Natural Language Processing

Scaling Law

Instructor: Yangqiu Song

Slides credits: Anna Goldie, John Hewitt, Carter Blair and Cole Wyeth, Wenhu Chen

Outline

- The Scaling Law
  - Scaling Laws
    - Kaplan et al., 2020 (OpenAI)
    - Hoffman et al., 2022 (DeepMind)
    - DiLoCo (Google Research and DeepMind)
  - Emergent Abilities
    - Wei et al., 2022

Scaling Laws

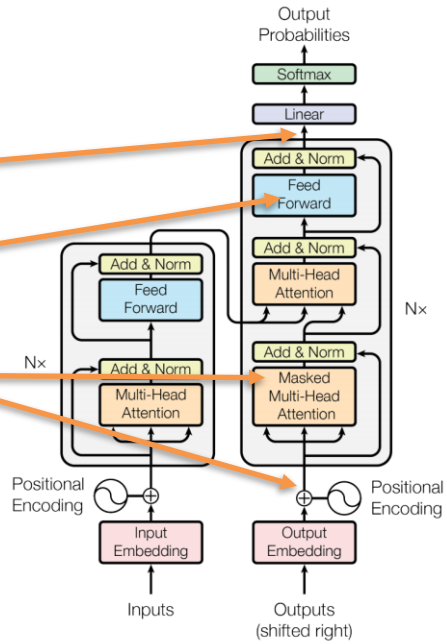
Scaling Laws for Neural Language Models, Kaplan et al. 2020 (OpenAI)

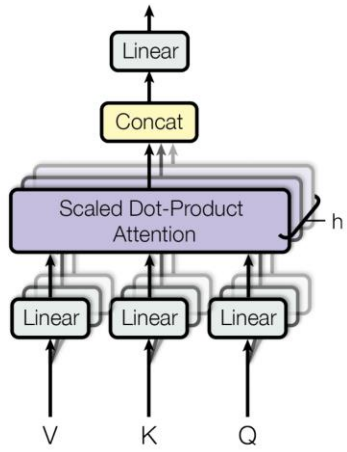
Main RQ: What is the relationship between compute C, data D, the number of parameters N, and performance L (loss)?

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. arXiv preprint arXiv:2001.08361, 2020.

Notations

- Performance mostly depends on scale
- d_{model}: dimension of residual stream
- d_{ff}: dimension of intermediate feed forward layer
- d_{attn}: dimension of the attention output
- n_{layer}: number of layers
- n_{heads}: number of attention heads per layer (merged in the d_{attn} in following computation)
- Embedding matrix $n_{\text{vocab}} d_{\text{model}}$
- positional embeddings $n_{\text{ctx}} d_{\text{model}}$
- Non-embedding parameter count N





$$N \approx 2 d_{\text{model}} n_{\text{layer}} \left(2 d_{\text{attn}} + d_{\text{ff}}\right)$$
$$= 12 n_{\text{layer}} d_{\text{model}}^2 \quad \text{with the standard} \quad d_{\text{attn}} = d_{\text{ff}}/4 = d_{\text{model}}$$

Operation	Parameters	FLOPs per Token
Embed	$(n_{\text{vocab}} + n_{\text{ctx}})\, d_{\text{model}}$	$4 d_{\text{model}}$
Attention: QKV	$n_{\text{layer}} d_{\text{model}} 3 d_{\text{attn}}$	$2 n_{\text{layer}} d_{\text{model}} 3 d_{\text{attn}}$
Attention: Mask	—	$2 n_{\text{layer}} n_{\text{ctx}} d_{\text{attn}}$
Attention: Project	$n_{\text{layer}} d_{\text{attn}} d_{\text{model}}$	$2 n_{\text{layer}} d_{\text{attn}} d_{\text{embd}}$
Feedforward	$n_{\text{layer}} 2 d_{\text{model}} d_{\text{ff}}$	$2 n_{\text{layer}} 2 d_{\text{model}} d_{\text{ff}}$
De-embed	—	$2 d_{\text{model}} n_{\text{vocab}}$
Total (Non-Embedding)	$N = 2 d_{\text{model}} n_{\text{layer}} \left(2 d_{\text{attn}} + d_{\text{ff}}\right)$	$C_{\text{forward}} = 2N + 2 n_{\text{layer}} n_{\text{ctx}} d_{\text{attn}}$

- FLOPs = **Fl**oating point **op**erations

Model Shape

- Hold total non-embedding parameter $N \approx 12 n_{layer} d_{model}^2$ fixed
  - Transformer performance depends very weakly on the shape parameters.
  - n_{heads}: number of attention heads per layer merged in the d_{attn}

$$N \approx 2 d_{\text{model}} n_{\text{layer}} \left(2 d_{\text{attn}} + d_{\text{ff}}\right)$$
$$= 12 n_{\text{layer}} d_{\text{model}}^2 \quad \text{with the standard} \quad d_{\text{attn}} = d_{\text{ff}}/4 = d_{\text{model}}$$

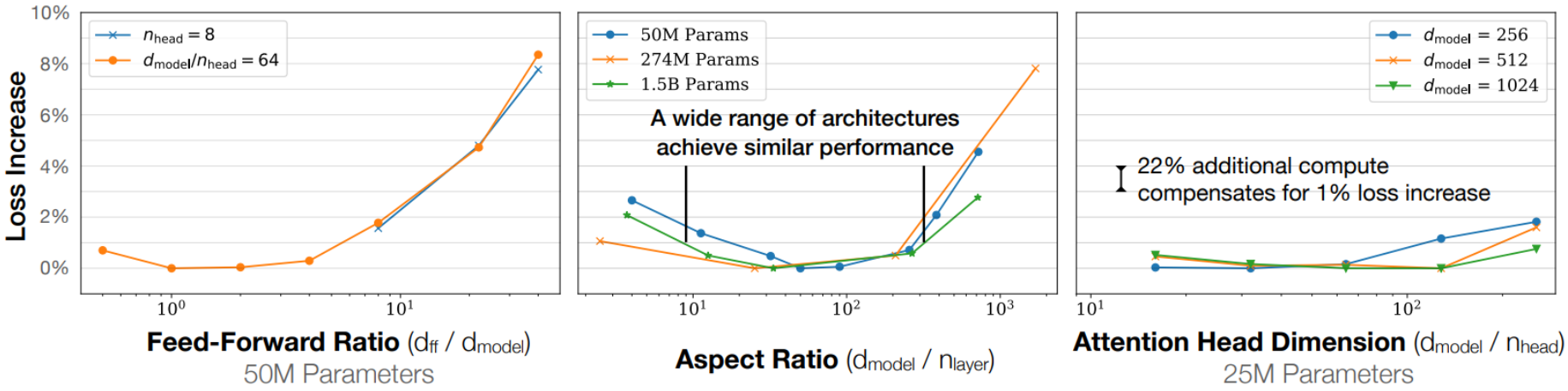


Figure 5 Performance depends very mildly on model shape when the total number of non-embedding parameters N is held fixed. The loss varies only a few percent over a wide range of shapes. Small differences in parameter counts are compensated for by using the fit to $L(N)$ as a baseline. Aspect ratio in particular can vary by a factor of 40 while only slightly impacting performance; an $(n_{\text{layer}}, d_{\text{model}}) = (6, 4288)$ reaches a loss within 3% of the $(48, 1600)$ model used in [RWC+19].

Empirical Results and Basic Power Laws

- Performance has a power-law relationship with each of the three scale factors N, D, C when not bottlenecked by the other two, with trends spanning more than six orders of magnitude

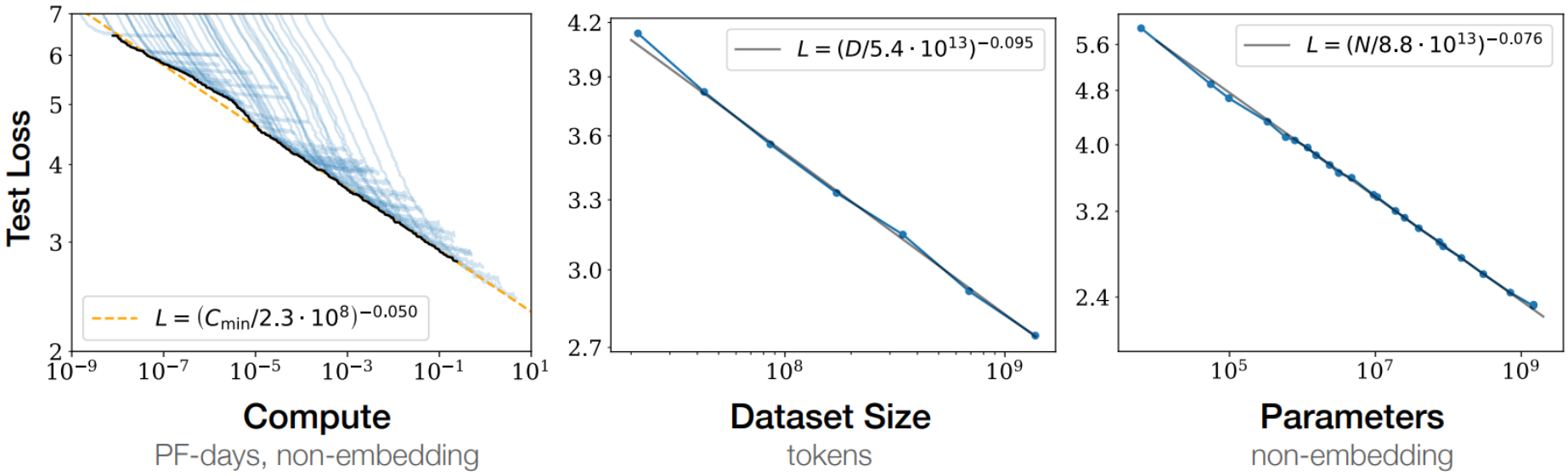


Figure 1 Language modeling performance improves smoothly as we increase the model size, datasetset size, and amount of compute[2] used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when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

Infinite Data Limit and Overfitting

- Proposed L(N, D) Equation

$$L(N, D) = \left[\left(\frac{N_c}{N} \right)^{\frac{\alpha_N}{\alpha_D}} + \frac{D_c}{D} \right]^{\alpha_D}$$

- Curve fitting results

Parameter	α_N	α_D	N_c	D_c
Value	0.076	0.103	6.4×10^{13}	1.8×10^{13}

Table 2 Fits to $L(N, D)$

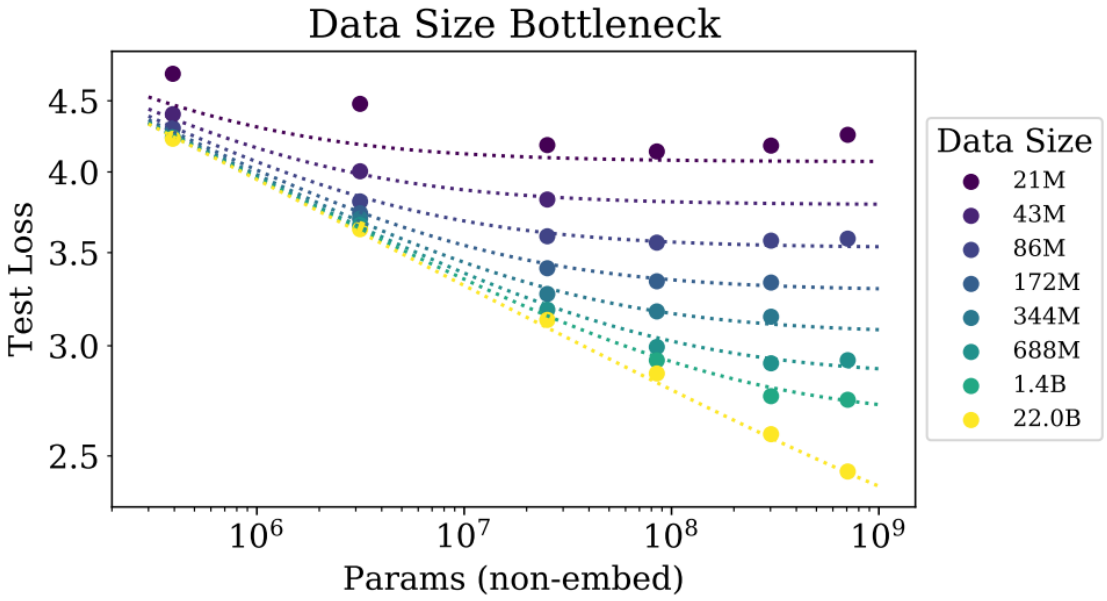
The precise numerical values of Nc and Dc depend on the vocabulary size and tokenization and hence do not have a fundamental meaning.

Infinite Data Limit and Overfitting

- With some analytics of the extent of overfitting of curve L(N,D), we should scale the dataset size as

$$D \gtrsim (5 \times 10^3)\, N^{0.74}$$

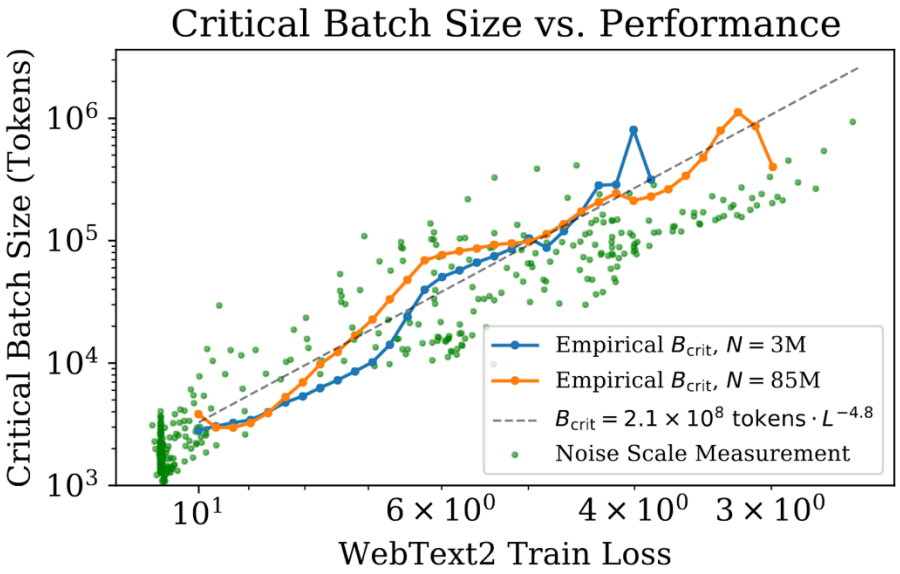
- That means, every time we increase the model size 8x, we only need to increase the data by roughly 5x to avoid a penalty.



Critical Batch Size

- The critical batch size can be fit with a power-law in the loss

$$B_{\text{crit}}(L) \approx \frac{B_*}{L^{1/\alpha_B}} \qquad B_* \approx 2 \times 10^8 \text{ and } \alpha_B \approx 0.21.$$

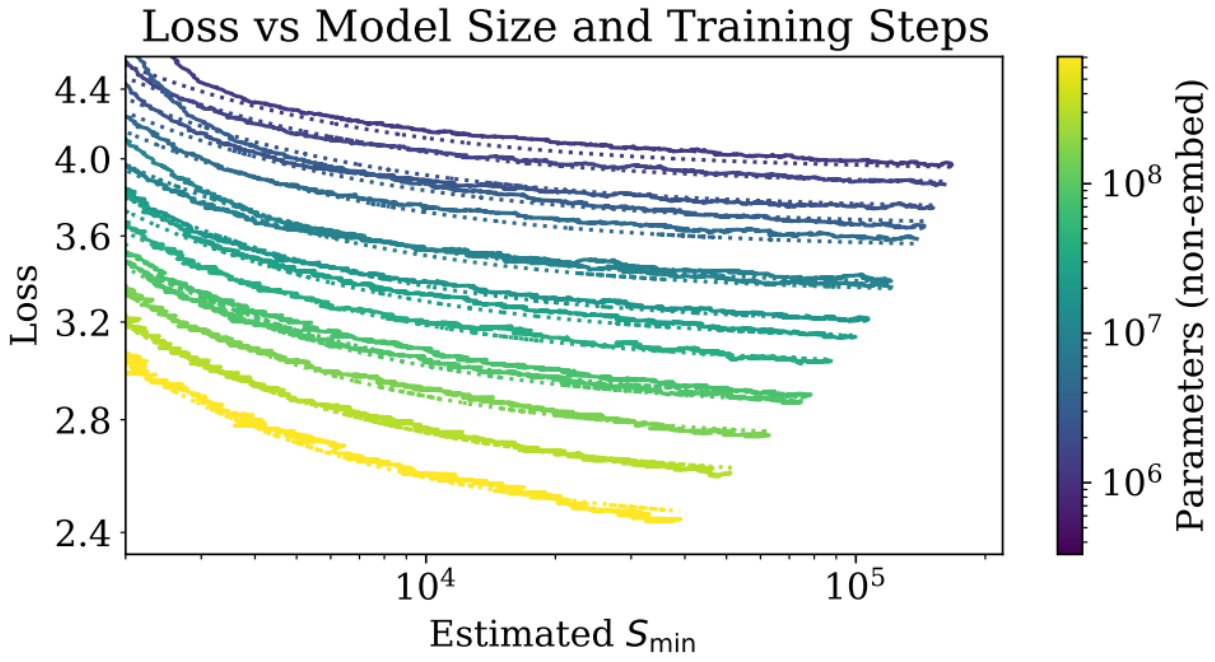


It is roughly 1-2 million tokens at convergence for the largest models we can train

Figure 10 The critical batch size B_{crit} follows a power law in the loss as performance increase, and does not depend directly on the model size. We find that the critical batch size approximately doubles for every 13% decrease in loss. B_{crit} is measured empirically from the data shown in Figure 18, but it is also roughly predicted by the gradient noise scale, as in [MKAT18].

Scaling Laws with Model Size and Training Time

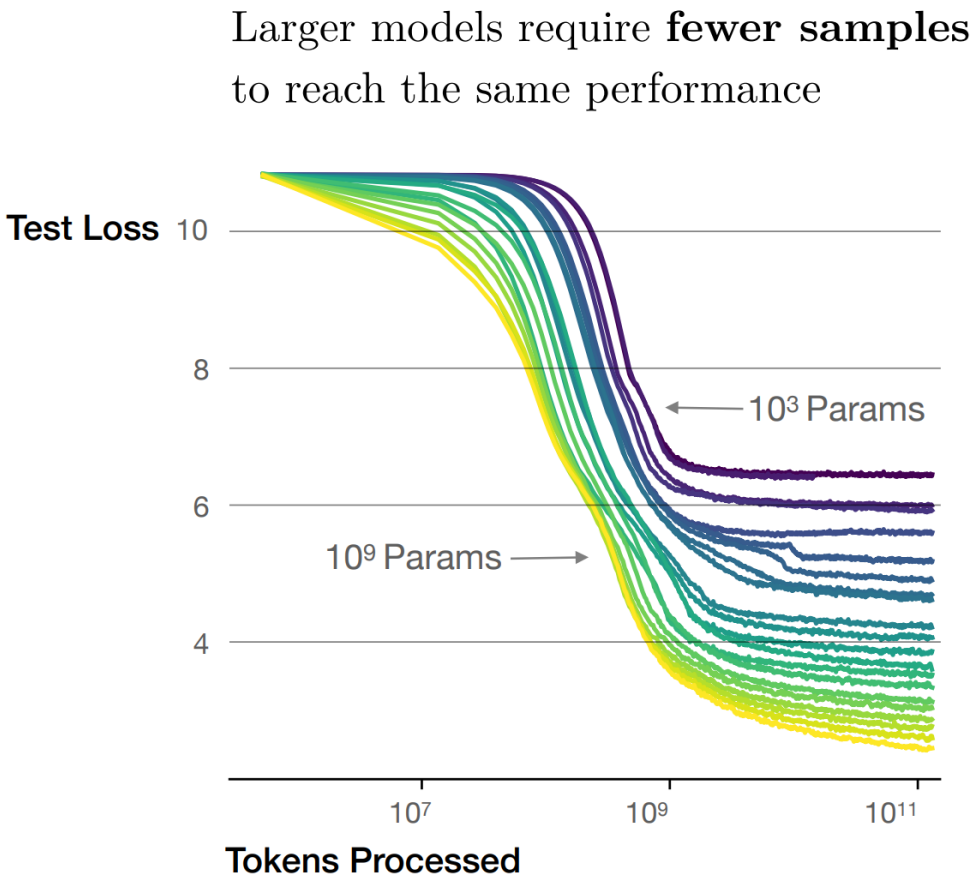
- Smin: minimum possible number of optimization steps which accounts for the fact that most of our models have not been trained at an optimal batch size



batch size affects the estimated minimum steps

Sample Efficiency

- **Large models are more sample-efficient than small models**, reaching the same level of performance with fewer data points



Scaling Laws with Model Size and Training Time

- L(N, Smin) and Performance with Model Size and Compute
  - S: parameter update steps in the infinite data limit

$$L(N, S_{\min}) = \left(\frac{N_c}{N}\right)^{\alpha_N} + \left(\frac{S_c}{S_{\min}}\right)^{\alpha_S}$$

Parameter	α_N	α_S	N_c	S_c
Value	0.077	0.76	6.5×10^{13}	2.1×10^3

Table 3 Fits to $L(N, S)$

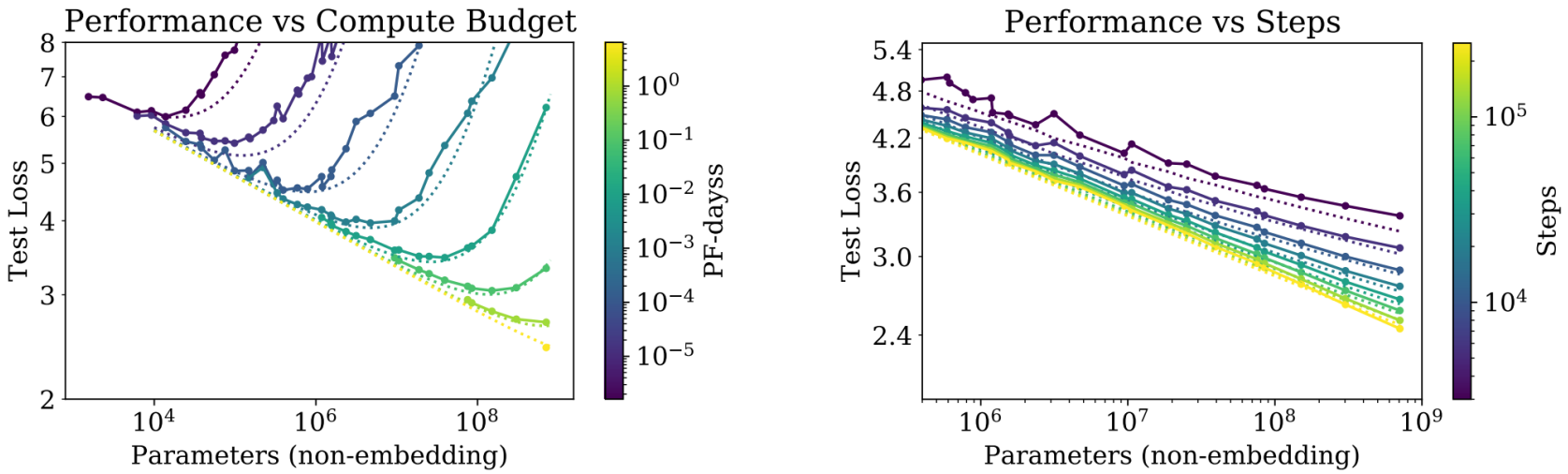


Figure 11 When we hold either total compute or number of training steps fixed, performance follows $L(N, S)$ from Equation (5.6). Each value of compute budget has an associated optimal model size that maximizes performance. Mediocre fits at small S are unsurprising, as the power-law equation for the learning curves breaks down very early in training.

Training Compute Optimal Large Language Models, Hoffmann et. al. 2022 (DeepMind)

Main RQ: What is the optimal tradeoff between model size and number of tokens given a fixed compute budget?

Hoffman et. al. 2022

- A better learning rate schedule leads to experimental results drastically different from Kaplan et. al., 2020
- ". . . we find that setting the learning rate schedule to approximately match the number of training tokens results in the best final loss regardless of model size." (Hoffman et. al. 2022)
- They used learning rate cosine schedule

DeepSeek

- “As for 22 the learning rate scheduling, we first linearly increase it from 0 to 2.2 × 10^−4 during the first 2K steps. Then, we keep a constant learning rate of 2.2 × 10^−4 until the model consumes 10T training tokens. Subsequently, we gradually decay the learning rate to 2.2 × 10^−5 in 4.3T tokens, following a cosine decay curve. During the training of the final 500B tokens, we keep a constant learning rate of 2.2 × 10^−5 in the first 333B tokens, and switch to another constant learning rate of 7.3 × 10^−6 in the remaining 167B tokens. The gradient clipping norm is set to 1.0.”

Modelling the Scaling Behavior

- Goal is to find optimal parameter count N_{opt} and token count D_{opt} given a fixed compute budget C

$$N_{opt}(C), D_{opt}(C) = \underset{N,D \text{ s.t. } \mathrm{FLOPs}(N,D)=C}{\operatorname{argmin}} L(N, D).$$

- The authors tried three approaches to estimating these values

Approach 1: Fix model sizes and vary number of training tokens

- Trained models for each parameter count N with different number of tokens
- Find the minimal loss per flop and plot them on the right two figures
- fit and find estimated number of parameters and tokens for optimal models

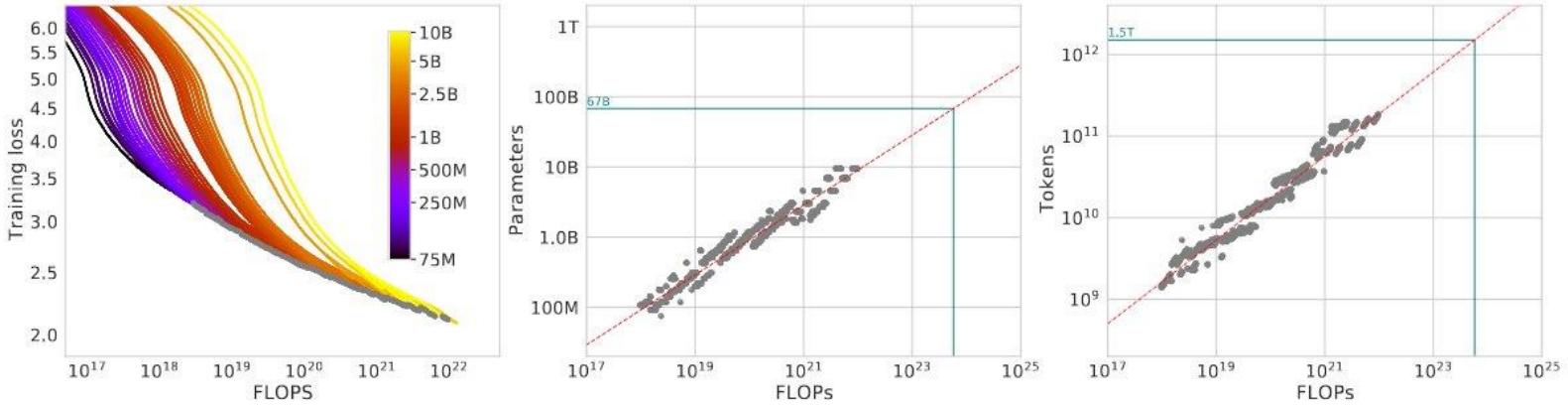


Figure 2 | **Training curve envelope.** On the **left** we show all of our different runs. We launched a range of model sizes going from 70M to 10B, each for four different cosine cycle lengths. From these curves, we extracted the envelope of minimal loss per FLOP, and we used these points to estimate the optimal model size (**center**) for a given compute budget and the optimal number of training tokens (**right**). In green, we show projections of optimal model size and training token count based on the number of FLOPs used to train *Gopher* (5.76×10^{23}).

$$N_{opt} \propto C^a \text{ and } D_{opt} \propto C^b$$

Approach 2: IsoFLOP profiles

- For each FLOP budget, fits the final training loss
- Find the valley's optimal point for right two figures to fit and find estimated number of parameters and tokens for optimal models

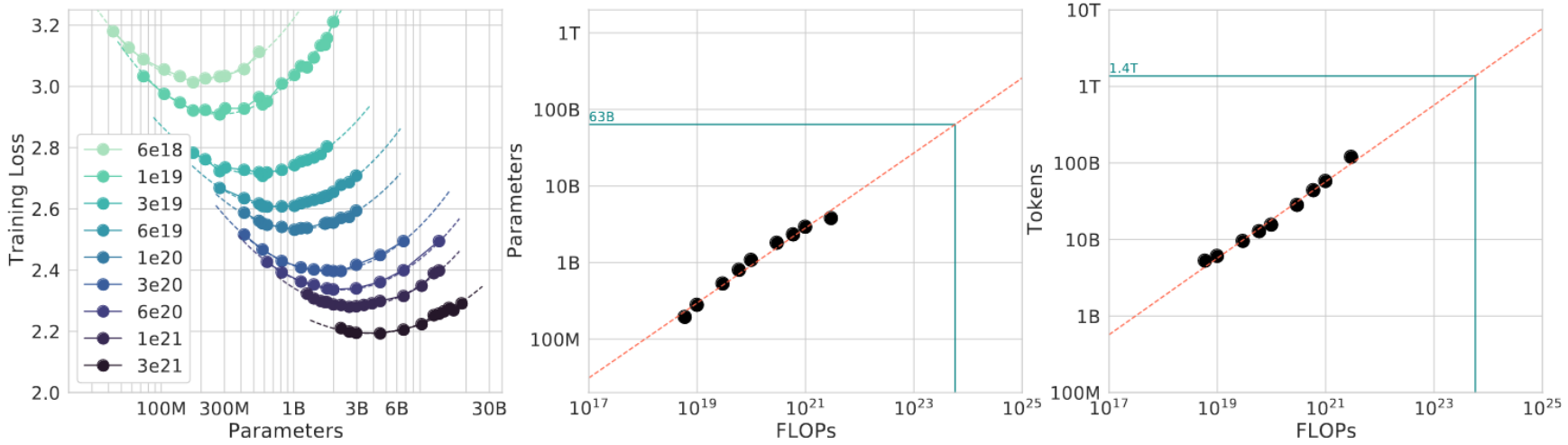


Figure 3 | **IsoFLOP curves.** For various model sizes, we choose the number of training tokens such that the final FLOPs is a constant. The cosine cycle length is set to match the target FLOP count. We find a clear valley in loss, meaning that for a given FLOP budget there is an optimal model to train (**left**). Using the location of these valleys, we project optimal model size and number of tokens for larger models (**center** and **right**). In green, we show the estimated number of parameters and tokens for an *optimal* model trained with the compute budget of *Gopher*.

$$N_{opt} \propto C^a \text{ and } D_{opt} \propto C^b$$

Approach 3 : Fitting a parametric loss function

- Fit an explicit model by minimizing Huber loss

$$\hat{L}(N, D) \triangleq E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}}.$$

- Exponents are a = 0.46 and b = 0.54 for N_{opt} and D_{opt} resp.

Hoffman et. al. 2022

- The authors conclude that with *Gopher's* compute budget, a smaller model should have been trained on more tokens
- They train *Chinchilla,* a 70 billion parameter model, on 1.4 trillion tokens
- *Chinchilla's* smaller size reduces inference and finetuning time

Model	Layers	Number Heads	Key/Value Size	d_{model}	Max LR	Batch Size
Gopher 280B	80	128	128	16,384	4×10^{-5}	3M $\rightarrow$ 6M
Chinchilla 70B	80	64	128	8,192	1×10^{-4}	1.5M $\rightarrow$ 3M

Table 4 | ***Chinchilla* architecture details.** We list the number of layers, the key/value size, the bottleneck activation size d_{model}, the maximum learning rate, and the training batch size (# tokens). The feed-forward size is always set to $4 \times d_{model}$. Note that we double the batch size midway through training for both *Chinchilla* and *Gopher*.

DiLoCo (Google)

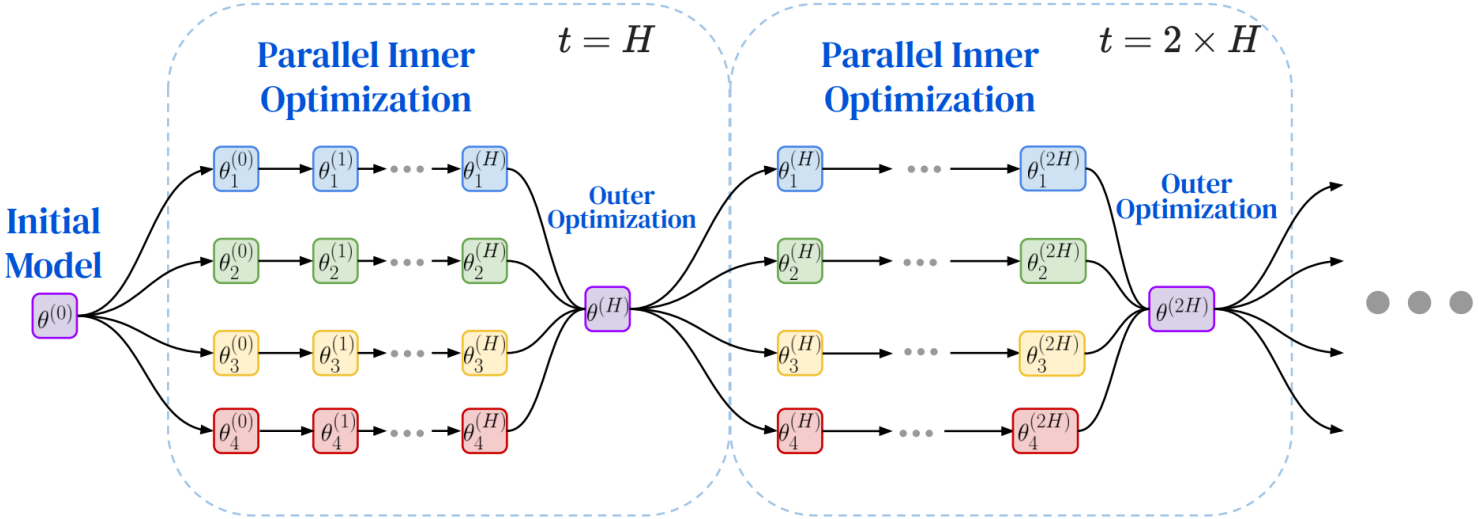
Main RQ: Can we have a better optimization method for better scaling?

Communication-Efficient Language Model Training Scales Reliably and Robustly: Scaling Laws for DiLoCo (Google)

- "As we scale to more massive machine learning models, the frequent synchronization demands inherent in data-parallel approaches create significant slowdowns, posing a critical challenge to further scaling."

- DiLoCo: Distributed low communication training of language models.
  - A variant of the popular Federated Averaging (FedAvg) algorithm, or a particular instantiation with a momentum-based optimizer as in the FedOpt algorithm

DiLoCo operates on M parallel model parameters
H: inner optimization steps

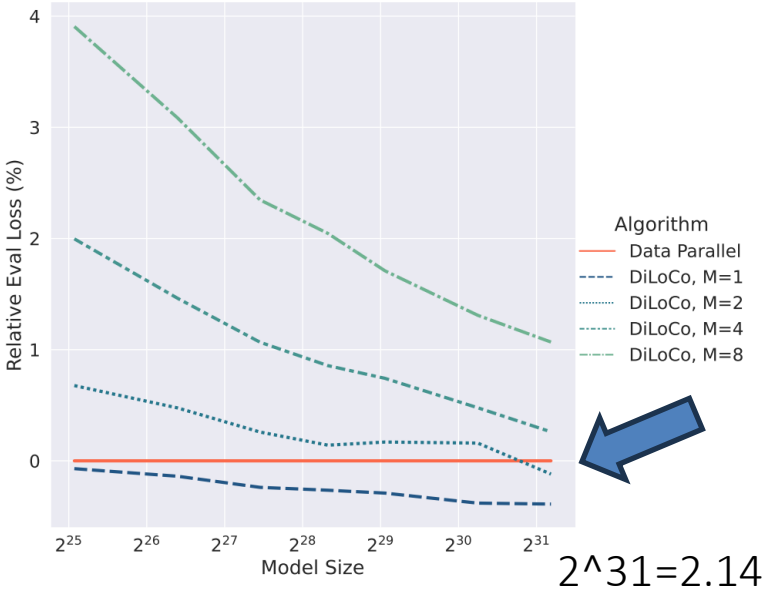


Arthur Douillard, Qixuan Feng, Andrei A Rusu, Rachita Chhaparia, Yani Donchev, Adhiguna Kuncoro, Marc'Aurelio Ranzato, Arthur Szlam, and Jiajun Shen. DiLoCo: Distributed lowcommunication training of language models. arXiv preprint arXiv:2311.08105, 2023

Zachary Charles, Gabriel Teston, Lucio Dery, Keith Rush, Nova Fallen, Zachary Garrett, Arthur Szlam, Arthur Douillard. Communication-Efficient Language Model Training Scales Reliably and Robustly: Scaling Laws for DiLoCo (Google), 2025

Finding 1: Scale

- DiLoCo's evaluation loss improves relative to Data-Parallel as N increases
- Scaling laws predict DiLoCo with M = 2 achieves lower loss than Data-Parallel above several billion parameters,
  - a phenomenon validated by both the largest model over which we performed tuning and our 4B and 10B model training runs



2^31=2.14 billion

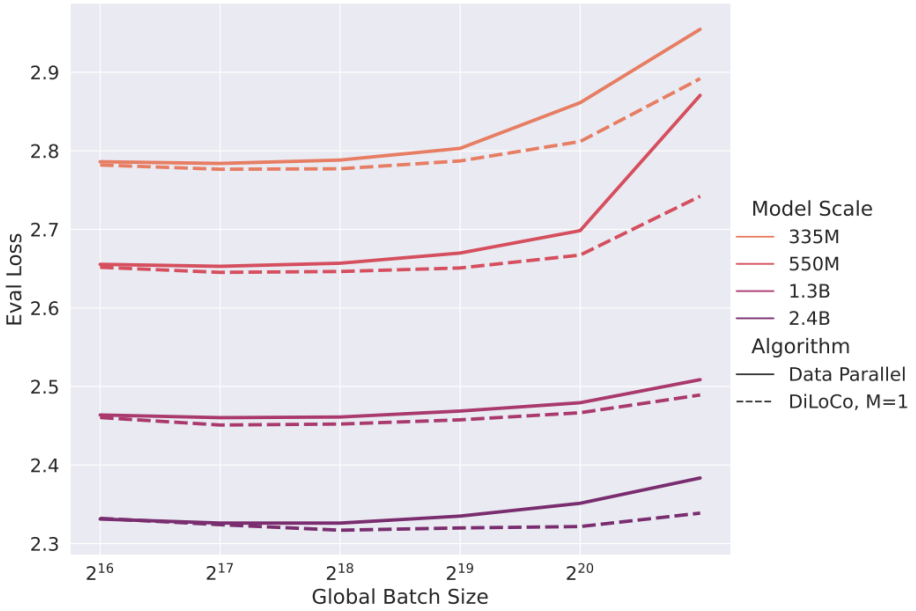
(b) Percentage difference in evaluation loss, relative to Data-Parallel.

N	DP	DiLoCo	
		$M = 1$	$M = 2$
35M	3.485	**3.482 (−0.1%)**	3.508 (+0.7%)
90M	3.167	**3.162 (−0.1%)**	3.182 (+0.5%)
180M	2.950	**2.943 (−0.2%)**	2.957 (+0.3%)
335M	2.784	**2.777 (−0.3%)**	2.788 (+0.1%)
550M	2.653	**2.645 (−0.3%)**	2.657 (+0.2%)
1.3B	2.460	**2.451 (−0.4%)**	2.464 (+0.2%)
2.4B	2.326	**2.317 (−0.4%)**	**2.323 (−0.1%)**

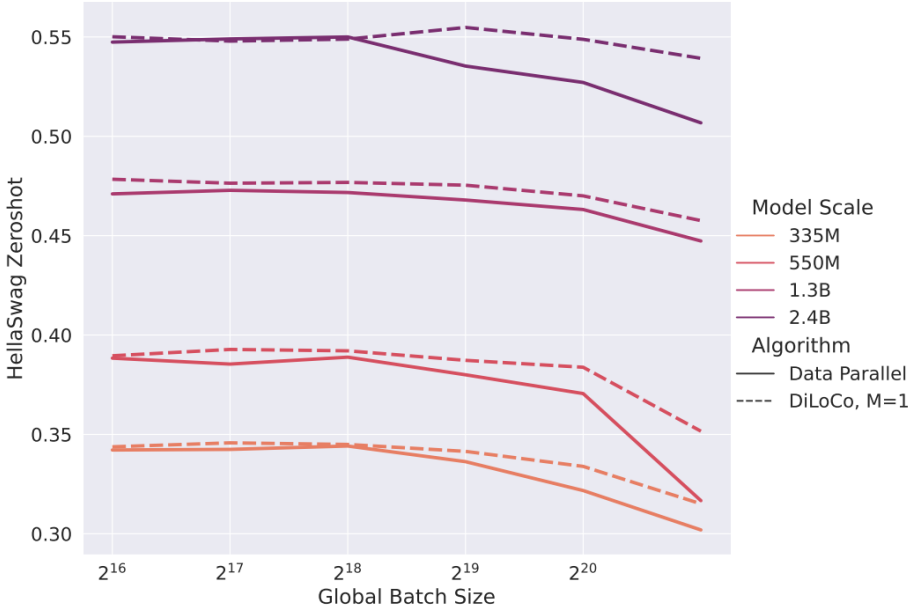
Algorithm	Loss	
	4B	10B
Data-Parallel	2.224	2.090
DiLoCo, $M = 1$	**2.219 (-0.22%)**	**2.086 (-0.19%)**
DiLoCo, $M = 2$	**2.220 (-0.18%)**	**2.086 (-0.19%)**
DiLoCo, $M = 4$	2.230 (+0.18%)	2.096 (+0.29%)

Finding 2: Single Replica DiLoCo

- DiLoCo with M = 1 attains lower evaluation loss than Data-Parallel across model scales
  - Generalizes better than Data-Parallel



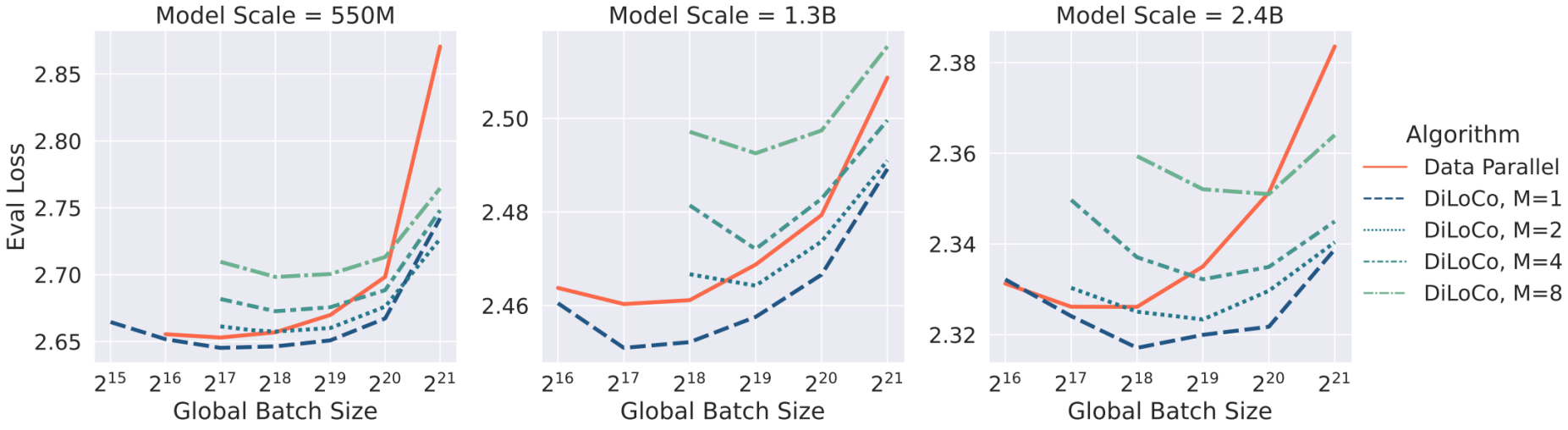
(a) Evaluation loss.



(b) Zero-shot accuracy on HellaSwag.

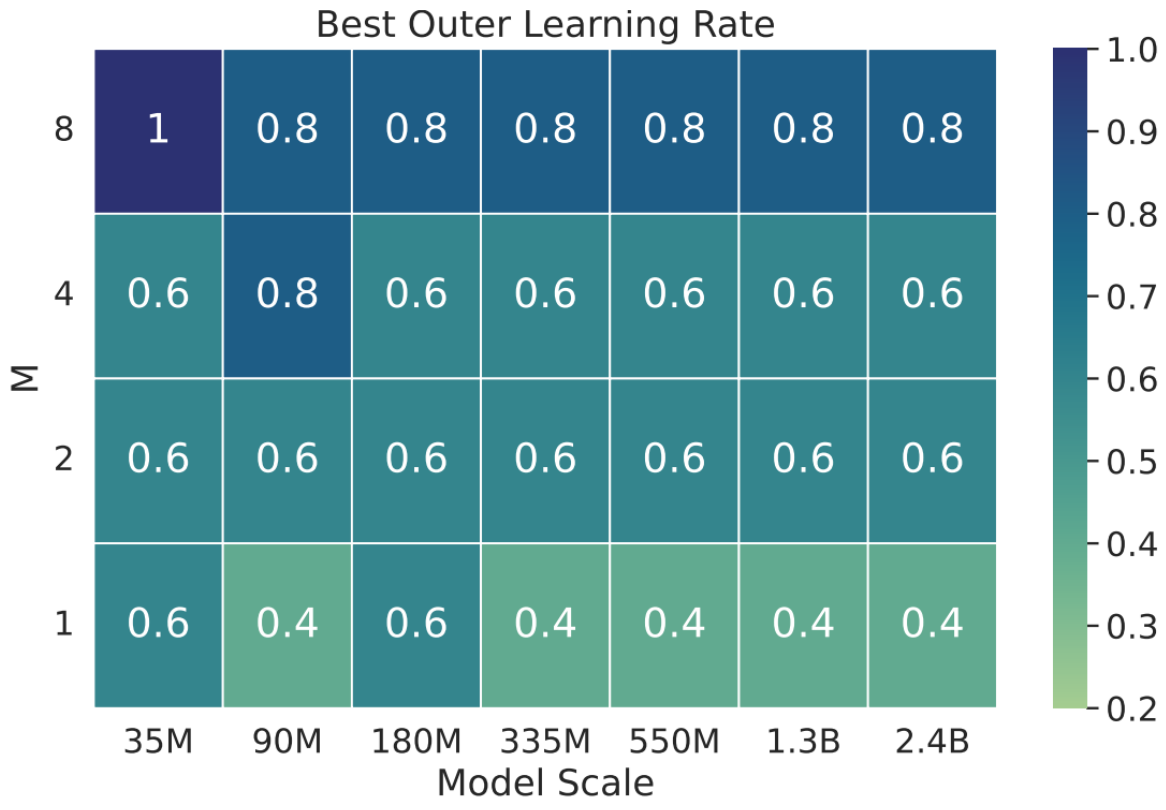
Finding 3: Optimal batch size

- DiLoCo increases the optimal batch size and moreover, the optimal global batch size increases with M
  - This means that DiLoCo improves horizontal scalability relative to Data-Parallel



Finding 4: Outer learning rate

- The optimal outer learning rate is essentially constant with respect to the model size N, but varies with M



Joint scaling laws

- They use a two-variable power law $f(N, M) = AN^{\alpha}M^{\beta}$
- They do this for loss L, inner learning rate γ and global batch size B.

	A	α	β
L	19.226	-0.0985	0.0116
γ	22256	-0.8827	0.2929
B	0.00709	0.4695	0.3399

Extrapolating to larger models

- Generally find that the loss values are predicted very well

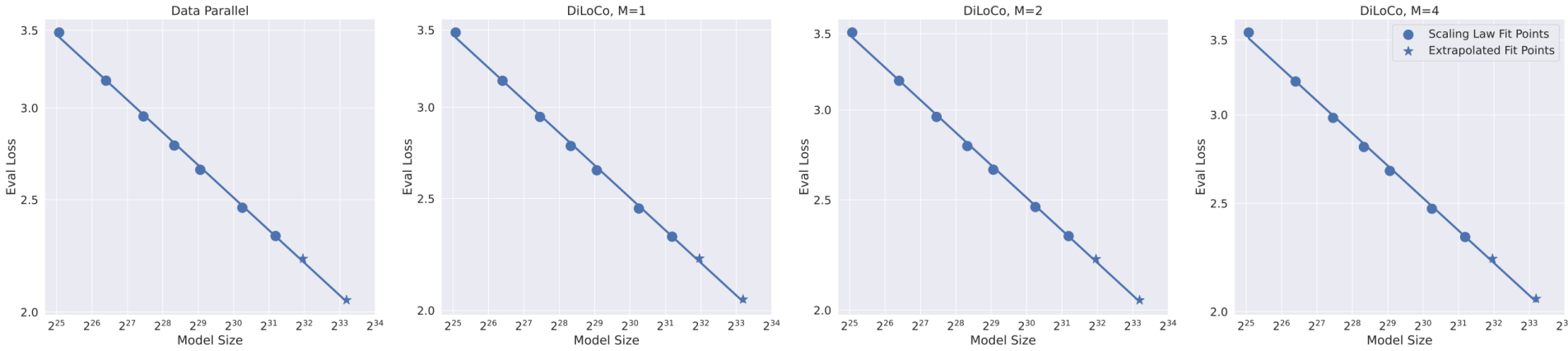


Figure 13: **DiLoCo scaling laws extrapolate well to larger models.** We present loss scaling laws for Data-Parallel and DiLoCo. Pictured are both the loss values to form the scaling law (by training models up to a scale of 2.4B) and loss values attained on larger models (4B and 10B). While we present the individual-fit scaling laws for simplicity, the joint fit also predicts loss well.

A Recent New Paper

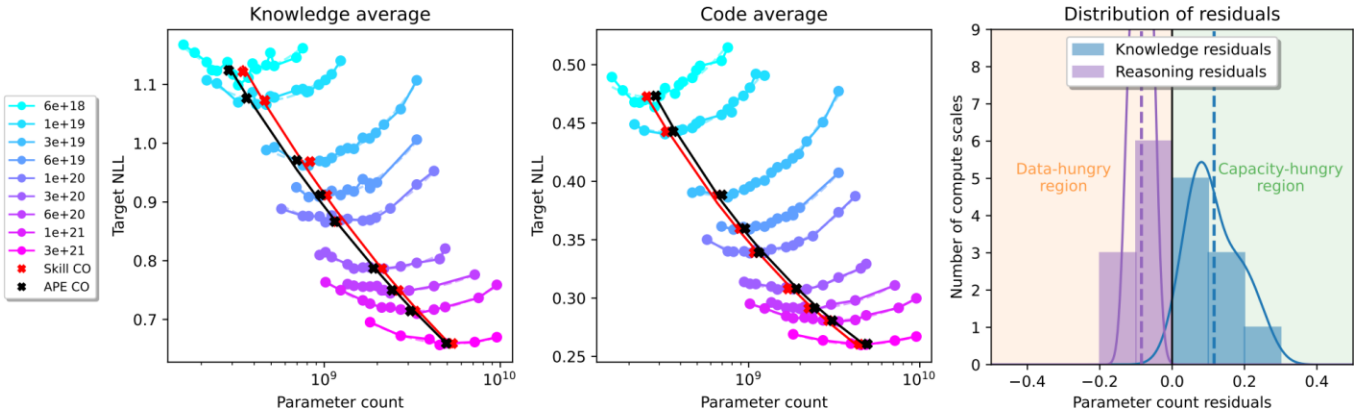
Compute Optimal Scaling of Skills: Knowledge vs Reasoning

Nicholas Roberts[μ†] **Niladri Chatterji**[σ] **Sharan Narang**[σ] **Mike Lewis**[σ] **Dieuwke Hupkes**[σ]

[μ]University of Wisconsin [σ]GenAI at Meta

[†]Work done during an internship at Meta.

Correspondence: nick11roberts@cs.wisc.edu dieuwkehupkes@meta.com



(a) Knowledge QA. (b) Code. (c) Data- vs capacity-hunger.

Figure 1: **Isoflop curves and COs for code and knowledge QA, along with the APE COs.** In (a), we see that on average across held-out datasets, knowledge QA tends to be capacity-hungry compared to the APE CO. On the other hand, in (b), we see that code tends to be data-hungry relative to the APE CO. We show the distribution of these relationships in (c), where we plot distributions of the log-scale differences in parameter count between skill-dependent COs and APE COs and find their means lie on opposite sides of the APE CO. The black curves in (a) and (b) represent the predicted APE COs from Dubey et al. (2024), mapped to their respective IsoFLOP groups.

Outline

- The Scaling Law
  - Scaling Laws
    - Kaplan et al., 2020 (OpenAI)
    - Hoffman et al., 2022 (DeepMind)
    - DiLoCo (Google Research and DeepMind)
  - Emergent Abilities
    - Wei et al., 2022

Emergent Abilities

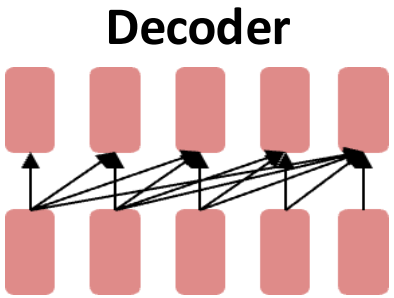
Emergent Abilities of Large Language Models, Wei et al, 2022

Main RQ: A survey and discussion of emergent abilities and their causes

"Emergent abilities of large language models are abilities that are not present in smaller-scale models but are present in large scale models"

Emergent abilities of large language models: GPT (2018)

- Let's revisit the Generative Pretrained Transformer (GPT) models from OpenAI as an example:
  - **GPT** (117M parameters; Radford et al., 2018)
    - Transformer decoder with 12 layers.
    - Trained on BooksCorpus: over 7000 unique books (4.6GB text).



  - Showed that language modeling at scale can be an effective pretraining technique for downstream tasks like natural language inference.
    - entailment
    - [START] *The man is in the doorway* [DELIM] *The person is near the door* [EXTRACT]

Emergent abilities of large language models: GPT-2 (2019)

- Let's revisit the Generative Pretrained Transformer (GPT) models from OpenAI as an example:
  - **GPT-2** (1.5B parameters; Radford et al., 2019)
    - Same architecture as GPT, just bigger (117M -> 1.5B)
    - But trained on **much more data**: 4GB -> 40GB of internet text data (WebText)
      - Scrape links posted on Reddit w/ at least 3 upvotes (rough proxy of human quality)

Language Models are Unsupervised Multitask Learners

Alec Radford [*1] Jeffrey Wu [*1] Rewon Child [1] David Luan [1] Dario Amodei [**1] Ilya Sutskever [**1]

Emergent zero-shot learning

- One key emergent ability in GPT-2 is **zero-shot learning**: the ability to do many tasks with **no examples,** and **no gradient updates,** by simply:
- Specifying the right sequence prediction problem (e.g. question answering):
- `Passage: Tom Brady... Q: Where was Tom Brady born? A: ...`
- Comparing probabilities of sequences (e.g. Winograd Schema Challenge [Levesque, 2011]):

`The cat couldn't fit into the hat because it was too big.`

Does `it = the cat` or `the hat`?

≡ Is `P(...because **the cat** was too big) >=`

`P(...because **the hat** was too big)`?

[Radford et al., 2019]

Emergent zero-shot learning

- GPT-2 beats SoTA on language modeling benchmarks with no task-specific fine-tuning

Context: "Why?" "I would have thought you'd find him rather dry," she said. "I don't know about that," said Gabriel. "He was a great craftsman," said Heather. "That he was," said Flannery.
Target sentence: "And Polish, to boot," said _____.
Target word: Gabriel

	LAMBADA (PPL)	LAMBADA (ACC)	CBT-CN (ACC)	CBT-NE (ACC)	WikiText2 (PPL)
SOTA	99.8	59.23	85.7	82.3	39.14
117M	**35.13**	45.99	**87.65**	**83.4**	**29.41**
345M	**15.60**	55.48	**92.35**	**87.1**	**22.76**
762M	**10.87**	**60.12**	**93.45**	**88.0**	**19.93**
1542M	**8.63**	**63.24**	**93.30**	**89.05**	**18.34**

LAMBADA (language modeling w/ long discourse dependencies) [Paperno et al., 2016]

Emergent zero-shot learning

- You can get interesting zero-shot behavior if you're creative enough with how you specify your task!
- Summarization on CNN/DailyMail dataset [See et al., 2017]:

```
SAN FRANCISCO,
California (CNN) --
A magnitude 4.2
earthquake shook
the San Francisco
...
overturn unstable
objects. TL;DR:
```

"Too Long, Didn't Read"

"Prompting"?

2018 SoTA

Supervised (287K)

Select from article

	ROUGE		
	R-1	R-2	R-L
Bottom-Up Sum	**41.22**	**18.68**	**38.34**
Lede-3	40.38	17.66	36.62
Seq2Seq + Attn	31.33	11.81	28.83
GPT-2 TL;DR:	29.34	8.27	26.58
Random-3	28.78	8.63	25.52

[Radford et al., 2019]

Emergent abilities of large language models: GPT-3 (2020)

- **GPT-3** (175B parameters; Brown et al., 2020)
- Another increase in size (1.5B -> **175B**)
- and data (40GB -> **over 600GB**)

Language Models are Few-Shot Learners

Tom B. Brown* Benjamin Mann* Nick Ryder* Melanie Subbiah*

Jared Kaplan† Prafulla Dhariwal Arvind Neelakantan Pranav Shyam Girish Sastry

Amanda Askell Sandhini Agarwal Ariel Herbert-Voss Gretchen Krueger Tom Henighan

Rewon Child Aditya Ramesh Daniel M. Ziegler Jeffrey Wu Clemens Winter

Christopher Hesse Mark Chen Eric Sigler Mateusz Litwin Scott Gray

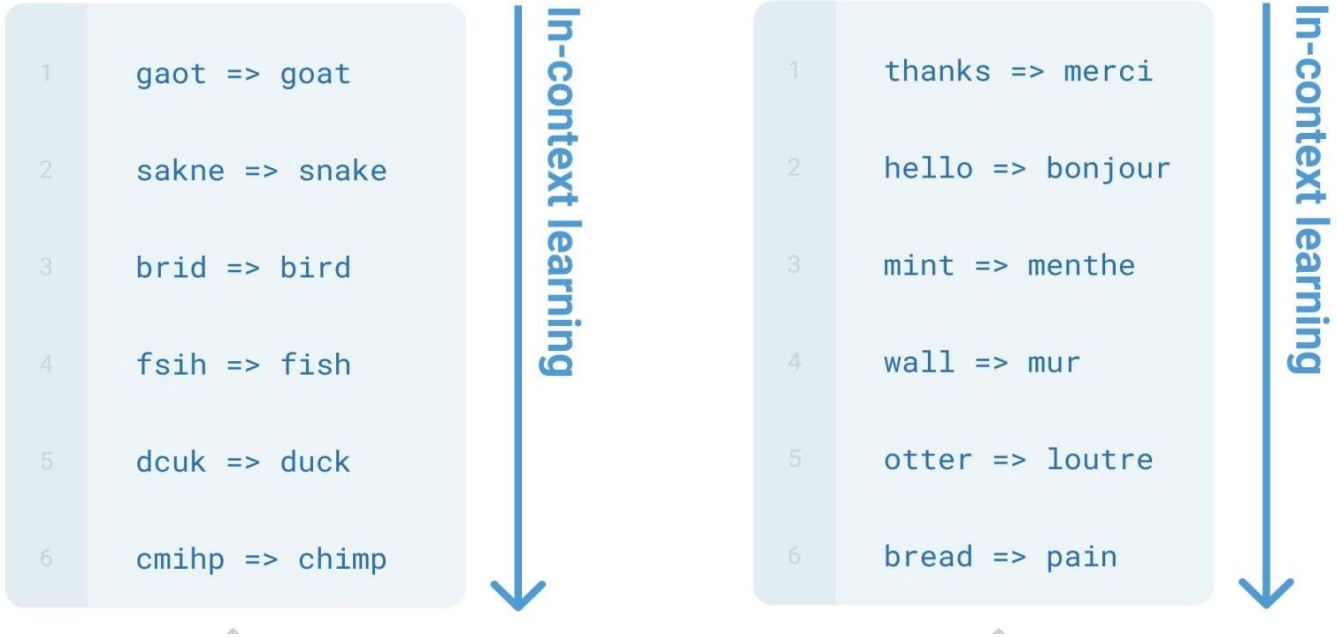
Benjamin Chess Jack Clark Christopher Berner

Sam McCandlish Alec Radford Ilya Sutskever Dario Amodei

OpenAI

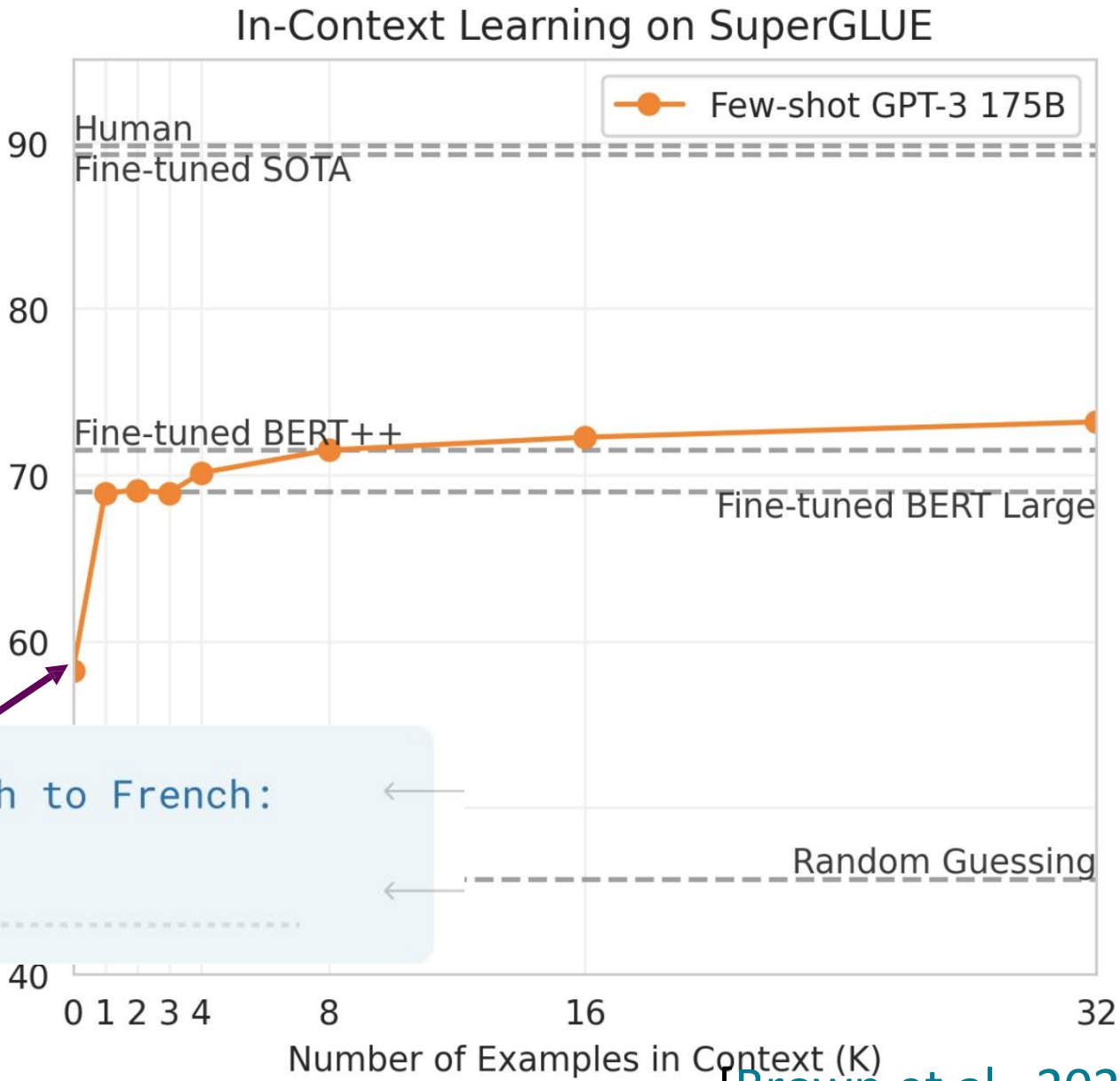
Emergent few-shot learning

- Specify a task by simply **prepending examples of the task before your example**
- Also called **in-context learning,** to stress that *no gradient updates* are performed when learning a new task (there is a separate literature on few-shot learning with gradient updates)



[Brown et al., 2020]

Emergent few-shot learning

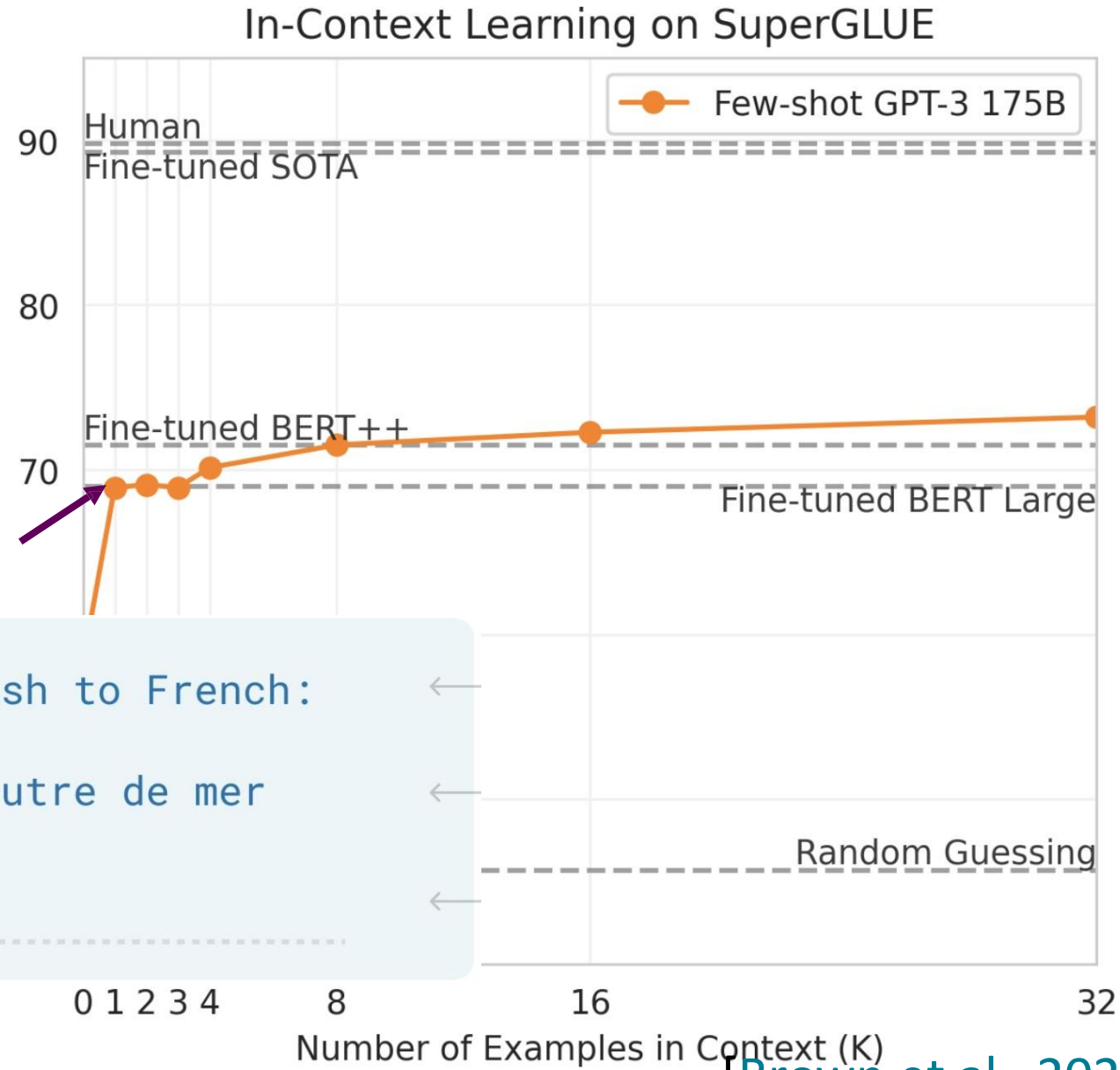


Zero-shot

```
1  Translate English to French:
2  cheese =>
```

[Brown et al., 2020]

Emergent few-shot learning

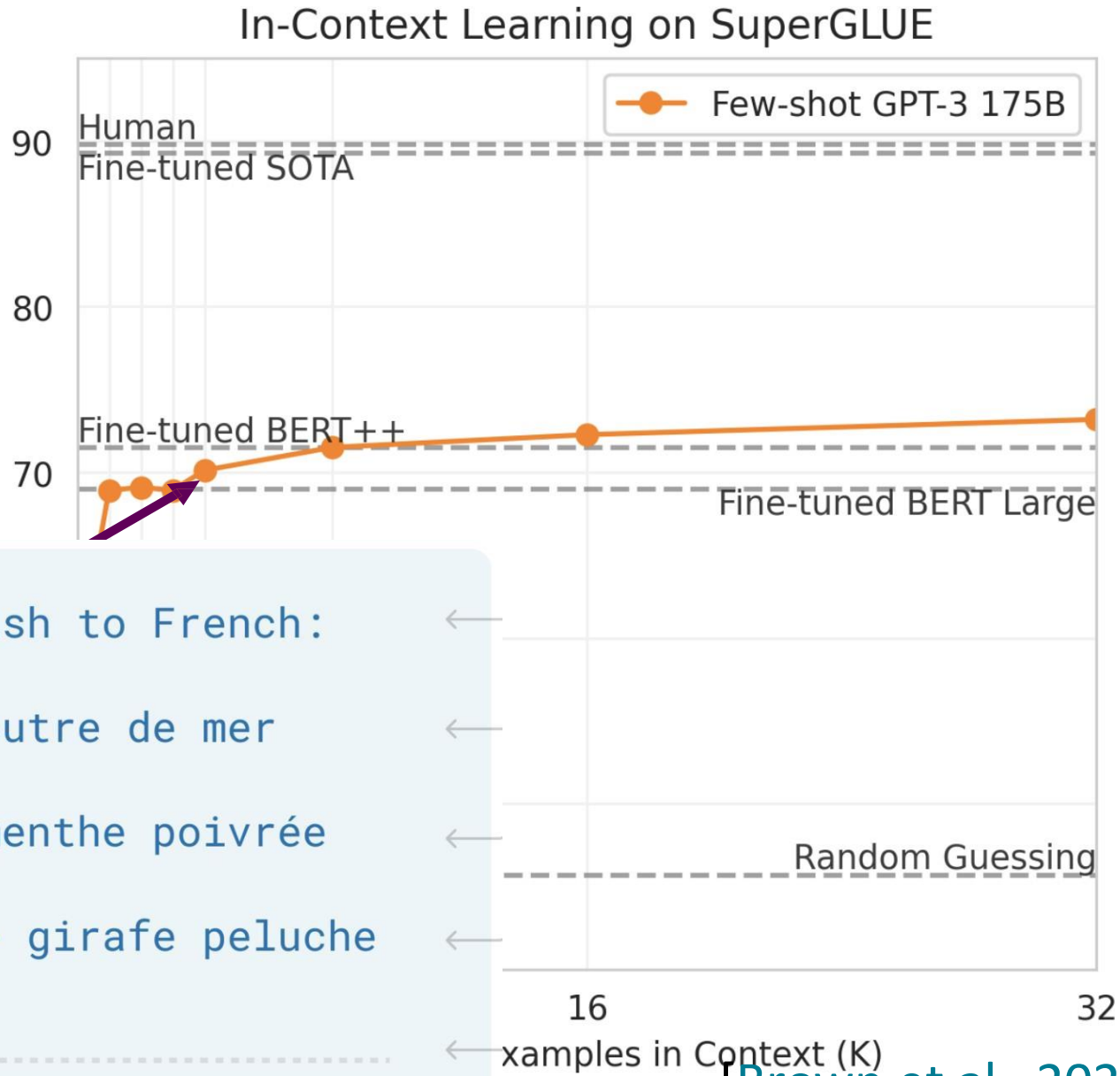


One-shot

```
1  Translate English to French:
2  sea otter => loutre de mer
3  cheese =>
```

[Brown et al., 2020]

Emergent few-shot learning



Few-shot

```
1  Translate English to French:
2  sea otter => loutre de mer
3  peppermint => menthe poivrée
4  plush girafe => girafe peluche
5  cheese =>
```

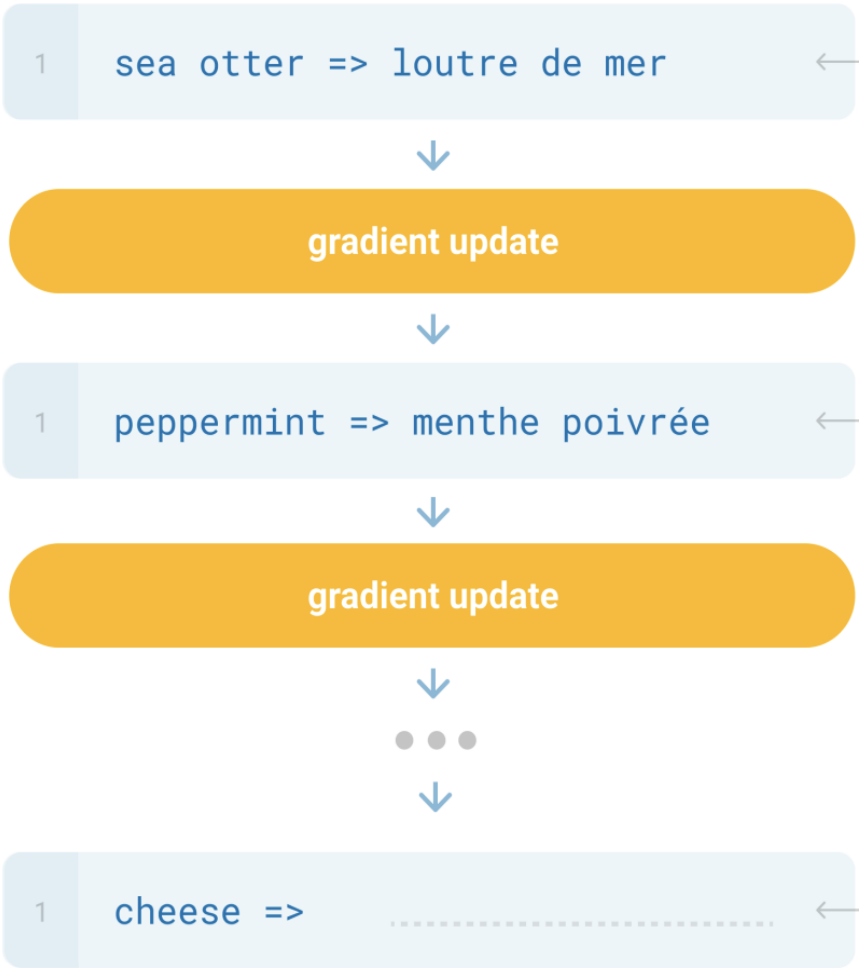
[Brown et al., 2020]

New methods of “prompting” LMs

Zero/few-shot prompting

```
Translate English to French:
sea otter => loutre de mer
peppermint => menthe poivrée
plush girafe => girafe peluche
cheese =>
```

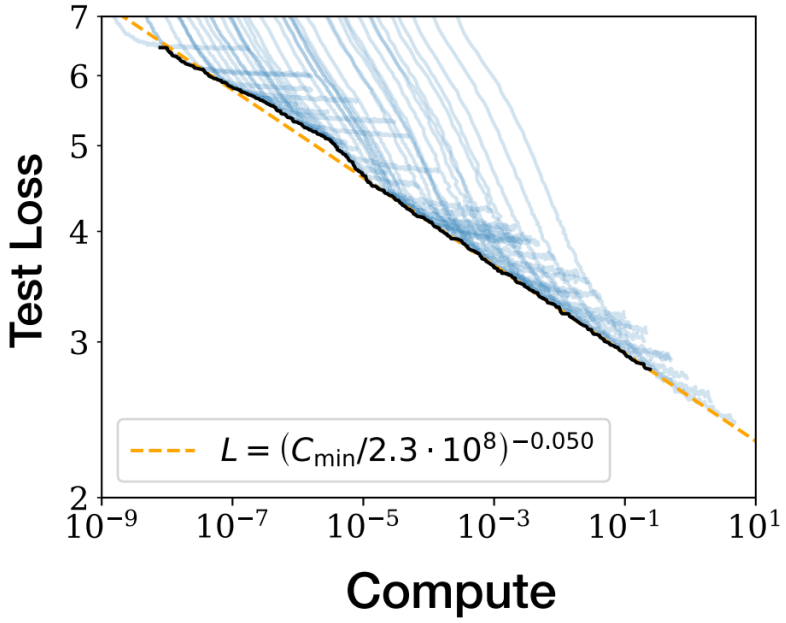
Traditional fine-tuning

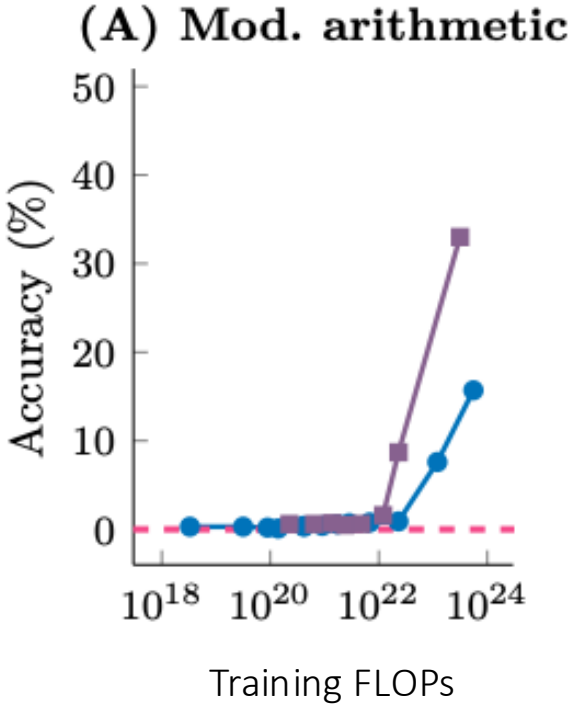


[Brown et al., 2020]

Emergent Abilities of Large Language Models, Wei et al, 2022

- Scaling Laws: A predictable and smooth relationship between scale (FLOPs or params.) and cross-entropy loss (left)
- Emergent abilities: Abilities/performance metrics that are a discontinuous function of scale (right)

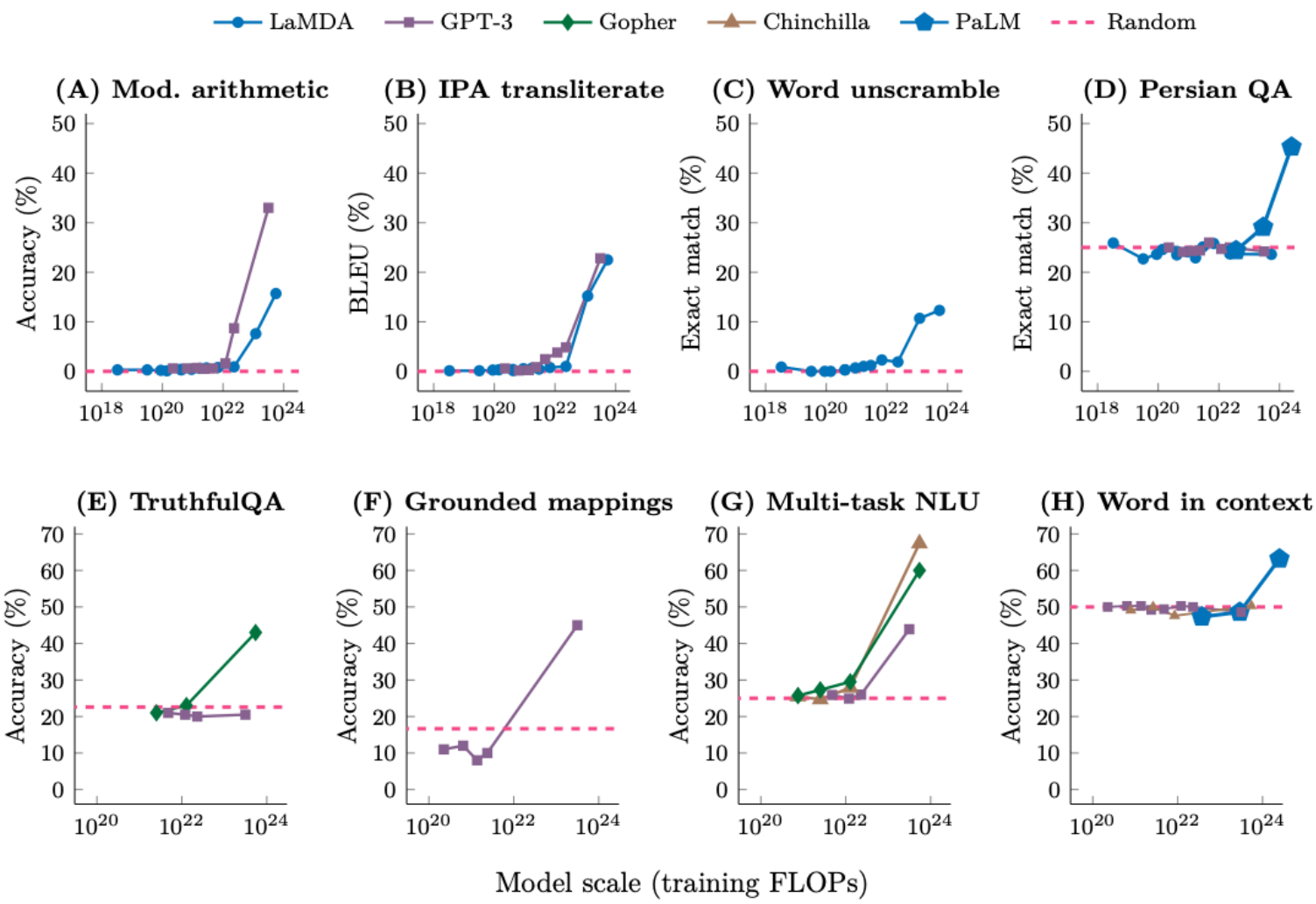




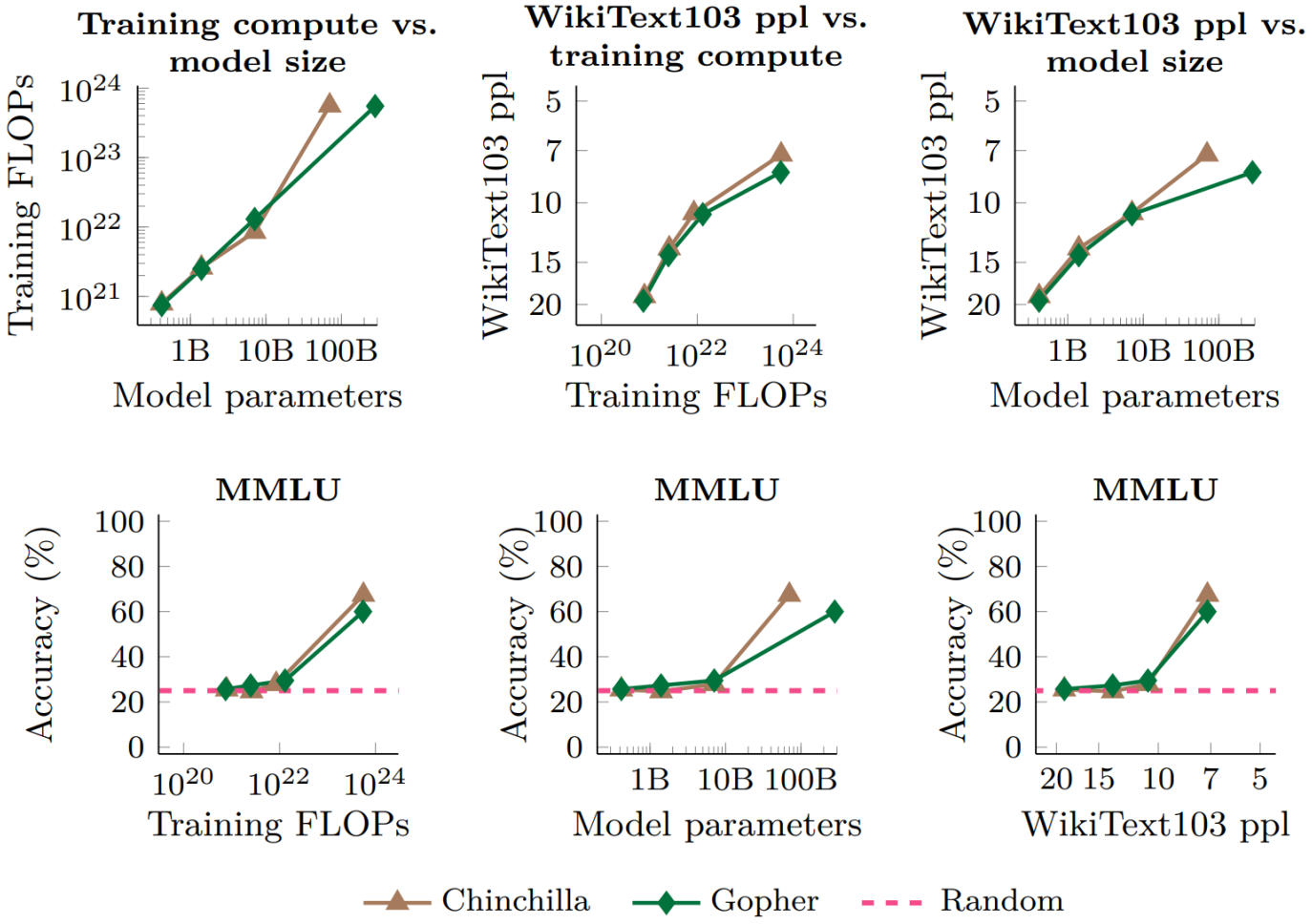
Emergent Abilities of Large Language Models, Wei et al, 2022

- Examples of emergent abilities

Tasks from BIG-Bench



Emergent Abilities of Large Language Models, Wei et al, 2022



Top row: the relationships between training FLOPs, model parameters, and perplexity (ppl) on WikiText103 (Merity et al., 2016) for Chinchilla and Gopher. Bottom row: Overall performance on the massively multi-task language understanding benchmark (MMLU; Hendrycks et al., 2021a) as a function of training FLOPs, model parameters, and WikiText103 perplexity.

Emergent Abilities of Large Language Models, Wei et al, 2022

- In conclusion:
  - Overall, emergent abilities should probably be viewed as a function of many correlated variables.
  - This analysis does not explain why downstream metrics are emergent or enable us to predict the scale at which emergence occurs.

Summary

- The Scaling Law
  - Scaling Laws
    - Kaplan et al., 2020 (OpenAI)
    - Hoffman et al., 2022 (DeepMind)
    - DiLoCo (Google Research and DeepMind)
  - Emergent Abilities
    - Wei et al., 2022