REINFORCED FAST WEIGHTS WITH NEXT-SEQUENCE PREDICTION

Hee Seung Hwang* **Xindi Wu*** **Sanghyuk Chun** **Olga Russakovsky**

Princeton University

ABSTRACT

Fast weight architectures offer a promising alternative to attention-based transformers in long-context settings, but their potential is limited by the next-token prediction (NTP) training paradigm. The NTP objective ignores semantic relations of multiple tokens that follow a prefix by optimizing for single-token predictions. As a result, fast weight models, which dynamically update their parameters to store contextual information, learn suboptimal mappings of token representations. We introduce REFINE (**Re**inforced **F**ast we**I**ghts with **N**ext s**E**quence prediction), a method that leverages reinforcement learning to train fast weight models under the next-sequence prediction (NSP) objective. REFINE selects informative token positions with high entropy, generates multi-token rollouts, assigns self-supervised sequence-level rewards, and optimizes the model with GRPO. Our experiments on the LaCT-760M and DeltaNet-1.3B models show that REFINE consistently outperforms supervised fine-tuning under NTP in needle-in-a-haystack tasks, long-context QA, and various subtasks in LongBench. REFINE thus provides an effective and versatile solution for improving long-context modeling of fast weight architectures.

1 INTRODUCTION

While attention-based transformers have demonstrated strong performance on various language tasks, their computational and memory costs grow quadratically Keles et al. (2023) with context length. This fundamental scaling problem has become a bottleneck for both training and inference, particularly for long contexts. Fast weight architectures offer a promising alternative to attention-based transformers in long context settings by replacing global attention with a fixed-size memory that is dynamically updated as new tokens are processed, allowing contextual information to be stored directly in the model weights. This design enables efficient inference with constant memory overhead even for long contexts Tandon et al. (2025).

Despite their architectural differences, fast weight models are typically pre-trained with the same next-token prediction (NTP) objective used for standard transformer LLMs Sun et al. (2024); Behrouz et al. (2024; 2025a); Yang et al. (2024; 2025); Zhang et al. (2025). In this work, we argue that NTP is a *suboptimal* objective for fast weight models. The NTP objective only has an immediate effect on the next token and disregards the quality of subsequent predictions that depend on the same internal state Gloeckle et al. (2024). As a result, NTP's token-level feedback encourages parameter updates that optimize only short-term likelihood, limiting the adaptive capacity of fast weights and model behavior over longer horizons.

To better align the training objective with the intended function of fast weights as long-context memory, we propose the **next-sequence prediction (NSP)** objective as a variation of NTP. NSP encourages a model to predict a semantically coherent sequence of future tokens conditioned on a given prefix. We formulate NSP as a reinforcement learning (RL) problem and propose REFINE (Reinforced Fast Weights with Next Sequence Prediction), a framework can be applied in multiple stages of the language model training lifecycle. Our experiments show that REFINE improves fast weight behavior during mid-,post-, and test-time-training, highlighting its flexibility and practicality in improving long-context modeling of fast weight architectures.

*Equal contribution

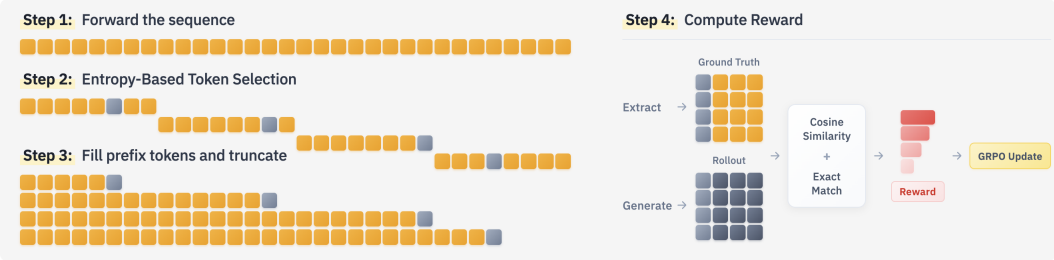

Figure 1: **REFINE.** We forward the sequence through the policy model and compute token-level entropy values. Sequences are split into chunks and a target token position is sampled from each chunk based on the entropy (**Entropy-Based Token Selection**). Prefixes are copied from the original sequence up to each target token. The policy model predicts continuations from the prefixes (**Rollout Generation**). Reward is computed based on the generated rollouts and ground truth tokens (**Reward Assignment**). Finally, we update the policy model with GRPO (**Optimization with RL**).

2 METHOD

Our REFINE framework (Fig. 1) consists of four key steps: (1) entropy-based token selection, (2) rollout generation, (3) reward assignment, and (4) optimization with RL.

2.1 REFINE

Entropy-Based Token Selection. Given an input sequence $S = (x_1, \ldots, x_T)$, we forward S through the policy model π_θ and compute the NTP entropy values H_t at each token position t. We smooth the entropy distribution within S with a 1-D average pooling with kernel size k. We partition the input sequence S into c contiguous chunks of equal length: $S = (S_1, S_2, \ldots, S_c)$. For each chunk S_i, we sample one token position with probability proportional to the softmax of its entropy, yielding a set of sampled positions $\mathcal{T}^* = \{t_1, \ldots, t_c\}$.

Rollout Generation. For each high-entropy position $t_i \in \mathcal{T}^*$, we construct a truncated prefix $x_{\leq t_i}$, yielding c distinct partial sequences $\{x_{\leq t_1}, \ldots, x_{\leq t_c}\}$ from S. These prefixes capture the full context leading up to each high-entropy position that is sampled. From each truncated prefix $x_{\leq t_i}$, we generate a k-token continuation $\hat{x}_{t_i+1:t_i+k}$ using the current policy and extract the hidden states of the final layer before the logits. We also extract the hidden states of the ground-truth continuation $x_{t_i+1:t_i+k}$ from the initial forward pass:

$$\mathbf{h}_k^{\text{pred}}(t_i) = \big(\mathbf{h}^{\text{pred}}(t_i+1), \ldots, \mathbf{h}^{\text{pred}}(t_i+k)\big) \qquad \mathbf{h}_k^{\text{gt}}(t_i) = \big(\mathbf{h}^{\text{gt}}(t_i+1), \ldots, \mathbf{h}^{\text{gt}}(t_i+k)\big). \quad (1)$$

Reward Assignment. Given the hidden states of predicted and ground truth continuations $\mathbf{h}_k^{\text{pred}}(t_i), \mathbf{h}_k^{\text{gt}}(t_i) \in \mathbb{R}^{k \times d}$, we assign a smooth similarity reward for an arbitrary similarity function φ defined as:

$$R_k^{\varphi}(t_i) = \frac{1}{k} \sum_{j=1}^{k} \varphi\big(\mathbf{h}^{\text{pred}}(t_i+j), \mathbf{h}^{\text{gt}}(t_i+j)\big). \quad (2)$$

We use cosine similarity for φ. This reward encourages the model to produce hidden representations that align with those induced by the ground-truth token. It assigns smooth, non-zero rewards to semantically similar tokens that lead to hidden state embeddings that are closer in the latent space.

Optimization with RL. Once we compute the reward for each rollout, we have a set of rollouts $o_{\mathcal{T}^*} = \{\hat{x}_{t_1+1:t_1+k}, \ldots, \hat{x}_{t_c+1:t_c+k}\}$ and corresponding rewards $\mathcal{R}_{\mathcal{T}^*} = \{R_k^{\varphi}(t_1), \ldots, R_k^{\varphi}(t_c)\}$. The rewards from the same sequence S are standardized to compute the advantage following Shao et al. (2024). We employ the GRPO algorithm (Shao et al., 2024) to compute the NSP loss based on the rollouts and their relative advantages. The policy gradients therefore maximize the following:

$$\mathcal{J}(\theta) = \mathbb{E}_{x_{\leq t} \sim \mathcal{D},\, \hat{x}_{t+1:t+k} \sim \pi_{\theta_{\text{old}}}(\cdot \mid x_{\leq t})}[R_k^{\varphi}(t)], \quad (3)$$

where $\mathcal{D}$ is the set of all $\{x_{\leq t}\}_{t=1}^{T}$. To prevent catastrophic forgetting, the final loss is a weighted sum of the NSP loss and the standard NTP loss (computed over the entire sequence S), with coefficients λ_{RL} and λ_{SFT} respectively. The weights are adjusted based on the training phase.

Table 1: **Performance on Long-Context Retrieval Tasks.** We evaluate mid-trained (**MidTr**) models on the NIAH tasks in RULER at 4K, 8K, and 16K context lengths (standard SFT vs. REFINE). Highest scores in each category are highlighted in **bold**.

	RULER S-NIAH				RULER MK-NIAH				RULER MQ-NIAH				RULER MV-NIAH			
	4K	8K	16K	Avg	4K	8K	16K	Avg	4K	8K	16K	Avg	4K	8K	16K	Avg
LaCT-760M	98.8	91.2	95.8	95.3	70.2	44.8	24.2	46.2	39.2	24.9	17.4	27.1	40.6	27.0	17.7	28.4
+ SFT MidTr	98.4	90.8	**97.6**	95.6	**70.6**	45.0	24.2	46.6	**40.7**	25.2	17.4	27.8	**41.7**	26.3	17.7	28.5
+ REFINE MidTr	**99.0**	**93.0**	96.8	**96.3**	70.4	**46.0**	**26.6**	**47.7**	40.5	**25.8**	**18.0**	**28.1**	41.2	**29.5**	**18.6**	**29.8**
DeltaNet-1.3B	**100.0**	**100.0**	**100.0**	**100.0**	23.6	17.8	3.4	14.9	27.7	3.9	1.8	11.1	23.9	5.4	2.0	10.4
+ SFT MidTr	**100.0**	**100.0**	**100.0**	**100.0**	23.8	19.2	7.8	16.9	33.2	16.8	3.5	17.8	28.3	19.3	3.7	17.1
+ REFINE MidTr	**100.0**	**100.0**	99.8	99.9	**25.0**	**21.4**	**8.8**	**18.4**	**37.3**	**18.8**	**3.6**	**19.9**	**29.0**	**20.4**	**4.0**	**17.8**

Table 2: **Performance on Multi-Doc QA Tasks.** We evaluate various training strategies during mid-training (**MidTr**), post-training (**PostTr**), and test-time training (**TTT**) for multi-document question and answering tasks. For Nested SFT and Nested REFINE, we train the model with the training method on the prompt portion of the sample as described in §3.2. Higher is better. First and second highest scores on each task are highlighted in **bold** and underline, respectively.

	MidTr	PostTr	TTT	RULER SQuADQA				RULER HotpotQA			
				4K	8K	16K	Avg	4K	8K	16K	Avg
LaCT 760M	-	-	-	17.5	14.0	6.5	12.7	20.5	14.5	12.0	15.7
	SFT	-	-	19.5	14.0	6.0	13.2	21.5	15.0	12.0	16.2
	REFINE	-	-	20.5	15.5	8.0	14.7	23.0	16.0	12.5	17.2
	REFINE	SFT	-	30.5	20.0	7.5	19.3	25.0	16.5	11.5	17.7
	REFINE	Nested SFT	-	38.0	20.0	7.5	21.8	24.0	16.0	12.5	17.5
	REFINE	Nested REFINE	-	43.5	24.5	8.5	25.5	27.0	19.5	13.0	19.8
	REFINE	Nested REFINE	SFT	43.5	21.5	8.5	24.5	26.5	24.0	13.0	21.2
	REFINE	Nested REFINE	REFINE	**45.5**	**25.5**	**10.0**	**27.0**	**28.5**	**25.5**	**15.0**	**23.0**
DeltaNet 1.3B	-	-	-	11.0	5.0	2.0	6.0	10.0	3.0	2.5	5.2
	SFT	-	-	9.0	6.0	3.0	6.0	9.0	8.0	4.5	7.2
	REFINE	-	-	10.0	6.5	4.0	6.8	12.0	9.5	5.5	9.0
	REFINE	SFT	-	11.0	7.0	5.0	7.7	14.0	11.0	6.5	10.5
	REFINE	Nested SFT	-	12.0	8.0	5.0	8.3	16.5	10.0	7.5	11.3
	REFINE	Nested REFINE	-	14.0	10.5	6.5	10.3	15.0	11.0	**8.5**	11.5
	REFINE	Nested REFINE	SFT	16.5	10.5	6.5	11.2	16.0	10.5	6.5	11.0
	REFINE	Nested REFINE	REFINE	**17.5**	**12.5**	**7.0**	**12.3**	**18.0**	**13.5**	8.0	**13.2**

2.2 HYBRID REWARD FOR RL

TTT introduces unique constraints. First, evaluations are usually conducted in the low-data regime, which leads to smaller batch sizes and limited room for meta-adaptation across episodes. Second, effective memorization of the given context becomes more important than contextual generalization. For scenarios that require stronger context memorization (*e.g.*, TTT), we introduce a binary exact match reward R^{binary} defined as:

$$R_k^{\text{binary}}(t_i) = \frac{1}{k}\sum_{j=1}^{k}\mathcal{I}[x_{t+j} = \hat{x}_{t+j}]. \tag{4}$$

For post-training, we add R^{φ} and R^{binary} to balance contextual generalization and memorization.

3 EXPERIMENTS

3.1 IMPACT OF REFINE ON MID-TRAINING

We mid-train the pre-trained LaCT-760M Zhang et al. (2025) and DeltaNet-1.3B Yang et al. (2024) checkpoints on Long-Data-Collections TogetherAI (2024) for 100 steps using a batch size of 128 (≈200M training tokens). We compare with the pure SFT baseline trained under identical conditions.

We evaluate the mid-trained models on four NIAH tasks in RULER Hsieh et al. (2024) (Single NIAH, Multi-key NIAH, Multi-query NIAH, and Multi-value NIAH) at 4K, 8K, and 16K context lengths using Language Model Evaluation Harness Gao et al. (2024).

Table 1 shows that the mid-trained model by REFINE consistently outperforms the original pre-trained model and the SFT mid-trained model on different tasks and models. For example, REFINE significantly improves DeltaNet in the Multi-key NIAH (+23.5% from no mid-training and +8.8% from SFT mid-training). This suggests that REFINE leads to improvements in long context retrieval.

Table 3: **Performance on Long-Context Tasks in LongBench.** We study the impact of the learning algorithm during mid-training (**MidTr**) and test-time training (**TTT**) on tasks with long-context. **SFT** denotes the supervised fine-tuning with next-token prediction. We evaluate on 12 tasks in LongBench, filtered for samples with at most 16K tokens. Details are similar to Table 2.

			Single-doc QA			Multi-doc QA		Summarization		Few-shot QA			Coding		
	MidTr	TTT	NQ	QR	MF	HP	2W	QM	MN	SS	TC	TQ	LC	RP	Avg
LaCT 760M	-	-	6.5	10.5	7.2	11.7	9.8	13.6	9.2	14.2	10.5	8.0	26.7	29.7	13.1
	SFT	-	5.8	10.1	7.4	12.6	9.2	13.1	10.5	13.8	7.5	12.2	29.8	30.1	13.5
	REFINE	-	6.5	11.1	12.6	**19.6**	18.0	**14.3**	15.9	17.0	11.0	12.9	**32.9**	31.1	16.9
	REFINE	SFT	5.1	12.6	13.5	15.9	18.4	13.2	**16.2**	**17.4**	12.5	**16.2**	32.0	31.4	17.0
	REFINE	REFINE	**6.7**	**14.5**	**14.1**	18.4	**22.8**	13.9	15.9	**17.4**	**15.5**	11.8	32.2	**32.3**	**18.0**
DeltaNet 1.3B	-	-	6.5	8.7	10.0	4.8	6.4	12.4	**16.3**	9.4	17.7	15.1	33.8	29.0	14.2
	SFT	-	5.7	9.4	8.3	9.2	8.6	14.7	16.1	15.5	22.9	15.2	33.1	29.2	15.7
	REFINE	-	6.5	9.5	10.1	**9.6**	**8.7**	**15.2**	15.0	16.2	25.0	21.4	**35.9**	**31.1**	17.0
	REFINE	SFT	7.2	**9.6**	10.5	6.8	7.2	14.9	16.2	**17.3**	28.0	16.6	34.1	29.2	16.5
	REFINE	REFINE	**7.5**	9.2	**11.5**	9.5	8.6	14.7	16.1	16.5	**31.5**	**24.7**	35.2	30.0	**17.9**

In addition to long-context retrieval tasks, we also report the effectiveness of REFINE mid-training on multi-doc QA tasks and long-context tasks in Table 2 and 3, respectively. In Table 2, REFINE mid-trained models (3rd rows) outperform SFT mid-trained models (2nd rows) with large gaps.

3.2 IMPACT OF REFINE ON POST-TRAINING

We show that REFINE strengthens post-training, which aims to align the model's responses to a given task. In our experiments, we fine-tune the mid-trained models in §3.1 on synthetically generated samples for the target tasks of RULER Hsieh et al. (2024) SQuADQA and HotpotQA.

During post-training, we apply REFINE as a nested learning algorithm. Within each training loop, we first update the model on the prompt with REFINE before generating a final response, which is fine-tuned to align with a reference response. We compare three post-training scenarios. (1) *SFT*: we fine-tune the model directly on the post-training dataset with NTP (no nested learning). (2) *Nested SFT*: we apply nested training strategy with NTP loss. (3) *Nested* REFINE: we apply nested training strategy with REFINE.

Table 2 shows that post-training with nested REFINE (6th rows) outperforms SFT (4th rows) and nested SFT (5th rows). These results suggest that NSP provides better task-agnostic learning signals than NTP to capture the context distribution in the fast weights.

3.3 IMPACT OF REFINE ON TEST-TIME-TRAINING (TTT)

REFINE can be used during inference to improve performance on a target task. During inference, we apply REFINE on the prompt before letting the model generate the final response. In order to maximize memory capacity over long context, we provide more direct learning signals by using binary exact match reward ($\mathcal{R}^{\text{binary}}$) and a higher RL loss coefficient ($\lambda_{\text{RL}} = 0.4$). We replace REFINE with pure SFT on the prompt for comparison.

We apply TTT on the post-trained models for multi-doc QA tasks in §3.2. The 7th and 8th rows of Table 2 show the results of SFT TTT and REFINE TTT, respectively. REFINE consistently outperforms SFT during TTT similar to mid-training and post-training scenarios.

We observe a similar result in long-context tasks. Table 3 shows that TTT with REFINE yields superior performance compared to SFT across diverse subtasks in LongBench Bai et al. (2024). This suggests that NSP provides stronger adaptation signals to facilitate compression of contextual information in fast weights not only at the token-level as in NTP, but also at the sequence level.

Discussion We introduce the NSP training objective for fast weight language models to address the limitations of NTP in providing sequence-level feedback. We propose REFINE, a RL framework which leverages entropy-based token selection and sequence-level rewards to efficiently train fast weight models under the NSP objective. Our experiments demonstrate that REFINE is effective throughout the training life cycle of fast weight models, showing consistent improvements in long-context benchmarks. REFINE presents RL for NSP as a flexible and practical pathway towards long context modeling of fast weight architectures.

REFERENCES

Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*, 2023.

Ekin Akyürek, Mehul Damani, Adam Zweiger, Linlu Qiu, Han Guo, Jyothish Pari, Yoon Kim, and Jacob Andreas. The surprising effectiveness of test-time training for few-shot learning. *arXiv preprint arXiv:2411.07279*, 2024.

Simran Arora, Brandon Yang, Sabri Eyuboglu, Avanika Narayan, Andrew Hojel, Immanuel Trummer, and Christopher Ré. Language models enable simple systems for generating structured views of heterogeneous data lakes. *arXiv preprint arXiv:2304.09433*, 2023.

Stephen Bach, Victor Sanh, Zheng-Xin Yong, Albert Webson, Colin Raffel, Nihal V Nayak, Abheesht Sharma, Taewoon Kim, M Saiful Bari, Thibault Fevry, et al. Promptsource: An integrated development environment and repository for natural language prompts. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 93–104, 2022.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3119–3137, 2024.

Rachit Bansal, Aston Zhang, Rishabh Tiwari, Lovish Madaan, Sai Surya Duvvuri, Devvrit Khatri, David Brandfonbrener, David Alvarez-Melis, Prajjwal Bhargava, Mihir Sanjay Kale, et al. Let's (not) just put things in context: Test-time training for long-context llms. *arXiv preprint arXiv:2512.13898*, 2025.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. *arXiv preprint arXiv:2501.00663*, 2024.

Ali Behrouz, Zeman Li, Praneeth Kacham, Majid Daliri, Yuan Deng, Peilin Zhong, Meisam Razaviyayn, and Vahab Mirrokni. Atlas: Learning to optimally memorize the context at test time. *arXiv preprint arXiv:2505.23735*, 2025a.

Ali Behrouz, Meisam Razaviyayn, Peilin Zhong, and Vahab Mirrokni. Nested learning: The illusion of deep learning architectures. *arXiv preprint arXiv:2512.24695*, 2025b.

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

Rewon Child. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

Kevin Clark, Kelvin Guu, Ming-Wei Chang, Panupong Pasupat, Geoffrey Hinton, and Mohammad Norouzi. Meta-learning fast weight language models. *arXiv preprint arXiv:2212.02475*, 2022.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*, 2024.

Qingxiu Dong, Li Dong, Yao Tang, Tianzhu Ye, Yutao Sun, Zhifang Sui, and Furu Wei. Reinforcement pre-training. *arXiv preprint arXiv:2506.08007*, 2025.

Yossi Gandelsman, Yu Sun, Xinlei Chen, and Alexei A Efros. Test-time training with masked autoencoders. In *Advances in Neural Information Processing Systems*, volume 35, pp. 29374–29385, 2022.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL `https://zenodo.org/records/12608602`.

Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large language models via multi-token prediction. *arXiv preprint arXiv:2404.19737*, 2024.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First conference on language modeling*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 8342–8360, 2020.

Ali Hatamizadeh, Syeda Nahida Akter, Shrimai Prabhumoye, Jan Kautz, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, and Yejin Choi. Rlp: Reinforcement as a pretraining objective. *arXiv preprint arXiv:2510.01265*, 2025.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.

Audrey Huang, Adam Block, Dylan J Foster, Dhruv Rohatgi, Cyril Zhang, Max Simchowitz, Jordan T Ash, and Akshay Krishnamurthy. Self-improvement in language models: The sharpening mechanism. *arXiv preprint arXiv:2412.01951*, 2024.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pp. 5156–5165. PMLR, 2020.

Feyza Duman Keles, Pruthuvi Mahesakya Wijewardena, and Chinmay Hegde. On the computational complexity of self-attention. In *International conference on algorithmic learning theory*, pp. 597–619. PMLR, 2023.

Wojciech Kryściński, Nazneen Rajani, Divyansh Agarwal, Caiming Xiong, and Dragomir Radev. Booksum: A collection of datasets for long-form narrative summarization. In *Findings of the association for computational linguistics: EMNLP 2022*, pp. 6536–6558, 2022.

Yangzhou Liu, Yue Cao, Hao Li, Gen Luo, Zhe Chen, Weiyun Wang, Xiaobo Liang, Biqing Qi, Lijun Wu, Changyao Tian, et al. Sequential diffusion language models. *arXiv preprint arXiv:2509.24007*, 2025.

Colin Lockard, Prashant Shiralkar, and Xin Luna Dong. Openceres: When open information extraction meets the semi-structured web. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 3047–3056, 2019.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. Cross-task generalization via natural language crowdsourcing instructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3470–3487, 2022.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35: 27730–27744, 2022.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc-Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The lambada dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 1525–1534, 2016.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8732–8740, 2020.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International conference on machine learning*, pp. 9229–9248. PMLR, 2020.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, et al. Learning to (learn at test time): Rnns with expressive hidden states. *arXiv preprint arXiv:2407.04620*, 2024.

Arnuv Tandon, Karan Dalal, Xinhao Li, Daniel Koceja, Marcel Rød, Sam Buchanan, Xiaolong Wang, Jure Leskovec, Sanmi Koyejo, Tatsunori Hashimoto, et al. End-to-end test-time training for long context. *arXiv preprint arXiv:2512.23675*, 2025.

Yi Tay, Mostafa Dehghani, Vinh Q Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Siamak Shakeri, Dara Bahri, Tal Schuster, et al. Ul2: Unifying language learning paradigms. *arXiv preprint arXiv:2205.05131*, 2022.

Qwen Team. Qwen3 technical report, 2025. URL `https://arxiv.org/abs/2505.09388`.

TogetherAI. Long data collections database. `https://huggingface.co/datasets/togethercomputer/Long-Data-Collections`, 2024.

Gido M van de Ven, Nicholas Soures, and Dhireesha Kudithipudi. Continual learning and catastrophic forgetting. *arXiv preprint arXiv:2403.05175*, 2024.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Maurice Weber, Dan Fu, Quentin Anthony, Yonatan Oren, Shane Adams, Anton Alexandrov, Xiaozhong Lyu, Huu Nguyen, Xiaozhe Yao, Virginia Adams, et al. Redpajama: an open dataset for training large language models. *Advances in neural information processing systems*, 37: 116462–116492, 2024.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. *Advances in neural information processing systems*, 37: 115491–115522, 2024.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *International Conference on Learning Representations*, 2025.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297, 2020.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

Tianyuan Zhang, Sai Bi, Yicong Hong, Kai Zhang, Fujun Luan, Songlin Yang, Kalyan Sunkavalli, William T Freeman, and Hao Tan. Test-time training done right. *arXiv preprint arXiv:2505.23884*, 2025.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, et al. Ttrl: Test-time reinforcement learning. *arXiv preprint arXiv:2504.16084*, 2025.

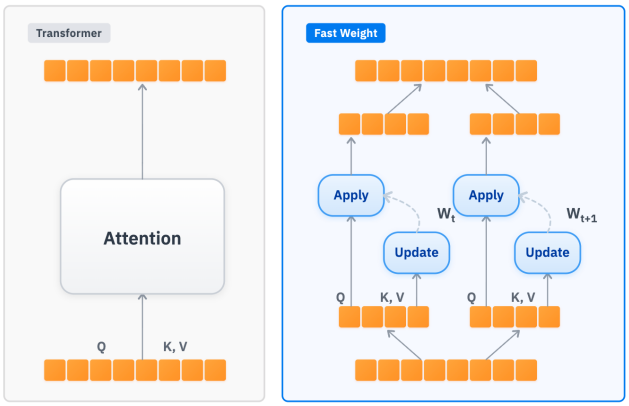

Figure .2: **Comparison of Standard Transformer and Fast Weight Models**, adapted from Zhang et al. (2025). Fast weight models replace attention with a fixed-size memory implemented as a weight matrix (W), and updated according to Eq. (A.1).

Table A.1: **Training Phases for Fast Weight Language Models.** REFINE can be applied across all phases beyond pre-training to improve long-context modeling, using different data sources.

Training Phase	Source of Training Data
Pre-training	General web-scraped data
Mid-training	Similar data to pre-training
Post-training	Task / instruction / preference data
Test-Time Training	Test data

A BACKGROUND

Fast Weight Architectures. Fast weight architectures replace global attention in standard transformers with fixed-size memory parameterized as weight matrices. Instead of keeping a growing key-value cache, fast weight models continually update the weight matrices as tokens are processed to store contextual information. As a result, fast weight models are often associated with test-time training Behrouz et al. (2024) and meta-learning Clark et al. (2022) due to the continual and task-agnostic nature of their weight updates. The update rule for fast weights can be generalized as follows Zhang et al. (2025):

$$W_{t+1} \leftarrow W_t - \eta \nabla_{W_t} \ell\big(W_t k_t, v_t\big) \tag{A.1}$$

where W denotes the fast weight, η is the learning rate, and k_t, v_t are the key, value representations of the input token at position t. This update rule can be viewed as learning the online mapping from key to value representations. The output representation is retrieved from fast weights via an apply operation, *i.e.*, $W_t q_t$, where q_t is the token's query representation. Fig. .2 illustrates the difference between standard transformer and fast weight language models.

Training Phases for Language Models. Pre-trained language models undergo additional training stages that rely on different sources of data and supervision (Table A.1). We follow the standard taxonomy on three additional training phases: *mid-training*, *post-training*, and *test-time training*.

Mid-training is an extension (or continued version) of pre-training, generally used for the adaptation of a pre-trained model to specific domains or capability (Gururangan et al., 2020). In this paper, we apply REFINE to mid-train models on the same training dataset as pre-training to adapt pre-trained models to the NSP objective and reward.

Post-Training fine-tunes pre-trained models to follow instructions and align their responses with human preferences (Ouyang et al., 2022; Rafailov et al., 2024; Guo et al., 2025). This phase typically involves SFT on task-specific instruction-response pairs, with relatively fewer gradient steps compared to pre-training. We apply REFINE during post-training through the nested learning Behrouz et al.

(2025b) technique: within each training loop, we first use REFINE to update the model on the instruction prompt alone, and then use SFT to fine-tune the model's final response.

Test-Time Training (TTT) adapts model parameters at inference time using self-supervised objectives to handle distribution shifts from source to target (Sun et al., 2020; Wang et al., 2021; Gandelsman et al., 2022; Sun et al., 2024). TTT naturally integrates with fast weight architectures by design: each input token updates the fast weights via gradient-based rules, enabling the model to memorize and adapt to the given context on-the-fly (Zhang et al., 2025; Behrouz et al., 2024). However, fast weight models trained purely with NTP can still struggle on long-context retrieval (e.g., Needle-in-a-Haystack (Hsieh et al., 2024)) due to unstable fast weight updates or insufficient long-horizon supervision.

RL for Language Modeling. Recent work has shown that NTP can be formulated as a reward maximization problem with RL Dong et al. (2025); Hatamizadeh et al. (2025). These methods produce reasoning traces from the model before predicting the next token and provide rewards based on the similarity between the prediction and the ground truth. Existing works focus on applying RL on standard transformer LLMs with basic reasoning capability, but it is still an open question whether RL can be applied to pre-trained fast weight models. We demonstrate that RL can improve long-context capabilities of fast weight models during mid-, post-, and test-time training even without prior instruction tuning. More details are discussed in Appendix §B.

B RELATED WORK

Multi-Token Prediction. Predicting more than one token as a training objective has been explored in previous studies. Gloeckle et al. (2024) tackled this problem by estimating k-tokens in parallel using k independent output heads. They achieve significant gains in throughput by applying architectural modifications to the language model, with minimal degradation of performance on downstream tasks. However, this approach is limited in capturing dependencies among the predicted tokens and relies on a fixed prediction horizon k. REFINE computes rewards based on each multi-token rollout as a whole, capturing the semantic connection among them.

Liu et al. (2025) uses diffusion-based generation to predict multiple masked tokens simultaneously and optimizes the cross-entropy loss between the predicted tokens and masked ground truth tokens. Their method is primarily designed for standard attention-based transformer architectures that predict the next token with masking. However, REFINE is designed to train fast weight language models that store information and update its parameters with a fundamentally different set of rules. We view this work as a valuable source of motivation experimented on a different class of models.

Continued Pre-Training with RL. Studies have explored using RL as a tool for the next-token prediction objective. Dong et al. (2025) samples reasoning traces before next-token prediction and assigns rewards based on the similarity of the byte-sequences of predicted and ground truth tokens. Hatamizadeh et al. (2025) takes a similar approach by sampling reasoning traces for next-token prediction but assigning rewards by measuring the gap between the log-likelihood of the ground truth token with and without the reasoning trace as context. However, these works focus on attention-based transformer models (DeepSeek-R1-Distill-Qwen-14B Guo et al. (2025), and Qwen3-1.7B-Base Team (2025), respectively) and assume basic reasoning capabilities that enable exploration with Chain-of-Thought. Whether RL can be used for pre-trained fast weight models without prior instruction tuning or human preference optimization, especially in long context settings, is yet to be explored. Further, REFINE leverages RL for training on the NSP objective which optimizes sequence-level predictions rather than single-token predictions under the NTP objective.

TTT in Language Models. Recent work has explored training standard transformer-based language models during inference to improve performance on target tasks. These methods usually involve extracting task-related learning signals from the model itself for offline adaptation. Akyürek et al. (2024) generates relevant in-context examples from the given task and trains the model on those examples before generating the final answer to the actual task. RL-based methods extract pseudo-labels by aggregating multiple responses to the same task and assigning rewards to each response based on their similarity to the pseudo-label Zuo et al. (2025); Huang et al. (2024). Recently, context based test-time training has also been explored in transformer-based language models. In an attempt

to overcome the inherent limitations of static attention, Bansal et al. (2025) executes gradient updates on the query projection matrices using the context while keeping other parameters frozen. These approaches apply task-aware or task-agnostic TTT methods to standard transformer-based language models. TTT with REFINE, on the other hand, aims to improve contextual adaptation and memory of fast weights for long-context, which is a novel setting that has not yet been explored.

Efficient Attention Variants. Standard transformer-based language models with full attention incur quadratic computational complexity as a function of context length. Recent work has developed techniques to reduce the computational and memory overhead in long context modeling. Sparse attention addresses this problem by introducing computational sparsity in the original attention mechanism. Grouped Query Attention Ainslie et al. (2023) employs sparsity along the head dimension to assign each query head to different groups that share a single key and value heads. Sliding window attention Child (2019); Beltagy et al. (2020); Zaheer et al. (2020) architectures leverage sparsity along the context by computing local attention on a fixed number of contiguous tokens.

Linear attention approximates the softmax kernel in the attention formulation to achieve linear computational complexity. Linformer Wang et al. (2020) replaces the attention mechanism with low rank matrix operations, which has shown to be effective for sequence processing but limited in autoregressive generation. Performer Choromanski et al. (2020) uses orthogonal random features to approximate softmax kernels in the attention, achieving linear complexity without employing low rank matrices. Similarly, Linear Transformer Katharopoulos et al. (2020) approximates the softmax with linear dot-product of kernel feature maps. The success of linear attention has motivated the development of architectures that operate on linear computational complexity by design, including State Space Models (SSMs), such as Mamba Gu & Dao (2024); Dao & Gu (2024). Unlike these attention variants that aim to approximate full attention, fast weight models rely on fixed-size memory with predefined online update rules to directly store contextual information in the parameters. We therefore propose REFINE as a training framework targeting fast weight models that are fundamentally different in terms of architecture compared to attention-based transformer models.

C FROM NEXT-TOKEN TO NEXT-SEQUENCE PREDICTION

Our goal is to obtain better fast weight initializations for long-context modeling by leveraging the NSP objective. We present RL as a solution to the limitations of SFT in optimizing sequence-level predictions, explained below.

Next Token Prediction (NTP). Standard language model pre-training involves minimizing the cross-entropy (CE) loss of the NTP objective. Given an input sequence $S = (x_1, \ldots, x_T)$, the CE loss is computed using the predicted probability distributions at each token position and the corresponding ground truth tokens:

$$\mathcal{L}_{\text{NTP}} = \sum_t -\log p(x_{t+1} \mid x_{\leq t}). \tag{C.1}$$

The NTP loss has two key limitations for long-context modeling. First, each term in the summation only considers single-token prediction, ignoring the semantic relationships among multiple tokens that follow the prefix. Second, NTP ignores local regions in the sequence that may be useful over the long-context by aggregating the terms uniformly.

Next Sequence Prediction (NSP). We aim to resolve the shortcomings of standard NTP by proposing the NSP objective for training fast weight models. Unlike NTP which optimizes token-by-token predictions, NSP optimizes multi-token sequence alignment at selected positions $\mathcal{T}^* \subseteq \{1, \ldots, T\}$:

$$\mathcal{L}_{\text{NSP}} = \sum_{t \in \mathcal{T}^*} \mathcal{L}_{\text{seq}}(\hat{x}_{t+1:t+k}, x_{t+1:t+k}), \quad k > 1 \tag{C.2}$$

where $\mathcal{L}_{\text{seq}}$ measures the discrepancy between the predicted sequence $\hat{x}_{t+1:t+k}$ given prefix $x_{\leq t}$ and the ground truth continuation $x_{t+1:t+k}$.

A straightforward choice for $\mathcal{L}_{\text{seq}}$ is the CE loss. However, naively applying the CE loss at every position t requires generating k-token completions given all possible prefixes, which is computationally expensive especially for long contexts. Furthermore, directly matching a single reference will

Table D.1: Summary of datasets and benchmarks used across training phases.

Phase	Dataset	Metric	Context	Size
Mid-training	Long-Data-Collections	-	16K	∼200M tokens
	RULER NIAH	recall	4K/8K/16K	500 per context
	Booksum	NTP Accuracy, CE loss	≤16K	9600
Post-training	RULER SQuADQA	recall	4K/8K/16K	1600 train / 200 test
	RULER HotpotQA	recall	4K/8K/16K	1600 train / 200 test
Test-time	NarrativeQA (NQ)	F1	≤16K	56
	Qasper (QR)	F1	≤16K	184
	MultiFieldQA (MF)	F1	≤16K	136
	HotpotQA (HP)	F1	≤16K	96
	2WikiMHQA (2W)	F1	≤16K	184
	QMSum (QM)	rouge	≤16K	104
	MultiNews (MN)	rouge	≤16K	192
	SAMSum (SS)	rouge	≤16K	152
	TREC (TC)	accuracy	≤16K	200
	TriviaQA (TQ)	accuracy	≤16K	120
	LCC (LC)	code similarity	≤16K	488
	RepoBench-P (RP)	code similarity	≤16K	320
Commonsense	PIQA	accuracy		All
	HellaSwag(Hella.)	normalized accuracy		All
	WinoGrande(Wino.)	accuracy		All
	ARC-e	accuracy		All
	ARC-c	normalized accuracy		All
	Wikitext(Wiki.)	perplexity		All
	LAMBADA(LMB.)	perplexity, accuracy		All
	FDA	recall		All
	SWDE	recall		All

over-penalize plausible answers not exactly matching the ground truth. For example, for the ground truth sequence "cars are fast", a semantically equivalent sequence "automobiles move quickly" may still result in a high CE loss.

We propose two approaches to tackle this issue. First, instead of unrolling tokens at every index t, we select informative positions $\mathcal{T}^*$ with high NTP entropy, which indicate high uncertainty. Second, we optimize Eq. (C.2) using an RL algorithm that maximizes the expected self-supervised reward R of sequence predictions. Let π_θ denote the language model parameterized by θ. We define the sequence-level loss as follows:

$$\mathcal{L}_{\text{seq}} = -\mathbb{E}_{\hat{x}_{t+1:t+k}\sim\pi_\theta(\cdot|x_{\leq t})}\big[R(\hat{x}_{t+1:t+k}, x_{t+1:t+k})\big]. \tag{C.3}$$

This formulation has two advantages: (1) optimizing k-step continuations leverages higher information content compared to optimizing single-token predictions; (2) we can assign rewards to multiple plausible continuations based on their semantic similarity to the ground truth. For brevity, we use $R(t)$ to denote $R(\hat{x}_{t+1:t+k}, x_{t+1:t+k})$.

While prior work has explored NSP for standard transformer LLMs (Gloeckle et al., 2024; Liu et al., 2025), we are the first to investigate RL-based NSP for fast weight models. More discussion can be found in Appendix §B.

D Additional Experiment Details

D.1 Setup

Models. We use two fast weight language models, LaCT-760M Zhang et al. (2025) and DeltaNet-1.3B Yang et al. (2024), as the pre-trained models. LaCT adapts the model by updating its fast weight parameters, whereas DeltaNet keeps parameters fixed but updates a parallelizable memory state. We show that REFINE can improve these distinct fast weight mechanisms in mid-, post-, and test-time-training.

Table D.2: Training hyperparameters.

Params	Values	Params	Values
Actor gradient clip	0.2	Learning rate	10^{-6}
Mid-Train Batch size	128	Adam (β_1, β_2)	$(0.9,\ 0.999)$
Post-Train Batch size	64	Weight decay	0.01
Test-Time-Train Batch size	8	Sampling temperature τ	1.0
Mid-Train PPO mini batch size	32	Max prompt length	16384
Post-Train PPO mini batch size	16	Entropy loss coefficient	0
Test-Time-Train PPO mini batch size	4	KL loss coefficient	0
Mid-Train λ_{RL}	0.2	n (rollouts / position)	1
Post-Train λ_{RL}	0.2	k (rollout length)	5
Test-Time-Train λ_{RL}	0.4	c (chunks / sequence)	8
Mid-Train λ_{SFT}	1.0	Mid-Train Reward	R^{φ}
Post-Train λ_{SFT}	1.0	Post-Train Reward	R^{hybrid}
Test-Time-Train λ_{SFT}	1.0	Test-Time-Train Reward	R^{binary}

Datasets and Benchmarks. As shown in Table A.1, data for each training phase comes from different sources. For mid-training, we employ a training dataset similar to that used to pre-train the fast weight models. Specifically, *we perform mid-training with Long-Data-Collections* TogetherAI (2024), which is the pre-training dataset for LaCT (Zhang et al., 2025). We evaluate the quality of the mid-trained models on RULER NIAH tasks Hsieh et al. (2024) and Booksum Kryściński et al. (2022).

We consider two additional scenarios: (1) multi-doc QA tasks and (2) long-context tasks. For multi-doc QA tasks, we *conduct post-training on synthetically generated SQuADQA and HotpotQA tasks from RULER* Hsieh et al. (2024). Then, we apply TTT during evaluation. For the long-context tasks, we employ 12 tasks from LongBench Bai et al. (2024) and apply TTT on the mid-trained models.

We summarize the datasets and benchmarks used for training and evaluation in Table D.1. The Long-Data-Collections TogetherAI (2024) dataset contains a 68.8B-token pre-training corpus subsampled from RedPajama Weber et al. (2024), Pile Gao et al. (2020)), UL2 Oscar Tay et al. (2022), NI Mishra et al. (2022), and P3 Bach et al. (2022). We use a 200M-token subset of Long-Data-Collections for mid-training. For LongBench Bai et al. (2024), we select 12 subtasks that are English-based. We leave out MuSiQue and GovReport tasks as they have less than 20 samples under 16K tokens.

D.2 Training Configurations

Hyperparameters. We provide the full list of training hyperparameters used for REFINE during all training phases in Table D.2. We only adjust the train batch size, reward function, and RL loss coefficient by training phase, while keeping all else equal.

Compute. We use 8 L40 GPUs for mid-training and post-training, and 4 L40 GPUs for TTT. We use fewer GPUs for TTT because the train batch size for TTT is smaller (8 samples per batch) compared to other training phases (128 for mid-training, 64 for post-training). Mid-training LaCT-760M and DeltaNet-1.3B with REFINE on 200M tokens from Long-Data-Collections TogetherAI (2024) at 16K context takes approximately 24 hours.

E Additional Experiments

E.1 Ablations

Rollout Length. We examine the effect of rollout length k on REFINE mid-training, which is the number of tokens to unroll per rollout. k determines how far the model is expected to predict given a prefix. We mid-train both models with different values of k and evaluate on LongBench tasks in Table 3. We observe that the average score increases until $k = 5$ and decreases when $k = 7$ (Fig. E.1, left). We hypothesize that the sharpness of the reward starts to degrade when the rewards are averaged over longer rollouts.

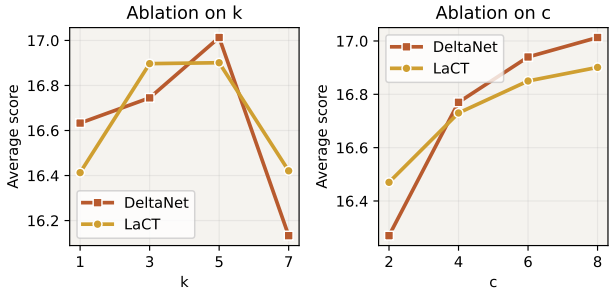

Figure E.1: **Ablation on k and c.** We mid-train models with different numbers of tokens per rollout k (**left**) and numbers of chunks per sequence c (**right**). We evaluate on 16K-context samples from 12 tasks in LongBench Bai et al. (2024). With cosine similarity reward, there is an optimal k. Higher c leads to more NSP training per sequence, which leads to better overall performance.

Table E.1: **Impact of reward function on mid-training.** We compare $\mathcal{R}^{\text{binary}}$ (binary exact match) and $\mathcal{R}^{\varphi}$ (ours) reward strategies on 12 tasks in LongBench.

	Reward	Avg. Score
LaCT-760M	$\mathcal{R}^{\text{binary}}$	16.6
+REFINE MidTr	$\mathcal{R}^{\varphi}$ (ours)	16.9
DeltaNet-1.3B	$\mathcal{R}^{\text{binary}}$	16.5
+REFINE MidTr	$\mathcal{R}^{\varphi}$ (ours)	17.0

Number of Chunks per Sequence. We study the impact of number of chunks c per sequence on downstream performance by mid-training the models with different numbers of chunks. c determines the number of target tokens sampled based on entropy values, as well as the total number of rollouts per sequence. We evaluate the mid-trained models on LongBench tasks in Table 3. We find that the average score increases consistently as the number of chunks per sequence increases (Fig. E.1, right). The average score increases from 16.5 ($c = 2$) to 16.9 ($c = 8$) for LaCT-760M and from 16.3 ($c = 2$) to 17.0 ($c = 8$) for DeltaNet-1.3B. This indicates that the quality of fast weight initializations increases as the frequency of sequence-level predictions increases.

E.2 ANALYSIS

Impact of Reward Functions. We analyze the impact of alternative reward functions for REFINE. During mid-training, REFINE assigns a smooth, semantically driven reward to each rollout based on the cosine similarity of the hidden states of predicted and ground truth tokens (Eq. (2)). We repeat the mid-training process after replacing cosine similarity rewards ($\mathcal{R}^{\varphi}$) with binary exact match rewards ($\mathcal{R}^{\text{binary}}$). Table E.1 shows that $\mathcal{R}^{\varphi}$ achieves superior performance on both models for mid-training: +1.8% over $\mathcal{R}^{\text{binary}}$ for LaCT-760M and +3.0% for DeltaNet-1.3B. This demonstrates that the similarity-based reward leads to better generalization under the NSP training objective.

REFINE uses binary exact match reward $\mathcal{R}^{\text{binary}}$ during test-time for offline adaptation of the model before generating the response. We repeat TTT with cosine similarity reward instead and report the average score on LongBench tasks in Table 3. Tab E.2 shows that the binary reward is optimal for TTT, but the cosine similarity reward also performs well, higher than pure SFT for TTT.

In order to investigate the stability of mid-training with REFINE, we track the mean and standard deviation of the cosine similarity reward for different rollout lengths ($k = 3, 5, 7$) as shown in Fig. E.2. We find that the mean and the standard deviation of the reward remains stable across training steps. However, as rollout length increases, the mean and standard deviation of the reward decrease, suggesting that the learning signal from the reward may lose sharpness for larger k.

Analysis of Entropy-Based Token Selection. We analyze the role of entropy-based token selection on REFINE mid-training. REFINE samples a target token from each chunk weighted by the token-level NTP entropy. Rollouts are generated to predict the local region following the sampled

Table E.2: **Impact of reward function on TTT.** We compare different TTT reward strategies on 12 tasks in LongBench: binary exact match between predicted and ground truth completion ($\mathcal{R}^{\text{binary}}$), and cosine similarity of the hidden states of predicted and ground truth completions ($\mathcal{R}^{\varphi}$).

	MidTr	TTT	MidTr Reward	TTT Reward	Avg. Score
LaCT-760M	REFINE	-	$\mathcal{R}^{\varphi}$	-	16.9
	REFINE	SFT	$\mathcal{R}^{\varphi}$	-	17.0
	REFINE	REFINE	$\mathcal{R}^{\varphi}$	$\mathcal{R}^{\varphi}$	17.5
	REFINE	REFINE	$\mathcal{R}^{\varphi}$	$\mathcal{R}^{\text{binary}}$	18.0
DeltaNet-1.3B	REFINE	-	$\mathcal{R}^{\varphi}$	-	17.0
	REFINE	SFT	$\mathcal{R}^{\varphi}$	-	16.5
	REFINE	REFINE	$\mathcal{R}^{\varphi}$	$\mathcal{R}^{\varphi}$	17.6
	REFINE	REFINE	$\mathcal{R}^{\varphi}$	$\mathcal{R}^{\text{binary}}$	17.9

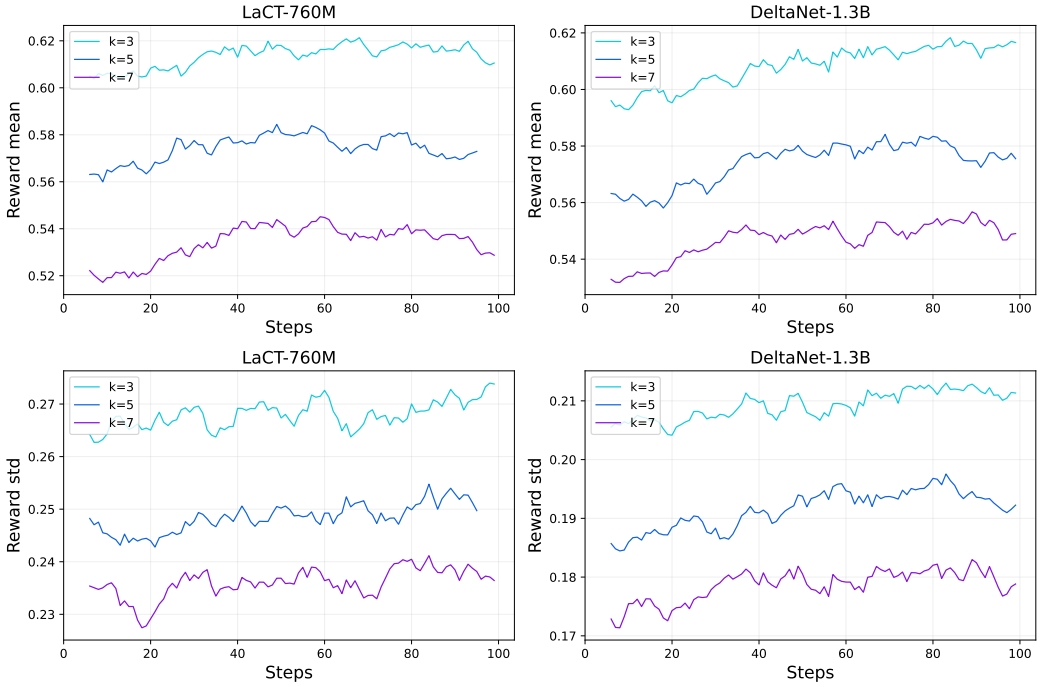

Figure E.2: **Reward Distribution.** We report the mean and standard deviation of the cosine similarity reward during mid-training for different values of k. As the rollout length increases, the mean reward (**upper left**, **upper right**) decreases and the standard deviation (**lower left**, **lower right**) also decreases.

target tokens. We repeat the mid-training process after replacing entropy-based sampling with three alternative sampling methods: uniform sampling, maximum entropy selection, and minimum entropy selection. Table E.3 shows that entropy-weighted sampling achieves the best performance on both models: +4.3% over uniform, +3.0% over max entropy, and +1.8% over min entropy for LaCT-760M; +6.9% over uniform, +1.8% over max entropy, and +1.2% over min entropy for DeltaNet-1.3B. This shows that NSP training is most effective when applied to regions with a balanced mixture of uncertainty levels.

Validation Loss We report the validation loss on the Booksum Kryściński et al. (2022) dataset in Long-Data-Collections TogetherAI (2024) during mid-training. The validation loss for SFT on LaCT stays constant as the mid-training data is the same as its pre-training data. However, we see a notable decrease in validation loss with REFINE (Fig. E.3), indicating that NSP provides learning signal that is unique from standard NTP training.

Table E.3: **Impact of token selection.** We compare various token selection strategies on 12 tasks in LongBench: uniform random, selecting the token with maximum entropy ($\arg\max H$) or minimum entropy ($\arg\min H$), and our entropy-weighted sampling.

	Sampling	Avg. Score
LaCT-760M +REFINE MidTr	Uniform	16.2
	$\arg\max H$	16.4
	$\arg\min H$	16.6
	Ours	16.9
DeltaNet-1.3B +REFINE MidTr	Uniform	15.9
	$\arg\max H$	16.7
	$\arg\min H$	16.8
	Ours	17.0

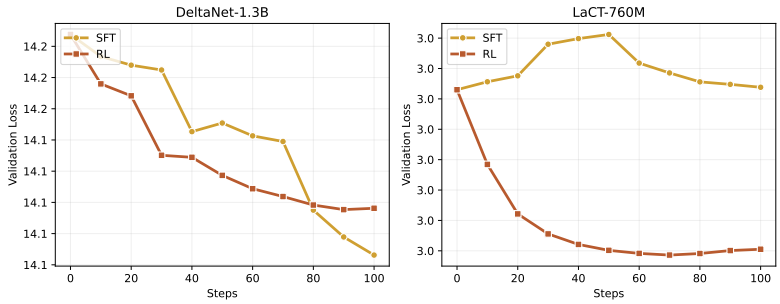

Figure E.3: **Validation Loss on Booksum Dataset.** We track the NTP loss on the Booksum validation dataset in Long-Data-Collections TogetherAI (2024) throughout mid-training. The validation loss for SFT on LaCT-760M does not decrease as the pretrained model has already been pre-trained on the mid-training dataset.

Entropy Distribution. We report the NTP entropy distribution of a randomly selected sample in order to illustrate the effects of entropy-based token selection in REFINE (Fig. E.4). We find that there are no index-dependent patterns in the distribution, which justifies per-chunk target token sampling weighted by entropy.

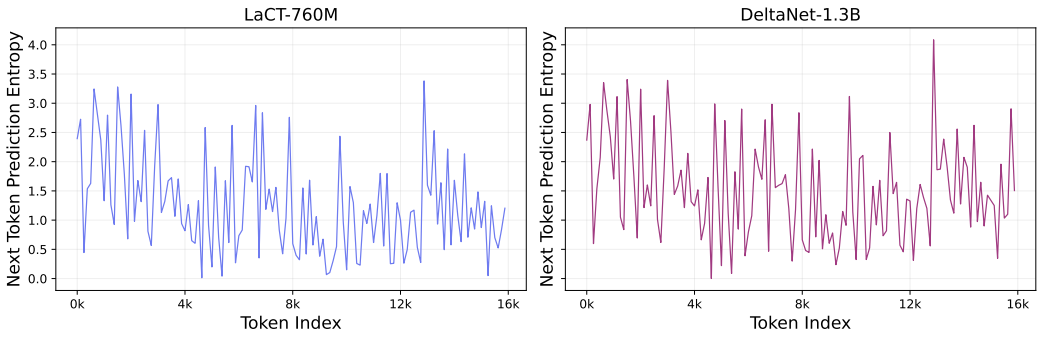

Figure E.4: **NTP Entropy Distribution.** We compute the token-level NTP entropy of a randomly selected sample. We use the same sample to extract the entropy distribution from both models.

Performance on Short-Context Tasks. We investigate whether enhancing long context handling capabilities of fast weight models leads to degradation of performance on out-of-distribution tasks such as short-context in-context retrieval and commonsense reasoning. We evaluate mid-trained models on 9 relevant short-context tasks with lm-evaluation-harness Gao et al. (2024): PIQA Bisk et al. (2020), HellaSwag(Hella.) Zellers et al. (2019), WinoGrande(Wino.) Sakaguchi et al. (2020), ARC-easy(ARC-e), ARC-challenge(ARC-c) Clark et al. (2018), Wikitext(Wiki) Merity et al. (2016),

Table E.4: **Performance on Short-Context Tasks.** We evaluate mid-trained models on short-context benchmarks to verify that REFINE does not cause catastrophic forgetting.

Models	Wiki ppl ↓	LMB. ppl ↓	FDA recall ↑	SWDE recall ↑	LMB. acc ↑	PIQA acc ↑	Hella. acc_n ↑	Wino. acc ↑	ARC-e acc ↑	ARC-c acc_n ↑	Avg
LaCT-760M	20.8	29.8	36.7	66.0	32.4	67.5	41.5	53.4	48.4	27.6	45.1
+SFT MidTr	20.7	28.9	37.2	66.2	32.9	67.4	41.4	52.5	48.4	27.1	45.0
+REFINE MidTr	20.8	30.2	36.1	66.5	32.3	67.3	41.4	52.8	48.8	27.5	45.0
DeltaNet-1.3B	16.7	14.7	18.0	54.4	43.0	70.8	50.5	53.7	58.3	25.9	50.4
+SFT MidTr	16.8	15.3	17.7	54.5	42.7	71.1	50.1	53.8	58.4	26.4	50.4
+REFINE MidTr	16.8	15.6	17.5	54.5	42.2	71.1	50.2	53.6	58.1	26.2	50.2

LAMBADA(LMB) Paperno et al. (2016), FDA Arora et al. (2023), and SWDE Lockard et al. (2019). Table E.4 shows that REFINE sustains performance in these tasks, suggesting that NSP complements NTP without inducing catastrophic forgetting van de Ven et al. (2024).

F QUALITATIVE EXAMPLES

We provide qualitative examples of cosine similarity reward assignment during mid-training in **??**. We randomly sample prefixes from the mid-training dataset and generate four $k = 5$ continuations each using the pre-trained LaCT-760M model. The cosine similarity reward is designed to capture the semantic similarity between the predicted and ground truth continuations. The highest reward values for each example are highlighted in bold. The examples demonstrate that the reward effectively captures semantic similarity beyond exact lexical matching. For instance, in example 2, "enjoyed every minute of it" achieves near-perfect alignment (0.961) with "loved every minute of it", while semantically divergent predictions like "would not recommend it at" receive much lower scores (0.463). Similarly, example 6 shows sensitivity to mathematical concepts, where "is a convergent integral" receives a high score (0.758) for preserving the convergence concept from "is also convergent according", while less relevant predictions such as "shall not be less than" receive lower rewards (0.512). These examples illustrate that the cosine similarity reward provides meaningful learning signals for training the model to generate semantically coherent continuations.

Table F.1: Qualitative examples of cosine similarity reward assignment during mid-training. GT denotes the ground truth continuation, and P1–P4 denote the four predicted continuations generated by the model.

Example	Type	Text	Cosine Reward
Ex. 1	**GT**	*very laid back lodge*	–
	P1	perfect hill	0.508
	P2	unique hotel to spend a	0.498
	P3	great place to stay	**0.535**
	P4	top-notch l	0.418
Ex. 2	**GT**	*loved every minute of it*	–
	P1	would not recommend it at	0.463
	P2	were so impressed by this	0.566
	P3	enjoyed every minute of it	**0.961**
	P4	skied daily. Excell	0.461
Ex. 3	**GT**	*screen washer fluid are*	–
	P1	screen washer software are	**0.922**
	P2	screen wipers are changed	0.641
	P3	screen washing and dipping	0.555
	P4	screen washer fluids	0.824
Ex. 4	**GT**	*bound on the predictive*	–
	P1	bound on the trace of	**0.840**
	P2	-ranked trace-	0.535
	P3	bound on	0.594
	P4	bound on P(	0.582
Ex. 5	**GT**	*continues to grow; and*	–
	P1	reflects wave after wave of	0.484
	P2	stretches into space.	**0.531**
	P3	heats ocean waters a	0.461
	P4	cushions us all from	0.500
Ex. 6	**GT**	*is also convergent according*	–
	P1	is a convergent integral	**0.758**
	P2	shall not be less than	0.512
	P3	must have a value,	0.574
	P4	is convergent. Because	0.578