DeepSeek-V3.2-Exp: Boosting Long-Context Efficiency with DeepSeek Sparse Attention

DeepSeek-AI

research@deepseek.com

Abstract

We introduce DeepSeek-V3.2-Exp, an experimental sparse-attention model, which equips DeepSeek-V3.1-Terminus with DeepSeek Sparse Attention (DSA) through continued training. With DSA, a fine-grained sparse attention mechanism powered by a lightning indexer, DeepSeek-V3.2-Exp achieves significant efficiency improvements in both training and inference, especially in long-context scenarios. The model checkpoints are available at https://huggingface.co/deepseek-ai/DeepSeek-V3.2-Exp.

1. Architecture

Compared with DeepSeek-V3.1-Terminus, the last version of DeepSeek-V3.1, the only architectural modification of DeepSeek-V3.2-Exp is the introduction of DeepSeek Sparse Attention (DSA) through continued training.

Prototype of DSA. The prototype of DSA primarily consists of two components: a lightning indexer and a fine-grained token selection mechanism.

The **lightning indexer** computes the index score $I_{t,s}$ between the query token $\mathbf{h}_t \in \mathbb{R}^d$ and a preceding token $\mathbf{h}_s \in \mathbb{R}^d$, determining which tokens to be selected by the query token:

$$I_{t,s} = \sum_{j=1}^{H^I} w_{t,j}^I \cdot \mathrm{ReLU}\left(\mathbf{q}_{t,j}^I \cdot \mathbf{k}_s^I\right), \tag{1}$$

where H^I denotes the number of indexer heads; $\mathbf{q}_{t,j}^I \in \mathbb{R}^{d^I}$ and $w_{t,j}^I \in \mathbb{R}$ are derived from the query token $\mathbf{h}_t$; and $\mathbf{k}_s^I \in \mathbb{R}^{d^I}$ is derived from the preceding token $\mathbf{h}_s$. We choose ReLU as the activation function for throughput consideration. Given that the lightning indexer has a small number of heads and can be implemented in FP8, its computational efficiency is remarkable.

Given the index scores $\{I_{t,s}\}$ for each query token $\mathbf{h}_t$, our **fine-grained token selection mechanism** retrieves only the key-value entries $\{\mathbf{c}_s\}$ corresponding to the top-k index scores. Then, the attention output $\mathbf{u}_t$ is computed by applying the attention mechanism between the query token $\mathbf{h}_t$ and the sparsely selected key-value entries $\{\mathbf{c}_s\}$:

$$\mathbf{u}_t = \mathrm{Attn}\big(\mathbf{h}_t, \left\{\mathbf{c}_s \,\middle|\, I_{t,s} \in \mathrm{Top\text{-}k}(I_{t,:})\right\}\big). \tag{2}$$

deepseek

DeepSeek-V3.2-Exp: Boosting Long-Context Efficiency with DeepSeek Sparse Attention

DeepSeek-AI

research@deepseek.com

Abstract

We introduce DeepSeek-V3.2-Exp, an experimental sparse-attention model, which equips DeepSeek-V3.1-Terminus with DeepSeek Sparse Attention (DSA) through continued training. With DSA, a fine-grained sparse attention mechanism powered by a lightning indexer, DeepSeek-V3.2-Exp achieves significant efficiency improvements in both training and inference, especially in long-context scenarios. The model checkpoints are available at https://huggingface.co/deepseek-ai/DeepSeek-V3.2-Exp.

1. Architecture

Compared with DeepSeek-V3.1-Terminus, the last version of DeepSeek-V3.1, the only architectural modification of DeepSeek-V3.2-Exp is the introduction of DeepSeek Sparse Attention (DSA) through continued training.

2. Training

Starting from a base checkpoint of DeepSeek-V3.1-Terminus, whose context length has been extended to 128K, we perform continued pre-training followed by post-training to create DeepSeek-V3.2-Exp.

2.1. Continued Pre-Training

The continued pre-training of DeepSeek-V3.2-Exp consists of two training stages. For both stages, the distribution of training data is totally aligned with the 128K long context extension data used for DeepSeek-V3.1-Terminus.

Dense Warm-up Stage. We first use a short warm-up stage to initialize the lightning indexer. In this stage, we keep dense attention and freeze all model parameters except for the lightning indexer. To align the indexer outputs with the main attention distribution, for the t-th query token, we first aggregate the main attention scores by summing across all attention heads.

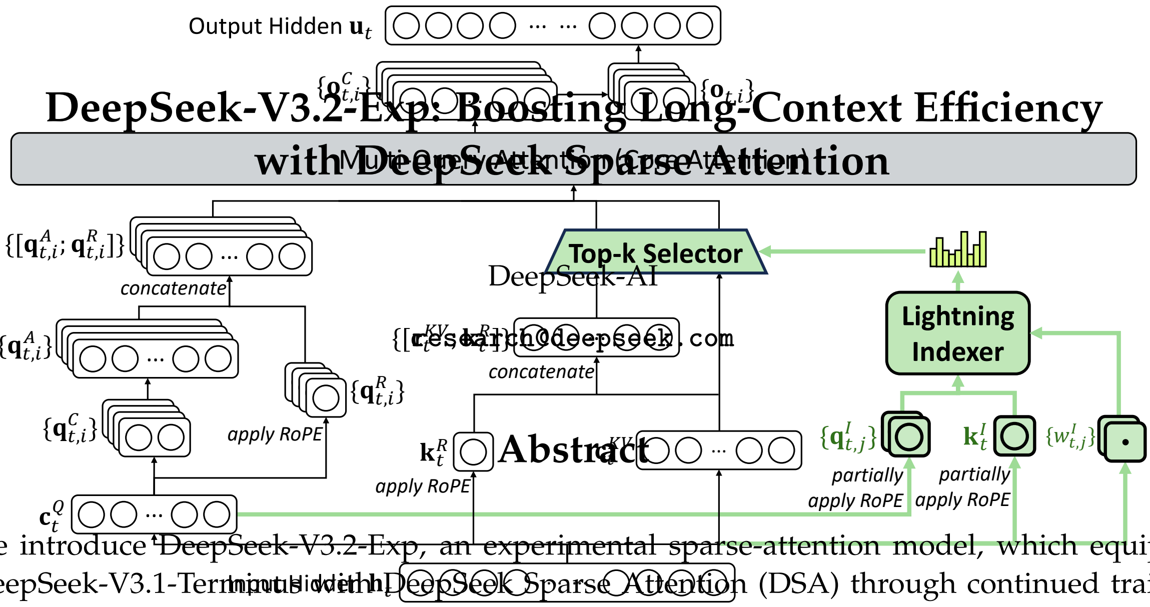


Figure 1 | Attention architecture of DeepSeek-V3.2-Exp, where DSA is instantiated under MLA. The green part illustrates how DSA selects the top-k key-value entries according to the indexer.

Instantiate DSA Under MLA. For the consideration of continued training from DeepSeek-V3.1-Terminus, we instantiate DSA based on MLA (DeepSeek-AI, 2024) for DeepSeek-V3.2-Exp. At the kernel level, each key-value entry must be shared across multiple queries for computational efficiency (Yuan et al., 2025). Therefore, we implement DSA based on the MQA (Shazeer, 2019) mode of MLA[1], where each latent vector (the key-value entry of MLA) will be shared across all query heads of the query token. The DSA architecture based on MLA is illustrated in Figure 1. We also provide an open-source implementation of DeepSeek-V3.2-Exp[2] to specify the details unambiguously.

Prototype of DSA. The prototype of DSA primarily consists of two components: a lightning indexer and a fine-grained token selection mechanism.

The **lightning indexer** computes the index score $I_{t,s}$ between the query token $\mathbf{h}_t \in \mathbb{R}^d$ and a preceding token $\mathbf{h}_s \in \mathbb{R}^d$, determining which tokens to be selected by the query token:

$$I_{t,s} = \sum_{j=1}^{H^I} w^I_{t,j} \cdot \text{ReLU}\left(\mathbf{q}^I_{t,j} \cdot \mathbf{k}^I_s\right), \tag{1}$$

where H^I denotes the number of indexer heads; $\mathbf{q}^I_{t,j} \in \mathbb{R}^{d^I}$ and $w^I_{t,j} \in \mathbb{R}$ are derived from the query token $\mathbf{h}_t$; and $\mathbf{k}^I_s \in \mathbb{R}^{d^I}$ is derived from the preceding token $\mathbf{h}_s$. We choose ReLU as the activation function for throughput consideration. Given that the lightning indexer has a small number of heads and can be implemented in FP8, its computational efficiency is remarkable.

Given the index score $\{I_{t,s}\}$ for each query token $\mathbf{h}_t$, our **fine-grained token selection mechanism** retrieves only the key-value entries $\{\mathbf{c}_s\}$ corresponding to the top-k index scores. Then, the attention output $\mathbf{u}_t$ is computed by applying the attention mechanism between the query token $\mathbf{h}_t$ and the sparsely selected key-value entries $\{\mathbf{c}_s\}$:

$$\mathbf{u}_t = \text{Attn}\left(\mathbf{h}_t, \left\{\mathbf{c}_s \mid I_{t,s} \in \text{Top-k}(I_{t,:})\right\}\right). \tag{2}$$

[1]We illustrate the difference between the MQA and MHA modes of MLA in Appendix A.

[2]https://huggingface.co/deepseek-ai/DeepSeek-V3.2-Exp/tree/main/inference

DeepSeek-V3.2-Exp: Boosting Long-Context Efficiency with DeepSeek Sparse Attention

DeepSeek-AI

research@deepseek.com

Abstract

We introduce DeepSeek-V3.2-Exp, an experimental sparse attention model, which equips DeepSeek-V3.1-Terminus with DeepSeek Sparse Attention (DSA) through continued training. With DSA, a fine-grained sparse attention mechanism powered by a lightning indexer, DeepSeek-V3.2-Exp achieves significant efficiency improvements in both training and inference, especially in long-context scenarios. The model checkpoints are available at https://huggingface.co/deepseek-ai/DeepSeek-V3.2-Exp.

1. Architecture

Compared with DeepSeek-V3.1-Terminus, the last version of DeepSeek-V3.1, the only architectural modification of DeepSeek-V3.2-Exp is the introduction of DeepSeek Sparse Attention (DSA) through continued training.

Prototype of DSA. The prototype of DSA primarily consists of two components: a lightning indexer and a fine-grained token selection mechanism.

The **lightning indexer** computes the index score $I_{t,s}$ between the query token $\mathbf{h}_t \in \mathbb{R}^d$ and a preceding token $\mathbf{h}_s \in \mathbb{R}^d$, determining which tokens to be selected by the query token:

$$I_{t,s} = \sum_{j=1}^{H^I} w^I_{t,j} \cdot \text{ReLU}\left(\mathbf{q}^I_{t,j} \cdot \mathbf{k}^I_s\right), \tag{1}$$

where H^I denotes the number of indexer heads; $\mathbf{q}^I_{t,j} \in \mathbb{R}^{d^I}$ and $w^I_{t,j} \in \mathbb{R}$ are derived from the query token $\mathbf{h}_t$; and $\mathbf{k}^I_s \in \mathbb{R}^{d^I}$ is derived from the preceding token $\mathbf{h}_s$. We choose ReLU as the activation function for throughput consideration. Given that the lightning indexer has a small number of heads and can be implemented in FP8, its computational efficiency is remarkable.

Given the index scores $\{I_{t,s}\}$ for each query token $\mathbf{h}_t$, our **fine-grained token selection mechanism** retrieves only the key-value entries $\{\mathbf{c}_s\}$ corresponding to the top-k index scores. Then, the attention output $\mathbf{u}_t$ is computed by applying the attention mechanism between the query token $\mathbf{h}_t$ and the sparsely selected key-value entries $\{\mathbf{c}_s\}$:

$$\mathbf{u}_t = \text{Attn}\left(\mathbf{h}_t, \left\{\mathbf{c}_s \,\middle|\, I_{t,s} \in \text{Top-k}(I_{t,:})\right\}\right). \tag{2}$$

This sum is then L1-normalized along the sequence dimension to produce a target distribution $p_{t,:} \in \mathbb{R}^t$. Based on $p_{t,:}$, we set a KL-divergence loss as the training objective of the indexer:

$$\mathcal{L}^I = \sum_t \mathbb{D}_{\text{KL}}\left(p_{t,:} \,\|\, \text{Softmax}(I_{t,:})\right). \tag{3}$$

For warm-up, we use a learning rate of 10^{-3}. We train the indexer for only 1000 steps, with each step consisting of 16 sequences of 128K tokens, resulting in a total of 2.1B tokens.

Sparse Training Stage. Following indexer warm-up, we introduce the fine-grained token selection mechanism and optimize all model parameters to adapt the model to the sparse pattern of DSA. In this stage, we also keep aligning the indexer outputs to the main attention distribution, but considering only the selected token set $\mathcal{S}_t = \{s \,|\, I_{t,s} \in \text{Top-k}(I_{t,:})\}$:

$$\mathcal{L}^I = \sum_t \mathbb{D}_{\text{KL}}\left(p_{t,\mathcal{S}_t} \,\|\, \text{Softmax}(I_{t,\mathcal{S}_t})\right). \tag{4}$$

It is worth noting that we detach the indexer input from the computational graph for separate optimization. The training signal of the indexer is from only $\mathcal{L}^I$, while the optimization of the main model is according to only the language modeling loss. In this sparse training stage, we use a learning rate of 7.3×10^{-6}, and select 2048 key-value tokens for each query token. We train both the main model and the indexer for 15000 steps, with each step consisting of 480 sequences of 128K tokens, resulting in a total of 943.7B tokens.

2.2. Post-Training

After continued pre-training, we perform post-training to create the final DeepSeek-V3.2-Exp. The post-training of DeepSeek-V3.2-Exp also employs sparse attention in the same way as the sparse continued pre-training stage. In pursuit of a rigorous assessment of the impact of introducing DSA, for DeepSeek-V3.2-Exp, we maintain the same post-training pipeline, algorithm, and data as used for DeepSeek-V3.1-Terminus, which are detailed as follows.

Specialist Distillation. For each task, we initially develop a specialized model dedicated exclusively to that particular domain, with all specialist models being fine-tuned from the same pre-trained DeepSeek-V3.2 base checkpoint. In addition to writing tasks and general question-answering, our framework encompasses five specialized domains: mathematics, competitive programming, general logical reasoning, agentic coding, and agentic search. Each specialist is trained with large-scale Reinforcement Learning (RL) computing. Furthermore, we employ different models to generate training data for long chain-of-thought reasoning (thinking mode) and direct response generation (non-thinking mode). Once the specialist models are prepared, they are used to produce the domain-specific data for the final checkpoint. Experimental results demonstrate that models trained on the distilled data achieve performance levels only marginally below those of domain-specific specialists, with the performance gap being effectively eliminated through subsequent RL training.

Mixed RL Training. For DeepSeek-V3.2-Exp, we still adopt Group Relative Policy Optimization (GRPO) (DeepSeek-AI, 2025; Shao et al., 2024) as the RL training algorithm. Unlike in previous DeepSeek models, which are trained with multi-stage reinforcement learning, we merge reasoning, agent, and human alignment training into one RL stage. This approach effectively balances performance across diverse domains while circumventing the catastrophic forgetting issues commonly associated with multi-stage training paradigms. For reasoning and

	Benchmark (Metric)	DeepSeek-V3.1-Terminus	DeepSeek-V3.2-Exp
General	MMLU-Pro (EM)	85.0	85.0
	GPQA-Diamond (Pass@1)	80.7	79.9
	Humanity's Last Exam (Pass@1)	21.7	19.8
Search Agent	BrowseComp (Acc.)	38.5	40.1
	BrowseComp_zh (Acc.)	45.0	47.9
	SimpleQA (Acc.)	96.8	97.1
Code	LiveCodeBench (2408-2505) (Pass@1)	74.9	74.1
	Codeforces-Div1 (Rating)	2046	2121
	Aider-Polyglot (Acc.)	76.1	74.5
Code Agent	SWE Verified (Agent mode)	68.4	67.8
	SWE-bench Multilingual (Agent mode)	57.8	57.9
	Terminal-bench (Terminus 1 framework)	36.7	37.7
Math	AIME 2025 (Pass@1)	88.4	89.3
	HMMT 2025 (Pass@1)	86.1	83.6

Table 1 | Evaluations of DeepSeek-V3.1-Terminus and DeepSeek-V3.2-Exp. Overall, DeepSeek-V3.2-Exp does not show substantial performance degradation compared with DeepSeek-V3.1-Terminus. The performance of DeepSeek-V3.2-Exp on GPQA, HLE, and HMMT 2025 is lower than that of DeepSeek-V3.1-Terminus because DeepSeek-V3.2-Exp generates fewer reasoning tokens. However, this performance gap closes when using intermediate checkpoints that produce a comparable number of tokens.

agent tasks, we employ rule-based outcome reward, length penalty, and language consistency reward. For general tasks, we employ a generative reward model where each prompt has its own rubrics for evaluation. Our reward design carefully balances two key trade-offs: (1) length versus accuracy and (2) language consistency versus accuracy.

3. Evaluations

Model Capabilities. We evaluate DeepSeek-V3.2-Exp on a suite of benchmarks, which focus on diverse capabilities, and compare it with DeepSeek-V3.1-Terminus in Table 1. While DeepSeek-V3.2-Exp significantly improves computational efficiency on long sequences, we do not observe substantial performance degradation compared with DeepSeek-V3.1-Terminus, on both short- and long-context tasks. In addition, we also compare the reinforcement learning training curves of DeepSeek-V3.2-Exp and DeepSeek-V3.1-Terminus, as shown in Figure 2. The performance of both models on BrowseComp and SWE Verified improves steadily throughout the training process, with closely aligned curves, which reflects the training stability of DSA.

Inference Costs. DSA reduces the core attention complexity of the main model from $O(L^2)$ to $O(Lk)$, where k ($\ll L$) is the number of selected tokens. Although the lightning indexer still has a complexity of $O(L^2)$, it requires much less computation compared with MLA in DeepSeek-V3.1. Combined with our optimized implementation, DSA achieves a significant end-to-end speedup in long-context scenarios. Figure 3 presents how token costs of DeepSeek-V3.1-Terminus and DeepSeek-V3.2-Exp vary with the token position in the sequence. These costs are estimated from benchmarking the actual service deployed on H800 GPUs, at

DeepSeek-V3.2-Exp: Boosting Long-Context Efficiency with DeepSeek Sparse Attention

DeepSeek-AI

research@deepseek.com

Abstract

We introduce DeepSeek-V3.2-Exp, an experimental sparse-attention model, which equips DeepSeek-V3.1-Terminus with DeepSeek Sparse Attention (DSA) through continued training. With DSA, a fine-grained sparse attention mechanism powered by a lightning indexer, DeepSeek-V3.2-Exp achieves significant efficiency improvements in both training and inference, especially in long-context scenarios. The model checkpoints are available at https://huggingface.co/deepseek-ai/DeepSeek-V3.2-Exp.

1. Architecture

Compared with DeepSeek-V3.1-Terminus, the last version of DeepSeek-V3.1, the only architectural modification of DeepSeek-V3.2-Exp is the introduction of DeepSeek Sparse Attention (DSA) through continued training.

Prototype of DSA. The prototype of DSA primarily consists of two components: a lightning indexer and a fine-grained token selection mechanism.

The **lightning indexer** computes the index score $I_{t,s}$ between the query token $\mathbf{h}_t \in \mathbb{R}^d$ and a preceding token $\mathbf{h}_s \in \mathbb{R}^d$, determining which tokens to be selected by the query token:

$$I_{t,s} = \sum_{j=1}^{H^I} w^I_{t,j} \cdot \text{ReLU}\left(\mathbf{q}^I_{t,j} \cdot \mathbf{k}^I_s\right), \tag{1}$$

where H^I denotes the number of indexer heads; $\mathbf{q}^I_{t,j} \in \mathbb{R}^{d^I}$ and $w^I_{t,j} \in \mathbb{R}$ are derived from the query token $\mathbf{h}_t$; and $\mathbf{k}^I_s \in \mathbb{R}^{d^I}$ is derived from the preceding token $\mathbf{h}_s$. We choose ReLU as the activation function for throughput consideration. Given that the lightning indexer has a small number of heads and can be implemented in FP8, its computational efficiency is remarkable.

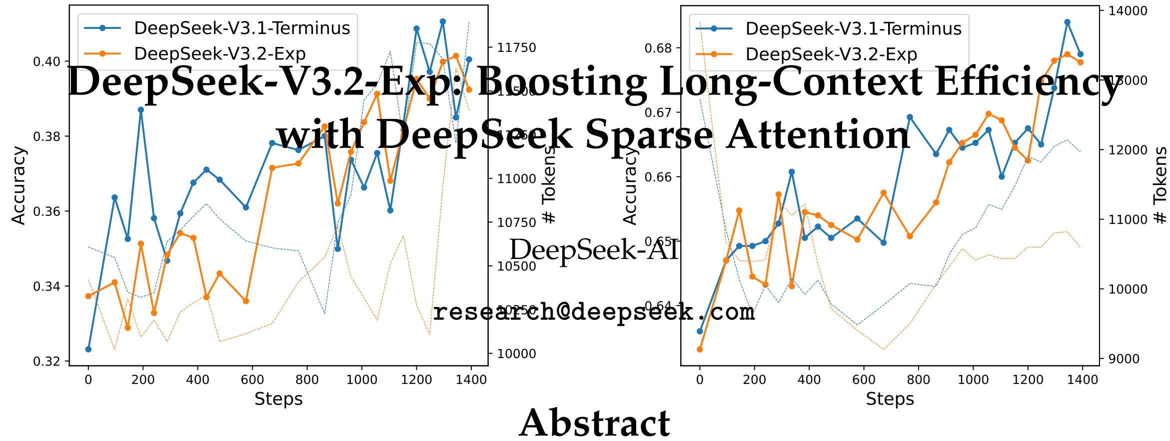
Given the index scores $\{I_{t,s}\}$ for each query token $\mathbf{h}_t$, our **fine-grained token selection mechanism** retrieves only the key-value entries $\{\mathbf{c}_s\}$ corresponding to the top-k index scores. Then, the attention output $\mathbf{u}_t$ is computed by applying the attention mechanism between the query token $\mathbf{h}_t$ and the sparsely selected key-value entries $\{\mathbf{c}_s\}$:

$$\mathbf{u}_t = \text{Attn}\left(\mathbf{h}_t, \left\{\mathbf{c}_s \mid I_{t,s} \in \text{Top-k}(I_{t,:})\right\}\right). \tag{2}$$

DeepSeek-V3.2-Exp: Boosting Long-Context Efficiency with DeepSeek Sparse Attention

DeepSeek-AI

research@deepseek.com

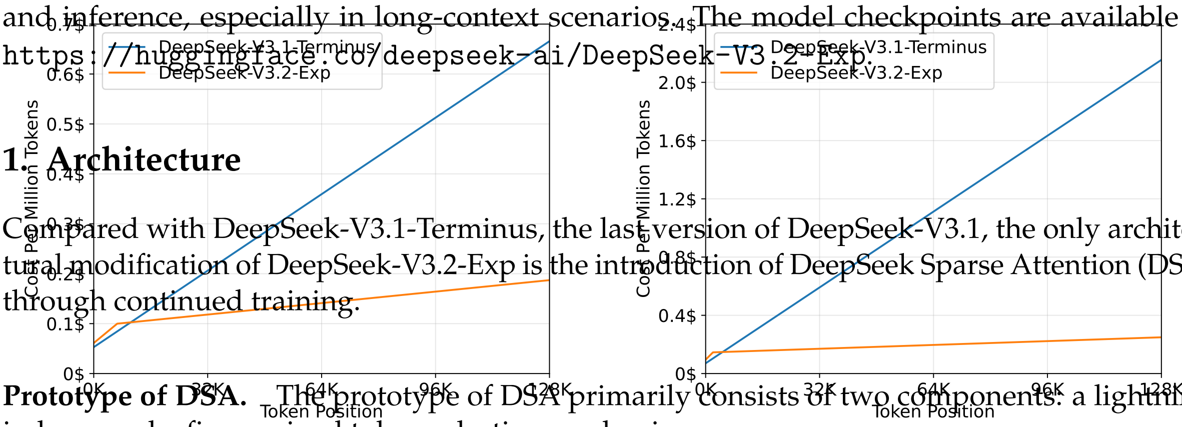


(a) BrowseComp Training Curve (b) SWE Training Curve

Figure 2 | RL training curves of DeepSeek-V3.1-Terminus and DeepSeek-V3.2-Exp on BrowseComp and SWE Verified. The solid lines denote the accuracy and dashed lines denote the average output tokens, respectively.

Abstract

We introduce DeepSeek-V3.2-Exp, an experimental sparse attention model, which equips DeepSeek-V3.1-Terminus with DeepSeek Sparse Attention (DSA) through continued training. With DSA, a fine-grained sparse attention mechanism powered by a lightning indexer, DeepSeek-V3.2-Exp achieves significant efficiency improvements in both training and inference, especially in long-context scenarios. The model checkpoints are available at https://huggingface.co/deepseek-ai/DeepSeek-V3.2-Exp.



(a) Prefilling (b) Decoding

Figure 3 | Inference costs of DeepSeek-V3.1-Terminus and DeepSeek-V3.2-Exp on H800 clusters.

1. Architecture

Compared with DeepSeek-V3.1-Terminus, the last version of DeepSeek-V3.1, the only architectural modification of DeepSeek-V3.2-Exp is the introduction of DeepSeek Sparse Attention (DSA) through continued training.

Prototype of DSA. The prototype of DSA primarily consists of two components: a lightning indexer and a fine-grained token selection mechanism.

The **lightning indexer** computes the index score $I_{t,s}$ between the query token $\mathbf{h}_t \in \mathbb{R}^d$ and a preceding token $\mathbf{h}_s \in \mathbb{R}^d$, determining which tokens to be selected by the query token:

$$I_{t,s} = \sum_{j=1}^{H^I} w^I_{t,j} \cdot \mathrm{ReLU}\left(\mathbf{q}^I_{t,j} \cdot \mathbf{k}^I_s\right), \tag{1}$$

where H^I denotes the number of indexer heads; $\mathbf{q}^I_{t,j} \in \mathbb{R}^{d^I}$ and $w^I_{t,j} \in \mathbb{R}$ are derived from the query token $\mathbf{h}_t$; and $\mathbf{k}^I_s \in \mathbb{R}^{d^I}$ is derived from the preceding token $\mathbf{h}_s$. We choose ReLU as the activation function for throughput consideration. Given that the lightning indexer has a small number of heads and can be implemented in FP8, its computational efficiency is remarkable.

Given the index scores $\{I_{t,s}\}$ for each query token $\mathbf{h}_t$, our **fine-grained token selection mechanism** retrieves only the key-value entries $\{\mathbf{c}_s\}$ corresponding to the top-k index scores. Then, the attention output $\mathbf{u}_t$ is computed by applying the attention mechanism between the query token $\mathbf{h}_t$ and the sparsely selected key-value entries $\{\mathbf{c}_s\}$:

$$\mathbf{u}_t = \mathrm{Attn}\left(\mathbf{h}_t, \left\{\mathbf{c}_s \,\middle|\, I_{t,s} \in \mathrm{Top\text{-}k}(I_{t,:})\right\}\right). \tag{2}$$

a rental price of 2 USD per GPU hour. Note that for short-sequence prefilling, we specially implement a masked MHA mode to simulate DSA, which can achieve higher efficiency under short-context conditions.

Future Validation in Real World. Although our internal evaluations show promising results of DeepSeek-V3.2-Exp, we are actively pursuing further large-scale testing in real-world scenarios to uncover potential limitations of the sparse attention architecture.

References

DeepSeek-AI. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. CoRR, abs/2405.04434, 2024. doi: 10.48550/ARXIV.2405.04434. URL https://doi.org/10.48550/arXiv.2405.04434.

DeepSeek-V3.2-Exp: Boosting Long-Context Efficiency with DeepSeek Sparse Attention

DeepSeek-AI
research@deepseek.com

Abstract

We introduce DeepSeek-V3.2-Exp, an experimental sparse-attention model, which equips DeepSeek-V3.1-Terminus with DeepSeek Sparse Attention (DSA) through continued training. With DSA, a fine-grained sparse attention mechanism powered by a lightning indexer, DeepSeek-V3.2-Exp achieves significant efficiency improvements in both training and inference, especially in long-context scenarios. The model checkpoints are available at https://huggingface.co/deepseek-ai/DeepSeek-V3.2-Exp.

1. Architecture

Compared with DeepSeek-V3.1-Terminus, the last version of DeepSeek-V3.1, the only architectural modification of DeepSeek-V3.2-Exp is the introduction of DeepSeek Sparse Attention (DSA) through continued training.

Prototype of DSA. The prototype of DSA primarily consists of two components: a lightning indexer and a fine-grained token selection mechanism.

The **lightning indexer** computes the index score $I_{t,s}$ between the query token $\mathbf{h}_t \in \mathbb{R}^d$ and a preceding token $\mathbf{h}_s \in \mathbb{R}^d$, determining which tokens to be selected by the query token:

$$I_{t,s} = \sum_{j=1}^{H^I} w_{t,j}^I \cdot \mathrm{ReLU}\left(\mathbf{q}_{t,j}^I \cdot \mathbf{k}_s^I\right), \tag{1}$$

where H^I denotes the number of indexer heads; $\mathbf{q}_{t,j}^I \in \mathbb{R}^{d^I}$ and $w_{t,j}^I \in \mathbb{R}$ are derived from the query token $\mathbf{h}_t$; and $\mathbf{k}_s^I \in \mathbb{R}^{d^I}$ is derived from the preceding token $\mathbf{h}_s$. We choose ReLU as the activation function for throughput consideration. Given that the lightning indexer has a small number of heads and can be implemented in FP8, its computational efficiency is remarkable.

Given the index scores $\{I_{t,s}\}$ for each query token $\mathbf{h}_t$, our **fine-grained token selection mechanism** retrieves only the key-value entries $\{\mathbf{c}_s\}$ corresponding to the top-k index scores. Then, the attention output $\mathbf{u}_t$ is computed by applying the attention mechanism between the query token $\mathbf{h}_t$ and the sparsely selected key-value entries $\{\mathbf{c}_s\}$:

$$\mathbf{u}_t = \mathrm{Attn}\big(\mathbf{h}_t, \{\mathbf{c}_s \,|\, I_{t,s} \in \mathrm{Top\text{-}k}(I_{t,:})\}\big). \tag{2}$$

DeepSeek-AI. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. Nature, 645(8081):633–638, 2025.

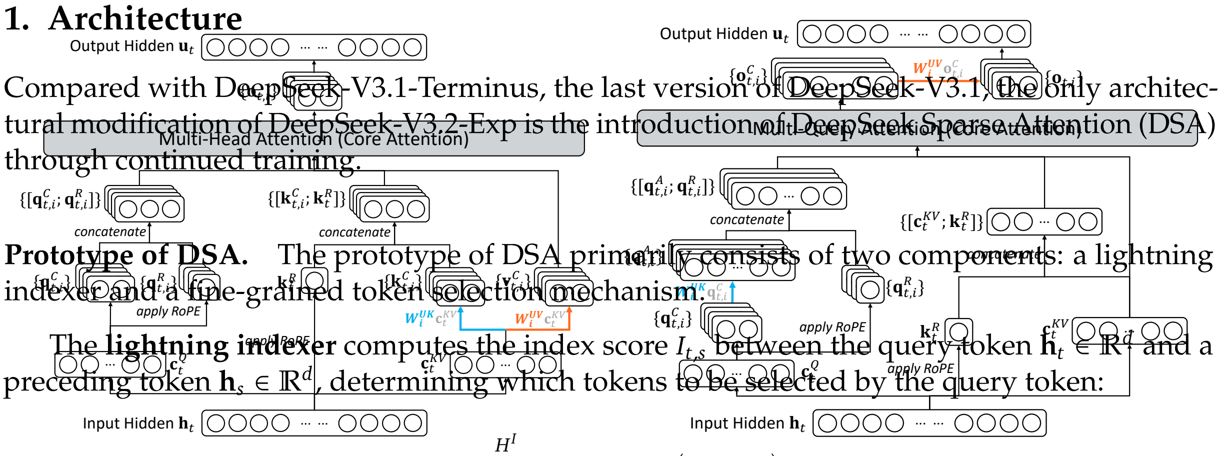
Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, M. Zhang, Y. K. Li, Y. Wu, and D. Guo. Deepseek-math: Pushing the limits of mathematical reasoning in open language models. CoRR, abs/2402.03300, 2024. doi: 10.48550/ARXIV.2402.03300. URL https://doi.org/10.48550/arXiv.2402.03300.

N. Shazeer. Fast transformer decoding: One write-head is all you need. CoRR, abs/1911.02150, 2019. URL http://arxiv.org/abs/1911.02150.

J. Yuan, H. Gao, D. Dai, J. Luo, L. Zhao, Z. Zhang, Z. Xie, Y. Wei, L. Wang, Z. Xiao, Y. Wang, C. Ruan, M. Zhang, W. Liang, and W. Zeng. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In W. Che, J. Nabende, E. Shutova, and M. T. Pilehvar, editors, Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, pages 23078–23097. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.acl-long.1126/.

Appendices

A. MHA and MQA Modes of MLA



(a) MHA mode of MLA. (b) MQA mode of MLA.

Figure 4 | Illustration of the MHA and MQA modes of MLA. For DeepSeek-V3.1-Terminus, the MHA mode is used for training and prefilling, while the MQA mode is used for decoding.

Figure 4 illustrates two aspects of MLA — the MHA and MQA modes — as well as the transformation between them.