Training Compute-Optimal Large Language Models

Hoffman, Borgeaud, Mensch, Sifre, et al. | DeepMind | March 2022

REFORM Reading Group | October 23, 2024

High-level summary

There are a lot of design choices when training large language models (LLMs)

Because LLMs are by nature “large,” it’s impossible to explore the design space

This work studies how, for a fixed compute budget, how to vary the **model size** and the **number of training tokens**

They take three approaches to this question. The main conclusions are:

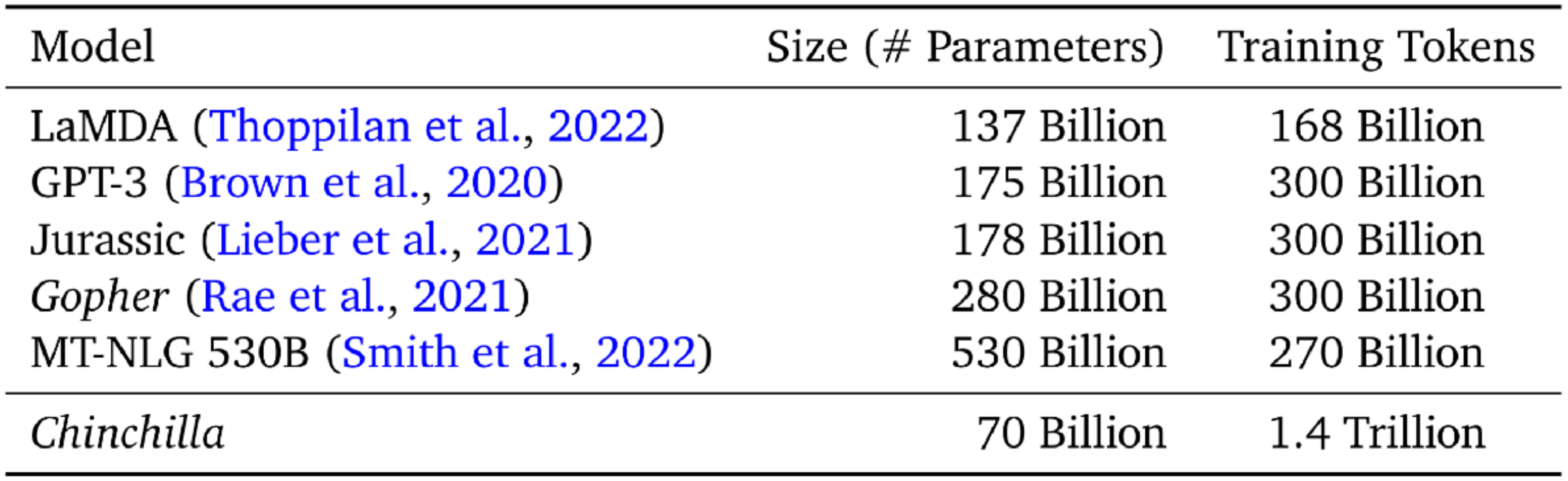
1. Many models are oversized
2. Models can be smaller and instead trained on more data
3. For a fixed compute budget, model size and # training tokens should scale equally

They demonstrate the importance of these insights by training a new model *Chinchilla* that outperforms previous (larger) models across various benchmarks

Large language models (in 2022)

Largest LLMs models were 500 billion parameters (now, ~1 trillion)

Typically trained on ~300 billion tokens



Model	Size (# Parameters)	Training Tokens
LaMDA (Thoppilan et al., 2022)	137 Billion	168 Billion
GPT-3 (Brown et al., 2020)	175 Billion	300 Billion
Jurassic (Lieber et al., 2021)	178 Billion	300 Billion
Gopher (Rae et al., 2021)	280 Billion	300 Billion
MT-NLG 530B (Smith et al., 2022)	530 Billion	270 Billion
Chinchilla	70 Billion	1.4 Trillion

Large language models (in 2022)

Largest LLMs models were 500 billion parameters (now, ~1 trillion)

- Typically trained on ~300 billion tokens

Compute cost for training is LARGE and prohibits training lots of models (i.e., prevents exploration of the full design space). Typically, can only train 1 LLM.

Kaplan et al. (2020): Power law relationship between model size and performance

- This caused many to continually increase model size
- This paper finds that one can achieve same (or better) performance with smaller models and more training data!

Problem statement

N = model size

D = number of training tokens

C = compute budget

$L(N, D)$ is pre-training loss given N and D

$\approx 6ND$

Question: Give a compute (FLOPs) budget C, what are the optimal values for the model size N and number of training tokens D?

High-level approach

Trained >400 models

Varied model size from 700M to 16B

Varied number of tokens from 5B to 400B

Important: tuned hyperparameters based on # data

Took 3 different approaches to estimating loss $L(N, D)$:

1. Fix model size N and vary number of training tokens D
2. For fixed FLOP counts, vary the model size N
3. Fit a parametric loss function

(All three yielded similar results)

Approach 1: Fix model size and vary # training tokens

```
For each $N$
    For each $D$ from 1x to 16x
        Train over 4 different training sequences
    This produces a curve of the training loss as a function of training tokens (for a given $N$)
    Smooth & interpolate training loss curve
```

(Next slide gives visual intuition)

Note: FLOPs can be computed from N and D, $\approx 6ND$

Training loss curves for every N (each in different colors) as # training tokens D increases (indicated by FLOPs)

Using left plot, for each FLOPs value, find the N (color) with the lowest loss

Since N implicitly defines a D for a given FLOPs size, we can generate the analogous plot for D

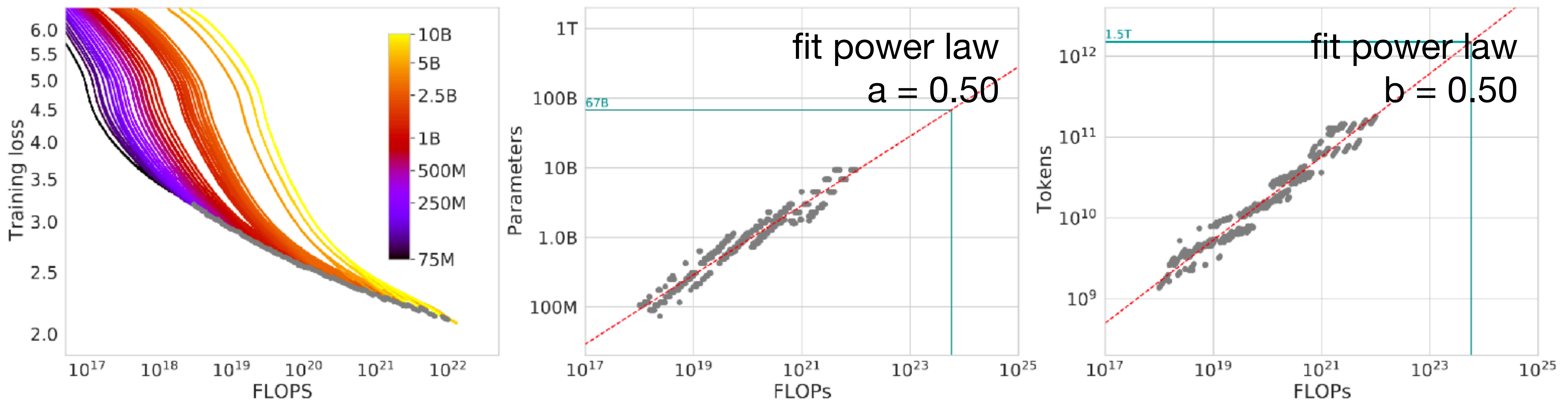


Figure 2 | **Training curve envelope.** On the **left** we show all of our different runs. We launched a range of model sizes going from 70M to 10B, each for four different cosine cycle lengths. From these curves, we extracted the envelope of minimal loss per FLOP, and we used these points to estimate 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}).

Approach 2: Vary model size for fixed FLOPs

For a given FLOP budget C (from 6e18 to 3e12),

- For different model sizes N
  - Train on the number of tokens that would give you C total compute
- This gives a curve of the loss as a function of N (under a given C)
- (See next slide)

final loss curves for different C
(each in different colors)
as model size N increases

For each FLOPs value, determine N with minimal loss

Since N implicitly defines a D for a given C, generate analogous plot for D

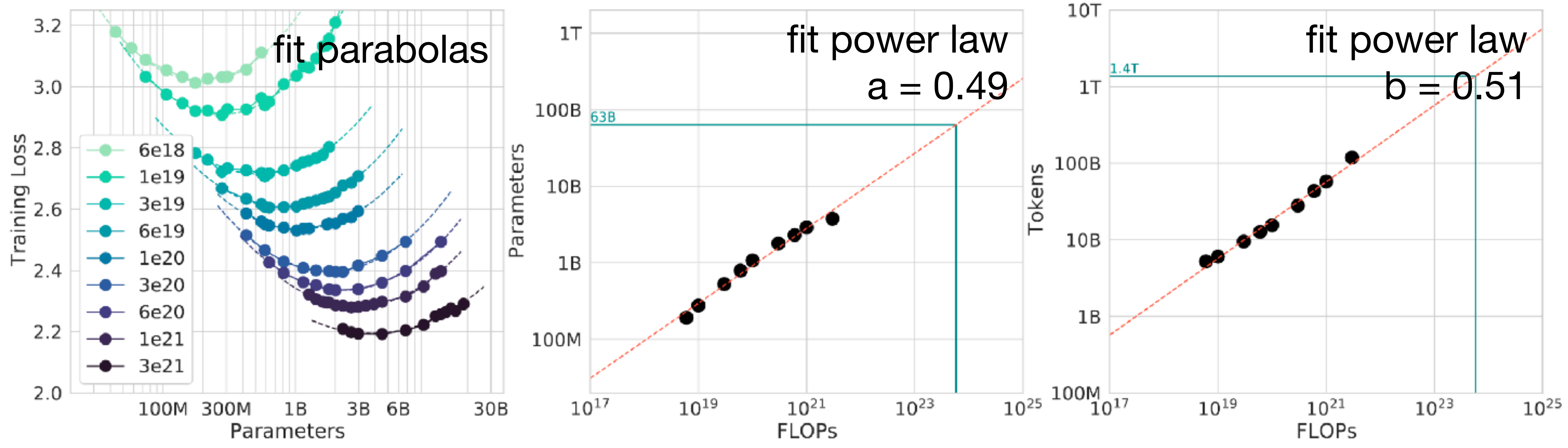


Figure 3 | **IsoFLOP curves.** For various model sizes, we choose the number of training tokens such that the final FLOPs is a constant. The cosine cycle length is set to match the target FLOP count. We find a clear valley in loss, meaning that for a given FLOP budget there is an optimal model to train (**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*.

Approach 3: Fit parametric loss function

Using experiments from Approaches 1 and 2 (no new experiments in Approach 3)…

Fit the following parametric function:

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

gap due to limited
amt of training data

entropy of natural text
(loss for ideal generative process)

gap due to limited
number of parameters

Approach 3: Fit parametric loss function

Using experiments from Approaches 1 and 2 (no new experiments in Approach 3)…

Fit the following parametric function:

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

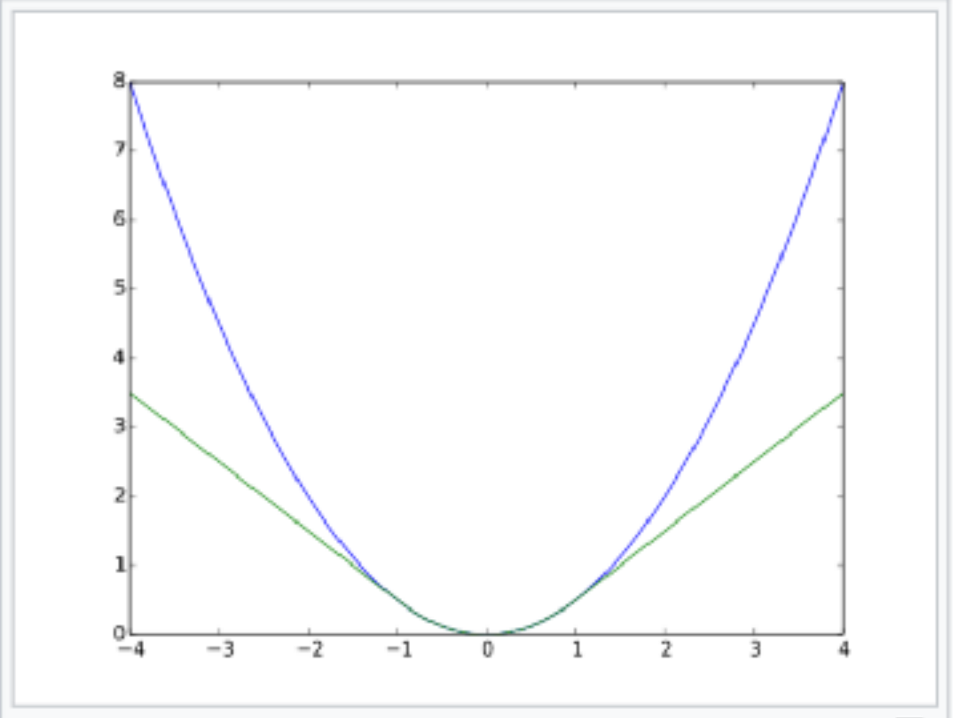
gap due to limited amt of training data

entropy of natural text
(loss for ideal generative process)

gap due to limited number of parameters

Minimize Huber loss:
$$L_{\delta}(a) = \begin{cases} \frac{1}{2}a^2 & \text{for } |a| \le \delta, \\ \delta \cdot \left(|a| - \frac{1}{2}\delta\right), & \text{otherwise.} \end{cases}$$

wikipedia.com


Huber loss (green, $\delta = 1$) and squared error loss (blue) as a function of $y - f(x)$

blue line gives optimal loss for given FLOPs

curves give fixed compute

a = 0.46
b = 0.54

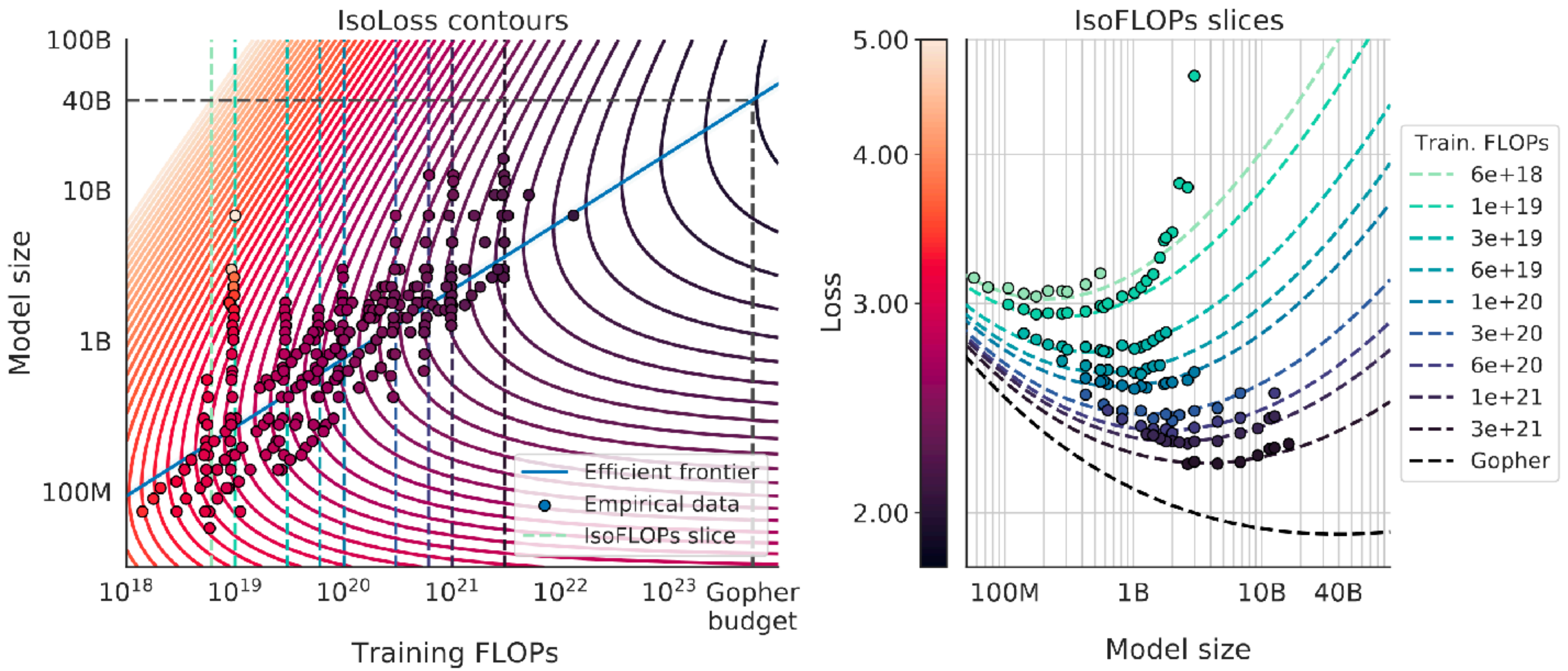
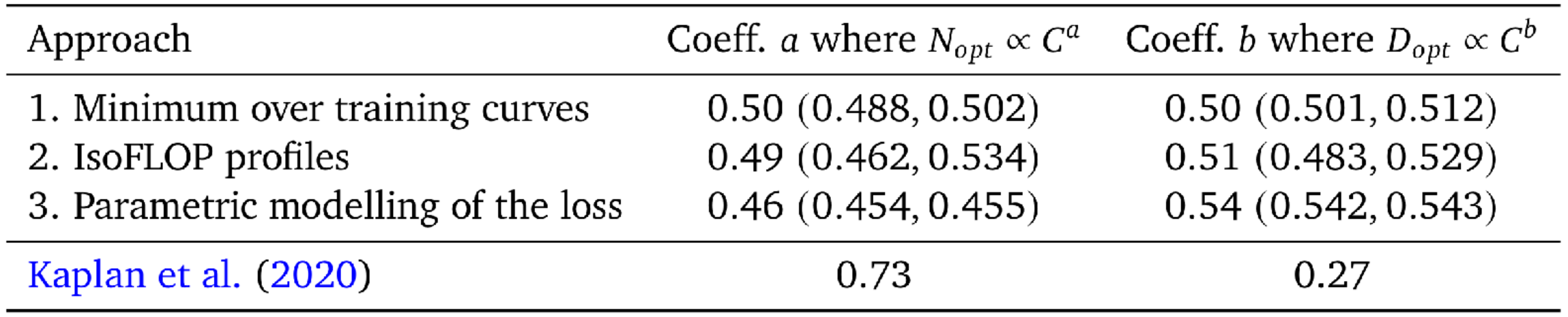


Figure 4 | **Parametric fit.** We fit a parametric modelling of the loss $\hat{L}(N, D)$ and display contour (**left**) and isoFLOP slices (**right**). For each isoFLOP slice, we include a corresponding dashed line in the left plot. In the left plot, we show the efficient frontier in blue, which is a line in log-log space. Specifically, the curve goes through each iso-loss contour at the point with the fewest FLOPs. We project the optimal model size given the *Gopher* FLOP budget to be 40B parameters.

Implications



Approach	Coeff. a where $N_{opt} \propto C^a$	Coeff. b where $D_{opt} \propto C^b$
1. Minimum over training curves	$0.50\ (0.488, 0.502)$	$0.50\ (0.501, 0.512)$
2. IsoFLOP profiles	$0.49\ (0.462, 0.534)$	$0.51\ (0.483, 0.529)$
3. Parametric modelling of the loss	$0.46\ (0.454, 0.455)$	$0.54\ (0.542, 0.543)$
Kaplan et al. (2020)	0.73	0.27

These results suggest that model size and # of training tokens should scale ~equally

Third approach suggests that # of training tokens is *more* important than model size

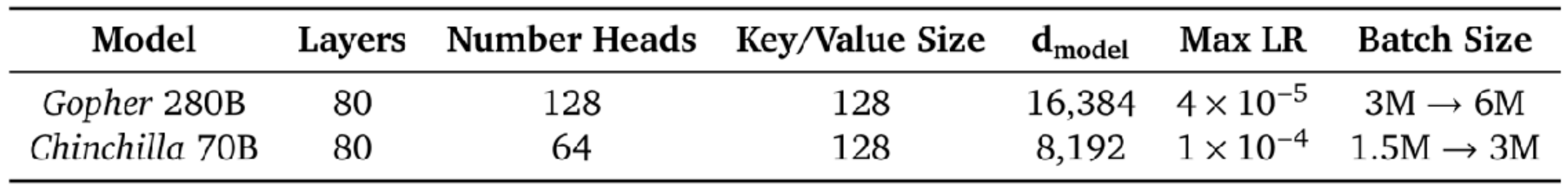
Suggests current (in 2022) models are oversized and that dataset collection is important

Chinchilla: Design

Used this analysis to construct a new model: *Chinchilla*

Same number of FLOPs as *Gopher* but smaller in model size

Details: (i) trained on *MassiveText*, (ii) uses AdamW rather than Adam, (iii) modified SentencePiece tokenizer w/ no NFKC normalization, (iv) bfloat16 for forward/backward passes and float32 for distributed optimizer state



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

Chinchilla: Motivation

Compared to Gopher, can reduce model size by 4x and increase training by 4x

Chinchilla trained on 1.4 trillion tokens

Why is this good? Smaller model size implies…

- Cheaper to run inference
- Cheaper to fine-tune
- Can be deployed on smaller hardware downstream

Chinchilla: Results

Evaluated on many tasks, including **Language Modeling, Reading Comprehension, QA, MMLU, BIG-bench**, **Common Sense** tasks

- Compared to Gopher, sometimes to GPT-3.5, humans, MT-NLG 530B
- TL;DR Chinchilla outperforms all, except
    - Human experts on MMLU
    - MT-NLG 530B on Common Sense

Considered **toxicity** and **bias**

- Chinchilla resolves pronouns better than Gopher, no difference in toxicity

Discussion: data-dependence

We show scaling results from an IsoFLOP (Approach 2) analysis after training on two different datasets: C4 (Raffel et al., 2020b) and GitHub code (we show results with data from Rae et al. (2021)), results are shown in Table A2. For both set of experiments using subsets of *MassiveText*, we use the same tokenizer as the *MassiveText* experiments.

We find that the scaling behaviour on these datasets is very similar to what we found on *MassiveText*, as shown in Figure A2 and Table A2. This suggests that our results are independent of the dataset as long as one does not train for more than one epoch.

Approach	Coef. a where $N_{opt} \propto C^a$	Coef. b where $D_{opt} \propto C^b$
C4	0.50	0.50
GitHub	0.53	0.47

(Sorscher et al., 2022)

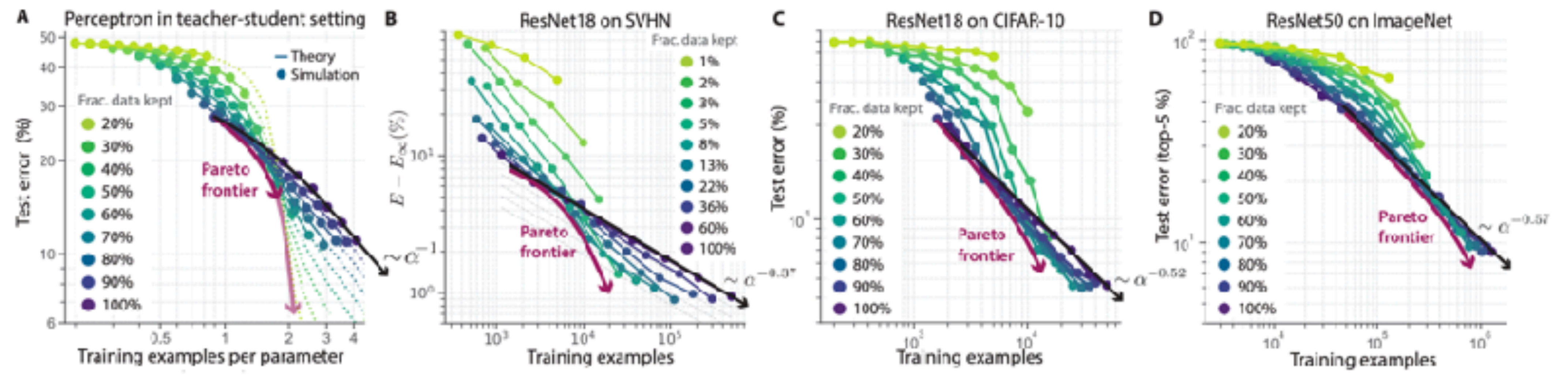
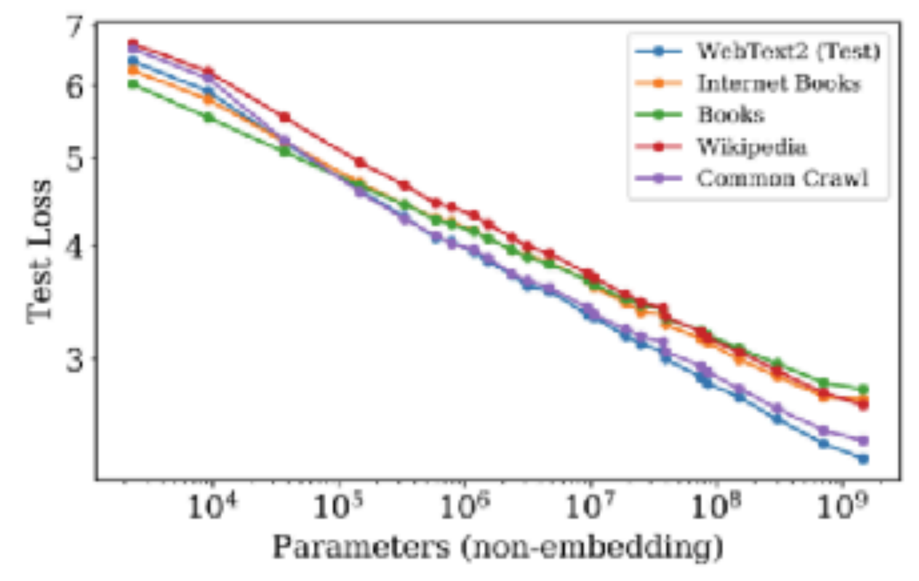
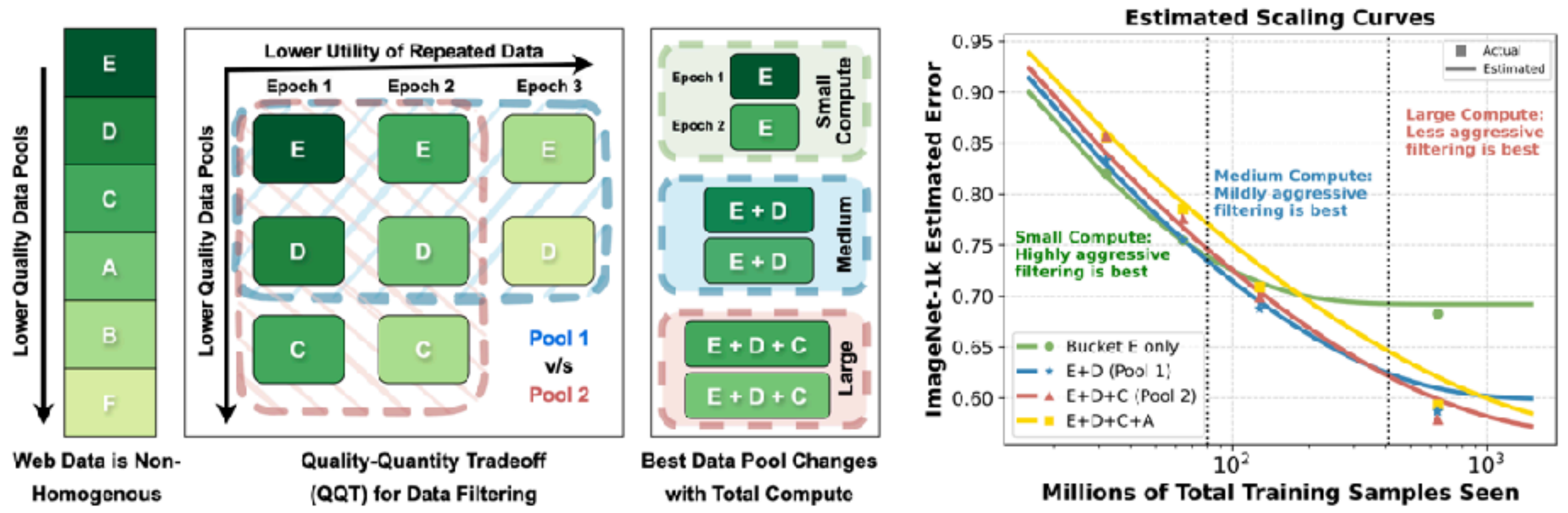


Figure 3: Beating power law scaling in practice. **A–D:** Curves of test error against pruned dataset size in 4 settings. Pruning scores were EL2N [10] for CIFAR-10 and SVHN and memorization [13] for ImageNet. See App. B for all pruning/training details and App. D for similar ImageNet plots with EL2N. Note solid curves reflect performance with a fixed total dataset size; if we prune more aggressively with even larger datasets, scaling could improve further (e.g., dashed lines in **A**). Error curves with no data pruning ($f = 1$) are labeled with their best-fit power law scaling $\sim \alpha^{-\nu}$. (Note that for SVHN in **B** an asymptotic constant error $E(P \to \infty) = 1.1\%$ is subtracted from each of the curves to visualize the power law scaling more clearly.)

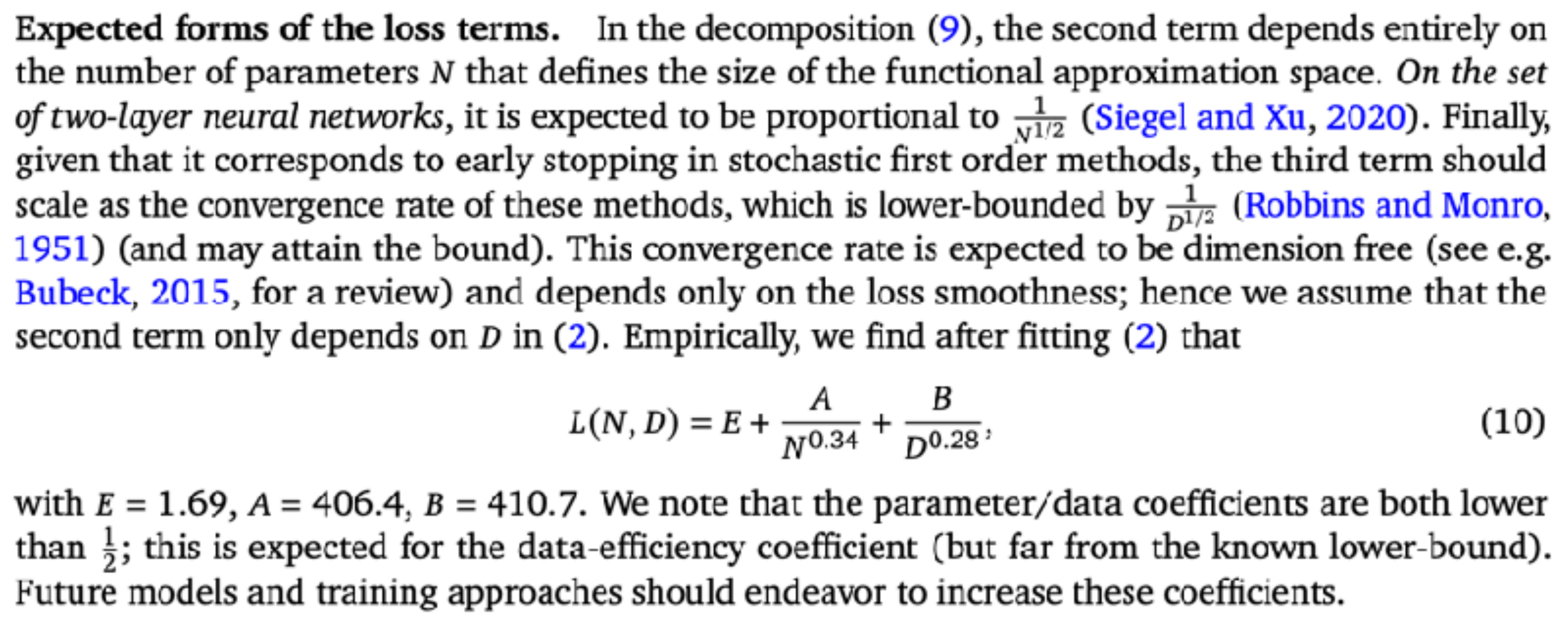


[Kaplan+ 2021]



(Goyal et al., 2024)

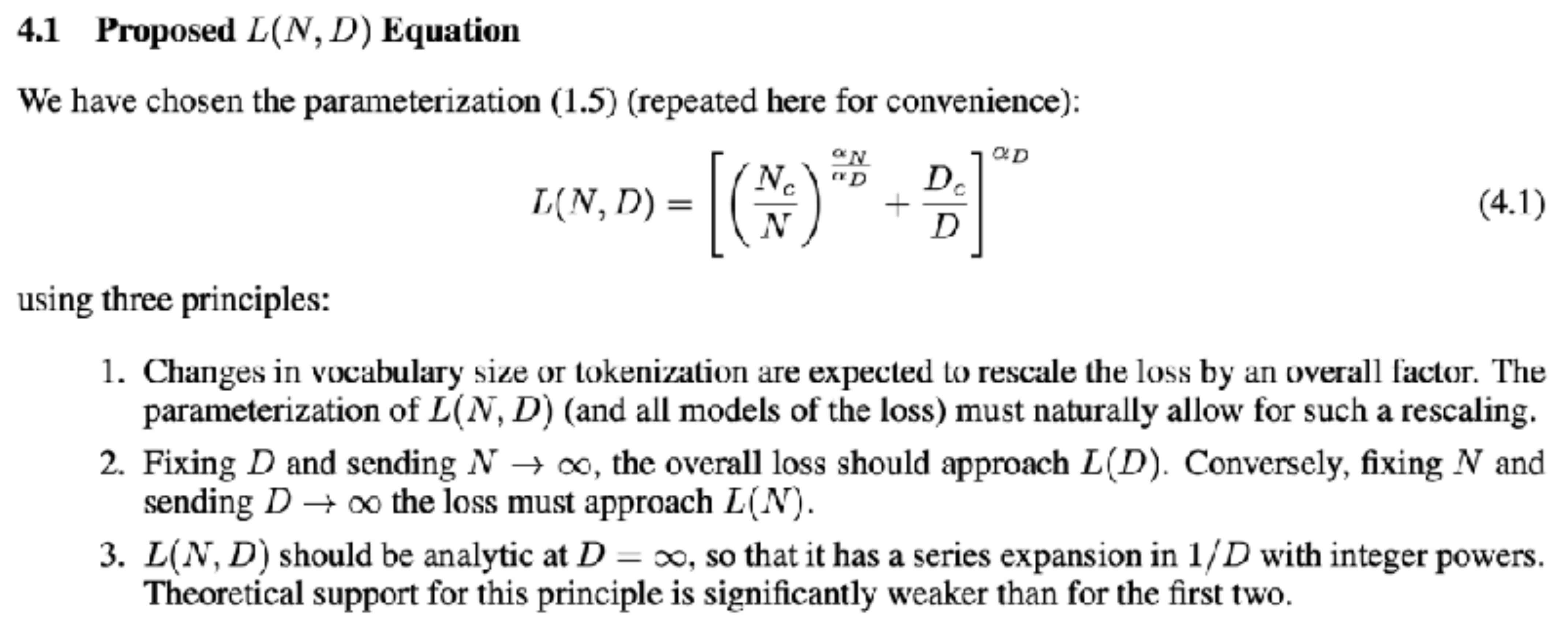
Discussion: Choice of power law



Expected forms of the loss terms. In the decomposition (9), the second term depends entirely on the number of parameters N that defines the size of the functional approximation space. *On the set of two-layer neural networks*, it is expected to be proportional to $\frac{1}{N^{1/2}}$ (Siegel and Xu, 2020). Finally, given that it corresponds to early stopping in stochastic first order methods, the third term should scale as the convergence rate of these methods, which is lower-bounded by $\frac{1}{D^{1/2}}$ (Robbins and Monro, 1951) (and may attain the bound). This convergence rate is expected to be dimension free (see e.g. Bubeck, 2015, for a review) and depends only on the loss smoothness; hence we assume that the second term only depends on D in (2). Empirically, we find after fitting (2) that

$$L(N, D) = E + \frac{A}{N^{0.34}} + \frac{B}{D^{0.28}}, \tag{10}$$

with $E = 1.69$, $A = 406.4$, $B = 410.7$. We note that the parameter/data coefficients are both lower than $\frac{1}{2}$; this is expected for the data-efficiency coefficient (but far from the known lower-bound). Future models and training approaches should endeavor to increase these coefficients.



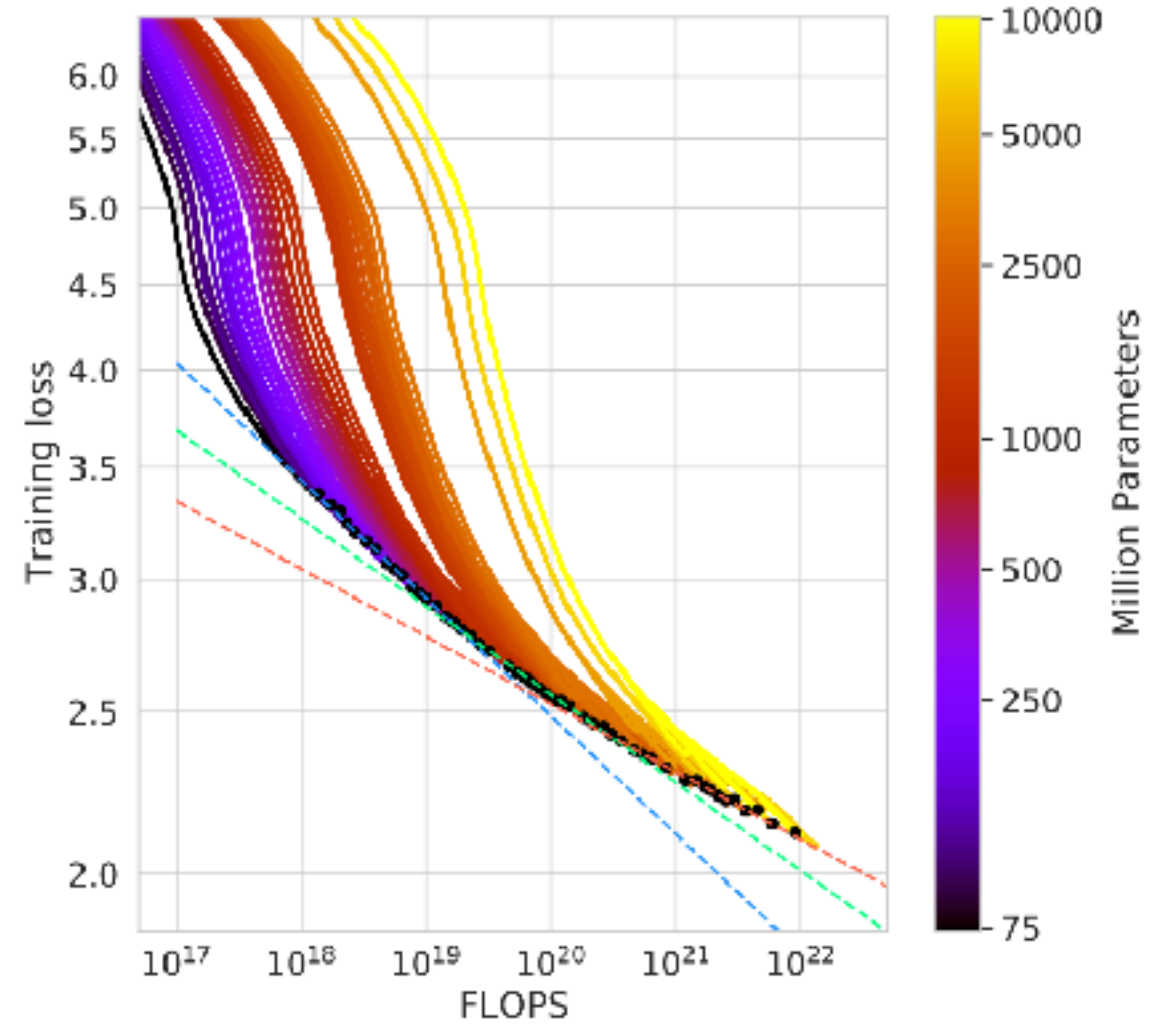
4.1 Proposed $L(N, D)$ Equation

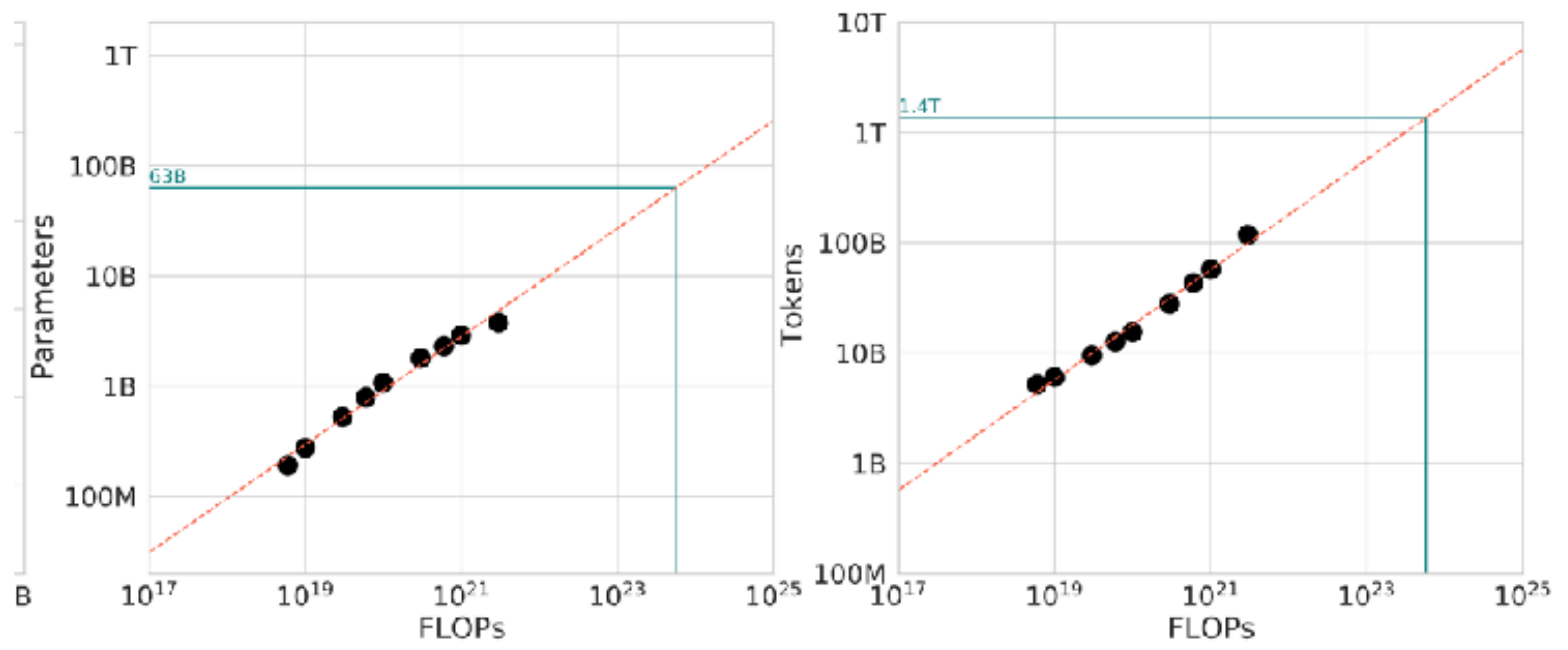
We have chosen the parameterization (1.5) (repeated here for convenience):

$$L(N, D) = \left[\left(\frac{N_c}{N}\right)^{\frac{\alpha_N}{\alpha_D}} + \frac{D_c}{D}\right]^{\alpha_D} \tag{4.1}$$

using three principles:

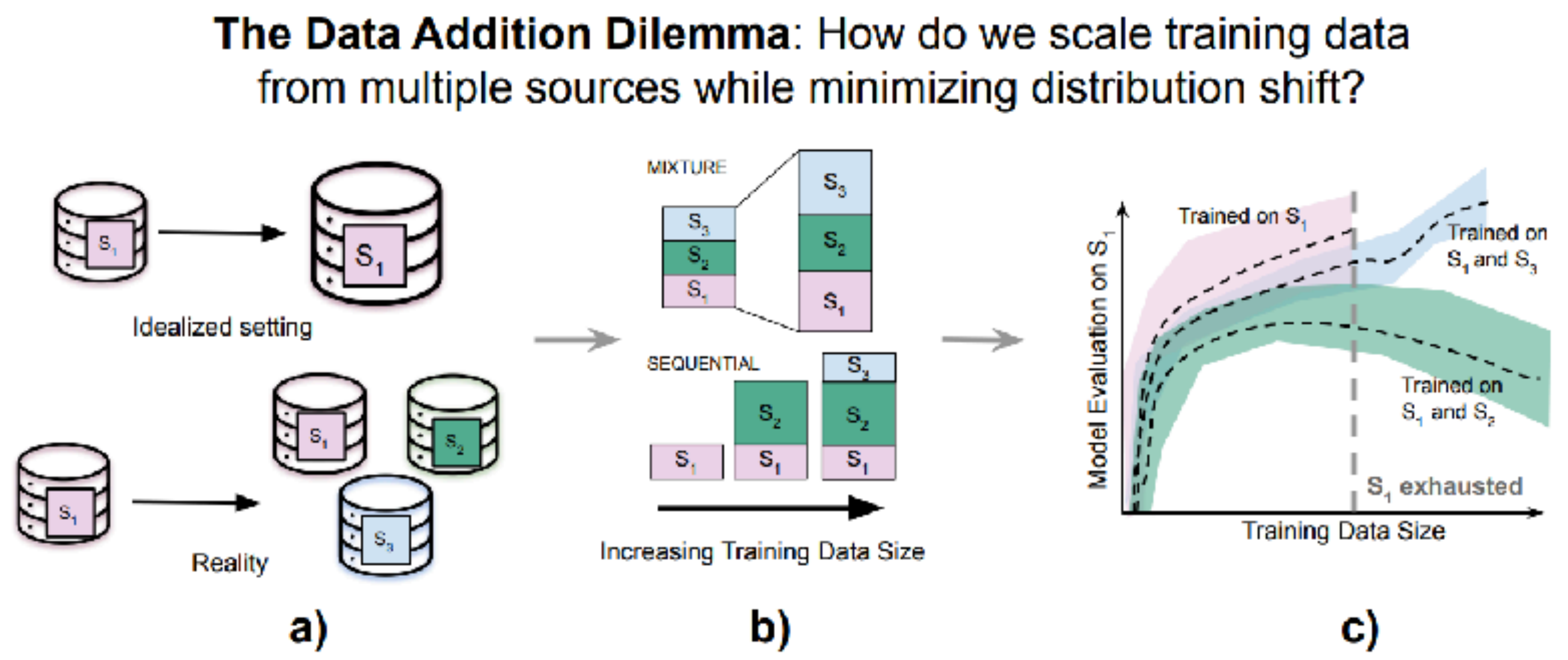
1. Changes in vocabulary size or tokenization are expected to rescale the loss by an overall factor. The parameterization of $L(N, D)$ (and all models of the loss) must naturally allow for such a rescaling.
2. Fixing D and sending $N \to \infty$, the overall loss should approach $L(D)$. Conversely, fixing N and sending $D \to \infty$ the loss must approach $L(N)$.
3. $L(N, D)$ should be analytic at $D = \infty$, so that it has a series expansion in $1/D$ with integer powers. Theoretical support for this principle is significantly weaker than for the first two.





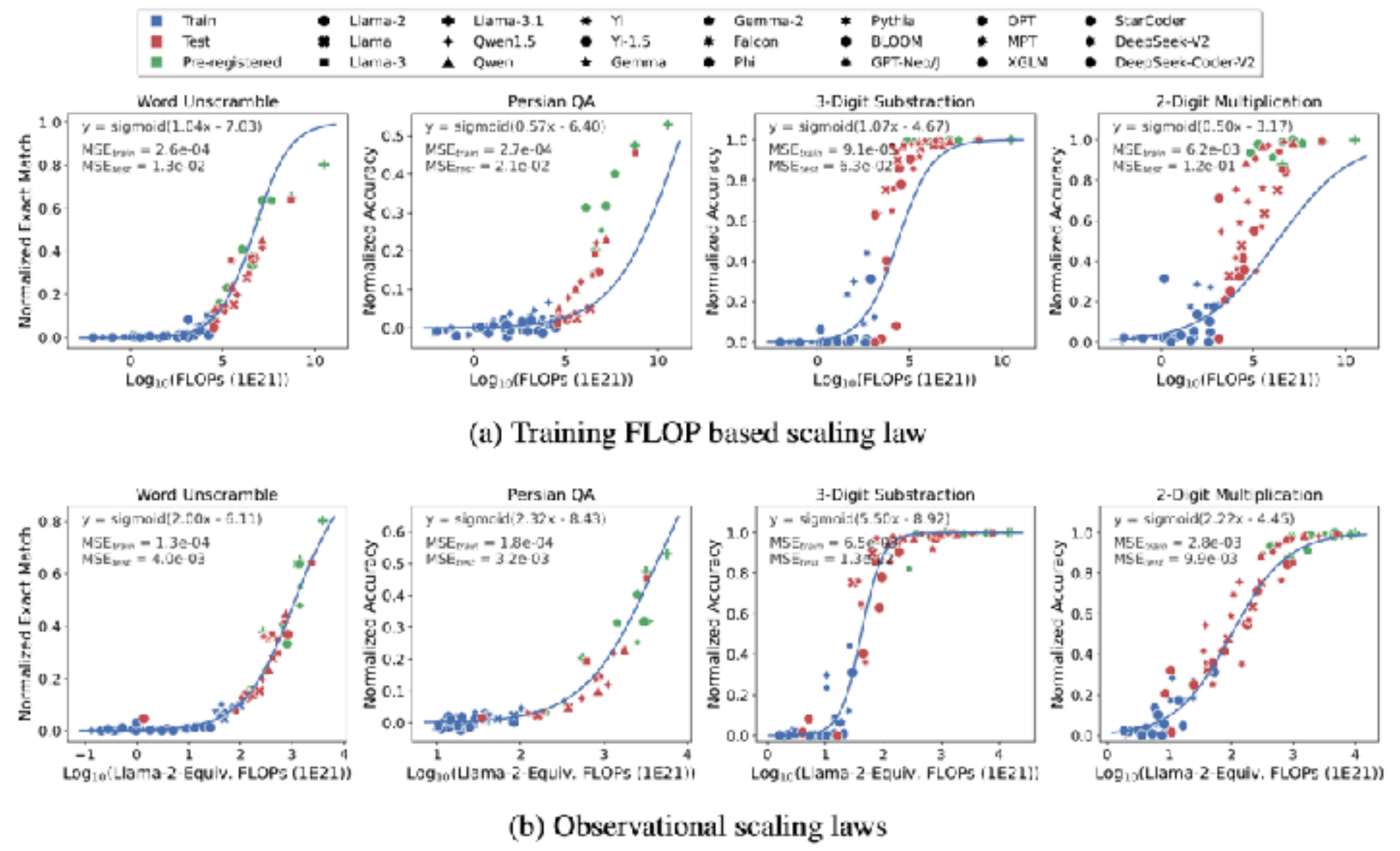
Discussion: Adding flexibility

More realistic models of data addition?



(Shen et al., 2024)

Observational data?



(Ruan et al., 2024)

Discussion: Other questions

- FLOP counting with large context length?
- Small errors have big implications—what is the right notion of “robustness" here?
- What is the “right” (a) functional form and (b) set of parameters?
- Theoretical explanations—can we do better than random features?
- Scaling laws that incorporate test-time computation?