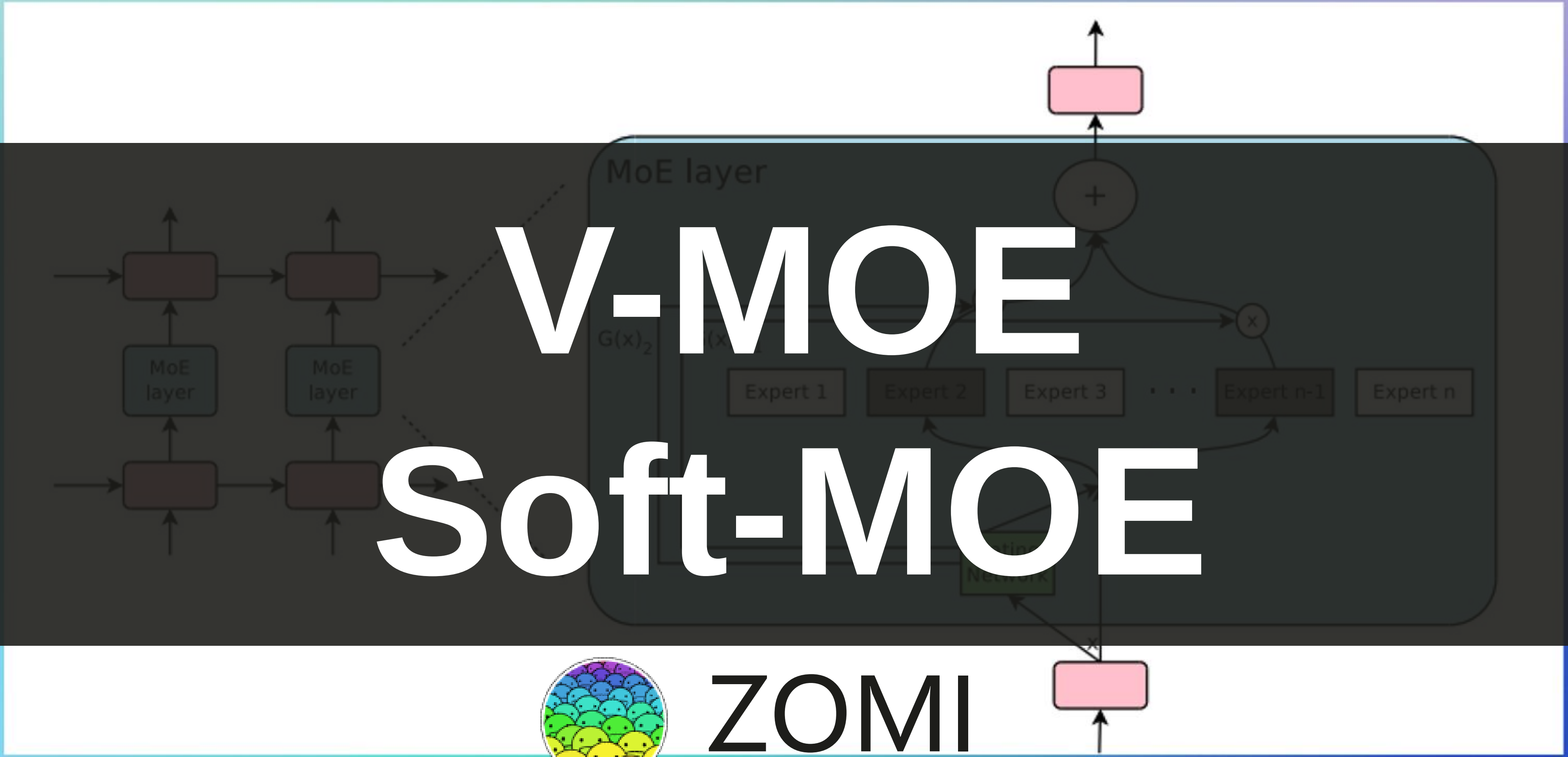
Mixture of Experts (MoE)



V-MOE
Soft-MOE

ZOMI

视频目录大纲

1. V-MOE 可视化原理
2. Soft-MOE 可视化原理

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 3
  - 大模型遇上 MoE 4
  - 手撕 MoE 代码 5
  - MoE 的未来
    - V-MoE：视觉和多任务 MoE
    - Soft MoE：可微的稀疏混合专家

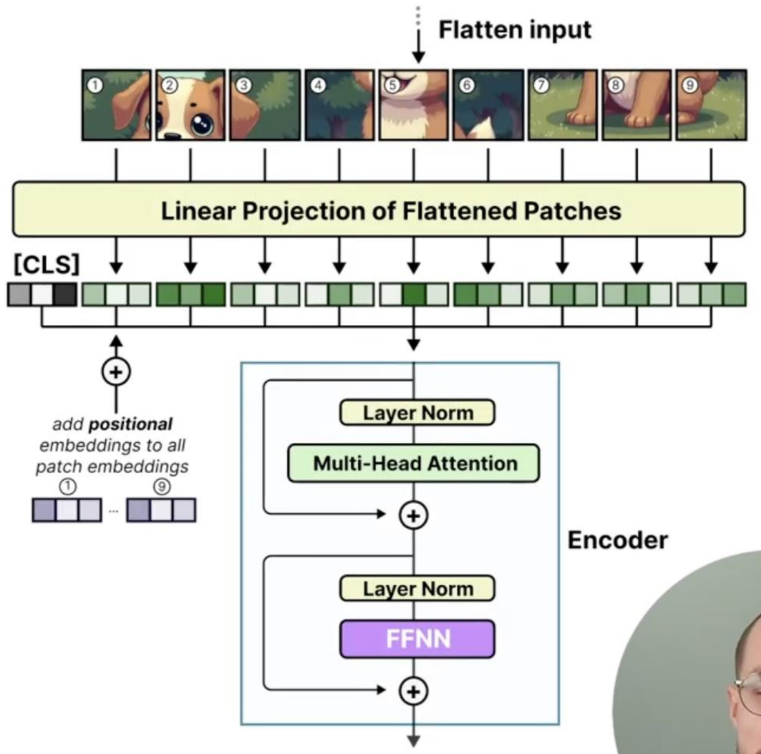
01

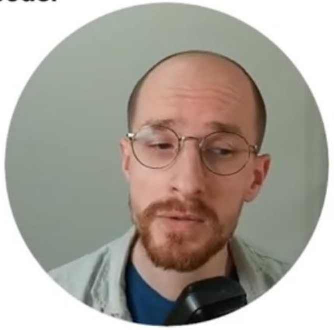
V-MOE 可视化原理

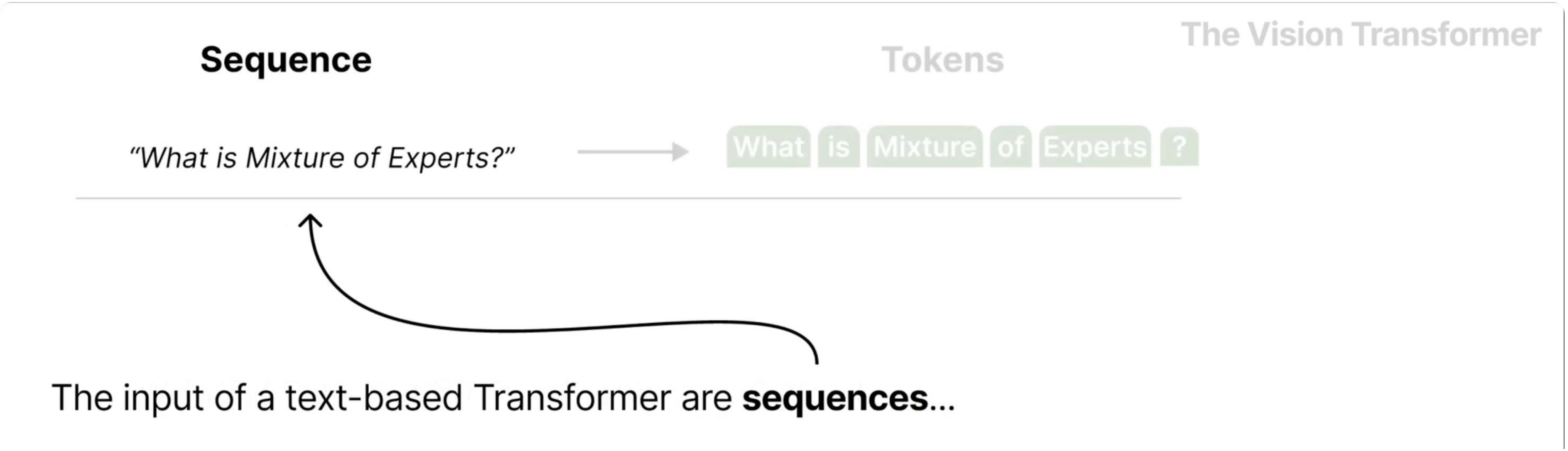
Now that we have explored the basics of Mixture of Experts in **Large Language Models**...

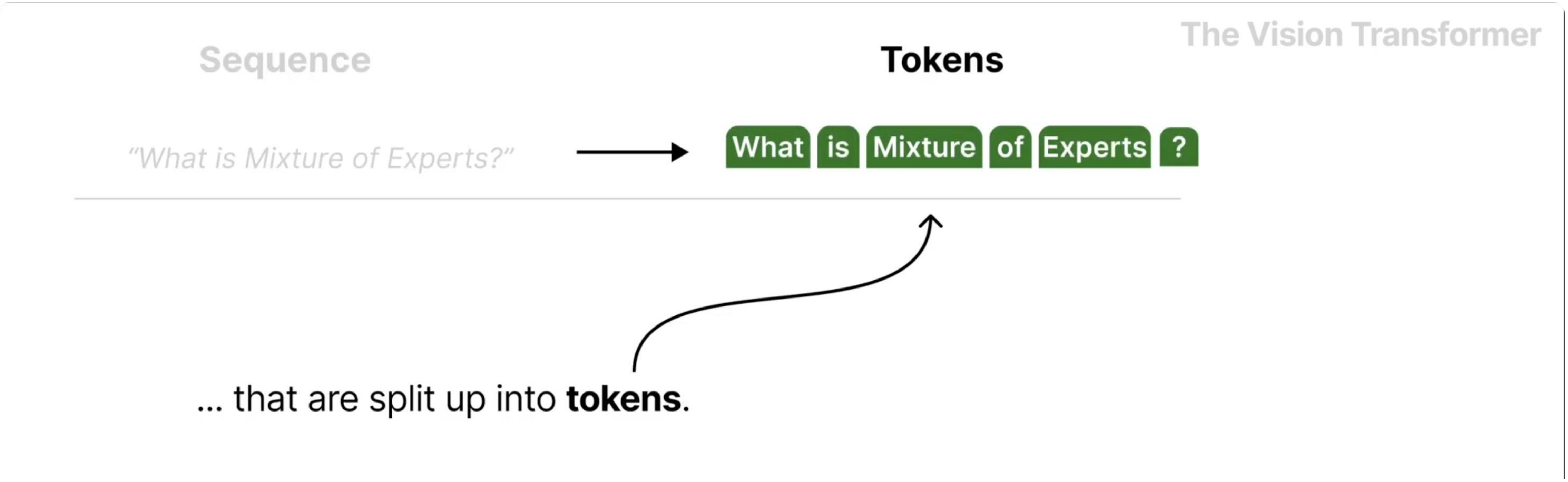
... we can do the same for **Vision Models**!

To explore MoE in vision models, let's recap the **Vision-Transformer** first.





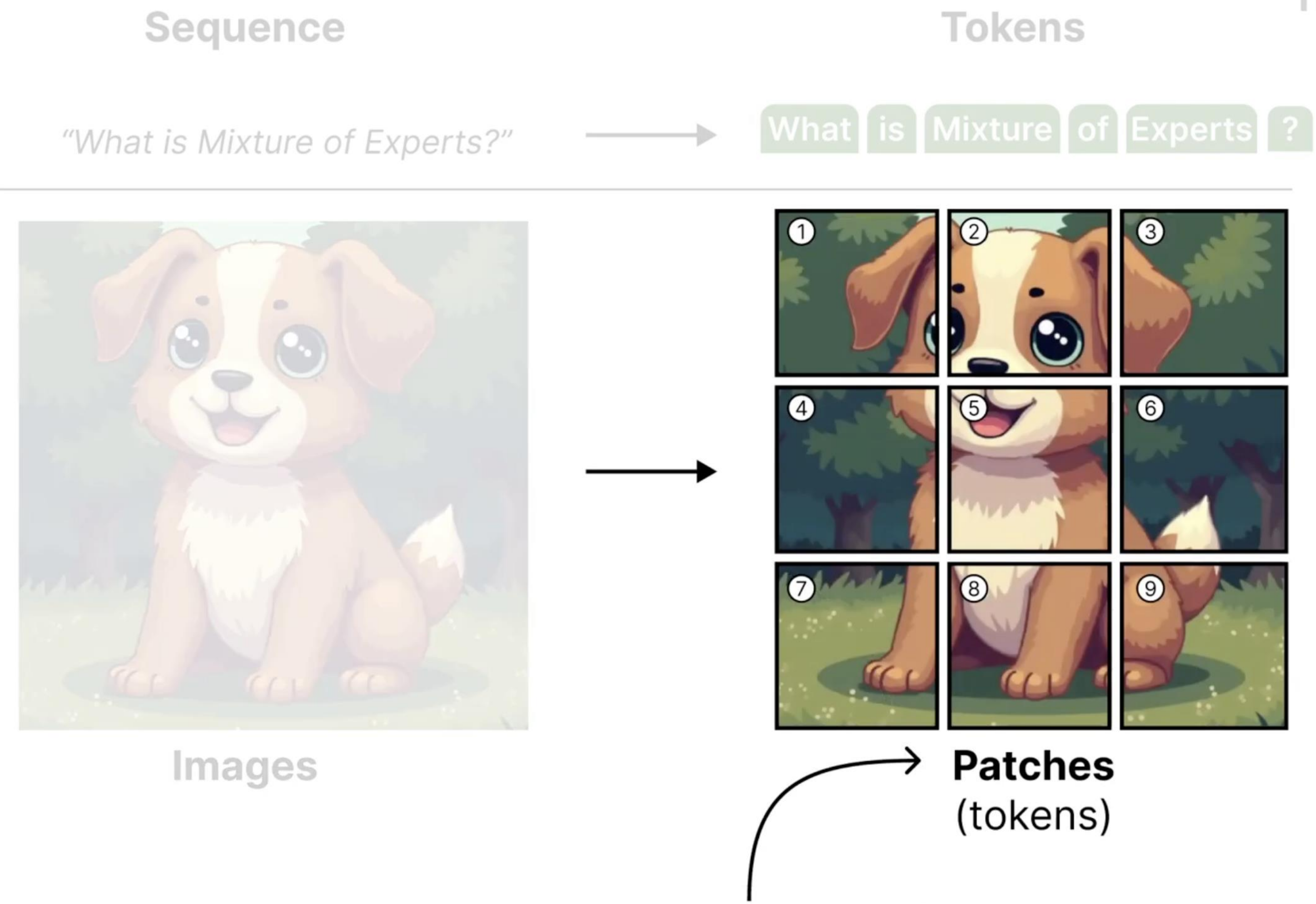

The Vision Transformer
Sequence
Tokens
"What is Mixture of Experts?"
What
is
Mixture
of
Experts
?
The input of a text-based Transformer are **sequences**...


The Vision Transformer
Sequence
Tokens
"What is Mixture of Experts?"
What
is
Mixture
of
Experts
?
... that are split up into **tokens**.

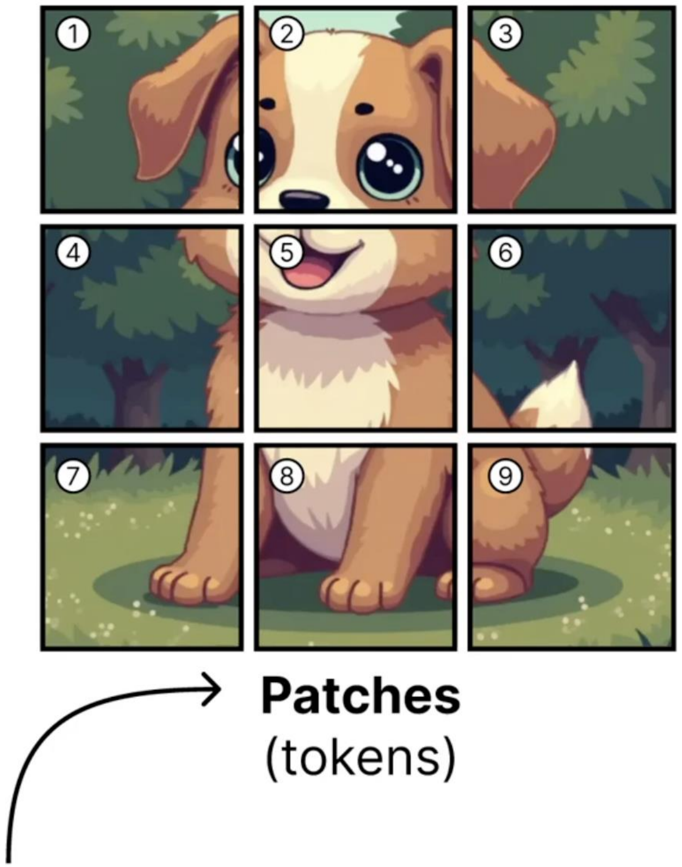
Sequence	Tokens
"What is Mixture of Experts?" →	What · is · Mixture · of · Experts · ?

Images

To perform the same tokenization process with **images**...



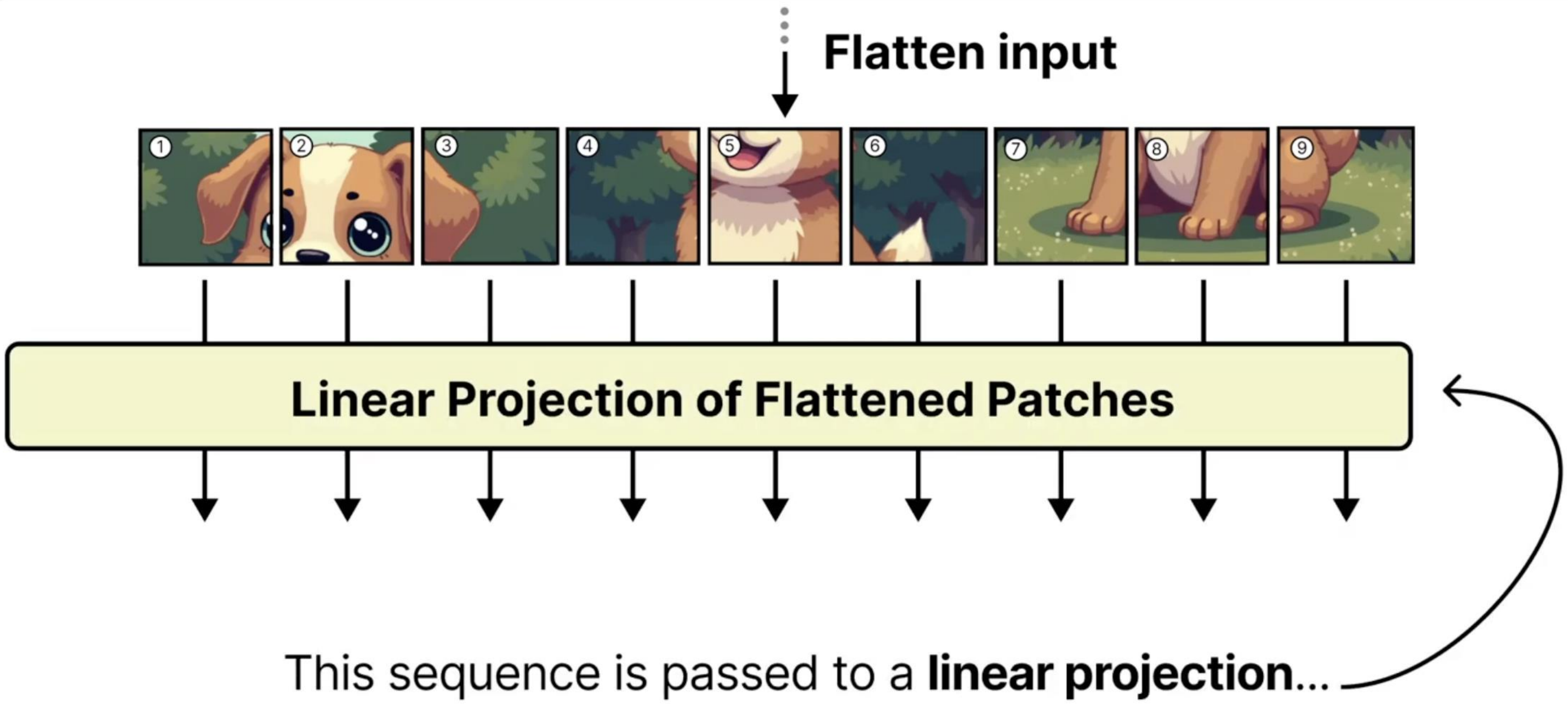
... we instead convert them into **patches** (or also called tokens).

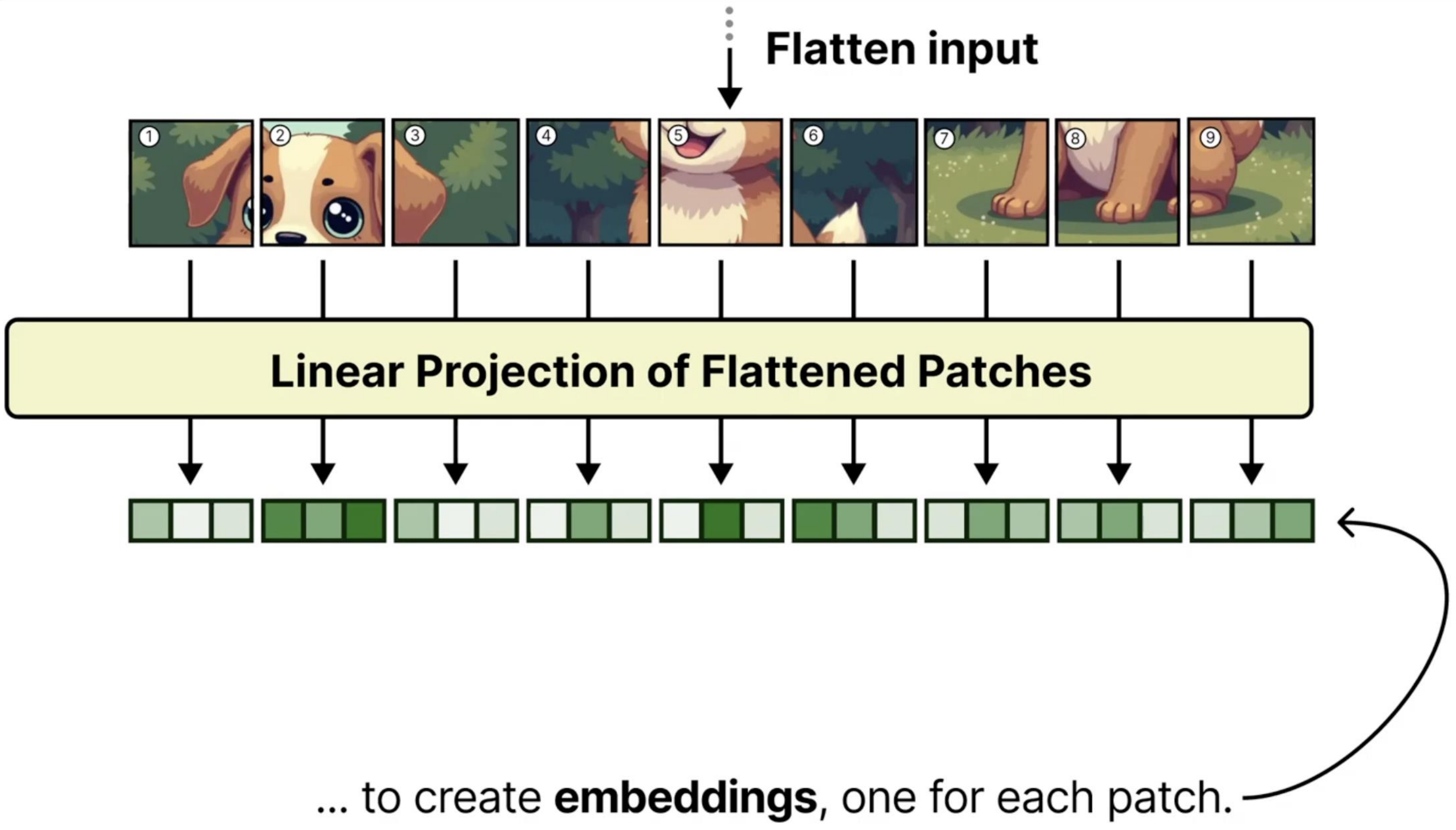


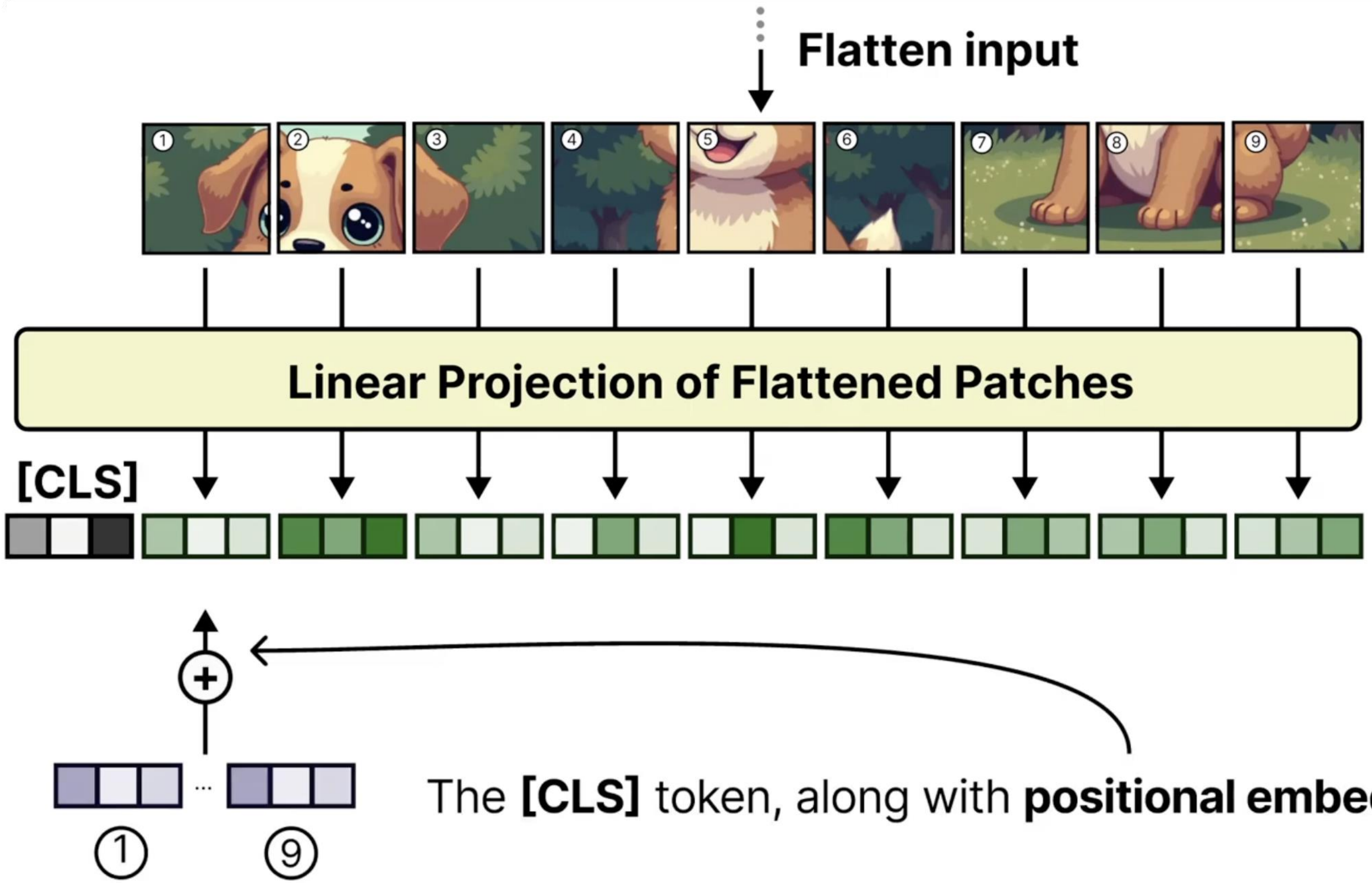
To further process these patches...

Flatten input

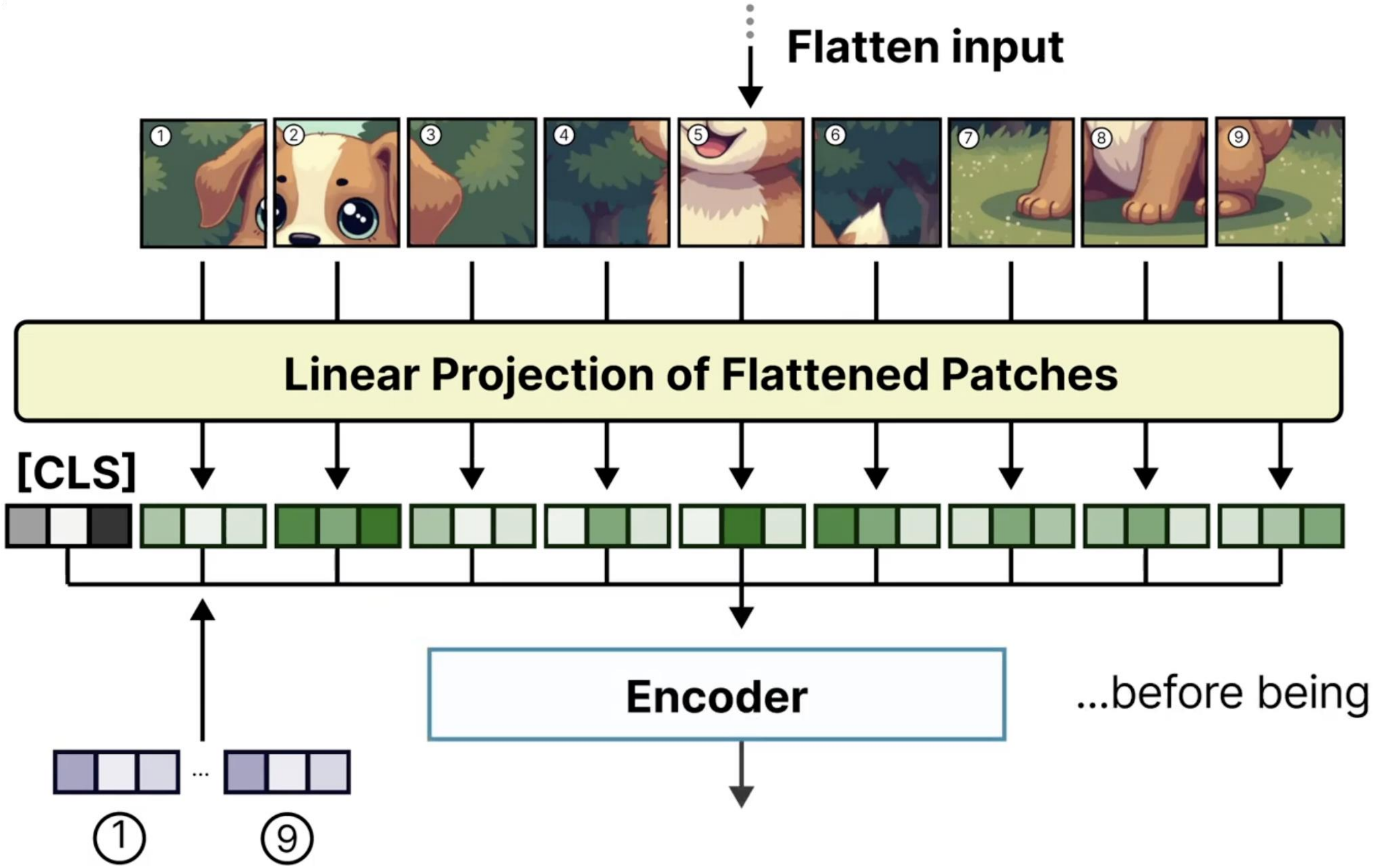
... we flatten them into a **sequence of patches**.


Flatten input
1
2
3
4
5
6
7
8
9
Linear Projection of Flattened Patches
This sequence is passed to a **linear projection**...

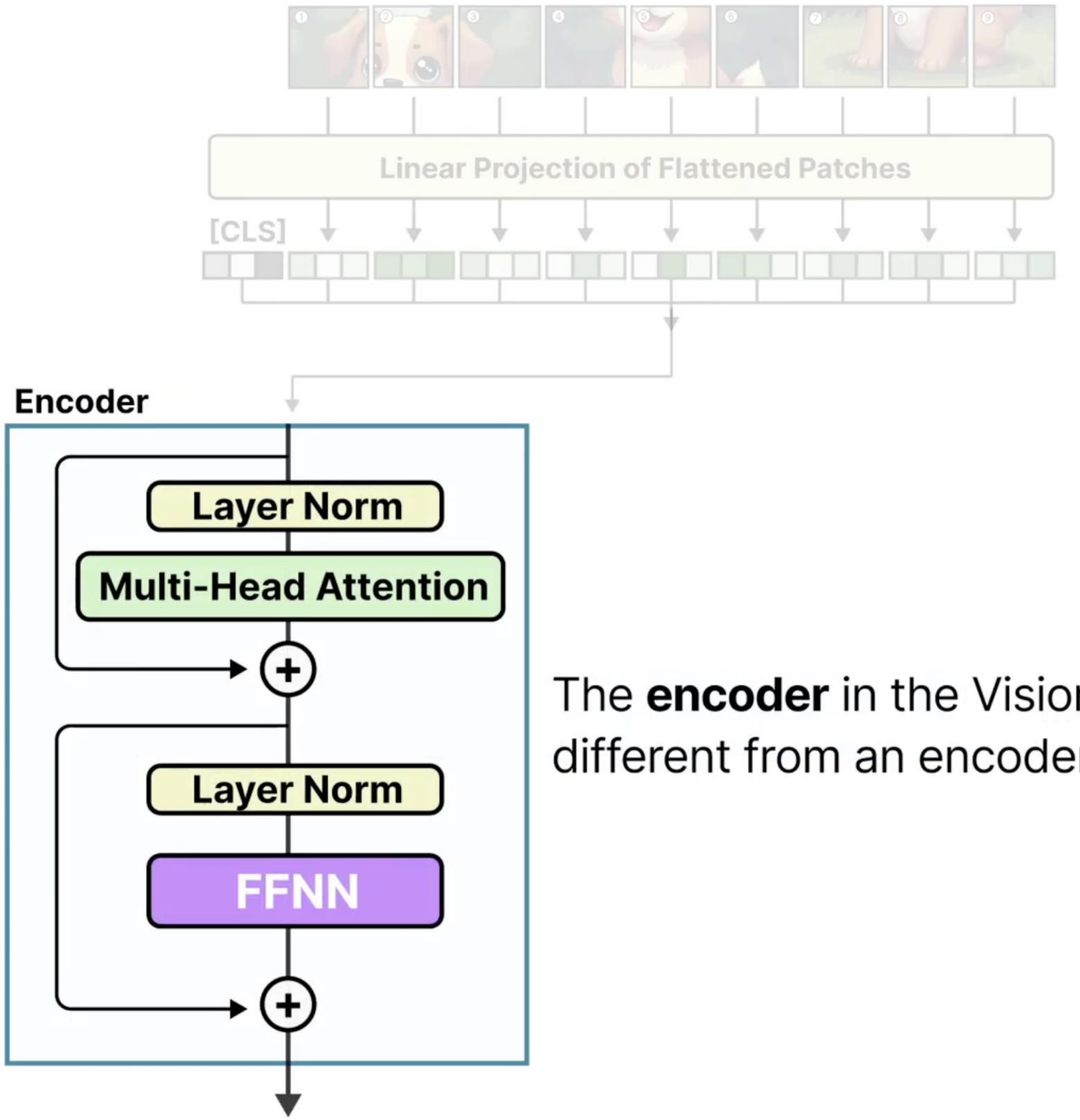

Flatten input
1
2
3
4
5
6
7
8
9
Linear Projection of Flattened Patches
... to create **embeddings**, one for each patch.



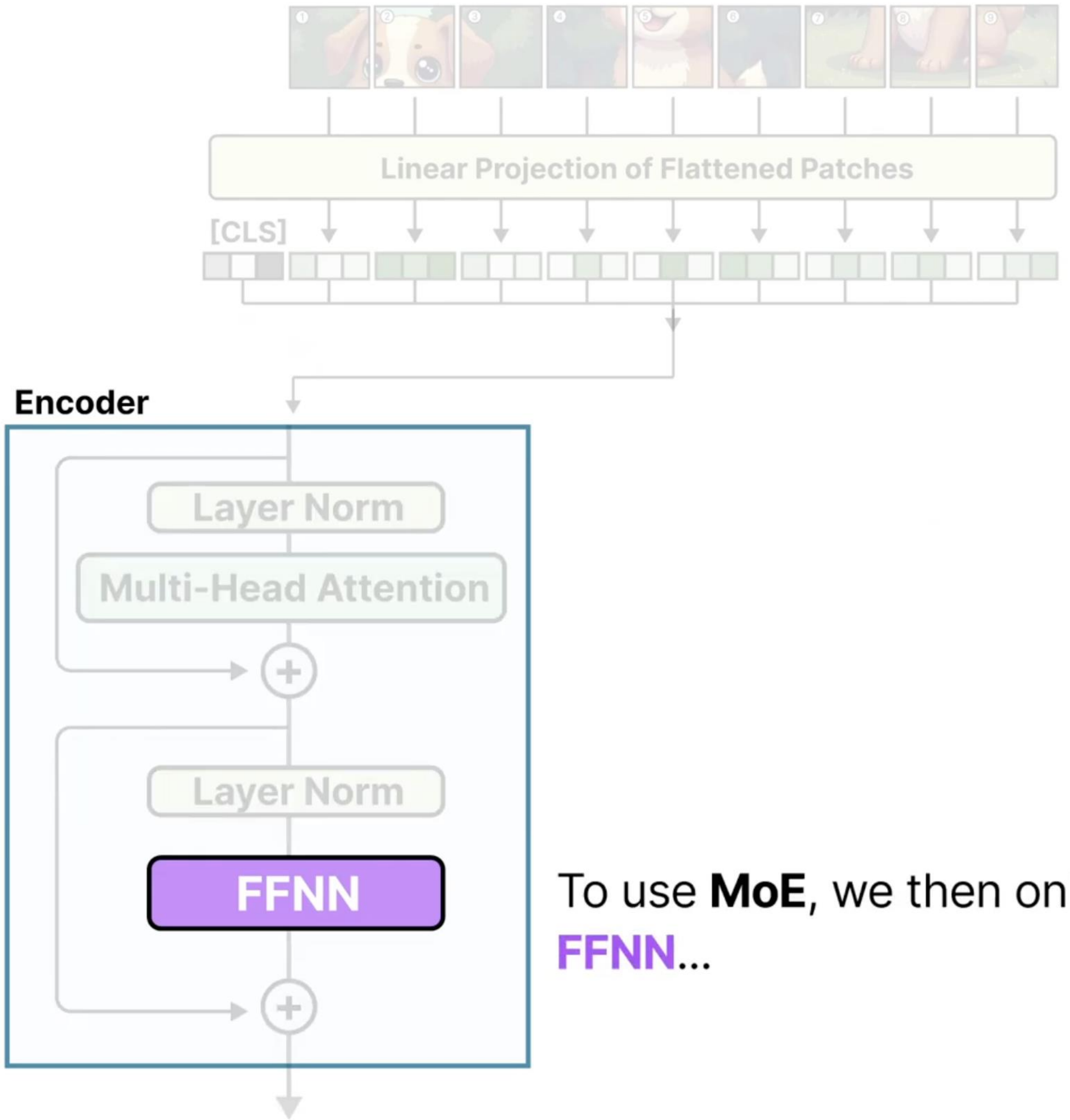
The **[CLS]** token, along with **positional embeddings** are added...



...before being passed to the **encoder**.

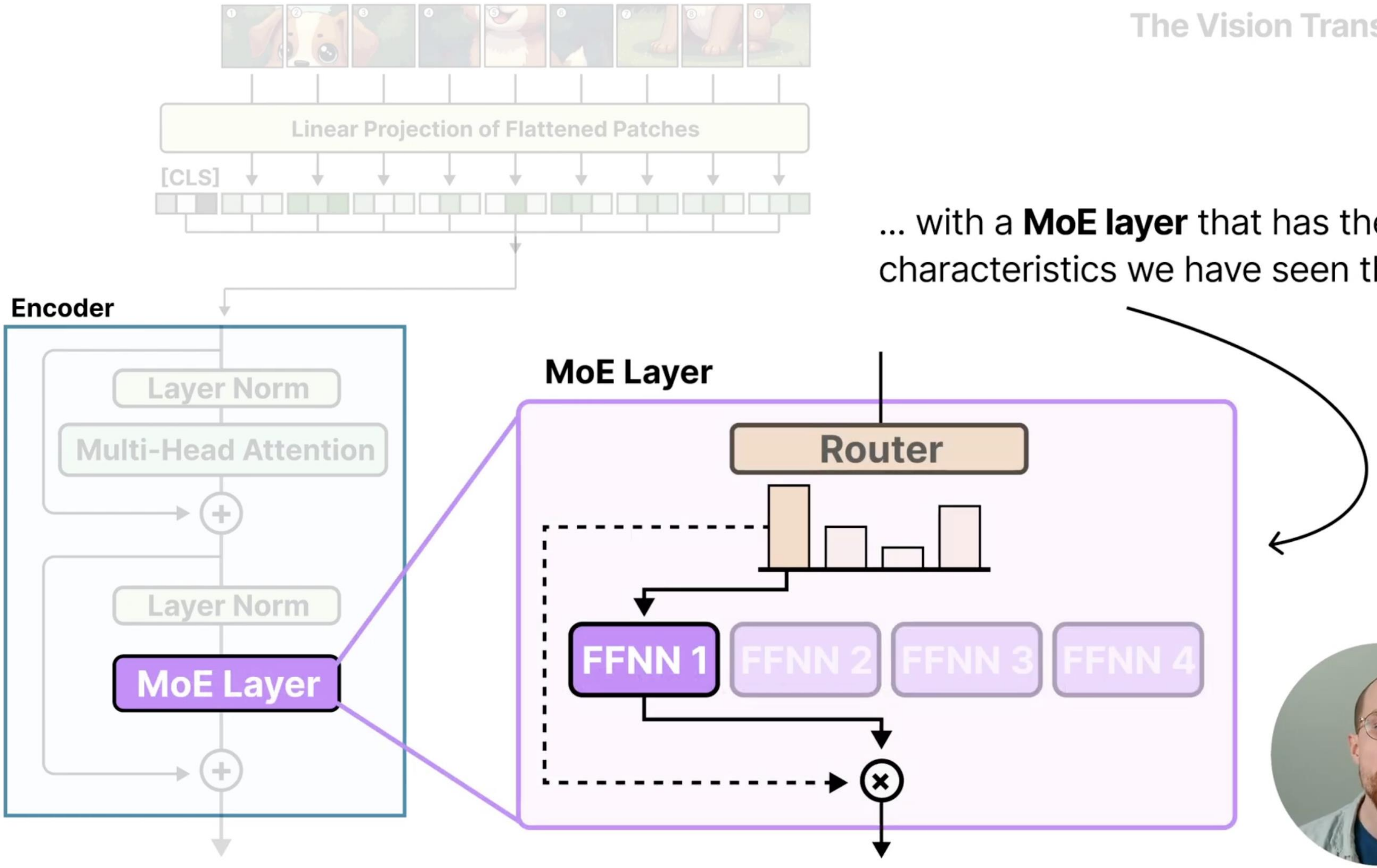


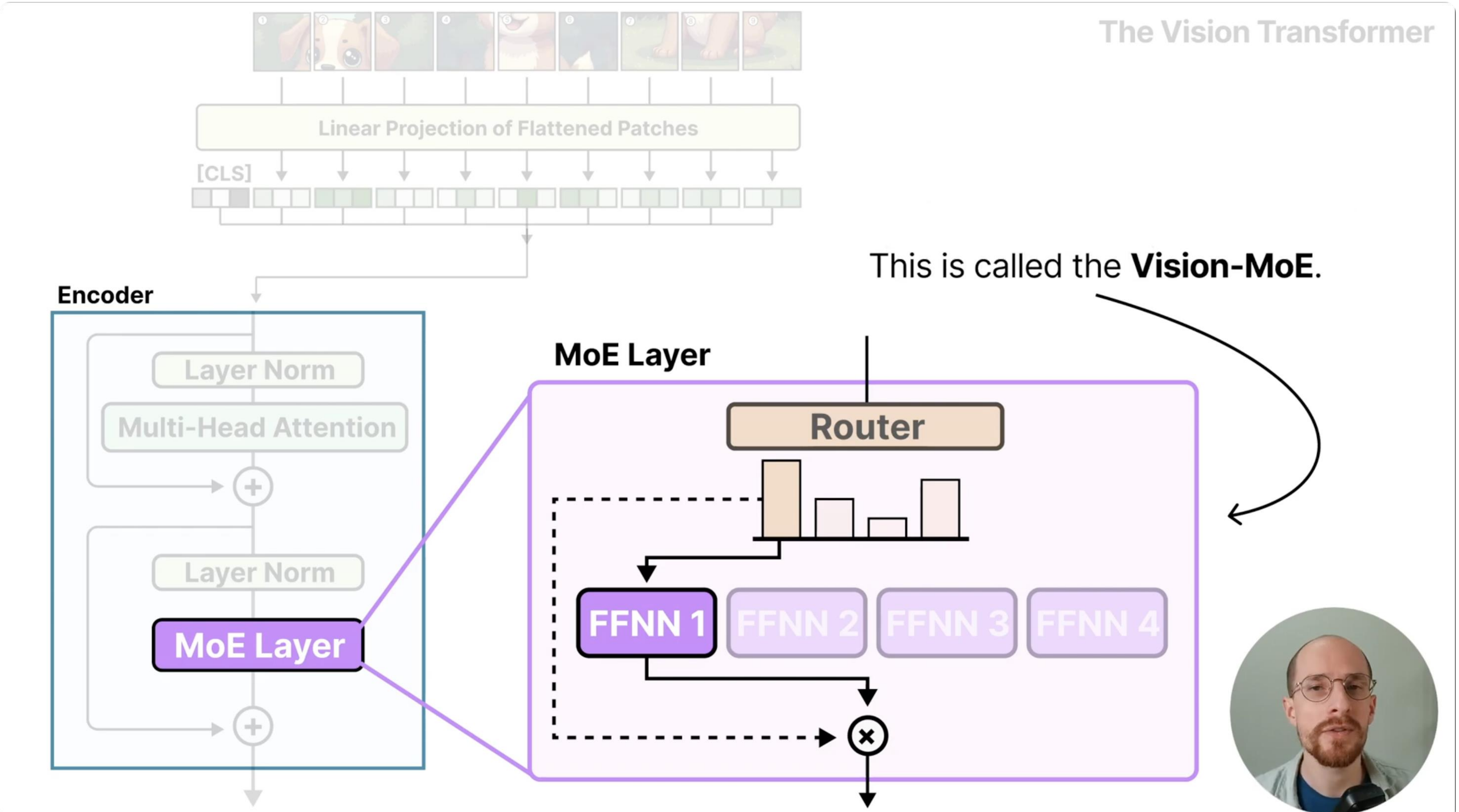
The **encoder** in the Vision Transformer, is no different from an encoder that processes text.

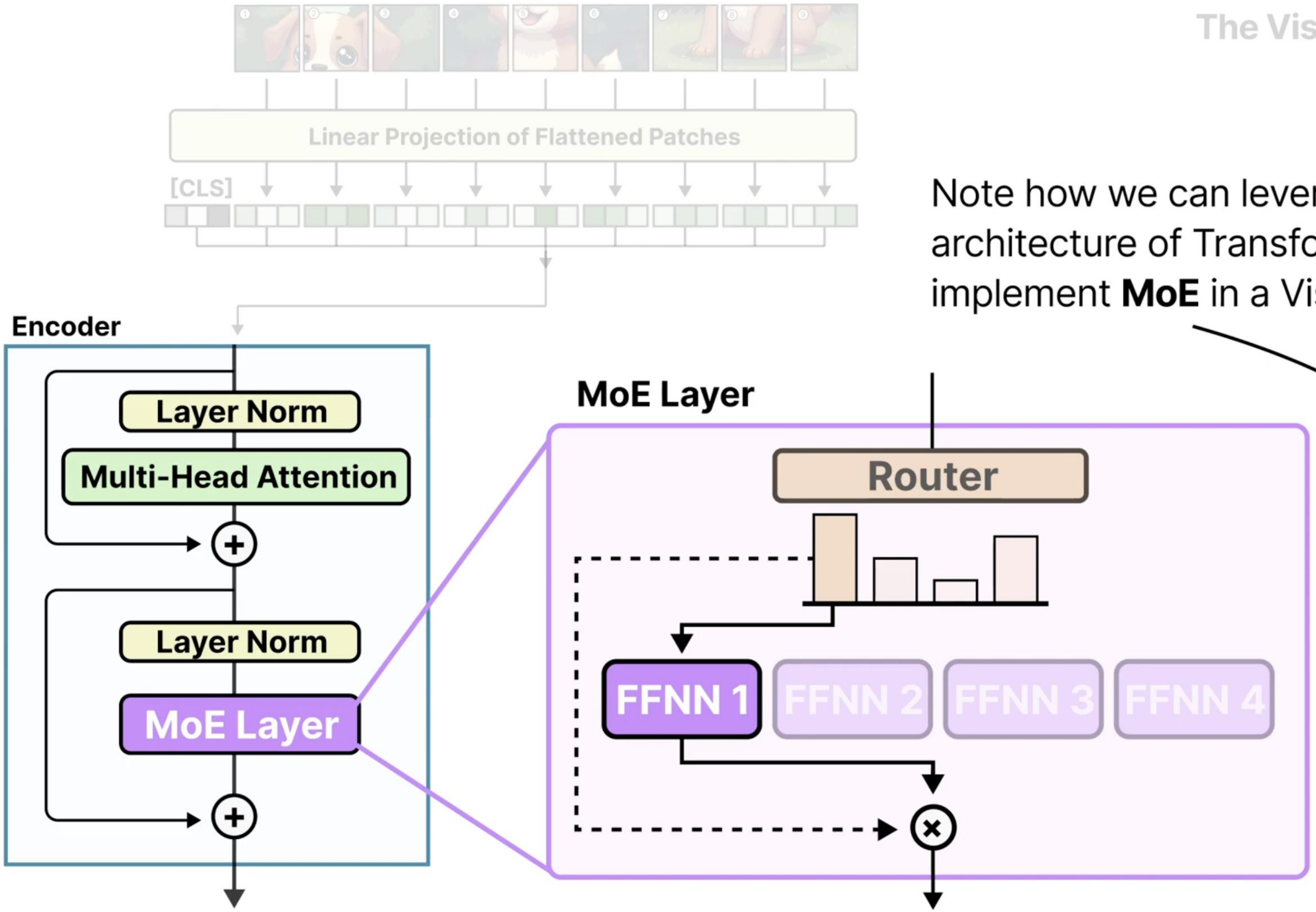


To use **MoE**, we then only need to replace the **FFNN**...

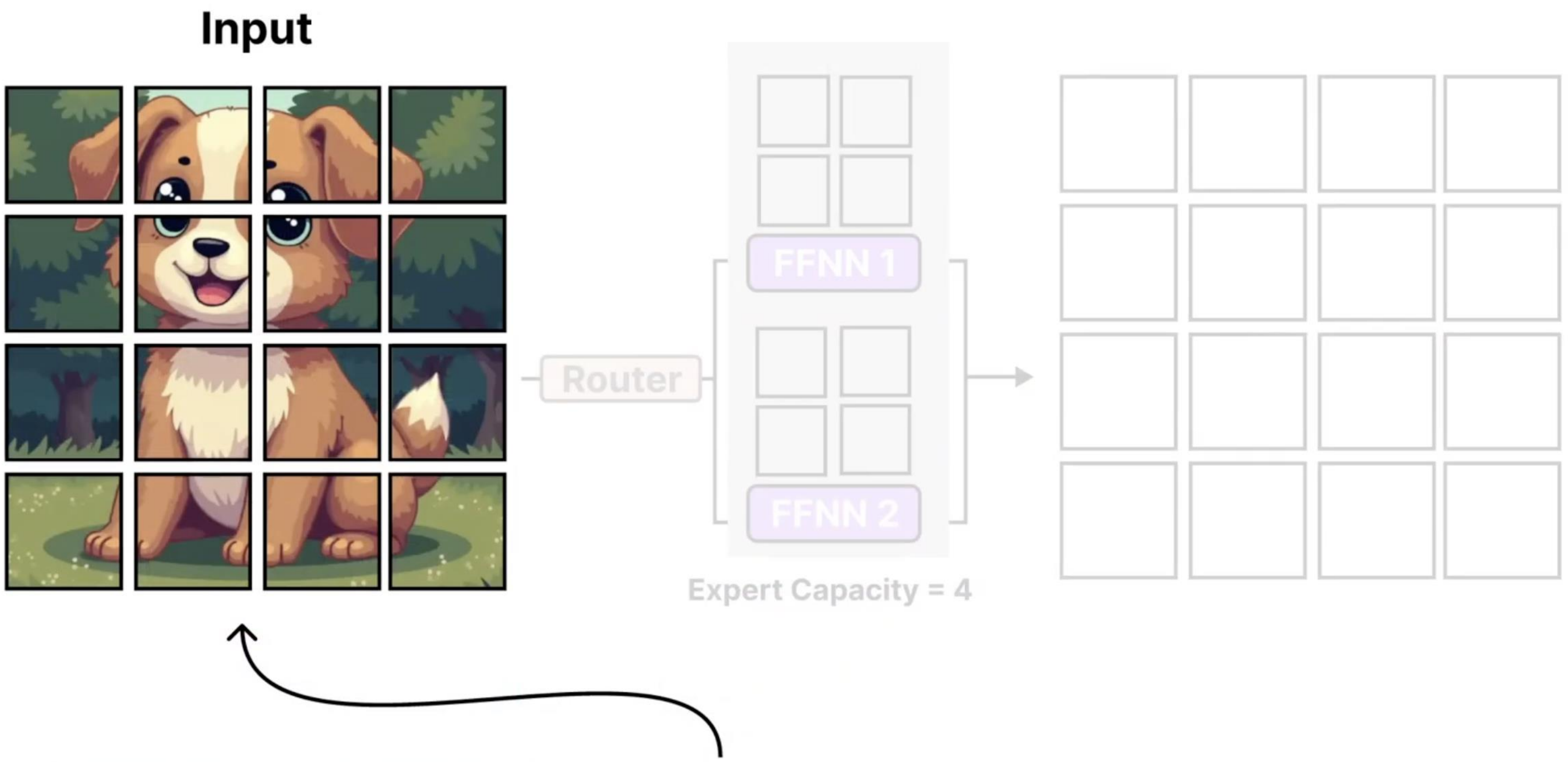
... with a **MoE layer** that has the same characteristics we have seen thus far.



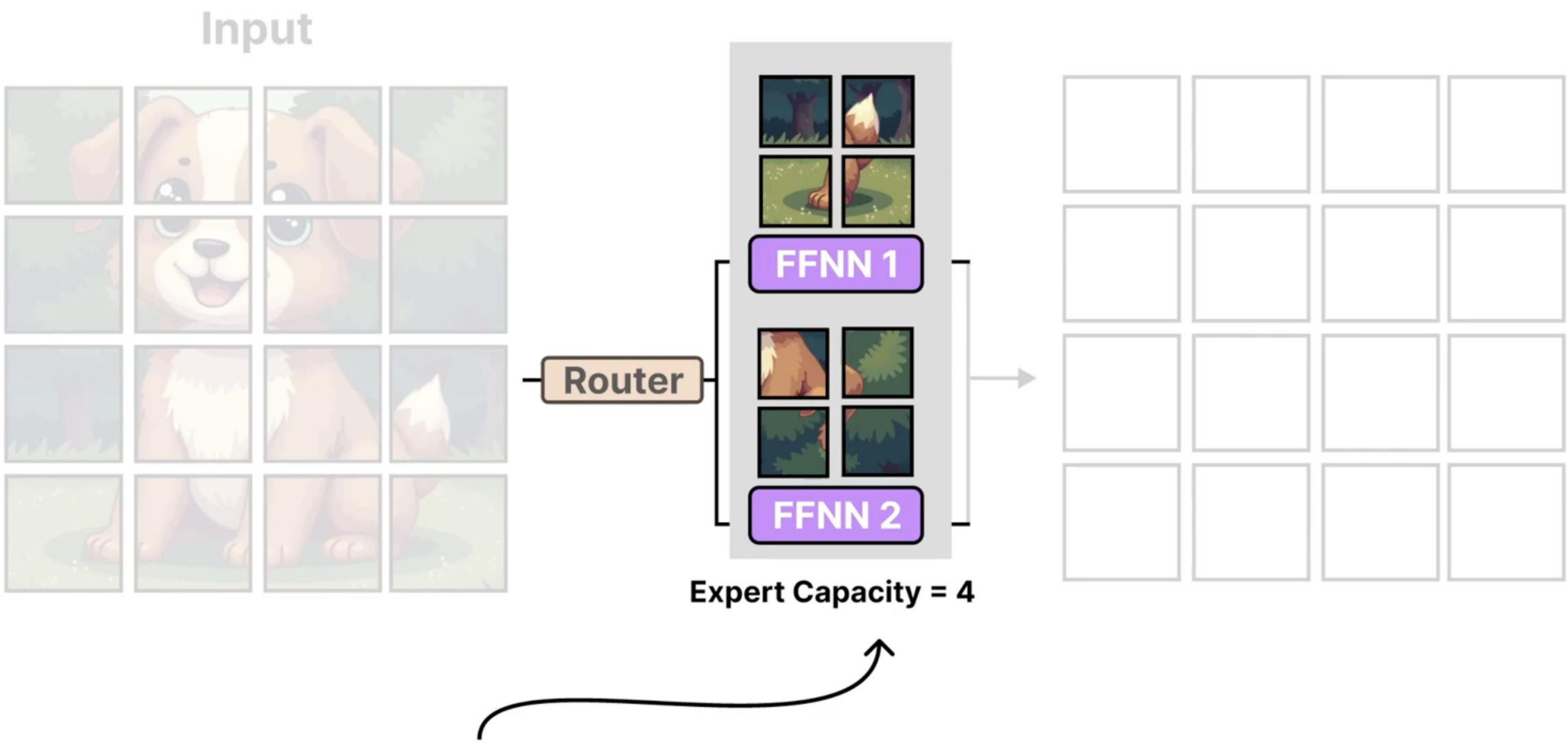

The Vision Transformer
Linear Projection of Flattened Patches
[CLS]
Encoder
Layer Norm
Multi-Head Attention
Layer Norm
MoE Layer
This is called the **Vision-MoE**.
MoE Layer
Router
FFNN 1
FFNN 2
FFNN 3
FFNN 4



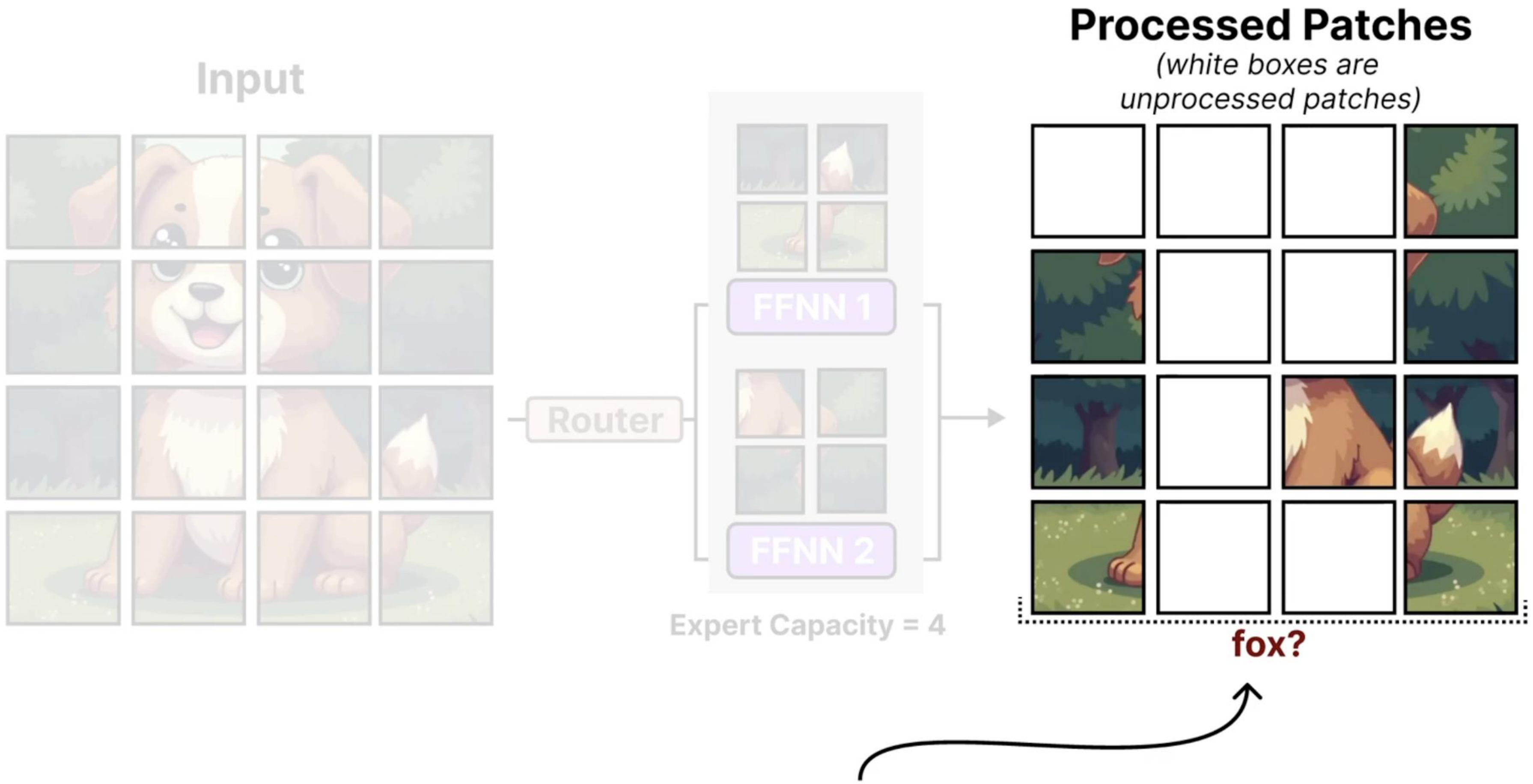
Note how we can leverage the existing architecture of Transformers to implement **MoE** in a Vision Model.



Since images generally have many patches...

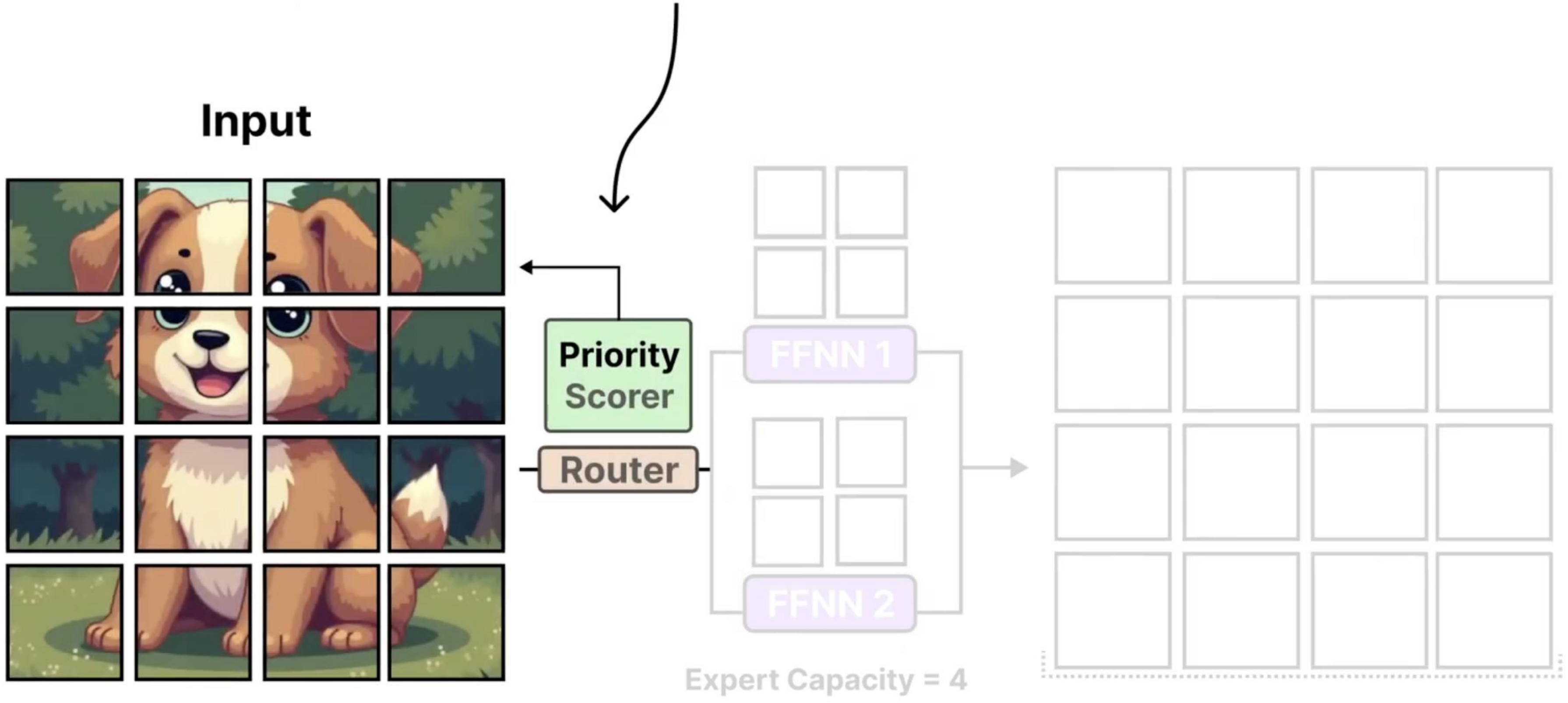


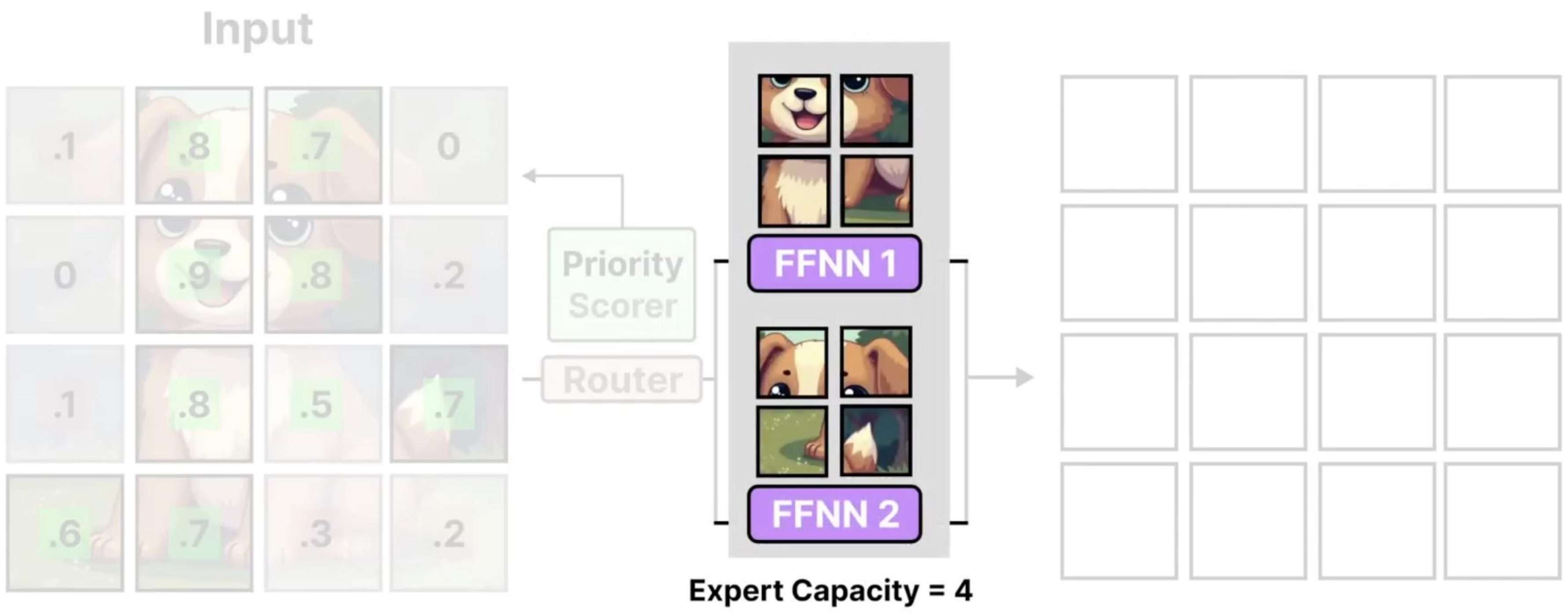
... a low **expert capacity** is used for each expert to reduce hardware constraints.



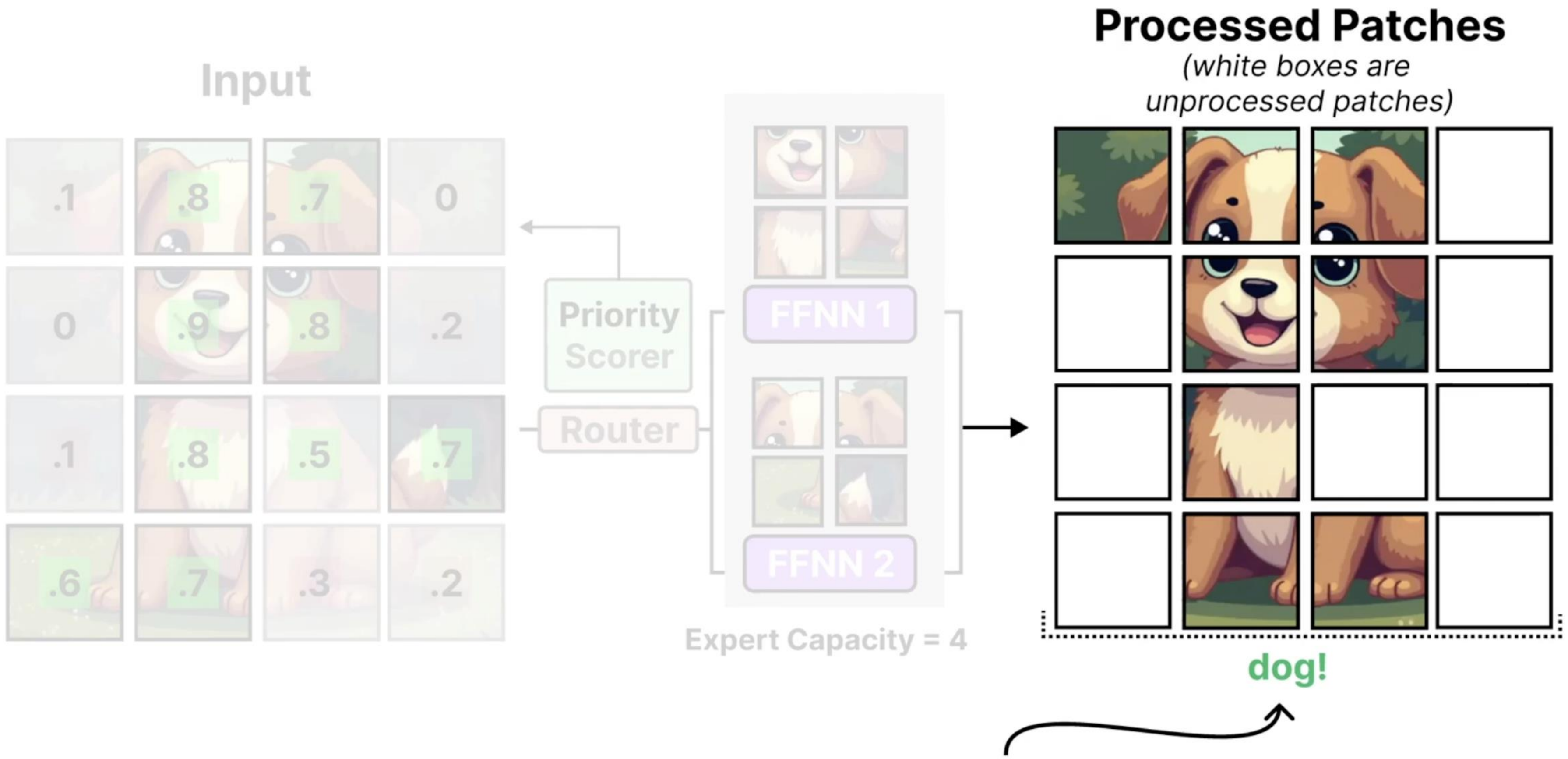
However, a low capacity tends to lead to patches being dropped (akin to **token overflow**).

To keep the capacity low, a **priority scorer**...

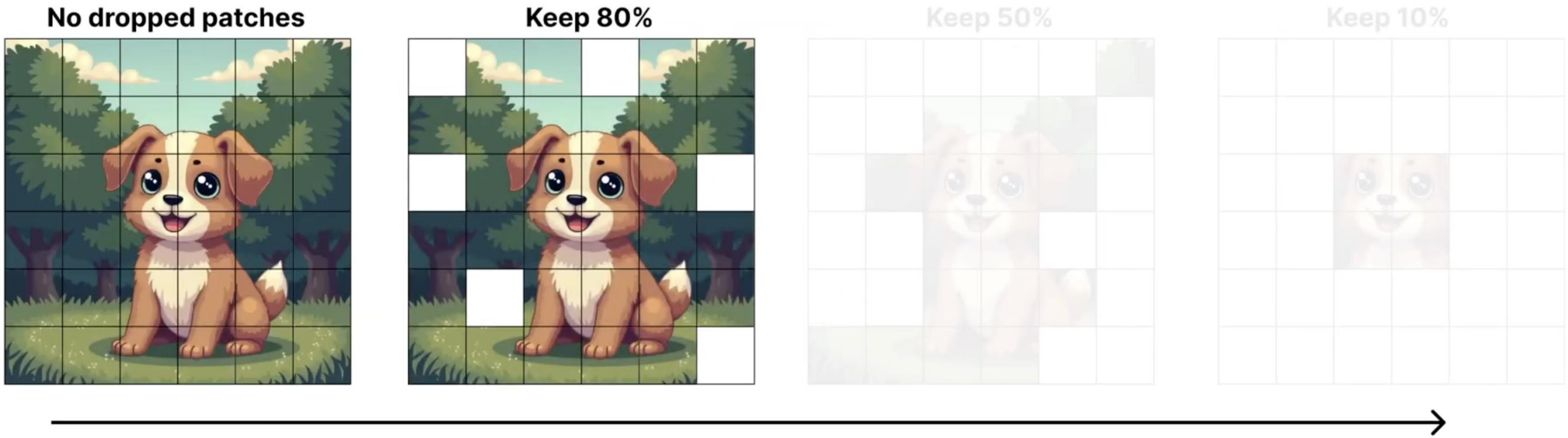




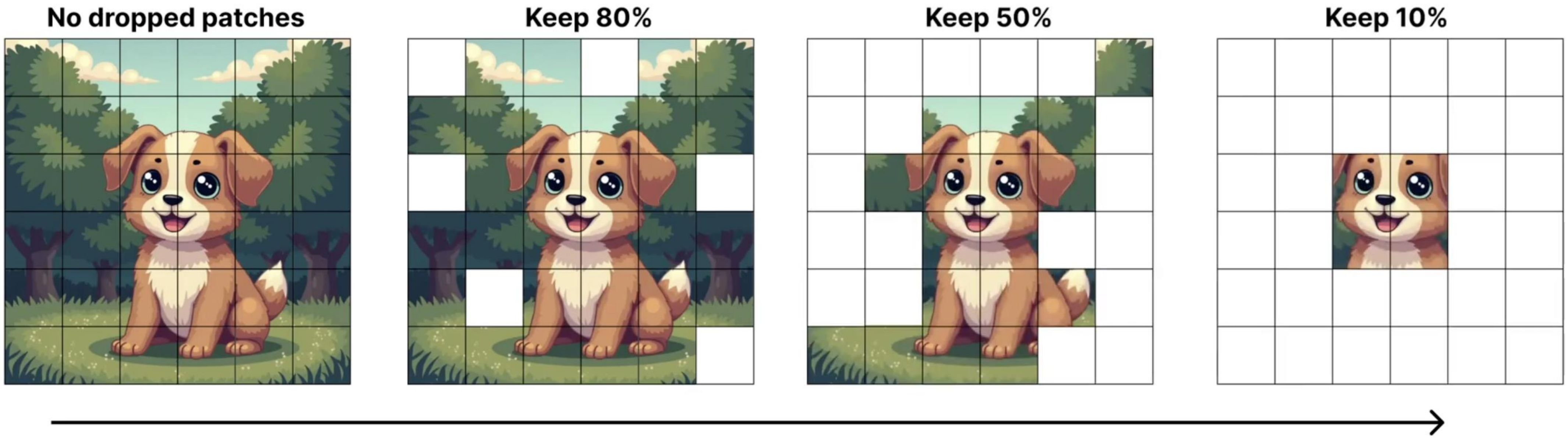
... so that the most important patches are processed first.



This results in a much more accurate representation of the original image.



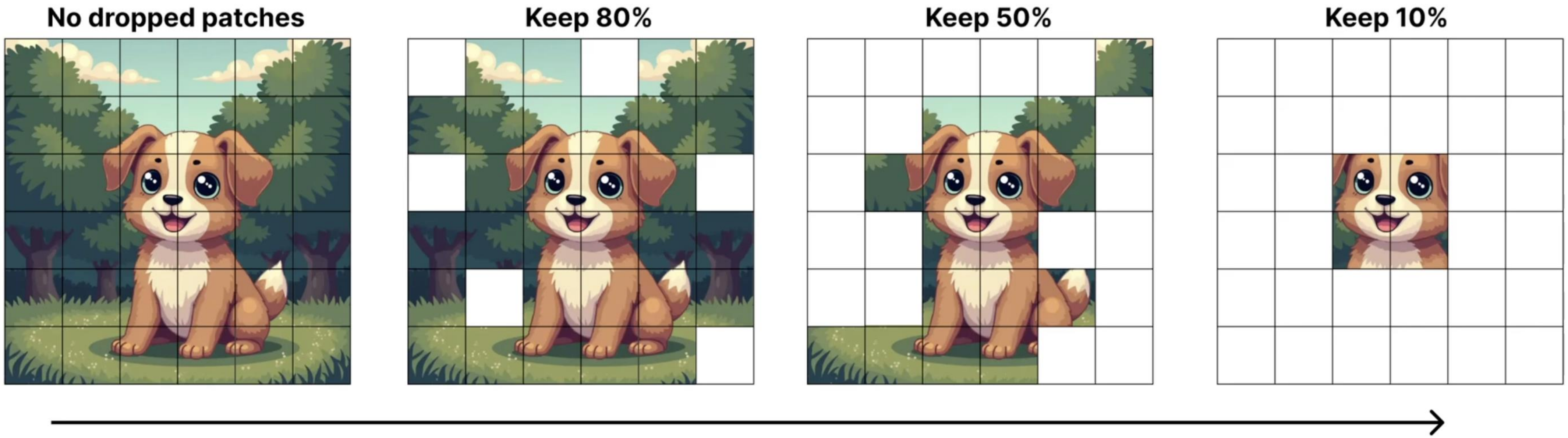
As a result, we should still see **important patches** routed if the percentage of tokens decreases.



As a result, we should still see **important patches** routed if the percentage of tokens decreases.

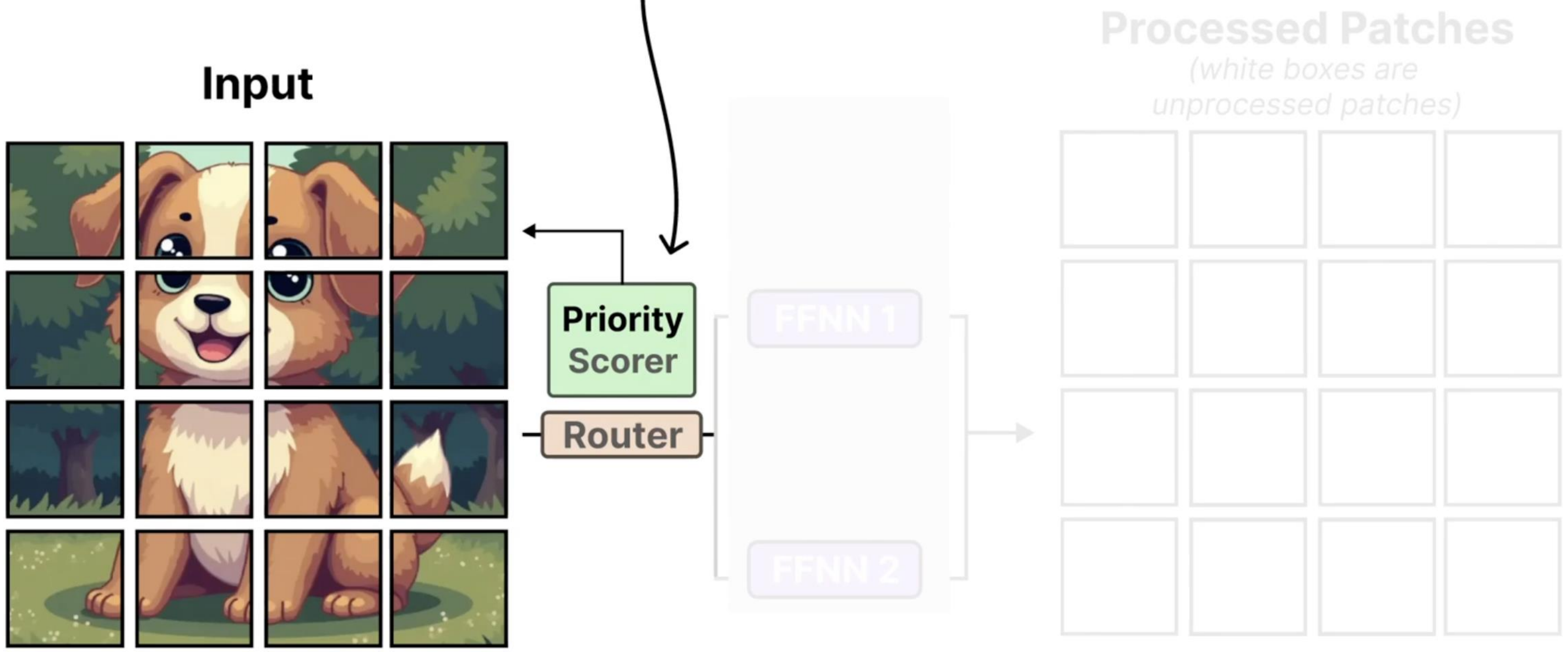
02

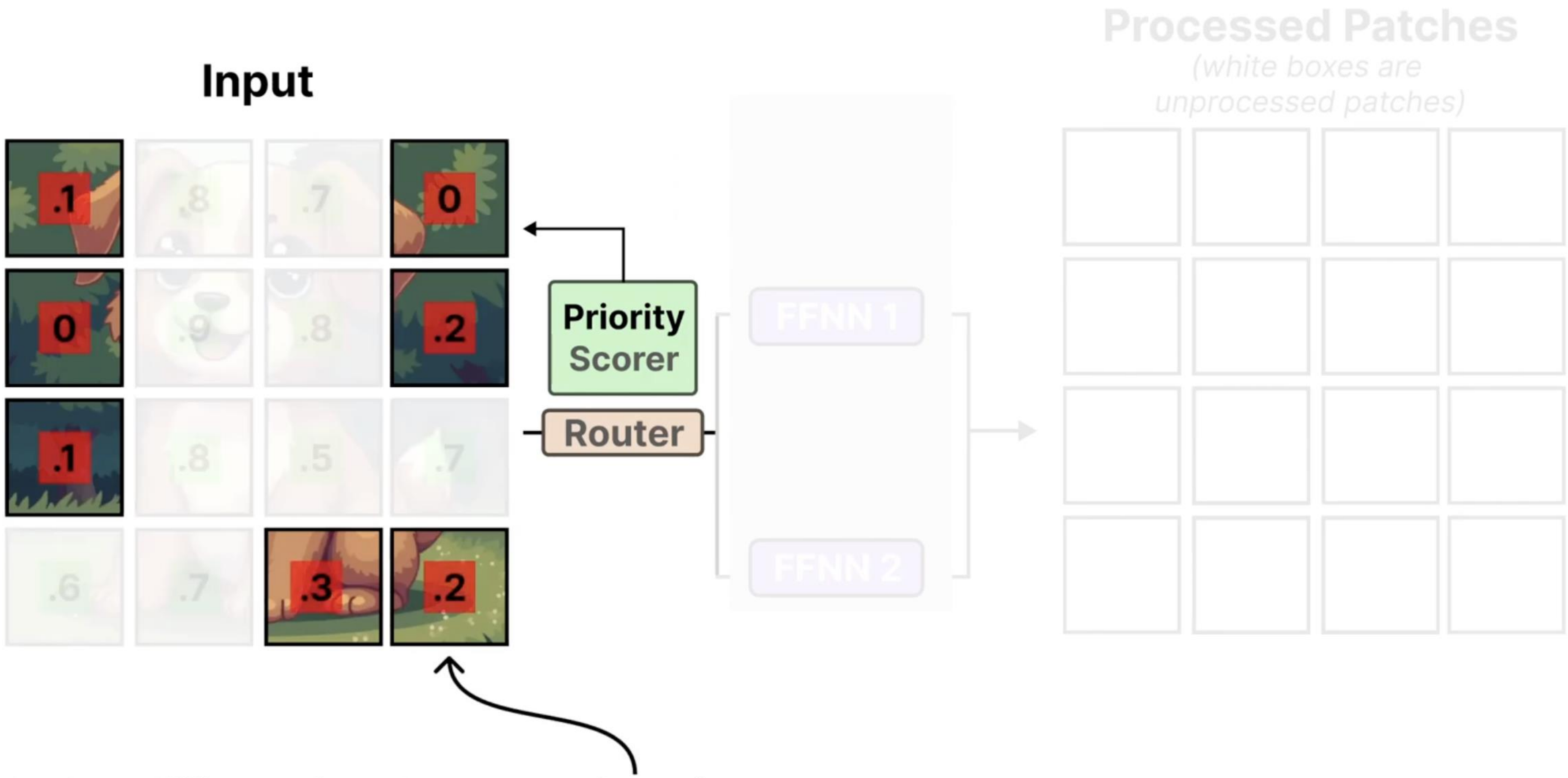
Soft-MOE 可视化原理



However, **Vision-MoE** still needs to drop tokens, so information is lost. Instead, let's look at an alternative, namely **Soft-MoE.**

In Vision-MoE, the **priority scorer**...

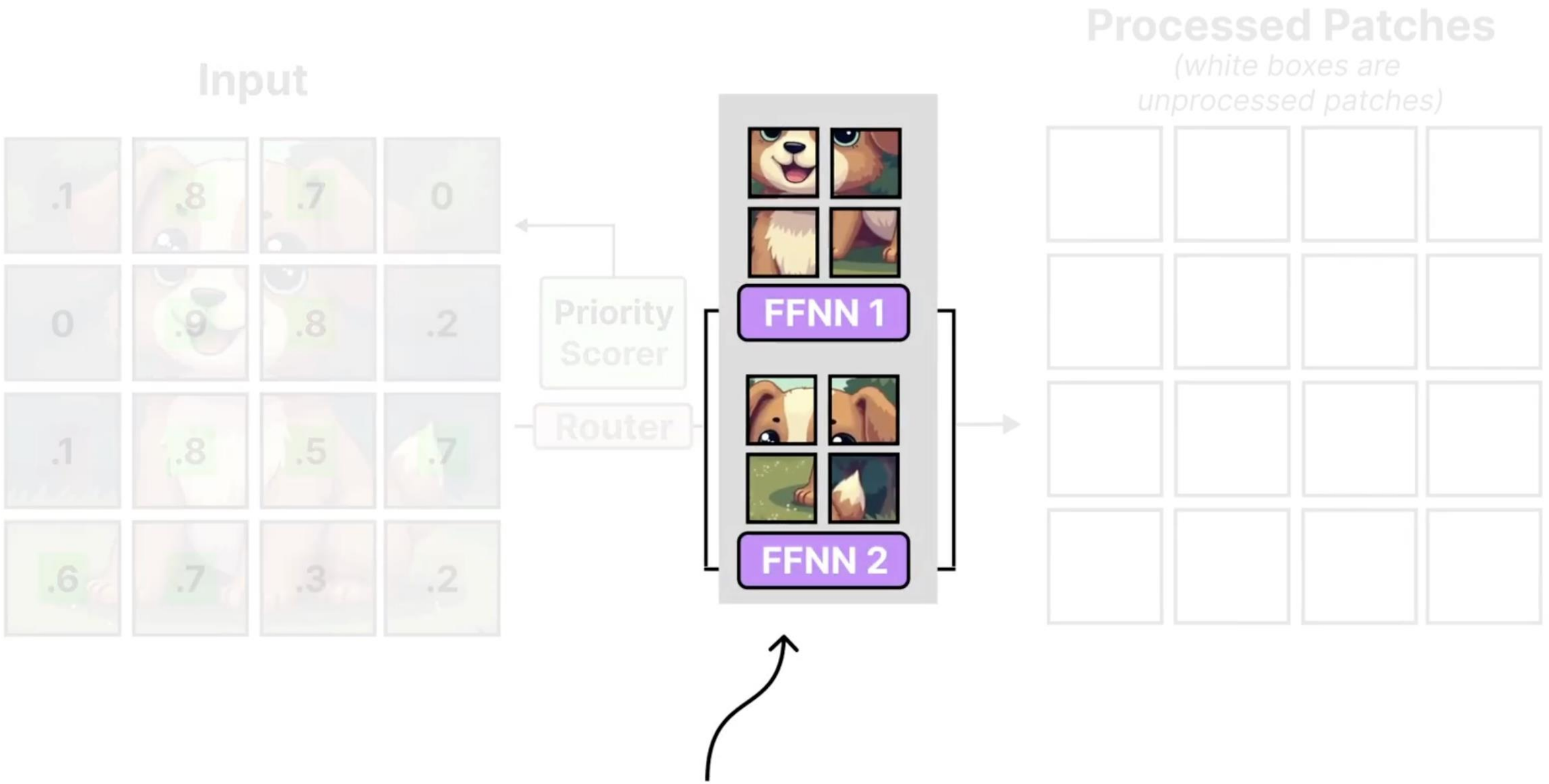




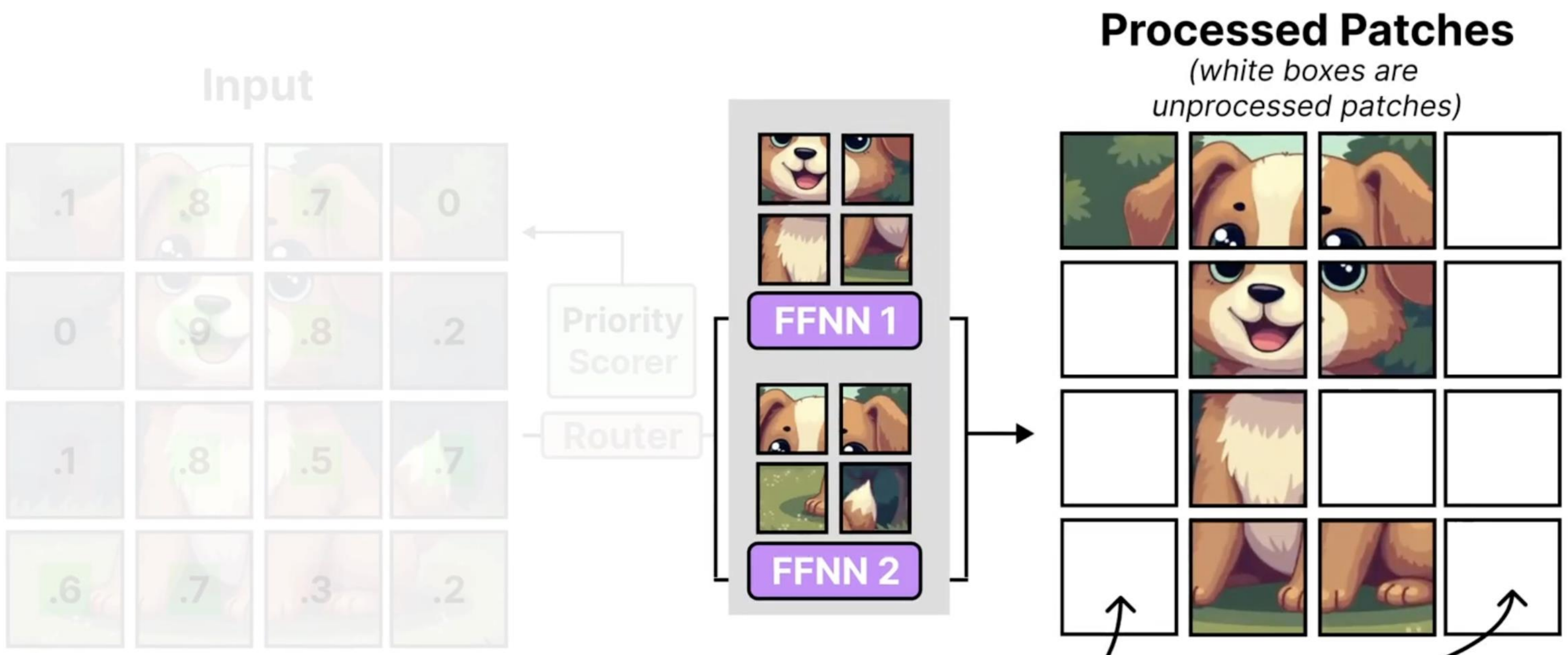
... helps differentiate between **less important** patches...

From Sparse to Soft MoE

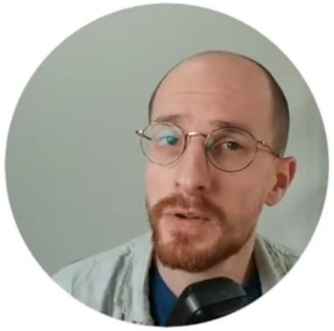
... and **more important** patches.

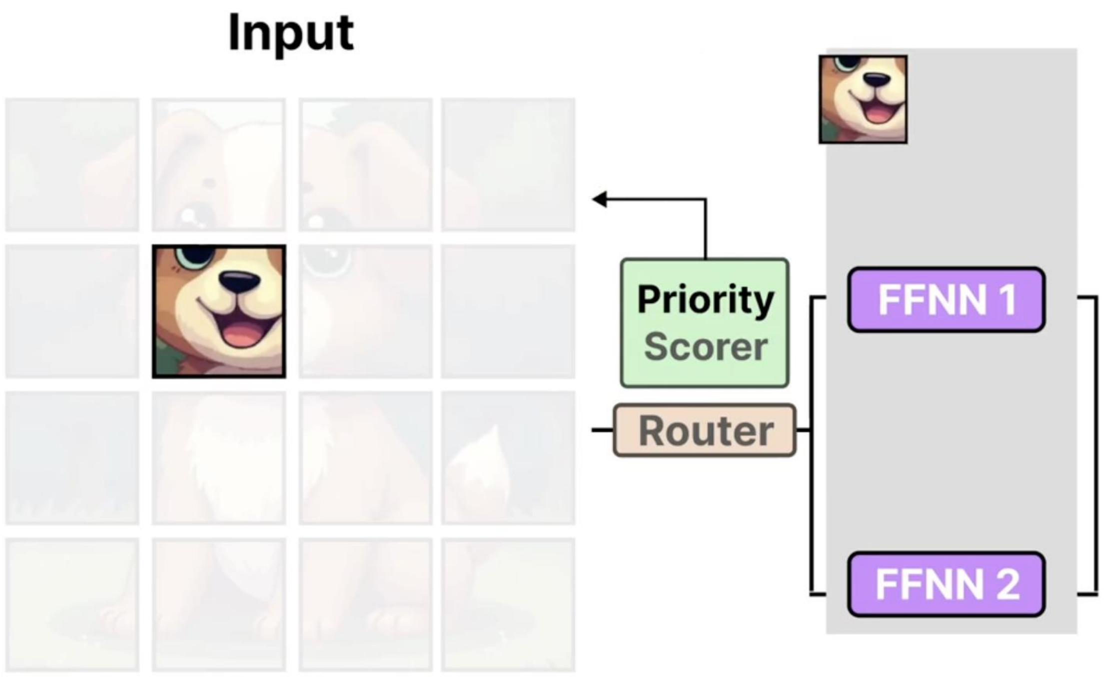


However, a **subset** of patches are assigned to each expert...

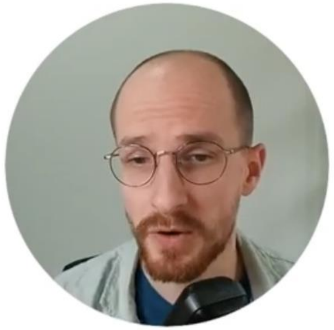


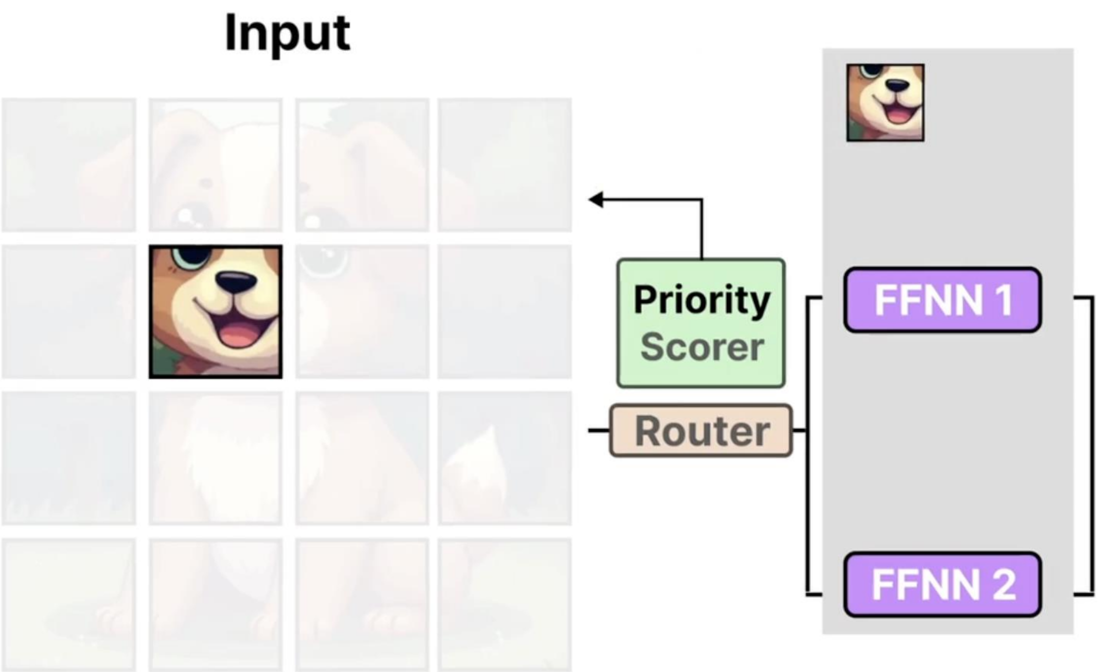
... and information in **unprocessed** patches is lost.



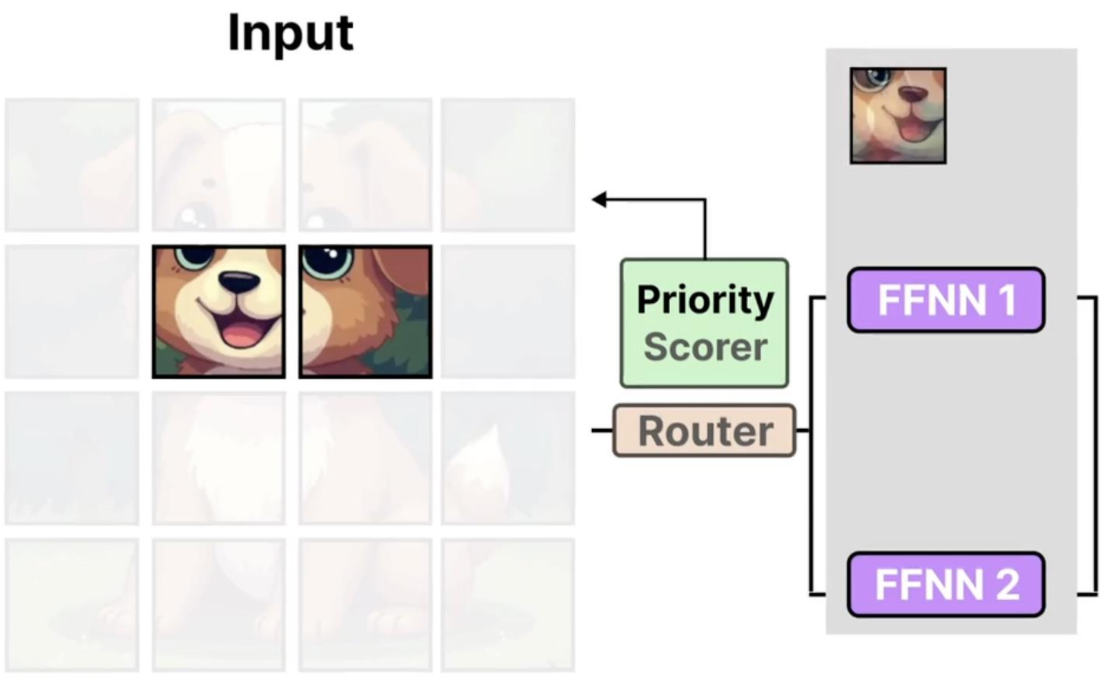


Soft-Moe solves this problem, and aims to go from a **discrete** patch assignment...

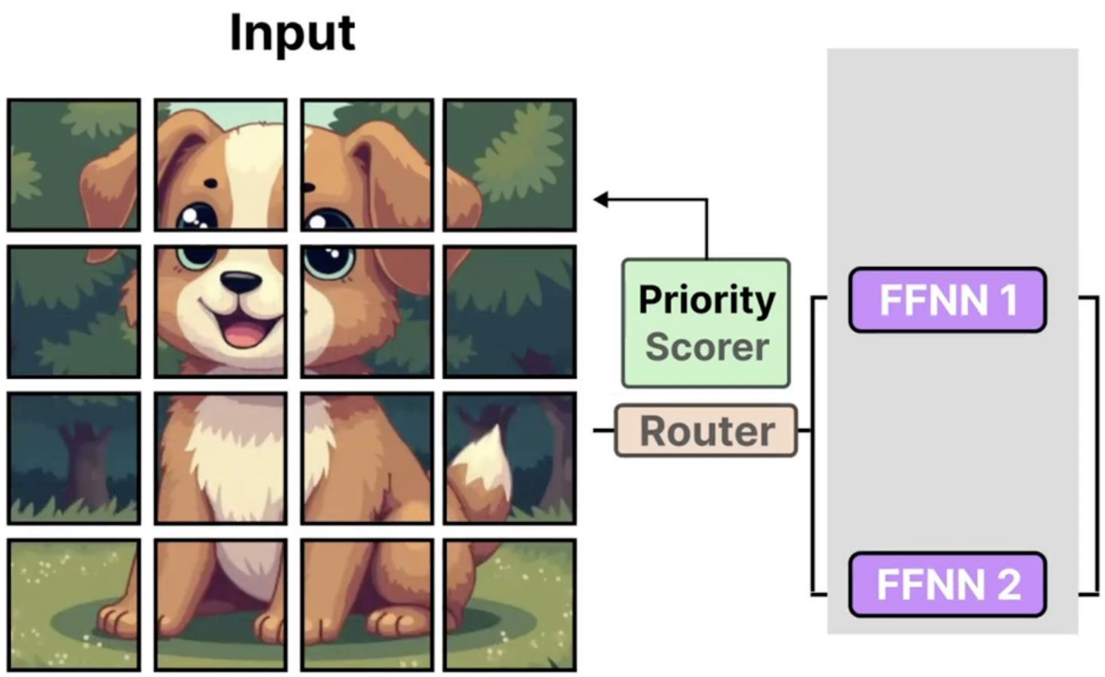




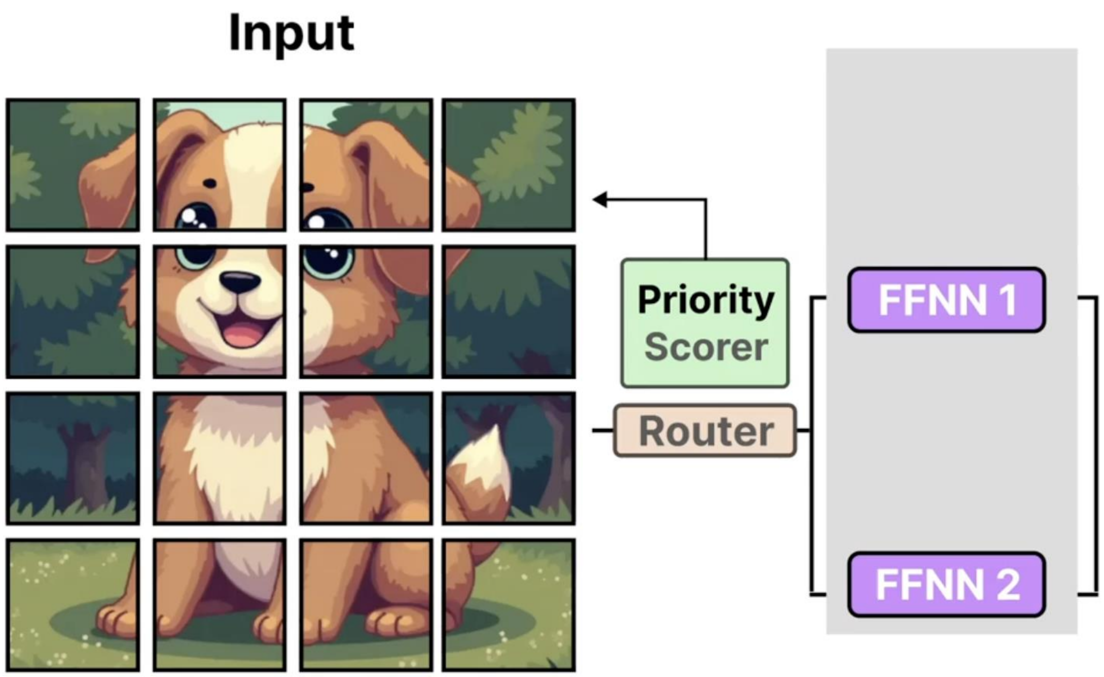
Soft-Moe solves this problem, and aims to go from a **discrete** patch assignment...



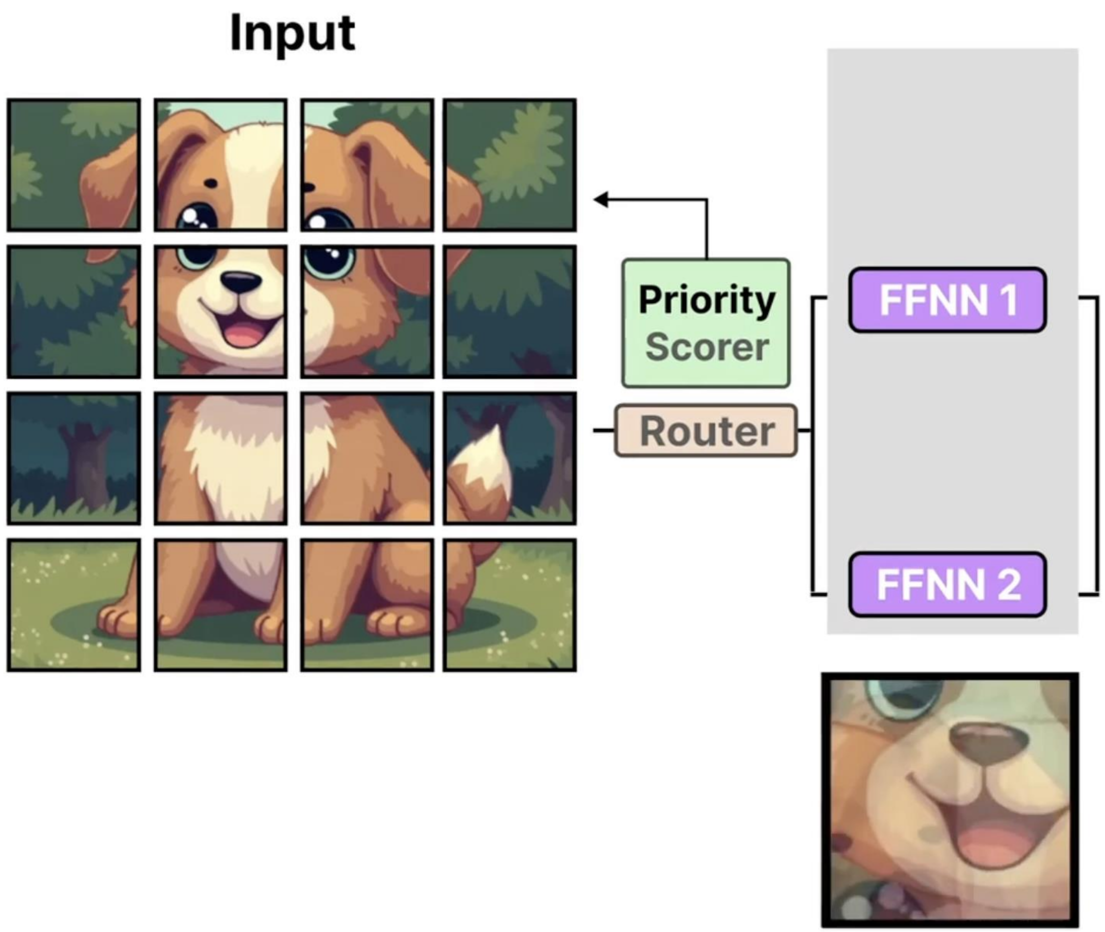
... to a **soft** patch assignment by mixing patches.



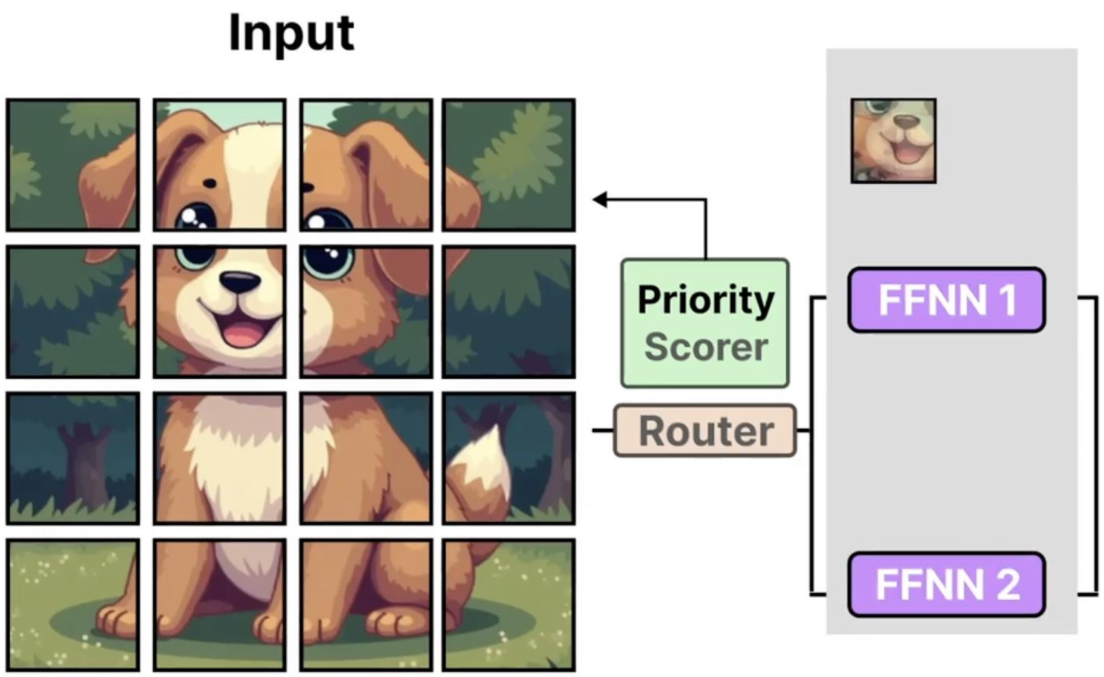
In practice, **all patches** are used instead of only a subset to create this soft patch.



We **weigh** each patch using the priority scorer...

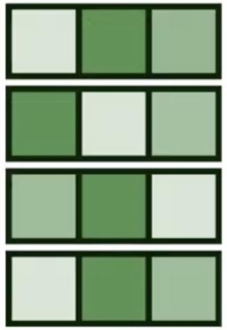


... and then take a **linear combination** of these weighted patches...



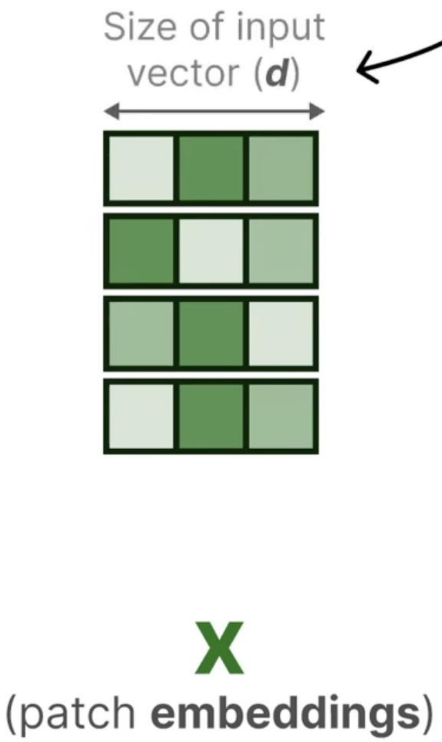
... before we send the **soft patch** to the chosen expert.

To create these soft patches, **Soft-MoE** takes the input **X** (patch embeddings one for each patch).

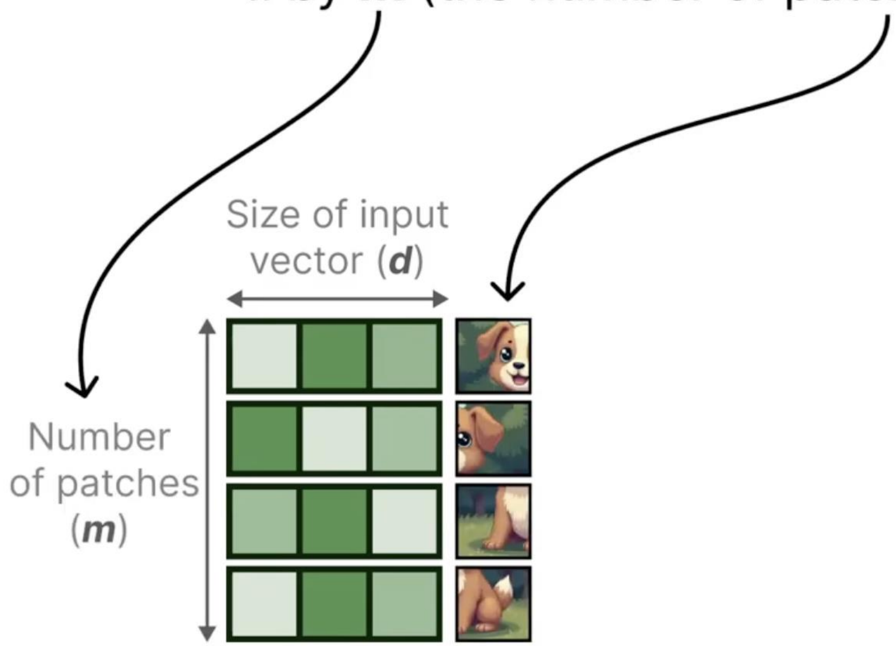


X
(patch **embeddings**)

With dimensions **d** (size of input vector)...

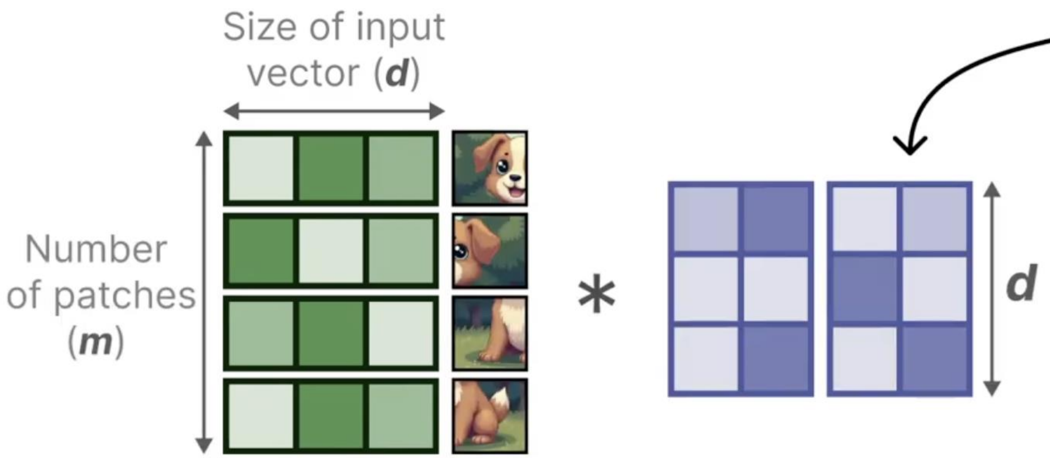


.. by **m** (the number of patches).



X
(patch **embeddings**)

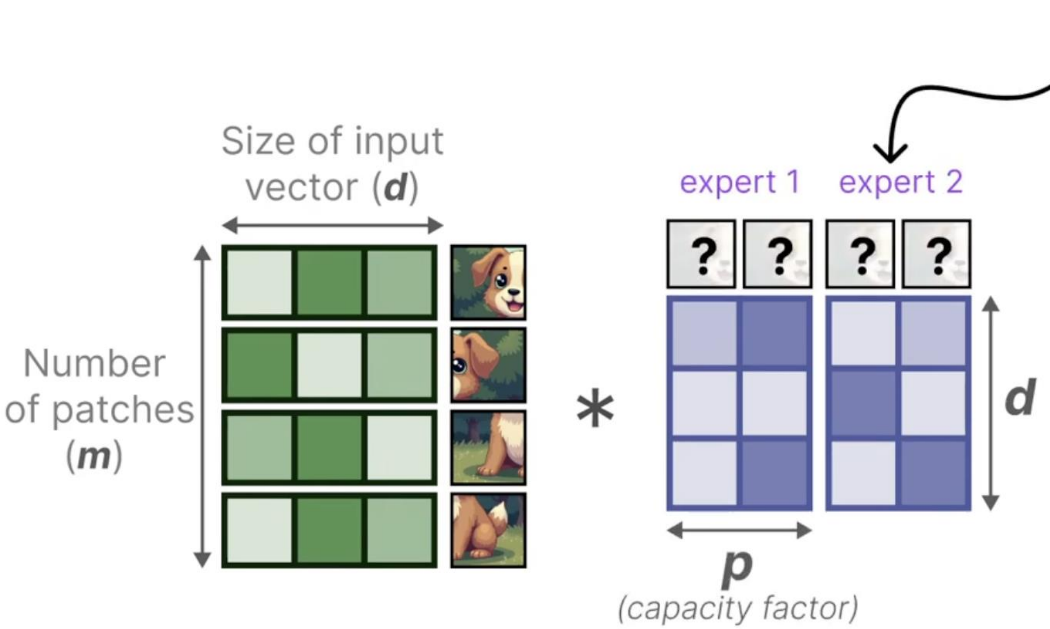
X is then multiplied by a learnable matrix **Φ** with dimensions **d** by...



X
(patch **embeddings**)

Φ
(learnable matrix)

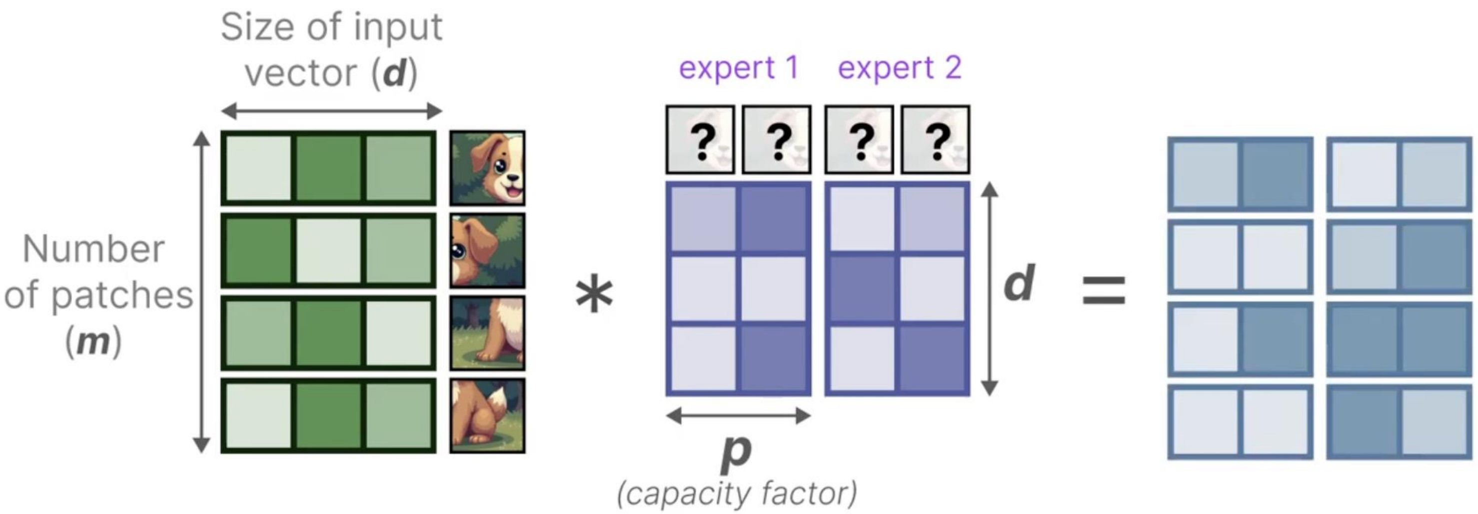
... the capacity factor for each **expert**.



X
(patch **embeddings**)

Φ
(learnable matrix)

This gives us the routing matrix **R**.

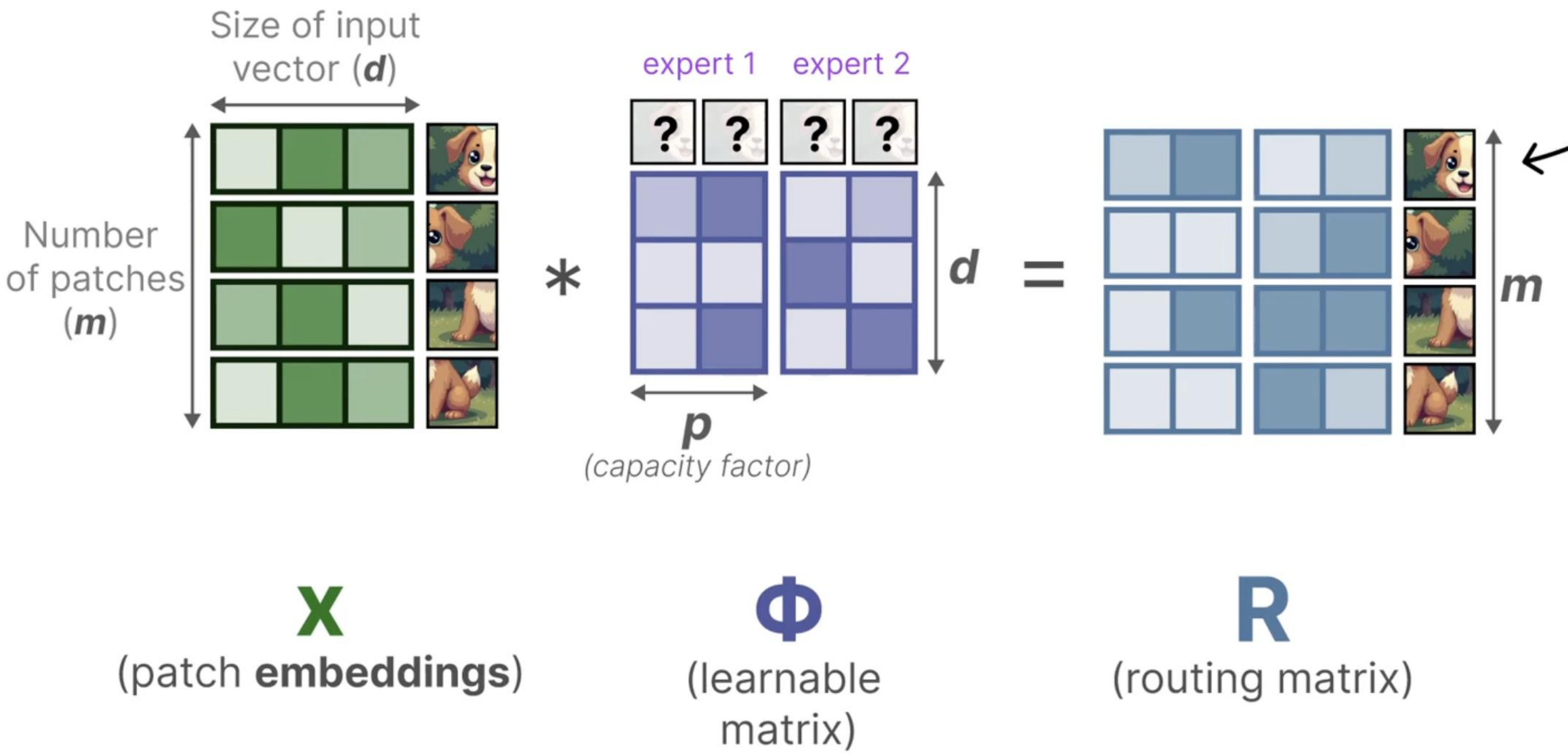


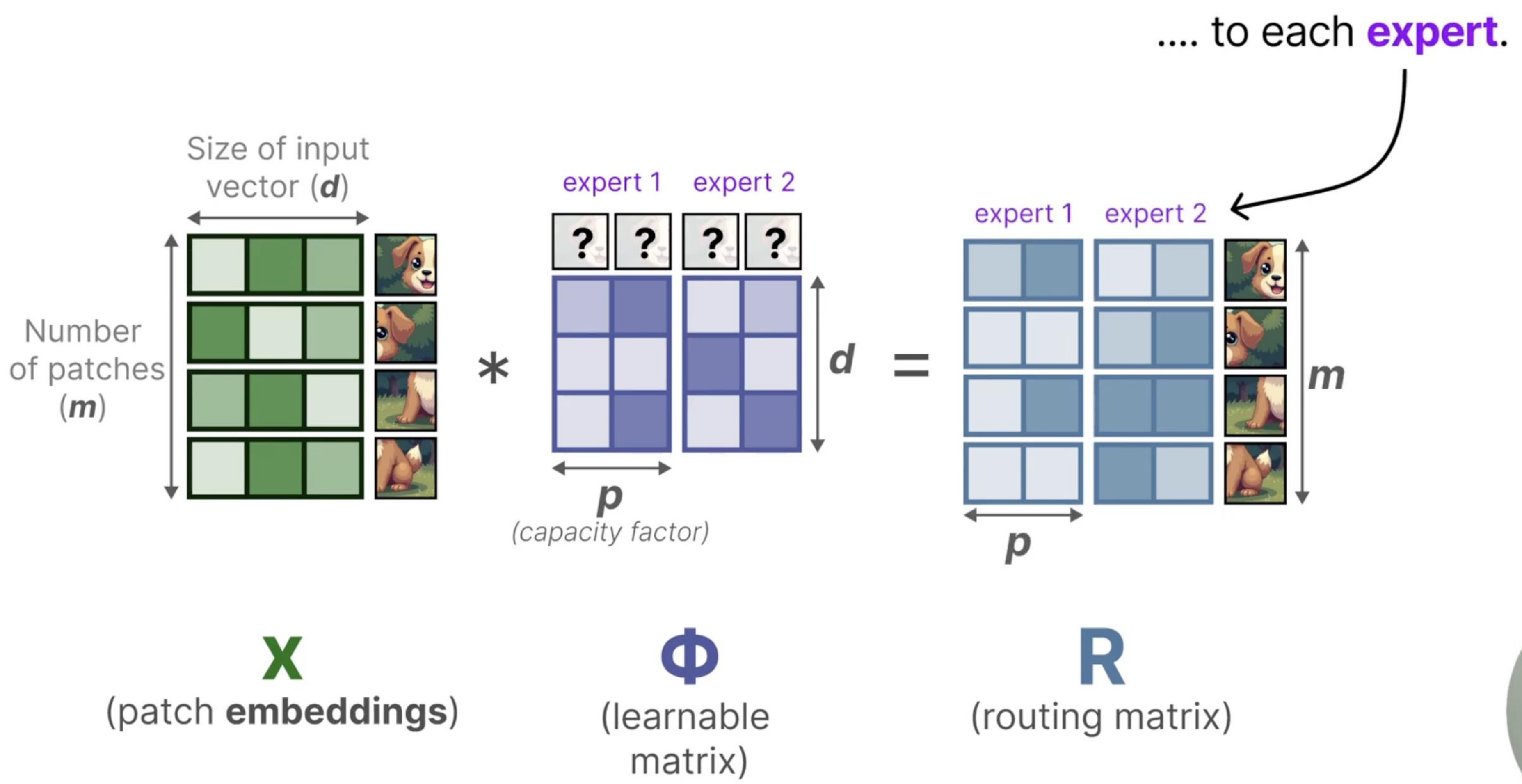
X (patch **embeddings**)

Φ (learnable matrix)

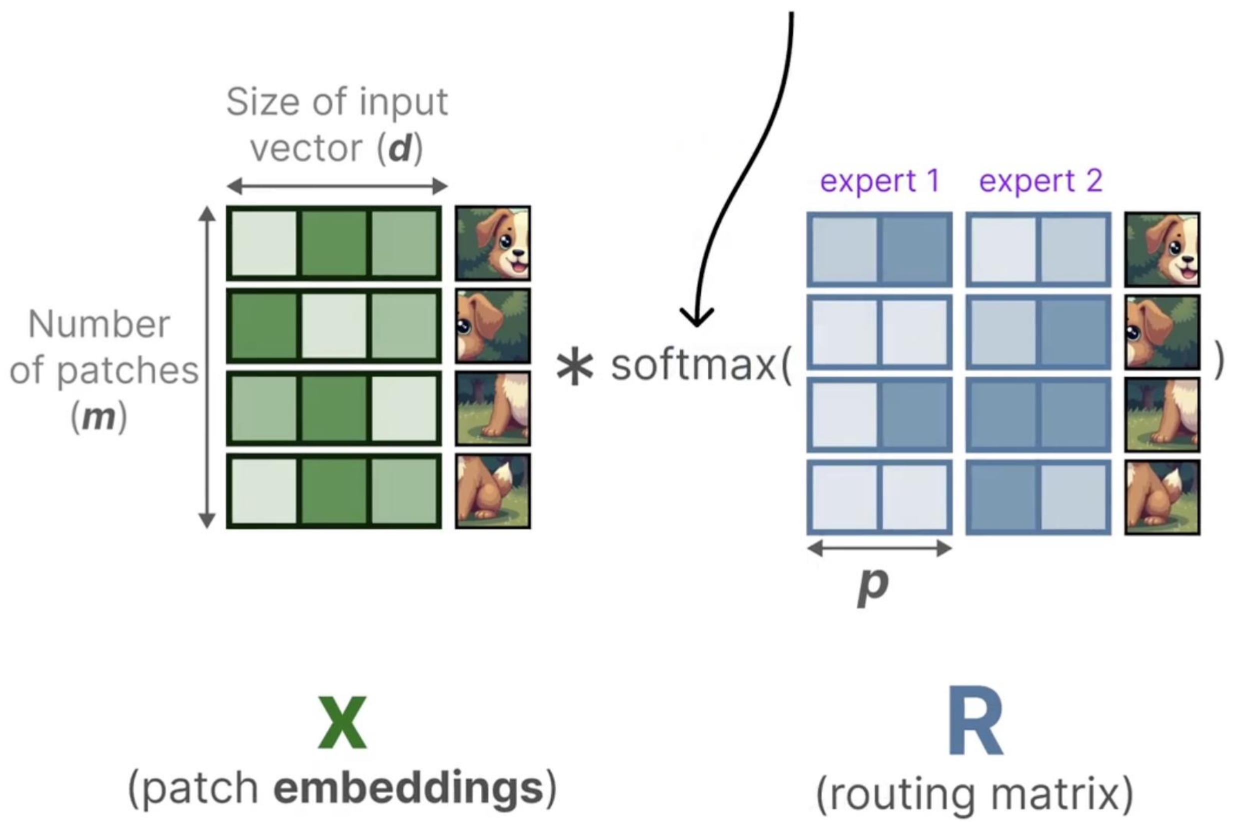
R (routing matrix)

... which tells us how related a certain **patch** is...

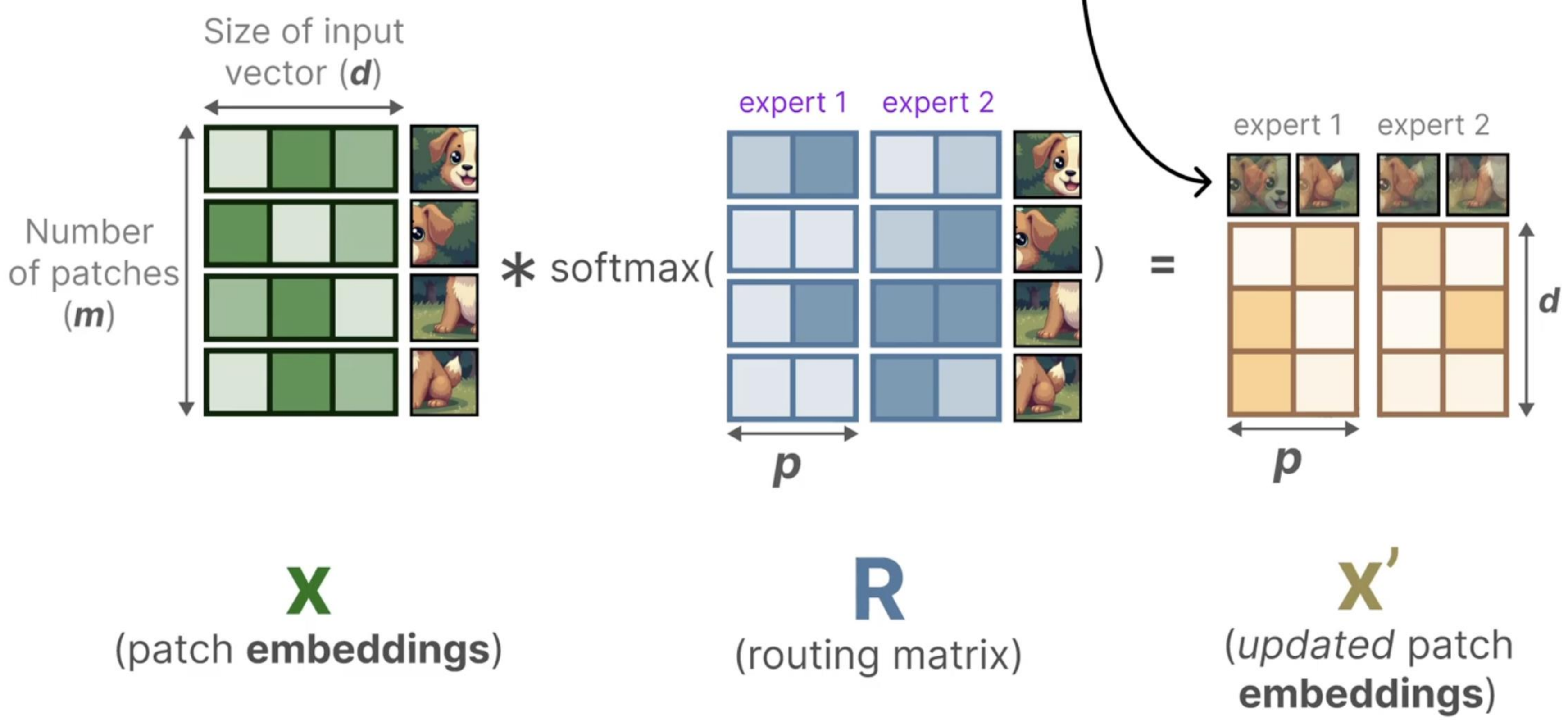



.... to each expert.
Size of input vector (d)
Number of patches (m)
expert 1
expert 2
? ? ? ?
d
p
(capacity factor)
expert 1
expert 2
m
p
X
(patch embeddings)
Φ
(learnable matrix)
R
(routing matrix)

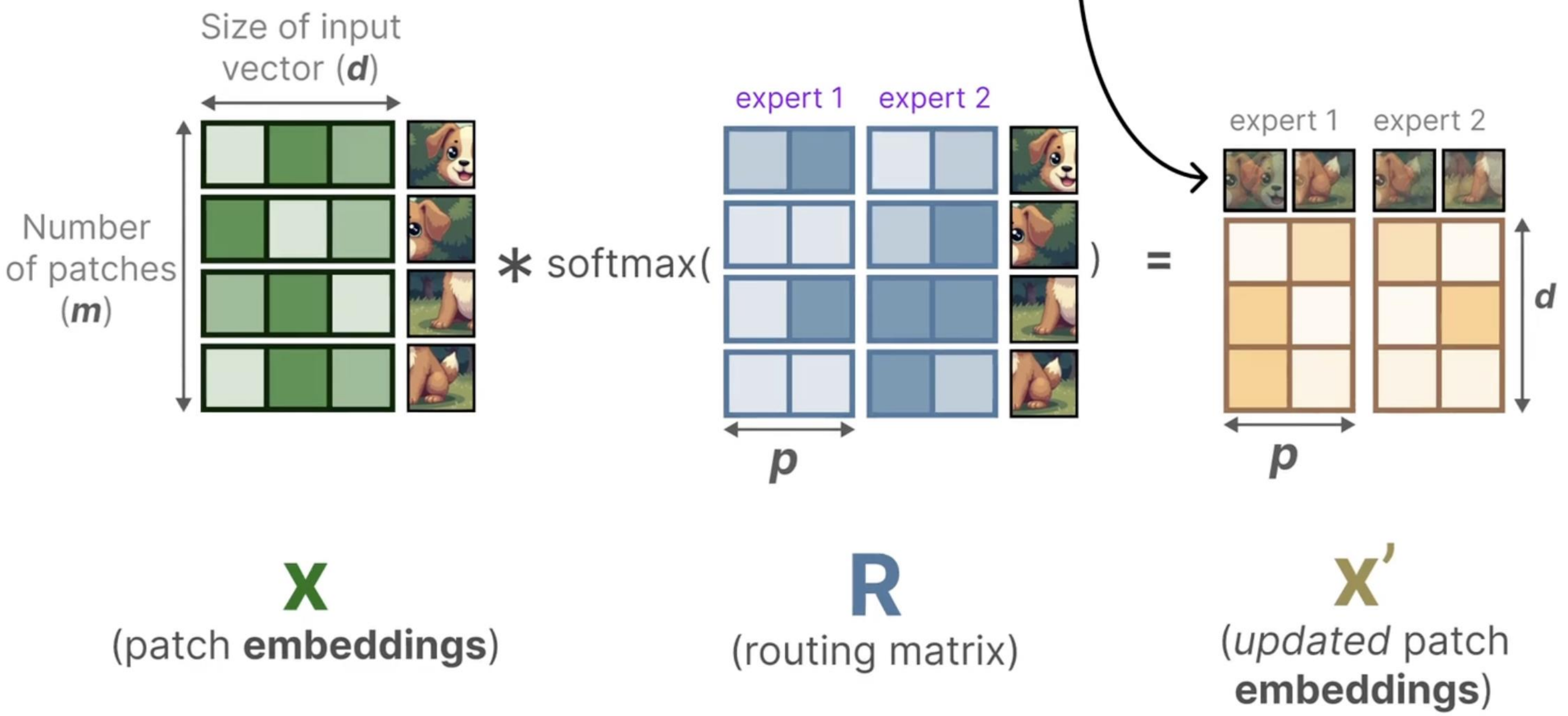
By then taking the **softmax** of the **router matrix** (on the columns) ....



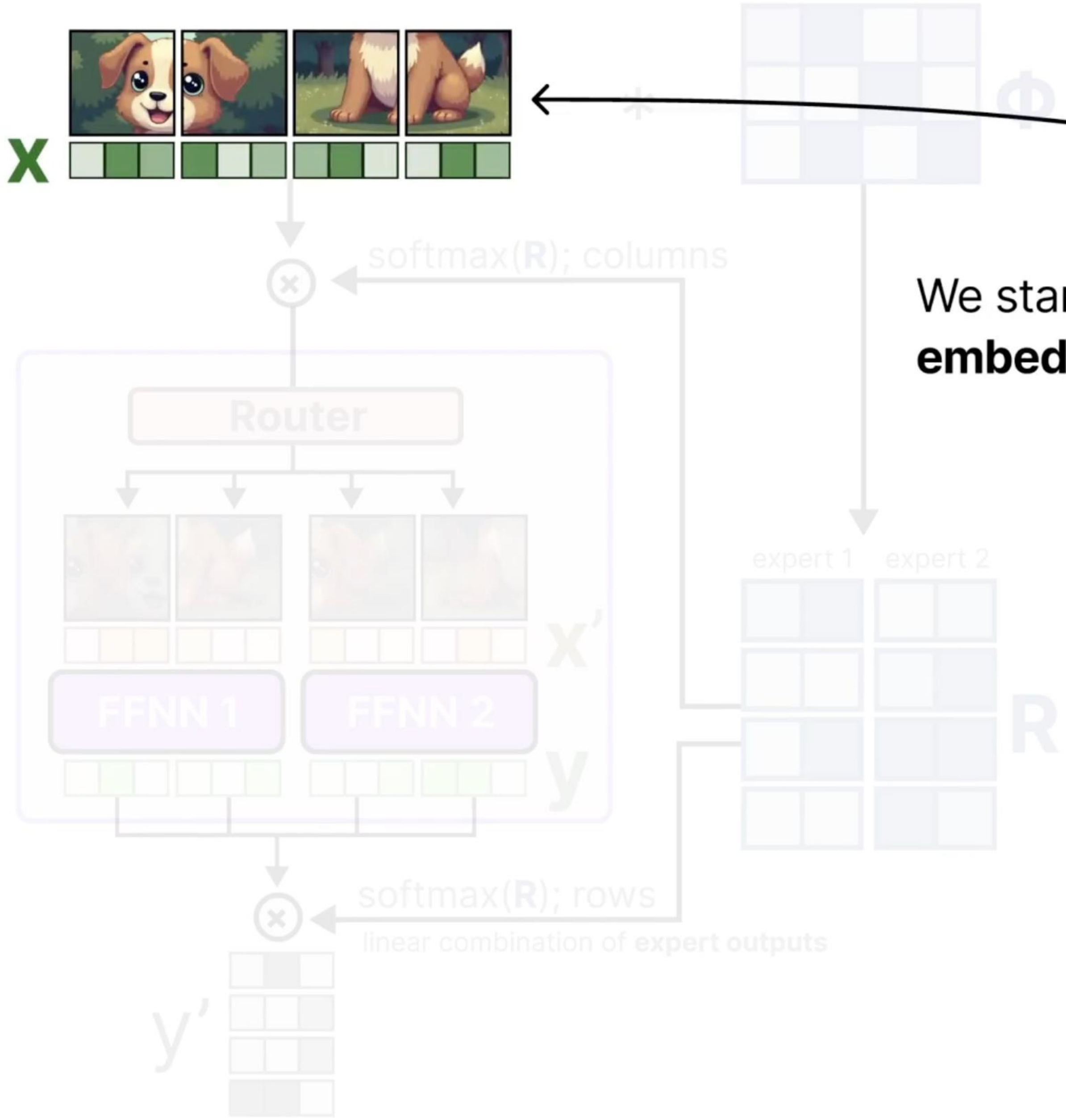
...we **update** the embeddings of each patch.



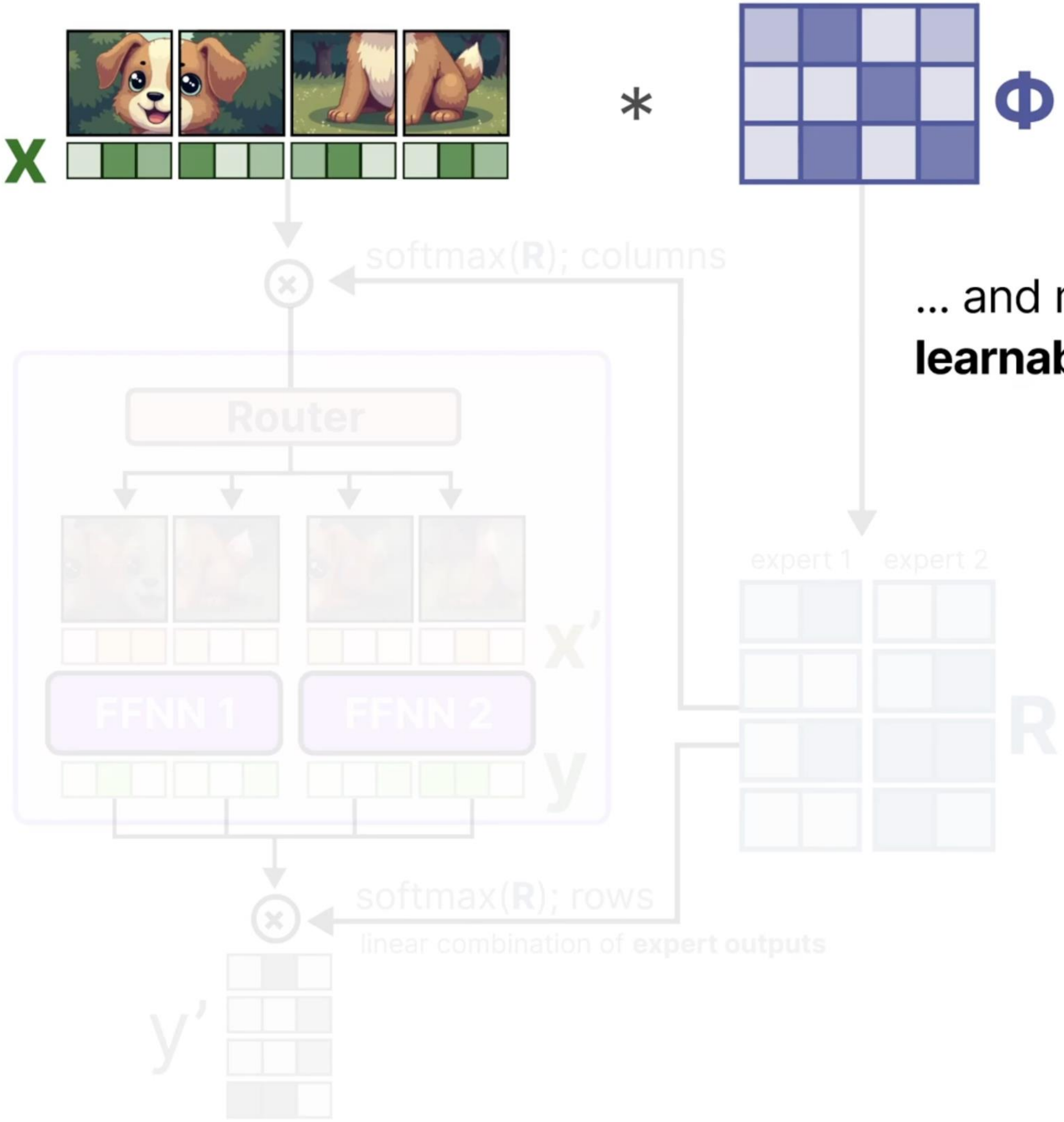
As a result, they are a **linear combination** of the input patches as we saw before.



Let's go through this **step-by-step**.

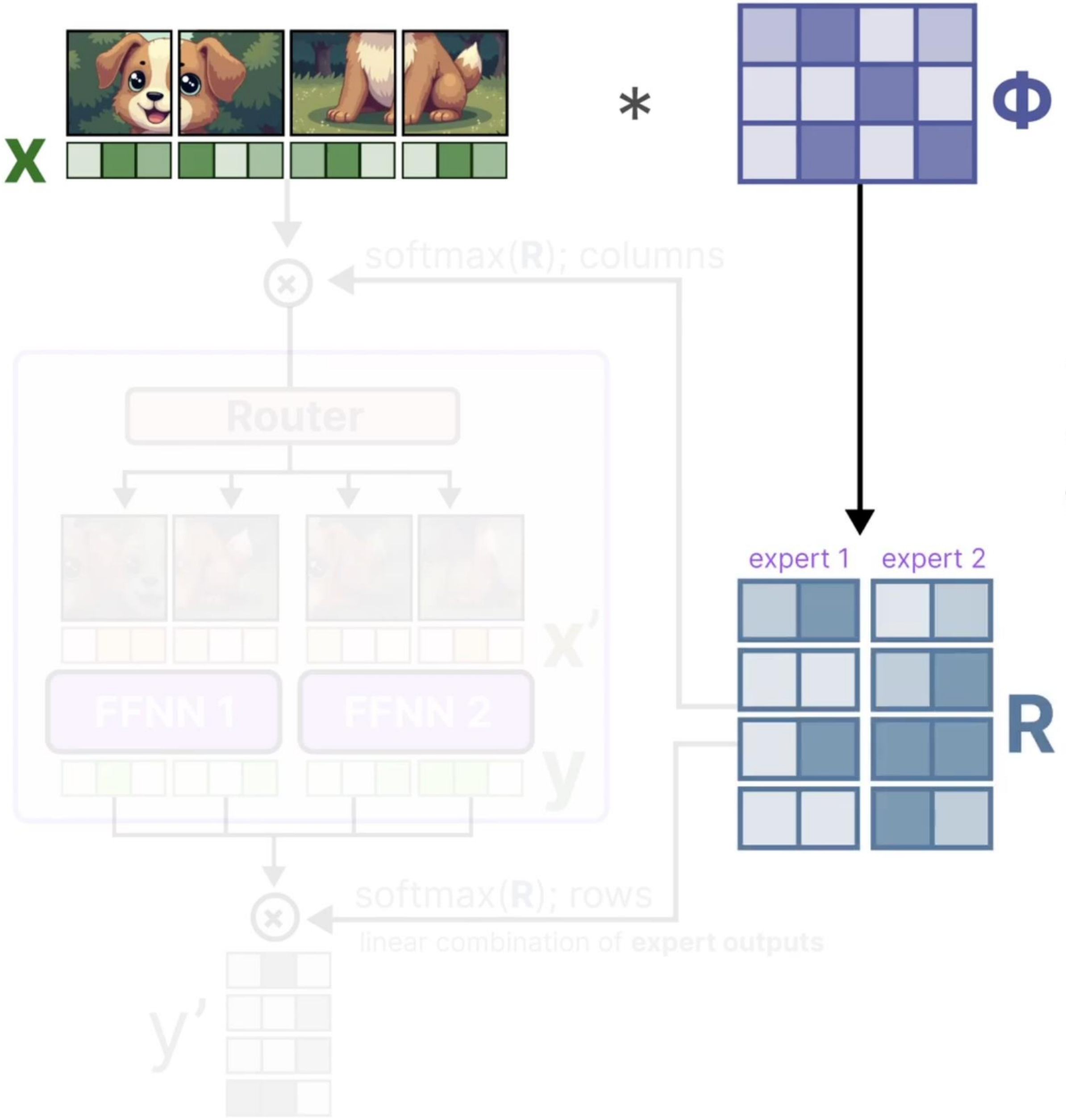


We start with the input **X**, **embeddings** for each patch.

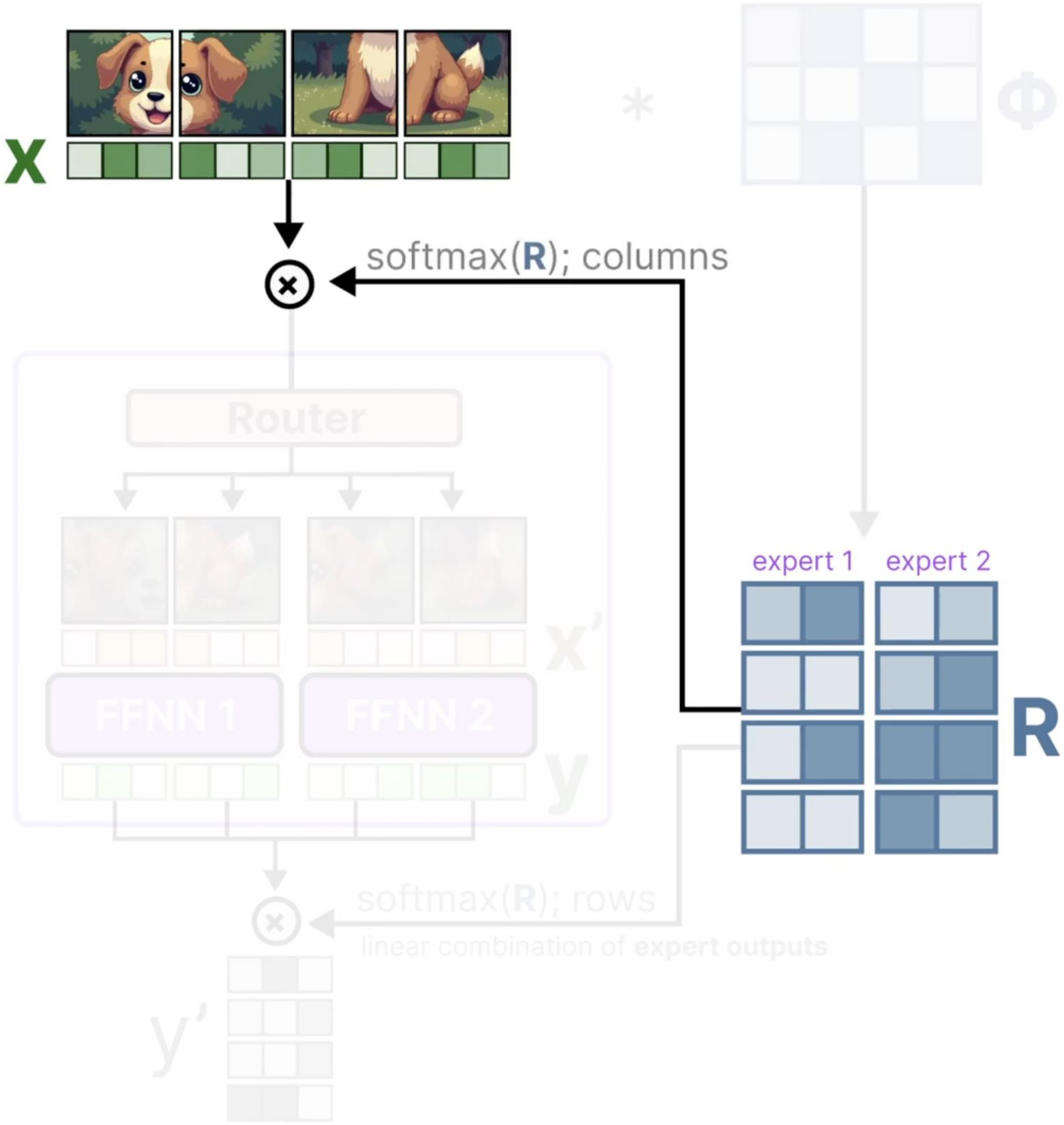


... and multiply that with a **learnable matrix Φ**.

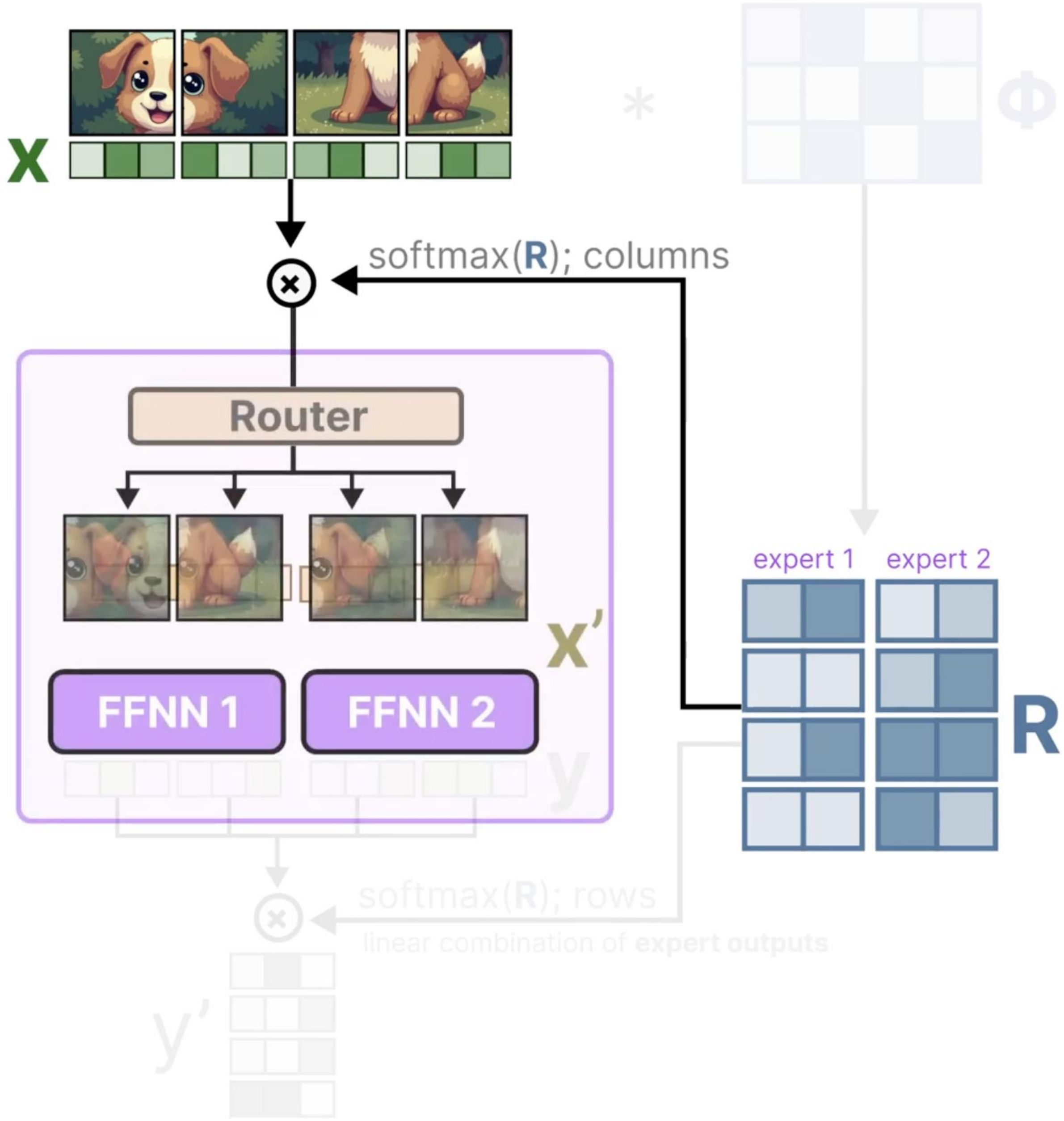
From Sparse to Soft MoE



This gives us the **router matrix** R, which tells us how related a certain token is to a given expert.

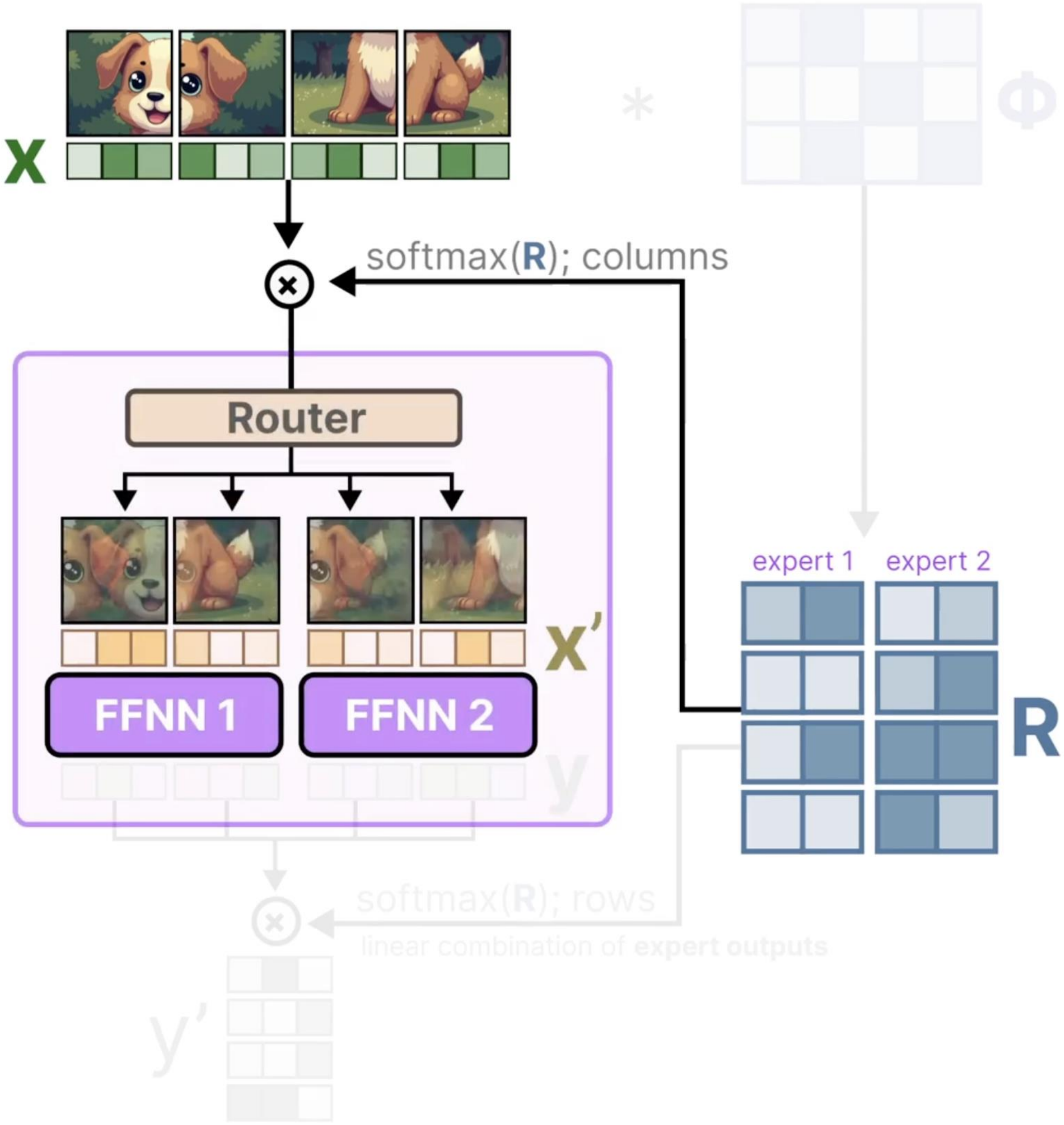


Taking the softmax of R and multiplying that with the input gives us a **linear combination of patch inputs**.

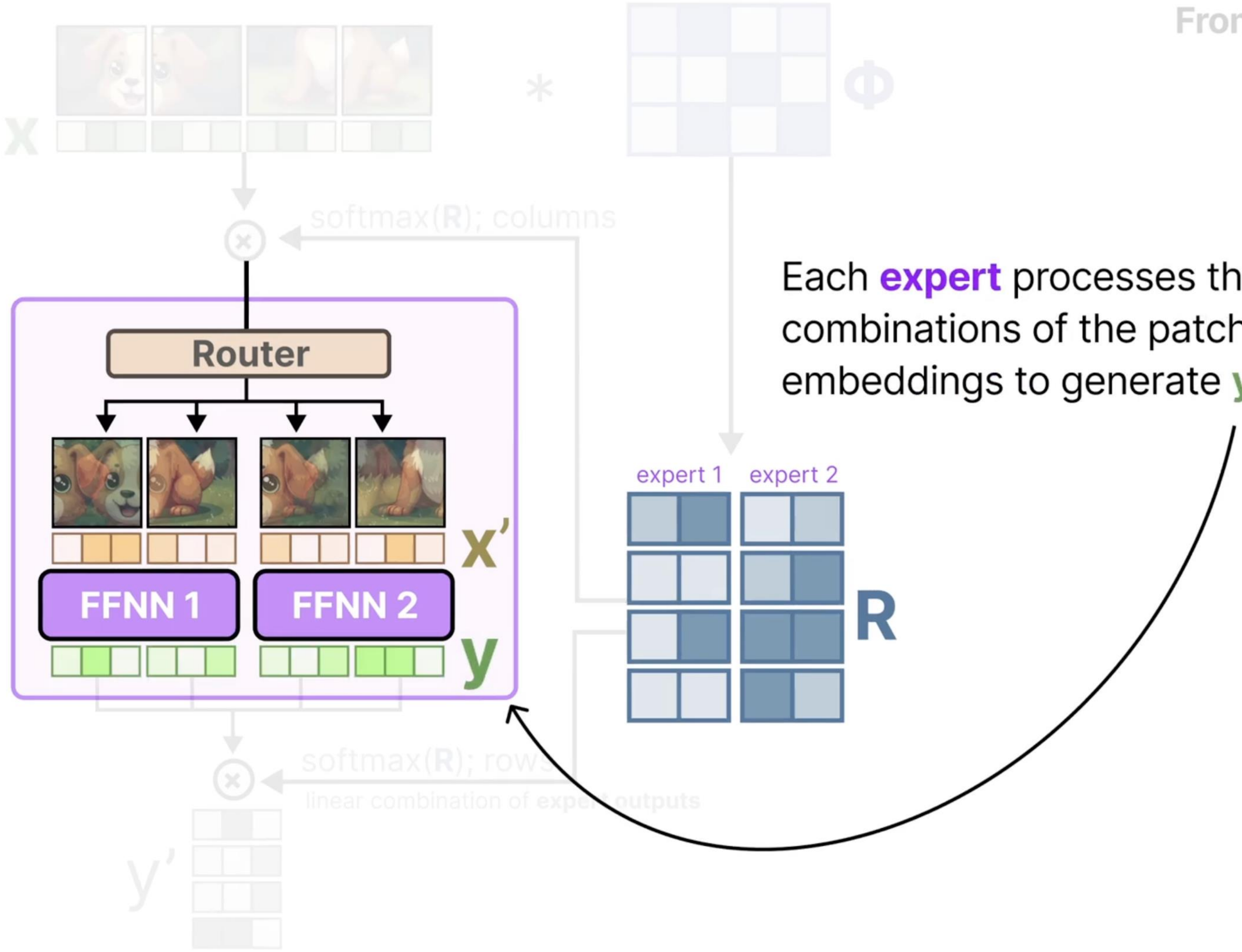


These are routed to the best suited **expert** for a particular patch input.

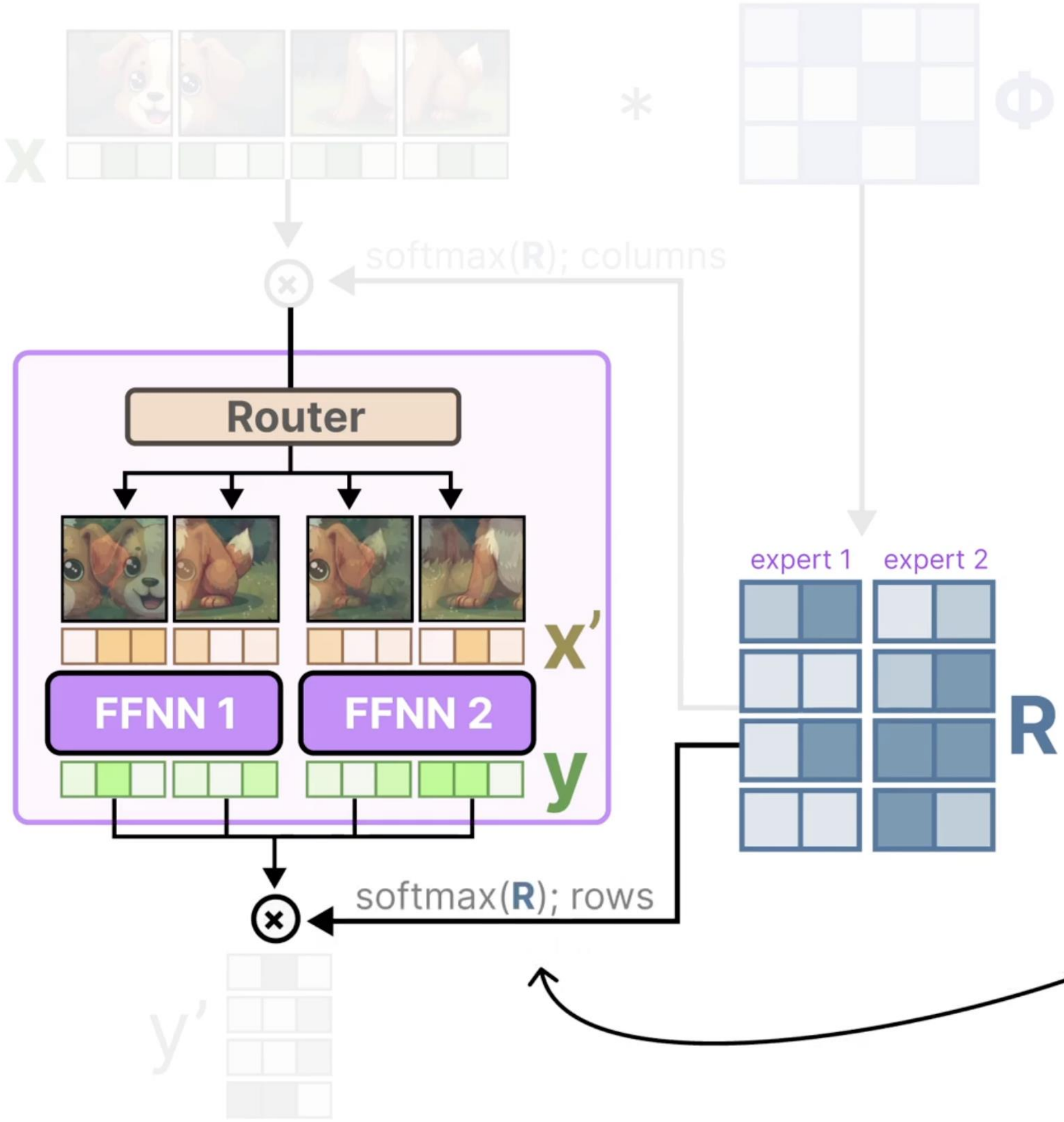
From Sparse to Soft MoE



These are routed to the best suited **expert** for a particular patch input.

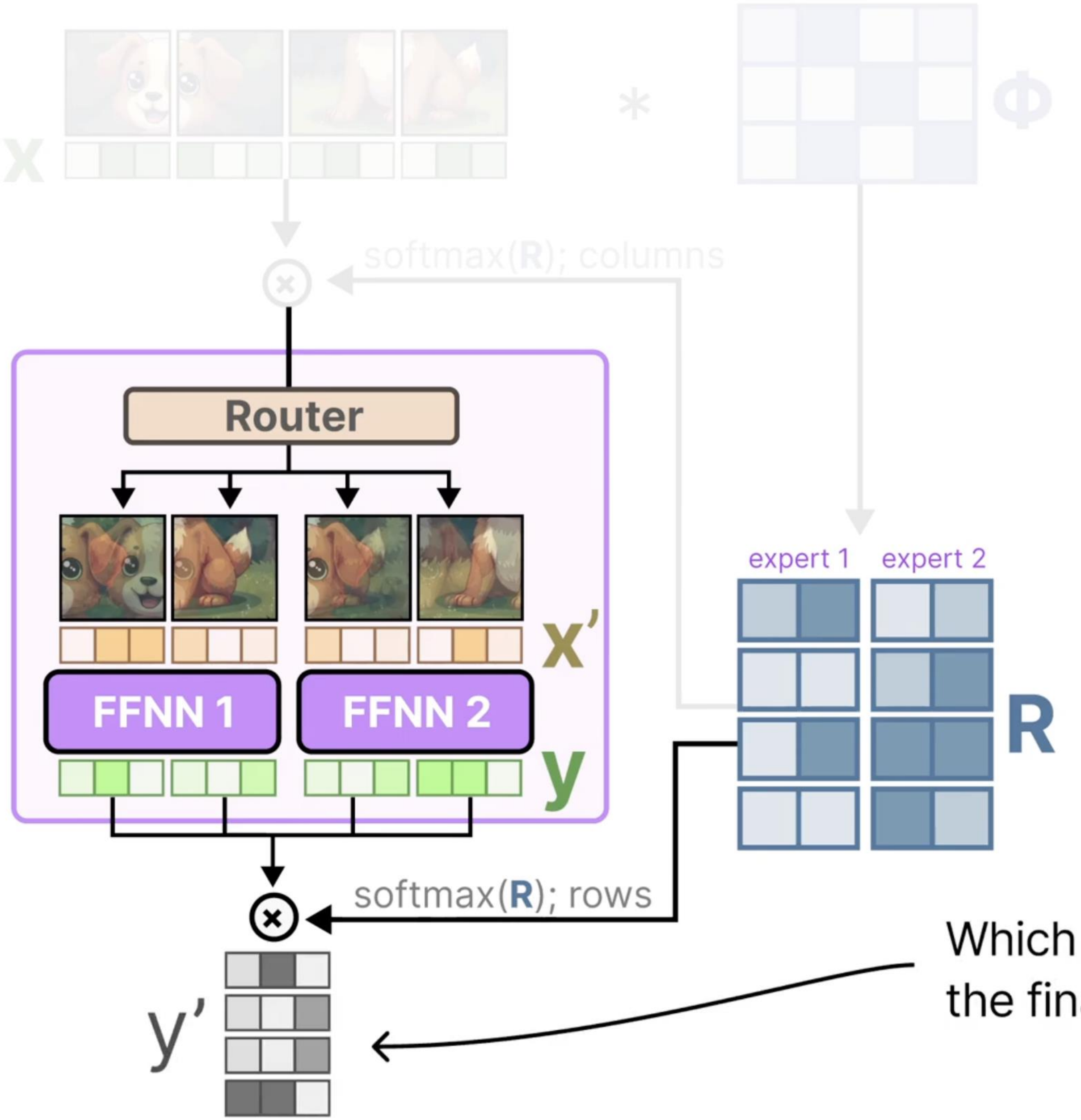


Each **expert** processes these linear combinations of the patch embeddings to generate **y**.

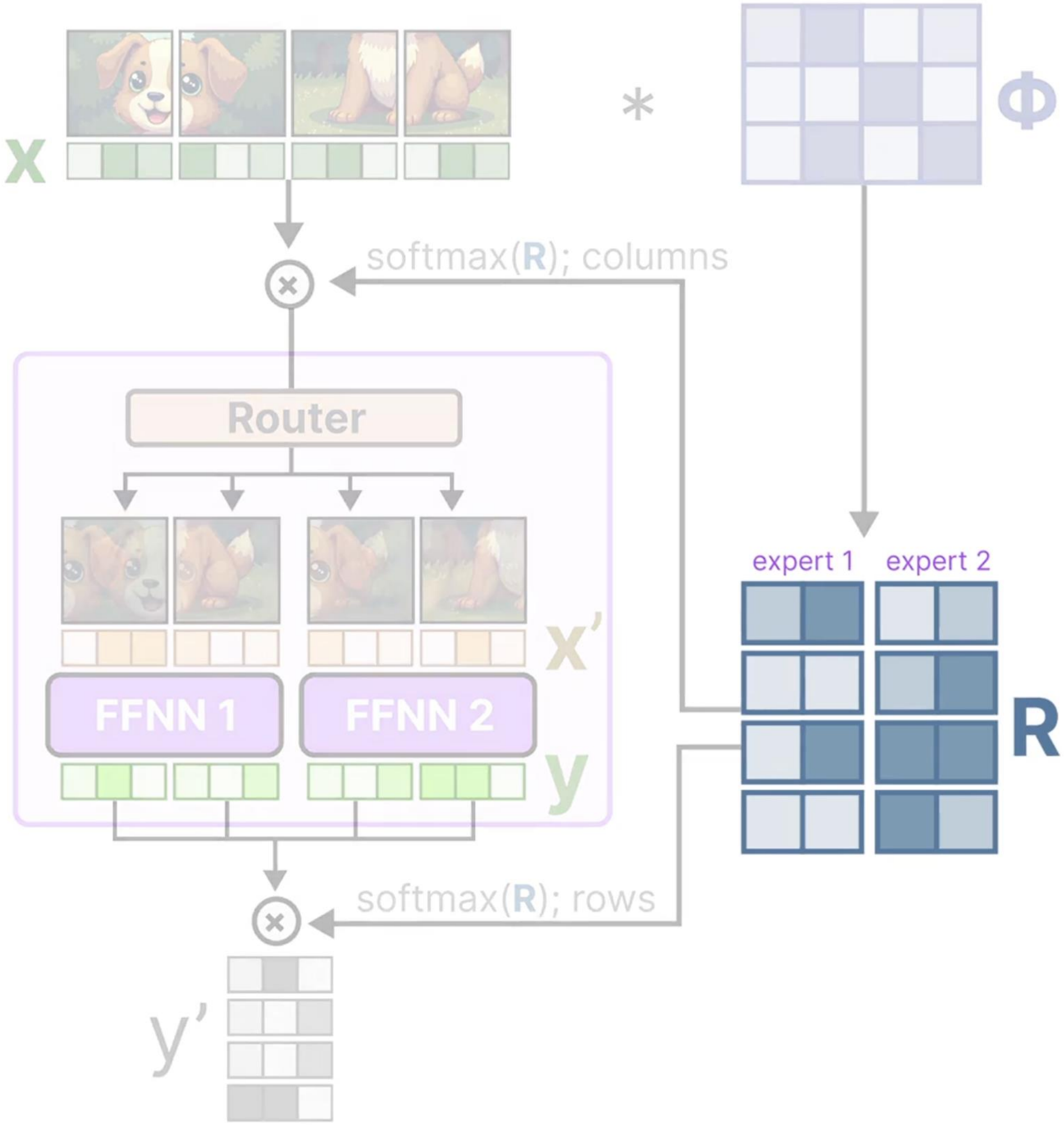


The output **y** is multiplied with the softmax of **R** but this time row-wise, creating a **linear combination of expert outputs** instead.

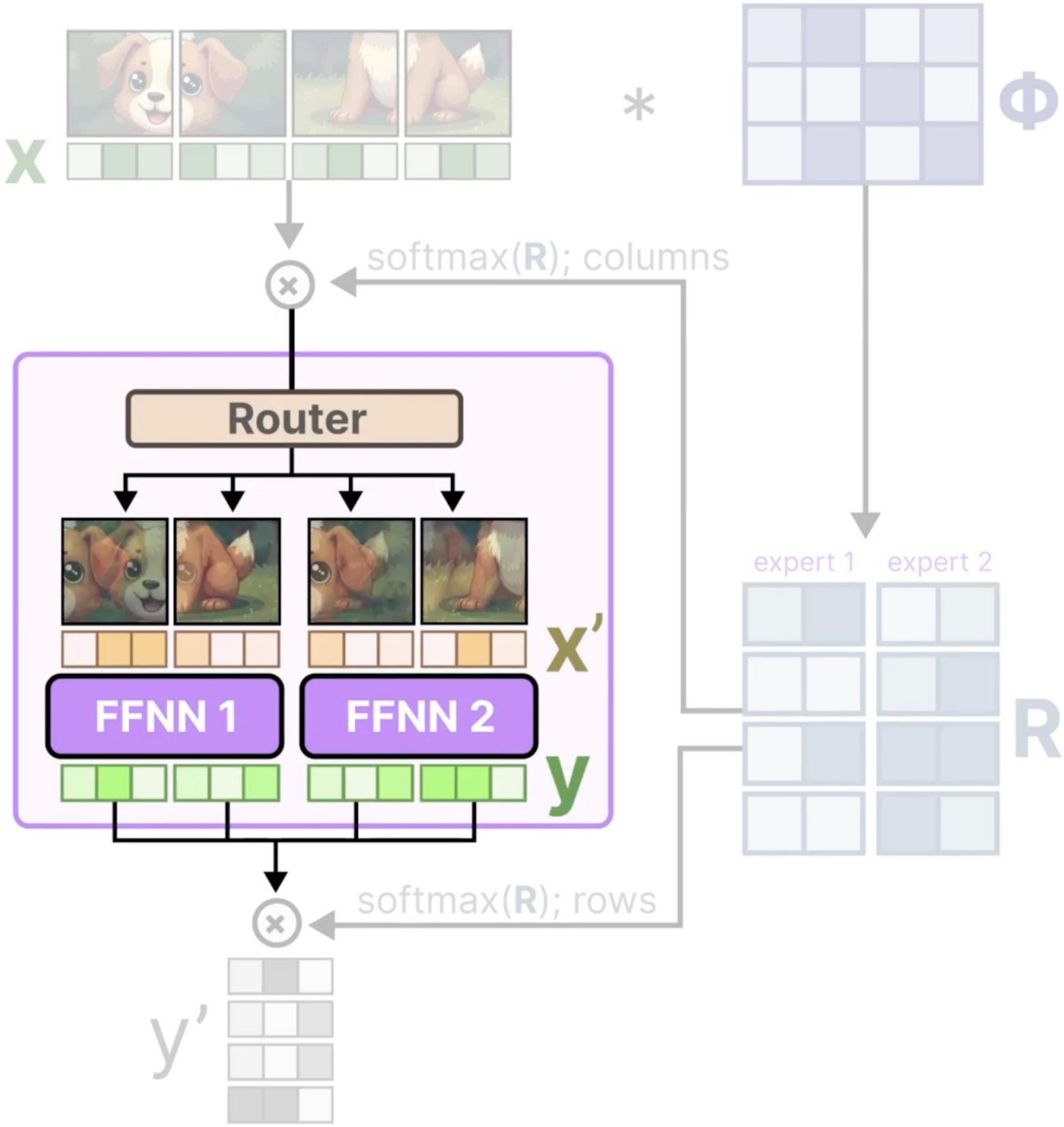
From Sparse to Soft MoE



Which gives us
the final output **y'**.

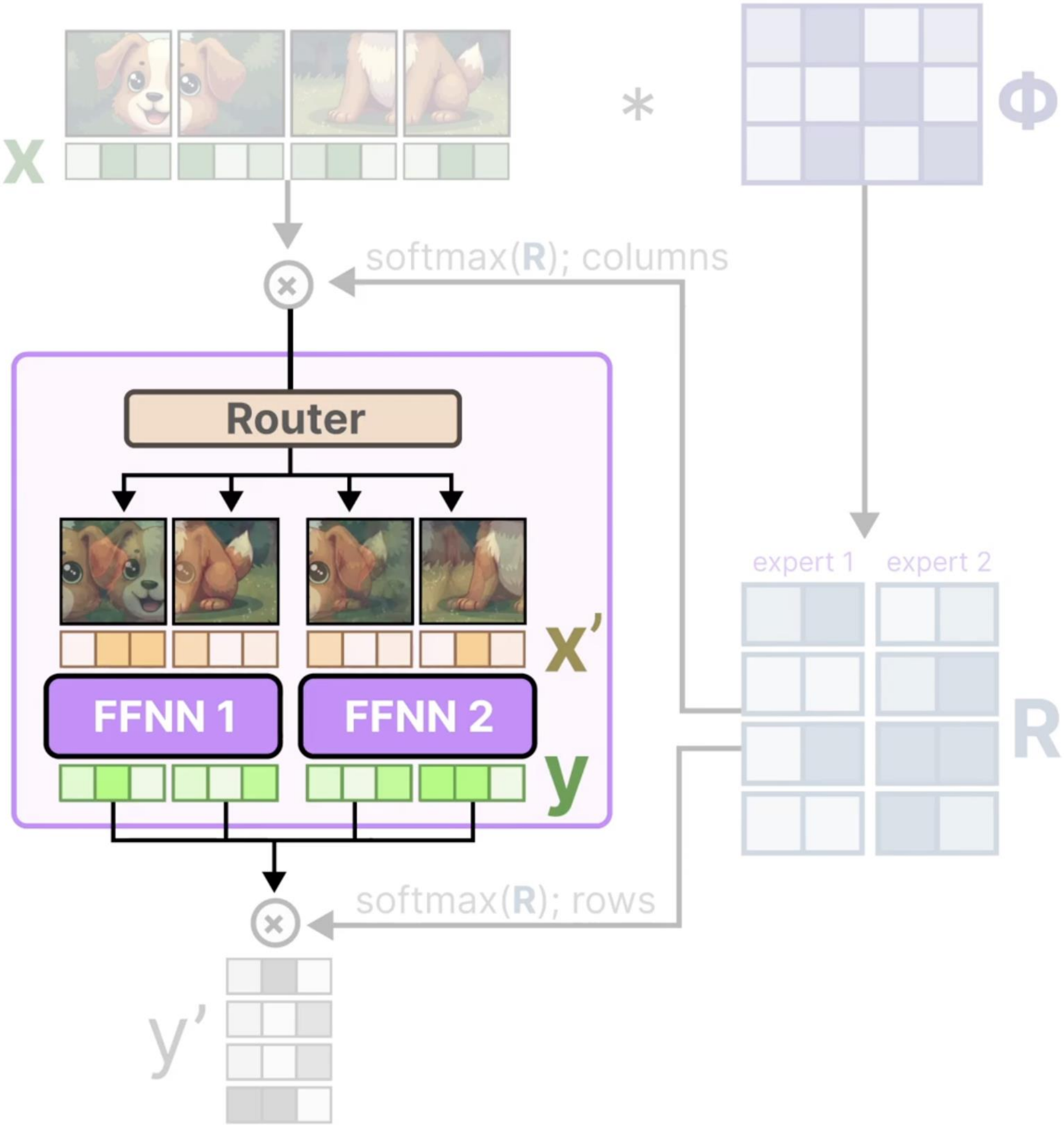


This architecture, together with the **router matrix**, affects the input on a **patch** level and the output on an **expert** level.



Since Transformers has found its way into the domain of vision, techniques like **MoE** are surprisingly transferable across domains.

From Sparse to Soft MoE



So, keep in mind that whatever you learn about LLMs might also have theoretical and practical implications to **multimodal models**!

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://www.youtube.com/watch?v=sOPDGQjFcuM&t=1s
- https://arxiv.org/abs/2308.00951
- https://arxiv.org/abs/2106.05974

- PPT 开源在：https://github.com/chenzomi12/AIInfra