Adaptive Targeted Dynamic Chunking for Tokenization-Free Hierarchical Model

1st Thang Dang
Fujitsu Research of America
Pittsburgh, PA, USA
tdang@fujitsu.com

2nd Akira Nakagawa
Fujitsu Limited
Kawasaki, Kanagawa, Japan
anaka@fujitsu.com

3rd Kenichi Kobayashi
Fujitsu Limited
Kawasaki, Kanagawa, Japan
kenichi@fujitsu.com

4th Koichi Shirahata
Fujitsu Limited
Kawasaki, Kanagawa, Japan
k.shirahata@fujitsu.com

***Abstract*—Tokenization-free hierarchical models are emerging as a promising alternative to traditional Large Language Models (LLMs), addressing inherent preprocessing issues such as vocabulary design complexity, out-of-vocabulary (OOV) errors, and language-specific constraints. However, a significant challenge in these byte-level methods is the optimization of the compression ratio, a critical factor that dictates model performance for processing bytes data via chunks. In this paper, we propose Adaptive Targeted Dynamic Chunking (ATDC), a novel byte-compression control mechanism designed to enhance the effectiveness of dynamic chunking within hierarchical architectures. Our approach utilizes curriculum learning to progressively adjust the compression ratio during training, transitioning from low to high compression to stabilize the learning process. We provide an analysis establishing the relationship between the target compression ratio and Bytes-Per-Innermost-Chunk (BPIC), allowing for tracking of chunk-size evolution throughout the training phase. Evaluations conducted on the FineWeb-Edu 100B dataset demonstrate that hierarchical models equipped with ATDC achieve competitive Bits-Per-Byte (BPB) performance compared to conventional baselines operating at both byte and token levels. Furthermore, the proposed method exhibits more stable training dynamics and superior final performance across diverse downstream tasks compared to models using fixed compression ratios, while maintaining the inherent robustness and flexibility of byte-level processing.**

***Index Terms*—Tokenization-free, Hierachical Models, Dynamic Chunking**

I. Introduction

The development of Large Language Models (LLMs) has traditionally depended on subword tokenization techniques, including Byte-Pair Encoding (BPE) [1], [2], WordPiece [3], and hybrid Byte-Word approaches [4] applied during preprocessing. Although these methods have proven effective, they impose notable limitations. Tokenization demands intricate vocabulary construction, which frequently results in out-of-vocabulary (OOV) problems and reduced resilience to typographical errors or morphological diversity [5]. Moreover, tokenizers exhibit pronounced bias toward high-resource languages [6], exacerbating performance disparities in multilingual contexts. These intrinsic drawbacks, often termed the "overheads" of tokenization, have driven interest in tokenization-free architectures [7] that process raw bytes directly. While byte-level models hold considerable potential, they encounter a core efficiency hurdle: individual byte or meaningful chunks span numerous bytes, necessitating substantial compression to handle extended contexts and control computational overhead effectively. Hierarchical designs, such as H-Net [8], mitigate this through mechanisms like dynamic chunking [9] or patching [7] to form higher-level abstractions from byte sequences. Nevertheless, determining and regulating the compression ratio in these systems remains predominantly heuristic and suboptimal. Existing approaches typically employ static compression rates (for example, compression rate N is equal to 6 indicates that average of 6 bytes are expected to be compressed into a single chunk) that fail to adapt to evolving data complexity throughout training, often causing unstable optimization or suboptimal dynamic compression. In this work, we address these shortcomings by proposing Adaptive Targeted Dynamic Chunking (ATDC). Our approach is motivated by the insight that a model's compression capability should adapt in tandem with its growing comprehension of the data distribution. We conceptualize compression management as a form of curriculum learning, wherein the method gradually modulates the compression ratio and targeted chunk sizes. This enables a smooth progression from low-compression (fine-grained) processing to high-compression (coarse-grained) regimes as training advances.

The primary contributions of this work are as follows:

- A Novel Compression Control Method: We propose ATDC, which dynamically regulates byte compression in hierarchical architectures.
- Indicator Framework: We provide a formal analysis establishing the relationship between a target compression ratio and the Bytes-Per-Innermost-Chunk (BPIC).
- Empirical Validation: Through extensive training on 100B tokens of the FineWeb-Edu [10] dataset, we demonstrate that byte-level models using ATDC achieve bits-per-byte performance competitive with both byte-level H-Net [9] and token-level Llama3.2 [11]-based models.
- Textual Robustness: We demonstrate that our adaptive approach leads to increased robustness against HellaSwag textual perturbations across a variety of tasks compared to models with a fixed compression rate.

II. Related works

Tokenization has become a significant limitation in modern Large Language Models, particularly due to its sensitivity to spelling variations, typos, capitalization, and morphological

changes [12]. Byte-level modeling, which directly processes raw bytes, offers a promising alternative by eliminating many of these preprocessing assumptions.

MegaByte [13] introduced a multiscale decoder capable of modeling extremely long byte sequences (over 1 million bytes). However, its reliance on fixed-size patches and simple byte embedding concatenation limits flexibility.

MambaByte [14] advanced this direction by training a Mamba-based [15] model directly on raw bytes, leveraging its efficient fixed-size state. Still, compressing all contextual information into a single fixed hidden state remains a potential bottleneck.

More recently, the Byte Latent Transformer (BLT) [7] proposed a dynamic patching strategy that groups bytes according to next-byte prediction entropy. This approach adaptively assigns more computation to complex regions while using less for predictable sequences. However, BLT's entropy-based method can suffer from instability or "drift," especially on highly repetitive text such as in multiple-choice reasoning tasks.

Dynamic Chunking [9] utilizes an end-to-end routing mechanism and exponential moving averages to learn chunk boundaries differentiably. However, its reliance on a fixed compression ratio introduces sensitive, manually-tuned hyperparameters. While H-Net++ [16] attempts to improve this via a lightweight Transformer context-mixer, its evaluation on limited data (1.4B tokens) and smaller model scales prevents a comprehensive performance comparison.

Despite these advancements, Dynamic Chunking is limited by its use of a fixed compression rate for chunk size control. Such a mechanism results in highly empirical hyperparameters that often demand manual tuning. Our work is motivated by the need to move beyond these manual constraints, a goal we achieve through our proposed ATDC method.

III. Hierarchical Models

H-Net [9] is a tokenizer-free hierarchical language model that directly processes raw byte sequences ($\mathcal{V}vocab = 256$). The primary innovation is Dynamic Chunking (DC), a mechanism that identifies semantic boundaries in byte sequences to facilitate efficient hierarchical processing. The general workflow is: Input Bytes $\rightarrow$ Encoder $\rightarrow$ Routing Module $\rightarrow$ Chunking $\rightarrow$ Main Network $\rightarrow$ Dechunking $\rightarrow$ Decoder $\rightarrow$ Output.

A. Hierarchical Decomposition

Given an input sequence $\mathbf{x} = (x_1, x_2, \ldots, x_L)$ of length L, H-Net constructs a hierarchical representation through recursive compression. At any stage s, the process is defined by:

- **Boundary Prediction:** A routing module predicts discrete decisions $\mathbf{b}^{(s)} \in \{0,1\}^{L_s}$ and continuous probabilities $\mathbf{p}^{(s)} \in [0,1]^{L_s}$ from hidden states $\mathbf{h}^{(s)} \in \mathbb{R}^{L_s \times d_s}$.
- **Compression:** The sequence length for the subsequent stage is determined by the number of boundaries, $M_{s+1} = \|\mathbf{b}^{(s)}\|_1$.

For a K-stage hierarchy with downsampling factors $\{n_1, n_2, \ldots, n_K\}$, the total compression ratio is given by $\prod_{i=1}^{K} n_i$. For example, a two-stage model with $N_1 = N_2 = 3$ achieves approximately $9\times$ compression.

B. Boundary Detection via Semantic Discontinuity

The routing module identifies boundaries by quantifying the semantic shift between consecutive hidden states using a learned cosine similarity:

$$\sigma_t = \frac{\langle Q(\mathbf{h}_{t-1}), K(\mathbf{h}_t) \rangle}{\|Q(\mathbf{h}_{t-1})\| \cdot \|K(\mathbf{h}_t)\|} \tag{1}$$

where Q and K are learnable linear projections initialized as identity matrices. The boundary probability p_t and discrete decision b_t are then derived as:

$$p_t = \text{clip}\left(\frac{1-\sigma_t}{2}, 0, 1\right), \quad b_t = \text{argmax}(1-p_t, p_t) \tag{2}$$

High similarity ($\sigma_t \approx 1$) signals semantic continuity ($p_t \approx 0$), while low similarity ($\sigma_t \approx -1$) triggers a boundary ($p_t \approx 1$). This formulation learns a data-dependent metric space where cosine distance aligns with natural semantic transitions.

C. Component-wise Analysis

1) Encoder and Routing: The encoder $\mathcal{E}_{\text{enc}}$ contextualizes byte embeddings using L layers alternating between Mamba-2 State-Space Models (SSMs) [17], multi-head attention, and SwiGLU [18] feed-forward networks. The routing function R maps these states to probabilities $\mathbf{p} = R(\mathbf{h}_{\text{enc}}; \mathbf{h}_{\text{route}})$. To allow gradient flow through the discrete argmax, we use a Straight-Through Estimator (STE) [19], treating the discretization as an identity during backpropagation:

$$\frac{\partial L}{\partial \mathbf{h}} = \frac{\partial L}{\partial \mathbf{p}} \cdot \frac{\partial \mathbf{p}}{\partial \mathbf{h}} \tag{3}$$

2) Dechunking via EMA: The dechunking operator $\mathcal{D}$ reconstructs the original sequence length using an Exponential Moving Average (EMA) [20] recurrence:

$$\mathbf{y}_t = p_t \cdot \mathbf{c}_t + (1-p_t) \cdot \mathbf{y}_{t-1} \tag{4}$$

where $\mathbf{c}_t$ represents the chunk value at boundaries. This operation is causal, differentiable, and can be efficiently computed via parallel scan kernels by reformulating the recurrence as a linear SSM.

3) Gated Residual Connections: To ensure stable hierarchical training, we utilize gated residual connections:

$$\mathbf{h}_{\text{out}} = \mathbf{h}_{\text{dechunk}} \cdot \text{STE}(\mathbf{p}) + W_{\text{res}}(\mathbf{h}_{\text{enc}}) \tag{5}$$

where W_{res} is a zero-initialized projection. This creates an identity-path initialization, allowing the model to gradually learn the hierarchical features without destabilizing initial training.

D. Main Network: Recursive Hierarchical Processing

The main network $\mathcal{M}_{\text{main}}$ processes the compressed latent representations $\mathbf{h}_{\text{chunk}}$ generated by the Dynamic Chunking (DC) mechanism. Its architecture can be configured in two primary modes depending on the hierarchical depth K:

1) **Recursive H-Net Stage:** For $K > 1$, the main network recursively applies another H-Net stage. This creates a multi-scale hierarchy where each subsequent layer operates on a progressively coarser granularity, effectively building a tree-like computation graph.
2) **Isotropic Model:** In the final stage, the compressed sequence is processed by a standard sequence model (e.g., a deep Transformer or a Mamba-2 backbone). Since the sequence length is reduced from L to $M \approx L/r$, the attention mechanism's quadratic complexity $\mathcal{O}(M^2)$ or the SSM's linear complexity $\mathcal{O}(M)$ is significantly mitigated.

a) Mathematical Formulation: For a K-stage hierarchy, let $\mathcal{M}_{\text{main}}^{(k)}$ denote the processing block at stage k. The state transition is defined as:

$$\mathbf{h}^{(k)} = \begin{cases} \mathcal{M}_{\text{base}}(\mathbf{h}^{(k-1)}) & k = K \\ \mathcal{M}_{\text{main}}^{(k)}(\mathbf{h}^{(k-1)}) & k < K \end{cases} \tag{6}$$

This recursive structure allows the model to capture long-range dependencies at the coarsest levels while maintaining the ability to reconstruct fine-grained local details during the dechunking and decoding phases. By bottlenecking the information flow through a learned semantic boundary, the Main Network focuses its parameters on the most salient features of the byte stream.

IV. Adaptive Targeted Dynamic Chunking

We propose the *Adaptive Targeted Dynamic Chunking* (ATDC) mechanism to dynamically regulate dynamic chunking during training. We retain the original routing formulation from [9] but modify the compression control mechanism. ATDC stabilizes the learning process by employing a scheduled linear growth of the compression factor N, supplemented by a performance-driven adaptation trigger and a localized balancing loss to maintain structural integrity.

A. Adaptive Routing Schedule

The compression factor $N(t)$ dictates the target reduction in sequence length at training step t. To facilitate a stable transition from a low-compression (simplified) architecture to a high-capacity state, we implement a multi-phase schedule. During the initial warm-up period ($t < T_w$), $N(t)$ is held constant at N_{init} to allow the router to converge on basic patterns. Subsequently, $N(t)$ follows a linear interpolation toward a target value N_{fnl} based on the training progress:

$$N_{sched}(t) = \begin{cases} N_{init} & t < T_w \\ N_{init} + \frac{t-T_w}{T-T_w}(N_{fnl} - N_{init}) & t \geq T_w \end{cases} \tag{7}$$

Adaptive Targeted Dynamic Chunking

Input: Steps T, Warmup T_w, Targets $\mathbf{N}_{init}, \mathbf{N}_{fnl} \in \mathbb{R}^S$, Rate γ, Threshold τ, Window W
$t \leftarrow 0, \mathcal{H} \leftarrow \emptyset$ ▷ Initialize for metric-based adaptation
while $t < T$ **do** ▷ Adaptive Scheduling
  if $t < T_w$ **then**
    $\mathbf{N}_{sched} \leftarrow \mathbf{N}_{init}$
  else
    $\rho \leftarrow \min\left(\frac{t-T_w}{T-T_w}, 1\right)$
    $\mathbf{N}_{sched} \leftarrow \mathbf{N}_{init} + \rho \cdot (\mathbf{N}_{fnl} - \mathbf{N}_{init})$
  end if
  if $|\mathcal{H}| \geq W$ **then** ▷ Wait for window to fill
    $\bar{\ell} \leftarrow \text{mean}(\mathcal{H}[-W:])$ ▷ Average loss
    $\mathbf{N}_{curr} \leftarrow (\bar{\ell} < \tau)?(\mathbf{N}_{sched} \cdot \gamma) : \mathbf{N}_{sched}$
  else
    $\mathbf{N}_{curr} \leftarrow \mathbf{N}_{sched}$
  end if
  $(\mathbf{y}, \{\mathcal{R}_s\}_{s=1}^{S}) \leftarrow \mathcal{M}(\mathbf{x})$ ▷ Forward Pass & Balancing
  $\mathcal{L}_{bal} \leftarrow 0$
  for $s \leftarrow 1$ **to** S **do**
    $N_s \leftarrow \mathbf{N}_{curr}[s]$
    $\mathcal{L}_{bal} \leftarrow \mathcal{L}_{bal} + \text{BalancingLoss}(\mathcal{R}_s, N_s)$
  end for
  $\mathcal{L}_{total} \leftarrow \text{CrossEntropy}(\mathbf{y}, \mathbf{x}) + \alpha\mathcal{L}_{bal}$
  Update $\mathcal{M}$ via $\nabla_\theta \mathcal{L}_{total}$; $\mathcal{H}$.append($\mathcal{L}_{total}$)
  $t \leftarrow t + 1$
end while

---

function BalancingLoss(out$_s$, N)
  $Y \leftarrow \text{mean}(\text{out}_s.\text{mask})$
  $Y' \leftarrow \text{mean}(\text{out}_s.\text{prob})$
  $\mathcal{L}_{bal} \leftarrow \frac{N}{N-1} \times \Big[(N-1)YY' + (1-Y)(1-Y')\Big]$
  return $\mathcal{L}_{bal}$
end function

To further optimize the training trajectory, we introduce a metric-based adaptation. We maintain a history of the language modeling loss within a sliding window W. If the windowed average loss $\bar{\ell}$ falls below a predefined threshold τ, indicating the model has reached a sufficient level of proficiency, we apply an adaptation rate γ to accelerate capacity acquisition:

$$N(t) = \begin{cases} N_{sched}(t) \cdot \gamma & \text{if } \bar{\ell} < \tau \\ N_{sched}(t) & \text{otherwise} \end{cases} \tag{8}$$

B. Balancing Loss

To regularize the routing mechanism and prevent degenerate solutions, and we utilize a balancing loss [9] $\mathcal{L}_{bal}$. This term aligns the router's soft probabilistic outputs Y' with the discrete routing decisions Y actually executed by the model. Given the current compression control N, the loss for a sequence of length L is formulated as:

$$\mathcal{L}_{bal} = \left(\frac{N}{N-1}\right)\Big[(N-1)YY' + (1-Y)(1-Y')\Big] \quad (9)$$

where Y represents the fraction of vectors actually selected by the boundary mask, and Y' denotes the average boundary probability:

$$Y = \frac{1}{L}\sum_{t=1}^{L} b_t, \quad Y' = \frac{1}{L}\sum_{t=1}^{L} p_t \quad (10)$$

In this framework, $b_t \in \{0,1\}$ is the binary boundary mask and $p_t \in [0,1]$ is the router's predicted probability for the t-th vector.

- **The Boundary Penalty** $(N-1)YY'$**:** This term penalizes the density of boundary assignments. As the target N increases, the model is increasingly incentivized to minimize boundary selections, thereby enforcing higher compression.
- **The Consistency Term** $(1-Y)(1-Y')$**:** This ensures that for vectors not selected as boundaries, the router remains confident in its non-selection (predicting low p_t), which stabilizes the gradients for "skipped" tokens.
- **Normalization** $\frac{N}{N-1}$**:** This coefficient ensures that the balancing loss remains numerically stable and comparable in magnitude across different phases of the N schedule, preventing the regularization from dominating the global loss as N grows.

C. Compression Control by ATDC and Indicator as BPIC

Our method ATDC regulates the chunk size of router, via the ratio loss, it acts as a soft constraint on compression, allowing the model to deviate when semantically necessary while maintaining overall efficiency.

1) Compression Control and Load Balancing: Recall N is the target compression ratio and $\rho = 1/N$ be the target boundary fraction. The ATDC mechanism optimizes a ratio loss $\mathcal{L}_{bal}$ defined in (9). The total objective loss function is:

$$\mathcal{L}_{\text{total}} = \alpha\mathcal{L}_{bal} + \mathcal{L}_{\text{entropy}} \quad (11)$$

where α is a scaling hyperparameter, $\mathcal{L}_{\text{entropy}}$ is a cross-entropy loss of the next bytes prediction in the sequence. This objective ensures that the model maintains the target throughput while using the entropy term to prevent over-confident or uniform boundary predictions.

2) Boundary Logic and Segmentation: The router determines boundaries by measuring the semantic shift between consecutive states $\mathbf{h}_t$ and $\mathbf{h}_{t+1}$ in a learned metric space:

$$p_t = \frac{1}{2}\left(1 - \frac{Q(\mathbf{h}_t)\cdot K(\mathbf{h}_{t+1})}{\|Q(\mathbf{h}_t)\|\|K(\mathbf{h}_{t+1})\|}\right) \quad (12)$$

A discrete boundary mask $b \in \{0,1\}$ is generated by $b_t = \mathbb{I}(p_t > 0.5)$. Sequential positions between active indices in b are grouped into chunks. We monitor the empirical compression performance by calculating **Bytes Per Innermost Chunk (BPIC)**:

$$\text{BPIC} = \frac{1}{M}\sum_{i=1}^{M} \text{size}(C_i) \quad (13)$$

where M is the total number of chunks and size(C_i) is the length of the i-th segment.

V. Implementation

A. Model

We compare against the byte-level H-Net model [9]. We additionally compare against token-level architecture Llama3.2 [11] model. H-Net supports K-stage, for computational cost reason, we only use one stage (K=1) for all H-Net models. The isotropic model or main network of H-Net is Transformer architecture (vanilla Llama architecture).

B. Cross-Model Evaluation Metrics

We use 100B tokens subset sampled from the English Fineweb-edu [10] dataset. To ensure a fair comparison between the byte-level model and the token-level baseline, we normalize all performance results to Bits-Per-Byte (BPB). While the token-level model operates on a sequence length of $L_{token} = 1732$, the byte-level model processes $L_{byte} = 8192$ UTF-8 encoded bytes, representing an equivalent volume of raw information.

The token-level perplexity (PPL) is converted to BPB using the equation: $BPB = (\log_2(PPL) \cdot L_{token})/L_{byte}$

This normalization accounts for the tokenizer's compression ratio (approximately 4.73 bytes per token) and allows for a vocabulary-agnostic comparison of the models' generative efficiency.

C. Hyperparameters

The performance of the ATDC mechanism relies on the selection of the adaptation threshold τ and the acceleration rate γ. These parameters govern the trade-off between predictive accuracy and computational efficiency.

1) Adaptive targeted N: We utilize the same settings as [9] for all H-Net baseline models, with $N = 6.0$. For the ADTC, we set $N_{init} = 5.0$ and $N_{fnl} = 6.5$. To maintain training stability, we set the warm-up period to $T_w = 60\%T$. Following this, adaptive scheduling is deployed to automatically scale up N_{sched} using metric-based adaptation (7).

2) Selection of Adaptation Threshold (τ): The threshold τ serves as a "proficiency trigger." It defines the maximum allowable language modeling loss at which the model is deemed stable enough to handle increased compression.

- **Empirical Estimation:** To determine τ, we first conduct a baseline training run with a static N_{init}. τ is set to the loss value achieved at the end of the first epoch or the point where the loss curve begins to plateau. Our empirical τ value is 0.05.
- **Sensitivity:** Setting τ too high may result in premature compression and routing instability. Conversely, a τ that is too low may never trigger the adaptation.

3) Selection of Adaptation Rate (γ): The rate γ determines the magnitude of the opportunistic boost in the compression factor. In our experiments, we fixed γ is equal 1.05 (5% boost in compression). Because γ modifies $N(t)$, which scales $\mathcal{L}_{bal}$, we use a windowed average $\bar{\ell}$ over W (100) steps to ensure the trigger loss changing is based on sustained trends rather than batch-level.

4) Scaling hyperparameter of loss function (α): Purpose: Balances the two objectives $\mathcal{L}_{bal}$ and $\mathcal{L}_{\text{entropy}}$ in (11):

- α too small: Model ignores load balancing $\rightarrow$ poor chunking, inefficient hierarchical processing.
- α too large: Model focuses too much on balanced chunks $\rightarrow$ worse language modeling performance.
- α = 0.03: Empirically optimal value that maintains good language modeling while encouraging efficient dynamic chunking.

5) Others: We utilized learning rate modulation [9] scaled according to model size. Specifically, the peak learning rates were set as follows: 7.5×10^{-4} for the 350M H-Net model, 6.25×10^{-4} for the 680M H-Net and 770M Llama 3.2 models, and 5.0×10^{-4} for the 1.3B H-Net and 1.5B Llama 3.2 models. Training was conducted using the AdamW optimizer with Decoupled Weight Decay Regularization [21] and a Warmup-Stable-Decay (WSD) [22] scheduler. This schedule included a 10% initial warmup phase and a 20% inverse-square-root decay phase [23]. We employed a batch size of 256 across 8 NVIDIA H200 GPUs, resulting in approximately 2M bytes per gradient update.

D. Baselines and Metrics

To evaluate the ATDC method, we compare it against the following configurations:

- **Static Routing:** The compression ratio N remains fixed at N_{init} throughout the entire training process.
- **Adaptive Scheduling:** The value of N follows the ATDC, metric-based adjustments used in our approach.
- **Token-level Reference:** We train and evaluate a token-level Llama 3.2 model to provide a performance benchmark against standard tokenization methods. Because token-level models do not utilize the same byte-level compression metrics, we omit the Llama 3.2 validation BPB from our figures to maintain consistency across the comparative analysis of BPIC and N.

E. Results

Table I presents a comprehensive evaluation of our proposed method compared to the token-level Llama3.2 and a byte-level H-Net baseline models. The performance is measured using validation BPB (as Fig. 1) for language modeling quality and zero-shot accuracy across seven standard benchmarks. A primary observation is that byte-level models consistently outperform their token-based counterparts (Llama3.2) across all parameter scales, despite having slightly fewer parameters in some configurations. Notably, the H-Net 680M (ours) achieves an average accuracy of 50.7%, significantly surpassing the Llama3.2 770M (42.1%) and even the larger Llama3.2 1.5B (48.5%). This suggests that direct byte-level modeling captures linguistic nuances and structural information that tokenization-based preprocessing may obscure. Our implementation consistently achieves the lowest (best) BPB scores in every category, reaching a minimum of 0.760 at the 1.3B parameter scale.

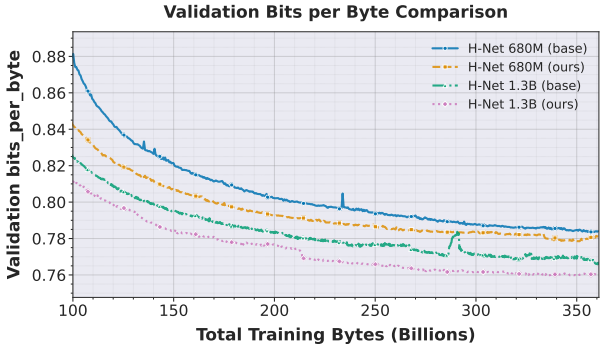


Fig. 1: Validation BPB.

In terms of zero-shot reasoning: the ADTC outperforms the base H-Net model in nearly every metric, particularly in high-stakes reasoning tasks like ARC-Challenge (ARC-C) and PIQA.

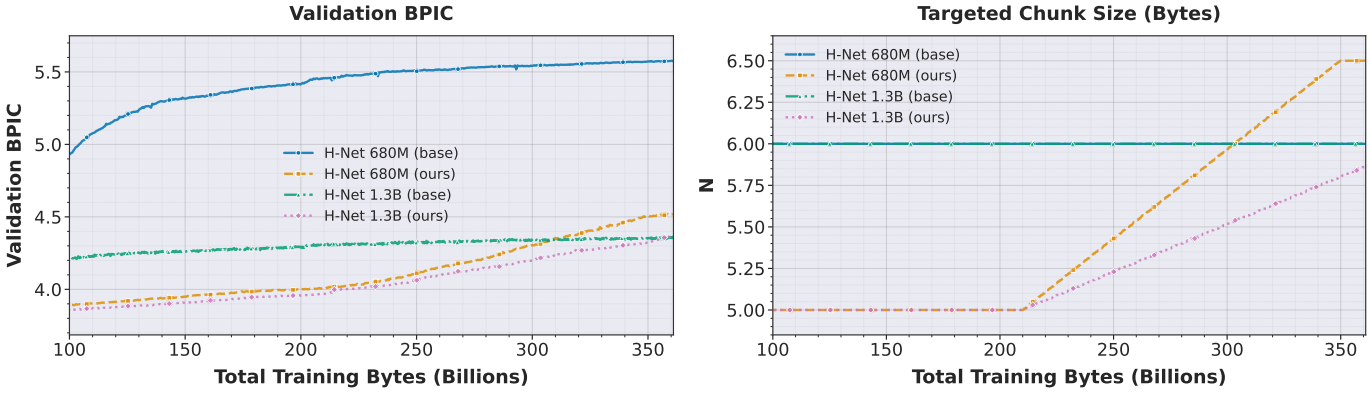
Robustness to Textual Perturbations. To evaluate the structural resilience of our proposed method, we conducted a stress test on the HellaSwag benchmark using five types of textual perturbations: Antspeak (added spacing), Drop (character deletion), RandomCase, Repeat (character duplication), and Uppercase. The results, summarized in Table II, highlight a significant gap in robustness between token-based and byte-level architectures.

Resilience Against Out-of-Distribution Noise. The token-based Llama 3.2 models exhibit a sharp decline in performance when faced with even minor character-level noise. For instance, at the 1.5B scale, Llama 3.2's accuracy drops to an average of 27.7% under perturbation. In contrast, our H-Net 1.3B (ours) maintains an average accuracy of 31.6%. This confirms that subword tokenizers are highly sensitive to spelling variations and formatting changes, as these perturbations often result in "unknown" tokens or fragmented subword sequences that disrupt the model's semantic understanding.

Comparison of Byte-Level Strategies. While both byte-level models (base and ours) significantly outperform the token-based baseline, our method consistently achieves the highest accuracy across nearly all noise categories.

Case Sensitivity. Our model shows a particular advantage in the UPPERCASE and RandomCase categories. Specifically, at the 1.3B scale, our method achieves 39.7% on uppercase text, outperforming the base byte model and vastly exceeding Llama 3.2 (29.6%).

Structural Consistency. In the Antspeak and Repeat, which simulate common typos and stylized web text, our method demonstrates superior stability. This suggests that the adaptive thresholding in our dynamic chunking mechanism is better at identifying core morphological units.



(a) Comparison on BPIC.

(b) Rate compression control N.

Fig. 2: Validation BPIC curves to track the N values changing during training phase.

TABLE I: Validation Bits-per-byte (BPB) and Zero-shot performance comparison across multiple benchmarks

MODEL	INPUT	BPB↓	BPIC	ARC-C↑	ARC-E↑	HELLA↑	LAMB↑	OPEN↑	PIQA↑	WINO↑	AVG↑
Llama3.2 770M	Token	0.889	NA	28.1	53.7	38.3	26.7	32.4	64.5	51.0	42.1
H-Net 680M (base)	Byte	0.783	5.58	33.4	60.1	52.3	40.9	**39.2**	70.9	55.7	50.4
H-Net 680M (ours)	Byte	**0.778**	4.52	**33.9**	**61.2**	**52.4**	**42.1**	37.2	**71.7**	**56.1**	**50.7**
Llama3.2 1.5B	Token	0.800	NA	33.1	60.3	49.0	37.3	34.8	70.2	54.7	48.5
H-Net 1.3B (base)	Byte	0.766	4.35	34.4	58.7	**55.0**	43.4	**38.4**	70.7	**56.8**	51.1
H-Net 1.3B (ours)	Byte	**0.760**	4.37	**36.9**	**60.5**	54.9	**44.4**	37.4	**71.5**	56.2	**51.7**

TABLE II: The Evaluation on HellaSwag with textual perturbations

MODEL	INPUT	ANTSPEAK	DROP	RANDOMCASE	REPEAT	UPPERCASE	AVERAGE-ACC↑
Llama3.2 770M	Token	28.0	26.0	25.5	26.0	27.1	26.5
H-Net 680M (base)	Byte	30.0	27.9	27.5	**29.2**	37.7	30.5
H-Net 680M (ours)	Byte	**30.6**	**28.0**	**27.8**	29.0	**38.0**	**30.7**
Llama3.2 1.5B	Token	29.7	26.6	25.7	26.9	29.6	27.7
H-Net 1.3B (base)	Byte	31.2	28.2	26.9	29.0	38.8	30.8
H-Net 1.3B (ours)	Byte	**31.3**	**28.7**	**28.3**	**29.9**	**39.7**	**31.6**

Visualization of Chunking Boundaries.

We utilize our largest model variant (1.3B parameters) to analyze chunk boundaries and compare our proposed method against the conventional fixed-rate approach. As illustrated in Fig. 3, the H-Net model determines these boundaries by calculating the boundary probability via (12), which subsequently dictates the masking and chunk compression strategy. The visualization reveals that the conventional method (second row) produces less semantically meaningful segmentations compared to our method (third row). For instance, the conventional approach splits the word "checking" across two separate chunks, C_0 ("ch") and C_1 ("ecking"), while simultaneously grouping the word "the" into C_1. In contrast, our method maintains the integrity of the word "checking" within a single chunk, C_0. A similar pattern is observed with the word "model": the conventional method fragments it into three distinct chunks (C_7, C_8, C_9), whereas our approach successfully preserves it as a single unit in C_9. These observations suggest that our adaptive mechanism better captures linguistic structures than fixed-rate alternatives.

Indicator BPIC and adaptive N analysis.

Fig. 2 illustrates the validation BPIC curves and the corresponding shifts in the adaptive N value after 200B bytes. These indicators allow us to verify that the adjustments to N are accurately reflected in the average number of bytes per chunk. While BPIC is not a direct performance metric, it serves as a vital diagnostic tool for assessing the variance in chunk sizes during the dynamic chunking process. This analysis is crucial for understanding the model's linguistic abstractions and provides a quantitative measure of how the targeted compression ratio N influences internal semantic segmentation across two distinct training phases:

- **Phase 1 (Initial Feature Acquisition):** During early training, we set N to a low range. In this stage, the router is encouraged to be highly sensitive, treating even moderate feature differences as segment boundaries. This results in smaller chunks and a fine-grained representation, minimizing the risk of information loss while the model learns basic byte-level patterns and local context.
- **Phase 2 (Compression Transition):** As training pro-

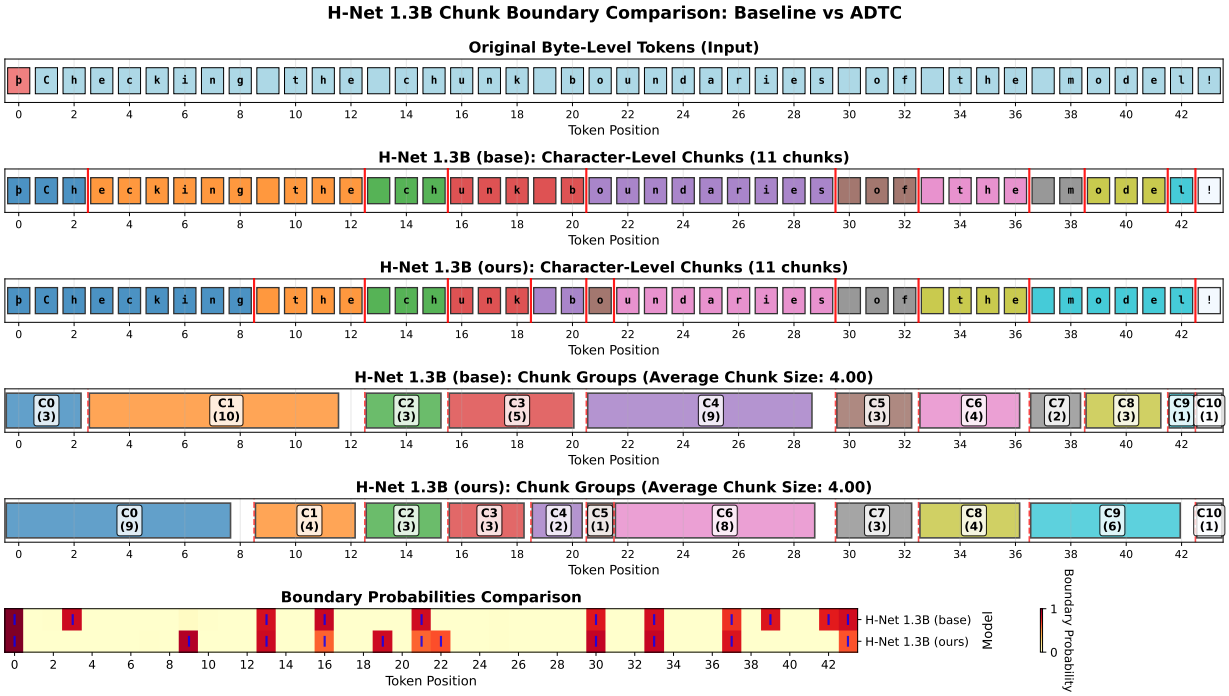


Fig. 3: Chunk Boundary Visualization of H-Net 1.3B.

gresses, N is gradually increased. This forces the router to become more selective, identifying only significant differences as boundaries. The model begins to learn how to compress information into larger, more efficient blocks, shifting its focus from local byte sequences to broader contextual structures.

VI. Ablation Study

In this section, we perform an ablation analysis to evaluate the impact of the compression ratio on hierarchical byte-level models. To manage computational costs, these experiments were conducted using the H-Net 350M variant, trained on a 40% random subset of the 100B FineWeb-Edu dataset. We investigate whether maintaining a fixed compression ratio N ranging from low ($N = 7.0$) and medium ($N = 8.0$) to high ($N = 9.0$).

The results presented in Table III and Table IV confirm that our method of adapting N values from 8.0 up to 9.5 generally outperforms conventional fixed-rate methods across standard benchmarks and robustness tests. While the validation BPB for all models is similar (approximately 0.828) due to the reduced data volume and model size, the downstream tasks provide a clearer demonstration of our method's effectiveness.

Impact of Fixed vs. Adaptive Compression.

Table III illustrates the trade-offs inherent in fixed-rate compression. While specific fixed rates show localized strengths, such as base.upper performing well on ARC-Easy (57.0) and LAMBADA (39.4), none of the fixed configurations maintain high performance across all tasks.

In contrast, our method achieves the highest Average Accuracy (47.3%). Our method demonstrates a superior ability to generalize, particularly in the ARC-Challenge (+1.0% over the best baseline) and OpenBookQA tasks. This suggests that the adaptive mechanism effectively allocates information density where it is most needed, rather than forcing a uniform sequence length that may discard critical details in complex contexts.

Textual perturbations

We further evaluated the impact of compression on model robustness under textual perturbations, as shown in Table IV. The performance across the different fixed-rate baselines is relatively stagnant, with average accuracies hovering between 28.8% and 28.9%. Our adaptive method achieves the highest overall robustness score (29.0%), with a notable peak in the UPPERCASE category (34.5%). While the margins in the robustness ablation are tighter than in the standard benchmarks, the consistency of our method across Drop and Uppercase perturbations indicates that learning segmentation strategies jointly with the network provides a more stable representation than heuristic-based fixed chunking.

VII. Conclusion

In this work, we have addressed the inherent limitations of subword tokenization by proposing ATDC (Adaptive Targeted Dynamic Chunking), a novel byte-level hierarchical framework. Through extensive evaluation across multiple scales from 350M to 1.3B parameters. Our results demonstrate that ATDC consistently outperforms both traditional token-based architectures like Llama 3.2 and state-of-the-art byte-level baselines using fixed compression strategies.

Our empirical analysis yields three primary contributions. First, we demonstrate that byte-level modeling provides su-

TABLE III: **Ablation Study.** H-Net 350M trained on 40% random Fineweb-Edu variants across different benchmarks

MODEL	BPIC	ARC-C	ARC-E	HELLA	LAMB	OPEN	PIQA	WINO	AVERAGE-ACC
H-Net 350M (base.lower)	5.11	30.1	56.4	**45.7**	38.9	34.0	68.1	**54.1**	46.8
H-Net 350M (base.mid)	5.42	29.4	52.7	45.0	36.6	38.0	**68.7**	53.2	46.2
H-Net 350M (base.upper)	5.68	31.0	**57.0**	45.6	**39.4**	36.4	68.2	51.5	47.0
H-Net 350M (ours)	5.51	**32.0**	56.2	45.6	38.5	**37.4**	68.0	53.1	**47.3**

TABLE IV: **Ablation Study.** Textual Perturbation evaluation for H-Net 350M variants

MODEL	ANTSPEAK	DROP	RANDOMCASE	REPEAT	UPPERCASE	AVG-ACC
H-Net 350M (base.lower)	**29.2**	26.9	**26.3**	27.6	34.2	28.9
H-Net 350M (base.mid)	**29.2**	27.2	**26.3**	**27.8**	34.0	28.9
H-Net 350M (base.upper)	29.1	27.1	26.2	27.6	33.8	28.8
H-Net 350M (ours)	29.1	**27.3**	**26.3**	27.5	**34.5**	**29.0**

perior language modeling quality (BPB) and zero-shot reasoning capabilities compared to token-based models. Notably, our 680M variant outperforms the Llama 3.2 1.5B model, suggesting that direct byte-level processing captures linguistic nuances that subword tokenization often obscures. Second, our robustness evaluations prove that ATDC is significantly more resilient to out-of-distribution noise and textual perturbations. By avoiding the "tokenization preprocessing", our model maintains high performance in the presence of typos, case variations, and formatting shifts where traditional models operate worse (Table II).

Finally, our ablation studies and boundary visualizations confirm that adaptive targeted chunking is essential for effective hierarchical modeling. Unlike fixed-rate compression, which often fragments semantic units, our adaptive mechanism learns to align chunk boundaries with meaningful morphological structures. This allow the model to dynamically allocate information density, leading to more stable representations and improved generalization across complex reasoning tasks. We believe that ATDC provides a robust and efficient foundation for the next generation of token-free large language models.

ACKNOWLEDGMENT

Large language models (LLMs) were used solely to assist with proofreading and improving the clarity of the text. All research ideas, theoretical development, experiments, and implementation were carried out entirely by the authors.

REFERENCES

[1] T. Kudo and J. Richardson, "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," in *Proceedings of the 2018 Conference on EMNLP*, 2018, pp. 66–71.

[2] C. W. Schmidt, V. Reddy, C. Tanner, and Y. Pinter, "Boundless byte pair encoding: Breaking the pre-tokenization barrier," in *Second Conference on Language Modeling*, 2025.

[3] X. Song, A. Salcianu, Y. Song, D. Dopson, and D. Zhou, "Fast wordpiece tokenization," in *Proceedings of the 2021 conference on empirical methods in natural language processing*, 2021, pp. 2089–2103.

[4] P. Neitemeier, B. Deiseroth, C. Eichenberg, and L. Balles, "Hierarchical autoregressive transformers: Combining byte- and word-level processing for robust, adaptable language models," in *The 13th ICLR*, 2025.

[5] O. Ahia, S. Kumar, H. Gonen, J. Kasai, D. R. Mortensen, N. A. Smith, and Y. Tsvetkov, "Do all languages cost the same? tokenization in the era of commercial language models," in *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

[6] A. Petrov, E. La Malfa, P. Torr, and A. Bibi, "Language model tokenizers introduce unfairness between languages," *Advances in neural information processing systems*, vol. 36, pp. 36 963–36 990, 2023.

[7] A. Pagnoni, R. Pasunuru, P. Rodriguez, J. Nguyen, B. Muller, M. Li, C. Zhou, L. Yu, J. E. Weston, L. Zettlemoyer *et al.*, "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)*, 2025, pp. 9238–9258.

[8] P. Nawrot, S. Tworkowski, M. Tyrolski, Ł. Kaiser, Y. Wu, C. Szegedy, and H. Michalewski, "Hierarchical transformers are more efficient language models," in *Findings of the Association for Computational Linguistics: NAACL 2022*, 2022, pp. 1559–1571.

[9] S. Hwang, B. Wang, and A. Gu, "Dynamic chunking for end-to-end hierarchical sequence modeling," in *The 14th ICLR*, 2026.

[10] G. Penedo, H. Kydlíček, A. Lozhkov, M. Mitchell, C. A. Raffel, L. Von Werra, T. Wolf *et al.*, "The fineweb datasets: Decanting the web for the finest text data at scale," *Advances in Neural Information Processing Systems*, vol. 37, pp. 30 811–30 849, 2024.

[11] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan *et al.*, "The llama 3 herd of models," *arXiv preprint arXiv:2407.21783*, 2024.

[12] L. Xue, A. Barua, N. Constant, R. Al-Rfou, S. Narang, M. Kale, A. Roberts, and C. Raffel, "Byt5: Towards a token-free future with pre-trained byte-to-byte models," *Transactions of the Association for Computational Linguistics*, vol. 10, pp. 291–306, 2022.

[13] J. Wang, T. Gangavarapu, J. N. Yan, and A. M. Rush, "Mambabyte: Token-free selective state space model," in *First Conference on Language Modeling*, 2024.

[14] L. YU, D. Simig, C. Flaherty, A. Aghajanyan, L. Zettlemoyer, and M. Lewis, "MEGABYTE: Predicting million-byte sequences with multiscale transformers," in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[15] A. Gu, K. Goel, and C. Re, "Efficiently modeling long sequences with structured state spaces," in *International Conference on Learning Representations*, 2022.

[16] M. Zakershahrak and S. Ghodratnama, "H-net++: Hierarchical dynamic chunking for tokenizer-free language modelling in morphologically-rich languages," *arXiv preprint arXiv:2508.05628*, 2025.

[17] T. Dao and A. Gu, "Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality," in *Proceedings of the 41st ICML*, vol. 235. PMLR, 21–27 Jul 2024, pp. 10 041–10 071.

[18] N. Shazeer, "Glu variants improve transformer," *arXiv preprint arXiv:2002.05202*, 2020.

[19] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[20] D. Morales-Brotons, T. Vogels, and H. Hendrikx, "Exponential moving average of weights in deep learning: Dynamics and benefits," *Transactions on Machine Learning Research*, 2024.

[21] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[22] S. Hu, Y. Tu, X. Han, C. He, G. Cui, X. Long, Z. Zheng, Y. Fang, Y. Huang, W. Zhao *et al.*, "Minicpm: Unveiling the potential of small language models with scalable training strategies," *arXiv preprint arXiv:2404.06395*, 2024.

[23] A. Ibrahim, B. Thérien, K. Gupta, M. L. Richter, Q. G. Anthony, E. Belilovsky, T. Lesort, and I. Rish, "Simple and scalable strategies to continually pre-train large language models," *Transactions on Machine Learning Research*, 2024.