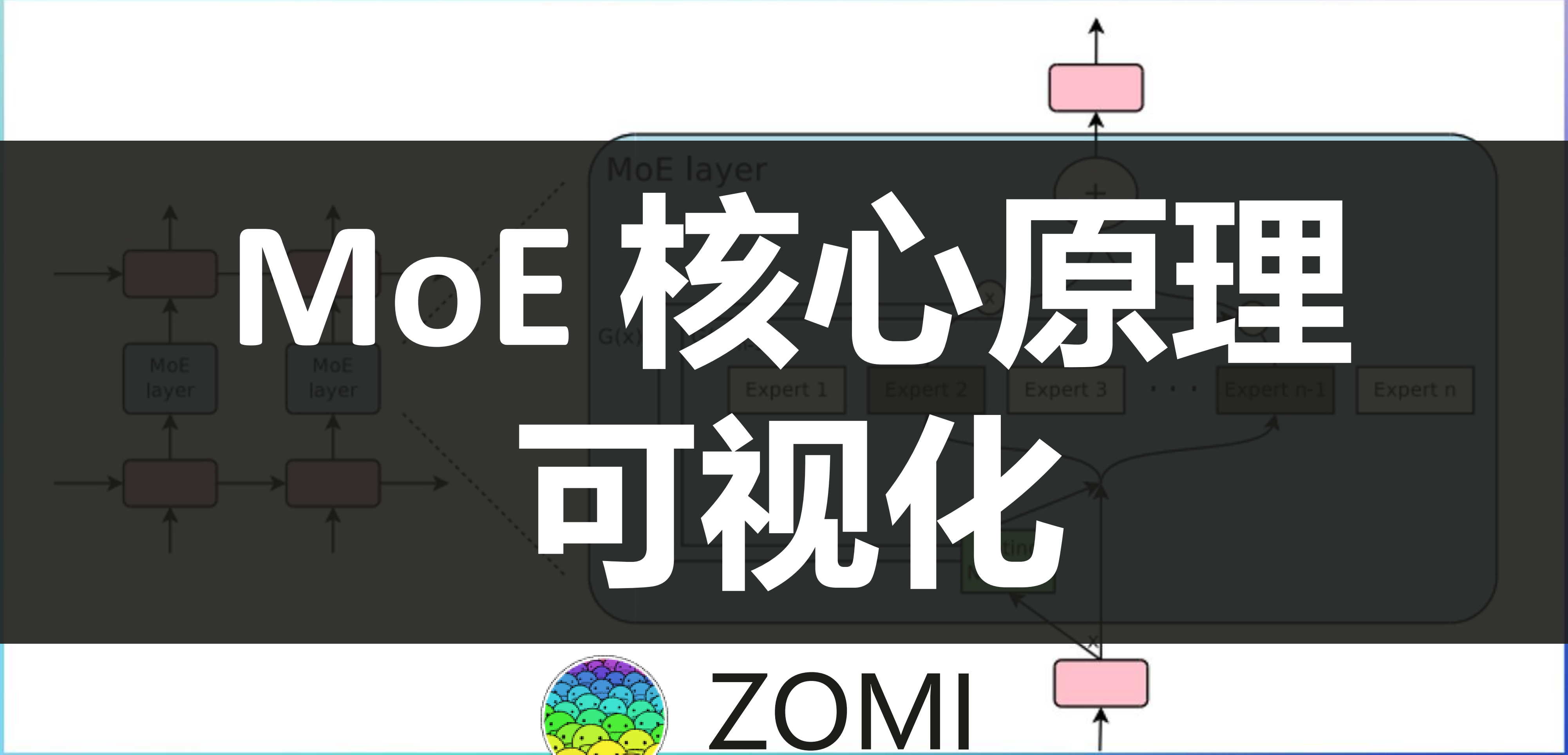
Mixture of Experts (MoE)

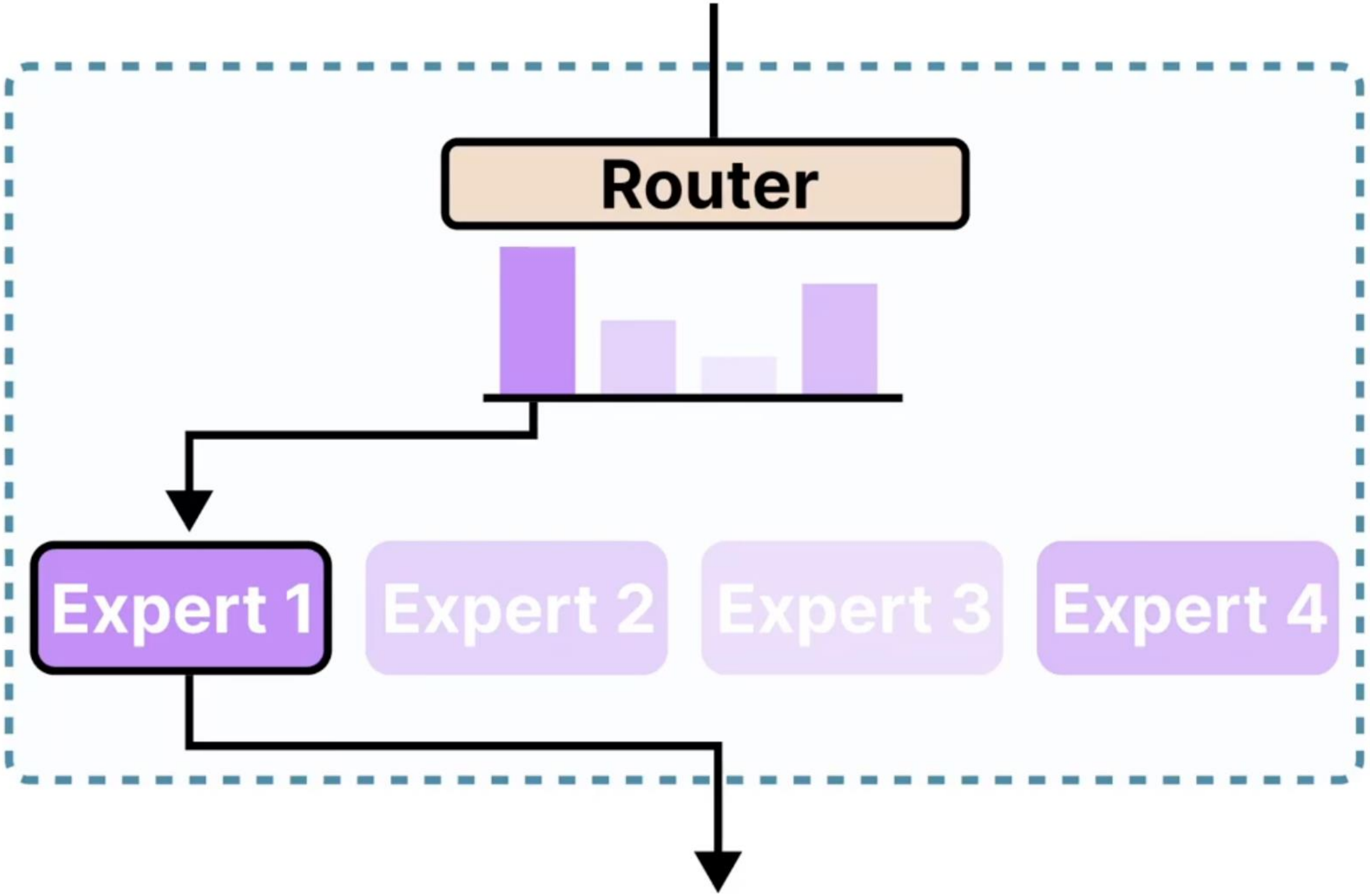


MoE 核心原理
可视化

ZOMI

MoE 核心原理可视化

https://www.youtube.com/watch?v=sOPDGQjFcuM



Contents

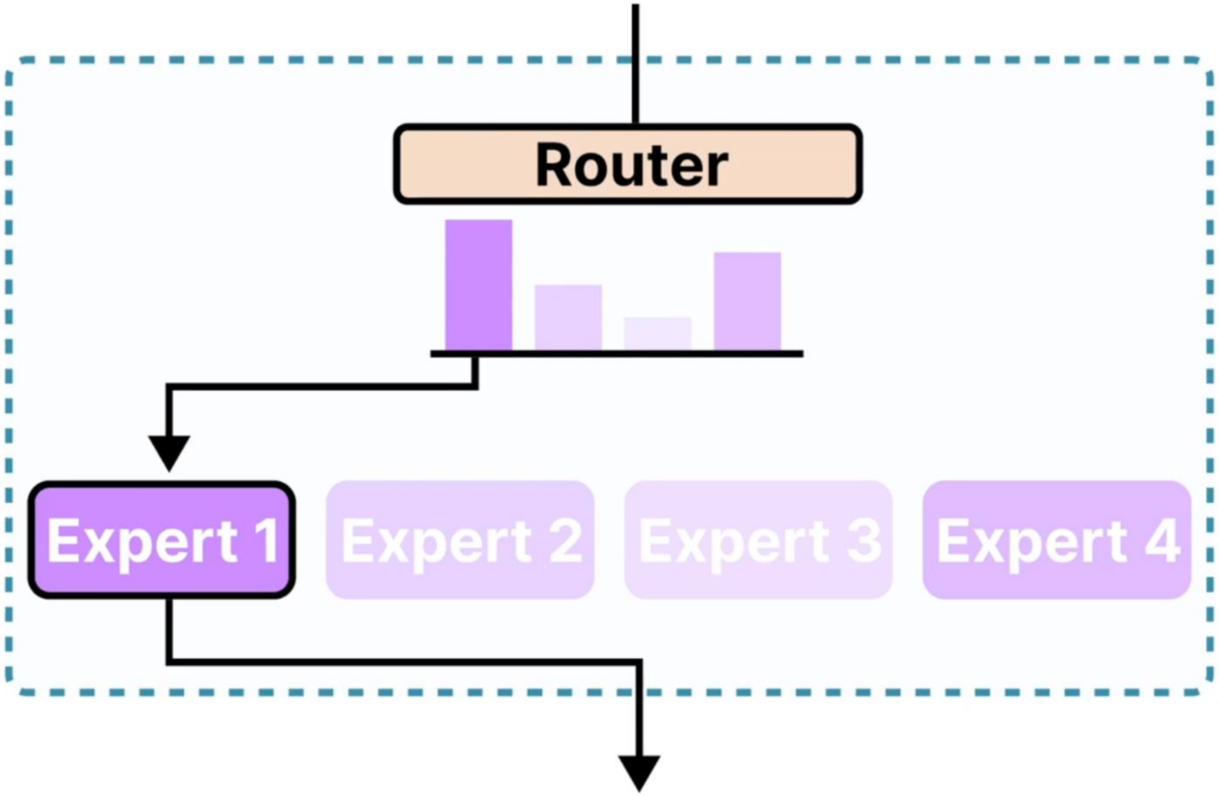
视频目录大纲

- MoE 混合专家
  - MoE 基本介绍 (4)
  - MoE 前世今生 (5)
  - MoE 核心工作 (6)
  - MoE 架构原理
    - 门控网络 Gating Network 到路由 Switch Routing
    - 平衡专家利用率（Balancing Expert Utilization）
    - 高效训练：数据、模型、专家并行
  - 大模型遇上 MoE (4)
  - 手撕 MoE 代码 (5)
  - MoE 的未来 (2)

01

Introduction：MOE 基本原理

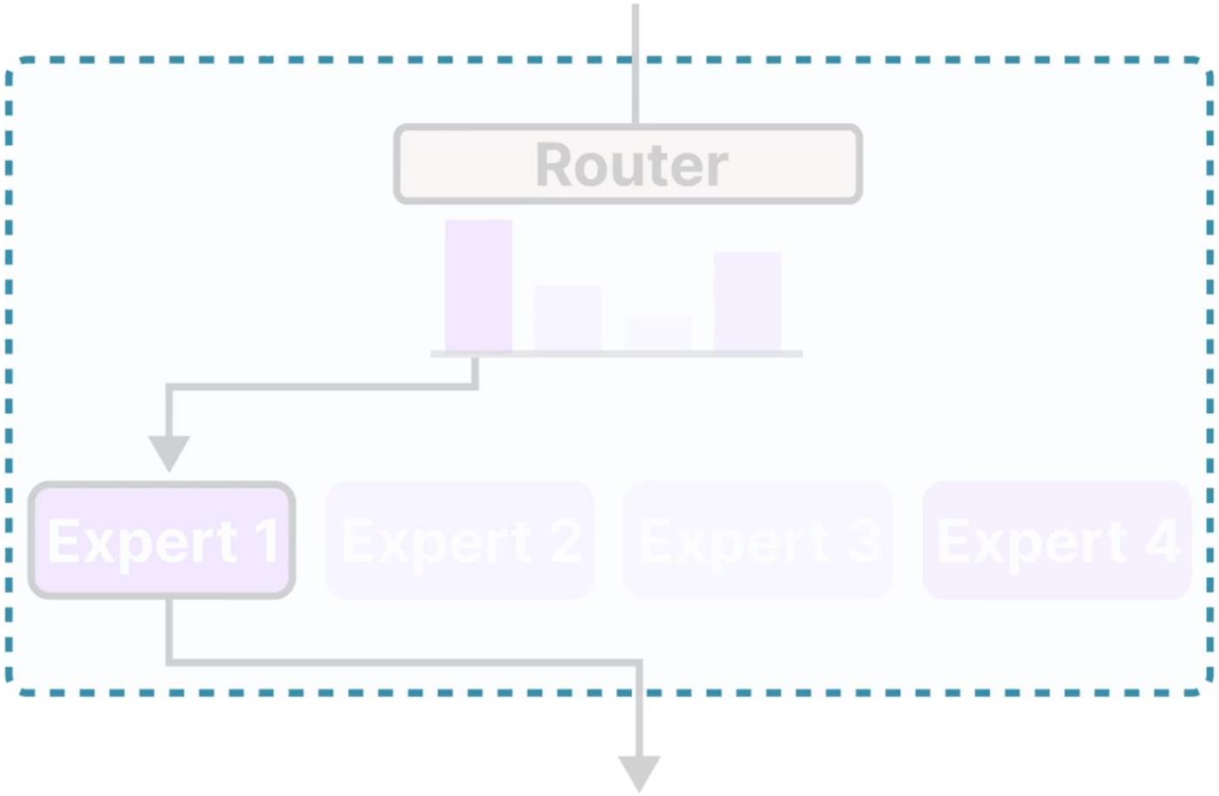
Introduction： MOE 基本原理



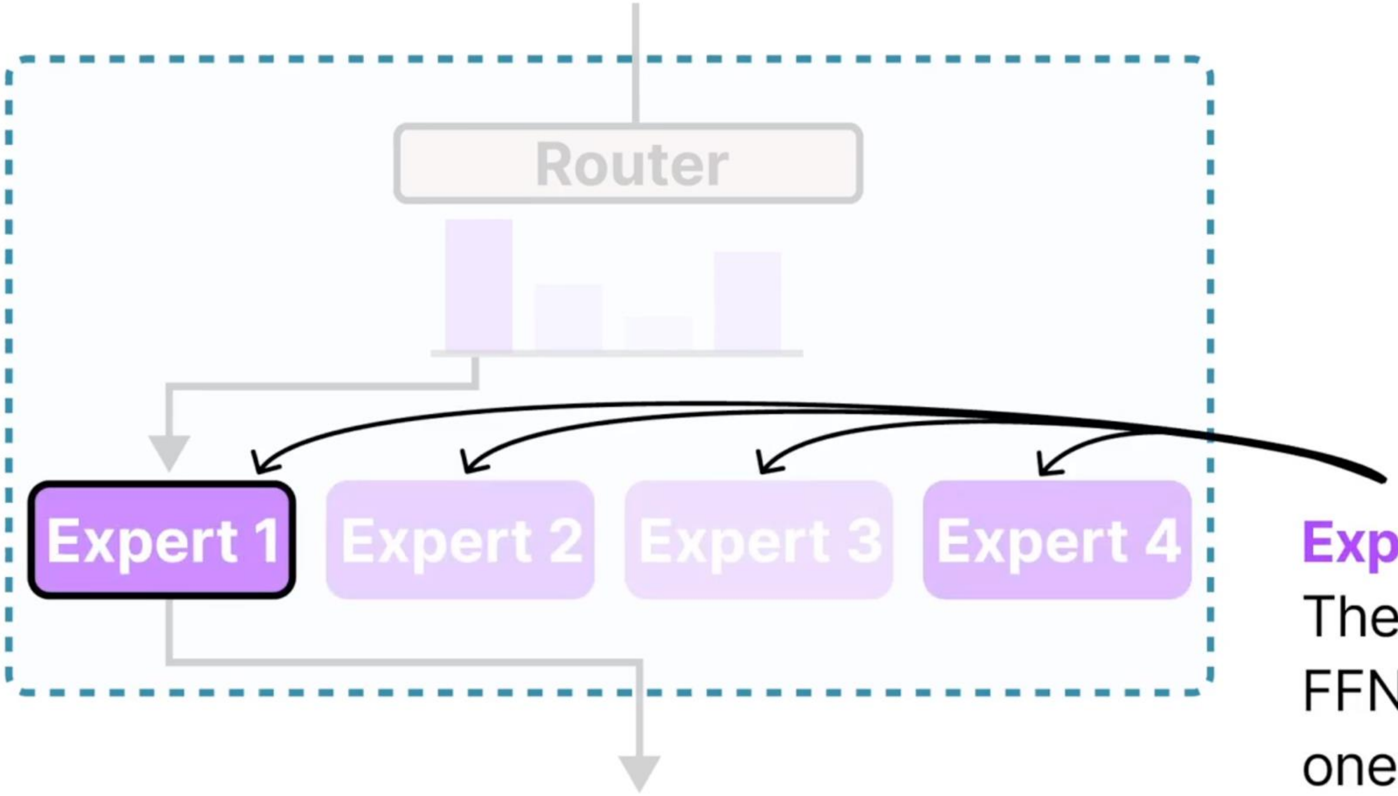
Mixture of Experts (*MoE*) is a technique that uses different sub-models ("**experts**") to improve the quality of **LLMs**.

Introduction： MOE 基本原理

Two main components define a **MoE**:



Introduction： MOE 基本原理

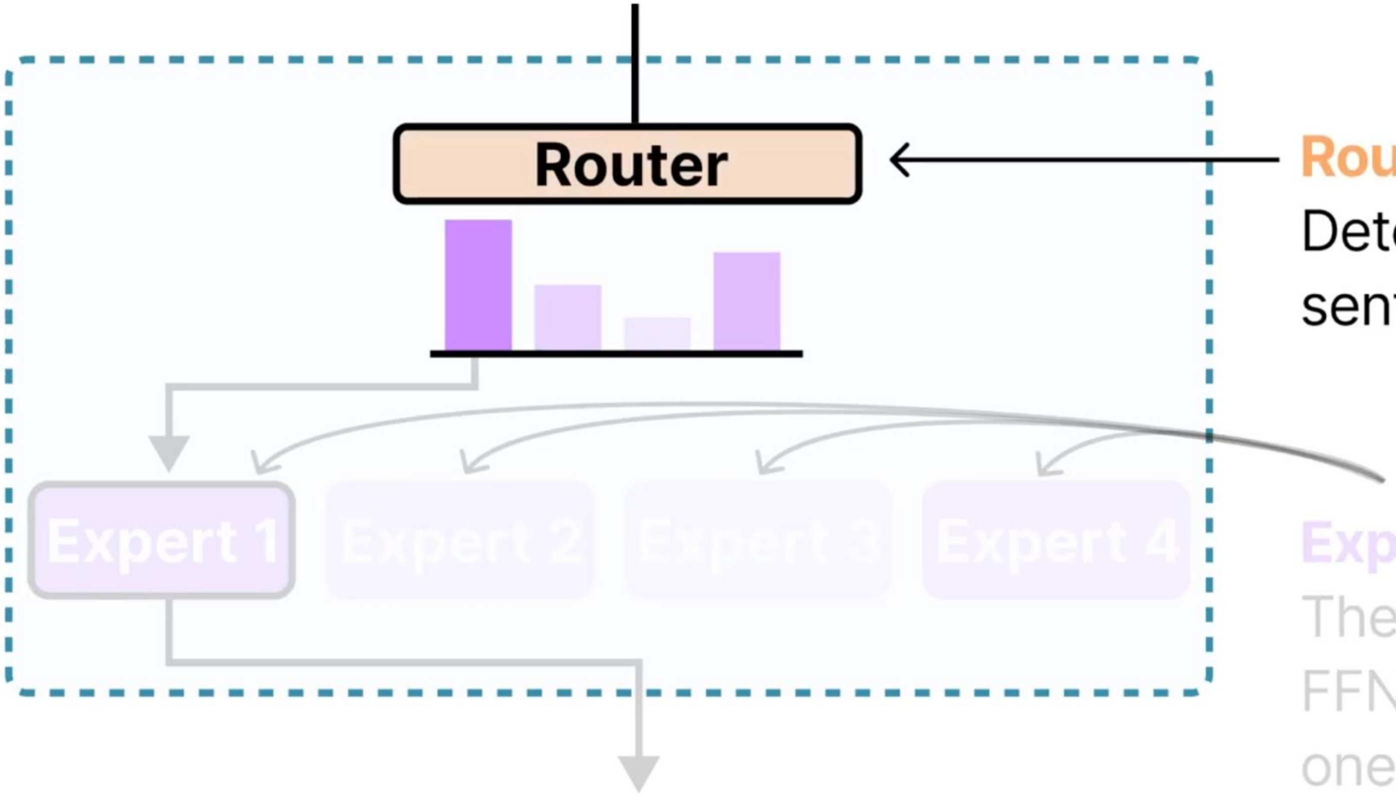


Two main components define a **MoE**:

Experts

These "experts" are FFNNs and at least one can be activated

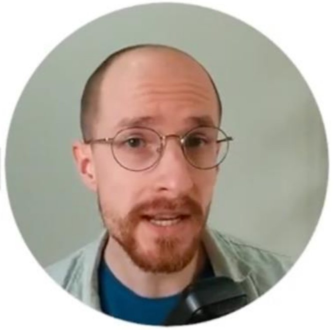
Introduction： MOE 基本原理



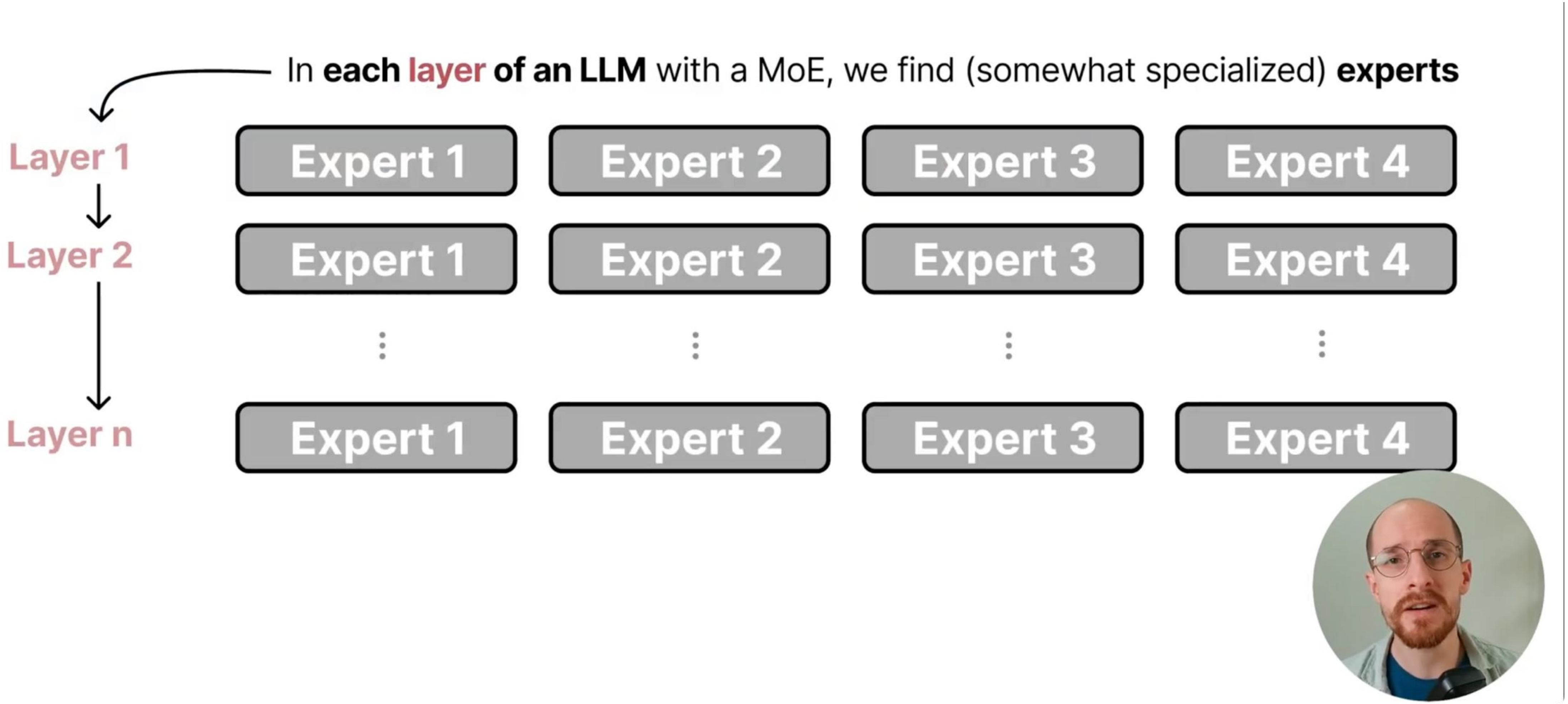
Two main components define a **MoE**:

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

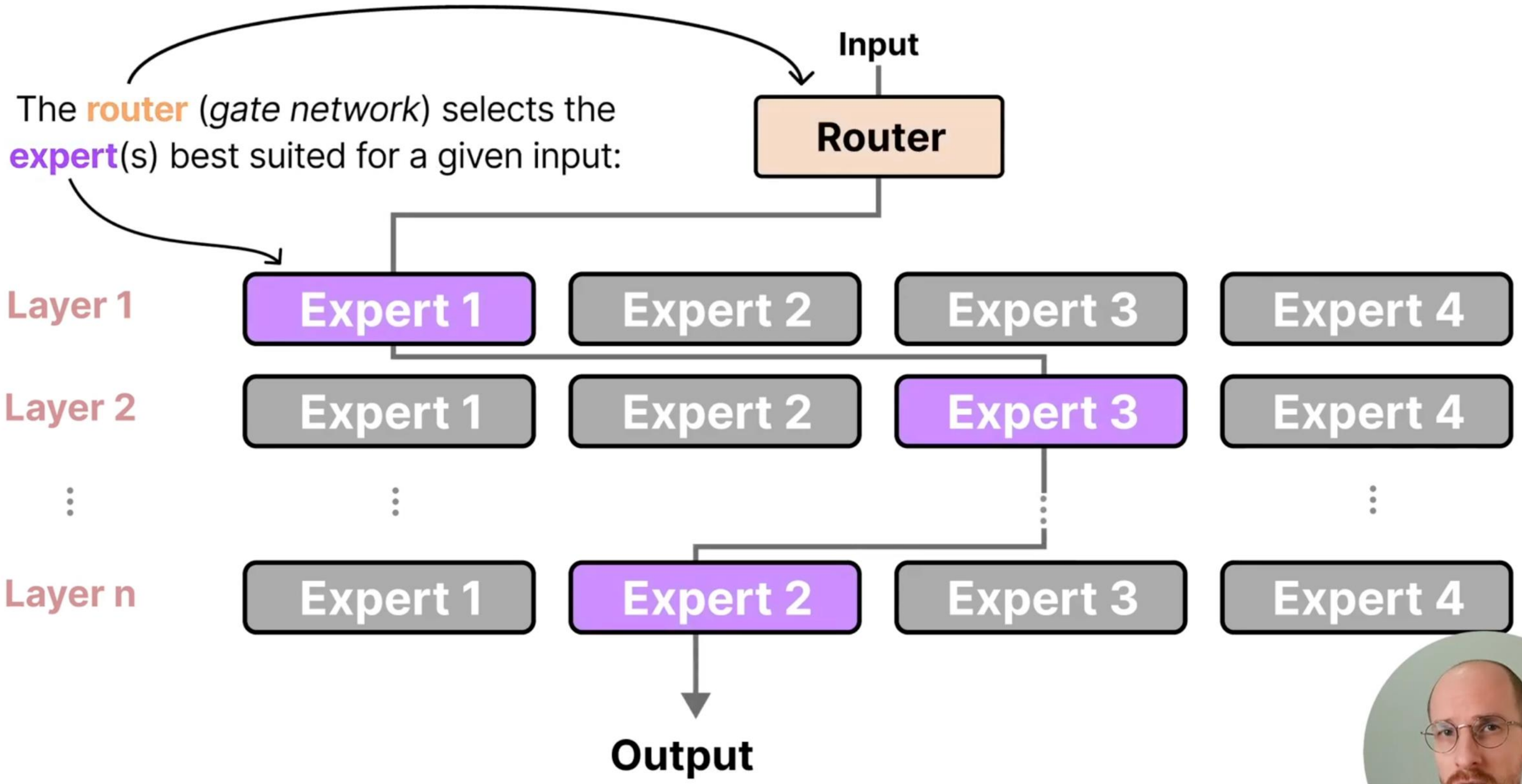
Experts
These “experts” are FFNNs and at least one can be activated

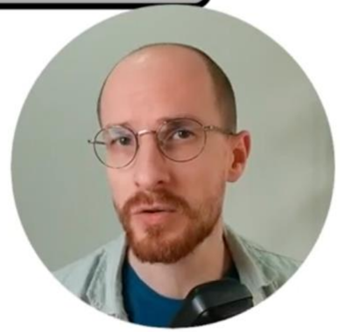


Introduction： MOE 基本原理



Introduction： MOE 基本原理


Input
Router
The router (gate network) selects the
expert(s) best suited for a given input:
Layer 1
Expert 1
Expert 2
Expert 3
Expert 4
Layer 2
Expert 1
Expert 2
Expert 3
Expert 4
Layer n
Expert 1
Expert 2
Expert 3
Expert 4
Output

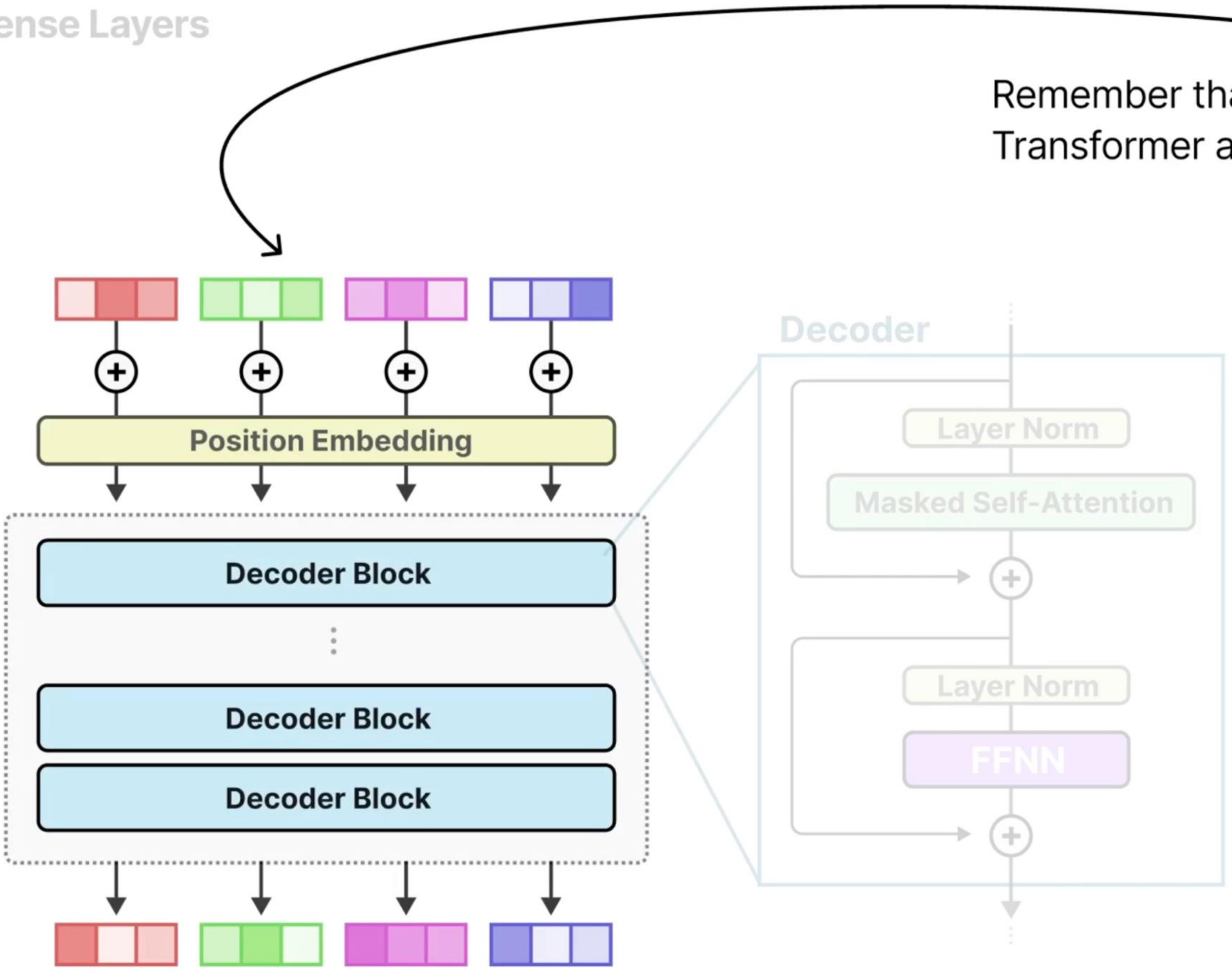


Introduction： MOE 基本原理

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

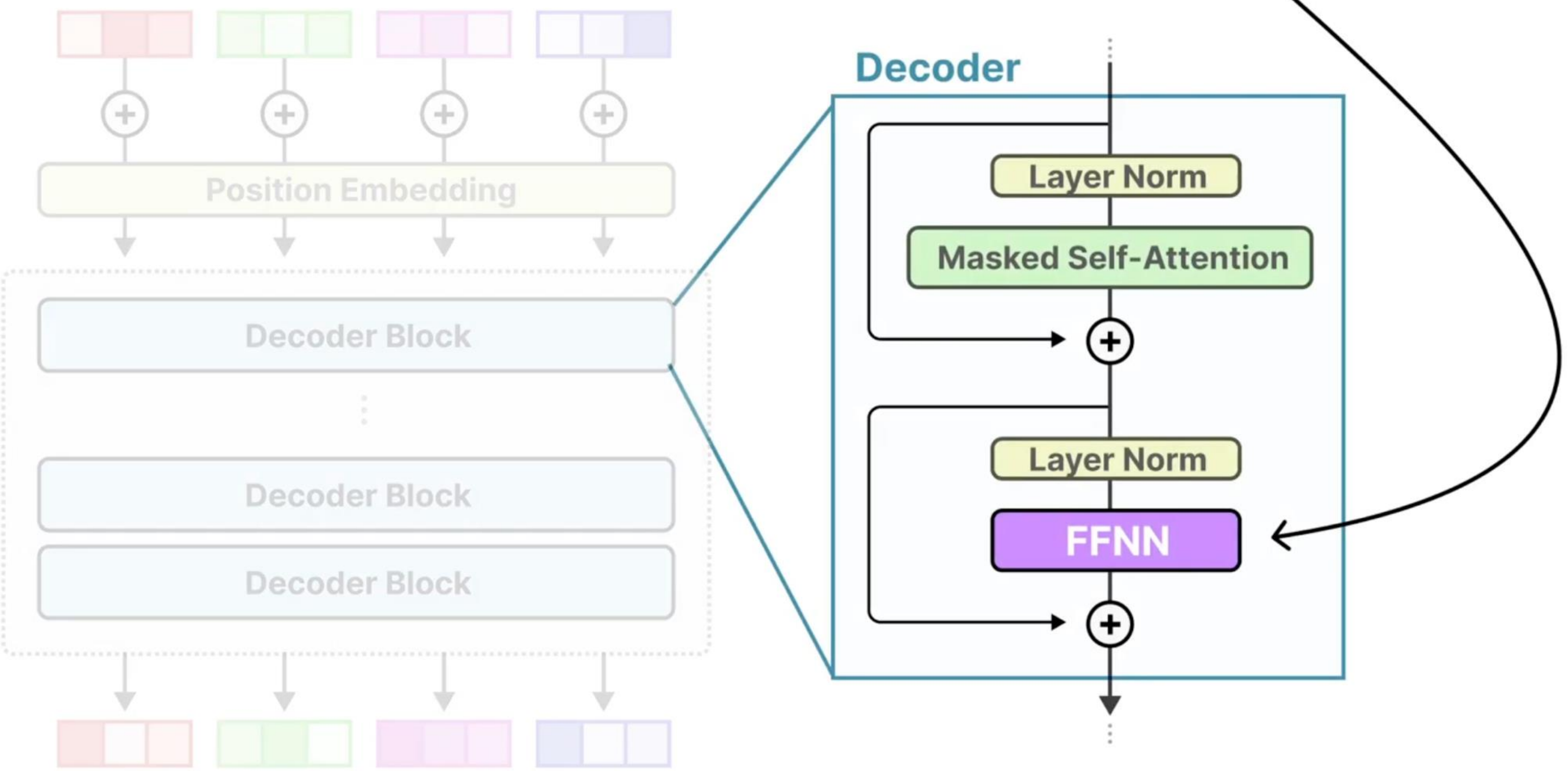
Dense Layers

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

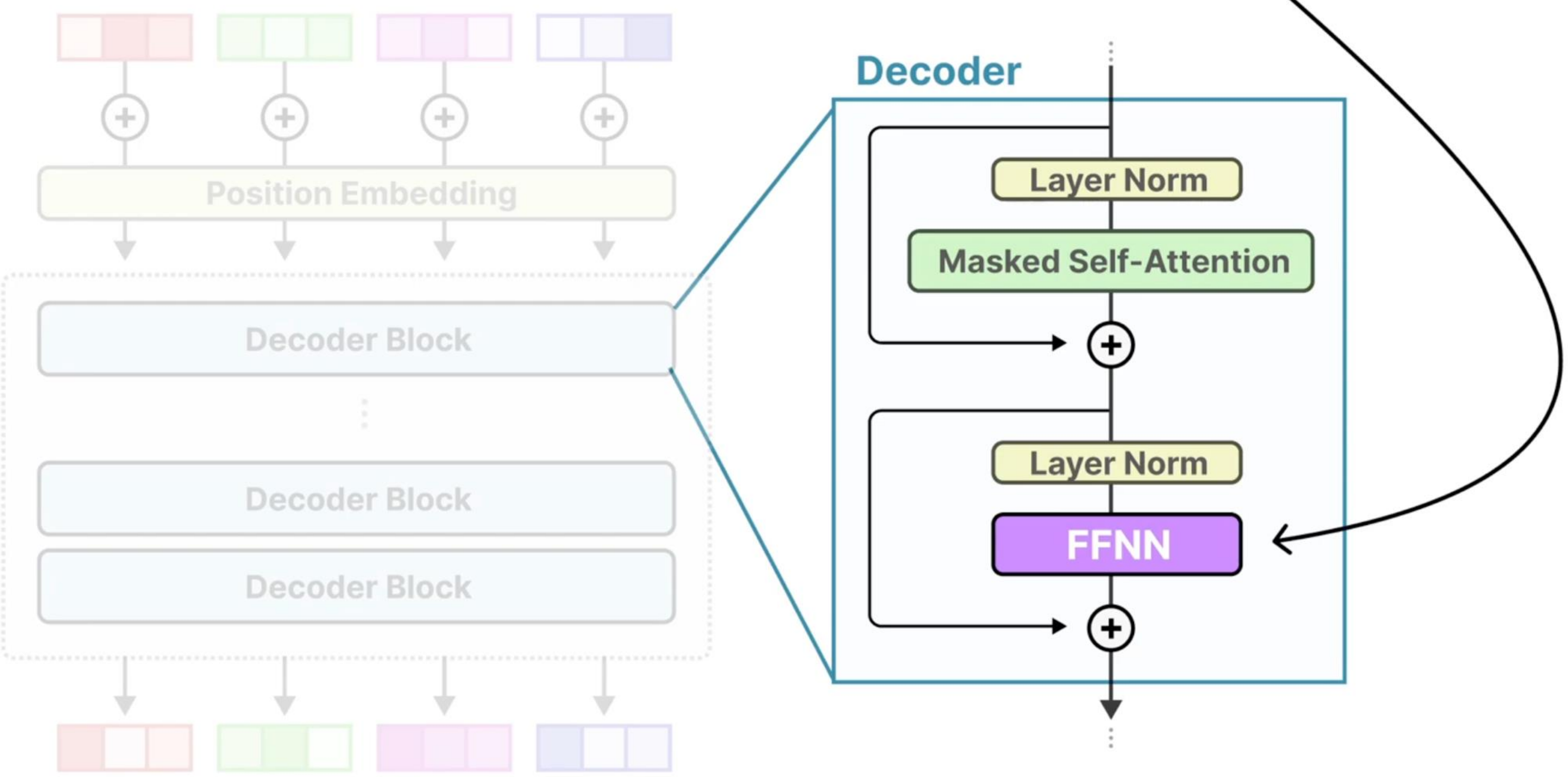


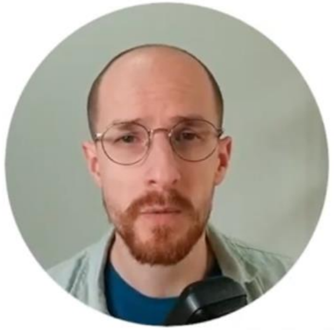
Introduction： MOE 基本原理

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

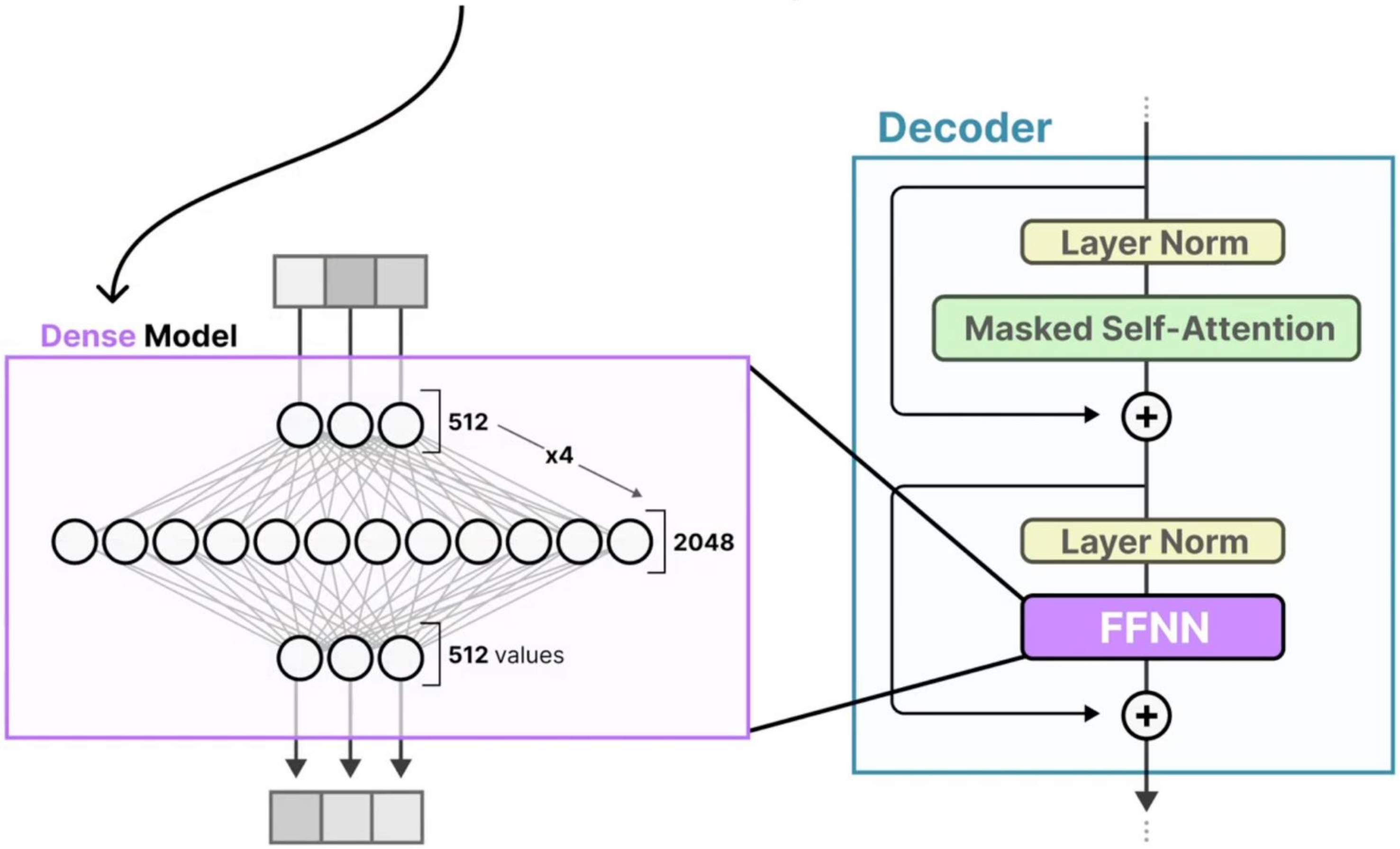


The FFNN uses the contextual information created by attention to capture complex relationships in the data.

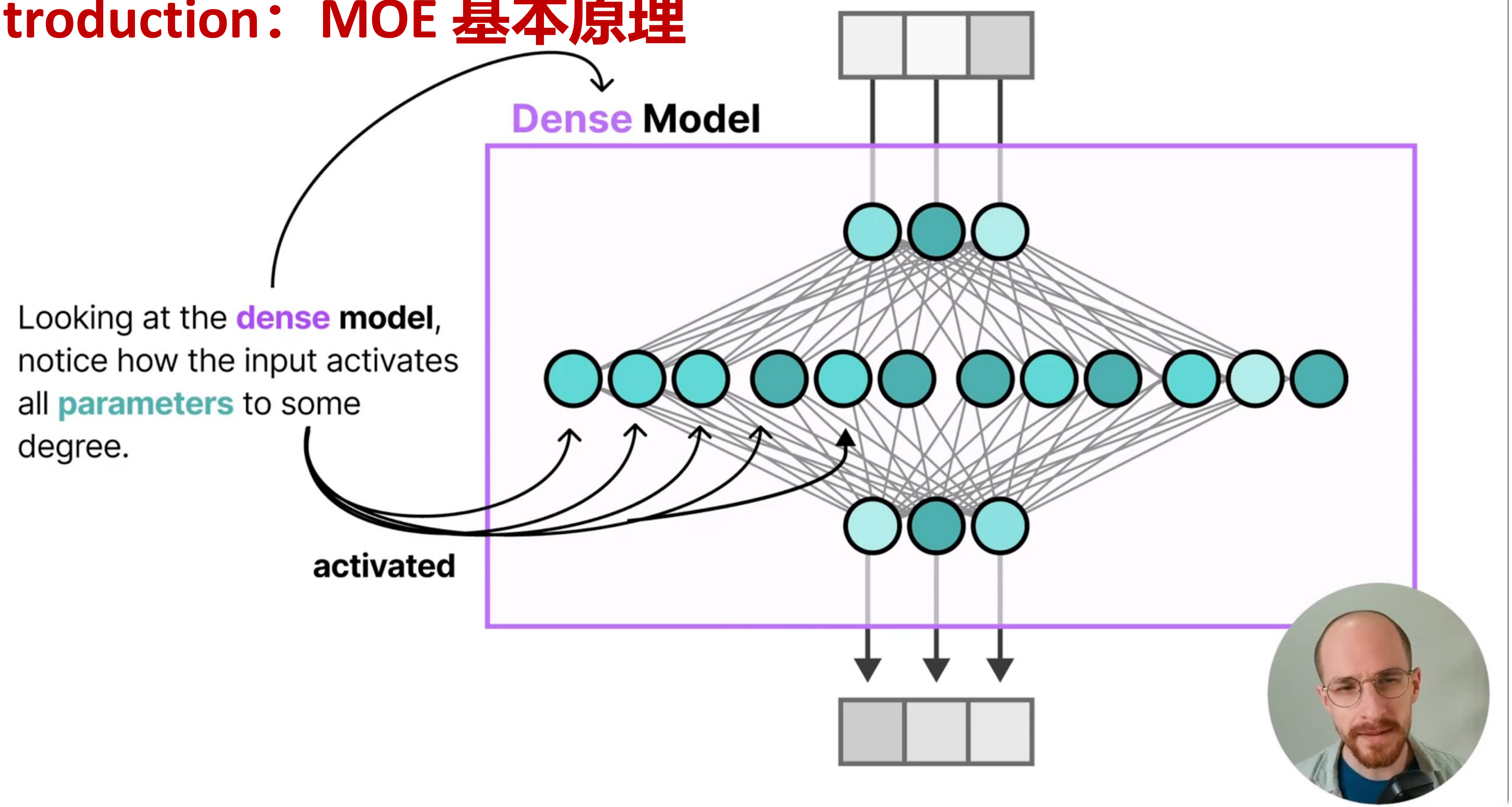




The **FFNN** in a **decoder-only** model is called
a **dense model** since all parameters are activated.

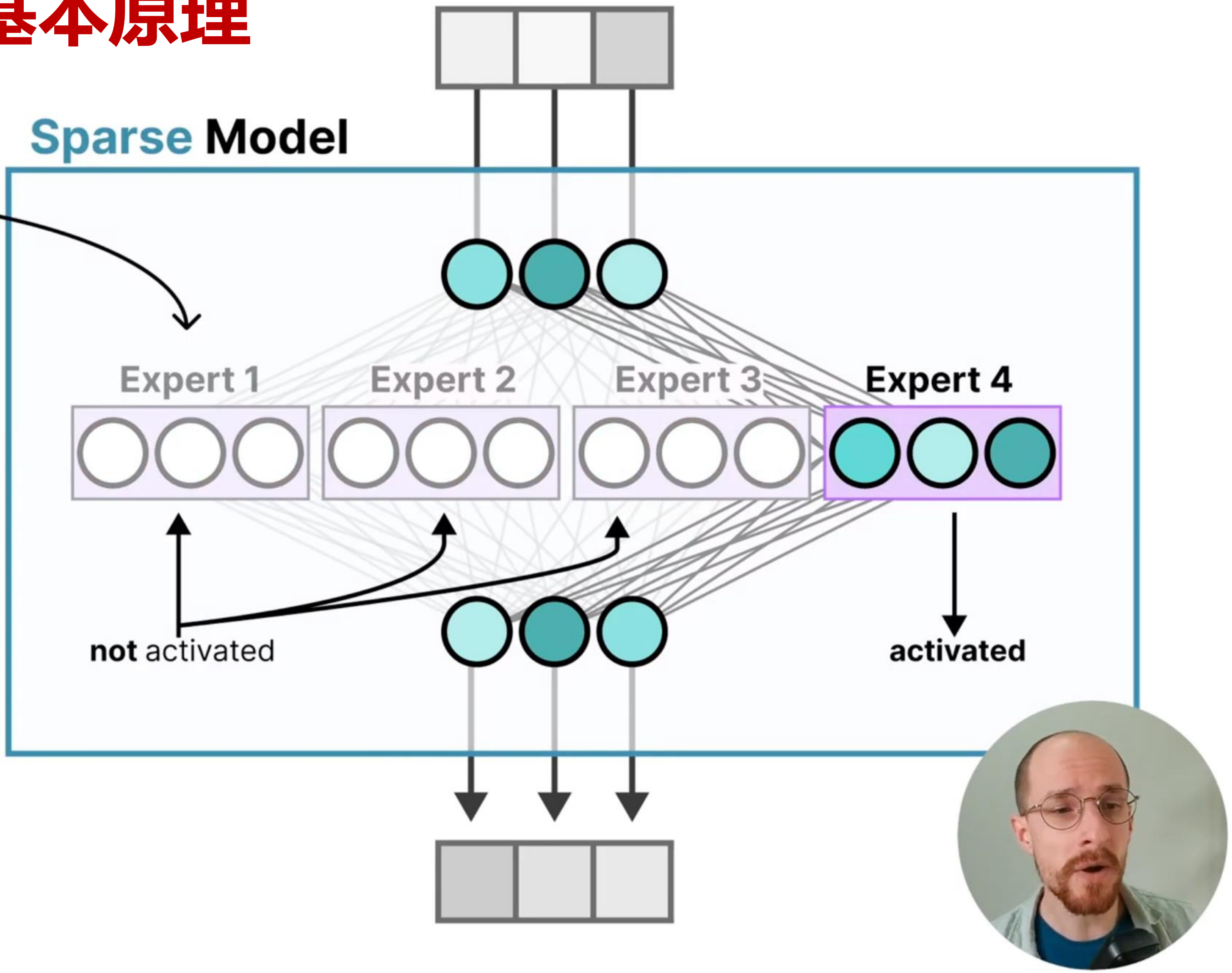


Introduction： MOE 基本原理



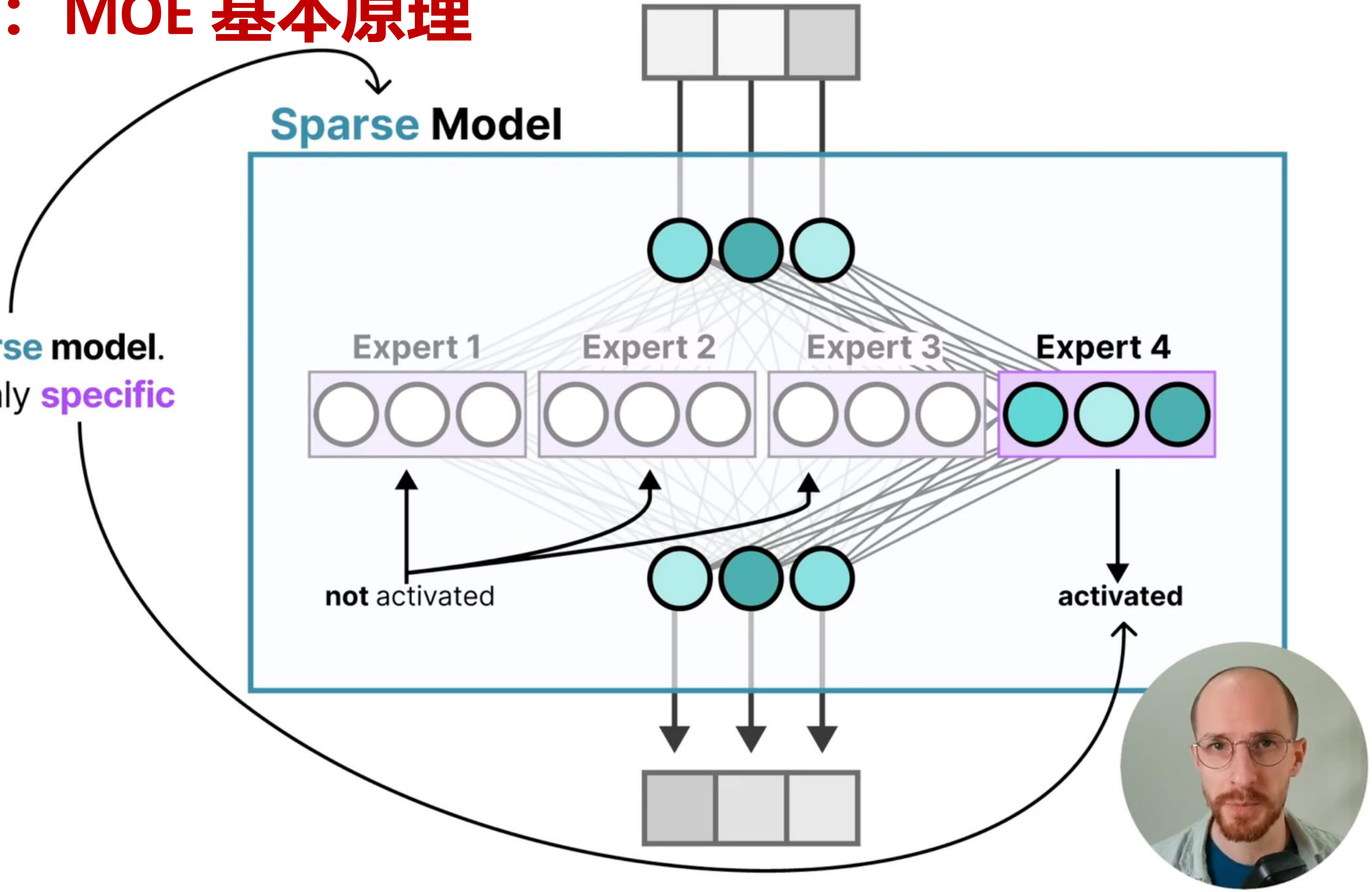
Introduction： MOE 基本原理

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.



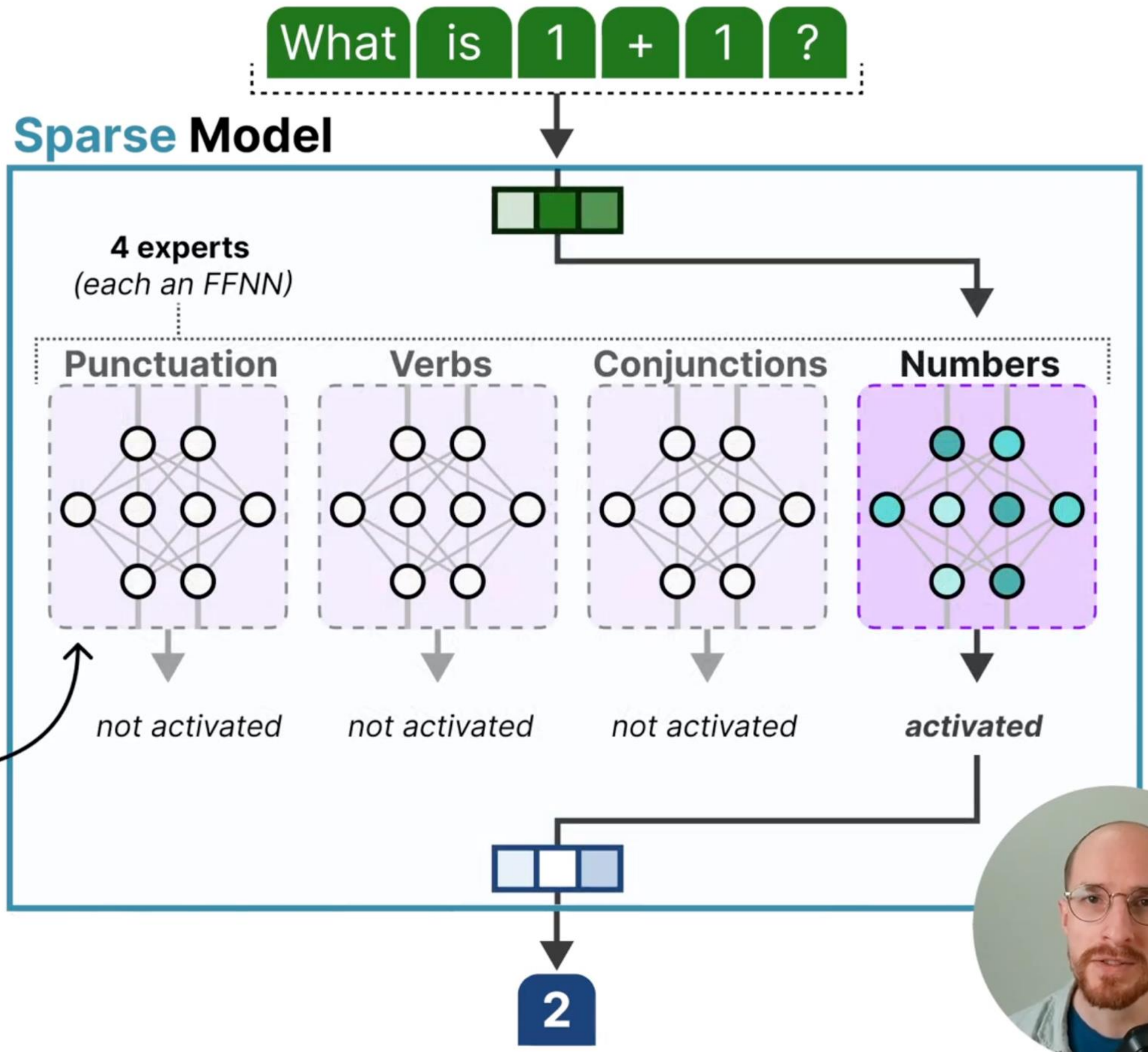
Introduction： MOE 基本原理

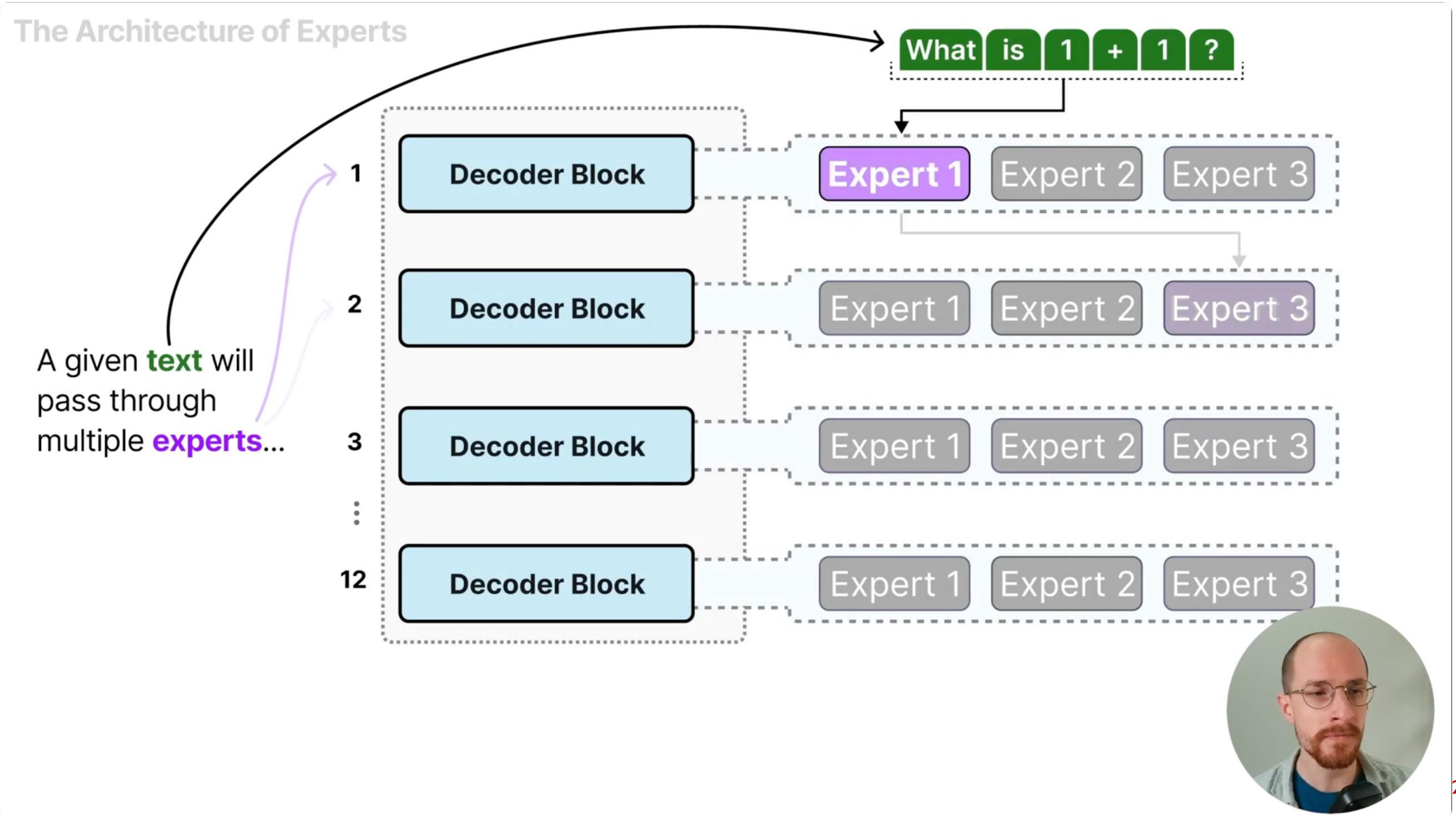
This is called a Sparse **model**. During inference, only **specific** experts are used.



Sparse Layers

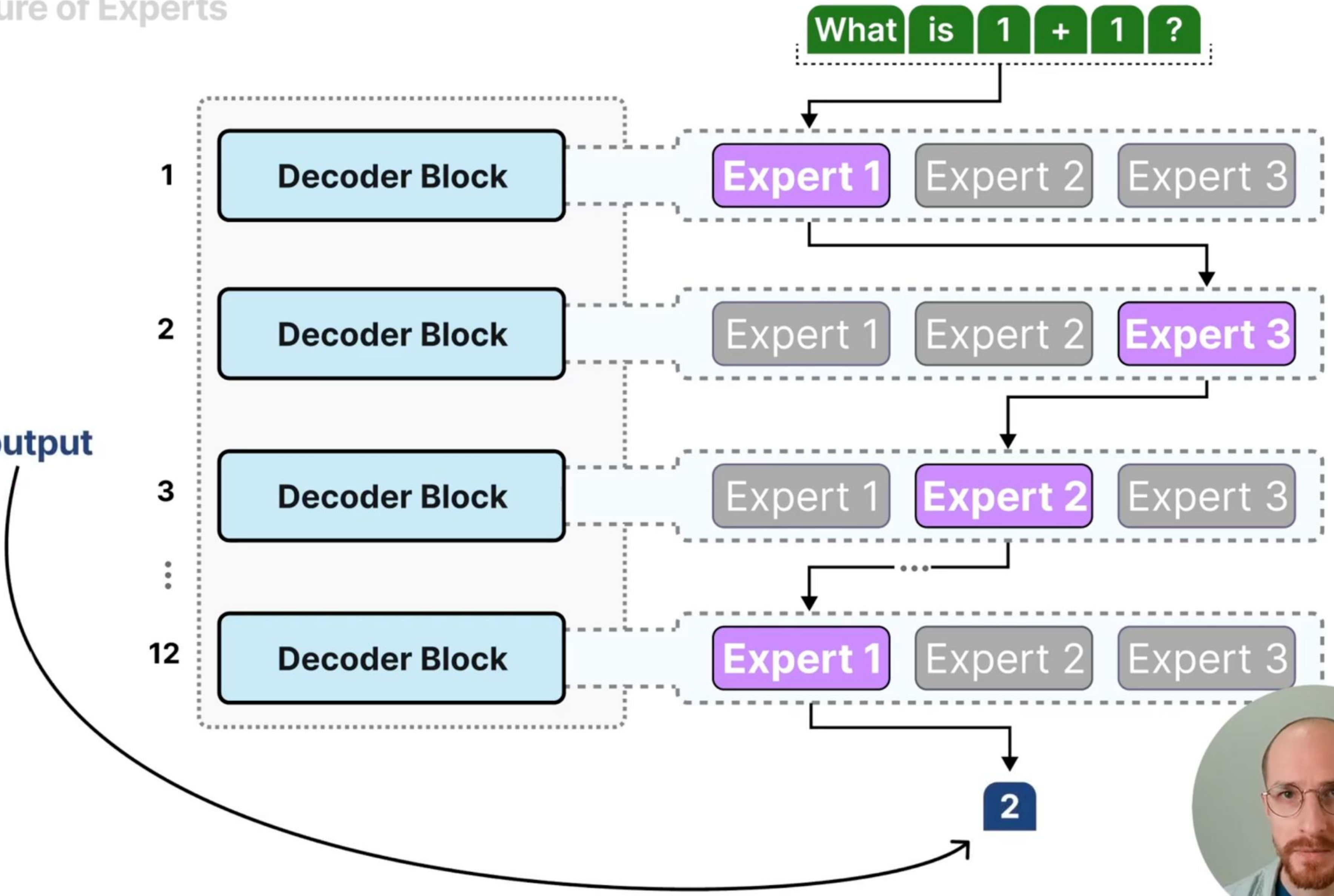
In practice, experts are typically **whole FFNNs** themselves...




The Architecture of Experts
What
is
1
+
1
?
1
2
3
12
Decoder Block
Decoder Block
Decoder Block
Decoder Block
Expert 1
Expert 2
Expert 3
Expert 1
Expert 2
Expert 3
Expert 1
Expert 2
Expert 3
Expert 1
Expert 2
Expert 3
A given text will
pass through
multiple experts...

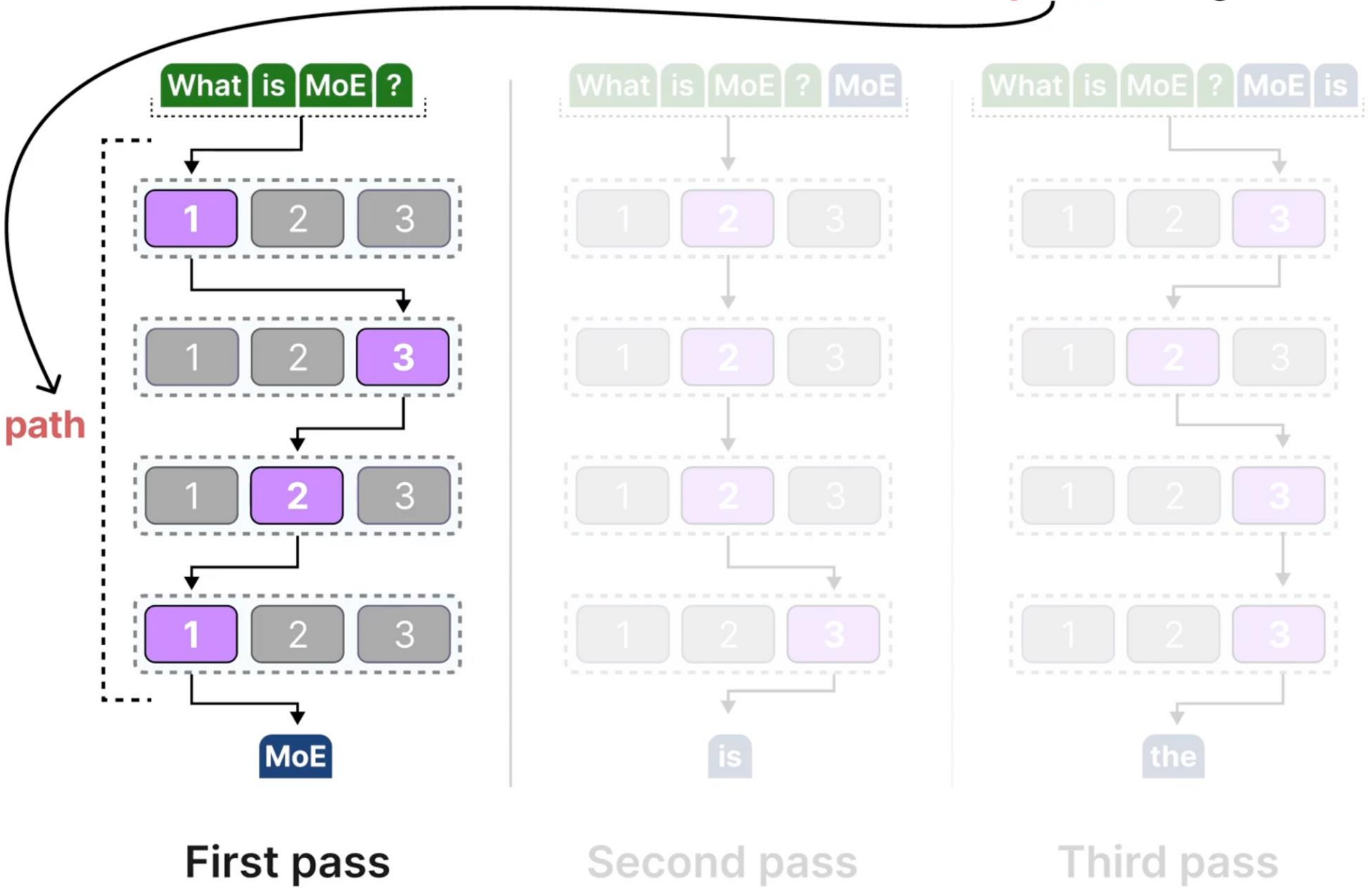
The Architecture of Experts

... before the **output** is generated.

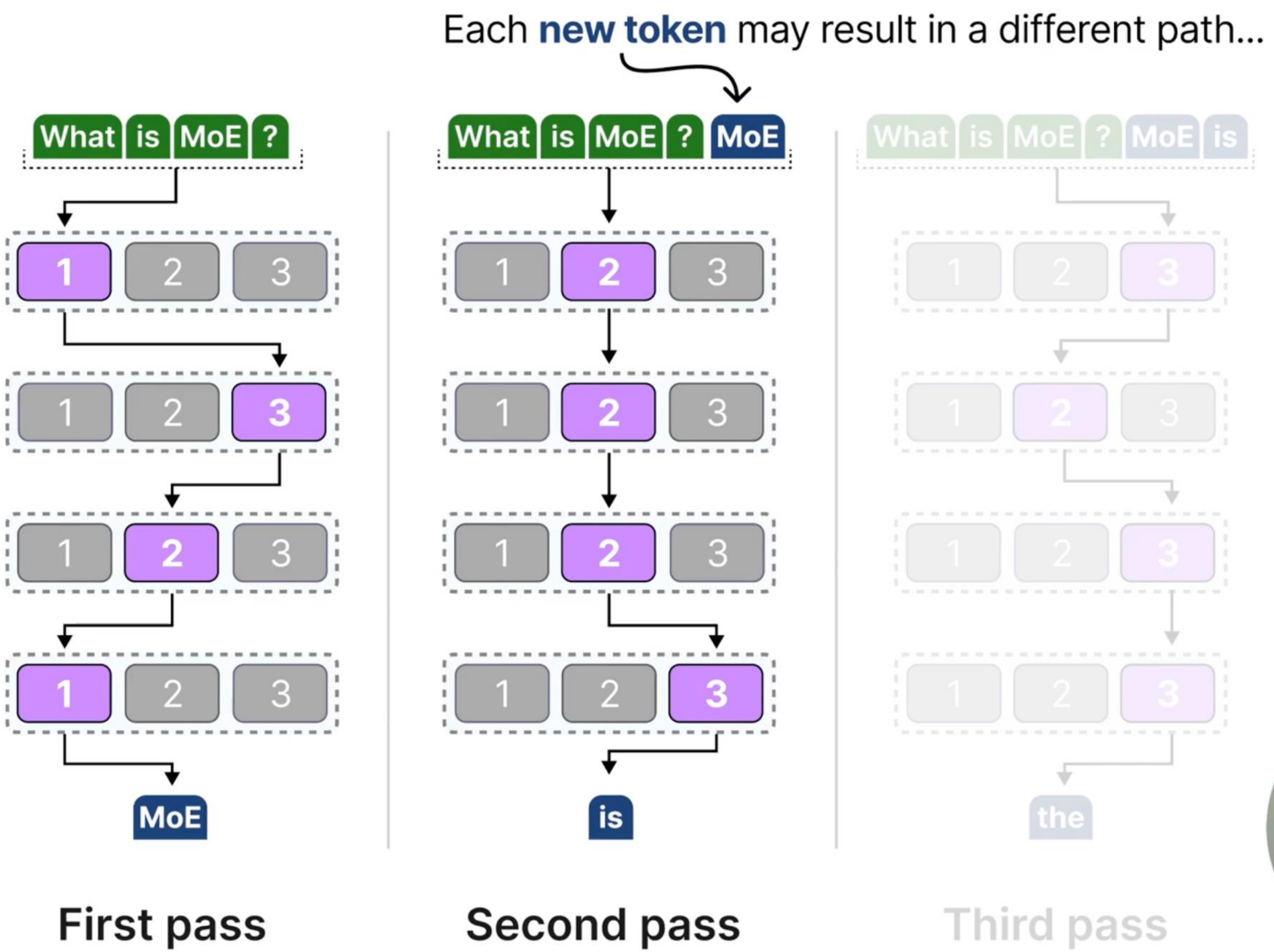


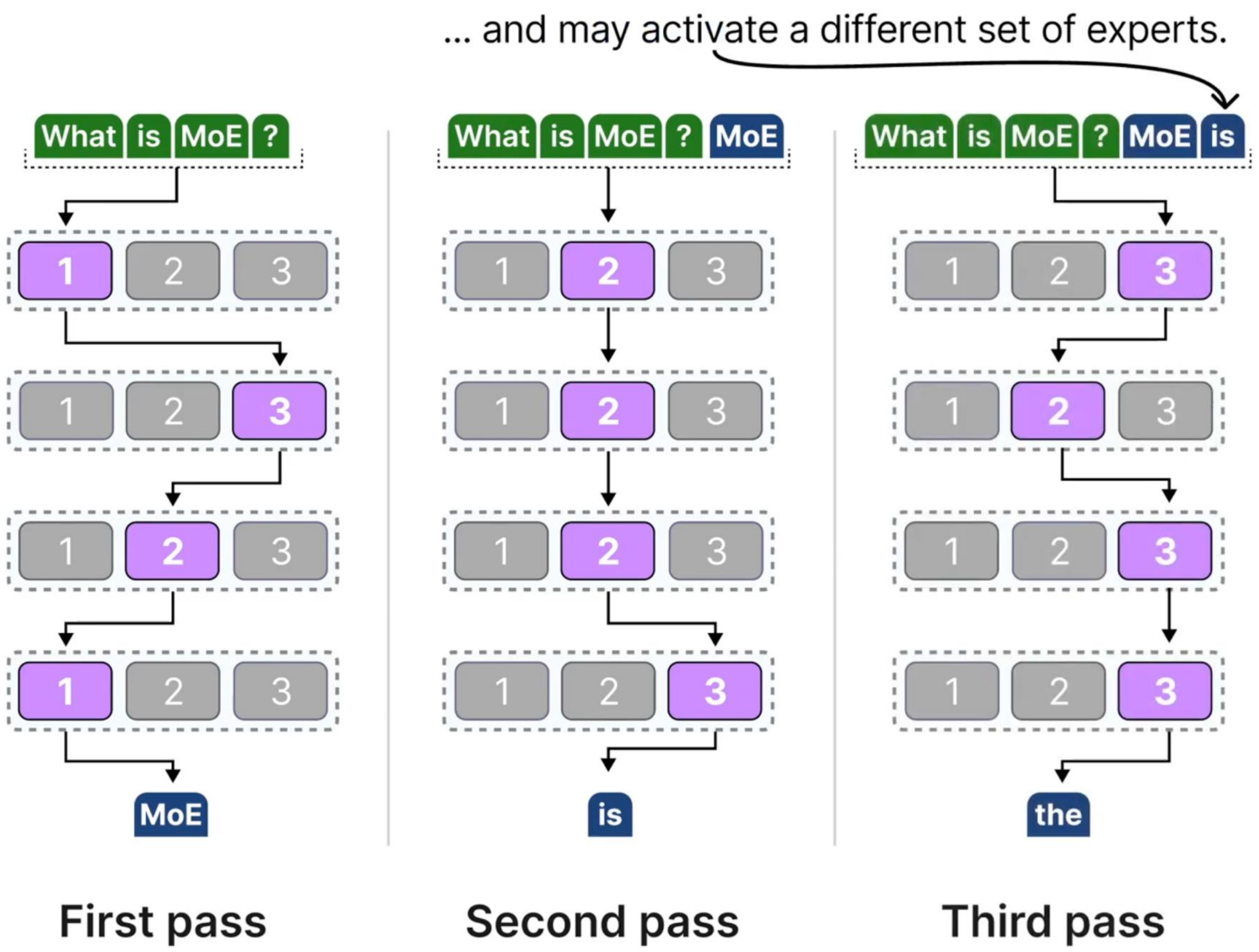
The Architecture of Experts

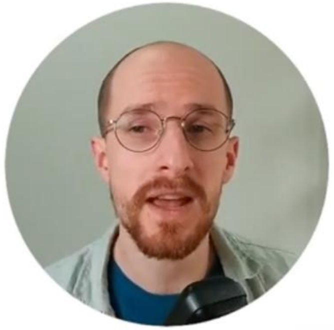
The **chosen experts** likely differ between tokens which results in different “**paths**” being taken.



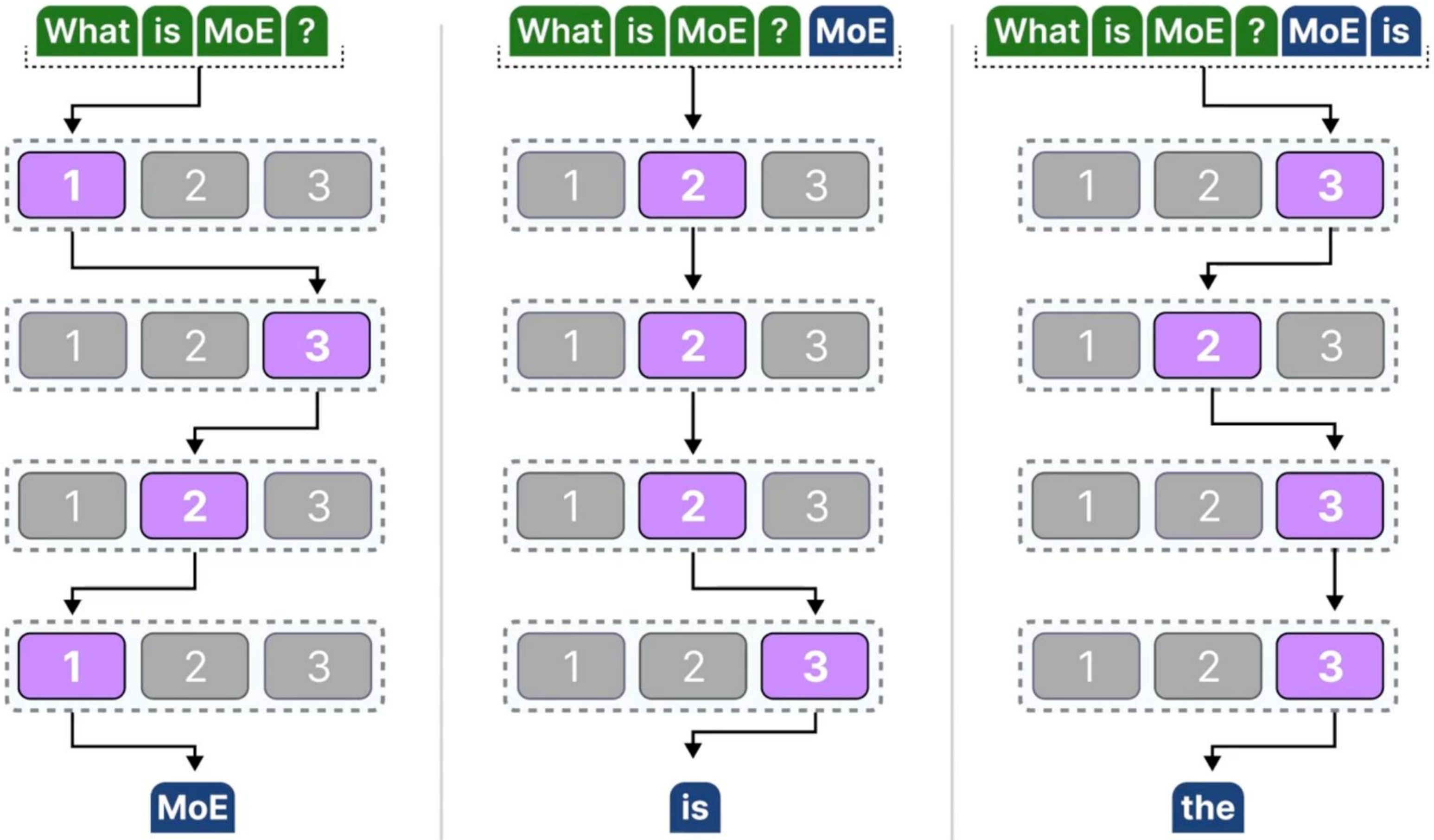
The Architecture of Experts




... and may activate a different set of experts.
What is MoE ?
1 2 3
MoE
What is MoE ? MoE
is
What is MoE ? MoE is
the
First pass
Second pass
Third pass

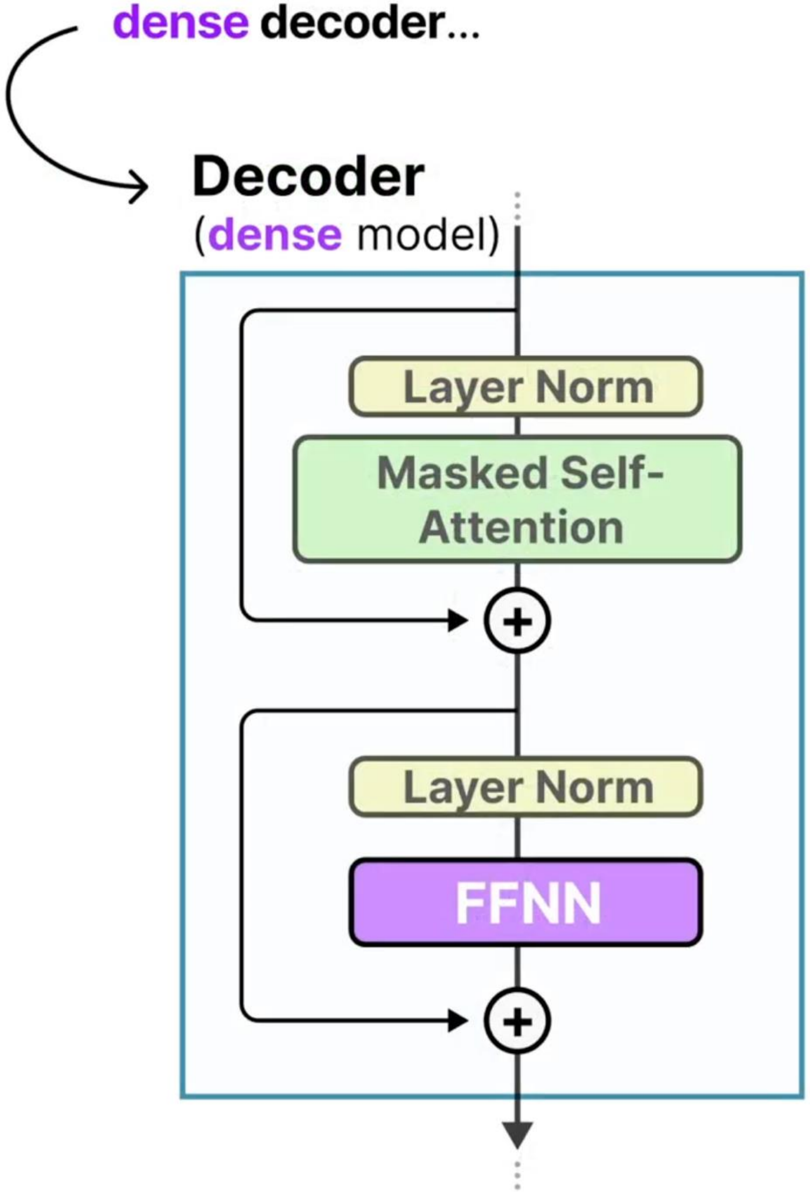


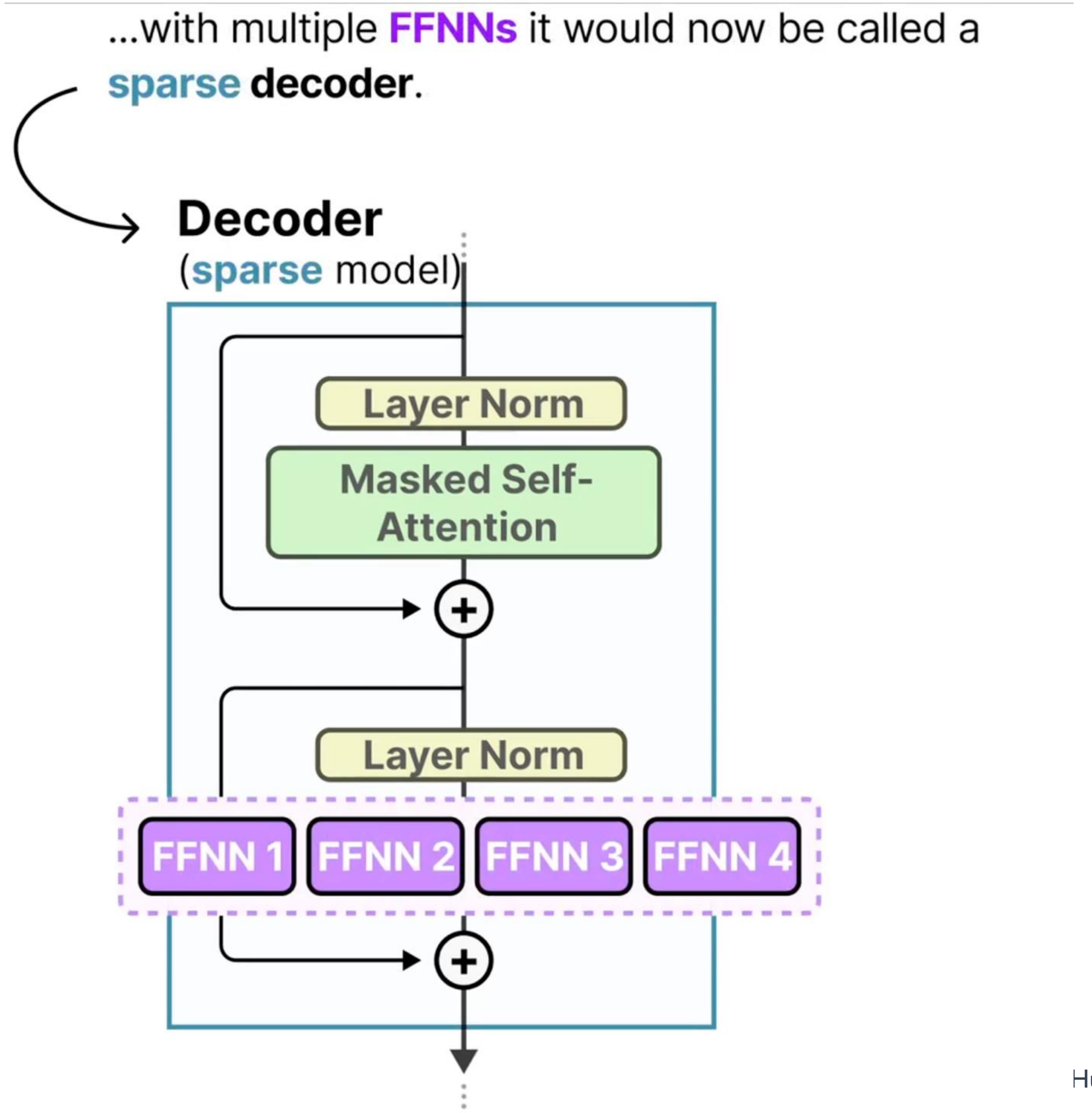
This means that each time you run inference, a set of **experts** is chosen that are best suited for the input.



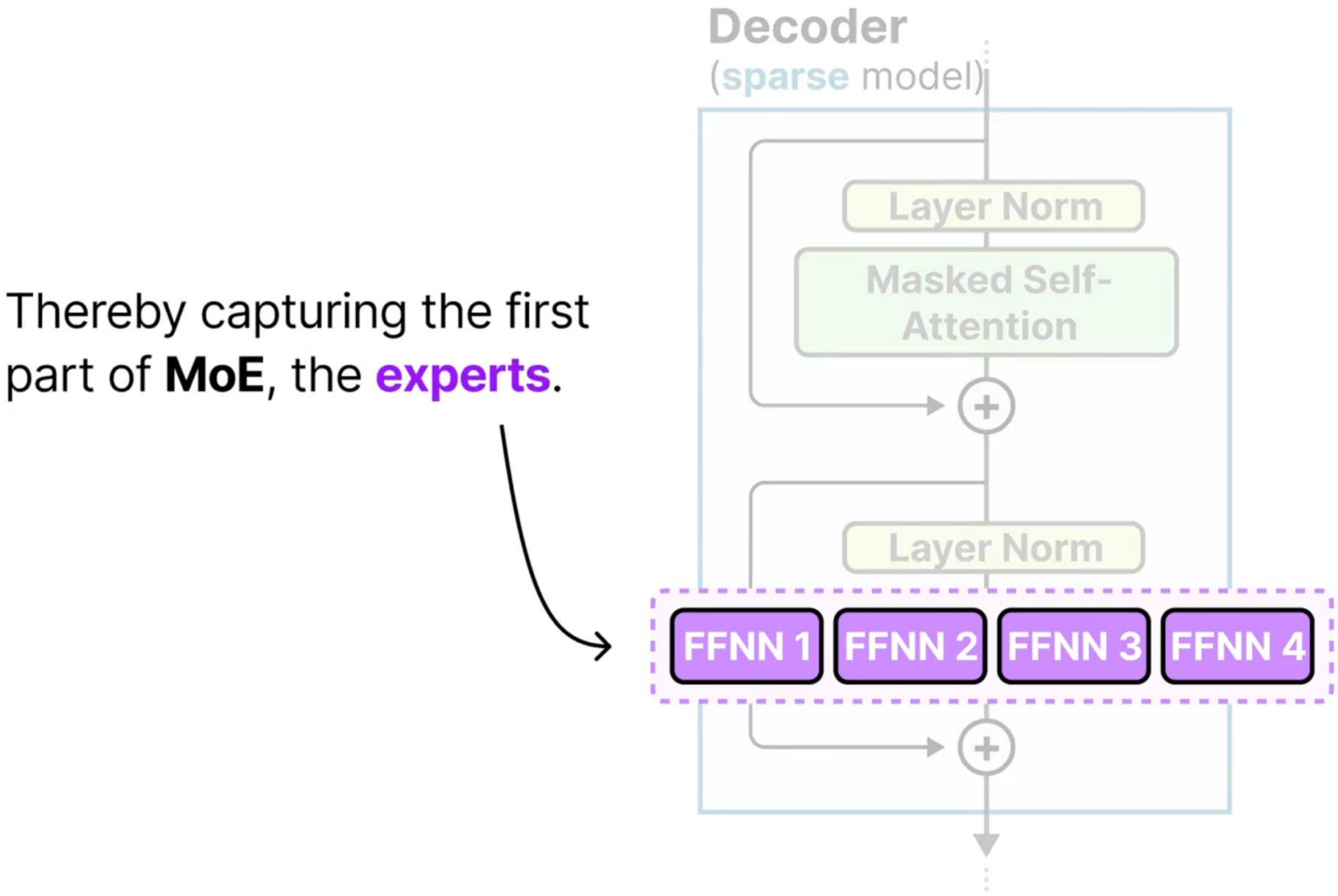
First pass　Second pass　Third pass

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




...with multiple FFNNs it would now be called a sparse decoder.
Decoder
(sparse model)
Layer Norm
Masked Self-Attention
Layer Norm
FFNN 1
FFNN 2
FFNN 3
FFNN 4

Introduction： MOE 基本原理

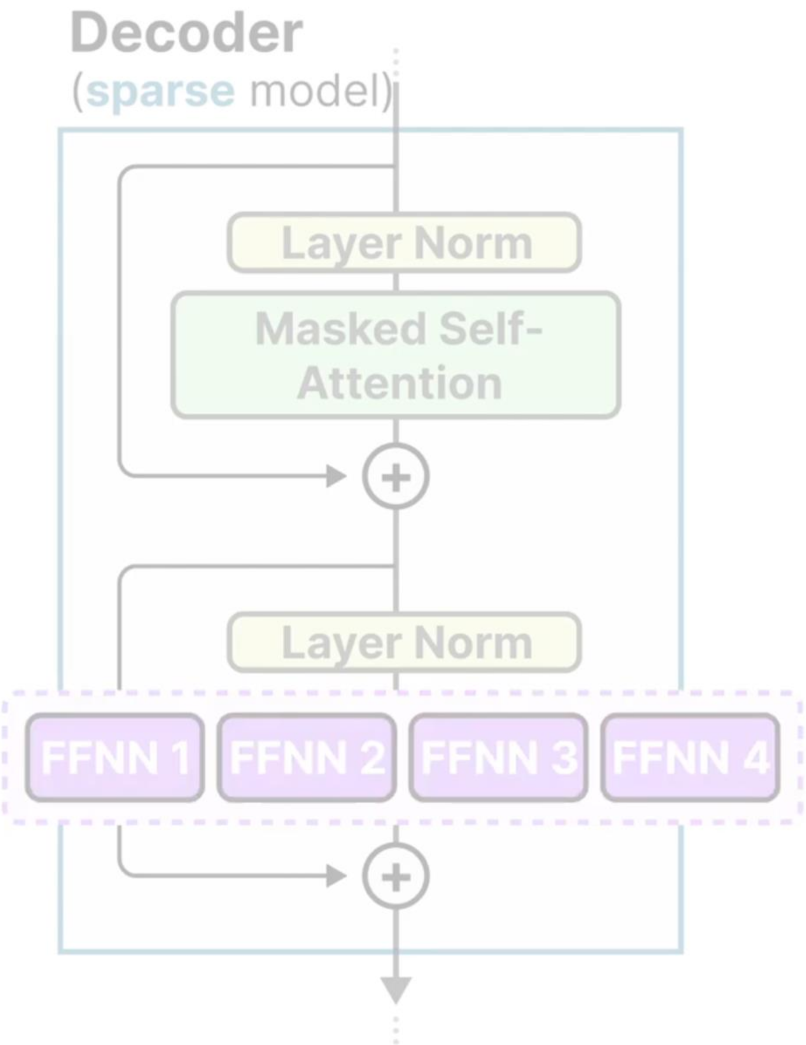


02

The Router：路由原理

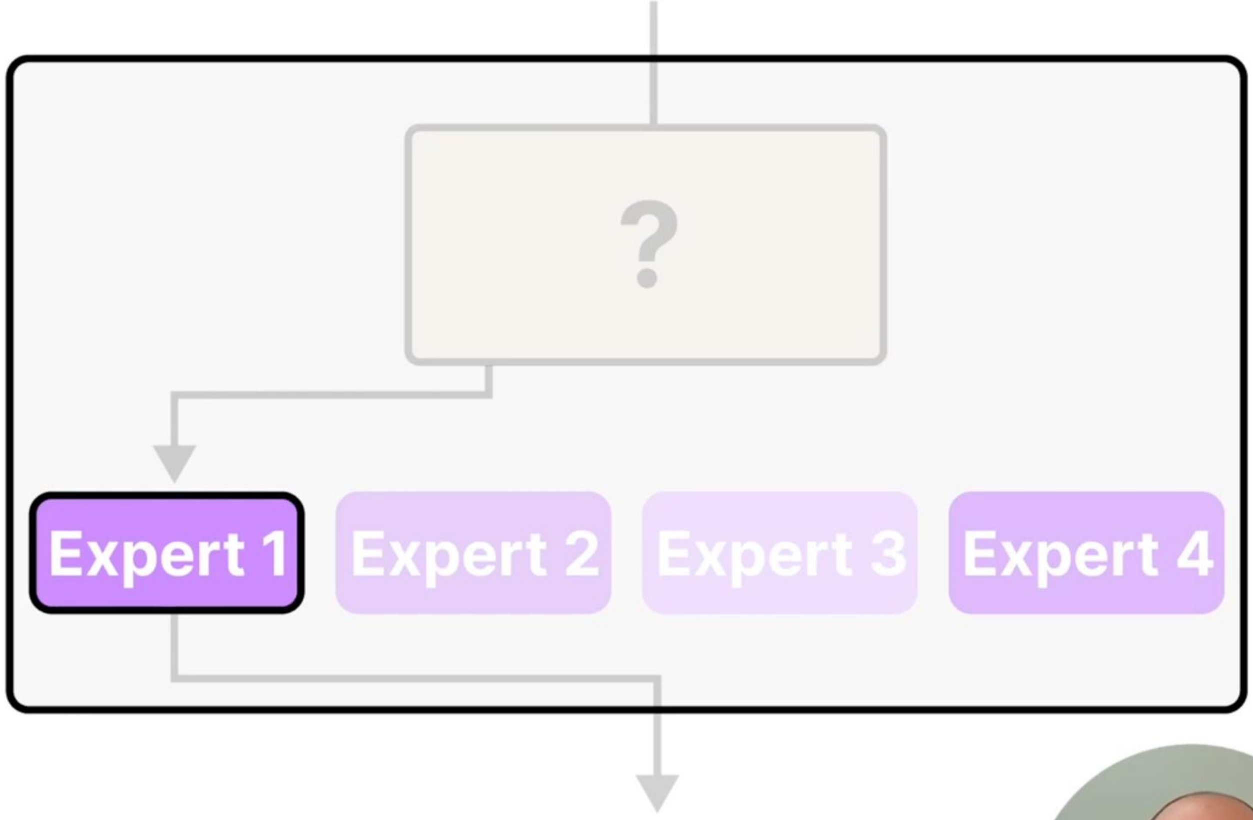
Introduction： MOE 基本原理

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



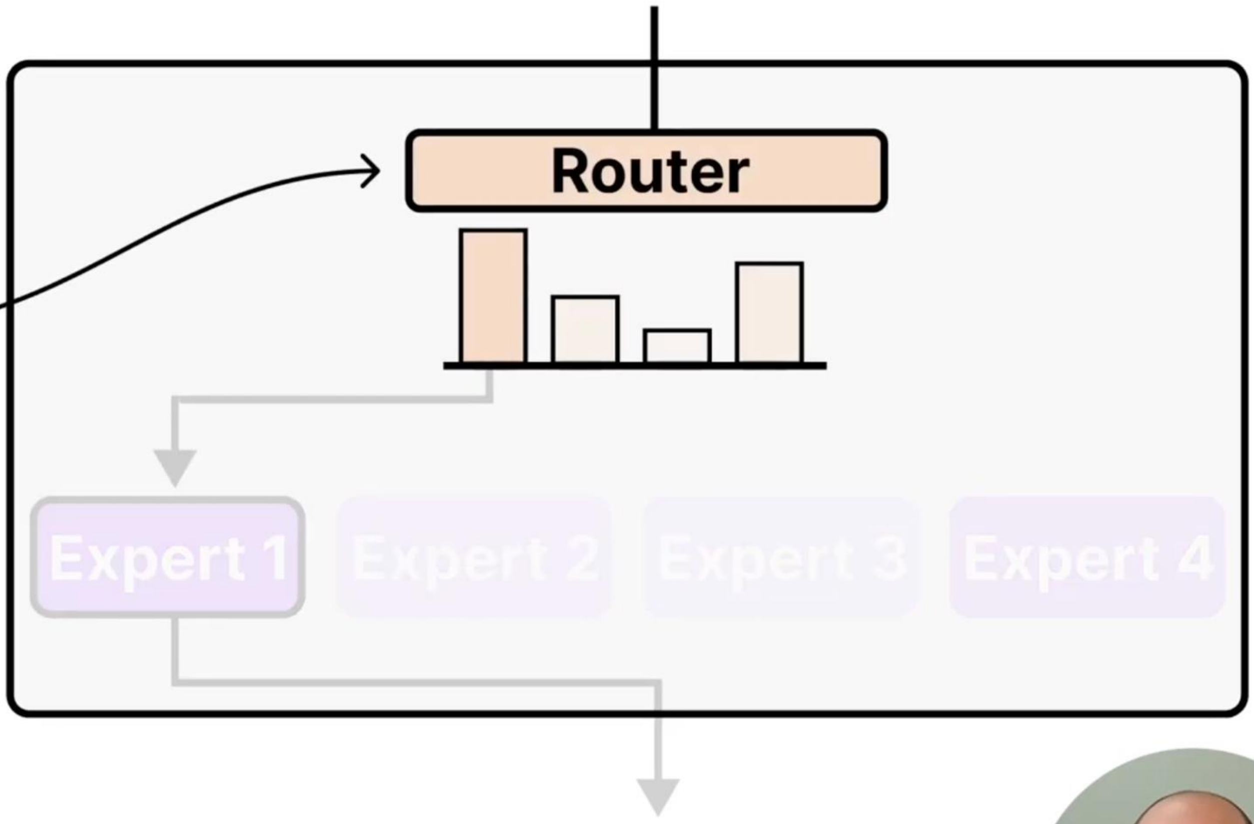
The Router

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

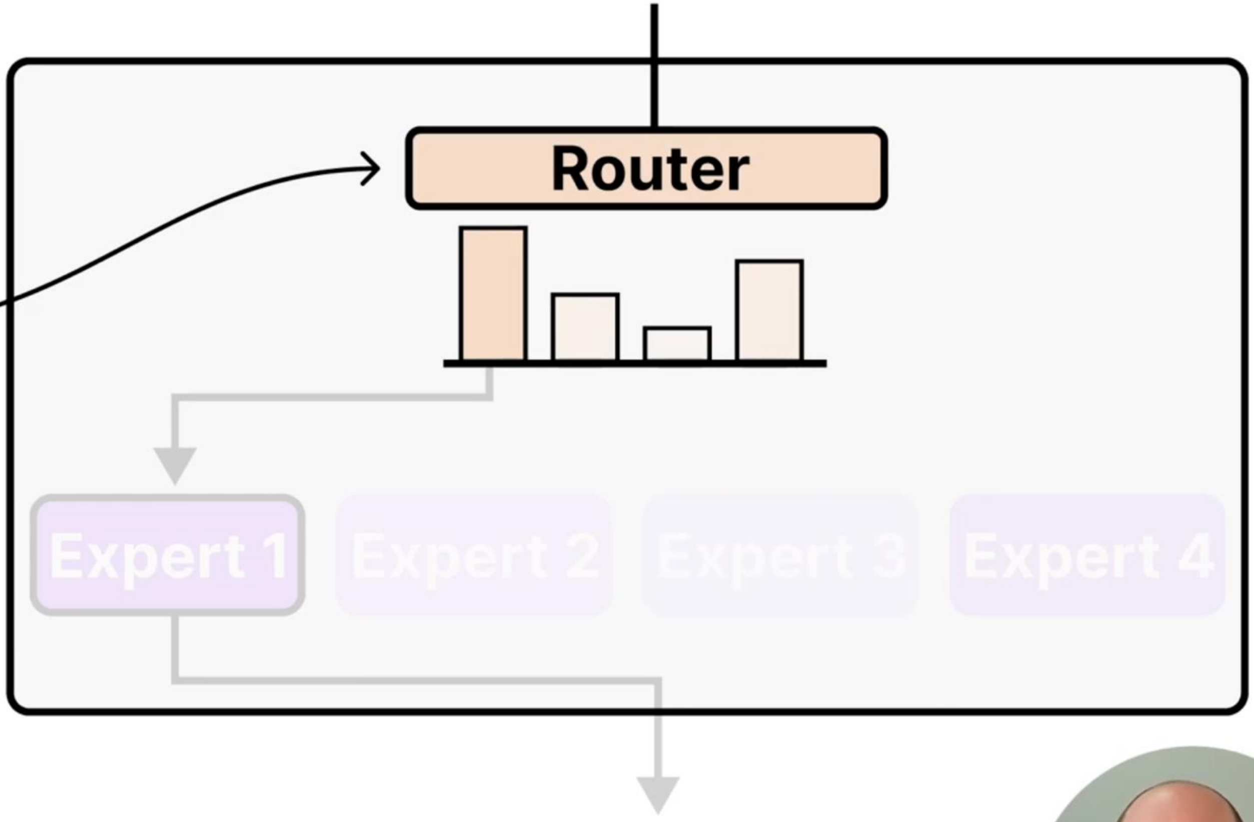


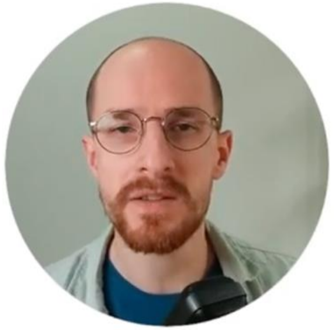
The Router

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



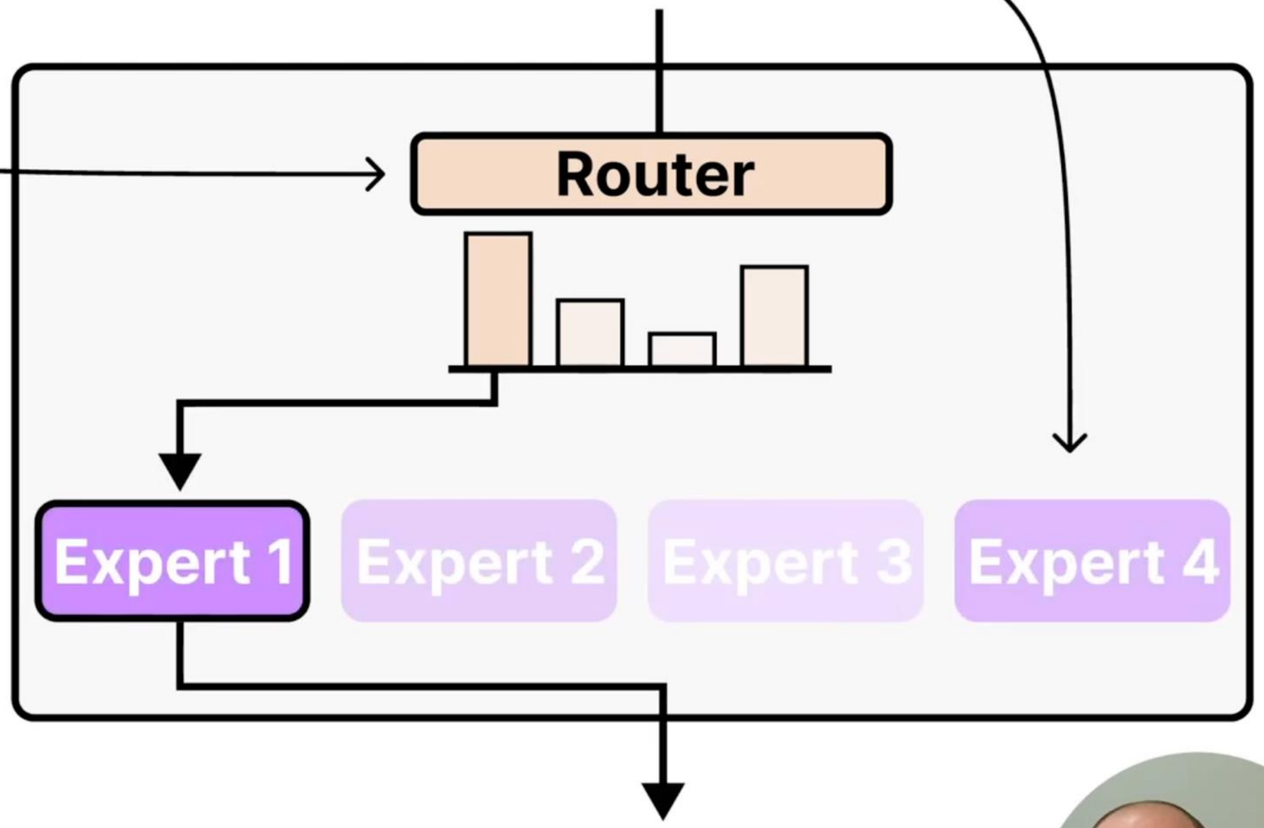
... it helps us decide which **expert** is best suited for a given input.





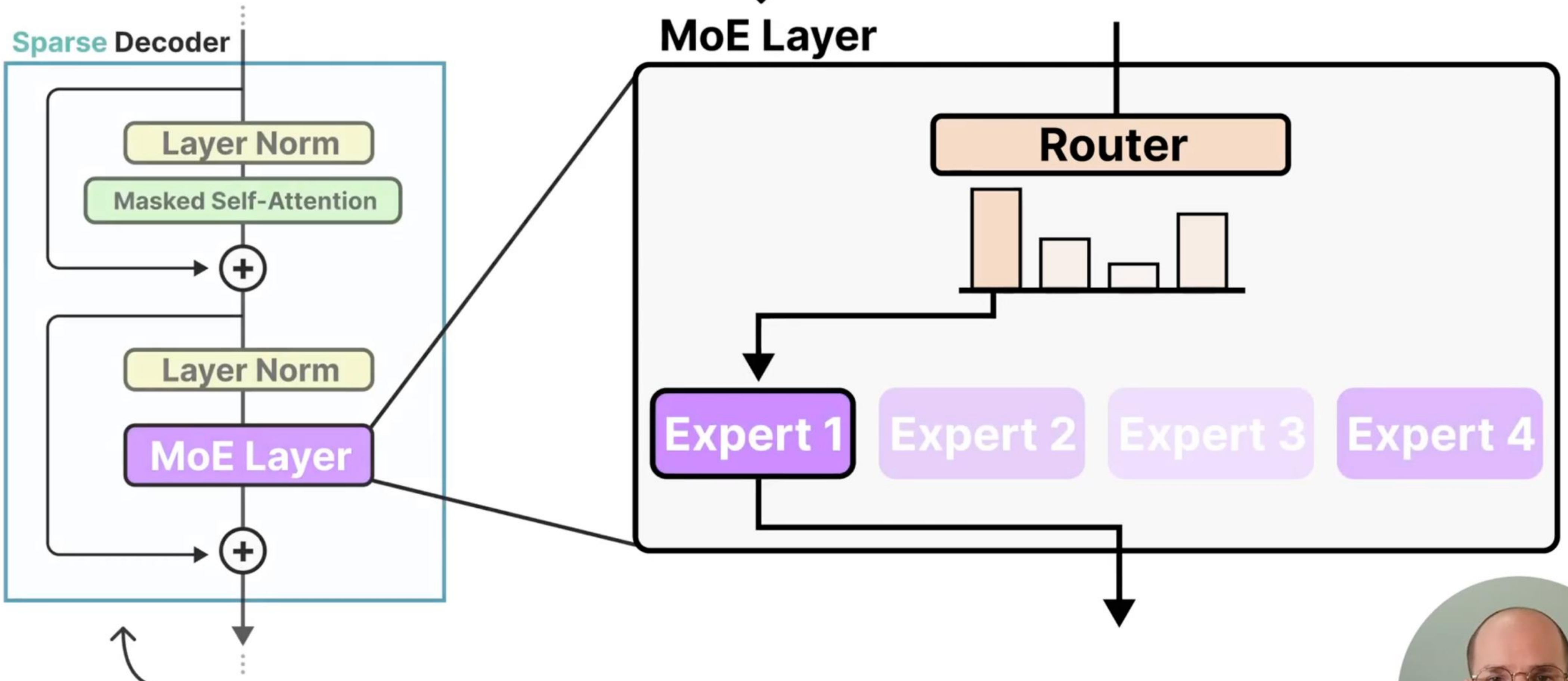
The Router

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



The Router

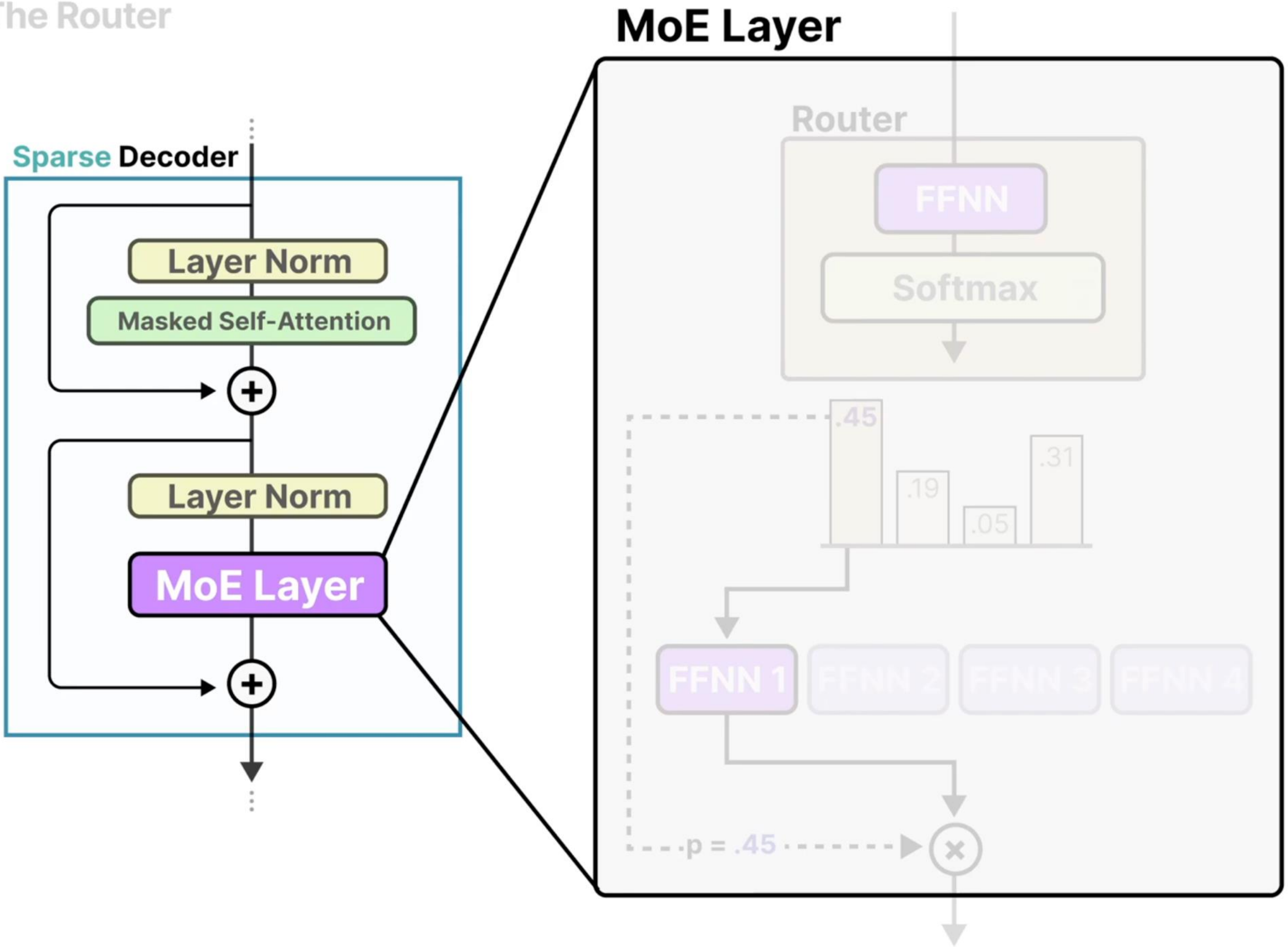
... make up the **MoE layer**...



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

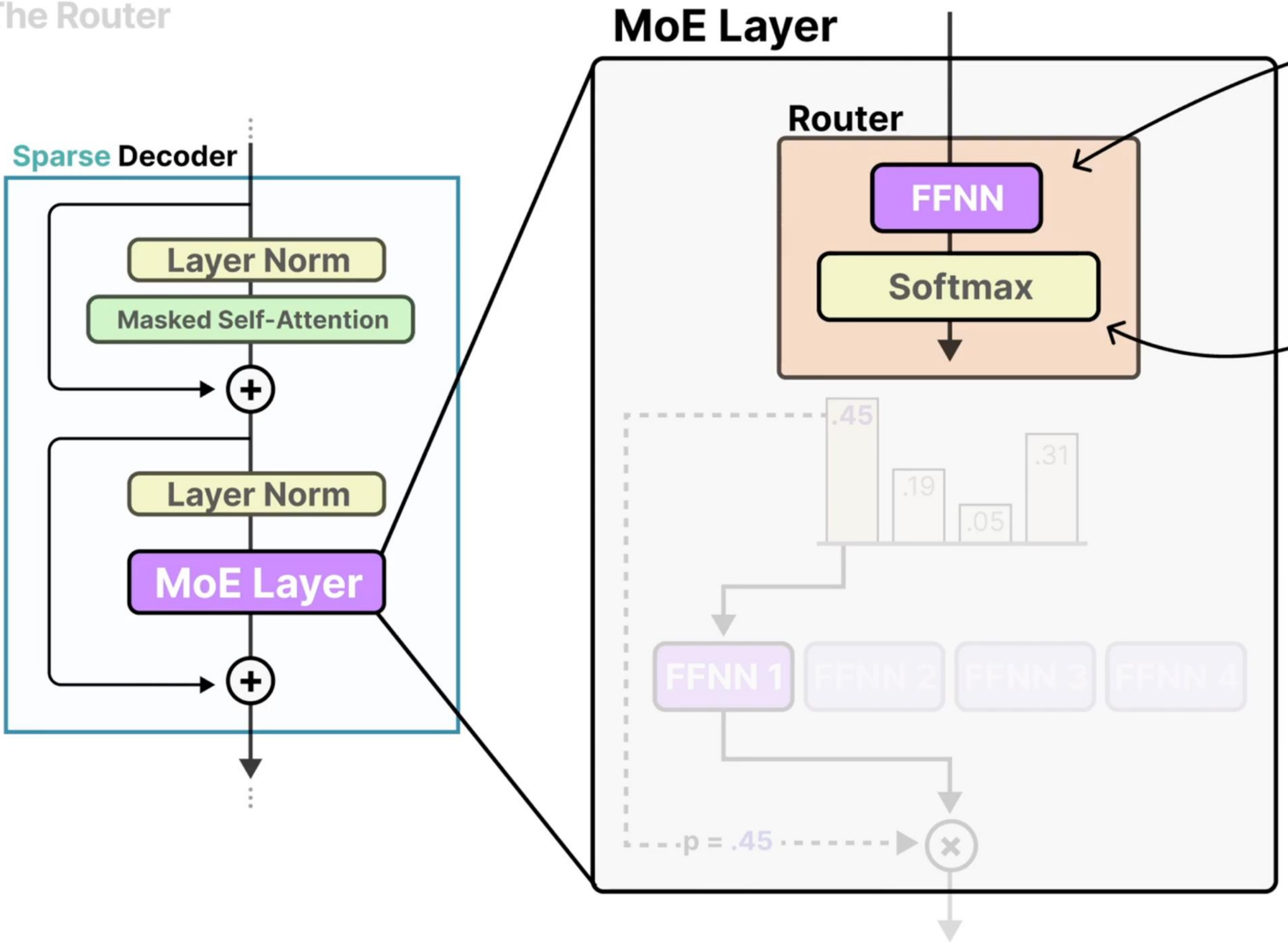
Z 2/AIInfra

The Router



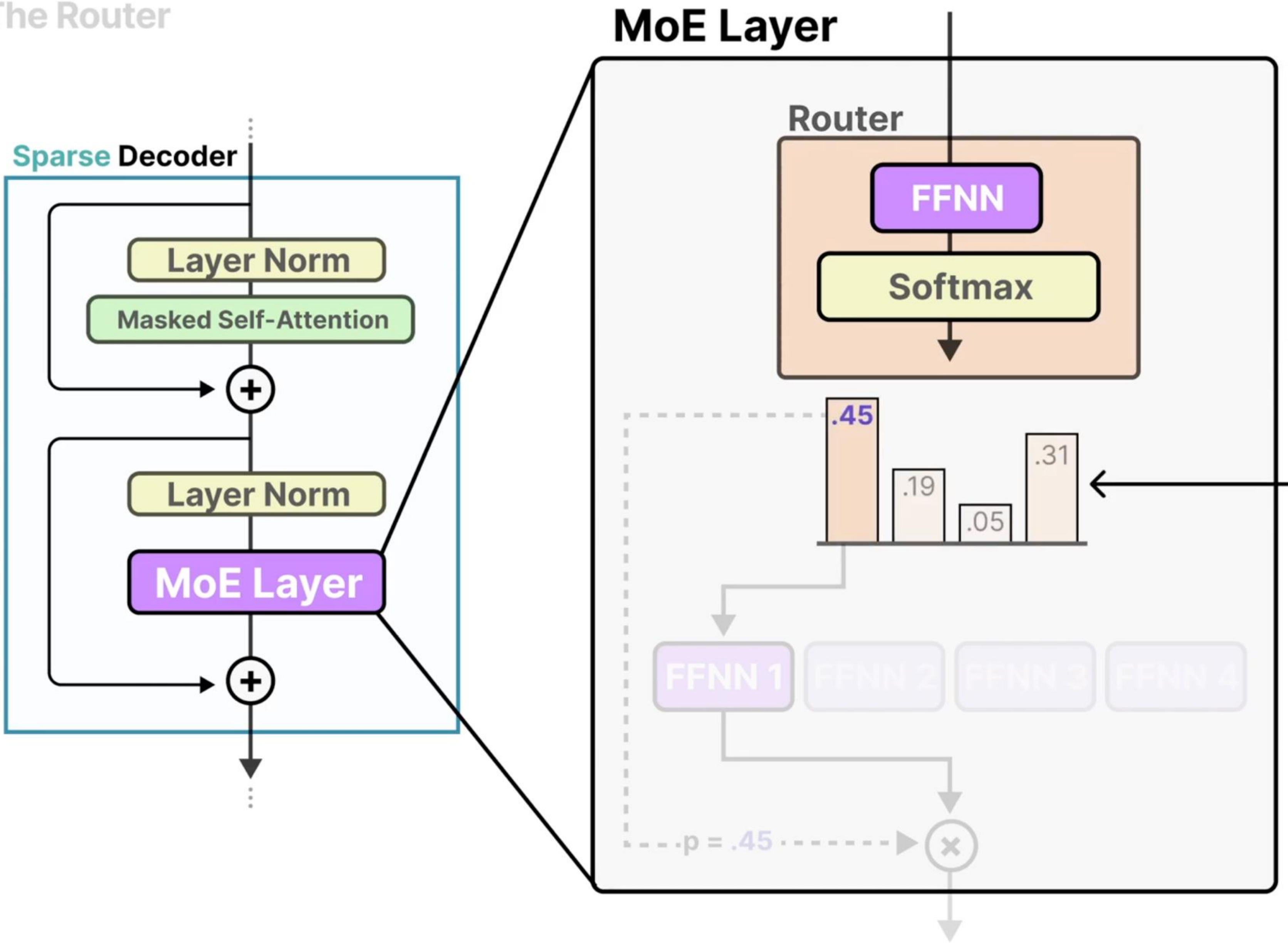
We can zoom in on the **MoE** layer and explore how the router works in detail.

The Router



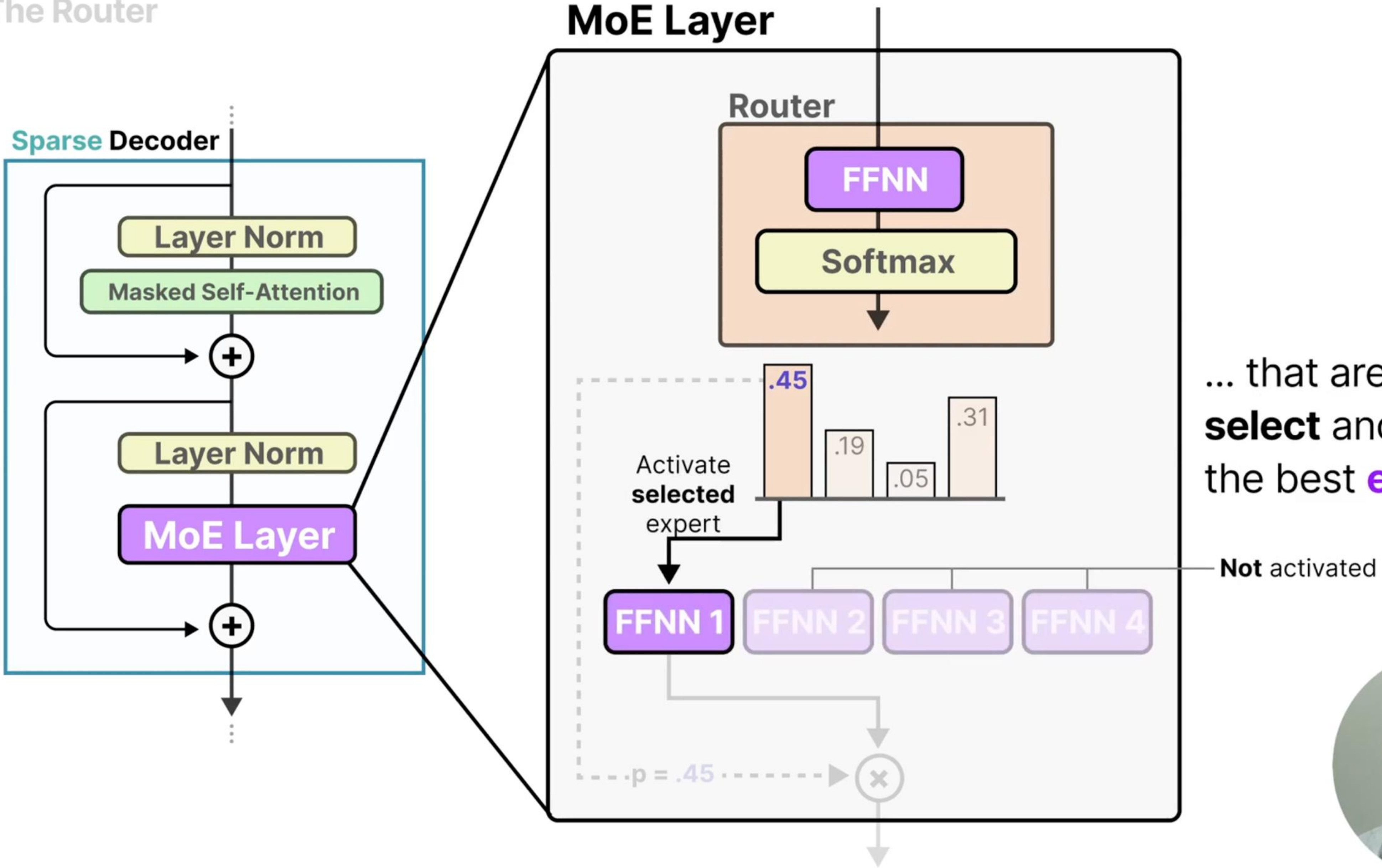
After the **FFNN** in the **router**, we see a **softmax function...**

The Router



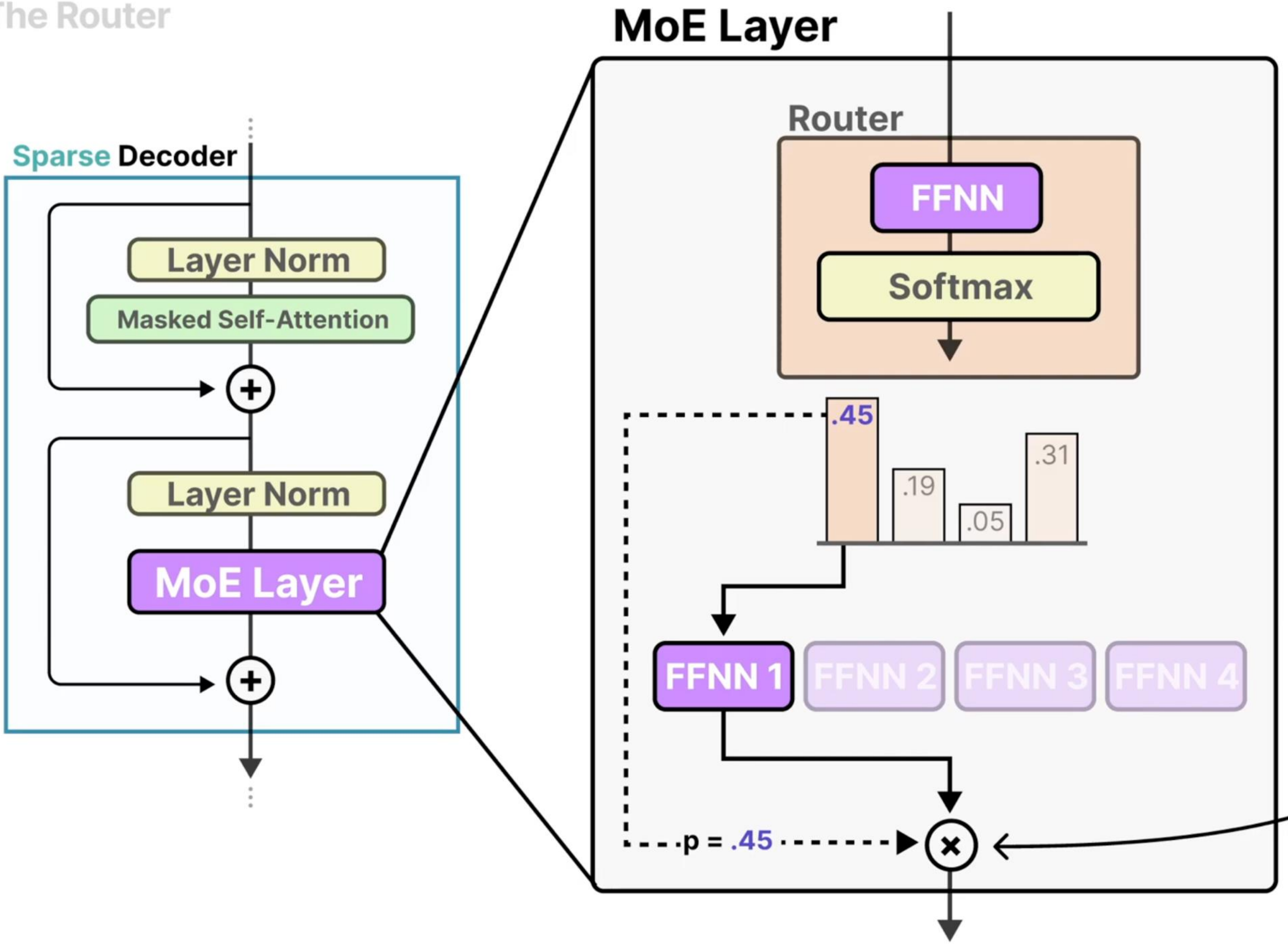
... creating **probabilities** for each expert...

The Router



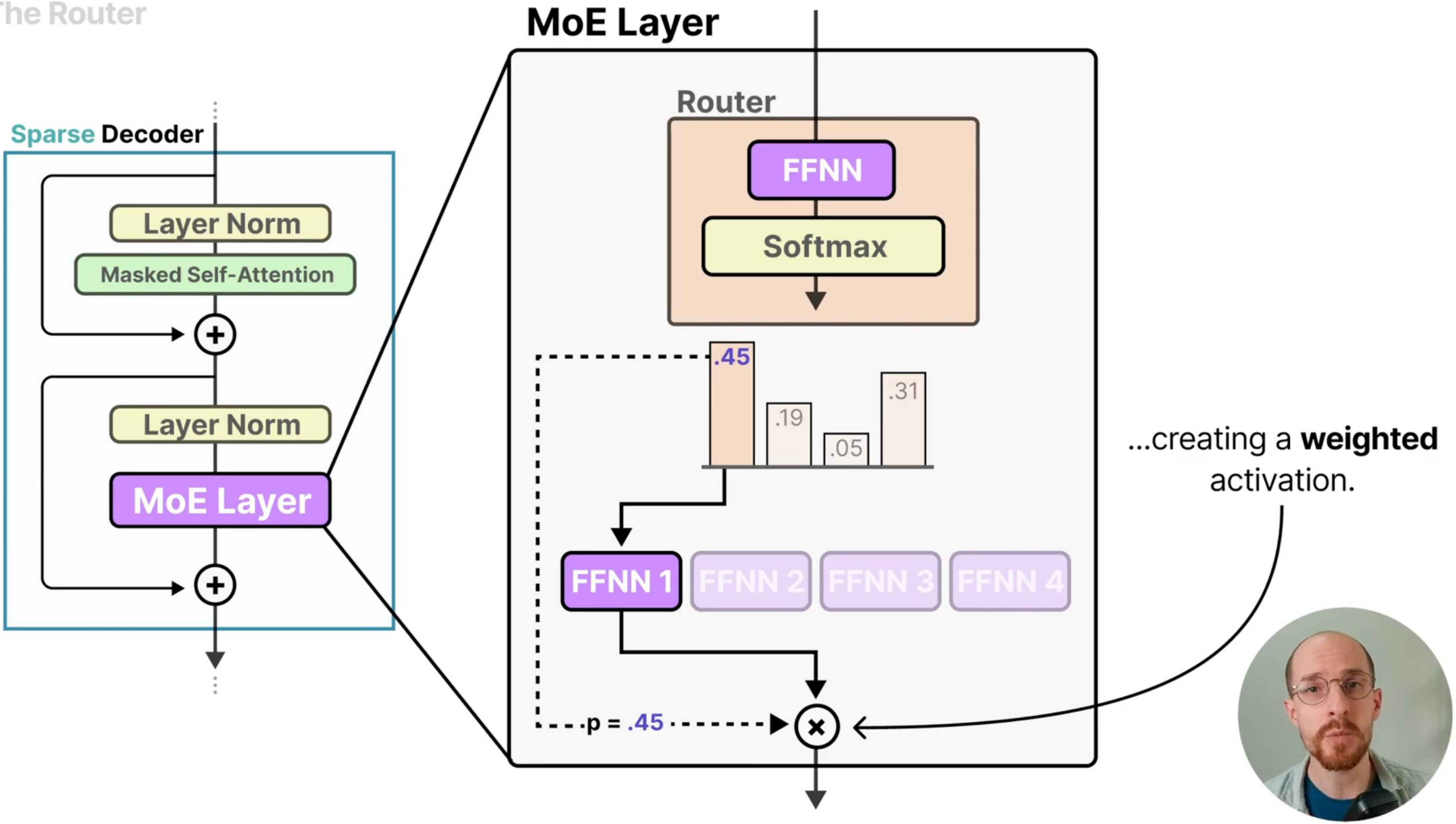
... that are used to **select** and **activate** the best **expert**.

The Router



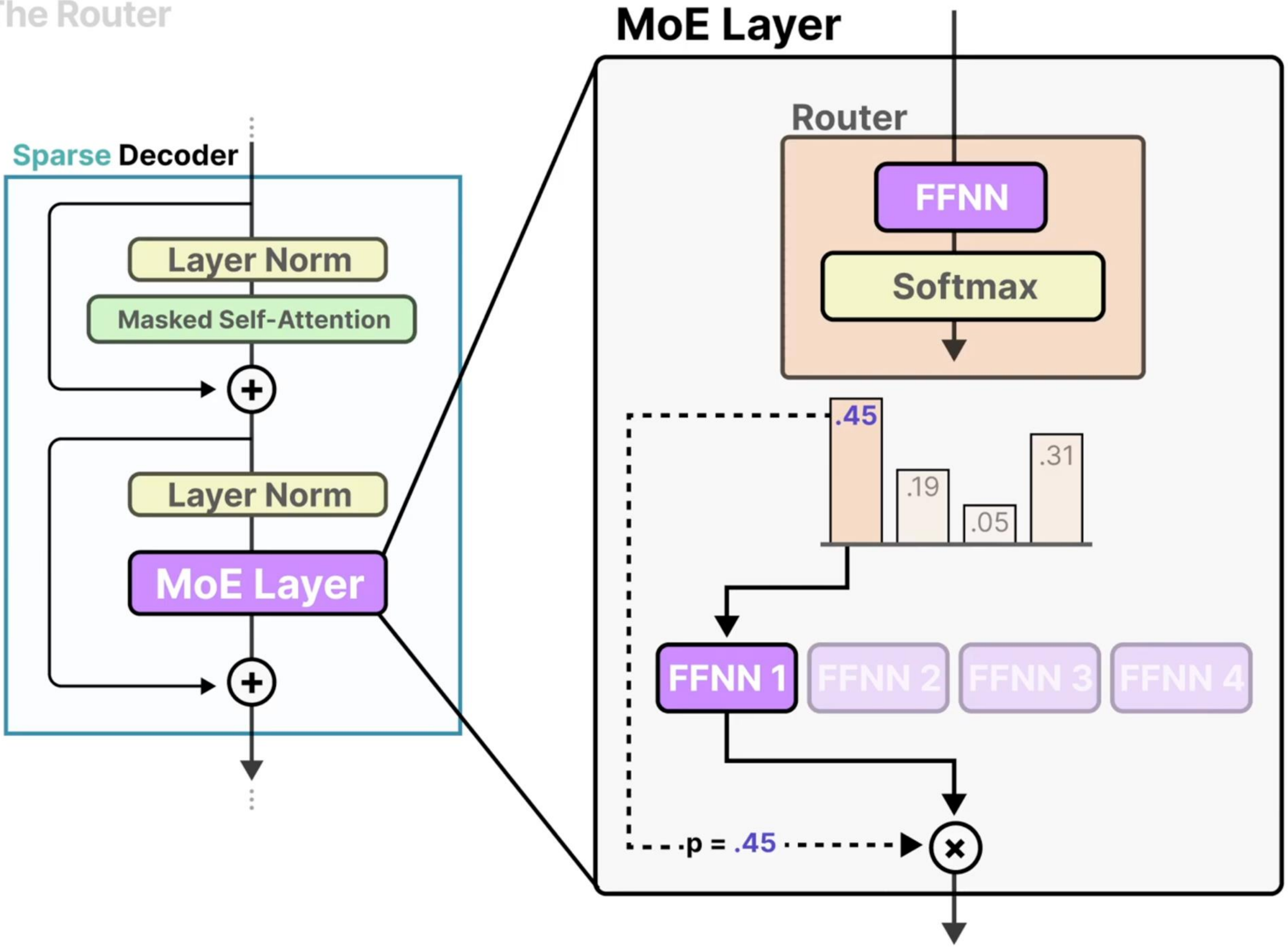
The final output is generated by multiplying the router **probability** with the expert **output**...

The Router



MoE Layer
Sparse Decoder
Layer Norm
Masked Self-Attention
Layer Norm
MoE Layer
Router
FFNN
Softmax
.45
.19
.05
.31
FFNN 1
FFNN 2
FFNN 3
FFNN 4
p = .45
...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).

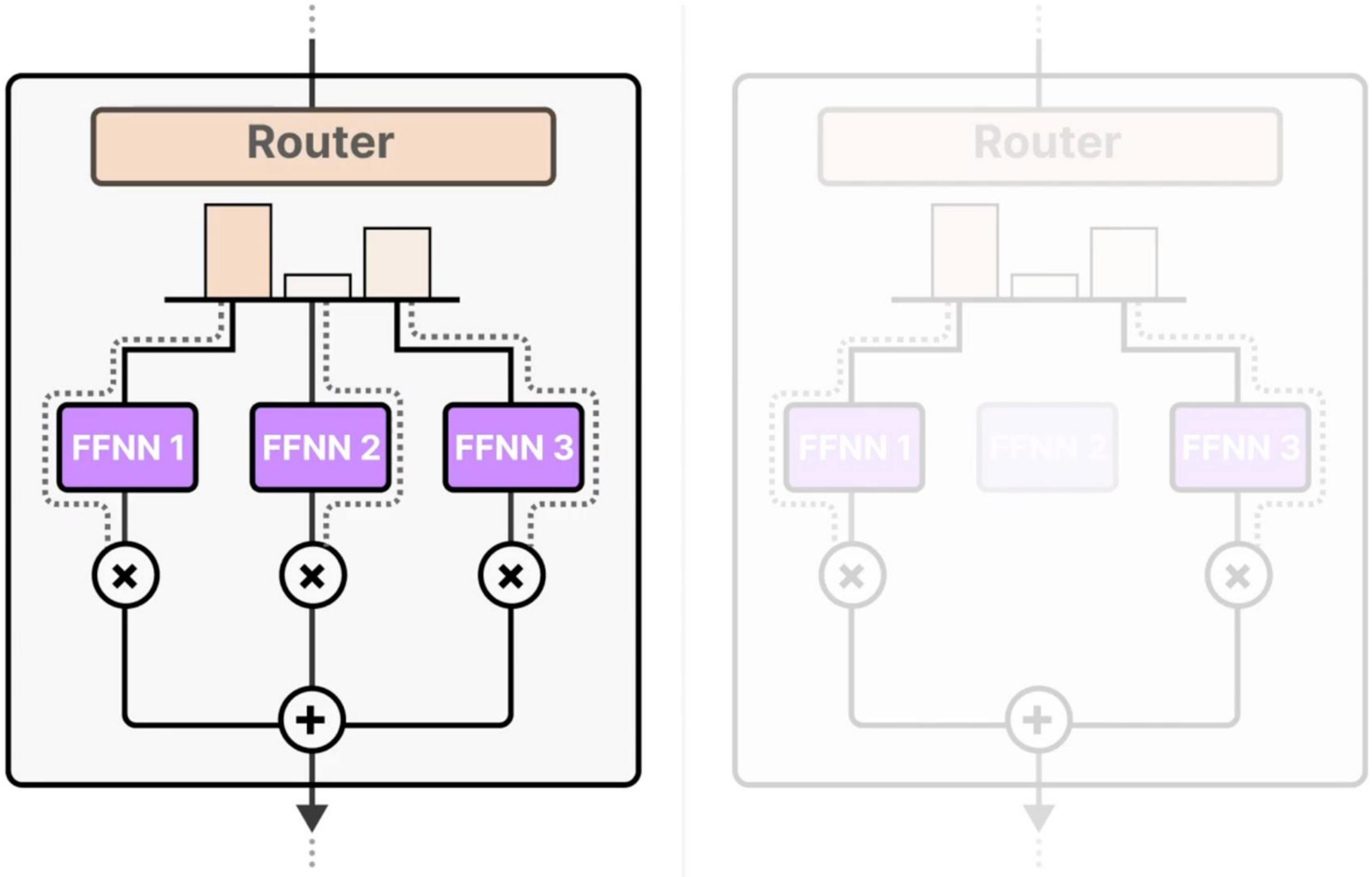
03

Architecture：模型结构

Dense versus Sparse MoE

A given MoE layer comes in two sizes...

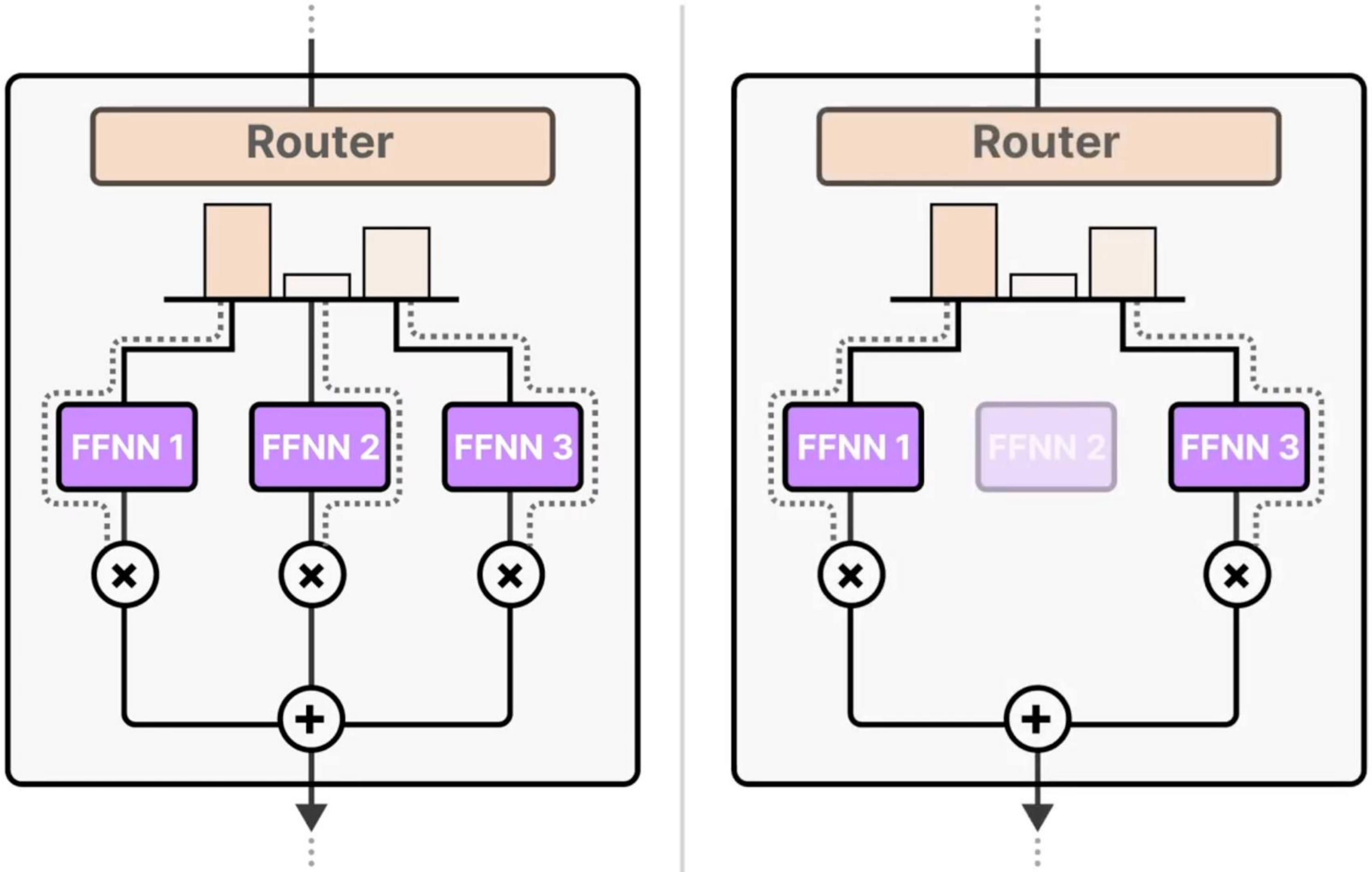
Dense versus Sparse MoE



Dense MoE   **Sparse** MoE

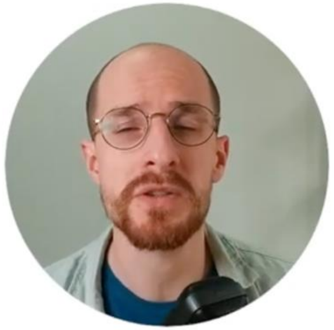
A given MoE layer comes in two sizes, either a **dense**...

Dense versus Sparse MoE

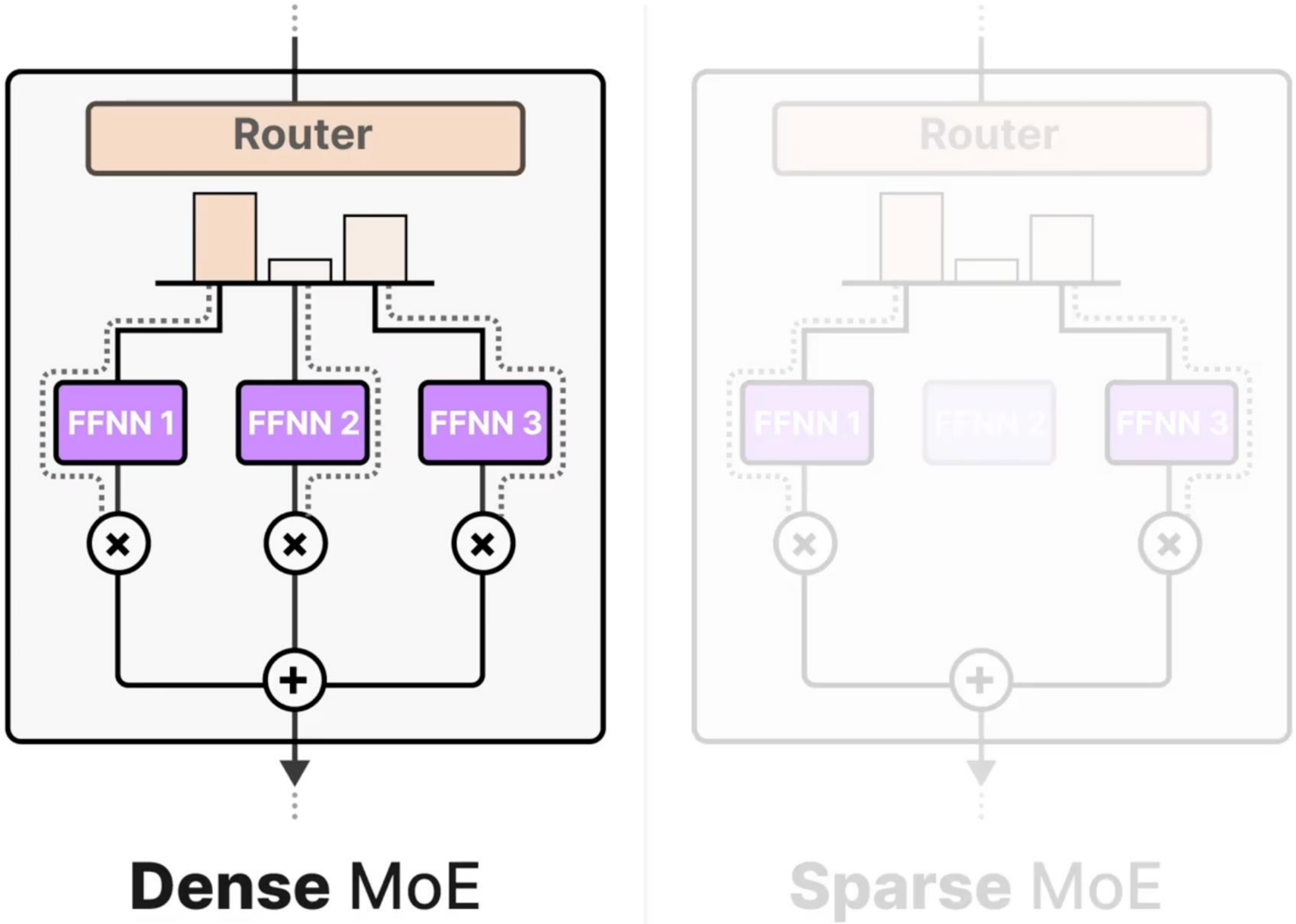


Dense MoE | **Sparse** MoE

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

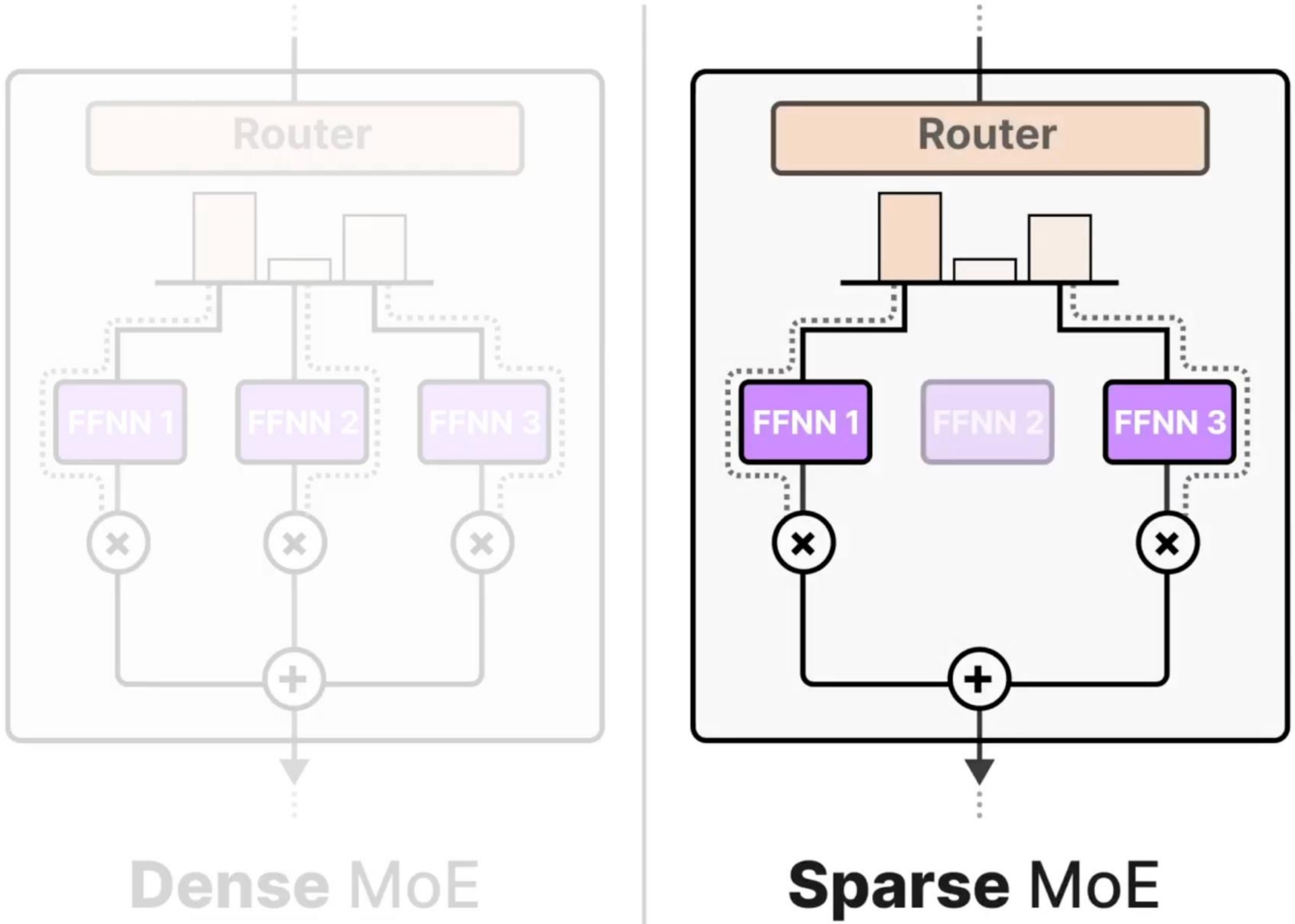


Dense versus Sparse MoE



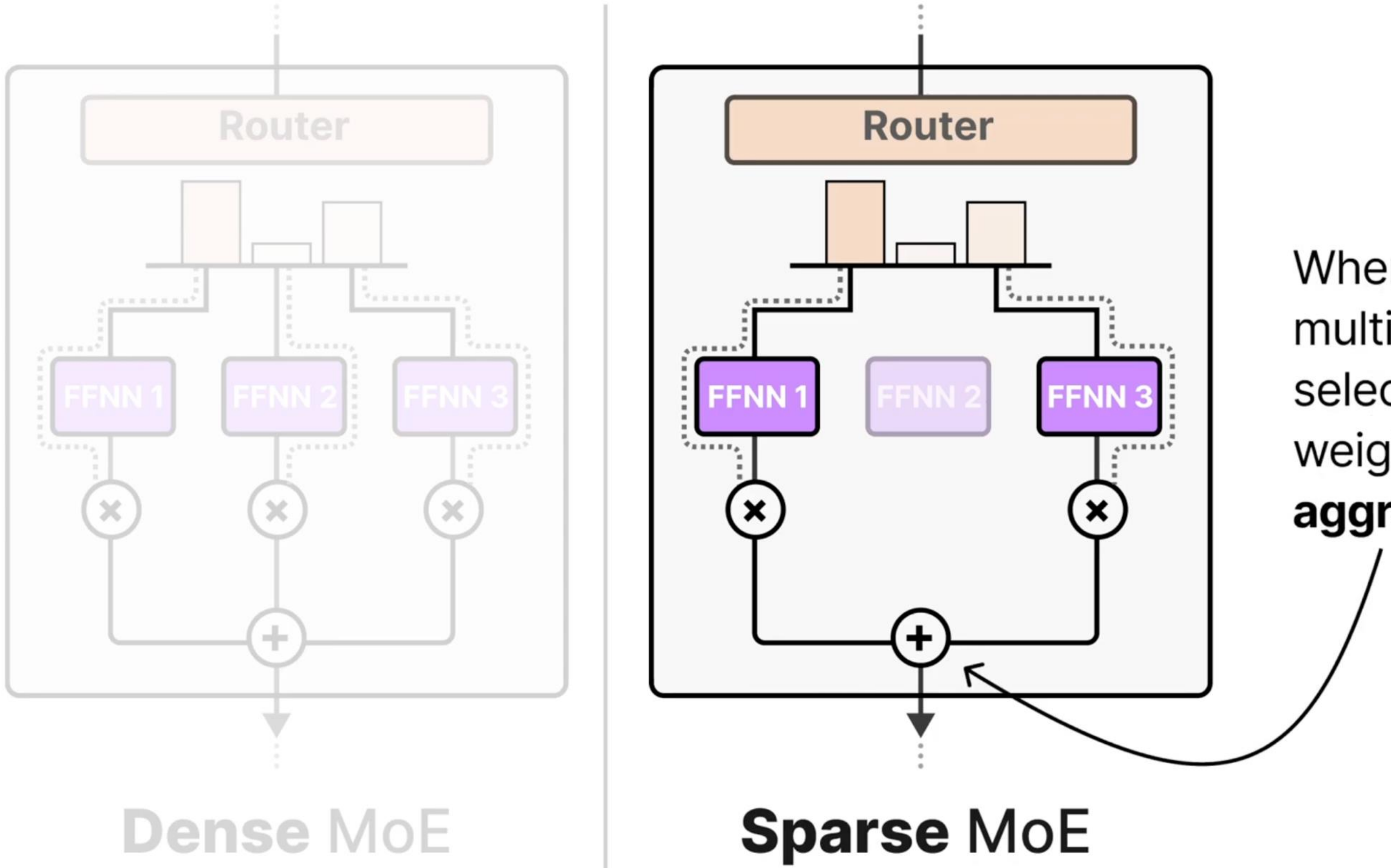
A **Dense MoE** will distribute the tokens across all experts...

Dense versus Sparse MoE

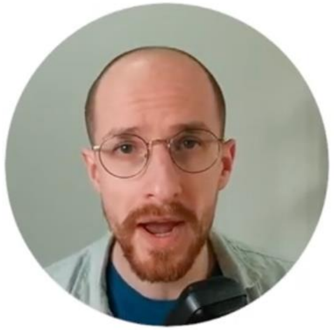


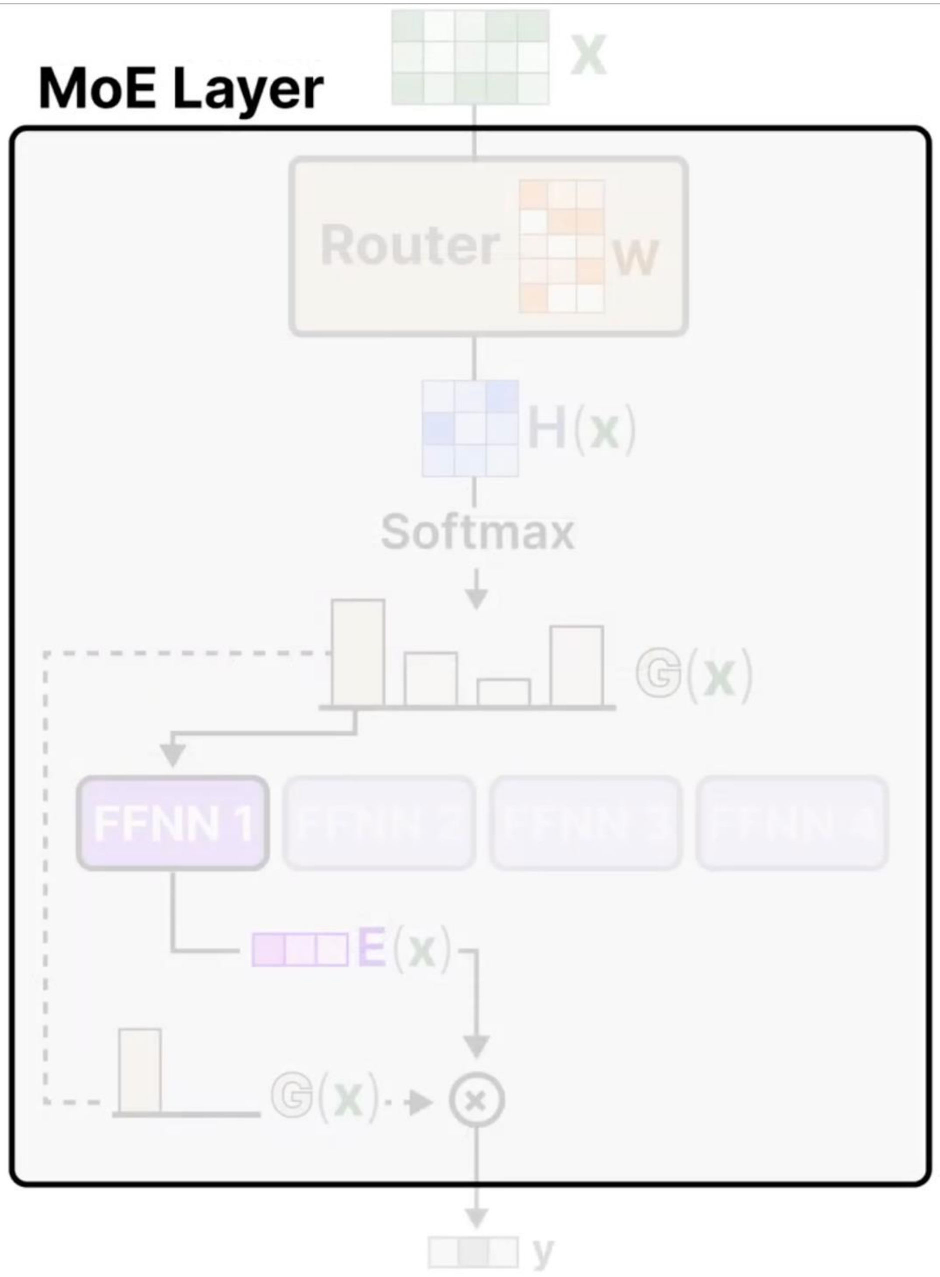
... whereas a **Sparse MoE** will only select a few experts.

Dense versus Sparse MoE

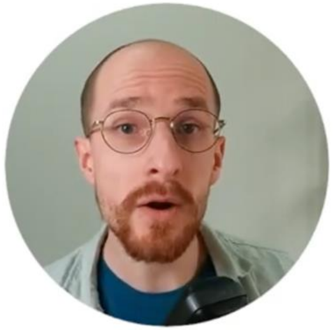


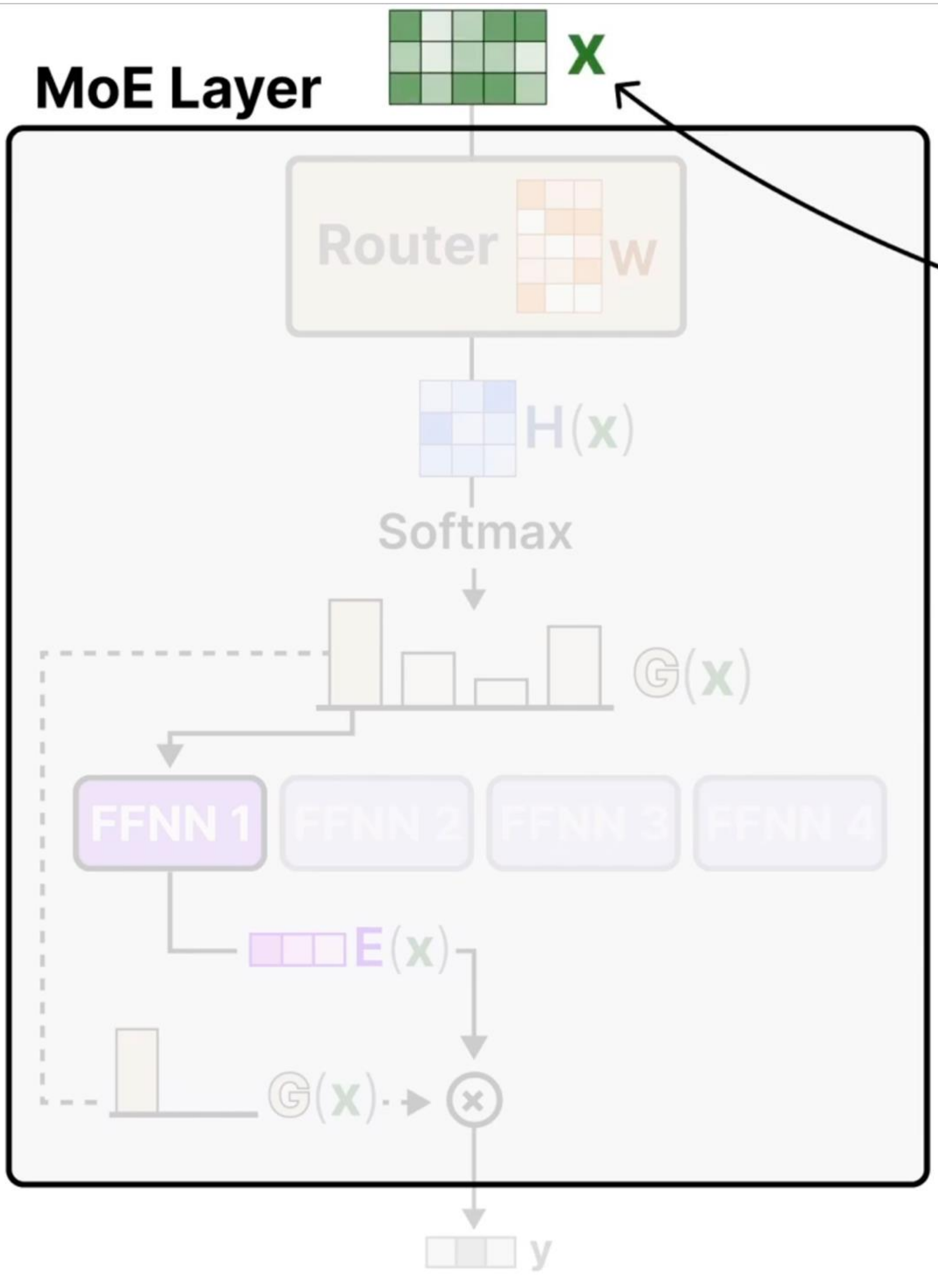
When we have multiple experts selected, their weighted outputs get **aggregated**.



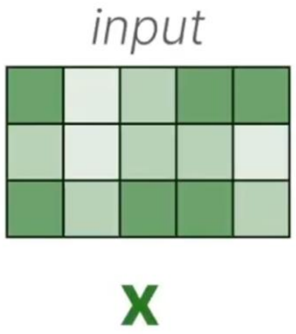


Let's explore how data flows through the **MoE Layer**.

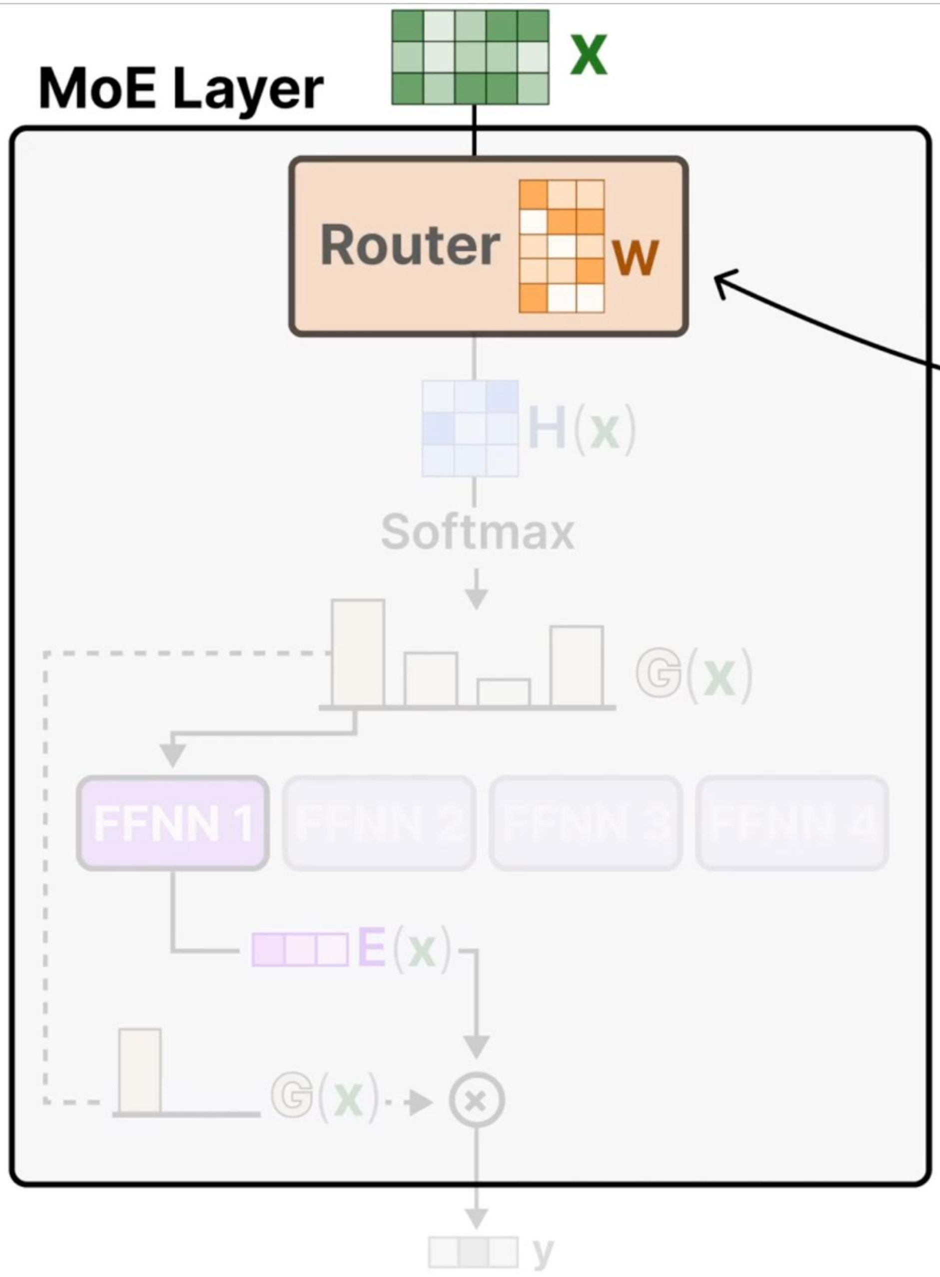


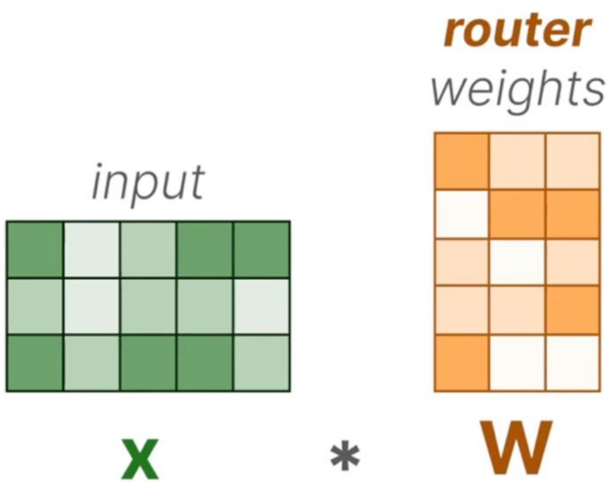


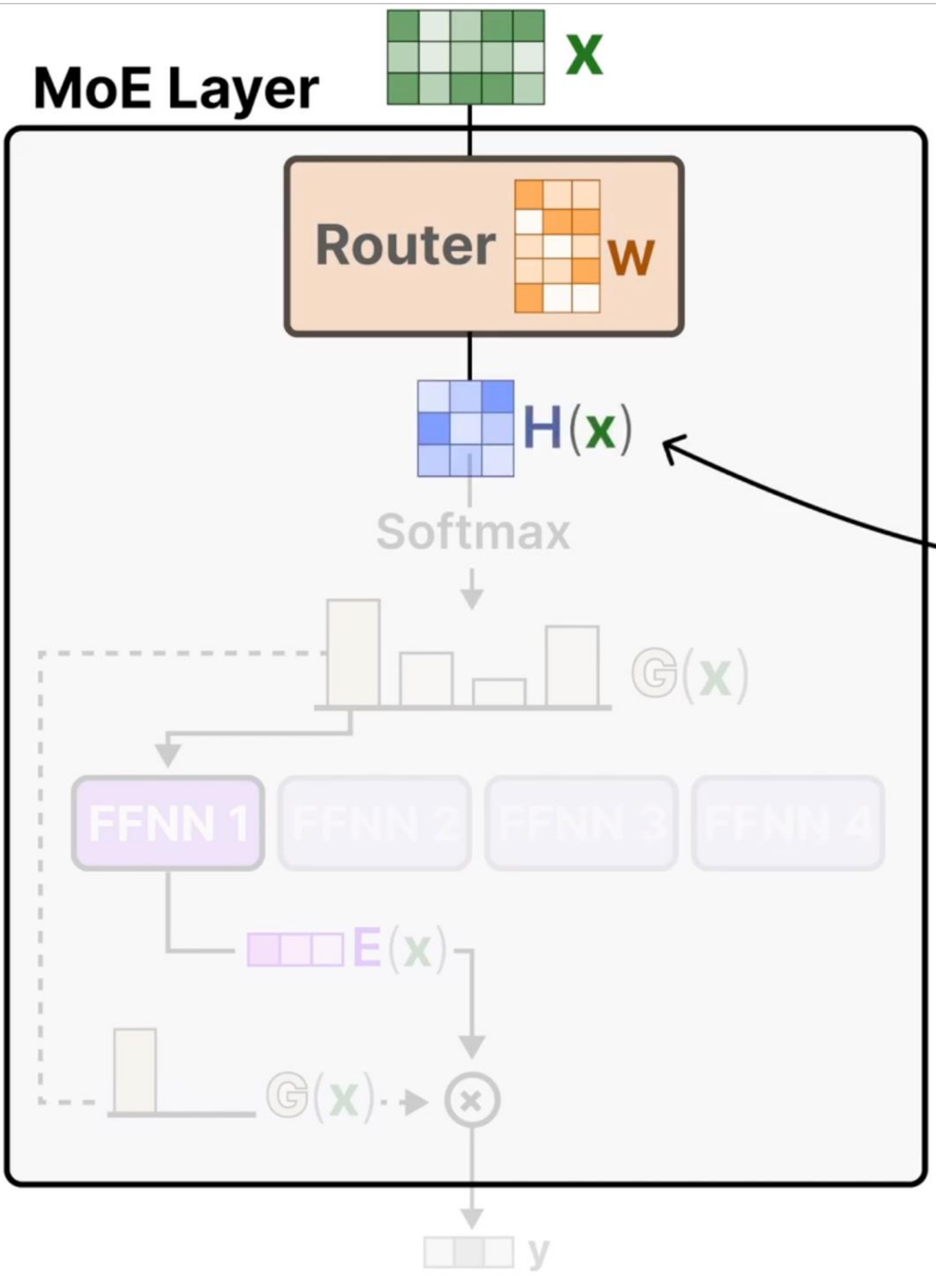
In its most basic form, we multiply the **input** (**x**) ...



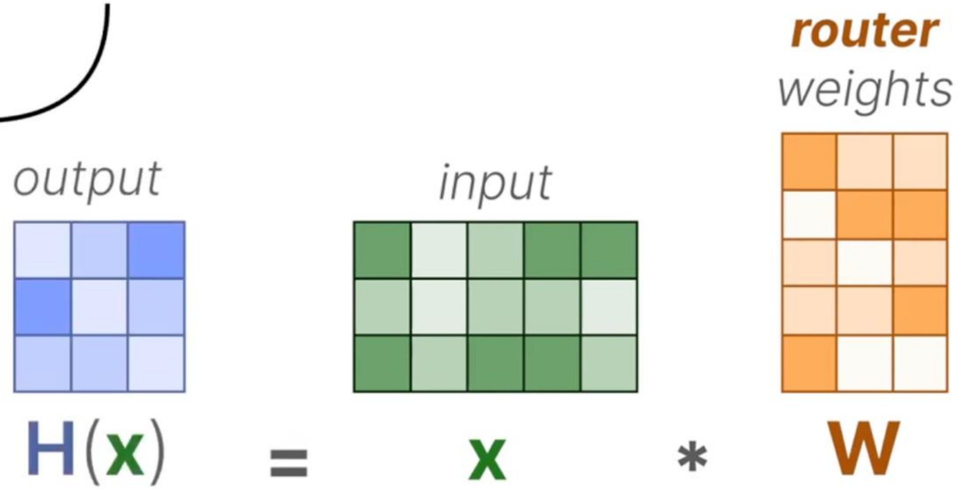
... by the **router weight matrix (W)** ...



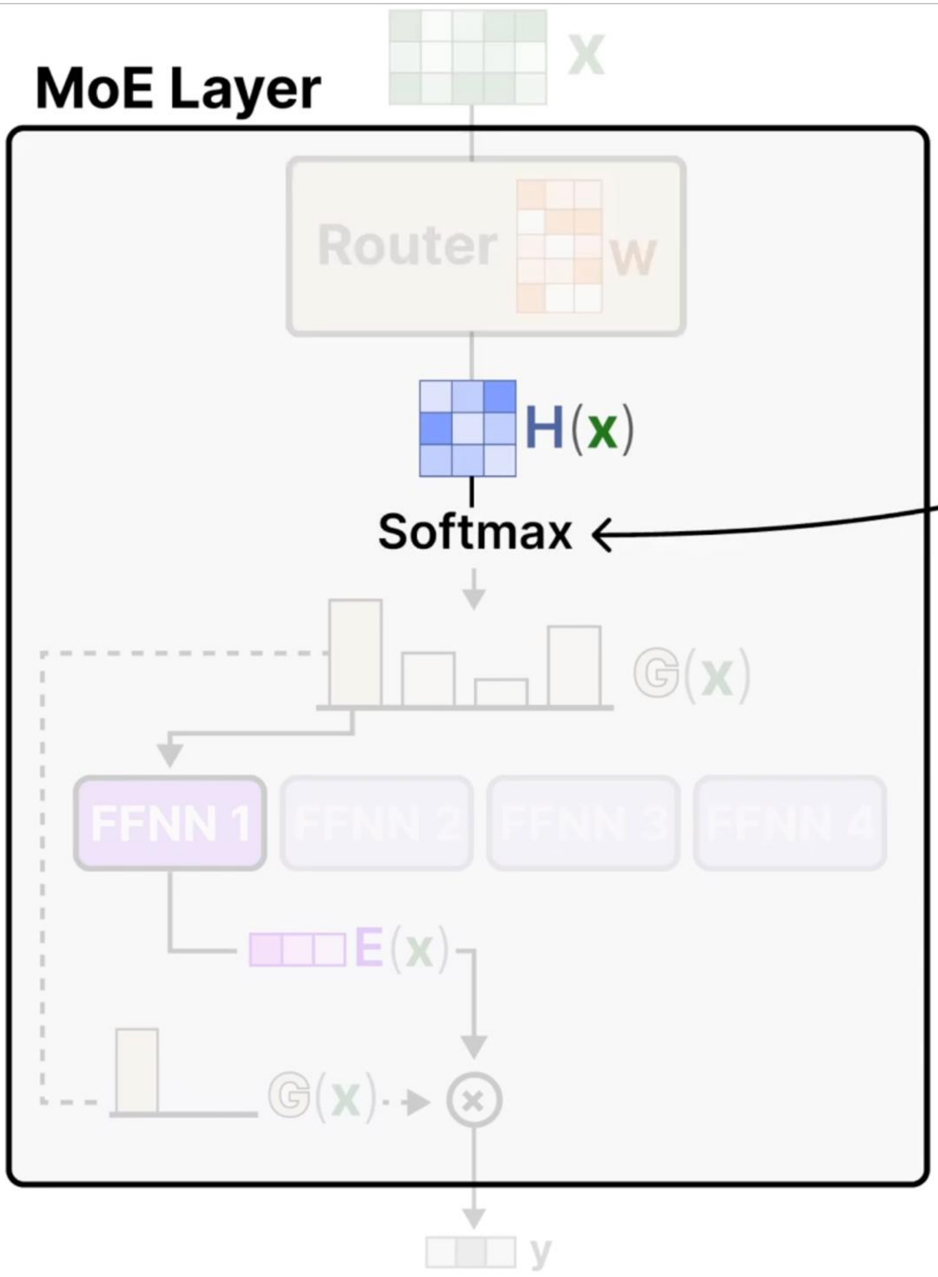


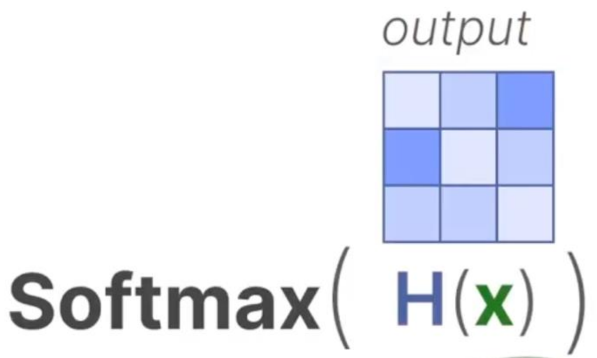


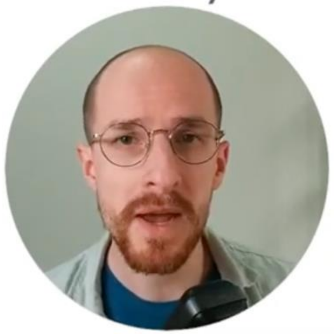
... to create the **output** of the router H(x).

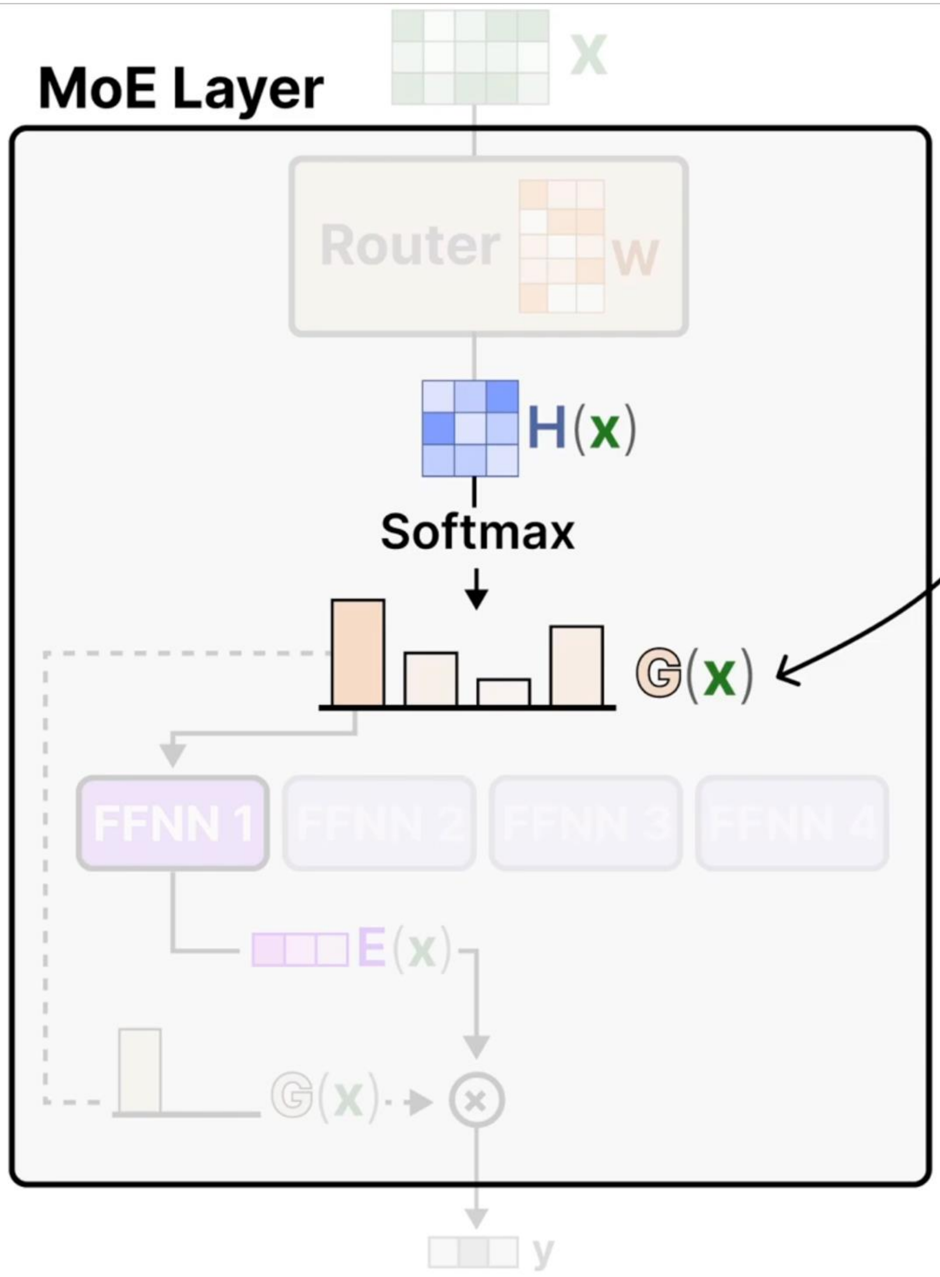


Then, the **softmax** of the output is taken...

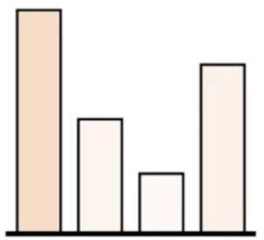


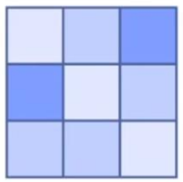




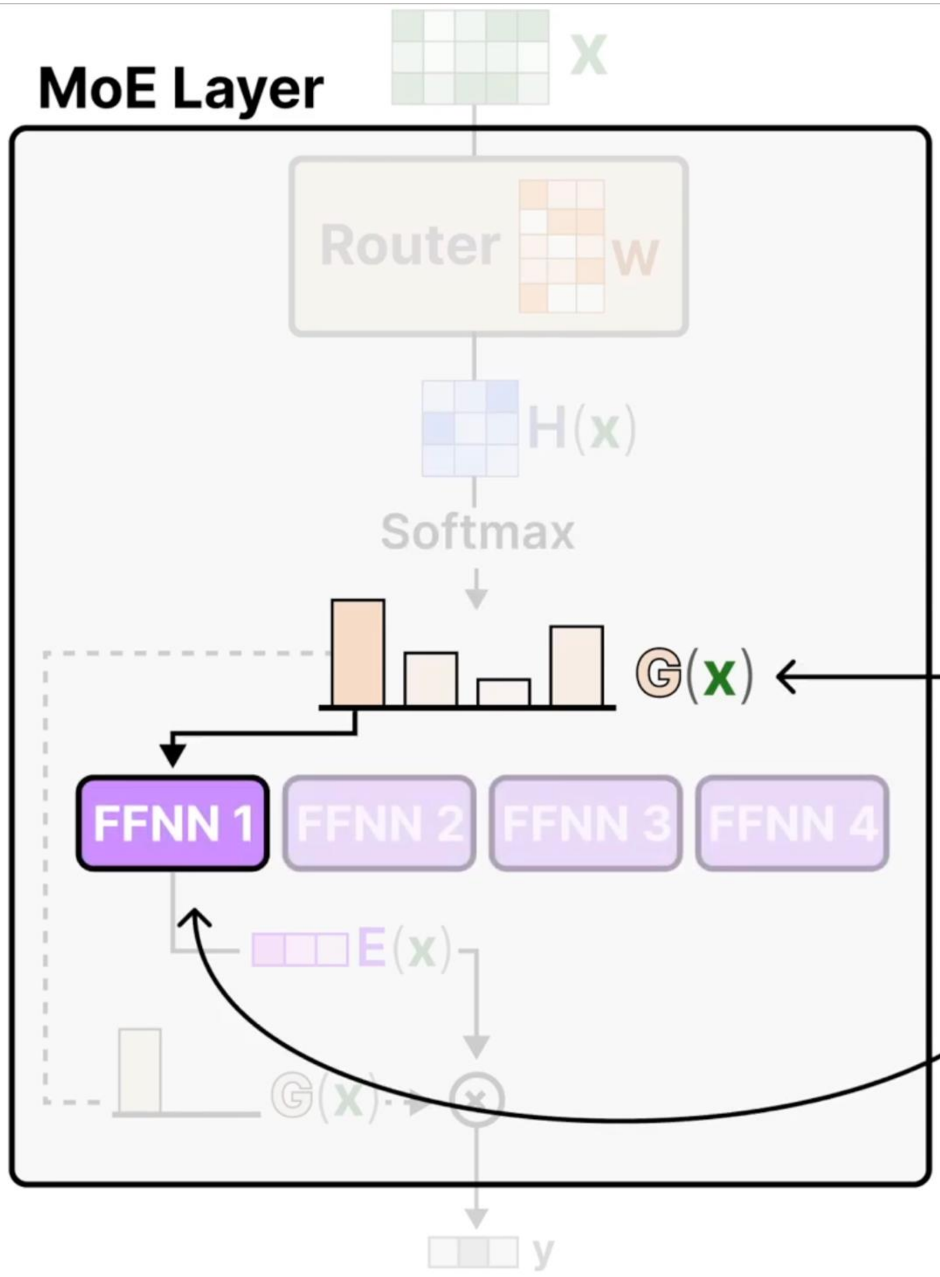


... to create **probabilities**, G(x), one for each expert.

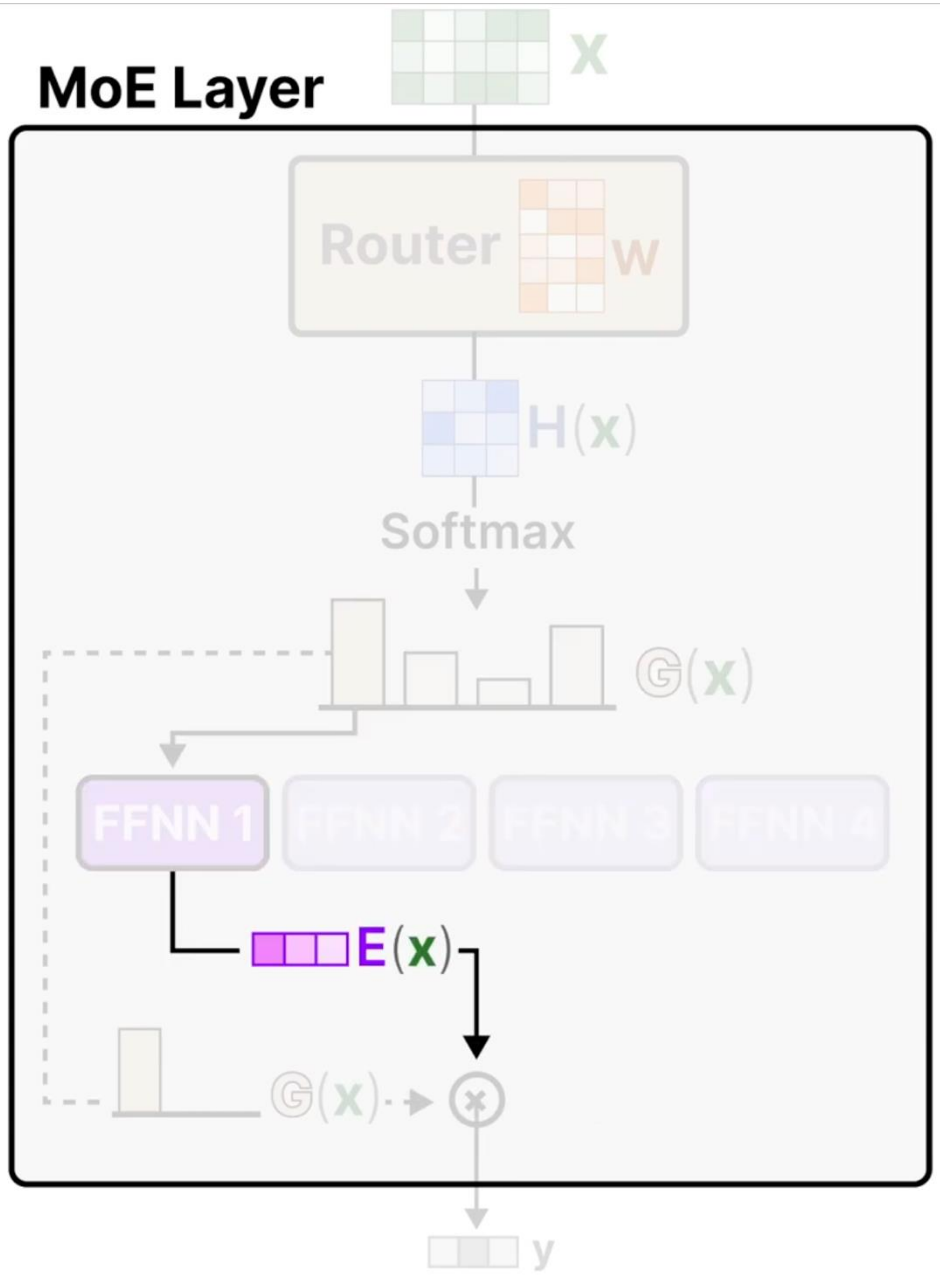




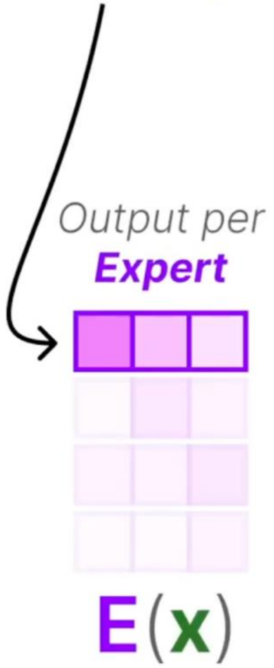
$$G(x) = \text{Softmax}(\ H(x)\ )$$

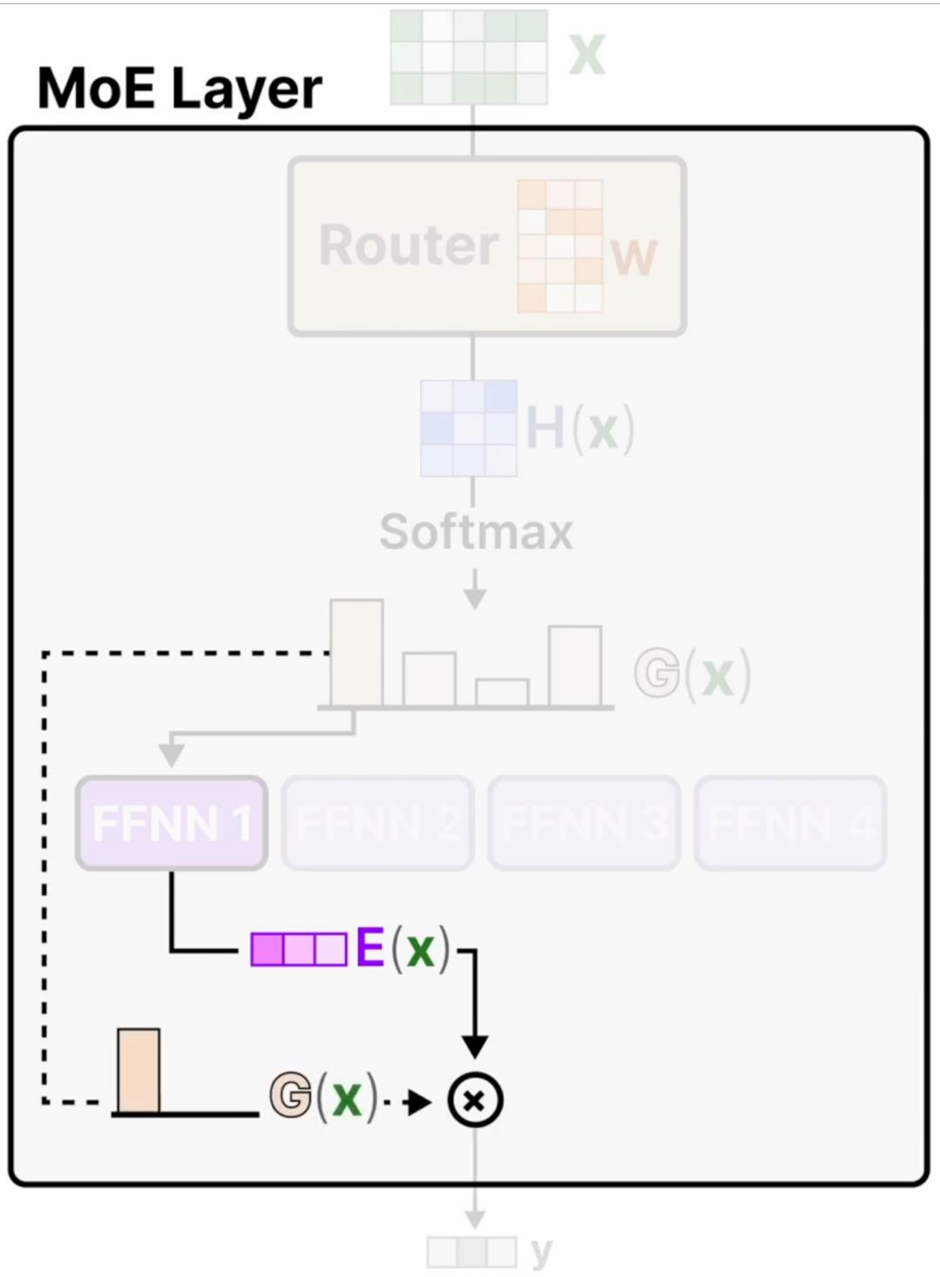


The router uses this **probability distribution** to choose the best matching **expert**.

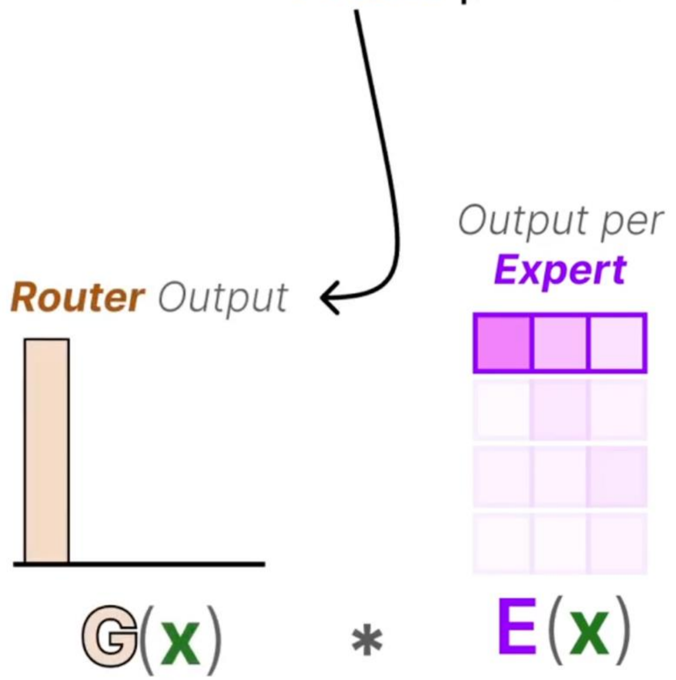


We then take the output per selected **expert**...

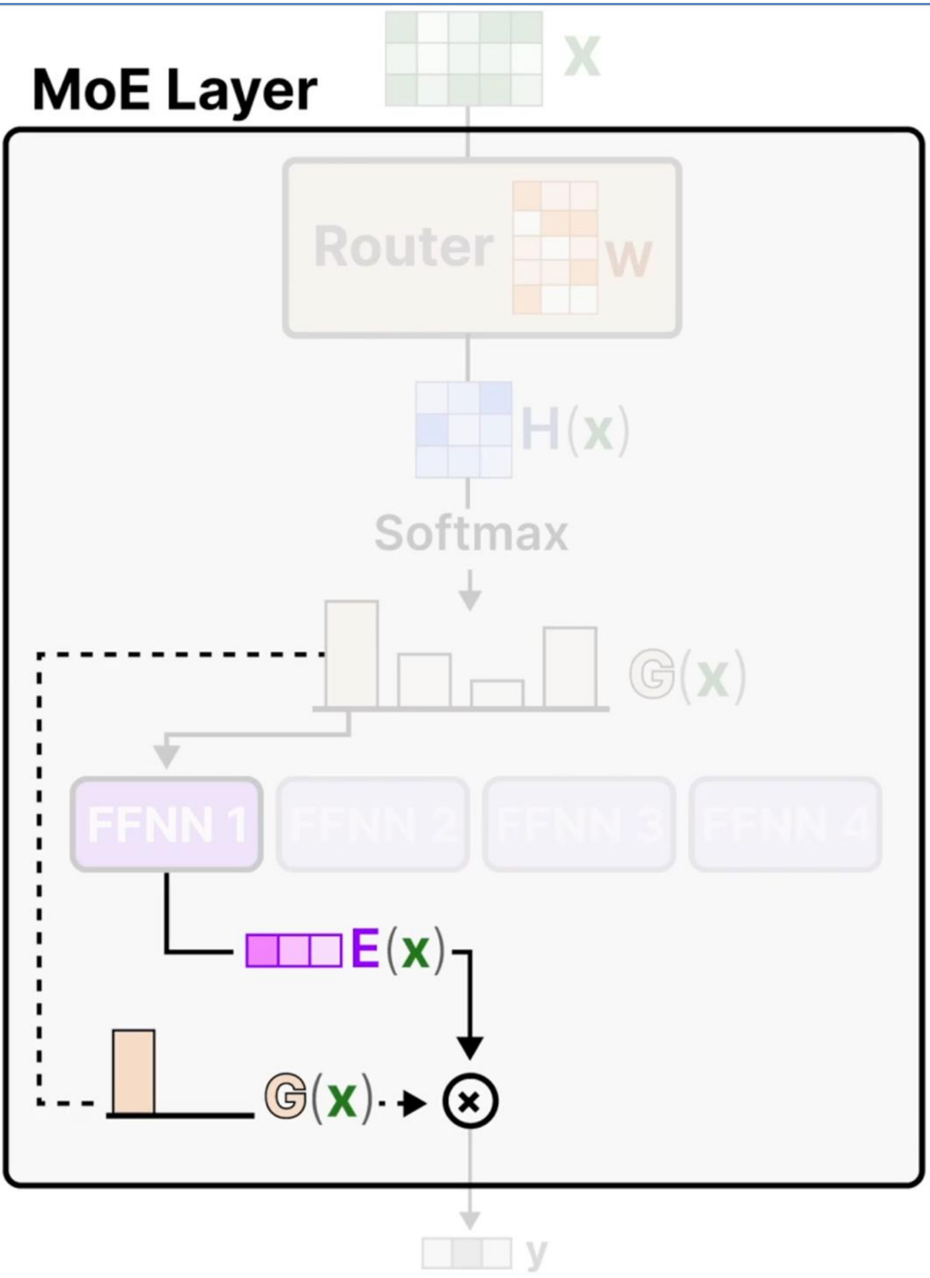




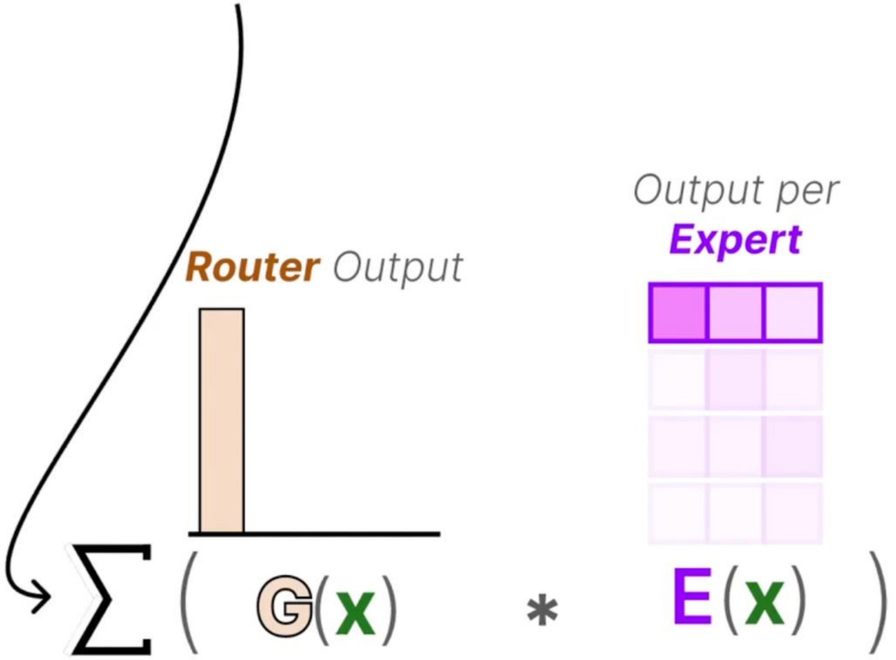
... and multiply that with the **router** probabilities.

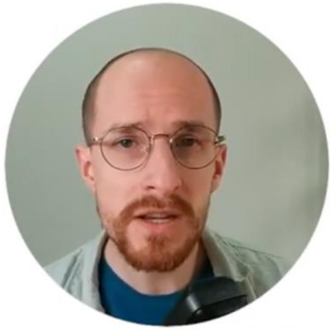


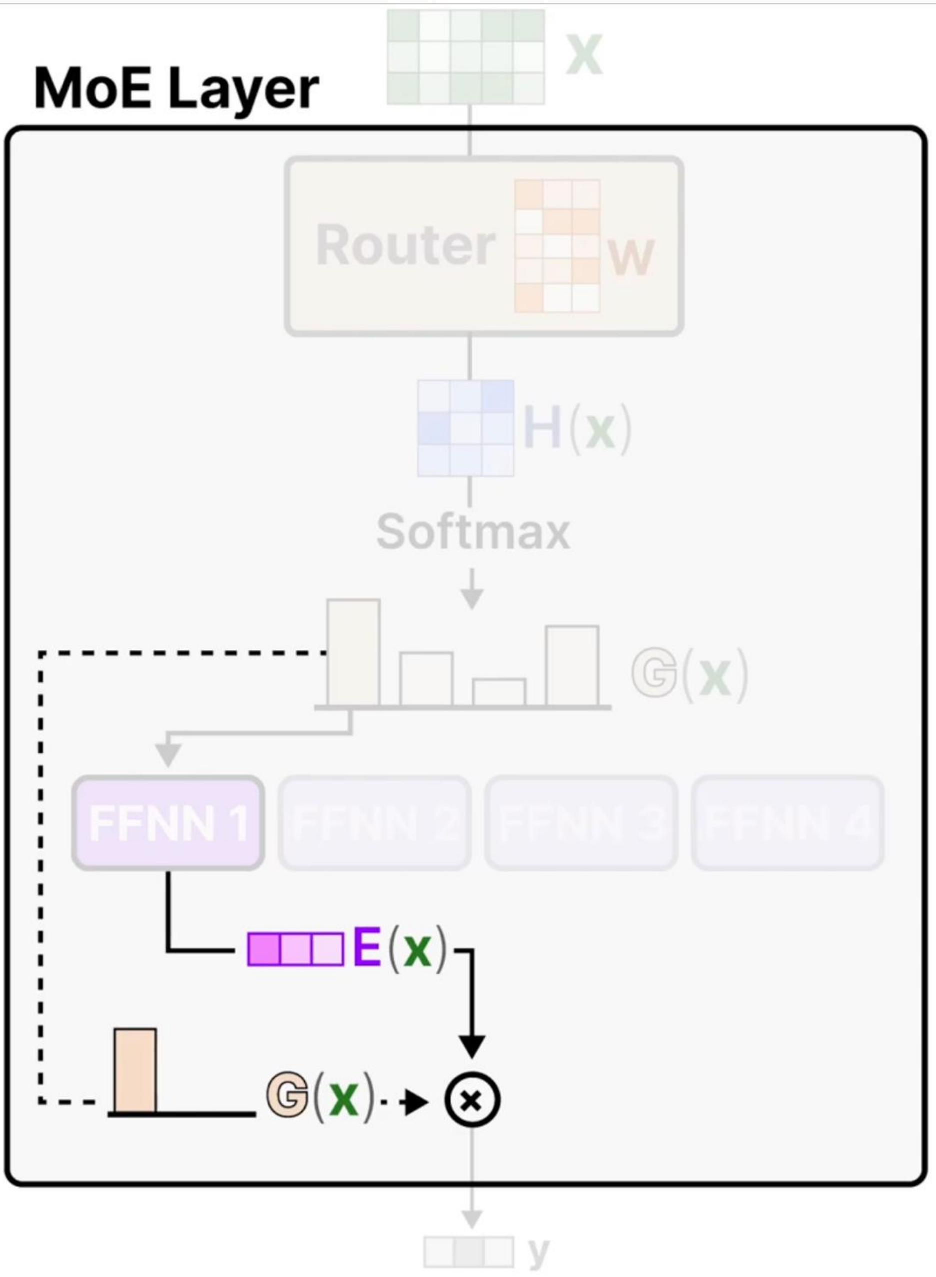
Selection of Experts



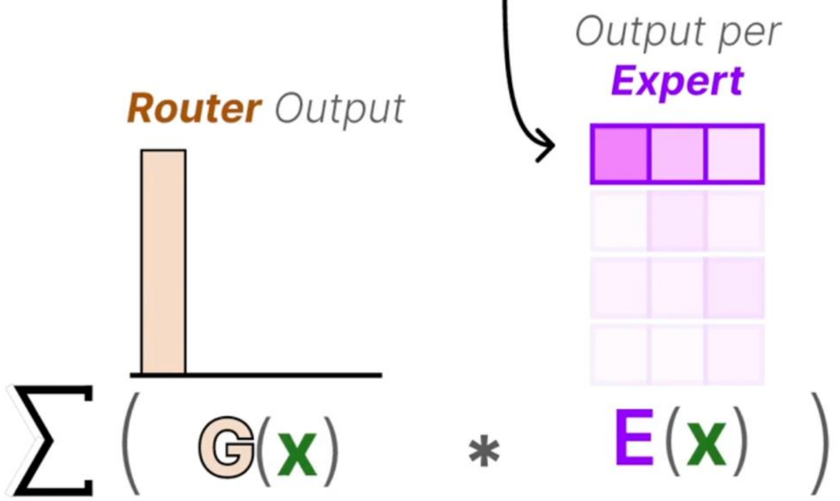
We do this for **every expert** selected.

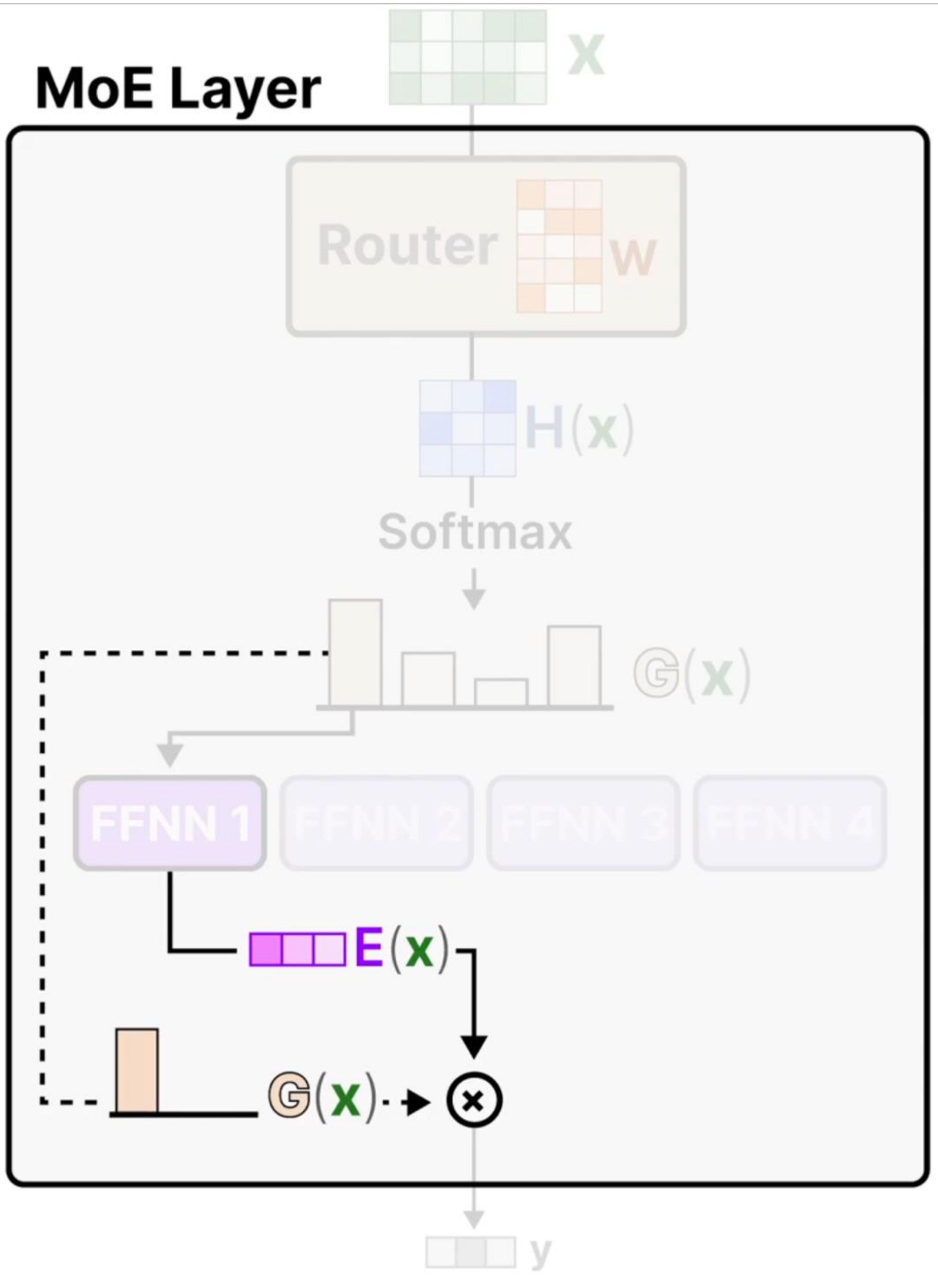




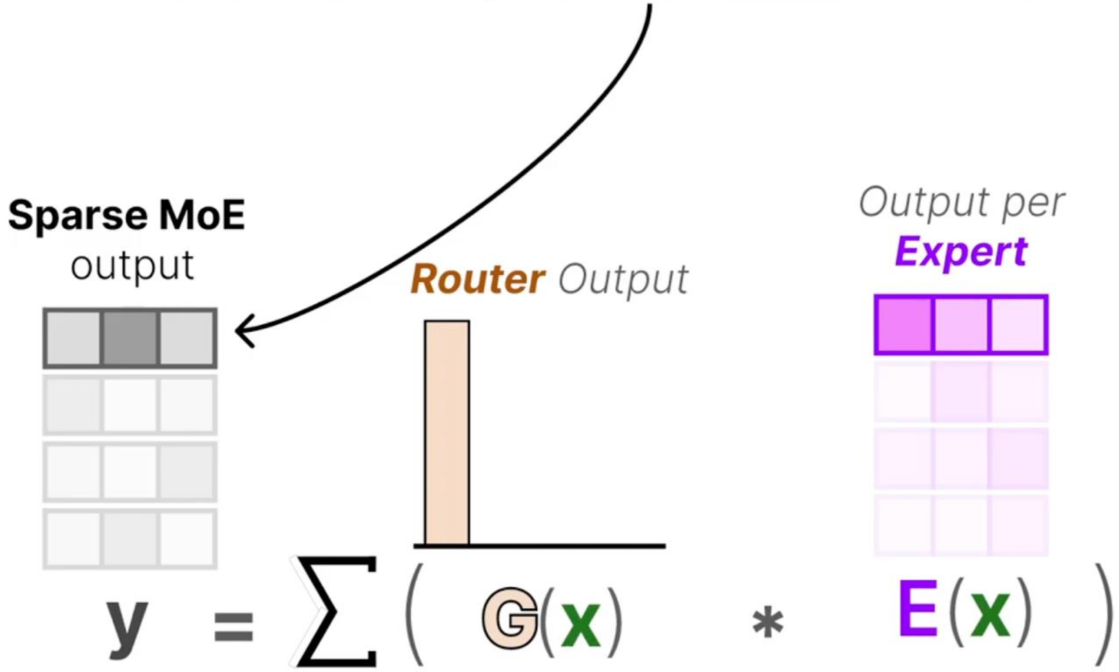


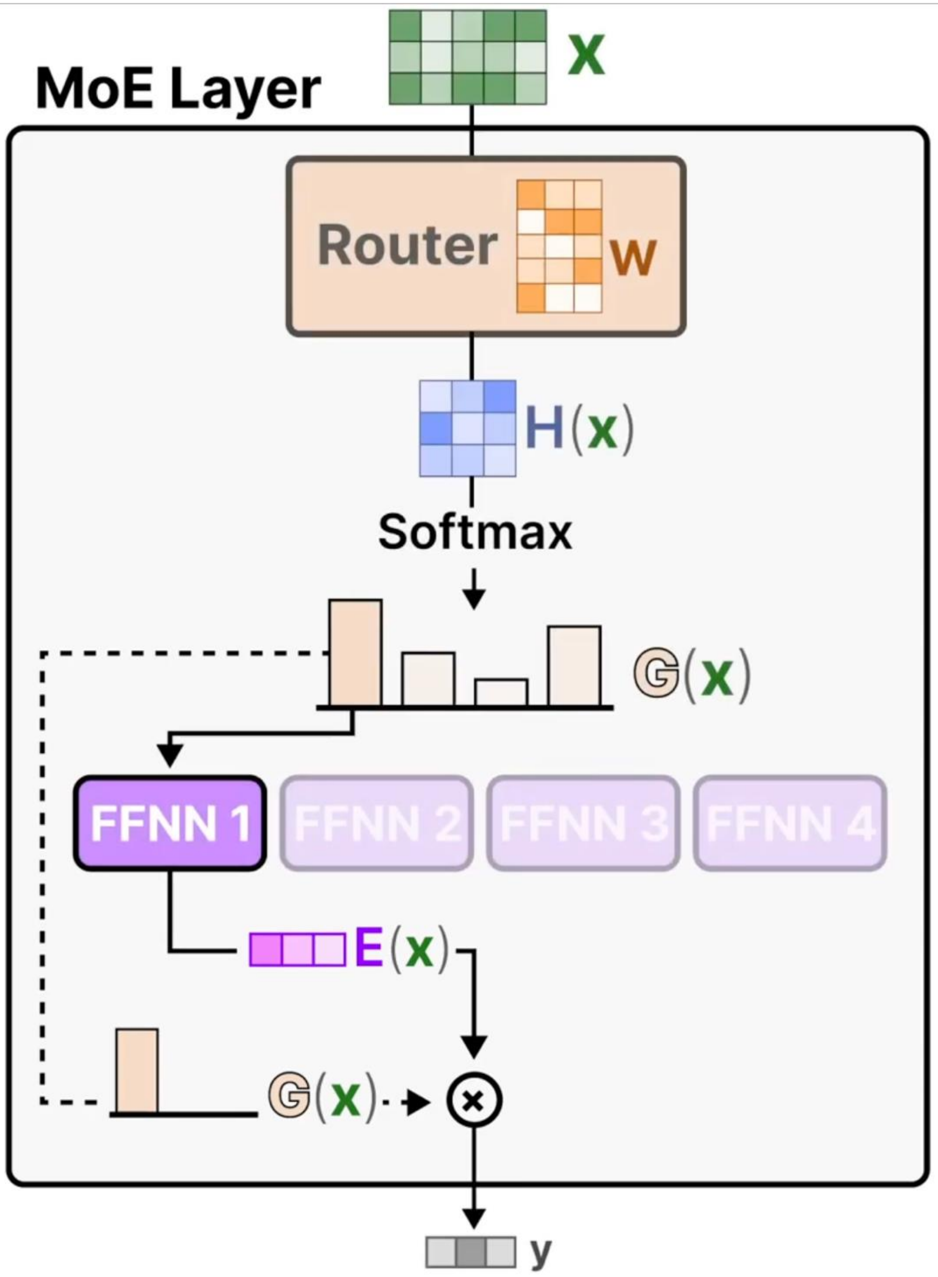
Since we choose only **one expert**, we are only doing this calculation once.





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



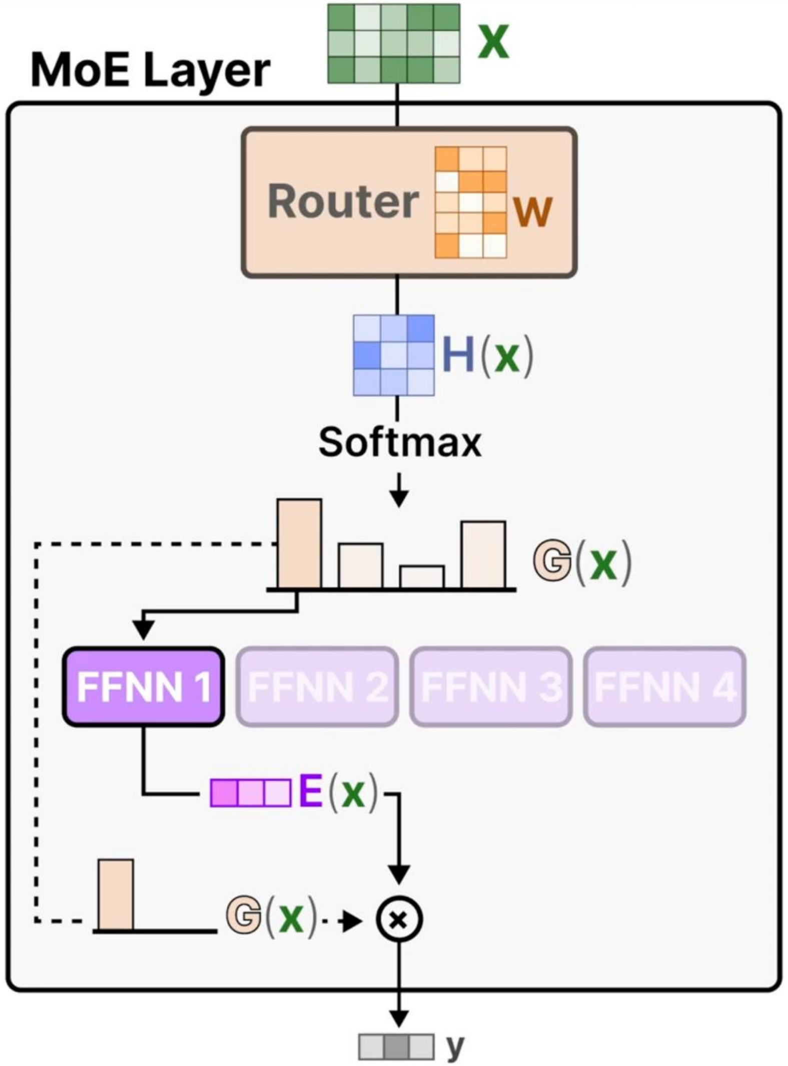


And that is how a typical **MoE layer** processes the data.

04

Load Balancing：均衡负载

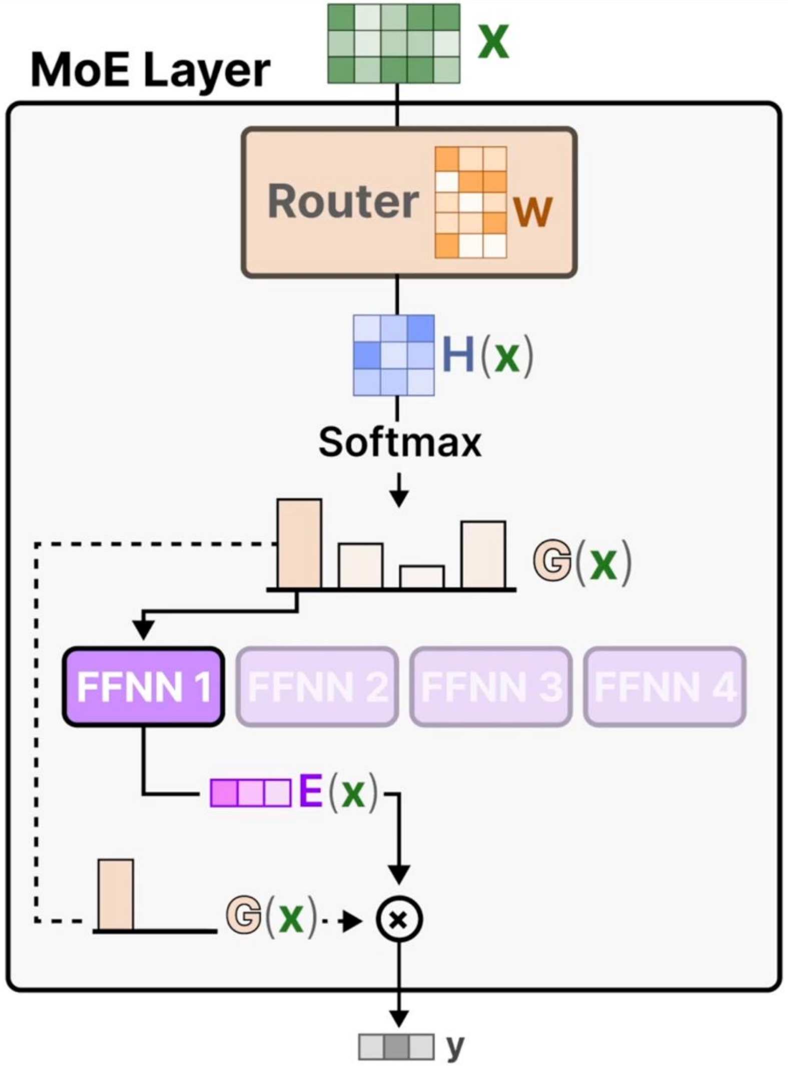
Load Balancing：均衡负载



The Complexity of Routing

Seems straightforward, right?

Load Balancing：均衡负载

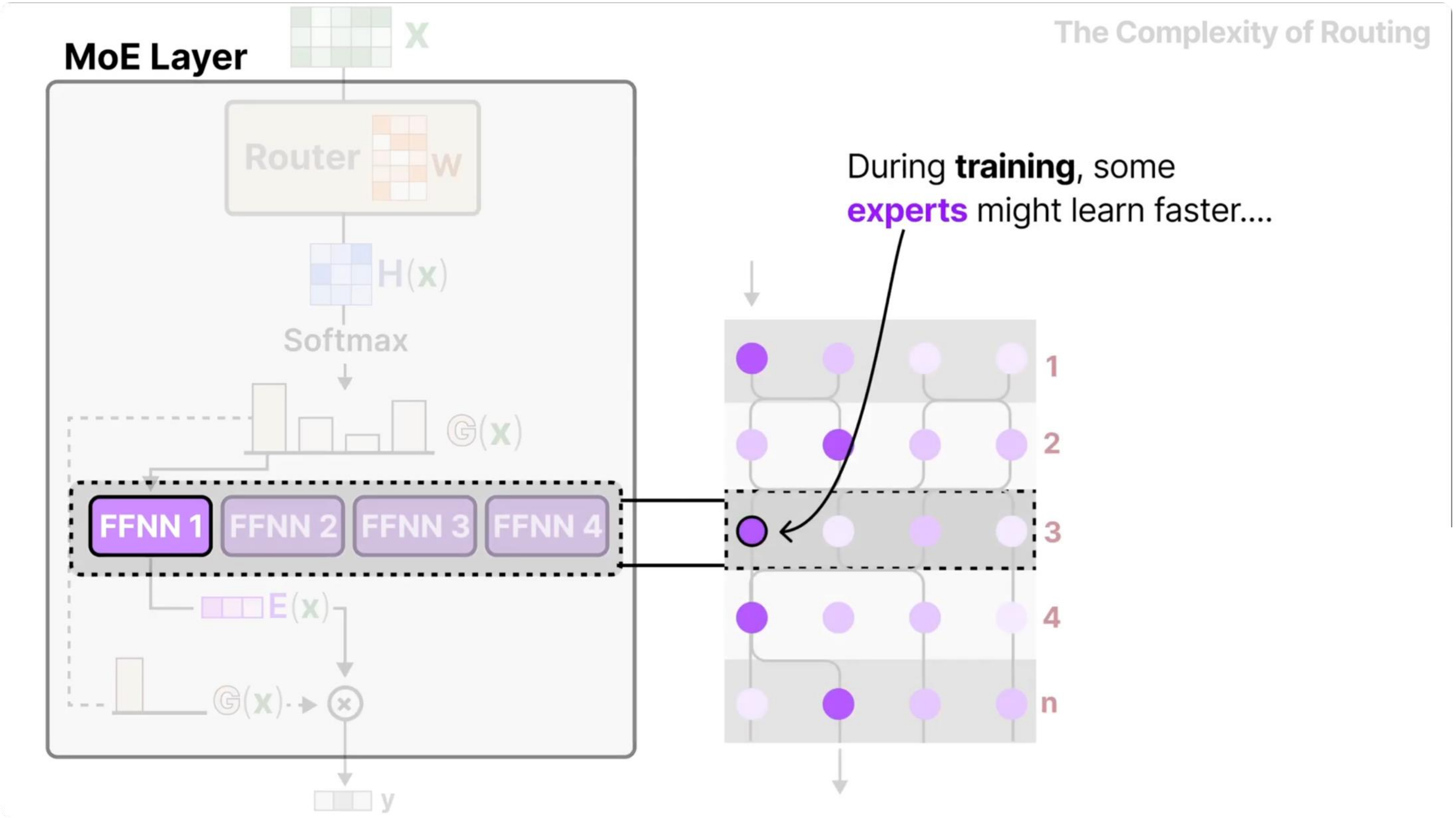


The Complexity of Routing

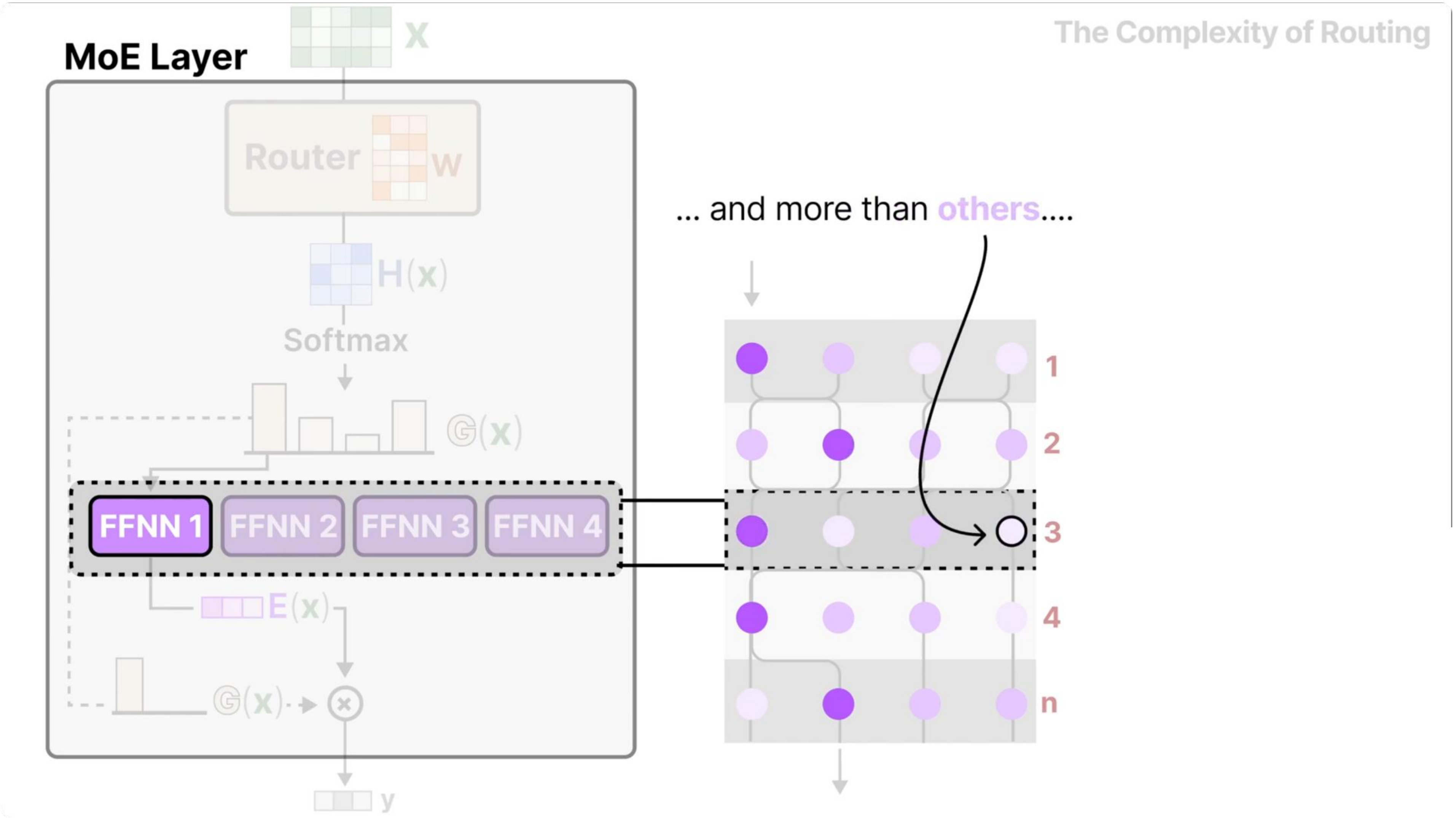
Seems straightforward, right?

Well, there is one big **disadvantage**...

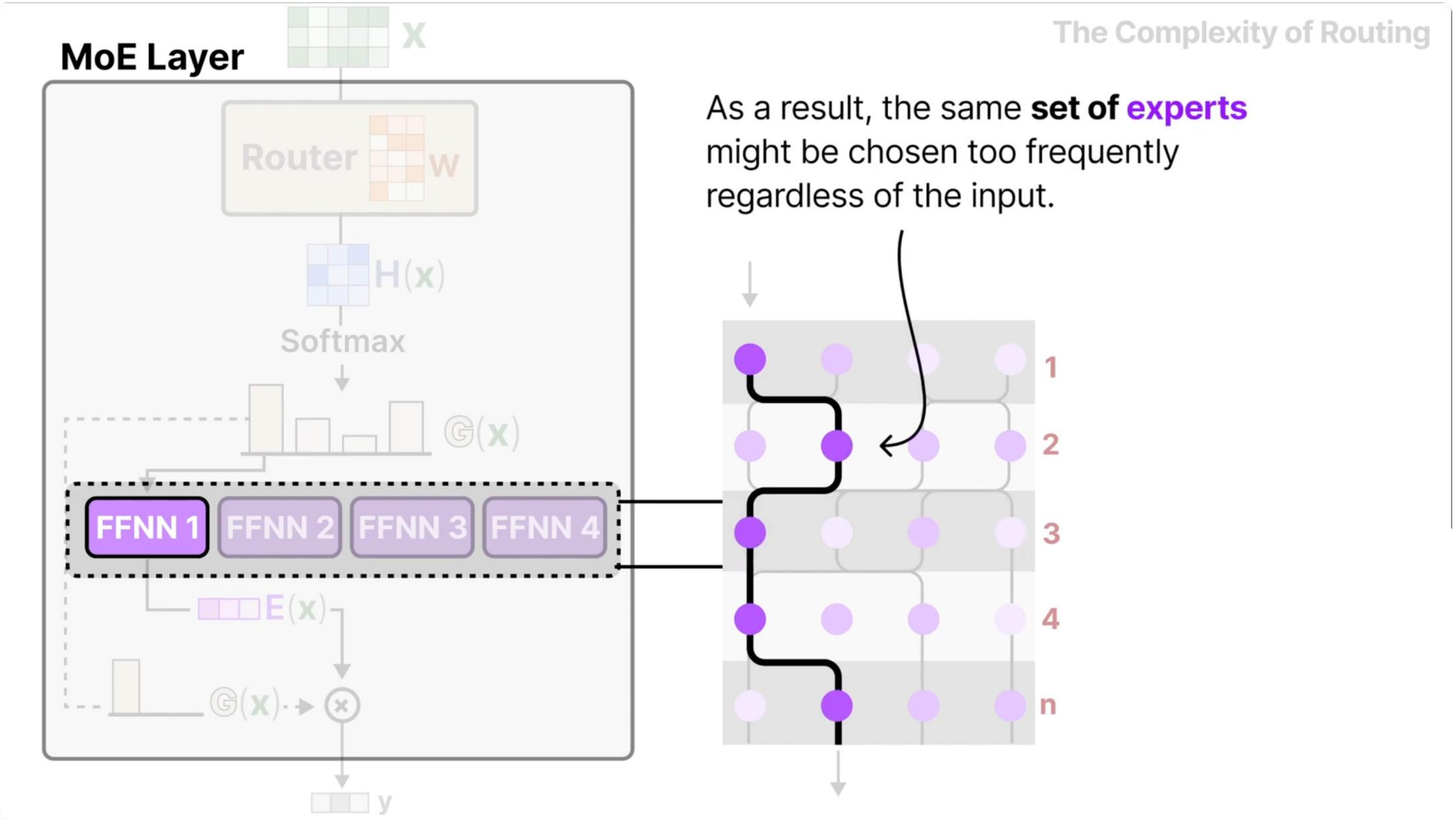
Load Balancing：均衡负载



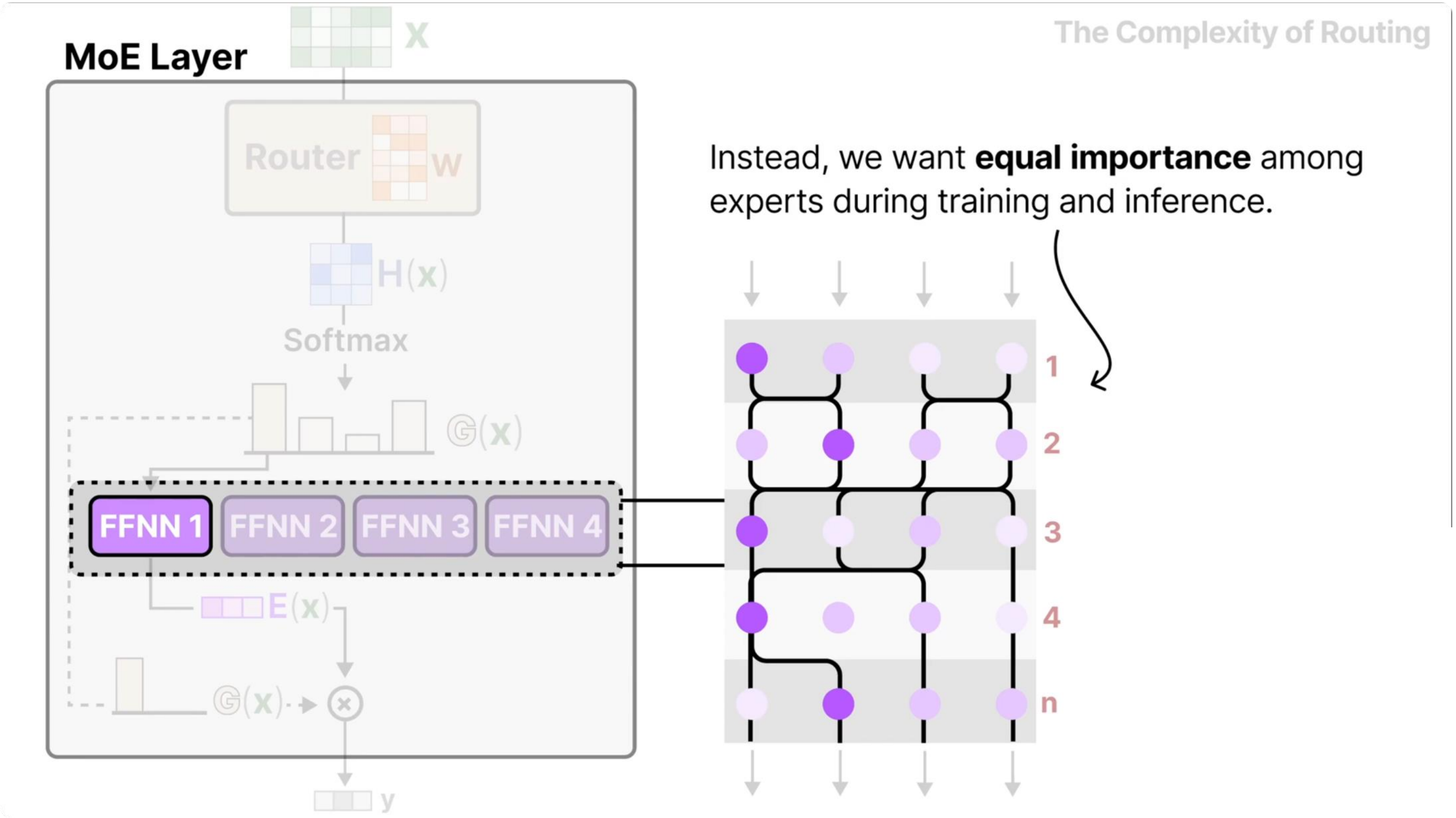
Load Balancing：均衡负载



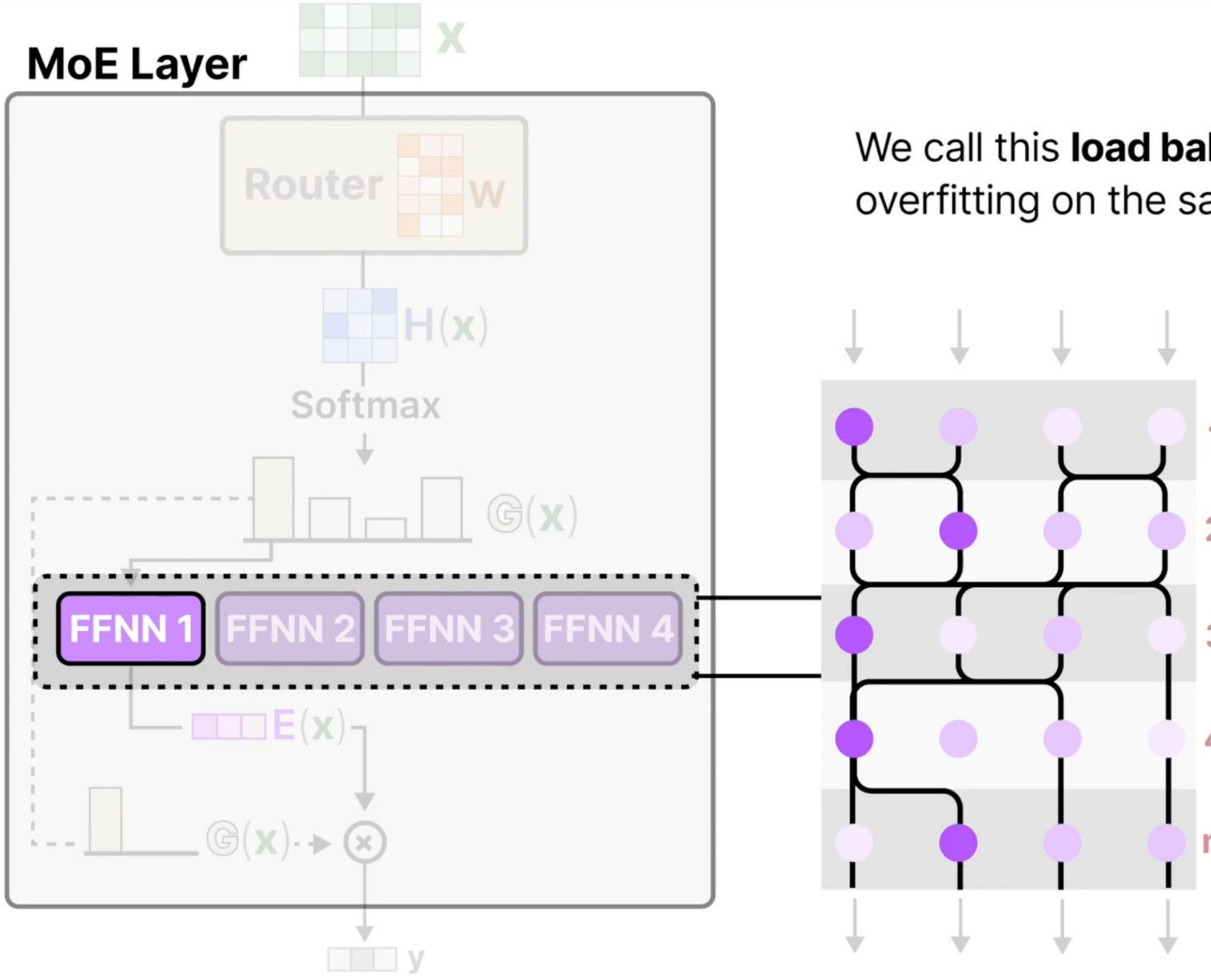
Load Balancing：均衡负载



Load Balancing：均衡负载



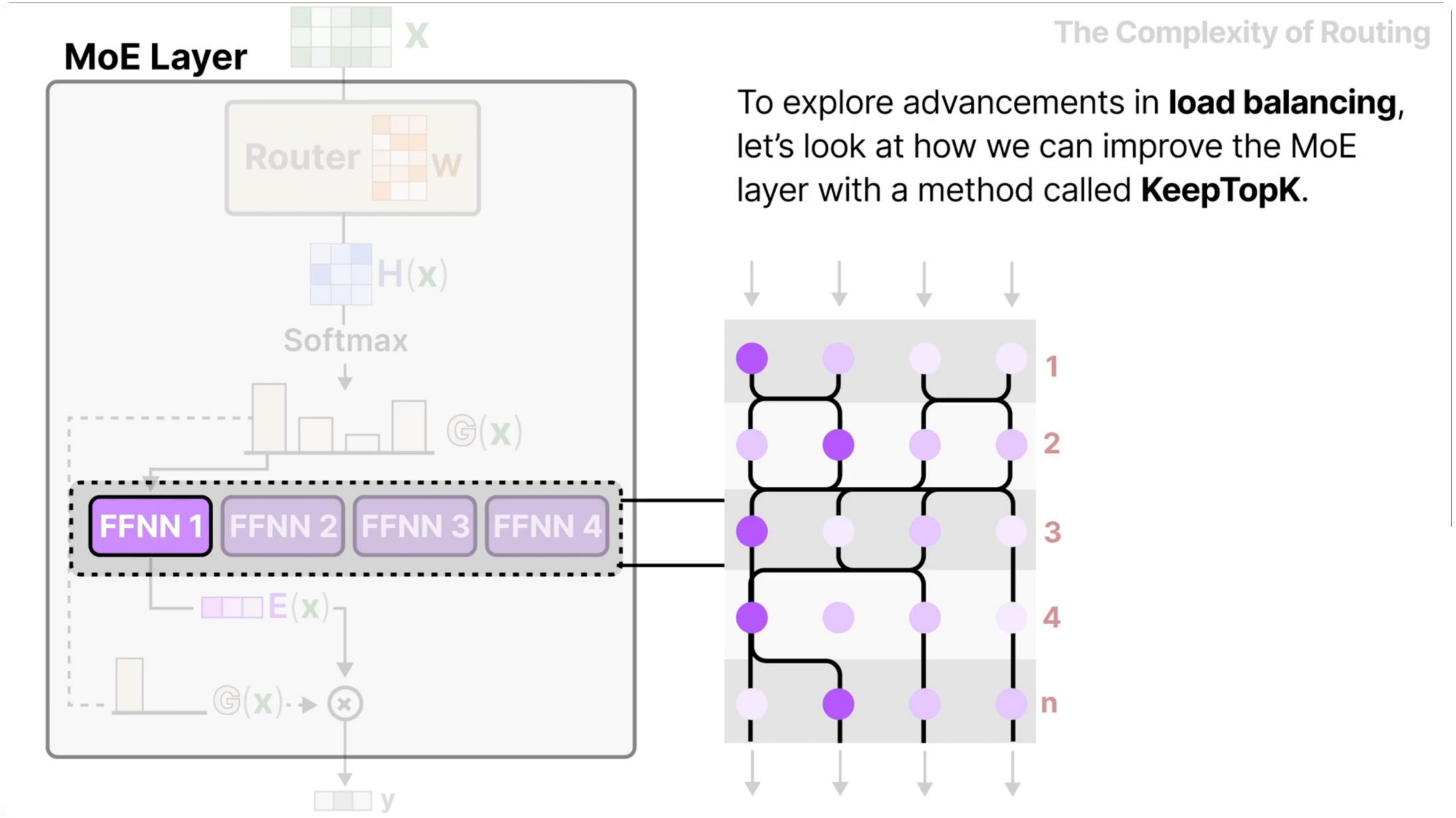
Load Balancing：均衡负载



The Complexity of Routing

We call this **load balancing**. It's to prevent overfitting on the same experts.

Load Balancing：均衡负载

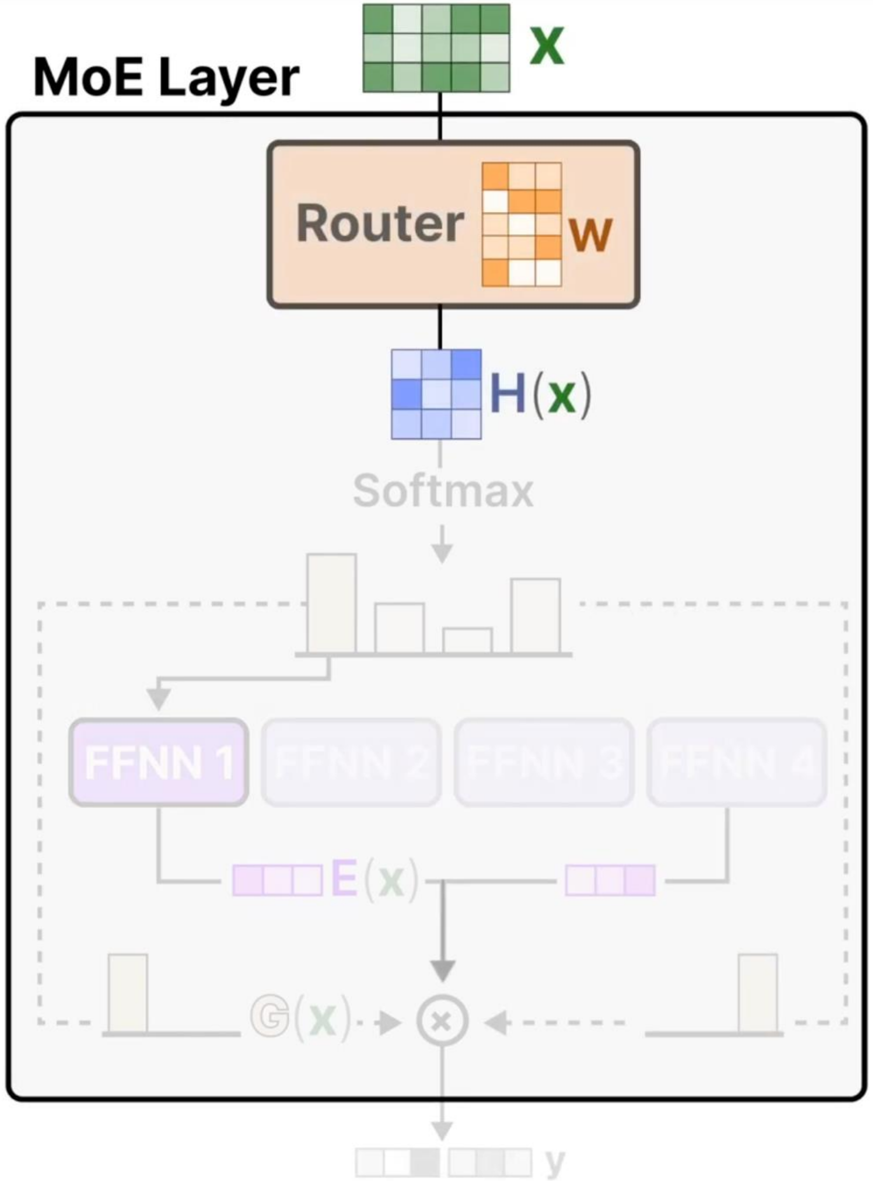


To explore advancements in **load balancing**, let's look at how we can improve the MoE layer with a method called **KeepTopK**.

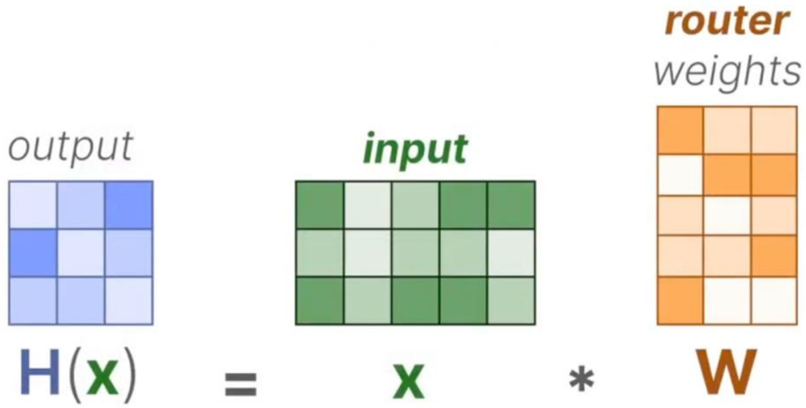
05

Keep Top-K：专家选择

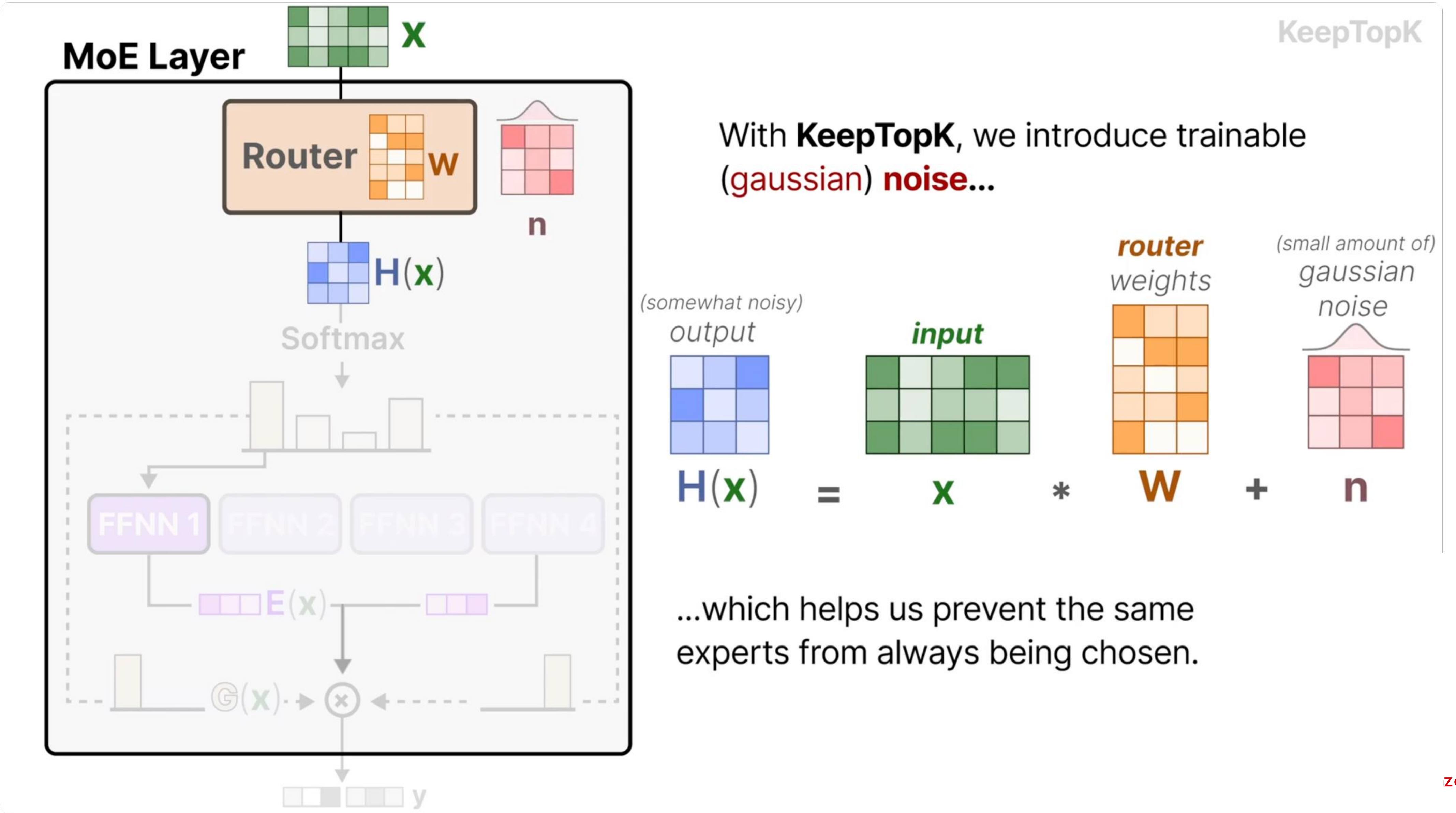
Keep Top-K：专家选择



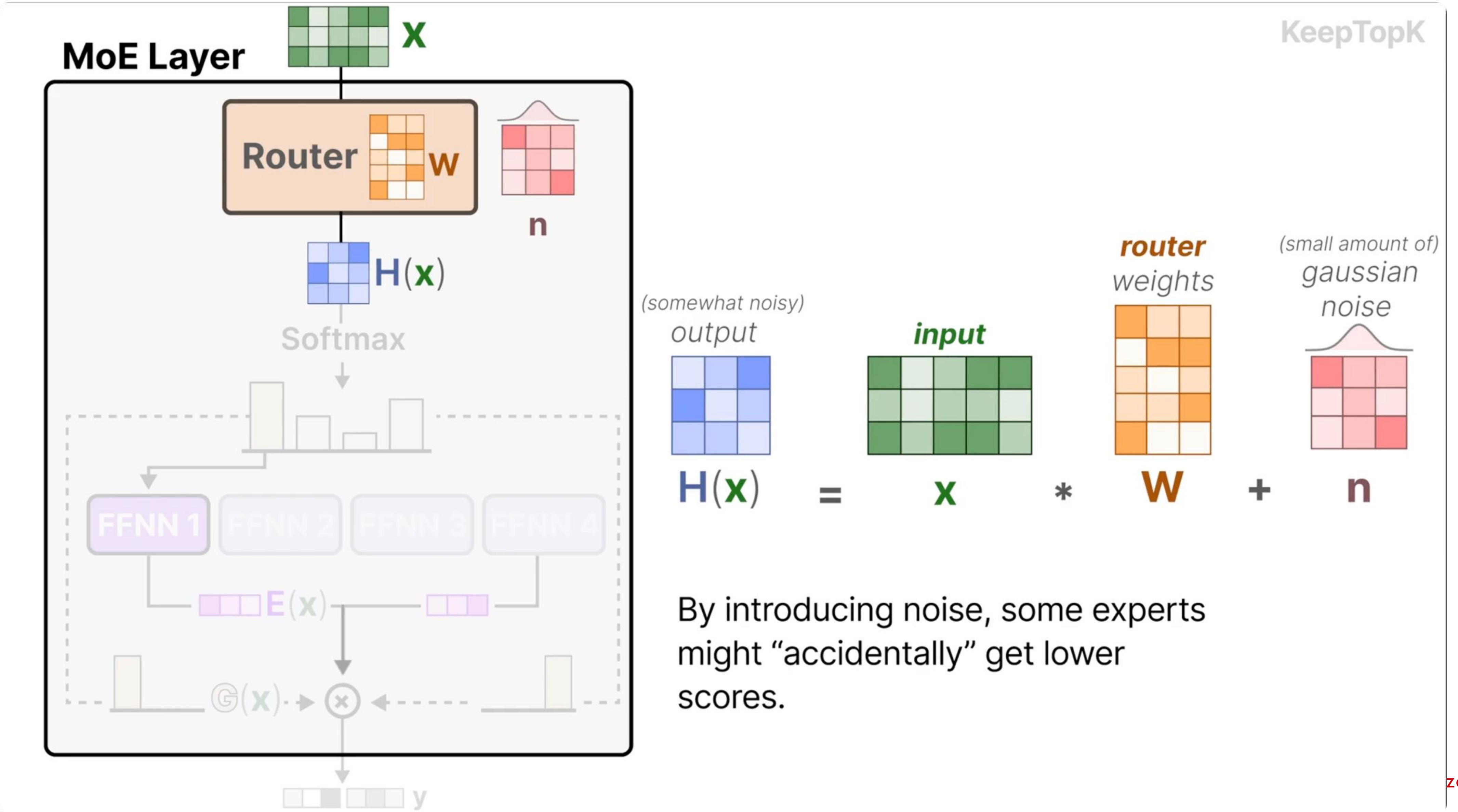
Remember that in the first step, we multiply the **input** with the **router** weights to create the **output** of the router.



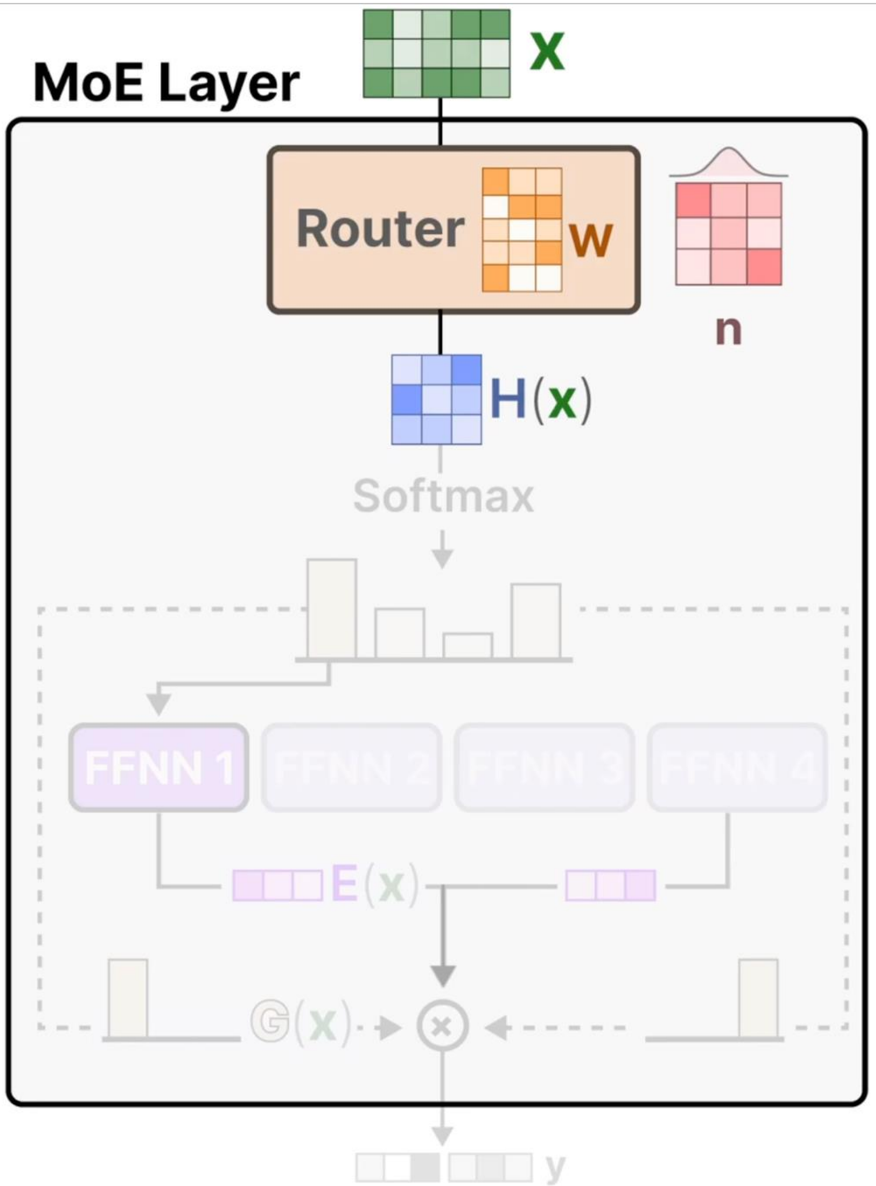
Keep Top-K：专家选择

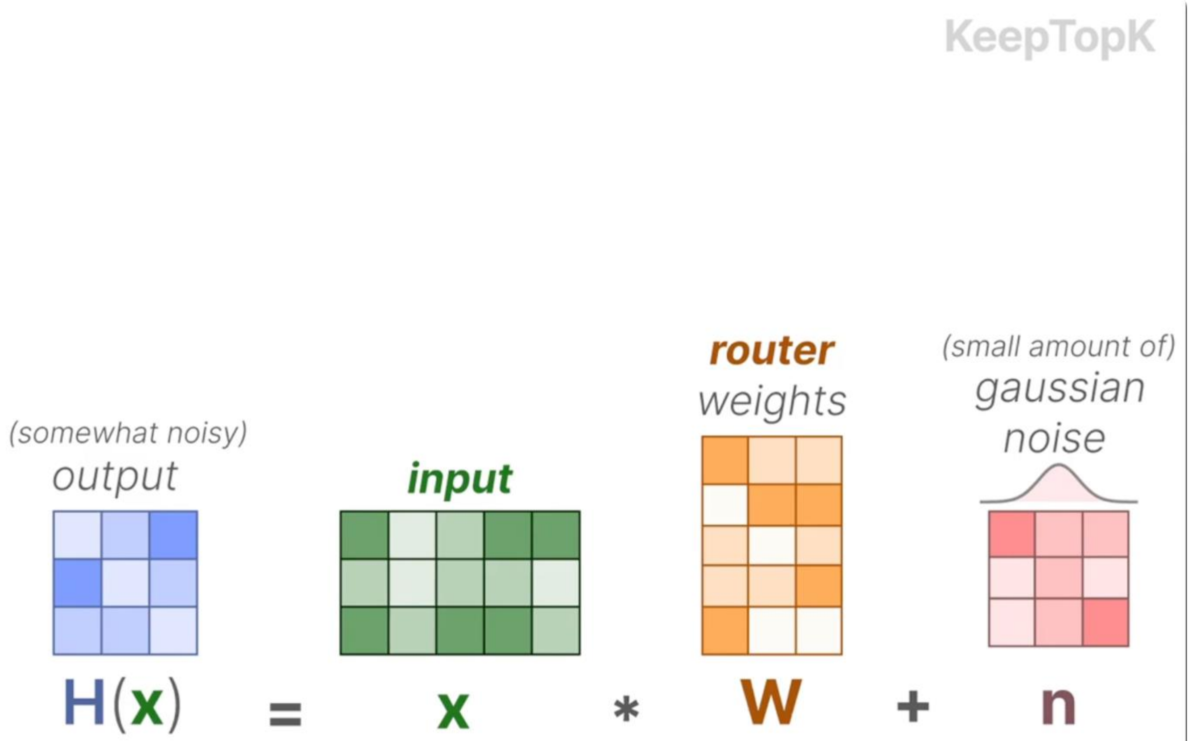


Keep Top-K：专家选择



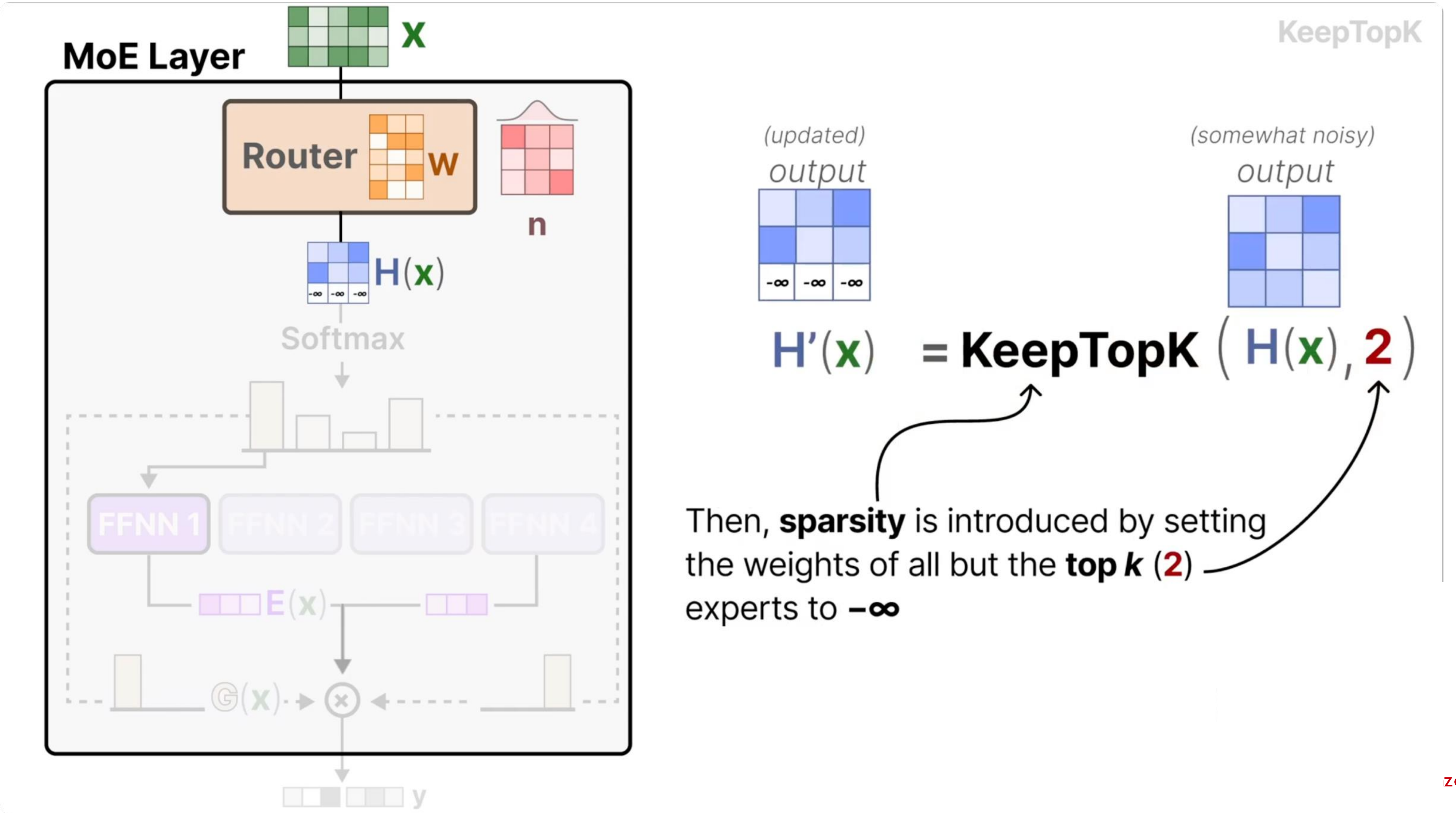
Keep Top-K：专家选择



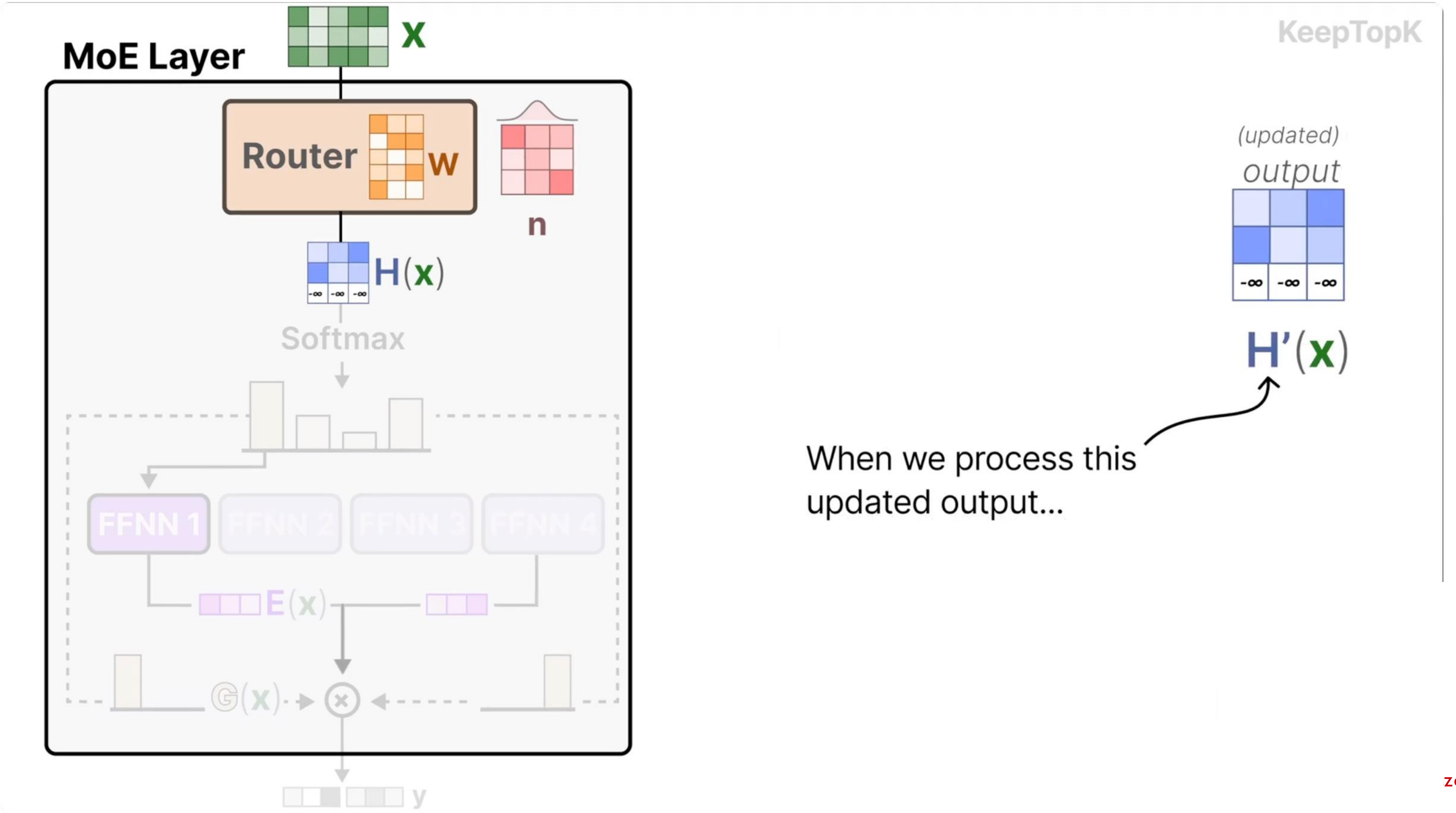


Thereby giving more opportunity for other experts to train.

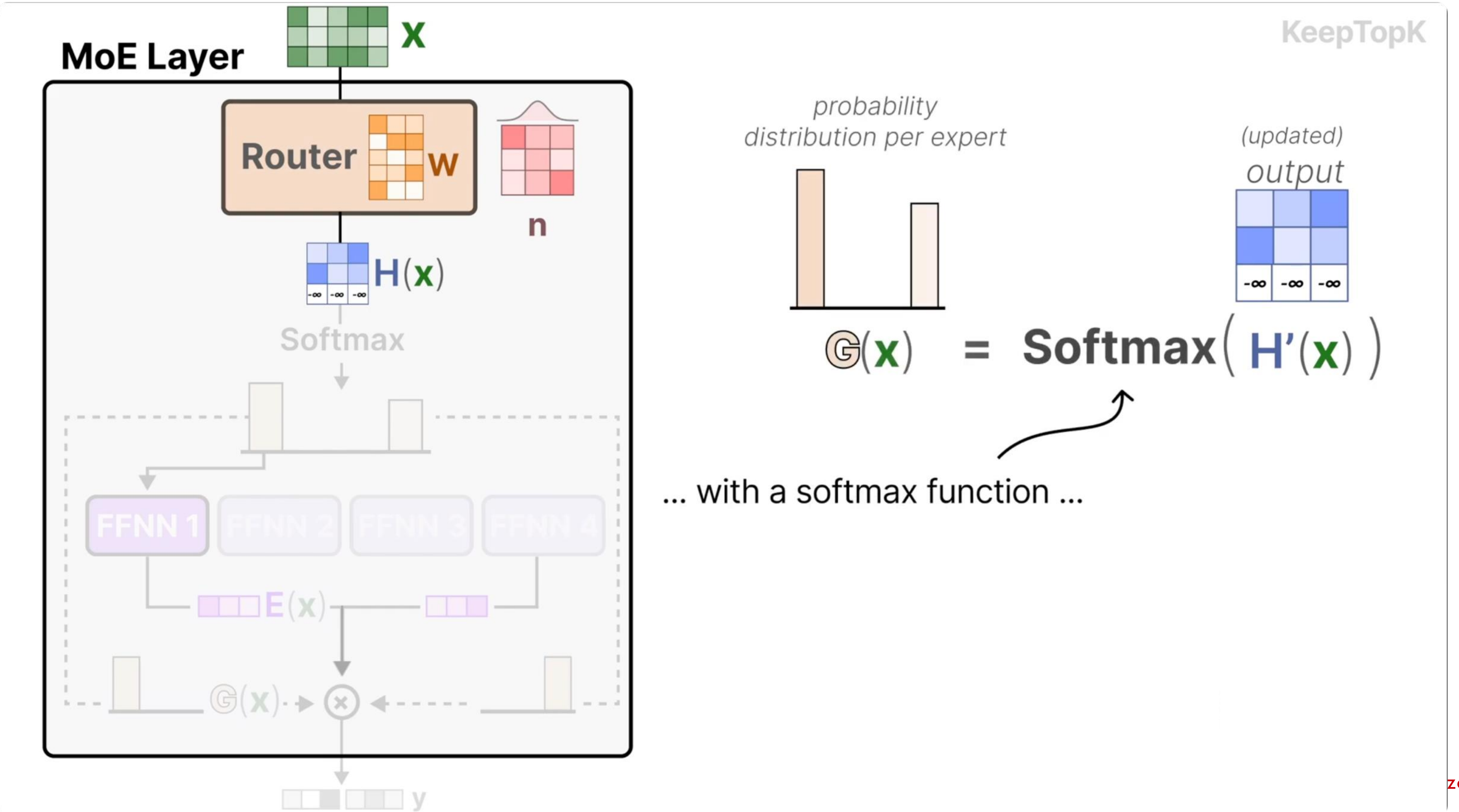
Keep Top-K：专家选择



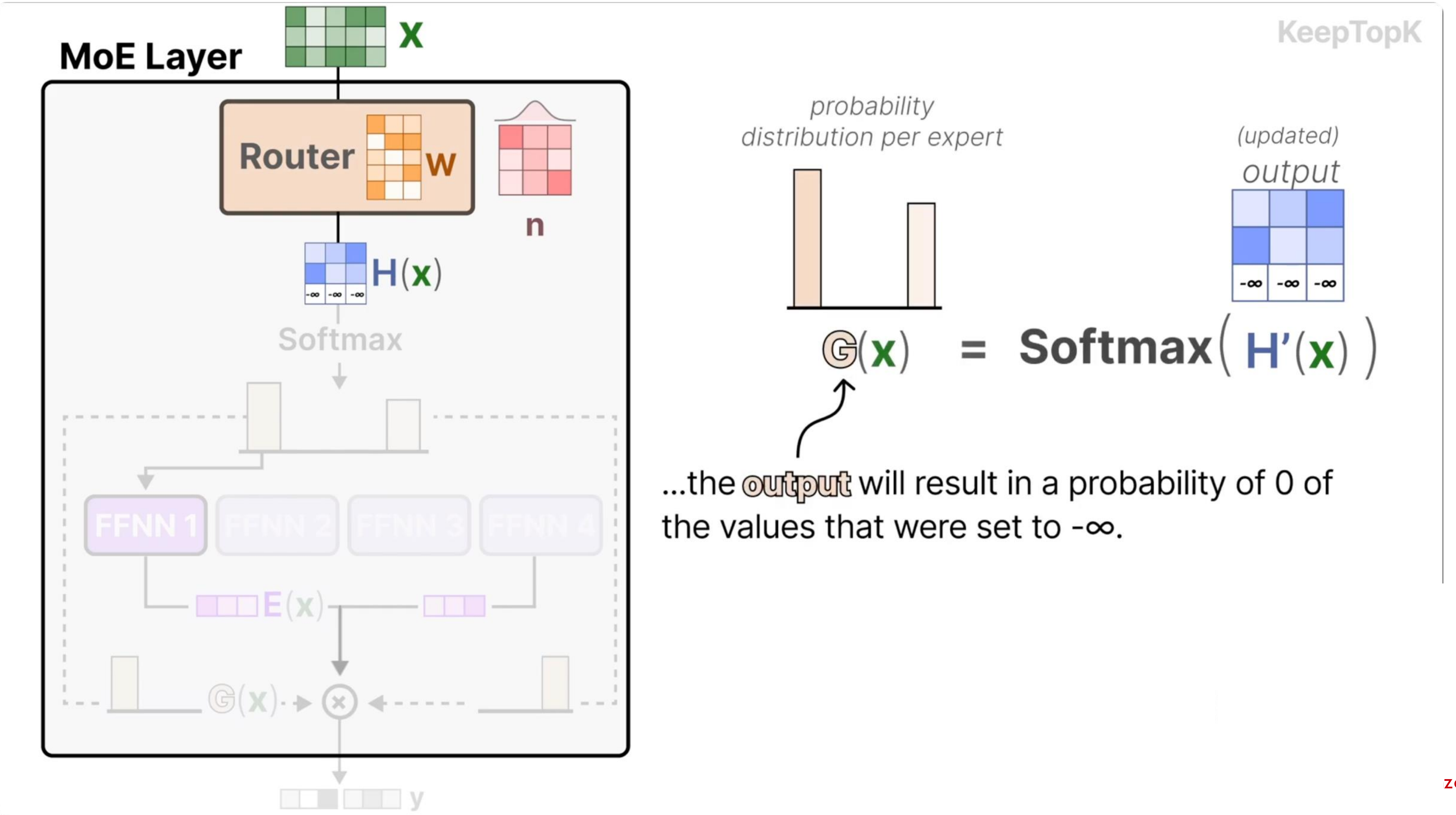
Keep Top-K：专家选择



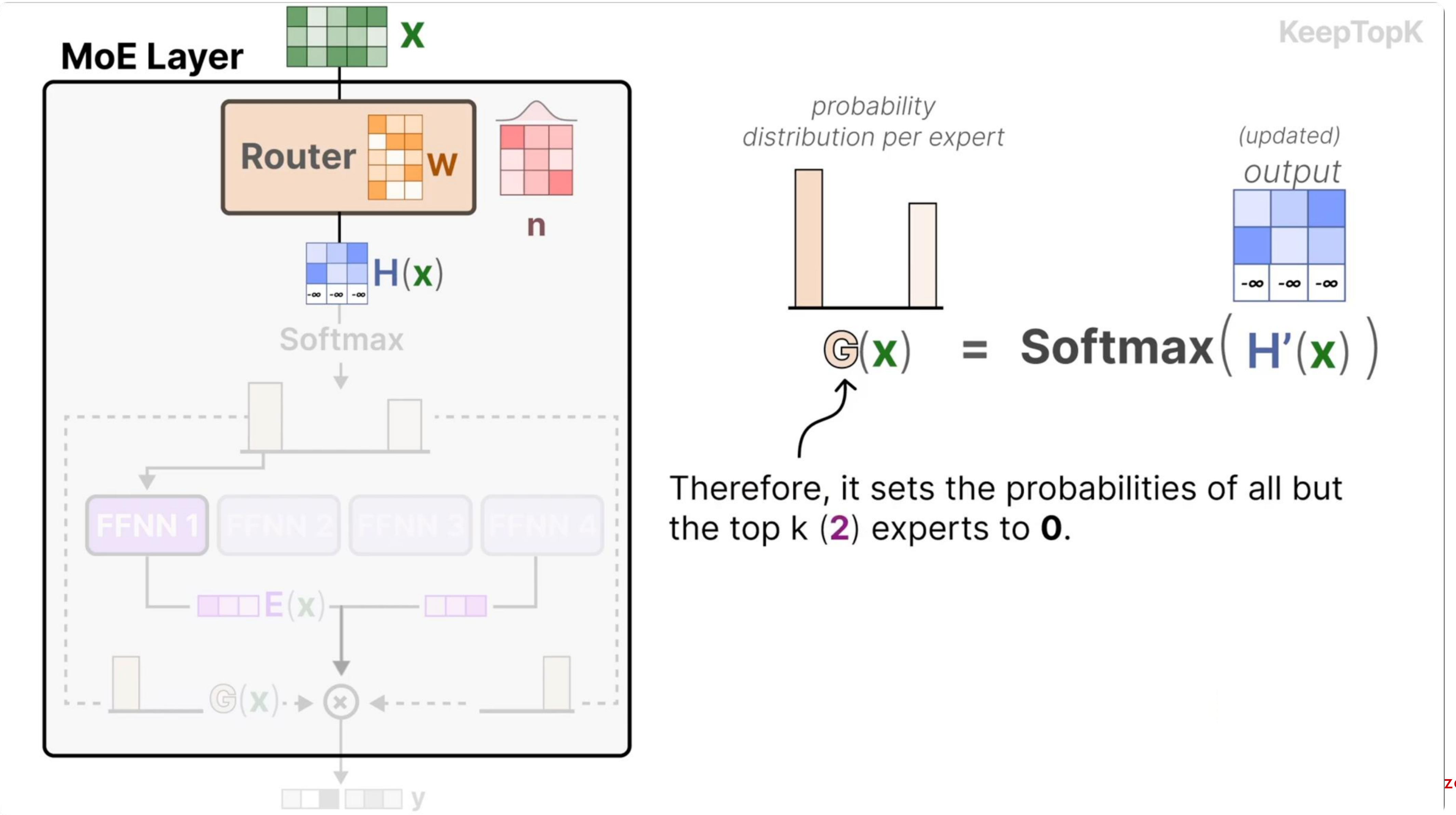
Keep Top-K：专家选择



Keep Top-K：专家选择

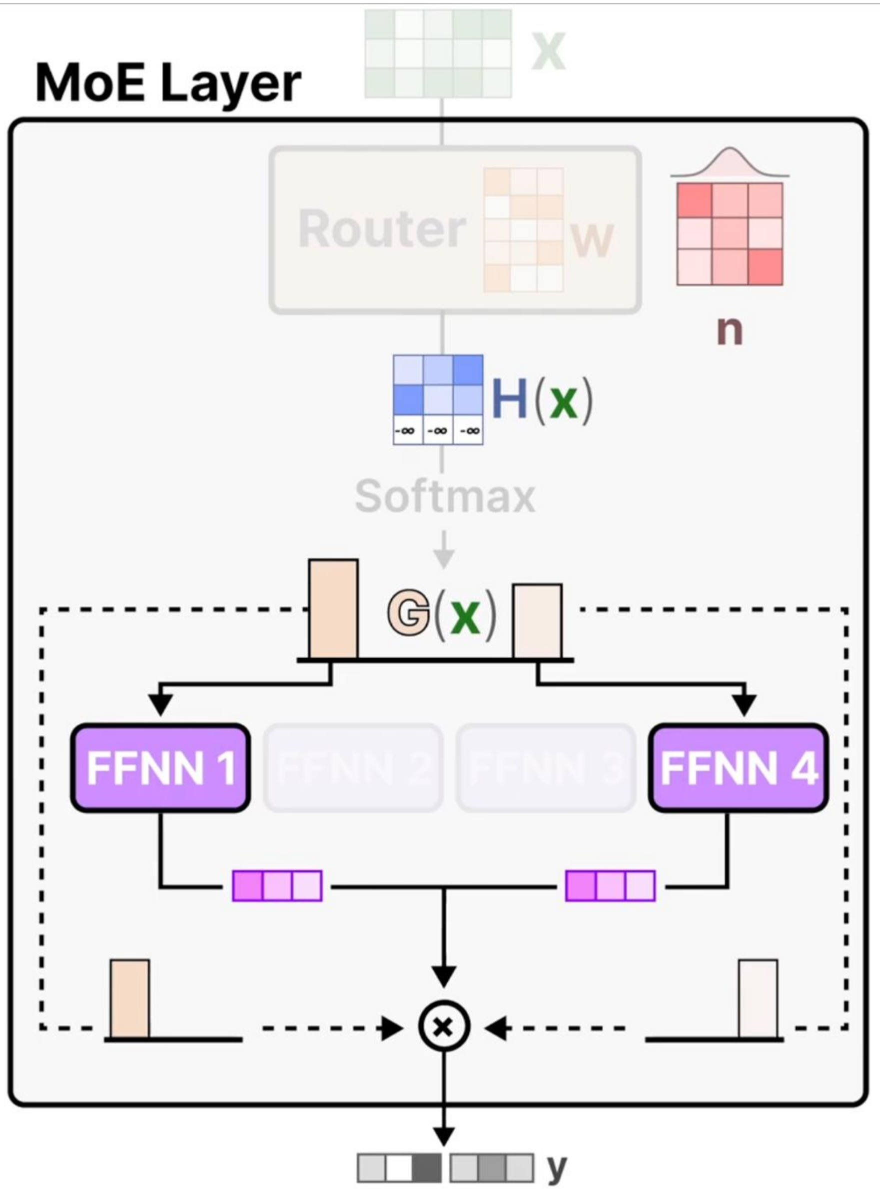


Keep Top-K：专家选择



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

Keep Top-K：专家选择

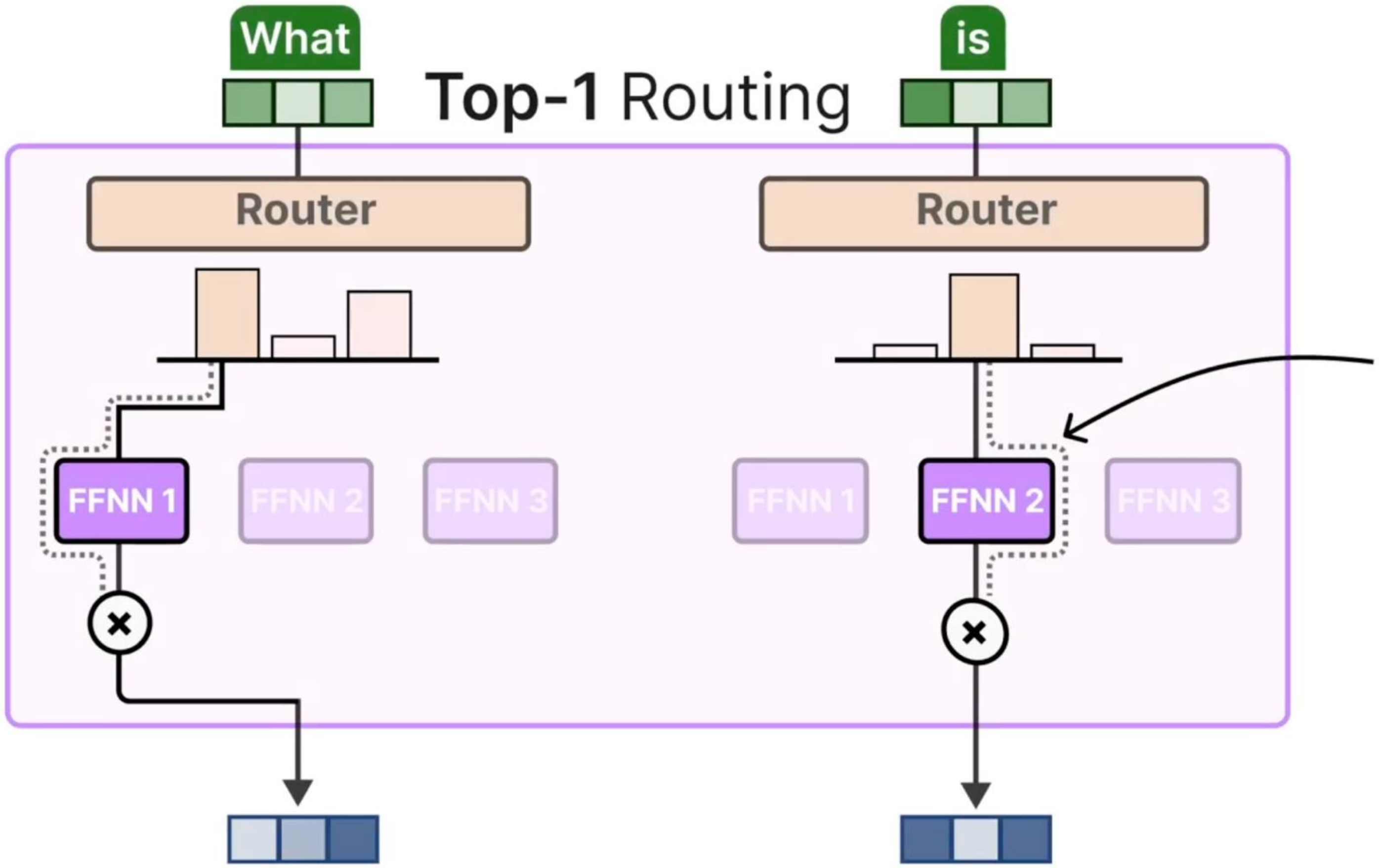


Combined, they allow undertrained **experts** to "*catch up*" to experts that were chosen more frequently and therefore had more opportunity to train.

Keep Top-K：专家选择

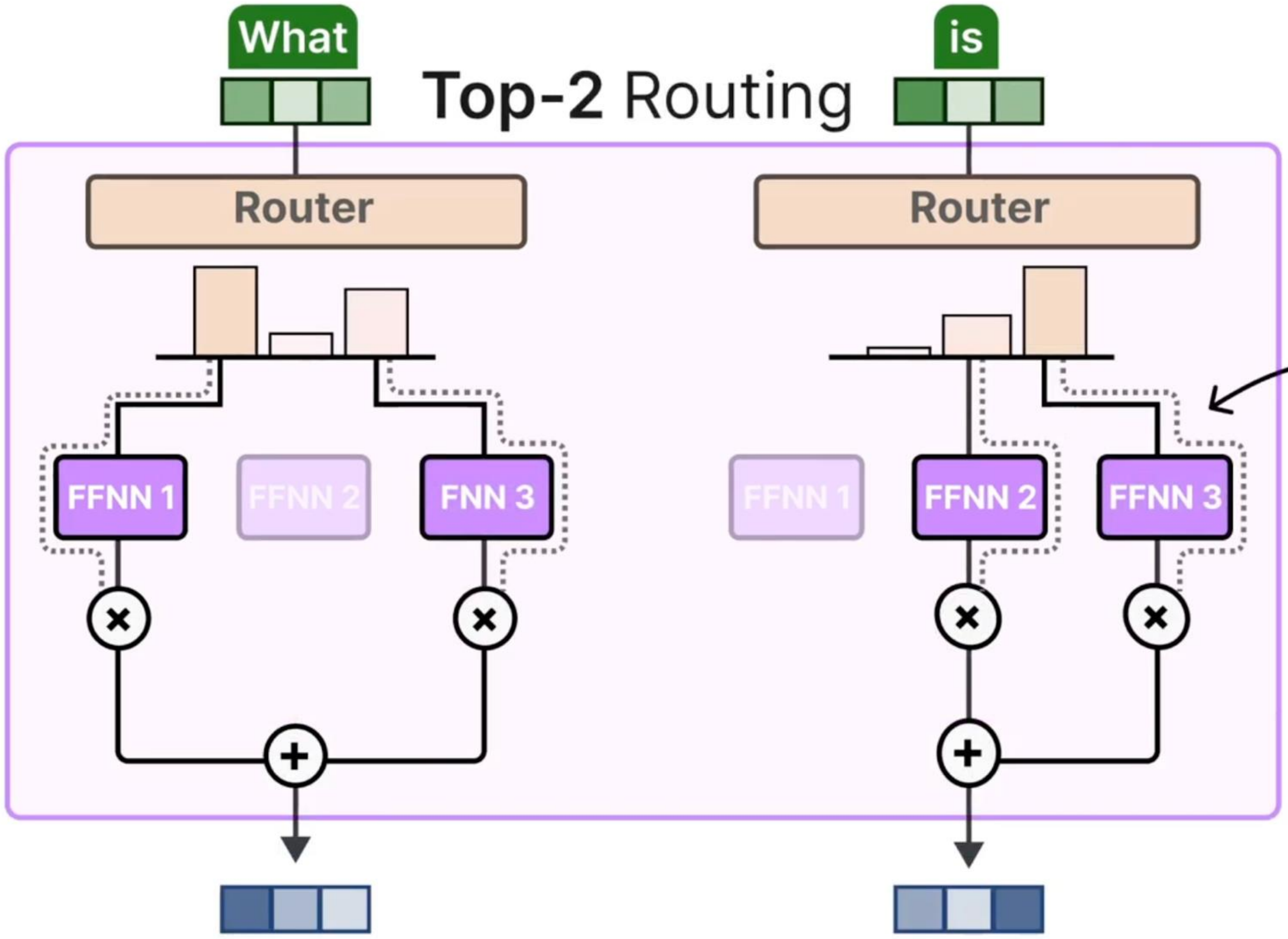
Routing tokens to a few selected experts is also called **Token Choice.**

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



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

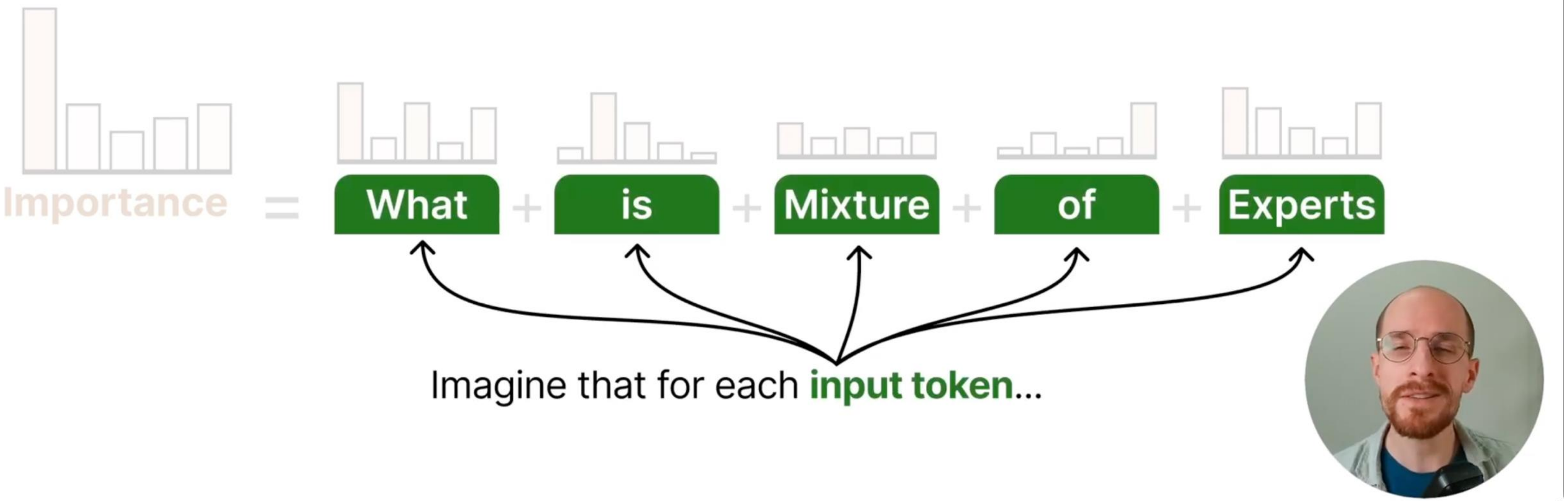
06

Auxiliary Loss：辅助损失

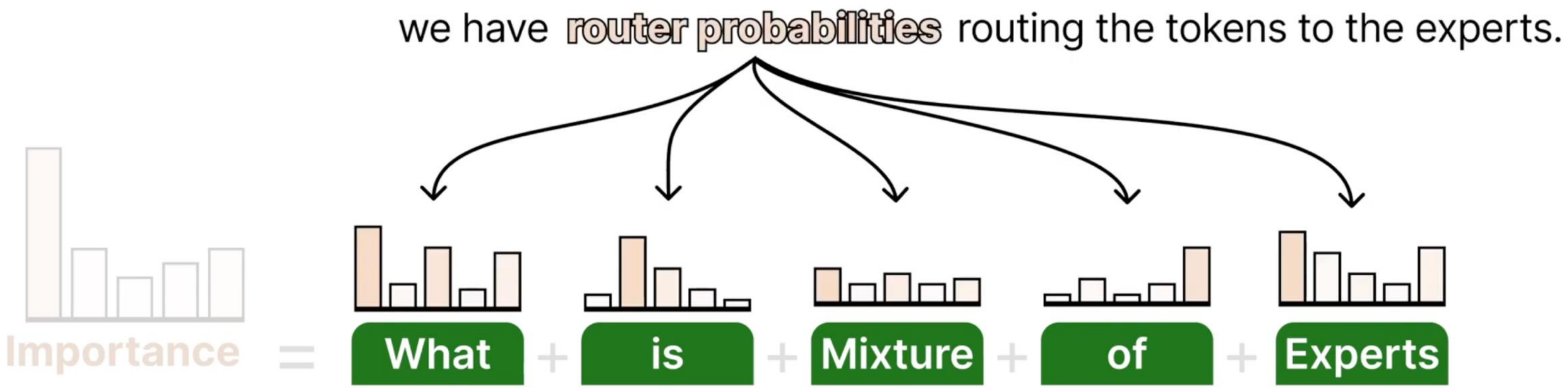
Auxiliary Loss：辅助损失

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

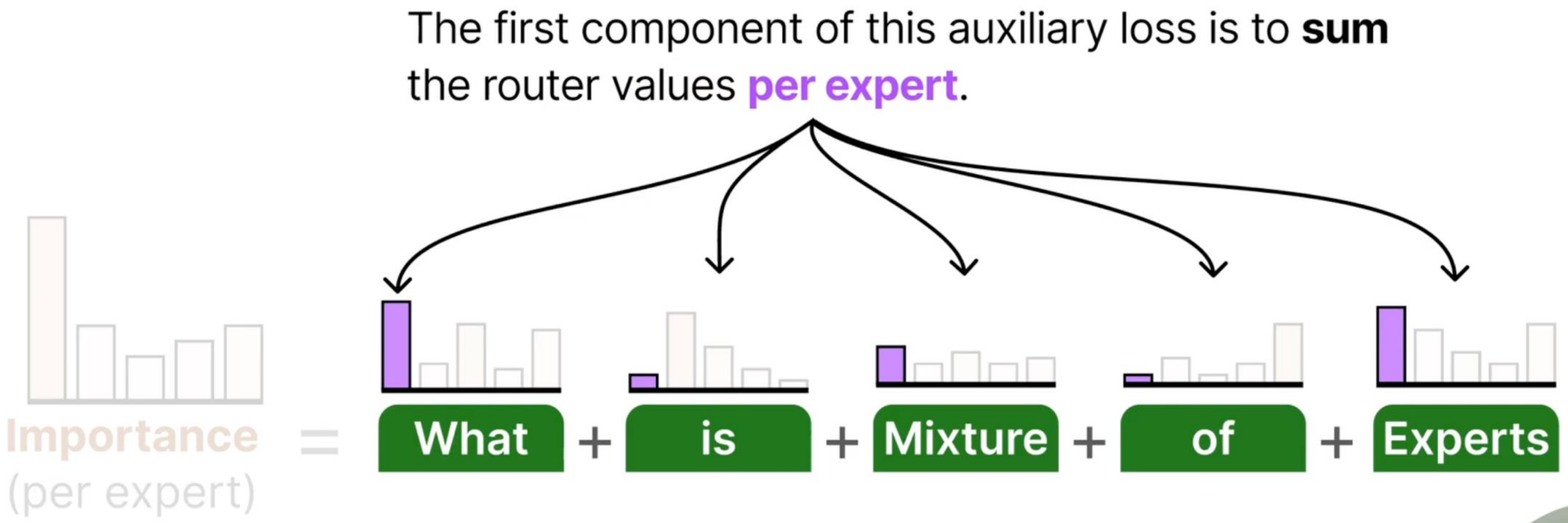
Auxiliary Loss：辅助损失



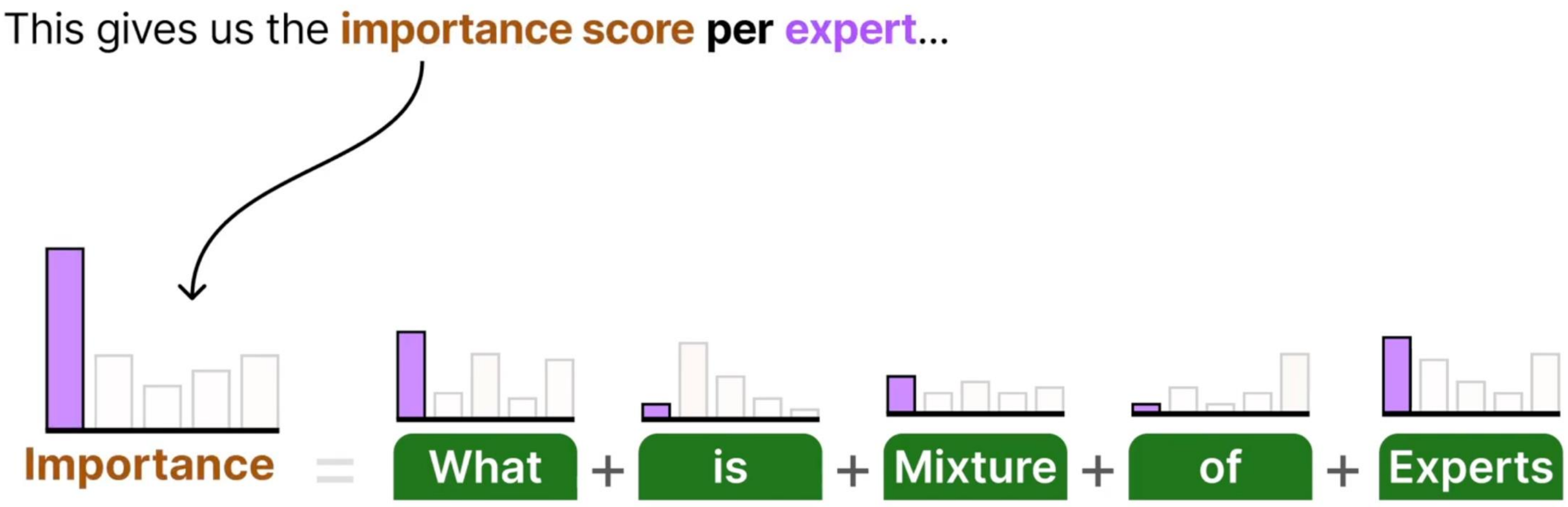
Auxiliary Loss：辅助损失



Auxiliary Loss：辅助损失



Auxiliary Loss：辅助损失



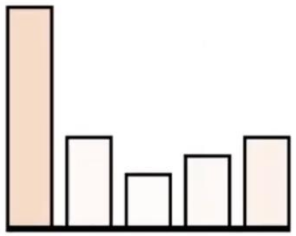
Auxiliary Loss：辅助损失

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

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

Auxiliary Loss：辅助损失

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



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

Auxiliary Loss：辅助损失

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

Auxiliary Loss：辅助损失

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

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

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....

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

(constant)
scaling factor

Auxiliary Loss：辅助损失

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

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

(constant)
scaling factor

Auxiliary Loss：辅助损失

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

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

Auxiliary Loss：辅助损失

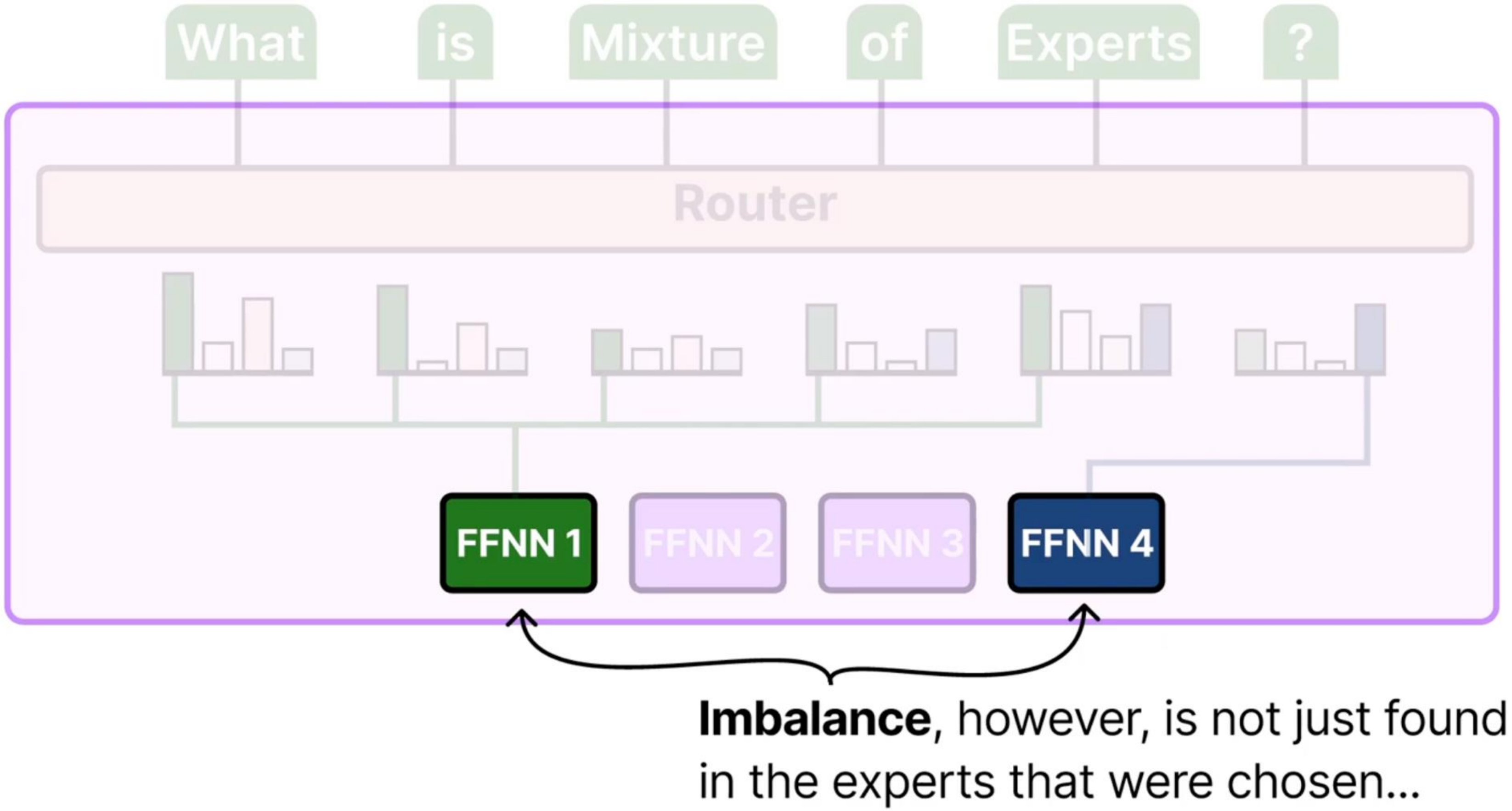
This additional loss therefore results in a more **stable training** procedure where all experts are given (somewhat) equal chance to train.

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

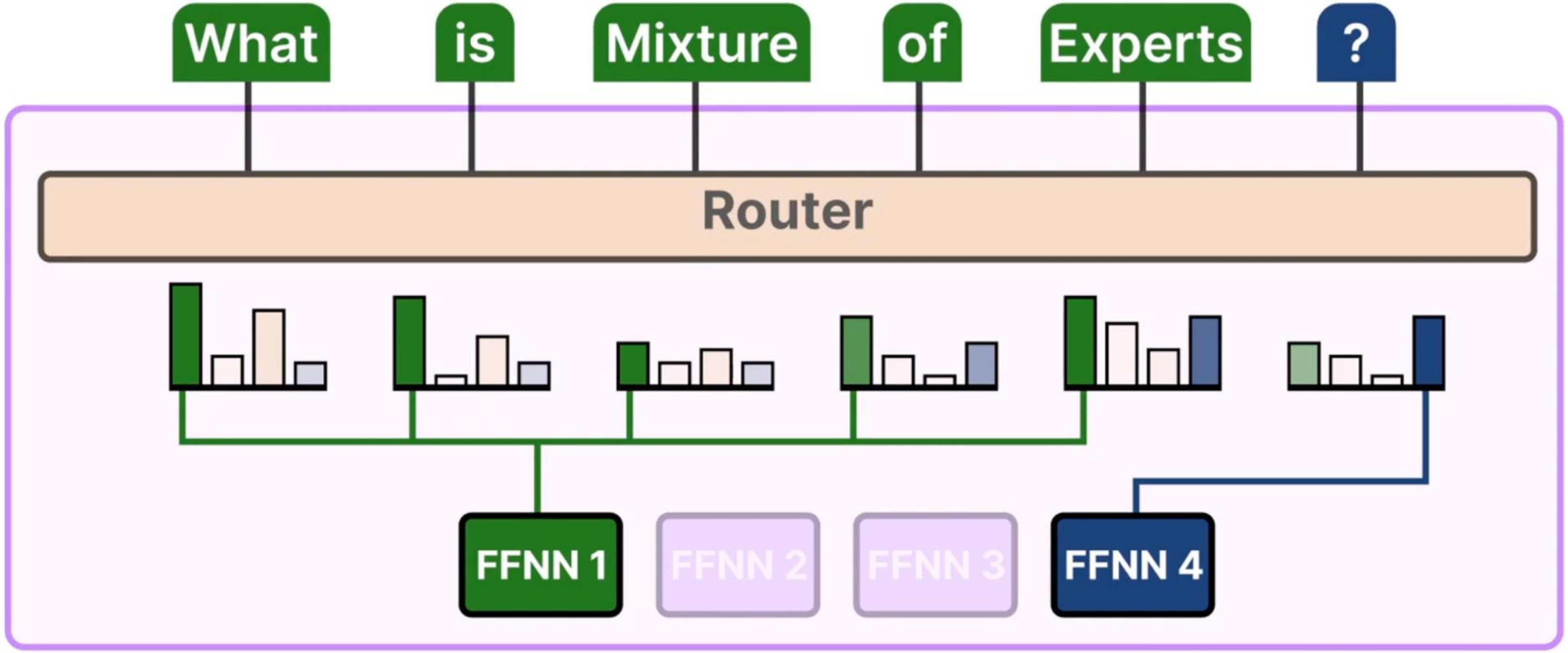
07

Expert Capacity：专家容量

Expert Capacity：专家容量

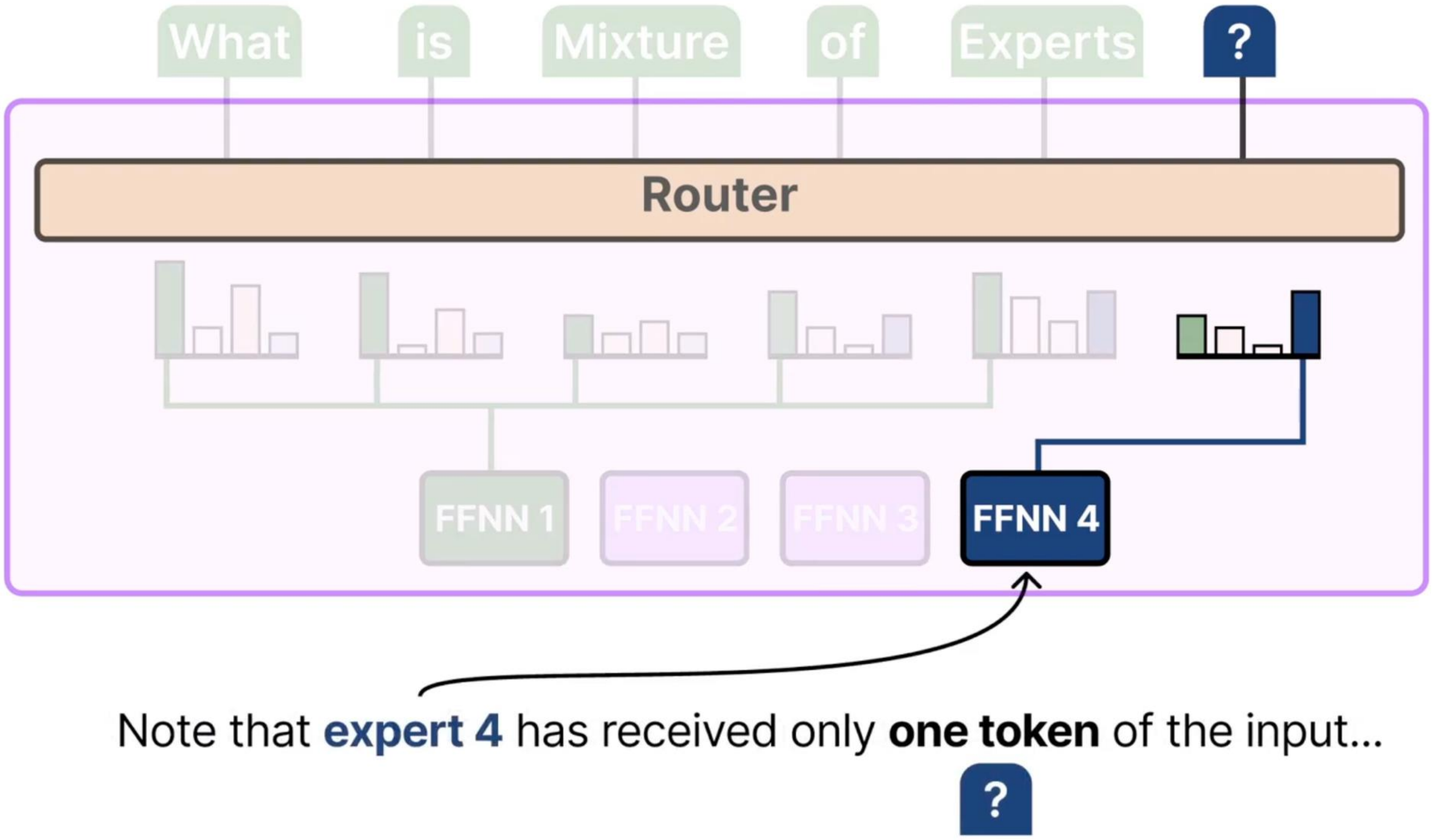


Expert Capacity：专家容量

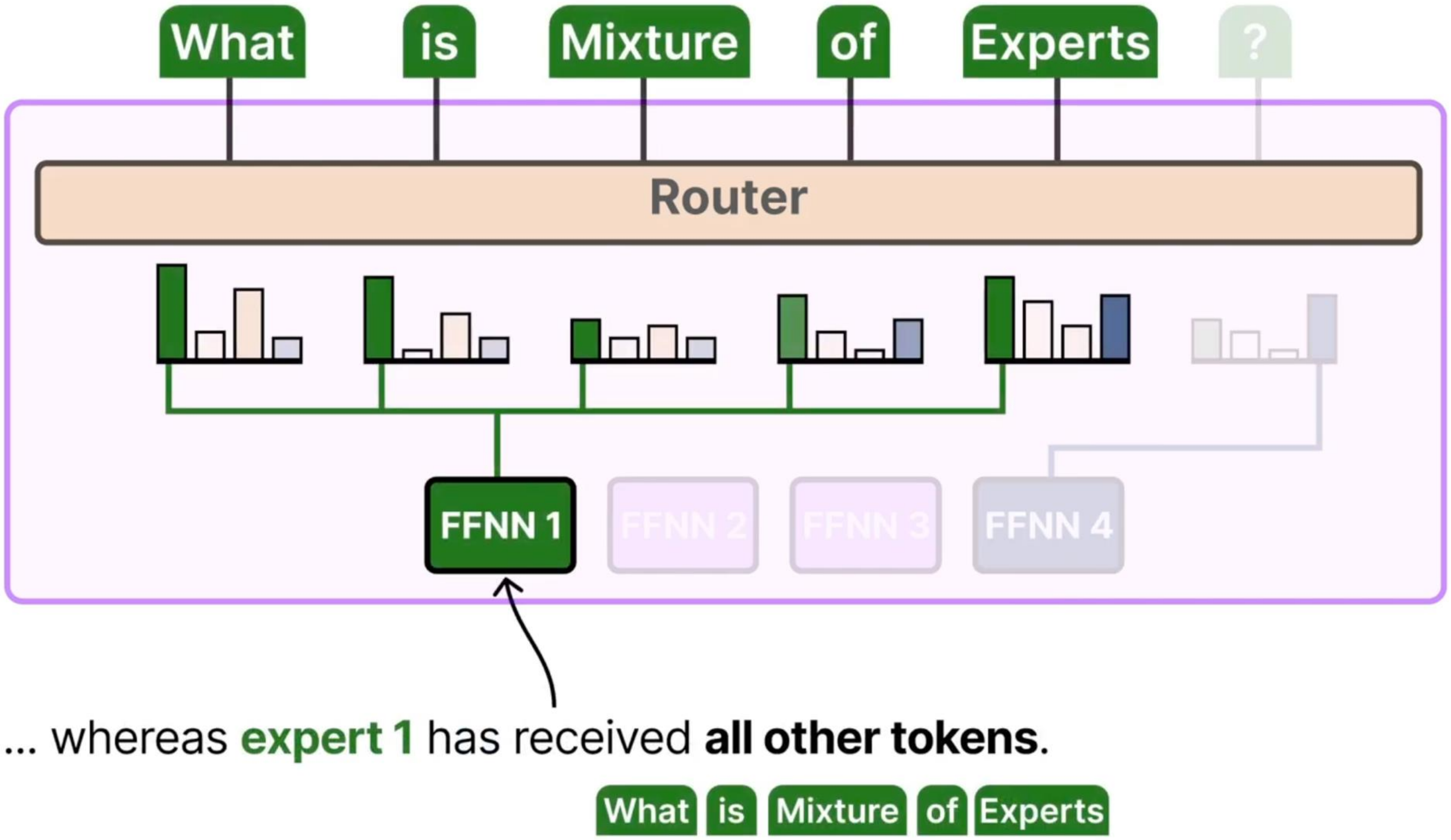


... but also in the distributions of tokens that are sent to the expert.

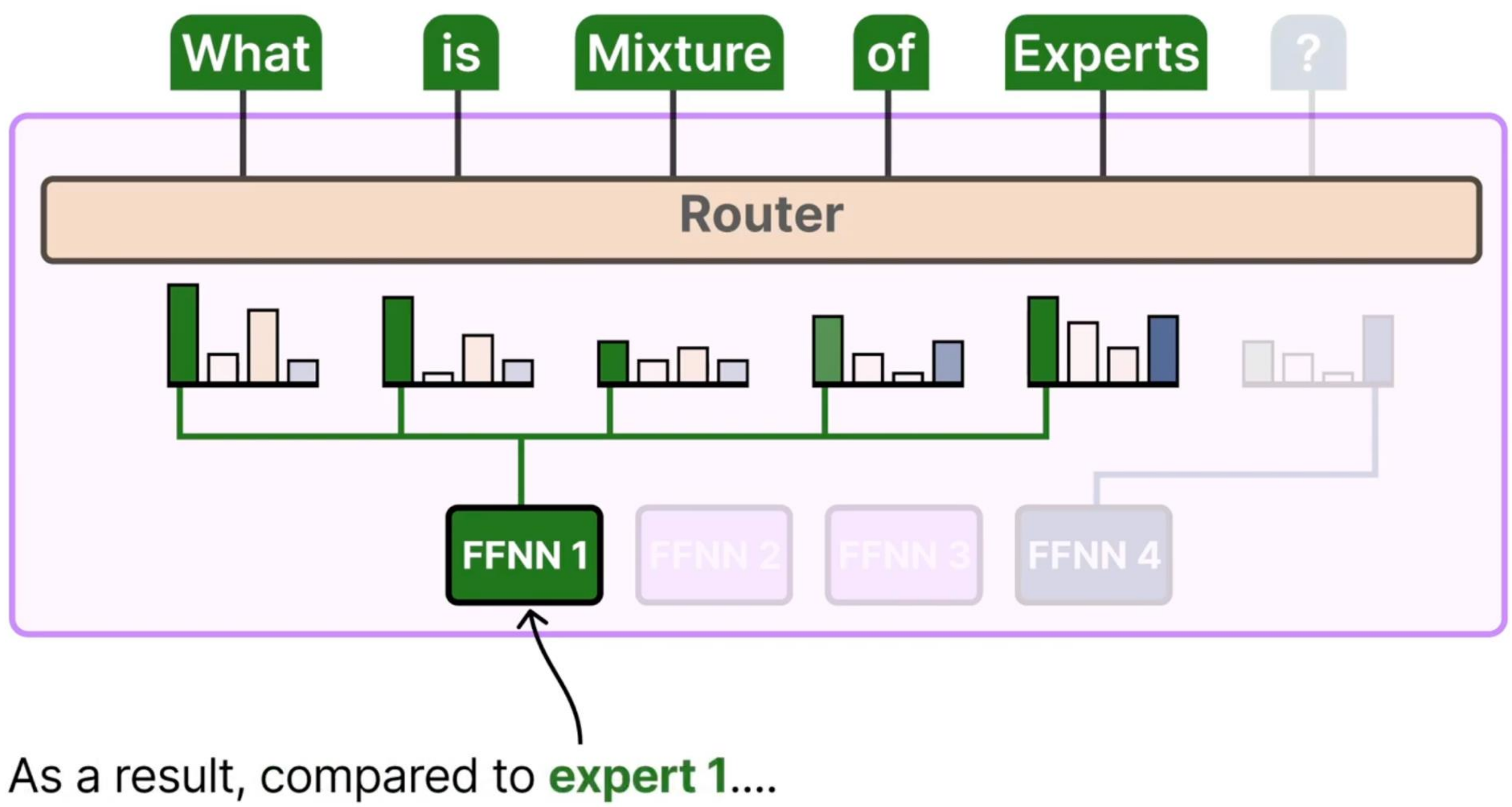
Expert Capacity：专家容量



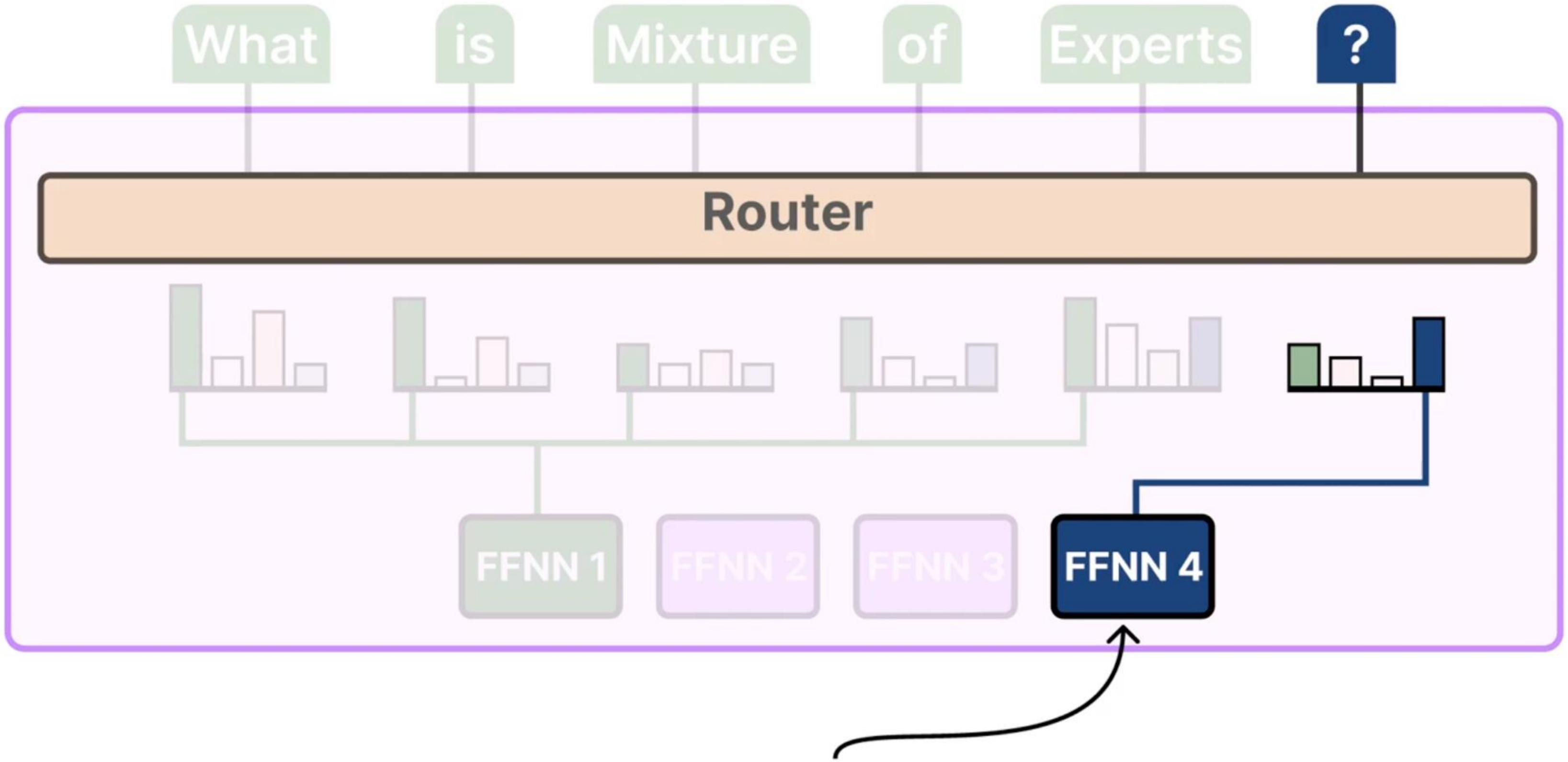
Expert Capacity：专家容量



Expert Capacity：专家容量

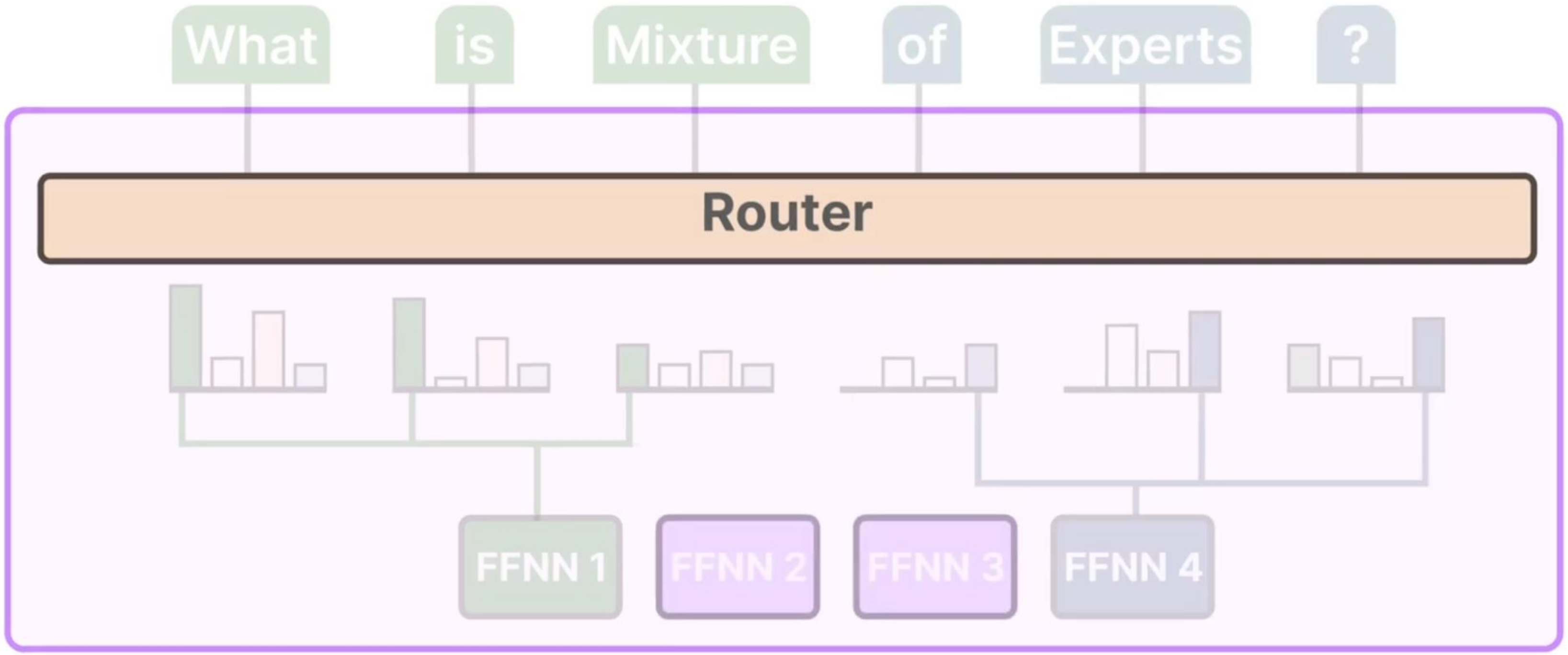


Expert Capacity：专家容量



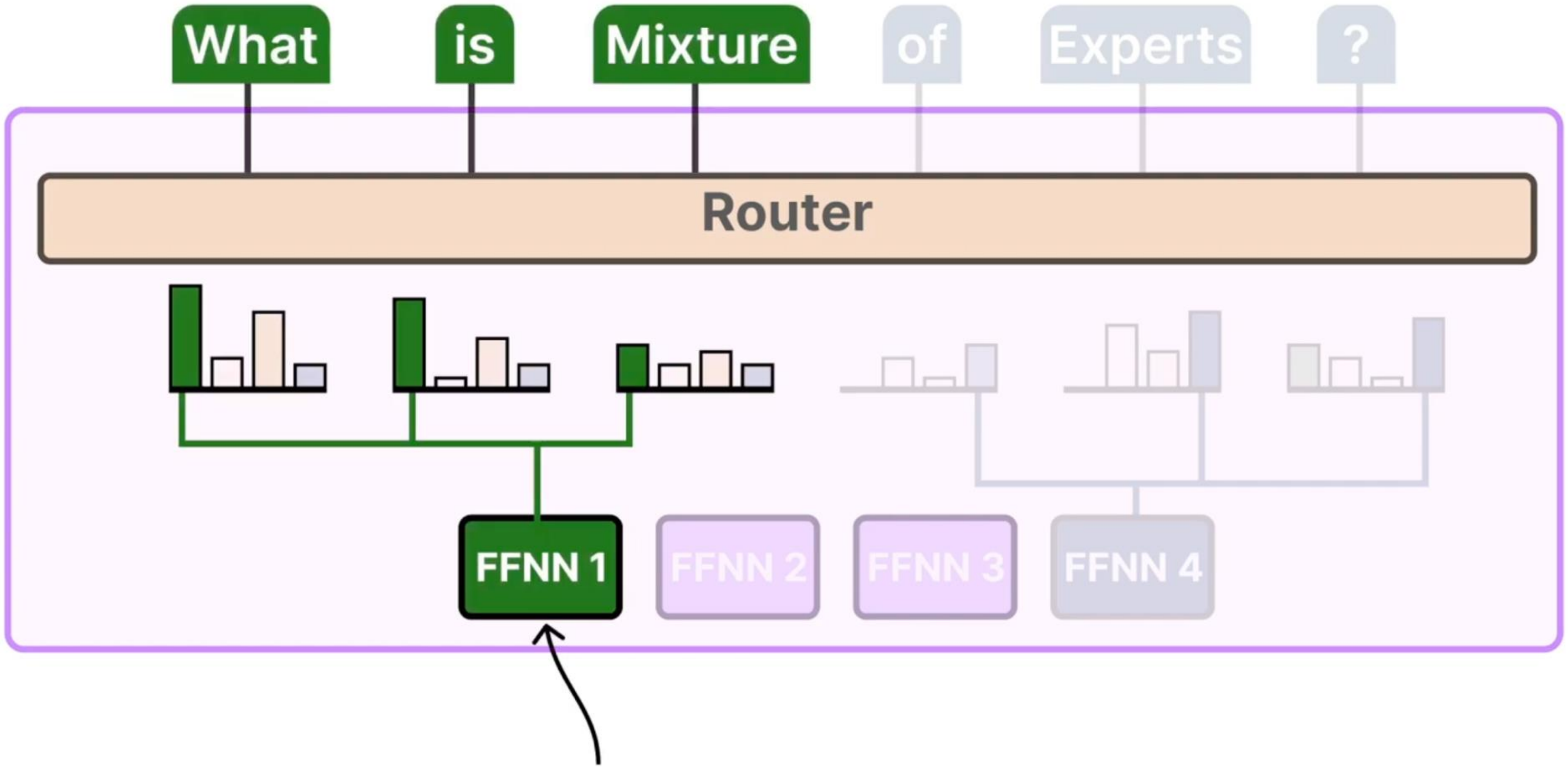
...**expert 4** ends up undertrained since it receives so few tokens during training.

Expert Capacity：专家容量



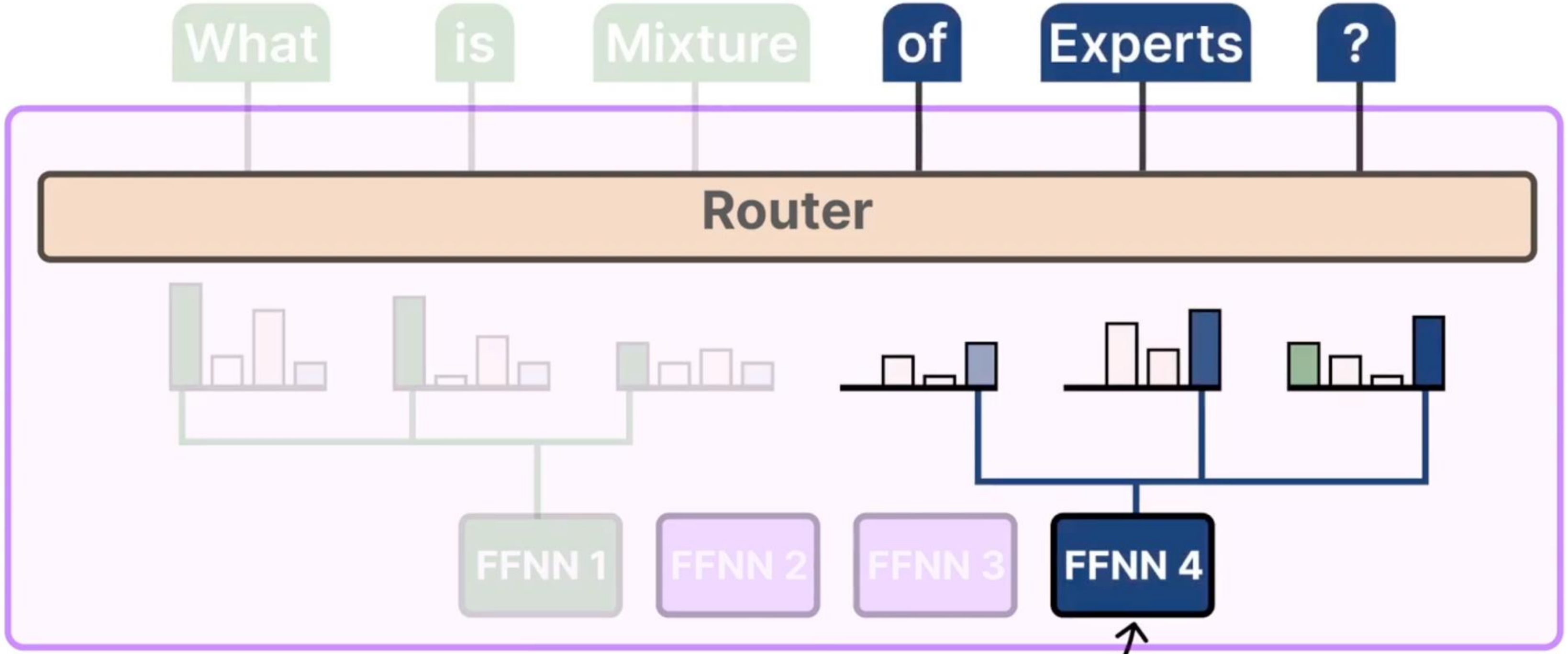
To prevent this problem, we **limit** how many tokens they can process, called the **expert capacity**.

Expert Capacity：专家容量



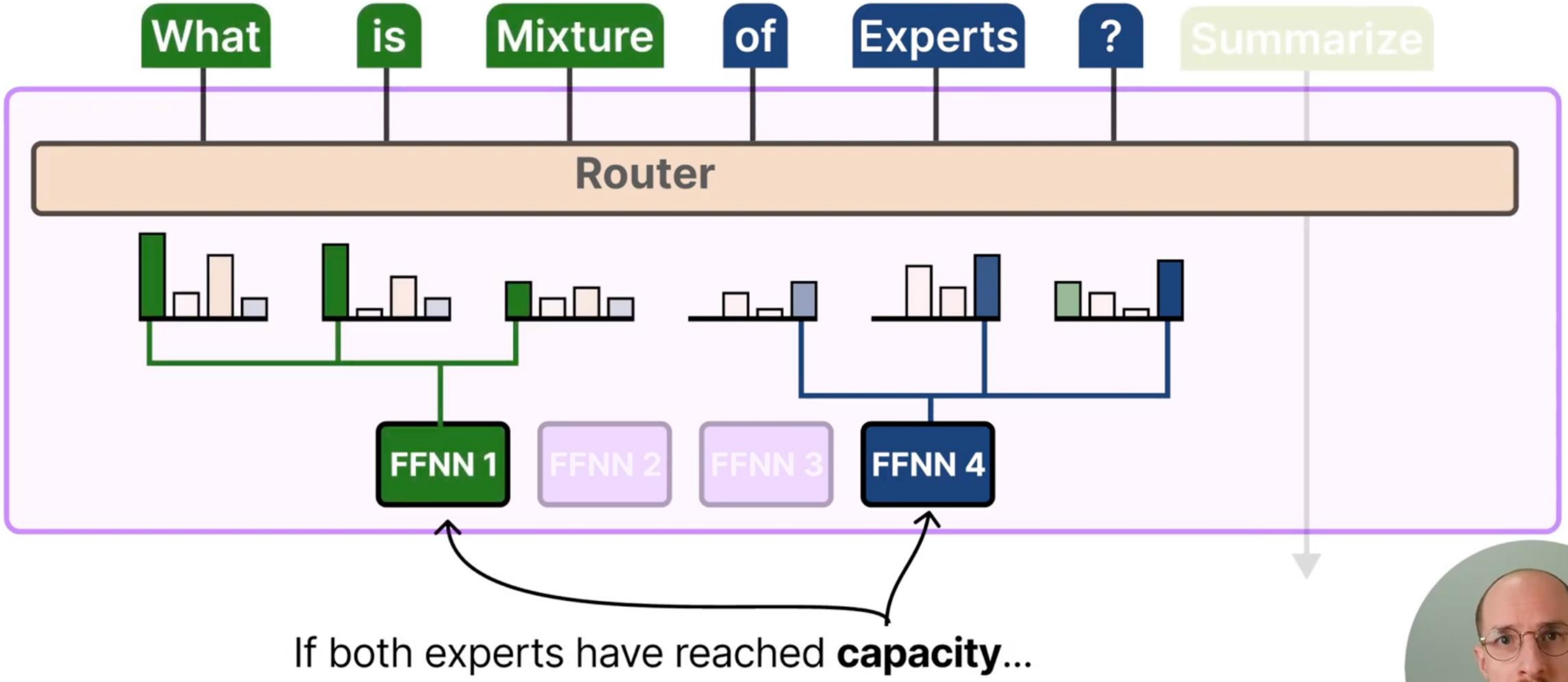
By setting the **expert capacity** to **3**, this expert can only process 3 tokens.

Expert Capacity：专家容量

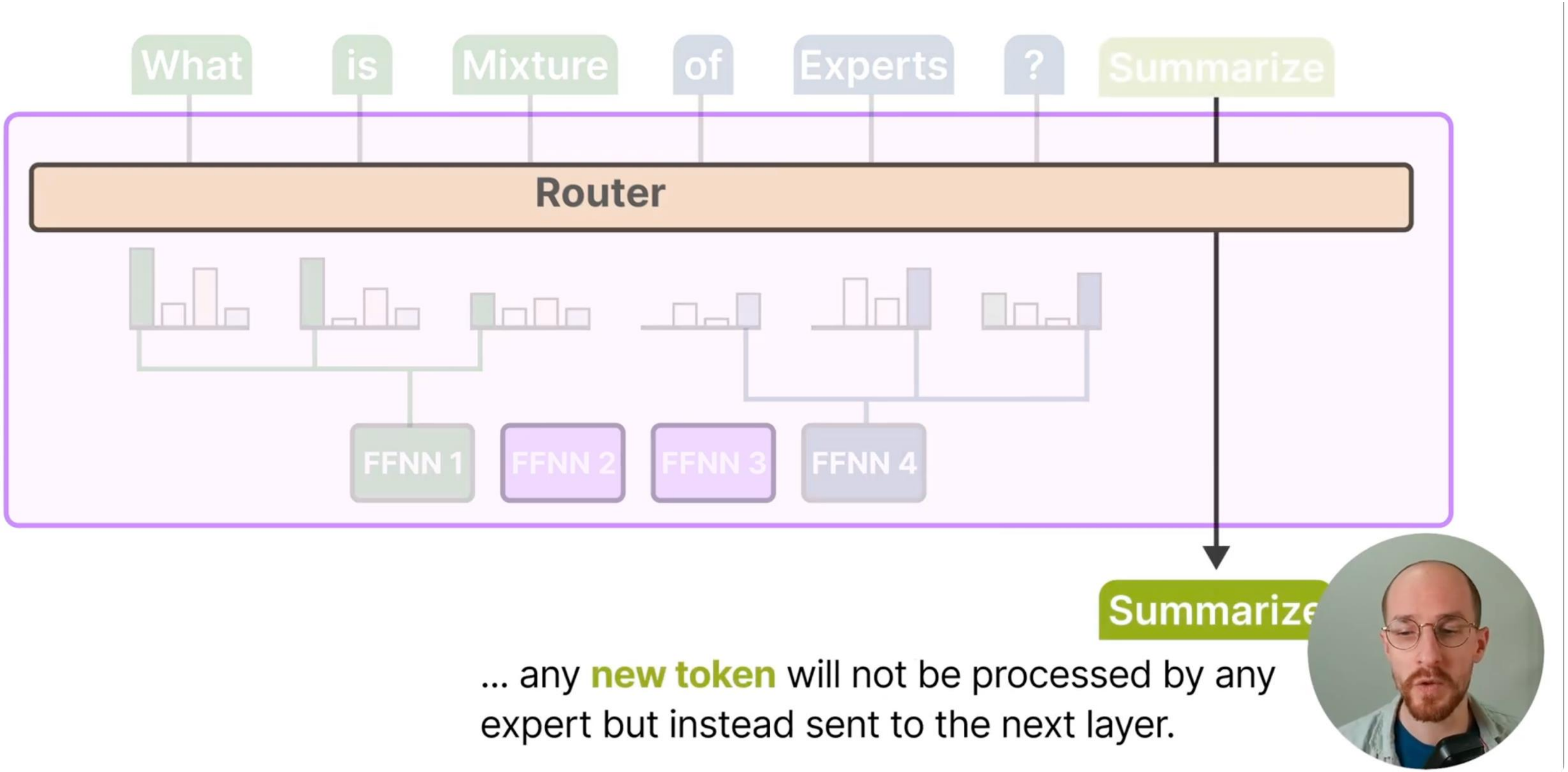


Tokens that exceed this threshold are routed to the next most likely expert.

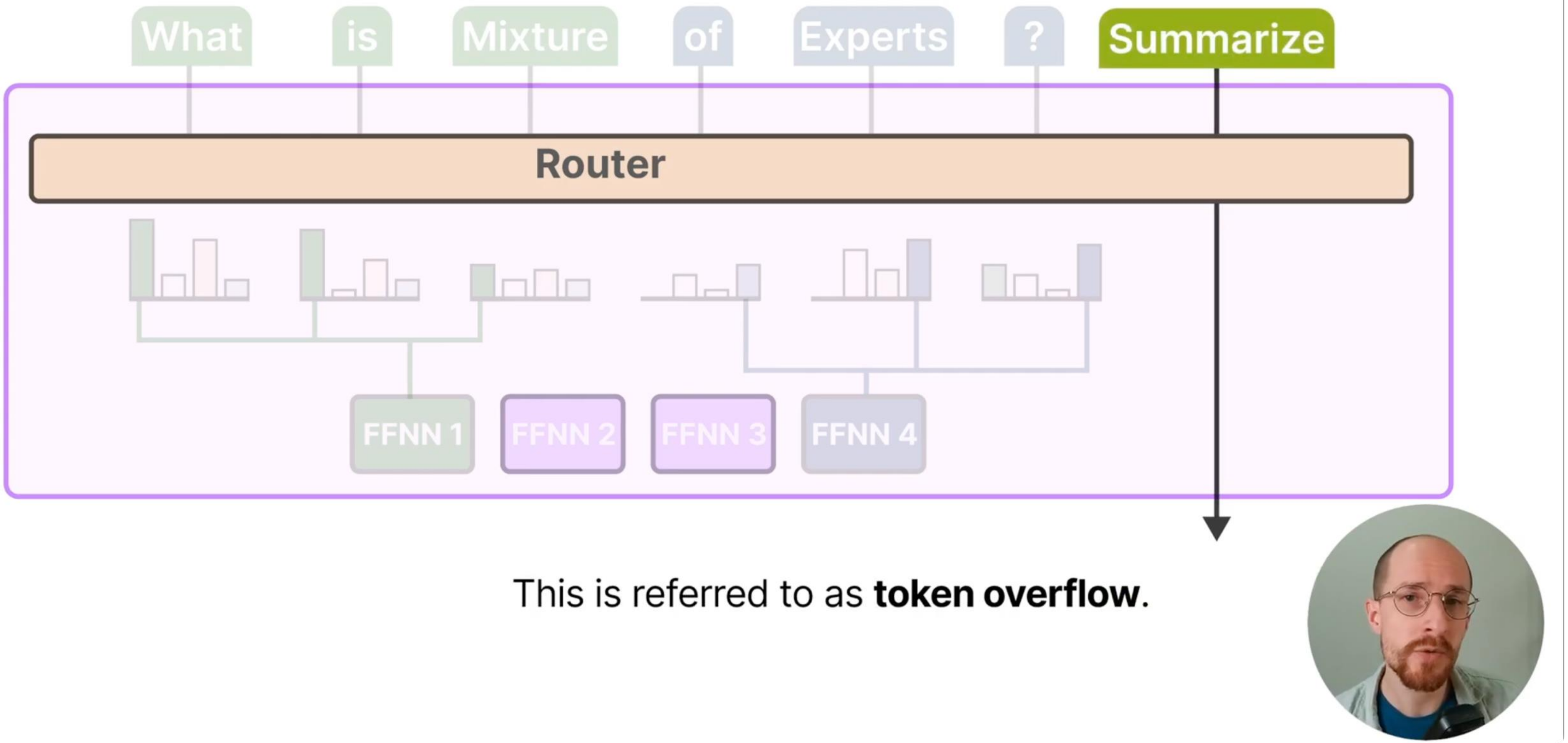
Expert Capacity：专家容量



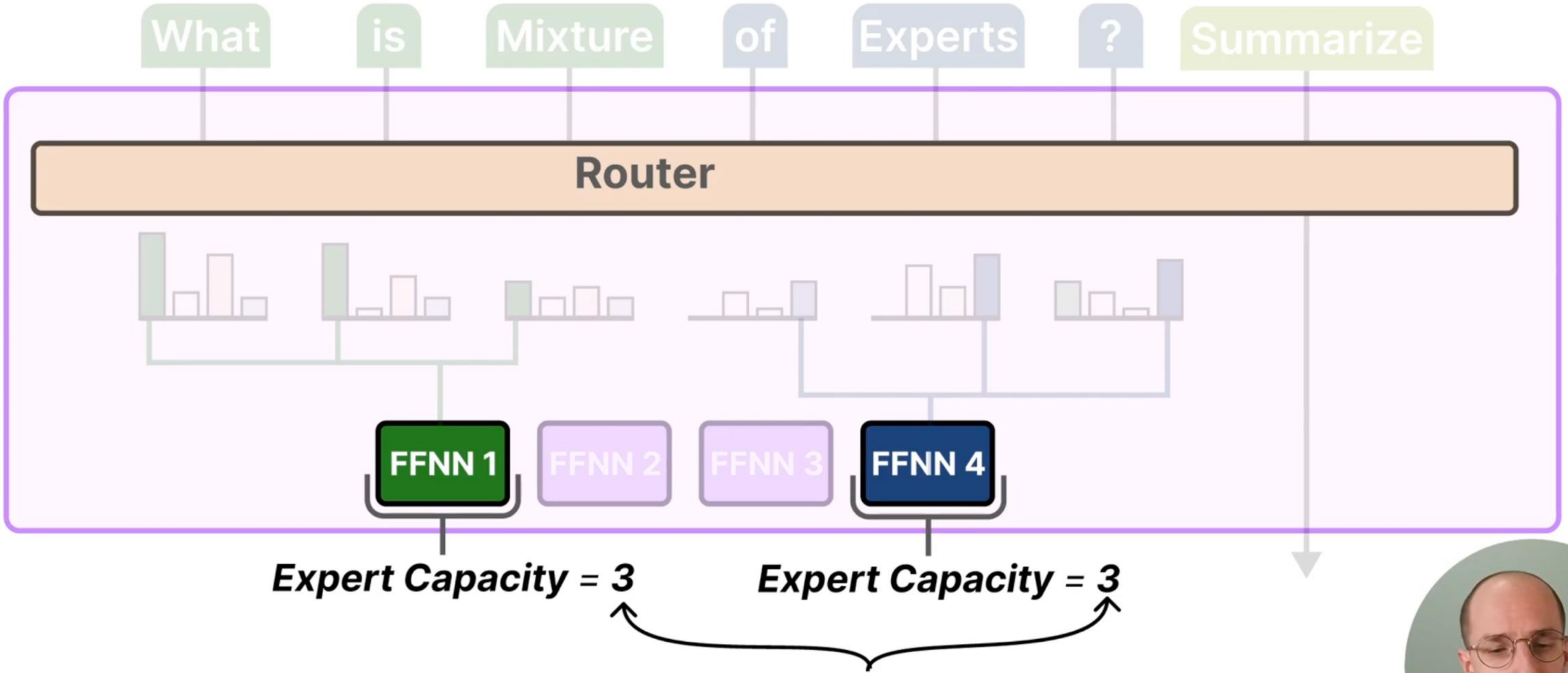
Expert Capacity：专家容量



Expert Capacity：专家容量

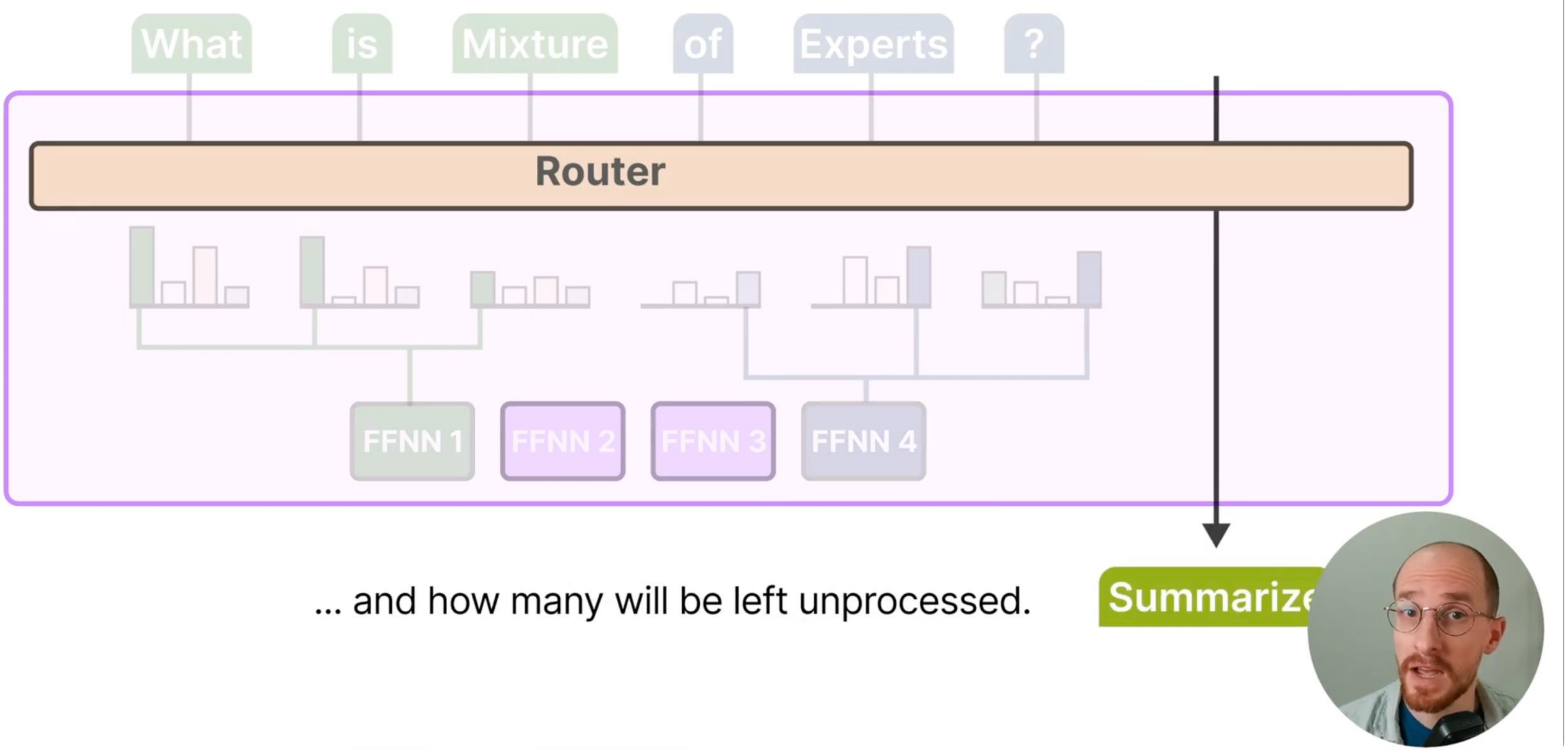


Expert Capacity：专家容量



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

Expert Capacity：专家容量



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.

ZOMI

GitHub https://github.com/chenzomi12/AIInfra

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/AIInfra
- 夸克链接：https://pan.quark.cn/s/74fb24be8eff