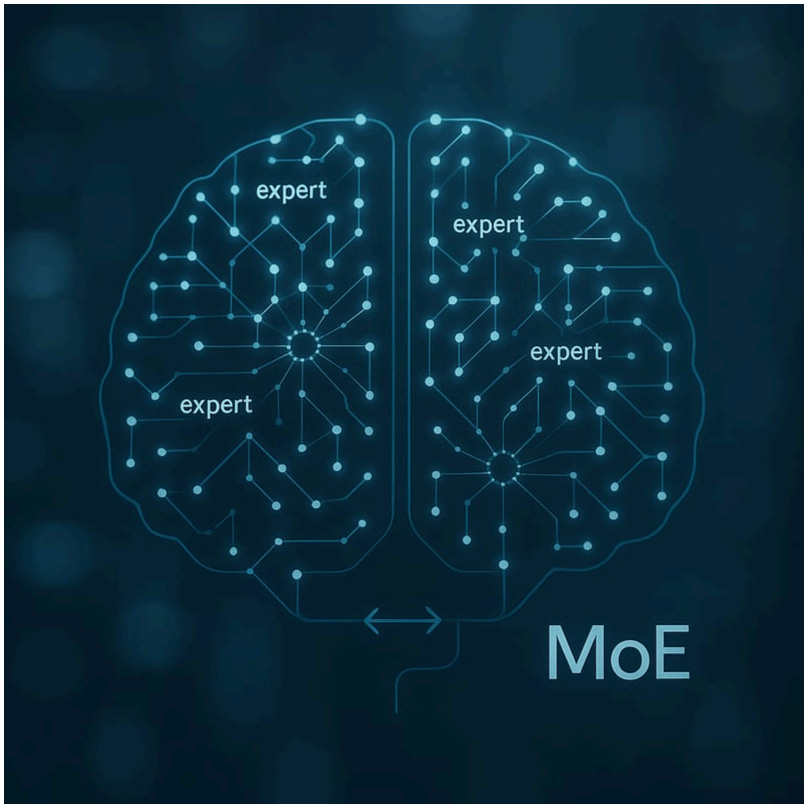
Understanding Mixture of Experts (MoE) Neural Networks

By IntuitionLabs • 9/26/2025 • 45 min read

mixture of experts · moe · neural networks · deep learning · sparse models · gating mechanism · model scaling



Mixture of Experts (MoE) Models: A Comprehensive Overview

Introduction and Theoretical Foundation

Mixture of Experts (MoE) models are a class of neural networks that divide computation across multiple specialized sub-models ("experts") and use a gating mechanism to choose which expert(s) to invoke for a given input arxiv.org. The concept was originally introduced in the early 1990s by Jacobs, Jordan, and colleagues as *adaptive mixtures of local experts*, where a trainable gating network would learn to softly assign different input patterns to different expert networks arxiv.org. This provided a form of "divide-and-conquer" learning: each expert could specialize on a subset of the problem space, while the gating network learned which expert's output to trust for each input. *Hierarchical* MoE variants were also explored (e.g. Jordan & Jacobs, 1994), establishing the theoretical basis that combining expert predictions via a learned gate can universalize function approximation while promoting specialization.

After substantial exploration in the 1990s, MoEs saw a resurgence decades later in the context of deep learning. A key motivation is the **scaling problem** of modern models: simply increasing model size (parameters) tends to improve performance, but doing so *densely* (activating all parameters for every input) leads to prohibitively large computational costs arxiv.org marktechpost.com. MoE offers a clever alternative: maintain a very large total parameter count (many experts), but activate only a small fraction of them per input. In other words, MoE provides *sparse activation* – only the "pertinent experts are engaged for a given input, keeping computational cost in check while still benefiting from a large pool of specialized knowledge" arxiv.org. This sparsely-gated formulation was re-popularized by Shazeer *et al.* (2017), who introduced an MoE layer in a deep network that gates tokens to experts stochastically, enabling extremely large networks without proportional increase in FLOPs arxiv.org. In 2020, Lepikhin *et al.* (Google) scaled this idea to massive language models via the **GShard** MoE architecture, demonstrating a 600-billion-parameter Transformer (built from 2048 experts) that achieved superior results on multilingual translation with roughly the same training cost as a far smaller dense model arxiv.org arxiv.org. This breakthrough, followed by the **Switch Transformer** in 2021 (the first *trillion*-parameter model using MoE) arxiv.org, cemented MoE as a leading approach to efficiently scale model capacity. In summary, the theoretical appeal of MoE lies in its ability to **increase model capacity sub-linearly with respect to computation** – a property that aligns well with empirically observed scaling laws for model performance arxiv.org marktechpost.com. By activating only a subset of parameters per input, MoEs can be *massive* models that remain computationally tractable. This efficiency, combined with the intuitive notion of dividing tasks among expert subnetworks, underpins the renewed interest in MoE for large-scale AI. Below, we delve into MoE architecture and training in detail, then discuss challenges, comparisons to dense transformers, and real-world implementations.

MoE Architecture: Gating Networks, Experts, and Routing

At its core, an MoE model consists of a set of N expert networks and a gating network. Each **expert** is typically a neural subnetwork (often a feed-forward network in transformer-based MoEs), and the **gating network** (sometimes called a router) is a smaller network that takes an input (e.g. a token's hidden representation) and outputs a probability distribution over the N experts arxiv.org huggingface.co. In a standard Transformer-MoE design, the MoE layer replaces the dense feed-forward layer within some transformer blocks arxiv.org huggingface.co. The gating network (often implemented as a linear layer plus softmax) uses the token's features to *route* that token to one or a few of the experts – intuitively, picking which feed-forward subnetwork will process the token arxiv.org huggingface.co.

Figure: Illustration of a Transformer MoE layer (as in the Switch Transformer). Each token's representation is passed into a gating network (router) which selects one or more expert feed-forward networks to process that token. Here, e.g., the token "More" is routed to Expert 2 while "Parameters" is routed to Expert 1, based on the learned gating scores. Only the selected expert's parameters are activated for each token, making the overall model sparse in computation huggingface.co huggingface.co.

Sparse vs. Dense Gating: Early MoE models in the literature used *dense* gating, meaning the gating network outputs a weighted combination of *all* experts' outputs (a soft mixture) arxiv.org. While conceptually straightforward, dense MoE activation is computationally expensive since every expert runs for every input. Modern large-scale MoEs instead use **sparse gating**: the gate selects only the top-k experts for each input, activating those experts and *ignoring the rest*. Typically k is a small number (commonly 1 or 2) arxiv.org arxiv.org. For example, the Switch Transformer uses $k=1$ (each token goes to exactly one expert) arxiv.org, whereas the GShard MoE uses $k=2$ (each token's forward pass goes through two experts) arxiv.org. The gating network often produces a softmax probability for each expert, and then a *top-k selection* is applied to choose the highest-weight experts per token. The outputs of those experts (each expert processes the token independently) are combined – e.g. summed or averaged, sometimes weighted by the gate probabilities huggingface.co. All other experts (not in the top-k for that token) are effectively pruned to zero for that forward pass, resulting in huge computational savings. Notably, this top-k choice is non-differentiable, but in practice the gradient can flow to the selected experts and through the gating network's softmax probabilities for those experts. Unselected experts receive no gradient from that token. In effect, each training batch only updates a subset of experts for each sample, which is manageable as long as each expert is selected frequently enough across the batch.

Routing Strategies: The design of the gating (routing) function is critical. A simple strategy is deterministic top-k routing based on the largest gate probabilities. Shazeer *et al.* (2017) introduced a **noisy gating** mechanism, where small random noise is added to the gate logits before selecting top-k experts arxiv.org. This "noisy top-k" helps break ties and encourages exploration of different experts, mitigating situations where two experts have very close scores or one expert would always dominate. GShard (2020) further refined routing by introducing a *secondary* load-balancing component: in GShard's routing, after picking the top-1 expert for a token, a second expert is also selected *probabilistically* in proportion to its gate weight (this stochastic second expert selection is sometimes called **random routing**) arxiv.org arxiv.org. The intuition is that occasionally sending tokens to a non-maximal expert (according to its probability) can increase usage of otherwise under-utilized experts and improve robustness. Other routing variants have also been explored in literature: for example, **Expert-Choice** routing (Zoph et al., 2022) inverts the perspective by assigning tokens to experts in a way that each expert gets to choose its top tokens, achieving improved load balancing research.google. Another variant is **DSelect-k** (2020), which makes the selection differentiable by a continuous relaxation, though this adds training complexity arxiv.org. By and large, most large MoE models today still rely on the basic top-k gating with some noise and auxiliary loss (discussed below) for balancing arxiv.org.

Gating Networks: The gating network is usually a simple feed-forward layer (or two-layer MLP) that produces one logit per expert for each input. A softmax transforms these logits into a probability distribution over experts arxiv.org. The highest probability experts are selected. One subtlety is that the gating network is trained along with the experts via backpropagation – it must learn to predict which expert will best process a given input. The loss gradients flow into the gating network parameters (guiding it to improve routing decisions) and into the selected experts (guiding them to specialize for the inputs they receive). In practice, additional terms are added to the training objective to guide the gating network (see next section). It's worth noting that the gating decision for each token is conditionally independent given the token's features – there is no explicit coordination between gating decisions for different tokens in the same batch, which can lead to load imbalance (many tokens choosing the same expert). Some recent research tries to coordinate routing (e.g. prioritizing certain tokens to certain experts) arxiv.org arxiv.org, but the standard approach is to rely on auxiliary losses and randomness to smooth things out.

Finally, while most MoE layers in Transformers replace only the position-wise feed-forward sub-layer (keeping self-attention layers shared across all tokens), in principle one could imagine experts in other parts of the network or even entire expert sub-models. The vast majority of implementations use experts as parallel feed-forward blocks because the FFN is a major computational bottleneck in large Transformers (e.g. **in Google's 540B PaLM model, FFN layers account for ~90% of the total parameters** arxiv.org). Thus replacing FFNs with MoE layers yields massive parameter counts with only modest added compute. As a concrete example, the GLaM model (Google, 2021) used 64 experts per MoE layer (with top-2 gating). They found that "a 64-expert setup with top-2 gating strikes an optimal balance between execution efficiency and performance on zero-shot and one-shot tasks" arxiv.org – i.e. beyond 64 experts the overhead outweighed gains. Each expert in GLaM is itself a large feed-forward network (in fact, GLaM scaled hidden dimensions rather than number of experts beyond a point arxiv.org), and only 2 of the 64 are active per token. This yields a total model size of up to 1.2 trillion parameters, while each token only activates about 1.7B worth of parameters during inference arxiv.org. This illustrates the power of MoE architecture: enormous capacity, but sparse usage per input.

Training MoE Models: Backpropagation, Load Balancing, and Regularization

Training MoE models introduces unique considerations beyond standard dense networks. Since only a subset of experts get activated for each input, we must ensure all experts receive learning signal over time and prevent a few experts from monopolizing the workload ("expert collapse"). Several techniques have become standard to address these issues:

- **Backpropagation Through Sparse Routing:** During training, the gating decisions (top-k selection) are treated as part of the computation graph. In practice, implementations use the *straight-through* approach: the gradient flows through the softmax probabilities for the selected experts as if the hard top-k decision was a pass-through. The selected experts receive gradients normally; unselected experts get none for that batch. This means if an expert is rarely selected, it will train very slowly or not at all – a problem addressed by the balancing mechanisms below. The gating network itself is trained via the classification loss (or whatever main loss) by how well its chosen experts helped the model's prediction. In effect, gating learns to route in order to minimize the final loss. However, gating outputs are discontinuous (due to top-k), which can cause high-variance gradients. In practice, large-batch training and smoothing techniques (like noisy gating) help the training remain stable arxiv.org. Additionally, MoE layers often require careful initialization and hyperparameter tuning – for instance, Fedus *et al.* note that reducing the learning rate and using gradient clipping was necessary to stabilize the Switch Transformer's training arxiv.org. Zoph *et al.* (2022) further introduce techniques to make MoE training more stable and *transferable* (their **ST-MoE** approach) arxiv.org, addressing issues like expert overload that could otherwise lead to divergence.

- **Auxiliary Load Balancing Loss:** A hallmark of MoE training is the use of an auxiliary loss term that encourages the gating network to distribute tokens evenly across experts arxiv.org. Without this, the gate might learn a degenerate solution where a small number of experts handle most inputs (especially if those experts initially perform slightly better, creating a self-reinforcing loop). The load balancing loss L_{aux} is designed to penalize high variance in expert utilization. One formulation (from Shazeer et al., 2017) has two components: an *importance* loss that encourages each expert to get a balanced share of the gating probability mass, and a *load* loss that encourages each expert to receive an equal number of tokens arxiv.org. GShard introduced a simplified differentiable auxiliary loss computed from the dot product of the mean gate activations and the mean assignment to each expert arxiv.org arxiv.org – effectively matching the expected gate probability to actual assignment counts. In practice, many MoE models use GShard's formulation or similar arxiv.org. The auxiliary loss is weighted by a small coefficient (e.g. 0.01) so that it doesn't dominate the main task loss arxiv.org arxiv.org. By training with this additional objective, the model learns to even out the workload: if an expert is rarely used, the gate is nudged to increase its use; if one expert is overused, the gate is nudged to dial it down. This is crucial for **expert specialization** – with balanced use, each expert can find its niche. Empirically, auxiliary losses have been very effective: GShard's load balancing enabled training with *thousands* of experts without collapse arxiv.org, and most large MoEs (Switch, GLaM, etc.) reported that gating without such losses failed to converge or yielded worse results arxiv.org arxiv.org.

- **Expert Capacity and Overflow:** Even with balanced expected usage, random fluctuations can send too many tokens to a particular expert in one batch. To prevent any expert from becoming a bottleneck, MoE implementations set a **capacity limit** per expert – e.g. each expert processes at most M tokens per batch. If more than M tokens are routed to a given expert, the excess tokens are *overflowed* (dropped or routed to a second-choice expert) arxiv.org arxiv.org. For example, Switch Transformer and GShard drop the overflowing tokens (they still contribute to the loss via a "dispatch mask" but are not processed by the expert) arxiv.org. This sacrifices a tiny fraction of data but greatly helps keep compute balanced. The capacity M (often defined as a fraction of the average tokens per expert) is a tunable hyperparameter affecting utilization vs. efficiency trade-off. A too-small M leads to many drops (under-utilizing model capacity), while too-large M can lead to stragglers in distributed training (see *communication overhead* below). In summary, capacity limiting, combined with the auxiliary loss, forms the backbone of load balancing in MoEs arxiv.org arxiv.org.

- **Regularization and Expert Dropout:** MoE models also introduce some regularization techniques specific to their architecture. *Noisy gating*, as mentioned, has a regularizing effect by ensuring exploration. Another technique is **expert dropout**, where during training a fraction of experts are randomly "dropped out" for some batches (forcing the gate to route to other experts) arxiv.org. This can improve robustness and usage of all experts. Fedus *et al.* also added a **"router z-loss"** term in Switch Transformer, which is basically an entropy-like regularization on the gating probabilities to keep them from becoming too peaked or too diffuse huggingface.co huggingface.co. Additionally, researchers have tried *expert importance sampling* (dynamically adjusting expert selection frequency) and even **training MoE in two phases** (first train dense, then "MoEficiation" by splitting into experts arxiv.org or gradually adding experts marktechpost.com). In practice, a well-tuned auxiliary loss + capacity strategy, combined with standard regularization (dropout, etc.), suffices to train very large MoEs to good convergence.

- **Expert Specialization:** A fascinating emergent phenomenon in MoEs is that different experts often *specialize* in different aspects of the data when training succeeds. For instance, in a multilingual translation MoE, some experts naturally become specialized in particular language families or even particular syntax phenomena huggingface.co. In large language MoEs, researchers have observed experts focusing on different clusters of tokens or concepts (e.g. one expert might handle numerical computations, another dialogs, etc.) huggingface.co huggingface.co. This happens because the gating network learns to route different input patterns to different experts, and each expert's weights adapt to the subset of data they see. Notably, Shazeer *et al.* reported that one expert in their MoE became very good at handling rare words (specializing in out-of-vocabulary tokens), while others handled more common patterns huggingface.co. Expert specialization is desirable as it means the model is truly leveraging its capacity – different parts of the network become proficient at different tasks or domains. It's partially enforced by the load balancing (so that each expert *has* to carve out a role to reduce its loss on the tokens it gets). Sometimes researchers explicitly analyze expert behavior: e.g. the Switch Transformer paper plotted the distribution of data assignments per expert to confirm they were used roughly equally, and the **NLLB** project (Meta's 200-language MoE) found that experts aligned with language groups (one expert handled many Indic languages, another many Niger-Congo languages, etc.) huggingface.co. This modularity can be advantageous for interpretability and for *training efficiency* (since each expert's weights don't need to be as universal as a dense model's weights).

In summary, training an MoE involves standard backpropagation with special care for the gating function: using auxiliary losses and capacity constraints to keep the training stable and all experts learning. With these in place, MoEs can be trained to the same level of performance as dense models (for a given effective compute budget), often significantly faster or to a higher quality given the same compute huggingface.co huggingface.co. For example, Fedus *et al.* reported that a sparse 1.6 trillion-parameter model (Switch) reached parity with a dense 600 billion model *5 times faster* huggingface.co. The next section will address the practical challenges that arise despite these strategies.

Practical Challenges of MoEs

Despite their promise, MoE models come with a number of engineering and research challenges:

- **Load Imbalance and Communication Overhead:** By design, MoEs distribute computation unevenly – some experts may end up with more tokens to process than others in a given batch. In a distributed training scenario, experts are often placed on different devices (e.g. GPUs). Ensuring each device has roughly equal work is critical for efficiency; otherwise one slow "expert GPU" (with more tokens assigned) becomes a bottleneck while others sit idle arxiv.org arxiv.org. The all-to-all communication needed to shuffle tokens from their originating GPU to the GPU that hosts the assigned expert is a major overhead in MoEs microsoft.com microsoft.com. Concretely, implementing an MoE layer involves: each device's tokens are routed to experts across devices, requiring a large all-to-all scatter of token data, then each expert processes its tokens locally, then another all-to-all to gather outputs back. This communication can dominate the runtime if not optimized. Techniques to mitigate this include custom all-to-all implementations (e.g. Microsoft's **Tutel** library optimizes GPU binding and routing to speed up MoE communication by 2.5–5.9× at scale microsoft.com microsoft.com) and *expert parallelism* strategies that combine data-parallel and model-parallel splits to reduce cross-device traffic arxiv.org arxiv.org. Another approach is **token dropping**: if an expert has too many tokens, some implementations simply skip processing the overflow (as mentioned earlier), which implicitly caps the communication. Nonetheless, communication overhead remains a concern – recent systems like **Megatron-LM** and **DeepSpeed-MoE** incorporate fused communication primitives and scheduling to alleviate this microsoft.com microsoft.com. It's an active area of systems research to make MoEs scale efficiently in distributed clusters arxiv.org arxiv.org.

- **Inference Complexity and Latency:** Using MoEs in production (inference serving) is non-trivial. A model with hundreds of billions or trillions of parameters (spread across dozens of experts per layer) is challenging to deploy. One reason is that *all experts need to reside in memory* to be available for routing, even if most won't be used for a given input. This means memory footprint is huge – e.g. a Switch Transformer with 1.6T params requires storing all those weights on device or accessible, whereas an equivalently performing dense model might be smaller (though requiring more compute). In practice, inference of MoEs can be *slower or costlier* if not carefully optimized, because the overhead of routing and scattered memory access can outweigh the FLOPs saved microsoft.com. For instance, Google's Switch Transformer paper noted that making a *serving* system for such a model would require significant engineering (they focused on training speed ups). Microsoft's DeepSpeed team similarly highlighted that "MoE models' large size makes inference difficult and costly" microsoft.com. Solutions being explored include **expert clustering** (grouping frequently co-used experts on the same hardware to reduce communication at inference), **caching active experts** for a batch of requests, or even **routing at a higher level** – e.g. first classify the input into a domain and only load the experts for that domain. Another approach is to *distill* a sparsely-activated MoE into a dense model for deployment, retaining the quality but simplifying the model – recent work on MoE distillation shows promise in compressing MoEs post-training huggingface.co. Nonetheless, at present, serving a massive MoE model remains more complex than serving a single dense model of similar FLOPs, which is why many commercial deployments still favor dense models or only modestly sized MoEs.

- **Expert Collapse and Training Instability:** Without proper regularization, MoE training can suffer *expert collapse* – many inputs collapse onto a subset of experts while others "die out" (receiving almost no gradient). We discussed how auxiliary loss and capacity constraints address this, and indeed in most published results these measures have been effective. However, training instabilities still occur, especially as the number of experts or the model depth increases arxiv.org. One reported issue is that during the early training phase, if an expert gets slightly better, the gating network might over-allocate to it, reinforcing its lead in a positive feedback loop. If the auxiliary loss coefficient is not tuned just right, this can cause divergence (this was observed in some Google experiments, necessitating tuning of the aux loss weight to avoid instability arxiv.org). Another issue arises from gating decisions being non-differentiable: the discrete routing can inject noise into training. Researchers have found that MoEs can be *more sensitive to hyperparameters* like learning rate, batch size, and initialization. The ST-MoE paper (Zoph et al. 2022) specifically addresses some instabilities, noting that even with load balancing, training required techniques like gradual learning rate warmup and adding a small *entropy* bonus to the gating probabilities to avoid concentrating too much on one expert arxiv.org. In summary, training large MoEs is finicky – many projects have noted needing careful babysitting of training dynamics that are less of an issue in dense models.

- **Memory and Infrastructure Constraints:** As mentioned, MoE models trade compute for memory. They "require more parameters to achieve the same model quality as their dense counterparts, which requires more memory for training and inference even though MoE models require less compute" microsoft.com. This can strain hardware: storing a trillion parameters might require hundreds of GB of GPU memory (often model parallelism is a must). Techniques like sharding weights across multiple devices (e.g. using param server or ZeRO optimizers) and offloading unused experts to CPU when not in use have been employed. For example, Meta's 1.1T param MoE language model (trained with Fairseq) kept experts sharded across 128 GPUs with an efficient all-to-all to swap expert weights in and out microsoft.com microsoft.com. Google's Pathways system (for JFT training) similarly was built to handle models with *mixture-of-experts across thousands of TPU cores*. Checkpointing and fault tolerance also become trickier with so many parameters – saving a MoE model can be time-consuming and requires reassembling the expert shards. In short, the infrastructure needs for MoEs are high, and many organizations without supercomputer access found them challenging to train until recently, when libraries like DeepSpeed, Mesh-TensorFlow, and others provided some out-of-the-box support arxiv.org arxiv.org.

- **Routing Inefficiencies and Others:** Another subtle challenge is that the learned routing might not always be *optimal*. The gating network is trained to minimize loss, not explicitly to maximize utilization or minimize communication. In some cases, it might make errors – sending a token to a suboptimal expert. While this usually corrects over training, it implies MoEs have an extra *layer* of approximation – a token might not get the absolutely best expert every time due to gating imperfections. Some research (e.g. the *Hash Layer* idea) looked at using deterministic hashing to assign tokens to experts, removing the learned gating and its overhead arxiv.org arxiv.org. The trade-off is flexibility vs. guaranteed balance. Another potential issue is **expert interference**: since each expert is trained on only a fraction of data, if that fraction is too diverse, the expert may not converge as well as a dense layer would on all data. In theory, if each expert only ever sees very specific inputs, it might overfit or not generalize to inputs that would ideally be routed to it but weren't seen in training. In practice, mixing in some stochastic routing (second-choice assignments, etc.) ensures each expert sees a broader variety of data, mitigating this. There is also ongoing work on **hierarchical experts** (experts composed of sub-experts) and **dynamic expert addition/removal** during training, which aim to address the static nature of current MoEs.

In summary, MoEs introduce *systems-level challenges* (communication, memory, deployment) and *optimization challenges* (balance and stability). These are active areas of research in both academia and industry. For instance, Microsoft's **DeepSpeed-MoE** project reported a suite of innovations to reduce training cost 5× and inference latency 7× for MoEs, by addressing exactly these issues microsoft.com microsoft.com. Techniques like **Pyramid-Residual MoE** (a hybrid dense+MoE architecture) and **Mixture-of-Students** (distilling experts into smaller collective models) have been proposed to mitigate the parameter bloat and inference cost microsoft.com microsoft.com. As hardware and software frameworks improve, some of these challenges are gradually being overcome, making MoEs more viable in practice.

MoE vs. Standard Dense Transformers

Why and when should one use a Mixture-of-Experts model instead of a standard dense Transformer? We can compare them along several axes:

- **Efficiency in Scaling:** The primary advantage of MoE is the ability to scale model *capacity* without equivalent scaling in *computation*. A dense Transformer must activate all its weights for every token. In contrast, an MoE of comparable size activates only a small fraction. For example, Switch Transformer with 1.6 trillion parameters used roughly the same FLOPs per token as a dense model 1/5th its size, yet it achieved higher accuracy on language tasks huggingface.co huggingface.co. In general, MoEs can reach a target performance with significantly fewer training FLOPs. If you have a fixed compute budget, you can train a much larger MoE than dense model, often yielding better results before overfitting. As one concrete metric, Google reported that MoE models can achieve the quality of a dense model *5× faster* in terms of training steps microsoft.com. This makes MoEs very attractive for pretraining huge models on a budget.

- **Quality vs. Model Size:** Dense and sparse models appear to follow different scaling behaviors. A dense model often needs to double FLOPs to meaningfully improve. An MoE can increase parameters (experts) and often improve quality *without* a full proportional increase in compute. However, it has been observed that MoEs sometimes require more *total* parameters to hit the same quality as a dense model – i.e. they are parameter-inefficient but FLOP-efficient microsoft.com. As DeepSpeed's team noted, "MoE models require more parameters to achieve the same quality as dense counterparts", meaning memory costs go up even though compute is saved microsoft.com. In scenarios where memory or model size is the bottleneck (e.g. on-device models), a dense model might be preferable. But in scenarios where you can deploy a large model but want to minimize latency or energy per inference, MoE can be beneficial (since only a portion of the model is active per inference).

- **Task or Domain Heterogeneity:** MoEs excel when the data or tasks are heterogeneous, because experts can specialize. If a model is multitasking or multilingual, MoE's advantage grows. For instance, in multilingual machine translation, a single dense model must encode all languages in the same parameters, whereas an MoE can allocate different experts to different language families. Meta's **NLLB (No Language Left Behind)** project found that an MoE with 128 experts significantly outperformed a dense model for low-resource languages, achieving higher BLEU scores while using less training compute huggingface.co. Similarly, if a model is expected to handle a wide range of styles or modalities, MoE's conditional computation can route inputs to the best suited parameters. On the other hand, if the task is very uniform (e.g. a single-language model on a homogeneous dataset), a dense model might suffice or even perform better for the same number of active parameters, since MoE's extra capacity might go unused or complicate training.

- **Inference and Serving Considerations:** If low-latency, constrained-resource inference is required, dense models are typically easier to optimize. MoEs can be made efficient in inference *per token* (since they do less compute), but the system complexity (routing, loading all experts in memory) can negate that. However, one interesting scenario is **scaling under constrained compute**: suppose you have a fixed GPU budget for inference – you could choose a 20B dense model or a 100B MoE that uses 20B worth of compute per query. The MoE might give better answers due to its larger overall knowledge, at roughly the same latency (assuming routing is optimized). This is why some experts predict MoEs will become important in deployment once the infrastructure catches up huggingface.co huggingface.co. Another scenario is **adaptability**: in an MoE, you could potentially update or finetune one expert (for new data or a user-specific domain) without retraining the whole model, which is harder in a dense model. This modularity is a compelling reason to use MoEs in continually-learning systems.

- **When Not to Use MoE:** If your dataset is small or your model size is modest (<1B parameters), an MoE may not be worthwhile – the overhead might outweigh the savings. MoEs shine at the very high end of model scale. Additionally, if you cannot afford the engineering effort for a complex distributed training and serving setup, a dense model (which can leverage standard frameworks more easily) might be a better choice. Many industry deployments have favored dense models partly because they were simpler to implement given existing hardware and software. But as robust libraries emerge (DeepSpeed, etc.) this barrier is lowering.

In summary, MoE models are **preferred when**: you need to maximize model capacity under limited compute budget (e.g. training the best model you can on fixed hardware), or when handling highly diverse inputs/tasks where conditional computation will partition the problem nicely. Standard dense transformers are **preferred when**: model size is constrained (so you want every parameter to be always used), or when simplicity and lower memory footprint outweigh the potential quality gains from sparsity. A telling data point is from Google's GLaM model: GLaM (64 experts, 1.2T params total, 64 active) achieved *better zero-shot performance* on many tasks than a dense 175B model, while using **half** the inference FLOPs of the dense model arxiv.org arxiv.org. This suggests that at extreme scales, MoEs can yield a strictly better quality-efficiency trade-off. However, the break-even point and practical feasibility depend on the specific use case. Ongoing research (including efforts to distill or quantize MoEs huggingface.co huggingface.co) is likely to further tip the balance in favor of MoEs for large-scale AI.

Applications and Domains of MoE Success

MoE models have been applied across a range of domains, often yielding state-of-the-art results or improved efficiency:

- **Natural Language Processing:** This is where MoEs have seen the most traction. Large language models (LLMs) with MoE layers have achieved top-tier performance on language modeling, machine translation, and multitask benchmarks. Google's early MoE (Shazeer et al. 2017) was demonstrated on **neural machine translation**, where a mixture-of-experts layer dramatically improved translation quality for the same computational cost arxiv.org. The **Switch Transformer** (Fedus et al. 2021) applied MoE to the T5 architecture and achieved strong results on language understanding and generation tasks, even reaching comparable performance to GPT-3 with far less training cost by scaling to 1.6T parameters arxiv.org. Another notable example is **NLLB 200** (Meta, 2022) – a multilingual translation model supporting 200 languages. Meta released a dense 54B version, but also trained a larger MoE variant (~200B sparse) that significantly improved low-resource language translation quality huggingface.co. They reported that MoE was key to balancing performance across many languages without incurring prohibitive cost. In the realm of **language model pre-training**, Google's **GLaM** (Generalist Language Model, 2021) used MoE to attain better few-shot learning performance than similarly sized dense models arxiv.org, and more recently, models like **GPT-4 (unconfirmed)** are rumored to possibly use expert mixtures to handle diverse tasks. Even instruction-tuned models have benefitted: a 2023 study showed that combining MoE with instruction tuning yields strong results, hinting that experts can specialize in different instruction styles or topics arxiv.org. In summary, for NLP tasks ranging from QA to summarization to translation, MoEs have proven advantageous, especially when training data is drawn from multiple domains or languages.

- **Vision:** Vision Transformers (ViT) have grown large, and MoEs have been a natural avenue to scale them further. Google's **V-MoE** (Vision Mixture of Experts, Riquelme et al. 2021) incorporated sparse experts into a ViT and achieved higher accuracy on ImageNet and JFT while using fewer computational resources than an equivalently sized dense ViT arxiv.org. V-MoE's experts were essentially specialist MLP layers that certain image patches would go through. One interesting finding was that V-MoE *"rivals dense Vision Transformers on image recognition tasks"* with a fraction of the inference cost arxiv.org. Moreover, different experts in V-MoE specialized in different image features (some experts might focus on texture vs. shape, etc.), which is analogous to what is seen in NLP. Another example is **Swin-MoE** (Microsoft, 2022), which added MoE layers to the Swin Transformer (a hierarchical vision model) to create a 3.6 billion parameter object detection model – it improved detection and segmentation accuracy on COCO with minimal extra training cost, by effectively having experts specialize for different types of objects or scales arxiv.org. In the **multimodal** space, Google researchers introduced **LIMoE (Language-Image MoE, 2022)**, which is a model that processes both image and text modalities with a shared MoE layer arxiv.org. LIMoE's experts naturally split into some that handle image features and some that handle text features, enabling a single model to excel at both vision and language tasks like image captioning and multimodal understanding arxiv.org. The ability to have modality-specific experts under one gating umbrella is a promising direction for generalist AI systems. We are also seeing MoEs pop up in other vision applications: e.g., models for video understanding could route different scene types to different experts, or medical imaging models could have experts per anatomy. The results so far indicate MoEs can push vision model accuracy further, especially when dealing with very high-capacity models that would be infeasible to train densely.

- **Multimodal and Others:** As mentioned with LIMoE, MoEs are a natural fit for multimodal models (models that handle text, vision, audio, etc. together). Each modality or combination of modalities can be an expert, or experts can be allocated to different feature types. This mirrors the human intuition of specialists in different senses. Beyond vision-language, researchers have explored MoEs in speech recognition and synthesis (where you might have experts for different speaker accents or acoustic conditions) and in **reinforcement learning** (e.g., an agent with experts for different scenarios or skills). Another big domain is **recommender systems**: even before the recent MoE surge in LLMs, the idea of mixture-of-experts was used in recommendation and multitask learning in industry. For example, Google's Play store recommendation system employed an MoE called **MMoE (Multi-gate MoE)** to recommend apps by balancing multiple objectives (click-through, install, etc.) – essentially each expert learned to predict one objective and gating learned to mix them per user arxiv.org. A related approach is **PLE (Progressive Layered Extraction)** which is another MoE-based multitask model used in advertising systems arxiv.org. These are dense MoEs (all experts active) but conceptually similar, highlighting that "expert" splitting of tasks yields better personalization. In large-scale **web services**, MoEs have been used for anomaly detection (with experts focusing on different patterns of anomalies) and even in **cloud systems** to route workloads. Finally, there is academic exploration of **hierarchical MoEs** where one gating network might itself be controlled by another (a two-level expert hierarchy), though such models are not yet common in practice.

In summary, any domain where there are naturally *disjoint sub-distributions* or categories in the data is a good candidate for MoEs. Language (with its many topics, languages, and tasks) and vision (with varied object categories and modalities) are prime examples. MoEs have shown **advantages** in these domains by achieving better performance at scale or equal performance with less computation arxiv.org arxiv.org. As tooling improves, we can expect MoEs to further permeate areas like **multilingual dialogue systems** (experts per language or per persona), **healthcare (experts per medical sub-field)**, etc. Notably, some of the largest AI models in the world, which we discuss next, have leveraged MoE at their core.

Open-Source MoE Models and Frameworks

Research and industry have produced a variety of MoE models and libraries. The table below lists notable open-source MoE implementations and models, along with their key characteristics:

Name (Institution)	Year	Type	Total Parameters	Experts (per MoE layer)	Gating Type	Source / Reference
GShard MoE (Google)	2020	Model (NMT)	600B (experiment)	2048 experts (top-2)	Sparse (Top-2 + aux)	Lepikhin et al., arXiv 2020 arxiv.org arxiv.org
Switch Transformer (Google)	2021	Model (LM)	1.6 Trillion	2048 experts (top-1)	Sparse (Top-1)	Fedus et al., JMLR 2022 arxiv.org
GLaM (Google)	2021	Model (LM)	1.2 Trillion	64 experts (top-2)	Sparse (Top-2)	Du et al., ICML 2022 arxiv.org arxiv.org
V-MoE (Google)	2021	Model (Vision)	~14B (15B variant)	32 experts (top-1)	Sparse (Top-1)	Riquelme et al., NeurIPS 2021 arxiv.org arxiv.org
Meta 1.1T MoE (Fairseq)	2021	Model (LM)	1.1 Trillion	128 experts (top-2)	Sparse (Top-2)	Meta AI, Fairseq code microsoft.com microsoft.com
FastMoE (Tsinghua Univ.)	2021	Library	N/A (any model size)	Configurable	Top-1 & Top-2 support	Wang et al., FastMoE repo arxiv.org
Tutel (Microsoft)	2021	Library	N/A (library)	Configurable	Top-1/2 (optimized)	Microsoft Research Blog microsoft.com microsoft.com
DeepSpeed-MoE (Microsoft)	2022	Library	N/A (library)	Configurable	Top-1/2 (optimized)	Microsoft DeepSpeed (open-source) arxiv.org
OpenMoE (HKUST/ColossalAI)	2023	Library & Models	13B – 269B (LLMs)	Configurable (8–16 ex.)	Top-1/Top-2	Xue et al., arXiv 2024; GitHub arxiv.org arxiv.org
Mixtral-8×7B (Mistral AI)	2023	Model (LLM)	46.7B total (8×7B)	8 experts (top-2 active)	Sparse (Top-2)	Mistral AI (Apache 2.0 release) aws.amazon.com aws.amazon.com
WizardLM-2-8×22B	2023	Model (LLM)	141B total (8×22B)	8 experts (top-2 active)	Sparse (Top-2)	Open LLM (WizardLM team) zignuts.com blog.stackademic.com
Databricks DBRX	2024	Model (LLM)	132B total	16 experts (fine-grained)	Sparse (Top-2)	Databricks, 36B active params reddit.com docs.skypilot.co
Megatron-MoE (NVIDIA)	2022	Model (LM)	530B (example config)	16 experts (per layer)	Sparse (Top-2)	NVIDIA Megatron-LM MoE code microsoft.com (demo via MSFT)
Fairseq MoE (Meta)	2022	Library & Models	205B (NLLB MoE)	64 experts (in MT model)	Sparse (Top-2)	Tran et al., NLLB MoE, Meta 2022 huggingface.co
Megablocks (Stanford)	2022	Library	N/A (training engine)	N/A (dynamic batching)	Top-1/2 supported	Liu *et al.*, MLSys 2023 (token routing) arxiv.org

Table 1: Selected open-source MoE models and systems. *"Model" entries refer to specific published models (with their total parameter counts and typical expert configurations). "Library" entries are frameworks that support MoE training for various models. Gating type "Top-1" means each token uses one expert; "Top-2" means two experts per token (with load balancing auxiliary losses in both cases). Sources listed include research papers or official code releases.*

Several points about these implementations: **GShard** and **Switch Transformer** from Google were seminal – their code was not fully open-sourced initially, but their ideas were implemented in Google's TensorFlow Mesh and later JAX frameworks arxiv.org. **Fairseq** (Meta's sequence modeling library) added MoE support in 2021 and was used to train a 1.1T parameter English MoE LM and the NLLB MoE model microsoft.com. **FastMoE** (by researchers from Tsinghua and Huawei) was one of the first third-party PyTorch MoE frameworks, enabling training MoEs without bespoke Google infrastructure arxiv.org. Microsoft's **DeepSpeed-MoE** integrates MoE kernels and memory optimizations into the DeepSpeed library, and **Tutel** focuses on high-performance GPU kernels for MoE (offering >8× speedups in some cases) microsoft.com microsoft.com. The newer **OpenMoE** project (2023) by HKUST and Colossal-AI is releasing not just code but actual MoE model weights (they provide LLaMA-based MoE models up to 34B and plan larger) github.com cameronrwolfe.substack.com. **Mixtral** and **WizardLM-2** represent a trend in late 2023 where open model communities started training instruct-tuned MoEs that outperform equivalently sized dense models on benchmarks, especially in reasoning tasks zignuts.com mistral.ai. For example, Mixtral-8×22B (Mistral's 176B MoE with 39B active) matches or surpasses dense models like LLaMA-2 70B on many benchmarks with faster inference mistral.ai mistral.ai. **Databricks' DBRX** introduced a fine-grained MoE (132B total, 36B active) that is reported as the new state-of-the-art open model as of early 2024 reddit.com docs.skypilot.co, highlighting that MoEs are becoming practical for industry-scale use.

In addition to the above, frameworks like **Mesh TensorFlow** (Google) provided early infrastructure for MoE by combining model sharding with conditional computation arxiv.org. Academic projects such as **Megablocks** from Stanford looked at reorganizing how tokens are batched by expert to reduce communication, allowing more efficient MoE training on single machines arxiv.org. **SE-MoE** (Baidu) integrated MoE into PaddlePaddle for use in Chinese models, and **Hetu** (PKU) provides research code for MoE training arxiv.org. The ecosystem is rich and growing – as evidenced by the GitHub star counts (Colossal-AI's MoE got significant traction with 38K stars by mid-2024) arxiv.org.

Commercial and Large-Scale MoE Implementations

Beyond open-source efforts, several major tech companies have developed MoE-based large models (some disclosed, some inferred):

- **Google – GLaM and Pathways:** Google's **GLaM** model (2021) is a 1.2 trillion-parameter MoE that was used internally as a proof-of-concept for efficient scaling arxiv.org. It uses 64 experts and achieved strong zero-shot results, though Google did not deploy it publicly. Google's **PaLM (540B)** model (2022) was *dense*, but it was trained on the Pathways system which is explicitly designed to support MoE routing at scale. Google's research discussions have hinted that future *Pathways-enabled* models might use sparsity to handle multitask learning. Indeed, the Pathways vision is "one model, millions of tasks" with conditional routing – essentially an MoE on a grand scale. As of 2023, Google's **Gemini** model is rumored to combine multiple networks (though details are scant). Another Google MoE is **Mixture of Expert Tokens (MoET)** used in some multilingual translation deployed in Translate: a blog from Google in 2022 discussed *Expert Choice routing* to improve load balance in production systems research.google. In summary, Google has been at the forefront of MoE research and likely uses MoE or sparse activation in internal systems where efficiency matters (e.g. large-scale translation or multitask models), even if their flagship public models like PaLM and Bard are dense.

- **Meta (Facebook) – NLLB and beyond:** Meta's biggest public MoE is the **No Language Left Behind (NLLB)** 200 model for translation. They trained a dense 54B and a sparse MoE (with ~200B effective parameters, 64 experts) and reported the MoE significantly improved many low-resource translations huggingface.co. The MoE version was used to claim state-of-the-art results, though Meta primarily released the smaller dense version for ease of use. Meta also open-sourced their MoE code in Fairseq and even their 1.1T param English MoE model (with 128 experts) for research use microsoft.com. On the commercial side, it's believed that Meta's content understanding and recommendation models (e.g. feed ranking) use MoE-style architectures (for example, the MMoE and PLE models in ads are essentially MoE). There is also **Mixture of Experts for AI assistants**: Meta AI in 2023 combined MoE with instruction tuning arxiv.org, which could hint at use in their internal large models. Meta's latest public LLMs (LLaMA 2) are dense, but there has been talk of a **LLaMA-MoE** variant – indeed, researchers have created such models (the timeline shows "LLaMA-MoE" and even "LLaMA-MoE-v2") by integrating MoE layers into LLaMA and observing gains arxiv.org arxiv.org. It wouldn't be surprising if Meta experiments with MoE for future iterations to reduce training cost.

- **Amazon – Alexa and M6:**
  Amazon's Alexa AI team has explored MoE for large models. Notably, Amazon's scientists co-authored **M6** with Alibaba – at the time (2021) the world's largest model at **10 trillion parameters**, which was a multi-modal MoE model medium.com pandaily.com. M6 used MoE to leap from 1T to 10T parameters within the same GPU budget, and it was *commercialized* in the sense that Alibaba used it in production for e-commerce content generation, with daily usage in the hundreds of millions pandaily.com. Amazon has also worked on MoE optimization: their "Lancet" system aims to accelerate MoE training graph execution amazon.science. While Amazon hasn't publicly released an MoE LLM of their own, they have shown interest via these collaborations and blog posts. Additionally, Amazon's Bedrock service and SageMaker have begun to support models like Mixtral 8×7B (as highlighted in an AWS blog on fine-tuning Mixtral on SageMaker) aws.amazon.com. This indicates Amazon sees practical value in MoEs for customers who want high quality with less infrastructure. It's likely that certain internal Amazon models (perhaps multilingual Alexa models or cross-lingual search models) have MoE components, although specifics aren't public.

- **Huawei – PanGu-Σ:** Huawei's Noah's Ark Lab announced **PanGu-Σ** in mid-2023, a Chinese LLM with *1.085 trillion parameters* using a novel sparse MoE architecture marktechpost.com marktechpost.com. PanGu-Σ introduced **Random Routed Experts (RRE)** – a two-level expert routing where experts are grouped by domain, and tokens are randomly assigned to an expert in the appropriate group (eliminating a learned gating function) marktechpost.com. This approach avoids the overhead of a softmax gate and ensures even distribution by design (since routing is random within a group). The model was trained over 100 days on 512 Ascend AI processors marktechpost.com marktechpost.com. By decoupling experts by domain, they could extract sub-models for specific tasks (e.g. a conversation bot vs. a code generation model) from the larger mixture marktechpost.com. PanGu-Σ reportedly achieved strong results on Chinese benchmarks, outperforming earlier models like Huawei's dense PanGu-13B and Baidu's dense ERNIE-260B in zero-shot evaluations marktechpost.com. This model is a prime example of industry pushing MoE to extreme scale. Although not open-sourced, it demonstrates that a trillion-parameter MoE is feasible in a commercial lab. Notably, Huawei had to innovate on system optimizations (they mention an expert computation and storage separation mechanism for efficiency) to get this to work marktechpost.com, showing how companies are investing in solving MoE's challenges.

- **Alibaba – M6 and Qwen:** Alibaba's **M6** multi-modal MoE (mentioned above) was deployed in 2021 for tasks like product search and design generation, boasting creative abilities (like generating fashion designs) beyond what smaller models could do pandaily.com pandaily.com. By achieving 10T parameters with only 512 GPUs in 10 days pandaily.com, M6 underscored the efficiency of MoE – it reportedly used only 1% of the energy GPT-3 used for a similar scale pandaily.com. More recently, Alibaba released **Qwen-7B and 14B** dense models (2023). There were rumors of an experimental **Qwen-2.5 14B MoE** model – indeed an industry news story involved Huawei alleging that a Huawei MoE showed uncanny similarity to Alibaba's Qwen, implying Qwen might have an MoE version or weights; Huawei denied any copying reuters.com reuters.com. This intrigue aside, it suggests Alibaba likely has MoE research ongoing (perhaps Qwen-MoE for internal use). Alibaba's DAMO Academy has also used MoE in multi-task learning for recommendation and language (their publication of **PLE** is widely used arxiv.org). Given Alibaba's scale (e-commerce, cloud), they would benefit from MoEs to handle myriad user queries and languages efficiently.

- **Microsoft – zCode and Phi:** Microsoft's research has been very active in MoE (DeepSpeed, Tutel, etc.), but what about their products? One clue is **Project Z-Code**, part of Microsoft's Turing initiative for language. In 2021, they announced **Z-Code Mixture of Experts** models for machine translation and representation learning, integrating MoE into some of their Azure Translator models arxiv.org. For example, Microsoft's ZCode-MoE model achieved SOTA on multilingual translation in WMT benchmarks, and they deployed variants in production for certain language pairs. Microsoft also hinted at MoE in their **"Phi-1"** series models – the timeline from the survey shows *Phi-3.5-MoE* in late 2024 arxiv.org. "Hunyuan" is another name: Microsoft's large Chinese model **Hunyuan 13B** is dense, but an internal larger Hunyuan might use MoE (the timeline lists "Hunyuan-Large" and "Phi-3.5-MoE" in the same timeframe arxiv.org). Furthermore, the open WizardLM-2 MoE model was possibly trained with support from Microsoft (the LangDB snippet called it "Microsoft AI's most advanced Wizard model" langdb.ai, suggesting collaboration). All in all, Microsoft's commercial strategy (e.g. Azure OpenAI services) hasn't surfaced MoEs yet, but their research and internal models very likely leverage MoEs to reduce training cost for future Bing or Office copilots, etc. Notably, the **Orca-13B** model that Microsoft researchers used for instruction-following had a teacher model which might have been an MoE – there's speculation some of their unseen large models are MoE-boosted to get more bang for the buck.

- **Others:** Several Chinese research labs and startups have embraced MoE for record-setting models. The Beijing Academy of AI (BAAI) built **WuDao 2.0**, a 1.75 trillion-parameter MoE in 2021, which combined 64 experts (Sparse) with 200 billion dense parameters to reach that scale marktechpost.com. Although details were scarce, WuDao 2.0 was a milestone demonstrating China's capability in training ultra-large MoEs. Another is Inspur's **Yuan 1.0 and 2.0** models – Yuan 1.0 (245B dense) and Yuan 2.0 (1+ trillion, MoE with 32 experts) are referenced in surveys arxiv.org. The timeline in our survey shows "Yuan 2.0-M32" which implies that model used 32 experts, aligning with MoE arxiv.org. These models were likely used in experimental deployments for Chinese language tasks and possibly as backend models for startups. Elon Musk's **xAI** company introduced **Grok-1** in late 2023, which the survey cites as an MoE model arxiv.org – Grok-1 is believed to be an AI assistant model with potentially MoE architecture to allow quick scaling (xAI's GitHub suggests it's open, but details are sparse). **Databricks' DBRX** (mentioned above in open models) is essentially a commercial effort to offer a strong MoE LLM to enterprises, and its performance beating dense models like LLaMA-70B shows the business appeal of MoE reddit.com docs.skypilot.co. It's likely we'll see companies like OpenAI and Anthropic consider MoE or already test it internally: OpenAI's public statements have been mum on MoE, but some experts speculate GPT-4 could be an ensemble or MoE under the hood (though evidence is lacking). Anthropic's Claude models are dense as per their publications, but they have cited Fedus et al.'s MoE work in their discussions of efficient training, so it's not off the table.

In conclusion, many of the *largest AI models to date* have used MoE in one form or another – from Google's and Meta's research prototypes to Chinese trillion-param systems. Commercially, MoEs are on the cusp: they promise economically training enormous models (as shown by Alibaba M6's 1% energy claim pandaily.com and DeepSpeed's 5× cost reduction claim microsoft.com) and serving them in targeted ways. The trend in late 2023–2024 of open MoE models (Mixtral, WizardLM-2, DBRX) indicates that the community is overcoming earlier hurdles, and we can expect more widespread adoption in production. For instance, a future personal assistant AI might route your query to different expert modules (coding, medical, legal, etc.)

internally – an MoE by another name. As one survey put it, “with the convergence of model architecture design in industrial products, system design has emerged as pivotal” arxiv.org – meaning the burden is now on making these MoE systems efficient and robust for real-world use. Given the rapid progress, MoE models are poised to become a staple technique for building advanced AI that is both powerful and efficient.

References and Further Reading

- Jacobs, R. et al. (1991). *Adaptive Mixtures of Local Experts*. **Neural Computation, 3(1)**: 79–87. *(Foundational MoE concept)*
- Jordan, M. & Jacobs, R. (1994). *Hierarchical Mixtures of Experts and the EM Algorithm*. **Neural Computation, 6(2)**: 181–214.
- Shazeer, N. et al. (2017). *Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer*. **arXiv:1701.06538** arxiv.org arxiv.org.
- Lepikhin, D. et al. (2020). *GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding*. **arXiv:2006.16668** arxiv.org arxiv.org.
- Fedus, W. et al. (2022). *Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity*. **J. of Machine Learning Research, 23(120)**: 1–39 arxiv.org.
- Du, N. et al. (2022). *GLaM: Efficient Scaling of Language Models with Mixture-of-Experts*. **ICML 2022** arxiv.org arxiv.org.
- Riquelme, C. et al. (2021). *Scaling Vision with Sparse Mixture of Experts*. **NeurIPS 34**: 8583–8595 arxiv.org arxiv.org.
- Zoph, B. et al. (2022). *ST-MoE: Designing Stable and Transferable Sparse Expert Models*. **arXiv:2202.08906** arxiv.org.
- Mustafa, B. et al. (2022). *LIMoE: Language-Image Mixture of Experts for Efficient Multimodal Fusion*. **NeurIPS 35**: 9564–9576 arxiv.org.
- Artetxe, M. et al. (2022). *No Language Left Behind: Scaling Human-Centered Machine Translation*. **arXiv:2207.04672**. (NLLB 200, includes MoE variant) huggingface.co.
- Xue, F. et al. (2024). *OpenMoE: An Early Effort on Open Mixture-of-Experts Language Models*. **arXiv:2402.01739** arxiv.org.
- Sanseviero, O. et al. (2023). *Mixture of Experts Explained*. **HuggingFace Blog** huggingface.co huggingface.co.
- Microsoft DeepSpeed Team (2022). *DeepSpeed-MoE: Advancing Mixture-of-Experts Inference and Training* (Blog) microsoft.com microsoft.com.

- Microsoft Research (2021). *Tutel: An Efficient MoE Implementation for Large DNNs* (Blog) microsoft.com microsoft.com.
- Mistral AI (2024). *Cheaper, Better, Faster, Stronger: Introducing Mixtral 8×22B* (Blog) mistral.ai mistral.ai.
- Databricks (2024). *Introducing DBRX: A New State-of-the-Art Open LLM* (Blog) reddit.com docs.skypilot.co.
- Huawei Noah's Ark Lab (2023). *PanGu-Σ: A Trillion-Parameter Sparse LLM* (Paper & Press) marktechpost.com marktechpost.com.
- (Additional references are cited inline in text above arxiv.org microsoft.com etc.)

IntuitionLabs - Industry Leadership & Services

North America's #1 AI Software Development Firm for Pharmaceutical & Biotech: IntuitionLabs leads the US market in custom AI software development and pharma implementations with proven results across public biotech and pharmaceutical companies.

Elite Client Portfolio: Trusted by NASDAQ-listed pharmaceutical companies including Scilex Holding Company (SCLX) and leading CROs across North America.

Regulatory Excellence: Only US AI consultancy with comprehensive FDA, EMA, and 21 CFR Part 11 compliance expertise for pharmaceutical drug development and commercialization.

Founder Excellence: Led by Adrien Laurent, San Francisco Bay Area-based AI expert with 20+ years in software development, multiple successful exits, and patent holder. Recognized as one of the top AI experts in the USA.

Custom AI Software Development: Build tailored pharmaceutical AI applications, custom CRMs, chatbots, and ERP systems with advanced analytics and regulatory compliance capabilities.

Private AI Infrastructure: Secure air-gapped AI deployments, on-premise LLM hosting, and private cloud AI infrastructure for pharmaceutical companies requiring data isolation and compliance.

Document Processing Systems: Advanced PDF parsing, unstructured to structured data conversion, automated document analysis, and intelligent data extraction from clinical and regulatory documents.

Custom CRM Development: Build tailored pharmaceutical CRM solutions, Veeva integrations, and custom field force applications with advanced analytics and reporting capabilities.

AI Chatbot Development: Create intelligent medical information chatbots, GenAI sales assistants, and automated customer service solutions for pharma companies.

Custom ERP Development: Design and develop pharmaceutical-specific ERP systems, inventory management solutions, and regulatory compliance platforms.

Big Data & Analytics: Large-scale data processing, predictive modeling, clinical trial analytics, and real-time pharmaceutical market intelligence systems.

Dashboard & Visualization: Interactive business intelligence dashboards, real-time KPI monitoring, and custom data visualization solutions for pharmaceutical insights.

AI Consulting & Training: Comprehensive AI strategy development, team training programs, and implementation guidance for pharmaceutical organizations adopting AI technologies.

Contact founder Adrien Laurent and team at https://intuitionlabs.ai/contact for a consultation.