Building LLM Reasoners

Lecture 5: Scaling Laws

Greg Durrett

Slide credit: Tatsu Hashimoto &
Percy Liang, Stanford CS 336

Administrative details and recap

Hyperparameter Optimization

Scaling Law History

Data Scaling Laws

Model Scaling Laws

Chinchilla: Data x Model

Putting it together

Administrative details and recap

Hyperparameter Optimization

Scaling Law History

Data Scaling Laws

Model Scaling Laws

Chinchilla: Data x Model

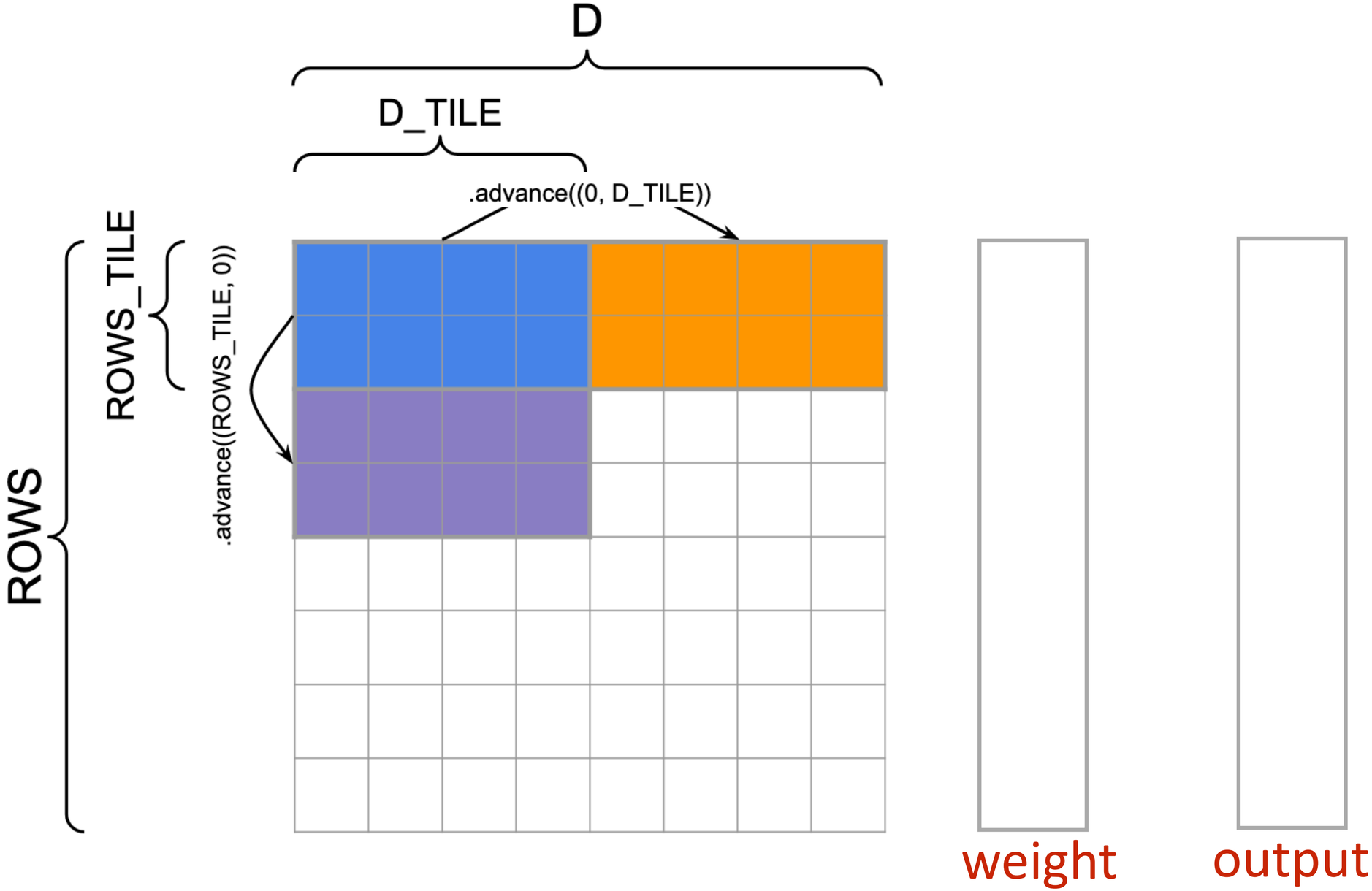
Putting it together

Administrative details and recap

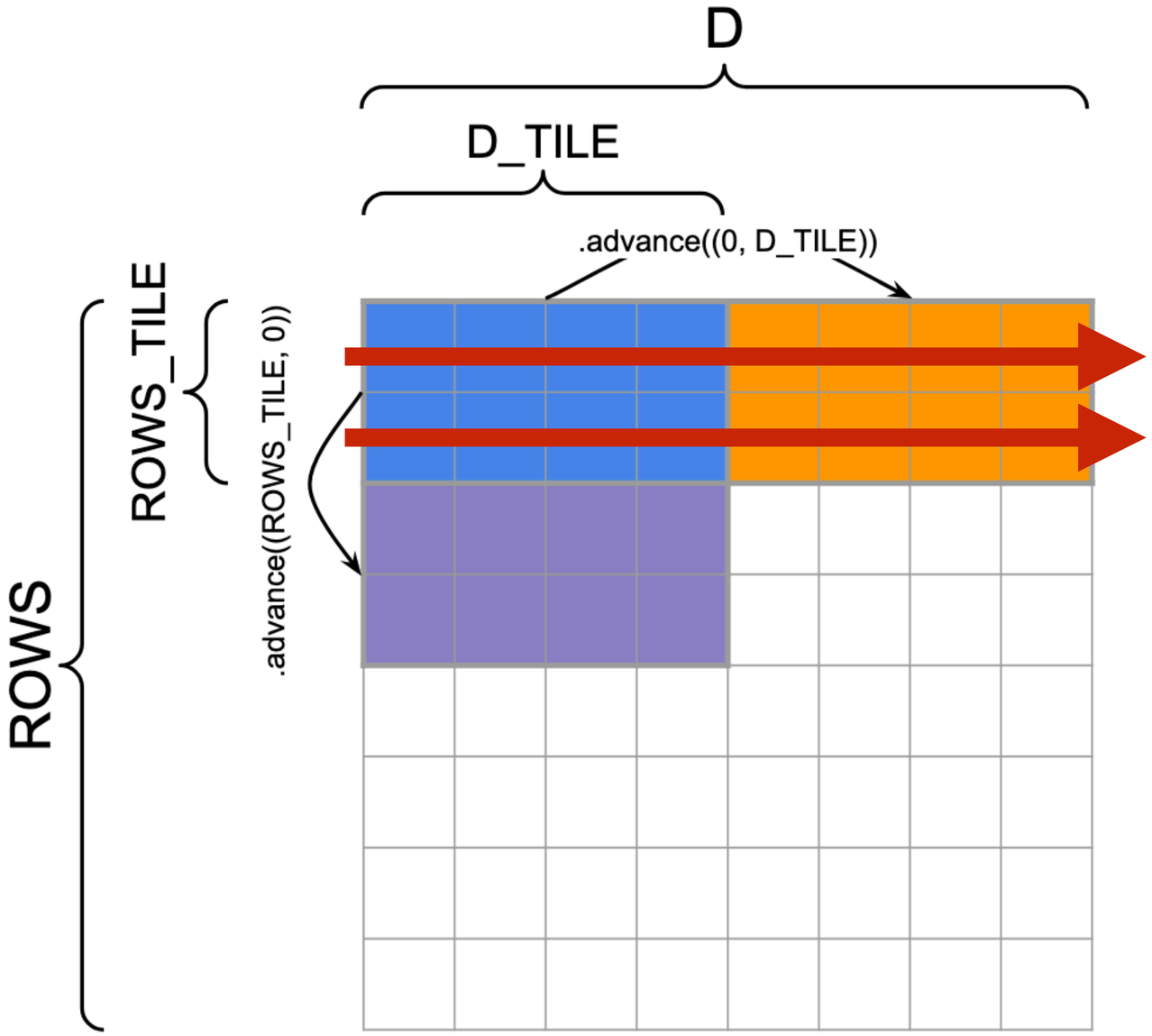
Administrivia

- Quiz 1 back
- Assignment 2 released, due in two weeks (along with quiz)
- Then Assignment 3 out, then midterm
- Final project: post-spring break

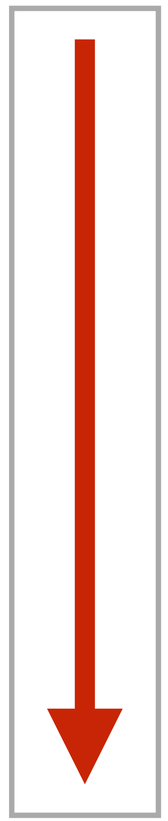
Triton: Weighted Sum



Triton: Weighted Sum



One execution of the kernel



weight

output

Memory coalescence depends on sizes

FlashAttention

Forward pass of attention:

$$\mathbf{S} = \mathbf{Q}\mathbf{K}^\top \in \mathbb{R}^{N\times N}, \quad \mathbf{P} = \mathrm{softmax}(\mathbf{S}) \in \mathbb{R}^{N\times N}, \quad \mathbf{O} = \mathbf{P}\mathbf{V} \in \mathbb{R}^{N\times d}$$

How to do this with as few reads and writes as possible?

Idea: construct **S** in tiles, use those to compute **P**, and build **O** directly

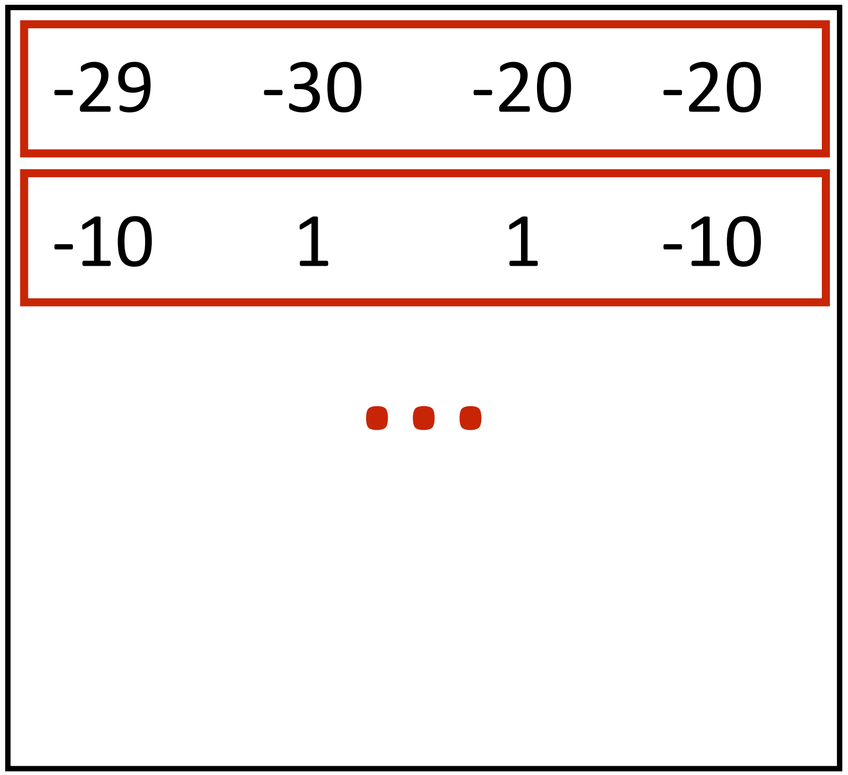
$$\text{Split } \mathbf{Q} \text{ into } T_q = \left\lceil \frac{N_q}{B_q} \right\rceil \text{ tiles } \mathbf{Q}_1, \ldots, \mathbf{Q}_{T_q} \text{ of size } B_q \times d$$
$$\text{Split } \mathbf{K}, \mathbf{V} \text{ into } T_k = \left\lceil \frac{N_k}{B_k} \right\rceil \text{ tiles } \mathbf{K}^{(1)}, \ldots, \mathbf{K}^{(T_k)} \text{ and } \mathbf{V}^{(1)}, \ldots, \mathbf{V}^{(T_k)} \text{ of size } B_k \times d$$

Each query becomes a row of attention. Handle in blocks of Bq rows

Why is this not so simple?

Softmax

S



Subtract off max (-20)

-9 -10 0 0

Exponentiate

0.0001 0.000045 1 1

Sum 2.000145

Normalize

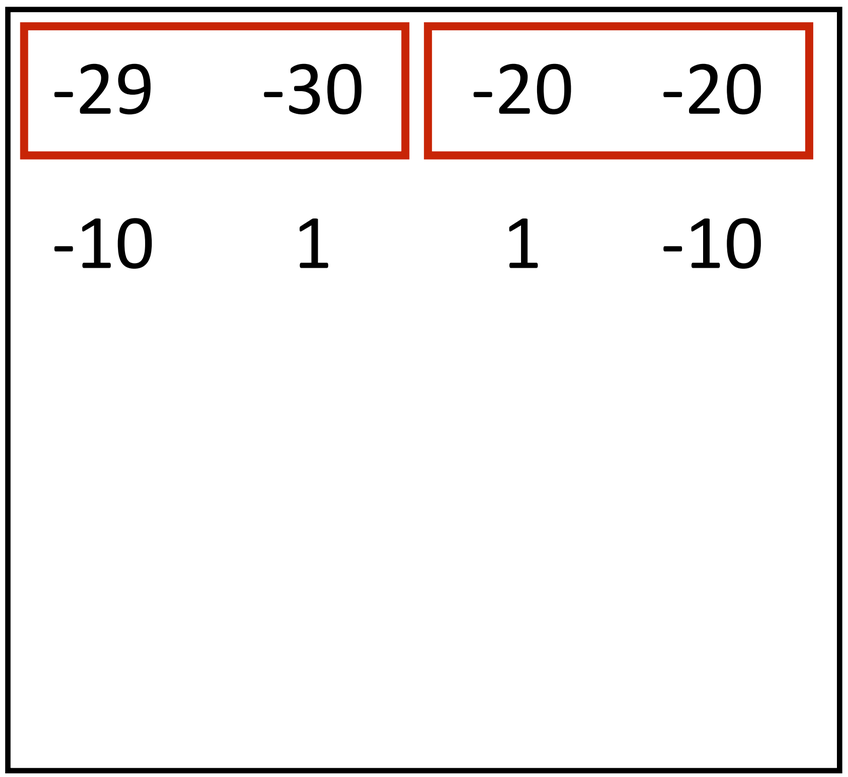
Then multiply by V to get O

$$\mathbf{P} = \mathrm{softmax}(\mathbf{S}) \in \mathbb{R}^{N \times N}$$

$$\mathbf{O} = \mathbf{PV} \in \mathbb{R}^{N \times d}$$

Softmax

S



Subtract off max (-20), but that's not in the first tile

Exponentiate

Sum (only partial)

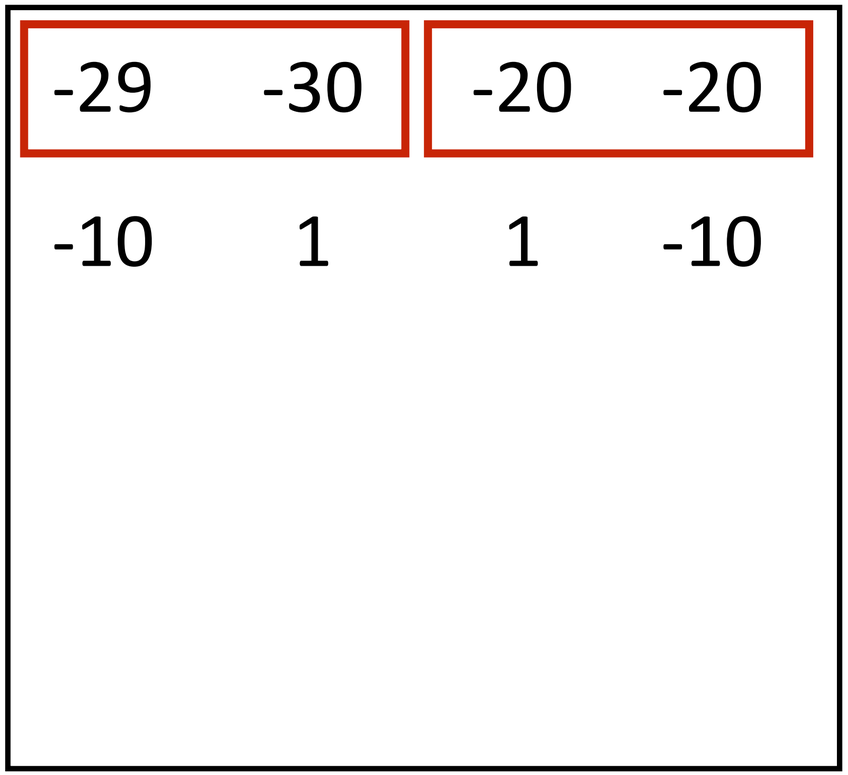
Normalize

Then multiply by V to get O

$$\mathbf{P} = \mathrm{softmax}(\mathbf{S}) \in \mathbb{R}^{N \times N}$$

$$\mathbf{O} = \mathbf{PV} \in \mathbb{R}^{N \times d}$$

Online Softmax



$$m^{(1)} = \text{rowmax}(\mathbf{S}^{(1)}) \in \mathbb{R}^{B_r}$$

$$\ell^{(1)} = \text{rowsum}(e^{\mathbf{S}^{(1)} - m^{(1)}}) \in \mathbb{R}^{B_r}$$

-29

partial sum with m1 “held out”

$$\tilde{\mathbf{P}}^{(1)} = \text{diag}(\ell^{(1)})^{-1} e^{\mathbf{S}^{(1)} - m^{(1)}} \in \mathbb{R}^{B_r \times B_c}$$

partially normalized

$$\mathbf{O}^{(1)} = \tilde{\mathbf{P}}^{(1)} \mathbf{V}^{(1)} = \text{diag}(\ell^{(1)})^{-1} e^{\mathbf{S}^{(1)} - m^{(1)}} \mathbf{V}^{(1)} \in \mathbb{R}^{B_r \times d}$$

$$m^{(2)} = \max(m^{(1)}, \text{rowmax}(\mathbf{S}^{(2)})) = m$$

-20

$$\ell^{(2)} = e^{m^{(1)} - m^{(2)}} \ell^{(1)} + \text{rowsum}(e^{\mathbf{S}^{(2)} - m^{(2)}})$$

new partial sum with m2 “held out”

Require: $\mathbf{Q} \in \mathbb{R}^{N_q \times d}$, $\mathbf{K}, \mathbf{V} \in \mathbb{R}^{N_k \times d}$, tile sizes B_q, B_k

Split $\mathbf{Q}$ into $T_q = \left\lceil \frac{N_q}{B_q} \right\rceil$ tiles $\mathbf{Q}_1, \ldots, \mathbf{Q}_{T_q}$ of size $B_q \times d$

Split $\mathbf{K}, \mathbf{V}$ into $T_k = \left\lceil \frac{N_k}{B_k} \right\rceil$ tiles $\mathbf{K}^{(1)}, \ldots, \mathbf{K}^{(T_k)}$ and $\mathbf{V}^{(1)}, \ldots, \mathbf{V}^{(T_k)}$ of size $B_k \times d$

for $i = 1, \ldots, T_q$ **do**

  Load $\mathbf{Q}_i$ from global memory

  Initialize $\mathbf{O}_i^{(0)} = \mathbf{0} \in \mathbb{R}^{B_q \times d}, l_i^{(0)} = 0 \in \mathbb{R}^{B_q}, m_i^{(0)} = -\infty \in \mathbb{R}^{B_q}$

  for $j = 1, \ldots, T_k$ **do**

    Load $\mathbf{K}^{(j)}, \mathbf{V}^{(j)}$ from global memory

    Compute tile of pre-softmax attention scores $\mathbf{S}_i^{(j)} = \frac{\mathbf{Q}_i (\mathbf{K}^{(j)})^\top}{\sqrt{d}} \in \mathbb{R}^{B_q \times B_k}$

    Compute $m_i^{(j)} = \max\left(m_i^{(j-1)}, \text{rowmax}\left(\mathbf{S}_i^{(j)}\right)\right) \in \mathbb{R}^{B_q}$

    Compute $\tilde{\mathbf{P}}_i^{(j)} = \exp\left(\mathbf{S}_i^{(j)} - m_i^{(j)}\right) \in \mathbb{R}^{B_q \times B_k}$

    Compute $l_i^{(j)} = \exp\left(m_i^{(j-1)} - m_i^{(j)}\right) l_i^{(j-1)} + \text{rowsum}\left(\tilde{\mathbf{P}}_i^{(j)}\right) \in \mathbb{R}^{B_q}$

    Compute $\mathbf{O}_i^{(j)} = \text{diag}\left(\exp\left(m_i^{(j-1)} - m_i^{(j)}\right)\right) \mathbf{O}_i^{(j-1)} + \tilde{\mathbf{P}}_i^{(j)} \mathbf{V}^{(j)}$

  end for

  Compute $\mathbf{O}_i = \text{diag}\left(l_i^{(T_k)}\right)^{-1} \mathbf{O}_i^{(T_k)}$

  Compute $L_i = m_i^{(T_k)} + \log\left(l_i^{(T_k)}\right)$

  Write $\mathbf{O}_i$ to global memory as the i-th tile of $\mathbf{O}$.

  Write L_i to global memory as the i-th tile of L.

end for

Return the output $\mathbf{O}$ and the logsumexp L.

why is *L* needed at all? backward pass!

for $i = 1, \ldots, T_q$ **do**

Load $\mathbf{Q}_i$ from global memory

Initialize $\mathbf{O}_i^{(0)} = \mathbf{0} \in \mathbb{R}^{B_q \times d}, l_i^{(0)} = 0 \in \mathbb{R}^{B_q}, m_i^{(0)} = -\infty \in \mathbb{R}^{B_q}$

for $j = 1, \ldots, T_k$ **do**

Load $\mathbf{K}^{(j)}, \mathbf{V}^{(j)}$ from global memory

Compute tile of pre-softmax attention scores $\mathbf{S}_i^{(j)} = \frac{\mathbf{Q}_i (\mathbf{K}^{(j)})^\top}{\sqrt{d}} \in \mathbb{R}^{B_q \times B_k}$

Compute $m_i^{(j)} = \max\left(m_i^{(j-1)}, \text{rowmax}\left(\mathbf{S}_i^{(j)}\right)\right) \in \mathbb{R}^{B_q}$

Compute $\tilde{\mathbf{P}}_i^{(j)} = \exp\left(\mathbf{S}_i^{(j)} - m_i^{(j)}\right) \in \mathbb{R}^{B_q \times B_k}$

Compute $l_i^{(j)} = \exp\left(m_i^{(j-1)} - m_i^{(j)}\right) l_i^{(j-1)} + \text{rowsum}\left(\tilde{\mathbf{P}}_i^{(j)}\right) \in \mathbb{R}^{B_q}$

Compute $\mathbf{O}_i^{(j)} = \text{diag}\left(\exp\left(m_i^{(j-1)} - m_i^{(j)}\right)\right) \mathbf{O}_i^{(j-1)} + \tilde{\mathbf{P}}_i^{(j)} \mathbf{V}^{(j)}$

end for

Compute $\mathbf{O}_i = \text{diag}\left(l_i^{(T_k)}\right)^{-1} \mathbf{O}_i^{(T_k)}$

Compute $L_i = m_i^{(T_k)} + \log\left(l_i^{(T_k)}\right)$

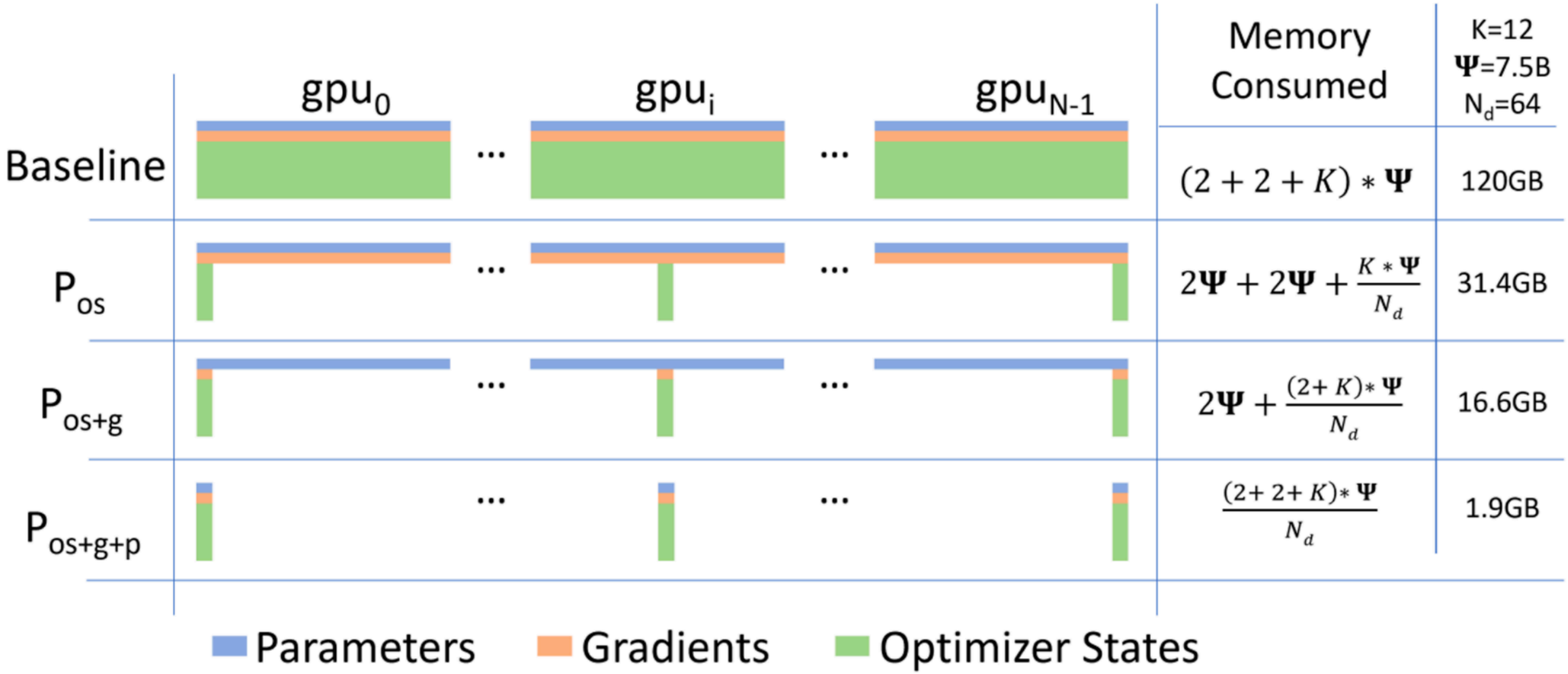
m, *l* are easy to store

finally normalize by *l*

sanity-check: why is *m* not needed here?

ZeRO

- Split computation across GPUs



Administrative details and recap

Hyperparameter Optimization

Scaling Law History

Data Scaling Laws

Model Scaling Laws

Chinchilla: Data x Model

Putting it together

Hyperparameter Optimization

Hyperparameter Optimization

- How do we typically optimize hyperparameters?
- Define a grid of values

Learning rate: {1e-6, 3e-6, 1e-5, 3e-5}…

Weight decay: {0, 1e-4, …}

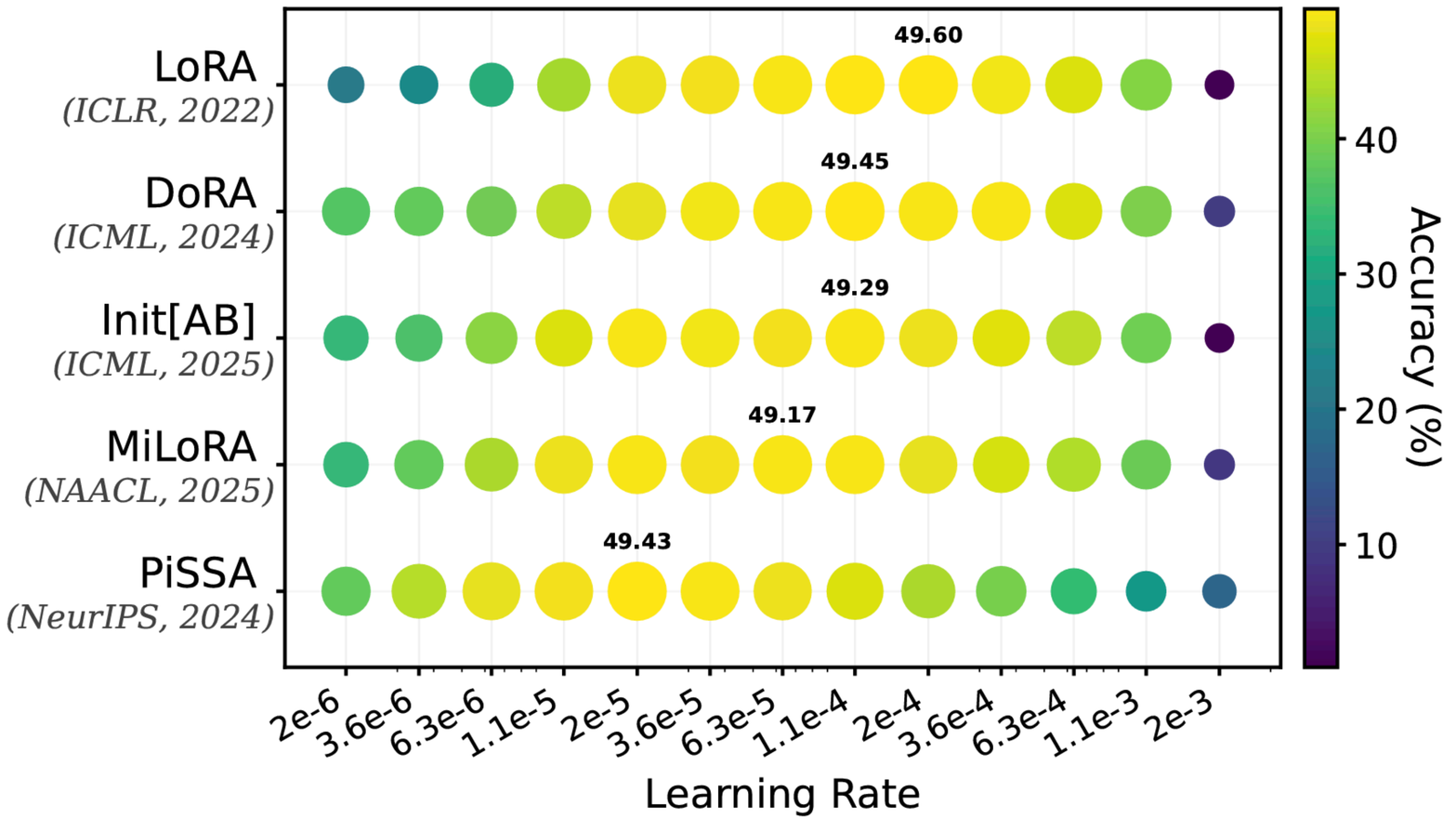
Batch size: {64, 128, 256}

- Loop over all combinations, pick the one which is best on a validation set
- What are the drawbacks of this?

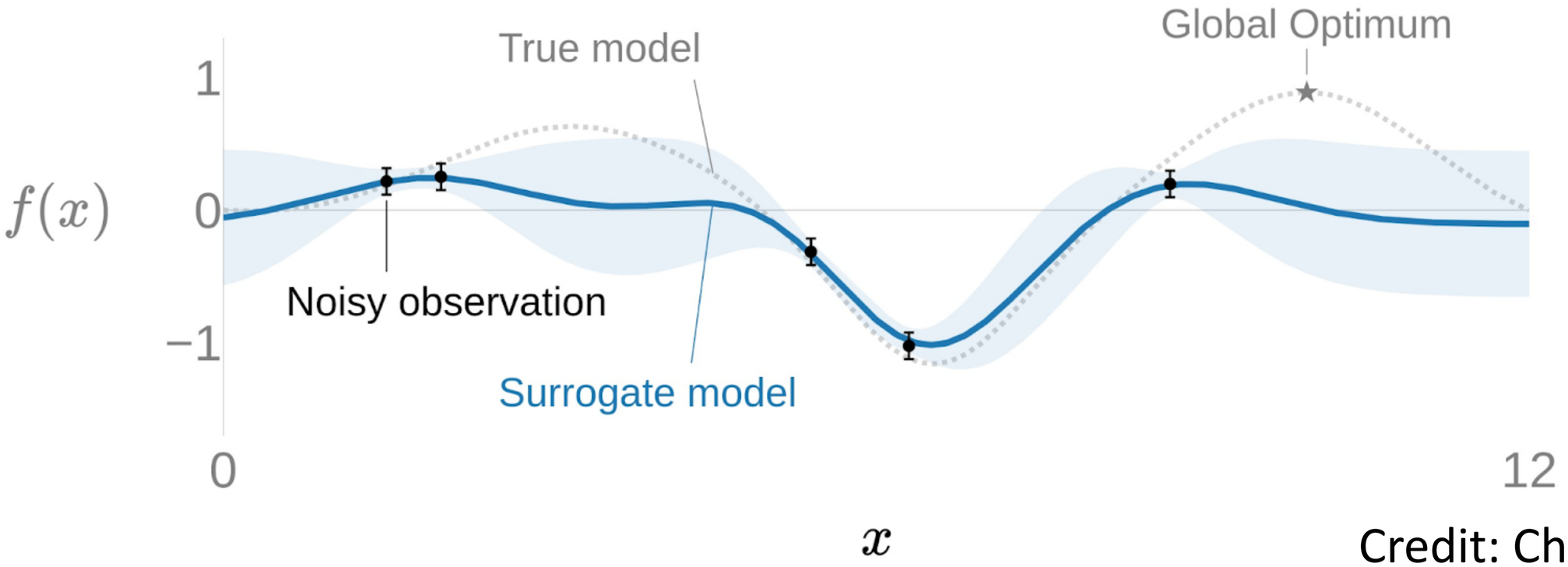
How important is tuning?

Learning Rate Matters: Vanilla LoRA May Suffice for LLM Fine-tuning

Yu-Ang Lee [1] Ching-Yun Ko [2] Pin-Yu Chen [2] Mi-Yen Yeh [1 3]



Bayesian Optimization



Credit: ChatGPT

- Step 1: Define surrogate model (e.g., Gaussian process)
- Step 2: Decide where to evaluate next (where will give us expected improvement?)
- Step 3: Evaluate and update surrogate

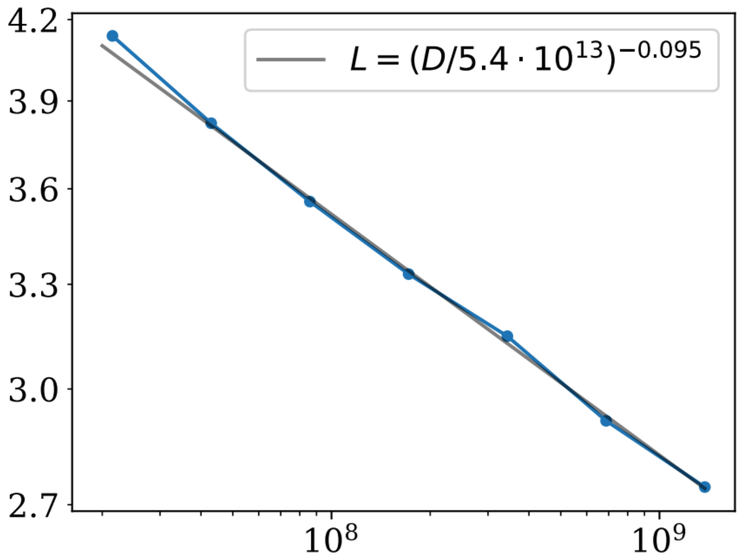
How slow is this?

- Grid search: 3x3x3 grid is 27 values, can maybe get away with ~10 evaluations if we're smart
- Bayesian HPO: Even in low-dimensional spaces, we likely need at least 20 evaluations

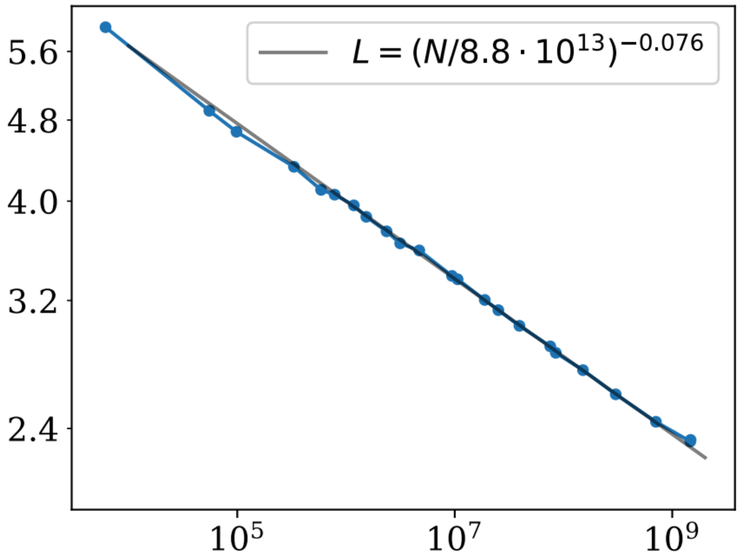
- How much does it cost to train a frontier model? How many evaluations can we afford?

1

Scaling Laws



Dataset Size



Parameters

Kaplan et al. (2020)

Scaling Laws

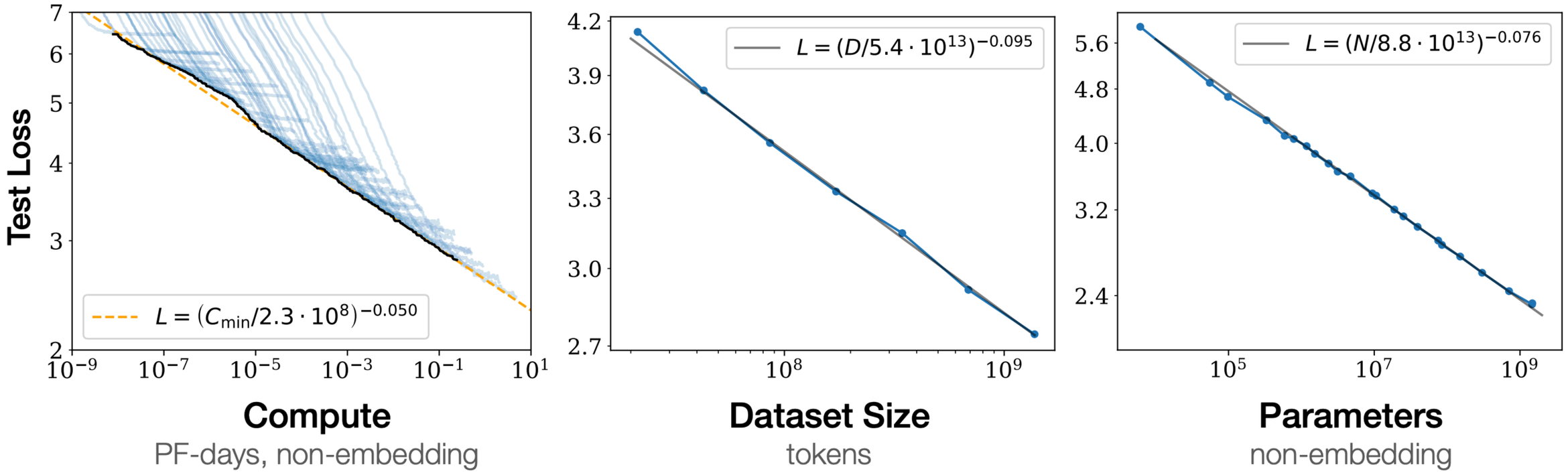


Figure 1 Language modeling performance improves smoothly as we increase the model size, datasetset size, and amount of compute[2] used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when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

- What does a relationship like this allow us to do?

Kaplan et al. (2020)

Administrative details and recap

Hyperparameter Optimization

Scaling Law History

Data Scaling Laws

Model Scaling Laws

Chinchilla: Data x Model

Putting it together

Scaling Law History

Slide credit: Tatsu Hashimoto

Early Scaling Laws

Learning Curves: Asymptotic Values and Rate of Convergence

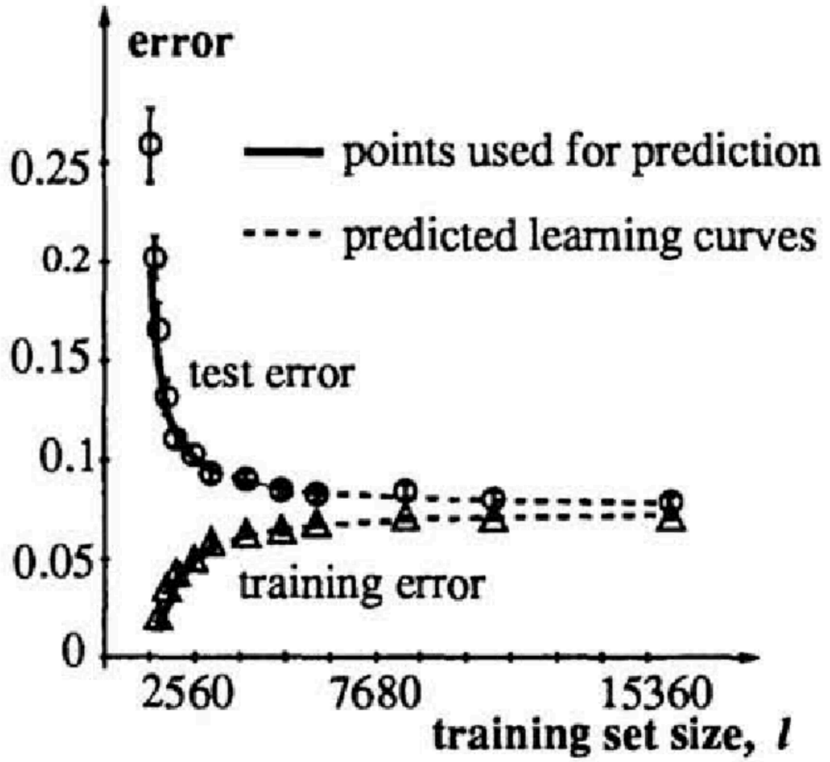
Corinna Cortes, L. D. Jackel, Sara A. Solla, Vladimir Vapnik, and John S. Denker
AT&T Bell Laboratories
Holmdel, NJ 07733

Abstract

Training classifiers on large databases is computationally demanding. It is desirable to develop efficient procedures for a reliable prediction of a classifier's suitability for implementing a given task, so that resources can be assigned to the most promising candidates or freed for exploring new classifier candidates. We propose such a practical and principled predictive method. Practical because it avoids the costly procedure of training poor classifiers on the whole training set, and principled because of its theoretical foundation. The effectiveness of the proposed procedure is demonstrated for both single- and multi-layer networks.

A typical example of learning curves is shown in Fig. 2. The test error is always larger than the training error, but asymptotically they reach a common value, a. We model the errors for large sizes of the training set as power-law decays to the

$$\mathcal{E}_{\text{test}} = a + \frac{b}{l^{\alpha}} \qquad \text{and} \qquad \mathcal{E}_{\text{train}} = a - \frac{c}{l^{\beta}}$$



Early Scaling Laws

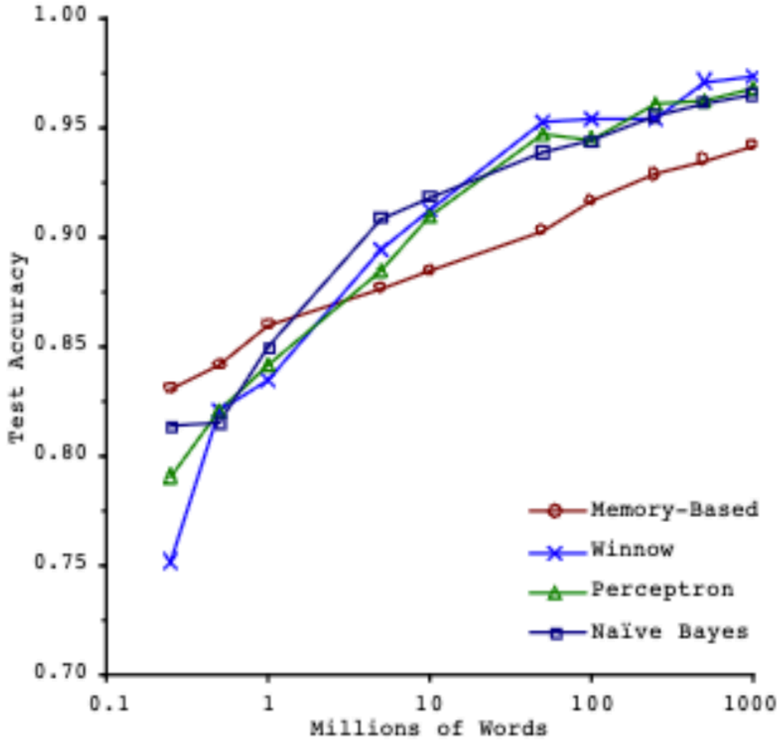


Figure 1. Learning Curves for Confusion Set Disambiguation

these results suggest that we may want to reconsider the trade-off between spending time and money on algorithm development versus spending it on corpus development. At least for the problem of confusable disambiguation, none of the learners tested is close to asymptoting in performance at the training corpus size commonly employed by the field.

Log-linear scaling with data [Banko and Brill ‘01]

Early Scaling Laws

Large Language Models in Machine Translation

Thorsten Brants **Ashok C. Popat** **Peng Xu** **Franz J. Och** **Jeffrey Dean**

Google, Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94303, USA
{brants,popat,xp,och,jeff}@google.com

2007

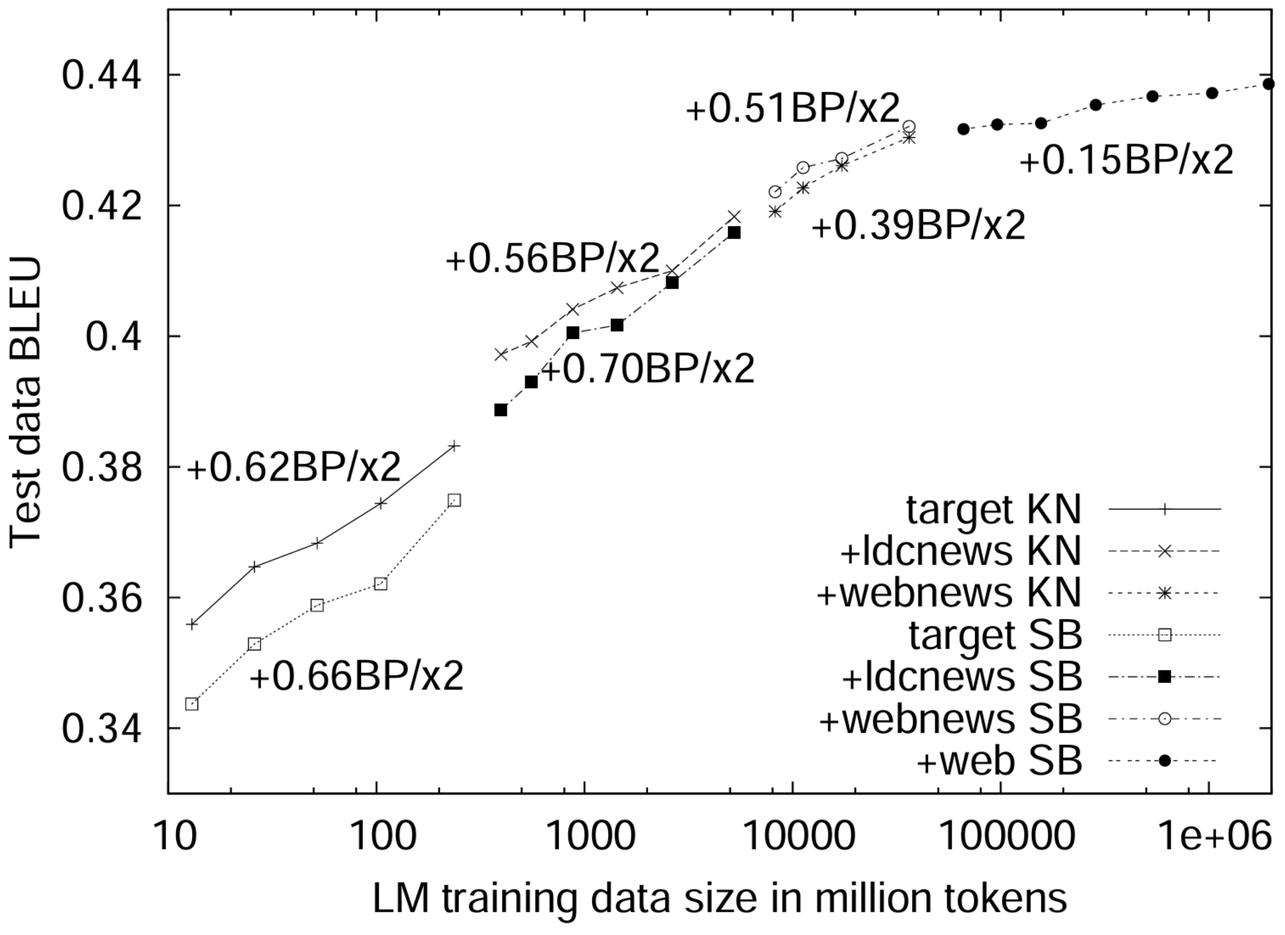
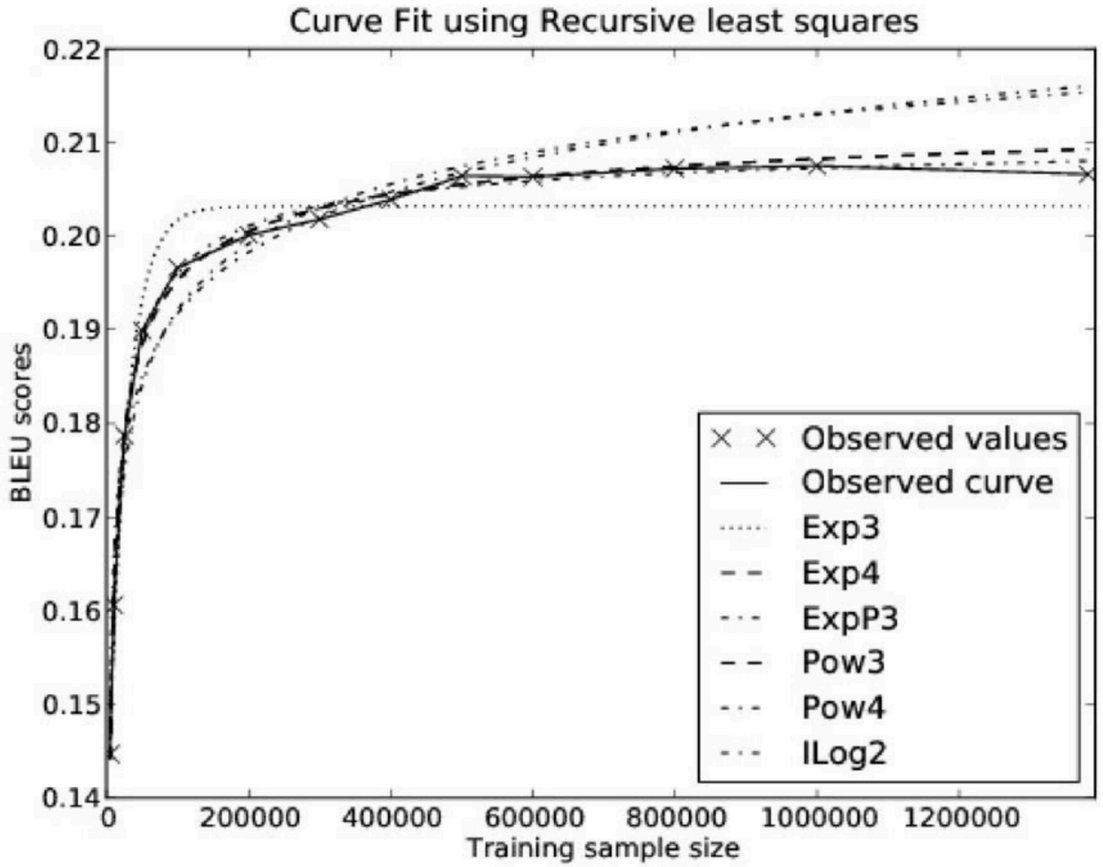


Figure 5: BLEU scores for varying amounts of data using Kneser-Ney (KN) and Stupid Backoff (SB).

Early Scaling Laws



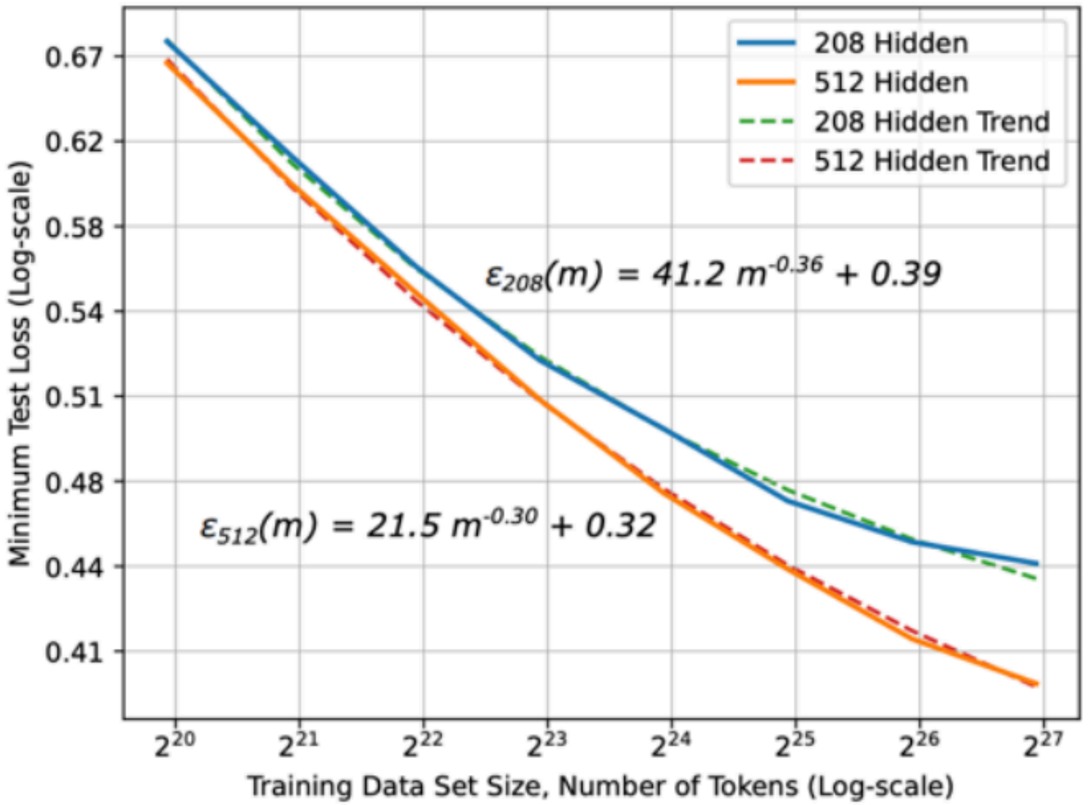
Model	Formula
Exp_3	$y = c - e^{-ax+b}$
Exp_4	$y = c - e^{-ax^{\alpha}+b}$
ExpP_3	$y = c - e^{(x-b)^{\alpha}}$
Pow_3	$y = c - ax^{-\alpha}$
Pow_4	$y = c - (-ax+b)^{-\alpha}$
ILog_2	$y = c - (a/\log x)$

Table 1: Curve families.

Early tests of functional forms

Kolachina et al 2012 – power law relation between data and downstream performance

Early Scaling Laws



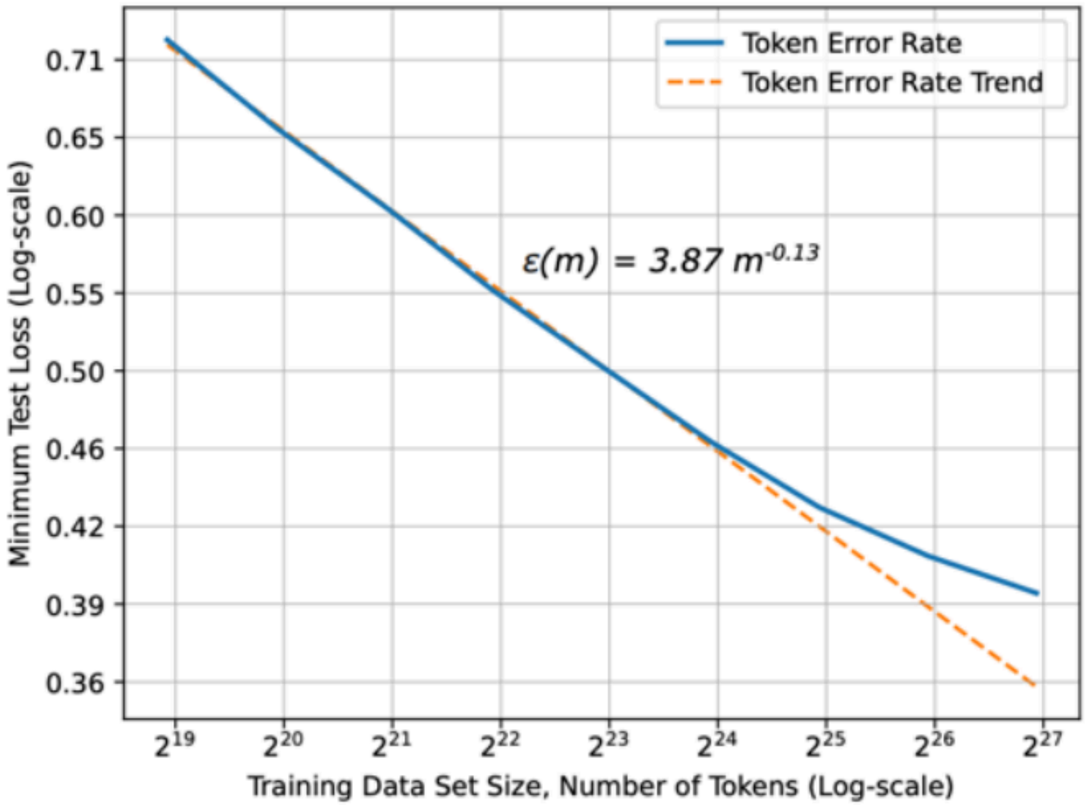
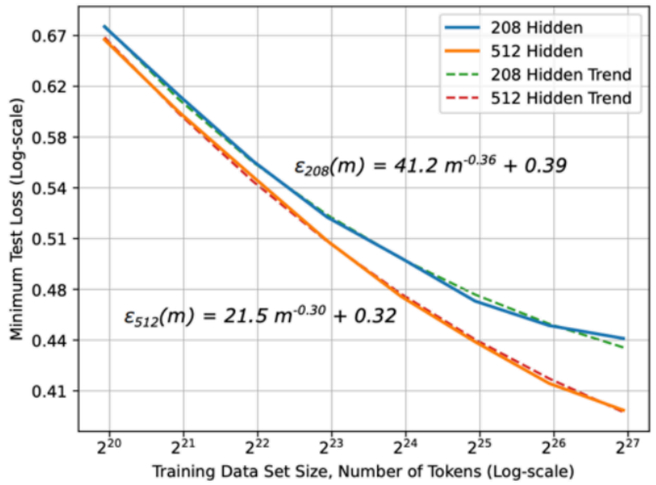


Figure 1: Neural machine translation learning curves. Left: the learning curves for separate models follow $\varepsilon(m) = \alpha m^{\beta_g} + \gamma$. Right: composite learning curve of best-fit model at each data set size.

Hestness et al., 2017

Early Scaling Laws



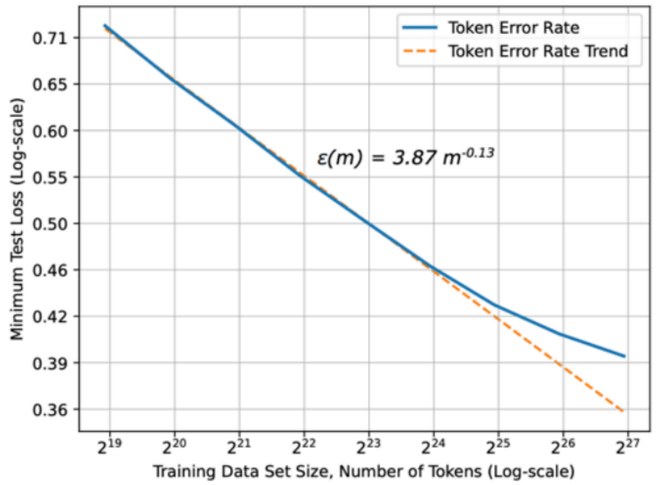
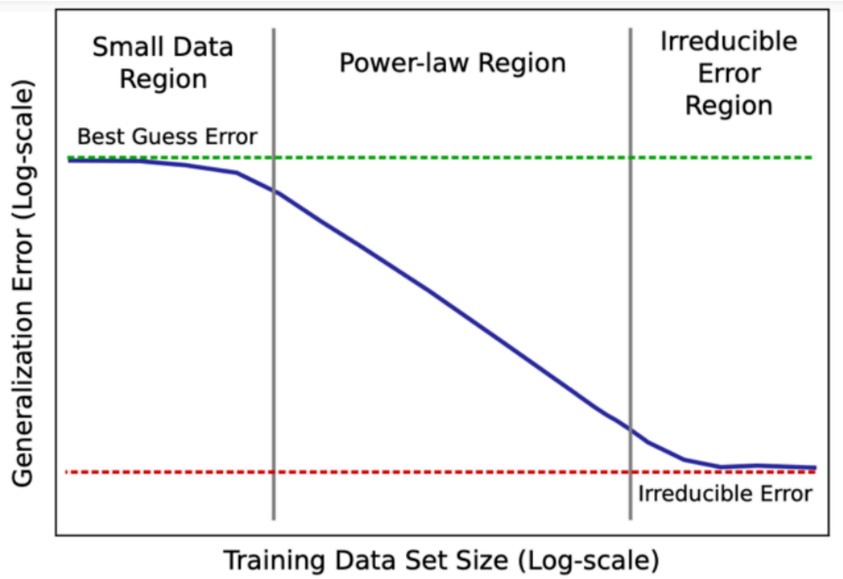


Figure 1: Neural machine translation learning curves. Left: the learning curves for separate models follow $\varepsilon(m) = \alpha m^{\beta_g} + \gamma$. Right: composite learning curve of best-fit model at each data set size.



Earliest 'large scale neural' scaling work: Hestness 2017

Predictable scaling on many tasks (MT, LM, Speech) and hypothesized scaling shape.

Early Scaling Laws

Computational Limits: If we have identified a desirable model to scale to larger training sets, the next potential limitation is the speed of computation. In some cases, training large models on very large data sets would take months or years of critical path compute time, making these training runs impractical for any real world problem on existing systems. However, predictable learning and model size curves may offer a way to project the compute requirements to reach a particular accuracy level. The compute requirements could inform decisions about how to scale computational capacity to unlock these compute-limited applications.

The Performance-Accuracy Trade-off: Many DL software and hardware techniques impose a trade-off between model accuracy and the speed of computation. Learning curves and model size growth can indicate whether these techniques could regain lost accuracy by improving the speed of computation. For example, low-precision computation/quantization and sparse models give up some model accuracy (e.g., up to 20%) in order to improve compute throughput. If the compute throughput improvements allow DL developers to train larger models on larger data sets, these accuracy losses might be easily recoverable.

Hestness et al., 2017

Administrative details and recap

Hyperparameter Optimization

Scaling Law History

Data Scaling Laws

Model Scaling Laws

Chinchilla: Data x Model

Putting it together

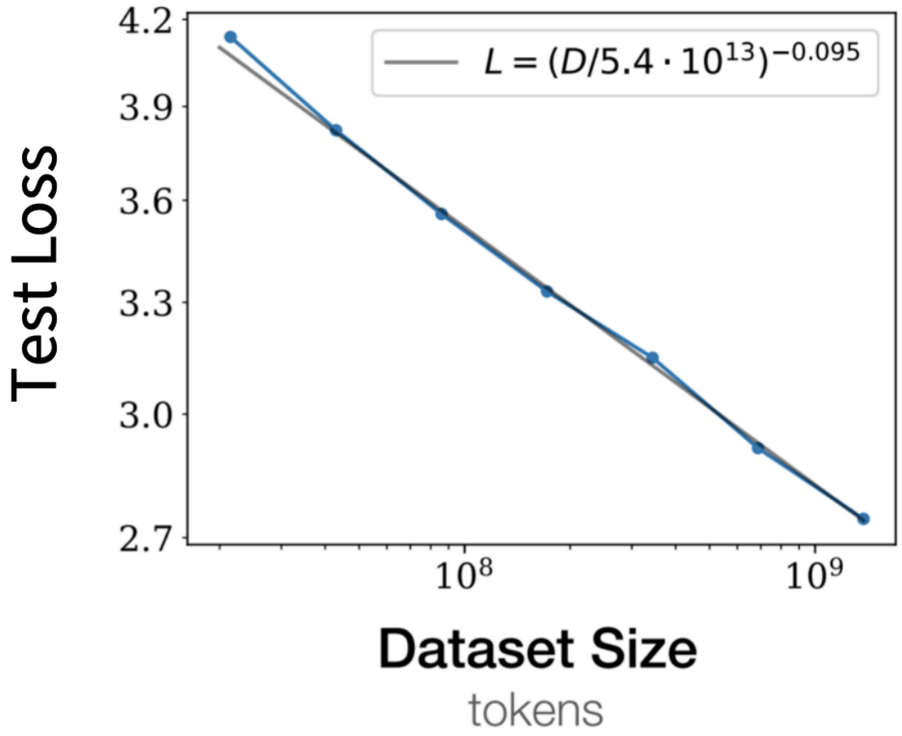
Data Scaling Laws

Slide credit: Tatsu Hashimoto

Data Scaling Laws

First, an empirical observation

Loss and dataset size is linear on a log-log plot



"Scale-free" or
"Power law"

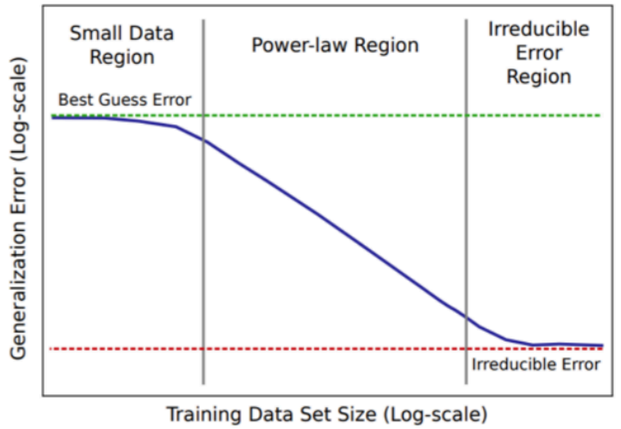
(For language modeling, from Kaplan+ 2020)

Foundations

Q: Why do scaling laws show up?

We know error should be monotone

But why is it a power law / linear in log-log?



A (?): Estimation error naturally decays polynomially.

But this answer may take a moment to understand. Let's work through an example.

Example: If our task is to estimate the mean of a dataset, what's the scaling law?

Example: Mean Estimation

Input: $x_1 \ \ldots x_n \ \sim N(\mu, \sigma^2)$

Task: estimate the average as $\hat{\mu} = \frac{\sum_i x_i}{n}$

What's the error?

Example: Mean Estimation

Input: $x_1 \ \ldots x_n \ \sim N(\mu, \sigma^2)$

Task: estimate the average as $\hat{\mu} = \frac{\sum_i x_i}{n}$

What's the error? By the Central Limit Theorem:

$$\mathrm{E}[(\hat{\mu} - \mu)^2] = \frac{\sigma^2}{n}$$

This is a 'scaling law'

$$\log(Error) = -\log n + 2\log\sigma$$

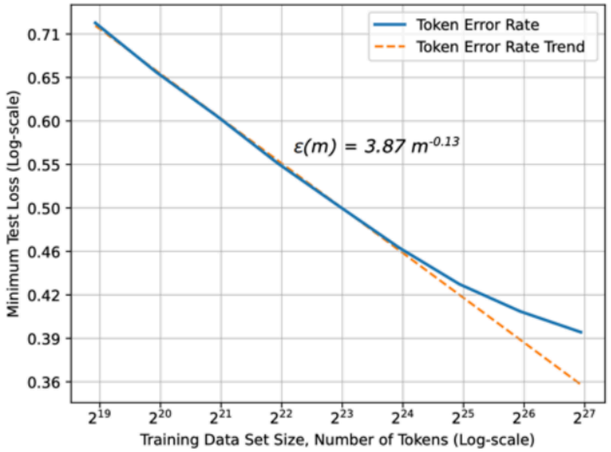
More generally, any polynomial rate $1/n^{\alpha}$ is a scaling law

Scaling Laws in Practice

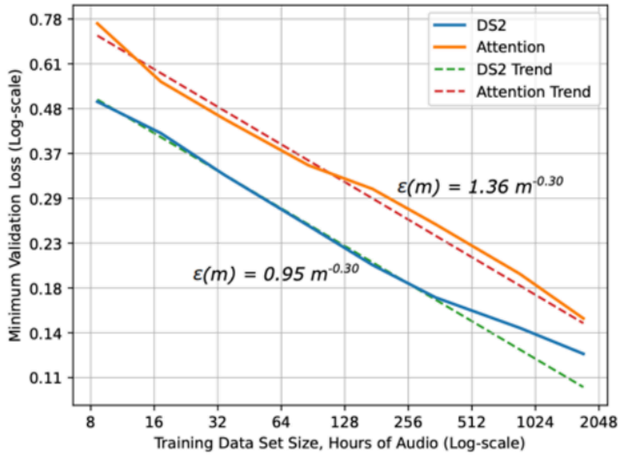
Fact: Similar arguments show most 'classical' models (regression, etc) have $\frac{1}{n}$ scaling

This means we should see $y = -x + C$

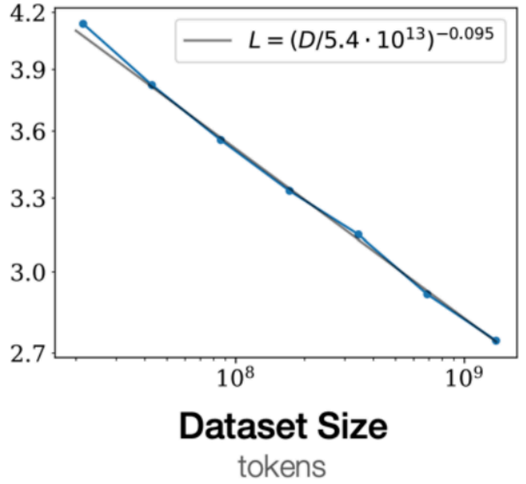
What do we find in neural scaling laws?



Machine translation



Speech



Language modeling

Very different from predictions.. Why might this be?

Function Approximation

Neural nets can approximate arbitrary functions. Lets turn that into an example.

Input: $x_1 \ldots x_n$ uniform in 2D unit box. $y_i = f(x_i) + N(0,1)$

Task: estimate f(x) by averaging over samples in the box

Approach: cut up the 2D space into boxes with length $n^{-\frac{1}{4}}$

What's our estimation error?

Informally, we have $\sqrt{n}$ boxes, each box gets $\sqrt{n}$ samples.

$$Error \approx \frac{1}{\sqrt{n}} + (other\ smoothness\ terms)$$

(essentially the error of the mean estimate in the box)

Function Approximation

Neural nets can approximate arbitrary functions. Lets turn that into an example.

Input: $x_1 \ldots x_n$ uniform in 2D unit box. $y_i = f(x_i) + N(0,1)$

Task: estimate f(x) by averaging over samples in the box

Approach: cut up the 2D space into boxes with length $n^{-\frac{1}{4}}$

What's our estimation error?

Informally, we have $\sqrt{n}$ boxes, each box gets $\sqrt{n}$ samples.

$$Error \approx \frac{1}{\sqrt{n}} + (other\ smoothness\ terms)$$

(essentially the error of the mean estimate in the box)

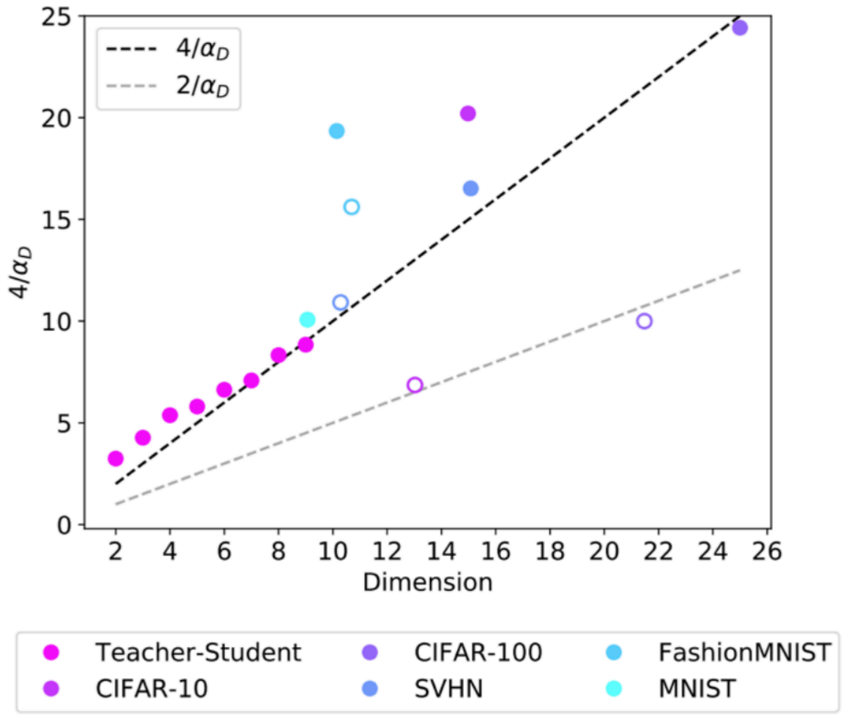
In d-dimensions, this becomes $Error = n^{-1/d}$ - **This means scaling is $y = -\frac{1}{d}x + C$**

Takeaway: flexible 'nonparametric' learning has dimension dependent scaling laws.

Intrinsic Dimensionality

Some have made the following argument (Bahri 2021)

1. Scaling laws arise due to polynomial rates of learning $\frac{1}{n^{\alpha}}$
2. Some argue the slope α is closely connected to the *intrinsic dimensionality* of the data.



But estimators of intrinsic dimension are sketchy, and this is not airtight..

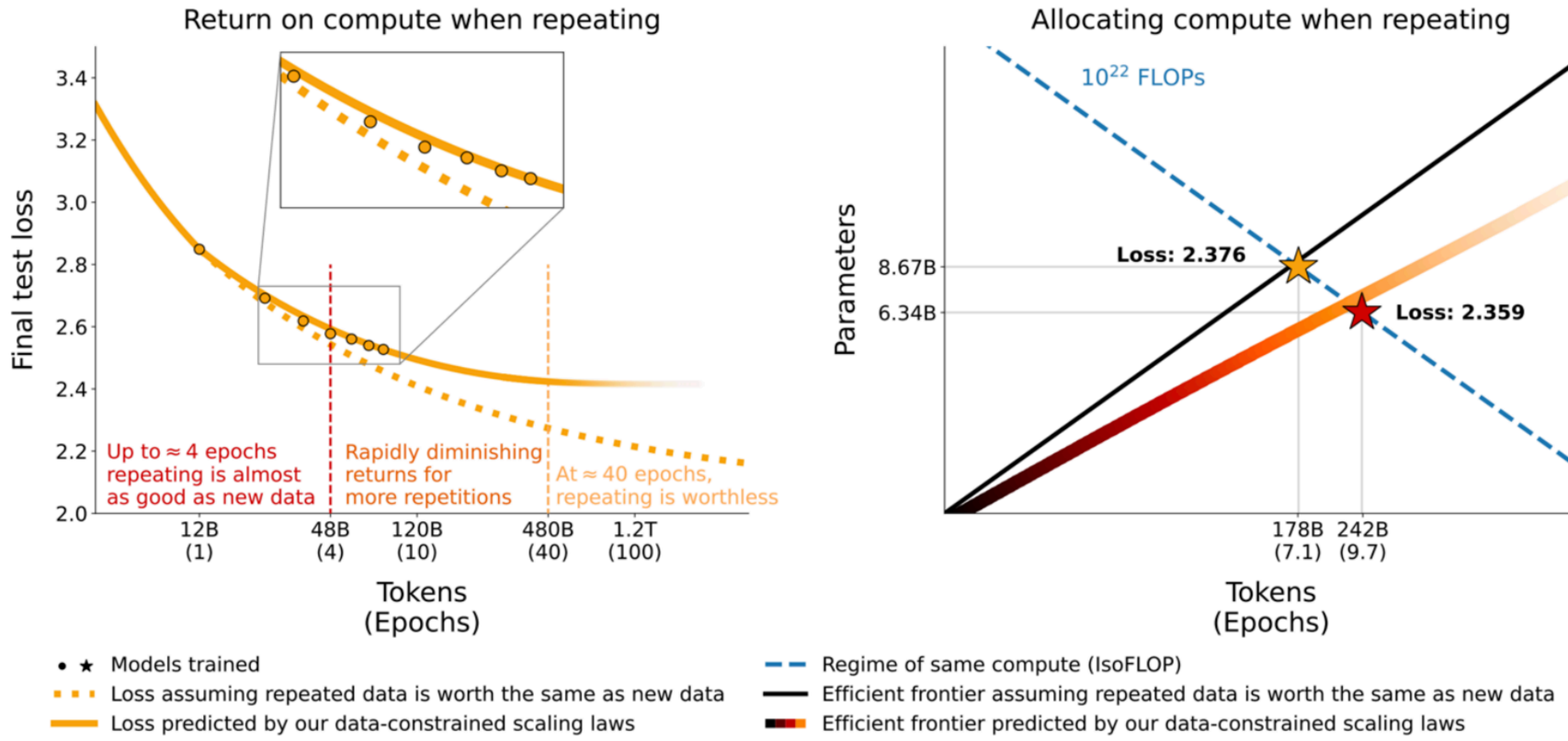
Intrinsic Dimensionality

Data scaling thus far: how does dataset size relate to performance?
Related question: how does dataset *composition* affect performance

- Picking optimal data mixture using small scale models
  (we'll revisit this later)
- Deciding whether to repeat data or not
- Combing the two and balancing quality with repetition rate

Repeating Data

In practice, we have finite data – how does repeating examples affect scaling?



Muennighoff et al. (2023)

$$D' = U_D + U_D R_D^* (1 - e^{\frac{-R_D}{R_D^*}}) \ .$$

D' = Effective data
Ud = Unique tokens
Rd* = Constant
Rd = Repetition

Data Scaling Laws

- Remarkably linear relationship between log-data size and log-error
- Holds across domains and models
- Theory understanding: similar to generalization bounds: mean estimation example
- Applications: data collection / curation

Administrative details and recap

Hyperparameter Optimization

Scaling Law History

Data Scaling Laws

Model Scaling Laws

Chinchilla: Data x Model

Putting it together

Model Scaling Laws

Slide credit: Tatsu Hashimoto

Model Scaling Laws

Our motivation: how can we efficiently design huge LMs?

- LSTMs vs Transformers
- Adam vs SGD

How should we allocate our limited resources?

- Train models longer vs train bigger models?
- Collect more data vs get more GPUs?

Scaling laws provide a simple procedure to answer these.

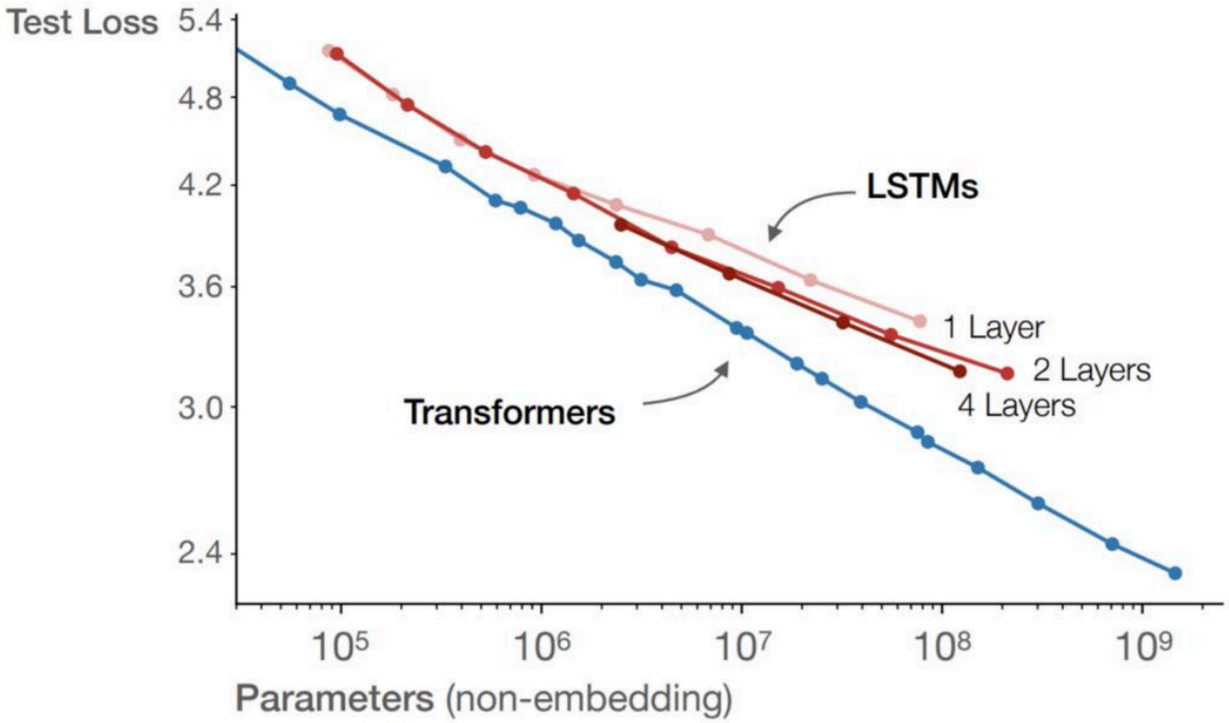
Consider: 1. Architecture; 2. Optimizer; 3. Aspect ratio/depth; 4. Batch size

Architecture

Q: Are transformers better than LSTMs?

Brute force way: spend tens of millions to train a LSTM GPT-3

Scaling law way:

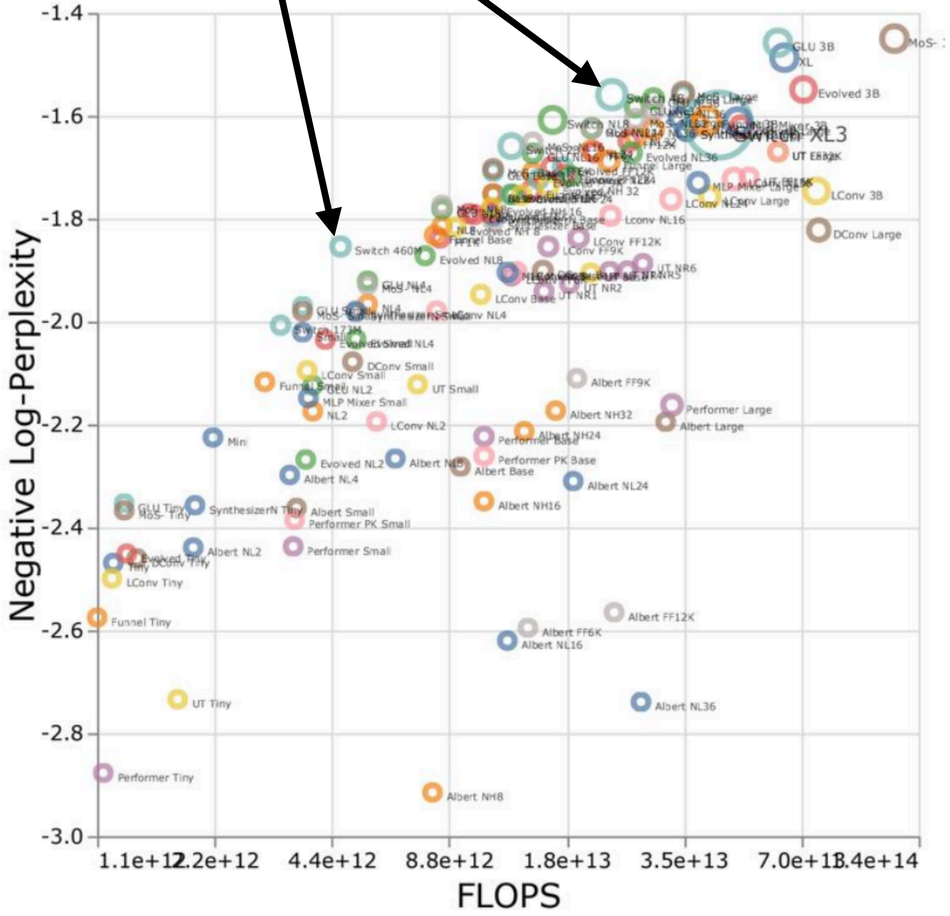


[Kaplan+ 2021]

Switch Transformer

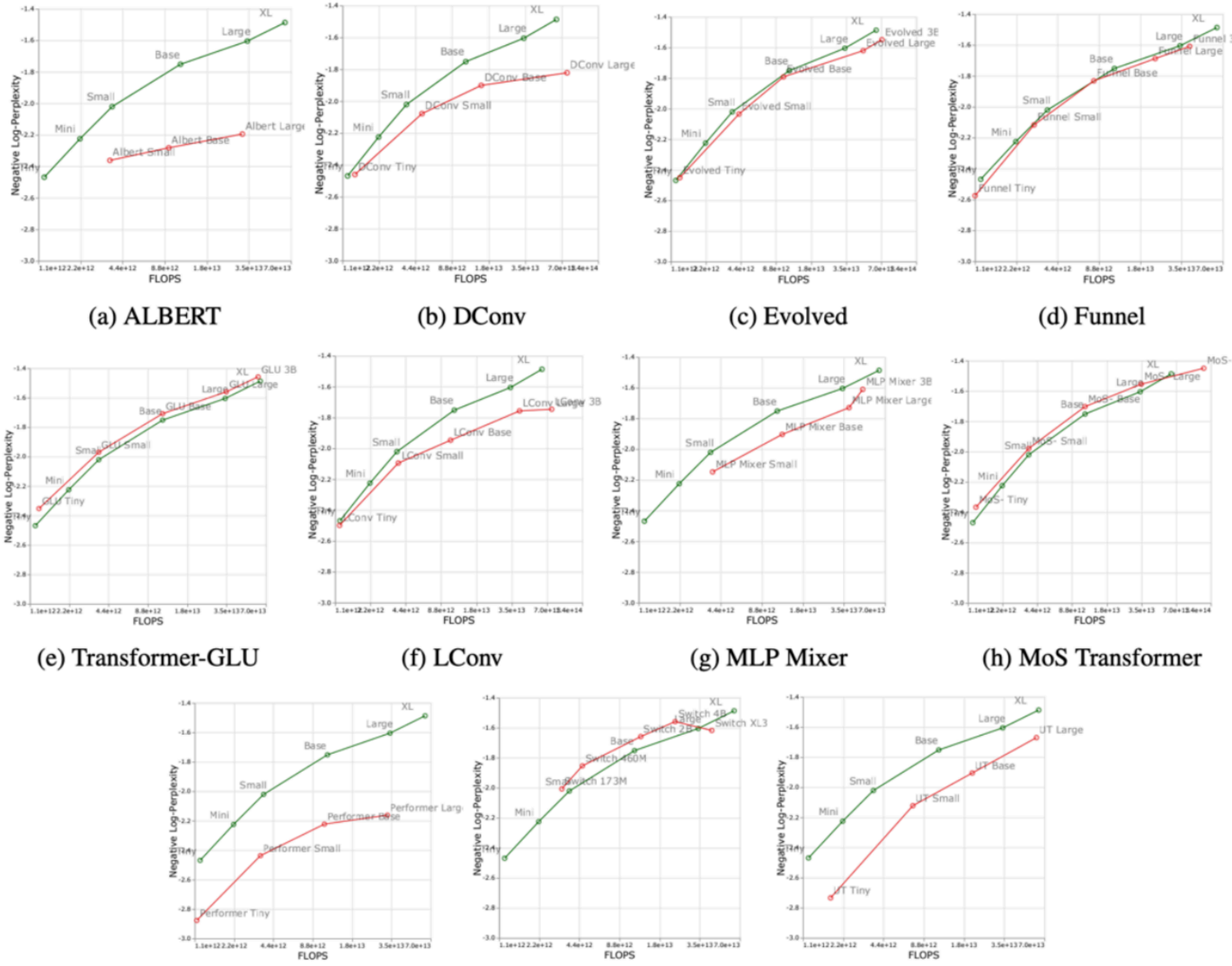
Architecture

1. Many architectures



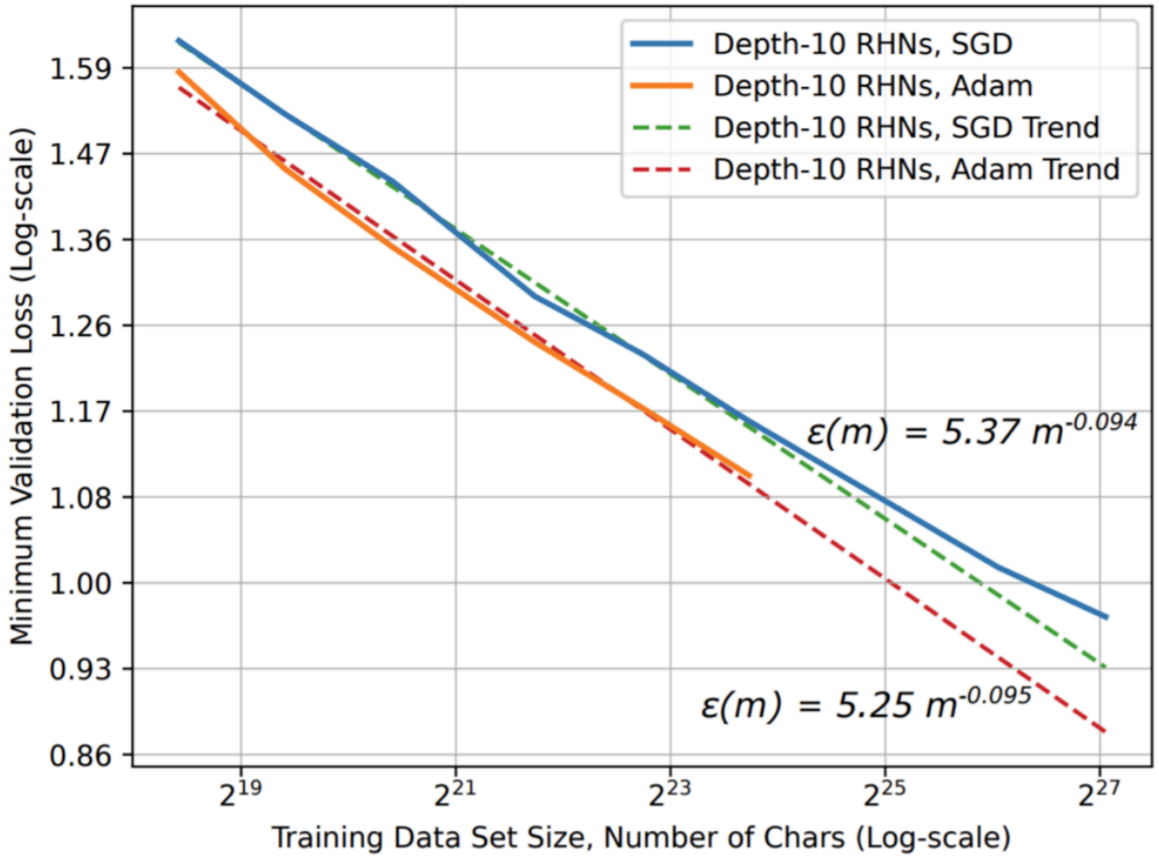
size = num params

Tay et al. (2022)



Optimizer

What about ADAM vs SGD?

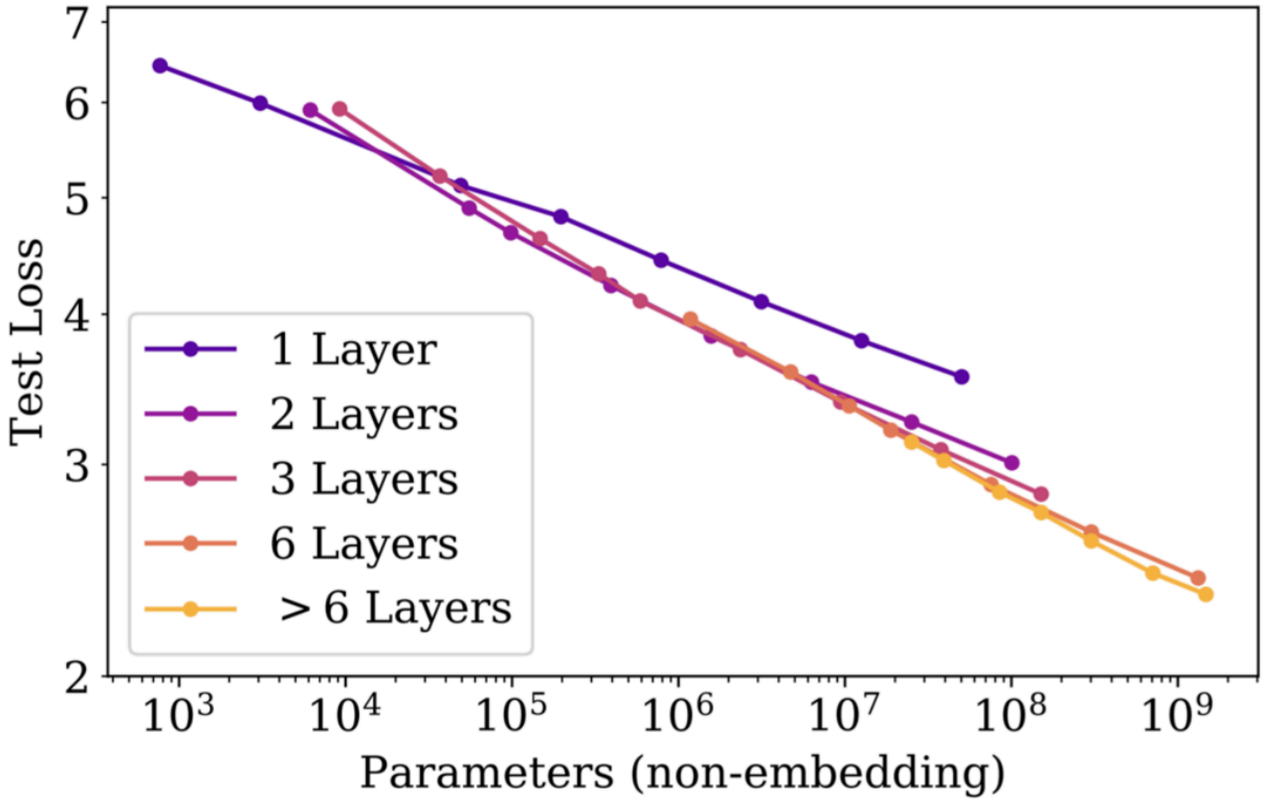


[Hestness+ 2017]

(Note, this is in 2017, so pre-transformers. RHN is recurrent highway nets)

Depth/Width

Does depth or width make a huge difference?

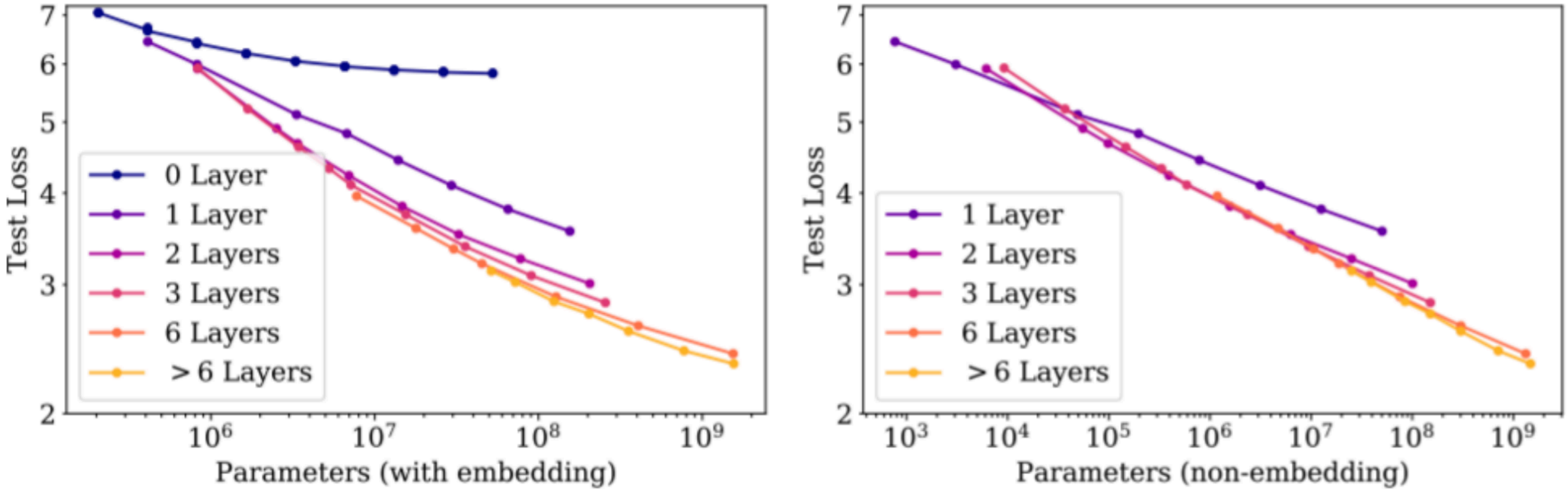


Kaplan et al. (2020)

- 1 vs 2 layers makes a huge difference.
- More layers have diminishing returns below 10^7 params

Depth/Width

We've been thinking about 'parameters' but not all parameters are equal



Embedding layer parameters don't behave the same!

Kaplan et al. (2020)

Related: recent papers on scaling laws for mixtures of experts.

Depth/Width

Do hyperparameters like the aspect ratio depend on scale?

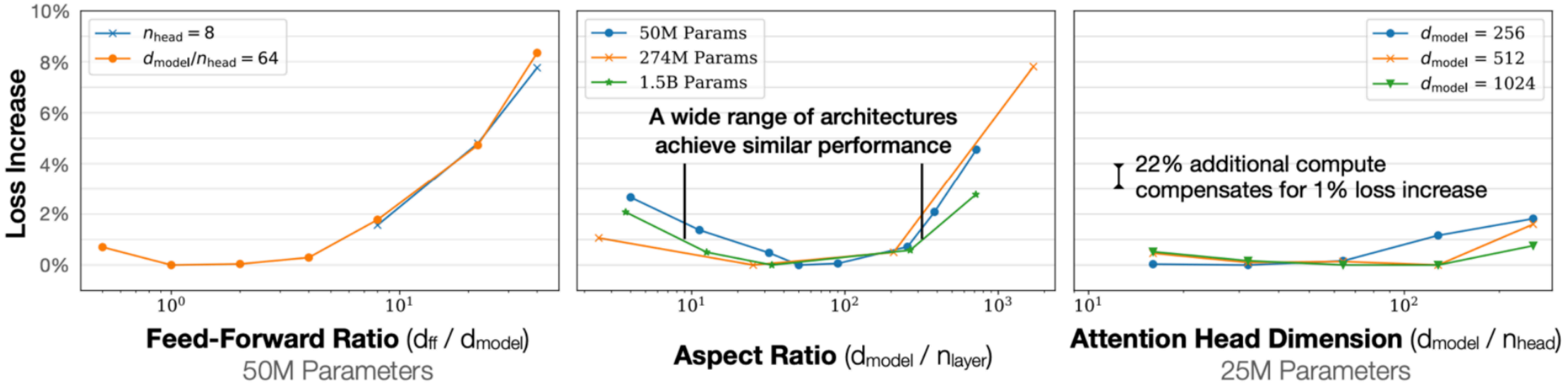
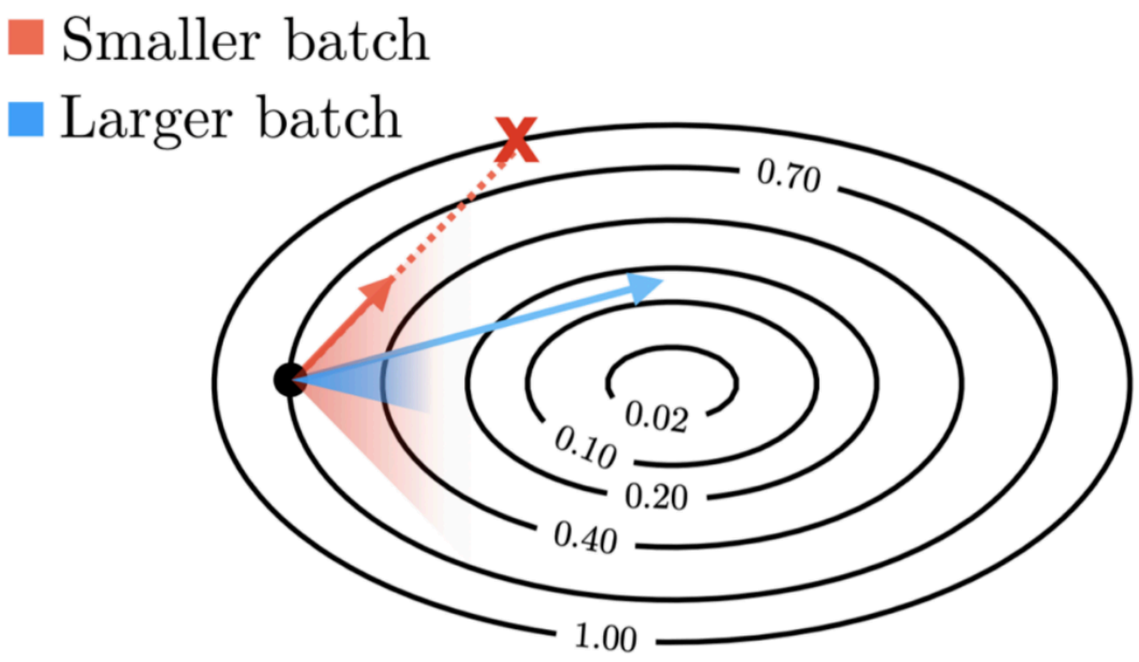
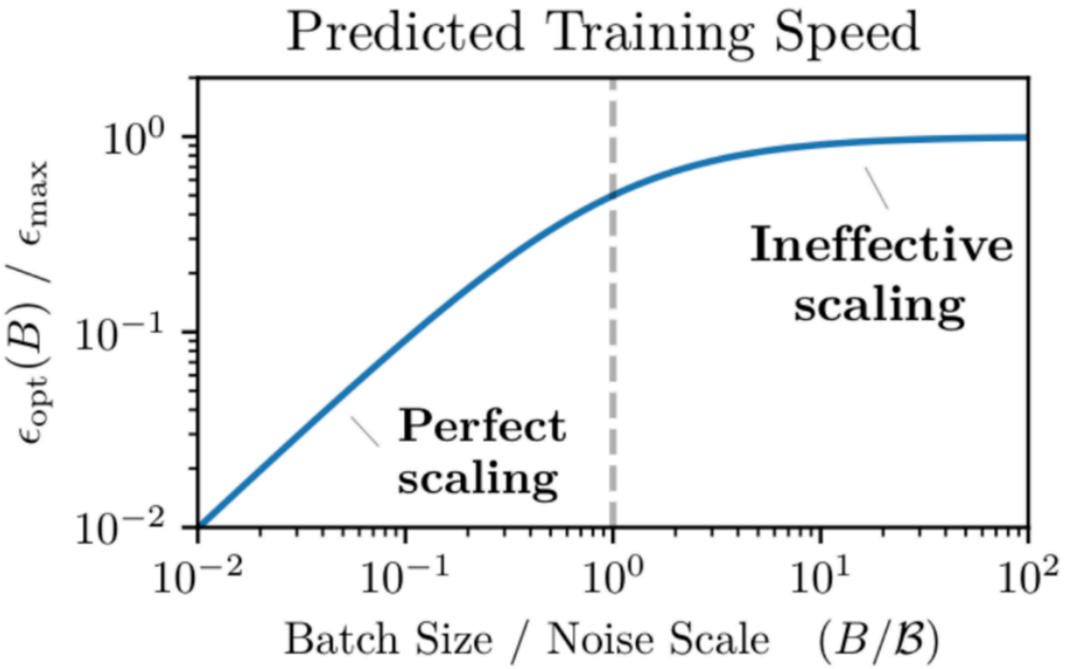


Figure 5 Performance depends very mildly on model shape when the total number of non-embedding parameters N is held fixed. The loss varies only a few percent over a wide range of shapes. Small differences in parameter counts are compensated for by using the fit to $L(N)$ as a baseline. Aspect ratio in 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].

Kaplan et al. (2020)

Batch Size

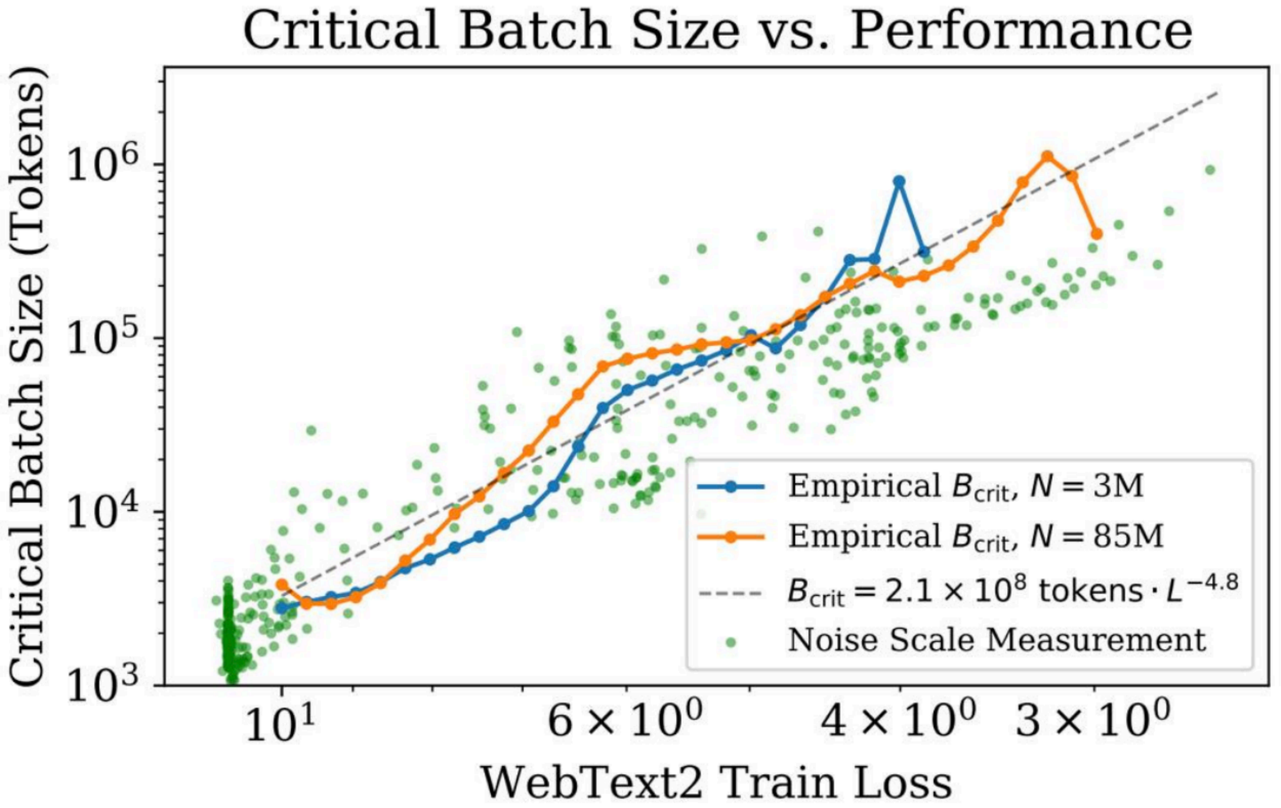




Batch size – known to have strong diminishing returns past a certain point.

Critical batch = min number of examples for target loss / min number of steps for target loss

Batch Size



The smaller the loss target,
The bigger the batch

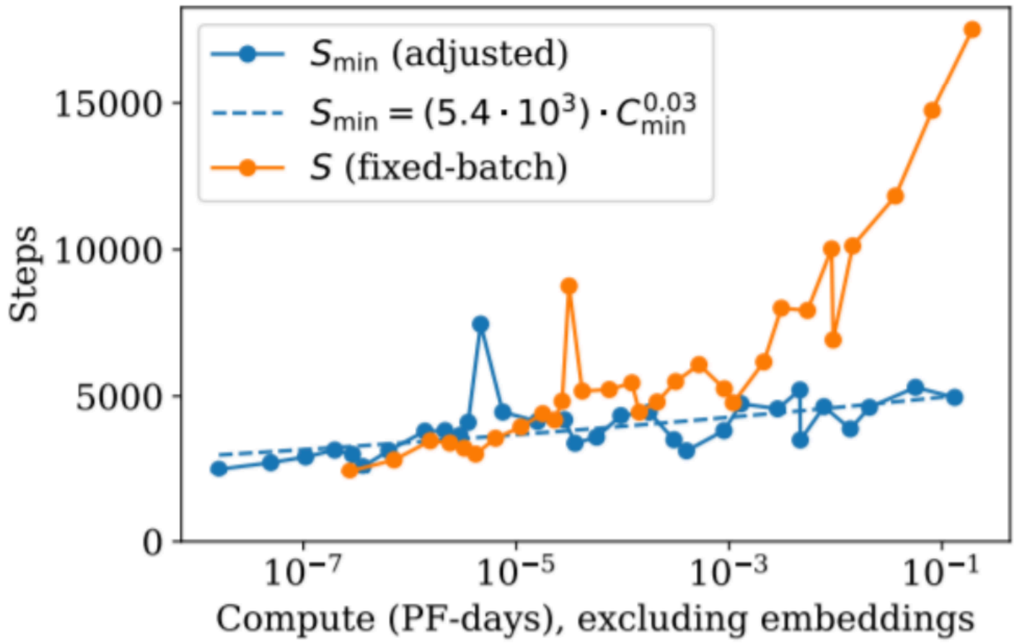
$$C_{\min}(C) \equiv \frac{C}{1 + B/B_{\text{crit}}(L)} \quad \text{(minimum compute, at } B \ll B_{\text{crit}}\text{)}$$

Kaplan et al. (2020)

Batch Size

Q: as we increase both compute and model size, how should we scale training?

- Huge batches, same number of steps
- Fixed batches, more steps



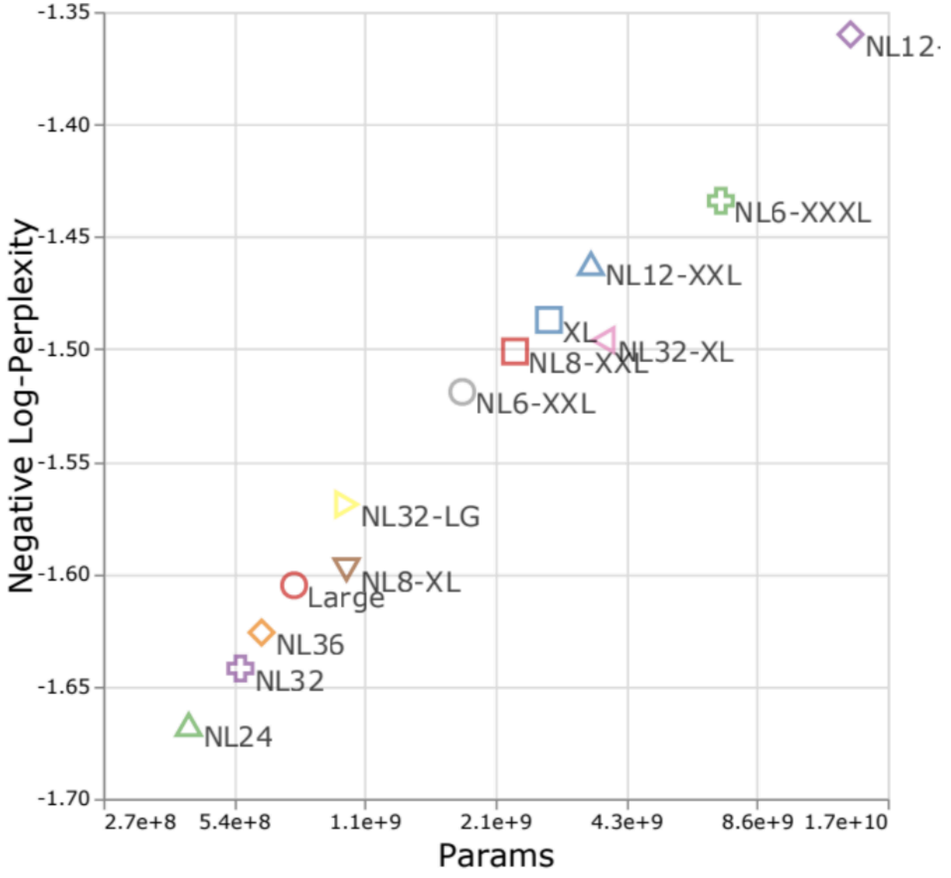
Kaplan et al. (2020)

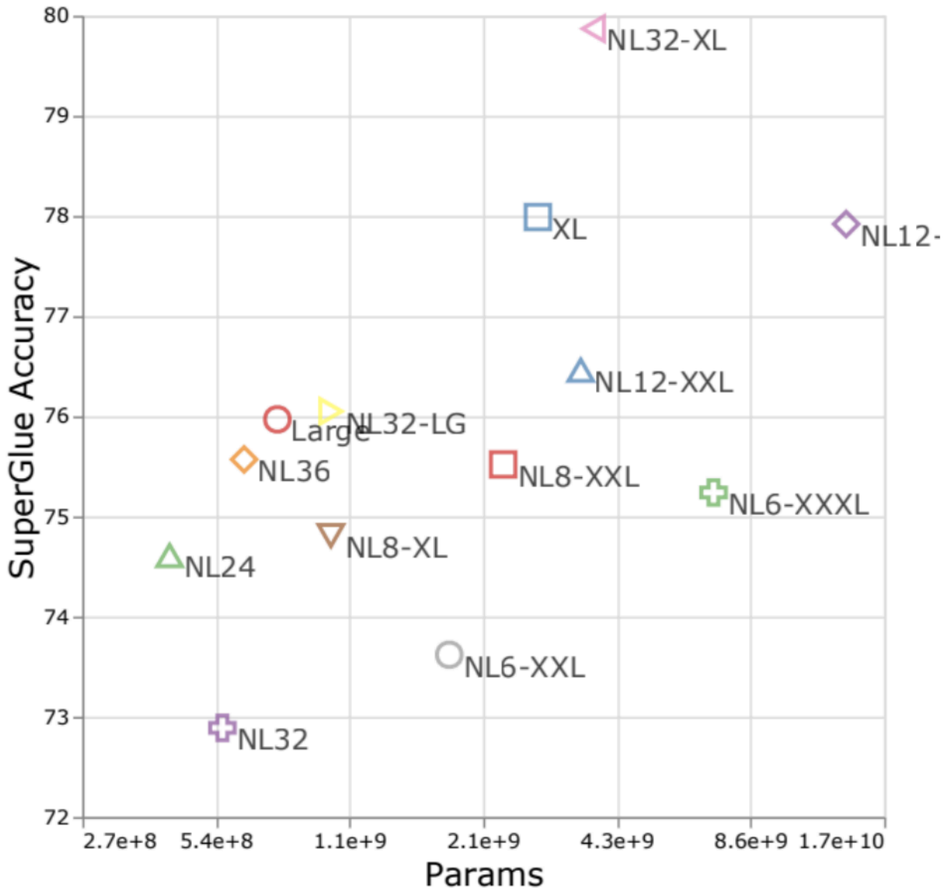
Good news for data parallel processing (?)

Scaling: Downstream Eval

Thus far: scaling is predictable and depends mainly on parameters

Catch: downstream scaling can often be much less predictable





Tay et al 2023

Scaling: Downstream Eval

The effect of hyperparameters on big LMs can be predicted *before* training!

- Optimizer choice
- Model depth
- Architecture choice

The scaling law based design procedure.

1. Train a few smaller models
2. Establish a scaling law (e.g. ADAM vs SGD scaling law)
3. Select optimal hyperparam based on the scaling law prediction.

Administrative details and recap

Hyperparameter Optimization

Scaling Law History

Data Scaling Laws

Model Scaling Laws

Chinchilla: Data x Model

Putting it together

Chinchilla: Data x Model

Slide credit: Tatsu Hashimoto

Model x Data Scaling

Joint data-model scaling laws describe how the two relate

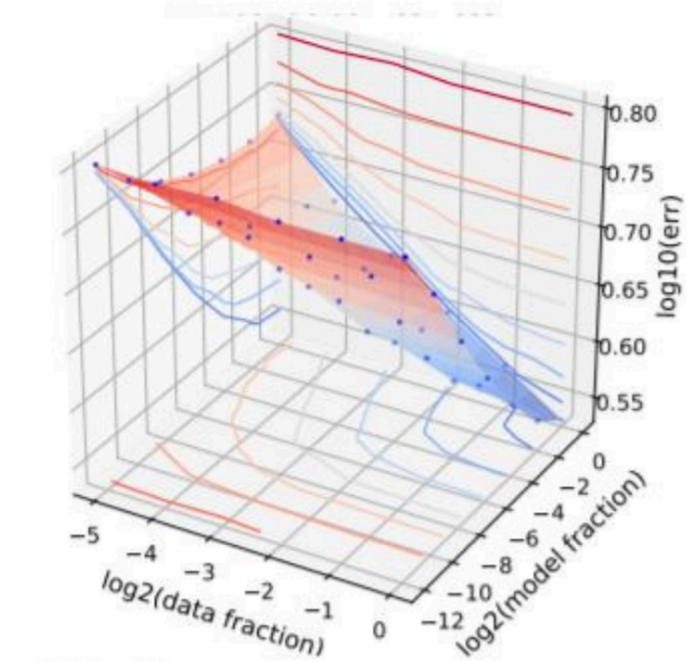
From Rosenfeld+ 2020,

$$Error = n^{-\alpha} + m^{-\beta} + C$$

From Kaplan+ 2020

$$Error = [m^{-\alpha} + n^{-1}]^{\beta}$$

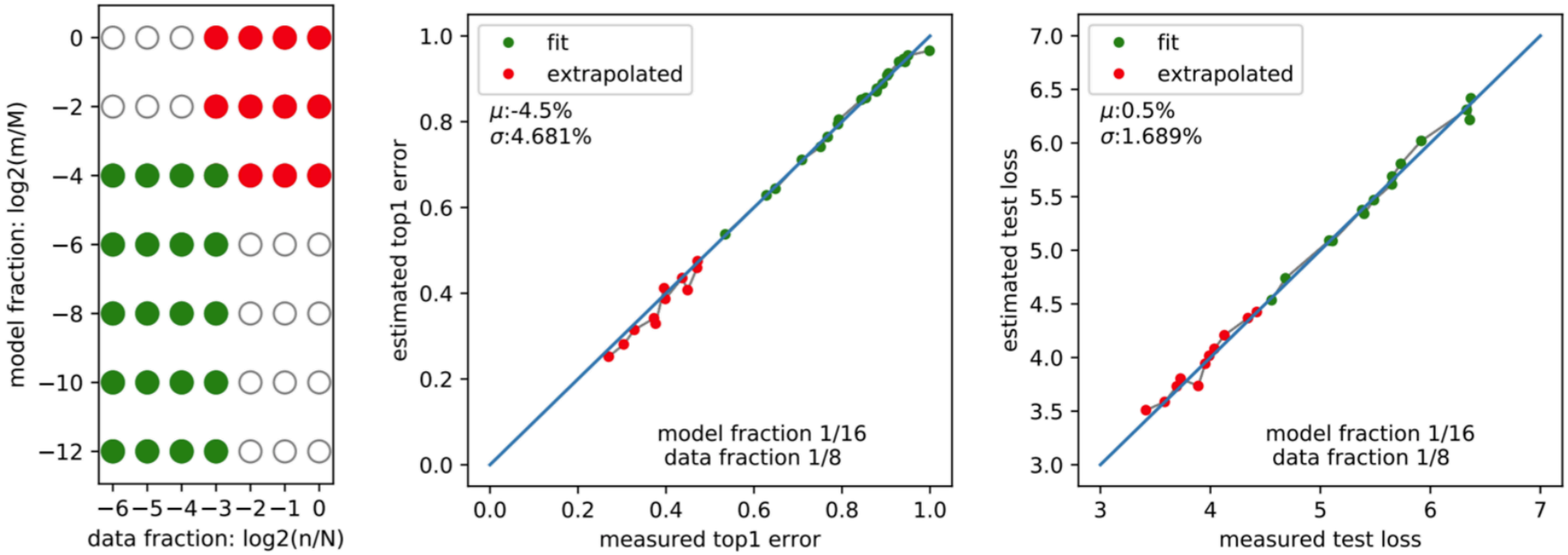
Provides surprisingly good fits to model-data joint error.



(a) Wiki103 error (cross entropy) landscape.

Model x Data Scaling

From Rosenfeld – fit scaling exponents on small data, small models. Predict rest.



(a) Illustration. (b) Extrapolation on ImageNet (c) Extrapolation on WikiText-103.

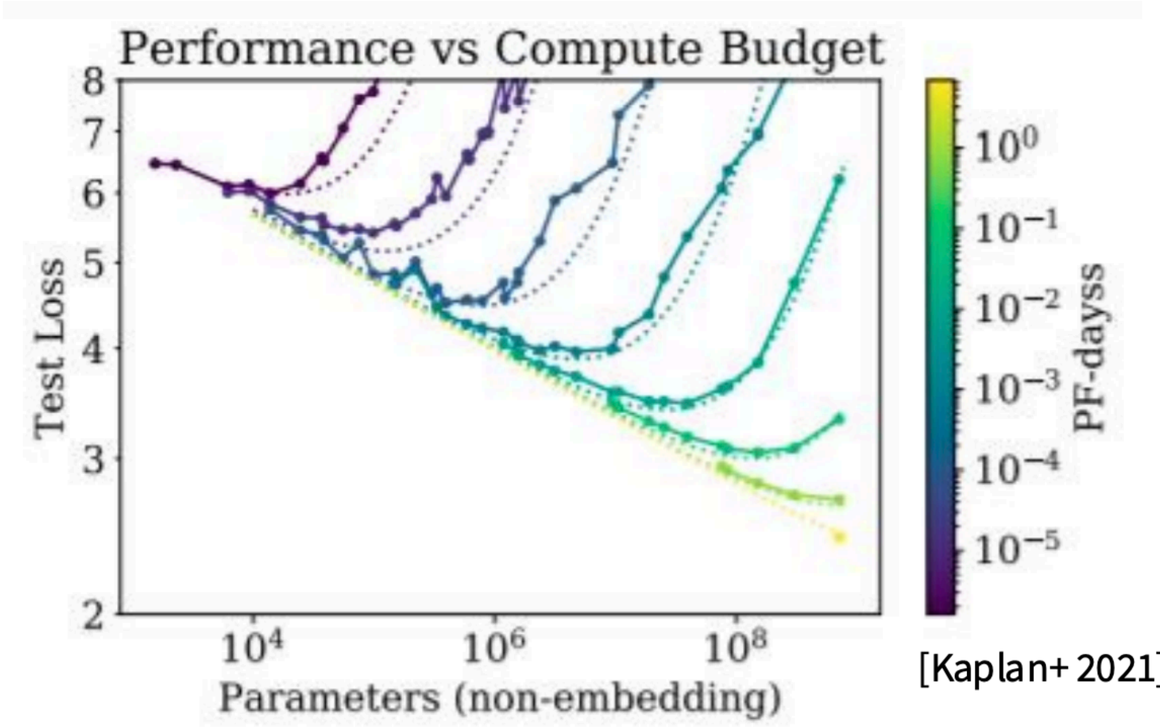
Trading off data size and model size: optimize $n^{-\alpha}+m^{-\beta}+C$ with your costs.

Compute Tradeoffs

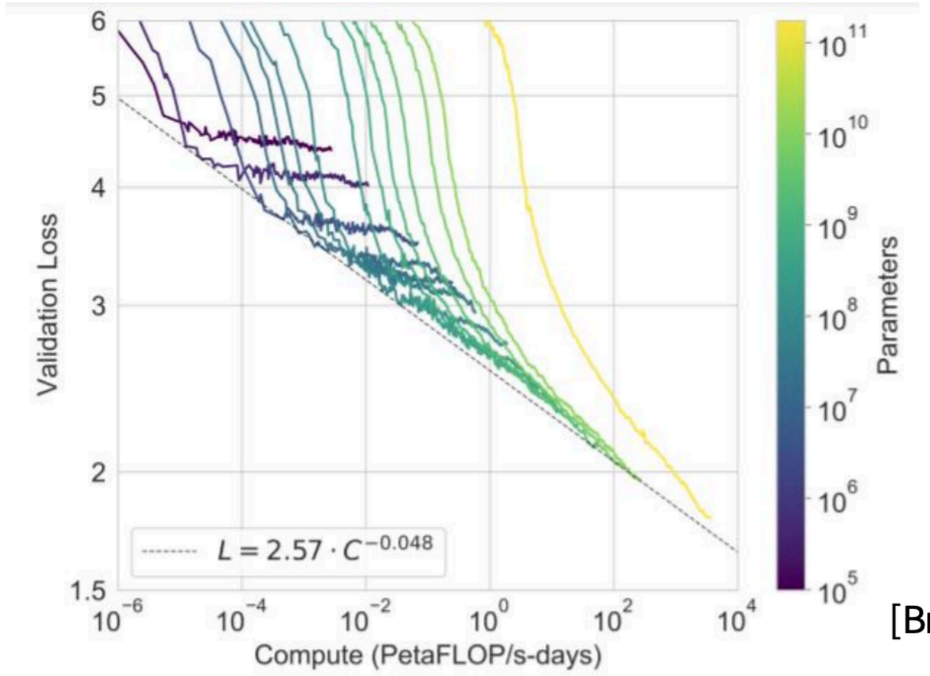
Q: what about other resources? Compute vs performance?

For a fixed compute budget…

Big model that's undertrained vs small model that's well trained?



[Kaplan+ 2021]



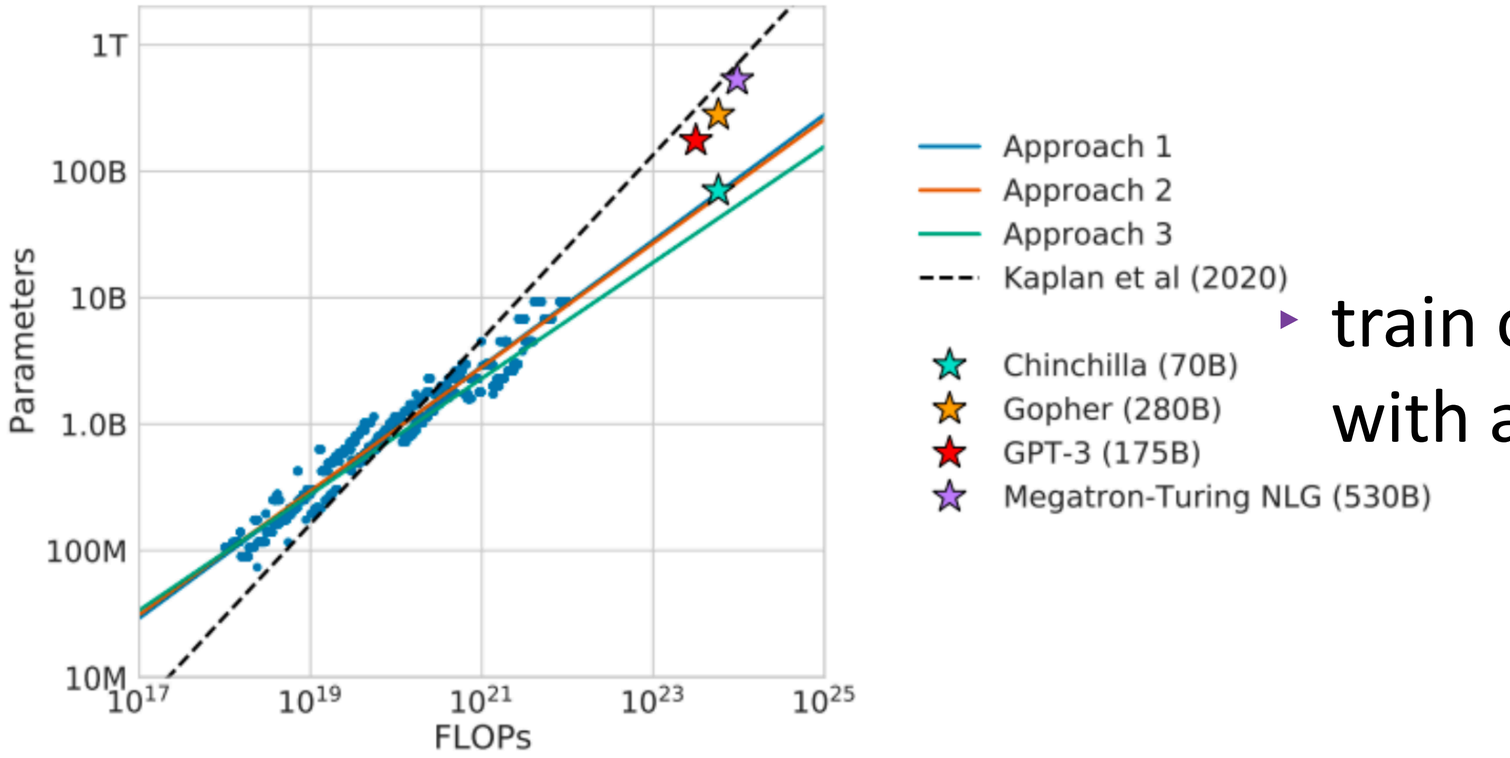
[Brown+ 2020]

Scaling laws let us navigate this tradeoff

Compute Tradeoffs

Rosenfeld, Kaplan both predict relationship of data, model and perf.

Chinchilla [Hoffman et al] argue these fits are quite off.



- train on more data with a smaller model

Hoffman+ 2022

Compute Tradeoffs

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

The chinchilla authors suggest 3 ways of fitting scaling laws – we'll go over each.

They mostly (minus method 3) suggest similar constants. More on this later..

Method 1: Minimum over Runs

Similar to the FLOPS figure on Kaplan –
the minimum over the union of all training curves is a power law.

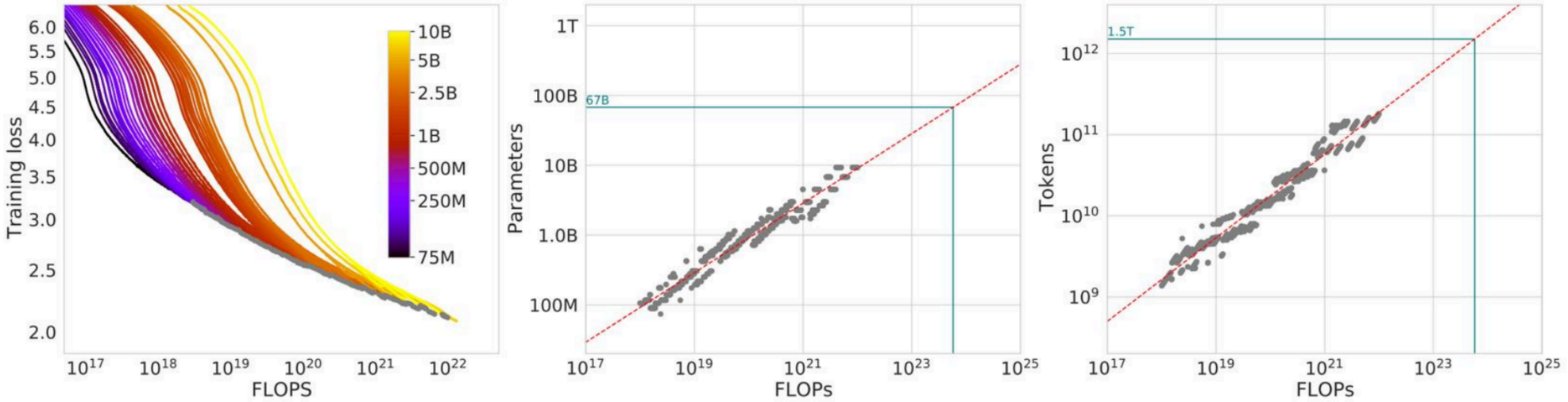


Figure 2 | **Training curve envelope.** On the **left** we show all of our different runs. We launched a range of model sizes going from 70M to 10B, each for four different cosine cycle lengths. From these curves, we extracted the envelope of minimal loss per FLOP, and we used these points to estimate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

Method 2: IsoFLOPs

Pick a range of FLOP budgets, vary the total parameter count, take the min over these convex shapes. The minima form a power law.

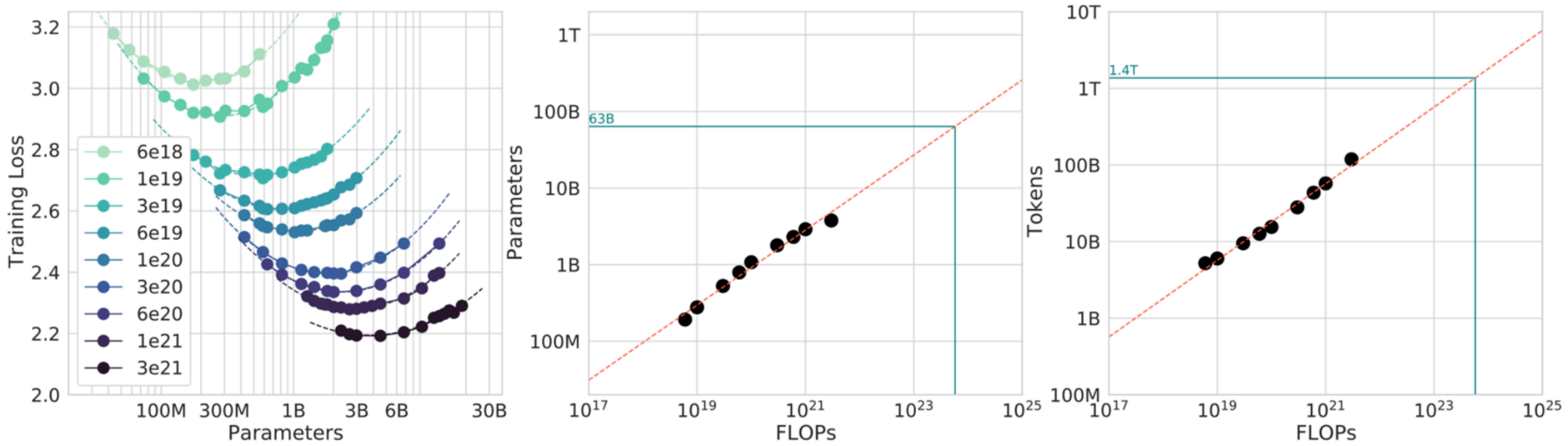


Figure 3 | **IsoFLOP curves.** For various model sizes, we choose the number of training tokens such that the final FLOPs is a constant. The cosine cycle length is set to match the target FLOP count. We find a clear valley in loss, meaning that for a given FLOP budget there is an optimal model to train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

Method 3: Joint Fits

Run a bunch of models on the size-data grid. Use least squares to fit a joint scaling law

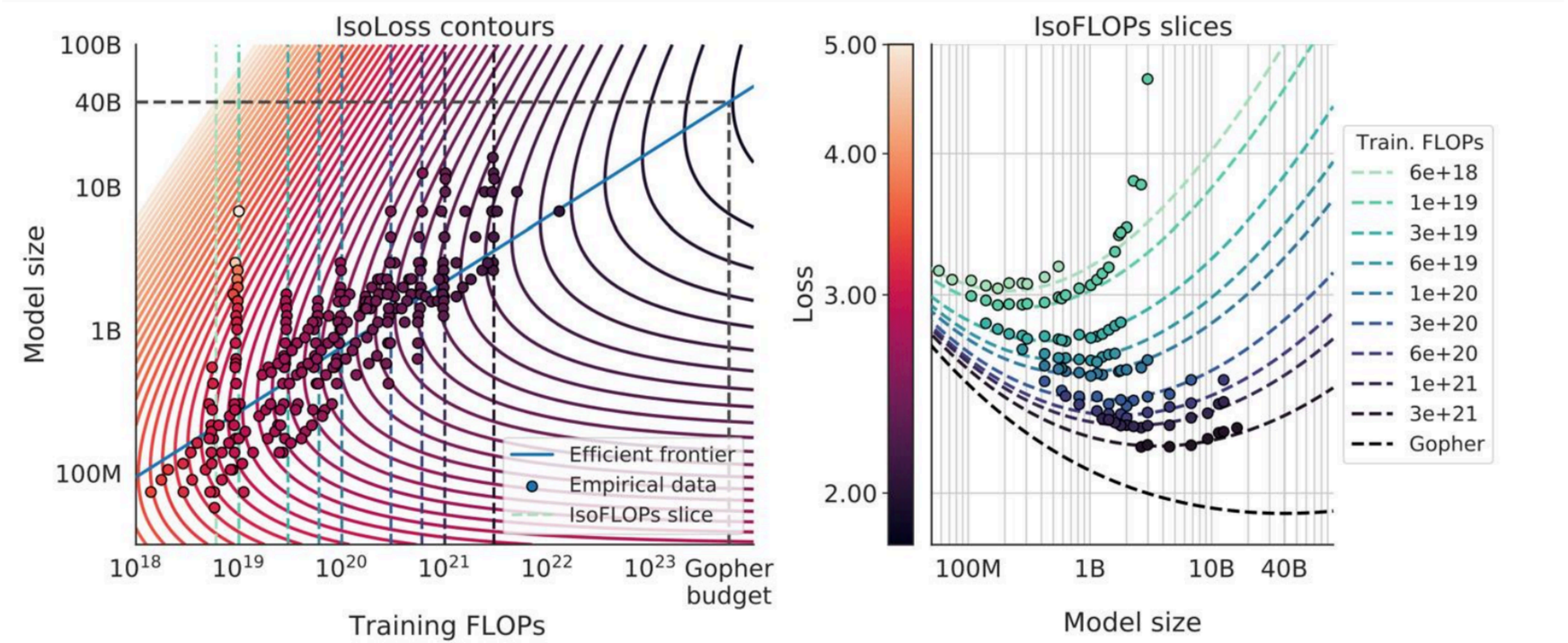


Figure 4 | **Parametric fit.** We fit a parametric modelling of the loss $\hat{L}(N, D)$ and display contour (**left**) and isoFLOP slices (**right**). For each isoFLOP slice, we include a corresponding dashed line in the left plot. In the left plot, we show the efficient frontier in blue, which is a 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.

Overtraining

Chinchilla aims to tell you what gives the best model for fixed training compute..

But most of the compute in a real deployment is inference.. So we should ‘over’ train

- **GPT3** – 2 tokens / param
- **Chinchilla** – 20 tokens / param
- **LLaMA65B** – 22 tokens / param
- **Llama 2 70B** – 29 tokens / param
- **Mistral 7B** – 110 tokens / param
- **Llama 3 70B** – 215 tokens / param

The more usage we expect, the more it becomes worth it to pay the upfront cost

Administrative details and recap

Hyperparameter Optimization

Scaling Law History

Data Scaling Laws

Model Scaling Laws

Chinchilla: Data x Model

Putting it together

Putting it together: DCLM and OLMo 3

Slide credit: Tatsu Hashimoto

DataComp Language Modeling

DataComp-LM: In search of the next generation of training sets for language models

Jeffrey Li*[1,2] Alex Fang*[1,2] Georgios Smyrnis*[4] Maor Ivgi*[5]
Matt Jordan[4] Samir Gadre[3,6] Hritik Bansal[8] Etash Guha[1,15] Sedrick Keh[3] Kushal Arora[3]
Saurabh Garg[13] Rui Xin[1] Niklas Muennighoff[22] Reinhard Heckel[12] Jean Mercat[3]
Mayee Chen[7] Suchin Gururangan[1] Mitchell Wortsman[1] Alon Albalak[19,20] Yonatan Bitton[14]
Marianna Nezhurina[9,10] Amro Abbas[23] Cheng-Yu Hsieh[1] Dhruba Ghosh[1] Josh Gardner[1]
Maciej Kilian[17] Hanlin Zhang[18] Rulin Shao[1] Sarah Pratt[1] Sunny Sanyal[4] Gabriel Ilharco[1]
Giannis Daras[4] Kalyani Marathe[1] Aaron Gokaslan[16] Jieyu Zhang[1] Khyathi Chandu[11]
Thao Nguyen[1] Igor Vasiljevic[3] Sham Kakade[18] Shuran Song[6,7] Sujay Sanghavi[4]
Fartash Faghri[2] Sewoong Oh[1] Luke Zettlemoyer[1] Kyle Lo[11] Alaaeldin El-Nouby[2]
Hadi Pouransari[2] Alexander Toshev[2] Stephanie Wang[1] Dirk Groeneveld[11] Luca Soldaini[11]
Pang Wei Koh[1] Jenia Jitsev[9,10] Thomas Kollar[3] Alexandros G. Dimakis[4,21]
Yair Carmon[5] Achal Dave[†3] Ludwig Schmidt[†1,7] Vaishaal Shankar[†2]

[1]University of Washington, [2]Apple, [3]Toyota Research Institute, [4]UT Austin,
[5]Tel Aviv University, [6]Columbia University, [7]Stanford, [8]UCLA, [9]JSC, [10]LAION, [11]AI2,
[12]TUM, [13]CMU, [14]Hebrew University, [15]SambaNova, [16]Cornell, [17]USC, [18]Harvard,
[19]UCSB, [20]SynthLabs, [21]Bespokelabs.AI, [22]Contextual AI, [23]DatologyAI

DataComp Language Modeling

Followed on earlier “DataComp” projects for language modeling

Fixed dataset to allow for testing of algorithms with data held constant

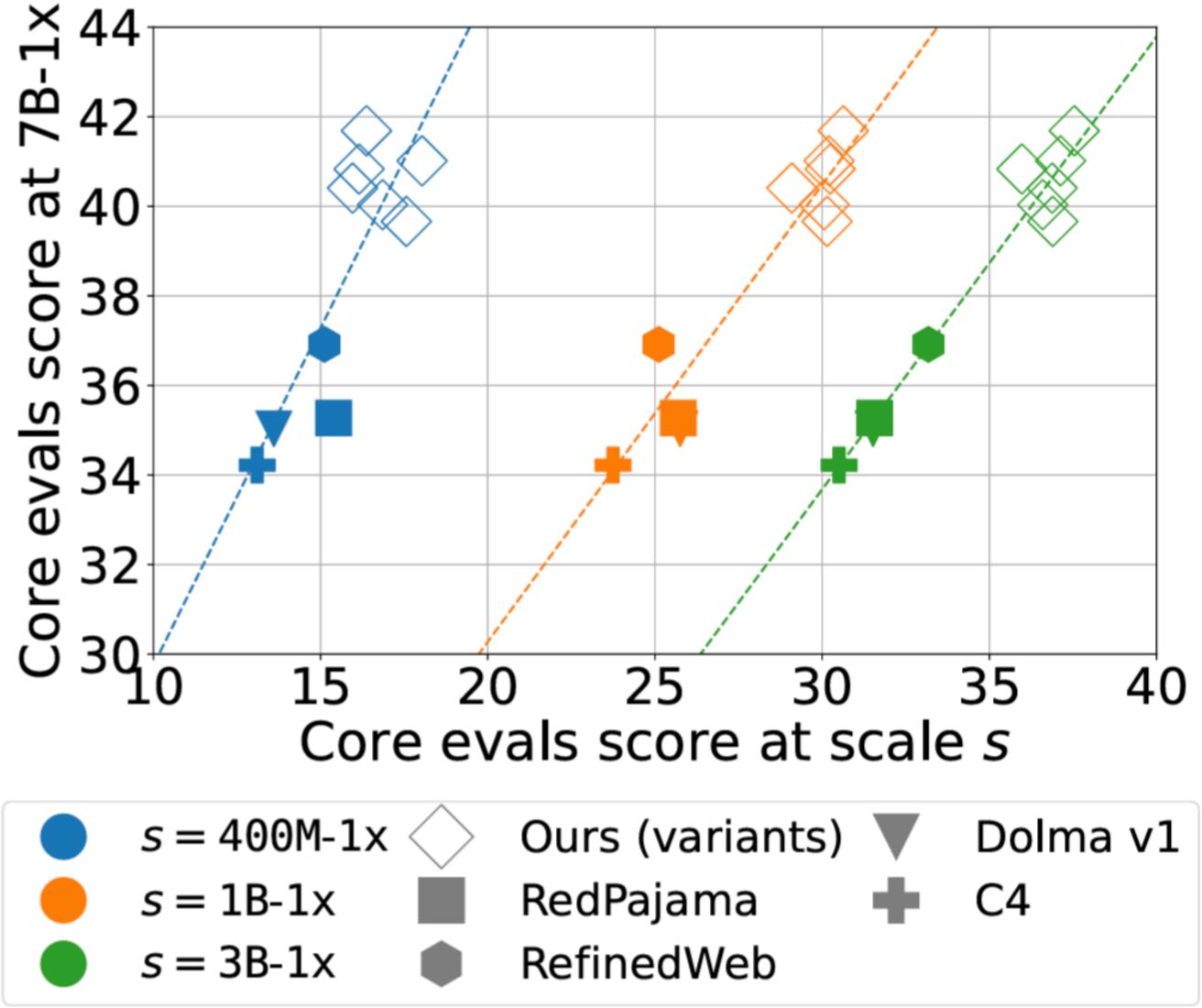
Scale	Model parameters	Train tokens	Train FLOPs	Train H100 hours	Pool size
400M-1x	412M	8.2B	2.0e19	26	469B
1B-1x	1.4B	28.8B	2.4e20	240	1.64T
3B-1x	2.8B	55.9B	9.4e20	740	3.18T
7B-1x	6.9B	138B	5.7e21	3,700	7.85T
7B-2x	6.9B	276B	1.1e22	7,300	15.7T

…but what people ended up using was actually just the dataset they made

1x = number of repeats of the dataset

Predictability

Evals at small scale predict scores at higher scale (positive correlation)



Data Filtering

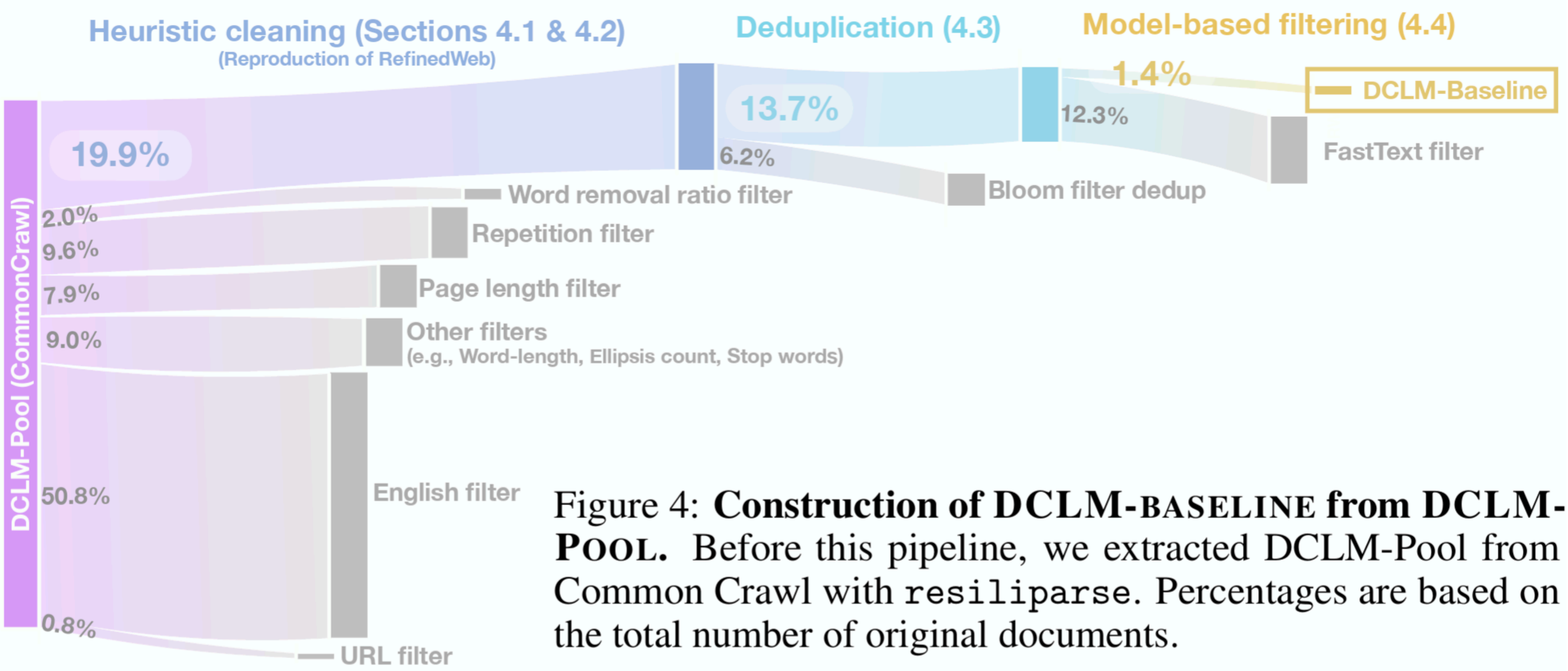
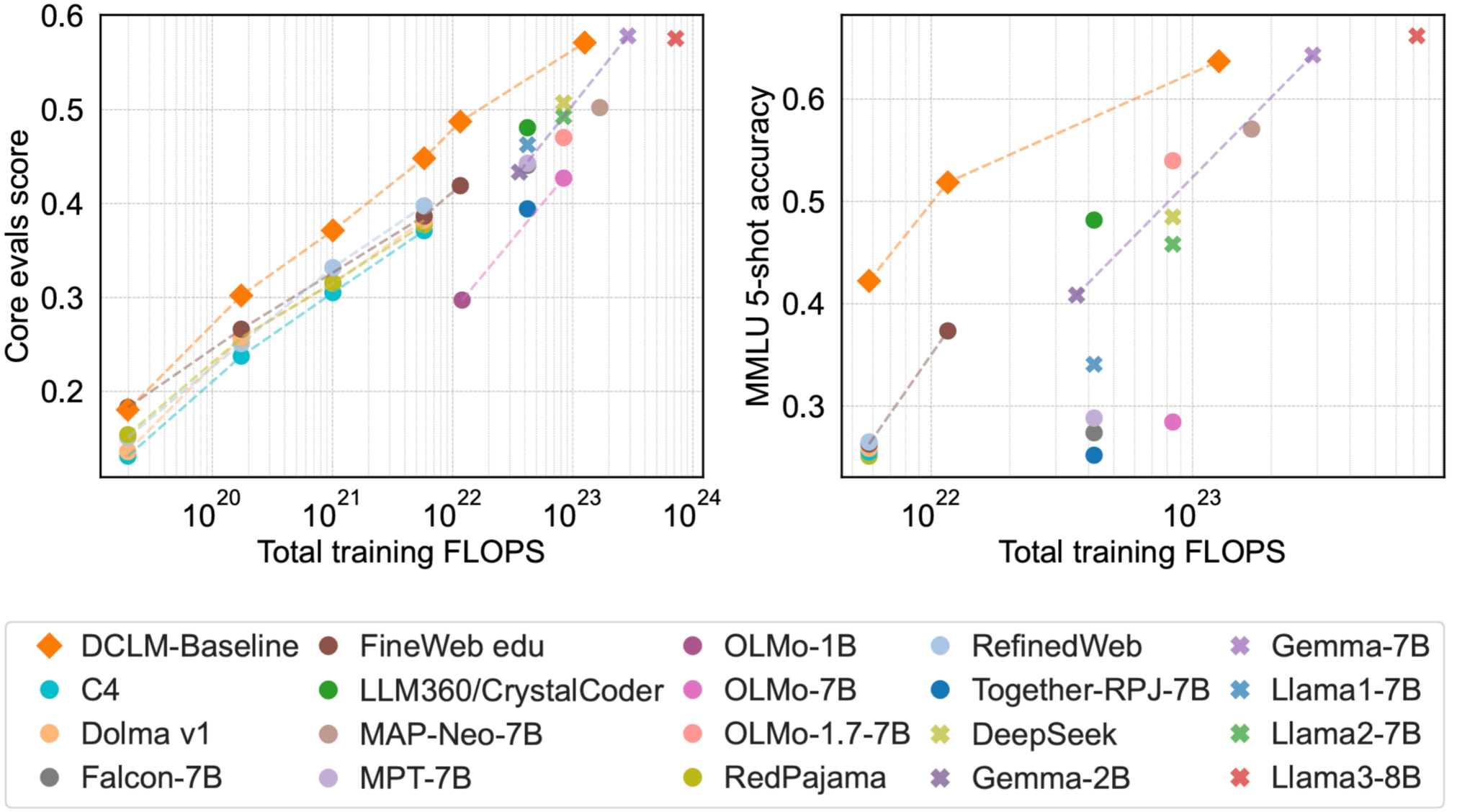


Figure 4: **Construction of DCLM-BASELINE from DCLM-POOL.** Before this pipeline, we extracted DCLM-Pool from Common Crawl with `resiliparse`. Percentages are based on the total number of original documents.

Data Filtering



Olmo 3

Olmo Team[★]

Allyson Ettinger[♥1] Amanda Bertsch[♥1,3] Bailey Kuehl[♥1] David Graham[♥1]
David Heineman[♥1] Dirk Groeneveld[♥1] Faeze Brahman[♥1] Finbarr Timbers[♥1]
Hamish Ivison[♥1,2] Jacob Morrison[♥1,2] Jake Poznanski[♥1] Kyle Lo[♥1,2] Luca Soldaini[♥1]
Matt Jordan[♥1] Mayee Chen[♥1,4] Michael Noukhovitch[♥1,5,6] Nathan Lambert[♥1]
Pete Walsh[♥1] Pradeep Dasigi[♥1] Robert Berry[♥1] Saumya Malik[♥1] Saurabh Shah[♥1]
Scott Geng[♥1,2] Shane Arora[♥1] Shashank Gupta[♥1] Taira Anderson[♥1] Teng Xiao[♥1]
Tyler Murray[♥1] Tyler Romero[♥1] Victoria Graf[♥1,2]

Akari Asai[1,3] Akshita Bhagia[1] Alexander Wettig[7] Alisa Liu[2] Aman Rangapur[1]
Chloe Anastasiades[1] Costa Huang[1] Dustin Schwenk[1] Harsh Trivedi[1] Ian Magnusson[1,2]
Jaron Lochner[1] Jiacheng Liu[1] Lester James V. Miranda[1] Maarten Sap[1,3] Malia Morgan[1]
Michael Schmitz[1] Michal Guerquin[1] Michael Wilson[1] Regan Huff[1] Ronan Le Bras[1]
Rui Xin[2] Rulin Shao[2] Sam Skjonsberg[1] Shannon Zejiang Shen[8] Shuyue Stella Li[2]
Tucker Wilde[1] Valentina Pyatkin[1] Will Merrill[1] Yapei Chang[9] Yuling Gu[1] Zhiyuan Zeng[1,2]

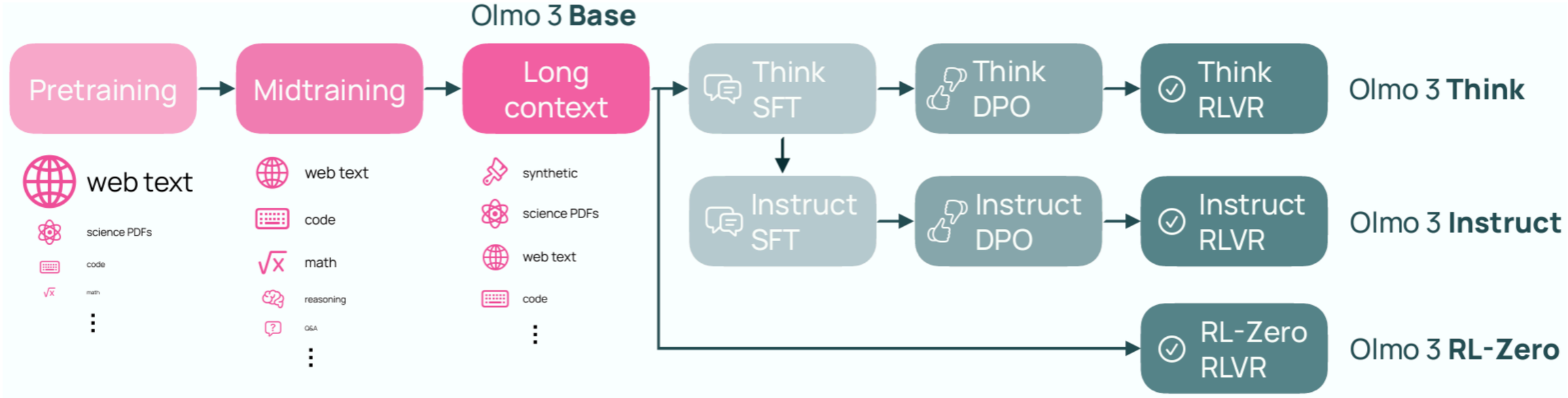
Ashish Sabharwal[1] Luke Zettlemoyer[2] Pang Wei Koh[1,2]
Ali Farhadi[1,2] Noah A. Smith[♥1,2] Hannaneh Hajishirzi[♥1,2]

[1]Allen Institute for AI [2]University of Washington [3]Carnegie Mellon University [4]Stanford University [5]Mila
[6]Université de Montréal [7]Princeton University [8]Massachusetts Institute of Technology [9]University of Maryland

OLMo 3

Along with DCLM, one of the few “fully open” language models: open weights, open data, ability to reproduce given the compute.
(Note: Llama/Qwen/etc. lack open data.)

Several models; we will cover what all this means more with post-training:



OLMo 3

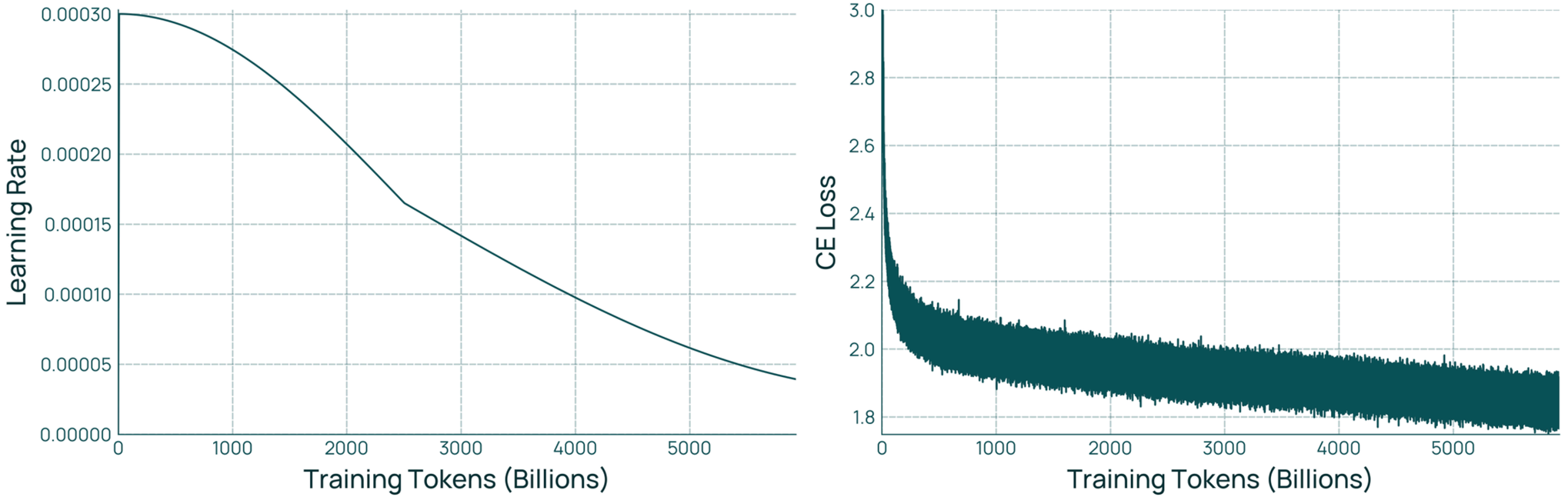


Figure 3 Learning rate schedule and loss for Olmo 3 Base 7B. The first half of the learning rate schedule is a cosine schedule over 5T tokens. We stretch the second half of the schedule to reach a target length of one epoch (5.93T tokens). Warm-up is 2000 steps, the peak learning rate is 3×10^{-4}, and the final learning rate is 10% of the peak LR.

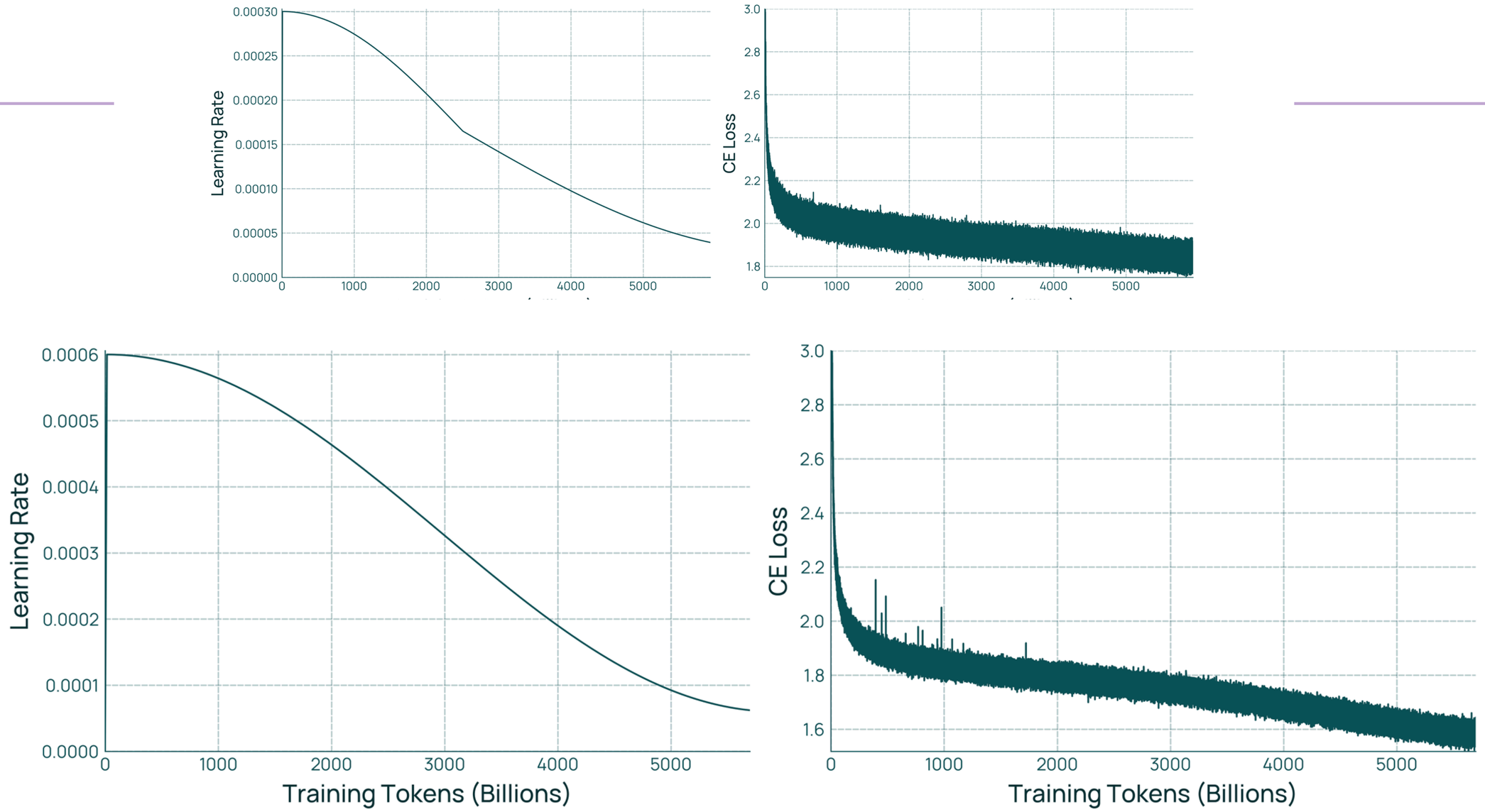
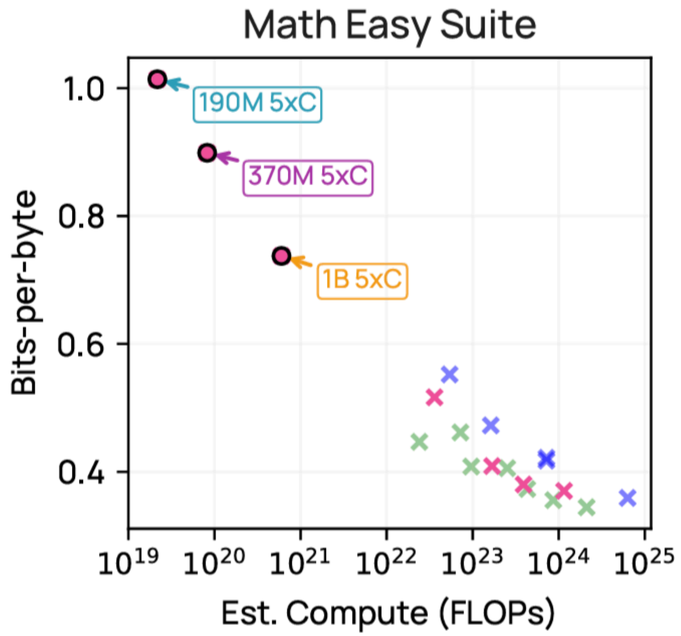


Figure 4 Learning rate schedule and loss for Olmo 3 Base 32B. The learning rate schedule is a cosine schedule over one epoch (5.93T tokens), truncated at 5.5T tokens. Warm-up is 2000 steps, and the peak learning rate is 6×10^{-4}.

Evaluating Scaling

Evaluating base models is hard: we want to make sure they will help with math and coding

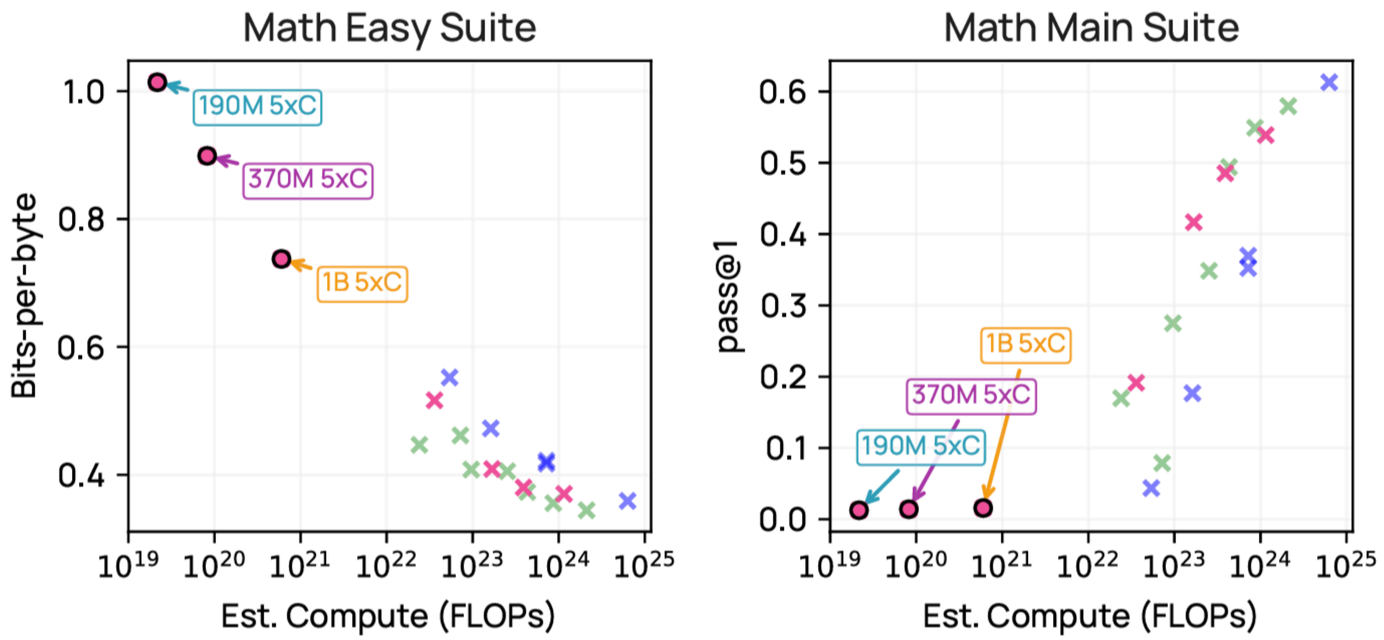
Want to find: metrics on easy data where small models can improve.



Evaluating Scaling

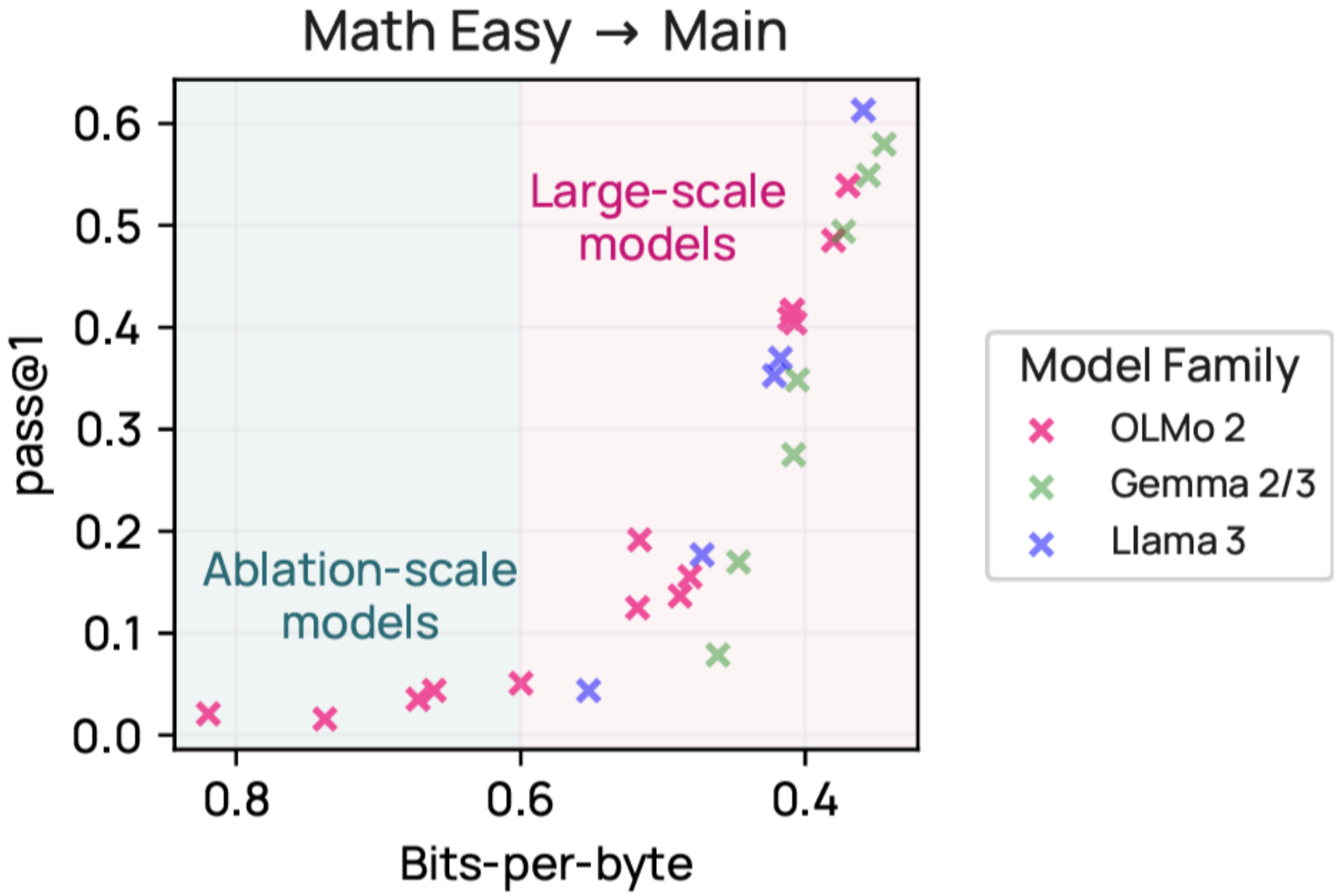
Evaluating base models is hard: we want to make sure they will help with math and coding

Want to find: metrics on easy data where small models can improve.
Then: evidence that those gains transfer to large models.



Evaluating Scaling

"Math Easy" and main eval performance are very correlated!



Administrative details and recap

Hyperparameter Optimization

Scaling Law History

Data Scaling Laws

Model Scaling Laws

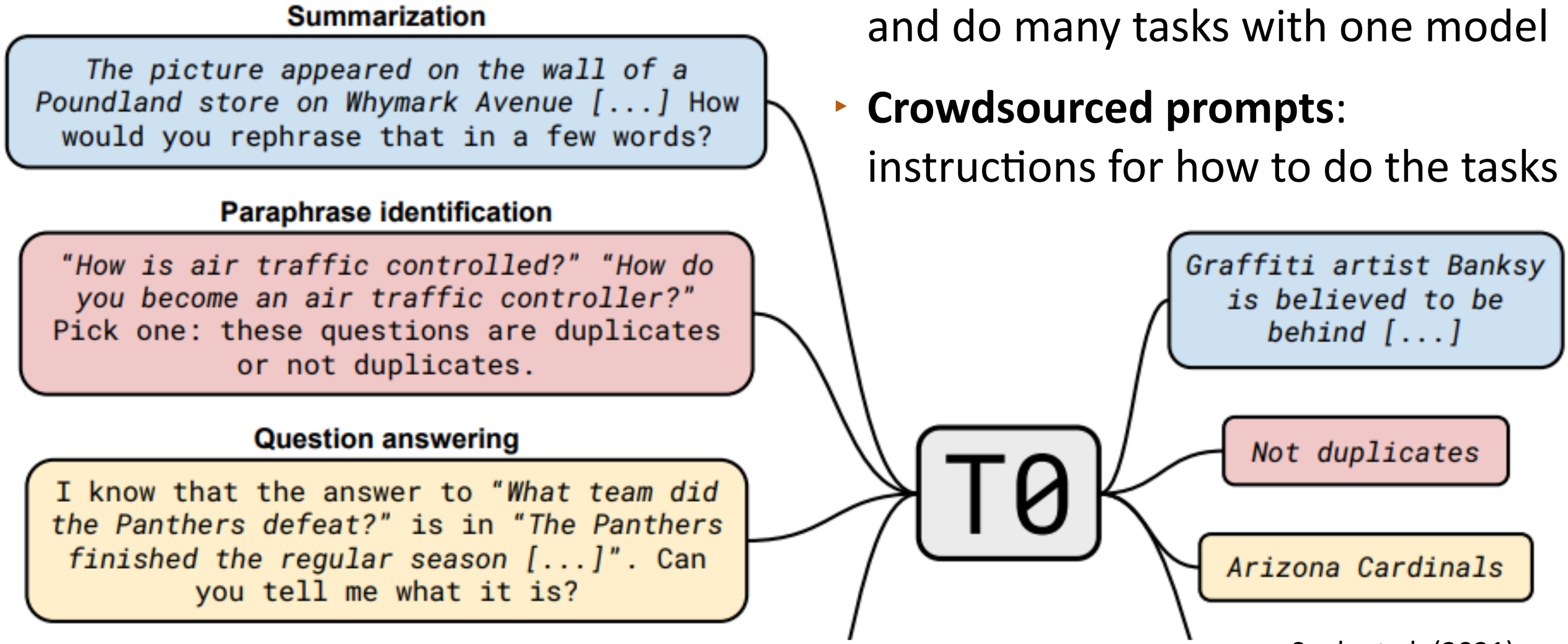
Chinchilla: Data x Model

Putting it together

SFT

Task Generalization: T0

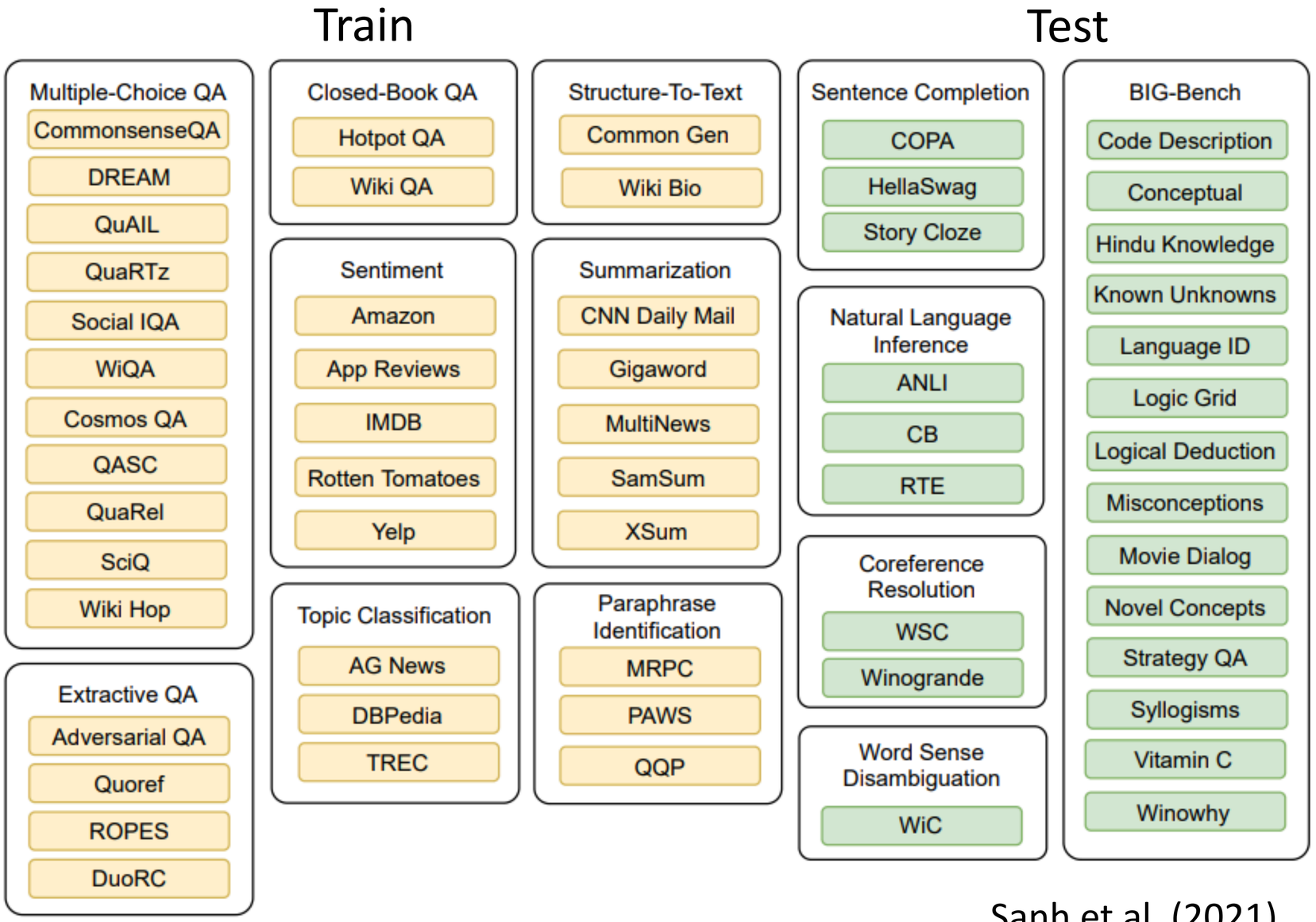
- T0: tries to deliver on the goal of T5 and do many tasks with one model
- **Crowdsourced prompts**: instructions for how to do the tasks



Sanh et al. (2021)

Task Generalization

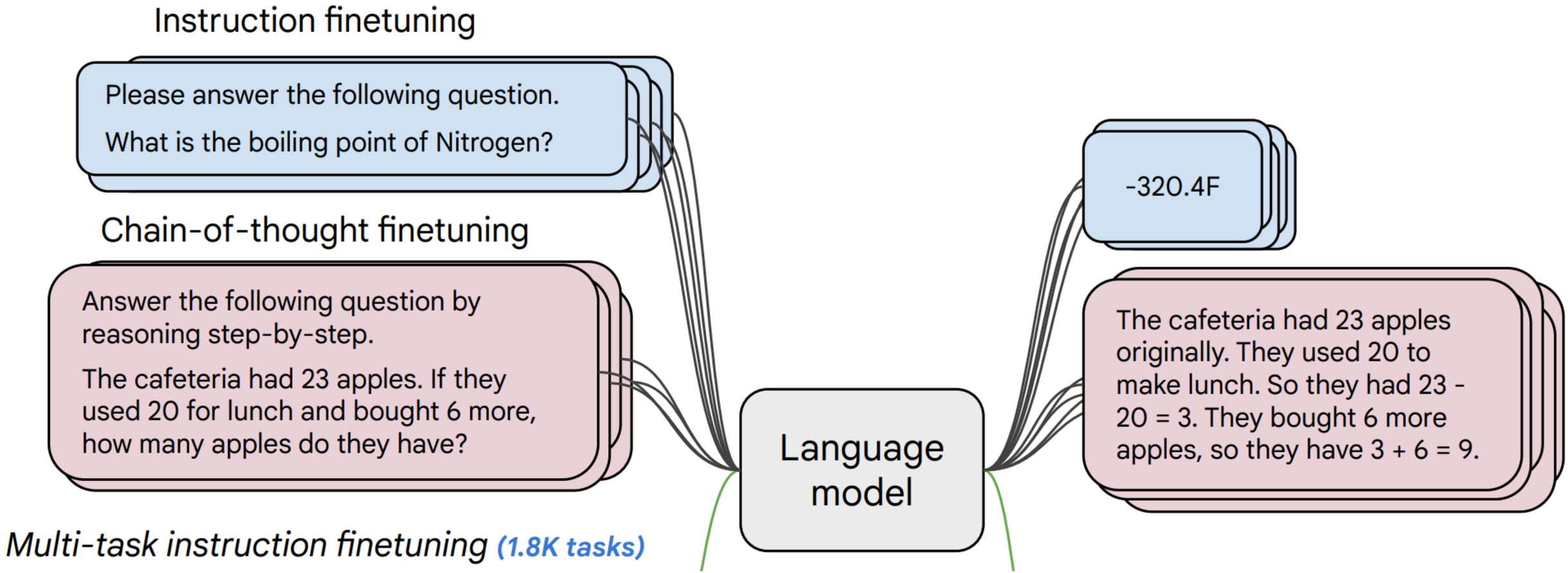
- Pre-train: T5 task
- Train: a collection of tasks with prompts. **This uses existing labeled training data**
- Test: a new task specified only by a new prompt. **No training data in this task**



Sanh et al. (2021)

Flan-PaLM

- Flan-PaLM (October 20, 2022): 1800 tasks, 540B parameter model fine-tuned on many tasks after pre-training



Chung et al. (2022)

Flan-PaLM

- Flan-PaLM (October 20, 2022): 1800 tasks, 540B parameter model
- MMLU task (Hendrycks et al., 2020): 57 high school/college/professional exams:

Conceptual Physics

When you drop a ball from rest it accelerates downward at 9.8 m/s². If you instead throw it downward assuming no air resistance its acceleration immediately after leaving your hand is

(A) 9.8 m/s² ✔
(B) more than 9.8 m/s² ✘
(C) less than 9.8 m/s² ✘
(D) Cannot say unless the speed of throw is given. ✘

College Mathematics

In the complex z-plane, the set of points satisfying the equation $z^2 = |z|^2$ is a

(A) pair of points ✘
(B) circle ✘
(C) half-line ✘
(D) line ✔

Figure 4: Examples from the Conceptual Physics and College Mathematics STEM tasks.

Chung et al. (2022)

Flan-PaLM

- Flan-PaLM (October 20, 2022): 1800 tasks, 540B parameter model
- MMLU task (Hendrycks et al., 2020): 57 high school/college/professional exams:

-	Random	25.0
-	Average human rater	34.5
May 2020	GPT-3 5-shot	43.9
Mar. 2022	Chinchilla 5-shot	67.6
Apr. 2022	PaLM 5-shot	69.3
Oct. 2022	**Flan-PaLM 5-shot**	**72.2**
	Flan-PaLM 5-shot: CoT + SC	**75.2**
-	Average human expert	89.8

Chung et al. (2022)

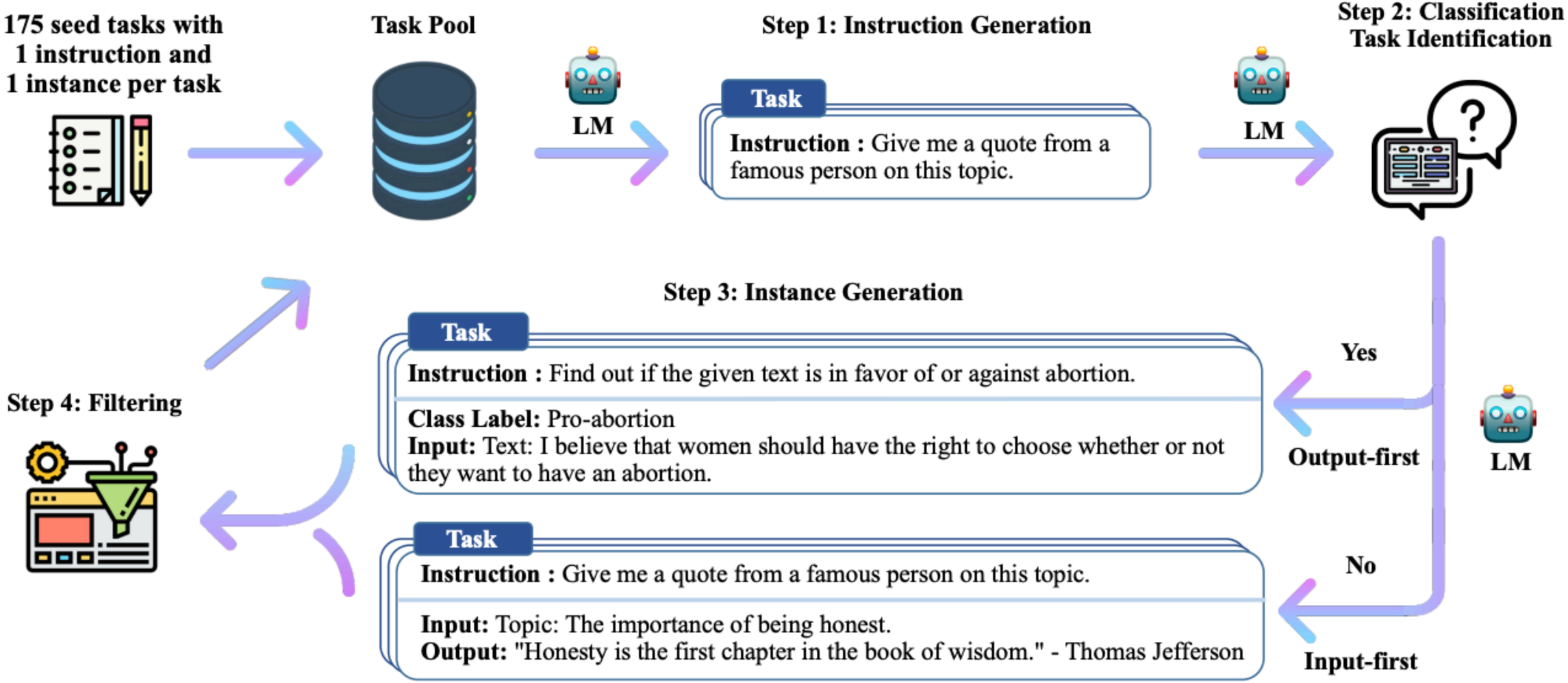
Flan-PaLM

Model	Finetuning Mixtures	Tasks	Norm. avg.	MMLU		BBH	
				Direct	CoT	Direct	CoT
540B	None (no finetuning)	0	49.1	71.3	62.9	49.1	63.7
	CoT	9	52.6 (+3.5)	68.8	64.8	50.5	61.1
	CoT, Muffin	89	57.0 (+7.9)	71.8	66.7	56.7	64.0
	CoT, Muffin, T0-SF	282	57.5 (+8.4)	72.9	**68.2**	57.3	64.0
	CoT, Muffin, T0-SF, NIV2	1,836	**58.5** (+9.4)	**73.2**	68.1	**58.8**	**65.6**

- Human performance estimates are ~80 on Big-Bench (BBH)

Chung et al. (2022)

Self-Instruct/Alpaca



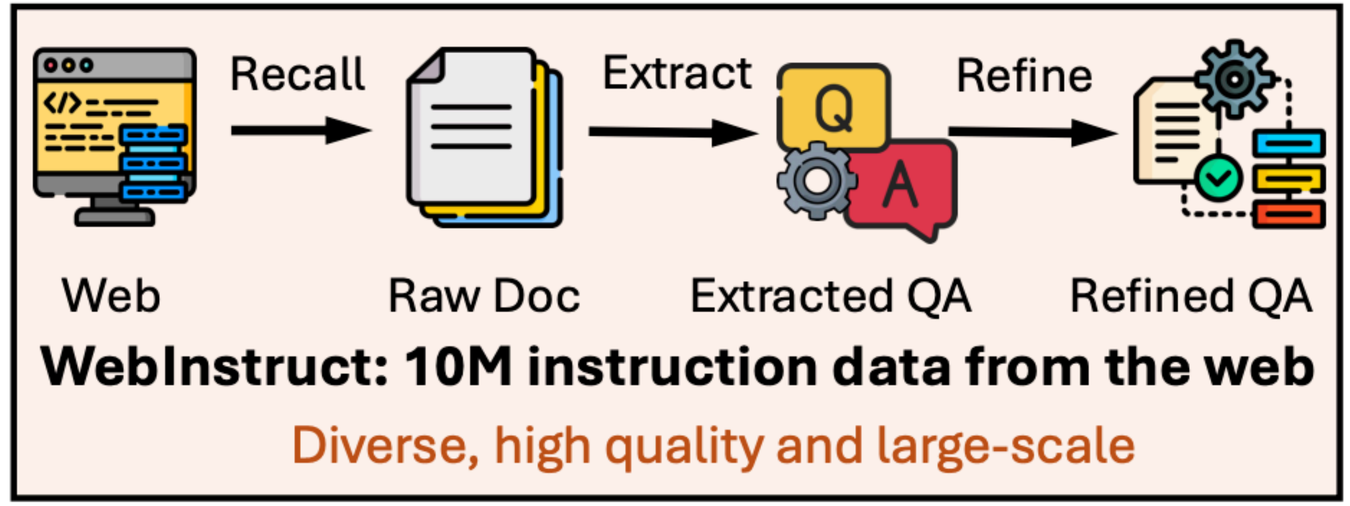
- Fine-tune Llama on 52k outputs with answers generated by text-davinci-003

Yizhong Wang et al. (2023) Self-Instruct

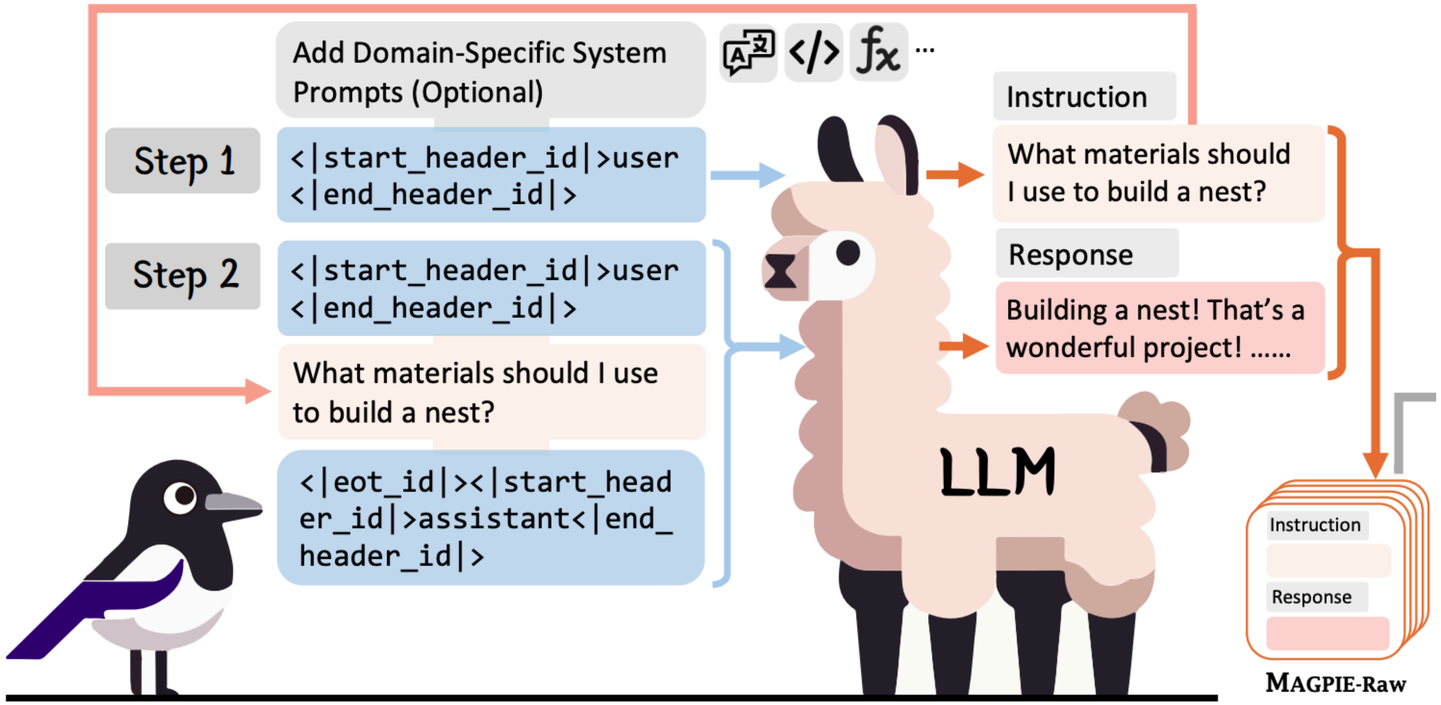
Ronen Taori et al. (2023) Alpaca

“Found” Instruction Data

- MAmmoTH2: extract instruction data from the web (using LLMs to reformulate it)



- MAGPIE: generate user prompts and then the responses from scratch using an LLM, then filter them and train on that data



Administrative details and recap

Hyperparameter Optimization

Scaling Law History

Data Scaling Laws

Model Scaling Laws

Chinchilla: Data x Model

Putting it together

Next time

- Post-training: understanding supervised fine-tuning (SFT)

- In two classes: GRPO/RLHF