Appendix

A. Scaling curves for average performance across 18 tasks

In Figure 4 we showed the obtained scaling curves for each of the data pools, with ImageNet as the downstream performance metric. We share similar curves for each of the data pool, with average performance across 18 tasks as the downstream performance metric, in Figure 7.

Finally, Figure 8 shows the estimated scaling curves for different data pool mixtures, for average prformance across 18 tasks as the downstream performance metric.

B. Downstream Evaluation Datasets and Metrics

Following prior work [37, 45], we evaluate our models on a variety of image classification and retrieval datasets to assess their zero-shot capabilities. While the Datacomp [14] benchmark averages performance across 38 different datasets, we use a subset of 18 such datasets where medium-scale models give better than random performance in order to be able to develop reliable scaling laws. More specifically, we select the following datasets:

1. ImageNet: a 1000-class image classification challenge [40].
2. ImageNet-OOD: Six associated Imagenet distribution shifts—ImageNet-V2 [39], ImageNet-R [19], ImageNet-A [18], ImageNet-Sketch [44], ImageNet-O [18], and ObjectNet [4].
3. VTAB: 6 out of 12 datasets from the Visual Task Adaptation Benchmark [49], including Caltech-101 [13], CIFAR10 [28], CIFAR100 [28], Oxford Flowers-102 [34], Oxford-IIIT Pets [35], and RESISC45 [8].
4. Additional classification datasets: Food-101 Bossard et al. [5], Pascal VOC 2007 [10], and Stanford Cars [27].
5. Retrieval: 2 retrieval tasks of MSCOCO [7] and Flickr [47].

Most of the evaluation datasets constitute image-classification tasks. We use the 'Accuracy metric' to evaluate the zero-shot performance of the model on these datasets. The only exceptions include:

1. VTAB: We report 'Mean per Class Recall' for Caltech-101 [13], Oxford Flowers-102 [34], Oxford-IIIT Pets [35] datasets. This follows the standard evaluation protocol in past benchmarks [14] and is done because of the large number of classes in these datasets.
2. Retrieval: For all the retrieval datasets we report the 'Mean Recall @ 1' which tells how probable is it for the top-recall entry to be relevant.

C. Scaling curves with CLIP Score as the Data Quality Metric

We share the scaling curves for data pools based on the CLIP score in Figure 9. Similar to the trends observed in § 5, we note that the magnitude of data quality parameter (b) decreases as we go towards worse pools.

D. Pareto data filtering for T-MARS

Figure 10 shows the variation in optimal data filtering strategy for T-MARS as the compute is scaled from 32M to 640M.

E. Finding the Optimal Scaling Parameters

Optimizing scaling parameters is a critical step in the validation of any scaling laws, particularly when these models are subjected to scaling laws that govern their behavior across different magnitudes of computation. This process, however, is fraught with challenges, primarily due to the sensitive nature of the optimization landscape, which is characterized by numerous local minima and a complex objective function.

E.1. Challenges in Parameter Optimization

In our initial approach to optimizing for the scaling parameters like a, b, c, τ we employed optimization libraries such as SciPy to find the best fit on the training points. Despite their widespread use, these methods often result in instability and inefficiency in the fitting process, especially being sensitive to initialization. The primary reason for this instability is the high sensitivity of the optimization problem, where slight variations in parameters can lead to significantly different outcomes, and at the same time, the existence of many solutions that fit the curve very well. As a remedy to this, we developed optimization methods ranging from gradient-based optimization to custom-built optimizers. In all of these approaches, we found that we could attain solutions that (a) had very high loss on the data points being optimized over; and (b) conversely, had a low loss on the points it was trying to minimize, but high loss on the points being extrapolated (mixture of data buckets). This was a rather unsatisfactory, because we should find scaling parameters *without* peeking at the points to extrapolate to.

E.2. Grid Search as a Stable Solution

Given the limitations of conventional optimization methods, our work advocates for the use of grid search as a more stable and reliable method for determining optimal scaling parameters. Grid search, unlike heuristic or gradient-based optimization methods, systematically explores a specified parameter space, evaluating the performance of each parameter combination to identify the one that yields the best

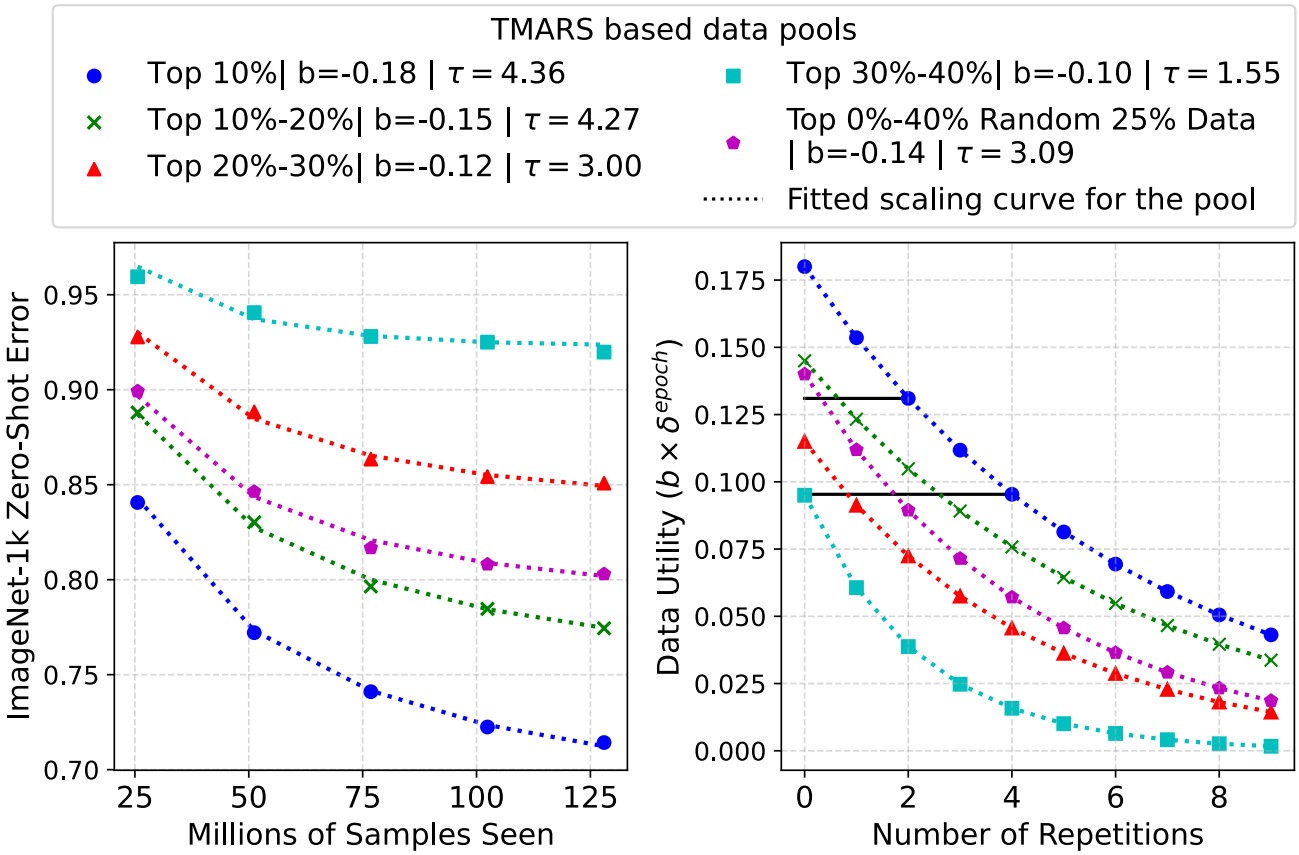

Figure 7. Scaling curves with repeated data for visual-language models: We share the scaling curves for T-MARS based data pools, with average zeroshot performance across 18 tasks as the downstream performance metric.

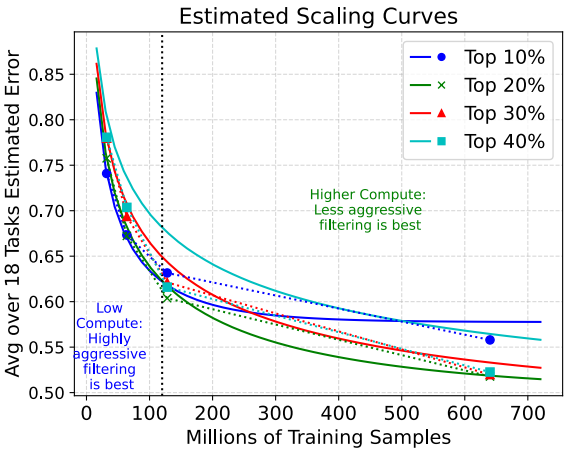

Figure 8. Estimated Scaling curves for different data pool mixtures (data pools based on T-MARS scores). Again, we observe that highly aggressive filtering performs the best at low compute, while moderate filtering is better at higher computes.

results. This exhaustive search process is especially advantageous in the context of scaling parameter optimization, where the landscape is sensitive to initialization. This approach mitigates the risk of overlooking optimal parameters due to the presence of local minima or the complexities of the objective function. Furthermore, grid search facilitates a reproducible parameter fitting process, enabling a clearer understanding of how different parameters influence the model's performance.

E.3. A view of the loss grid

We perform a gridsearch for the normalizer $a \in [0.001, 1]$, data utility parameter $b \in [-0.005, -0.5]$, repetition parameter $\tau \in [1, 50]$ and $d \in [0.01, 0.02, 0.05, 0.10, 0.2]$. These grids were converged upon by continuously expanding the grid limits till the estimated optimal parameters for all the data pools lied in between the grid.

E.4. Finding normalizing constant a

Recall from § 4.2 that a refers to the normalizer term in the scaling equation which is kept constant across the pools.

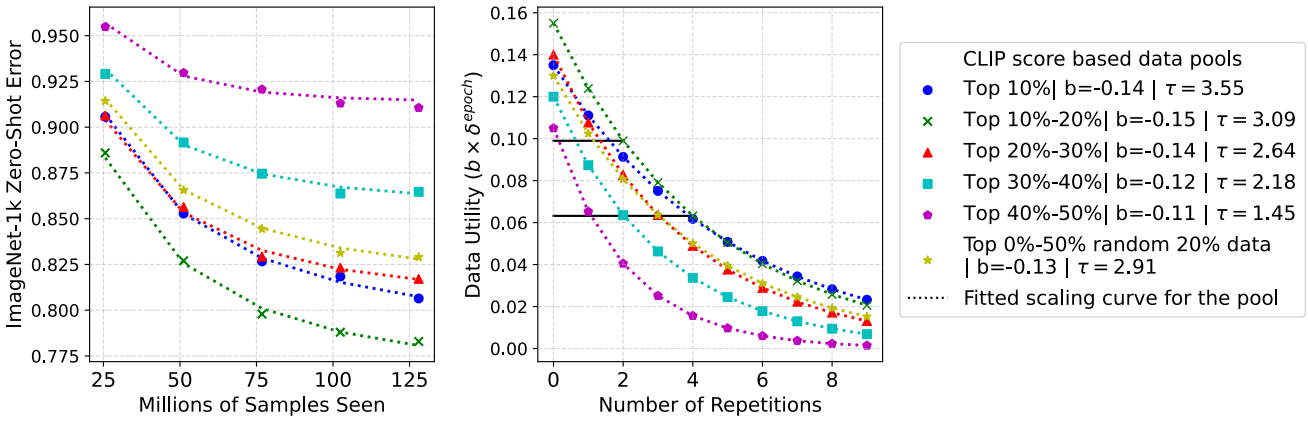

Figure 9. **Scaling curves for data pools based on CLIP score**:

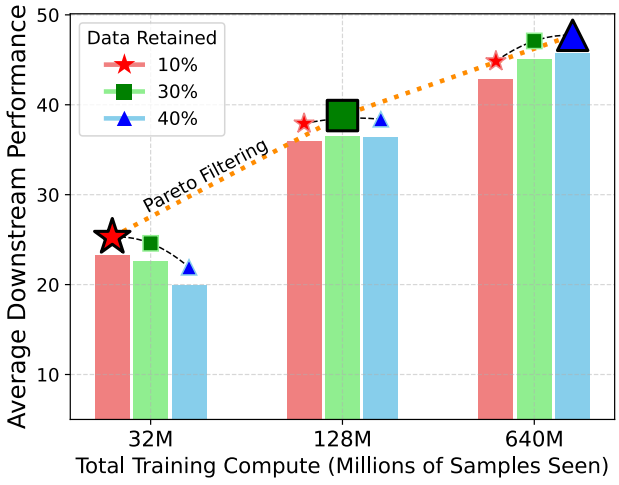

Figure 10. We modify the state of the art data curation approach T-MARS by changing the filtering threshold after ranking the data by their metric. While the original paper proposed retaining 30% of the data, our results show that depending on the ratio of compute to data pool size, we must adaptively 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.

We learn a by performing a gridsearch over 100 possible values of $a \in [0, 1]$, and choosing the one which minimizes the combined loss across all the data quality pools. While we fix a to be same across the pools, we let b and τ to vary. Figure 11 shows the loss surface as we jointly optimize for a over data pools. We observe a well behaved loss surface over the gridsearch range, giving an optimal a value.

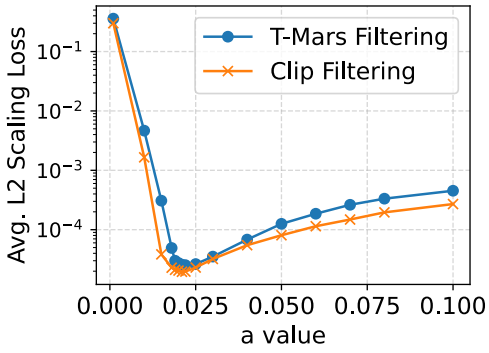

Figure 11. Loss surface as we jointly optimize for a over the data pools. There exists a well-defined minima over our chosen grid-search range for a.

E.5. Learning the data repetition parameter τ

Recall from § 5 that we learnt different data repetition parameter (which denotes data diversity) τ for each of the data pools. One could alternatively chose to have the same diversity parameter across the pools, assuming that diversity varies uniformly in the web data. However, as shown in Figure 12, learning a same τ value for all the buckets gives a much worse (high) L2 fitting error compared to learning different values for each of the pool (see the horizontal dotted curve).

F. What is the Axiom of Scaling?

Scaling laws have been studied for a while, especially when considering the scaling of large language models with increasing amounts of data [21, 26]. Since these formulations have abstracted away the 'per-sample' nature of scaling to the general formulation of the form $y = n^b$, it remains an open question to determine what is the actual 'axiom' of scaling. The answer to this question becomes extremely im-

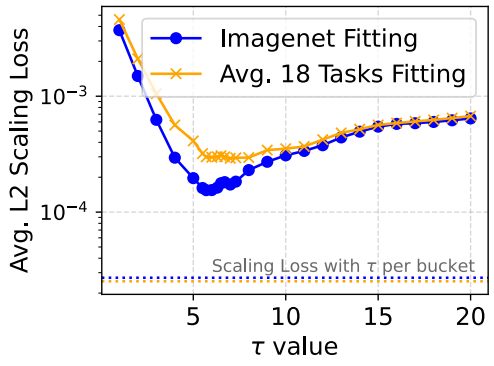

Figure 12. Keeping the data diversity parameter (τ) different across the pools is necessary for good scaling curve fitting. Keeping the τ value same across the pools give a much worse (higher) L2 fitting loss compared to using different for each pool (see the horizontal dotted line).

portant in enabling us to study the dynamics of a mixture of data buckets of differing utilities, and that too, in an environment where the utility of data is constantly diminishing. In this section, we sequentially consider three different 'axioms' of scaling. We consider three different formulations that can potentially explain the sample level dynamics of scaling laws—(i) first, we assume that the derivative of the original scaling laws hold universally true, in our utility based formulation; (ii) second we consider the formulation that considers the effective data in the system at any point but considers the utility of data to be constant, as in Muennighoff et al. [32]; (iii) and finally a formulation that models the decay of data and utility simultaneously. In each of the discussions, we begin with certain assumptions, and examine if those assumptions hold at the event of a mixture of data buckets, that is, starting from an axiom, can we get back to that to predict the system loss in a mixture of data buckets.

F.1. Utility Based Decay

We begin with our first scaling law formulation that models only the instantaneous system state and not the global state. The intuition behind such a setup is that it allows the mixing of different buckets naturally by calculating the instantaneous change in the loss by training on any sample from any data bucket. Assume that 5 represents the axiom of scaling of loss with data seen.

$$\frac{dy}{dn} = \frac{y(n)}{n} b \tag{5}$$

The above can be thought of as the derivative of the more generally recognized scaling law $y = n^b$. Then, rearranging we have that,

$$\frac{dy}{y} = \frac{dn}{n} b \tag{6}$$

$\Rightarrow$ Assume that 'b' utility term is a constant for a given region of the training curve. Then, between two such points in the state where we have seen n_1, n_2 samples respectively,

$$\int_{y_1}^{y_2} \frac{dy}{y} = \int_{n_1}^{n_2} \frac{dn}{n} b$$
$$\log \frac{y_2}{y_1} = b \log n_2 - b \log n_1$$

$$\log \frac{y_2}{y_1} = b \log \frac{n_2}{n_1} \tag{7}$$

Given a single data bucket, the assumption that 'b' is constant, should for instance, hold true during a given epoch. This gives us the following closed form relationship between loss values after seeing n_1, n_2 samples respectively.

$$y_2 = y_1 \left(\frac{n_2}{n_1}\right)^b \tag{8}$$

The closed-form solution for the rate of decay or loss, y_k, between training epochs is given by:

$$y_k = y_1 \left(\frac{n_2}{n_1}\right)^{b_2} \left(\frac{n_3}{n_2}\right)^{b_3} \cdots \left(\frac{n_k}{n_{k-1}}\right)^{b_k} \tag{9}$$

Where y_1 is the loss at the end of the first epoch, and

$$y_k = n_1^{b_1} \prod_{j=2}^{k} \left(\frac{n_j}{n_{j-1}}\right)^{b_j}.$$

The scaling term b is assumed to be a constant for a given region of the training curve. For example, during a given epoch, the relationship between y_2 and y_1 is given by $y_2 = y_1 \left(\frac{n_2}{n_1}\right)^b$.

F.1.1 Decay of the utility parameter

Now, let us move into the paradigm of repeated epochs. We ask the question, how do we model the change in utility of a bucket as we see the dame data point multiple times. We assume that the utility parameter b decays exponentially at every epoch at a constant factor δ. This means that,

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

F.1.2 Estimating Data Mixtures under Utility-based decay

Now that we have established the basic laws for the system state under the scaling laws modeling utility-based decay, let us examine how these assumptions play together in a setup of a mixture of buckets. For simplicity, whenever we consider a system with multiple data buckets, we will assume that the data points from these buckets are cycled alternately. Let us assume that we are at a given system state where the model loss and samples seen are determined by (y_0, n_0) respectively. Assume two data buckets parametrized by (b_1, δ_1) and (b_2, δ_2) as their respective initial utility, and utility decay rates. Let us first sample a data point from bucket 1, and then from bucket 2. Assume that the loss after seeing a single sample from the two buckets is y_1, y_2 respectively. Using the relationship in Equation 8:

$$\frac{y_1}{y_0} = \left(\frac{n_0+1}{n_0}\right)^{b_1} = \left(1+\frac{1}{n_0}\right)^{b_1}$$
$$\frac{y_2}{y_1} = \left(\frac{n_0+2}{n_0+1}\right)^{b_2} = \left(1+\frac{1}{n_0+1}\right)^{b_2}$$

Using Taylor expansion and ignoring higher order term assuming that $n_0 \gg 1$, we can write the above as:

$$\begin{aligned}\frac{y_2}{y_0} &= \left(1+\frac{1}{n_0}\right)^{b_1}\left(1+\frac{1}{n_0+1}\right)^{b_2} \\ &\approx \left(1+\frac{1}{n_0}\right)^{b_1}\left(1+\frac{1}{n_0}\right)^{b_2} \\ &= \left(1+\frac{1}{n_0}\right)^{b_1+b_2}\end{aligned}$$

Let us assume that the effective utility of the mixture of data buckets is b_{eff}. Then, in the same steps between y_0 and y_2, we can write:

$$\frac{y_2}{y_0} = \left(1+\frac{2}{n_0}\right)^{b_{\text{eff}}}$$

Since the two formulations should be equivalent, we have that

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

This gives us a way to estimate the effective utility of the mixture of data buckets at any given epoch k of training as,

$$b_{\text{eff}}^{(k)} = \frac{b_1\delta_1^{k-1} + b_2\delta_2^{k-1}}{2}$$

As we move into the next formulation, let us highlight that the key challenge behind the validity of this formulation is the initial axiom we assumed. We assume that the system is determined by instantaneous interactions where the effective number of samples seen is constant, independent of the number of times it has been seen before, and only its utility decreases every time.

F.2. Effective Data Based Formulation

Let us now consider a formulation that only accounts for the 'effective data' in the system to predict the loss of the model on seeing 'n' samples. The central assumption behind this formulation is that the 'utility' of data itself does not decay, and stays constant throughout the training process, rather only the effective data in the system decays. This is the formulation that has been considered in the past by Muennighoff et al. [32]. Let us consider that the 'axiom' of scaling law is now represented by the following equation:

$$\begin{aligned} y &= n_{\text{eff}}^{b}, \\ n_{\text{eff}} &= \eta(1+\delta+\delta^2+\cdots+\delta^{k-2}) + \gamma\delta^{k-1} \end{aligned} \tag{10}$$

where η represents the number of unique samples in the training data, δ represents the rate of decay of effective data when seeing a repeated epoch of the data, and $\gamma < \eta$ represents the number of examples seen in the current epoch of training. This suggests that at a given system state, $dn_{\text{eff}} = \delta^{k-1}dn$

$$\frac{dy}{dn_{\text{eff}}} = b\frac{y(n_{\text{eff}})}{n_{\text{eff}}}$$
$$\frac{dn_{\text{eff}}}{dn} = \delta^{k-1}$$

Note that since the system state has a fixed utility value throughout training, we can use the formulation from Equation 8 in terms of n_{eff} as follows:

$$y_2 = y_1\left(\frac{n_{\text{eff2}}}{n_{\text{eff1}}}\right)^{b}$$

F.2.1 Estimating Data Mixtures under Effective Data-based decay

Let us now consider the case where we have a mixture of data buckets, and we are trying to estimate loss of the system characterized by the mixture of data buckets. As before, let us assume that we are at a given system state where the model loss and samples seen are determined by (y_0, n_0) respectively. Assume two data buckets parametrized by (η_1, δ_1) and (η_2, δ_2) as their respective initial effective data, and effective data decay rates. Let us first sample a data point from bucket 1, and then from bucket 2. For simplicity, we will consider the case of training on the second epoch of

data:

$$\frac{y_1}{y_0} = \left(\frac{n_0+\delta_1}{n_0}\right)^{b_1} = \left(1+\frac{\delta_1}{n_0}\right)^{b_1}$$
$$\frac{y_2}{y_1} = \left(\frac{n_0+\delta_1+\delta_2}{n_0+\delta_1}\right)^{b_2} = \left(1+\frac{\delta_2}{n_0+\delta_1}\right)^{b_2}$$
$$\frac{y_2}{y_0} = \left(1+\frac{\delta_1}{n_0}\right)^{b_1}\left(1+\frac{\delta_2}{n_0+\delta_1}\right)^{b_2}$$
$$\approx \left(1+\frac{\delta_1}{n_0}\right)^{b_1}\left(1+\frac{\delta_2}{n_0}\right)^{b_2}$$
$$= \left(1+\frac{\delta_1}{n_0}\right)^{b_1}\left(1+\frac{\delta_2}{n_0}\right)^{b_2}$$

By Taylor expansion, and keeping only the first order terms, in the expansion of $(1+x)^a$ for very small x, we have that $(1+x)^a \approx 1+ax$. This gives us that,

$$\frac{y_2}{y_0} \approx \left(1+\frac{\delta_1}{n_0}\right)^{b_1}\left(1+\frac{\delta_2}{n_0}\right)^{b_2}$$
$$\approx \left(1+\frac{\delta_1}{n_0}b_1\right)\left(1+\frac{\delta_2}{n_0}b_2\right)$$
$$\approx 1+\frac{\delta_1 b_1+\delta_2 b_2}{n_0}$$

But, we know that this should be equivalent to the formulation where we consider a fixed utility of data throughout training, given by b_{eff}.

$$\frac{y_2}{y_0} = \left(1+\frac{\delta_1+\delta_2}{n_0}\right)^{b_{\text{eff}}}$$

Since the two formulations should be equivalent, we have that

$$b_{\text{eff}} = \frac{\delta_1 b_1+\delta_2 b_2}{\delta_1+\delta_2}$$

Notice that in the final formulation, b_{eff} is a weighted average of the utility of the data in the two buckets, with the weights being the effective data in the system at the time of training. However, this contradicts our initial assumption that it is only the data that decays in the system, and the utility stays constant. While the utility does not 'decay', it does get re-weighted as the effective data in the system changes. This is a key insight that we gain from this analysis, which requires us to consider the change in both the effective data and the utility of data in the system. Note that these results still do not violate the formulation proposed in the work of Muennighoff et al. [32], because they consider data to come from a single bucket. This means that $b_{\text{eff}} = {}^{\delta_1 b_1}/_{\delta_1} = b_1$. However, in the case of a mixture of data buckets, the utility of data in the system is a weighted average of the utility of data in the individual buckets.

An alternate paradigm in which such an equation may work is when the rate of decay (δ) associated with all buckets is the same. However, we find that adding the constraint of fixed decay factor between all buckets leads to significantly higher loss even on the points to be fit for scaling law, let alone the extrapolation points.

F.3. Decaying Effective Data with changing Utility

Based on the insights from the previous two formulations, we now consider a formulation that models the decay of both the utility of data and the effective data in the system. It is clear from the previous part that we need a formulation where we can model the instantaneous utility of data in order to be able to mix different data buckets. The key difference from the formulation based on only utility decay is that we consider that there is a decay factor associated with the effective number of samples in the system. Let us first rewrite the instantaneous system state in terms of the effective data and utility of data as follows:

$$\frac{dy}{dn_{\text{eff}}} = \frac{y(n_{\text{eff}})}{n_{\text{eff}}}b, \qquad \frac{dn_{\text{eff}}}{dn} = \delta^{k-1} \tag{11}$$

From the formulation in the previous part in Section F.2.1, it then follows that,

$$b_{\text{eff}}^{(k)} = \frac{b_1\delta_1^{k-1}+b_2\delta_2^{k-1}}{\delta_1^{k-1}+\delta_2^{k-1}},$$

where $b_{\text{eff}}^{(k)}$ is the effective utility of the mixture of data buckets at any given epoch k of training.

Using the above, we can consider the closed form solution for the loss of the system at any given epoch k of training as a product of the loss at the end of the first epoch, and the ratio of the effective data seen at the end of the kth epoch to the power effective utility.

$$y_k = y_1\left(\frac{n_{\text{eff2}}}{n_{\text{eff1}}}\right)^{b_{\text{eff}}^{(2)}}\left(\frac{n_{\text{eff3}}}{n_{\text{eff2}}}\right)^{b_{\text{eff}}^{(3)}}\cdots\left(\frac{n_{\text{effk}}}{n_{\text{eff(k-1)}}}\right)^{b_{\text{eff}}^{(k)}}$$
$$y_k = y_1\prod_{j=2}^{k}\left(\frac{n_{\text{effj}}}{n_{\text{eff(j-1)}}}\right)^{b_{\text{eff}}^{(j)}}$$

F.3.1 Estimating Data Mixtures under Decaying Effective Data with changing Utility

We will directly use the data mixing results from F.2.1 to estimate the loss of the system at any given epoch k of training. Note that this is governed under the assumptions of exponential decay of data with repetitions.

F.4. Equivalence of Formulations A and C

We will now show that under certain conditions, the decay of data and utility, and the decay of just utility reduce to the same formulation. Let us consider the case where the decay factor of the effective data is the same as the decay factor of the utility of data. Consider two buckets parametrized by (b_1, δ) and (b_2, δ) as their respective utility and decay factor. We will now study the ratio of the two losses after seeing a single sample from the two buckets. Once again assume that the initial system state is characterized by (y_0, n_0). For simplicity, we will consider the case of training on the second epoch of data:

Effecttive Utility Formulation Recall that

$$b_{\text{eff}} = \frac{b_1\delta^{k-1} + b_2\delta^{k-1}}{2}.$$

Then, we can write the loss after seeing a single sample as:

$$\begin{aligned}
\frac{y_1}{y_0} &= \left(1 + \frac{1}{n_0}\right)^{b_{\text{eff}}} \\
&= \left(1 + \frac{1}{n_0}\right)^{\frac{b_1\delta_1 + b_2\delta_2}{2}} \\
&\approx \left(1 + \frac{1}{n_0}\right)^{\frac{b_1\delta_1 + b_2\delta_2}{2}} \\
&\approx \left(1 + \frac{b_1\delta_1 + b_2\delta_2}{2n_0}\right)
\end{aligned}$$

Effective Data, Dynamic Utility Based Formulation In this case, the

$$\begin{aligned}
b_{\text{eff}} &= \frac{\delta_1 b_1 + \delta_2 b_2}{\delta_1 + \delta_2} \\
\delta_{\text{eff}} &= \frac{\delta_1 + \delta_2}{2}
\end{aligned}$$

Then, we can write the loss after seeing a single sample from the two buckets as:

$$\begin{aligned}
\frac{y_1}{y_0} &= \left(1 + \frac{\delta_{\text{eff}}}{n_0}\right)^{b_{\text{eff}}} \\
&= \left(1 + \frac{\delta_1 + \delta_2}{2n_0}\right)^{\frac{\delta_1 b_1 + \delta_2 b_2}{\delta_1 + \delta_2}} \\
&\approx \left(1 + \frac{\delta_1 b_1 + \delta_2 b_2}{2n_0}\right)
\end{aligned}$$

Notice that, under the Taylor approximation, and assumption that $n_0 \gg 1$, the two formulations are equivalent. This suggests that under the assumption that the decay factor of the effective data is the same as the decay factor of the utility of data, the two formulations reduce to the same formulation. This is a key insight that we gain from this analysis, which requires us to consider the change in both the effective data and the utility of data in the system. Note that these results still do not violate the formulation proposed in the work of Muennighoff et al. [32], because they consider data to come from a single bucket. This means that $b_{\text{eff}} = {}^{\delta_1 b_1}/_{\delta_1} = b_1$. However, in the case of a mixture of data buckets, the utility of data in the system is a weighted average of the utility of data in the individual buckets.

G. Rate of Repetition Decay

Conventionally scaling laws have been studied in the context of language models [21, 26]. To the best of our knowledge, the only large scale attempt at predicting scaling of vision language models was done in the work of Cherti et al. [9]. However, the extrapolated graphs in their work showed large errors as compared to the actual performance of the models. Hence, the scaling of vision language models is still an open question, especially when trained under contrastive loss.

Challenge: Squared Contrastive Pairs This is a fundamentally new challenge because in a data pool of N samples, there exist N^2 pairs of comparisons that contribute to the loss. In such a setting, *should the utility of a data pool still diminish exponentially after seeing N samples, or should it diminish after seeing N^2 pairs?* This is a fundamental question that we address in this section.

The contrastive loss for any given batch of data can be given by the following equation:

$$\mathcal{L} = -\log \frac{\ell((x_i, t_i))}{\sum_{k=1}^{B-1} \ell(x_i, t_k)}, \tag{12}$$

where B is the batch size, and ℓ is some loss function based on the cosine similarity between the embeddings of the image and the text. x_i and t_i are the image and text embeddings of the ith sample in the batch. The goal is to maximize the similarity between the image and text embeddings of the same sample, and minimize the similarity between the embeddings of different samples. We can decompose the loss into the numerator and the denominator. In a given batch, there are a total of B samples seen in the numerator, and $B \times (B-1)$ samples seen in the denominator. We will call this $B \times B$ for simplicity.

Let us assume that the training set up uses a batch size of B for contrastive learning. Given a dataset of N samples, the number of batches seen in a single epoch is given by $\frac{N}{B}$. Therefore, the total number of comparisions seen in the denominator in a single epoch is given by $N \times B$.

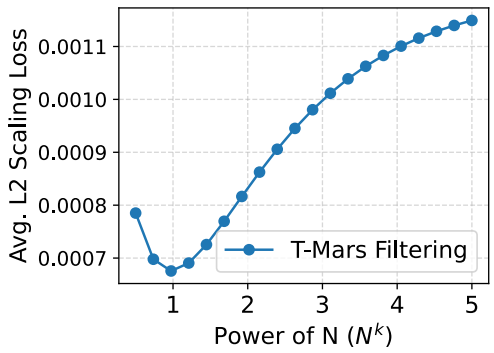

Figure 13. Loss surface as we find the best rate of decay of utility as a function of N^k.

Utility Decay Assumption We assume that the utility decays exponentially, however, only after seeing each unique comparison in the dataset. For a data pool of N samples, while conventionally such a decay event would happen every epoch after seeing N samples, in the case of contrastive learning, this decay event should happen after seeing N^2 pairs of samples. This means that the number of epochs after which the utility of the data pool decays should be given by $\frac{N^2}{N \times B} = \frac{N}{B}$.

Decay formulation Let us assume that δ_{g} is the gold rate of decay after seeing N^2 samples. Since the modeling till now was done based on utility decay per epoch, we can write a relationship between the gold rate of decay and the rate of decay after seeing N samples as follows:

$$\delta_g = \delta^{N/B} = \frac{1}{2}^{N/B\tau}$$

Finding decay value on merging data buckets When we merge data buckets, a unique phenomenon happens. The overall pool size increases. While in the conventional language modeling paradigm this may not have a significant impact, in the case of contrastive learning, this changes decay rate because the number of comparisons in the denominator increase at the squared rate of total samples. Using δ_g we can find the new value of $\hat{\tau}$ as follows:

$$\delta_g = \frac{1}{2}^{N/B\tau} = \frac{1}{2}^{\hat{N}/B\hat{\tau}}$$
$$\hat{\tau} = \frac{\hat{N}}{N}\tau$$

Hence, $\tau_p = p\tau$ when merging p different buckets of a given size.

Finding the best decay rate In the above discussion, we assumed that the rate of decay should happen after seeing N^2 pairs of samples. However, this is an assumption that we made. To test how this empirically holds with respect to different rates of change in N we allow for the flexibility of $\tau_p = p^k\tau$. Then we run an estimation analysis, at various values of k as in Figure 13 to find the best value of k that minimizes the loss on the data points to be fit. We notice that the results of the theoretically derived relationship exactly match the empirically determined value of minimum loss at $k = 1$. Hence, for the purposes of all the merging experiments, we use $\tau_p = p\tau$ as the rate of decay of utility of data in the system when merging p different data buckets.