Mixture of Experts

Basic concepts

- **FLOPS**: floating point operations per second，指每秒浮点运算次数，为计算速度，是一个衡量硬件性能的指标。
- **FLOPs**: floating point operations，指浮点运算数，为计算量，可以用来衡量算法/模型的复杂度。
- **Compute Budget**: 1 PF-day = 1 PetaFLOPS * day = 10^15 × 24 × 3600 = 8.64 × 10^19 floating point operations.

Number	Order of magnitude	Prefix	Order of magnitude	Abbreviation	Computer performance	Storage capacity
Thousand	10^3	kilo-	2^{10}	K		kilobyte(KB)
Million	10^6	mega-	2^{20}	M		megabyte(MB)
Billion	10^9	giga-	2^{30}	G	gigaFLOPS(GFLOPS)	gigabyte(GB)
Trillion	10^{12}	tera-	2^{40}	T	teraFLOPS(TFLOPS)	terabyte(TB)
Quadrillion	10^{15}	peta-	2^{50}	P	petaFLOPS(PFLOPS)	petabyte(PB)
Quintillion	10^{18}	exa-	2^{60}	E	exaFLOPS(EFLOPS)	exabyte(EB)
Sextillion	10^{21}	zetta-	2^{70}	Z	zettaFLOPS(ZFLOPS)	zettabyte(ZB)

Scaling Laws for Neural Language Models

Throughout we will observe precise power-law scalings for performance as a function of **training time, context length, dataset size, model size, and compute budget.**

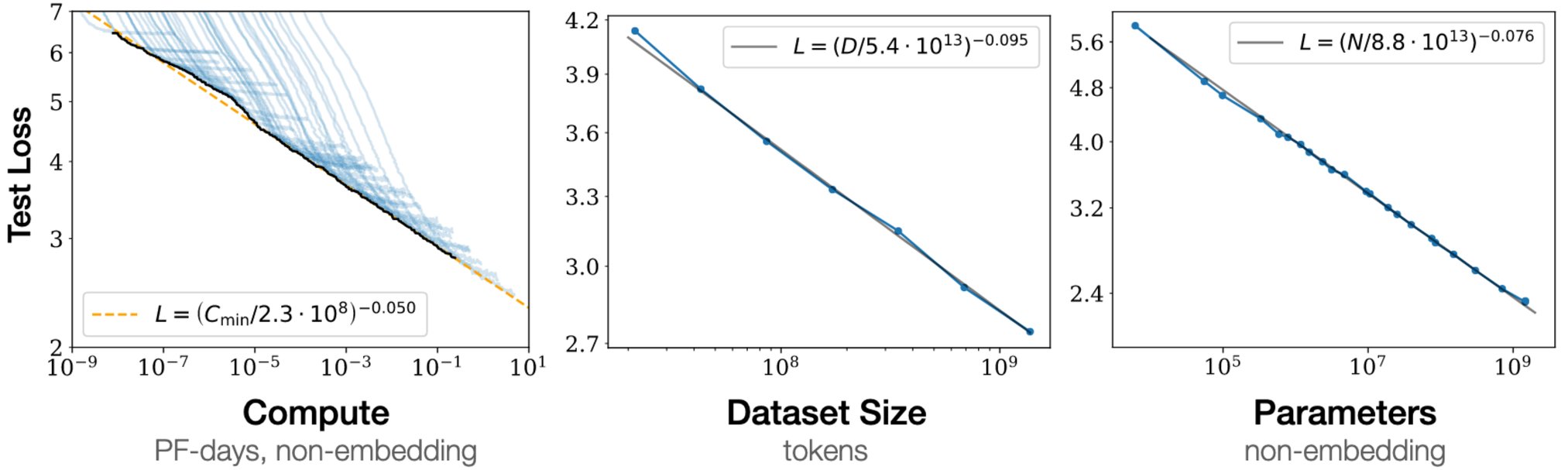


Figure 1 Language modeling performance improves smoothly as we increase the model size, datasetset size, and amount of compute[2] used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.

Scaling Laws for Neural Language Models

Large models are more sample-efficient than small models, reaching the same level of performance with fewer optimization steps.

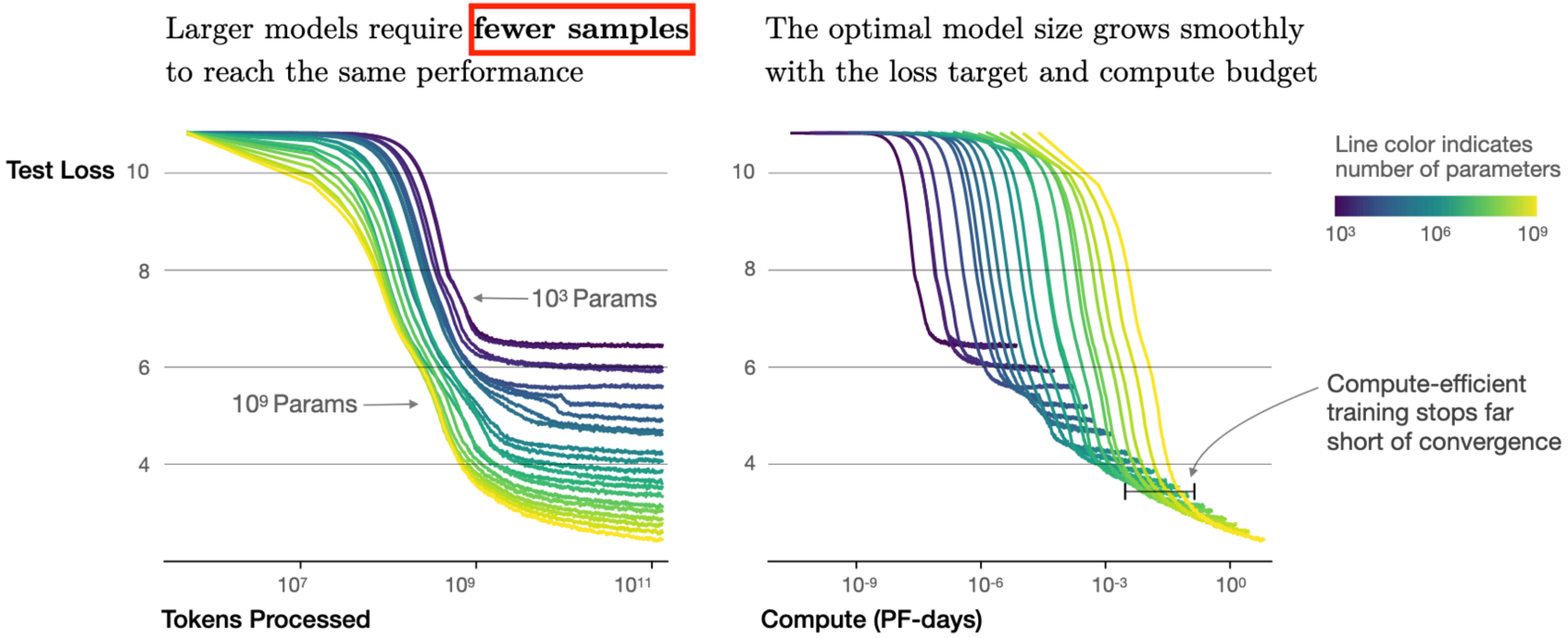
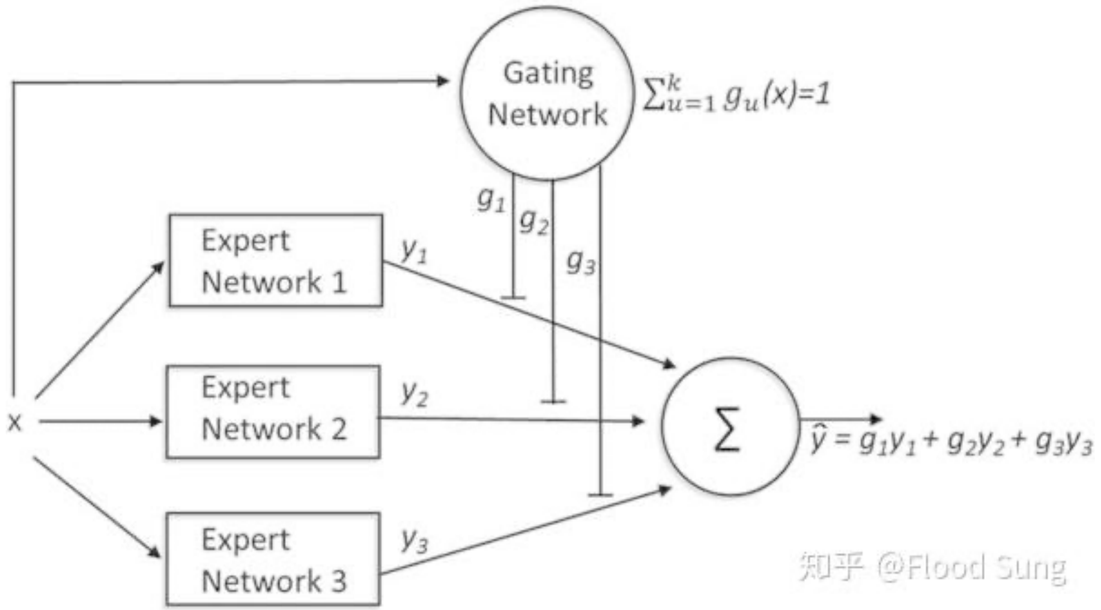


Figure 2 We show a series of language model training runs, with models ranging in size from 10^3 to 10^9 parameters (excluding embeddings).

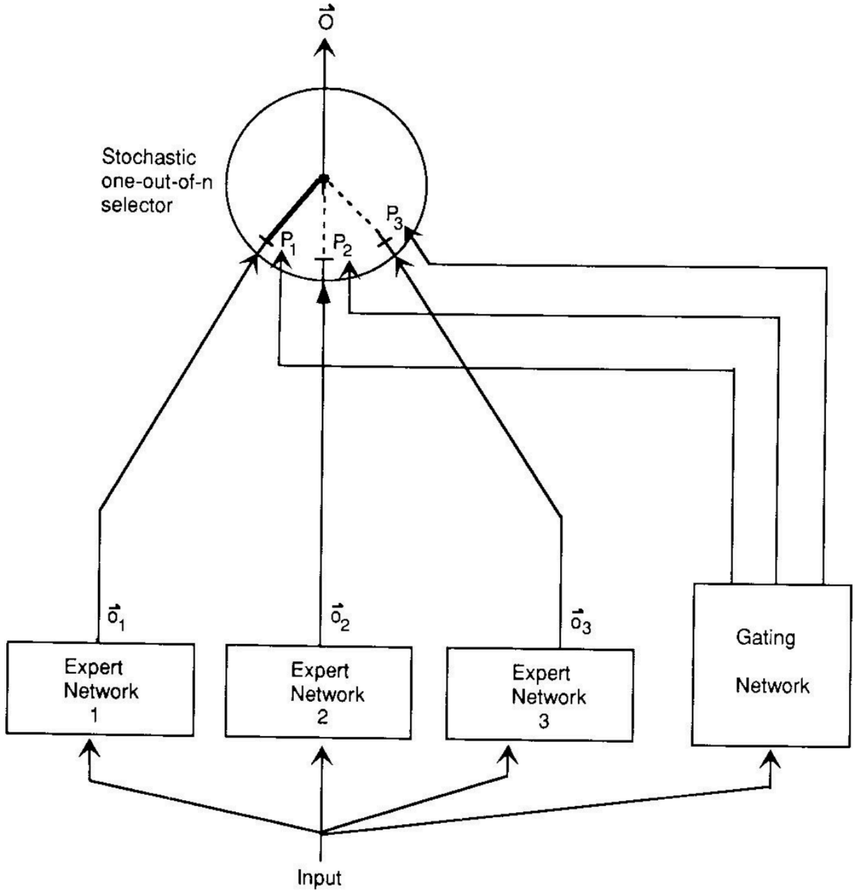
Adaptive Mixtures of Local Experts

Motivation: 引入了 competitive learning，改进 Mixture of Experts



所有不同的 expert network 加权输出最后的结果，这样的处理存在一个问题是不同的 expert 是互相影响的，只要一个 expert 发生变化，整个梯度就会发生改变，所有的 expert 也会跟着更新。

除了让 expert network 互相合作，还可以互相竞争，也就是改成希望每次通过 gating network 只选择一个 expert，即希望每个 expert 处理一种类型的输入



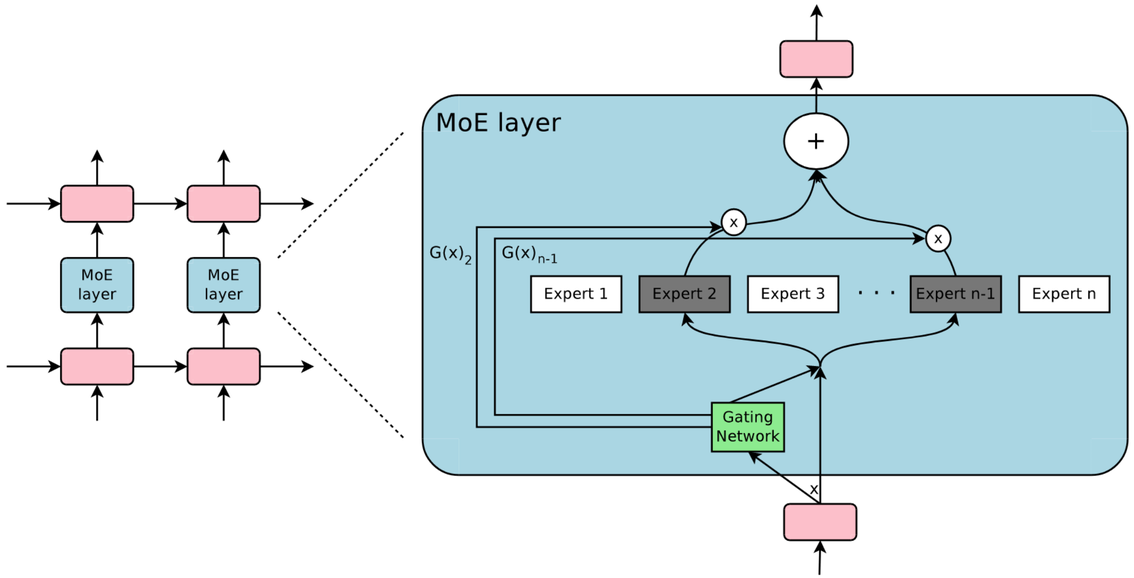
$$E^c = <\|\vec{d}^c - \vec{o}_i^c\|^2> = \sum_i p_i^c \|\vec{d}^c - \vec{o}_i^c\|^2$$

$$\frac{\partial E^c}{\partial \vec{o}_i^c} = -2p_i^c(\vec{d}^c - \vec{o}_i^c)$$

$$E^c = -\log \sum_i p_i^c e^{-\frac{1}{2}\|\vec{d}^c - \vec{o}_i^c\|^2}$$

$$\frac{\partial E^c}{\partial \vec{o}_i^c} = -\left[\frac{p_i^c e^{-\frac{1}{2}\|\vec{d}^c - \vec{o}_i^c\|^2}}{\sum_j p_j^c e^{-\frac{1}{2}\|\vec{d}^c - \vec{o}_j^c\|^2}}\right](\vec{d}^c - \vec{o}_i^c)$$

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



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

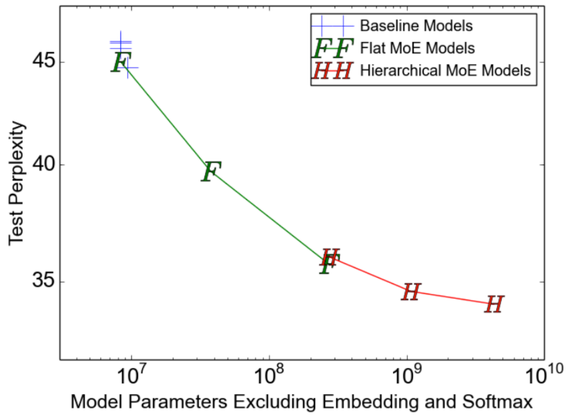
$$H(x)_i = (x \cdot W_g)_i + StandardNormal() \cdot Softplus((x \cdot W_{noise})_i)$$

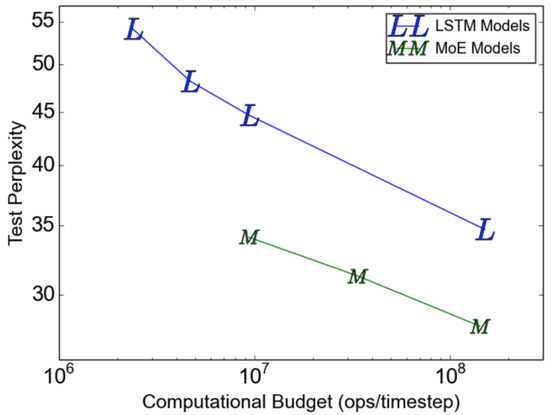
$$KeepTopK(v, k)_i = \begin{cases} v_i & \text{if } v_i \text{ is in the top } k \text{ elements of } v. \\ -\infty & \text{otherwise.} \end{cases}$$

- 平均每个 expert 的 batch size 变小:
  - Data Parallelism and Model Parallelism
  - Token level MoE, all tokens are stacked into one batch
- Expert 负载均衡

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

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

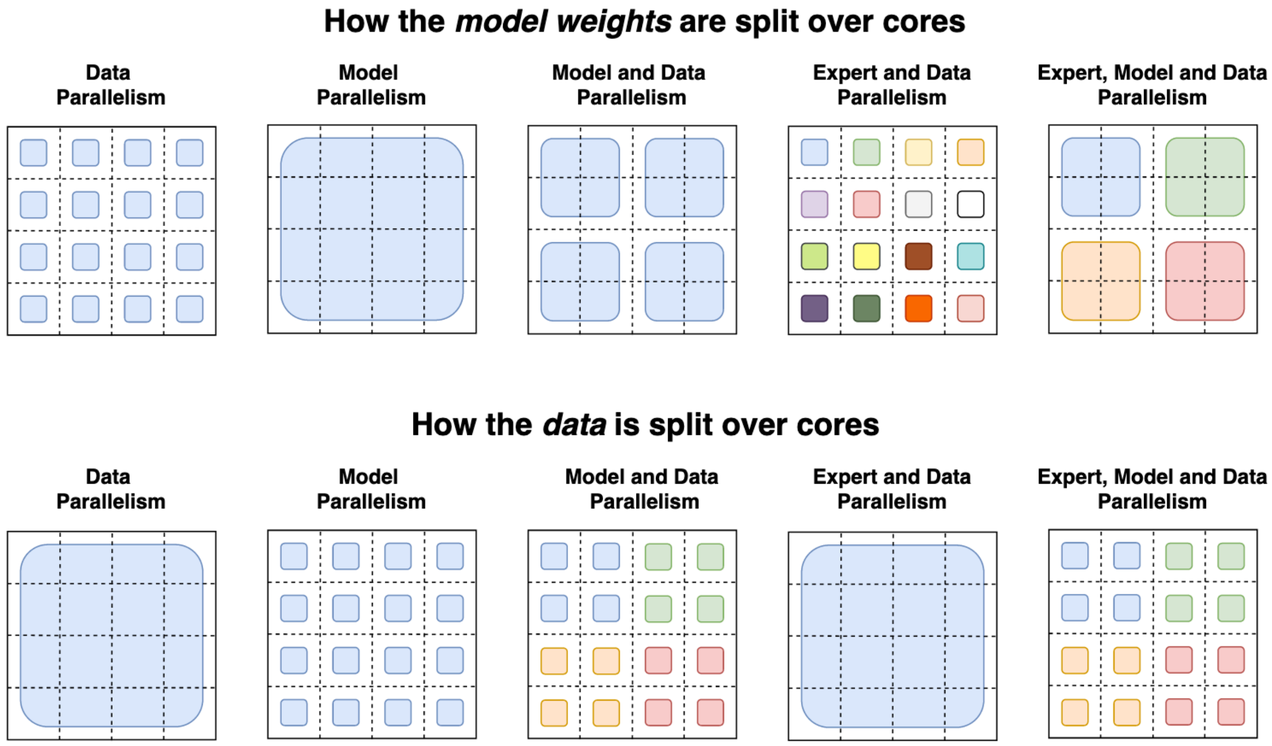


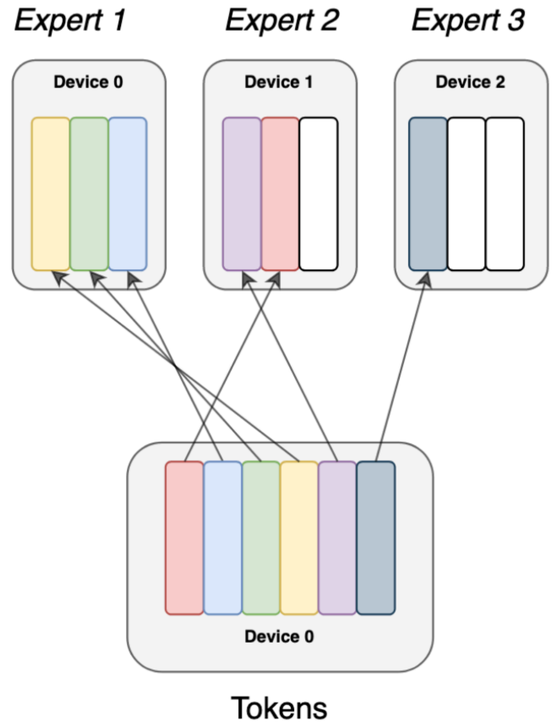


	Test Perplexity 10 epochs	Test Perplexity 100 epochs	#Parameters excluding embedding and softmax layers	ops/timestep	Training Time 10 epochs	TFLOPS /GPU
Best Published Results	34.7	30.6	151 million	151 million	59 hours, 32 k40s	1.09
Low-Budget MoE Model	34.1		4303 million	8.9 million	15 hours, 16 k40s	0.74
Medium-Budget MoE Model	31.3		4313 million	33.8 million	17 hours, 32 k40s	1.22
High-Budget MoE Model	**28.0**		4371 million	142.7 million	47 hours, 32 k40s	**1.56**

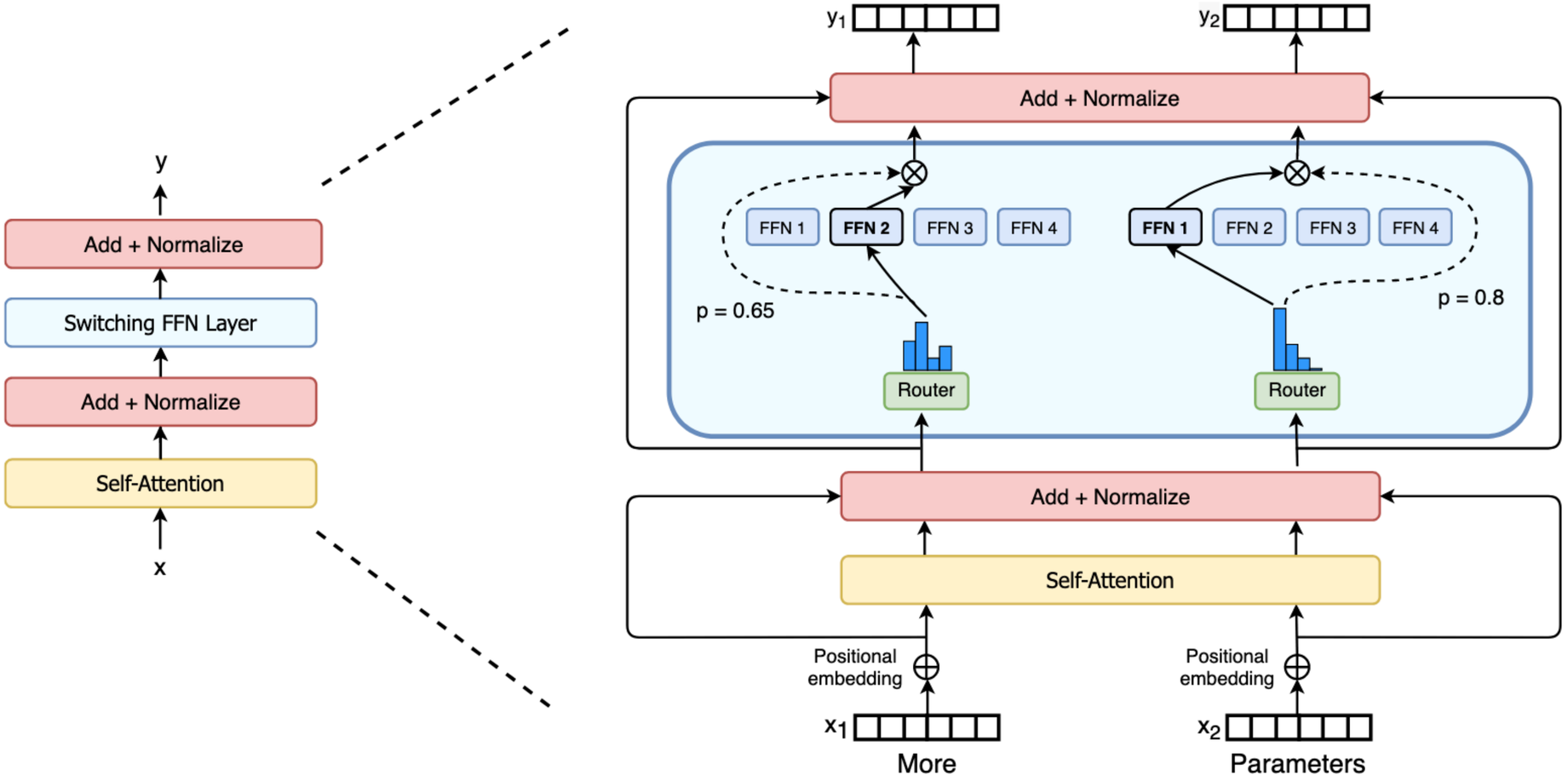
Disadvantages of MoE

- **Model Complexity:** Hundreds or thousands of expert models. Activate different expert models based on the routing network.
- **High Communication Cost:** In expert parallelism, the training bottleneck can be the high inter-expert communication cost.
- **Training Instability:** precision, model initialization and so on.
- **Mapping of MoE and hardware:** Expert Parallelism. Large matrix multiplication is a computationally intensive operation that can benefit from GPU-based computation.





Switch Transformer Layer



Improved Training Methodology

- Selective precision
- Reduced initialization scale and slower learning rate warmup
- Higher regularization of experts
- Differentiable auxiliary loss to balance the load across experts

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

Table 2: Selective precision. We cast the local routing operations to float32 while preserving bfloat16 precision elsewhere to stabilize our model while achieving nearly equal speed to (unstable) bfloat16-precision training. We measure the quality of a 32 expert model after a fixed step count early in training its speed performance. For both Switch-Base in float32 and with Selective prevision we notice similar learning dynamics.

Improved Training Methodology

- Selective precision
- Reduced initialization scale and slower learning rate warmup
- Higher regularization of experts
- Differentiable auxiliary loss to balance the load across experts

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

Table 3: Reduced initialization scale improves stability. Reducing the initialization scale results in better model quality and more stable training of Switch Transformer. Here we record the average and standard deviation of model quality, measured by the negative log perplexity, of a 32 expert model after 3.5k steps (3 random seeds each).

Improved Training Methodology

- Selective precision
- Reduced initialization scale and slower learning rate warmup
- Higher regularization of experts
- Differentiable auxiliary loss to balance the load across experts

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.

Improved Training Methodology

- Selective precision
- Reduced initialization scale and slower learning rate warmup
- Higher regularization of experts
- Differentiable auxiliary loss to balance the load across experts

A Differentiable Load Balancing Loss. To encourage a balanced load across experts we add an auxiliary loss (Shazeer et al., 2017, 2018; Lepikhin et al., 2020). As in Shazeer et al. (2018); Lepikhin et al. (2020), Switch Transformers simplifies the original design in Shazeer et al. (2017) which had separate load-balancing and importance-weighting losses. For each Switch layer, this auxiliary loss is added to the total model loss during training. 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,

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

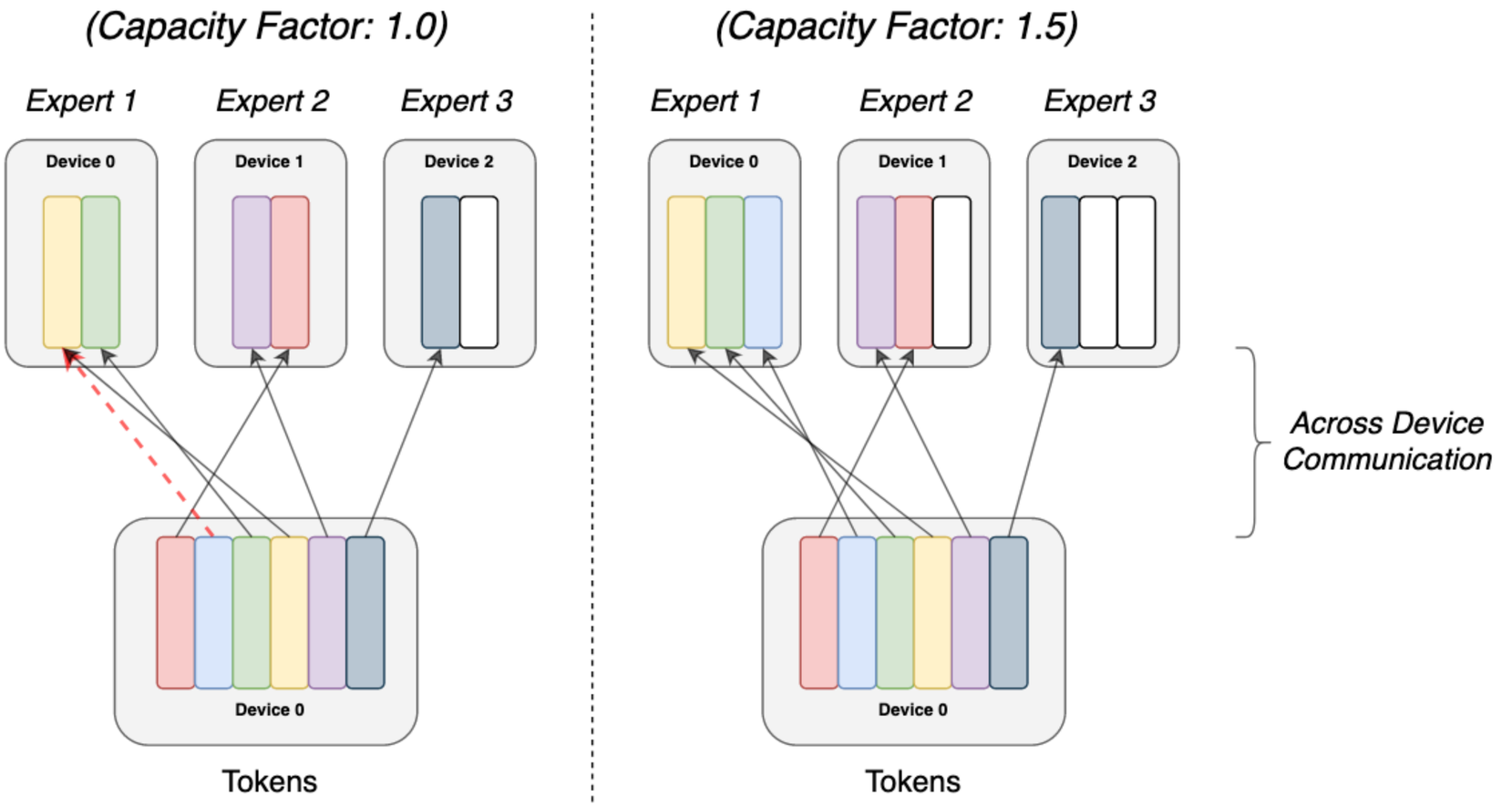
and P_i is the fraction of the router probability allocated for expert i, [2]

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

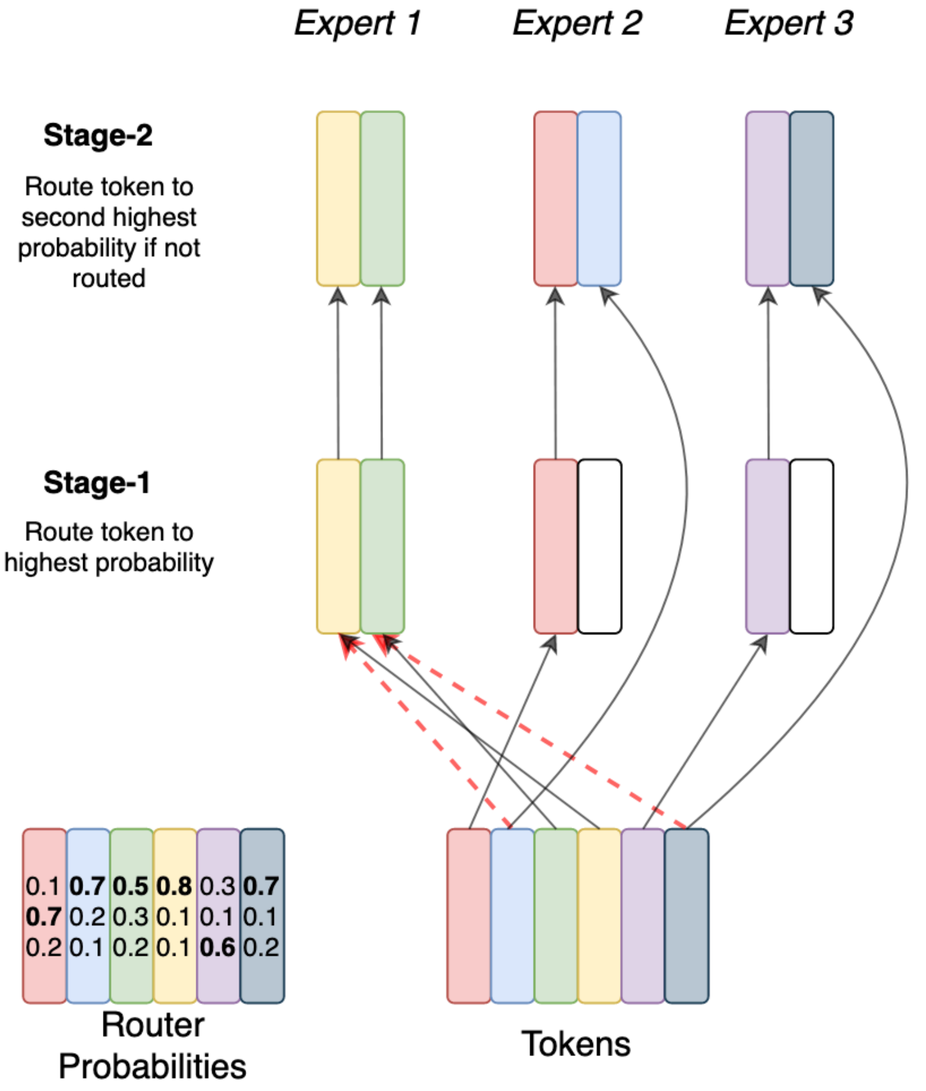
Issue: Statically Compiled Graph, but a Dynamic Architecture

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.



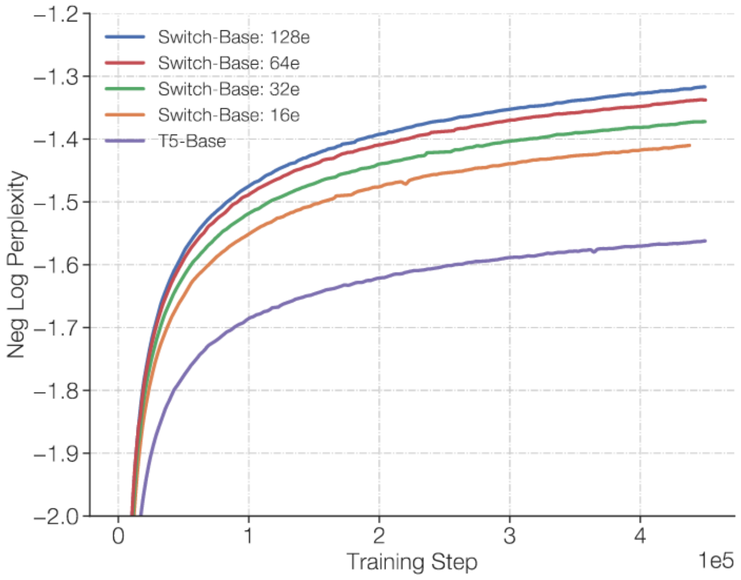
No-Token-Left-Behind

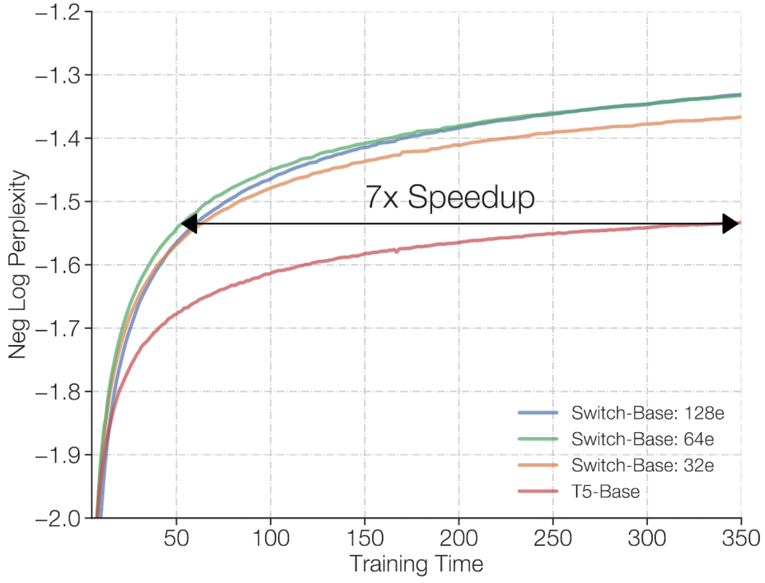


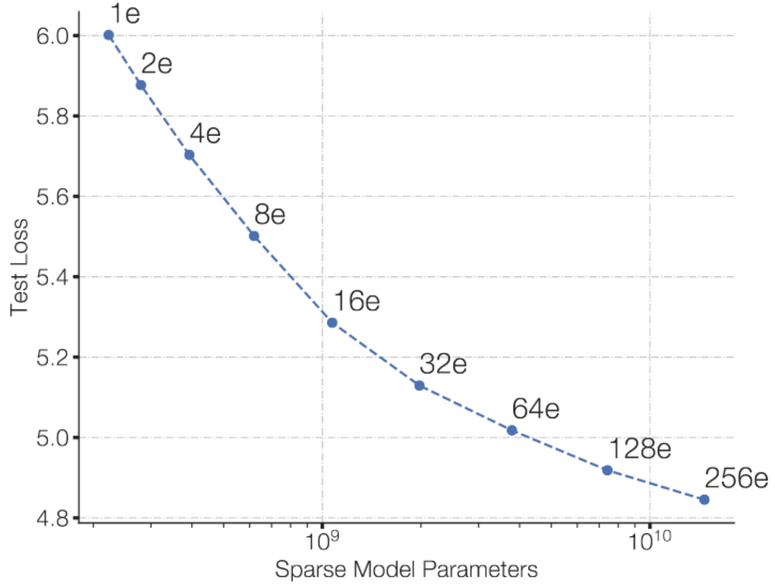
Comparison of MoE (top-2) and Switch (top-1)

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

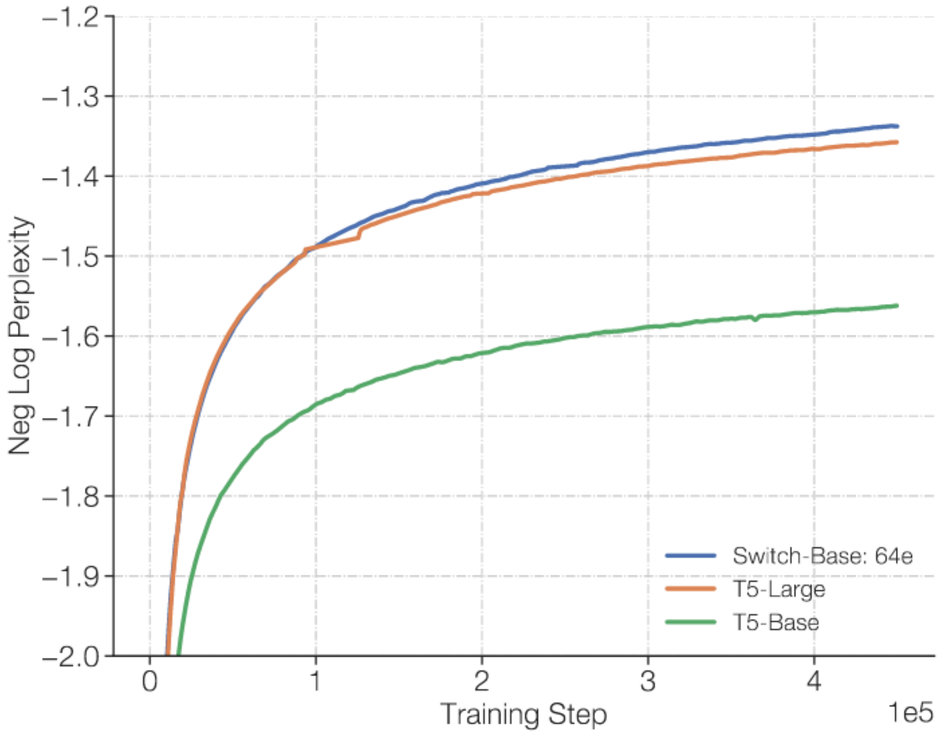
Scaling Up Number of Experts

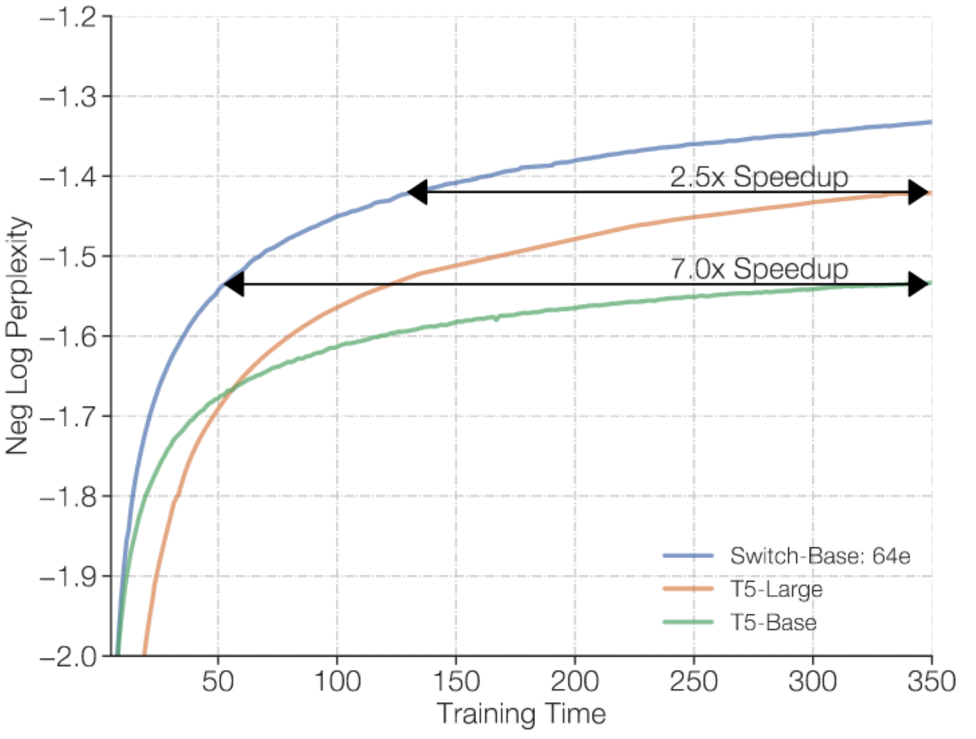






Expert-Parallelism vs. Model-Parallelism





Design Choices Switch Transformers

Model	Parameters	FLOPs/seq	d_{model}	FFN_{GEGLU}	d_{ff}	d_{kv}
T5-XXL	13B	8.7T	4096	✓	10240	64
Switch-XXL	395B	8.7T	4096	✓	10240	64
Switch-C	1571B	890B	2080		6144	64

Model	Num. Heads	Num. Layers	Num. Experts	Neg. Log Perp. @250k	Neg. Log Perp. @ 500k
T5-XXL	128	24	–	-1.147	-1.095
Switch-XXL	128	24	64	**-1.086**	**-1.008**
Switch-C	32	15	2048	-1.096	-1.043

My mostly unsubstantiated theory is that parameters are good for "knowledge" and compute [FLOPs] is good for "intelligence" whatever those terms mean.

— Noam Shazeer

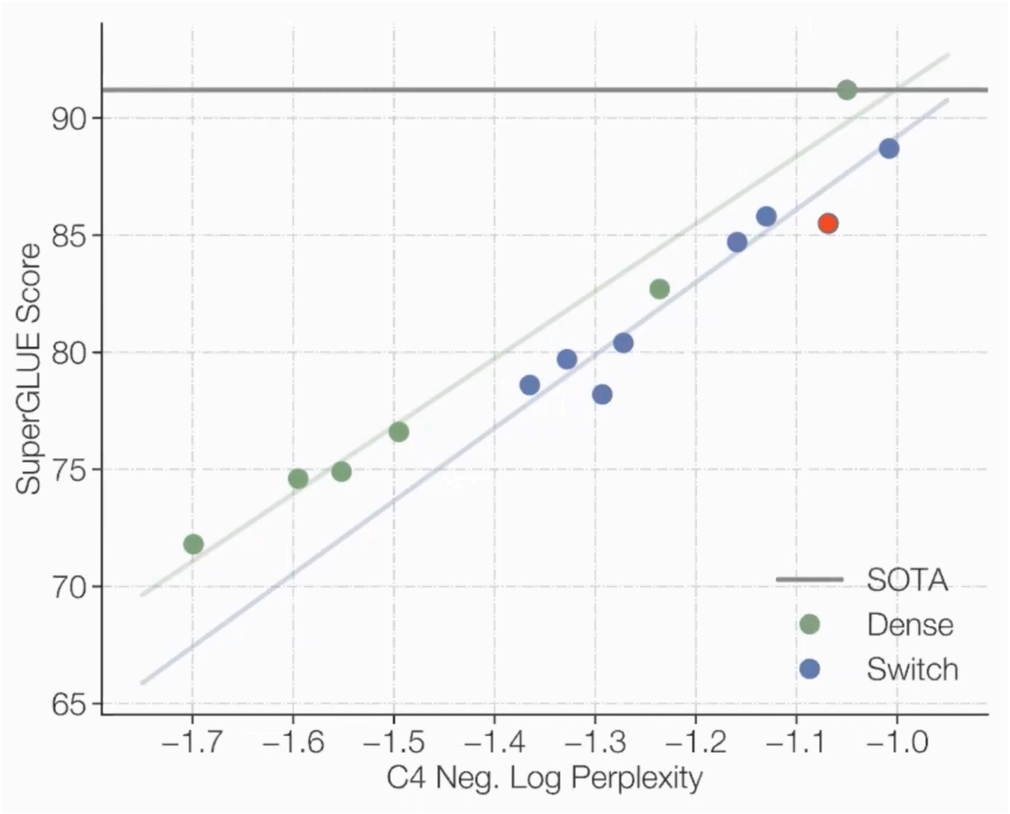
For a fixed quality on an upstream pre-training task, do parameters independently matter?

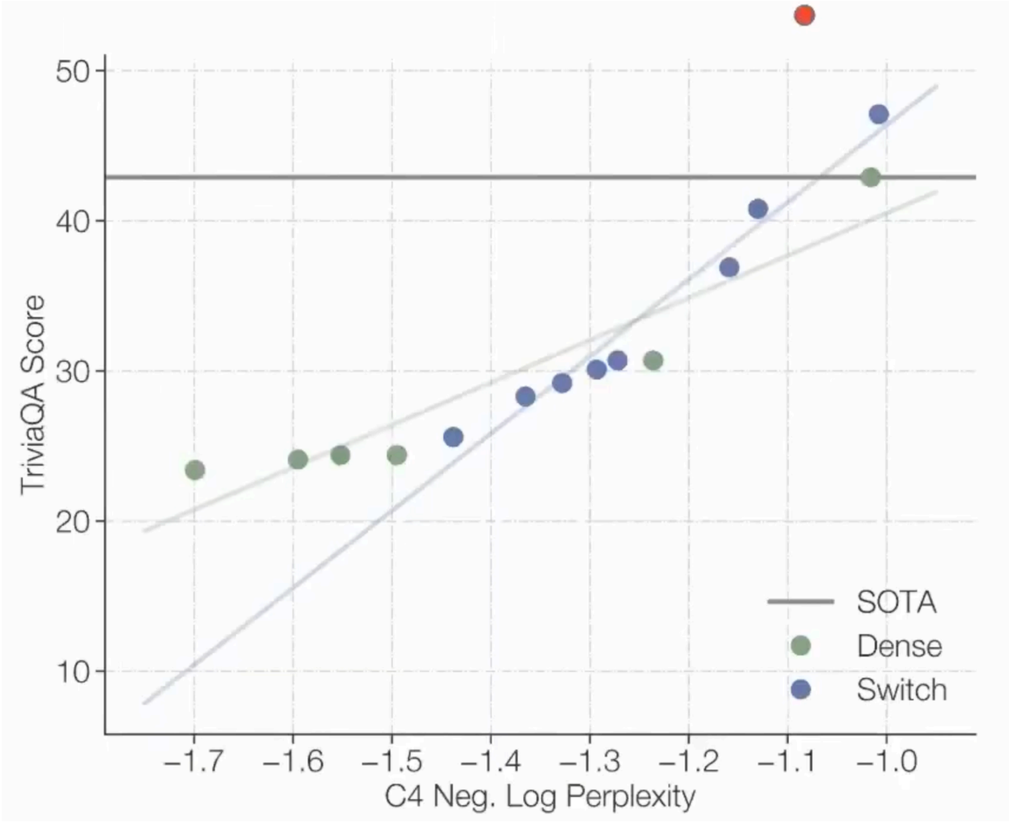
Upstream task: fill-in-the-blank on scraped web pages (C_4 corpus)

Downstream tasks:

- SuperGLUE (reasoning proxy)
- TriviaQA (knowledge proxy)

Upstream Quality —> SuperGLUE, TriviaOA Quality





Fine-tuning

Model	GLUE	SQuAD	SuperGLUE	Winogrande (XL)
T5-Base	84.3	85.5	75.1	66.6
Switch-Base	**86.7**	**87.2**	**79.5**	**73.3**
T5-Large	87.8	88.1	82.7	79.1
Switch-Large	**88.5**	**88.6**	**84.7**	**83.0**

Model	XSum	ANLI (R3)	ARC Easy	ARC Chal.
T5-Base	18.7	51.8	56.7	**35.5**
Switch-Base	**20.3**	**54.0**	**61.3**	32.8
T5-Large	20.9	56.6	**68.8**	**35.5**
Switch-Large	**22.3**	**58.6**	66.0	**35.5**

Model	CB Web QA	CB Natural QA	CB Trivia QA
T5-Base	26.6	25.8	24.5
Switch-Base	**27.4**	**26.8**	**30.7**
T5-Large	27.7	27.6	29.5
Switch-Large	**31.3**	**29.5**	**36.9**

Table 5: Fine-tuning results. Fine-tuning results of T5 baselines and Switch models across a diverse set of natural language tests (validation sets; higher numbers are better). We compare FLOP-matched Switch models to the T5-Base and T5-Large baselines. For most tasks considered, we find significant improvements of the Switch-variants. We observe gains across both model sizes and across both reasoning and knowledge-heavy language tasks.

Multilingual pre-training

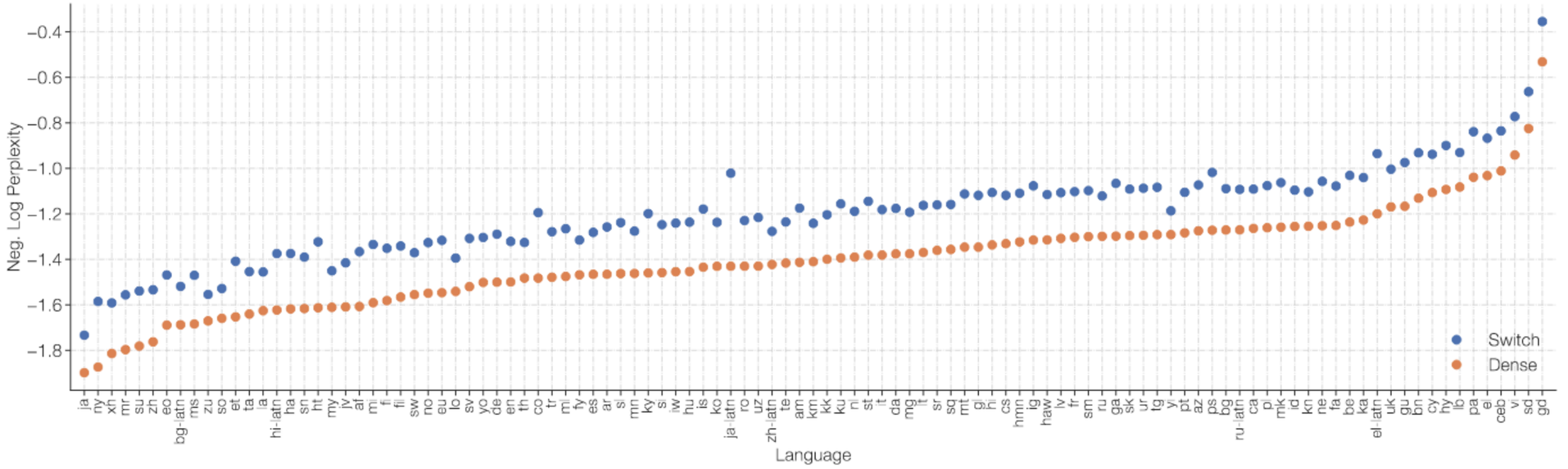


Figure 7: Multilingual pre-training on 101 languages. Improvements of Switch T5 Base model over dense baseline when multi-task training on 101 languages. We observe Switch Transformers to do quite well in the multi-task training setup and yield improvements on all 101 languages.

Distilling technique

Technique	Parameters	Quality (↑)
T5-Base	223M	-1.636
Switch-Base	3,800M	-1.444
Distillation	223M	(3%) -1.631
+ Init. non-expert weights from teacher	223M	(20%) -1.598
+ 0.75 mix of hard and soft loss	223M	(29%) -1.580
Initialization Baseline (no distillation)		
Init. non-expert weights from teacher	223M	-1.639

	Dense	Sparse				
Parameters	223M	1.1B	2.0B	3.8B	7.4B	14.7B
Pre-trained Neg. Log Perp. (↑)	-1.636	-1.505	-1.474	-1.444	-1.432	-1.427
Distilled Neg. Log Perp. (↑)	—	-1.587	-1.585	-1.579	-1.582	-1.578
Percent of Teacher Performance	—	37%	32%	30 %	27 %	28 %
Compression Percent	—	82 %	90 %	95 %	97 %	99 %