DeepSeek-V3

Architecture, Design and Training

Houston Machine Learning Meetup

DeepSeek-V3: Highlights

- DeepSeek V3 has **671 billion parameters**, stands as one of the largest models in its class.
- During inference, only **37 billion parameters** are actively engaged, ensuring optimal resource utilization and energy efficiency.
- The model was trained on an extensive dataset comprising **14.8 trillion tokens**, equivalent to approximately **11.1 trillion words**.
- DeepSeek V3 achieved comparable or better capabilities with only **2.8 million GPU hours** comparing to the LLaMA 3 model consuming **30 million GPU hours**.
- Comparable or better performance comparing to Claude 3.5 and GPT-4o
- Open source!

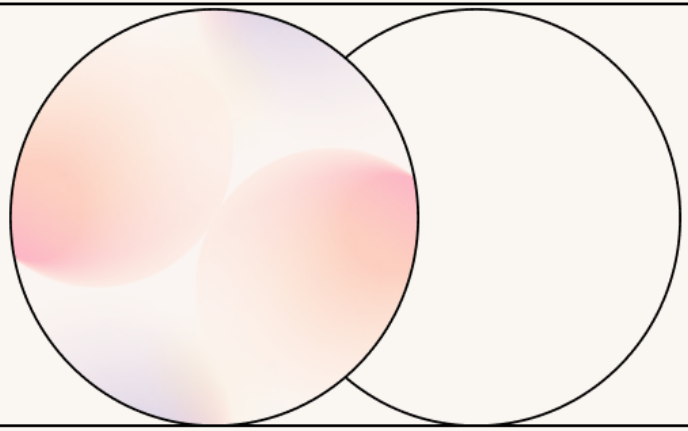
The Architecture and Design of DeepSeek-V3

Architecture

1. Multi-head latent attention (MLA)
2. DeepSeekMoE: Mixture of Experts

Training objective

3. Auxiliary-Loss-Free Load Balancing
4. Complementary Sequence-Wise Auxiliary Loss
5. Multi-token prediction training



DeepSeek-V3: Architecture Overview

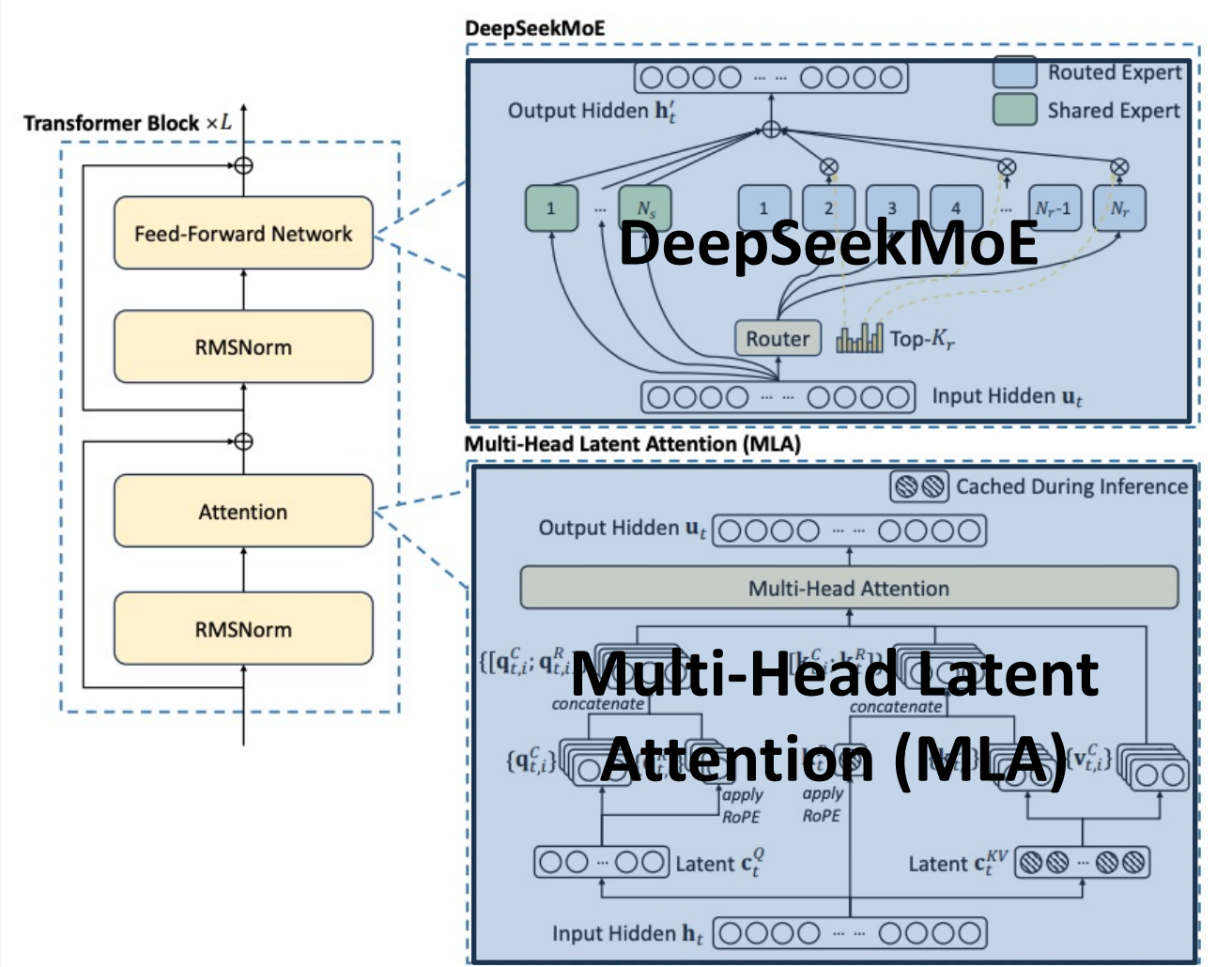
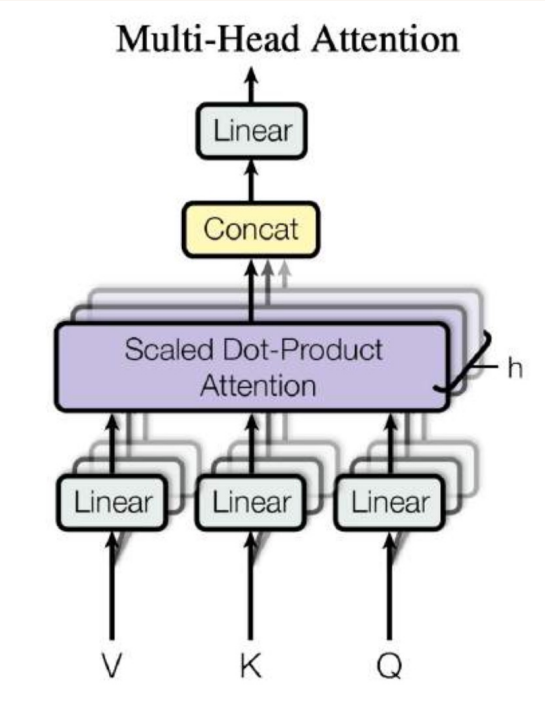
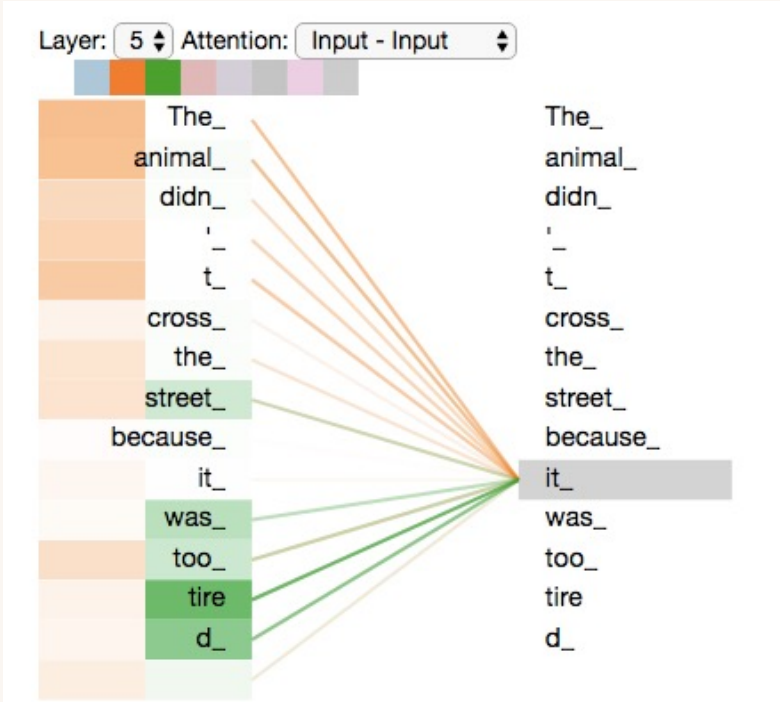


Figure 2 | Illustration of the basic architecture of DeepSeek-V3. Following DeepSeek-V2, we adopt MLA and DeepSeekMoE for efficient inference and economical training.

1. Multi-head latent attention (MLA)

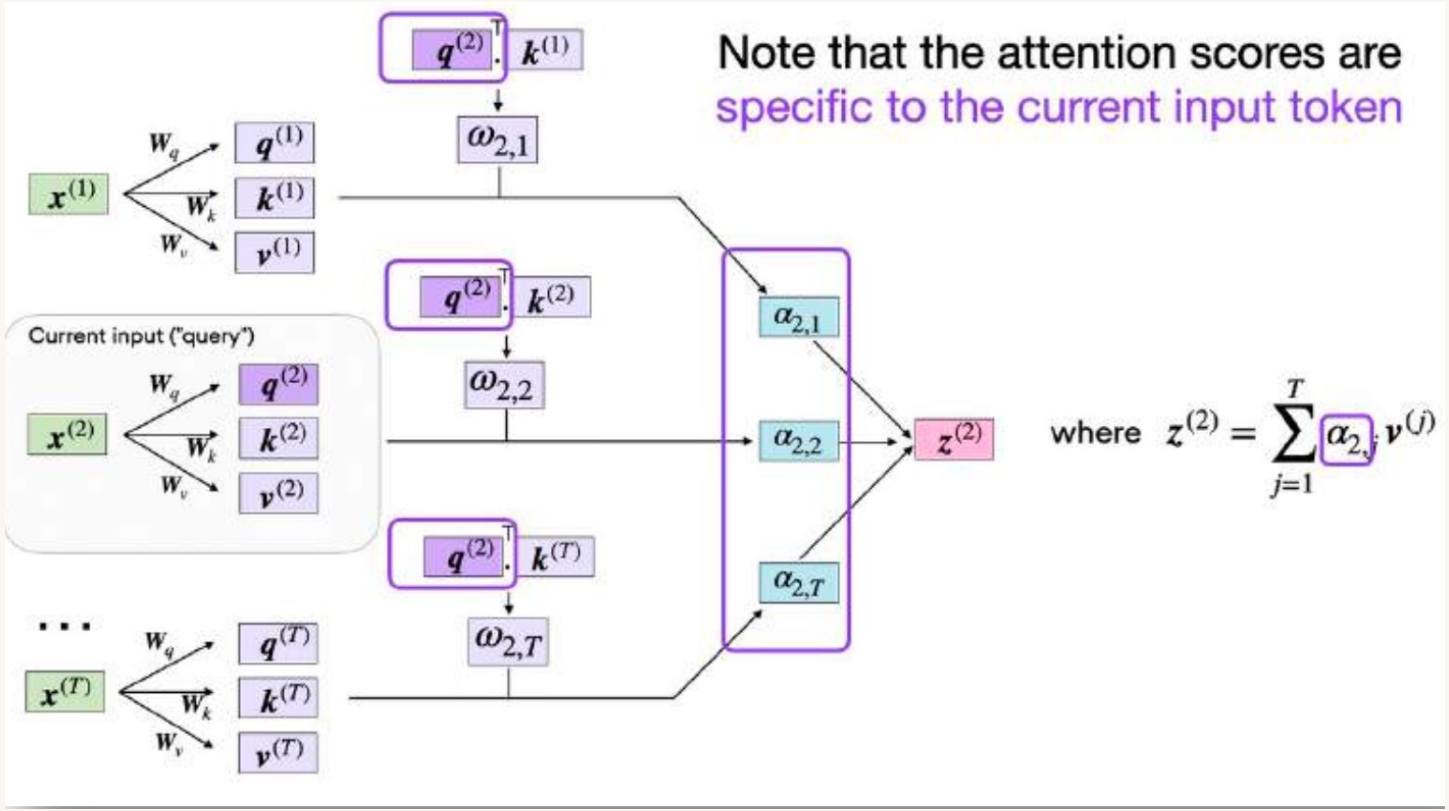
Multi-head attention



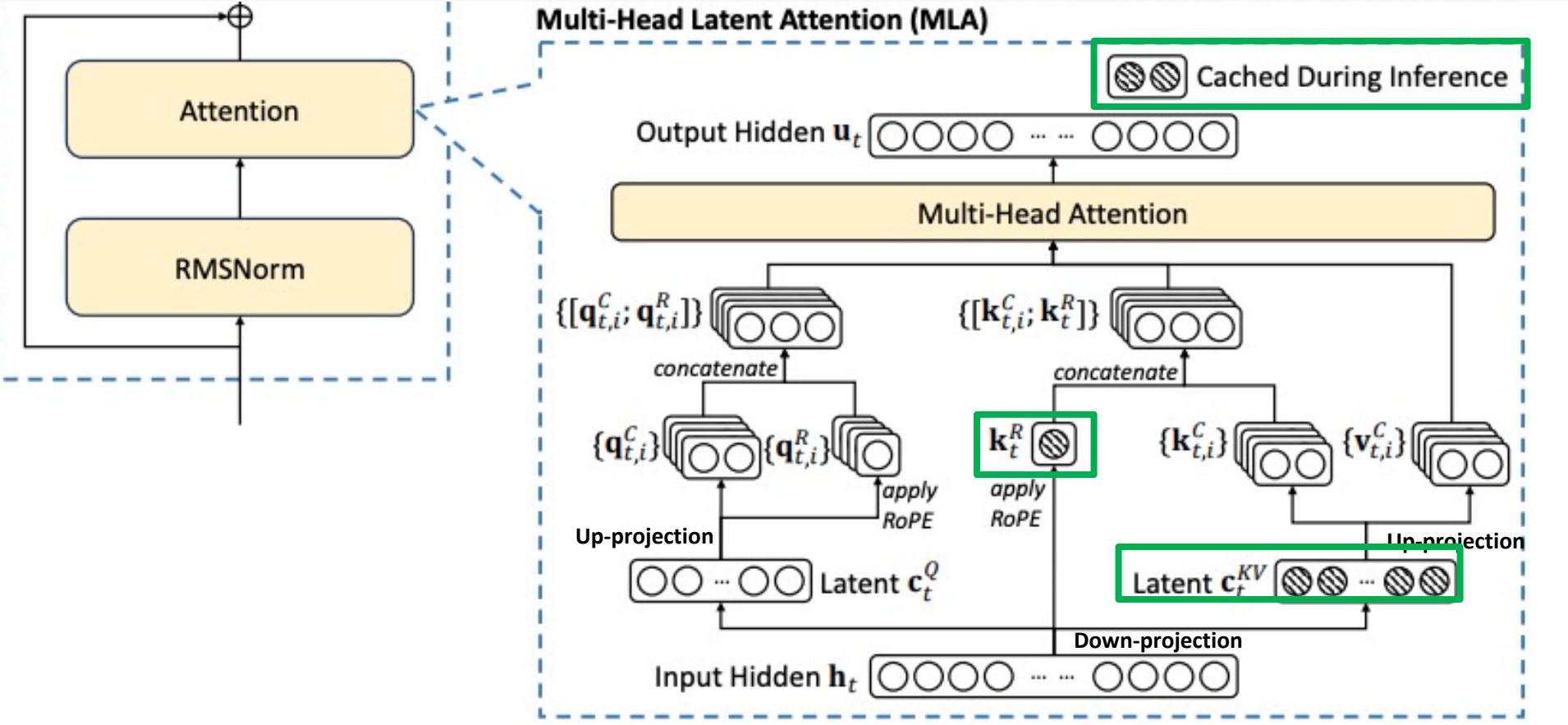


1. Multi-head latent attention (MLA)

Key-Value (KV) cache



1. Multi-head latent attention (MLA)



1. Multi-head latent attention (MLA)

Keys and Values:

$$\boxed{\mathbf{c}_t^{KV}} = W^{DKV}\mathbf{h}_t, \quad \text{Down-projection}$$

$$[\mathbf{k}_{t,1}^C; \mathbf{k}_{t,2}^C; \ldots; \mathbf{k}_{t,n_h}^C] = \mathbf{k}_t^C = W^{UK}\mathbf{c}_t^{KV}, \quad \text{Up-projection}$$

$$\boxed{\mathbf{k}_t^R} = \mathrm{RoPE}(W^{KR}\mathbf{h}_t), \quad \text{Rotary Positional Embeddings}$$

$$\mathbf{k}_{t,i} = [\mathbf{k}_{t,i}^C; \mathbf{k}_t^R],$$

$$[\mathbf{v}_{t,1}^C; \mathbf{v}_{t,2}^C; \ldots; \mathbf{v}_{t,n_h}^C] = \mathbf{v}_t^C = W^{UV}\mathbf{c}_t^{KV}, \quad \text{Up-projection}$$

1. Multi-head latent attention (MLA)

Queries:

$$\mathbf{c}_t^Q = W^{DQ}\mathbf{h}_t, \quad \text{Down-projection}$$

$$[\mathbf{q}_{t,1}^C; \mathbf{q}_{t,2}^C; \ldots; \mathbf{q}_{t,n_h}^C] = \mathbf{q}_t^C = W^{UQ}\mathbf{c}_t^Q, \quad \text{Up-projection}$$

$$[\mathbf{q}_{t,1}^R; \mathbf{q}_{t,2}^R; \ldots; \mathbf{q}_{t,n_h}^R] = \mathbf{q}_t^R = \mathrm{RoPE}(W^{QR}\mathbf{c}_t^Q), \quad \text{Rotary Positional Embeddings}$$

$$\mathbf{q}_{t,i} = [\mathbf{q}_{t,i}^C; \mathbf{q}_{t,i}^R],$$

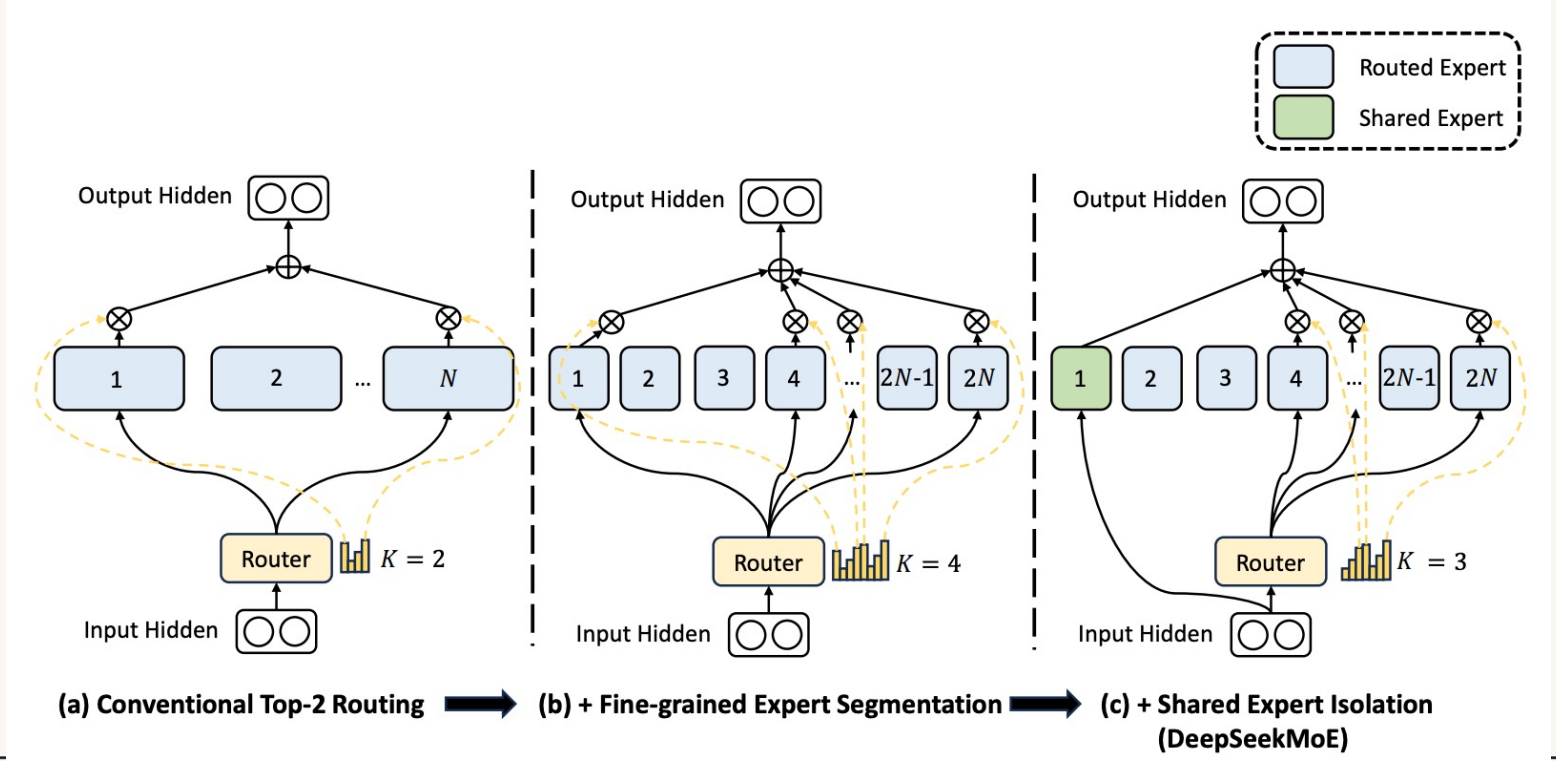
Outputs:

$$\mathbf{o}_{t,i} = \sum_{j=1}^{t} \mathrm{Softmax}_j\left(\frac{\mathbf{q}_{t,i}^T \mathbf{k}_{j,i}}{\sqrt{d_h + d_h^R}}\right)\mathbf{v}_{j,i}^C,$$

$$\mathbf{u}_t = W^O[\mathbf{o}_{t,1}; \mathbf{o}_{t,2}; \ldots; \mathbf{o}_{t,n_h}],$$

2. **DeepSeekMoE:** Mixture of Experts

MoE offers a compromise: it allows us to scale up model size to **increase model capacity** while **keeping training and inference cost manageable** by activating only a small fraction of the total parameters for each input token.



2. **DeepSeekMoE:** Mixture of Experts

Shared Routed

$$\mathbf{h}_t' = \mathbf{u}_t + \sum_{i=1}^{N_s} \mathrm{FFN}_i^{(s)}(\mathbf{u}_t) + \sum_{i=1}^{N_r} g_{i,t}\,\mathrm{FFN}_i^{(r)}(\mathbf{u}_t),$$

Output

$$g_{i,t} = \frac{g'_{i,t}}{\sum_{j=1}^{N_r} g'_{j,t}},$$

Normalization

$$g'_{i,t} = \begin{cases} s_{i,t}, & s_{i,t} \in \mathrm{Topk}(\{s_{j,t} | 1 \leqslant j \leqslant N_r\}, K_r), \\ 0, & \text{otherwise}, \end{cases}$$

Select the TopK

$$s_{i,t} = \mathrm{Sigmoid}\left(\mathbf{u}_t{}^T \mathbf{e}_i\right),$$

token-to-expert affinity

DeepSeek V3: The overall training loss

$$L_{\text{Total}} = L_{\text{MTP}} + L_{\text{Bal}}$$

Multi-Token Prediction **Balance loading**

Across Batch: Auxiliary-Loss-Free Load Balancing by dynamically adjusting bias for the expert routing

Across sequence:
Sequence-Wise Auxiliary Loss

$$\mathcal{L}_{\text{Bal}} = \alpha \sum_{i=1}^{N_r} f_i P_i,$$

Multi-Token Prediction

$$L_{\text{MTP}} = \frac{\lambda}{D} \sum_{k=1}^{D} L_{\text{MTP}}^{k}$$

3. DeepSeekMoE: Auxiliary-Loss-Free Load Balancing

Automatically learned routing strategies may encounter the issue of load imbalance, which manifests two notable defects:

- there is a risk of routing collapse (Shazeer et al., 2017), i.e., the model always selects only a few experts, preventing other experts from sufficient training.

- if experts are distributed across multiple devices, load imbalance can exacerbate computation bottlenecks.

3. **DeepSeekMoE:** Auxiliary-Loss-Free Load Balancing

- Instead of using an auxiliary loss, DeepSeek-V3 **dynamically adjusts routing** using bias terms.
- For each expert i, a **bias term** b_i is introduced and added to the **affinity score** $s_{i,t}$.
- The modified score determines which experts are selected:

$$g'_{i,t} = \begin{cases} s_{i,t}, & s_{i,t} + b_i \in \text{Topk}(\{s_{j,t} + b_j | 1 \leqslant j \leqslant N_r\}, K_r), \\ 0, & \text{otherwise.} \end{cases}$$

  - The bias term ensures that underutilized experts get a higher chance of selection.

- **At the end of each training step**, the load of each expert is monitored:
  - If an expert is **overloaded**, its bias term b_i is **decreased** by γ.
  - If an expert is **underloaded**, its bias term b_i is **increased** by γ.
  - γ is a **hyperparameter** controlling bias update speed.

4. **DeepSeekMoE:** Complementary Sequence-Wise Auxiliary Loss

To prevent **extreme imbalance within a single sequence**, it also applies a **complementary sequence-wise auxiliary loss**.

The **sequence-wise balance loss** is computed as:

$$L_{\text{Bal}} = \alpha \sum_{i=1}^{N_r} f_i P_i$$

where:

- α is a **small hyperparameter** that controls the impact of this loss.
- N_r is the number of **routed experts**.
- f_i represents the **load distribution of expert** i in a sequence.
- P_i denotes the **probability of expert** i **being selected**.

If the expert load is high, We want to decrease the probability of being selected.

4. **DeepSeekMoE:** Complementary Sequence-Wise Auxiliary Loss

3. Measuring Expert Load in a Sequence

The expert load f_i for expert i is given by:

$$f_i = \frac{N_r}{K_r T} \sum_{t=1}^{T} 1(s_{i,t} \in \text{Top-K}(\{s_{j,t} | 1 \leq j \leq N_r\}, K_r))$$

where:

- K_r is the **number of experts selected** per token.
- T is the **number of tokens in the sequence**.
- $s_{i,t}$ is the **affinity score** for expert i at token t.
- The indicator function $1(\cdot)$ counts when an expert is in the **Top-K selection**.

4. **DeepSeekMoE:** Complementary Sequence-Wise Auxiliary Loss

4. Normalizing Expert Probabilities

To ensure fair probability distribution, the expert selection probabilities are normalized:

$$s'_{i,t} = \frac{s_{i,t}}{\sum_{j=1}^{N_r} s_{j,t}}$$

Then, the average probability P_i for expert i across a sequence is:

$$P_i = \frac{1}{T} \sum_{t=1}^{T} s'_{i,t}$$

$$s_{i,t} = \text{Sigmoid}\left(\mathbf{u}_t{}^T \mathbf{e}_i\right),$$

4. **DeepSeekMoE:** Complementary Sequence-Wise Auxiliary Loss

Putting together

$$\mathcal{L}_{\text{Bal}} = \alpha \sum_{i=1}^{N_r} f_i P_i,$$

$$f_i = \frac{N_r}{K_r T} \sum_{t=1}^{T} \mathbb{1}\left(s_{i,t} \in \text{Topk}(\{s_{j,t} | 1 \leqslant j \leqslant N_r\}, K_r)\right),$$

$$s'_{i,t} = \frac{s_{i,t}}{\sum_{j=1}^{N_r} s_{j,t}},$$

$$P_i = \frac{1}{T} \sum_{t=1}^{T} s'_{i,t},$$

$$s_{i,t} = \text{Sigmoid}\left(\mathbf{u}_t{}^T \mathbf{e}_i\right),$$ token-to-expert affinity

5. Multi-Token Prediction Training Objective

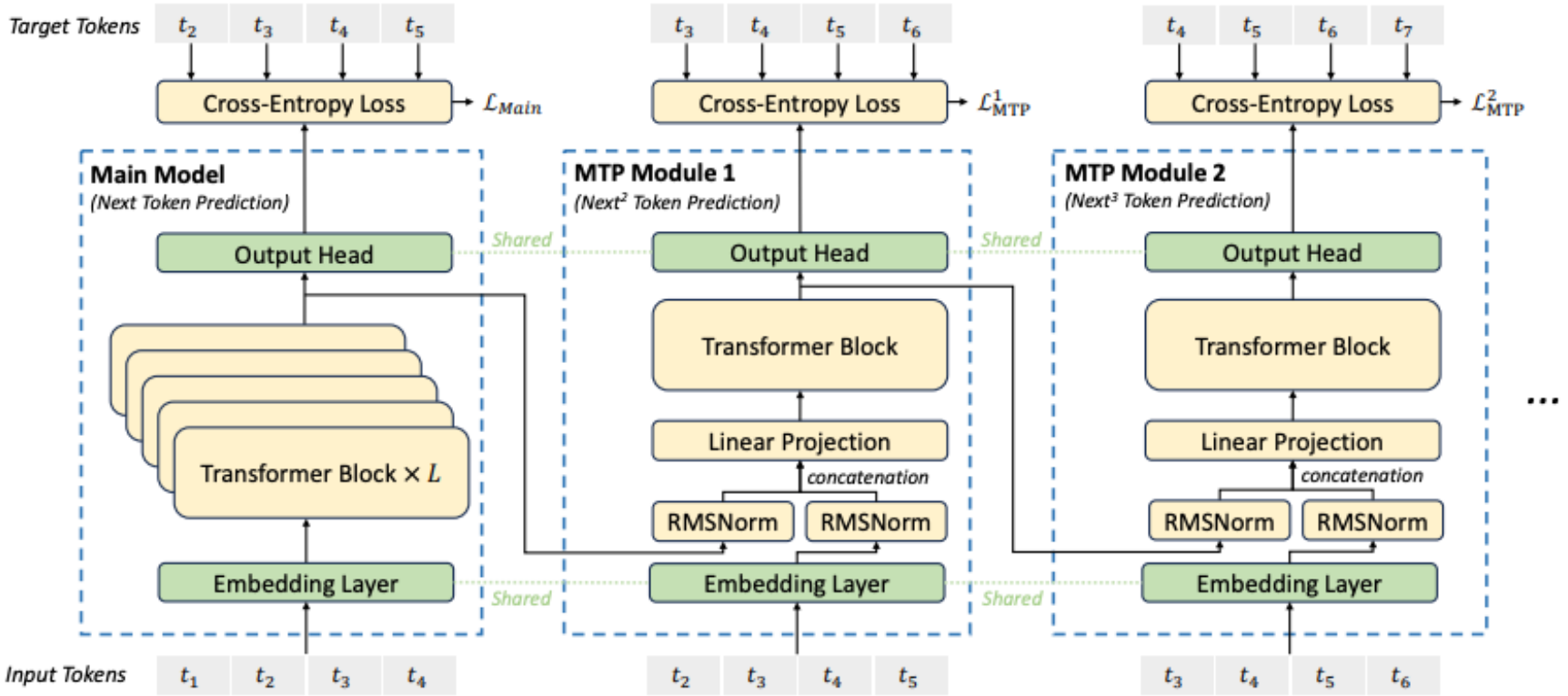


Figure 3 | Illustration of our Multi-Token Prediction (MTP) implementation. We keep the complete causal chain for the prediction of each token at each depth.

DeepSeek V3: The overall training loss

$$L_{\text{Total}} = L_{\text{MTP}} + L_{\text{Bal}}$$

Across Batch: Auxiliary-Loss-Free Load Balancing by dynamically adjusting bias for the expert routing

Across sequence:
Sequence-Wise Auxiliary Loss

$$\mathcal{L}_{\text{Bal}} = \alpha \sum_{i=1}^{N_r} f_i P_i,$$

Multi-Token Prediction

$$L_{\text{MTP}} = \frac{\lambda}{D} \sum_{k=1}^{D} L_{\text{MTP}}^{k}$$

DeepSeekMoE evaluation

Metric	# Shot	Dense	Hash Layer	Switch	GShard	DeepSeekMoE
# Total Params	N/A	0.2B	2.0B	2.0B	2.0B	2.0B
# Activated Params	N/A	0.2B	0.2B	0.2B	0.3B	0.3B
FLOPs per 2K Tokens	N/A	2.9T	2.9T	2.9T	4.3T	4.3T
# Training Tokens	N/A	100B	100B	100B	100B	100B
Pile (Loss)	N/A	2.060	1.932	1.881	1.867	**1.808**
HellaSwag (Acc.)	0-shot	38.8	46.2	49.1	50.5	**54.8**
PIQA (Acc.)	0-shot	66.8	68.4	70.5	70.6	**72.3**
ARC-easy (Acc.)	0-shot	41.0	45.3	45.9	43.9	**49.4**
ARC-challenge (Acc.)	0-shot	26.0	28.2	30.2	31.6	**34.3**
RACE-middle (Acc.)	5-shot	38.8	38.8	43.6	42.1	**44.0**
RACE-high (Acc.)	5-shot	29.0	30.0	30.9	30.4	**31.7**
HumanEval (Pass@1)	0-shot	0.0	1.2	2.4	3.7	**4.9**
MBPP (Pass@1)	3-shot	0.2	0.6	0.4	0.2	**2.2**
TriviaQA (EM)	5-shot	4.9	6.5	8.9	10.2	**16.6**
NaturalQuestions (EM)	5-shot	1.4	1.4	2.5	3.2	**5.7**

Table 1 | Evaluation results for validation experiments. **Bold** font indicates the best. Compared with other MoE architectures, DeepSeekMoE exhibits a substantial performance advantage.

DeepSeekMoE evaluation

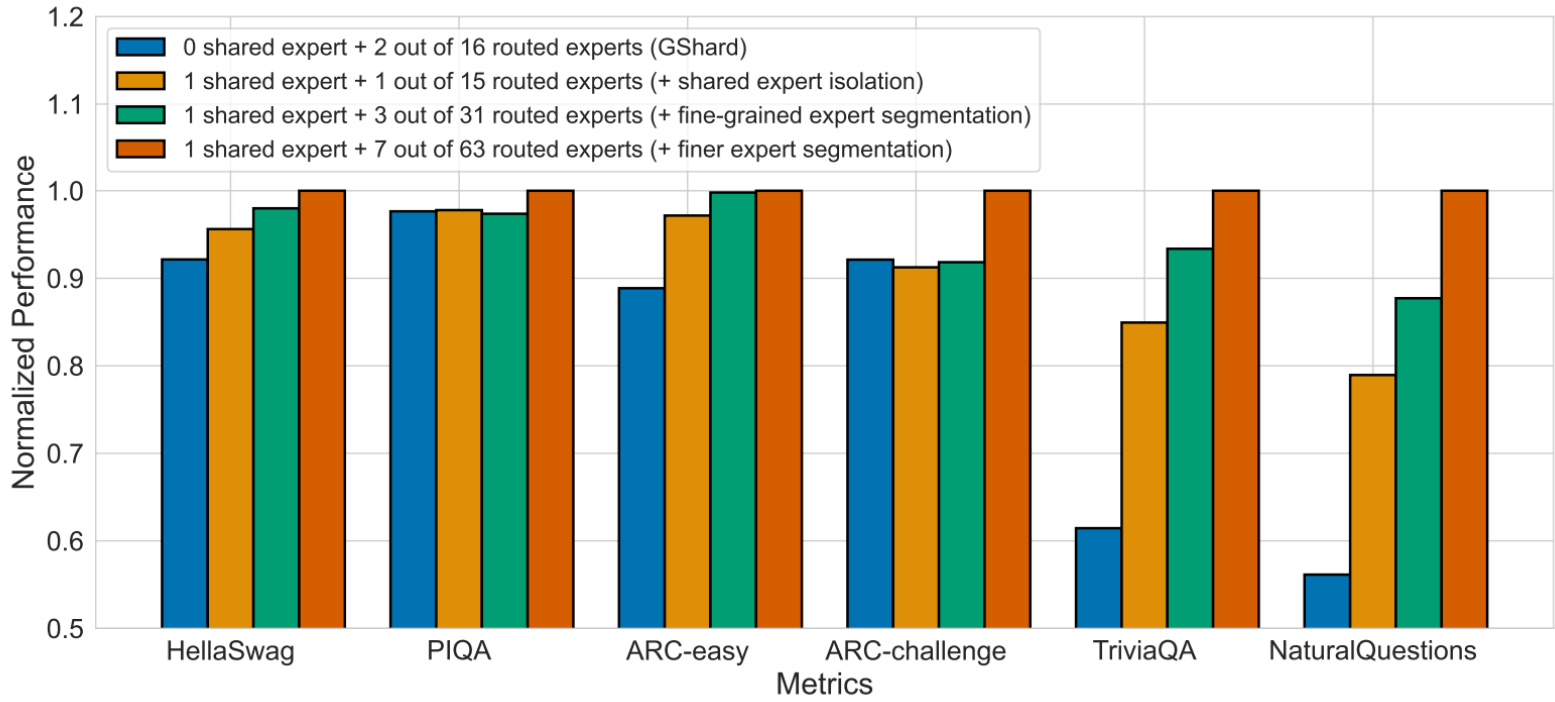


Figure 3 | Ablation studies for DeepSeekMoE. The performance is normalized by the best performance for clarity in presentation. All compared models have the same number of parameters and activated parameters. We can find that fine-grained expert segmentation and shared expert isolation both contribute to stronger overall performance.

Effectiveness of the auxiliary-loss-free balancing strategy

Benchmark (Metric)	# Shots	Small MoE Aux-Loss-Based	Small MoE Aux-Loss-Free	Large MoE Aux-Loss-Based	Large MoE Aux-Loss-Free
# Activated Params	-	2.4B	2.4B	20.9B	20.9B
# Total Params	-	15.7B	15.7B	228.7B	228.7B
# Training Tokens	-	1.33T	1.33T	578B	578B
Pile-test (BPB)	-	0.727	**0.724**	0.656	**0.652**
BBH (EM)	3-shot	37.3	**39.3**	66.7	**67.9**
MMLU (EM)	5-shot	51.0	**51.8**	**68.3**	67.2
DROP (F1)	1-shot	38.1	**39.0**	**67.1**	**67.1**
TriviaQA (EM)	5-shot	**58.3**	**58.5**	66.7	**67.7**
NaturalQuestions (EM)	5-shot	**23.2**	**23.4**	27.1	**28.1**
HumanEval (Pass@1)	0-shot	22.0	**22.6**	40.2	**46.3**
MBPP (Pass@1)	3-shot	**36.6**	35.8	59.2	**61.2**
GSM8K (EM)	8-shot	27.1	**29.6**	70.7	**74.5**
MATH (EM)	4-shot	**10.9**	**11.1**	37.2	**39.6**

Table 5 | Ablation results for the auxiliary-loss-free balancing strategy. Compared with the purely auxiliary-loss-based method, the auxiliary-loss-free strategy consistently achieves better model performance on most of the evaluation benchmarks.

Effectiveness of the Multi-token Prediction

Benchmark (Metric)	# Shots	Small MoE Baseline	Small MoE w/ MTP	Large MoE Baseline	Large MoE w/ MTP
# Activated Params (Inference)	-	2.4B	2.4B	20.9B	20.9B
# Total Params (Inference)	-	15.7B	15.7B	228.7B	228.7B
# Training Tokens	-	1.33T	1.33T	540B	540B
Pile-test (BPB)	-	**0.729**	**0.729**	0.658	**0.657**
BBH (EM)	3-shot	39.0	**41.4**	70.0	**70.7**
MMLU (EM)	5-shot	50.0	**53.3**	**67.5**	66.6
DROP (F1)	1-shot	39.2	**41.3**	68.5	**70.6**
TriviaQA (EM)	5-shot	56.9	**57.7**	**67.0**	**67.3**
NaturalQuestions (EM)	5-shot	**22.7**	22.3	27.2	**28.5**
HumanEval (Pass@1)	0-shot	20.7	**26.8**	44.5	**53.7**
MBPP (Pass@1)	3-shot	35.8	**36.8**	61.6	**62.2**
GSM8K (EM)	8-shot	25.4	**31.4**	72.3	**74.0**
MATH (EM)	4-shot	10.7	**12.6**	38.6	**39.8**

Table 4 | Ablation results for the MTP strategy. The MTP strategy consistently enhances the model performance on most of the evaluation benchmarks.

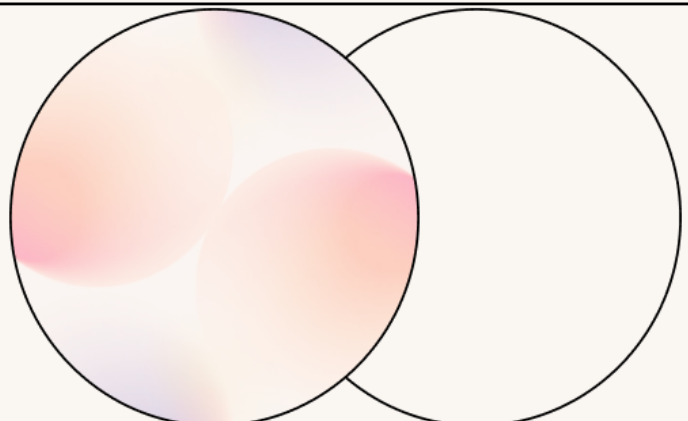
The Pre-training and Post-training of DeepSeek-V3

Pre-training:

1. Data curation
2. Model hyper-parameters
3. Long context extension

Post-training:

4. Supervised fine-tuning
5. Reinforcement learning



1. Pre-training: Data Construction **14.8T high-quality and diverse tokens**

Compared with DeepSeek-V2, we optimize the pre-training corpus by enhancing the ratio of mathematical and programming samples, while expanding multilingual coverage beyond English and Chinese.

The data processing pipeline is refined to minimize redundancy while maintaining corpus diversity

Fill-in-Middle (FIM) strategy does not compromise the next-token prediction capability while enabling the model to accurately predict middle text based on contextual cues.

be specific, we employ the Prefix-Suffix-Middle (PSM) framework to structure data as follows:

Middle session to predict

<|fim_begin|>f_{pre}<|fim_hole|>f_{suf}<|fim_end|>f_{middle}<|eos_token|>.

This structure is applied at the document level as a part of the pre-packing process. The FIM strategy is applied at a rate of 0.1, consistent with the PSM framework.

2. Pre-training: Model Hyper-Parameters

number of Transformer layers = 61
hidden dimension = 7168.
all learnable parameters are randomly initialized with a standard deviation of 0.006.

In MLA:
the number of attention heads = 128
per-head dimension = 128
KV compression dimension = 512
query compression dimension = 1536.

In MoE:
Substitute all Fully Forward Network except for the first three layers with MoE layers.
Each MoE layer consists of 1 shared expert and 256 routed experts
Among the routed experts, 8 experts will be activated for each token

The multi-token prediction depth is set to 1, i.e., besides the exact next token, each token will predict one additional token.

3. Pre-training: Long context extension

After the pre-training stage, we apply **YaRN** (Yet Another Rope extensioN method, Peng et al., 2023a) for context extension and **perform two additional training phases**, each comprising 1000 steps, to progressively expand the context window from 4K to 32K and then to 128K.

4. Post-training: Supervised Fine-Tuning

1.5M instruction instances

We curate our instruction-tuning datasets to include 1.5M instances spanning multiple domains.

Reasoning Data:
mathematics,
code competition problems,
and logic puzzles etc.

Generated by leveraging an internal DeepSeek-R1 model.

Non-Reasoning Data:
creative writing, role-play, and simple question answering etc.

Generated by DeepSeek-V2.5 to generate responses and use human annotators to verify the accuracy and correctness.

5. Post-training: Reinforcement Learning

Rule-based RM:
mathematics,
code competition problems,
and logic puzzles etc.

Reward: Hard-coded validation rules

Model-based RM:
creative writing, role-play, and simple question answering etc.

Reward: DeepSeek-V3 SFT model trained with human preference data that includes the final reward and chain-of-thoughts

Group Relative Policy Optimization (GRPO)

Evaluation

Pretrained model:
DeepSeek-V3 Base

	Benchmark (Metric)	# Shots	DeepSeek-V2 Base	Qwen2.5 72B Base	LLaMA-3.1 405B Base	DeepSeek-V3 Base
	Architecture	-	MoE	Dense	Dense	MoE
	# Activated Params	-	21B	72B	405B	37B
	# Total Params	-	236B	72B	405B	671B
English	Pile-test (BPB)	-	0.606	0.638	**0.542**	0.548
	BBH (EM)	3-shot	78.8	79.8	82.9	**87.5**
	MMLU (EM)	5-shot	78.4	85.0	84.4	**87.1**
	MMLU-Redux (EM)	5-shot	75.6	83.2	81.3	**86.2**
	MMLU-Pro (EM)	5-shot	51.4	58.3	52.8	**64.4**
	DROP (F1)	3-shot	80.4	80.6	86.0	**89.0**
	ARC-Easy (EM)	25-shot	97.6	98.4	98.4	**98.9**
	ARC-Challenge (EM)	25-shot	92.2	94.5	**95.3**	**95.3**
	HellaSwag (EM)	10-shot	87.1	84.8	**89.2**	**88.9**
	PIQA (EM)	0-shot	83.9	82.6	**85.9**	84.7
	WinoGrande (EM)	5-shot	**86.3**	82.3	85.2	84.9
	RACE-Middle (EM)	5-shot	73.1	68.1	**74.2**	67.1
	RACE-High (EM)	5-shot	52.6	50.3	**56.8**	51.3
	TriviaQA (EM)	5-shot	80.0	71.9	**82.7**	**82.9**
	NaturalQuestions (EM)	5-shot	38.6	33.2	**41.5**	40.0
	AGIEval (EM)	0-shot	57.5	75.8	60.6	**79.6**
Code	HumanEval (Pass@1)	0-shot	43.3	53.0	54.9	**65.2**
	MBPP (Pass@1)	3-shot	65.0	72.6	68.4	**75.4**
	LiveCodeBench-Base (Pass@1)	3-shot	11.6	12.9	15.5	**19.4**
	CRUXEval-I (EM)	2-shot	52.5	59.1	58.5	**67.3**
	CRUXEval-O (EM)	2-shot	49.8	59.9	59.9	**69.8**
Math	GSM8K (EM)	8-shot	81.6	88.3	83.5	**89.3**
	MATH (EM)	4-shot	43.4	54.4	49.0	**61.6**
	MGSM (EM)	8-shot	63.6	76.2	69.9	**79.8**
	CMath (EM)	3-shot	78.7	84.5	77.3	**90.7**
Chinese	CLUEWSC (EM)	5-shot	82.0	82.5	**83.0**	**82.7**
	C-Eval (EM)	5-shot	81.4	89.2	72.5	**90.1**
	CMMLU (EM)	5-shot	84.0	**89.5**	73.7	88.8
	CMRC (EM)	1-shot	**77.4**	75.8	76.0	76.3
	C3 (EM)	0-shot	77.4	76.7	**79.7**	78.6
	CCPM (EM)	0-shot	**93.0**	88.5	78.6	92.0
Multilingual	MMMLU-non-English (EM)	5-shot	64.0	74.8	73.8	**79.4**

Evaluation

Post-trained model:
DeepSeek-V3

	Benchmark (Metric)	DeepSeek V2-0506	DeepSeek V2.5-0905	Qwen2.5 72B-Inst.	LLaMA-3.1 405B-Inst.	Claude-3.5-Sonnet-1022	GPT-4o 0513	DeepSeek V3
	Architecture	MoE	MoE	Dense	Dense	-	-	MoE
	# Activated Params	21B	21B	72B	405B	-	-	37B
	# Total Params	236B	236B	72B	405B	-	-	671B
English	MMLU (EM)	78.2	80.6	85.3	**88.6**	**88.3**	87.2	**88.5**
	MMLU-Redux (EM)	77.9	80.3	85.6	86.2	**88.9**	88.0	**89.1**
	MMLU-Pro (EM)	58.5	66.2	71.6	73.3	**78.0**	72.6	75.9
	DROP (3-shot F1)	83.0	87.8	76.7	88.7	88.3	83.7	**91.**
	IF-Eval (Prompt Strict)	57.7	80.6	84.1	86.0	**86.5**	84.3	86.1
	GPQA-Diamond (Pass@1)	35.3	41.3	49.0	51.1	**65.0**	49.9	59.1
	SimpleQA (Correct)	9.0	10.2	9.1	17.1	28.4	**38.2**	24.9
	FRAMES (Acc.)	66.9	65.4	69.8	70.0	72.5	**80.5**	73.3
	LongBench v2 (Acc.)	31.6	35.4	39.4	36.1	41.0	48.1	**48.7**
Code	HumanEval-Mul (Pass@1)	69.3	77.4	77.3	77.2	81.7	80.5	**82.6**
	LiveCodeBench (Pass@1-COT)	18.8	29.2	31.1	28.4	36.3	33.4	**40.5**
	LiveCodeBench (Pass@1)	20.3	28.4	28.7	30.1	32.8	34.2	**37.6**
	Codeforces (Percentile)	17.5	35.6	24.8	25.3	20.3	23.6	**51.6**
	SWE Verified (Resolved)	-	22.6	23.8	24.5	**50.8**	38.8	42.0
	Aider-Edit (Acc.)	60.3	71.6	65.4	63.9	**84.2**	72.9	79.7
	Aider-Polyglot (Acc.)	-	18.2	7.6	5.8	45.3	16.0	**49.6**
Math	AIME 2024 (Pass@1)	4.6	16.7	23.3	23.3	16.0	9.3	**39.2**
	MATH-500 (EM)	56.3	74.7	80.0	73.8	78.3	74.6	**90.2**
	CNMO 2024 (Pass@1)	2.8	10.8	15.9	6.8	13.1	10.8	**43.2**
Chinese	CLUEWSC (EM)	89.9	90.4	**91.4**	84.7	85.4	87.9	90.9
	C-Eval (EM)	78.6	79.5	86.1	61.5	76.7	76.0	**86.5**
	C-SimpleQA (Correct)	48.5	54.1	48.4	50.4	51.3	59.3	**64.8**

Table 6 | Comparison between DeepSeek-V3 and other representative chat models. All models are evaluated in a configuration that limits the output length to 8K. Benchmarks containing fewer than 1000 samples are tested multiple times using varying temperature settings to derive robust final results. DeepSeek-V3 stands as the best-performing open-source model, and also exhibits competitive performance against frontier closed-source models.

Model	Arena-Hard	AlpacaEval 2.0
DeepSeek-V2.5-0905	76.2	50.5
Qwen2.5-72B-Instruct	81.2	49.1
LLaMA-3.1 405B	69.3	40.5
GPT-4o-0513	80.4	51.1
Claude-Sonnet-3.5-1022	85.2	52.0
DeepSeek-V3	**85.5**	**70.0**

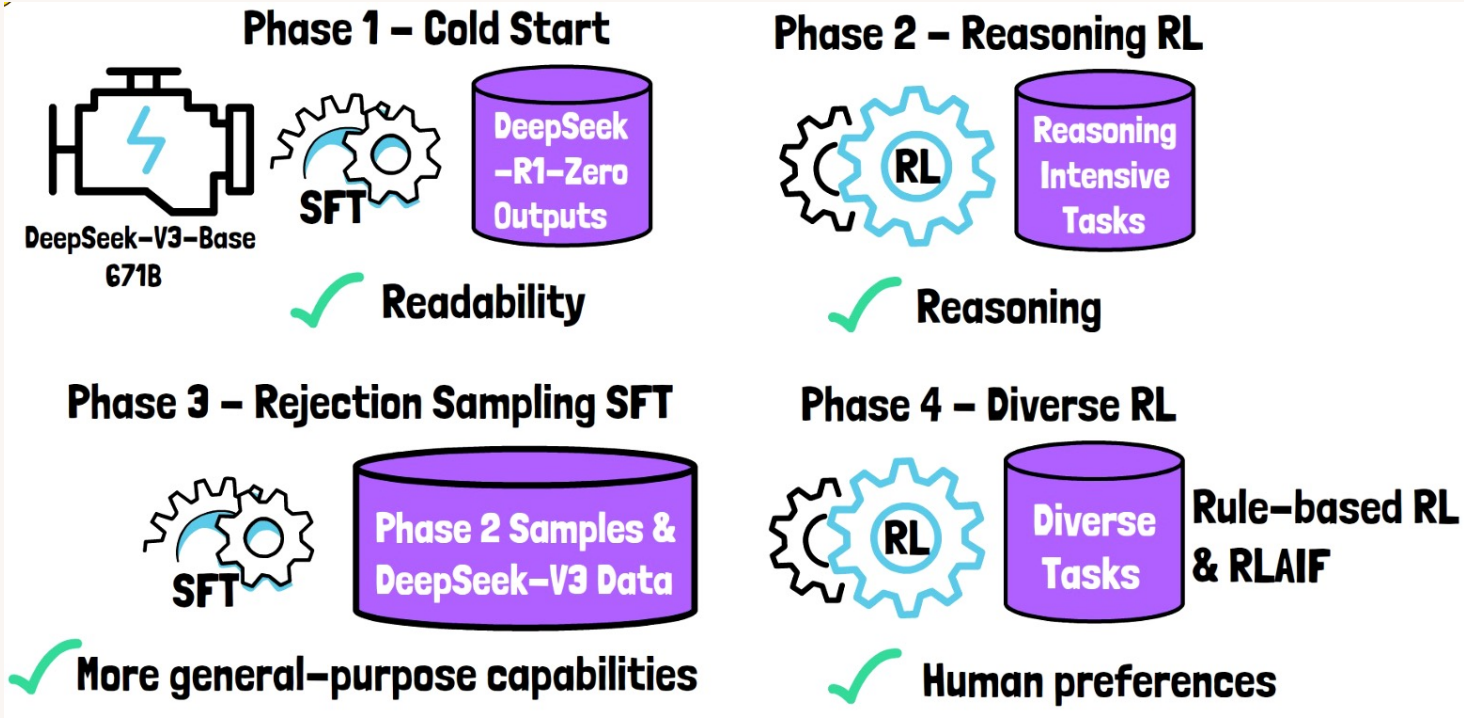
GPT-4-Turbo is used to **judge the quality of responses**.

Table 7 | English open-ended conversation evaluations. For AlpacaEval 2.0, we use the length-controlled win rate as the metric.

Model	Chat	Chat-Hard	Safety	Reasoning	Average
GPT-4o-0513	96.6	70.4	86.7	84.9	84.7
GPT-4o-0806	96.1	76.1	88.1	86.6	86.7
GPT-4o-1120	95.8	71.3	86.2	85.2	84.6
Claude-3.5-sonnet-0620	96.4	74.0	81.6	84.7	84.2
Claude-3.5-sonnet-1022	96.4	79.7	91.1	87.6	88.7
DeepSeek-V3	96.9	79.8	87.0	84.3	87.0
DeepSeek-V3 (maj@6)	96.9	82.6	89.5	89.2	89.6

Table 8 | Performances of GPT-4o, Claude-3.5-sonnet and DeepSeek-V3 on RewardBench.

DeepSeek R1 starts with V3-Base (the pre-trained model)



Thank You

Find out more @ YanAITalk!