Introduction to Mixture of Experts (MoEs)

Yatin Nandwani
Research Scientist, IBM Research

Large Language Models: Introduction and Recent Advances

ELL881 · AIL821

Semester 1, 2024-2025

IBM Research, India
Conversational-AI

Yatin Nandwani

Sonam Mishra

Dinesh Khandelwal

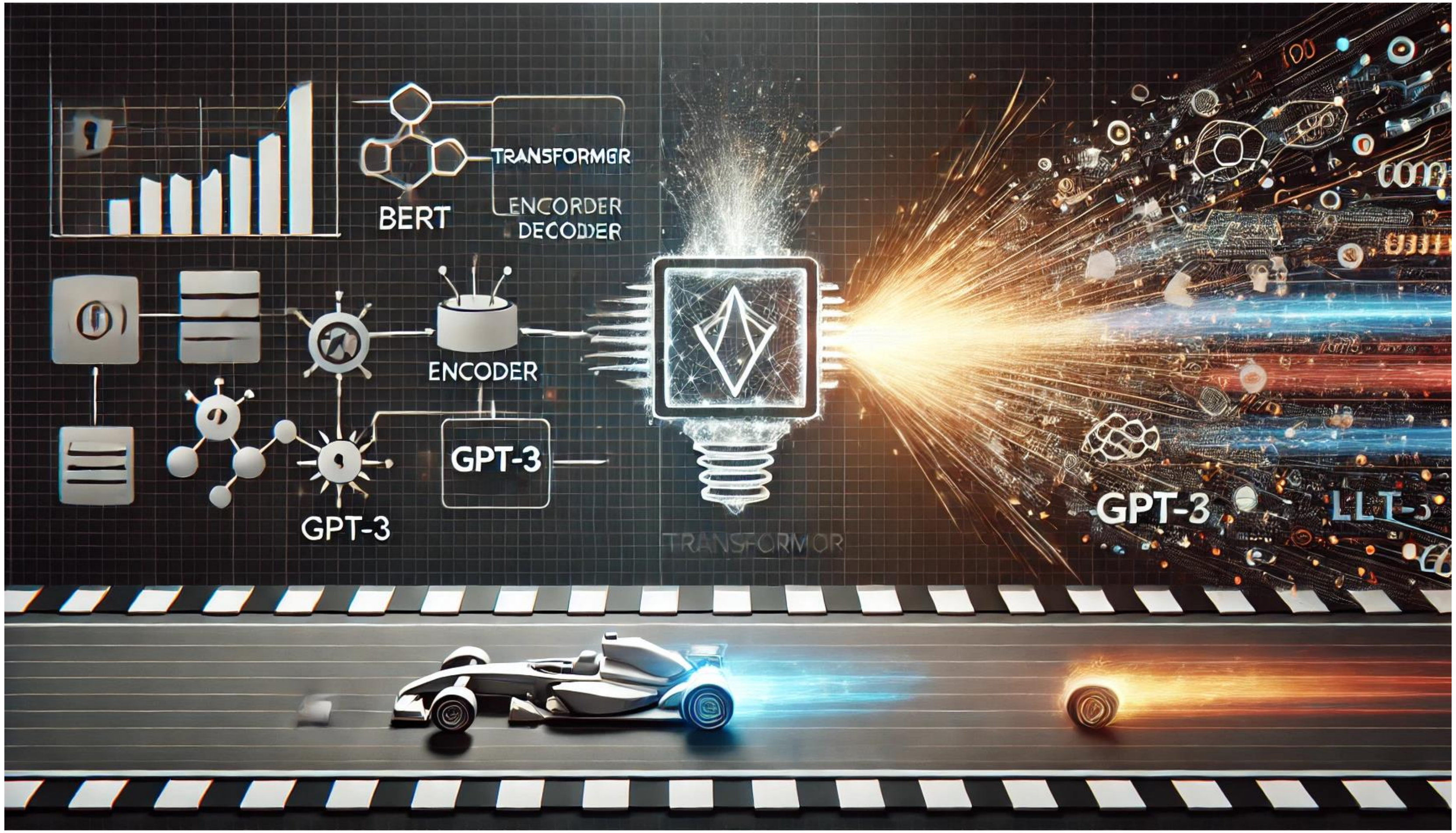
Gaurav Pandey

Vineet Kumar

Dinesh Raghu

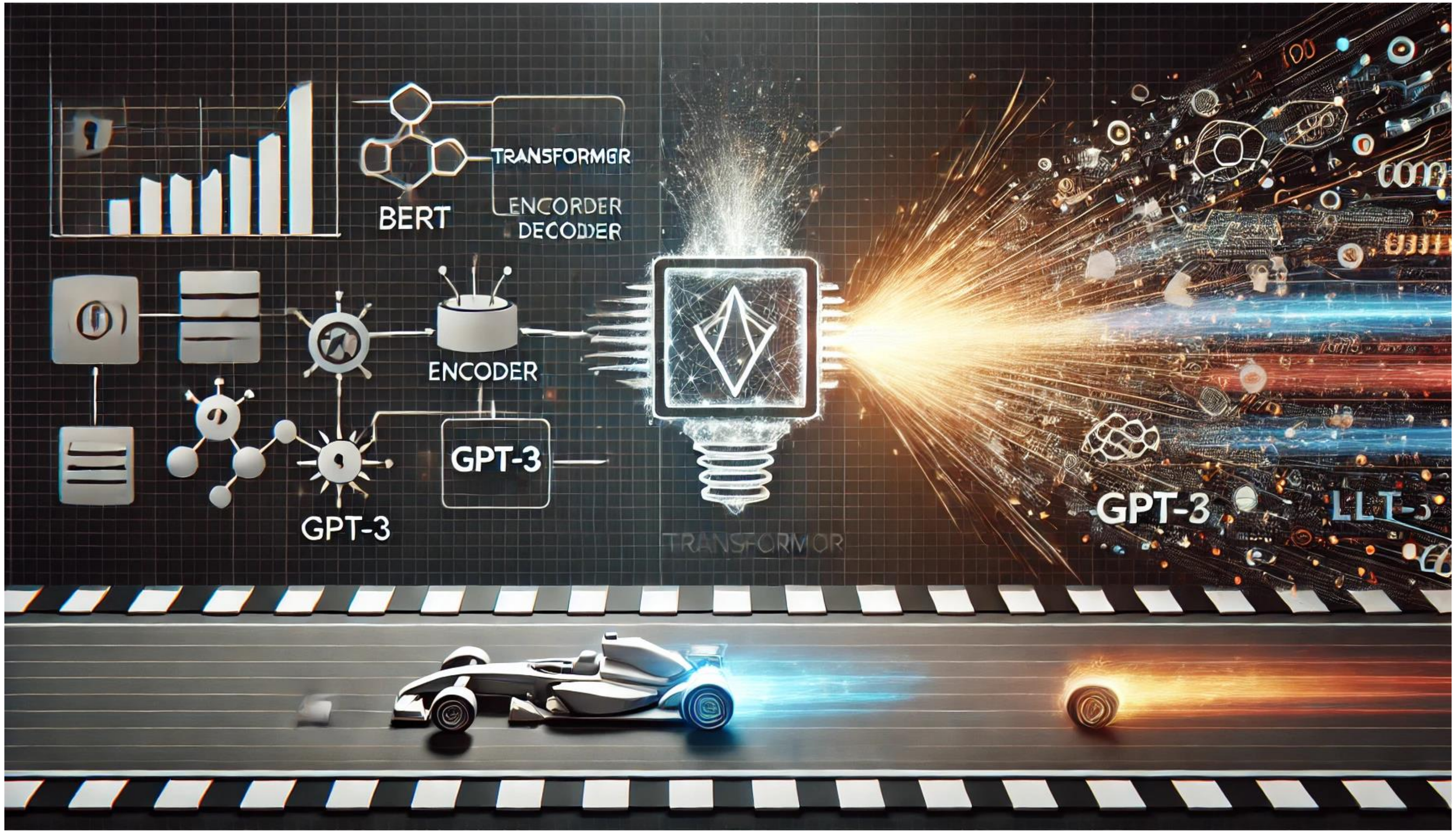
Sachindra Joshi

Large Language Models



Content credits: Chat GPT

Large Language Models

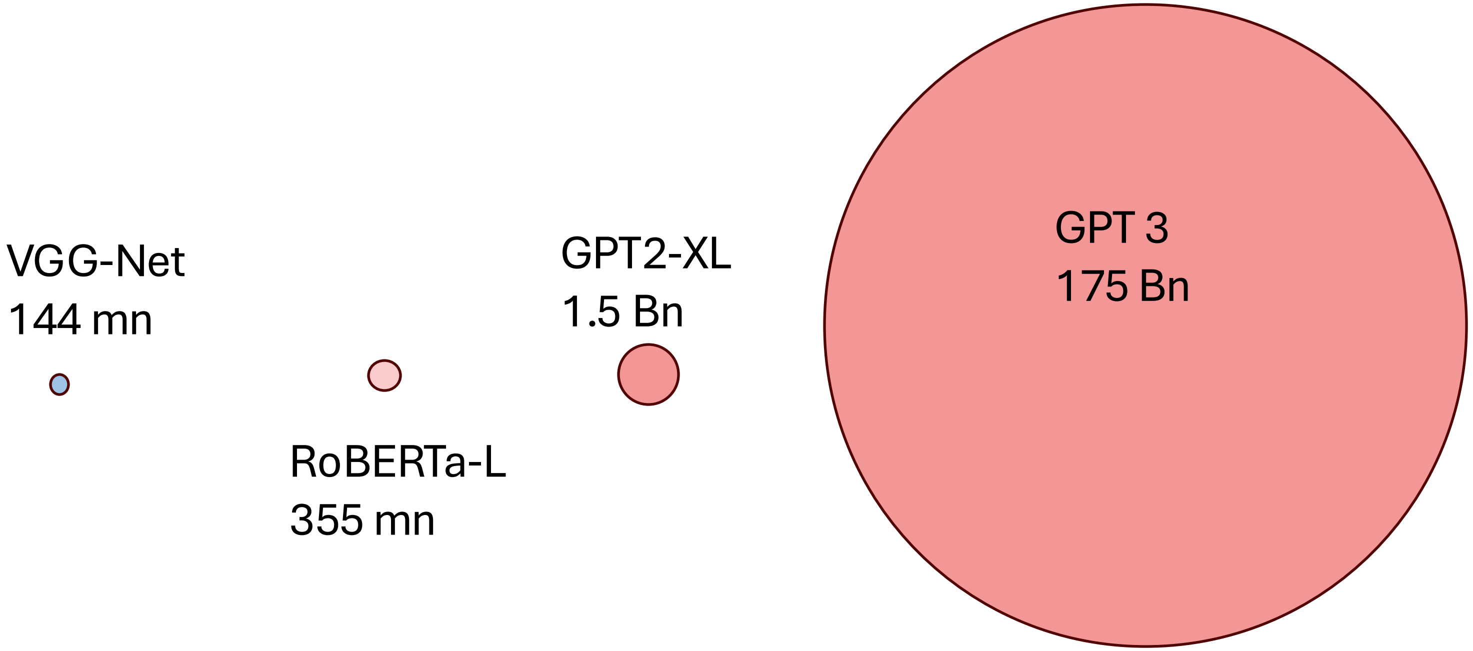


Pretraining
- Pretrain the model on a large dataset.

Finetuning
- Fine-tune of a specific task using a small dataset

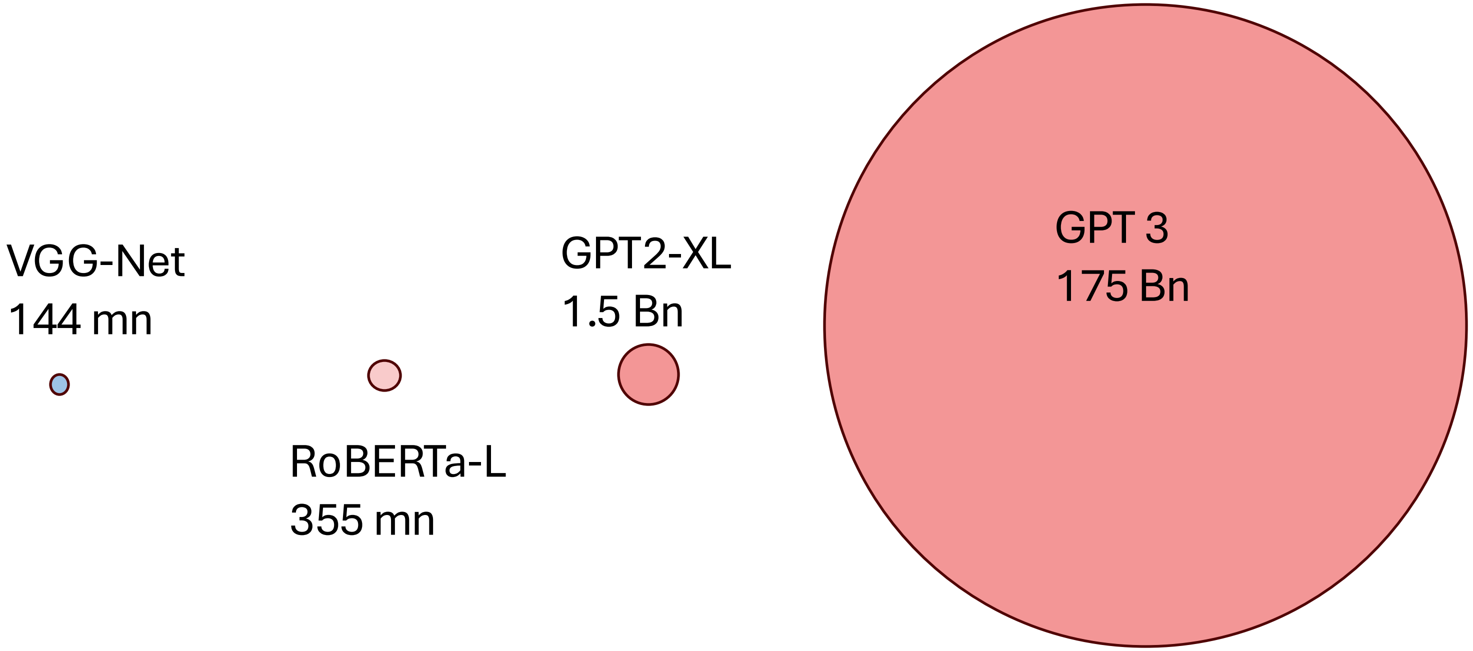
Content credits: Chat GPT

Large Language Models



Content credits: https://www.youtube.com/watch?v=TwHPxUAuqy4

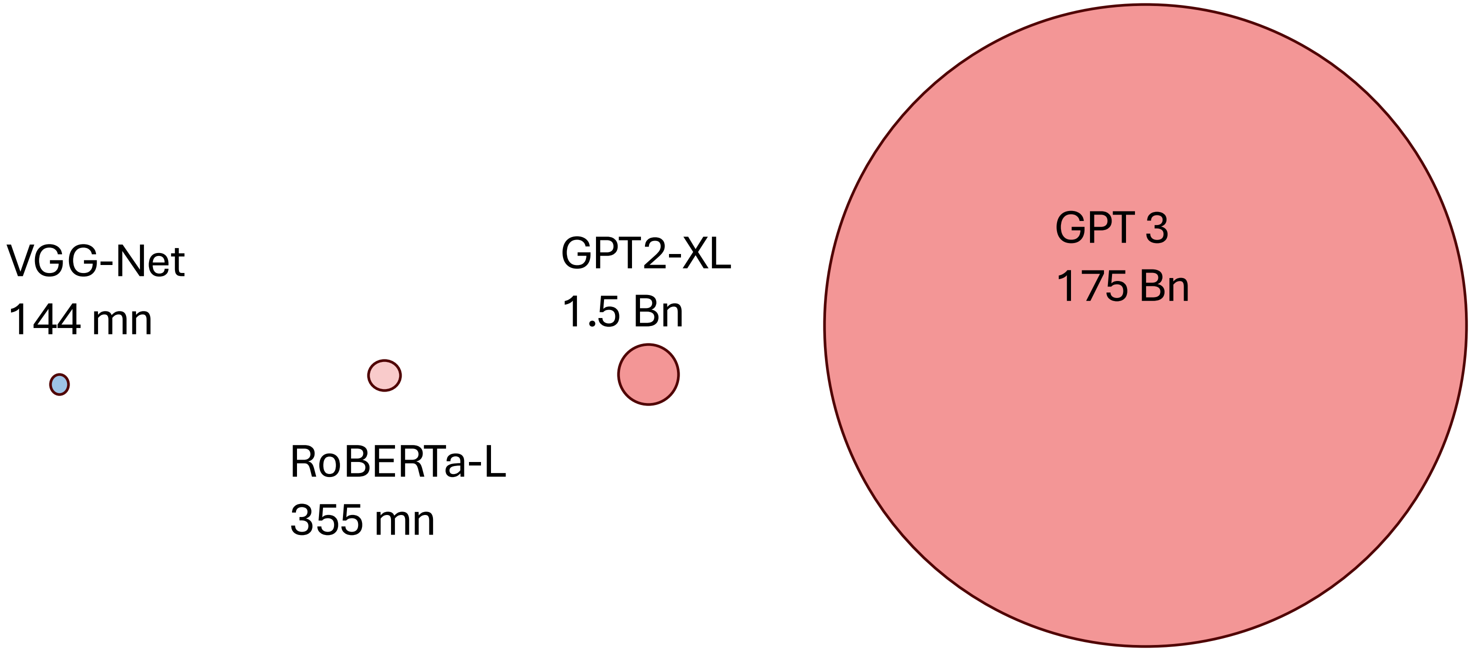
Large Language Models



Model	Size	Approx. Training Time
RoBERTa-L	355 Mn	410 A100-days (410 A100s on 1 day)
GPT-3	175 Bn	60k A100-days (2k A100 on 30 days)

Content credits: https://www.javatpoint.com/nlp

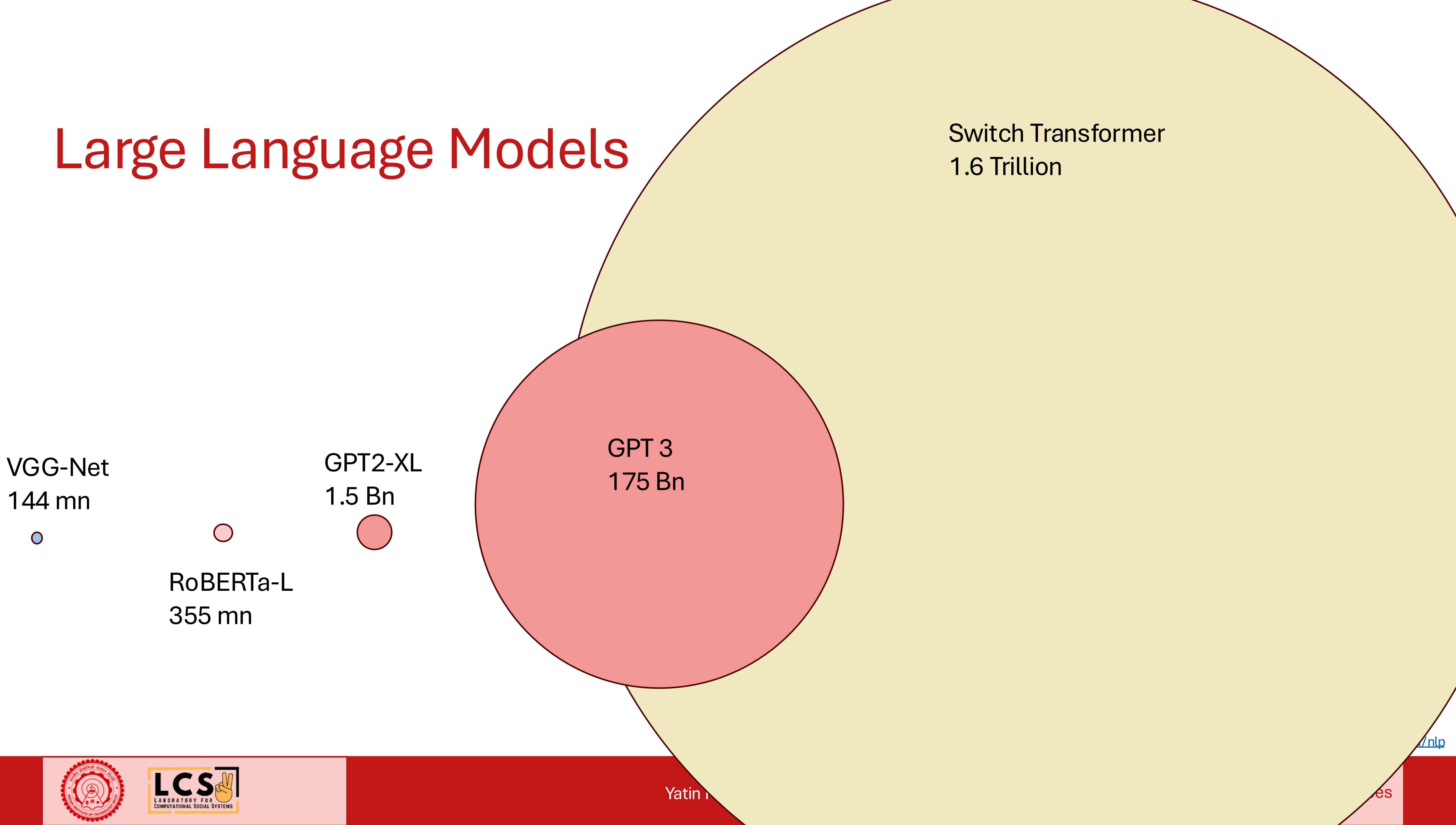
Large Language Models



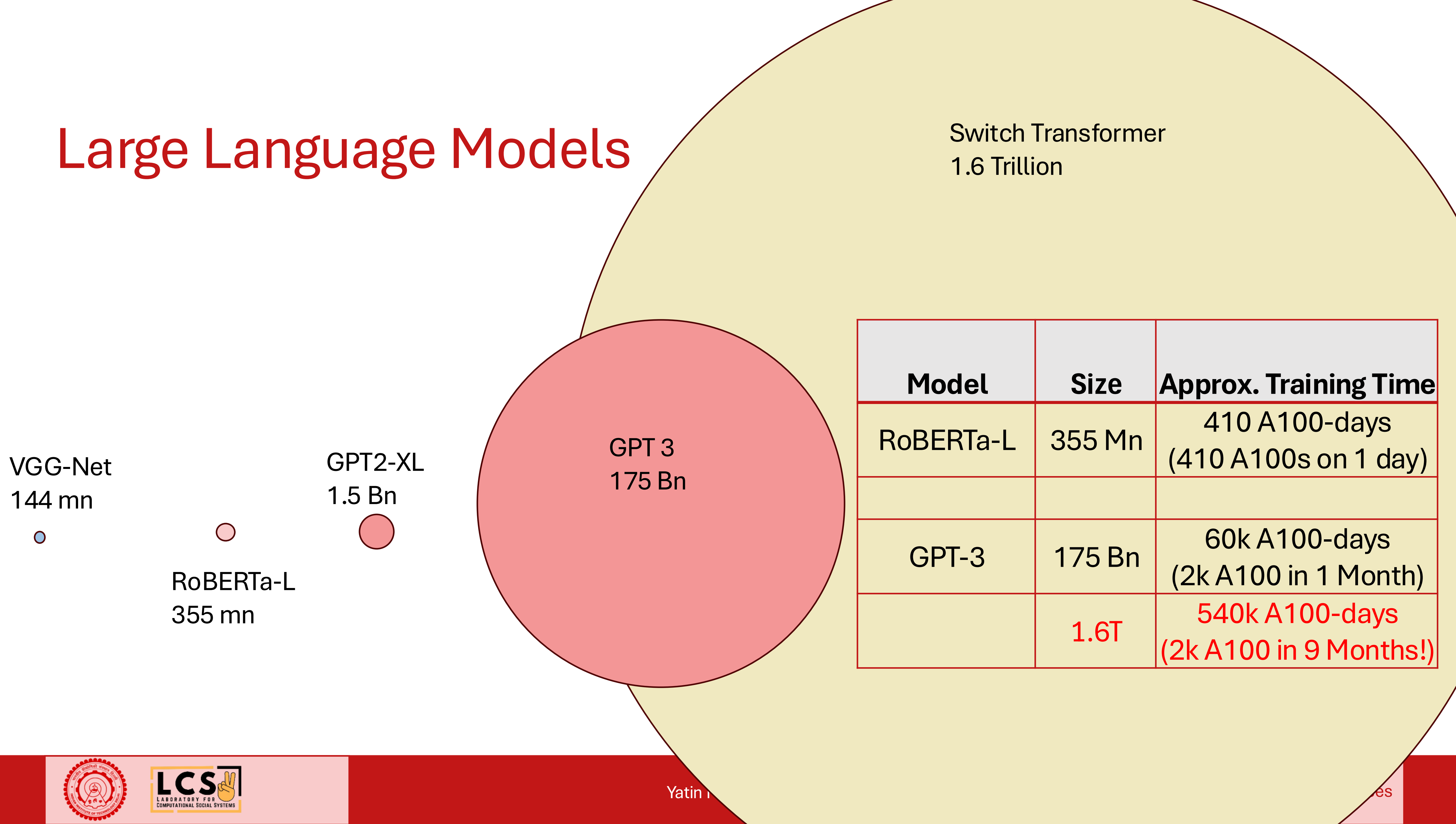
Model	Size	Approx. Training Time
RoBERTa-L	355 Mn	410 A100-days (410 A100s on 1 day)
GPT-3	175 Bn	60k A100-days (2k A100 on 30 days)

Content credits: https://www.javatpoint.com/nlp

Large Language Models



Large Language Models



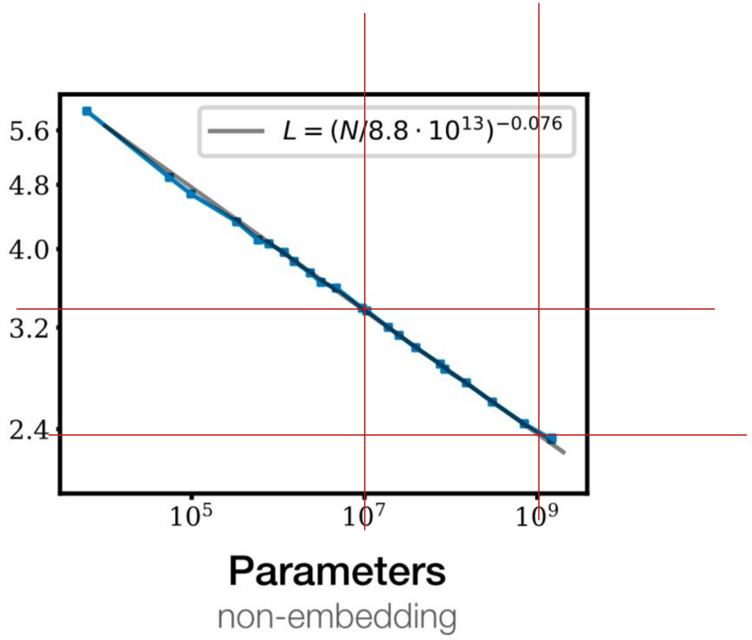
Model	Size	Approx. Training Time
RoBERTa-L	355 Mn	410 A100-days (410 A100s on 1 day)
GPT-3	175 Bn	60k A100-days (2k A100 in 1 Month)
	1.6T	540k A100-days (2k A100 in 9 Months!)

Why care about model size?

Content credits: https://www.javatpoint.com/nlp

Neural Scaling Laws

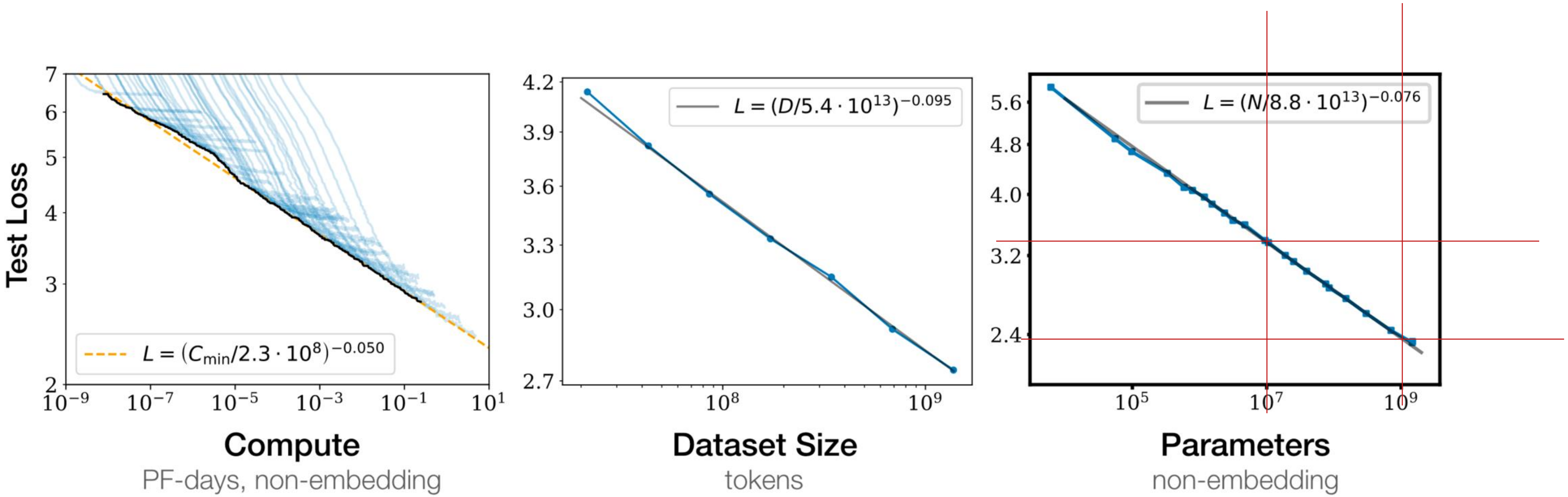
- Performance improve smoothly as we increase the compute, dataset size, or the model size



Source: Scaling Laws for Neural Language Models, Kaplan et al . 2020, Open AI

Neural Scaling Laws

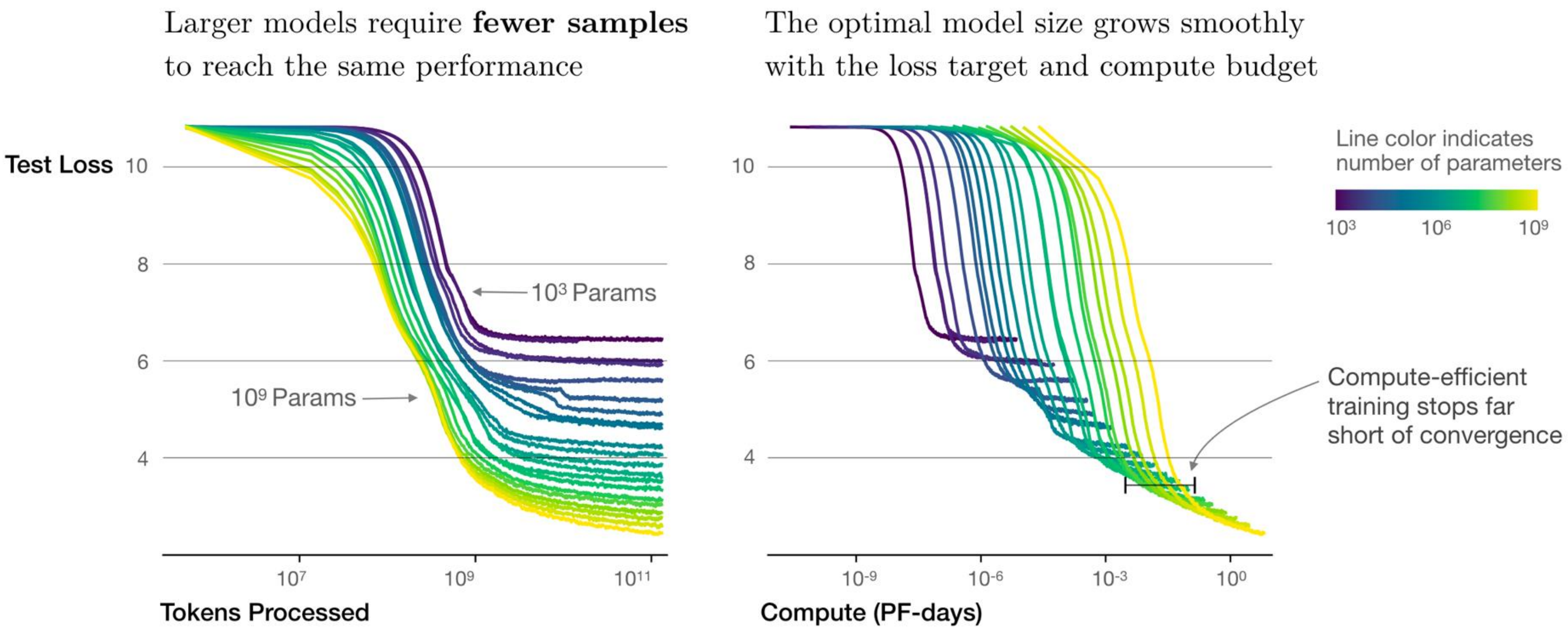
- Performance improve smoothly as we increase the compute, dataset size, or the model size



Source: Scaling Laws for Neural Language Models, Kaplan et al . 2020, Open AI

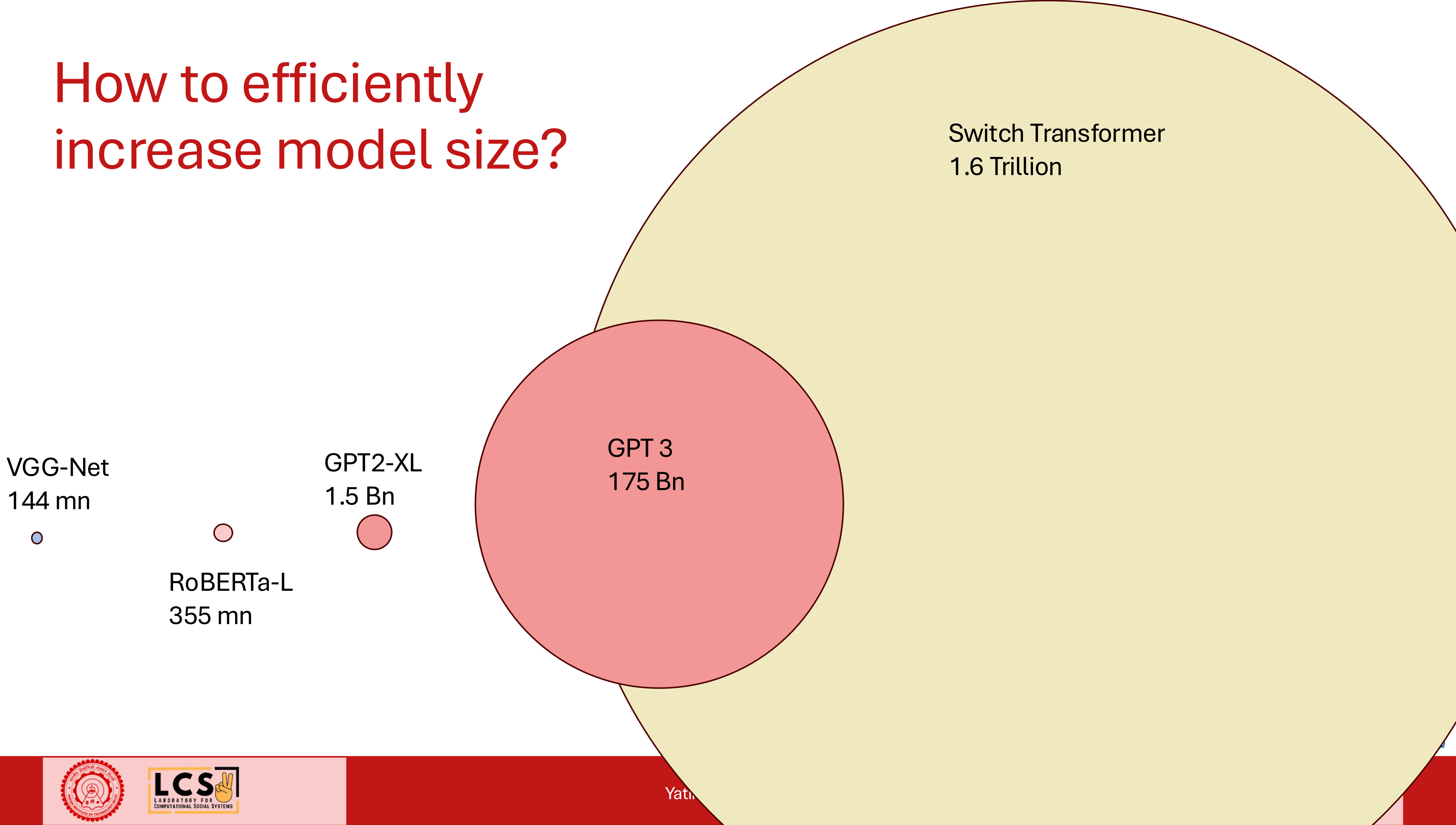
Neural Scaling Laws

- Performance improve smoothly as we increase the compute, dataset size, or the model size
- **Large models are more sample efficient** -- given a fixed computing budget, training a larger model for fewer steps is better than training a smaller model for more steps.

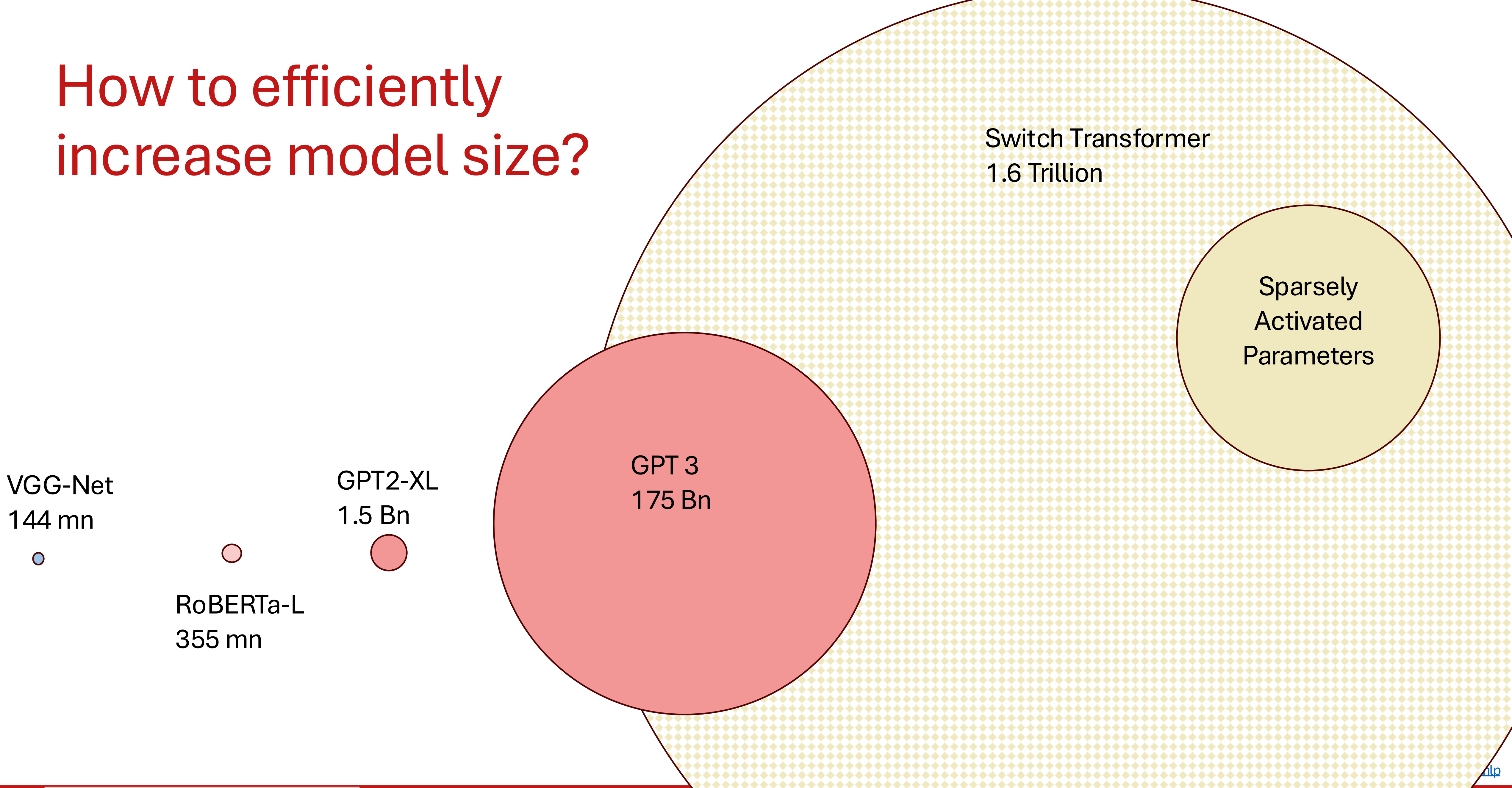


Source: Scaling Laws for Neural Language Models, Kaplan et al . 2020, Open AI

How to efficiently increase model size?



How to efficiently increase model size?



Mixture of Experts

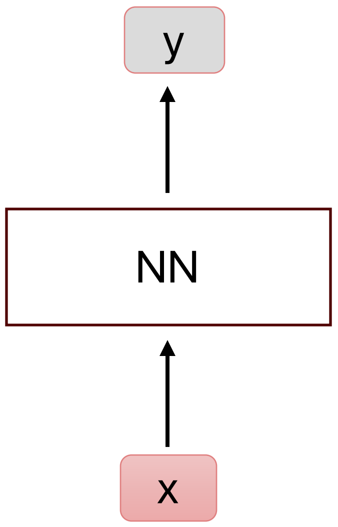
Adaptive Mixtures of Local Experts

Robert A. Jacobs
Michael I. Jordan
Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology, Cambridge, MA 02139 USA

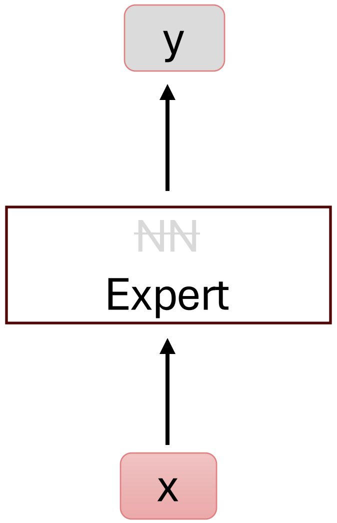
Steven J. Nowlan
Geoffrey E. Hinton
Department of Computer Science, University of Toronto, Toronto, Canada M5S 1A4

We present a new supervised learning procedure for systems composed of many separate networks, each of which learns to handle a subset of the complete set of training cases. The new procedure can be viewed either as a modular version of a multilayer supervised network, or as an associative version of competitive learning. It therefore provides a new link between these two apparently different approaches. We demonstrate that the learning procedure divides up a vowel discrimination task into appropriate subtasks, each of which can be solved by a very simple expert network.

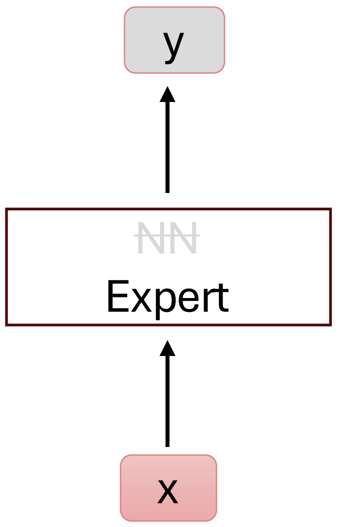
Mixture of Experts

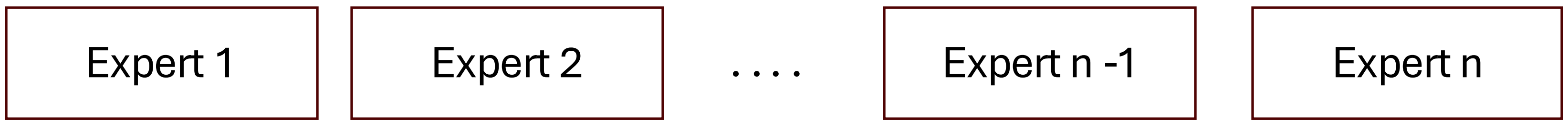


Mixture of Experts

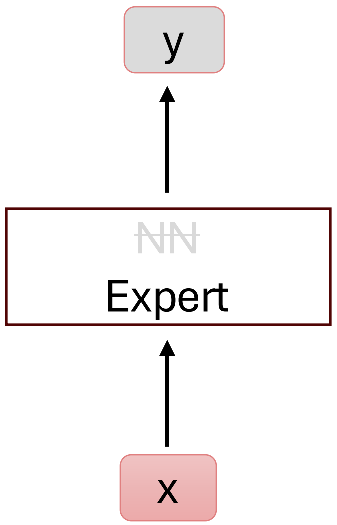


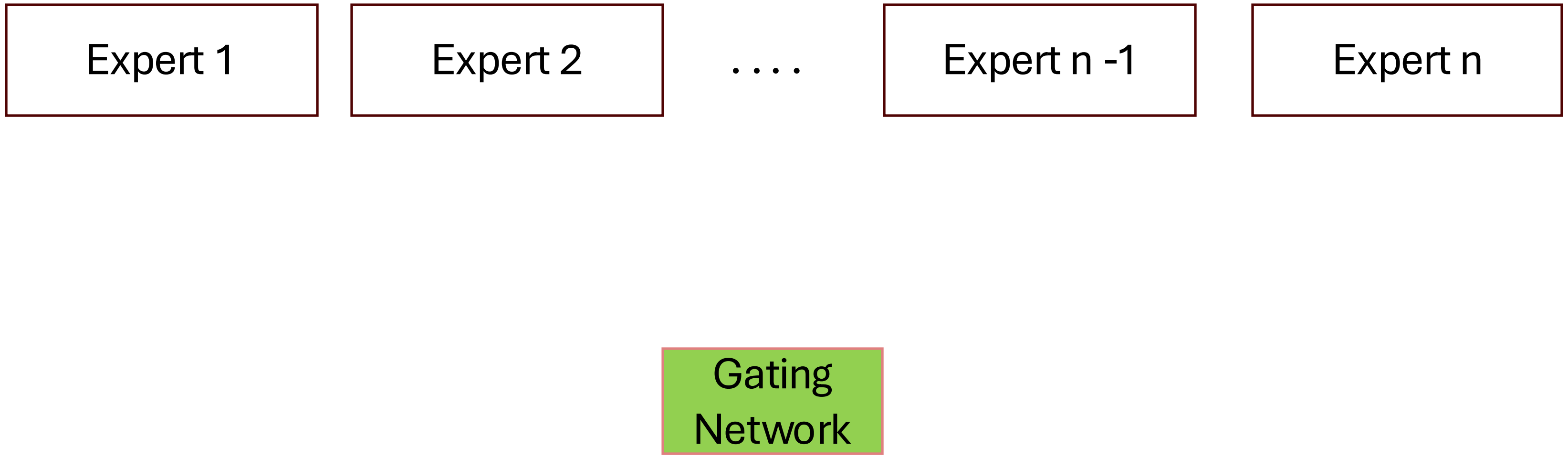
Mixture of Experts



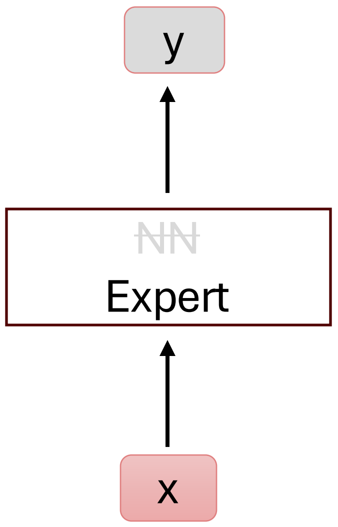


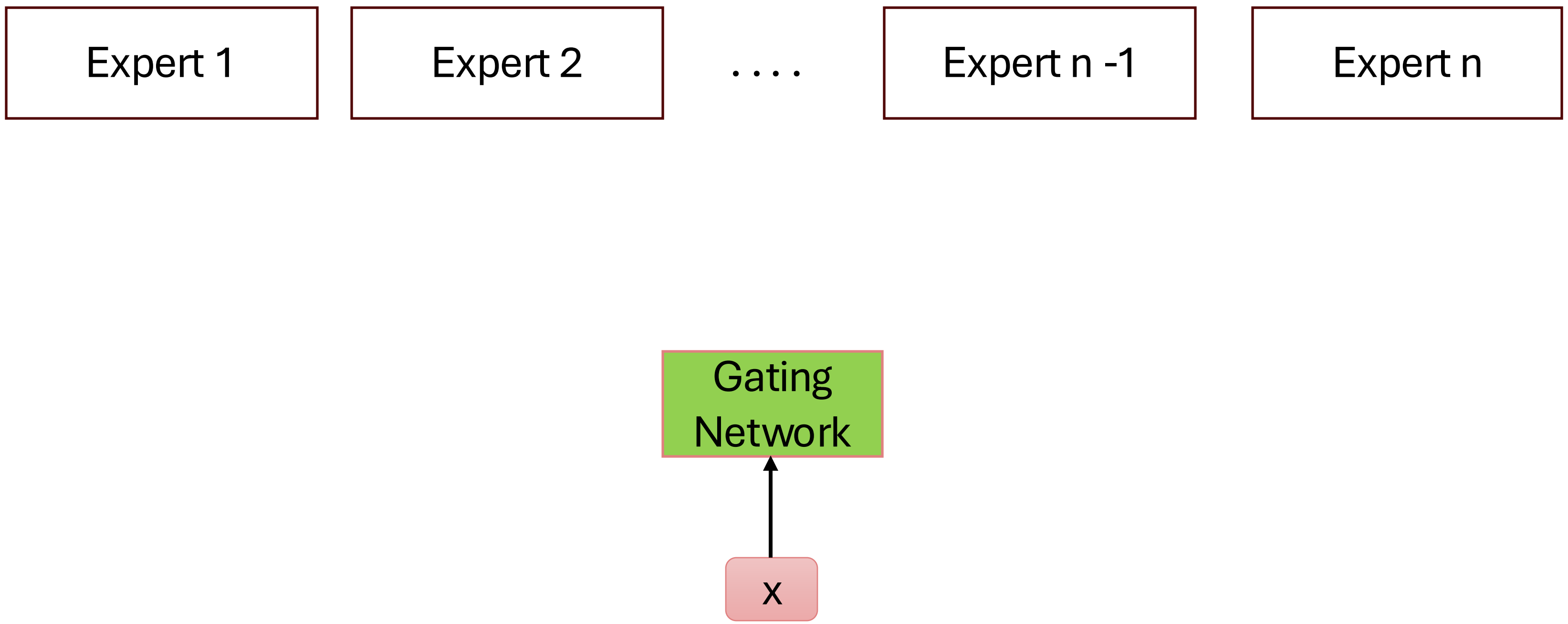
Mixture of Experts



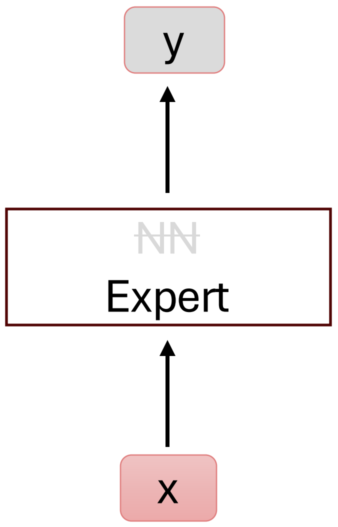


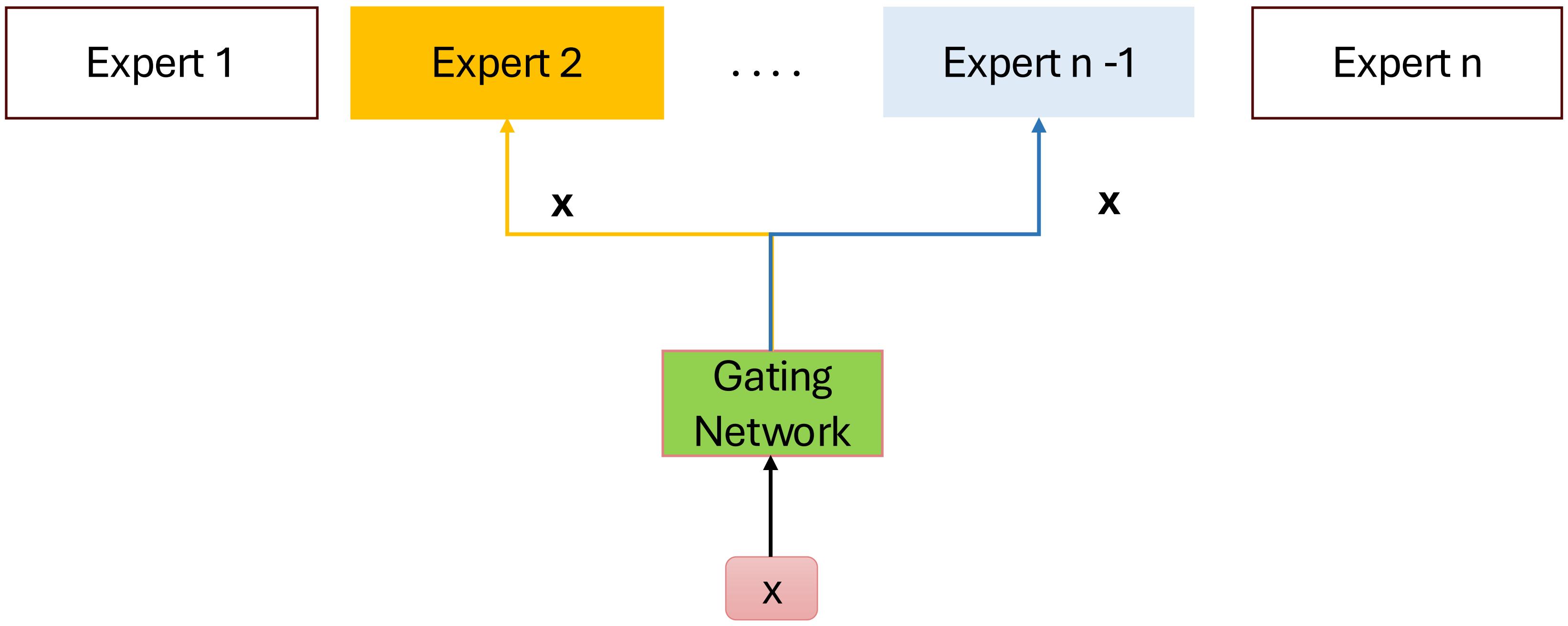
Mixture of Experts



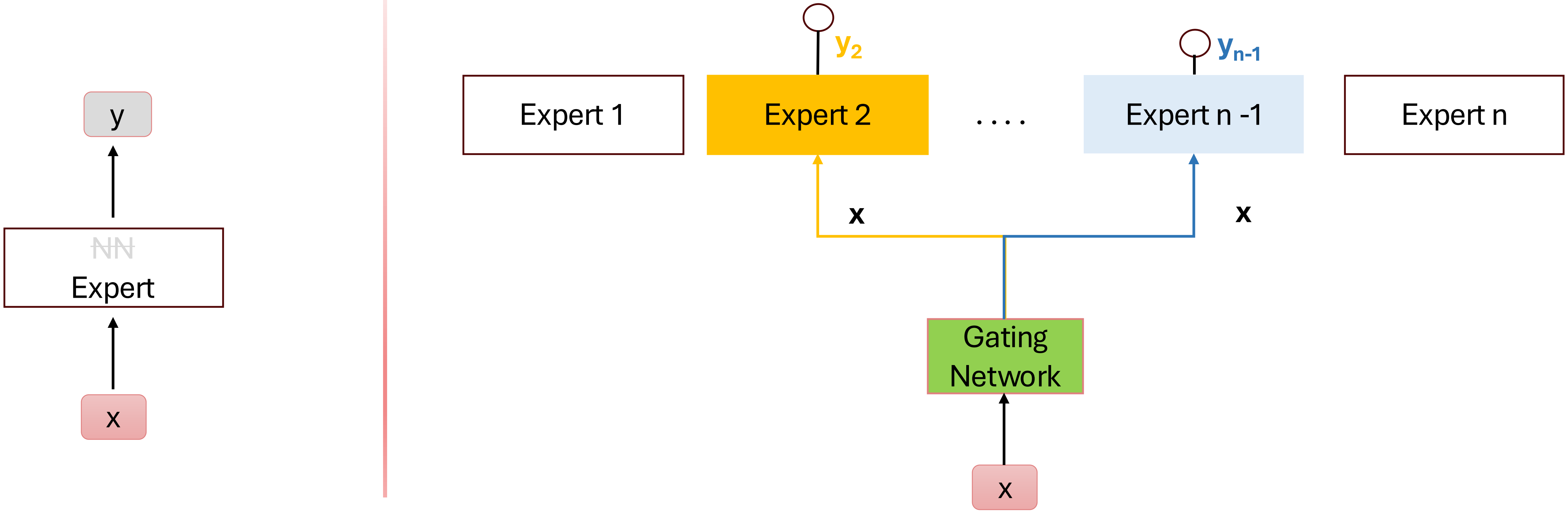


Mixture of Experts

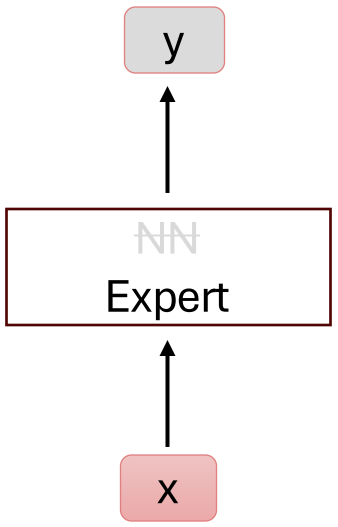


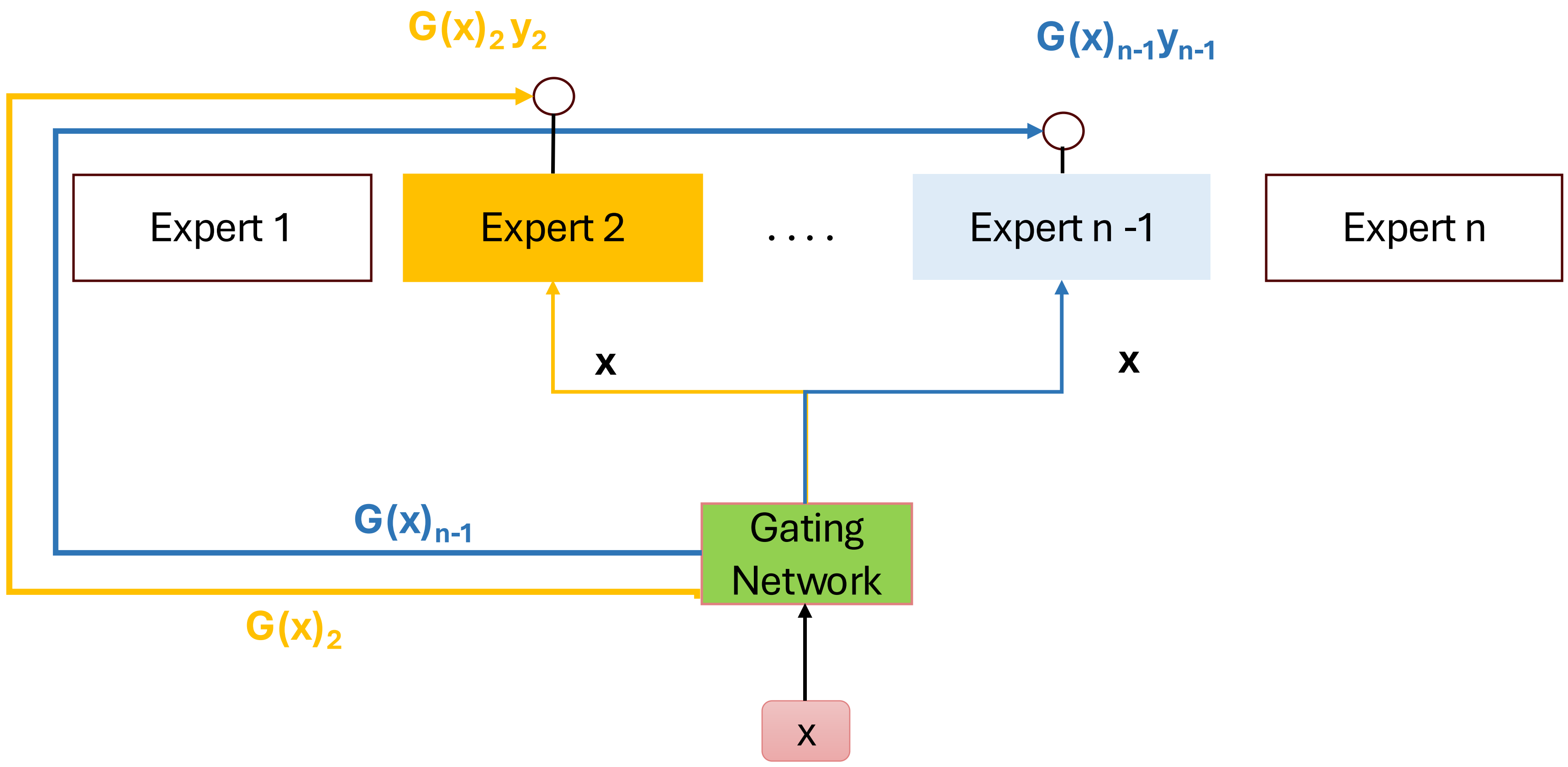


Mixture of Experts

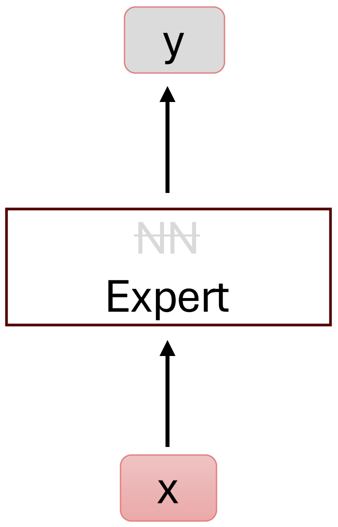


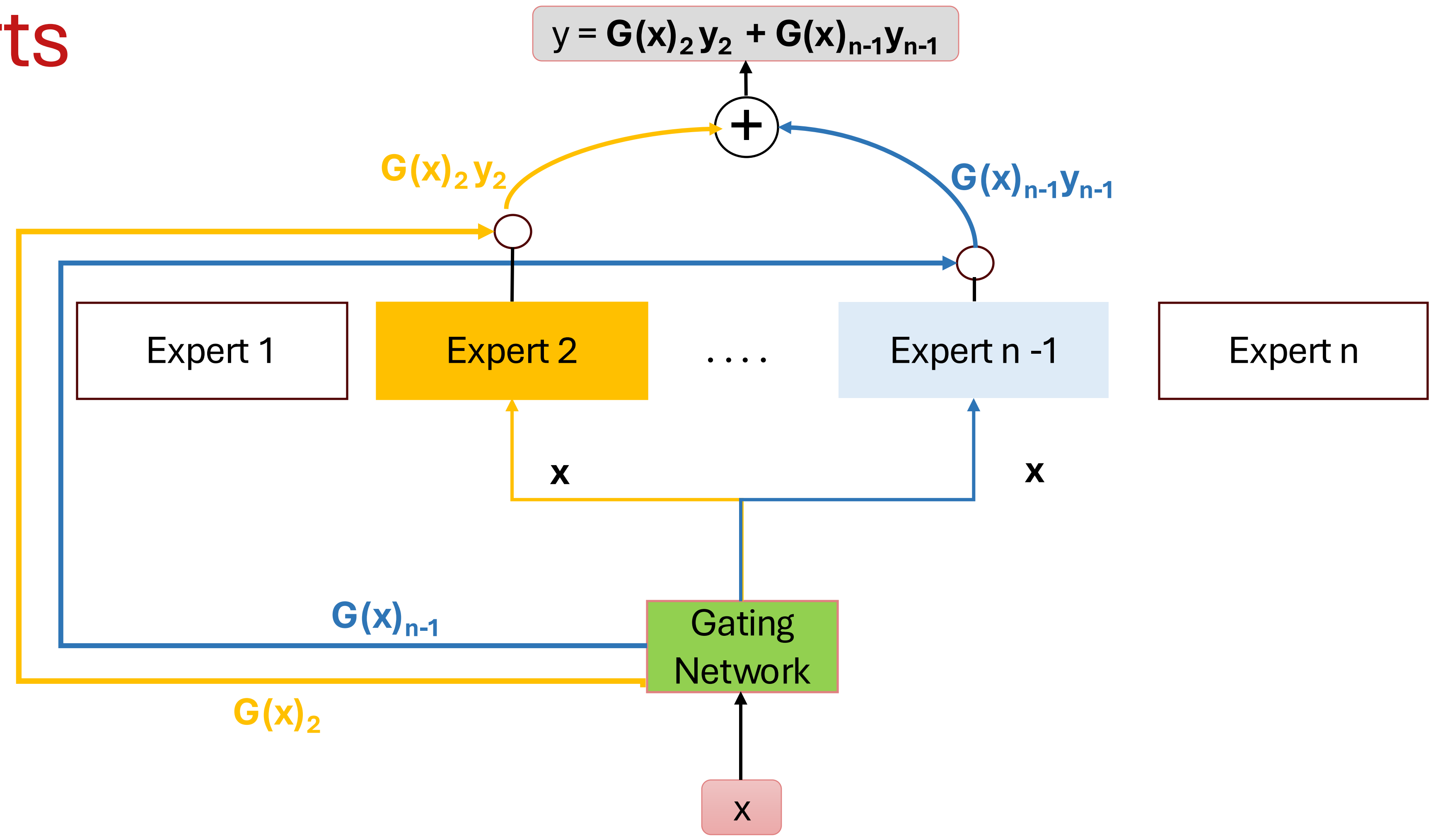
Mixture of Experts





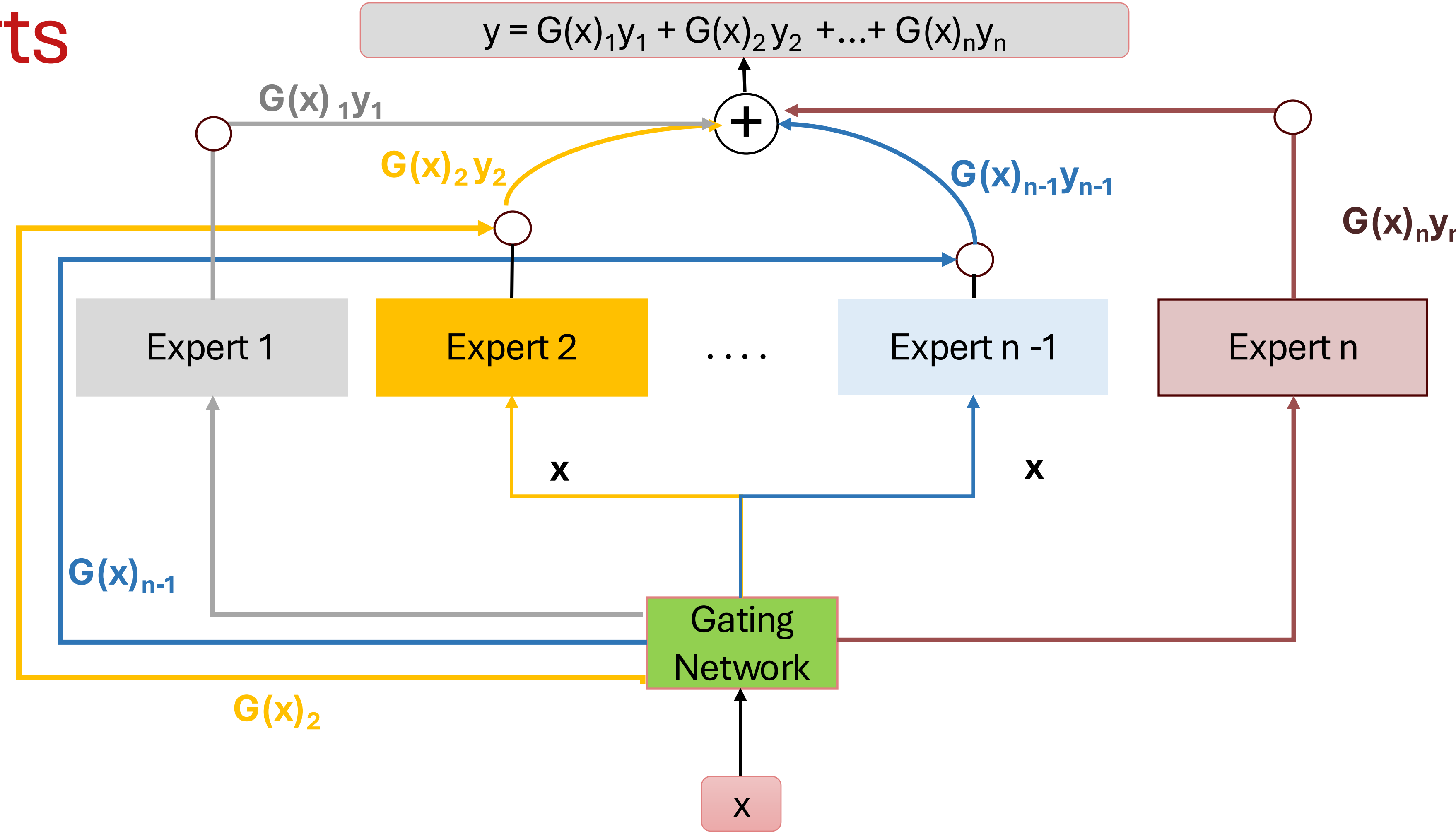
Mixture of Experts





Mixture of Experts

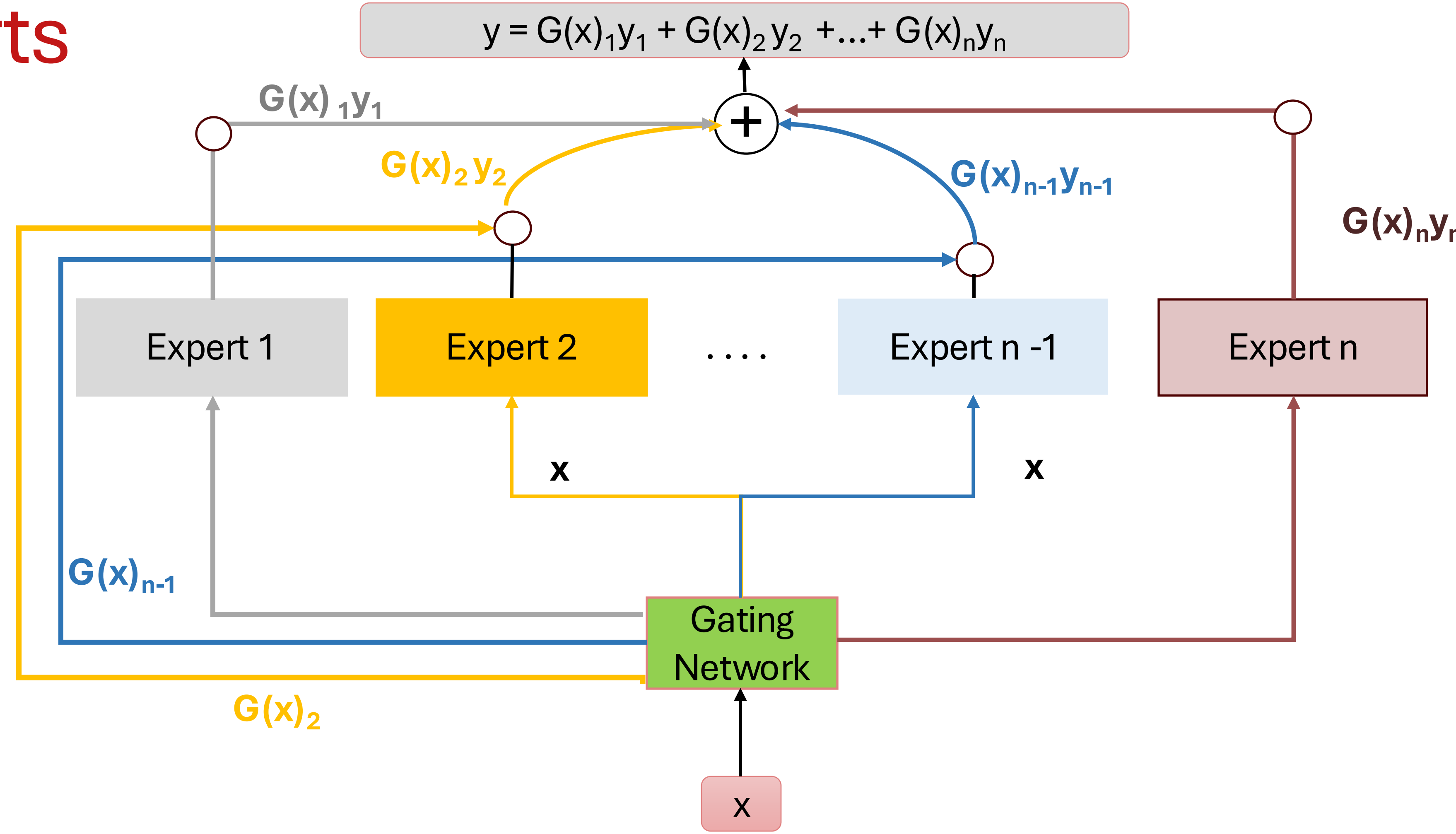
$$G_\sigma(x) = \text{Softmax}(x \cdot W_g)$$



Mixture of Experts

$$G_\sigma(x) = \text{Softmax}(x \cdot W_g)$$

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



Mixture of Experts - Chronology

- Mixture of Experts Model [Jacobs et al., 1991; Jordan and Jacobs, 1994; Jordan et al., 1997; Tresp, 2001; Collobert et al., 2002;]

Content credits: https://www.youtube.com/watch?v=TwHPxUAuqy4

Mixture of Experts - Chronology

- Mixture of Experts Model [Jacobs et al., 1991; Jordan and Jacobs, 1994; Jordan et al., 1997; Tresp, 2001; Collobert et al., 2002;]

- MoE layer in Deep Learning [Eigen et al., 2013; Shazeer et al., 2017; Dauphin et al., 2017; Vaswani et al., 2017]

Content credits: https://www.youtube.com/watch?v=TwHPxUAuqy4

Mixture of Experts - Chronology

- Mixture of Experts Model [Jacobs et al., 1991; Jordan and Jacobs, 1994; Jordan et al., 1997; Tresp, 2001; Collobert et al., 2002;]

- MoE layer in Deep Learning [Eigen et al., 2013; Shazeer et al., 2017; Dauphin et al., 2017; Vaswani et al., 2017]

- MoE layer in Transformer based LLMs [Fedus et al., 2021; Du, Nan, et al., 2021]

Content credits: https://www.youtube.com/watch?v=TwHPxUAuqy4

Mixture of Experts - Chronology

- Mixture of Experts Model [Jacobs et al., 1991; Jordan and Jacobs, 1994; Jordan et al., 1997; Tresp, 2001; Collobert et al., 2002;]

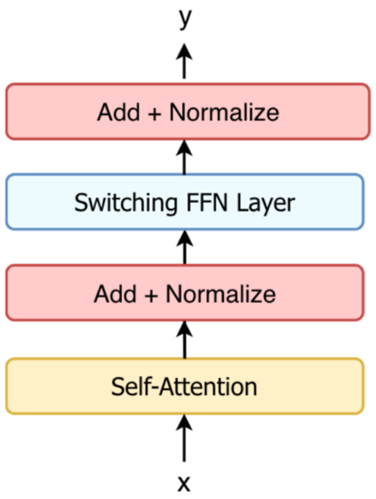
- MoE layer in Deep Learning [Eigen et al., 2013; Shazeer et al., 2017; Dauphin et al., 2017; Vaswani et al., 2017]

- MoE layer in Transformer based LLMs [Fedus et al., 2021; Du, Nan, et al., 2021]

- Mixtral-8x7B [Jiang et al. 2024] - Apache 2.0 license; surpasses GPT-3.5 Turbo, Claude-2.1, Gemini Pro, and Llama 2 70B - chat model on human benchmarks

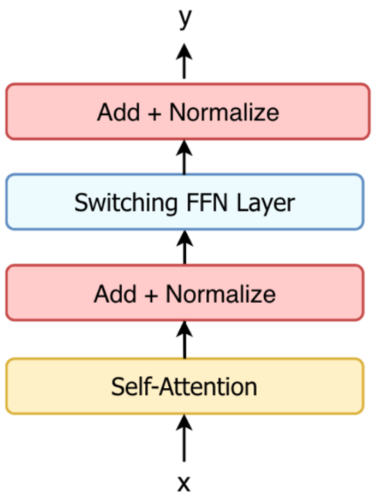
Content credits: https://www.youtube.com/watch?v=TwHPxUAuqy4

Mixture of Experts as a Layer

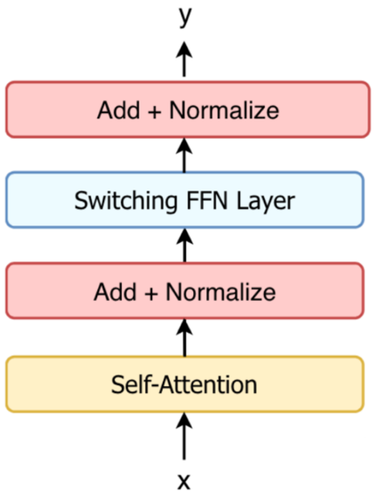


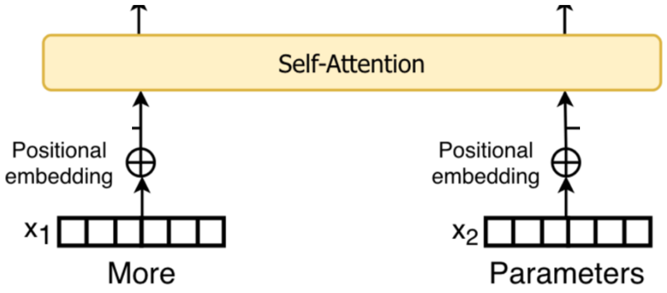
Content credits: Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity

Mixture of Experts as a Layer

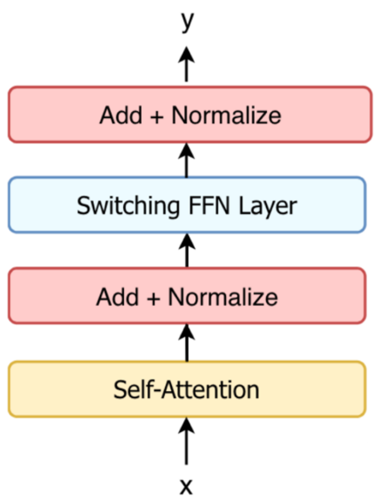


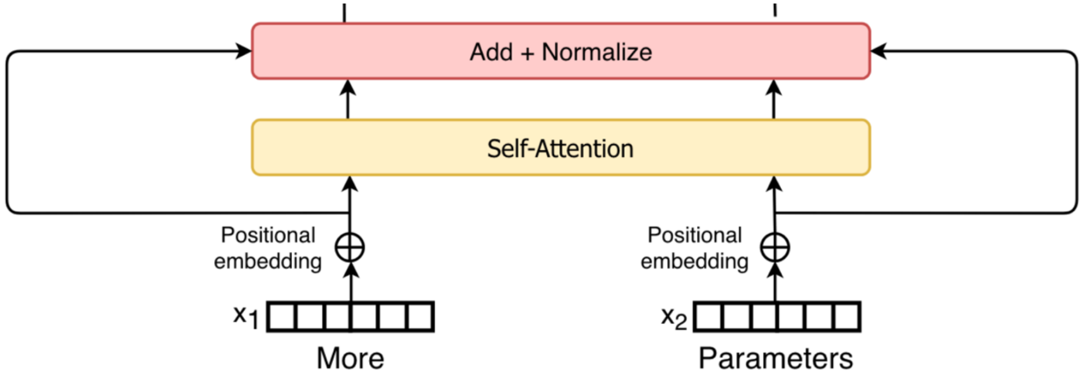
Mixture of Experts as a Layer



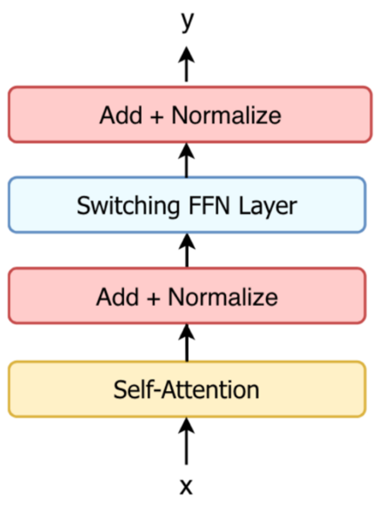


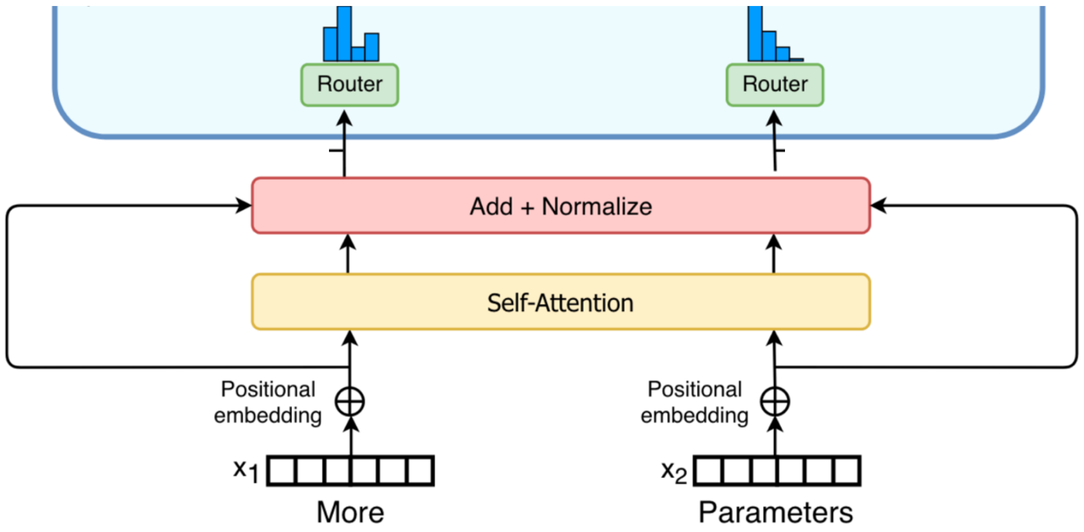
Mixture of Experts as a Layer



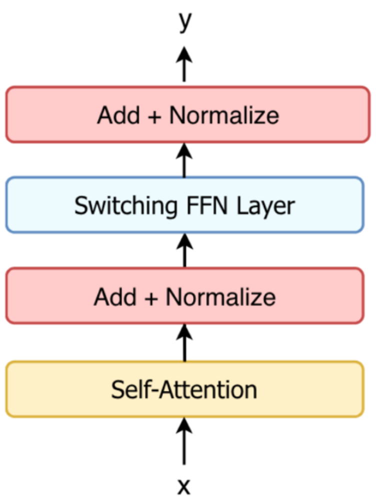


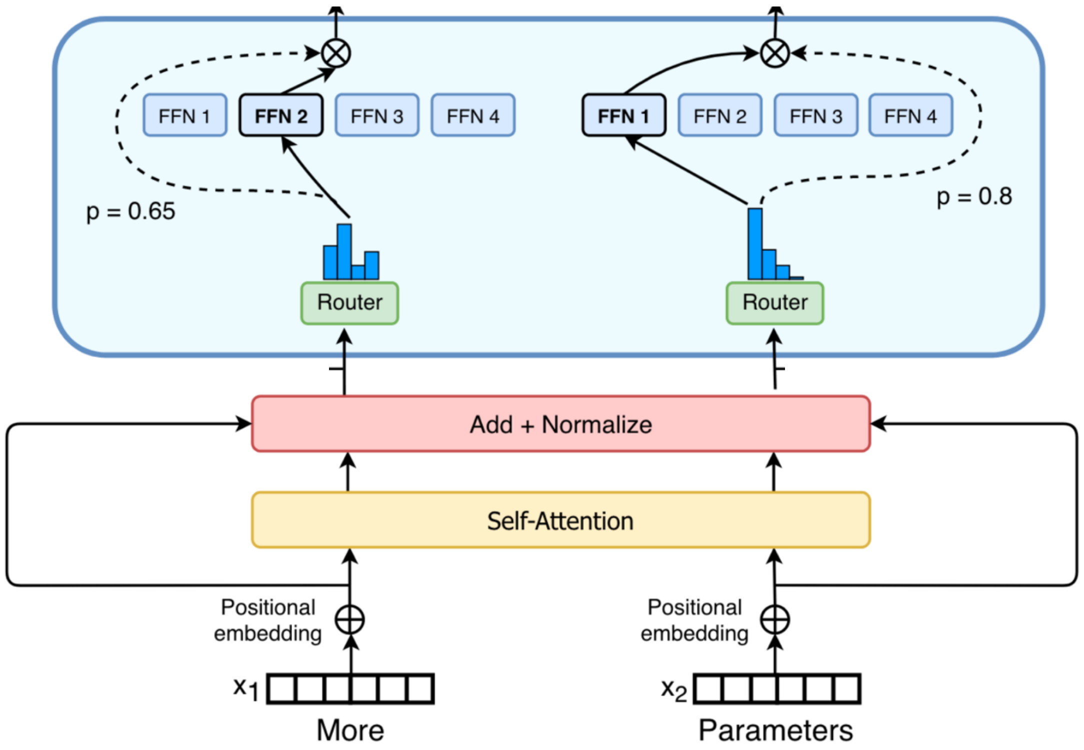
Mixture of Experts as a Layer



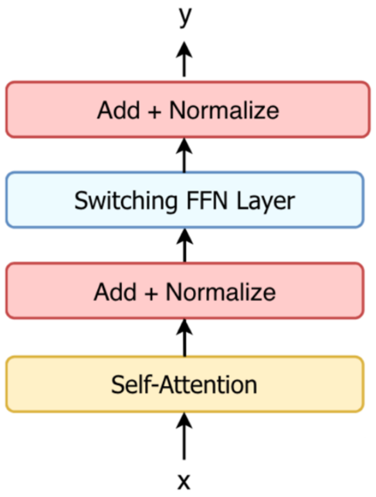


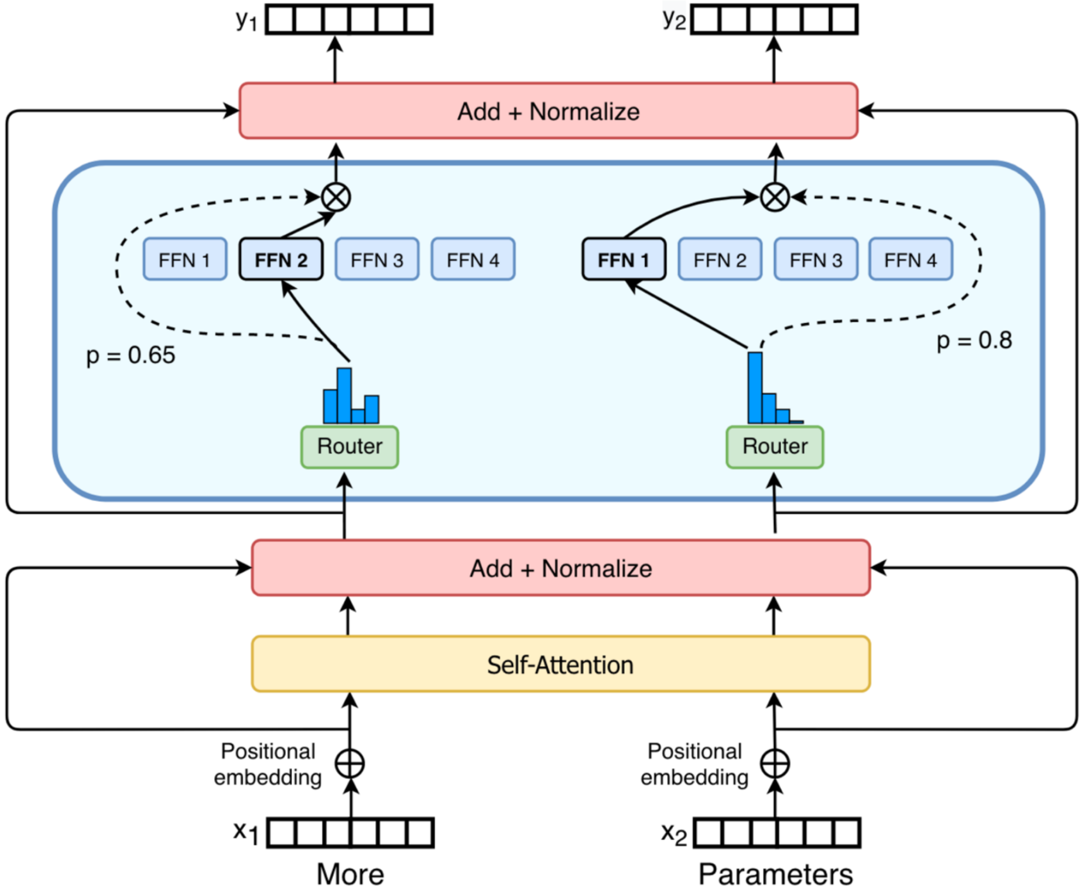
Mixture of Experts as a Layer



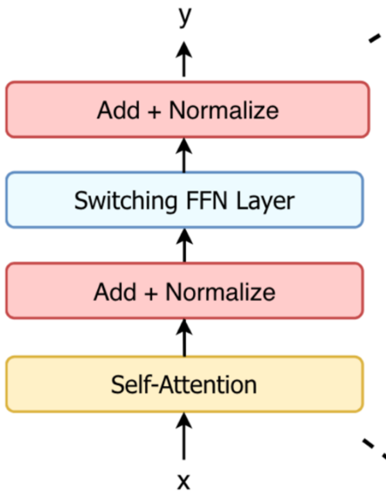


Mixture of Experts as a Layer

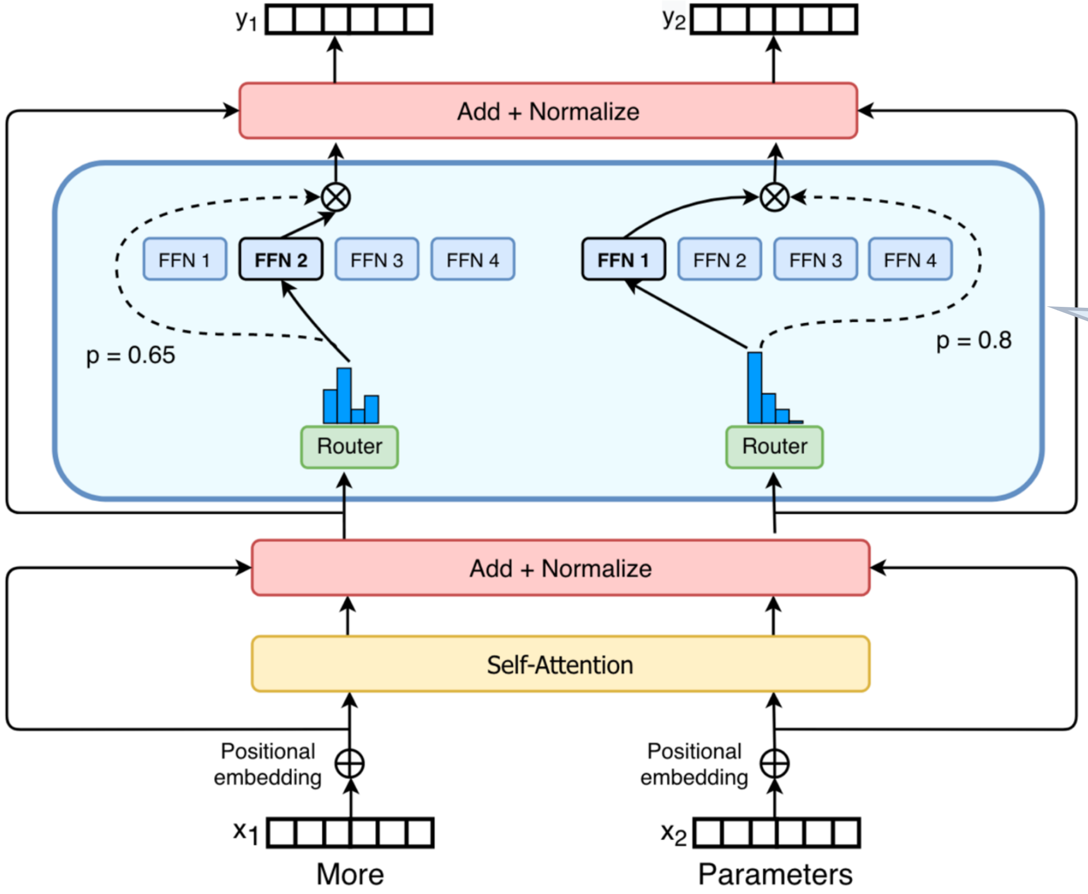




Mixture of Experts as a Layer



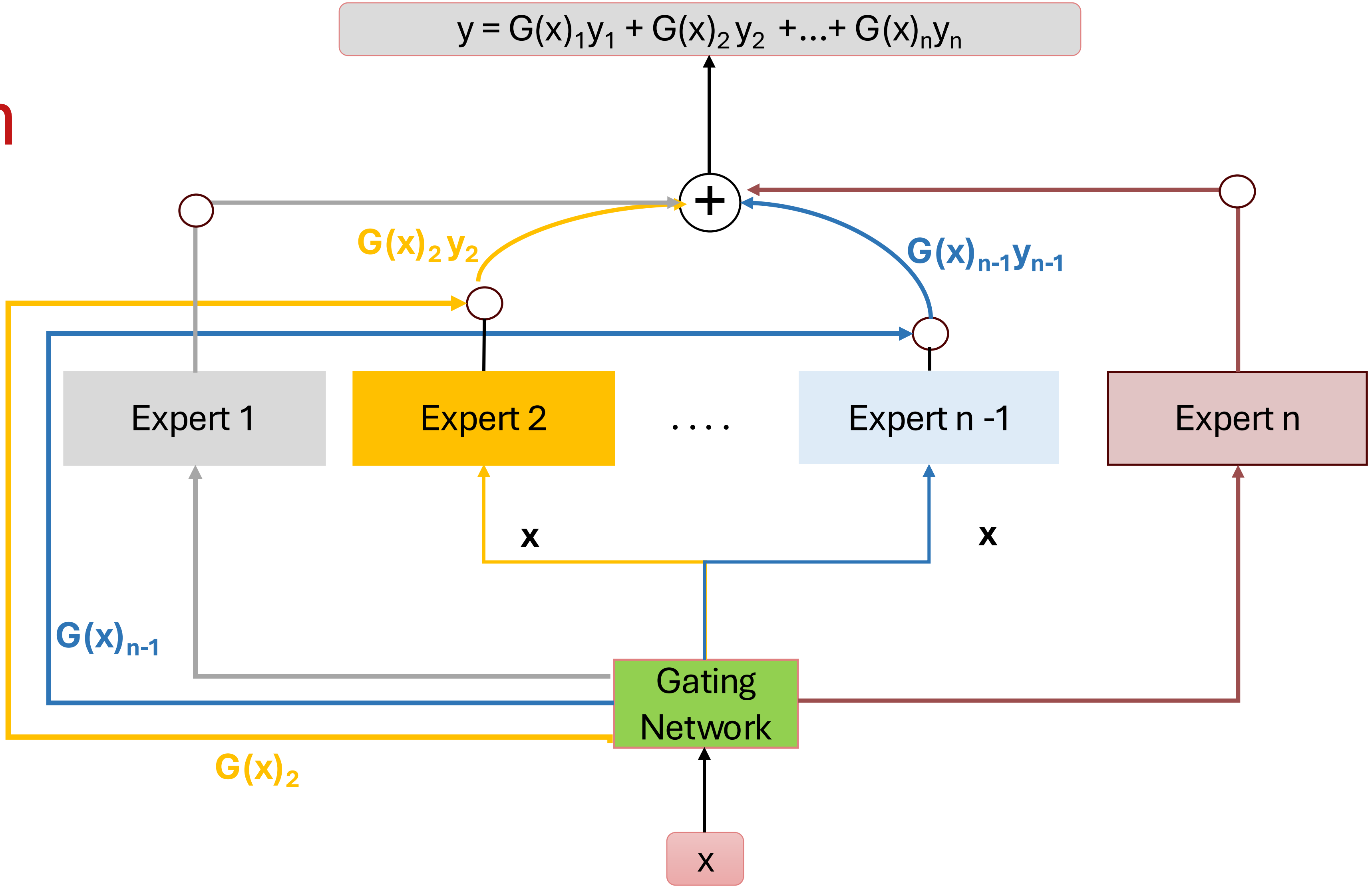
Dense Transformer Block



MoE Layer

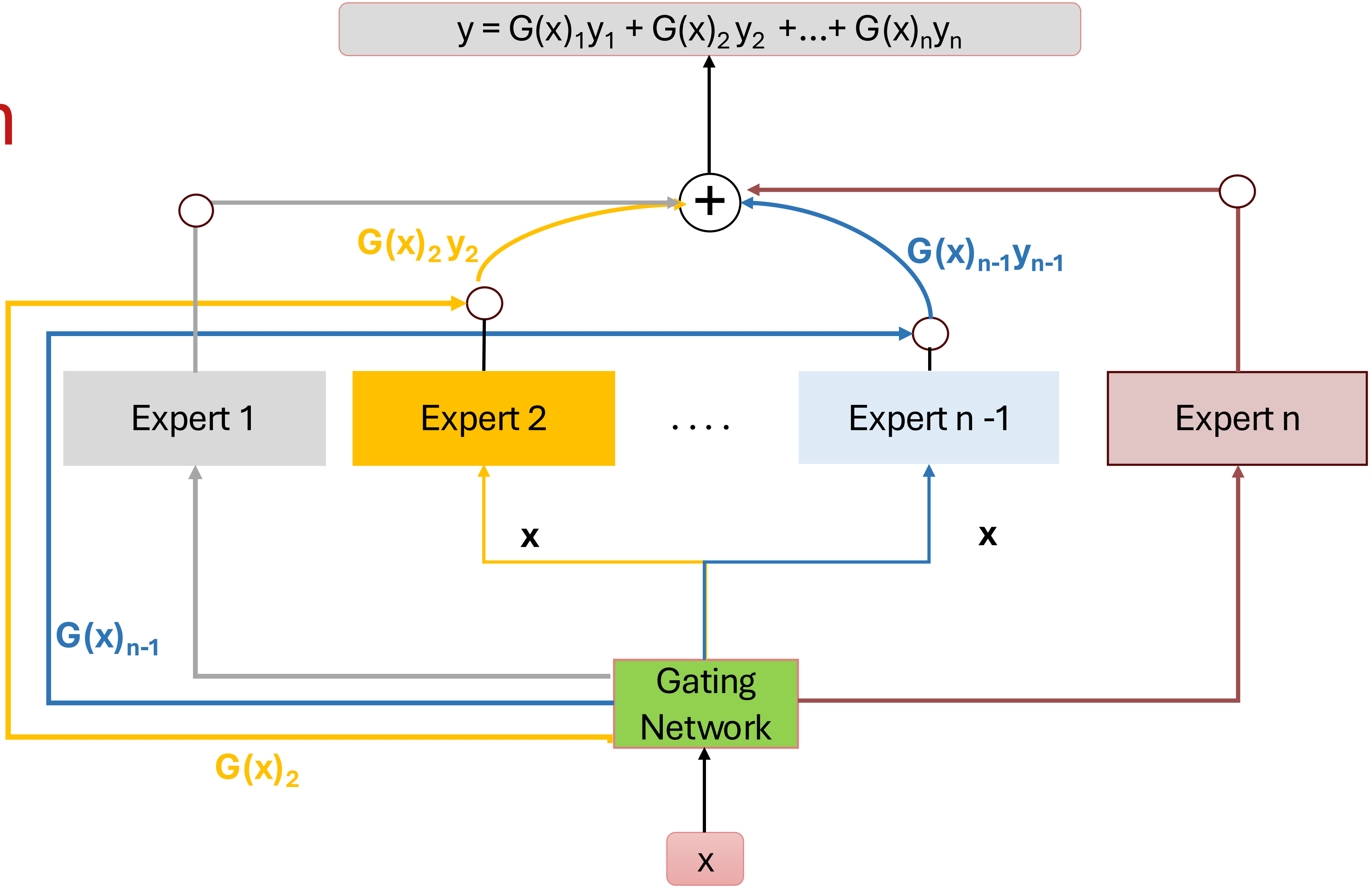
Sparsely Activated MoE Transformer Block

MoE Layer Backpropagation



MoE Layer Backpropagation

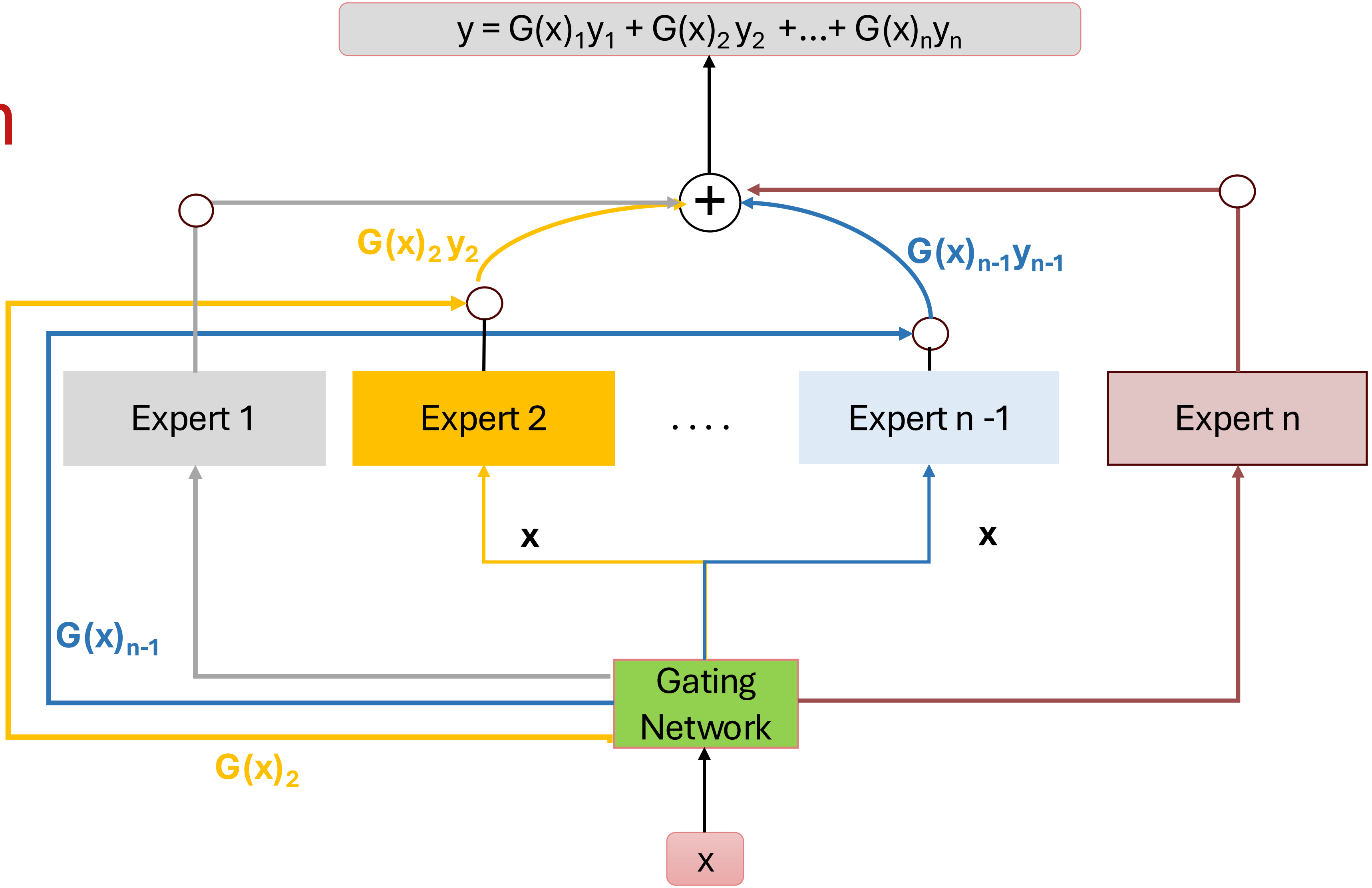
$$G_\sigma(x) = \text{Softmax}(x \cdot W_g)$$



MoE Layer Backpropagation

$$G_\sigma(x) = \text{Softmax}(x \cdot W_g)$$

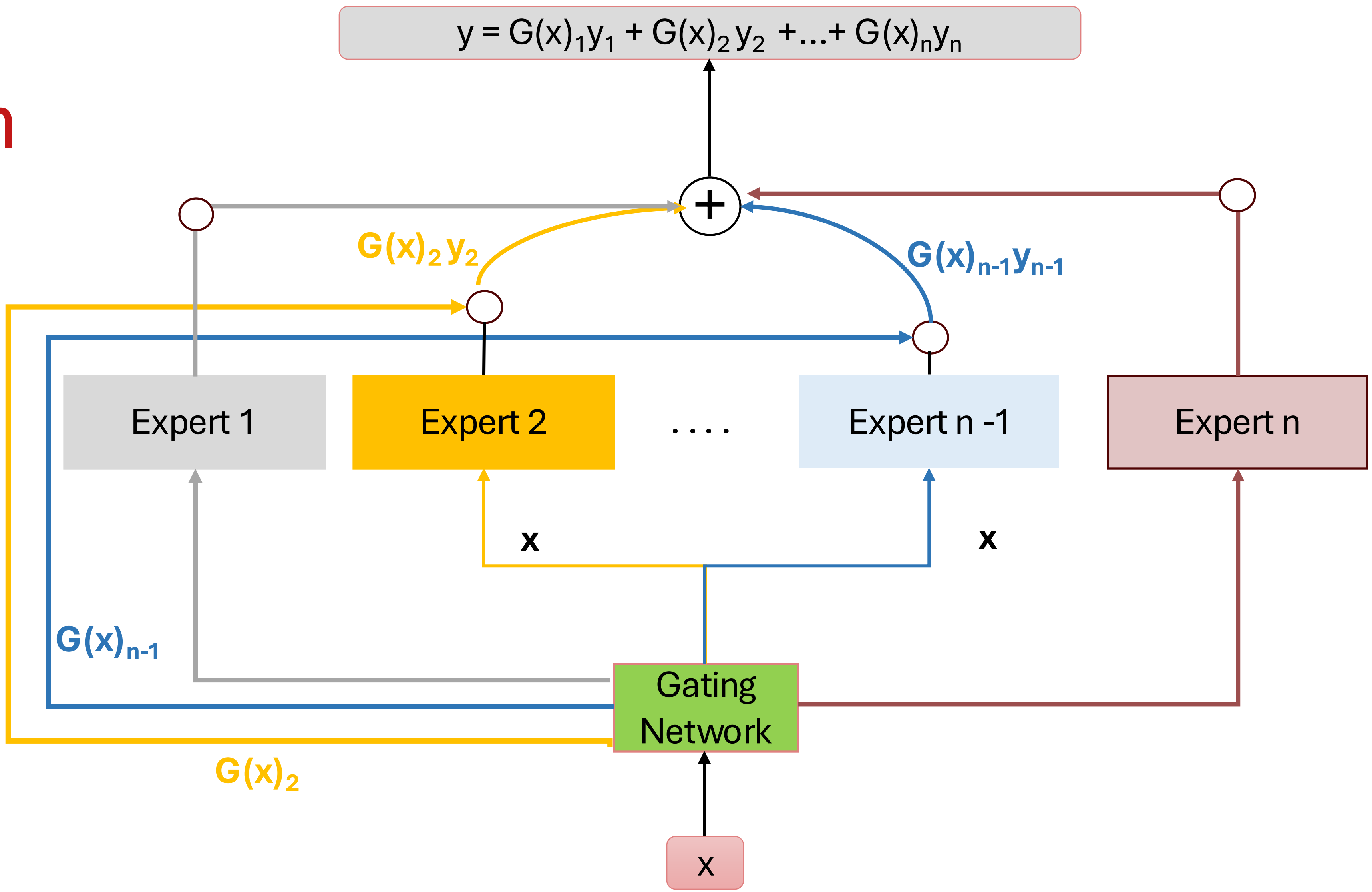
$$y = \sum_{i=1}^{n} G(x)_i E_i(x)$$



MoE Layer Backpropagation

$$G_\sigma(x) = \text{Softmax}(x \cdot W_g)$$

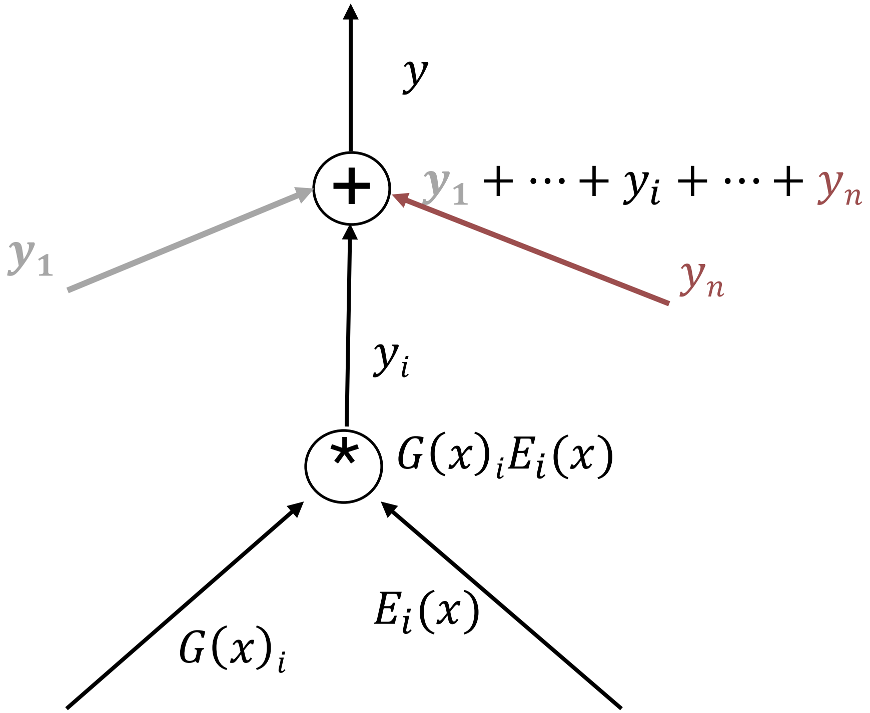
$$y = \sum_{i=1}^{n} \underbrace{G(x)_i E_i(x)}_{y_i}$$



MoE Layer Backpropagation

$$G_\sigma(x) = \text{Softmax}(x \cdot W_g)$$

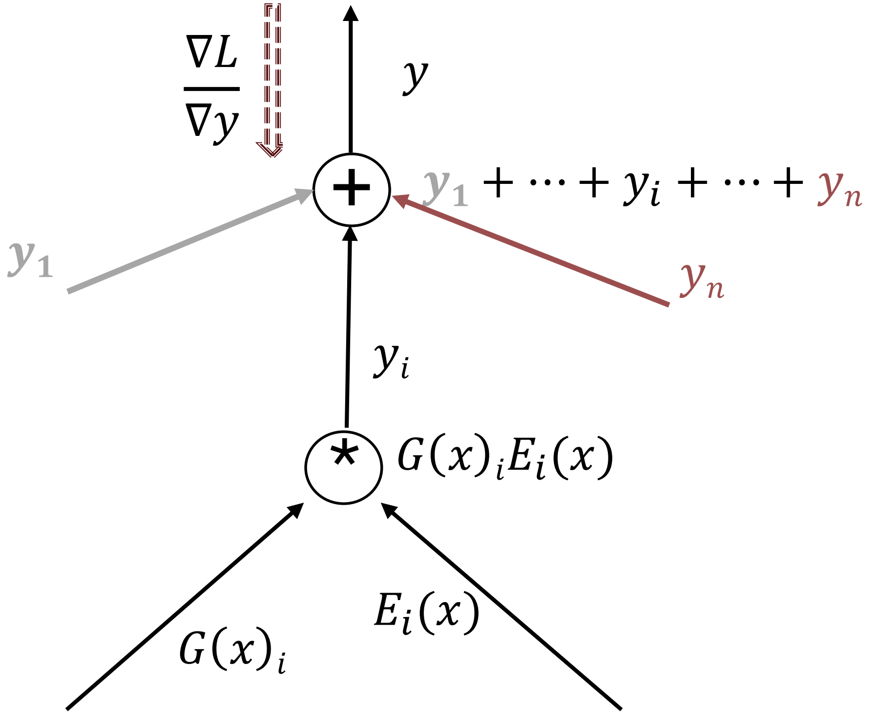
$$y = \sum_{i=1}^{n} \underbrace{G(x)_i E_i(x)}_{y_i}$$



MoE Layer Backpropagation

$$G_\sigma(x) = \text{Softmax}(x \cdot W_g)$$

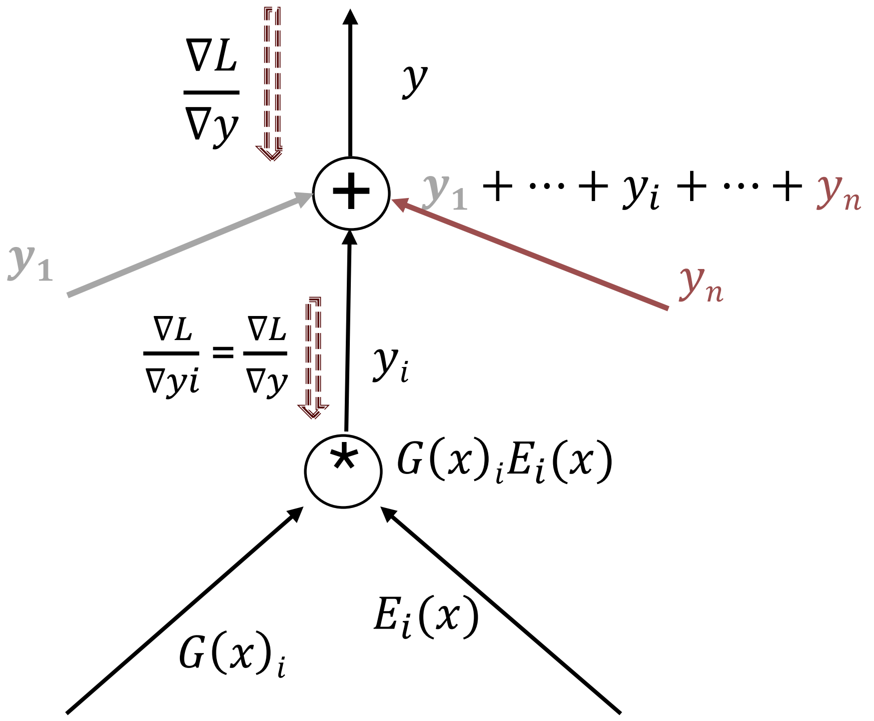
$$y = \sum_{i=1}^{n} \underbrace{G(x)_i E_i(x)}_{y_i}$$



MoE Layer Backpropagation

$$G_\sigma(x) = \text{Softmax}(x \cdot W_g)$$

$$y = \sum_{i=1}^{n} \underbrace{G(x)_i E_i(x)}_{y_i}$$



MoE Layer Backpropagation

$$G_\sigma(x) = \text{Softmax}(x \cdot W_g)$$

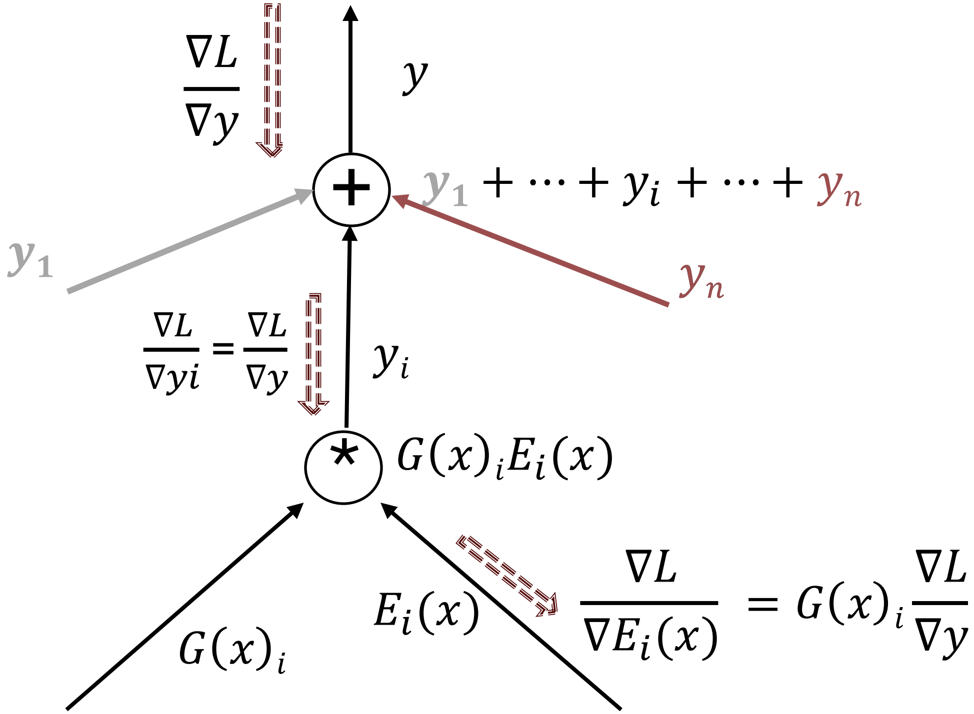
$$y = \sum_{i=1}^{n} \underbrace{G(x)_i E_i(x)}_{y_i}$$

$$\frac{\nabla L}{\nabla E_i(x)} = G(x)_i \frac{\nabla L}{\nabla y}$$

d-dim.

Scalar

d-dim. vector



MoE Layer Backpropagation

$$G_\sigma(x) = \text{Softmax}(x \cdot W_g)$$

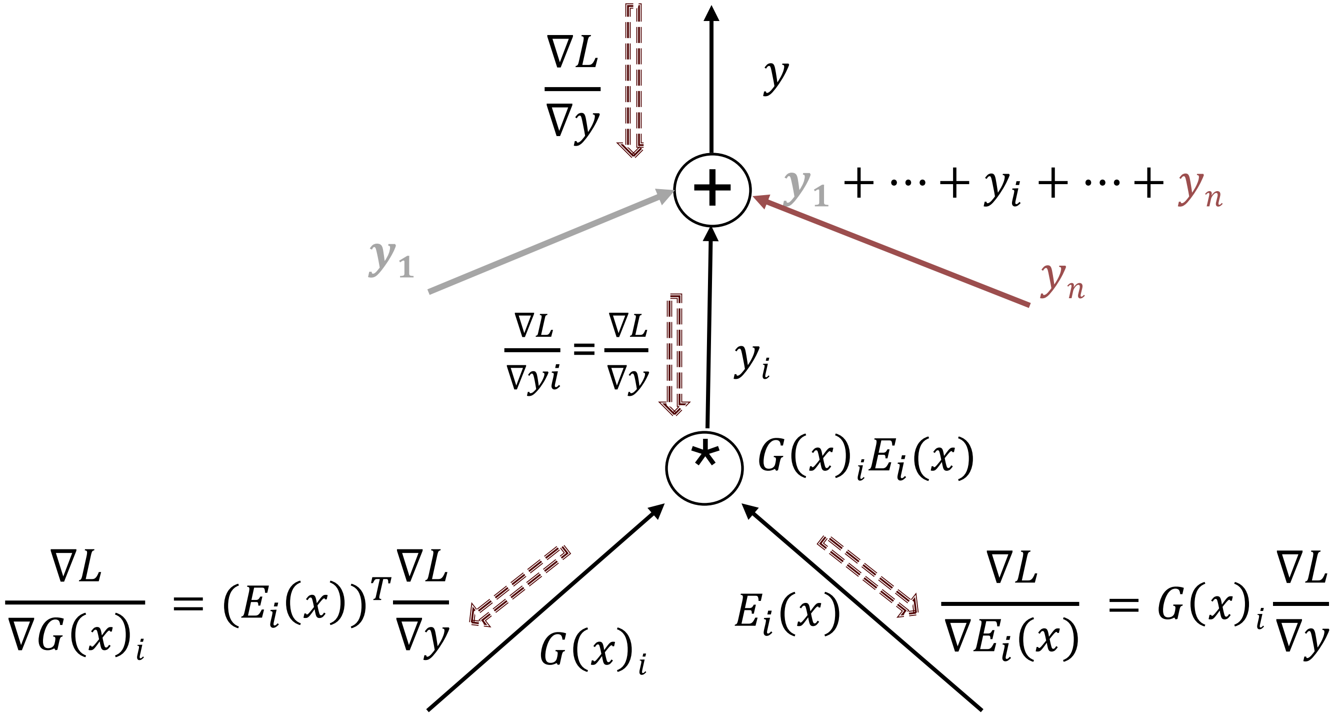
$$y = \sum_{i=1}^{n} \underbrace{G(x)_i E_i(x)}_{y_i}$$

$$\frac{\nabla L}{\nabla E_i(x)} = G(x)_i \frac{\nabla L}{\nabla y}$$

d-dim. | Scalar | d-dim. vector

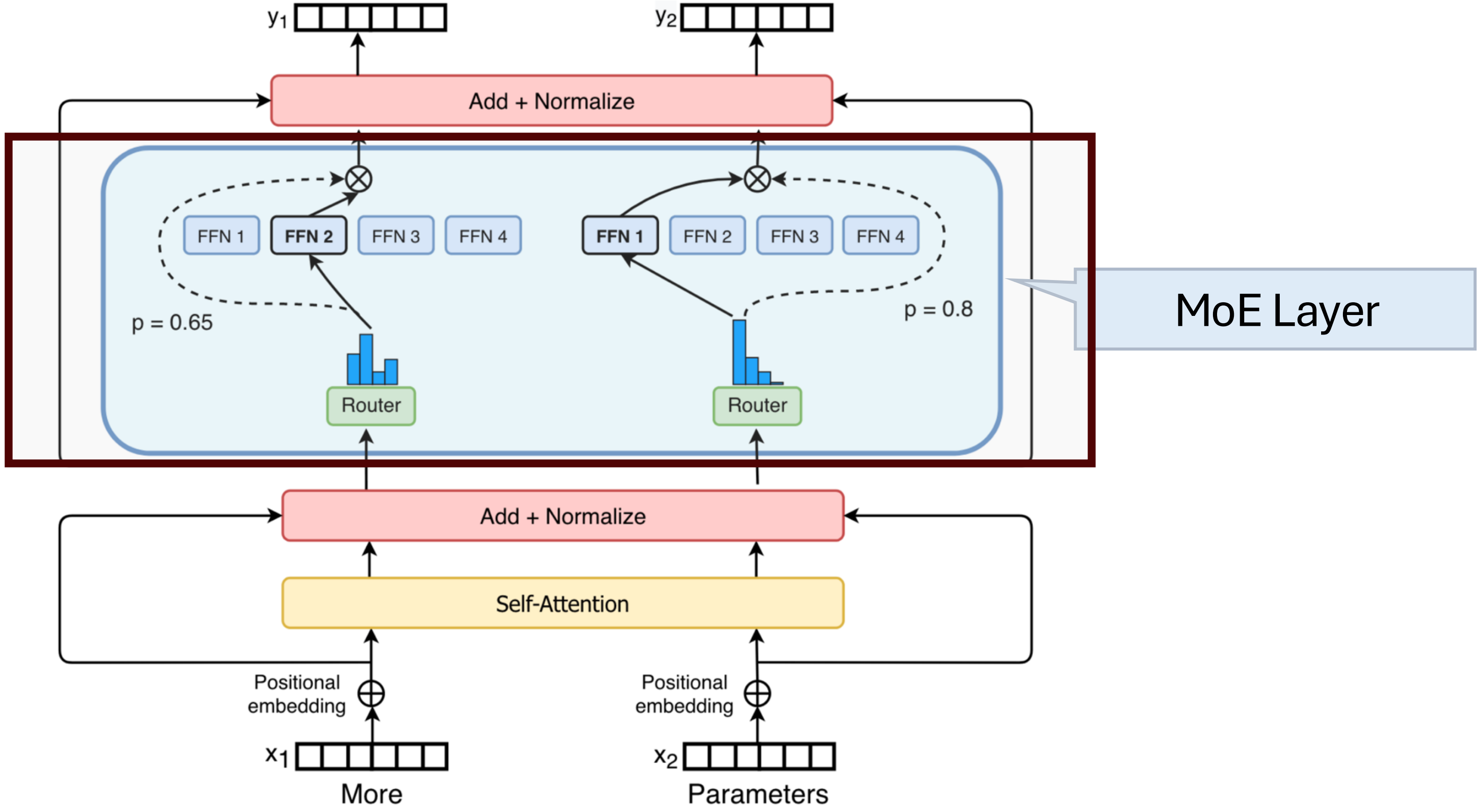
$$\frac{\nabla L}{\nabla G(x)_i} = (E_i(x))^T \frac{\nabla L}{\nabla y}$$

Scalar | d-dim. vector | d-dim. vector



Sparse MoE Layer – Greedy expert selection

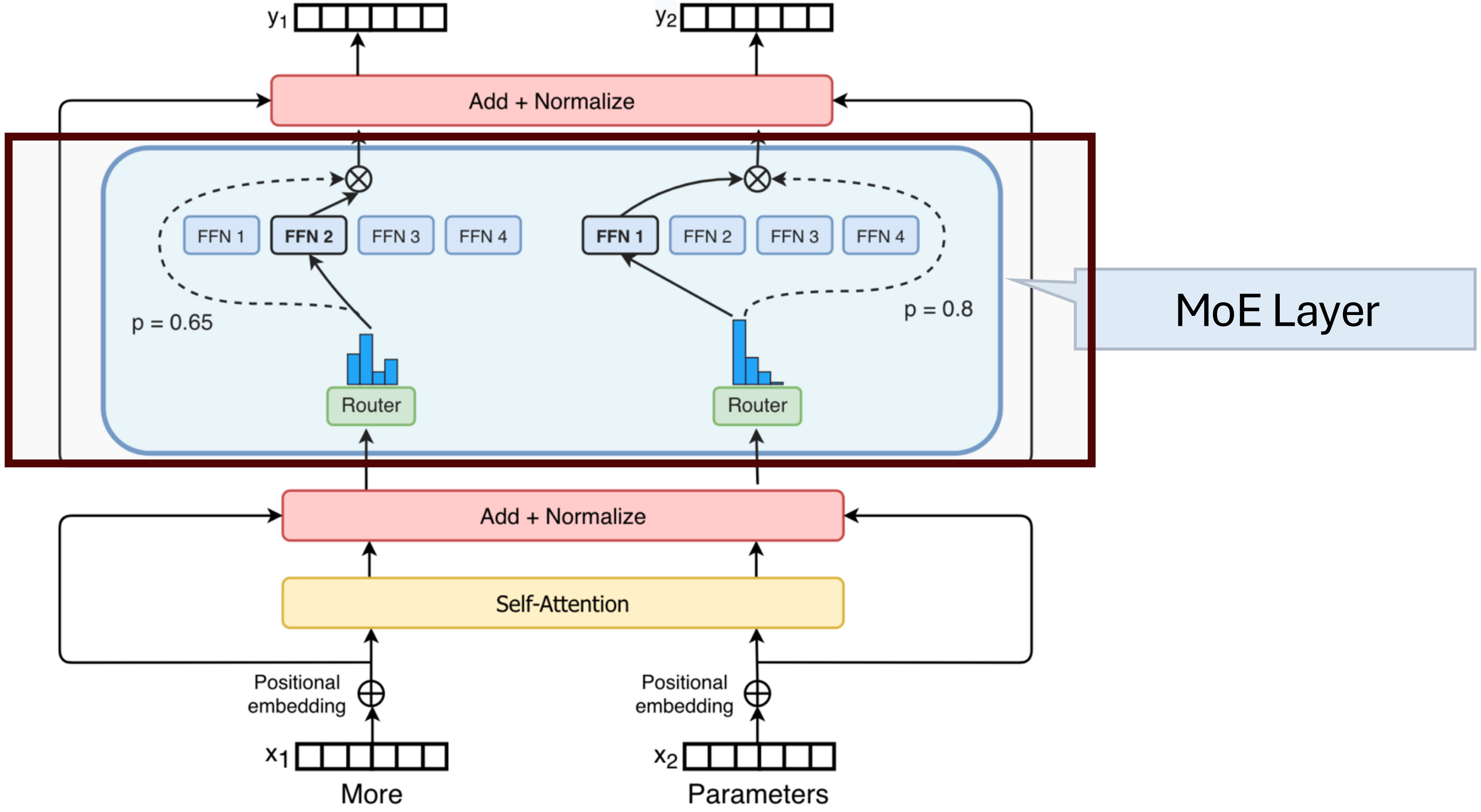
$$G_\sigma(x) = \text{Softmax}(x \cdot W_g)$$



Sparse MoE Layer – Greedy expert selection

$$G_\sigma(x) = \text{Softmax}(x \cdot W_g)$$

$$i* = argmax\ G_\sigma(x)$$

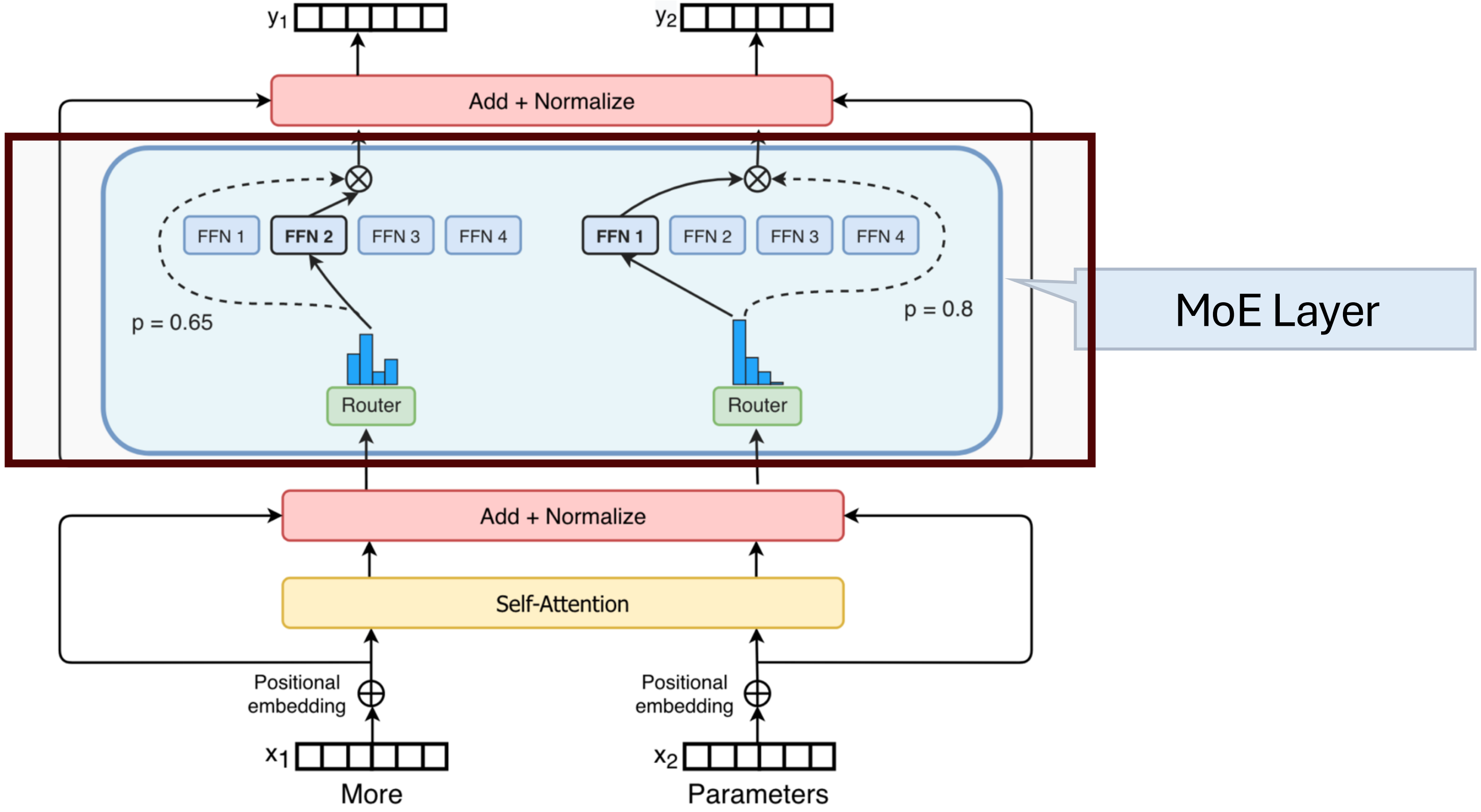


Sparse MoE Layer – Greedy expert selection

$$G_\sigma(x) = \text{Softmax}(x \cdot W_g)$$

$$i* = argmax\ G_\sigma(x)$$

$$y = G(x)_{i*}E_{i*}(x)$$



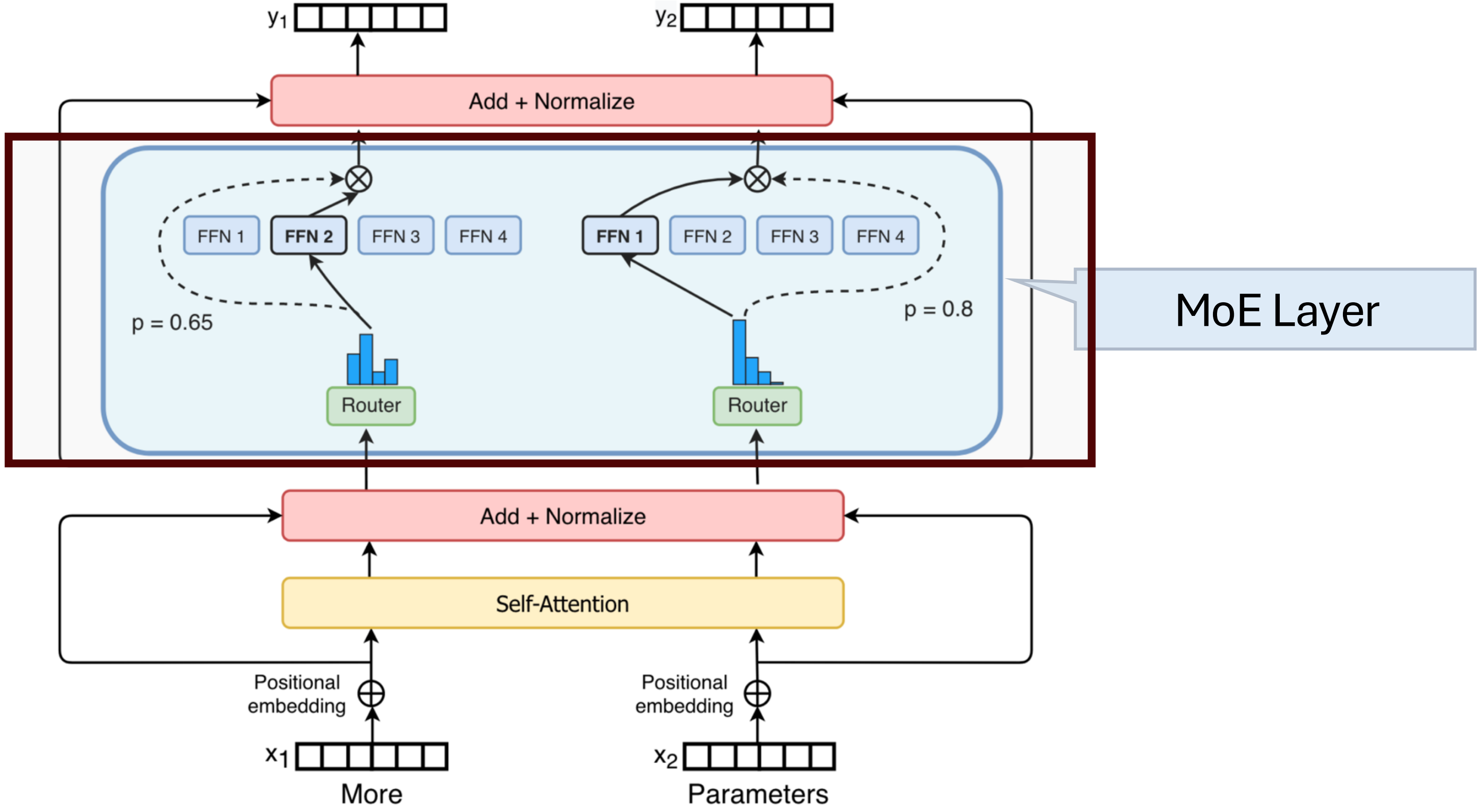
Sparse MoE Layer – Greedy expert selection

$$G_\sigma(x) = \mathrm{Softmax}(x \cdot W_g)$$

$$i* = argmax\ G_\sigma(x)$$

$$y = G(x)_{i*} E_{i*}(x)$$

$$\frac{\nabla L}{\nabla E_{i*}(x)} = G(x)_{i*} \frac{\nabla L}{\nabla y}$$



Sparse MoE Layer – Greedy expert selection

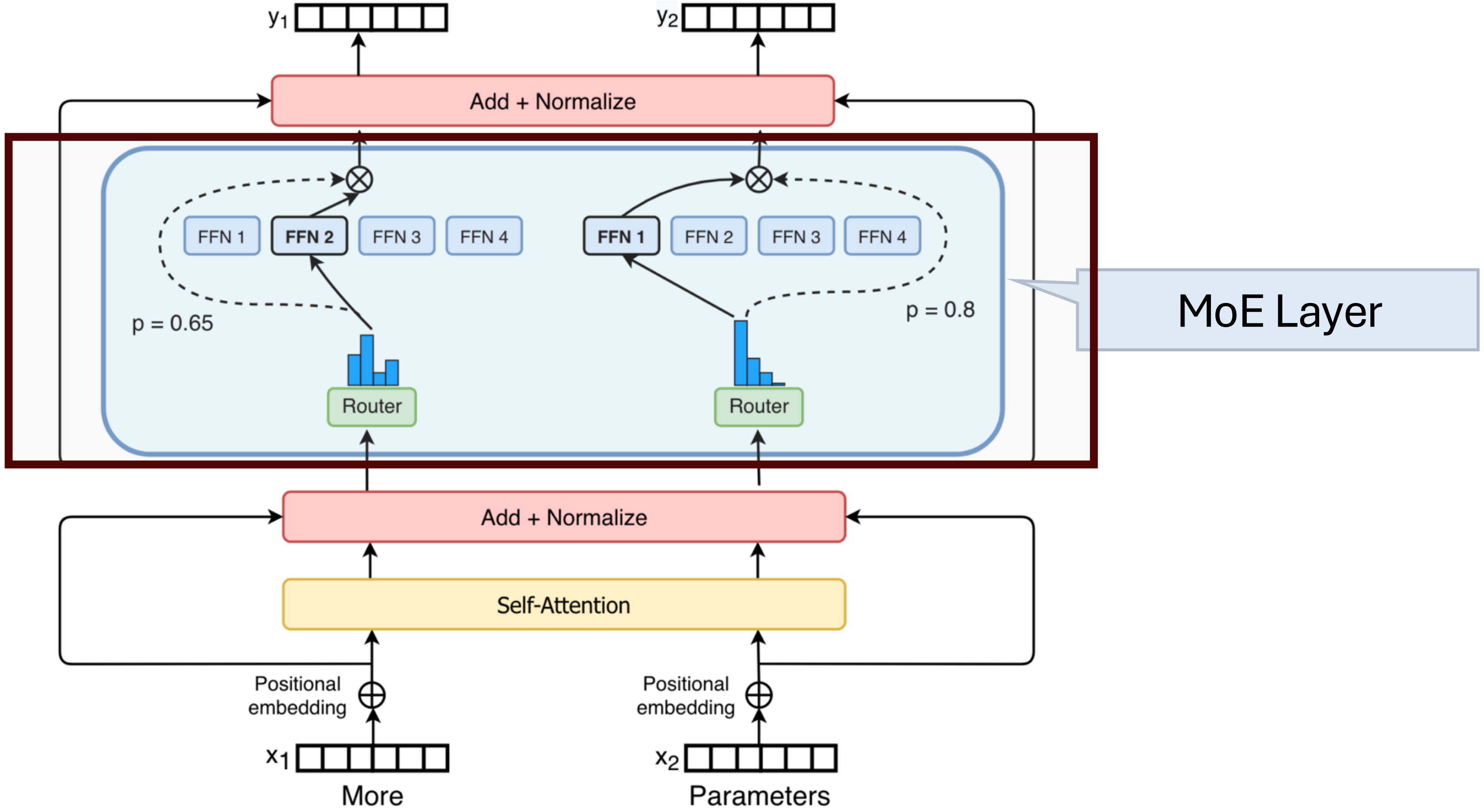
$$G_\sigma(x) = \mathrm{Softmax}(x \cdot W_g)$$

$$i* = argmax\ G_\sigma(x)$$

$$y = G(x)_{i*} E_{i*}(x)$$

$$\frac{\nabla L}{\nabla E_{i*}(x)} = G(x)_{i*} \frac{\nabla L}{\nabla y}$$

$$\frac{\nabla L}{\nabla G(x)_{i*}} = (E_{i*}(x))^T \frac{\nabla L}{\nabla y}$$



Sparse MoE Layer – Greedy expert selection

$$G_\sigma(x) = \text{Softmax}(x \cdot W_g)$$

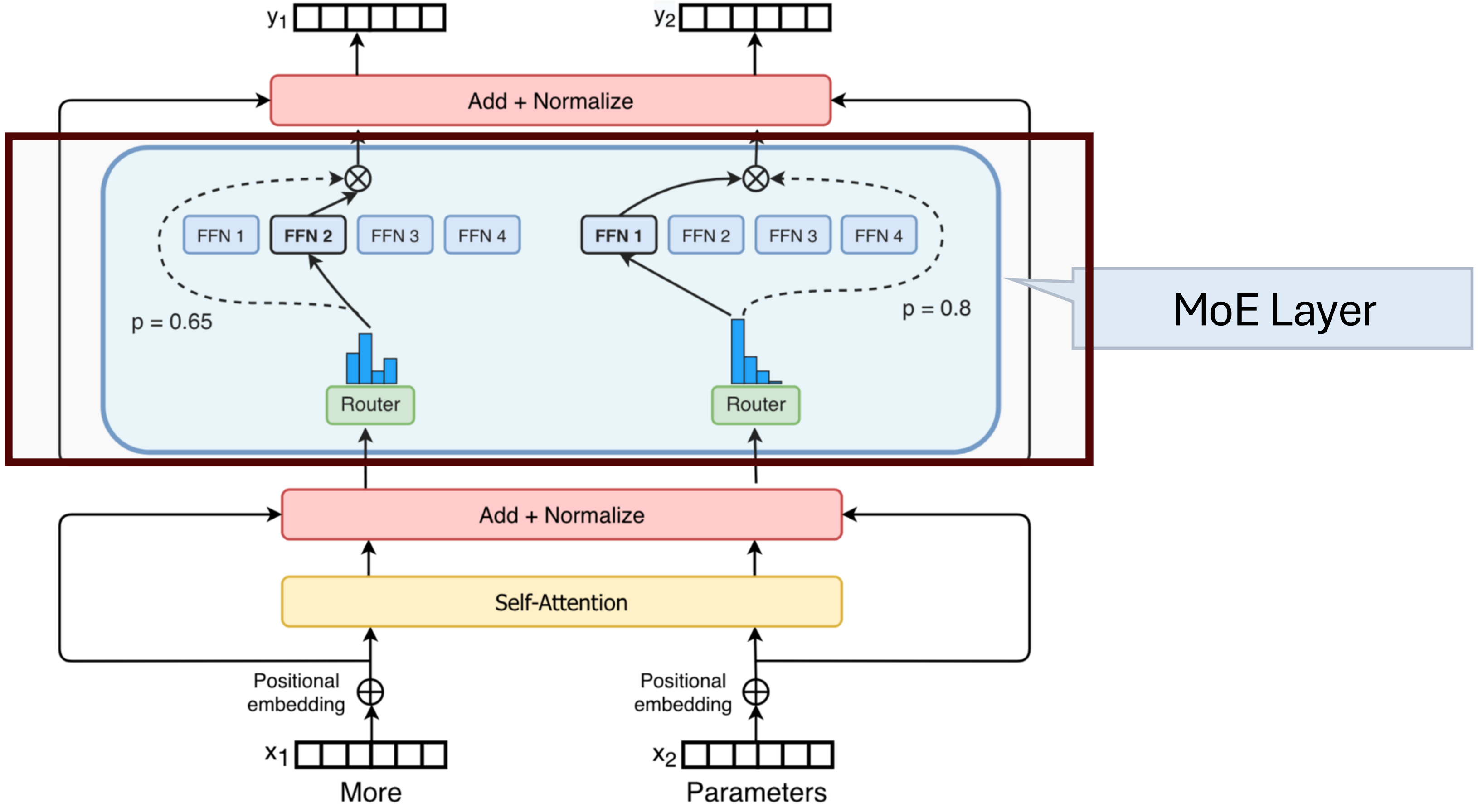
$$i* = argmax\ G_\sigma(x)$$

$$y = G(x)_{i*} E_{i*}(x)$$

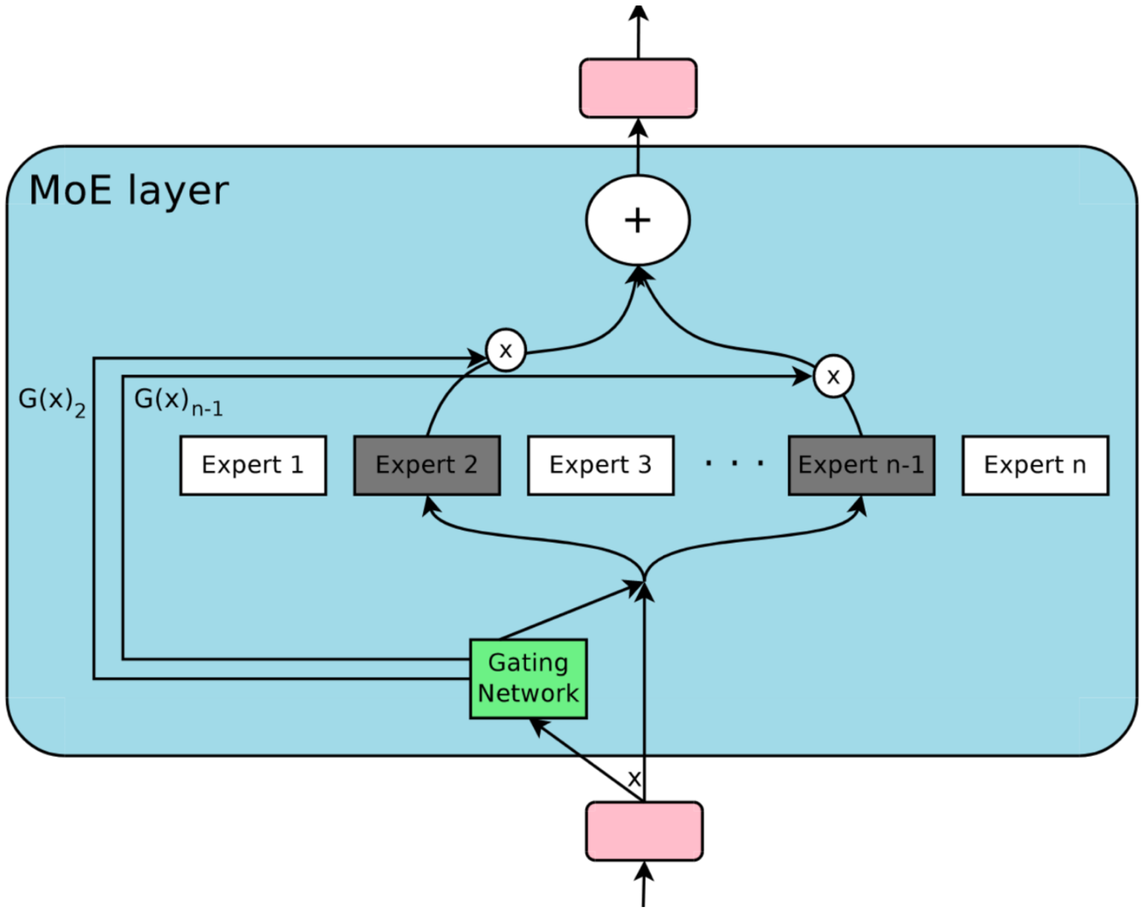
$$\frac{\nabla L}{\nabla E_{i*}(x)} = G(x)_{i*} \frac{\nabla L}{\nabla y}$$

$$\frac{\nabla L}{\nabla G(x)_{i*}} = (E_{i*}(x))^T \frac{\nabla L}{\nabla y}$$

$$\frac{\nabla L}{\nabla E_i(x)} = \mathbf{0}\,;\ \frac{\nabla L}{\nabla G(x)_i} = 0\ for\ i\ \neq i*$$



Sparse MoE Layer - Noisy Top-K Gating



Content credits: Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer

Sparse MoE Layer - Noisy Top-K Gating

$$H(x)_i = (x \cdot W_g)_i$$

i^{th} logit of the Gating Network

$$G_\sigma(x) = Softmax(H(x))$$

Sparse MoE Layer - Noisy Top-K Gating

$$H(x)_i = (x \cdot W_g)_i + StandardNormal()$$

$$G_\sigma(x) = Softmax(H(x))$$

Sparse MoE Layer - Noisy Top-K Gating

$$H(x)_i = (x \cdot W_g)_i + StandardNormal() \cdot Softplus((x \cdot W_{noise})_i)$$

$$G_\sigma(x) = Softmax(H(x))$$

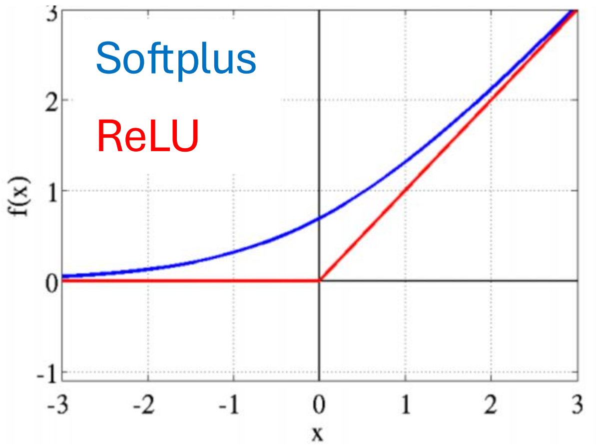
Sparse MoE Layer - Noisy Top-K Gating

$$H(x)_i = (x \cdot W_g)_i + StandardNormal() \cdot Softplus((x \cdot W_{noise})_i)$$

Learnable parameter
Mean

Learnable parameter
Std. Dev

$$\text{Softplus}(x) = \frac{1}{\beta} * \log(1 + \exp(\beta * x))$$



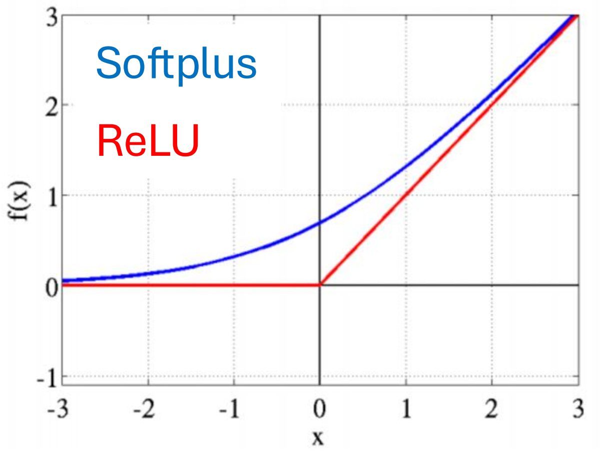
Sparse MoE Layer - Noisy Top-K Gating

$$H(x)_i = (x \cdot W_g)_i + StandardNormal() \cdot \boxed{Softplus}((x \cdot W_{noise})_i)$$

Learnable parameter
Mean

Learnable parameter
Std. Dev

Sparse MoE Layer - Noisy Top-K Gating



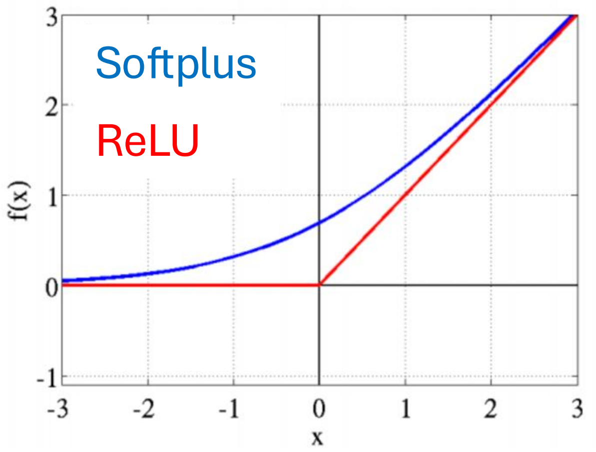
$$H(x)_i = (x \cdot W_g)_i + StandardNormal() \cdot Softplus((x \cdot W_{noise})_i)$$

Learnable parameter
Mean

Learnable parameter
Std. Dev

$$KeepTopK(v,k)_i = \begin{cases} v_i & \text{if } v_i \text{ is in the top } k \text{ elements of } v. \\ -\infty & \text{otherwise.} \end{cases}$$

Sparse MoE Layer - Noisy Top-K Gating



$$H(x)_i = (x \cdot W_g)_i + StandardNormal() \cdot Softplus((x \cdot W_{noise})_i)$$

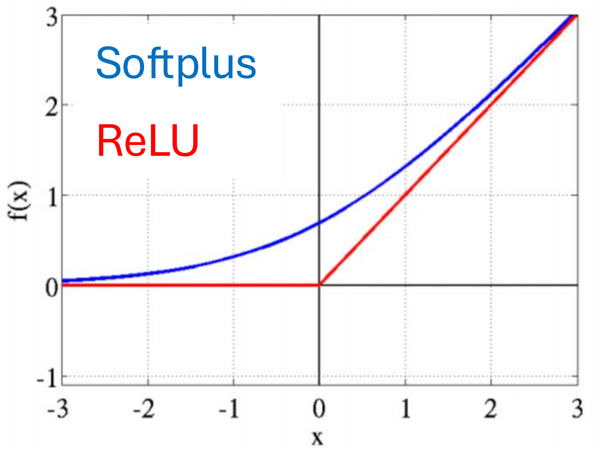
Learnable parameter
Mean

Learnable parameter
Std. Dev

$$KeepTopK(v,k)_i = \begin{cases} v_i & \text{if } v_i \text{ is in the top } k \text{ elements of } v. \\ -\infty & \text{otherwise.} \end{cases}$$

Ensures 0 probability after softmax

Sparse MoE Layer - Noisy Top-K Gating



$$H(x)_i = (x \cdot W_g)_i + StandardNormal() \cdot Softplus((x \cdot W_{noise})_i)$$

Learnable parameter
Mean

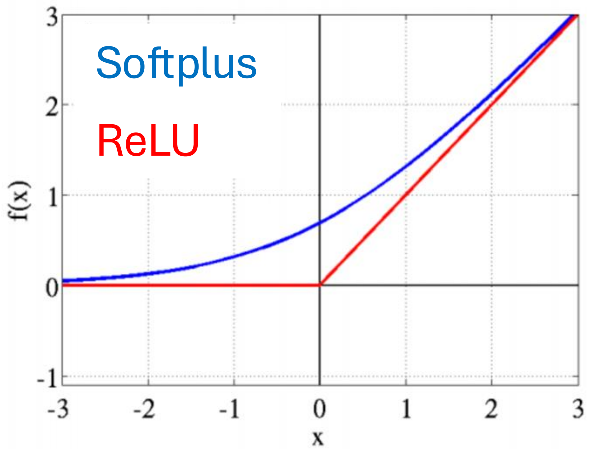
Learnable parameter
Std. Dev

$$KeepTopK(v,k)_i = \begin{cases} v_i & \text{if } v_i \text{ is in the top } k \text{ elements of } v. \\ -\infty & \text{otherwise.} \end{cases}$$

Ensures 0 probability after softmax

$$H(x)$$

Sparse MoE Layer - Noisy Top-K Gating



$$H(x)_i = (x \cdot W_g)_i + StandardNormal() \cdot Softplus((x \cdot W_{noise})_i)$$

Learnable parameter
Mean

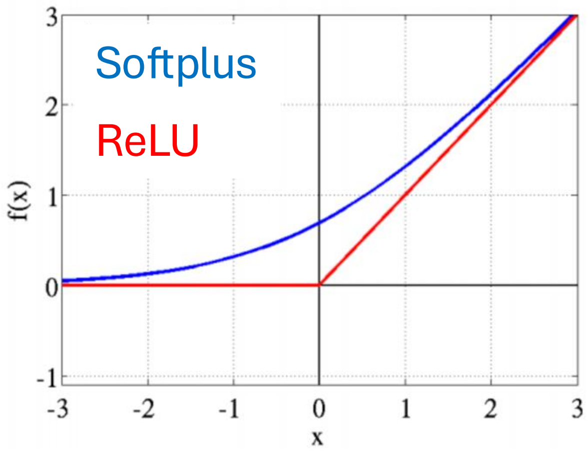
Learnable parameter
Std. Dev

$$KeepTopK(v,k)_i = \begin{cases} v_i & \text{if } v_i \text{ is in the top } k \text{ elements of } v. \\ -\infty & \text{otherwise.} \end{cases}$$

Ensures 0 probability after softmax

$$KeepTopK(H(x), k)$$

Sparse MoE Layer - Noisy Top-K Gating



$$H(x)_i = (x \cdot W_g)_i + StandardNormal() \cdot Softplus((x \cdot W_{noise})_i)$$

Learnable parameter
Mean

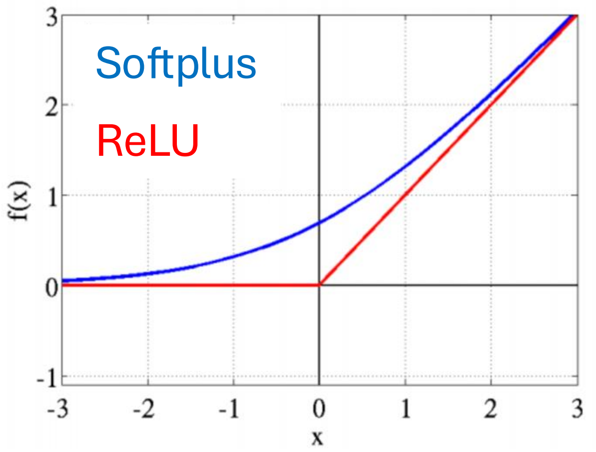
Learnable parameter
Std. Dev

$$KeepTopK(v,k)_i = \begin{cases} v_i & \text{if } v_i \text{ is in the top } k \text{ elements of } v. \\ -\infty & \text{otherwise.} \end{cases}$$

Ensures 0 probability after softmax

$$G(x) = Softmax(KeepTopK(H(x),k))$$

Sparse MoE Layer - Noisy Top-K Gating



$$H(x)_i = (x \cdot W_g)_i + StandardNormal() \cdot Softplus((x \cdot W_{noise})_i)$$

Learnable parameter
Mean

Learnable parameter
Std. Dev

$$KeepTopK(v, k)_i = \begin{cases} v_i & \text{if } v_i \text{ is in the top } k \text{ elements of } v. \\ -\infty & \text{otherwise.} \end{cases}$$

Ensures 0 probability after softmax

$$G(x) = Softmax(KeepTopK(H(x), k))$$

Only *k* non-zero elements; add up to *1*

Why do experts not collapse in practice?

❖ All experts have:

- Same architecture
- Same initialization scheme
- Trained with same optimizer

❖ So why don't they collapse? I.e.

Expert 1 = Expert 2 = ... = Expert n -1 = Expert n ?

Why do experts not collapse in practice?

Towards Understanding the Mixture-of-Experts Layer in Deep Learning

Zixiang Chen
Department of Computer Science
University of California, Los Angeles
Los Angeles, CA 90095, USA
chenzx19@cs.ucla.edu

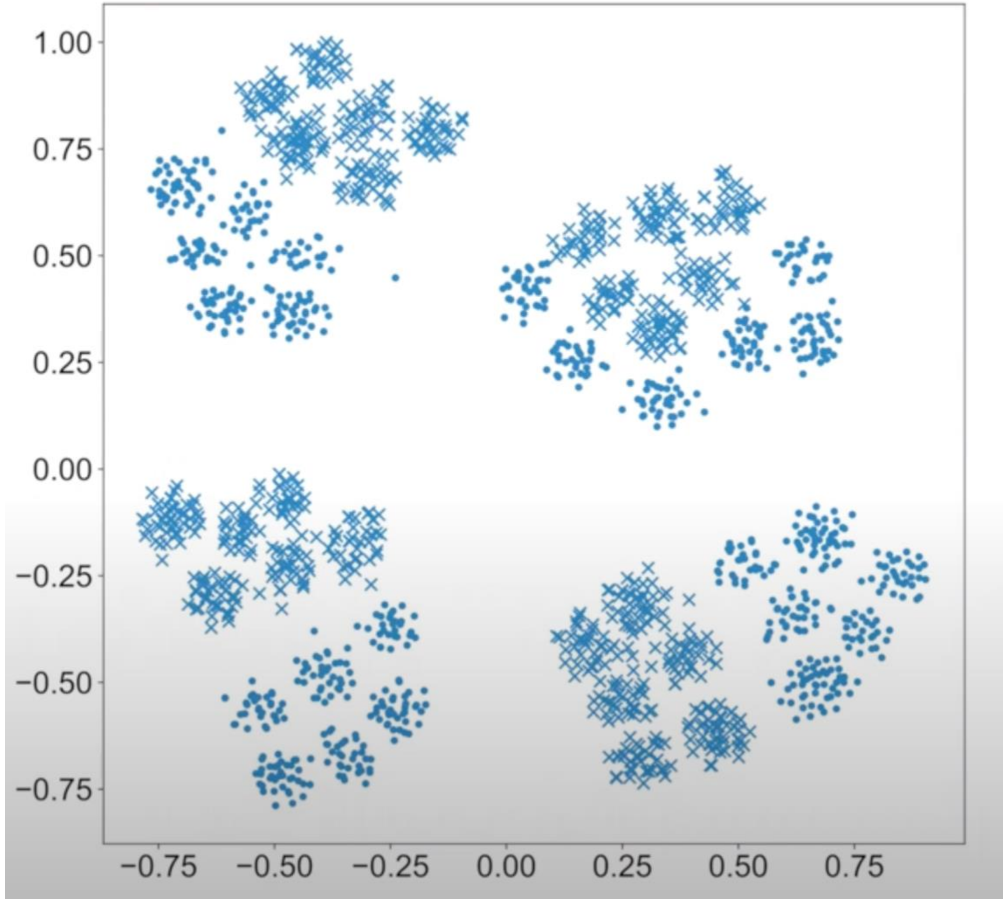
Yihe Deng
Department of Computer Science
University of California, Los Angeles
Los Angeles, CA 90095, USA
yihedeng@cs.ucla.edu

Yue Wu
Department of Computer Science
University of California, Los Angeles
Los Angeles, CA 90095, USA
ywu@cs.ucla.edu

Quanquan Gu
Department of Computer Science
University of California, Los Angeles
Los Angeles, CA 90095, USA
qgu@cs.ucla.edu

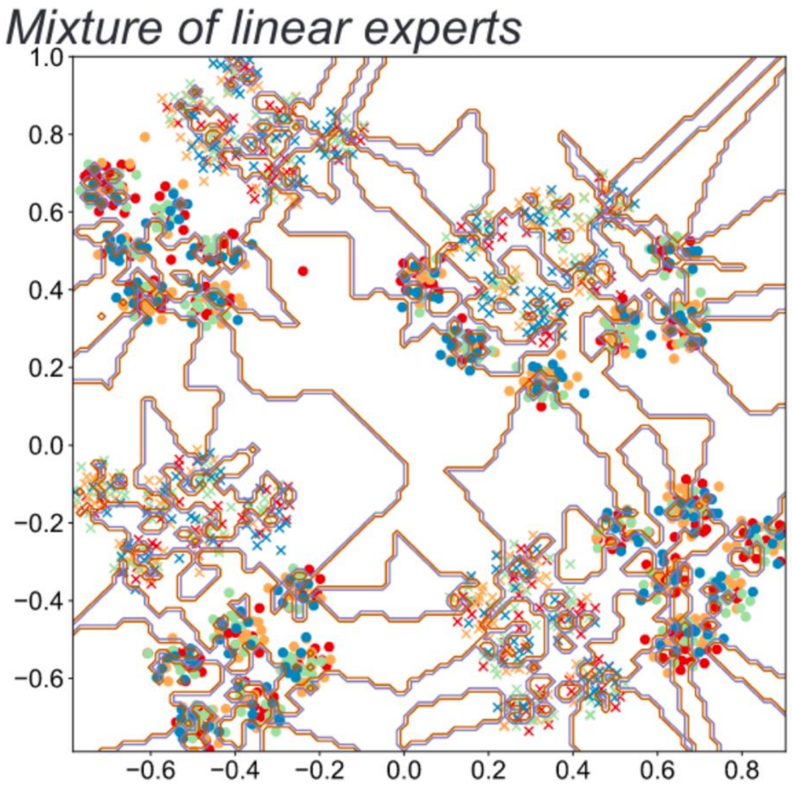
Yuanzhi Li
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA
yuanzhil@andrew.cmu.edu

Why do experts not collapse in practice?



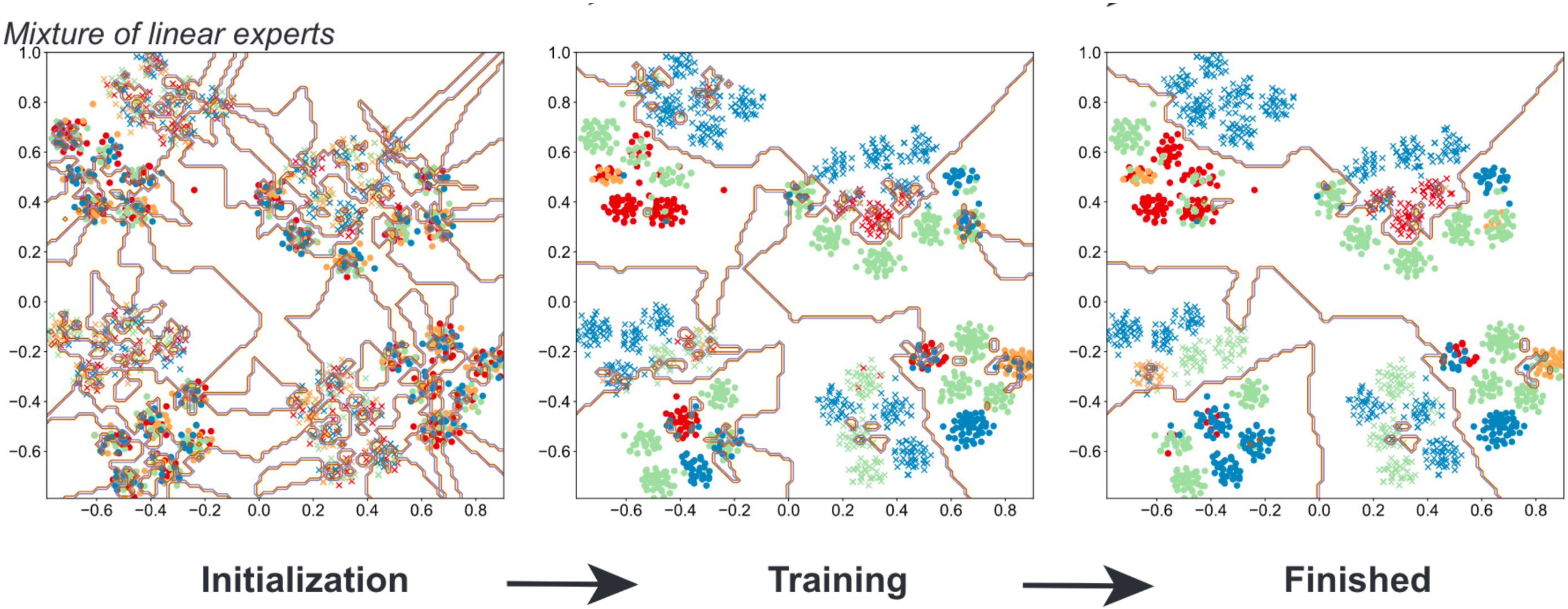
- ✓ Synthetically generated 50 dim. data
- ✓ Visualization in 2-D space
- ✓ 4 clusters
- ✓ Each cluster is linearly separable
- ✓ Ideally, a mixture of 4 linear experts sufficient for classification

Why do experts not collapse in practice?

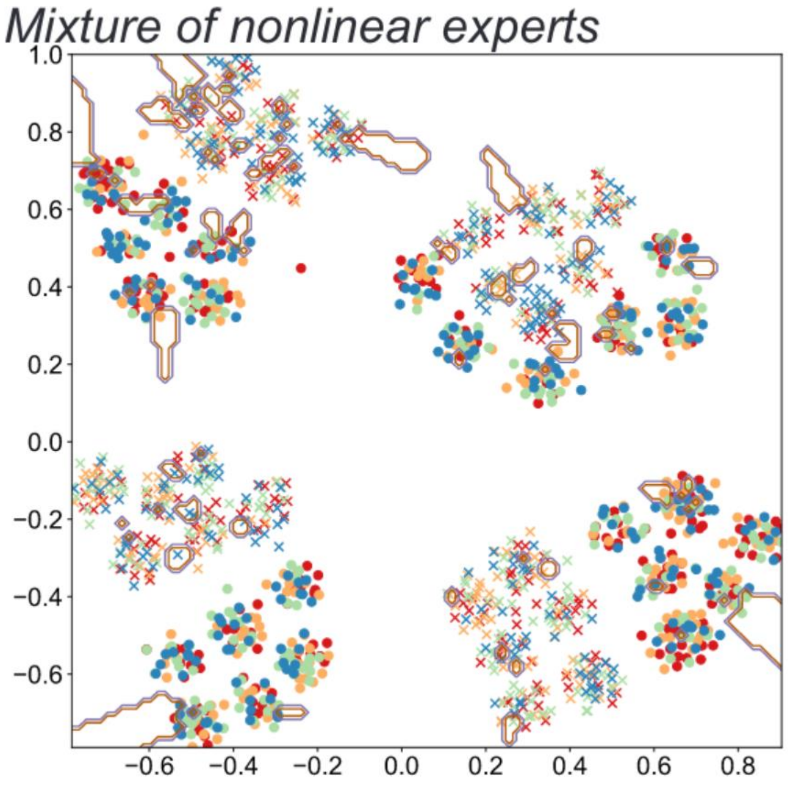


Initialization

Why do experts not collapse in practice?

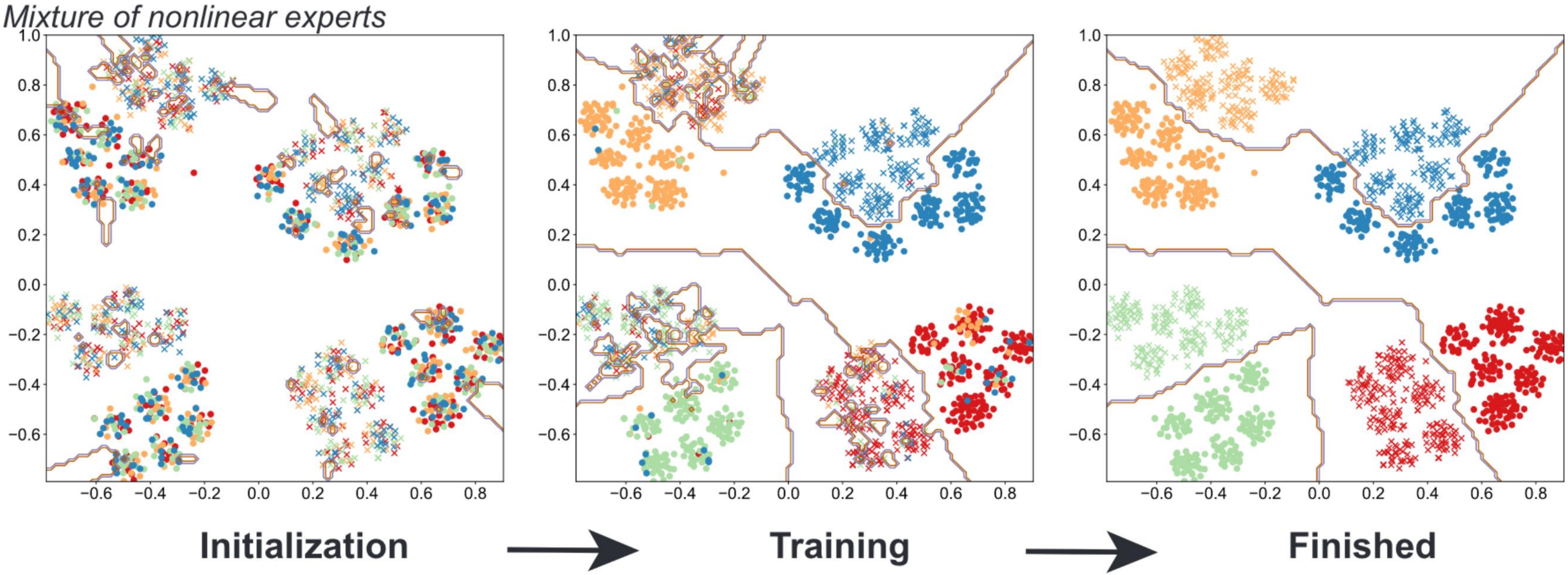


Why do experts not collapse in practice?

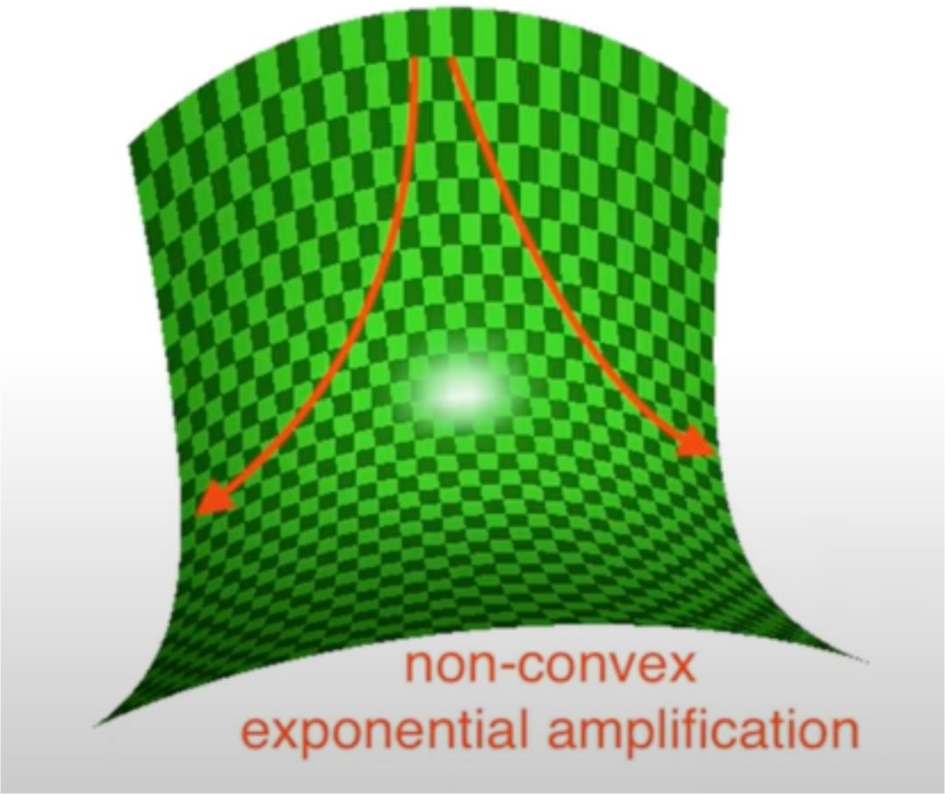


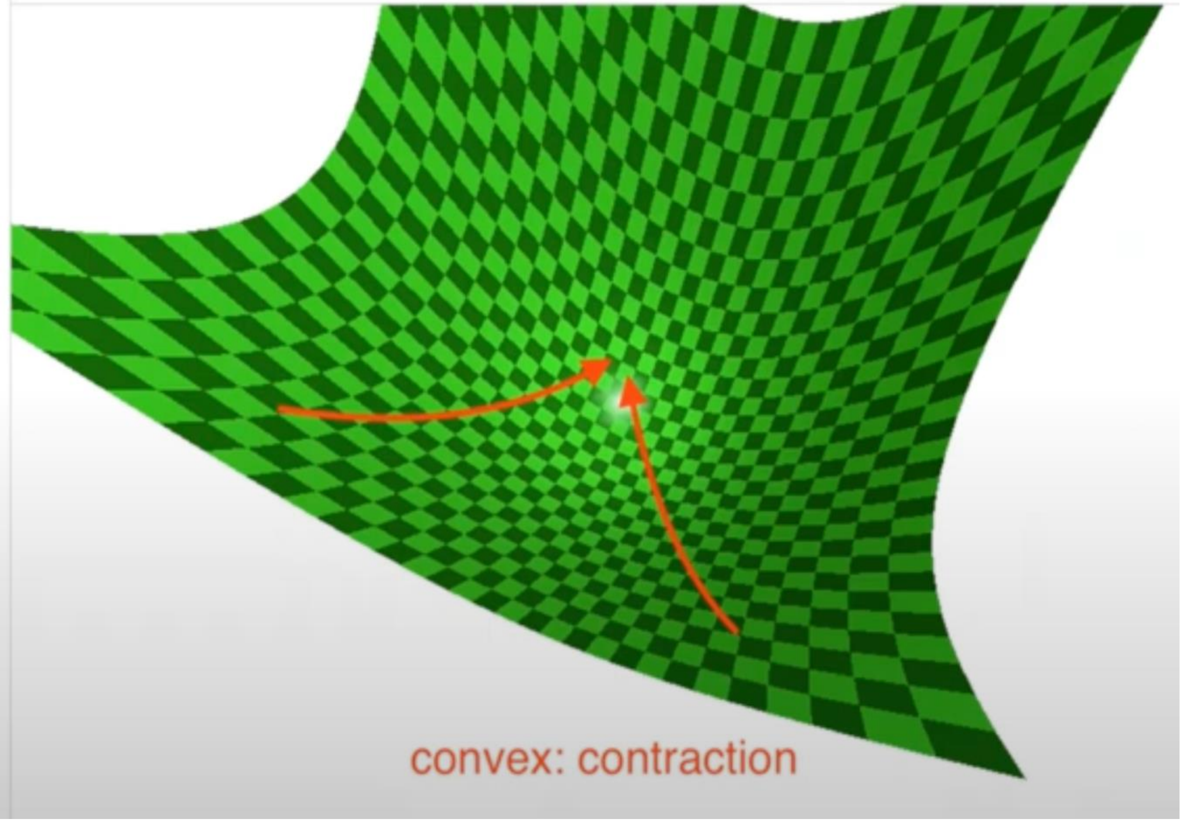
Initialization

Why do experts not collapse in practice?



Why do experts not collapse in practice?





Learning dynamics of experts and router

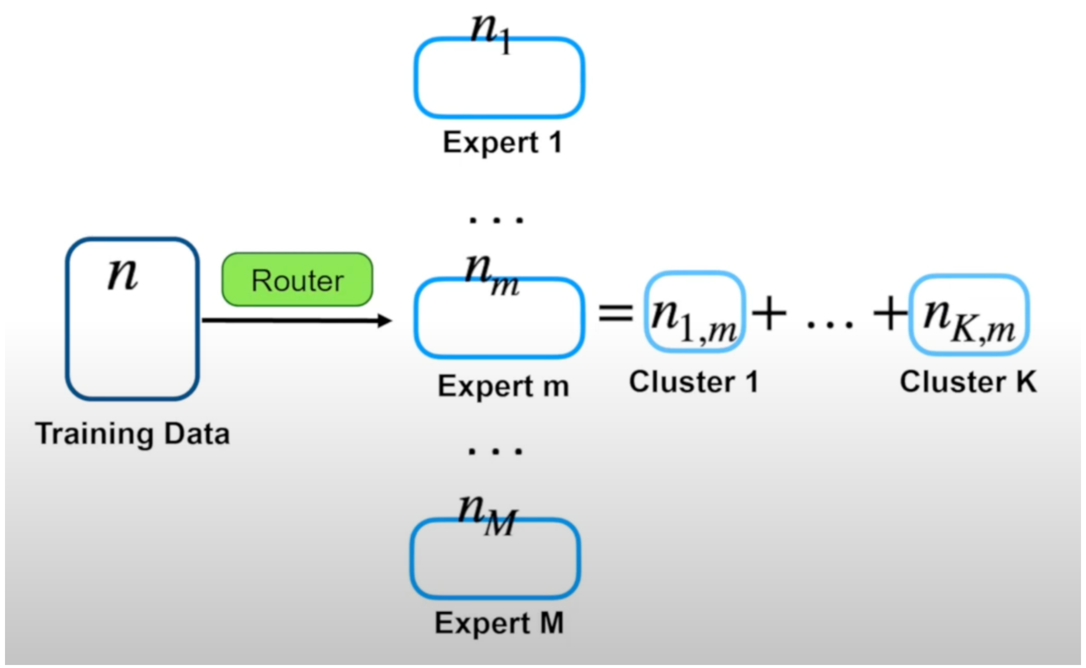
- Exploration stage:
  - experts diverge; router nearly untrained
- Router learning stage:
  - router learns to dispatch

Learning dynamics of experts and router

- **Exploration stage:**
  - experts diverge; router nearly untrained

- **Router learning stage:**
  - router learns to dispatch

$n_{k,m}$: # of samples from Cluster k routed to Expert m
n_m : # of samples routed to Expert m $= \sum_{k=1}^{K} n_{k,m}$
n : # of total samples $= \sum_{m=1}^{M} n_m$



Learning dynamics of experts and router

- Exploration stage:
  - experts diverge; router nearly untrained

- Router learning stage:
  - router learns to dispatch

$$\text{entropy} = -\sum_{m=1, n_m \neq 0}^{M} \frac{n_m}{n} \sum_{k=1}^{K} \frac{n_{k,m}}{n_m} \cdot \log\left(\frac{n_{k,m}}{n_m}\right)$$

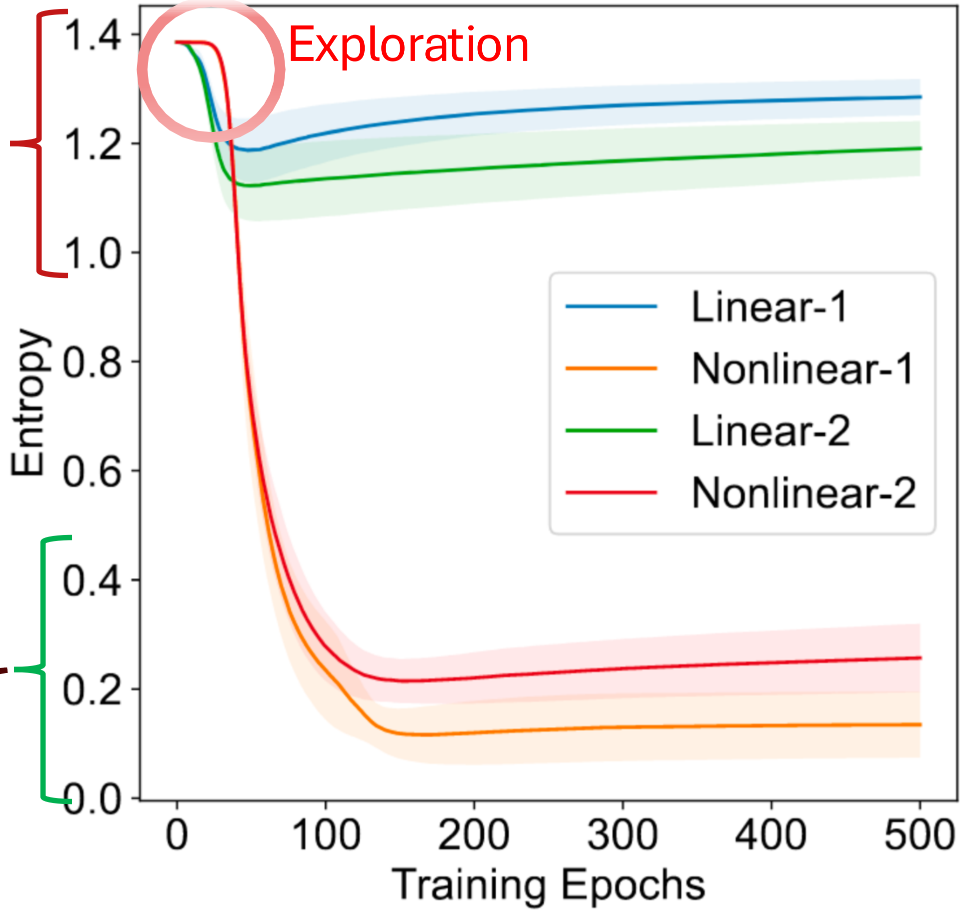
$n_{k,m}$: *# of samples from Cluster k routed to Expert m*
n_m : *# of samples routed to Expert m* $= \sum_{k=1}^{K} n_{k,m}$
n : *# of total samples* $= \sum_{m=1}^{M} n_m$

Learning dynamics of experts and router

$$\text{entropy} = -\sum_{m=1, n_m \neq 0}^{M} \frac{n_m}{n} \sum_{k=1}^{K} \frac{n_{k,m}}{n_m} \cdot \log\left(\frac{n_{k,m}}{n_m}\right)$$

Entropy is high if an input from cluster *k* is routed uniformly to all the experts

Entropy is low if an input from cluster k is routed to one expert



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)