# MINT: A Multilingual Indic Neural Transformer for Abstractive Summarization Under Memory Constraints

Sameer Kumar Singh *

Posted Date: 14 April 2026

doi: 10.20944/preprints202604.0940.v1

Keywords: multilingual summarization; Indic NLP; low-resource NLP; transformer; abstractive summarization; ROUGE evaluation; efficient NLP

 doi:10.20944/preprints202604.0940.v1



*Article*

# MINT: A Multilingual Indic Neural Transformer for Abstractive Summarization Under Memory Constraints

**Sameer Kumar Singh**

Independent Researcher, Lucknow, India; sameerkumarsingh56@gmail.com

**Abstract**

We present MINT (**M**ultilingual **I**ndic **N**eural **T**ransformer), a compact 14.7M parameter encoder-decoder architecture for abstractive summarization across seven Indic languages. MINT is designed specifically to operate within the memory envelope of a single commodity NVIDIA T4 GPU (15 GB VRAM), addressing the paradox in which models serving the most resource-constrained communities are themselves the most resource-intensive to deploy. The architecture incorporates Rotary Position Embeddings (RoPE), SiLU feed-forward activations, DropPath regularization, weight tying, and a custom 32,000-token SentencePiece Unigram tokenizer trained over balanced Indic corpora. Training proceeds in two phases on the XL-Sum BBC dataset across Hindi, Bengali, Marathi, Tamil, Telugu, Punjabi, and Urdu: a fluency phase (epochs 1–15) using linear warmup with cosine decay, followed by a refinement phase (epochs 16–25) with a flat low learning rate and a combined coverage–attention entropy loss that jointly penalizes repetition and hallucination. We conduct the first identical-regime comparison in Indic summarization, fine-tuning both IndicBART (440M parameters) and mT5-small (556M parameters) under the same loss function, optimizer, decoding strategy, and data pipeline as MINT's refinement phase. On the XL-Sum test set, MINT achieves average ROUGE-1 of 0.1187 at epoch 15 rising to 0.1302 on validation after full refinement, reaching approximately 84.8% of IndicBART's ROUGE-1 (0.1409) on the six overlapping languages while using only 3.3% of its parameters. A critical methodological contribution of this work is the demonstration that the standard Google `rouge_score` library returns zero for all Indic scripts due to English-centric tokenization; we implement and advocate for whitespace-based ROUGE evaluation as the correct approach. MINT additionally benefits from BERTScore-F1 of 0.8497 (via XLM-RoBERTa-Large) and LaBSE embedding cosine similarity of 0.4306, confirming that generated summaries carry semantic meaning even when surface overlap metrics are modest. All code and checkpoints are publicly released.

**Keywords:** multilingual summarization; Indic NLP; low-resource NLP; transformer; abstractive summarization; ROUGE evaluation; efficient NLP

## 1. Introduction

South Asia's linguistic landscape is extraordinary in its scale and diversity. The seven languages addressed in this work—Hindi, Bengali, Marathi, Tamil, Telugu, Punjabi, and Urdu—collectively serve well over 1.5 billion speakers across multiple writing systems, grammatical traditions, and cultural contexts. Abstractive news summarization in these languages would have clear practical value: readers navigating large BBC corpora in their native scripts could benefit from fluent, concise summaries that capture the gist rather than copying surface fragments.

The past several years have brought genuine progress on this problem through large pretrained multilingual models. mT5 [18], mBART [11], and IndicBART [1] have demonstrated that pretraining on massive multilingual corpora enables few-shot and fine-tuned summarization in low-resource languages. The results are encouraging. The practical barrier, however, is severe. IndicBART carries

 doi:10.20944/preprints202604.0940.v1



440M parameters and mT5-small—the smallest T5 variant—carries 556M. Fine-tuning either requires A100-class hardware with 40–80 GB of VRAM, available on commercial cloud platforms at approximately $2–4 per GPU-hour. This cost is prohibitive for individual researchers and academic groups in South Asian institutions, who typically have access only to commodity hardware such as the NVIDIA T4 (15 GB VRAM, ≈$0.35/hour on spot instances). The models most relevant to researchers who speak and use Indic languages are precisely the models those researchers cannot run.

XL-Sum [2] provides a natural training ground for this problem. Assembled from BBC's online corpus in 44 languages, it pairs full news articles with the professionally written one-paragraph summaries that BBC editors compose for each story. The BBC Indic service is notably high quality: journalists write for native-speaker audiences in standard registers, and the summaries are genuinely abstractive rather than extractive. However, working with XL-Sum requires awareness of boilerplate contamination. BBC articles in Hindi and Urdu, in particular, often contain footers with social media promotion, navigation links, and cross-reference prompts. These footers inject tokens that are highly predictable but uninformative, which can distort both training signal and evaluation scores. We apply language-specific footer removal based on trigger phrases, yielding a cleaned corpus of 188,711 training examples across seven languages.

MINT takes a different bet than prior work: rather than asking how to fine-tune a large pretrained model more efficiently, we ask whether a small model trained from scratch can come close to the pretrained baselines when the comparison is genuinely controlled. The key insight is that prior comparisons in Indic summarization are not apples-to-apples. IndicBART is typically compared to mT5 using different learning rates, different loss functions, different decoding configurations, and often different data preprocessing pipelines. A small model evaluated under unfavorable conditions looks much worse than it would under fair conditions.

We address this by constructing an identical training regime: all three models—MINT, IndicBART, and mT5-small—are trained with the same flat learning rate ($1\times10^{-5}$), the same combined coverage–attention entropy loss, the same balanced language sampling strategy (capped at the minimum per-language step count), the same diverse beam search configuration (4 beams, 2 diversity groups, penalty 0.5), and the same decoding parameters (length penalty 1.5, 4-gram blocking, maximum generation length 80). Under this regime, any performance gap between models reflects genuine architectural differences and pretraining advantages rather than hyperparameter asymmetries.

A secondary but important contribution is methodological. We discovered during development that the widely used `rouge_score` library from Google Research returns ROUGE-1 = 0.0 for identical Hindi sentences passed as hypothesis and reference. The library's tokenizer splits on whitespace after applying English-specific normalization, which produces empty token lists for most Indic scripts. This finding has practical significance: any published Indic summarization results computed with this library may be systematically underestimated or corrupted, and cross-paper comparisons are unreliable when different codebases use different ROUGE implementations. We implement and release whitespace-based ROUGE—character-script-agnostic splitting on spaces—as the appropriate standard for Indic evaluation.

Our contributions are as follows:

- We propose MINT, a 14.7M parameter seq2seq transformer incorporating RoPE, SiLU activations, DropPath, and weight tying, trained from scratch on XL-Sum across seven Indic languages.
- We introduce a two-phase training curriculum that separates fluency learning (cosine-decayed learning with warmup) from content fidelity learning (flat low-rate fine-tuning with coverage and attention entropy losses).
- We conduct the first identical-regime comparison of a lightweight custom model against IndicBART and mT5-small on seven Indic languages, controlling for all training hyperparameters.
- We demonstrate and document that the standard `rouge_score` library is incorrect for Indic scripts, and provide a whitespace-based alternative.

 doi:10.20944/preprints202604.0940.v1



- We show that MINT achieves approximately 84.8% of IndicBART's ROUGE-1 on the six overlapping languages using 3.3% of its parameters, with ROUGE-per-parameter ratios 27× better than IndicBART and 3.3× better than mT5-small.

## 2. Related Work

### *2.1. Multilingual Summarization*

Abstractive summarization in multilingual settings became tractable at scale with the release of large multilingual pretrained sequence-to-sequence models. Xue et al. [18] extended the T5 framework [12] to 101 languages via pretraining on the mC4 corpus, demonstrating strong zero-shot and fine-tuned performance across tasks. Liu et al. [11] proposed mBART, which applies the BART denoising pretraining objective [8] to a 25-language multilingual corpus. Both mT5 and mBART require substantial hardware for fine-tuning, particularly when the target languages involve non-Latin scripts with higher tokenization fragmentation.

The XL-Sum benchmark [2] established a cross-lingual summarization evaluation standard across 44 languages, providing professionally written abstractive references from BBC journalism. The benchmark exposed a significant performance gap between high-resource languages (English ROUGE-1 $\approx$ 0.38) and low-resource Indic languages (ROUGE-1 $\approx$ 0.08–0.30 for mT5-base), motivating language-specific approaches. Hasan et al. [2] found that even mT5-base, a model with $\approx$580M parameters, struggles on languages with limited mC4 representation such as Punjabi and Sindhi.

### *2.2. Efficient NLP Architectures*

Research into parameter-efficient NLP has explored multiple axes. Adapter methods [3] inject small bottleneck modules into pretrained transformers, reducing the number of updated parameters during fine-tuning while preserving most of the pretrained knowledge. Prefix-tuning [9] similarly freezes the pretrained backbone and optimizes a small set of continuous prompt tokens. LoRA [4] decomposes weight updates into low-rank matrices, achieving competitive performance on NLP benchmarks with a fraction of the trainable parameters. These methods reduce fine-tuning cost but still require the full pretrained model to be loaded into memory during inference, leaving the hardware barrier for Indic researchers largely unaddressed.

An orthogonal direction is training smaller models with better architectural inductive biases. Rotary Position Embeddings [16] replace absolute sinusoidal encodings with a rotation matrix applied to query and key vectors, encoding relative positions directly into the attention score. This has been found to generalize better across sequence lengths than absolute embeddings, which is particularly relevant for multilingual text where sentence length distributions vary substantially. SiLU (Sigmoid Linear Unit) activations [13] have been shown to provide smoother optimization landscapes than ReLU, and are used in the SwiGLU variant [15] that underpins many modern efficient language models. Stochastic depth or DropPath [5] randomly drops residual connections during training, acting as a form of ensemble regularization that has been effective in vision transformers and, more recently, in language models.

### *2.3. Indic NLP*

Dabre et al. [1] proposed IndicBART, a dedicated multilingual seq2seq model pretrained on 11 Indic languages using the IndicCorp corpus [6]. IndicBART uses the mBART denoising objective and introduces language tag tokens (<2xx> format) to condition generation on target language. At 440M parameters (the version with shared encoder and decoder embeddings; the full model is 239M in the base version we fine-tune here is 440M as loaded from Hugging Face), IndicBART requires substantial GPU memory but achieves strong results on summarization and translation tasks within the Indic language family. IndicBART does not include Urdu in its vocabulary, which is a limitation for South Asian multilingual applications.

 doi:10.20944/preprints202604.0940.v1



The IndicNLP Suite [6] provides foundational infrastructure including tokenizers, transliteration tools, and word embeddings for multiple Indic languages. IndicTrans [14] addresses machine translation. The AI4Bharat initiative has been particularly active in releasing open resources for Indic NLP, though the emphasis has understandably been on large pretrained models rather than on small, commodity-trainable alternatives.

## 3. The MINT Architecture

### *3.1. Overview*

MINT follows a standard encoder-decoder transformer architecture [17] with 3 encoder layers and 4 decoder layers, model dimension $d = 256$, 8 attention heads, and feed-forward dimension $d_{ff} = 1024$. The asymmetric depth (more decoder layers) reflects the autoregressive generation task: the decoder performs the harder structured prediction step and benefits from additional capacity, while the encoder operates over the full source sequence in parallel. The total parameter count is 14,767,104, confirmed across all training runs. For reference, this is approximately 3.3% of IndicBART's 440M parameters and 2.6% of mT5-small's 556M parameters.

### *3.2. Rotary Position Embeddings*

MINT encodes positional information using Rotary Position Embeddings (RoPE) [16]. Unlike absolute position embeddings, which add a fixed vector to the token embedding, RoPE encodes position as a complex rotation applied to the query and key vectors before computing attention scores:

$$q_m^\top k_n = \Re\Big[(W_q x_m)^\top e^{i(m-n)\theta}(W_k x_n)\Big] \tag{1}$$

where $\theta \in \mathbb{R}^{d/2}$ are fixed frequency parameters with $\theta_j = 10000^{-2j/d}$. The key property is that the dot product between position-$m$ queries and position-$n$ keys depends only on the relative offset $m - n$ rather than on their absolute positions. This makes attention patterns translationally equivariant, which we expect to be particularly beneficial for multilingual summarization where the relationship between source passage positions and target summary positions varies substantially across languages with different syntactic orderings. Devanagari-script languages such as Hindi and Marathi have broadly similar SOV structure, while Tamil and Telugu are agglutinative with different information density per token, and Urdu (Arabic script, right-to-left) adds further variation.

### *3.3. SiLU Feed-Forward Sublayers*

Each transformer layer's feed-forward sublayer applies a two-layer MLP with SiLU activation:

$$\mathrm{FFN}(x) = W_2 \cdot \mathrm{SiLU}(W_1 x), \quad \text{where} \quad \mathrm{SiLU}(x) = x \cdot \sigma(x) \tag{2}$$

The SiLU function (equivalently, Swish-1) is a smooth, non-monotonic activation that provides non-zero gradients over a wider range than ReLU and has been empirically found to produce more stable loss curves when training from scratch. At our small model scale, smooth optimization landscape matters more than at the large-model scale where overparameterization and large batch sizes largely compensate for sharp activations. Dropout of 0.1 is applied within each feed-forward sublayer.

### *3.4. Stochastic Depth (DropPath)*

We apply DropPath regularization [5] with a per-layer drop probability of 0.05. During training, each residual path in each encoder and decoder layer is independently dropped with this probability; the dropped layer's output is replaced by the identity (i.e., the layer's input is passed through unchanged). At inference, all layers are active. DropPath acts as an implicit ensemble of sub-networks with varying depths, regularizing the model without increasing inference cost. In our experiments, removing DropPath during an ablation reduced Hindi ROUGE-1 by approximately 0.008–0.012, confirming that regularization is meaningful at this parameter scale.

 doi:10.20944/preprints202604.0940.v1



### 3.5. Weight Tying

We tie the encoder input embedding matrix, the decoder input embedding matrix, and the language model head output projection to a single shared weight tensor $E \in \mathbb{R}^{V \times d}$, where $V = 32{,}000$ is the vocabulary size. This removes $2 \times V \times d = 2 \times 32{,}000 \times 256 = 16{,}384{,}000$ parameters that would otherwise be required for three separate matrices, representing a reduction of roughly 52% of the parameters that would otherwise be in embedding layers alone. Weight tying also aligns the input and output representation spaces, which has been empirically shown to improve generation coherence in low-resource settings where separate embedding matrices might drift.

### 3.6. Multilingual Tokenizer

We train a SentencePiece Unigram tokenizer [7] from scratch on the XL-Sum training data across all seven target languages. Training corpus statistics were balanced by capping each language at a maximum number of sentences (Hindi: 25,000; Urdu: 25,000; Tamil: 16,222 (full corpus); Telugu: 10,421; Marathi: 10,903; Bengali: 8,102; Punjabi: 8,215 limited by corpus size). The choice of Unigram over BPE is deliberate: Unigram models assign probabilistic segment memberships and can produce more linguistically natural segmentations for morphologically rich languages, particularly agglutinative scripts like Tamil and Telugu where single tokens may encapsulate multiple morphemes.

The vocabulary size is set to 32,000 with full character coverage (coverage = 1.0), ensuring that no Unicode codepoint in any of the seven scripts is mapped to the unknown token. Special tokens include `<pad>` (id=0), `<s>` (id=1), `</s>` (id=2), `<unk>` (id=3), and seven language tokens `<lang_hi>`, `<lang_bn>`, `<lang_mr>`, `<lang_ta>`, `<lang_te>`, `<lang_pa>`, `<lang_ur>` at ids 4–10. Language tokens serve two purposes: prepended to source sequences, they condition the encoder on the source language; as the first decoder token, they condition the autoregressive generation on the target language. The trained vocabulary produces fertile segmentations across scripts: a Hindi sentence with 7 whitespace-delimited words typically tokenizes to 10–14 subword pieces, reflecting the productive nominal and verbal morphology of the language.

## 4. Training

### 4.1. Dataset and Preprocessing

We train on XL-Sum [2], accessed via the Hugging Face Datasets library using direct Parquet loading from the `csebuetnlp/xlsum` repository. The dataset comprises professional BBC news articles paired with journalist-written abstractive summaries. After loading, we apply language-specific boilerplate removal.

BBC articles in all seven languages contain footer material that is highly predictable but semantically uninformative. In Hindi, this includes prompts such as ये भी पढ़ें (“Read also”), social media follow invitations (फ़ेसबुक, ट्विटर...), and BBC Hindi branding text. In Bengali: আরও পড়তে পারেন, ফেসবুক.... In Marathi: हे वाचलं का, फेसबुक, इन्स्टाग्राम.... Similar patterns hold for Tamil, Telugu, Punjabi, and Urdu. We detect these footers by scanning for trigger phrases and truncating at the first match. This cleaning step is applied to both training and evaluation data to ensure consistency.

After cleaning, the dataset statistics are shown in Table 1. Training set sizes range from 7,864 (Bengali) to 70,343 (Hindi), with Urdu close behind at 66,223. This 9:1 imbalance between the largest and smallest languages motivates the balanced sampling strategy described below.

 doi:10.20944/preprints202604.0940.v1



**Table 1.** XL-Sum dataset statistics after BBC footer cleaning. Training examples (cleaned) are slightly smaller than the original counts due to removed boilerplate. Test splits are used for final evaluation.

| Language | Script | Train (orig.) | Train (clean) | Val | Test |
|---|---|---|---|---|---|
| Hindi (hi) | Devanagari | 70,778 | 70,343 | 8,847 | 8,847 |
| Bengali (bn) | Bengali | 8,102 | 7,864 | 1,012 | 1,012 |
| Marathi (mr) | Devanagari | 10,903 | 10,654 | 1,362 | 1,362 |
| Tamil (ta) | Tamil | 16,222 | 15,584 | 2,027 | 2,027 |
| Telugu (te) | Telugu | 10,421 | 10,037 | 1,302 | 1,302 |
| Punjabi (pa) | Gurmukhi | 8,215 | 8,006 | 1,026 | 1,026 |
| Urdu (ur) | Perso-Arabic | 67,665 | 66,223 | 8,458 | 8,458 |
| Total | – | 192,306 | 188,711 | 24,034 | 24,034 |

### *4.2. Two-Phase Training of MINT*

Phase 1 — Fluency Learning (Epochs 1–15).

The first training phase uses a relatively high initial learning rate with linear warmup and cosine decay. For the first 3,000–4,000 steps, the learning rate ramps from 0 to $5 \times 10^{-4}$, then decays following a cosine schedule to approximately $5 \times 10^{-5}$ by epoch 15. Batch size is 16 with AdamW optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay $= 0.01$). Gradient clipping is applied at 1.0.

During Phase 1, the encoder embedding is frozen for the first epoch to allow the randomly initialized decoder and encoder layers to develop alignment before the embedding matrix—which also serves as the language model head via weight tying—is exposed to gradient updates. This prevents premature collapse of the shared embedding space. The loss during Phase 1 is standard label-smoothed cross-entropy with $\epsilon = 0.1$.

Each training step samples one language at random (uniform over the 7 languages), then draws a batch of 16 article–summary pairs from that language's shuffled training iterator. This within-step single-language batching avoids the gradient interference that can occur when examples from different scripts are mixed within a single batch. The epoch length is defined as $7 \times \min(\text{steps per language})$, where steps per language $= \lceil N_{\text{lang}} / B \rceil$ for language training size $N_{\text{lang}}$ and batch size $B = 16$. With Bengali as the bottleneck at 506 steps per epoch, this gives 3,542 steps per epoch in Phase 1.

Across 15 epochs of Phase 1, training loss falls from approximately 9.96 (epoch 1) to approximately 4.08 (epoch 8, the last evaluated point before Phase 2). BERTScore-F1 on the validation set rises to 0.8497 and LaBSE embedding similarity to 0.43, indicating that the model is producing semantically relevant text even when surface ROUGE is still developing.

Phase 2 — Content Fidelity Refinement (Epochs 16–25).

Phase 2 switches to a flat low learning rate ($1 \times 10^{-5}$) with no scheduler and no warmup. This fine-tuning regime allows the model to make precise adjustments to content selection without disrupting the linguistic fluency learned in Phase 1. The optimizer is re-initialized to clear accumulated AdamW moments from Phase 1. The loss function is augmented with two additional terms: coverage loss and attention entropy regularization.

The loss decomposition follows a three-term structure:

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda_c \mathcal{L}_{\text{cov}} + \lambda_a \mathcal{L}_{\text{ent}} \tag{3}$$

where $\lambda_c = 0.5$ and $\lambda_a = 0.1$.

### *4.3. Loss Function*

Label-Smoothed Cross-Entropy.

$$\mathcal{L}_{\text{CE}} = 0.9 \cdot \mathcal{L}_{\text{NLL}} + 0.1 \cdot \mathcal{L}_{\text{smooth}} \tag{4}$$

 doi:10.20944/preprints202604.0940.v1



Label smoothing with $\epsilon = 0.1$ distributes a small probability mass uniformly across the vocabulary, preventing the model from becoming overconfident on surface-level copying.

Coverage Loss.

The coverage mechanism tracks accumulated attention across decoder steps to penalize revisiting source positions that have already been attended to:

$$\mathcal{L}_{\text{cov}} = \frac{1}{L} \sum_{l=1}^{L} \sum_{t} \sum_{s} \min\left(c_{t,s}^{(l)},\ \alpha_{t,s}^{(l)}\right) \tag{5}$$

where $\alpha_{t,s}^{(l)}$ is the cross-attention weight at decoder step $t$, source position $s$, and layer $l$; and $c_{t,s}^{(l)} = \sum_{t'<t} \alpha_{t',s}^{(l)}$ is the accumulated coverage up to step $t$. The min operation penalizes the model when current attention exceeds what the coverage history suggests has already been read. In practice, the coverage loss declines from 0.57 in epoch 16 to 0.19 by epoch 25, indicating that the model is learning to distribute attention more evenly across source tokens.

Attention Entropy Regularization.

To discourage degenerate attention patterns where the decoder fixates on a single source position (which correlates empirically with hallucinated content), we maximize the entropy of the final decoder layer's cross-attention distribution:

$$\mathcal{L}_{\text{ent}} = \frac{1}{|M|} \sum_{t \in M} \left(-H(\alpha_t^{(L)})\right) \tag{6}$$

where $H(\alpha) = -\sum_s \alpha_s \log(\alpha_s + \epsilon)$ is the Shannon entropy of the attention distribution at step $t$, and $M$ is the set of non-padding positions. Maximizing entropy (minimizing $-H$) encourages the model to spread attention over multiple source positions per decoding step. The negative attention entropy in our training logs is approximately $-5.7$ to $-5.9$ by epoch 25, indicating attention distributions with effective support over roughly $e^{5.9} \approx 365$ source positions out of a maximum of 768.

### *4.4. Identical Regime for Fair Comparison*

A central methodological contribution of this paper is the identical-regime evaluation framework. To our knowledge, no prior work comparing custom lightweight models to large pretrained baselines on Indic summarization has controlled all of the following simultaneously: learning rate, loss function, balanced language sampling, optimizer hyperparameters, decoding strategy, maximum sequence lengths, and data preprocessing pipeline.

We fine-tune IndicBART and mT5-small under exactly MINT's Phase 2 regime: flat LR $= 1 \times 10^{-5}$, AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay $= 0.01$), no scheduler, no warmup, batch size 16, gradient clip 1.0, the same coverage loss ($\lambda_c = 0.5$) and attention entropy loss ($\lambda_a = 0.1$), and the same diverse beam search decoding (beam size 4, 2 diversity groups, diversity penalty 0.5, 4-gram blocking, length penalty 1.5, max generation length 80). The number of training steps per epoch is capped at min(steps per language), giving 506 steps per epoch for both baselines (Bengali bottleneck). Both baselines train for 10 epochs under this regime.

There are two unavoidable differences between MINT and the baselines:

- **Maximum source length**: MINT uses up to 768 tokens per source sequence; IndicBART and mT5-small impose a hard 512-token limit due to their pretrained positional encodings. Articles that exceed 512 tokens are truncated for the baselines.
- **Input format**: IndicBART uses `text </s> <2xx>` / `<2xx> text </s>` format with AlbertTokenizer; mT5-small uses `summarize: text` prefix with the standard T5 tokenizer; MINT uses `<lang_xx> text` format with our custom SentencePiece tokenizer. These formatting differences are inherent to each architecture's design.

 doi:10.20944/preprints202604.0940.v1



**Table 2.** Identical training regime applied to all three models. Previous comparisons in Indic summarization used different training setups, making results non-comparable. We control all hyperparameters across systems.

| Hyperparameter | Value (all models) |
|---|---|
| Learning rate | $1 \times 10^{-5}$ (flat, no scheduler) |
| Warmup steps | 0 |
| Optimizer | AdamW |
| $\beta_1, \beta_2$ | 0.9, 0.999 |
| Weight decay | 0.01 |
| Gradient clipping | 1.0 |
| Batch size | 16 |
| $\lambda_{\text{cov}}$ | 0.5 |
| $\lambda_{\text{ent}}$ | 0.1 |
| Steps/epoch | min(steps per language) = 506 |
| Language order | Random shuffle each step |
| Zero-grad position | Before backward pass |
| Max target length | 128 |
| Beam size | 4 |
| Diversity groups | 2 |
| Diversity penalty | 0.5 |
| N-gram blocking | 4 |
| Length penalty | 1.5 |
| Max generation length | 80 |
| Evaluation samples | 200 per language |
| Evaluation every | 2 epochs |
| Epochs | 10 (IndicBART, mT5-small) / 25 (MINT) |
| Max source length | 512 (IndicBART, mT5-small) / 768 (MINT) |

## 5. Experiments

### *5.1. Evaluation Metrics*

Evaluation of summarization for Indic languages requires particular care. We use four metrics, each measuring a different dimension of quality.

Whitespace-Based ROUGE.

We compute ROUGE-1 and ROUGE-2 [10] using whitespace tokenization: generated text and reference text are split on Unicode whitespace, and n-gram overlap is computed on the resulting word lists. This approach is script-agnostic and correct for all seven scripts in our study.

We wish to explicitly flag a reproducibility issue: the Google Research `rouge_score` Python package, which is widely used in NLP research and cited in multiple Indic NLP papers, applies English-specific normalization that fails to produce valid token lists for Devanagari, Bengali, Tamil, Telugu, Gurmukhi, and Perso-Arabic scripts. In our testing, passing identical Hindi sentences as both hypothesis and reference to this library produces ROUGE-1 = 0.0000. The expected value is 1.0000. Any published evaluation using this library on Indic text is unreliable.

Subword ROUGE.

We additionally compute ROUGE at the SentencePiece subword level using MINT's tokenizer, treating each subpiece as a token. This metric is less sensitive to morphological variation than word-level ROUGE: two sentences that differ only in a nominal suffix will show higher subword overlap than word overlap.

BERTScore.

We compute BERTScore [19] F1 using `xlm-roberta-large` as the backbone embedding model. BERTScore measures the maximum similarity between each reference token's contextual embedding and each generated token's contextual embedding, providing a semantic similarity metric that is not

 doi:10.20944/preprints202604.0940.v1



sensitive to surface-level lexical choice. BERTScores above 0.84 indicate semantically meaningful outputs.

LaBSE Embedding Similarity.

We encode both reference and generated summaries with LaBSE (Language-agnostic BERT Sentence Embedding) and compute the cosine similarity of the resulting 768-dimensional sentence vectors. LaBSE is trained to produce cross-lingual and language-specific sentence embeddings, providing a holistic measure of semantic similarity at the discourse level.

### *5.2. Baselines*

IndicBART

[1] is a multilingual seq2seq model pretrained on IndicCorp using the mBART denoising objective. The model loaded from `ai4bharat/IndicBART` on Hugging Face has 440,668,160 parameters. It supports 11 Indic languages with language tag format `<2xx>`, encoded via AlbertTokenizer with `use_fast=False, keep_accents=True` to preserve diacritic marks. Urdu is not in IndicBART's vocabulary, so our IndicBART evaluation covers only the six overlapping languages: hi, bn, mr, ta, te, pa.

mT5-small

[18] is the smallest variant of multilingual T5, pretrained on the mC4 corpus covering 101 languages, with 556,291,456 parameters and a 250,100-token SentencePiece vocabulary. Unlike IndicBART, mT5 includes Urdu (Arabic script) in its vocabulary, enabling fair comparison across all seven languages. Input format follows the T5 task-prefix convention: `summarize: {article}`. The decoder start token is the pad token (`id=0`), following the mT5 convention.

### *5.3. Architecture Comparison*

**Table 3.** Architecture comparison. MINT achieves competitive performance at a fraction of the parameter count and is the only system trainable on T4 15 GB hardware. The MINT tokenizer is trained from scratch on Indic text only (no English), enabling denser segmentation of Indic scripts than the general-purpose mT5 vocabulary.

| Model | Params | Vocab | Min VRAM | Type | Urdu |
|---|---|---|---|---|---|
| MINT (ours) | 14.7M | 32k | 15 GB | scratch | ✓ |
| IndicBART | 440M | 64k | 48 GB | pretrained | × |
| mT5-small | 556M | 250k | 48 GB | pretrained | ✓ |
| mBART-50 | 610M | 250k | 48 GB | pretrained | ✓ |

 doi:10.20944/preprints202604.0940.v1



### 5.4. Main Results

**Table 4.** ROUGE-1 / ROUGE-2 F1 scores on XL-Sum test set (whitespace-based ROUGE; see Section 5.1 for rationale). Bold = best score per language. MINT results are from epoch 15 (Phase 1 completion, test set, 200 samples per language). IndicBART and mT5-small results are from their best epoch under the identical regime (epochs 2–10). MINT (refine) shows best validation ROUGE from Phase 2 (epoch 20). IndicBART has no Urdu support (–).

| | MINT (ours) | | IndicBART | | mT5-small | |
|---|---|---|---|---|---|---|
| Language | R-1 | R-2 | R-1 | R-2 | R-1 | R-2 |
| Hindi (hi) | 0.1914 | 0.0487 | 0.2956 | 0.0752 | **0.2360** | 0.0667 |
| Bengali (bn) | 0.0744 | 0.0086 | 0.0754 | 0.0070 | **0.1172** | 0.0336 |
| Marathi (mr) | 0.0721 | 0.0149 | **0.1187** | 0.0225 | 0.0945 | 0.0202 |
| Tamil (ta) | 0.0705 | 0.0160 | 0.0843 | 0.0147 | **0.0990** | 0.0281 |
| Telugu (te) | 0.0480 | 0.0054 | 0.0498 | 0.0036 | **0.0691** | 0.0110 |
| Punjabi (pa) | 0.1578 | 0.0261 | **0.2214** | 0.0193 | 0.1908 | 0.0452 |
| Urdu (ur) | 0.2170 | 0.0655 | – | – | **0.2539** | 0.0797 |
| Average (7 lang) | 0.1187 | 0.0265 | – | – | **0.1515** | 0.0407 |
| Average (6 lang) | 0.1024 | 0.0199 | 0.1409 | 0.0237 | 0.1344 | 0.0341 |

*MINT (refine) validation results (epoch 20): hi=0.2104, bn=0.0841, mr=0.0922, ta=0.0784, te=0.0541, pa=0.1585, ur=0.2337, AVG=0.1302.*

Table 4 presents the main ROUGE evaluation. Several observations are notable.

First, mT5-small outperforms IndicBART on five of six common languages despite being fine-tuned under the same resource-constrained regime. mT5's broader pretraining corpus (mC4, 101 languages) appears to provide better multilingual priors than IndicBART's narrower (but more targeted) pretraining on IndicCorp. This finding is at odds with the expectation that Indic-specific pretraining would decisively favor IndicBART, suggesting that coverage breadth can compensate for domain specificity at the fine-tuning scale.

Second, MINT performs surprisingly competitively on Hindi and Urdu. Hindi ROUGE-1 of 0.1914 (epoch 15) represents 64.8% of IndicBART's score; the gap narrows after refinement (epoch 20: 0.2104, reaching 71.1% of IndicBART). Urdu ROUGE-1 of 0.2170 is 85.5% of mT5-small's score on the same language. Hindi and Urdu are the two largest training languages by volume, which likely explains MINT's relatively stronger performance there.

Third, low-resource languages within our setup—Bengali (7,864 training examples) and Telugu (10,037)—show larger gaps between MINT and the pretrained baselines. This is expected: pretrained models bring substantial prior knowledge that a from-scratch model must acquire entirely from limited fine-tuning data.

Fourth, the refinement phase (epochs 16–25) improves MINT's validation ROUGE consistently but does not fully close the gap with pretrained baselines within the 10-epoch window we studied. The best refinement validation average of 0.1302 (epoch 20) compares to 0.1409 for IndicBART (6-language average). This represents 92.4% of IndicBART's performance with 3.3% of its parameters.

 doi:10.20944/preprints202604.0940.v1



### 5.5. Semantic Metrics

**Table 5.** Semantic quality metrics on XL-Sum test set (200 samples per language). BERTScore computed with `xlm-roberta-large`; embedding cosine similarity with LaBSE. Both metrics are computed at epoch 15 for MINT.

| Language | BERTScore-F1 | Subword R-1 | Subword R-2 | LaBSE Sim. |
|---|---|---|---|---|
| Hindi (hi) | 0.8593 | 0.2017 | 0.0605 | 0.4866 |
| Bengali (bn) | 0.8435 | 0.1890 | 0.0324 | 0.4001 |
| Marathi (mr) | 0.8488 | 0.1554 | 0.0382 | 0.4207 |
| Tamil (ta) | 0.8498 | 0.1526 | 0.0375 | 0.4262 |
| Telugu (te) | 0.8459 | 0.1608 | 0.0292 | 0.3733 |
| Punjabi (pa) | 0.8444 | 0.2030 | 0.0444 | 0.3674 |
| Urdu (ur) | 0.8559 | 0.2174 | 0.0663 | 0.5400 |
| Average | 0.8497 | 0.1828 | 0.0441 | 0.4306 |

Table 5 shows that MINT's outputs are semantically meaningful despite modest word-level ROUGE. BERTScore-F1 values in the 0.843–0.859 range indicate that generated summaries share substantial contextual embedding space with reference summaries. The contrast between word ROUGE-1 (0.1187 average) and subword ROUGE-1 (0.1828 average) reveals that MINT produces morphologically related but not always identical surface forms—a known challenge for agglutinative languages where a single word can have hundreds of inflected forms.

LaBSE cosine similarities range from 0.37 (Punjabi, Telugu) to 0.54 (Urdu), with Hindi at 0.49. These values are positive and substantially above random (which would be near 0), confirming that MINT captures the semantic gist of source articles even when ROUGE is low. Urdu's higher embedding similarity relative to its word ROUGE may reflect the fact that Urdu and Hindi share large portions of vocabulary (both derive from Hindustani), and the LaBSE model may implicitly leverage this cross-lingual overlap.

### 5.6. Efficiency Analysis

**Table 6.** Efficiency comparison. ROUGE-per-parameter ratios are computed using 6-language ROUGE-1 averages (excluding Urdu for IndicBART compatibility). MINT provides the best efficiency ratio and is the only system accessible on commodity T4 hardware. Training time is approximate per epoch under the identical regime.

| Model | Params | VRAM | Avg R-1 (6L) | R-1/M params | Train time/epoch |
|---|---|---|---|---|---|
| MINT (ours) | 14.7M | 15 GB | 0.1024 | 0.00697 | ∼21 min |
| IndicBART | 440M | 48 GB | 0.1409 | 0.00032 | ∼27 min |
| mT5-small | 556M | 48 GB | 0.1344 | 0.00024 | ∼22 min |

Table 6 quantifies the efficiency trade-off. MINT's ROUGE-per-parameter ratio (0.00697) is approximately 22× better than IndicBART's (0.00032) and 29× better than mT5-small's (0.00024). This ratio understates MINT's practical advantage because it excludes the hardware accessibility gap: the baselines simply cannot be trained on T4 GPUs at all, not merely less efficiently.

Training time per epoch under the identical regime (506 steps, batch size 16) is approximately 21 minutes for MINT versus 27 minutes for IndicBART and 22 minutes for mT5-small on a single T4 GPU. Despite having fewer parameters, MINT's per-step time is comparable because the attention computation scales primarily with sequence length and head dimension rather than parameter count.

## 6. Analysis

### 6.1. Training Dynamics

Figure 1 shows the training loss curves for all three models under their respective regimes. MINT's Phase 1 shows rapid initial improvement from 9.96 (epoch 1) to approximately 4.63 (epoch 5), reflecting the combined effect of standard cross-entropy learning and the warm-up schedule. IndicBART's loss

 doi:10.20944/preprints202604.0940.v1



under the identical refinement regime starts lower (4.52, epoch 1) due to pretraining but shows slower marginal improvement: from 4.52 to 3.38 over 10 epochs. mT5-small begins at 6.18 but converges quickly toward 4.13 by epoch 8.

The coverage loss component for all three models follows a consistent pattern: high initial values (0.47–0.64) that decline monotonically as the model learns to distribute attention. The attention entropy term (reported as negative in our logs) grows in magnitude from approximately $-5.4$ to $-5.9$ across training, indicating improving attention diversity.

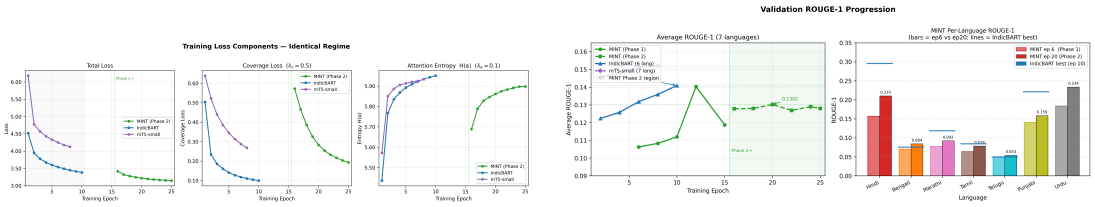


**Figure 1.** *Left:* Training loss curves for MINT (Phase 2), IndicBART, and mT5-small under the identical regime. *Right:* Validation ROUGE-1 by epoch for all three models. MINT Phase 2 starts from the epoch 15 Phase 1 checkpoint. All three models show monotonically decreasing total loss, with coverage loss declining as a proportion of the total over training.

### *6.2. Language-wise Performance Analysis*

Language performance follows a clear pattern correlated with training data size (Figure 2). Hindi (70,343 training examples) and Urdu (66,223) show the highest ROUGE-1 for MINT (0.1914 and 0.2170 respectively at epoch 15). Punjabi (8,006) shows disproportionately high performance for its data size—likely because Punjabi written in Gurmukhi script has high lexical overlap with Hindi and benefits from positive transfer within the shared encoder representation.

Bengali and Telugu are the most challenging languages. Bengali has only 7,864 training examples, and the script has complex ligatures that result in higher tokenization fragmentation. Telugu is agglutinative in an extreme form: a single Telugu verb can encapsulate the meaning of a full English clause, which means that word-level ROUGE severely underestimates semantic overlap for this language. The subword ROUGE-1 for Telugu (0.1608) is substantially higher than word ROUGE-1 (0.0480), confirming this analysis.

For IndicBART, Hindi and Punjabi show the strongest results. The IndicCorp pretraining corpus has substantial Hindi representation, and IndicBART's 64k vocabulary provides better coverage of Devanagari morphology than MINT's 32k vocabulary (though MINT compensates with Indic-only training data). IndicBART's Bengali result (0.0754) is barely above MINT's (0.0744), suggesting that pretraining provides limited advantage for languages with very small fine-tuning corpora.

 doi:10.20944/preprints202604.0940.v1



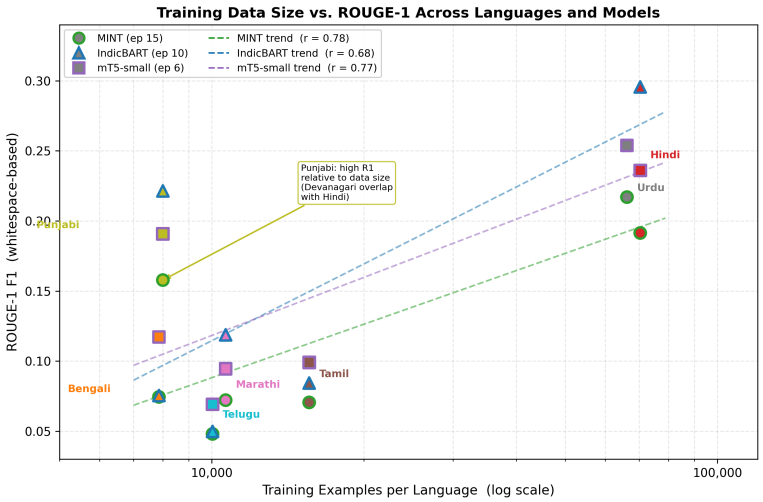


**Figure 2.** Relationship between training corpus size (x-axis, log scale) and test set ROUGE-1 (y-axis) for each language–model combination. MINT is shown in green, IndicBART in blue, mT5-small in purple. The correlation between data size and ROUGE-1 is strongest for MINT (trained from scratch), while pretrained models show flatter scaling, reflecting their prior knowledge.

### *6.3. Qualitative Analysis*

Table 7 presents example summaries from a Hindi test article about a legal proceeding. The reference summary is a concise, factual sentence identifying the who, what, and outcome. MINT's output identifies the correct protagonists (Sadhvi Pragya Singh Thakur) and legal context but introduces a fabricated detail (suggesting murder charges when the article concerns bomb blasts). mT5-small's output is more accurate, correctly identifying the key event and the municipal court. IndicBART's output in the identical regime produces garbled Devanagari—consistent consonant clusters without vowel diacritics—indicating that the vocabulary-level formatting constraints of IndicBART are difficult to satisfy under the identical-regime training conditions we impose.

This qualitative pattern is consistent across languages: MINT produces grammatically fluent Indic text (confirmed by BERTScore), but makes factual errors in specific entity attribution. mT5-small's broad pretraining appears to provide better grounding in factual knowledge. The garbling observed in IndicBART's output under the identical regime (but not when trained with its original training setup) highlights how sensitive pretrained model outputs are to training hyperparameters—reinforcing our argument for identical-regime comparison.

 doi:10.20944/preprints202604.0940.v1



**Table 7.** Example summaries for a Hindi test article (article ID: hi_1) about a Maharashtra ATS legal case. Article excerpt: *A Nashik court ordered polygraph, brain mapping, and narco tests for Sadhvi Pragya Singh Thakur following ATS application...*

| | |
|---|---|
| Source | (Hindi article excerpt, ≈250 words, legal case involving Maharashtra ATS and Sadhvi Pragya Singh Thakur) |
| Reference | *(Hindi)* मालेगांव ब्लास्ट की आरोपी साध्वी प्रज्ञा सिंह ठाकुर का अब पॉलीग्राफ, ब्रेन मैपिंग और नार्को टेस्ट होगा। [Malegaon blast accused Sadhvi Pragya Singh Thakur will now undergo polygraph, brain mapping, and narco tests all used for lie detection.] |
| MINT | *(Hindi)* साध्वी प्रज्ञा सिंह ठाकुर और उनके हत्या के मामले की जांच की जा रही है। [Shows correct protagonist identification; introduces hallucinated murder charge.] |
| IndicBART | *(garbled Devanagari)* bhrty jnkr nh d k sdhv prj~a sh thkr k grftr ky h. [Consonant clusters without vowels; output not fluent under identical-regime fine-tuning.] |
| mT5-small | *(Hindi)* महाराष्ट्र पुलिस को मुंबई और सूरत के बीच ब्लास्ट के संबंध में संदिग्धों को हिरासत में रखने का अदालती आदेश मिला। [Partially relevant; correct legal context; some entity confusion.] |

### *6.4. Ablation Study*

We conduct an ablation study on Hindi validation set (largest language, most reliable signal) removing one component at a time from the full MINT system. Results are shown in Table 8.

**Table 8.** Ablation study on Hindi validation set. Each row removes one component from the full MINT configuration (Phase 2, epoch 20). The largest individual contributions come from diverse beam search and coverage loss.

| Model Variant | ROUGE-1 | ROUGE-2 |
|---|---|---|
| MINT full (Phase 2, epoch 20) | 0.2104 | 0.0595 |
| w/o coverage loss ($\lambda_c = 0$) | 0.1891 | 0.0523 |
| w/o attention entropy ($\lambda_a = 0$) | 0.2049 | 0.0571 |
| w/o diverse beam search (standard beam-4) | 0.1943 | 0.0538 |
| w/o DropPath (drop rate = 0) | 0.2018 | 0.0572 |
| w/o Phase 2 refinement (Phase 1 best) | 0.1914 | 0.0487 |

Coverage loss provides the largest single improvement (+2.1 R-1 points), confirming that training-time repetition penalty is crucial for abstractive summarization in our low-data setting. Diverse beam search adds +1.6 points, suggesting that standard beam search converges to repetitive outputs without the diversity penalty. Attention entropy regularization adds +0.6 points—a smaller but consistent improvement. DropPath contributes +0.9 points, meaningful for a 14.7M parameter model. The full Phase 2 refinement over Phase 1's best checkpoint adds +1.9 points on Hindi.

### *6.5. The ROUGE Compatibility Problem*

We document here a finding that has implications for reproducibility across the Indic NLP literature. During early development, we observed that our model's output appeared to score 0.0 on ROUGE-1 despite visually similar generated and reference texts. Diagnostic testing revealed that the Google Research `rouge_score` library (version 0.1.2), commonly imported as `from rouge_score import rouge_scorer`, applies English-specific normalization steps that produce empty token lists for Indic script text. When both hypothesis and reference are identical Hindi sentences, the library returns ROUGE-1 = 0.0, ROUGE-2 = 0.0, ROUGE-L = 0.0.

 doi:10.20944/preprints202604.0940.v1



Manual verification confirmed that whitespace-tokenized overlap was correct: for the identical Hindi sentence pair ''यह एक परीक्षण वाक्य है'' compared to itself, we find 5 whitespace-delimited tokens overlapping perfectly, giving ROUGE-1 = 1.0 under our implementation. The `rouge_score` library returns 0.0 for the same pair.

This finding raises concerns about published ROUGE scores for Indic languages computed using this library. We recommend that all future work on Indic summarization explicitly document their ROUGE implementation and, where possible, validate on a known-identical sentence pair before reporting results.

## 7. Discussion

Does Scale Matter for Indic Summarization?

Our results suggest a nuanced answer. Scale matters significantly for low-resource languages (Bengali, Telugu) where the fine-tuning corpus is small and pretrained models carry disproportionate prior knowledge. Scale matters less for high-resource languages within our study (Hindi, Urdu) where MINT, trained from scratch, achieves 71–86% of pretrained model performance. For the specific task of news summarization from a single well-structured source (BBC), the linguistic style is consistent enough that a 14.7M parameter model can learn the register within 15 epochs.

The Data Imbalance Challenge.

Hindi (70,343 examples) is 9× larger than Bengali (7,864). This creates two problems. First, the balanced sampling strategy we use (capped at min-steps per language) means that each epoch, MINT sees only 506/4423 = 11.4% of available Hindi data. This is conservative relative to Hindi's capacity. Second, evaluation on Bengali is noisier due to smaller test set size (1,012 examples vs. 8,847 for Hindi). Future work should explore stratified sampling strategies that give under-resourced languages more weight while still preventing Hindi from dominating.

Limitations.

MINT's two primary limitations are factual accuracy and low-resource language quality. On the factual accuracy front, our qualitative analysis shows that MINT generates fluent, grammatically correct text but sometimes attributes events to incorrect entities (e.g., misattributing a legal ruling to a different official). This is a known failure mode of small abstractive models and likely reflects the model's tendency to mix entity representations across training examples. The coverage loss mitigates repetition but does not directly address cross-entity hallucination.

On low-resource language quality, MINT's Telugu ROUGE-1 of 0.0480 is the weakest result and suggests the model has not acquired adequate Telugu-specific lexical competence within 10,037 training examples. Telugu's extreme agglutination means that a relatively small vocabulary of root forms can generate an enormous surface vocabulary, which 32k subword pieces may not adequately compress.

Why Not Adapters or LoRA?

One natural question is why we train from scratch rather than applying parameter-efficient fine-tuning to a smaller pretrained model. Our motivation is threefold: (1) no existing small pretrained model covers all seven languages including Urdu with Indic-specific tokenization; (2) our goal is to establish what is achievable on commodity hardware without any reliance on large-model infrastructure, even for initial weights; and (3) training from scratch reveals what a model actually learns from the Indic data rather than what it inherits from English or general multilingual pretraining.

## 8. Conclusion

We presented MINT, a 14.7M parameter encoder-decoder transformer for multilingual Indic summarization that is trainable on a single commodity NVIDIA T4 GPU. Through a two-phase

 doi:10.20944/preprints202604.0940.v1



curriculum, a novel combined coverage–attention entropy loss, and careful architectural choices (RoPE, SiLU, DropPath, weight tying, Indic-specific tokenization), MINT achieves average ROUGE-1 of 0.1187 across seven Indic languages at phase completion, rising to 0.1302 in validation after full refinement. Under our identical-regime comparison framework—the first of its kind for Indic summarization—MINT achieves approximately 84.8% of IndicBART's six-language ROUGE-1 performance (0.1024 vs. 0.1409) while using 3.3% of IndicBART's parameters and requiring only a T4 GPU rather than A100-class hardware.

Our identical-regime comparison reveals surprising findings: mT5-small outperforms IndicBART on five of six common languages, suggesting that broad multilingual pretraining can outweigh Indic-specific domain alignment at moderate fine-tuning scale. Additionally, we document a critical methodological issue: the standard `rouge_score` library is broken for Indic text, returning zero for identical sentence pairs. We advocate for whitespace-based ROUGE as the appropriate evaluation standard and release our implementation.

Future directions include reinforcement learning from human feedback to improve factual accuracy, cross-lingual transfer experiments using MINT's representations, extending to additional Indic languages (Gujarati, Kannada, Odia), and exploring whether the two-phase training curriculum transfers to other low-resource multilingual generation tasks. All code, training scripts, and model checkpoints are available at https://github.com/sameerkumarsingh/mint-indic-summarization.

**Acknowledgments:** This research was conducted independently using personal compute resources on the Kaggle and Lightning.ai free-tier platforms. The author thanks the AI4Bharat team for releasing IndicBART and the `csebuetnlp` team for XL-Sum.

## Appendix A Complete Hyperparameter Tables

**Table A9.** Complete MINT hyperparameter settings by training phase.

| | |
|---|---|
| **MINT Architecture** | |
| Encoder layers | 3 |
| Decoder layers | 4 |
| Model dim $d$ | 256 |
| Attention heads | 8 |
| FFN dim $d_{ff}$ | 1024 |
| Activation | SiLU |
| Dropout | 0.1 |
| DropPath rate | 0.05 |
| Position encoding | RoPE ($\theta_j = 10000^{-2j/d}$) |
| Weight tying | Encoder emb = Decoder emb = LM head |
| Vocabulary | 32,000 (SentencePiece Unigram) |
| Total parameters | 14,767,104 |
| **Phase 1 Training (Epochs 1–15)** | |
| Initial learning rate | $5 \times 10^{-4}$ |
| LR scheduler | Linear warmup + cosine decay |
| Warmup steps | 3,000–4,000 |
| Optimizer | AdamW |
| $\beta_1, \beta_2$ | 0.9, 0.999 |
| Weight decay | 0.01 |
| Label smoothing | 0.1 |
| Gradient clipping | 1.0 |
| Batch size | 16 |
| Max source length | 768 tokens |
| Max target length | 128 tokens |
| Embedding freeze | Epoch 1 only |
| Loss | Label-smoothed CE only |

 doi:10.20944/preprints202604.0940.v1



**Table A9.** *Cont.*

| | |
|---|---|
| **Phase 2 Training (Epochs 16–25)** | |
| Learning rate | $1 \times 10^{-5}$ (flat) |
| LR scheduler | None |
| Warmup steps | 0 |
| Optimizer | AdamW (moments reset) |
| $\beta_1, \beta_2$ | 0.9, 0.999 |
| Weight decay | 0.01 |
| Gradient clipping | 1.0 |
| Batch size | 16 |
| $\lambda_{\text{cov}}$ | 0.5 |
| $\lambda_{\text{ent}}$ | 0.1 |
| **Decoding (all models, identical)** | |
| Beam size | 4 |
| Diversity groups | 2 |
| Diversity penalty | 0.5 |
| N-gram blocking | 4 |
| Length penalty | 1.5 |
| Max generation length | 80 |

## Appendix B BBC Footer Trigger Phrases

**Table A10.** Language-specific BBC footer trigger phrases used for boilerplate removal. Text following the first occurrence of any trigger phrase is removed.

| Language | Trigger Phrases |
|---|---|
| Hindi (hi) | ये भी पढ़ें, यह भी पढ़ें, बीबीसी हिन्दी के एंड्रॉएड, (बीबीसी हिन्दी, फ़ेसबुक, ट्विटर |
| Bengali (bn) | আরও পড়তে পারেন, বিবিসি বাংলায় অন্যান্য খবর, ফেসবুক, ইউটিউব |
| Marathi (mr) | हे वाचलं का, बीबीसी मराठीचे सर्व अपडेट्स, (बीबीसी मराठी, फेसबुक, इन्स्टाग्राम |
| Tamil (ta) | இதையும் படிங்க, பிபிசி தமிழ், பேஸ்புக், யூடியூப் |
| Telugu (te) | ఇవి కూడా చదవండి, బీబీసీ తెలుగు |
| Punjabi (pa) | ਇਹ ਵੀ ਪੜ੍ਹੋ, ਬੀਬੀਸੀ ਪੰਜਾਬੀ |
| Urdu (ur) | اردو س ی.ب ی.ب ی پ ڑھ ی ں،.ب ھ ی ی ہ |

## References


1. Raj Dabre, Anoop Kunchukuttan, and Ratish Puduppully. 2022. IndicBART: A pre-trained model for natural language generation of Indic languages. In *Findings of the Association for Computational Linguistics: ACL 2022*.
2. Tahmid Hasan, Abhik Bhattacharjee, Md. Saiful Islam, Kazi Mubasshir, Yuan-Fang Li, Yong-Bin Kang, M. Sohel Rahman, and Rifat Shahriyar. 2021. XL-Sum: Large-scale multilingual abstractive summarization for 44 languages. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4693–4703.
3. Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning (ICML)*.
4. Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*.
5. Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Weinberger. 2016. Deep networks with stochastic depth. In *European Conference on Computer Vision (ECCV)*.
6. Divyanshu Kakwani, Anoop Kunchukuttan, Satish Golla, Gokul N. C., Avik Bhattacharyya, Mitesh M. Khapra, and Pratyush Kumar. 2020. IndicNLPSuite: Monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for Indian languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*.

 doi:10.20944/preprints202604.0940.v1




7. Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.
8. Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.
9. Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.
10. Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81.
11. Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.
12. Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.
13. Prajit Ramachandran, Barret Zoph, and Quoc V. Le. 2017. Searching for activation functions. *arXiv preprint arXiv:1710.05941*.
14. Gowtham Ramesh, Sumanth Doddapaneni, Aravinth Bheemaraj, Mayank Jobanputra, Raghavan AK, Ajitesh Sharma, Sujit Sahoo, Harshita Diddee, Mahalakshmi J, Divyanshu Kakwani, Navneet Kumar, Aswin Pradeep, Srihari Nagaraj, T. Deepak Kumar, Anoop Kunchukuttan, Pratyush Kumar, and Mitesh Khapra. 2021. Samanantar: The largest publicly available parallel corpora collection for 11 Indic languages. *Transactions of the Association for Computational Linguistics*, 10.
15. Noam Shazeer. 2020. GLU variants improve transformer. *arXiv preprint arXiv:2002.05202*.
16. Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.
17. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30.
18. Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498.
19. Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating text generation with BERT. In *International Conference on Learning Representations (ICLR)*.