Introduction to Mixture of Experts (Part 2)

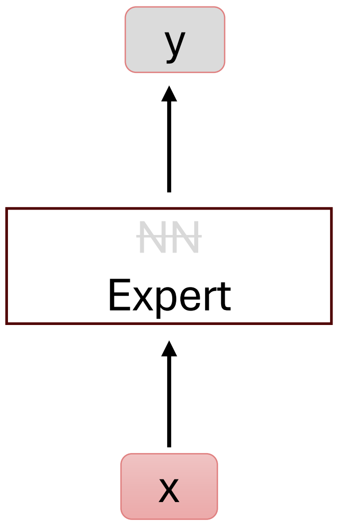
Yatin Nandwani
Research Scientist, IBM Research

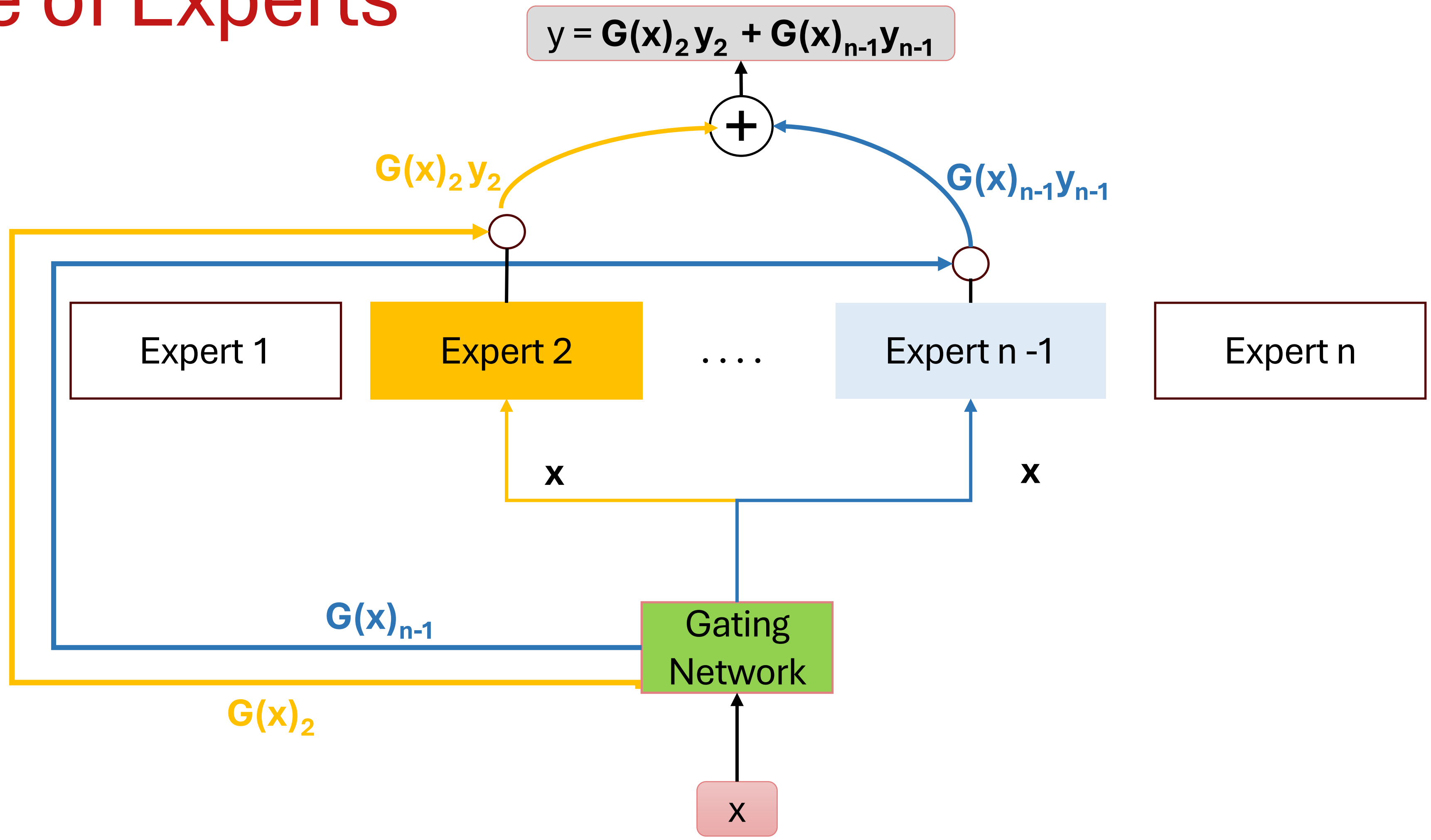
Large Language Models: Introduction and Recent Advances

ELL881 · AIL821

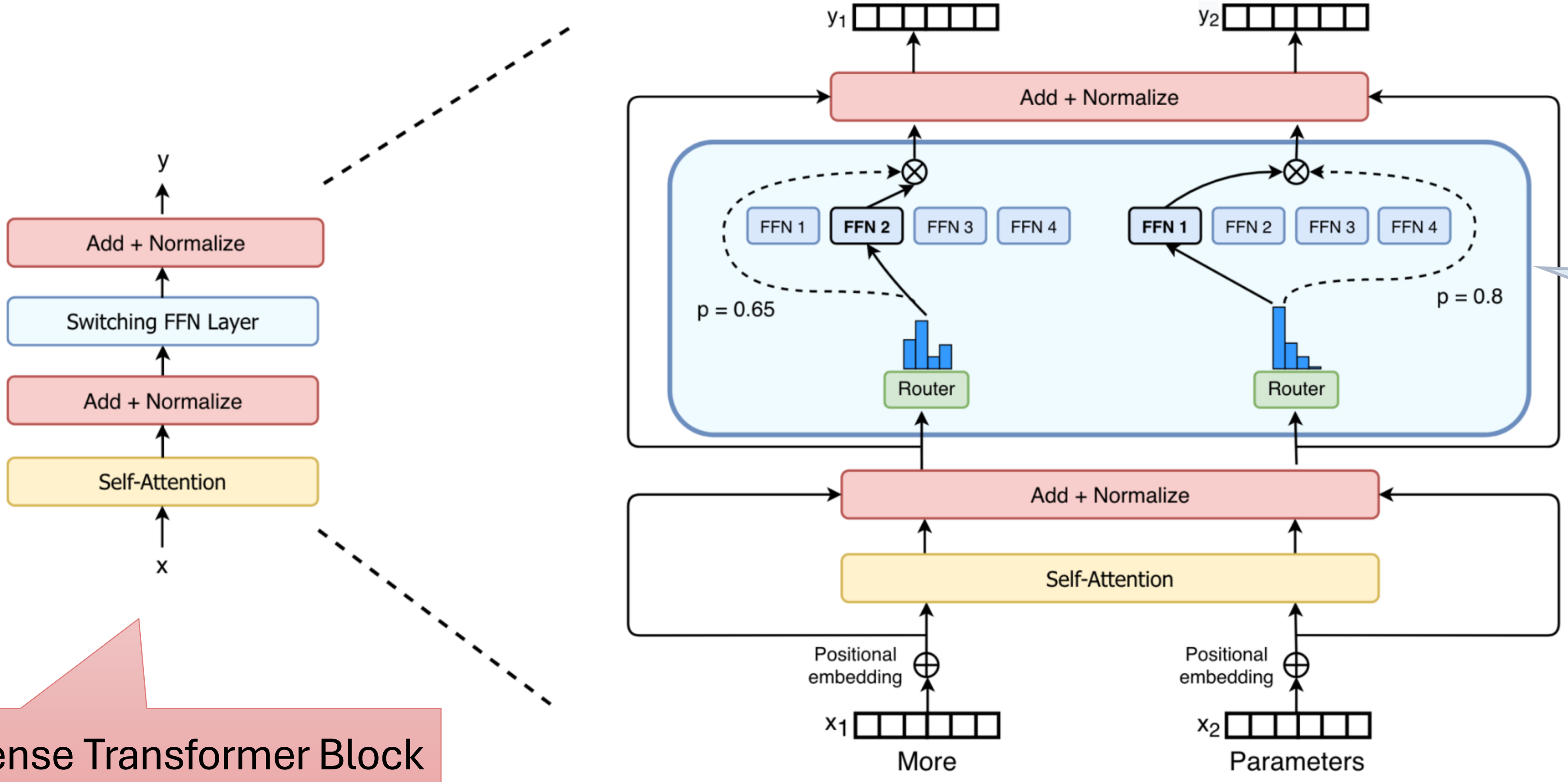
Semester 1, 2024-2025

Sparse Mixture of Experts





Sparse Mixture of Experts as a Layer



MoE Layer

Dense Transformer Block

Sparsely Activated MoE Transformer Block

Pros and Cons of Sparse MoE Layer

Pros

- 👍 Increased model parameters
- 👍 Efficient pretraining due to conditional (sparse) computation
- 👍 Faster inference

Cons

- 👎 Unstable training
  - ☹️ Router collapse– router sends all tokens to the same expert
  - ☹️ May diverge
- 👎 High memory requirement - all parameters need to be loaded in vRAM (GPU memory)

Switch Transformer Layer

Content credits: Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
https://www.youtube.com/watch?v=U8J32Z3qV8s&t=2816s

Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity

William Fedus*
LIAMFEDUS@GOOGLE.COM

Barret Zoph*
BARRETZOPH@GOOGLE.COM

Noam Shazeer
NOAM@GOOGLE.COM

Google, Mountain View, CA 94043, USA

Editor: Alexander Clark

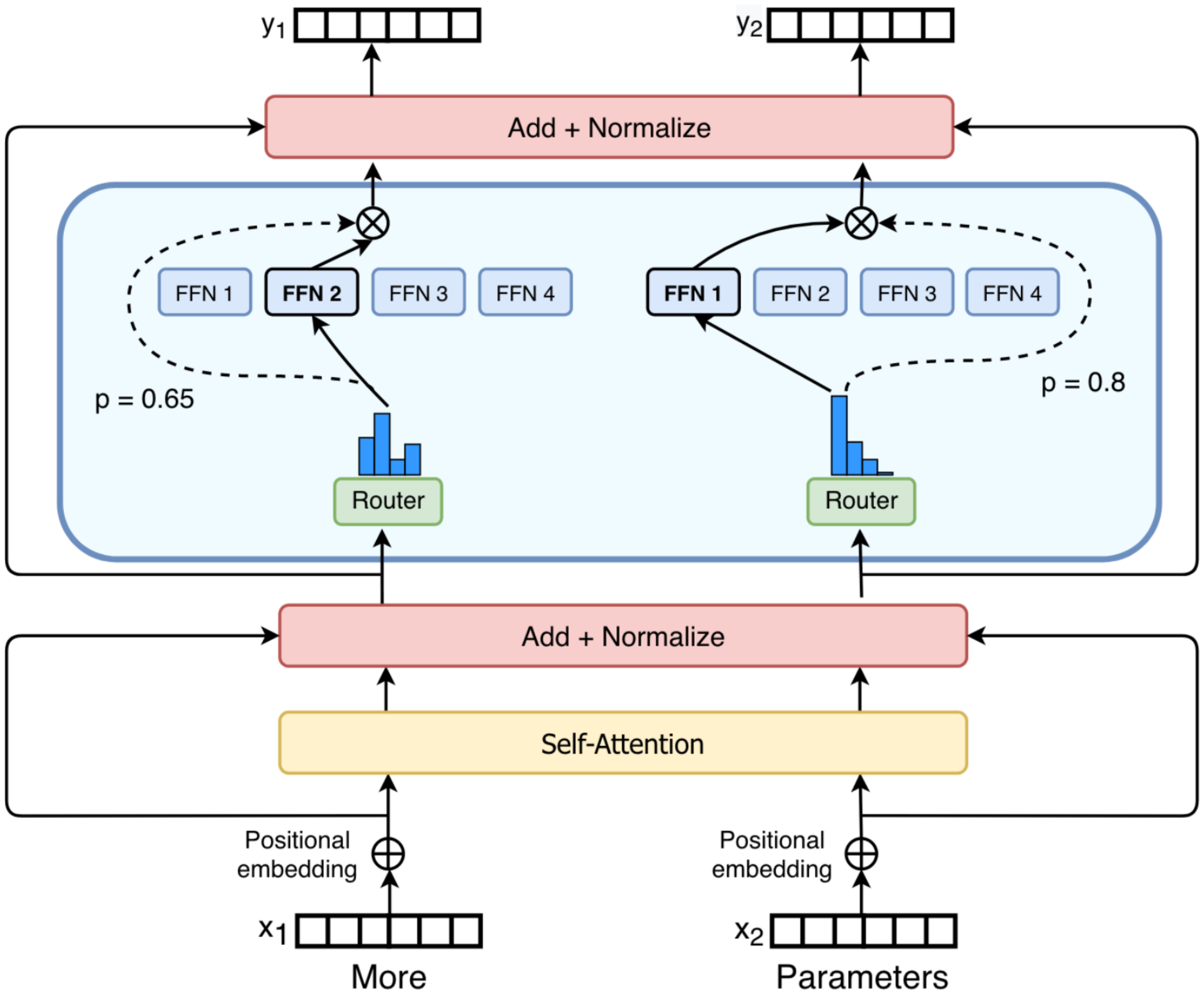
Abstract

In deep learning, models typically reuse the same parameters for all inputs. Mixture of Experts (MoE) models defy this and instead select *different* parameters for each incoming example. The result is a sparsely-activated model—with an outrageous number of parameters—but a constant computational cost. However, despite several notable successes of MoE, widespread adoption has been hindered by complexity, communication costs, and training instability. We address these with the introduction of the Switch Transformer.

Content credits: Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
https://www.youtube.com/watch?v=U8J32Z3qV8s&t=2816s

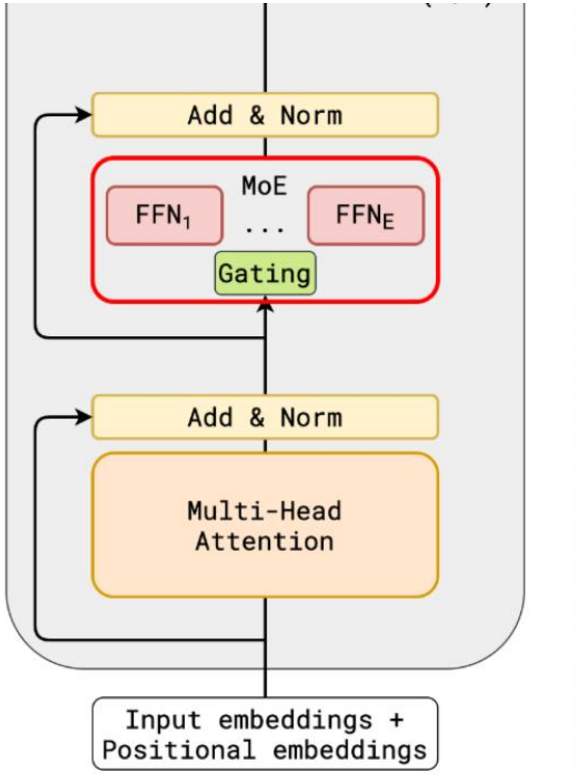
Switch Transformer Layer

- Greedy routing to only 1 expert

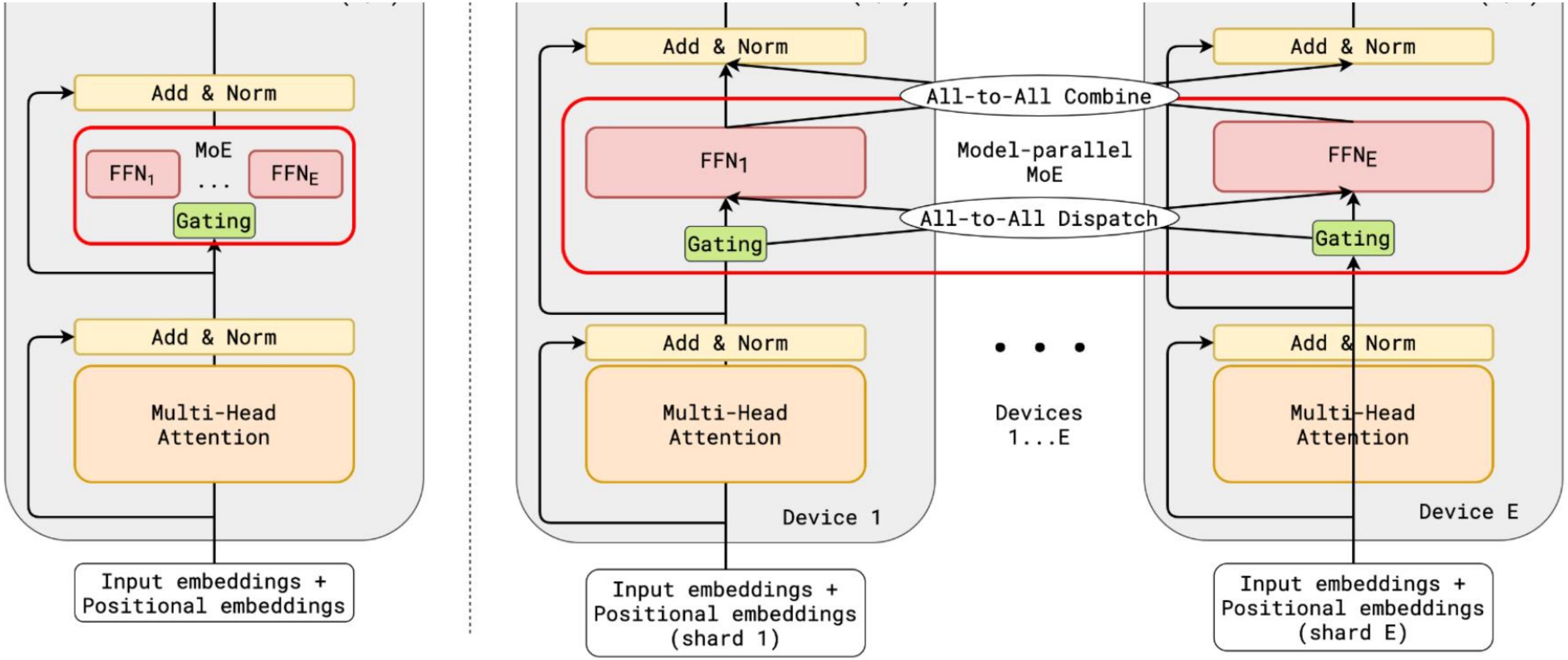


Content credits: Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
https://www.youtube.com/watch?v=U8J32Z3qV8s&t=2816s

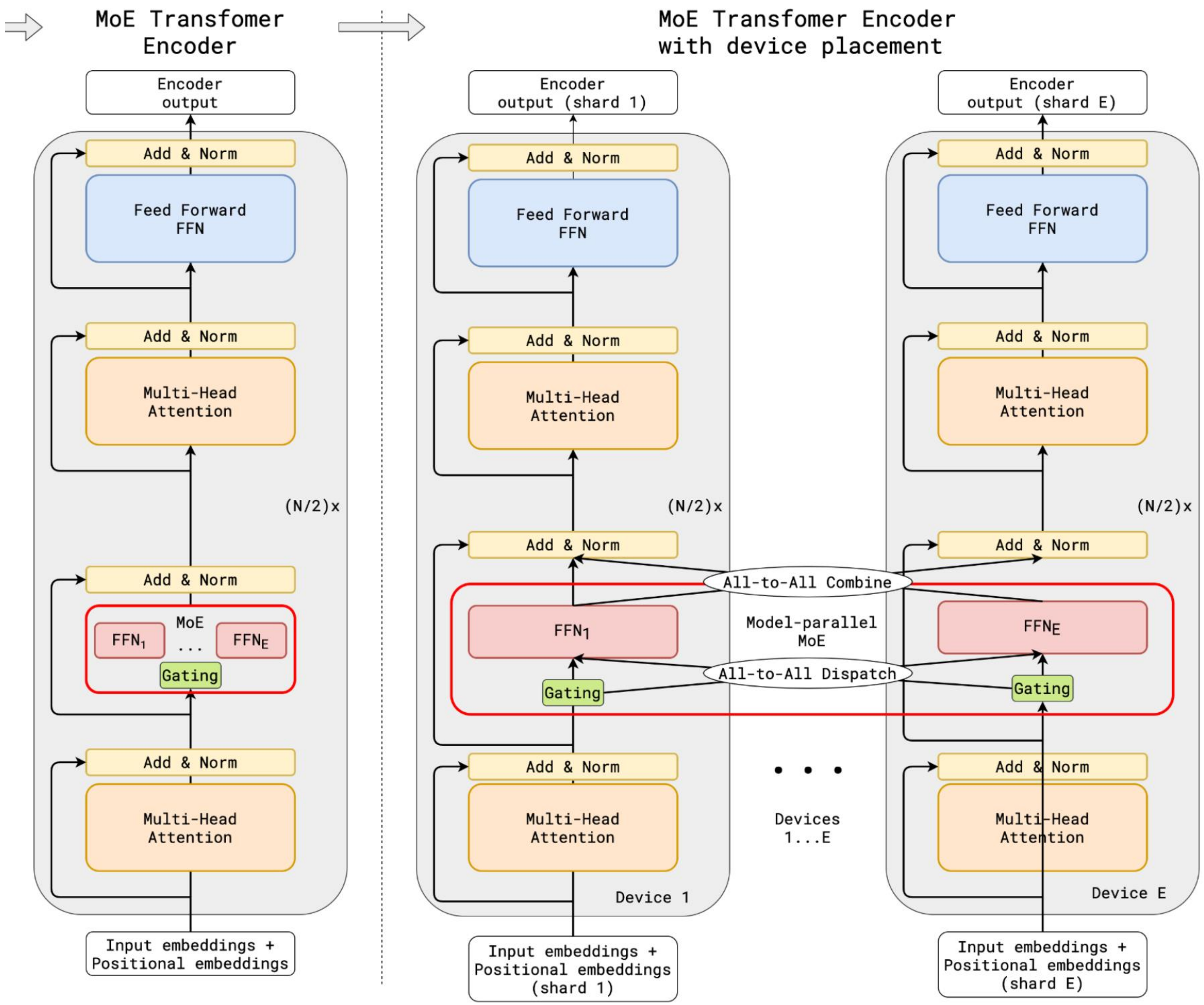
Expert Parallel for Sparse MoEs



Expert Parallel for Sparse MoEs



Expert Parallel for Sparse MoEs



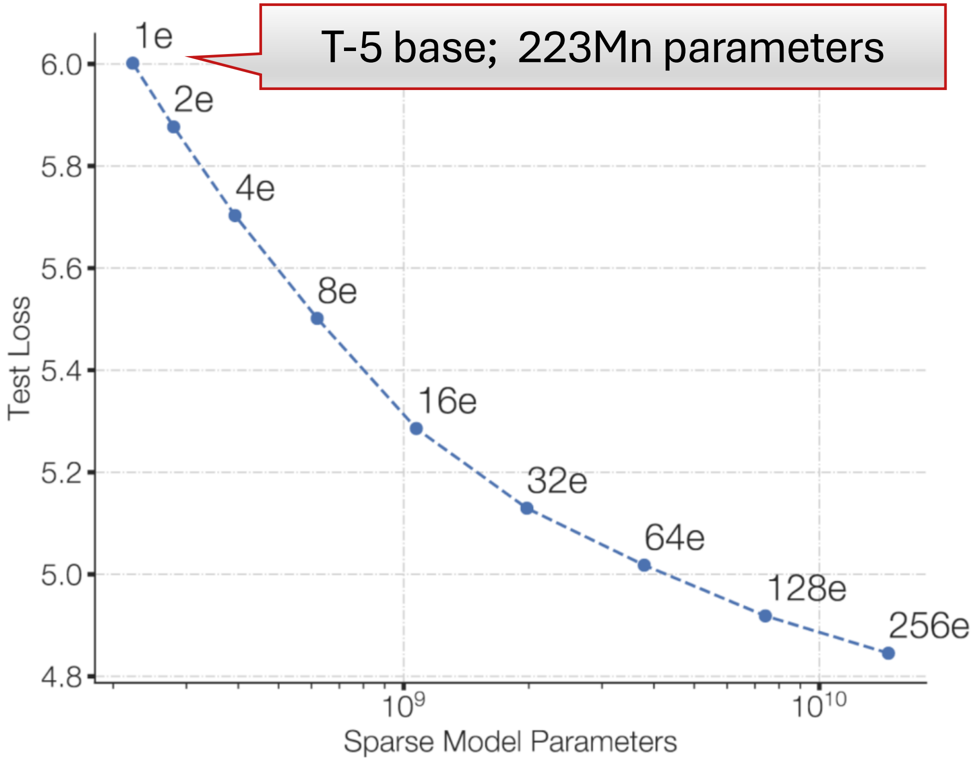
Switch Transformer Layer

- MoE-fication of T5 models

Content credits: Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
https://www.youtube.com/watch?v=U8J32Z3qV8s&t=2816s

Switch Transformer Layer

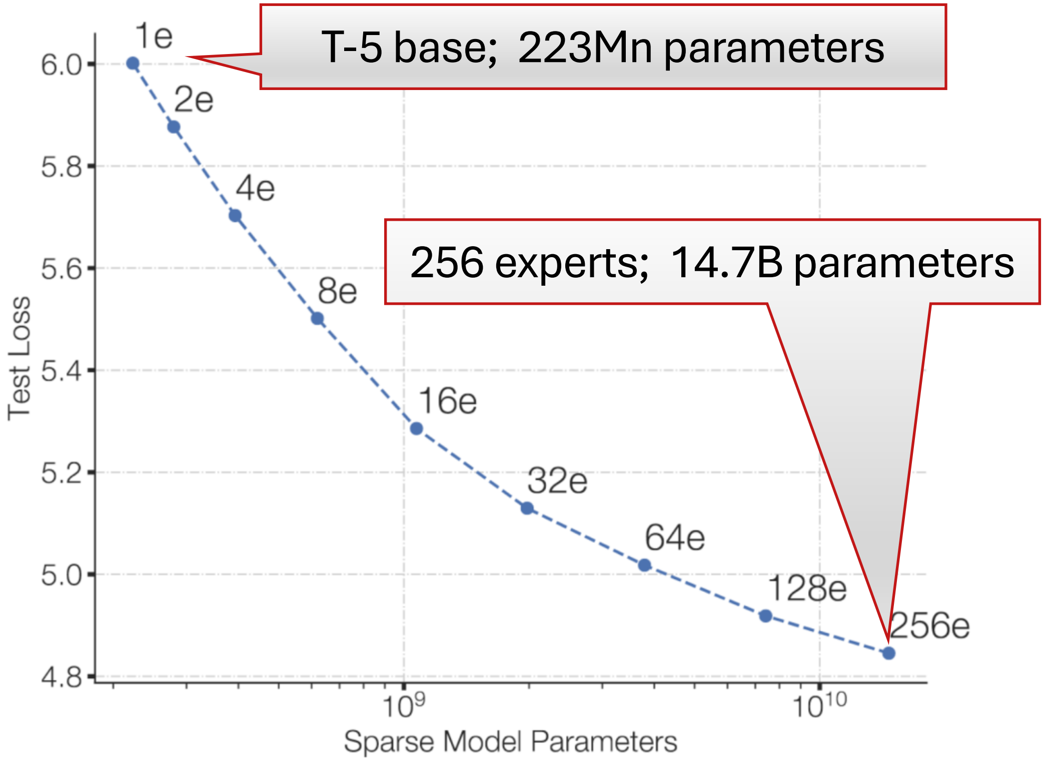
- MoE-fication of T5 models



Content credits: Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
https://www.youtube.com/watch?v=U8J32Z3qV8s&t=2816s

Switch Transformer Layer

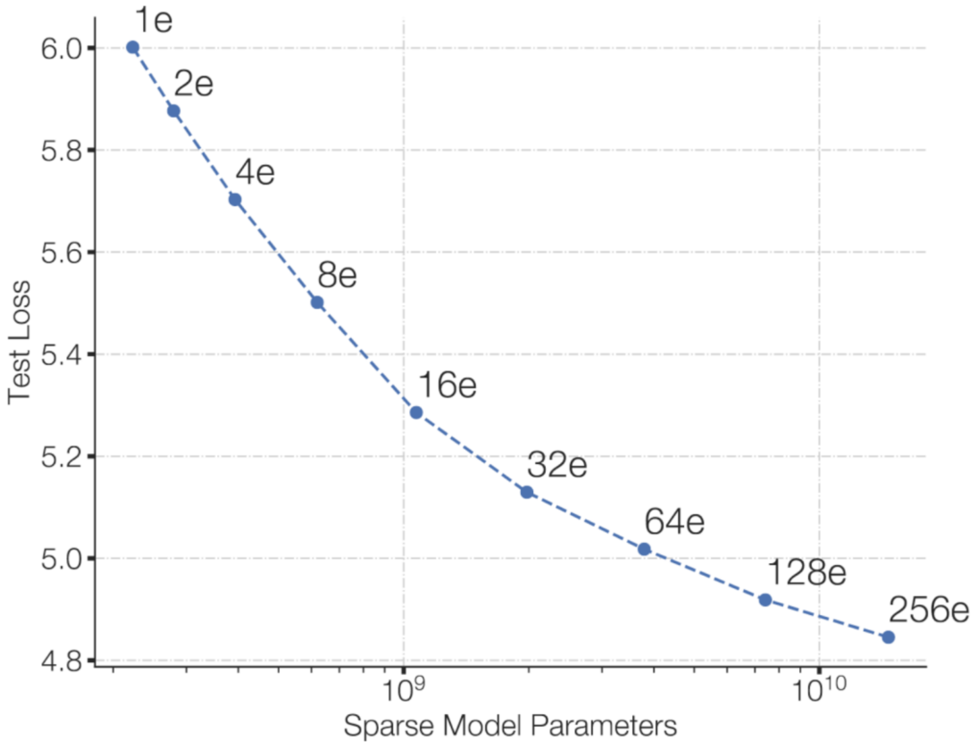
- MoE-fication of T5 models

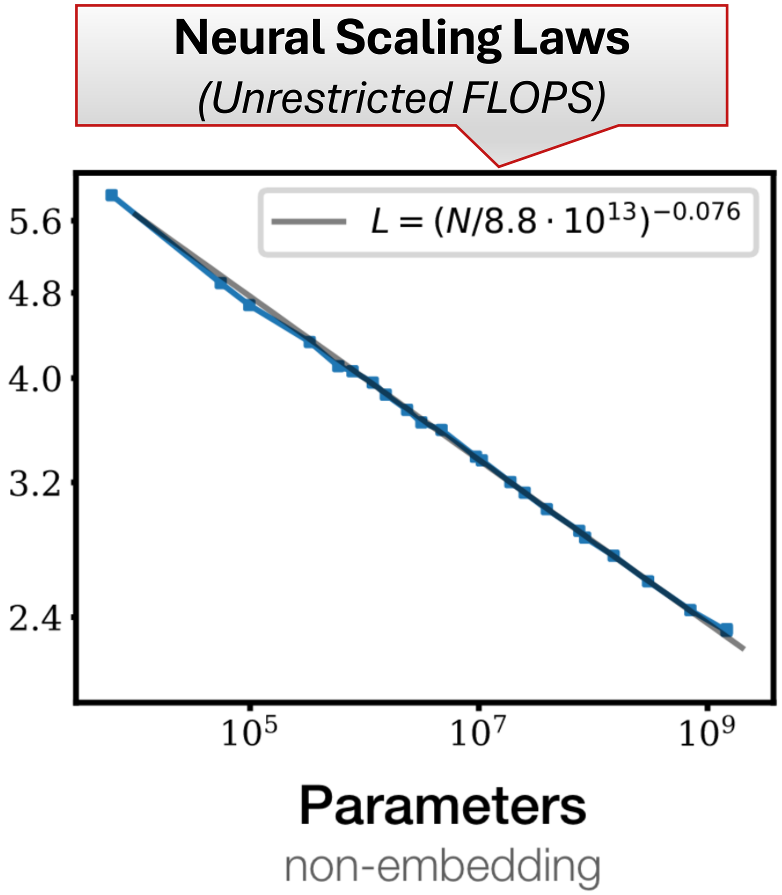


Content credits: Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
https://www.youtube.com/watch?v=U8J32Z3qV8s&t=2816s

Switch Transformer Layer

- MoE-fication of T5 models



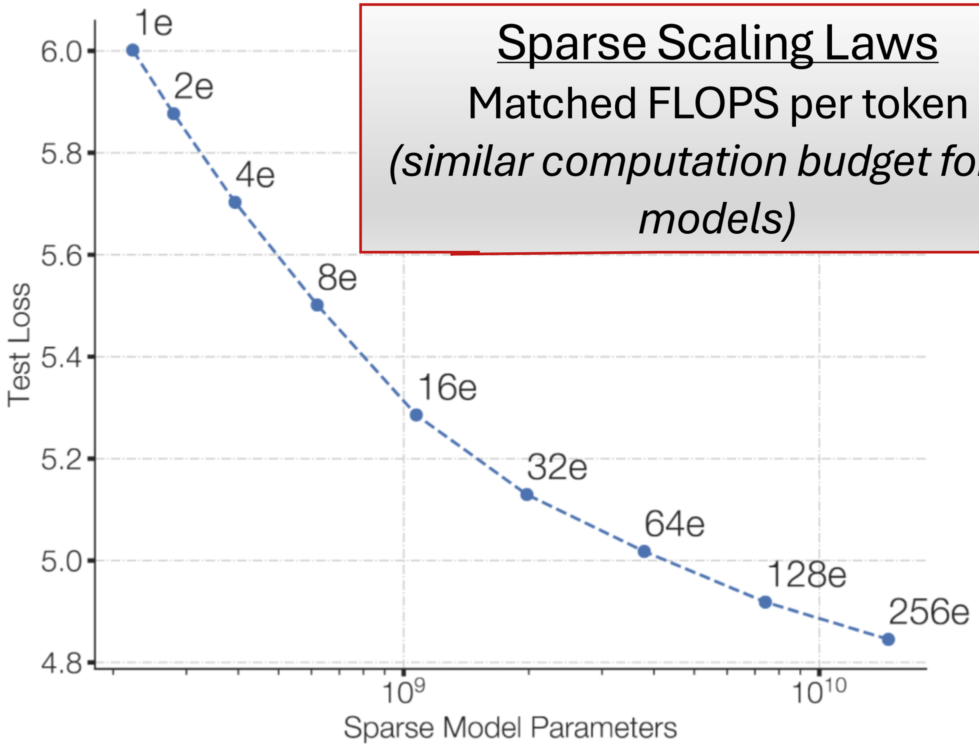


Content credits: Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
https://www.youtube.com/watch?v=U8J32Z3qV8s&t=2816s

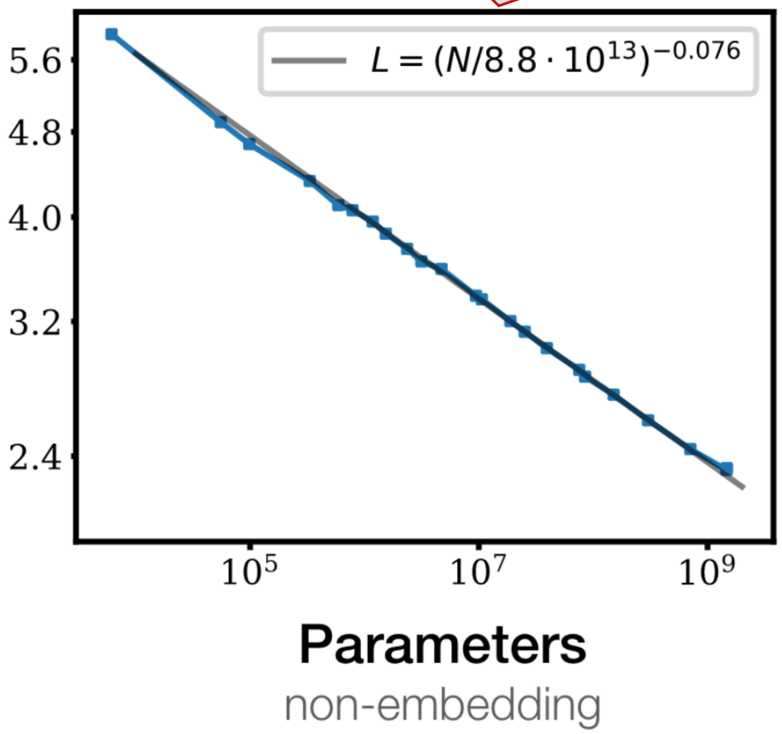
Switch Transformer Layer

- MoE-fication of T5 models

Sparse Scaling Laws
Matched FLOPS per token
(similar computation budget for all models)



Neural Scaling Laws
(Unrestricted FLOPS)

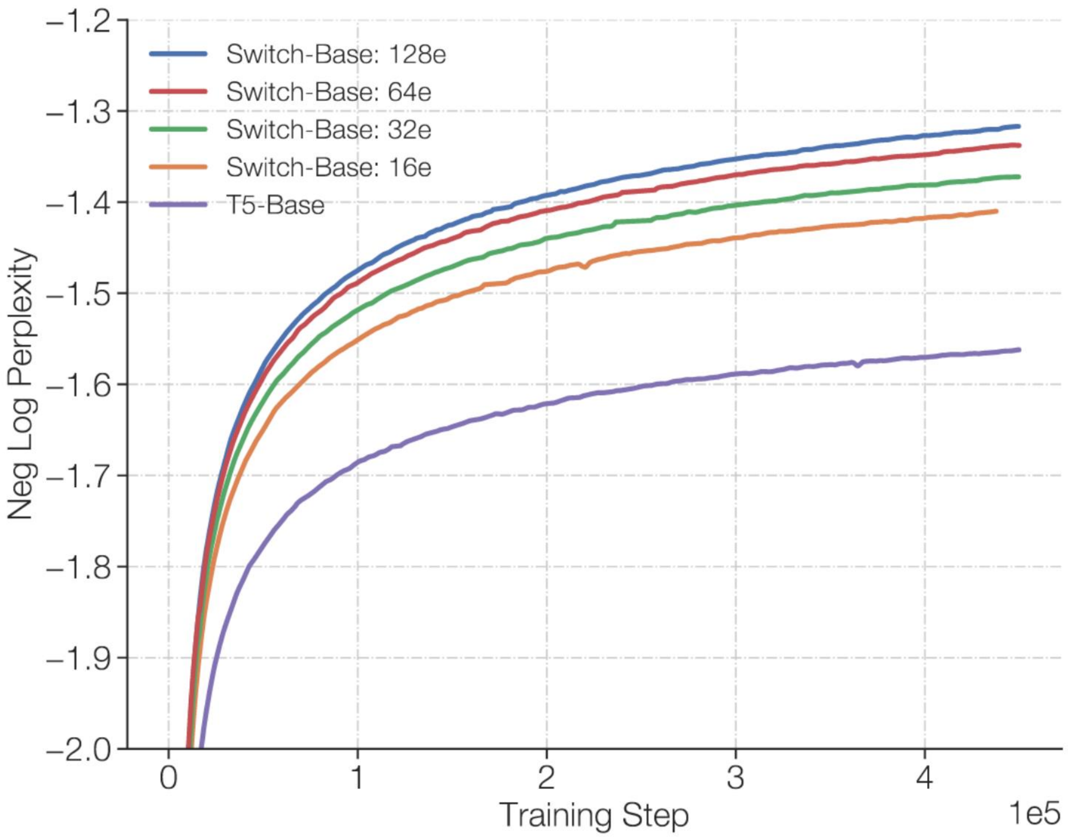


Content credits: Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
https://www.youtube.com/watch?v=U8J32Z3qV8s&t=2816s

Switch Transformer Layer

- MoE-fication of T5 models

On C4 corpus *(introduced in T-5 paper)*



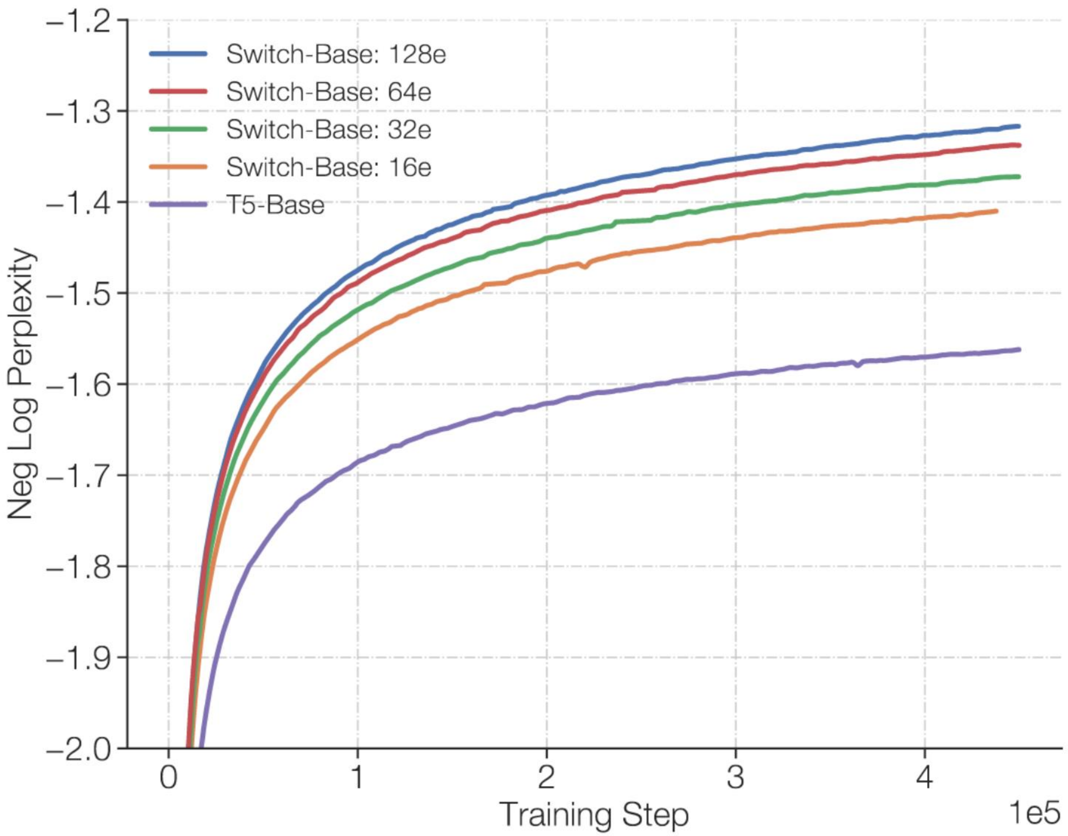
- ❖ Better asymptotic performance
- ❖ Improved sample efficiency
- ❖ Diminishing returns as we increase #experts

Content credits: Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
https://www.youtube.com/watch?v=U8J32Z3qV8s&t=2816s

Switch Transformer Layer

- MoE-fication of T5 models

FLOPS per token are matched, but additional clock time due to:
1. Extra communication cost
2. Router computation

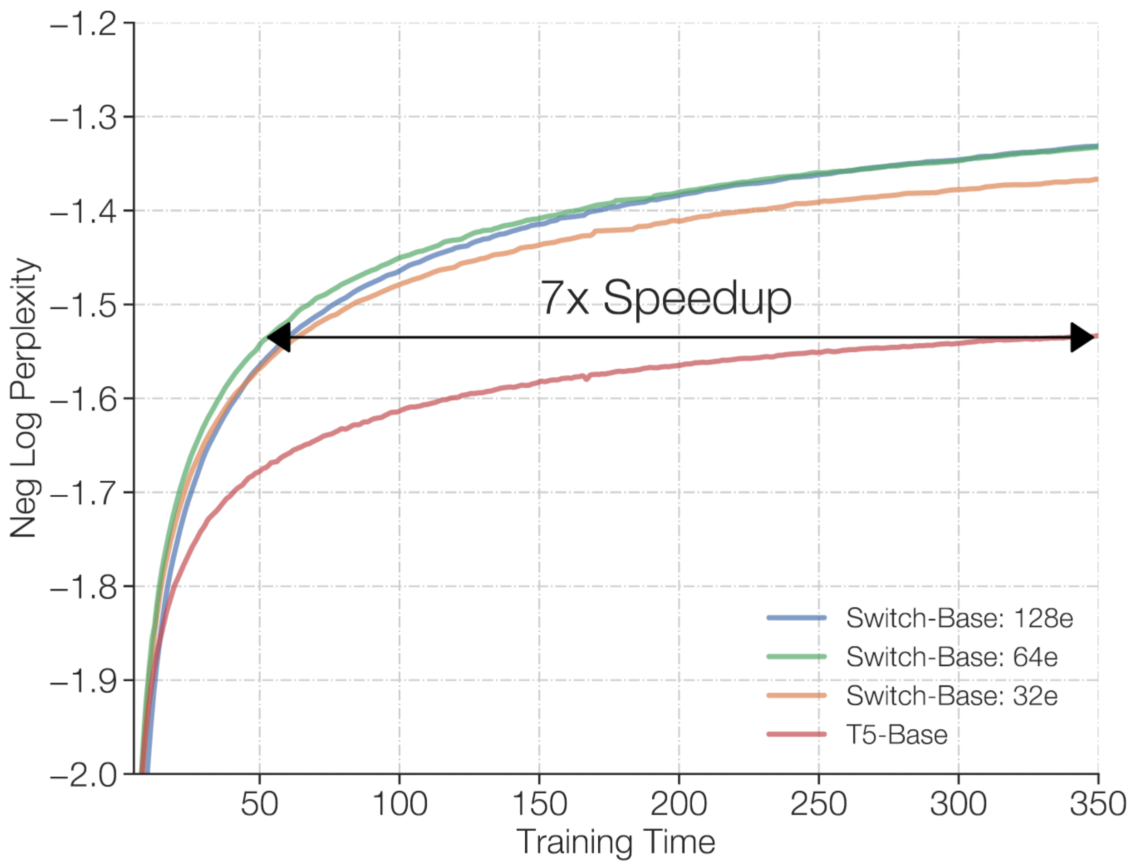


Content credits: Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
https://www.youtube.com/watch?v=U8J32Z3qV8s&t=2816s

Switch Transformer Layer

- MoE-fication of T5 models

7x faster than the base model!



Content credits: Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
https://www.youtube.com/watch?v=U8J32Z3qV8s&t=2816s

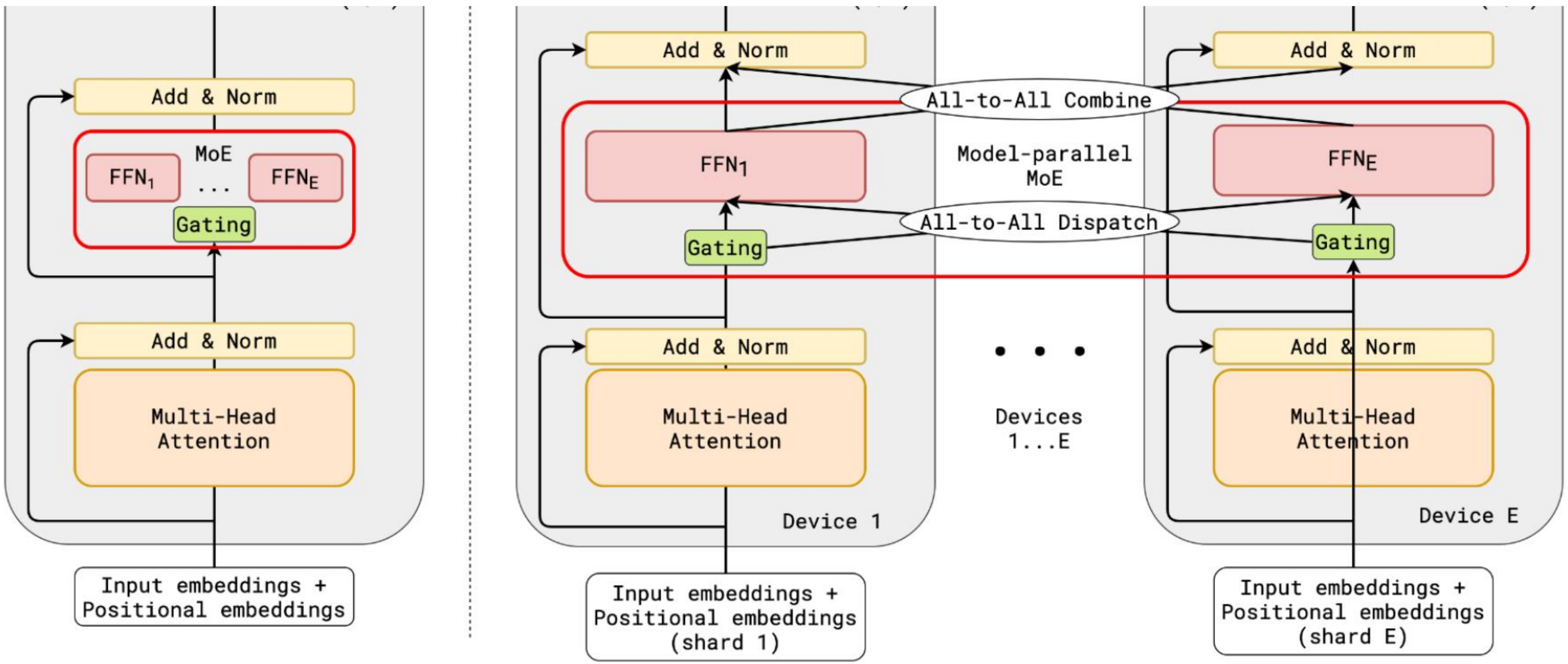
Switch Transformer Layer

- MoE-fication of T5 models

But what about comparison with a larger dense model?

Content credits: Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
https://www.youtube.com/watch?v=U8J32Z3qV8s&t=2816s

Expert Parallel for Sparse MoEs



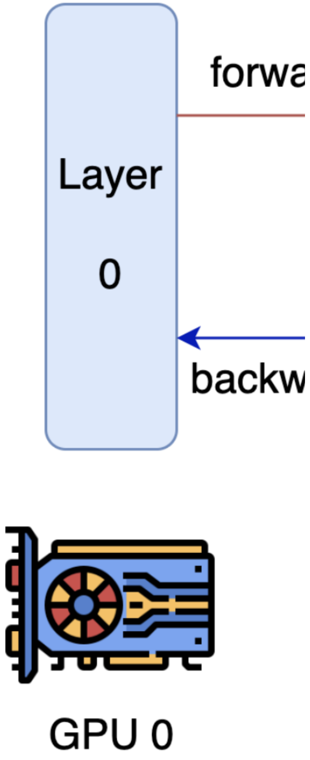
Model Parallelism for Larger Dense Model

- **Pipeline Parallelism:**
  - Different Layers on different devices

Content credits: \https://colossalai.org/docs/concepts/paradigms_of_parallelism/

Model Parallelism for Larger Dense Model

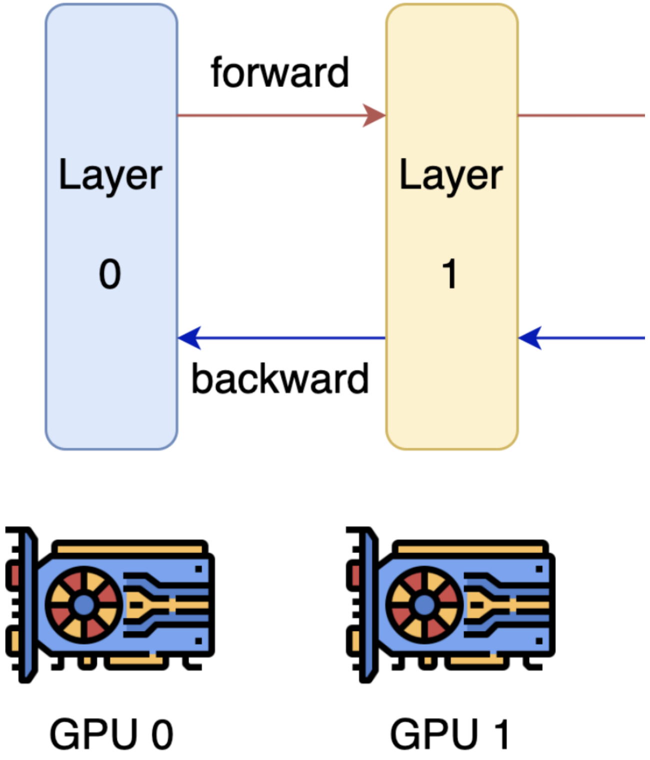
- **Pipeline Parallelism:**
  - Different Layers on different devices



Content credits: \https://colossalai.org/docs/concepts/paradigms_of_parallelism/

Model Parallelism for Larger Dense Model

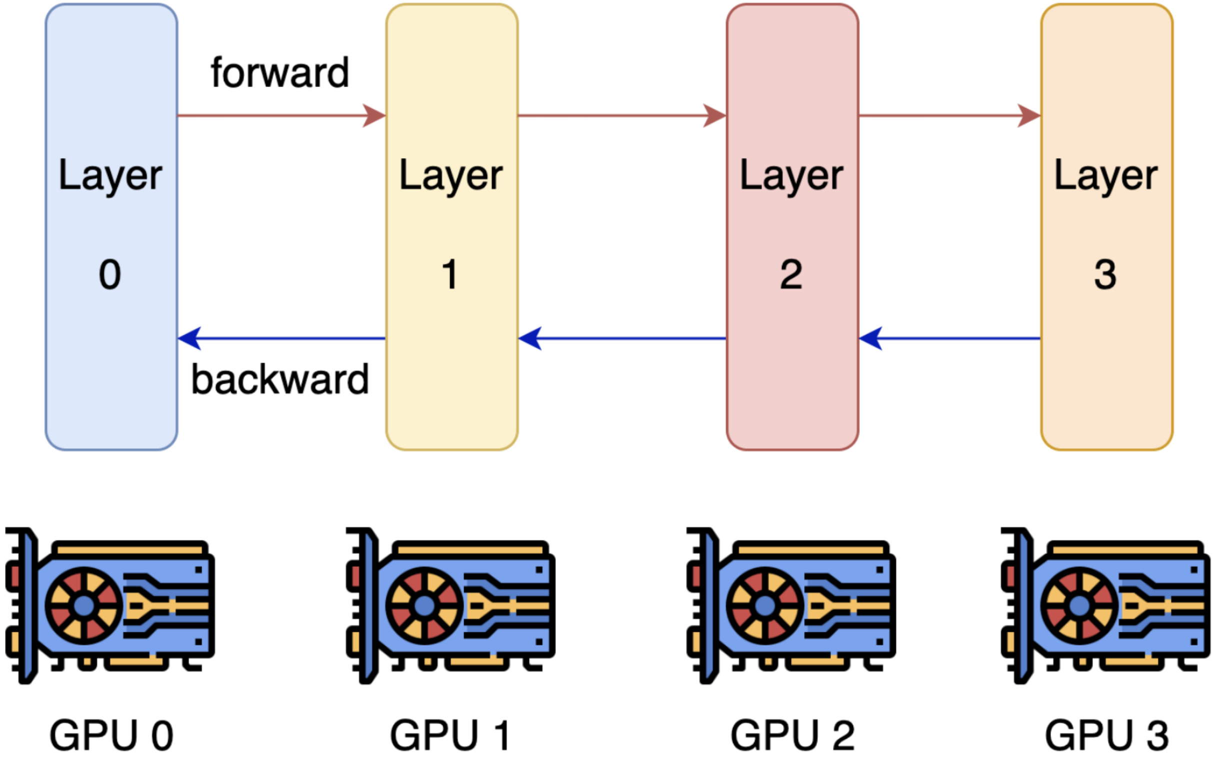
- **Pipeline Parallelism:**
  - Different Layers on different devices



Content credits: \https://colossalai.org/docs/concepts/paradigms_of_parallelism/

Model Parallelism for Larger Dense Model

- **Pipeline Parallelism:**
  - Different Layers on different devices



Content credits: \https://colossalai.org/docs/concepts/paradigms_of_parallelism/

Model Parallelism for Larger Dense Model

- **Pipeline Parallelism:**
  - Different Layers on different devices

- **Tensor Parallelism:**

Content credits: \https://colossalai.org/docs/concepts/paradigms_of_parallelism/

Model Parallelism for Larger

- **Pipeline Parallelism:**
  - Different Layers on different devices

- **Tensor Parallelism:**
  1. Column-wise splitting

Column-Splitting Tensor Parallel

of_parallelism/

Model Parallelism for Larger

Non-distributed

- **Pipeline Parallelism:**
  - Different Layers on different devices

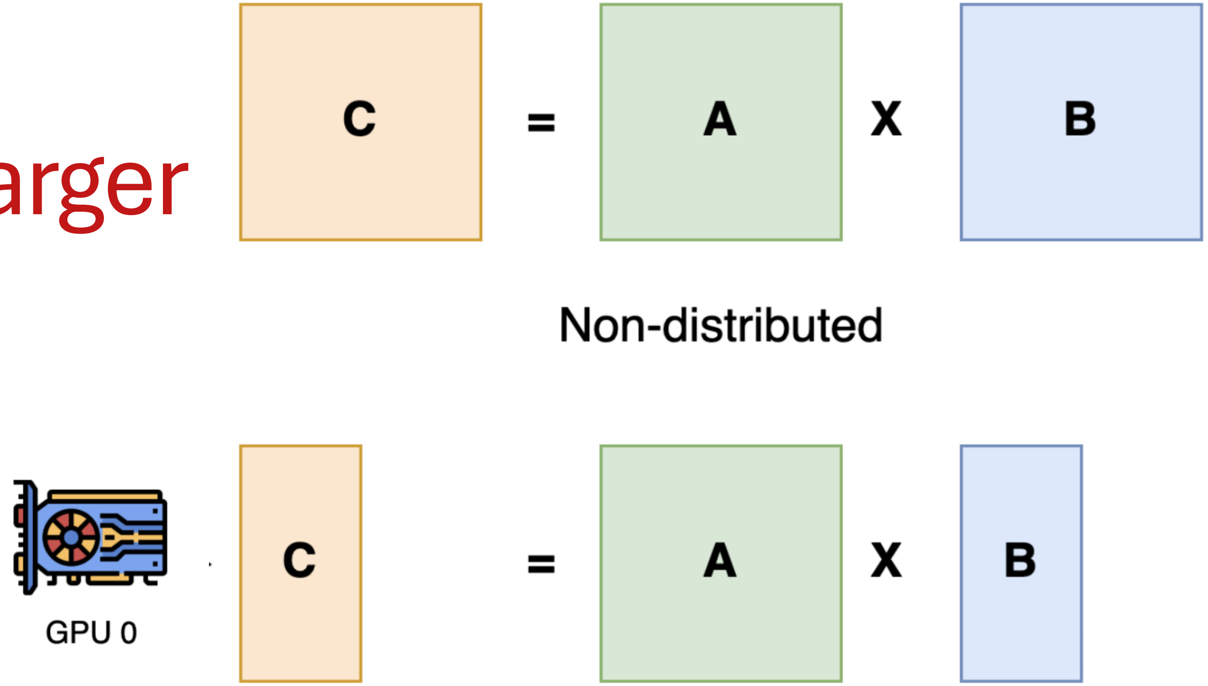
- **Tensor Parallelism:**
  1. Column-wise splitting

Column-Splitting Tensor Parallel

of_parallelism/

Model Parallelism for Larger

- **Pipeline Parallelism:**
  - Different Layers on different devices
- **Tensor Parallelism:**
  1. Column-wise splitting

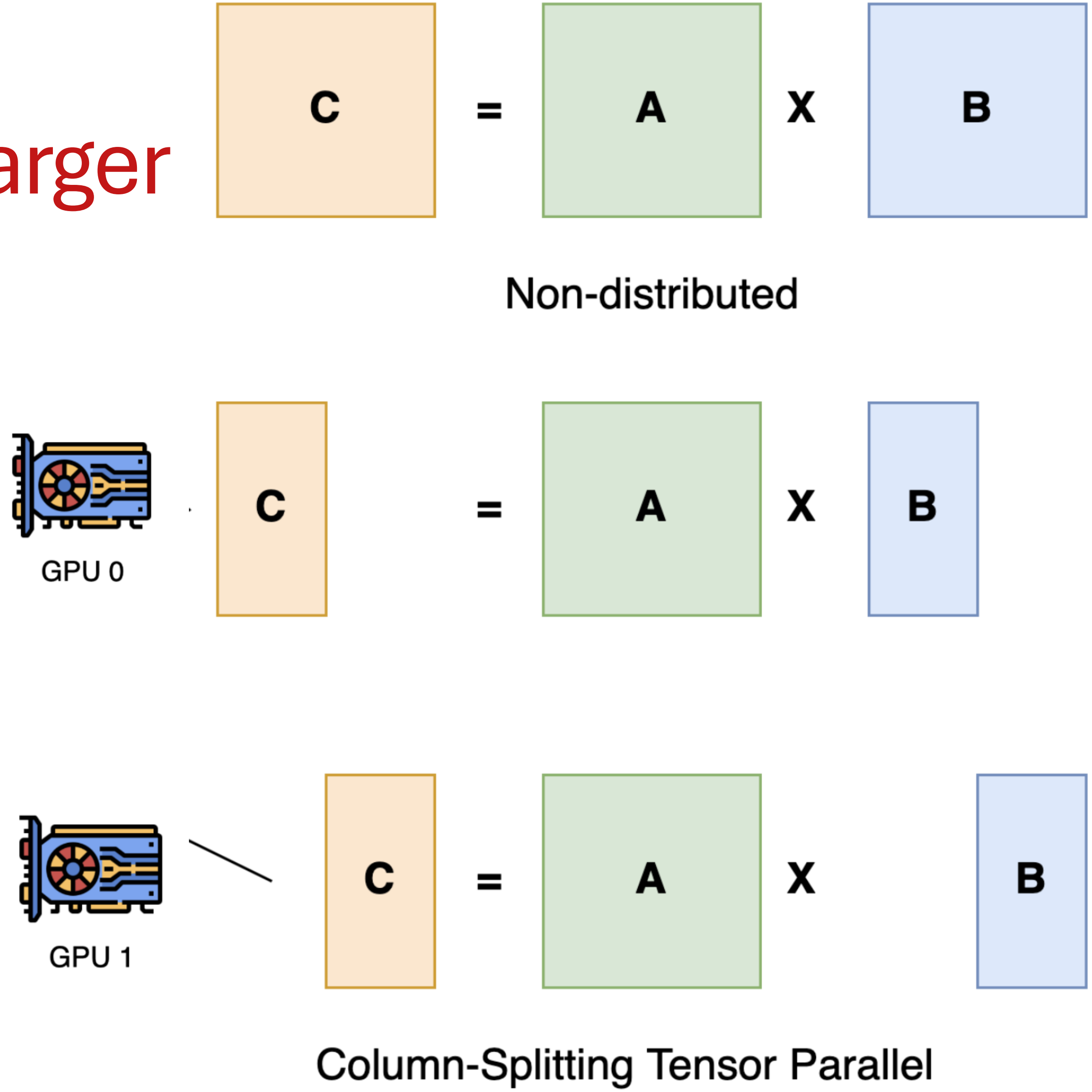


Column-Splitting Tensor Parallel

of_parallelism/

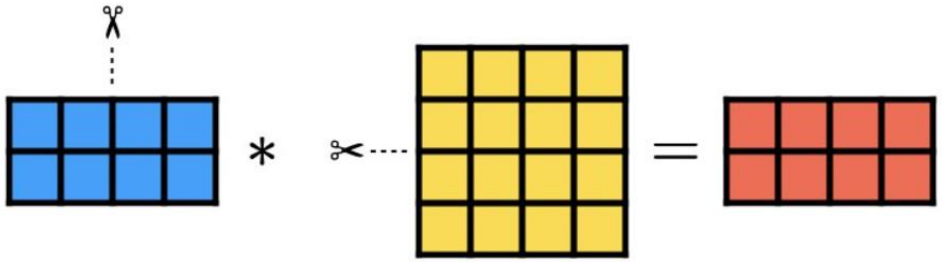
Model Parallelism for Larger

- **Pipeline Parallelism:**
  - Different Layers on different devices
- **Tensor Parallelism:**
  1. Column-wise splitting



of_parallelism/

Model Parallelism for Larger

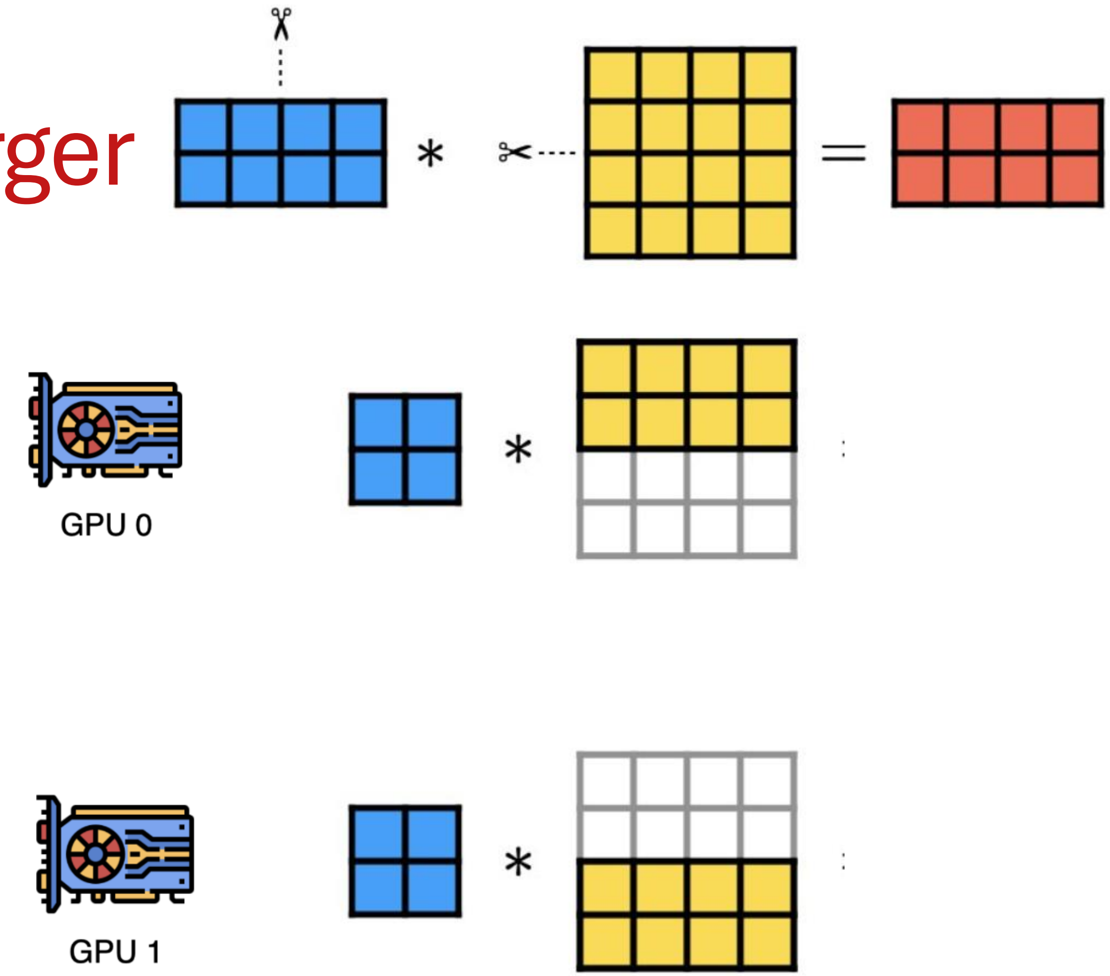


- **Pipeline Parallelism:**
    - Different Layers on different devices

- **Tensor Parallelism:**
    1. Column-wise splitting
    2. Row-wise splitting

Content credits: https://lightning.ai/docs/pytorch/stable/advanced/model_parallel/tp.html

Model Parallelism for Larger

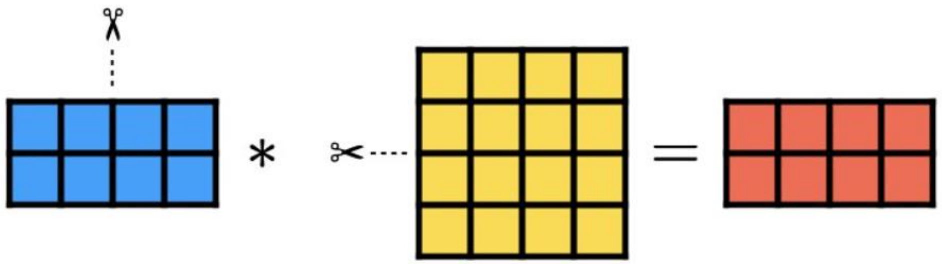


- **Pipeline Parallelism:**
    - Different Layers on different devices

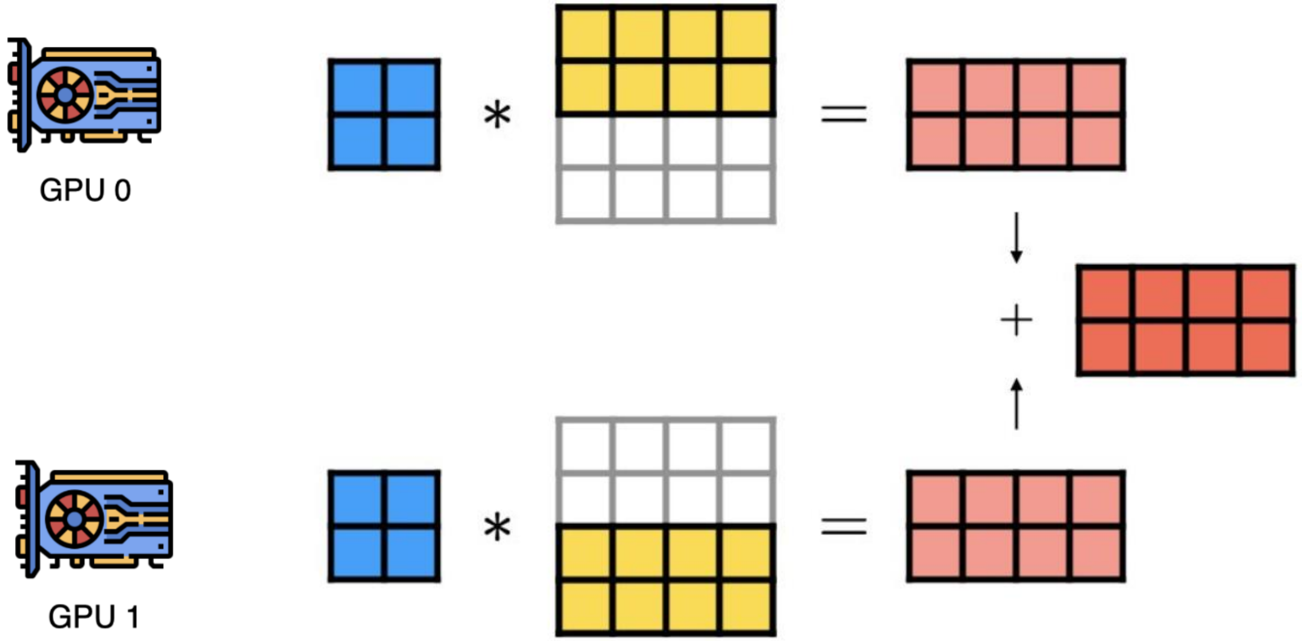
- **Tensor Parallelism:**
    1. Column-wise splitting
    2. Row-wise splitting

Content credits: https://lightning.ai/docs/pytorch/stable/advanced/model_parallel/tp.html

Model Parallelism for Larger



- **Pipeline Parallelism:**
  - Different Layers on different devices

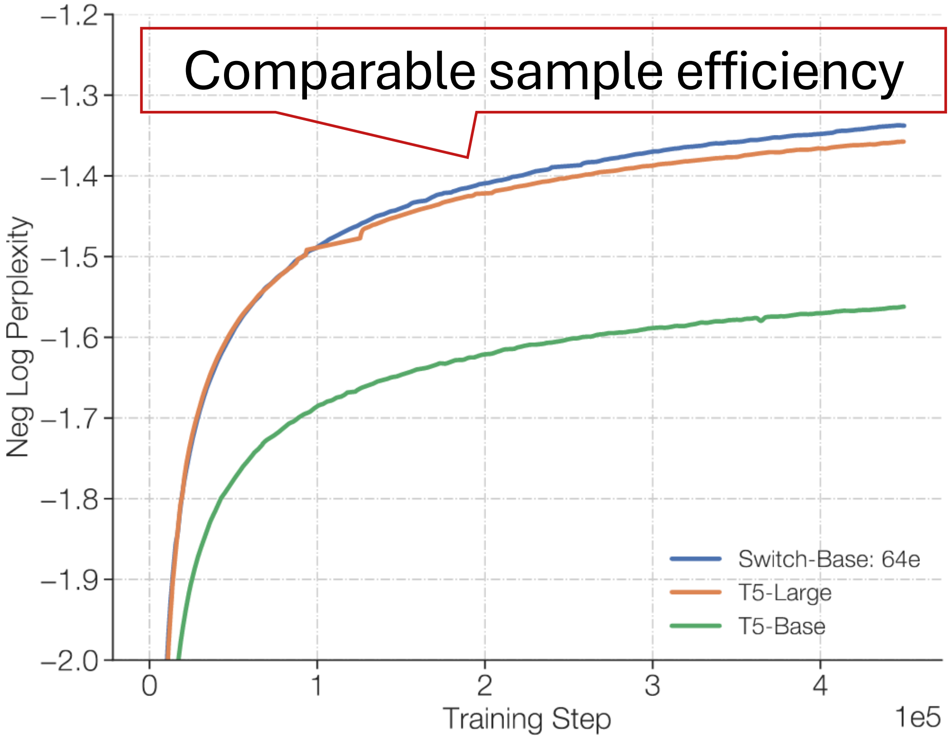


- **Tensor Parallelism:**
  1. Column-wise splitting
  2. Row-wise splitting

Content credits: https://lightning.ai/docs/pytorch/stable/advanced/model_parallel/tp.html

Switch Transformer Layer

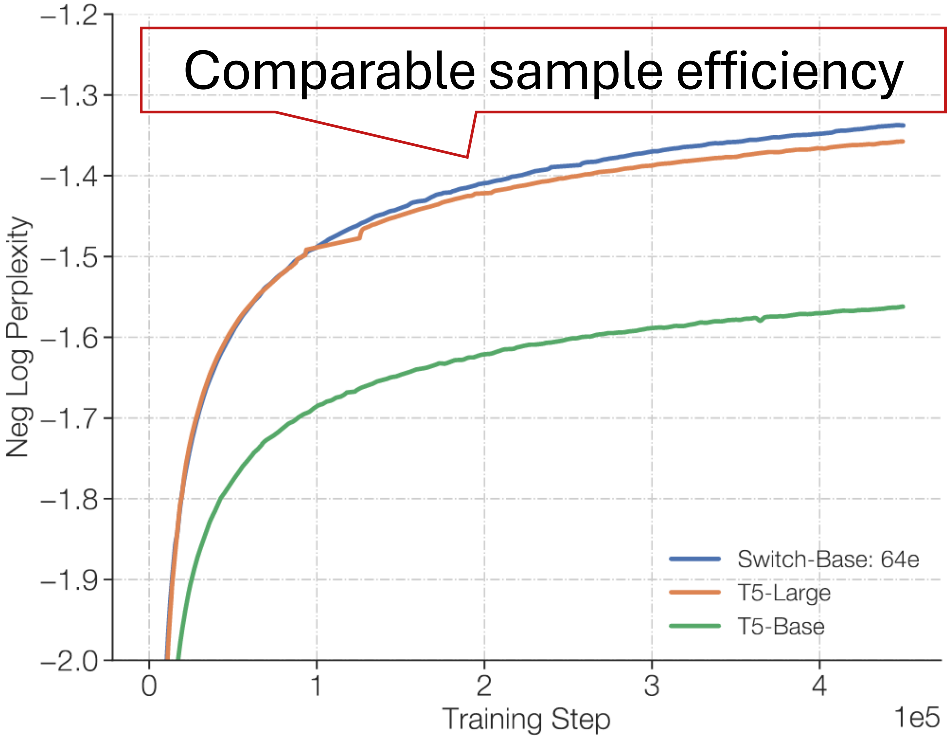
Comparison with T-5 Large (770M), with 3.5x more FLOPs per token

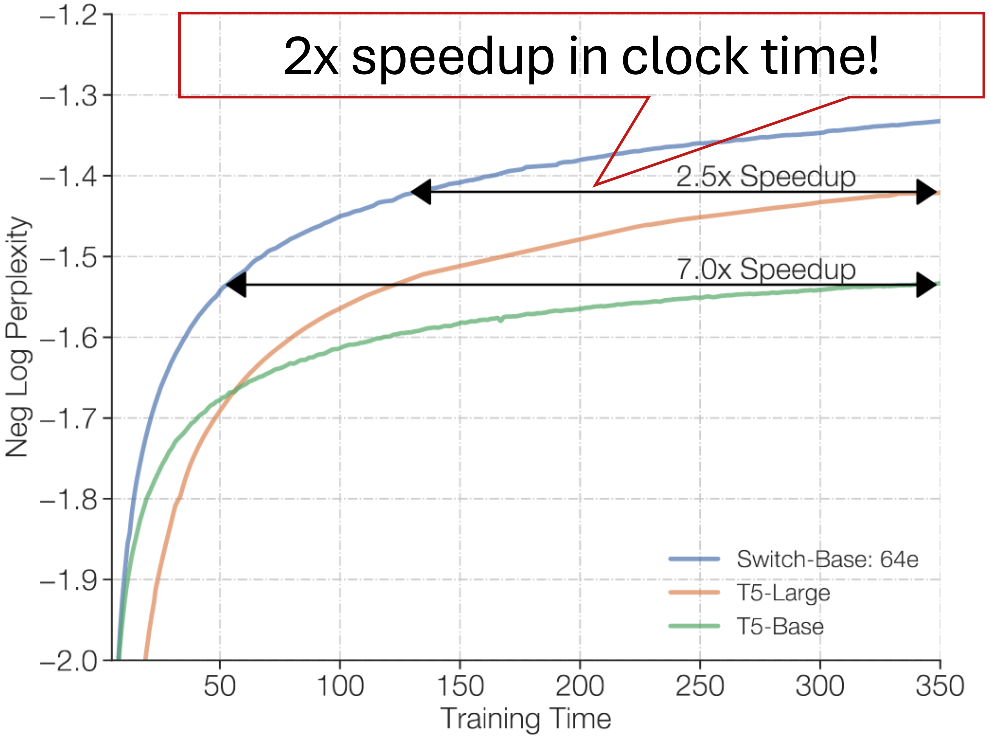


Content credits: Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
https://www.youtube.com/watch?v=U8J32Z3qV8s&t=2816s

Switch Transformer Layer

Comparison with T-5 Large (770M), with 3.5x more FLOPs per token

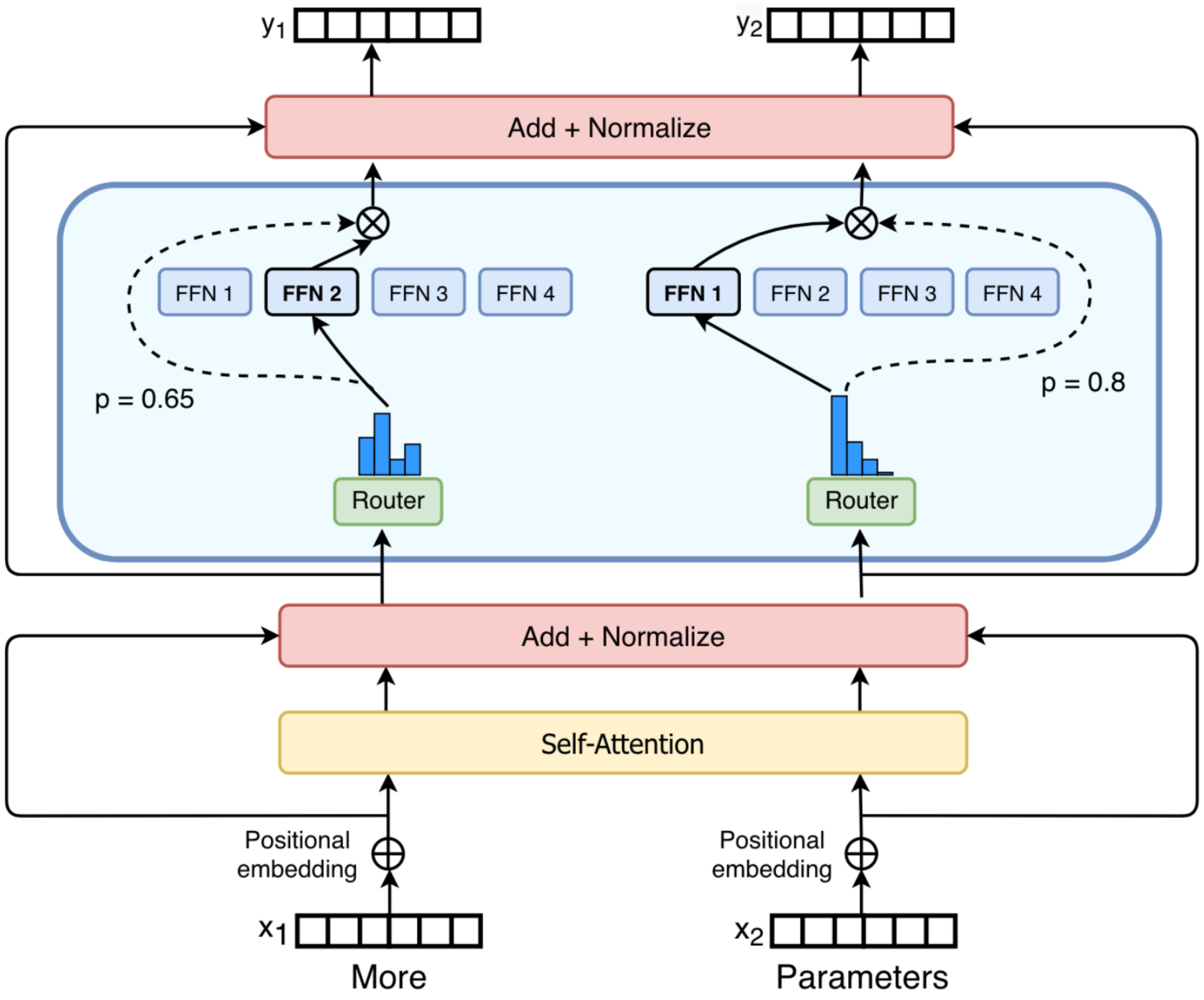




Content credits: Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
https://www.youtube.com/watch?v=U8J32Z3qV8s&t=2816s

Switch Transformer Layer

- **Issues Addressed:**
    - Complexity of MoE
    - Communication cost



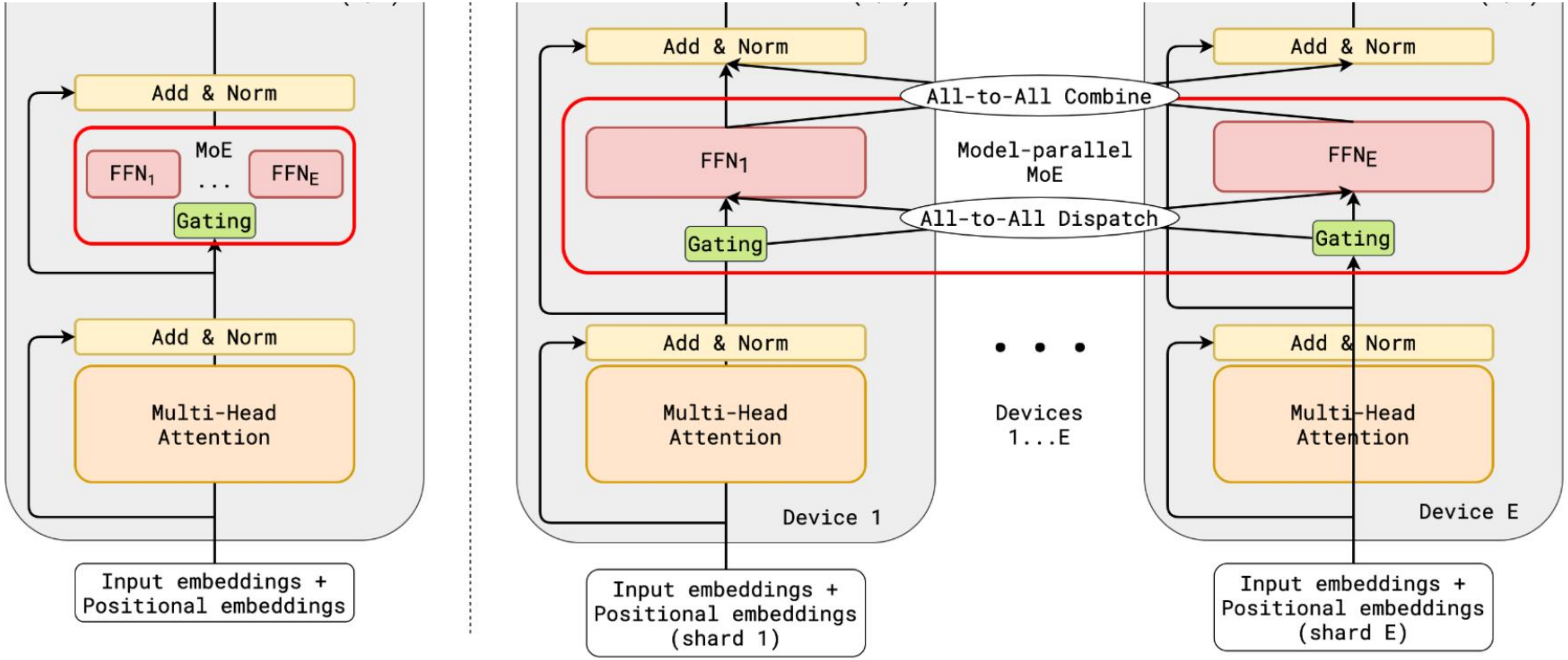
Content credits: Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
https://www.youtube.com/watch?v=U8J32Z3qV8s&t=2816s

Switch Transformer Layer

- **Issues Addressed:**
  - Complexity of MoE
  - Communication cost

Top-1 greedy routing: Challenged the belief that we need to route to at least 2 experts for meaningful learning of router

Content credits: Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
https://www.youtube.com/watch?v=U8J32Z3qV8s&t=2816s



Content credits: GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding

Switch Transformer Layer

- **Issues Addressed:**
    - Complexity of MoE
    - Communication cost
    - Training Instability

Top-1 greedy routing: Challenged the belief that we need to route to at least 2 experts for meaningful learning of router

Content credits: Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
https://www.youtube.com/watch?v=U8J32Z3qV8s&t=2816s

Switch Transformer Layer

- **Issues Addressed:**
  - Complexity of MoE
  - Communication cost
  - Training Instability

Top-1 greedy routing: Challenged the belief that we need to route to at least 2 experts for meaningful learning of router

Improved Training Techniques:

1. Differentiable load balancing loss (avoids router collapse)

Content credits: Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
https://www.youtube.com/watch?v=U8J32Z3qV8s&t=2816s

Load Balancing Loss

- N experts; T tokens in a batch $\mathfrak{B}$
- f_i: Fraction of tokens dispatched to expert i

Content credits: Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
https://www.youtube.com/watch?v=U8J32Z3qV8s&t=2816s

Load Balancing Loss

- N experts; T tokens in a batch $\mathfrak{B}$
- f_i: Fraction of tokens dispatched to expert i

$$f_i = \frac{1}{T} \sum_{x \in \mathcal{B}} \mathbb{1}\{\text{argmax}\, p(x) = i\}$$

Content credits: Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
https://www.youtube.com/watch?v=U8J32Z3qV8s&t=2816s

Load Balancing Loss

- N experts; T tokens in a batch $\mathfrak{B}$
- f_i: Fraction of tokens dispatched to expert i

$$f_i = \frac{1}{T} \sum_{x \in \mathcal{B}} \mathbb{1}\{\text{argmax}\, p(x) = i\}$$

- P_i: Expected Probability of selecting expert i

Content credits: Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
https://www.youtube.com/watch?v=U8J32Z3qV8s&t=2816s

Load Balancing Loss

- N experts; T tokens in a batch $\mathfrak{B}$
- f_i: Fraction of tokens dispatched to expert i

$$f_i = \frac{1}{T} \sum_{x \in \mathcal{B}} \mathbb{1}\{\operatorname{argmax} p(x) = i\}$$

- P_i: Expected Probability of selecting expert i

$$P_i = \frac{1}{T} \sum_{x \in \mathcal{B}} p_i(x).$$

Content credits: Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
https://www.youtube.com/watch?v=U8J32Z3qV8s&t=2816s

Load Balancing Loss

- N experts; T tokens in a batch $\mathfrak{B}$
- f_i: Fraction of tokens dispatched to expert i

$$f_i = \frac{1}{T} \sum_{x \in \mathcal{B}} \mathbb{1}\{\text{argmax}\, p(x) = i\}$$

- P_i: Expected Probability of selecting expert i

$$P_i = \frac{1}{T} \sum_{x \in \mathcal{B}} p_i(x).$$

Using sample mean as an empirical estimate

Content credits: Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
https://www.youtube.com/watch?v=U8J32Z3qV8s&t=2816s

Load Balancing Loss

- N experts; T tokens in a batch $\mathfrak{B}$
- f_i: Fraction of tokens dispatched to expert i

$$f_i = \frac{1}{T} \sum_{x \in \mathcal{B}} \mathbb{1}\{\text{argmax}\, p(x) = i\}$$

- P_i: Expected Probability of selecting expert i

$$P_i = \frac{1}{T} \sum_{x \in \mathcal{B}} p_i(x).$$

$$\boxed{\text{loss} = \alpha \cdot N \cdot \sum_{i=1}^{N} f_i \cdot P_i}$$

Content credits: Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
https://www.youtube.com/watch?v=U8J32Z3qV8s&t=2816s

Load Balancing Loss

- N experts; T tokens in a batch $\mathfrak{B}$
- f_i: Fraction of tokens dispatched to expert i

$$f_i = \frac{1}{T} \sum_{x \in \mathcal{B}} \mathbb{1}\{\operatorname{argmax} p(x) = i\}$$

- P_i: Expected Probability of selecting expert i

$$P_i = \frac{1}{T} \sum_{x \in \mathcal{B}} p_i(x).$$

$$\boxed{\text{loss} = \alpha \cdot N \cdot \sum_{i=1}^{N} f_i \cdot P_i}$$

👍 Prevents router collapse

👍 Improves training efficiency by using all the devices equally (remember that each expert is on a separate device)

Content credits: Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
https://www.youtube.com/watch?v=U8J32Z3qV8s&t=2816s

Switch Transformer Layer

- **Issues Addressed:**
  - Complexity of MoE
  - Communication cost
  - Training Instability

Top-1 greedy routing: Challenged the belief that we need to route to at least 2 experts for meaningful learning of router

Improved Training Techniques:

1. Differentiable load balancing loss (avoids router collapse)
2. Selective Precision

Content credits: Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
https://www.youtube.com/watch?v=U8J32Z3qV8s&t=2816s

Selective Precision

- Training in `bfloat16`:
    - 👍 Reduces communication cost

Content credits: Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
https://www.youtube.com/watch?v=U8J32Z3qV8s&t=2816s

Selective Precision

- Training in `bfloat16`:
  - 👍 Reduces communication cost
  - 👎 Increases instability - common practice is to use optimizer in `float32`

Content credits: Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
https://www.youtube.com/watch?v=U8J32Z3qV8s&t=2816s

Selective Precision

- Training in `bfloat16`:
  - 👍 Reduces communication cost
  - 👎 Increases instability - common practice is to use optimizer in `float32`
  - 💡 Cast router to `float32` - because exp. is sensitive to small errors

Model (precision)	Quality (Neg. Log Perp.) (↑)	Speed (Examples/sec) (↑)
Switch-Base (float32)	-1.718	1160
Switch-Base (bfloat16)	-3.780 [*diverged*]	**1390**
Switch-Base (Selective precision)	**-1.716**	1390

Content credits: Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
https://www.youtube.com/watch?v=U8J32Z3qV8s&t=2816s

Switch Transformer Layer

- **Issues Addressed:**
  - Complexity of MoE
  - Communication cost
  - Training Instability

Top-1 greedy routing: Challenged the belief that we need to route to at least 2 experts for meaningful learning of router

Improved Training Techniques:

1. Differentiable load balancing loss (avoids router collapse)
2. Selective Precision
3. Reduced initialization scale

Content credits: Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
https://www.youtube.com/watch?v=U8J32Z3qV8s&t=2816s

Smaller parameter initialization for stability

- Default initialization: $\mu = 0;\ \sigma = \sqrt{1/d}$; resample if beyond 2σ

Content credits: Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
https://www.youtube.com/watch?v=U8J32Z3qV8s&t=2816s

Smaller parameter initialization for stability

- Default initialization: $\mu = 0;\ \sigma = \sqrt{1/d}$; resample if beyond 2σ
- Recommended initialization: $\mu = 0;\ \sigma = \sqrt{\mathbf{0.1}/d}$; resample if beyond 2σ

Content credits: Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
https://www.youtube.com/watch?v=U8J32Z3qV8s&t=2816s

Smaller parameter initialization for stability

- Default initialization: $\mu = 0;\ \sigma = \sqrt{1/d}$; resample if beyond 2σ
- Recommended initialization: $\mu = 0;\ \sigma = \sqrt{\mathbf{0.1}/d}$; resample if beyond 2σ

Model (Initialization scale)	Average Quality (Neg. Log Perp.)	Std. Dev. of Quality (Neg. Log Perp.)
Switch-Base (0.1x-init)	**-2.72**	**0.01**
Switch-Base (1.0x-init)	-3.60	0.68

Performance of 32 expert model after 3.5k steps (3 random seeds)

Content credits: Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
https://www.youtube.com/watch?v=U8J32Z3qV8s&t=2816s

Switch Transformer Layer

- **Issues Addressed:**
  - Complexity of MoE
  - Communication cost
  - Training Instability

Top-1 greedy routing: Challenged the belief that we need to route to at least 2 experts for meaningful learning of router

Improved Training Techniques:

1. Differentiable load balancing loss (avoids router collapse)
2. Selective Precision
3. Reduced initialization scale
4. Higher regularization of experts

Content credits: Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
https://www.youtube.com/watch?v=U8J32Z3qV8s&t=2816s

Higher regularization for Experts during fine-tuning

- Pretrain and then finetune on downstream tasks
  - 👎 MoEs prone to overfitting due to high parameter count

Higher regularization for Experts during fine-tuning

- Pretrain and then finetune on downstream tasks
  - 👎 MoEs prone to overfitting due to high parameter count
  - 💡 Increase expert dropout for increased regularization

Higher regularization for Experts during fine-tuning

- Pretrain and then finetune on downstream tasks
  - 👎 MoEs prone to overfitting due to high parameter count
  - 💡 Increase expert dropout for increased regularization

Model (dropout)	GLUE	CNNDM	SQuAD	SuperGLUE
T5-Base (d=0.1)	82.9	**19.6**	83.5	72.4
Switch-Base (d=0.1)	84.7	19.1	**83.7**	**73.0**
Switch-Base (d=0.2)	84.4	19.2	**83.9**	**73.2**
Switch-Base (d=0.3)	83.9	19.6	83.4	70.7
Switch-Base (d=0.1, ed=0.4)	**85.2**	**19.6**	**83.7**	**73.0**

- Pretrained on 34B tokens; Uniform dropout performs worse;
- Low dropout for non-experts and high dropout for expert layers perform the best

Switch Transformer Layer

- **Issues Addressed:**
  - Complexity of MoE
  - Communication cost
  - Training Instability

Top-1 greedy routing: Challenged the belief that we need to route to at least 2 experts for meaningful learning of router

Improved Training Techniques:
1. Differentiable load balancing loss (avoids router collapse)
2. Selective Precision
3. Reduced initialization scale
4. Slower learning rate warmup
5. Higher regularization of experts

Content credits: Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
https://www.youtube.com/watch?v=U8J32Z3qV8s&t=2816s

Distributed Switch Implementation

- Trained on TPUs using Mesh-Tensorflow
    - 👍 Facilitates efficient model-parallel architectures (*i.e.* experts on different cores)

Distributed Switch Implementation

- Trained on TPUs using Mesh-Tensorflow
  - 👍 Facilitates efficient model-parallel architectures (*i.e.* experts on different cores)
  - 👎 Statically compiled computational graph – fixed tensor shapes but dynamic computation

Distributed Switch Implementation

- Trained on TPUs using Mesh-Tensorflow
  - 👍 Facilitates efficient model-parallel architectures (*i.e.* experts on different cores)
  - 👎 Statically compiled computational graph – fixed tensor shapes but dynamic computation

How to set Expert Capacity?

(*Number of tokens processed by each expert*)

Distributed Switch Implementation

- Trained on TPUs using Mesh-Tensorflow
  - 👍 Facilitates efficient model-parallel architectures (*i.e.* experts on different cores)
  - 👎 Statically compiled computational graph – fixed tensor shapes but dynamic computation

How to set Expert Capacity?

(*Number of tokens processed by each expert*)

$$\text{expert capacity} = \left(\frac{\text{tokens per batch}}{\text{number of experts}} \right)$$

Distributed Switch Implementation

- Trained on TPUs using Mesh-Tensorflow
  - 👍 Facilitates efficient model-parallel architectures (*i.e.* experts on different cores)
  - 👎 Statically compiled computational graph – fixed tensor shapes but dynamic computation

How to set Expert Capacity?

(*Number of tokens processed by each expert*)

$$\text{expert capacity} = \left(\frac{\text{tokens per batch}}{\text{number of experts}}\right)$$

Uniform distribution of tokens to all experts

Distributed Switch Implementation

- Trained on TPUs using Mesh-Tensorflow

  👍 Facilitates efficient model-parallel architectures (*i.e.* experts on different cores)

  👎 Statically compiled computational graph – fixed tensor shapes but dynamic computation

How to set Expert Capacity?
(*Number of tokens processed by each expert*)

$$\text{expert capacity} = \left(\frac{\text{tokens per batch}}{\text{number of experts}}\right) \times \text{capacity factor}.$$

Uniform distribution of tokens to all experts

Distributed Switch Implementation

- Trained on TPUs using Mesh-Tensorflow
  - 👍 Facilitates efficient model-parallel architectures (*i.e.* experts on different cores)
  - 👎 Statically compiled computational graph – fixed tensor shapes but dynamic computation

How to set Expert Capacity?

(*Number of tokens processed by each expert*)

$$\text{expert capacity} = \left(\frac{\text{tokens per batch}}{\text{number of experts}}\right) \times \text{capacity factor}.$$

Uniform distribution of tokens to all experts

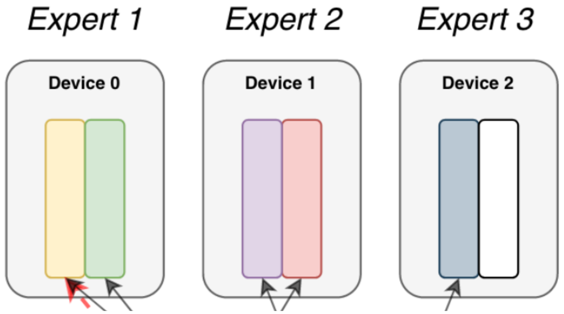
Buffer for skewed distribution while training

Modulating Expert Capacity via Capacity Factor

Terminology

- **Experts:** Split across devices, each having their own unique parameters. Perform standard feed-forward computation.

- **Expert Capacity:** Batch size of each expert. Calculated as
- (tokens_per_batch / num_experts) * capacity_factor



Modulating Expert Capacity via Capacity Factor

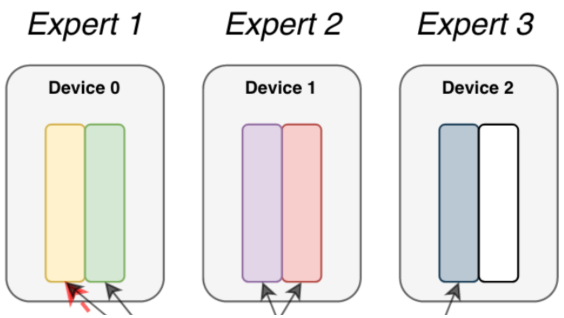
Terminology

- **Experts:** Split across devices, each having their own unique parameters. Perform standard feed-forward computation.

- **Expert Capacity:** Batch size of each expert. Calculated as
- (tokens_per_batch / num_experts) * capacity_factor

- **Capacity Factor:** Used when calculating expert capacity. Expert capacity allows more buffer to help mitigate token overflow during routing.

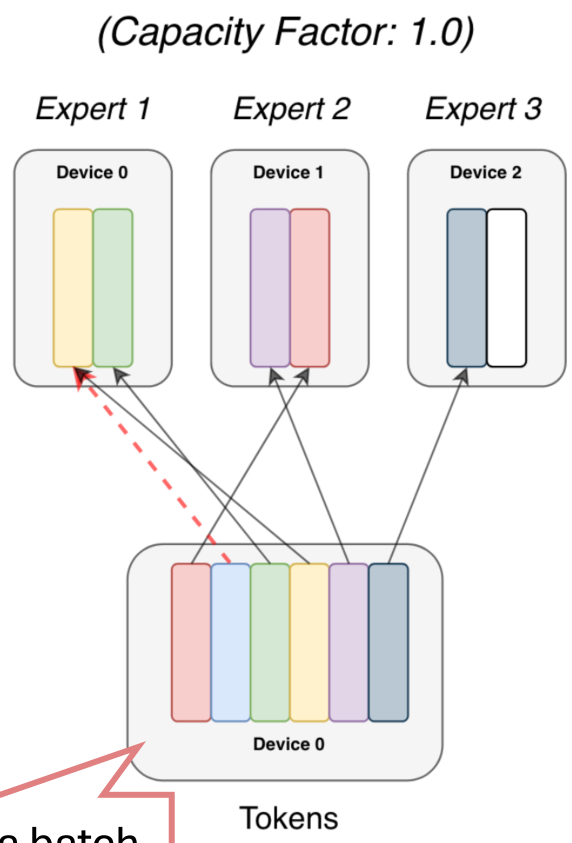
(Capacity Factor: 1.0)



Modulating Expert Capacity via Capacity Factor

Terminology

- **Experts:** Split across devices, each having their own unique parameters. Perform standard feed-forward computation.
- **Expert Capacity:** Batch size of each expert. Calculated as
- (tokens_per_batch / num_experts) * capacity_factor
- **Capacity Factor:** Used when calculating expert capacity. Expert capacity allows more buffer to help mitigate token overflow during routing.

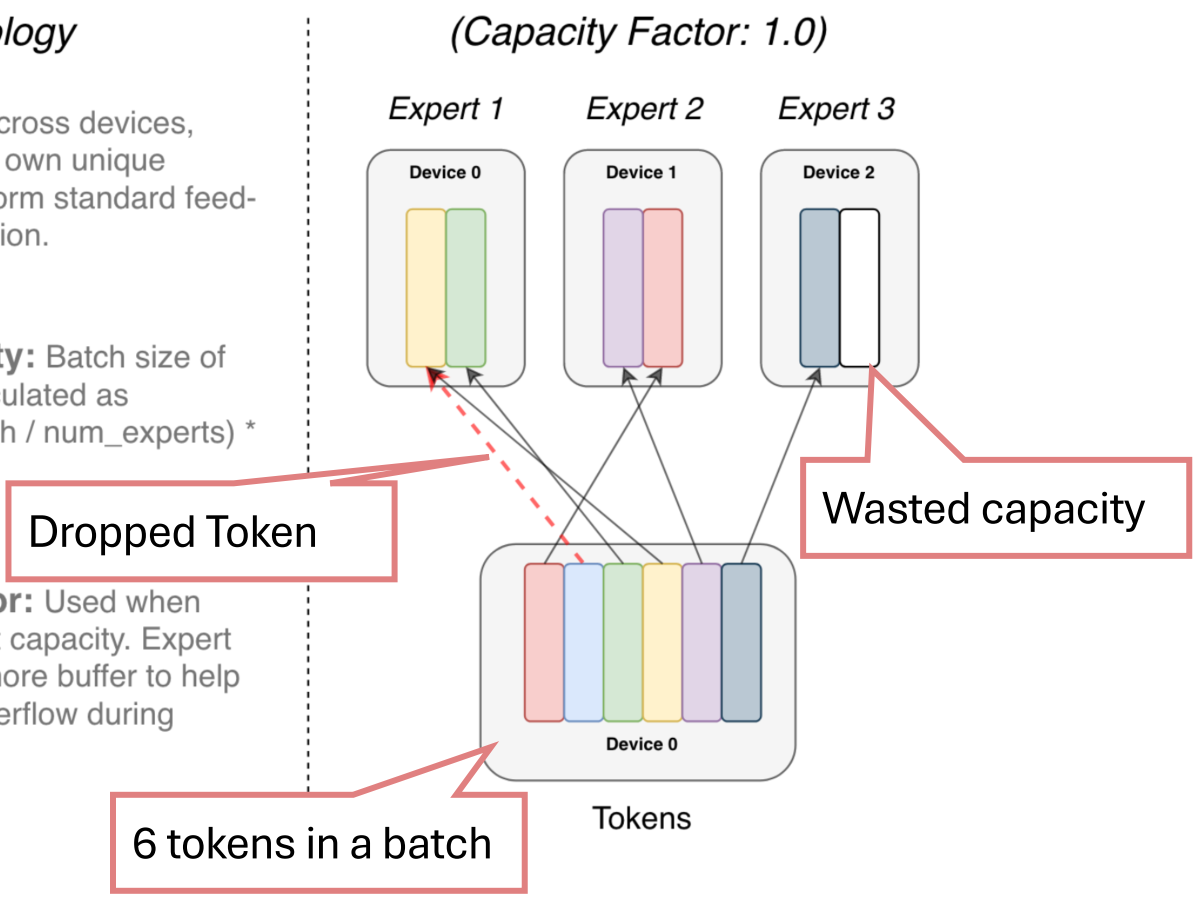


6 tokens in a batch

Modulating Expert Capacity via Capacity Factor

Terminology

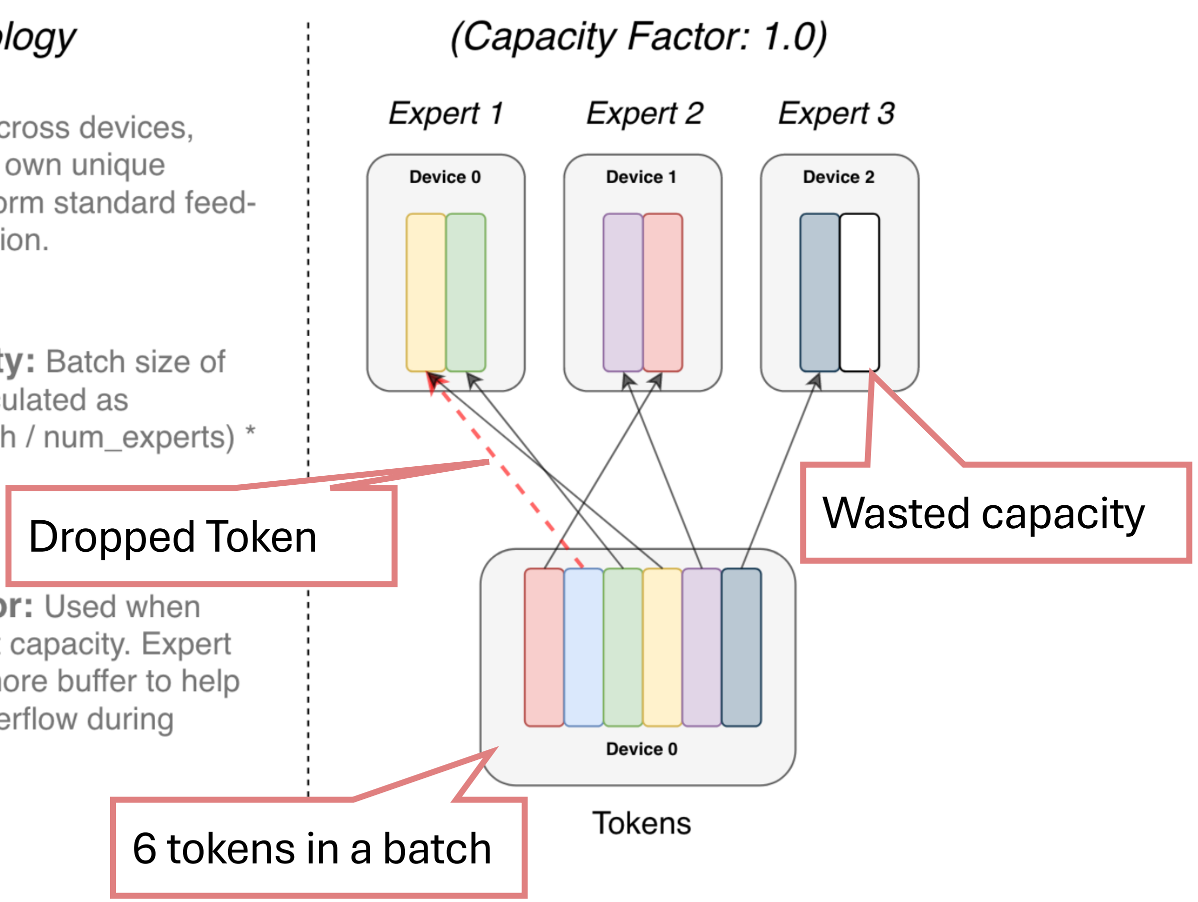
- **Experts:** Split across devices, each having their own unique parameters. Perform standard feed-forward computation.
- **Expert Capacity:** Batch size of each expert. Calculated as
- (tokens_per_batch / num_experts) * capacity_factor
- **Capacity Factor:** Used when calculating expert capacity. Expert capacity allows more buffer to help mitigate token overflow during routing.



Modulating Expert Capacity via Capacity Factor

Terminology

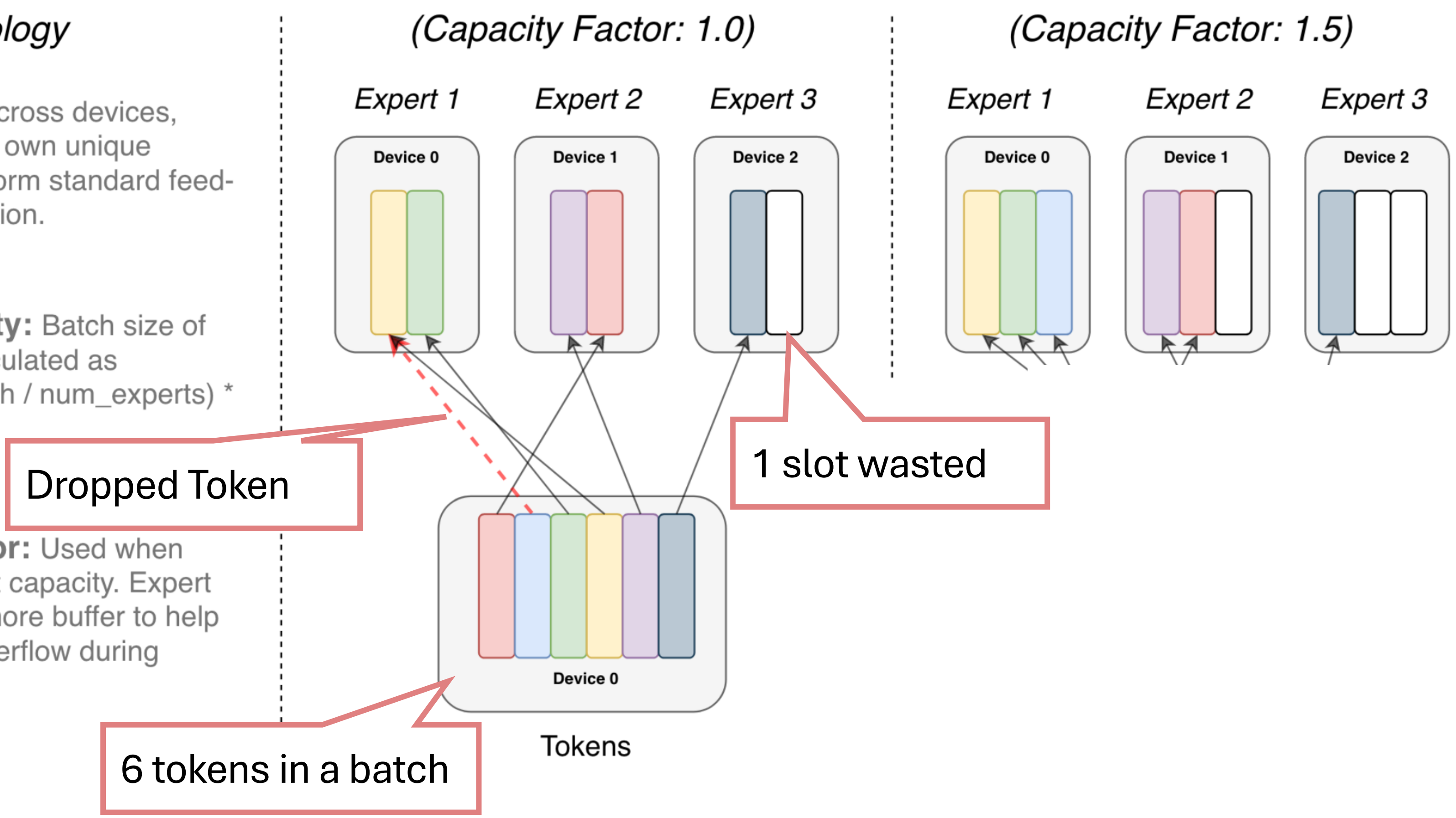
- **Experts:** Split across devices, each having their own unique parameters. Perform standard feed-forward computation.
- **Expert Capacity:** Batch size of each expert. Calculated as
- (tokens_per_batch / num_experts) * capacity_factor
- **Capacity Factor:** Used when calculating expert capacity. Expert capacity allows more buffer to help mitigate token overflow during routing.



Modulating Expert Capacity via Capacity Factor

Terminology

- **Experts:** Split across devices, each having their own unique parameters. Perform standard feed-forward computation.
- **Expert Capacity:** Batch size of each expert. Calculated as
- (tokens_per_batch / num_experts) * capacity_factor
- **Capacity Factor:** Used when calculating expert capacity. Expert capacity allows more buffer to help mitigate token overflow during routing.



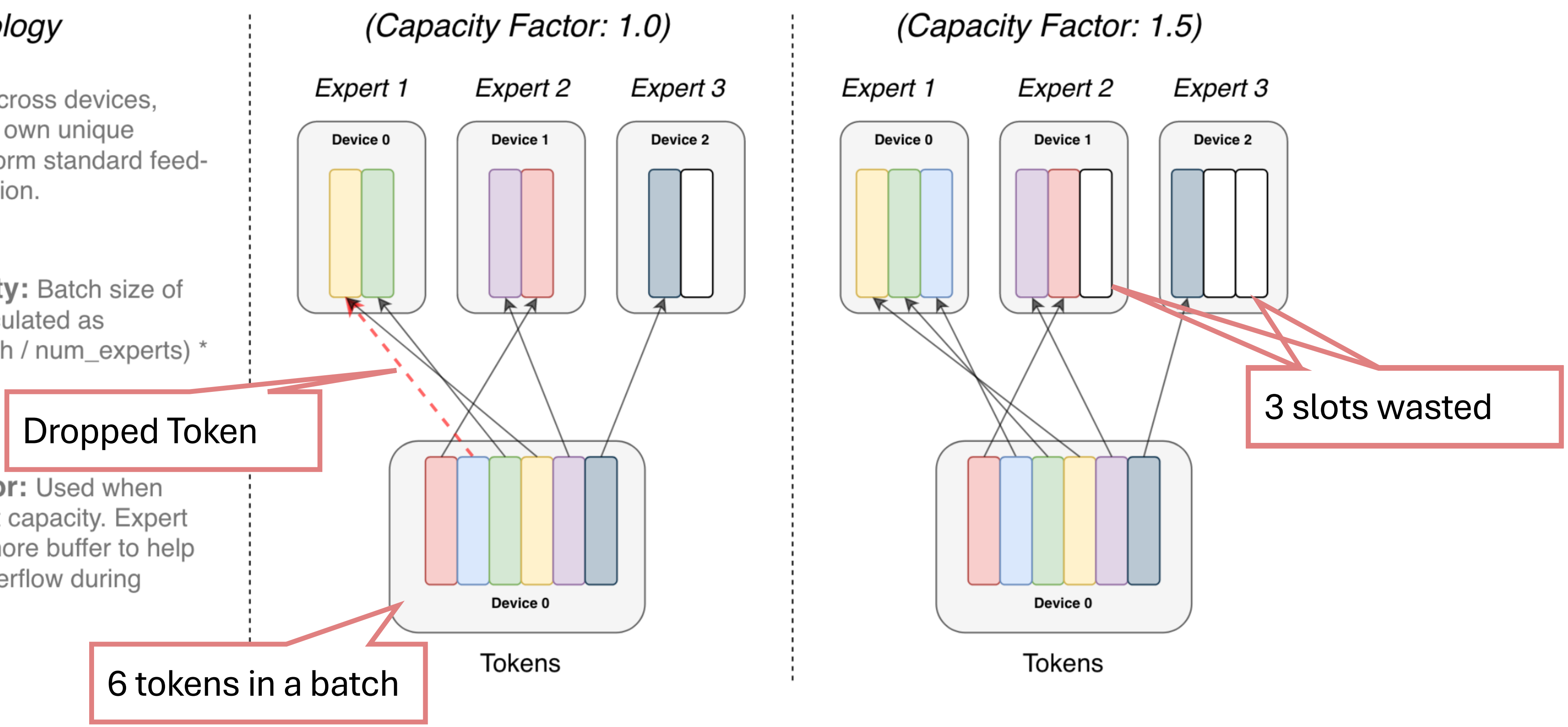
Modulating Expert Capacity via Capacity Factor

Terminology

- **Experts:** Split across devices, each having their own unique parameters. Perform standard feed-forward computation.

- **Expert Capacity:** Batch size of each expert. Calculated as
- (tokens_per_batch / num_experts) * capacity_factor

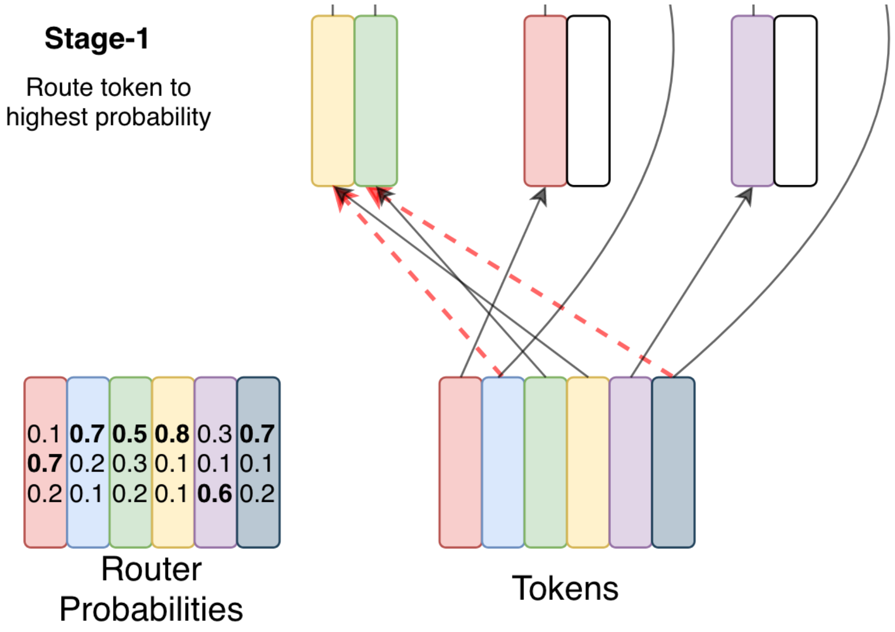
- **Capacity Factor:** Used when calculating expert capacity. Expert capacity allows more buffer to help mitigate token overflow during routing.



No token left behind!

Two stage routing:

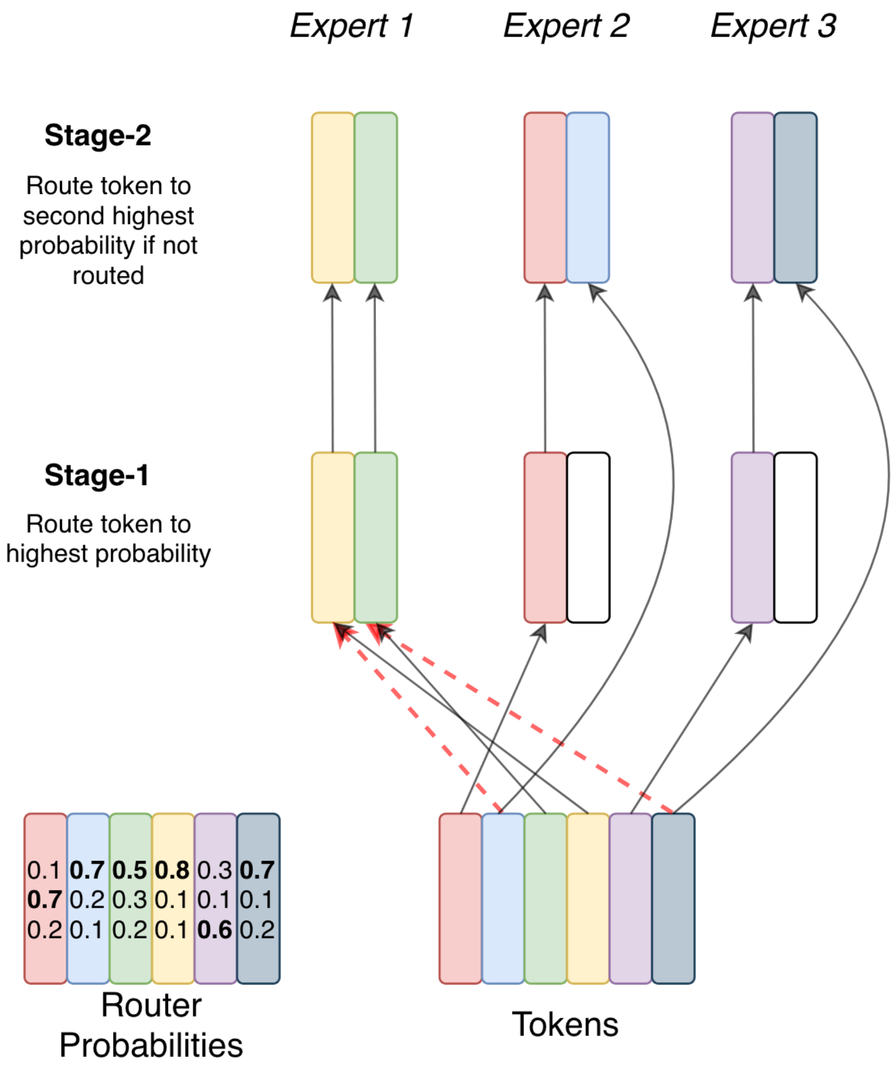
❑ Stage 1: Route to highest probability expert



No token left behind!

Two stage routing:

- ❑ Stage 1: Route to highest probability expert
- ❑ Stage 2: Route the dropped tokens to second best expert

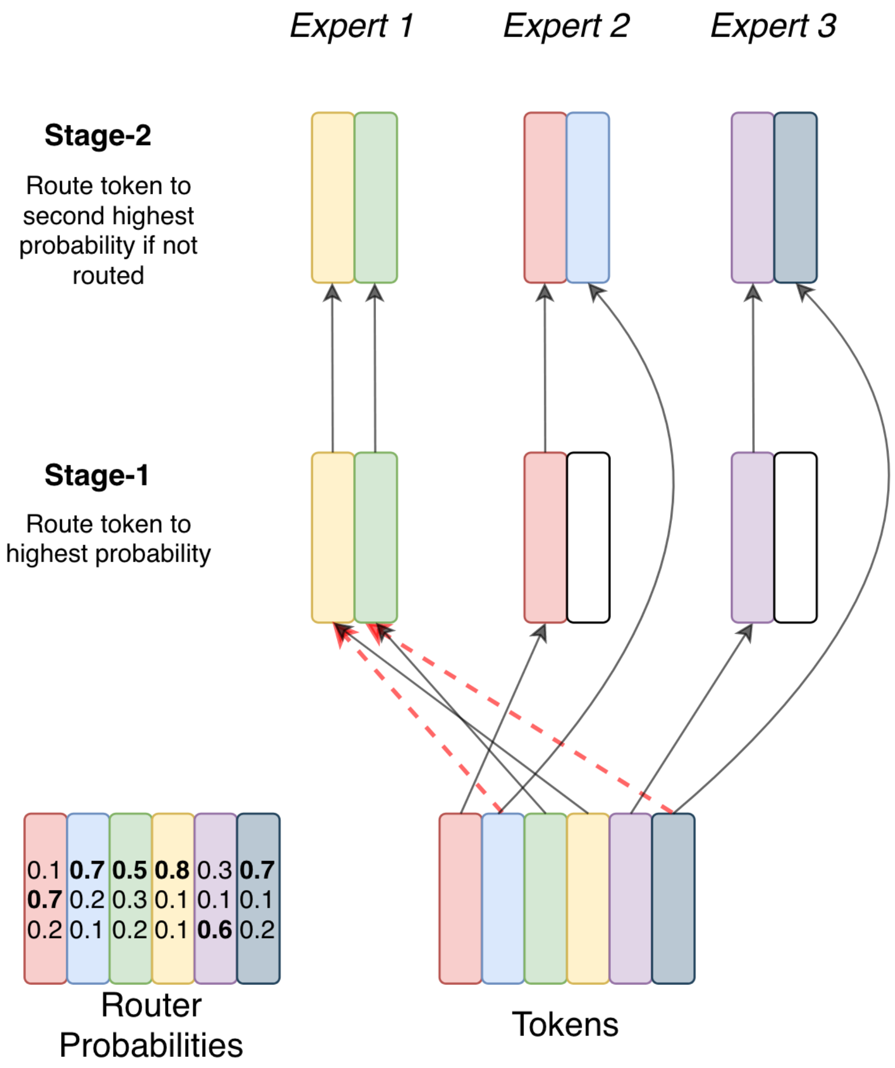


No token left behind!

Two stage routing:

- ❑ Stage 1: Route to highest probability expert
- ❑ Stage 2: Route the dropped tokens to second best expert

Can be iterated till no token left behind!



No token left behind!

Two stage routing:

- ❑ Stage 1: Route to highest probability expert
- ❑ Stage 2: Route the dropped tokens to second best expert

Can be iterated till no token left behind!

- ❖ Doesn’t work empirically!
- ❖ Tokens prefer to be routed to same expert
- ❖ Maybe token dropping introduces regularization

Benchmarking Switch (top-1) versus MoE (noisy top-2)

Time to reach -1.5 Neg. Log Perplexity

Model	Capacity Factor	Quality after 100k steps (↑) (Neg. Log Perp.)	Time to Quality Threshold (↓) (hours)	Speed (↑) (examples/sec)
T5-Base	—	-1.731	Not achieved†	1600
T5-Large	—	-1.550	131.1	470

Benchmarking Switch (top-1) versus MoE (noisy top-2)

Time to reach -1.5 Neg. Log Perplexity

- 128 experts
- Alternate layers

Model	Capacity Factor	Quality after 100k steps (↑) (Neg. Log Perp.)	Time to Quality Threshold (↓) (hours)	Speed (↑) (examples/sec)
T5-Base	—	-1.731	Not achieved†	1600
T5-Large	—	-1.550	131.1	470
MoE-Base	2.0	-1.547	68.7	840
Switch-Base	2.0	-1.554	72.8	860

Benchmarking Switch (top-1) versus MoE (noisy top-2)

Time to reach -1.5 Neg. Log Perplexity

- 128 experts
- Alternate layers

Model	Capacity Factor	Quality after 100k steps (↑) (Neg. Log Perp.)	Time to Quality Threshold (↓) (hours)	Speed (↑) (examples/sec)
T5-Base	—	-1.731	Not achieved†	1600
T5-Large	—	-1.550	131.1	470
MoE-Base	2.0	-1.547	68.7	840
Switch-Base	2.0	-1.554	72.8	860
MoE-Base	1.25	-1.559	80.7	790
Switch-Base	1.25	-1.553	65.0	910

Benchmarking Switch (top-1) versus MoE (noisy top-2)

Time to reach -1.5 Neg. Log Perplexity

- 128 experts
- Alternate layers

Model	Capacity Factor	Quality after 100k steps (↑) (Neg. Log Perp.)	Time to Quality Threshold (↓) (hours)	Speed (↑) (examples/sec)
T5-Base	—	-1.731	Not achieved†	1600
T5-Large	—	-1.550	131.1	470
MoE-Base	2.0	-1.547	68.7	840
Switch-Base	2.0	-1.554	72.8	860
MoE-Base	1.25	-1.559	80.7	790
Switch-Base	1.25	-1.553	65.0	910
MoE-Base	1.0	-1.572	80.1	860
Switch-Base	1.0	-1.561	**62.8**	1000

Benchmarking Switch (top-1) versus MoE (noisy top-2)

Time to reach -1.5 Neg. Log Perplexity

- 128 experts
- Alternate layers

Increase hidden dim. & no. of heads till it matches speed of top-2 routing

Model	Capacity Factor	Quality after 100k steps (↑) (Neg. Log Perp.)	Time to Quality Threshold (↓) (hours)	Speed (↑) (examples/sec)
T5-Base	—	-1.731	Not achieved†	1600
T5-Large	—	-1.550	131.1	470
MoE-Base	2.0	-1.547	68.7	840
Switch-Base	2.0	-1.554	72.8	860
MoE-Base	1.25	-1.559	80.7	790
Switch-Base	1.25	-1.553	65.0	910
MoE-Base	1.0	-1.572	80.1	860
Switch-Base	1.0	-1.561	**62.8**	1000
Switch-Base+	1.0	**-1.534**	67.6	780

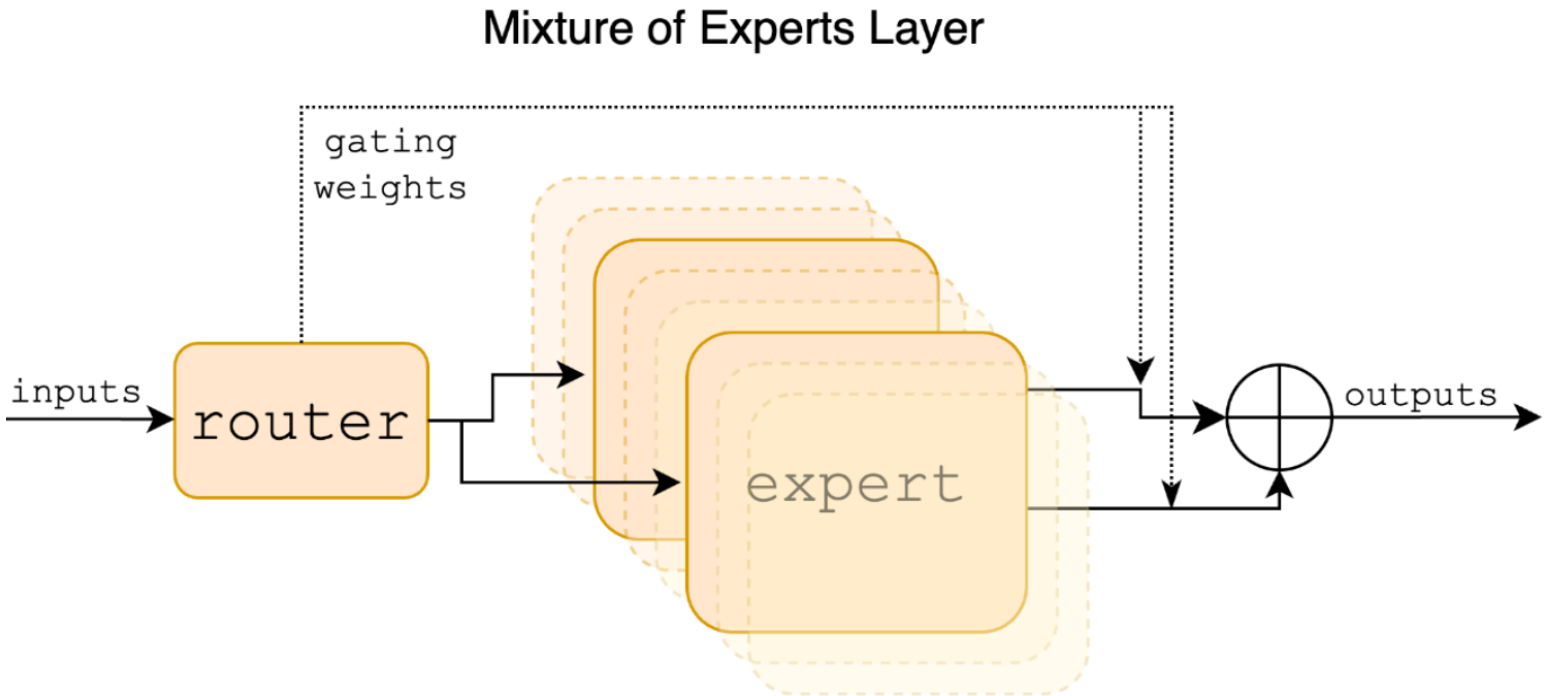
Mixtral of Experts

Abstract

We introduce Mixtral 8x7B, a Sparse Mixture of Experts (SMoE) language model. Mixtral has the same architecture as Mistral 7B, with the difference that each layer is composed of 8 feedforward blocks (i.e. experts). For every token, at each layer, a router network selects two experts to process the current state and combine their outputs. Even though each token only sees two experts, the selected experts can be different at each timestep. As a result, each token has access to 47B parameters, but only uses 13B active parameters during inference. Mixtral was trained with a context size of 32k tokens and it outperforms or matches Llama 2 70B and GPT-3.5 across all evaluated benchmarks. In particular, Mixtral vastly outperforms Llama 2 70B on mathematics, code generation, and multilingual benchmarks. We also provide a model fine-

Mixture of Experts: 8x7B

Parameter	Value
dim	4096
n_layers	32
head_dim	128
hidden_dim	14336
n_heads	32
n_kv_heads	8
context_len	32768
vocab_size	32000
num_experts	8
top_k_experts	2

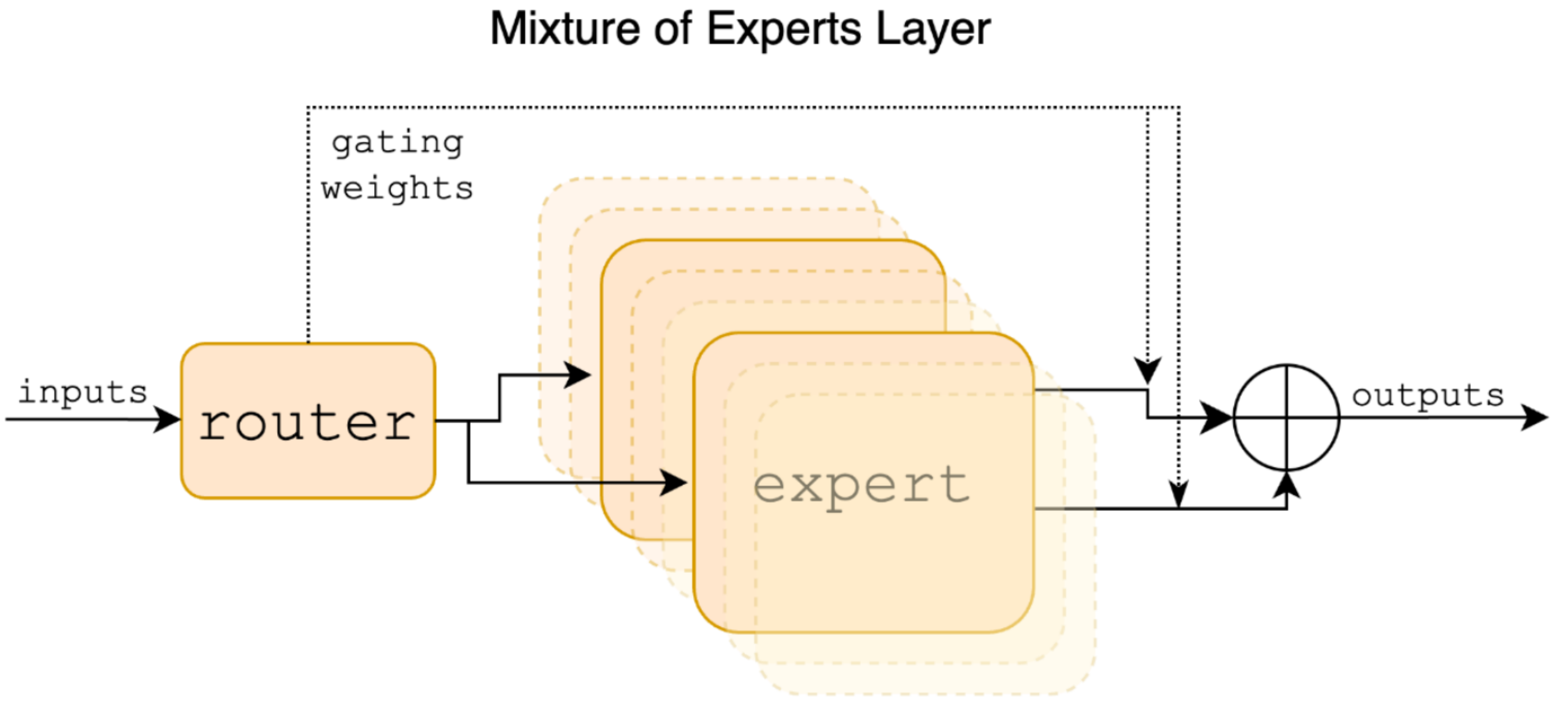


Replace FFN with MoE in all layers; unlike Switch Transformers

Mixture of Experts: 8x7B

$$G(x) := \text{Softmax}(\text{TopK}(x \cdot W_g)),$$

$$\sum_{i=0}^{n-1} G(x)_i \cdot E_i(x)$$



Mixture of Experts: 8x7B

$$G(x) := \text{Softmax}(\text{TopK}(x \cdot W_g))$$

$$y = \sum_{i=0}^{n-1} \underbrace{\text{Softmax}(\text{Top2}(x \cdot W_g))_i \cdot \text{SwiGLU}_i(x)}_{E_i(x)}$$

Mixture of Experts: 8x7B

$$G(x) := \text{Softmax}(\text{TopK}(x \cdot W_g))$$

$$y = \underbrace{\sum_{i=0}^{n-1} \text{Softmax}(\text{Top2}(x \cdot W_g))_i \cdot \text{SwiGLU}_i(x)}_{E_i(x)}$$

Combines Swish Activation with Gated Linear Unit (GLU)

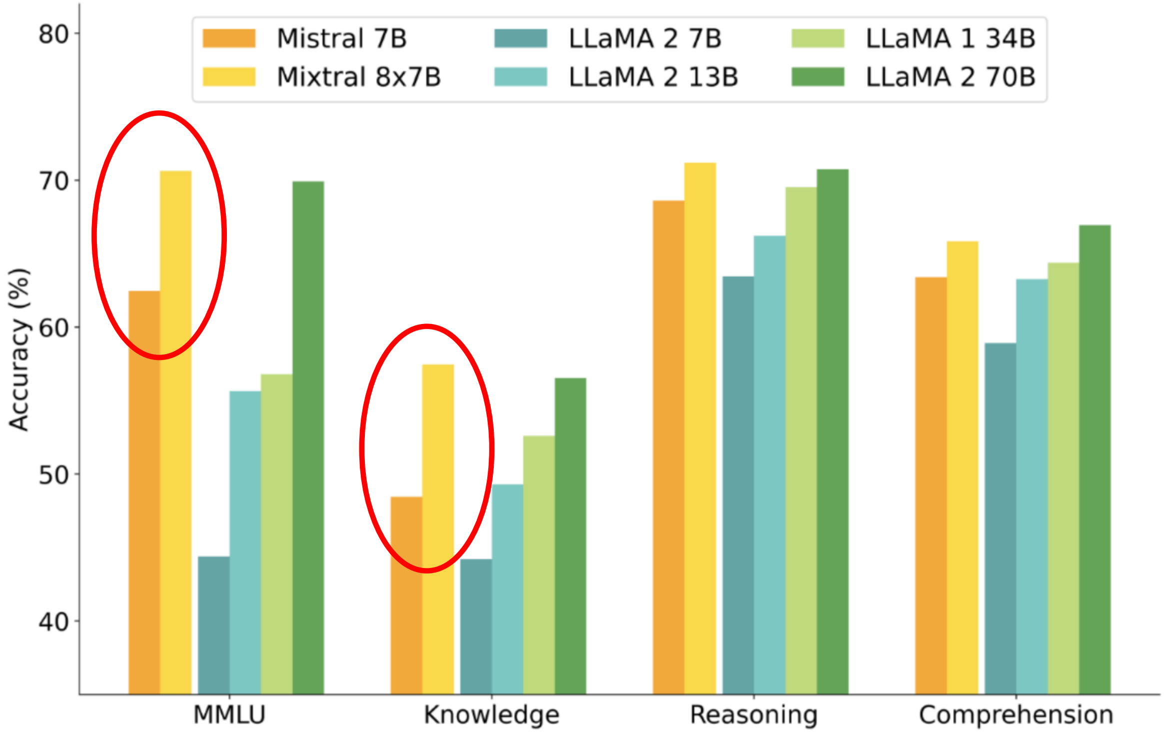
```
SwiGLU(x) = x * sigmoid(beta * x) + (1 - sigmoid(beta * x)) * (Wx + b)
```

Reasoning vs knowledge intensive tasks

- FFN layers account for knowledge
- Attention layers account for reasoning or algorithms

Content Credit: https://www.youtube.com/watch?v=RcJ1YXHLv5o&t=2835s

Reasoning vs knowledge intensive tasks



Knowledge intensive tasks

- Huge gap b/w dense and corresponding sparse models on knowledge intensive tasks

Content Credit: https://www.youtube.com/watch?v=RcJ1YXHLv5o&t=2835s

Interpreting routing decisions

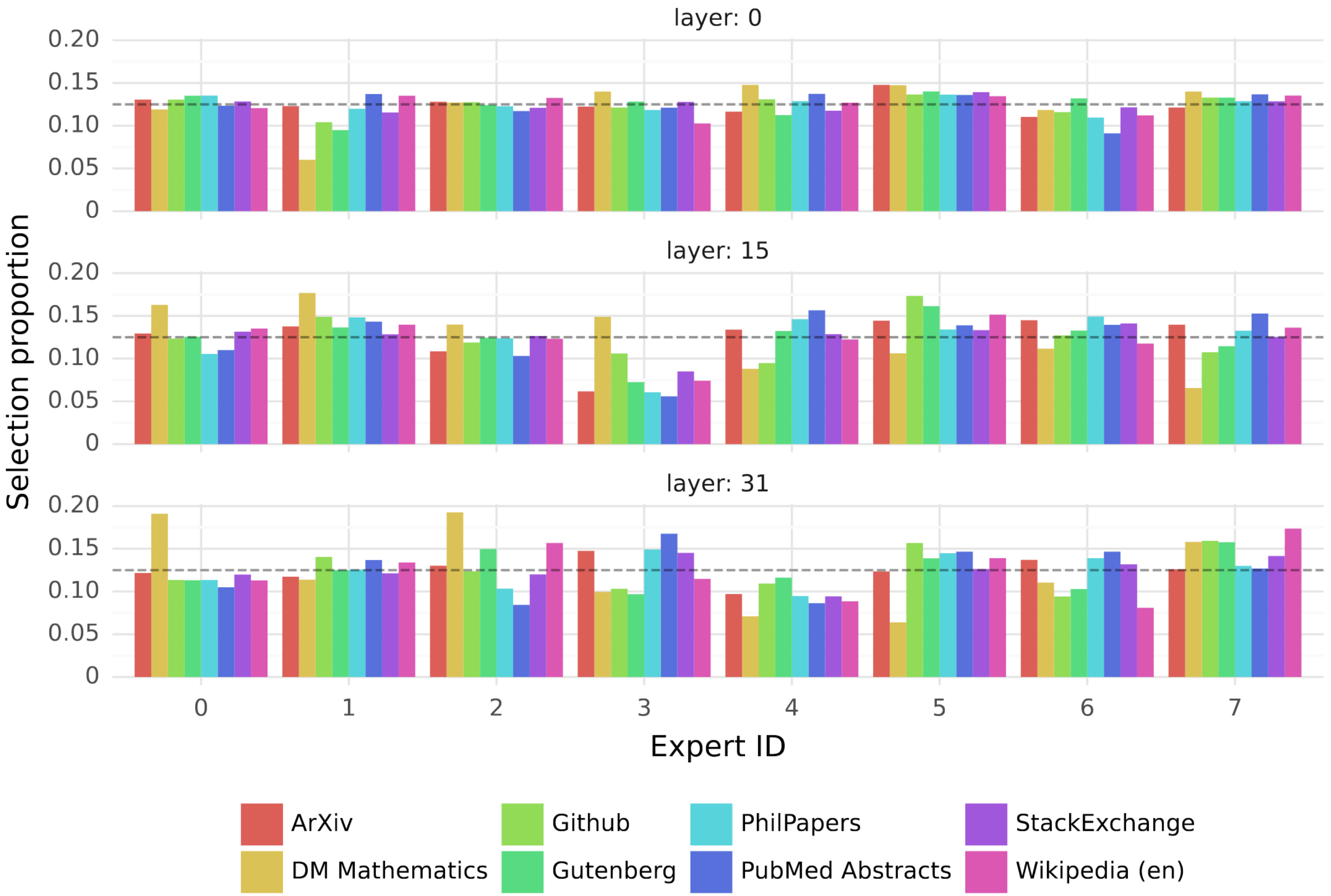
- Self-attention is often used as an interpretation tool-
  - Which token in the input are we attending to while generating the next token?

- Can we use routing decisions for interpreting the model?
  - Which tokens are routed to a particular expert?

Content Credit: https://www.youtube.com/watch?v=RcJ1YXHLv5o&t=2835s

Interpreting routing decisions

- Validation split of Pile Dataset
- Proportion of tokens assigned to each expert on different domains
- Done for Layer 0, layer 15, and layer 31



Content Credit: https://www.youtube.com/watch?v=RcJ1YXHLv5o&t=2835s

Routing of Consecutive Tokens

- How many times two consecutive tokens are routed to the same expert?

Content Credit: https://www.youtube.com/watch?v=RcJ1YXHLv5o&t=2835s

Routing of Consecutive Tokens

- How many times two consecutive tokens are routed to the same expert?

- Repetitions at the first layer are close to random
- Significantly higher at layers 15 and 31.
- The high number of repetitions shows that expert choice exhibits high temporal locality at these layers.

	First choice Layer 0	Layer 15	Layer 31
ArXiv	14.0%	27.9%	22.7%
DM Mathematics	14.1%	28.4%	19.7%
Github	14.9%	28.1%	19.7%
Gutenberg	13.9%	26.1%	26.3%
PhilPapers	13.6%	25.3%	22.1%
PubMed Abstracts	14.2%	24.6%	22.0%
StackExchange	13.6%	27.2%	23.6%
Wikipedia (en)	14.4%	23.6%	25.3%

Content Credit: https://www.youtube.com/watch?v=RcJ1YXHLv5o&t=2835s

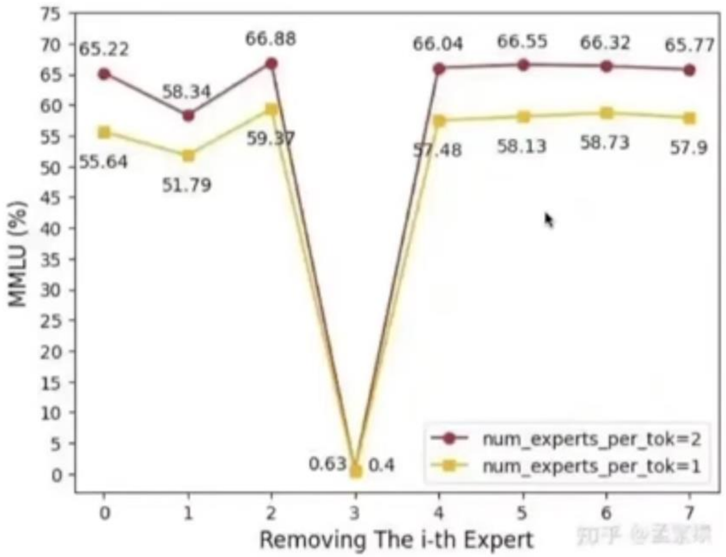
Which experts are active for different tokens?

- Colors represent different experts
- Experts do not specialize in any domain like coding, or maths.

Content Credit: https://www.youtube.com/watch?v=RcJ1YXHLv5o&t=2835s

Interpreting experts

- There is one expert in one of the layers that's particularly crucial.



Content Credit: https://www.youtube.com/watch?v=RcJ1YXHLv5o&t=2835s

Questions