EPJ Web of Conferences **380**, 02002 (2026) https://doi.org/10.1051/epjconf/202638002002
ICISCT'25

DeepSeek-V3: Architecture and Optimizations -A Practical Review

Yassine Zouhdi [1] and *Boutaina* Hdioud [2]

[1]Information Retrieval and Data Analytics Team (IRDA) ENSIAS, Mohammed V University Rabat, Morocco

[2] Information Retrieval and Data Analytics Team (IRDA) ENSIAS, Mohammed V University Rabat, Morocco

Abstract. The design of transformer-based Large Language Models (LLMs) is being radically changed through new architectures that are able to overcome scalability limitations of previous designs, including Mixture-of-Experts (MoE), Multi-Head Latent Attention (MLA), and Multi-Token Prediction (MTP). As an open-weighted model released at the end of 2024, which has both state of the art architectural transparency and production scale efficiency, DeepSeek-V3 represents the ultimate testing ground for investigating these modern technologies. This paper provides a comprehensive analysis of the architectural structure of DeepSeek-V3 based upon information from the DeepSeek-V3 Technical Report, industry benchmarking data and independent latency testing, to demonstrate how various techniques can be used to optimize training while still providing competitive performance in code generation and mathematical reasoning. In addition, latency testing conducted on a Distilled version of DeepSeek-V3, with approximately 14 billion parameters, running on a T4 GPU, reveals that although significant improvements have been made in optimizing latency there remains substantial barriers to deploying these models. Through this context, this research will serve as a reference document for practitioners and researchers who wish to understand current trends and challenges in increasing accessibility to high performance AI models.

1 Introduction

In their groundbreaking paper *Scaling Laws for Neural Language Models* [1], **Kaplan** et al. demonstrated how larger models can achieve better results than smaller ones, but because traditional large model designs using dense layers have reached physical limits on compute and memory resources, researchers in the AI community now are applying new architectural inventions – including Mixture-of-Experts (MoE) Multi-Head Latent Attention (MLA) and Multi-Token Prediction (MTP) -- which are becoming the standards for all next-generation

Large Language Models such as OpenAI's GPT-4, xAI's Grok and Meta's Llama 4 Family. Consequently, DeepSeek-V3 is one of the first models to be transparent use-case for integration of multiple architectural innovations in conjunction with new optimization techniques for training. By analyzing the architectural foundations of DeepSeek-V3 [2], this article uses its design as a concrete vehicle to understand the training and inference optimizations that are currently reshaping the state of the art.

2 Related work

With the advent of Vaswani's Transformers (2017) [3], a breakthrough was achieved in machine learning, as the first architecture to incorporate a scalable parallelizable multi head attention mechanism. This innovation has since led to a second wave of Large Language Models (LLMs). This trend toward LLMs gained momentum with OpenAI's GPT-1 (2018) [4]. GPT-2 (2019) [5] demonstrated coherent text generations using a very limited prompt, and the democratization of LLMs further accelerated with the release of GPT-3 (2020) [6], a 175 billion parameter model whose impressive performance provided evidence of scaling laws: bigger is better. [1].

In addition to providing impressive results, however, this race for scale also exposed some serious limitations. The cost of training GPT-3 is estimated to be $4.6 million. [7] GPT-4 (2023) [8] costs are reported to be in the hundreds of millions. At the same time, closed source models such as GPT-4, Claude 3.5 Sonnet and Gemini Ultra began to dominate the market but remained out of reach to most researchers due to their closed source nature.

To help mitigate these issues optimized architectures have been developed. In 2021, Su et al. [9] developed RoPe, a method to utilize rotation-based transformations for encoding positions into tokens thereby allowing for longer context handling. Also in 2023, Google DeepMind introduced multi-token prediction [10], with this technique, LLMs can produce in parallel up to four tokens per step to accelerate inference. Additionally, DeepSeek AI, contributed to optimizing transformer architectures with its work on Mixture-of-Experts (MOE) [11] in 2024 to enable the better utilization of the models' parameters. That same year DeepSeek AI also released DeepSeek-V2 with Multi-Head Latent Attention [12], an optimized version of the original attention mechanism. By using all previous optimizations of transformers architectures DeepSeek-V3 [2] truly revolutionized the AI industry through employing multiple techniques in conjunction with several others and establishing new standards for cost-effectiveness when working with large language models.

3 Architectural innovations

In this chapter we will show architectural innovations that modified the classic transformer. Those are rotary positional embeddings (RoPE) [9], multi-head latent attention (MLA) [12] in place of traditional multi-head attention, mixture of experts (MOE) [11] with a brand-new load balancing strategy as opposed to a classical MLP feed-forward network, and a multi-token prediction instead of the typical token-by-token generation of autoregressive models.Technologies which were all implemented on the DeepSeek-V3 model.

3.1 Multi head latent attention (MLA)

The primary objective of MLA is to reduce the size of both query and key-value caches in the attention mechanism. This memory optimization strategy allows the model to efficiently process longer sequences of tokens.

3.1.1 Generation of keys and values

The hidden state is first projected into a lower dimensional latent space:

$$\boldsymbol{c_t^{KV}} = W^{DKV}\boldsymbol{h_t} \quad (1)$$

where $W^{DKV} \in R^{d_c \times d}$ is the down-projection matrix which enables compression of the keys and values, creating the latent vector $c_t^{KV} \in R^{d_c}$ (where $d_c \ll d \times n_h$) that encodes the semantic components of the keys and values in a compressed format. The de-compressed representations of the keys and values for each attention head are obtained as follows:

$$\mathbf{k_t^C} = \begin{bmatrix} \mathbf{k_{t,1}^C} \\ \mathbf{k_{t,2}^C} \\ \vdots \mathbf{k_{t,n_h}^C} \end{bmatrix} = W^{UK}\mathbf{c_t^{KV}} \quad (2)$$

$$\mathbf{v_t^C} = \begin{bmatrix} \mathbf{v_{t,1}^C} \\ \mathbf{v_{t,2}^C} \\ \vdots \mathbf{v_{t,n_h}^C} \end{bmatrix} = W^{UV}\mathbf{c_t^{KV}} \quad (3)$$

Here, W^{UK} and W^{UV} represent the up-projection matrices for the keys and values that project the compressed components to their respective spaces while retaining most of the important information. In the subsequent stage, MLA decouples the positional information from the semantic content. The positional component for the keys is computed as:

$$\mathbf{k_t^R} = \mathrm{RoPE}(W^{KR}\mathbf{h_t}) \quad (4)$$

Where W^{KR} represents a projection matrix, and RoPE(•) applies rotary position embedding, injecting position-dependent rotations. Rotary Positional Embedding (RoPE) is a method for integrating positional information into transformer models. Unlike the classical approach of simply adding a positional vector to each token embedding, RoPE rotates embeddings by an angle dependent on the token's position. Therefore, this method encodes token order in a smooth, continuous way and enables the model to better capture relative positions and deal with longer sequences.

At last, for each attention head i, we get the final key $k_{t,i}$ by concatenating its semantic and positional components:

$$\mathbf{k_{t,i}} = \left[\mathbf{k_{t,i}^C}; \mathbf{k_t^R}\right] \quad (5)$$

3.1.2 Generation of Queries.

The queries are also derived from the same hidden state $\mathbf{h_t}$ in a similar decomposition:

$$\mathbf{q_t} = [\mathbf{q_t^C}; \mathbf{q_t^R}] = \left[W^{UQ}\mathbf{c_t^Q}, \mathrm{RoPE}(W^{QR}\mathbf{h_t})\right] \quad (6)$$

Where $\mathbf{c_t^Q} = W^{DQ}\mathbf{h_t}$ is a latent representation for queries, and W^{UQ} and W^{QR} represent the corresponding projection matrices.

3.1.3Attention Computation.

Once queries $\mathbf{q_{t,i}}$, keys and values $\mathbf{v_{s,i}}$, have been obtained for each head i, the attention is calculated as usual:

$$\text{score}_{t\to s}^{(i)} = \frac{\mathbf{q}_{t,i}^{\top}\mathbf{k}_{s,i}}{\sqrt{d_{\text{head}} + d_R}} \quad (7)$$

$$\alpha_{t\to s}^{(i)} = \text{softmax}_s\left[\text{score}_{t\to s}^{(i)}\right] \quad (8)$$

$$\mathbf{z_{t,i}} = \sum_s \alpha_{t\to s}^{(i)}\mathbf{v_{s,i}^C} \quad (9)$$

Where d_{head} represents the dimensionality of the semantic key/query vectors, and d_R is the dimensionality of the positional RoPE vectors.

The output vectors from all attention heads are then concatenated and projected back to the model dimension d using a final output projection matrix W^O:

$$\mathbf{z_t} = W^O\begin{bmatrix}\mathbf{z_{t,1}}\\ \mathbf{z_{t,2}}\\ \vdots\ \mathbf{z_{t,n_h}}\end{bmatrix} \quad (10)$$

3.2 Mixture of experts (MoE)

Like its predecessor DeepSeek-V2, DeepSeek-V3 uses a Mixture of Experts (MoE), instead of the classical MLP feedforward layer. In this MoE architecture, the classical monolithic layer is replaced with multiple neural networks called "experts", that work in parallel. The idea behind this is to use only the experts which are best suited to process each individual token. So fewer parameters will be active. In DeepSeek-V3, the overall effect is a reduction from a total of 671 billion to only 37 billion parameters activated per token.

3.2.1 Generation of keys and values

A fixed set of Feed-Forward Networks are used as "shared" experts. These networks are fed all tokens regardless of content. Let us denote the input token representation (or hidden state) of token t by $\mathbf{u_t}$. Then output of shared experts can be calculated as follows:

$$\sum_{i=1}^{N_s} \text{FFN}_i^{(s)}(\mathbf{u_t}) \quad (11)$$

Where:

- N_s is the number of shared experts.
- $\text{FFN}_i^{(s)}(\cdot)$ represents the feed-forward transformation performed by the i-th shared expert.

Shared experts apply a consistent baseline transformation to every token without any gating mechanism.

3.2.2 Routed experts and the gating mechanism

Besides the shared experts, DeepSeek-V3 also utilizes a larger pool of routed experts. For each token, a gating mechanism dynamically selects the top K experts from this pool and calculates their respective weights, in doing so, it ensures only the most relevant experts process each token. This design helps minimize the number of active parameters. The output of the routed expert is as follows:

$$\sum_{i=1}^{N_r} g_{i,t}\mathrm{FFN}_i^{(r)}(u_t) \quad (12)$$

$$g_{i,t} = \frac{g'_{i,t}}{\sum_{j=1}^{N_r} g'_{j,t}} \quad (13)$$

$$g'_{i,t} = \begin{cases} s_{i,t}, & \text{if } s_{i,t} \in \mathrm{Topk}\left(\{s_{j,t} \mid 1 \le j \le N_r\}, K_r\right) \\ 0 \; otherwise \end{cases} \quad (14)$$

$$s_{i,t} = \mathrm{Sigmoid}(\mathbf{u_t}^{\top}\mathbf{e_i}) \quad (15)$$

Where:

- K_r is the number of routed experts chosen (via top-K selection).
- $\mathrm{FFN}_i^{(r)}$ denotes the transformation by the i-th routed expert.
- $g_{i,t}$ is the gating weight for the i-th routed expert, indicating its relative contribution for token t.
- $g'_{i,t}$ is a sparse mask retaining only the top-K_r scores.
- $s_{i,t}$ measures the relevance of the i-th expert to $\mathbf{u_t}$, where $\mathbf{e_i}$ represents the centroid vector of expert i.

3.2.3 Routed experts and the gating mechanism

The final output for each token is generated by adding together the input representations and the contribution of the shared-experts with the weighted contributions of the routed-experts:

$$\mathbf{h'_t} = \mathbf{u_t} + \sum_{i=1}^{N_s} \mathrm{FFN}_i^{(s)}(\mathbf{u_t}) + \sum_{i=1}^{N_r} g_{i,t}\mathrm{FFN}_i^{(r)}(\mathbf{u_t}) \quad (16)$$

Equation 16 ensures that each token has a consistent base-line via the shared-experts while also having unique transformations generated via the routed-experts.

3.2.4 Auxillary-Loss-Free Load Balancing

As stated in the DeepSeek-V3 Technical Report [2], Traditional MoE architectures can cause Routing Collapse due to an overuse of a few experts when using unbalanced expert workload [13]. The traditional fixes for this problem are based on auxiliary Losses [14] that enforce load balancing, however, these losses degrade model performance too much. In contrast, DeepSeek-V3 introduces an Auxiliary-Loss Free Load Balancing Mechanism [15] that balances expert usage without adding additional loss terms. A modified Gating formulation

achieves this functionality. Let i denote the raw score for the ith routed expert for token t and $\mathrm{b_i}$ denote a bias term that promotes balanced usage of experts. The free load balancing gating output $\mathrm{g'_{i,t}}$is defined as follows:

$$\mathrm{g'_{\{i,t\}}} = \begin{cases} \mathrm{s_{i,t}}, & \text{if } \mathrm{s_{i,t}} + \mathrm{b_i} \in \mathrm{T_K} \\ 0 \end{cases} \quad (17)$$

Where $T_K = \mathrm{TopK}\big(\{s_{j,t} + b_j \mid 1 \le j \le N_r\}, K_r\big)$ and :

- $\mathrm{N_r}$ denotes the total number of routed experts.
- $\mathrm{K_r}$ denotes the number of routed experts to be selected per token.

Gating values defined with this new method are only used for routing, the coefficients in the routed expert's equation (Equation 13) do not change. A dynamic adjustment to the bias term b_i occurs at the completion of each step. If an expert becomes overloaded, then the corresponding bias will be reduced by a fixed amount γ. On the other hand, if an expert becomes underloaded, the value of b_i is increased by γ.

Note that DeepSeek-V3 uses a complementary sequence-wise auxiliary loss [2], which discourages extreme imbalance.

3.3 Multi-token prediction (MTP)

DeepSeek-V3 includes Multi Token Predictor (MTP) to predict all of the subsequent tokens at once. Unlike standard autoregressive models, which predict one token per time step, the MTP modules of DeepSeek-V3 use sequential MTP for predicting additional tokens beyond the immediate next token.

3.3.1 MTP implementation

Each token $\mathrm{t_I}$ in an input sequence (length T) has its own set of MTP modules defined as follows:

- A shared embedding layer Emb(·). This is the same as used in the main model.
- A linear projection matrix M_k. It combines the previous depth representation with the embedding of the $i + k$-th token.
- A Transformer block $\mathrm{TRM}_k(\cdot)$ specific to the k-th prediction depth.
- A shared Output Head $\mathrm{OutHead}(\cdot)$. The same as the output head of the main model.

Let $\boldsymbol{h_i^{(k-1)}} \in R^d$ be the representation of token t_i at the $(k-1)$-th depth, and let $\mathrm{Emb}(\boldsymbol{t_{i+k}}) \in R^d$ be the embedding of the $i + k$-th token. We first combine these two vectors via:

$$\boldsymbol{h'^{(k)}} = M_k \left[\mathrm{RMSNorm}\big(\boldsymbol{h_i^{(k-1)}}\big) \middle| \mathrm{RMSNorm}\big(\mathrm{Emb}(\boldsymbol{t_{i+k}})\big)\right] \quad (18)$$

Here $(.|.)$ denotes vector concatenation, and M_k is a linear projection matrix for the k-th MTP module. RMSNorm(·) is root mean square normalization. When $\mathrm{k} = 1$, $h^{(k-1)}$ is the representation given by the final layer of the main model.

The combined vector $\boldsymbol{h'^{(k)}}$ then serves as an input to the Transformer block $\mathrm{TRM}_k$the k-th depth:

$$\boldsymbol{h}_{\mathbf{1}:\boldsymbol{T}-\boldsymbol{k}}^{(\boldsymbol{k})} = \mathrm{TRM}_k(\boldsymbol{h}_{\mathbf{1}:\boldsymbol{T}-\boldsymbol{k}}^{\prime(\boldsymbol{k})}) \quad (19)$$

Here T denotes the input sequence length. As a notation i : j represents slicing from index i up to but not including j. Thus, this equation gives us the output representation $\boldsymbol{h}_{\boldsymbol{i}}^{(\boldsymbol{k})}$ of the current depth.

Finally, for an input i, the output head OutHead(·) computes a probability distribution for the (k)-th additional token:

$$\boldsymbol{P}_{\boldsymbol{i}+\boldsymbol{k}}^{(\boldsymbol{k})} = \mathrm{OutHead}\left(\mathbf{h}^{(\mathbf{k})}\right) \quad (20)$$

Where $\mathbf{P}_{\mathbf{i}+\mathbf{k}}^{(\mathbf{k})} \in R^V$ and V represents vocabulary size. Internally, OutHead(·) applies a linear map to produce logits followed by a softmax to generate prediction probabilities.

3.3.2 MTP training and inference

For each prediction depth k, we compute a cross entropy loss $L_{\mathrm{MTP}}^{(k)}$.

$$L_{MTP}^{(k)} = CrossEntropy\left(\boldsymbol{P}_{\boldsymbol{2}+\boldsymbol{k}:\boldsymbol{T}+\boldsymbol{1}}^{(\boldsymbol{k})}, \boldsymbol{t}_{\boldsymbol{2}+\boldsymbol{k}:\boldsymbol{T}+\boldsymbol{1}}\right) = -\frac{1}{T}\sum_{i=2+k}^{T+1} \log \boldsymbol{P}_{\boldsymbol{i}}^{(\boldsymbol{k})}[\boldsymbol{t}_{\boldsymbol{i}}] \quad (21)$$

Where $\mathrm{t_i}$ is the ground truth token at position i and $\boldsymbol{P}_{\boldsymbol{i}}^{(\boldsymbol{k})}[\boldsymbol{t}_{\boldsymbol{i}}]$ is the predicted probability of token $\mathrm{t_i}$ We then average the MTP losses over all depths k = 1, … , D and multiply it with a weighting factor λ to form the overall MTP loss:

$$L_{\mathrm{MTP}} = \lambda \sum_{k=1}^{D} L_{\mathrm{MTP}}^{(k)} \quad (22)$$

The MTP modules can be discarded during inference allowing the main model to function normally for single-token prediction or to utilize them for speculative decoding which will speed up generation by predicting multiple tokens in parallel.

4 Optimizations and training strategies

DeepSeek-V3 applies several state-of-the-art methods to efficiently train large models while minimizing communication overhead and memory footprint. We discuss four main components of our training strategy here:

4.1 Parallelism Strategies

DeepSeek-V3 uses three different types of parallelisms to address computational loads as follows:

Pipeline Parallelism (PP)- 16-way DeepSeek-V3, which has 61 layers, is segmented horizontally into 16 separate stages. Each of these segments are allocated a group of GPUs and the data are divided into small batches. By dividing the deep learning model into 16 horizontal stages, each GPU can perform work on an uninterrupted block of layers. This reduces idle GPU time by allowing each GPU to do its own processing while others are working on their portion of the network.

Expert Parallelism (EP)-64 way Within the MoE architecture previously discussed, there are 256 experts. However, only eight of them will be activated per token. There are 64 GPUs (four GPUs/node) distributed over eight nodes. Intelligent routing is applied to ensure each

token is sent to a maximum of four nodes. Therefore, we reduce the number of network communications between nodes. Additionally, custom all-to-all kernels along with FP8 quantization provide faster communication throughput between experts.

Data Parallelism (ZeRO-1) Data parallelism is utilized via ZeRO Stage 1. With ZeRO Stage 1, only the optimizer states are split (AdamW), as opposed to splitting all model parameters. This allows us to increase the batch size (from 3072 tokens/micro-batch to 15360 tokens/micro-batch). While we save memory, we also keep GPU synchronization efficient since it avoids the high amount of communication necessary to shard all parameters.

4.2 DualPipe algorithm

The Dual Pipe Algorithm is another innovative method developed to mask the delay associated with communication latency by performing intelligent overlaps of computation with communication. Each data chunk is divided into four distinct processes:

- **Attention Computation:** The attention operations performed on the activations.
- **All-to-all Dispatch (Communication):** Sending the activations or gradients from one GPU to all other relevant GPUs.
- **MLP Computation:** Processing the activation through a multi-layer perceptron for nonlinear transformation.
- **All-to-all Combine (Communication):** Aggregating the results exchanged between GPUs.

These steps are ordered such that the communication phases (all-to-all dispatch and combine) occur concurrently with the attention and MLP computations. A bidirectional pipelined scheduling strategy is used to allow micro-batches to be processed at the same time in both forward and backward directions. As a result, the two phases minimize GPU idle times and maximize resource utilization.

4.3 Communication optimizations

Communication among the GPUs is an important factor for achieving scale with large models like DeepSeek-V3. It utilizes a hybrid topology for communication among the GPUs:

- **Between GPUs connected by the same physical machine;** These use NVLink for communication, which has approximately 160 GB/s bandwidth.
- **Among GPUs physically located in different machines;** They utilize InfiniBand for communication, and it typically has a lower bandwidth than NVlink at approximately 50 GB/s.

As mentioned before, in order to minimize the amount of data transferred over the Infiniband network, each token will be forwarded to no more than four physical machines. All-to-all kernels customized for deep learning networks utilizing warp specialization dynamically allocate SM resources to communication tasks. Since this hybrid design includes both InfiniBand and NVLink as separate systems for transferring tokens, tokens are initially transferred using InfiniBand for transfer between physical machines, then they are transferred using NVLink for faster transfer within a physical machine. Based on the previous explanation, since only 20 SM can fully utilize the bandwidth of the available hardware, DeepSeek-V3 achieves very low overhead in terms of total communications.

4.4 Memory optimizations

DeepSeek-V3 contains multiple optimizations to achieve low memory utilization:

- **Recomputing:** Unlike traditional methods where the outputs of the layers are stored (i.e. layer outputs after RMSNorm layers or MLA projection layers), DeepSeek-V3 recomputes these values when backpropagation occurs. This reduces memory requirements due to fewer storage locations.
- **Storing EMA Parameters on CPU:** Although the exponential moving averages (EMAs) are used to help stabilize the training process, they have been moved into the cpu memory instead of gpu memory, thereby allowing DeepSeek-V3 to store less information in VRAM for other computations.
- **Shared Critical Parameters:** The critical parameters of the embeddings and the output heads of DeepSeek-V3 are shared between the original model and MTP module. By doing so, DeepSeek-V3 decreases redundancy and maintains consistent representations throughout the network, while decreasing its memory footprint.

5 Evaluating the performance of DeepSeek-V3

The results show that DeepSEEK-V3 is performing at or near the top among many others (including GPT-4o and Claude 3.5 Sonnet), demonstrating the superior ability of this model due to its advanced architectural structure and efficient training strategies, which are allowing it to achieve high levels of success in all areas assessed by the benchmarks.

5.1 General Benchmark Results

DeepSeek-V3 has achieved very strong results across a number of evaluation benchmarks.

Natural Language Understanding: Robust results were obtained from the MMLU-Pro and GPQA-Diamond benchmarks which demonstrate that DeepSeek-V3 is able to effectively comprehend language.

Code Generation: DeepSEEK-V3 consistently excelled over other models evaluated in these coding challenges (Codeforces and SWE-bench Verified) showing its applicability within Software Engineering.

Mathematical Reasoning: Excellent analytical reasoning was exhibited by DeepSEEK-V3 on math benchmark assessments; MATH500 and AIME 2024.

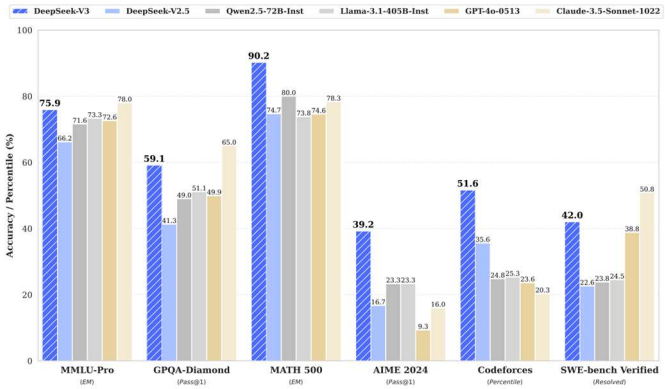

Fig. 1. Performance comparison of DeepSeek-V3 against other leading models. Adapted from [2], Fig. 1 .

DeepSeek-V3 shows itself to be an extremely versatile next-generation AI technology as evidenced by the fact that it is reaching excellent results across each of three main subject

areas. Through its demonstration of advanced reasoning in complex linguistic contexts through its natural language processing capabilities, along with its code generation capabilities being capable of providing highly efficient solutions to the problems encountered during software development, and through its presentation of accurate analytical problem solving techniques in math-related tasks, DeepSeek-V3 bridges the gap between innovative theory-based technologies and those applicable to real-world business needs. As well, because the architecture is scalable, it allows DeepSeek-V3 to be used in numerous different ways, ranging from automation to decision-making intelligence. For this reason, DeepSeek-V3 is positioned as a highly effective, interdisciplinary solution to today's many AI-related challenges.

5.2 Cost Effectiveness

DeepSeek-V3 stands out not only for matching—or even surpassing—state-of-the-art results, but also for offering substantial savings in both training and inference. Thanks to the various innovations it implements, such as MoE and its unique training strategy, DeepSeek-V3's overall training cost was only $5.576M, compared to the tens—even hundreds—of millions of dollars estimated for other closed source models.

Table 1, Offcial training cost of DeepSeek-V3 using NVIDIA H800 GPU.

Stage	GPU Hours	Cost (USD)
Pre-Training	2664K	$5.328M
Context Extension	119K	$0.238M
Post-Training	5K	$0.01M
Total	2788K	$5.576M

DeepSeek-V3 also offers one of the best intelligence-to-price ratios among non-reasoning models (i.e., those that do not implement chain-of-thought techniques), as demonstrated by the Artificial Analysis Intelligence Index.

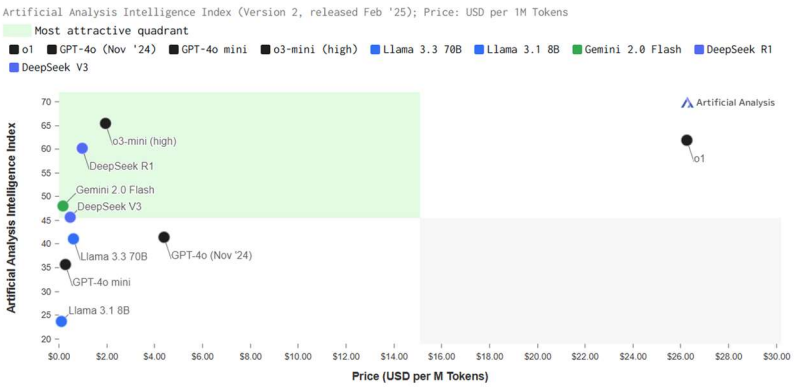

Fig. 2. artificialanalysis.ai Intelligence Index: Price vs. Intelligence Comparison (Feb 25).

As shown by the figure, DeepSeek-V3 surpasses its competitors, such as LLaMA 3.3 70B and GPT-4o (Nov 2024), in both performance and cost-effectiveness. DeepSeek-R1, the reasoning model built on DeepSeek-V3, clearly benefits from its architecture by outperforming its direct competitor, GPT o1, in the index. This was likely the reason behind OpenAI releasing o3-mini at a competitive price as a countermeasure. DeepSeek AI may have started a trend toward high-quality, low-cost models. Even though these models have a lower price per token, both R1 and o3-mini tend to generate more tokens through chain-of-thought processes, which deserves a separate discussion.

5.3 Testing a distilled model on Consumer Hardware

We tested the distilled version of DeepSeek-V3 using a minimally-provisioned T4 GPU in Google Colab. Although there are no official pre-trained versions of DeepSeek-V3 available as a "distilled" model, we used the arcee-ai/Virtuoso-Small-v2 model, which is based on the same architecture as our models and is available on Hugging Face. We ran several experiments with a variety of prompt types including: creative writing (example: "Create a short poem about the beauty that lies beneath artificial intelligence"), summarization of deep learning topics (summarize concepts related to deep learning), general QA (example: "What is the capital of France?"), contextual QA (example: "Who is the CEO of Apple?" - although most of these were questions where answers would be known), and generating code (such as a python function for addition).

Our objective was to determine if a well-optimized version of such a large language model could provide acceptable performance when running on consumer grade equipment.

TABLE 2 Benchmark Results (Latency & Efficiency)

Type Of Task	Time (s)	Input Tokens	Gen-Tks/s
Creative writing	404.4	22	0.49
Summarization	691.8	151	0.29
General QA	377.0	10	0.53
Contextual QA	416.7	29	0.48
Instruction Following	381.4	12	0.52

Logit Distillation performed on the 14B model trained on more than 5 billion tokens revealed some significant challenges even though optimizations were made. Deployed on a T4 GPU in a Colab environment with aggressive quantization (BitsAndBytesConfig in 4-bit, nf4 type, compute in float16), the critical latency (0.29 to 0.53 tokens/second), indicates that even a greatly-reduced 14-billion parameter model, post quantization will still be unsuitable for consumer-grade hardware. For instance, producing a short poem required over 400 seconds for 200 tokens, while answering a simple question ("What is the capital of France") required 377 seconds.

These results reveal a paradox: even after applying very aggressive 4-bit quantization to an already highly optimized model, the computational footprint of said model will remain too great for consumer-grade hardware. The source of this inefficiency comes from both hardware limitations (bandwidth of memory on the T4 GPU and time to perform dequantization/quantization) and a basic problem with classical logit distillation: it simply replicates the output distribution of the teacher without inheriting any of DeepSeek-V3's architectural innovations (MoE, MLA, MTP). Therefore, the distilled model receives absolutely none of the computational advantages associated with V3 and remains limited by sequential decoding and non-optimized kernel processing. Even though distilling successfully reduces the size of a model, it does not have the capability to address the architectural gap necessary to deploy models that can operate on every day devices. To close this gap one needs to develop techniques that allow one to replicate the ability of larger models while incorporating architectural innovations such as MoE, MLA, MTP — rather than solely attempting to reproduce what its teachers produce.

6 Conclusion

In conclusion, the use of novel architecture including the Mixture-of-Experts, Multi-Head Latent Attention and Multi-Tokens Prediction and a revolutionary training process including the use of the DualPipe empowers open source models like DeepSeek-V3 to achieve either competitive or higher than comparative performance benchmarks across all three areas of

natural language understanding, code generation and mathematical reasoning, competing with state of the art closed source models. Our independent benchmark on a 14 billion parameter distilled model (Arcee-ai / Virtuoso-Small-v2) running on a Single T4 GPU with aggressive 4-Bit Quantization shows that while highly optimized LLMs are capable of imposing prohibitive computational burden on consumer grade hardware; there exists a growing need for researchers to develop solutions to optimize powerful language models for architectures with severely limited resources.

REFERENCES

1. J. Kaplan, J. McCandlish, T. Henighan, T.B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, D. Amodei, Scaling Laws for Neural Language Models. arXiv preprint arXiv:2001.08361 (2020). https://doi.org/10.48550/arXiv.2001.08361
2. DeepSeek AI, DeepSeek-V3 Technical Report. arXiv preprint arXiv:2412.19437 (2024). https://doi.org/10.48550/arXiv.2412.19437
3. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin, Attention Is All You Need. Advances in Neural Information Processing Systems 30 (2017). https://doi.org/10.48550/arXiv.1706.03762
4. A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, Improving Language Understanding by Generative Pre-Training. OpenAI Technical Report (2018). https://doi.org/10.21428/d587d40a.5d9e33d2
5. A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, Language Models are Unsupervised Multitask Learners. OpenAI Technical Report (2019). https://doi.org/10.21428/d587d40a.5d9e33d2
6. T. Brown et al., Language Models are Few-Shot Learners. Advances in Neural Information Processing Systems 33, 1877-1901 (2020). https://doi.org/10.48550/arXiv.2005.14165
7. D. Patterson et al., Carbon Emissions and Large Neural Network Training. arXiv preprint arXiv:2104.10350 (2021). https://doi.org/10.48550/arXiv.2104.10350
8. OpenAI, GPT-4 Technical Report. arXiv preprint arXiv:2303.08774 (2023). https://doi.org/10.48550/arXiv.2303.08774
9. J. Su, Y. Cao, D. Murthaza, S. Wen, Y. Chang, B. Wang, RoFormer: Enhanced Transformer with Rotary Position Embedding. arXiv preprint arXiv:2104.09864 (2021). https://doi.org/10.48550/arXiv.2104.09864
10. F. Gloeckle, B. Youbi Idrissi, B. Rozière, D. Lopez-Paz, G. Synnaeve, Better & Faster Large Language Models via Multi-token Prediction. arXiv preprint arXiv:2404.19737 (2024). https://doi.org/10.48550/arXiv.2404.19737
11. D. Dai et al., Deepseekmoe: Towards Ultimate Expert Specialization in Mixture-of-Experts Language Models. CoRR abs/2401.06066 (2024). https://doi.org/10.48550/arXiv.2401.06066
12. DeepSeek AI, DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model. CoRR abs/2405.04434 (2024). https://doi.org/10.48550/arXiv.2405.04434
13. N. Shazeer et al., Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. Proceedings of the International Conference on Learning Representations (ICLR) (2017). https://doi.org/10.48550/arXiv.1701.06538

14. W. Fedus, B. Zoph, N. Shazeer, Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. arXiv preprint arXiv:2101.03961 (2021). https://doi.org/10.48550/arXiv.2101.03961
15. L. Wang, H. Gao, C. Zhao, X. Sun, D. Dai, Auxiliary Loss-Free Load Balancing Strategy for Mixture-of-Experts. arXiv preprint arXiv:2408.15664 (2024). https://doi.org/10.48550/arXiv.2408.15664