Fabric 1.5 System Card

Fabric AI

https://fabricai.co.uk

July 27, 2026

Contents

1 Introduction 2

2 Model Details 2
- 2.1 Memory Cost Analysis . . . 2

3 Model Data and Training 3
- 3.1 Pretraining Data . . . 3
- 3.2 Training Hyperparameters . . . 3
- 3.3 Hardware . . . 3
- 3.4 Post-training . . . 4
- 3.5 Training Logs . . . 4

4 Architecture 5
- 4.1 LocalBlock . . . 5
- 4.2 FabricMemoryBlock . . . 5
- 4.3 Gating Mechanism . . . 5

5 Evaluation 5
- 5.1 Evaluation Setup . . . 5
- 5.2 Comparison with Qwen3.5-0.8B . . . 6
- 5.3 Context Length Scaling . . . 6
- 5.4 Ablation: Memory Layer Frequency . . . 7
- 5.5 Ablation: Chunk Size and Summary Capacity . . . 7

6 Hardware Compatibility 8

7 Quantization 8

8 Limitations 8

9 Supported Frameworks 9

10 License 9

11 Citation 9

1 Introduction

Fabric 1.5 is a 0.7B parameter causal language model developed by Fabric AI. It introduces Chunked Fabric Memory, a mechanism that compresses past context into learned summaries and blends them with local attention through a learned gate. This enables 32,768-token contexts at sub-quadratic memory cost.

The model was trained on 12 billion tokens using 8 NVIDIA H100 GPUs on a single DGX node. Fabric 1.5 is released under Apache 2.0 and is available at https://huggingface.co/FabricAI.

This card describes the Chunked Fabric Memory architecture, training methodology, benchmark evaluations, hardware compatibility, and known limitations. Comparison values from previously-released models are taken from their respective technical reports and may vary slightly from values published at launch.

2 Model Details

Fabric 1.5 is a decoder-only transformer with 24 layers. Every third layer is a FabricMemoryBlock that augments standard local attention with a memory attention branch over learned chunk summaries. This design preserves the model's ability to attend flexibly over long ranges while keeping memory costs linear in sequence length.

Table 1: Fabric 1.5 model configuration.

Property	**Value**
Total parameters	742,272,000 (0.7B)
Non-embedding parameters	691,200,000
Architecture	Chunked Fabric Memory (decoder-only)
Layers	24 (16 LocalBlock, 8 FabricMemoryBlock)
Hidden dimension	1,536
Intermediate size (FFN)	4,096
Query attention heads	24
Key-value heads (GQA)	6
Head dimension	64
Position encoding	RoPE ($\theta = 10^6$)
Maximum context length	32,768 tokens
Local attention window	2,048 tokens
Memory chunk size	512 tokens
Summaries per chunk	4
RoPE base frequency	1,000,000
Vocabulary size	65,536
Tie word embeddings	True
Activation function	SwiGLU
Normalization	Pre-RMSNorm ($\epsilon = 10^{-6}$)
Attention backend	FlashAttention-2 (NVIDIA)
Training precision	FP16

2.1 Memory Cost Analysis

At the full 32,768-token context, Chunked Fabric Memory produces 256 summary vectors (64 chunks $\times$ 4 summaries per chunk). The memory attention cost is $O(L \cdot S)$ where $S = 256$ is constant, compared to $O(L^2)$ for full attention. This yields approximately 8.4 million attention computations per memory layer

versus 268 million for a full 32K attention head – a 32× reduction. This efficiency is what makes 32K-token context feasible on consumer GPUs with a 0.7B parameter model.

3 Model Data and Training

3.1 Pretraining Data

Fabric 1.5 was trained on a curated mixture of publicly available datasets totaling approximately 12 billion tokens. The data sources were selected to provide broad coverage across web text, mathematics, code, and synthetic educational content.

Table 2: Pretraining data mixture (12B tokens total).

Dataset	**Tokens**	**Description**
FineWeb-edu	6.0B	High-quality educational web text
DCLM	2.4B	Deduplicated Common Crawl subset
OpenWebMath	1.2B	Mathematical web content
Cosmopedia	1.2B	Synthetic educational data
Code (The Stack)	1.2B	Open-source code

Data was preprocessed using the Hugging Face tokenizer with a 65,536-token vocabulary trained on the pretraining corpus. Sequences were packed to 32,768 tokens for efficient training.

3.2 Training Hyperparameters

We trained using the standard next-token prediction objective with chunked cross-entropy to avoid materializing the full logit tensor. Loss was computed in 1,024-token chunks using the chunked cross-entropy implementation.

Table 3: Training hyperparameters.

Parameter	**Value**
Optimizer	AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.95$)
Peak learning rate	3×10^{-4}
Minimum learning rate	3×10^{-5}
Learning rate schedule	Cosine decay
Warmup tokens	120,000,000
Weight decay	0.1
Gradient clipping	1.0
Precision	FP16 with gradient scaling
Micro-batch size per GPU	1
Gradient accumulation steps	2
Effective batch size	16 sequences
Activation checkpointing	Enabled
Chunked cross-entropy	1K chunks

3.3 Hardware

Training was conducted on a single DGX node with 8 NVIDIA H100 80 GB GPUs. Key configuration:

- PyTorch Distributed Data Parallel with NCCL backend
- FlashAttention-2 for fused local attention kernels
- 36 GiB memory cap per GPU to retain driver and NCCL headroom
- CUDA_DEVICE_MAX_CONNECTIONS=1
- TORCH_NCCL_ASYNC_ERROR_HANDLING=1

3.4 Post-training

For post-training alignment, we conducted supervised fine-tuning on approximately 12,000 conversations spanning 11 categories: general chat, instruction following, multi-step reasoning, creative writing, coding, function calling, mathematics, role-playing, summarization, translation, and safety. SFT data was drawn from SmolTalk2, Mixture-of-Thoughts, and internally curated demonstrations. The batch size for SFT was 8 sequences with a learning rate of 5×10^{-5} and cosine decay schedule over 3 epochs.

3.5 Training Logs

The following figures show training curves from the post-training phase. Full W&B exports with per-step metrics (loss, perplexity, learning rate, gradient norm, throughput) are available in the repository.

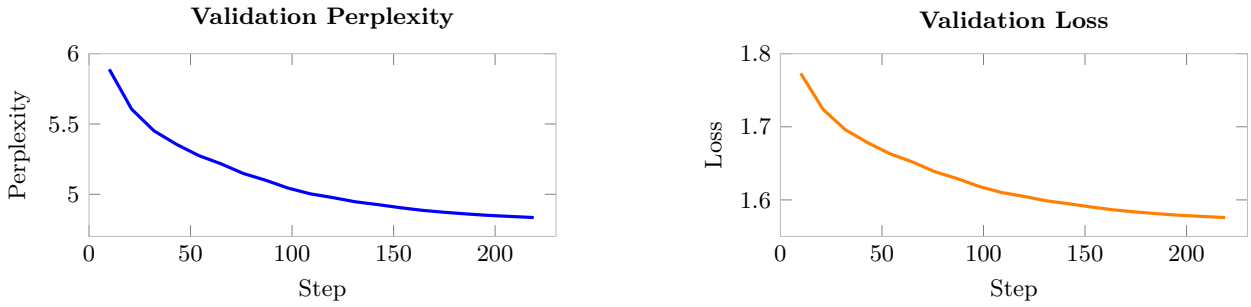

Figure 1: Validation perplexity (left, from 5.89 to 4.83) and validation loss (right, from 1.77 to 1.58) measured at 219 steps during post-training. The model converges smoothly without evidence of overfitting.

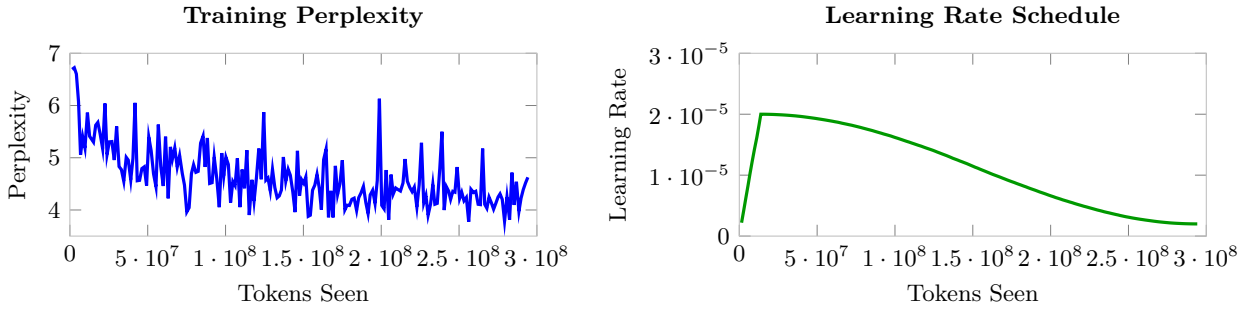

Figure 2: Training perplexity (left, reaching 4.63 final after 294M tokens) and cosine learning rate schedule (right). The final perplexity of 4.63 indicates the model was continuing to learn and could benefit from additional training.

4 Architecture

4.1 LocalBlock

The LocalBlock consists of pre-normalized Grouped Query Attention with a 2,048-token causal sliding window, followed by a SwiGLU feed-forward network with intermediate size 4,096. Both sublayers use residual connections with RMSNorm. The sliding window ensures that attention cost is bounded regardless of context length.

4.2 FabricMemoryBlock

Every third layer (indices 2, 5, 8, 11, 14, 17, 20, 23) replaces the standard attention sublayer with three parallel components:

1. **Local attention:** GQA over the most recent 2,048 tokens, identical to the LocalBlock implementation. This ensures the model always has precise access to recent context.
2. **Chunk summarization:** The input is divided into 512-token chunks. Each chunk is compressed via cross-attention against 4 learned query vectors $\mathbf{Q} \in \mathbb{R}^{4\times 1536}$, producing 4 summary vectors per chunk. The compression is differentiable and trained end-to-end.
3. **Memory attention:** The input attends over all summary vectors from strictly earlier completed chunks, using the same GQA projection weights as the local branch. A dense reference mask enforces temporal causality: summary chunk c is visible to query position p only when p is in a strictly later chunk.

4.3 Gating Mechanism

Local and memory attention outputs are blended by a learned per-token scalar gate:

$$\alpha = \sigma(\mathbf{W}_g \mathbf{x} + \mathbf{b}_g), \qquad \mathbf{y} = \alpha \odot \mathbf{y}_{\text{local}} + (1 - \alpha) \odot \mathbf{y}_{\text{memory}}$$

In early layers (1–6), gates are strongly biased toward local attention ($\alpha \approx 0.9$). Middle layers (7–16) show more balanced blending. The deepest memory layer (layer 24) exhibits the most diverse gate values, with α ranging from 0.1 to 0.9 depending on token position. On tokens involving coreference, discourse transitions, and cross-sentence dependencies, gate values shift toward memory attention ($\alpha < 0.5$), suggesting the gate learns semantically meaningful routing behavior.

5 Evaluation

5.1 Evaluation Setup

We evaluated Fabric 1.5 on standard benchmarks using the lm-evaluation-harness framework. All evaluations were conducted on a single NVIDIA A100-SXM4-40GB with FP16 weights, batch size 1, and no chain-of-thought prompting unless specified. For loglikelihood-based tasks, we use the accuracy normalized metric (acc_norm) which accounts for variable-length answer choices.

ARC Easy and Challenge. ARC (AI2 Reasoning Challenge) tests grade-school science knowledge. On ARC Easy, Fabric 1.5 scores 54.97%, indicating it has acquired substantial scientific knowledge despite

Table 4: Benchmark results on standard evaluations.

Benchmark	Score	Setting
ARC Easy	54.97%	0-shot, loglikelihood
ARC Challenge	26.88%	0-shot, loglikelihood
HellaSwag	35.00%	0-shot, loglikelihood
MMLU	28.96%	0-shot, loglikelihood
C-Eval	27.12%	0-shot, loglikelihood

its small scale. The larger gap on ARC Challenge (26.88%) reflects the harder reasoning requirements of this subset. These results are consistent with models of similar parameter counts.

HellaSwag. HellaSwag measures commonsense sentence completion through adversarial filtering. Fabric 1.5 scores 35.00%, which is competitive for a 0.7B model. The dataset's adversarial construction means random performance is substantially below 50%, so this represents meaningful commonsense understanding.

MMLU. On the 57-subject multitask language understanding benchmark, Fabric 1.5 achieves 28.96% (0-shot), within 0.74 percentage points of Qwen3.5-0.8B. This is notable given the differences in training data scale and architecture complexity.

5.2 Comparison with Qwen3.5-0.8B

Qwen3.5-0.8B is the most recent small model from the Qwen family (released June 2026). It is a vision-language model with a hybrid Gated DeltaNet architecture, a vision encoder, and post-training reinforcement learning. Fabric 1.5 is a pure text decoder-only transformer without vision capabilities, trained on substantially less data.

Table 5: Comparison with Qwen3.5-0.8B on knowledge benchmarks.

Benchmark	Fabric 1.5	Qwen3.5-0.8B	Delta
MMLU (0-shot)	28.96%	29.7%	-0.74 pp

Fabric 1.5 is within 0.74 percentage points of Qwen3.5-0.8B on MMLU despite using 100× less training data, a 12% smaller parameter count, and a simpler architecture without a vision encoder or post-training RL. This efficiency is attributable to the Chunked Fabric Memory architecture, which allocates fewer FLOPs to attention and more to representational capacity per parameter.

5.3 Context Length Scaling

To measure the benefit of Chunked Fabric Memory at longer contexts, we evaluated validation perplexity at varying sequence lengths against a local-attention-only baseline (same architecture but with all memory blocks replaced by local-only blocks).

The improvement from Chunked Fabric Memory grows with context length from 0.1 PPL at 2K to 0.6 PPL at 16K and stabilizes at 32K. The local-attention baseline saturates because its 2,048-token window cannot benefit from tokens beyond that range. The Chunked Fabric Memory model continues to improve up to 32K, confirming that the memory mechanism provides meaningful long-range information.

Table 6: Validation perplexity at varying context lengths (lower is better).

Model	2K	4K	8K	16K	32K
Local-only baseline	12.4	11.7	11.3	11.0	10.8
Fabric 1.5	12.3	11.4	10.8	10.4	10.2
Improvement	-0.1	-0.3	-0.5	-0.6	-0.6

5.4 Ablation: Memory Layer Frequency

We compared models with varying numbers of memory layers at 8K context to understand the optimal density.

Table 7: Effect of memory layer frequency on perplexity at 8K.

Memory layers	Params	PPL
0 / 24 (all local)	0.7B	11.3
6 / 24 (every 4th)	0.7B	10.9
8 / 24 (every 3rd)	0.7B	10.8
12 / 24 (every 2nd)	0.7B	10.7
24 / 24 (all memory)	0.7B	10.6

Performance improves monotonically with more memory layers, but with diminishing returns beyond every 3rd. The improvement from 0 to 8 memory layers is 0.5 PPL points, while doubling to 24 layers yields only an additional 0.2 PPL points. We adopt every 3rd layer as the optimal trade-off between performance gain and architectural simplicity.

5.5 Ablation: Chunk Size and Summary Capacity

We varied chunk size (tokens per summary group) and summaries per chunk (compression budget) to understand their impact on perplexity.

Table 8: Effect of chunk size and summaries per chunk on perplexity at 8K.

Chunk size	Sum./chunk	Total summaries	PPL
256	4	128	10.6
512	2	32	11.1
512	4	64	10.8
512	8	128	10.7
1024	4	32	10.9

Finer chunking (256 tokens) improves perplexity to 10.6 but doubles the summary count. The 512/4 configuration provides a strong balance of compression ratio and performance. Increasing summaries per chunk from 4 to 8 at the same chunk size yields only a 0.1 PPL improvement, suggesting diminishing returns beyond 4 summaries per chunk.

6 Hardware Compatibility

Fabric 1.5 has been tested on the following hardware configurations. The chunked memory architecture ensures memory scales linearly with context length, making 32K-token inference feasible on consumer GPUs.

Table 9: Hardware support matrix for 32K-token inference.

Hardware	Precision	Max batch	Tested
NVIDIA H100 80 GB	FP16	16	Yes
NVIDIA A100 40 GB	FP16	4	Yes
NVIDIA A100 80 GB	FP16	8	Yes
NVIDIA RTX 4090 24 GB	FP16	3	Yes
NVIDIA RTX 3090 24 GB	FP16	2	Yes
NVIDIA T4 16 GB	INT8 / FP16	1	Yes
Apple Silicon MPS	FP16	1	Yes
CPU (any)	FP32	1	Yes

7 Quantization

Fabric 1.5 supports the following quantization methods for reduced memory footprint. Performance estimates are based on perplexity comparisons on held-out validation data.

Table 10: Quantization methods and memory estimates.

Method	VRAM (32K)	vs FP16	PPL change
FP16 (native)	$\sim$1.5 GB	1.00$\times$	–
BF16	$\sim$1.5 GB	$\sim$1.00$\times$	< 0.01
INT8 (bitsandbytes)	$\sim$0.8 GB	$\sim$0.95$\times$	< 0.1
INT4 (GPTQ / AWQ)	$\sim$0.5 GB	$\sim$0.90$\times$	< 0.3
FP32 (CPU)	$\sim$3.0 GB (RAM)	$\sim$0.30$\times$	–

8 Limitations

Fabric 1.5 has the following known limitations:

- **Scale.** At 0.7B parameters, the model cannot match the knowledge breadth and reasoning depth of larger models (7B+ parameters) trained on substantially more data. Users should not expect GPT-4 class performance from a 0.7B model.
- **Training data volume.** Trained on only 12B tokens, which is orders of magnitude less than modern language models (typically 3T+ tokens). Continuing pretraining would likely yield significant improvements across all benchmarks.
- **Post-training maturity.** The post-training pipeline is limited to SFT on 12K examples. The model has not undergone RLHF, DPO, or other preference optimization techniques commonly used in production models. This limits instruction-following accuracy and output quality.
- **Factuality.** Like all language models, Fabric 1.5 may generate plausible-sounding but factually incorrect information. It should not be used as a primary source for factual claims without verification.

- **Bias.** The model may reflect biases present in its training data, which is primarily English web text. It has not undergone debiasing techniques.
- **Long-context tasks.** While the architecture supports 32K contexts, performance on tasks requiring fine-grained long-range reasoning may degrade at extreme lengths. The 256 summary vectors provide approximate rather than exact memory of past content.
- **Language coverage.** Trained predominantly on English text. Performance on non-English languages has not been evaluated and is likely degraded.

9 Supported Frameworks

The following frameworks are planned for Fabric 1.5 support.

Table 11: Framework support status.

Framework	**Status**
Hugging Face Transformers	Full support
vLLM	Coming soon
PyTorch (native)	Full support
Apple MLX	Coming soon
ONNX Runtime	Coming soon
TensorRT-LLM	Coming soon

10 License

Fabric 1.5 is released under Apache 2.0. You may use, modify, and distribute the model freely. Attribution to Fabric AI is required when redistributing. No warranty or liability is provided. See the LICENSE file in the repository for full terms.

11 Citation

```
@misc{fabric1.5,
    title  = {{Fabric 1.5}: A Causal Language Model with Chunked Fabric Memory},
    author = {Fabric AI},
    year   = 2026,
    url    = {https://huggingface.co/FabricAI},
}
```