# LiteByte: Efficient and Fast-Adapting MLPs for Online Byte-Level Prediction

**Yu Mao** [1] **Yuyan Lin** [2 3] **Xue Liu** [2 3] **Chun Jason Xue** [1]

## Abstract

Transformer-based architectures have become the de facto standard for sequence modeling, largely due to their scalability and ability to capture long-range dependencies. However, their high computational cost, reliance on long contexts, and limited adaptability under online updates make them less suitable for small-scale or streaming scenarios. In this paper, we revisit MLP-based models for byte-level next-token prediction under fully online training. We propose a simple yet effective architecture, LiteByte, which is composed of alternating feedforward layers and soft-shared expert projections, without attention or recurrence. Each sample is dynamically routed through a learned mixture of compact shared MLPs, enabling adaptive token-wise transformations with minimal overhead. Despite its simplicity, our model achieves significantly faster convergence and lower perplexity than Transformer, RNN, and vanilla MLP baselines on Enwik8, Text8, and a curated Dickens corpus. It also demonstrates superior runtime efficiency in terms of inference latency and throughput. We further argue that the soft expert mechanism introduces a reusable and modular structure that may serve as a lightweight adapter or differentiable controller in broader applications such as LoRA-style fine-tuning or modular agents.

## 1. Introduction

Transformer-based models have become the dominant paradigm for sequence modeling, with widespread adoption in language (Vaswani et al., 2017; Brown et al., 2020; Touvron et al., 2023), code (Chen et al., 2021), and multi-modal systems (Radford et al., 2021). Their success is driven by self-attention's ability to model long-range dependencies and by the scalability of their architecture on large datasets. However, this dominance has led to a strong implicit assumption: that attention is essential for all sequence modeling tasks, and that scale is necessary for success.

This assumption is increasingly being questioned in practical low-resource and small-scale learning settings. In many real-world scenarios, such as online log processing, local adaptation on edge devices, or streaming user data, models must be trained or fine-tuned with limited memory, low latency, and no access to large offline corpora. Under such constraints, the suitability of attention-heavy architectures becomes unclear. These models often require long context windows, incur high memory overhead, and remain difficult to update incrementally (Tay et al., 2022).

In this work, we revisit sequence modeling in the small-scale setting, where the focus is not on achieving state-of-the-art accuracy, but on understanding what minimal architectural elements are truly necessary for adaptation and prediction under limited resources. We propose a simple yet effective alternative to Transformers: a fully MLP-based model for byte-level next-token prediction, which is named as LiteByte. LiteByte consists of a stack of feedforward layers without recurrence or attention. Each token is processed independently, and the model parameters are continuously updated as new data arrives.

Despite its simplicity, LiteByte exhibits two key properties desirable in small-scale settings. First, it converges rapidly to the current distribution, with significantly faster training loss reduction compared to parameter-matched Transformer and MLP baselines. Second, it achieves competitive perplexity and accuracy on byte-level benchmarks such as enwik8 and OpenWebText subsets, despite lacking long-range modeling capacity.

These results support the hypothesis that attention is not a universal requirement for effective sequence modeling. In byte-level tasks, where local dependencies dominate and the prediction horizon is short, a lightweight MLP with local

[*]Equal contribution [1]Department of Computer Science, McGill University [2]Department of Computer Science, MBZUAI [3]Mila. Correspondence to: Chun Jason Xue <jason.xue@mbzuai.ac.ae>.

*Accepted at Methods and Opportunities at Small Scale (MOSS), ICML 2025*, Vancouver, Canada.

inductive bias can be both sufficient and preferable. Furthermore, the modularity of our design, based on a soft routing mechanism over shared MLP experts, offers a path toward broader applicability. The expert module can act as a lightweight adapter, similar in spirit to LoRA (Hu et al., 2021), or as a differentiable routing layer in modular agents (Andreas et al., 2017; Hu et al., 2023), while retaining low compute and parameter costs. Our goal is not to propose a universal architecture, but to highlight that under small-scale and online constraints, revisiting simpler alternatives can uncover models that are more adaptive, more efficient, and in some cases, more effective than their overparameterized counterparts.

## 2. Related Work

MLP-based models have recently re-emerged as efficient alternatives to attention for sequence and vision tasks (Tolstikhin et al., 2021; Liu et al., 2021; Tay et al., 2021). In the byte-level setting, where local dependencies dominate, prior work has shown that simpler architectures can perform competitively (Press et al., 2022; Xue et al., 2022; Henighan et al., 2020). Our work follows this line, focusing on online learning under compute-constrained conditions. Unlike transformers (Vaswani et al., 2017; Tay et al., 2022), our architecture eliminates attention entirely while retaining adaptability through a soft routing mechanism over shared MLP experts. This structure is related to mixture-of-experts models (Shazeer et al., 2017; Lepikhin et al., 2021; Fedus et al., 2022) and parameter-efficient adaptation techniques such as LoRA (Hu et al., 2021) and prefix-tuning (Li & Liang, 2021), but uses a fully differentiable, continuous gating function to blend shared experts. The resulting modularity also resembles recent designs for learnable neural agents (Andreas et al., 2017; Hu et al., 2023). Our motivation builds on prior work in scalable, efficient sequence modeling (Mao et al., 2022b;a), extending this line toward lightweight, attention-free architectures that can operate in fully online, streaming environments.

## 3. Method

We propose a lightweight, fully attention-free neural architecture for byte-level next-token prediction under online training. The model, LiteByte, is composed of a stack of alternating feedforward layers, each designed to efficiently extract and propagate local patterns through dynamic and hierarchical transformations. Unlike Transformer-based models, our approach relies solely on MLP structures and avoids any form of positional encoding, attention, or recurrence.

### 3.1. Input Embedding and Reshaping

**Byte-level Embedding and Patch Reshaping.** Let $x = \{x_1, x_2, \ldots, x_T\} \in \{0, \ldots, 255\}^T$ be a stream of raw bytes, processed in mini-batches of size $B$. Each byte is first mapped to a dense vector through a learnable lookup table $\mathrm{Embed} : \{0, \ldots, 255\} \to \mathbb{R}^{d_{\text{vocab}}}$. To prevent extreme activations and to interpret each channel as a "soft one-hot" code, an element-wise sigmoid is applied:

$$\mathbf{e}_t = \sigma\big(\mathrm{Embed}(x_t)\big) \in (0,1)^{d_{\text{vocab}}}, \qquad t = 1, \ldots, T.$$

**Local patch construction.** Define an integer scaling factor

$$\text{scale} = \frac{d_{\text{hidden}}}{d_{\text{vocab}}} \in \mathbb{N}_{\geq 1},$$

which specifies how many consecutive embeddings are concatenated. For every non-overlapping window of length scale we stack the corresponding vectors and flatten them, producing a single feature of dimension $d_{\text{hidden}}$:

$$\mathbf{h}_i = \big[\mathbf{e}_{(i-1)\text{scale}+1}; \ldots; \mathbf{e}_{i\text{scale}}\big] \in \mathbb{R}^{d_{\text{hidden}}}, \qquad i = 1, \ldots, L,\ L = \frac{T}{\text{scale}}.$$

Collecting all patches across the batch yields the hidden tensor

$$\mathbf{H}_0 \in \mathbb{R}^{B \times L \times d_{\text{hidden}}},$$

where the second axis now indexes contiguous byte windows rather than individual bytes. This reshaping reduces the effective sequence length by a factor of scale, cutting the arithmetic cost of subsequent routers and feed-forward layers while preserving fine-grained locality within each patch. Setting $d_{\text{hidden}}$ to a power of two (e.g. 512 or 1024) aligns memory accesses with common hardware vector widths, further accelerating the batched matrix multiplications that follow.

### 3.2. Hierarchical Local-Global Feedforward Architecture

**Soft Shared MLP Experts.** To project each local patch into a richer feature space the model employs a *soft mixture of shared MLP experts*. Instead of allocating one weight matrix per sample, which would scale the parameter count with the mini-batch, we maintain a fixed bank of $K$ expert matrices $\{ W_k \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}} \}_{k=1}^{K}$ and biases $\{ b_k \in \mathbb{R}^{d_{\text{out}}} \}_{k=1}^{K}$.

**Gating.** For the $i$-th patch the model first forms a compact summary $x_i^{\text{summary}} = \frac{1}{N} \sum_{j=1}^{N} \mathbf{H}_0[i, j] \in \mathbb{R}^{d_{\text{in}}}$ by mean-pooling across the $N$ token rows of $\mathbf{H}_0[i] \in \mathbb{R}^{N \times d_{\text{in}}}$. A lightweight gating network $g : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{K}$ maps the summary to unnormalised logits that are converted to a simplex via softmax:

$$\alpha_i = \text{softmax}\big(g(x_i^{\text{summary}})\big) \in [0, 1]^K, \qquad \sum_{k=1}^{K} \alpha_{i,k} = 1.$$

**Expert projection.** Each expert independently transforms the entire patch:

$$\mathbf{Z}_{i,k} = \mathbf{H}_0[i]\, W_k + b_k \in \mathbb{R}^{N \times d_{\text{out}}}, \qquad k = 1, \ldots, K.$$

**Soft aggregation.** The outputs are fused by a convex combination driven by $\alpha_i$:

$$\mathbf{H}_i = \sum_{k=1}^{K} \alpha_{i,k}\, \mathbf{Z}_{i,k} \in \mathbb{R}^{N \times d_{\text{out}}}.$$

This design preserves sample-specific adaptability while keeping the total parameter count independent of the batch size. Because the same expert bank serves all patches, the effective model capacity scales with $K\, d_{\text{in}}\, d_{\text{out}}$ rather than $B\, L\, d_{\text{in}}\, d_{\text{out}}$.

**Global Feature Integration with Feedforward Layer.** Following the expert block the tensor is collapsed along the token axis, yielding a vector $v_i = \text{reshape}(\mathbf{H}_i) \in \mathbb{R}^{N d_{\text{out}}}$. A conventional two-layer feedforward network with intermediate GELU, LayerNorm preactivation, and residual averaging then provides globally conditioned processing:

$$\tilde{v}_i = v_i + \frac{1}{2}\Big(\text{GELU}\big(v_i W_1 + b_1\big)\, W_2 + b_2\Big),$$

where $W_1 \in \mathbb{R}^{N d_{\text{out}} \times d_{\text{ff}}}$ and $W_2 \in \mathbb{R}^{d_{\text{ff}} \times N d_{\text{out}}}$. The LayerNorm that precedes the linear stack ensures scale stability, while the residual *average* (rather than sum) prevents the magnitude explosion often observed in deep online learners.

The juxtaposition of *local perception* (soft experts) and *global integration* (feedforward) allows the model to learn frequent byte-level transitions and long-range semantic dependencies within the same fully feedforward architecture.

**Output Projection.** For step-wise generation the current hidden state $\mathbf{H}_t \in \mathbb{R}^{B \times 1 \times d_{\text{hidden}}}$ is passed through a linear classifier that shares no parameters with the embedding table:

$$\hat{y}_t = \mathbf{H}_t\, W_{\text{out}} + b_{\text{out}} \in \mathbb{R}^{B \times 1 \times 256}.$$

The logits $\hat{y}_t$ feed a standard cross-entropy loss with respect to the ground-truth byte at position $t$. Because no masking or attention is used the entire pipeline supports strictly online inference with latency governed only by matrix multiplications.

### 3.3. Architecture Illustration

Figure 1 shows the *individual–block* and *mix–block* as a cascade of two conceptually identical stages, each composed of **(i)** a *soft expert router* and **(ii)** a *shared feed-forward network* (FFN1→FFN2). Given a hidden vector $\mathbf{h} \in \mathbb{R}^{d_{\text{hidden}}}$, the router compares $\mathbf{h}$ with $K$ learned expert keys, assigns a probability simplex $\boldsymbol{\alpha} = \text{softmax}(g(\mathbf{h})) \in [0, 1]^K$, and produces an aggregated expert output $\mathbf{z} = \sum_{k=1}^{K} \alpha_k\, \text{Expert}_k(\mathbf{h})$. This *soft selection* (annotated by *Selected Expert* in the figure) lets the network specialise locally while sharing parameters globally, striking a balance between capacity and efficiency. The aggregated vector is then refined by a two-layer GELU →Linear module (FFN1→FFN2), with per-patch LayerNorm and *residual averaging* instead of summation to keep activation scales depth-invariant.

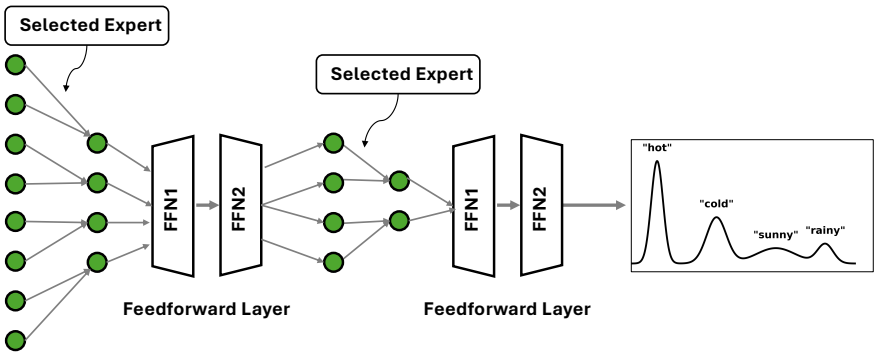


*Figure 1.* An illustration of our MLP-based online prediction pipeline. It's a two-layer architecture where each stage routes features through soft-selected experts, followed by a shared feedforward transformation. The output is a token-level distribution for next-token prediction.

Because stage 1 operates on byte-level *patches* of length scale $= d_{\text{hidden}}/d_{\text{vocab}}$, it focuses on short-range regularities, character n-grams, punctuation patterns, UTF-8 fragments, that dominate raw byte streams. Stage 2, fed by the already transformed representation, sees a lower-resolution view and therefore captures broader semantic context. Together they form an implicit hierarchy: rapid, patch-wise adaptation in the lower tier; gradual abstraction of long-range structure in the upper tier. Unlike subword LLMs, byte-level models cannot rely on pre-segmented tokens. The proposed hierarchy compensates by allowing the first stage to emulate a data-driven tokeniser (through dynamic patch transforms) while the second stage plays the role of a conventional encoder. Attention-based models often diverge when trained with continual streaming updates; they accumulate scale drift because residual sums grow with depth (Raffel et al., 2020). By replacing summation with *averaging* and performing LayerNorm within each patch, our blocks bound the variance of the hidden state irrespective of the number of updates, enabling months-long online training without manual learning-rate resets or gradient clipping. After the second FFN the hidden state $\mathbf{h}_t \in \mathbb{R}^{d_{\text{hidden}}}$ is linearly projected onto the 256-way byte vocabulary, yielding logits $\hat{\mathbf{y}}_t$. Common continuations such as "`hot`" or "`cold`" emerge with high likelihood because the expert bank allocates specialists to frequently recurring local patterns, while the shared FFNs ensure global consistency across positions.

## 4. Experiments

We evaluate LiteByte on byte-level next-token prediction across three representative datasets: **Enwik8** (Hutter, 2006), **Text8** (Mahoney), and a subset of **Charles Dickens**' collected works from Project Gutenberg (Gutenberg). All models are trained under the same configuration with a sequence length of 64, batch size of 512, and online autoregressive training. Evaluation perplexity is reported on a held-out 20% test split.

### 4.1. Comparison with Baselines

We first compare LiteByte with three commonly used sequence models: a Transformer encoder (8 layers, 4 heads), a 8-layer GRU-based RNN, and a standard 8-layer MLP. All models are matched in parameter count and trained with the same optimizer and learning rate. Figure **??** shows the training loss curves on the three datasets. LiteByte achieves faster convergence than all baselines, especially in the early stage of training. Notably, Transformer training is slower and less stable under the same online setting, likely due to its dependency on attention mechanisms and positional encodings.

### 4.2. Ablation Study: Soft Expert Routing

To evaluate the contribution of soft expert routing, we compare the full LiteByte model with an ablated variant where the dynamic expert mixture is removed and replaced by a single shared projection. Figure 2 shows the result. The absence of soft expert routing leads to consistently higher loss and slower convergence, indicating that dynamic routing plays a critical role in LiteByte's adaptability, especially in the early phases of online training.

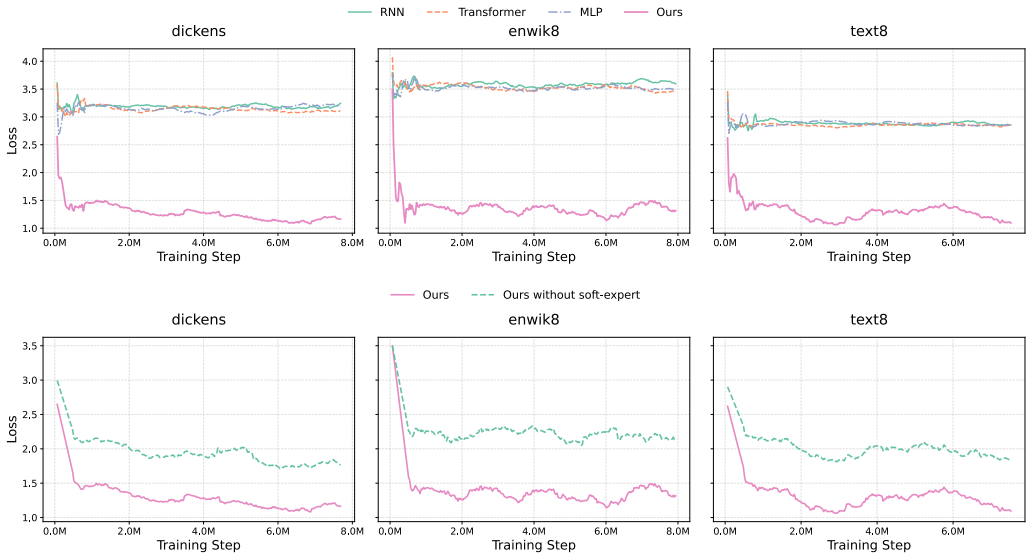


*Figure 2.* Training loss across three datasets (Dickens, Enwik8, and Text8). The top row compares LiteByte to baseline models including Transformer, RNN, and MLP, showing that LiteByte converges faster and achieves lower loss in all cases. The bottom row presents an ablation study, comparing the full model with a variant that removes soft expert routing. The expert mechanism leads to improved convergence speed and final performance.

### 4.3. Efficiency Analysis

To assess the runtime efficiency of LiteByte, we benchmark its inference latency and throughput against three baseline models: TinyMLP, TinyRNN, and TinyTransformer. All models are evaluated on the same hardware (NVIDIA 4090 GPU). As shown in Table 1, LiteByte achieves the lowest latency (0.0073s per input) and the highest throughput (137.5 items/sec), outperforming all baselines by a significant margin. These improvements stem from its fully feedforward design and batch-aware expert routing, which enable both fast computation and efficient GPU utilization.

| Model | Throughput (tokens/sec) | GPU Usage (GB) |
|---|---|---|
| TinyMLP | 7127.04 | 10.97 |
| TinyRNN | 43694.08 | 2.7 |
| TinyTransformer | 3921.92 | 13.3 |
| LiteByte (ours) | **140800** | **0.97** |

*Table 1.* Inference latency, throughput, and peak GPU memory usage for all models under identical evaluation conditions. LiteByte achieves the best runtime performance.

## 5. Conclusion

We introduced a lightweight, attention-free architecture for byte-level next-token prediction, composed entirely of multilayer perceptrons with alternating structure and soft expert routing. Despite its simplicity, our model achieves competitive or superior performance compared to Transformer, RNN, and MLP baselines, while offering significant improvements in memory efficiency and training speed. Our design is particularly well-suited for low-resource and online settings: it avoids tokenization, operates at byte granularity, and adapts quickly through online updates. Importantly, the model runs efficiently on resource-constrained environments such as Colab Free Tier GPUs, making it highly deployable in edge or streaming scenarios. We hope this work encourages further exploration of non-attentional architectures in small-scale and adaptive learning regimes, where traditional Transformers may be unnecessarily complex.

## References


Andreas, J., Klein, D., and Levine, S. Modular multitask reinforcement learning with policy sketches. In *ICML*, 2017.

Brown, T. B., Mann, B., Ryder, N., and et al. Language models are few-shot learners. *NeurIPS*, 2020.

Chen, M., Tworek, J., Jun, H., and et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Fedus, W., Zoph, B., and Shazeer, N. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Gutenberg, P. The Works of Charles Dickens (plain text). https://www.gutenberg.org/. Curated subset used in this work; Accessed: 2025-05-25.

Henighan, T., Kaplan, J., and et al. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*, 2020.

Hu, E. J., Shen, Y., Wallis, P., and et al. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Hu, E. J., Lee, K., Chowdhery, A., Lester, B., Narasimhan, K., Dean, J., Roberts, A., and Raffel, C. Transformer agents: Interactive decision-making with language models. *arXiv preprint arXiv:2305.01680*, 2023.

Hutter, M. The Human Knowledge Compression Contest (enwik8). http://prize.hutter1.net/, 2006. Accessed: 2025-05-25.

Lepikhin, D., Lee, H., Xu, Y., Chen, D., Firat, O., Johnson, M., Macherey, W., Krikun, M., Chen, N., Zhou, Y., et al. Gshard: Scaling giant models with conditional computation and automatic sharding. In *ICLR*, 2021.

Li, X. and Liang, P. Prefix-tuning: Optimizing continuous prompts for generation. *ACL*, 2021.

Liu, H., Dai, Z., So, D. R., and Le, Q. V. Pay attention to mlps. *NeurIPS*, 34:9204–9215, 2021.

Mahoney, M. Text8: A compressed corpus for NLP. http://mattmahoney.net/dc/textdata. Accessed: 2025-05-25.

Mao, Y., Cui, Y., Kuo, T.-W., and Xue, C.-J. Accelerating general-purpose lossless compression via simple and scalable parameterization. In *Proceedings of the 30th ACM International Conference on Multimedia*, pp. 3205–3213, 2022a.

Mao, Y., Cui, Y., Kuo, T.-W., and Xue, C.-J. Trace: A fast transformer-based general-purpose lossless compressor. In *Proceedings of the ACM Web Conference 2022*, pp. 1829–1838, 2022b.

Press, O., Smith, N. A., and Levy, O. Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2202.07765*, 2022.

Radford, A., Kim, J. W., Hallacy, L., and et al. Learning transferable visual models from natural language supervision. *ICML*, 2021.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *ICLR*, 2017.

Tay, Y., Dehghani, M., Bahri, D., and Metzler, D. A simple and effective architecture for modeling long-range dependencies. *arXiv preprint arXiv:2005.10862*, 2021.

Tay, Y., Dehghani, M., Bahri, D., and Metzler, D. Efficient transformers: A survey. *ACM Computing Surveys*, 2022.

Tolstikhin, I., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Keysers, D., Uszkoreit, J., Lucic, M., et al. Mlp-mixer: An all-mlp architecture for vision. *NeurIPS*, 34:24261–24272, 2021.

Touvron, H., Lavril, T., Izacard, G., and et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Vaswani, A., Shazeer, N., Parmar, N., and et al. Attention is all you need. In *NeurIPS*, 2017.

Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., Barham, P., and Raffel, C. Byt5: Towards a token-free future with pre-trained byte-to-byte models. In *Proceedings of ACL*, 2022.