THE UNIVERSITY
of NORTH CAROLINA
at CHAPEL HILL

SOFT MERGING OF EXPERTS WITH ADAPTIVE ROUTING

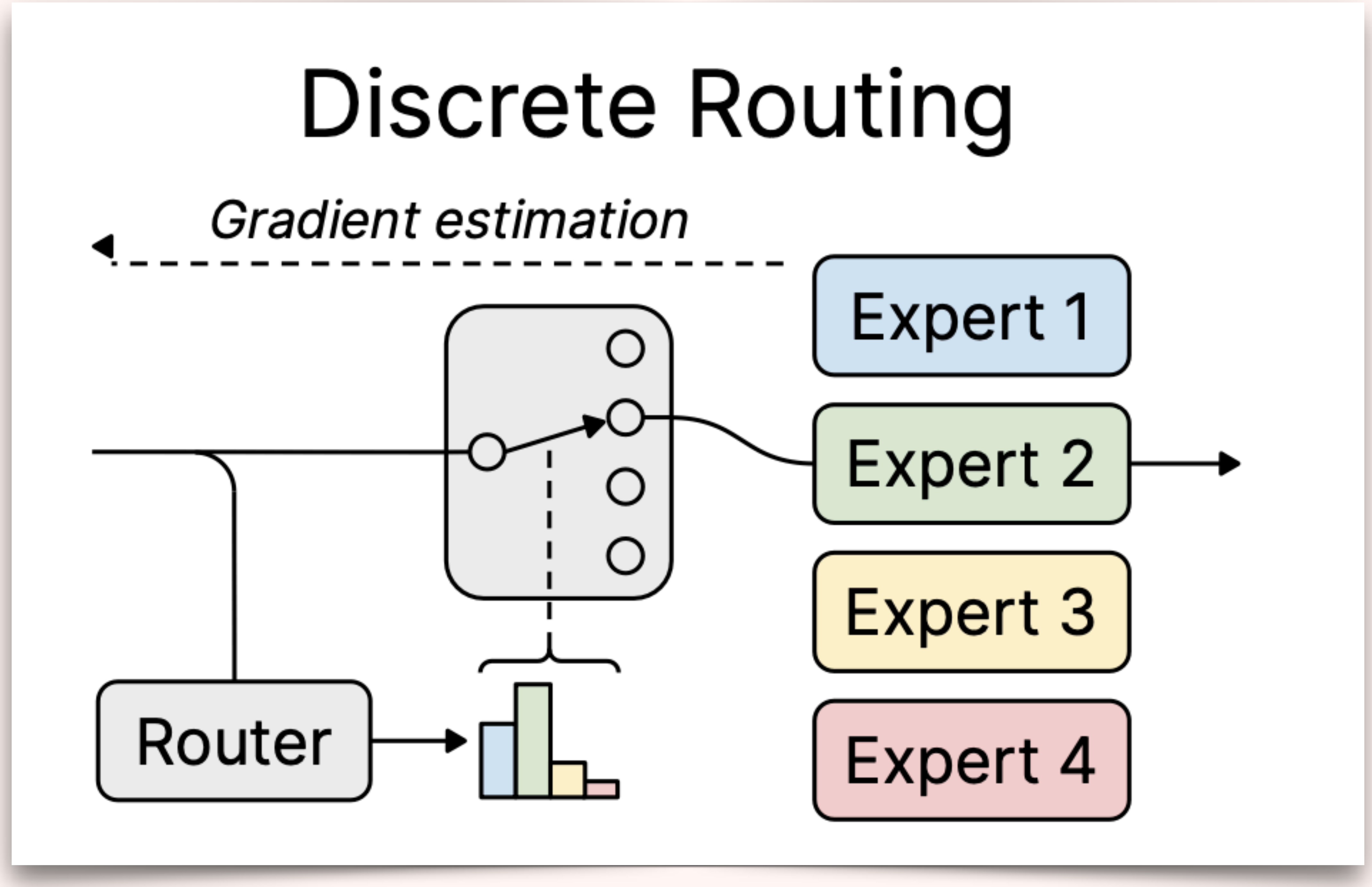
Mohammed Muqeeth, Haokun Liu, Colin Raffel

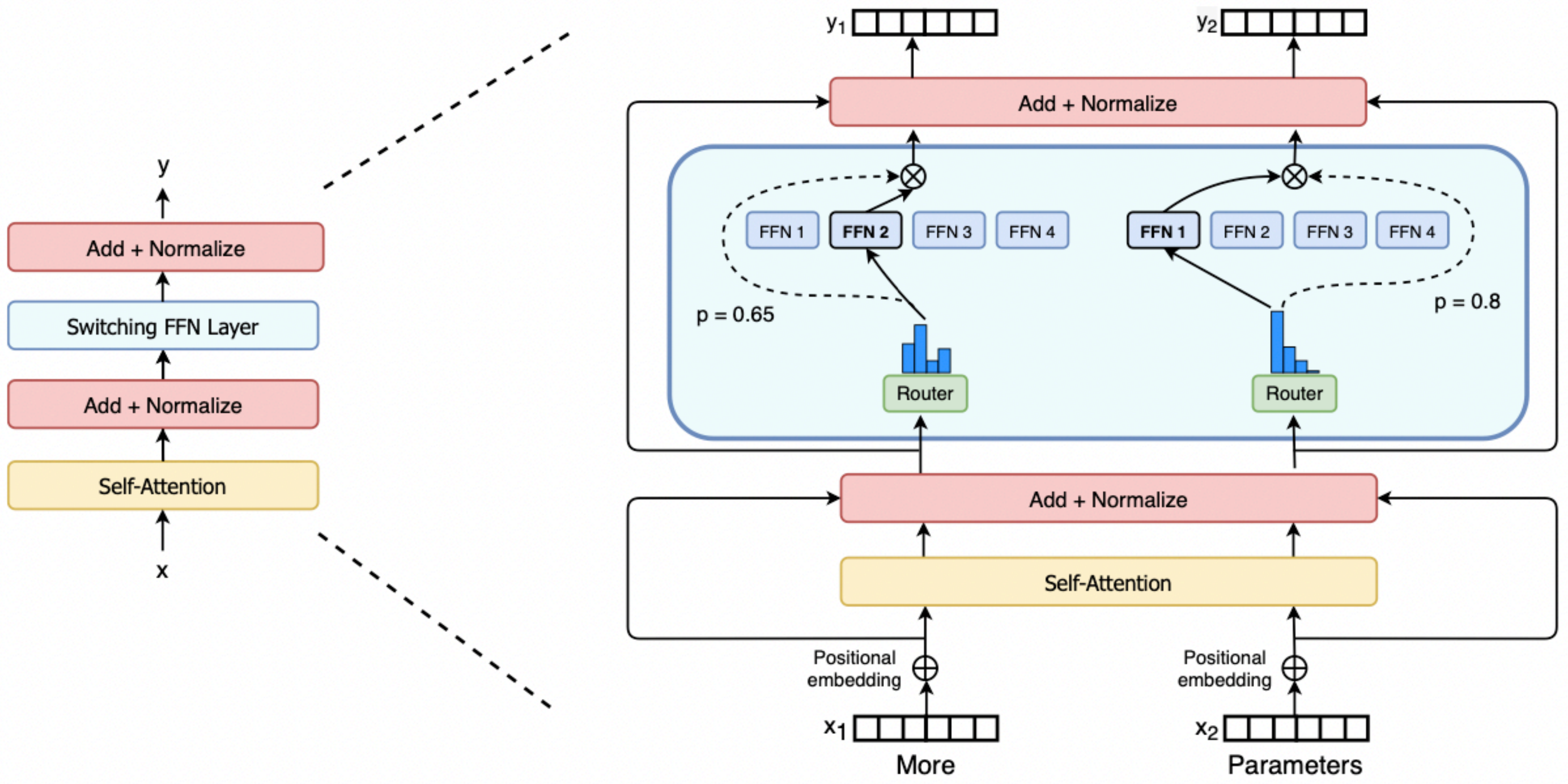
TYPICAL NEURAL NETWORKS

- Computation $\propto$ Number of parameters
- As models scale, computation becomes prohibitively expensive
- Suffer from task interference

MODELS WITH CONDITIONAL COMPUTATION

- Introduce modularity through learned routing
  - decouple computation and number of parameters
  - specialization to different inputs



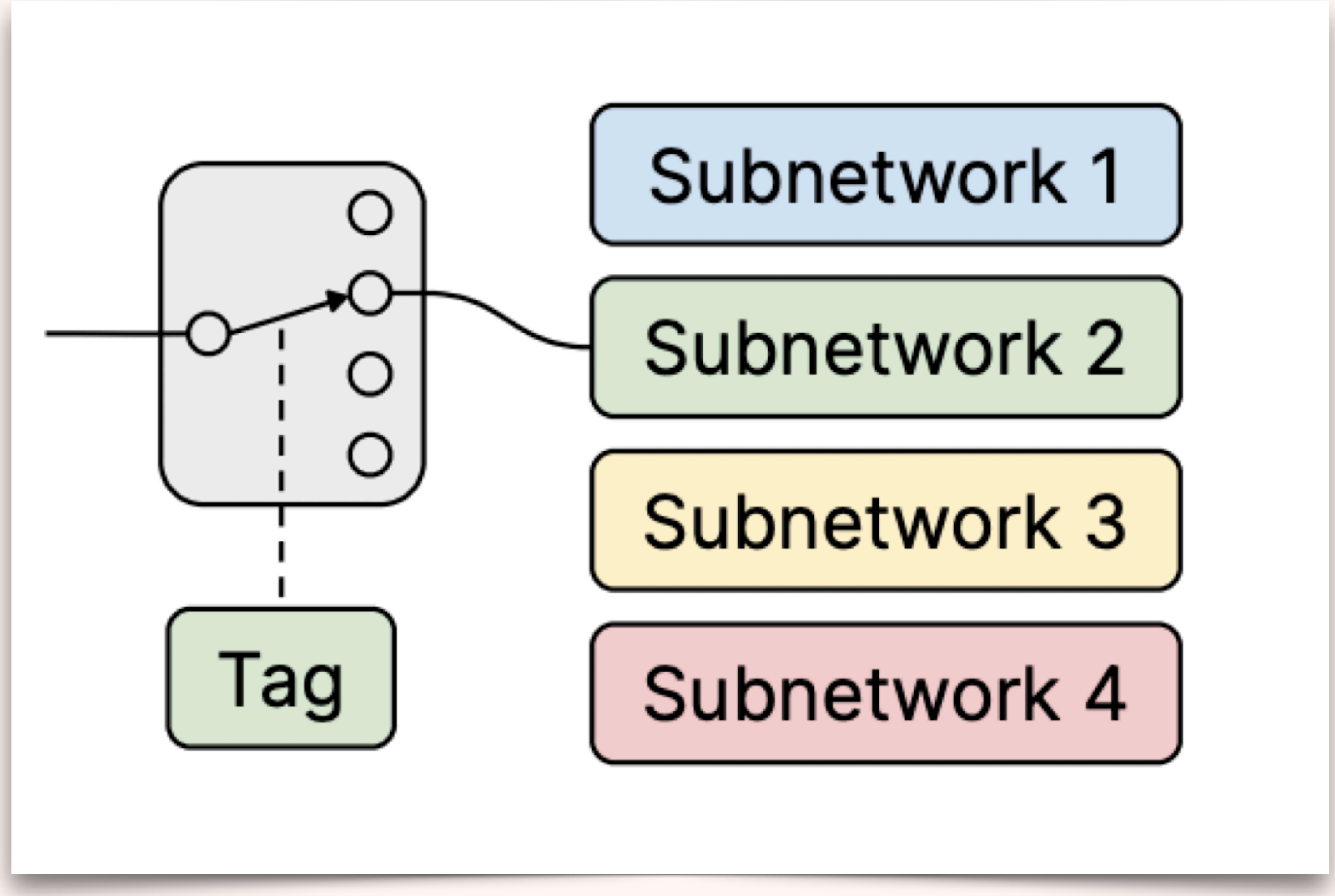

y
Add + Normalize
Switching FFN Layer
Add + Normalize
Self-Attention
x
y1
y2
Add + Normalize
FFN 1
FFN 2
FFN 3
FFN 4
FFN 1
FFN 2
FFN 3
FFN 4
p = 0.65
p = 0.8
Router
Router
Add + Normalize
Self-Attention
Positional embedding
Positional embedding
x1
x2
More
Parameters

ARE MODELS WITH CONDITIONAL COMPUTATION HOLDING THE PROMISE?

- Learned routing typically underperforms heuristic ones
  - In machine translation, Kudugunta et al., 2021, heuristic task level routing outperforms learned routing
  - In Downstream GLUE, Switch Transformer 3.4B (86.7)< T5 large 740M (87.8)
  - Roller et al. achieve comparable performance of learned routing with hash routing

ROUTING VIA HEURISTICS

- Tags associated with input examples
  - Task/ Dataset
  - Domain
- Hash
- Monolithic (fixed for all examples)



LEARNED ROUTING VIA GRADIENT ESTIMATORS

B **: expert routing block**

N **: total number of experts**

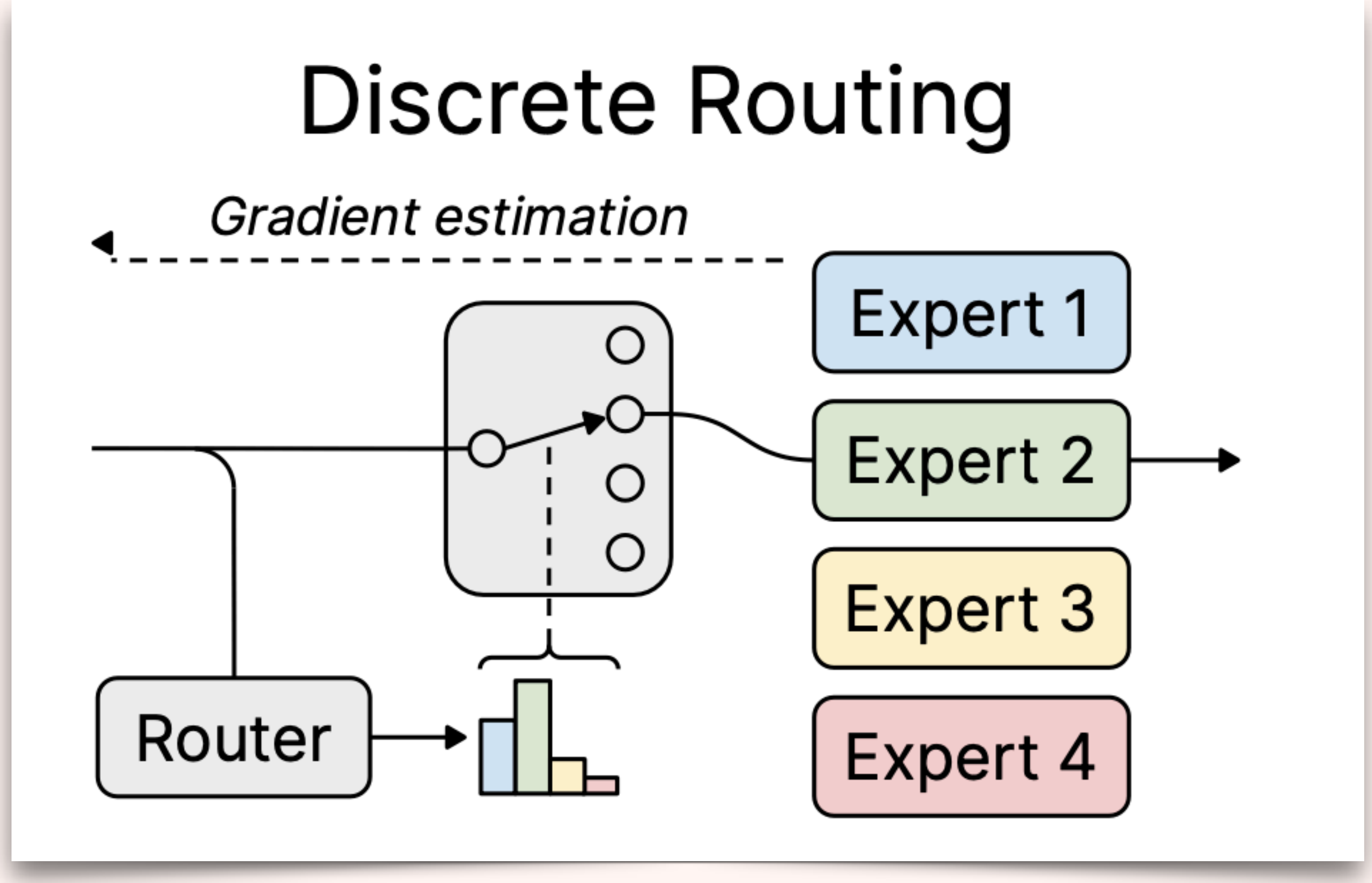
$\{f_1(\,.\,,\theta_1), f_2(\,.\,,\theta_2), \ldots f_N(\,.\,,\theta_N)\}$

u **: activation for the example x at current layer**

v **: activation at same layer or a different layer**

$P(v)$ **: router probability distribution**

i **: selected expert**



TOP-K

B : **expert routing block**

N : **total number of experts**

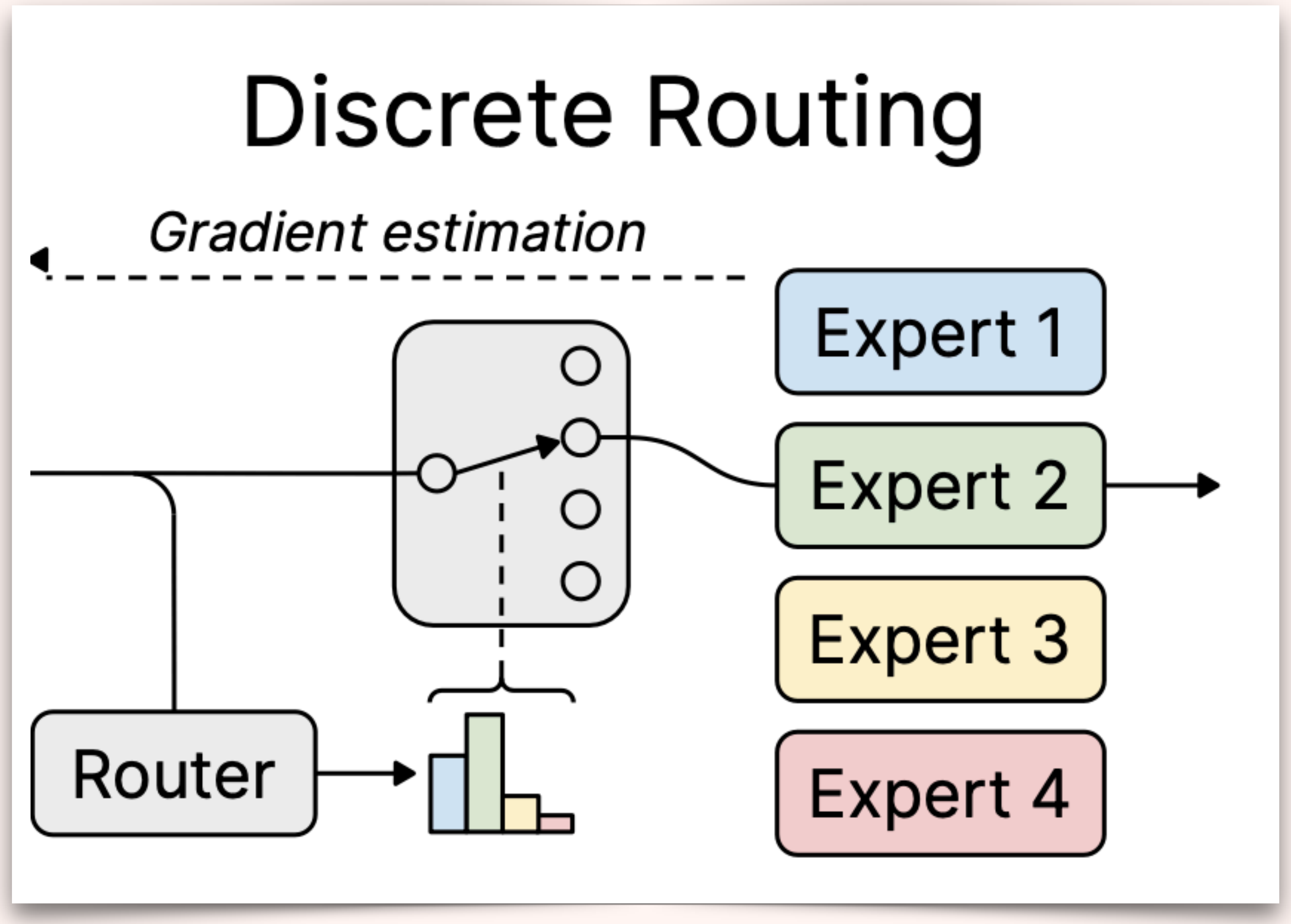
$\{f_1(\,.\,,\theta_1), f_2(\,.\,,\theta_2), \ldots f_N(\,.\,,\theta_N)\}$

u : **activation for the example x at current layer**

v : **activation at same layer or a different layer**

$P(v)$: **router probability distribution**

i : **selected expert**



$i = \text{argmax}_i(P(v))$

Output of the B is $P(v)_i \; f_i(u, \theta_i)$

ST-GUMBEL

B **: expert routing block**

N **: total number of experts**

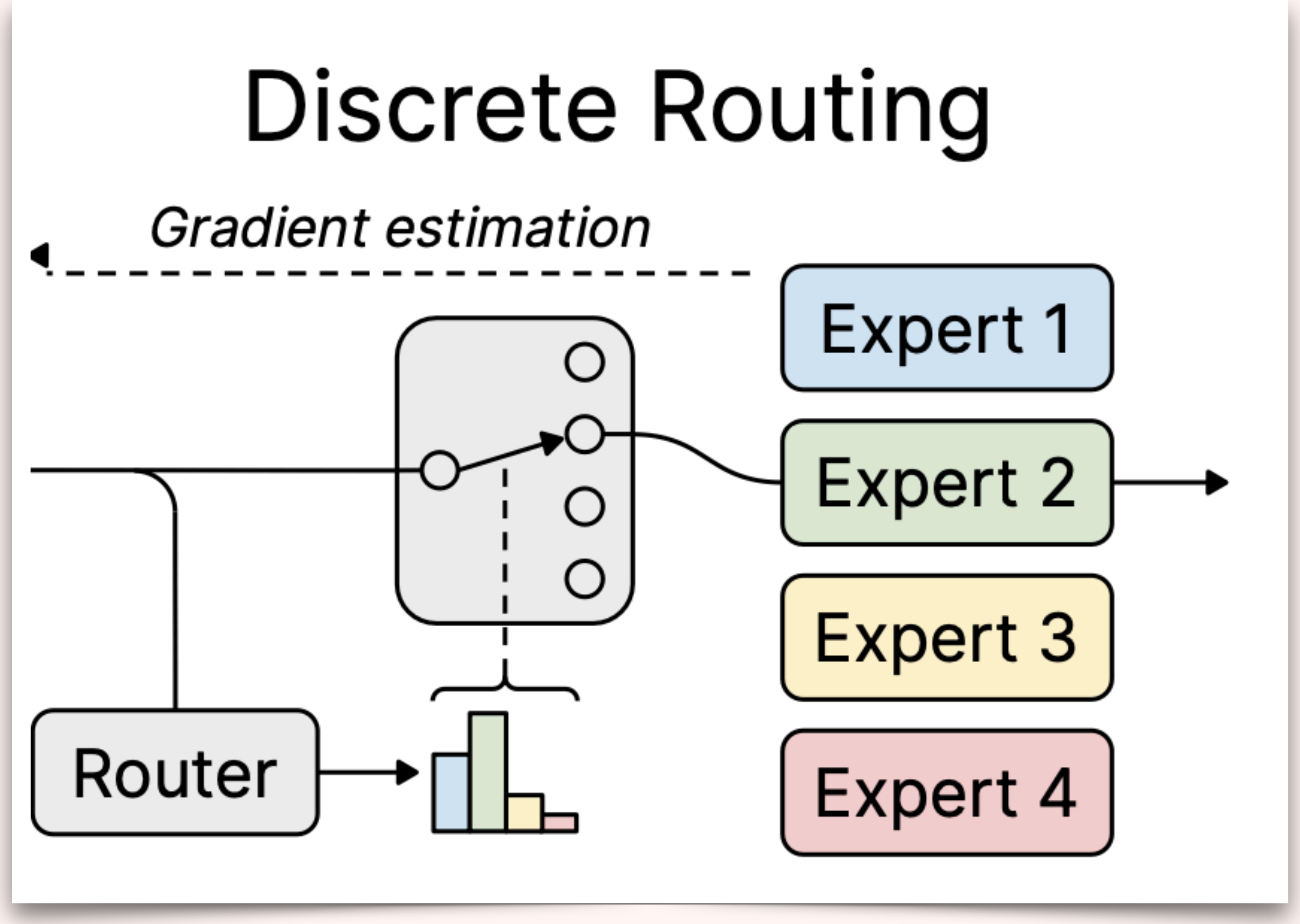
$\{f_1(\,.\,,\theta_1), f_2(\,.\,,\theta_2), \ldots f_N(\,.\,,\theta_N)\}$

u **: activation for the example** x **at current layer**

v **: activation at same layer or a different layer**

$P(v)$ **: router probability distribution**

i **: selected expert**



$$\hat{P}(v)_i = \frac{\exp((\log(P(v)_i) + g_i)/\tau)}{\sum_{j=1}^{N} \exp((\log(P(v)_i) + g_i)/\tau)}$$

$$g_i \sim Gumbel(0,1)$$

$$i = \text{argmax}_i(\hat{P}(v))$$

Output of the B **is** $(1 - \textbf{sg}[\hat{P}(v)_i] + \hat{P}(v)_i)\ \ f_i(u, \theta_i)$

REINFORCE

B : **expert routing block**

N : **total number of experts**

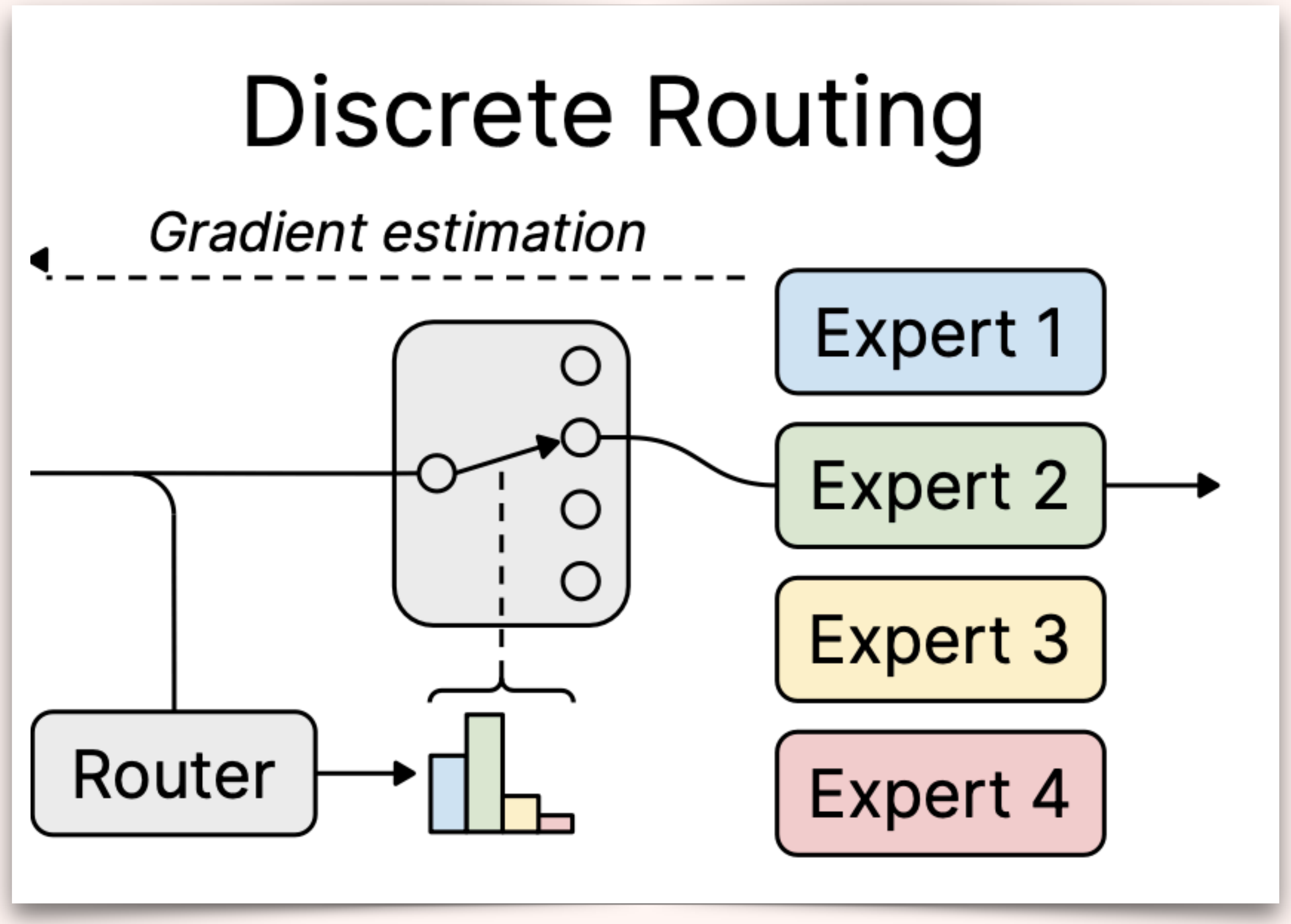
$\{f_1(\,.\,,\theta_1), f_2(\,.\,,\theta_2), \ldots f_N(\,.\,,\theta_N)\}$

u : **activation for the example x at current layer**

v : **activation at same layer or a different layer**

$P(v)$: **router probability distribution**

i : **selected expert**



$$J = \mathbb{E}_{i\sim P(v)}\ \alpha \log P(v)_i\ (r - b)$$

$$+\beta P(v) \log P(v) - \gamma L_{\textbf{Huber}}(r, b)$$

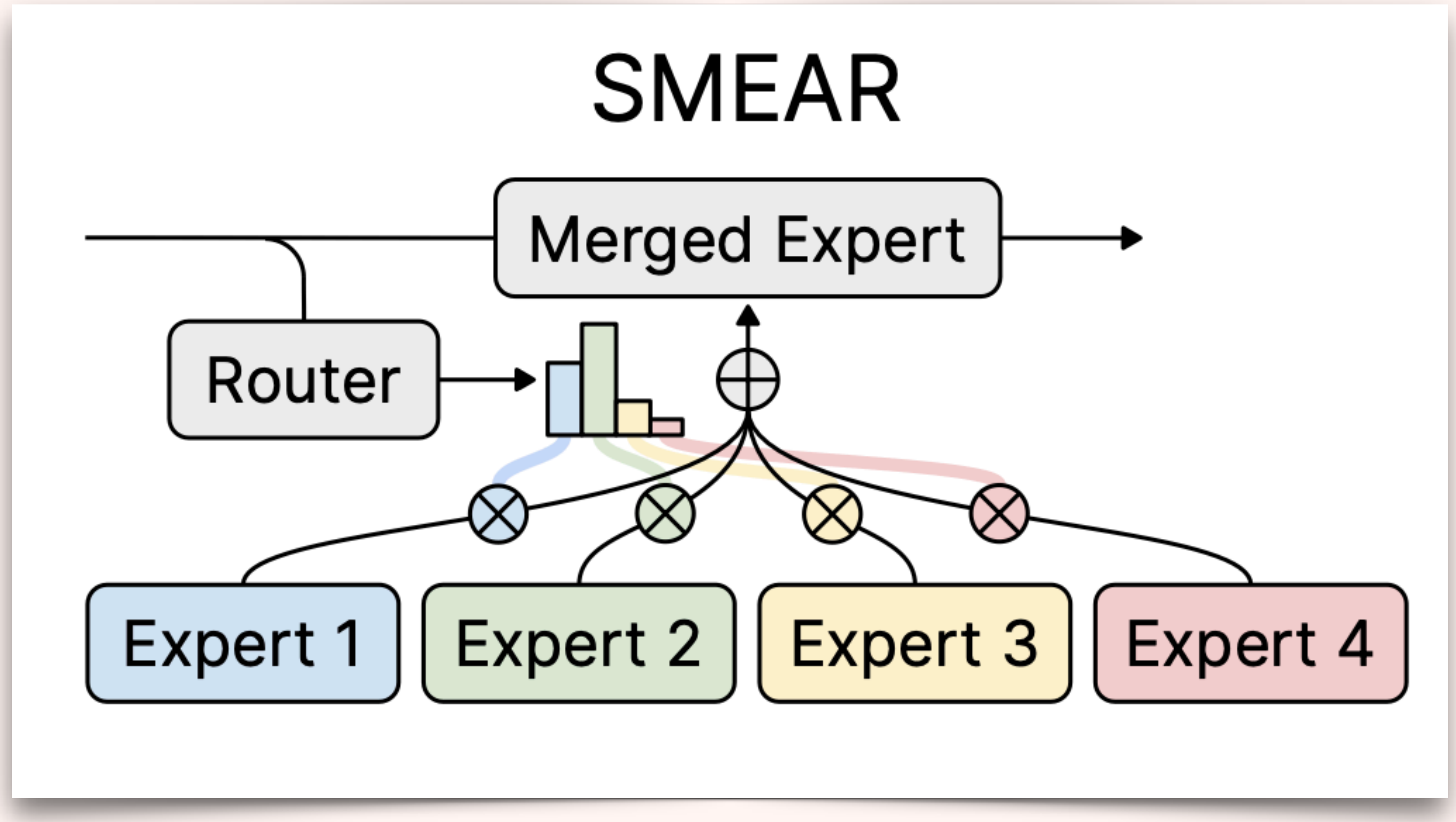
Output of the B is $f_i(u, \theta_i)$

ENSEMBLE ROUTING

- **Exactly compute** $\mathbb{E}_{i \sim P(v)} f_i(u, \theta_i)$
  - **End-to-end differentiable** 👍
  - **Computationally expensive** 👎

SMEAR

- **Computes a merged expert**
  - $\bar{f}(u, \sum_i P(v)_i \theta_i)$
- **End-to-end differentiable** 👍
- **Almost same computation as discrete routing** 👍
  - **Provided we share expert across the input**



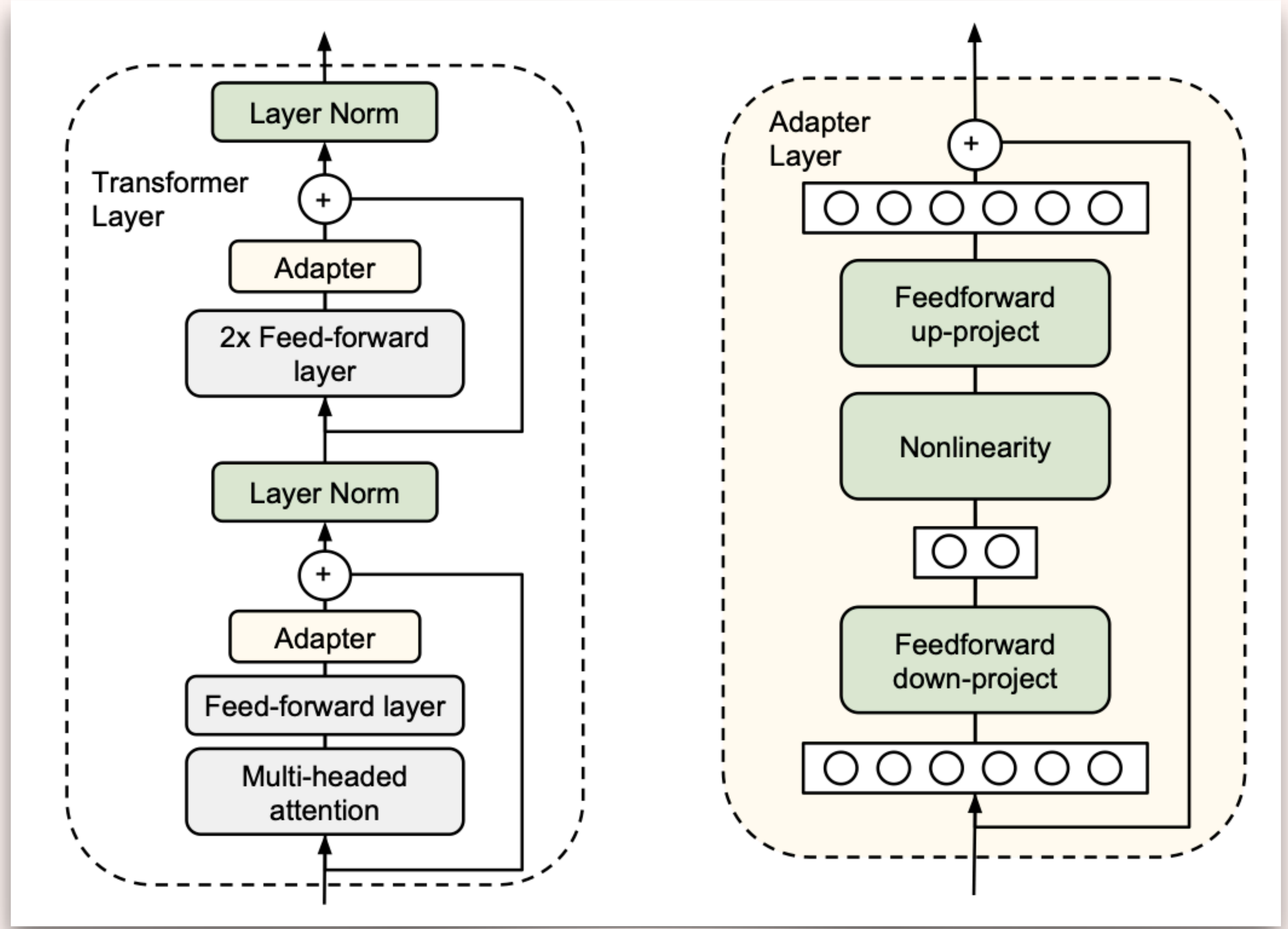
EXPERIMENTS

MULTITASK/MULTIDOMAIN

- T5-GLUE
  - 8 datasets: RTE, MNLI, QNLI, SST2, CoLA, QQP, MRPC, STSB
  - T5-Base 1.1
- ResNet-DomainNet
  - 6 domains: Clipart, Infograph, Painting, Quickdraw, Real, Sketch
  - ResNet-18

SETUP

- Experts are Adapters
  - Only trainable
  - Added after every self-attention and feed-forward layer in Transformer
  - Added after each ResNet-Block



NOW, NUMBERS!

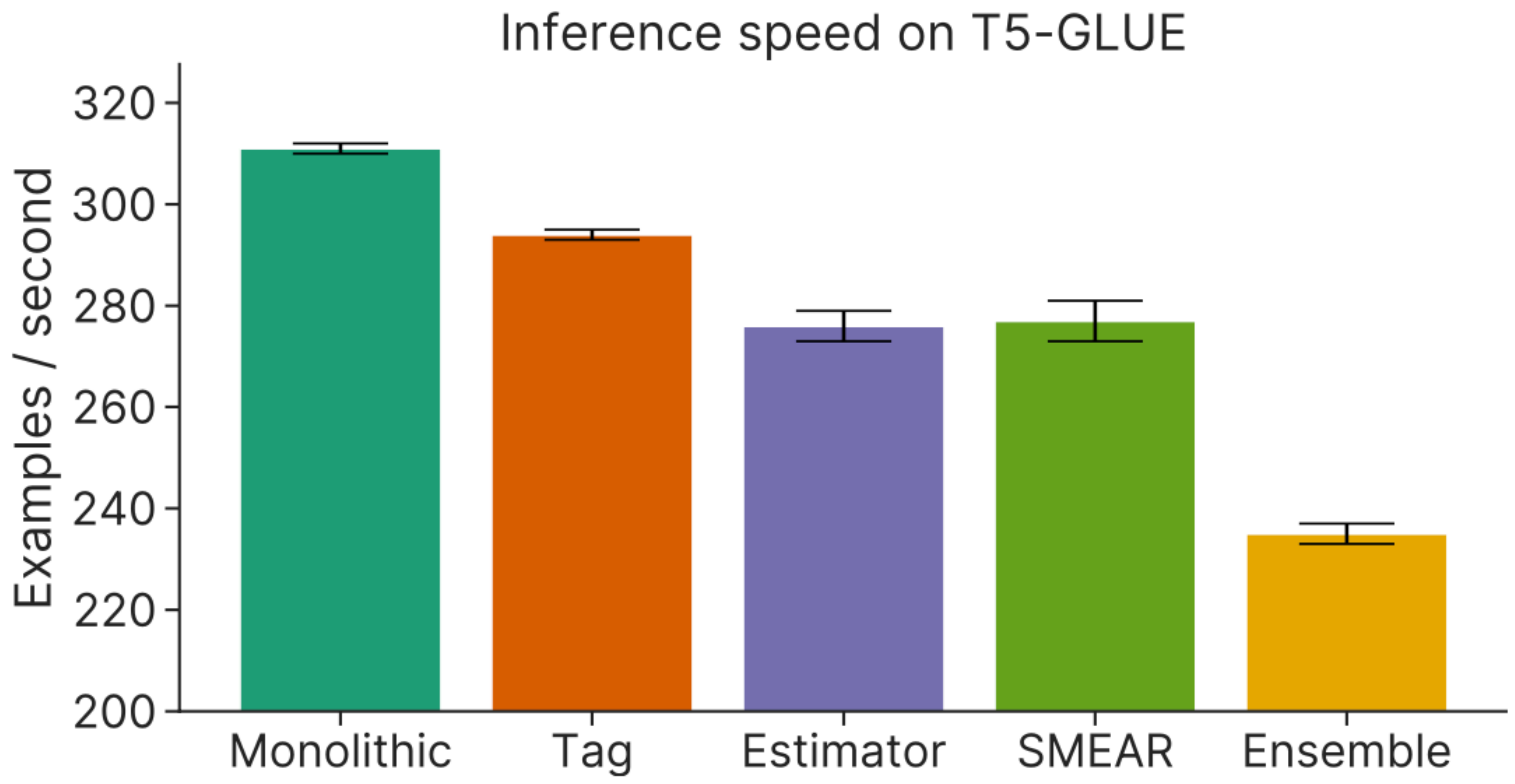
- Most estimators underperform heuristics
- SMEAR outperforms all
  - On T5-GLUE, 81.6 versus next best REINFORCE 80.0
  - On ResNet-DomainNet, 62.0 versus next best Tag 61.4

Routing	T5-GLUE	ResNet-DomainNet
Tag	$78.0_{1.2}$	$61.4_{0.1}$
Tag+	$78.5_{1.2}$	–
Hash	$66.9_{0.9}$	$52.4_{0.1}$
Monolithic	$78.3_{1.2}$	$59.0_{0.1}$
Top-k	$78.2_{0.9}$	$60.0_{0.1}$
ST-Gumbel	$77.9_{0.4}$	$58.5_{0.2}$
REINFORCE	$80.0_{0.8}$	$60.0_{0.1}$
SMEAR	$\mathbf{81.6}_{1.0}$	$\mathbf{62.0}_{0.1}$
Expert ensemble	$81.7_{1.0}$	$62.3_{0.1}$

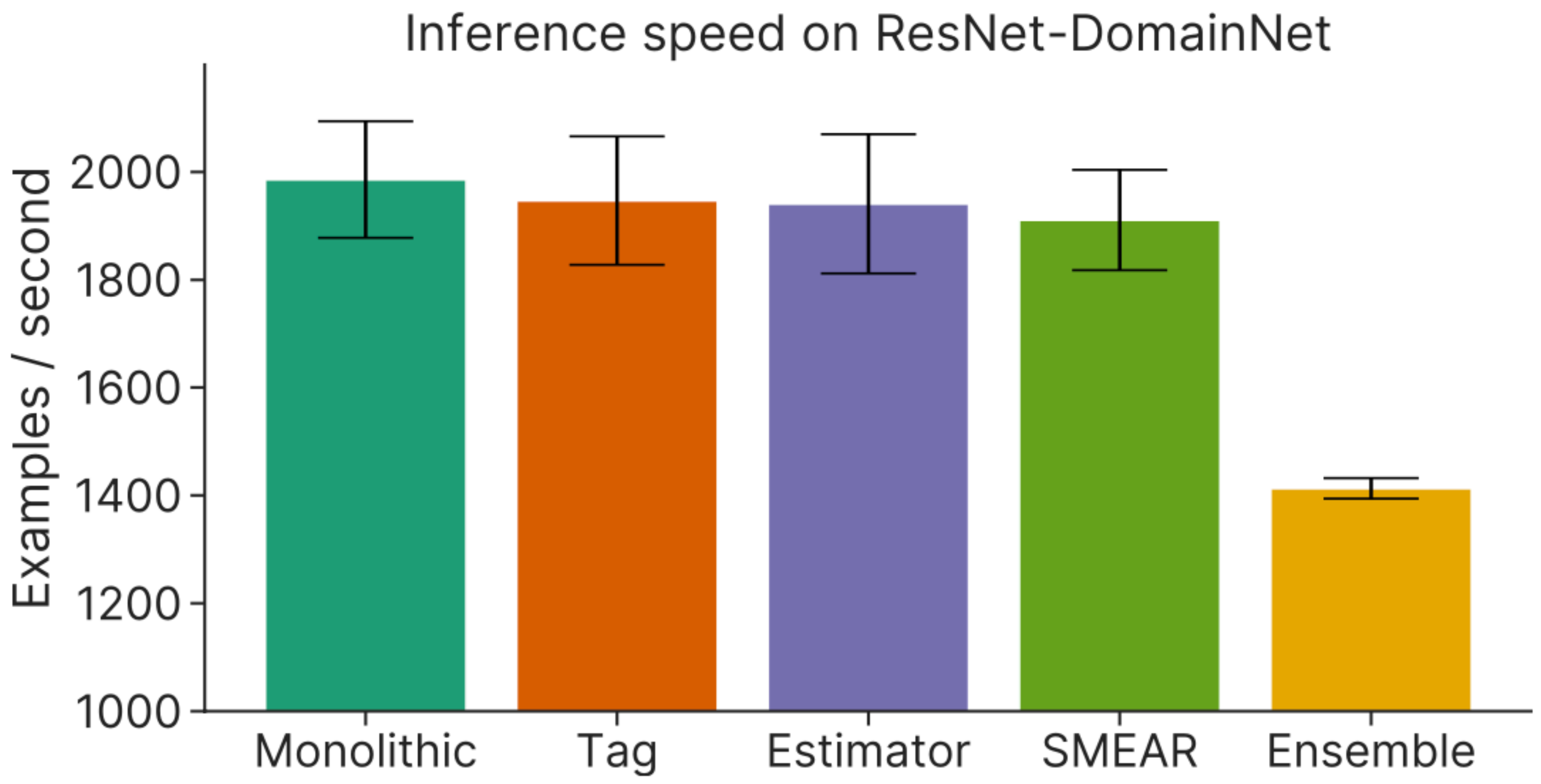
NUMBERS!

- **Monolithic - Parameter matched**
  - **A large expert with parameters = N * single expert**
  - **T5-GLUE ($80.2_{0.9}$) , ResNet-DomainNet ($60.8_{0.1}$)**

Routing	T5-GLUE	ResNet-DomainNet
Tag	$78.0_{1.2}$	$61.4_{0.1}$
Tag+	$78.5_{1.2}$	–
Hash	$66.9_{0.9}$	$52.4_{0.1}$
Monolithic	$78.3_{1.2}$	$59.0_{0.1}$
Top-k	$78.2_{0.9}$	$60.0_{0.1}$
ST-Gumbel	$77.9_{0.4}$	$58.5_{0.2}$
REINFORCE	$80.0_{0.8}$	$60.0_{0.1}$
SMEAR	$\mathbf{81.6}_{1.0}$	$\mathbf{62.0}_{0.1}$
Expert ensemble	$81.7_{1.0}$	$62.3_{0.1}$



(a)



(b)

Figure 2. Comparison of inference speed for various routing strategies in T5-GLUE (a) and ResNet-DomainNet (b). SMEAR has comparable speed with that of discrete routing with estimators, whereas computing an ensemble of experts ("Ensemble") is the slowest.

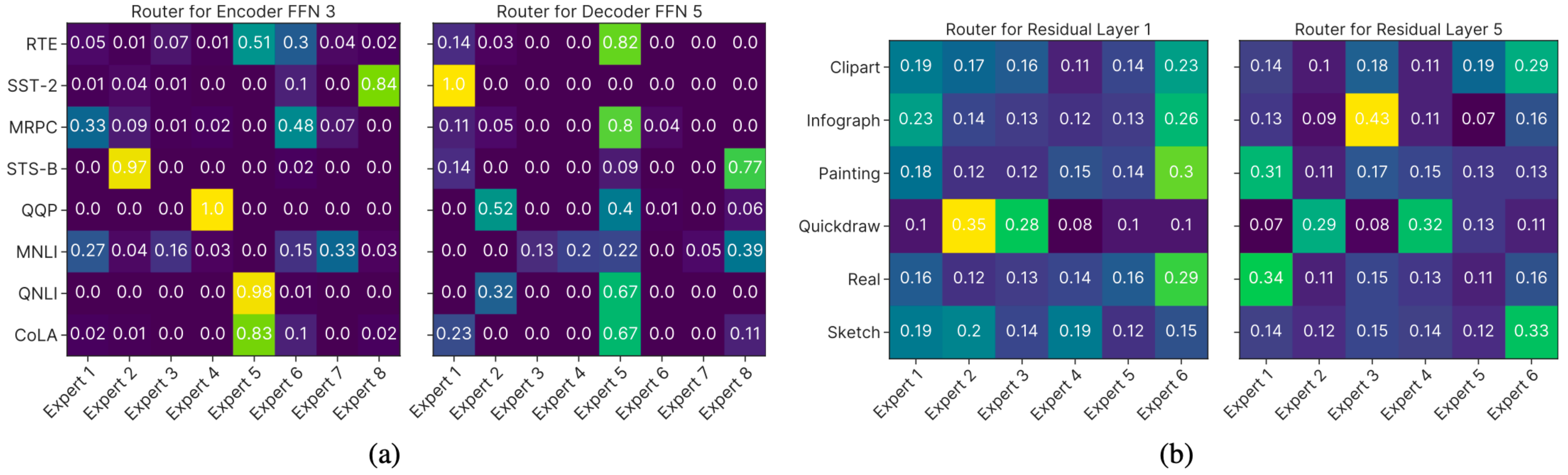


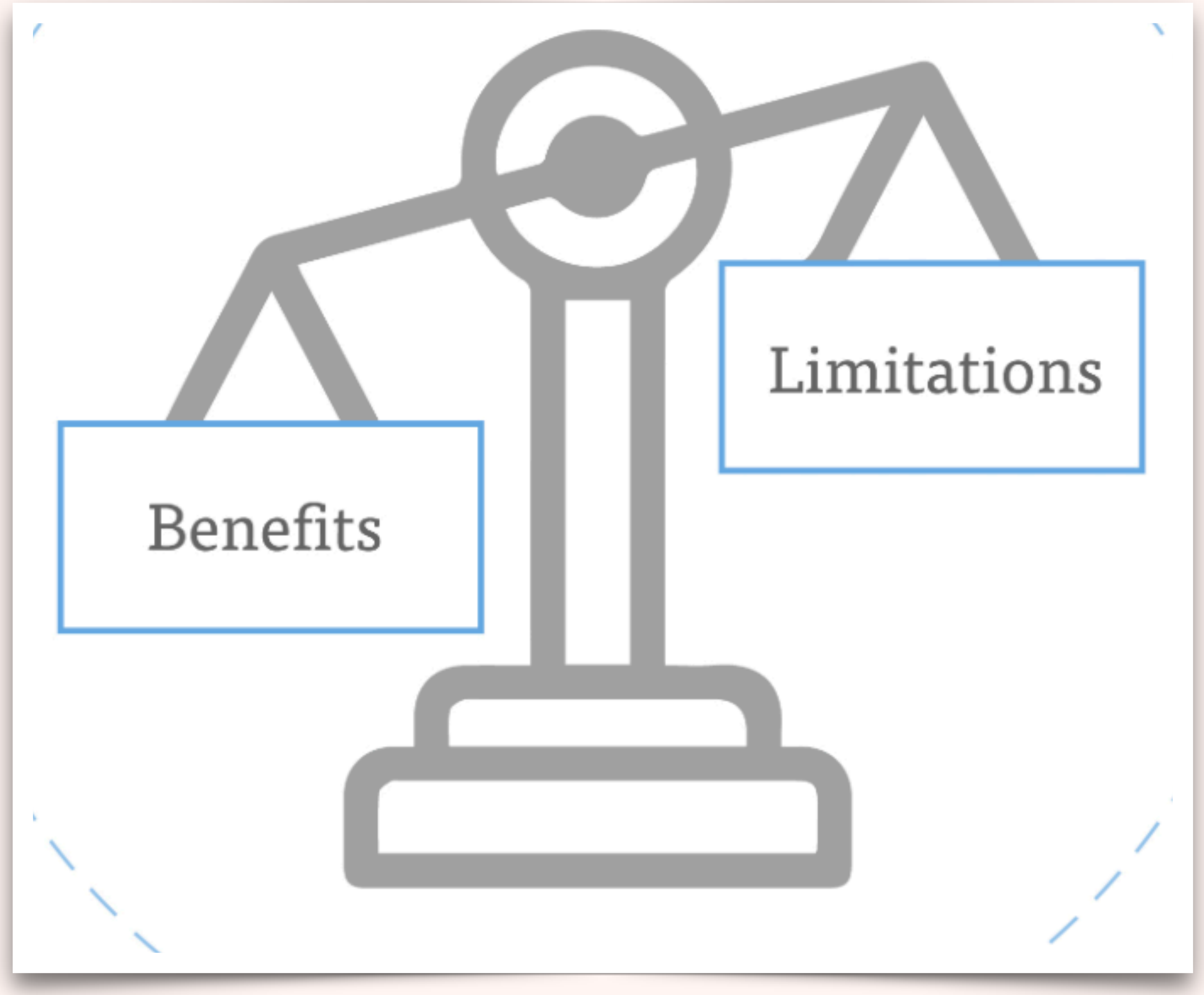
Figure 3. Average routing distributions produced by SMEAR for two routers from the T5-GLUE model (a) and two from the ResNet-DomainNet model (b). For a given router, we average all routing distributions across all examples from a given dataset.

TRAINING QUIRKS

- Load balancing does not work in our case
- LayerNorm to the input of the router &
- LayerNorm (any normalization) in the rows of Router
- Randomly dropping experts and re-normalizing expert distribution helps in Top-K and SMEAR
- Effect of scale when using Adaptive optimizers $\theta_t = \theta_{t-1} - \frac{\alpha \cdot m_t}{\sqrt{v_t} + \epsilon}$

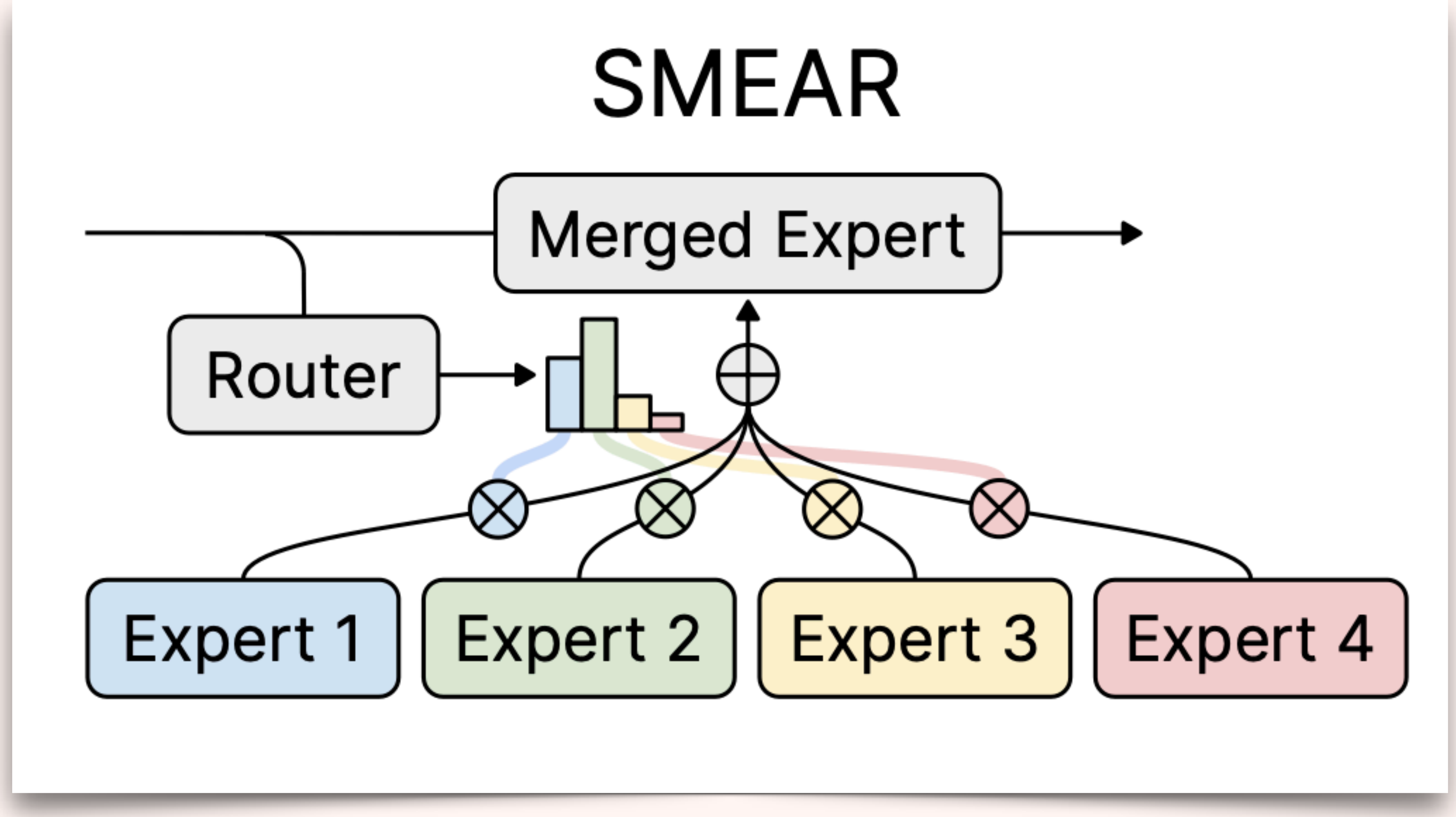
LIMITATIONS

- SMEAR needs experts loaded in the memory
  - Need weighted all-reduce if experts reside on different GPUs
- Pretraining methods use token-level routing
- Downstream tasks use sequence-level routing



TAKE AWAYS

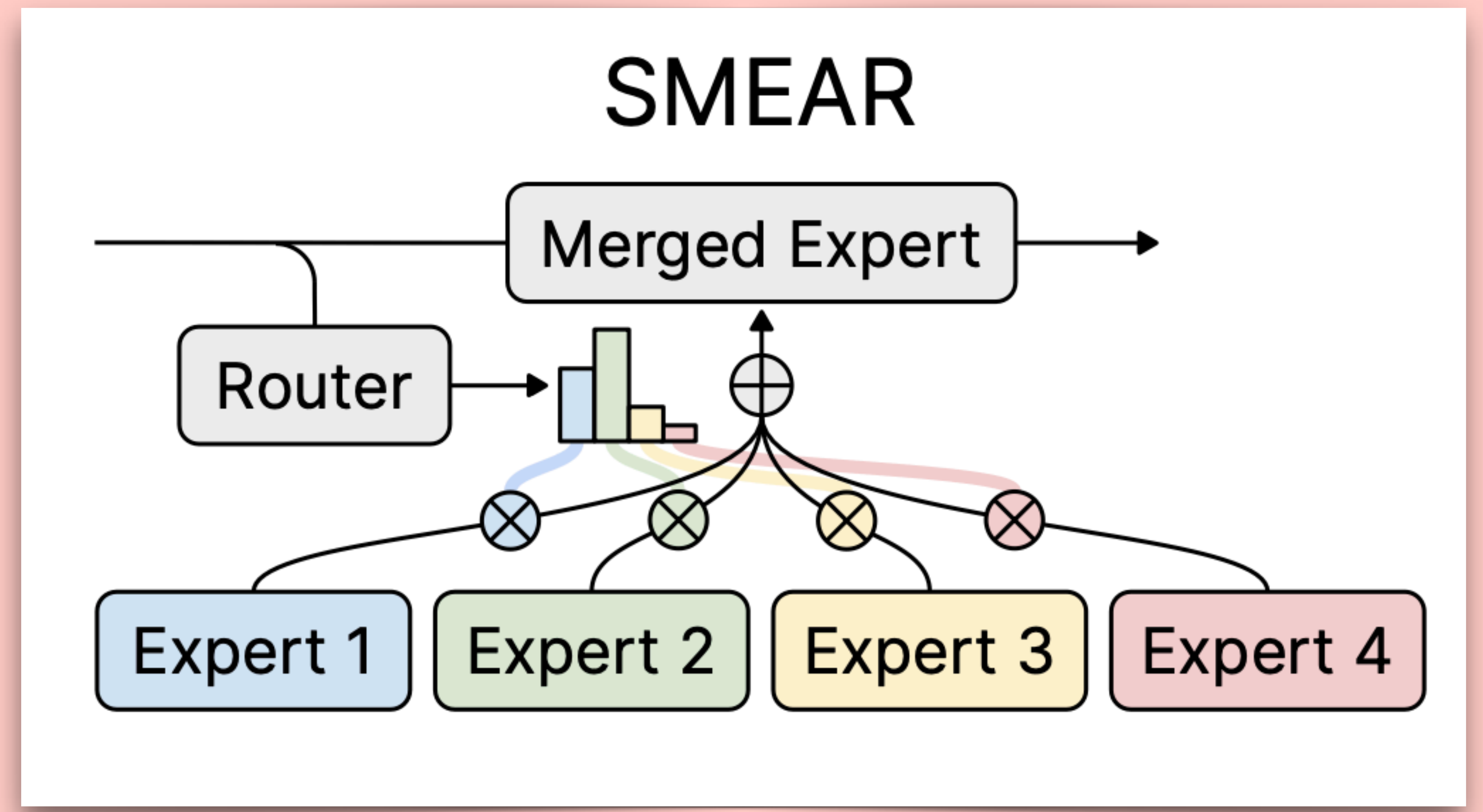
- **SMEAR learns routing by being end-to-end differentiable**
- **Outperforms estimators and heuristics**
- **Has comparable computation cost to discrete routing**



WHAT'S NEXT

- SMEAR is an excellent choice when task boundaries are not clear
  - Instruction following datasets
  - Preference datasets
- Experts themselves are reasonable sized
  - Parameter Efficient Modules match full-model finetuning
- Learn to control the capacity of the merged expert

THANK YOU!
QUESTIONS?



Collaborative Model Development:
https://github.com/r-three/git-theta
http://bit.ly/cccml-community