Data selection 1

IMPROVING PRETRAINING DATA USING PERPLEXITY CORRELATIONS

By Tristan Thrush, Christopher Potts, Tatsunori Hashimoto

Setup

Goal: select pre-training data in order to optimize performance on a fixed benchmark

Often: train small model on a bunch of mixtures of domains (data sets), check performance. Computationally expensive!

Here: only use “observational data” to find optimal data mix, no LLM training required.

Input: publicly available models + data sets + their performance in benchmarks

Intuition

If for a fixed data set…

If a LLM has low loss on validation data -> high score on target benchmark

and

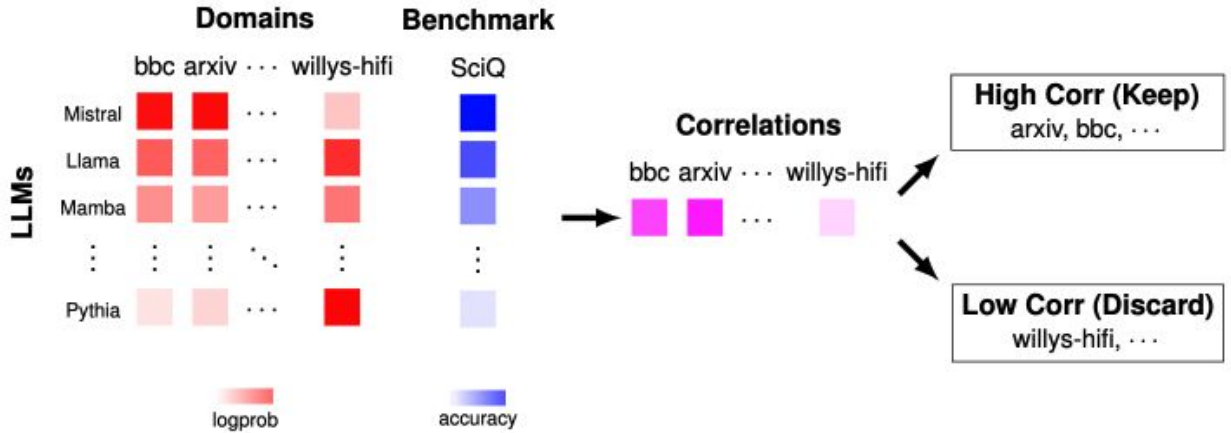
If a LLM has large loss on validation data -> low score on target benchmark

We would want to use this data set for pre-training (?)

Proposal

Run a regression of benchmark accuracies on domains (data set)

Each row is a LLM, each column is a domain (data set)



LLMs: 90 taken from Open LLM Leaderboard,
Domains: 9841 domains from sample-100B RedPajama V2

Seems weird, but at second glance it’s not that unusual

Similar ideas have appeared in the causal inference literature.

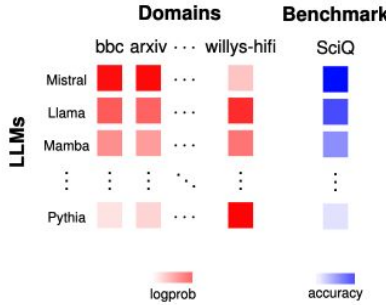
In the synthetic control approach, we regress a target data set (California) on several source data sets (Texas, Washington,...)

Columns: data sets (here: data sets)

Rows: pre-treatment variables (here: LLMs)

Entries: means of pre-treatment variables (here: log-likelihoods)

Target vector: mean of post-treatment outcome (here: accuracy)



Assumption

The perplexity-performance hypothesis. We formulate the task of predicting errors y_i from negative log-probabilities $\mathbf{x}_i$ as a single-index model (SIM),

$$y_i = f(\langle \boldsymbol{\theta}^*, \mathbf{x}_i \rangle + \epsilon_i) \tag{1}$$

where $f : \mathbb{R} \mapsto \mathbb{R}$ is some unknown monotonically increasing univariate function, ϵ_i is zero-mean noise which is independent of $\mathbf{x}$, and $\boldsymbol{\theta}^* \in \mathbb{R}^D$ are unknown weights over D domains.

Outcomes y_i: accuracy of LLM i on benchmark data set

Covariates x_{ij}: negative log-likelihood of LLM i on document j

Parameter theta: best prediction

This is just a low-dimensional single index model that we can learn

Assumption: single index model holds not only for log-likelihoods of trained models in leaderboard, but also for the log-likelihoods when training a new model

-> To optimize benchmark performance (left hand side), we can just optimize weighted log-likelihoods (right hand side)

Estimating theta (idea from previous papers)

If X follow a standard Gaussian distribution, using integration by parts

$$E[X_j Y] = E[X_j f(\langle \theta^*, X \rangle + \epsilon)] = E[\theta_j f'(\langle \theta^*, X \rangle + \epsilon)] = \theta_j^* C$$

where C>0. We are averaging over LLMs.

So, we can just compute the correlation between X and Y to estimate theta (up to a constant).

Estimation

The actual proposed algorithm avoids estimating f.

Distribution of loglikelihoods has heavy tails (solution: rank correlations, transformation)

Solutions theta may be negative and not sum up to one (solution: project theta on probability simplex - set of nonnegative vectors that sum up to one)

Algorithm 1 Perplexity Correlation Based Data Selection

Input: Benchmark error vector $\mathbf{y} \in [0,1]^N$, log-loss matrix normalized as bits-per-byte $\mathbf{X} \in \mathbb{R}_0^{+\,N \times D}$, available tokens per domain $\mathbf{a} \in \mathbb{N}^D$, and pretraining token target $b \in \mathbb{N}$.
Output: Target token counts per domain $\mathbf{t} \in \mathbb{N}_0^D$, a fastText classifier to filter pretraining data.
Initialize: $\boldsymbol{\gamma} \leftarrow \mathbf{0} \in \mathbb{R}^D$, $\mathbf{t} \leftarrow [0 \ldots] \in \mathbb{N}_0^D$, counter $\leftarrow 0$.
$\mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_N \leftarrow \mathrm{rank}(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_N)$ ▷ 1. Compute the $\boldsymbol{\gamma}$ correlation coefficient
for $i, j \in 0$ **to** N **do**
    $\boldsymbol{\gamma} \leftarrow \boldsymbol{\gamma} + \mathrm{sign}(y_i - y_j) \cdot (\mathbf{r}_i - \mathbf{r}_j)$
for $i \in$**ArgSort**($\boldsymbol{\gamma}$, descending=True) **do** ▷ 2. Select most to least correlated domains
    $t_i \leftarrow \min(a_i, b - \text{counter})$
    counter $\leftarrow$ counter $+ a_i$
    if counter $\geq b$ **then**
        Break
classifier = trainFastText(positive $= 1_{t>0}$, negative $= 1_{t=0}$)
Return $\mathbf{t}$, classifier

In practice…

1) No weights (domain included yes/no)
2) Train FastText classifier on selected domains to be able to select documents, not just domains

Evaluation

Fix pre-training architecture, change data sets.

Pre-training on Pythia 160M LLM configuration from Biderman et al. (2023)

Hyperparameters including learning rate, weight decay, and warmup are optimized to minimize loss on the uniform sampling (no selection algorithm) baseline.

Benchmarks

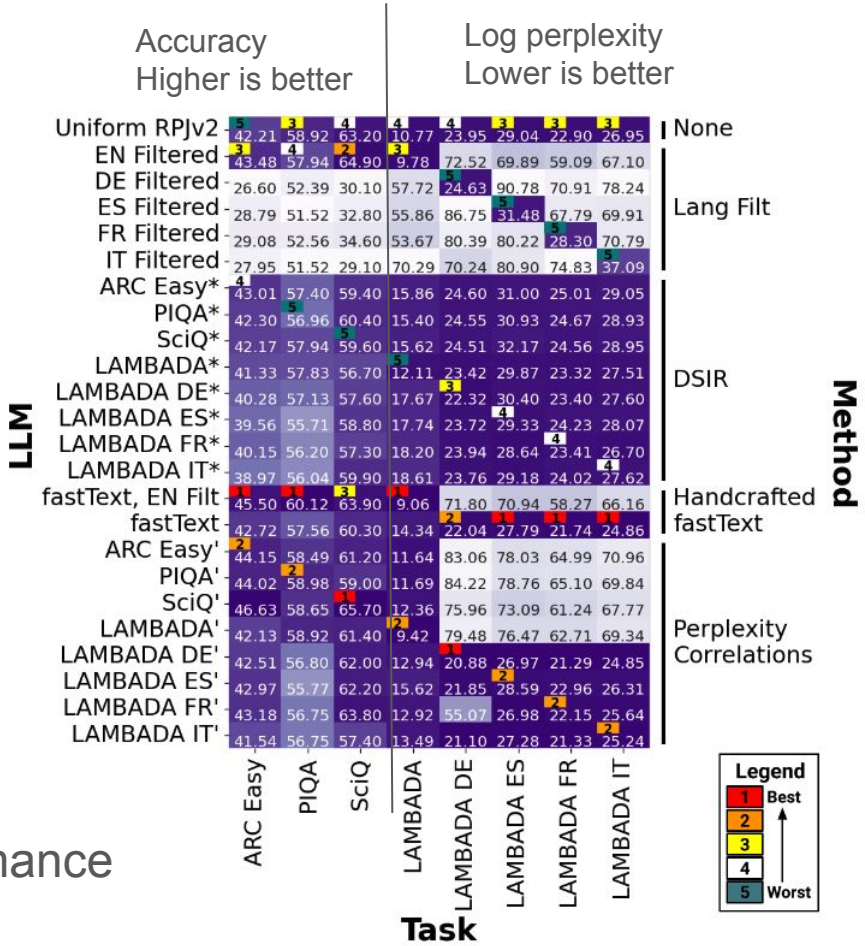
SciQ - scientific questions

ARC easy - grade school level questions

PIQA - commonsense reasoning, physics questions

Lambada - recover missing word from corpus of text (multilingual)

Data selection method



equal weights

language filter

Domain selection via importance sampling

you get performance

TLDR: fastText is great, perplexity pretty good

Aside: what is fastText (SOTA)?

https://huggingface.co/mlfoundations/fasttext-oh-eli5

Text classifier

Filter data by sorting pages by “high quality” score, until we had reached 3.2B unique tokens.

Comments

- Should we do data selection at the domain level or document level (or both?)
- Can we improve performance by using weights (instead of include data yes/no?). The greedy strategy should be suboptimal
- We should have a principled approach for choosing between data collection strategies based on what type of distribution shift we have
- We probably want to find weights that do well universally across tasks
- Rank correlations are not very satisfying since they hide whether the changes are large

Awesome!

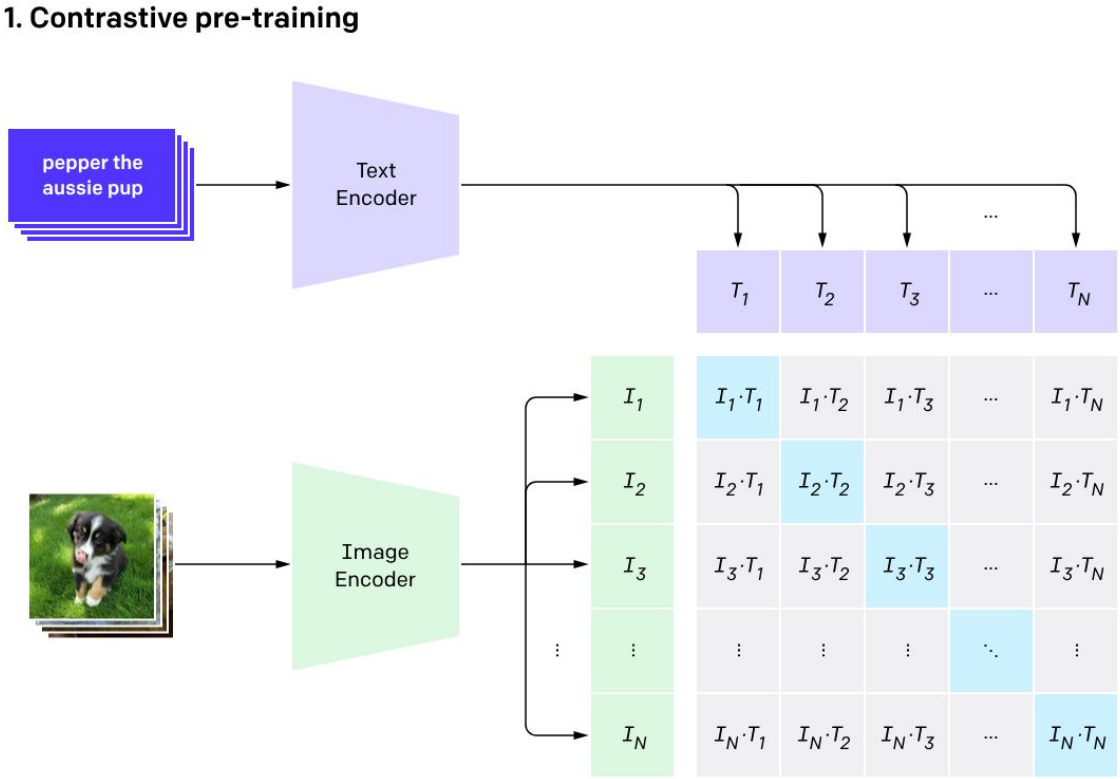
(...) we use this paper also as a preregistration for further pretraining experiments using different data sources and evaluation benchmarks at the 1.4B model scale.

We commit to updating the arXiv version of our manuscript with both positive and negative results on these preregistered validation experiments.

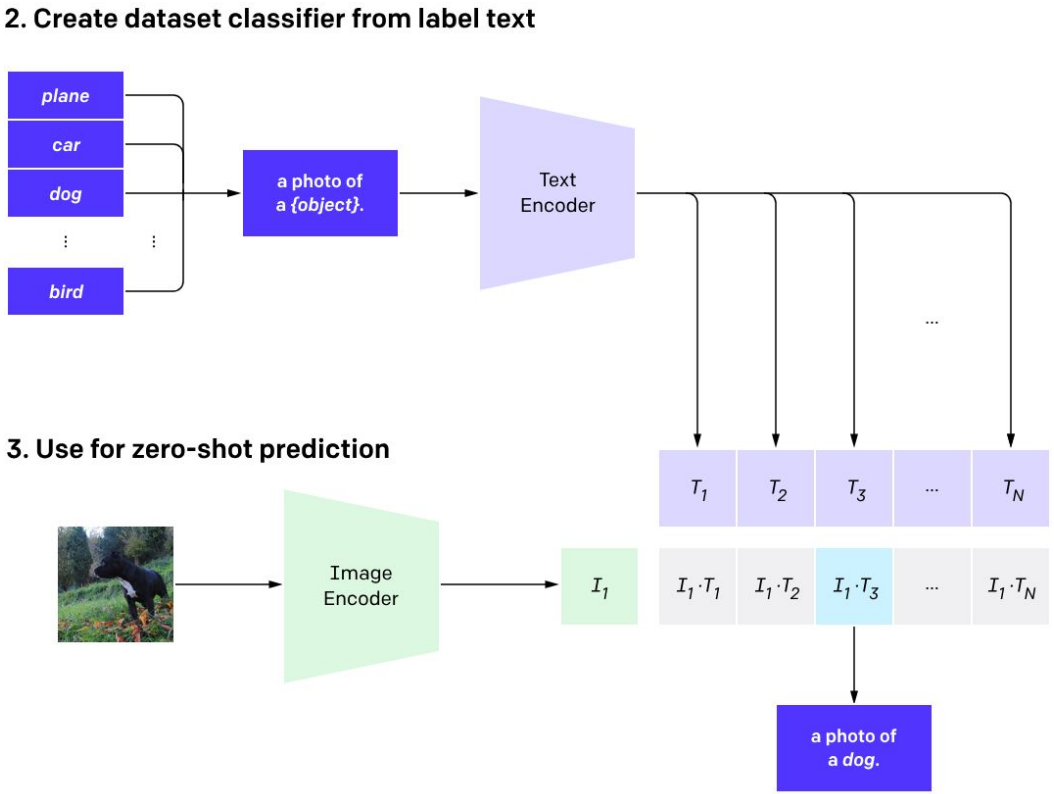
SCALING LAWS FOR DATA FILTERING – DATA CURATION CAN NOT BE COMPUTE AGNOSTIC

Sachin Goyal, Pratyush Maini
Zachary Lipton, Aditi Raghunathan, Zico Kolter

CLIP (Contrastive Language-Image Pretraining)



CLIP (Contrastive Language-Image Pretraining)



Setting

128M image-caption pairs, taken from DataComp (Gadre et al, 2023) medium

Split into 10 buckets of 12.8M images either by

- CLIP Score - cosine similarity between text and image embeddings
- T-MARS (Maini et al, 2023) - CLIP score after masking out text in images

Setting

128M image-caption pairs, taken from DataComp (Gadre et al, 2023) medium

Split into 10 buckets of 12.8M images either by

- CLIP Score - cosine similarity between text and image embeddings
- T-MARS (Maini et al, 2023) - CLIP score after masking out text in images

For a sufficiently high compute budget, there is an inherent tradeoff between seeing new samples or repeating existing high quality samples

Optimal filtering depends on compute budget

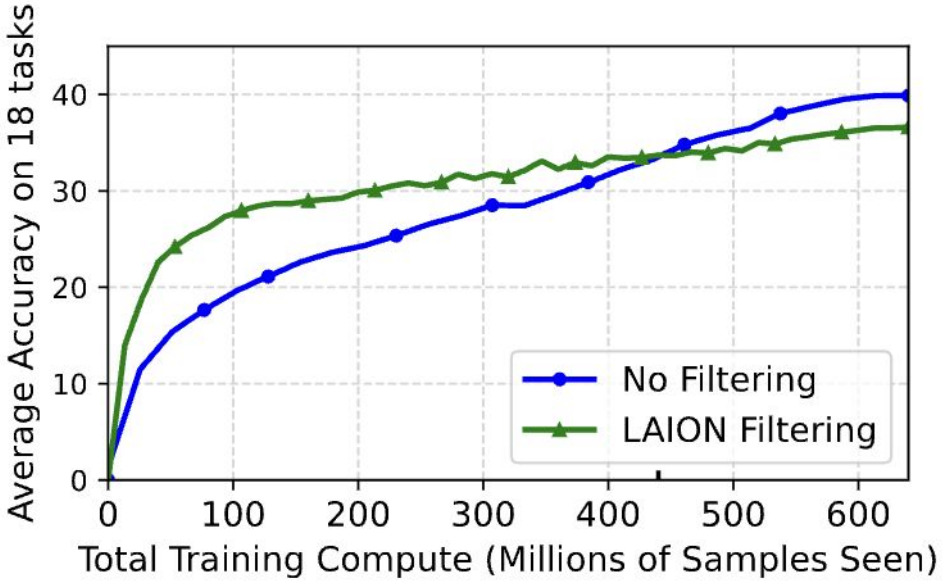


Figure 2. Given an initial data pool of 128M samples, we train ViT-B/32 CLIP models for a total of 640M samples. As we train for longer, the accuracy gains using the LAION-filtering subset that filters the common crawl to 10% of its initial size plateau. Surprisingly, even no-filtering of the common crawl is better than the popular LAION-filtering after seeing more than 450M samples.

Optimal filtering depends on compute budget

As you scale the compute budget, diversity becomes more important than quality

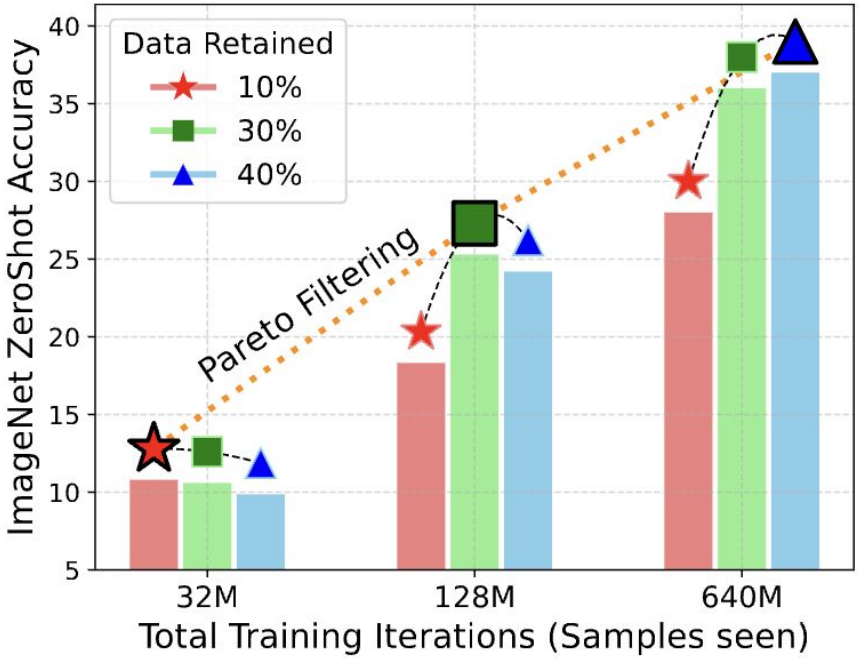


Figure 3. We vary the CLIP filtering threshold after ranking the data by their metric. While the original paper proposed retaining 30% of the data, our results show that depending on the ratio of compute to data pool size, we must adaptively make the filtering less (or more) aggressive to account for the diminishing utility of good data with repetitions. Results are presented on an average of 18 visual understanding tasks with a global data pool size of 128M samples, and varying compute scales.

Can we predict the downstream performance?

Scaling laws for a single bucket, single epoch

y = downstream performance

n = number of samples

a, d = problem-dependent constants

b = sample utility

$$y = a \cdot n^b + d$$

Scaling laws for a single bucket, single epoch

y = downstream performance

n = number of samples

a, d = problem-dependent constants

b = sample utility

$$y = a \cdot n^b + d$$

$$\frac{dy}{dn} = b \cdot \frac{y}{n}$$

Scaling laws for a single bucket, multiple epochs

$$y = a \cdot n^b + d \qquad \text{(1 epoch)}$$

Scaling laws for a single bucket, multiple epochs

$$y = a \cdot n^b + d \qquad \text{(1 epoch)}$$

$$y = a \cdot n^b \left(\frac{2n}{n}\right)^b \dots \left(\frac{kn}{(k-1)n}\right)^b + d \qquad (k \text{ epochs with no utility decrease})$$

Scaling laws for a single bucket, multiple epochs

$$y = a \cdot n^b + d \qquad \text{(1 epoch)}$$

$$y = a \cdot n^b \left(\frac{2n}{n}\right)^b \dots \left(\frac{kn}{(k-1)n}\right)^b + d \qquad (k \text{ epochs with no utility decrease})$$

$$y = a \cdot n^{b^{(1)}} \left(\frac{2n}{n}\right)^{b^{(2)}} \dots \left(\frac{kn}{(k-1)n}\right)^{b^{(k)}} + d \qquad (k \text{ epochs with different utilities})$$

Scaling laws for a single bucket, multiple epochs

$$y = a \cdot n^b + d \qquad \text{(1 epoch)}$$

$$y = a \cdot n^b \left(\frac{2n}{n}\right)^b \dots \left(\frac{kn}{(k-1)n}\right)^b + d \qquad (k \text{ epochs with no utility decrease})$$

$$y = a \cdot n^{b^{(1)}} \left(\frac{2n}{n}\right)^{b^{(2)}} \dots \left(\frac{kn}{(k-1)n}\right)^{b^{(k)}} + d \qquad (k \text{ epochs with different utilities})$$

$$y = a \cdot n^{b \cdot \delta^0} \left(\frac{2n}{n}\right)^{b \cdot \delta^1} \dots \left(\frac{kn}{(k-1)n}\right)^{b \cdot \delta^{k-1}} + d \qquad \text{(assume utility exponentially decays)}$$

$$\boxed{b^{(k)} = b \cdot \delta^{k-1}}$$

Scaling laws for multiple buckets, single epoch

Initialize model state at performance y_0, trained on n_0 samples

Make a mixed dataset of two samples with utility and decay (b_1, δ_1) and (b_2, δ_2)

What is the effective utility of a sample in this new dataset?

Scaling laws for multiple buckets, single epoch

Initialize model state at performance y_0, trained on n_0 samples

Make a mixed dataset of two samples with utility and decay (b_1, δ_1) and (b_2, δ_2)

What is the effective utility of a sample in this new dataset?

$$\frac{y_2}{y_0} = \left(\frac{n_0 + 1}{n_0}\right)^{b_1} \left(\frac{n_0 + 2}{n_0 + 1}\right)^{b_2}$$

(two gradient steps)

Scaling laws for multiple buckets, single epoch

Initialize model state at performance y_0, trained on n_0 samples

Make a mixed dataset of two samples with utility and decay (b_1, δ_1) and (b_2, δ_2)

What is the effective utility of a sample in this new dataset?

$$\frac{y_2}{y_0} = \left(\frac{n_0+1}{n_0}\right)^{b_1} \left(\frac{n_0+2}{n_0+1}\right)^{b_2} \qquad \text{(two gradient steps)}$$

$$\approx \left(\frac{n_0+1}{n_0}\right)^{b_1} \left(\frac{n_0+1}{n_0}\right)^{b_2} \qquad \text{(assume large } n_0\text{)}$$

Scaling laws for multiple buckets, single epoch

Initialize model state at performance y_0, trained on n_0 samples

Make a mixed dataset of two samples with utility and decay (b_1, δ_1) and (b_2, δ_2)

What is the effective utility of a sample in this new dataset?

$$\frac{y_2}{y_0} = \left(\frac{n_0+1}{n_0}\right)^{b_1} \left(\frac{n_0+2}{n_0+1}\right)^{b_2} \qquad \text{(two gradient steps)}$$

$$\approx \left(\frac{n_0+1}{n_0}\right)^{b_1} \left(\frac{n_0+1}{n_0}\right)^{b_2} \qquad \text{(assume large } n_0\text{)}$$

$$= \left(\frac{n_0+1}{n_0}\right)^{b_1+b_2} \qquad \text{(collapse into single term)}$$

Scaling laws for multiple buckets, single epoch

Initialize model state at performance y_0, trained on n_0 samples

Make a mixed dataset of two samples with utility and decay (b_1, δ_1) and (b_2, δ_2)

What is the effective utility of a sample in this new dataset?

$$\frac{y_2}{y_0} = \left(\frac{n_0+1}{n_0}\right)^{b_1}\left(\frac{n_0+2}{n_0+1}\right)^{b_2} \qquad \text{(two gradient steps)}$$

$$\approx \left(\frac{n_0+1}{n_0}\right)^{b_1}\left(\frac{n_0+1}{n_0}\right)^{b_2} \qquad \text{(assume large } n_0\text{)}$$

$$= \left(\frac{n_0+1}{n_0}\right)^{b_1+b_2} \qquad \text{(collapse into single term)}$$

$$\approx \left(\frac{n_0+2}{n_0}\right)^{\frac{b_1+b_2}{2}} \qquad \text{(split back into two data points)}$$

Scaling laws for multiple buckets, single epoch

Initialize model state at performance y_0, trained on n_0 samples

Make a mixed dataset of two samples with utility and decay (b_1, δ_1) and (b_2, δ_2)

What is the effective utility of a sample in this new dataset?

$$\frac{y_2}{y_0} = \left(\frac{n_0+1}{n_0}\right)^{b_1}\left(\frac{n_0+2}{n_0+1}\right)^{b_2} \qquad \text{(two gradient steps)}$$

$$\approx \left(\frac{n_0+1}{n_0}\right)^{b_1}\left(\frac{n_0+1}{n_0}\right)^{b_2} \qquad \text{(assume large } n_0\text{)}$$

$$= \left(\frac{n_0+1}{n_0}\right)^{b_1+b_2} \qquad \text{(collapse into single term)}$$

$$\approx \left(\frac{n_0+2}{n_0}\right)^{\frac{b_1+b_2}{2}} \qquad \text{(split back into two data points)}$$

$$\boxed{b_{\text{eff}} = \frac{b_1+b_2}{2}}$$

Scaling laws for multiple buckets, multiple epochs

$$b_{\text{eff}} = \frac{b_1 + b_2}{2} \qquad \text{(two buckets, one epoch)}$$

Scaling laws for multiple buckets, multiple epochs

$$b_{\text{eff}} = \frac{b_1 + b_2}{2} \qquad \text{(two buckets, one epoch)}$$

$$b_{\text{eff}} = \frac{1}{p} \sum_{i \in [p]} b_i \qquad (p \text{ buckets, one epoch})$$

Scaling laws for multiple buckets, multiple epochs

$$b_{\text{eff}} = \frac{b_1 + b_2}{2} \qquad \text{(two buckets, one epoch)}$$

$$b_{\text{eff}} = \frac{1}{p} \sum_{i \in [p]} b_i \qquad (p \text{ buckets, one epoch})$$

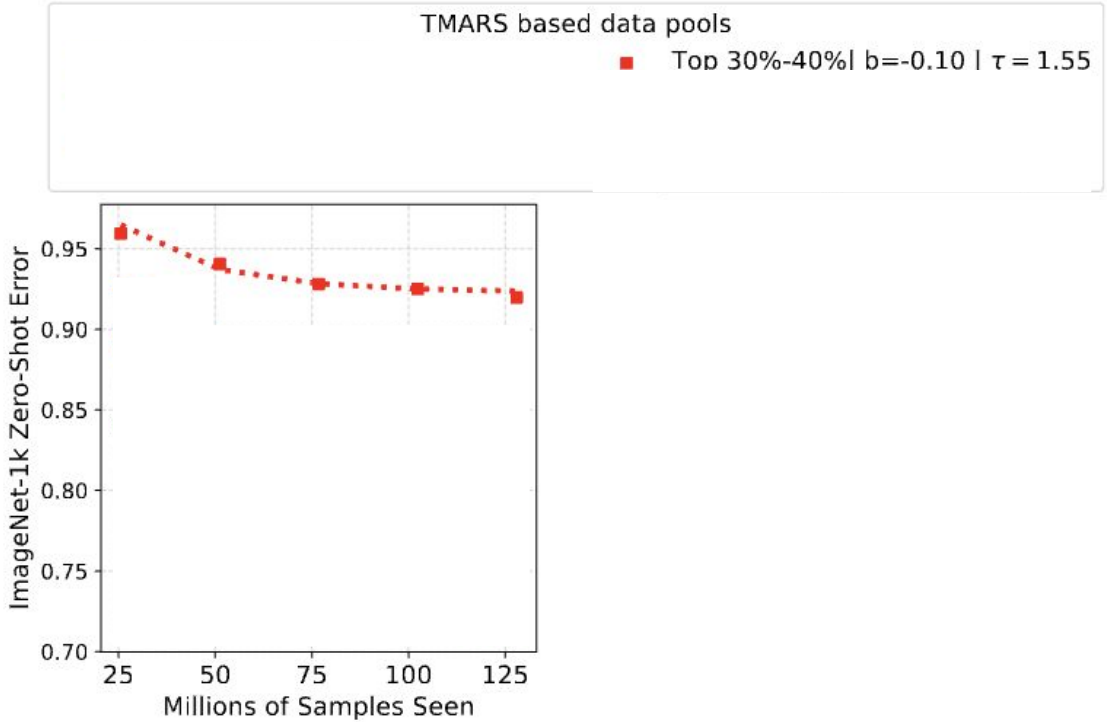
$$b_{\text{eff}}^{(k)} = \frac{1}{p} \sum_{i \in [p]} b_i \delta_i^{k-1} \qquad (p \text{ buckets, } k \text{ epochs})$$

Alternative formulations, not tested

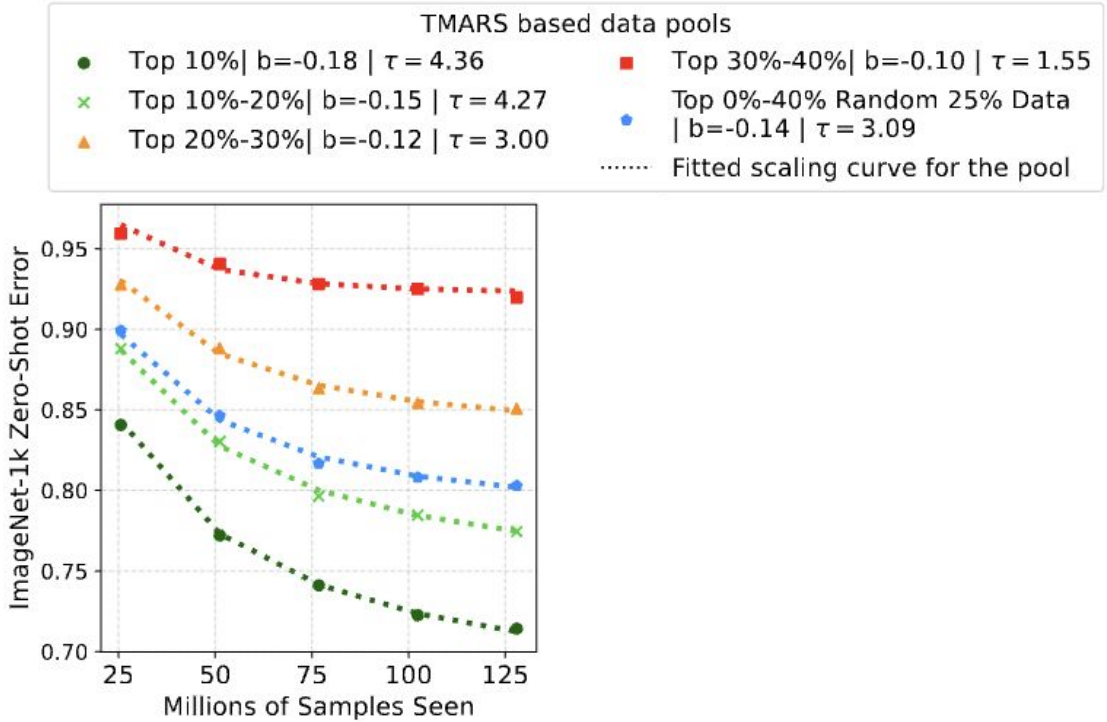
Utility does not have to decay exponentially

Instead of decreasing effective utility, you could decrease effective sample count (Muenninghoff et al, 2023)

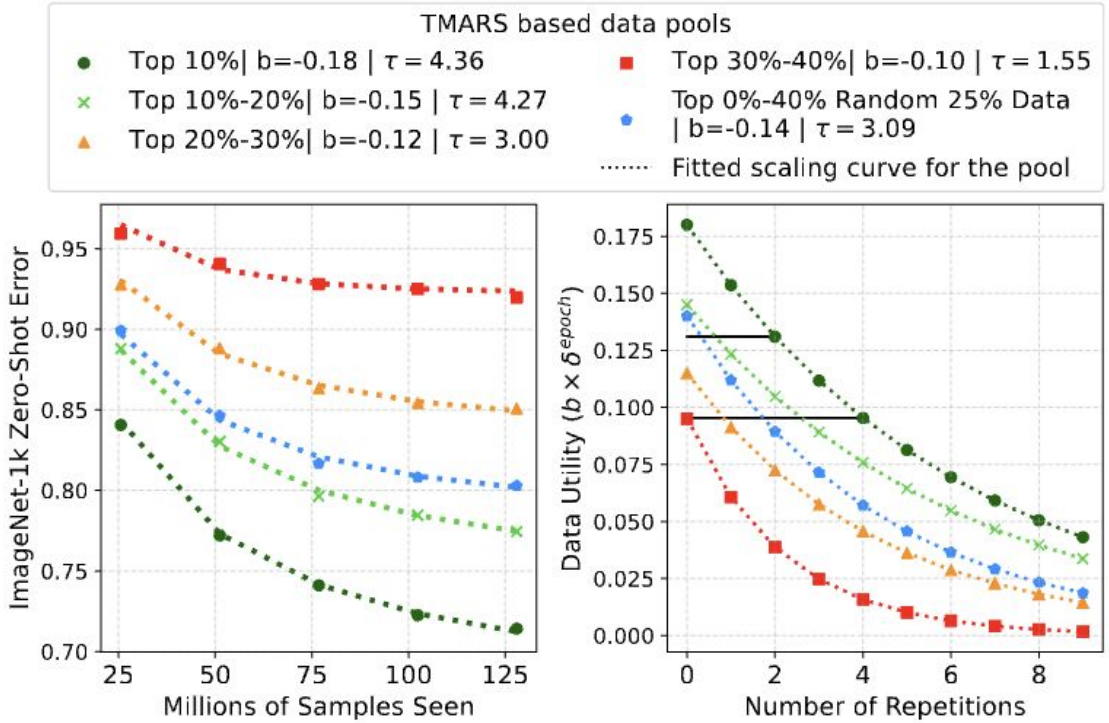
Fitting scaling laws to CLIP buckets



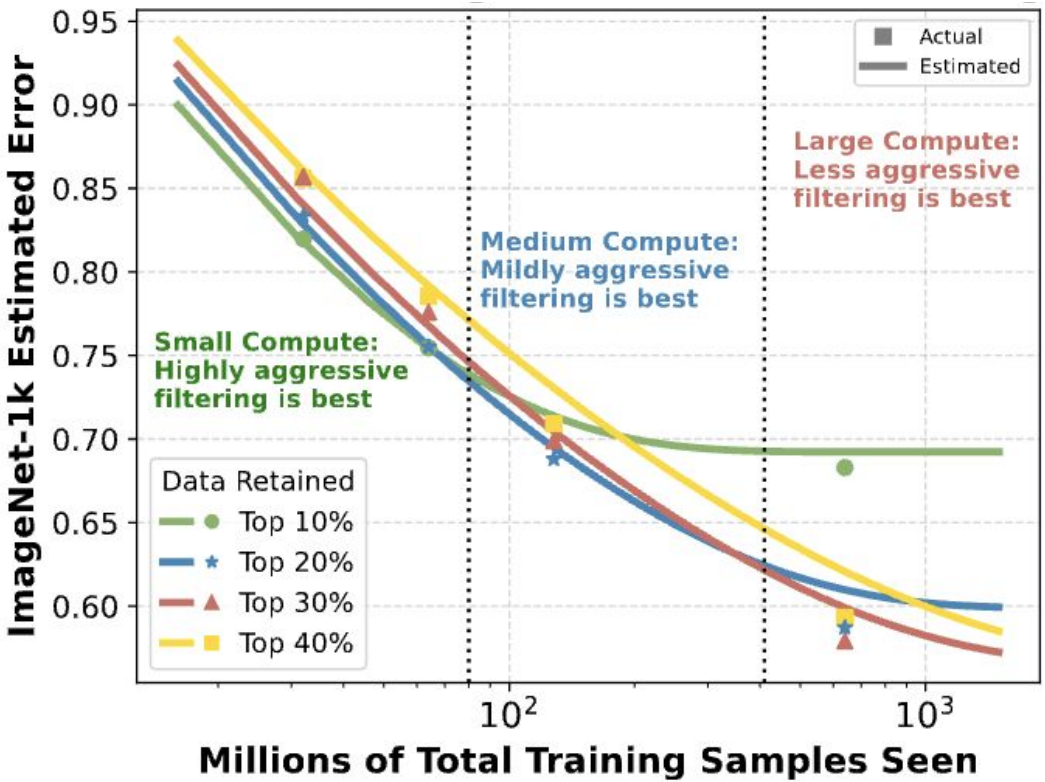
Fitting scaling laws to CLIP buckets



Fitting scaling laws for single bucket, multi epoch



Testing scaling laws for multiple buckets, multi epoch



Conclusion

In settings where repetition reduces the value of a data point, there is a tradeoff between same high quality data or more low quality data

The optimal decision depends on the amount of available compute

Scaling laws (may be able to) predict the optimal mixture

Open Problems

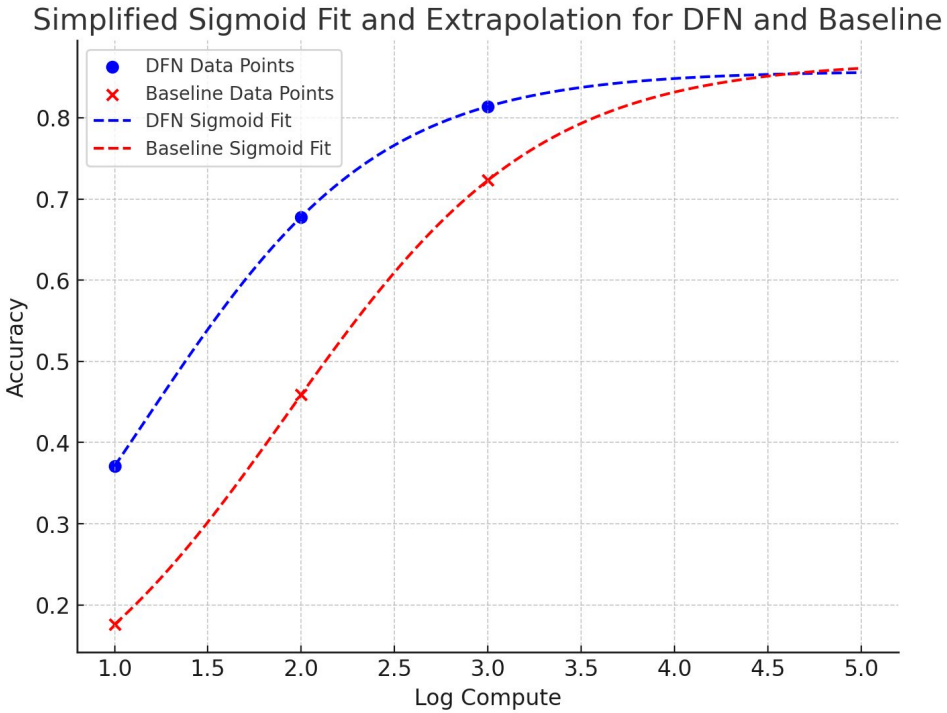
1. Importance of filtering in the limit

If compute increases and data stays fixed, data filtering will be less relevant. How much data filtering would one like to do in the infinite compute limit?

1. Importance of filtering in the limit

If compute increases and data stays fixed, data filtering will be less relevant. How much data filtering would one like to do in the infinite compute limit?

Open problem: build scaling laws for the effective benefit of data as compute increases



2. Distributionally aware filtering

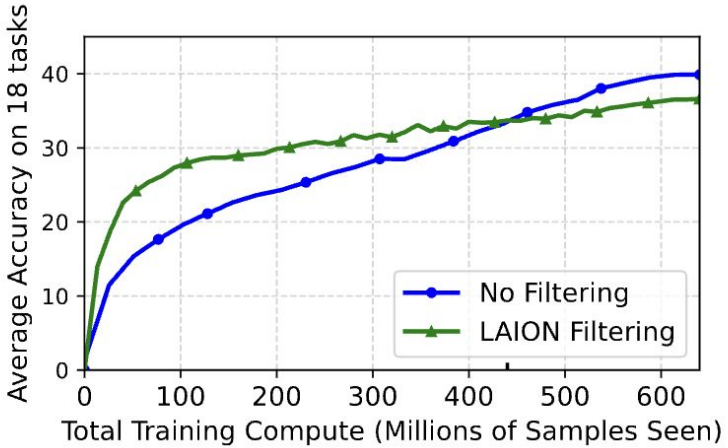
Current data filtering strategies consider each domain to have additive impact. However, the composition of your data matters: the benefit of code data may depend on the amount of math data. With unlimited data, today's strategies may just pick a single domain, which is known to be suboptimal

$$y_i = f(\langle \boldsymbol{\theta}^*, \mathbf{x}_i \rangle + \epsilon_i) \qquad y = a \cdot n^b + d$$

This is really important for synthetic data, since one possible future has infinite data from each domain with strong mode collapse concerns

3. Curriculum: does varying mixture over time help?

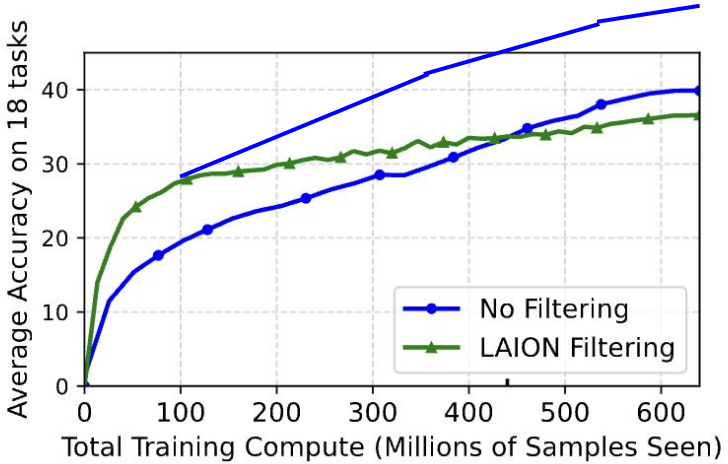
Given domains and a target validation set, we know that the optimal mixture can be better than the mixture of the validation set.



3. Curriculum: does varying mixture over time help?

Given domains and a target validation set, we know that the optimal mixture can be better than the mixture of the validation set.

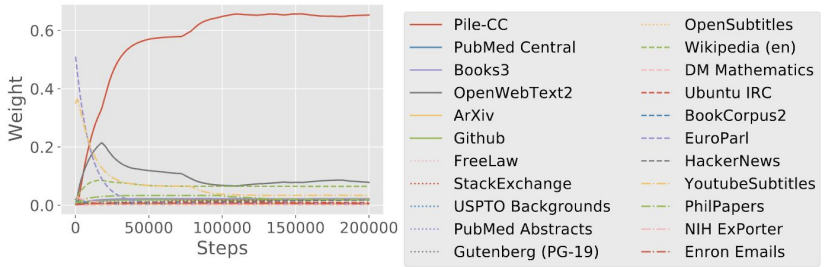
Can varying the mixture over time be much better than the optimal mixture?



3. Curriculum: does varying mixture over time help?

Given domains and a target validation set, we know that the optimal mixture can be better than the mixture of the validation set.

Can varying the mixture over time be much better than the optimal mixture?

One Evidence Against	One Evidence For
Doremi mixture stabilizes quickly 	Industry models mix high quality data at the end “We used curriculum learning for pretraining, changing the data mix during training in ways we found to substantially improve model quality.” - DBRX “We stage training to alter the mixture composition during training – increasing the weight of domain-relevant data towards the end of training” - Gemini