DeepSeek V3

Liu, Aixin, et al. "Deepseek-v3 technical report." arXiv preprint arXiv:2412.19437 (2024).

Presented by Adnan Al Ali

Overview

- **Mixture-of-Experts** language model
- **671 B** parameters, **37 B** activated for each token
- **Cost-effective** training and efficient inference
- **New state-of-the-art** reached on certain benchmarks
- Together with DeepSeek R1 strongly **impacted the LLM/tech market**

Related Work

What lead to DeepSeek V3?

The Transformer

Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (**2017**).

What the Transformer introduced

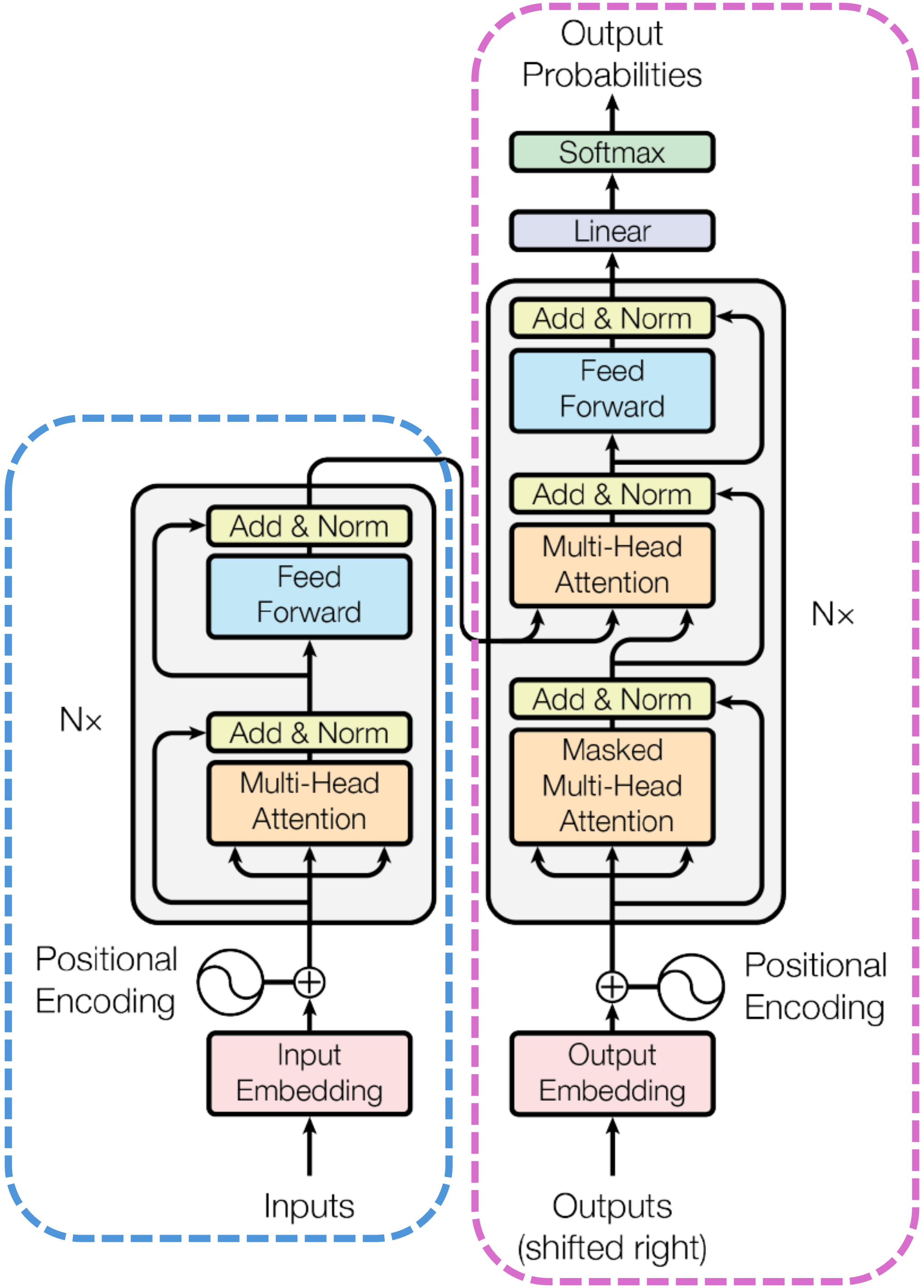
- Originally an architecture for machine translation (MT)
- Replaced the RNNs and CNNs (popular in NLP at the time) by **attention mechanism** alone[1]:
  - RNNs are difficult to parallelize
  - CNNs struggle with long distance relationships
- Encoder-decoder architecture

[1] The attention mechanism had been used before, in combination with other modules

How the Architecture Works

- The input is tokenized into **sub-word tokens** from a fixed-size vocabulary and embedded into a vector $\in \mathbb{R}^{d_{\text{model}}}$
- **Positional encodings** are added (attention mechanism is position-unaware by default)
- The encoder calculates a contextualized vector representation for each input token $\in \mathbb{R}^{d_{\text{model}}}$
- The decoder starts with an empty sequence and uses its previous outputs (**autoregressively** on inference) and the outputs of the encoder to generate the **next token**
- In the decoder, only tokens can attend to **earlier tokens only**

Encoder



Decoder

Source: Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

Scaled Dot-Product Attention

- The vector token representations are linearly transformed into 3 matrices: **Q**ueries, **K**eys, and **V**alues
- "**Compatibility**" between the Keys and Queries:

$$\operatorname{softmax}\left(\frac{QK^T}{\sqrt{d_{keys}}}\right)$$

- Compatibility is used to weight the values as $\operatorname{softmax}\left(\frac{QK^T}{\sqrt{d_{keys}}}\right)V$
- One attention layer contains h such **attention heads** — the outputs are concatenated and linearly transformed back to d_{model}

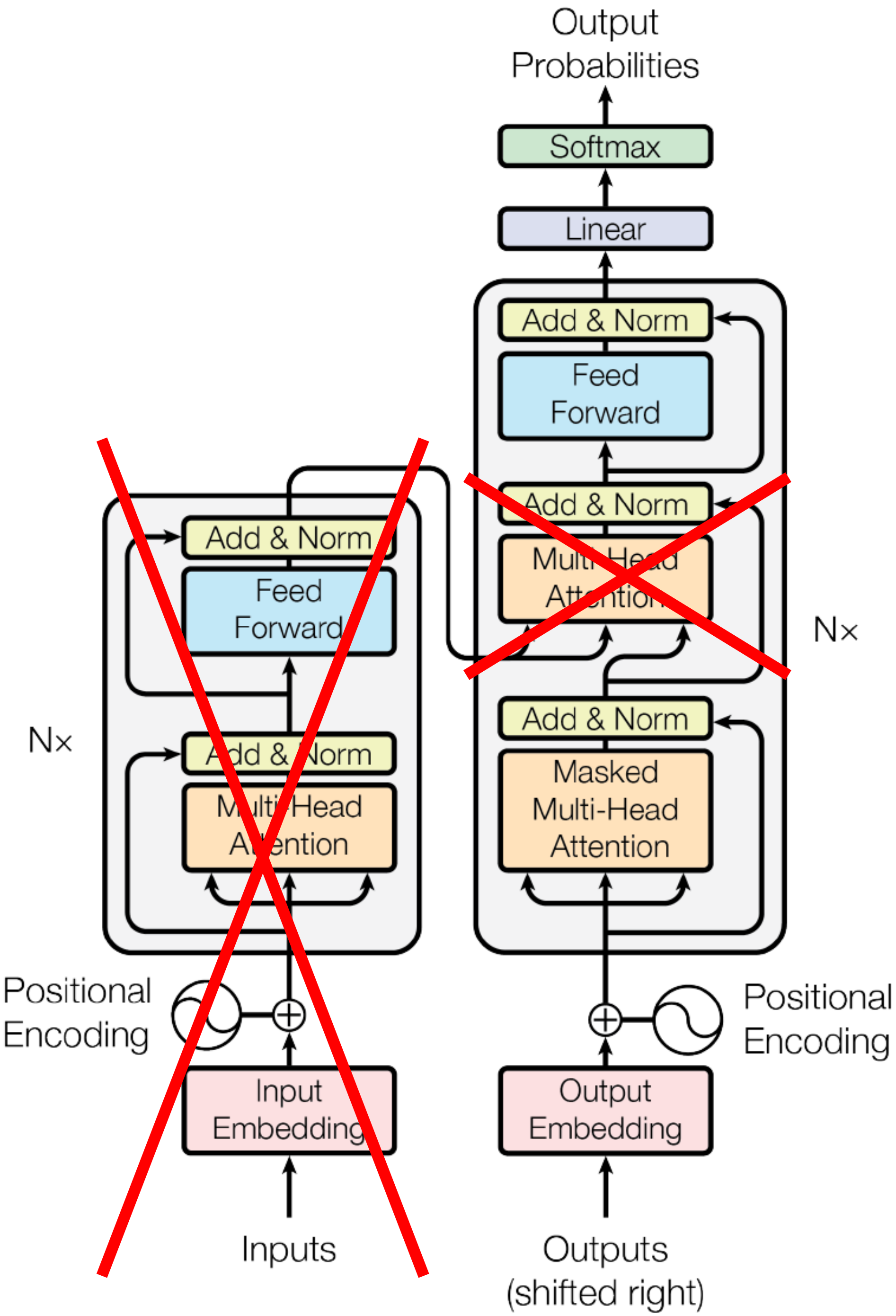
Quiz question: what was the first decoder-only model?

Decoder-only Model

Liu, Peter J., et al. "Generating wikipedia by summarizing long sequences." arXiv preprint arXiv:1801.10198 (**2018**).

Multi-document Summarization

- Task: given a collection of source texts, generate a **Wikipedia-style summary**
- Authors **drop the encoder** part entirely, instead feed the input tokens directly into the decoder as sequences:
  `#Source1` `lorem` `ipsum` `…` `#Source2` `dolor` `sit` `…` `[SEP]` `#Wikipedia` `amet` `consectetur` `…`
- During training, predicting all tokens, including the sources
- On inference, the sources are given as if they were already generated



Source: Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

Mixture of Experts

Dai, Damai, et al. "Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models." arXiv preprint arXiv:2401.06066 (2024).

Mixture of Experts

- Feed-forward (FFN) layers constitute **two-thirds** of a transformer model's **parameters** and store **factual information**[1]
- The aim is to substitute them with a **MoE layer** — a set of $\boldsymbol{N}$ smaller FFN layers of which only a subset of $\boldsymbol{K}$ is used for each token
- Output from the Self-Att layer for token t: u_t
- **Token-to-expert** affinity: $s_{i,t} = \mathrm{sigmoid}(u_t^T c_i)$
- **Top K** experts with the highest $s_{i,t}$ are considered:

$$h_t = \sum_{j \in \mathrm{TopK}_i(s_{i,t})} \left(\frac{s_{j,t}}{\alpha_t} \mathrm{FFN}_j(u_t) \right) + u_t$$

learned "expert centroid"

residual connection

normalizing factor

[1]Geva, Mor, et al. "Transformer feed-forward layers are key-value memories." arXiv preprint arXiv:2012.14913 (2020).

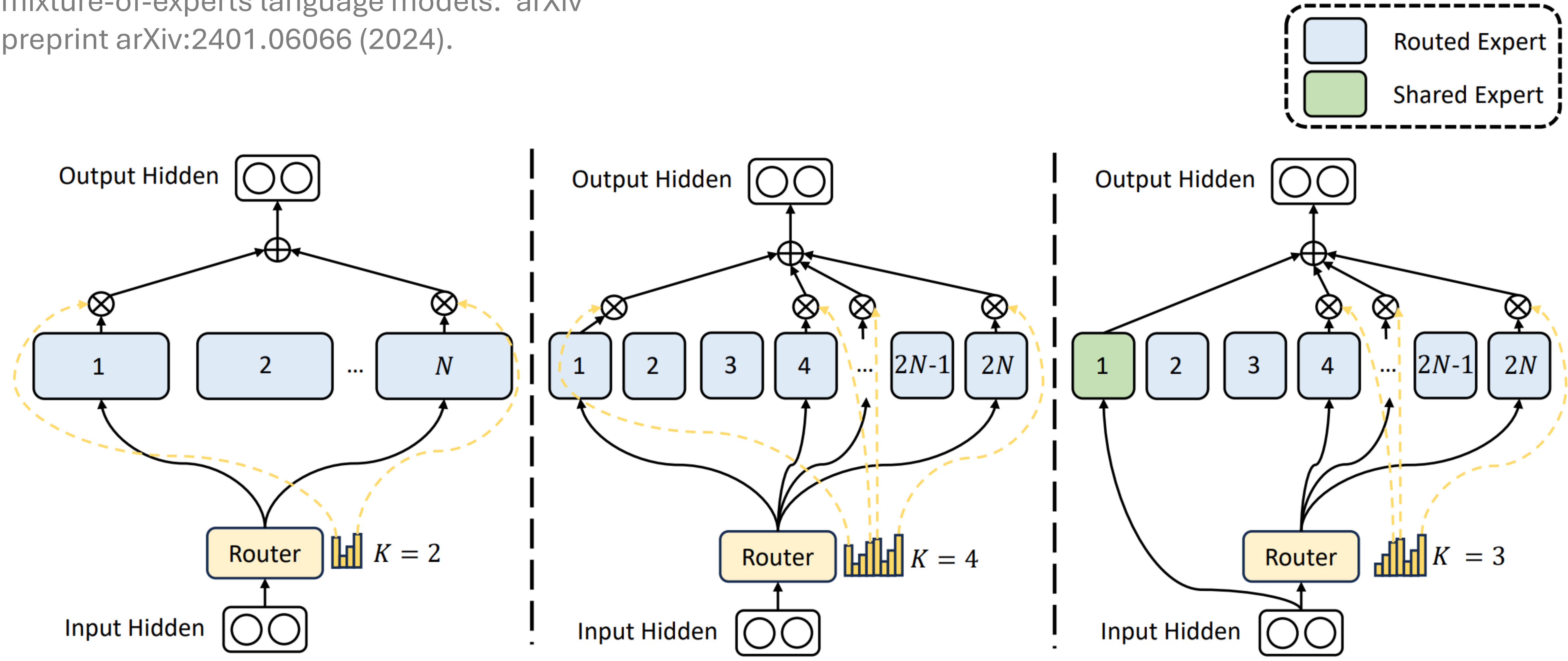
MoE Challenge: Knowledge Hybridity

- Previous architectures had a **small number of experts** (8 or 16) which had to cover diverse knowledge → hard to utilize at once
- Solution: **fine-grained expert segmentation**:
  - Segment each expert into m equally sized experts ($\frac{1}{m}$ of the original size)
  - Increase K' to mK
- Why it works — combinatorial explosion/**flexibility**:
  - For $N = 16, K = 2$ the number of expert combinations is $\binom{16}{2} = 120$
  - Fine-grained by $\mathrm{m} = 4$: $\binom{64}{8} = 4,426,165,368$

MoE Challenge: Knowledge Redundancy

- Some knowledge is **required for all/most tokens** and under the conventional architecture, all experts have to learn it
- Solution: **shared expert isolation**:
  - A small number (K_s) of experts is activated for each token
  - The remaining $K - K_s$ experts are selected excluding the shared ones

Source: Dai, Damai, et al. "Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models." arXiv preprint arXiv:2401.06066 (2024).



(a) Conventional Top-2 Routing → **(b) + Fine-grained Expert Segmentation** → **(c) + Shared Expert Isolation (DeepSeekMoE)**

MoE challenge: routing collapse

- Automatically learned expert routing may lead to **repetitive selection** of a **few experts** regardless of the token
- Solution: expert-level balance loss:
  - Per-token average expert affinity: $P_i = \frac{1}{T}\sum_{t=1}^{T}\frac{s_{i,t}}{\alpha_t}$
  - Per-token average expert utilization: $f_i = \frac{(N-K_S)}{(K-K_S)}\frac{1}{T}\sum_{t=1}^{T}\mathbb{1}[\text{Token } t \text{ selects Expert } i]$
  - Loss function: $\mathcal{L}_{\text{ExpBal}} = \eta_1 \sum_i f_i P_i$

indicator function

Quiz question: what's the goal of the DeepSeekMoE architecture?

a) Adding more knowledge to the model
b) Saving GPU memory
c) Decreasing the number of computations
d) Explicitly assigning expertise to parts of the model

Quiz question: what's the goal of the DeepSeekMoE architecture?

a) Adding more knowledge to the model
b) Saving GPU memory
c) Decreasing the number of computations
d) Explicitly assigning expertise to parts of the model

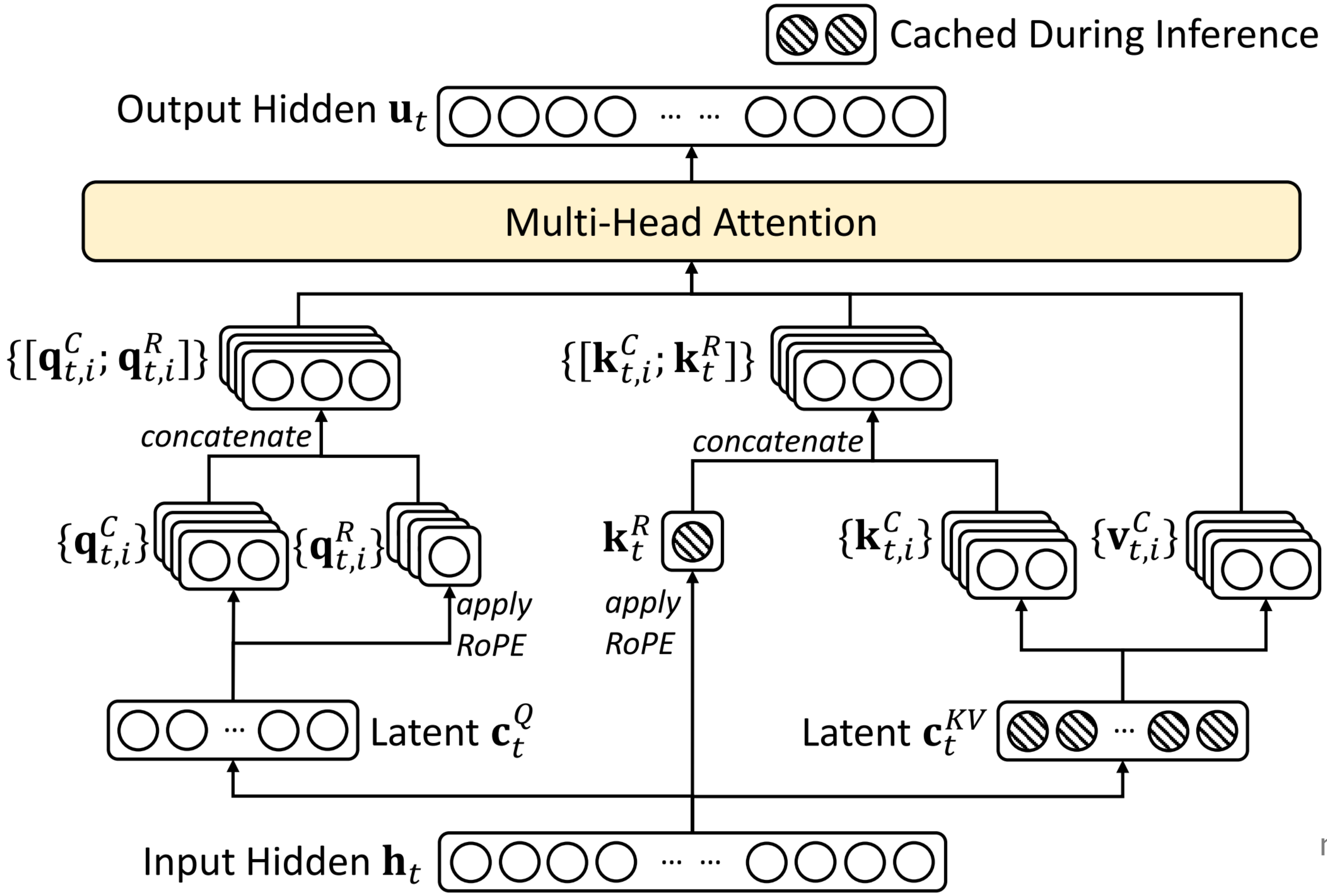
DeepSeek V2

Liu, Aixin, et al. "Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model." arXiv preprint arXiv:2405.04434 (2024).

Multi-Head Latent Attention

- In the original Transformer attention, the heavy **Key-Value cache** slows down the inference: $2(\#\text{heads})d(\#\text{blocks})$ values per token
- Solution: **low-rank key-value joint compression**:
  - Before applying the linear transformation into the **K**eys and **V**alues for each head, the attention input (h_t) is transformed into a low-dimensional compressed space: $c_t^{KV} = W^{DKV} h_t \in \mathbb{R}^{d_c}$
  - On inference, only the c_t^{KV} is **cached**
  - Similarly, the **Q**ueries are computed from a compressed vector
- Positional embedding (RoPE[1]) are computed before the compression

[1]Su, Jianlin, et al. "Roformer: Enhanced transformer with rotary position embedding." Neurocomputing 568 (2024): 127063.



Source: Liu, Aixin, et al. "Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model." arXiv preprint arXiv:2405.04434 (2024).

Quiz question: why aren't the **Q**ueries cached?

a) We don't need them in the future computations
b) They are easier to compute
c) They would take up too much memory
d) They change dynamically and cannot be cached

Quiz question: why aren't the **Q**ueries cached?

a) **We don't need them in the future computations**
b) They are easier to compute
c) They would take up too much memory
d) They change dynamically and cannot be cached

Device-Limited Routing

- Individual experts are often loaded on different GPUs (**expert parallelism**)
- Communication between the GPUs is costly
- Solution: **limiting the number of GPUs per token to $\boldsymbol{M}$**
- Implementation:
  - Select the top M devices based on the affinity and all the experts on those devices
  - Use the top $(K - K_s)$ experts from the selection.
- For $M \geq 3$ results are **comparable to unrestricted** selection

DeepSeek V3

Liu, Aixin, et al. "Deepseek-v3 technical report." arXiv preprint arXiv:2412.19437 (2024).

Chapter 2: Architecture

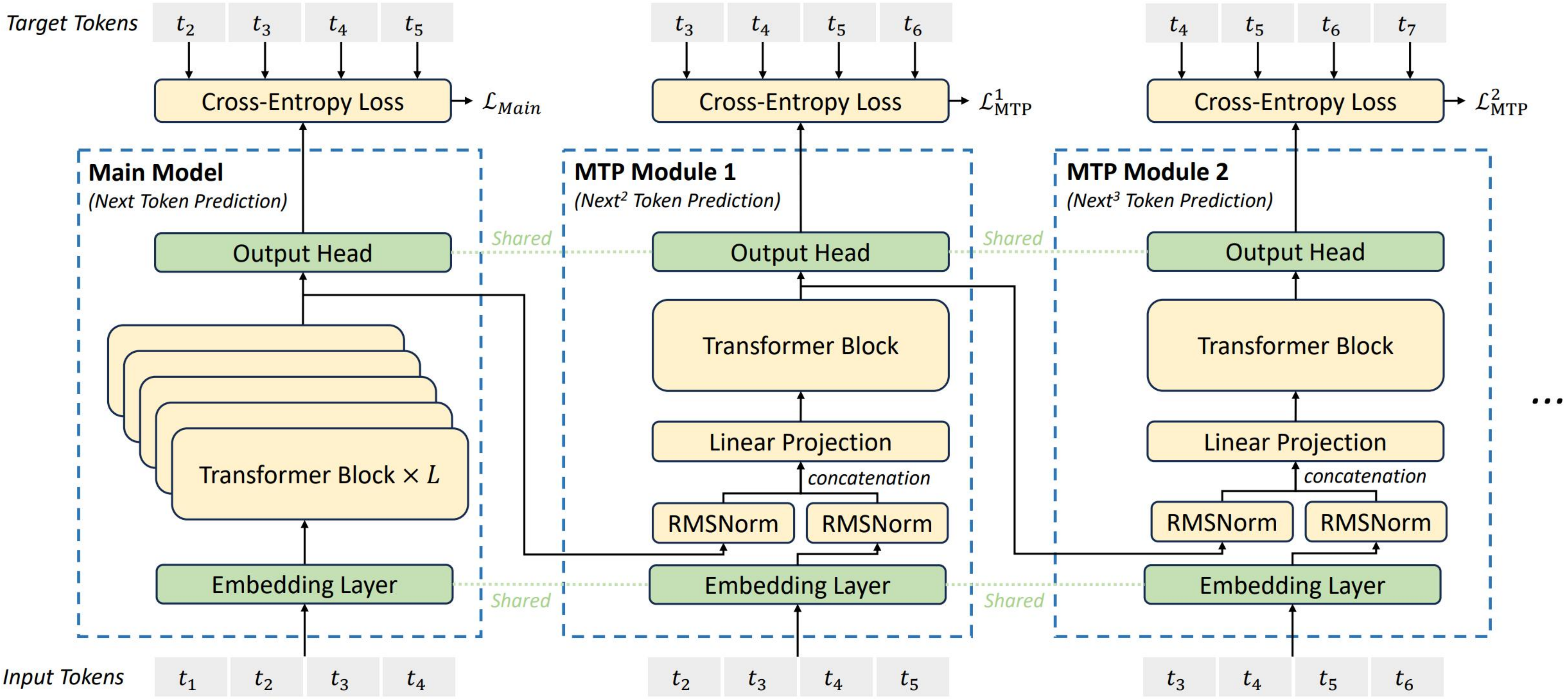
Architecture

- **Multi-head latent attention** (as described in DeepSeek V2)
- **Mixture of 256 experts** (as described in DeepSeekMoE and V2)
- Two approaches to the experts load balancing:
  1. **Auxiliary-loss-free** load balancing:
     - When computing the top $(K - K_s)$ experts for a token t, a bias b_i is added to the affinity $s_{i,t}$
     - The bias is dynamically in-/decreased by a hyperparameter γ during the training to account for under-/overloaded experts
  2. Complementary expert-level (auxiliary) balance loss (as described in DeepSeekMoE), with a **small learning rate** to preserve the performance[1]

[1]Wang, Lean, et al. "Auxiliary-loss-free load balancing strategy for mixture-of-experts." arXiv preprint arXiv:2408.15664 (2024).

Additional Training Objective: MTP

- **Multi-token prediction** (MTP) predicts $D + 1$ future tokens at each step (instead of one)
- Aim: **densify** the training process, enable the model to **pre-plan** for future token predictions
- MTP procedure:
  - Run the main model and obtain representations for each token
  - MTP modules are sequential: each taking the representations from the previous module concatenated with the true next token embeddings
  - Pass through a linear projection, a Transformers layer and the output head
- The embedding layer and the output head are shared
- MTP loss: $\mathcal{L}_{\mathrm{MTP}} = \frac{\lambda}{D} \sum_{k=1}^{D} \mathcal{L}_{\mathrm{MTP}}^{k}$ where $\mathcal{L}_{\mathrm{MTP}}^{k}$ is the cross-entropy loss



Source:Liu, Aixin, et al. "Deepseek-v3 technical report." arXiv preprint arXiv:2412.19437 (2024).

Chapter 3: Infrasctructures

Training Infrastructure

- Trained on a cluster of **2048 NVIDIA H800 GPUs** (80 GB VRAM), 8 GPUs per node
- Expert parallelism spanning 8 nodes
  - This introduces communication overhead similar to the computation time
  - Solution: DualPipe — overlapping communication and computation

Mixed-Precision Training

- Quantization to FP8 increases effectivity but is **limited by outliers**
- Most of the **core computation** (such as matric multiplication) is done in **FP8**
- **Original precision** is preserved in some modules
- Fine-grained quantization:
  - Standard practice: scale the **maximum absolute value** from the samples to the **maximum representable FP8** → one outlier ruins the accuracy
  - Fine-grained approach: split the sequence into **blocks of size N_C**; each block has its own scale based on its maximum absolute value

Inference: Pre-Filling (stage I)

- During pre-filling, the user's **prompt is processed** and cached items are pre-computed
- Minimum deployment unit: **4x8 GPUs**
- Parallelism strategies for the attention modules:
  - 4-way **Tensor Parallelism** (TP4) (= weights distributed over 4 devices)
  - **Sequence Parallelism** (SP) (= sequence is split for some operations)
  - 8-way **Data Parallelism** (DP8) (= 8 independent copies of the sub-model)
- Parallelism strategies for the MoE modules:
  - 32-way **Expert Parallelism** (EP32) (= experts distributed over 32 devices)
- 32 **redundant experts** are maintained, dynamically changed.

Quiz question: what is being cached for each input token?

☐ key projections for each head $k_{t,i}$

☐ queries projections each head $q_{t,i}$

☐ values projections each head $v_{t,i}$

☐ compressed latent vector c_t^{KV} for keys and values

☐ compressed latent vector c_t^{Q} for queries

Quiz question: what is being cached for each input token?

☐ key projections for each head $k_{t,i}$

☐ queries projections each head $q_{t,i}$

☐ values projections each head $v_{t,i}$

☑ **compressed latent vector c_t^{KV} for keys and values**

☐ compressed latent vector c_t^Q for queries

Inference: Decoding (stage II)

- During decoding, the model **predicts the tokens** autoregressively
- Minimum deployment unit: **40x8 = 320 GPUs**
- Attention parallelism: TP4 + SP + DP80
- Parallelism strategies for the MoE modules:
  - 320-way **Expert Parallelism** (EP320)
  - Each **GPU hosts one expert**
  - 64 GPUs host the shared and redundant experts

Chapter 4: Pre-Training

Training Data

- 14.8 T of multilingual diverse tokens, with a large portion of math and code
- Byte-Pair Encoding tokenization
- Data augmentation: FIM with the rate of 0.1
  - Text is split into three parts: $f_{prefix}, f_{middle}, f_{suffix}$ and transformed: `[BEGIN]` f_{prefix} `[HOLE]` f_{suffix} `[END]` f_{middle} `[EOS]`

Hyper-Parameters

- **Model parameters:**
- 61 Transformer layers
- 128 attention heads
- attention dim: 128
- KV compressed dim: 512
- Q compressed dim: 1536
- hidden dim: 7168
- first 3 FFNs kept dense
- 256 routed experts
- 8 of them activated per tok
- max 4 nodes per tok
- 1 shared expert
- expert hidden dim: 2048
- 1 extra MTP token

- **Training parameters**
- AdamW with $\beta_1 = 0.9, \beta_2 = 0.95, WD = 0.1$
- Context window: 4096 (later extended using YaRN[1])
- LR linear increase from 0 to 2.2×10^{-4}, constant for 10 T tokens, then decreased to 7.3×10^{-6}
- Gradient clipped to 1.0
- BS increased from 3072 to 15360 over 469 B tokens
- Auxiliary-free loss update $\gamma = 0.001$, then 0 for the last 500 B tokens
- Complementary balance lost weight: $\eta_1 = 0.0001$
- MTP loss weight $\lambda = 0.3$ for the first 10 T tokens, then $\lambda = 0.1$

[1]Peng, Bowen, et al. "Yarn: Efficient context window extension of large language models." arXiv preprint arXiv:2309.00071 (2023).

	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**

Source:Liu, Aixin, et al. "Deepseek-v3 technical report." arXiv preprint arXiv:2412.19437 (2024).

	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
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**

Source: Liu, Aixin, et al. "Deepseek-v3 technical report." arXiv preprint arXiv:2412.19437 (2024).

Chapter 5: Post-Training

Reasoning Data Generation

- Reasoning data partially based on a **DeepSeek-V2.5-based R1 prototype**
- **Problem:** R1 models **overthink** and generate very long sequences
- **Solution:** create an **expert model** for data generation using SFT and RL pipeline (different for coding, math, general reasoning…)
- SFT is done on **two kinds of samples**: $\langle \text{problem}, \text{original response} \rangle$ and $\langle \text{system prompt}, \text{problem}, \text{R1 response} \rangle$; the system prompt guides the model through the reasoning
- During the RL, **system prompt is removed**, and responses are sampled at a **high temperature**
- Final result: **concise** answers retaining R1 **thinking patterns**

Supervised Fine-Tuning (SFT)

- **Instruction-tuning datasets** including $1.5\mathrm{M}$ instances
- Largely generated:
  - Non-reasoning data responses **generated by DeepSeek-V2.5** and verified by human annotators
  - Reasoning data generated by an **expert model** (see last slide)
- Hyperparameters:
  - 2 epochs
  - cosine LR decay from 5×10^{-6} to 1×10^{-6}

Quiz question: what is reinforcement learning?

Reinforcement Learning (RL)

- Two types of **reward models** (RM):
  - **Rule-based:** for questions that can be objectively validates (e. g. correct $\boxed{\text{solution}}$)
  - **Model-based:** for questions with a free-form ground-truth answers, a dedicated ML model is trained — based on DeepSeek-V3 SFT
- Group relative policy optimization (**GRPO**) strategy:
  - **Omits the critic** (value) model
  - **Group scores** used instead
  - For a question q, outputs $\{o_1, o_2, \cdots, o_G\}$ are sampled from the old policy model $\pi_{\theta_{old}}$ and π_θ optimized by maximizing the objective…

Group Relative Policy Optimization (GRPO)

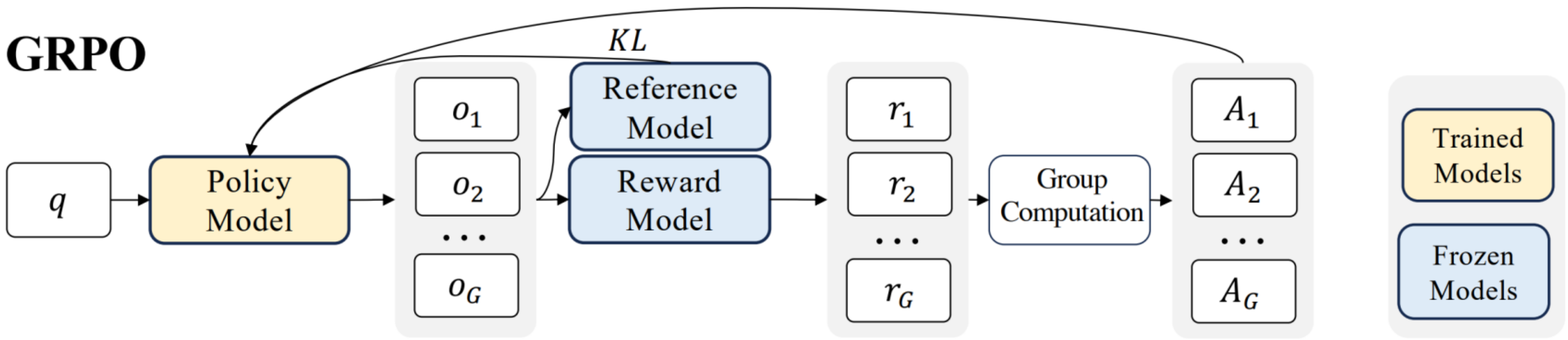
- Maximizing the objective:

$$\mathcal{J}_{GRPO} = \mathbb{E}\left[q \sim P(Q), \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(O|q)\right]$$

$$\frac{1}{G}\sum_{i=1}^{G}\left(\min\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}A_i, \text{clip}\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}, 1-\varepsilon, 1+\varepsilon\right)A_i\right) - \beta \mathrm{D}_{KL}\left(\pi_\theta || \pi_{ref}\right)\right)$$

where advantage $A_i = \frac{r_i - mean(\{r_1, r_2, \cdots, r_G\})}{std(\{r_1, r_2, \cdots, r_G\})}$

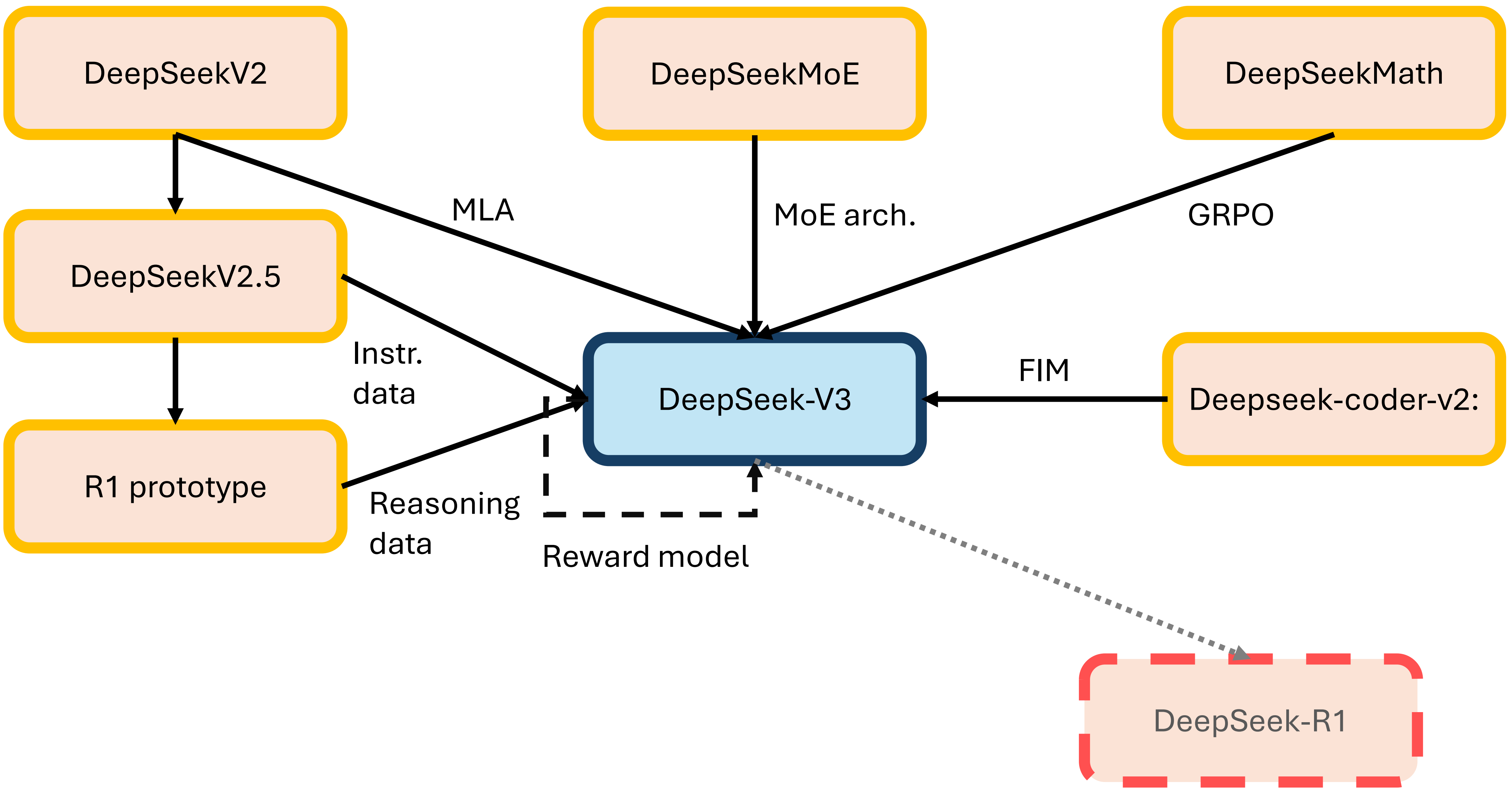
Compare this to the advantage from the REINFORCE with baseline[1] algorithm:
$a_\pi(s,a) = q_\pi(s,a) - v_\pi(s)$



[1]Sutton, Richard S., and Andrew G. Barto. Reinforcement learning: An introduction. Vol. 1. No. 1. Cambridge: MIT press, 1998.
Figure source: Shao, Zhihong, et al. "Deepseekmath: Pushing the limits of mathematical reasoning in open language models." arXiv preprint arXiv:2402.03300 (2024).

	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**

Source: Liu, Aixin, et al. "Deepseek-v3 technical report." arXiv preprint arXiv:2412.19437 (2024).


DeepSeekV2
DeepSeekMoE
DeepSeekMath
MLA
MoE arch.
GRPO
DeepSeekV2.5
Instr.
data
FIM
DeepSeek-V3
Deepseek-coder-v2:
R1 prototype
Reasoning
data
Reward model
DeepSeek-R1

Chapter 6: Conclusion, Limitations, and Future Directions

Conclusion

- DeepSeek-V3, a large MoE model with 671B parameters, **37B activated parameters**
- **MLA**, **DeepSeekMoe** architecture, **auxiliary-loss-free** strategy, **multi-token** prediction training objective, **FP8** training
- Distilled reasoning from the **R1** prototype
- **Strongest open-source model** at the time, comparable results to GPT-4o and Claude-3.5-Sonnet
- **2.788M** H800 **GPU hours** for full training (=57 days with 2048 GPUs)

Limitations

- Large deployment unit recommended (inaccessible to smaller teams)
- Generation speed is still limited (more advanced hardware anticipated)

Future Directions

- Consistently adhere to the **open-source** philosophy and longtermism
- Aiming toward the artificial general intelligence (**AGI**)
- Study and refine the architectures, possibly **beyond Transformer**
- Improve the quality and quantity of the **training data**, explore other sources of training signals
- Explore the **deep-thinking capabilities**
- Explore a more **comprehensive** way of **evaluation**, instead of optimizing for a fixed set of benchmarks


Thank you for your attention
TECHNOLOGY
SERVICES
BASIC MATERIALS
HEALTHCARE
FINANCIAL
CONSUMER GOODS
INDUSTRIAL GOODS
UTILITIES
GOOGL -10.27%
FB
MSFT -11.84%
AMZN -8.04%
BABA -8.63%
WMT -7.80%
INTC -12.71%
BRK-B -10.92%
AAPL -12.46%
JPM -18.30%
BAC -17.59%
V
MA