Mixture of Experts

Sparse Architectures, Routing, and Serving Infrastructure

February 2026 • Technical Report

Table of Contents

1. Introduction

Mixture of Experts (MoE) is a neural network architecture paradigm that enables models to scale to very large total parameter counts while keeping the computation required per input token manageable. Rather than passing every token through every parameter in the model, MoE architectures route each token to a small subset of specialized sub-networks (experts), activating only a fraction of the total parameters per forward pass. This sparse activation pattern decouples model capacity from computational cost, allowing MoE models to achieve the quality of much larger dense models at a fraction of the training and inference FLOPS.

This report examines MoE architectures in the context of large language models, covering the design principles, routing mechanisms, training challenges, and the unique infrastructure requirements for deploying MoE models in production.

2. Architecture Fundamentals

2.1 Structure of an MoE Layer

In a standard transformer, each layer contains a self-attention block followed by a feed-forward network (FFN). In an MoE transformer, some or all FFN layers are replaced with MoE layers. Each MoE layer contains N expert networks (typically 8 to 128, each structured identically to the original FFN) and a lightweight gating network (router) that determines which experts process each token. For a given input token, the router produces a probability distribution over experts, and the token is sent to only the top-k experts (commonly k equals 1 or 2). The outputs from the selected experts are combined using the router's probability weights.

2.2 Sparsity and Effective Parameters

The key advantage of MoE is that while the total parameter count may be very large, the **active parameter count** per token is much smaller. For example, Mixtral 8x7B has approximately 47 billion total parameters across its 8 experts, but each token activates only 2 experts per layer, yielding roughly 13 billion active parameters per token. This means inference compute comparable to a 13B dense model but with the representational capacity of a much larger model. Switch Transformer and GShard demonstrated this principle at even larger scales with top-1 routing across hundreds of experts.

3. Routing Strategies

3.1 Token-Choice Routing

In token-choice routing, each token independently selects its top-k experts based on the router's output probabilities. This is the most common approach and is used by Mixtral,

Switch Transformer, and most recent MoE models. The router is typically a simple linear layer that projects the token representation into a score for each expert, followed by a softmax and top-k selection.

3.2 Expert-Choice Routing

Expert-choice routing inverts the selection process: each expert selects the top-k tokens it wants to process from the full batch. This naturally achieves perfect load balance since every expert processes exactly the same number of tokens. However, it introduces the complication that different tokens may be processed by different numbers of experts (including zero), which can affect quality. Expert-choice routing has shown strong results in training but adds complexity to serving systems.

3.3 Shared and Fine-Grained Experts

Some architectures include shared experts that process every token alongside the routed experts. DeepSeek-V2 and V3, for instance, combine a small number of always-active shared experts with a larger pool of routed fine-grained experts. This hybrid approach ensures a baseline level of processing for every token while allowing specialization through routing. Fine-grained experts (smaller but more numerous) allow more precise routing decisions and better load distribution than fewer, larger experts.

4. Training Challenges

4.1 Load Balancing

A fundamental challenge in MoE training is ensuring that tokens are distributed reasonably evenly across experts. Without intervention, routing tends to collapse: a few popular experts receive most tokens while others are starved. This wastes capacity and creates computational bottlenecks. The standard mitigation is an auxiliary load-balancing loss that penalizes uneven expert utilization, added to the main training objective. Tuning the weight of this auxiliary loss is critical: too little allows collapse, while too much forces uniform routing that prevents meaningful specialization.

4.2 Expert Specialization

Effective MoE models develop experts that specialize in different aspects of the input distribution, such as different languages, domains, or syntactic patterns. However, the degree and nature of specialization is difficult to control directly and emerges from the training dynamics. Research has shown that specialization patterns vary significantly across layers, with early layers showing less specialization than later layers. Understanding and encouraging useful specialization remains an active area of research.

5. Inference and Serving Challenges

5.1 Memory Requirements

Despite their computational efficiency, MoE models have large total parameter counts that must be stored in memory. All expert weights must be resident in GPU memory (or accessible via fast offloading) even though only a subset is active for any given token. This means a Mixtral 8x7B model, despite computing like a 13B model, requires memory closer to a 47B model for weight storage. This memory-versus-compute asymmetry is the central deployment challenge for MoE architectures.

5.2 Batching Complexity

Batching MoE inference is more complex than batching dense models because different tokens in the same batch may route to different experts. This creates uneven workloads across experts and requires an all-to-all communication pattern when experts are distributed across multiple GPUs. The expert parallelism paradigm places different experts on different devices, requiring tokens to be dispatched to the appropriate device, processed, and then gathered back. Efficient implementation of this dispatch-compute-gather pattern is critical for serving performance.

5.3 Expert Offloading

For deployments where GPU memory is insufficient to hold all experts simultaneously, expert offloading strategies load experts from CPU memory or storage on demand. Since the router determines which experts are needed before the expert computation begins, prefetching can be used to load the required experts while other computation proceeds. Systems like MoE-Offloading and Fiddler have demonstrated that with careful scheduling, offloading can enable MoE inference on a single consumer GPU with acceptable throughput, though latency increases compared to fully GPU-resident deployment.

6. Comparison with Dense Models

Characteristic	Dense Model	MoE Model
Parameters per token	All parameters active	Subset active (e.g., 2 of 8 experts)
Training FLOPS efficiency	Baseline	Better quality per FLOP
Memory footprint	Proportional to active params	Proportional to total params
Inference throughput	Predictable, uniform	Higher but variable per token
Serving complexity	Standard	Requires expert parallelism, load

		balancing
Hardware utilization	Consistent	Can be uneven across devices

7. Notable MoE Models

Several landmark MoE models have shaped the field. **Switch Transformer** demonstrated top-1 routing at scale with up to 1.6 trillion parameters. **Mixtral 8x7B and 8x22B** from Mistral AI proved that open-weight MoE models could match or exceed much larger dense models in quality. **DeepSeek-V2 and V3** introduced fine-grained experts with shared expert designs and achieved state-of-the-art efficiency. **Grok-1** from xAI used a large MoE architecture for their conversational AI system. These models collectively demonstrate that MoE has moved from a research curiosity to a mainstream architectural choice for frontier language models.

8. Future Directions

MoE architectures continue to evolve in several directions. Larger expert counts with finer-grained experts are showing improved routing precision and specialization. Hybrid architectures that mix dense and MoE layers within a single model allow designers to place MoE layers where sparsity is most beneficial. Improved routing algorithms, including learned and reinforcement-learning-based routers, aim to better balance specialization with load distribution. On the infrastructure side, custom kernels and communication libraries optimized for the MoE dispatch pattern are narrowing the gap between theoretical and realized efficiency. As hardware evolves with faster interconnects and more flexible memory hierarchies, the practical advantages of MoE architectures are expected to grow further.

9. Conclusion

Mixture of Experts architectures represent a fundamental shift in how language models scale, decoupling model capacity from per-token computation. While MoE introduces complexities in training (load balancing, specialization) and serving (memory footprint, expert parallelism, uneven batching), the efficiency advantages are compelling: frontier-quality models that train and run at a fraction of the cost of equivalent dense architectures. As the ecosystem of MoE-optimized infrastructure matures, these models are poised to become the dominant architecture for large-scale language model deployment.