DYNAMIC LARGE CONCEPT MODELS: LATENT REASONING IN AN ADAPTIVE SEMANTIC SPACE

Xingwei Qu[*1,2] **Shaowen Wang**[*1,5] **Zihao Huang**[*1]
Kai Hua[1] Fan Yin[1] Rui-Jie Zhu[1] Jundong Zhou[1] Qiyang Min[†1]
Zihao Wang[5] Yizhi LI[2] Tianyu Zhang[3] He Xing[5] Zheng Zhang[1] Yuxuan Song[1]
Tianyu Zheng[5] Zhiyuan Zeng[1]
Chenghua Lin[†2] Ge Zhang[†1,5] Wenhao Huang[†1]

[1]ByteDance Seed [2]University of Manchester [3]Mila - Quebec AI Institute
[4]Tsinghua University [5]M-A-P

ABSTRACT

Large language models apply uniform computation to all tokens, despite language exhibiting highly non-uniform information density. We propose Dynamic Large Concept Models (DLCM), which learn variable-length semantic concepts from latent representations and perform reasoning in a compressed concept space. By reallocating computation from redundant token processing to concept-level reasoning, DLCM enables adaptive compute allocation aligned with semantic structure. We further introduce a compression-aware scaling law and a decoupled μP parametrization for heterogeneous token- and concept-level modules. With approximately 34% lower inference FLOPs, DLCM achieves a 2.69% average improvement across 12 zero-shot benchmarks, with gains concentrated on reasoning-intensive tasks.

1 INTRODUCTION

Large Language Models (LLMs) have achieved remarkable success across natural language understanding, reasoning, and generation. Despite differences in scale and training data, nearly all state-of-the-art models share a common architectural assumption: language is processed uniformly at the token level, with identical depth and computation applied to every position in the sequence.

This assumption conflicts with the structure of natural language. Information density is highly non-uniform: long spans of locally predictable tokens are interspersed with sparse but semantically critical transitions where new concepts are introduced and reasoning difficulty concentrates. Yet standard LLMs expend full computation on both regimes alike, leading to substantial redundancy and systematic misallocation of model capacity.

More fundamentally, this inefficiency reflects a limitation of token-level modeling. Reasoning is inherently hierarchical: humans reason over abstract units such as ideas or concepts before committing to surface realizations. Token-level autoregressive models, however, lack any explicit abstraction mechanism and are forced to repeatedly infer high-level structure implicitly through next-token prediction.

Prior work has explored relaxing token-level computation, but with important limitations. Latent reasoning approaches operate in continuous hidden spaces without explicit token generation, while sentence-level concept models rely on fixed, human-defined segmentation. Neither allows models to learn where semantic computation should be concentrated.

We argue that effective reasoning requires a learned intermediate granularity: neither raw tokens nor predefined sentences, but variable-length semantic concepts discovered directly from representation space. Based on this insight, we propose **Dynamic Large Concept Models (DLCM)**, a hierarchical next-token prediction framework that dynamically segments token sequences into concepts, performs

[*]Core Contributors. Work done at ByteDance Seed.
[†]Corresponding authors.

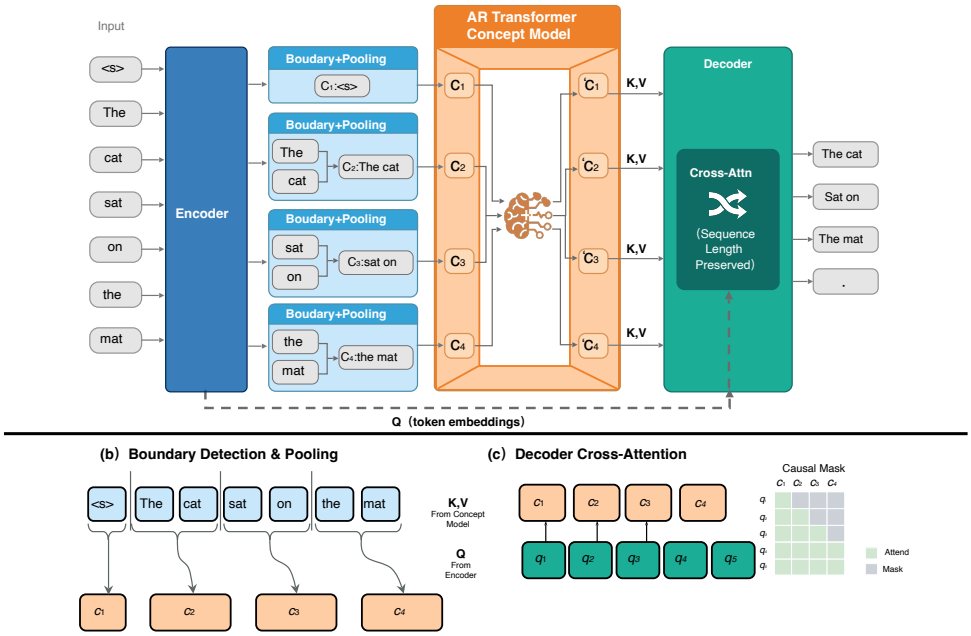

Figure 1: Overview of DLCM.

deep reasoning in a compressed concept space, and reconstructs token-level predictions via causal cross-attention.

This design decouples *what to reason about* from *how to reason*. By learning semantic boundaries and relocating computation from redundant token processing to concept-level reasoning, DLCM enables adaptive compute allocation aligned with information density.

Recent work such as H-NET Hwang et al. (2025) demonstrates the promise of learned boundary detection, but operates at the byte level and has not been validated in modern next-token prediction pipelines. In contrast, DLCM introduces *concept-level latent reasoning* directly within standard autoregressive language modeling.

Throughout this paper, we use the term *concept* to denote variable-length latent segments discovered from representation space, rather than linguistically predefined units. DLCM implements this idea through a four-stage pipeline (Figure 1):

1. **Encoding:** A lightweight encoder extracts fine-grained token representations.
2. **Dynamic Segmentation:** A boundary detector identifies semantic breakpoints via local representational dissimilarity.
3. **Concept-Level Reasoning:** Tokens within each segment are pooled into concept representations, which are processed by a high-capacity transformer.
4. **Token-Level Decoding:** Token predictions are reconstructed via causal cross-attention to the reasoned concepts.

By explicitly separating semantic abstraction from surface realization, DLCM allocates computation based on semantic structure rather than token count, yielding improved modeling of high-information, low-predictability regions.

Our main contributions are:

- **Concept-level latent reasoning with learned boundaries.** We introduce DLCM, a hierarchical next-token prediction architecture that learns variable-length semantic concepts and performs deep reasoning in a compressed concept space.

- **Compression-aware scaling laws.** We derive a scaling law $L(N, D, R, P)$ that captures the interaction between model size, data, compression ratio, and concept-level capacity under fixed FLOPs.
- **Decoupled μP for heterogeneous architectures.** We adapt Maximal Update Parametrization to heterogeneous token- and concept-level modules, enabling stable training and zero-shot hyperparameter transfer across widths and compression regimes.
- **Improved reasoning efficiency with lower compute.** At $R = 4$, DLCM reallocates computation toward concept-level reasoning, reducing inference FLOPs by approximately 34% while improving average accuracy by 2.69% across 12 zero-shot benchmarks.

2 Related Work

Standard LLMs allocate uniform computation to every token, ignoring that some tokens (e.g., predictable function words) require minimal processing while others (e.g., concept boundaries) demand more effort Kaplan et al. (2020); Hoffmann et al. (2022). Recent work explores adaptive allocation mechanisms. The Universal Transformer Dehghani et al. (2018) introduced recurrence in depth, applying the same transformation block repeatedly with a learned halting mechanism that determines when each position has been sufficiently refined. Mixture of Experts (MoE) models Shazeer et al. (2017) achieve conditional computation by routing each token to a subset of expert sub-networks. However, these approaches focus on parameter efficiency and scaling rather than fundamentally addressing the information density problem.

H-NET Hwang et al. (2025) directly addresses adaptive allocation through learned boundary detection. The model predicts semantic boundaries by analyzing local patterns (e.g., similarity between consecutive hidden states), segments the sequence into variable-length chunks, and processes the compressed chunk representations hierarchically. Critically, boundary detection is differentiable and trained end-to-end, allowing task-appropriate chunking strategies to emerge Hwang et al. (2025). This yields substantial gains: learned boundaries align with linguistic structures even without supervision, and compression (4-8× reduction) translates to quadratic attention savings Hwang et al. (2025). The approach implicitly allocates more computation to high-information boundaries where new concepts begin—precisely where prediction is most difficult. However, H-NET's primary focus is on efficient representation through hierarchical bit-level modeling rather than token-level generation in state-of-the-art autoregressive LLMs Hwang et al. (2025). This leaves unaddressed the critical problem of computational waste in modern decoder-only language models, where every token—regardless of its predictability or information content—receives identical processing through the full model depth. Our DCLM architecture bridges this gap by adapting H-NET's dynamic boundary detection principles to the token-level generation paradigm of current LLMs Brown et al. (2020); Touvron et al. (2023). By segmenting sequences into concept chunks and performing compressed reasoning before token-level decoding, DCLM enables adaptive computation allocation in the exact architectural context where it matters most: next-token prediction in large-scale autoregressive models.

3 Methodology

3.1 Overview

We describe the core design of DLCM (Figure 1). DLCM consists of four stages: token encoding, dynamic segmentation into latent concepts, concept-level autoregressive reasoning, and token-level decoding:

$$\begin{aligned} \mathbf{H} &= \mathcal{E}(\mathbf{x}), \\ \mathbf{C} &= \Phi(\mathbf{H}), \\ \mathbf{Z} &= \mathcal{M}(\mathbf{C}), \\ \hat{\mathbf{y}} &= \mathcal{D}(\mathbf{H}, \mathbf{Z}). \end{aligned} \tag{1}$$

3.2 DYNAMIC SEGMENTATION

We hypothesize that concept boundaries correspond to sharp transitions in latent representations. Given encoder outputs $\mathbf{H}$, boundary scores are computed via local representational dissimilarity:

$$p_t = \frac{1 - \cos(\mathbf{q}_{t-1}, \mathbf{k}_t)}{2}, \quad \mathbf{q}_t = \mathbf{W}_q \mathbf{h}_t, \ \mathbf{k}_t = \mathbf{W}_k \mathbf{h}_t. \tag{2}$$

Tokens are grouped into variable-length segments according to predicted boundaries and pooled into concept representations. A global regularization mechanism maintains a target compression ratio while allowing content-adaptive granularity (details in Appendix).

3.3 CONCEPT-LEVEL REASONING

The concept-level backbone $\mathcal{M}$ is a causal Transformer operating on the compressed concept sequence $\mathbf{C} \in \mathbb{R}^{M \times d_{\text{concept}}}$. Its computational cost scales with $M \ll L$, allowing deeper reasoning to be performed on semantic units rather than surface tokens:

$$\mathbf{Z} = \mathcal{M}(\mathbf{C}). \tag{3}$$

3.4 TOKEN-LEVEL DECODING

Token predictions are reconstructed via causal cross-attention to the reasoned concepts. For token representations $\mathbf{H}$ and concept representations $\mathbf{Z}$, decoding is defined as:

$$\hat{\mathbf{y}} = \mathcal{D}(\mathbf{H}, \mathbf{Z}), \tag{4}$$

where causality ensures that each token attends only to concepts formed up to its position. This design bridges heterogeneous token- and concept-level representations, allowing concept-level reasoning to directly guide next-token prediction.

3.5 IMPLEMENTATION AND EFFICIENT CROSS-ATTENTION

Packed Sequence Training. We adopt the variable-length (VarLen) training strategy from FlashAttention Dao et al. (2022) so that global compression statistics are computed over a diverse mixture of tokens. This is essential for stabilizing batch-level compression regularization when sequence lengths vary across samples.

Normalization for Heterogeneous Representations. Bridging token-level and concept-level representations introduces heterogeneous activation statistics. Following prior work Henry et al. (2020); Dehghani et al. (2023), we apply RMSNorm to queries and keys before attention computation to stabilize training:

$$\mathbf{Q}' = \mathrm{RMSNorm}(\mathbf{Q}), \quad \mathbf{K}' = \mathrm{RMSNorm}(\mathbf{K}). \tag{5}$$

This normalization is applied consistently across both self-attention and cross-attention modules.

Training Objective. The overall objective combines next-token prediction with a global compression regularizer:

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda \mathcal{L}_{\text{aux}}, \tag{6}$$

where $\mathcal{L}_{\text{aux}}$ enforces a target average compression ratio at the batch level while allowing content-adaptive variation.

Efficient Cross-Attention via Concept Replication. Decoder cross-attention must handle variable-length alignments between tokens and latent concepts, which leads to irregular attention patterns in a naive $L \times M$ formulation. Direct implementations with dynamic masks incur substantial overhead due to irregular memory access.

To address this, we adopt a concept replication strategy that converts token–concept attention into a standard causal pattern. For each token position, the corresponding concept representation is replicated according to segment lengths:

$$\begin{aligned} \tilde{\mathbf{K}} &= \mathrm{repeat\_interleave}(\mathbf{K}, \text{segment\_lengths}), \\ \tilde{\mathbf{V}} &= \mathrm{repeat\_interleave}(\mathbf{V}, \text{segment\_lengths}). \end{aligned} \tag{7}$$

This aligns the key/value sequence length with the token sequence length, enabling the use of optimized FlashAttention kernels with standard causal masking. Conceptually, this is equivalent to cross-attention with shared keys and values within each concept segment, analogous to grouped-query attention.

4 DATA

To ensure experimental reproducibility, we build our corpus entirely from open-source data and tokenize it using the DeepSeek-v3 Liu et al. (2024) tokenizer. Our corpus spans multiple domains, including web text (English and Chinese), mathematics, and code, forming a comprehensive foundation for core language understanding across linguistic, factual, and reasoning abilities.

The composition of our pretraining data is intentionally designed to serve two critical objectives. First, it balances breadth and specialization: web text provides broad natural language coverage, while mathematics and code enhance structured reasoning. Second, and more importantly for our architecture, this diversity is essential for learning robust dynamic segmentation. By exposing the model to domains with drastically different information densities (e.g., highly structured code syntax vs. verbose natural language prose), we force the learned boundary predictor to discover content-adaptive segmentation strategies that generalize across diverse tasks. English and Chinese web text are weighted more heavily to ensure multilingual alignment, while specialized datasets like MegaMath-Web and OpenCoder-Pretrain are included to fine-tune the model's handling of high-entropy transitions.

To demonstrate the architectural benefits of DLCM rather than gains from data curation, we do not apply aggressive filtering; instead, we use data whose quality aligns with standard open-source corpora. Table 1 summarizes the statistics.

Table 1: **Statistics of the pretraining data.** Total corpus: 1,000B tokens.

Source	**Ratio**	**Tokens**
Nemotron-CC Su et al. (2025)	50%	500B
MAP-CC Du et al. (2024)	25%	250B
OpenCoder Huang et al. (2025)	15%	150B
MegaMath Zhou et al. (2025)	10%	100B
Total	**100%**	**1,000B**

5 SCALING LAWS FOR DLCM

To determine the optimal architecture and hyperparameters for DLCM, we conduct a comprehensive exploration using scaling laws. We first introduce a decoupled optimization strategy to handle the heterogeneous nature of our architecture, followed by the mathematical formulation of our scaling objectives.

5.1 DECOUPLED μP FOR HETEROGENEOUS ARCHITECTURES

Our architecture contains heterogeneous components with unequal widths at the token and concept levels. To ensure stable optimization and zero-shot hyperparameter transfer across scales, we adopt a decoupled Maximal Update Parametrization (μP), assigning separate width-dependent scaling rules to token-level modules and the concept-level backbone. Specifically, learning rates and initialization variances are scaled inversely with the respective module widths, ensuring consistent feature learning dynamics across compression ratios and model sizes. Empirical validation confirms that μP-derived hyperparameters tuned on a small proxy model transfer reliably to larger scales (Appendix F).

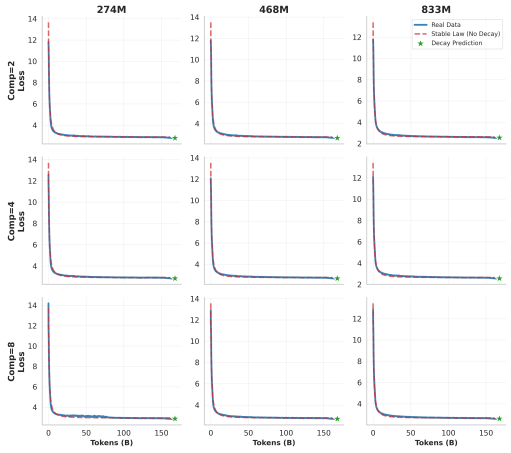

Figure 2: **Full training trajectory fit.** Comparison between predicted and empirical loss across model sizes (274M–833M), compression factors $R \in \{2, 4, 8\}$, and training budgets. The joint fit achieves $R^2 > 0.98$.

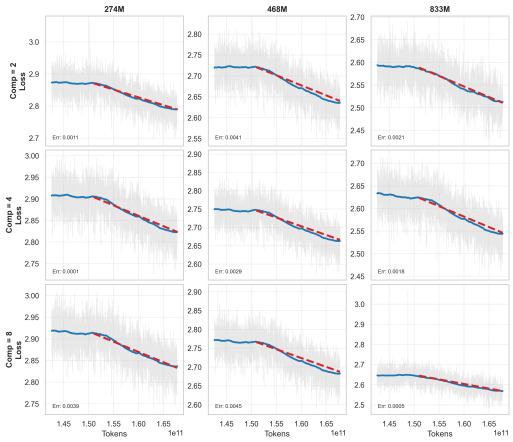

Figure 3: **Decay-phase fit.** Fit on the final portion of training tokens, validating that the WSD scaling law captures late-stage behaviour with $R^2 = 0.93$.

5.2 Compression-Aware Scaling Laws

We extend the Chinchilla scaling framework to explicitly account for hierarchical compression and architectural decomposition. The resulting loss model is:

$$L(N, D, R, P) = E_0 + \frac{A_{\text{token}}}{(N(1-P))^{\delta_1}} + \frac{A_{\text{concept}} R^{\gamma}}{(NP)^{\delta_2}} + \frac{A_{\text{data}}}{D^{\alpha}}. \tag{8}$$

This formulation disentangles token-level processing, concept-level reasoning, and data scaling under fixed compute. The compression exponent γ captures the efficiency gain from reallocating computation to the concept backbone.

Across all model sizes, we find that moderate compression ($R = 4$) and a majority allocation of parameters to the concept backbone yield the best loss–FLOPs trade-off, balancing training stability and reasoning capacity (Figure 4).

5.2.1 Decay-Phase Power Law

Since our training protocol involves Weight-Sharing-and-Decay (WSD), we explicitly model the late-stage regime. We fit a simplified decay law to the fractional loss reduction Δ_{decay} in the 90%–99% token window:

$$\Delta_{\text{decay}} = k\, L_{\text{stable}}^{a} R^{b} N^{c} \tag{9}$$

Obtained via log-linear regression, this model achieves $R^2 = 0.93$, accurately predicting late-stage loss drops across all scales.

6 Experiments

6.1 Main Results

We compare DLCM against a parameter-matched baseline that follows the LLaMA Touvron et al. (2023) architecture. Both models are trained from scratch on our proprietary dataset, using the same global batch size, learning rate, and sequence length as reported in the LLaMA paper. Each model is trained on 1T tokens. Results on 12 standard zero-shot benchmarks are summarized in Table 2.

Our model follows an *encoder–compressor–decoder* architecture with learned concept circulation, explicitly redistributing computation from uniform token-level processing to adaptive concept-level

reasoning. As a consequence, we do not expect uniform gains across all benchmarks. Instead, performance differences directly reflect the architectural bias induced by semantic compression and boundary-aware compute allocation.

Overall, DLCM achieves an average accuracy of **43.92%**, surpassing the baseline score of 41.23% by **+2.69%**. However, these gains are highly non-uniform across tasks, revealing a clear separation between reasoning-dominant benchmarks and those that rely on fine-grained token-level alignment.

Reasoning-Dominant Tasks. We observe consistent and often substantial improvements on benchmarks that emphasize multi-step reasoning, hypothesis selection, and implicit commonsense inference. Notable gains are achieved on **CommonsenseQA (+1.64%)**, **HellaSwag (+0.67%)**, **OpenBookQA (+3.00%)**, **PIQA (+2.42%)**, and both **ARC Easy (+2.61%)** and **ARC Challenge (+1.77%)**. These tasks are characterized by non-uniform information density, where prediction difficulty concentrates around semantic transitions rather than being evenly distributed across tokens. By compressing locally predictable spans and allocating the majority of model capacity to a high-dimensional concept backbone, DLCM focuses computation on structurally salient regions. This behavior is consistent with our loss distribution analysis in Section 7.2, which shows systematic loss reduction near concept boundaries.

Granularity-Sensitive Text Understanding. In contrast, we observe mild regressions on **BoolQ (-1.47%)** and **RACE (-0.72%)**. These benchmarks depend heavily on fine-grained sentence-level entailment, polarity resolution, and subtle lexical cues. The encoder–compress–decode paradigm inevitably reduces token-level granularity within concept interiors, which can obscure micro-level distinctions required for such tasks. Importantly, this degradation is localized rather than uniform: while boundary tokens are modeled more accurately, mid-concept positions may trade off fine-grained precision for improved global coherence. This trade-off manifests as the U-shaped loss profile observed in our mechanistic analysis.

Knowledge and Multilingual Benchmarks. For encyclopedic knowledge evaluation, we observe mixed behavior. While **C-Eval (+1.71%)** benefits from adaptive segmentation enabled by the Global Parser, slight regressions appear on **MMLU (-0.30%)** and **CMMLU (-0.24%)**. These datasets reward relatively uniform factual recall across tokens, leaving less opportunity for boundary-aware compute reallocation. This result further supports our central claim: DLCM is structurally optimized for reasoning under non-uniform information density, rather than uniform memorization-heavy retrieval.

Architecture and Parameter Efficiency. Although DLCM contains nearly **2**$\times$ the total parameters of the baseline model (2.3B vs. 1.3B), this increase is deliberately concentrated in the concept-level backbone ($d_{\text{concept}} = 3072$). Because this backbone operates on a sequence compressed by $4\times$, the effective FLOPs per inference step remain comparable to the smaller baseline. This validates our core design principle: shifting computation from redundant token-level processing to dense concept-level reasoning enables substantially larger effective capacity without incurring proportional inference cost.

6.2 Analysis: Compute Allocation in Concept-Based Models

6.2.1 Experimental Setup

To isolate the impact of concept-based compression on model behavior, we conduct a controlled comparison between our proposed concept model and a standard Transformer baseline. Both models utilize the same backbone architecture (1.3B parameters) and were trained on an identical subset of 100B tokens from our pretraining corpus. This ensures that any observed differences in loss distribution are attributable solely to the compression mechanism and architectural changes, rather than discrepancies in training data or compute budget.

6.2.2 Loss Distribution Analysis

To understand how the model allocates computational resources, we evaluate the loss distribution across relative positions within concepts. We randomly select 600 validation samples and align token-level losses by their position within a segmented concept (e.g., the i-th token in a concept).

Table 2: **Performance Comparison: DLCM vs. Baseline.** Zero-shot accuracy (%) categorized by task type. Improvements are shown in green and regressions in red.

Task / Category	DLCM (Ours)	Baseline	Diff.
Multi-choice General Knowledge / Common Sense			
Commonsense QA	**21.38**	19.74	+1.64
HellaSwag	**46.66**	45.99	+0.67
Winogrande	**57.22**	56.20	+1.02
OpenBookQA	**26.80**	23.80	+3.00
PIQA	**75.52**	73.10	+2.42
ARC Challenge	**34.81**	33.04	+1.77
ARC Easy	**69.91**	67.30	+2.61
MMLU	25.40	**25.70**	-0.30
Multi-choice Text Understanding			
BoolQ	62.54	**64.01**	-1.47
RACE	35.31	**36.03**	-0.72
Culture / Multilingual Knowledge			
C-Eval	**26.08**	24.37	+1.71
CMMLU	25.23	**25.47**	-0.24
Average	**43.92**	41.23	**+2.69**

Table 3: **Architecture Details.**

Metric	Baseline	DLCM (Ours)
General & Dimension Settings		
Model Type	Transformer	**DLCM**
Total Params	1.3B	**2.3B**
Hidden Size (d_{token})	1,536	1,536
Main Hidden (d_{concept})	–	**3,072**
Interm. Hidden (Self / Cross)	4k	6k
Layer Configuration		
Total Layers	32	32
Enc / Back / Dec Layers	–	**10 / 16 / 6**
Attention Configuration		
Attn / Backbone Heads	24	**48**
KV / Backbone KV Heads	24	**12 / 24**

Figure 4 shows the average loss over the first 20 relative positions within each concept and the corresponding difference ΔL. The results reveal a distinct "U-shaped" improvement pattern:

Boundary Proficiency (Positions 0–2 and 16+). The concept model consistently outperforms the baseline near segment boundaries, indicating improved handling of semantic transitions. By explicitly modeling concept boundaries, the model reduces ambiguity at the start and end of semantic units, whereas the baseline treats tokens uniformly.

Internal Complexity (Mid-positions). In the middle of a concept (approximately positions 4–15), performance is more mixed. Localized degradations (red bars) suggest a trade-off where compression reduces fine-grained token-level precision in exchange for maintaining higher-level semantic coherence.

This reallocation aligns with our hypothesis: the concept model sacrifices uniform token-level predictability (resulting in minor degradation at specific internal positions) to gain superior performance at semantic boundaries and structurally critical tokens. This strategic trade-off allows the model to "spend" its capacity on maintaining global coherence, explaining the downstream improvements despite non-uniform loss reduction.

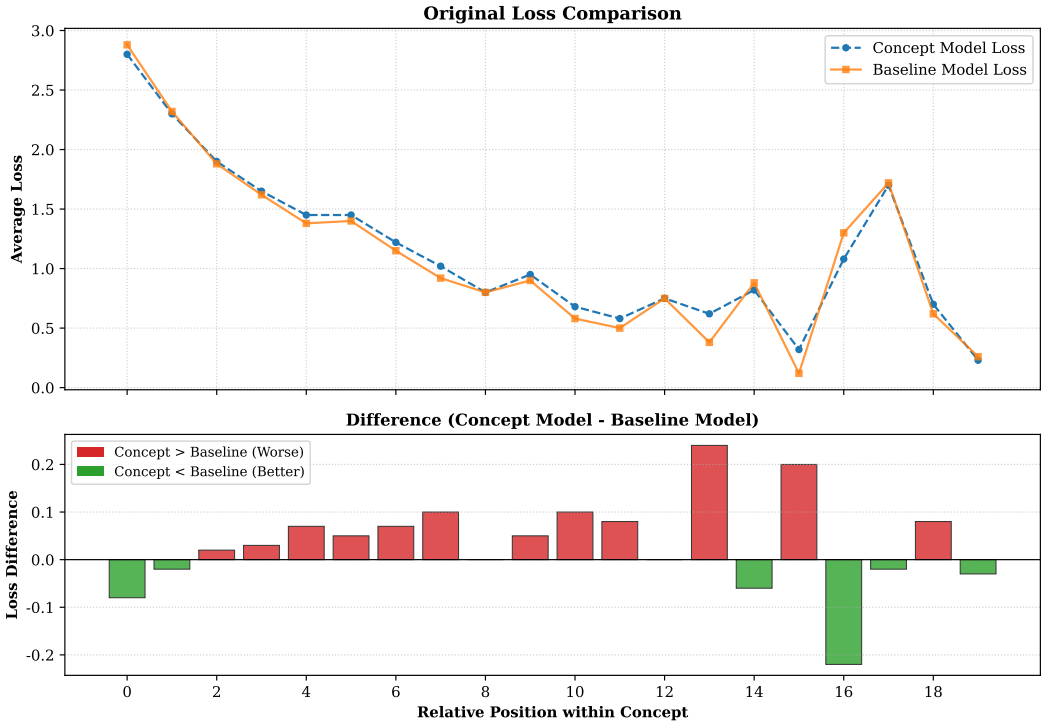

Figure 4: **Loss distribution across relative positions within concepts.** Top: Average loss of the concept model vs. the baseline. Bottom: $\Delta L = L_{\text{concept}} - L_{\text{baseline}}$, where green indicates improvement and red indicates degradation.

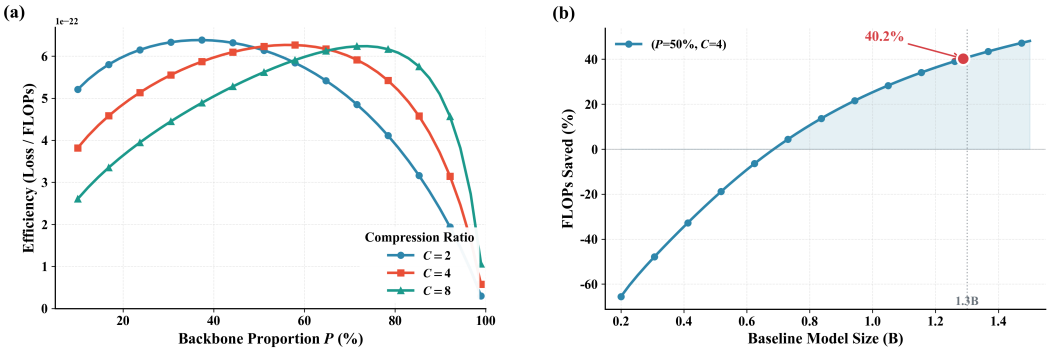

Figure 5: Efficiency analysis of the DLCM architecture. (a) Architectural efficiency (Loss/FLOPs) across backbone proportions P for different compression ratios R. (b) FLOPs savings compared to baseline models of varying sizes, with DLCM configured at $P = 60\%, R = 4$.

7 Ablation Studies

7.1 Global Regularization via Gradient Accumulation

To further stabilize the learned boundary predictor, we investigate an alternative regularization strategy: computing the compression ratio loss over accumulated training examples rather than individual sequences (Section B). This **global regularization** approach computes boundary statistics F_{global} and G_{global} across all tokens in K micro-batches.

We train two 2.3B parameter models for 1T tokens with a target compression ratio of $R = 2$: one with per-sequence regularization ("Normal") and one with global regularization ("Global Parser"). Table 4 presents the downstream performance and the actual realized compression ratios.

Table 4: **Ablation Study: Global Parser vs. Normal.** Both models target $R = 4$. Global Parser achieves a ratio closer to the target.

Task	Global Parser	Normal
ARC Challenge	**0.3038**	0.2858
ARC Easy	**0.6296**	0.6242
Commonsense QA	**0.2457**	0.2228
HellaSwag	**0.3507**	0.3499
OpenBookQA	0.3220	**0.3280**
PIQA	**0.6806**	0.6785
Avg. Improvement	**+2.1%**	–
Realized Ratio ($R \approx 4$)	**3.92**	3.15

The global regularization approach achieves consistently better performance across most tasks (5 out of 6). Crucially, as shown in the bottom row of Table 4, the Global Parser maintains a realized compression ratio ($\sim$3.9) much closer to the target (4.0) compared to the Normal formulation, which tends to degrade towards lower compression.

The key insight is that enforcing a fixed compression ratio per sequence is overly restrictive. Real-world data exhibits varying information density. By relaxing the constraint to operate at the batch level, the global regularization allows the model to learn adaptive behavior—compressing repetitive code more aggressively while preserving dense technical text—effectively allocating the compression "budget" where it matters most.

7.2 Content-Adaptive Compression Benefits

To verify the adaptive behavior enabled by global regularization, we analyzed the segmentation granularity across different domains. Table 5 presents empirical measurements of the average tokens per concept.

Content Type	Target 8×	Target 4×	Target 2×
Casual English	7.47	3.53	1.76
Casual Chinese	8.38	4.36	1.76
Technical English	10.58	3.85	1.92
Technical Chinese	6.09	3.27	1.76
Code	6.14	3.66	1.98
Math/Science	7.42	4.41	1.91

Table 5: Average tokens per concept across content types and compression ratios. Values represent the actual granularity achieved for each target compression setting.

The data reveals significant variation in compression density across content types. For instance, at the 8× target, Technical English retains significantly more tokens per concept (10.58) compared to Technical Chinese (6.09) or Code (6.14).

While the precise ranking of "optimal" length varies across compression targets (as noted in the fluctuation between content types), the **existence of this variation** is the critical finding. It confirms that the global regularization mechanism successfully decouples the compression objective from rigid per-sequence constraints. The model is not forcing a uniform segment length; instead, it adapts the granularity based on the inherent semantic density of the content. Code and structured text tend to be compressed into shorter, syntactic units, whereas dense prose is preserved in longer semantic chunks. This adaptivity—regardless of the specific order—allows the model to maximize information retention within the global compression budget.

8 CONCLUSION

We introduced **Dynamic Large Concept Models (DLCM)**, a hierarchical language modeling framework that departs from uniform token-level computation. By learning semantic boundaries from latent representations and shifting computation from tokens to variable-length concepts, DLCM enables reasoning to occur in a compact, semantically aligned space. We further proposed a compression-aware scaling law and a decoupled μP parametrization for principled and stable optimization in heterogeneous architectures. Empirically, DLCM achieves consistent improvements on reasoning-intensive benchmarks while reducing redundant computation, demonstrating a favorable accuracy–efficiency trade-off. These results suggest that effective scaling of language models depends not only on increasing parameters or data, but also on allocating computation according to semantic structure. More broadly, our findings highlight concept-level abstraction as a practical mechanism for integrating adaptive computation into autoregressive language models.

IMPACT STATEMENT

This work aims to advance the field of machine learning by improving the efficiency and scalability of large language models. Dynamic Large Concept Models (DLCM) introduce adaptive concept-level processing that reallocates computation toward semantically informative regions, enabling more efficient use of model capacity without increasing overall computational cost. We do not anticipate negative societal impacts beyond those commonly associated with large-scale language modeling.

REFERENCES

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness, 2022. URL https://arxiv.org/abs/2205.14135.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, Rodolphe Jenatton, Lucas Beyer, Michael Tschannen, Anurag Arnab, Xiao Wang, Carlos Riquelme, Matthias Minderer, Joan Puigcerver, Utku Evci, Manoj Kumar, Sjoerd van Steenkiste, Gamaleldin F. Elsayed, Aravindh Mahendran, Fisher Yu, Avital Oliver, Fantine Huot, Jasmijn Bastings, Mark Patrick Collier, Alexey Gritsenko, Vighnesh Birodkar, Cristina Vasconcelos, Yi Tay, Thomas Mensink, Alexander Kolesnikov, Filip Pavetić, Dustin Tran, Thomas Kipf, Mario Lučić, Xiaohua Zhai, Daniel Keysers, Jeremiah Harmsen, and Neil Houlsby. Scaling vision transformers to 22 billion parameters, 2023. URL https://arxiv.org/abs/2302.05442.

Xinrun Du, Zhouliang Yu, Songyang Gao, Ding Pan, Yuyang Cheng, Ziyang Ma, Ruibin Yuan, Xingwei Qu, Jiaheng Liu, Tianyu Zheng, et al. Chinese tiny llm: Pretraining a chinese-centric large language model. *arXiv preprint arXiv:2404.04167*, 2024.

Alex Henry, Prudhvi Raj Dachapally, Shubham Pawar, and Yuxuan Chen. Query-key normalization for transformers, 2020. URL https://arxiv.org/abs/2010.04245.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Siming Huang, Tianhao Cheng, J. K. Liu, Jiaran Hao, Liuyihan Song, Yang Xu, J. Yang, Jiaheng Liu, Chenchen Zhang, Linzheng Chai, Ruifeng Yuan, Zhaoxiang Zhang, Jie Fu, Qian Liu, Ge Zhang, Zili Wang, Yuan Qi, Yinghui Xu, and Wei Chu. Opencoder: The open cookbook for top-tier code large language models, 2025. URL https://arxiv.org/abs/2411.04905.

Sukjun Hwang, Brandon Wang, and Albert Gu. Dynamic chunking for end-to-end hierarchical sequence modeling, 2025. URL https://arxiv.org/abs/2507.07955.

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

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017.

Dan Su, Kezhi Kong, Ying Lin, Joseph Jennings, Brandon Norick, Markus Kliegl, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. Nemotron-cc: Transforming common crawl into a refined long-horizon pretraining dataset, 2025. URL https://arxiv.org/abs/2412.02595.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Greg Yang, Edward J. Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tensor programs v: Tuning large neural networks via zero-shot hyperparameter transfer, 2022. URL https://arxiv.org/abs/2203.03466.

Fan Zhou, Zengzhi Wang, Nikhil Ranjan, Zhoujun Cheng, Liping Tang, Guowei He, Zhengzhong Liu, and Eric P. Xing. Megamath: Pushing the limits of open math corpora. *arXiv preprint arXiv:2504.02807*, 2025. Preprint.

A Segmentation Examples

We provide representative examples of how boundary prediction behaves under different compression ratios across three content types: casual English text, Python code, and mathematical exposition. Predicted segment boundaries are denoted using a vertical bar (|).

A.1 Casual English Text

Original (90 tokens).

> So I've been trying to perfect my morning coffee routine lately. It's funny how something so simple can have so many variables. I started with a basic drip machine, which was fine, but a bit boring. Then I went down the French press rabbit hole – way more flavor, but the cleanup is a real hassle, you know?

Compression $8\times$ (11 segments).

> So I | 've been trying to perfect my morning coffee routine lately. | It's funny how something | so simple can have so many variables. | I started with | a basic drip machine, which | was fine, but a | bit boring. | Then I went down the French press rabbit hole | way more flavor, but the cleanup is a | real hassle, you know?

Compression $4\times$ (35 segments).

So I | 've been trying | to perfect | my morning coffee routine lately. | It's | funny how something | so simple can | have so many variables. | I started | with | a basic drip machine, which was fine, but | a bit boring.

Compression $2\times$ (56 segments).

So I | 've | been trying to | perfect | my | morning coffee routine | lately. | It | 's funny how | something so | simple | can | have so many | variables.

A.2 PYTHON CODE

Original (87 tokens).

```
import torch
from torch.utils.data import Dataset, DataLoader
class SimpleTextDataset(Dataset):
    """A simple dataset for loading text data."""
    def __init__(self, texts, tokenizer, max_length=128):
        self.texts = texts
        self.tokenizer = tokenizer
```

Compression $8\times$ (12 segments).

```
import torch | from torch.utils.data import Dataset, DataLoader
class SimpleTextDataset(Dataset):
"""A simple dataset for loading text data."""
def __init__(self, texts, tokenizer, max_length=128):
self.texts = | texts | self.tokenizer = | tokenizer
```

Compression $4\times$ (29 segments).

```
import torch | from torch.utils.data | import Dataset | , DataLoader
class Simple | TextDataset(Dataset): | """ | A | simple dataset for
    loading text data."""
def | __init__(self, texts, tokenizer, max_length= | 128):
self.texts = | texts
```

Compression $2\times$ (58 segments).

```
import | torch | from torch.utils.data import | Dataset, | DataLoader
class | Simple | TextDataset(Dataset): | """ | A simple | dataset for
    loading text data."""
def | __init__(self, | texts, | tokenizer | , | max_length= | 128):
self | .texts | = | texts
```

A.3 MATHEMATICAL TEXT

Original (95 tokens).

Euler's formula is a mathematical formula in complex analysis that establishes the fundamental relationship between the trigonometric functions and the complex exponential function. The formula states that for any real number x, $e^{ix} = \cos(x) + i\sin(x)$, where e is the base of the natural logarithm and i is the imaginary unit.

Compression $8\times$ (13 segments).

Euler's formula is a | mathematical formula in complex analysis that establishes | the fundamental relationship between the trigonometric functions and the complex exponential function. | The formula states that | for any real number x, e^{ix} = | $\cos(x)+i\sin(x)$, | where e | is the | base of the natural logarithm, i is the | imaginary unit.

Compression $4\times$ (32 segments).

Euler's | formula is | a mathematical formula in | complex analysis that establishes | the fundamental relationship between | the trigonometric functions and the | complex exponential function. | The formula states | that for any | real number x |, e^{ix} = | $\cos(x) + i\sin(x)$, | where e | is | the base of the natural logarithm.

Compression $2\times$ (61 segments).

Euler | 's | formula is | a | mathematical formula | in | complex analysis that | establishes | the fundamental relationship between | the trigonometric functions | and | the complex exponential | function. | The | formula states | that | for | any | real number x |, | e^{ix} = | $\cos(x)$ | + | $i\sin(x)$,

B Implementation Details

Packed Sequence Training. We adopt the Variable Length (VarLen) training strategy from FlashAttention Dao et al. (2022) to ensure that global compression statistics are computed over a diverse mixture of tokens. This is particularly important for stabilizing batch-level compression regularization when sequence lengths vary significantly across samples.

Normalization for Heterogeneous Representations. Bridging token-level and concept-level representations introduces heterogeneous statistical properties. Following prior work Henry et al. (2020); Dehghani et al. (2023), we apply RMSNorm to queries and keys before attention computation to stabilize training:

$$\mathbf{Q}' = \mathrm{RMSNorm}(\mathbf{Q}), \quad \mathbf{K}' = \mathrm{RMSNorm}(\mathbf{K}). \tag{10}$$

This normalization is applied consistently across both self-attention and cross-attention modules.

Training Objective. The overall objective combines next-token prediction with a global compression regularizer:

$$\mathcal{L} = \mathcal{L}_{\mathrm{CE}} + \lambda \mathcal{L}_{\mathrm{aux}}, \tag{11}$$

where $\mathcal{L}_{\mathrm{aux}}$ enforces a target average compression ratio at the batch level while allowing content-adaptive variation.

—

C Efficient Cross-Attention via Concept Replication

The decoder cross-attention in DLCM must handle variable-length mappings between tokens and latent concepts, leading to irregular attention patterns in the naive formulation. Specifically, tokens attend to a compressed concept sequence whose alignment depends on dynamically predicted segment boundaries.

Directly implementing such irregular $L \times M$ attention with dynamic masks incurs substantial overhead due to irregular memory access and mask construction. To address this, we adopt a *concept replication* strategy that converts the problem into a standard causal attention pattern.

For each token position, the corresponding concept representation is replicated using `repeat_interleave` according to segment lengths, producing aligned key and value sequences:

$$\tilde{\mathbf{K}} = \mathrm{repeat\_interleave}(\mathbf{K}, \text{segment\_lengths}), \quad \tilde{\mathbf{V}} = \mathrm{repeat\_interleave}(\mathbf{V}, \text{segment\_lengths}). \tag{12}$$

This transformation aligns the key/value sequence length with the token sequence length, enabling the use of highly optimized FlashAttention kernels with standard causal masking. Conceptually, this can be viewed as a specialized form of self-attention in which keys and values are locally constant within each concept segment.

—

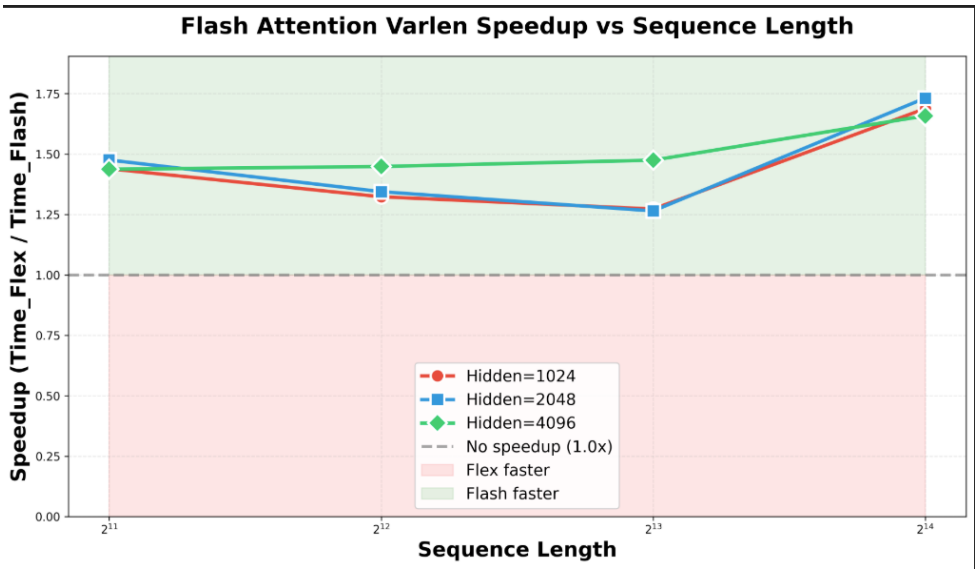

Figure 6: Flash Attention Varlen speedup ($T_{\text{flex}}/T_{\text{flash}}$) vs. sequence length. This plot visualizes the data from Table 6.

Table 6: Performance comparison (Batch=1, Heads=32, Interval=6).

Seq Len	Hidden	Flex (ms)	Flash Varlen (ms)	**Speedup**
2048	1024	32.35	22.48	**1.44**×
2048	2048	33.31	22.58	**1.48**×
2048	4096	32.42	22.56	**1.44**×
4096	1024	59.75	45.15	**1.32**×
4096	2048	60.72	45.17	**1.34**×
4096	4096	65.88	45.48	**1.45**×
8192	1024	116.35	91.42	**1.27**×
8192	2048	114.65	90.66	**1.26**×
8192	4096	142.75	96.79	**1.47**×
16384	1024	314.35	186.21	**1.69**×
16384	2048	323.53	186.83	**1.73**×
16384	4096	315.69	190.38	**1.66**×

D KERNEL BENCHMARKS

We benchmark the proposed concept replication strategy against a baseline implementation using Flex Attention with dynamic masking. Benchmarks are conducted using isolated kernel profiling to measure end-to-end attention latency under varying sequence lengths and hidden dimensions.

Hidden sizes (1024, 2048, 4096) are selected to evaluate algorithmic scalability rather than to match the exact architectural dimensions of DLCM. We report the relative speedup $T_{\text{flex}}/T_{\text{flash}}$.

Across all tested configurations, FlashAttention with concept replication consistently outperforms Flex Attention, with speedups ranging from $1.26\times$ to $1.73\times$. Notably, the performance advantage increases with sequence length, indicating that the overhead of dynamic mask generation and irregular memory access dominates at longer contexts.

These results suggest that concept replication offers a practical and scalable solution for integrating dynamic segmentation with efficient autoregressive decoding, without requiring custom attention kernels.

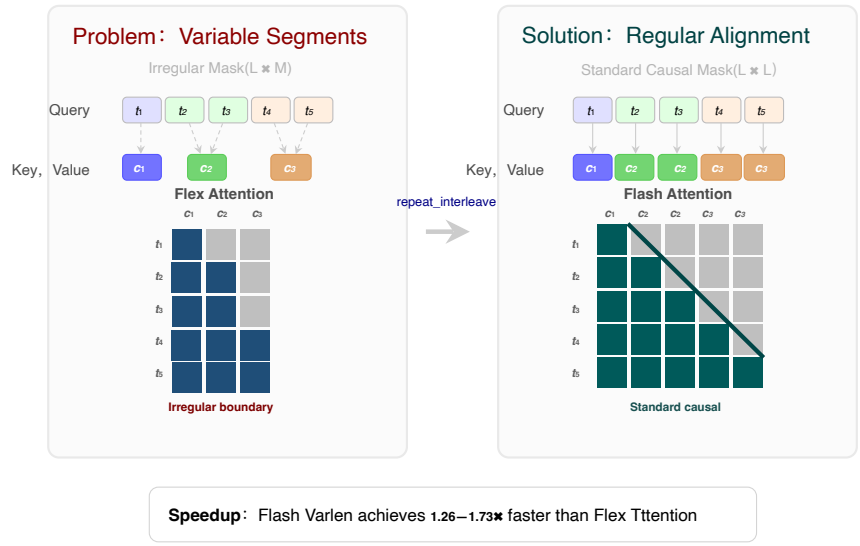

Figure 7: **Cross-Attention Optimization via Concept Replication.** *Left:* The decoder's cross-attention creates an irregular $L \times M$ mask due to variable token-to-concept mappings. *Right:* By replicating concepts via `repeat_interleave` to match token positions, we obtain a standard $L \times L$ causal mask, enabling optimized Flash Attention kernels.

E Details of Dynamic Segmentation

E.1 Boundary Sampling Strategy

While boundary probabilities p_t are learned continuously from latent representations, downstream concept formation requires discrete segmentation decisions. During training, we adopt stochastic sampling to encourage exploration, while inference uses deterministic thresholding for stability.

Specifically, during training we sharpen probabilities using a temperature parameter and sample boundary indicators $b_t \sim \text{Bernoulli}(p_t)$. During inference, we apply a hard threshold rule $b_t = [p_t \geq 0.5]$. This decoupling allows stable optimization while preserving adaptive granularity.

E.2 Global Compression Regularization

To maintain a target average compression ratio R while allowing content-adaptive local variation, we impose constraints at the global batch level. Let $\mathcal{T}$ denote all tokens across the distributed batch. We define:

$$G_{\text{global}} = \frac{1}{|\mathcal{T}|} \sum_{(i,t)\in\mathcal{T}} p_{i,t}, \tag{13}$$

$$F_{\text{global}} = \frac{1}{|\mathcal{T}|} \sum_{(i,t)\in\mathcal{T}} b_{i,t}. \tag{14}$$

We optimize an auxiliary loss that encourages F_{global} to match the expected boundary rate $1/R$, while allowing per-sequence deviation. This mechanism, referred to as the *Global Parser*, enables stable, content-adaptive compression across heterogeneous batches.

—

F DECOUPLED μP FOR HETEROGENEOUS ARCHITECTURES

F.1 WIDTH-DEPENDENT PARAMETERIZATION

Unlike standard transformers with uniform width, DLCM contains heterogeneous components operating at different dimensionalities. Token-level modules operate at width d_{token}, while the concept-level backbone operates at width d_{concept}.

We define width multipliers relative to a base width d_{base}:

$$s_{\text{token}} = \frac{d_{\text{token}}}{d_{\text{base}}}, \quad s_{\text{concept}} = \frac{d_{\text{concept}}}{d_{\text{base}}}. \tag{15}$$

Initialization variances and optimizer hyperparameters are scaled inversely with the corresponding width multiplier, following the Maximal Update Parametrization (μP) principle.

F.2 LEARNING RATE AND INITIALIZATION SCALING

Hidden-layer weights are initialized with variance $\sigma^2_{\text{base}} s^{-1}$, where s corresponds to the component width. Learning rates for token-level modules and the concept backbone are scaled as:

$$\eta_{\mathcal{E},\mathcal{D}} = \eta^{\text{base}}_{\text{token}} s^{-1}_{\text{token}}, \tag{16}$$

$$\eta_{\mathcal{M}} = \eta^{\text{base}}_{\text{concept}} s^{-1}_{\text{concept}}. \tag{17}$$

Embedding layers and bias parameters retain fixed learning rates. The decoder output projection is scaled during the forward pass to ensure logits remain $O(1)$ across widths.

F.3 HYPERPARAMETER TRANSFER VALIDATION

Following Yang et al. (2022), we tune base learning rates on a small proxy model (87M parameters) and verify zero-shot transfer to larger models (up to 834M). Empirical sweeps confirm that the optimal learning rates predicted by μP remain near-optimal across scales, validating the effectiveness of decoupled parameterization for heterogeneous architectures.

—

G EXTENDED SCALING LAW ANALYSIS

G.1 EXPERIMENTAL GRID

Scaling experiments vary the compression ratio $R \in \{2, 4, 8\}$ and the concept-backbone parameter ratio $P \in \{30\%, 50\%, 70\}$ across three model scales (274M, 468M, 834M parameters). All models are trained on identical token budgets, and scaling exponents are shared globally across configurations to avoid post-hoc overfitting.

G.2 DECAY-PHASE POWER LAW

To account for late-stage optimization behavior under Weight-Sharing-and-Decay (WSD), we fit a simplified power law to the fractional loss reduction in the final training window:

$$\Delta_{\text{decay}} = k L^{a}_{\text{stable}} R^{b} N^{c}. \tag{18}$$

The resulting fit achieves $R^2 = 0.93$, accurately capturing late-stage convergence behavior across compression ratios and model sizes.

G.3 FULL-TRAJECTORY FITTING

We jointly fit the full training trajectory using the proposed compression-aware scaling law. The combined estimator achieves $R^2 > 0.98$ across all evaluated configurations, indicating strong generalization across architectural and data scales.

—

H EFFICIENT CROSS-ATTENTION VIA CONCEPT REPLICATION

Dynamic segmentation induces irregular token-to-concept attention patterns that are inefficient to implement directly. To address this, we adopt a concept replication strategy that converts irregular $L \times M$ attention into standard $L \times L$ causal attention.

For each token position, the corresponding concept representation is replicated using `repeat_interleave` based on segment lengths:

$$\tilde{\mathbf{K}} = \text{repeat\_interleave}(\mathbf{K}, \text{segment\_lengths}), \tag{19}$$

$$\tilde{\mathbf{V}} = \text{repeat\_interleave}(\mathbf{V}, \text{segment\_lengths}). \tag{20}$$

This enables the use of highly optimized FlashAttention kernels without requiring custom attention operators.

—

I KERNEL BENCHMARKS

We benchmark the concept replication strategy against Flex Attention with dynamic masking using isolated kernel profiling. Hidden sizes of 1024, 2048, and 4096 are evaluated to assess algorithmic scalability.

Across all tested configurations, FlashAttention with concept replication consistently outperforms Flex Attention, with speedups ranging from $1.26\times$ to $1.73\times$. Notably, the relative advantage increases with sequence length, indicating superior scalability under long-context decoding.

These results demonstrate that dynamic segmentation can be integrated into autoregressive decoding efficiently using existing optimized kernels.

I.1 ANALYSIS: END-TO-END DISCRETE BOUNDARY LEARNING VS. DECOUPLED SEGMENTATION

This experiment compares two boundary prediction mechanisms for sequence compression: a learned neural predictor with compression rate regularization (Section B) and a rule-based predictor using cosine similarity. Starting with sequences of length $L = 8192$, we track the average compressed length during training.

Figure 4 reveals starkly different behaviors. The **learned predictor (red)** exhibits severe instability: after initial compression to $\sim$2000 tokens, the compressed length steadily increases, eventually stabilizing at $\sim$4300 tokens ($1.9\times$ compression). This "creep-up" indicates the model progressively learns to compress less over time. In contrast, the **rule-based predictor (purple)** demonstrates exceptional stability, rapidly converging to $\sim$2000 tokens ($4\times$ compression) and maintaining this level consistently throughout training.

The learned predictor's instability stems from conflicting optimization objectives. Despite the compression rate regularization term $\mathcal{L}_{\text{aux}}$ designed to maintain the target compression ratio R, the primary cross-entropy (CE) loss creates much stronger gradients that penalize information loss and discourage compression:

$$\nabla_\theta \mathcal{L}_{\text{total}} = \underbrace{\nabla_\theta \mathcal{L}_{\text{CE}}}_{\text{anti-compression}} + \lambda \underbrace{\nabla_\theta \mathcal{L}_{\text{aux}}}_{\text{pro-compression}} \tag{21}$$

Since $\|\nabla_\theta \mathcal{L}_{\text{CE}}\| \gg \lambda \|\nabla_\theta \mathcal{L}_{\text{aux}}\|$, the CE loss eventually dominates, forcing the predictor to reduce segmentation despite the regularization term.

The rule-based predictor avoids this conflict through a fixed decision rule: $p_t = \frac{1-\cos(h_t, h_{t+1})}{2}$, with boundaries inserted when $p_t > \tau$. While the representations h_t are learned, the segmentation rule itself is not optimized by the CE loss. This decoupling prevents the task loss from undermining the compression mechanism, ensuring stable and controllable compression ratios through the threshold parameter τ.