When Tokenizers Fail: Byte-Level Chunking for Zero-Shot Transfer to Low-Resource Languages

Sanjeev Kumar[1,2] **Atsuki Yamaguchi**[2] **Nikolaos Aletras**[2]

[1]Department of Computer Science and Engineering, IIT Bombay, India
[2]School of Computer Science, University of Sheffield, United Kingdom

sanjeev@cse.iitb.ac.in, {ayamaguchi1, n.aletras}@sheffield.ac.uk

Abstract

Subword tokenization hinders low-resource language processing by imposing frequency patterns from dominant languages onto script-sharing variants. Byte-level models bypass this issue by processing raw UTF-8 characters, yet they create a granularity mismatch for word-level tasks in non-Latin scripts. Hierarchical byte-level architectures address this mismatch by grouping bytes into word-aligned chunks. However, these architectures require massive training data and suffer from representational misalignment when paired with frozen subword-based language models. In this paper, we propose an adapted hierarchical network framework that bridges this modality gap without extensive training. Our method initializes byte embeddings directly from the subword representations of a frozen base model. We apply a chunk alignment loss to project dynamically grouped byte chunks toward precomputed subword targets, and interleave lightweight part-of-speech (POS) supervision to guide boundary detection. Experiments across six languages demonstrate that our tokenizer-free approach improves performance for word-level morphological tasks, yielding up to a 13.3% improvement on POS tagging.[1]

1 Introduction

Natural language processing (NLP) for low-resource languages is fundamentally constrained by tokenization. Modern multilingual models, such as Qwen3 (Yang et al., 2025) and Gemma3 (Gemma Team et al., 2025), rely on subword tokenizers trained on large corpora, typically using algorithms such as Byte Pair Encoding (BPE) (Sennrich et al., 2016) or unigram language models (LMs) (Kudo, 2018). While these tokenizers capture meaningful morphological structures for well-represented languages, they often perform poorly for low-resource languages due to limited vocabulary coverage and segmentation mismatches (Bostrom and Durrett, 2020; Xue et al., 2022; Clark et al., 2022).

In this work, we focus explicitly on Indic languages. The Indic language family serves as an ideal testbed because these languages feature high morphological complexity and variable script overlap, making them susceptible to tokenizer over-fragmentation. For instance, Indic languages such as Bhojpuri and Magahi share the Devanagari script with resource-rich Hindi, causing tokenizers to apply Hindi-derived subwords. Table 1 provides quantitative evidence of this tokenization mismatch. The Qwen3 tokenizer splits 97.7% of Bhojpuri words into multiple subword tokens, with an average of 4.43 tokens per word. Given modern subword tokenizers do not generate out-of-vocabulary tokens due to byte-fallback mechanisms, the tokenizer forces representations into available subwords, masking segmentation failures beneath valid, yet linguistically incorrect, token sequences. Consequently, models must operate on fragmented and linguistically misaligned input units, degrading performance on downstream tasks.

A natural alternative is to eliminate the dependency on subword tokenization entirely. Byte-level models operate directly on raw UTF-8 representations, avoiding segmentation errors and providing robustness across languages (Xue et al., 2022; Clark et al., 2022). However, operating at the byte level introduces a granularity mismatch for word-level tasks. Non-Latin scripts such as Devanagari encode each character using multiple UTF-8 bytes, which dilutes the word-level signal across multiple positions. Hierarchical byte-level architectures, such as H-Net (Hwang et al., 2026) and Bolmo (Minixhofer et al., 2026), address this issue by dynamically grouping bytes into word-aligned chunks.

However, existing hierarchical paradigms are not suitable for adapting models to low-resource languages. They rely on (pre-)training with mas-

[1]Code: github.com/snjev310/ByteChunk.

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing, pages 48638–48651
October 24-29, 2026 ©2026 Association for Computational Linguistics

Language	Words	Qwen3 Subword			Gemma3 Subword			H-Net (Qwen3)		H-Net (Gemma3)	
		#SW	SW/W	Split (%)	#SW	SW/W	Split (%)	C/W	1:1 (%)	C/W	1:1 (%)
Hindi	281,057	1,155,744	4.11	93.7	445,511	1.59	42.1	1.24	92.5	0.98	90.4
Bhojpuri	6,665	29,491	4.43	97.7	11,708	1.76	52.9	1.17	91.0	0.95	79.6
Marathi	2,997	12,917	4.31	81.6	5,354	1.79	56.0	1.42	82.5	0.94	89.0
Magahi	7,702	37,740	4.90	97.2	13,401	1.74	52.5	1.17	87.8	0.97	91.5
Sanskrit	1,843	10,542	5.72	98.9	4,533	2.46	75.9	1.96	87.3	1.00	99.6
Urdu	14,806	44,714	3.02	95.4	28,403	1.92	60.0	1.06	93.2	0.59	0.2

Table 1: Tokenizer fragmentation and H-Net chunk alignment statistics across six Indic languages. **#SW** stands for the total number of subwords. **SW/W** and **C/W** denote the subword-to-word and chunk-to-word ratios, respectively. **Split (%)** indicates the percentage of words fragmented into two or more subword tokens, while **1:1 (%)** represents the percentage of chunks covering exactly one word.

sive token budgets, offering no feasible path for data-constrained scenarios. Furthermore, naively combining a frozen subword-based backbone with randomly initialized byte-level components creates a representational mismatch. The resulting byte-level chunk embeddings lack explicit alignment with the subword semantic space of the backbone, which prevents effective cross-lingual transfer.

In this paper, we introduce an adapted H-Net framework, a novel hybrid approach that bridges this modality gap without requiring large-scale training. First, our method initializes byte embeddings directly from the subword embeddings of the backbone. Second, it applies a chunk alignment loss to pull byte-level chunk embeddings toward pre-computed subword alignment targets. Unlike unguided byte encoding that produces incompatible representations, our alignment mechanism ensures that the frozen backbone can effectively interpret the newly formed chunks. Finally, we interleave lightweight part-of-speech (POS) supervision during training to guide boundary detection. This process allows our adapted model to bypass tokenization bottlenecks while retaining the robust linguistic knowledge of the pretrained backbone.

We evaluate our approach on POS tagging, named entity recognition (NER), and sentiment analysis across five Indic languages using two model families at three parameter scales (1.7B, 4B, and 12B). Our contributions are:

1. A novel adapted hierarchical framework that bridges byte-level inputs and frozen subword LMs without extensive training, addressing the bottleneck of tokenizer overfragmentation in low-resource languages. (§3.2)

2. The proposed architecture consistently improves zero-shot cross-lingual transfer across languages. It outperforms a subword baseline on POS tagging by up to 13.3%, achieving highly competitive target-language word-level morphological processing capabilities. (§5)

3. Our analysis shows that backbone-initialized embeddings prevent training collapse under limited data, task-guided pretraining improves chunk-level representations by up to 16.4 points, and tokenizer-aware alignment targets are essential for SentencePiece backbones. (§6)

2 Related Work

Byte-level LMs. Byte-level models eliminate multilingual NLP vocabulary bottlenecks by operating directly on raw UTF-8 bytes (Xue et al., 2022). Previous work utilizes hierarchical architectures to manage long sequences (YU et al., 2023; Pagnoni et al., 2025), group adjacent representations (Hwang et al., 2026), or align distinct modalities (Minixhofer et al., 2026). However, these methodologies are strictly constrained to high-resource settings and demand massive training corpora that range from 43B to 4T bytes. Furthermore, current frameworks focus primarily on Latin scripts, require tokenizer-derived boundaries, or rely on large-scale end-to-end pretraining of all components, rendering them impractical for low-resource target languages. Related work on vocabulary adaptation (Downey et al., 2023) transfers embedding structure across subword vocabularies, Feher et al. (2025) retrofit LLMs with dynamic tokenization, and Remy et al. (2024) adapt subword vocabularies for low-resource NLP.

Unlike these approaches, which all operate in the subword space, our framework processes raw bytes and learns word-aligned chunks without any tokenizer. Specifically, the proposed method targets low-resource Indic languages by adapting off-the-shelf, subword-based LMs. We replace the original embeddings and LM heads with H-Net byte-level modules. We then train only these newly introduced components on a minimal corpus of

500K sentences ($\sim$ 1B bytes), leaving the backbone frozen.

Tokenization & low-resource languages. Prior work identifies structural limitations of subword tokenization in low-resource settings (Ahia et al., 2023; Petrov et al., 2023). Rust et al. (2021) show that BPE tokenizers overfragment text in low-resource languages, and Adelani et al. (2022) report analogous segmentation failures for African languages. Foroutan et al. (2026) further establish that standard BPE vocabulary selection inherently disadvantages underrepresented languages under data constraints. For Indic languages, Pattnayak et al. (2025) observe that BPE tokenizers degrade zero-shot cross-lingual NER, and Kumar et al. (2024) demonstrate that POS tagging suffers directly from tokenizer overfragmentation. Brahma et al. (2026) attribute these vulnerabilities to script diversity, complex morphology, and restricted pretraining coverage. Building on these observations, we quantify this severe tokenizer fragmentation across target Indic languages in Table 1, motivating the design of our byte-level framework.

Cross-lingual transfer. Cross-lingual transfer relies on shared vocabulary and tokenizer coverage. Pires et al. (2019) show that shared subword vocabularies facilitate transfer in multilingual BERT, while Wu and Dredze (2020) find that vocabulary overlap correlates with transfer quality. Similar observations emerge across models, where lexical overlap and tokenizer coverage substantially affect cross-lingual representation alignment and downstream transfer performance (Conneau et al., 2020; Devlin et al., 2019; Rust et al., 2021; Yamaguchi et al., 2024, 2026). Consequently, for low-resource languages lacking tokenizer coverage, traditional transfer mechanisms degrade rapidly, isolating them from the benefits of pretrained models. To resolve this dependency, the proposed framework bypasses the requirement for subword overlap entirely. By aligning byte-level chunks directly with the semantic space of a frozen subword backbone, the proposed framework enables robust cross-lingual transfer without relying on shared tokenization.

3 Methodology

3.1 Preliminaries: H-Net

H-Net (Hwang et al., 2026) is a hierarchical byte-level architecture that processes raw UTF-8 bytes directly, bypassing subword tokenization. Given a byte sequence, H-Net operates in four stages.

Stage 1: Local encoder. First, the model projects each input byte x to a D-dimensional vector via a learned embedding table $\mathbf{E} \in \mathbb{R}^{256\times D}$: $\mathbf{h}^{(0)} = \mathbf{E}[\mathbf{x}] \in \mathbb{R}^{T\times D}$, where T denotes the total sequence length. A local encoder comprising L sequential Mamba (Gu and Dao, 2024) layers then contextualizes these embeddings using local neighborhood information. The representation at the l-th layer is updated via a residual connection:

$$\mathbf{h}^{(l)} = \mathbf{h}^{(l-1)} + \text{Mamba}(\mathbf{h}^{(l-1)}) \in \mathbb{R}^{T\times D},$$

where $l \in \{1, \ldots, L\}$. The final output $\mathbf{h} = \mathbf{h}^{(L)}$ provides the contextualized byte representations necessary to capture the short-range dependencies required for boundary detection.

Stage 2: Dynamic chunking. Next, the module partitions bytes into variable-length chunks. At each position t, the system computes a boundary probability based on the cosine similarity between adjacent byte representations:

$$p_t = \frac{1}{2}(1 - \cos(\mathbf{h}_t, \mathbf{h}_{t-1})).$$

A high p_t value indicates a probable word boundary, whereas a low p_t value implies a within-word continuation. The model assigns hard boundaries where $p_t \geq \tau$. For each chunk, the module selects the byte representation at the boundary position as the initial chunk embedding.

A smoothing module subsequently refines this representation by applying self-attention across all byte representations contained within the chunk:

$$\mathbf{z}_i = \text{Smooth}(\mathbf{h}_{b_i}) \in \mathbb{R}^D,$$

where b_i denotes the boundary position of chunk i.

Stage 3: LM backbone. The sequence of C chunk embeddings $\mathbf{Z} = (\mathbf{z}_1, \ldots, \mathbf{z}_C) \in \mathbb{R}^{C\times D}$ serves as input to a global LM backbone:

$$\mathbf{Z}' = \text{Backbone}(\mathbf{Z}) \in \mathbb{R}^{C\times D}. \quad (1)$$

Because the backbone processes chunk-level representations rather than subword tokens, it avoids tokenizer-induced overfragmentation.

Stage 4: Local decoder. Finally, the Exponential Moving Average (EMA) DeChunk operation (Hwang et al., 2026) distributes chunk representations back to byte positions using exponentially decaying weights determined by the distance

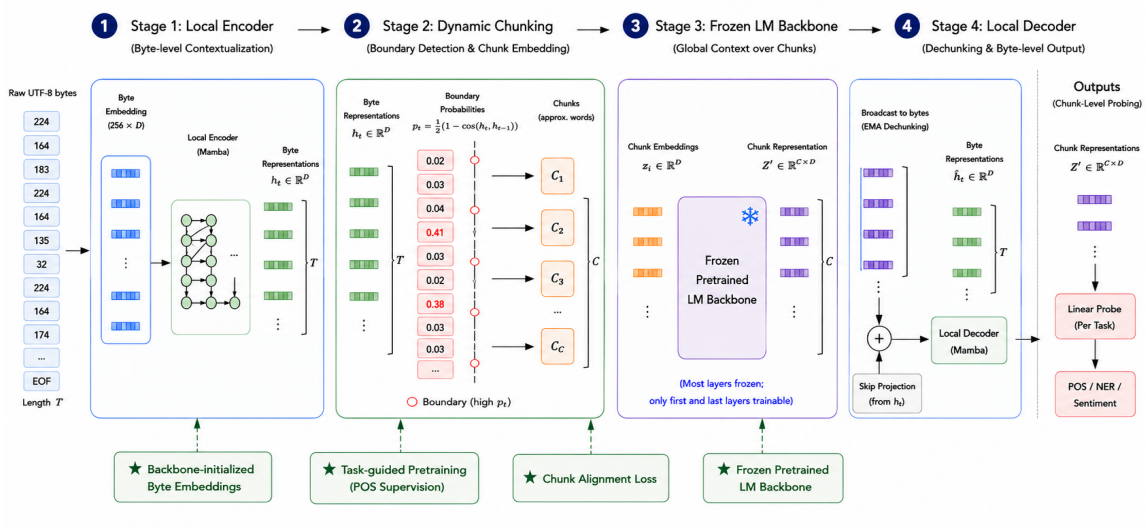

Figure 1: Overview of the proposed H-Net architecture for low-resource languages.

from chunk boundaries:

$$\hat{\mathbf{h}} = \text{Mamba}\big(\text{EMADeChunk}(\mathbf{Z}') + \text{SkipProj}(\mathbf{h})\big) \in \mathbb{R}^{T \times D} \quad (2)$$

These final representations integrate local byte-level information with global chunk-level context.

Pretraining objectives. The architecture optimizes two primary losses. It employs standard next-byte prediction via a linear byte-level LM head:

$$\mathcal{L}_{\text{AR}} = -\sum_{t=1}^{T-1} \log p(x_{t+1} \mid \hat{\mathbf{h}}_t).$$

To maintain optimal chunk sizes, a boundary ratio loss regularizes the mean boundary probability toward a target rate ρ^*, which approximately corresponds to one boundary per word:

$$\mathcal{L}_{\text{ratio}} = (\bar{p} - \rho^*)^2\,, \quad (3)$$

where $\bar{p} = \frac{1}{T}\sum_t p_t$ defines the mean boundary probability. The complete H-Net pretraining loss combines these terms:

$$\mathcal{L}_{\text{H-Net}} = \mathcal{L}_{\text{AR}} + \lambda_{\text{ratio}}\,\mathcal{L}_{\text{ratio}},$$

where λ_{ratio} is a hyperparameter controlling the boundary regularization penalty.

3.2 Adapting H-Net for low-resource languages

We adapt H-Net by reusing a frozen, subword-based pretrained LM backbone rather than pretraining the entire architecture from scratch. Although this strategy provides a computationally efficient foundation, applying it to low-resource languages poses three challenges: slow convergence caused by random byte embedding initialization, a lack of explicit word-level supervision for boundary detection, and a representational gap between byte-level chunk embeddings and the semantic space of the backbone. We introduce three targeted modifications to resolve each issue.

Backbone-initialized byte embeddings. We replace random initialization by averaging the backbone subword embeddings that contain each specific byte value:

$$\mathbf{E}[b] = \frac{1}{|\mathcal{T}_b|}\sum_{v \in \mathcal{T}_b} \mathbf{W}_v, \quad (4)$$

where $\mathcal{T}_b$ is the set of subword tokens containing byte b, and $\mathbf{W}_v$ is the backbone embedding for token v. This strategy places the byte representations within the pretrained semantic space before training, accelerating convergence under limited data.

Chunk alignment loss. To align byte-level chunks with the backbone embedding space, we introduce a loss function that pulls each chunk embedding toward a byte alignment target, defined as the average of the corresponding backbone subword embeddings:

$$\mathcal{L}_{\text{align}} = \frac{1}{C}\sum_{i=1}^{C} \left(1 - \cos(\mathbf{z}_i, \bar{\mathbf{t}}_i)\right), \quad (5)$$

where $\bar{\mathbf{t}}_i$ is the target for chunk i. This ensures the frozen backbone "interprets" the new chunk em-

beddings without requiring extensive fine-tuning, thereby facilitating effective transfer.

Task-guided pretraining. The standard boundary ratio loss (Eq. 3) often fails to learn linguistic boundaries in low-resource settings due to insufficient supervision signals. To mitigate this, we interleave POS supervision on a target language. Every k steps (with $k = 10$ in this study), the model trains a linear POS head jointly with the encoder:

$$\mathcal{L}_{\text{POS}} = -\sum_{t \in \mathcal{V}} \log p(\text{tag}_t \mid \hat{\mathbf{h}}_t), \tag{6}$$

where $\mathcal{V}$ defines the valid byte positions (excluding padding and spaces). The two objectives use separate batches and forward/backward passes. Autoregressive steps process FineWeb2 Hindi text with $\mathcal{L} = \mathcal{L}_{\text{AR}} + \lambda_{\text{ratio}}\mathcal{L}_{\text{ratio}} + \lambda_{\text{align}}\mathcal{L}_{\text{align}}$, while POS steps process Hindi CoNLL-U sentences with $\mathcal{L} = \lambda_{\text{POS}}\mathcal{L}_{\text{POS}}$ only.

We discard this head after pretraining. The POS data acts as a structural signal for learning span representations (Tenney et al., 2019; Hewitt and Manning, 2019).

Full pretraining objective. The complete pretraining loss combines our novel contributions ($\mathcal{L}_{\text{POS}}$, $\mathcal{L}_{\text{align}}$) with the standard H-Net methodology ($\mathcal{L}_{\text{AR}}, \mathcal{L}_{\text{ratio}}$):

$$\begin{aligned}\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{AR}} + \lambda_{\text{ratio}}\,\mathcal{L}_{\text{ratio}} \\ + \lambda_{\text{POS}}\,\mathcal{L}_{\text{POS}} + \lambda_{\text{align}}\,\mathcal{L}_{\text{align}},\end{aligned}$$

where λ_{POS} and λ_{align} are hyperparameters controlling the POS supervision and alignment terms.

4 Experimental Setup

Target languages. We evaluate the proposed framework on five low-resource Indic languages: Bhojpuri, Marathi, Magahi, Sanskrit, and Urdu. We select Hindi as the source adaptation language because it offers sufficient training data while sharing linguistic roots, structural characteristics, or the Devanagari script with the target variants. This typological proximity allows Hindi to function as an effective anchor for cross-lingual transfer.

Models. We utilize pretrained base model variants from Qwen3 (Yang et al., 2025) and Gemma3 (Gemma Team et al., 2025), with two scale variants from each model family: Qwen3-1.7B, Qwen3-4B, Gemma3-4B, and Gemma3-12B. These model families provide a clear contrast in their multilingual pretraining corpora, as Gemma3 incorporates broader multilingual data than Qwen3. Qwen3 is pretrained on 119 languages, whereas Gemma3 is pretrained on 140 languages. Neither model reports explicit pretraining coverage for Bhojpuri, Magahi, and Sanskrit. Marathi is likely covered in both families, as it uses the Devanagari script and is closely related to Hindi, which is well represented in both pretraining corpora. Urdu is likely covered because it uses the Nastaliq script, which is also used by Arabic and Persian.

Training. We use 500K Hindi sentences from the FineWeb2 `hin_Deva` subset (1.02B bytes) (Penedo et al., 2025) for adaptation. During training, we unfreeze the first and the last transformer layers of the backbone model to optimize computational efficiency while ensuring effective adaptation (Remy et al., 2024; Yamaguchi et al., 2025). For task-guided pretraining (Eq. 6), we use POS data from the Universal Dependencies (UD) Hindi treebank (Nivre et al., 2020) (`hi-hdtb`, 13K sentences). Following adaptation, we train a two-layer MLP probe on frozen chunk representations for downstream evaluation. Full hyperparameter details are reported in Appendix A.3.

Baselines. We compare the proposed approach (H-Net) against two baseline configurations:

- **Subword** (Sub): The frozen pretrained backbone acts as a feature extractor. For word-level tasks, we train a two-layer MLP probe on the last-layer hidden states. For the sentence-level task, we train the probe on the mean-pooled last-layer hidden states over all non-padding tokens.
- **Continued pretraining** (CPT): We continually pretrain the backbone on the identical 500K Hindi sentences used for the proposed method. We unfreeze the first and the last layers to match the setup as the proposed method. This baseline isolates the contribution of the byte-level architecture by controlling for the effect of additional Hindi exposure.

Evaluation tasks, metrics, and protocols. We primarily evaluate the models on two word-level sequence labeling tasks: **POS tagging** using UD treebanks (Nivre et al., 2020), and **NER** using WikiANN (Rahimi et al., 2019). These tasks directly assess whether the byte-level chunks of the

	Language	Script	Qwen3-1.7B			Qwen3-4B			Gemma3-4B			Gemma3-12B		
			Sub	CPT	H-Net	Sub	CPT	H-Net	Sub	CPT	H-Net	Sub	CPT	H-Net
	Part-of-Speech Tagging (Accuracy %)													
①	Bhojpuri	Devanagari	43.8	48.1	**54.4**	47.9	49.3	**57.9**	**59.2**	56.9	56.9	55.2	56.4	**56.8**
	Marathi	Devanagari	39.1	35.3	**46.3**	44.0	36.8	**46.0**	55.6	44.5	**59.2**	53.9	41.7	**58.4**
	Magahi	Devanagari	41.0	40.9	**53.1**	45.3	43.2	**46.5**	**55.3**	51.9	53.1	49.8	50.4	**52.3**
	Sanskrit	Devanagari	21.6	30.2	**30.6**	25.7	**35.1**	33.2	21.9	**32.8**	30.6	30.6	31.1	**32.7**
	Urdu	Nastaliq	38.8	48.1	**52.1**	48.6	51.0	**55.6**	46.1	**52.4**	52.1	**70.2**	65.6	53.1
	Average		36.9	40.5	**47.3**	42.3	43.1	**47.8**	47.6	47.7	**50.4**	**51.9**	49.0	50.7
	Named Entity Recognition (Entity-level F1 %)													
②	Urdu	Nastaliq	**12.7**	5.2	12.1	2.1	4.0	**14.4**	8.1	3.8	**9.4**	5.7	4.9	**12.8**
	Marathi	Devanagari	**14.8**	14.2	14.3	13.0	13.0	**13.3**	22.4	16.6	**27.7**	29.1	21.6	**32.5**
	Sanskrit	Devanagari	9.3	10.4	**19.9**	10.5	10.2	**15.4**	15.1	10.7	**19.3**	12.2	**14.6**	14.5
	Average		12.3	9.9	**15.4**	8.5	9.1	**14.4**	15.2	10.4	**18.8**	15.7	13.7	**19.9**
	Sentiment Analysis (Accuracy %)													
③	Marathi	Devanagari	**72.7**	53.6	63.3	**54.4**	54.3	**54.4**	**91.8**	85.3	78.0	**90.0**	85.5	75.0
	Urdu	Nastaliq	**49.3**	**49.3**	**49.3**	49.3	49.5	**51.0**	**69.3**	50.0	65.4	**84.0**	79.8	68.0
	Average		**61.0**	51.5	56.3	51.9	51.9	**52.7**	**80.6**	67.7	71.7	**87.0**	82.7	71.5

Table 2: Zero-shot performance across three downstream tasks. Bold indicates the best result among Sub, CPT, and H-Net per task and model family.

adapted architecture produce linguistically meaningful word-level representations. As a supplementary evaluation, we include **sentiment analysis** using IndicSentiment (Doddapaneni et al., 2023), a sentence-level semantic task, to examine whether the byte-level representations generalize beyond morphological structure to broader semantic understanding. For POS tagging, we report token-level accuracy. For NER, we report entity-level F1. For sentiment analysis, we report accuracy.

For the proposed framework, we evaluate at the chunk level: each contextualized chunk embedding $\mathbf{z}'_i \in \mathbf{Z}'$ (Eq. 1) approximates one word and receives the label of the word that contributes the majority of its bytes. Chunks covering only spaces or punctuation are excluded (ignore_index$=-100$). For subword baselines, we evaluate only the first subword token per word (Pires et al., 2019; Wolf et al., 2020; Wu and Dredze, 2020).

We conduct all evaluations in a zero-shot setting. We apply a probe trained on Hindi representations directly to the target languages without any target-language supervision. Further details on dataset statistics, language morphology, and script characteristics are provided in Appendix A.1.

5 Results

POS tagging. Our adapted H-Net models consistently outperform the corresponding subword baselines on POS tagging (Table 2 ①). Specifically, H-Net (Qwen3-1.7B) surpasses the subword baseline in all five low-resource languages with an average gain of 10.4 points, and H-Net (Qwen3-4B) achieves an average gain of 5.5 points. Similarly, H-Net (Gemma3-4B) outperforms the subword baseline on three out of five languages (+2.8 points average), with the maximal gain observed on Sanskrit (+8.7 points). In contrast, CPT yields negligible gains across models providing an average increase of only 0.8 points for Qwen3-4B and 0.1 points for Gemma3-4B. This confirms that additional Hindi exposure alone cannot resolve the tokenizer bottleneck.

The advantage of H-Net peaks where tokenizer overfragmentation is most severe. Magahi provides a clear example of this dynamic. The Qwen3 tokenizer splits Magahi words into an average of 4.90 tokens (Table 1), causing severe degradation for the subword models, whereas H-Net (Qwen3-1.7B) achieves a 12.1 point performance gain.

Beyond fragmentation, results on Urdu suggest that byte-level transfer in H-Net can generalize beyond shared byte sequences. Despite the use of the Perso-Arabic Nastaliq script, H-Net (Qwen3-1.7B) achieves 52.1% zero-shot accuracy (+13.3 points). We hypothesize that linguistic relatedness plays a role in morphological transfer even when byte patterns differ.

NER. This structural advantage of our adapted H-Net models extends to NER. Table 2 ② shows that H-Net broadly outperforms the subword baseline (Table 2). The gains average 3.1 points for Qwen3-1.7B and 3.6 points for Gemma3-4B, reaching a maximum improvement on Sanskrit (10.6 points, Qwen3-1.7B), where tokenization is most fragmented (5.72 tokens/word; Table 1). Conversely, CPT often underperforms both the proposed method and the subword baseline across this task. This failure reinforces our finding in POS tagging: simply exposing a model to more source-language data without hierarchical byte-level boundary adaptation degrades the cross-lingual representation of entity-level semantics.

Sentiment analysis. Unlike the word-level tasks, subword baselines dominate sentence-level sentiment classification (Table 2 ③). Larger subword models (Gemma3-4B at 80.6%, Gemma3-12B at 87.0%) outperform all H-Net variants. In addition, CPT underperforms the subword baseline across different backbones. This outcome aligns with the finding that semantic tasks benefit from pretraining scale rather than Hindi-specific adaptation (Raffel et al., 2020; Kaplan et al., 2020; Xue et al., 2021). H-Net, adapted on only 500K Hindi sentences, has substantially less exposure to sentiment-bearing vocabulary than Qwen3 or Gemma3, which are trained on trillions of tokens across hundreds of languages. All Qwen3 models collapse to near-random performance on Urdu (∼49%), indicating that sentiment-bearing lexical cues do not transfer reliably across divergent scripts such as Devanagari and Nastaliq in a zero-shot setting.

Takeaways. H-Net achieves peak performance for word-level tasks (POS and NER), successfully bypassing tokenizer fragmentation to capture syntactic and entity-level structures. However, it does not outperform the subword baseline in sentiment analysis. Furthermore, CPT consistently underperforms H-Net across tasks, confirming that the observed gains arise directly from the H-Net architecture rather than additional Hindi training exposure. Overall, these results support our hypothesis (§1): hierarchical byte-level models benefit tasks that depend on accurate morphological structure, such as POS tagging and NER, where word-boundary quality directly affects label prediction. In contrast, sentiment analysis relies more on opinion-bearing vocabulary and semantics derived from large-scale pretraining corpora.

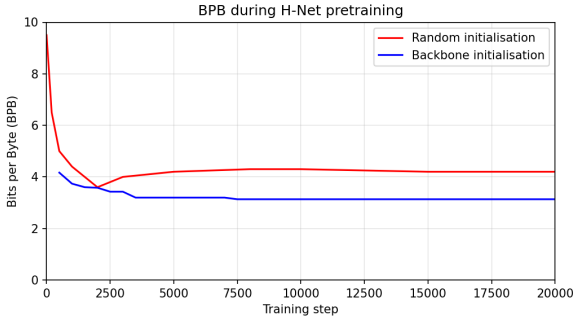

Figure 2: BPB during H-Net pretraining under random (red) vs backbone (blue) byte embedding initialization.

In-language Hindi evaluation is provided in §A.4.

6 Analysis

This section verifies the specific contribution of each architectural adaptation introduced in §3.2. Due to resource constraints, we use only Qwen3-1.7B for our analysis.

Backbone-initialized byte embeddings. We evaluate the necessity of initializing byte embeddings directly from the subword representations of the backbone (Eq. 4). Standard random initialization introduces an immediate semantic mismatch with the pretrained backbone, which severely hinders training dynamics (Figure 2). Specifically, the bits-per-byte (BPB) metric for the random initialization model begins at approximately 9.5, drops sharply to 3.6, and subsequently rises to plateau at 4.2, showing no further improvement over 20K steps. In contrast, our backbone initialization mitigates this mismatch from the outset; it starts at a BPB of 4.17 and converges smoothly to 3.13 within 7.5K steps, achieving a 25% relative reduction.

The autoregressive loss (i.e., BPB) converges normally in both cases (Figure 2), confirming that the model memorizes Hindi byte sequences regardless of initialization. However, the routing module fails to learn word-level boundaries under random initialization: the boundary ratio fluctuates between 0.43 and 0.49 throughout training (Figure 3, left). This constitutes a substantial deviation from the target of $\rho^* = 0.056$, which corresponds to the expected frequency of one boundary per word. Consequently, the average chunk length remains between 200 and 1200 bytes, well above the target of 18 bytes per word (Figure 3, right). Backbone initialization instead places byte embeddings in the correct semantic neighborhood from the first training step, the boundary ratio

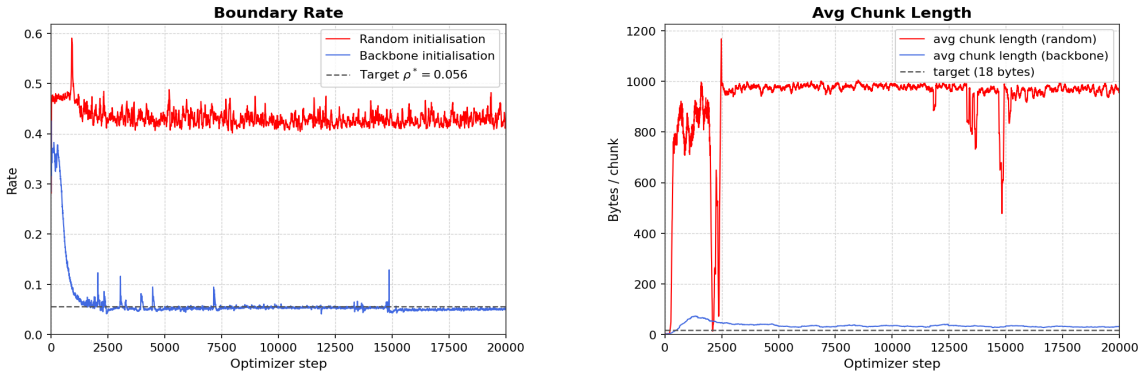

Figure 3: Training dynamics with random (red) vs backbone (blue) byte embedding initialization. **Left**: average boundary ratio over 20,000 steps oscillates at 0.43–0.49. **Right**: average chunk length.

Pretraining	Byte-level	Chunk-level
Without POS	32.5	43.1
With POS	32.8	59.5

Table 3: Hindi POS accuracy (%) with and without task-guided pretraining (Qwen3-1.7B backbone).

Language	Full model		w/o align loss	
	C/W	1:1%	C/W	1:1%
Hindi	1.24	92.5	0.05	5.5
Bhojpuri	1.17	91.0	0.06	3.4
Marathi	1.42	82.5	0.13	0.0
Magahi	1.17	87.8	0.07	0.0
Sanskrit	1.96	87.3	0.13	0.4
Urdu	1.06	93.2	0.07	27.5

Table 4: Chunk-word alignment statistics for the full H-Net model vs without chunk alignment loss ($\lambda_{\text{align}} = 0$).

drops from 0.50 at initialization to ≈0.05 (target: $\rho^* = 0.056$), providing a meaningful gradient signal to the alignment loss immediately.

Task-guided pretraining. We next evaluate the impact of task-guided pretraining on the structural quality of chunk boundaries. Table 3 demonstrates that introducing explicit POS supervision improves chunk-level accuracy by 16.4 points (from 43.1% to 59.5%) while maintaining raw byte prediction capabilities (a 0.3 point increase in byte-level accuracy). This gain highlights the limitations of relying solely on the default boundary ratio loss. As defined in §3.1, the default objective (Eq. 3) targets a global average boundary rate ($\rho^* = 0.056$, corresponding to approximately one boundary per word). While the network successfully achieves this global statistical target, it frequently misaligns with true linguistic boundaries at the local level due to insufficient structural signals. Interleaving POS supervision resolves this misalignment by forcing the encoder to match gold standard Hindi word boundaries. Furthermore, this structural improvement generalizes across tasks. The framework trained with POS supervision outperforms subword baselines on NER (Table 2), proving that linguistically structured boundaries transfer effectively even in the absence of task-specific morphological supervision.

Chunk alignment loss. Finally, we analyze the necessity of the chunk alignment loss (Eq. 5) in bridging the representational gap between the newly introduced byte-level modules and the frozen backbone. Table 4 shows that without the alignment loss, C/W collapses to a range of 0.06 to 0.13 across all six languages. This indicates that each chunk spans between 8 and 17 words on average, heavily deviating from the target ratio of approximately 1.0. Furthermore, the 1:1% metric drops to near zero, reflecting a complete failure of word-level boundary detection. These results demonstrate that without explicit alignment, chunk embeddings occupy a completely different region of the embedding space compared to the subword tokens the backbone processed during pretraining. Consequently, the frozen backbone produces degraded contextualized representations.

The root cause of this boundary detection failure is the geometry of the byte alignment targets determined by the backbone tokenizer. The Qwen3 tokenizer assigns the space character a dedicated token with a distinct embedding, creating a space byte target geometrically separate from Devanagari byte targets (cosine similarity of 0.40). In contrast, the Gemma3 tokenizer absorbs spaces into word-initial tokens (e.g., `_word`). Consequently, the isolated space byte lacks a dedicated embedding and receives a near-zero target vector (norm

of 0.007). This vector yields a cosine similarity of 0.487 with Devanagari byte targets, eliminating the boundary signal ($p_{\max} = 0.16$). While the 0.087 similarity difference between Qwen3 and Gemma3 appears small, a value of 0.487 approaches the 0.5 random baseline for these embeddings, and masks the boundary entirely. This lack of contrast prevents the routing module from distinguishing genuine space-induced word boundaries from standard within-word byte continuations. To resolve this limitation, the proposed method computes the space target as the centroid of all word-initial token embeddings. This intervention restores geometric contrast between space and text bytes (cosine similarity of 0.032) and recovers word-level chunking.

7 Conclusion

We presented an adapted H-Net framework that addresses the tokenizer overfragmentation bottleneck for low-resource Indic languages. By combining backbone-initialized byte embeddings, interleaved POS supervision, and a tokenizer-aware chunk alignment loss, the proposed method produces word-aligned chunk representations from only 500K Hindi sentences. Experiments across five languages show that the adapted architecture improves performance on word-level morphological tasks, yielding average gains of 10.4 points on POS tagging and 7.8 points on NER. This framework provides a computationally efficient mechanism to extend the structural knowledge of high-resource LMs to extremely low-resource languages, entirely bypassing the dependency on shared tokenization vocabularies.

Limitations

Pretraining language. In our current setup, we only consider Indic low-resource languages as our family of interest. H-Net is pretrained exclusively on Hindi, a high-resource Devanagari language. Transfer to target languages relies on script sharing (Bhojpuri, Magahi, Marathi, Sanskrit) or linguistic relatedness (Urdu). Languages that share neither Hindi's script nor its morphological structure are unlikely to benefit from this approach without additional pretraining data.

Script coverage. Our evaluation covers only two scripts: Devanagari and Nastaliq. H-Net's boundary detection exploits Devanagari-specific byte patterns (matra structure), and its effectiveness for other non-Latin scripts, such as Tamil, Chinese, or Bengali, remains to be validated.

Task scope. Our evaluation covers POS tagging, NER, and sentiment analysis tasks. Performance on tasks requiring deeper semantic understanding, such as machine translation or natural language inference, remains unexplored. Evaluation on machine translation using FLORES-200 (Goyal et al., 2022) or IndicMT Eval (Sai B et al., 2023), potentially combined with related-source mixing (Kumar et al., 2026), is left for future work.

Efficiency. We do not report wall-clock inference time. H-Net processes raw bytes via a local Mamba encoder, which incurs additional computational overhead compared to subword tokenization. However, dynamic chunking reduces the sequence length fed to the backbone by $\sim 18\times$, partially compensating for this overhead. A systematic efficiency comparison is left for future work.

Compute asymmetry. H-Net and the CPT baseline use the same training data but different compute budgets. H-Net processes 1.02B byte tokens while CPT processes approximately 7.5M subword tokens. A fully compute-matched comparison remains an open question.

Acknowledgements

We thank the anonymous reviewers for their insightful and constructive comments. This work was conducted while SK was a visiting student at the University of Sheffield. SK thanks Prof. Preethi Jyothi (IIT Bombay) for supporting the research visit. SK gratefully acknowledges support from the Amazon–IITB AI/ML initiative towards conference expenses, the IBM AI Horizon Networks–IIT Bombay initiative for financial support towards the visit to the University of Sheffield, and the TCS Research Foundation for Ph.D. fellowship support for conducting research on extremely low-resource Indian languages. AY is supported by the Engineering and Physical Sciences Research Council (EPSRC) [grant number EP/W524360/1] and the Japan Student Services Organization (JASSO) Student Exchange Support Program (Graduate Scholarship for Degree Seeking Students).

References

David Ifeoluwa Adelani, Graham Neubig, Sebastian Ruder, Shruti Rijhwani, Michael Beukman, Chester

Palen-Michel, Constantine Lignos, Jesujoba O. Alabi, Shamsuddeen H. Muhammad, Peter Nabende, Cheikh M. Bamba Dione, Andiswa Bukula, Rooweither Mabuya, Bonaventure F. P. Dossou, Blessing Sibanda, Happy Buzaaba, Jonathan Mukiibi, Godson Kalipe, Derguene Mbaye, and 26 others. 2022. MasakhaNER 2.0: Africa-centric transfer learning for named entity recognition. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4488–4508, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Orevaoghene Ahia, Sachin Kumar, Hila Gonen, Jungo Kasai, David Mortensen, Noah Smith, and Yulia Tsvetkov. 2023. Do all languages cost the same? tokenization in the era of commercial language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9904–9923, Singapore. Association for Computational Linguistics.

Kaj Bostrom and Greg Durrett. 2020. Byte pair encoding is suboptimal for language model pretraining. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4617–4624, Online. Association for Computational Linguistics.

Maharaj Brahma, N J Karthika, Rajat Verma, Nagasai Saketh Naidu, Rohit Saluja, Maunendra Sankar Desarkar, and Ganesh Ramakrishnan. 2026. Multilingual tokenization through the lens of Indian languages: Challenges and insights. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 32614–32632, San Diego, California, United States. Association for Computational Linguistics.

Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting. 2022. Canine: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics*, 10:73–91.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Sumanth Doddapaneni, Rahul Aralikatte, Gowtham Ramesh, Shreya Goyal, Mitesh M. Khapra, Anoop Kunchukuttan, and Pratyush Kumar. 2023. Towards leaving no Indic language behind: Building monolingual corpora, benchmark and models for Indic languages. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12402–12426, Toronto, Canada. Association for Computational Linguistics.

C.M. Downey, Terra Blevins, Nora Goldfine, and Shane Steinert-Threlkeld. 2023. Embedding structure matters: Comparing methods to adapt multilingual vocabularies to new languages. In *Proceedings of the 3rd Workshop on Multi-lingual Representation Learning (MRL)*, pages 268–281, Singapore. Association for Computational Linguistics.

Darius Feher, Ivan Vulić, and Benjamin Minixhofer. 2025. Retrofitting large language models with dynamic tokenization. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 29866–29883, Vienna, Austria. Association for Computational Linguistics.

Negar Foroutan, Clara Meister, Debjit Paul, Joel Niklaus, Sina Ahmadi, Antoine Bosselut, and Rico Sennrich. 2026. Parity-aware byte-pair encoding: Improving cross-lingual fairness in tokenization. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7514–7538, San Diego, California, United States. Association for Computational Linguistics.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, and 197 others. 2025. Gemma 3 technical report. *Preprint*, arXiv:2503.19786.

Naman Goyal, Cynthia Gao, Vishrav Chaudhary, Peng-Jen Chen, Guillaume Wenzek, Da Ju, Sanjana Krishnan, Marc'Aurelio Ranzato, Francisco Guzmán, and Angela Fan. 2022. The Flores-101 evaluation benchmark for low-resource and multilingual machine translation. *Transactions of the Association for Computational Linguistics*, 10:522–538.

Albert Gu and Tri Dao. 2024. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

Sukjun Hwang, Brandon Wang, and Albert Gu. 2026. Dynamic chunking for end-to-end hierarchical sequence modeling. In *The Fourteenth International Conference on Learning Representations*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *Preprint*, arXiv:2001.08361.

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)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

Sanjeev Kumar, Preethi Jyothi, and Pushpak Bhattacharyya. 2024. Part-of-speech tagging for extremely low-resource Indian languages. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 14422–14431, Bangkok, Thailand. Association for Computational Linguistics.

Sanjeev Kumar, Preethi Jyothi, and Pushpak Bhattacharyya. 2026. SrcMix: Mixing of related source languages benefits extremely low-resource machine translation. In *Findings of the Association for Computational Linguistics: EACL 2026*, pages 6306–6323, Rabat, Morocco. Association for Computational Linguistics.

Benjamin Minixhofer, Tyler Murray, Tomasz Limisiewicz, Anna Korhonen, Luke Zettlemoyer, Noah A. Smith, Edoardo M. Ponti, Luca Soldaini, and Valentin Hofmann. 2026. Bolmo: Byteifying the next generation of language models. *Preprint*, arXiv:2512.15586.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal Dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4034–4043, Marseille, France. European Language Resources Association.

Artidoro Pagnoni, Ramakanth Pasunuru, Pedro Rodriguez, John Nguyen, Benjamin Muller, Margaret Li, Chunting Zhou, Lili Yu, Jason E Weston, Luke Zettlemoyer, Gargi Ghosh, Mike Lewis, Ari Holtzman, and Srini Iyer. 2025. Byte latent transformer: Patches scale better than tokens. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9238–9258, Vienna, Austria. Association for Computational Linguistics.

Priyaranjan Pattnayak, Hitesh Patel, and Amit Agarwal. 2025. Tokenization matters: Improving zero-shot ner for indic languages. In *2025 IEEE International Conference on Electro Information Technology (eIT)*, pages 456–462.

Guilherme Penedo, Hynek Kydlíček, Vinko Sabolčec, Bettina Messmer, Negar Foroutan, Amir Hossein Kargaran, Colin Raffel, Martin Jaggi, Leandro Von Werra, and Thomas Wolf. 2025. FineWeb2: One pipeline to scale them all — adapting pre-training data processing to every language. In *Second Conference on Language Modeling*.

Aleksandar Petrov, Emanuele La Malfa, Philip Torr, and Adel Bibi. 2023. Language model tokenizers introduce unfairness between languages. In *Advances in Neural Information Processing Systems*, volume 36, pages 36963–36990. Curran Associates, Inc.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, Florence, Italy. Association for Computational Linguistics.

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.

Afshin Rahimi, Yuan Li, and Trevor Cohn. 2019. Massively multilingual transfer for NER. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 151–164, Florence, Italy. Association for Computational Linguistics.

François Remy, Pieter Delobelle, Hayastan Avetisyan, Alfiya Khabibullina, Miryam de Lhoneux, and Thomas Demeester. 2024. Trans-tokenization and cross-lingual vocabulary transfers: Language adaptation of LLMs for low-resource NLP. In *First Conference on Language Modeling*.

Phillip Rust, Jonas Pfeiffer, Ivan Vulić, Sebastian Ruder, and Iryna Gurevych. 2021. How good is your tokenizer? on the monolingual performance of multilingual language models. 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)*, pages 3118–3135, Online. Association for Computational Linguistics.

Ananya Sai B, Tanay Dixit, Vignesh Nagarajan, Anoop Kunchukuttan, Pratyush Kumar, Mitesh M. Khapra, and Raj Dabre. 2023. IndicMT eval: A dataset to meta-evaluate machine translation metrics for Indian languages. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14210–14228, Toronto, Canada. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725,

Berlin, Germany. Association for Computational Linguistics.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Shijie Wu and Mark Dredze. 2020. Are all languages created equal in multilingual BERT? In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 120–130, Online. Association for Computational Linguistics.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022. ByT5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306.

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, Online. Association for Computational Linguistics.

Atsuki Yamaguchi, Terufumi Morishita, Aline Villavicencio, and Nikolaos Aletras. 2025. Adapting chat language models using only target unlabeled language data. *Transactions on Machine Learning Research*.

Atsuki Yamaguchi, Aline Villavicencio, and Nikolaos Aletras. 2024. An empirical study on cross-lingual vocabulary adaptation for efficient language model inference. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 6760–6785, Miami, Florida, USA. Association for Computational Linguistics.

Atsuki Yamaguchi, Aline Villavicencio, and Nikolaos Aletras. 2026. How can we effectively expand the vocabulary of LLMs with 0.01GB of target language text? *Computational Linguistics*, 52(1):295–330.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

LILI YU, Daniel Simig, Colin Flaherty, Armen Aghajanyan, Luke Zettlemoyer, and Mike Lewis. 2023. Megabyte: Predicting million-byte sequences with multiscale transformers. In *Advances in Neural Information Processing Systems*, volume 36, pages 78808–78823. Curran Associates, Inc.

A Appendix

A.1 Language complexity

The selected languages span a broad range of morphological complexity and script variation. Hindi and Urdu are closely related languages that are largely mutually intelligible in speech, differing primarily in script (Devanagari vs. Nastaliq) and formal vocabulary. Bhojpuri and Magahi are closely related to Hindi, sharing similar morphological structure and the Devanagari script, but lack standardized orthography and large-scale NLP resources. Marathi exhibits richer nominal and verbal inflection than Hindi, including a larger inventory of case markers. Sanskrit is the most morphologically complex language in our study: a classical language with highly synthetic morphology, extensive sandhi phenomena (phonological fusion at word boundaries), and relatively free word order, making it particularly challenging for both tokenization and byte-level boundary detection. Urdu presents an additional challenge due to its Nastaliq script, which differs substantially from the Devanagari script used in the pretraining data.

A.2 Language and Dataset

We evaluate across five Indic languages spanning two scripts and varying levels of morphological complexity, resource availability, and typological proximity to Hindi. Bhojpuri and Magahi are extremely low-resource Indo-Aryan languages closely related to Hindi and written in Devanagari, making them suitable for evaluating cross-lingual transfer under severe tokenizer overfragmentation. Marathi also uses Devanagari but exhibits richer inflectional morphology and greater lexical divergence from Hindi. Sanskrit is highly morphologically rich, with extensive inflection and compounding, leading to particularly severe subword fragmentation. Urdu differs both linguistically and script-wise, using the Perso-Arabic Nastaliq script

rather than Devanagari, which poses a challenging setting for cross-script transfer.

For POS tagging, we use Universal Dependencies (UD) treebanks (Nivre et al., 2020). Hindi (HI-HDTB) serves as the source language and contains 13,306 training sentences and 1,684 test sentences (281,057 and 35,430 words, respectively). The remaining languages are substantially smaller: Bhojpuri (BHO-BHTB, 357 sentences, 6,665 words), Marathi (MR-UFAL, 373 sentences, 3,253 words), Magahi (MAG-MGTB, 550 sentences, 7,702 words), Sanskrit (SA-UFAL, 230 sentences, 2,102 words), and Urdu (UR-UDTB, 535 sentences, 14,806 words). These dataset sizes reflect the extremely limited supervision available for many Indic languages.

For NER, we use WikiANN (Rahimi et al., 2019). Hindi contains 5,000 training sentences, while Urdu and Marathi each contain 1,000 test sentences, and Sanskrit contains only 100 test sentences. For sentiment analysis, we use IndicSentiment (Doddapaneni et al., 2023) with 1,000 examples each for Hindi, Marathi, and Urdu using binary positive/negative classification.

Together, these datasets provide evaluation across morphologically grounded sequence-labeling tasks (POS and NER) and semantic classification (sentiment analysis), under both cross-lingual and cross-script transfer settings.

A.3 Training Details

We pretrain H-Net using AdamW (lr $= 5 \times 10^{-5}$, weight_decay $= 0.01$, max_grad_norm $= 0.5$) with cosine learning rate decay and 5% linear warmup. We use a batch size of 8 sequences of 2,048 bytes (2 sequences $\times$ 4 gradient accumulation steps) in bfloat16 precision. The local encoder and decoder each use a single Mamba layer (d_state $= 16$, d_conv $= 4$, expand $= 2$ for Qwen; expand $= 1$ for Gemma). For downstream evaluation, we train a two-layer MLP probe (d_model $\rightarrow$ 256 $\rightarrow$ n_tags, GELU, LayerNorm) using AdamW (lr $= 2 \times 10^{-4}$, weight_decay $= 0.01$, cosine decay, 5% warmup) for 10 epochs (POS, NER) or 10 epochs (sentiment). All remaining hyperparameters are listed in Table 5.

All H-Net byte-level components are pretrained on a single NVIDIA A100 GPU, taking approximately 26 hours per backbone. Downstream probe training is performed once on Hindi and evaluated directly on all target languages without retraining, taking approximately 10–30 minutes.

Setting	Hyperparameter	Value
H-Net pretrain	Optimiser	AdamW
	Learning rate	5×10^{-5}
	Weight decay	0.01
	Max grad norm	0.5
	LR schedule	Cosine + 5% warmup
	Batch size	8 sequences
	Sequence length	2,048 bytes
	Precision	bfloat16
	λ_{ratio}	0.8 (follow Hwang et al. (2026))
	λ_{align}	0.3
	λ_{POS}	0.3
	POS frequency	every 10 AR steps
	Target rate ρ^*	0.056
	Mamba d_state	16
	Mamba d_conv	4
	Mamba expand (Qwen)	2
	Mamba expand (Gemma)	1
CPT baseline	Optimiser	AdamW
	Learning rate	5×10^{-5}
	Weight decay	0.01
	LR schedule	Cosine + 5% warmup
	Unfrozen layers	first & last
Probe (POS/NER)	Optimiser	AdamW
	Learning rate	2×10^{-4}
	Weight decay	0.01
	Batch size	32
	Epochs	10
Probe (Sentiment)	Optimiser	AdamW
	Learning rate	2×10^{-4}
	Weight decay	0.01
	Batch size	16
	Epochs	10

Table 5: Hyperparameters for H-Net pretraining, CPT baseline, and downstream probe training.

A.4 In-Language Hindi Evaluation

Table 6 reports in-language Hindi evaluation on the held-out UD Hindi test set. For NER, H-Net outperforms the subword baseline across all models (+5.3, +9.4, +3.2 pp), confirming that byte-level chunk representations benefit entity boundary detection even at lower fragmentation rates (2.61 tok/word). For POS tagging, H-Net underperforms on Hindi for most models. Hindi is the adaptation language with relatively low fragmentation, so the subword baseline has a full advantage in the supervised setting. The zero-shot setting reverses this pattern, with H-Net winning across all five target languages where fragmentation of the standard tokenizer is higher (3.02–5.72 tok/word). For sentiment, H-Net underperforms consistently relative to target-language results, confirming that semantic tasks benefit more from the training data scale rather than effectively processing their morphological structure.

B Use of Generative AI Tools

The authors acknowledge the use of LLMs in preparing this work. In particular, GPT-4o was used to improve the grammar, clarity, and organization of the manuscript, as well as to assist with

Task	Model	Sub	H-Net
POS	Qwen3-1.7B	**65.3**	59.5
	Qwen3-4B	71.6	**72.1**
	Gemma3-4B	**94.3**	87.4
NER	Qwen3-1.7B	24.6	**29.9**
	Qwen3-4B	25.2	**34.6**
	Gemma3-4B	52.2	**55.4**
Sentiment	Qwen3-1.7B	**88.5**	71.5
	Qwen3-4B	**89.5**	84.5
	Gemma3-4B	**95.0**	88.5

Table 6: In-language Hindi evaluation on the held-out UD Hindi test set. Bold indicates the best result per row.

code debugging. All technical content, experimental design, implementation decisions, and analysis are implemented and verified by the authors.