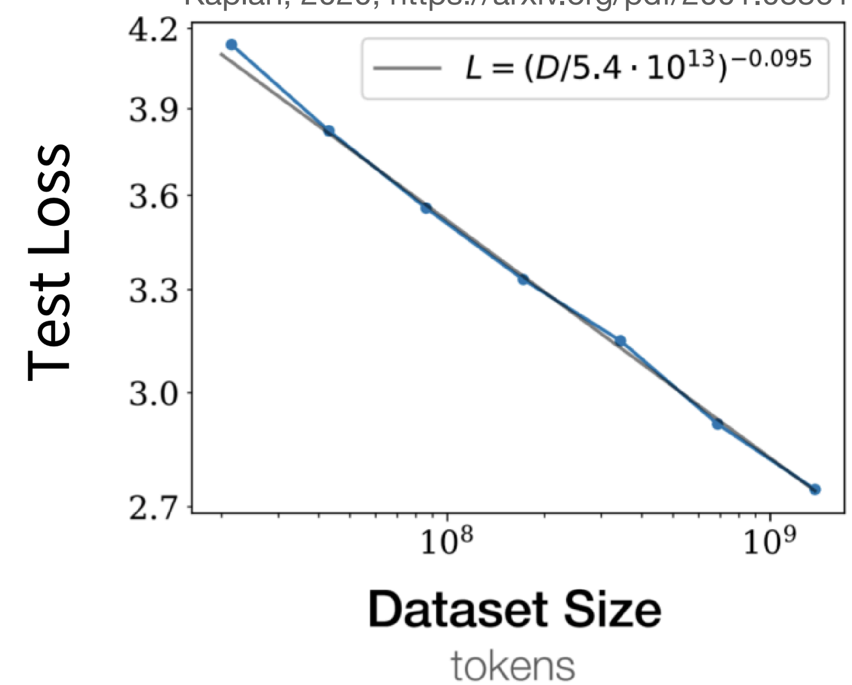
Goal Today: Different Approaches to Hyperparameter Tuning

How can we tune hyperparameters?

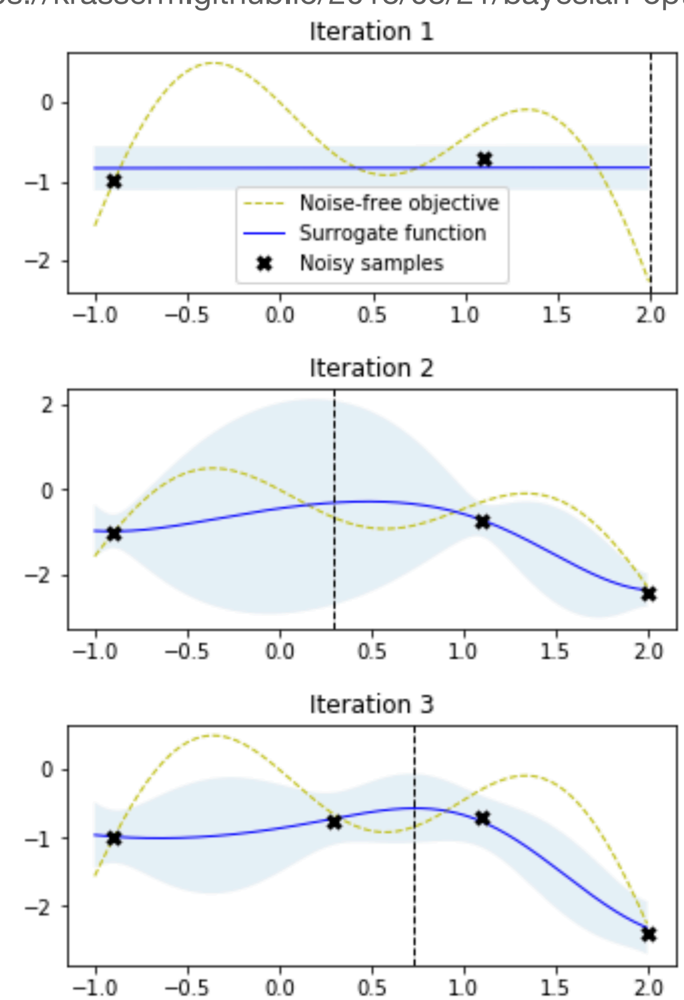
- Try and Pray
- Grid Search (costly)
- Do small-scale experiments. Then “extrapolate”

  1. Draw a line: **Scaling Law**
     https://stanford-cs324.github.io/winter2022/assets/pdfs/Scaling%20laws%20pdf.pdf
  2. (Multi-fidelity) Bayesian Optimization
  3. Update hyperparameters online (specifically data)

Kaplan, 2020, https://arxiv.org/pdf/2001.08361



https://krasserm.github.io/2018/03/21/bayesian-optimization/



https://stanford-cs324.github.io

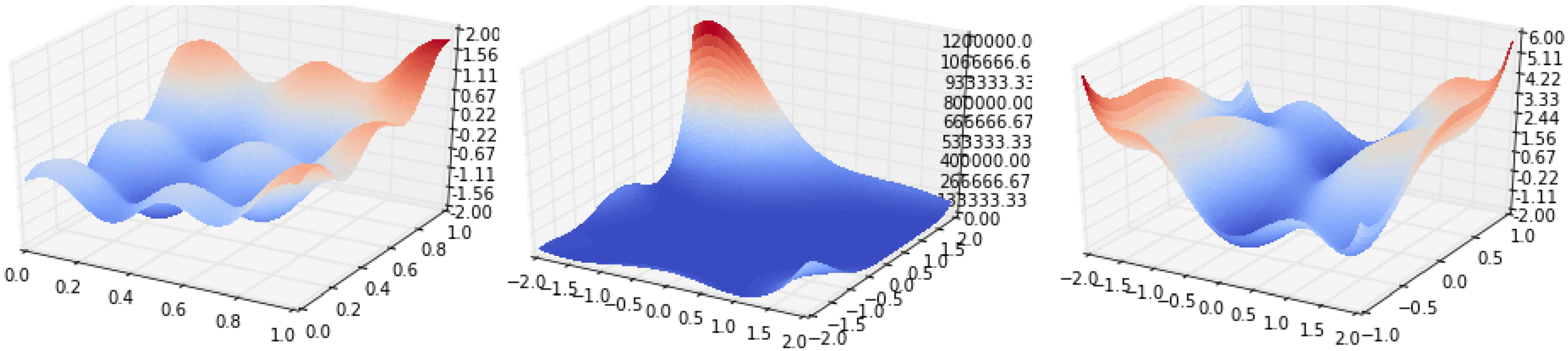
Hyperparameter Optimization

Problem: which training parameters should I use?

• 0.01 or 0.001 learning rate?

• 0.9 or 0.99 momentum?

• 4 or 5 decoder blocks?



Evaluation of f is expensive

Hyperparameter Tuning is Costly

Consumption	CO_2e (lbs)
Air travel, 1 passenger, NY↔SF	1984
Human life, avg, 1 year	11,023
American life, avg, 1 year	36,156
Car, avg incl. fuel, 1 lifetime	126,000
Training one model (GPU)	
NLP pipeline (parsing, SRL)	39
w/ tuning & experimentation	78,468
Transformer (big)	192
w/ neural architecture search	626,155

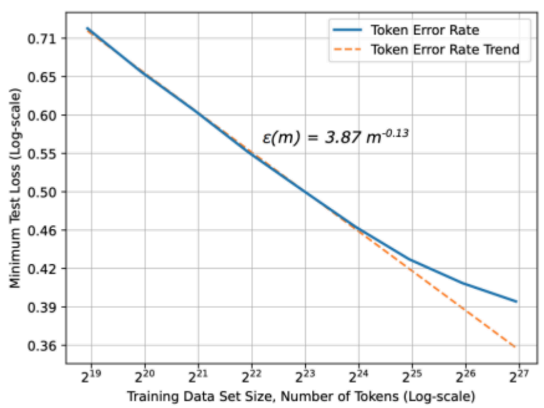
Table 1: Estimated CO_2 emissions from training common NLP models, compared to familiar consumption.[1]

GPT-4 cost more than 100 Millions!

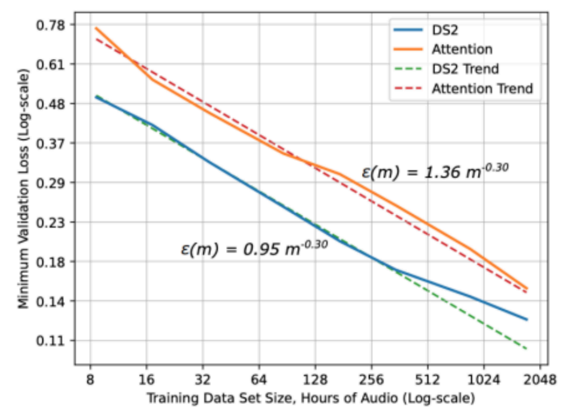
Infeasible to train many models

Scaling Laws: An Interesting Phenomenon

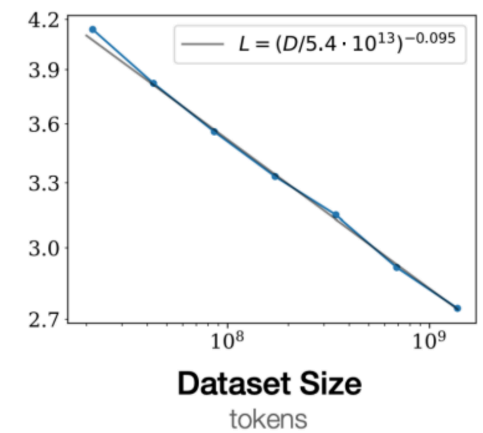
Scaling laws hold in many domains



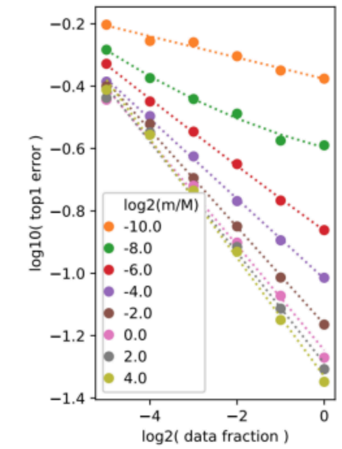
Machine translation



Speech



Language modeling



Object recognition

Hestness et al 2017.

Kaplan et al 2020.

Rosenfeld 2020.

Significance:

1. Predicting large-scale results => Efficient Design Choice

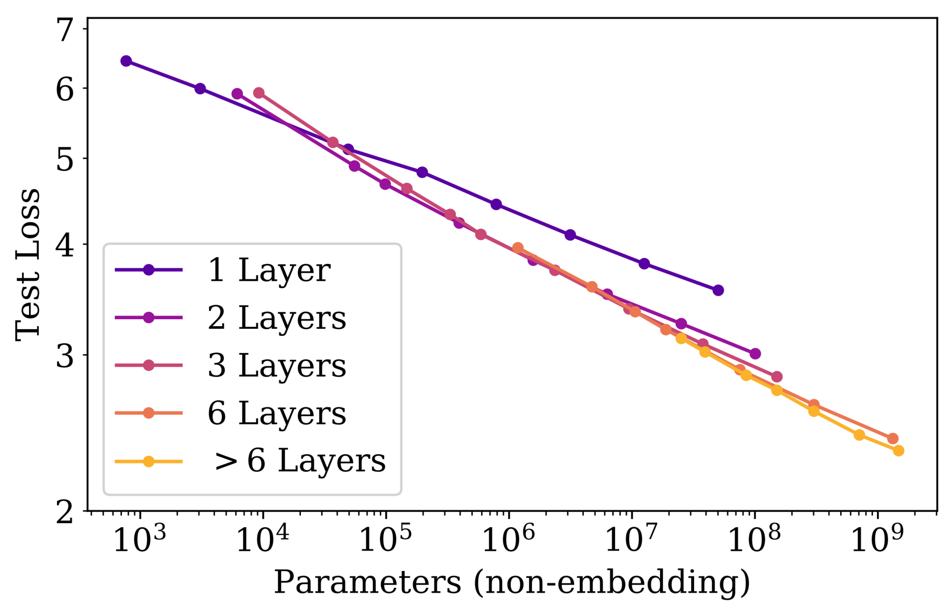
2. Allow low-budget contribution from research community

3. Enable resource allocation decisions (# of nuclear plants/gpus needed)

Scaling Law: Identify Best Hyperparameters From Trend

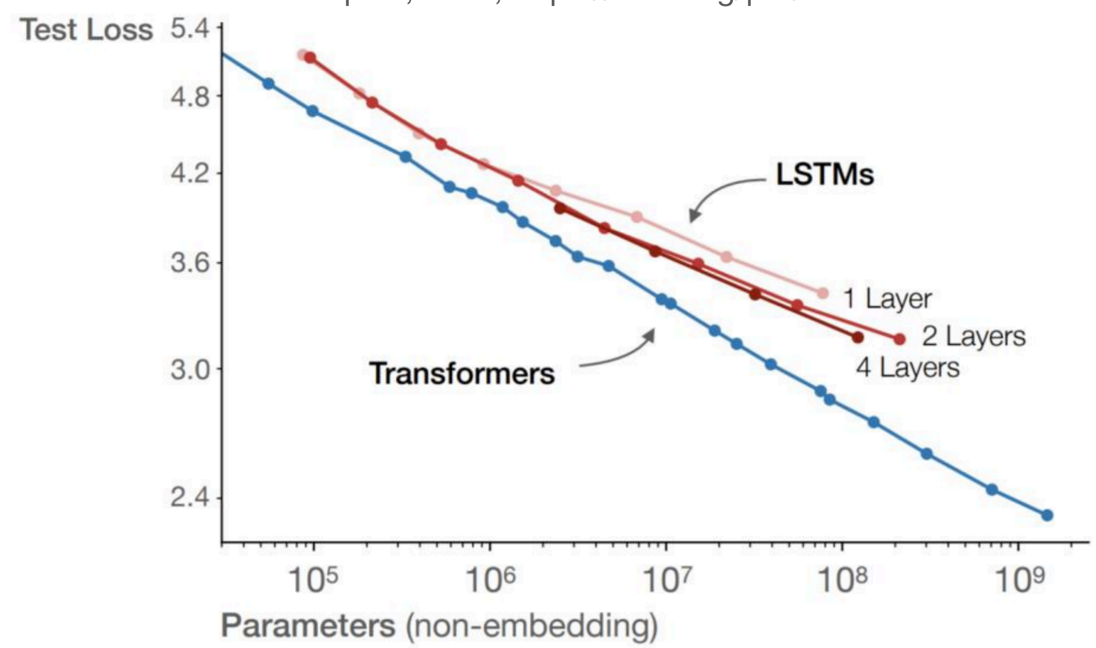
Best # of layers?

Kaplan, 2020, https://arxiv.org/pdf/2001.08361



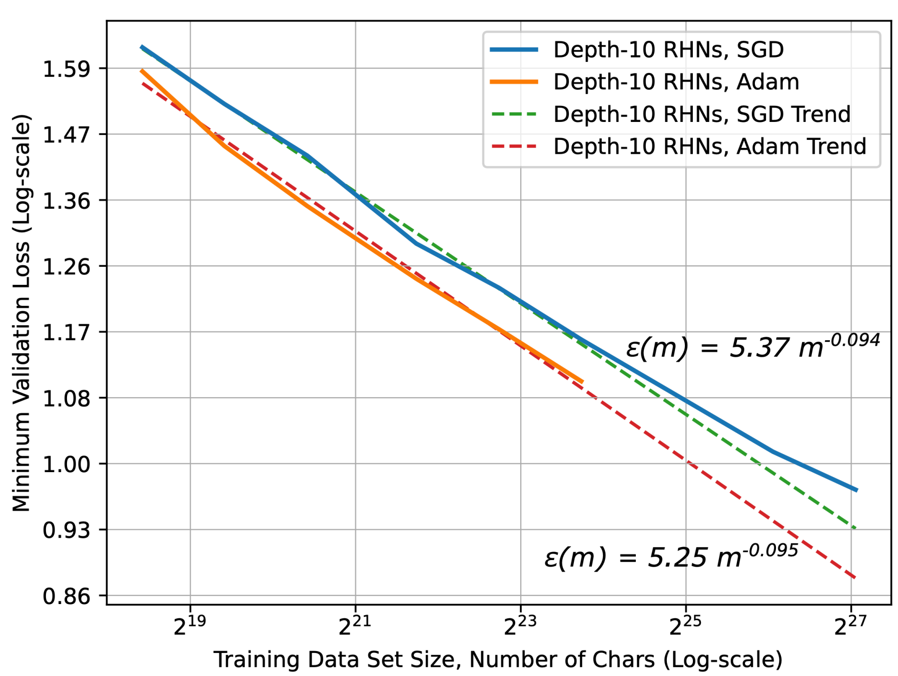
Best architecture?

Kaplan, 2020, https://arxiv.org/pdf/2001.08361



Best optimizer?

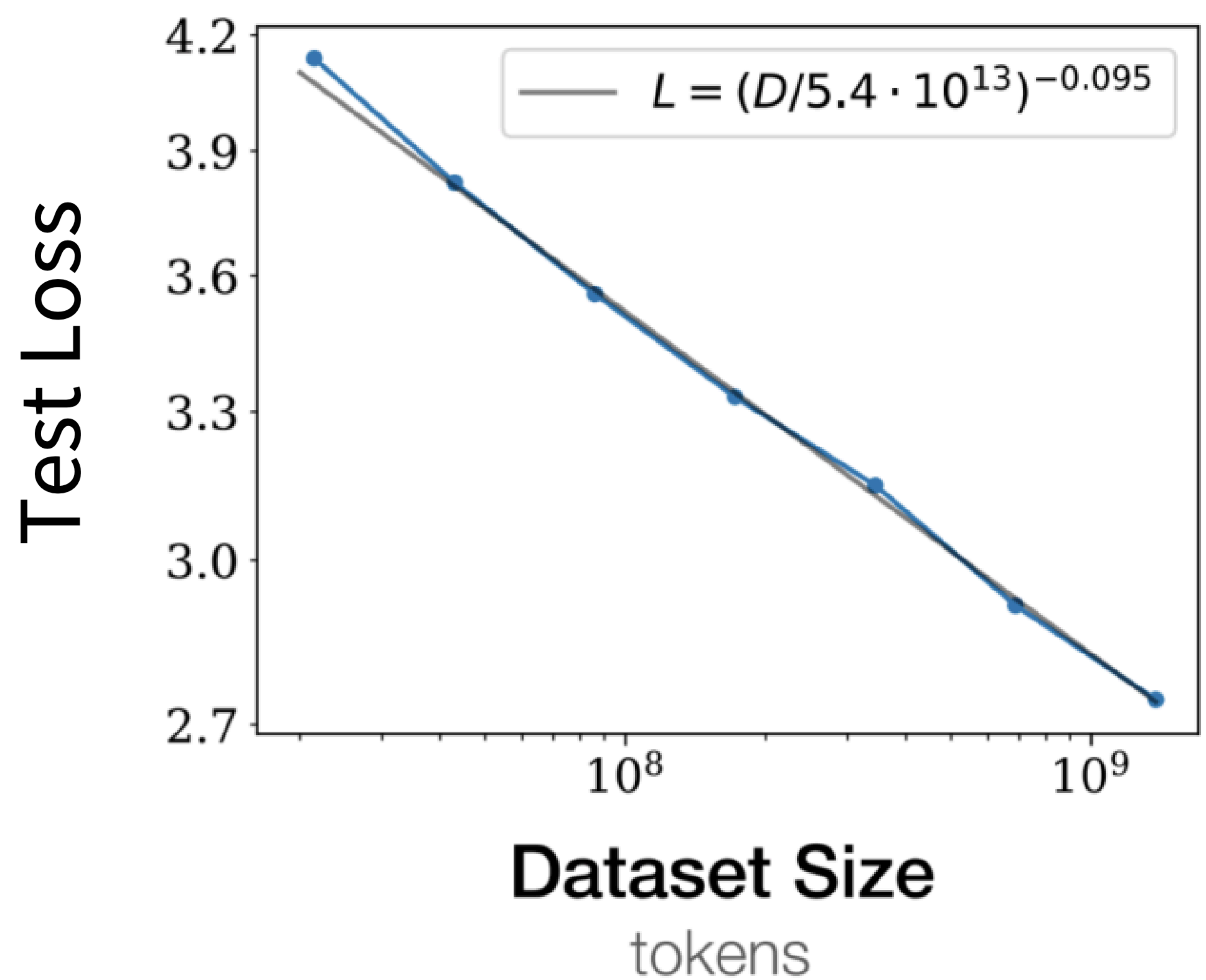
Hestness, 2017, https://arxiv.org/pdf/1712.00409



Data Scaling Laws for Language Models

An empirical observation:

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

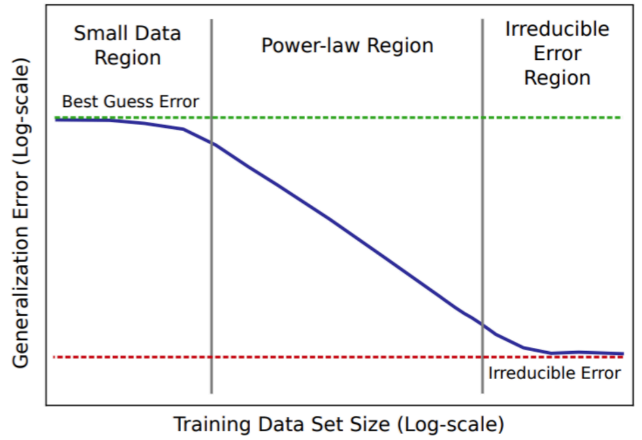


Kaplan, 2020, https://arxiv.org/pdf/2001.08361

Conceptual foundations of data scaling laws.

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?

Toy example: mean estimation

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

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

What's the error? By standard arguments..

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

This is a scaling law!!

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

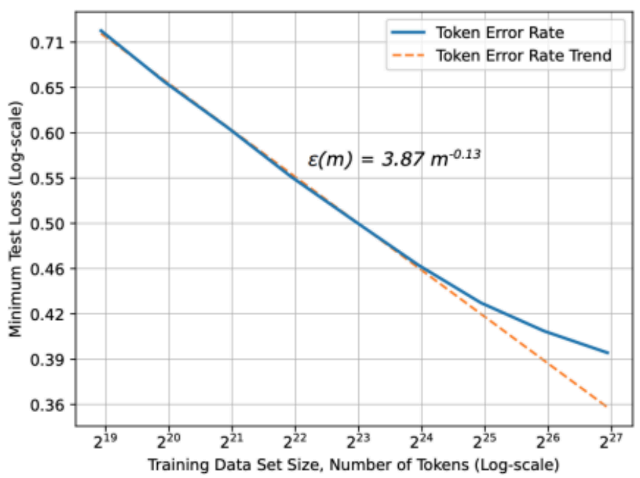
More generally, any polynomial rate $1/n^{\alpha}$ is a scaling law

Scaling law exponents: an intriguing mystery

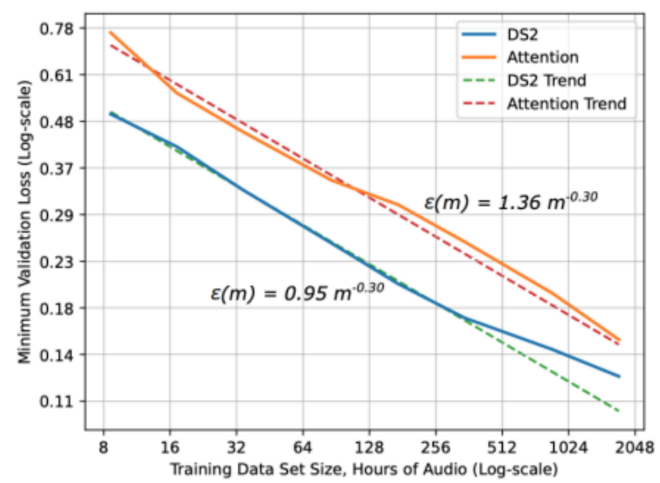
Fact: Similar arguments show most ‘classical’ models (regression, etc) have $\frac{1}{n}$ scaling

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

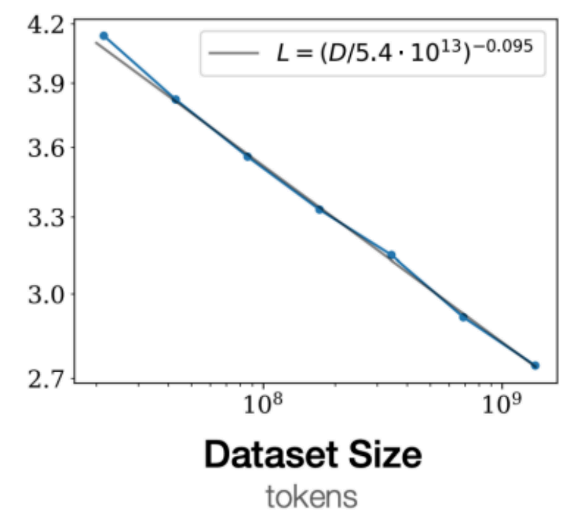
What do we find in neural scaling laws?



Machine translation



Speech



Language modeling

Very different from predictions.. Why might this be?

Detour: scaling laws for (nonparametric) learning

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)

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

What's our estimation error?

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

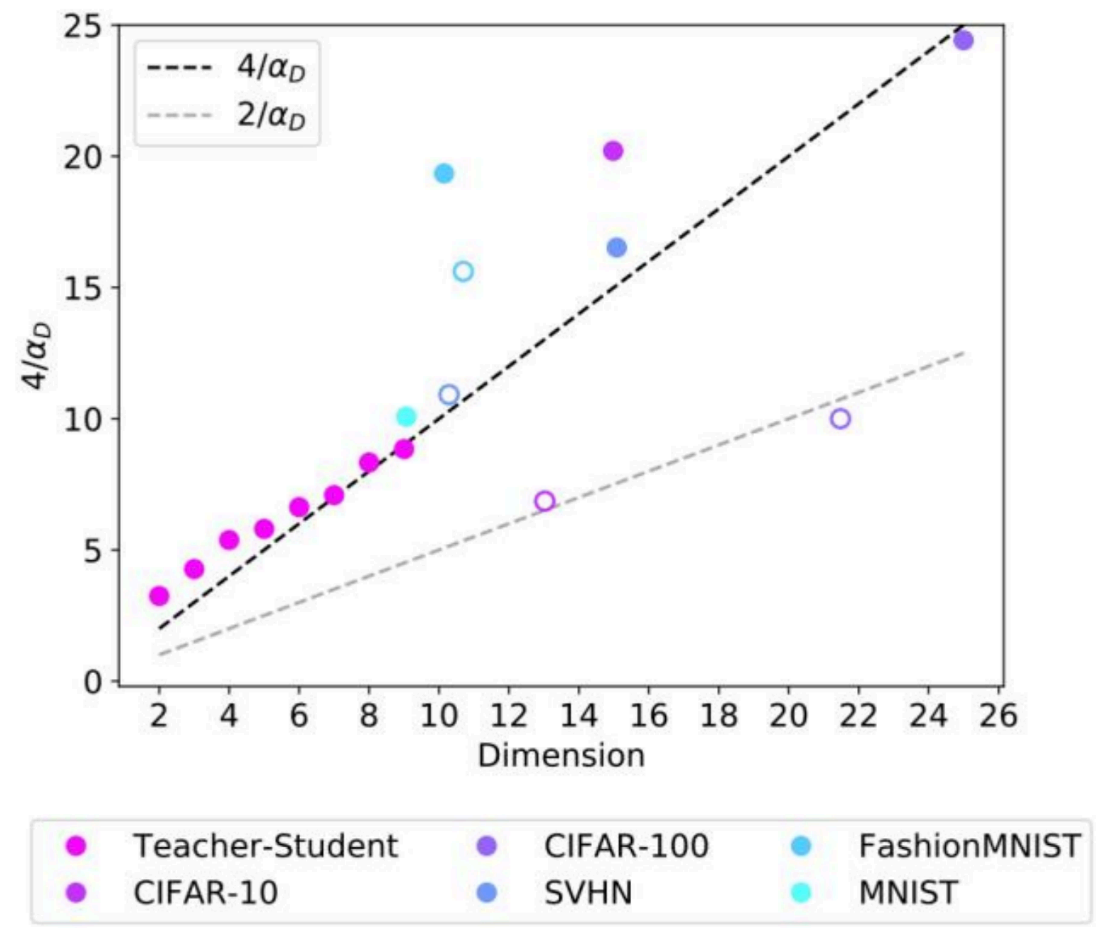
$$Error \approx \frac{1}{\sqrt{n}} + (other\ smoothness\ terms)$$

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

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

Intrinsic dimensionality theory of data scaling laws

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

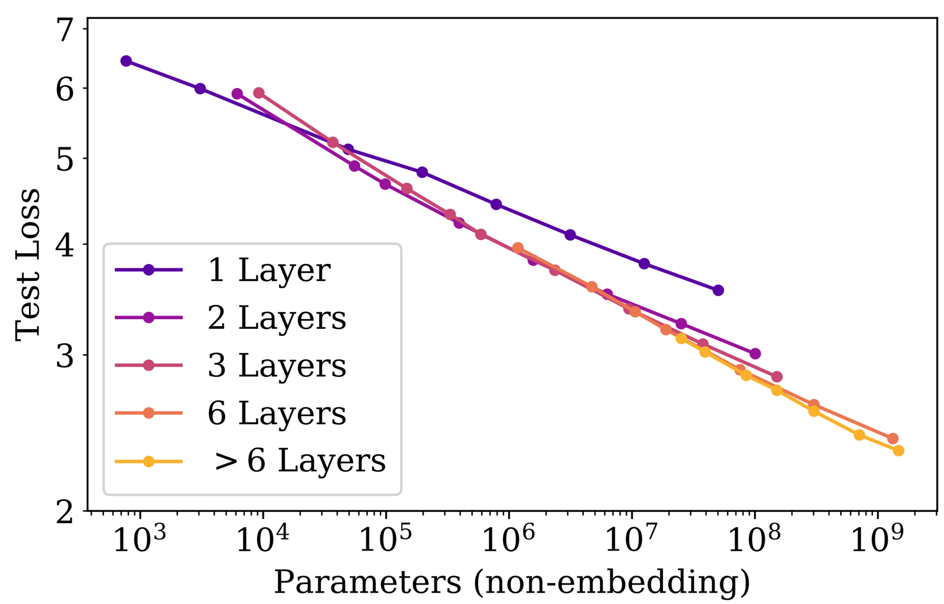


Some recent work (Bahri+ 2021) have tried to verify this empirically

X-axis of Scaling Laws Can Be Different (# of parameters)

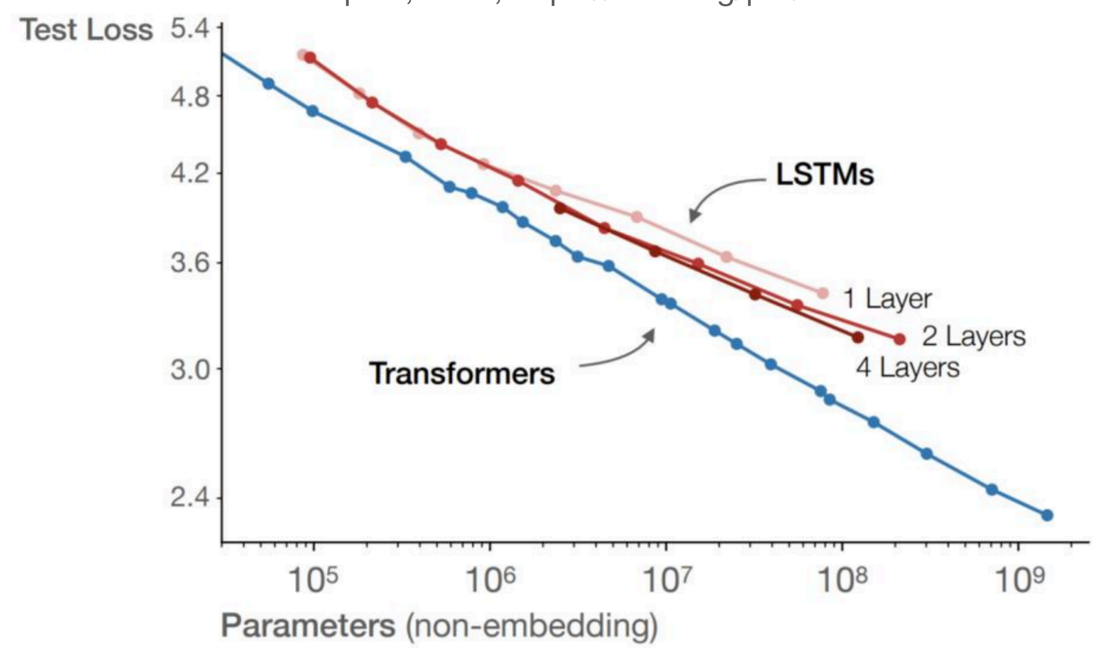
Best # of layers?

Kaplan et al., 2020, https://arxiv.org/pdf/2001.08361



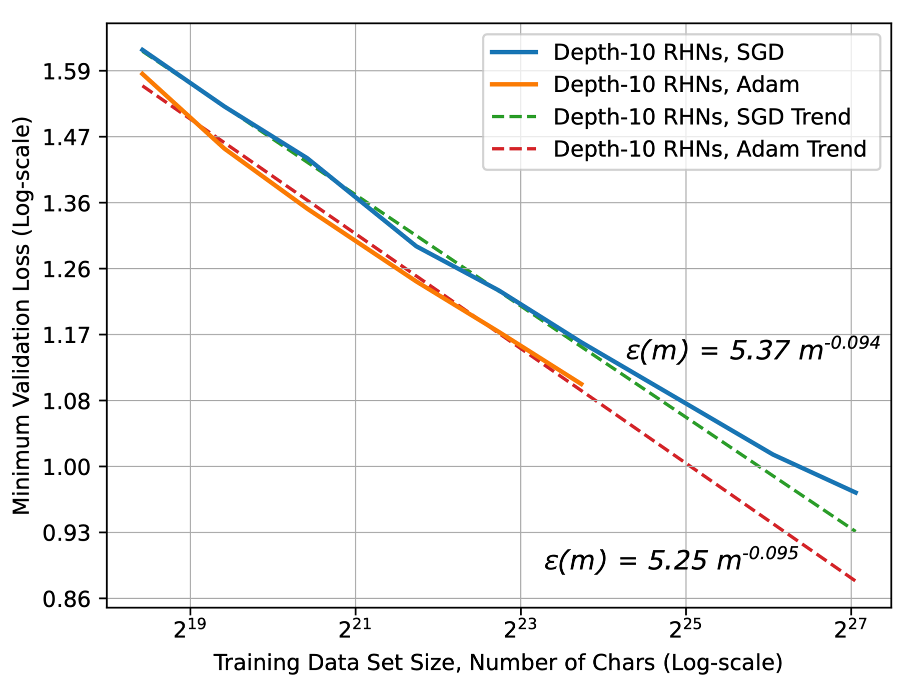
Best architecture?

Kaplan, 2020, https://arxiv.org/pdf/2001.08361



Best optimizer?

Hestness, 2017, https://arxiv.org/pdf/1712.00409



https://stanford-cs324.github.io

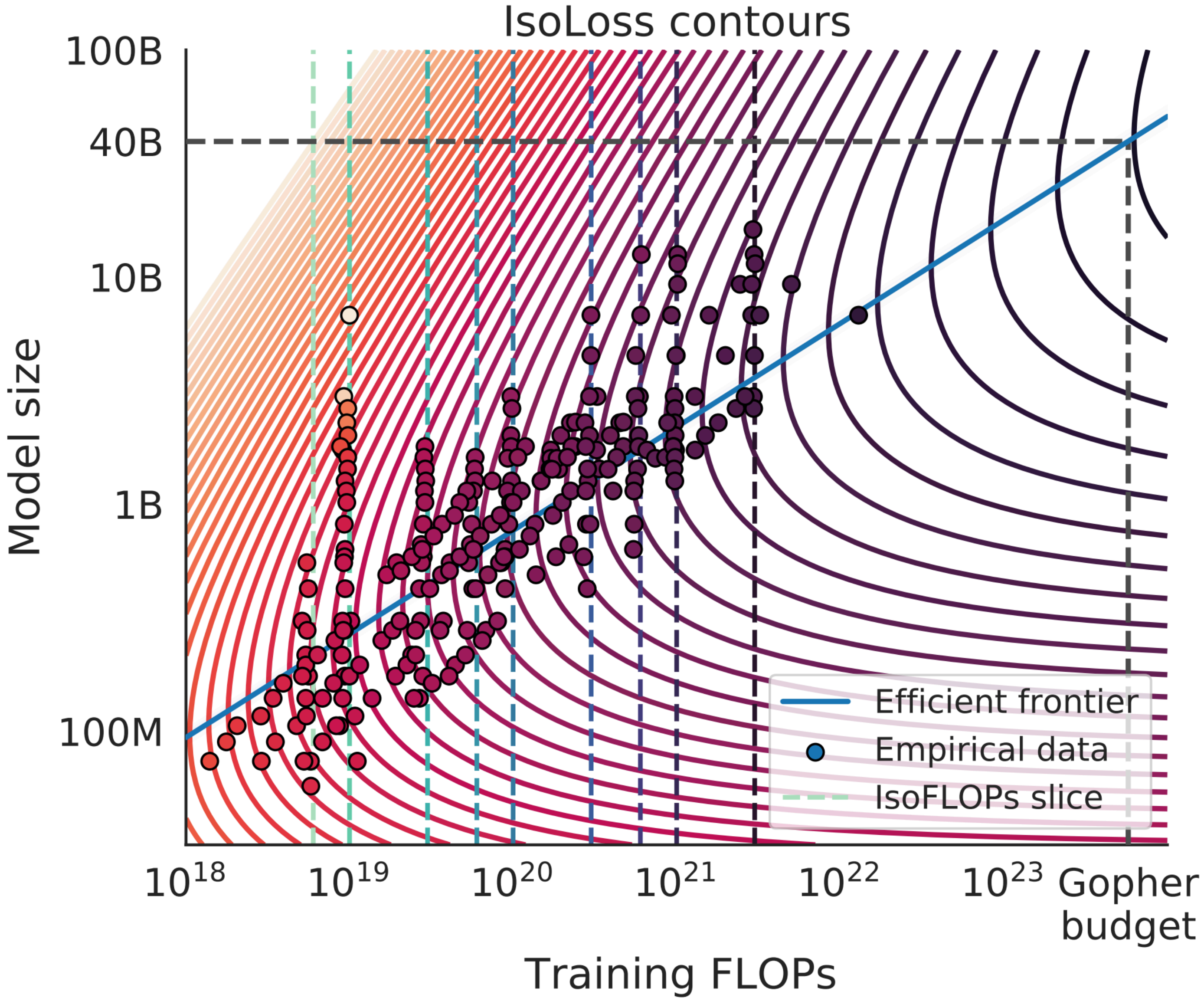
Joint parameter-data scaling Law. How to scale?

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

N Number of Tokens

D Number of Parameters

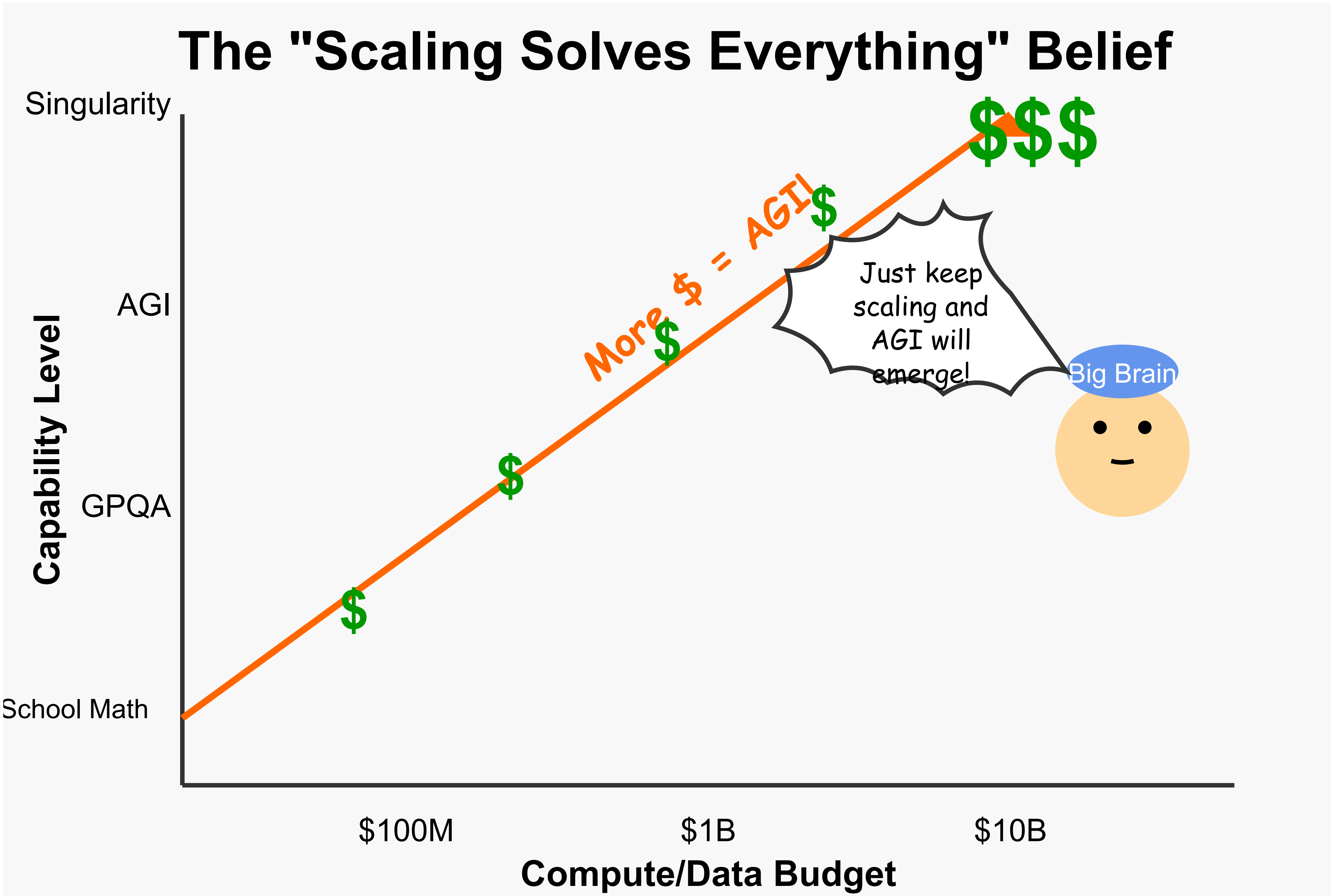
Hoffmann, 2022, https://arxiv.org/pdf/2203.15556



Empirically $\alpha \approx \beta$

Scale data and model size proportionally

AGI Soon?



The "Scaling Solves Everything" Belief
Singularity
AGI
GPQA
School Math
Capability Level
$100M
$1B
$10B
Compute/Data Budget
More $ = AGI!
$
$
$
$
$$$
Just keep scaling and AGI will emerge!
Big Brain

Putting on reviewer 2's hat:

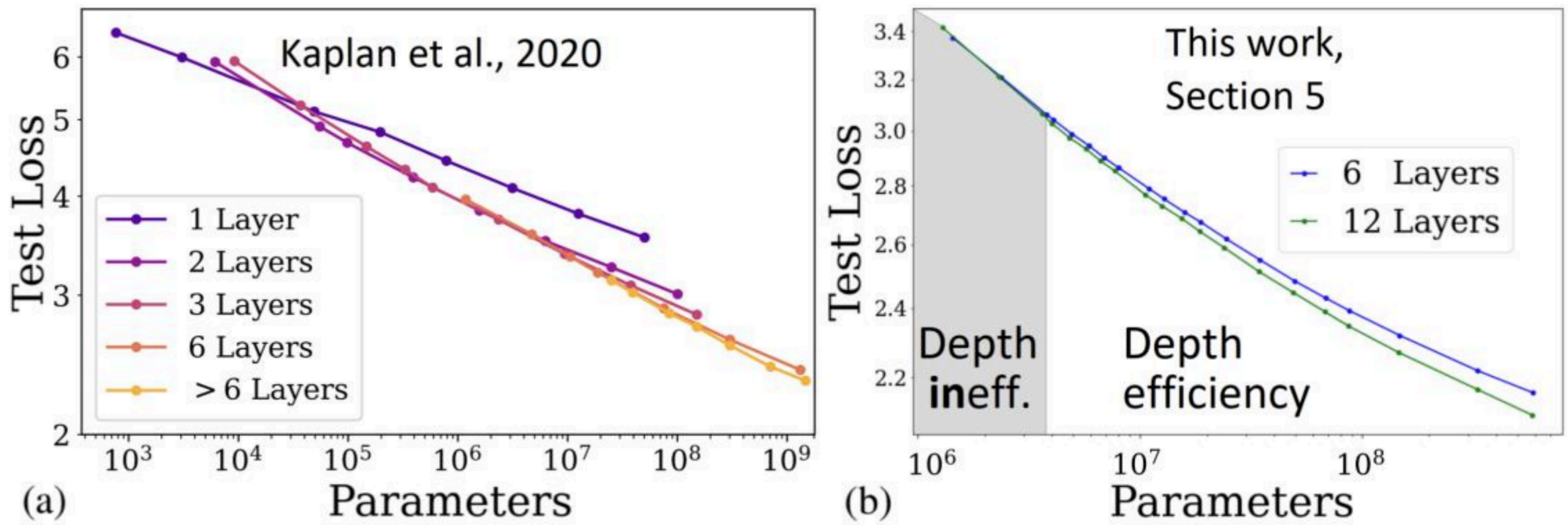
1. How well does it extrapolate? Does best parameter remain fixed at all scale?
2. What is x? Are all parameters/data equal?
3. What is y? What's the scaling behaviour on task accuracy? Or on OOD data
4. How do parameters of scaling law depend on architecture, or on the relationship between train/test data?

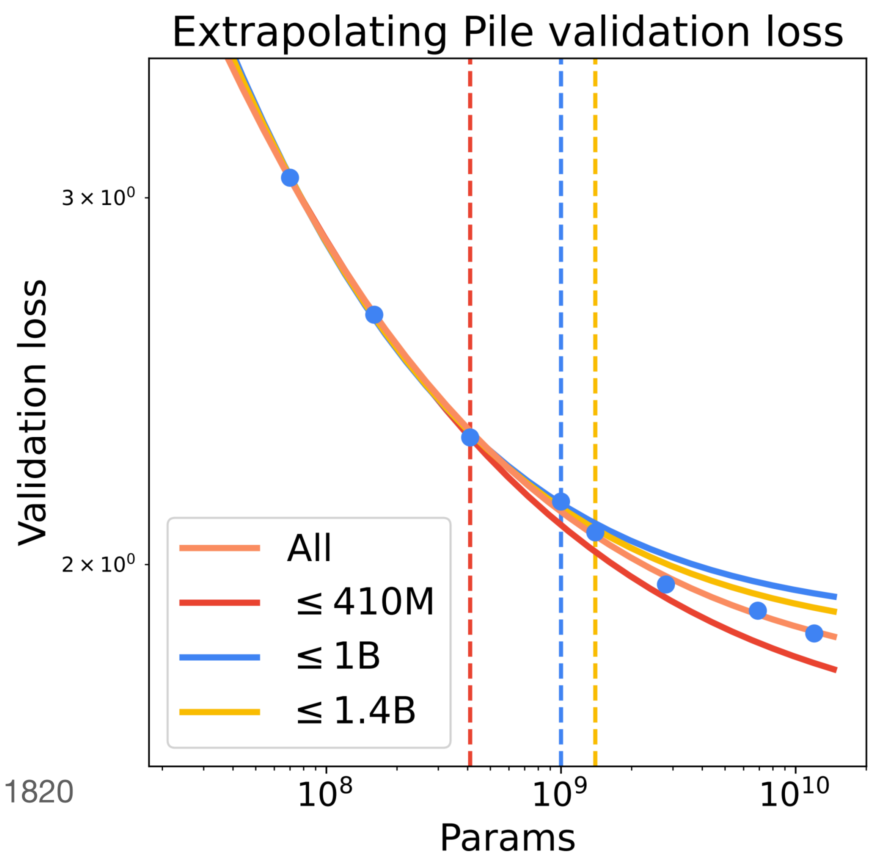
$$L = L_\epsilon + \beta n^{-\alpha}$$

$$\alpha = f(\text{architecture, relevance of train to test data})?$$

Scaling Law's extrapolation can lead you astray

Levine et. al, 2021

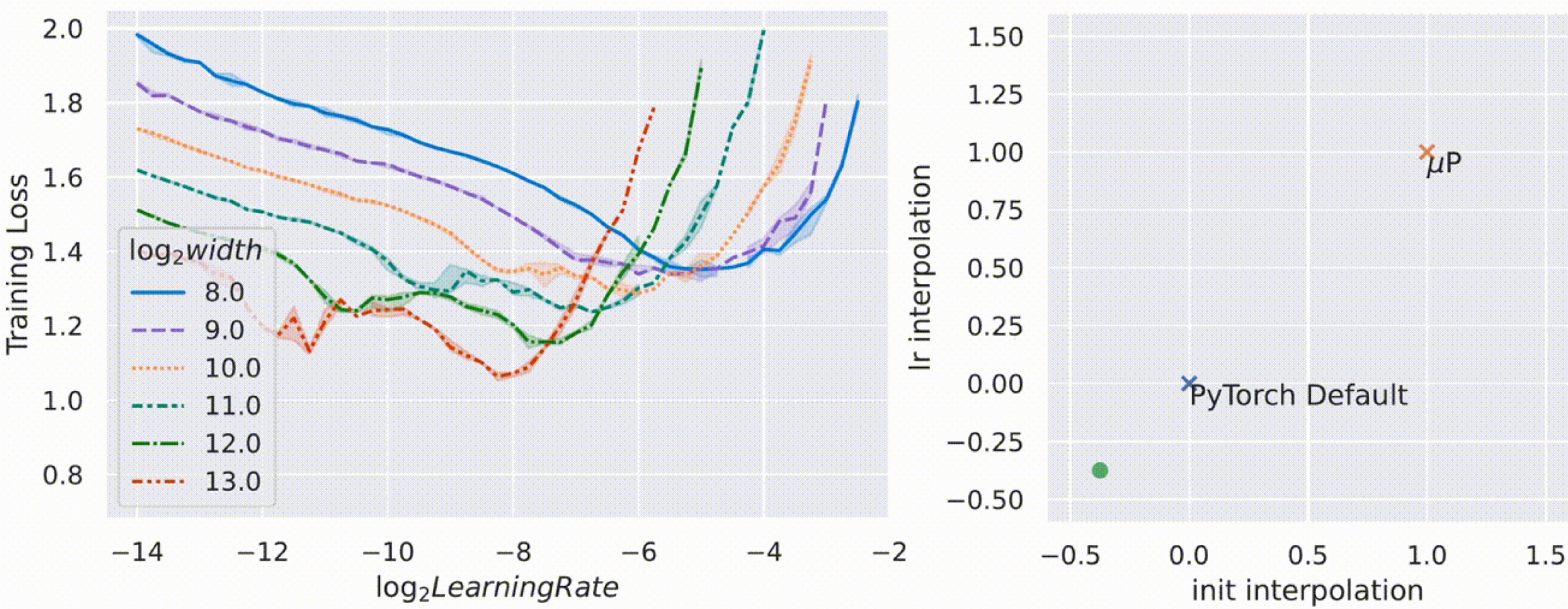




Jiang et. al, 2024 https://arxiv.org/pdf/2410.11820

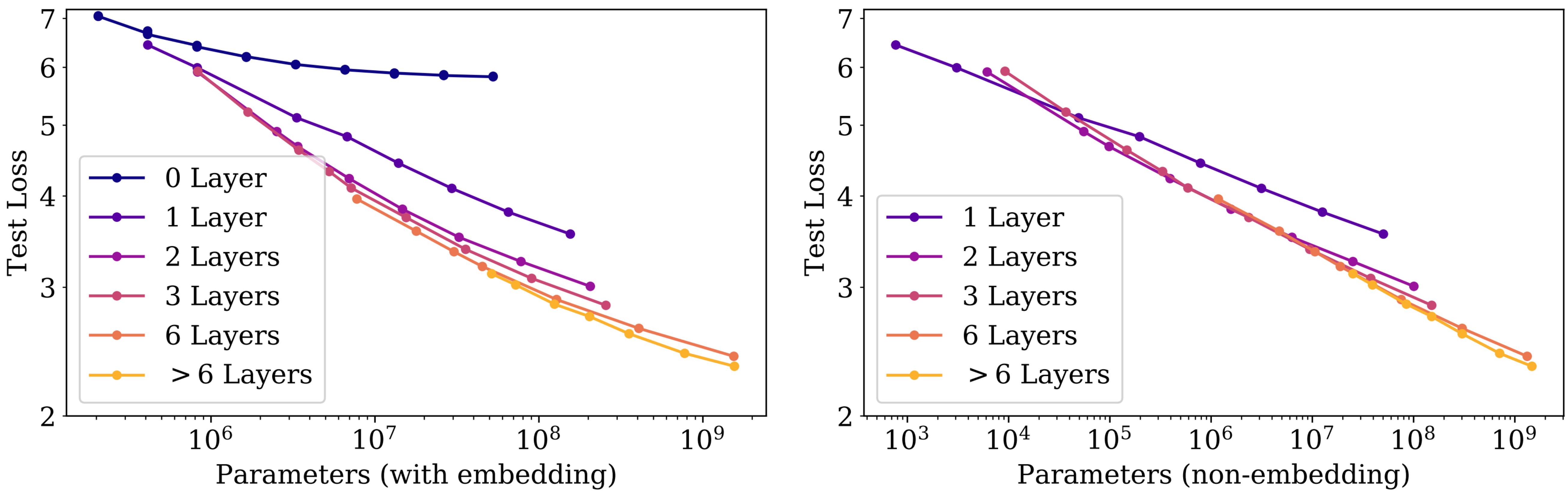
Scaling Law’s extrapolation can lead you astray

https://www.microsoft.com/en-us/research/blog/µtransfer-a-technique-for-hyperparameter-tuning-of-enormous-neural-networks/



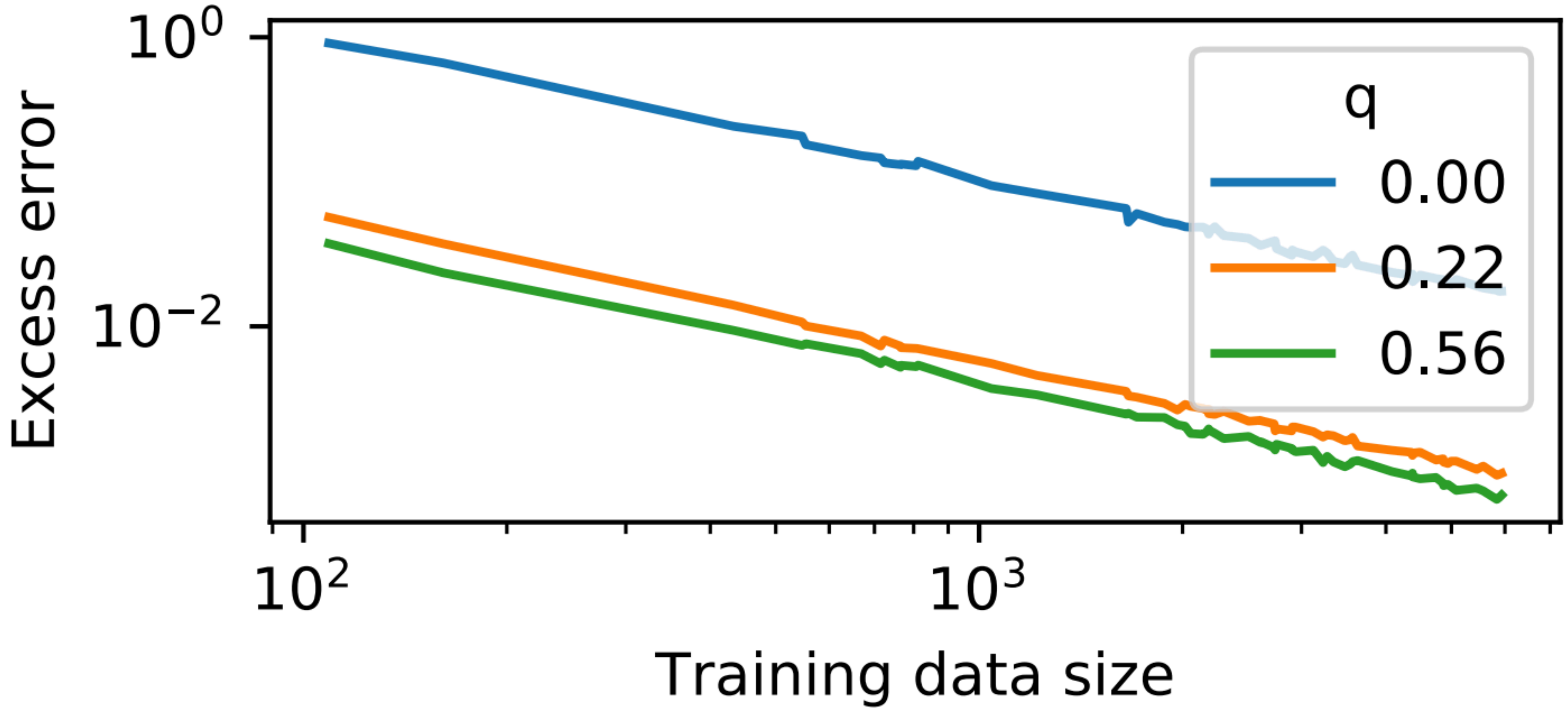
Are all parameters equal?

Kaplan, 2020, https://arxiv.org/pdf/2001.08361



Are all data equal? Intercept changes but exponent does not

Hashimoto, 2021, https://proceedings.mlr.press/v139/hashimoto21a/hashimoto21a.pdf



q: proportion of one of the two training data

$$\log(L(n,q)) \approx \log(V(n,q)) := \alpha(q)\log(n) + C(q).$$ ✗

$$\log(L(n,q)) \approx \log(V(n,q)) := -\alpha\log(\mathbf{n}) + \log(C(q)).$$ ✓

Data composition does not affect the slope?

Are all data equal? Different data sources changes exponent

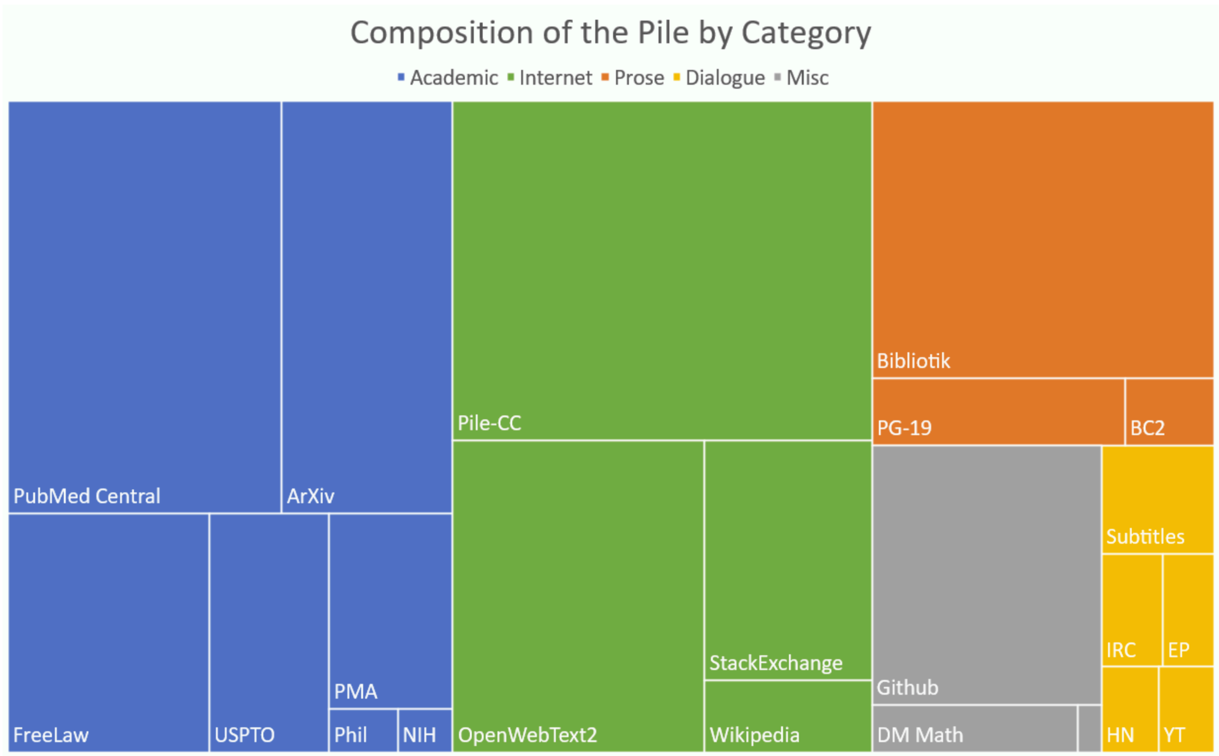


Figure 1: Treemap of Pile components by effective size.

$$L_i(r_{1\ldots M}) = c_i + k_i \exp\left(\sum_{j=1}^{M} t_{ij} r_j\right)$$

i: Domain of validation data

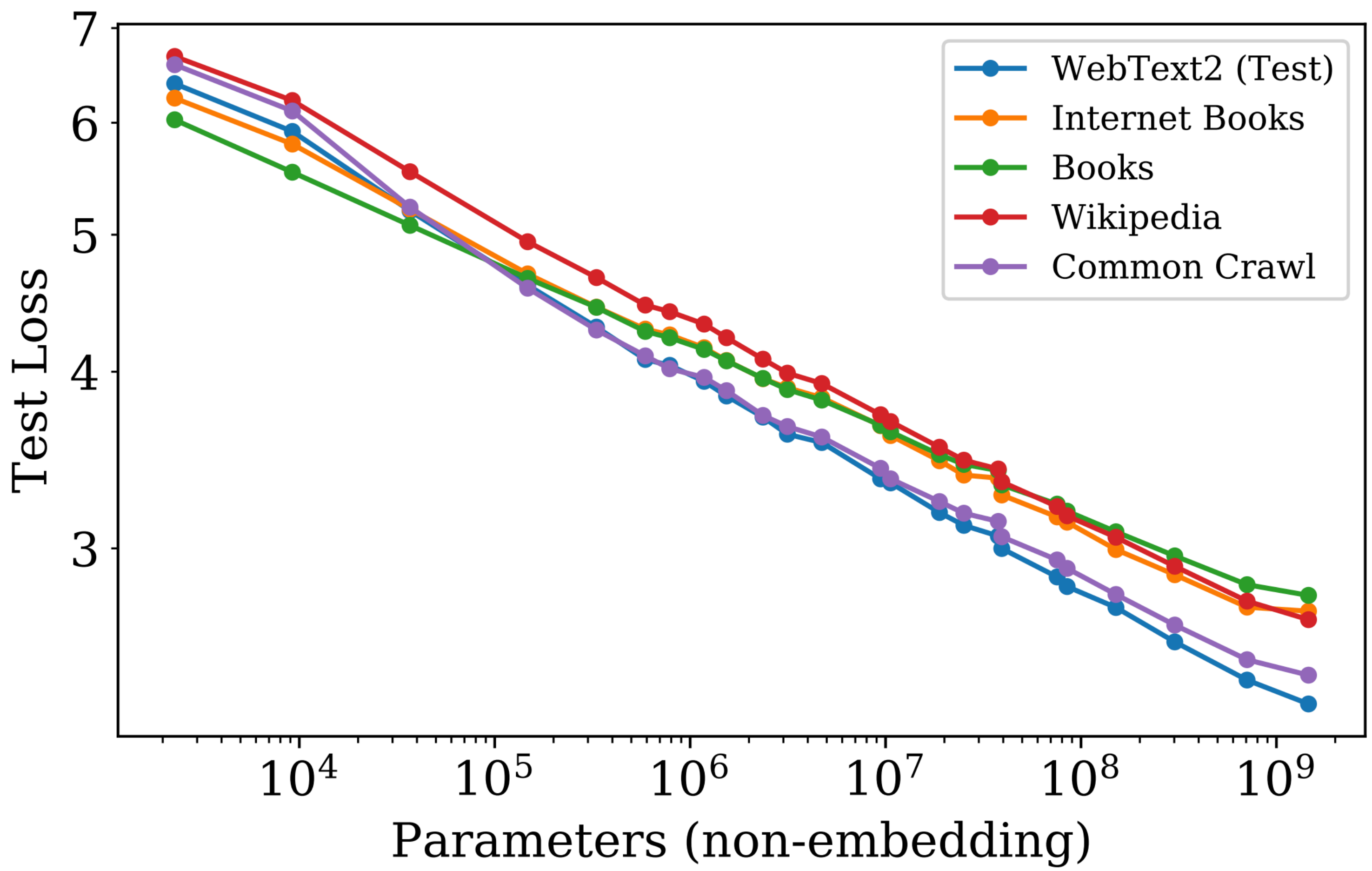
j: Domain of training data

r_j: Proportion of training data from domain j

t_ij: How much does training domain j helps validation domain I

Are all data equal? Different target data changes exponent

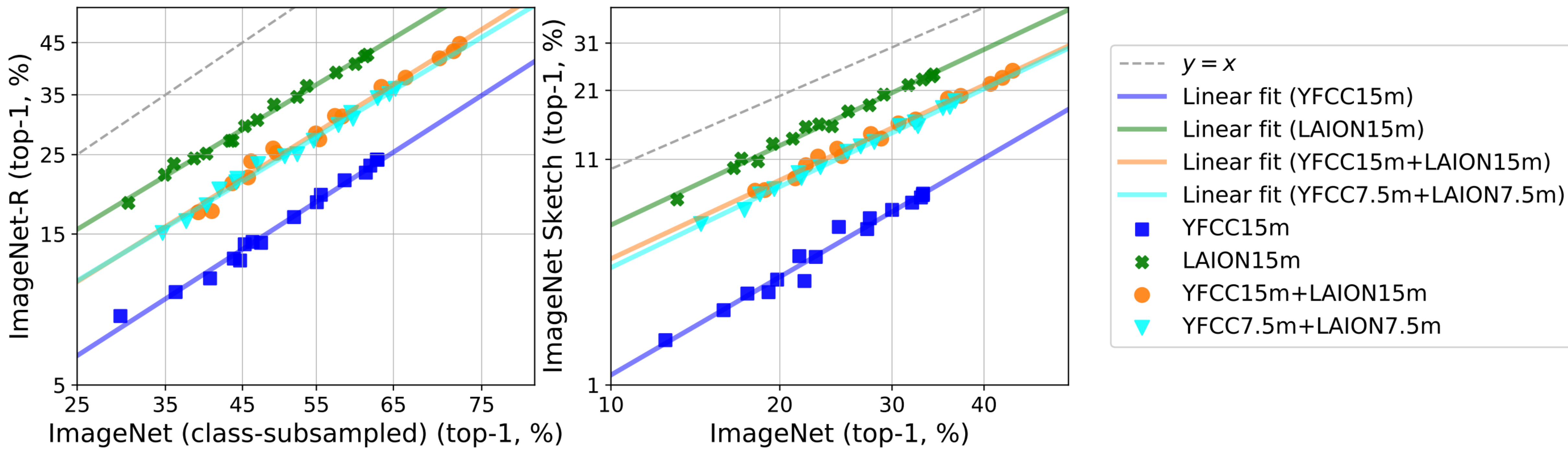
Kaplan et al., 2020, https://arxiv.org/pdf/2001.08361



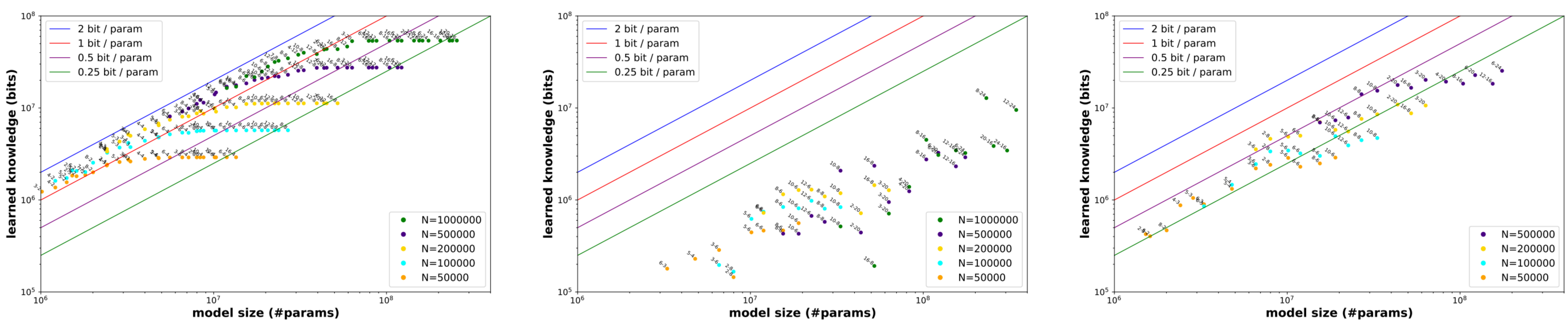
Clearly Webtext & Common Crawl have slopes different from Book's

Are all data equal? Bad data can worsen your model

Nguyen et al., 2022, https://arxiv.org/pdf/2208.05516



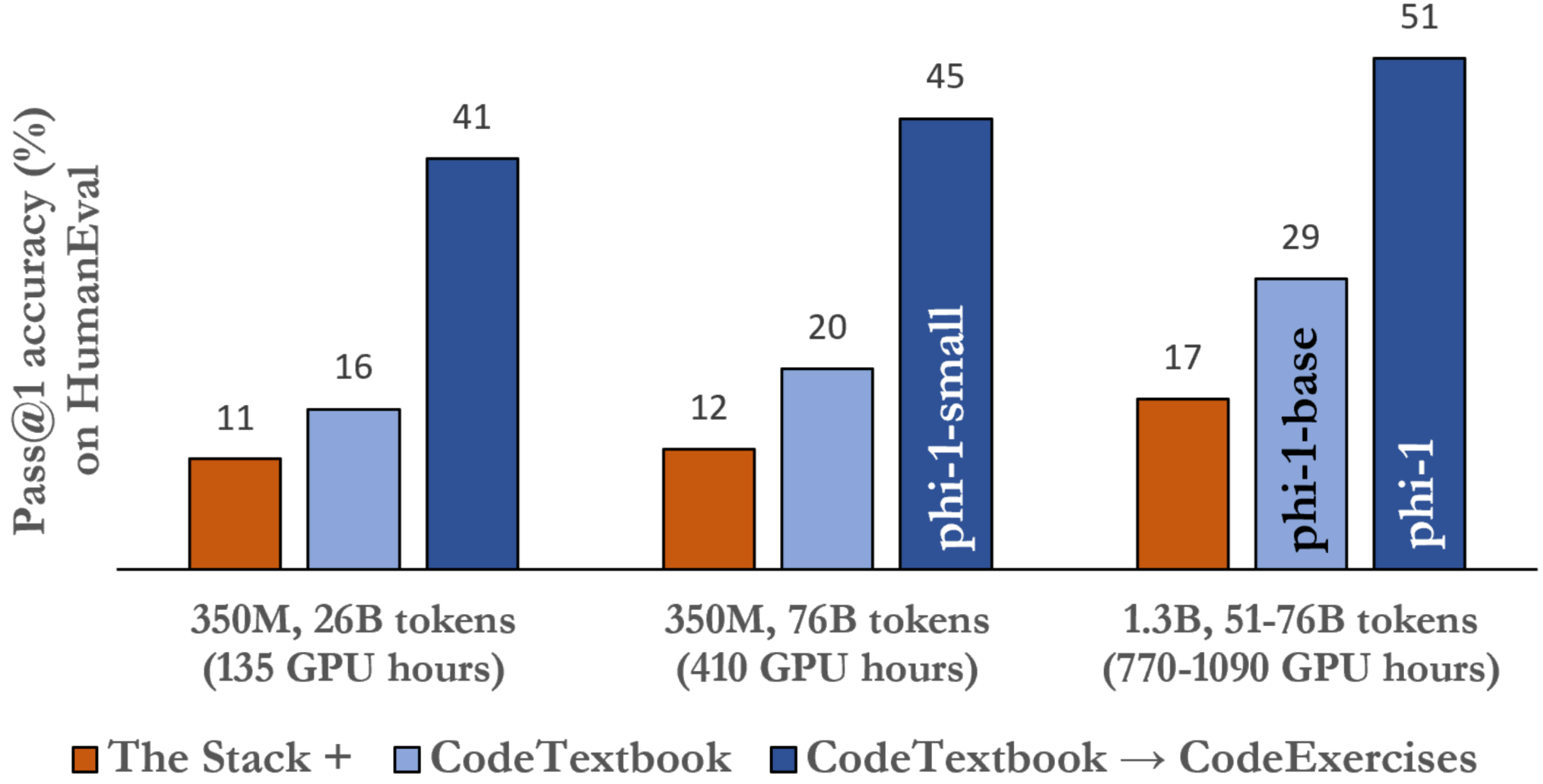
Zhu et al., 2024, https://arxiv.org/pdf/2404.05405



(a) no junk, 100 exposures (b) 7/8 junk, 100 exposures (c) 7/8 junk, 300 exposures

Are all data equal? High-quality data gives you a lot more

Gunasekar et al., 2023, https://arxiv.org/pdf/2306.11644



Tiny Stories: 10M-sized model can generate coherent English

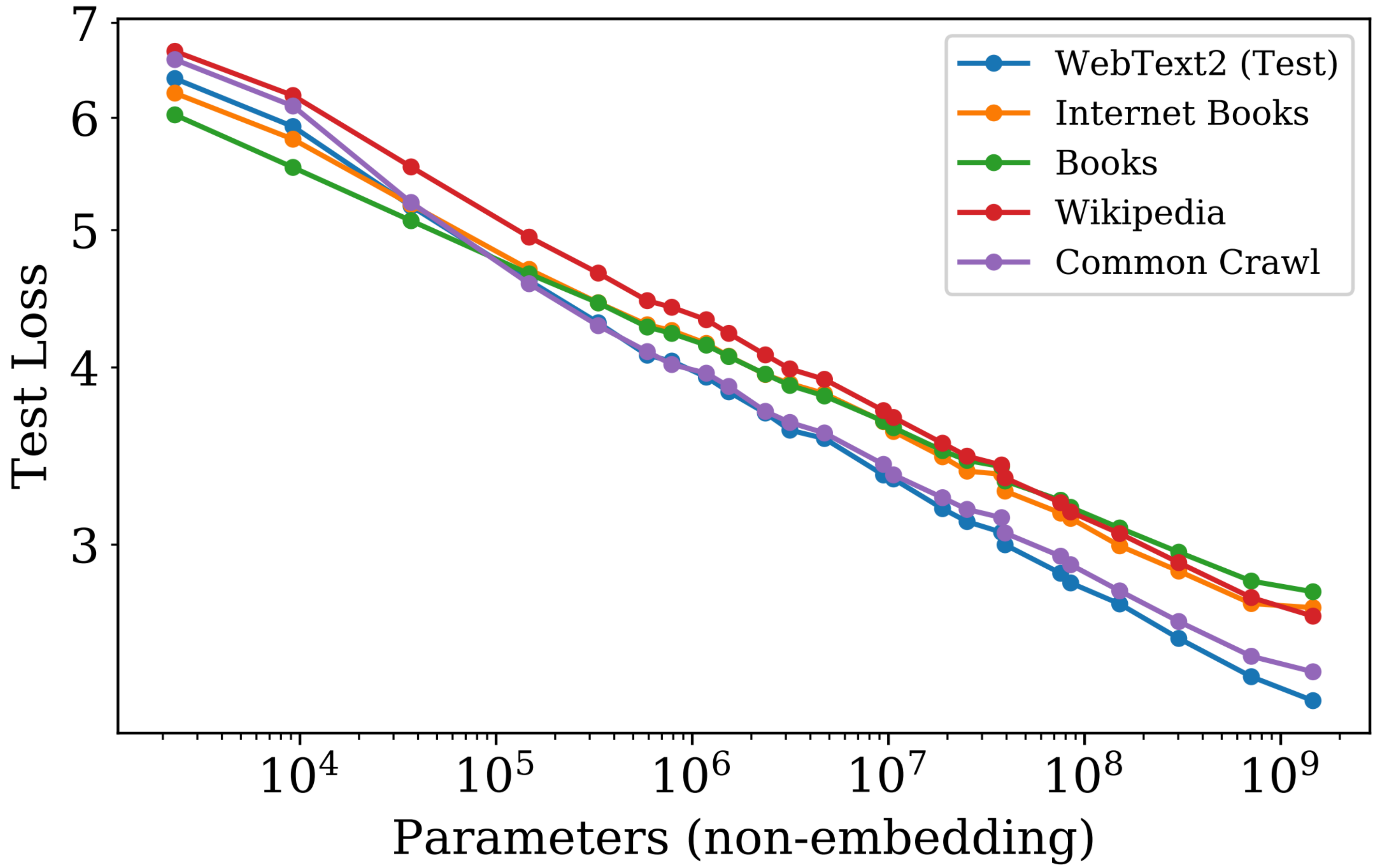
when 125M models (GPT-Neo, GPT-2) cannot.

Eldan et al., 2023, https://arxiv.org/pdf/2305.07759

Does Scaling Law work out of distribution?

Trained on WebText. Evaluate on the rest.

Kaplan, 2020, https://arxiv.org/pdf/2001.08361

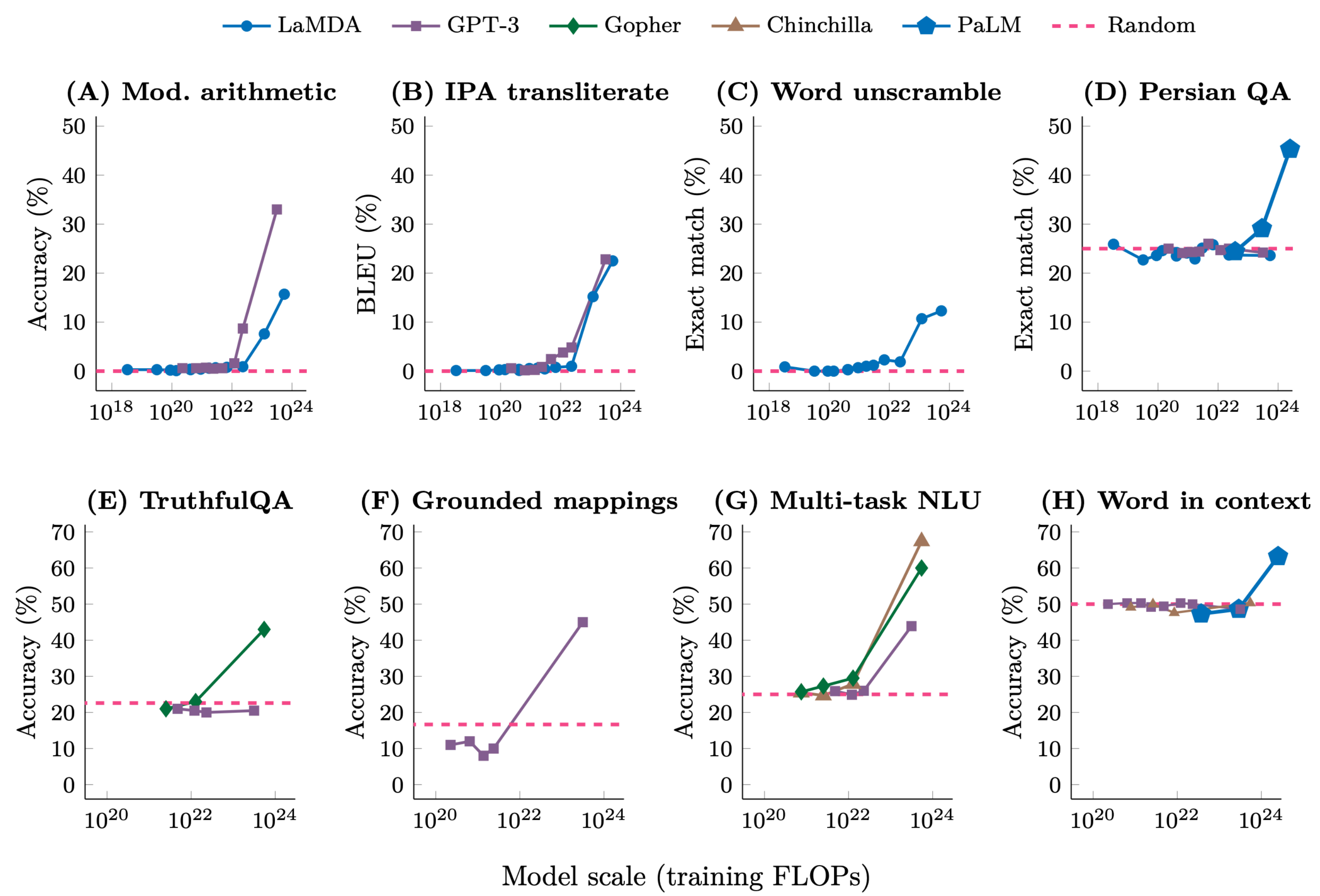


Scaling Law still holds, albeit with different intercept (and slope?)

Does Scaling Law work for downstream tasks?

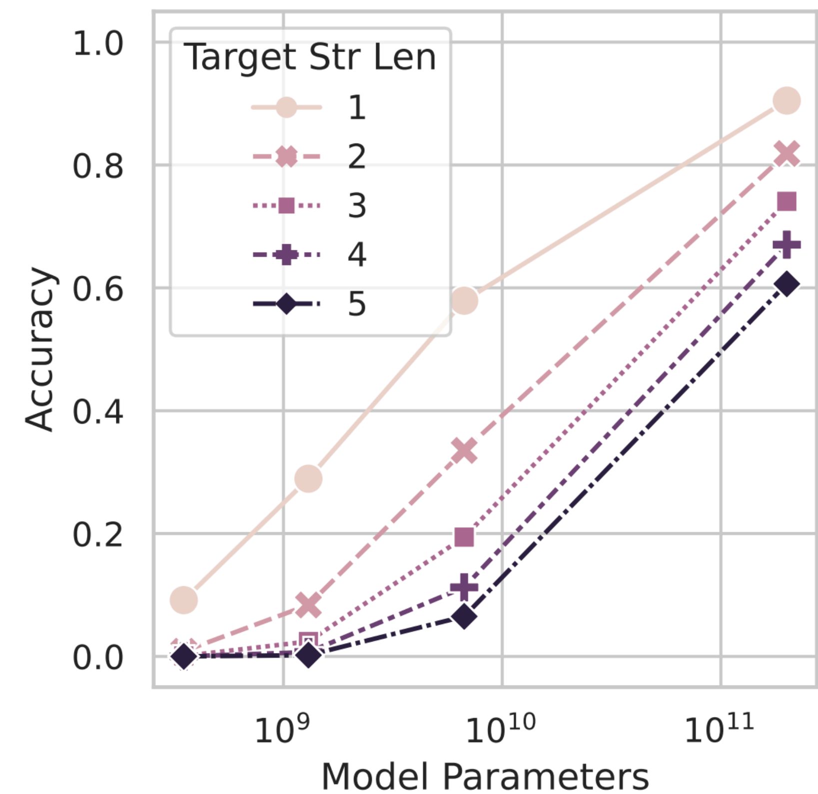
No, at first glance…

Emergent Behaviour

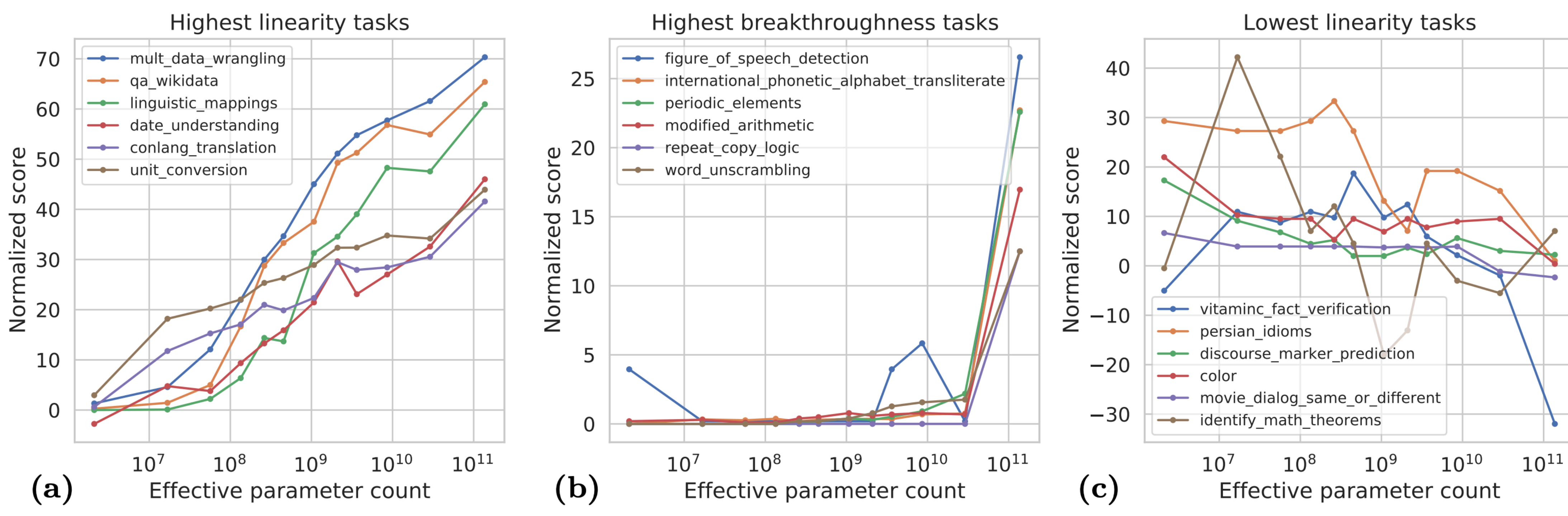


Does Scaling Law work for downstream tasks?

Just because LLM needs to be correct multiple times



Task dependent:

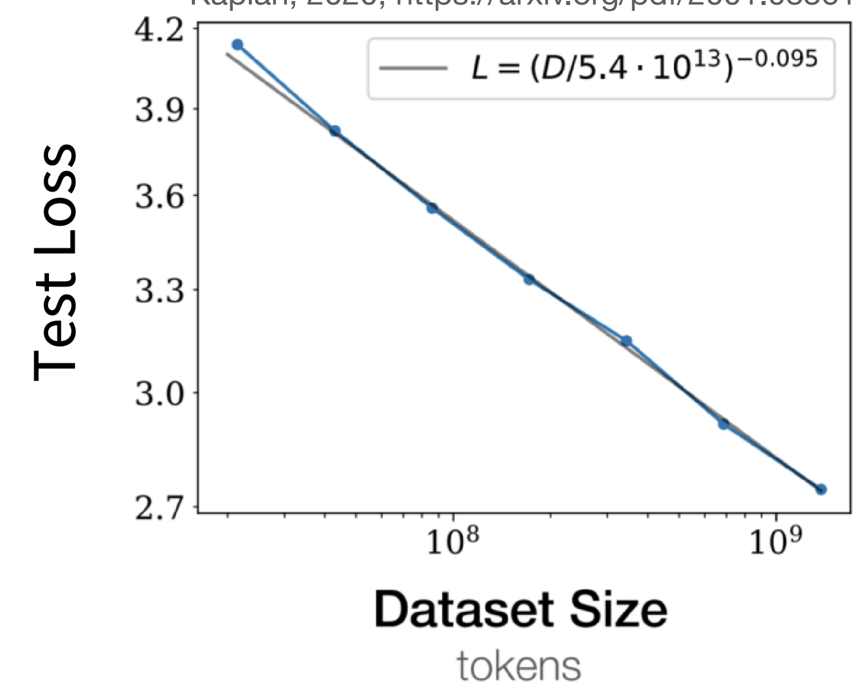


Recap on Scaling Law

- Surprisingly robust pattern. Has theoretical foundation.
- Doesn’t always work. Need to carefully think about the axis.

Motivating problem: hyperparameter costs

Kaplan, 2020, https://arxiv.org/pdf/2001.08361



How can we solve this?

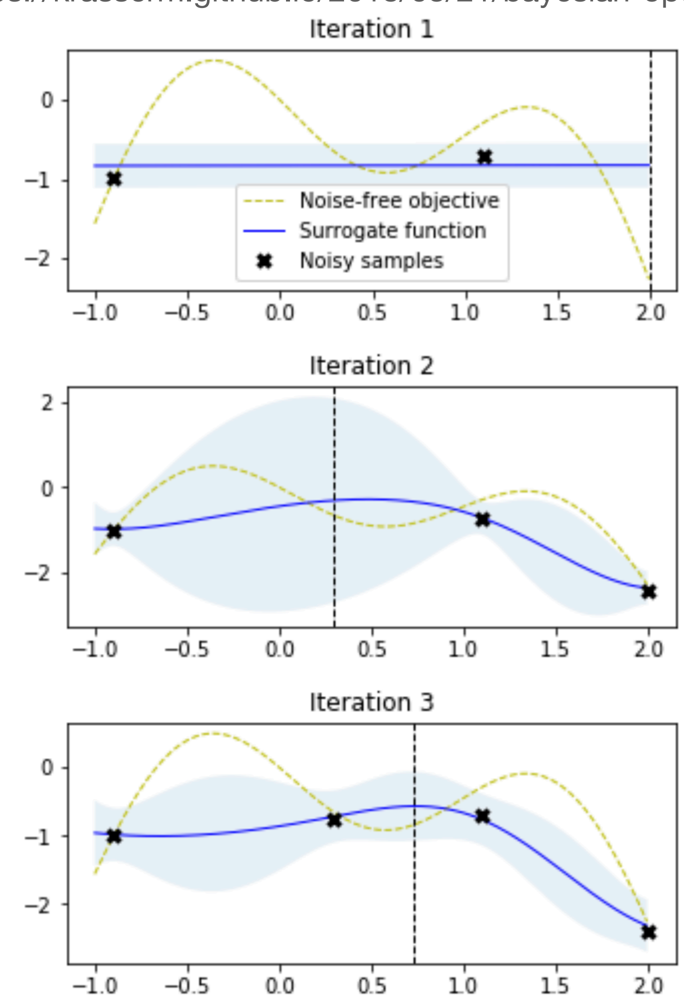
- Guess
- Grid Search
- Do small-scale experiments. Then "extrapolate"

1. Draw a line: **Scaling Law**

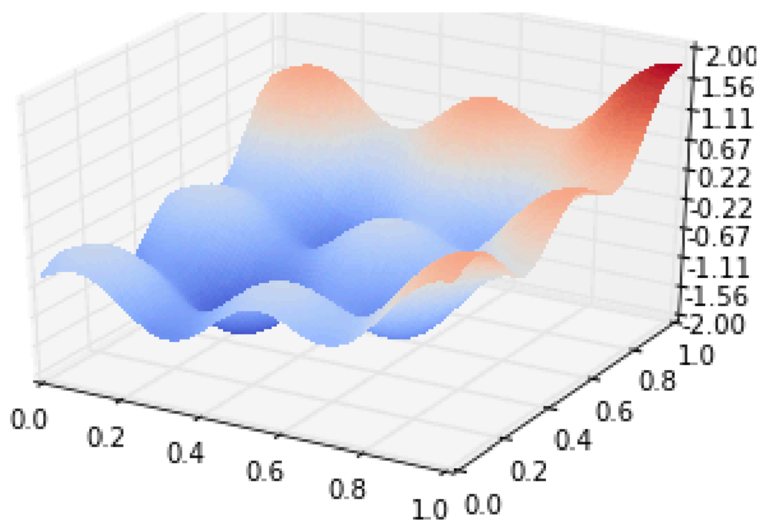
https://stanford-cs324.github.io/winter2022/assets/pdfs/Scaling%20laws%20pdf.pdf

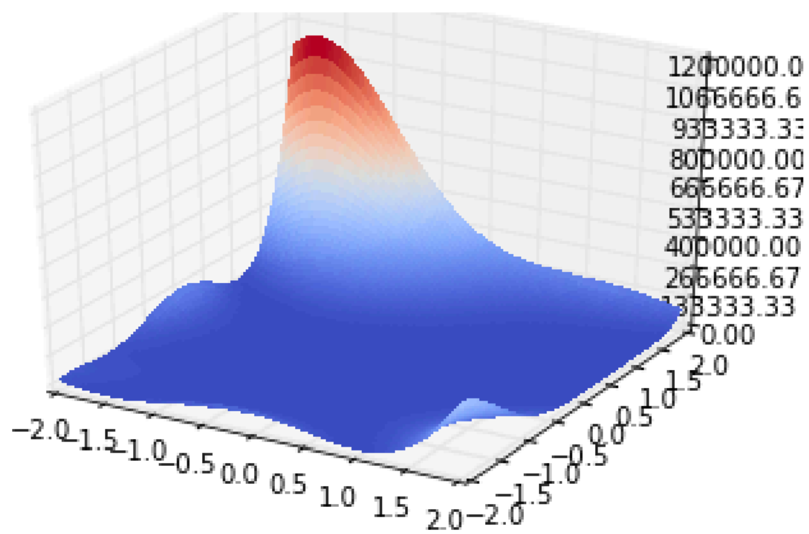
2. (Multi-fidelity) Bayesian Optimization
3. Update hyperparameters (specifically data) online

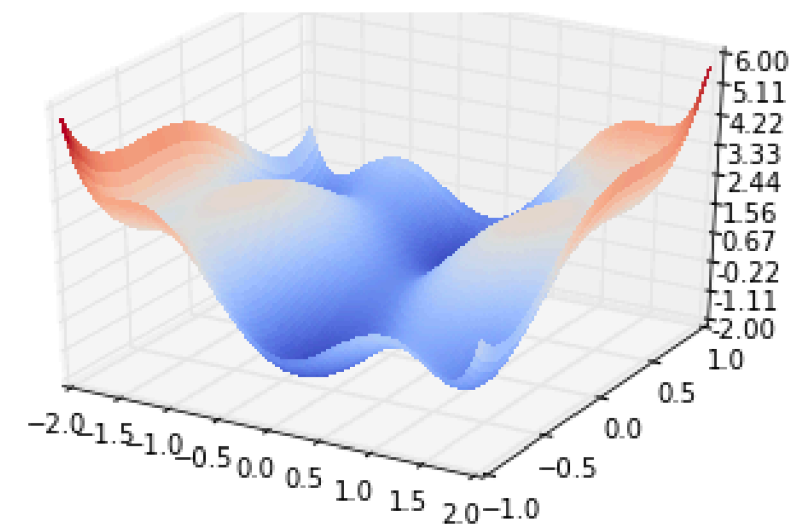
https://krasserm.github.io/2018/03/21/bayesian-optimization/



Hyperparameter Optimization







$$x^* = \arg\min f(x)$$

- f is unknown (performance of data)
- x is hyperparameter (data mixture, optimizer, learning rate etc.)
- No gradients
- Evaluation of f is expensive

Bayesian Optimization - Bayesian Statistics

$$\underbrace{P(\theta|D)}_{\text{Posterior}} = \frac{\overbrace{P(D|\theta)}^{\text{Likelihood}}\,\overbrace{P(\theta)}^{\text{Prior}}}{\underbrace{P(D)}_{\text{Evidence}}}$$

D data $\quad$ θ something we do not observe

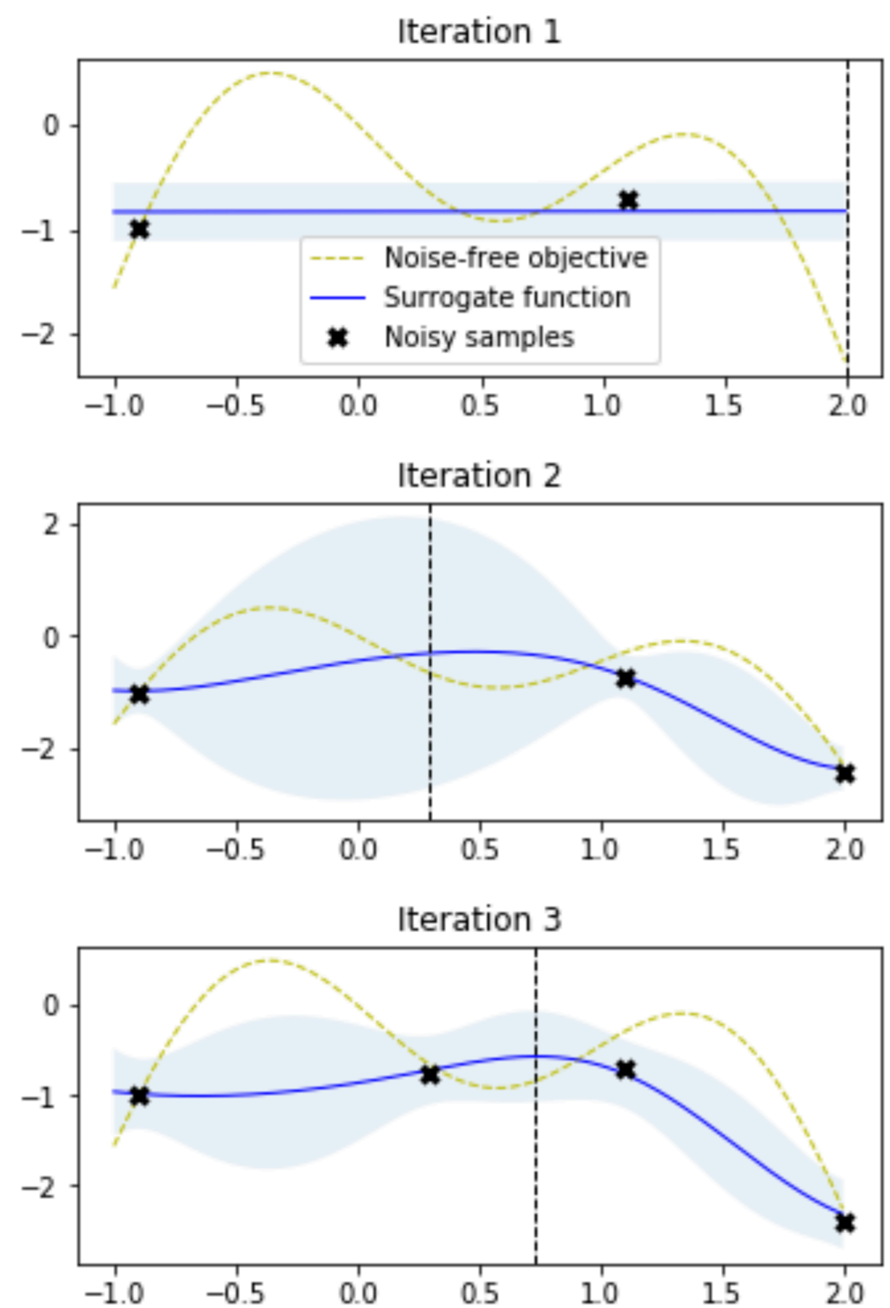
$\boxed{P(\theta)}$ initial belief of distribution of what we don’t know

$\boxed{P(D|\theta)}$ the data generative process

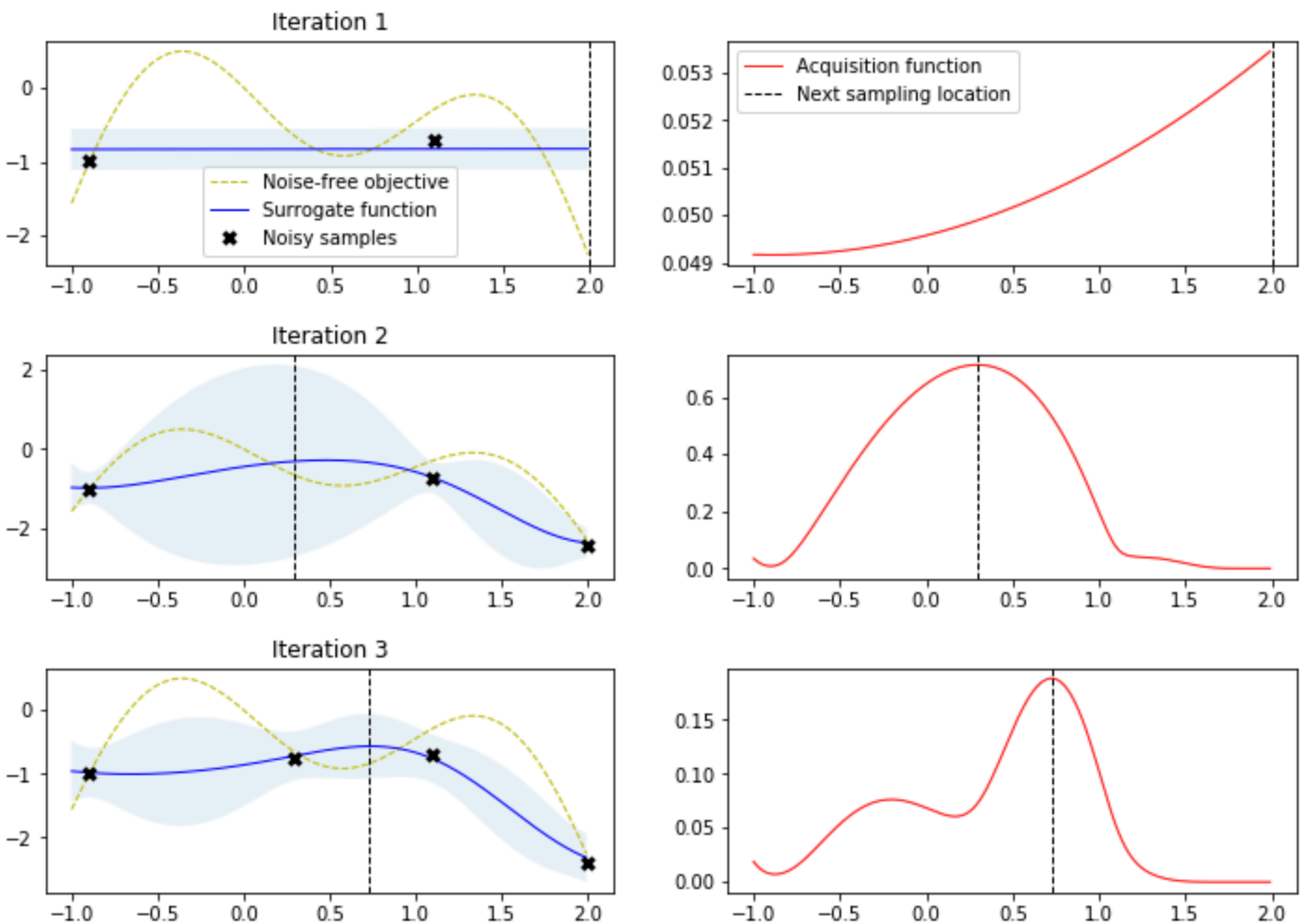
Bayesian Optimization - Distribution of Function Given Data

D data θ functions

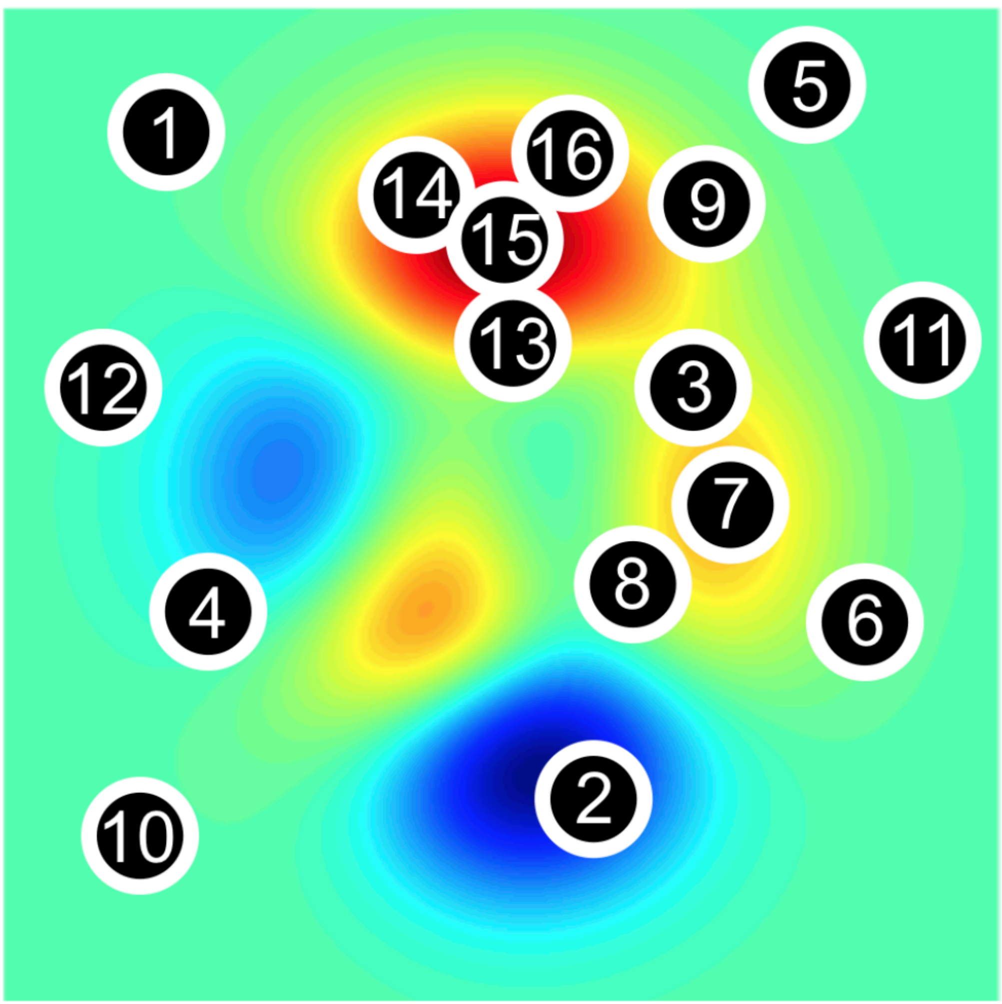
$P(\theta)$ $P(D|\theta)$ determined by Gaussian process



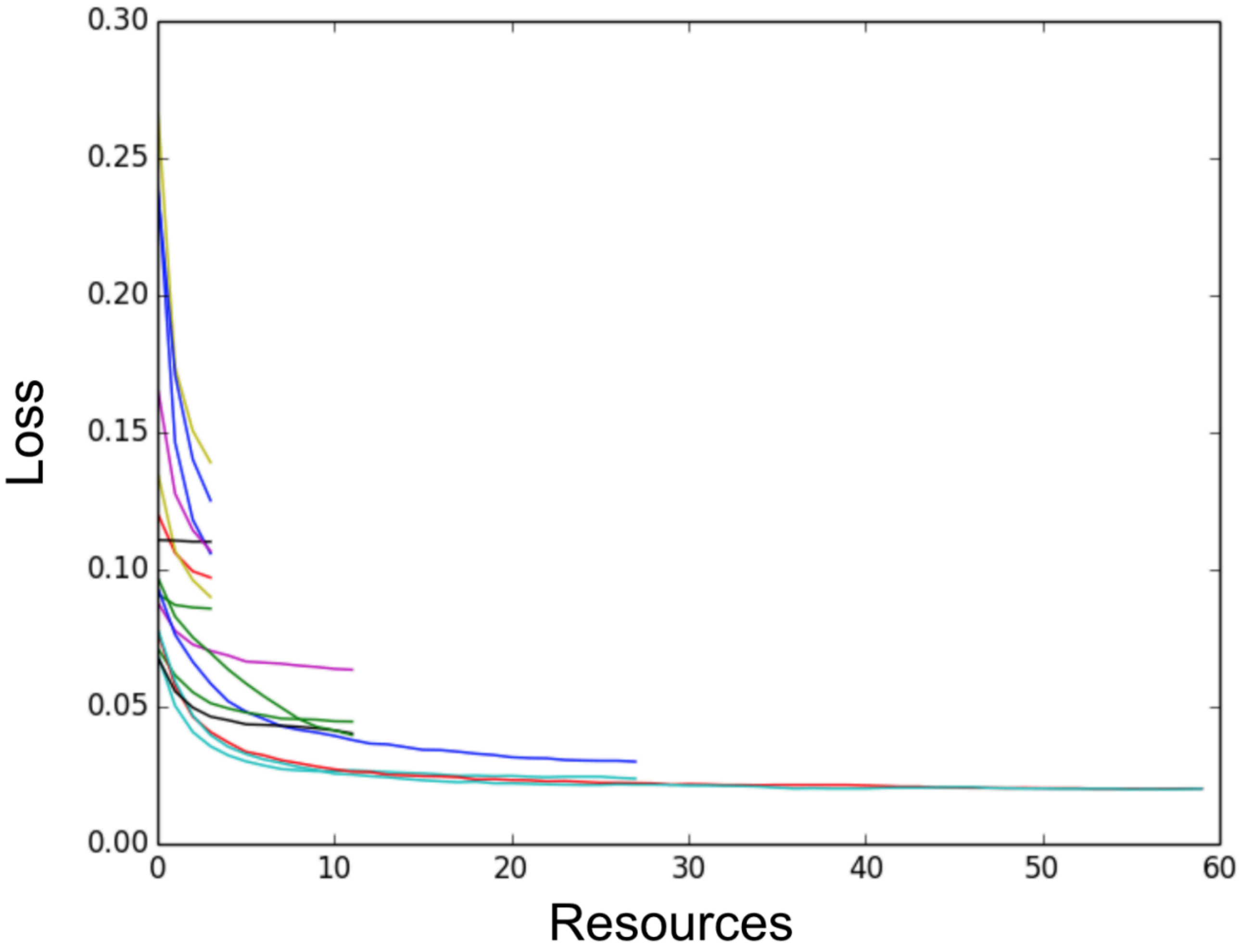
Bayesian Optimization - What's the next point I should choose?



Separate Idea: Multi-fidelity



(a) Configuration Selection



(b) Configuration Evaluation

Multi-fidelity Bayesian Optimization

$$x^* = \arg\min f(x, s^*)$$

e.g. I want to train the best model for s^*=100 steps

I can train any x with s < s^* steps with cost c(s)

Multi-fidelity Multi-scale Bayesian Optimization

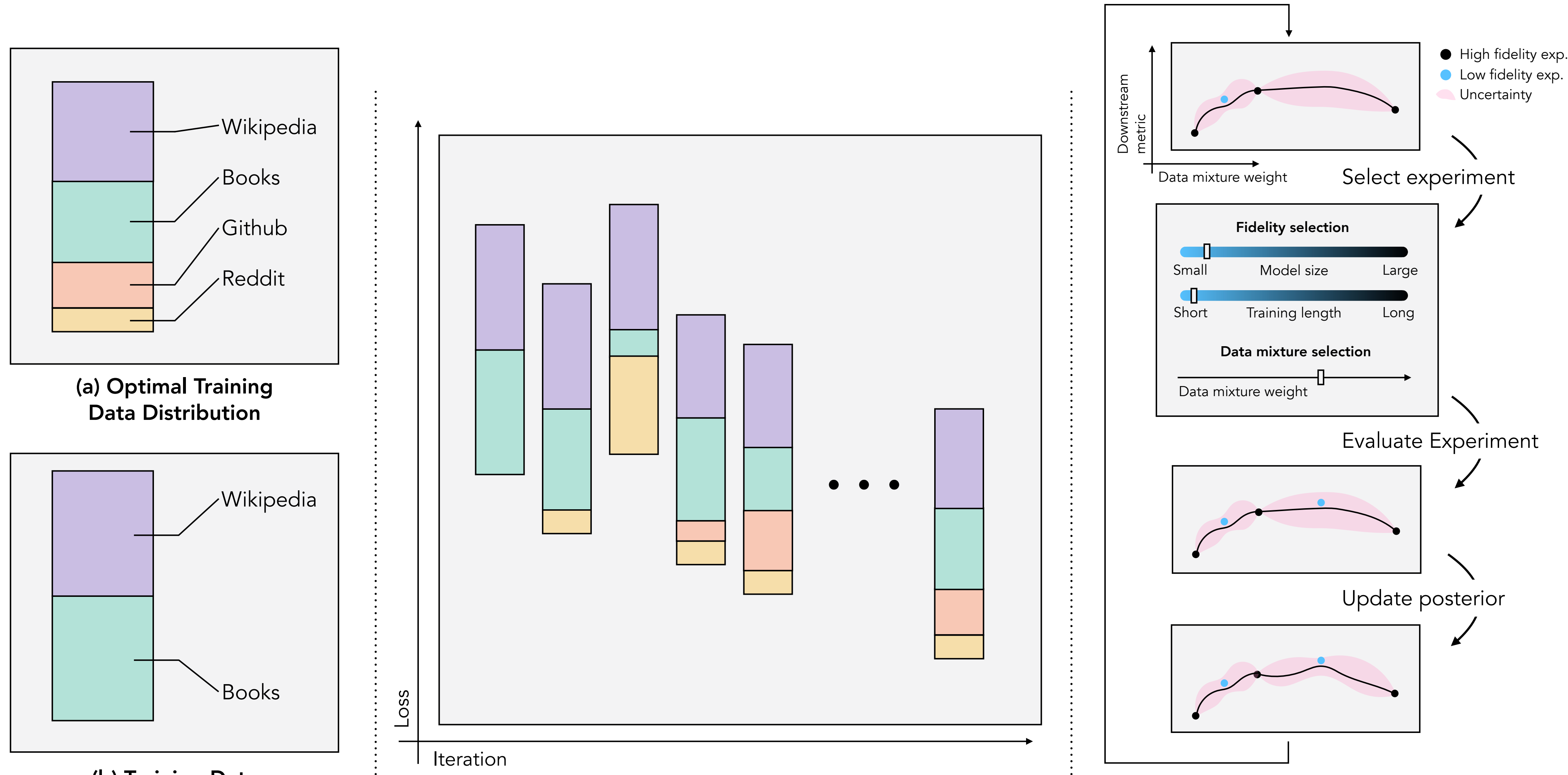
$$x^* = \arg\min f(x, s^*, m^*)$$

e.g. I want to train the best 1B model for s^*=100 steps

I can train any x with s < s^* steps and any smaller model m < 1B

The cost is c(s, m)

Multi-fidelity Multi-scale Bayesian Optimization (Data Mixing)



(a) Optimal Training Data Distribution

(b) Training Data Distribution with Heuristics-based Filtering

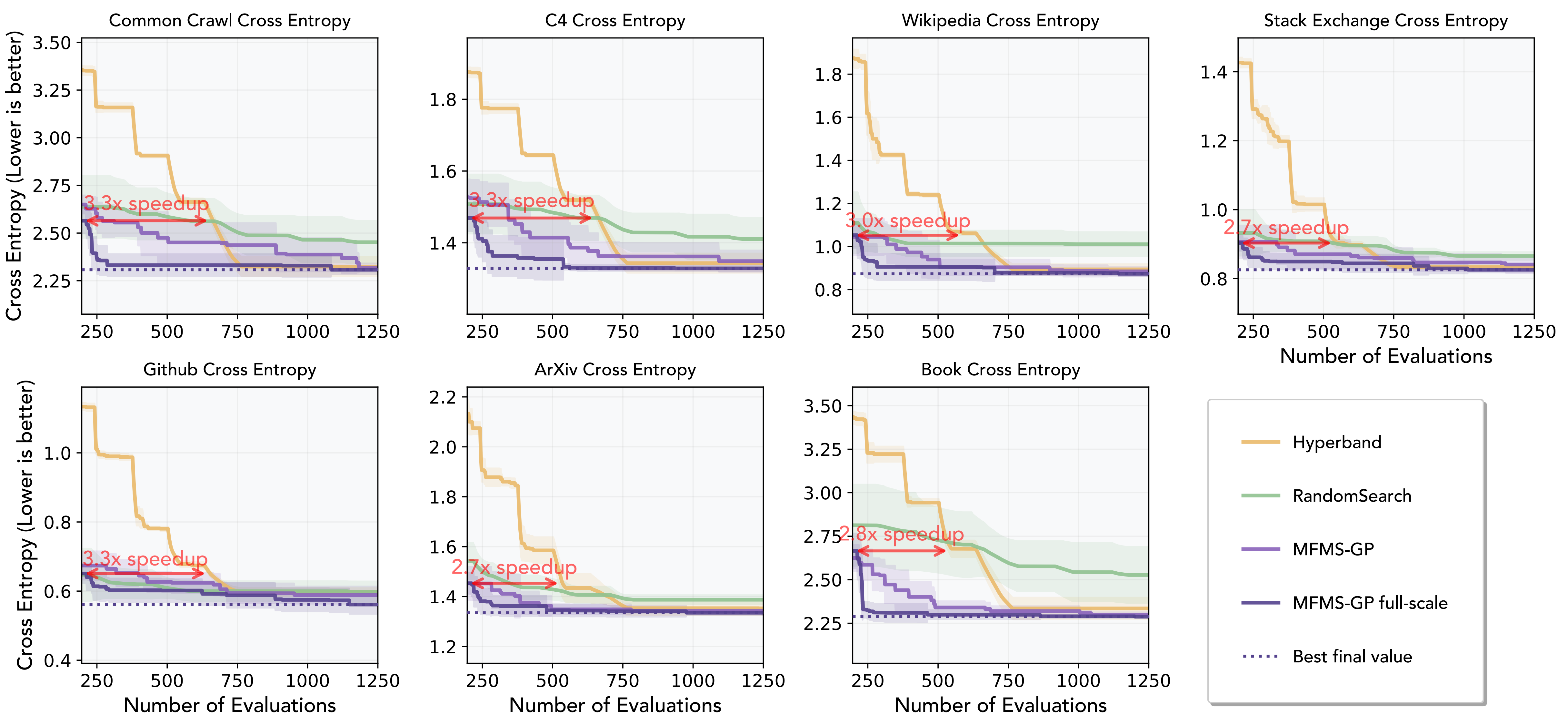
(c) Data mixture as an adaptive optimization problem

(d) Our multi-scale multi-fidelity bayesian optimization setup

At any iteration, I can train with data mixture x, model scale m, steps s

Multi-fidelity Multi-scale Bayesian Optimization (Data Mixing)

Extremely simple implementation of GP (using EI) works

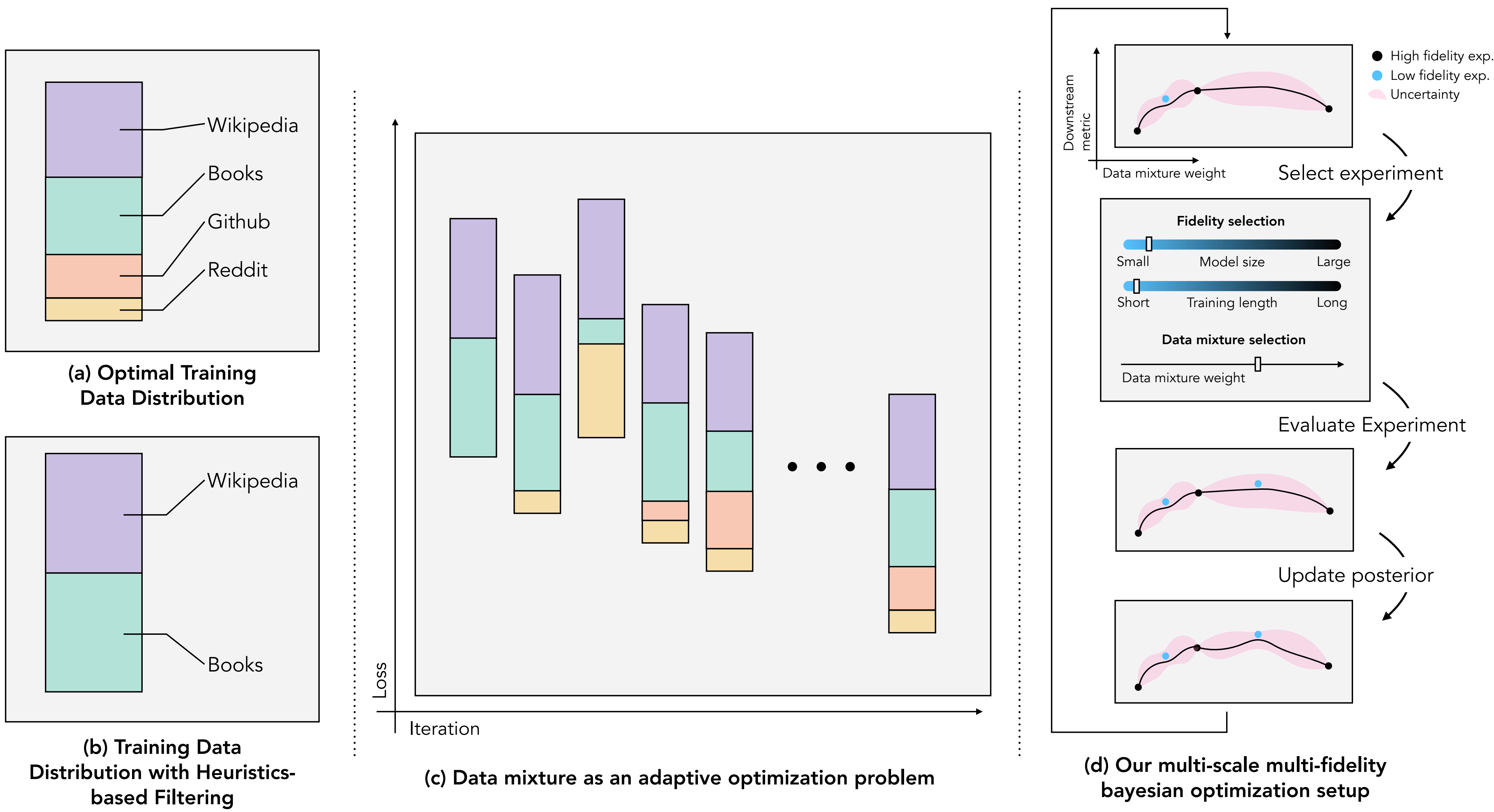


Recap on Bayesopt

- Instead of believing in a line, we can use Bayesian optimization
- Data mixing coefficients (could) transfer better from smaller scale experiments
- Difference between model scale and steps provide rich structure for future work

Something we've been missing: change data mixture on the fly

Akin to Curriculum learning: learn easy then hard

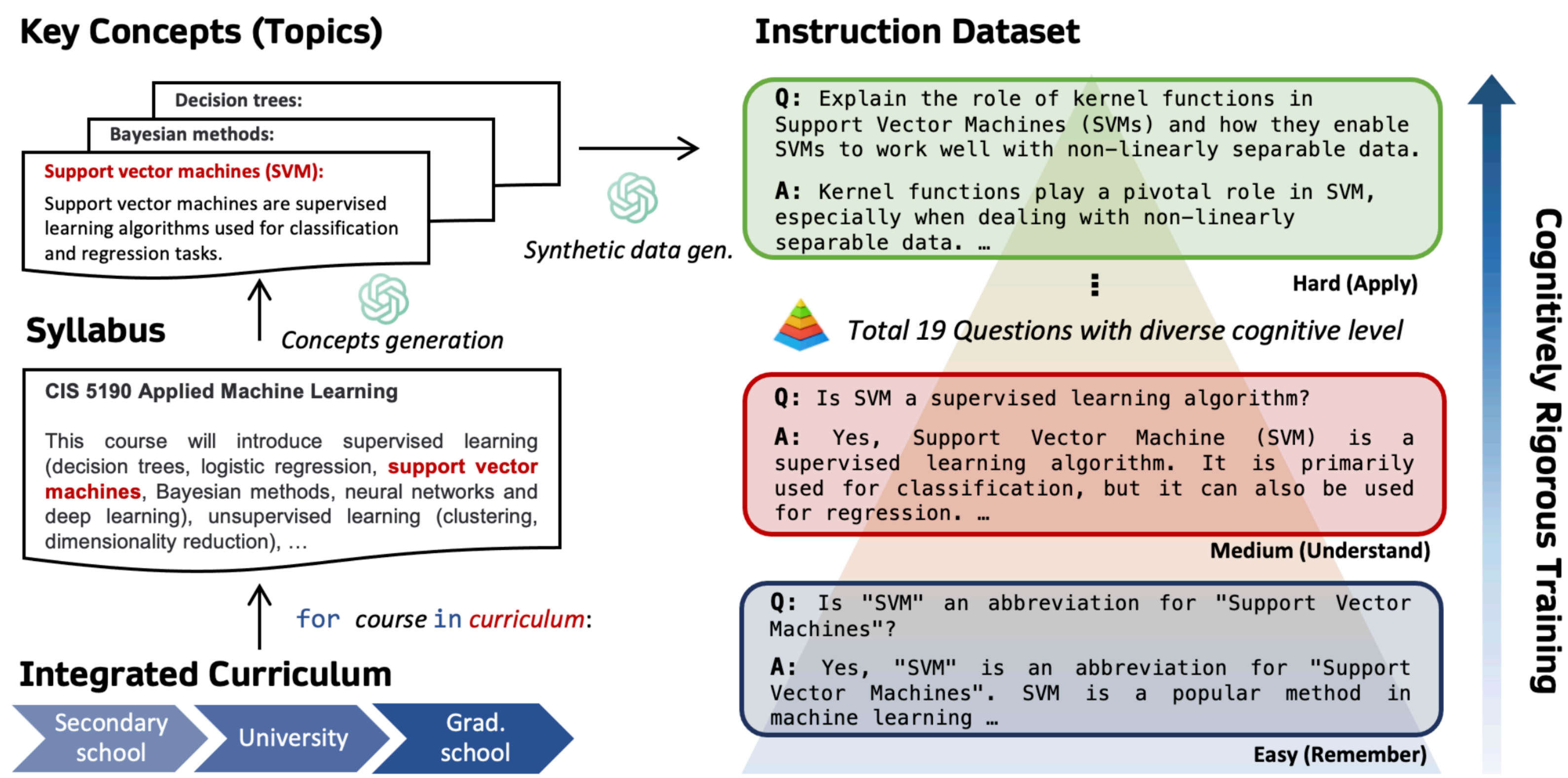


Project w/ Andrew, Tony

Some heuristic curriculum training

Lee et al., 2023, https://arxiv.org/pdf/2310.09518

Human Curriculum



Sequence-length

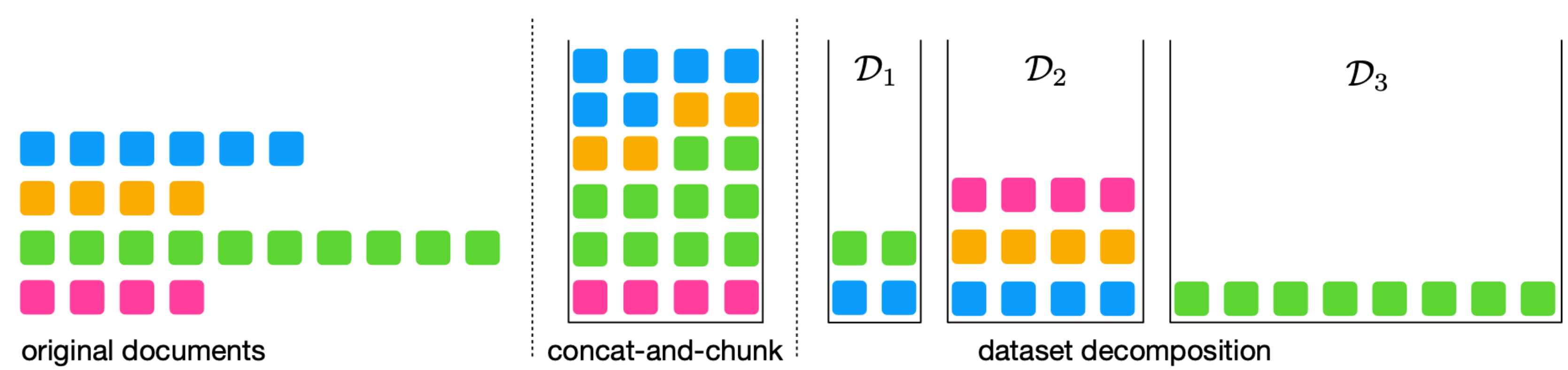


Figure 2: Each cell in the figure represents a token. **Left:** Original documents with variable lengths. **Middle:** Concat-and-chunk baseline to form sequences with a fixed target length (here $= 4$). **Right:** Dataset decomposition method with $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_3$ buckets .

Pouransari et al., 2024, https://arxiv.org/pdf/2405.13226

Training-dynamic-based approaches

Train more on higher loss

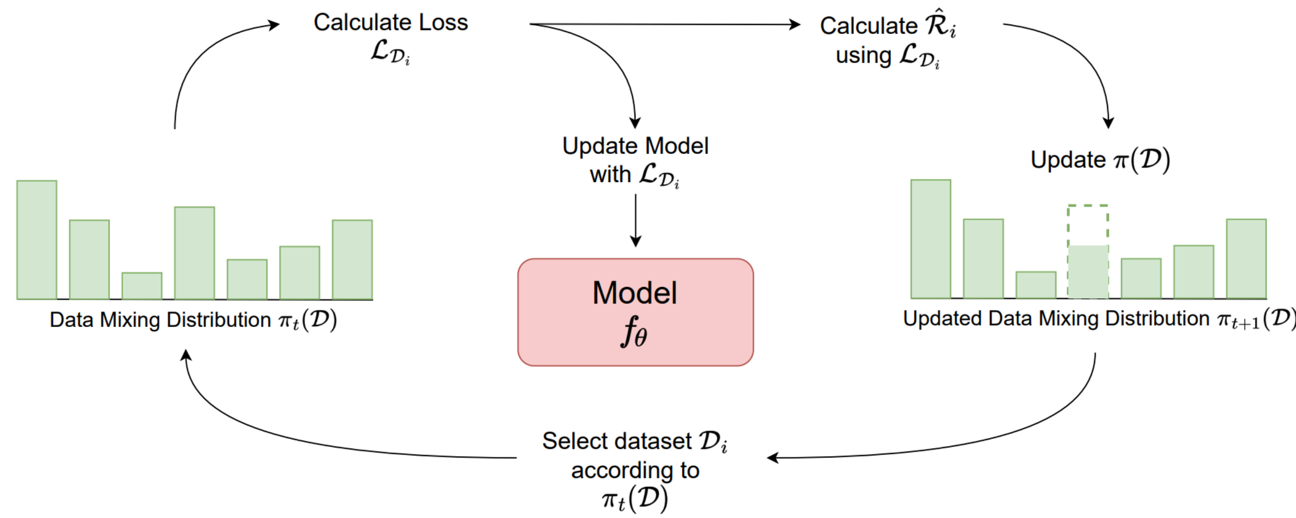


Figure 2: **Overview of Online Data Mixing (ODM) as a multi-armed bandit**. At each iteration of training, t, a dataset $\mathcal{D}_i$ is sampled according to the data mixing distribution π. The loss $\mathcal{L}_{\mathcal{D}_i}$ is calculated w.r.t the model f_θ and subsequently used to update the model. Simultaneously, a reward $\hat{\mathcal{R}}_i$ is calculated and used to update π for the next iteration, $i + 1$.

Albalak et al., 2024, https://arxiv.org/pdf/2312.02406

Use (domain-specific) Scaling Law to tell which is more learnable

$$\frac{d\widehat{\mathcal{L}}_k(n)}{dn} = \frac{-\alpha_k \beta_k n^{-\alpha_k}}{n} = -\frac{1}{n} \underset{\text{Learning speed}}{\alpha_k} \underset{\text{Reducible loss}}{\left(\widehat{\mathcal{L}}_k(n) - \varepsilon_k\right)} .$$

Jiang et al., 2024, https://arxiv.org/pdf/2410.11820

Training-dynamic-based approaches

$$U^{(t)}(S; z^{(\mathrm{val})}) := \ell(w_t, z^{(\mathrm{val})}) - \ell(\widetilde{w}_{t+1}(S), z^{(\mathrm{val})})$$

(which data improves validation loss the most)

Expensive to compute.

Applicable only for selecting a mini batch from a batch

Project w/ Mohamed, Tao & Xincan

Recap on adaptive data mixing:

- Ongoing research. Extremely simple heuristics at the moment
- Likely a lot to do here

Data Categories

Coarse categories

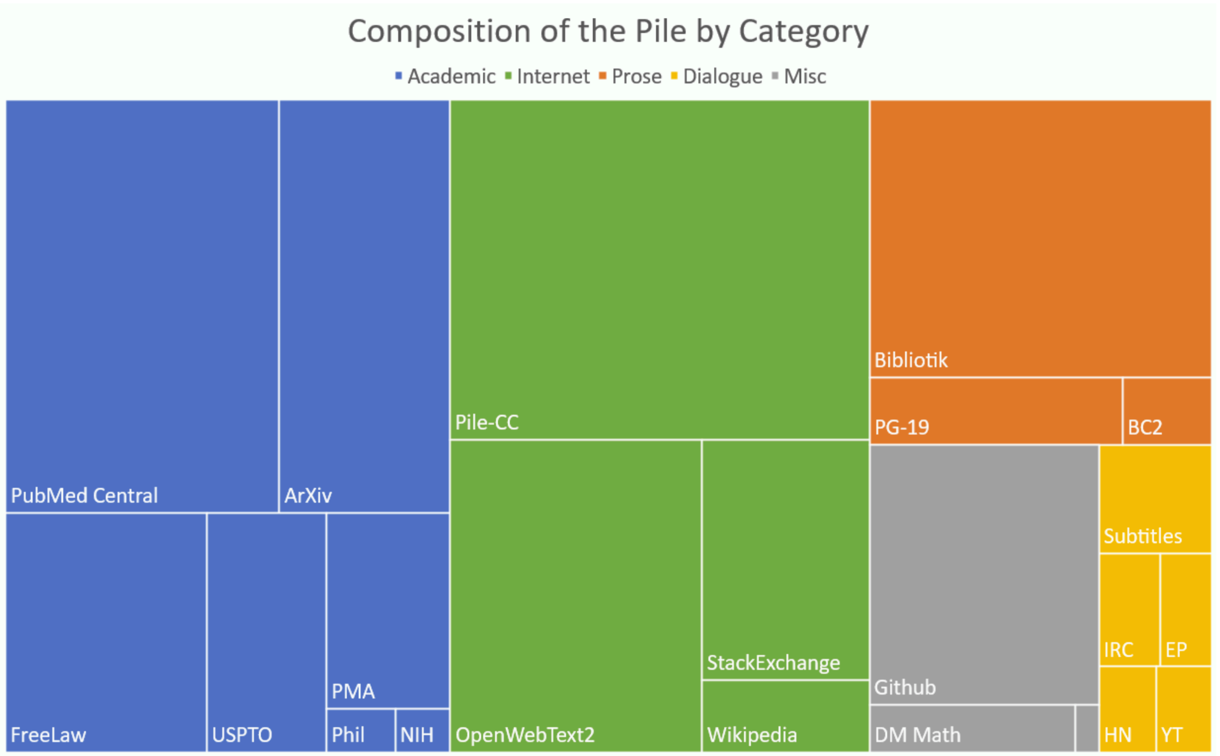


Figure 1: Treemap of Pile components by effective size.

Is there anything better?