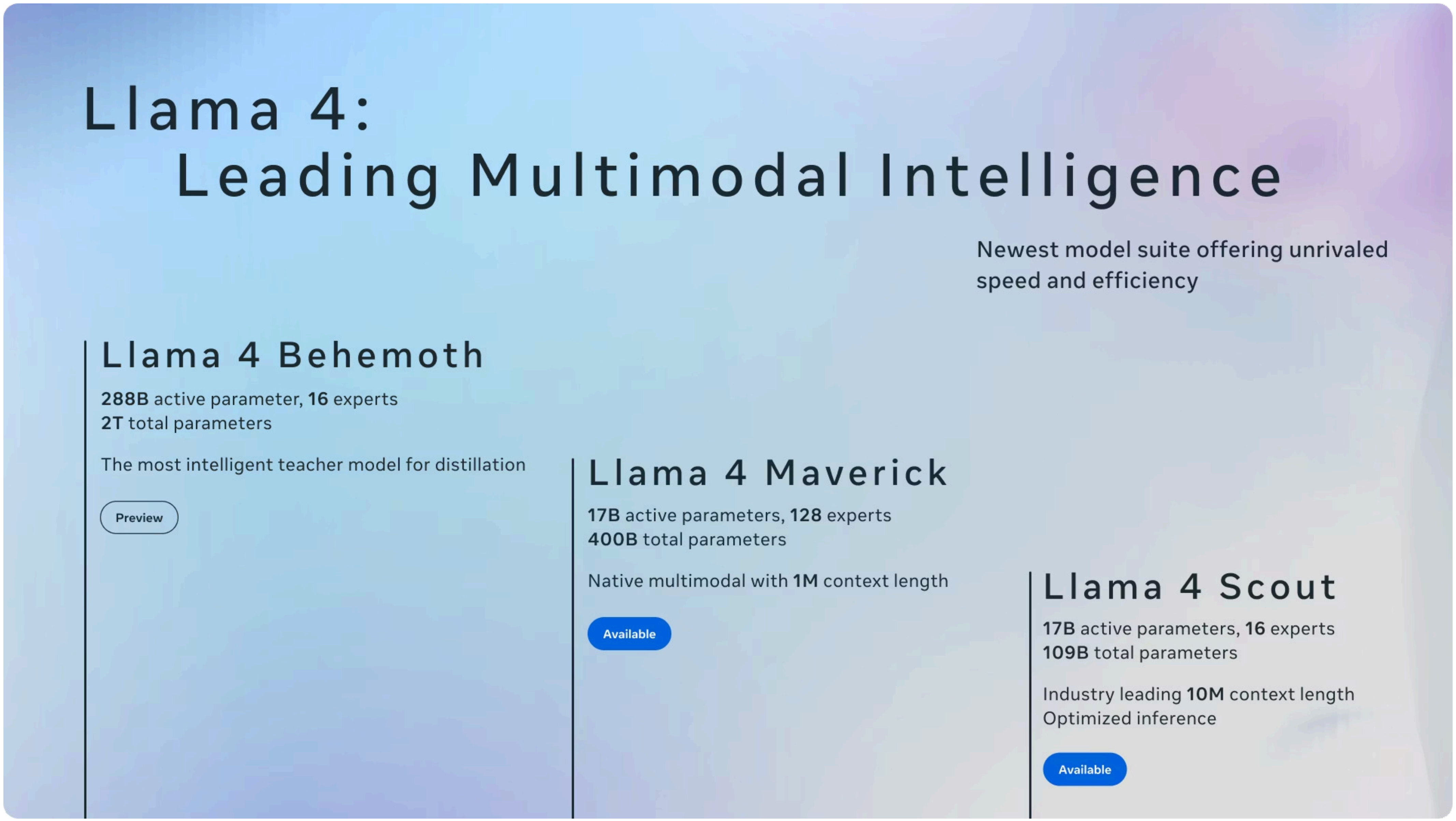
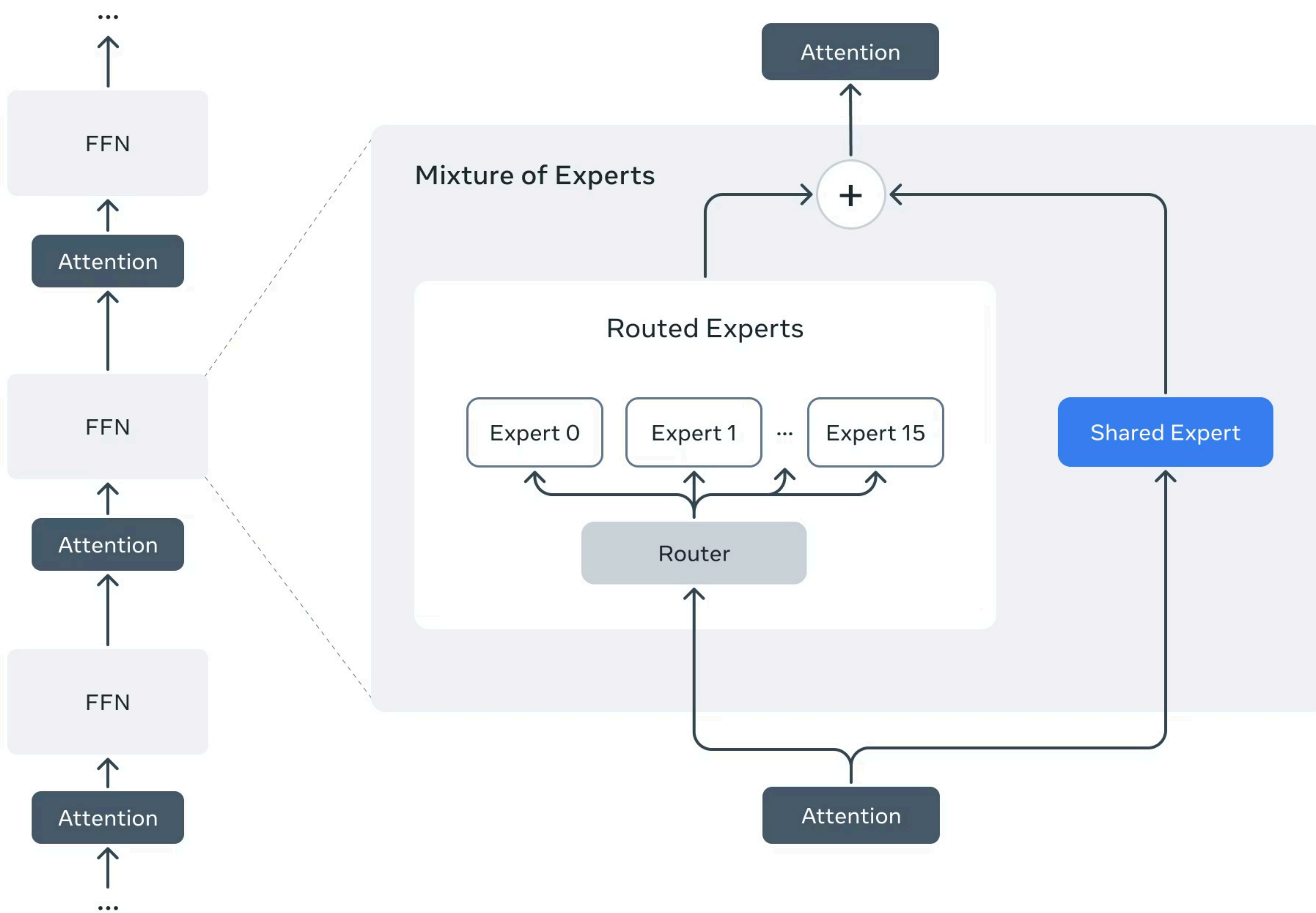
# Unveiling LLaMA 4: Meta's Next-Gen Large Language Model

This presentation explores the architectural advancements and key features of Meta's LLaMA 4, focusing on its innovative Mixture-of-Experts (MoE) design and multimodal capabilities.



# LLaMA 4: An Overview

LLaMA 4 is Meta's latest large language model, designed for enhanced performance and efficiency across a broader range of applications. It builds upon the successes of LLaMA 2 and LLaMA 3, introducing significant architectural innovations.



# Evolutionary Leaps: LLaMA 4 vs. Predecessors

### Expanded Vocabulary

LLaMA 4 features a significantly larger vocabulary size (202048), improving its ability to handle diverse linguistic nuances and rare tokens more effectively.

### Vast Training Data and Larger context Length

Trained on a considerably larger and more diverse dataset than LLaMA 2/3, and siginificantally large context Window(i.e  10M for LLama 4 scout and 1M for Maverick)

3

### Mixture Of Experts

LLaMA 4 uses a Mixture of Experts (MoE) architecture where only a few experts are activated per token, making it computationally efficient while expanding model capacity. Each expert specializes in different types of data, improving task-specific performance and generalization.

### MultiModal Approcah

LLaMA 4 follows a **multimodal approach**, allowing it to process and understand both **text and images** within a unified architecture. It integrates vision encoders alongside language models

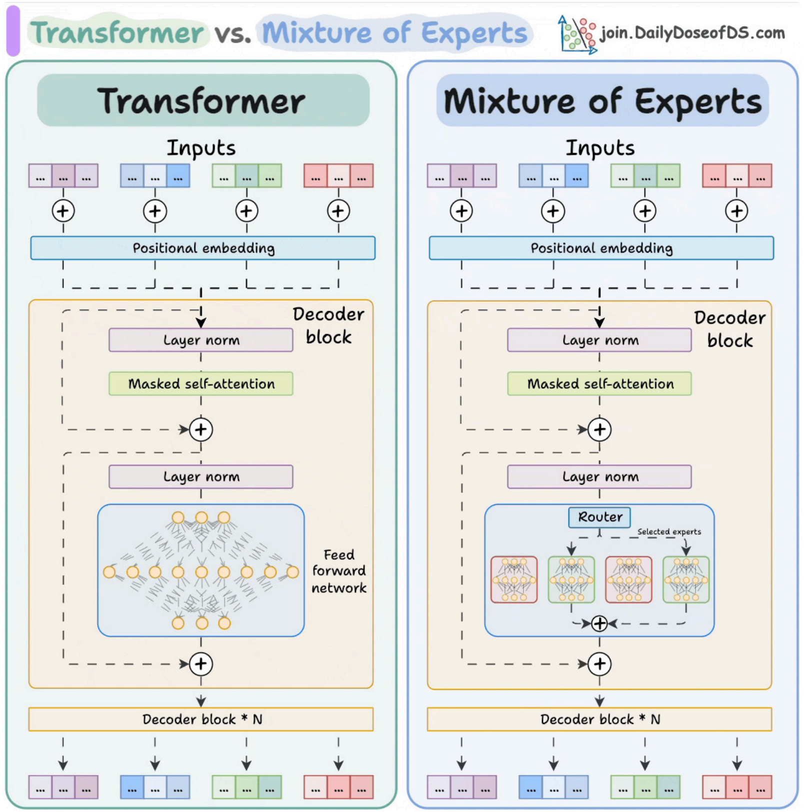
# Mixture of Experts (MoE): The Core Innovation

## What is MoE?

MoE is a neural network architecture where different "expert" sub-networks specialize in processing different types of inputs or tasks. Instead of one large model, it uses several smaller, specialized ones.

## Intuitive Explanation

Imagine a team of specialists. When a complex problem arises, a "router" directs it to the most relevant expert(s) instead of having one generalist try to solve everything. This allows for highly efficient and specialized processing.



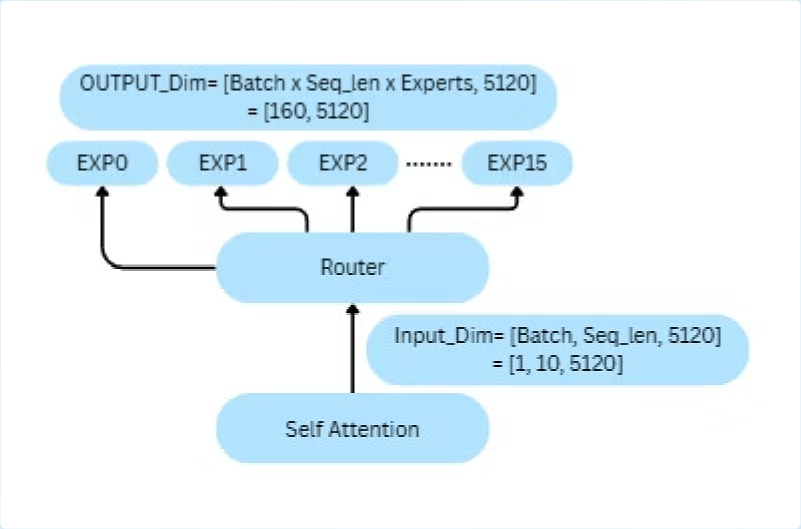

Made with GAMMA

## MOE - Routing Layer

- **Input:** Each token comes from the attention block as a vector (e.g., 5120-dim).
- **Goal:** Decide **which experts** should process each token (not all experts process all tokens).
- **How?**

The router is a small **linear layer** that transforms each token into a **score for each expert**.

  - If there are 16 experts, each token gets **16 scores** (one per expert).
    - ***Tokens = Batch_size*Seq_len*** ***(i.e 1×10)***
    - ***L(Tokens, Experts)=(X(Tokens, Model_Dim))*(W(Model_Dim,Experts))***
  - This means: "How suitable is expert is for this token?"
- **Top-k selection:**
  - For each token, the router picks the top `k` experts (usually `k=1` or `2`) based on the highest scores.
- **Score masking:**
  - All other experts are ignored by masking their scores (setting them to `-inf`).
  - Then a **sigmoid** function is applied to convert the top-k scores into soft weights in range (0, 1).
    - ***Score(Tokens, Experts) = Sigmoid(masked(L(Tokens, Experts)))***
- **Routing Weights:**
  - These weights represent **how much each selected expert should influence this token**.
- **Broadcasting:**
  - Each token is sent to *all* experts for uniform processing, but only the **selected experts receive non-zero inputs** (others are zeroed out using the weights).



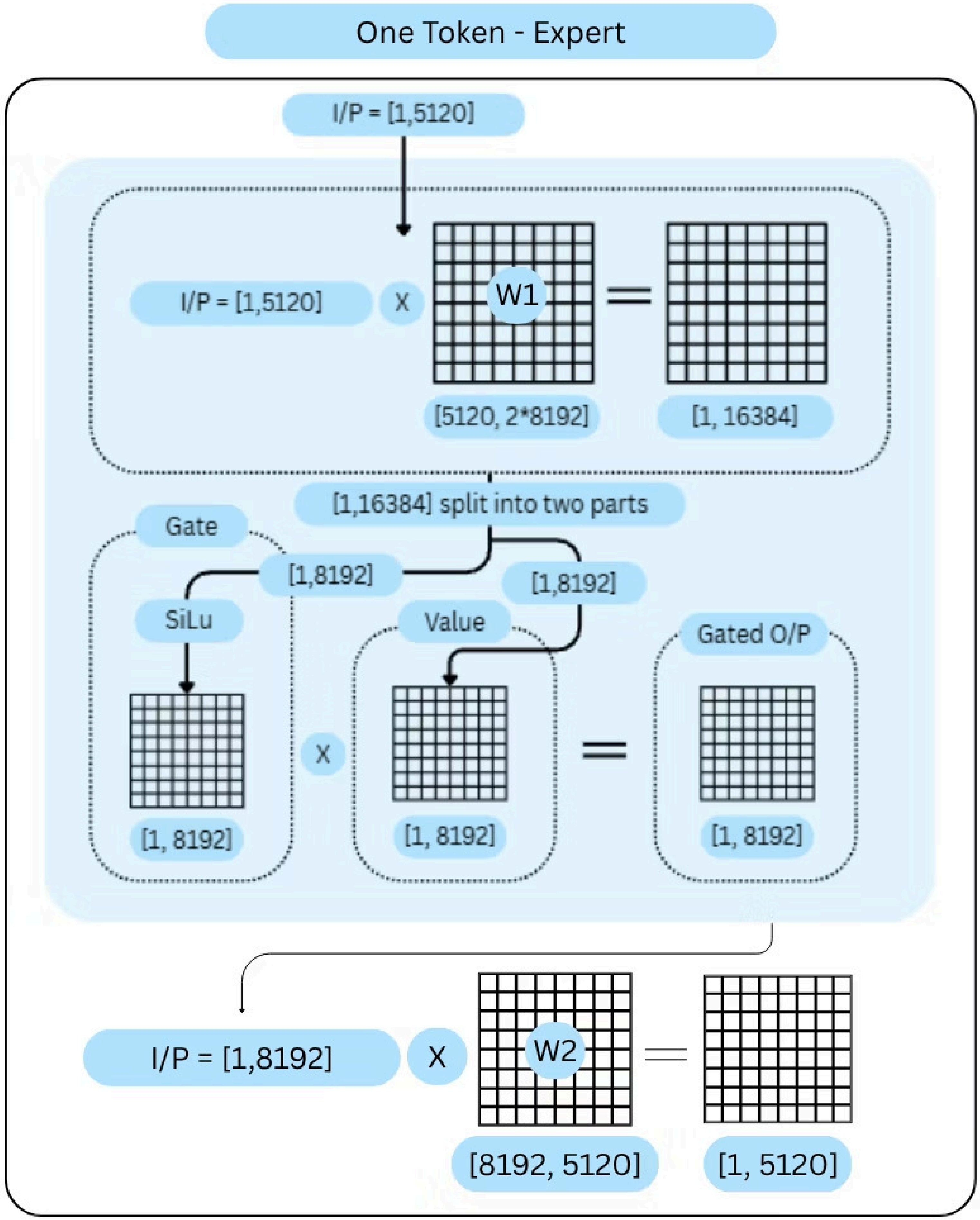
# Expert In MOE

Each expert is a **Gated Feedforward Network** (a modified MLP), and it works in the following way:

- **Input:** A token that has been routed (selected) for this expert — a single 5120-dimensional vector.
- **Goal:** Transform this token's representation using a **feedforward neural network (FFN)**, specialized per expert.

1. **Upscale the token:**
   - The expert **linearly projects** the token from `hidden_size` (e.g. 5120) up to **twice the intermediate size**, i.e. `2 × 8192 = 16384`(Using weight matrix w1).
   - This output is then **split into two equal parts**:
     - One part becomes the **gate**
     - The other part becomes the **value**
2. **Apply nonlinearity:**
   - The **gate part** goes through a nonlinearity (typically **SiLU**).
   - This controls what parts of the value to keep or suppress.
   - It's like saying: "How strongly should each part of the up-projected vector be expressed?"
3. **Element-wise gating:**
   - Multiply the SiLU-activated gate with the value part → produces the **gated output**.
   - This allows the model to control **information flow dynamically**.
4. **Downscale back:**
   - Finally, the gated output is **linearly projected down** to the original model dimension (`5120`)(using weight matrix w2).
   - This completes the expert transformation.

# Expert Walk Through



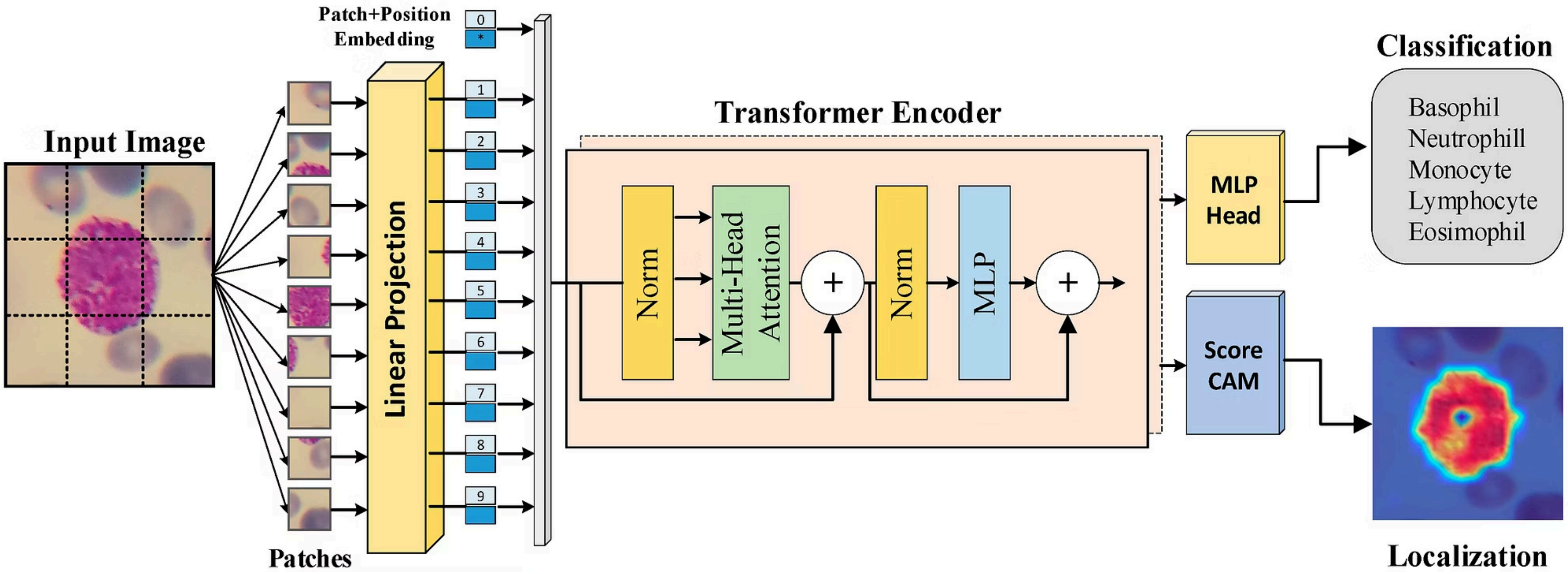
# Output for MOE

```
Initialized Llama4TextMoe:
  Num experts       : 16
  Experts per token : 1
  Hidden dim        : 5120

[TextBlock] Input shape: torch.Size([1, 10, 5120])
[TextBlock] After LayerNorm: torch.Size([1, 10, 5120])

[Llama4TextMoe] Input shape: torch.Size([1, 10, 5120])
[Step 1] Flattened tokens for routing: torch.Size([10, 5120])
[Step 2] Router logits shape: torch.Size([10, 16])
[Step 3] Top-k values shape: torch.Size([10, 1])
[Step 3] Top-k indices shape: torch.Size([10, 1])
Example top-2 experts for first token: [11]
[Step 4] Router score matrix (after sigmoid): torch.Size([16, 10])
[Step 5] Repeat indices shape: torch.Size([16, 10])
[Step 6] Expert inputs shape (before masking): torch.Size([160, 5120])
[Step 7] Expert inputs shape (after masking): torch.Size([160, 5120])
[Step 8] Expert outputs shape: torch.Size([160, 5120])
[Step 9] Shared MLP output shape: torch.Size([10, 5120])
[Step 10] Output after combining expert + shared: torch.Size([10, 5120])
[TextBlock] After MoE output shape: torch.Size([1, 10, 5120])
[TextBlock] Output shape (with residual): torch.Size([1, 10, 5120])
```

# Vision Transformer(ViT)



# Benefits of MoE in LLaMA 4

### Compute Efficiency

Only a fraction of the model's parameters are active per token, drastically reducing computational cost during inference.

### Specialization

Experts can learn to specialize in different aspects of the data, leading to higher quality representations and outputs.

### Scalability

Enables training and deploying models with trillions of parameters while keeping inference costs manageable.