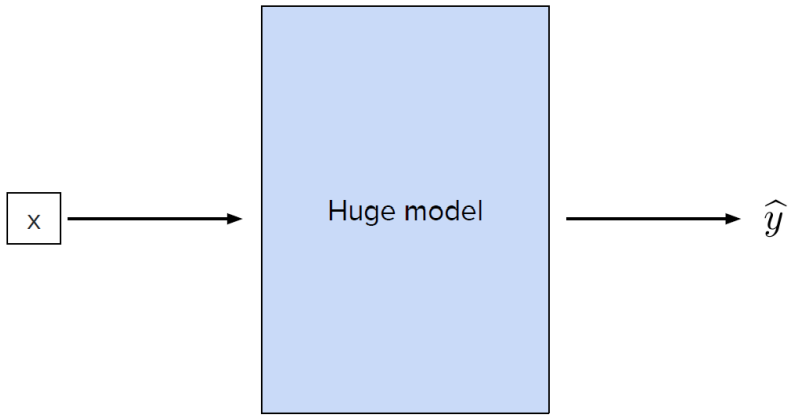
Introduction to Large Language Models

Spring 2026

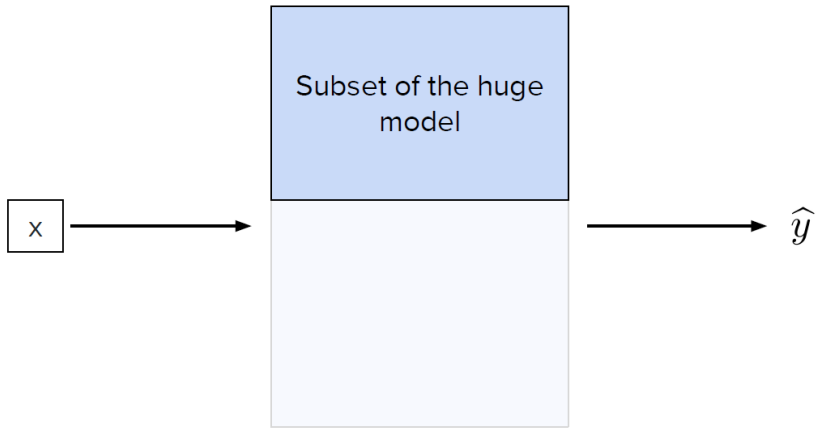
Transformer-Based Architectures
Mixture of Experts

(Some slides adapted from Ralph Grishman at NYU,
Yejin Choi at UWashington, N. Tomura at UDepaul, Jurafsky and Martin,
CS224N, CS224d, CME295 at Stanford and other resourses on the web)

Huge LLN Model



Idea. Not all weights are useful in the forward pass

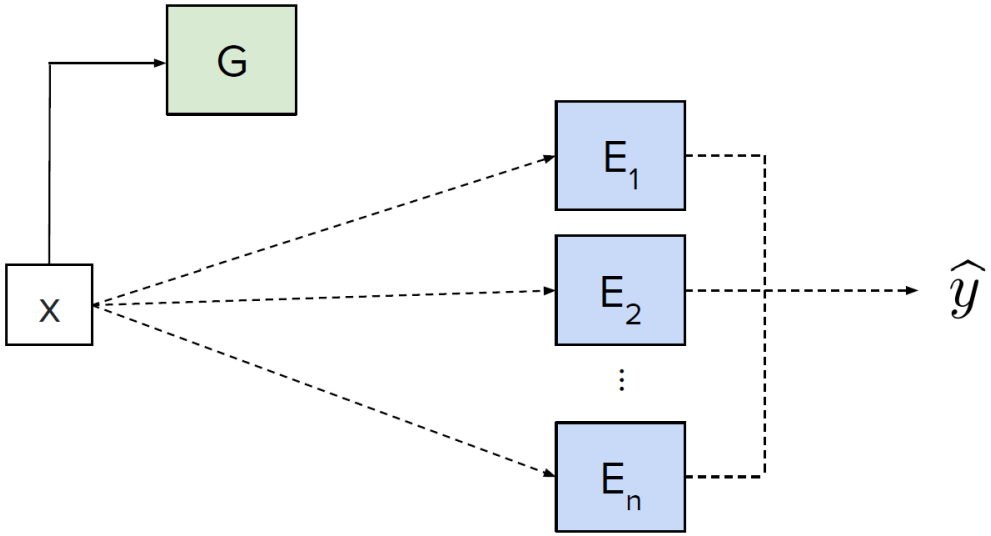


Source: CME295

- In a huge dense model, *all parameters* are used.
  - Modern LLMs are *extremely large* (tens or hundreds of billions of parameters).
  - During inference (the forward pass), **all parameters** are used for *every single input token*.
  - This is computationally expensive and wastes resources because **not all parts of the model are always needed**.
- **Key idea:** Full dense models apply all weights to every token, even when unnecessary. For any specific input, **only certain parts of the model actually contribute** meaningfully.
- **Mixture of Experts (MoE):**
  If we could selectively use **only the relevant subset** of parameters per input, we could:
  - reduce compute
  - improve efficiency
  - increase model capacity without proportional compute cost
- This idea leads directly to MoE.

Mixture of Experts (MoEs)

- A **Mixture of Experts (MoE)** model is a neural architecture where a **gating network selects only a few specialized expert subnetworks** to process each input, giving **huge capacity at low computational cost**.

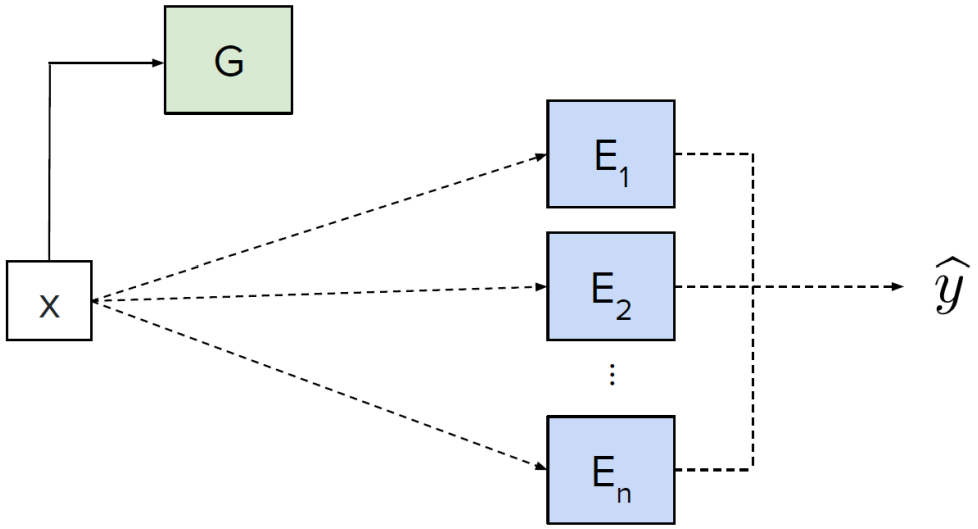


- Instead of one huge monolithic network, imagine splitting the model into many **experts**.
- Each expert specializes in different patterns:
  - Expert 1: reasoning
  - Expert 2: math
  - Expert 3: translation
  - etc.

- We need a way to **decide which experts to use for each input**.
- This decision is made by a **gating network**, also called a **router**.
- The gate takes input x and outputs a set of scores telling each expert how relevant it is for this token.

Source: CME295

Mixture of Experts (MoEs)



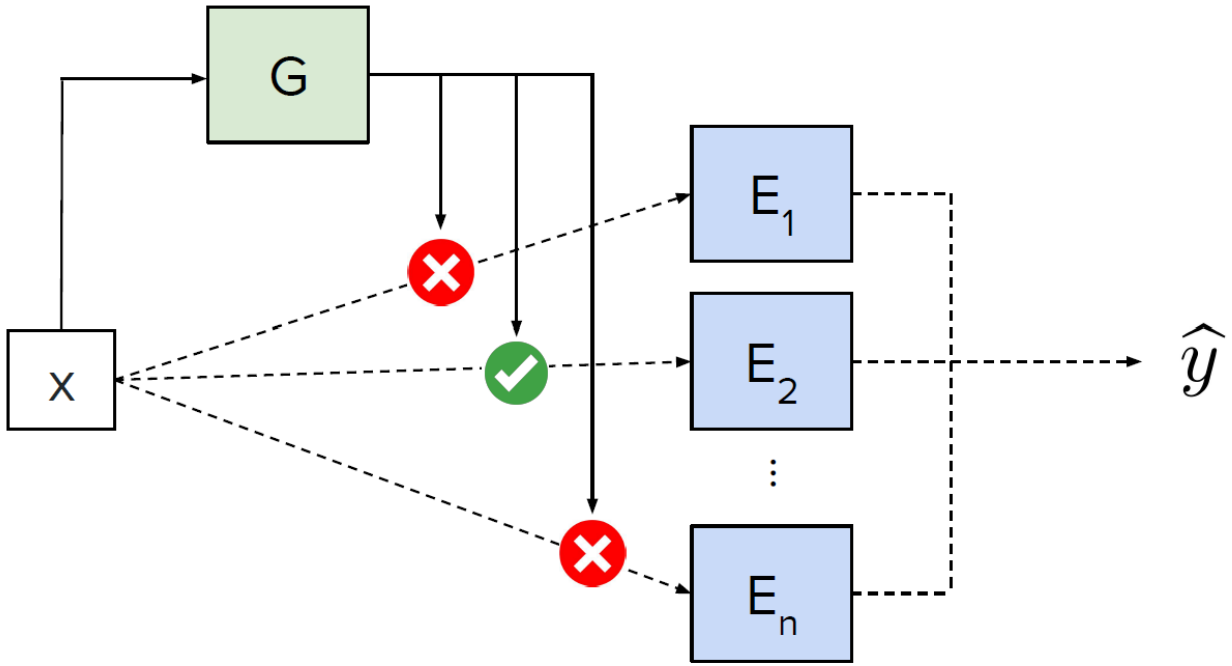
An MoE layer contains:

- A **gating network** $G(x)$
- A set of **experts** $E_1, E_2, \ldots, E_n$

For each token embedding x:

1. Gate computes a vector of scores
$$G(x) \in \mathbb{R}^n$$
2. We pick experts
3. Only those experts compute forward passes
4. Their outputs are combined (weighted sum)

Sparse MoEs



- Only a few experts selected (others crossed out).

$$\hat{y} = \sum_{i \in \mathcal{I}_k} G(x)_i \, E_i(x)$$

Where $\mathcal{I}_k$ is the top-k selected experts.

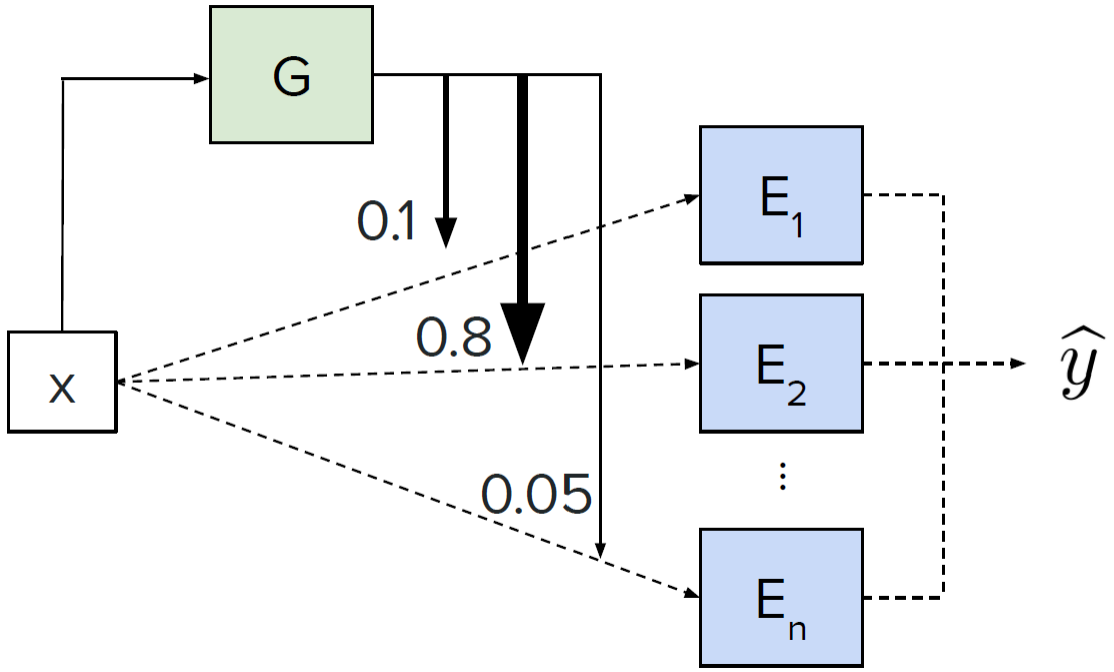
Meaning:

- Sparse MoE evaluates only the *selected* experts.
- Gating still assigns weights, but only to the chosen ones.
- Compute cost scales with **k**, not **n** (total experts).

Sparse MoEs = practical, scalable solution used in modern LLMs (like Mixtral, DeepSeek, GLaM).

Source: CME295

Dense MoE



- Weights like 0.1, 0.2, 0.05 applied to each expert
- all experts contribute

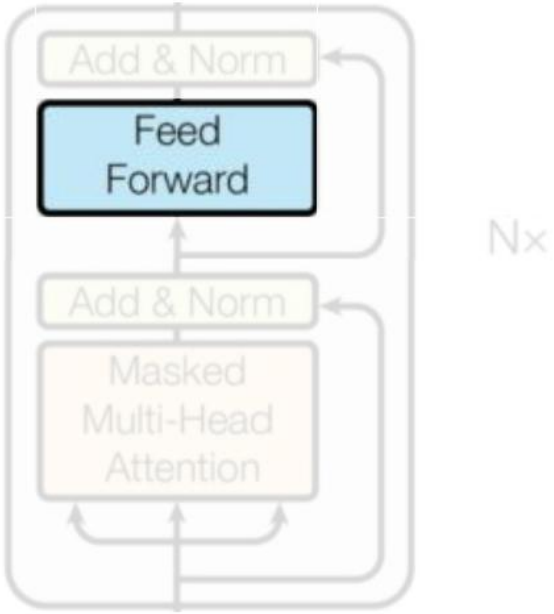
$$\hat{y} = \sum_{i=1}^{n} G(x)_i \, E_i(x)$$

Meaning:

- Dense MoE uses *all* experts every time.
- Each expert output $E_i(x)$is weighted by the gating score $G(x)_i$.
- Normally, this sum is over *all experts*.
- This is the **dense** version of MoE.
- Very expensive — rarely used in LLMs.

Source: CME295

FFN replacement

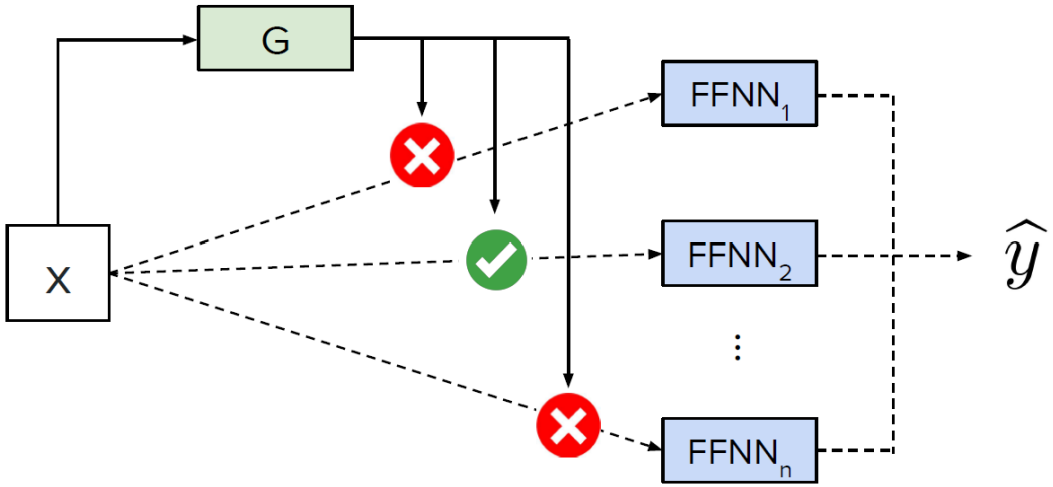


- A Transformer block includes
  - `Attention`
  - `Feed Forward`

  Layers.

- In MoEs, Feed Forward replaced with `MoE Layer`
  - MoE is inserted into the **Feed-Forward Network (FFN)** part of a Transformer layer.
  - Instead of a single FFN, you have many experts.
  - Attention remains dense;
  - only the FFN becomes sparse.

FFN layers



- Each token has its own routing decision.

  ```
  x_token → Gate → top-k experts → output
  ```

- The gating network computes routing decisions **per token**, not per sequence.
- Each token in a sequence may choose different experts.
- This specialization leads to:
  - better performance
  - emergent expert specialization
  - scalable capacity

Source: CME295

MoE Training

MoE training aims to train a model that has:

- **many experts** (sub-networks)
- but **only a few experts are used per token**

This lets the model:

- have **huge capacity** (100B–1T+ parameters)
- while keeping **compute cost low**

Training MoEs is more complex than training dense models because it involves:

1. Training the **experts**
2. Training the **gating network**
3. Ensuring **balanced utilization**
4. Managing **routing**, **load**, and **communication**

How MoE Training Works

Forward Pass

1. Gate scores each expert
2. Top-k experts are selected
3. Tokens are dispatched to those experts
4. Expert outputs are combined to form the final output

Backward Pass

1. Only selected experts receive gradients
2. Gate receives gradient from routing decisions
3. Auxiliary loss ensures balanced expert usage

Expert/Routing collapse

- **Expert/Routing collapse** is a common failure mode in **Mixture-of-Experts (MoE)** models where **only a few experts get used frequently**, while the remaining experts receive **little or no routing**, and therefore **do not learn anything useful**.
- This breaks the core idea of MoE—that different experts should specialize and share the workload.
- If collapse occurs:
  - You lose specialization
  - You effectively reduce the number of experts
  - Model capacity shrinks back to a small dense network
  - Training efficiency and performance degrade
  - Model becomes unstable
  - Load across GPUs becomes imbalanced

Fixing Expert/Routing Collapse: Auxiliary Loss

- Force other experts to be "part of the game" too via auxiliary loss:

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

Where:

- f_i = fraction of tokens routed to expert i
  - Measures **how many tokens** each expert is actually processing
- P_i = average gating probability for expert i
  - Measures **how strongly** the gate wants to pick each expert

The model is penalized if:

- A small number of experts receive almost all the tokens
- Or if the gate consistently assigns higher probabilities to only a few experts

Fixing Expert/Routing Collapse: Auxiliary Loss

Problem

Routing collapse: Router always picks same expert → others unused → wasted capacity.

Solution

Auxiliary load-balancing loss:

- Penalizes uneven usage
- Encourages the gate to distribute tokens across experts
- Prevents collapse

Result

Experts begin to specialize:

- Visualizable via routing patterns
- Leads to improved performance and efficient scaling

Summary

Mixture of Experts (MoE) is motivated by:

1. **Dense models are huge and expensive.**
2. **Not all parameters are needed for every token.**
3. **Split model into many experts.**
4. **Use a gating network to select the right experts.**
5. **Use sparse activation (top-k)** to reduce compute.
6. **Insert MoE into FFN layers of Transformers.**
7. **Route each token independently.**

Result:

- Massive model capacity
- Low compute cost
- High specialization
- State-of-the-art performance (used by Mixtral, DeepSeek-V3, Google's GLaM)

MoE Design:

MoE design is more complex than dense LLM architecture because MoEs must dynamically route tokens, balance experts, manage cross-device communication, and prevent expert collapse.