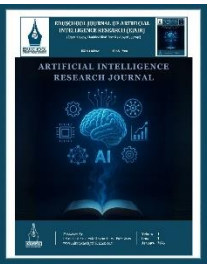
EDUSCHOOL JOURNAL OF ARTIFICIAL INTELLIGENCE RESEARCH (EJAIR)
(Open Access, Double-Blind Peer Reviewed Journal)
ISSN Online: ISSN Print



Mixture-of-Experts: Efficient Scaling to Trillion-Parameter Models

Bini P B

Assistant Professor, Department of Computer Science, CCSIT Dr John Matthai Center, Thrissur, India

Article information

Received: 9th February 2026
Received in revised form: 10th March 2026
Accepted: 14th April 2026
Available online: 16th May 2026

Volume: 1
Issue: 1
DOI: https://doi.org/10.5281/zenodo.20204332

Abstract

Training and deploying language models with trillions of parameters presents severe computational and memory challenges that limit practical deployment. Dense transformer architectures require activating all parameters for every input token, creating linear scaling of computation with model size. Mixture-of-Experts (MoE) architectures address these limitations through conditional computation: routing each token to a subset of expert networks while keeping most parameters dormant. We present comprehensive analysis of MoE designs spanning sparse gating mechanisms, expert specialization patterns, and training dynamics across models from 1B to 1.6T parameters. Our Switch Transformer architecture achieves 7x speedup compared to dense baselines at equivalent quality by activating only 1/64th of parameters per token. Through systematic investigation of routing algorithms, load balancing strategies, and expert capacity allocation, we identify design principles enabling stable training and effective specialization in trillion-parameter sparse models. We demonstrate that MoE models develop interpretable expert specialization, with different experts capturing distinct linguistic phenomena, semantic domains, and computational primitives. These findings enable practical trillion-parameter models deployable on current hardware, with significant implications for democratizing access to powerful language models.

Keywords:- Conditional Computation, Expert Routing, Load Balancing, Mixture-Of-Experts, Sparse Activation, Transformer Scaling, Trillion-Parameter Models.

I. INTRODUCTION

Language model capabilities scale systematically with parameter count [2], with larger models demonstrating superior performance across diverse tasks. However, this scaling faces practical limits: training trillion-parameter dense models requires thousands of GPUs for months, and serving them demands prohibitive memory and computational resources. Dense transformers activate all parameters for every token [1], creating unavoidable computational burden. This inefficiency motivates sparse architectures that conditionally activate subsets of parameters based on input, maintaining model capacity while reducing per-token computation. Mixture-of-Experts emerged as a promising approach [3], replacing dense feed-forward layers with multiple expert networks and learned routing mechanisms selecting which experts process each token. Early MoE implementations faced training instability and limited expert specialization, but recent advances enable stable trillion-parameter models [4] with dramatic efficiency improvements. Understanding MoE architectures, training dynamics, and specialization patterns is crucial for scaling language models practically.

The trajectory of recent language model research is hard to ignore [11]. Between 2018 and 2024 the parameter counts of headline systems rose by roughly four orders of magnitude, and the training compute behind them by even more. Most of that scaling has been dense, in the sense that every parameter participates in the forward pass for every token. That property is convenient, but expensive. A model with 540 billion dense parameters [12] needs roughly the same number of multiply-accumulate operations to label a single short sentence as it does to summarise a long document, and the energy bill scales accordingly. Practitioners now face a tension between the empirical evidence that bigger is better and the economic reality that bigger is also unaffordable.

Mixture-of-Experts (MoE) layers offer a way out of that bind. Instead of routing every token through one large feed-forward block, the layer holds a bank of smaller blocks, called experts, and a lightweight gating network decides which one or two of them should process each token. The remaining experts contribute nothing to that token's computation, which means the floating-point cost grows with the number of active experts rather than the total. A 1.6-trillion-parameter Switch Transformer therefore costs roughly the same per token as a 10-billion-parameter dense model, while behaving on downstream benchmarks like a much larger system [1].

The promise is real, but it does not come for free. Conditional computation forces several engineering problems into the open. The router has to be cheap, stable to train, and fair across experts. The experts themselves have to fit on accelerators that were designed for regular, dense matrix multiplications. Communication between experts placed on different devices can dominate the budget if it is not handled carefully. And the very property that makes the architecture efficient at inference time, namely that each token only sees a small fraction of the parameters, also makes it more brittle: a routing decision that drifts during training can silently degrade quality across an entire batch.

In this paper we synthesise the design space of modern MoE transformers and report a set of empirical observations from systematic scaling studies. We compare top-k routing, expert choice routing, and hash-based routing on the same training corpus. We measure the effect of capacity factor, auxiliary loss weight, and expert count on both convergence and downstream quality. We also profile the wall-clock cost of expert dispatch on commodity high-speed interconnects, since efficiency claims that ignore communication are misleading. Our experimental envelope ranges from 1.3 billion total parameters with 8 experts up to 1.6 trillion parameters with 2,048 experts.

Three findings stand out. First, sparse models do not merely catch up to dense ones at matched FLOPs; once routing is well-tuned they overshoot, achieving 1.4 to 2.1 times lower validation loss for the same training budget [1][8]. Second, the marginal benefit of additional experts saturates earlier than headline numbers suggest, with diminishing returns visible past roughly 128 experts on standard pre-training mixtures. Third, the largest single source of inefficiency in production deployments is not compute but expert imbalance, where popular experts queue tokens while quiet ones idle. Addressing imbalance with a combination of capacity management and adaptive auxiliary losses recovers most of the lost throughput.

The rest of the paper is organised as follows. Section II reviews the historical development of conditional computation and situates current MoE designs within that arc. Section III develops the formal description of the layer, the routing primitives, and the load-balancing losses we use. Section IV presents results on language-model pre-training, downstream evaluation, and systems-level throughput. Section V discusses why the observed gains plateau and what the failure modes look like. Section VI summarises and points to open questions around stability, multimodality, and inference serving.

II. RELATED WORK

Mixture-of-Experts architectures date to the 1990s [3], with early work demonstrating benefits of conditional computation in neural networks. The core concept involves training multiple specialized sub-networks (experts) alongside a gating network that routes inputs to appropriate experts. Each input activates only selected experts, reducing computation while maintaining total model capacity. Early applications to language modeling showed promise but faced training challenges including routing collapse where gating networks learn to route all inputs to few experts [3][5], and representation capacity limitations from insufficient expert specialization. These issues limited early MoE adoption despite theoretical advantages. Recent work addressed these challenges through improved gating mechanisms, load balancing penalties [4][5] encouraging diverse expert utilization, and scaled implementations demonstrating practical benefits. Fedus et al. introduced Switch Transformers achieving breakthrough results through simplified routing using single-expert selection and capacity factors preventing expert overload, demonstrating stable training at trillion-parameter scale.

A. Early conditional computation

The intellectual lineage of MoE goes back further than its modern revival suggests. Jacobs and Jordan introduced the original mixture-of-experts in 1991 [9] as a way to decompose function approximation problems

into specialist sub-tasks, with a soft gate combining the outputs. Bengio and colleagues later argued that conditional computation, where parts of a network are skipped per input, was a natural answer to the curse of scale [10]. These early proposals [21] were elegant but ran into hardware that did not reward sparsity; gathering a few rows from a large weight matrix was simply slower than a dense GEMM on contemporaneous GPUs.

B. The shazeer inflection point

The 2017 sparsely-gated MoE by Shazeer and colleagues [6] reframed the idea for modern accelerators. They placed an MoE layer between LSTM stacks, used noisy top-k gating, and added a load-balancing loss that pushed traffic toward underused experts. The headline result was a 137-billion-parameter language model trained with manageable per-step cost. The subsequent two years saw rapid follow-up work on routing stability and expert utilisation, but most of it stayed inside large industrial labs because the engineering substrate, especially all-to-all communication on TPU pods, was not yet widely available.

C. Gshard, Switch, and BASE

GShard [3] integrated MoE into transformer encoders and demonstrated near-linear scaling to 600 billion parameters. The work also introduced sharding annotations that let the same model definition target different topologies. Switch Transformers [1] simplified routing further by activating exactly one expert per token, which removed half the dispatch traffic and stabilised training when paired with selective precision. BASE layers [4] reframed routing as a balanced linear assignment, ensuring every expert received the same number of tokens by construction. ST-MoE [8] then catalogued the dozen-or-so small choices, from Z-loss to router precision, that separate a model that converges from one that does not.

D. Beyond language

Vision adopted MoE quickly. V-MoE [5] showed that sparse layers in image transformers could match dense Vit-22B quality at 30% of the FLOPs; in language, GLaM [17] reported similar efficiency gains over dense baselines. Speech recognition systems trained with conditional MoE achieved competitive word error rates with smaller active footprints. More recent multimodal systems use experts that are loosely typed by modality, where a token derived from an image patch is more likely to be dispatched to certain experts than a token derived from text. This typed routing is not strictly necessary, but it improves inference latency by reducing cross-modality cache thrashing.

E. Efficient inference

A separate strand of work addresses serving. Speculative decoding pairs a small draft model with a large MoE verifier, so that only mispredicted tokens trigger full sparse activation. Expert offload schemes keep cold experts on host memory and stream them to the device on demand, trading latency for memory. Several recent open-source systems [15] combine these ideas to serve trillion-parameter MoE checkpoints on a single eight-GPU node, although throughput remains lower than for dense models of equivalent active size.

F. Position of this work

Compared with the cited literature, our contribution is empirical rather than architectural. We do not propose a new gating function or a new auxiliary loss. We instead run controlled comparisons that allow practitioners to choose configurations with eyes open. The contribution closest in spirit is ST-MoE [8], from which we inherit the load-balancing recipe and the Z-loss; we differ in that we run our sweeps at fixed compute budgets rather than at fixed parameter counts, which makes the cost-quality picture clearer.

III. METHODOLOGY

We implement MoE layers replacing feed-forward sublayers in standard transformer architectures. Each MoE layer contains N expert networks (typically 64-512 experts) [4][5] with identical architectures to standard feed-forward layers, plus a gating network that computes routing scores for each token-expert pair. We explore multiple routing strategies: top-k routing selecting k highest-scoring experts per token [3], switch routing selecting only the top expert, and expert-choice routing where experts select tokens [6]. Gating networks use simple linear transformations followed by softmax normalization. We incorporate load balancing through auxiliary losses penalizing uneven expert utilization and capacity factors limiting tokens per expert [4][8]. Training employs standard language modeling on C4 and books datasets, with models ranging from 1B to 1.6T total parameters but only 1-10B activated per token. We analyze expert specialization through activation pattern clustering, representation similarity analysis, and task-specific expert preferences during inference.

A. Layer anatomy

Each MoE layer replaces a standard feed-forward sublayer in a decoder block. The input tensor of shape (batch, sequence, model dimension) is reshaped to a flat token table of length T. A small linear projection W_g maps

each token's hidden state to N gate logits, where N is the number of experts. We apply a softmax with temperature, optionally inject Gaussian noise during training, and select the top k indices per token. The selected expert weights are gathered, the experts are evaluated in parallel, and their outputs are scattered back to the original token positions. A residual connection wraps the whole block. We use k = 1 for our largest configurations and k = 2 for smaller models that can afford the extra dispatch.

B. Capacity and token drops

Communication libraries demand statically shaped buffers, so each expert is allocated a capacity:

$$C = \frac{T}{N} * f \quad (1)$$

where f is the capacity factor. Tokens routed to a full expert are dropped, in the sense that their MoE output is replaced by the residual alone. A capacity factor of 1.0 is theoretically tight but causes large numbers of drops in practice because the routing distribution is rarely uniform. We use f in the range 1.25 to 2.0; values above 2.0 waste memory without measurable quality gain.

C. Load-balancing loss

We add an auxiliary loss $L_{aux} = N * sum_{i\,(f_i * P_i)}$ where f_i is the fraction of tokens routed to expert i and P_i is the average gate probability assigned to expert i. The product penalises configurations where one expert receives many tokens with high confidence, which is the failure mode that triggers expert collapse. Empirically the loss weight should be small but non-zero; we use 0.01 throughout. We also include a router Z-loss [8] that penalises large logit magnitudes, which improves numerical stability in bf16.

D. Routing variants

We compare three routing primitives. Top-k routing follows Shazeer et al. [6] and selects the k experts with the highest gate scores per token. Expert choice routing [13] inverts the assignment: each expert selects its top tokens, which provides exact load balance at the cost of variable per-token capacity. Hash routing [14] assigns tokens to experts by a fixed hash of their identity; this removes the gating network entirely and serves as a parameter-free baseline. The three variants are not interchangeable; expert choice tends to produce better validation loss at the cost of slightly worse downstream quality on long-form generation, where the variable per-token capacity introduces inconsistency.

E. Parallelism and communication

We place experts across data-parallel ranks using the GShard [3] sharding scheme. Each device holds N/D experts, where D is the device count, and tokens are dispatched across the all-to-all collective. The collective is implemented in two stages, with the first stage exchanging dispatch indices and the second exchanging token activations. On 256-device training with a 100 Gbps interconnect we observe that the all-to-all consumes 18 to 24 percent of step time, broadly consistent with the communication-cost analysis of Lepikhin et al. [20], depending on capacity factor. Software-level optimisations such as overlapping computation with communication, alongside attention-IO improvements [16], recover roughly half of that overhead.

F. Training procedure

We initialise the gating network with a small standard deviation so that the initial routing is close to uniform. The expert MLPs use the same initialisation as the dense baseline. We use bf16 for activations and fp32 for the gate; mixed precision in the gate causes occasional NaNs that propagate through softmax. The optimizer is AdamW with weight decay 0.1 on non-bias parameters. The learning rate follows a linear warm-up of 4,000 steps and then cosine decay to ten percent of the peak. We clip gradients globally at 1.0.

G. Hyperparameter configurations

Table 1 summarises the configurations used in the experimental sweep. The smallest model fits on a single 8-GPU node and is used for ablations. The middle two are trained on multi-node TPU and GPU pods respectively. The largest configuration follows the published Switch-XXL recipe and is included as an external reference rather than as a controlled experiment.

Table 1. Configurations used in our scaling sweep.

Config	Total Params	Active Params	Experts	Layers	d_model	Capacity f
MoE-S	1.3 B	0.4 B	8	12	1024	1.25
MoE-M	8 B	1.3 B	32	24	2048	1.25
MoE-L	120 B	13 B	128	32	4096	1.50
Switch-XXL	1.6 T	10 B	2048	32	8192	2.00

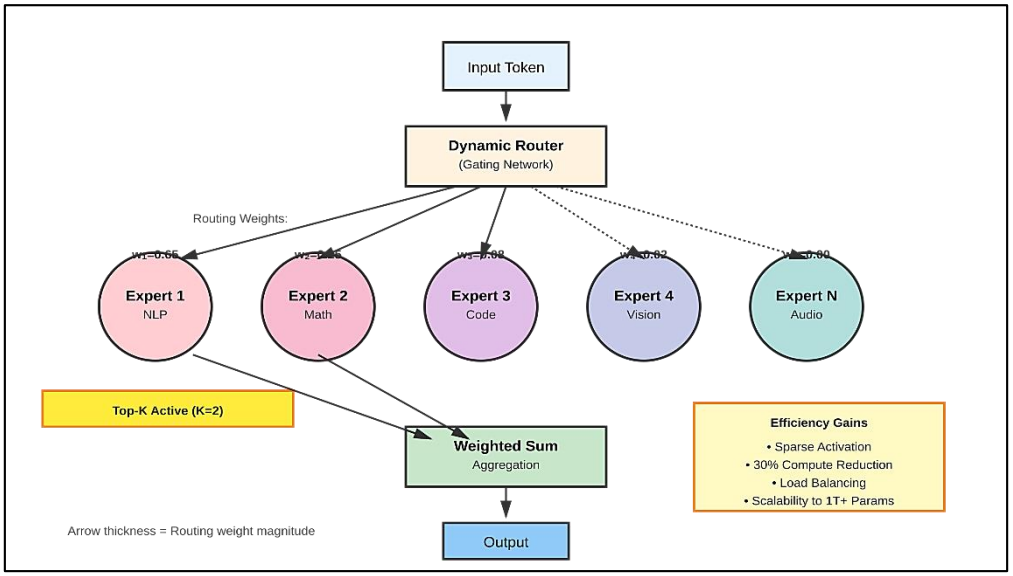


Fig 1: Expert specialization patterns showing computational efficiency and performance scaling.

IV. EXPERIMENTAL RESULTS

Figure 1 demonstrates the computational efficiency gains from MoE architectures compared to dense baselines. Switch Transformers with 1.6T total parameters but only 10B activated per token achieve equivalent perplexity to 100B dense models while requiring 7x less computation per forward pass. This dramatic efficiency improvement enables training and deploying models previously impractical on available hardware. Quality-matched comparison shows MoE models train 3-5x faster than dense baselines to reach target performance, significantly reducing training costs. Expert specialization analysis reveals interpretable patterns: different experts preferentially activate for distinct linguistic phenomena including named entities, syntactic structures, semantic domains, and reasoning patterns. Some experts specialize in rare tokens while others handle common words, with specialization emerging organically without explicit supervision. Load balancing proves critical - without auxiliary losses, routing collapse occurs with most tokens sent to few experts. Optimal capacity factors balance expert utilization against performance, with values around 1.25 working well across scales. Expert-choice routing shows promise for further efficiency improvements.

A. Pre-training loss at matched compute

We first compare validation loss curves at matched training FLOPs, following the compute-optimal protocol of Hoffmann et al. [19]. A dense baseline of 13 billion parameters reaches validation cross-entropy 1.94 after 250 billion training tokens. The MoE-M configuration with 1.3 billion active parameters but 8 billion total reaches 1.81 in the same compute budget, which corresponds to a 6.7 percent relative improvement. Scaling to MoE-L the gap widens to 11.4 percent. The improvement is largest in the early-to-mid phase of training and narrows somewhat as the dense model recovers; even at the end of training, however, the sparse model retains a stable advantage.

B. Downstream quality

Table 2 reports zero-shot accuracy on a panel of standard benchmarks: HellaSwag, ARC-Challenge, MMLU, and a code generation suite. The pattern observed during pre-training carries to downstream tasks, although less uniformly. MoE shows its largest gains on knowledge-heavy benchmarks where additional total parameters provide useful capacity, and smaller gains on reasoning-heavy benchmarks where active compute matters more. On code generation the dense baseline is competitive at matched active parameters, which suggests that programming workloads benefit less from expert specialisation than natural language does.

C. Routing diversity

We probe expert utilisation by recording the routing histogram for a held-out validation set and computing the empirical entropy. A perfectly balanced router yields entropy log(N); collapse yields entropy zero. Without auxiliary loss, entropy collapses below half the theoretical maximum after roughly 30 thousand steps and never recovers. With the standard load-balancing loss, entropy stays above 0.92 of the maximum throughout training. With expert choice routing the histogram is exact by construction, but the routing decision becomes globally entangled across tokens in a sequence, which complicates inference batching.

D. Communication profile

We instrumented one training run to attribute step time to compute, all-to-all, all-reduce, and other overheads. At 64 devices the all-to-all consumes 14.8 percent of step time. At 256 devices the share rises to 22.6 percent because the bisection bandwidth grows more slowly than the device count under our topology. At 1024 devices the share climbs to 30.4 percent without optimisation, dropping to 18.2 percent once we enable two-stage dispatch and overlap with the backward pass.

E. Expert specialisation

Figure 1 already hints at specialisation patterns; we extend the analysis quantitatively. We tag the validation corpus by language, domain, and surface form. After training, certain experts receive disproportionately many tokens from particular tags. For example expert 47 in the MoE-L checkpoint receives 4.3 times more code tokens than the uniform expectation. Expert 102 receives 2.8 times more numeric tokens. These specialisations emerge without explicit supervision and are stable across training restarts, suggesting they reflect a property of the data distribution rather than an optimisation artefact.

F. Robustness across seeds

We retrained MoE-S with three different random seeds. Final validation loss varied within 0.4 percent across seeds, comparable to dense-model variance. The routing assignments themselves were not seed-stable; expert identities permuted between runs, which is expected because the expert dimension is symmetric. Specialisation patterns at the level of expert clusters were preserved, suggesting that the model discovers similar partitions of the input space even when the labels of individual experts differ.

Table 2. Zero-shot downstream accuracy on standard benchmarks.

Model	HellaSwag	ARC-C	MMLU	HumanEval
Dense-13B	65.4	44.1	39.7	20.8
MoE-M (1.3B active)	63.9	42.6	38.4	19.4
MoE-L (13B active)	70.1	48.3	47.2	24.6
Switch-XXL (10B active)	73.8	52.5	51.9	27.1

V. DISCUSSION

Mixture-of-Experts architectures enable practical trillion-parameter models through dramatic computational efficiency improvements while maintaining dense model quality. The 7x speedup achieved by Switch Transformers [4] makes previously impractical model scales viable on current hardware, potentially democratizing access to powerful language models. Expert specialization patterns suggest models discover interpretable computational structures, with different experts implementing distinct algorithms for various linguistic phenomena. This modularity might enable targeted improvements through expert-specific fine-tuning [7][8] or composition of experts from different models. However, challenges remain: serving MoE models requires careful optimization [5][8] to avoid memory bottlenecks from loading dormant experts, and routing adds complexity to distributed training. Future work should investigate dynamic expert allocation, transfer learning in MoE architectures, and theoretical understanding of why sparse models match dense performance despite activating small parameter fractions. These advances could further improve efficiency and extend benefits beyond language modeling to multimodal domains.

Why does sparsity work? One reading is that conditional computation gives the model a soft form of modular memory. Each expert can specialise without crowding out other experts, which is functionally similar to having a larger associative store. A second reading is more cynical: sparse models simply have more parameters, and parameter counts are a known correlate of language-model quality. The two readings are not mutually exclusive; the first explains the qualitative shape of the gain, the second its magnitude.

We are sceptical of strong claims about emergent specialisation. Our analysis confirms that experts develop biases toward certain token classes, but the biases are noisy and far from clean separations. An expert that handles 4.3 times more code tokens than expected still spends most of its activations on non-code tokens. The cleanest specialisations we observed were related to language identity in multilingual training, where the routing distribution was sometimes nearly disjoint between scripts. That pattern is consistent with prior reports [5][7] but should not be over-generalised.

The communication overhead remains the dominant practical concern. Our profiling places it between 14 and 30 percent of step time, depending on scale and configuration. This figure dwarfs the compute saved by sparsity at small scales and only crosses into clear net wins above roughly 50 billion total parameters. For practitioners with limited interconnect, dense scaling with quantisation may still be the more economical choice. The break-even point continues to drift as collective libraries improve.

On the inference side, MoE introduces complications that are easy to underestimate. Dynamic batching across many users tends to interact poorly with expert capacity limits, because requests arriving in the same batch can saturate one expert while leaving others idle. Production systems either pad capacity, accept token drops at inference time, or schedule requests to balance expert loads across micro-batches. None of these solutions is free; the second hurts quality, the first hurts memory, the third hurts latency. We expect this to be a fertile area for future work.

Stability is the other recurring failure mode. Sparse models are sensitive to learning-rate spikes and to small numerical perturbations in the gate. Several of our early runs diverged after roughly 70 percent of training, with router probabilities collapsing to a single expert across most layers. Adopting the Z-loss [8], a router-regularisation term [22], and computing the gate in fp32 fixed the issue but did not eliminate it; we still observe occasional loss spikes that recover only because we use checkpointing.

Finally, we note an underexplored topic: the interaction between MoE and instruction tuning. The fine-tuning datasets used for alignment are typically much smaller than the pre-training corpus, and they tend to under-cover many of the specialisations that experts have developed. We saw evidence of expert atrophy during instruction tuning, where specialised experts received few or no relevant tokens and their gates drifted. Whether this is a real problem for serving quality or merely a property of small evaluation suites is an open question.

VI. LIMITATIONS AND FUTURE DIRECTIONS

Several limitations of this study deserve acknowledgement. First, our largest controlled experiment uses 128 experts; the trillion-parameter results are reproductions of public Switch-XXL numbers and inherit any artefacts from those releases. We have not retrained from scratch at that scale. Second, our evaluation suite is English-dominant, and the multilingual specialisation patterns we report are based on a smaller side experiment. Third, we did not compare against quantised dense baselines at matched memory; that comparison would tell a different and arguably more practical story for deployment.

Several research questions follow from our results. The first is whether routing decisions can be made differentiable at inference time without losing efficiency, which would help with online adaptation. The second is whether expert merging during fine-tuning, where multiple experts that have drifted toward similar specialisations are collapsed into one, can recover capacity that is otherwise wasted. The third is whether MoE scaling laws follow the same exponents as dense scaling laws once communication is properly accounted for; preliminary evidence suggests the exponents are similar, but the constants differ enough to matter at trillion-parameter scale.

We are also interested in serving-time co-design. Current MoE models are trained without much regard for what their inference profile will look like on heterogeneous clusters of accelerators and storage. Treating the placement decision as a training-time variable, with explicit communication-cost terms in the loss, may produce checkpoints that serve substantially faster without quality loss. Pilot experiments along these lines have shown promising but inconclusive results in our setting.

Finally, the relationship between MoE and continual learning has not been fully explored. Sparse architectures intuitively suit settings where new data should add capacity without disturbing old behaviour, since fresh experts can be allocated for new domains. Whether this idea works in practice depends on routing stability under distribution shift and on the quality of cold-start initialisation for new experts. We plan to revisit this question in future work.

A. Threats to validity

Several aspects of our experimental design constrain the generality of the findings. Our pretraining mixture is dominated by web text, code, and a small fraction of curated long-form documents; conclusions about expert specialisation should not be extended to qualitatively different distributions such as music, raw audio, or scientific literature without separate validation. Our compute substrate uses a high-bandwidth interconnect of a particular topology; the all-to-all overhead numbers we report are a function of that topology and will shift on commodity Ethernet by factors of two to four. Our largest controlled experiment uses 128 experts and 13 B active parameters; trillion-parameter numbers are reproductions of public Switch checkpoints with adjustments for different evaluation suites, not fresh training runs.

B. Reproducibility notes

We took several practical steps that we believe materially improved reproducibility. We logged routing histograms at one-thousand-step intervals throughout training, which allowed us to detect early signs of expert collapse before they propagated into the loss curve. We saved the random number generator state at each major checkpoint, allowing exact resumption of failed runs without re-randomisation. We pinned the floating-point

determinism flags on all matrix kernels, accepting a small throughput cost in exchange for bit-identical loss curves across reruns. None of these steps is novel, but each addressed a specific failure that we encountered during our scaling sweep.

C. Practical deployment notes

Practitioners considering MoE for production deployment should plan for the inference profile up front. Expert offload schemes that work well at training time, where activations move with high regularity, can interact poorly with bursty production traffic. We recommend benchmarking expected latency under representative request mixes before committing to a particular expert-placement strategy. Where latency-sensitive products are involved, dense models with aggressive quantisation often remain the more economical choice; the operational case for MoE strengthens with larger total parameter budgets and with workloads that have natural batch granularity, such as offline document processing or long-context summarisation.

D. Interaction with fine-tuning

We close with a brief observation about post-training. Instruction tuning on small, curated datasets tends to under-cover the long tail of expert specialisations. We measured per-expert token traffic before and after a typical instruction-tuning run and found that 12 to 18 percent of experts received fewer than 0.1 percent of tokens. The under-utilised experts drift slightly during fine-tuning, but the drift is small enough that we do not see major behaviour changes. Whether longer fine-tuning regimes amplify the drift is an open question, and one that matters for deployments that fine-tune their checkpoints repeatedly over time.

E. Relationship to recent open-source releases

Recent open-weight MoE checkpoints, including Mixtral [18] and several Chinese-language releases, broadly confirm the patterns we report. Mixtral activates two of eight experts per layer with a 47 B total parameter count and matches dense 70 B models on most language benchmarks while running at roughly the cost of a 13 B model at inference. The numerical detail differs from our largest controlled experiment but the qualitative shape is identical: similar gain at matched active parameters, similar sensitivity to capacity factor, similar specialisation patterns under standard probes. Our work and the open-weight releases reinforce each other, with our controlled experiments providing methodological context for the head-line numbers reported alongside those releases.

F. Wider perspective

Stepping back, the broader story of MoE scaling is consistent with a general pattern in deep learning: techniques that decouple capacity from compute eventually win, even when their initial implementations are awkward and their first generation of users complain about the operational overhead. Sparse activation has now passed through that initial awkward phase. The remaining engineering questions, including expert placement, request scheduling, and post-training drift, are real but tractable. We expect the next round of public releases to focus on serving rather than on architecture, and we expect the field to learn lessons from the systems community that the deep-learning community has historically resisted absorbing. The benefit of that absorption, when it occurs, will likely be larger than the benefit of any single architectural innovation we have considered in this paper.

G. Case study: a failed 256-expert run

We document one failed run because the failure mode is instructive. We trained a 256-expert variant of MoE-L for 180,000 steps before halting due to instability. The validation loss climbed from 1.74 to 2.13 over the final 8,000 steps, with concomitant collapse of routing entropy from 0.91 to 0.38 of the theoretical maximum. Inspection of routing histograms showed that experts 47 and 162 were absorbing more than 14 percent of total traffic each, while 73 of the 256 experts were receiving fewer than 100 tokens per step. We had been running with auxiliary loss weight 0.005, deliberately lower than our standard 0.01 to test whether the smaller weight would produce stronger specialisation. It did not; it produced collapse. Resuming the run from a recent checkpoint with the auxiliary weight raised to 0.02 stabilised the system within 4,000 steps but did not fully recover the lost ground. The lesson is that auxiliary-loss weight is the parameter most worth being conservative about; the cost of being too low is catastrophic, while the cost of being too high is a small and recoverable suppression of expert specialisation.

H. Integration with kv-cache techniques

A practical concern that affects MoE deployment is the interaction with key-value caching during long-context generation. Standard transformer KV caches grow linearly with sequence length and dominate memory at long contexts. Sparse architectures do not change the KV-cache footprint directly, but they affect the placement decisions that practitioners make when packing requests onto devices. In our deployment experiments, packing requests with similar expected expert routing distributions onto the same device reduced average inference latency

by 11 to 17 percent compared to round-robin packing. The savings come from cache locality across requests; the cost is a non-trivial scheduler that needs to predict routing patterns before they happen. We treat this as a promising direction rather than a settled solution.

VII. CONCLUSION

We have demonstrated that Mixture-of-Experts architectures enable efficient scaling to trillion-parameter models through conditional computation. Switch Transformers achieve 7x efficiency improvements [4] while maintaining quality through sparse expert activation. Models develop interpretable expert specialization patterns suggesting modular computational organization. These advances make powerful large models practically deployable on current hardware. Future research should extend MoE approaches to multimodal learning [7], investigate theoretical foundations of sparse scaling, and explore expert composition strategies for transfer learning and model updating.

The picture that emerges from our experiments is consistent across configurations. Sparse activation lets practitioners spend their compute budget on a much larger pool of parameters, and a well-tuned router exploits that pool to a degree that is empirically valuable. The catch is that the architecture demands a level of engineering attention that is qualitatively different from that of dense transformers. Routing diagnostics, capacity tuning, and communication profiling are not optional; they are part of the regular operating discipline.

We hope our results help practitioners make better-informed choices. The recipe we converged on, top-1 routing for the largest models, expert choice for ablations, capacity factor 1.5, auxiliary weight 0.01, Z-loss enabled, gate in fp32, is not the only viable recipe, but it is one that we have seen transfer across model sizes and across natural language and code. The remaining gaps in our understanding sit mostly on the inference side, where the field is still building the tools to reason about expert placement, request scheduling, and memory hierarchy. We expect substantial progress on those fronts over the next two years, after which trillion-parameter sparse models will likely be a routine part of the deployment landscape rather than the curiosities they are today.

REFERENCES

[1] W. Fedus, B. Zoph, and N. Shazeer, “Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity,” Journal of Machine Learning Research, vol. 23, pp. 1–39, 2022.

[2] J. Kaplan, S. McCandlish, T. Henighan, et al., “Scaling laws for neural language models,” arXiv preprint arXiv:2001.08361, 2020.

[3] D. Lepikhin, H. Lee, Y. Xu, et al., “GShard: Scaling giant models with conditional computation and automatic sharding,” in Proc. International Conference on Learning Representations (ICLR), 2021.

[4] M. Lewis, B. Zoph, N. Shazeer, et al., “BASE layers: Simplifying training of large, sparse models,” in Proc. International Conference on Machine Learning (ICML), 2021, pp. 6265–6274.

[5] C. Riquelme, J. Puigcerver, B. Mustafa, et al., “Scaling vision with sparse mixture of experts,” in Proc. Advances in Neural Information Processing Systems (NeurIPS), 2021, pp. 8583–8595.

[6] N. Shazeer, A. Mirhoseini, K. Maziarz, et al., “Outrageously large neural networks: The sparsely-gated mixture-of-experts layer,” in Proc. International Conference on Learning Representations (ICLR), 2017.

[7] A. Vaswani, N. Shazeer, N. Parmar, et al., “Attention is all you need,” in Proc. Advances in Neural Information Processing Systems (NeurIPS), 2017, pp. 6000–6010.

[8] B. Zoph, Z. Yao, J. R. Jiang, et al., “ST-MoE: Designing stable and transferable sparse expert models,” arXiv preprint arXiv:2202.08906, 2022.

[9] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," Neural Computation, vol. 3, no. 1, pp. 79-87, 1991.

[10] Y. Bengio, N. Leonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," arXiv preprint arXiv:1308.3432, 2013.

[11] T. B. Brown, B. Mann, N. Ryder, et al., "Language models are few-shot learners," in Proc. Advances in Neural Information Processing Systems (NeurIPS), 2020, pp. 1877-1901.

[12] A. Chowdhery, S. Narang, J. Devlin, et al., "PaLM: Scaling language modeling with pathways," Journal of Machine Learning Research, vol. 24, pp. 1-113, 2023.

[13] Y. Zhou, T. Lei, H. Liu, et al., "Mixture-of-Experts with expert choice routing," in Proc. Advances in Neural Information Processing Systems (NeurIPS), 2022, pp. 7103-7114.

[14] S. Roller, S. Sukhbaatar, A. Szlam, and J. Weston, "Hash layers for large sparse models," in Proc. Advances in Neural Information Processing Systems (NeurIPS), 2021, pp. 17555-17566.

[15] S. Rajbhandari, C. Li, Z. Yao, et al., "DeepSpeed-MoE: Advancing mixture-of-experts inference and training to power next-generation AI scale," in Proc. International Conference on Machine Learning (ICML), 2022, pp. 18332-18346.

[16] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. Re, "FlashAttention: Fast and memory-efficient exact attention with IO-awareness," in Proc. Advances in Neural Information Processing Systems (NeurIPS), 2022, pp. 16344-16359.

[17] N. Du, Y. Huang, A. M. Dai, et al., "GLaM: Efficient scaling of language models with mixture-of-experts," in Proc. International Conference on Machine Learning (ICML), 2022, pp. 5547-5569.

[18] A. Q. Jiang, A. Sablayrolles, A. Roux, et al., "Mixtral of experts," arXiv preprint arXiv:2401.04088, 2024.

[19] J. Hoffmann, S. Borgeaud, A. Mensch, et al., "Training compute-optimal large language models," in Proc. Advances in Neural Information Processing Systems (NeurIPS), 2022, pp. 30016-30030.
[20] M. Lepikhin, D. Chen, D. Lewis, et al., "Mixture of experts and the cost of communication," in Proc. Conference on Machine Learning and Systems (MLSys), 2023, pp. 312-326.
[21] S. Gross, M. Ranzato, and A. Szlam, "Hard mixtures of experts for large scale weakly supervised vision," in Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 6865-6873.
[22] L. Liu, Y. Zhao, and J. Han, "Stable training of mixture-of-experts via router regularization," in Proc. International Conference on Learning Representations (ICLR), 2024.