

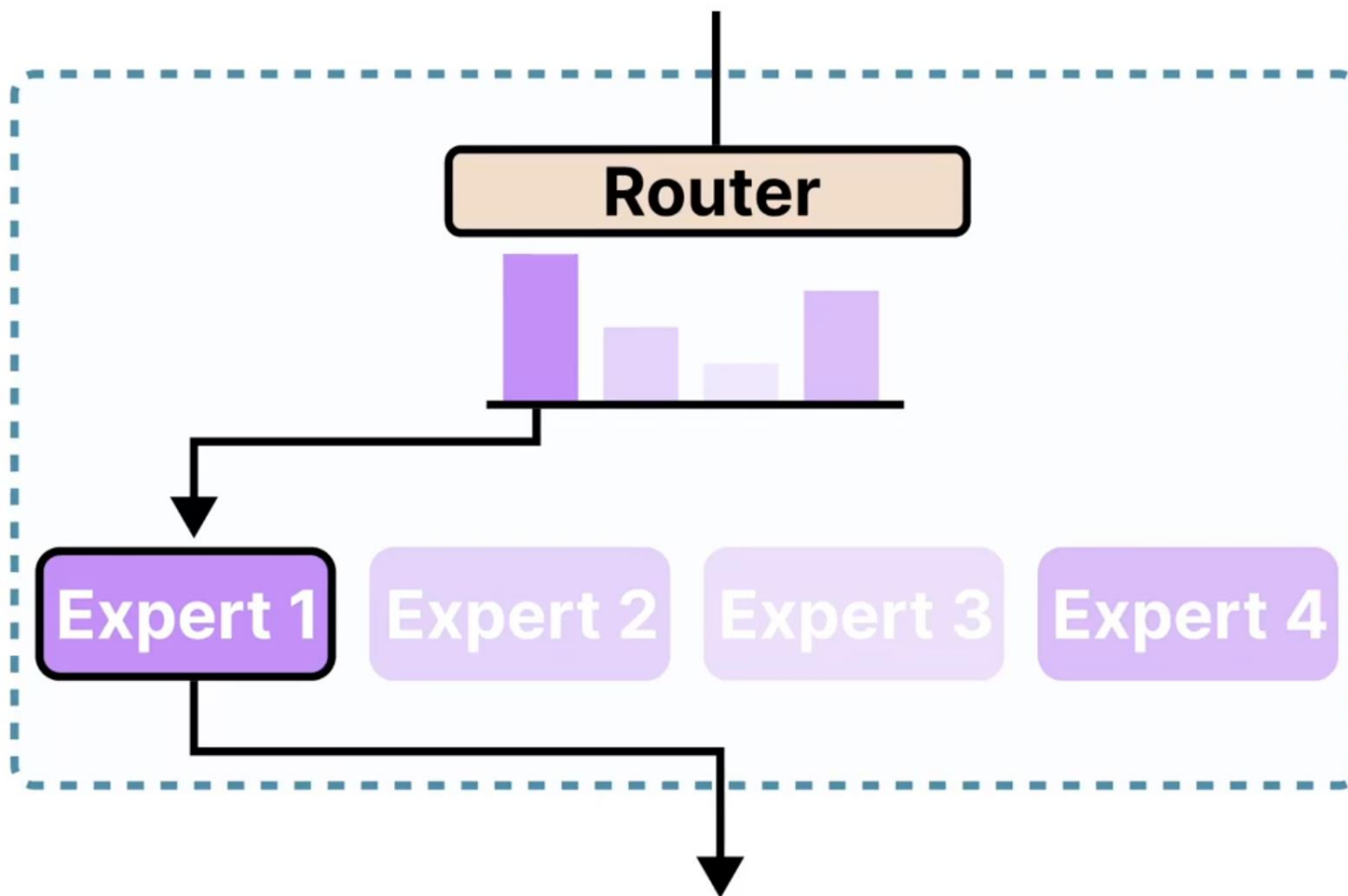
Mixture of Experts (MoE)



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

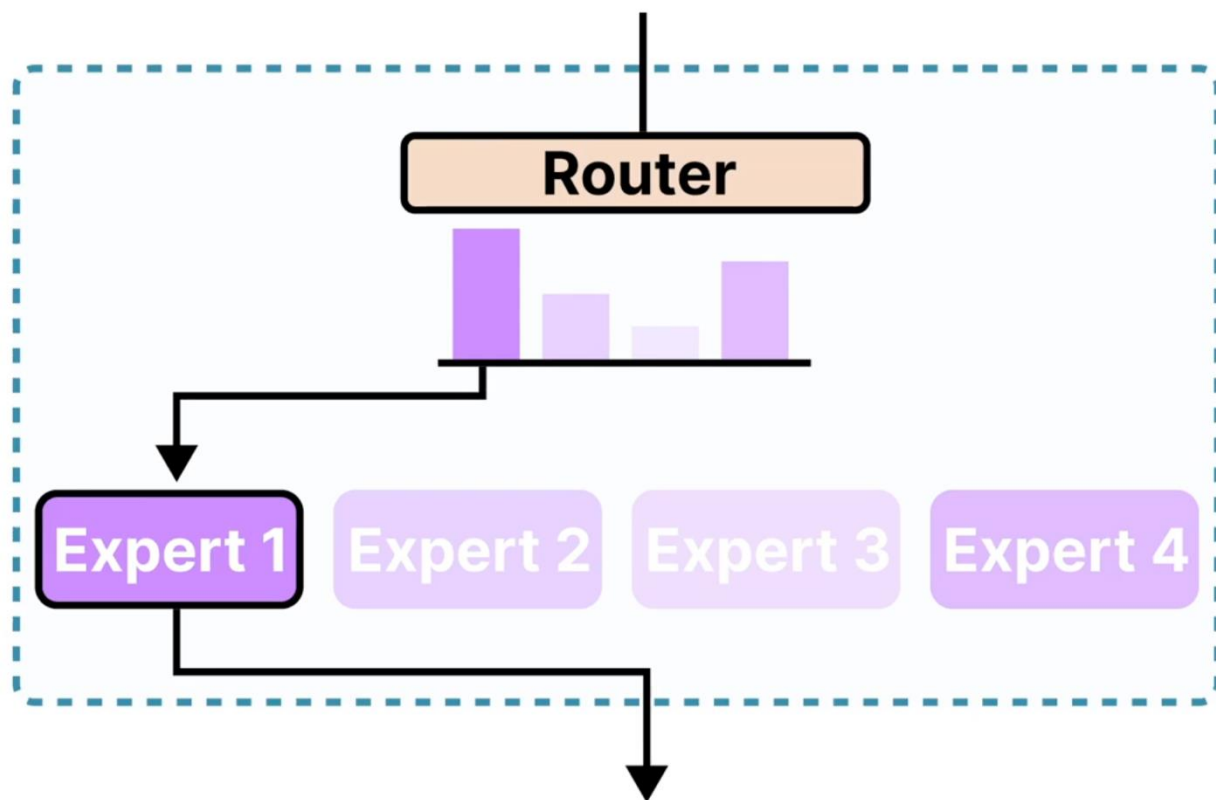


01

Introduction: MOE 基本原理



Introduction: MOE 基本原理

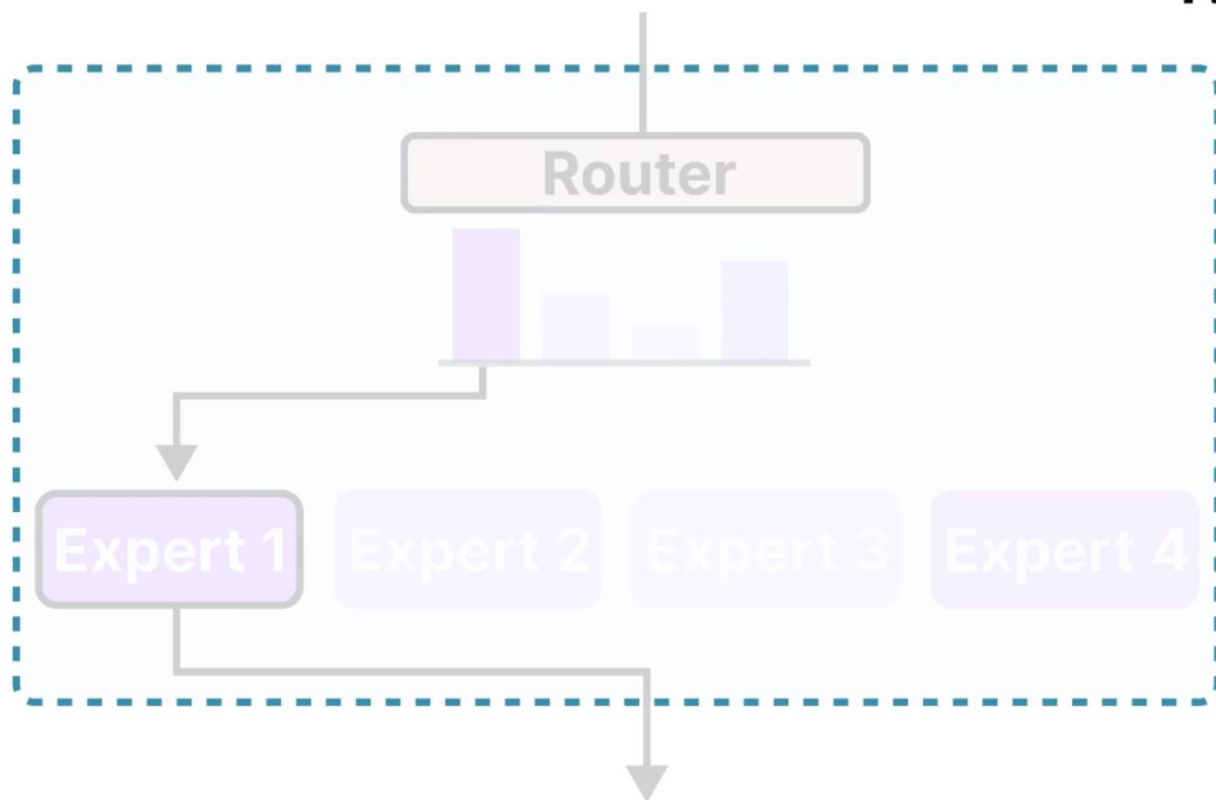


Mixture of Experts (MoE) is a technique that uses different sub-models (“**experts**”) to improve the quality of **LLMs**.



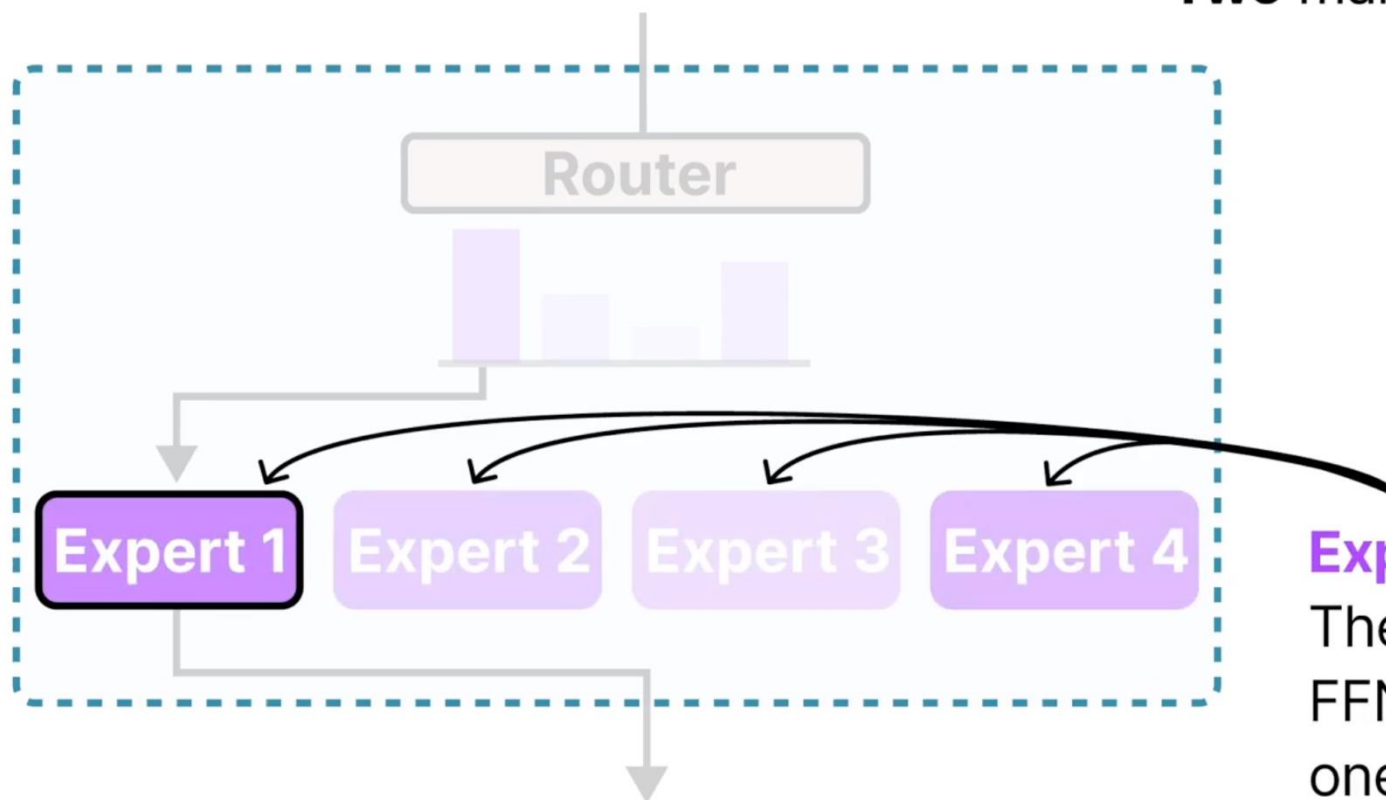
Introduction: MOE 基本原理

Two main components define a **MoE**:



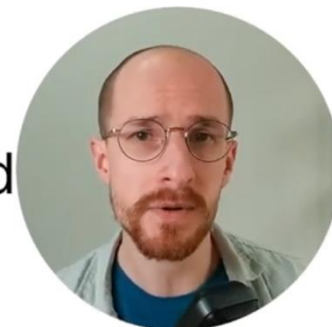
Introduction: MOE 基本原理

Two main components define a **MoE**:



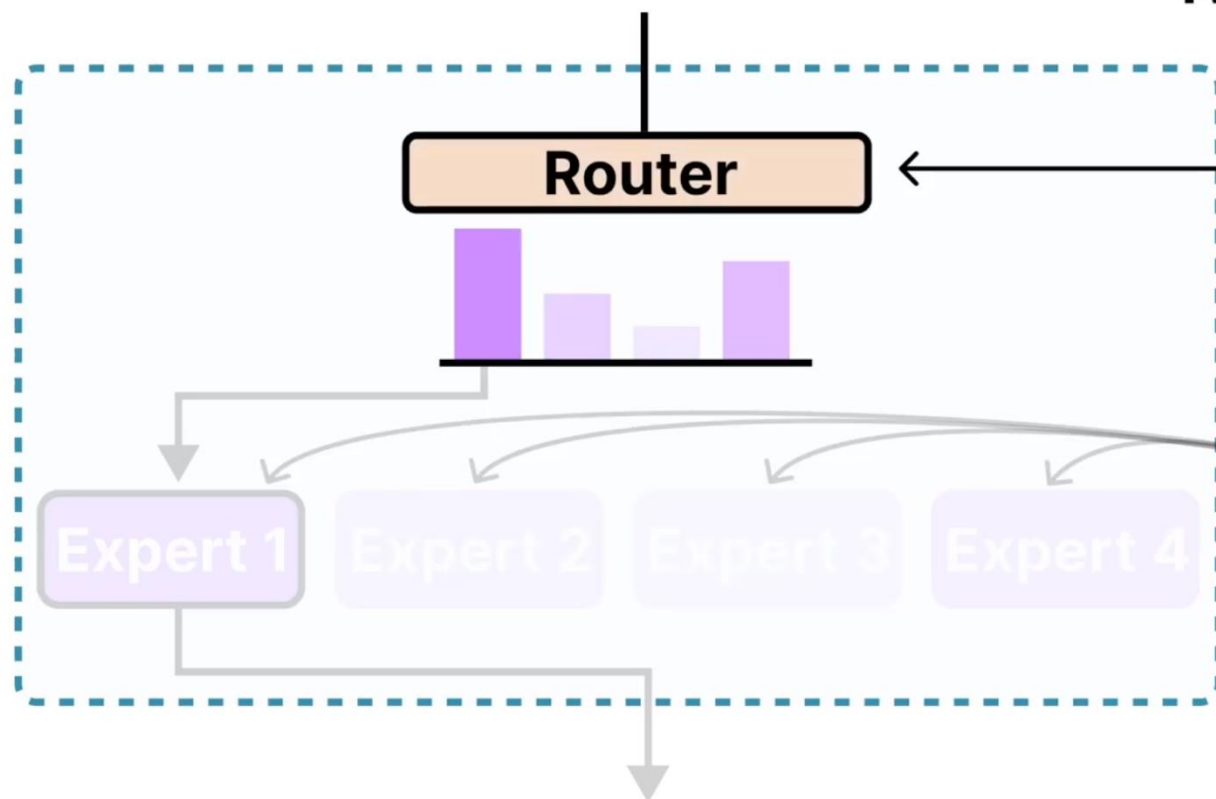
Experts

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



Introduction: MOE 基本原理

Two main components define a **MoE**:

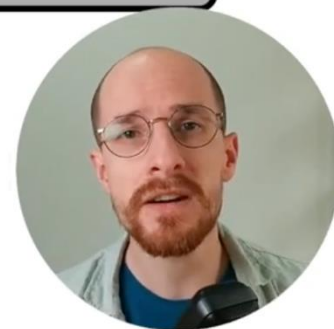


Router (gate network)
Determines which **tokens** are sent to which experts.

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

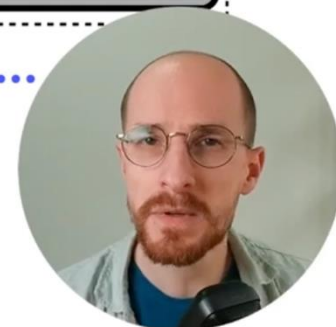
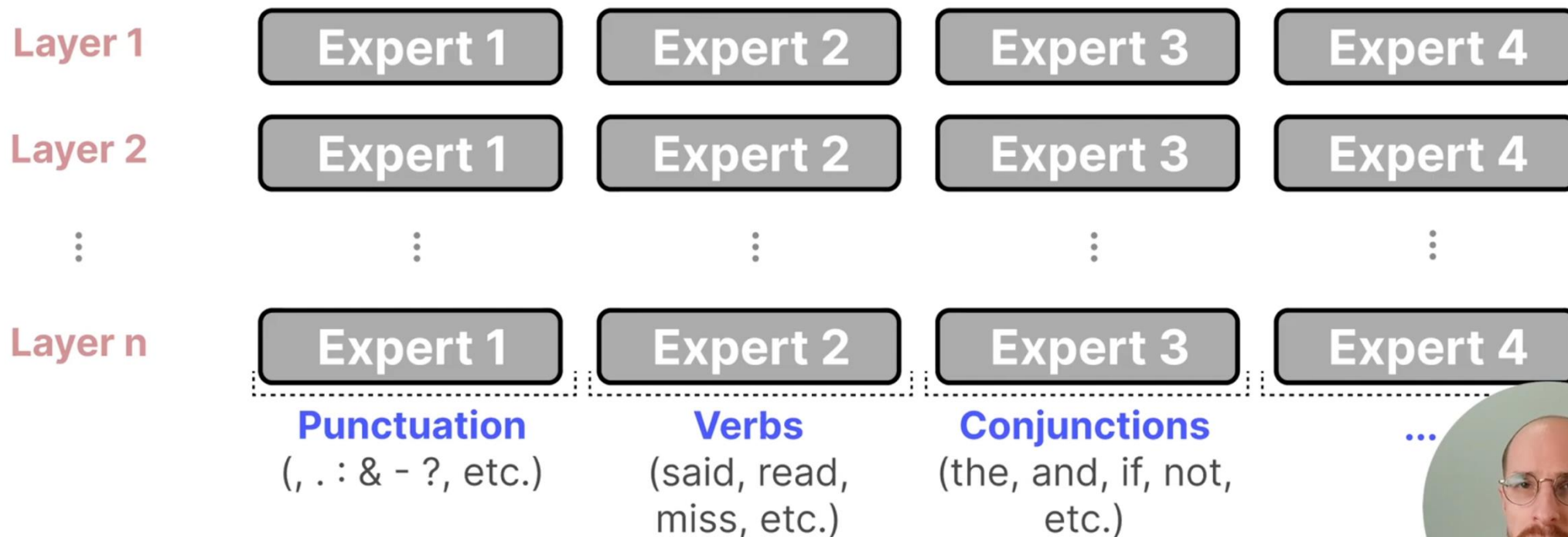


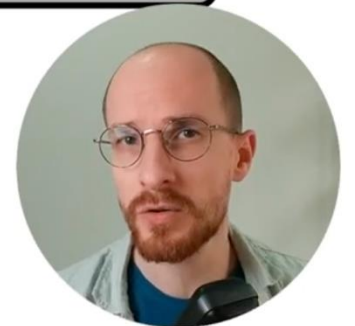
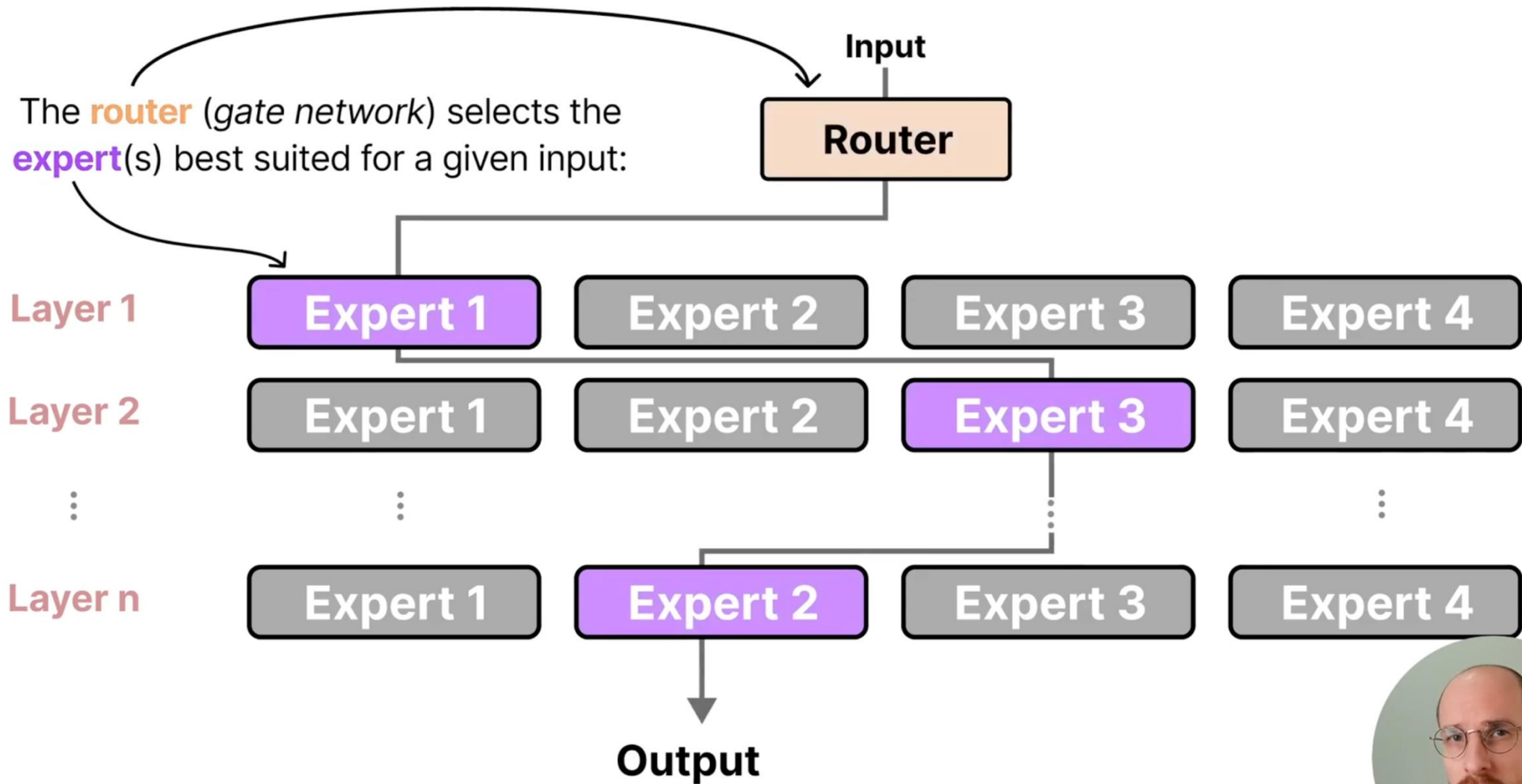
Introduction: MOE 基本原理



Introduction: MOE 基本原理

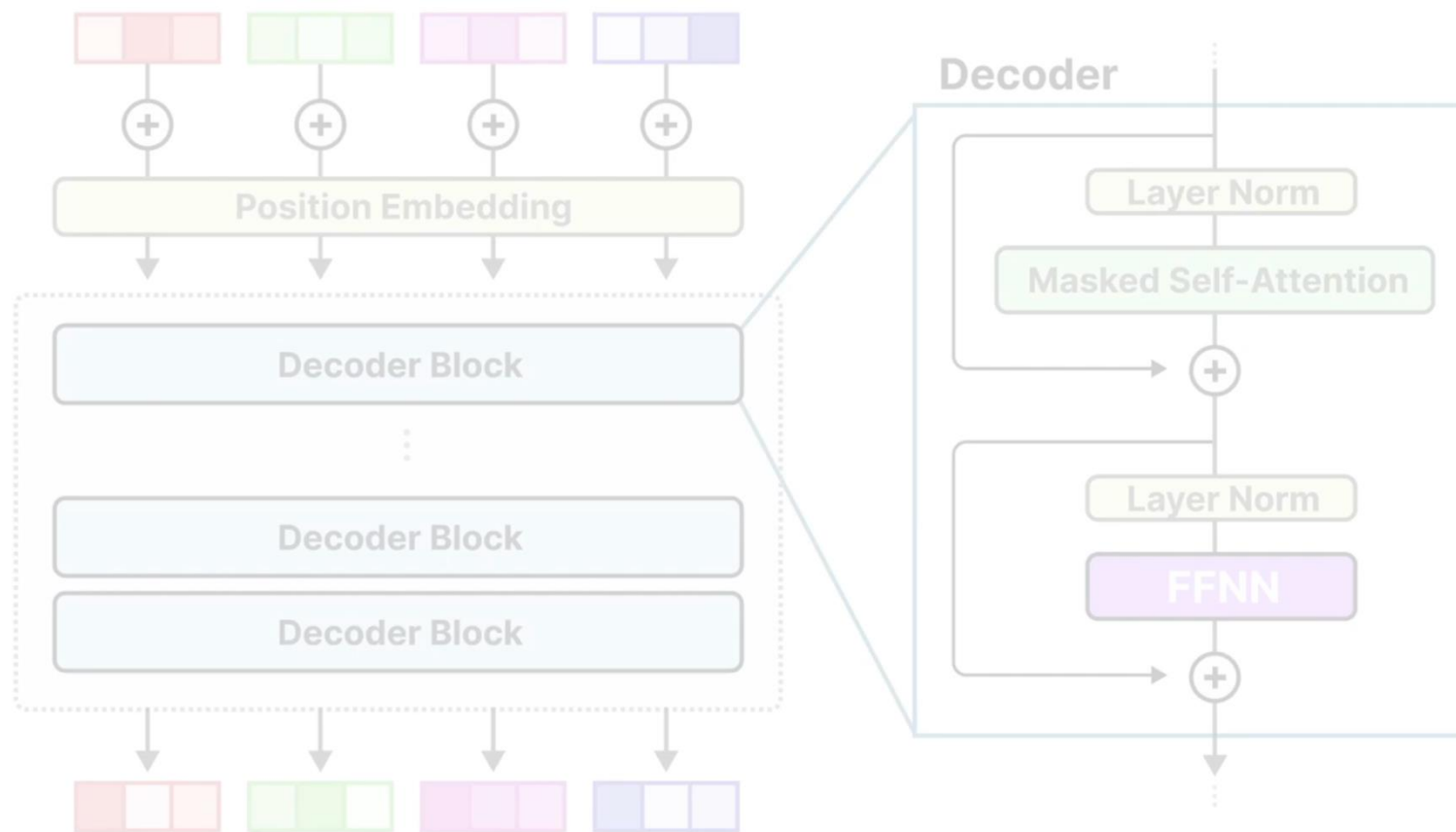
Know that an “**expert**” is not specialized in a specific domain like “**Psychology**” or “**Biology**”. At most, it learns **syntactic information** on a **token** level instead.





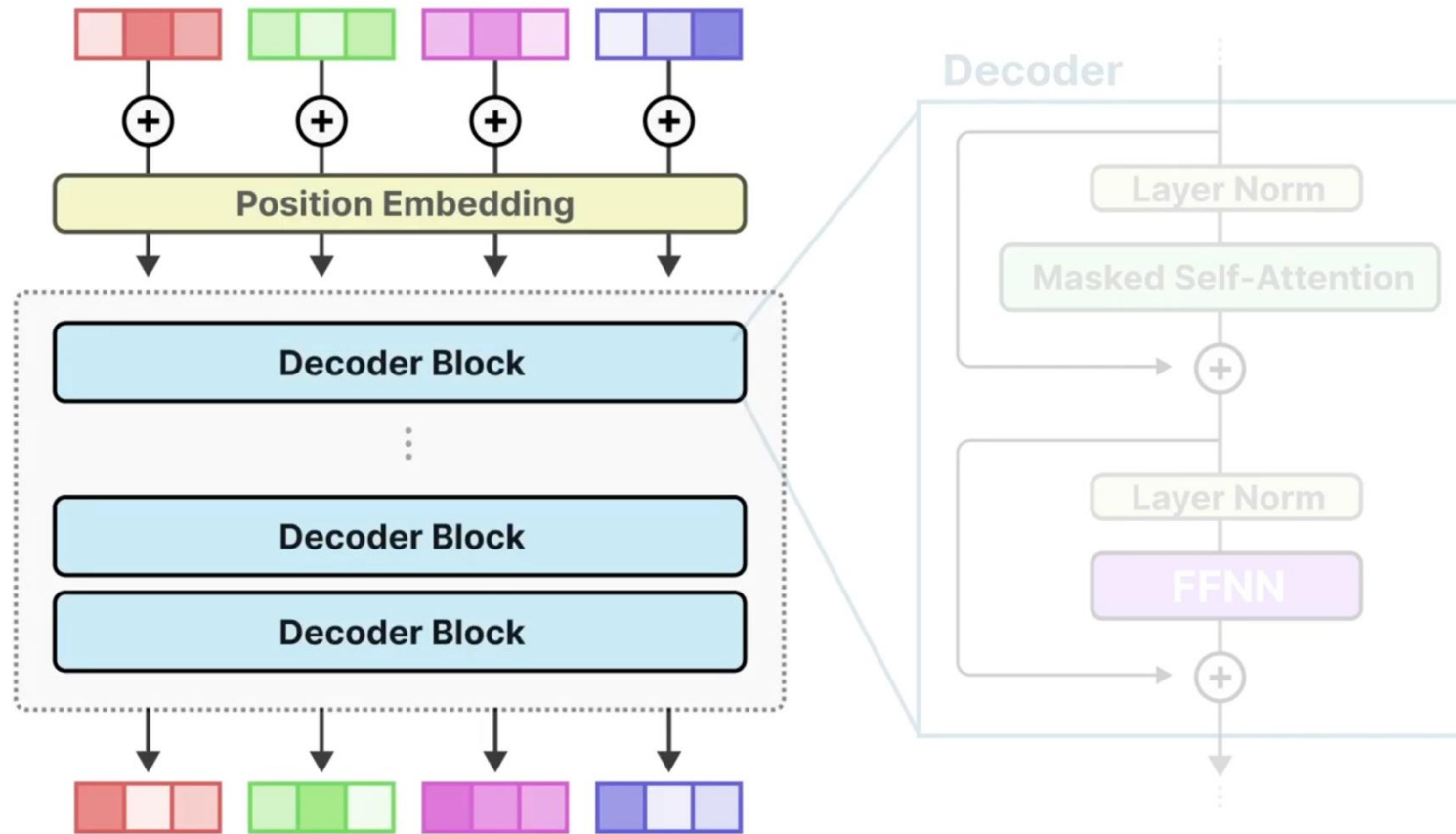
Introduction: MOE 基本原理

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



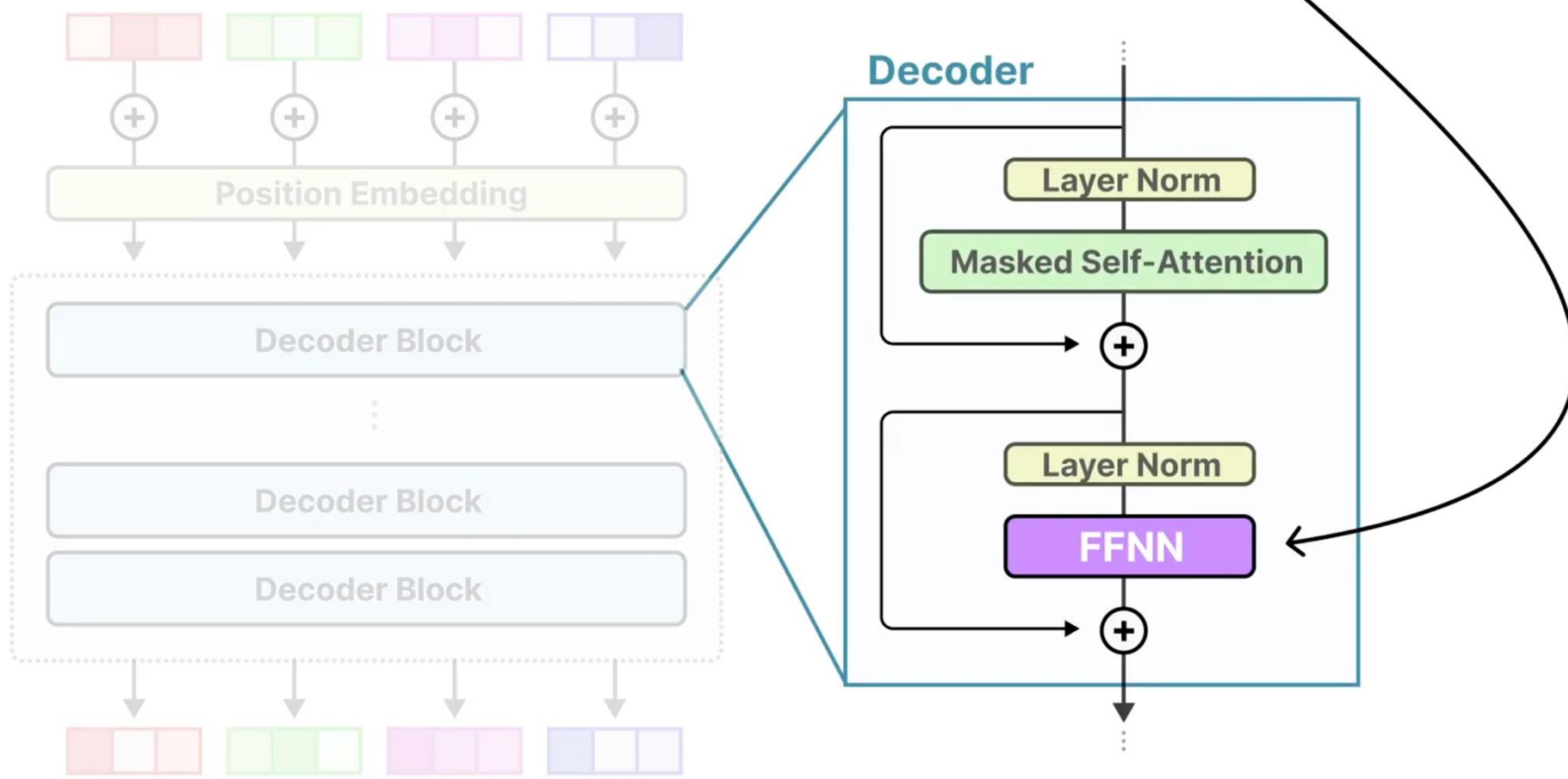
Dense Layers

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

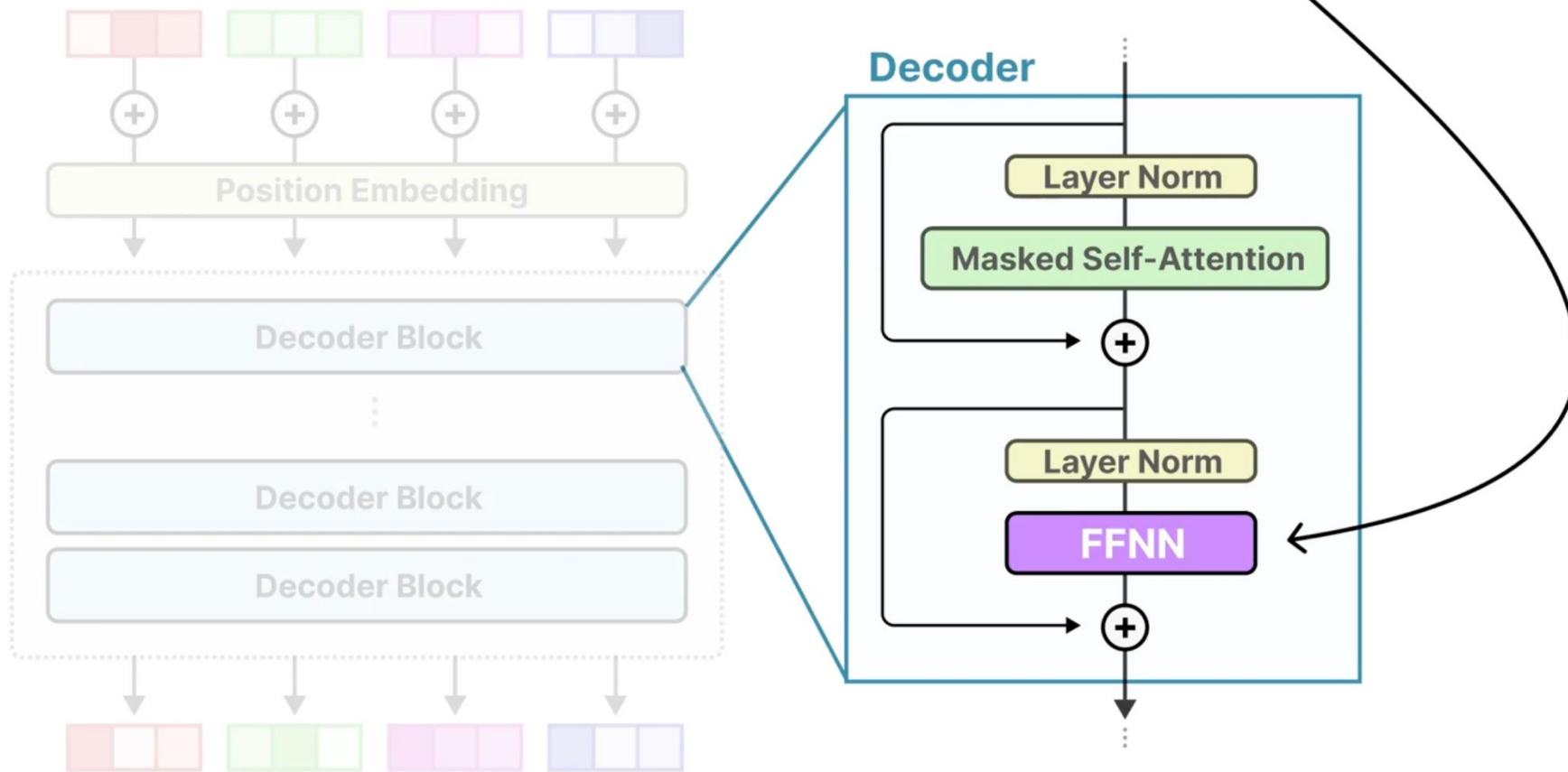


Introduction: MOE 基本原理

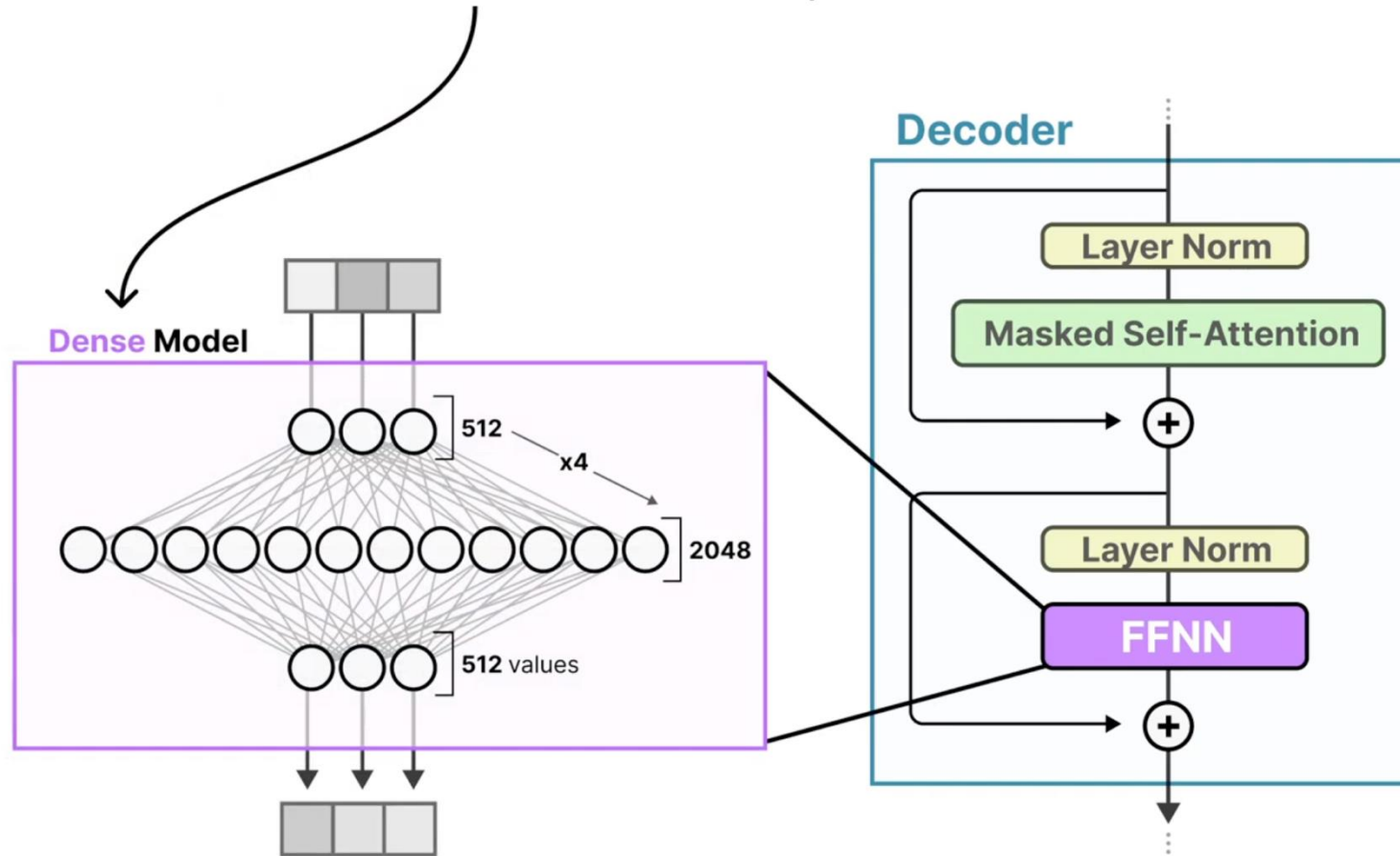
...has the **Feed Forward Neural Network (FFNN)** applied after layer normalization.



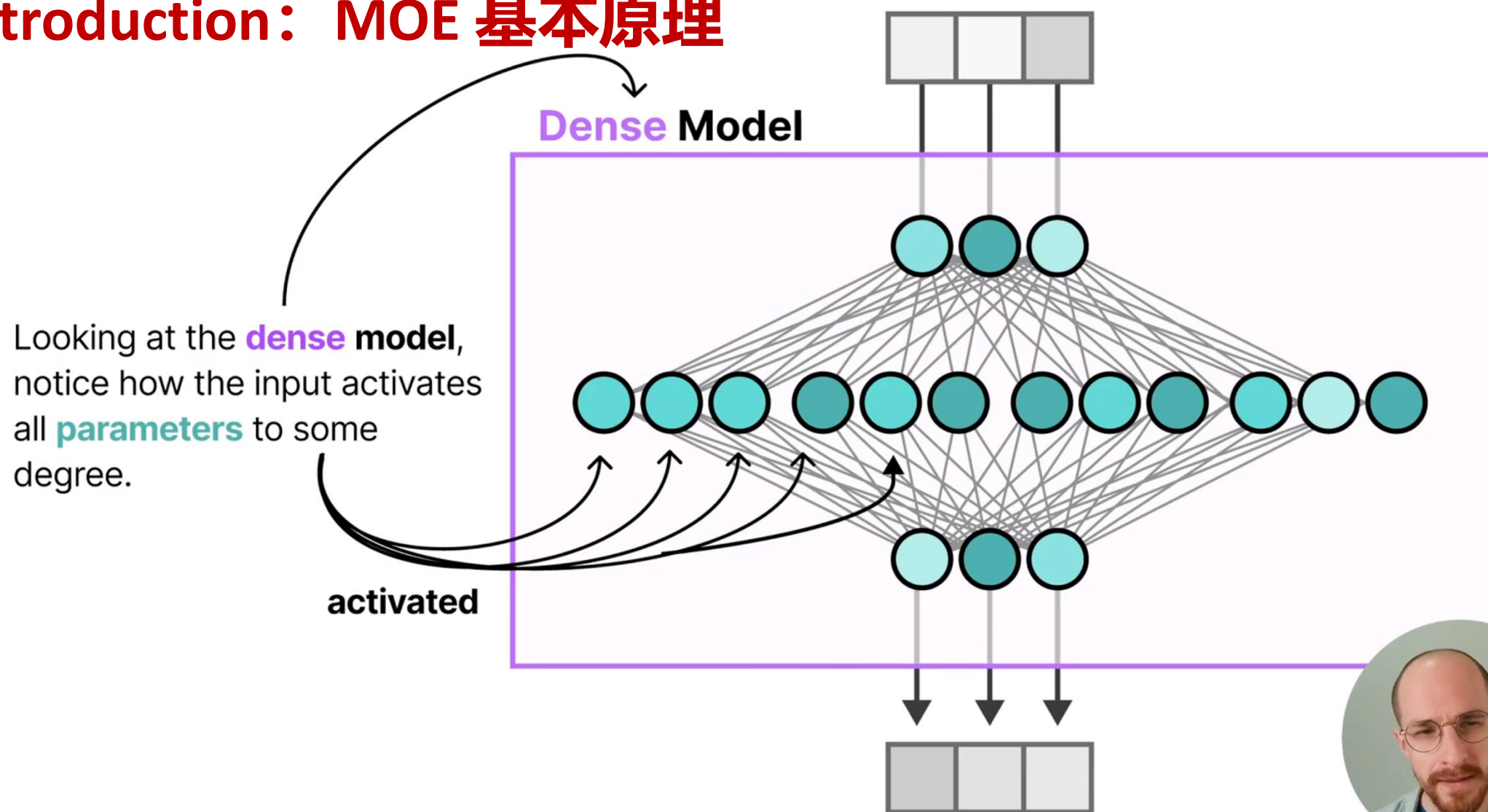
The **FFNN** uses the contextual information created by attention to capture complex relationships in the data.



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

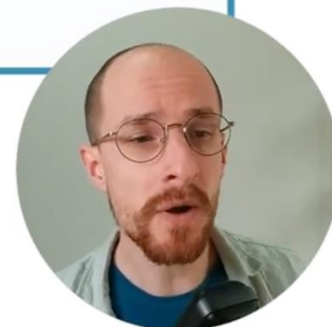
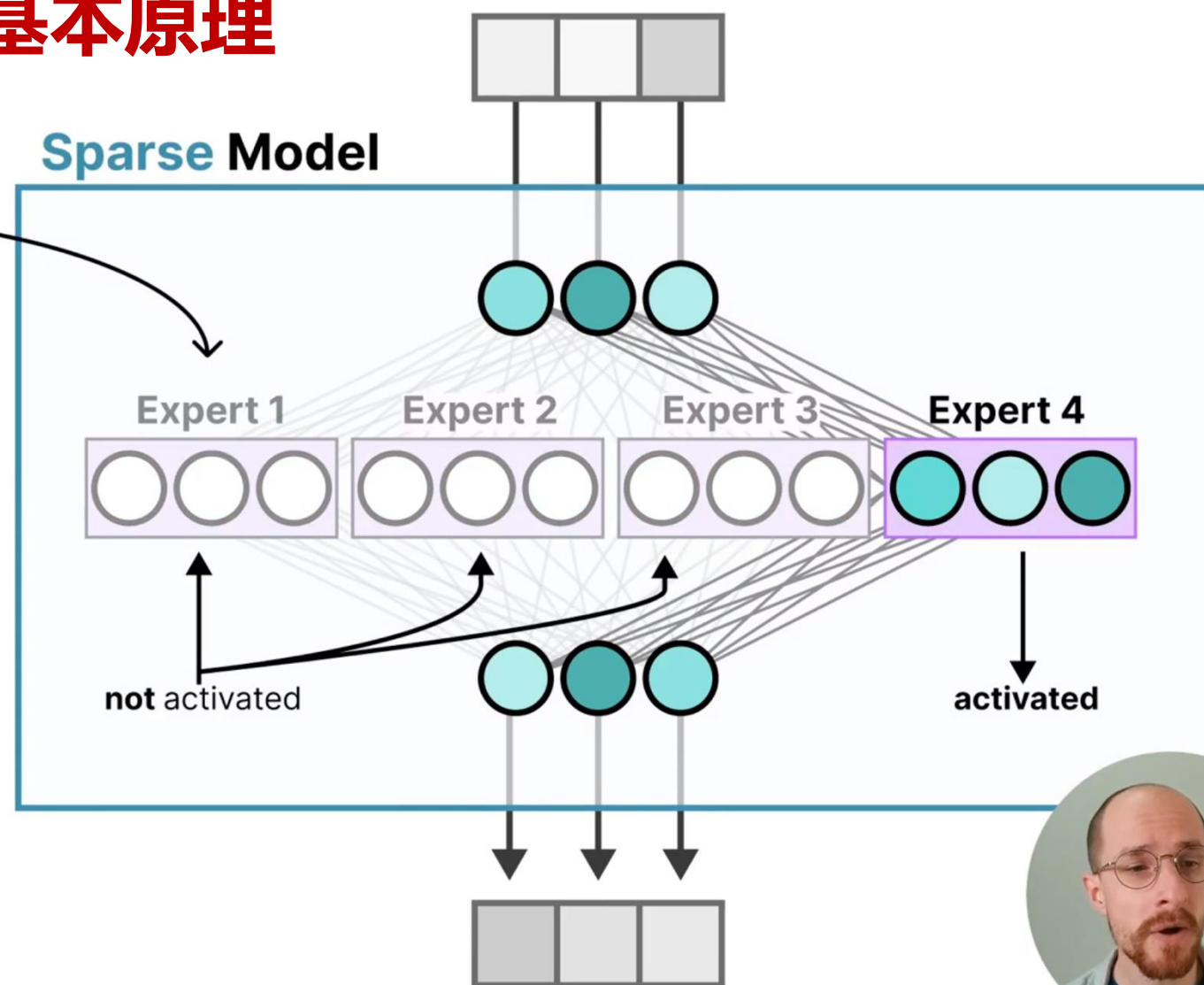


Introduction: MOE 基本原理



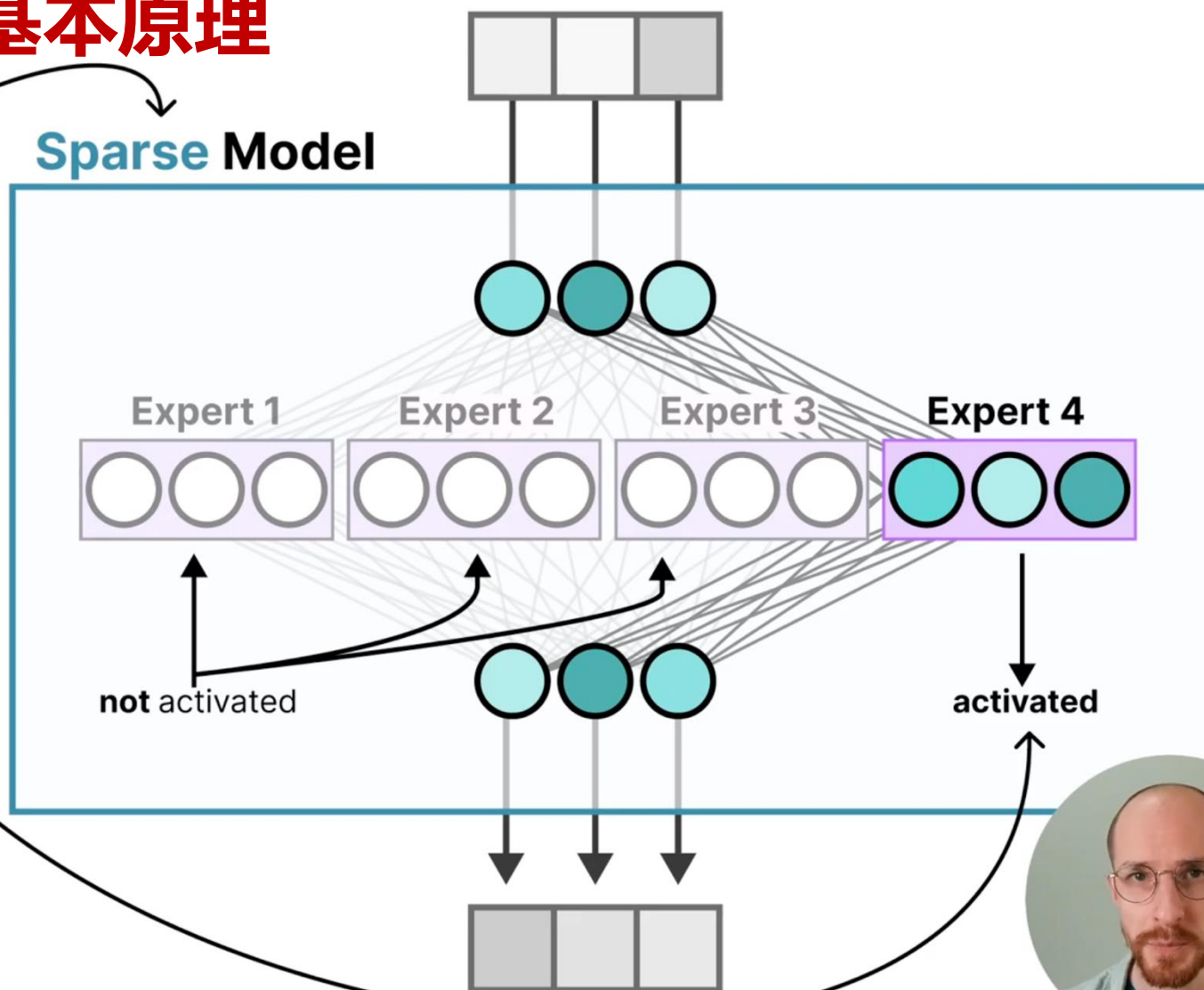
Introduction: MOE 基本原理

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.



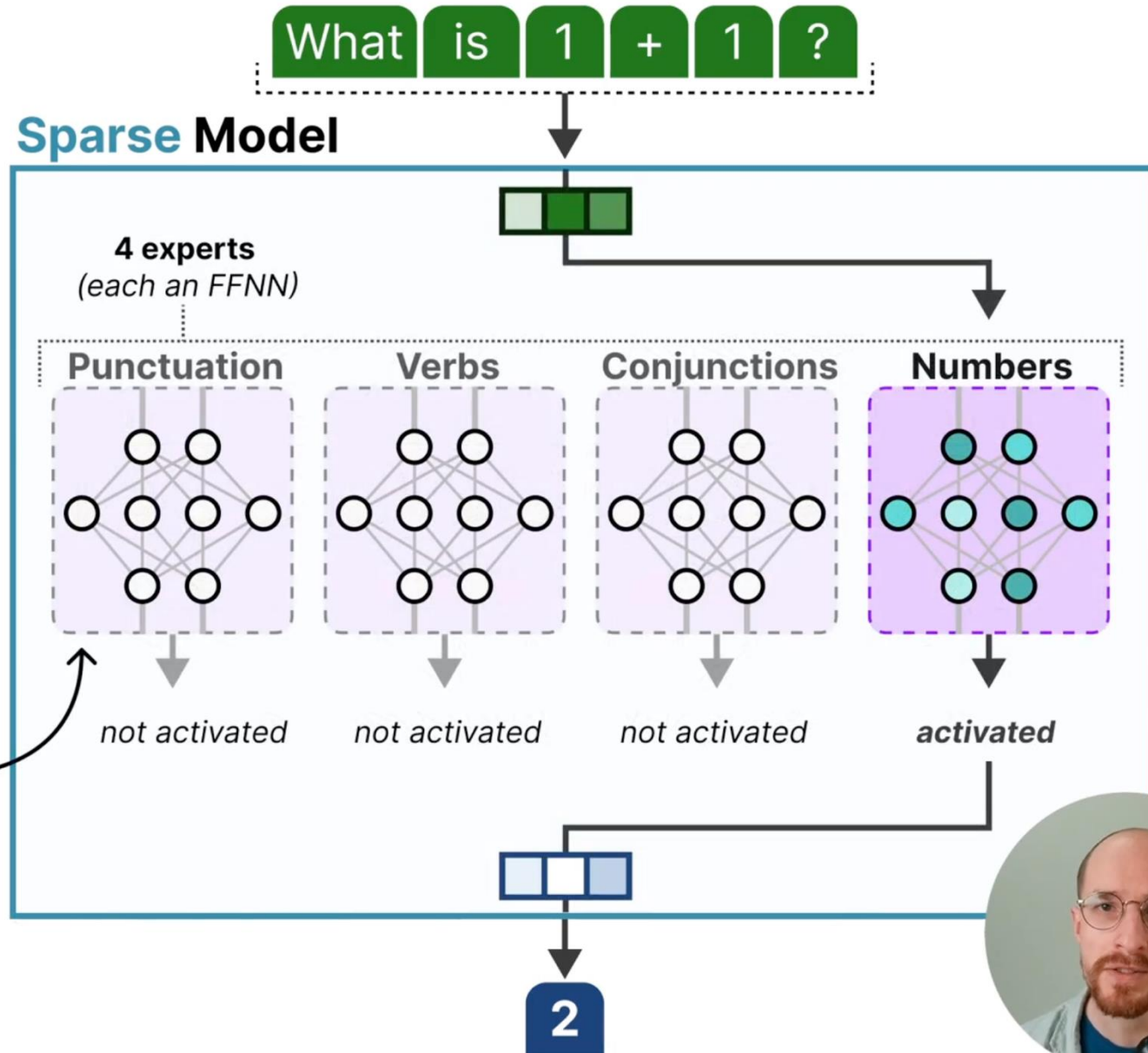
Introduction: MOE 基本原理

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

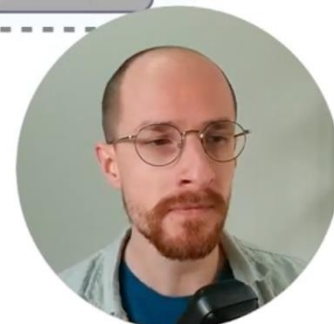
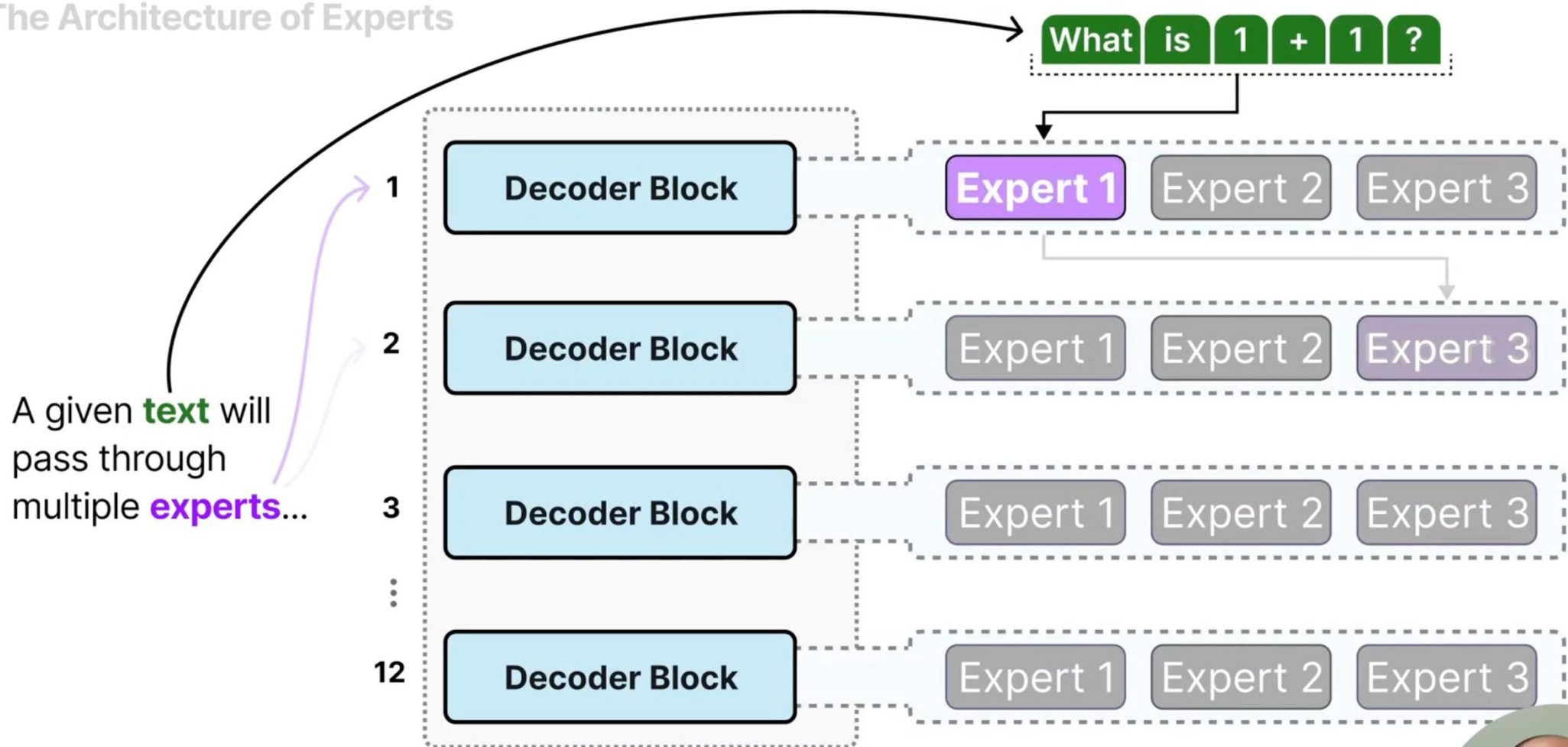


Sparse Layers

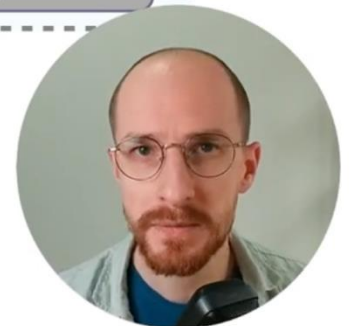
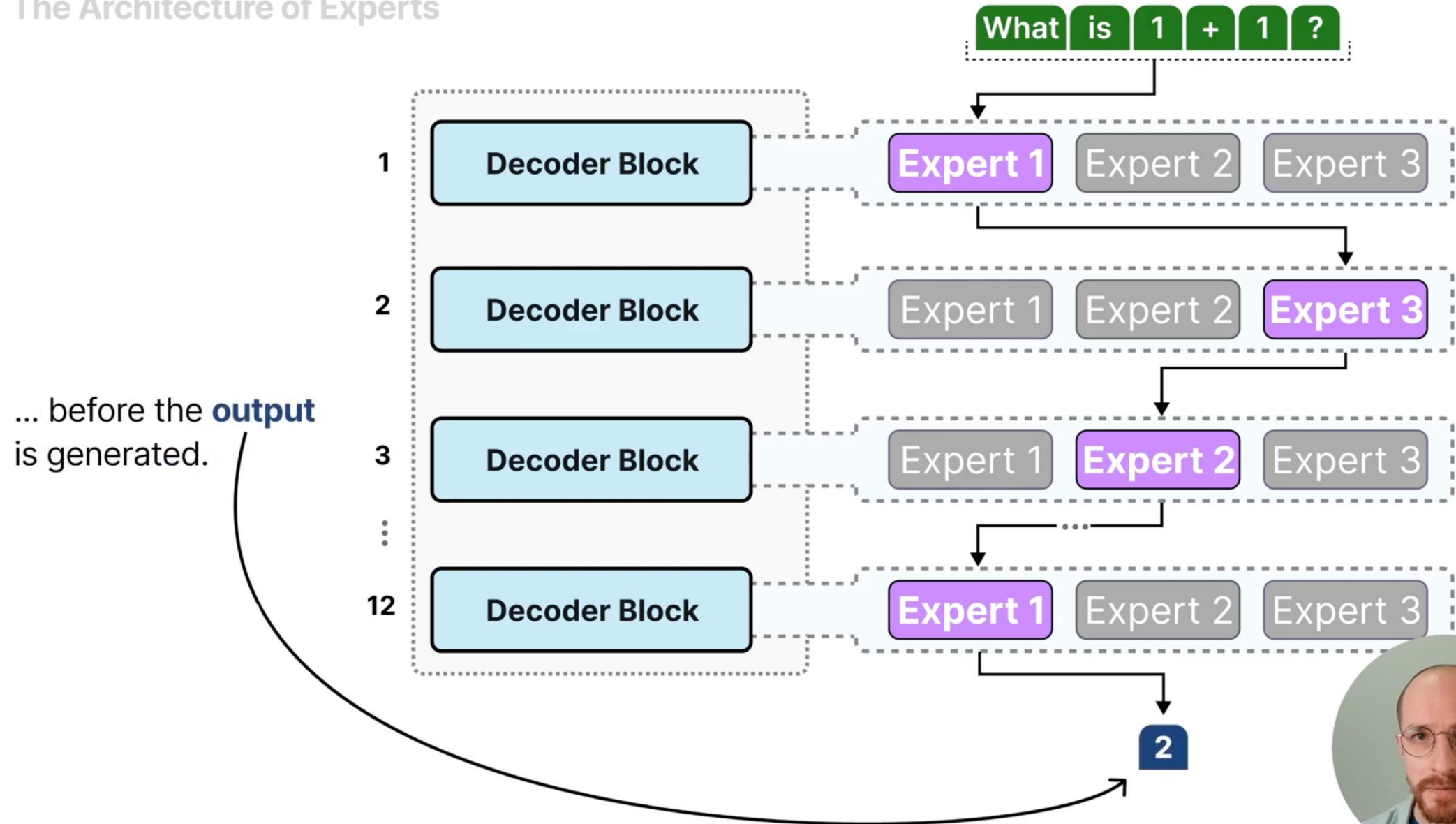
In practice, experts are typically **whole FFNNs** themselves...



The Architecture of Experts

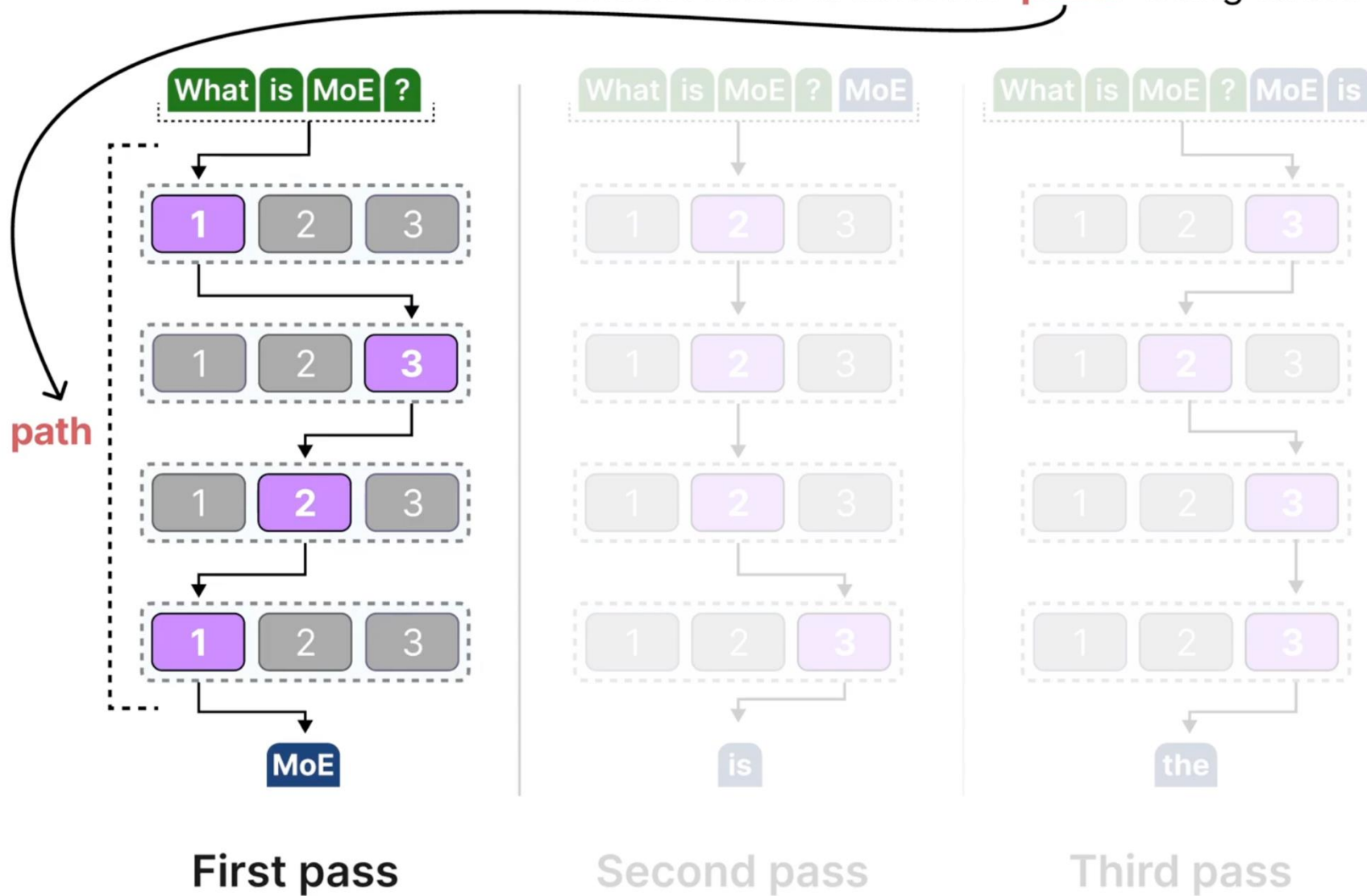


The Architecture of Experts



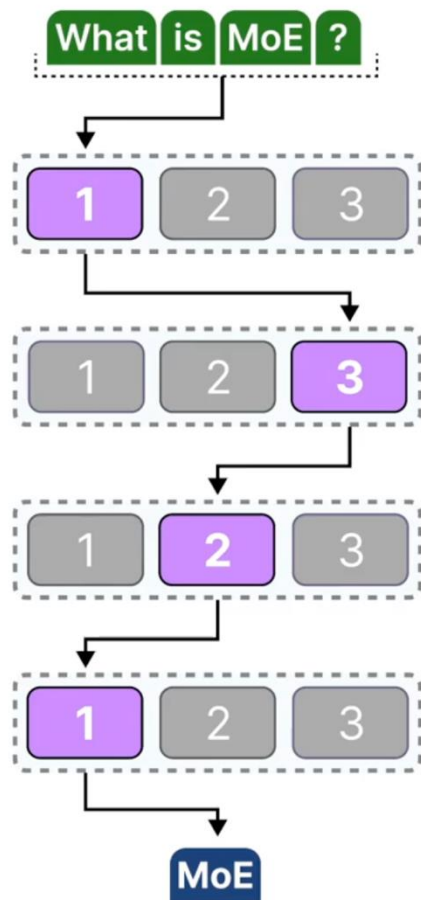
The Architecture of Experts

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

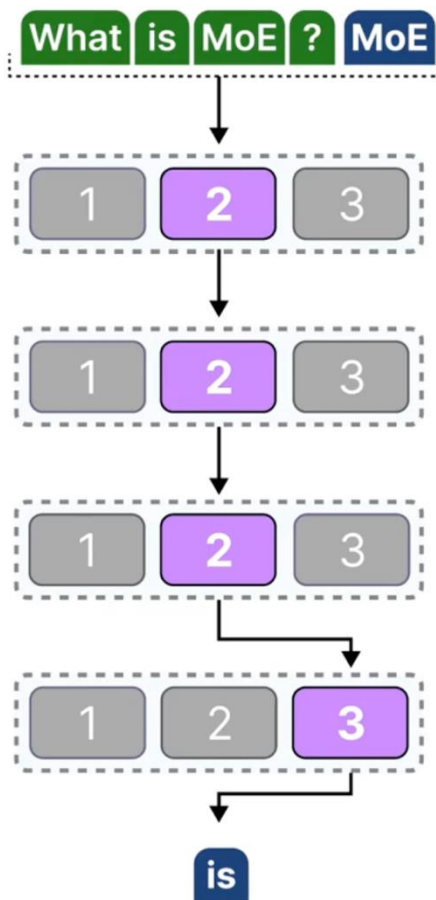


The Architecture of Experts

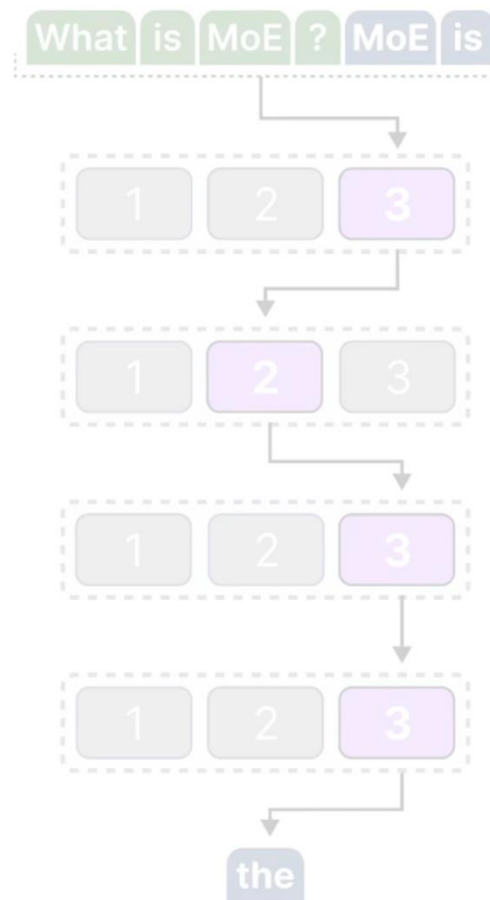
Each **new token** may result in a different path...



First pass



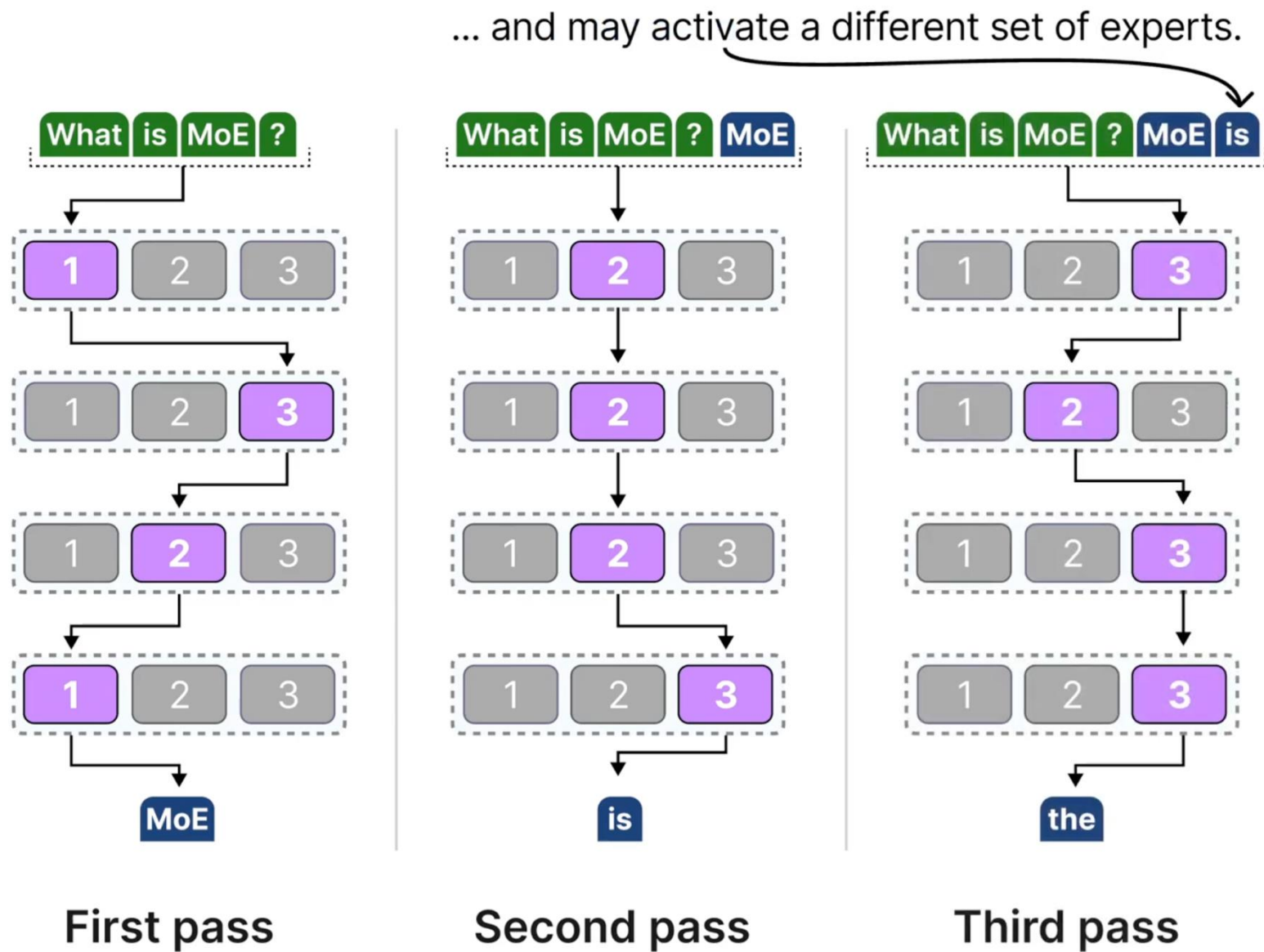
Second pass



Third pass

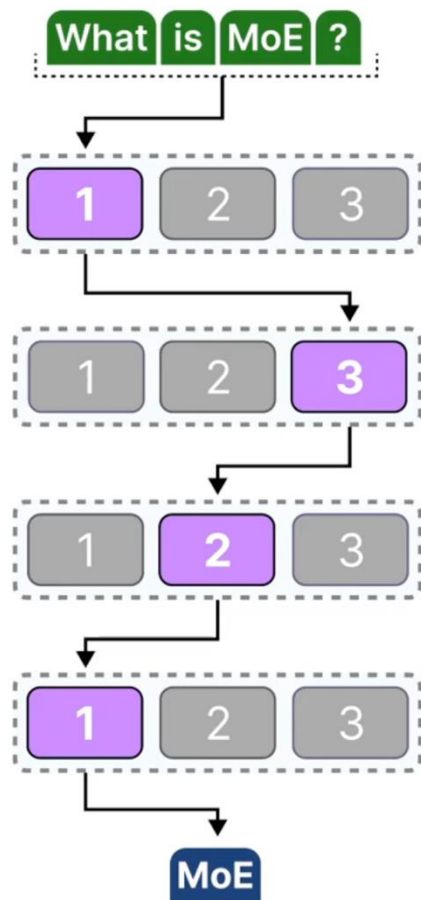


The Architecture of Experts

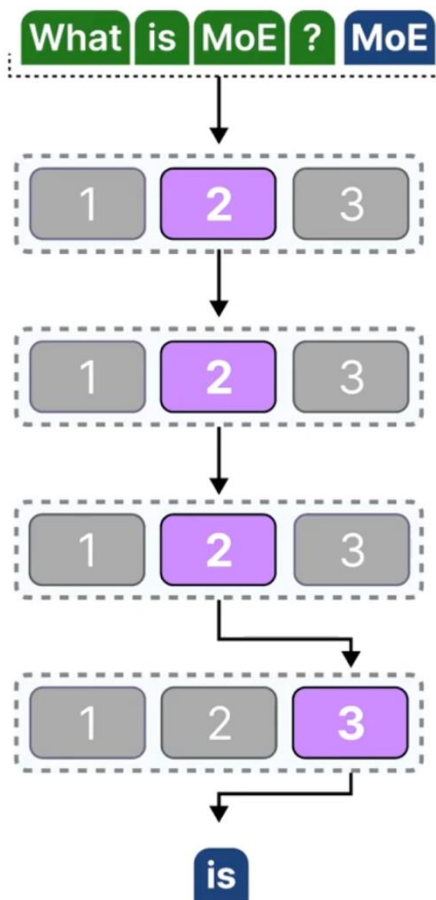


The Architecture of Experts

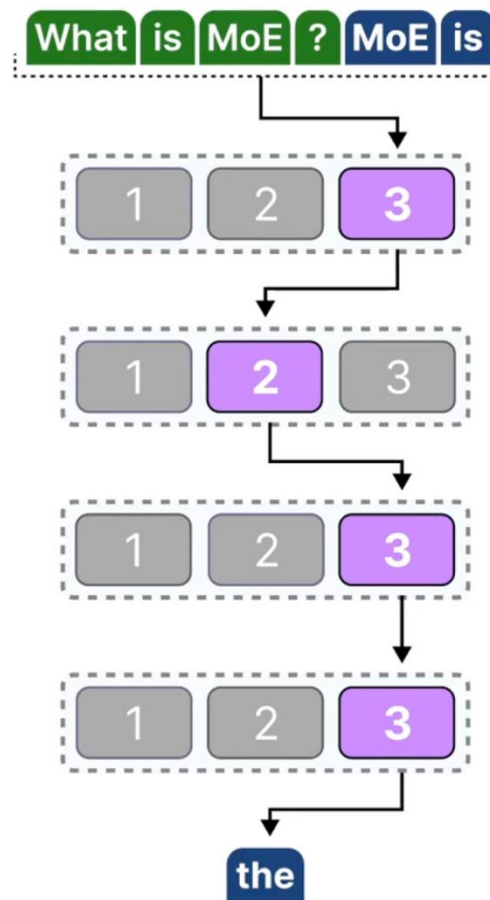
This means that each time you run inference, a set of **experts** is chosen that are best suited for the input.



First pass



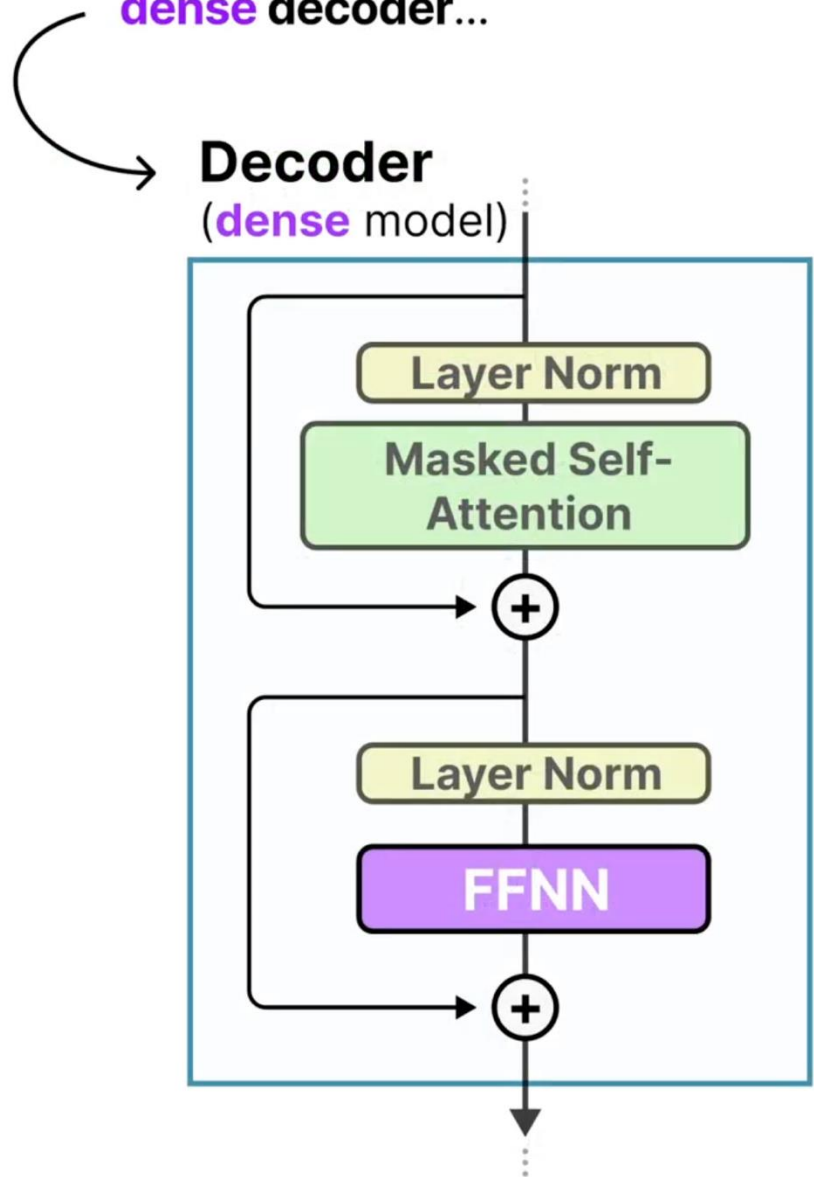
Second pass



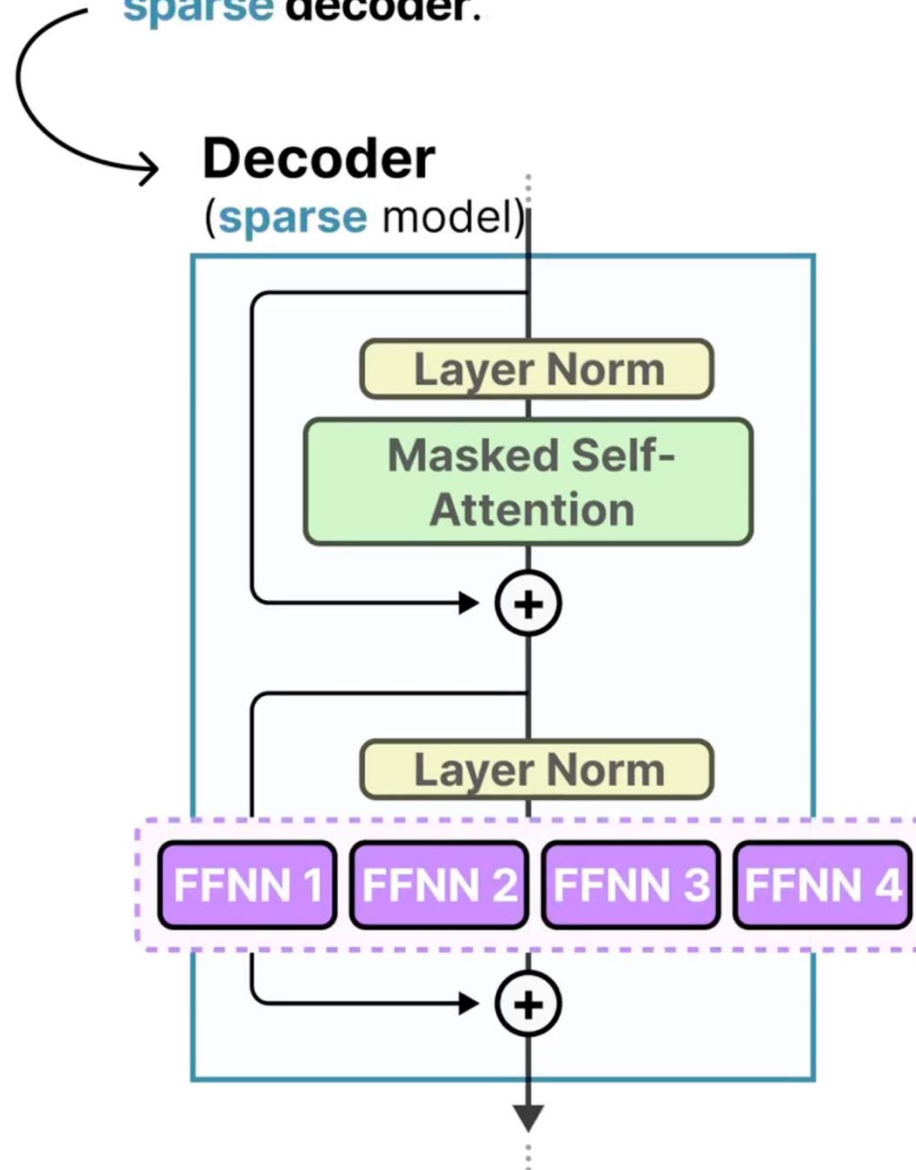
Third pass



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

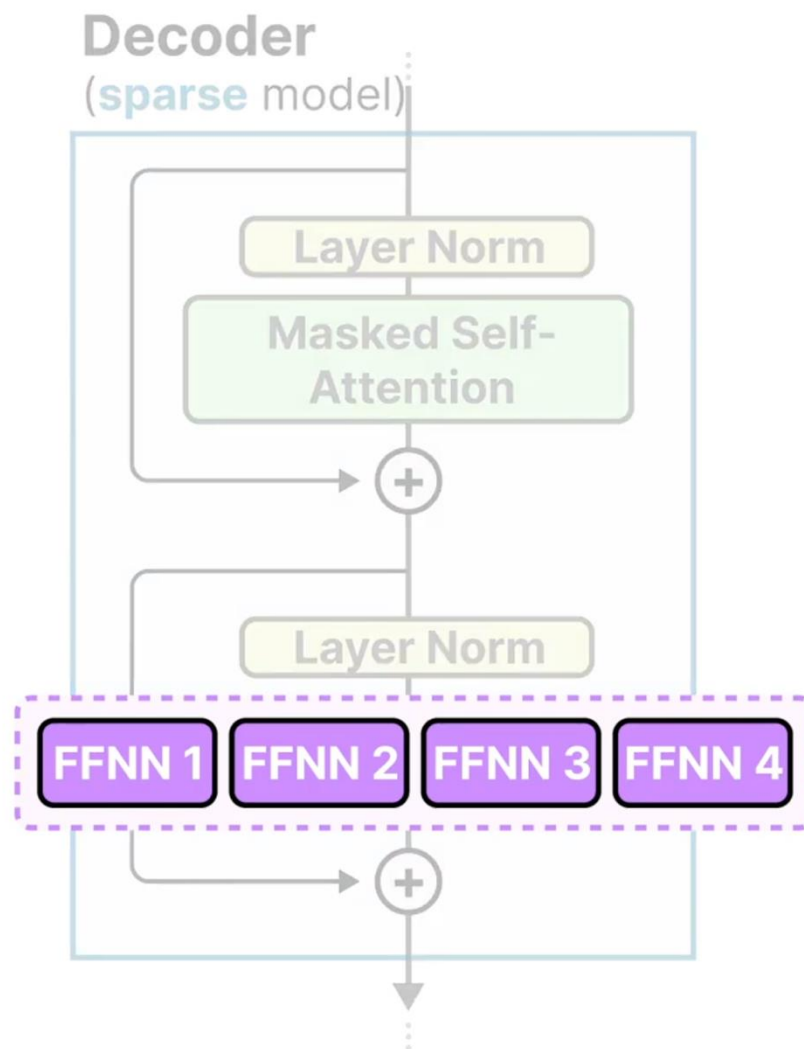


...with multiple **FFNNs** it would now be called a **sparse decoder**.



Introduction: MOE 基本原理

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



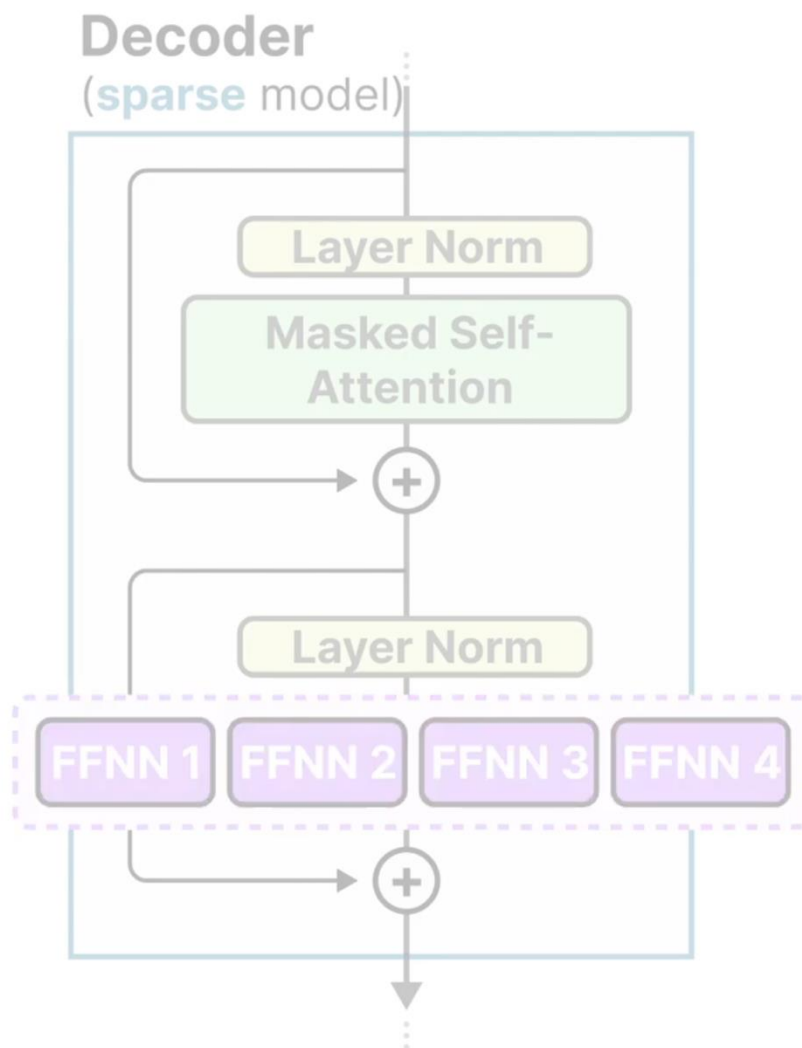
02

The Router: 路由原理



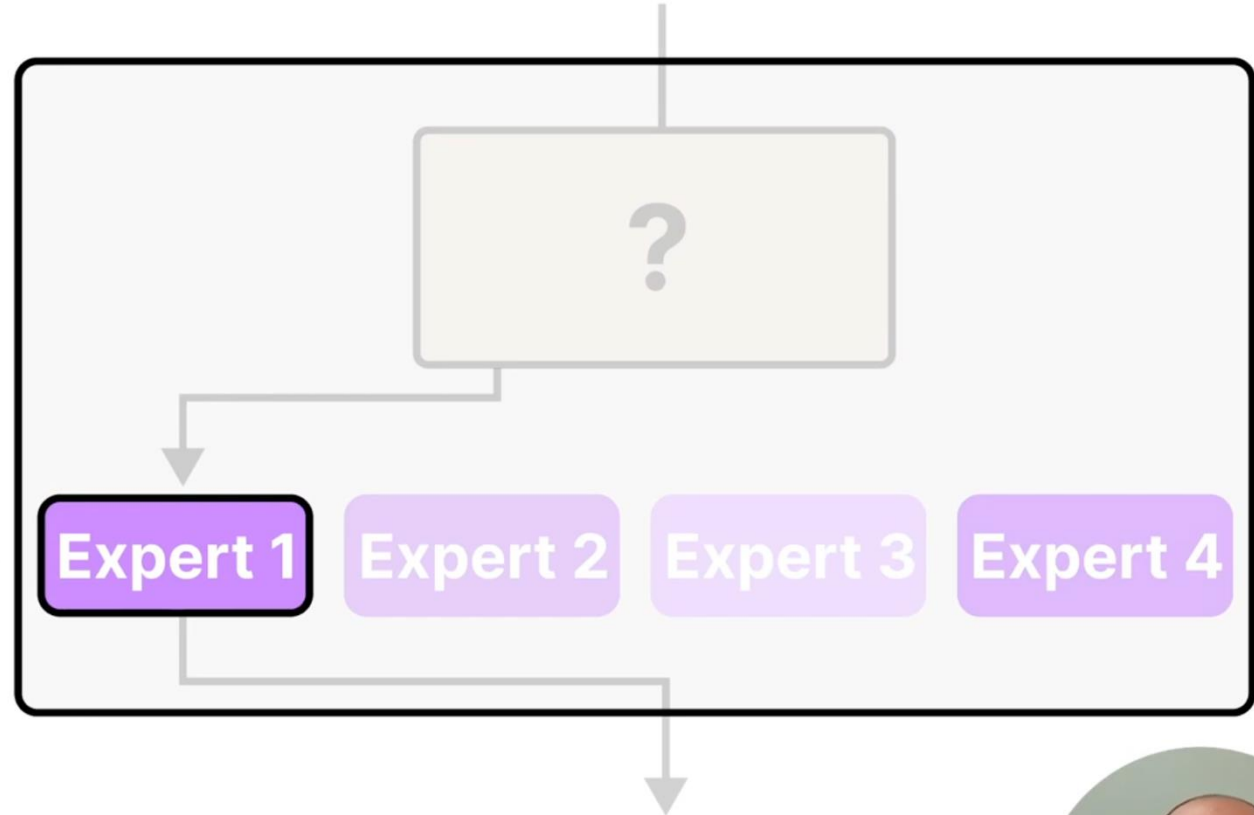
Introduction: MOE 基本原理

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



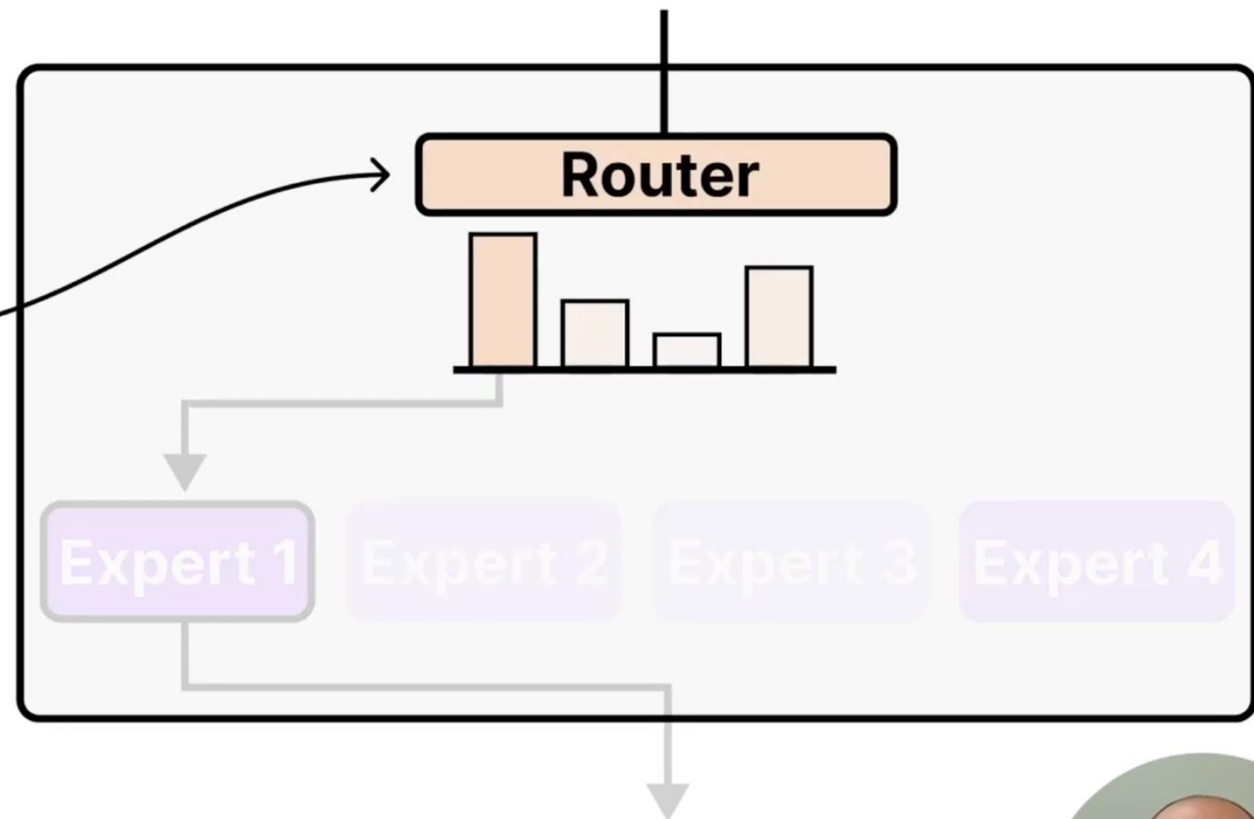
The Router

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



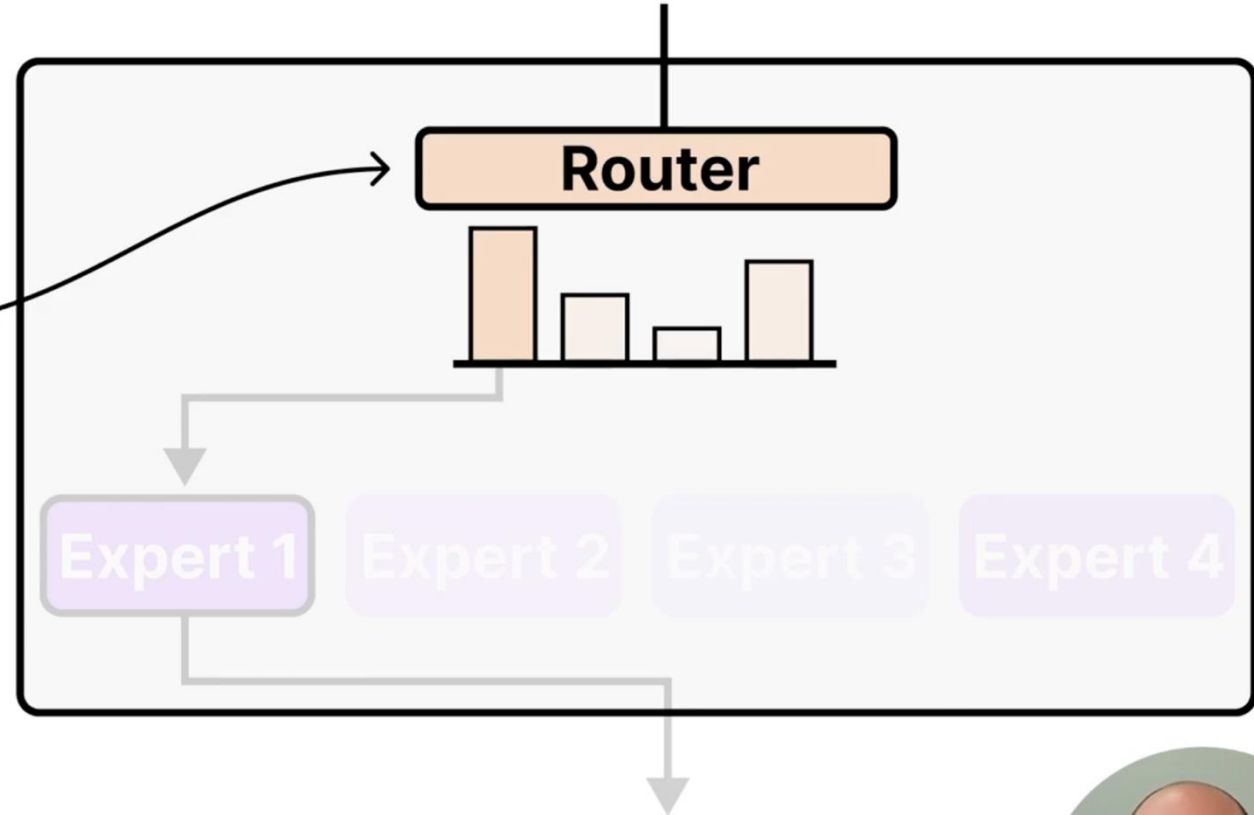
The Router

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



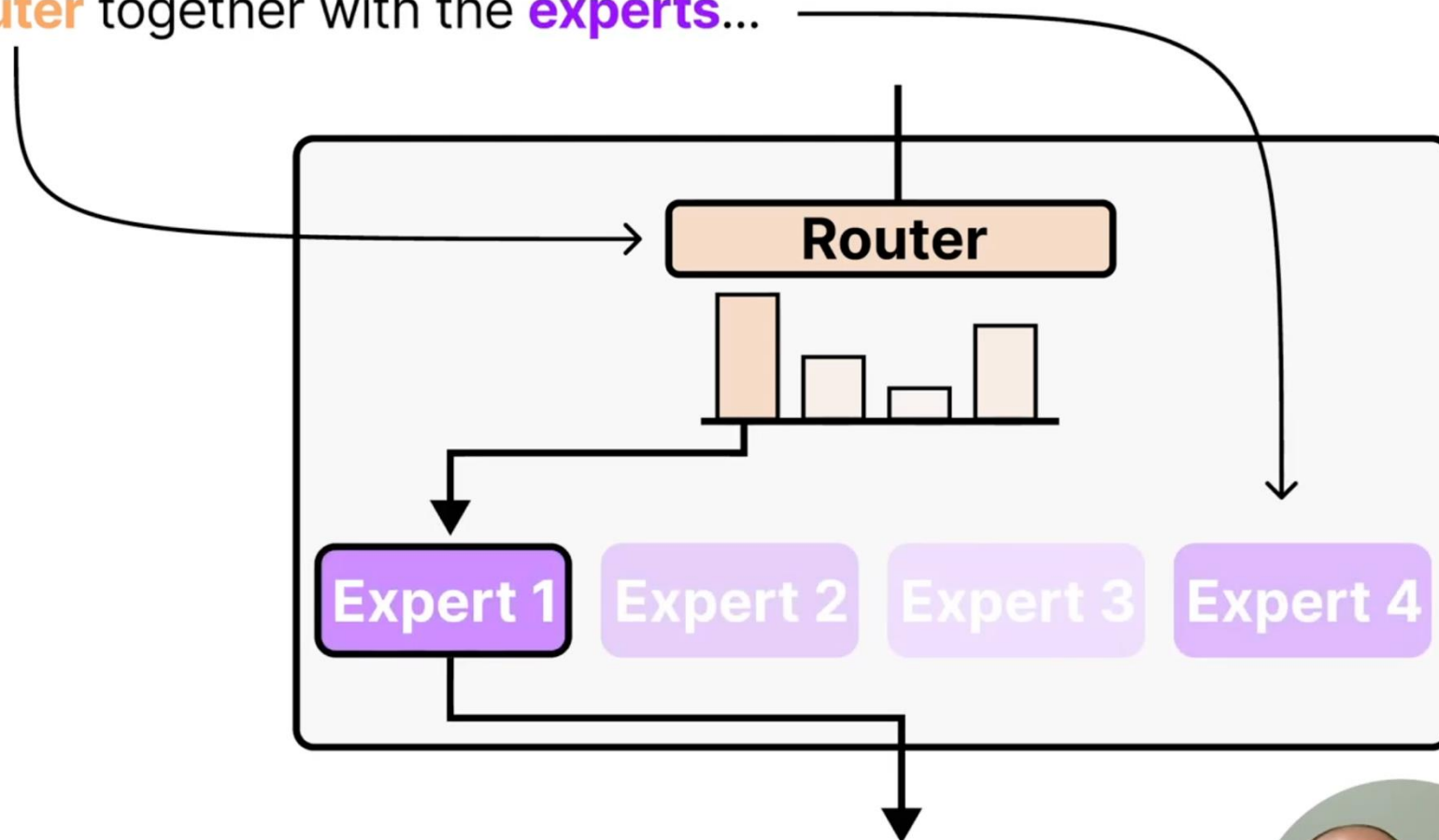
The Router

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



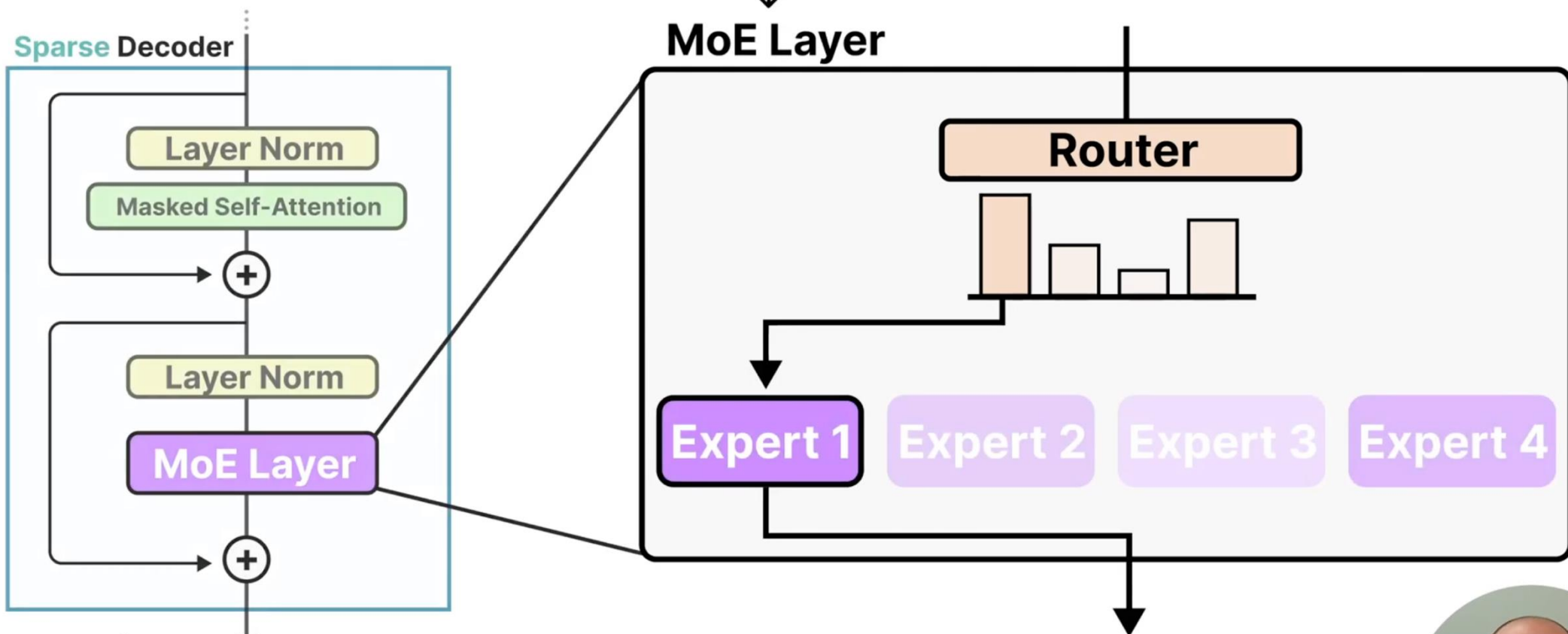
The Router

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



The Router

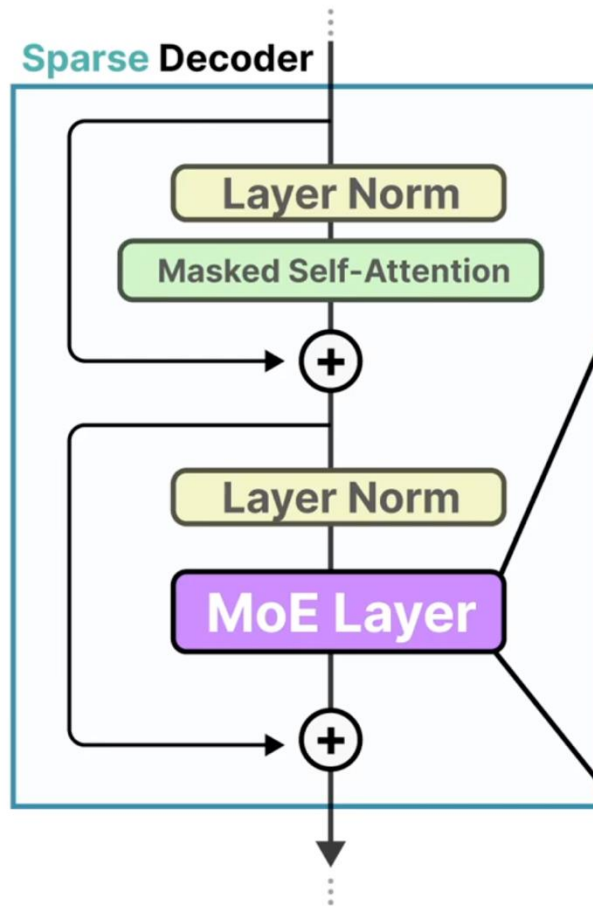
... make up the **MoE layer**...



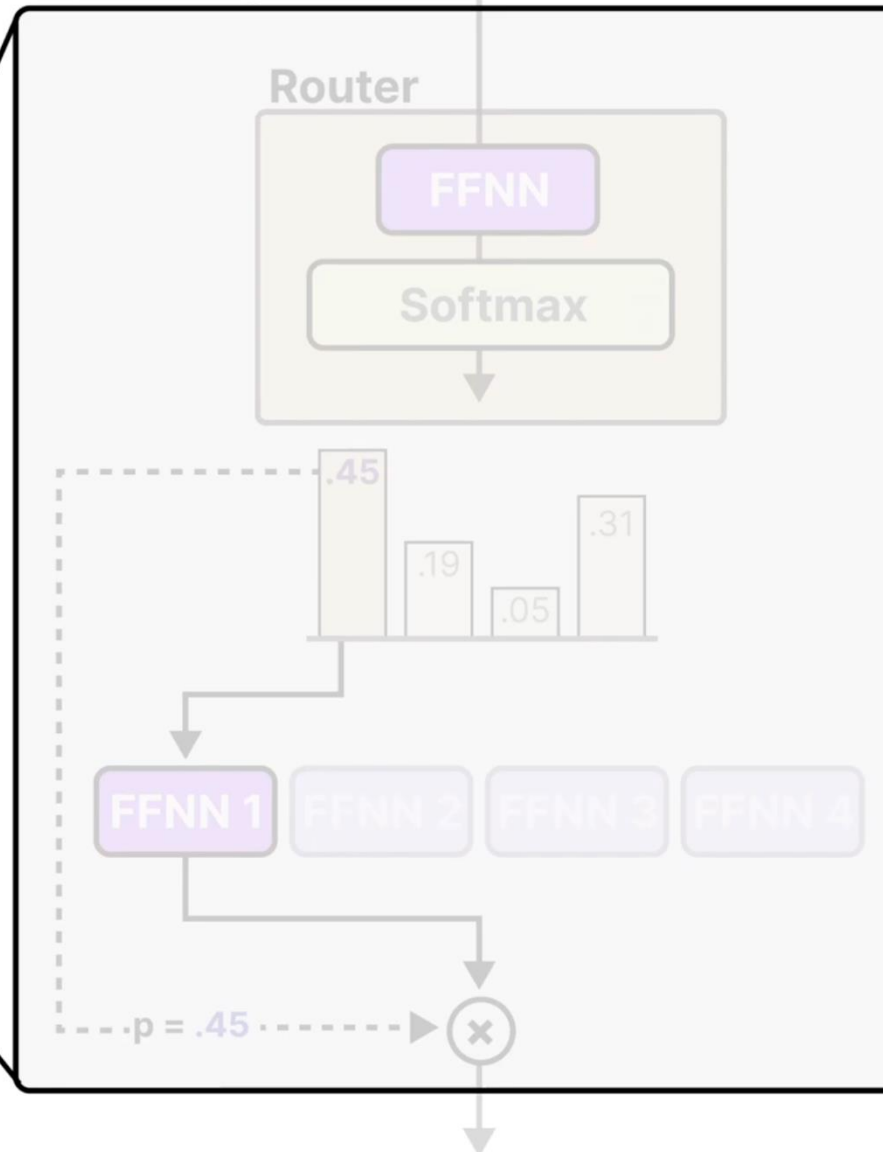
... in a **sparse decoder** and replace the single FFNN.



The Router



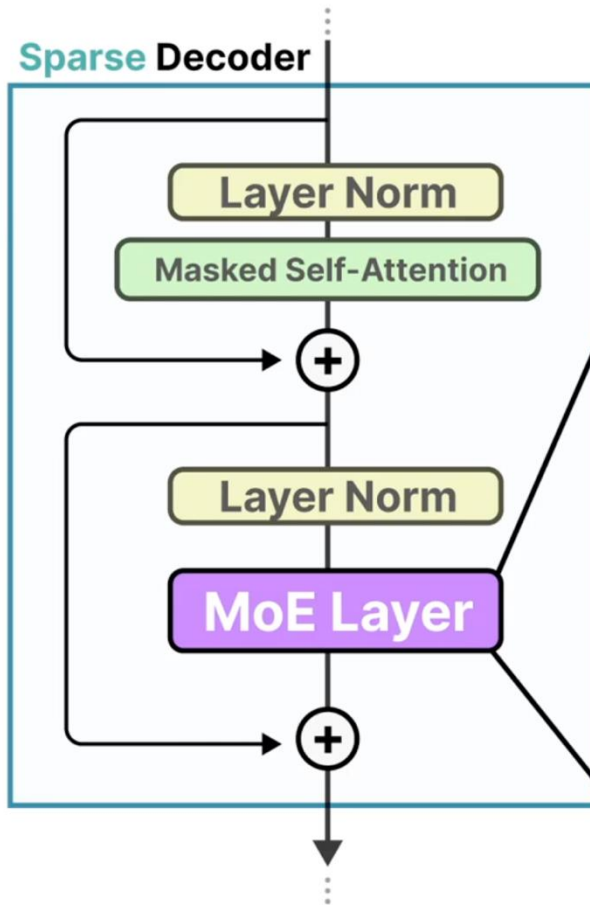
MoE Layer



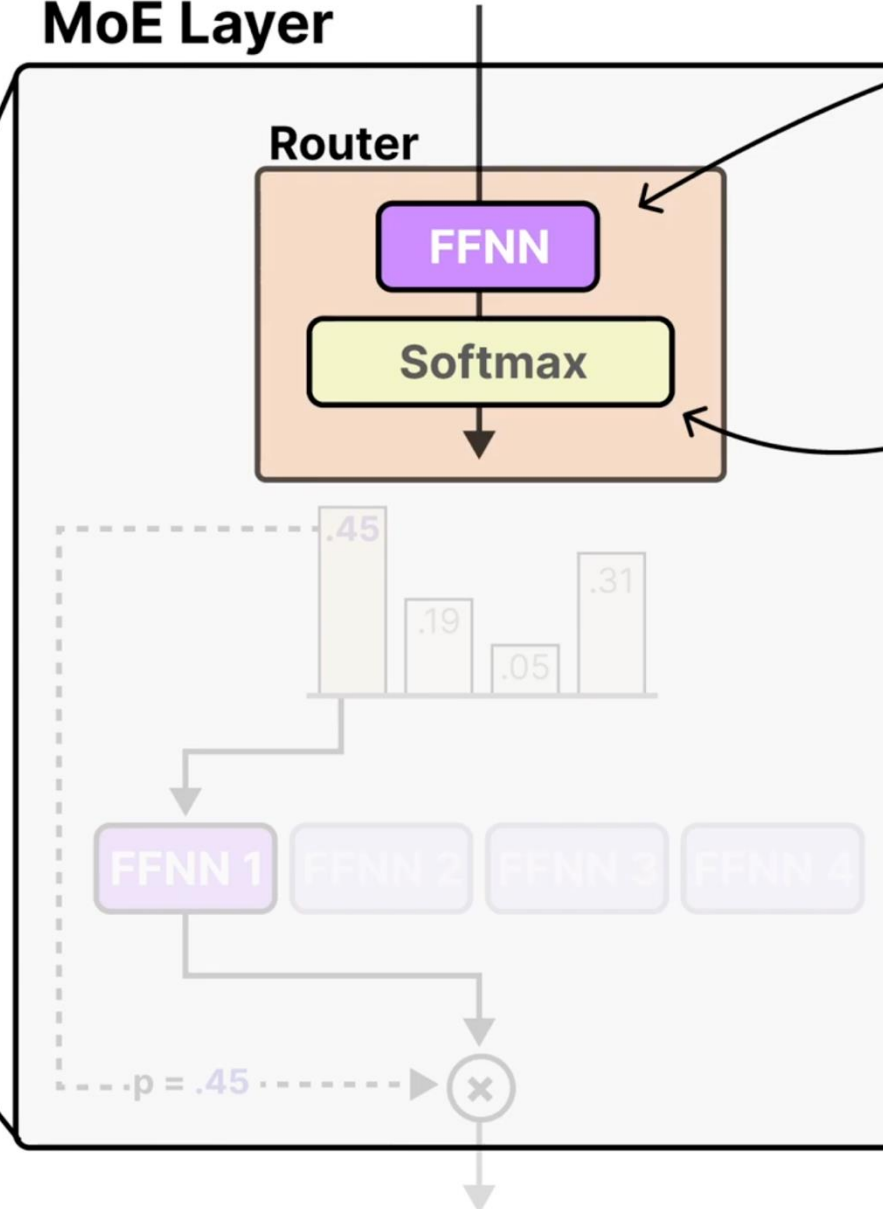
We can zoom in on the **MoE** layer and explore how the **router** works in detail.



The Router



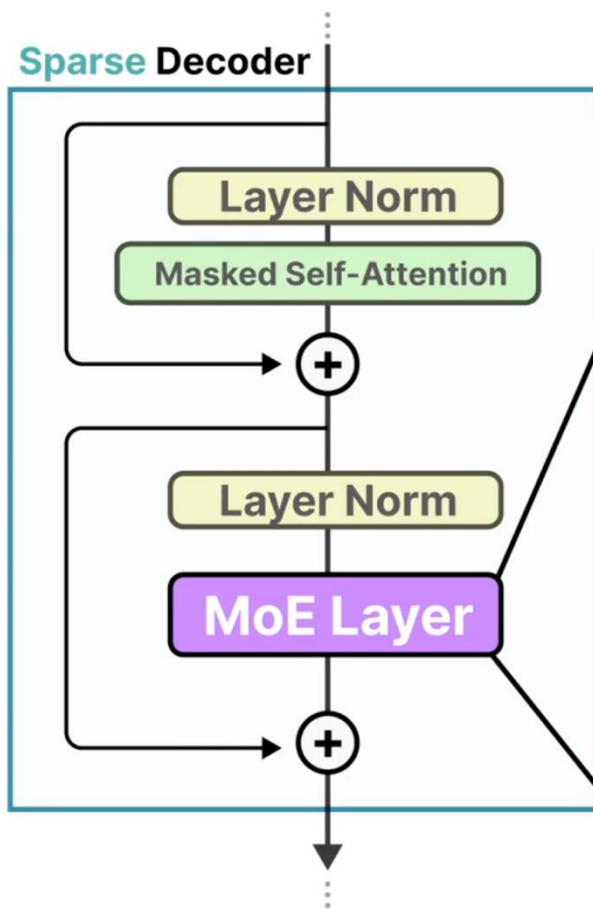
MoE Layer



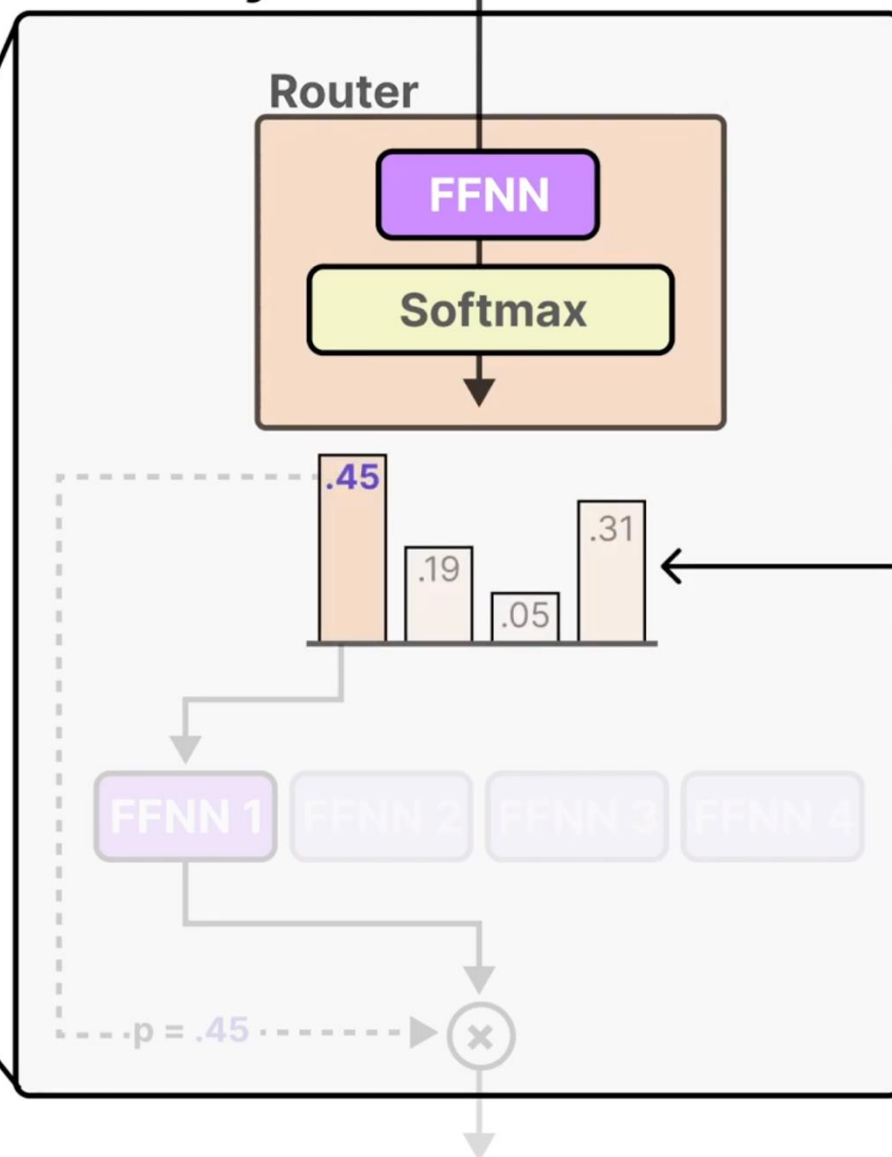
After the **FFNN** in the **router**, we see a **softmax function**...



The Router



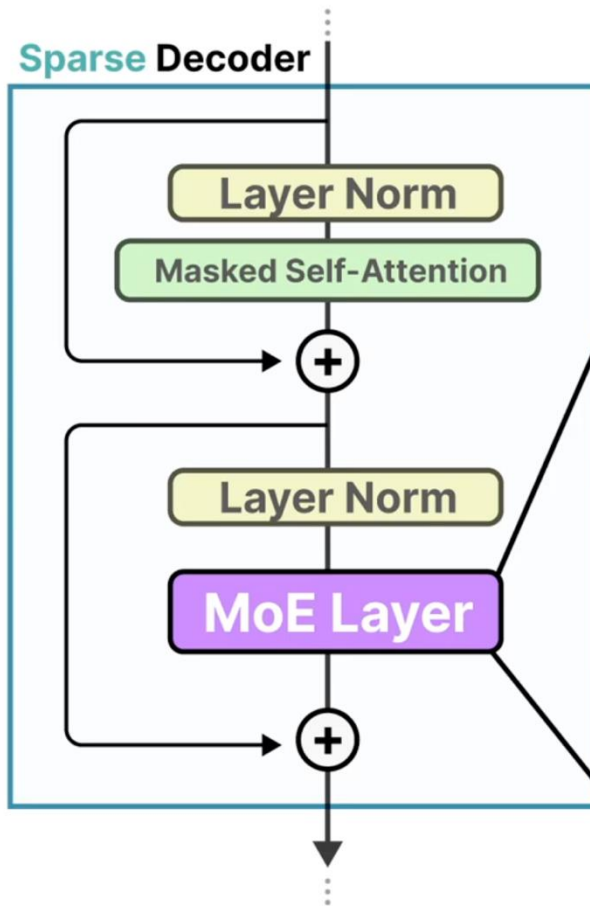
MoE Layer



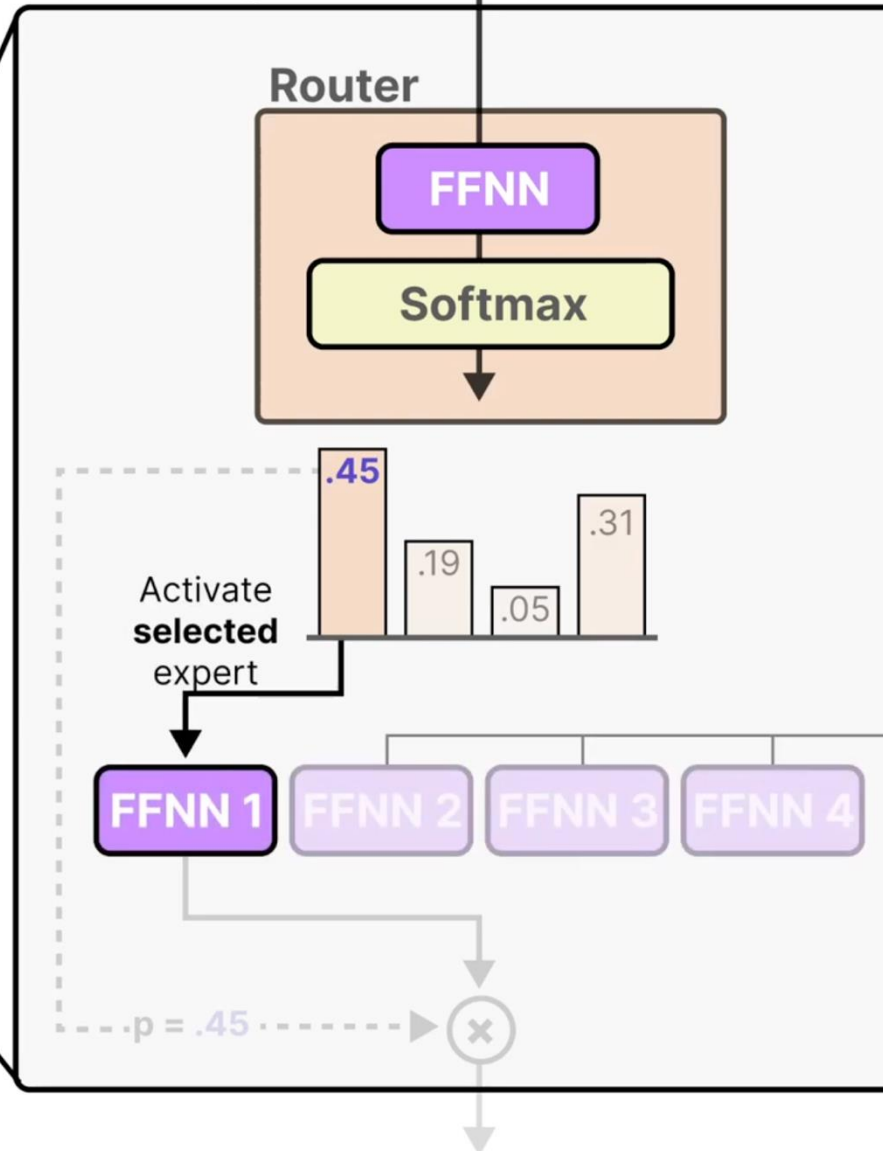
... creating **probabilities** for each expert...



The Router



MoE Layer

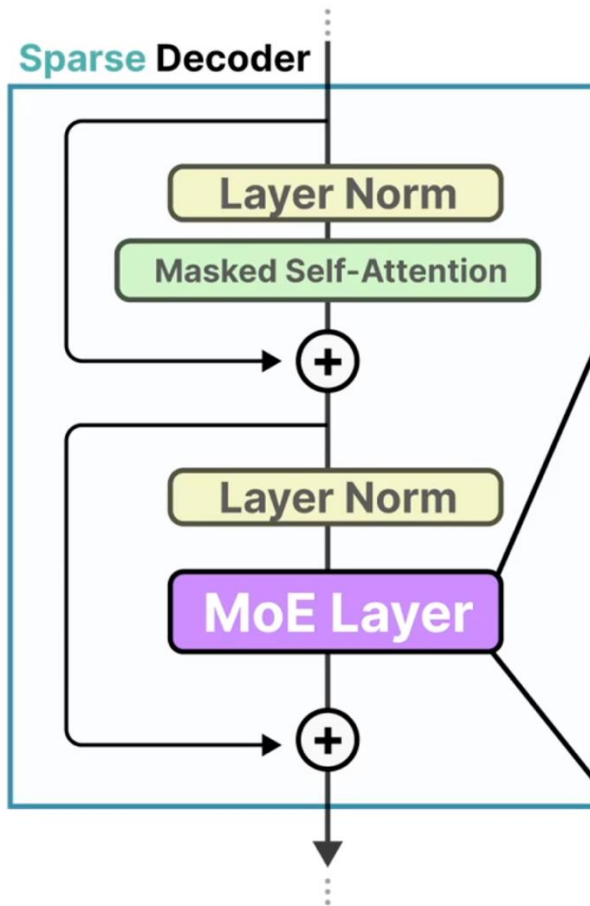


... that are used to **select** and **activate** the best **expert**.

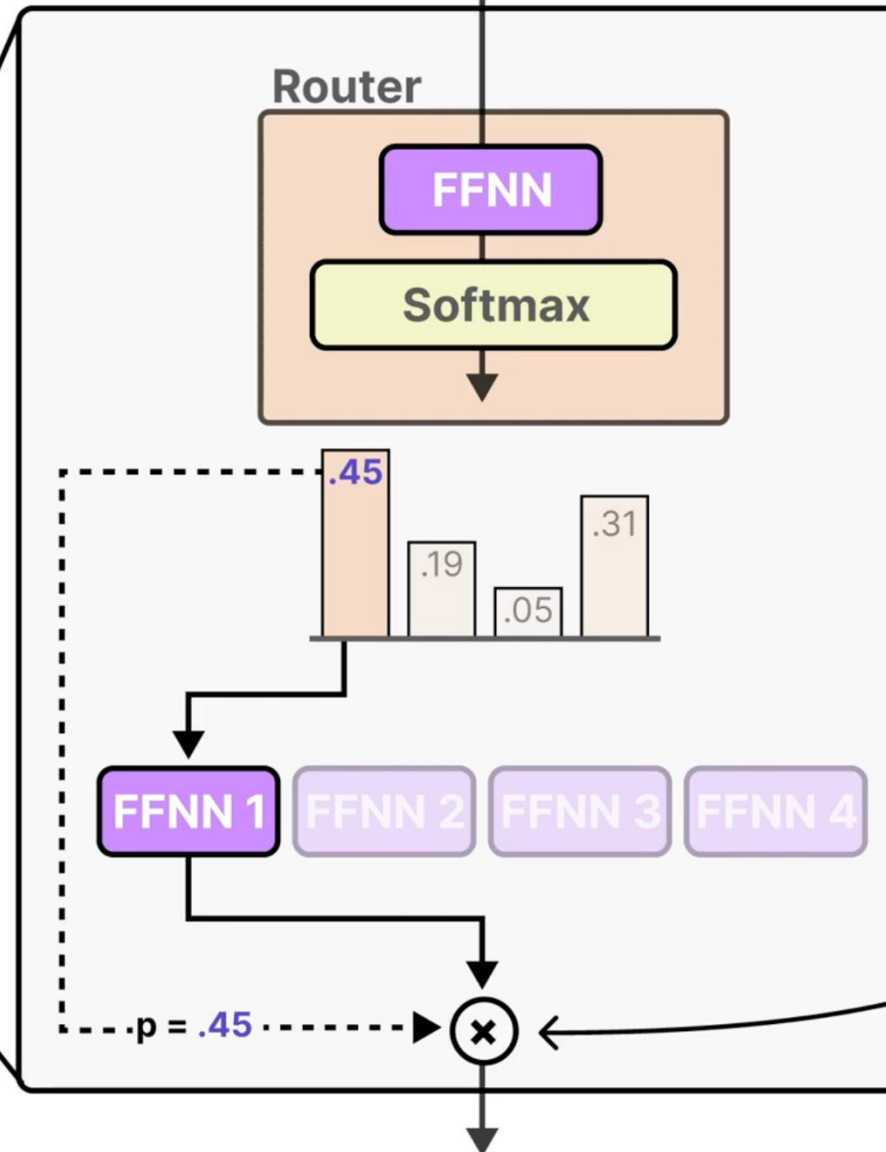
Not activated



The Router



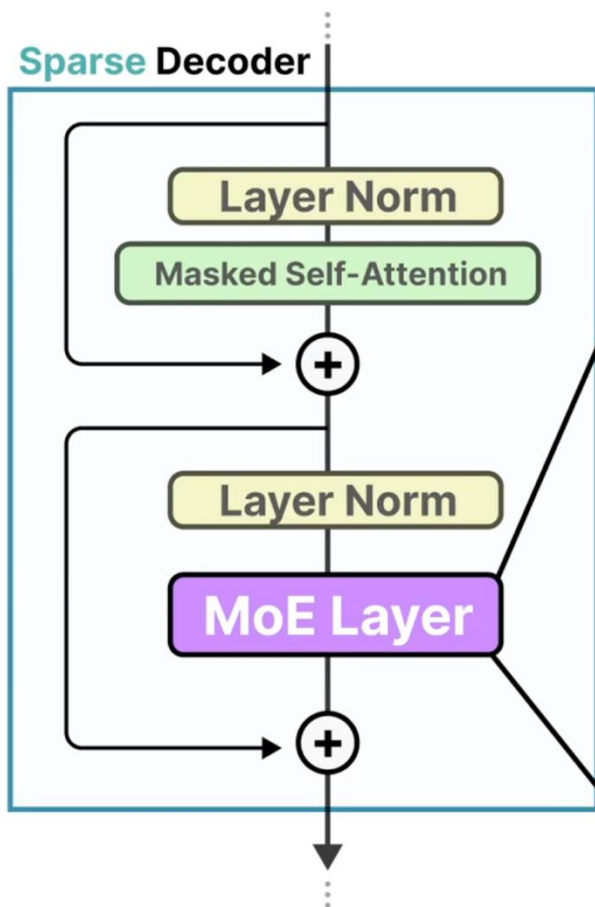
MoE Layer



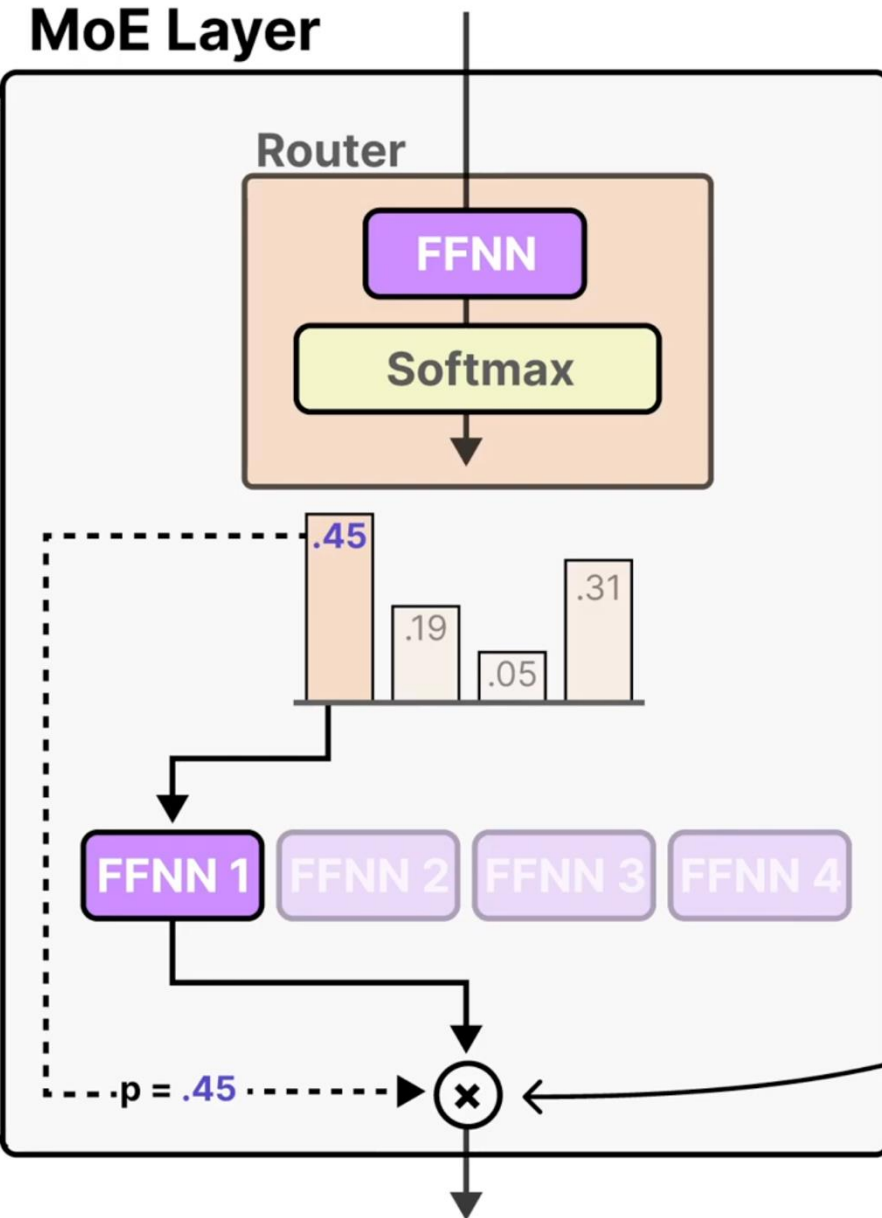
The final output is generated by multiplying the **router probability** with the **expert output**...



The Router



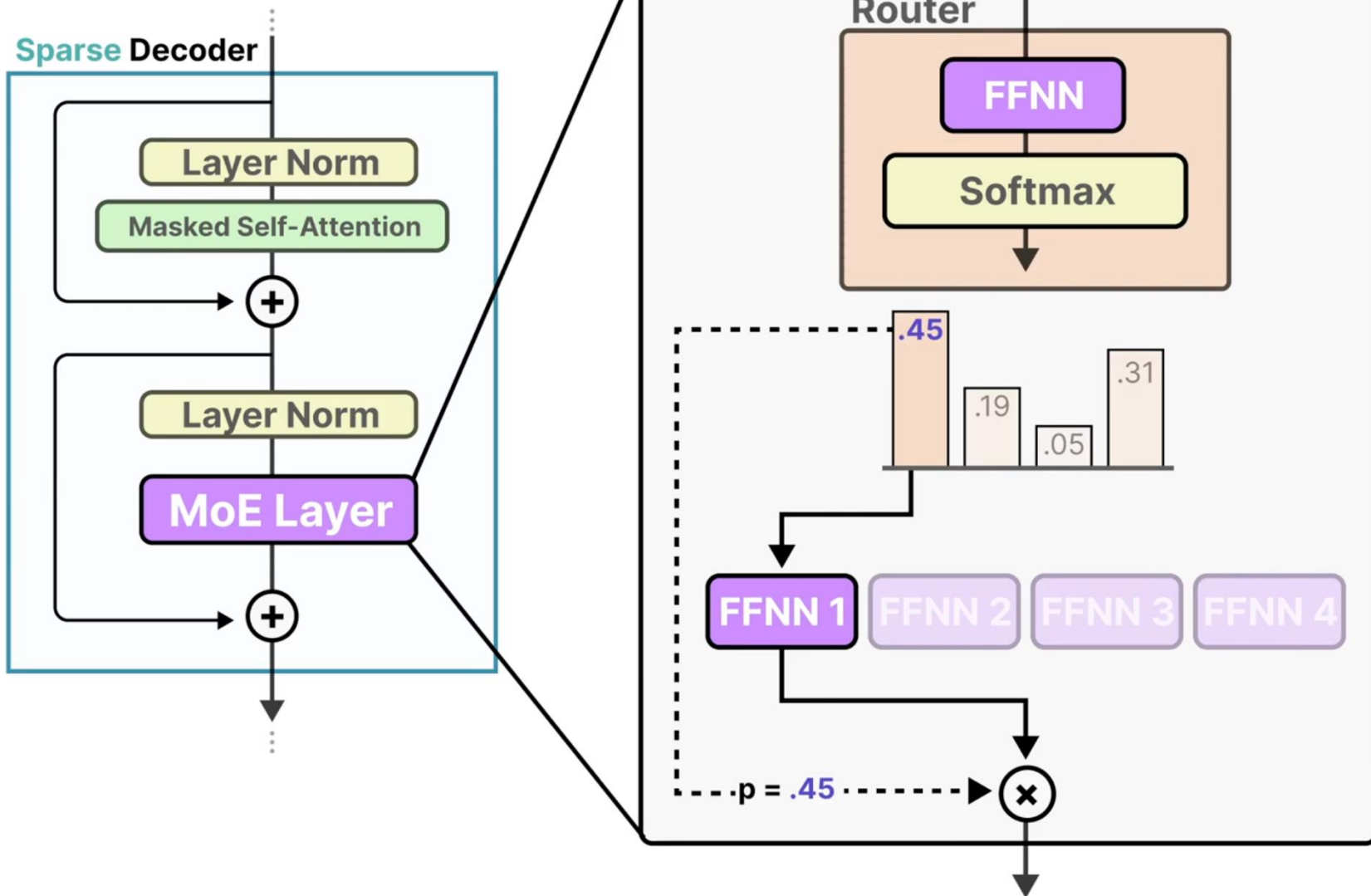
MoE Layer



...creating a **weighted** activation.



The Router



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

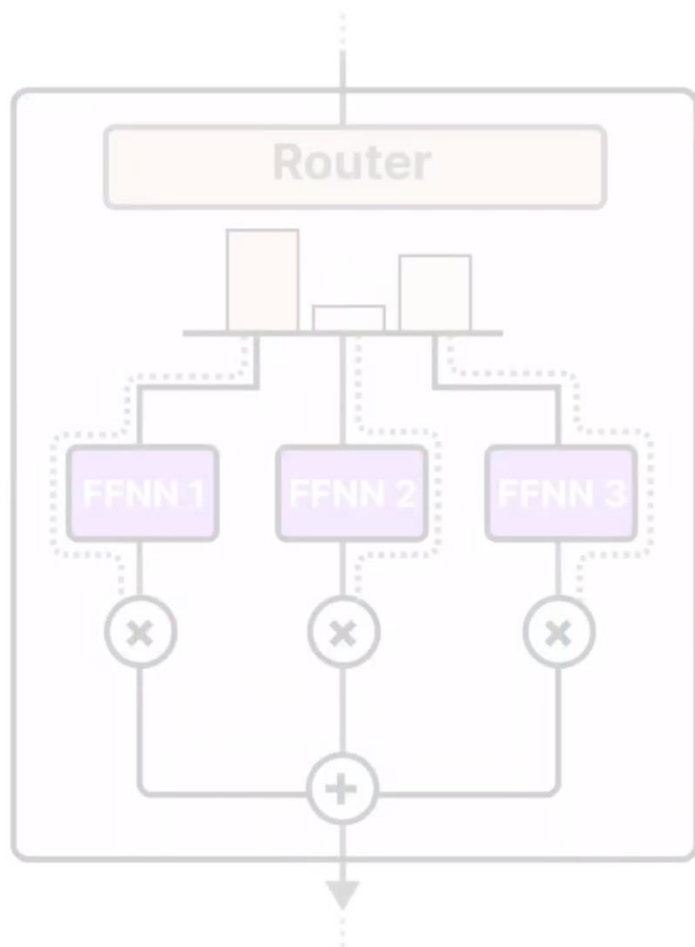


03

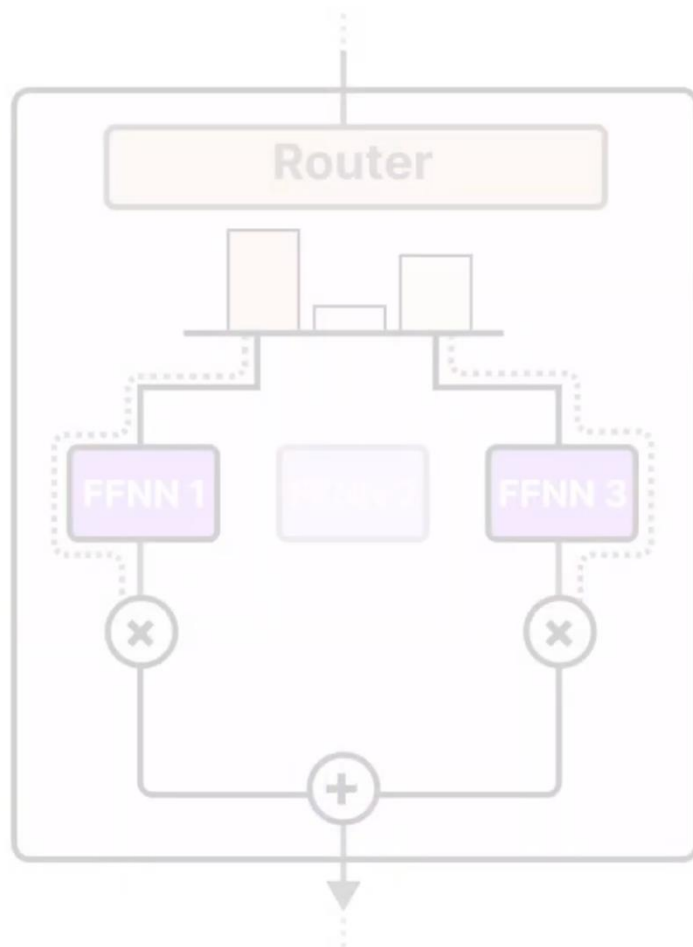
Architecture: 模型结构



Dense versus Sparse MoE



Dense MoE

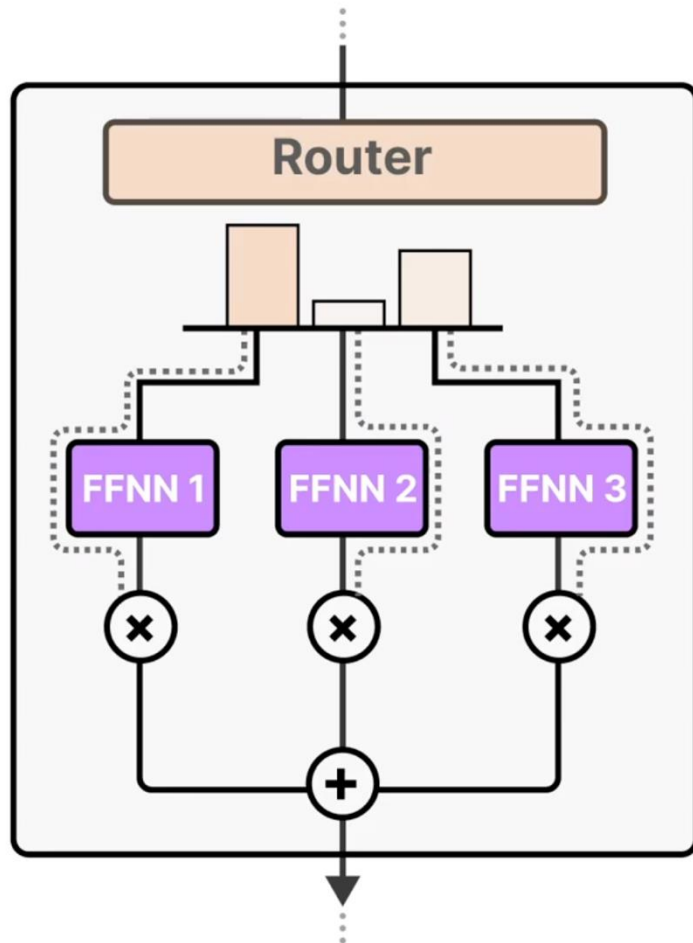


Sparse MoE

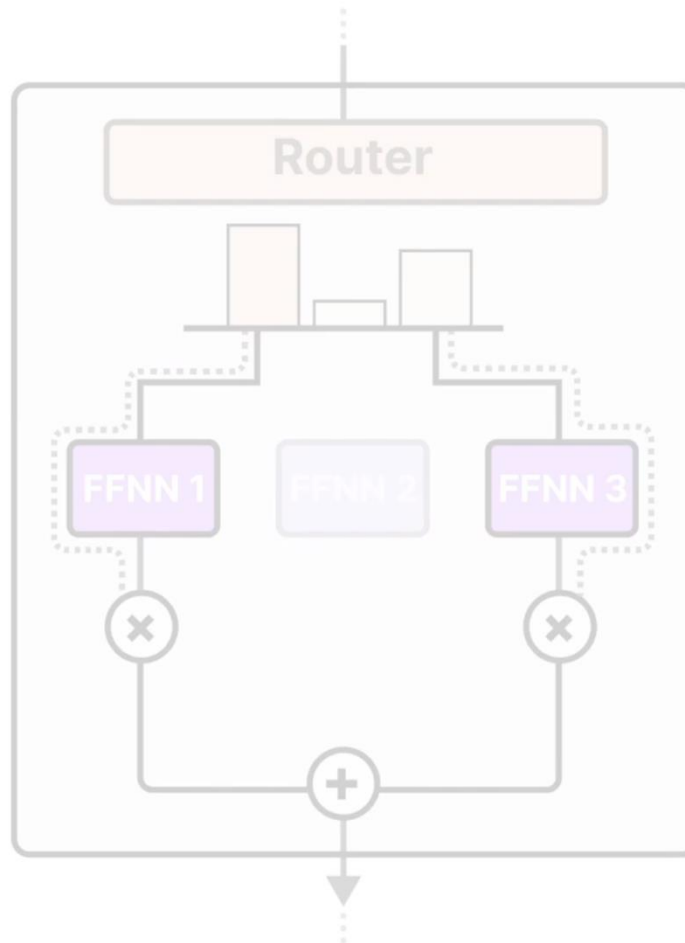
A given MoE layer comes in two sizes...



Dense versus Sparse MoE



Dense MoE

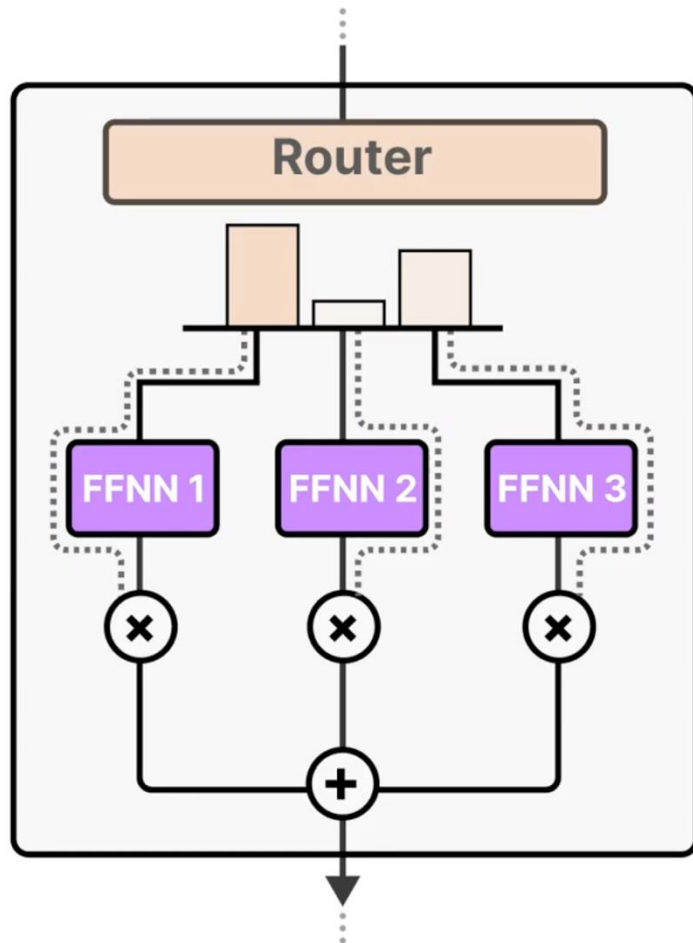


Sparse MoE

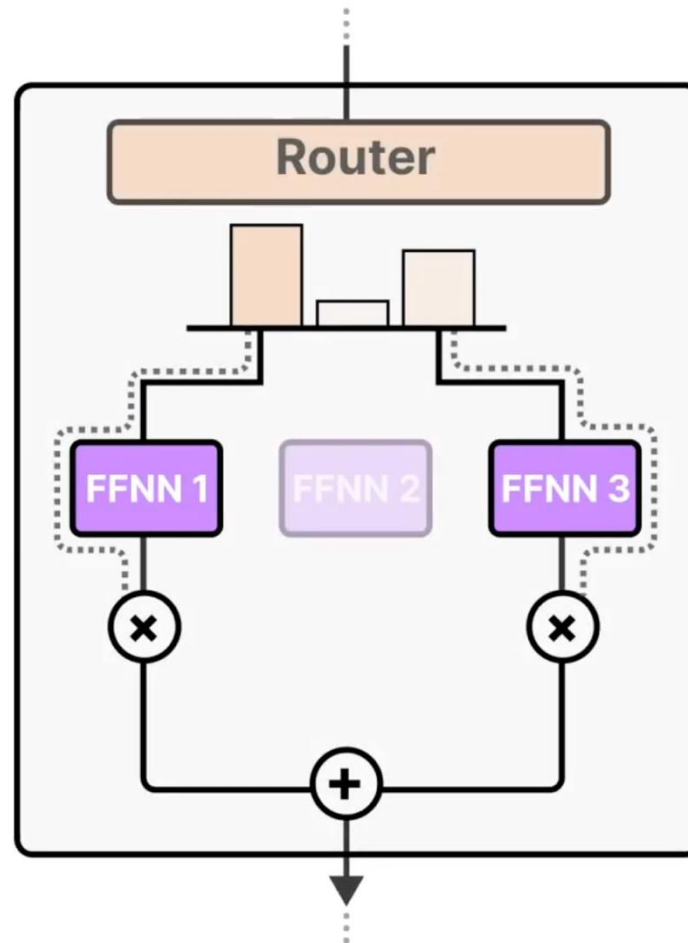
A given MoE layer comes in two sizes, either a **dense**...



Dense versus Sparse MoE



Dense MoE

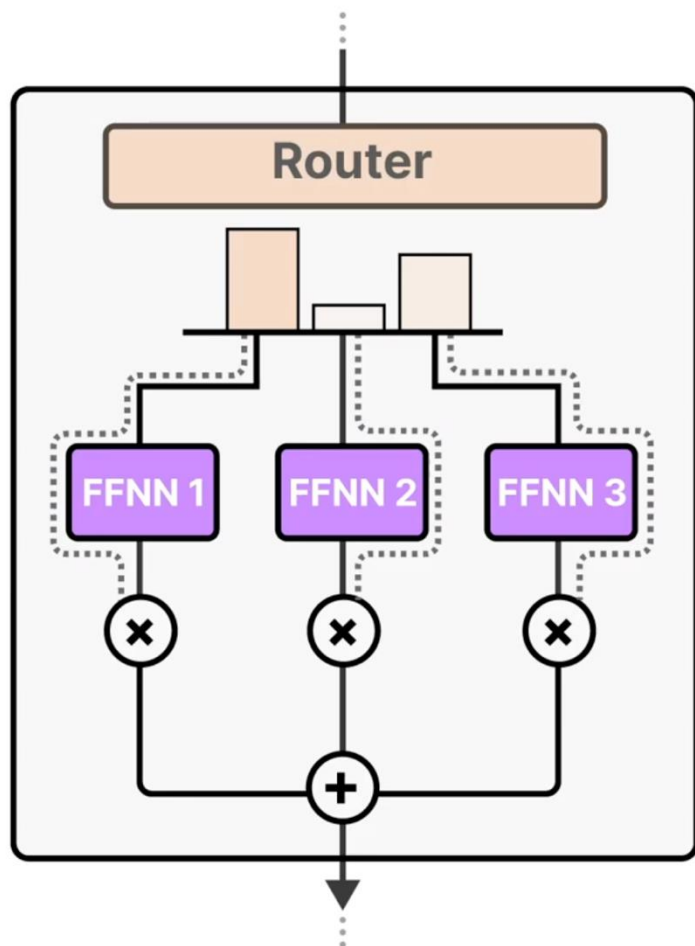


Sparse MoE

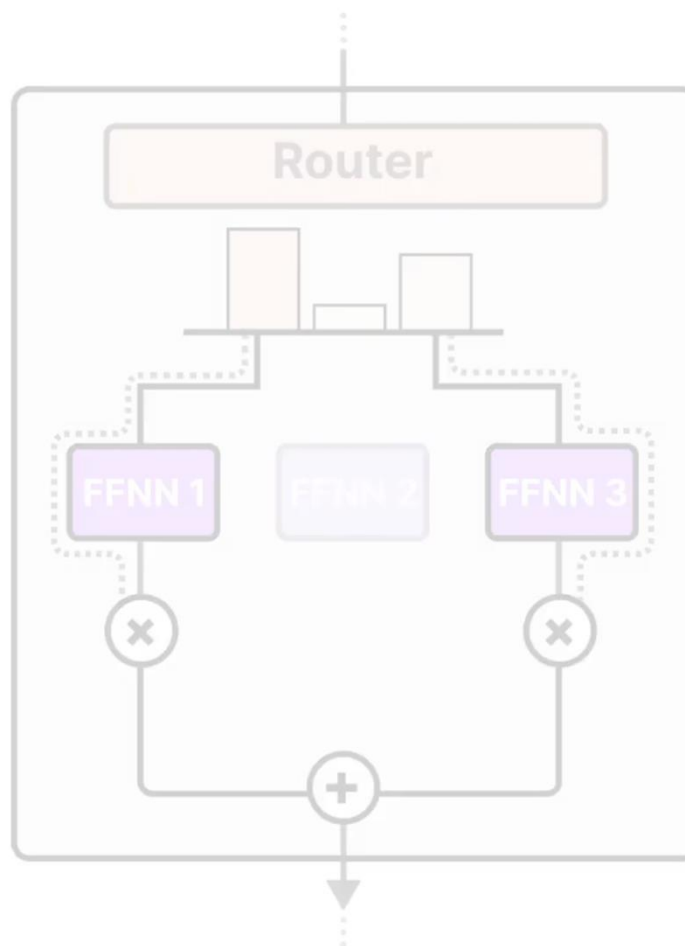
A given MoE layer comes in two sizes, either a **dense** or a **sparse** Mixture of Experts.



Dense versus Sparse MoE



Dense MoE

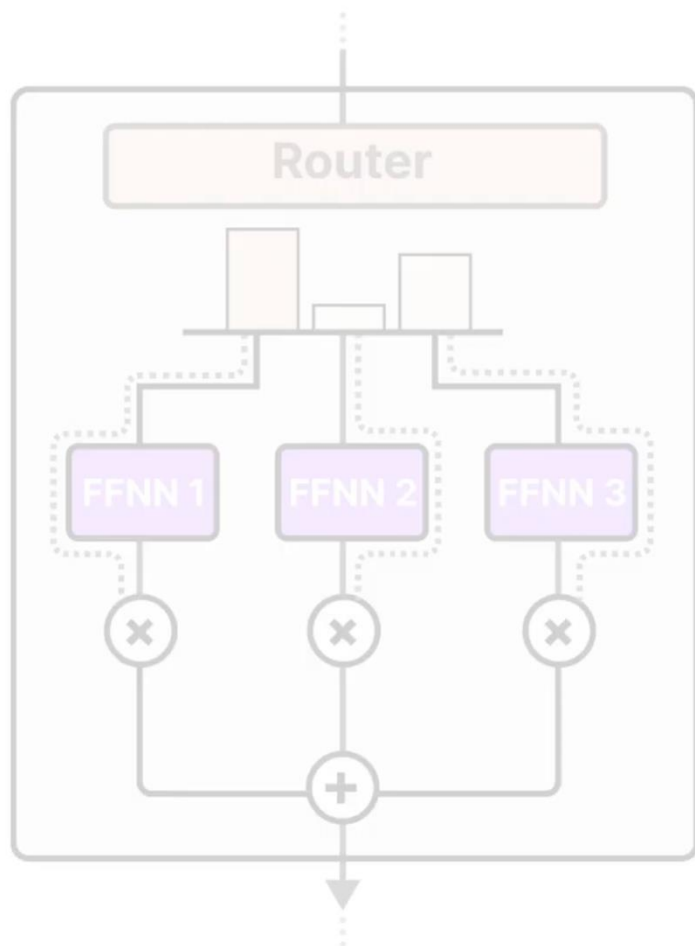


Sparse MoE

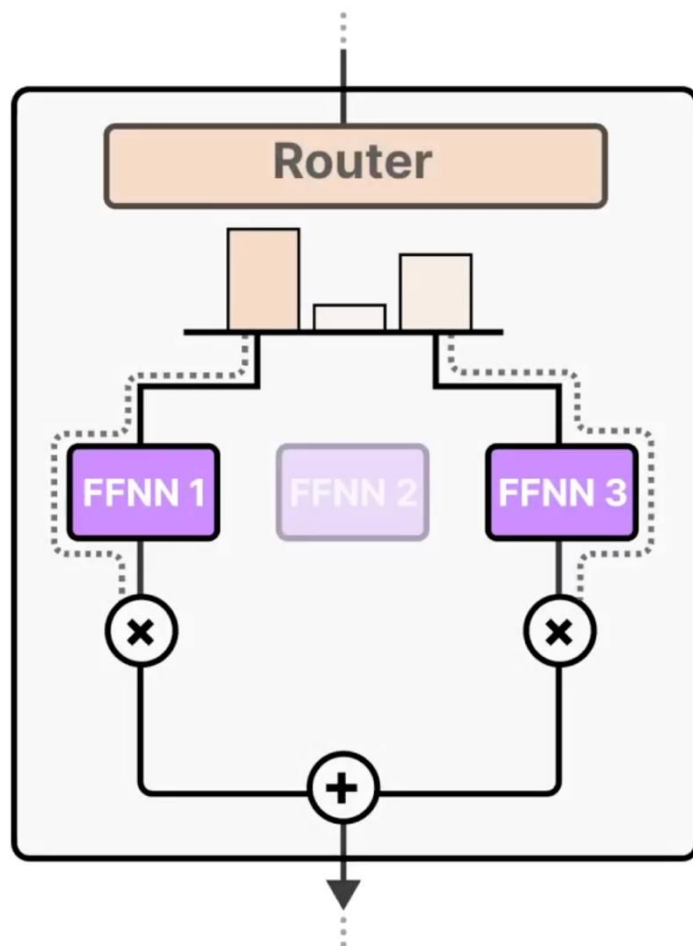
A **Dense MoE** will distribute the tokens across all experts...



Dense versus Sparse MoE

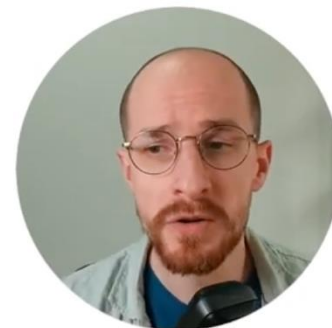


Dense MoE

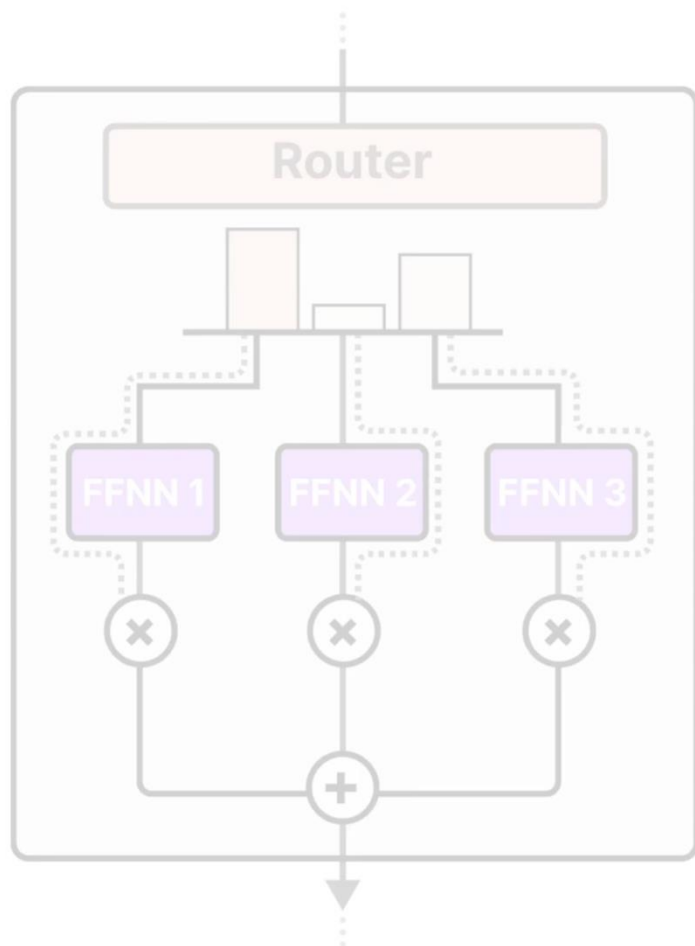


Sparse MoE

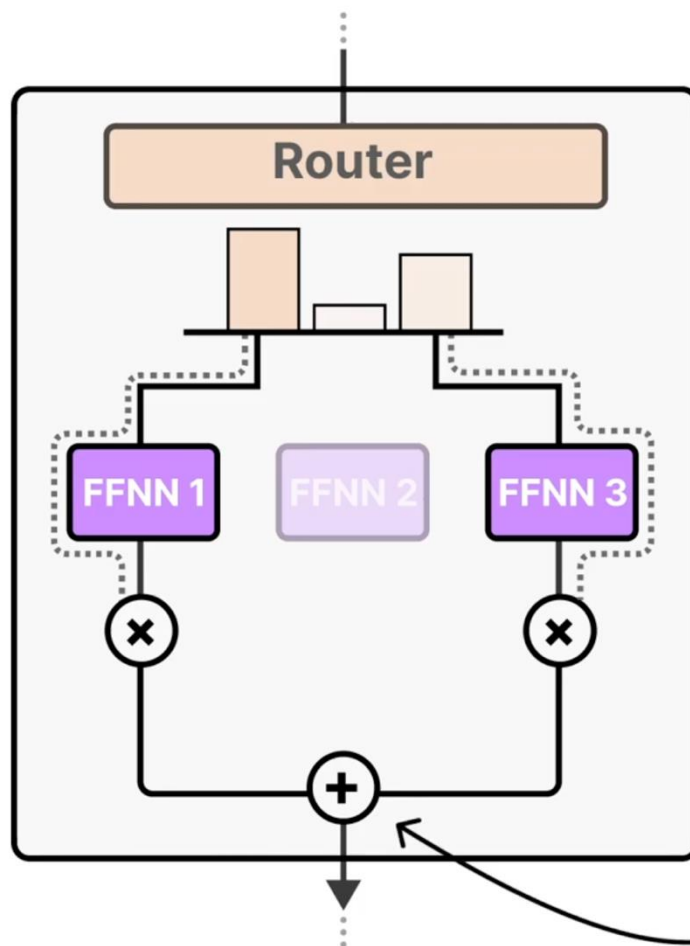
... whereas a **Sparse MoE** will only select a few experts.



Dense versus Sparse MoE



Dense MoE

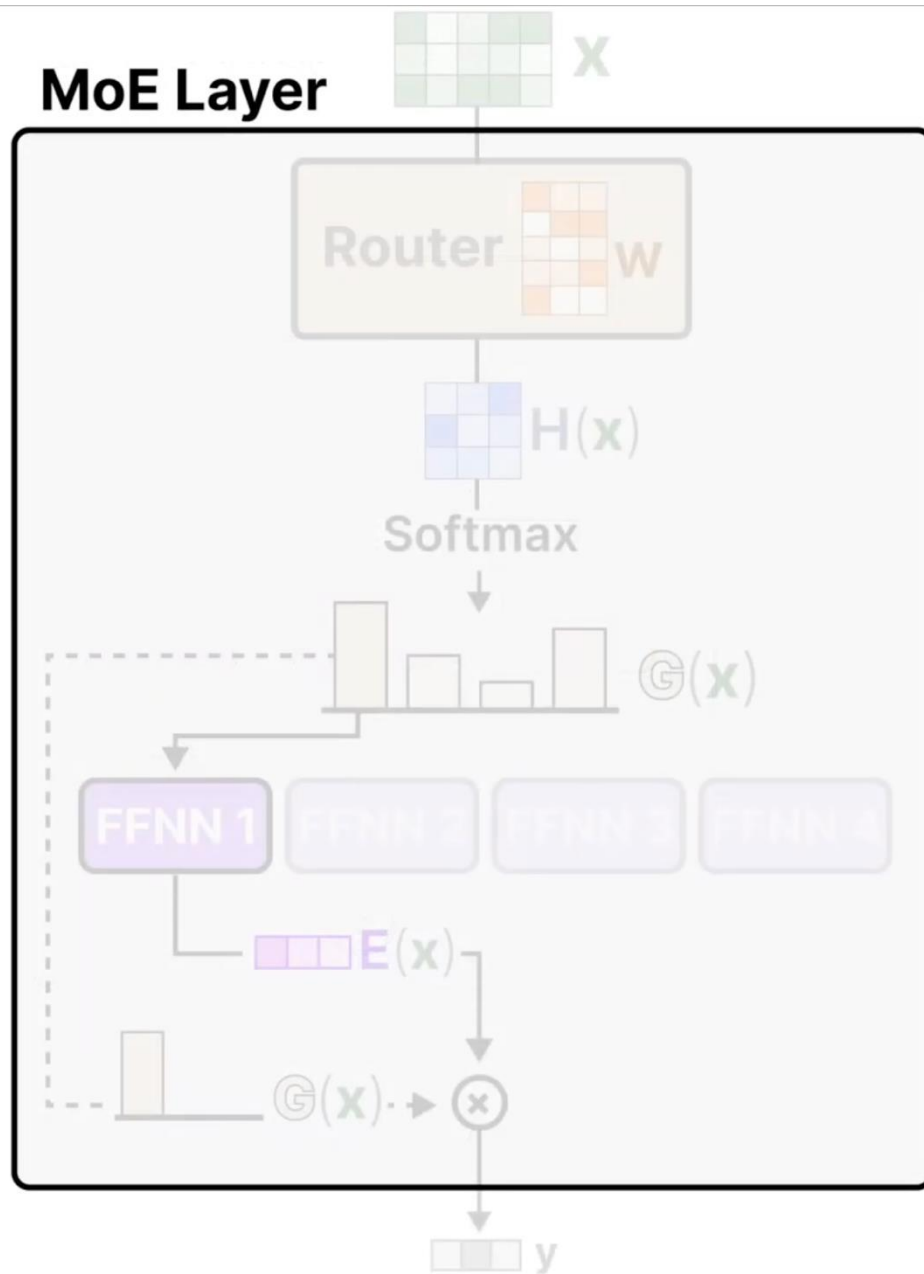


Sparse MoE

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



MoE Layer

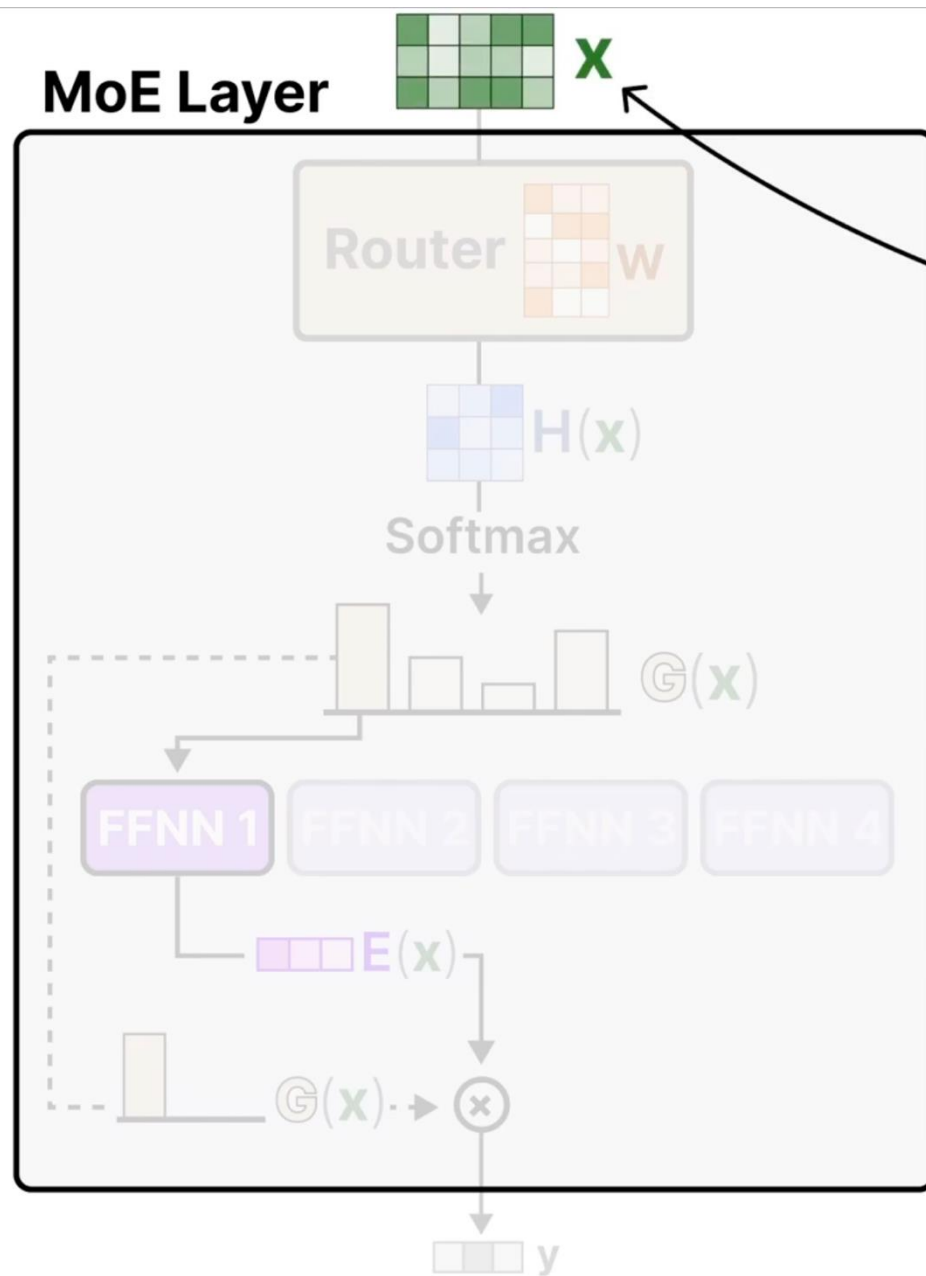


Selection of Experts

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

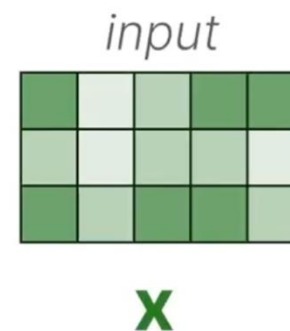


MoE Layer

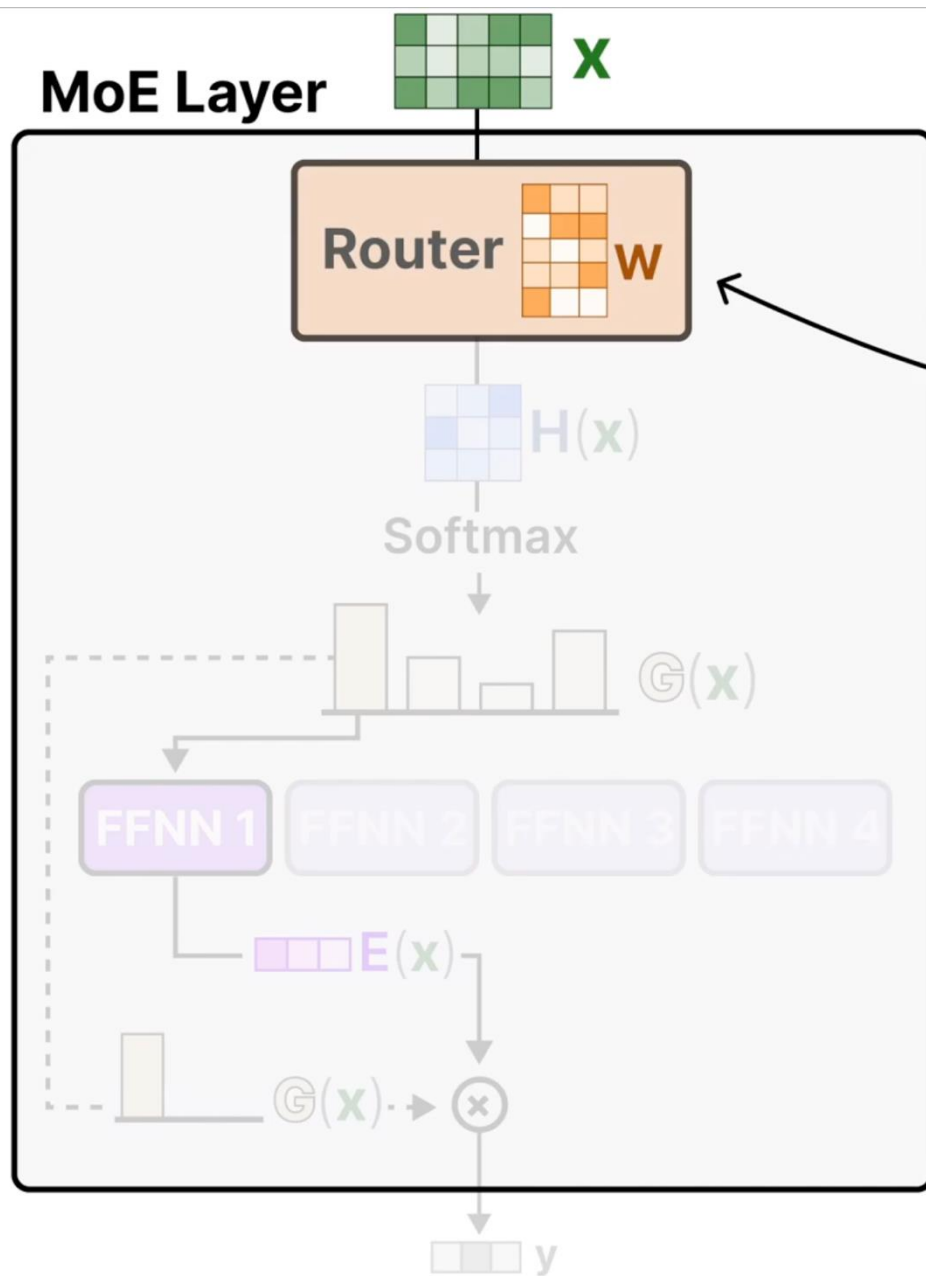


In its most basic form, we multiply the **input** (x) ...

Selection of Experts

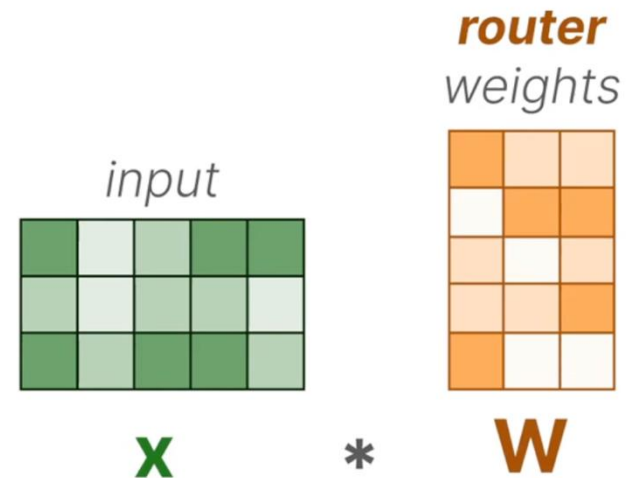


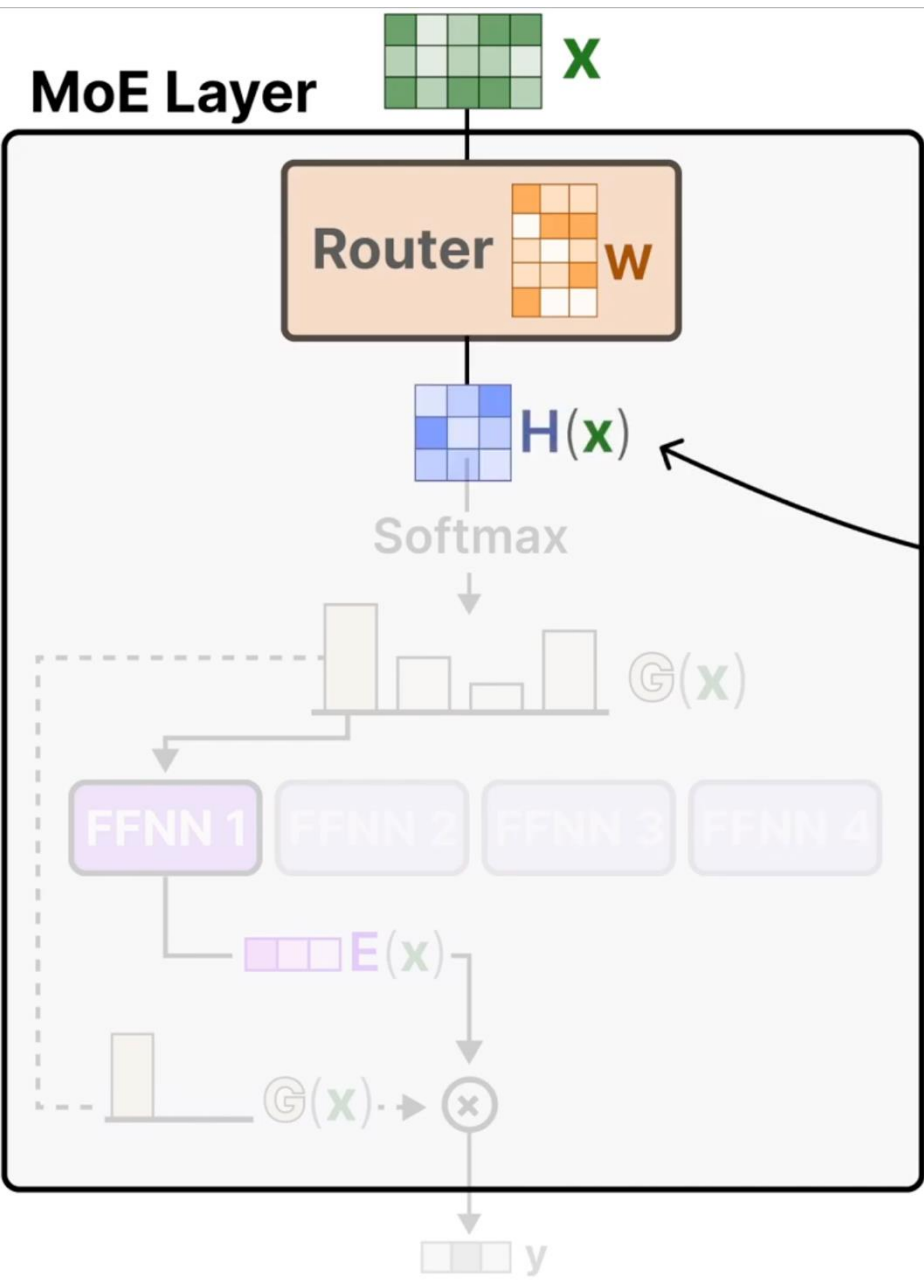
MoE Layer



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

Selection of Experts





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

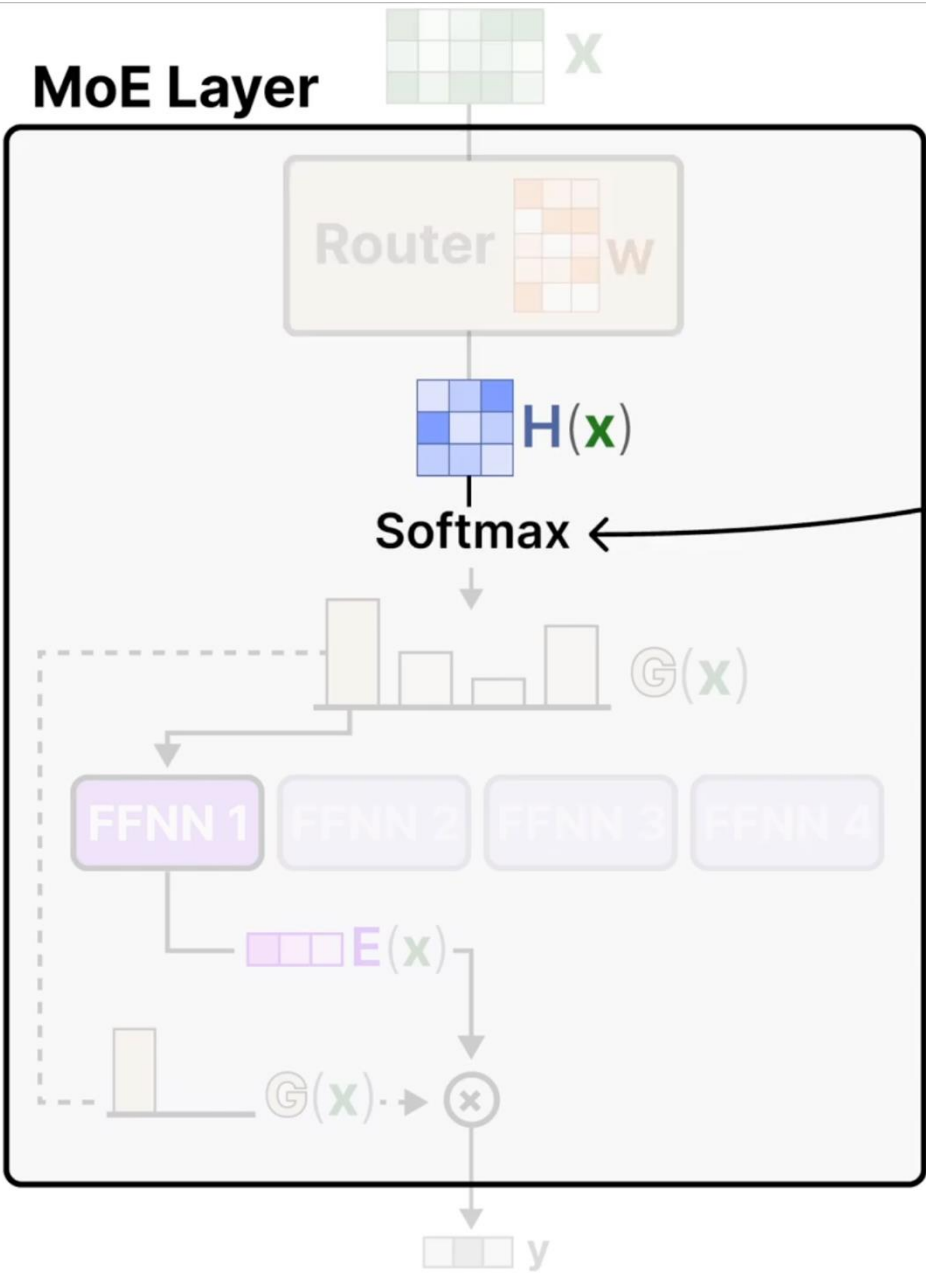
output

input

$H(x) = x * W$

router weights



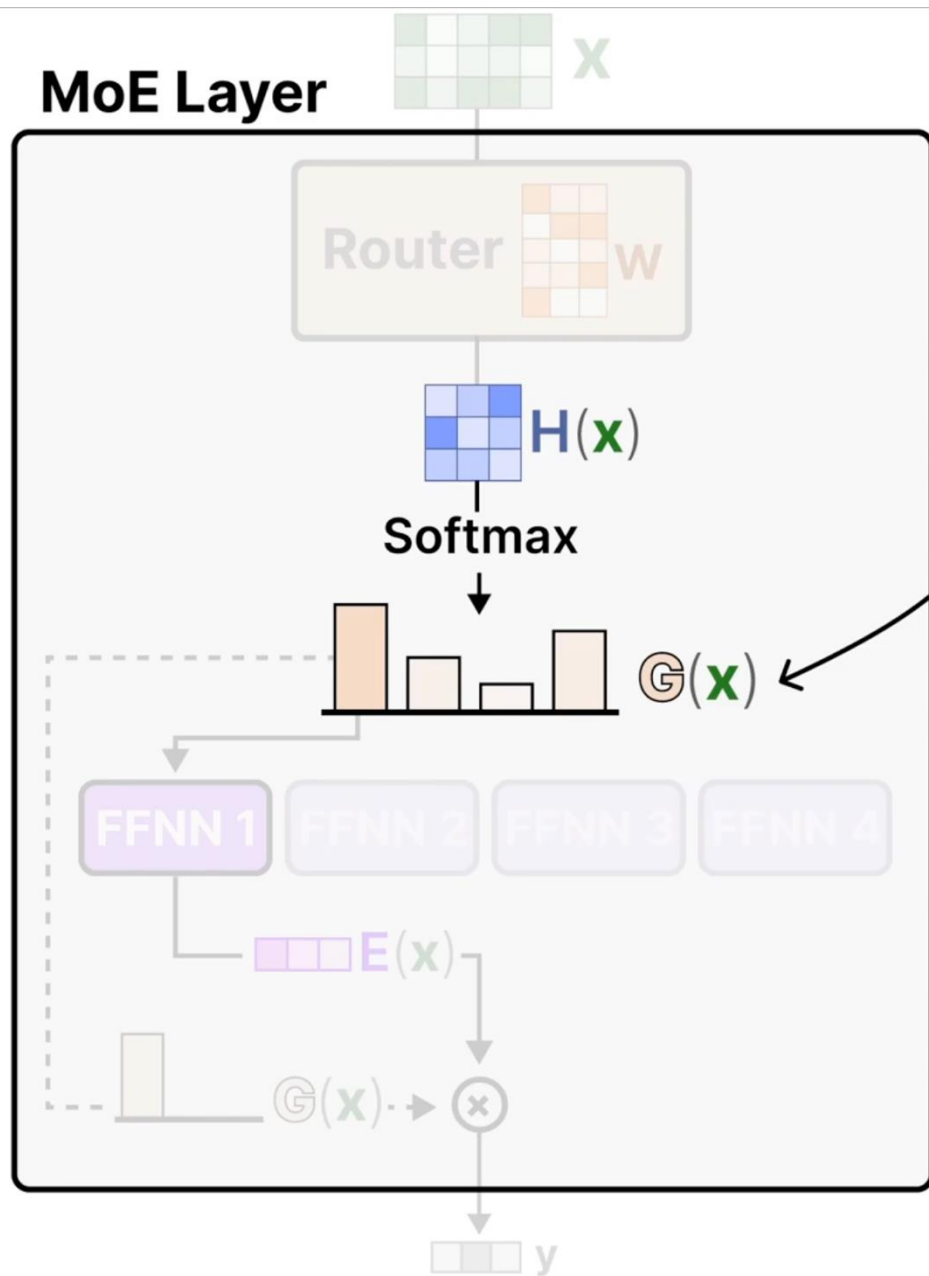


Then, the **softmax** of the output is taken...

output

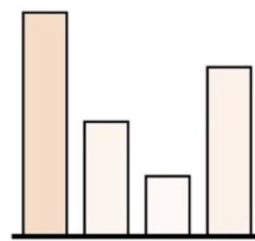
$\text{Softmax} \left(\begin{matrix} H(x) \end{matrix} \right)$





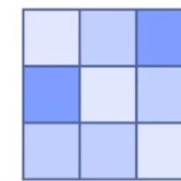
... to create **probabilities**
 $G(x)$, one for each expert.

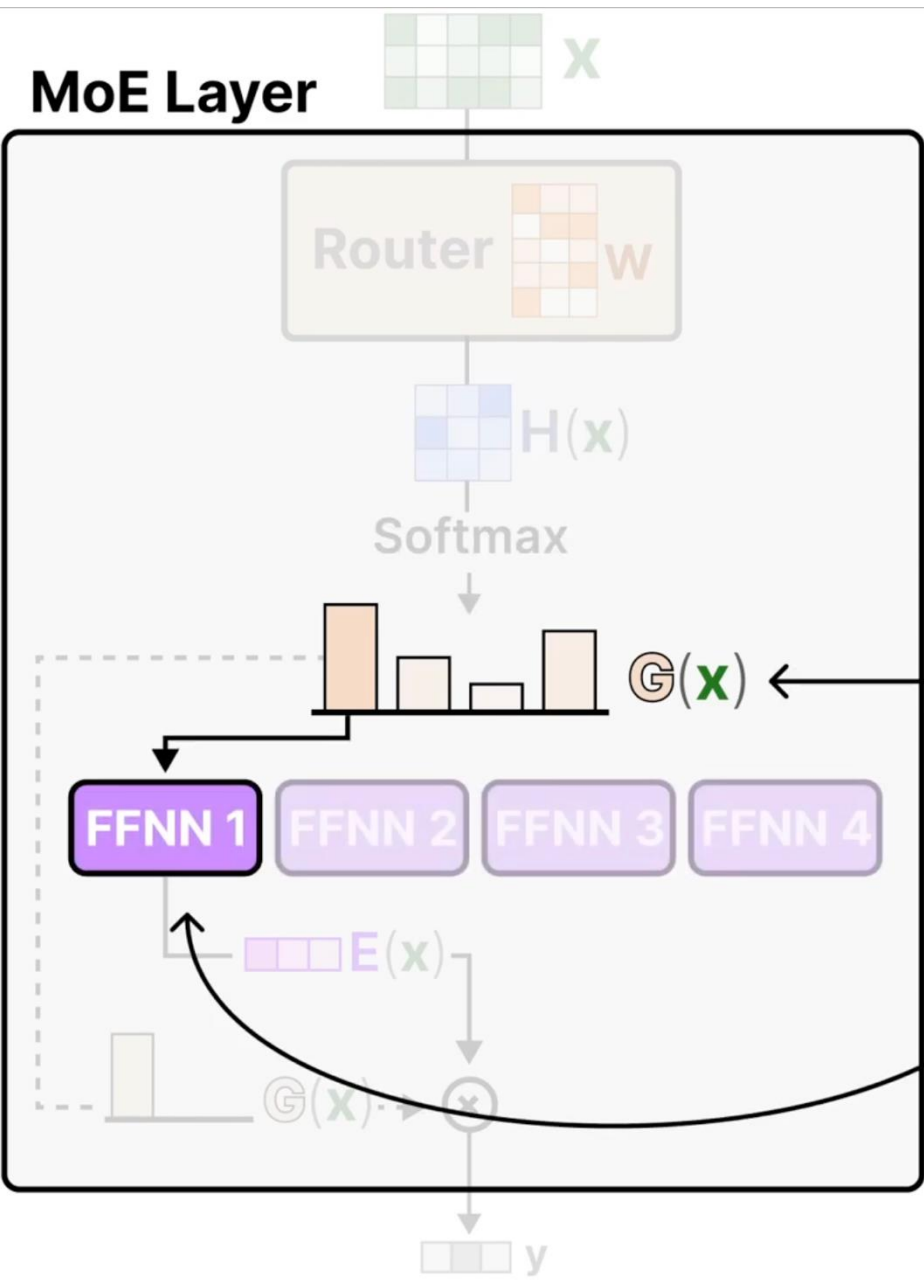
probability
distribution per expert



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

output

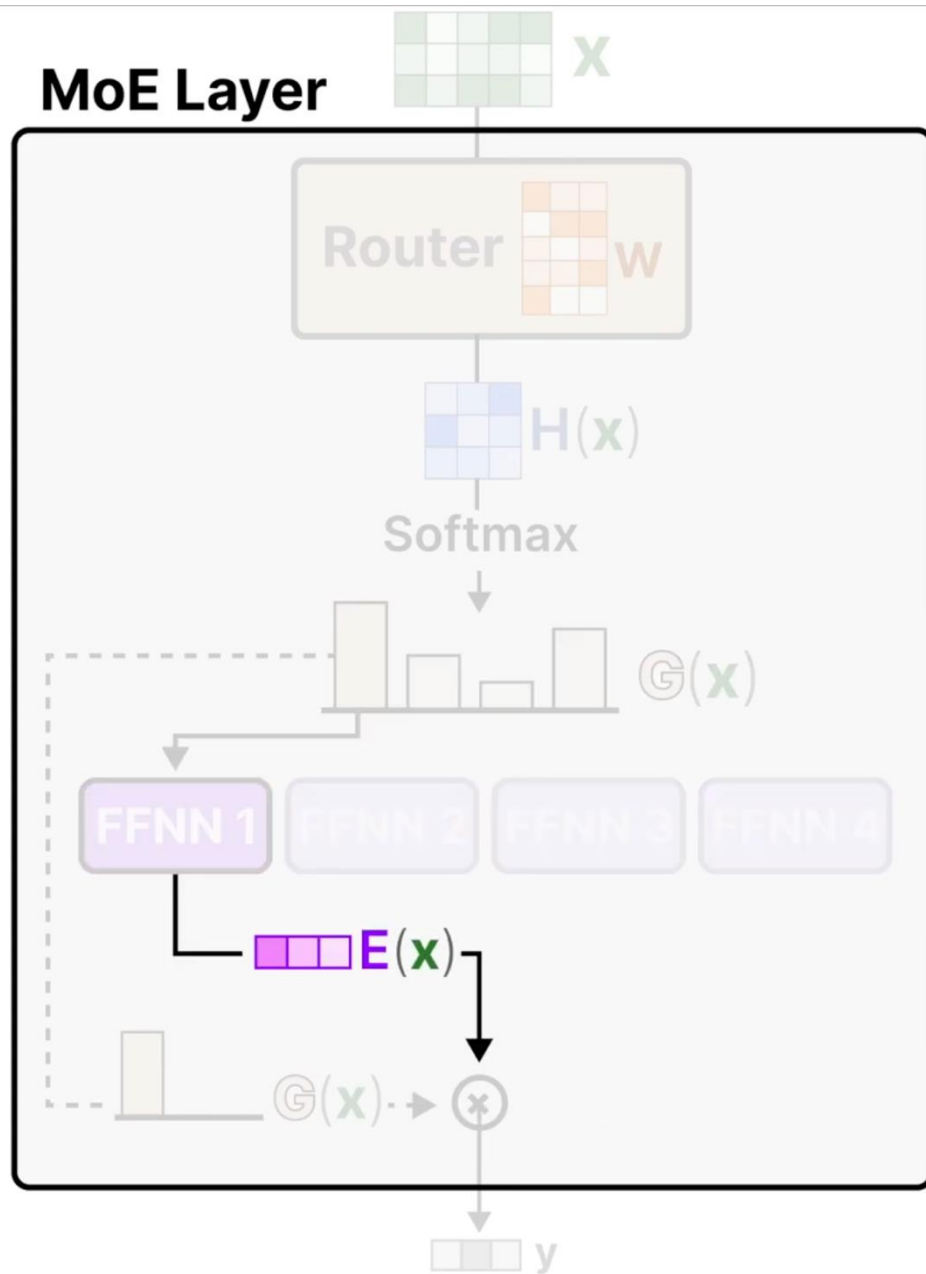




The router uses this **probability distribution** to choose the best matching **expert**.

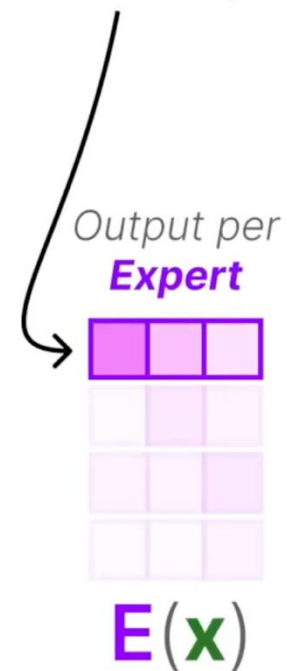


MoE Layer

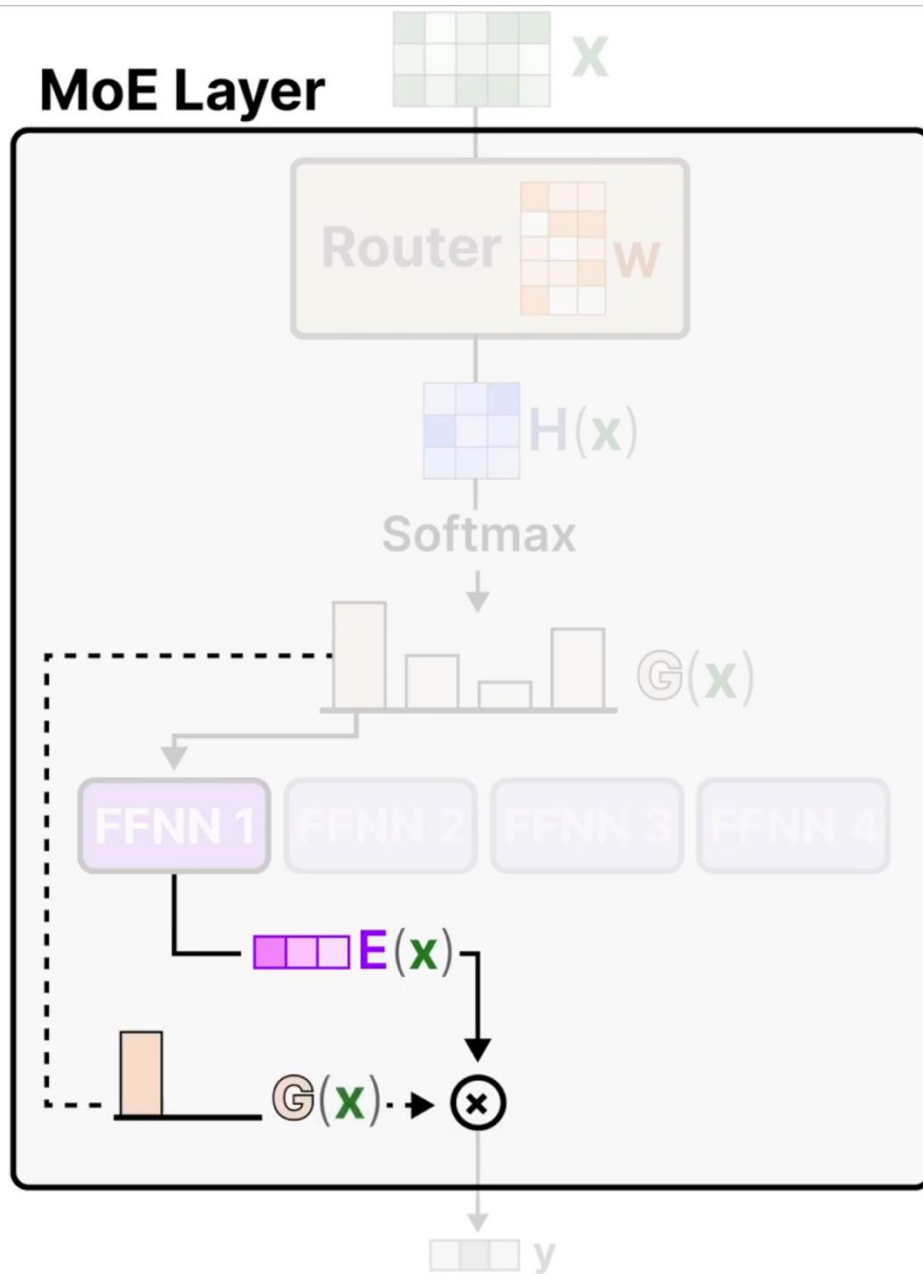


Selection of Experts

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

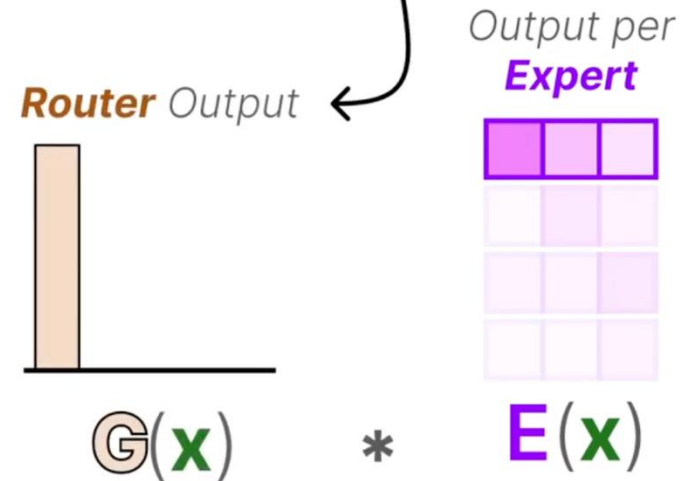


MoE Layer

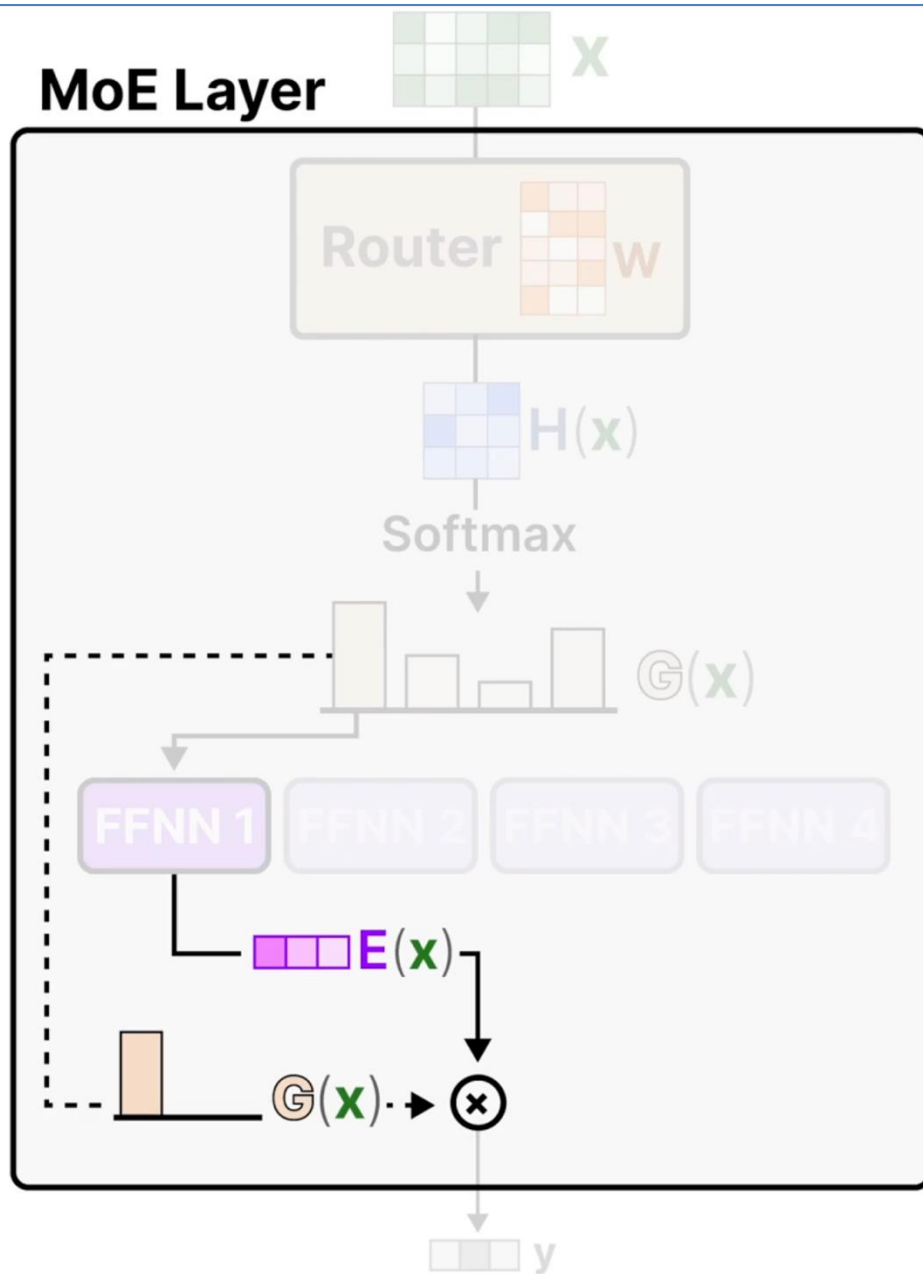


Selection of Experts

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



MoE Layer



Selection of Experts

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

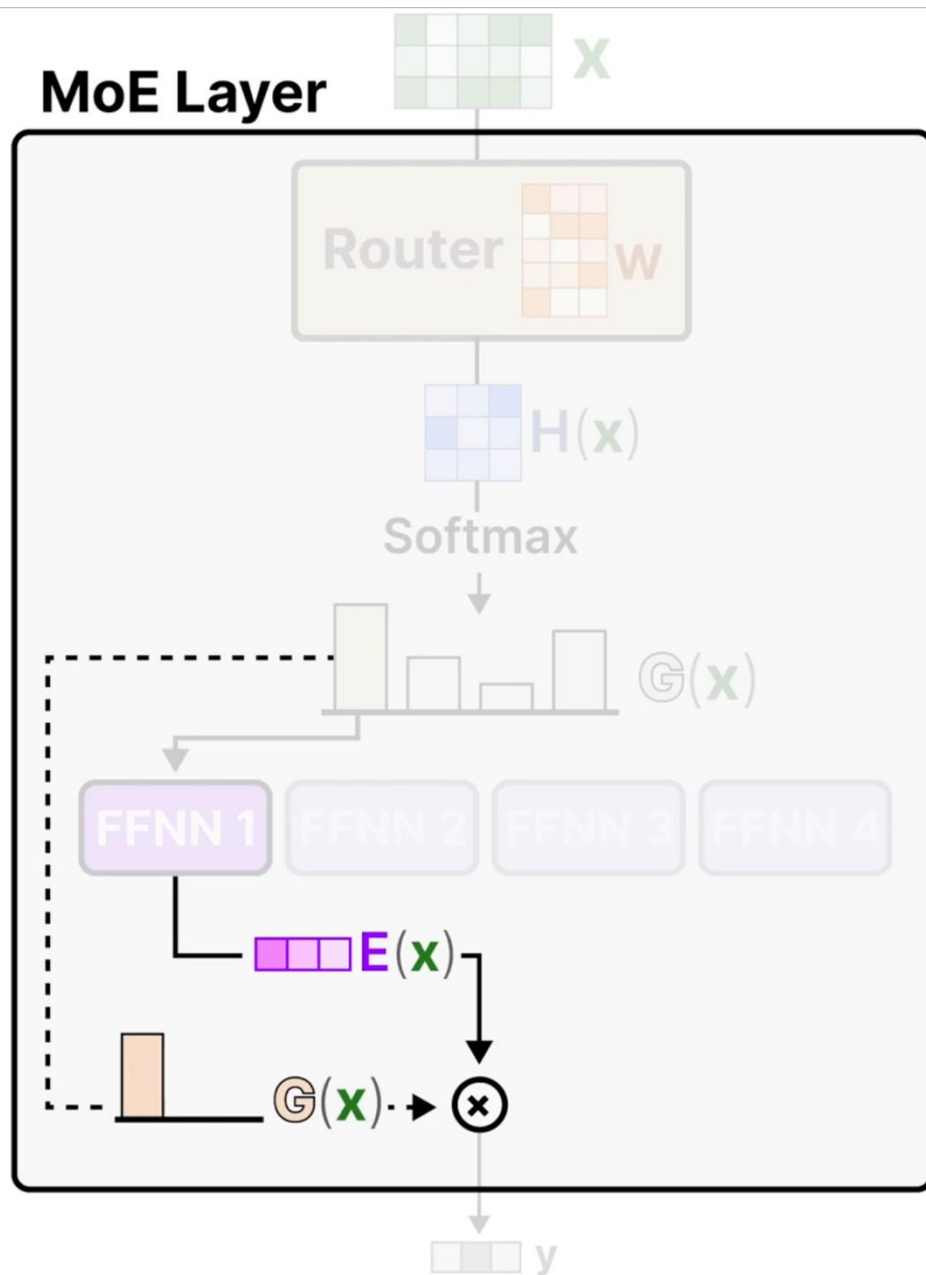
The diagram shows the summation process for the selected experts. A curved arrow points from the text "We do this for every expert selected." to a summation symbol $\sum$. The summation is over the product of the Router Output (represented as a bar chart) and the expert output $E(x)$ (a 4x3 grid). The equation is:

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

The Router Output is also labeled as $G(x)$ in the equation.



MoE Layer



Selection of Experts

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

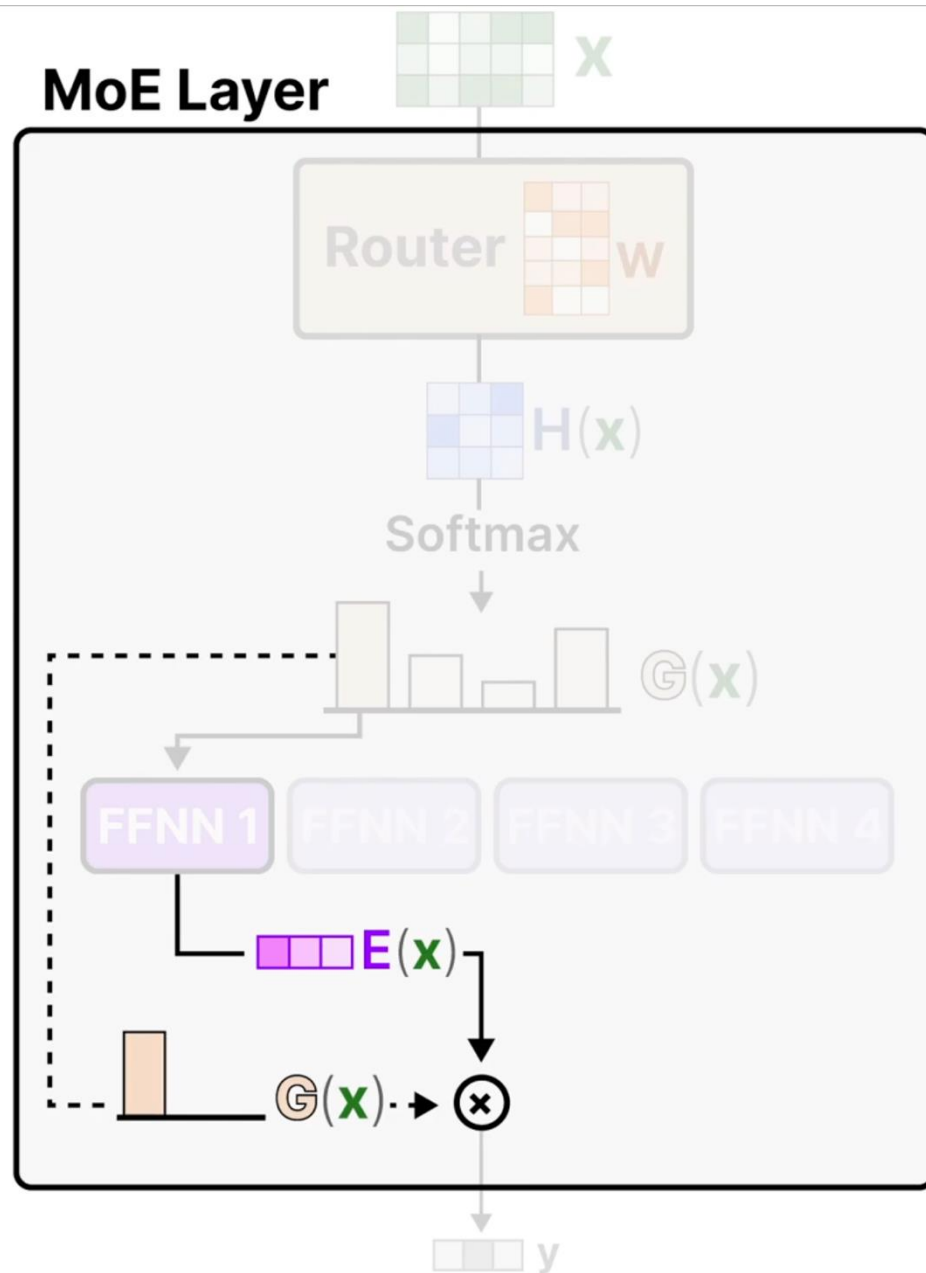
Router Output

Output per Expert

$$\sum \left(G(x) * E(x) \right)$$



MoE Layer



Selection of Experts

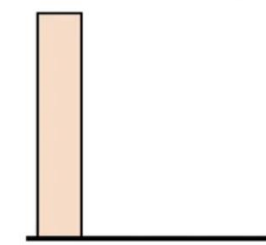
This creates our output, **one vector** for each **expert**.

Sparse MoE
output

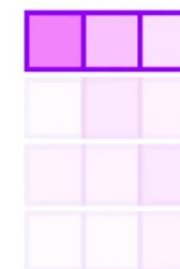


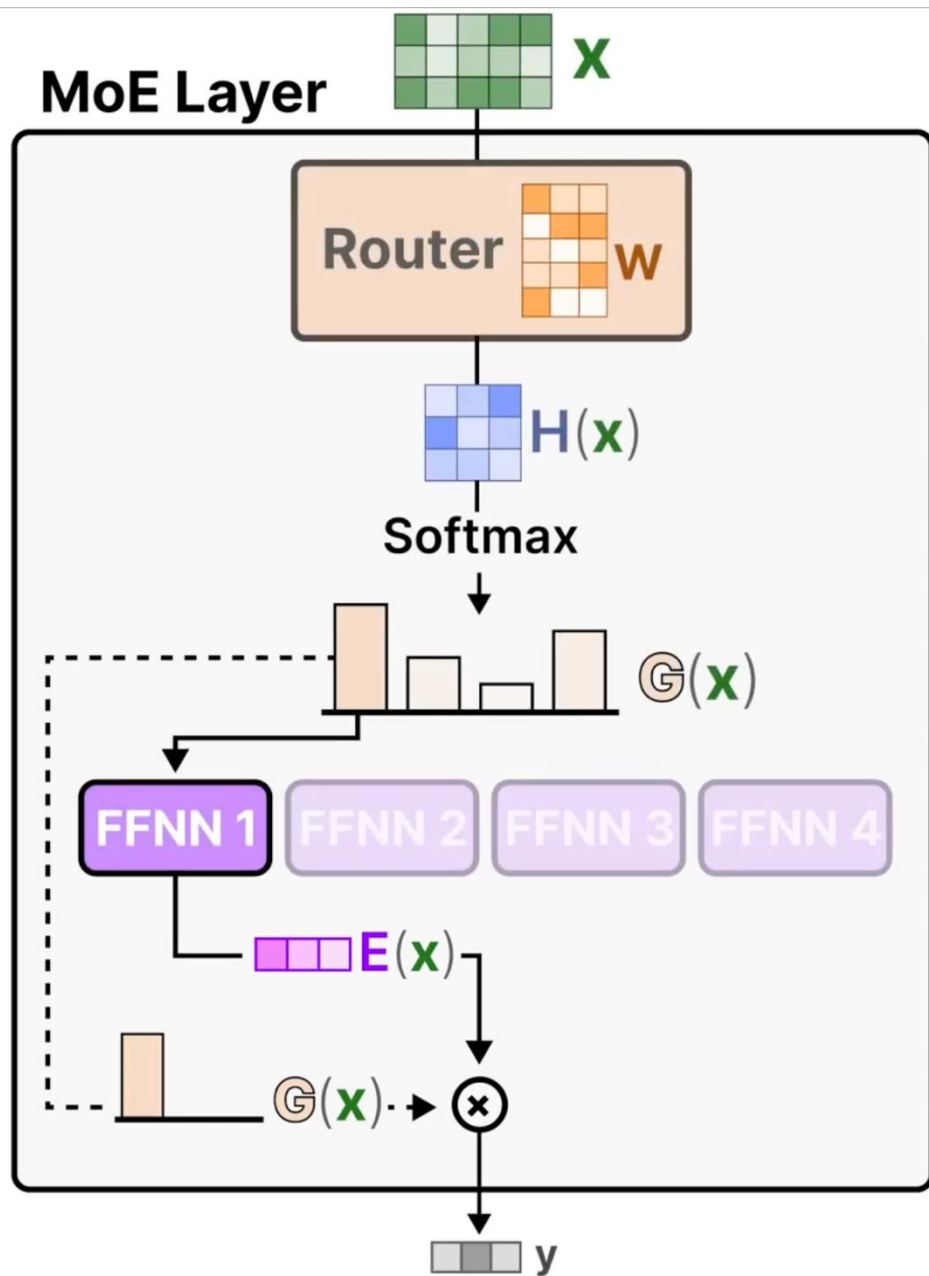
$$y = \sum (G(x) * E(x))$$

Router Output



Output per
Expert





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

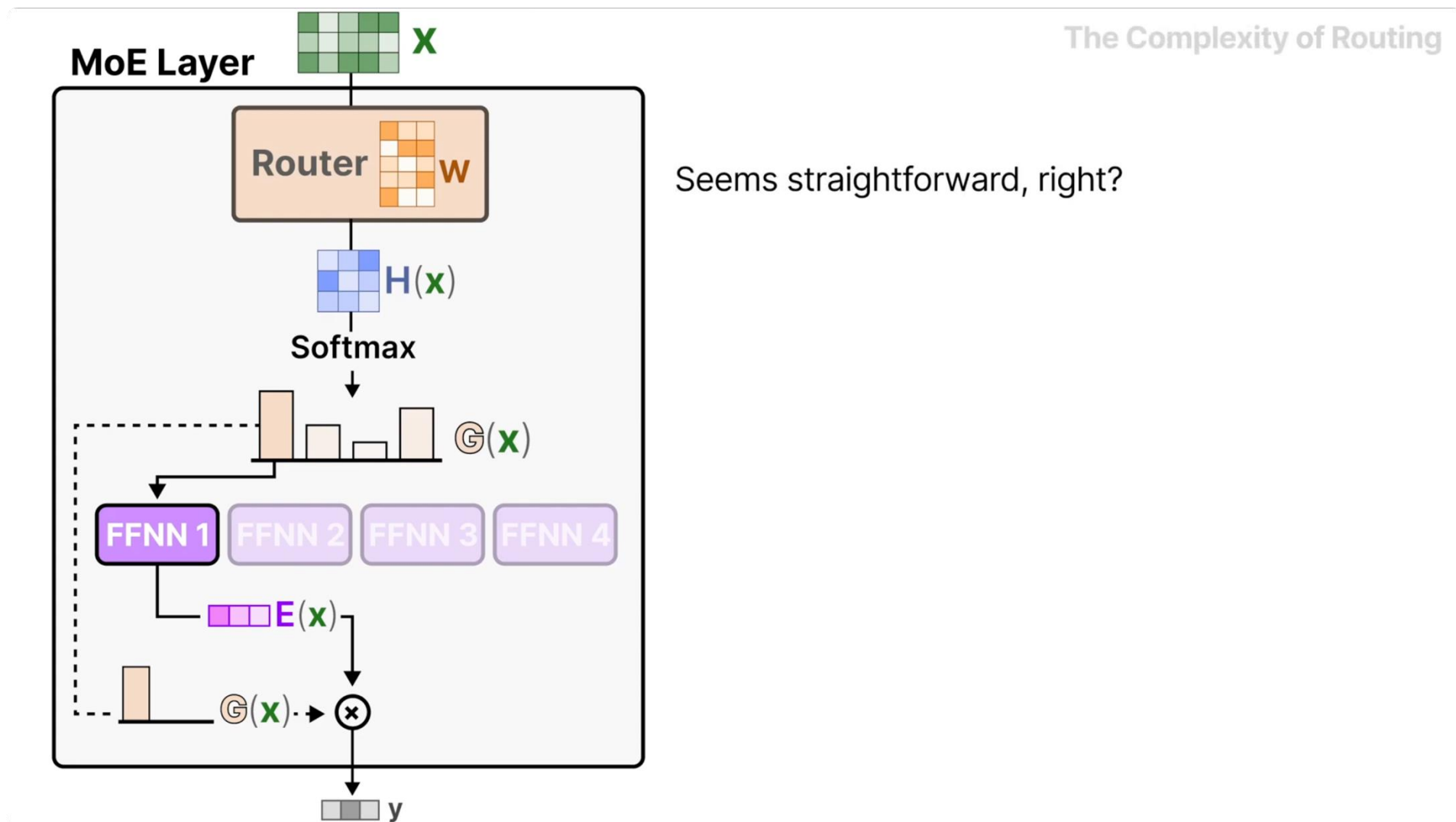


04

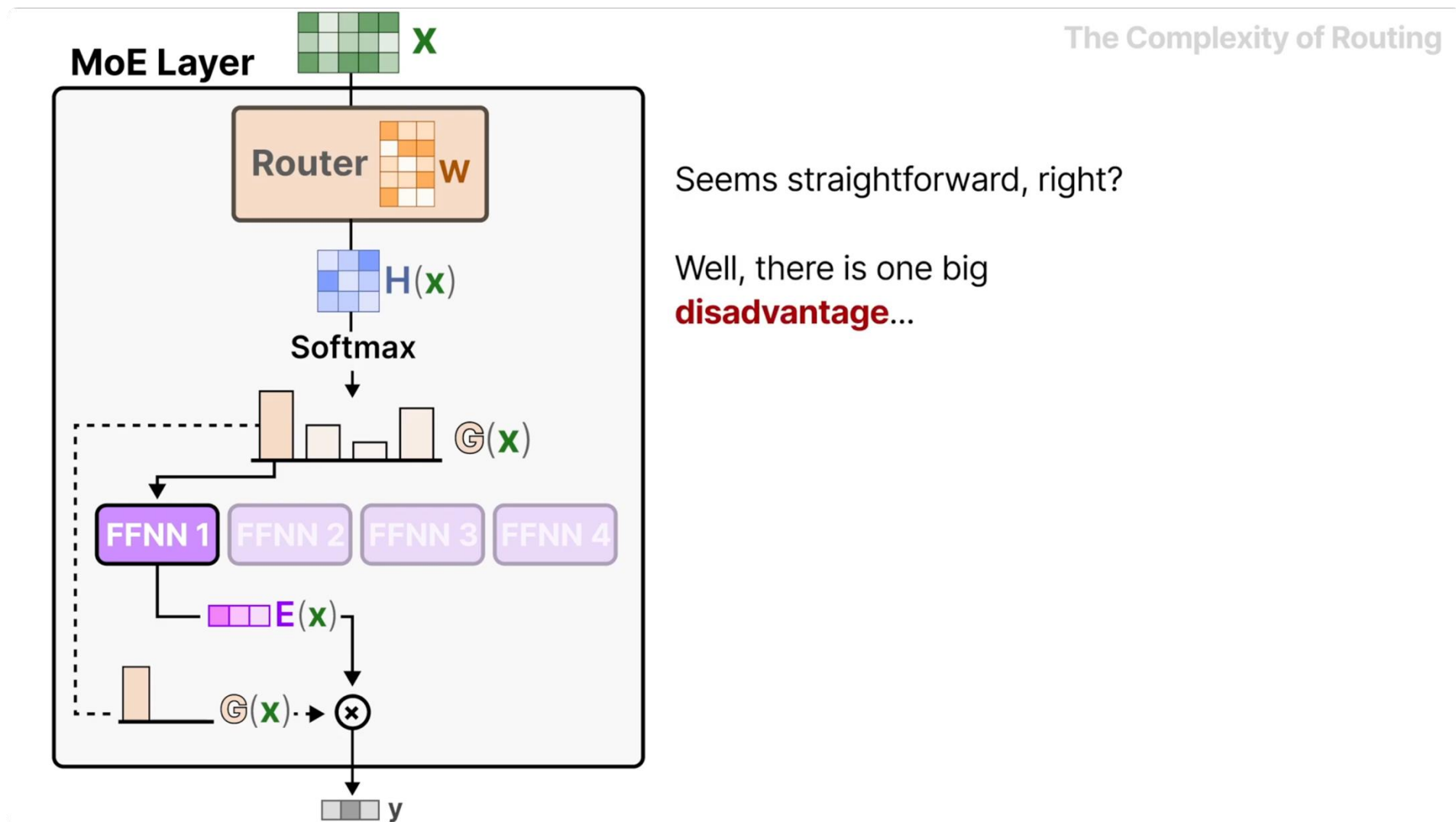
Load Balancing: 均衡负载



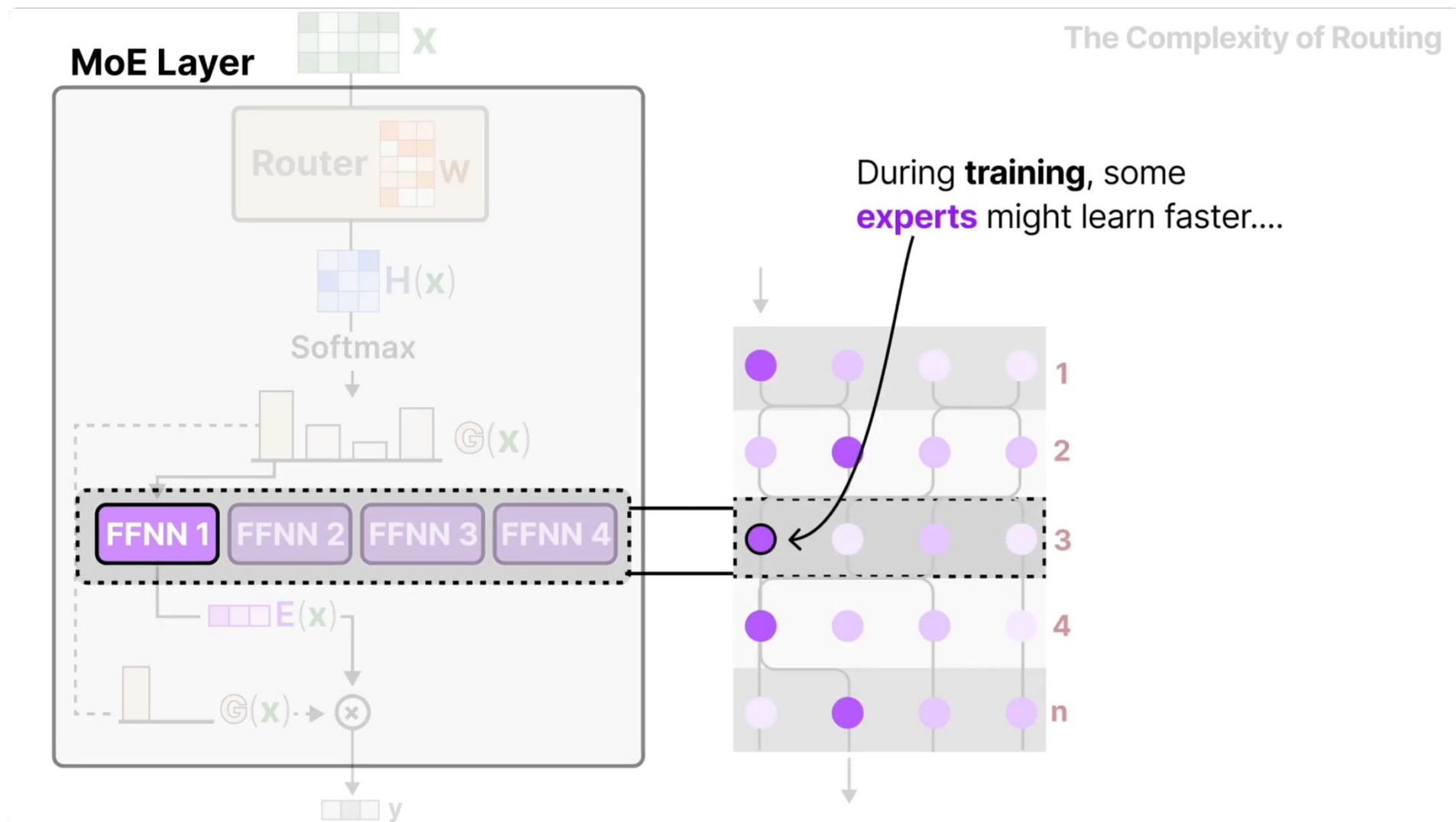
Load Balancing: 均衡负载



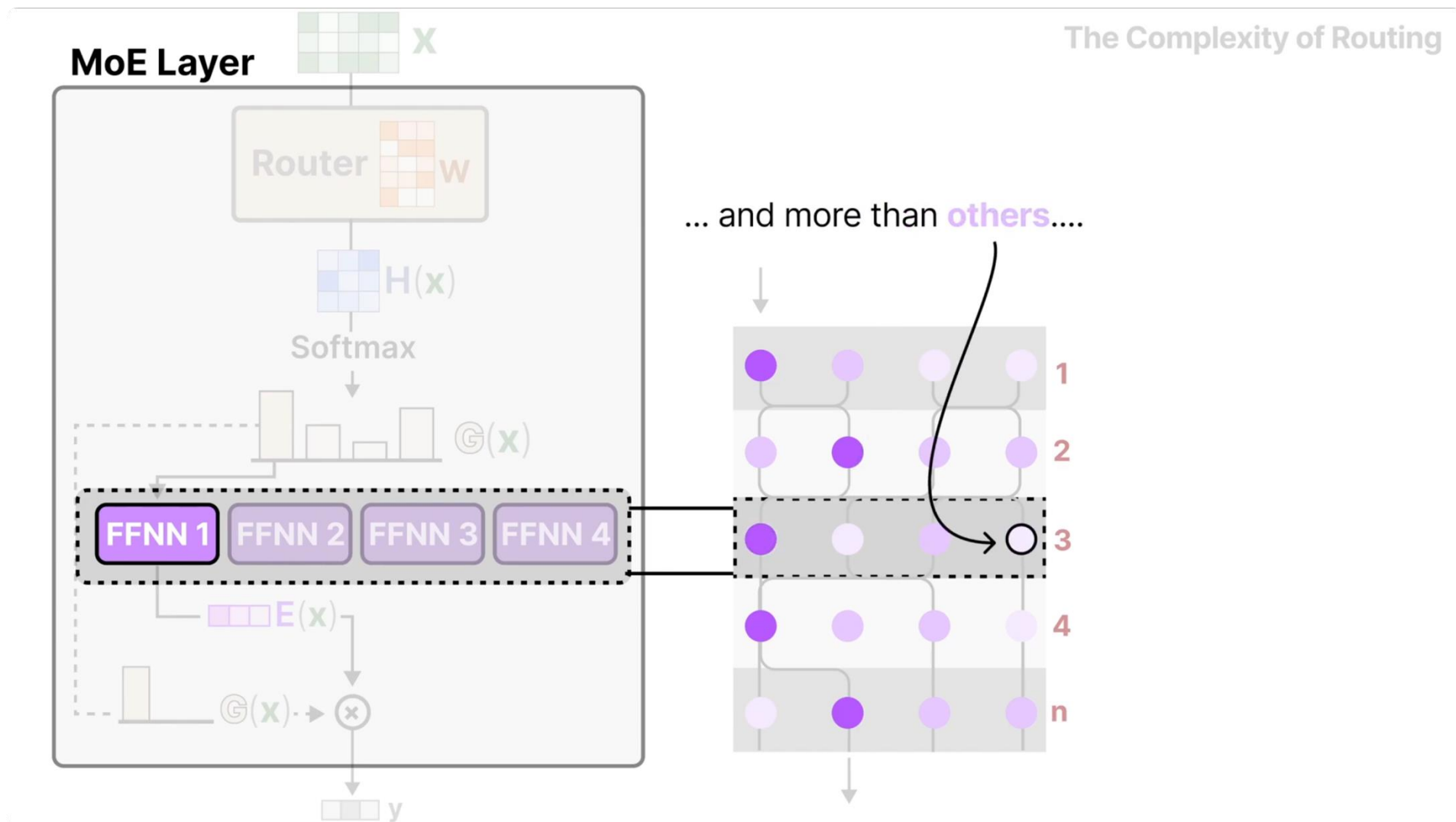
Load Balancing: 均衡负载



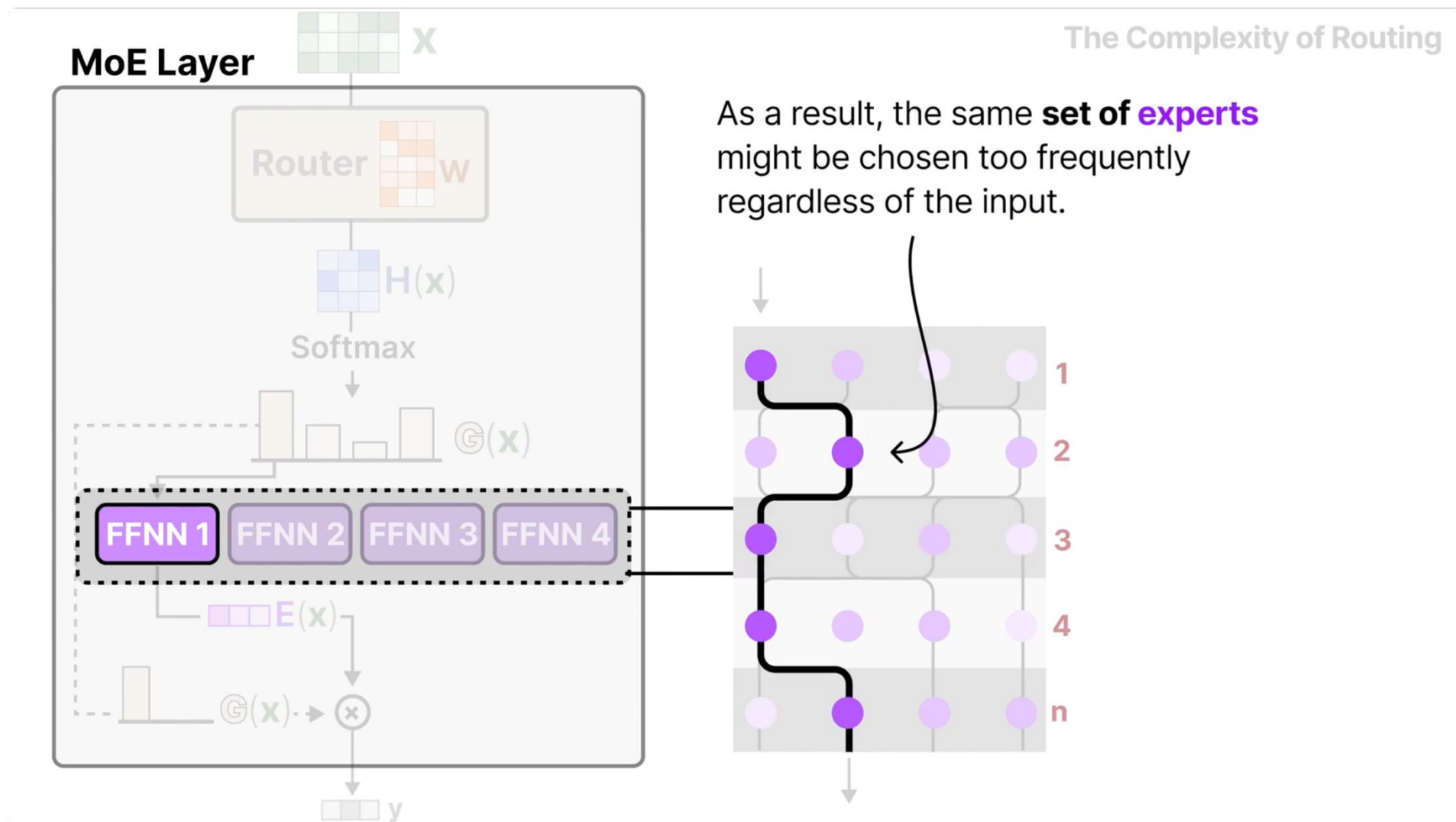
Load Balancing: 均衡负载



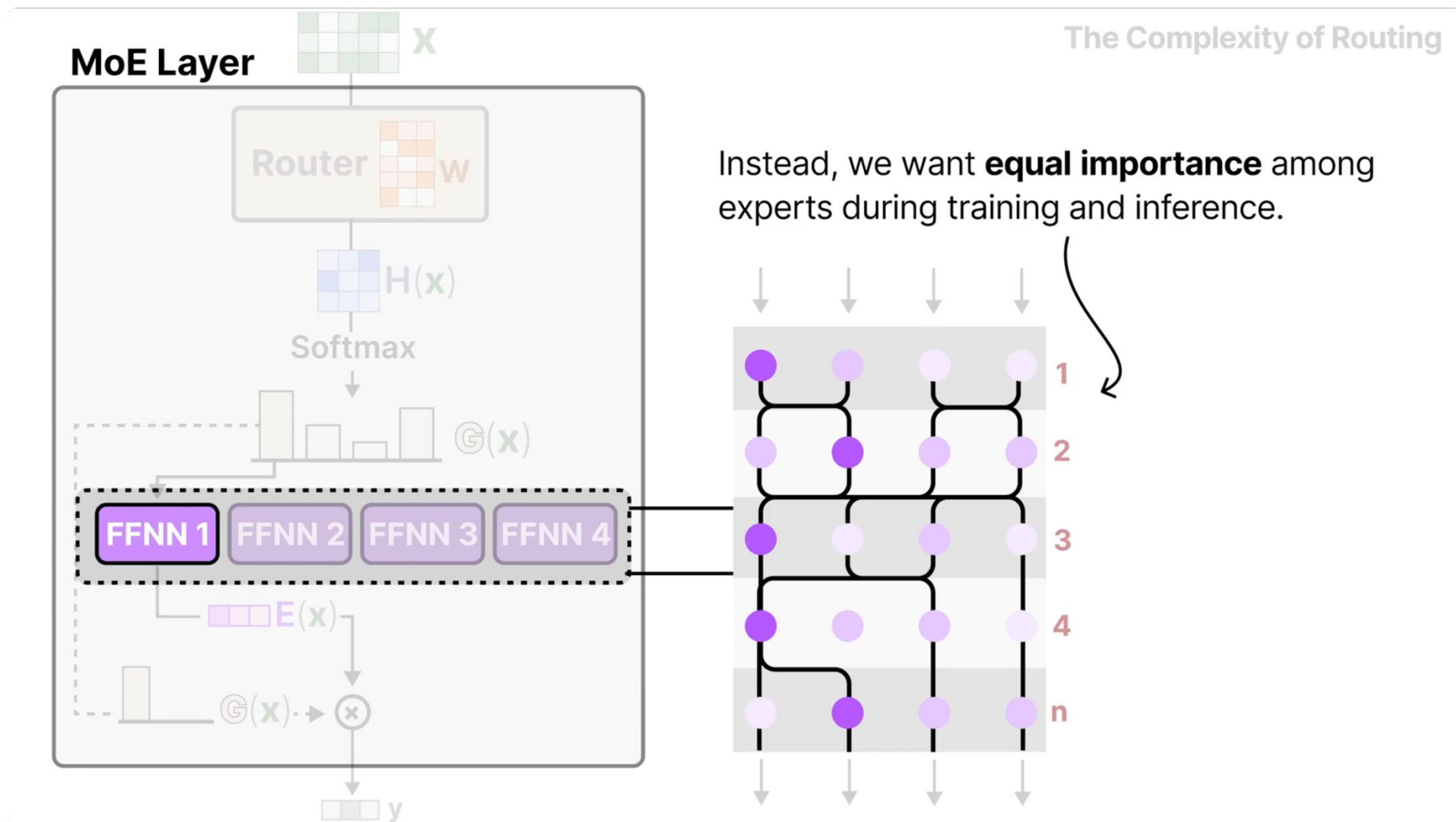
Load Balancing: 均衡负载



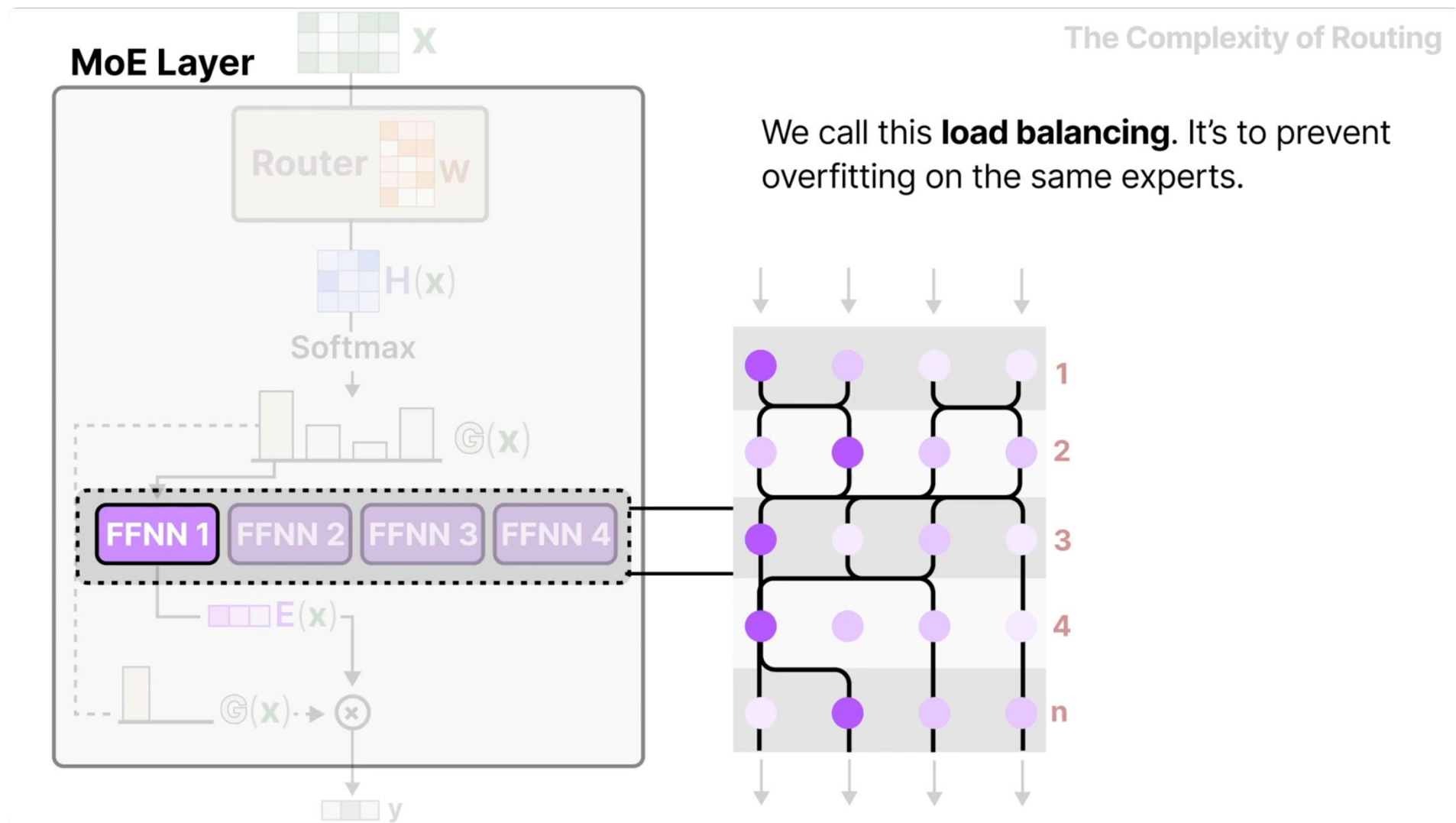
Load Balancing: 均衡负载



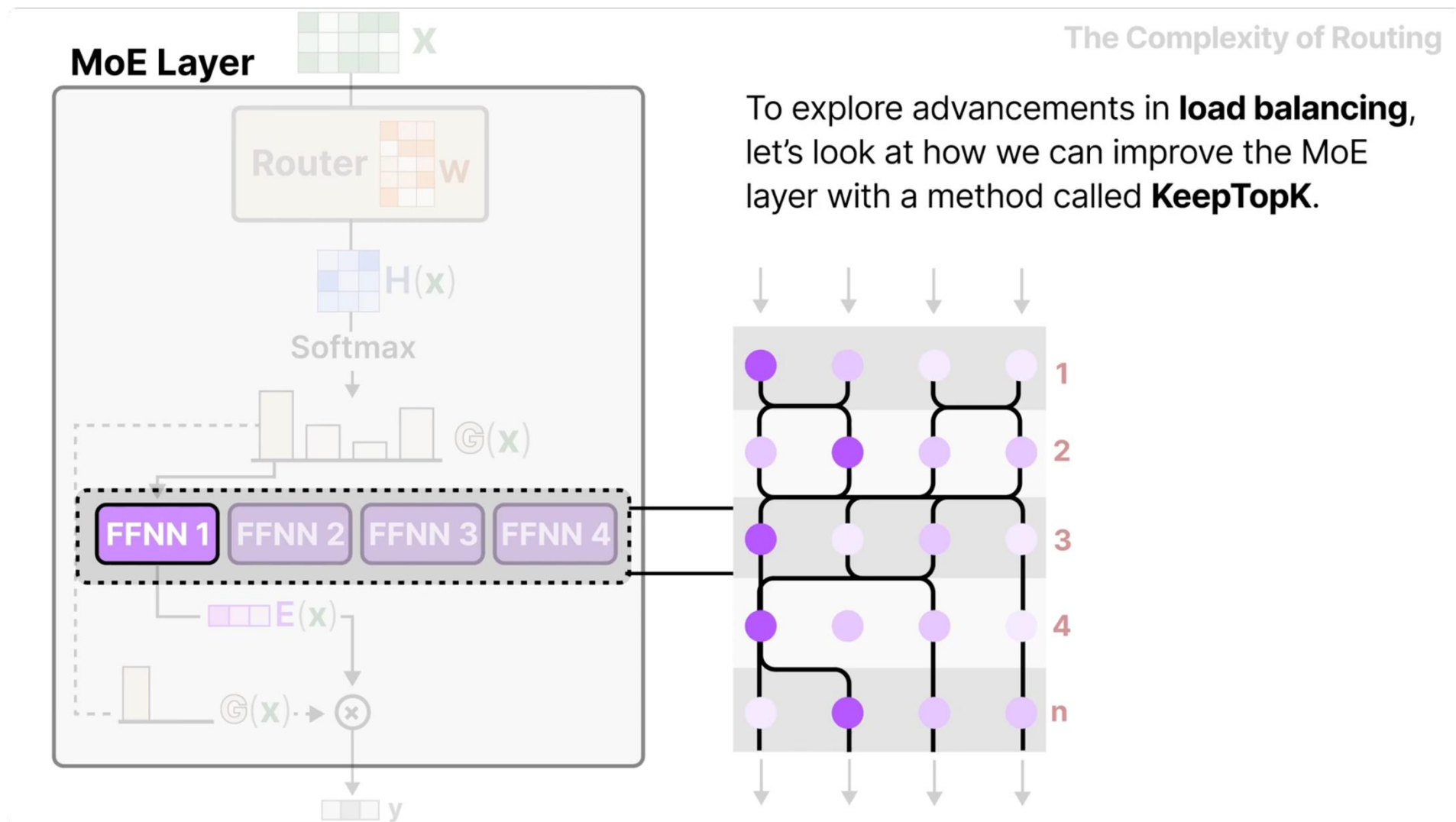
Load Balancing: 均衡负载



Load Balancing: 均衡负载



Load Balancing: 均衡负载

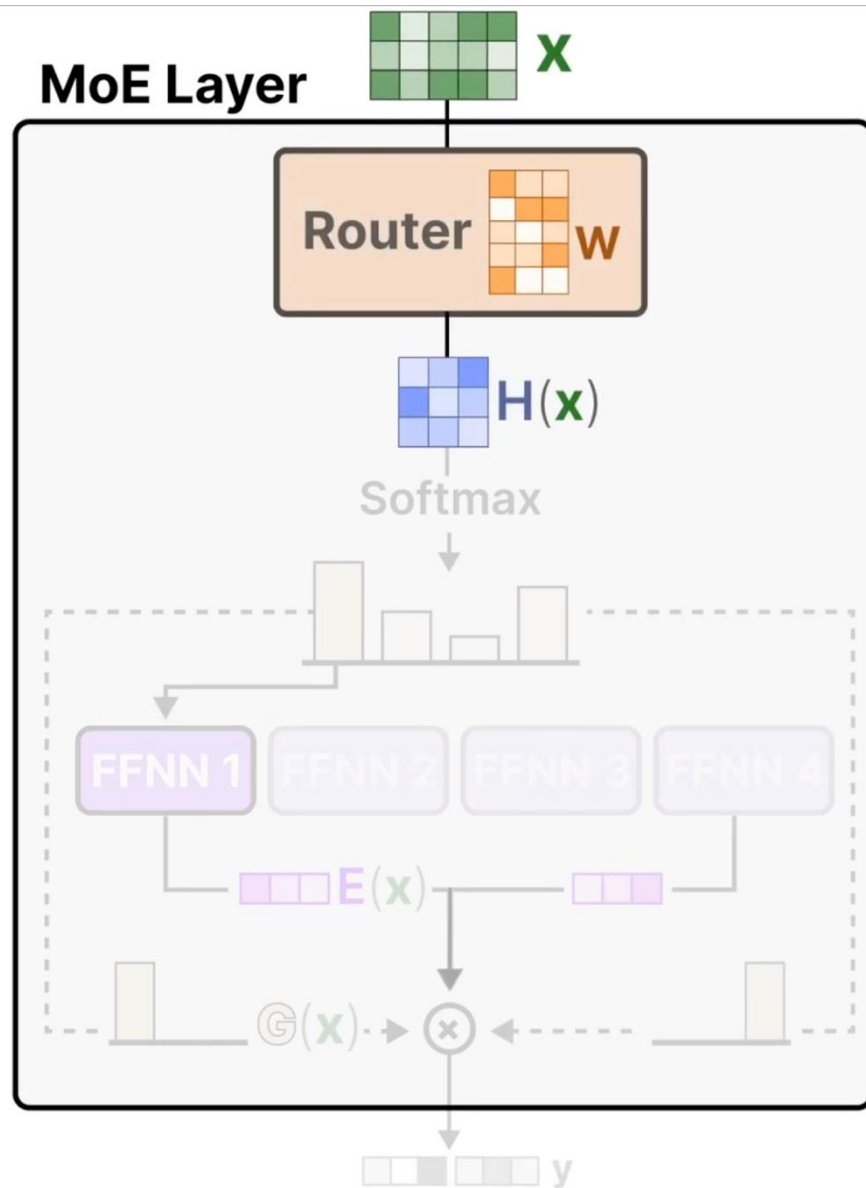


05

Keep Top-K: 专家选择



Keep Top-K: 专家选择



KeepTopK

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

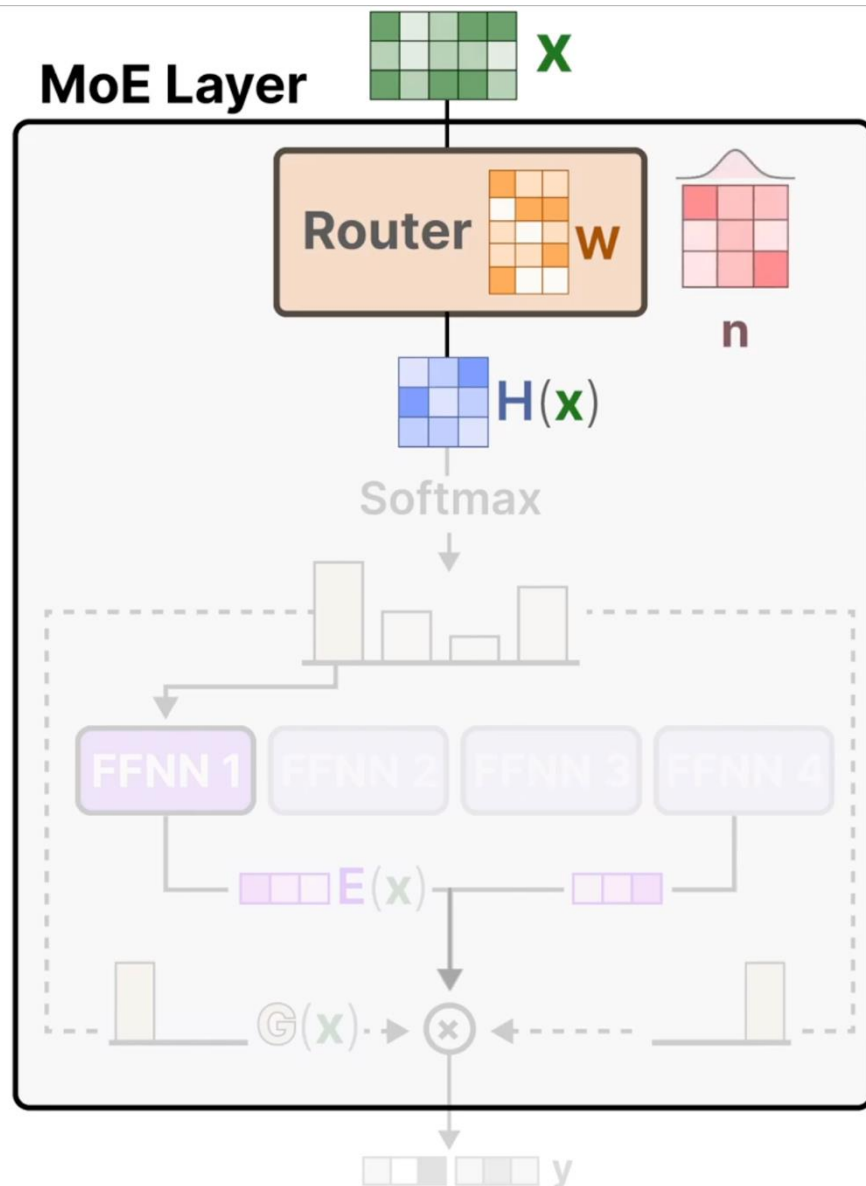
output

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

router weights



Keep Top-K: 专家选择



KeepTopK

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

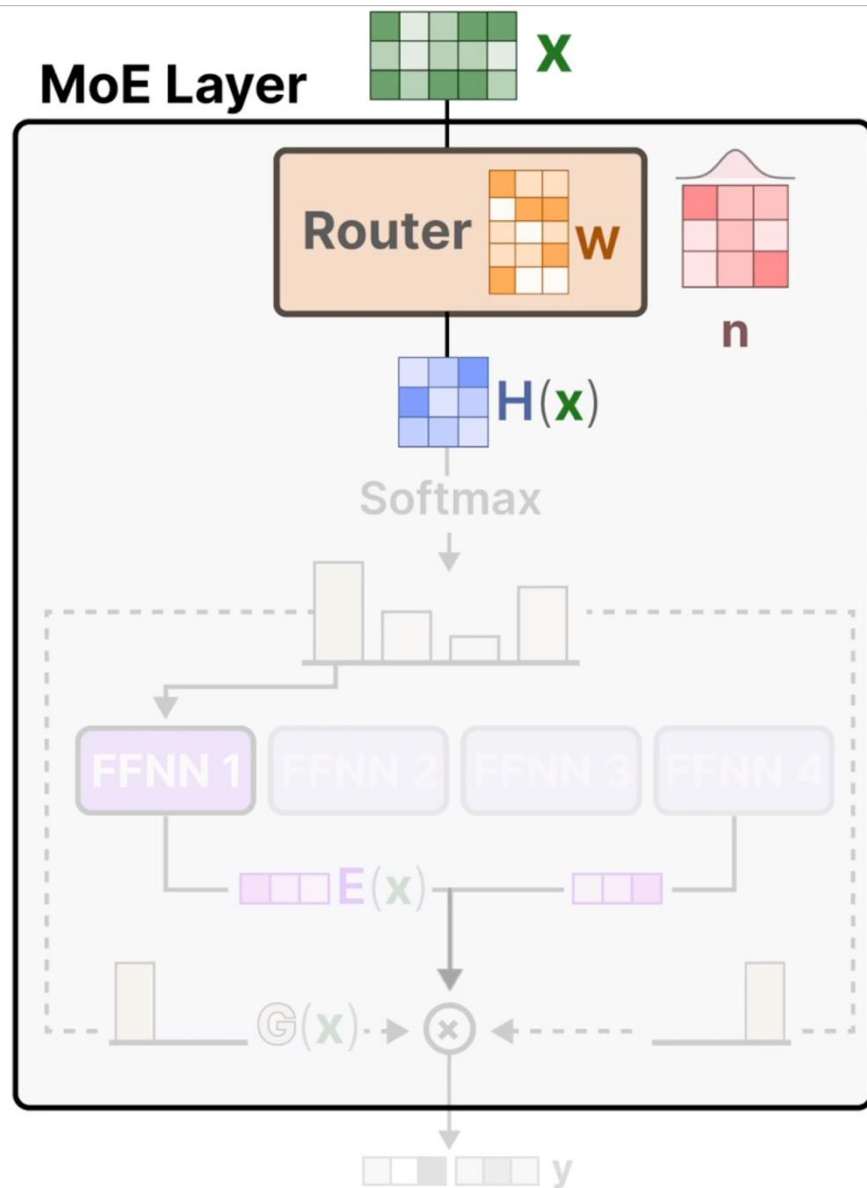
(somewhat noisy) output

$$H(x) = x * W + n$$

input router weights (small amount of) gaussian noise

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

Keep Top-K: 专家选择



KeepTopK

(somewhat noisy) output

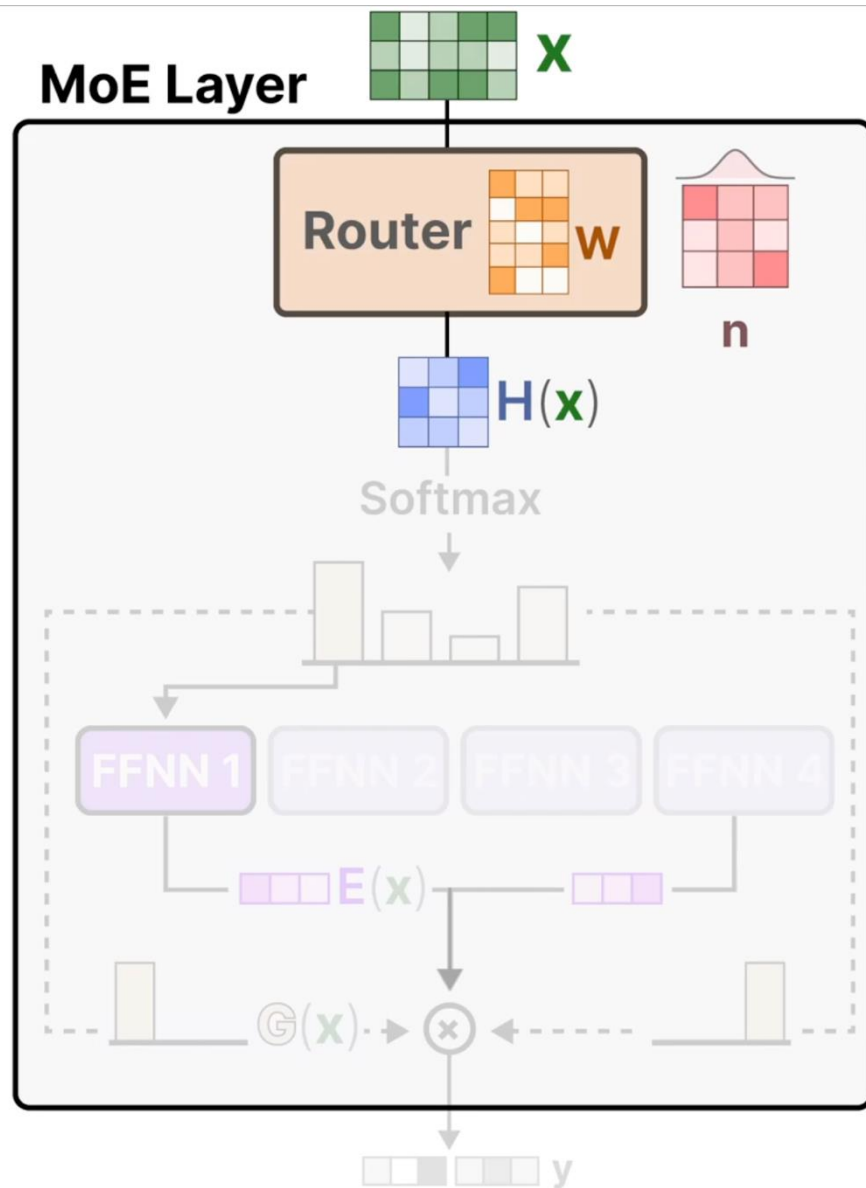
$$H(x) = x * W + n$$

x (input) W (router weights) n (small amount of gaussian noise)

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



Keep Top-K: 专家选择



KeepTopK

(somewhat noisy) output

$H(x) = x * W + n$

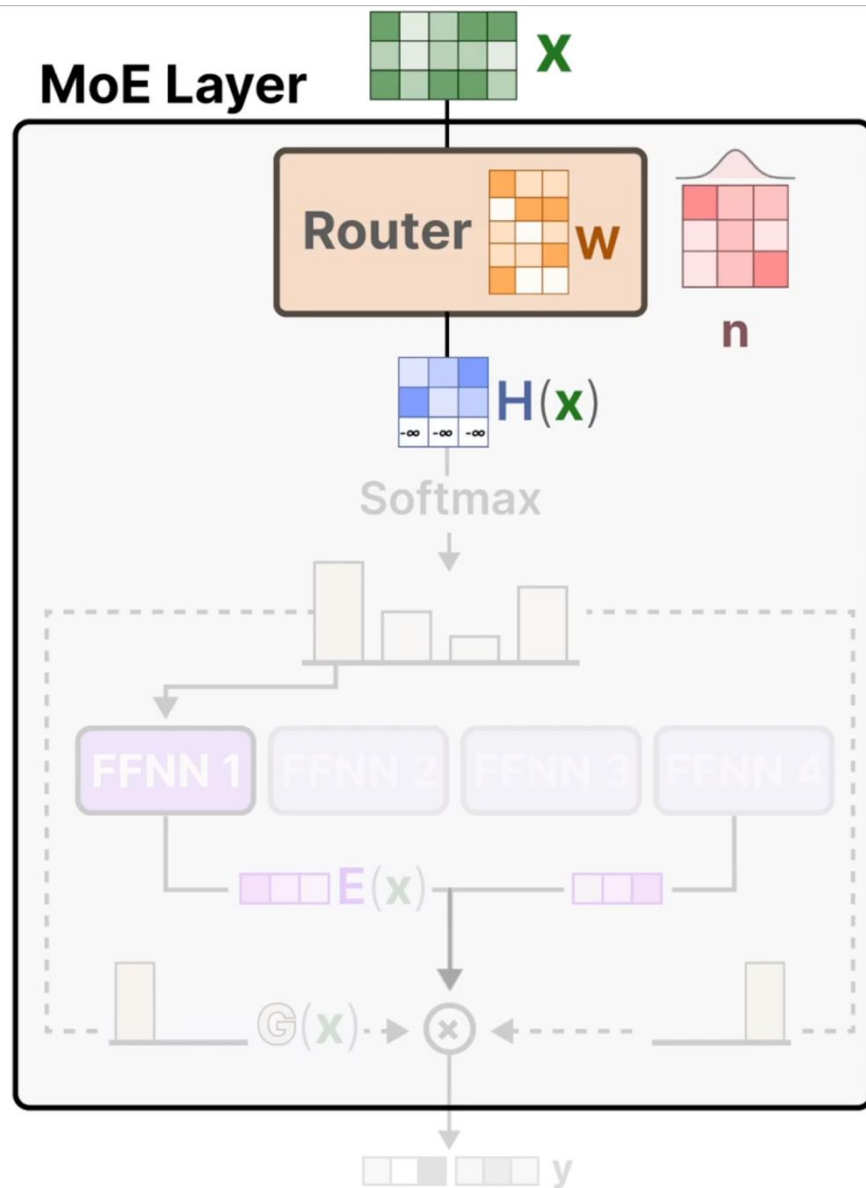
x (input)

W (router weights)

n (small amount of gaussian noise)

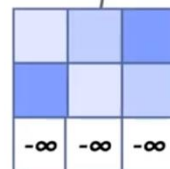
Thereby giving more opportunity for other experts to train.

Keep Top-K: 专家选择

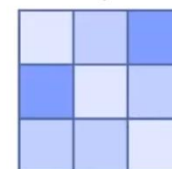


KeepTopK

(updated)
output



(somewhat noisy)
output

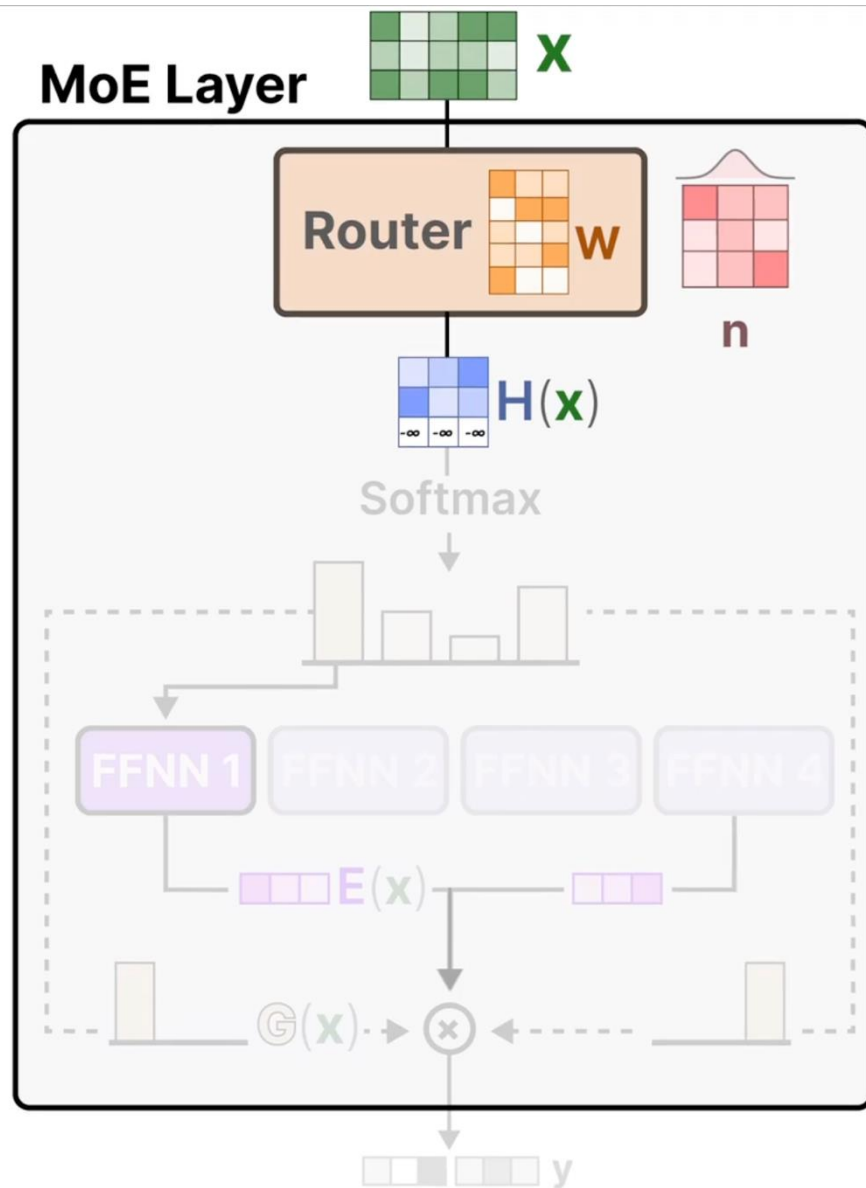


$$H'(x) = \text{KeepTopK} (H(x), 2)$$

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

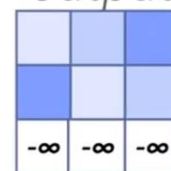


Keep Top-K: 专家选择



KeepTopK

(updated)
output

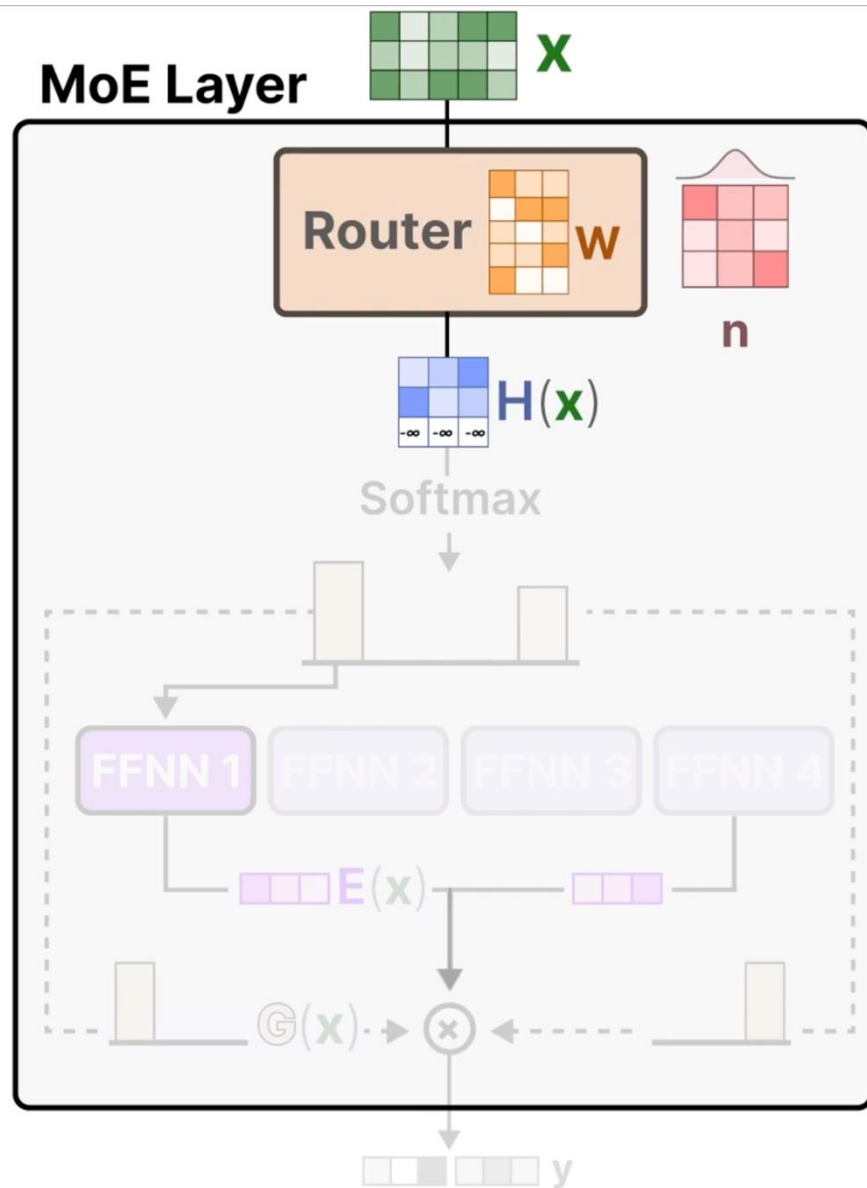


$H'(x)$

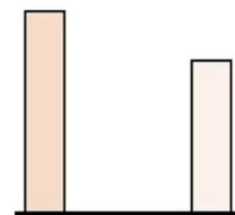
When we process this updated output...



Keep Top-K: 专家选择

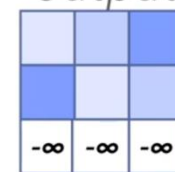


probability
distribution per expert



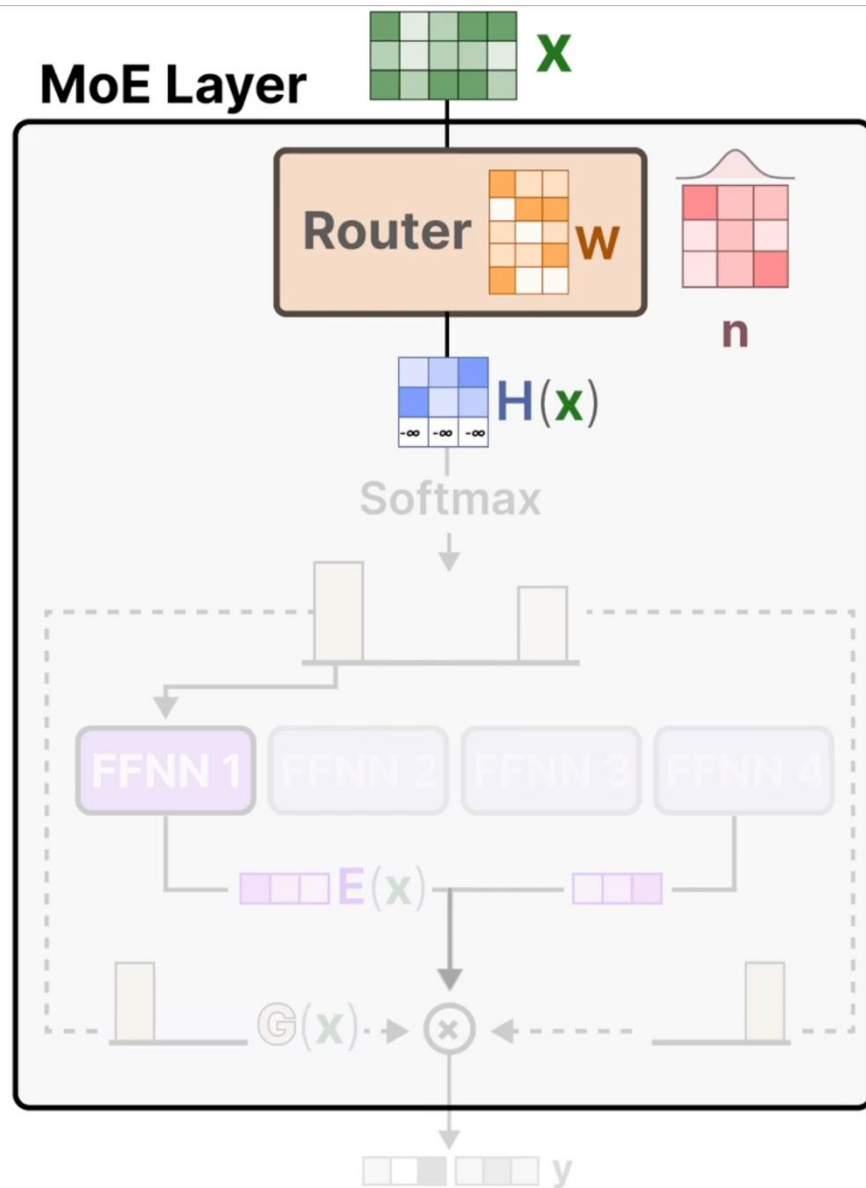
$$G(x) = \text{Softmax}(H'(x))$$

(updated)
output



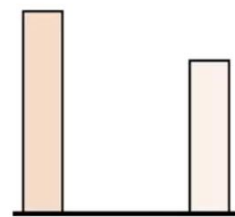
... with a softmax function ...

Keep Top-K: 专家选择

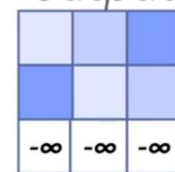


KeepTopK

probability
distribution per expert



(updated)
output

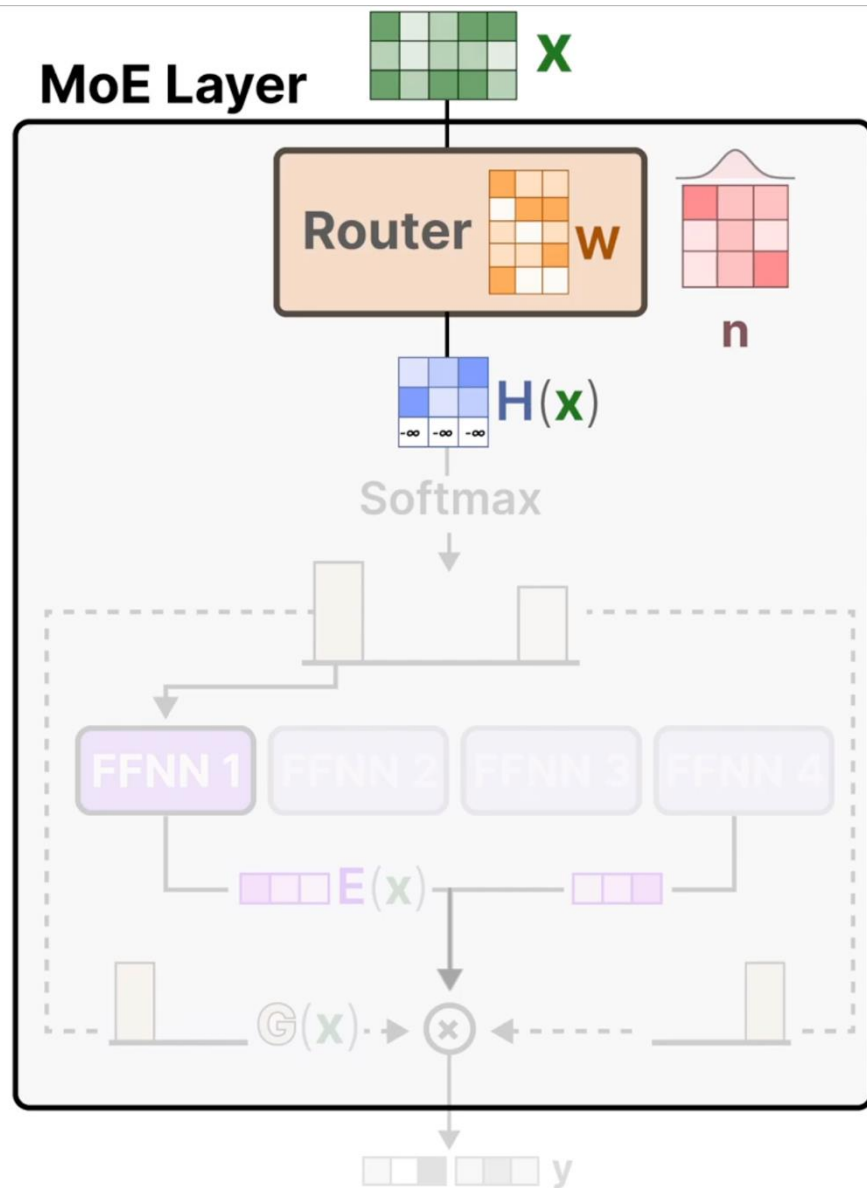


$$G(x) = \text{Softmax}(H'(x))$$

...the **output** will result in a probability of 0 of the values that were set to $-\infty$.

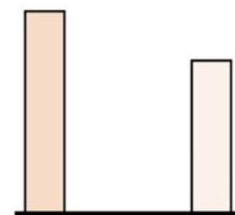


Keep Top-K: 专家选择

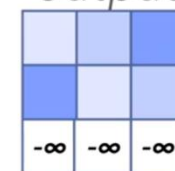


KeepTopK

probability
distribution per expert



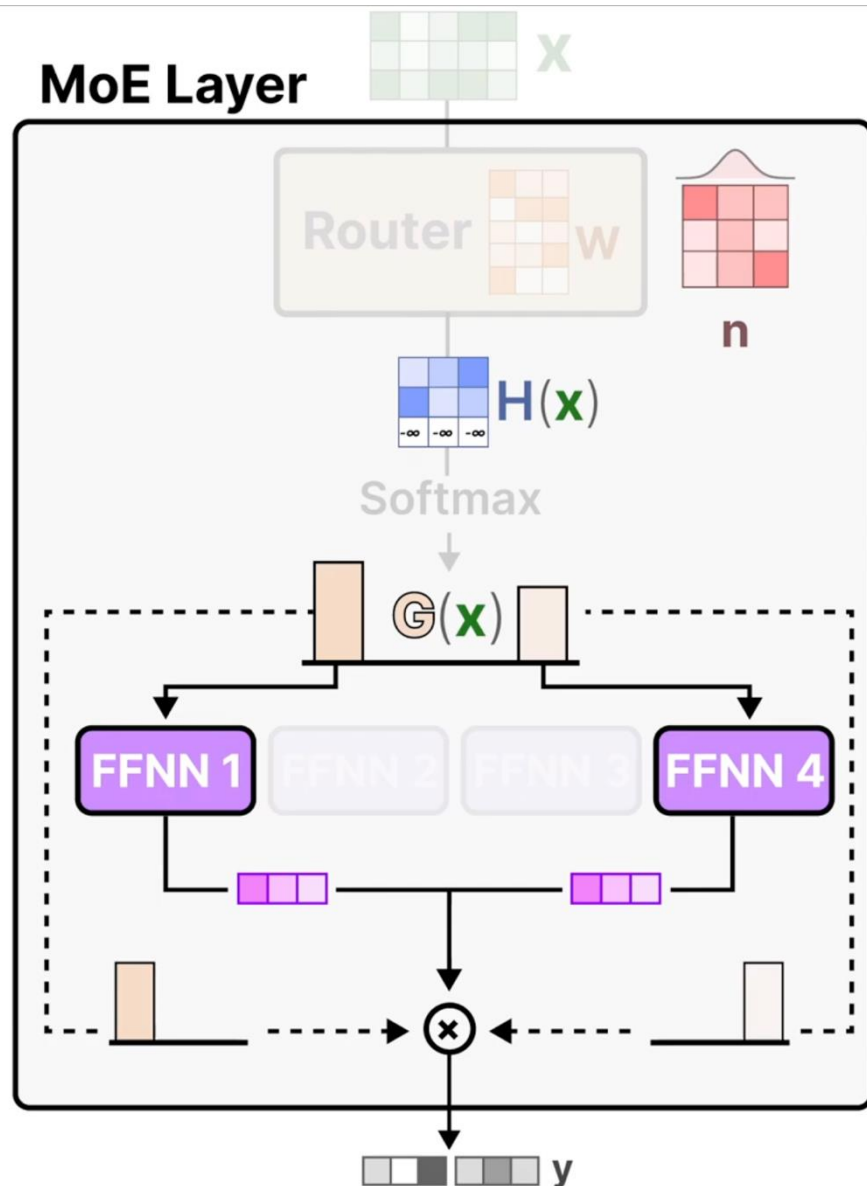
(updated)
output



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

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

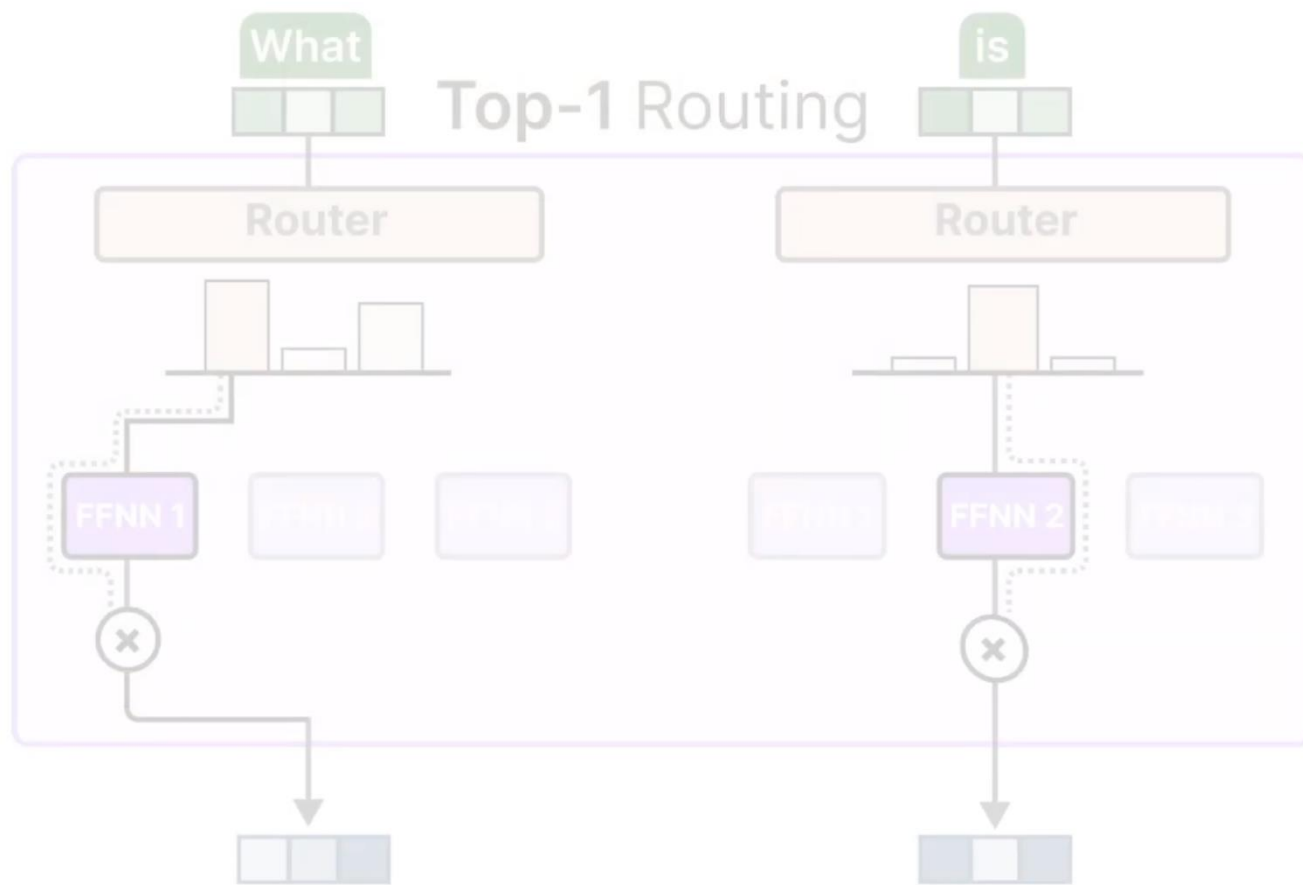
Keep Top-K: 专家选择



KeepTopK

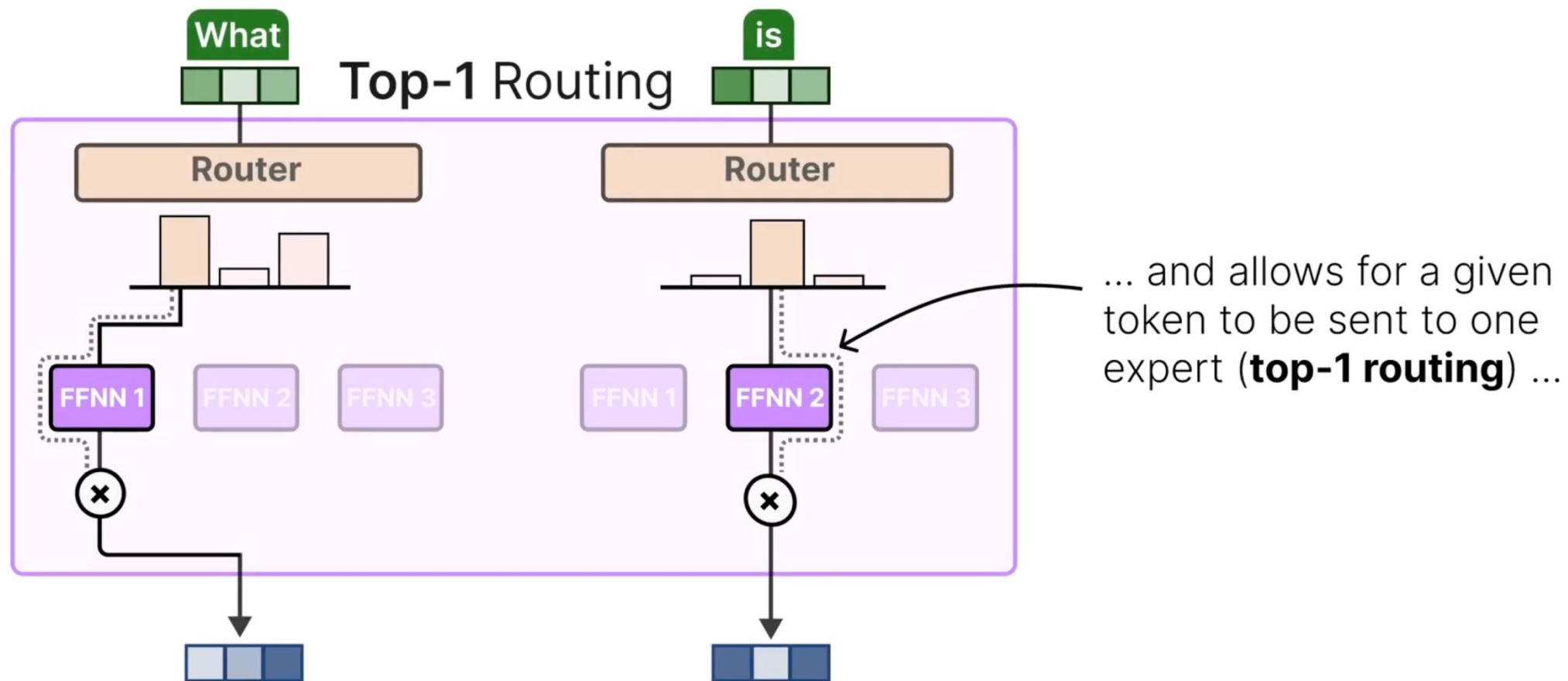
Combined, they allow undertrained **experts** to “catch up” to experts that were chosen more frequently and therefore had more opportunity to train.

Keep Top-K: 专家选择

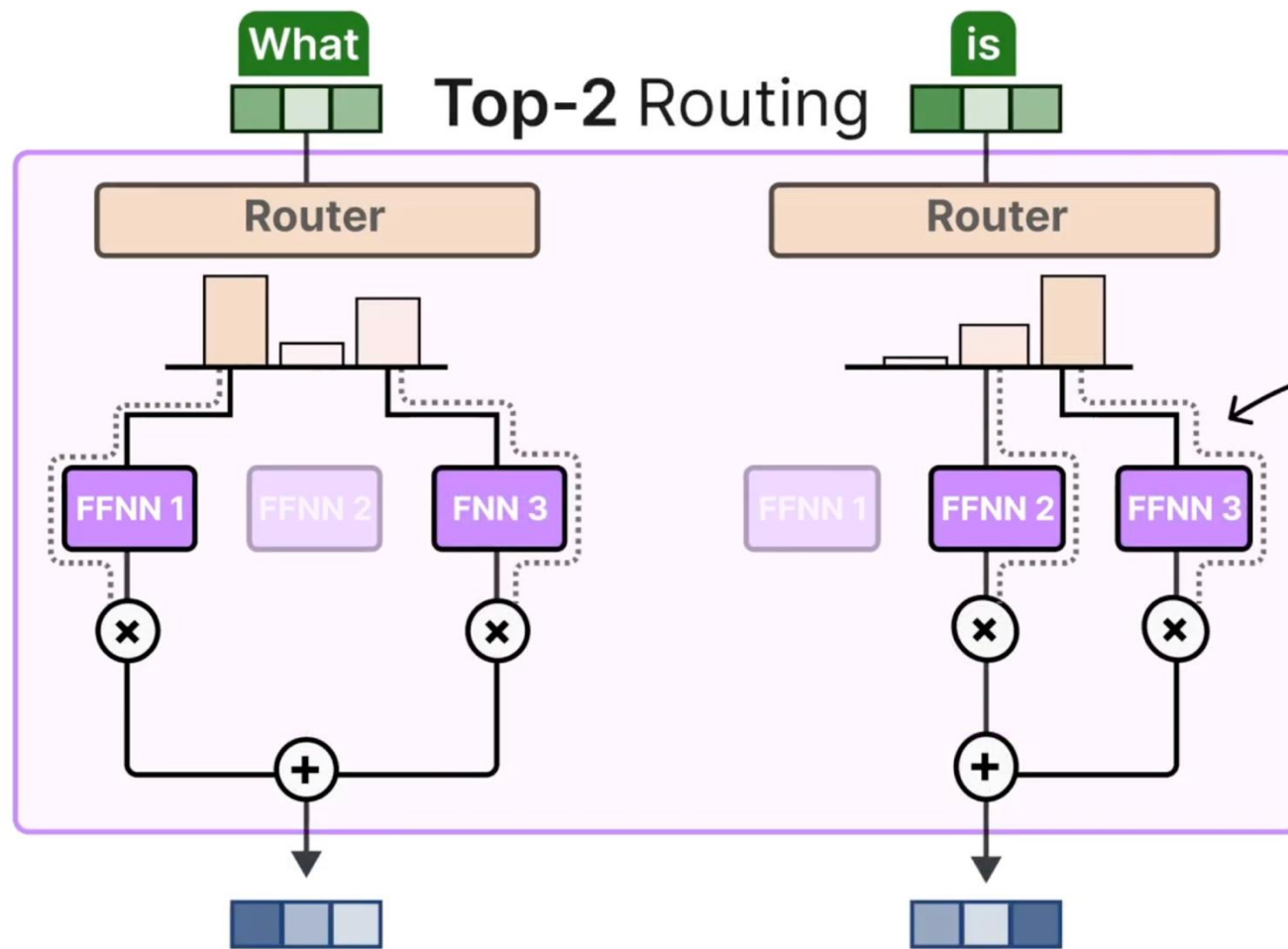


Routing tokens to a few selected experts is also called **Token Choice**.

Keep Top-K: 专家选择



Keep Top-K: 专家选择



... or to more than one expert (**top-k routing**).

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.



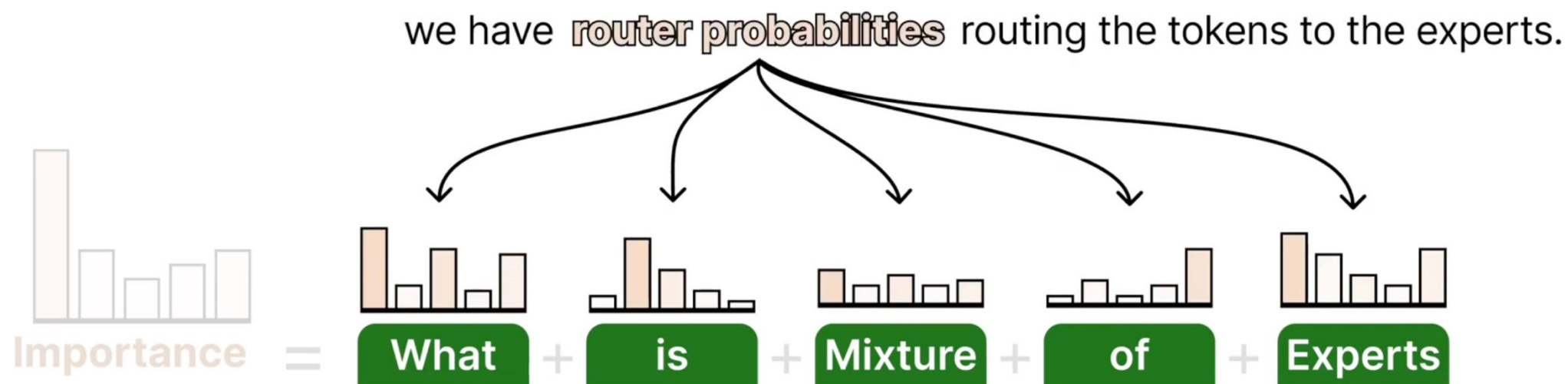
Auxiliary Loss: 辅助损失



Imagine that for each **input token**...

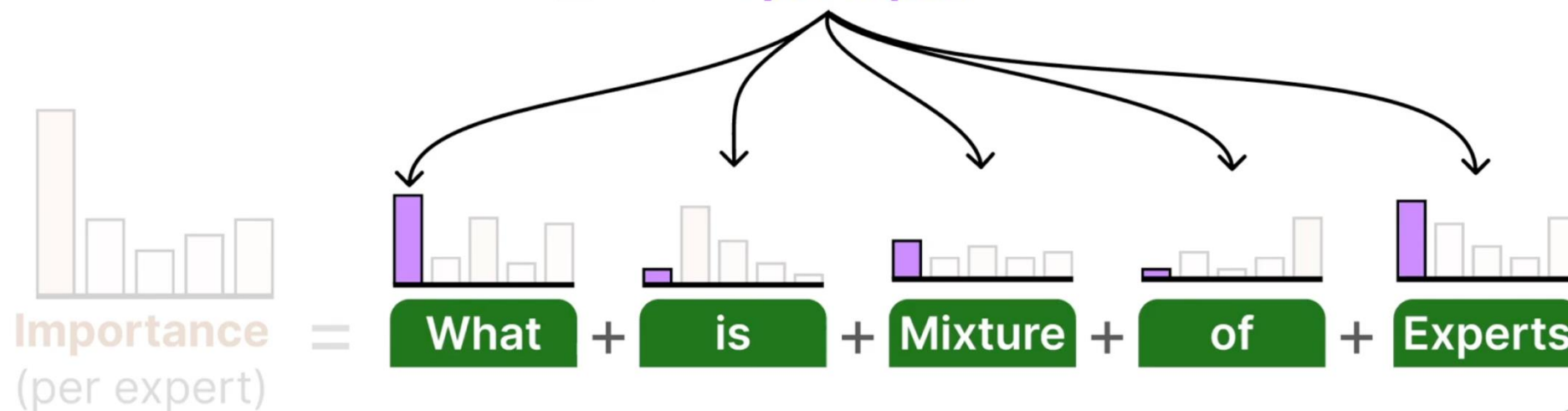


Auxiliary Loss: 辅助损失



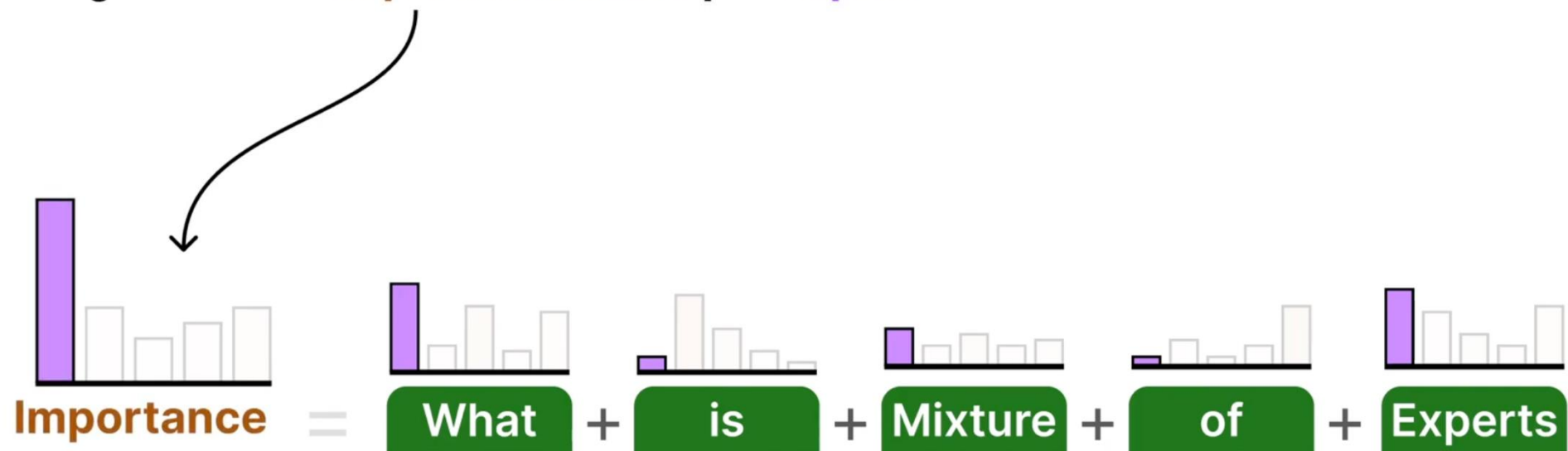
Auxiliary Loss: 辅助损失

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

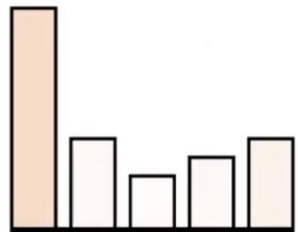
How equal the distribution of importance scores is, can be calculated with

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



Auxiliary Loss: 辅助损失

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


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

An arrow points from the word "high" in the text above to the result "1.11" in the equation.



Auxiliary Loss: 辅助损失

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


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

An arrow points from the word "low" in the text above to the result "0.19".



Auxiliary Loss: 辅助损失

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

$$\text{Auxiliary Loss} = \text{CV} (\text{Importance})^2 * \underbrace{w}_{\substack{\text{(constant)} \\ \text{scaling factor}}}$$




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....


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



Auxiliary Loss: 辅助损失

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


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



Auxiliary Loss: 辅助损失

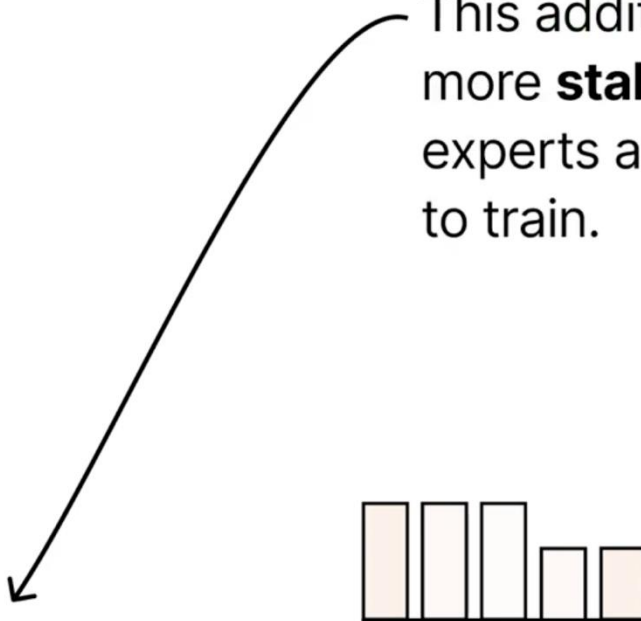
The **Auxiliary loss** is added as a separate loss to optimize during training.


$$\text{Auxiliary Loss} = \text{CV} (\text{Importance})^2 * \underbrace{W_{\text{importance}}}_{\text{(constant) scaling factor}}$$



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.


$$\text{Auxiliary Loss} = \text{CV}(\text{Importance})^2 * \underbrace{W_{\text{importance}}}_{\text{(constant) scaling factor}}$$

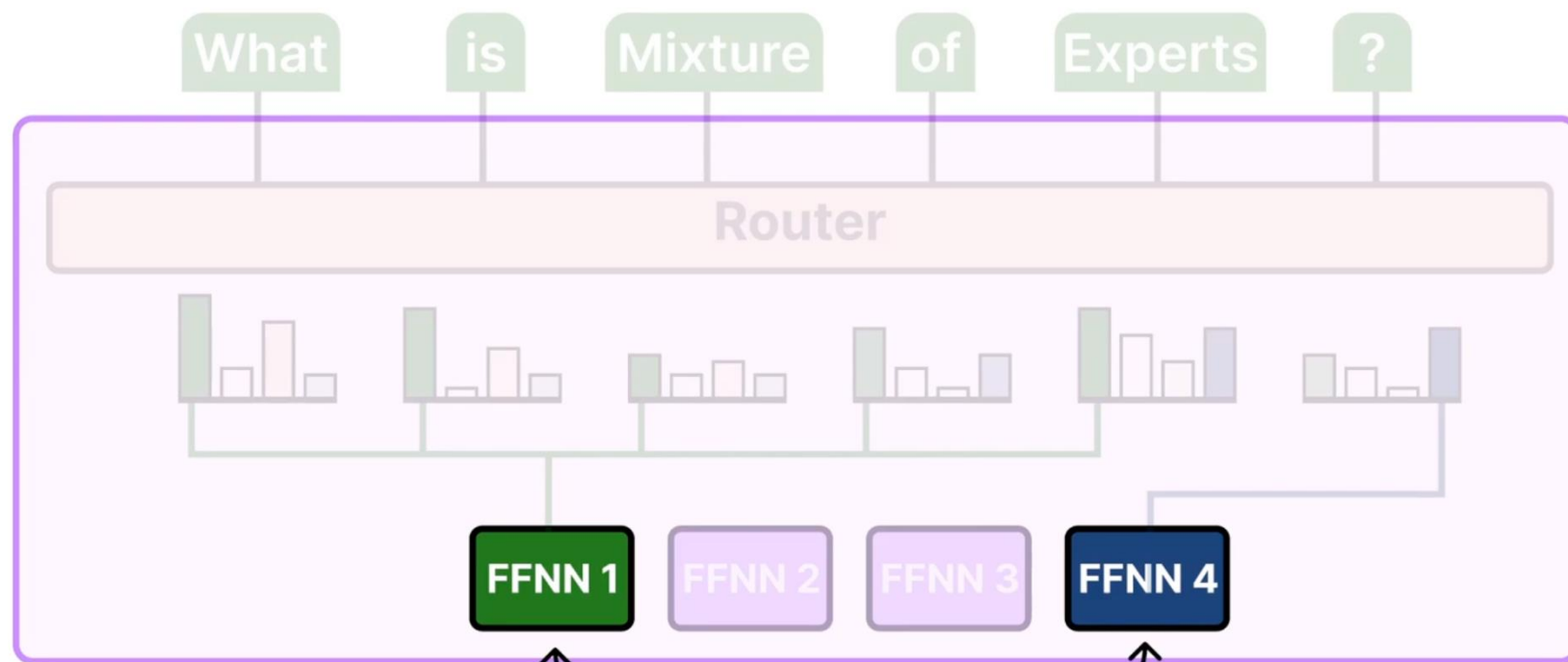


07

Expert Capacity: 专家容量



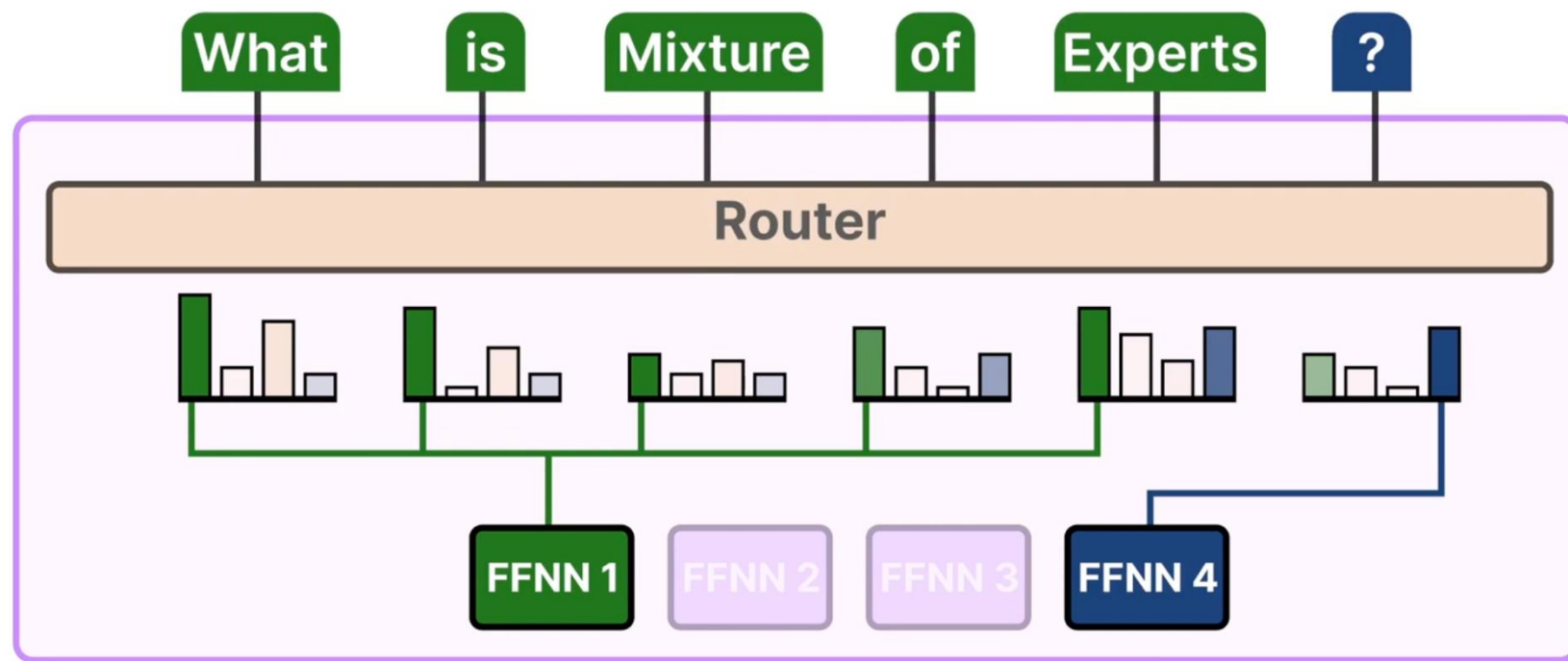
Expert Capacity: 专家容量



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



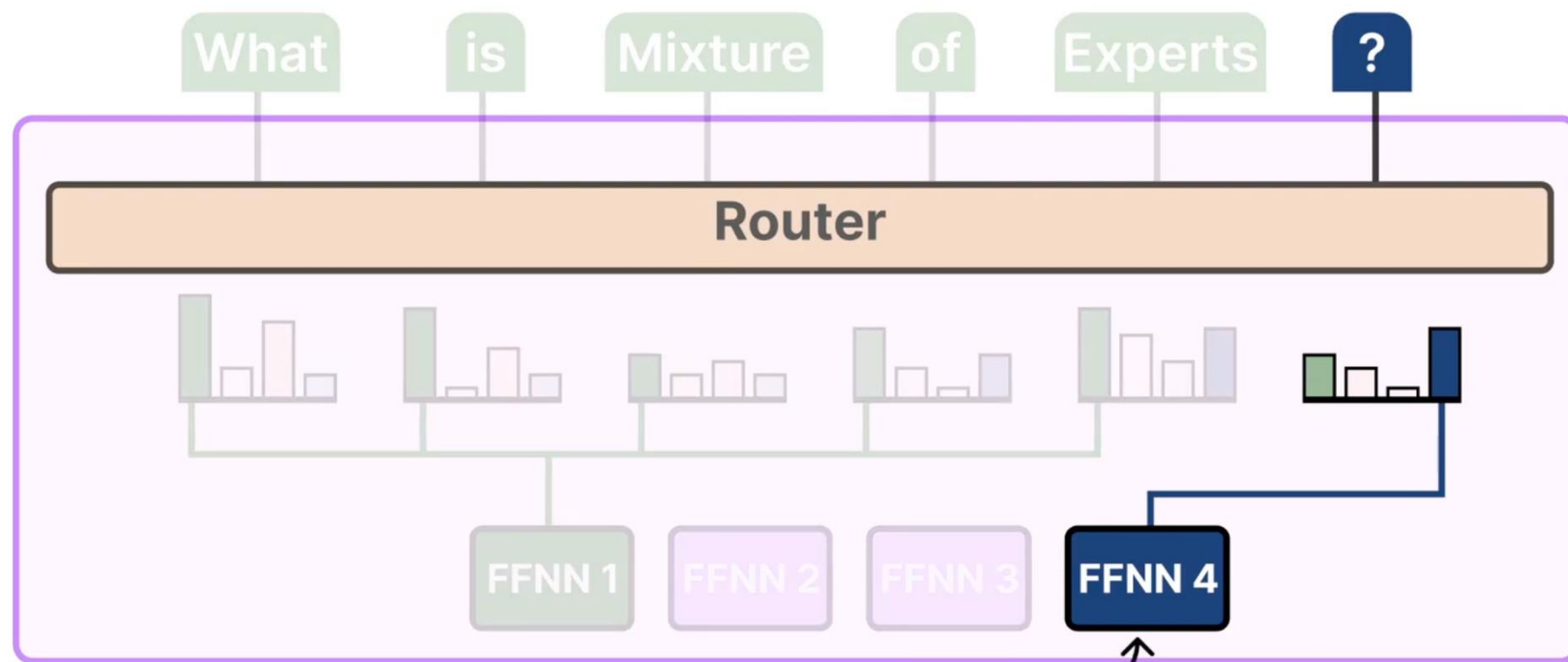
Expert Capacity: 专家容量



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



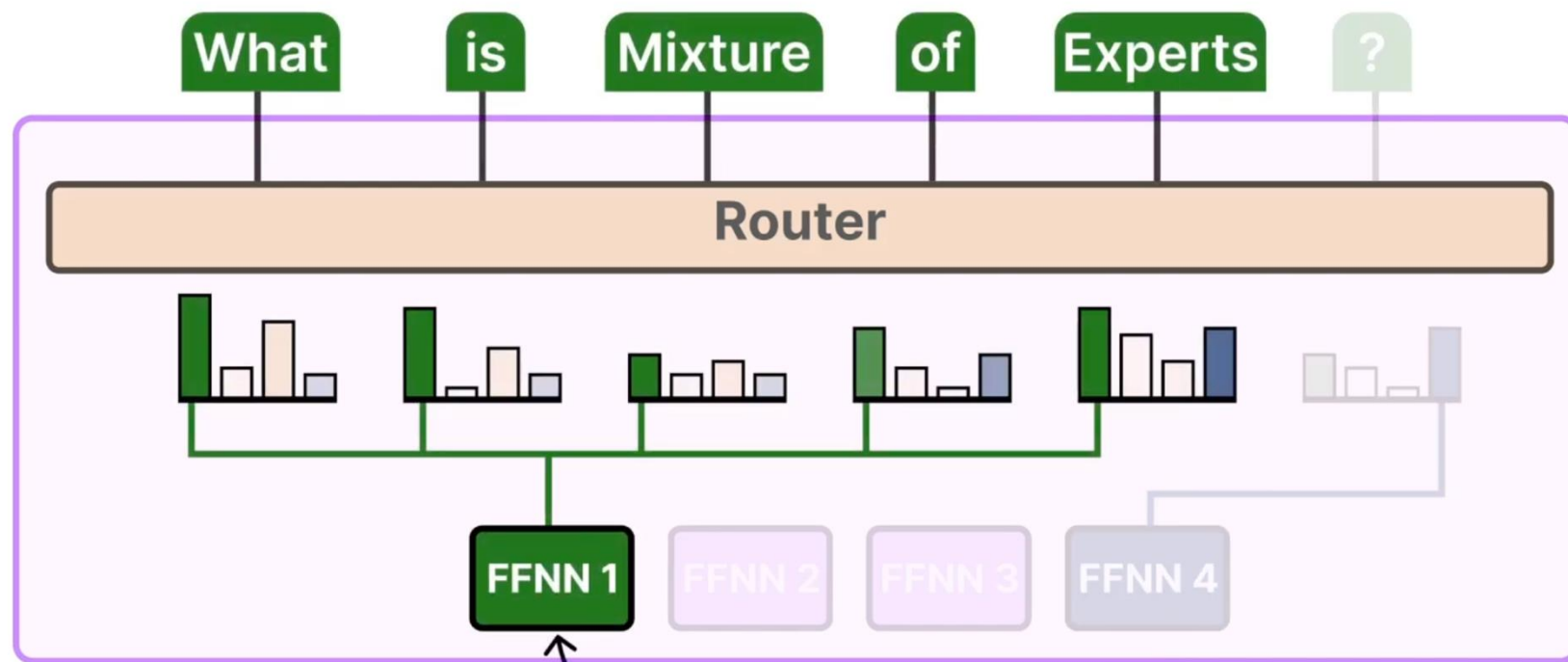
Expert Capacity: 专家容量



Note that **expert 4** has received only **one token** of the input...



Expert Capacity: 专家容量

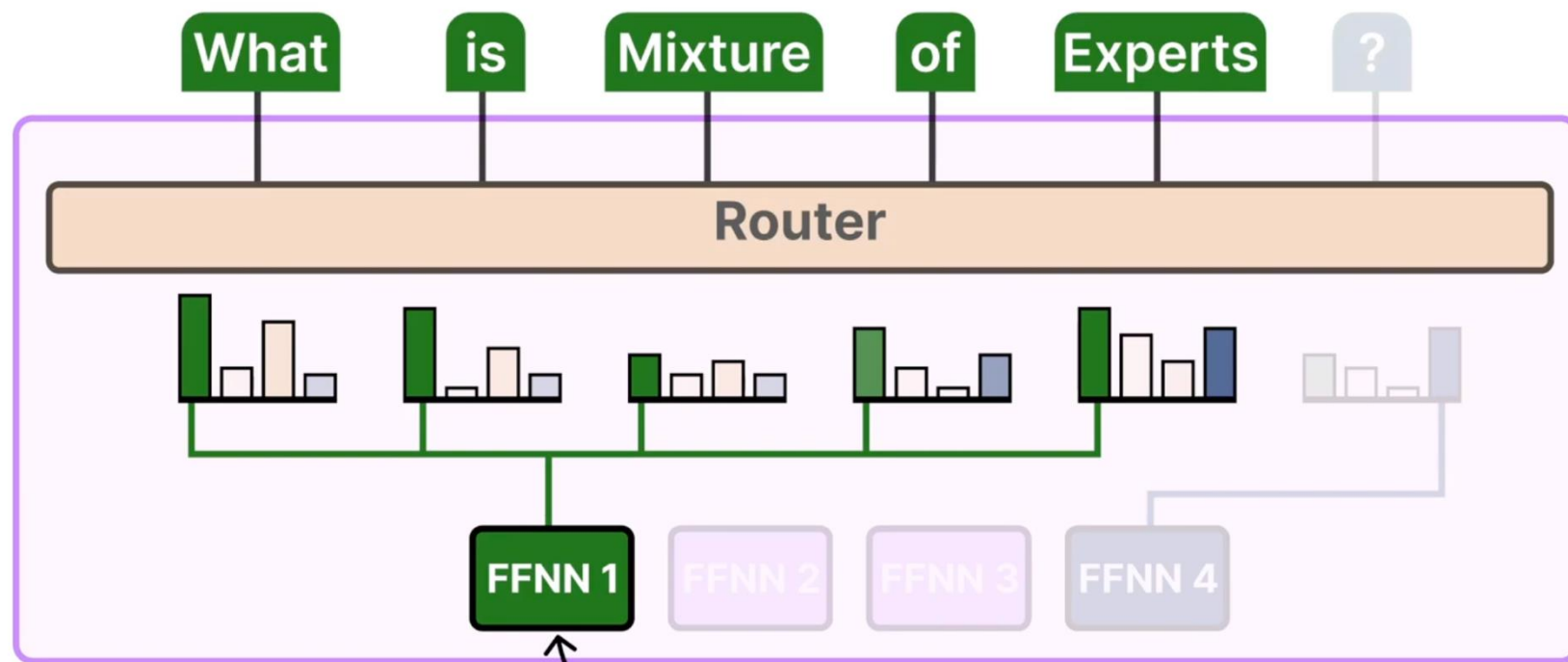


... whereas **expert 1** has received **all other tokens**.

What is Mixture of Experts



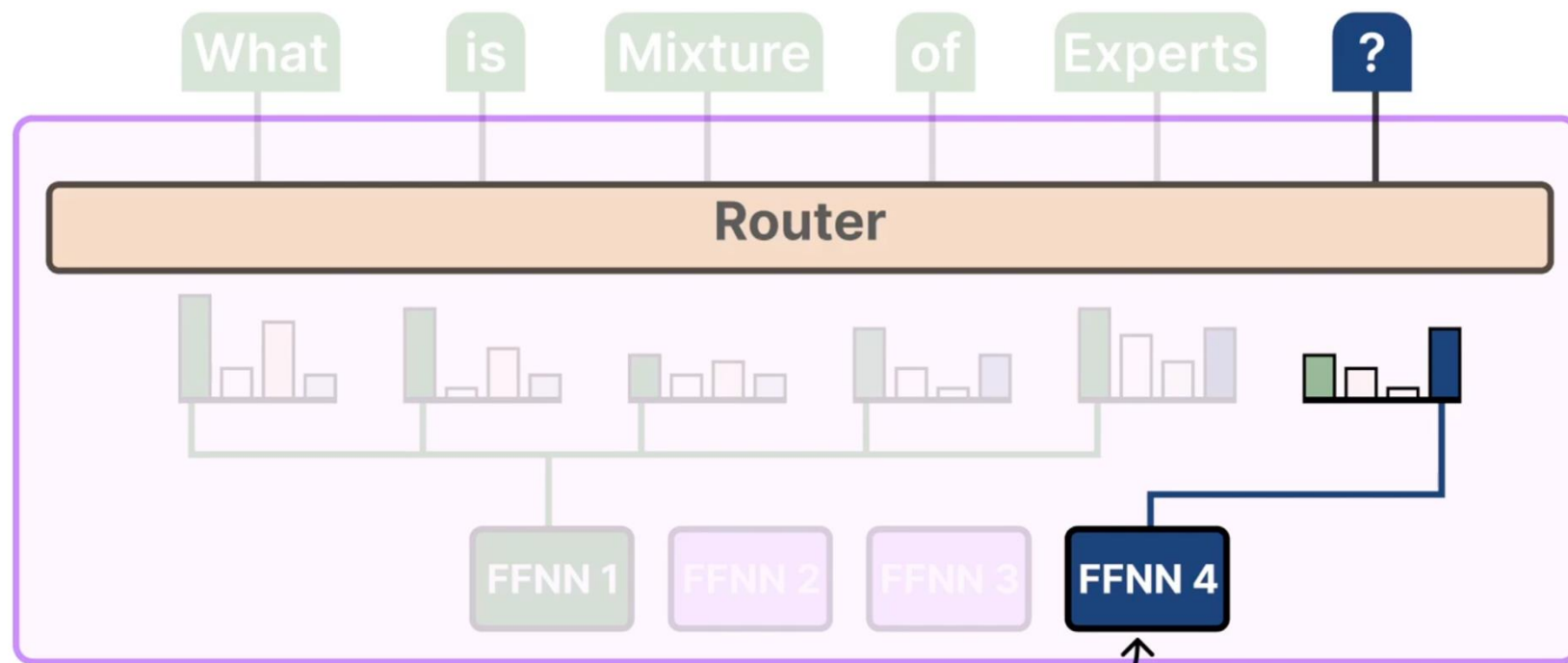
Expert Capacity: 专家容量



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



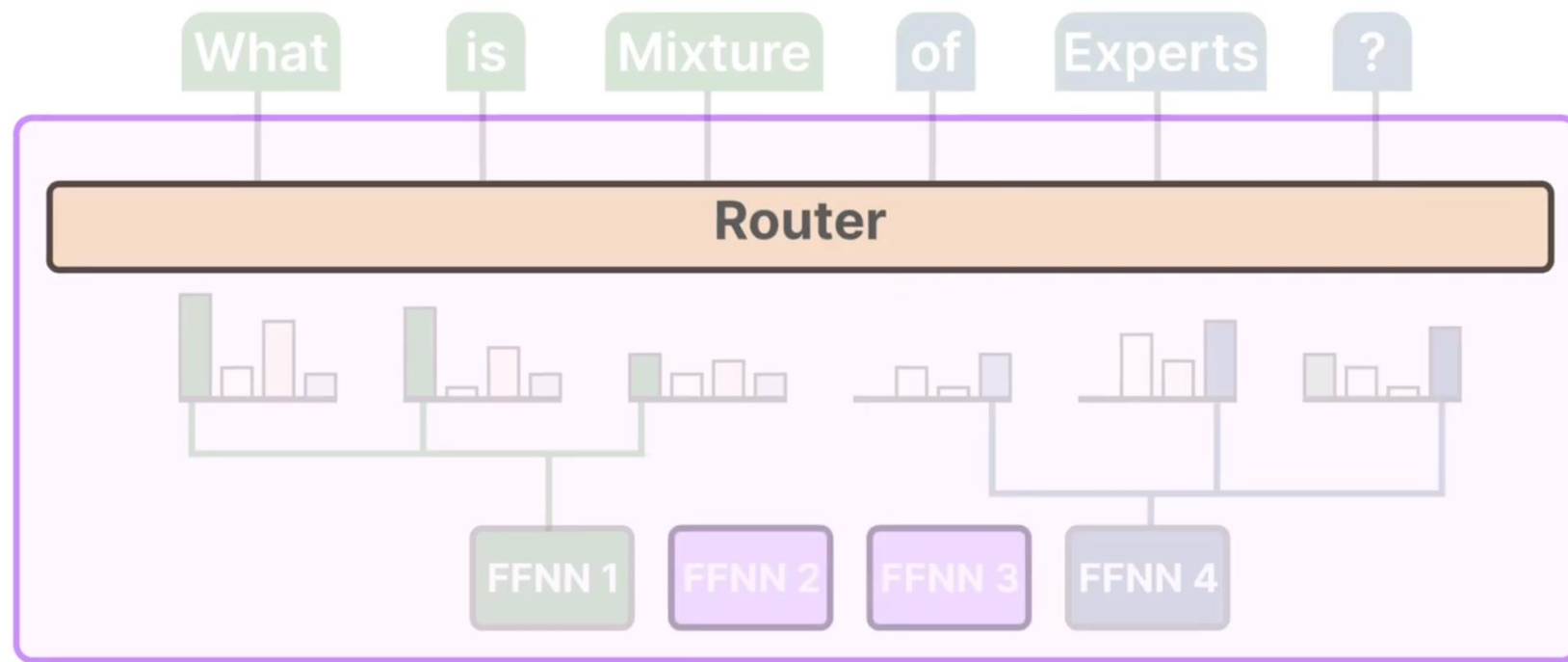
Expert Capacity: 专家容量



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



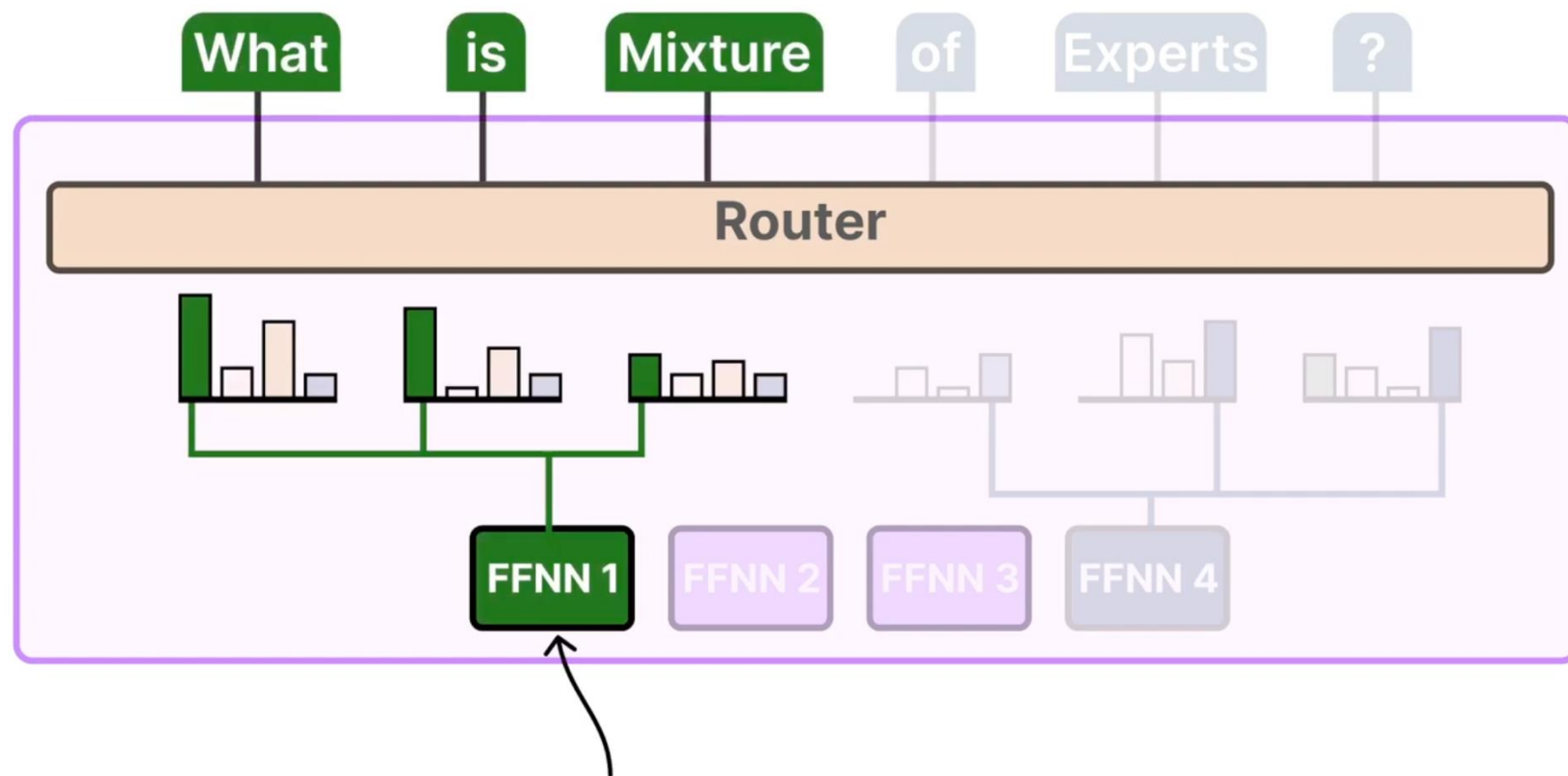
Expert Capacity: 专家容量



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



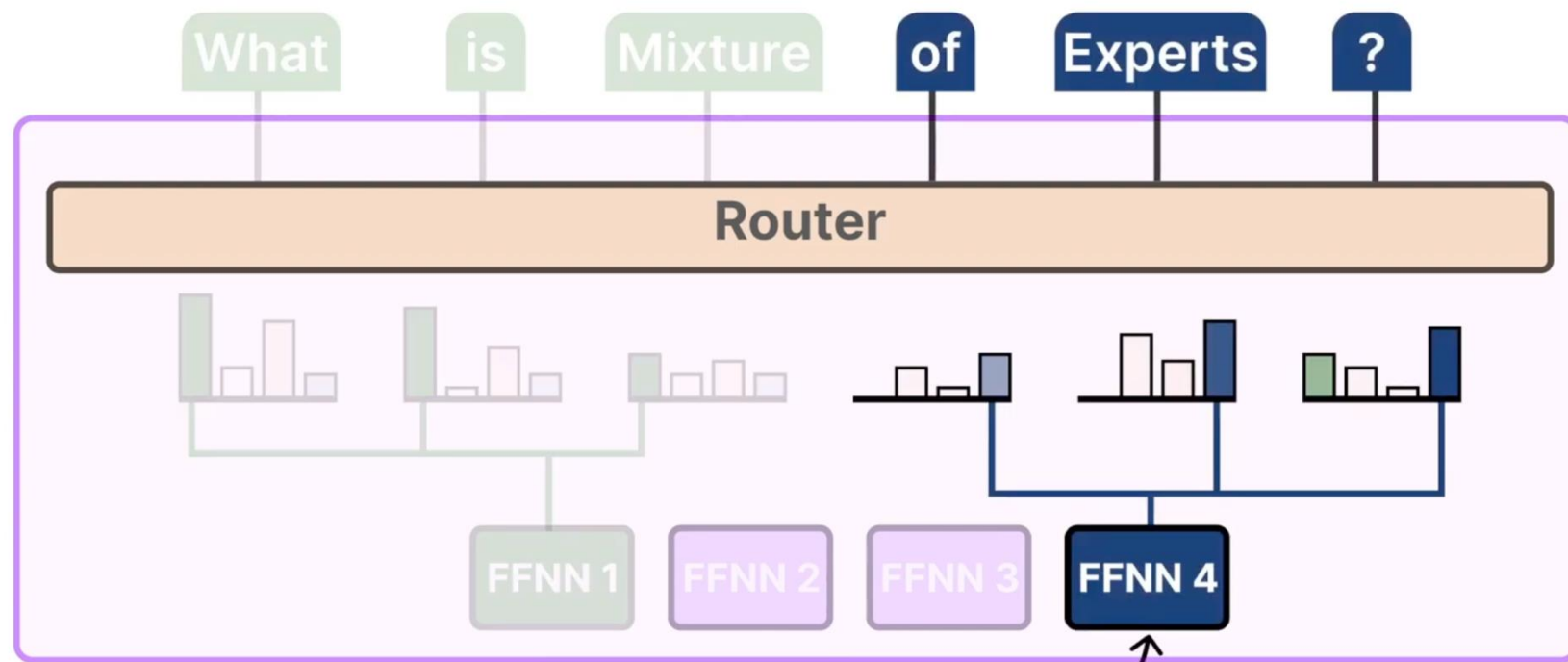
Expert Capacity: 专家容量



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



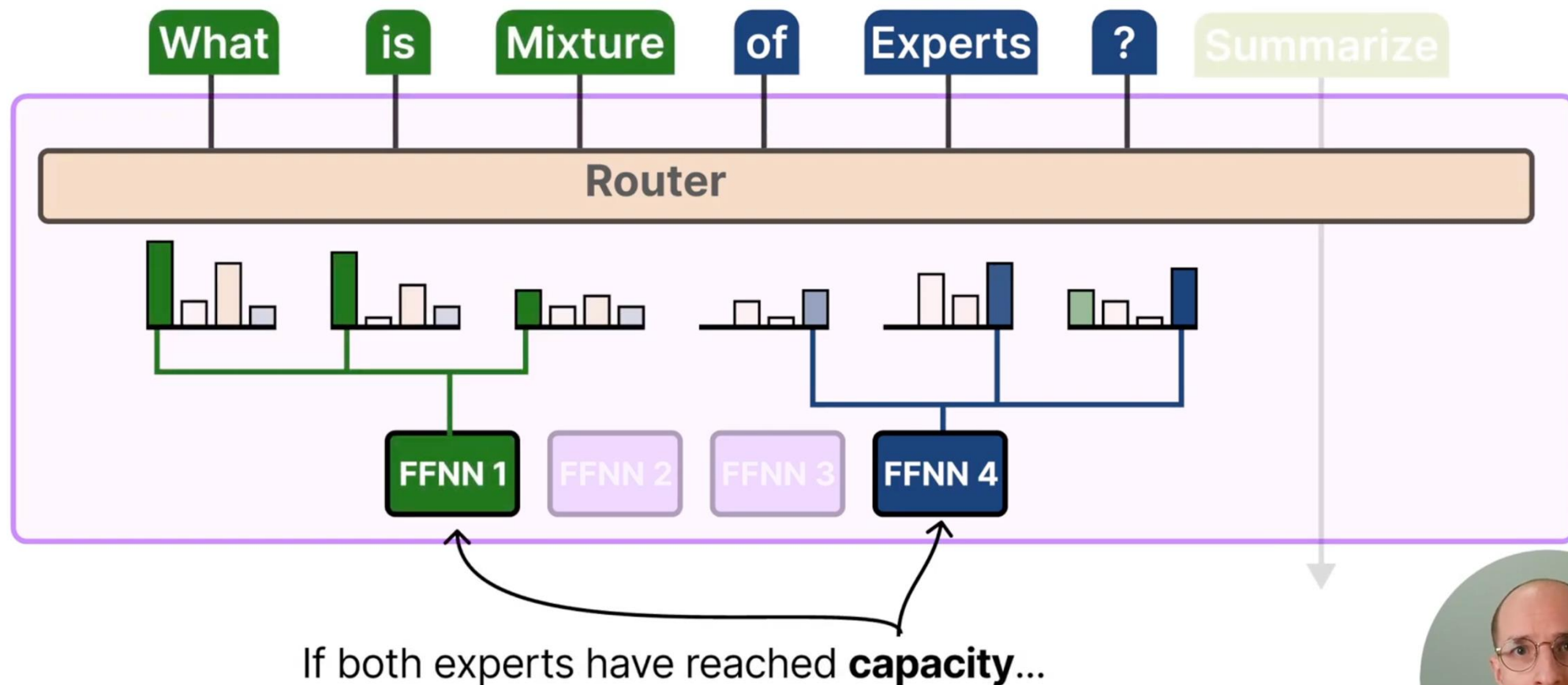
Expert Capacity: 专家容量



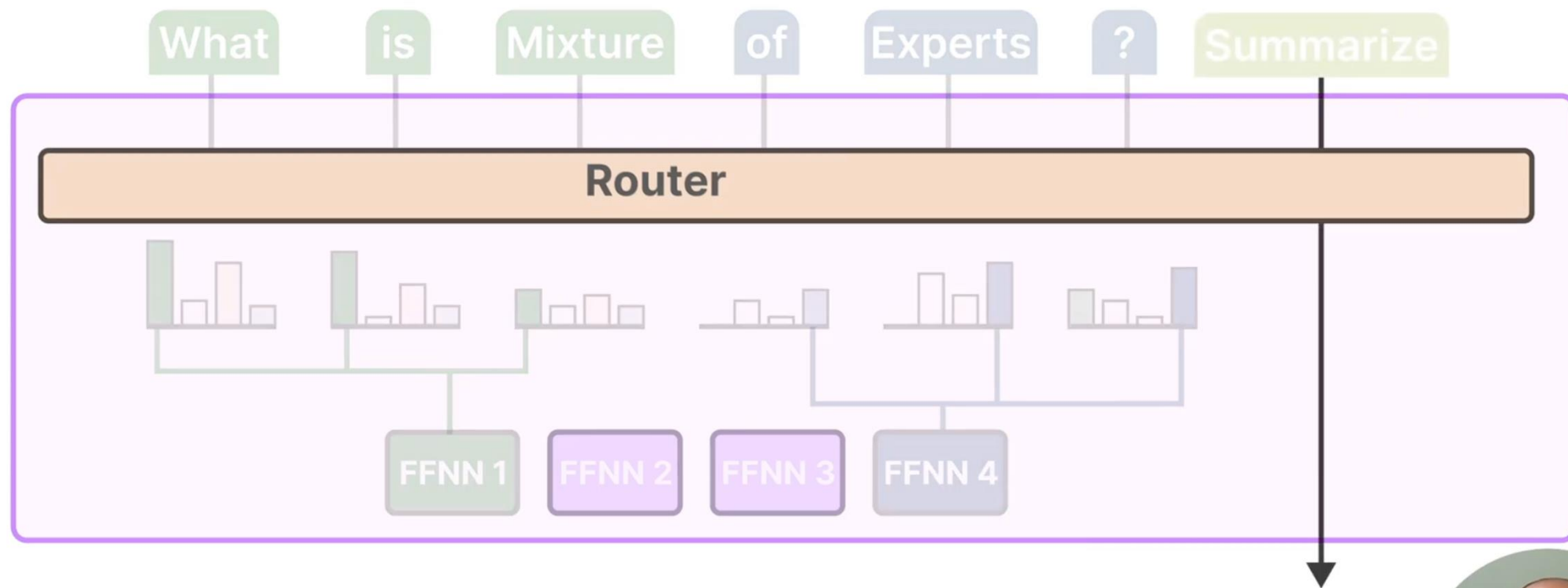
Tokens that exceed this threshold are routed to the next most likely expert.



Expert Capacity: 专家容量



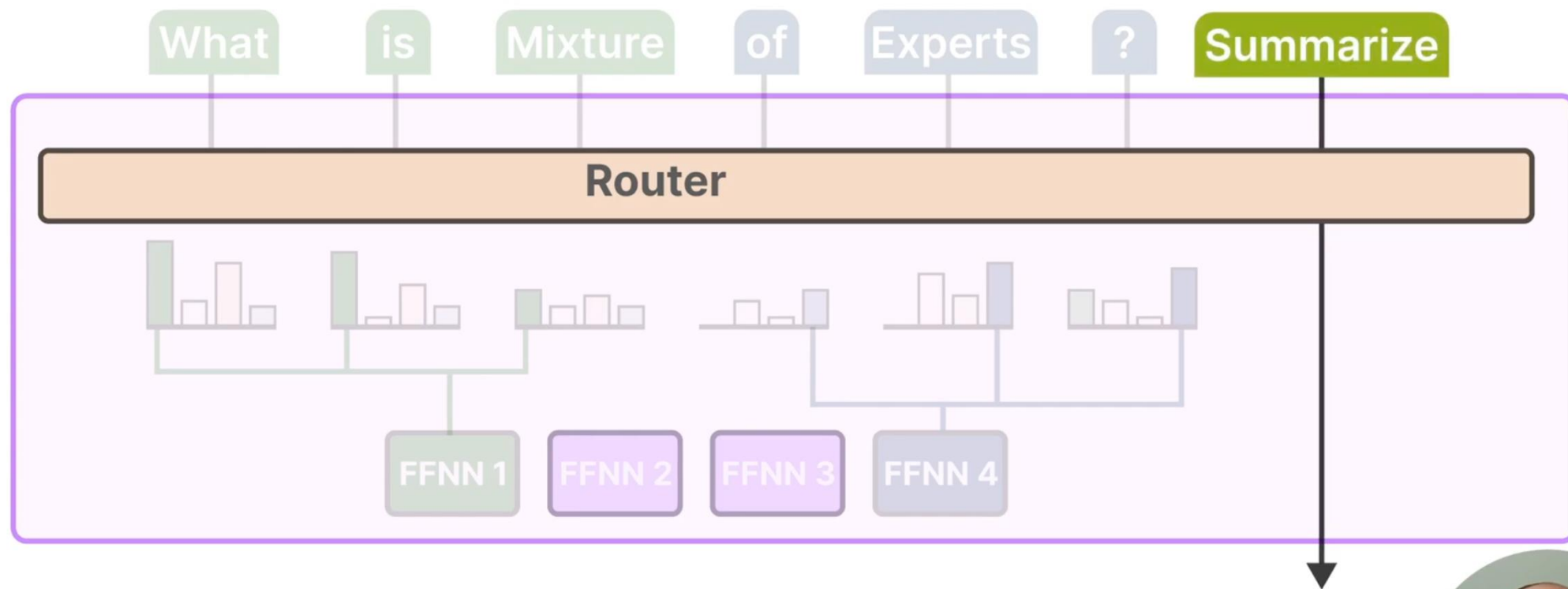
Expert Capacity: 专家容量



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



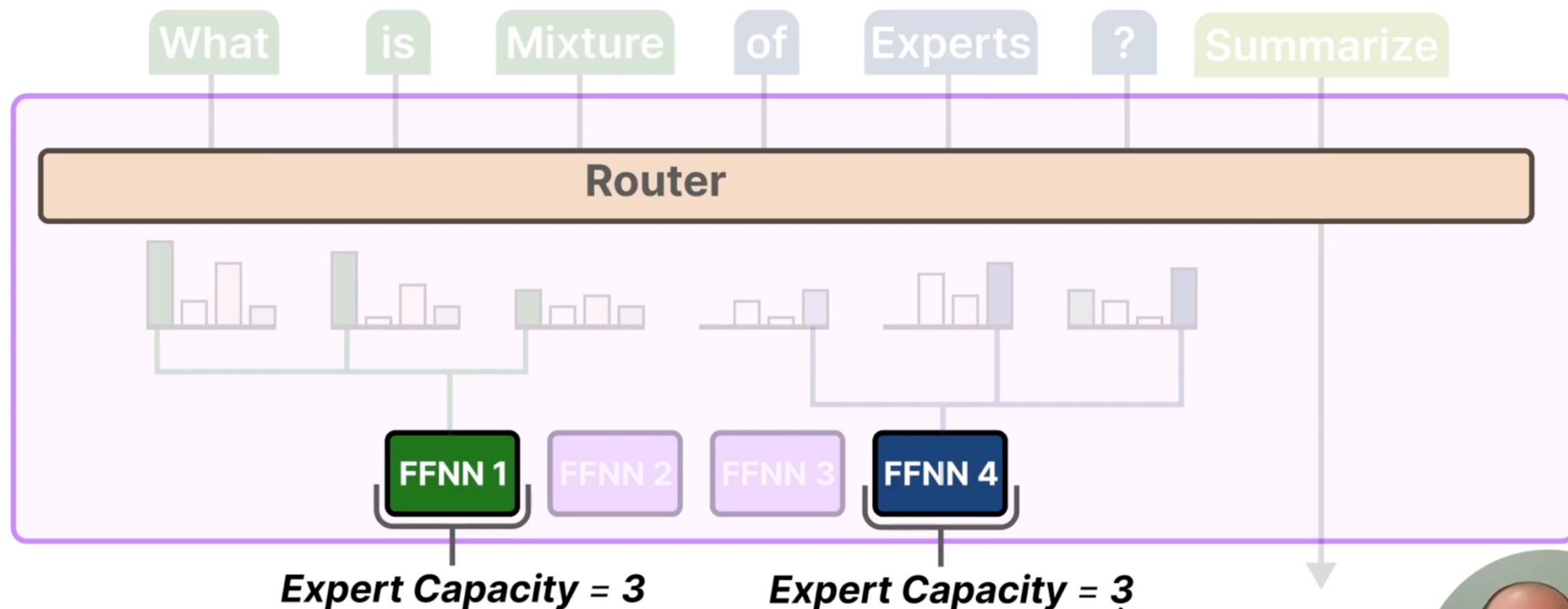
Expert Capacity: 专家容量



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



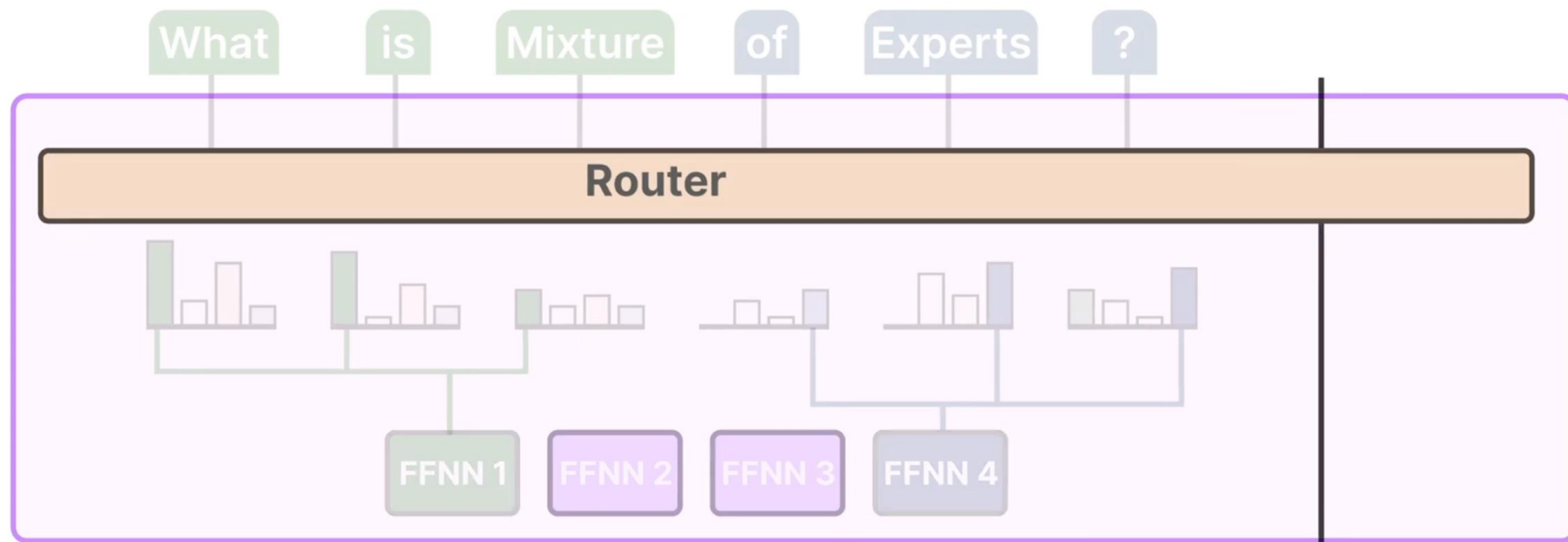
Expert Capacity: 专家容量



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>



ZOMI

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/x39hqf8xn1cUlnxEIM0_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-llm-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>

