A Review of DeepSeek Models' Key Innovative Techniques

Chengen Wang ✉
University of Texas at Dallas
chengen.wang@utdallas.edu

Murat Kantarcioglu
Virginia Tech
muratk@vt.edu

Abstract

DeepSeek-V3 and DeepSeek-R1 are leading open-source Large Language Models (LLMs) for general-purpose tasks and reasoning, achieving performance comparable to state-of-the-art closed-source models from companies like OpenAI and Anthropic—while requiring only a fraction of their training costs. Understanding the key innovative techniques behind DeepSeek's success is crucial for advancing LLM research. In this paper, we review the core techniques driving the remarkable effectiveness and efficiency of these models, including refinements to the transformer architecture, innovations such as Multi-Head Latent Attention and Mixture of Experts, Multi-Token Prediction, the co-design of algorithms, frameworks, and hardware, the Group Relative Policy Optimization algorithm, post-training with pure reinforcement learning and iterative training alternating between supervised fine-tuning and reinforcement learning. Additionally, we identify several open questions and highlight potential research opportunities in this rapidly advancing field.

Keywords: DeepSeek, Multi-Head Latent Attention, Mixture of Experts, Group Relative Policy Optimization (GRPO)

1 Introduction

The emergence of ChatGPT in late 2022 [Ope25a] ushered in a new era of Large Language Model (LLM) research. LLMs have since advanced rapidly, with models like GPT [Ope25b] and Claude [Ant25] demonstrating exceptional performance. While open-source LLMs such as LLaMA [GDJ+24] have achieved competitive results in certain metrics, their overall performance still lags behind proprietary models.

In January 2025, DeepSeek rattled markets and made headlines [Reu25] with DeepSeek-V3 [LFX+24] and newly-launched DeepSeek-R1 models [GYZ+25]. These models achieve performance comparable to that of state-of-the-art GPT models, while requiring only a fraction of training resources. Understanding the techniques underlying the remarkable effectiveness and efficiency of these models is crucial for advancing LLM research.

In this paper, we review the key techniques behind DeepSeek Models' success. These include the refinement to the transformer architecture—specifically, Multi-Head Latent Attention (MLA) and Mixture of Experts (MoE); Multi-Token Prediction; the co-design of algorithms, frameworks and hardware; the Group Relative Policy Optimization (GRPO) reinforcement learning algorithm; and post-training techniques, such as pure reinforcement learning and multi-stage iterative training that alternates between Supervised Fine-Tuning (SFT) and reinforcement learning.

Additionally, we identify several issues that are not addressed in DeepSeek's technical report or ablation studies, highlighting potential research opportunities.

In the following, we first provide a concise yet in-depth review of the above-mentioned innovative techniques in Section 2, followed by a discussion of the open problems and potential research directions in Section 3, and conclude the paper in Section 4.

2 The Innovative Techniques

In this section, we examine the key innovative techniques that drive the success of the DeepSeek models. While these techniques are integrated into DeepSeek-V3 and DeepSeek-R1, some may have been introduced in earlier DeepSeek models.

2.1 Multi-Head Latent Attention

KV Cache Issue

KV cache is a technique used in the Multi-Head Attention (MHA) block of a transformer to accelerate inference by storing intermediate keys and values, eliminating the need for repeated calculations. However, the KV cache can become a bottleneck for long-context LLMs due to their high memory consumption. One approach to reducing the KV cache is to employ fewer attention heads, as seen in Multi-Query Attention (MQA) [Sha19] and Group-Query Attention (GQA) [ALTDJ+23]. Despite this, their performance does not match that of MHA. Later, an innovative attention mechanism called Multi-head Latent Attention (MLA) is proposed for DeepSeek-V2 [LFW+24], which requires far less KV cache while achieving better performance.

(i) Less #heads → MQA, GQA

(2) MLA

2.1.1 Standard Multi-Head Attention

In the standard MHA [VSP+17], the queries, keys and values are obtained through the projection matrices $W^Q, W^K, W^V \in \mathbb{R}^{d_hn_h\times d}$, transforming $\mathbf{h_t} \in \mathbb{R}^d$, the input of t-th token, to queries, keys and values $\mathbf{q}_t = W^Q\mathbf{h}_t, \mathbf{k}_t = W^K\mathbf{h}_t, \mathbf{v}_t = W^V\mathbf{h}_t, \mathbf{q}_t, \mathbf{k}_t, \mathbf{v}_t \in \mathbb{R}^{d_hn_h}$, respectively, where d is the dimension of the input embedding, n_h is the number of heads and d_h is the dimension per head.

The dimension $d_h \times n_h$ indicates how the $\mathbf{q}_t, \mathbf{k}_t, \mathbf{v}_t$ are sliced into n_h heads with dimension d_h per head for the multi-head attention mechanism [LFW+24, Eq. (4)-(8)]:

$$[\mathbf{q}_{t,1};\mathbf{q}_{t,2};...;\mathbf{q}_{t,n_h}] = \mathbf{q}_t, \tag{1}$$

$$[\mathbf{k}_{t,1};\mathbf{k}_{t,2};...;\mathbf{k}_{t,n_h}] = \mathbf{k}_t, \tag{2}$$

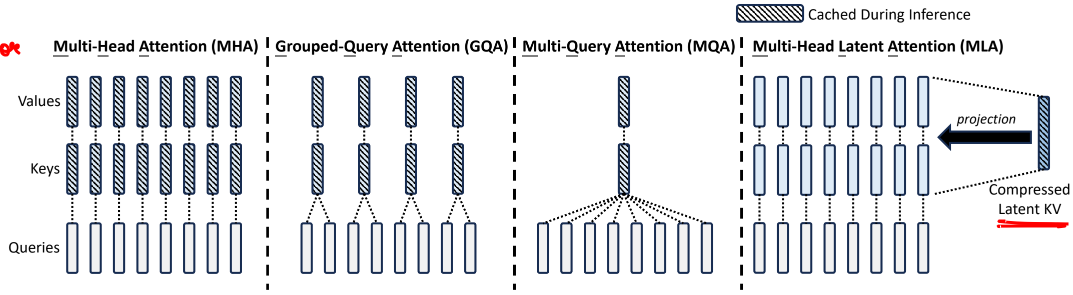
$$[\mathbf{v}_{t,1};\mathbf{v}_{t,2};...;\mathbf{v}_{t,n_h}] = \mathbf{v}_t, \tag{3}$$

$$\mathbf{o}_{t,i} = \sum_{j=1}^{t} \mathrm{Softmax}_j\left(\frac{\mathbf{q}_{t,i}^T\mathbf{k}_{j,i}}{\sqrt{d_h}}\right)\mathbf{v}_{j,i}, \tag{4}$$

$$\mathbf{u}_t = W^O[\mathbf{o}_{t,1};\mathbf{o}_{t,2};...;\mathbf{o}_{t,n_h}], \tag{5}$$

where $\mathbf{q}_{t,i}, \mathbf{k}_{t,i}, \mathbf{v}_{t,i} \in \mathbb{R}^{d_h}$ represent the query, key, and value of the i-th head, respectively, and $W^O \in \mathbb{R}^{d\times d_hn_h}$ is the output projection matrix. During inference, each token requires KV cache of size $2n_hd_hl$, where l is the number of layers.

2 for K,V
additional factor
2 possible for
half precision



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

Figure 1: A simplified illustration of Multi-Head Attention (MHA), Grouped-Query Attention (GQA), Multi-Query Attention (MQA), and Multi-head Latent Attention (MLA). Adapted from [LFW+24, Figure 3].

2.1.2 Low-Rank Key-Value Joint Compression

The core idea of MLA is to decompose the projection matrix into two lower-rank matrices: $W = W^UW^{DKV}$, where $W^{DKV} \in \mathbb{R}^{d_c\times d}$ is the down-projection matrix for both keys and values, $W^U \in \mathbb{R}^{d_hn_h\times d_c}$ is the up-projection matrix, and $d_c \ll d_hn_h$. The down-projection matrix compresses *both* keys and values into *one* latent vector $\mathbf{c}_t^{KV} = W^{DKV}\mathbf{h}_t, \mathbf{c}_t^{KV} \in \mathbb{R}^{d_c}$ [LFW+24, Eq. (9)]. Since $d_c \ll d_hn_h$, for each token, saving $\mathbf{c}_t^{KV}$, of size d_cl instead of both $\mathbf{k}_t$ and $\mathbf{v}_t$, of size $2d_hn_hl$, greatly reduces the KV cache.

$C_t^{KV} = W^{DKV} \quad h_t \to C_t^{KV}$

$(n \times d_c) \quad (n \times d)$

↳ Same for both K & V

The keys and values are computed from the latent vector $\mathbf{c}_t^{KV}$ as follows [LFW+24, Eq. (10)-(11)]:

$$\boxed{\begin{aligned} \mathbf{k}_t^C &= W^{UK}\mathbf{c}_t^{KV}, \\ \mathbf{v}_t^C &= W^{UV}\mathbf{c}_t^{KV}, \end{aligned}} \qquad \begin{aligned} &(6) \\ &(7) \end{aligned}$$

where $W^{UK}, W^{UV} \in \mathbb{R}^{d_h n_h \times d_c}$ denote the up-projection matrices for keys and values, respectively. Importantly, W^{UK} will be absorbed into W^Q and W^{UV} absorbed into W^O during inference, so we do not need compute $\mathbf{k}_t^C, \mathbf{v}_t^C$ explicitly. The architecture of MLA is illustrated in Figure 1.

Moreover, *low-rank* compression of queries is applied to reduce the activation memory during training [LFW+24, Eq. (12)-(13)]:

$$\boxed{\begin{aligned} \mathbf{c}_t^Q &= W^{DQ}\mathbf{h}_t, \\ \mathbf{q}_t^C &= W^{UQ}\mathbf{c}_t^Q, \end{aligned}} \qquad \begin{aligned} &(8) \\ &(9) \end{aligned}$$

where $\mathbf{c}_t^Q \in \mathbb{R}^{d'_c}$ represents the compressed latent vectors for queries, with $d'_c \ll d_h n_h$, and $W^{DQ} \in \mathbb{R}^{d'_c \times d}, W^{UQ} \in \mathbb{R}^{d_h n_h \times d'_c}$ denote the down-projection and up-projection matrices, respectively.

2.1.3 Decoupled Rotary Position Embedding

DeepSeek-V2 utilizes the Rotary Position Embedding (RoPE) [SAL+24]:

$$\mathbf{q}_i^T\mathbf{k}_j = \mathbf{h}_i^T(W^Q)^T \,\mathrm{RoPE}_{\Theta,j-i}(W^K\mathbf{h}_j) \tag{10}$$

$$= \mathbf{h}_i^T(W^Q)^T \,\mathrm{RoPE}_{\Theta,j-i}(W^{UK}W^{DKV}\mathbf{h}_j) \tag{11}$$

where $\mathrm{RoPE}_{\Theta,\text{j-i}}(\cdot)$ denotes the operation that applies the RoPE matrix, Θ is pre-defined parameters, and i, j are the i-th and j-th positions. As a result, W^{UK} will not be absorbed into W^Q, leading to significant computational cost during inference.

To address this issue, DeepSeek-V2 proposes to decouple RoPE into a separate set of queries and keys: multi-head queries $\mathbf{q}_{t,i}^R \in \mathbb{R}^{d_h^R}$ and a key $\mathbf{k}_t^R \in \mathbb{R}^{d_h^R}$ *shared* by all heads, where d_h^R represents the per-head dimension of the decoupled queries and keys. This decoupling strategy essentially computes two separate sets of attention weights, which are then added together. The full MLA computation is as follows [LFW+24, Eq. (14)-(19)]:

$$[\mathbf{q}_{t,1}^R; \mathbf{q}_{t,2}^R; ...; \mathbf{q}_{t,n_h}^R] = \mathbf{q}_t^R = \mathrm{RoPE}(W^{QR}\mathbf{c}_t^Q), \tag{12}$$

$$\mathbf{k}_t^R = \mathrm{RoPE}(W^{KR}\mathbf{h}_t), \tag{13}$$

→ Cached

$$\mathbf{q}_{t,i} = [\mathbf{q}_{t,i}^C; \mathbf{q}_{t,i}^R], \tag{14}$$

$$\mathbf{k}_{t,i} = [\mathbf{k}_{t,i}^C; \mathbf{k}_t^R], \tag{15}$$

$$\mathbf{o}_{t,i} = \sum_{j=1}^{t} \mathrm{Softmax}_j\left(\frac{\mathbf{q}_{t,i}^T\mathbf{k}_{j,i}}{\sqrt{d_h + d_h^R}}\right)\mathbf{v}_{j,i}^C, \tag{16}$$

$$\mathbf{u}_t = W^O[\mathbf{o}_{t,1}; \mathbf{o}_{t,2}; ...; \mathbf{o}_{t,n_h}], \tag{17}$$

Shared by all heads

where $W^{QR} \in \mathbb{R}^{d_h^R n_h \times d'_c}$ and $W^{KR} \in \mathbb{R}^{d_h^R \times d}$ denote matrices used to generate the decoupled queries and key, respectively, $\mathrm{RoPE}(\cdot)$ refers to the operation that applies the RoPE matrix, with the subscripts omitted, and $[\cdot;\cdot]$ represents the concatenation operation. During inference, the decoupled key $\mathbf{k}_t^R$ with dimension d_h^R is also cached. As a result, each token requires cache of size $(d_c + d_h^R)l$ in total. For DeepSeek-V2, where $d_c = 4d_h$ and $d_h^R = \frac{d_h}{2}$, the KV cache per token is $\frac{9}{2}d_h l$.

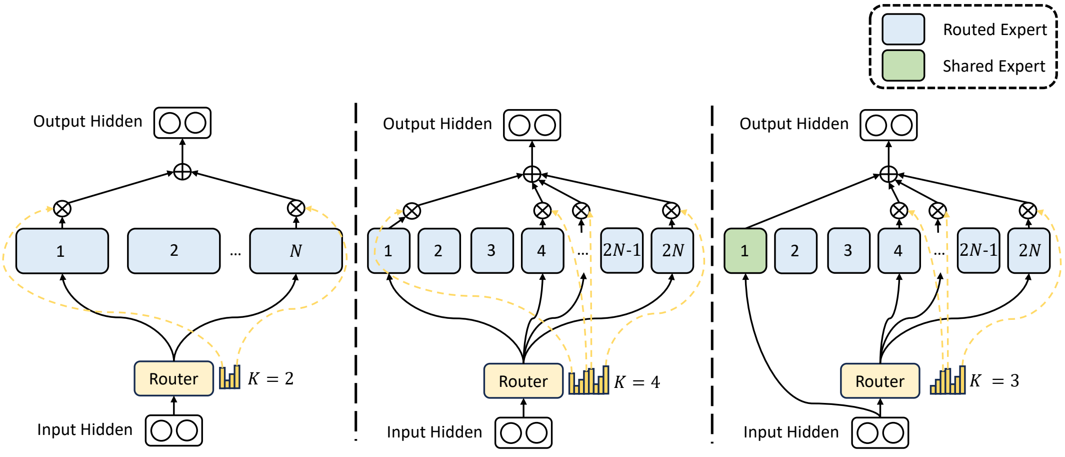
$4d_h$ $\quad \frac{d_h}{2}$

$(d_c + d_h^R)l$

↳ Per token KV cache

It has been reported that MLA outperforms MHA [LFW+24, Table 9], which is surprising considering that MLA uses low-rank matrices, inherently containing less information than the original projection matrices for keys and values. Therefore, this performance gain is likely due to the introduction of the decoupled RoPE, which differs from the original RoPE. However, no ablation study for the decoupled RoPE has been reported, making it a worthwhile direction for further investigation.

$\left(\frac{9}{2}d_h l\right)$



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

Figure 2: An illustration of DeepSeekMoE. Note the number of expert parameters and computational cost remain the same across the three architectures. Adapted from [DDZ+24, Figure 2].

2.2 Mixture of Experts

Mixture of Experts (MoE) is an architecture designed to reduce computational cost while scaling up model parameters. In an MoE model, the Feed-Forward Network (FFN) layers in a Transformer are typically replaced with MoE layers at specified intervals. Each MoE layer consists of multiple experts, all structurally identical to a standard FFN. Tokens are routed to one or two experts [FZS22, LLX+20]. The DeepSeekMoE architecture [DDZ+24] introduces two key innovations: fine-grained expert segmentation and shared expert isolation. These innovations are built upon the conventional MoE.

2.2.1 Fine-Grained Expert Segmentation

On top of the conventional MoE architecture shown in Figure 2(a), each FFN is segmented into m smaller experts by dividing the FFN hidden dimension evenly. As a result, if the total number of experts is N and the number of activated experts for each token is K in a conventional MoE, then the total number of experts is increased to mN and the number of activated experts is increased to mK for a fine-grained MoE architecture, as illustrated in Figure 2(b). This fine-grained segmentation strategy greatly improves the combinatorial flexibility of the activated experts.

Split FFN into smaller experts (m)
Use mK experts at a time
(if previously K)

2.2.2 Shared Expert Isolation

Shared experts are dedicated to capture the common knowledge across diverse contexts, reducing parameter redundancy among different experts. Specifically, K_s experts are reserved as shared experts, and each token will be always assigned to these shared experts in additional to their respective routed experts. To maintain a constant computational cost, the *total* number of *routed* experts N_r is reduced to $mN - K_s$ and the number of routed experts for each token is $mK - K_s$.

With the novel strategy of fine-grained expert segmentation and shared expert isolation, an MoE layer in the DeepSeekMoE architecture is defined as follows [DDZ+24, Eq. (9)-(11)]:

$$\mathbf{h}_t^l = \sum_{i=1}^{K_s} \mathrm{FFN}_i\left(\mathbf{u}_t^l\right) + \sum_{i=K_s+1}^{mN} \left(g_{i,t}\,\mathrm{FFN}_i\left(\mathbf{u}_t^l\right)\right) + \mathbf{u}_t^l, \longrightarrow \underline{\text{Residual}} \tag{18}$$

$$g_{i,t} = \begin{cases} s_{i,t}, & s_{i,t} \in \mathrm{Topk}(\{s_{j,t} | K_s + 1 \leq j \leq mN\}, mK - K_s), \\ 0, & \text{otherwise}, \end{cases} \tag{19}$$

$$s_{i,t} = \mathrm{Softmax}_i\left({\mathbf{u}_t^l}^T \mathbf{e}_i^l\right), \tag{20}$$

where $\mathrm{FFN}_i(\cdot)$ refers to the i-th expert FFN, $\mathbf{u}_t^l \in \mathbb{R}^d$ is the hidden state of the t-th token after the l-th attention module, and $\mathbf{h}_t^l \in \mathbb{R}^d$ is the output hidden state of the t-th token after the l-th MoE layer. $g_{i,t}$ represents the gate value for the i-th expert, $s_{i,t}$ is the token-to-expert affinity, $\mathrm{Topk}(\cdot, K)$ gives the set of top K affinity scores calculated for the t-th token across all N experts, and $\mathbf{e}_i^l$ represents the centroid of the i-th expert in the l-th layer.

2.2.3 Load Balancing

Load Balancing

① Tokens route only to a few experts

② Other experts remain undertraind

③ Inter device comm when Expert Parallel

The automatically learned routing strategy may face the issue of load imbalance, where either a few experts are always selected while others are not sufficiently trained, or the activated experts are distributed across multiple devices, leading to significant inter-device communication cost. These issues are address by an auxiliary loss for load balancing [FZS22]. The expert-level balance loss is formulated as follows [DDZ+24, Eq. (12)-(14)]:

$$\boxed{\begin{aligned}
\mathcal{L}_{\text{ExpBal}} &= \alpha \sum_{i=1}^{N'} f_i P_i, && (21)\\
f_i &= \frac{N'}{K'T} \sum_{t=1}^{T} \mathbb{1}(\text{Token } t \text{ selects Expert } i), && (22)\\
P_i &= \frac{1}{T} \sum_{t=1}^{T} s_{i,t}, && (23)
\end{aligned}}$$

where α is a hyper-parameter, $N' = mN - K_s$ and $K' = mK - K_s$ for simplicity, and $\mathbb{1}(\cdot)$ represents the indicator function. When the load is uniformly-distributed among the experts, $\mathcal{L}_{\text{ExpBal}}$ is minimized, $f_i = 1$, $P_i = \frac{K'}{N'}$, and we have $\sum_{i=1}^{N'} f_i P_i = N' \cdot 1 \cdot \frac{K'}{N'} = K'$.

Let f_i' and P_i' be the normalized version of f_i and P_i, respectively, $f_i' = \frac{f_i}{N'}, P_i' = \frac{P_i}{K'}$, such that both form probability distributions. The limitation of this expert-level loss formulation is that when P_i' is uniformly-distributed, i.e., $P_i' = \frac{1}{N'}$ for all i, then for *any* distribution of f_i', we have $\sum_{i=1}^{N'} f_i \cdot P_i = \sum_{i=1}^{N'} f_i' \cdot K' = K'$. Under these circumstances, the auxiliary loss fails to push toward balanced experts utilization. If a uniform distribution of P_i' inherently produce a uniform distribution of f_i' in practice, then incorporating f_i in the loss function appears redundant. Given the widespread adoption of this formulation [FZS22, LFX+24, JZYY24], it is worthwhile to investigate its theoretical justification and explore potential improvements.

Besides expert-level load balancing, device-level and communication load balancing [DDZ+24, LFW+24] are proposed to ensure balanced computation and communication across different devices. The formulations of these loss functions follow a similar pattern.

Since the auxiliary loss may degrade model performance, an auxiliary-loss-free load balancing strategy is proposed to strike a better trade-off between load balance and model performance [WGZ+24]. Specifically, a bias term b_i for each expert i is added to the affinity score $s_{i,t}$ to determine the top-K selection [LFX+24, Eq. (16)]:

$$g_{i,t}' = \begin{cases} s_{i,t}, & s_{i,t} + b_i \in \mathrm{Topk}(\{s_{j,t} + b_j | 1 \le j \le N_r\}, K_r), \\ 0, & \text{otherwise}, \end{cases} \tag{24}$$

where N_r represents the number of routed experts and K_r is the number of activated routed experts. During training, the bias term b_i will be decreased by γ if the expert is overloaded and increased by γ if the expert is underloaded, where γ is a hyperparameter. Note that the bias term is used solely for top-K selection and the gating value is still using the original affinity score $s_{i,t}$, as shown in Eq. (24). In this equation, $s_{i,t} + b_i$ serves as the input to the $\mathrm{Topk}(\cdot)$ function, while $s_{i,t}$ is the value of $g_{i,t}'$ if $s_{i,t}$ is among the top-K. In DeepSeek-V3, a complementary sequence-wise auxiliary loss is also employed to avoid extreme imbalance within any single sequence [LFX+24].

2.3 Multi-Token Prediction

DeepSeek-V3 employs Multi-Token Prediction (MTP) [GIR+24] to improve training performance. For each token, instead of predicting next-token only, MTP predicts D additional tokens in a causal chain,

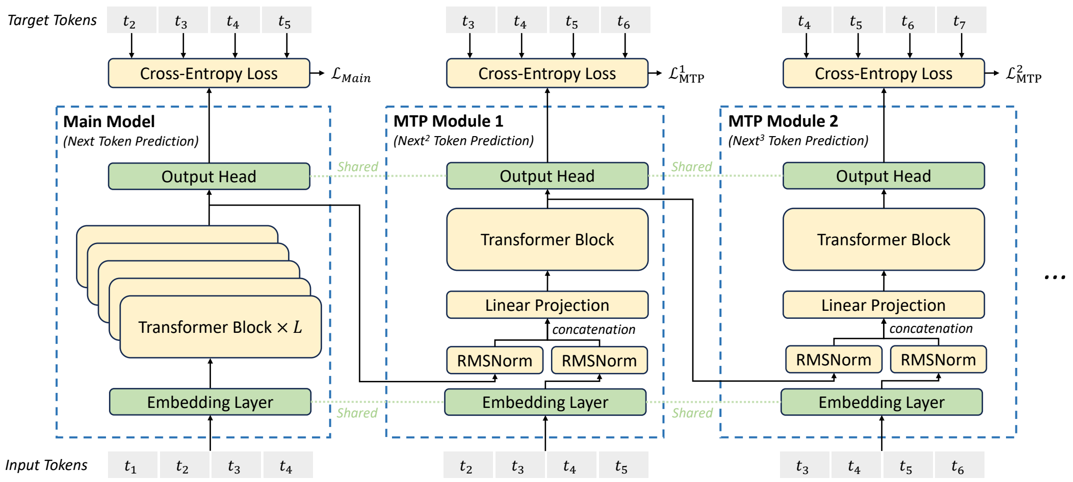


Figure 3: An illustration of the Multi-Token Prediction (MTP) implementation in DeepSeek-V3, which keeps the complete causal chain for predicting each token at every depth. Adapted from [LFX+24, Figure 3].

MTP
Embedding & Output layer are shared

as shown in Figure 3. At each depth k of the D MTP modules, there are a shared Embedding Layer and a shared Output Head, an independent Transformer Block, and an independent Linear Projection layer. The input to the Linear Projection layer is an concatenation of the embedding at the current depth and the output embedding from the previous depth.

The MTP training objective, $\mathcal{L}_{\text{MTP}}$, is the average of the cross-entropy loss $\mathcal{L}^k_{\text{MTP}}$ at each depth $k \in \{1, 2, \cdots, D\}$ [LFX+24, Eq. (24)-(25)]:

$$\mathcal{L}^k_{\text{MTP}} = \text{CrossEntropy}(P^k_{2+k:T+1}, t_{2+k:T+1}) = -\frac{1}{T}\sum_{i=2+k}^{T+1} \log P^k_i[t_i], \tag{25}$$

$$\mathcal{L}_{\text{MTP}} = \frac{\lambda}{D}\sum_{k=1}^{D} \mathcal{L}^k_{\text{MTP}}, \tag{26}$$

where T represents the length of the input sequence, t_i is the ground-truth token at the i-th position, and $P^k_i[t_i]$ denotes the prediction probability of t_i at depth k.

The advantage of MTP lies in its higher sample efficiency during training [GIR+24], leading to improved performance. However, the causal chain formed by the MTP modules introduces additional *training* time overhead beyond conventional next-token prediction, a factor not addressed in the ablation study for MTP in DeepSeek-V3 [LFX+24, Sec. 4.5.1].

2.4 Co-design of Algorithms, Frameworks and Hardware

Through the co-design of algorithms, frameworks and hardware, with meticulous engineering optimizations, DeepSeek-V3 significantly enhances the training efficiency and completes the pre-training of the model on 14.8 trillion tokens with 2.788 million H800 GPU hours [LFX+24].

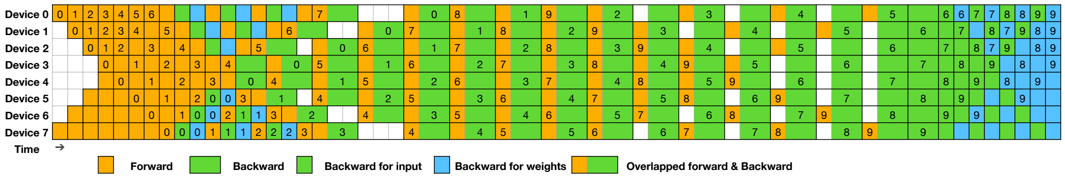


Figure 4: An example of DualPipe scheduling for 8 PP ranks and 20 micro-batches in two directions. The micro-batches in the reverse direction are symmetric to those in the forward direction. Adapted from [LFX+24, Figure 5].

2.4.1 DualPipe

To reduce the communication overhead introduced by cross-node expert parallelism, an innovative pipeline parallelism algorithm called DualPipe [LFX+24] is proposed to overlap the computation and communication within a pair of individual forward and backward chunks. The algorithm divides each chunk into four components, with the backward computation chunk further divided into two parts for input and weights, respectively [QWHL23], to reduce pipeline bubbles. A specific ratio of GPU SMs are dedicated to communication, ensuring that communication remains fully hidden during execution, effectively achieving near-zero all-to-all communication overhead. The DualPipe algorithm employs a bidirectional pipeline scheduling, feeding data from both ends of the pipeline, as illustrated in Figure 4.

DualPipe requires keeping two copies of the models parameters, leading to additional memory consumption. It turns out the *bidirectional* part is unnecessary and can be removed with a "cut-in-half" procedure, as outlined in [QWHL25].

2.4.2 FP8 Mixed Precision Training

A mixed precision framework for training DeepSeek-V3 is introduced for efficient training without accuracy degradation. In order to accelerate training, the *majority* of core computation kernels—General Matrix Multiplication (GEMM)—are implemented in FP8 precision [DLBZ22, PWW+23, FCBS24]. Despite the efficiency advantage of FP8 format, DeepSeek-V3 maintains the original precision for certain operators due to their sensitivity to low-precision computations, to balance training efficiency and numerical stability. These operators include the embedding module, the output head, MoE gating modules, normalization operators, and attention operators.

Embedding layer
Output head
MoE gating
Normalization ops
Attention ops
↓
Original Precision

This framework utilizes a fine-grained quantization strategy to extend the *dynamic range* of the FP8 format: tile-wise grouping with $1 \times N_c$ elements or block-wise grouping with $N_c \times N_c$ elements, where N_c is the channel size, with $N_c = 128$ in DeepSeek-V3 model.

The accuracy of low-precision GEMM operations largely depends on high-precision *accumulation*. DeepSeek-V3 employs a strategy of promotion to CUDA Cores for higher precision, periodically copying intermediate results to FP32 registers on CUDA Cores at an interval N_c for full-precision FP32 accumulation.

FP8 GEMM ←
but FP32 accumulation

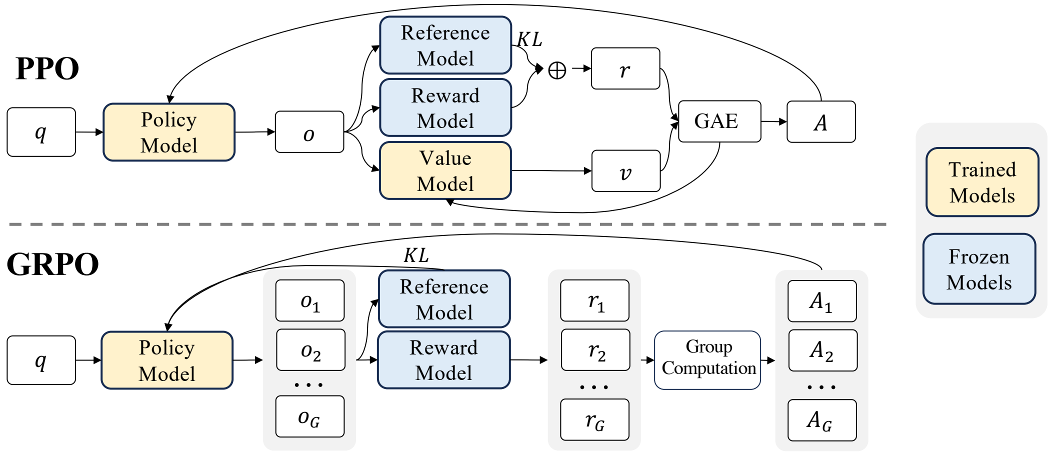


Figure 5: A comparison of PPO and GRPO. GRPO estimates the baseline from group scores, without a value model. Adapted from [SWZ+24, Figure 4].

2.5 Group Relative Policy Optimization

Group Relative Policy Optimization (GRPO) [SWZ+24] is an efficient and effective variant of Proximal Policy Optimization (PPO) [SWD+17]. GRPO eliminates the value function approximation in PPO by directly estimating the advantage, significantly reducing memory usage. In the context of LLM, where typically only the last token is assigned a reward, the training of a value function in PPO is challenging. The simplified GRPO can achieve comparable performance while being more efficient.

PPO

More specifically, the PPO maximizes the following objective [SWZ+24, Eq. (1)]:

$$\mathcal{J}_{PPO}(\theta) = \mathbb{E}[q \sim P(Q), o \sim \pi_{\theta_{old}}(O|q)] \frac{1}{|o|} \sum_{t=1}^{|o|} \min \left[\frac{\pi_\theta(o_t|q, o_{<t})}{\pi_{\theta_{old}}(o_t|q, o_{<t})} A_t, \text{clip} \left(\frac{\pi_\theta(o_t|q, o_{<t})}{\pi_{\theta_{old}}(o_t|q, o_{<t})}, 1 - \epsilon, 1 + \epsilon \right) A_t \right], \tag{27}$$

where q, o represent questions and outputs, π_θ and $\pi_{\theta_{old}}$ are the current and old policy models, respectively, and ϵ is a hyper-parameter related to clipping, and A_t represents the advantage, which is estimated using rewards and a learned value function.

Instead of training a value function, GRPO directly estimates the advantage by sampling a group of outputs $\{o_1, o_2, \cdots, o_G\}$ from the old policy $\pi_{\theta_{old}}$, yielding G rewards $\mathbf{r} = \{r_1, r_2, \cdots, r_G\}$, scored by a reward model. There are two ways to estimate the advantage: outcome supervision and process supervision. Output supervision provides a reward at the end of each output o_i, and set the advantages $\hat{A}_{i,t}$ of all tokens to the same normalized reward, i.e., $\hat{A}_{i,t} = \widetilde{r}_i = \frac{r_i - \text{mean}(\mathbf{r})}{\text{std}(\mathbf{r})}$ [SWZ+24, Sec. 4.1.2]. Process supervision provides a reward for each intermediate steps, and then calculate the advantage for each token by summing the normalized rewards obtained from the subsequent steps.

GRPO maximizes the following objective [SWZ+24, Eq. (3)]:

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^G \sim \pi_{\theta_{old}}(O|q)]$$
$$\frac{1}{G} \sum_{i=1}^{G} \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \left\{ \min \left[\frac{\pi_\theta(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q, o_{i,<t})} \hat{A}_{i,t}, \text{clip} \left(\frac{\pi_\theta(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q, o_{i,<t})}, 1 - \epsilon, 1 + \epsilon \right) \hat{A}_{i,t} \right] - \beta \mathbb{D}_{KL} \left[\pi_\theta || \pi_{ref} \right] \right\}, \tag{28}$$

where π_{ref} is the reference policy model, typically the initial base model or the SFT model. The GRPO algorithm is illustrated in Figure 5.

2.6 Post-Training: Reinforcement Learning on the Base Model

2.6.1 Pure Reinforcement Learning

DeepSeek-R1-Zero [GYZ+25] is trained on the base model DeepSeek-V3-Base with pure reinforcement learning (RL) without any supervised fine-tuning (SFT) data. The performance of DeepSeek-R1-Zero continuously improves throughout the RL training process. The reasoning behaviors, including reflection and the exploration of alternative approaches, naturally arise during the training, demonstrating the effectiveness of pure RL and the model's capability to learn and generalize solely through RL.

DeepSeek-R1-Zero utilizes the GRPO algorithm, as outlined in Section 2.5. The reward function includes two types of rewards: *accuracy rewards*, which assess the correctness of the model's response, and *format rewards*, which enforces the model to enclose its thinking process within the tags "`<think>`" and "`</think>`". A training template is designed to guide the model in following a specified format, first generating a reasoning process before presenting the final answer.

Although DeepSeek-R1-Zero achieves remarkable performance through pure RL, it encounters challenges such as poor readability and language mixing. To mitigate these issues and further enhance the model, DeepSeek-R1 [GYZ+25] is introduced, which employs an iterative training approach that alternates between SFT and RL.

2.6.2 Reinforcement Learning with Cold Start

DeepSeek-R1 employs a training pipeline consisting of four stages [GYZ+25]:

- **Cold Start** To mitigate the instability of the early cold start phase of RL training, thousands of long Chain-of-Thought (CoT) [WWS+22] examples are collected to fine-tune the DeepSeek-V3-Base model, which then serves as the foundation for subsequent reinforcement learning.

- **Reasoning-oriented RL** After fine-tuning DeepSeek-V3-Base on the cold-start data, the model undergoes the same RL training process as employed in DeepSeek-R1-Zero. To address language mixing, an additional language consistency reward is introduced, measured as the proportion of target words in the CoT.

- **Rejection Sampling and SFT** The goal of this phase is to improve the model's performance in writing, role-playing and other general-purpose tasks. Once Reasoning-oriented RL converges, 600k reasoning-related training samples are collected via rejection sampling from the checkpoint,

Generative Reward

For some there is CoT
for some simple queries
no CoT

retaining only the correct responses. Additionally, approximately 200k non-reasoning training samples are collected, either from parts of the SFT dataset of DeepSeek-V3 or generated by DeepSeek-V3.

- **RL Alignment** This phase aims to better align the model with human preferences, by improving its helpfulness and harmlessness while also refining its reasoning capabilities. Helpfulness is measure based on the utility and relevance of the response, while harmlessness is evaluated throughout the entire response to reduce potential risks, biases or harmful content.

3 Discussions

In this section, we identify several areas where DeepSeek has made innovations and highlight potential future directions.

- **Transformer Architecture Improvement** The transformer serves as the core building block of LLMs. While MLA improves the attention mechanism, MoE enhances the FFN block within the transformer. Together, these innovations have significantly contributed to the advancement of the DeepSeek-V3 model. Advancements in transformer architecture can greatly influence both the effectiveness and efficiency of the training process. For example, a comprehensive ablation study of the decoupled rotary position embedding, as discussed in Section 2.1.3, could provide deeper insights. Additionally, further theoretical justification for the load balancing objective, as outlined in Section 2.2.3, would be valuable for future research.

- **High Sample Efficiency** The introduction of multi-token prediction enhances the utilization of training data, thereby improving sample efficiency [GIR+24]. This demonstrates that better training efficiency can be achieved by developing algorithms that make more effective use of the training dataset. In Section 2.3, we mention the issue of the incurred longer training time, suggesting there is still room for further improvement.

- **Co-design of algorithms, frameworks and hardware** The DualPipe and FP8 mixed precision training are engineering techniques introduced to enhance training efficiency. These innovations emphasize the value of designing models from a holistic perspective, integrating architecture, algorithms and hardware in the most effective and efficient manner. Recently, an improvement to the DualPipe has been made by [QWHL25], as mentioned in Section 2.4.1.

- **Reinforcement Learning** The impressive performance of pure reinforcement learning in the post-training stage highlights a new research avenue in this area. The iterative training approach, which alternates between SFT and RL, is particularly inspiring. Furthermore, the introduction of GRPO demonstrates how a widely used RL algorithm can be improved to significantly reduce GPU memory usage.

4 Conclusion

In this paper, we have reviewed the key innovative techniques that contributed to the success of DeepSeek models. These include innovations in the transformer architecture, techniques for improving sample efficiency, the co-design of algorithms, frameworks and hardware, as well as the GRPO reinforcement learning algorithm and the application of reinforcement learning in the post-training stage. Our review highlights several open questions and potential research directions in this rapidly advancing field.

References

[ALTDJ+23] Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*, 2023. 2

[Ant25] Anthropic. All models overview. https://docs.anthropic.com/en/docs/about-claude/models/all-models, 2025. [Online; accessed 2025-3]. 1

[DDZ+24] Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y Wu, et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*, 2024. 4, 5

[DLBZ22] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in neural information processing systems*, 35:30318–30332, 2022. 7

[FCBS24] Maxim Fishman, Brian Chmiel, Ron Banner, and Daniel Soudry. Scaling fp8 training to trillion-token llms. *arXiv preprint arXiv:2409.12517*, 2024. 7

[FZS22] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022. 4, 5

[GDJ+24] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. 1

[GIR+24] Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large language models via multi-token prediction. *arXiv preprint arXiv:2404.19737*, 2024. 5, 6, 9

[GYZ+25] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. 1, 8

[JZYY24] Peng Jin, Bo Zhu, Li Yuan, and Shuicheng Yan. Moe++: Accelerating mixture-of-experts methods with zero-computation experts. *arXiv preprint arXiv:2410.07348*, 2024. 5

[LFW+24] Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024. 2, 3, 5

[LFX+24] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024. 1, 5, 6, 7

[LLX+20] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020. 4

[Ope25a] OpenAI. Introducing chatgpt. https://openai.com/index/chatgpt/, 2025. [Online; accessed 2025-3]. 1

[Ope25b] OpenAI. Models. https://platform.openai.com/docs/models, 2025. [Online; accessed 2025-3]. 1

[PWW+23] Houwen Peng, Kan Wu, Yixuan Wei, Guoshuai Zhao, Yuxiang Yang, Ze Liu, Yifan Xiong, Ziyue Yang, Bolin Ni, Jingcheng Hu, et al. Fp8-lm: Training fp8 large language models. *arXiv preprint arXiv:2310.18313*, 2023. 7

[QWHL23] Penghui Qi, Xinyi Wan, Guangxing Huang, and Min Lin. Zero bubble pipeline parallelism. *arXiv preprint arXiv:2401.10241*, 2023. 7

[QWHL25] Penghui Qi, Xinyi Wan, Guangxing Huang, and Min Lin. Dualpipe could be better without the dual. https://hackmd.io/@ufotalent/r1lVXsa9Jg, 2025. Blog. 7, 9

[Reu25] Reuters. Nasdaq drops 3% as china's deepseek ai model hits tech shares. https://www.reuters.com/markets/us/nasdaq-futures-tumble-chinas-ai-push-rattles-big-tech-2025-01-27/, 2025. [Online; accessed 2025-3]. 1

[SAL$^+$24] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. 3

[Sha19] Noam Shazeer. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019. 2

[SWD$^+$17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 7

[SWZ$^+$24] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. 7, 8

[VSP$^+$17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 2

[WGZ$^+$24] Lean Wang, Huazuo Gao, Chenggang Zhao, Xu Sun, and Damai Dai. Auxiliary-loss-free load balancing strategy for mixture-of-experts. *arXiv preprint arXiv:2408.15664*, 2024. 5

[WWS$^+$22] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022. 8