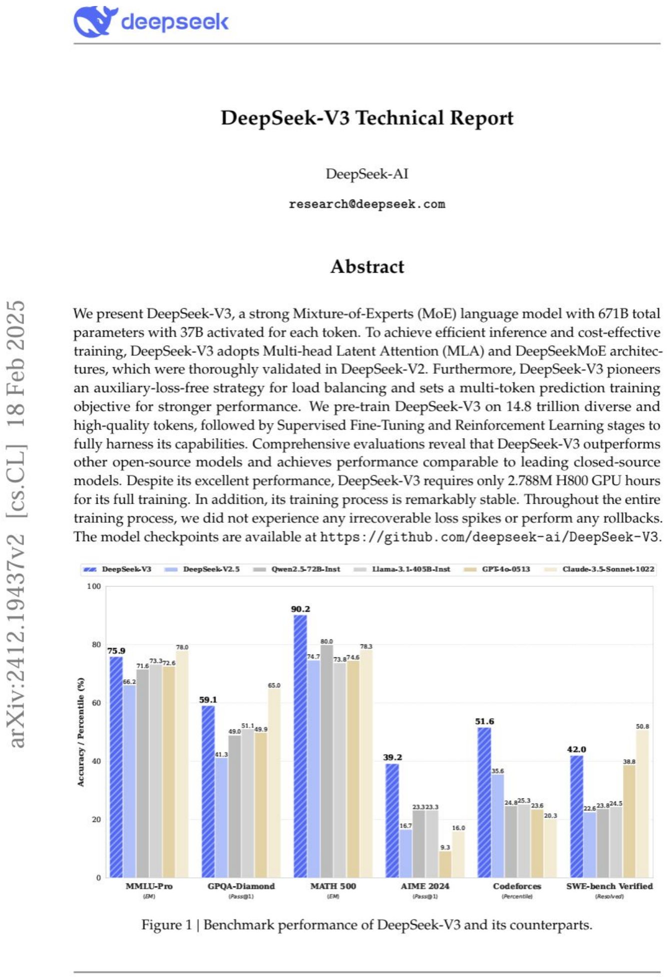
DeepSeek-V3:
Multihead Latent Attention (MLA) & DeepSeekMoE

Minwu Kim, Safal Shrestha

DeepSeek-V3

- 671B-parameter model with 37B active parameters during inference
- Performance on par with GPT-4o
- Backbone of DeepSeek-R1
- Optimized for efficient inference and cost-effective training



DeepSeek-V3 Technical Report

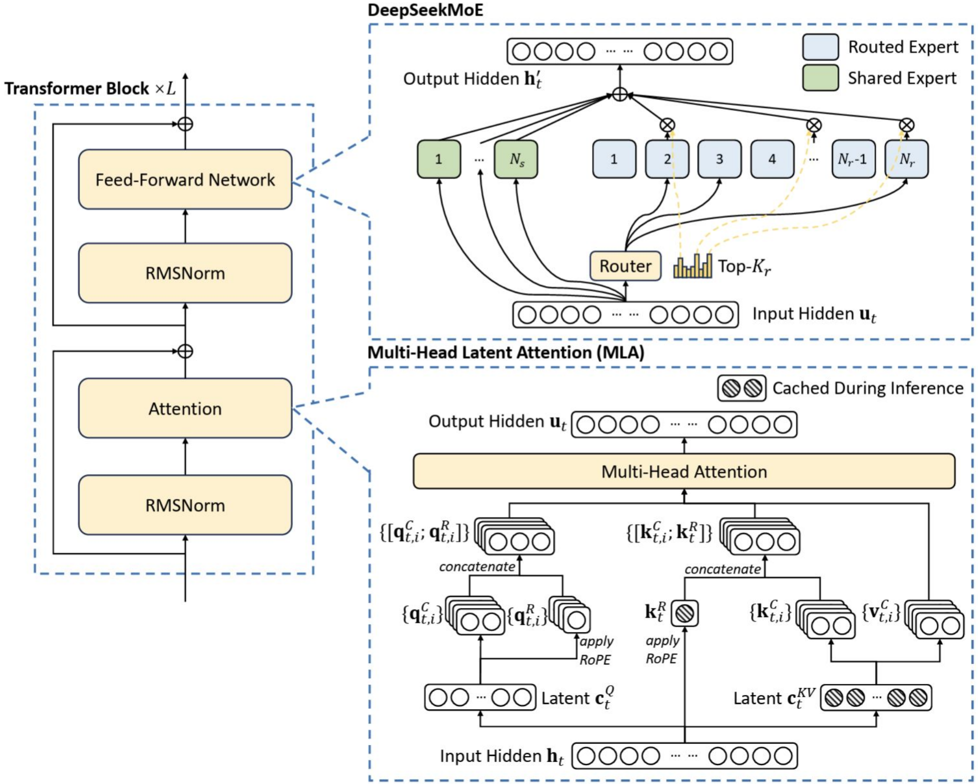
DeepSeek-AI

research@deepseek.com

Abstract

We present DeepSeek-V3, a strong Mixture-of-Experts (MoE) language model with 671B total parameters with 37B activated for each token. To achieve efficient inference and cost-effective training, DeepSeek-V3 adopts Multi-head Latent Attention (MLA) and DeepSeekMoE architectures, which were thoroughly validated in DeepSeek-V2. Furthermore, DeepSeek-V3 pioneers an auxiliary-loss-free strategy for load balancing and sets a multi-token prediction training objective for stronger performance. We pre-train DeepSeek-V3 on 14.8 trillion diverse and high-quality tokens, followed by Supervised Fine-Tuning and Reinforcement Learning stages to fully harness its capabilities. Comprehensive evaluations reveal that DeepSeek-V3 outperforms other open-source models and achieves performance comparable to leading closed-source models. Despite its excellent performance, DeepSeek-V3 requires only 2.788M H800 GPU hours for its full training. In addition, its training process is remarkably stable. Throughout the entire training process, we did not experience any irrecoverable loss spikes or perform any rollbacks. The model checkpoints are available at `https://github.com/deepseek-ai/DeepSeek-V3`.

Figure 1 | Benchmark performance of DeepSeek-V3 and its counterparts.



DeepSeek-V3

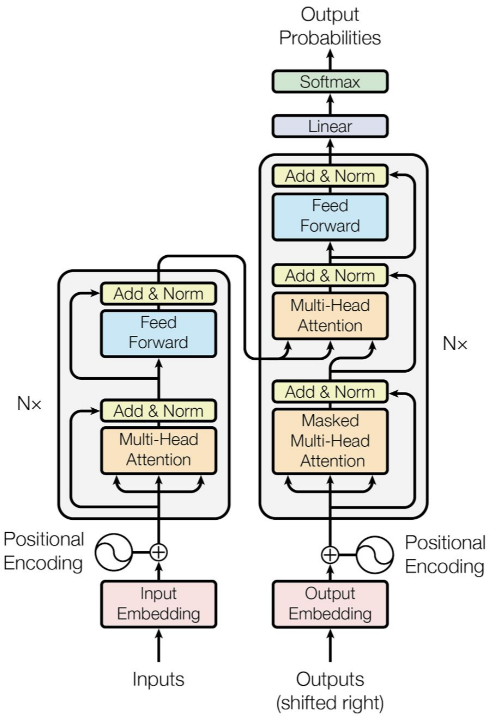
Efficiency achieved by:

- Multi-head Latent Attention
- DeepSeekMoE

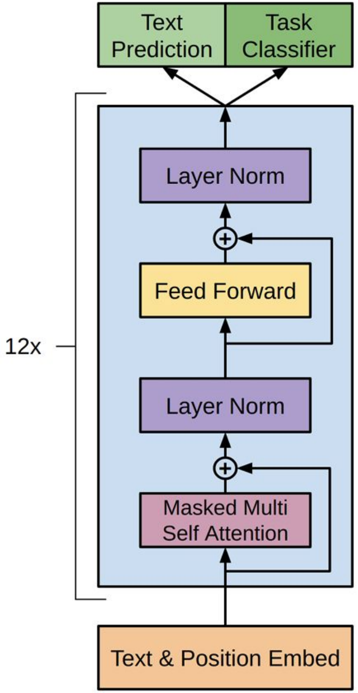
Overview

1. Recap for transformer
2. Multi-head Latent Attention (MLA)
   a. Self-attention
   b. KV cache
   c. MLA
3. DeepSeekMoE
   a. Feedforward component
   b. Mixture of Experts (MoE)
   c. DeepSeekMoE

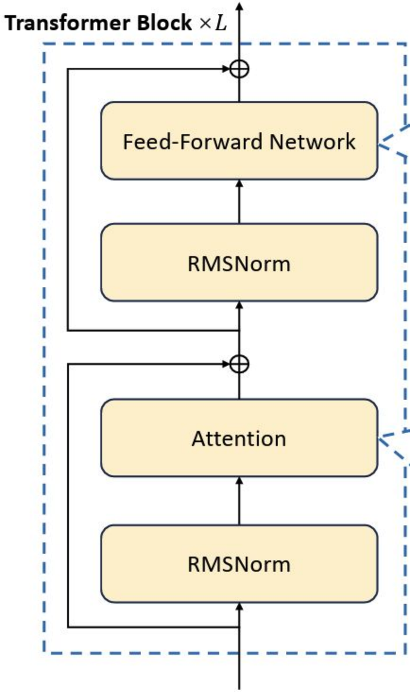
Preliminaries - Transformer Recap



Encoder-decoder transformer

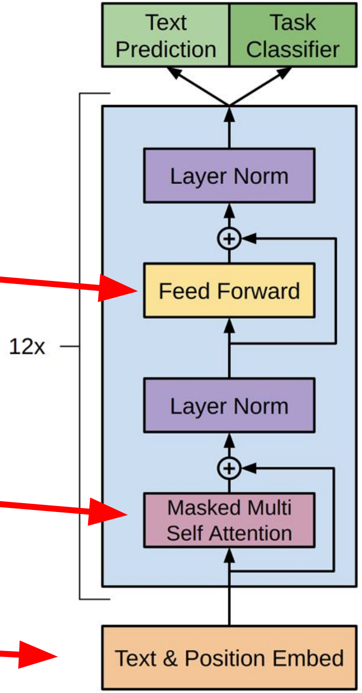


Decoder-only transformer (GPT)



Decoder-only transformer (V3)

Preliminaries - Transformer Recap



5. Output Representation

4. Repeat

3. MLP

2. Self-Attention

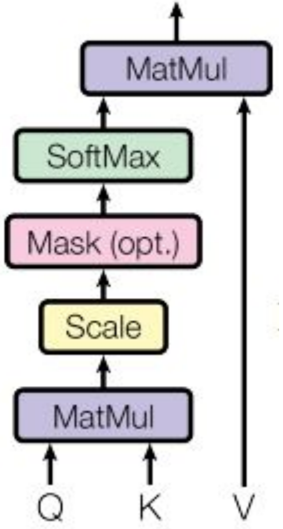
1. Input Embeddings

(each token = vector)

Decoder-only transformer (GPT)

Multi-head Latent Attention (MLA)

Preliminary 1: Self-attention

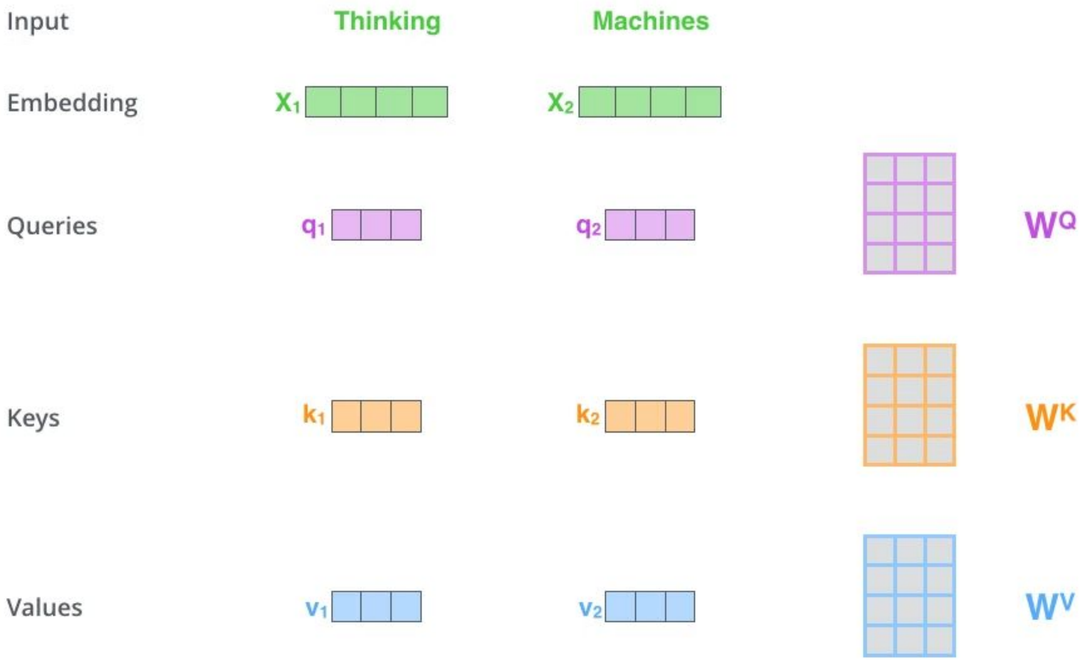


$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Multi-head Latent Attention (MLA)

Preliminary 1: Self-attention

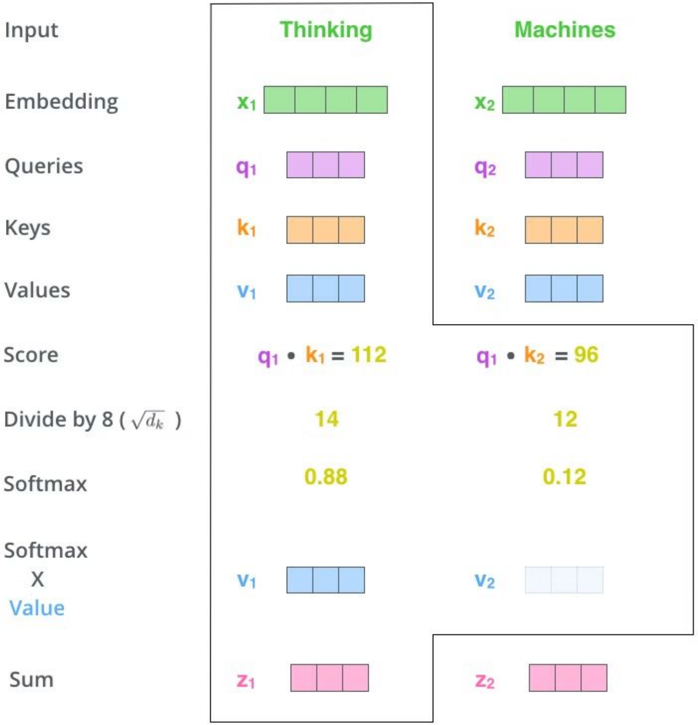
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



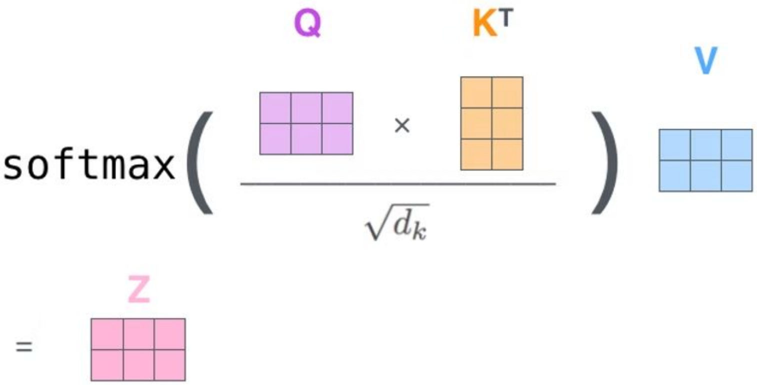
Multi-head Latent Attention (MLA)

Preliminary 1: Self-attention

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



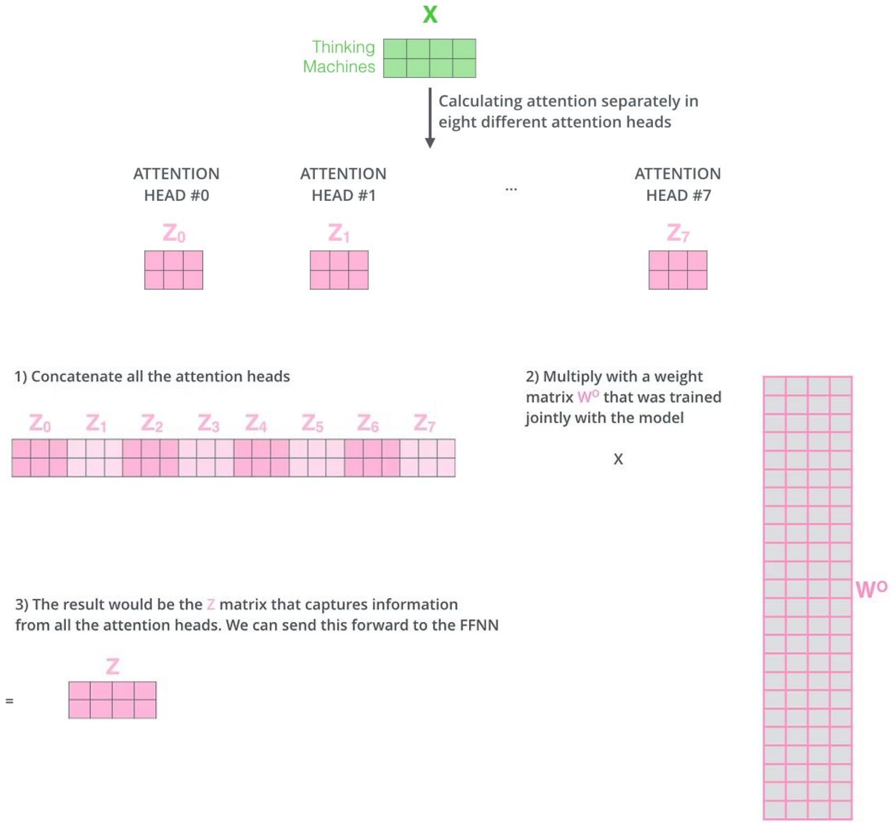
Input	Thinking	Machines
Embedding	x_1	x_2
Queries	q_1	q_2
Keys	k_1	k_2
Values	v_1	v_2
Score	$q_1 \bullet k_1 = 112$	$q_1 \bullet k_2 = 96$
Divide by 8 ($\sqrt{d_k}$)	14	12
Softmax	0.88	0.12
Softmax X Value	v_1	v_2
Sum	z_1	z_2



$$\text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right)V = Z$$

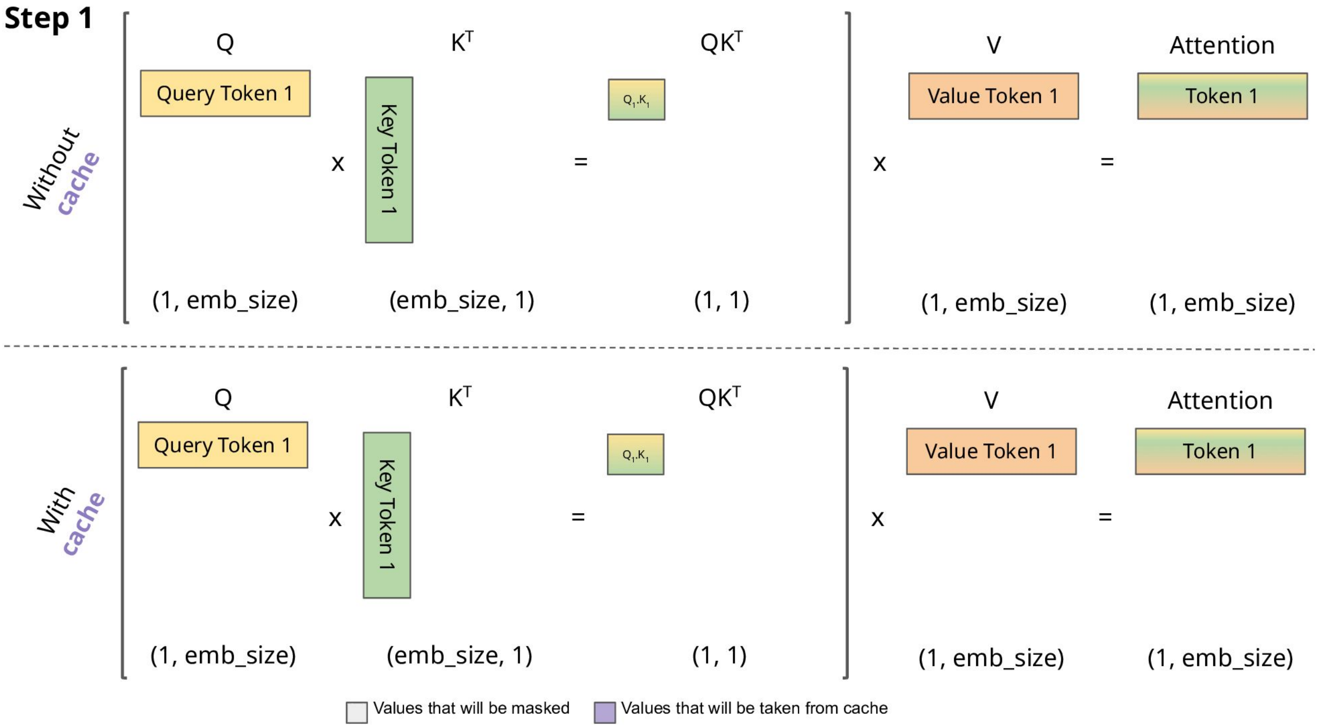
Multi-head Latent Attention (MLA)

Preliminary 2: Multi-head Attention


X
Thinking
Machines
Calculating attention separately in eight different attention heads
ATTENTION HEAD #0
ATTENTION HEAD #1
...
ATTENTION HEAD #7
Z_0
Z_1
Z_7
1) Concatenate all the attention heads
Z_0 Z_1 Z_2 Z_3 Z_4 Z_5 Z_6 Z_7
2) Multiply with a weight matrix W^O that was trained jointly with the model
X
W^O
3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN
=
Z

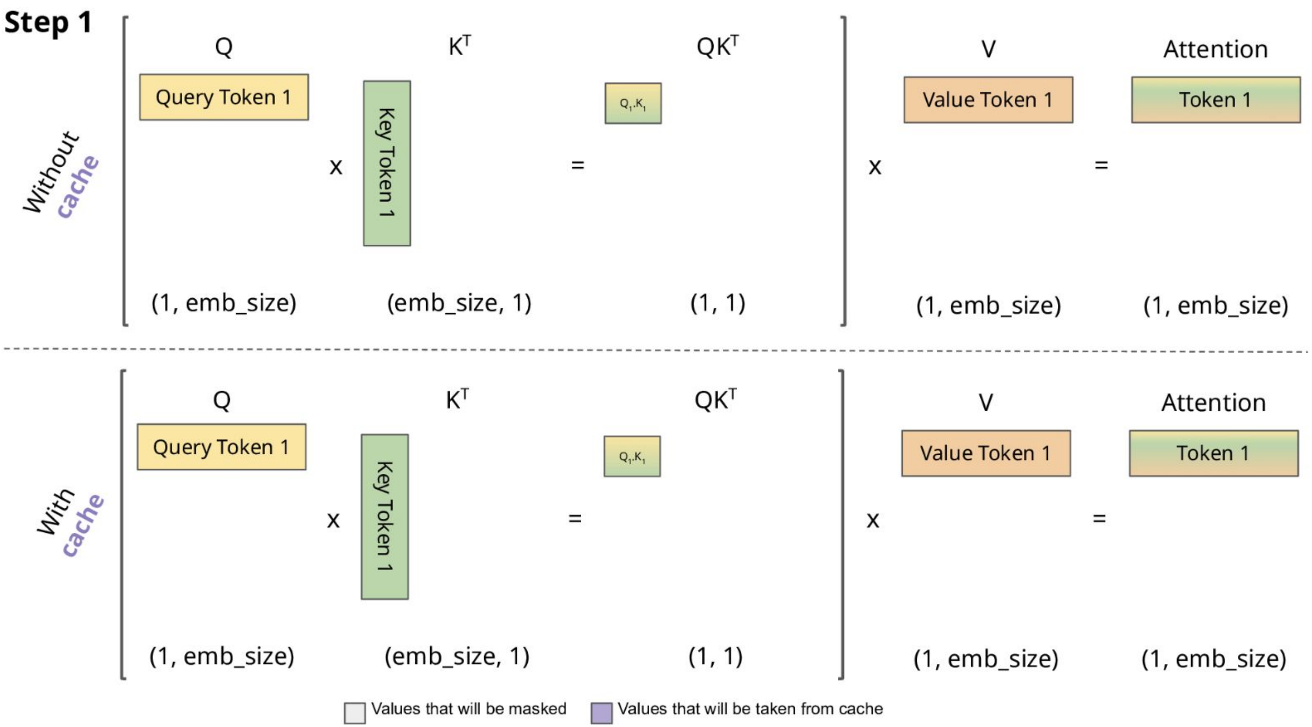
Multi-head Latent Attention (MLA)

Preliminary 3: Key-Value (KV) Caching



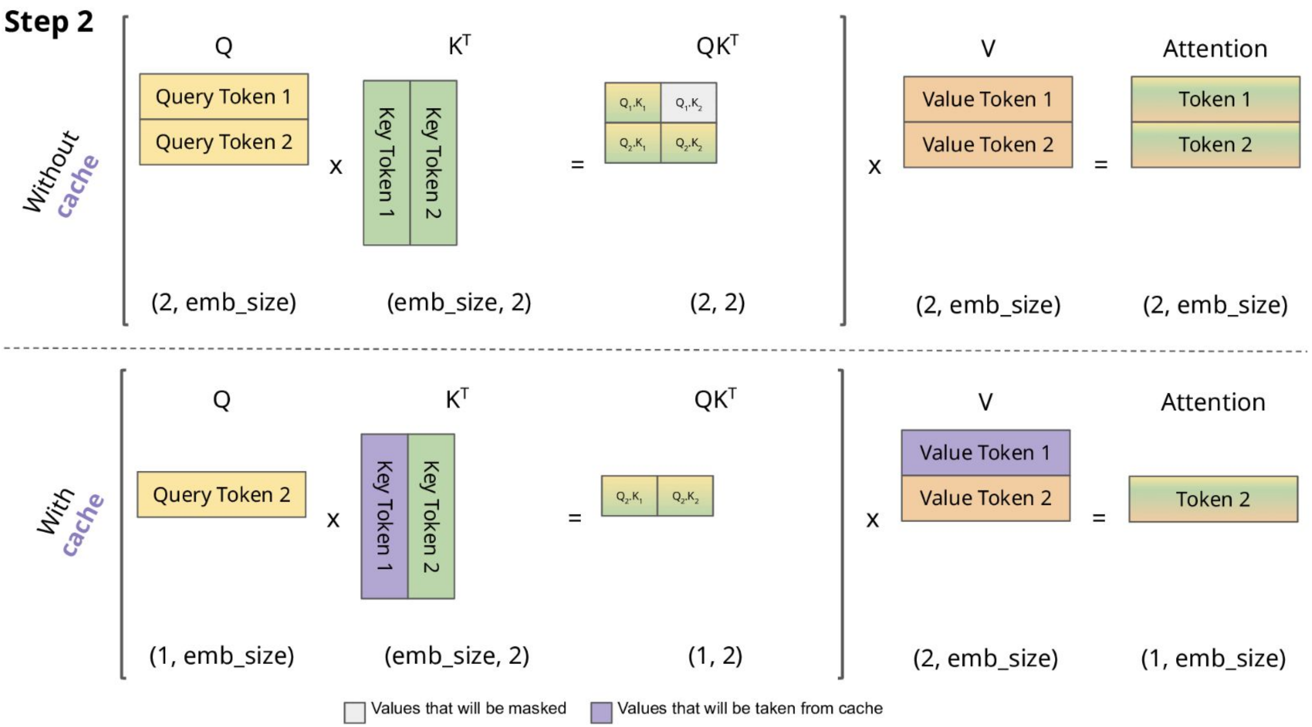
Multi-head Latent Attention (MLA)

Preliminary 3: Key-Value (KV) Caching



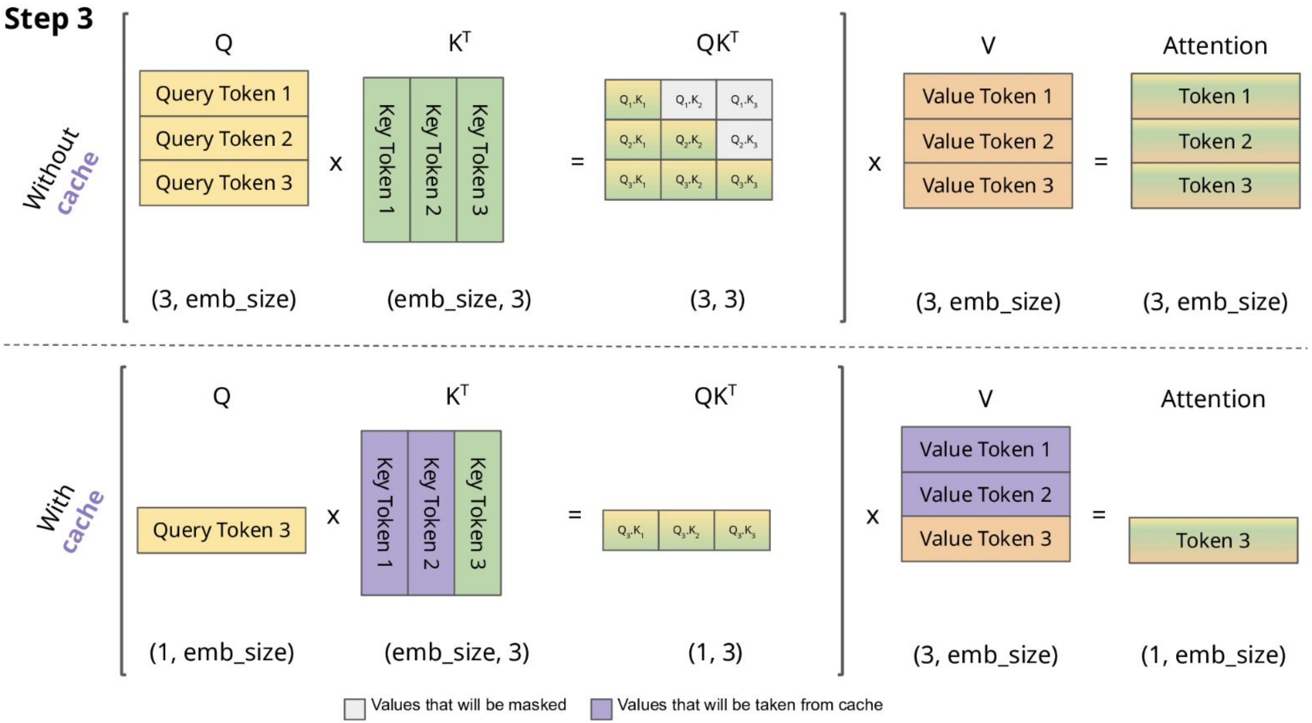
Multi-head Latent Attention (MLA)

Preliminary 3: Key-Value (KV) Caching



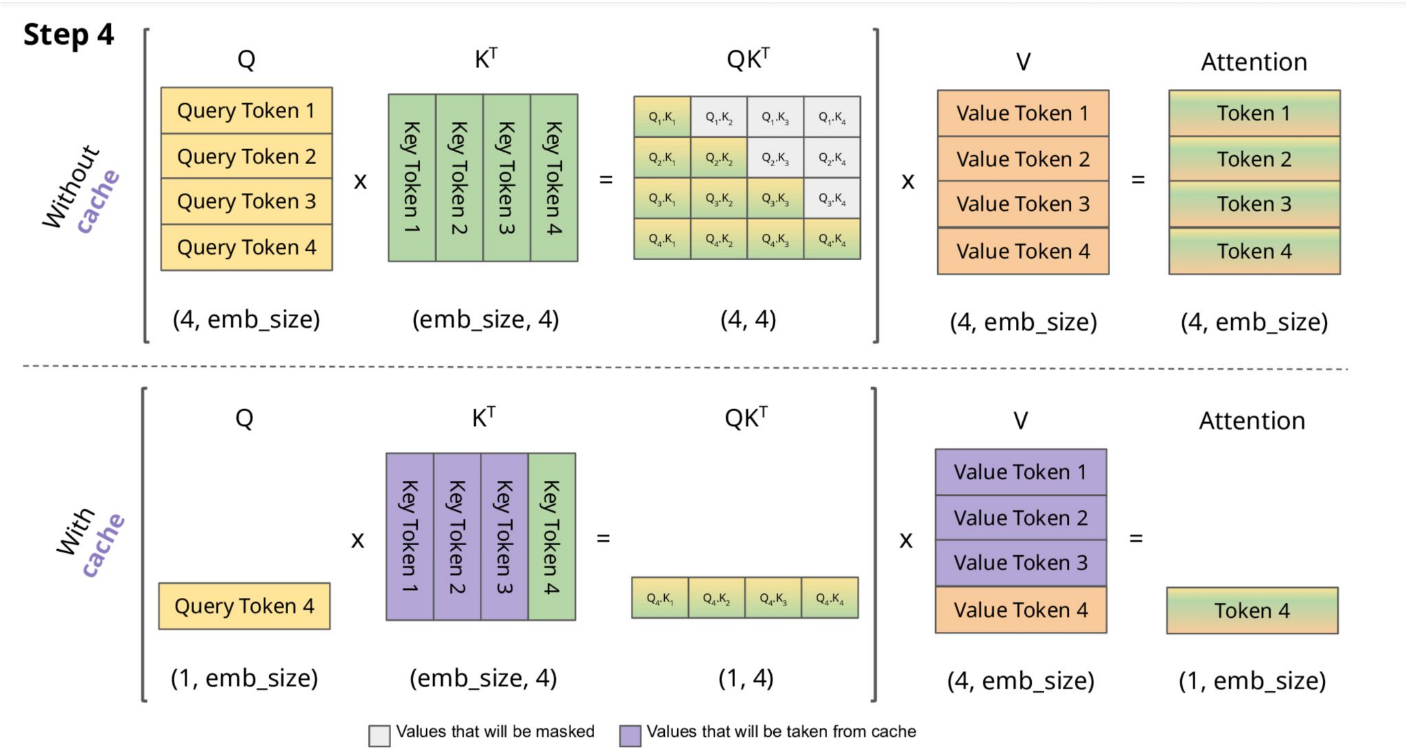
Multi-head Latent Attention (MLA)

Preliminary 3: Key-Value (KV) Caching



Multi-head Latent Attention (MLA)

Preliminary 3: Key-Value (KV) Caching



Multi-head Latent Attention (MLA)

Problem: High Memory Consumption

- As the context window grows, the KV cache scales its memory usage significantly, potentially straining system resources.

Motivation:

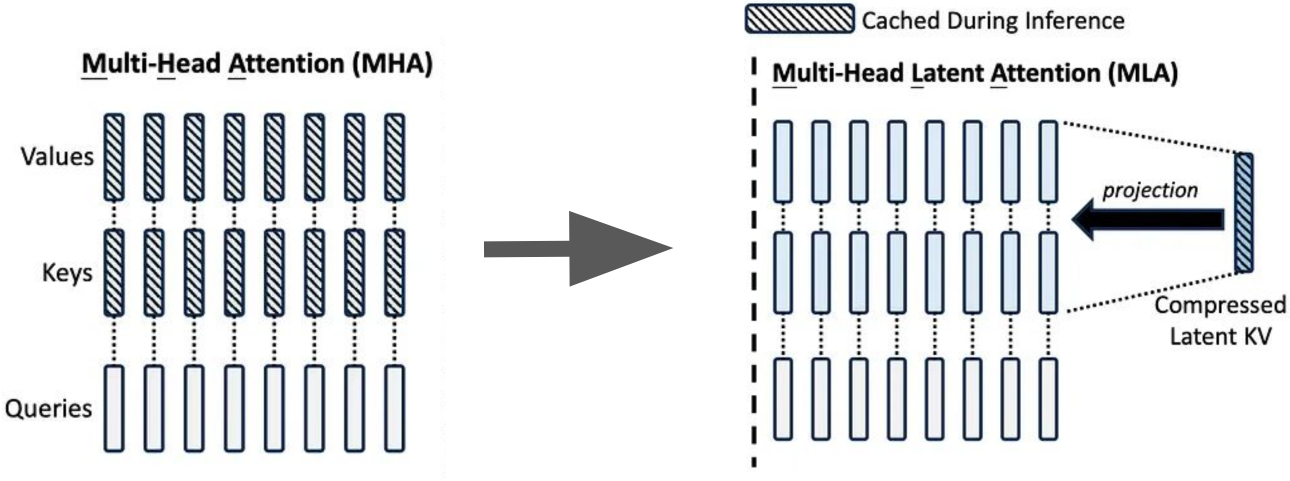
Is it possible to optimize or reduce the KV cache during inference without compromising performance?

Multi-head Latent Attention (MLA)

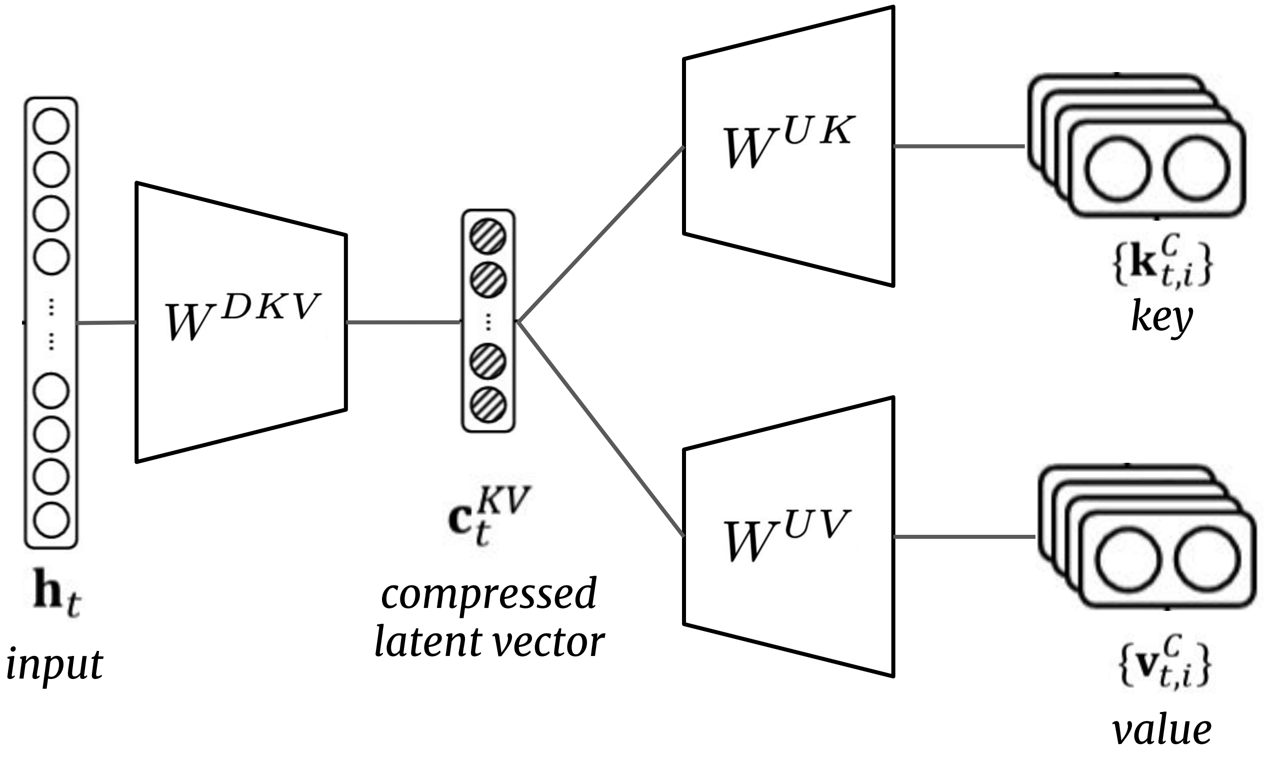
Can we reduce the KV cache during inference?

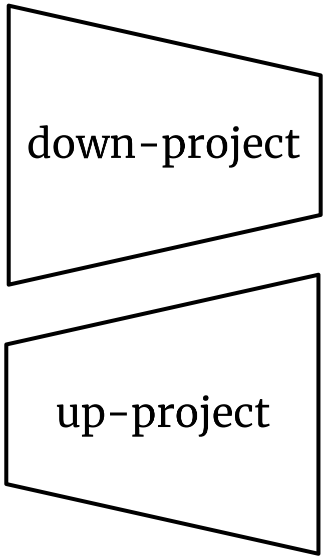
- 8 attention heads -> Cache information of 8 KV pairs

  Idea: Jointly compress the keys and values into a latent vector and project them back to original vector space.


▨ Cached During Inference
Multi-Head Attention (MHA)
Values
Keys
Queries
Multi-Head Latent Attention (MLA)
projection
Compressed Latent KV

Multi-head Latent Attention (MLA)





t: token position index
i: attention head index

Multi-head Latent Attention (MLA)

Do low-rank compression for key K and value V to a single vector and do up-projection to the original vector space during inference.

$$d_c << d_h n_h$$

Down-project to C (C: head-agnostic)

$$C_t^{KV} = W^{DKV} h_t$$

$\mathbf{c}_t^{KV} \in \mathbb{R}^{d_c}$ $\quad W^{DKV} \in \mathbb{R}^{d_c \times d}$

Up-project to K

$$K_t^C = W^{UK} C_t^{KV}$$

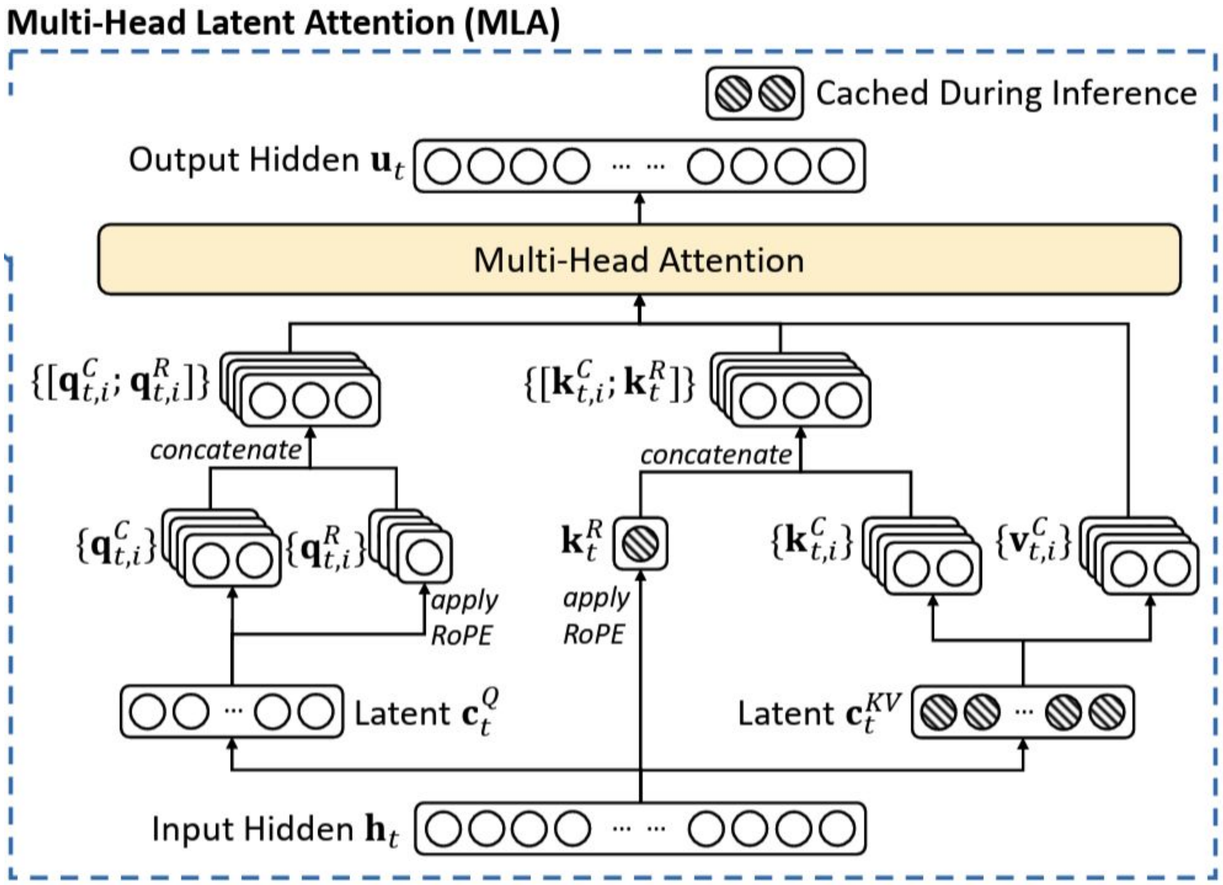
$W^{UK}, W^{UV} \in \mathbb{R}^{d_h n_h \times d_c}$

Up-project to V

$$V_t^C = W^{UV} C_t^{KV}$$

D: down-project
U: up-project

Multi-head Latent Attention (MLA)



Multi-head Latent Attention (MLA)

Full Picture:

- Query: In order to reduce the activation memory during training, they also perform low-rank compression for query Q, even if it cannot reduce the KV cache.
- Decoupled RoPE:Then use these additional vectors in the RoPE process only, while keeping the original keys isolated with the rotation matrix, and concatenate in the end.

$$
\begin{aligned}
\mathbf{c}_t^Q &= W^{DQ}\mathbf{h}_t, \\
[\mathbf{q}_{t,1}^C;\mathbf{q}_{t,2}^C;...;\mathbf{q}_{t,n_h}^C] = \mathbf{q}_t^C &= W^{UQ}\mathbf{c}_t^Q, \\
[\mathbf{q}_{t,1}^R;\mathbf{q}_{t,2}^R;...;\mathbf{q}_{t,n_h}^R] = \mathbf{q}_t^R &= \mathrm{RoPE}(W^{QR}\mathbf{c}_t^Q), \\
\mathbf{q}_{t,i} &= [\mathbf{q}_{t,i}^C;\mathbf{q}_{t,i}^R], \\
\boxed{\mathbf{c}_t^{KV}} &= W^{DKV}\mathbf{h}_t, \\
[\mathbf{k}_{t,1}^C;\mathbf{k}_{t,2}^C;...;\mathbf{k}_{t,n_h}^C] = \mathbf{k}_t^C &= W^{UK}\mathbf{c}_t^{KV}, \\
\boxed{\mathbf{k}_t^R} &= \mathrm{RoPE}(W^{KR}\mathbf{h}_t), \\
\mathbf{k}_{t,i} &= [\mathbf{k}_{t,i}^C;\mathbf{k}_t^R],
\end{aligned}
$$

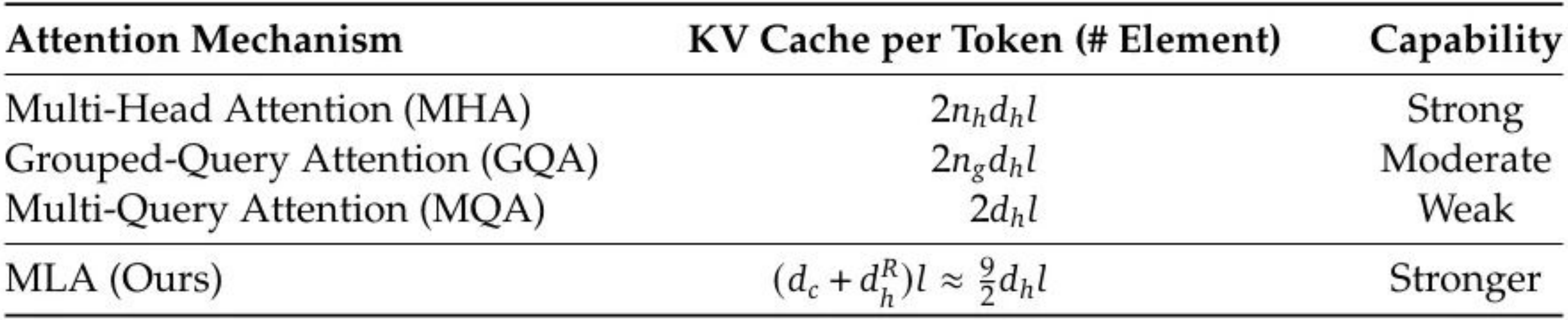
$$
\begin{aligned}
\mathbf{o}_{t,i} &= \sum_{j=1}^{t}\mathrm{Softmax}_j\left(\frac{\mathbf{q}_{t,i}^T\mathbf{k}_{j,i}}{\sqrt{d_h + d_h^R}}\right)\mathbf{v}_{j,i}^C, \\
\mathbf{u}_t &= W^O[\mathbf{o}_{t,1};\mathbf{o}_{t,2};...;\mathbf{o}_{t,n_h}],
\end{aligned}
$$

blue: cached components

Multi-head Latent Attention (MLA)

Performance vs classic MHA

- Better memory efficiency
- Better model capacity

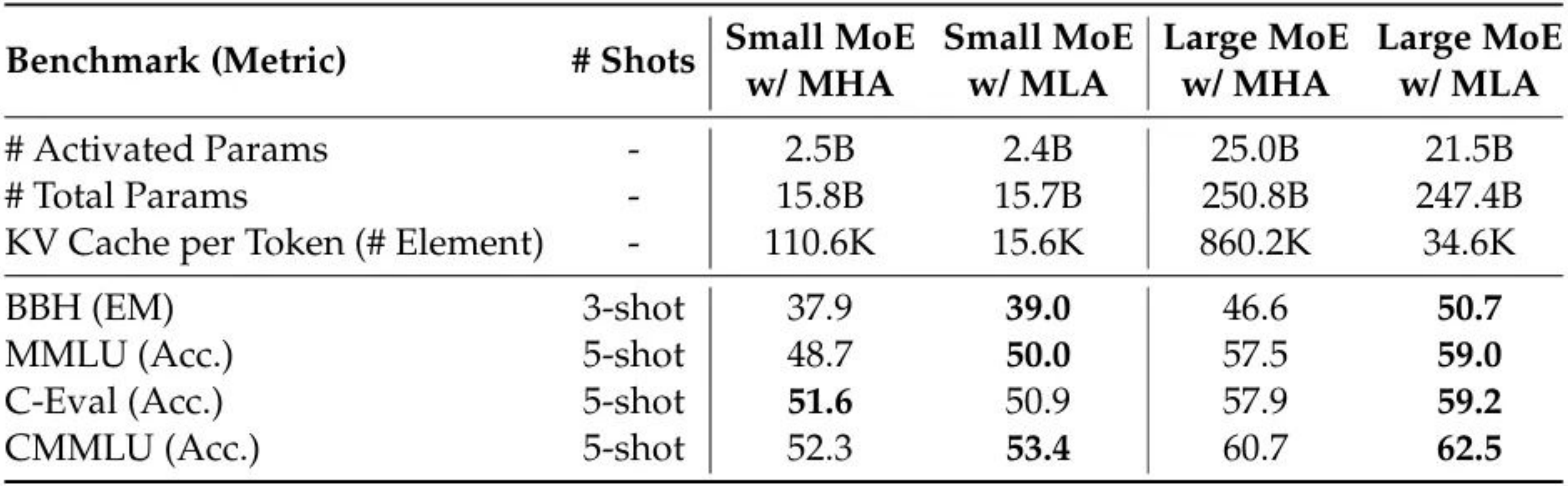


Attention Mechanism	KV Cache per Token (# Element)	Capability
Multi-Head Attention (MHA)	$2n_h d_h l$	Strong
Grouped-Query Attention (GQA)	$2n_g d_h l$	Moderate
Multi-Query Attention (MQA)	$2d_h l$	Weak
MLA (Ours)	$(d_c + d_h^R)l \approx \frac{9}{2}d_h l$	Stronger

Multi-head Latent Attention (MLA)

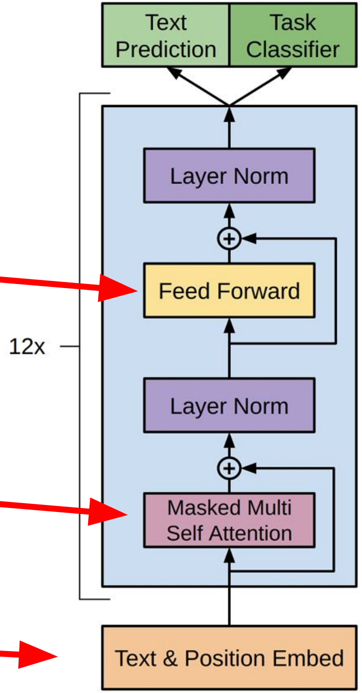
Performance vs classic MHA

- Better memory efficiency
- Better model capacity



Benchmark (Metric)	# Shots	Small MoE w/ MHA	Small MoE w/ MLA	Large MoE w/ MHA	Large MoE w/ MLA
# Activated Params	-	2.5B	2.4B	25.0B	21.5B
# Total Params	-	15.8B	15.7B	250.8B	247.4B
KV Cache per Token (# Element)	-	110.6K	15.6K	860.2K	34.6K
BBH (EM)	3-shot	37.9	**39.0**	46.6	**50.7**
MMLU (Acc.)	5-shot	48.7	**50.0**	57.5	**59.0**
C-Eval (Acc.)	5-shot	**51.6**	50.9	57.9	**59.2**
CMMLU (Acc.)	5-shot	52.3	**53.4**	60.7	**62.5**

DeepSeek MoE



5. Output

Representation

4. Repeat

3. MLP

2. Self-Attention

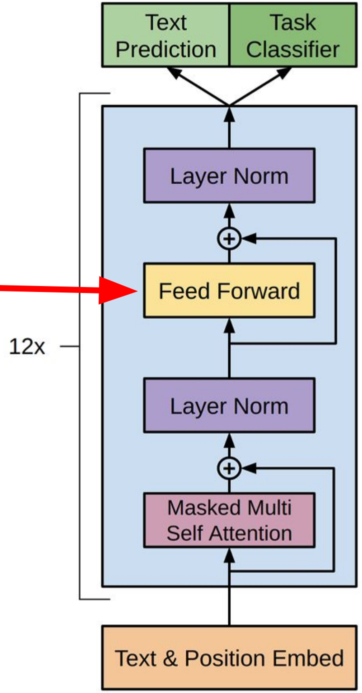
1. Input Embeddings

(each token = vector)

Decoder-only
transformer (GPT)

DeepSeek MoE: Feed Forward

Most of the parameters are here



Decoder-only transformer (GPT)

DeepSeek MoE: Architecture

- Imagine the human brain.
- We have distinct parts responsible for different tasks.
- For a specific task, we can invoke just the specific part of the brain to do the work
- This approach is more scalable & efficient since you only use a subset of available neurons
- Instead of having one FFN, we have multiple (relatively) smaller FFNs (now called experts)

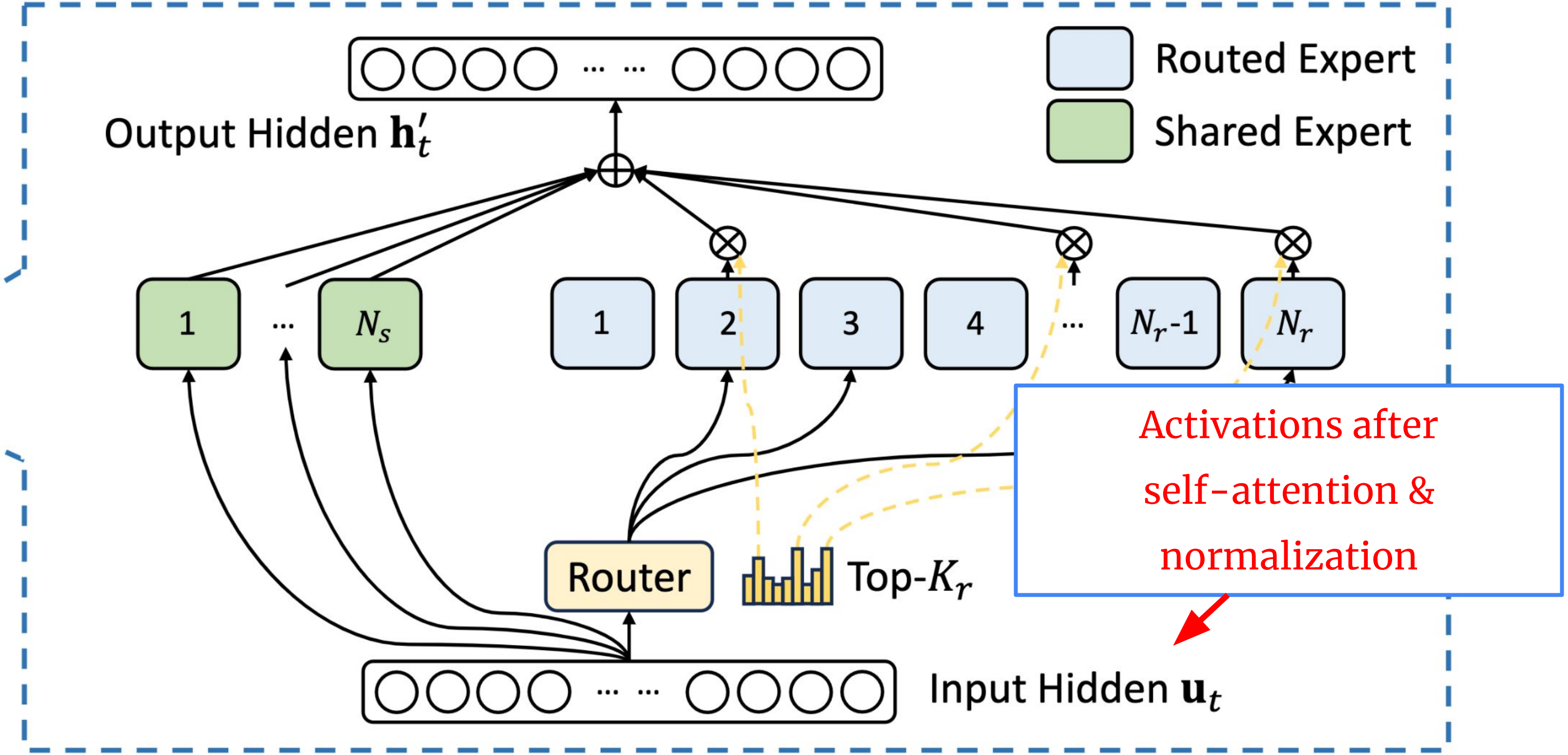
DeepSeek MoE: Architecture

- Previous MoE architectures had limited experts (N=2 to 16) with only ~2 of them being active for a token
- This did bring the efficiency gains.
- But, there are two problems:
  - Knowledge Hybridity:
    - Each expert is not really specific to a certain task.
    - It’s still quite big and holds diverse, hybrid knowledge.
  - Knowledge Redundancy
    - Same information is stored across multiple experts.
- **2-16 experts is not enough!**

DeepSeek MoE: Architecture

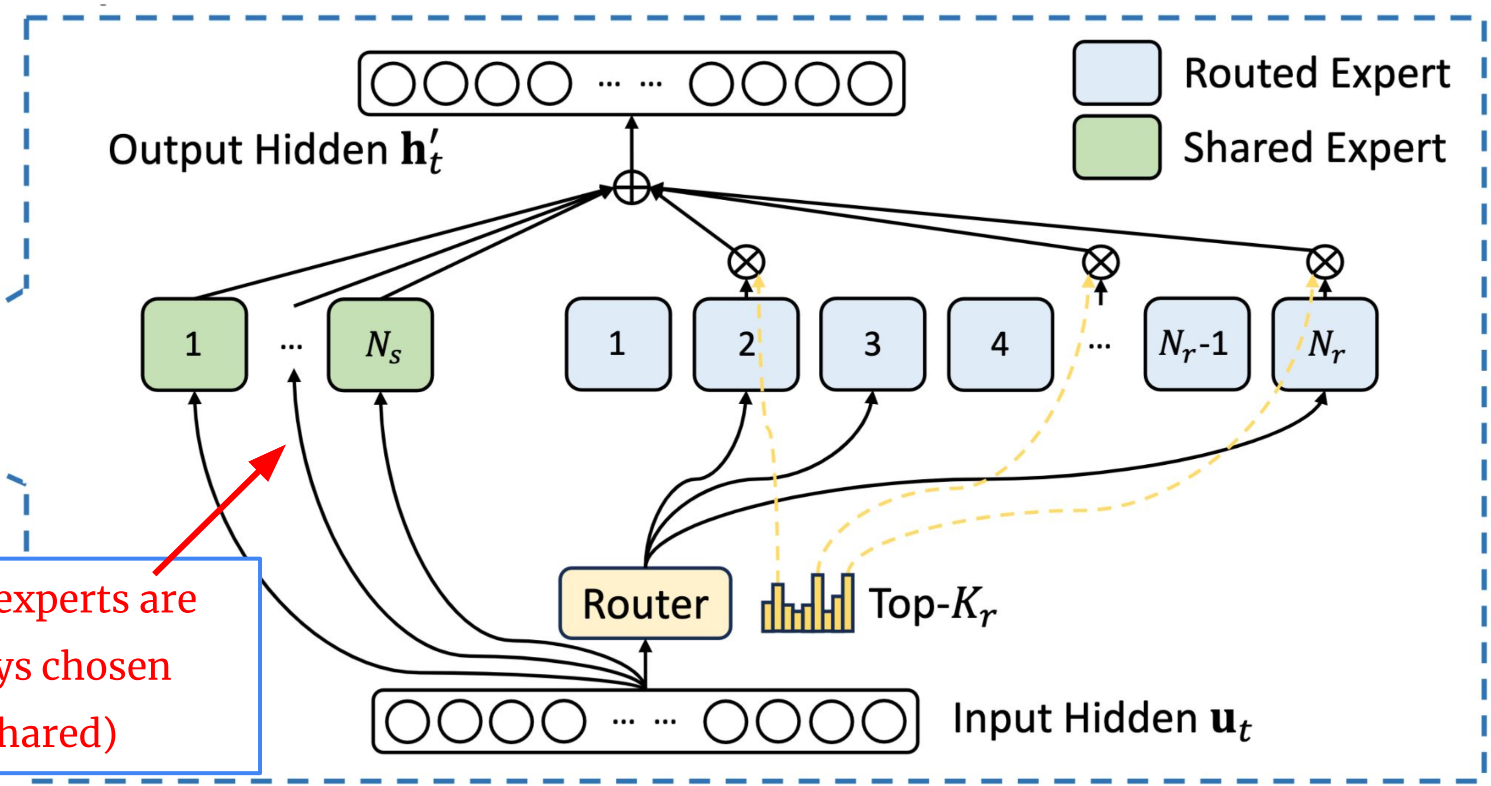
- To combat this, Deepseek introduces two new innovations:
  - Fine-grained expert segmentation (combat K. Hybridity)
    - Each expert is split into multiple smaller ones, keeping parameter count the same.
    - With more experts, knowledge can be decomposed and each expert can get more specialized in a certain area.
  - Shared experts
    - Some experts are always activated and can focus on common knowledge (combat K. Redundancy)

DeepSeek MoE



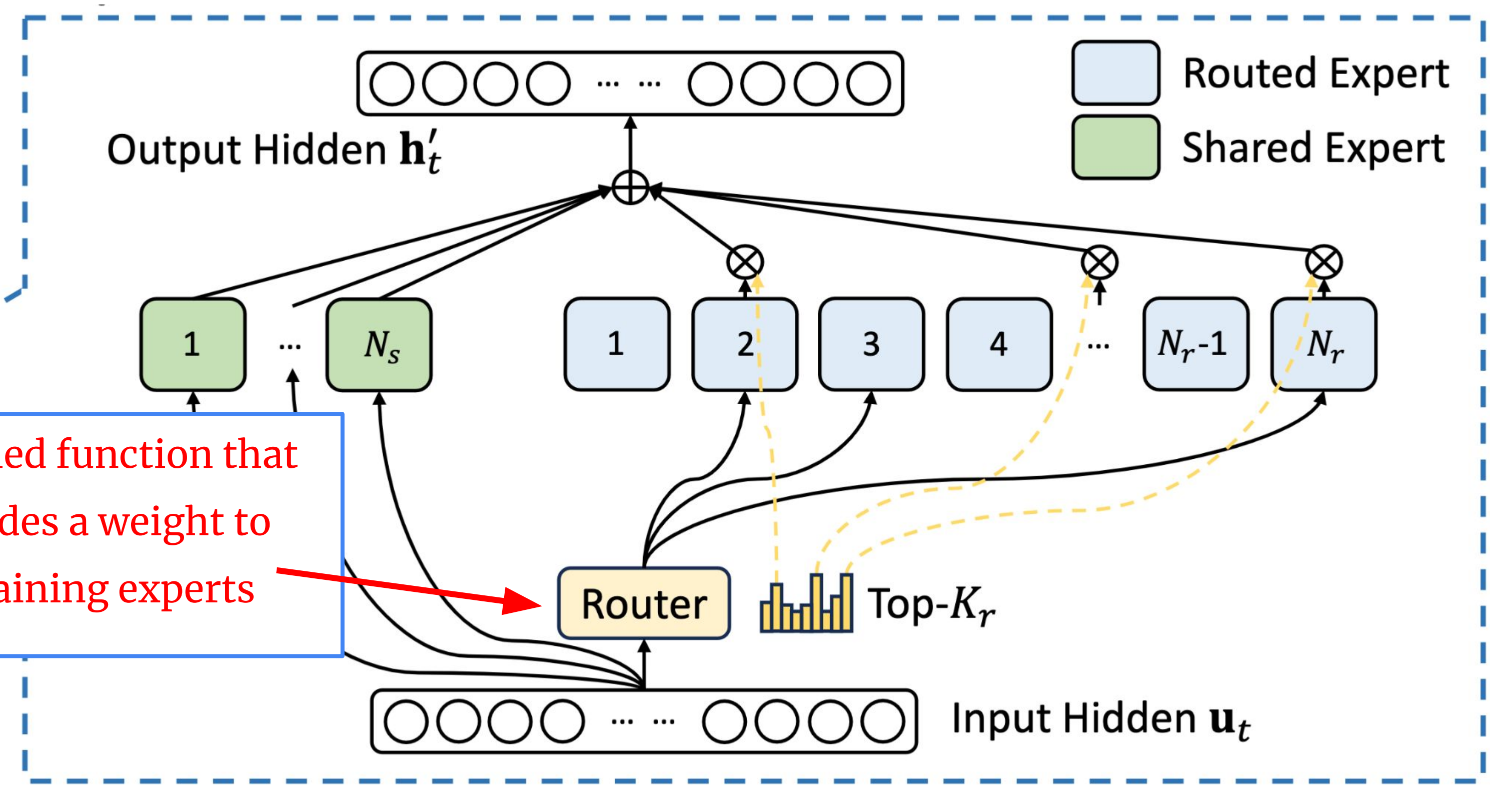
Activations after self-attention & normalization

DeepSeek MoE



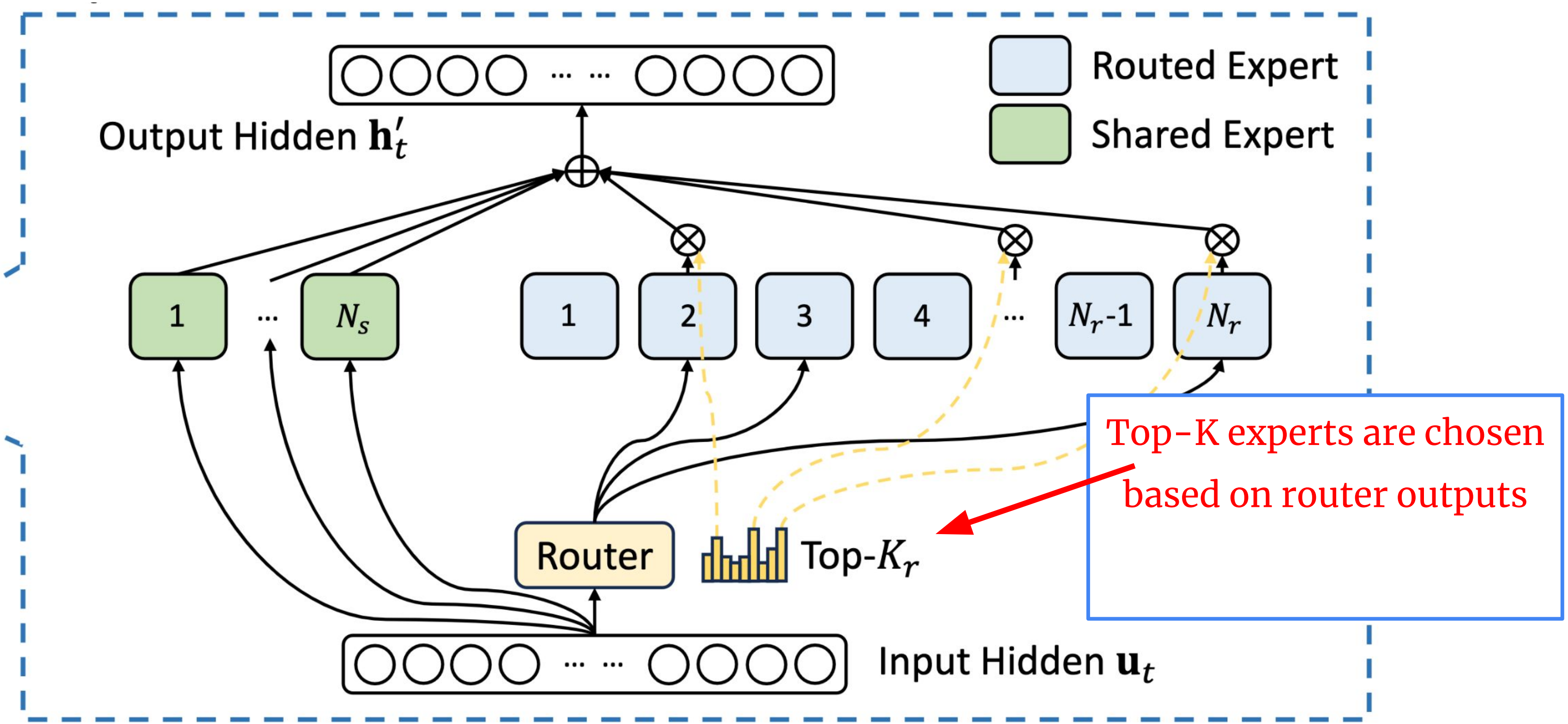
Some experts are always chosen (shared)

DeepSeek MoE



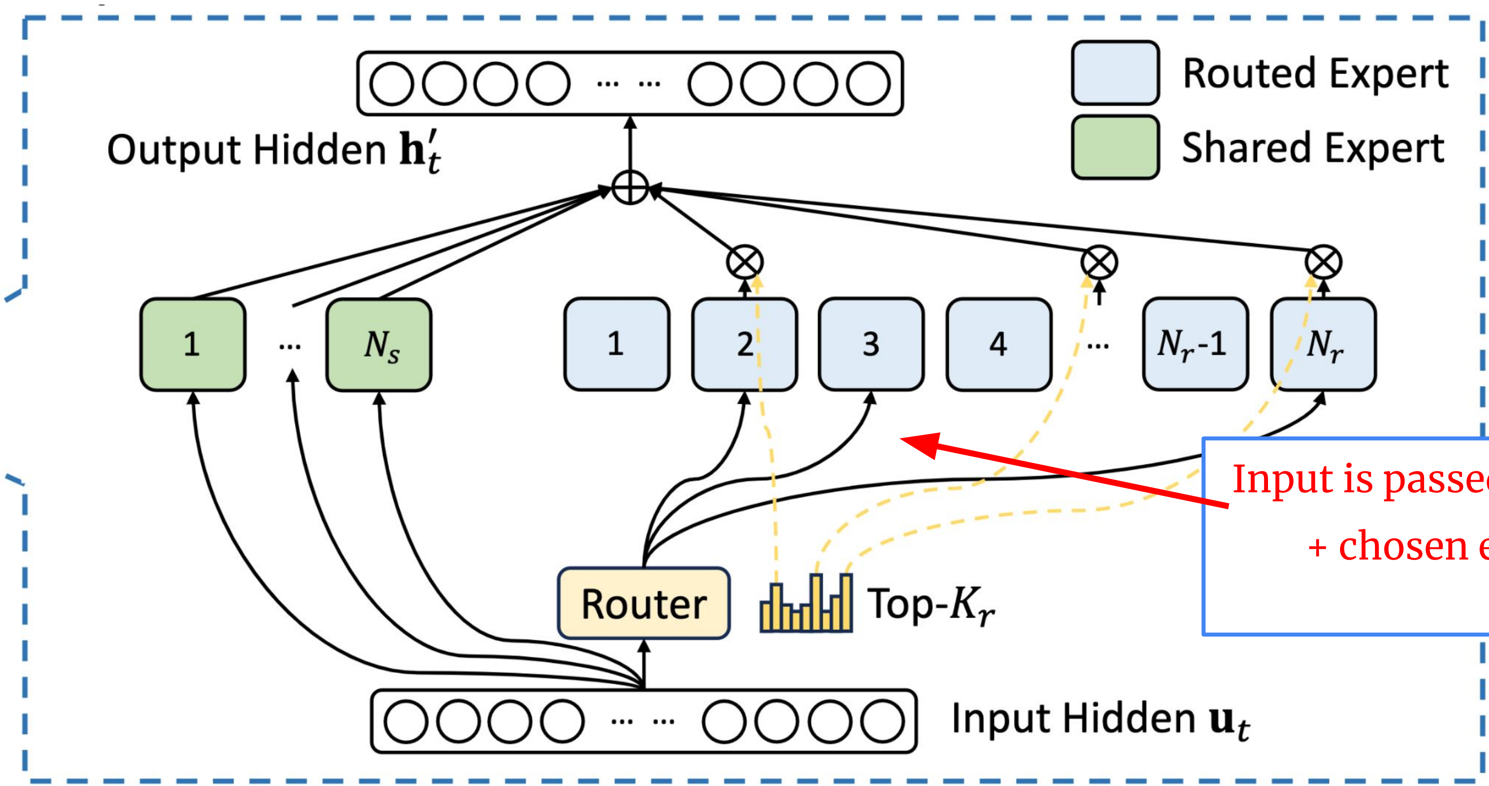
A learned function that provides a weight to remaining experts

DeepSeek MoE



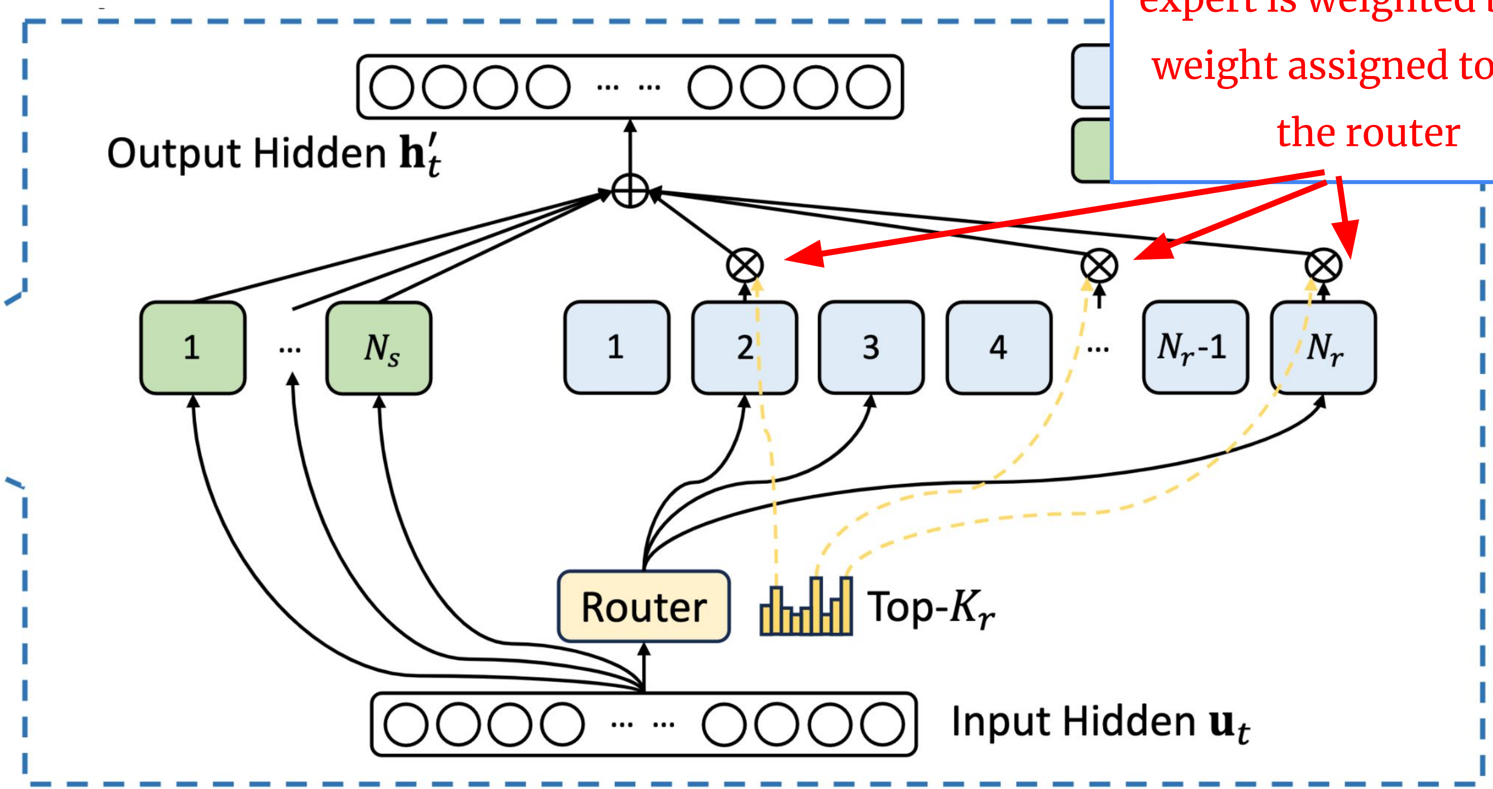
Top-K experts are chosen based on router outputs

DeepSeek MoE



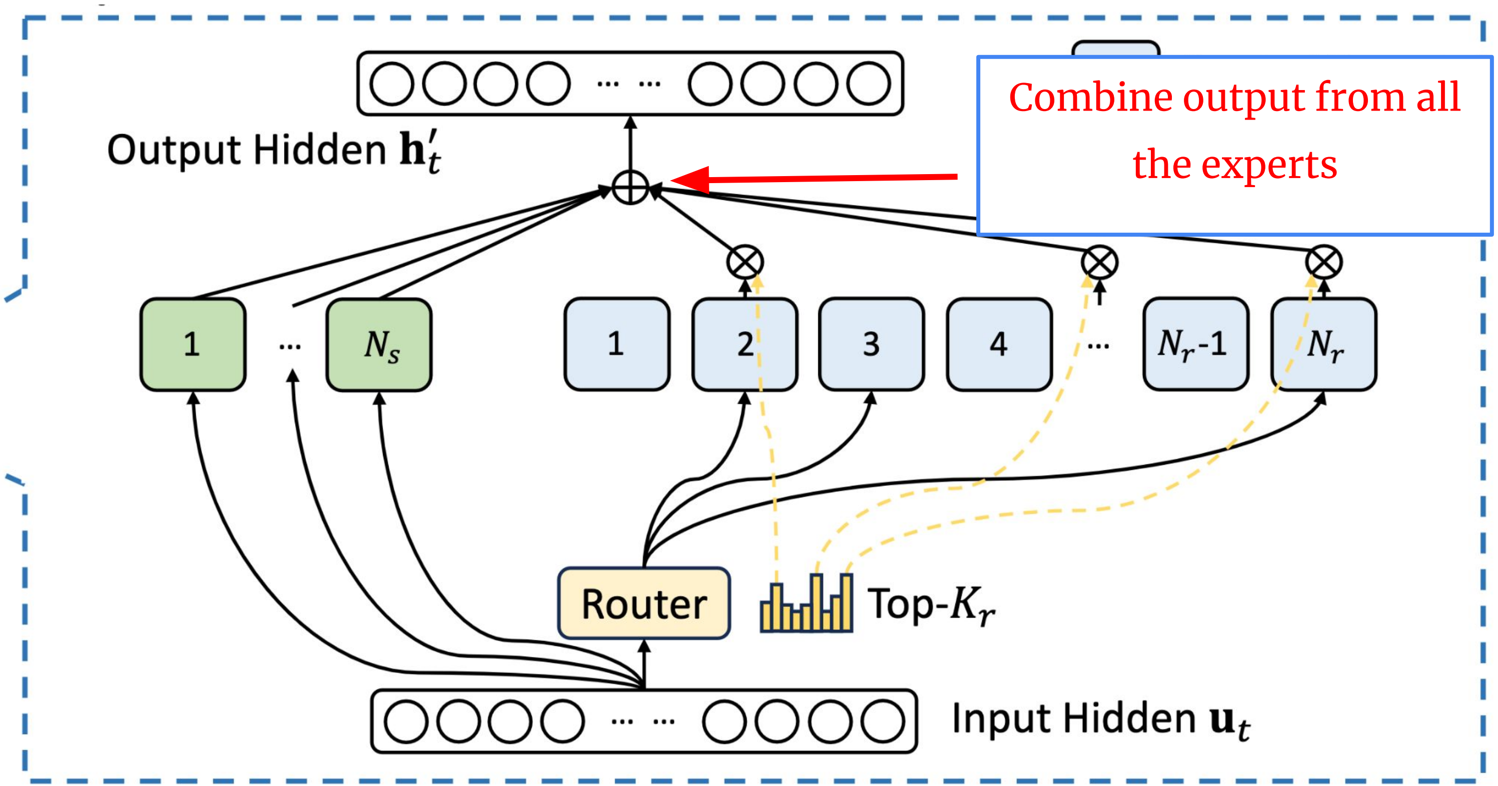
Input is passed to shared + chosen experts

DeepSeek MoE



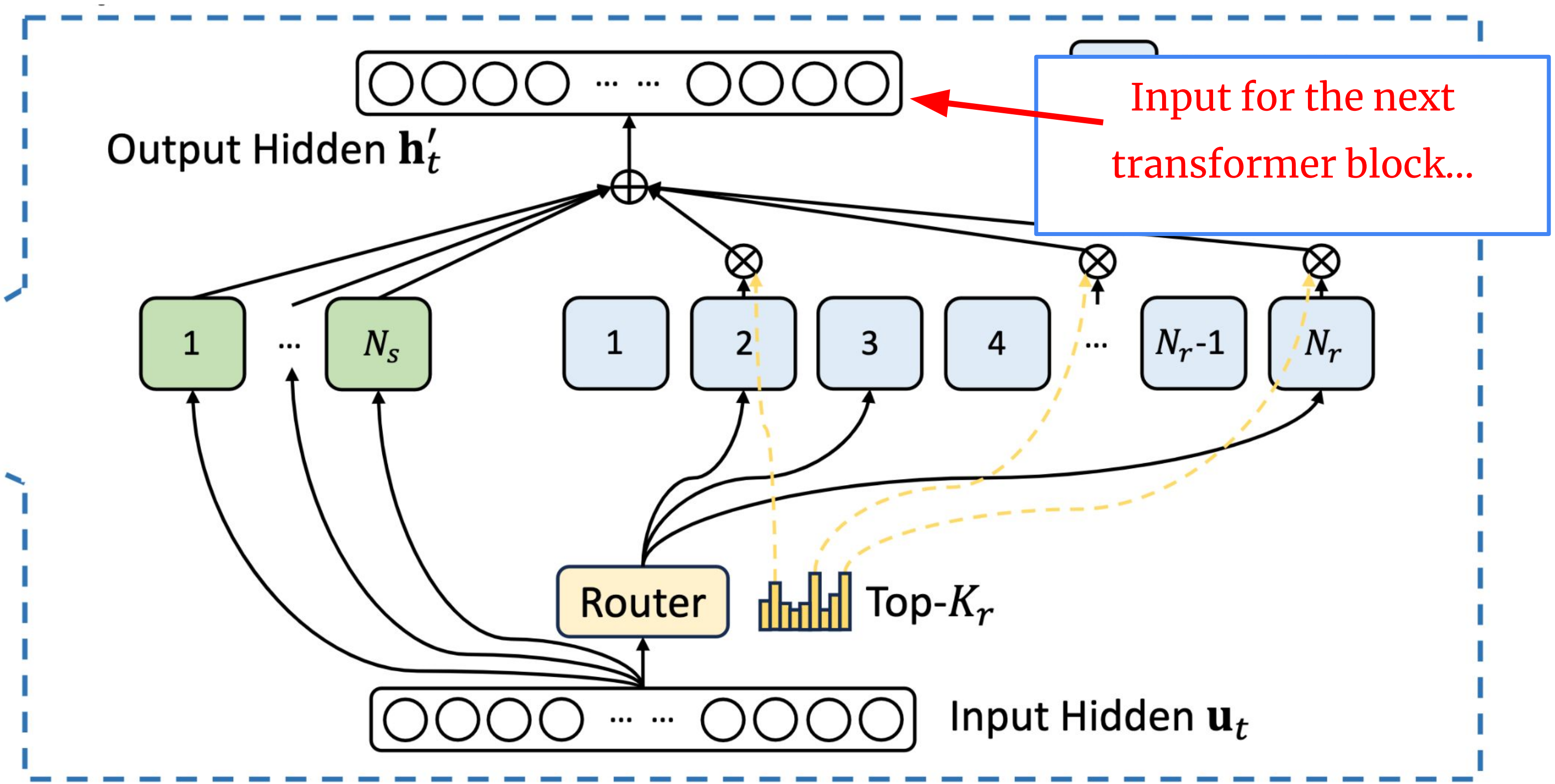
Output from each routed expert is weighted by the weight assigned to it by the router

DeepSeek MoE



Combine output from all the experts

DeepSeek MoE



Input for the next transformer block...

DeepSeek MoE

$\mathbf{u}_t$ FFN-Input of the t-th token (output from self-attention layer)

$\mathbf{e}_i$ Centroid vector of the i-th routed expert

(you can imagine this as a learned vector representation of the kind of tokens a certain expert is meant to deal with)

Taking a Dot Product
(How similar is this token to the kind of tokens expert-i is used to dealing with?)

$$\mathbf{u}_t{}^T \mathbf{e}_i$$

DeepSeek MoE

$\mathbf{u}_t$ FFN-Input of the t-th token (output from self-attention layer)

$\mathbf{e}_i$ Centroid vector of the i-th routed expert

Pass the similarity (affinity) score of token ‘t’ with expert ‘i’ through sigmoid to get a value from 0 to 1

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

DeepSeek MoE

$\mathbf{u}_t$ FFN-Input of the t-th token (output from self-attention layer)

$\mathbf{e}_i$ Centroid vector of the i-th routed expert

$s_{i,t}$ Affinity score of token-t with expert-i

Choose only TopK (K_r) experts based on affinity scores

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

DeepSeek MoE

$\mathbf{u}_t$ FFN-Input of the t-th token (output from self-attention layer)

$\mathbf{e}_i$ Centroid vector of the i-th routed expert

$s_{i,t}$ Affinity score of token-t with expert-i

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

Run softmax (without the exponential) over the N_r routed experts, getting a probability distribution over the routed experts. This is referred to as the 'gating value'

DeepSeek MoE

$\mathbf{u}_t$ FFN-Input of the t-th token (output from self-attention layer)

$g_{i,t}$ Gating value for expert-i with token-t

Putting everything together...

DeepSeek MoE

$\mathbf{u}_t$ FFN-Input of the t-th token (output from self-attention layer)

$g_{i,t}$ Gating value for expert-i with token-t

$$\mathbf{h}'_t = \mathbf{u}_t + \underbrace{\sum_{i=1}^{N_s} \mathrm{FFN}_i^{(s)}(\mathbf{u}_t)}_{\text{Pass input through the N\_s shared experts and combine them}} + \sum_{i=1}^{N_r} g_{i,t}\,\mathrm{FFN}_i^{(r)}(\mathbf{u}_t)$$

DeepSeek MoE

$\mathbf{u}_t$ FFN-Input of the t-th token (output from self-attention layer)

$g_{i,t}$ Gating value for expert-i with token-t

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

Pass input through the N_r routed experts

and do weighted sum based on gating value

DeepSeek MoE

$\mathbf{u}_\mathbf{t}$ FFN-Input of the t-th token (output from self-attention layer)

$g_{i,t}$ Gating value for expert-i with token-t

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

Add the residual (original input)

DeepSeek MoE

- DeepSeek v3 has 61 transformer blocks.
- Except first 3 FFNs, all are substituted with MoE.
- Each MoE layer has **1 shared expert** and **256 routed experts.**
  - Some details were abstracted earlier. In actual, we have 8 groups (large experts) among which 2 are chosen first. So, each group has 256 / 8 = 32 smaller experts.
- Among the routed experts, **8** will be chosen for each token.
- Intermediate dimension of each expert is just **2048.**
- So, although, DeepSeek V3 comprises of 671B parameters, **only 37B are activated for each token.**

Conclusion

- **Multi-Head Latent Attention (MLA)** dramatically reduces memory usage compared to traditional attention mechanisms
- **Mixture of Experts (MoE)** architecture activates only 37B of 671B total parameters per token, optimizing resource utilization
- DeepSeek v3 achieves state-of-the-art performance while significantly lowering computational costs

References

1. DeepSeek-AI. (2024). DeepSeek-V3 Technical Report. (link)
2. DeepSeek-AI. (2024). DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model. (link)
3. Dai, D., Deng, C., Zhao, C., Xu, R. X., & Gao, H. (2024). DeepSeekMoE: Towards Ultimate Expert Specialization in Mixture-of-Experts Language Models. (link)
4. Jiang, A. Q., Sablayrolles, A., Roux, A., Mensch, A., & Savary, B. (2024). Mixtral of Experts (link).