MULTISCALE BYTE LANGUAGE MODELS
A Hierarchical Architecture for Causal Million-Length Sequence Modeling

Eric Egli [1] **Matteo Manica** [1] **Jannis Born** [1]

Abstract

Bytes form the basis of the digital world and thus are a promising building block for multimodal foundation models. Yet the excessive length of bytestreams requires new architectural paradigms for Byte Language Models. Therefore, we present the Multiscale BLM (MBLM), a model-agnostic hierarchical decoder stack that allows training with context windows of 5M bytes on single GPU in full precision. Our experiments demonstrate that hybrid Transformer/Mamba architectures are efficient in handling extremely long byte sequences during training while achieving near-linear generational efficiency. Source code has already been publicly released and MBLM can be installed from PyPI at: https://github.com/ai4sd/multiscale-byte-lm.

1. Introduction

To address the computational overhead of long sequence modeling, prior work has aimed to mitigate the quadratic complexity of Transformers with computationally more efficient, hierarchical Transformers (Yu et al., 2023; Pagnoni et al., 2024; Nawrot et al., 2021) or Mamba models optimized for fast inference (Wang et al., 2024). However, these approaches depend on modality-specific setups which limit their generalization. We here introduce the Multiscale Byte Language Model (MBLM), a model– and modality-agnostic architecture for causal byte language modeling that composes Transformer, Mamba or even LSTM blocks. MBLMs extend the MegaByte hierarchy (Yu et al., 2023) to an unlimited number of stages, and predict the next byte of a large input bytestream by refining input sequence representations through a hierarchy of generic decoder models, while enabling precise control over stage parallelism. With a 2D MBLM composed of a global Mamba (Wang et al., 2024; Dao & Gu, 2024) and a local Transformer decoder (Vaswani et al., 2017), we demonstrate that hybrid hierarchies minimize computational requirements during both training and inference, outperforming other architectures on long sequences (>100K bytes). MBLMs provide granular control over the discretization intensity – we demonstrate that a 3D MBLM can efficiently train on sequences of up to 5M bytes on a single GPU. Yet, it has to be emphasized that there is no free lunch: MBLMs trade memory with performance, thus a 2D MBLM will always be inferior to a plain 1D Transformer *on a context window that the 1D model can fit*. MBLMs come into play to train on sequence lengths that yield memory errors with traditional approaches. Last, while we evaluate MBLMs on (multimodal) byte-streams, they can equally be combined with standard subword tokenization and thus find application in general NLP.

[1]IBM Research Europe, Zurich, Switzerland. Correspondence to: Jannis Born <jab@zurich.ibm.com>.

Proceedings of the 2^{nd} Workshop on Long-Context Foundation Models, Vancouver, Canada. 2025.

2. Related work

MBLMs builds upon MegaByte (Yu et al., 2023), a causal BLM with a hierarchy of two Transformer decoders, enabling subquadratic self-attention and context windows up to 1.2M bytes. MegaByte processes patch representations of the input sequence with a global decoder, refines these representations, and feeds them into a local model to autoregressively predicts bytes. Incorporating Mamba (Gu & Dao, 2023) at the byte level, MambaByte (Wang et al., 2024) demonstrated, without the need of a hierarchy, superior performance over MegaByte in a FLOP-controlled setting across various datasets. Our goal is to generalize the hierarchy of MegaByte to arbitrary depth and allow to compose flexibly Transformer and Mamba blocks. A concurrent, improvement of MegaByte is the Byte Latent Transformer (BLT), which dynamically splits bytes into patches based on the entropy of the next byte (Pagnoni et al., 2024).

3. Methods

3.1. MBLM

The MBLM consists of N causal decoder *models* $M_{i \leq N}$ that are stacked hierarchically. The first $N-1$ stages

$M_1, \ldots, M_{N-1}$ contain **global models**. The final stage M_N contains the **local model**. Each model M_i operates on inputs with a hidden state of dimension $D_i \in \mathbb{N}$ and a patch/context size $P_i \in \mathbb{R}$. Inputs to an MBLM module are sequences of B batches, each of length L. Similar to MegaByte (Yu et al., 2023), MBLMs scale through input length compression and aim to operate on sequences of length $L_{\max} = \prod_{i=1}^{N} P_i$, see Figure A1 for a 3D MBLM.

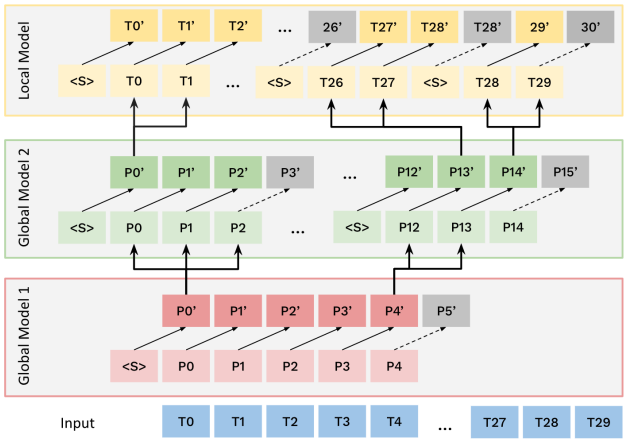

Figure 1: A 3D MBLM with two global and one local decoders and corresponding patch sizes $P_1 = 5, P_2 = 3, P_3 = 2$, operating on an input sequence $\mathbf{x} = \{x_0, x_2, \ldots, x_{29}\}$. Inputs to each stage are prepended with a trainable start token <S>. The updated patch representations of the input sequence output by the global models are added to the inputs of the next stage. The local model generates individual bytes, and the final outputs are concatenated.

Patch Embedder. MBLMs employ a patch embedder that ingests and embeds a sequence $\mathbf{x} \in \mathbb{R}^{B \times L}$, adds positional encodings and chunks it into patches for each stage:
First, **embed** the bytes in $\mathbf{x}$ for each stage i and **reshape** to a nested sequence of patch embeddings:

$$\mathbf{x}_i^{\text{emb}} \xrightarrow{\text{reshape}} \mathcal{P}_i^{\text{emb}} \in \mathbb{R}^{B \times P_1 \times \ldots \times P_N \times D_N} \tag{1}$$

If L cannot be factored into $P_1 \times \ldots \times P_N$, the inner sequence lengths P_2 to P_N are padded. Then, we **project token embeddings to patches** for the global stages. Recall that all $\mathcal{P}_i^{\text{emb}} \in \mathbb{R}^{B \times P_1 \times \ldots \times P_N \times D_N}$ are of the same shape. For each stage, we flatten the embeddings and apply a linear projection to the model dimension of stage i:

$$\begin{aligned} \boldsymbol{W}_i^{\text{patch}} : \; & \mathbb{R}^{B \times P_1 \times \ldots \times P_i \times (\boldsymbol{P_{i+1}} \times \ldots \times \boldsymbol{P_N} \times \boldsymbol{D_N})} \\ \rightarrow \; & \mathbb{R}^{B \times P_1 \times \ldots \times \boldsymbol{P_i} \times \boldsymbol{D_i}} \end{aligned} \tag{2}$$

Global Model Projections

The global models perform **inter-patch** modeling by capturing dependencies between patches and output updated patch representations. They are added to the token embeddings of the next stage, allowing patches to receive global sequence information from the leftward context. In order to process all patches contained in $\mathcal{P}_i^{\text{emb}}$ in parallel with M_i, we reshape $\mathcal{P}_i^{\text{emb}}$ to a new batch dimension K_i:

$$\mathcal{P}_i^{\text{emb}} \in \mathbb{R}^{B \times P_1 \ldots \times P_i \times D_i} \xrightarrow{\text{pack}} \mathcal{P}_i^{\text{emb}'} \in \mathbb{R}^{\boldsymbol{K_i} \times P_i \times D_i} \tag{3}$$

$$\text{with } K_i = B \cdot \prod_{j=1}^{i-1} P_j \quad \forall i > 1.$$

For deep hierarchies, $K \in \mathbb{R}$ becomes large. For this reason, all but the first stage trade performance for memory efficiency by leveraging gradient checkpointing: Instead of processing all K patches in parallel, we *optionally* divide them into c smaller chunks and recompute intermediate activations during the backward pass. This approach allows for much larger batch sizes and input sequences, albeit at the cost of slower training. To propagate information, outputs of global stage i are linearly projected to the dimension of the next stage $i+1$ with $\boldsymbol{W}_i^{\text{global}} : \mathbb{R}^{D_i} \rightarrow \mathbb{R}^{D_{i+1}}$ and added to the patch embedding $\mathcal{P}^{\text{emb}'}$ of the next stage. Expressed as a recurrence relation:

$$\overbrace{\mathcal{P}_i^{\text{in}}}^{\text{Input to } M_i} = \overbrace{\mathcal{P}_i^{\text{emb}'}}^{\text{Embedding of stage } i} + \overbrace{\mathcal{P}_{i-1}^{\text{out}} \boldsymbol{W}_{i-1}^{\text{global}}}^{\text{Output of } M_{i-1}} \tag{4}$$

$$\overbrace{\mathcal{P}_i^{\text{out}}}^{\text{Output of } M_i} = \underset{c}{\text{concat}} \left(M_i(\mathcal{P}_i^{\text{in}}) \right) \tag{5}$$

$$\text{with } \mathcal{P}_i^{\text{in}} \in \mathbb{R}^{\frac{K_i}{c} \times P_i \times D_i}$$

The first global stage has no parent patch representation. The base case of the recurrence and input to the first model in the hierarchy is thus given by $\mathcal{P}_1^{\text{in}} = \mathcal{P}_1^{\text{emb}'} \in \mathbb{R}^{B \times P_1 \times D_1}$.

Local Intra-Patch Modeling

The input to the local stage is given by $\mathcal{P}_N^{\text{in}} \in \mathbb{R}^{K_N \times P_N \times D_N}$. Unlike the global models, whose primary role is to contextualize patches, the local model performs byte-level **intra-patch** modeling by autoregressively predicting individual bytes starting from the trainable start token. The output of the local model is then projected to logits through a linear layer and reshaped to output $\mathbf{y} \in \mathbb{R}^{B \times L \times V}$.

Stage Models. Hierarchical sequence models have mostly leveraged Transformers and aimed to reduce the quadratic cost of self-attention. However, we show that even models with linear scaling like Mamba (Gu & Dao, 2023) can benefit from compression through patchification. Notably, a 1D MBLM with a Mamba block is roughly equivalent to Mambabyte (Wang et al., 2024) while a 2D MBLM with two Transformers is equivalent to Megabyte (Yu et al., 2023).

3.2. Datasets & Evaluation

We evaluate the performance of MBLMs in terms of language modeling on the Project Gutenberg (PG19) dataset

(Rae et al., 2019). PG19 contains 28,752 English-language books (11.6 GB). We select this dataset for comparability to prior art (Yu et al., 2023; Wang et al., 2024) and because, on average, each book is around 411 KB, which allows long-range language modeling on consecutive bytes in the same document. Additional statistics are included in Appendix B. We use bits-per-byte (BPB) (Gao et al., 2020) as the primary evaluation metric for byte-level modeling and report word-level perplexities (PPL) to facilitate comparisons with future work. BPB, related to perplexity, quantifies the average number of bits needed to encode each byte of data and can be seen as a compression measure where a lower value indicates a higher probability of correctly predicting the next byte (Rae et al., 2021): $\text{BPB} = \log_2(e^{\ell_{\text{byte}}}) = \frac{\ell_{\text{byte}}}{\ln 2}$ where ℓ_{byte} is the observed average negative log-likelihood stemming from a byte vocabulary. All MBLMs are matched to 360M parameters and trained on 8 NVIDIA A100 SXM4 80 GB GPUs in parallel using a custom-built distributed PyTorch trainer. For further details on model, training and evaluation metrics, please see Appendix A, C and D.

4. Results

Scaling Byte Language Models. As the first three-stage (3D) hierarchical model of its kind, an MBLM comprising a global Mamba followed by two Transformer decoders can process byte sequences of 5M bytes during training on a single A100 80 GB GPU with standard automatic mixed precision. Within 15 hours of training the MBLM processed 100 GB of UTF-8 bytes and achieved 2.448 BPB on the PG19 test set (Figure 2). Due to the unprecedented nature

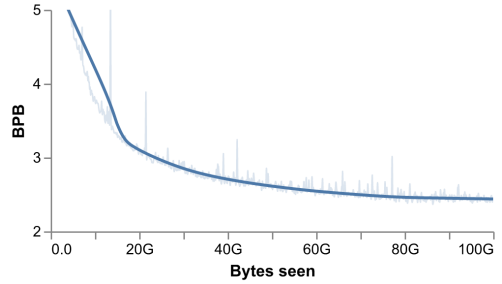

Figure 2: Training loss progression of a 3D MBLM with 350M parameters and a context window of 5M bytes on a single GPU.

of a 5M context window training, this experiment naturally lacks comparison to previous work. To undermine this, Table 1 shows how the same Transformer decoder scales to twice the sequence length when incorporated into a two- or three-stage MBLM, thanks to optimized computational efficiency through input compression. Naturally, since MBLMs scale by compressing the input sequence, regular 1D models outperform hierarchical models when the sequence fits into memory. This underscores that hierarchical architectures are specifically designed for extremely long-sequence modeling. Moreover, for a given sequence length $L_{\max}$, hierarchies with larger global sequence lengths will consume more memory but perform better (e.g., $L_{\max} = 10,000$ and a 2D MBLM with $P = [5000, 2]$ vs. $P = [1000, 10]$).

MBLM \ Context size	8192	16384	32768
1D Transformer	30.5	56.2	*out of memory*
2D Transformer	19.6	35.8	68.2
3D Transformer	**15.9**	**28.2**	**53.0**

Table 1: Memory usage (in GB) during training of three 360M parameter MBLMs on a single GPU. Hierarchical Transformers scale to 2x the sequence length. Batch size was 2.

Performant Hierarchies. When comparing various types of 2D MBLMs to MegaByte (Yu et al., 2023) (i.e., a 2D MBLM with two Transformers), we find that both hybrid and Mamba-based MBLMs outperform a Transformers-based MegaByte model when trained on 200 GB of PG19 text (Table 2). Unlike previous hierarchical architectures,

Hierarchy	Global & local model	Test PPL	Test BPB
MegaByte	Transformer (2x)	278.79	1.370
MBLM	Mamba, Transformer	163.29	1.240
MBLM	Mamba, Mamba	**119.37**	**1.164**

Table 2: Comparison of MegaByte to MBLM architectures on byte sequences of length 98,304. Hybrid and Mamba-based MBLMs outperform MegaByte on the same amount of data. All models used patch sizes of (8192, 12) for global/local model.

MBLMs can be configured with an unlimited amount of stages and different decoder models at each stage. On context windows exceeding 1M bytes, hybrid models again outperform Transformer MBLMs (Table 3). To fit the 3D

3D MBLM configuration	Test PPL	Test BPB
Transformer (3x)	5420.66	2.092
Mamba, Transformer (2x)	5351.71	**2.089**

Table 3: After training on 200 GB with a context window of more than 1M bytes (1,048,576), hybrid MBLMs with a first global Mamba perform slightly better than pure Transformer hierarchies.

models in Table 3 on a single GPU, we use a physical batch size of 1. With inner model context sizes of (8192, 16, 8), the input tensor at stage 3 is given by $\mathbf{x}_3 \in \mathbb{R}^{131072 \times 8 \times D_3}$. Previous multiscale models like MegaByte (Yu et al., 2023) advocate for full parallelism at every stage which is often infeasible for extremely long inputs. Instead MBLMs batch the 2nd and 3rd stage into 10 and 20 chunks, respectively, and re-computes intermediate activations. This enables each MBLM to train at approximately 75-80% memory utilization on a single A100 80 GB GPU.

Inference Context Extrapolation. To investigate inference throughput and context extrapolation capabilities, we evaluate four different MBLMs on byte input sequences ranging from 8,192 to 991,232 in length L. These include two 1D modules trained with an 8K context window and two 2D modules trained with a 100K context window. Since the 1D Transformer uses rotary position embeddings (Su et al., 2024), input length is only bound by compute requirements. Efficient inference solutions for 1D models, such as KV caches (Ott et al., 2019), have been widely adopted. In its recurrent mode, Mamba can even process each step in constant time by passing the SSM state through the recurrence. However, implementing a dedicated inference pipeline in a hierarchical setting poses significant challenges because patches form a compressed representation of chunks of the input sequence, making it infeasible to cache and reuse previously computed results effectively. As a result, all MBLMs containing a Mamba-2 block still compute a parallel scan over the sequence during inference. While both SSM representations are expected to be numerically equal, this results in longer generation times per token and constrains the model's scalability linearly with respect to the context size. Figure 3 visualizes the time-per-byte for 1D

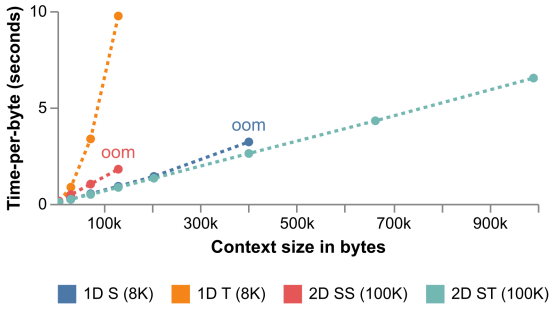

Figure 3: The time it takes to generate a single byte as a function of context size for 1D and 2D MBLMs. Hybrid MBLMs exhibit near-linear generational efficiency.

and 2D MBLMs as a function of context length. This result demonstrates that hybrid hierarchies with a global Mamba and local Transformer decoder are able to generate tokens with near-linear efficiently up to 1M bytes. Instead, generating bytes on extended context windows quickly becomes infeasible for regular Transformers due to their $\mathcal{O}(L^2)$ complexity. Surprisingly, when extending the context windows of the 1D models during inference by 120x (Zhao et al., 2024; Ben-Kish et al., 2024), they perform competitively to the 2D models *trained* with this context length, suggesting that much of the context is ignored by the models (Figure A2). We hypothesized that this is due to the PG19 data and questioned its suitability for large context extrapolation by conducting an ablation study with Llama 2-7B (Touvron et al., 2023) (pre-trained on a context size of 4K) and focus on small context sizes up to 8,192 bytes.

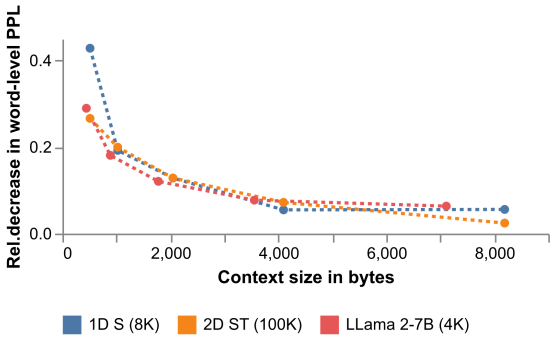

Figure 4: Relative improvement in word-level perplexities for consecutive context lengths for the 1D SSM, 2D SSM-Transformer and Llama baseline

Figure 4 shows the relative improvement in word-level perplexities for consecutive context lengths for the 1D SSM, 2D SSM-Transformer and Llama: On PG19, all models perform strictly better for larger context size. However, given a context length $\geq$ 4,000 bytes, the relative decrease in perplexity diminishes even for a performant LLM like Llama, indicating that 4K bytes are likely enough to reasonably predict the next few bytes in a PG19 book.

5. Discussion

Here we introduced the Multiscale Byte Language Model (MBLM), a hierarchical, model-agnostic architecture capable of scaling to the unprecedented length of 5M bytes on a single GPU. MBLMs operates in stages of autoregressive models: Byte sequences are divided into patches, embedded, and refined as they pass through the hierarchy, culminating in a local model that autoregressively predicts bytes within each patch. This approach enables efficient processing of very long byte sequences through compression. While Mamba-based hierarchies performed best, hybrid models combining Mamba for global stages and Transformer decoders for local stages achieved an optimal balance between performance and computational efficiency. Hybrid models also converged faster and exhibit near-linear generational efficiency during inference. We recommend evaluating MBLMs on tasks requiring long contexts, such as multimodal document summarization or needle in a haystack tasks and investigating their performance when scaled to billions of parameters. The MBLM architecture, available as a PyPi package, provides a modular and flexible framework for further development. With the right technical extensions, we believe MBLMs are well-suited to process sequences spanning tens of millions of bytes and driving future innovations in hierarchical architectures.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. Our work is particularly focused on improving context window length in language models which may allow future algorithms to sift through larger amounts of data in one shot. There are many potential societal consequences of such work, none which we feel must be specifically highlighted here.

References

Ben-Kish, A., Zimerman, I., Abu-Hussein, S., Cohen, N., Globerson, A., Wolf, L., and Giryes, R. Decimamba: Exploring the length extrapolation potential of mamba, 2024. URL https://arxiv.org/abs/2406.14528. under review.

Dao, T. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024.

Dao, T. and Gu, A. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *International Conference on Machine Learning (ICML)*, 2024.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., Presser, S., and Leahy, C. The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Gu, A. and Dao, T. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Kudo, T. and Richardson, J. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In Blanco, E. and Lu, W. (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 66–71, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-2012. URL https://aclanthology.org/D18-2012/.

Nawrot, P., Tworkowski, S., Tyrolski, M., Kaiser, Ł., Wu, Y., Szegedy, C., and Michalewski, H. Hierarchical transformers are more efficient language models. *arXiv preprint arXiv:2110.13711*, 2021.

Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Grangier, D., and Auli, M. fairseq: A fast, extensible toolkit for sequence modeling. In Ammar, W., Louis, A., and Mostafazadeh, N. (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pp. 48–53, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-4009. URL https://aclanthology.org/N19-4009/.

Pagnoni, A., Pasunuru, R., Rodriguez, P., Nguyen, J., Muller, B., Li, M., Zhou, C., Yu, L., Weston, J., Zettlemoyer, L., Ghosh, G., Lewis, M., Holtzman, A., and Iyer, S. Byte latent transformer: Patches scale better than tokens, 2024. URL https://arxiv.org/abs/2412.09871.

Rae, J. W., Potapenko, A., Jayakumar, S. M., Hillier, C., and Lillicrap, T. P. Compressive transformers for long-range sequence modelling. *arXiv preprint*, 2019. URL https://arxiv.org/abs/1911.05507.

Rae, J. W., Borgeaud, S., Cai, T., Millican, K., Hoffmann, J., Song, F., Aslanides, J., Henderson, S., Ring, R., Young, S., Rutherford, E., Hennigan, T., Menick, J., Cassirer, A., Powell, R., van den Driessche, G., Hendricks, L. A., Rauh, M., Huang, P.-S., et al. Scaling language models: Methods, analysis & insights from training gopher. *ArXiv*, abs/2112.11446, 2021. URL https://api.semanticscholar.org/CorpusID:245353475.

Su, J., Ahmed, M., Lu, Y., Pan, S., Bo, W., and Liu, Y. Roformer: Enhanced transformer with rotary position embedding. *Neurocomput.*, 568(C), March 2024. ISSN 0925-2312. doi: 10.1016/j.neucom.2023.127063. URL https://doi.org/10.1016/j.neucom.2023.127063.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., et al. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Wang, J., Gangavarapu, T., Yan, J. N., and Rush, A. M. Mambabyte: Token-free selective state space model. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=X1xNsuKssb.

Wu, S., Tan, X., Wang, Z., Wang, R., Li, X., and Sun, M. Beyond language models: Byte models are digital world simulators, 2024.

Yu, L., Simig, D., Flaherty, C., Aghajanyan, A., Zettlemoyer, L., and Lewis, M. Megabyte: Predicting million-byte sequences with multiscale transformers. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 78808–78823, 2023.

Zhao, L., Feng, X., Feng, X., Zhong, W., Xu, D., Yang, Q., Liu, H., Qin, B., and Liu, T. Length extrapolation of transformers: A survey from the perspective of positional encoding. In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N. (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 9959–9977, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.582. URL https://aclanthology.org/2024.findings-emnlp.582/.

A. Model Details

All our models pre-trained on the PG19 dataset are matched to 360 million parameters, which is achieved simply by varying the number of layers for each model in the hierarchy. Model names are abbreviated; **S** stands for Mamba-2 and **T** for Transformers.

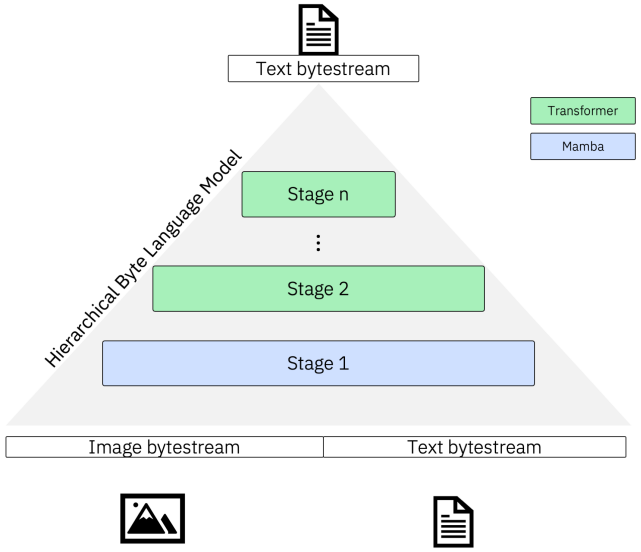

Figure A1: The Multiscale Byte Language Model (MBLM) processes bytestreams from any modality that can be serialized into bytes. Each stage in the hierarchical architecture employs a decoder model to generate a new representation for input patches, which is subsequently passed to the next stage as augmented input. The final output of the MBLM is a bytestream formed by concatenating the outputs of the last stage, n.

Common model configuration

We keep the model-specific configuration constant: For **Mamba-2** models, we use a model dimension of 1024, an SSM state expansion factor 128, a local convolution width of 4, a block expansion factor of 2 and 64 as the head dimension. Mamba-2 models operate without positional embeddings. **Transformer** models use a model dimension of 1024, 16 attention heads of dimension 64 and a feed-forward expansion factor of 2. The attention layers employ rotary positional embeddings (RoPE) (Su et al., 2024). Positional encodings are only employed in 2D or 3D multiscale hierarchies for Transformer decoder models. This ensures that all models can be used for context extrapolation experiments. An exception to the default configuration is the 5 million context size experiment, which uses hidden dimensions of size 256. Throughout all experiments, we used the same context/patch sizes for hierarchical constellations, which are summarized in Table A1.

Context size \ Hierarchy	1D	2D	3D
8192	8192	1024, 8	256, 8, 4
16384	16384	2048, 8	512, 8, 4
32768	32768	4096, 8	1024, 8, 4
98304	-	8192, 12	-
1048576	-	-	8192, 16, 8
5000000*	-	-	1000, 200, 25

Table A1: Context/patch sizes across all experiments, denoted from global to local. The MBLM with a 5M context size (denoted with *) uses different model configuration than others, as noted above.

Language modeling experiments

Table A2 summarizes the number of layers for each of the models pre-trained on PG19 (Rae et al., 2019). We train models with context sizes larger than 98,304 on 200 billion bytes and all others on 30 billion bytes form PG19.

Model \ Input length	8,192	16,384	32,768	98,304	524,288	1,048,576	5,000,000
1D T	42	41	39	-	-	-	-
1D S	54	-	-	-	-	-	-
2D SS	24, 28	-	-	27, 24	-	-	-
2D ST	25, 21	-	-	24, 21	-	-	-
2D TS	25, 20	-	-	-	-	-	-
2D TT	22, 19	22, 19	22, 19	21, 18	-	-	-
3D STT	-	-	-	-	14, 12, 9	9, 8, 8	1, 1, 1
3D TTT	15, 12, 10	15, 12, 10	15, 12, 10	-	-	8, 7, 7	-

Table A2: The number of layers, denoted from global to local, for the 360 million parameter models.

We do not use MBLMs' gradient checkpointing for 8K models. For the 100K we use two chunks at the second stage and for the 3D-1M model we use 10 chunks at stage 2 and 20 chunks at stage 3.

B. Dataset Details

For all experiments, we use a byte vocabulary of $255 + 1$ tokens, with token ID 257 designated as the `<pad>` token to enable sequence padding within minibatches. This method aligns with byte-level models like bGPT (Wu et al., 2024), which employ an `<eop>` (end-of-patch) token with ID 257 to pad patches. While our `<pad>` token is not used during training on PG19, it is required during inference to pad patches for prompts that are shorter than the context size. The individual books in PG19, stored in `.txt` format, are read as bytes from disk and combined into a single `bytearray` data structure without textual preprocessing. While most PG19 books are within the ASCII character set, some contain Unicode characters outside the ASCII range and are thus encoded in UTF-8. From this byte sequence, we sample subsequences for training based on the context size of the corresponding model. While our data ingestion process is simple and unbiased, the lack of language-specific preprocessing introduces noisy input data, including ASCII control characters like `NUL` and `CR` (carriage return), which are usually absent from subword-based vocabularies. A significant portion of the data comprises space characters and newlines. Table A3 contains statistics for PG19, which we use to derive word-level perplexities.

	L_B	L_W	L_B/L_W
Train	11,678,184,667	1,966,200,384	5.9395
Validation	17,733,002	3,007,061	5.8971
Test	41,289,101	6,966,499	5.9268

Table A3: PG19 (Rae et al., 2019) dataset statistics. L_B is the number of UTF-8 encoded bytes, L_W the number of space-separated words. To count the words, we read all books into a single Unicode string and then split at all common whitespace characters (“n, “r, “t, “f) using Python's `str.split`.

C. Training Recipes

Hyperparameters for all PG19 experiments are listed in Table A4. Prior to our experiments, we validated a few hyperparameters suggested by prior art to train hierarchical models and SSMs respectively:

Learning rate Unlike Megabyte (Yu et al., 2023), we find that using a peak learning rate of $1e{-}3$ results in the best performance on PG19 among the tested 1D and 2D models and other learning rates $4e{-}4$, $8e{-}4$

Positional encodings for Mamba In preliminary experiments, we test the addition of fixed positional embeddings to the input sequence and find that the SSM performs best without any positional information, regardless of the position in the hierarchy.

Similar to Yu et al. (2023) and Wang et al. (2024), we use the AdamW optimizer with $\beta = (0.9, 0.95)$ with a linear warmup of 10% of the total gradient steps followed by cosine annealing. While the physical batch sizes used vary between experiments and are set to maximize GPU efficiency, we use gradient accumulation to arrive at the same gradient step (see Table A4). We keep all models in full `float32` precision and use PyTorch's Automatic Mixed Precision package to enable

Parameter	Value
Learning rate	0.001
Gradient step	48
Gradient clipping	1
Attention/SSM dropout	0

Table A4: Hyperparameters for the PG19 experiments.

`float16` precision for the backward passes and the integration of FlashAttention (Dao, 2024). Our Mamba-2 models are built using the `mamba-ssm`[1] (version 2.2.2) and `causal-conv1d`[2] (version 1.4.0) packages. The Transformer models are based on `megabyte-pytorch`[3] (version 0.3.5), which also served as a baseline implementation for MBLM. Any parameter we did not explicitly mention above would use the default value in the corresponding package versions above. We use PyTorch 2.4.1 and train all models on 8 NVIDIA A100 SXM4 80GB GPUs in parallel using a custom-built distributed trainer. For each experiment, we follow a data-parallel approach and split the training datasets among the GPUs.

D. Evaluation

Given the average negative log-likelihood ℓ_{subword}, Gao et al. (2020) define bits-per-byte as:

$$\text{BPB} = \frac{L_S}{L_B} \log_2(e^{\ell_{\text{subword}}}) \tag{6}$$

where L_S and L_B is the length of the dataset in subwords/tokens and length of the dataset in bytes, respectively. If we solely model on bytes, i.e., $L_S = L_B$, this definition can be simplified to:

$$\text{BPB} = \ell_{\text{byte}} \log_2 e = \frac{\ell_{\text{byte}}}{\ln 2} \tag{7}$$

In order to translate between the metrics, we can derive *word-level* perplexities (Wang et al., 2024), which are often used in language modeling. Word-level perplexities are more interpretable and better aligned with human understanding as they measure uncertainty at the level of entire words rather than subwords or bytes. They also facilitate fairer comparisons between models with different tokenization schemes by reducing (though not eliminating) biases introduced by tokenizer differences through normalization. With L_W denoting the number of space-separated words in a corpus, PPL_{word} can be derived from either ℓ_{byte} or ℓ_{subword}:

$$\text{PPL}_{\text{word}} = \exp\left(\frac{L_B}{L_W}\ell_{\text{byte}}\right) \qquad \text{PPL}_{\text{word}} = \exp\left(\frac{L_S}{L_W}\ell_{\text{subword}}\right) \tag{8}$$

L_S, L_B and L_W for PG19 are summarized in Table A3. In practice, both PPL and BPB can be understood as scaled variants of the cross-entropy between two distributions. Importantly, minimizing cross-entropy will result in smaller absolute values for both PPL and BPB, which all indicate a more performant model.

E. Llama-7B Word-Level Perplexities

We calculate perplexity on subword context sizes varying from 64 to 8,192 on a quantized Llama-2-7B model[4] (Touvron et al., 2023) using the `llama.cpp` project[5]. We recall that there are a total of $L_W = 3,007,061$ whitespace-separated words in the PG19 validation set. Tokenizing the validation set with the SentencePiece-based Llama tokenizer results in $L_S = 5,106,780$ subwords. Using the `llama.cpp` CLI does not give us the direct negative log-likelihood, $\ell_{subword}$, so we have to convert the obtained subword-level *perplexity* values to word-level perplexities by continuing from Equation 8. Since $\ell_{subword} = \ln(\text{PPL}_{\text{subword}})$, using basic logarithm rules, we derive:

$$\text{PPL}_{\text{word}} = e^{\left(\frac{L_S}{L_W}\ell_{subword}\right)} = e^{(\ln \text{PPL}_{\text{subword}})\frac{L_S}{L_W}} = \text{PPL}_{\text{subword}}^{\frac{L_S}{L_W}} \tag{9}$$

[1] https://github.com/state-spaces/mamba
[2] https://github.com/Dao-AILab/causal-conv1d
[3] https://github.com/lucidrains/MEGABYTE-pytorch
[4] https://huggingface.co/TheBloke/Llama-2-7B-GGUF/blob/main/llama-2-7b.Q5_K_S.gguf
[5] https://github.com/ggerganov/llama.cpp

From the above numbers, $\frac{L_S}{L_W} \approx 1.6982$, meaning that a single word in PG19's validation set word consists of approximately 1.7 Llama-subwords. For comparison, Wang et al. (2024) report $\frac{L_S}{L_W} = 1.45$ when fitting a SentencePiece tokenizer (Kudo & Richardson, 2018) on PG19's validation set. We also note that there are ≈ 3.4724 bytes per subword, which we use to convert between subword- and byte-level context lengths.

F. Additional Figures

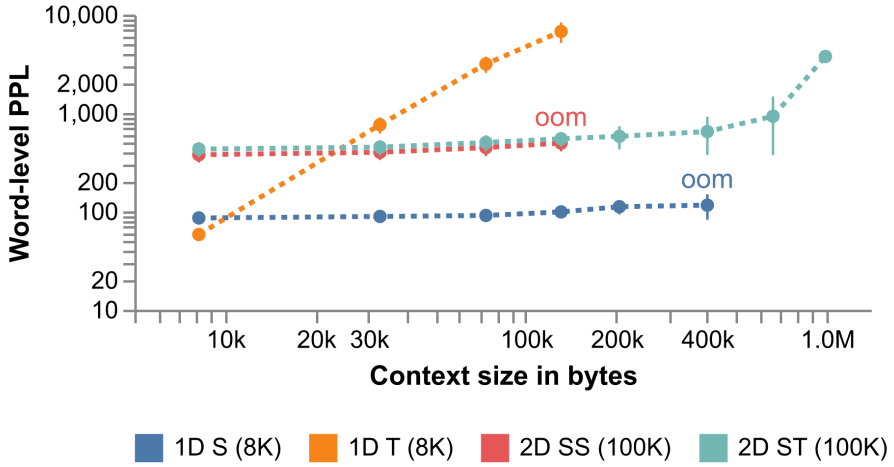

Figure A2: Word-level perplexities as a function of context size for 1D and 2D MBLMs.

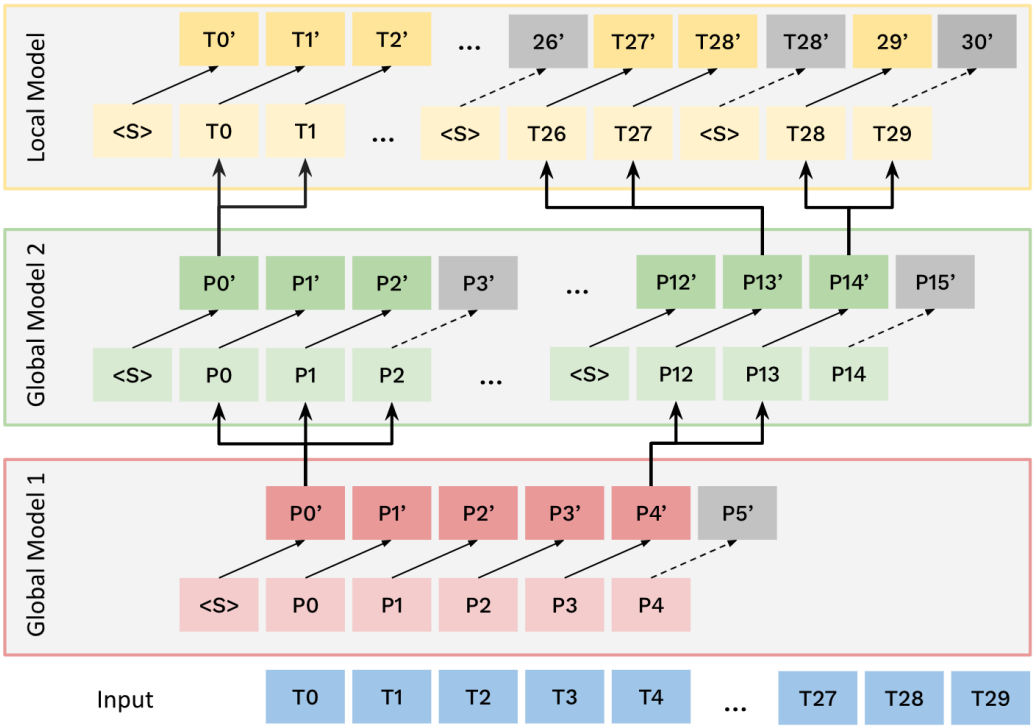

Figure A3: A 3D MBLM module with two global and one local decoder models and corresponding patch sizes $P_1 = 5, P_2 = 3, P_3 = 2$, operating on an input sequence $\mathbf{x} = \{x_0, x_2, \ldots, x_{29}\}$. Inputs to each stage are prepended with a trainable start token <S>. The updated patch representations of the input sequence output by the global models are added to the inputs of the next stage. The local model generates individual bytes, and the final outputs are concatenated.

G. PG19 Generational Examples

In all generated samples, whitespaces are removed. Based on the prompt, presented in red, 256 bytes are generated and converted to a string. For conciseness, we show the start and end of the prompt and omit some content, which is denoted by an ellipsis. All prompts originate from books contained in the PG19 validation set (Rae et al., 2019).

2D-100K SSM-TRANSFORMER

There is no sugar cane known anywhere to-day in th (...) number of otherwise remarkably distinct forms may be recognized some of which were illustrated in a previous publication, Bureau of Agriculture Bulletin No. 27, Citriculture in the Philippines, 1913, and referred to C. histrix with the statement that "some of these forms unquestionably will be recognized as subspecies on closer study, or possibly as separate species." Since then several plants of this type in the citrus collection assembled and antimony in Aloeso are also combined. Those in the first fifteenth century likewise have been distinguished, but the species occurs in emergency and consists of two parts in the caterpillar which though often taken no leaves are next reduced. Specimens of other Granata have been found in a museum which was found generally at Manila . . and A. and occurred in Colorado and other small communities. Buddhisches, tompsing scientifically the time occupied in scientific research, have previously

Dawson had often come in and out of the room durin (...) motion was required to alleviate the agony of fury that seized upon the Cagots at such times. In this desire for rapid movement, the attack resembled the Neapolitan tarantella; while in the mad deeds they performed during such attacks, they were not unlike the northern Berserker. In Béarn especially, those suffering from this madness were dreaded by the pure race; the Béarnais, going to cut their wooden clogs in the great forests that layaround them, accumulated their old equipment, and spent their supplies under small vessels and towards the pillaginian regions which they would have drawn from it. But in so far as the Fairies were concerned on the matter, they were left to grant their reasons to the cautious Battery and his friends. The only alley of considerable importance, and whence too many of the adventurous scientific men, have appeared when they entered the school, or where the place has been called a masquerade and the choi

2D-100K SSM-SSM

He had an envelope in his starboard mitten, and, c (...) are forbidden crossing this property, under penalty of the law.' But land! I'd used that short-cut ever sence I'd been in Bayport–which was more'n a year–and old man Davidson and me was good friends, so I cal'lated the signs was intended for boys, and hove ahead without paying much attention to 'em. 'Course I knew that the old man–and, what was more important, the old lady–had gone abroad and that the son was expected down, but that didn't make it any good. The time was fast enough in the morning to launch up a fat pirate lord in a thresh-open carriage an' walk out to the dock, that can stand it on fifty yards with his head turned to look at the cruise. It's most lucky for a while. He's not at home this time. I guess he's gone to the pier pond. Said I was there and he says that that he can tell his 'and that's more the truth. To-morrow night I'll go down to see how Jim Buck works. I can't see how he's going. There's a chance for

In a House of Commons that counted Pitt, Fox, Burk (...) usiness with his secretaries. Hundreds of times, probably, I have called him out of bed, and have, in short, seen him in every situation and in his most unreserved moments. As he knew I should not ask anything of him, and as he reposed so much confidence in me as to be persuaded that I should never use any information I might obtain from him for any unfair purpose, he talked freely before me of men and things, of actual, meditated, or questionable, general matters, and of all matters that require the utmost collision. On one occasion the project proposed to place my position at the head of some five or six influential members of parliament on the line of steamships, or those who had a distinct presidential interest in the theatre, a misdemeanour and inducement making me the interest of the community towards the committee; the people of the country and statesmen of eminence at Lichfield could not be more maturely charged than I am than Mr. Gr