INTRODUCTION TO NATURAL LANGUAGE PROCESSING

Mixture of Experts (MoE)

Victor Zhong

The Scaling Law Problem

Recall Scaling Laws

Performance scales with parameter count (N) and data (D).

Problem

Increasing N means increasing **FLOPs** (Floating Point Operations) per token during inference.

A 1 Trillion parameter dense model is extremely slow and expensive to run.

Dense vs. Sparse Models

Dense Model

Every parameter is used for every input.

Example: Llama-3-70B. Every token passes through all 70B weights.

Sparse Model

Only a subset of parameters is active for a given input.

Example: Mixtral 8×7B. The model has 47B parameters, but each token only uses 13B.

Conditional Computation

Intuition: Does "The" require the same computational power as "Quantum"? Probably not.

Conditional Computation

We activate different parts of the network based on the complexity or topic of the input token.

This allows us to decouple **model size** (memory) from **compute cost** (FLOPs).

The Promise of MoE

Massive Scale

We can train models with Trillions of parameters.

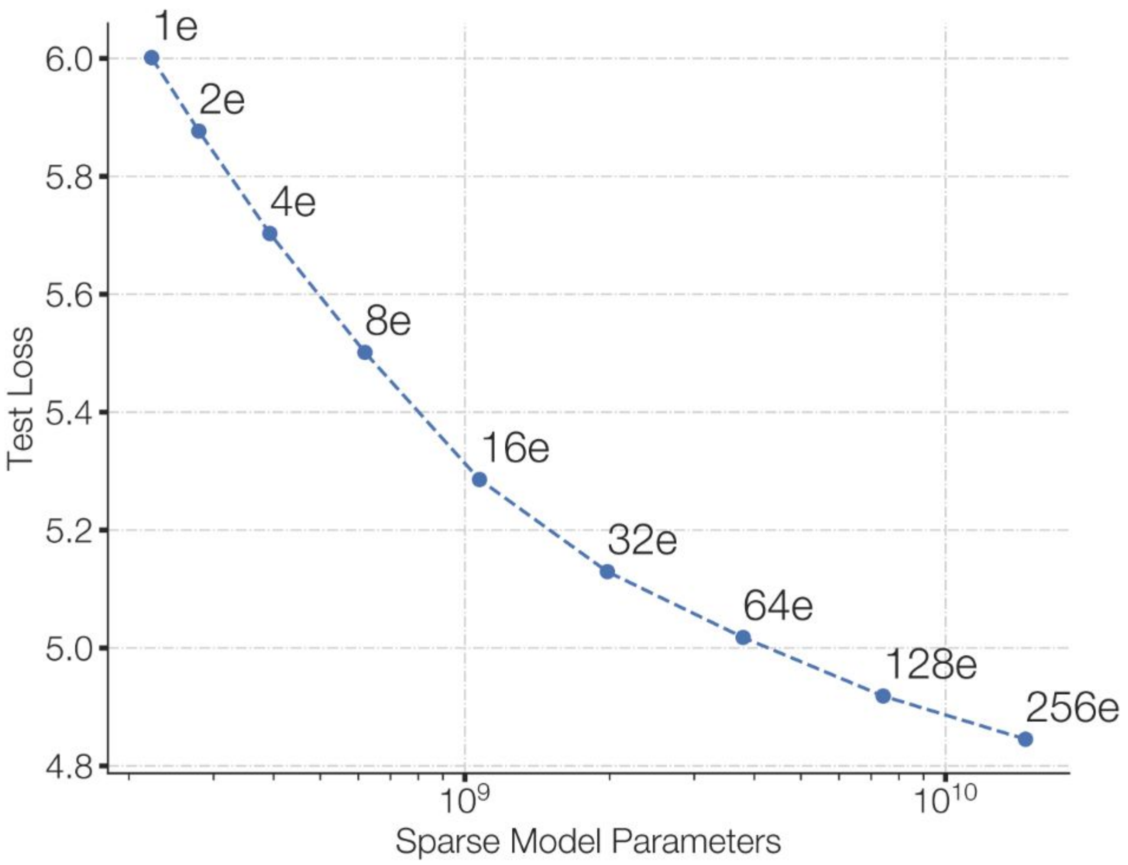
Fast Inference

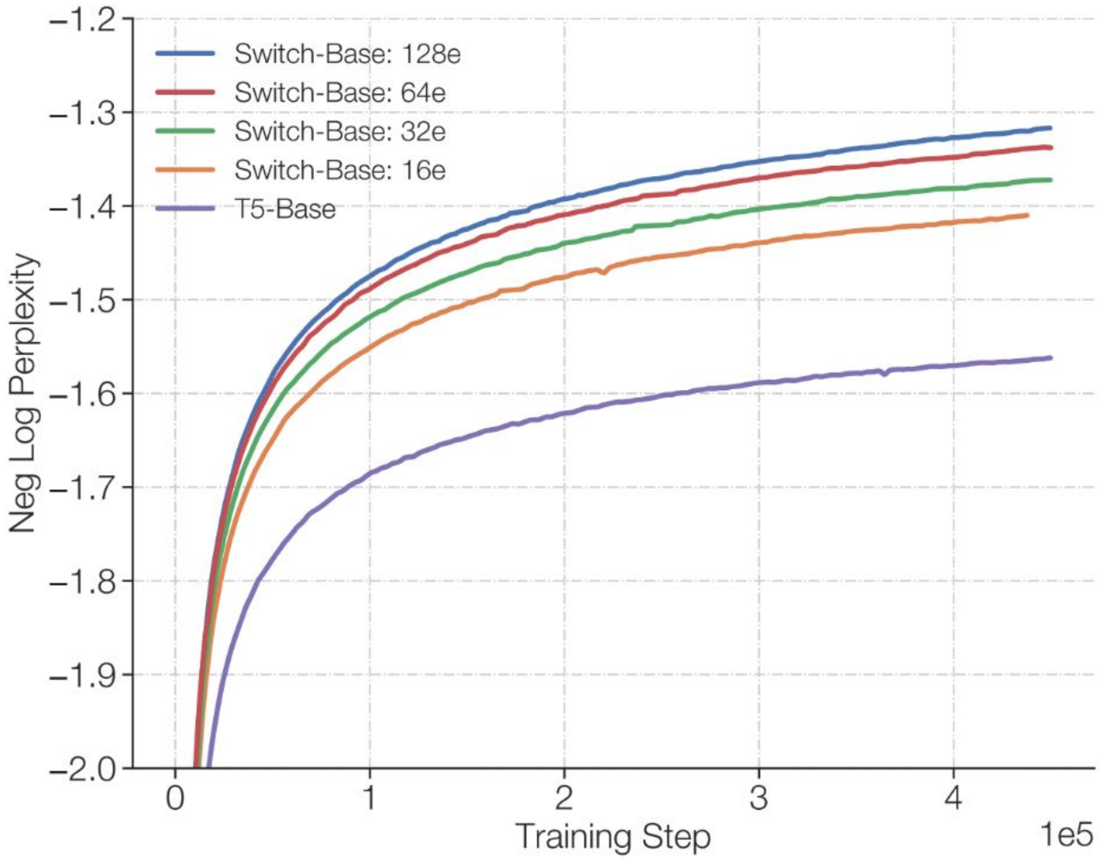
We only pay the compute cost of a much smaller model.

Result

Better performance than a dense model for the same inference budget.

Switch Transformer (Google)





[Switch Transformers. Fedus, 2022](#)

The Transformer Block Recap

Standard Transformer Block:

- **Self-Attention:** Mixes information between tokens.
- **Feed-Forward Network (FFN):** Processes each token independently.

FFN Equation

$$\mathrm{FFN}(\mathbf{x}) = \max(0, xW_1 + b_1)W_2 + b_2$$

where $\mathbf{x}$ is the input, W_1, W_2 are weight matrices, and b_1, b_2 are biases.

This FFN usually contains ~2/3 of the model's parameters.

The MoE Layer

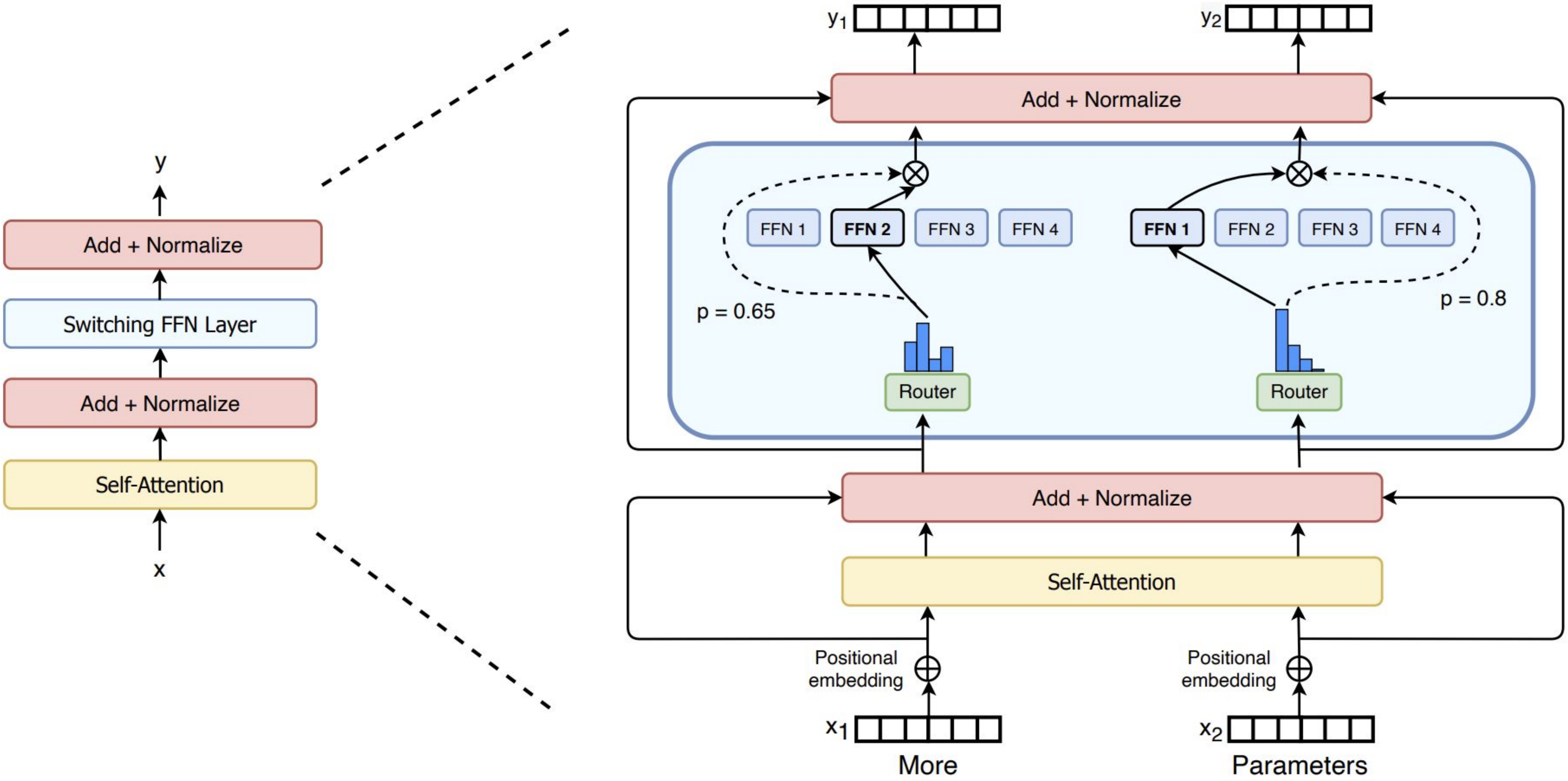
In an MoE Transformer, we replace the single large FFN with:

N Experts

A set of N independent, smaller FFNs.

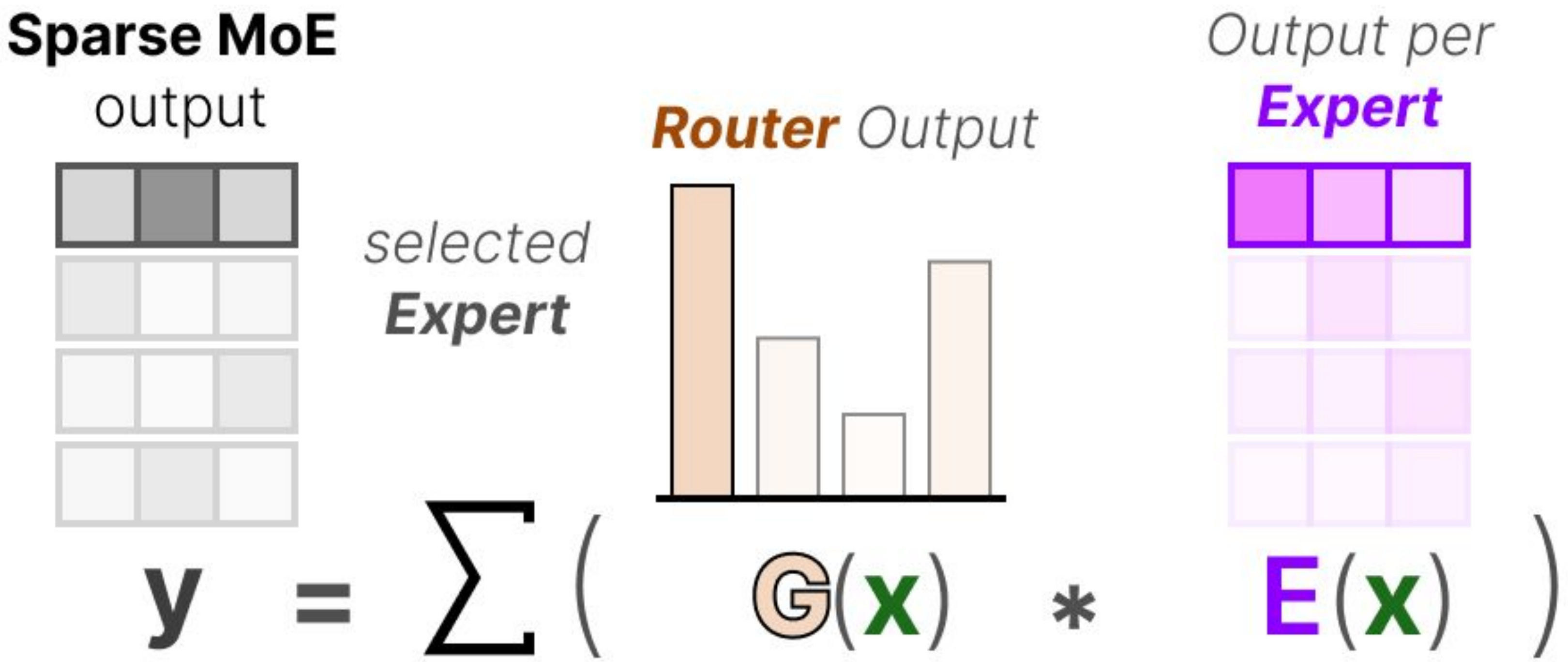
Router

A gating network to decide which expert(s) to send token x to.



[Switch Transformers. Fedus, 2022]

The MoE Equation



- $E_i(x)$: The output of the i-th expert network.
- $G(x)\_i$: The output of the gating network (probability) for expert i.

Typically, $G(x)$ is sparse (most values are 0).

Image from https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-mixture-of-experts

Visualizing MoE

Process

1. Input Token → Router
2. Router selects **Top-K** experts (e.g., K=2).
3. Token is processed only by those 2 experts.
4. Outputs are weighted and summed.

Result: A model with 8 experts has 8x the params, but only 2x the compute.

Token-Level Routing

Crucial Detail: Routing happens at the **Token Level**, not the sentence level.

Input: "The dog barked"

"The"	"dog"	"barked"
↓	↓	↓
Expert 1 & 4	Expert 2 & 3	Expert 1 & 5

This theoretically allows experts to specialize in syntax, entities, or concepts.

How does the Router work?

The Router is usually a simple linear layer followed by a Softmax.

> **Problem:** Softmax is dense (non-zero everywhere). We need sparsity.

Gating

$$\mathbf{H} = \sum^{N} \left(G(\mathbf{x})_i * \mathbf{E}(\mathbf{x})_i \right) + \mathbf{u}_t^l,$$

$$G(\mathbf{x})_i = \mathrm{Softmax}_i \left({\mathbf{u}_t^l}^T \mathbf{e}_i^l \right),$$

Gates selected by a logistic regressor

Top-K Routing

To force sparsity, we use Top-K Gating:

1. Compute router logits for all experts.
2. Keep only the top k values (set others to $-\infty$).
3. Apply Softmax.

This ensures we only compute $E_i(x)$ for k experts.

Gating

$$\mathbf{H} = \sum_{i=1}^{N} \left(\mathrm{G}(\mathbf{x})_i * \mathbf{E}(\mathbf{x})_i \right) + \mathbf{u}_t^l,$$

$$\mathrm{G}(\mathbf{x})_i = \begin{cases} s_{i,t}, & s_{i,t} \in \mathrm{Topk}(\{s_{j,t} | 1 < j < N\}, K), \\ 0, & \text{otherwise}, \end{cases}$$

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

Gates selected by a logistic regressor

Expert Specialization

Does the router actually learn meaningful specializations?

Myth

Experts map cleanly to "Biology Expert" or "French Expert".

Reality

They often specialize in syntax, punctuation, or abstract features.

Exception

In Multilingual MoEs, some experts do tend to specialize in language groups.

MoE Consideration: Expert Capacity

The Bottleneck

In hardware, we process tokens in batches. If all tokens want Expert 1, Expert 1 OOMs (Out Of Memory) while Experts 2-8 sit idle.

You choose the capacity factor and support it using additional buffers.

- 1.0: perfect uniform routing
- >1.0: allowing for some "spillover"

Expert Capacity

The maximum number of tokens an expert can process per batch.

$$\text{expert capacity} = \left(\frac{\text{tokens per batch}}{\text{number of experts}}\right) \times \text{capacity factor}$$

Exceeding Expert Capacity: Token Dropping

The Consequence

If an expert is over capacity, extra tokens are **dropped**.

- Dropped tokens are passed through a residual connection (identity) without processing.
- **Result:** Performance degrades significantly if too many tokens are dropped.

Goal: We need balanced routing.

MoE Consideration: The Load Balancing Problem

A naive router often converges to a **"Winner Take All"** scenario.

It learns to send all tokens to Expert 1 (because Expert 1 learned faster early on).

Result

Effectively a dense model with 1 active expert and $N - 1$ dead experts.

Load Balancing Loss

We add an auxiliary loss function to force uniform distribution.

$$\text{loss} = \alpha \cdot N \cdot \sum_{i=1}^{N} f_i \cdot P_i$$

- f_i : Fraction of tokens assigned to expert i.
- P_i : Average router probability for expert i

$$f_i = \frac{1}{T} \sum_{x \in \mathcal{B}} \mathbb{1}\{\operatorname{argmax} p(x) = i\}$$

$$P_i = \frac{1}{T} \sum_{x \in \mathcal{B}} p_i(x)$$

This penalizes the model if the distribution of routed tokens is uneven.

Router Z-Loss

MoE training is notoriously unstable (fp16 overflows from large exponentials).

Z-Loss

Penalizes large logits in the router.

x: logits going into router

B: number of tokens

$$L_z(x) = \frac{1}{B} \sum_{i=1}^{B} \left(\log \sum_{j=1}^{N} e^{x_j^{(i)}} \right)^2$$

Encourages numerical stability without changing the routing logic significantly.

Stable MoE. Zoph, 2022

Stochastic Routing (Jitter)

Problem

"Dead experts" early in training.

Solution: Jitter

Add **Gaussian noise** to router logits during training.

This encourages exploration: the router tries sending tokens to random experts, allowing them to get gradients and learn.

MoE Consideration: Expert Parallelism Training

How do you fit a 1 Trillion parameter model on GPUs?

Data Parallelism

Copy model to all GPUs. (Impossible for MoE due to size).

Expert Parallelism

- Place Expert 1 on GPU 0.
- Place Expert 2 on GPU 1.

All-to-All Communication required.

Communication Bottlenecks

MoE inference is often **bandwidth bound**, not compute bound.

> We spend a lot of time moving tokens between GPUs.

Optimizations

- **Epistemic Compression:** Compressing hidden states before sending.
- **DeepSpeed-MoE / MegaBlocks:** Specialized kernels to handle variable sequence lengths and sparse matrix multiplication efficiently.

Inference Challenges

VRAM Usage

You need enough RAM to hold all weights (47B for Mixtral), even if you only use a few.

Batched Inference

Efficient if many users query the model at once (high throughput).

Single User

Low latency, but inefficient memory access patterns.

The "Free Lunch"?

Is MoE a free lunch? **No.**

Pros

FLOP efficient, high throughput.

Cons

VRAM hungry, complex to train, unstable, harder to serve (requires specialized kernels).

Tricks: Mixture of Depths

Instead of skipping width (parameters), skip **depth** (layers).

Concept

Not every token needs 96 layers of processing.

Router

Decides whether a token processes the current block or skips it (via residual link).

Result

Faster inference, equivalent performance.

Tricks: Upcycling (Dense-to-MoE)

Can we turn a dense model into an MoE cheaply?

Method

1. Take a trained dense checkpoint.
2. Duplicate the FFN weights N times to create N experts.
3. Initialize a random router.
4. Continue training.

Result: Faster convergence than training MoE from random initialization.

Tricks: MoE on Consumer Hardware

Can you run Mixtral on a laptop?

Quantization

4-bit quantization reduces 47B params to ~24GB VRAM.

Offloading

Store experts in System RAM, load them to VRAM on-demand (slow but feasible).

llama.cpp

Efficient CPU inference for MoEs.

Some Famous MoEs: Mixtral 8×7B (Mistral AI)

Architecture

8 Experts, Top-2 Routing.

Parameters

47B Total, 13B Active (per token).

Significance

The first high-performing open-weights MoE that was easy to run.

Qwen1.5-MoE

Upcycling

Initialized from a dense model (Qwen-1.8B).

Instead of training from scratch, they copy the weights of a trained dense model to initialize the experts.

Significance: Reduces the massive compute cost of pretraining MoEs from scratch.

DeepSeek-V2/V3

Innovation: Fine-Grained Experts + Shared Experts.

Shared Experts

1-2 experts are **always active** for every token.

Captures "common knowledge".

Routed Experts

Many small experts (e.g., 64 or 256), selecting top-6 or top-8.

More granular specialization.

DeepSeek-V3 Details

- **Total Params:** 671 Billion.
- **Active Params:** 37 Billion.
- **Experts:** 256 Routed Experts + 1 Shared Expert.
- **Routing:** Top-8 routed experts.

Result

Performance rivaling GPT-4 and Claude 3.5 Sonnet at a fraction of the training cost ($6M vs $100M+).

Grok-1 (xAI)

Architecture

314 Billion parameters.

Routing

Top-2.

Active

~86 Billion.

A massive, raw MoE model demonstrating the pure scaling power of the architecture.

Comparison Table

Model	Total Params	Active Params	Experts	Routing
Switch	1.6T	~1B	2048	Top-1
Mixtral	47B	13B	8	Top-2
Grok-1	314B	86B	8	Top-2
DeepSeek V3	671B	37B	257	Top-8

Summary (1/2)

- **MoE:** A sparse architecture that decouples model size from compute cost.
- **Mechanism:** Replaces dense FFN with N experts and a Router.
- **Scaling:** Enables Trillion-parameter models with the inference latency of significantly smaller models.

Summary (2/2)

- **Routing:** Top-K (usually K=2) is the standard.
- **Balancing:** Auxiliary losses are required to prevent expert collapse.
- **Trends:** Moving toward fine-grained experts (DeepSeek) and shared experts to improve efficiency and specialization.

Final Thoughts

- MoE is currently the "winning" architecture for training massive models efficiently.
- It represents a shift from "brute force" scaling to "smart" scaling.
- Expect almost all future frontier models (GPT-5, etc.) to be some form of sparse MoE.

Next Frontier

1-Bit MoEs

Combining quantization (BitNet) with Sparsity.

Heterogeneous Experts

Experts with different architectures (some standard FFN, some lookup tables, some logical solvers).

Decentralized MoEs

Experts living on different devices over the internet.

Questions?

Reading

- DeepSeek-V3 Technical Report (2024).
- Mixtral of Experts (Jiang et al., 2024).
- Switch Transformers (Fedus et al., 2022).

Next Topic: History of NLP.

Key Papers (1)

- **Switch Transformer:** Fedus et al. (2022). "Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity".
- **GShard:** Lepikhin et al. (2020). "GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding".

Key Papers (2)

- **Mixtral:** Jiang et al. (2024). "Mixtral of Experts".
- **DeepSeek-V3:** DeepSeek AI (2024). "DeepSeek-V3 Technical Report".
- **Megablocks:** Gale et al. (2023). "MegaBlocks: Efficient Sparse Training with Mixture-of-Experts".

Deep Dive: DeepSeek Router Equation

Uses a specific bias term for load balancing instead of a complex auxiliary loss in V3 (Auxiliary-Loss-Free Balancing).

Deep Dive: Expert Segmentation

Standard MoE

8 Big Experts.

DeepSeek MoE

64 Small Experts.

Analogy

Instead of hiring 1 Physicist, hire a specialist in Thermodynamics, one in Optics, one in Quantum... and combine them as needed.

Implementation: MegaBlocks

Problem

If Expert A gets 10 tokens and Expert B gets 100, standard matrix multiplication is inefficient (padding).

MegaBlocks Solution

Drops padding entirely. Uses **Block-Sparse Matrix Multiplication** (Block-CSR format).

Result: Significant speedup in training variable-load MoEs.