Foreign Sparse Attention: Effective Distillation into Sparse Attention

Vijaykaarti Sundarapandiyan [1] **Tom Goldstein** [1] **Ashwinee Panda** [1]

Abstract

Transformer architectures have been often maligned for the quadratic complexity of global self-attention, but global self-attention has proven critical for performance in many applications. Recently, reasoning models have pushed the limits of token generation, with models generating tens of thousands of tokens in their chain-of-thought for a single query. Now more than ever, efficient attention alternatives are critical. Native sparse attention is a promising recent alternative to global self-attention, but has not been validated at the scale of frontier pretrained model releases. In this work, we present Foreign Sparse Attention: an effective and efficient distillation method for transferring global self-attention into native sparse attention. We validate that our distilled Qwen model performs competitively with the teacher, in some instances improving in accuracy on data we did not distill on while generating fewer tokens in its responses.

1. Related Work

Knowledge distillation. Knowledge distillation (Hinton et al., 2015) equips a *student* model with the behaviour of a larger *teacher* by training the student on the teacher's outputs. These techniques have proved effective for reducing parameter counts and latency, but almost all prior work keeps teacher and student within the same architectural family—most often a quadratic Transformer—so the $\mathcal{O}(L^2)$ self-attention cost remains untouched. When the architecture is shared, matching internal tensors is straightforward; once the token-mixing operator changes, additional design choices are required to decide *what* should be aligned and *where* supervision should be applied.

[1]University of Maryland. Correspondence to: Vijaykaarti Sundarapandiyan <vsundar1@umd.edu>.

Proceedings of the 42nd International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

Cross-architecture attention distillation. Recent efforts move beyond compression and attempt to transplant global attention patterns into faster, sub-quadratic operators. The related methods attempt to map global attention components to sub-quadratic attention operator components and then train the sub-quadratic model on various teacher-forcing and end-to-end schemes. An overview of many current methods is detailed in Appendix C.

These studies expose three open challenges. First, alternative token-mixing mechanisms must retain the teacher's ability to capture long-range dependencies. Second, effective supervision should balance *local* (within-layer) and *global* (sequence-level) alignment rather than relying on only one. Third, the optimisation should remain token-efficient; updating all parameters concurrently enlarges the search space and increases data requirements.

2. Preliminaries

To motivate sparse attention mechanisms and our Foreign Sparse Attention distillation scheme, we describe Transformers and global attention. We then review existing sub-quadratic token mixing operations and Native Sparse Attention.

For the rest of this paper, a *block* refers to repeating components, chained together end-to-end, that compose a model. A *layer* refers to sequential components that compose each block. For example, a global attention component and a feed-forward component are two layers within a decoder block.

2.1. Subquadratic Token Mixing

State of the art decoder-only Transformers rely on multi-head global attention, which computes attention scores between every pair of tokens in a sequence of length L, incurring $\mathcal{O}(L^2)$ time and memory complexity as L grows. To mitigate this bottleneck, several sub-quadratic token mixing methods approximate attention while retaining expressivity. For example, *Linear attention* (Katharopoulos et al., 2020) uses kernel feature maps to approximate attention linearly, while *State Space Models* (Gu et al., 2022) model sequences as continuous-time linear dynamical systems.

2.2. Native Sparse Attention

Native Sparse Attention (NSA) (Yuan et al., 2025), released by DeepSeek, employs a trainable, hardware-aligned sparse mechanism that dynamically constructs a reduced set of key–value pairs per query by combining three complementary streams. First, the keys and values are compressed into blocks. Attention is performed in compressed space. Then, the top-n most important blocks as measured by attention scores are chosen, and fine-grained attention is performed amongst all tokens within chosen blocks. Finally, a sliding window component is added to capture local patterns. Attention streams are combined via a learned gating function.

NSA matches or exceeds full global attention on long-context benchmarks while operating in sub-quadratic time and significantly reducing memory footprint. In practice, its hardware-aligned sparse patterns translate into substantial inference speedups and lower resource consumption.

3. Methods

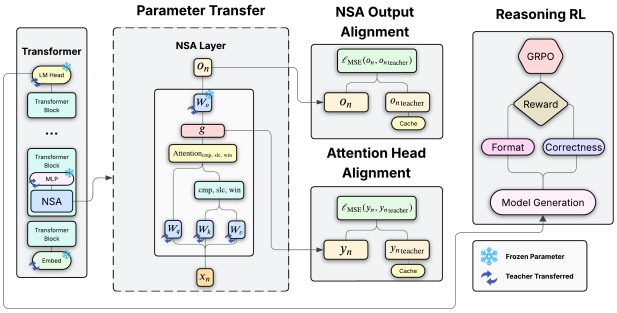


Figure 1: **Foreign Sparse Attention framework**. We distill a teacher transformer model using global attention into a student model using Native Sparse Attention by transferring all analogous parameters and freezing all but relevant attention parameters (Parameter Transfer), training the student model to match attention components of the teacher (NSA Output and Attention Head Alignment), and finally rectifying reasoning issues using GRPO (Reasoning RL).

In this section, we detail the Foreign Sparse Attention (FSA) distillation pipeline, comprising four stages (one initialization stage and three training stages): parameter transfer, attention head alignment, attention block alignment, and reasoning trace correction. These stages facilitate the effective transfer of capabilities from a global attention teacher model to a student model employing Native Sparse Attention (NSA). The resultant student model retains all teacher parameters except for the global attention layers, which are replaced by NSA layers.

3.1. Stage 0: Component Transfer

FSA starts with initializing the student model by direct parameter copying from the pre-trained teacher. This transfers learned knowledge, providing a robust initialization for attention distillation.

The process is:

1. **Non-Attention Parameters:** Non-attention parameters (e.g., token embeddings, layer normalizations, MLP weights) are copied from teacher to student and subsequently frozen throughout all distillation stages (1-3). This preserves the teacher's learned representations, focusing adaptation primarily on the attention mechanism.

2. **Attention Parameters:** Query ($\boldsymbol{W_Q}$), key ($\boldsymbol{W_K}$), value ($\boldsymbol{W_V}$), and output ($\boldsymbol{W_O}$) projection matrices are copied from the teacher's global attention layers to the corresponding student's NSA layers. NSA possesses no additional analogous parameters for direct transfer.

3. **Parameter Freezing within Attention:** The student's $\boldsymbol{W_O}$ matrices are also frozen for the entirety of our distillation scheme. This constrains the student to learn output representations consistent with the teacher's, despite differing token mixing strategies. Trainable parameters are thus restricted to internal NSA components (e.g. $\boldsymbol{W_Q}$, $\boldsymbol{W_K}$, $\boldsymbol{W_V}$, and the MLP φ).

This transfer and freeze narrows the parameter search space and adapts the teacher's knowledge. Further training now need only concentrate on adapting NSA to emulate teacher attention patterns.

3.2. Stage 1: Attention Head Alignment

This initial active distillation stage aligns internal attention representations at the head output level (pre-$\boldsymbol{W_O}$) via a block-wise training scheme.

Training Procedure:

1. **Layer Grouping:** Layers are partitioned into groups based on impact analysis; for instance, the final k layers might each form a distinct group, while earlier layers are consolidated into larger m-sized blocks. In practice, we find the earlier layers easier to distill and group them in large chunks, while we train deeper layers in isolation.

2. **Attention Bypass and Isolated Training:** Student layer groups are trained independently. Inputs are sourced via an "attention bypass," in which the first

layer in a layer group receives its input from the output of the corresponding previous layer in the teacher model. All other layers in the layer group receive their signal from the output of the previous layer in the layer group. This provides a fixed input distribution for each student group and mitigates error accumulation from preceding student layers.

3. **Loss Function:** The loss is the Mean Squared Error (MSE) between each student attention head's output (pre-$\boldsymbol{W_O}$) and the corresponding teacher head's output, averaged across all heads and layers within the group. For a given group g:

$$\mathcal{L}_{\text{Stage1}} = \frac{1}{N_L N_H} \sum_{l \in L_g} \sum_{h \in H_l} \text{MSE}(\boldsymbol{y}^{\text{head}}_{\text{student } l,h}, \boldsymbol{y}^{\text{head}}_{\text{teacher } l,h})$$

where L_g denotes layers in group g, H_l heads in layer l, and $\boldsymbol{y}^{\text{head}}$ individual head outputs.

4. **Parallelization:** Block-wise training with attention bypass facilitates parallelization, as teacher activations can be pre-computed and cached. Then, each layer group is trained together in parallel over the same tokens.

3.3. Stage 2: Attention Block Alignment

Stage 2 refines the distillation through end-to-end training of the student model. This allows for interaction between different NSA blocks and focuses on aligning the final output of each attention block (post-$\boldsymbol{W_O}$).

Training Procedure:

1. **End-to-End Architecture:** The student model is trained in its complete configuration, with inputs propagating through all its layers. Teacher model attention block outputs serve as targets.

2. **Loss Function:** The loss is the MSE between the student's attention block output (post-frozen $\boldsymbol{W_O}$) and the teacher's corresponding block output, averaged across all attention blocks:

$$\mathcal{L}_{\text{Stage2}} = \frac{1}{N_B} \sum_{b \in B} \text{MSE}(\boldsymbol{O}_{\text{student},b}, \boldsymbol{O}_{\text{teacher},b})$$

where B is the set of attention blocks and $\boldsymbol{O}$ is the block's final output.

This end-to-end alignment is crucial for learning complex reasoning patterns across multiple model layers. We also hypothesize that this stage enables the student to better leverage the null space of the $\boldsymbol{W_O}$ matrix in concordance with the teacher model.

3.4. Stage 3: Reasoning Trace Correction

The student model is generally performant after Stages 1 and 2, and the post-Stage 2 model performs well. However, the model may still exhibit minor logical fallacies or formatting inconsistencies (e.g., not adhering to the `\boxed{}` convention in mathematical reasoning) which makes it somewhat inconvenient to evaluate. In Stage 3, we use Reinforcement Learning from Verifiable Rewards (RLVR) to directly target formatting.

Generalized Reward Policy Optimization (GRPO): We utilize GRPO (Shao et al., 2024), a policy gradient method, to optimize model parameters by maximizing a task-specific reward signal. GRPO does not use a value network and instead computes the advantage as the z-score over a batch of rollouts per example. GRPO is a standard RLVR method, and because we are focusing our distillation efforts on reasoning, we choose GRPO for our final reasoning RLVR stage.

Training Procedure:

1. **Targeted Dataset:** Fine-tuning is performed on a small, specialized dataset, such as $\sim$100 samples from the GSM8K benchmark for mathematical reasoning tasks.

2. **Reward Function:** The reward function heavily penalizes incorrect final answers and incentivizes adherence to specific reasoning and output formats. Correct numerical answers with a complete reasoning trace producing an answer within the `\boxed{}` format receive the maximum reward. A small length penalty is applied to encourage the model to be succinct in its reasoning.

3. **Parameter Updates:** During RL, non-attention parameters and $\boldsymbol{W_O}$ matrices remain frozen; only internal NSA parameters are updated. Experimentally, unfreezing all parameters improved performance on the fine-tuning benchmark (GSM8K) but led to significant degradation on more challenging, out-of-distribution benchmarks (e.g., AIME). This suggests that broader unfreezing can induce catastrophic forgetting of the well-curated knowledge in MLP layers. This observation reinforces our selective freezing strategy.

This RL phase corrects minor reasoning and formatting issues within only a handful of gradient steps.

4. Empirical Validation

In this section, we present the empirical validation of our Foreign Sparse Attention (FSA) distillation method. We apply FSA to distill a native reasoner employing global

attention into a native reasoner employing Native Sparse Attention using 100M tokens, far fewer than the billions or trillions of tokens required to pre-train reasoning models.

Table 1: Three-stage FSA training pipeline.

Stage	Layers / groups	L	Notes
Stage 1	Group 1: Layers 0– 11	1150	easiest layers
	Group 2: Layers 12 – 24	1150	easy layers
	Group 3: Layer 25	3500	heavy layer
	Group 4: Layer 26	3500	heavier layer
	Group 5: Layer 27	3500	heaviest layer
Stage 2	End-to-end	1024	stitch layers
Stage 3	End-to-end	1024	reward correct

4.1. Experimental Setup

Teacher and Student models. We start from the publicly available DeepSeek-R1-Distill-Qwen-1.5B, a distillation of DeepSeek R1 (DeepSeek-AI et al., 2025), DeepSeek's flagship 671B parameter reasoning model into the much smaller Qwen2.5-Math-1.5B (Yang et al., 2024). This model serves as the teacher model, providing the ground truth for our distillation. We then surgically replace each global attention in the teacher model with Native Sparse Attention following (Yuan et al., 2025). This NSA enabled model serves as the student model. We then progressively apply each stage of FSA to distill the teacher model's attention patterns into the student model: 20k gradient steps for Stage 1 and Stage 2, and 100 gradient steps for Stage 3. See Table 1 for per-stage sequence length. We appropriately call the final distilled model DeepSeek-R1-Distill-QweNSA-1.5B. We refer to this distilled model as the student NSA model or simply QweNSA.

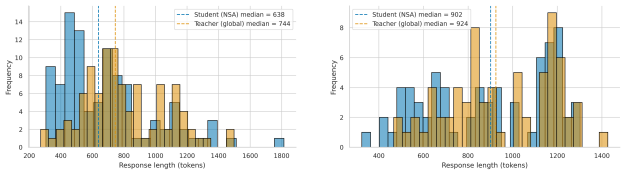
Dataset. All teacher-student distillation relies on the **DeepMath-103K** (He et al., 2025) corpus ($\sim$100M tokens of which we use), a carefully curated collection of formal mathematics that includes theorem statements, proof sketches, and informal commentary. The dataset provides the rich, long-range dependencies needed to exercise global attention and therefore serves as an ideal playground for distilling complex reasoning patterns into our student model. Stage 3 uses an additional 100 samples from GSM8k (Cobbe et al., 2021) to provide a reward signal that explicitly targets step-by-step numerical reasoning.

4.2. Final Results

We examine the performance of our distilled NSA model on downstream math, reasoning, and natural language tasks relative to the teacher model. We show that our distilled model outperforms the teacher model on a variety of reasoning tasks as shown in Table 2. Our model generates *fewer* tokens *faster* while being more accurate as seen in Figure 2. Interestingly, despite being distilled on a reasoning-intense math dataset, our NSA model is able to handle natural language tasks well. Notably, the student achieves a score of 27% on MMLU and 20% on MMLU-Pro, significantly outperforming the teacher's scores of 22% and 10%, respectively, despite being distilled using math-focused data.

Table 2: Final comparison between the original **teacher** (full global attention) and the distilled **student** (Native Sparse Attention). Higher is better on all, and all are zero shot scores.

Model	MMLU ↑	MMLU-Pro ↑	AIME ↑
Teacher (global)	0.220	0.100	$0.253_{0.032}$
Student (ours)	**0.270**	**0.200**	$0.240_{0.034}$



(a) MMLU (b) MMLU-Pro

Figure 2: Quantitative comparison of student and teacher models on MMLU and MMLU-Pro. Our NSA model generates fewer tokens than the global attention teacher model. The NSA student scores 27% on MMLU vs. the teacher's 22%, and 20% on MMLU-Pro vs. 10%.

5. Discussion

We propose FSA, a distillation method for efficiently turning a model trained with global attention into a model that utilizes NSA. We use FSA to distill a Qwen-1.5B into using NSA, and show that our model performs competitively with the teacher. Our QweNSA model uses fewer tokens to answer questions while either maintaining or improving accuracy, and it generates these tokens significantly faster than the teacher model. We believe that if our findings hold across model families and sizes, distillation to NSA may become a standard component of post-training pipelines.

Limitations. We report results with one model, distilled from one other model, and evaluated three datasets. We do not have access to a truly efficient CUDA kernel for NSA to benchmark its performance, so our understanding of our model speedups may be inaccurate. We urge the reader to check the supplementary material for further analyses.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

DeepSeek-AI, Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., Zhang, X., Yu, X., Wu, Y., Wu, Z. F., Gou, Z., Shao, Z., Li, Z., Gao, Z., Liu, A., Xue, B., Wang, B., Wu, B., Feng, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., Dai, D., Chen, D., Ji, D., Li, E., Lin, F., Dai, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Bao, H., Xu, H., Wang, H., Ding, H., Xin, H., Gao, H., Qu, H., Li, H., Guo, J., Li, J., Wang, J., Chen, J., Yuan, J., Qiu, J., Li, J., Cai, J. L., Ni, J., Liang, J., Chen, J., Dong, K., Hu, K., Gao, K., Guan, K., Huang, K., Yu, K., Wang, L., Zhang, L., Zhao, L., Wang, L., Zhang, L., Xu, L., Xia, L., Zhang, M., Zhang, M., Tang, M., Li, M., Wang, M., Li, M., Tian, N., Huang, P., Zhang, P., Wang, Q., Chen, Q., Du, Q., Ge, R., Zhang, R., Pan, R., Wang, R., Chen, R. J., Jin, R. L., Chen, R., Lu, S., Zhou, S., Chen, S., Ye, S., Wang, S., Yu, S., Zhou, S., Pan, S., Li, S. S., Zhou, S., Wu, S., Ye, S., Yun, T., Pei, T., Sun, T., Wang, T., Zeng, W., Zhao, W., Liu, W., Liang, W., Gao, W., Yu, W., Zhang, W., Xiao, W. L., An, W., Liu, X., Wang, X., Chen, X., Nie, X., Cheng, X., Liu, X., Xie, X., Liu, X., Yang, X., Li, X., Su, X., Lin, X., Li, X. Q., Jin, X., Shen, X., Chen, X., Sun, X., Wang, X., Song, X., Zhou, X., Wang, X., Shan, X., Li, Y. K., Wang, Y. Q., Wei, Y. X., Zhang, Y., Xu, Y., Li, Y., Zhao, Y., Sun, Y., Wang, Y., Yu, Y., Zhang, Y., Shi, Y., Xiong, Y., He, Y., Piao, Y., Wang, Y., Tan, Y., Ma, Y., Liu, Y., Guo, Y., Ou, Y., Wang, Y., Gong, Y., Zou, Y., He, Y., Xiong, Y., Luo, Y., You, Y., Liu, Y., Zhou, Y., Zhu, Y. X., Xu, Y., Huang, Y., Li, Y., Zheng, Y., Zhu, Y., Ma, Y., Tang, Y., Zha, Y., Yan, Y., Ren, Z. Z., Ren, Z., Sha, Z., Fu, Z., Xu, Z., Xie, Z., Zhang, Z., Hao, Z., Ma, Z., Yan, Z., Wu, Z., Gu, Z., Zhu, Z., Liu, Z., Li, Z., Xie, Z., Song, Z., Pan, Z., Huang, Z., Xu, Z., Zhang, Z., and Zhang, Z. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Goldstein, D., Alcaide, E., Lu, J., and Cheah, E. Radlads: Rapid attention distillation to linear attention decoders at scale, 2025. URL https://arxiv.org/abs/2505.03005.

Gu, A., Goel, K., and Ré, C. Efficiently modeling long sequences with structured state spaces, 2022. URL https://arxiv.org/abs/2111.00396.

He, Z., Liang, T., Xu, J., Liu, Q., Chen, X., Wang, Y., Song, L., Yu, D., Liang, Z., Wang, W., Zhang, Z., Wang, R., Tu, Z., Mi, H., and Yu, D. Deepmath-103k: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning, 2025. URL https://arxiv.org/abs/2504.11456.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531.

Hu, S., Tu, Y., Han, X., He, C., Cui, G., Long, X., Zheng, Z., Fang, Y., Huang, Y., Zhao, W., Zhang, X., Thai, Z. L., Zhang, K., Wang, C., Yao, Y., Zhao, C., Zhou, J., Cai, J., Zhai, Z., Ding, N., Jia, C., Zeng, G., Li, D., Liu, Z., and Sun, M. Minicpm: Unveiling the potential of small language models with scalable training strategies, 2024. URL https://arxiv.org/abs/2404.06395.

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. Transformers are rnns: Fast autoregressive transformers with linear attention, 2020. URL https://arxiv.org/abs/2006.16236.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y. K., Wu, Y., and Guo, D. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Yang, A., Zhang, B., Hui, B., Gao, B., Yu, B., Li, C., Liu, D., Tu, J., Zhou, J., Lin, J., Lu, K., Xue, M., Lin, R., Liu, T., Ren, X., and Zhang, Z. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement, 2024. URL https://arxiv.org/abs/2409.12122.

Yuan, J., Gao, H., Dai, D., Luo, J., Zhao, L., Zhang, Z., Xie, Z., Wei, Y. X., Wang, L., Xiao, Z., Wang, Y., Ruan, C., Zhang, M., Liang, W., and Zeng, W. Native sparse attention: Hardware-aligned and natively trainable sparse attention, 2025. URL https://arxiv.org/abs/2502.11089.

A. Transformers and Attention

State-of-the-art decoder-only Transformer models utilize stacked layers, each comprising a multi-head self-attention module followed by feed-forward layers. For a single attention head operating on an input sequence $\boldsymbol{X} \in \mathbb{R}^{L \times d}$ (where L is the sequence length and d is the model dimension), learned linear transformations are applied to obtain query, key, and value matrices: $\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W_Q}$, $\boldsymbol{K} = \boldsymbol{X}\boldsymbol{W_K}$, and $\boldsymbol{V} = \boldsymbol{X}\boldsymbol{W_V}$. The projection matrices are $\boldsymbol{W_Q}, \boldsymbol{W_K}, \boldsymbol{W_V} \in \mathbb{R}^{d \times d}$, and the block output is processed by $\boldsymbol{W_O} \in \mathbb{R}^{d \times d}$.

The level of "attention" or relevance between the n-th query vector $\boldsymbol{q_n}$ and the i-th key vector $\boldsymbol{k_i}$ is determined by their scaled dot product. These scores are then normalized across all possible keys for a given query using softmax to produce attention weights $a_{n,i}$:

$$a_{n,i} = \frac{\exp\left(\boldsymbol{q_n}^\top \boldsymbol{k_i} / \sqrt{d}\right)}{\sum_{j=1}^{L} \exp\left(\boldsymbol{q_n}^\top \boldsymbol{k_j} / \sqrt{d}\right)}.$$

The resulting output vector for position n, $\boldsymbol{y_n}$, is computed as a weighted sum of the value vectors $\boldsymbol{v_i}$, with weights given by the attention scores:

$$\boldsymbol{y_n} = \sum_{i=1}^{L} a_{n,i}\, \boldsymbol{v_i}.$$

Multi-head attention involves performing this scaled dot-product attention computation in parallel for h different sets of learned query, key, and value projections. The resulting output vectors from each head are then concatenated and linearly projected to produce the final output of the multi-head attention layer.

B. Native Sparse Attention (NSA)

B.1. NSA Streams

1. **Compressed Coarse-Grained Attention:**
   Aggregate tokens into blocks of length l via a learnable MLP φ, producing compressed keys $\tilde{\boldsymbol{k}}_i^{\text{comp}} \in \mathbb{R}^d$ and values $\tilde{\boldsymbol{v}}_i^{\text{comp}}$.

$$a_{n,i}^{\text{comp}} = \frac{\exp\left(\boldsymbol{q}_n^\top \tilde{\boldsymbol{k}}_i^{\text{comp}} / \sqrt{d}\right)}{\sum_j \exp\left(\boldsymbol{q}_n^\top \tilde{\boldsymbol{k}}_j^{\text{comp}} / \sqrt{d}\right)}$$
$$\boldsymbol{y}_n^{\text{comp}} = \sum_i a_{n,i}^{\text{comp}}\, \tilde{\boldsymbol{v}}_i^{\text{comp}}$$

2. **Selectively Retained Fine-Grained Attention:**
   Select the top-n blocks via compressed scores and attend to all original tokens within those blocks. Let $\mathcal{S}_n$ be the selected indices.

$$a_{n,i}^{\text{select}} = \frac{\exp\left(\boldsymbol{q}_n^\top \boldsymbol{k}_i / \sqrt{d}\right)}{\sum_{j \in \mathcal{S}_n} \exp\left(\boldsymbol{q}_n^\top \boldsymbol{k}_j / \sqrt{d}\right)}$$
$$\boldsymbol{y}_n^{\text{select}} = \sum_{i \in \mathcal{S}_n} a_{n,i}^{\text{select}}\, \boldsymbol{v}_i$$

3. **Sliding Window Attention:**
   Attend within a local window of size w around each query.

$$a_{n,i}^{\text{window}} = \frac{\exp\left(\boldsymbol{q}_n^\top \boldsymbol{k}_i / \sqrt{d}\right)}{\sum_{j=n-w}^{n-1} \exp\left(\boldsymbol{q}_n^\top \boldsymbol{k}_j / \sqrt{d}\right)}$$
$$\boldsymbol{y}_n^{\text{window}} = \sum_{i=n-w}^{n-1} a_{n,i}^{\text{window}}\, \boldsymbol{v}_i$$

Let $C = \{\text{comp}, \text{select}, \text{window}\}$. The streams are then combined via a learnable gating function as

$$\boldsymbol{y}_n = \boldsymbol{W_O} \sum_{c \in C} g_n^c\, \boldsymbol{y}_n^c.$$

B.2. NSA Speedup

Yuan et al. (2025) claim order of magnitude speedups (a $11.6\times$ speedup on decode, a $9\times$ speedup on forward pass and a $6\times$ speedup on backward pass over global attention) for long sequence lengths. However, they do not release their implementation. In Figure 3 we benchmark the speeds of NSA and FlashAttention with open-source kernels[1]. Although the speedups are only significant for very long sequence lengths, we emphasize that our model, which is small, can already generate sequences this long on AIME.

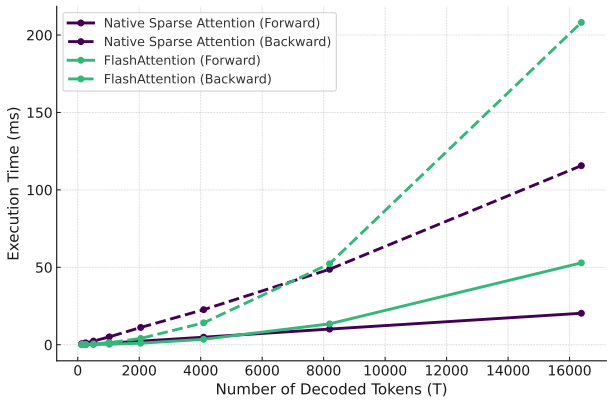


Figure 3: **NSA Speedup.** We plot the time in milliseconds to decode the Nth token in a sequence.

[1]GitHub url: https://github.com/fla-org/native-sparse-attention

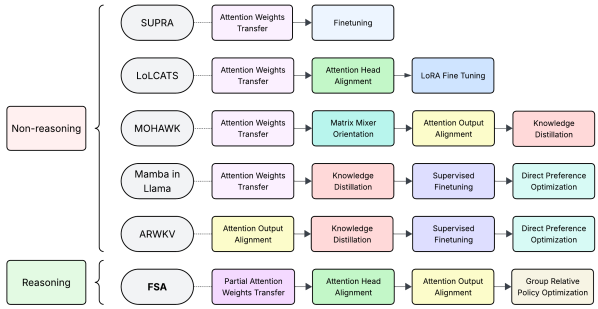


Figure 4: **Related Attention Distillation Schemes.** Existing methods distill global attention into sub-quadratic token–mixing operators using a variety of component-alignment and end-to-end objectives (Goldstein et al., 2025). Foreign Sparse Attention combines block-wise and global supervision to transfer global attention into Native Sparse Attention layers while preserving reasoning ability.

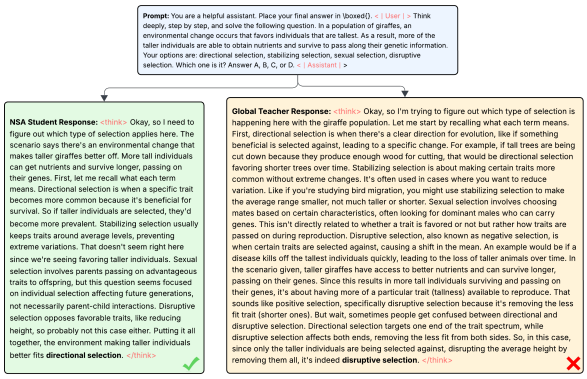


Figure 5: **Teacher vs Student Reasoning Traces.** Qualitative reasoning trace comparison between student and teacher on an MMLU task. The NSA student model reasons more succinctly and correctly. The teacher model's response contains filler and flawed heuristics, leading to an incorrect answer.

C. Cross-architecture Distillation Schemes

Various current cross-architecture distillation schemes, with their discrete distillation steps, are described in Figure 4.

D. Segmented Training

D.1. Block-wise Training

Block-wise training is motivated by the observation that later layers in LLMs are critical for complex reasoning and exhibit more intricate attention patterns. Accumulated error from previous student layers in full-layer training causes these deeper layers to receive noisier input signals. Combined with the "heavy-lifting" nature of these layers, we find it difficult to effectively distill them. However, isolating these "heavy-lifting" layers for focused training allows us to provide these layers with a perfect input signal from the teacher and thereby enables distillation. In addition, grouping earlier, "easier" layers, optimizes distillation token-efficiency and reduces discrepancies caused by isolated layer training.

D.2. End-to-End Training

While block-wise training ensures local fidelity, training all attention layers in tandem promotes global consistency by "stitching together" these independently initialized components to recover any global patterns block-wise training would miss. In this sense, Stage 1 serves to accelerate convergence in this phase by providing a strong initial parameterization, and Stage 2 completes the teacher-distillation process by ensuring global consistency.

E. Training Recipe Hyperparameters

We set the parameters for NSA as follows: NSA compression block size $\ell = 32$, sliding stride $d = 16$, selected block size $\ell' = 64$, selected block count $n = 16$, and sliding window size $w = 1024$. These parameters follow (Yuan et al., 2025).

Our FSA recipe comprises one static initialisation stage (Stage 0) followed by three training stages (Stages 1, 2, 3). Throughout training we employ the standard ADAM (Kingma & Ba, 2017) optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$) coupled with a warm-up $\rightarrow$ stable $\rightarrow$ decay (WSD) schedule (Hu et al., 2024). All other hyperparameters are stage-specific:

- **Stages 1 – 2 (head & block MSE alignment).** We train for 20,000 optimizer steps with 50 warm-up and 50 decay steps, a learning rate of 1×10^{-4}, and a batch size of 1.

- **Stage 3 (GRPO tuning).** We train for 100 optimizer steps with 10 warm-up and 10 decay steps, a learning rate of 2×10^{-5}, four roll-outs per prompt, and a KL-penalty coefficient $\beta_{\mathrm{KL}} = 0.04$.

The sequence length L varies only within Stage 1. Early layers are trained with $L = 1150$, whereas the three heaviest layers—responsible for the bulk of the reasoning workload—use $L = 3500$. All other training stages use $L = 1024$. A concise overview is given in Table 1.

F. Sample Traces

Qualitatively, our NSA model generates traces that are shorter, more information dense, and correct when compared to the teacher model as seen in Figure 5. We observe this conciseness in the student model even when both models correctly answer the question.

G. Implementation and Open-source

The efficiency of our method enables us to conduct all experiments by spending just a few hundred dollars on cloud credits for H100s. We open-source our model, which is available anonymously at doubleblind/DeepSeek-R1-Distill-QweNSA-1.5B