tokenizer is built on the E3 corpus: Unicode NFKC normalization followed by newline/space cleanup and an Indic-aware regex canonicalization. NFKC folds compatibility variants into a canonical form before BPE learning, while the regex stage aligns better with intended boundaries.

E1 (NFKC only) and E3 (NFKC plus regex) have identical average fertility to four decimal places, 2.1823, in the controlled normalization sweep. Fertility could not arbitrate between them. E3 is the shipped normalization. It changes segmentation and token identities at nearly the same sequence length.

The production lineage uses the Round-09 32K extension and a 163,072-entry vocabulary. A later 64K extension achieved lower fertility but remained exploratory and was not adopted (Appendix B).

3.3 Fertility results

Figure 1 reports the production tokenizer on the common comparison protocol: Samanantar, 2,000 samples per language, seed 42. The eight-language mean falls from 3.5472 to 2.1773 tokens per word, a 38.6% reduction. Gujarati improves most (47.1%), followed by Malayalam (44.3%).

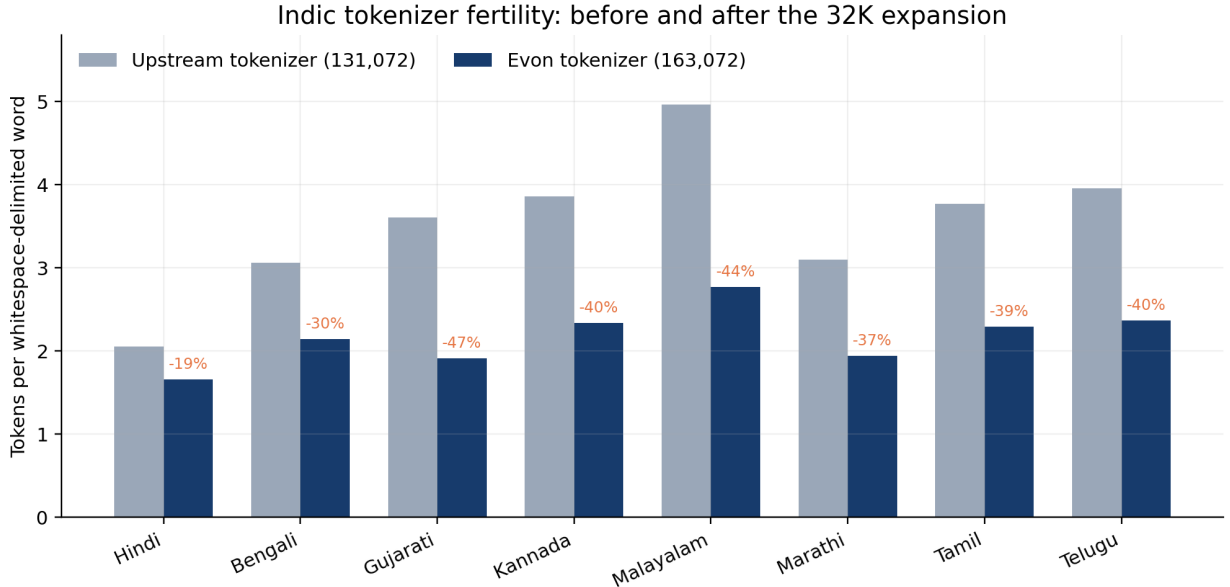


Figure 1: Tokenizer fertility on the held-out Samanantar evaluation slice. Lower is better. Percent labels show change relative to the upstream tokenizer.

Table 4: Per-language tokenizer fertility for the 163,072-entry tokenizer carried by the documented model lineage.

Language	Upstream	Evon v3.3	Change
Hindi	2.0513	1.6568	−19.2%
Bengali	3.0598	2.1416	−30.0%
Gujarati	3.6086	1.9099	−47.1%
Kannada	3.8643	2.3377	−39.5%
Malayalam	4.9654	2.7684	−44.3%
Marathi	3.0981	1.9387	−37.4%
Tamil	3.7705	2.2948	−39.1%
Telugu	3.9595	2.3706	−40.1%
Eight-language mean	3.5472	2.1773	−38.6%

3.4 An exploratory 64K vocabulary study

A larger 64,000-entry extension was trained and measured before the production vocabulary was fixed. It reaches lower fertility than the shipped tokenizer, at a cost of a second embedding matrix and a regression on one language, and it was not adopted. Appendix B gives the measurements and the reasoning.

3.5 Native versus romanized text

Romanized Indic remains substantially more expensive than native script. Across the new eight-language run, the macro romanized-to-native fertility ratio is 1.84. Seven languages fall in a range of approximately 1.59–1.94 times native fertility. Tamil is the clear outlier at 2.58 times: 2.314 tokens per word in native script versus 5.964 when romanized.

This does not invalidate the production tokenizer’s native-script improvement. It identifies its weak flank. Applications expecting heavy romanized Tamil or code-mixed Latin-script traffic should account for the larger token budget and may benefit from a future tokenizer revision.

Table 5: Native-script versus controlled ITRANS romanized fertility for the production tokenizer.

Language	Native T/W	Romanized T/W	Ratio
Bengali	2.106	3.355	1.59
Gujarati	1.917	3.168	1.65
Hindi	1.649	2.731	1.66
Kannada	2.343	4.219	1.80
Malayalam	2.832	4.958	1.75
Marathi	1.944	3.767	1.94
Tamil	2.314	5.964	2.58
Telugu	2.363	4.064	1.72
Macro average	2.183	4.028	1.84

The native column reproduces the standalone Round-09 fertility result closely (macro 2.183 versus 2.1773, with the largest per-language difference 0.064). The romanized samples are controlled ITRANS transliterations, not naturally occurring Latin-script web text, so the ratios isolate script conversion rather than broader domain differences.

The expansion did not directly target Punjabi or Odia. Their tokenizer efficiency should be measured separately before extending the eight-language fertility conclusions to them.

3.6 Initializing the new matrices

The input embedding and output projection are separate matrices (`tie_word_embeddings=false`). Each new BPE piece has an exact decomposition into upstream pieces. Its new row is initialized as the mean of those constituent rows, independently for the input and output matrices:

$$e_{\text{new}}(t) = \frac{1}{|S(t)|} \sum_{s \in S(t)} e_{\text{old}}(s), \quad w_{\text{new}}(t) = \frac{1}{|S(t)|} \sum_{s \in S(t)} w_{\text{old}}(s). \quad (2)$$

A new token starts near the representation of the old tokens that used to spell it. This is a much better starting point than a random vector, while preserving every inherited row exactly.

3.7 A tokenizer pathology worth reporting

One continued-BPE entry represented a degenerate space-plus-newline sequence. It could still be emitted through direct vocabulary lookup even after its merge path was removed. Because its initialized row was pathological, this single high-frequency whitespace entry contaminated many English encodings.

The slot was made unreachable without changing vocabulary size and was later repurposed as the atomic `<|file_sep|>` token at ID 144891. This preserves tensor alignment and provides a transparent provenance explanation for a special token located inside the vocabulary-extension range.

The broader lesson is simple: after tokenizer surgery, audit frequent encodings in every retention language. Merge-table reasoning alone is insufficient because production tokenizer backends may emit atomic vocabulary entries independently of the merge path expected by the model author.

4 Embedding Warmup

4.1 Why a separate stage is needed

Vocabulary expansion modifies only a small part of the model, but those rows participate in every read and every output softmax. The first adaptation stage therefore freezes the inherited backbone and inherited vocabulary rows while updating the new rows in both untied matrices.

The warmup consumes approximately 3B tokens from a 90% Indic / 10% English-anchor mixture. The English slice is small enough to keep the objective focused on the new entries, but large enough to preserve the competition between new and inherited output rows.

Table 6: Embedding-only warmup configuration.

Setting	Value
Trainable parameters	New vocabulary rows in the untied input and output matrices
Frozen parameters	Backbone and all inherited vocabulary rows
Data mixture	90% Indic, 10% English anchor
Training tokens	$\sim 3\text{B}$
Peak learning rate	1×10^{-4}
Parallelism	TP=1, EP=8, PP=1
Retention gate	HellaSwag plus bit-exact checks on frozen rows

4.2 Decoupled weight decay can move frozen rows

Setting a gradient to zero does not necessarily freeze a parameter under AdamW. The decoupled weight-decay update contains a direct parameter term:

$$\theta_{t+1} = \theta_t - \eta \hat{m}_t - \eta \lambda \theta_t. \quad (3)$$

If the gradient-derived term $\hat{m}_t$ is zero, the final term can still move θ_t . The first warmup implementation therefore allowed inherited rows to drift by about 4% in relative magnitude over 8,000 iterations, enough to collapse HellaSwag near random performance.

The fix snapshots inherited rows before each optimizer step and restores them immediately afterward. The guard is tensor-parallel aware, and a bit-exact comparison is run as a training invariant.

4.3 Recovery trajectory

Figure 2 shows the sequence of failures and corrections. Constituent-mean initialization reduced HellaSwag from 79.0% to 71.5%. The freeze-leak run reached 34.0%. An Indic-only warmup recovered to 50.0%, while the final anchored warmup returned to 79.0%.

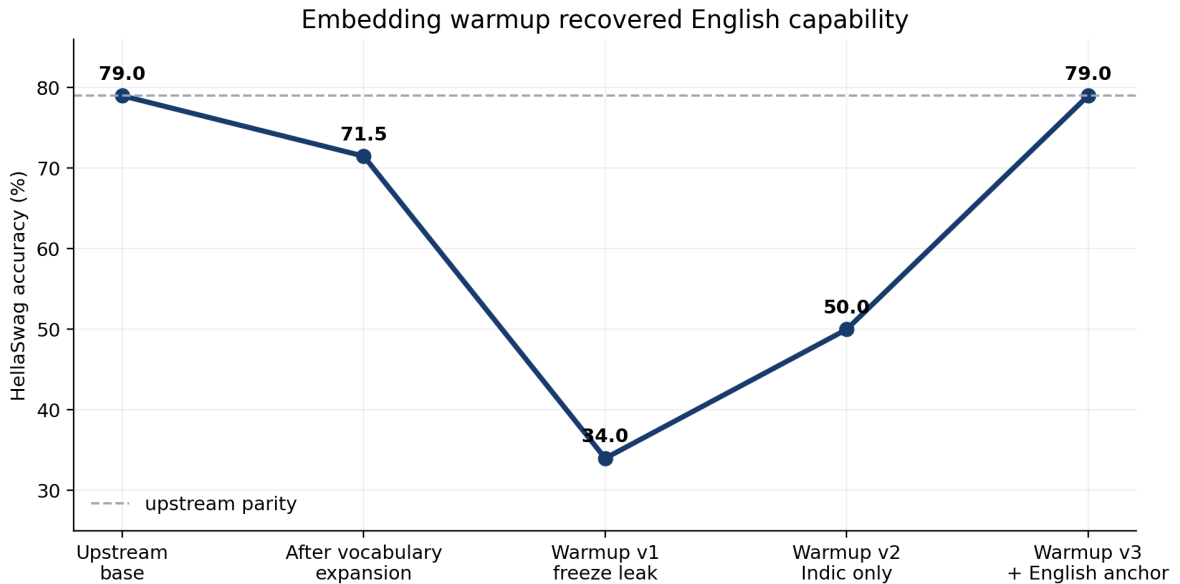


Figure 2: English-quality recovery across tokenizer initialization and embedding-only warmup.

This result is sometimes summarized incorrectly as a consequence of tied embeddings. The model is untied. The observed effect is instead about calibration: introducing and updating thousands of output rows changes the softmax competition faced by inherited tokens. A small retention-language anchor prevents the adaptation objective from optimizing only the new-language distribution.

5 Continual Pretraining

5.1 Data preparation and quality control

The CPT corpus combines English skill-carrying data, including code, mathematics, reasoning, and high-quality web text, with Indic web, academic, reference, and translated material [10]. The available Indic pool used for planning contains approximately 691B tokens.

Data processing applies exact deduplication followed by inter-document fuzzy deduplication. The implementation follows the practical ordering used in DataComp-LM processing: remove byte- or content-identical records first, then identify near-duplicate documents using fuzzy signatures [5]. Doing exact deduplication first reduces both compute and the number of trivial collisions seen by the fuzzy stage.

Every Indic document receives a continuous quality score and a discrete high-, medium-, or low-quality tier. The scorer combines structural length, script purity, character validity, repetition, boilerplate, URL density, language-model perplexity, and safety signals. Calibrated thresholds on the soft score assign the high, medium, and low tiers. Independent hard gates can reject a document regardless of its soft score.

Table 7: Quality-scoring signal groups.

Group	Examples	Role
Length and structure	word count, mean word length, mean line length	soft score
Script quality	target-script ratio, Latin ratio, invalid-character ratio	soft score / gate
Repetition	repeated words, characters, and n -grams	soft score
Web noise	symbol ratio, URL density, boilerplate density	soft score
Language fit	per-language KenLM log perplexity	soft score
Safety	NSFW and unsafe-content ratios	hard gate

Synthetic translations and rewrites are down-weighted in broad CPT and excluded from the highest-quality annealing slice. The aim is not to avoid synthetic data entirely. It is to prevent repeated translation style from dominating the late-stage distribution.

5.2 Choosing the learning rate

A four-arm, 1B-token-per-arm sweep compared peak learning rates of 10^{-4} , 5×10^{-5} , 10^{-5} , and 5×10^{-6} . Validation perplexity preferred 10^{-4} , but HellaSwag retention preferred 10^{-5} .

The disagreement is explainable. Validation perplexity was computed on a holdout close to the 90%-Indic adaptation mixture. It therefore rewards rapid Indic fit even when that movement costs English retention. The deployment objective was multilingual gain under a retention constraint, so 10^{-5} was selected.

Here, lower Indic-heavy validation perplexity did not imply a better multilingual model.

The selected schedule uses a peak learning rate of 10^{-5} , a minimum floor of 10^{-6} , a linear warmup, and a Warmup–Stable–Decay schedule with a minus-square-root decay tail. Broad CPT uses a global batch of 2,048 sequences at length 8,192.

5.3 Measured CPT performance ceiling

The production topology uses 256 H200 GPUs. Expert parallelism keeps each expert group within one eight-GPU node and its NVLink fabric. Table 8 gives the stage-level settings and measured throughput; Appendix E records the full configuration.

Aggregate throughput peaks at 2.87M tokens/s during annealing and falls to 1.68M during long-context training. Utilization still rises because attention adds more work per token.

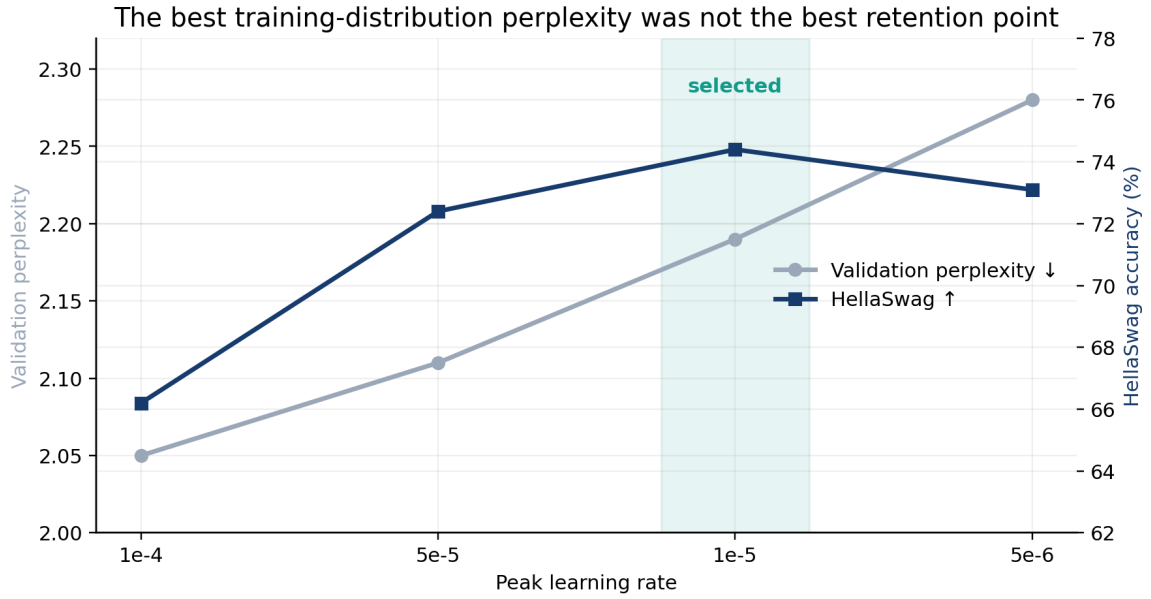


Figure 3: Learning-rate selection. Perplexity and downstream retention rank the candidates differently.

Table 8: Measured steady-state performance across the three CPT stages. MFU uses the 989 TFLOP/s BF16 peak.

Stage	s/iter	GFLOP/tok	tok/s/GPU	TFLOP/s/GPU	MFU
Broad CPT, 8k, CP=1	6.05	22.79	10,826	246.8	24.95%
Annealing, 8k, CP=1	5.84	22.79	11,226	255.9	25.87%
Long context, 128k, CP=8	159.77	40.91	6,563	268.5	27.15%

Appendix C reports timing and utilization, while Appendix D records non-lineage checks.

5.4 The realized training ladder

The training ladder contains three distinct roles. The broad phase builds Indic capacity without abandoning English skills. Annealing concentrates the distribution on higher-quality sources. The last phase trains directly on long documents at 131,072 tokens.

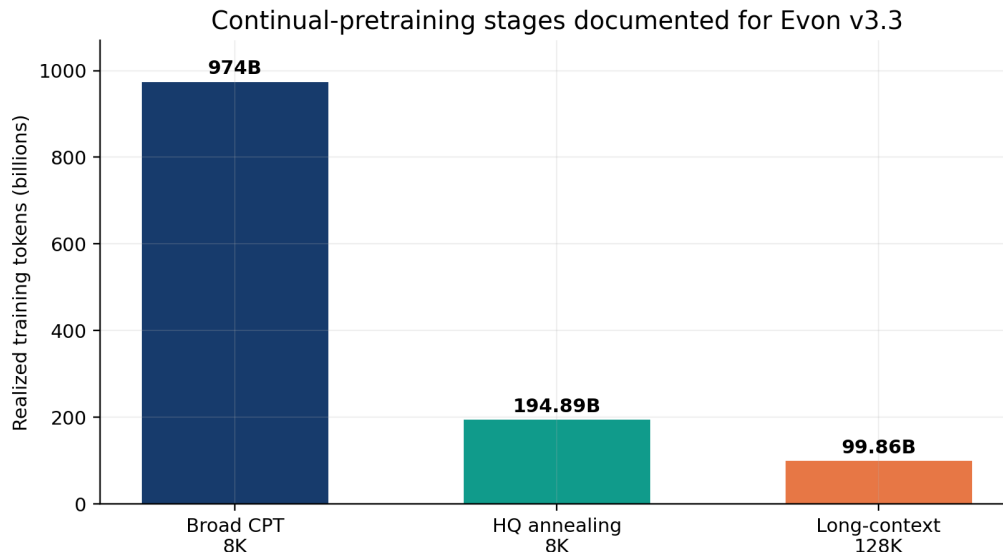


Figure 4: Realized production CPT token counts. The 128K extension consumed 99.86B of its 100B budget.

Table 9: Planned and realized continual-pretraining stages.

Stage	Context	Realized tokens	Role
Broad CPT	8,192	~974.49B	Diverse English–Indic skill and coverage phase.
HQ annealing	8,192	194.89B	Higher-quality continuation initialized from the best broad-CPT validation checkpoint.
Long-context extension	131,072	~99.86B	Direct NoPE long-document training; 372 of ~373 target iterations.

The long-context stage consumed 99.86B tokens and stopped one iteration short of its nominal 100B budget. Table 9 gives the batch, length, and stage totals.

Across broad CPT, annealing, and long-context training, the documented realized total is 1.269T tokens.

Appendix E.1 reproduces the available training panels for all three CPT stages, including learning rate, gradient norm, language-model loss, and MoE load balancing. The phase-1 panel covers the exported interval, not every retained checkpoint.

5.5 Checkpoint evaluation across CPT phase 1 and annealing

The matched checkpoint series shows where CPT changes the model. Figure 5 traces the trajectory, Table 10 gives selected endpoints, and Appendix F reports the complete 21-benchmark series. Absolute scores from this series should not be mixed with the earlier embedding-warmup evaluation.

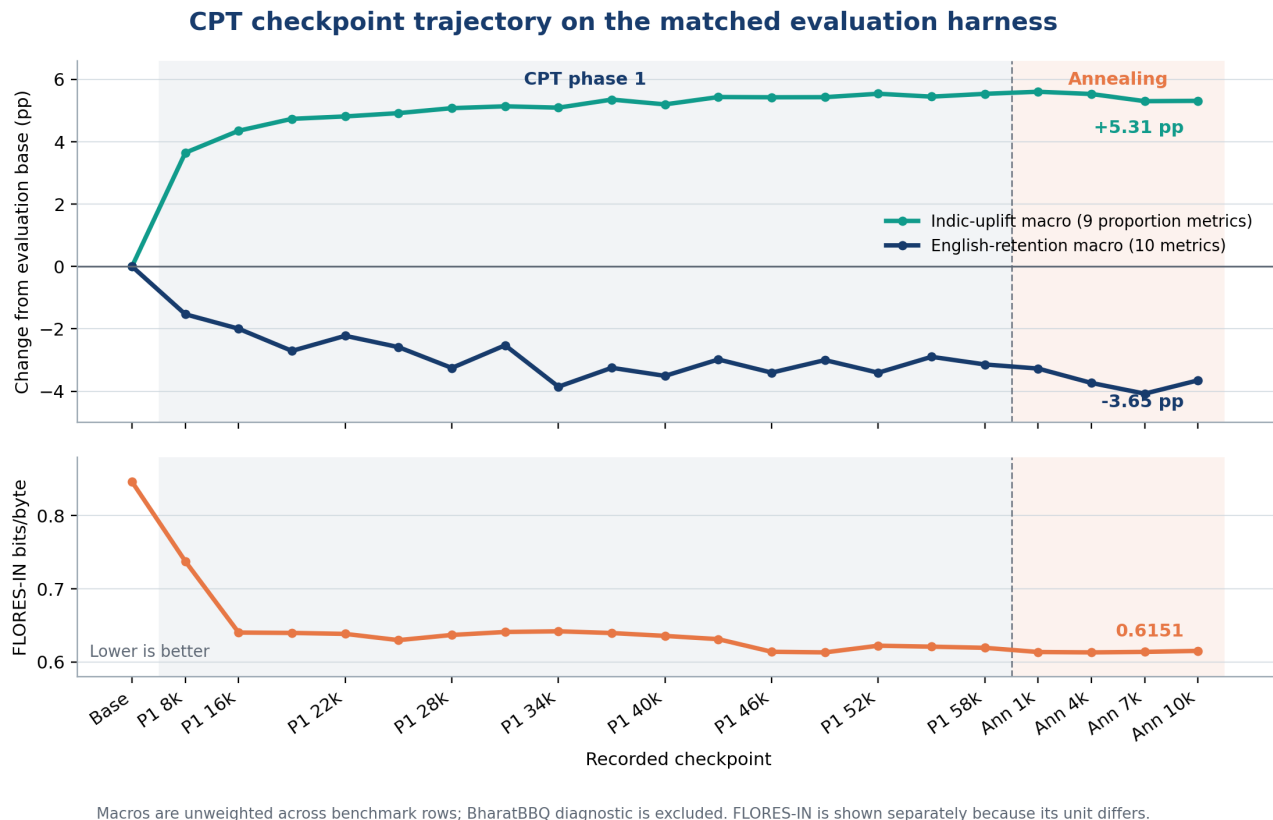


Figure 5: Matched CPT checkpoint trajectory. The upper panel shows Indic uplift and English retention; the lower panel shows FLORES-IN bits per byte. The divider marks the start of annealing.

Phase 1 produces most of the movement: broad Indic gains with a measurable English-retention cost. Annealing preserves those gains and further improves FLORES-IN bits per byte, while English results remain mixed. Because later SFT and RL evaluations use different protocols, only the trajectory within this series should be compared.

5.6 Ablation-guided language allocation

A 120B-token ablation showed that the weakest starting languages gained most, while a more Indic-heavy mixture did not improve the aggregate. These non-lineage studies informed later language allocation; Appendix D gives the results and planning weights.

5.7 Long-context mixture

The long-context manifest balances long Indic and English documents with mathematics, long-context SFT-style data, code, retention data, and a smaller multilingual component.

Table 10: Selected matched results across the CPT stages. The long-context column is Evon-Base.

Benchmark	Metric	Base	Phase 1	Annealing	Long context
Indic accuracy macro	accuracy (%)	41.09	46.83	46.63	46.83
MILU	accuracy (%)	40.63	53.54	53.52	53.24
ARC-Challenge-Indic	accuracy (%)	35.23	48.20	48.20	47.09
IndicGLUE-selected	score (%)	52.12	57.86	57.22	57.97
IndicMMLU-Pro	accuracy (%)	14.44	20.03	20.02	19.44
Indic BPB overall	bits/byte ↓	0.8453	0.6189	0.6132	0.6177
English 9-bench mean	accuracy (%)	70.62	67.98	68.40	67.95
MMLU	accuracy (%)	78.50	75.20	75.11	75.28
HellaSwag	accuracy (%)	85.57	83.52	83.69	83.69

Table 11: Composition of the 100B-token long-context pool. The selected run consumed ~ 99.86 B tokens.

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

We call the post-long-context checkpoint **Evon-Base**, and all later training starts from it. Appendix F compares it with the earlier CPT checkpoints.

6 Supervised Fine-Tuning

6.1 Role in the model story

Continual pretraining leaves Evon-Base fluent in the target languages and domains, but that does not by itself define a helpful assistant. SFT teaches the chat protocol: how to follow a request, produce a final answer, call tools, maintain a conversation, and use the requested language and script.

We call the public-facing SFT stage **Evon-Instruct**. Its retained path is SFT-phase-1 $\rightarrow$ SFT-phase-2 $\rightarrow$ SFT-rebuild-3 $\rightarrow$ corrected epoch 2. A continuation after SFT-phase-2 was discarded because of a data-splitting error, and the curriculum was rebuilt as SFT-rebuild-3. Experimental runs outside this path are not ancestors of the released RL model.

Table 12: SFT lineage relevant to the released RL checkpoint.

Step	Disposition	Role
SFT-phase-1	retained ancestor	first supervised curriculum stage
SFT-phase-2	retained ancestor	second supervised stage
intervening continuation	discarded	affected by data-point splitting; not in the released lineage
SFT-rebuild-3 epoch 1	retained branch point	first pass over the rebuilt SFT-rebuild-3 corpus. Iteration 2,000 is the branch point for epoch 2
Evon-Instruct (SFT-rebuild-3 epoch 2)	selected checkpoint	epoch-2 continuation from SFT-rebuild-3 epoch-1 iteration 2,000; [†] direct initialization for RL

[†]Epoch 2 restarts from SFT-rebuild-3 epoch-1 iteration 2,000 with fresh optimizer state; later epoch-1 checkpoints showed elevated MoE load-balancing loss.

The retained SFT stages also differ in corpus composition. Table 13 gives the row-level summary, and Appendix G gives the language mix and approximate source-family shares. The selected Evon-Instruct checkpoint uses the SFT-rebuild-3 corpus, whose materialized rows are 64.0% English.

Table 13: Materialized corpus summary across the retained SFT lineage.

Stage	Materialized rows	English share	Principal change
SFT-phase-1	$\sim 12.5\text{M}$	50.7%	Era-1 phase-1 pool; instruction following, mathematics, and code dominate.
SFT-phase-2	$\sim 14.7\text{M}$	54.6%	Era-1 phase-2 pool; code, science, and agentic shares increase.
SFT-rebuild-3	$\sim 15.2\text{M}$	64.0%	Era-2 rebuild with a larger English share.

6.2 Selected Evon-Instruct checkpoint

RL initializes from the iteration-2,400 export of the corrected SFT-rebuild-3 epoch-2 run. It represents 80.53B scheduled sequence positions, including padding. A direct sample of the assistant-only loss mask estimates roughly 15B loss-bearing positions. The run is stable at this point, with a recorded language-model loss of 0.614. Appendix G.1 gives the full export, packing, and schedule accounting.

6.3 Data construction

The SFT mixture covers English and eight Indic languages in native and romanized scripts [12, 13, 14]. It includes question answering, reasoning, mathematics, summarization, instruction following, tool interaction, and multi-turn dialogue. The final SFT-rebuild-3 build is 64.0% English and 36.0% Indic by materialized rows. Appendix G gives the source families; these row shares do not measure token exposure.

Gnani-published datasets use the Apache-2.0 terms stated on the Evon repository. Upstream datasets retain their original terms and attribution requirements.

Translation is structure-aware. User-facing prose can be translated, while code, mathematical spans, tool schemas, function names, and machine-readable arguments are preserved byte-for-byte. Failed or incomplete translations are quarantined rather than replaced with English text.

Conversations are normalized into a common schema with visible answer content separated from reasoning content. Deduplication is applied after normalization so that equivalent conversations stored in different source formats can match.

6.4 Masking and long-sequence packing

Training loss is applied only to assistant spans that should be learned. System and user history remain visible as context but are masked from the objective. Tool chains remain intact so that intermediate assistant actions and tool observations form one coherent training example.

Packing places multiple conversations into 131,072-token sequences while preserving their boundaries. Table 14 records the production topology and optimizer.

Table 14: SFT configuration for the long-context production family.

Setting	Value
Maximum sequence length	131,072
Precision	BF16
GPUs	256 NVIDIA H200
Parallelism	TP=1, PP=1, CP=4, EP=8, DP=64
Global / micro batch	256 / 1
Optimizer	Adam, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$
Weight decay / clipping	0.1 / 1.0
Packing objective	high utilization with assistant-only loss
Downstream role	initialization for both RL investigations

Appendix H reproduces the available Weights & Biases panels for SFT-phase-1, SFT-phase-2, and the final SFT-rebuild-3 stage. The stage-wise loss levels should not be compared as if they came from one stationary data distribution: each curriculum rebuild changes the source mixture, packing, and assistant-token composition.

6.5 What SFT establishes

The SFT initialization begins RL with strong native-script output: measured in-script rates range from 93.5% to 99.1% in one multilingual harness, and a later matched IndicIFEval-ground evaluation records 99.9%. It also establishes the response format and tool-call syntax expected by the RL parsers.

The purpose of reporting these pre-RL conventions is not cosmetic. RL optimizes whatever is rewarded by its environments. If a convention is present in SFT but absent from every RL reward, it can disappear quickly even while benchmark reasoning improves.

7 Reinforcement Learning

7.1 GRPO in one equation

The released RL stage uses group relative policy optimization (GRPO) [6]. For each prompt it samples $G = 16$ completions, centers each reward against the other fifteen, and scales all advantages by the whole group’s sample standard deviation:

$$\mu_{-i} = \frac{1}{G-1} \sum_{j \neq i} r_j, \quad (4)$$

$$\sigma = \sqrt{\frac{1}{G-1} \sum_j (r_j - \bar{r})^2}, \quad (5)$$

$$A_i = \frac{r_i - \mu_{-i}}{\sigma + \epsilon}, \quad \epsilon = 10^{-6}, \quad (6)$$

The division is applied only for $\sigma > 0$. Equal-reward groups contain no relative signal and are resampled. Appendix L records the estimator details and the later experimental floor.

The model learns which completion was better than the other completions for the same prompt. It does not require a separately trained critic to assign an absolute value to each response.

Dynamic sampling discards and replaces groups in which every completion receives the same reward. Consequently, a dataset’s nominal row share is not its effective training share:

$$s_{\text{effective},e} \propto s_{\text{nominal},e} p_{\text{mixed},e}, \quad (7)$$

where $p_{\text{mixed},e}$ is the fraction of prompt groups in environment e that contain more than one reward value. Realized shares must therefore be measured from rollout logs.

7.2 Released Evon-RL-Gen

Evon-RL-Gen is the public BF16 checkpoint and the source of the planned FP8 artifact. It initializes from Evon-Instruct and trains on four NeMo-Gym task environments:

- **math with judge**: final-answer equivalence after boxed-answer extraction;
- **structured outputs**: JSON validity and schema conformance;
- **workplace assistant**: correctness of an iterative tool-call sequence;
- **MCQA**: exact match on a selected option.

No custom reward shaping is applied in the released run. Each environment returns its own binary or task-specific scalar. There is no language reward, length reward, or repetition penalty.

Table 15: Released Evon-RL-Gen training configuration.

Setting	Value
Initialization	Evon-Instruct
Algorithm	GRPO with leave-one-out baseline and whole-group standard-deviation normalization
Resolved estimator flags	<code>use_leave_one_out_baseline=true; normalize_rewards=true</code>
Generations per prompt	16
Dynamic sampling	enabled; zero-variance groups resampled
Sampling temperature	1.0
Inference placement	colocated vLLM and training workers
Tool-call parser	Qwen3 XML parser
Reasoning parser	Nemotron-v3 parser
Custom reward shaping	none

The largest gains appear on tasks represented directly in the reward mixture: MATH rises 36.07 points and GPQA Diamond 26.84. Competition mathematics moves little because the SFT checkpoint is already strong there. Table 36 gives the matched results; extractive-QA comparisons use F1 because the exact-match rule changed between recorded runs.

The regressions cluster in Indic extractive question answering even as Indic free generation improves. On XQuAD-Indic, the share answered in English rises from 8.7% to 91.7% [15]. Conditional accuracy in the target script roughly doubles, although this is a self-selected subset rather than a controlled estimate.

English outputs are classified before grounding, so the lower ungrounded count is not independent evidence of better reading. The evidence points mainly to output language, although a smaller capability loss remains possible (Appendix J).

This behavior follows the reward boundary. The run contains no language reward, and English completions receive 0.1681 more reward on average, almost entirely from structured-output and workplace-assistant tasks. IndicIFEval is sensitive to evaluation state; the canonical release-protocol value is 33.05, and Appendix J records the other readings.

Released Evon-RL-Step-40 validation trajectory

Recorded validation means at steps 10, 20, 30, and 40; no step-0 row was supplied.

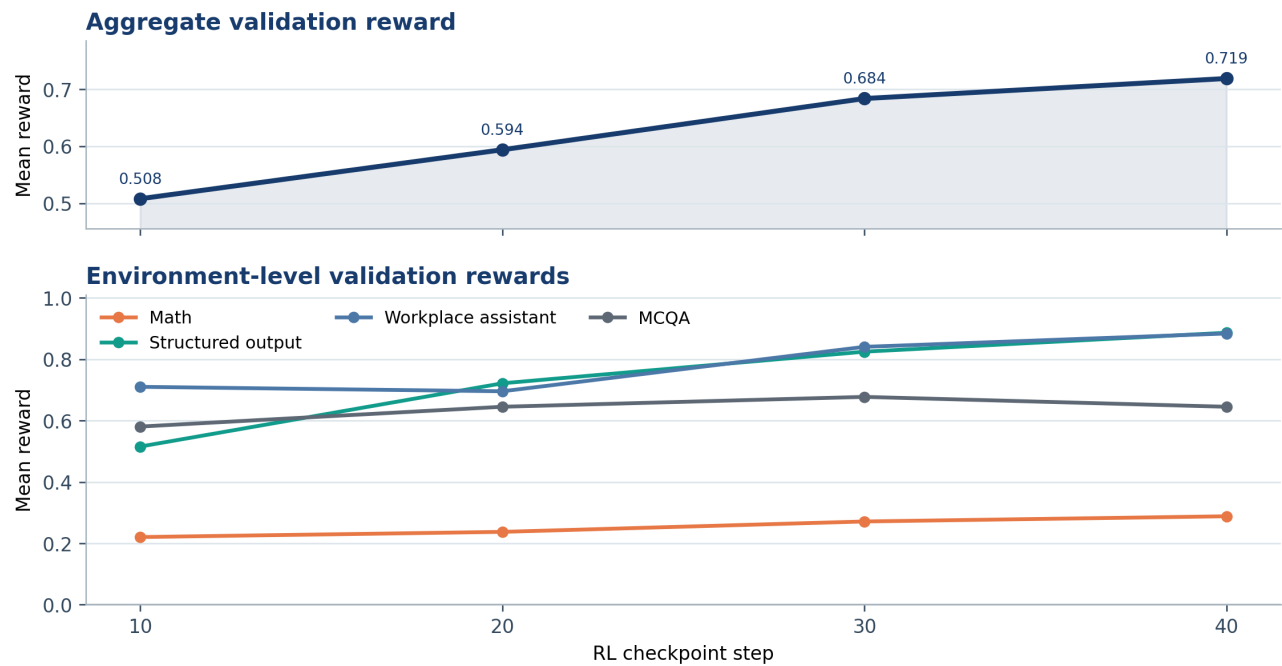


Figure 6: Released-run validation rewards. Structured output and workplace assistance improve most; the curves are optimization diagnostics, not benchmark accuracies.

7.3 Capability versus convention

The released RL study separates capability from emission convention. On 5,000 MATH problems, accuracy rises from 51.00 to 85.80, while conditional accuracy on boxed answers barely changes. Changing only the boxed-answer emission mix reproduces 94% of the gain; Appendix I gives the decomposition.

The exact decomposition depends on where the response is partitioned, but the conclusion does not: a benchmark swing can be carried largely by an output convention. Appendix I records the alternate partition and the unboxed-answer counts.

RL can destabilize a cheap output convention without erasing the capability underneath. Track in-script rate, boxed-answer rate, parse success, and tool-call count alongside benchmark accuracy.

This observation motivates a separate experimental branch: every product-critical convention needs a non-zero, correctly scoped training signal.

8 Release Evaluation

8.1 Comparison with open models

We evaluate Evon v3.3 and six open checkpoints on the same 35 benchmarks. The suite combines English knowledge, reasoning, and code benchmarks [20, 21, 22, 23, 24, 25, 26] with Indic question answering, instruction following, and generation benchmarks [16, 17, 18, 19]. Reference models use their published sampling settings, so the most reliable reading is movement within the Evon lineage, not a controlled ranking across providers. Two reference checkpoints emit a non-standard whitespace marker on code tasks; those code results are excluded.

Figure 7 summarizes equal-weight English-retention and Indic-capability aggregates. Relative to its architecture-identical starting point, the released model gains 5.2 Indic points while giving up 1.7 English points. The unreleased Indic-aware branch gains 14.6 for 5.4. Larger open models lead both aggregates; the figure is meant to show the movement produced by the pipeline.

8.2 Safety evaluation

IndicSafe evaluates 5,965 examples across 12 Indian languages. Overall accuracy is 86.50%. The model is strongest on harmless controls and ambiguous-but-safe prompts. The hardest categories are political manipulation and gender/caste discrimination.

BBQ moves in opposite directions by context type. Disambiguated accuracy rises from 94.13 to 97.70 and bias falls from 0.99 to 0.60; ambiguous accuracy falls from 92.29 to 88.69 and bias rises from 4.33 to 7.39. The model appears more willing to commit, which helps when context resolves the answer and increases stereotyped completions when it does not.

Safety scores are not a deployment guarantee. They measure one dataset and one decoding configuration. Applications should add domain-specific testing, moderation, logging, and human escalation.

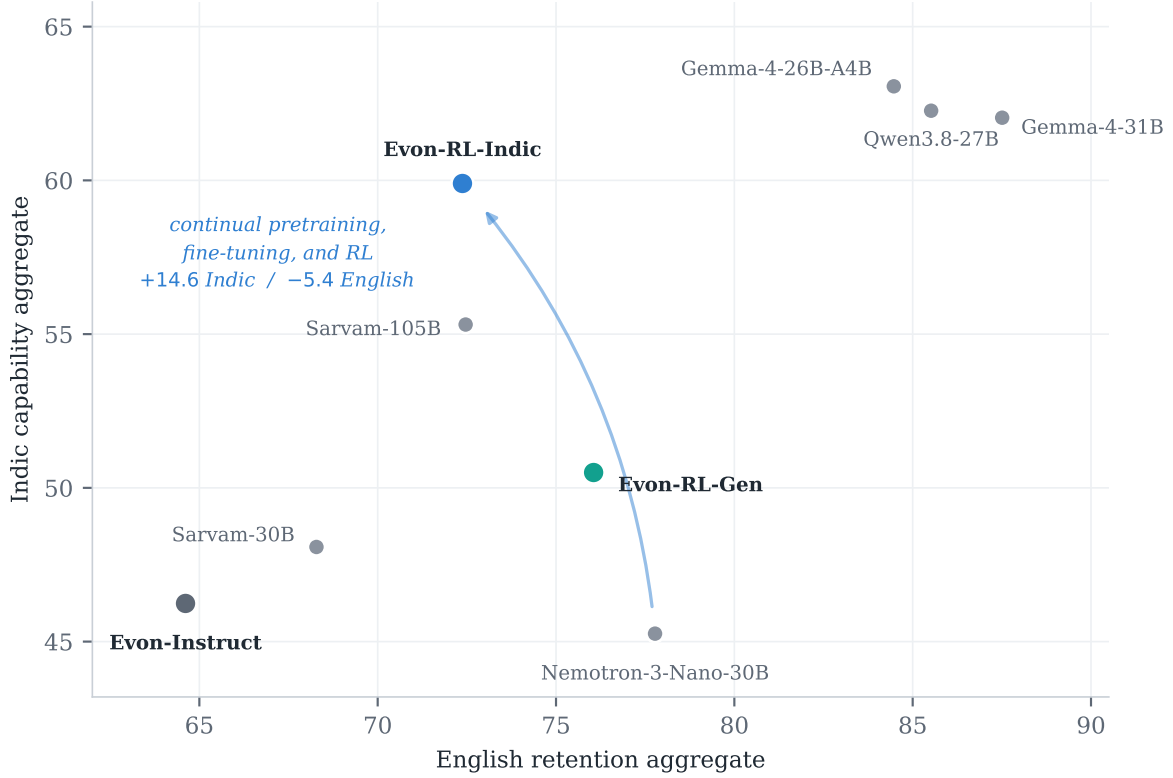


Figure 7: English retention and Indic capability. The arrow shows movement from the architecture-identical starting checkpoint; reference models use their published sampling settings.

Table 16: IndicSafe accuracy by harm category.

Category	Accuracy (%)
Harmless control	97.93
Tricky / ambiguous	97.77
Offensive or hate speech	89.80
Harmful instructions	85.62
Misinformation / conspiracy	83.50
Health misinformation	80.41
Religious hate / provocation	80.08
Political manipulation	72.49
Gender and caste discrimination	70.26

9 Experimental Research Branches (Not Released)

Everything in this section belongs to research lineage, not the public release. The experiments are included because their negative results transfer and clarify the limitation in Section 7.3.

9.1 A sibling Indic-aware branch

Evon-RL-Indic starts independently from the same Evon-Instruct export as the released run. It adds an Indic instruction-following environment, a scoped language reward, and a repetition penalty. On IndicIFEval-ground, the SFT score of 40.52 falls to 33.05 under the released mixture but rises to 60.00 under the experimental one. XQuAD-Indic F1 similarly moves from 36.51 to 17.19 and 50.44, respectively.

Because several ingredients change together, these gains cannot be attributed to the language reward alone. They do show that the released regression is a property of its reward mixture rather than an inevitable effect of RL. The branch also learns a one-tool-call-per-turn policy, showing that a targeted repair can create another convention. Appendix K gives the full analysis.

Thirteen of fourteen Indic benchmarks improve over the starting model, and eleven gain more than ten points. The losses cluster in competition mathematics, code, and instruction following (Figure 8).

The branch reaches 60.0 on grounded Indic instruction following, compared with 48.9 for the strongest open reference. It is competitive, rather than leading, on Indic extractive QA (Figure 9).

9.2 Learning from a judge

Verifiable rewards only reach tasks with a checkable answer. Mathematics, multiple choice, and schema conformance can be graded by a program. Open-ended helpfulness cannot. To extend reinforcement learning past that boundary we trained a reward model, a judge, and then optimized the policy against it. This section describes that judge, what training against it bought, and what it cost. The branch is a research lineage continuing from Evon-RL-Indic. It is not part of the released model.

An Indic preference set had to be built first. We combine native prompts with translated HelpSteer3-style prompts in a 70:30 ratio. Two models generate candidate responses and a third labels the preferred one. Of 98,744 pairs in Hindi, Telugu, Tamil, and Malayalam, 98,327 receive labels; Appendix M gives the construction details.

The judge writes a verdict rather than emitting a scalar. Initialized from a 26B-A4B model, it reasons over two responses and returns calibrated helpfulness scores plus a forced relative ranking. It scores 90.8 on RM-Bench against 87.3 for a 235B-parameter judge, with most of the margin on the hard slice. The comparison is against model-generated labels, however, and the judge has not been validated against human Indic preferences.

The instrument had the same disease as the patient. At a 4,096-token cap, JudgeBench falls from 88.3 to 83.7 because some responses never reach the verdict tag. Thus 4.6 apparent quality points come from the judge’s own emission convention; Appendix M gives the completion-rate analysis.

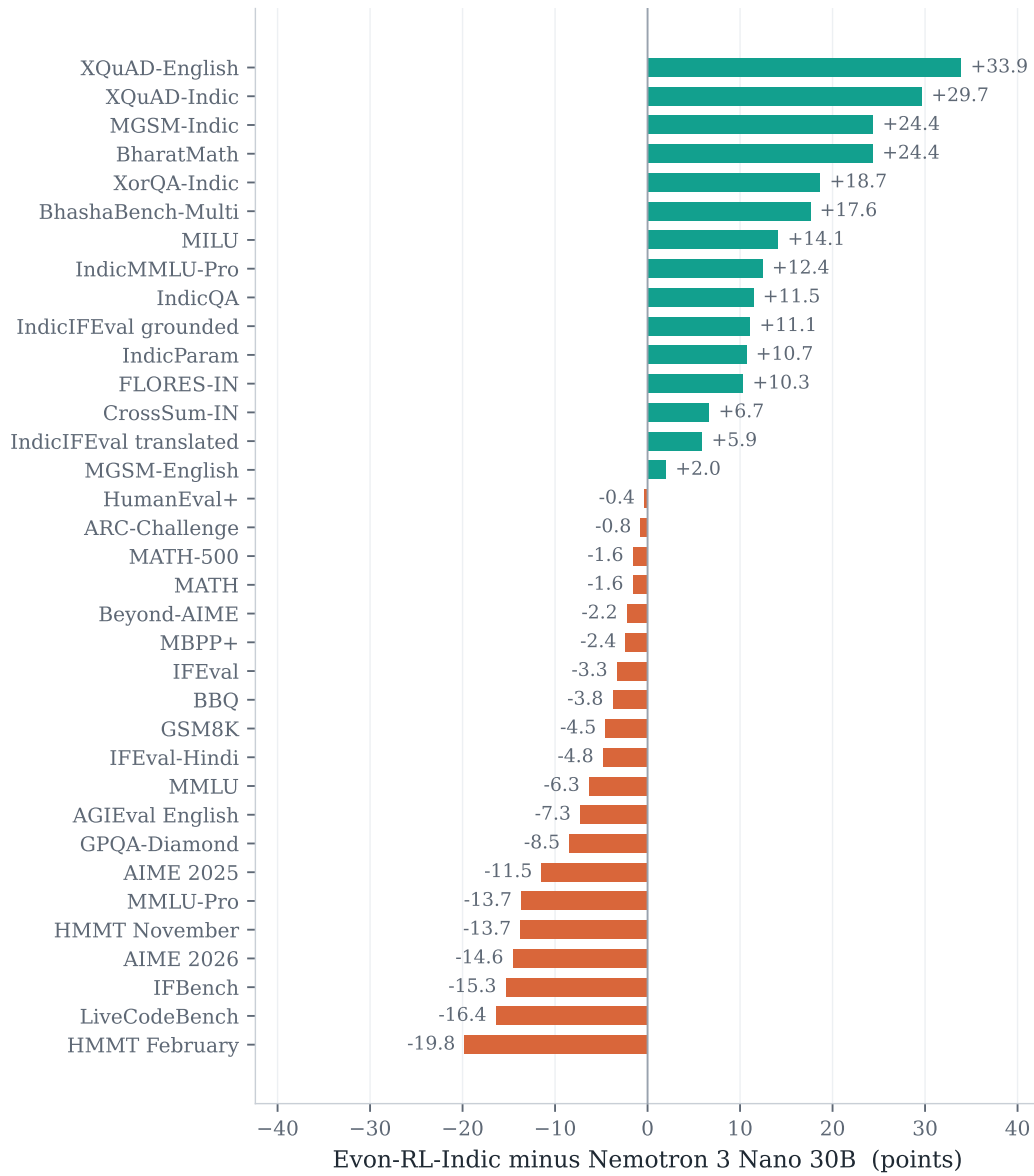


Figure 8: Per-benchmark change for the unreleased Indic-aware branch. Positive values favour the branch.

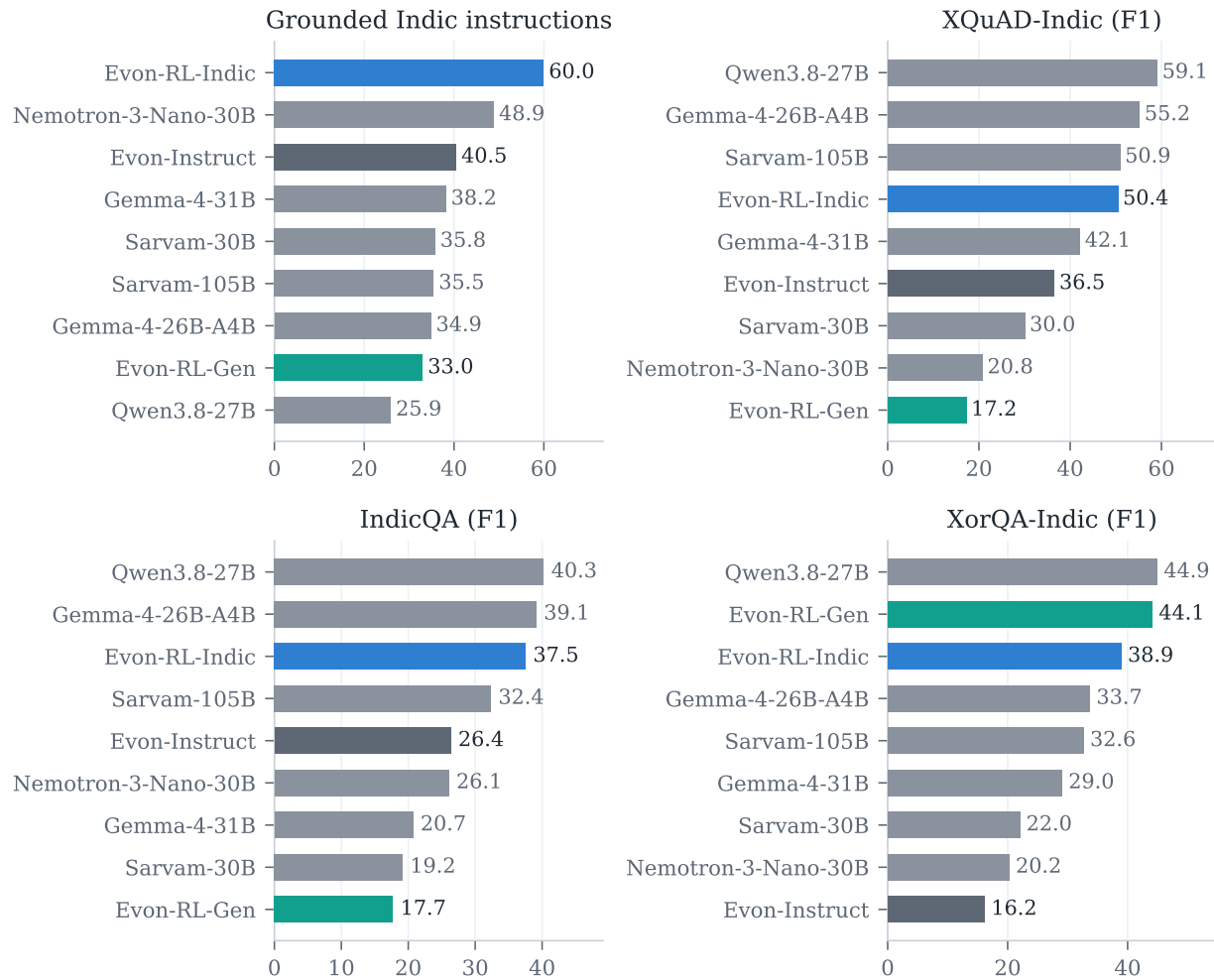


Figure 9: Grounded Indic instruction following and Indic extractive QA for the unreleased branch.

A reward model is measured by benchmarks that carry the same emission-convention confound as the policy’s own. Check a judge’s parse and completion rates before trusting its score, and re-check them at the generation cap you actually train with.

What training against the judge bought. The largest gains are on LiveCodeBench (+13.2) and MMLU-Pro (+8.8). Knowledge, reasoning, grounded Indic instruction following, and mathematics also improve, while empty or unterminated outputs fall on most gated benchmarks. Table 48 gives the complete set of headline gains; Appendix M also describes the ring comparison used for judging.

What it cost: the answer, wrapped in a sentence. Extractive question answering loses 24–38 F1 points even though answer spans remain present at similar rates. The parent returns a bare span; the judge-trained checkpoint wraps it in a sentence, adds a median of 9–17 tokens, and reduces overlap precision. Conversational elaboration has transferred to a terse-answer task.

What it cost: the requested language. FLORES-IN falls 17 chrF++ points for a different reason. English-target script compliance falls from 98.6% to 24.5%, and the model often copies the Indic source instead. Monolingual extraction changes the form of a largely preserved answer; cross-lingual tasks can lose the answer itself.

A length bonus drove reasoning to zero. Because the judge never sees hidden reasoning, reasoning length drifted. Rewarding the shortest trace drove its mean from 6,020 characters to zero while the judge score improved. A two-sided target still shortened reasoning by 60% and cost more than 25 AIME points. The adopted design is neutral below a generous budget and penalizes only traces above it.

Any bonus placed on the extreme of a property the model controls cheaply will be collected, and a two-sided length target is an attractor. Constrain such properties with a one-sided bound outside a permitted range rather than a reward inside it.

Mixing verifiable rewards back in did not work. A 70:30 mix of judge and binary-verifier prompts also failed. All-pass and all-fail verifier groups carry no relative signal, so dynamic sampling discarded 35–49% of them while every judge group survived. The intended mixture became roughly 80:20 at the gradient. The underlying pool contained too many problems the model almost always solved or almost always missed; Appendix M.5 gives the full analysis.

When two reward sources are mixed, the realized mixture is the one that survives filtering, not the one specified. Measure the share each source contributes at the gradient, and instrument every environment rather than the one most likely to be healthy.

9.3 Recovering a convention by distillation

The judge-trained checkpoint improves knowledge, code, and reasoning but loses useful conventions. On-policy distillation recovers those conventions without another reward model or verifier.

The parent teaches the child. The student starts from the judge-trained checkpoint, while Evon-RL-Indic serves as its teacher. On the student’s own responses, the teacher supplies a top-64 next-token distribution at every position:

$$\mathcal{L}(\theta) = \mathbb{E}_{x, y \sim \pi_S} \left[\frac{1}{|y|} \sum_t w \text{KL}(p_T^{(t)} \| p_S^{(t)}) + (1 - w) \text{KL}(p_S^{(t)} \| p_T^{(t)}) \right], \quad w = 0.5. \quad (8)$$

The two KL directions are weighted equally. One restores probability to forms retained by the teacher; the other suppresses forms the teacher rejects. Teacher and student share an architecture and tokenizer, and no reward is optimized during this stage.

Most of what was lost comes back, and most of what was gained stays. Across the six degraded benchmarks, distillation recovers 81% of the loss; extractive QA recovers 74–99%. Cross-lingual QA and translation return to their starting values within measurement noise. The student also retains all of the MMLU-Pro gain, about three quarters of the code and mathematics gains, at least half on MMLU, IndicMMLU-Pro, and AGIEval, and one third on GPQA Diamond.

The repair did not come from the repaired tasks. The 29,988 distillation prompts contain mathematics and multiple-choice questions, but no extraction, translation, or code. The repair therefore comes from the teacher’s token distribution rather than task-specific examples. Unlike a sparse reward, that distribution supervises every position, even when the desired form has become rare.

It is not free. Distillation moves the whole distribution, including behavior that did not need repair. IFEval falls 2.6 points, translated IndicIFEval 1.9, and Indic grade-school mathematics 4.8. Non-termination and empty-output rates also worsen on some tasks.

A convention lost to reward optimization can be restored by distilling from the checkpoint that still has it, without re-specifying the reward. The cost is that distillation moves everything, including behavior that did not need repair.

9.4 Weight-space alternatives to distillation

Two cheaper weight-space controls require no generation: equal interpolation with the parent and extrapolation beyond the judge-trained checkpoint. Neither is a release candidate.

Table 17 reports how much of the extractive question-answering loss each method returns. Distillation recovers 84% of it across the four benchmarks, interpolation recovers 41%, and extrapolation does not recover any of it. Extrapolation instead moves those benchmarks further in the direction the judge pushed them, ending well below the checkpoint it was meant to repair.

Table 17: Share of extractive-QA loss recovered by each repair. Negative values indicate further loss.

Benchmark	Loss to repair	Distillation	Interpolation	Extrapolation
XQuAD-English	38.25	+74.5%	+65.8%	−61.2%
XQuAD-Indic	32.77	+89.9%	+32.1%	−25.2%
IndicQA	24.45	+73.7%	+21.0%	−24.0%
XorQA-Indic	24.46	+99.1%	+46.1%	−34.6%
Macro		+84.3%	+41.3%	−36.2%

Interpolation preserves more of the judge’s gains but repairs only half as much extractive QA as distillation. Extrapolation makes the loss worse. Pushing farther along a reward direction therefore accelerates damage to behavior that the reward ignores.

10 FP8 Quantization and Serving

The FP8 checkpoint planned for release is produced from the frozen BF16 Evon-RL-Gen export using post-training quantization, not quantization-aware training. The pipeline uses NVIDIA Model Optimizer 0.45 through Megatron-Bridge [8]. No pruning, distillation, or INT4 weight-only compression is applied.

10.1 Selective E4M3 W8A8 recipe

Quantized sites use static E4M3 FP8 for both weights and inputs, with per-tensor symmetric scaling and no affine zero-point term. A max calibrator records the absolute maximum $a_{\max}$. Writing $F_{\max}$ for the largest finite E4M3 magnitude and Q_{E4M3} for rounding to the nearest representable E4M3 value with finite-range saturation, the conceptual quantize–dequantize operation is

$$s = \frac{a_{\max}}{F_{\max}}, \quad q = Q_{\text{E4M3}}\left(\frac{x}{s}\right), \quad \hat{x} = sq. \quad (9)$$

Unlike INT8, E4M3 is a floating-point format with a non-uniform representable grid determined by its exponent and mantissa. Quantization is therefore an FP8 cast after scaling, not uniform rounding to an integer code. Zero is represented directly by the floating-point format rather than introduced through an affine quantization zero point.

Each quantized tensor uses one scale that maps its largest observed magnitude into the E4M3 range. The stored values use roughly half the bytes of BF16, while sensitive modules remain in BF16.

The conservative recipe quantizes the routed and shared MoE expert input/output projections across all 23 MoE layers. Attention projections, attention matrix multiplications, Mamba projections and convolutions, routers, embeddings, final normalization, and the output head remain BF16.

Table 18: Precision allocation in the FP8 artifact planned for release.

Component	Precision	Rationale
Routed expert FC1/FC2	FP8 E4M3 W8A8	dominant parameter and compute block
Shared expert FC1/FC2	FP8 E4M3 W8A8	same expert-matrix structure
Attention QKV/output and BMM	BF16	excluded by conservative recipe
Mamba input/output projections and convolution	BF16	protects recurrent sequence-mixing path
Routers / routing gates	BF16	protects token-to-expert decisions
Embeddings, final norm, LM head	BF16	protects vocabulary and output calibration

FP8 covers approximately 94% of parameters and reduces the artifact from 63.5GB to 32GB, while routers remain in BF16. Appendix A provides tensor-level accounting.

10.2 Calibration and validation

Calibration requested 1,024 multilingual examples but exhausted the source at 253 packed rows. All artifact and paired-quality gates still pass, although a production recalibration should fill the intended sample. Weight scales use weight maxima; activation scales come from the calibration pass.

Artifact gates verify FP8 expert weights, BF16 exclusions, scale tensors, the hybrid layer pattern, tokenizer and EOS behavior, and the expected 32–34GB disk size. A no-op quantization that misses fused three-dimensional expert tensors remains near 62GB and fails this gate.

10.3 Serving performance

BF16 and FP8 are served with the same stack on one H200 GPU. At concurrency 16, FP8 raises aggregate throughput from 1,165 to 2,135 tokens per second and roughly doubles usable concurrency. The complete latency and rate measurements appear in Table 19 and Figure 10.

The gain differs from that of a dense transformer. Only six of the 52 layers contribute a growing key-value cache; the Mamba layers carry context-independent state. FP8 saves resident weight memory, increasing concurrency rather than usable context length.

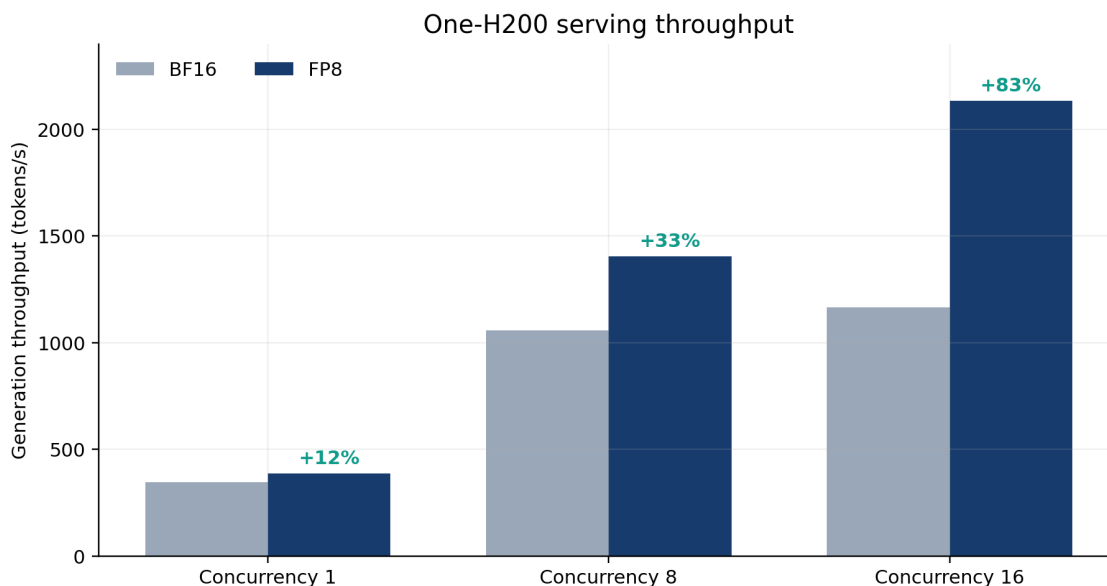


Figure 10: Paired BF16 and FP8 serving throughput on one NVIDIA H200 using the same serving stack.

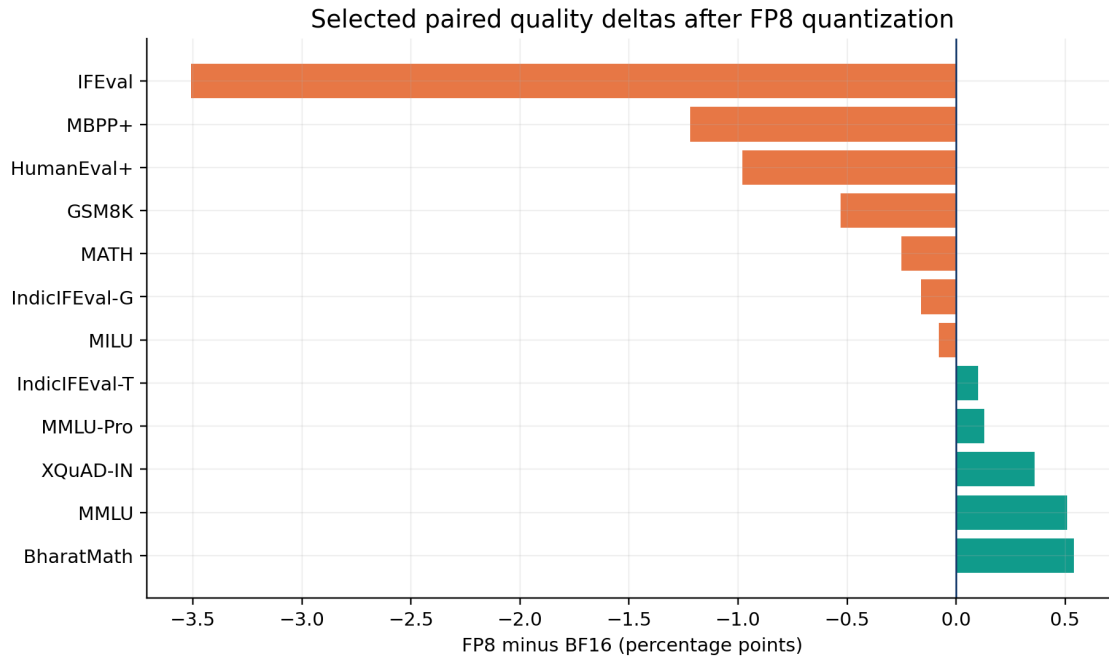
10.4 Matched BF16 versus FP8 quality

Across 32 paired accuracy benchmarks, the mean FP8-minus-BF16 delta is +0.0006, with seventeen positive and fifteen negative results. Only three absolute deltas exceed 0.02, all on samples of at most 541 items. IFBench is excluded because empty-response failures differ between the two arms. Appendix N lists every result. These are within-pair quantization deltas; their absolute scores should not be mixed with the release-evaluation campaign.

FLORES-IN and CrossSum-IN move slightly downward, as do HumanEval+ and MBPP+; Indic MCQ and QA are otherwise flat to slightly positive. On IFBench, 11.7% of FP8 and 10.3% of BF16 responses are

Table 19: Paired one-H200 serving measurements.

Metric	BF16	FP8
Warm TTFT, concurrency 1	348ms	341ms
Warm generation rate, concurrency 1	345.7 tok/s	387.3 tok/s
Generation-rate p50, concurrency 1	341.6 tok/s	382.9 tok/s
Aggregate generation rate, concurrency 8	1,059 tok/s	1,406 tok/s
Aggregate generation rate, concurrency 16	1,165 tok/s	2,135 tok/s
Usable concurrency under latency/rate gate	~8	~16
Disk footprint	~63.5GB	32GB

**Figure 11:** Selected paired accuracy deltas after FP8 PTQ. Small benchmark samples should not be over-interpreted.

empty, on only partly overlapping items, so the difference mixes instruction following with non-termination and is not included in the aggregate.

11 Compute and Infrastructure

Training uses NVIDIA H200 clusters, Megatron-Bridge for CPT and SFT, and NeMo-RL with NeMo-Gym for reinforcement learning. Expert and context parallelism distribute sparse routing and long sequences.

Table 20 reports GPU-hours for the released lineage. It excludes tests, discarded runs, retries, ablations, translation inference, and experimental branches. The SFT entry covers the retained curriculum.

Table 20: Recorded GPU-hours for relevant checkpoints in the released lineage.

Released-lineage training scope	Recorded GPU-hours
Embedding warmup	139
CPT broad 974B phase	~25,147
CPT annealing phase	~4,988
Complete 128K CPT long-context stage, allocated cluster time	4,977
Entire retained SFT curriculum	~17,205
Released Evon-RL-Gen production run	763
FP8 quantization and calibration	8

The 128K ledger uses allocated cluster time, including startup and recovery. The RL value covers the released step-40 run, and Appendix C gives detailed timing and MFU.

12 Responsible Deployment

12.1 Intended use

Evon v3.3 is intended for multilingual chat, long-context retrieval-augmented generation, English and Indic instruction following, mathematical reasoning, coding assistance, and tool-using agents. It can be deployed globally and self-hosted under the release license.

The model is not recommended as an autonomous authority for medical, legal, financial, employment, credit, or other high-stakes decisions. Deployers should evaluate the exact model artifact, quantization, prompt template, languages, and domain used in production.

12.2 Risks

The training corpus contains web and synthetic data and can reproduce factual errors, social bias, offensive material, or unsafe instructions. Multilingual safety is especially difficult because a safety behavior learned in one language may not transfer uniformly to another script, dialect, or code-mixed setting.

Reasoning traces can also expose unnecessary internal content, repeat sensitive prompt material, or create a false impression of certainty. Applications should decide explicitly whether reasoning is enabled and should not treat a generated chain of thought as an auditable proof.

12.3 Recommended deployment controls

- run target-language and target-domain safety evaluations before launch;
- apply input/output moderation appropriate to the jurisdiction and application;
- verify requested output language and script downstream for strict workflows;
- sandbox tools, validate arguments, and require confirmation for consequential actions;
- log model version, quantization, template, and decoding settings for incident analysis;
- disclose to users when they are interacting with an AI system;
- retain human review for high-impact or irreversible decisions.

13 Release and Reproducibility

The public BF16 weights are hosted at <https://huggingface.co/gnani/gnani-evon-v3.3-30B-A3B> under Apache License 2.0 [9]. Inherited components retain their original terms, documented in the release licence inventory. An FP8 derivative is planned; Evon-Base and intermediate checkpoints are not.

The release bundle records checkpoint and tokenizer identifiers, the chat template, the evaluation commit, and decoding settings. It also distinguishes report measurements from external results.

14 Conclusion

Evon v3.3 treats Indic capability as an end-to-end systems problem. The tokenizer reduces the cost of Indian scripts; embedding warmup integrates the new vocabulary; continual pretraining adds language capacity and direct 128K experience; SFT defines the assistant protocol; and the released RL stage improves reasoning and agentic performance.

The central lesson is that an optimizer cannot reliably preserve behavior that no environment rewards. The released mixture improves mathematics, structured output, multiple choice, and tool use, while native-script compliance becomes less reliable. An unreleased Indic-aware branch reverses that trend, but remains separate from claims about the public model.

The reward-model branch repeats the pattern: optimizing against a learned judge improves knowledge, code, and reasoning but weakens bare-span answers and English-target translation. The judge itself also loses 4.6 measured points when it fails to emit a verdict before the generation cap. Distillation from an earlier checkpoint restores most of the lost conventions without task-specific repair data.

Remaining work includes router-health traces for the shipped expert count, the exact NVIDIA source-repository identifier, checkpoint-specific SFT token accounting, the FP8 artifact identifier, and serving measurements beyond one device.

A Architecture and Artifact Accounting

The released model contains 31,749,975,232 parameters. Under NVIDIA’s convention, with six routed experts active and input embeddings excluded, 3,313,773,760 are active per token. Including the input embedding gives 3,752,111,296.

The vocabulary extension adds 32,000 rows to each of two untied matrices at model dimension 2,688:

$$\Delta P_{\text{vocab}} = 2 \times 32,000 \times 2,688 = 172,032,000. \quad (10)$$

Each MoE layer contains 128 routed experts of width 1,856 and one shared expert of width 3,712. With a `relu2` MLP, this gives 258 expert tensors per layer, or 5,934 across 23 layers. Routers remain separate and in high precision.

Ignoring biases, a routed expert contains 9,977,856 parameters and the shared module contains 19,955,712. The expert weights therefore contribute

$$P_{\text{MoE}} = (128 \times 9,977,856 + 19,955,712) \times 23 = 29,833,789,440, \quad (11)$$

$$P_{\text{MoE,active}} = (6 \times 9,977,856 + 19,955,712) \times 23 = 1,835,925,504. \quad (12)$$

The embeddings, output projection, Mamba, attention, normalization, and routing parameters reconcile these figures with the measured 31.75B total and NVIDIA’s 3.31B active-parameter convention.

B Exploratory 64K Vocabulary Study

mixv5 is the strongest completed fertility experiment. It uses an approximately 8B-token mixture, balanced across the eight target languages and spanning native, romanized, and transliterated text. It is a parallel study and does not alter the production CPT lineage.

The 64K extension reduces mean Samanantar fertility by 40.8% relative to the upstream tokenizer. It improves further on the selected 32K tokenizer overall, but Malayalam is slightly worse. The per-language results are reported in Table 21.

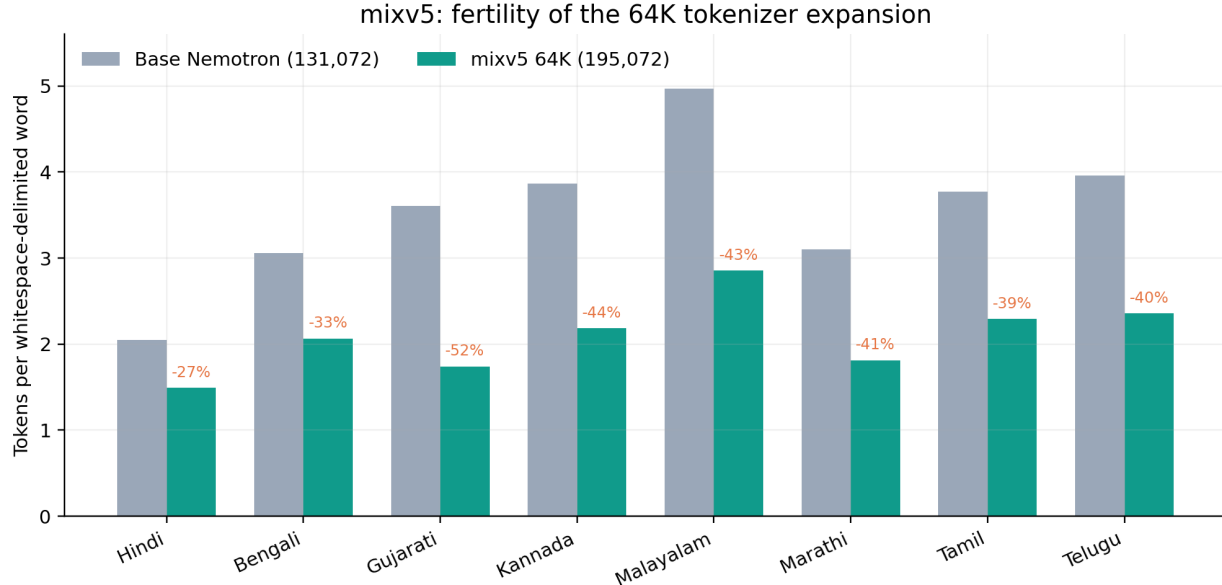


Figure 12: Fertility of the exploratory 64K mixv5 tokenizer against the upstream tokenizer. This tokenizer was not used by the production model.

Table 21: Per-language fertility for mixv5, evaluated on 2,000 Samanantar samples per language with seed 42.

Language	Base Nemotron	mixv5 64K	Change
Hindi	2.0513	1.4924	−27.2%
Bengali	3.0598	2.0637	−32.6%
Gujarati	3.6086	1.7402	−51.8%
Kannada	3.8643	2.1830	−43.5%
Malayalam	4.9654	2.8548	−42.5%
Marathi	3.0981	1.8135	−41.5%
Tamil	3.7705	2.2906	−39.2%
Telugu	3.9595	2.3585	−40.4%
Mean	3.5472	2.0996	−40.8%

The normalization variants are nearly indistinguishable by fertility. The shipped regex variant therefore reflects segmentation quality, not a measurable sequence-length advantage.

B.1 Production tokenizer chain of custody

The production tokenizer descends from AdaptMix Round 09. Of its 32,000 extension entries, 31,999 remain byte-identical in the staged CPT tokenizer. The only intentional exception is ID 144891.

Model preparation removes one degenerate merge and assigns ID 144891 to an unreachable placeholder. Before SFT export, the slot becomes `<|file_sep|>`; only its label changes.

Production therefore uses the 163,072-entry AdaptMix Round-09 lineage. The 195,072-entry mixv5 result is a later parallel study and should never be described as the released tokenizer.

An independent loader and sampler reproduces the native-script macro fertility within 0.006 of the reported result. A smaller off-protocol evaluation is retained only as a robustness check.

C Training-Time and Throughput Diagnostics

Iteration time was logged every tenth optimizer step. Table 22 reports the steady median after step 50 and the full logged range, including startup and recovery costs.

The CPT phase-1 figures combine both segments of that stage. Its step time drifts upward across training, so the median reflects the settled value rather than a stationary rate; the other stages are stationary.

Table 22: Optimizer-step timing across the production CPT and SFT stages.

Stage	Logged n	Median	p5–p95	Max / steady
CPT phase 1	5,808	6.05	5.63–6.18	16.73 / 6.05
CPT annealing	1,161	5.84	5.81–5.97	20.80 / 5.84
CPT long context	37	159.89	157.73–162.65	173.96 / 159.91
SFT-phase-1	207	16.71	15.43–18.09	52.85 / 16.70
SFT-phase-2	383	17.40	16.55–18.31	49.99 / 17.39
Evon-Instruct	266	17.58	16.74–18.42	60.07 / 17.57

These medians supply the throughput and utilization values in Table 8.

Table 23 reports medians after step 50 from the recorded SFT telemetry. MFU is relative to the H200 BF16 peak of 989 TFLOP/s per GPU.

Table 23: Measured steady SFT throughput by retained stage.

Stage	TFLOP/s/GPU	Aggregate TFLOP/s	Tokens/s	Tokens/s per GPU	Seq./s	MFU
SFT-phase-1	321.2	82.2k	1.938M	7,570	14.8	32.5%
SFT-phase-2	308.4	79.0k	1.894M	7,398	14.4	31.2%
Evon-Instruct	305.2	78.1k	1.870M	7,304	14.3	30.9%

D CPT Ablations and Non-lineage Validation Runs

The experiments in this section informed configuration and mixture choices but are not checkpoints in the released model’s ancestry.

D.1 Operational validation runs

A separate 5B-token smoke run validated 128K memory fit, NoPE execution, and cache construction. Smaller regression runs checked the topology. Neither is included in production quality or compute totals.

D.2 The 120B mixing ablation

The 55:45 English–Indic ablation showed that marginal value did not track data abundance. Kannada gained 14.4 points, while Hindi, the strongest starting language, declined slightly. A second, more Indic-heavy mixture did not improve the aggregate. Later planning therefore combined measured uplift with source quality and repetition constraints; Figure 13 and Table 24 give the full language breakdown.

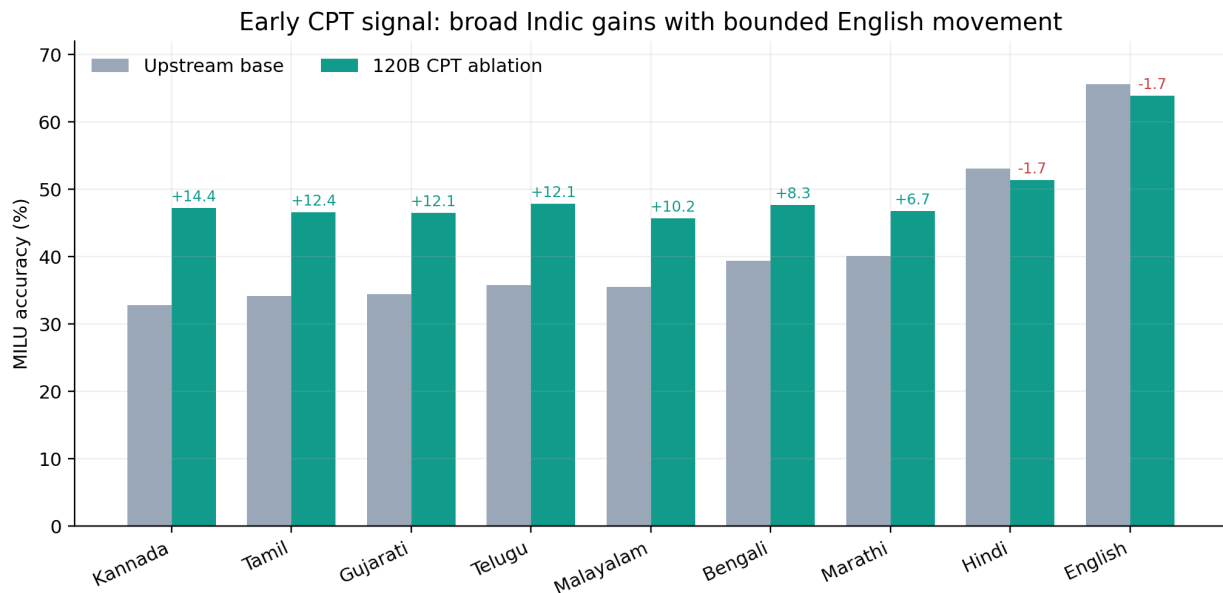


Figure 13: MILU accuracy before and after the 55:45 English–Indic 120B CPT ablation. Labels show percentage-point change. This run is not part of the released checkpoint ancestry.

Table 24: Per-language MILU results from the 120B ablation. Displayed endpoints are rounded to one decimal place; deltas are computed from the underlying higher-precision scores.

Language	Base	120B CPT	Δ
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

D.3 Uplift-weighted planning table

Table 25 records the planning weights derived from measured uplift and high-quality data availability.

Table 25: Language weights derived from 120B-ablation uplift and high-quality availability. MILU uplifts use the underlying higher-precision scores.

Language	MILU uplift	HQ availability	Weight	Est. epochs
Telugu	+12.1	20B	16%	1.03
Tamil	+12.4	12B	15%	1.60
Kannada	+14.4	6B	13%	2.78
Gujarati	+12.2	8B	12%	1.92
Malayalam	+10.1	17B	12%	0.90
Bengali	+8.4	19B	12%	0.81
Hindi	−1.8	35B	12%	0.44
Marathi	+6.7	9B	8%	1.14

E Detailed CPT Configuration

Table 26: Selected broad-CPT configuration fields.

Field	Value
Initialization	embedding-warmed 163,072-vocabulary checkpoint
Precision	BF16
Sequence length	8,192 for broad CPT and annealing
Broad-CPT global / micro batch	2,048 / 1
Parallelism	TP=1, EP=8, PP=1; data parallelism fills remaining ranks
Optimizer	distributed AdamW
Peak / minimum learning rate	1.0×10^{-5} / 1.0×10^{-6}
Schedule	Warmup–Stable–Decay with minus-square-root decay
Warmup	200 iterations in the locked production recipe
Stable fraction	0.8
MoE dispatcher	flex + HybridEP
Expert capacity	capacity factor 1.25 with pad-to-capacity
Kernels	grouped GEMM, selective recomputation, CUDA graph capture for attention and Mamba
Tracking	Weights & Biases; supplied production panels reproduced below

E.1 Production CPT training traces

The panels below reproduce the production training series under the public stage names. The sequence load-balancing curve is the MoE auxiliary objective, not the language-model loss.

CPT phase 1 ran in two segments with a 123-iteration overlap. Their language-model losses agree to 3×10^{-5} in that interval; Figure 14 shows the combined trace.

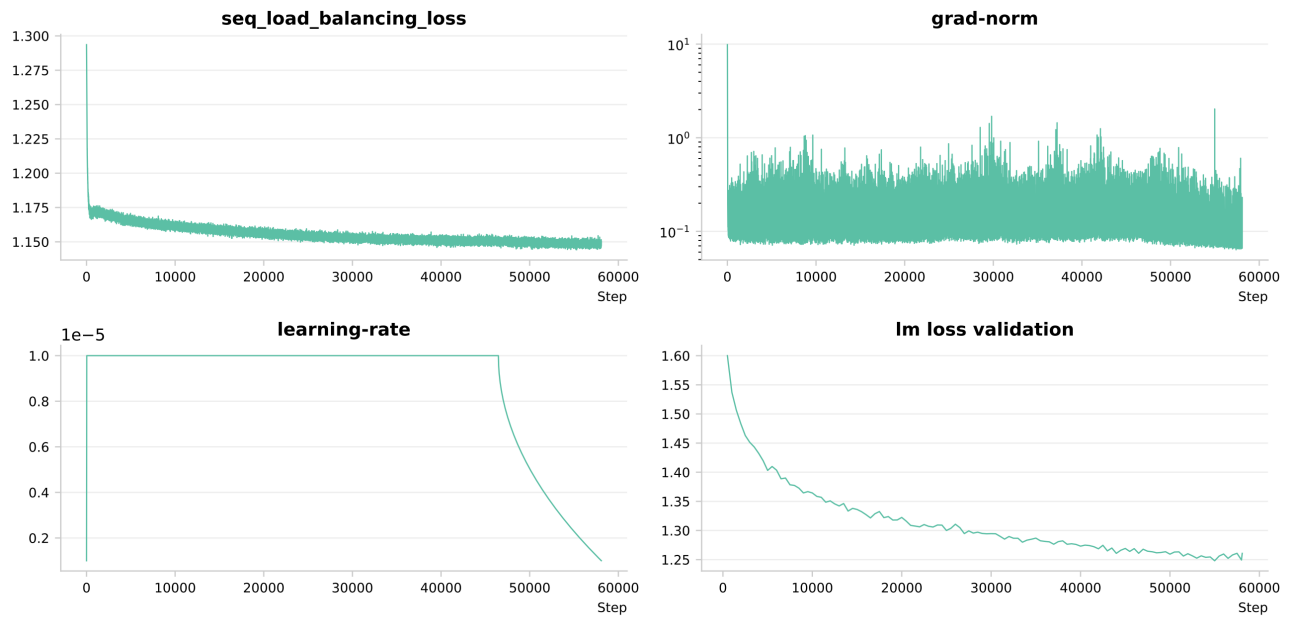
CPT phase 1

Figure 14: CPT phase-1 training traces through iteration 58,084.

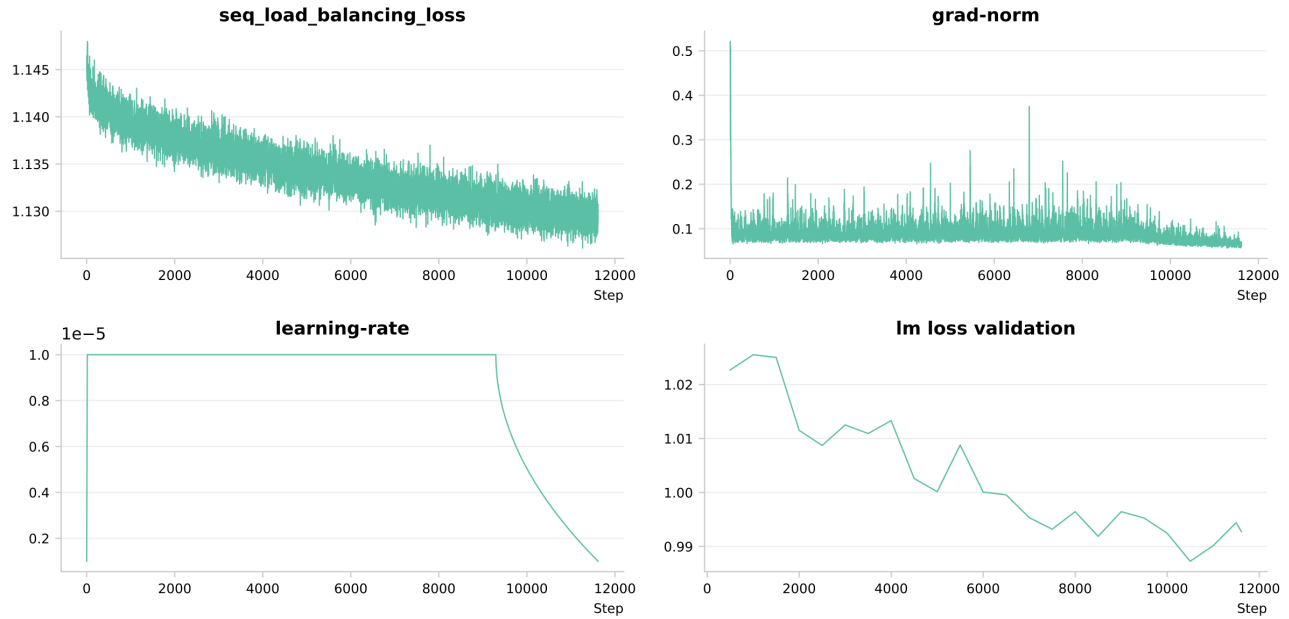
CPT annealing

Figure 15: CPT annealing traces. Learning-rate decay begins late while validation loss remains near 1.0.

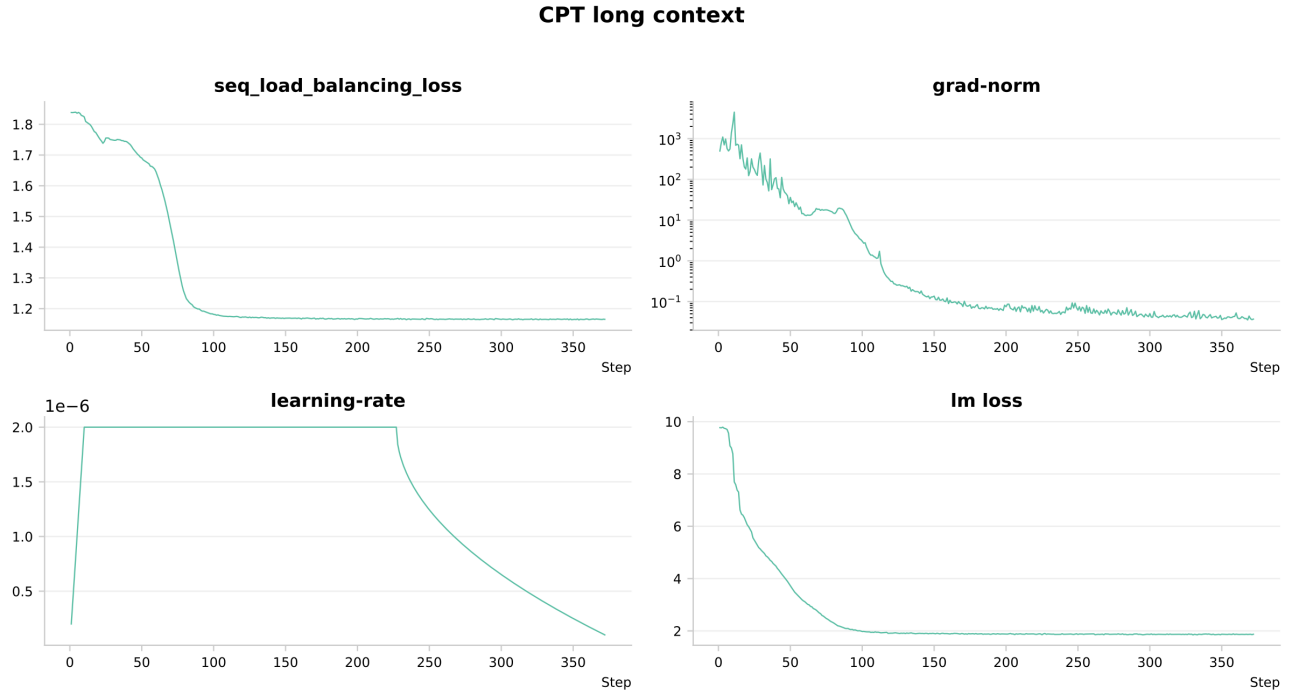


Figure 16: CPT long-context traces through iteration 372; the early transition settles into stable training.

E.2 CPT data composition

The three continual-pretraining stages draw on separate blend manifests. Broad CPT uses a 974B English-majority mixture and annealing a 196B high-quality mixture, while the long-context stage uses the pool already reported in Table 11. The broad-CPT manifest retains 184 shard prefixes; 90 shards from its source manifest were dropped for absent per-shard caches and the remaining weights renormalized to unity.

Two share conventions appear in the manifest audit block and are easy to conflate. A *target* share is a fraction within the English or Indic budget; a *sampling* share is a fraction of the whole mix. The same code allocation reads 53.2% one way and 29.1% the other. The figures below are realized sampling shares, computed as $\text{weight} \times 974.5\text{B}$ from the per-source accounting, and are therefore comparable across the whole table. Blend weights are sampling probabilities rather than guarantees: realized per-source consumption also depends on shard exhaustion, and several sources in the annealing manifest are scheduled for less than one epoch.

Table 27: Broad-CPT composition from the 184 retained shard weights.

Group	Target	Realized	Share	Δ
English	536.0B	532.7B	54.66%	−3.3B
Indic (12 languages)	438.5B	436.6B	44.80%	−1.9B
Other multilingual	–	5.3B	0.54%	–
Total	974.5B	974.5B	100%	

The largest shard is English code at 12.21% of the mix, followed by synthetic code question answering at 6.90%. No Indic shard exceeds 2.5%.

Table 28: Realized Indic allocation in broad CPT, by language. Only the first eight languages received dedicated vocabulary expansion (Section 3); Odia, Punjabi, Urdu and Nepali receive pretraining data on a small budget without tokenizer support.

Language	Tokens	Share	Language	Tokens	Share
Hindi	132.4B	13.59%	Marathi	17.1B	1.76%
Tamil	69.3B	7.11%	Odia	4.4B	0.45%
Telugu	66.6B	6.84%	Punjabi	1.6B	0.17%
Bengali	53.8B	5.52%	Urdu	1.4B	0.14%
Malayalam	33.5B	3.43%	Nepali	1.3B	0.14%
Kannada	28.7B	2.95%			
Gujarati	26.3B	2.70%			

Table 29: Annealing composition by bucket from the manifest’s per-dataset allocations.

Bucket	Allocated	Share
Reasoning	62.70B	32.2%
Knowledge retention	34.20B	17.5%
Mathematics and STEM	31.52B	16.2%
Code	25.65B	13.2%
Indic high-quality	17.10B	8.8%
Instruction	10.10B	5.2%
Multilingual	5.70B	2.9%
Web	5.70B	2.9%

The listed allocations are 1.14% below the stage budget because two dropped shards were redistributed across the survivors. Annealing contains 11.7% Indic data, down from 44.8% in broad CPT; its long-document complement supplied the long-context stage.

That reduction should not be read as the cause of the further MMLU decline recorded for this stage in Table 30. The English majority in annealing is skill-shaped rather than knowledge-shaped: reasoning, mathematics and code together account for 61.5% of the stage against 17.5% of explicit knowledge retention. A stage that is 88% English can still cost English knowledge when little of that English is knowledge-dense, which is the more plausible mechanism for the observed change.

F CPT Checkpoint Evaluation Series

The matched series covers the base and each CPT endpoint. Table 30 reports all benchmarks, with percentage and bits-per-byte metrics kept separate. Deltas compare long context with the base and should be read within rows; ‡ marks a diagnostic, and ◦ an unevaluated slice.

Table 30: Per-benchmark matched trajectory across the three CPT stages.

Benchmark	Metric	Base	Phase 1	Annealing	Long ctx.	Δ
<i>Indic capability</i>						
Indic macro (excl. BharatMath)	accuracy	41.09	46.83	46.63	46.83	+5.74
Indic macro (incl. BharatMath)	accuracy	39.54	46.03	45.66	46.01	+6.47
ARC-Challenge-Indic	accuracy	35.23	48.20	48.20	47.09	+11.86
Belebele	accuracy	37.49	42.17	42.27	42.54	+5.05
BharatMath	accuracy	29.99	41.06	39.70	40.95	+10.96
Global-MMLU-Lite	accuracy	32.50	36.17	36.00	35.75	+3.25
IndicGLUE-selected	score	52.12	57.86	57.22	57.97	+5.85
IndicMMLU-Pro	accuracy	14.44	20.03	20.02	19.44	+5.00
IndicParam	accuracy	28.08	31.59	31.55	31.58	+3.50
MILU	accuracy	40.63	53.54	53.52	53.24	+12.61
Sanskriti	accuracy	84.15	80.60	80.82	80.85	−3.30
FLORES-IN BPB, uncond. ↓	bits/byte	1.0945	0.8272	0.8140	0.8172	−0.2773
FLORES-IN BPB, cond. ↓	bits/byte	0.5962	0.4107	0.4124	0.4181	−0.1781
Indic BPB overall ↓	bits/byte	0.8453	0.6189	0.6132	0.6177	−0.2276
<i>English retention</i>						
Overall, nine-bench mean	mean	70.62	67.98	68.40	67.95	−2.67
ARC-Challenge	likelihood acc	67.66	67.49	67.92	67.15	−0.51
GSM8K, strict	exact match	86.13	84.46	84.46	84.00	−2.13
HellaSwag	likelihood acc	85.57	83.52	83.69	83.69	−1.88
MMLU	likelihood acc	78.50	75.20	75.11	75.28	−3.22
MMLU-Pro	CoT exact match	64.83	54.78	55.43	54.76	−10.07
OpenBookQA	likelihood acc	46.60	43.60	44.80	44.20	−2.40
PIQA	likelihood acc	84.44	83.03	83.46	82.75	−1.69
RACE	likelihood acc	42.39	41.72	42.39	41.53	−0.86
WinoGrande	likelihood acc	79.48	78.06	78.37	78.22	−1.26
HumanEval pass@1	pass@1	31.71	29.88	34.15	33.54	+1.83
<i>Diagnostic</i>						
BharatBBQ acc_norm	acc_norm	35.73	35.04	39.22‡	○	—

BharatMath uses precision-corrected accuracy because boxed-answer emission changes across the training ladder. Phase 1 and long-context accuracy are nearly identical, at 41.06 and 40.95, while the raw numeric-match score and response length move with answer format. This is the same capability-versus-convention pattern discussed in Section 7.3.

Eight of nine Indic proportion metrics improve from the base to the final long-context checkpoint; Sanskriti is the exception. The English mean falls 2.67 points, almost entirely during phase 1, with MMLU-Pro contributing the largest decline. Long-context training adds little further English loss. HumanEval finishes above its base score, while several other tasks recover during annealing before easing again.

F.1 HELMET long-context evaluation

HELMET uses base-model completion prompts, greedy decoding, and seed 42; values are on a 0–100 scale. Recall aggregates the harder retrieval slices and should not be compared directly with the broader RULER mean. Citation and re-ranking are omitted because their structured output format is not elicited by completion prompting. Indic slices were not evaluated at 128K.

Table 31: HELMET evaluation of Evon-Base, the checkpoint that concludes long-context continual pretraining.

Category	8K	16K	32K	64K	128K
Recall	95.9	81.4	57.0	37.3	14.0
RAG	62.7	60.2	56.6	49.4	37.2
ICL	59.2	71.7	78.6	81.8	81.7
LongQA	28.8	38.5	36.5	40.6	26.8
Indic-RAG	57.1	59.1	54.2	46.8	–
Indic-Reader	17.9	17.0	16.0	15.2	–
Indic-Abstention	99.7	99.6	99.0	95.6	–

G SFT Corpus Composition Across Retained Stages

The SFT curriculum was rebuilt across three retained stages. This appendix reports materialized source rows, while Table 32 separately records packed and scheduled exposure.

G.1 Selected checkpoint accounting

The RL initialization is the iteration-2,400 export recorded in the export metadata. At global batch size 256 and sequence length 131,072, it represents

$$2,400 \times 256 \times 131,072 = 80,530,636,800 \quad (13)$$

scheduled sequence positions, including padding. A direct mask measurement estimates that 18.2% are trainable, or roughly 15B loss-bearing tokens. Table 32 gives the exact sample and packing measurements.

The prepared epoch contains 702,824 packed sequences. Sequential bucket boundaries produce 2,752 configured steps, with masked padding in each final partial batch and no wrap-around.

The job reached iteration 2,665; RL uses the iteration-2,400 export. Table 32 records its loss, schedule, endpoint, and checkpoint accounting.

Table 32: Training record for the SFT checkpoint used to initialize RL.

Setting	Value
Public stage	Evon-Instruct
Export source	selected export, verified from export metadata
Scheduled sequence positions through export	80.53B, including padding
Approximate loss-bearing positions	$\sim 15\text{B}$ (18.2%; assistant-only loss mask)
Trainable-rate measurement	95,515 / 524,288 positions over four packed-parquet rows
Intra-pack efficiency	99.98%
Prepared packed-sequence capacity	92.12B positions across 702,824 sequences
Fully scheduled capacity	92.34B positions across 2,752 global batches
Packed rows / scheduled documents	702,824 / 15,160,245
Iterations: configured / recorded / RL export	2,752 / 2,665 / 2,400
Last job-log line / last checkpoint written	2,660 / 2,600
Epoch traversal	sequential bucket blocks; no repeat or wrap-around
Learning-rate schedule	cosine, 5×10^{-7} peak to 1×10^{-7} minimum
Optimizer	Adam, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$, weight decay 0.1
Materialized-row language balance	64.0% English and 36.0% Indic
RL-init loss	0.614

Table 33: Stage totals for the retained SFT corpus chain.

Stage	Materialized rows	Buckets	English	Key character
SFT-phase-1	$\sim 12.5\text{M}$	12	50.7%	Era-1 phase-1 pool; instruction following, mathematics, and code dominate.
SFT-phase-2	$\sim 14.7\text{M}$	14	54.6%	Era-1 phase-2 pool; code, science, and agentic shares increase.
SFT-rebuild-3	$\sim 15.2\text{M}$	15	64.0%	Era-2 rebuild with a larger English share.

Table 34: Materialized-row language mix across the retained SFT chain.

Language	SFT-phase-1	SFT-phase-2	SFT-rebuild-3
English	50.7%	54.6%	64.0%
Hindi	16.5%	15.2%	11.1%
Tamil	6.5%	6.0%	4.8%
Telugu	6.5%	6.0%	4.8%
Kannada	6.5%	6.0%	4.8%
Malayalam	6.5%	6.0%	4.8%
Bengali	3.4%	3.1%	2.7%
Gujarati	1.7%	1.6%	1.5%
Marathi	1.7%	1.6%	1.5%

G.2 Source-family evolution

Table 35 groups the SFT sources into broad families and reports approximate within-stage shares of materialized rows, rounded to the nearest percentage point. English sources are drawn primarily from NVIDIA’s publicly released Nemotron post-training collections [1], alongside Gnani-built Indic dialogue and agentic-trace data. Indic rows in each family include structure-aware translations of the corresponding English sources (Section 6.3).

Table 35: Approximate source-family composition across the retained SFT corpus chain (share of materialized rows).

Source family	SFT-phase-1	SFT-phase-2	SFT-rebuild-3
Mathematics (incl. proofs)	~29%	~28%	~15%
Code (incl. competitive programming)	~24%	~29%	~13%
Instruction following and chat	~28%	~15%	~19%
Science and reasoning	~17%	~19%	~9%
Agentic, tool use, and software engineering	~2%	~9%	~12%
Indic dialogue	–	–	~8%
Safety and domain-specialized	<1%	~1%	~1%
Other curriculum buckets	–	–	~23%

The final SFT-rebuild-3 build is 64.0% English and 36.0% Indic at the materialized-row level. These are not token-level sampling or assistant-loss shares, which may differ from the row distribution.

H SFT Training Traces

The SFT-phase-1, SFT-phase-2, and SFT-rebuild-3 panels below show the retained SFT training stages. Step-local changes in language-model loss reflect changes in curriculum buckets as well as optimization; they should not be interpreted as a single stationary loss curve across the three stages.

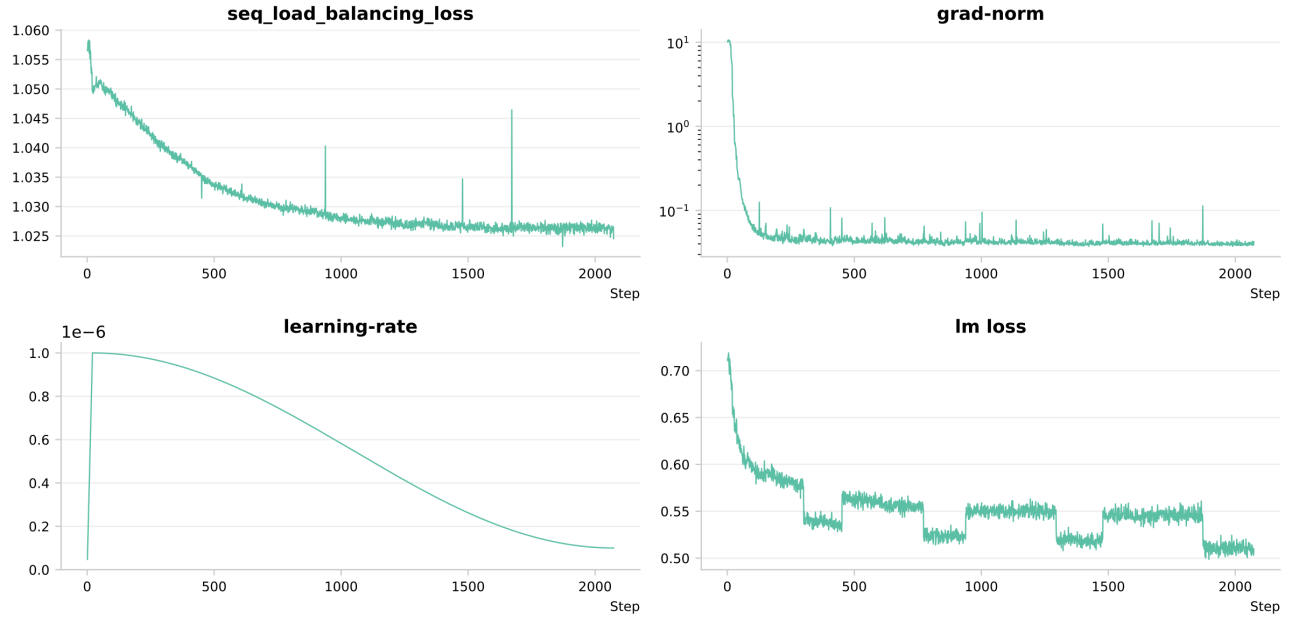
SFT p1

Figure 17: Weights & Biases traces for SFT-phase-1. The panels report the MoE sequence load-balancing loss, gradient norm, cosine learning-rate schedule, and assistant-masked language-model loss. Loss plateaus and transitions align with changes in the sequential curriculum buckets.

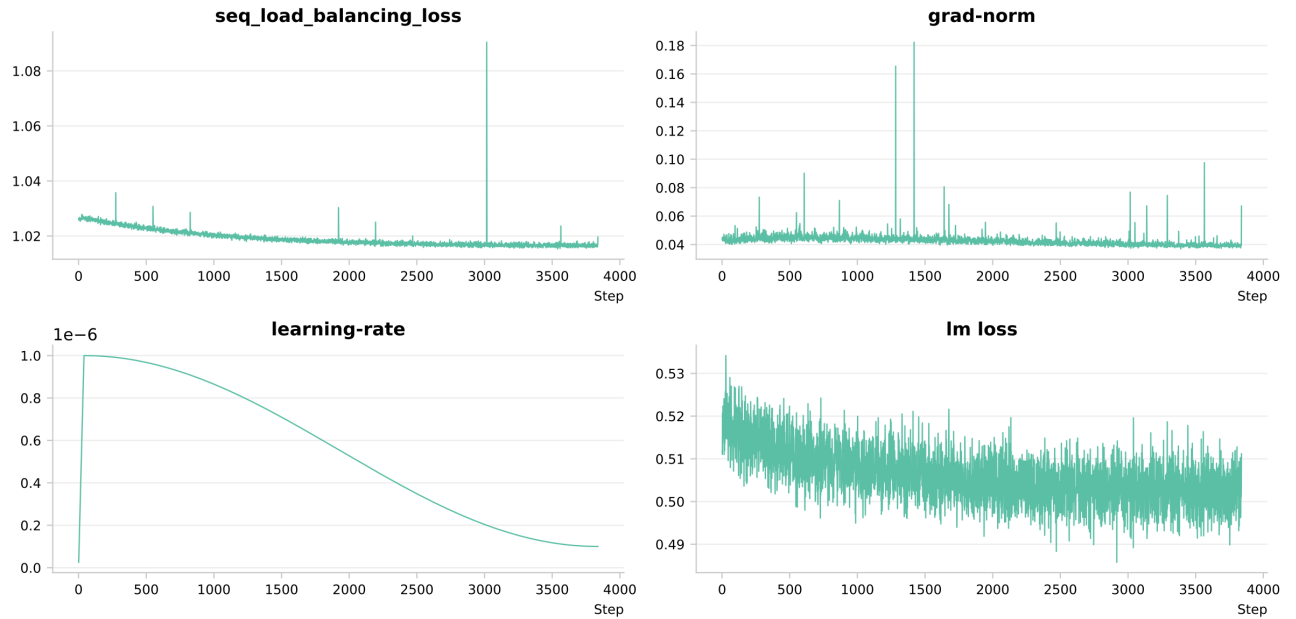
SFT p2

Figure 18: Weights & Biases traces for SFT-phase-2. The assistant-masked loss trends gradually downward under the cosine learning-rate schedule, while isolated gradient-norm and load-balancing spikes remain transient.

SFT r3

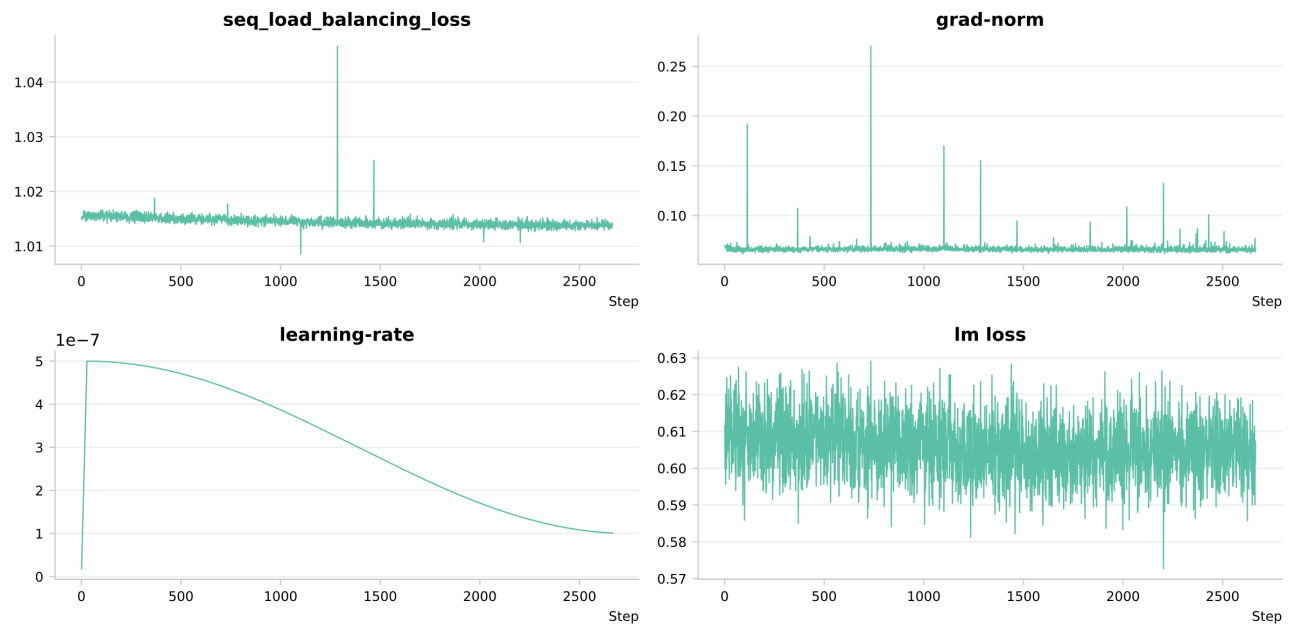


Figure 19: Final SFT-rebuild-3 SFT traces through iteration 2,665; Evon-Instruct lies within this run.

I Matched Evaluation of the Released Checkpoint

Both checkpoints use one matched 35-benchmark campaign. Four extractive-QA rows were rescored with strict exact match after a harness update; F1 was unaffected and is used in the narrative. Deltas are RL minus SFT. Accuracy metrics are percentages, while translation metrics use native sacreBLEU units. BharatMath is omitted pending a harness revision; ↓ means lower is better.

Table 36: Matched evaluation of Evon-Instruct and the released Evon-RL-Gen checkpoint.

Benchmark	Metric	Final SFT ckpt	RL ckpt step 40	Δ (RL – SFT)
<i>Math & Reasoning</i>				
MATH	exact match	48.69	84.76	+36.07
MATH-500	exact match	48.80	84.00	+35.20
GSM8K	exact match	85.75	94.31	+8.56
MGSM-English	exact match	91.60	98.40	+6.80
MGSM-Indic	exact match	79.60	84.60	+5.00
AIME 2025	pass@1	72.92	74.38	+1.46
AIME 2026	pass@1	72.08	74.58	+2.50
Beyond AIME	pass@1	48.50	53.56	+5.06
HMMT February 2025	pass@1	59.58	60.83	+1.25
HMMT November 2025	pass@1	58.54	63.33	+4.79
GPQA Diamond	pass@1	38.19	65.03	+26.84
<i>Code</i>				
LiveCodeBench	pass@1	51.35	61.27	+9.92
MBPP+	pass@1	75.24	78.78	+3.54
HumanEval+	pass@1	88.66	88.54	−0.12
<i>Knowledge and MCQ</i>				
MMLU (exact match) [†]	exact match	77.21	86.68	+9.47
MMLU-Pro	accuracy	66.67	77.26	+10.59
IndicMMLU-Pro	accuracy	47.07	68.88	+21.81
MILU	accuracy	73.16	78.33	+5.17
ARC-Challenge	exact match	93.26	95.90	+2.64
AGIEval English	exact match	60.02	73.47	+13.45
BhashaBench-Multi	accuracy	42.06	49.22	+7.16
<i>Instruction Following</i>				
IFEval	prompt strict	66.36	82.81	+16.45
IFEval	prompt loose	67.65	83.73	+16.08
IFEval	instruction strict	73.26	87.17	+13.91
IFEval	instruction loose	74.22	87.77	+13.55
IFEval-Hindi	prompt strict	52.71	50.59	−2.12
IFEval-Hindi	prompt loose	53.42	51.30	−2.12
IFEval-Hindi	instruction strict	60.51	59.64	−0.87
IFEval-Hindi	instruction loose	61.62	60.27	−1.35

Continued on next page

Table 36: continued

Benchmark	Metric	Final SFT ckpt	RL ckpt step 40	Δ (RL – SFT)
IndicFEval-translated	prompt strict	37.51	49.87	+12.36
IndicFEval-translated	prompt loose	38.88	50.87	+11.99
IndicFEval-translated	instruction strict	47.68	61.65	+13.97
IndicFEval-translated	instruction loose	49.04	62.43	+13.39
IndicFEval-ground	prompt/ instruction strict	40.52	33.05	−7.47
IndicFEval-ground	prompt/ instruction loose	43.59	33.43	−10.16
IFBench	prompt strict	21.67	44.33	+22.66
IndicParam	accuracy	32.94	39.28	+6.34
<i>QA and Translation</i>				
XQuAD-English (EM)	exact match	46.97	63.70	+16.73
XQuAD-English (F1)	F1	67.15	78.86	+11.71
XQuAD-Indic (EM)	exact match	18.84	12.86	−5.98
XQuAD-Indic (F1)	F1	36.51	17.19	−19.32
XorQA-Indic (EM)	exact match	6.67	35.24	+28.57
XorQA-Indic (F1)	F1	16.16	44.09	+27.93
IndicQA (EM)	exact match	12.77	12.22	−0.55
IndicQA (F1)	F1	26.36	17.72	−8.64
FLORES-IN	chrF++	14.55	36.33	+21.78
FLORES-IN	BLEU	3.64	14.94	+11.30
CrossSum-IN (chrF)	chrF	7.00	12.81	+5.81
<i>Bias diagnostic</i>				
BBQ	accuracy (disamb.)	94.13	97.70	+3.57
BBQ	accuracy (amb.)	92.29	88.69	−3.60
BBQ	bias score (disamb.) ↓	0.99	0.60	−0.39
BBQ	bias score (amb.) ↓	4.33	7.39	+3.06

[†]MMLU here is generative exact match and is not comparable to the loglikelihood accuracy reported in Table 30.

The MATH gain combines emission and conditional accuracy. In raw responses, boxed-answer emission rises from 2,911 to 4,612 of 5,000 problems, while boxed conditional accuracy rises from 86.4% to 92.2%. Emission explains 81–83% of the gain, depending on decomposition order. Section 7.3 instead partitions after reasoning is stripped, matching the live grader. These are two definitions applied to the same campaign and should not be mixed.

MATH emission decomposition. The main text partitions responses after reasoning is removed. Partitioning the raw response instead finds 150 additional boxed SFT outputs without changing the 2,516 correct boxed answers. The resulting denominators describe different conventions and must not be mixed.

J Indic Extractive-QA Failure Taxonomy

Section 7 attributes the Indic extractive-QA regression to an unrewarded output convention rather than to a loss of reading ability. This appendix records the item-level evidence.

Each response is assigned to one bin, in order: **B0**, no answer; **B1b**, English where the gold span is Indic; **B2**, a gold-equivalent answer recovered only by prose-tolerant regrading; **B3**, a target-script answer that is neither gold-equivalent nor a verbatim passage span; **B4**, a wrong passage span; and **OK** otherwise.

Table 37 reports the four largest bins, so rows do not sum to 100%. No response is empty, and the omitted bins contribute at most three percent per row. Shares are equal-weight macros over eight languages and should not be pooled across languages. This slice differs from the eleven-leaf scorecard macro in Table 36.

Table 37: XQuAD-Indic failure taxonomy, equal-weight macro over eight languages.

Checkpoint	Macro F1	B1b	B2	B3	OK
Evon-Instruct (SFT initialization)	40.38	8.7%	5.8%	62.6%	21.6%
Released Evon-RL-Gen	16.59	91.7%	0.1%	2.5%	2.8%
Indic-aware sibling (Sec. K)	53.43	15.3%	6.0%	45.4%	31.9%

IndicIFEval also varies across evaluation states. The FP8-paired BF16 row below is the canonical release-protocol reading used in the main text; the earlier rows are retained as sensitivity checks.

Table 38: Recorded IndicIFEval sensitivity across evaluation states.

Measurement	Ground strict	Translated strict	Ground in-script
Initial matched evaluation	28.66	47.45	5.4%
Dedicated IndicIFEval evaluation*	63.88	—	54.2%
FP8-paired BF16 baseline	33.05	49.87	—

*The loose-criterion score is 65.62.

IndicQA reproduces the language shift: B1b rises from 9.5% to 73.0%. A prose-tolerant regrade moves macro F1 by at most 0.24 points, so extraction alone does not explain the loss.

The taxonomy has two limits. B3 is neither gold-equivalent nor verbatim, but it is not independently classified as ungrounded because incorrect Indic answers rarely reproduce a passage span exactly. Likewise, OK means that the response contains the gold span, not necessarily that it is only the gold span. These distinctions matter most for the sibling branch, whose answers are usually in the target script.

Conditioning on target-script output separates emission rate from answer quality. Table 39 shows that the script rate collapses while conditional EM and F1 rise on both benchmarks. This is consistent with preserved reading ability and a changed output convention, although it is not a controlled estimate.

Table 39: Target-script rate and conditional accuracy before and after released RL.

Benchmark	Target-script rate		Conditional EM		Conditional F1	
	SFT	RL	SFT	RL	SFT	RL
XQuAD-Indic	89.9	5.4	23.22	51.50	37.69	61.19
IndicQA	86.9	24.3	20.24	45.23	27.00	50.54

Target-script answers form a self-selected subset, so the conditional rise may reflect selection. We use it as evidence against broad reading collapse, not as a capability estimate.

The sibling study’s Indic-aware checkpoint restores F1 above the initialization while its own B1b share, at 15.3%, remains above the initialization’s 8.7%. Rewarding target-script output recovered the task score without fully restoring the convention.

K Sibling Indic-Aware RL Study (Not Released)

K.1 Branch definition and reward

Evon-RL-Indic is a sibling research branch, not the public Evon v3.3 checkpoint and not a continuation of the released RL run. It initializes independently from the same Evon-Instruct iteration-2,400 checkpoint and adds a fifth environment, `indic_if`, designed for Indic instruction following. Its reward is

$$R = R_{\text{task}} + 0.3R_{\text{lang}} - 0.0R_{\text{len}} - 0.3R_{\text{rep}}, \quad (14)$$

Malformed or abandoned answers score -0.25 . The branch also floors the advantage-normalization denominator at 0.05, preventing tiny reward differences from becoming large updates.

R_{task} is the fraction of constraints satisfied. R_{lang} measures the share of answer words written in the row’s target script. R_{rep} detects repeated n -gram bursts and character runs across the full generation. The length term is disabled in the experimental run.

The scopes are deliberate. Language compliance is computed only over the answer after the final reasoning block, because English internal reasoning with an Indic user-visible answer is acceptable. Repetition is computed over the entire generation, because looping reasoning is undesirable in any language.

The reward is target-aware rather than “Indic is always good”. English examples target Latin script; Hindi targets Devanagari; Bengali targets Bengali script; and so on. A language-identification gate helps distinguish languages that share a script. Unknown language labels cause the language reward to be skipped rather than guessed.

K.1.1 Why the experimental mix retains an English guardrail pool

The final mix diverts 1,509 English rows, or 9.9% of the input, from the existing task environments into a shared guardrail pool. These rows are not an additional English corpus.

This guardrail is essential. An Indic-only reward taught the model to answer English prompts in Indic scripts. The English rows instead make “use the requested language” learnable.

Directive ablations show that the literal language name is the active cue: explicit Hindi directives rise from 38.4% to 90.5% in-script, while implicit forms remain near zero. The training prompts therefore vary wording and placement but retain explicit language supervision.

K.1.2 Experimental mixture and configuration

The experimental run reinitializes from Evon-Instruct and trains for 40 steps over 15,300 prompts. Table 40 and the configuration table give the mixture and rollout details.

The input proportions are cap-driven rather than specified as a target weight vector. Each rated environment is first filtered to a difficulty band and then capped; the final proportions are the row counts that survive those operations. Mathematics dominates because it has the largest eligible pool. `indic_if` and MCQA retain unrated rows and are therefore pool-size limited rather than difficulty selected. Logic environments are excluded from this experimental mix.

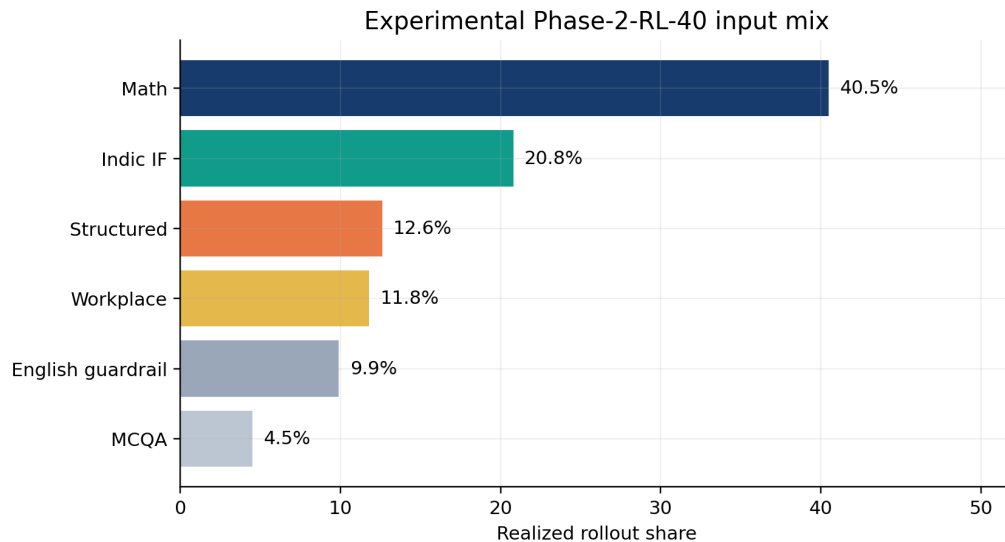


Figure 20: Experimental Evon-RL-Indic training-row composition after difficulty filtering, environment caps, and English-row diversion. This mixture is not used by the released model.

Table 40: Cap-driven composition of the experimental Evon-RL-Indic input mix. Component shares are rounded to one decimal place and therefore sum to 100.1%; row counts sum exactly to 15,300.

Pool	Rows	Final share
Mathematics	6,192	40.5%
Indic instruction following	3,179	20.8%
Structured outputs	1,932	12.6%
Workplace assistant	1,804	11.8%
English guardrail	1,509	9.9%
MCQA	684	4.5%
Total	15,300	100%

These row shares are not effective gradient shares: dynamic sampling removes all-pass and all-fail groups, so environment contributions must be measured from the surviving rollouts.

Table 41: Experimental Evon-RL-Indic configuration.

Setting	Value
Algorithm	GRPO with leave-one-out baseline and whole-group standard-deviation normalization
Inherited estimator flags	<code>use_leave_one_out_baseline=true; normalize_rewards=true</code>
Generations per prompt	8
Dynamic sampling	enabled
Sampling temperature	1.0
Training steps	40
Training rows / validation rows	15,300 / 256
Recorded rollouts	251,906
Maximum generated tokens	16,384
Maximum RL sequence length	49,152
Model context capability	remains 131,072
Advantage standard-deviation floor	0.05
Checkpoint cadence	every 5 steps

The 49,152-token RL cap is an optimization limit for rollout collection; it does not alter the model’s trained or exported 131,072-token context capability.

Aggregate validation reward rises from 0.320 at initialization to 0.649 at step 40. Indic instruction following and structured output contribute most of the gain; mathematics peaks at step 30 before easing. These branch-specific rewards are not directly comparable with the released run.

Sibling Indic-aware RL validation trajectory

Recorded validation means from initialization through step 40; this branch is not released.

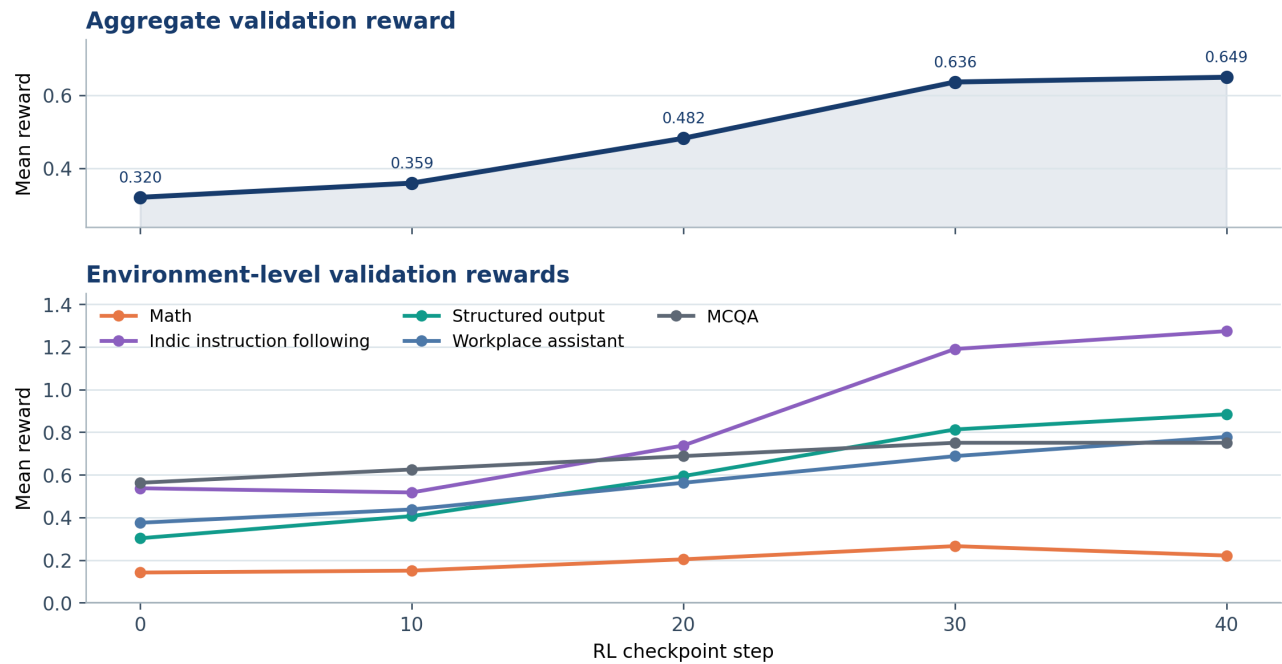


Figure 21: Validation rewards for the Indic-aware branch. These optimization diagnostics are not benchmark accuracies.

K.1.3 Experimental matched results

Evon-RL-Indic is the first experimental artifact in this programme to improve reasoning and Indic instruction following simultaneously. It is reported as evidence about reward design, not as a result for the released Evon v3.3 checkpoint.

Both columns come from the campaign in Table 36, allowing direct comparison. FLORES-IN chrF++ uses native sacreBLEU units, not percentages.

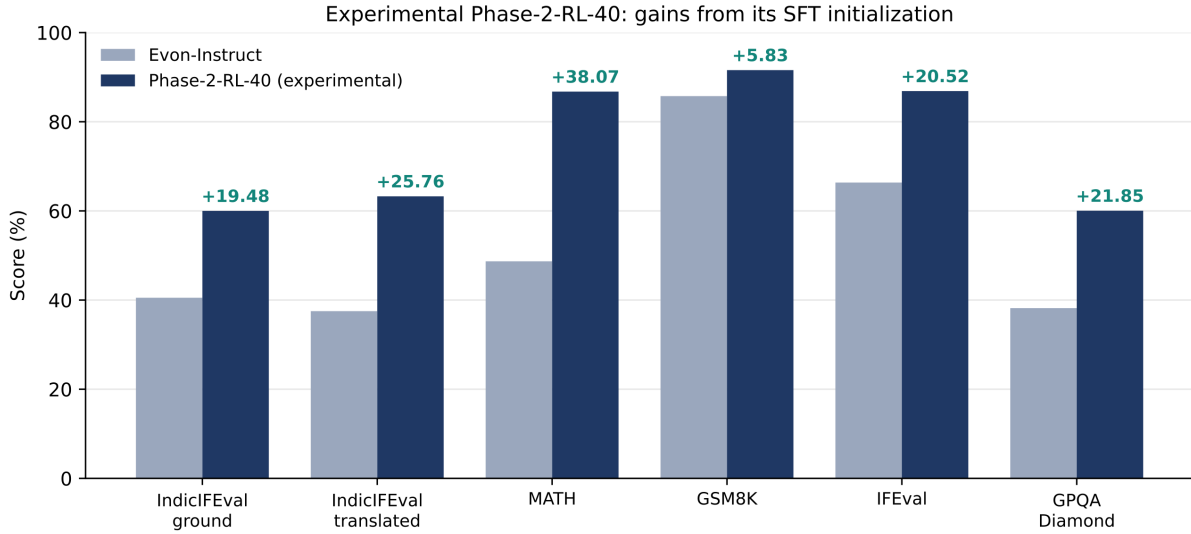


Figure 22: Selected Evon-Instruct to experimental Evon-RL-Indic results. Deltas are percentage points; the checkpoint is not released.

Table 42: Matched evaluation of the unreleased Indic-aware checkpoint against its SFT initialization.

Benchmark	Evon-Instruct	Evon-RL-Indic	Δ
IndicIFEval-ground, prompt strict	40.52	60.00	+19.48
IndicIFEval-translated, prompt strict	37.51	63.27	+25.76
XQuAD-Indic, F1	36.51	50.44	+13.93
MATH	48.69	86.76	+38.07
GSM8K	85.75	91.58	+5.83
IFEval, prompt strict	66.36	86.88	+20.52
GPQA Diamond	38.19	60.04	+21.85
FLORES-IN, chrF++	14.55	39.37	+24.82
Native-Indic in-script rate [†]	99.9	98.7	-1.2

[†]In-script rate is a separate measurement and is not part of the benchmark campaign above.

Relative to the released checkpoint, the largest gains are on IndicIFEval-ground (+26.95 prompt-strict) and XQuAD-Indic (+33.25 F1). The table above measures the same benchmark against Evon-Instruct rather than against the released checkpoint, which is why its figure differs. Figure 23 shows the remaining trade-offs.

Native-script retention is not uniform across prompt types. On the translated IndicIFEval split, in-script output falls from 95.1% for Evon-Instruct to 84.6% for the experimental checkpoint, even as task accuracy improves. This branch is therefore evidence for the reward-design mechanism, not a claim that every language-control problem is solved.

Step 40 is selected because it retains the step-30 mathematics gain while improving several other tracked tasks. Mathematics is well defended because it has the largest share of the mixture.

K.1.4 Direct comparison with the released checkpoint

The two branches are compared under one matched configuration, with strict exact match on both sides. Their 33-metric means are effectively equal, but the equality hides opposite trade-offs. The Indic-aware branch makes large gains on the instruction-following rows it targets, while the released checkpoint has more numerous, smaller advantages elsewhere. Removing the five targeted instruction-following rows gives the released checkpoint a 0.0257 lead. Figure 23 shows the largest differences.

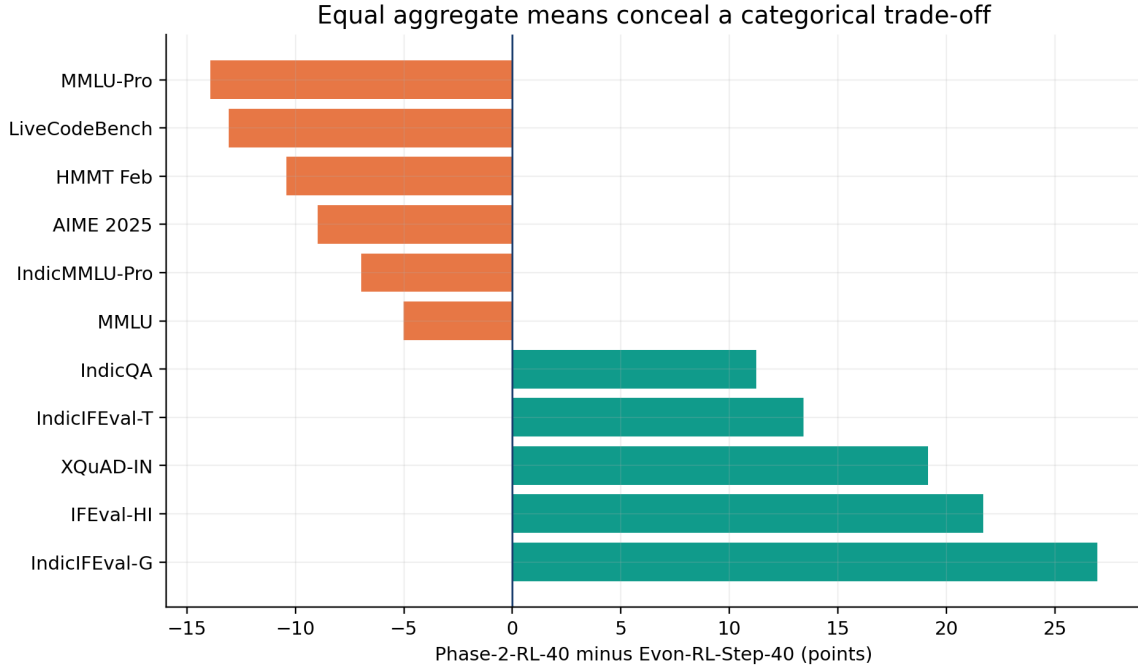


Figure 23: Selected matched differences between experimental Evon-RL-Indic and released Evon-RL-Gen. Positive values favor the experimental checkpoint.

Evon-RL-Indic leads on instruction following, translation, and several Indic tasks. The released checkpoint remains stronger on reasoning, code, and English knowledge.

The same trade-off appears in agentic evaluation: the branch improves the TauBench tasks but falls on SWE-bench Verified (Table 43).

Table 43: Additional released-to-experimental agentic comparison. Deltas are percentage points.

Benchmark	Evon-RL-Gen	Evon-RL-Indic	Δ
MultiChallenge	0.4925	0.5000	+0.75
TauBench airline	0.3733	0.4400	+6.67
TauBench retail	0.1520	0.2807	+12.87
TauBench telecom	0.0936	0.3626	+26.90
SWE-bench Verified	0.2280	0.1220	-10.60

K.1.5 Experimental tool-use decomposition

On BFCL v4, the overall 17-category score changes from 53.81 for SFT to 51.16 after RL. By itself, that suggests a general tool-use regression, but the category breakdown reveals a narrower issue.

Table 44: BFCL v4 decomposition.

Category	Evon-Instruct	Evon-RL-Indic
Headline, renormalized over 17 categories	53.81	51.16
Parallel / parallel-multiple	69.5 / 80.0	0.5 / 0.5
Live parallel / live parallel-multiple	75.0 / 75.0	0.0 / 0.0
Simple Python / multiple	86.8 / 81.0	94.5 / 93.0
Multi-turn base	34.0	58.0

The experimental policy usually emits one tool call when the benchmark expects two in the same assistant message. Its syntax parses correctly, and it issues the missing call on a second turn in 95 of 96 examined cases. The model retains the functional ability but has learned a one-action-per-turn policy from an environment that always follows a call with an observation.

L Reinforcement-Learning Configuration, Side by Side

This appendix sets the released run beside the Indic-aware branch of Section 9. The two share an initialization, an optimizer, a length budget, and every KL and clipping setting. They differ in how prompts are batched, and in what the reward mixture contains.

The saved branch configuration names 30 nodes, while the run itself used 16. It also carries three corrections absent from the released run. First, it clamps the advantage-normalization divisor at $\max(\sigma, 0.05) + 10^{-6}$. Without this floor, tiny reward differences can become full-sized updates.

Second, multi-turn rows are reduced from 37% to 11% of the Indic instruction pool. Third, empty content is treated as malformed rather than merely unrewarded.

M Reward Model and Distillation Study (Not Released)

This appendix records the reward-model, distillation, and abandoned-run configurations behind Sections 9.2 and 9.3. None belongs to the released lineage.

M.1 Preference data

Prompts combine native Indic instruction data with HelpSteer3-style English prompts translated into eight Indian languages, in a 70:30 translated-to-native ratio. Two models generate one candidate response each, and Gemini Flash labels the preference on a seven-point scale. HelpSteer3 contributes the format and protocol, not its data. Of 98,744 pairs in Hindi, Telugu, Tamil, and Malayalam, 99.6% were labeled.

Table 45: Training configuration for the released run and the Indic-aware branch, both read at step 40. “Same” means the two runs carry an identical setting.

Setting	Evon-RL-Gen (released)	Evon-RL-Indic (branch)
<i>Optimizer and learning rate</i>		
Optimizer	Adam, distributed, BF16, weight decay 0.01	Same
Peak learning rate	5×10^{-6}	Same
Minimum learning rate	5×10^{-7}	Same
Schedule	constant, 13-iteration warmup from the floor	Same
Gradient clipping	1.0	Same
Learning rate logged at step 40	5×10^{-6}	5×10^{-6}
<i>Batching</i>		
Prompts per step	128	255
Generations per prompt	16	8
Nominal rollouts per step	2,048	2,040
Training global batch	2,048	2,040
Training micro-batch	4	4
Generation batch	32	32
Retained samples at step 40	2,048, with 238 discarded	2,040, with 610 discarded
Dynamic sampling	on, up to 10 extra generation batches	Same
Cluster in the saved configuration	32 nodes, 8 GPUs each	30 nodes, 8 GPUs each
<i>KL, clipping, and advantage</i>		
PPO ratio clip	0.2 either side	Same
KL penalty in the loss	0	Same
KL penalty in the reward	off	Same
KL estimator	k_3 , input clamp 20, output clamp 10	Same
Advantage	GRPO, leave-one-out baseline, normalized	Same
KL penalty logged at step 40	0.0	0.0
<i>Lengths</i>		
Maximum total sequence	49,152	Same
Maximum new tokens per generation	16,384	Same
Rollout-engine context limit	49,152	Same
Sequence-packing tile	98,304 tokens	Same
Generated tokens at step 40, median	2,527	2,220
Generated tokens at step 40, 95th percentile	10,276	11,781

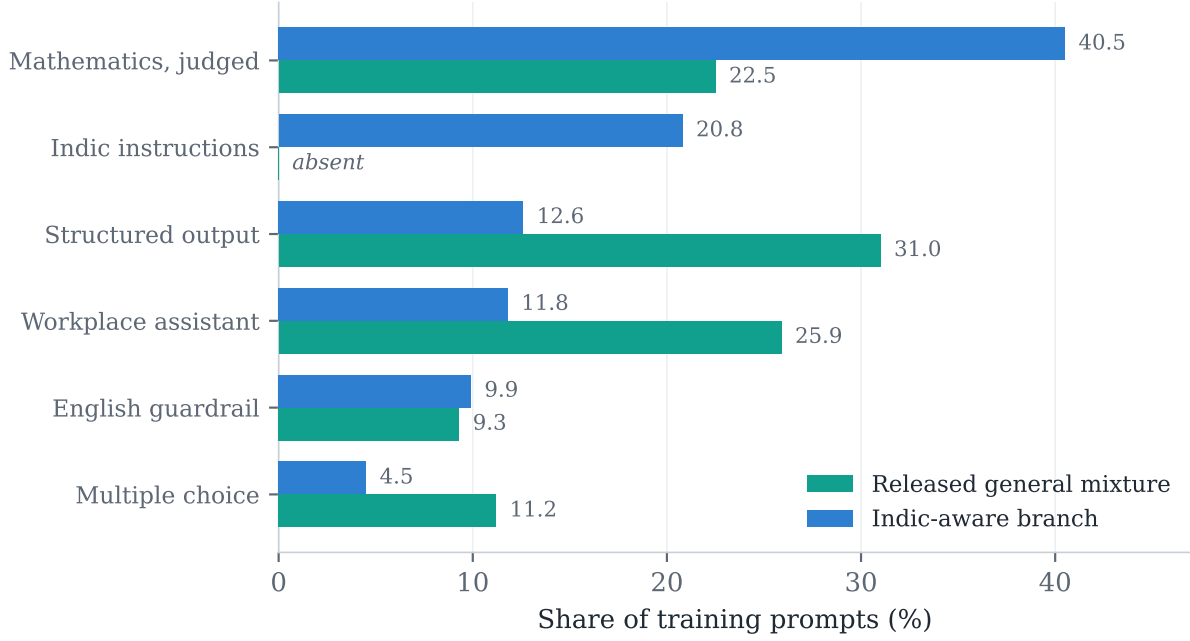


Figure 24: Nominal environment shares before dynamic sampling for the released and Indic-aware runs.

M.2 The judge’s reward

Initialized from Gemma 4 26B-A4B with its vision tower frozen, the judge emits two absolute helpfulness scores and a forced relative ranking. GRPO rewards the correct side, calibration, and agreement between the outputs; Table 46 gives the terms.

Table 46: Reward terms used to train the generative judge.

Term	Reward
Correct relative ranking	+4
Calibration	$-1.2H_{1.5}(d_r) - 0.6[H_1(d_{h_1}) + H_1(d_{h_2})]$
Absolute/relative inconsistency	-0.5
Unparseable verdict	-8
Parseable range	clipped to $[-6, 4]$, then adjusted for batch-level rank balance

Here H_δ is Huber loss, $d_{h_k} = \hat{h}_k - h_k$, and $d_r = \hat{r} - r$.

The format penalty lies below the clip floor, so every parseable verdict beats every unparseable one. Calibration also preserves a gradient when all samples choose the same side, which occurs in 78% of groups. Advantages are therefore centered but not divided by group standard deviation, avoiding the amplification of small calibration differences. Training uses 262,834 examples, with each pair shown in both orders; 72.4% come from the Indic set.

Calibration, rather than accuracy, continues to improve. The rank-balance term reduces total variation distance between predicted and target rank histograms from 0.327 to 0.083 without lowering accuracy. Step 100 is exported; later checkpoints regress on both measures.

RM-Bench uses pairwise comparisons, so its scores are not directly comparable with scalar reward-model scores. JudgeBench is a smaller capped-generation subset. The largest margin below is on the hard slice, where style and substance point to different responses.

Table 47: The 26B-A4B judge against a 235B-parameter generative reward model.

Model	RM-Bench								JudgeBench
	Chat	Math	Code	Safety	Easy	Normal	Hard	Overall	
Qwen3-Nemotron-235B-A22B-GenRM	76.5	96.9	81.4	94.4	94.0	90.5	77.4	87.3	87.4
Ours, 26B-A4B	76.1	93.9	84.4	95.7	94.1	92.0	86.4	90.8	88.3

M.3 Reinforcement learning against the judge

Each prompt samples $G = 16$ completions. Comparing neighbors around a ring requires 16 judge calls instead of all 120 pairs. Every completion appears once in each position, cancelling position bias, and the forced ranking prevents zero-variance groups. Reasoning is hidden from the judge. The policy reward also includes script consistency, repetition control, and a stochastic reasoning-length term.

Table 48: Selected gains from policy optimization against the judge, relative to Evon-RL-Indic.

Benchmark	Change
LiveCodeBench	+13.2
MMLU-Pro	+8.8
IndicMMLU-Pro	+6.3
MMLU	+5.8
GPQA Diamond	+5.8
AGIEval	+5.5
Grounded Indic instruction following	+4.4
Mathematics aggregate	+2.7

Evaluation and distillation use step 20 to limit drift. Step 20 is not the highest-scoring judge checkpoint, and the same regressions remain visible later, so this choice does not create them.

M.4 Length control

Three designs were tested. Rewarding the shortest trace drove reasoning to zero, while a two-sided ramp reduced it by 60% and cost more than 25 AIME points. The adopted one-sided penalty preserves judge quality without attracting the policy to an artificial length target.

M.5 The 70:30 mixture that failed

The recipe was retrained with 30% of prompts scored by binary mathematics and multiple-choice verifiers. For a problem solved with probability p , a group of $G = 16$ is degenerate with probability $z(p) = p^{16} + (1 - p)^{16}$. Dynamic sampling discards such all-pass and all-fail groups. The measured degenerate rate was high enough to trigger a second generation batch on 23 of 29 steps, making each step 2.27 times slower than the judge-only baseline.

All judge groups survived, while 35–49% of verifier groups were discarded. The specified 70:30 split therefore became roughly 80:20 at the gradient, and the run was stopped after thirty steps. About 40% of the verifier pool contained problems the model almost always solved or almost always missed. Restricting the pool to intermediate pass rates would reduce the discard probability by more than an order of magnitude. The original monitor missed this because it inspected only the judge environment, where groups cannot degenerate.

M.6 Distillation configuration

The teacher is Evon-RL-Indic, and the student starts from judge-training step 20. At each student token, supervision uses the teacher’s top-64 next-token distribution under Equation 8, with $w = 0.5$. The 29,988 mathematics and multiple-choice prompts cover English and eight Indian languages, with source-problem splits preventing translated leakage into validation. Training runs for 100 steps at a constant learning rate of 10^{-5} ; only step 20 is evaluated.

N Matched BF16 and FP8 Benchmark Results

Table 49: All 32 paired BF16 and FP8 accuracy results. Deltas are FP8 minus BF16.

Benchmark	BF16	FP8	Δ
AGIEval English	0.7347	0.7440	+0.0093
AIME 2025	0.7438	0.7500	+0.0062
AIME 2026	0.7458	0.7729	+0.0271
ARC-Challenge	0.9590	0.9582	−0.0008
BBQ	0.8869	0.8819	−0.0050
Beyond-AIME	0.5356	0.5337	−0.0019
BharatMath	0.7317	0.7370	+0.0053
BhashaBench-Multi	0.4922	0.4916	−0.0006
GPQA-Diamond	0.6503	0.6566	+0.0063
GSM8K	0.9431	0.9378	−0.0053
HMMT February	0.6083	0.6438	+0.0355
HMMT November	0.6333	0.6354	+0.0021
HumanEval+	0.8854	0.8756	−0.0098
IFEval	0.8281	0.7930	−0.0351
IFEval-Hindi	0.5059	0.4965	−0.0094
IndicIFEval grounded	0.3305	0.3290	−0.0015
IndicIFEval translated	0.4987	0.4997	+0.0010
IndicMMLU-Pro	0.6888	0.6882	−0.0006
IndicParam	0.3928	0.3939	+0.0011
IndicQA	0.1222	0.1320	+0.0098
LiveCodeBench v6	0.5774	0.5793	+0.0019
MATH	0.8476	0.8451	−0.0025
MATH-500	0.8400	0.8380	−0.0020
MBPP+	0.7878	0.7757	−0.0121
MGSM-English	0.9840	0.9680	−0.0160
MGSM-Indic	0.8460	0.8520	+0.0060
MILU	0.7833	0.7825	−0.0008
MMLU	0.8668	0.8719	+0.0051
MMLU-Pro	0.7726	0.7739	+0.0013
XorQA-Indic	0.3524	0.3528	+0.0004
XQuAD-English	0.6370	0.6378	+0.0008
XQuAD-Indic	0.1286	0.1321	+0.0035

O Glossary

BPE Byte-pair encoding, an iterative method that learns frequently occurring token merges.

CPT Continual pretraining, further next-token training of an existing base model.

CP Context parallelism, which partitions a long sequence across devices.

EP Expert parallelism, which partitions MoE experts across devices.

Fertility Average tokenizer tokens produced per whitespace-delimited word.

Constituent-mean initialization Initialization of a new token row from the old-token pieces that exactly compose it.

GQA Grouped-query attention, in which multiple query heads share fewer key/value heads.

GRPO Group relative policy optimization, which estimates relative advantages from several completions of the same prompt.

In-script rate Fraction of outputs written in the target writing system.

MoE Mixture of experts, a sparse layer that routes each token through a subset of expert networks.

NoPE No explicit positional embedding.

SFT Supervised fine-tuning on prompt/response or conversational demonstrations.

TP Tensor parallelism, which partitions individual tensors and matrix operations across devices.

WSD Warmup–Stable–Decay learning-rate schedule.

References

- [1] NVIDIA. *NVIDIA Nemotron 3 Nano Technical Report*. 2025. <https://research.nvidia.com/labs/nemotron/files/NVIDIA-Nemotron-3-Nano-Technical-Report.pdf>
- [2] The Soofi-Team. *A Sovereign, Open-Source Foundation Model for German and English: Soofi S Pretraining Report v1.0*. arXiv:2607.09424, 2026.
- [3] Jimmy T. H. Smith, Tarek Dakhraan, Alberto Cabrera, 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 Tokenization and Adaptive Data Mixtures*. arXiv:2508.06533, 2025.
- [5] Jeffrey Li, Alex Fang, Georgios Smyrnis, et al. *DataComp-LM: In Search of the Next Generation of Training Sets for Language Models*. arXiv:2406.11794, 2024.
- [6] Zhihong Shao, Peiyi Wang, Qihao Zhu, et al. *DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models*. arXiv:2402.03300, 2024.
- [7] Tri Dao and Albert Gu. *Transformers are SSMS: Generalized Models and Efficient Algorithms Through Structured State Space Duality*. arXiv:2405.21060, 2024.
- [8] NVIDIA Corporation. *NVIDIA Model Optimizer*. 2024–2026. <https://github.com/NVIDIA/Model-Optimizer>
- [9] Gnani.AI LLM Team. *Gnani-Evon-v3.3-30B-A3B Model Card and Evaluation*. 2026. <https://huggingface.co/gnani/gnani-evon-v3.3-30B-A3B>
- [10] Mohd Arshad Khan et al. *IndicLLMSuite: A Blueprint for Creating Pre-training and Fine-tuning Datasets for Indian Languages*. ACL 2024. <https://aclanthology.org/2024.acl-long.843/>
- [11] Christopher Toukmaji and Jeffrey Flanigan. *A Systematic Study of Low-Resource Language Adaptation for Large Language Models*. GEM Workshop, 2025. <https://aclanthology.org/2025.gem-1.61/>
- [12] Ahmet Üstün et al. *Aya Model: An Instruction Finetuned Open-Access Multilingual Language Model*. ACL 2024. <https://aclanthology.org/2024.acl-long.845/>
- [13] Niklas Muennighoff et al. *Crosslingual Generalization through Multitask Finetuning*. ACL 2023. <https://aclanthology.org/2023.acl-long.891/>
- [14] Pinzhen Chen et al. *Monolingual or Multilingual Instruction Tuning: Which Makes a Better Alpaca? Findings of EACL 2024*. <https://aclanthology.org/2024.findings-eacl.90/>
- [15] Yihao Li, Jiayi Xin, Miranda Muqing Miao, Qi Long, and Lyle Ungar. *The Impact of Language Mixing on Bilingual LLM Reasoning*. EMNLP 2025, pages 32531–32548. <https://aclanthology.org/2025.emnlp-main.1654/>
- [16] Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. *On the Cross-lingual Transferability of Monolingual Representations*. ACL 2020, pages 4623–4637. <https://aclanthology.org/2020.acl-main.421/>

- [17] Tarun Jayakumar, Mohammed Safi Ur Rahman Khan, Raj Dabre, Ratish Puduppully, and Anoop Kunchukuttan. *IndicIFEval: A Benchmark for Verifiable Instruction-Following Evaluation in 14 Indic Languages*. arXiv:2602.22125, 2026.
- [18] Harman Singh et al. *IndicGenBench: A Multilingual Benchmark to Evaluate Generation Capabilities of LLMs on Indic Languages*. ACL 2024. <https://aclanthology.org/2024.acl-long.595/>
- [19] Freda Shi, Mirac Suzgun, Markus Freitag, et al. *Language Models are Multilingual Chain-of-Thought Reasoners*. ICLR 2023. <https://arxiv.org/abs/2210.03057>
- [20] Dan Hendrycks, Collin Burns, Steven Basart, et al. *Measuring Massive Multitask Language Understanding*. ICLR 2021. <https://arxiv.org/abs/2009.03300>
- [21] Dan Hendrycks, Collin Burns, Saurav Kadavath, et al. *Measuring Mathematical Problem Solving with the MATH Dataset*. NeurIPS Datasets and Benchmarks, 2021. <https://arxiv.org/abs/2103.03874>
- [22] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, et al. *Training Verifiers to Solve Math Word Problems*. arXiv:2110.14168, 2021.
- [23] David Rein, Betty Li Hou, Asa Cooper Stickland, et al. *GPQA: A Graduate-Level Google-Proof Q&A Benchmark*. arXiv:2311.12022, 2023.
- [24] Mark Chen et al. *Evaluating Large Language Models Trained on Code*. arXiv:2107.03374, 2021.
- [25] Naman Jain, King Han, Alex Gu, et al. *LiveCodeBench: Holistic and Contamination Free Evaluation of Large Language Models for Code*. arXiv:2403.07974, 2024.
- [26] Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, et al. *Instruction-Following Evaluation for Large Language Models*. arXiv:2311.07911, 2023.