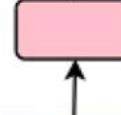


Mixture of Experts (MoE)

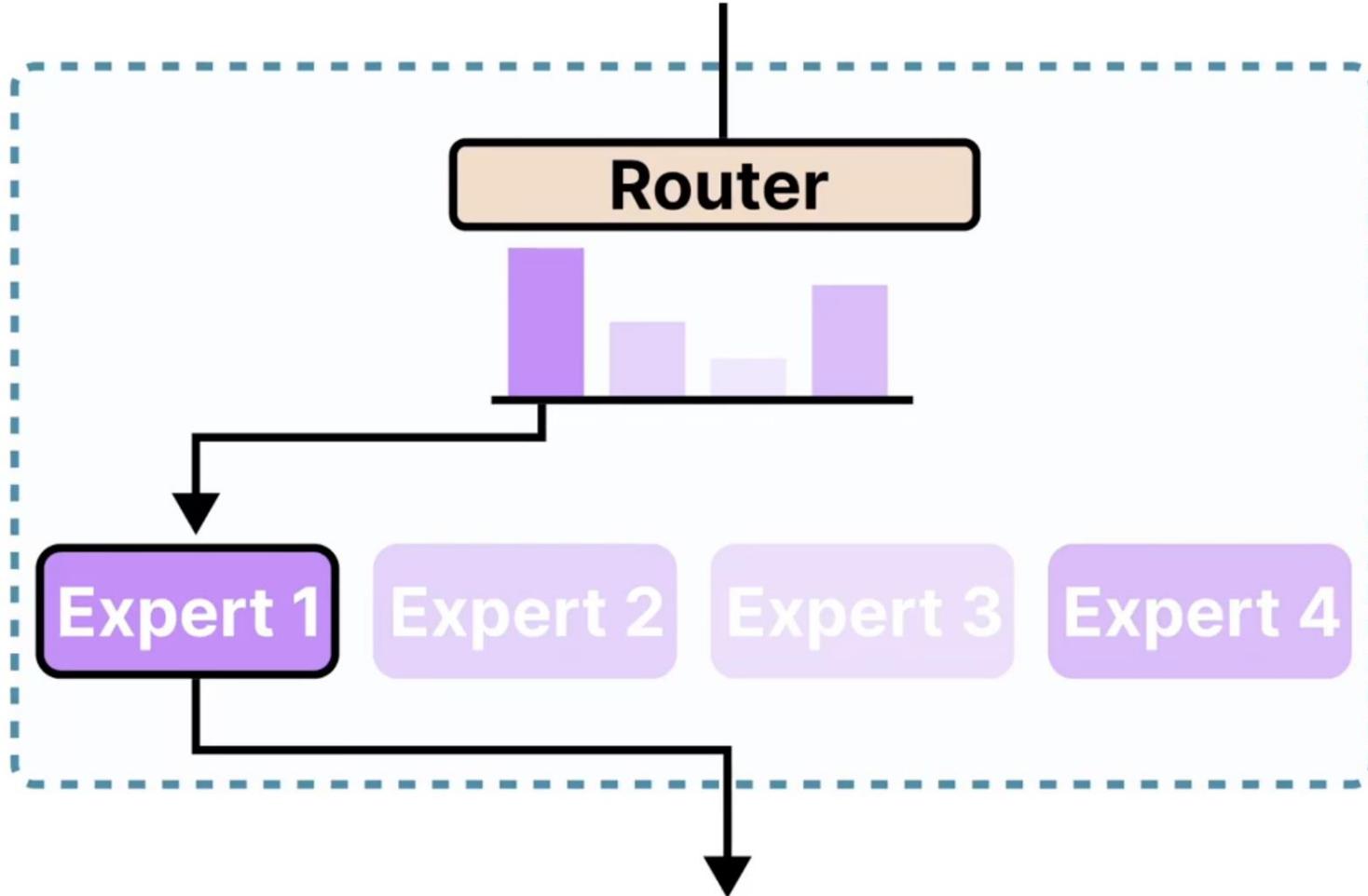


ZOMI



MoE 核心原理可视化

<https://www.youtube.com/watch?v=sOPDGQjFcum>



Contents

1. Introduction: MOE 基本原理
2. The Router: 路由原理
3. Architecture: 模型结构
4. Load Balancing: 均衡负载
5. Keep Top-K: 专家选择
6. Auxiliary Loss: 辅助损失
7. Expert Capacity: 专家容量



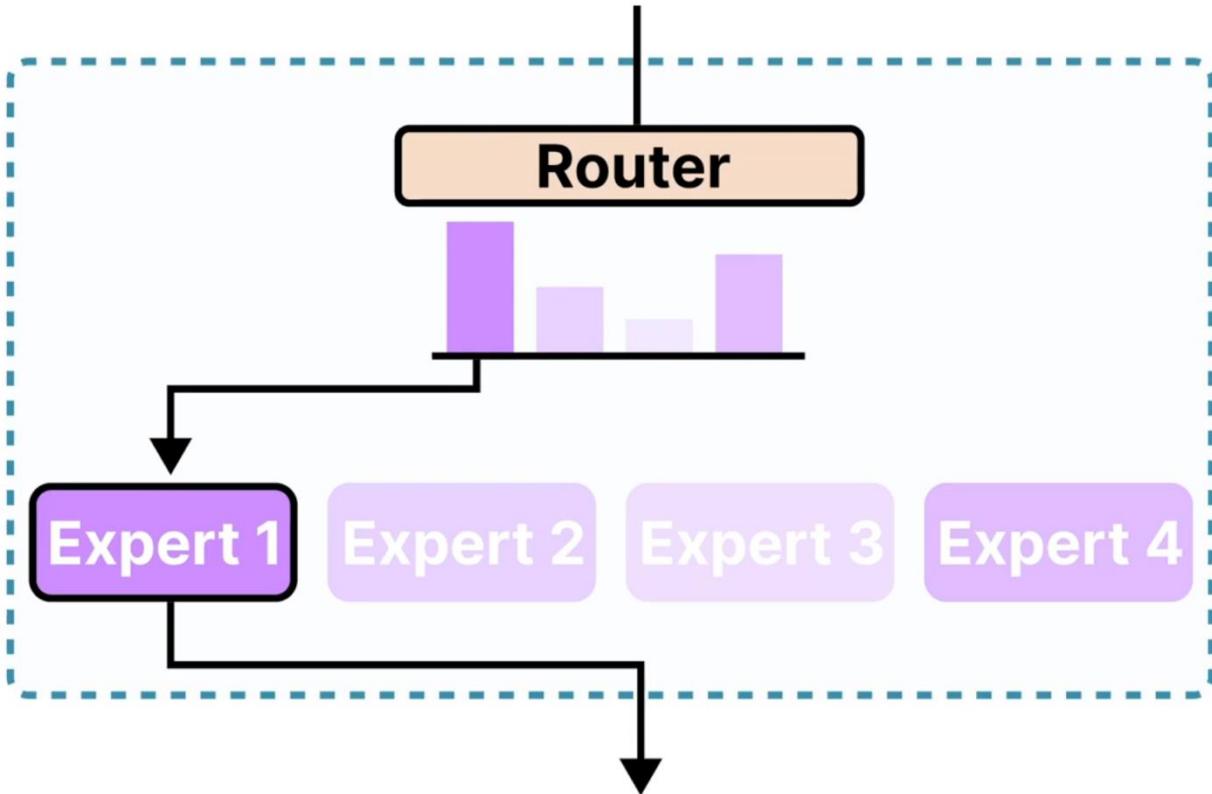
视频目录大纲



01

Introduction: MOE 基本原理

Introduction: MOE 基本原理

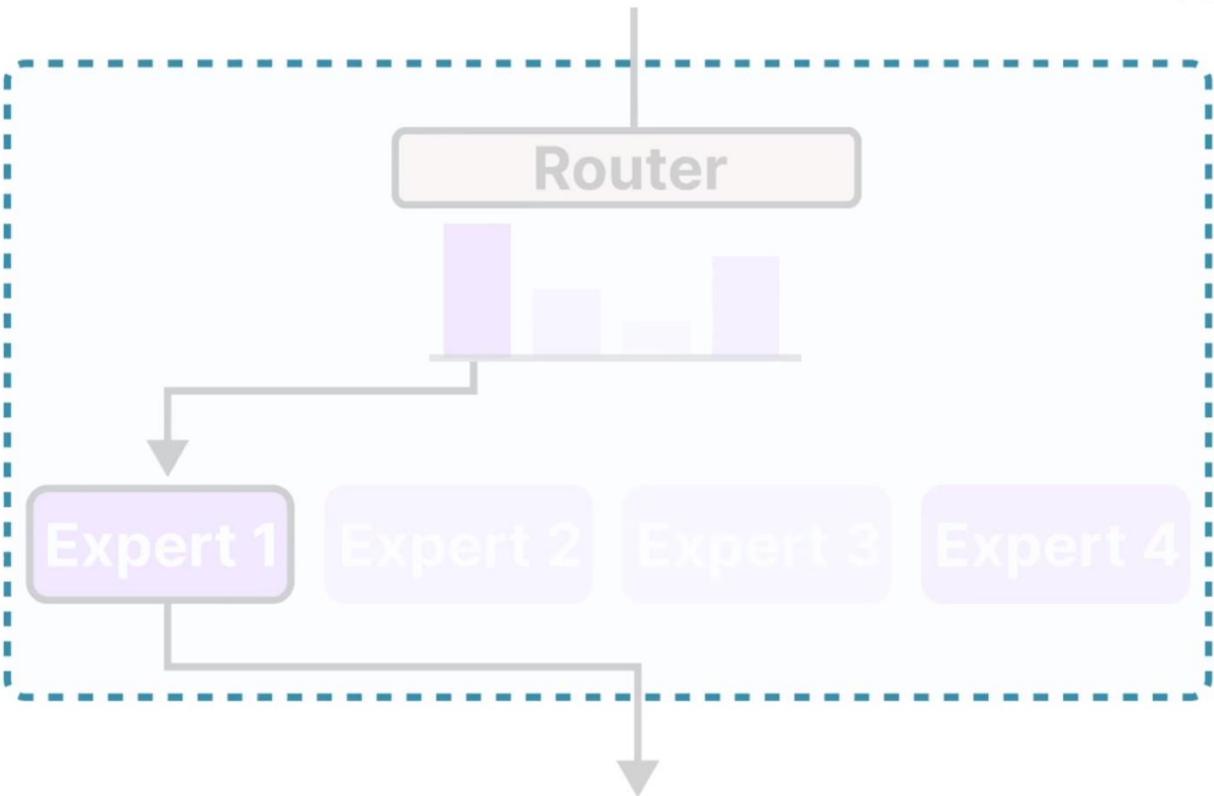


Mixture of Experts (MoE) is a technique that uses different sub-models (“**experts**”) to improve the quality of **LLMs**.



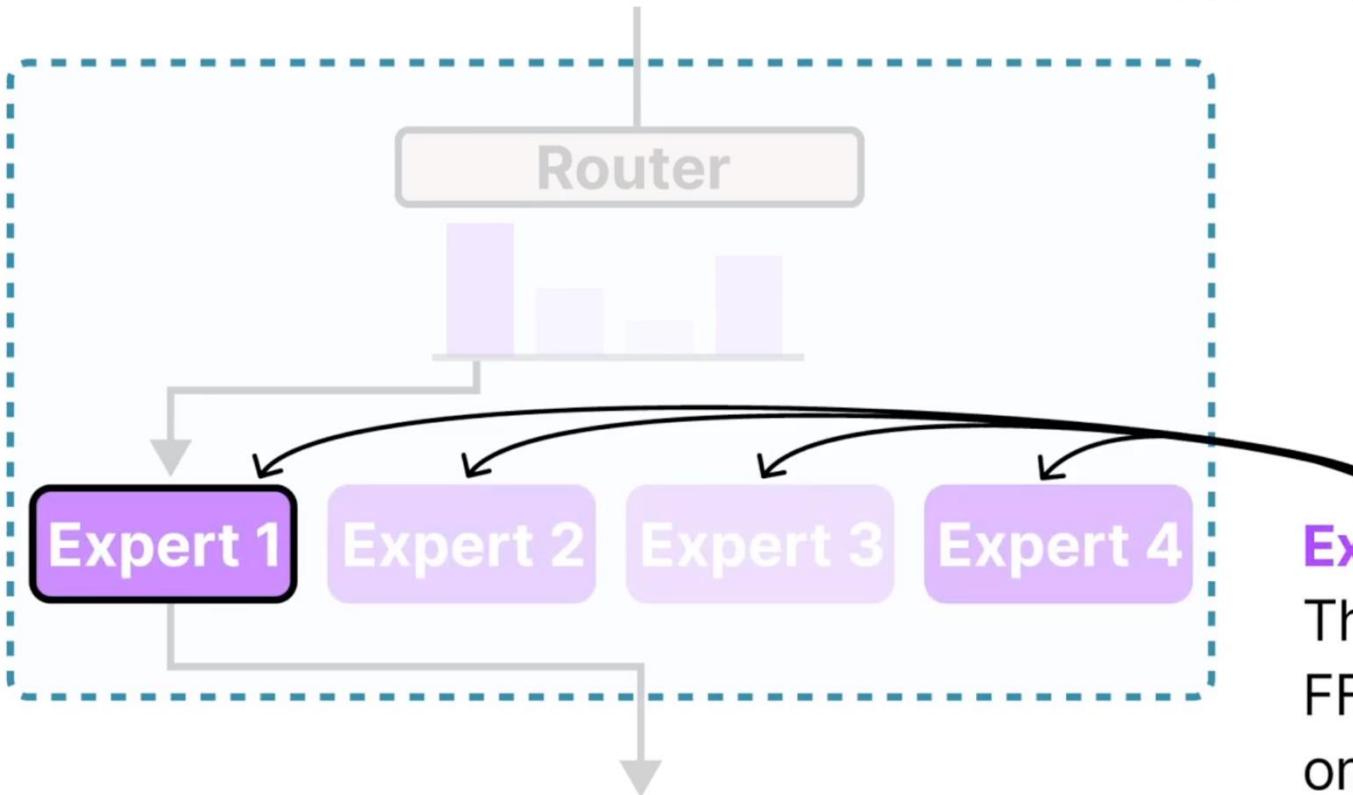
Introduction: MOE 基本原理

Two main components define a **MoE**:



Introduction: MOE 基本原理

Two main components define a **MoE**:

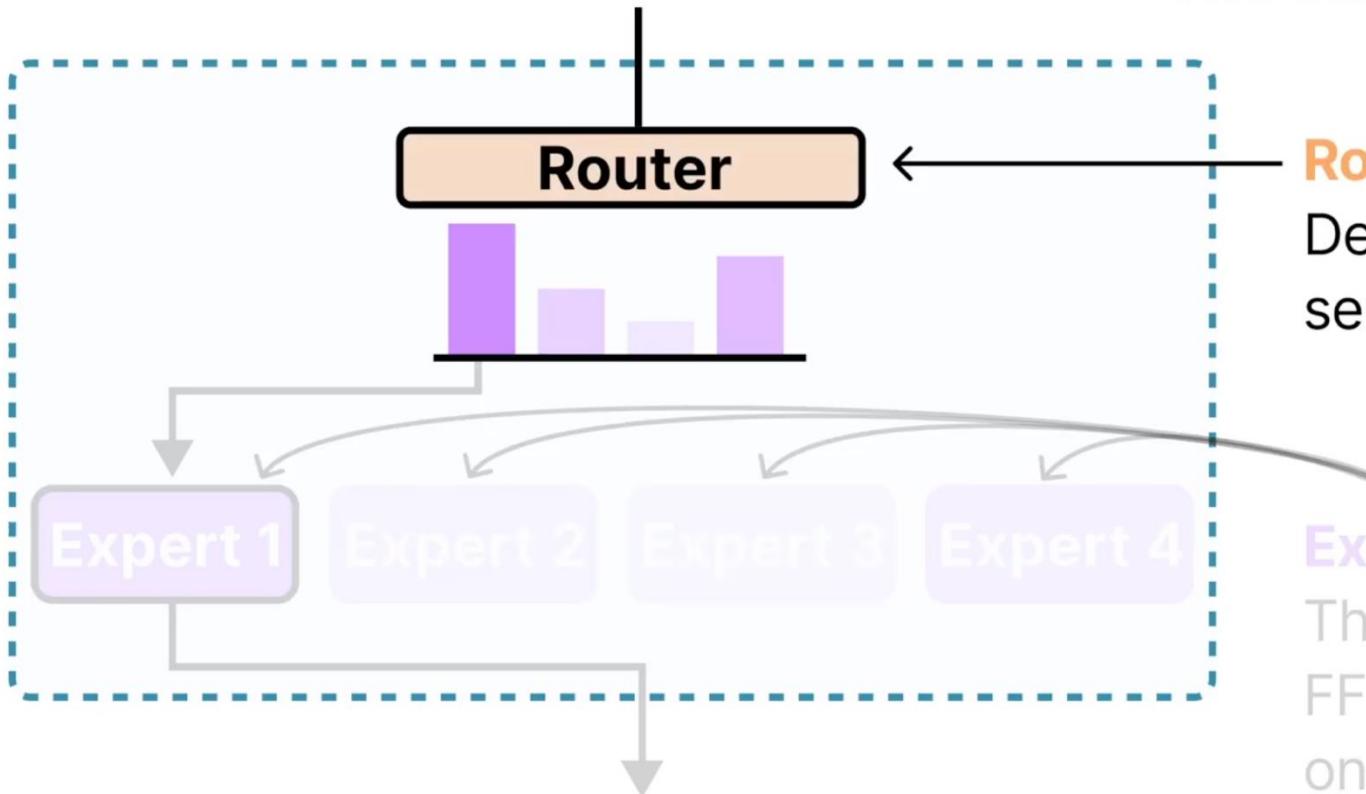


Experts

These “experts” are FFNNs and at least one can be activated



Introduction: MOE 基本原理



Two main components define a **MoE**:

Router (gate network)

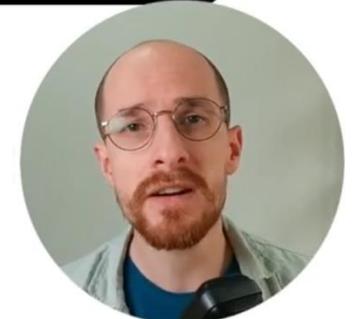
Determines which **tokens** are sent to which experts.

Experts

These “experts” are FFNNs and at least one can be activated



Introduction: MOE 基本原理

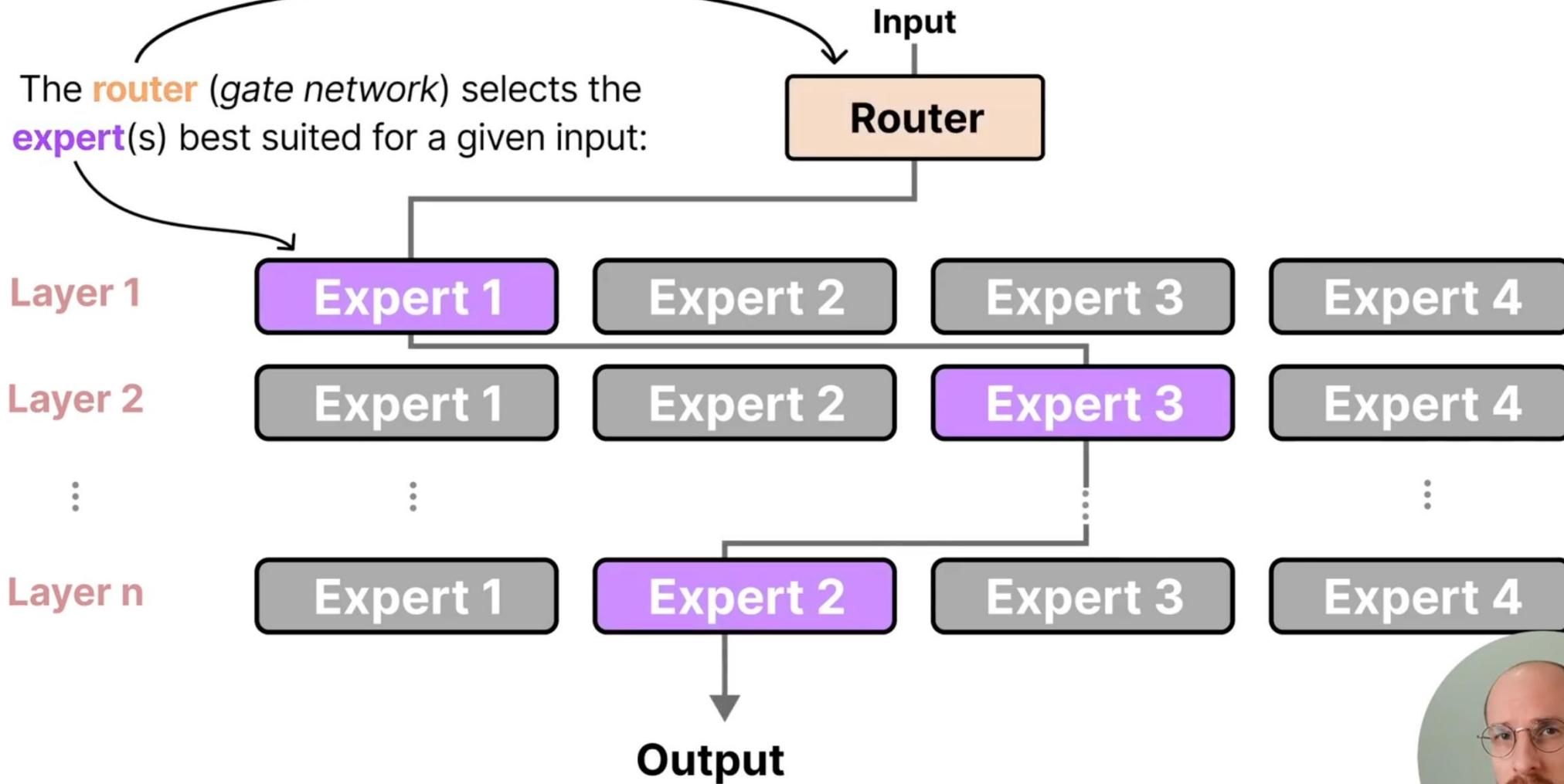


Introduction: MOE 基本原理

Know that an “**expert**” is not specialized in a specific domain like “**Psychology**” or “**Biology**”. At most, it learns **syntactic information** on a **token** level instead.

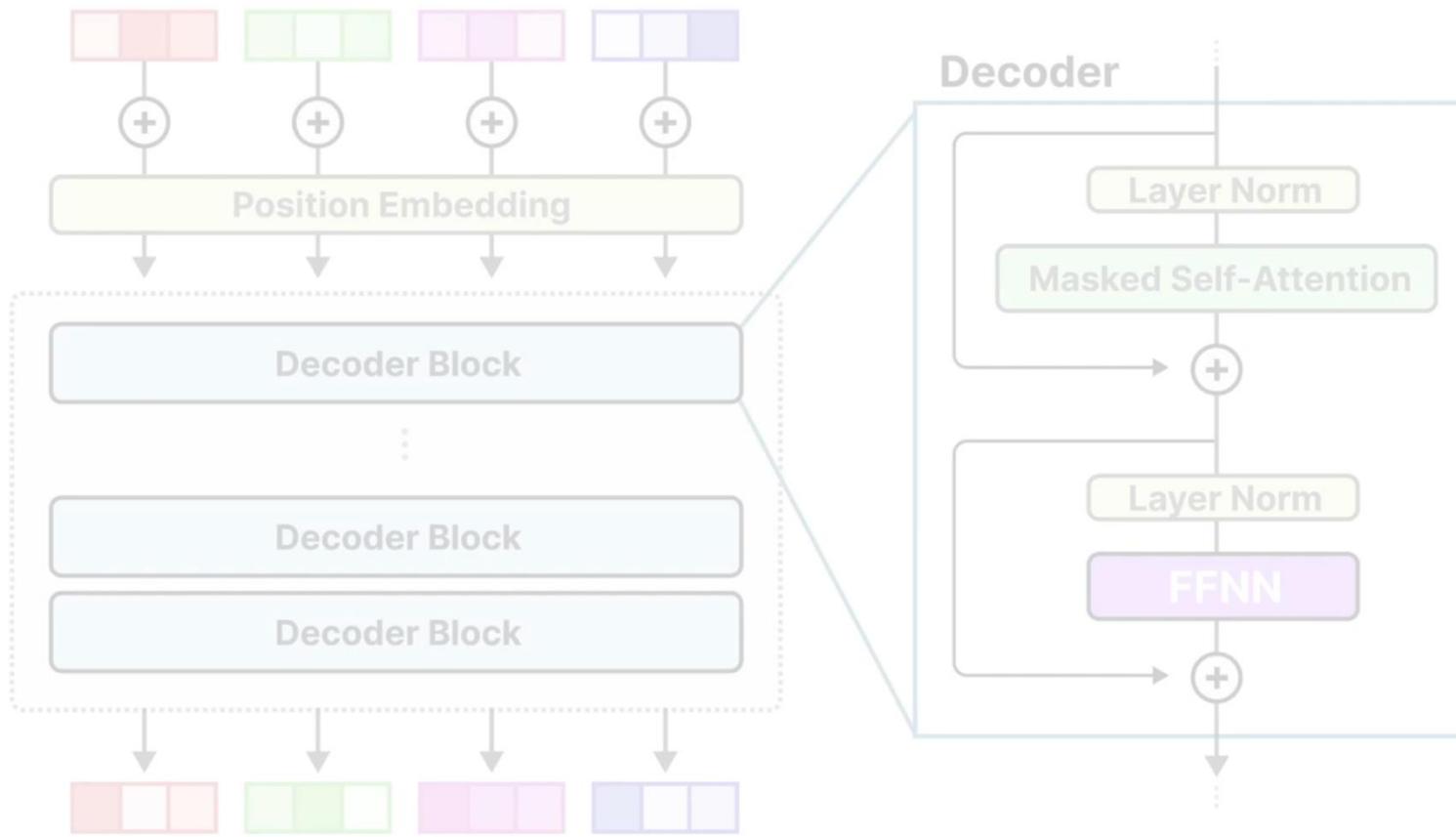


The **router** (*gate network*) selects the **expert**(s) best suited for a given input:

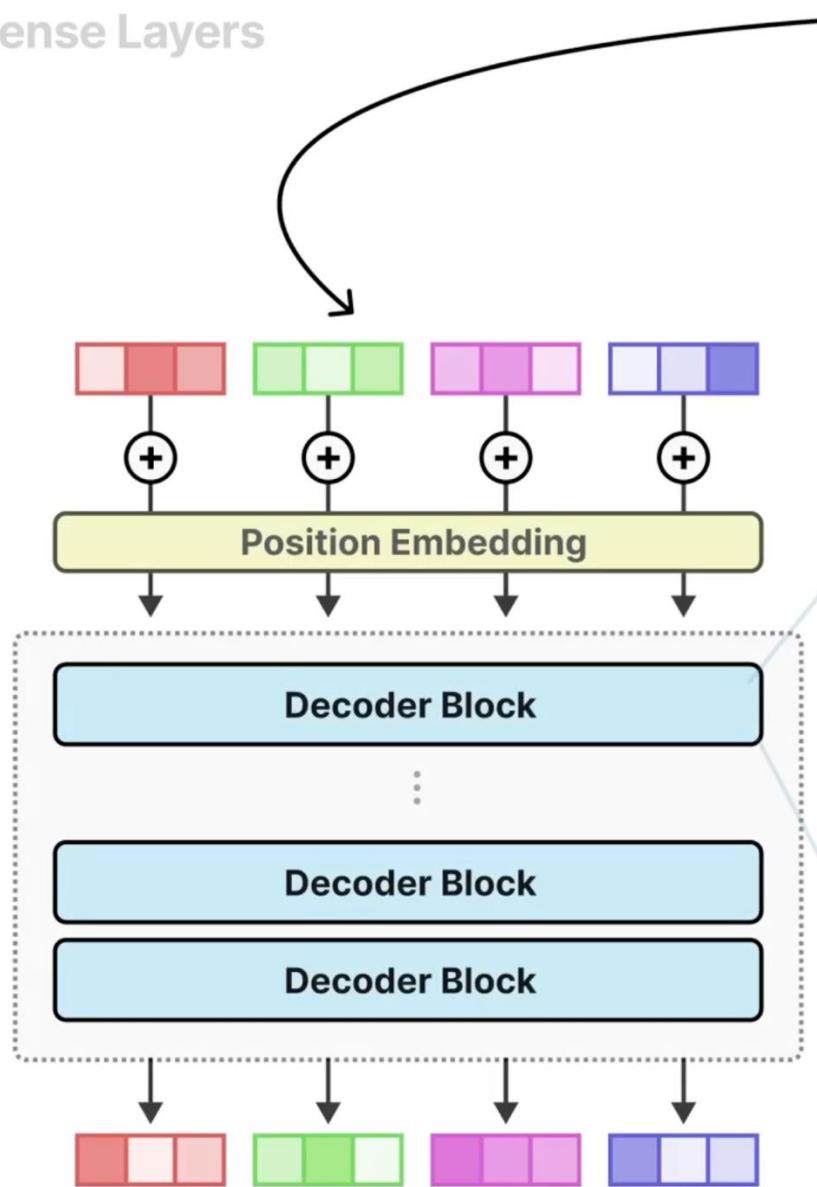


Introduction: MOE 基本原理

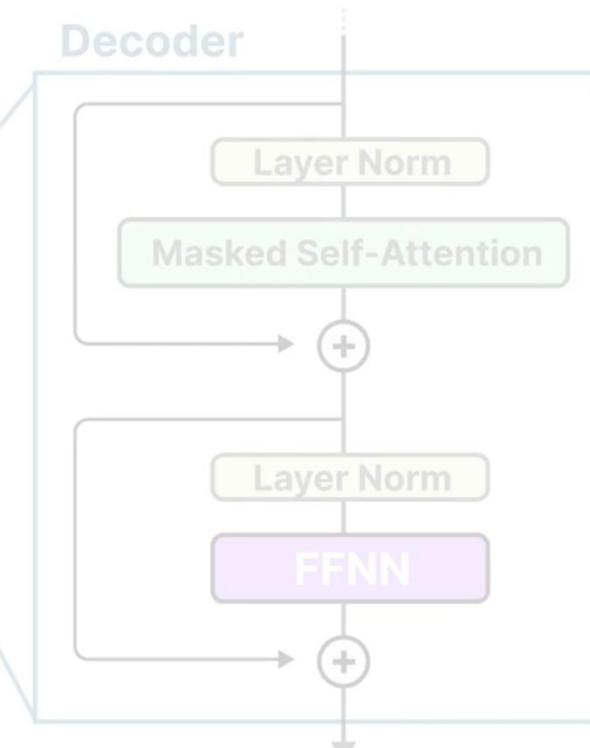
To explore what experts represent and how they work, let us first examine what MoE is supposed to replace; **the dense layers**.



Dense Layers

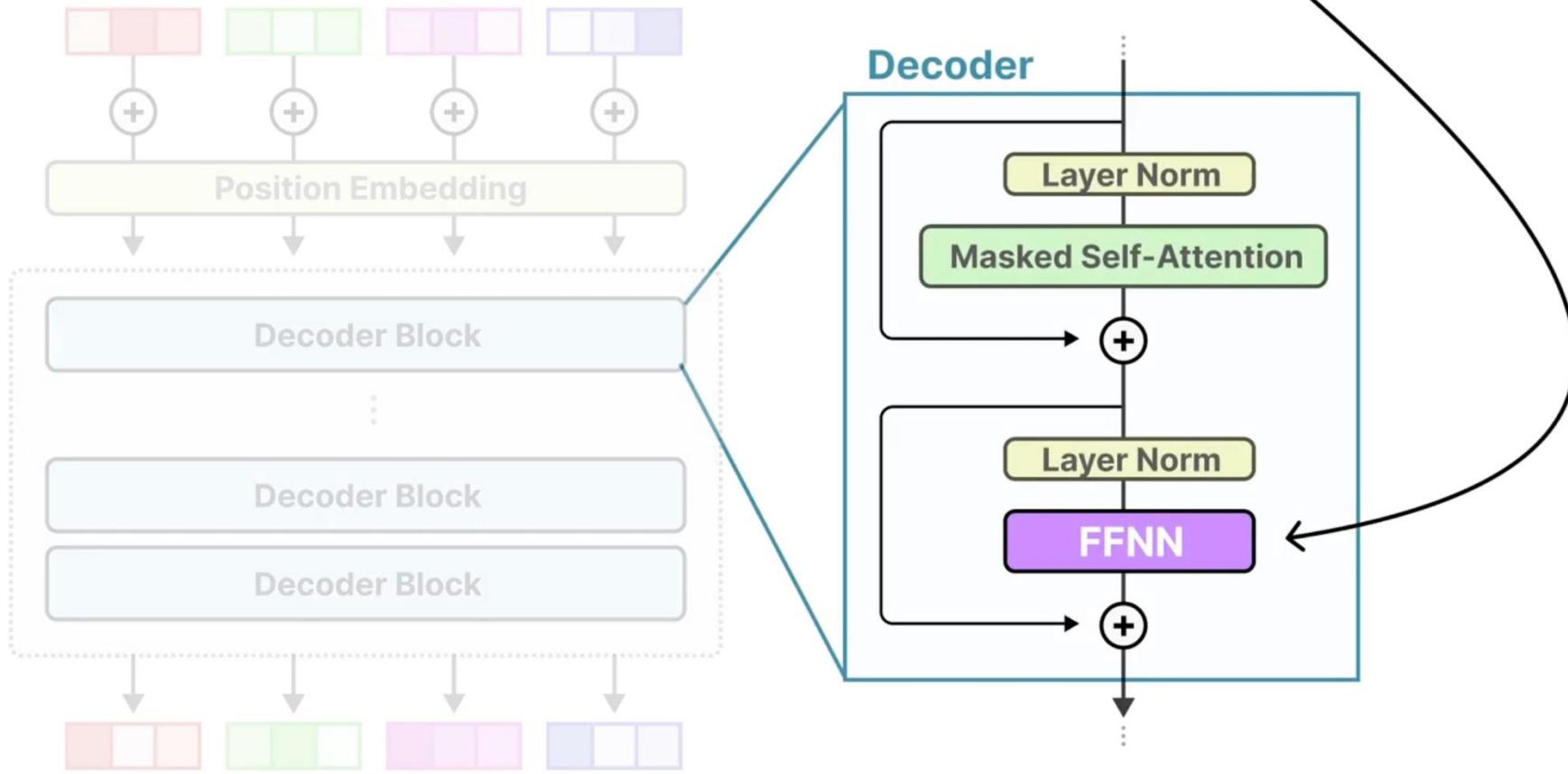


Remember that a standard **decoder-only** Transformer architecture....

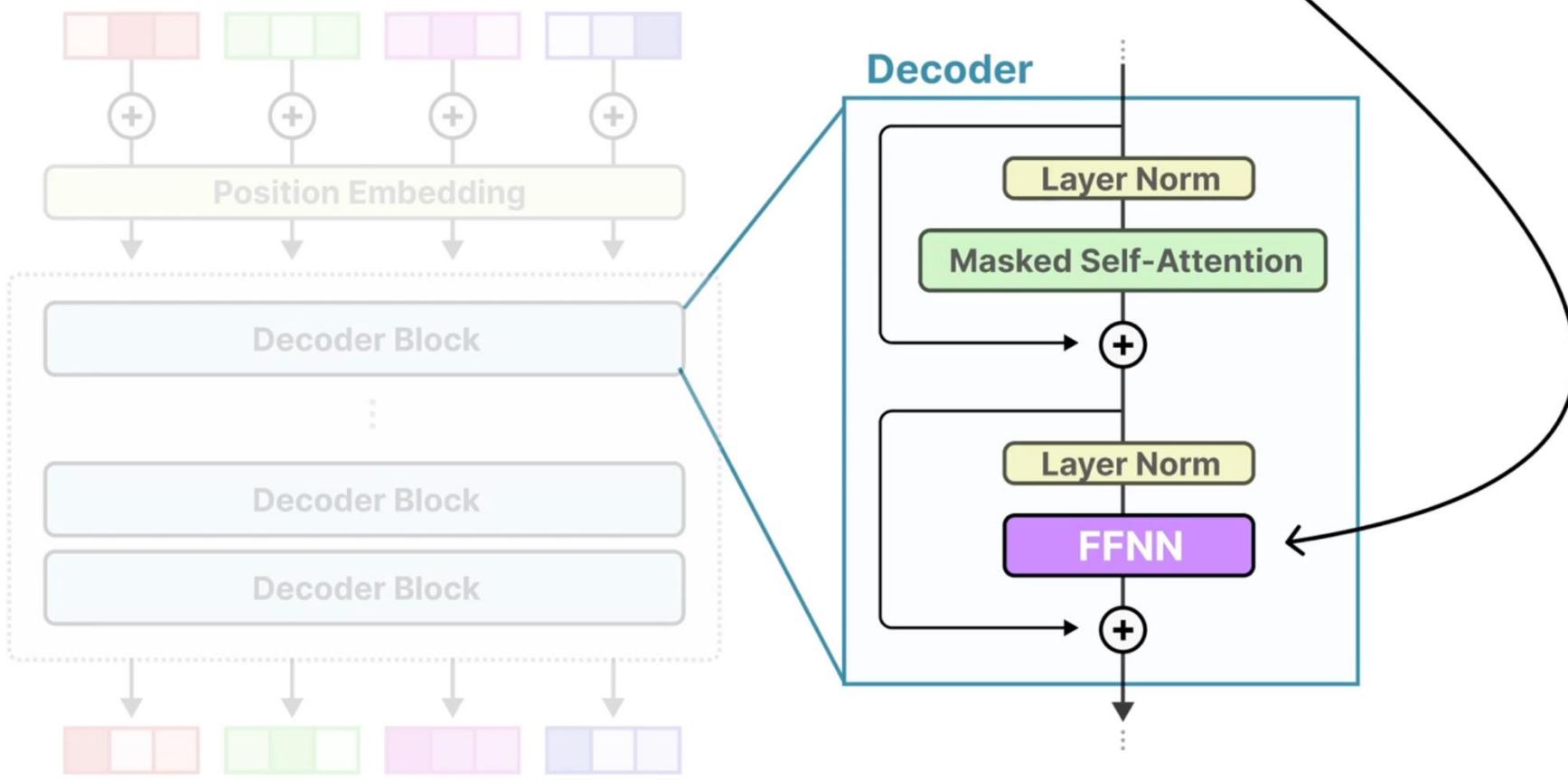


Introduction: MOE 基本原理

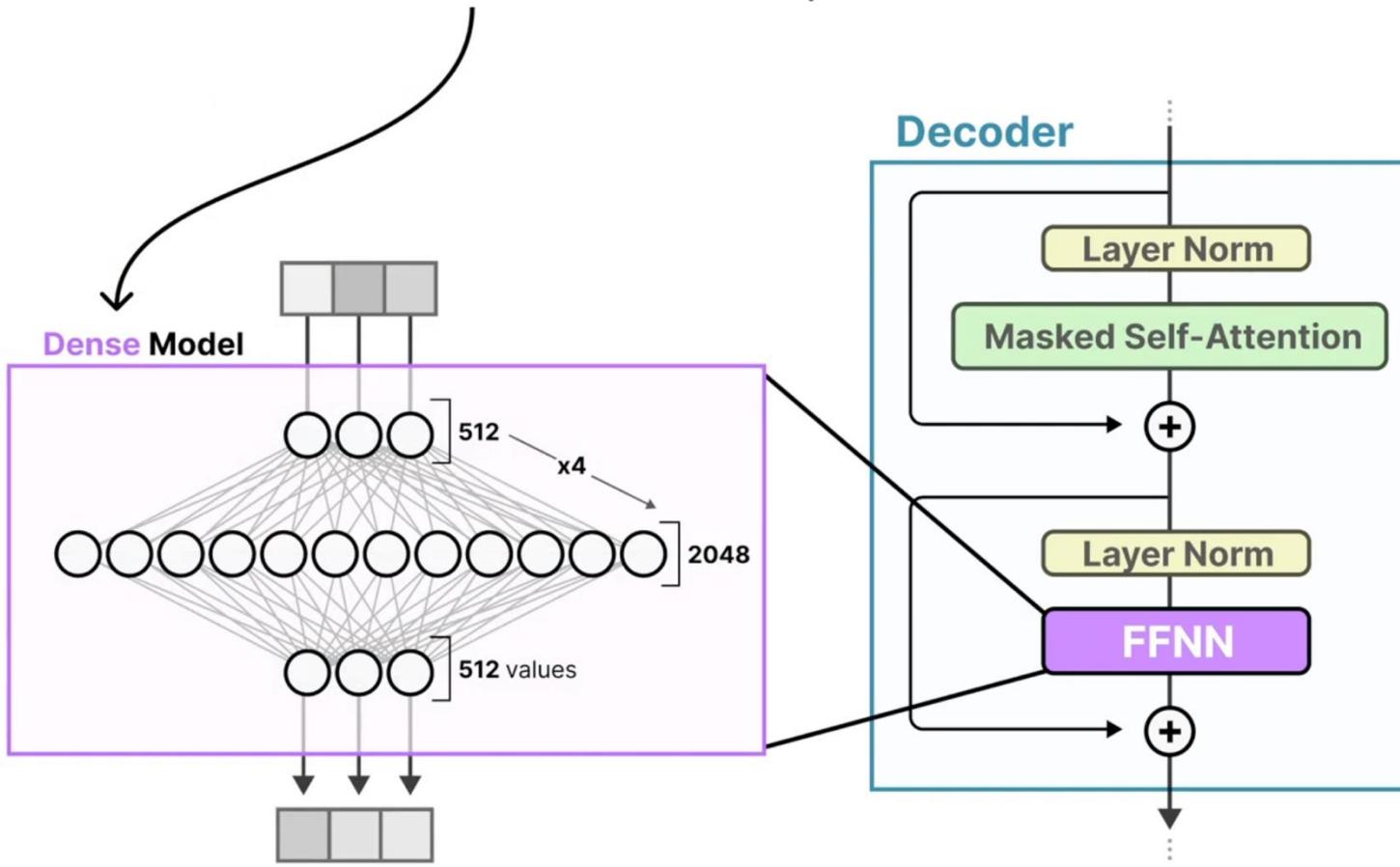
...has the **Feed Forward Neural Network (FFNN)** applied after layer normalization.



The **FFNN** uses the contextual information created by attention to capture complex relationships in the data.

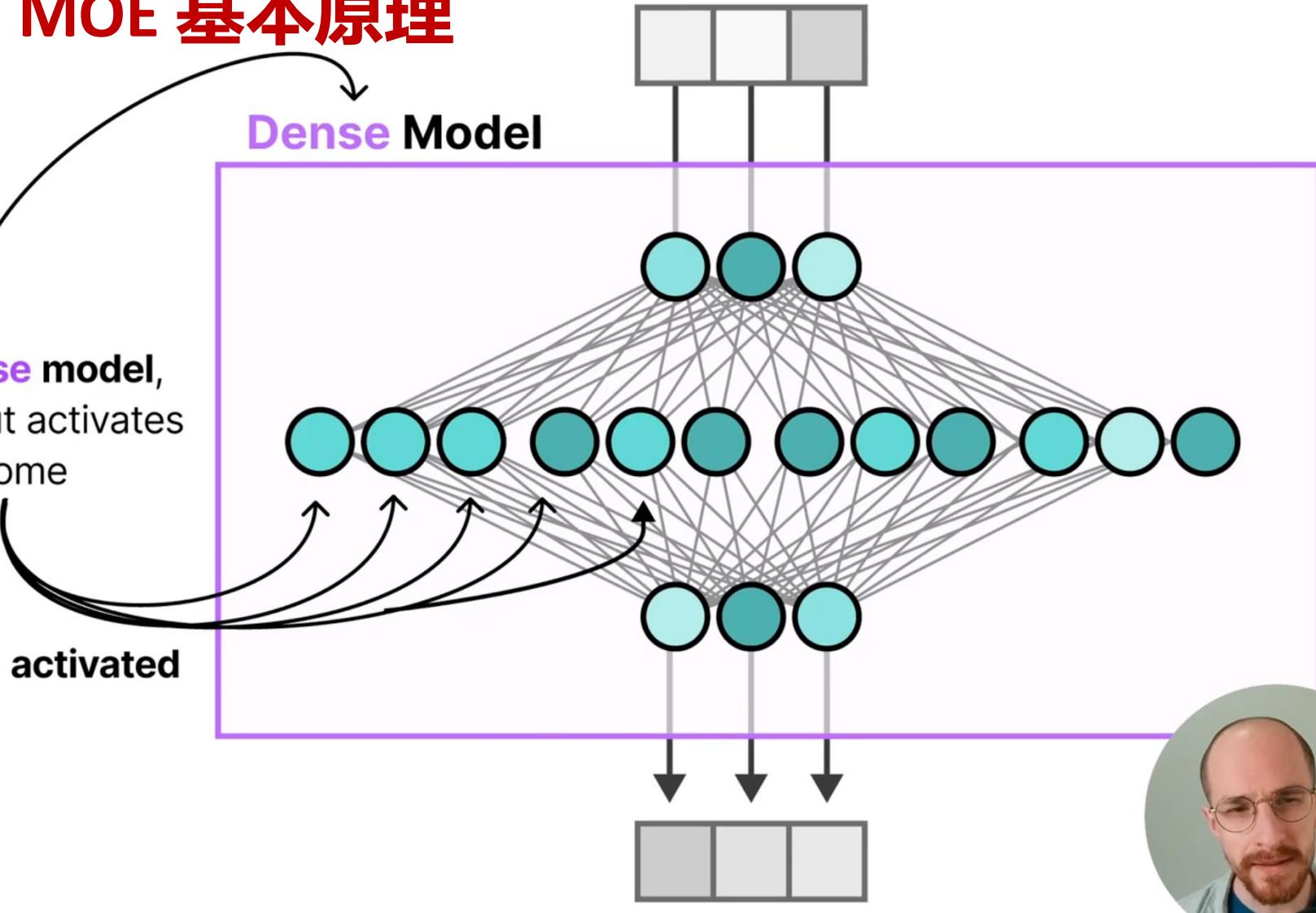


The **FFNN** in a **decoder-only** model is called a **dense model** since all parameters are activated.



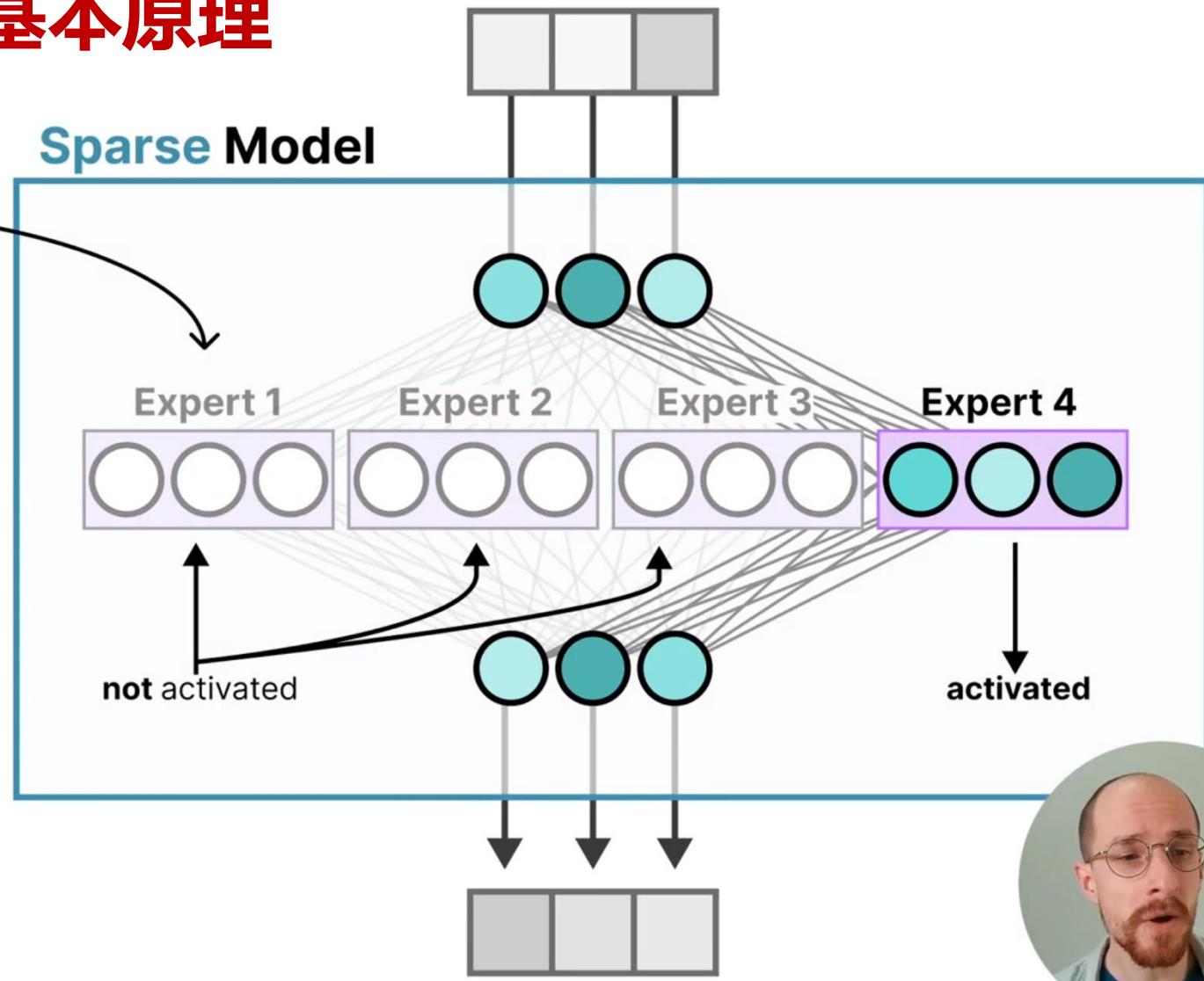
Introduction: MOE 基本原理

Looking at the **dense model**, notice how the input activates all **parameters** to some degree.



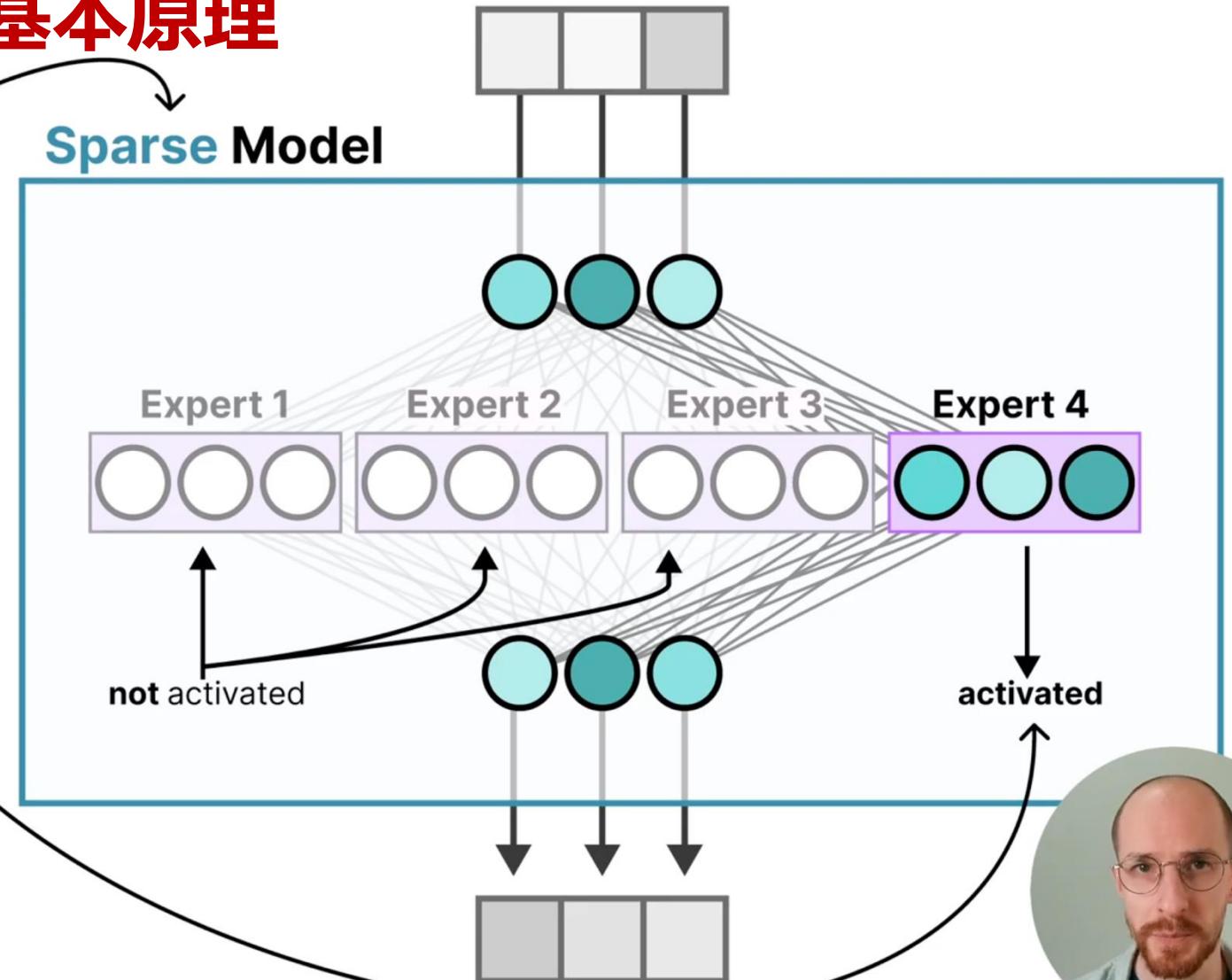
Introduction: MOE 基本原理

We can **chop up** our dense model into pieces (so-called **experts**), retrain it, and only activate a **subset of experts** at a given time.



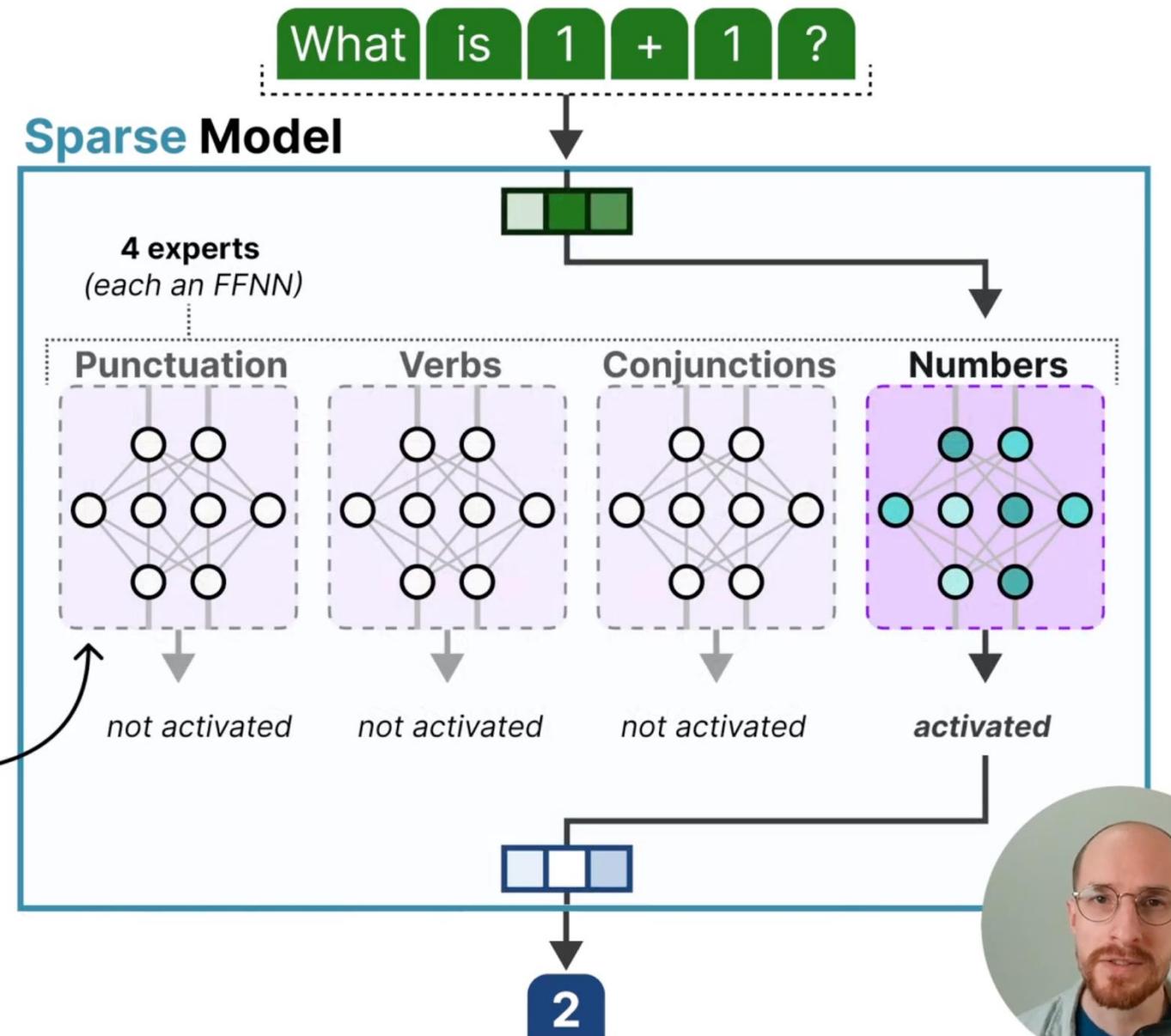
Introduction: MOE 基本原理

This is called a **Sparse model**. During inference, only **specific** experts are used.



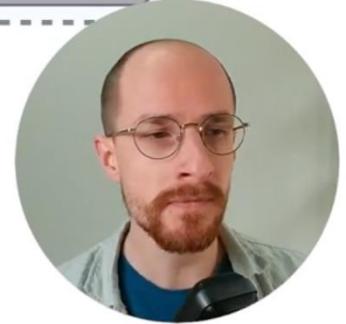
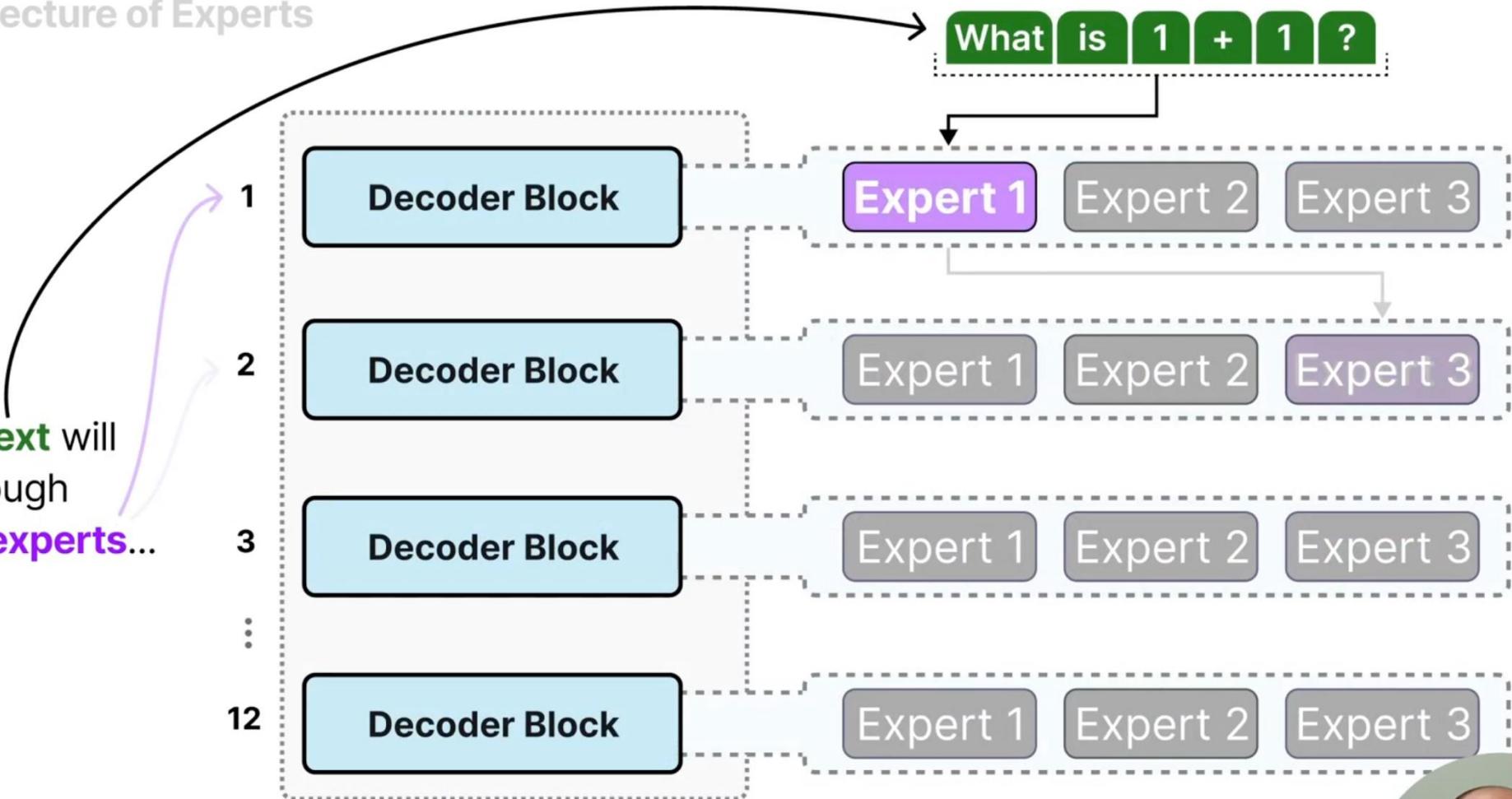
Sparse Layers

In practice, experts are typically
whole FFNNs themselves...

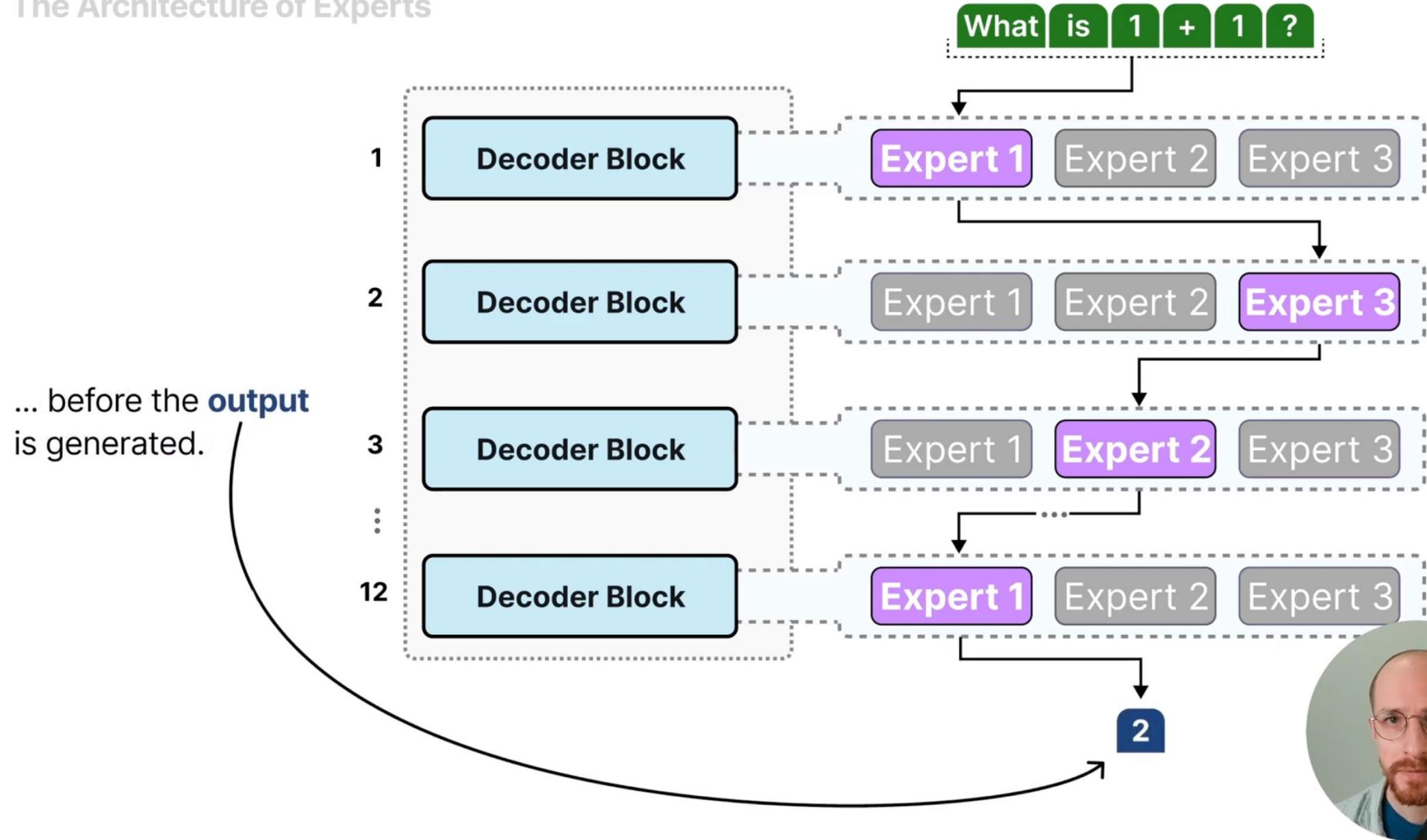


The Architecture of Experts

A given **text** will pass through multiple **experts**...

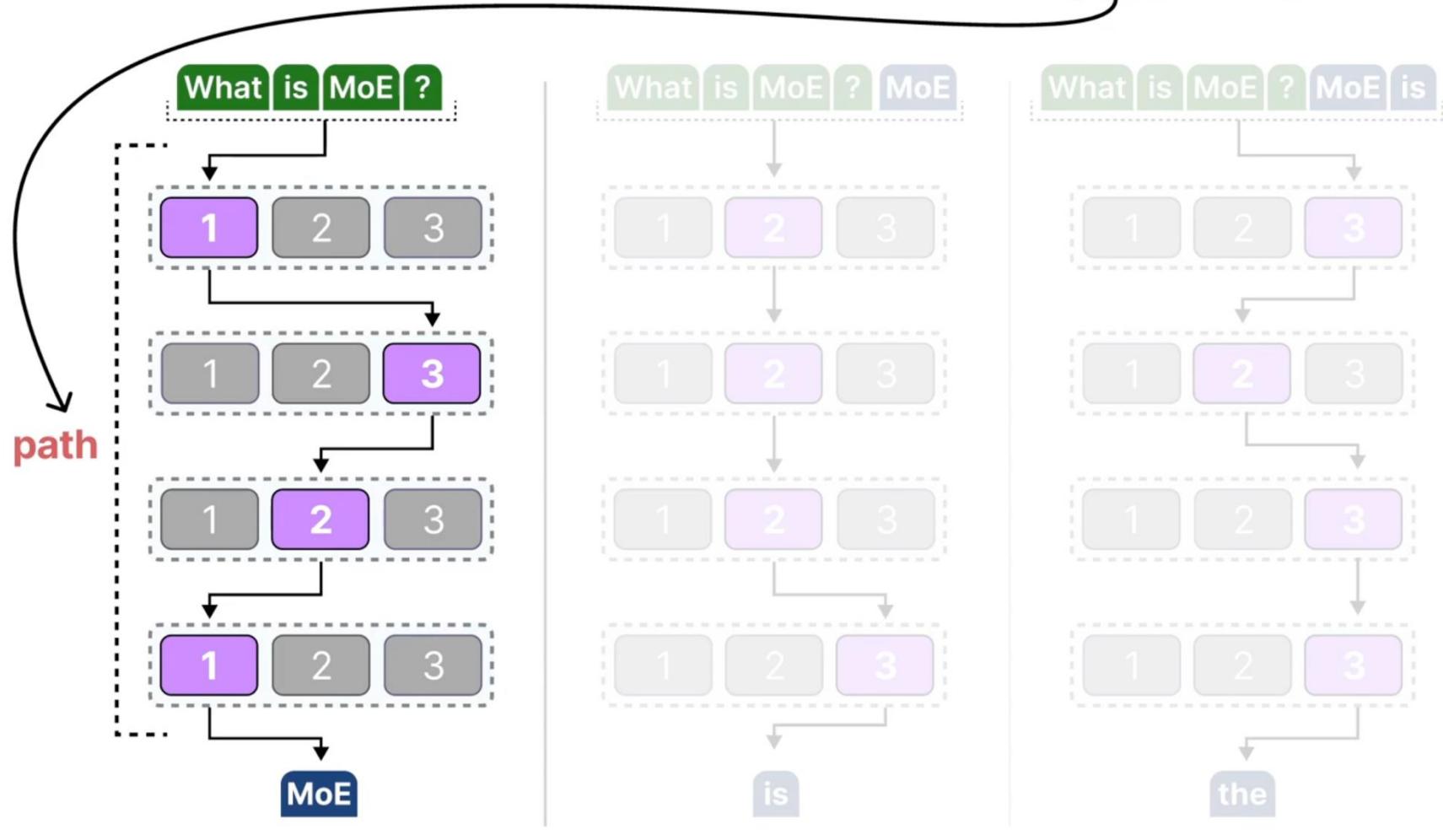


The Architecture of Experts



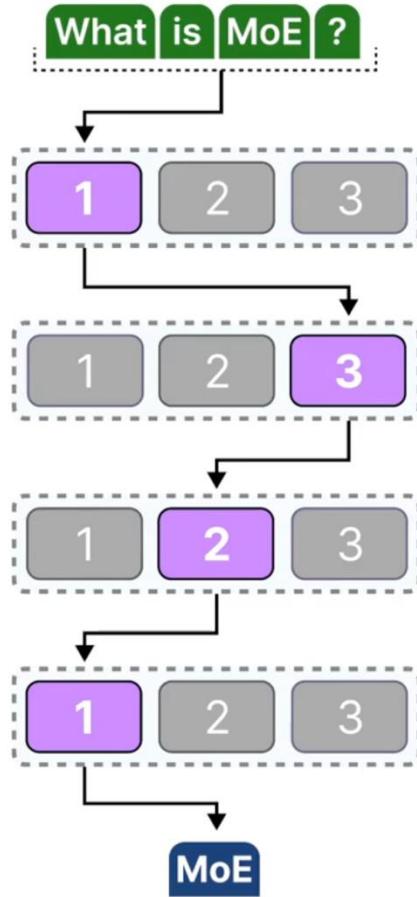
The Architecture of Experts

The **chosen experts** likely differ between tokens which results in different “**paths**” being taken.

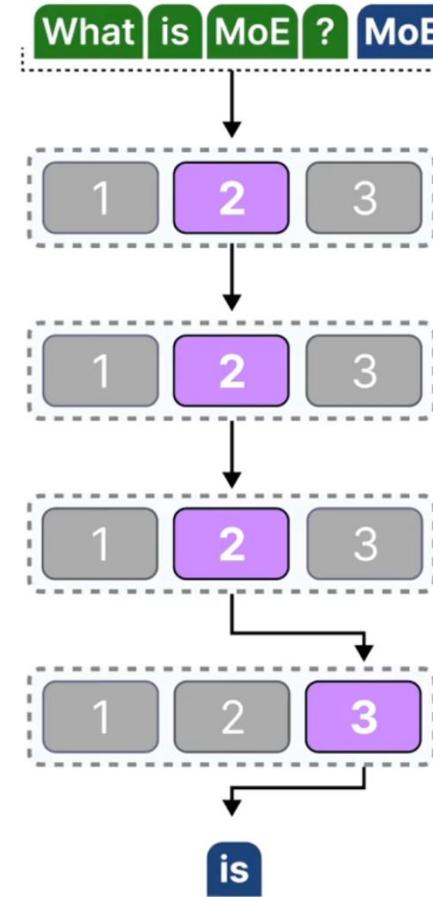


The Architecture of Experts

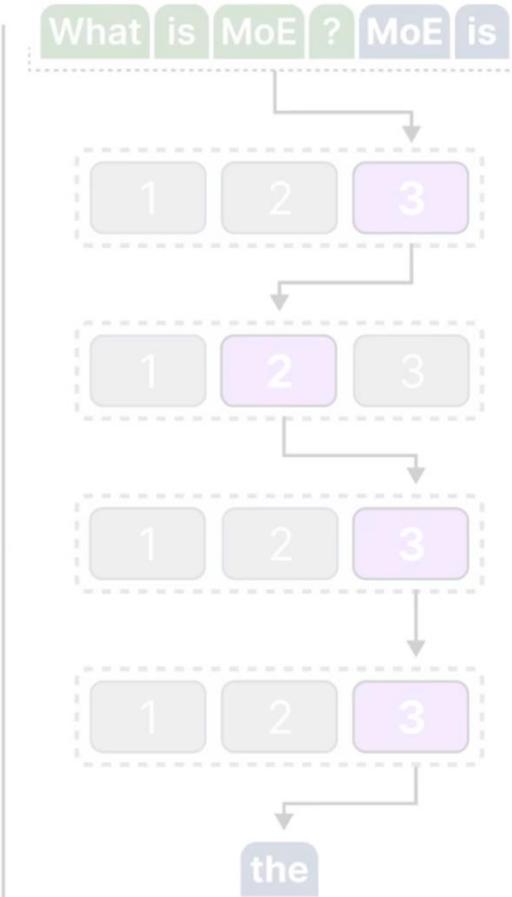
Each **new token** may result in a different path...



First pass



Second pass

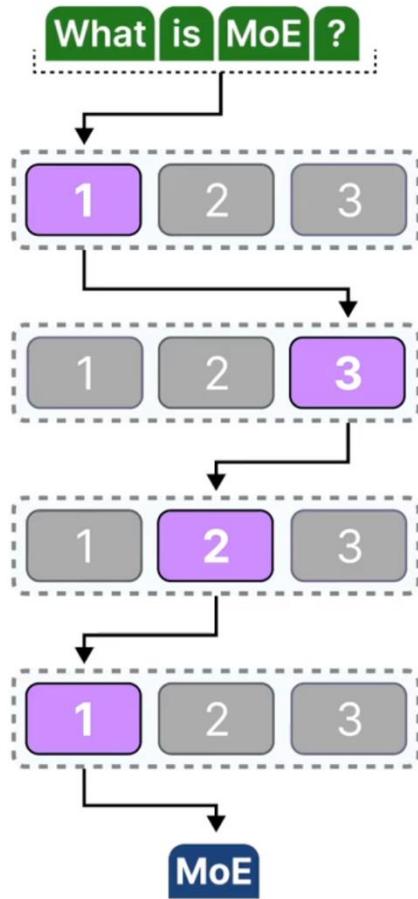


Third pass

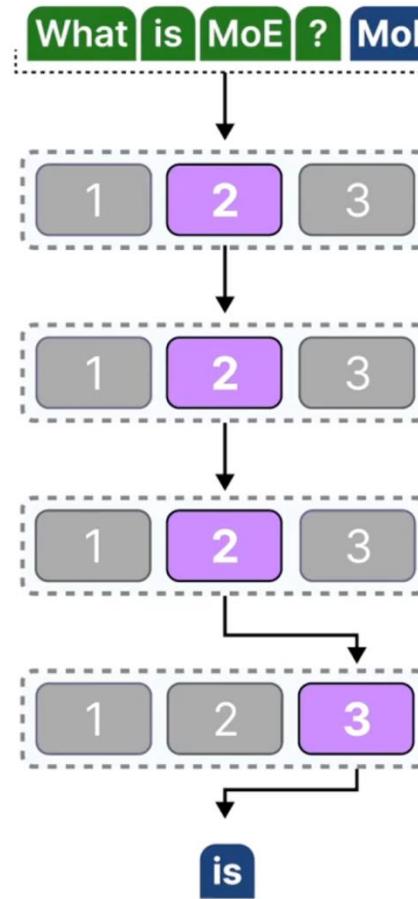


The Architecture of Experts

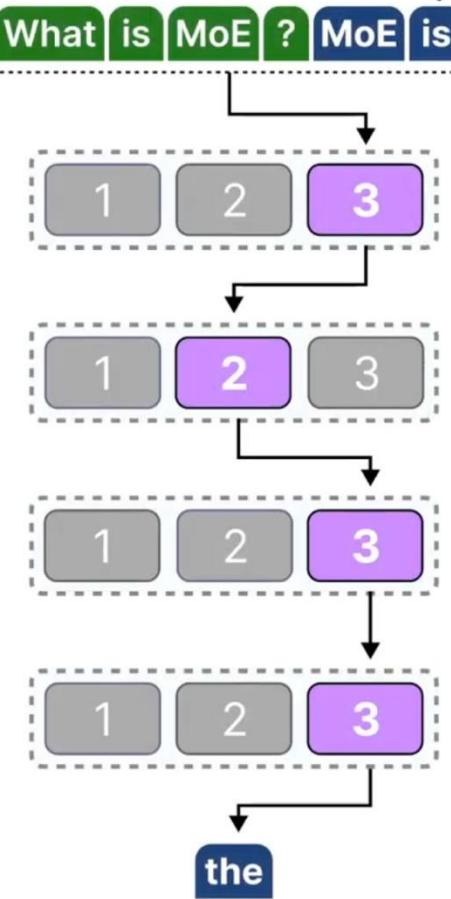
... and may activate a different set of experts.



First pass



Second pass

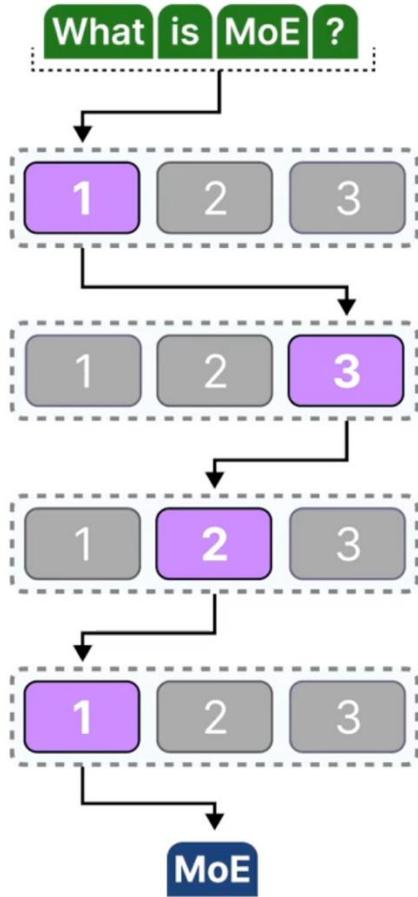


Third pass

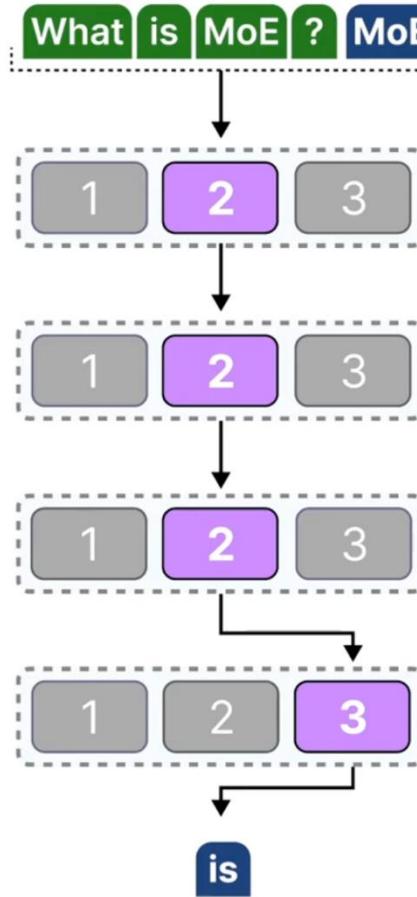


The Architecture of Experts

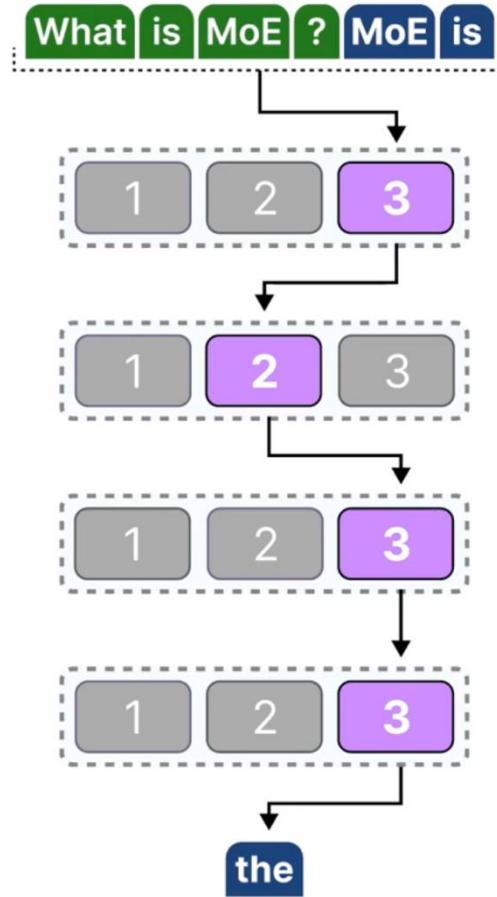
This means that each time you run inference, a set of **experts** is chosen that are best suited for the input.



First pass



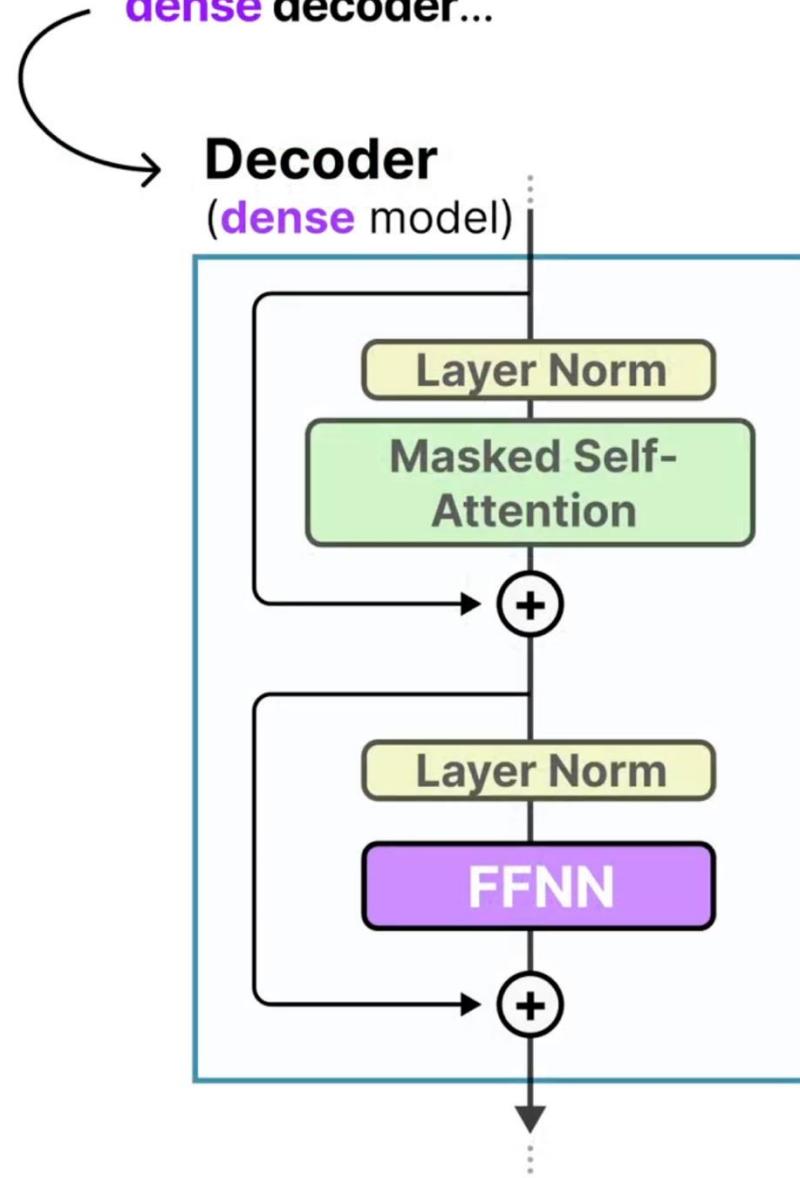
Second pass



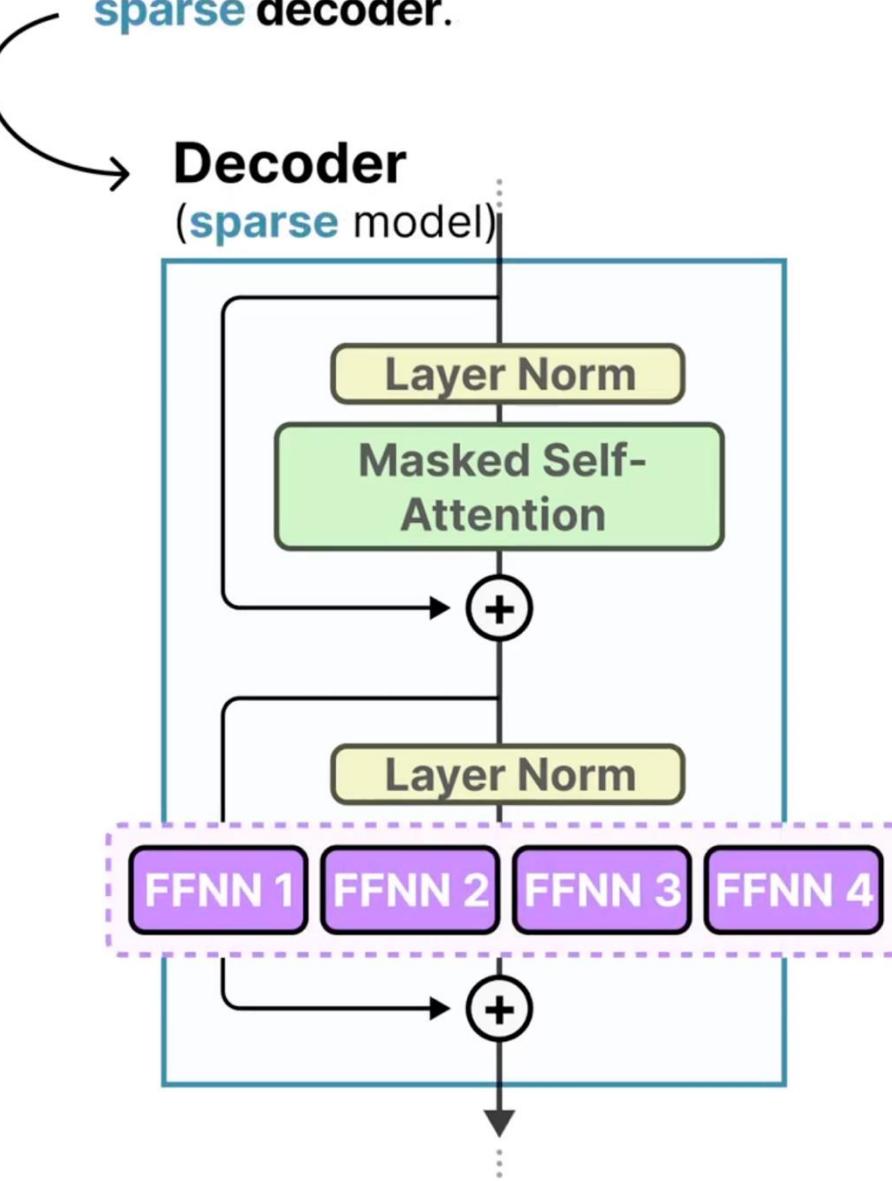
Third pass



If we update our visualization of the
dense decoder...

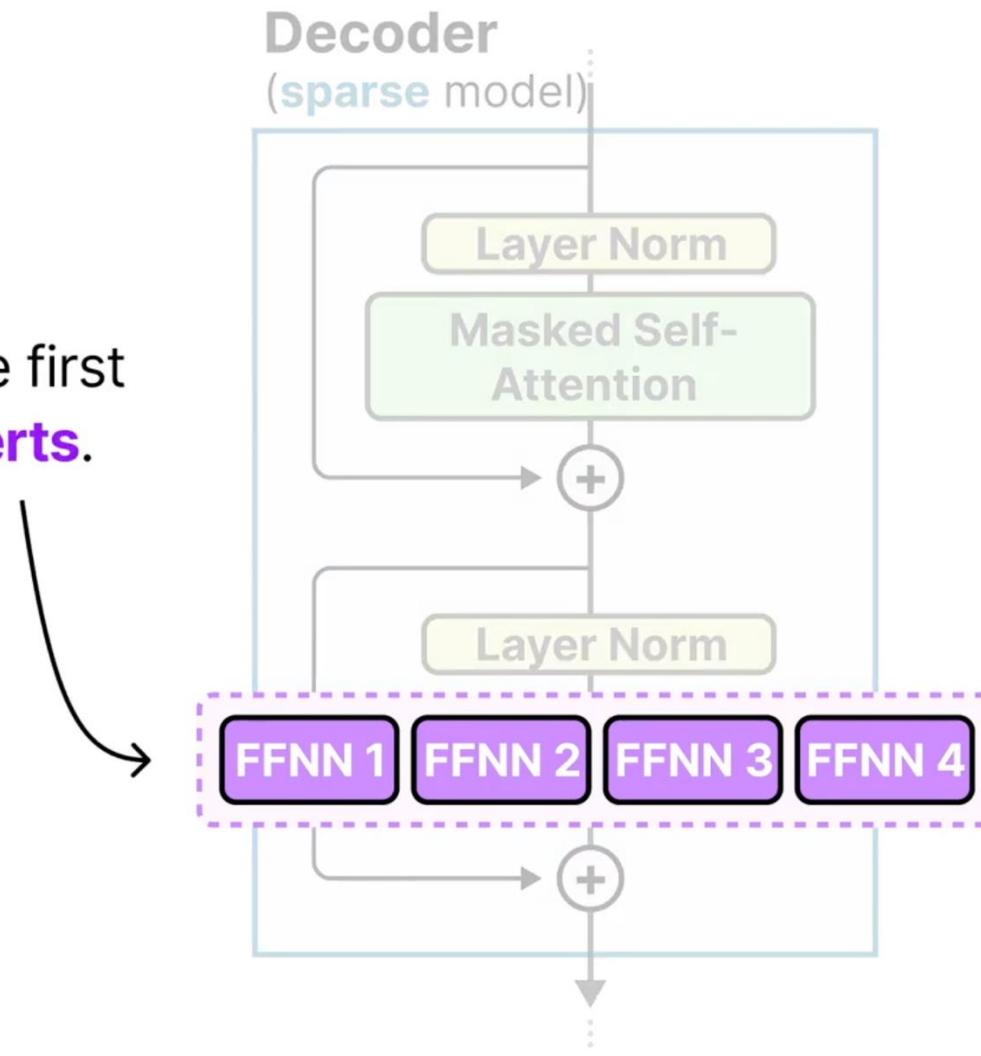


...with multiple **FFNNs** it would now be called a **sparse decoder**.



Introduction: MOE 基本原理

Thereby capturing the first part of **MoE**, the **experts**.

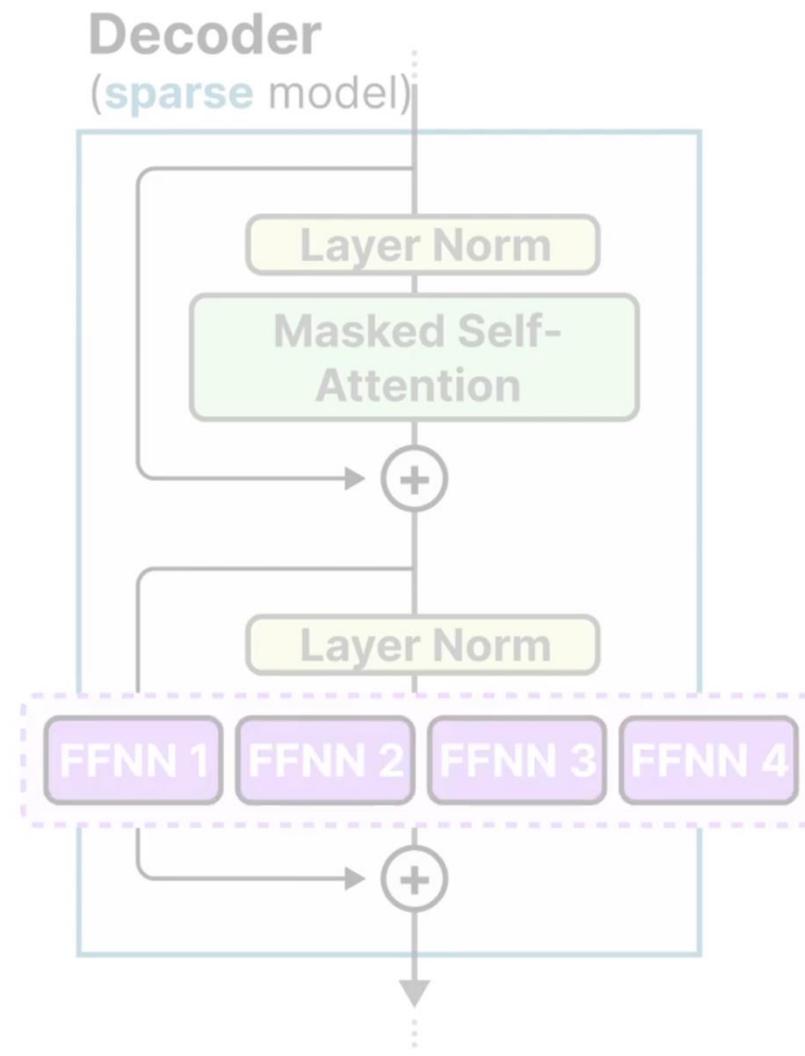


02

The Router: 路由原理

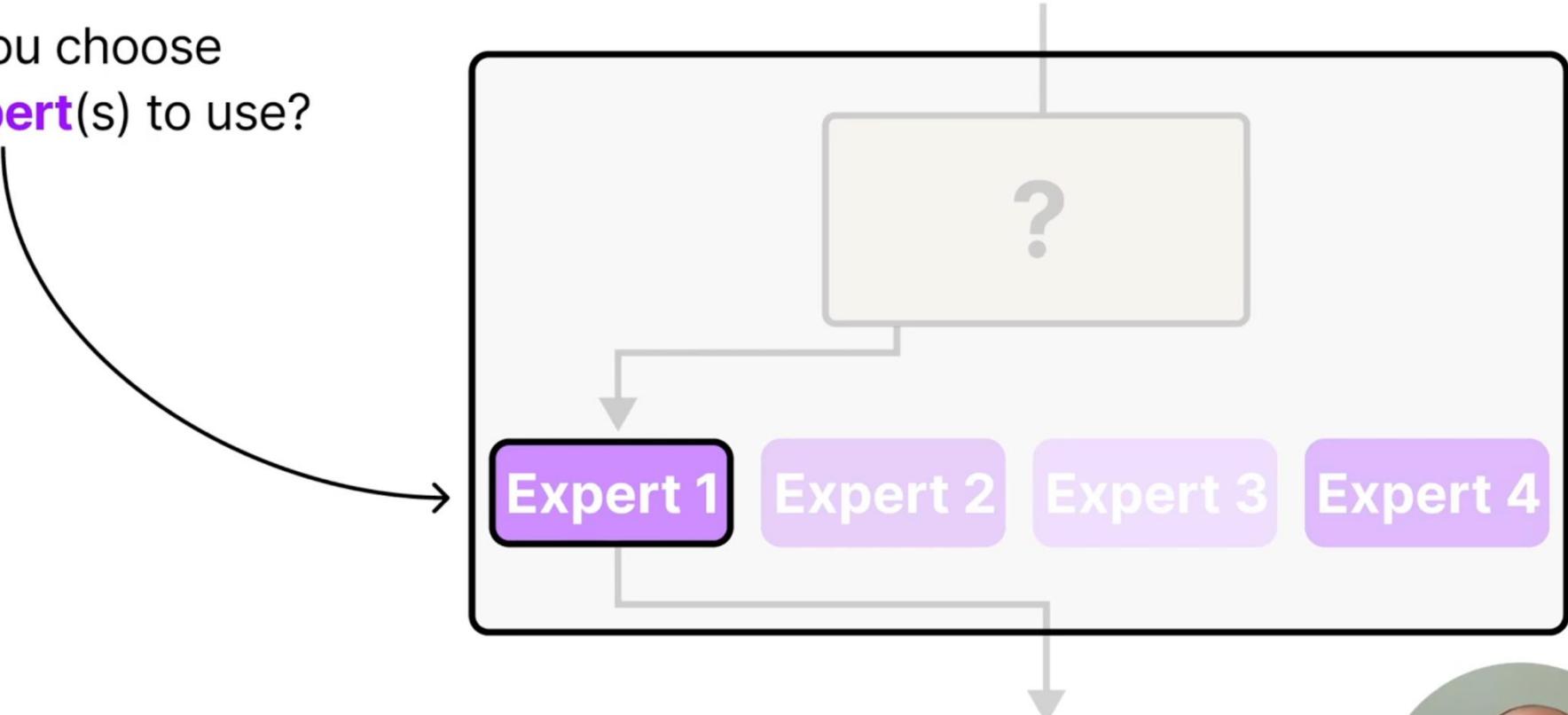
Introduction: MOE 基本原理

But there is still a piece
of the puzzle missing...



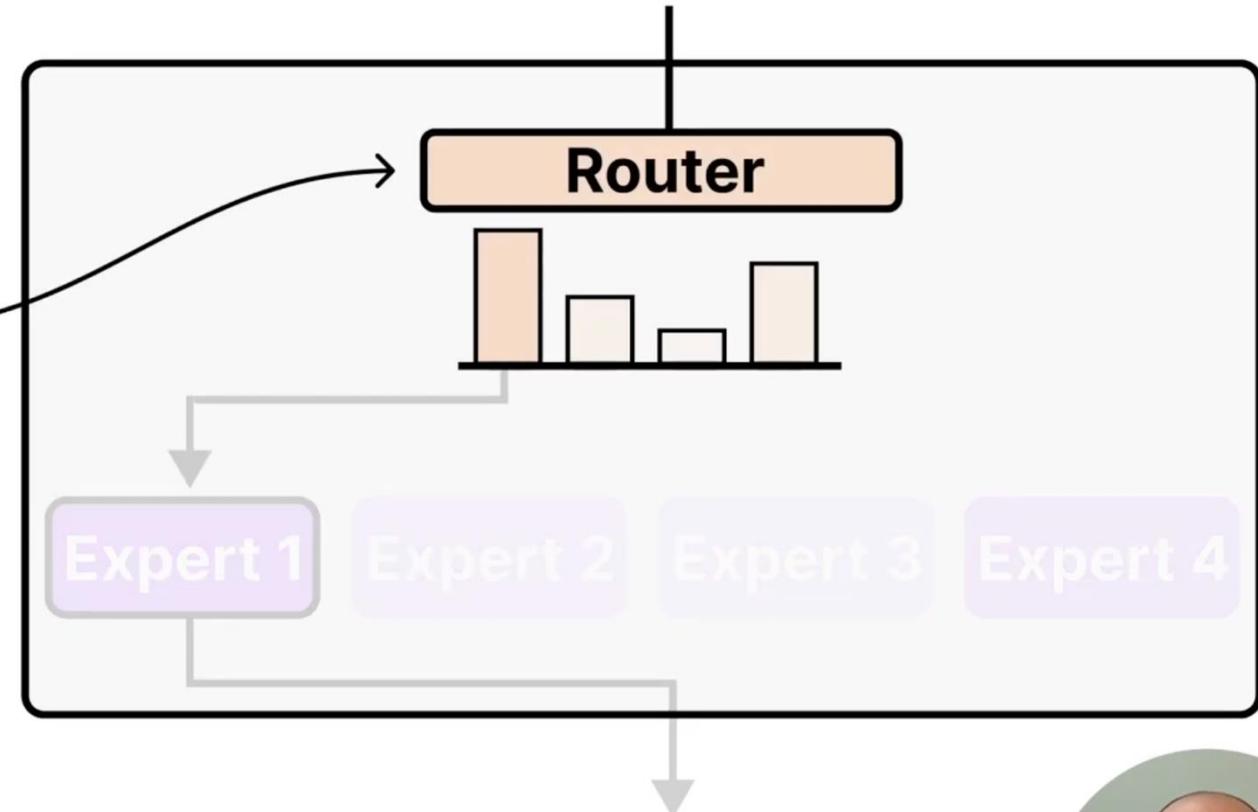
The Router

How do you choose
which **expert**(s) to use?



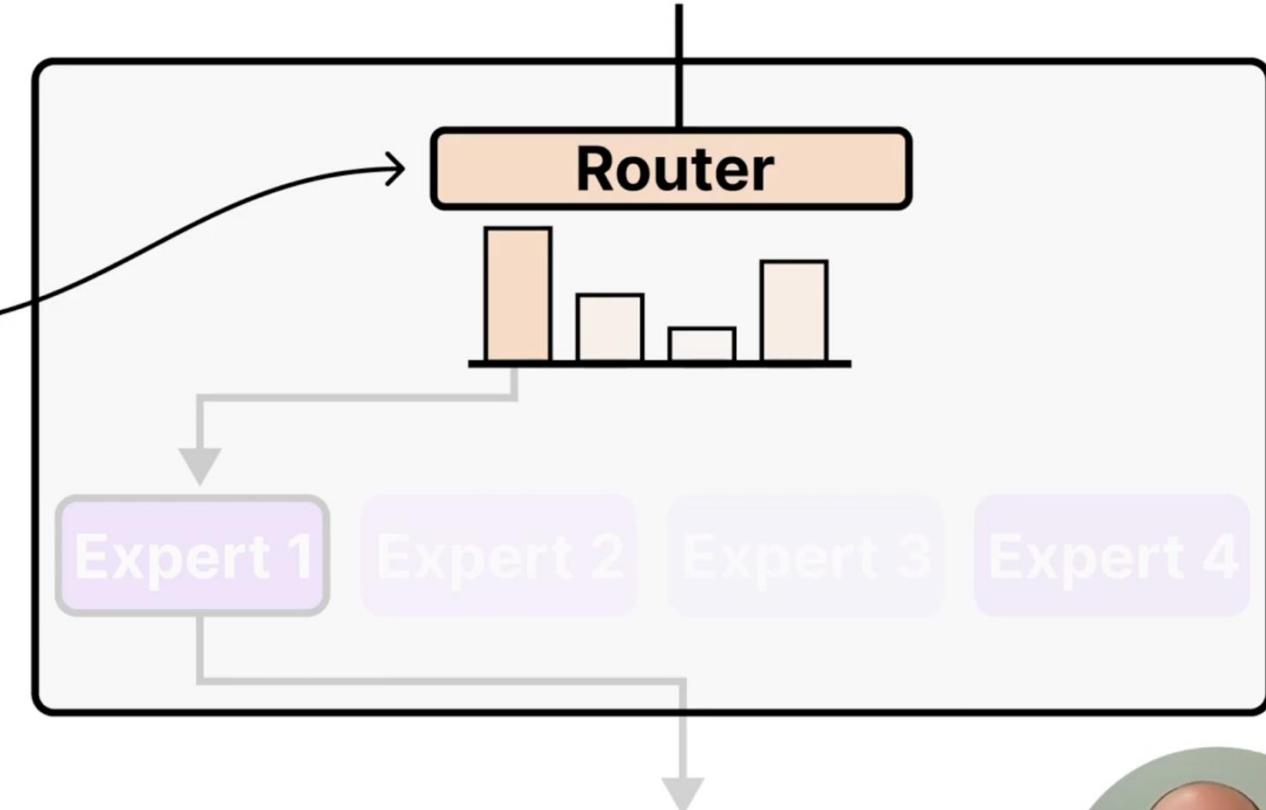
The Router

That's where the **router** comes in!



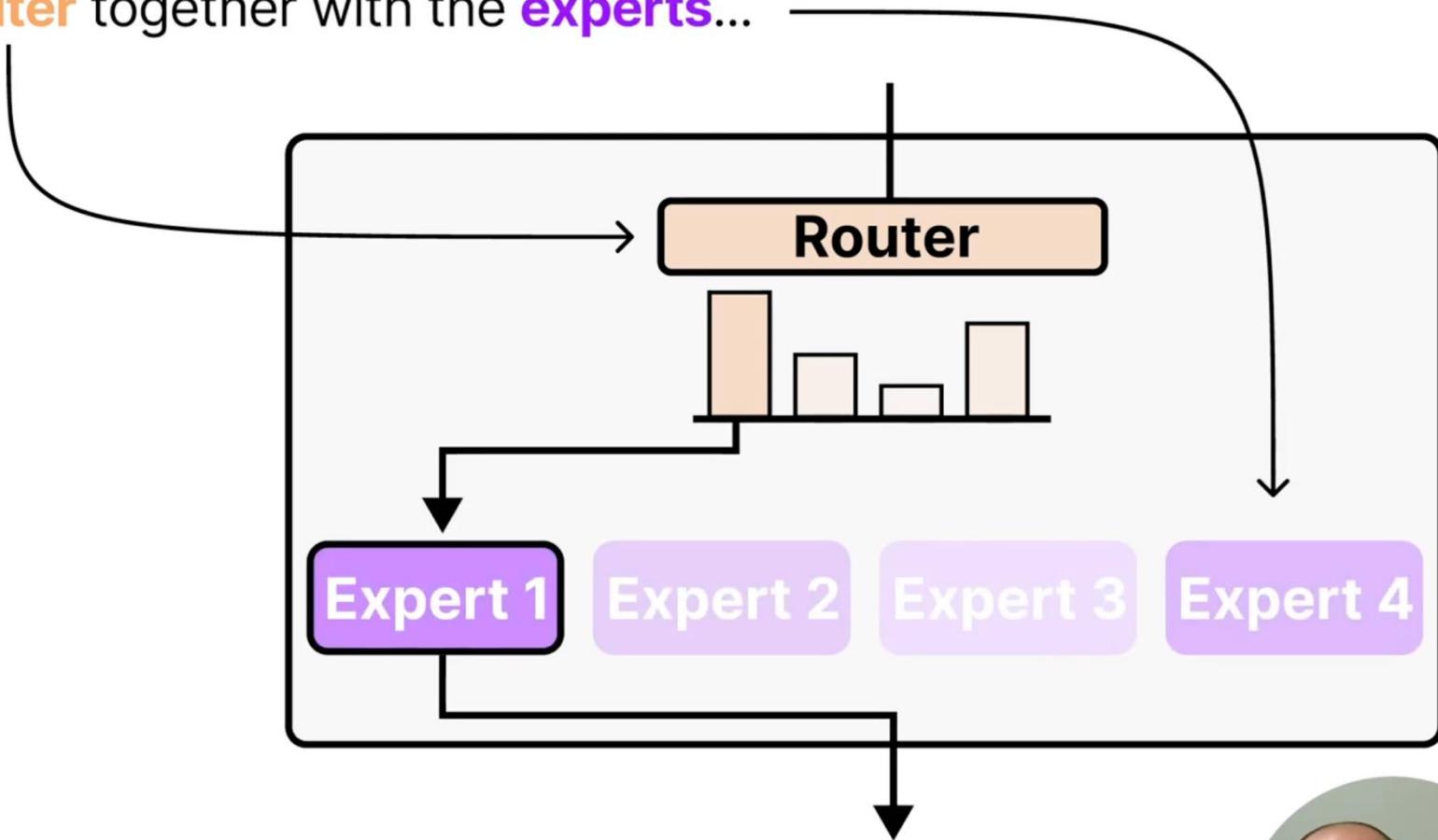
The Router

... it helps us decide which **expert** is best suited for a given input.

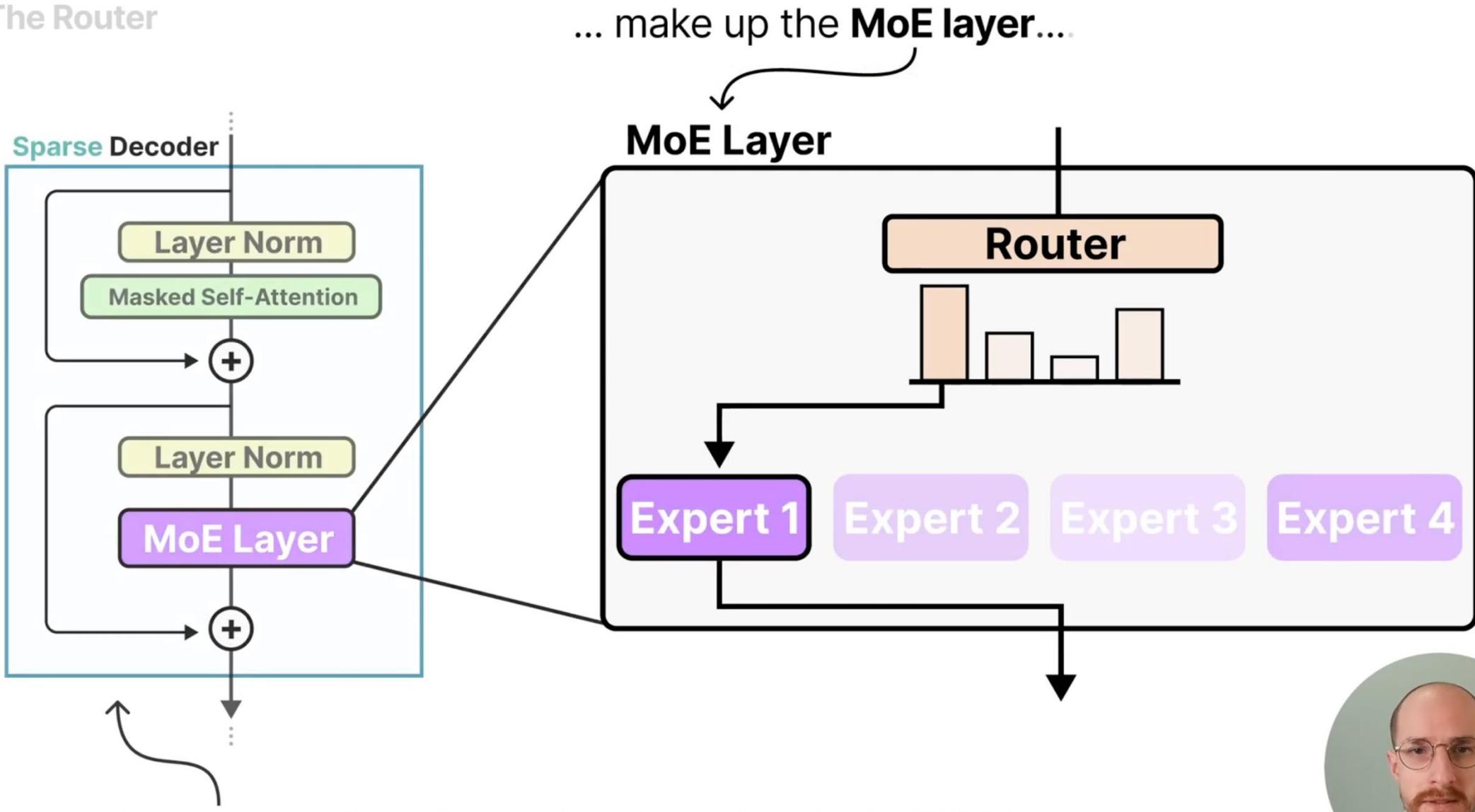


The Router

The **router** together with the **experts**...



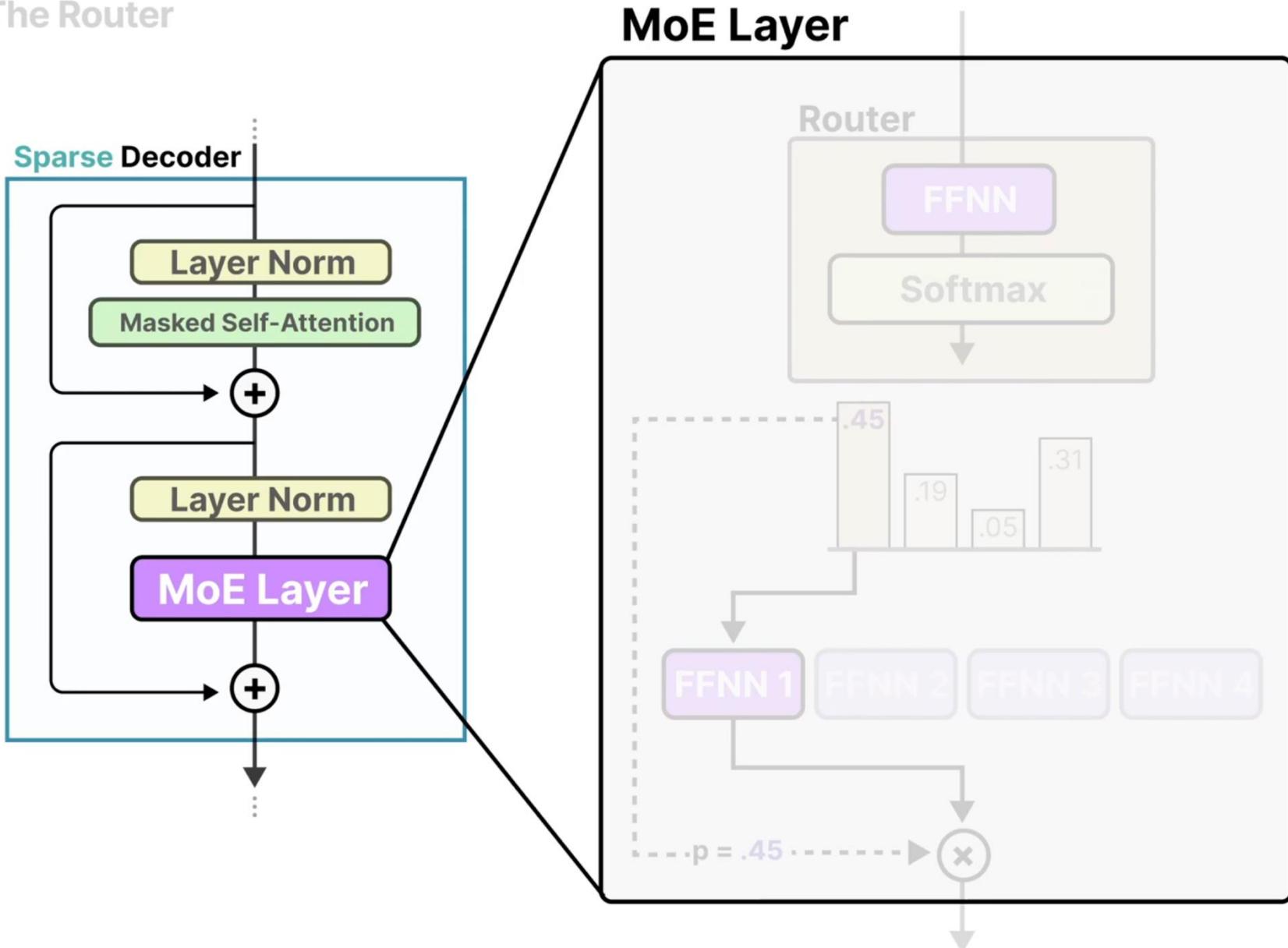
The Router



... in a **sparse decoder** and replace the single FFNN.



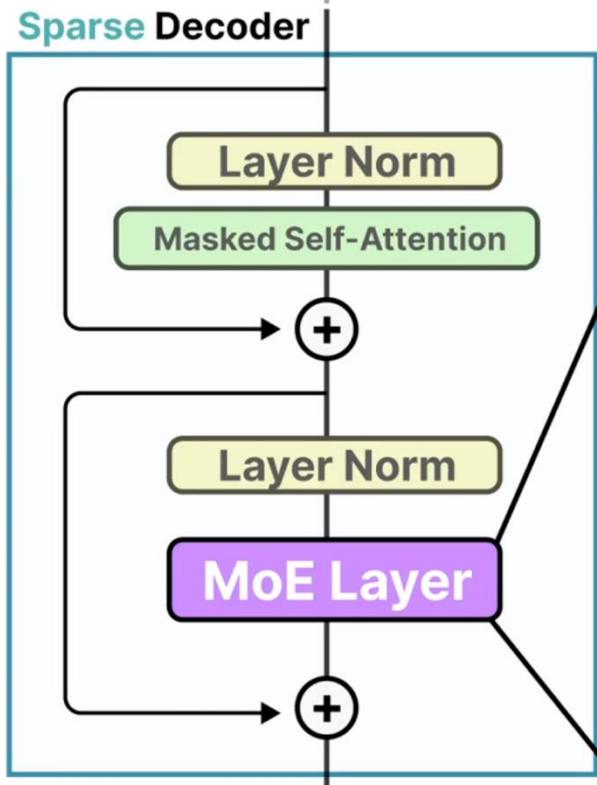
The Router



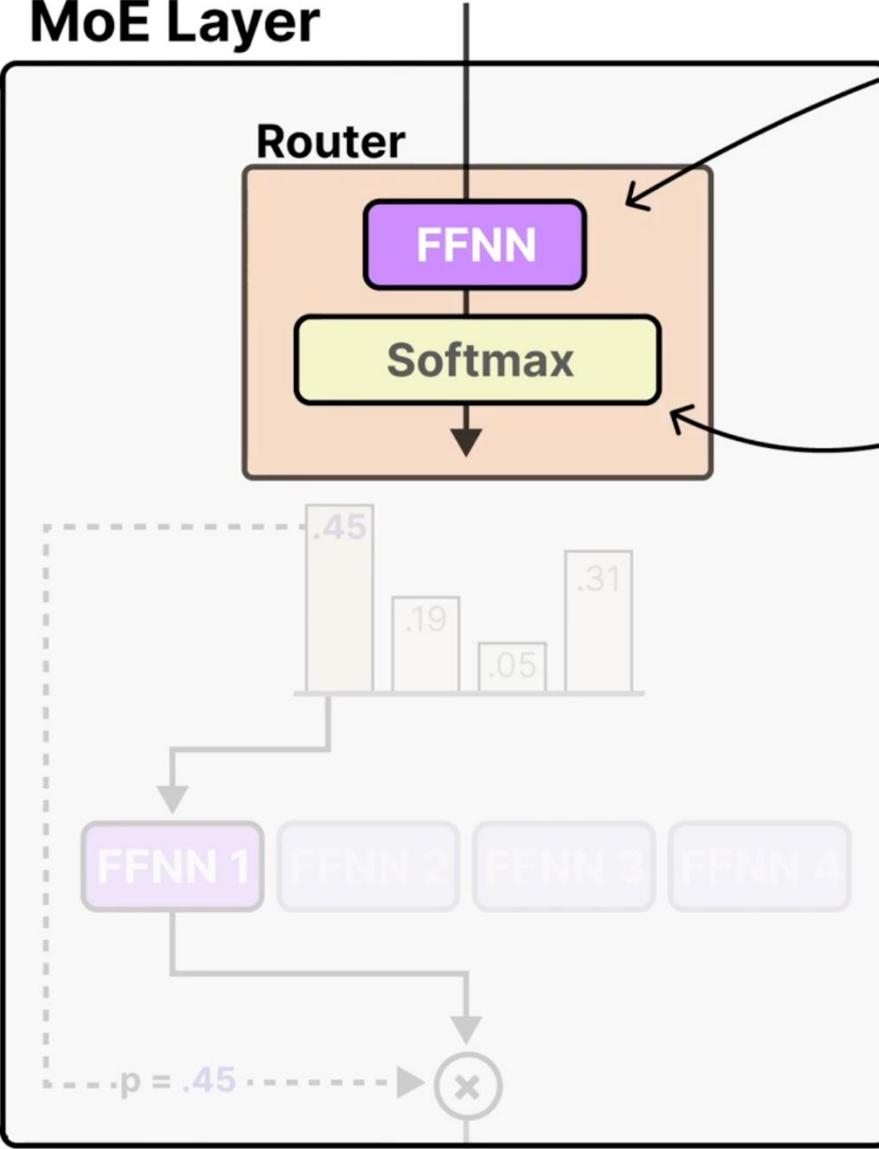
We can zoom in on the **MoE** layer and explore how the **router** works in detail.



The Router



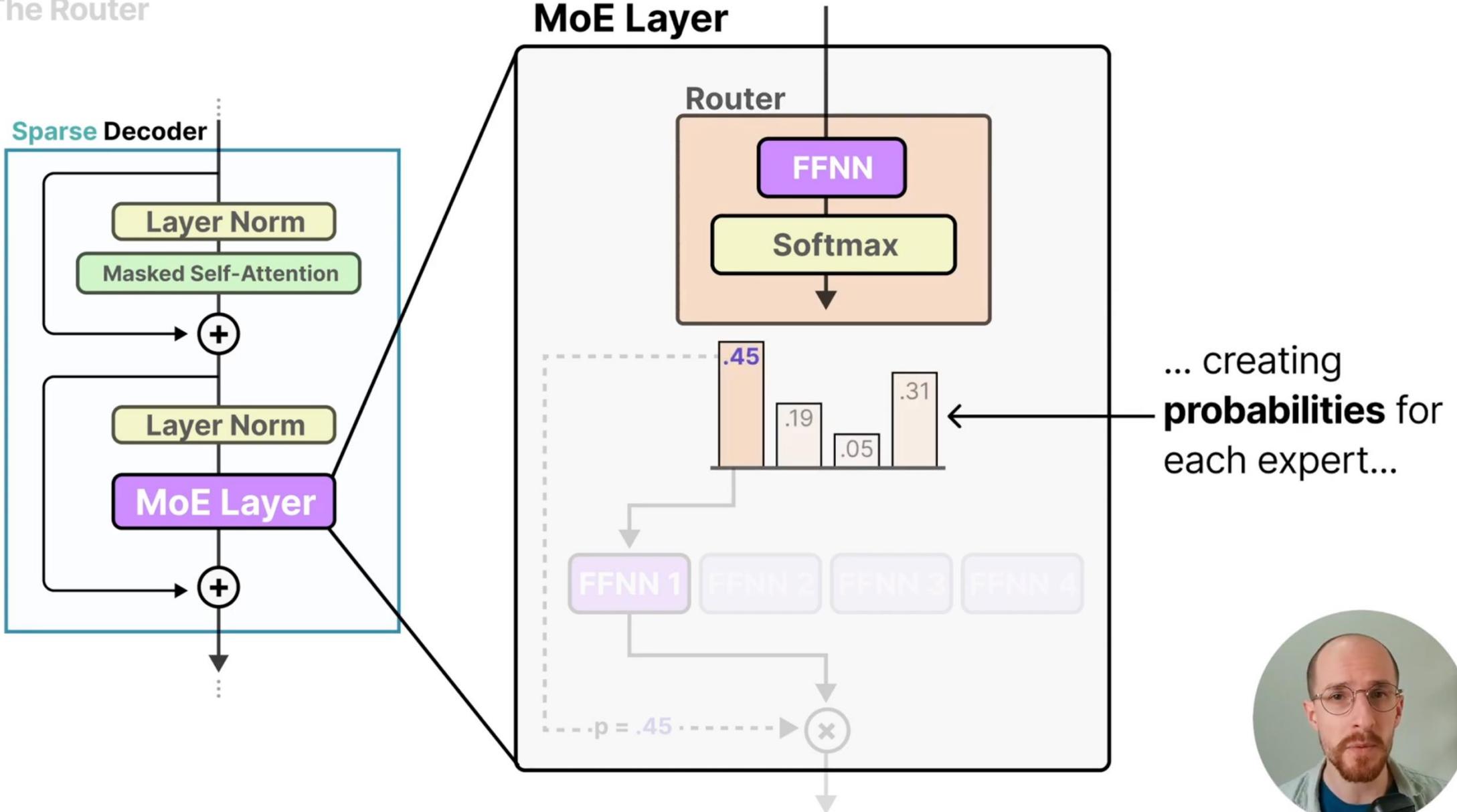
MoE Layer



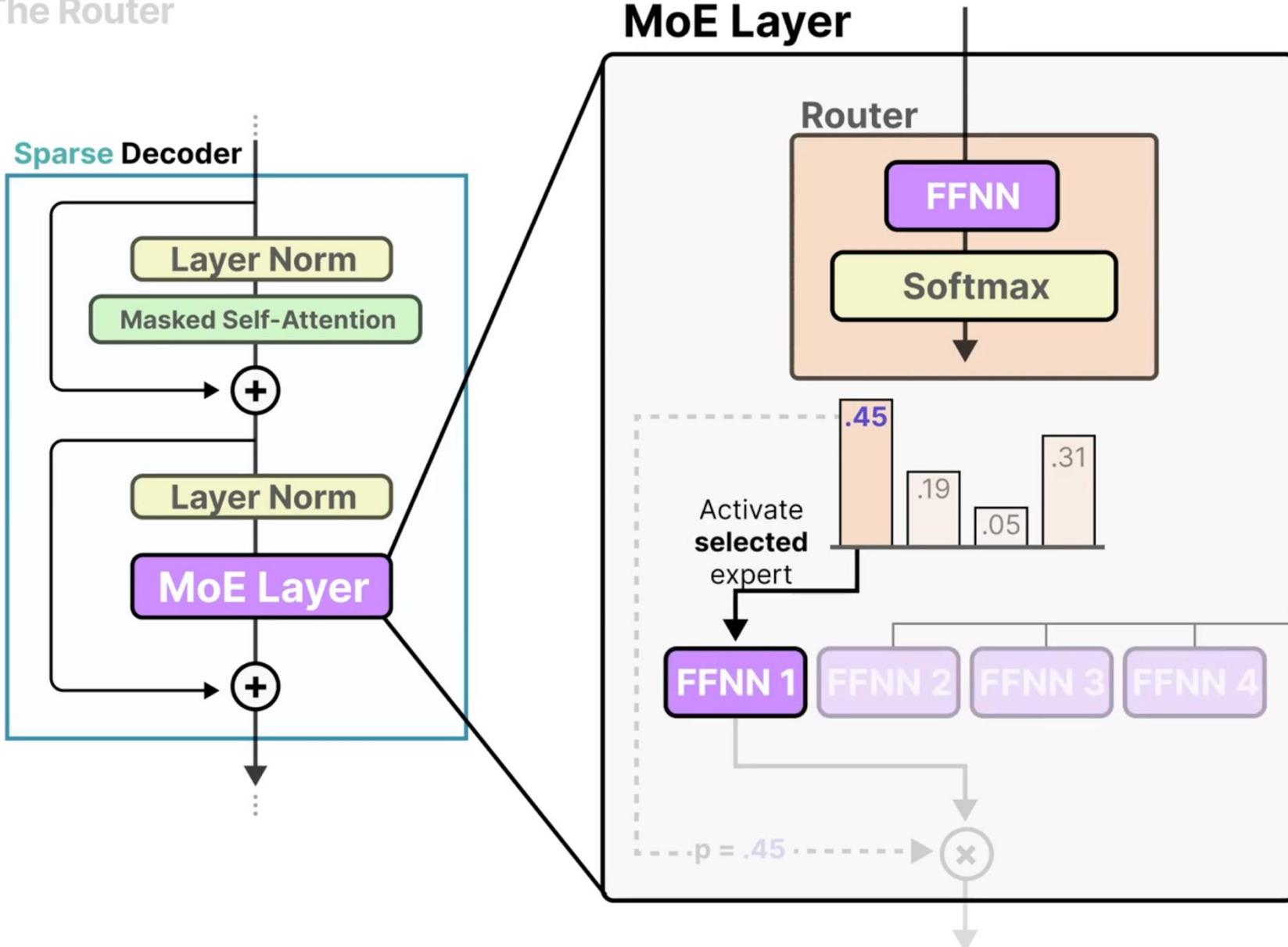
After the **FFNN** in the **router**, we see a **softmax function**...



The Router



The Router

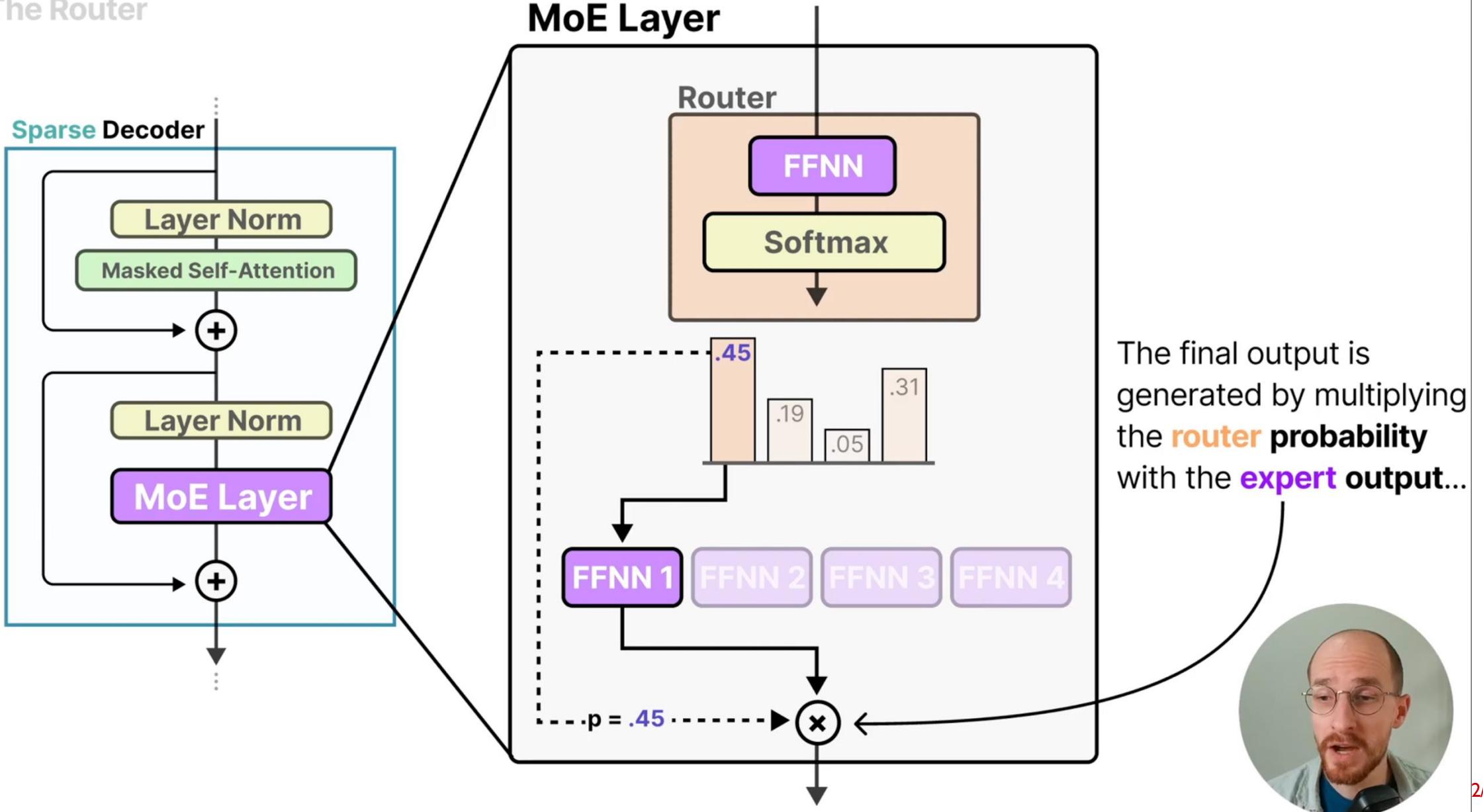


... that are used to
select and **activate**
the best **expert**.

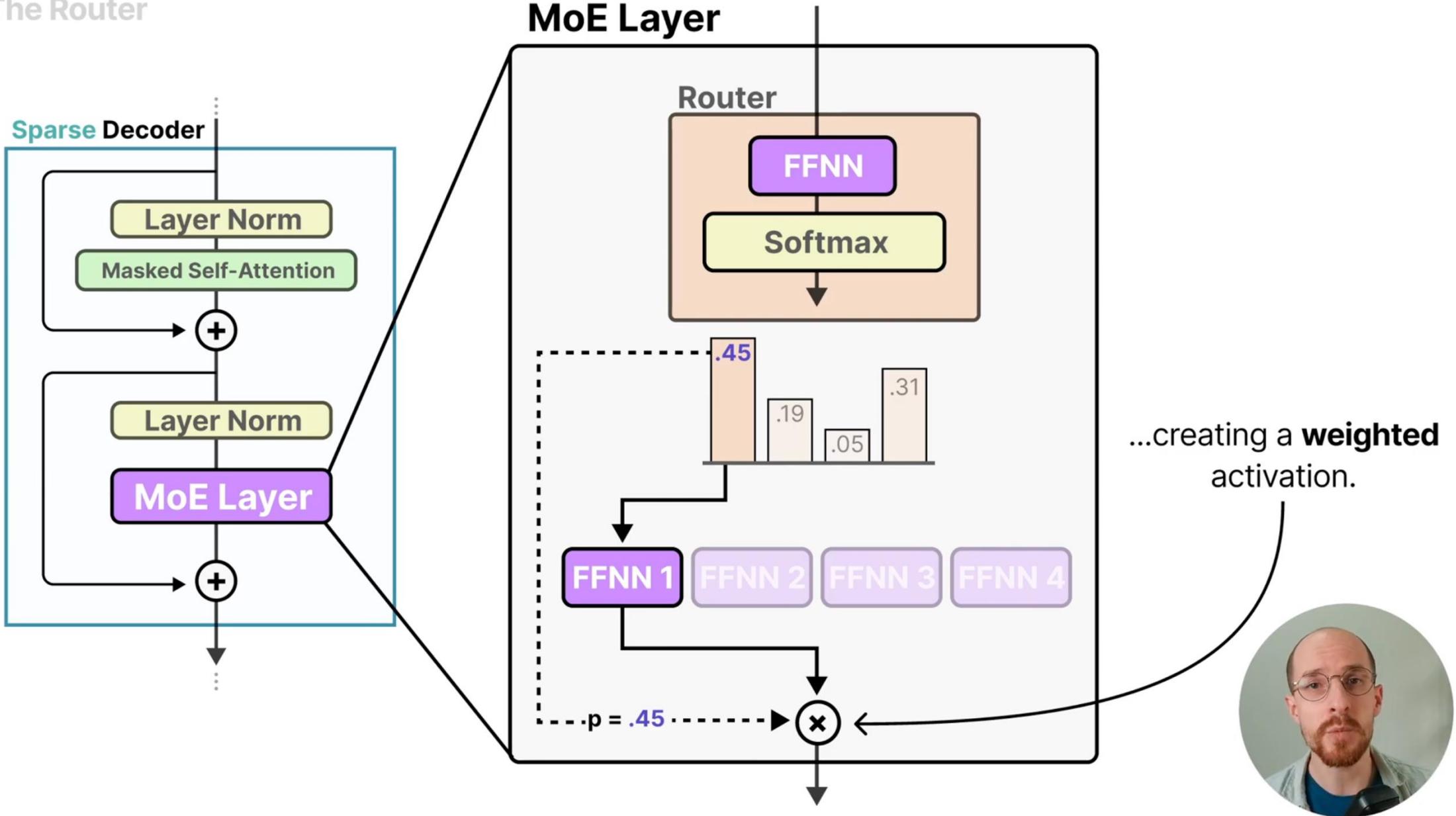
Not activated



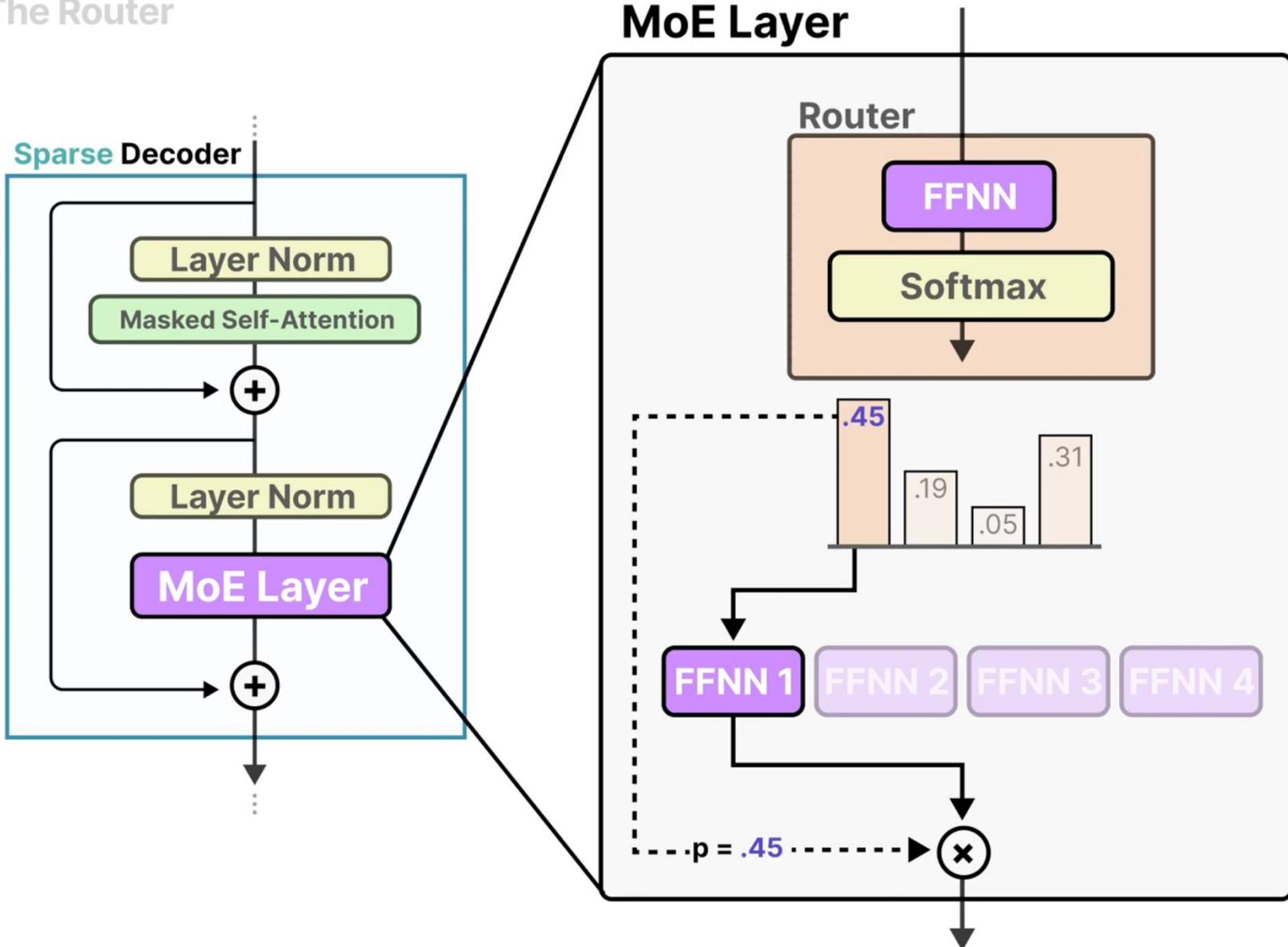
The Router



The Router



The Router



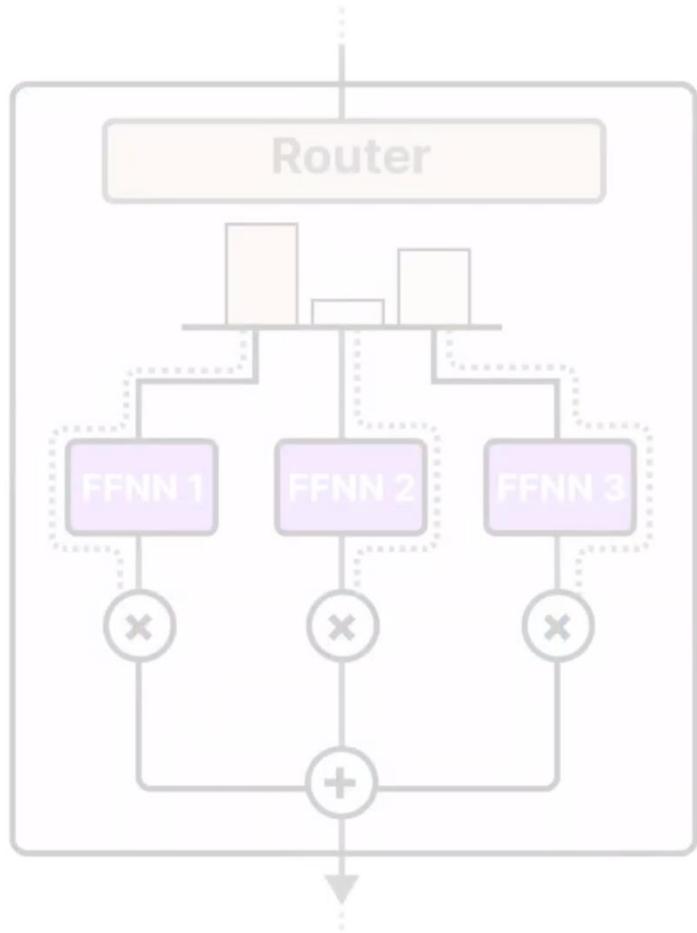
This entire architecture is therefore nothing more than multiple **FFNNs** and a **router** selecting the best(s) expert(s).



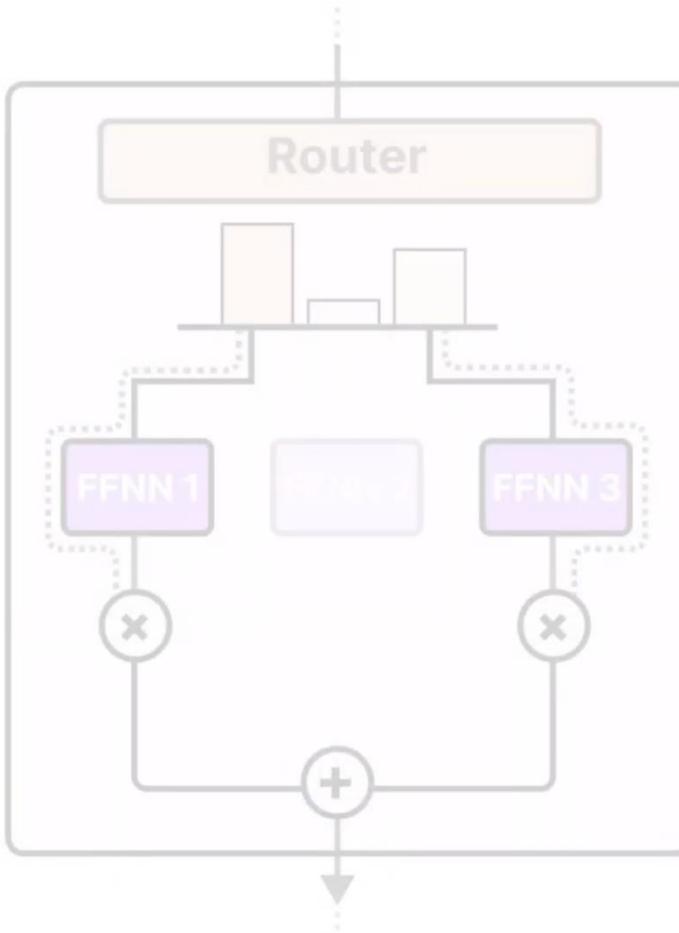
03

Architecture: 模型结构

Dense versus Sparse MoE



Dense MoE

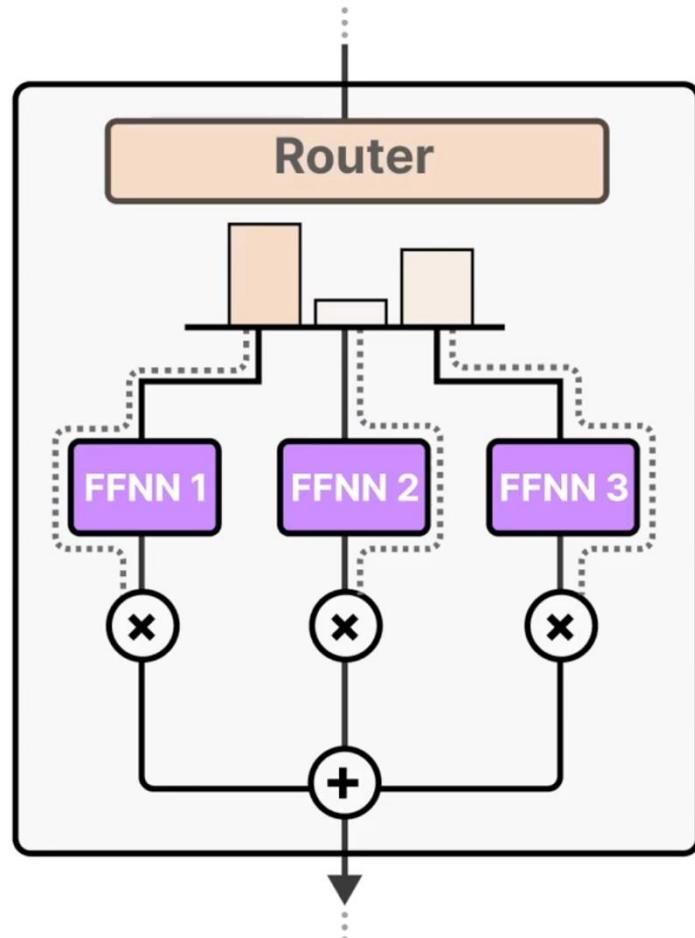


Sparse MoE

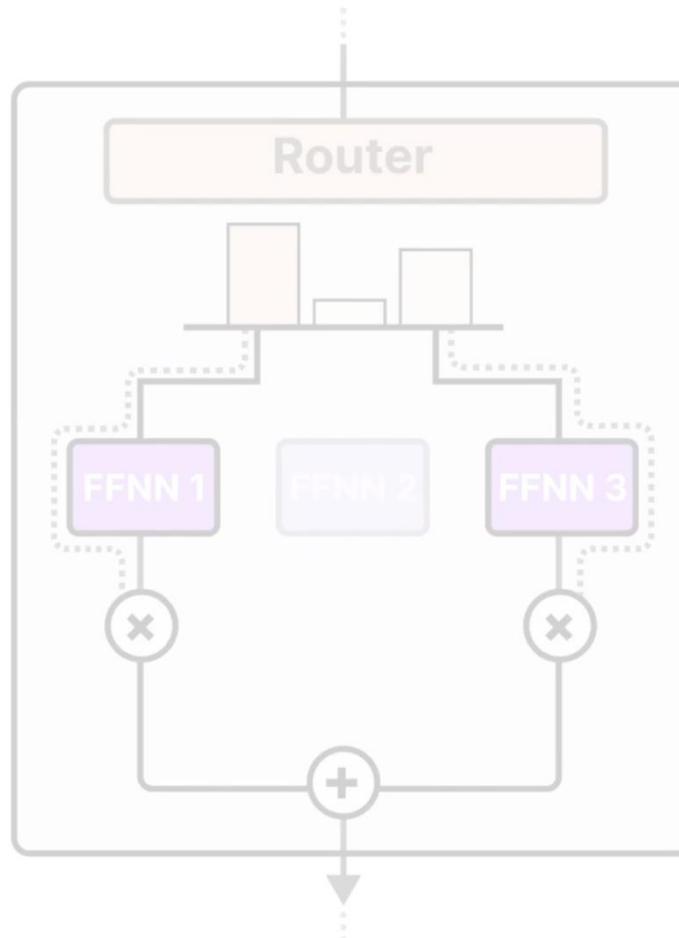
A given MoE layer
comes in two sizes...



Dense versus Sparse MoE



Dense MoE

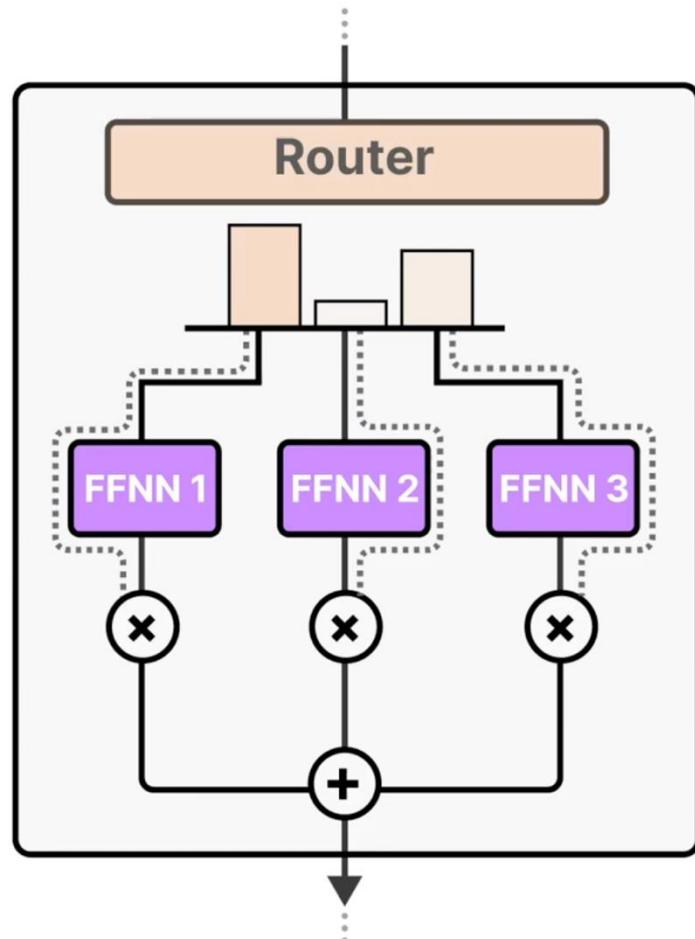


Sparse MoE

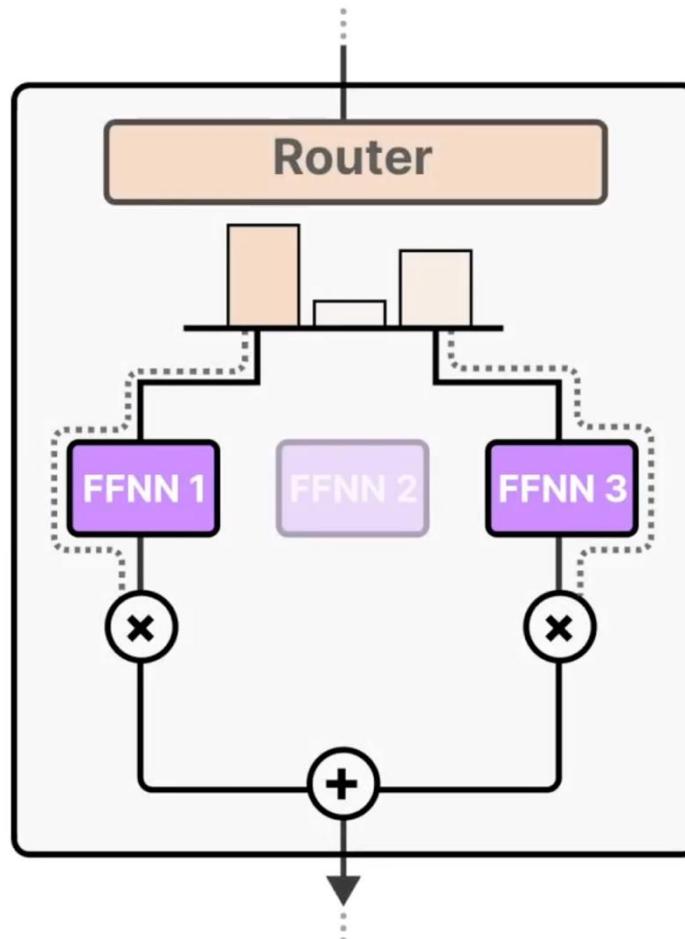
A given MoE layer comes in two sizes, either a **dense**...



Dense versus Sparse MoE



Dense MoE

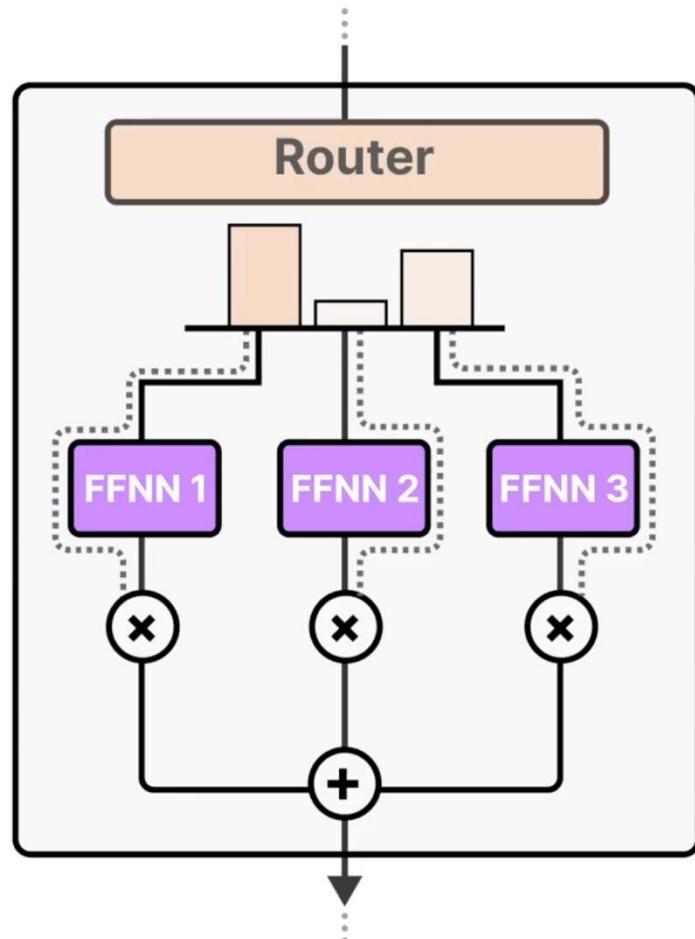


Sparse MoE

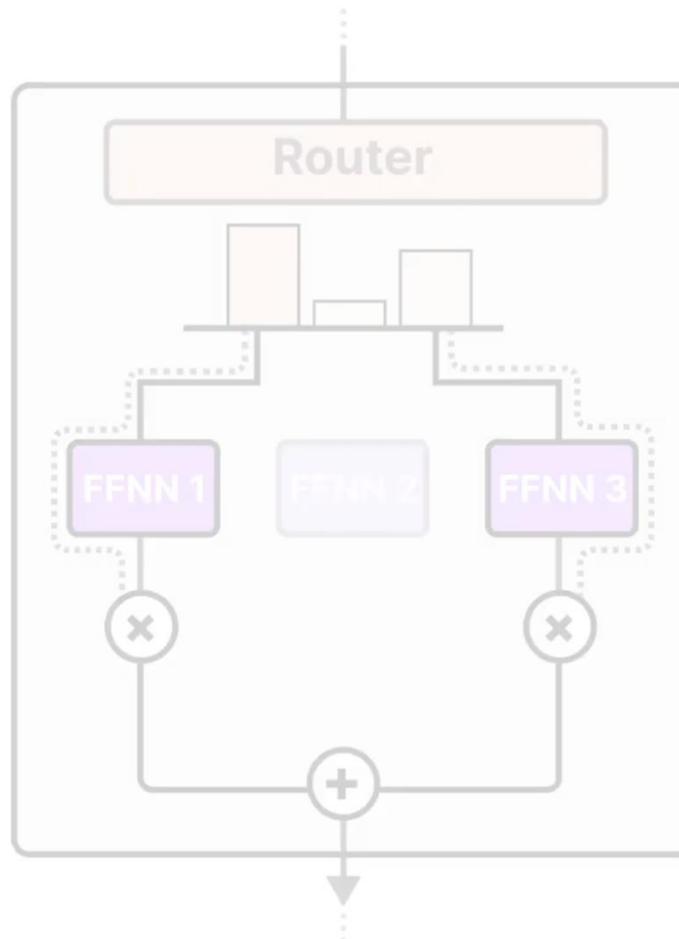
A given MoE layer comes in two sizes, either a **dense** or a **sparse** Mixture of Experts.



Dense versus Sparse MoE



Dense MoE

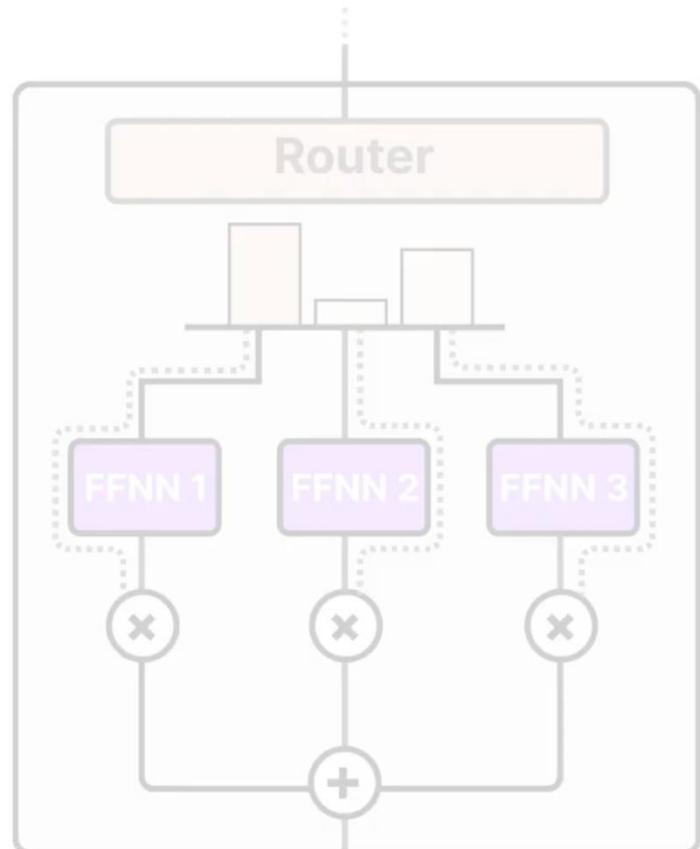


Sparse MoE

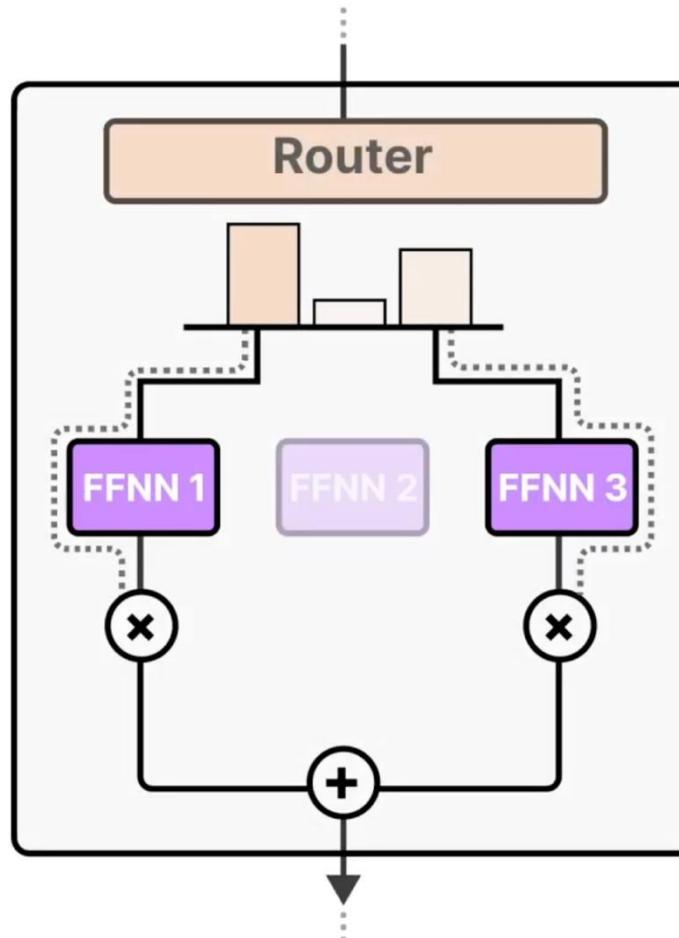
A **Dense MoE** will distribute the tokens across all experts...



Dense versus Sparse MoE



Dense MoE

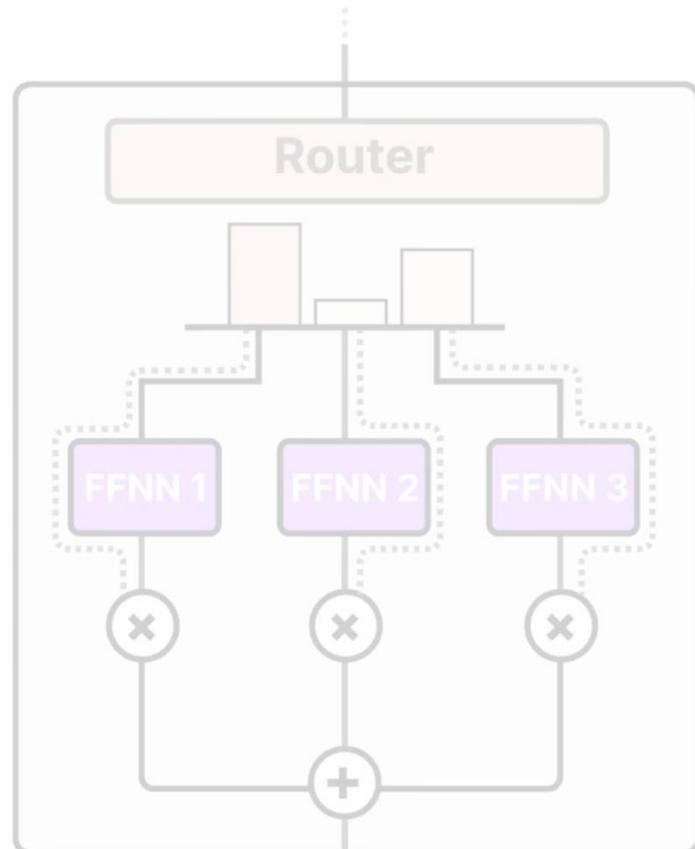


Sparse MoE

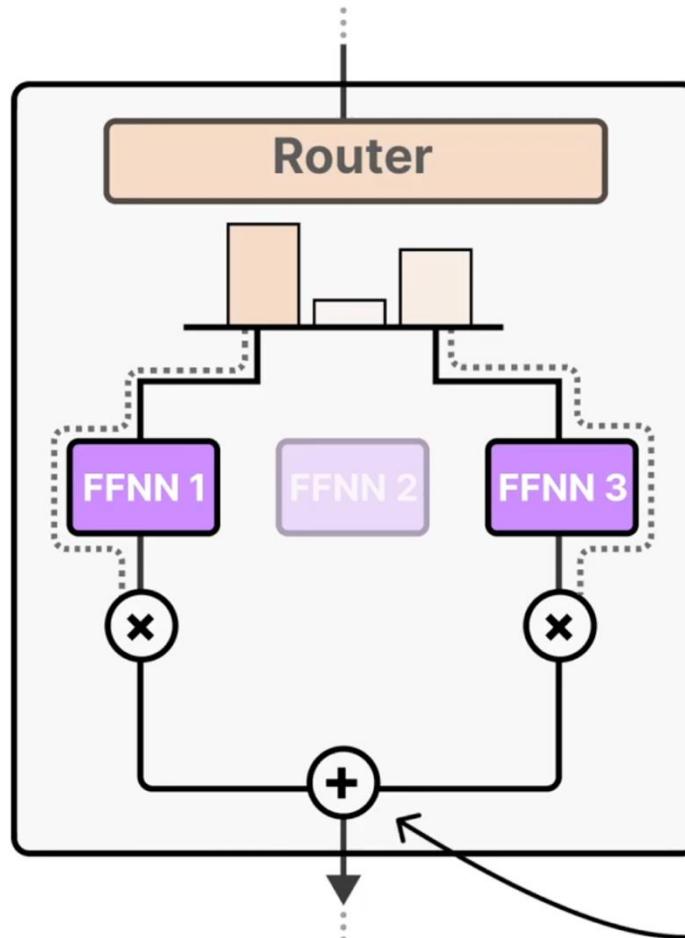
... whereas a **Sparse MoE** will only select a few experts.



Dense versus Sparse MoE



Dense MoE

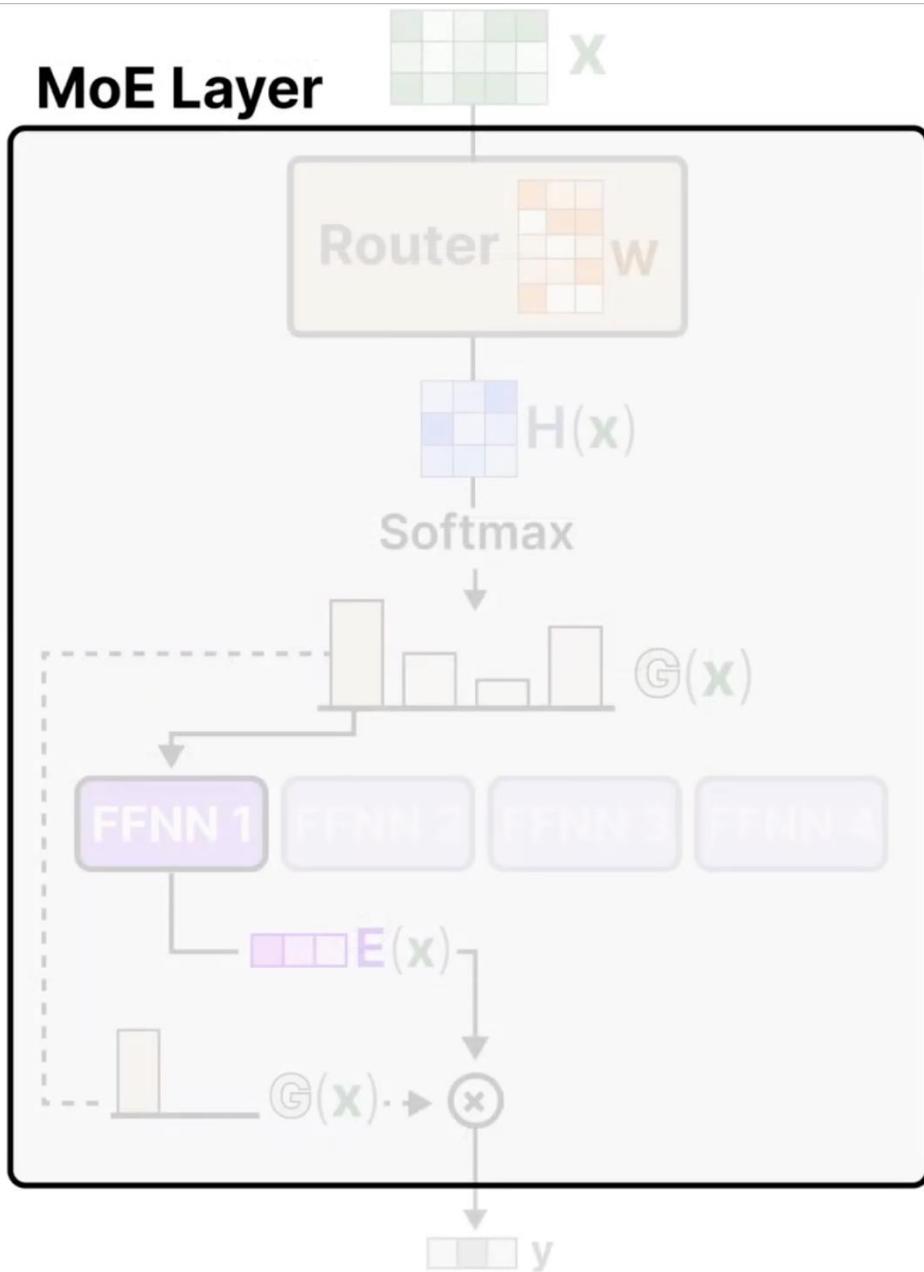


Sparse MoE

When we have multiple experts selected, their weighted outputs get **aggregated**.



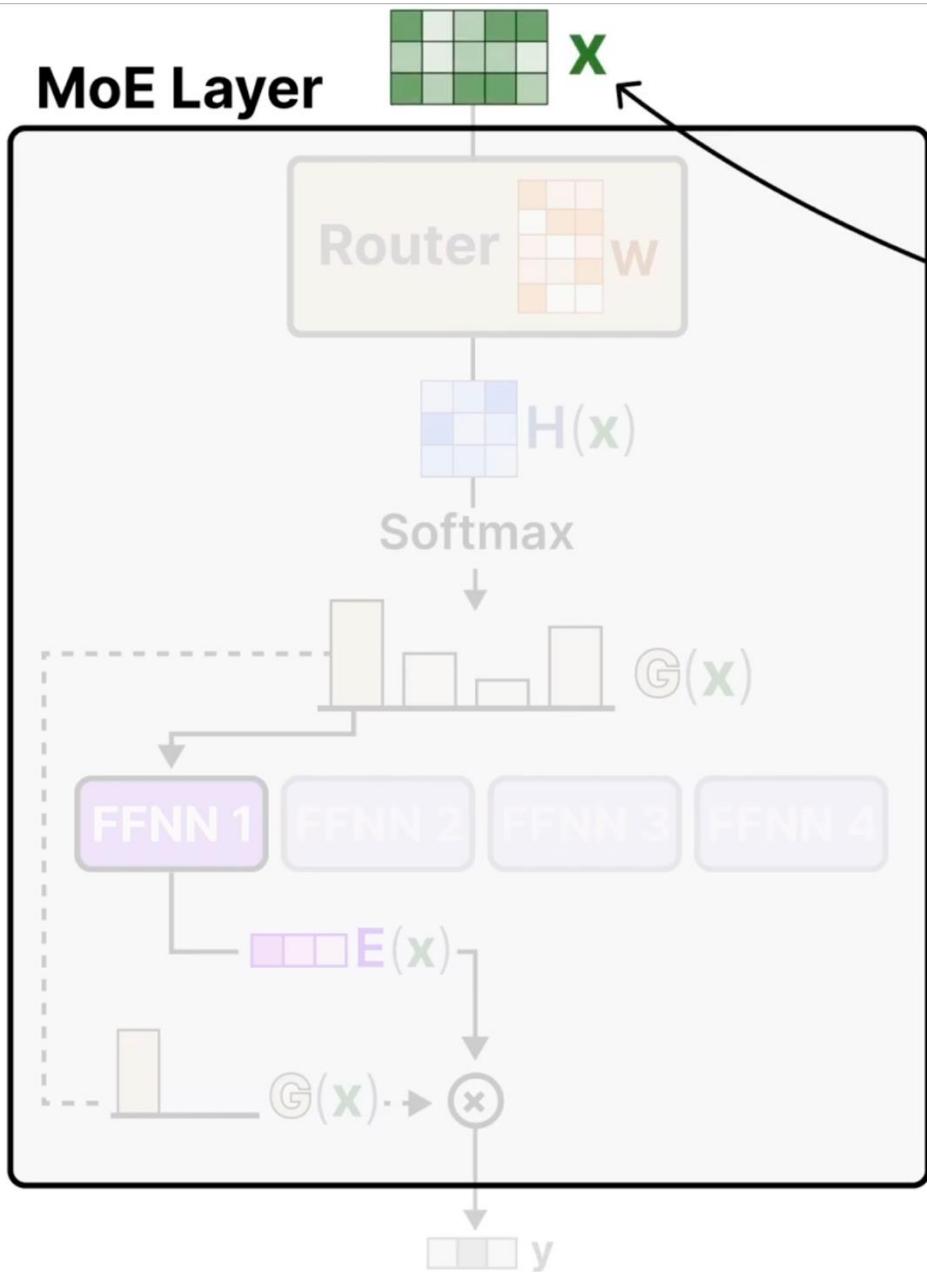
MoE Layer



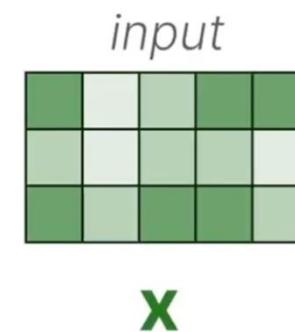
Let's explore how data flows through the **MoE Layer**.

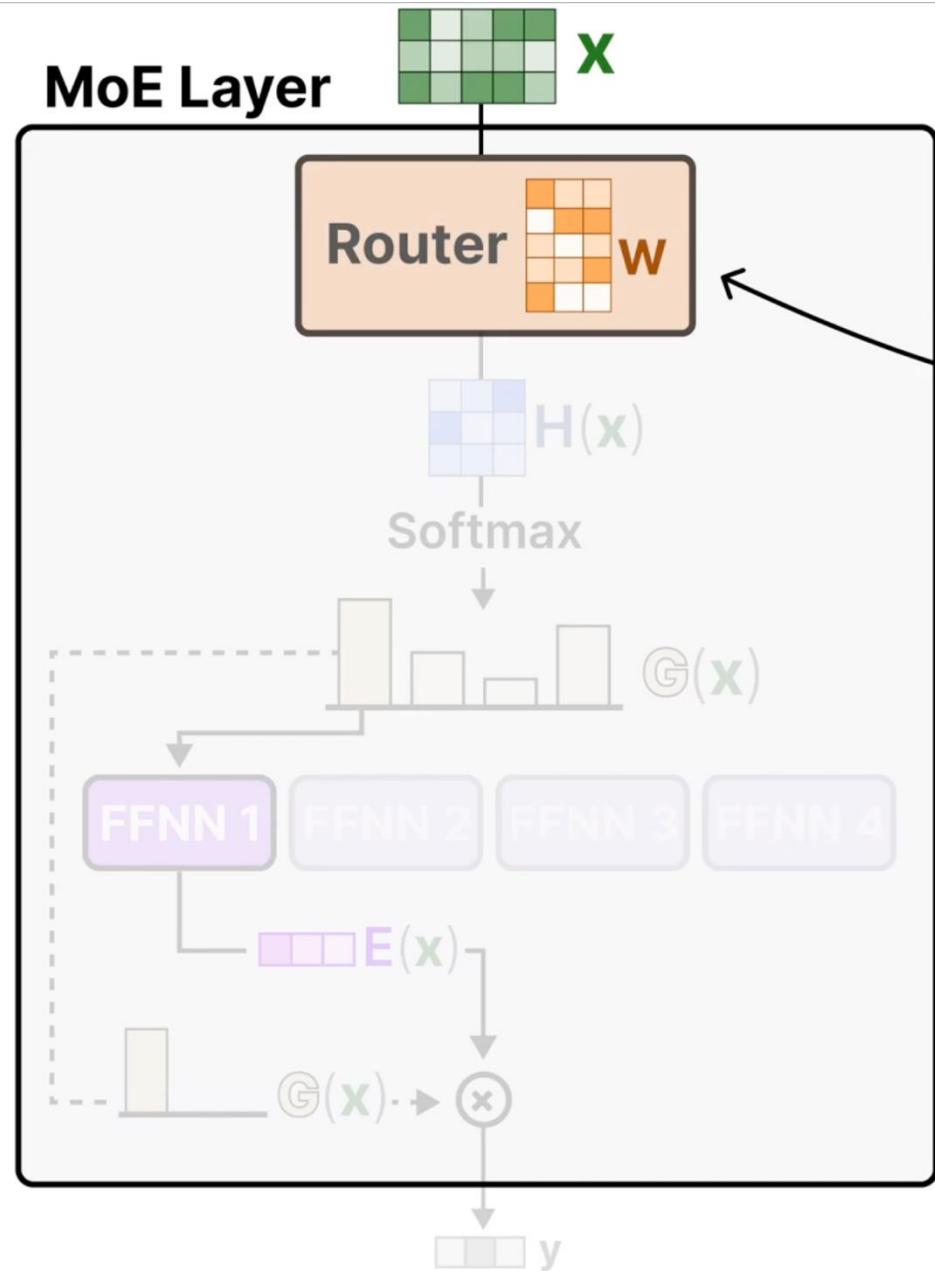


Selection of Experts



In its most basic form, we multiply the **input (x)** ...





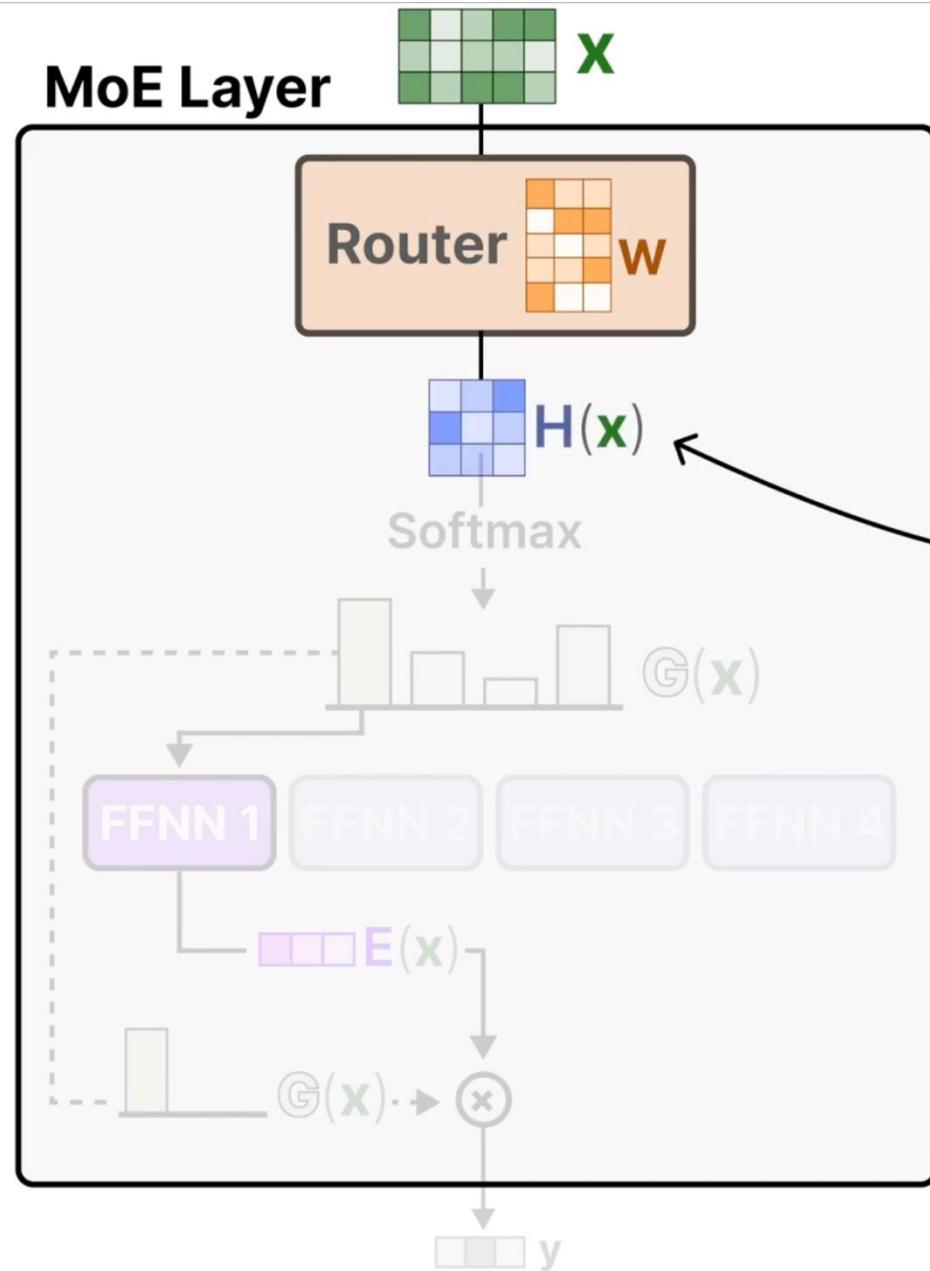
... by the **router weight matrix (W)** ...

Selection of Experts

router weights

$$\begin{array}{c}
 \text{input} \\
 \begin{matrix} \textcolor{green}{\square} & \textcolor{white}{\square} & \textcolor{green}{\square} & \textcolor{green}{\square} & \textcolor{green}{\square} \\ \textcolor{green}{\square} & \textcolor{white}{\square} & \textcolor{green}{\square} & \textcolor{green}{\square} & \textcolor{green}{\square} \\ \textcolor{green}{\square} & \textcolor{white}{\square} & \textcolor{green}{\square} & \textcolor{green}{\square} & \textcolor{green}{\square} \\ \textcolor{orange}{\square} & \textcolor{white}{\square} & \textcolor{orange}{\square} & \textcolor{orange}{\square} & \textcolor{orange}{\square} \end{matrix} \\
 \mathbf{X} \quad * \quad \mathbf{W}
 \end{array}$$



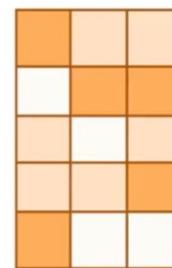


... to create the **output** of the router $H(x)$.

$$H(x) = X * W$$

output input
 $H(x)$ X *

router
weights



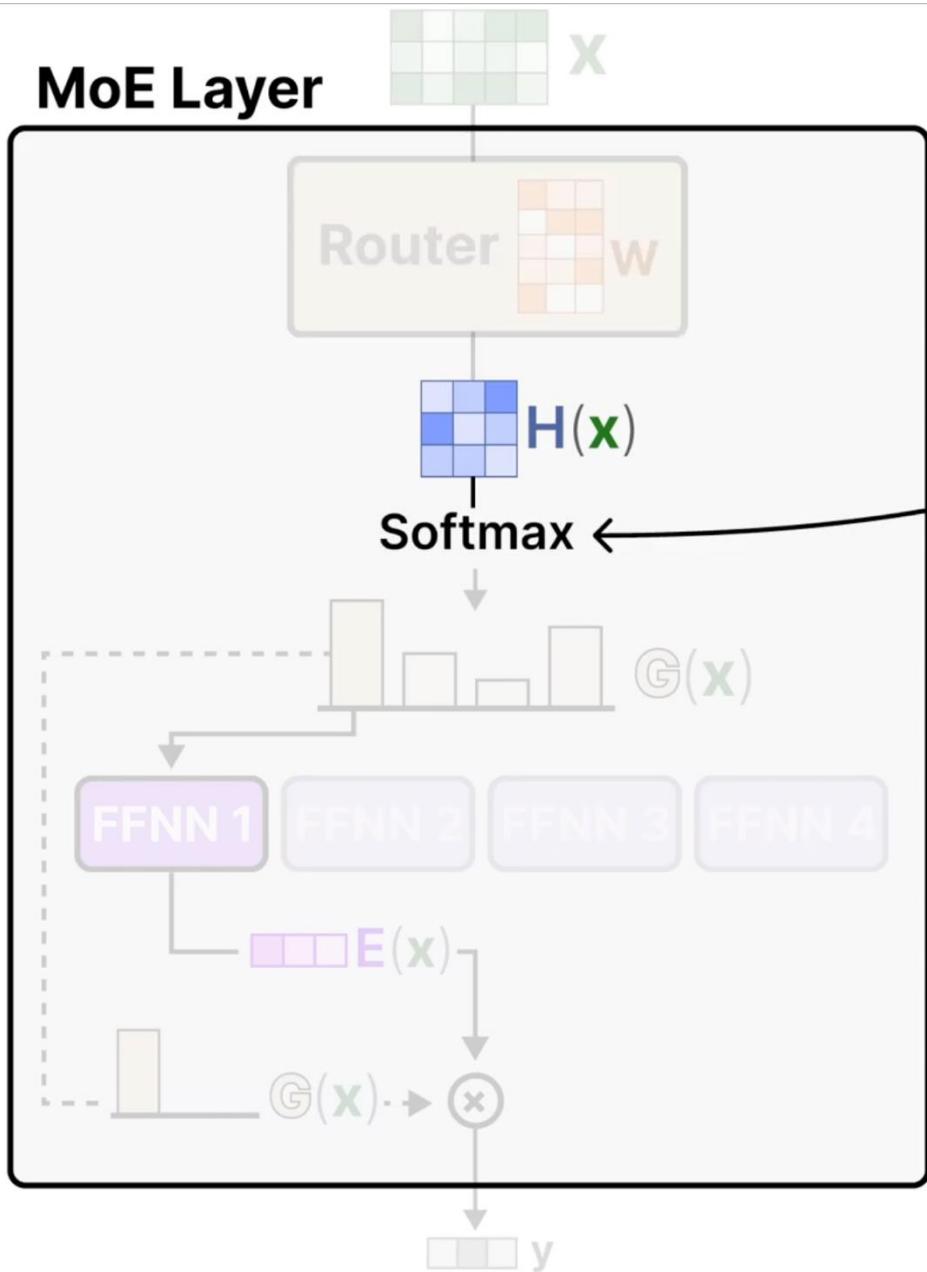
W



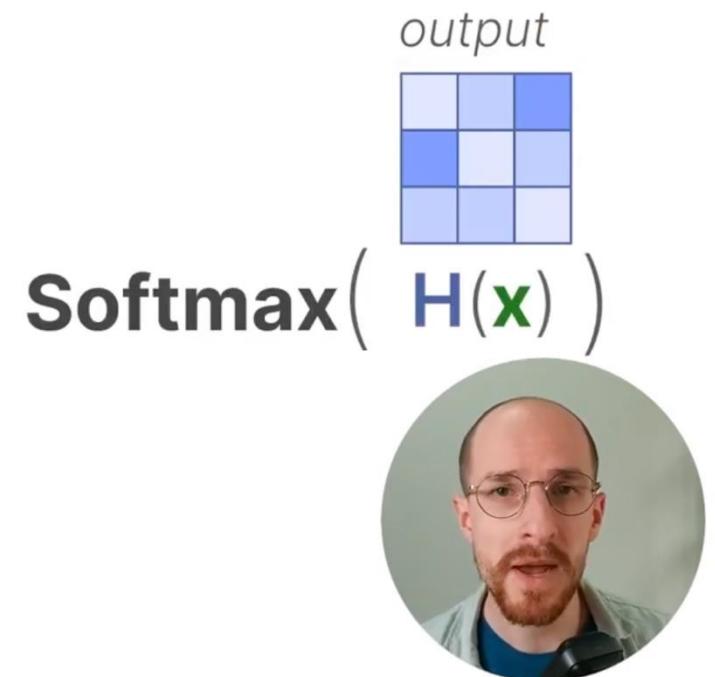
Selection of Experts

Selection of Experts

MoE Layer

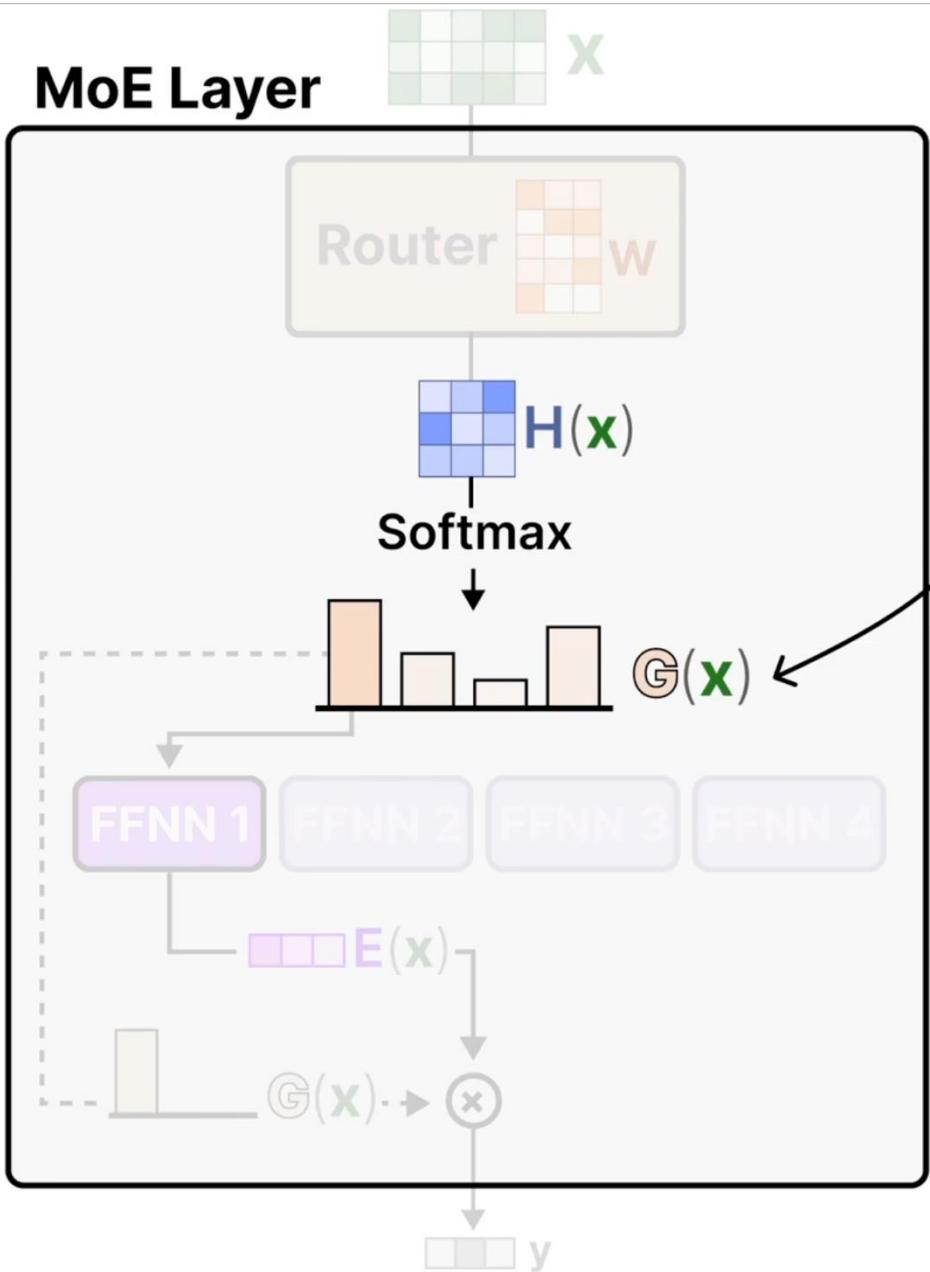


Then, the **softmax** of the output is taken...



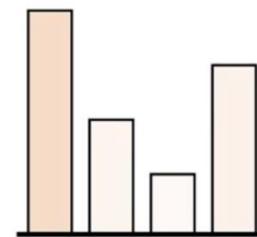
Selection of Experts

MoE Layer

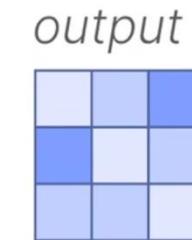


... to create **probabilities**, $G(x)$, one for each expert.

probability distribution per expert

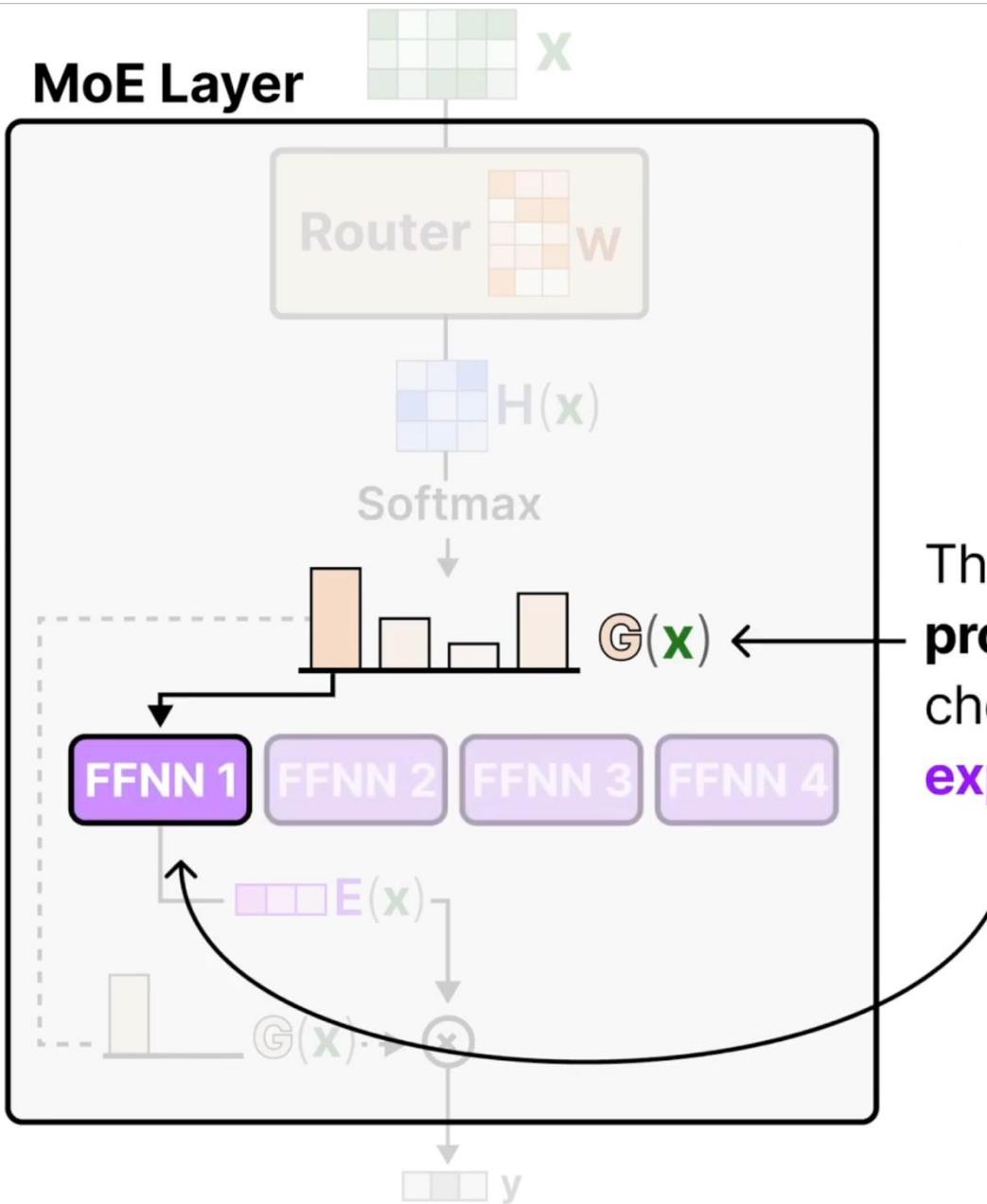


$$G(x) = \text{Softmax}(H(x))$$



Selection of Experts

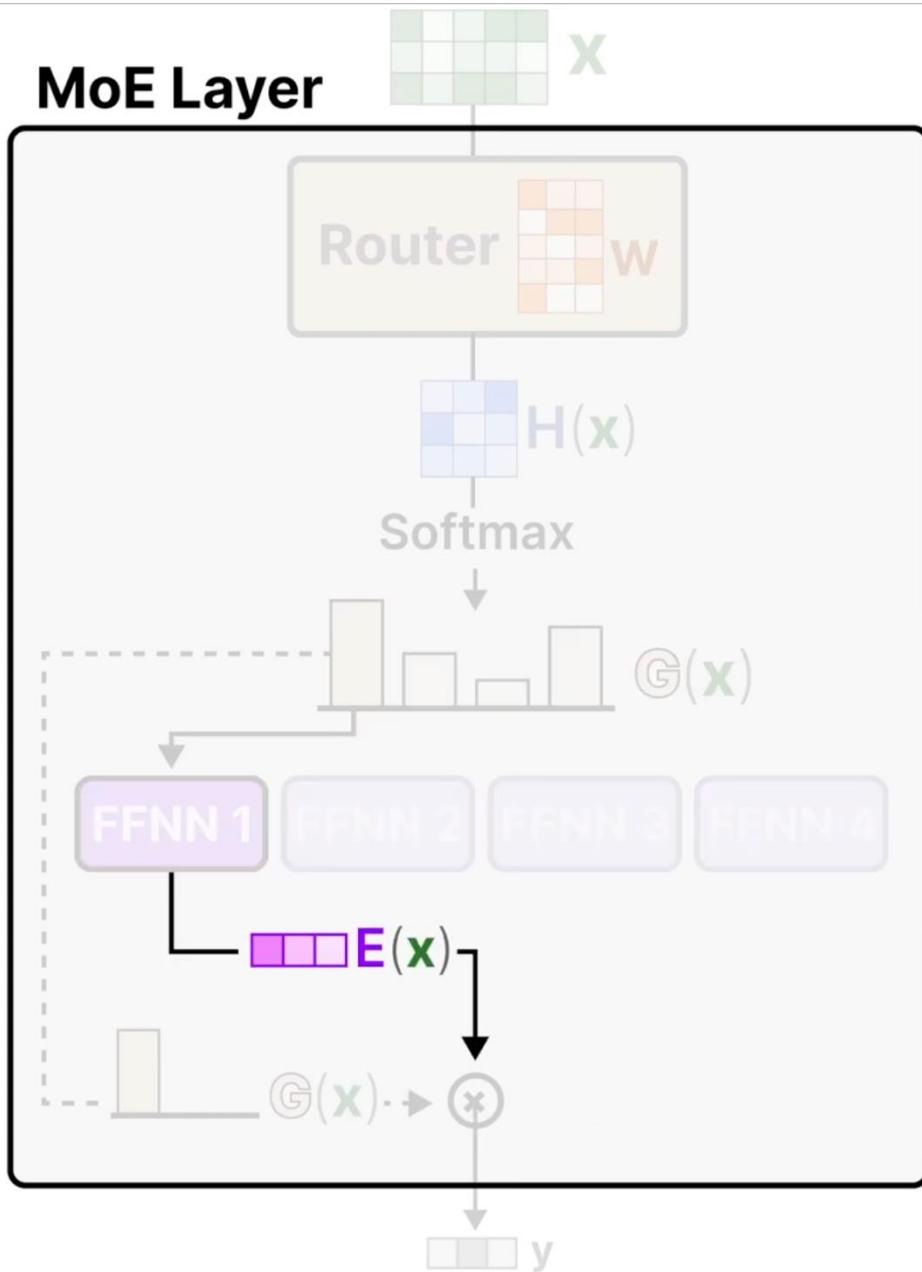
MoE Layer



The router uses this **probability distribution** to choose the best matching **expert**.

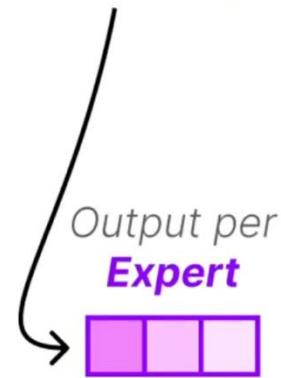


MoE Layer

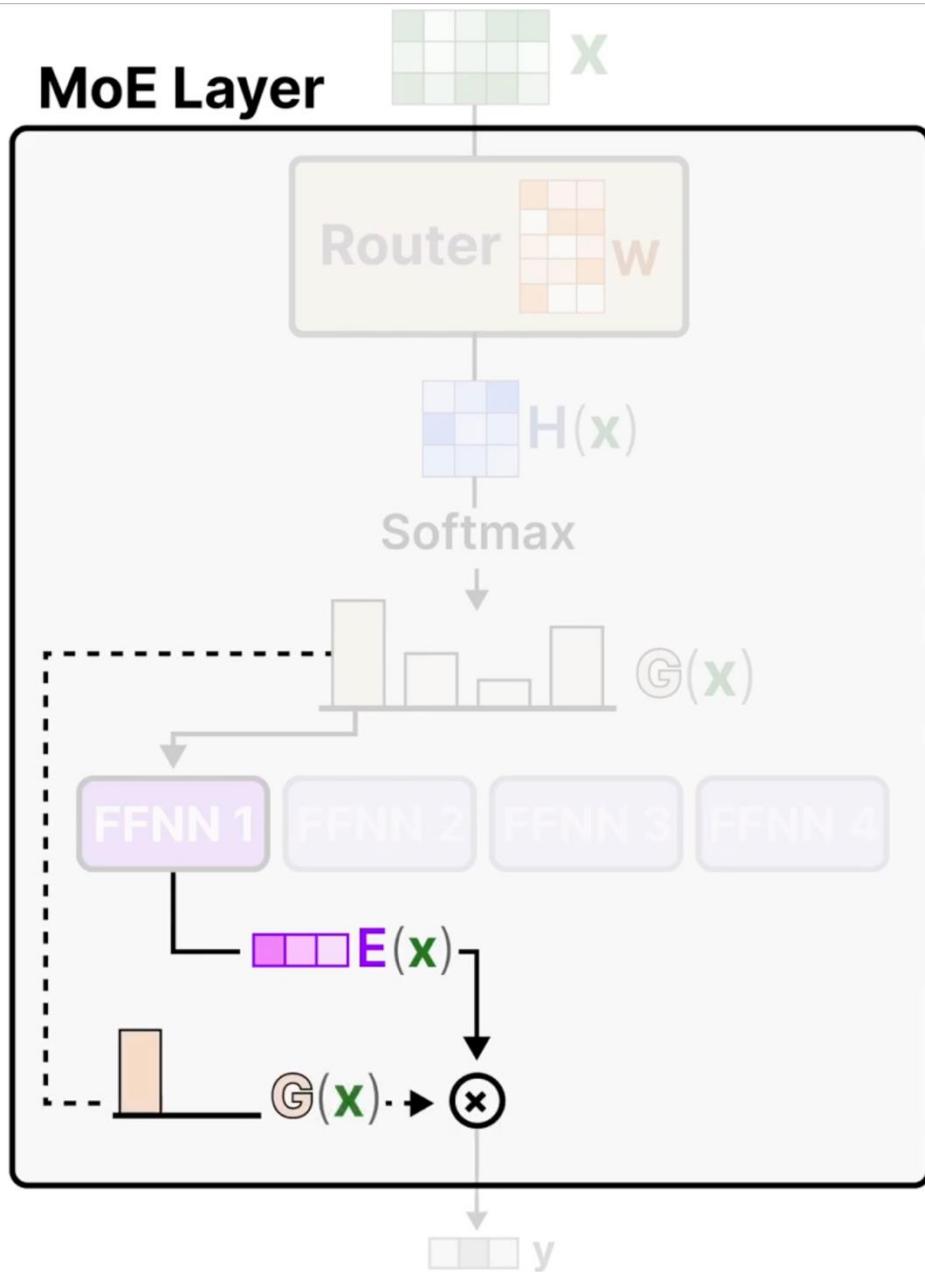


Selection of Experts

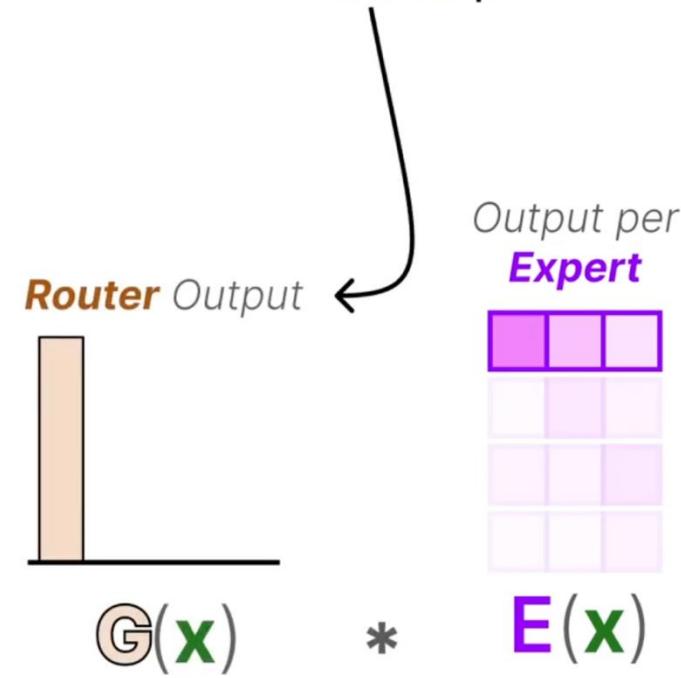
We then take the output per selected **expert**...



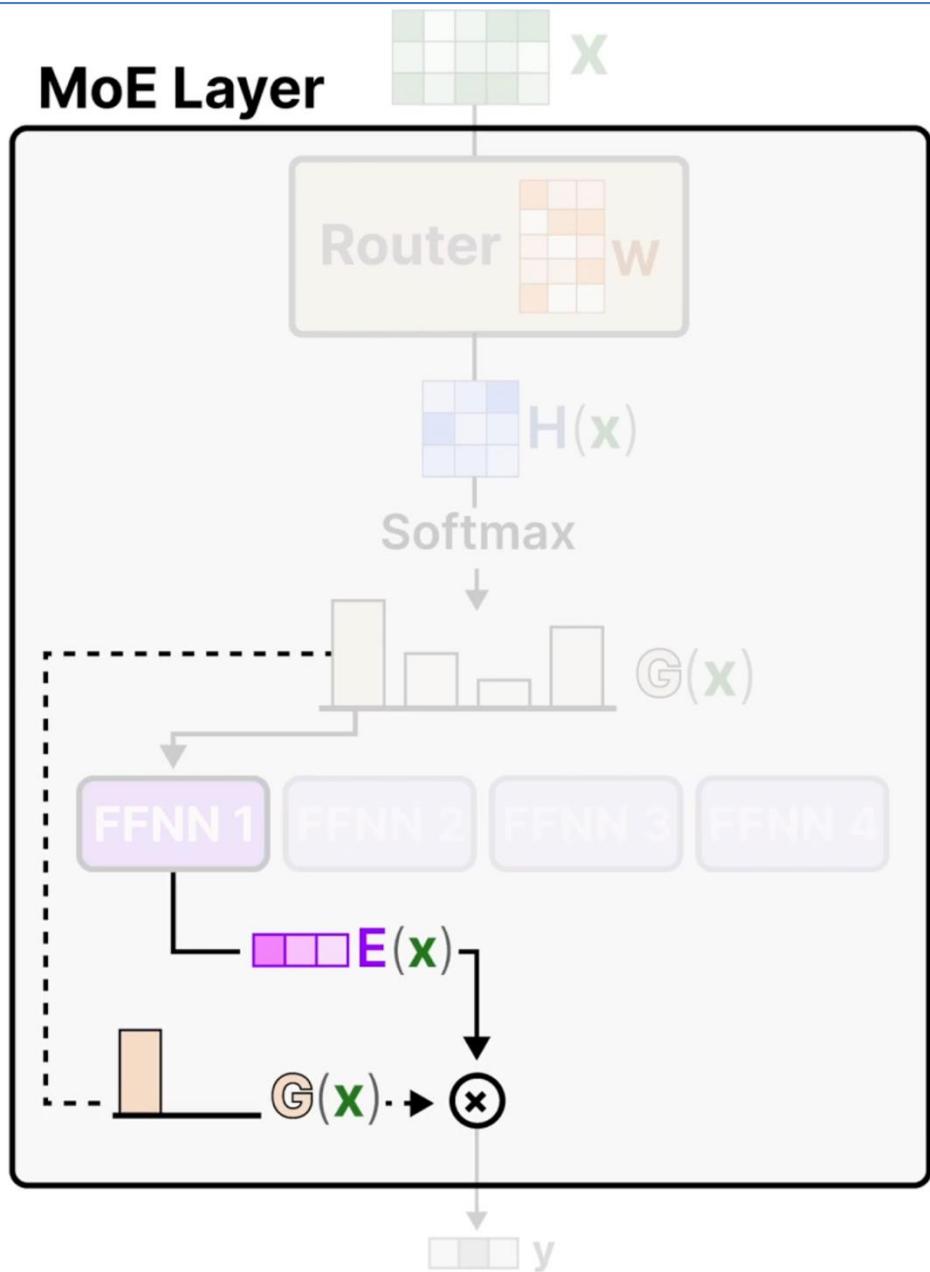
MoE Layer



... and multiply that with the **router** probabilities.



MoE Layer

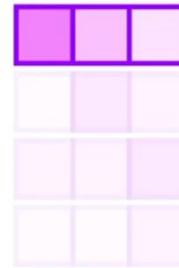


Selection of Experts

We do this for **every expert** selected.

$$\sum \left(\text{Router Output} \cdot G(\mathbf{x}) \right) * E(\mathbf{x})$$

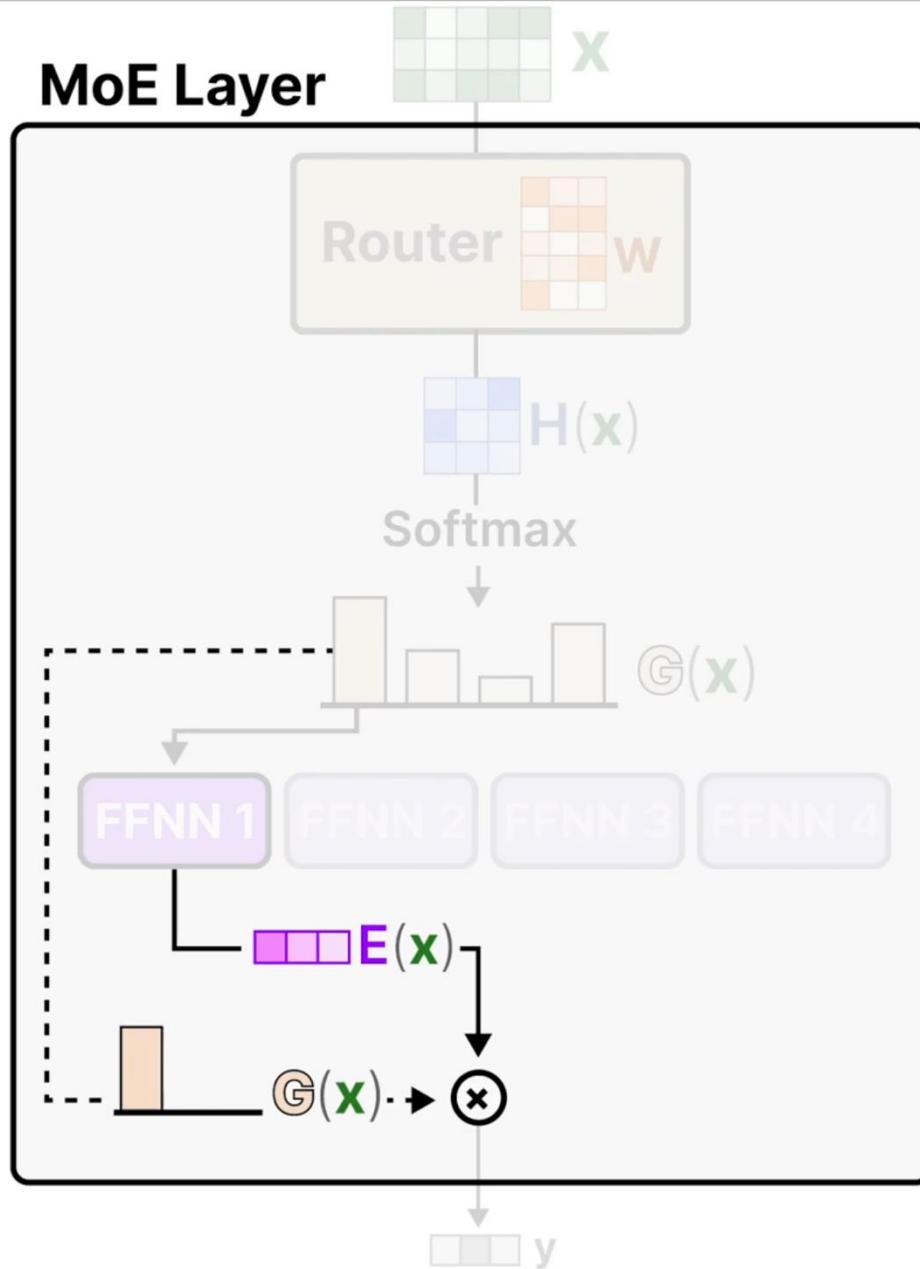
Output per
Expert



$E(\mathbf{x})$



MoE Layer



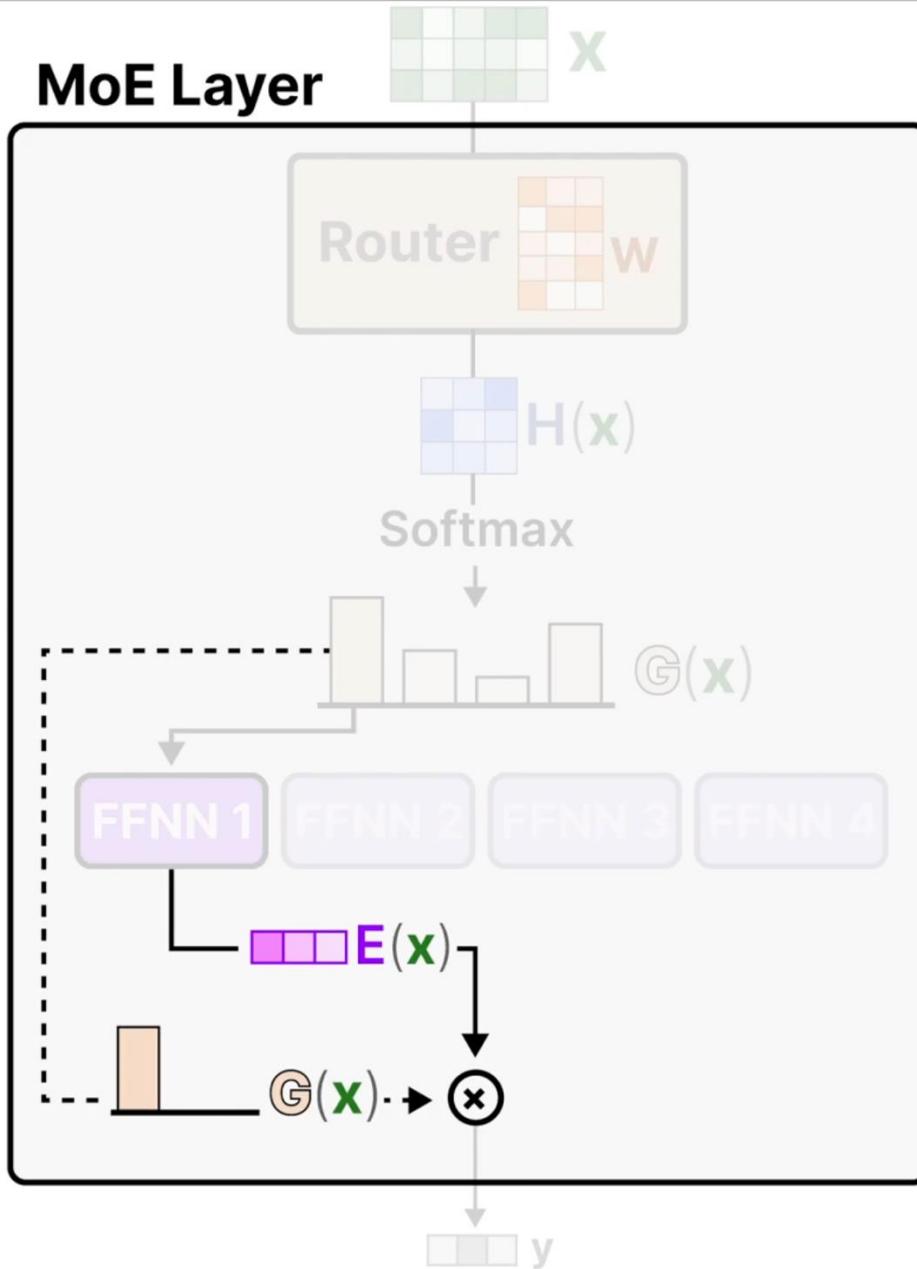
Since we choose only **one expert**, we are only doing this calculation once.

Router Output

$$\sum \left(\frac{G(x)}{G(x)} * E(x) \right)$$



MoE Layer



Selection of Experts

This creates our output, **one vector** for each **expert**.

Sparse MoE

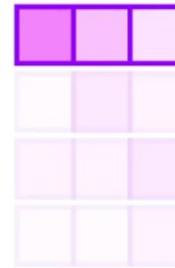
output



Router Output

$$y = \sum \left(G(x) \otimes E(x) \right)$$

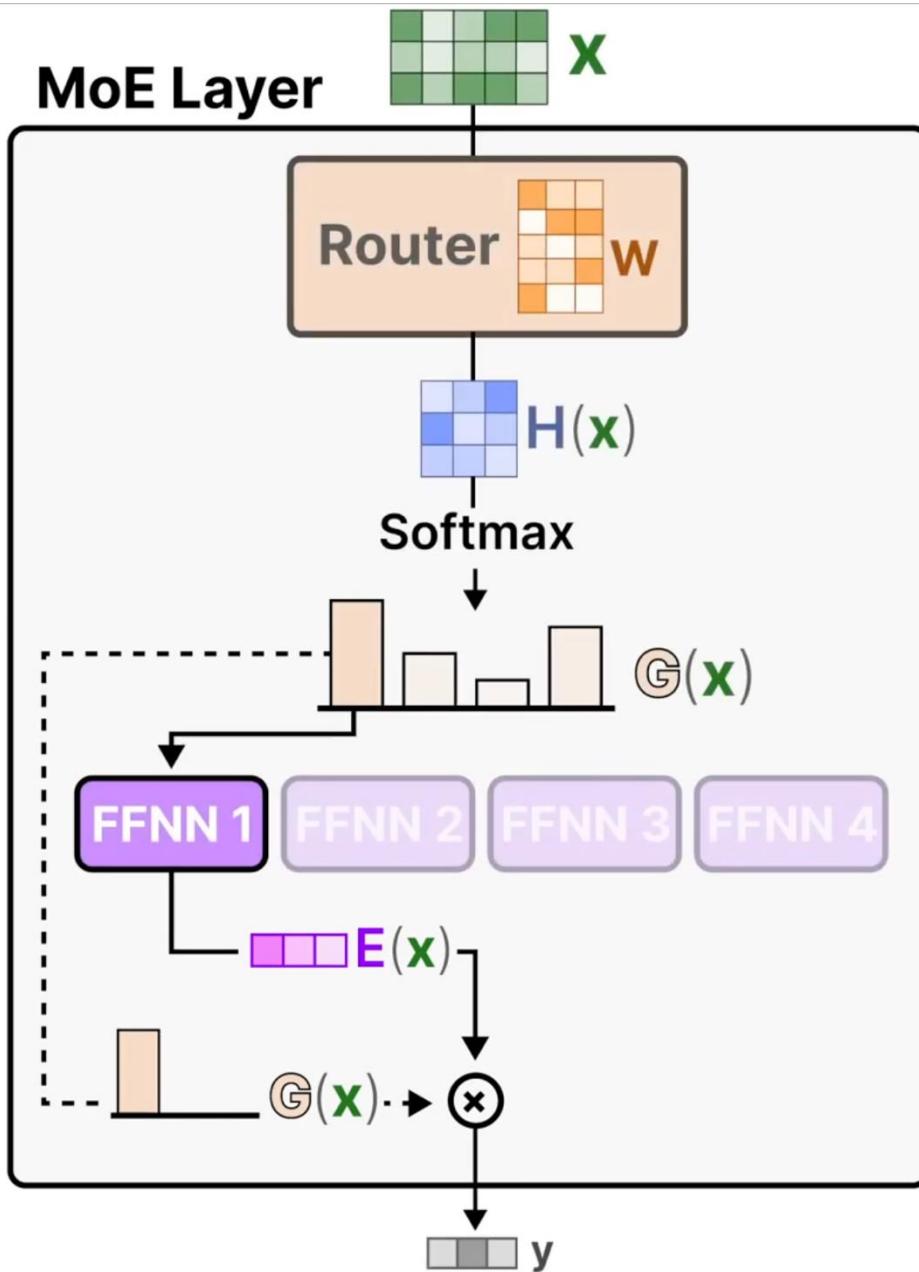
Output per
Expert



E(x)



Selection of Experts



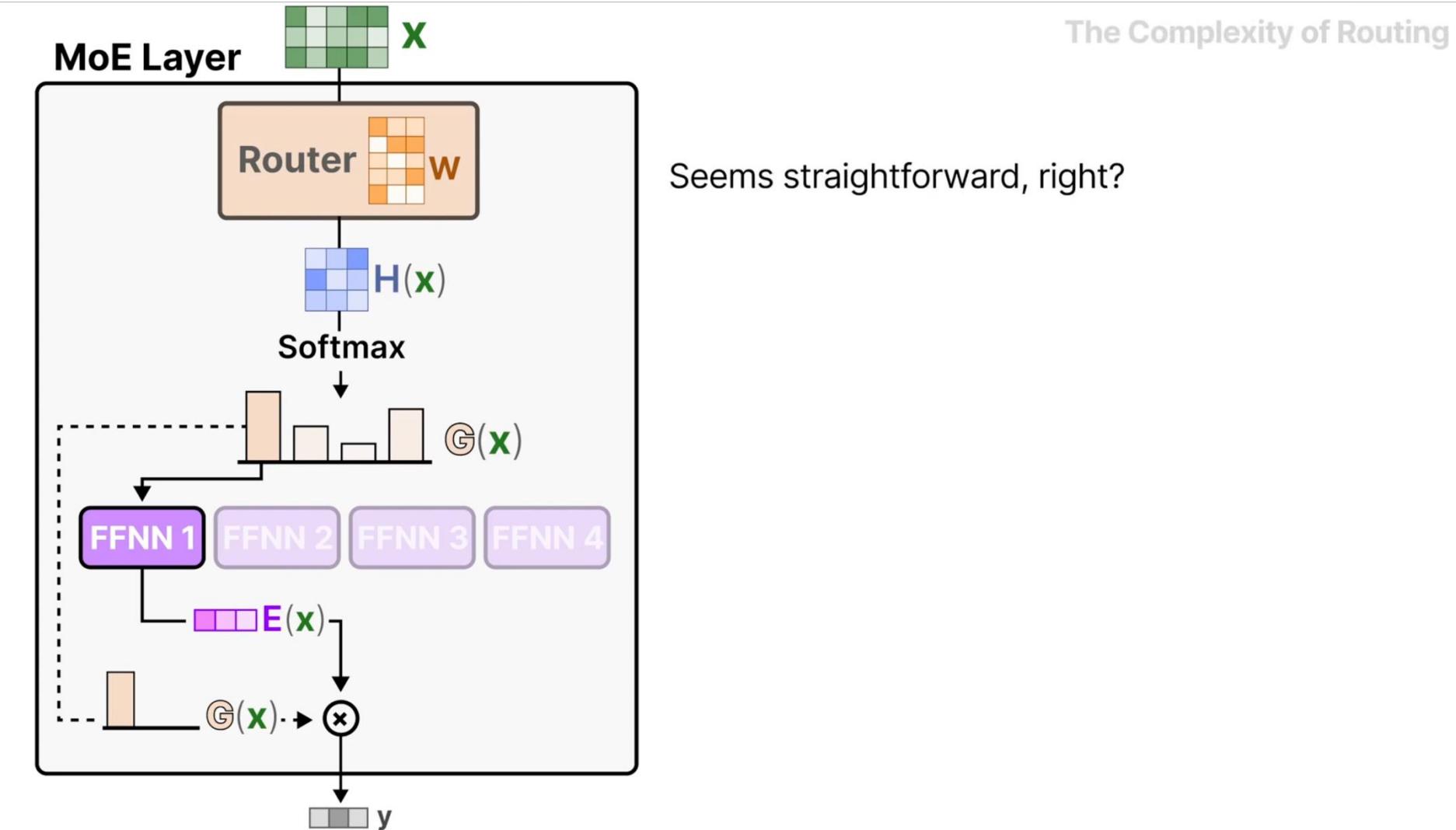
And that is how a typical **MoE layer** processes the data.



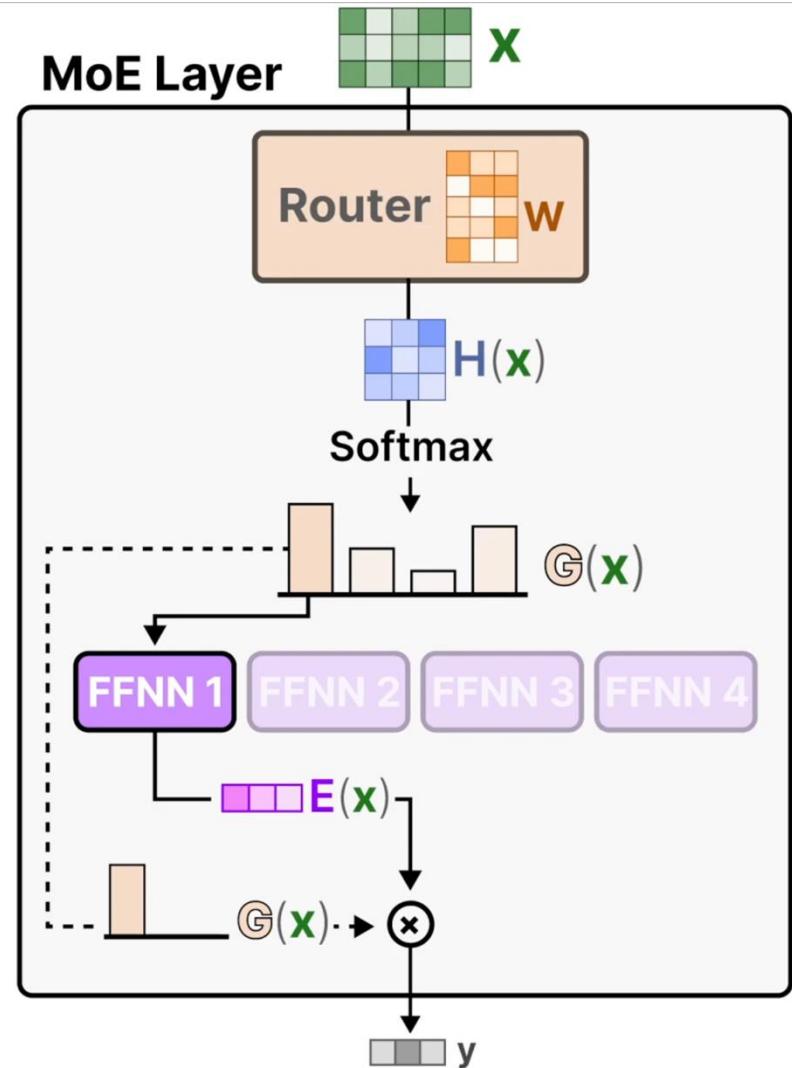
04

Load Balancing: 均衡负载

Load Balancing: 均衡负载



Load Balancing: 均衡负载

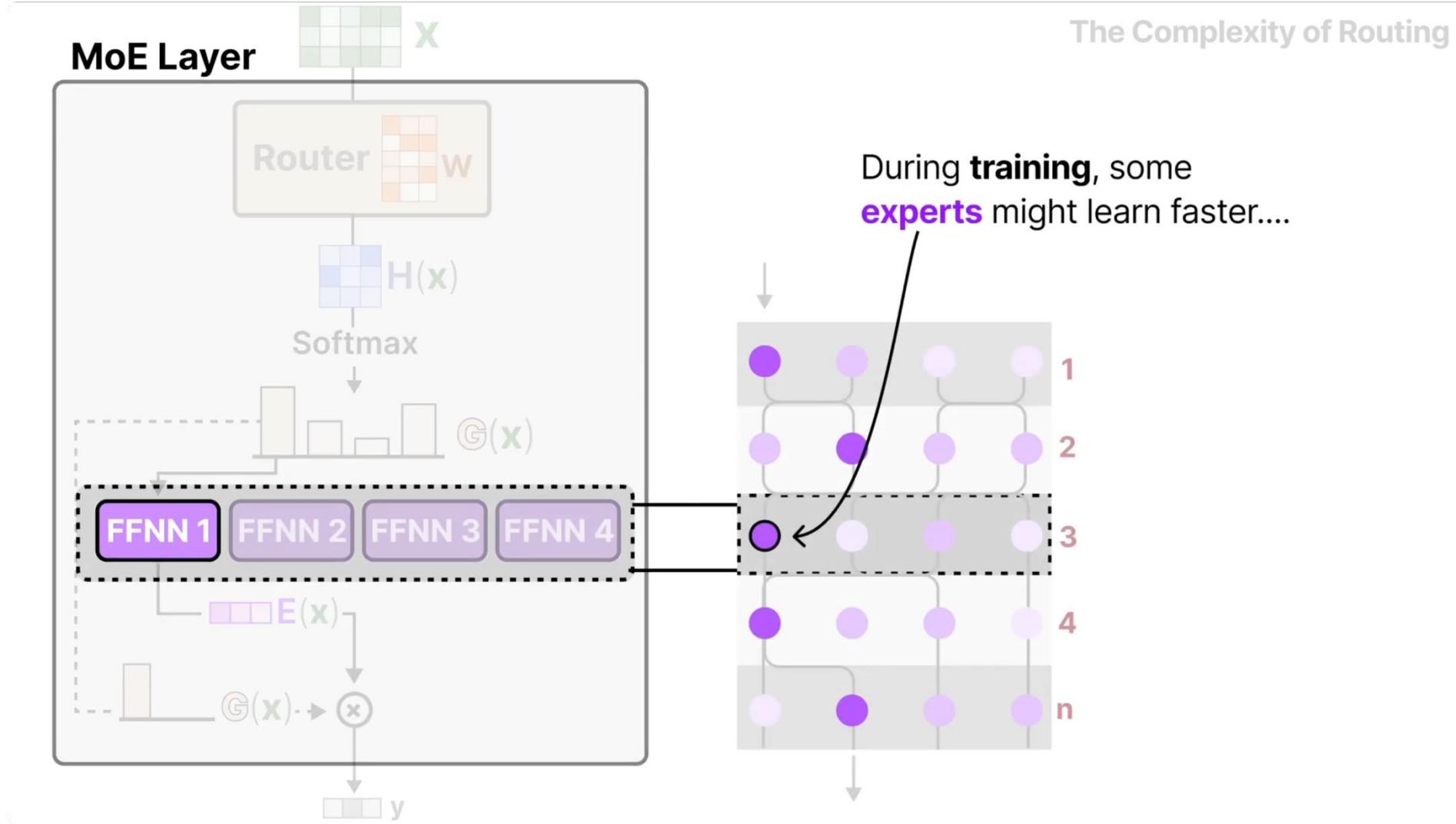


The Complexity of Routing

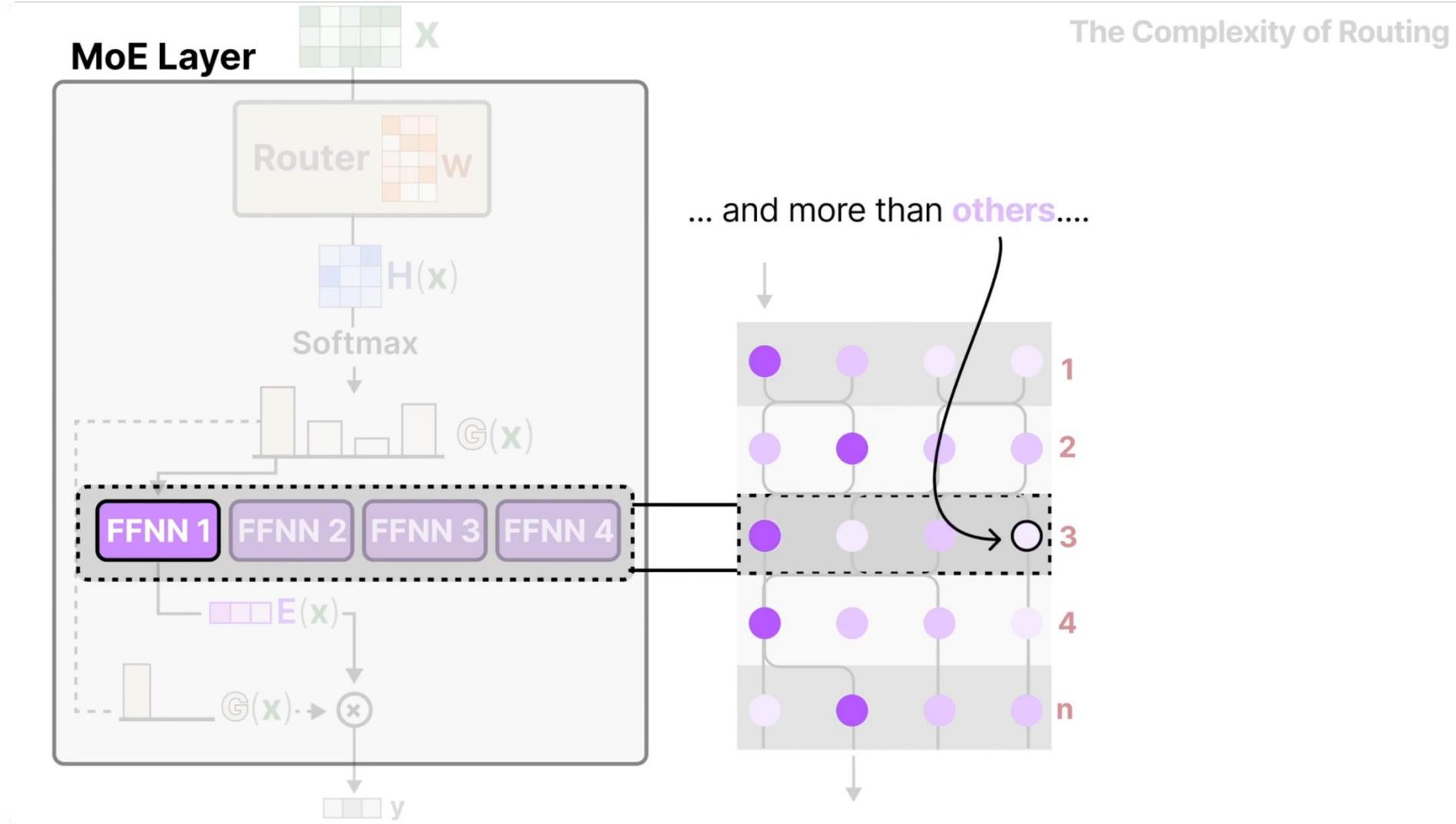
Seems straightforward, right?

Well, there is one big
disadvantage...

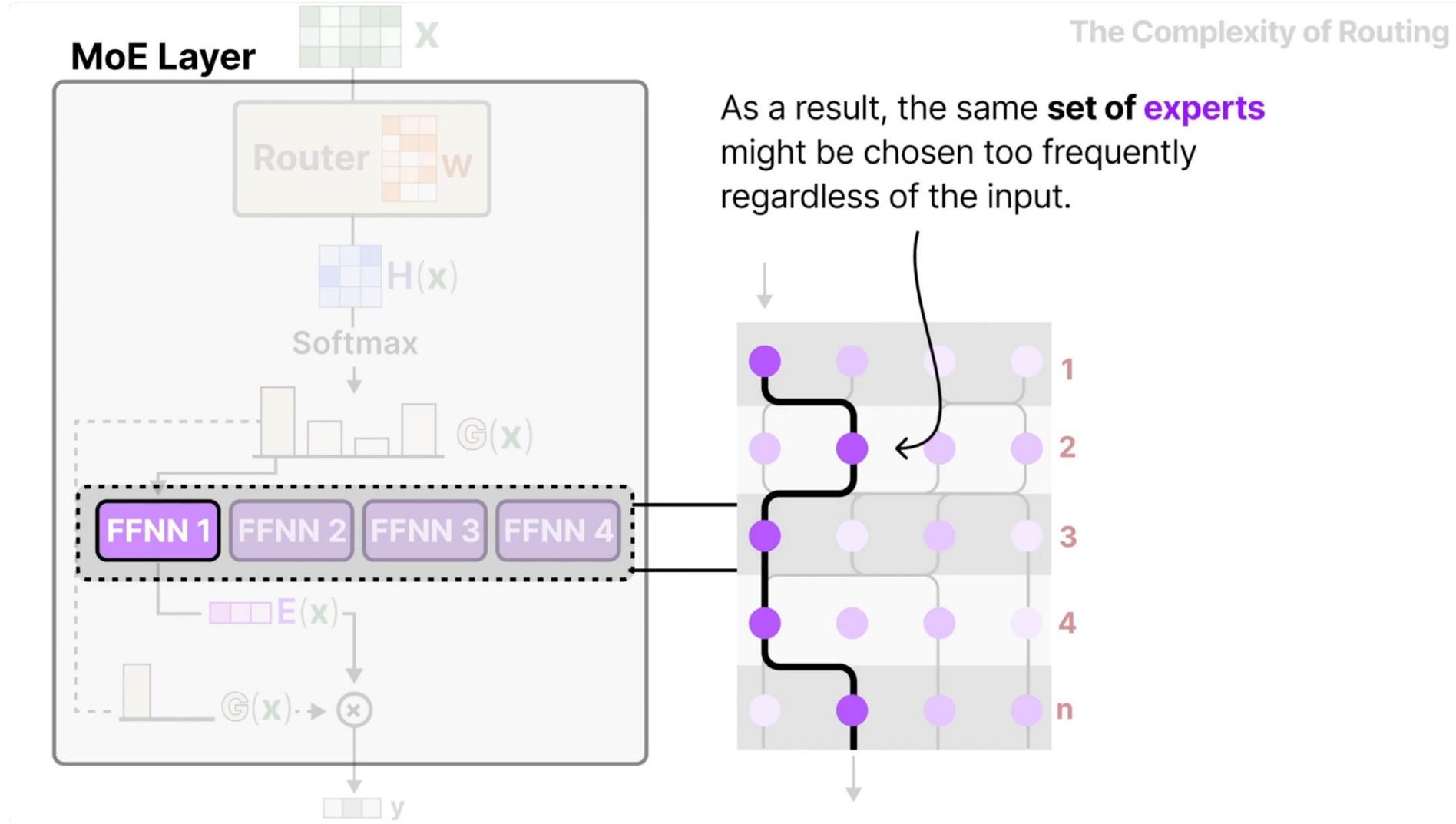
Load Balancing: 均衡负载



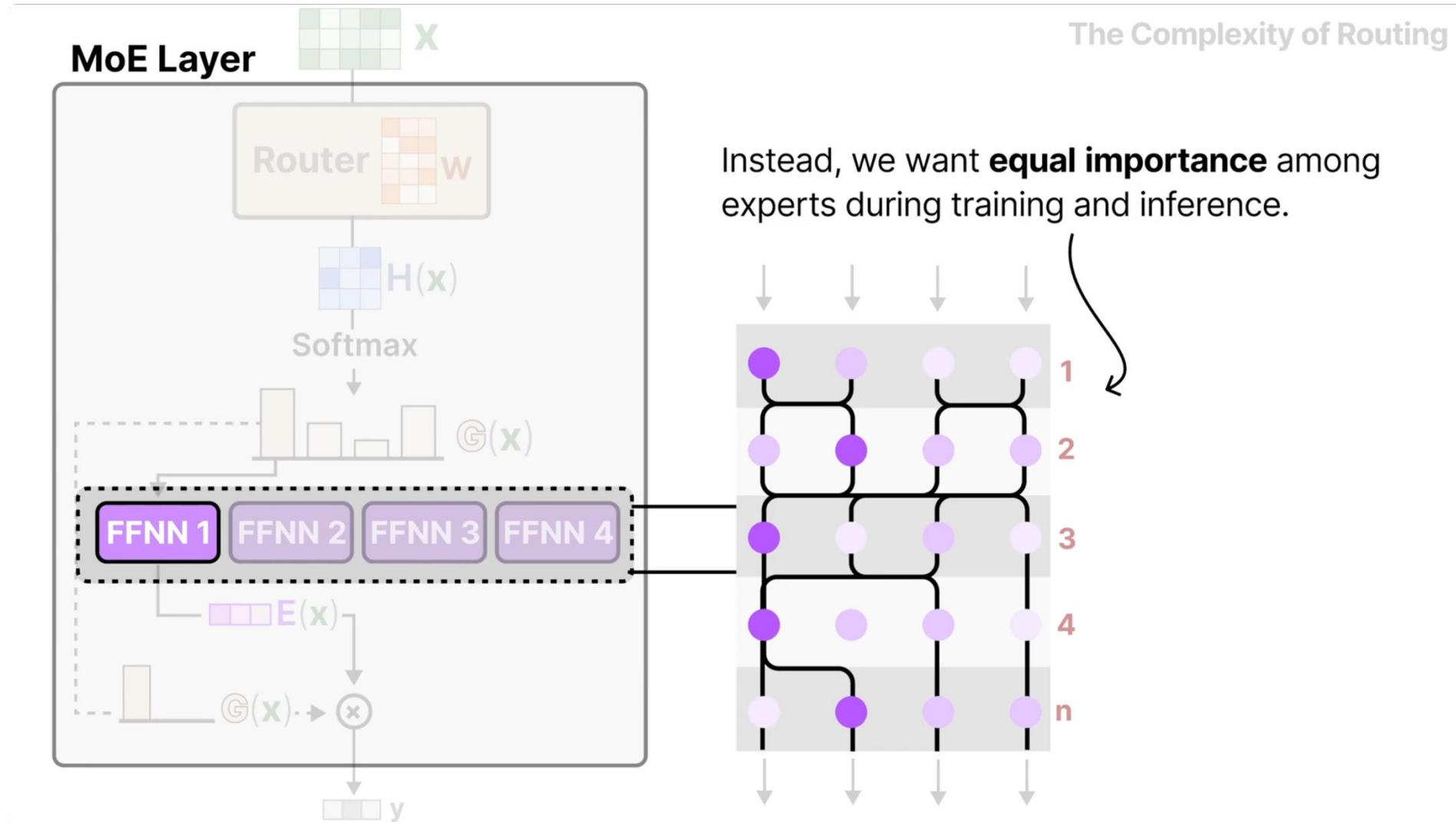
Load Balancing: 均衡负载



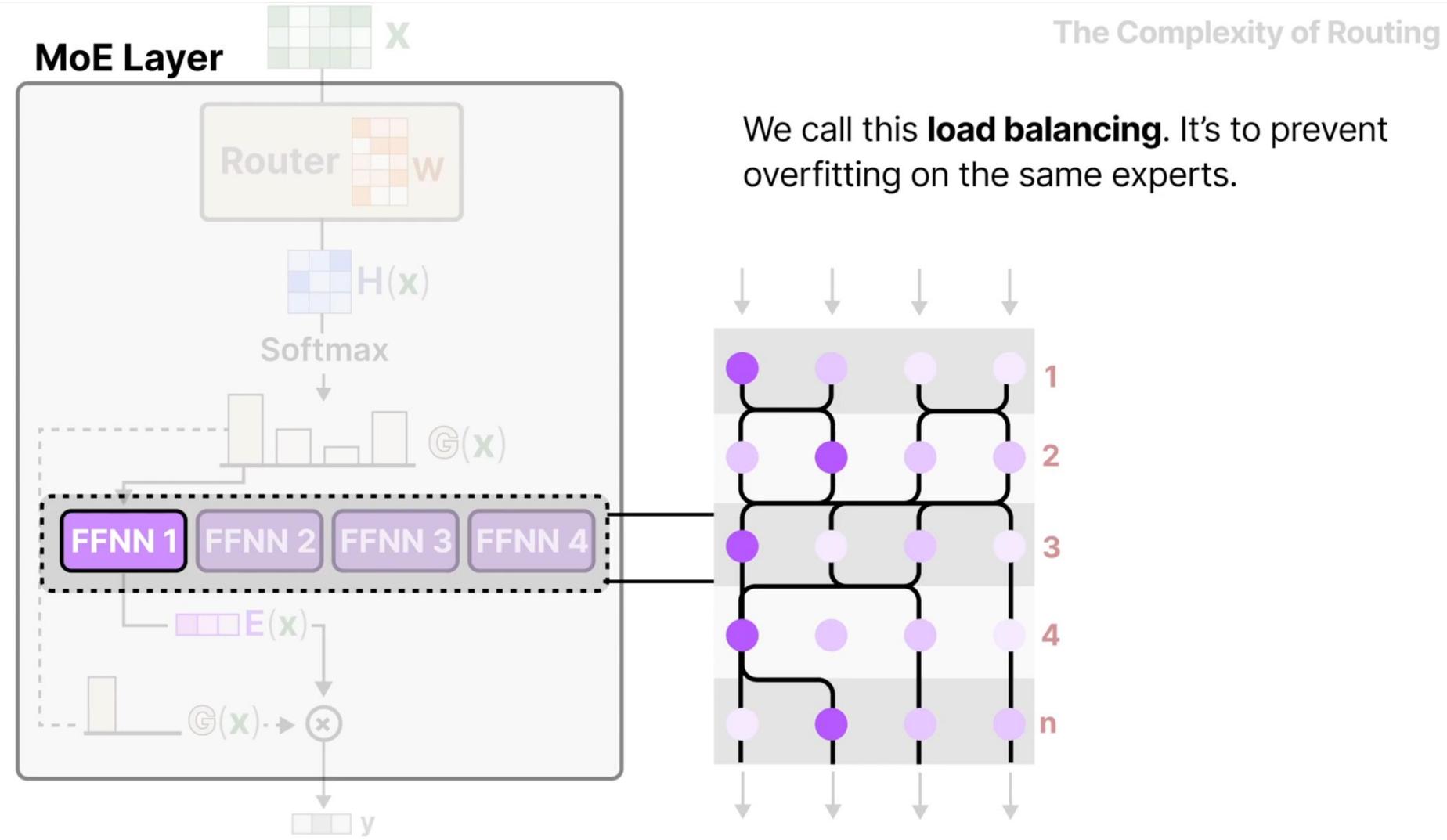
Load Balancing: 均衡负载



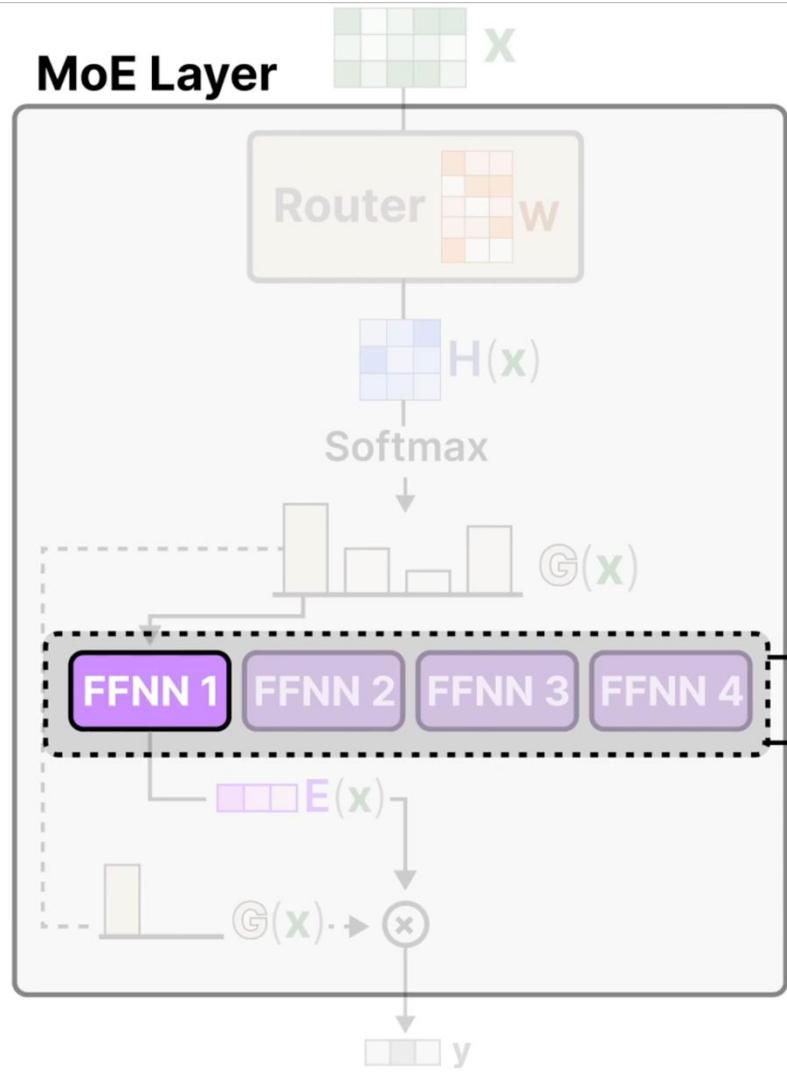
Load Balancing: 均衡负载



Load Balancing: 均衡负载

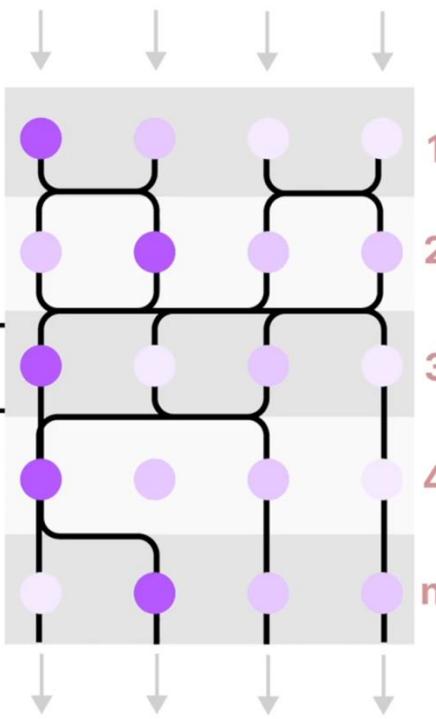


Load Balancing: 均衡负载



The Complexity of Routing

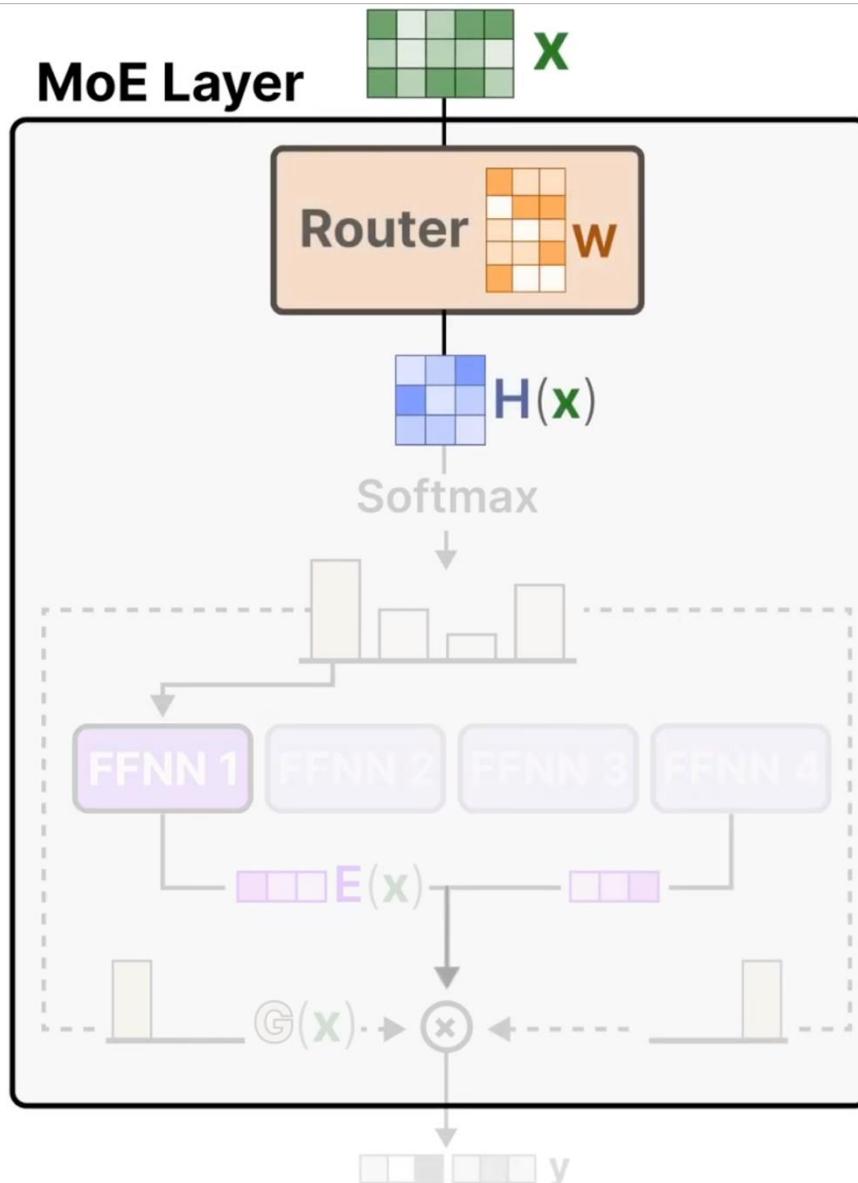
To explore advancements in **load balancing**, let's look at how we can improve the MoE layer with a method called **KeepTopK**.



05

Keep Top-K: 专家选择

Keep Top-K: 专家选择



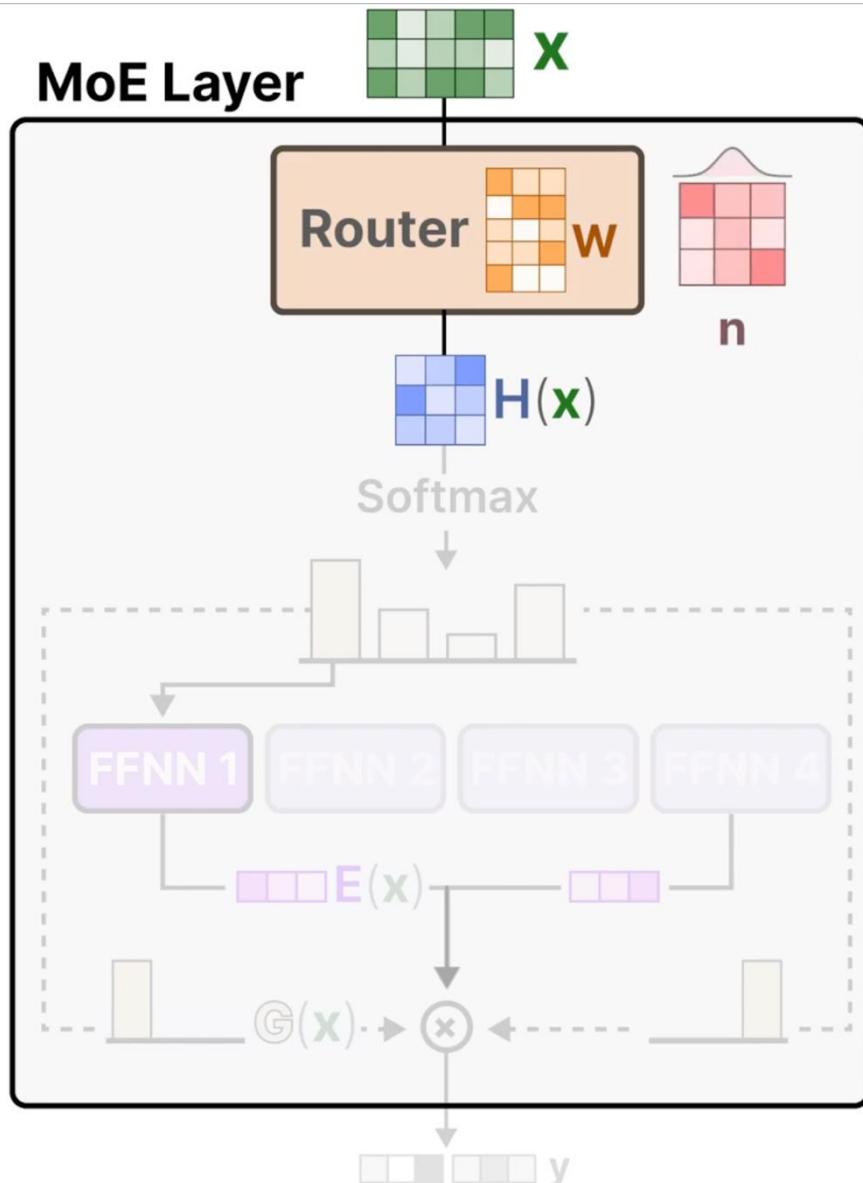
Remember that in the first step, we multiply the **input** with the **router** weights to create the **output** of the router.

$$\text{output} \quad \begin{matrix} & & & \\ & & & \\ & & & \\ & & & \end{matrix} \quad \begin{matrix} & & & \\ & & & \\ & & & \\ & & & \end{matrix} \quad \begin{matrix} & & & \\ & & & \\ & & & \\ & & & \end{matrix} \quad \begin{matrix} & & & \\ & & & \\ & & & \end{matrix}$$

$H(x) = X * W$

router weights

Keep Top-K: 专家选择

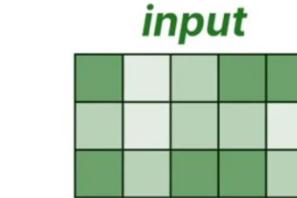


With **KeepTopK**, we introduce trainable (gaussian) **noise**...

(somewhat noisy)
output

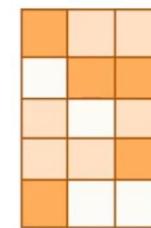


$H(x)$



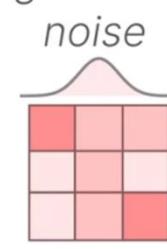
X

router
weights



W

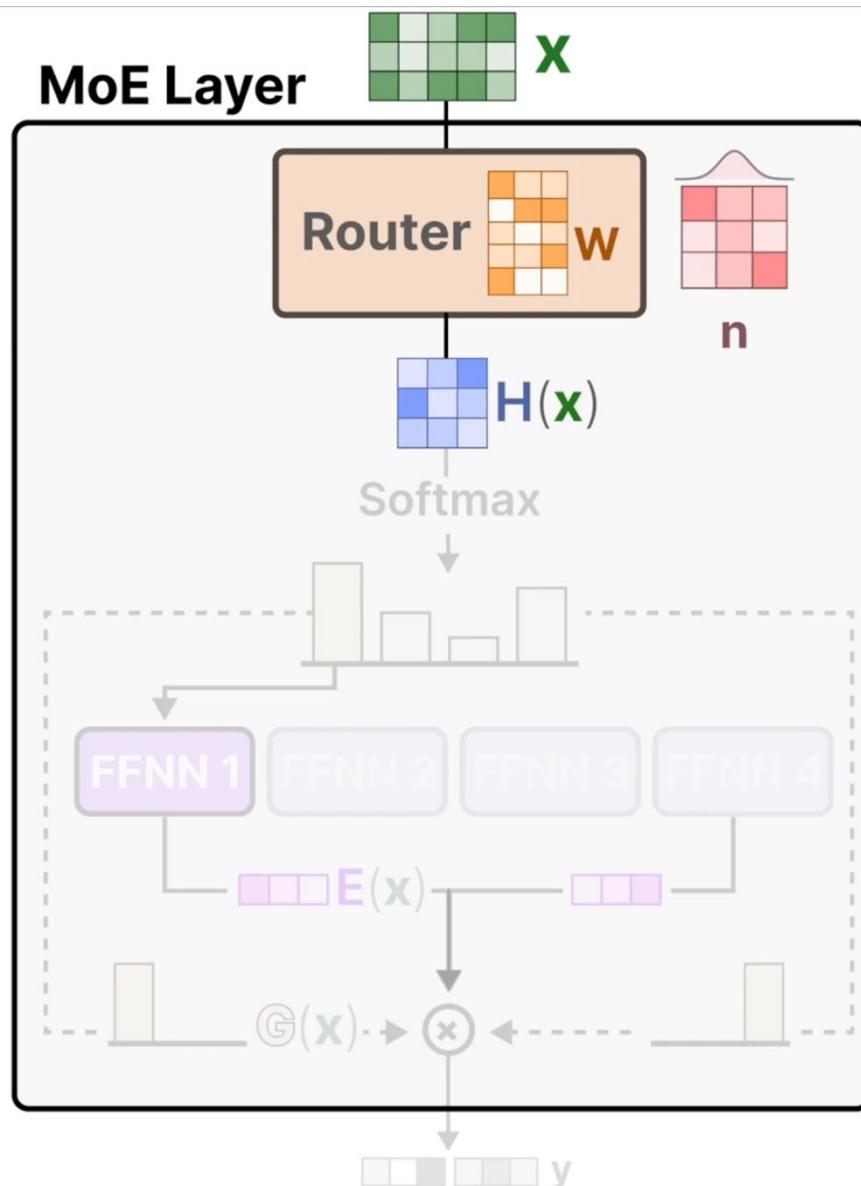
(small amount of)
gaussian
noise



n

...which helps us prevent the same experts from always being chosen.

Keep Top-K: 专家选择



KeepTopK

$$H(x) = \text{input} * W + n$$

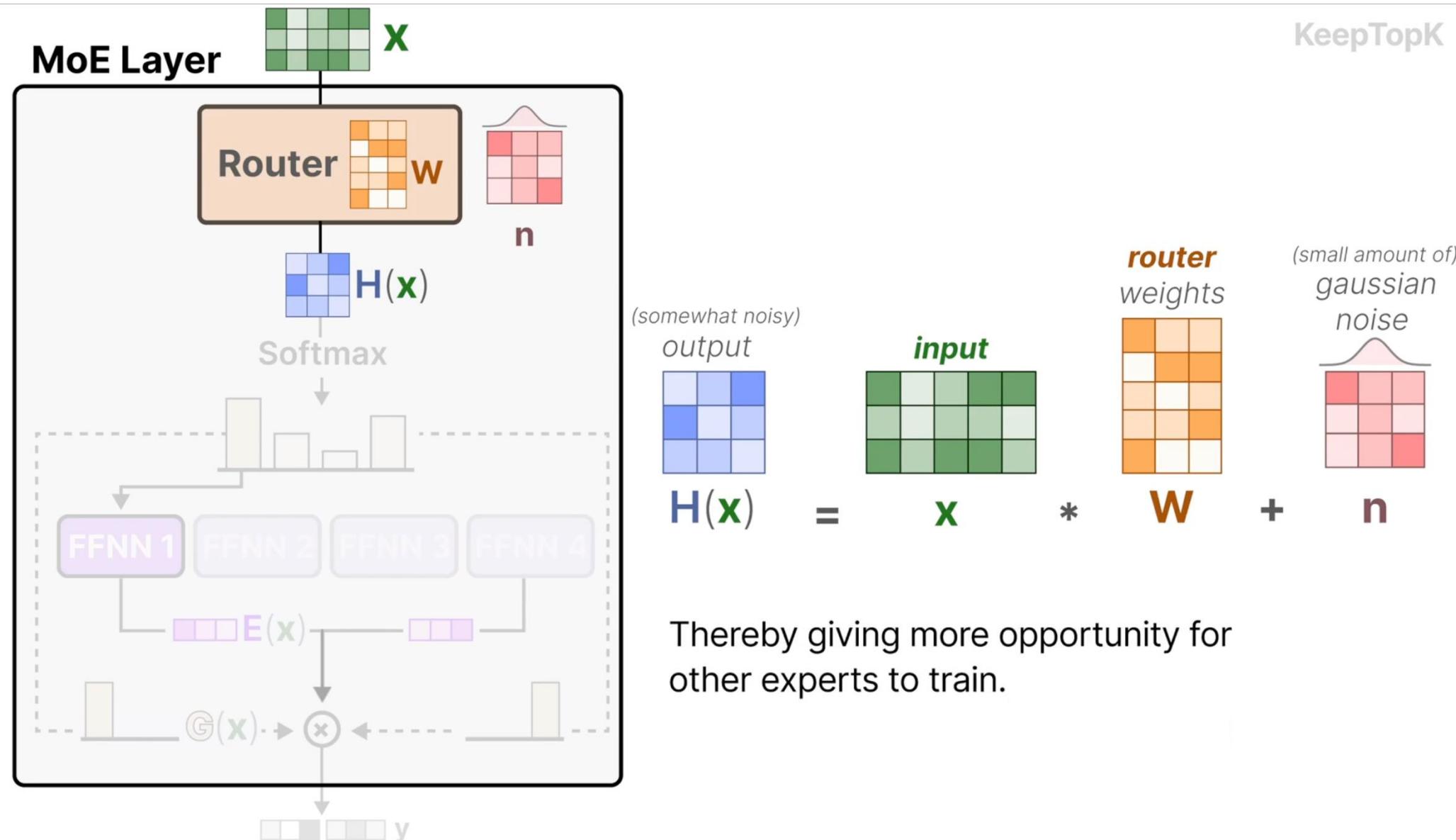
(somewhat noisy)
output

router
weights

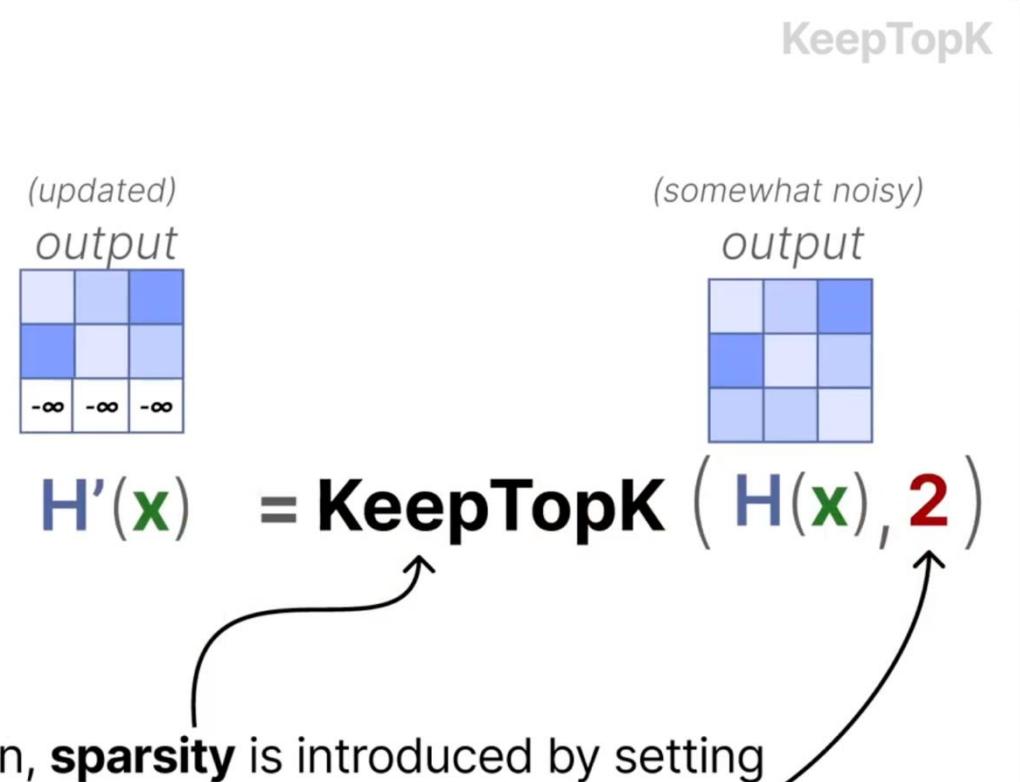
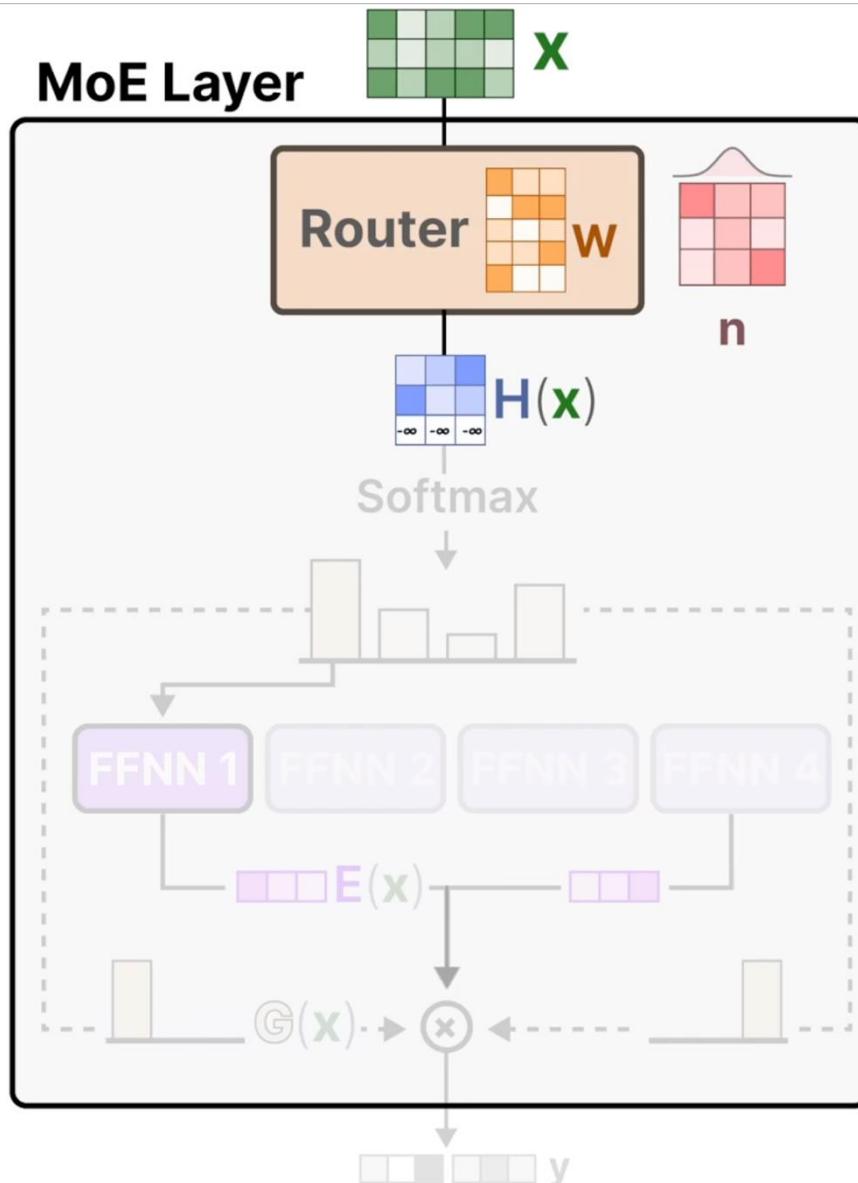
(small amount of)
gaussian
noise

By introducing noise, some experts might “accidentally” get lower scores.

Keep Top-K: 专家选择

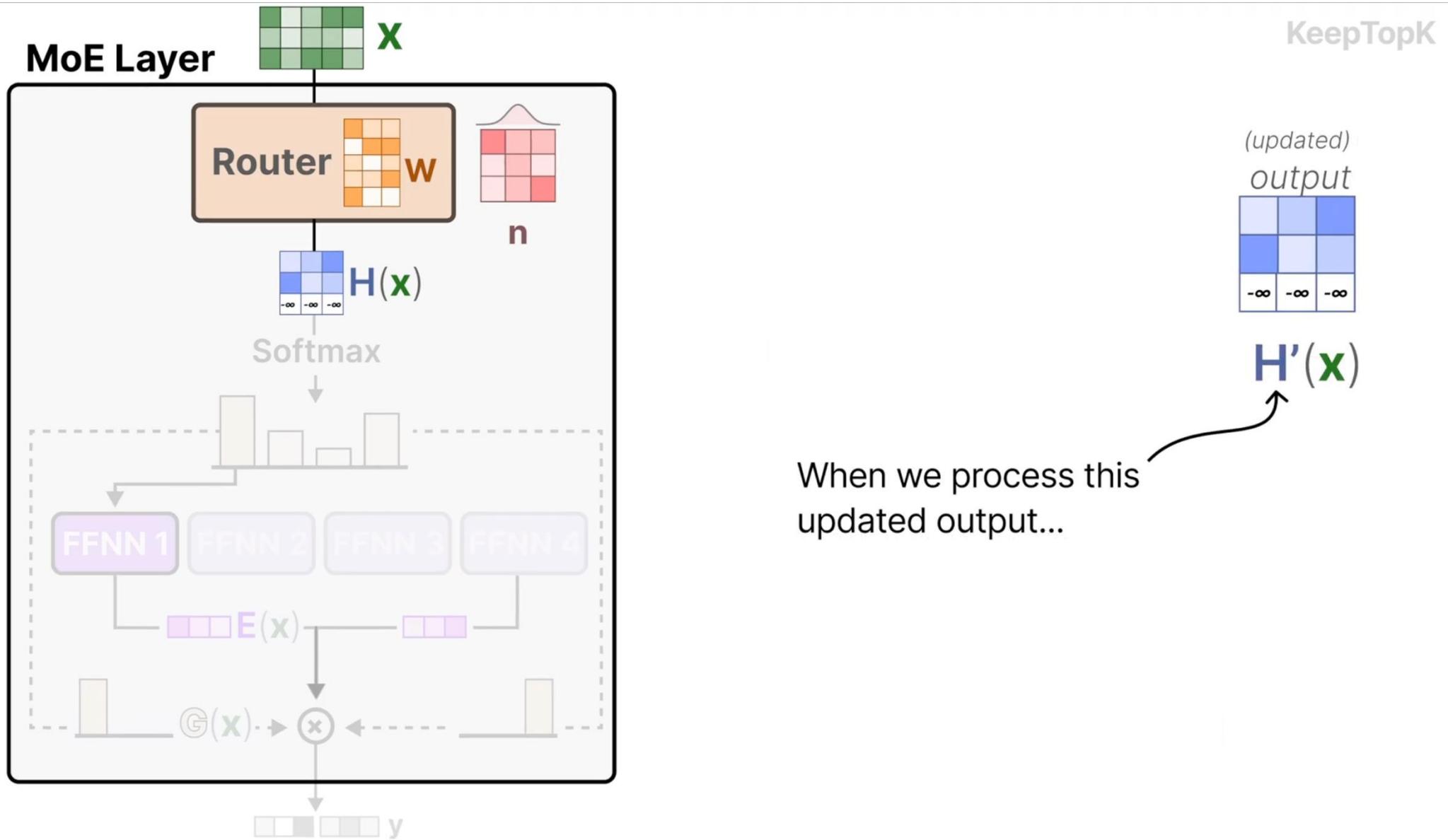


Keep Top-K: 专家选择

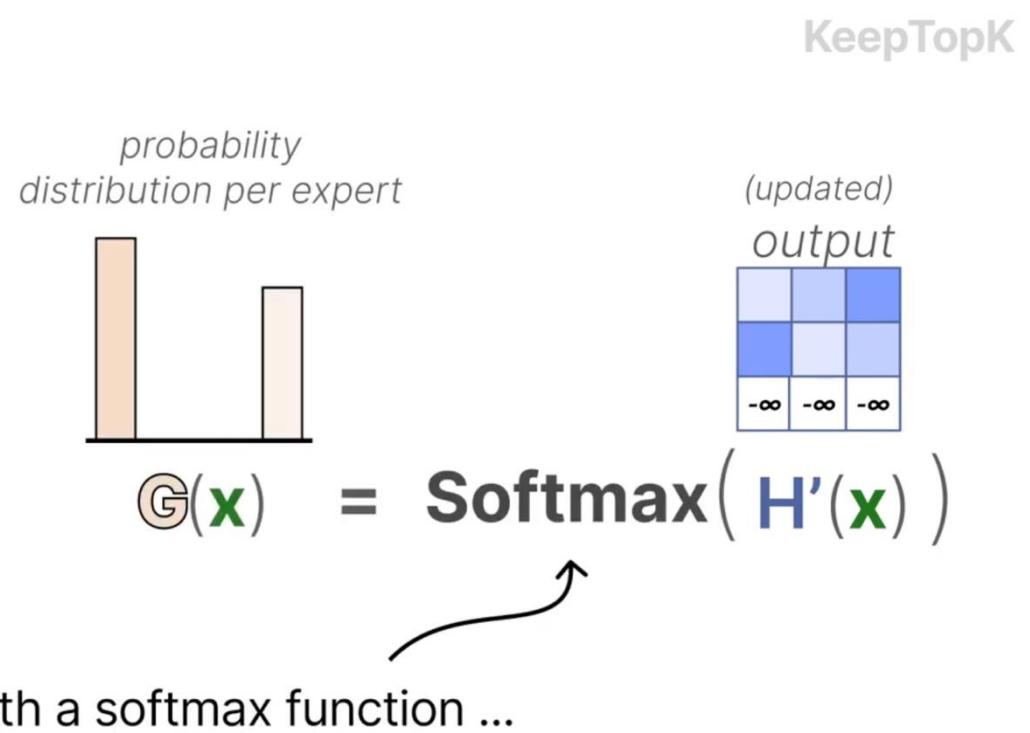
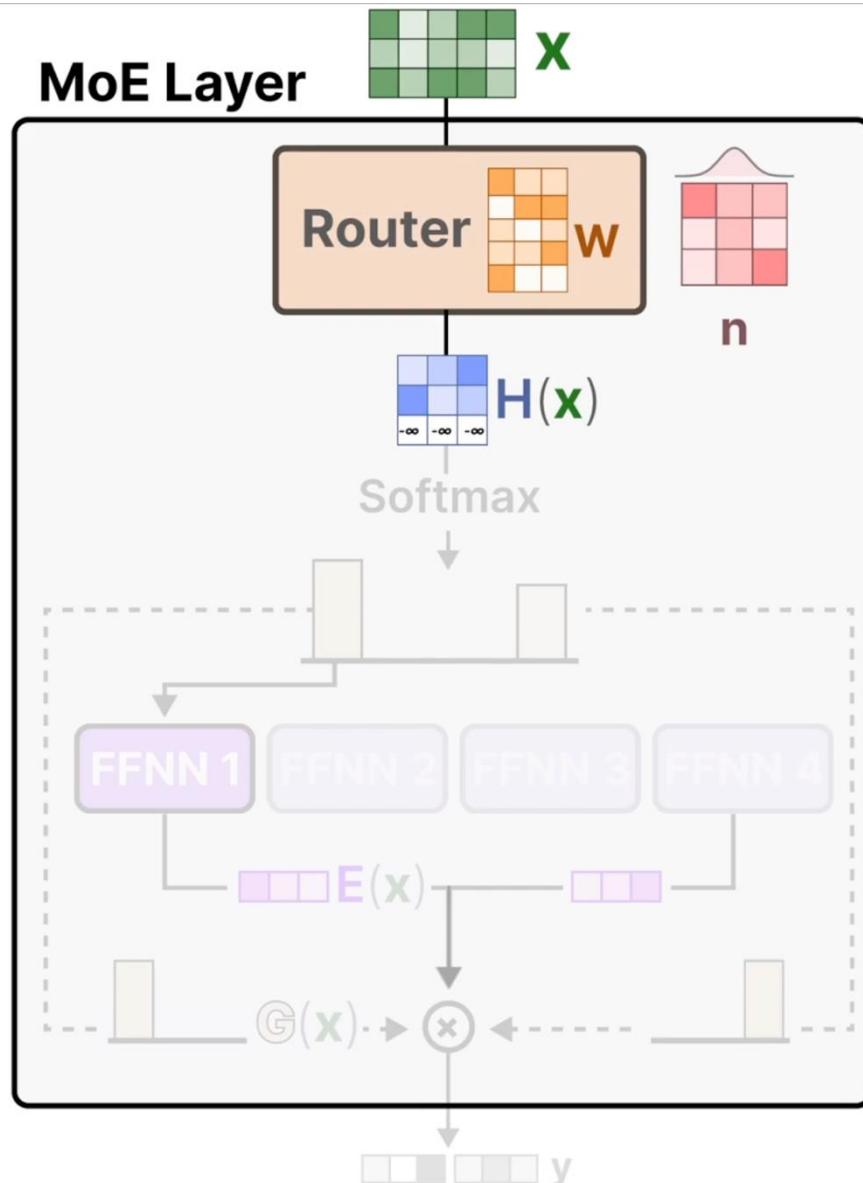


Then, **sparsity** is introduced by setting the weights of all but the **top k (2)** experts to $-\infty$

Keep Top-K: 专家选择

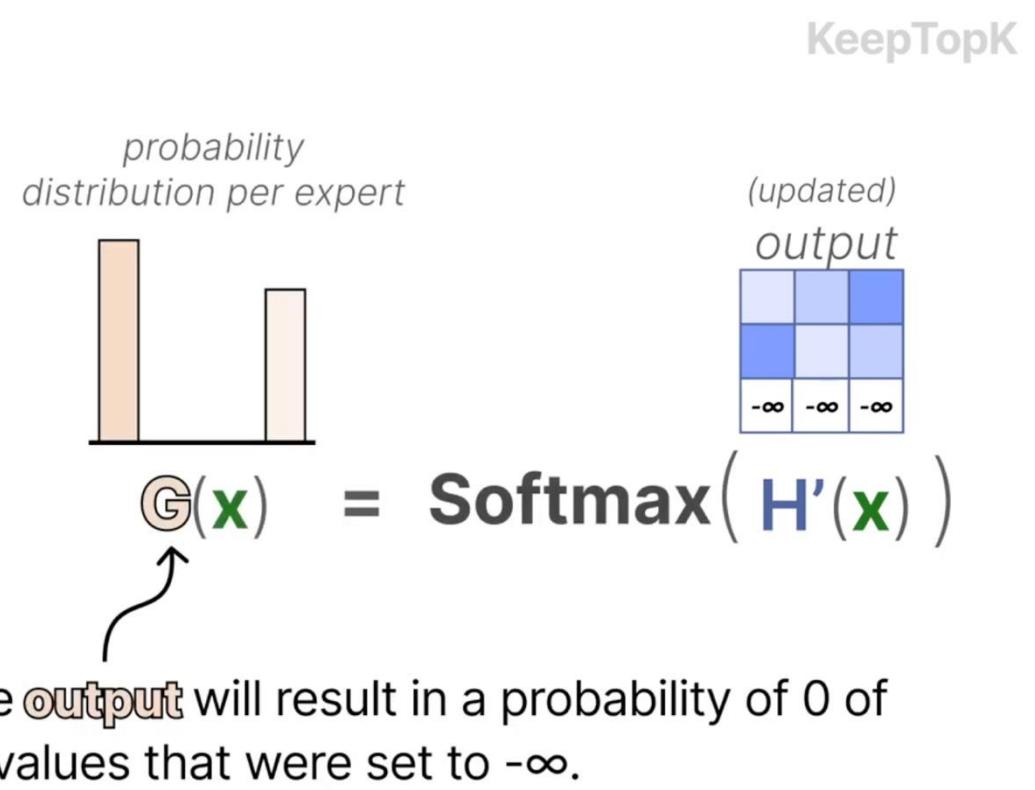
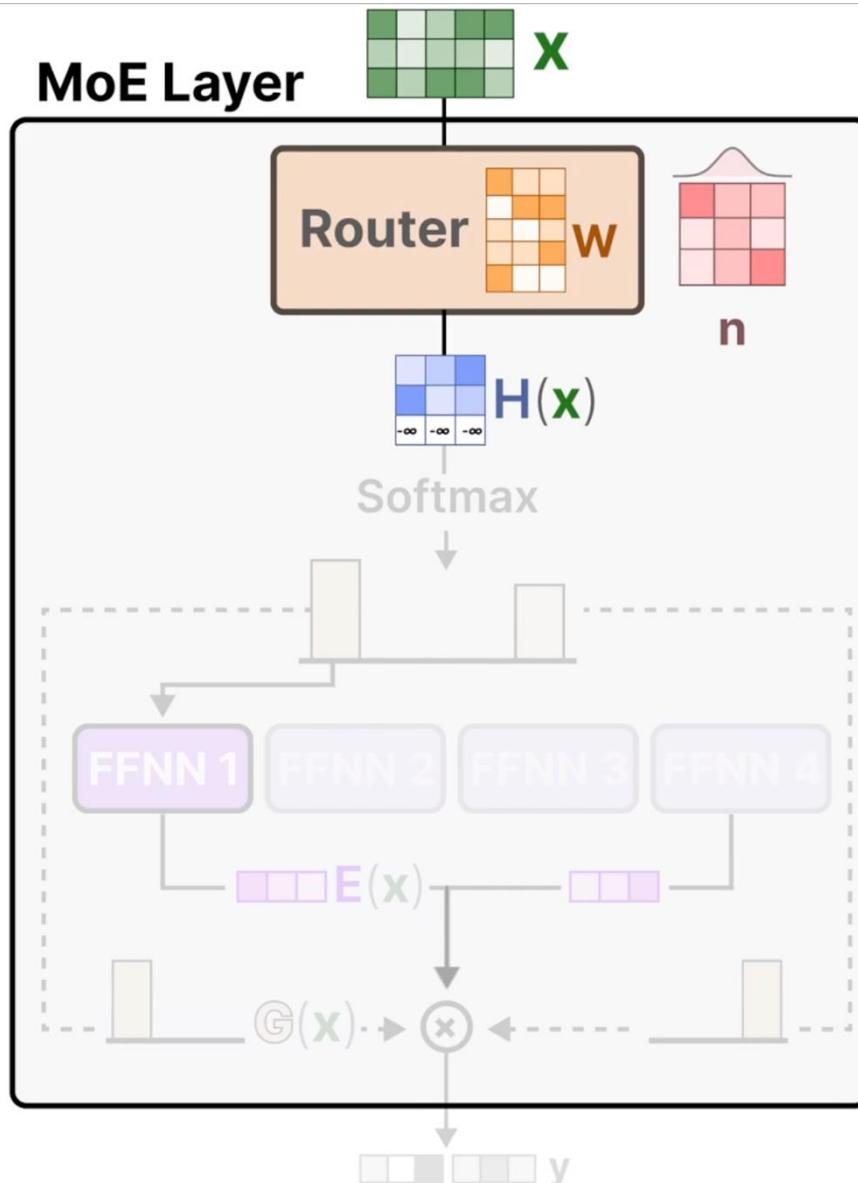


Keep Top-K: 专家选择

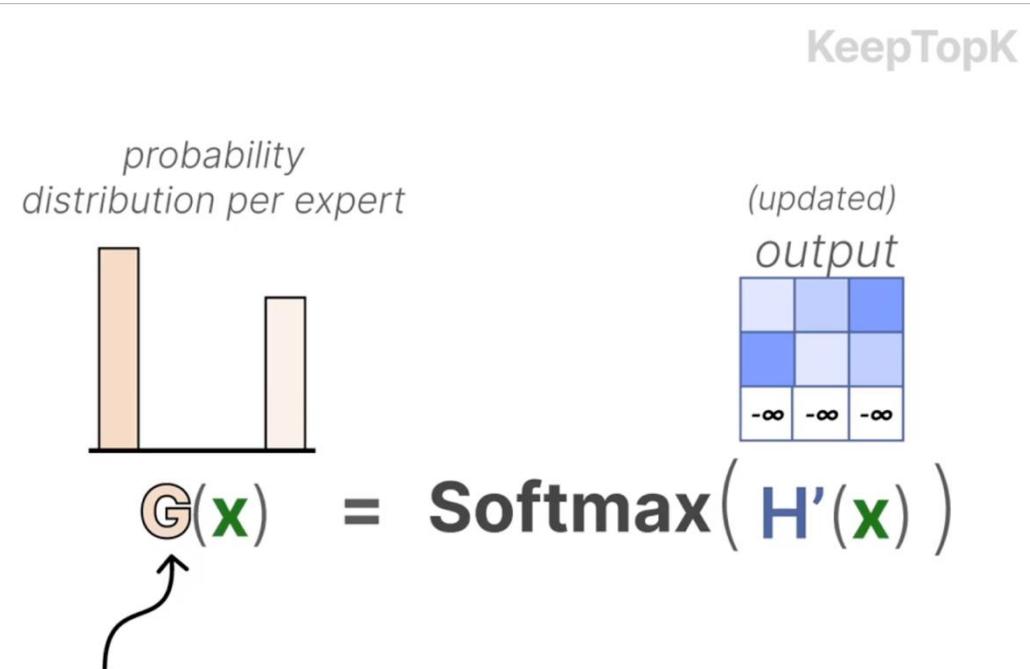
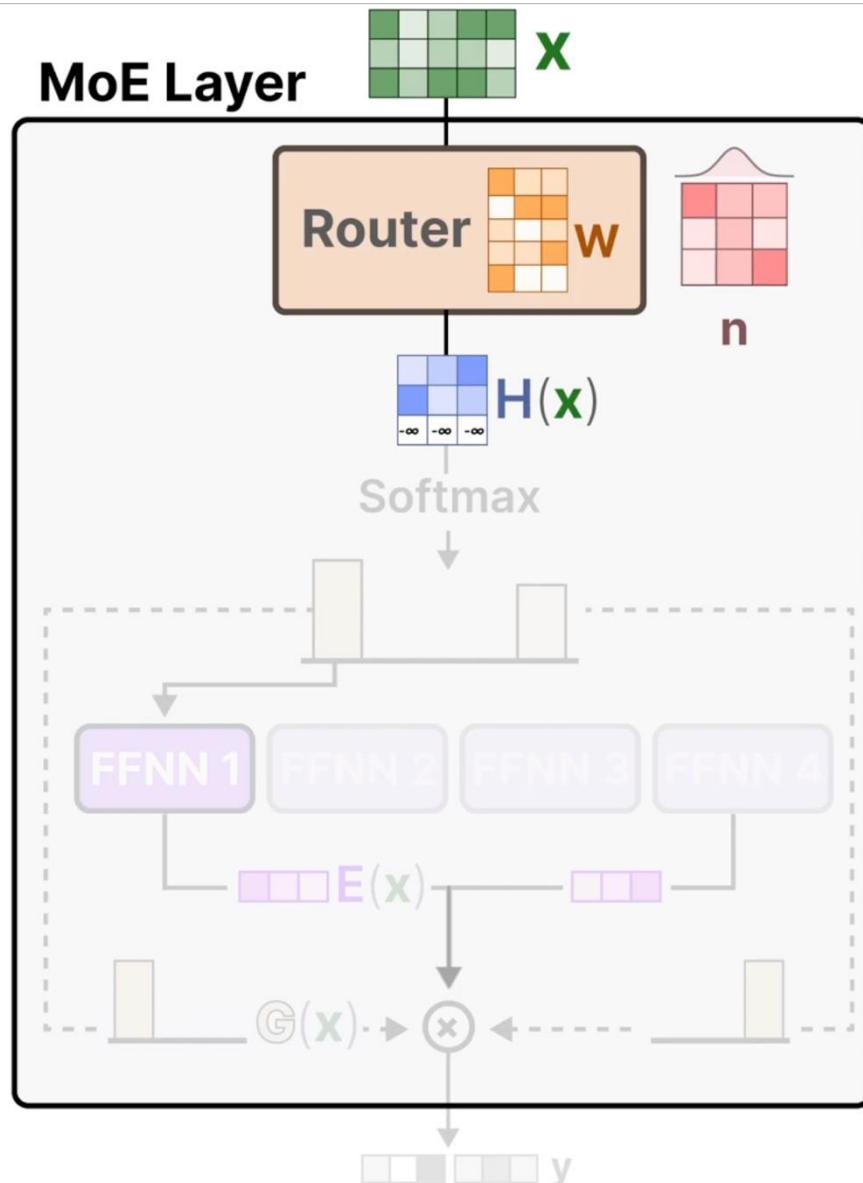


... with a softmax function ...

Keep Top-K: 专家选择

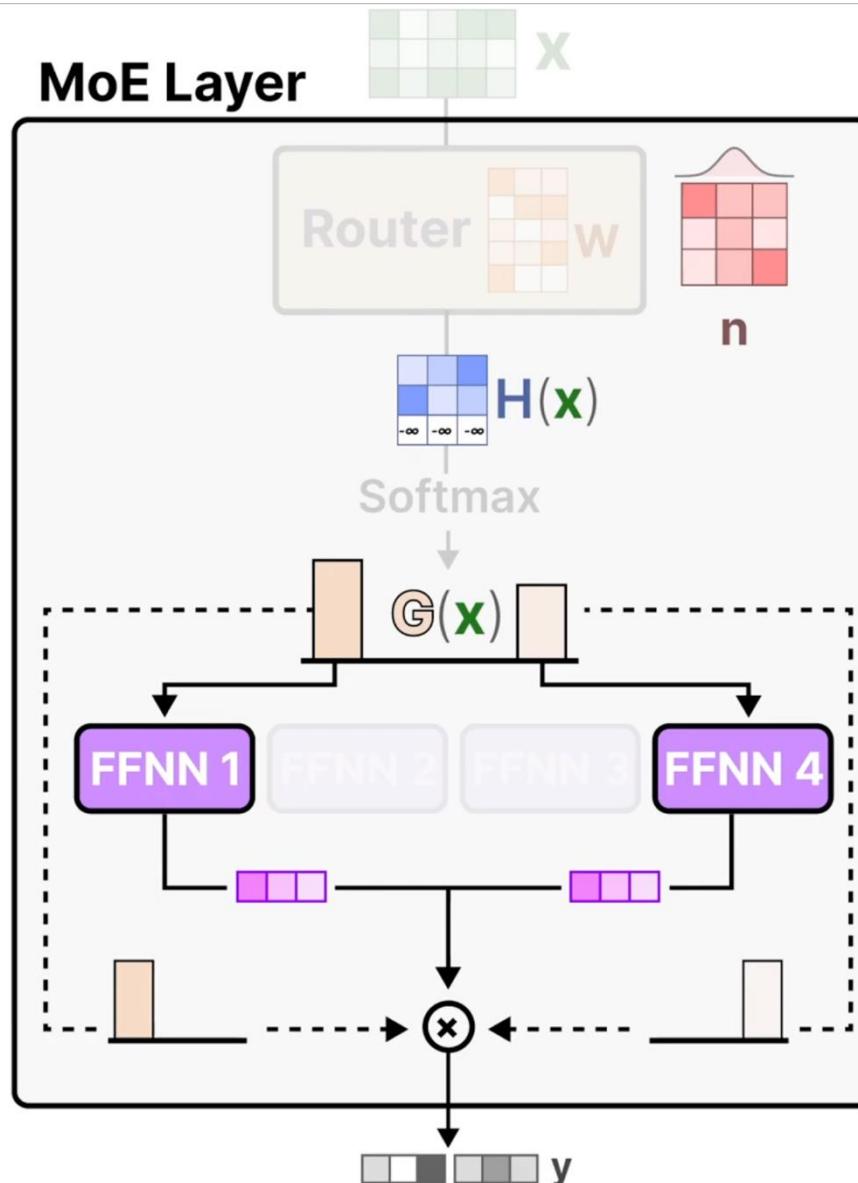


Keep Top-K: 专家选择



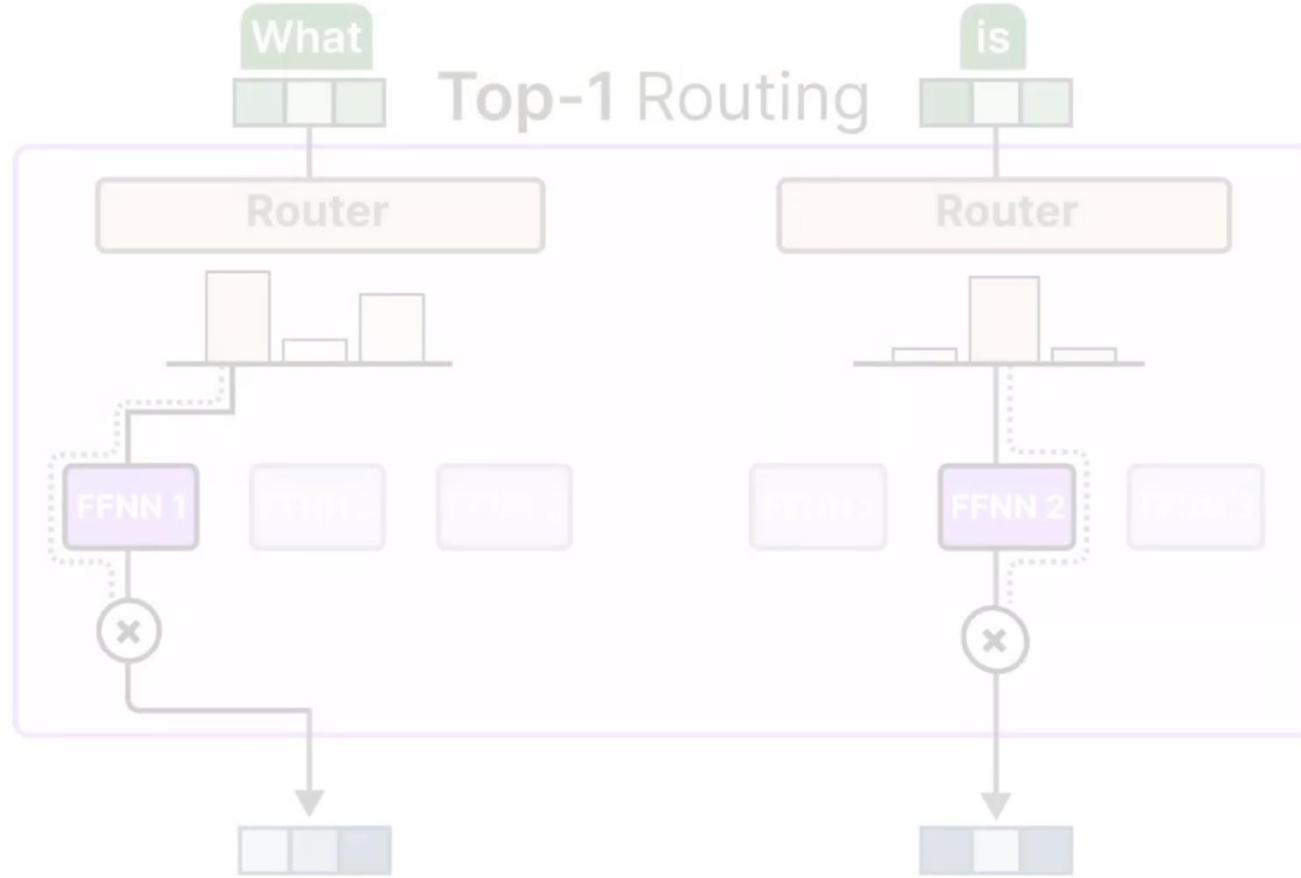
Therefore, it sets the probabilities of all but the top k (2) experts to **0**.

Keep Top-K: 专家选择



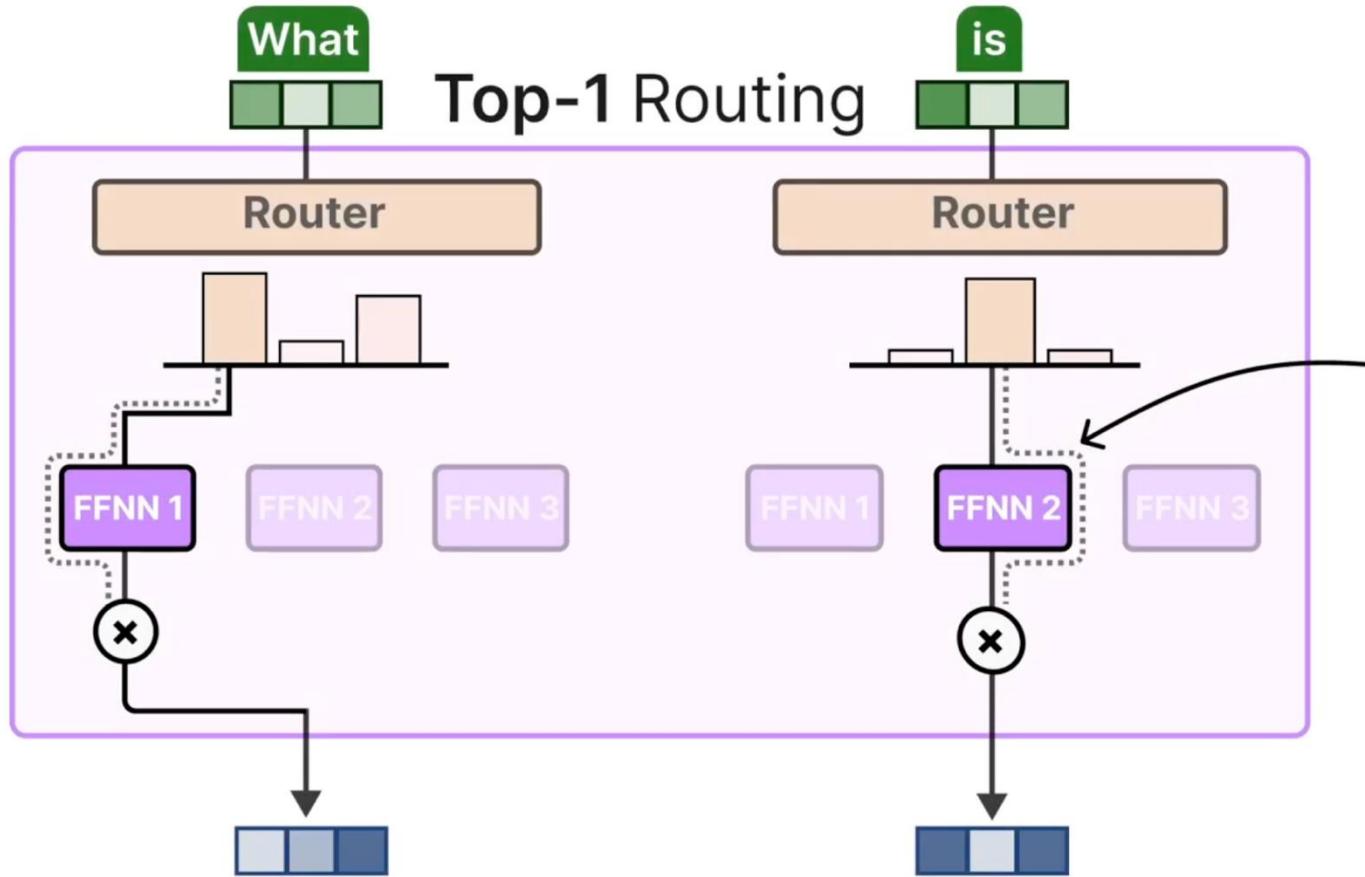
Combined, they allow undertrained **experts** to “catch up” to experts that were chosen more frequently and therefore had more opportunity to train.

Keep Top-K: 专家选择



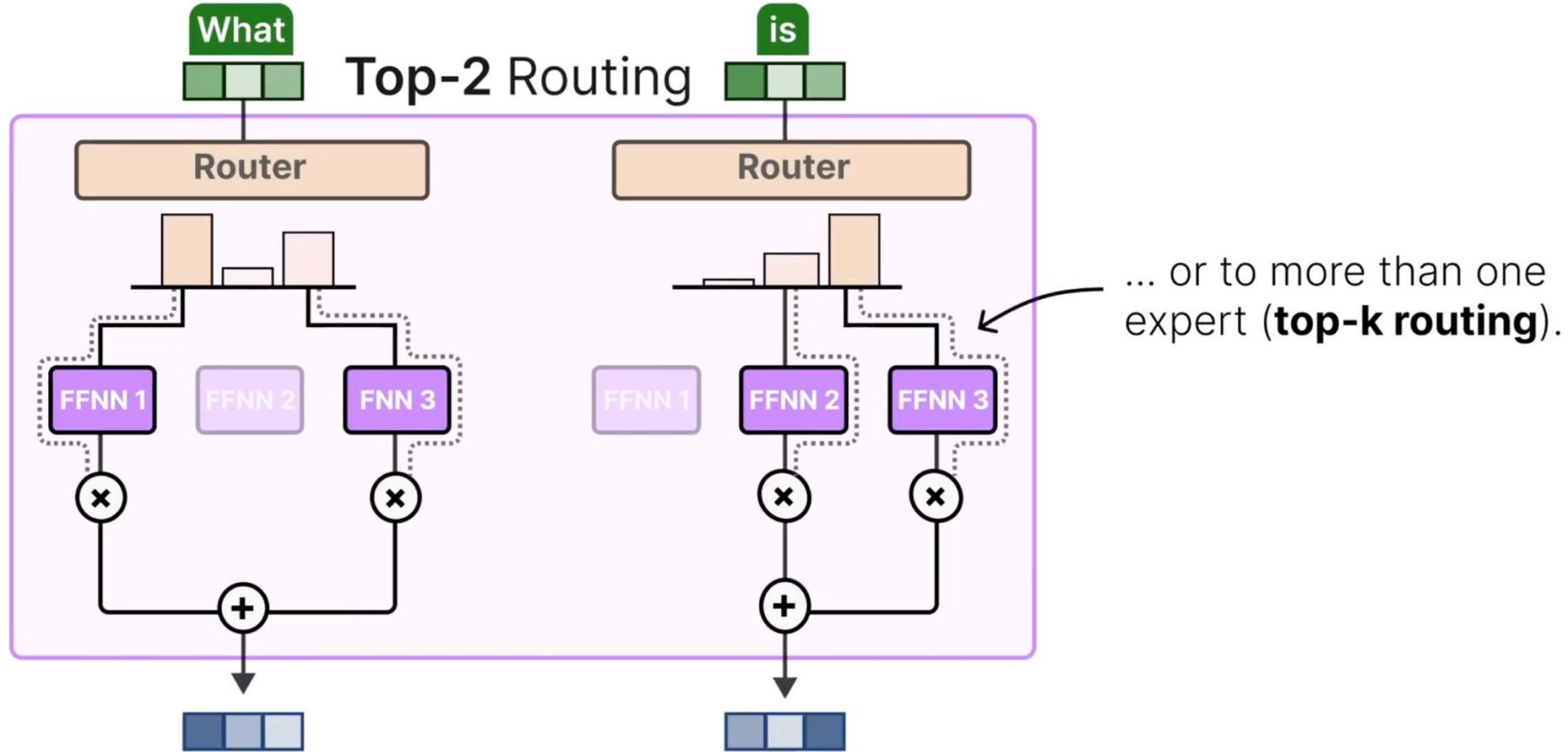
Routing tokens to a few selected experts is also called **Token Choice**.

Keep Top-K: 专家选择



... and allows for a given token to be sent to one expert (**top-1 routing**) ...

Keep Top-K: 专家选择



06

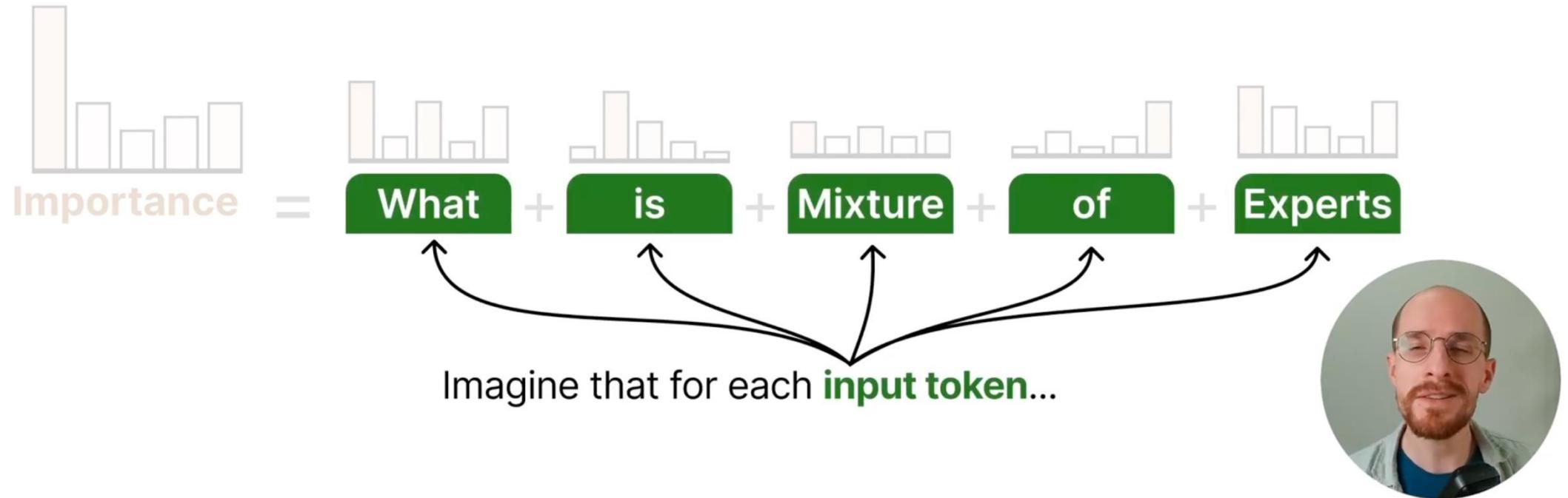
Auxiliary Loss: 辅助损失

Auxiliary Loss: 辅助损失

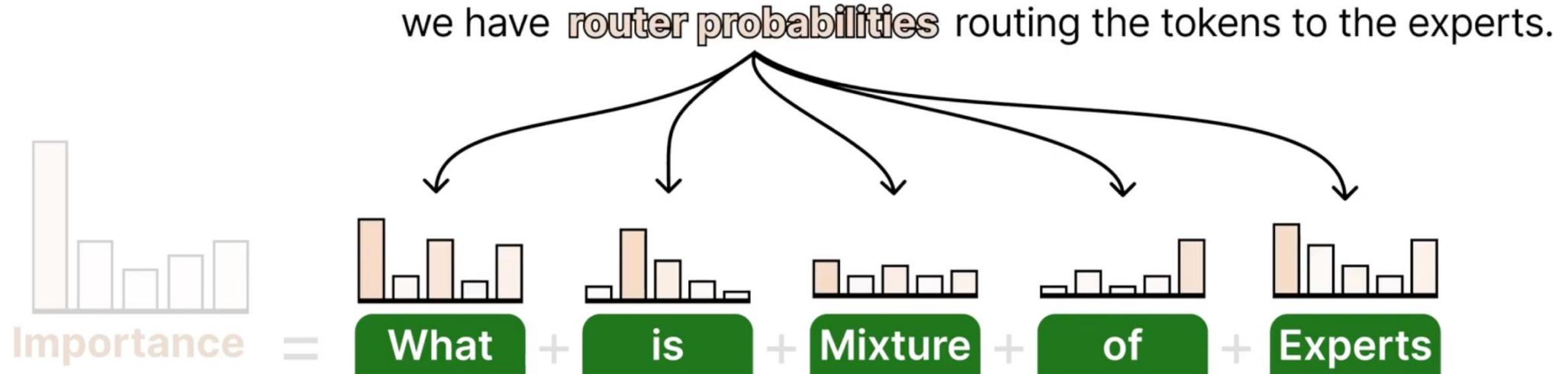
To further improve load balancing, we can add **auxiliary loss** (also called load balancing loss) to the network's regular loss.



Auxiliary Loss: 辅助损失

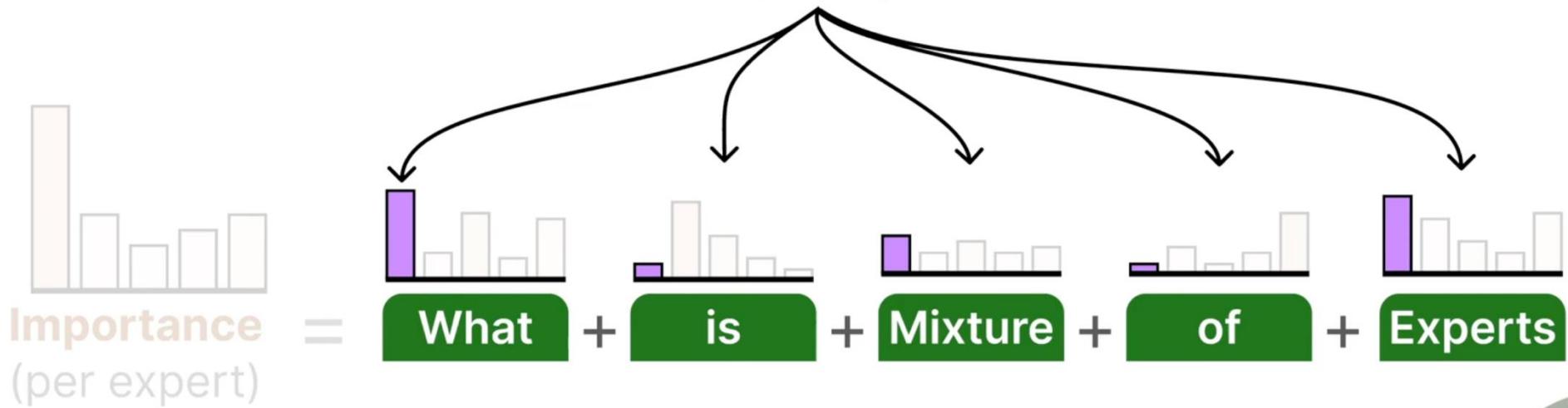


Auxiliary Loss: 辅助损失



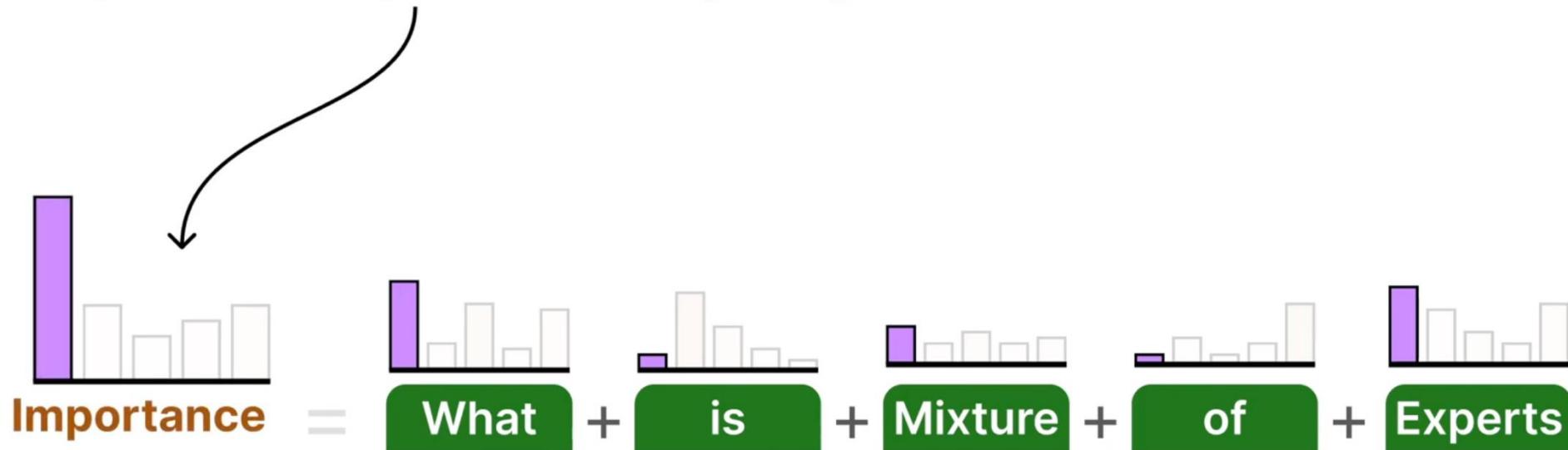
Auxiliary Loss: 辅助损失

The first component of this auxiliary loss is to **sum** the router values **per expert**.



Auxiliary Loss: 辅助损失

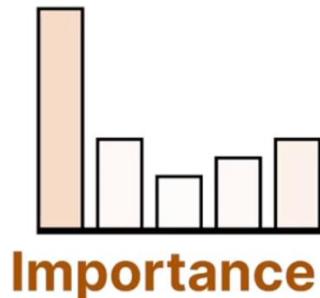
This gives us the **importance score per expert**...



Auxiliary Loss: 辅助损失

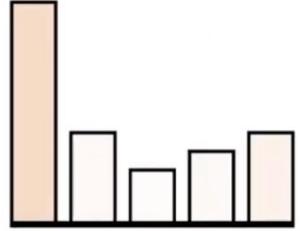
How equal the distribution of importance scores is, can be calculated with

$$\text{Coefficient Variation (CV)} = \frac{\text{standard deviation } (\sigma)}{\text{mean } (\mu)}$$



Auxiliary Loss: 辅助损失

For instance, if there are a lot of differences in importance scores, the **CV** will be **high**:


$$CV \text{ (Importance) } = \frac{.30}{.27} = 1.11$$

A hand-drawn arrow points from the text "the CV will be high:" down towards the calculated value of 1.11.



Auxiliary Loss: 辅助损失

if all experts have similar importance scores, the **CV** will be **low**:


$$CV \text{ (Importance)} = \frac{.05}{.26} = 0.19$$

A curved arrow points from the text "CV will be low:" down towards the calculated result of 0.19.



Auxiliary Loss: 辅助损失

Auxiliary loss is the **CV** multiplied by **w**, a constant scaling factor.

$$\text{Auxiliary Loss} = \text{CV} (\text{Importance})^2 * w$$

Diagram illustrating the calculation of Auxiliary Loss:

- A bar chart showing five bars of decreasing height, representing the magnitude of importance.
- The formula $\text{Auxiliary Loss} = \text{CV} (\text{Importance})^2 * w$ is shown.
- An arrow points from the term w to a dashed box labeled $w_{importance}$.
- Below the dashed box, the text "(constant) scaling factor" is written.



Auxiliary Loss: 辅助损失

The **Auxiliary loss** is updated during training such that it aims to lower the **CV** as much as possible....

$$\text{Auxiliary Loss} = \text{CV} (\text{Importance})^2 * w_{\text{importance}}$$

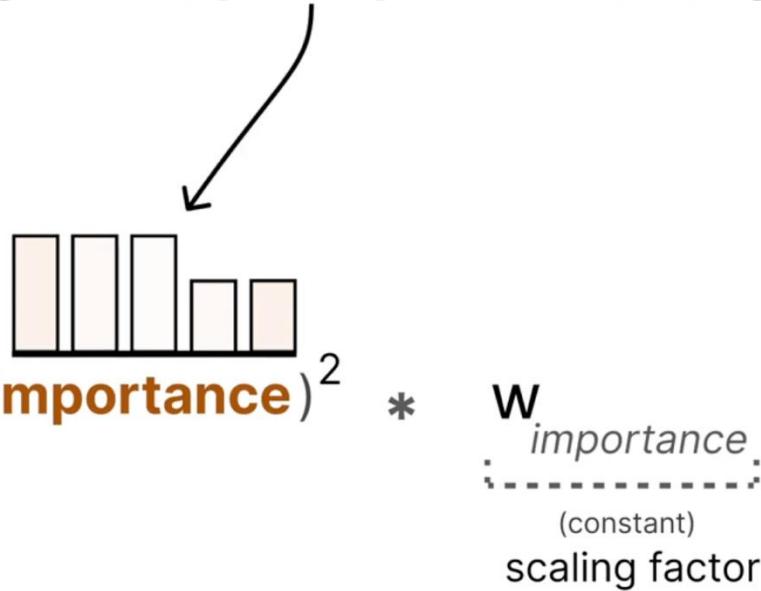
(constant)
scaling factor





Auxiliary Loss: 辅助损失

... thereby creating more **equal importance** among the experts.



Auxiliary Loss: 辅助损失

The **Auxiliary loss** is added as a separate loss to optimize during training.

$$\text{Auxiliary Loss} = CV \cdot (\text{Importance})^2 \cdot w_{\text{importance}}$$

(constant)
scaling factor





Auxiliary Loss: 辅助损失

This additional loss therefore results in a more **stable training** procedure where all experts are given (somewhat) equal chance to train.

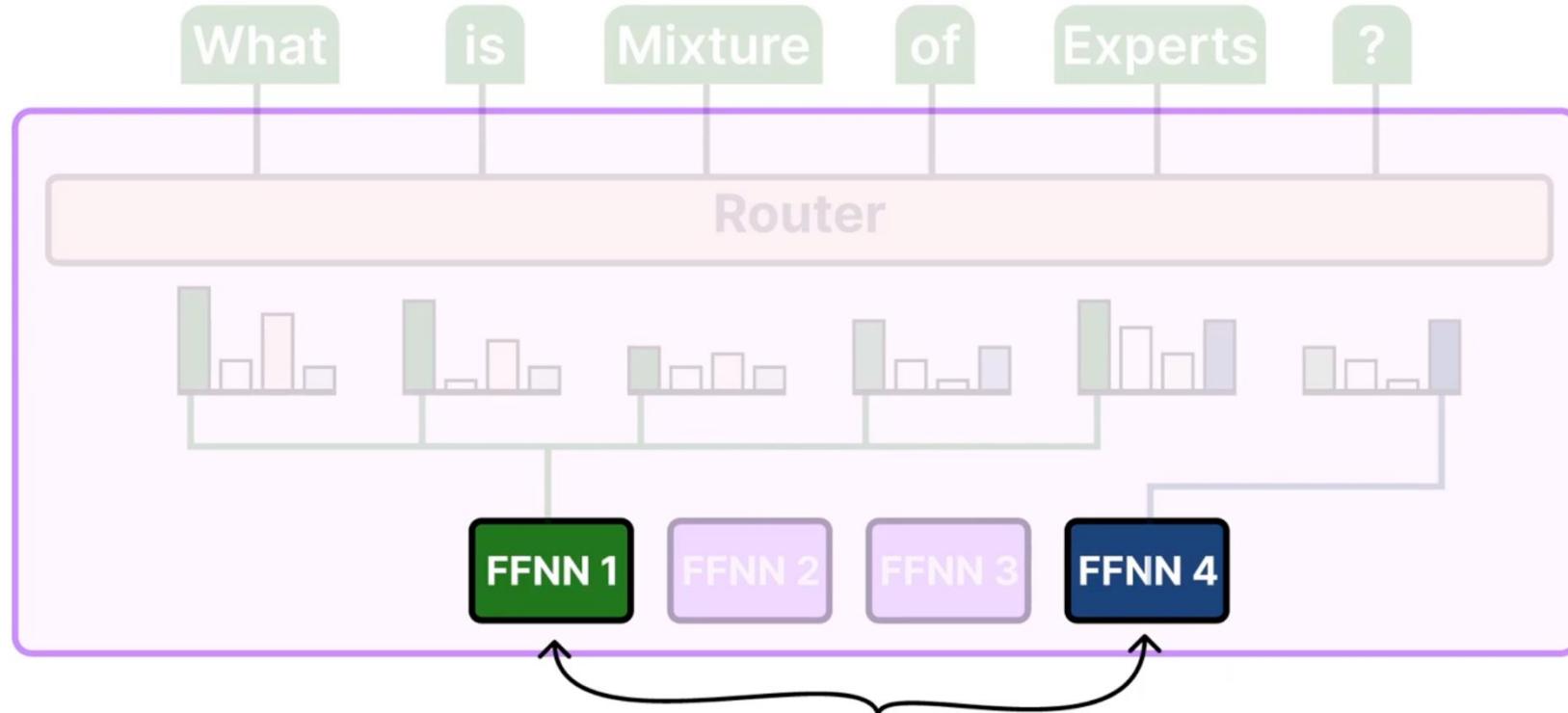
$$\text{Auxiliary Loss} = \text{CV} (\text{Importance})^2 * w_{\substack{\text{importance} \\ \text{(constant)} \\ \text{scaling factor}}}$$




07

Expert Capacity: 专家容量

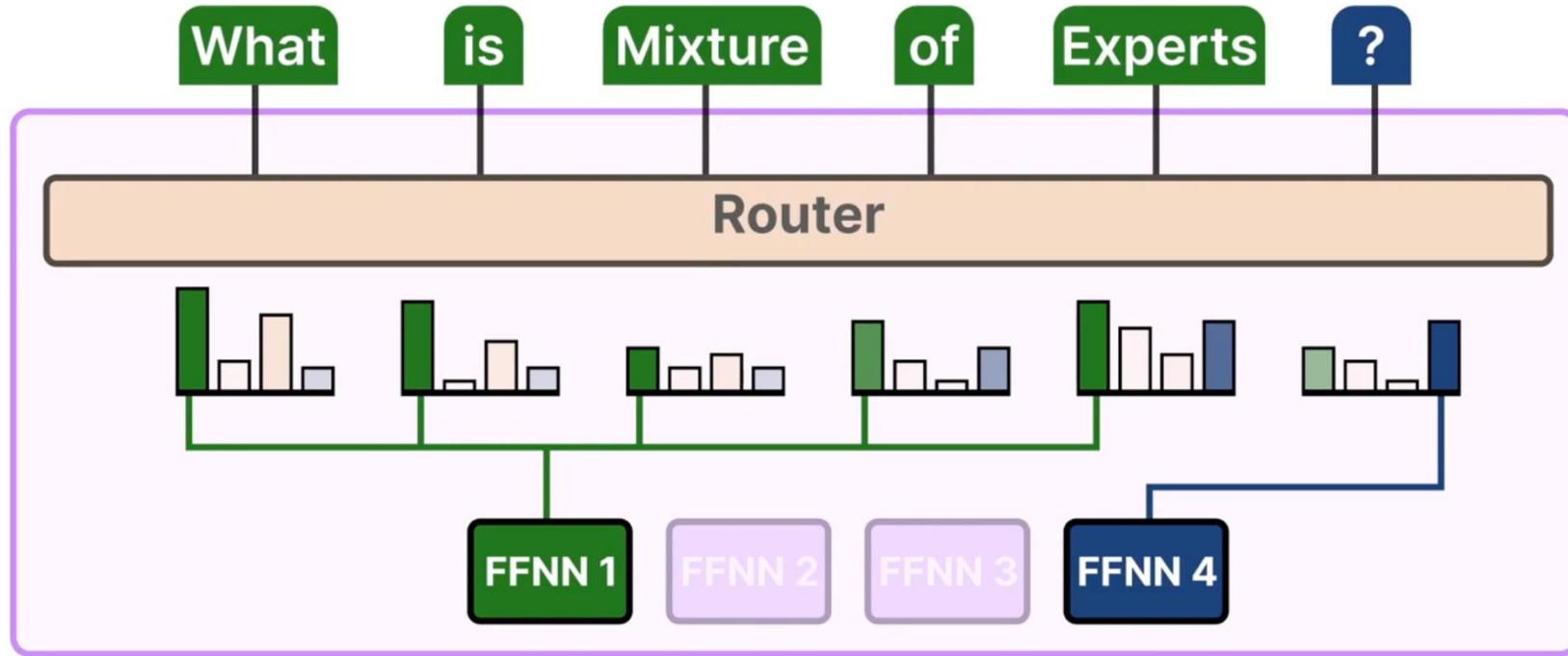
Expert Capacity: 专家容量



Imbalance, however, is not just found in the experts that were chosen...



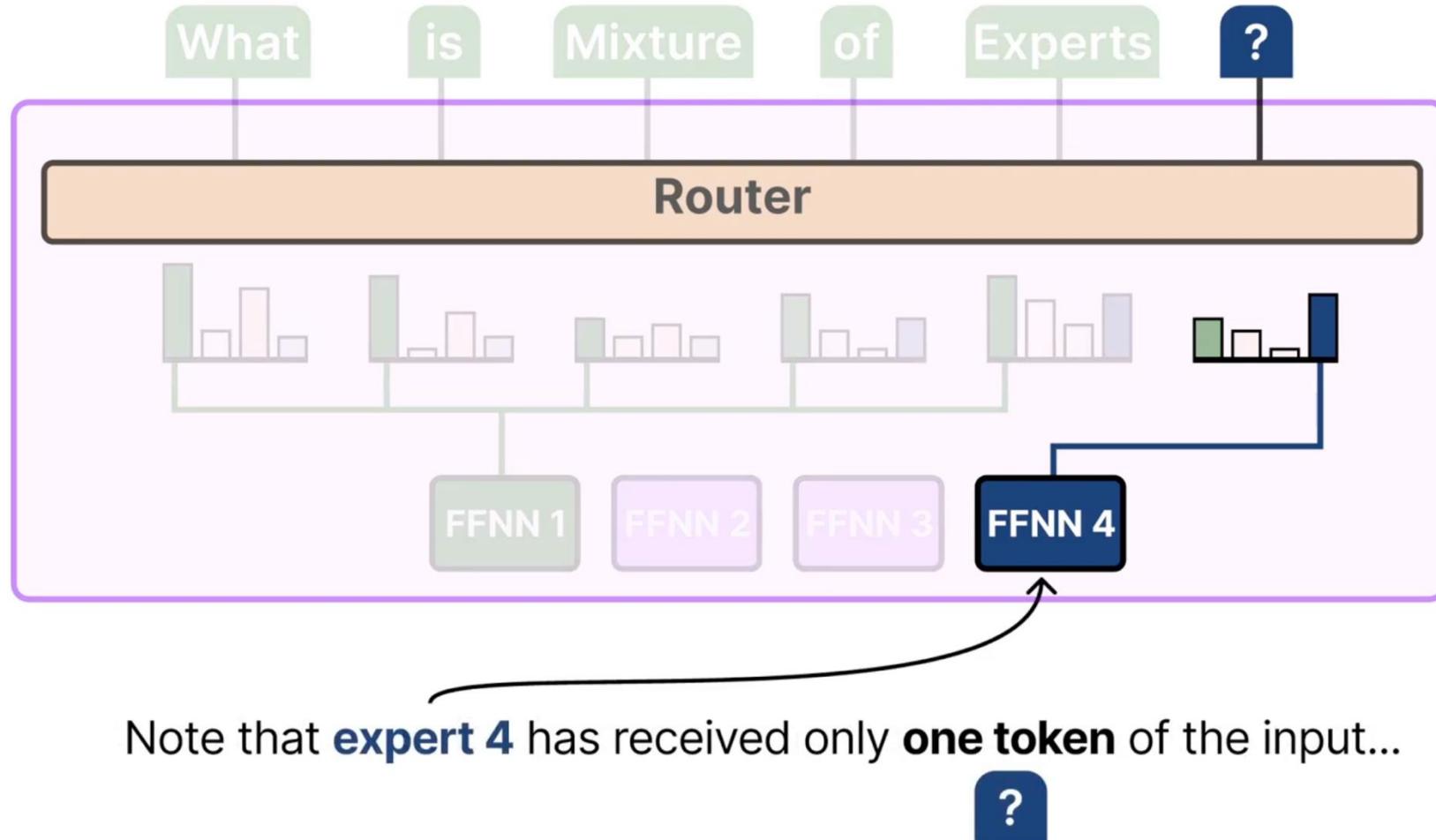
Expert Capacity: 专家容量



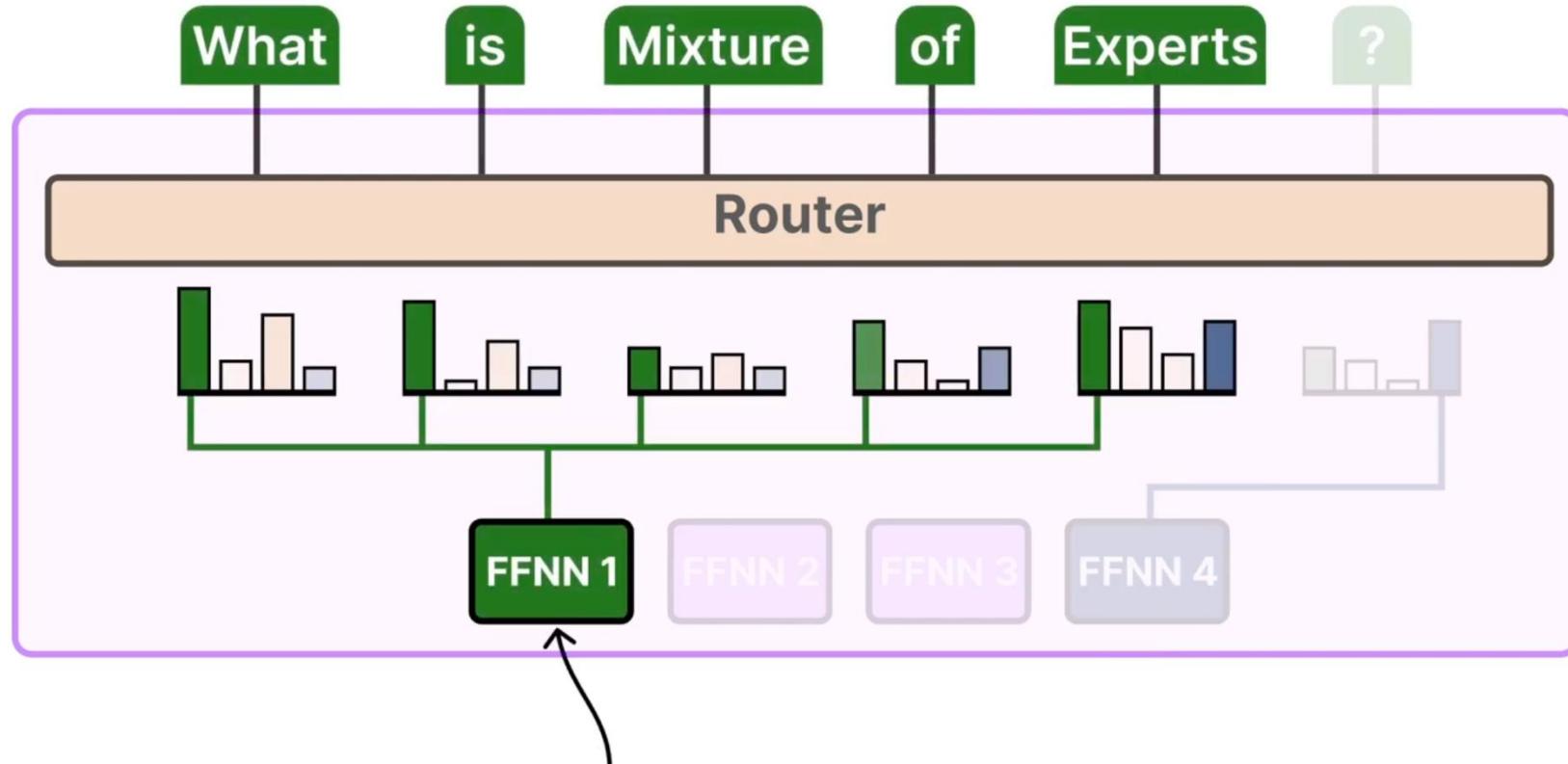
... but also in the distributions of tokens that are sent to the expert.



Expert Capacity: 专家容量



Expert Capacity: 专家容量

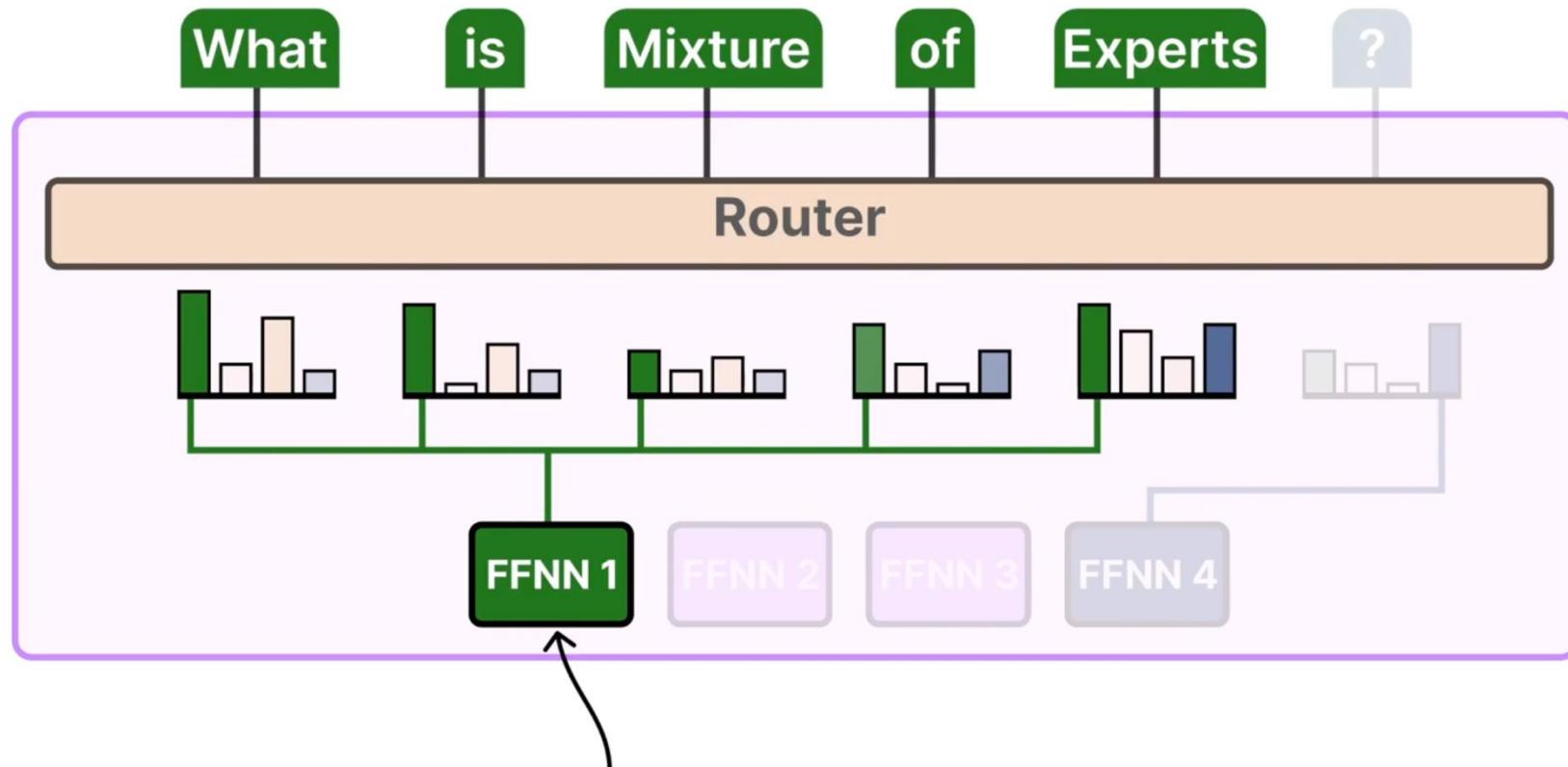


... whereas **expert 1** has received **all other tokens**.

What is Mixture of Experts



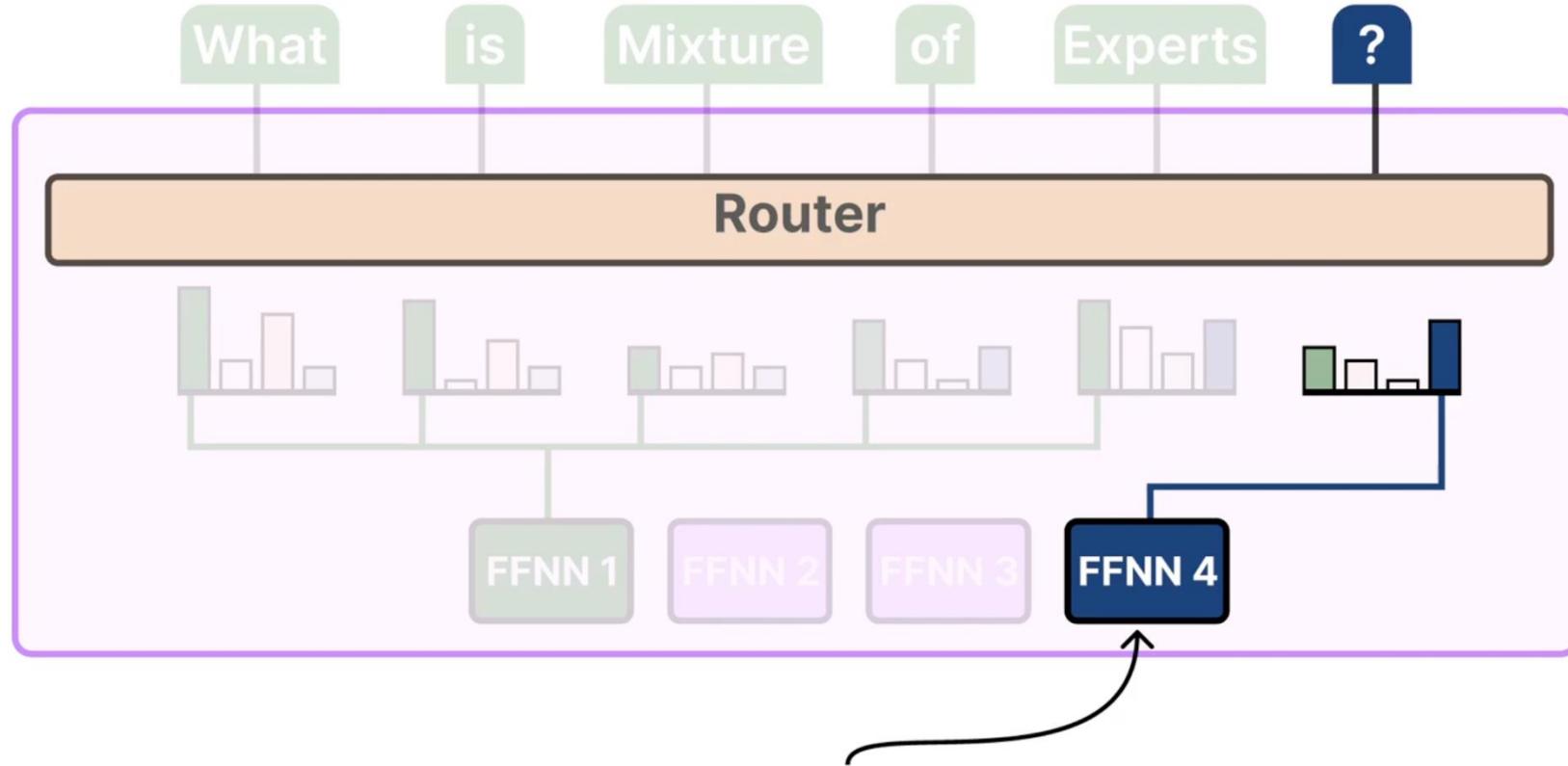
Expert Capacity: 专家容量



As a result, compared to **expert 1**....



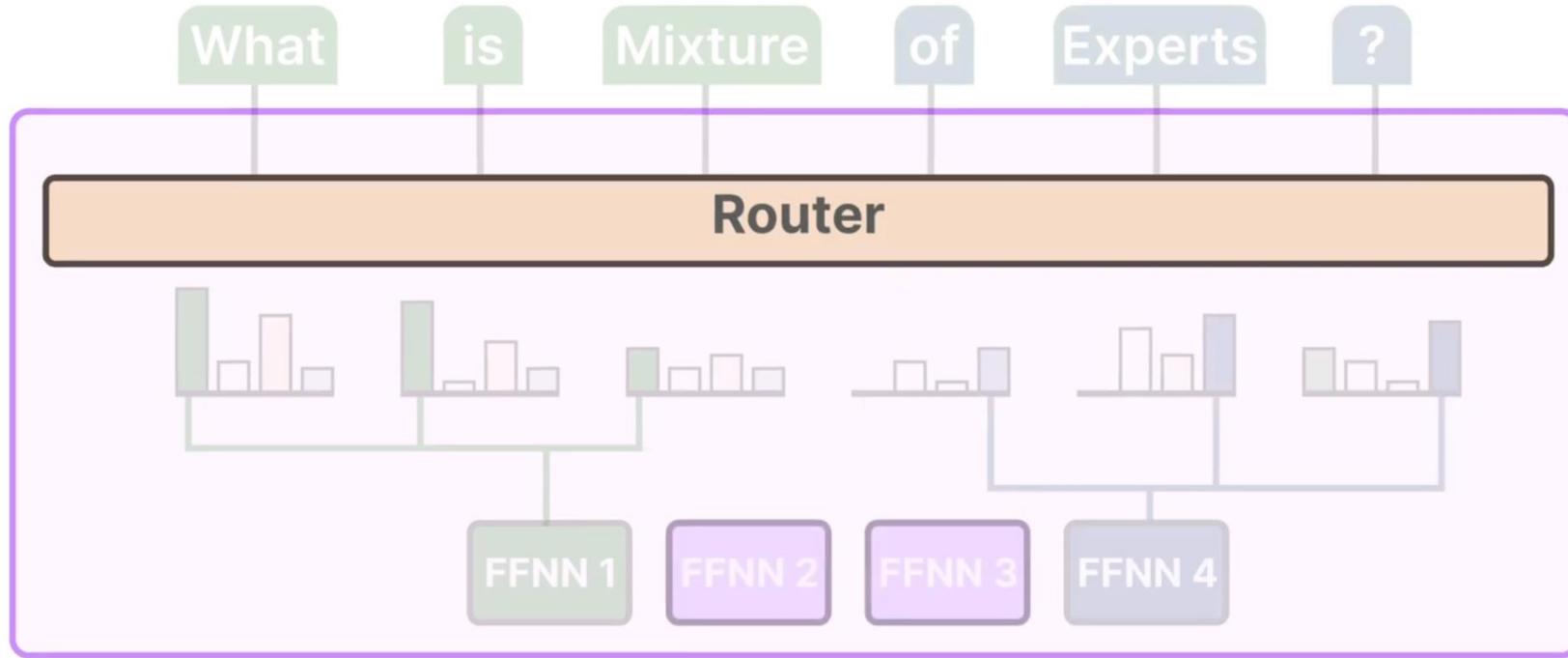
Expert Capacity: 专家容量



...expert 4 ends up undertrained since it receives so few tokens during training.



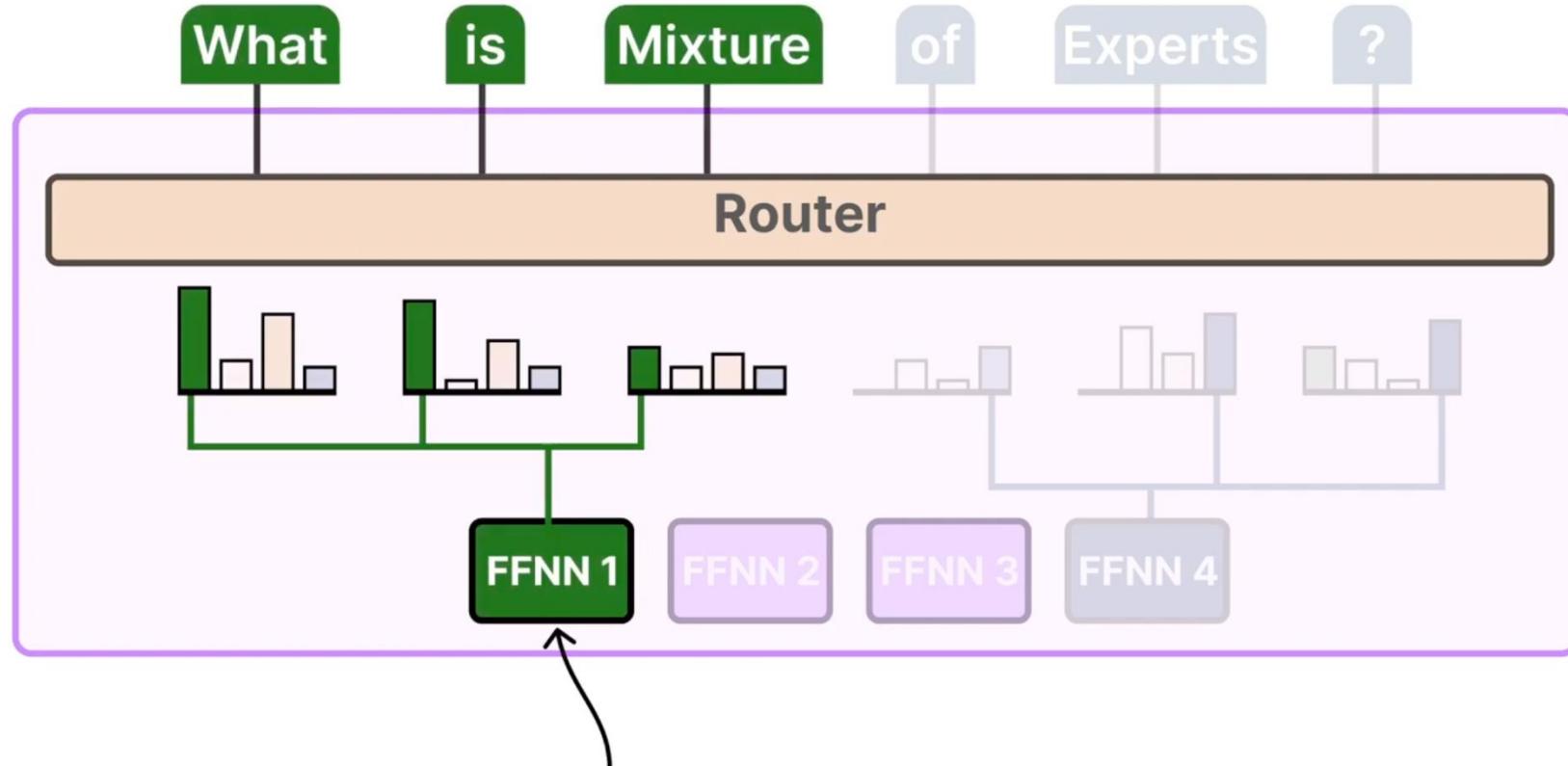
Expert Capacity: 专家容量



To prevent this problem, we **limit** how many tokens they can process, called the **expert capacity**.



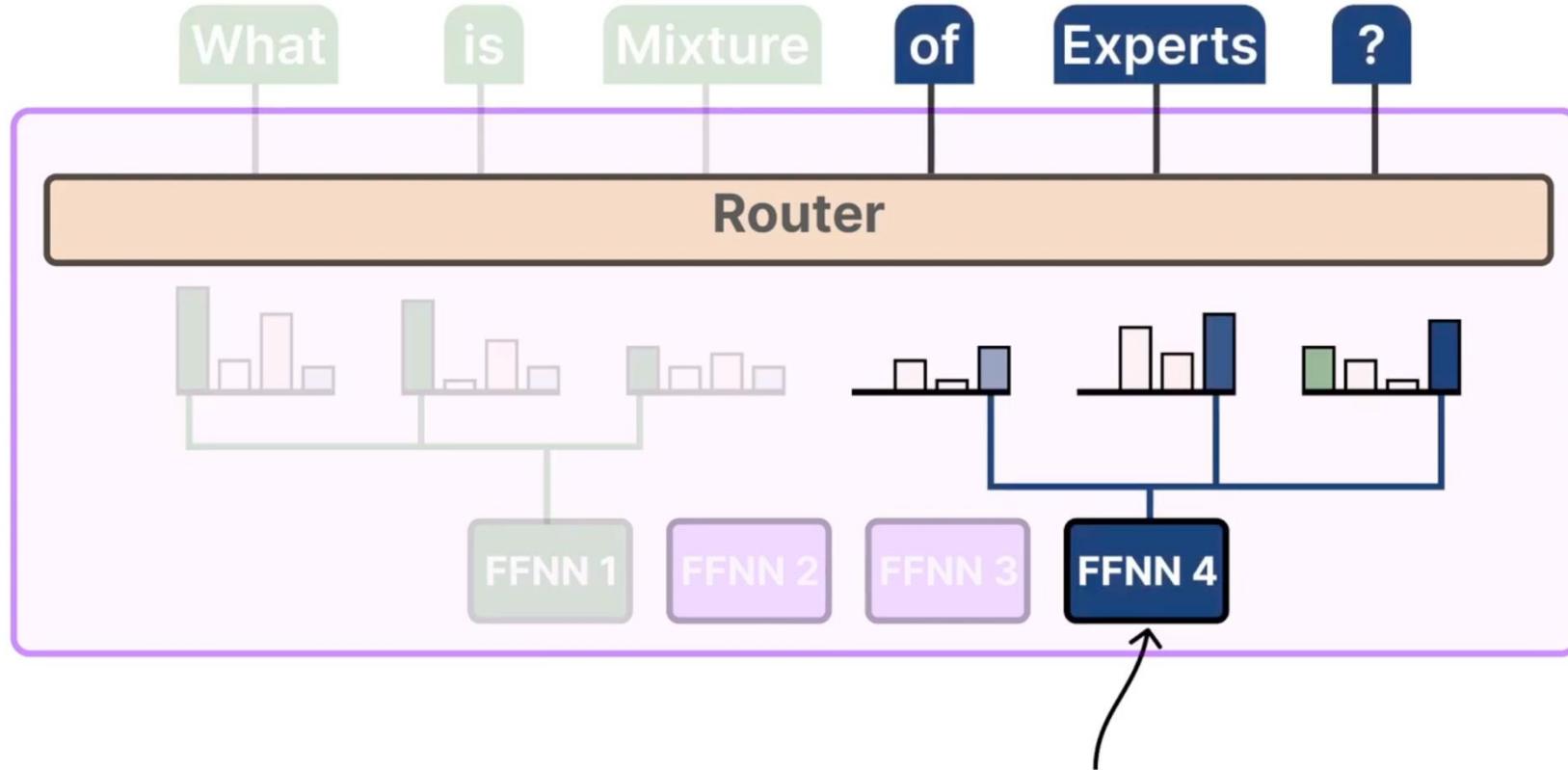
Expert Capacity: 专家容量



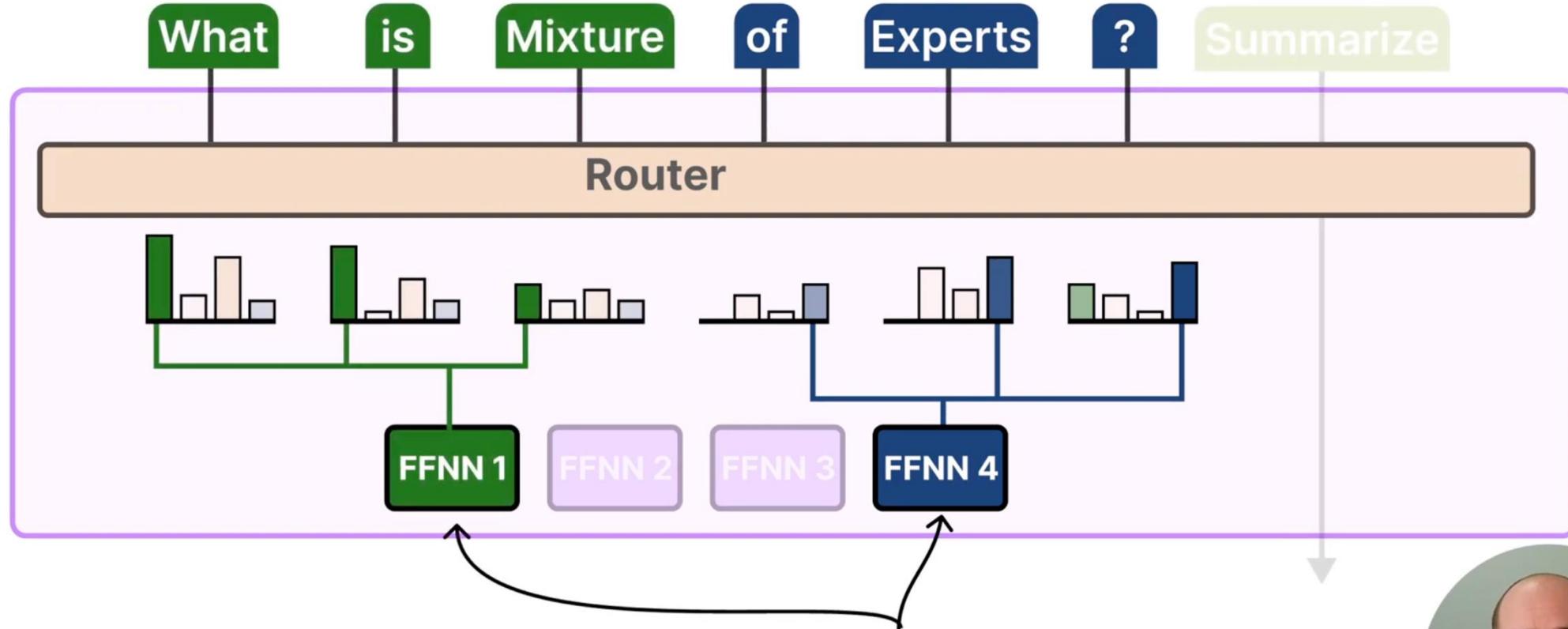
By setting the **expert capacity** to 3, this expert can only process 3 tokens.



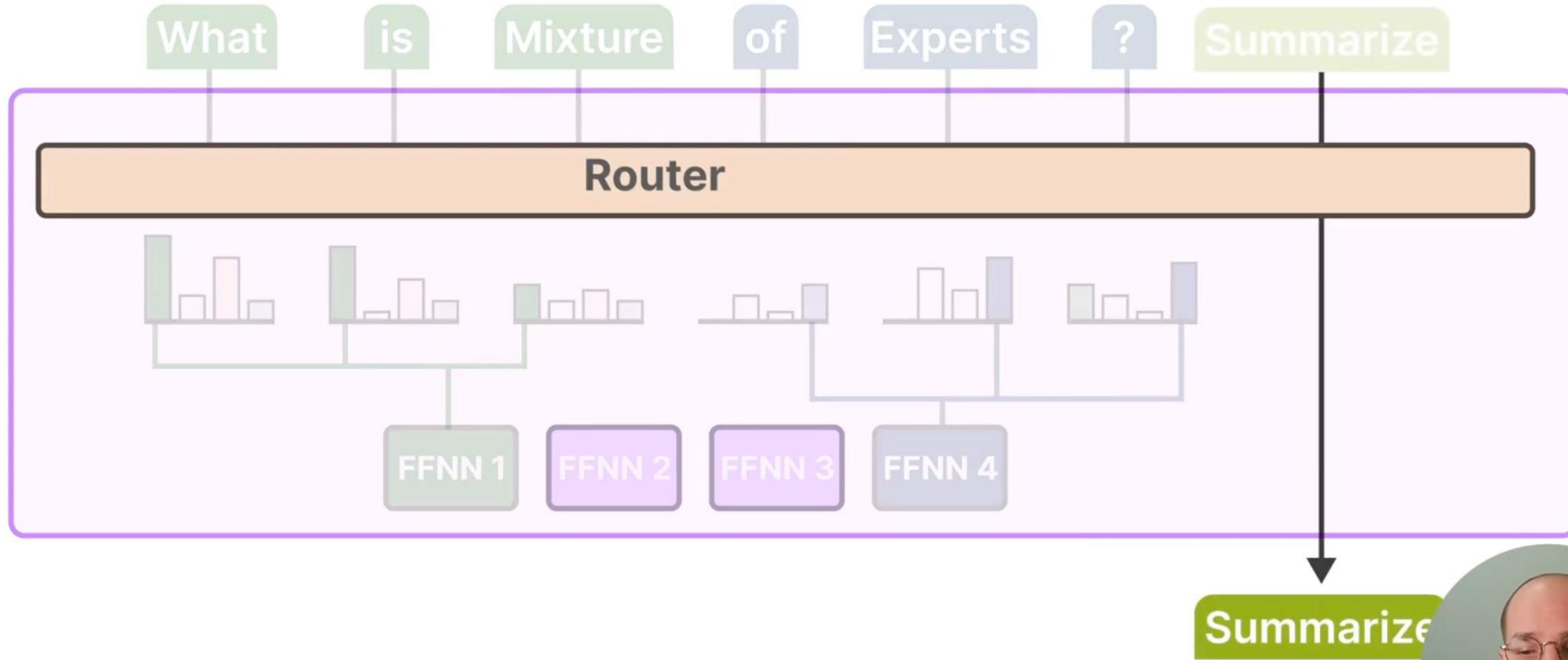
Expert Capacity: 专家容量



Expert Capacity: 专家容量

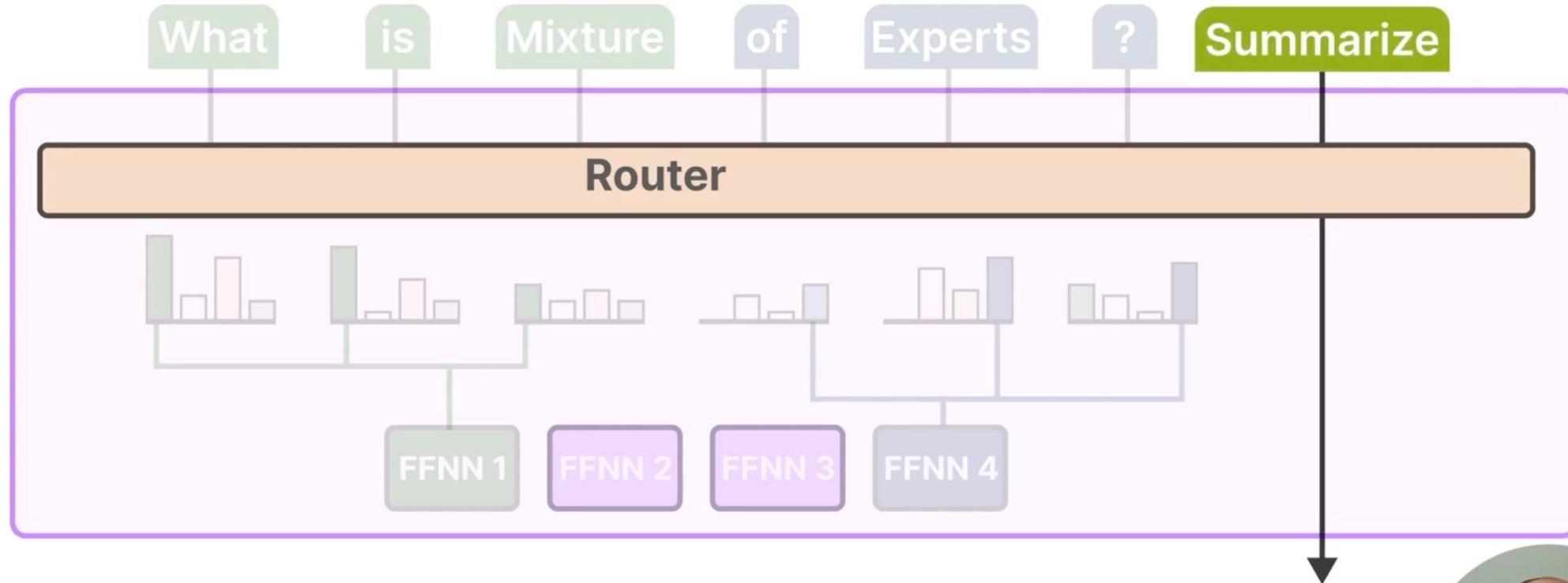


Expert Capacity: 专家容量



... any **new token** will not be processed by any expert but instead sent to the next layer.

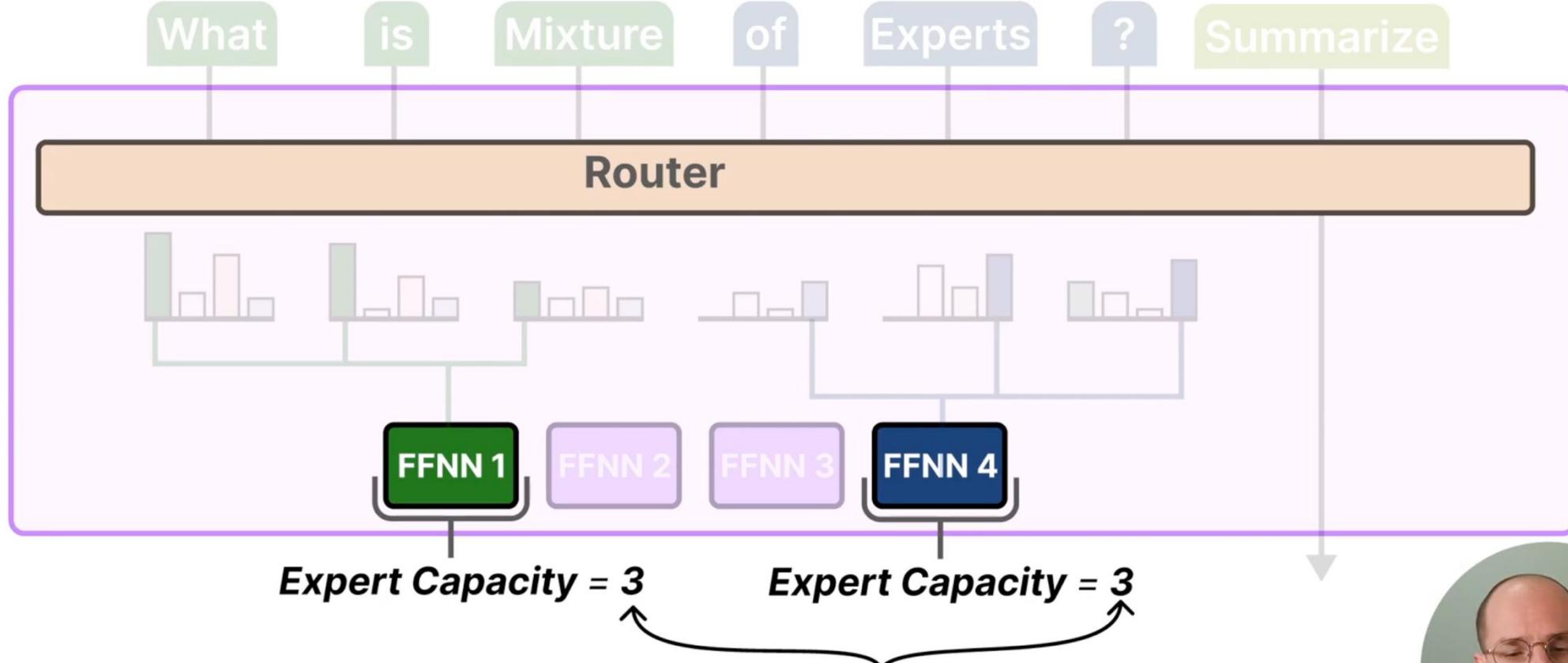
Expert Capacity: 专家容量



This is referred to as **token overflow**.



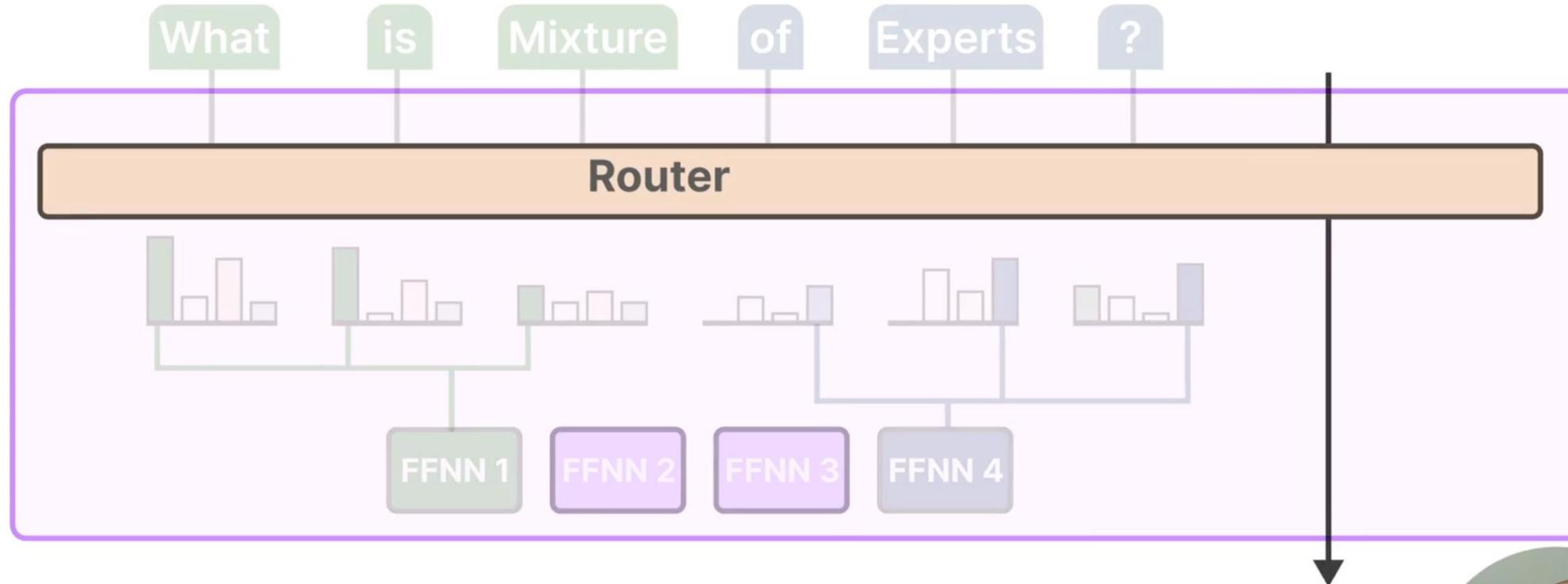
Expert Capacity: 专家容量



Therefore, it is important that we find a balance between
the number of tokens an expert can process...



Expert Capacity: 专家容量



... and how many will be left unprocessed.

Summarize



总结与思考

Switch Transformers



Thank you

把AI系统带入每个开发者、每个家庭、
每个组织，构建万物互联的智能世界

Bring AI System to every person, home and
organization for a fully connected,
intelligent world.

Copyright © 2024 XXX Technologies Co., Ltd.
All Rights Reserved.

The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. XXX may change the information at any time without notice.



ZOMI

GitHub <https://github.com/chenzomi12/Allinfra>

引用与参考

- <https://mp.weixin.qq.com/s/6kzCMsJuavkZPG0YCKgeig>
- https://www.zhihu.com/tardis/zm/art/677638939?source_id=1003
- <https://huggingface.co/blog/zh/moe>
- <https://mp.weixin.qq.com/s/mOrAYo3qEACjSwcRPG7fWw>
- https://mp.weixin.qq.com/s/x39hqf8xn1cUlnxElM0_ww
- <https://mp.weixin.qq.com/s/ZXjwnO103e-wXJGmmKi-Pw>
- <https://mp.weixin.qq.com/s/8Y281VYaLu5jHoAvQVvVJg>
- https://blog.csdn.net/weixin_43013480/article/details/139301000
- <https://developer.nvidia.com/zh-cn/blog/applying-mixture-of-experts-in-lilm-architectures/>
- <https://www.zair.top/post/mixture-of-experts/>
- <https://my.oschina.net/IDP/blog/16513157>
- PPT 开源：<https://github.com/chenzomi12/Allinfra>
- 夸克链接：<https://pan.quark.cn/s/74fb24be8eff>

