Mixture of Experts

Adrien Dubois

02/05/2025

PURDUE UNIVERSITY®

Sparsely Gated Mixture of Experts (SGMoE)

Motivation

- Authors:

Noam Shazeer[1], Azalia Mirhoseini[*†1], Krzysztof Maziarz[*2], Andy Davis[1], Quoc Le[1], Geoffrey Hinton[1] and Jeff Dean[1]

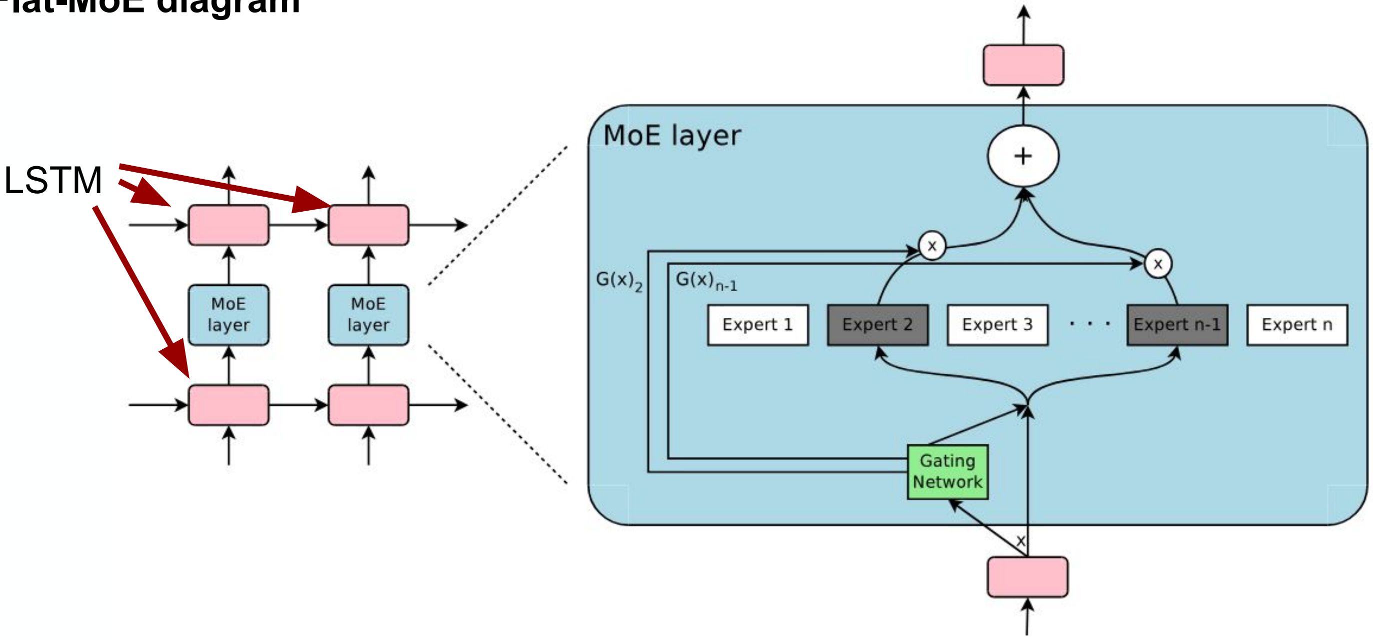
[1]Google Brain, {noam,azalia,andydavis,qvl,geoffhinton,jeff}@google.com
[2]Jagiellonian University, Cracow, krzysztof.maziarz@student.uj.edu.pl

- Release Date: Jan 2017
- Motivation:
  - Capacity of neural network to absorb information is limited by the number of parameters
    - More parameters = better performance
      - BUT
    - More parameters = more expensive
  - Conditional computation where only parts of the model are active at once allows for more parameters, without increasing computation costs.

PURDUE UNIVERSITY

Their approach

Flat-MoE diagram



Challenges with MoE at the time

- Branching computation is expensive, especially model parallelism (dividing 1 large model into subsequent devices)
- Large batches are critical to amortize the costs of parameter transfers and updates. However, conditional computation reduces batch sizes for each expert:

> **Note**
>
> We select a sparse number of experts for each input feature/token.
> Therefore, the input tokens in a batch don't all go through the experts, and one expert may get $\frac{b}{n}$ input tokens in a batch of size b. More on this later.

<< b

- Additional loss terms are required to favoring balancing the usage of each expert model
- At the time, the datasets were not large enough to train models with "millions, let alone billions of parameters"

PURDUE UNIVERSITY

Gating Architecture

- <u>(Basic, non-sparse) Softmax Gating (**Softmax classifier**):</u>
  - Multiply input x, by a trainable weight matrix W_g, then apply softmax to make it a valid probability distribution

$$G_\sigma(x) = Softmax(x \cdot W_g)$$

- <u>(Sparse) Noisy Top-k Gating (**the paper**):</u>
  - The authors add two things: sparsity and noise

sparsity

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

Noise

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

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

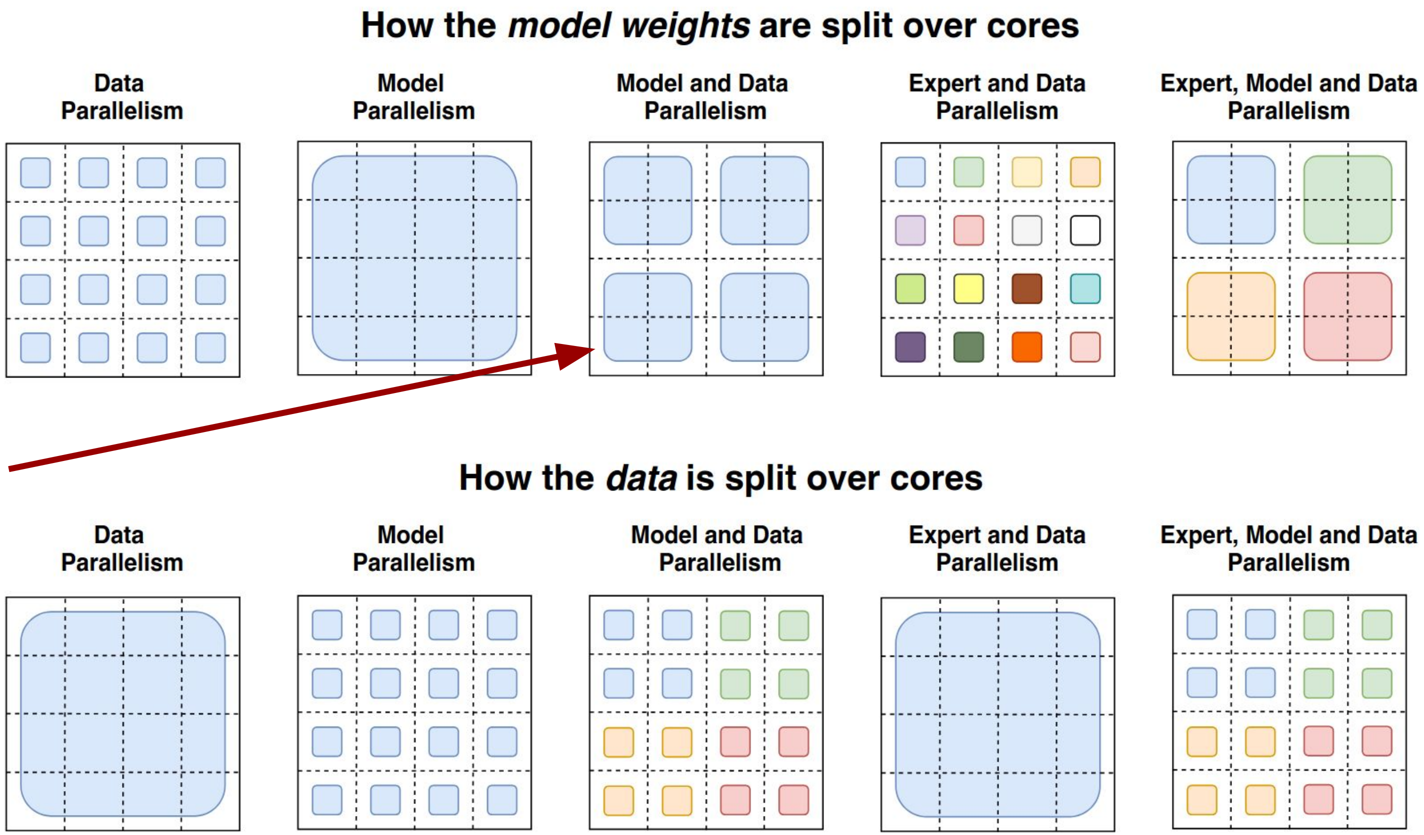
PURDUE UNIVERSITY

Training the gates and related issues

- Training the gating network:
  - The gating network is trained through normal backprop
  - Use k>1 experts, (explained why in future slides)
- Shrinking batch problem:
  - Reminder: As you increase num-experts, batch size per expert decreases which is inefficient
  - Solution: Mixing data parallelism and model parallelism
    - Normal: Model copies are run asynchronously, with synchronous
    - Here: each expert is only on one device as a shared resource. The rest of the model is parallelised. Batches are run synchronously, and relevant tokens are grouped at the expert device.
    - Increases expert batch from (kb/n << b) to (kbd/n) for:
      - k = num experts from topk
      - b = batch size
      - n = n experts
      - d = d devices for the experts

PURDUE UNIVERSITY

Switch Transf. on Parallelism Strategies



SGMoE

Updated loss function for balancing experts

- Without additional loss terms, the network tends to converge to only using the same few experts.
  - This imbalance is self-reinforcing
- So they introduce 2 auxiliary losses that provide a soft-constraint on the batch-wise average of each gate, favoring a uniform distribution.
- <u>Importance loss:</u> Encourages each expert to receive ~= num tokens within each batch

$$Importance(X) = \sum_{x \in X} G(x)$$

$$L_{importance}(X) = w_{importance} \cdot CV(Importance(X))^2$$

- <u>Load loss:</u> measures the total load of expert i over the entire dataset X

$$P(x,i) = Pr\Big((x \cdot W_g)_i + StandardNormal() \cdot Softplus((x \cdot W_{noise})_i) > kth\_excluding(H(x), k, i)\Big)$$

=

$$P(x,i) = \Phi\Big(\frac{(x \cdot W_g)_i - kth\_excluding(H(x), k, i)}{Softplus((x \cdot W_{noise})_i)}\Big)$$

$$Load(X)_i = \sum_{x \in X} P(x,i)$$

$$L_{load}(X) = w_{load} \cdot CV(Load(X))^2$$

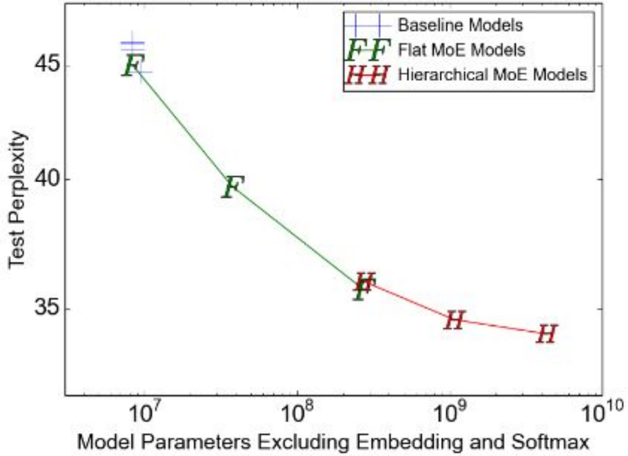
PURDUE UNIVERSITY

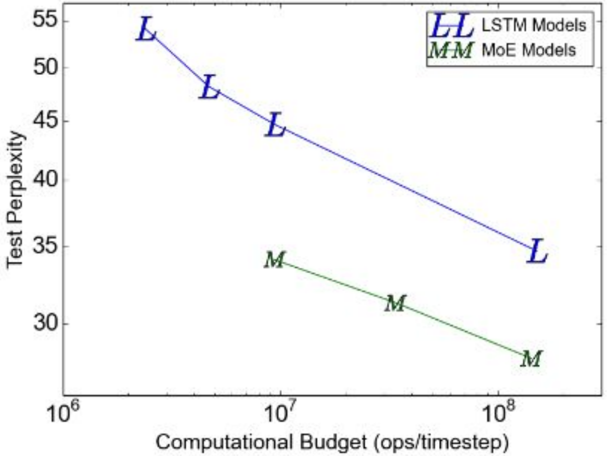
Determining the number of experts

- When training the MoE, the authors state that k>1 (for topk experts) is ideal due to:
  - **Improved load balancing:** Gating mechanism becomes too deterministic otherwise which means that the load balancing loss cannot effectively encourage a uniform distribution utilization
  - **Better gradient flow:** Backprop goes through multiple experts
    - Gradients are less likely to vanish/explode
    - Combining the outputs of multiple experts creates more robust representations.

SGMoE Evaluation

- Flat MoE are tested for 4, 32 and 256 experts
- Hierarchical MoEs are tested for 256, 1024, 4096 experts





SGMoE Evaluation 2

Table 2: Results on WMT'14 En→ Fr newstest2014 (bold values represent best results).

Model	Test Perplexity	Test BLEU	ops/timenstep	Total #Parameters	Training Time
MoE with 2048 Experts	2.69	40.35	85M	8.7B	3 days/64 k40s
MoE with 2048 Experts (longer training)	**2.63**	**40.56**	85M	8.7B	6 days/64 k40s
GNMT (Wu et al., 2016)	2.79	39.22	214M	278M	6 days/96 k80s
GNMT+RL (Wu et al., 2016)	2.96	39.92	214M	278M	6 days/96 k80s
PBMT (Durrani et al., 2014)		37.0			
LSTM (6-layer) (Luong et al., 2015b)		31.5			
LSTM (6-layer+PosUnk) (Luong et al., 2015b)		33.1			
DeepAtt (Zhou et al., 2016)		37.7			
DeepAtt+PosUnk (Zhou et al., 2016)		39.2			

Table 3: Results on WMT'14 En → De newstest2014 (bold values represent best results).

Model	Test Perplexity	Test BLEU	ops/timestep	Total #Parameters	Training Time
MoE with 2048 Experts	**4.64**	**26.03**	85M	8.7B	1 day/64 k40s
GNMT (Wu et al., 2016)	5.25	24.91	214M	278M	1 day/96 k80s
GNMT +RL (Wu et al., 2016)	8.08	24.66	214M	278M	1 day/96 k80s
PBMT (Durrani et al., 2014)		20.7			
DeepAtt (Zhou et al., 2016)		20.6			

Table 4: Results on the Google Production En→ Fr dataset (bold values represent best results).

Model	Eval Perplexity	Eval BLEU	Test Perplexity	Test BLEU	ops/timestep	Total #Parameters	Training Time
MoE with 2048 Experts	**2.60**	**37.27**	**2.69**	**36.57**	85M	8.7B	1 day/64 k40s
GNMT (Wu et al., 2016)	2.78	35.80	2.87	35.56	214M	278M	6 days/96 k80s

Switch Transformer (ST)

PURDUE UNIVERSITY

Motivation

- Authors:

William Fedus*
LIAMFEDUS@GOOGLE.COM

Barret Zoph*
BARRETZOPH@GOOGLE.COM

Noam Shazeer
NOAM@GOOGLE.COM
Google, Mountain View, CA 94043, USA

Same as MgMoE

- Release Date: June 2022

Motivation

- Fun goal:
  - Train a model with 1 trillion params on 1 billion words (people didn't think this was viable at the time)
- True motivation:

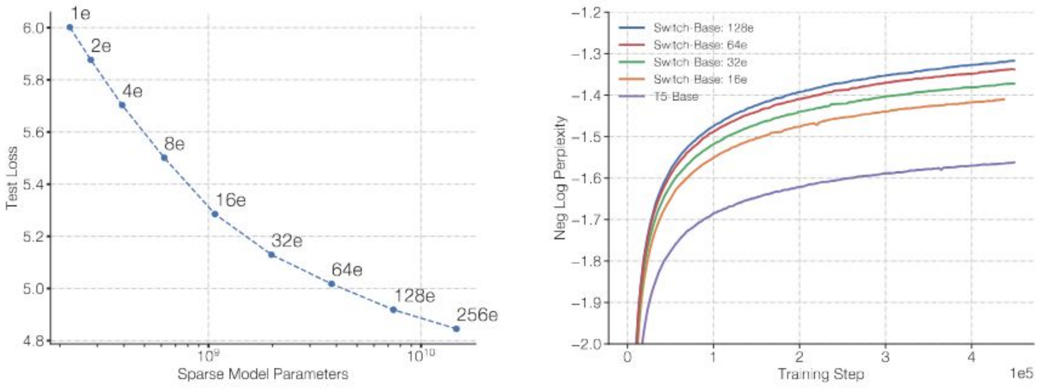
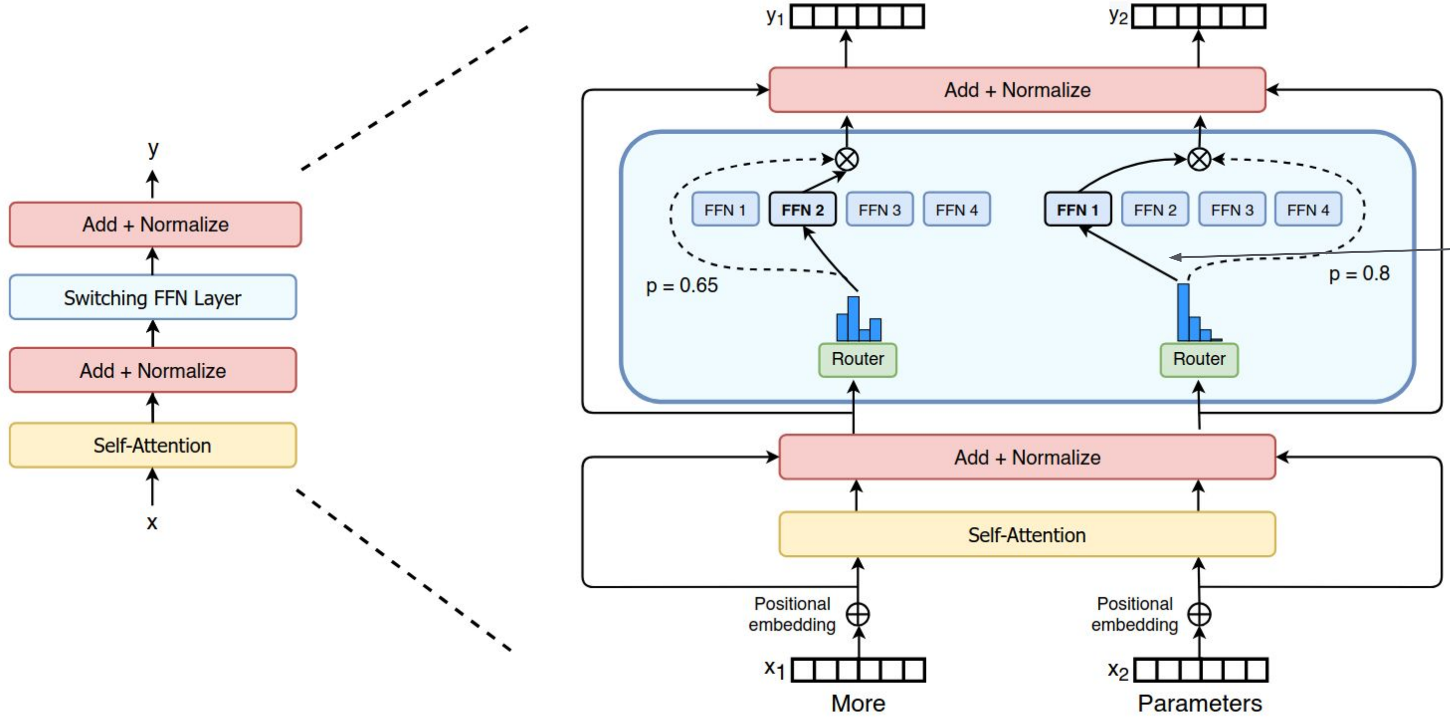


Figure 1: Scaling and sample efficiency of Switch Transformers. Left Plot: Scaling properties for increasingly sparse (more experts) Switch Transformers. Right Plot: Negative log perplexity comparing Switch Transformers to T5 (Raffel et al., 2019) models using the same compute budget.

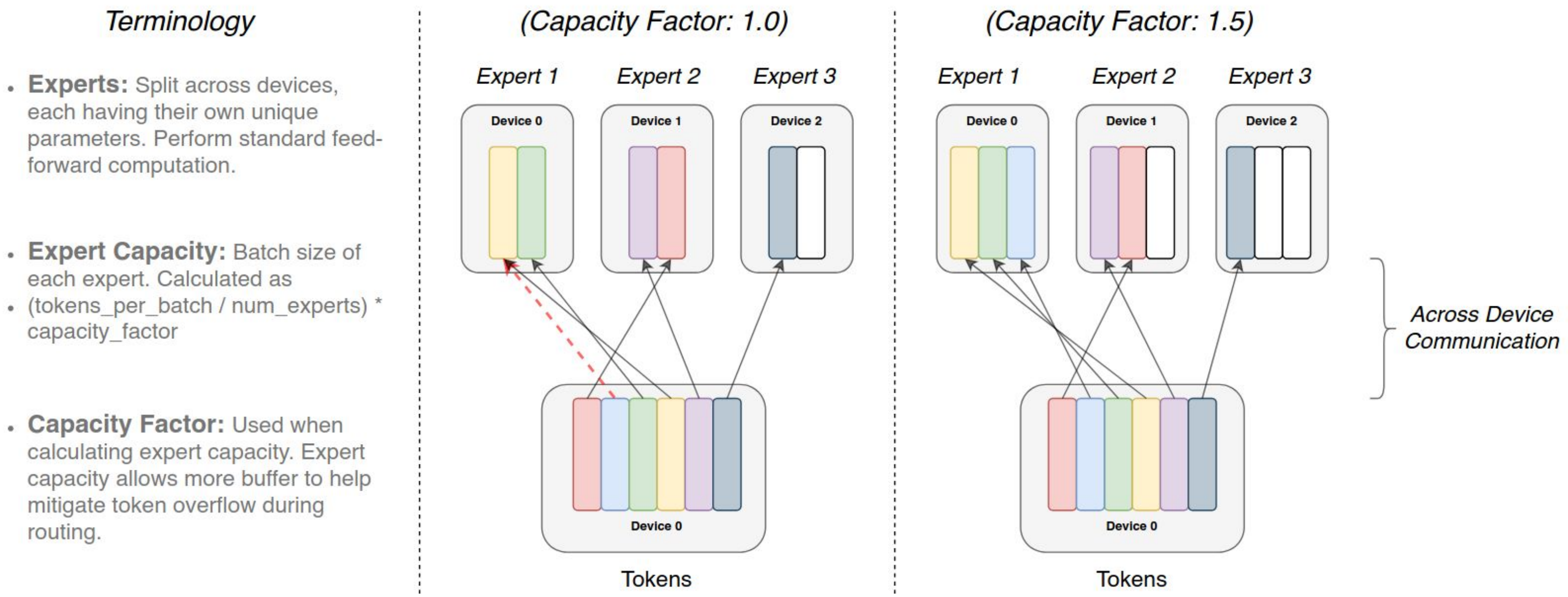
PURDUE UNIVERSITY

Integration of MoE with transformers



1 expert

Expert routing with capacity



- If the expert is already at capacity (red), that input token is dropped.

How to stabilize the learning with k=1

Model	Parameters	FLOPs/seq	d_{model}	FFN_{GEGLU}	d_{ff}	d_{kv}	Num. Heads
T5-Base	0.2B	124B	768	✓	2048	64	12
T5-Large	0.7B	425B	1024	✓	2816	64	16
T5-XXL	11B	6.3T	4096	✓	10240	64	64
Switch-Base	7B	124B	768	✓	2048	64	12
Switch-Large	26B	425B	1024	✓	2816	64	16
Switch-XXL	395B	6.3T	4096	✓	10240	64	64
Switch-C	1571B	890B	2080		6144	64	32

Model	Expert Freq.	Num. Layers	Num Experts	Neg. Log Perp. @250k	Neg. Log Perp. @ 500k
T5-Base	–	12	–	-1.599	-1.556
T5-Large	–	24	–	-1.402	-1.350
T5-XXL	–	24	–	-1.147	-1.095
Switch-Base	1/2	12	128	-1.370	-1.306
Switch-Large	1/2	24	128	-1.248	-1.177
Switch-XXL	1/2	24	64	**-1.086**	**-1.008**
Switch-C	1	15	2048	-1.096	-1.043

PURDUE UNIVERSITY

Dropout in gating mechanism

- Instead of adding random noise similar to SGMoE, the authors introduce dropout to the gating system
- This is mainly done at the expert level (ed = expert dropout).
    - This is possible since the experts are already sparse so even removing an entire expert may not reduce model capability by a huge amount.

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**

Table 4: Fine-tuning regularization results. A sweep of dropout rates while fine-tuning Switch Transformer models pre-trained on 34B tokens of the C4 data set (higher numbers are better). We observe that using a lower standard dropout rate at all non-expert layer, with a much larger dropout rate on the expert feed-forward layers, to perform the best.

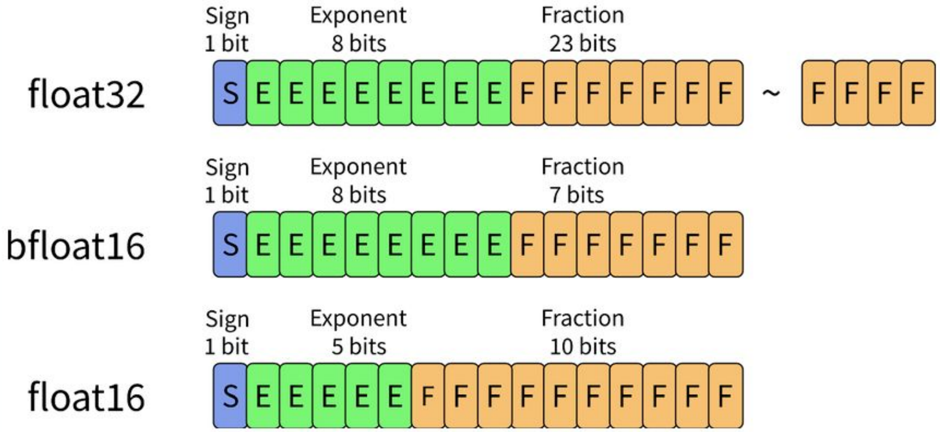
PURDUE UNIVERSITY

Smaller Parameter Initialization

- The authors initialize the model parameters from a truncated Normal distribution with:
  - mean = 0
  - stdev = $\sqrt{s/n}$
    - s = scale hyper-parameter (they reduce it by a factor of 10 as compared to default transformer)
    - n = number of input units in the weight tensor (fan-in)

Selective Precision

- Model instability hinders training fully at BF16, so previous papers required full float32 computation (default in Pytorch).
- This has expensive communication costs.
- So they "selectively cast" input tokens to float32 when entering each expert's device, then convert back to BF16 before leaving the GPU.
- This truncation is done by truncation of the "mantissa" (fraction) bits.



Load balancing Loss

- As in SGMoE, the authors encourage expert balancing through auxiliary loss. However, they are both combined into one simplified loss term here.

Given N experts indexed by $i = 1$ to N and a batch $\mathcal{B}$ with T tokens, the auxiliary loss is computed as the scaled dot-product between vectors f and P,

$$\text{loss} = \alpha \cdot N \cdot \sum_{i=1}^{N} f_i \cdot P_i \tag{4}$$

where f_i is the fraction of tokens dispatched to expert i,

$$f_i = \frac{1}{T} \sum_{x \in \mathcal{B}} \mathbb{1}\{\text{argmax } p(x) = i\} \tag{5}$$

and P_i is the fraction of the router probability allocated for expert i, [2]

$$P_i = \frac{1}{T} \sum_{x \in \mathcal{B}} p_i(x). \tag{6}$$

PURDUE UNIVERSITY

Evaluation 1

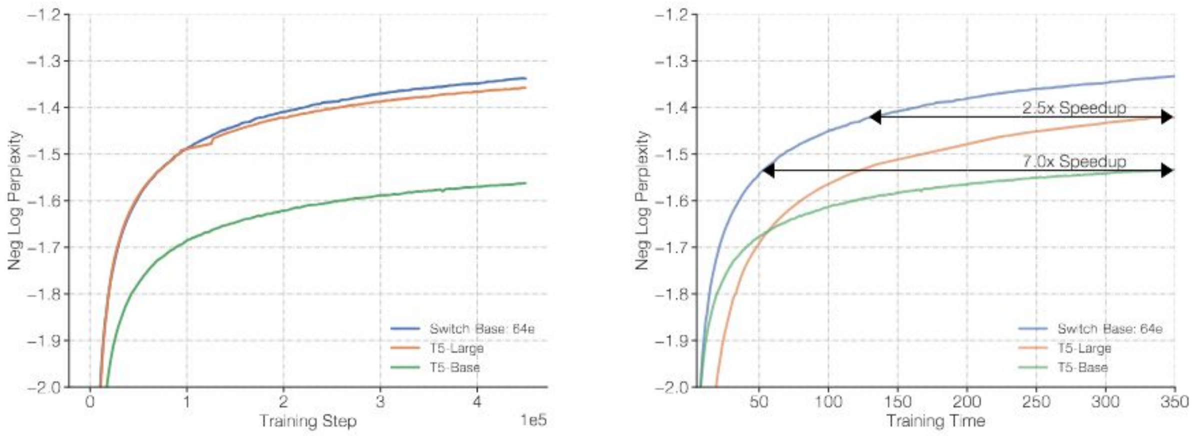


Figure 6: Scaling Transformer models with Switch layers or with standard dense model scaling. Left Plot: Switch-Base is more sample efficient than both the T5-Base, and T5-Large variant, which applies 3.5x more FLOPS per token. Right Plot: As before, on a wall-clock basis, we find that Switch-Base is still faster, and yields a 2.5x speedup over T5-Large.

PURDUE UNIVERSITY

Evaluation 2: converges faster

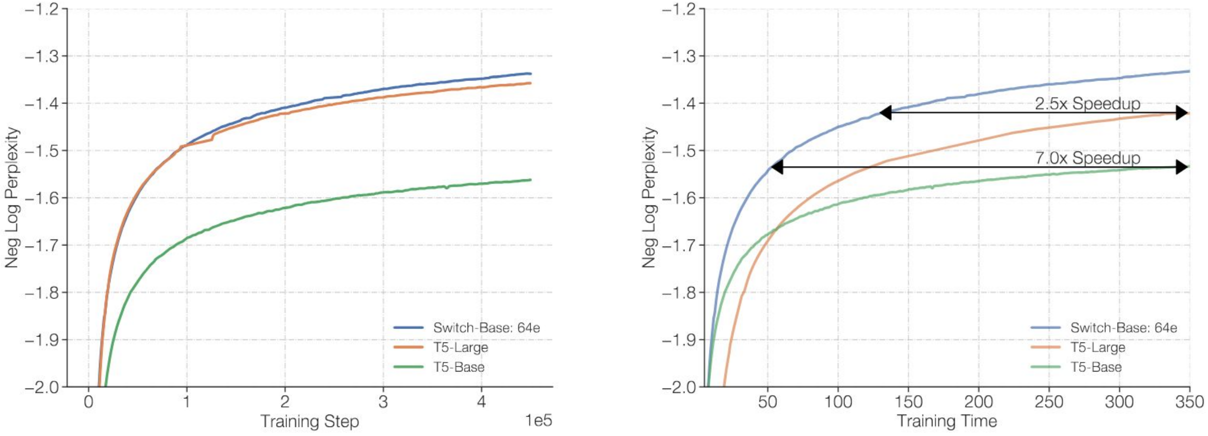


Figure 6: Scaling Transformer models with Switch layers or with standard dense model scaling. Left Plot: Switch-Base is more sample efficient than both the T5-Base, and T5-Large variant, which applies 3.5x more FLOPS per token. Right Plot: As before, on a wall-clock basis, we find that Switch-Base is still faster, and yields a 2.5x speedup over T5-Large.

PURDUE UNIVERSITY

Evaluation 2: higher accuracy in all langs.

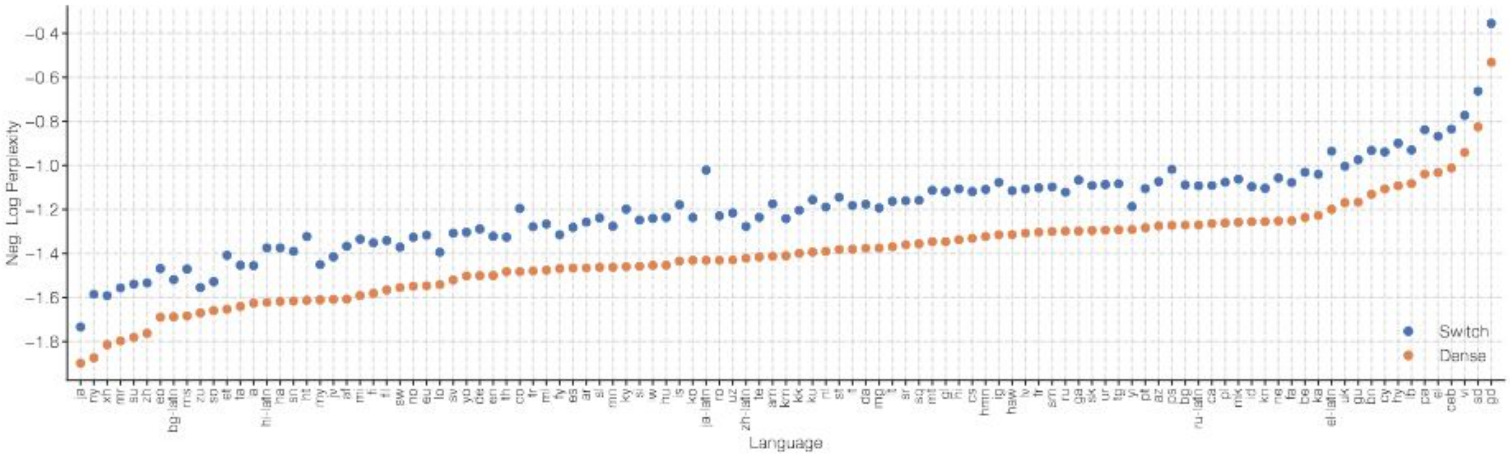
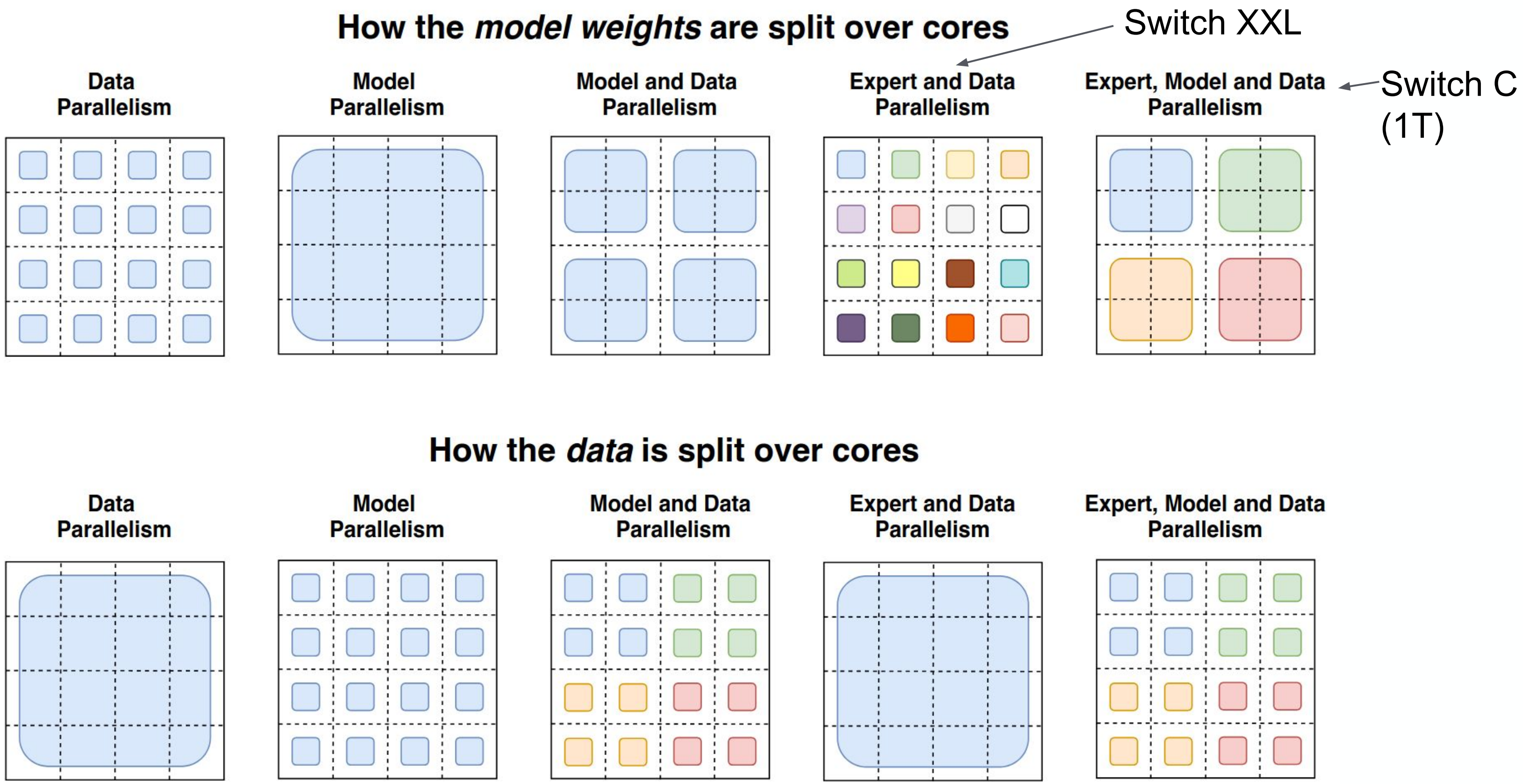


Figure 7: Multilingual pre-training on 101 languages. Improvements of Switch T5 Base model over dense baseline when multi-task training on 101 languages. We observe Switch Transformers to do quite well in the multi-task training setup and yield improvements on all 101 languages.

PURDUE UNIVERSITY

Switch Transf. on Parallelism Strategies

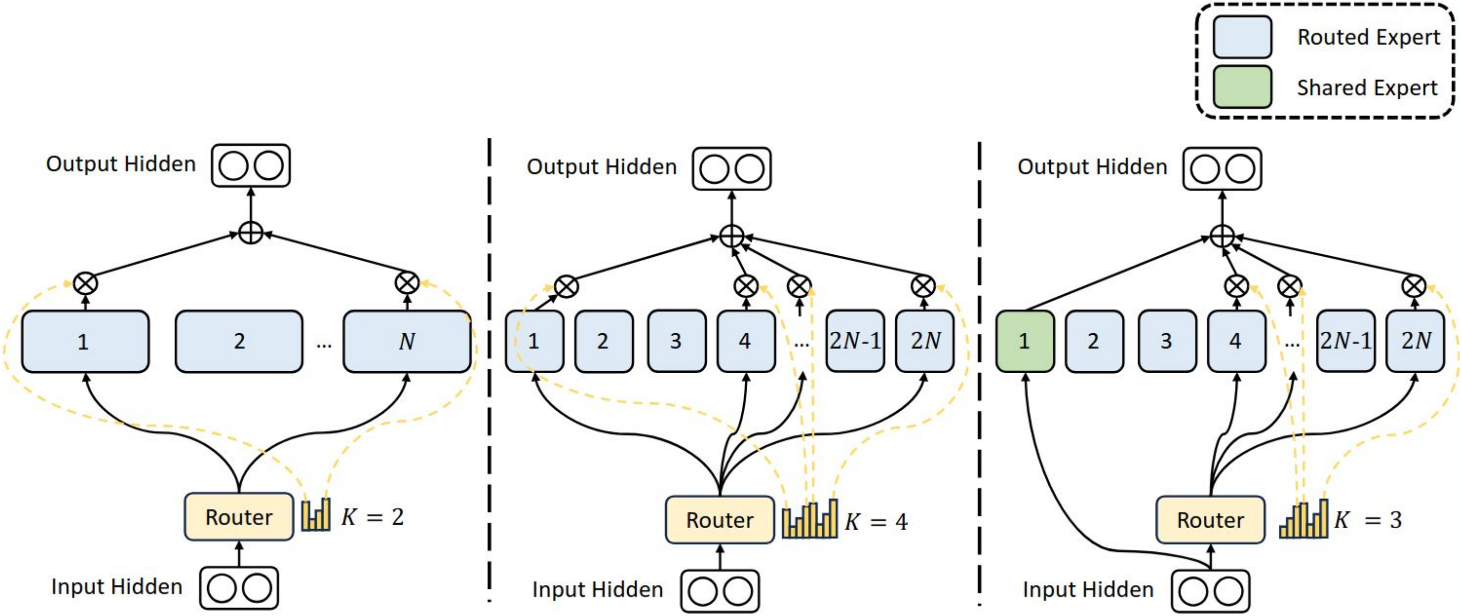


PURDUE UNIVERSITY

Deepseek Updates

PURDUE UNIVERSITY

Fine-Grained Expert Seg./Shared Expert Iso



(a) Conventional Top-2 Routing → **(b) + Fine-grained Expert Segmentation** → **(c) + Shared Expert Isolation (DeepSeekMoE)**

DeepSeek - Removal of Auxiliary Loss

- Drops auxiliary losses and instead uses a bias term:
  - If expert i is overused, decrease the bias
    - $b_i = b_i - \gamma$
  - Else:
    - $b_i = b_i + \gamma$
- The expert is thus chosen through this formula:

$$g'_{i,t} = \begin{cases} s_{i,t}, & s_{i,t} + b_i \in \operatorname{Topk}(\{s_{j,t} + b_j | 1 \leqslant j \leqslant N_r\}, K_r), \\ 0, & \text{otherwise.} \end{cases}$$

Thank You

Purdue RVL

PURDUE
UNIVERSITY®

Challenges with MoE at the time

- Authors:

Deepseek:

- <u>Hierarchical MoE structure.</u> The tokens go through the following hierarchies:
    - **Global Selection:** inputs are routed to an initial pool of experts using Softmax Affinity Scoring
    - **Cluster-Level Pruning:** within each pool, a secondary gating mechanism prunes experts based on entropy constraints
    - **Final expert assignments:** Top-k experts are chosen using either Entropy Aware Gating, or RL agent in Deepseek R1.
- <u>Auxiliary Losses:</u>
    - **Load balance loss:** we saw in switch transformer. Balances the usage of each individual expert.
    - **Device balance loss:** the experts are split into groups and assigned to devices. They want all devices to be used relatively equally.
    - **Communication balance loss:** balances the communication load across experts.
- <u>Token-Dropping Strategy:</u>
    - Capacity factor = 1
    - When overflowing, drop the token with the lowest affinity score
    - Randomly sample 10% of the input sequences to have no dropped tokens.

DeepseekV3:

- Drops auxiliary losses and instead uses a bias term:
  - If expert i is overused, decrease the bias
    - b_i = b_i - gamma
  - Else:
    - b_i = b_i + gamma
- Mostly engineering improvements on routing experts in distributed learning setting.

DeepseekR1:

- Introduces RL based expert routing
- Instead of using a learned linear layer with softmax activation for expert routing, Deepseek R1 utilizes a learned RL policy to dynamically assign tokens to experts.
- The policy is as follows
  - The expert selection function is formulated as an RL policy optimization problem, where the probability of selecting expert e_i for token t is adjusted dynamically based on token embeddings u_t:

$$g_{i,t} = \pi_\theta(e_i | u_t)$$

  - where π_θ is the policy network that selects experts based on contextual embeddings. The optimization objective follows GRPO:

$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{q \sim P(Q), \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}} \left[\frac{1}{G} \sum_{i=1}^{G} \min \left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)} A_i, \text{clip}(\cdot) \right) - \beta D_{\text{KL}}(\pi_\theta || \pi_{\text{ref}}) \right]$$

  - where D_{KL} regularizes the policy update to prevent drastic shifts.

PURDUE UNIVERSITY