Mixture of Experts

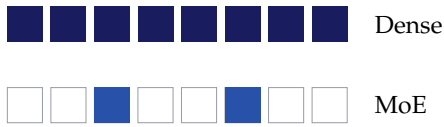
How it works, what's proven, and where it's spreading

Presented by Sourav · Foundation Models Seminar

Following Mu & Lin (2025), *A Comprehensive Survey of Mixture-of-Experts*

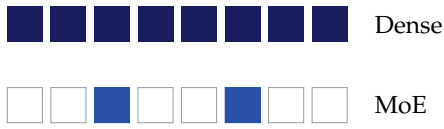
Why Mixture of Experts?

- A dense model runs every parameter on every token, so compute grows with size.
- MoE keeps many parameters but activates only a relevant subset per input.
- So it decouples capacity (total params) from cost (active params) [Shazeer '17].



Why Mixture of Experts?

- A dense model runs every parameter on every token, so compute grows with size.
- MoE keeps many parameters but activates only a relevant subset per input.
- So it decouples capacity (total params) from cost (active params) [Shazeer '17].



This talk follows the survey's arc:

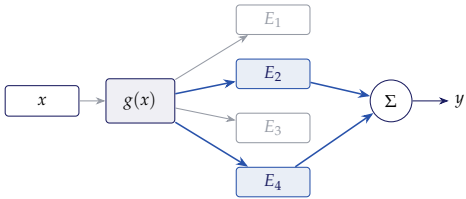
how it works → what's proven → where it's spreading.

The MoE layer: conditional computation

An FFN block becomes N expert FFNs + a router g [Shazeer '17]:

$$g(x) = \text{softmax}\big(\text{TopK}(x^\top W_g + \varepsilon,\, k)\big)$$

$$y = \sum_{i \in \text{TopK}} g(x)_i \, E_i(x)$$

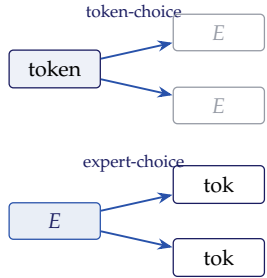


top-k active experts shown in blue

- Only k of N experts fire per token, so compute $\propto$ active params while capacity $\propto$ total (Mixtral: 13B active, 47B total).
- The noise $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ keeps the router exploring experts.

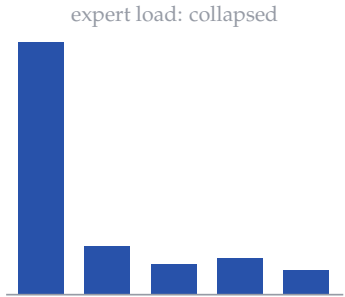
Gating & routing: the knobs

- How many per token: top-1 (Switch [Fedus '22]) vs. top-2 (GShard [Lepikhin '21], Mixtral [Jiang '24]).
- Who chooses: *token-choice* (tokens pick experts) vs. *expert-choice* (experts pick tokens ⇒ balance for free) [Zhou '22].
- Capacity factor C: a cap on tokens per expert; overflow is *dropped* [Fedus '22].



The crux, part 1: expert collapse

- Routers over-use a few experts, a rich-get-richer loop [Shazeer '17].
- Collapse: most experts starve, so the extra capacity is wasted.
- Unbalanced load also wrecks efficiency once experts live on different devices.



The crux, part 2: how to balance the load

- Auxiliary loss (Switch): add $\mathcal{L}_{\text{aux}} = \alpha\, N \sum_{i=1}^{N} f_i\, P_i$ to the task loss.

  f_i = fraction of tokens routed to expert i; P_i = mean router probability for i. Both equal $1/N$ at perfect balance, which minimises the sum [Fedus '22].

The crux, part 2: how to balance the load

- Auxiliary loss (Switch): add $\mathcal{L}_{\text{aux}} = \alpha\, N \sum_{i=1}^{N} f_i\, P_i$ to the task loss.
  f_i = fraction of tokens routed to expert i; P_i = mean router probability for i. Both equal $1/N$ at perfect balance, which minimises the sum [Fedus '22].

- Router z-loss (ST-MoE): $\mathcal{L}_z = \frac{1}{B} \sum_b \left(\log \sum_j e^{x_j^{(b)}} \right)^2$ stops the router logits from blowing up [Zoph '22].

The crux, part 2: how to balance the load

- Auxiliary loss (Switch): add $\mathcal{L}_{\text{aux}} = \alpha\, N \sum_{i=1}^{N} f_i\, P_i$ to the task loss.
  f_i = fraction of tokens routed to expert i; P_i = mean router probability for i. Both equal $1/N$ at perfect balance, which minimises the sum [Fedus '22].
- Router z-loss (ST-MoE): $\mathcal{L}_z = \frac{1}{B} \sum_b \left(\log \sum_j e^{x_j^{(b)}} \right)^2$ stops the router logits from blowing up [Zoph '22].
- Auxiliary-loss-free (DeepSeek): route on $s_i + b_i$: raise the bias b_i for under-used experts, lower it for over-used ones. Balances load with *no* interfering gradient [Wang '24].

All three regularise the routing distribution toward balance, so the task loss stays on the task.

PART II

What's proven

We've built the machinery: the layer, routing, load balancing.

Now: does routing actually create useful specialization, or could the mixture just *collapse* into one expert?

What's proven, part 1: the hard case

- The question: is "divide and conquer" *real*, or does the mixture collapse to one expert?
- Setup: cluster-structured data: sub-populations, each needing different features.



each cluster needs different features

What's proven, part 1: the hard case

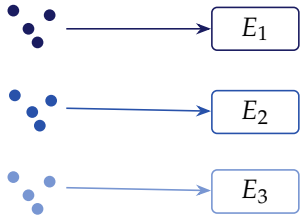
- The question: is "divide and conquer" *real*, or does the mixture collapse to one expert?
- Setup: cluster-structured data: sub-populations, each needing different features.
- Result: a single expert (even a nonlinear 2-layer CNN) *provably cannot* solve it.



each cluster needs different features

What's proven, part 2: MoE solves it

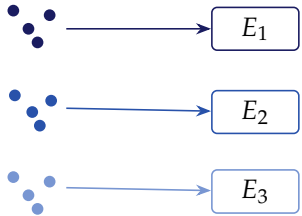
- An MoE of the same small experts *can* solve it, trained by plain gradient descent [Chen '22].
- Why: the router learns the *cluster-center* directions and routes each cluster to its own expert $\Rightarrow$ easy sub-problems.



router routes by cluster

What's proven, part 2: MoE solves it

- An MoE of the same small experts *can* solve it, trained by plain gradient descent [Chen '22].
- Why: the router learns the *cluster-center* directions and routes each cluster to its own expert $\Rightarrow$ easy sub-problems.
- So specialization *and* non-collapse are provable, given cluster structure + nonlinear experts.



router routes by cluster

PART III

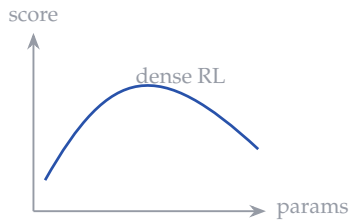
Where it's spreading

The specialization we just proved isn't only useful for language; the same idea is starting to reshape other fields.

Where it's spreading, part 1: the RL puzzle

Obando-Ceron et al., *MoEs Unlock Parameter Scaling for Deep RL*, ICML 2024

- In supervised learning, bigger = better (scaling laws) [Kaplan '20].
- In deep RL, adding parameters often hurts the agent [Obando-Ceron '24].

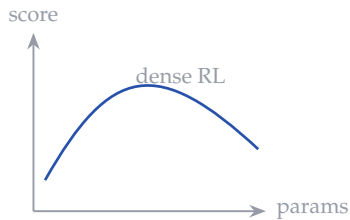


naive scaling turns *down*

Where it's spreading, part 1: the RL puzzle

Obando-Ceron et al., *MoEs Unlock Parameter Scaling for Deep RL*, ICML 2024

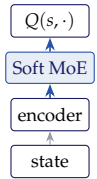
- In supervised learning, bigger = better (scaling laws) [Kaplan '20].
- In deep RL, adding parameters often hurts the agent [Obando-Ceron '24].
- Why: *non-stationary* bootstrapped targets and *plasticity loss*, so big nets ossify [Nikishin '22; Lyle '22].



naive scaling turns *down*

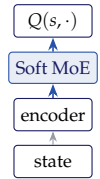
Where it's spreading, part 2: MoE fixes it

- Insert Soft-MoE modules [Puigcerver '24] into the *value network*.
- Fully differentiable (no hard top-k, no token dropping), which suits unstable, low-data RL.



Where it's spreading, part 2: MoE fixes it

- Insert Soft-MoE modules [Puigcerver '24] into the *value network*.
- Fully differentiable (no hard top-k, no token dropping), which suits unstable, low-data RL.
- Result: performance scales with parameter count and with the number of experts (Atari, many regimes) [Obando-Ceron '24].



Open problems & future directions

- Theory beyond toy settings: real deep MoE is still largely unexplained.
- Load balancing that doesn't trade away task quality.
- Interpretability: what do experts actually specialise in?
- Serving cost: memory for all experts, plus routing/communication at inference.
- When does MoE win over a dense model of equal compute? Not always.

Takeaways

- MoE = conditional computation: capacity (total params) decoupled from cost (active params).
- One mechanism: route $\rightarrow$ pick top-k experts $\rightarrow$ combine.
- Load balancing is the crux: read it as regularising the router.
- Specialization is provable (Chen '22) and pays off beyond language, even unlocking scaling in RL (Obando-Ceron '24).
- Still open: theory at scale, and the cost of serving many experts.

References

- Shazeer et al. Outrageously Large Neural Networks: The Sparsely-Gated MoE Layer. *ICLR* 2017.
- Lepikhin et al. GShard: Scaling Giant Models with Conditional Computation. *ICLR* 2021.
- Fedus, Zoph, Shazeer. Switch Transformers. *JMLR* 2022.
- Zhou et al. Mixture-of-Experts with Expert Choice Routing. *NeurIPS* 2022.
- Zoph et al. ST-MoE: Designing Stable and Transferable Sparse Expert Models. 2022.
- Chen et al. Towards Understanding the Mixture-of-Experts Layer in Deep Learning. *NeurIPS* 2022.
- Puigcerver et al. From Sparse to Soft Mixtures of Experts. *ICLR* 2024.
- Jiang et al. Mixtral of Experts. 2024.
- Obando-Ceron et al. Mixtures of Experts Unlock Parameter Scaling for Deep RL. *ICML* 2024.
- Wang et al. Auxiliary-Loss-Free Load Balancing Strategy for MoE (DeepSeek). 2024.
- Mu & Lin. A Comprehensive Survey of Mixture-of-Experts. 2025.