Training dataset size prediction when adopting image counts individually per class.

Thomas Mühlenstädt, Jelena Frtunikj

August 8, 2023

Abstract

To be added at the end.

1 Introduction

To be added at the end.

2 relevant literature

The authors of [**?**] go beyond estimation of data set requirements from power law function and investigate three alternative regression functions i.e. Arctan, Logarithmic, Algebraic Root. They show that all of the functions are well-suited towards estimating model performance however each function is almost always either overly optimistic i.e. under-estimating the data requirement or pessimistic i.e. over-estimating. This means that there is no one best regression function for all situations. Through simulation of a data collection workflow the paper shows that incrementally collecting data over multiple rounds and combining those with techniques that under-estimate leads to collecting up to 90% of the true amount of data needed. In addition the authors introduce a correction factor that can be learned by simulating on prior tasks and which helps to better fitting functions. The fitting of the three regression functions and the benefits of the usage of the correction factor have been applied on classification, detection, and segmentation tasks with different data sets incl. CIFAR10.

The paper [**?**] focuses on predicting the needed training dataset size for achieving necessary accuracy for a classification task in the medical domain. More precisely the use case is classifying axial Computed Tomography (CT) images into six anatomical classes. The authors apply the already introduced ([**?**]) power law fitting to the mentioned use case. This paper focuses only on the medical domain and provides among the first empirical results that the power law fitting for determining the training data set size also applies for the medical domain.

Inspired by the fact that power law scaling of the error w.r.t. data suggests that many training examples are redundant, the authors of [**?**] investigate the hypothesis of pruning training datasets to much smaller sizes and training on the smaller pruned datasets without sacrificing performance. The authors show in theory and in practice that one can break beyond power law scaling and even

reduce it to exponential scaling if one uses an efficient data pruning metric that ranks the order in which training examples are being discarded. The theory proof reveals two very interesting points: 1) the optimal pruning strategy changes depending on the amount of initial data i.e. with abundant (scarce) initial data, one should retain only hard (easy) examples and 2) exponential scaling is possible w.r.t. pruned dataset size only if one chooses an increasing Pareto optimal pruning fraction as a function of initial dataset size. The authors demonstrate empirically the exponential scaling of the error w.r.t. pruned dataset size for a ResNets trained from scratch on SVHN, CIFAR-10 and ImageNet, and Vision Transformers fine-tuned on CIFAR-10. In addition, the paper presents a new and comparably good unsupervised data pruning metric that does not require labels, unlike other prior unsupervised pruning metrics that require labels and much more compute. The k-means based pruning metric (clustering in the embedding space of an ImageNet pre-trained model) enables discarding of 20% of ImageNet data without sacrificing performance, on par with the best and most compute intensive supervised metric.

[**?**] [**?**] [**?**] [**?**] [**?**] [**?**] [**?**] [**?**]

- https://arxiv.org/pdf/2203.15556.pdf
- https://arxiv.org/pdf/2207.01725.pdf
- https://arxiv.org/pdf/2206.14486.pdf
- https://arxiv.org/pdf/2209.06640.pdf
- https://arxiv.org/pdf/2102.04074.pdf
- https://arxiv.org/pdf/1909.12673.pdf
- https://arxiv.org/pdf/1707.02968.pdf
- https://arxiv.org/abs/1511.06348
- Baidu paper?

3 Overview suggested method

The task considered here is image classification, i.e. we aim to find a model $y = f_\theta(x)$, where x is an image tensor and y being a one hot encoding for the classes contained in the dataset. The training data consists of k classes, which each class j having n_j^{max} labeled images in the training dataset with $n^{max} = \sum_{j=1}^{k} n_j^{max}$ being the overall number of training images. Also, for completeness, we assume there is a labelled test and/or validation data set used to calculate the performance of a trained model $f_{\hat{\theta}}(x)$. The performance metric we are using here is accurcy, denoted as $a(f_{\hat{\theta}}(.))$. But in general, the methods described here would also for other metrics like e.g. f_1 scores, precision or recall. We refer to standard literature like e.g. [**?**] and [**?**] for these general details. Most of the literatur for neural scaling laws focusses on assuming different training dataset sizes n^{train} for a number of different training jobs, i.e. to find a function $g_\omega(n^{train})$ which can predict the chosen performance metric for different training dataset sizes. This means implicitely that each class has the same importance for the performance. But this is not necessarily true, some classes might be more difficult to train on than others. Hence here an approach is taken, which allows

to estimate the importance of training images from each class individually. Also, the compute budget is an important aspect of performance, hence we also aim at considering how different number of training epochs affects performance. Hence, we are aiming at expanding the function $g_\omega(n^{train})$ to be more general: $g_\omega(n_1^{train}, \ldots, n_k^{train})$ shall be a function predicting the performance of our model depending on the individual class training image counts. Even more general, we can also incorporate the number of training epochs n^{epoch} used for estimating the parameters θ into the function, now becoming $g_\omega(n_1^{train}, \ldots, n_k^{train}, n^{epoch})$. In order to fit functions which allow for estimation of more differentiating effects between classes and epochs, more divers training datasets need to be generated. Hence one focus of this paper is to suggest an algorithm which allows for generation of divers training data sets. For modeling these data and making predictions and importance statements, non-linear models like powerlaw curves are fitted in a way that the input is a linear combination of the training dataset size per class plus the number of epochs trained. The coefficients of this linear combination are the parameters to be estimated.

4 Data generation algorithm

For a given training dataset, a smaller subset can sampled randomly by assigning the same probability of being included in the sample to each image. Although in general this is a good sampling strategy in many contexts, here it is not what is desired. Especially for large training datasets, the law of large numbers says that the class proportions in the sample subset will be very similar to the class distribution in the complete training dataset. In order to achieve a number of divers training datasets, both with respect to the total size as well with respect to the class proportions, an algorithm is suggested in the following. It takes some motivation from a special case of statisticial design of experiments, notably constrained space filling mixture designs. Please see [**?**] for more information on spacefilling mixture desigsn. Design of experiments is a set of methods, originating back to as early as Sir Ronald Fishers book on the topic [**?**], which all aim at collecting optimal data for fitting a specific statistical/mathematical model. This might be simple linear models (cite something) or much more complex models like nonlinear, multidimensional models [**?**], [**?**]. In contrast to many machine learning and active learning methods, design of experiments often aims at finding the minimal dataset with which a model can be fitted. Mixture designs in general are a class of designs originating out of chemistry: The composition of some ingredients to a chemical experiments needs to sum up to 100%, see for example [**?**], chapter 5.5.4 for a short introduction. The reason why this is fitting here is that, given a target dataset size n_{subset}, we want to create a number of different combinations of training datasets, which all have the same total number of training images but distributed differently according to the different classes. There are different ways to describe the optimality of a design of experiments. Here we will choose an optimality criterion which is considered space filling, the so called maximin optimality criterion, which tries to maximize the minimum (euclidean) distance between any two data points in the design. The reasoning behind this is that having a datapoint $d_2 \in D$, which are very close to a datapoint $d_1 \in D$ does not bringing additional benefit, as we have learned already the response of our target function $g(.)$ at position d_1. This choice is to a big degree subjective. We have chosen this optimality criterion here, as we want to generate a wide spread of different combinations of class counts, given a fixed overall count of training images. The design needs to be constrained, as for larger subset sizes, it might happen that a purely randomly generated combination of class counts is for some classes higher than the maximum number of available training images per class. The algorithm to find an optimal design has 2 stages: In the initilization phase, a design is constructed which is

fullfilling the following: each row summing up to the target dataset size and the constraint for the maximum number of images per class is full filled. In the second stage, this design is improved iteratively by a pointwise exchange algorith: The pair of design points with the minimal distance is found and one of them is replaced by a new, randomly generated candidate design point. If the maximin criterion for the new candidate design is improved compared to the previosly best design, the candidate design is accepted as new best design. Otherwise, the candidate design is rejected. This process is repeated a fixed number of times. Similar pointwise optimization procedures are quite common in the area of design experiments and are rooting back to early references like [**?**]. One important detail in this process is how design points are suggested, during the initialization phase as well as during the optimization phase. For the unconstrained case, candidate points can be suggested by using TBD. In the case of heavliy constrained mixture, a heuristc can be applied to increase the chance to sample a candidate fullfilling the constraint. For details, please check in the appendix.

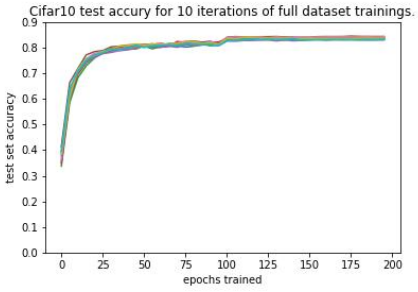
5 scaling law model fitting

- Power law scaling ([**?**]): $g_\omega(n) = \omega_1 n^{\omega_2} + \omega_3$.
- Arctan scaling: $g_\omega(n) = \frac{200}{\pi} \arctan(\omega_1 \frac{\pi}{2} n + \omega_2) + \omega_3$
- logarithmic scaling: $g_\omega(n) = \omega_1 \log(n + \omega_2) + \omega_3$
- Algebraic root: $g_\omega(n) = \frac{100n}{1+\|\omega_1 n\|^{1/\omega_2}} + \omega_3$
- For each of these models, we can replace n by a linear combination of n_c, the class counts: n
- Another model class used here is a logistic regression, as this naturally expands from a one dimensional (i.e. just using the overall training dataset size n) to the multidimensional use case (Using seperate class counts).

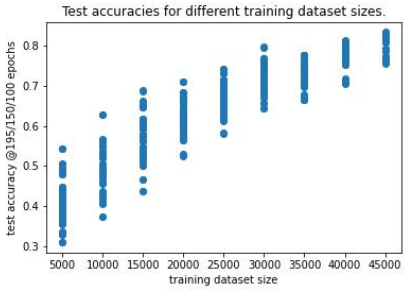
6 Experiments using CIFAR10

6.1 experimental setup

The CIFAR10 dataset [**?**] is a well known benchmark dataset for image classification. It consists of 50000 training images, in 10 classes, each class having 5000 training images of the shape 32x32x3 and 10000 test images of the same shape, also equally distributed across the 10 classes. As a model to be fitted, a standard Resnet18 architecture [**?**] has been used as standard model here, as implemented in the pytorch [**?**] modell zoo. For optimizing this model, a standard SGD optimizer with cross entropy loss has been used with $learning_rate = 0.1, momentum = 0.9$ and $weight_decay = 1e-4$, together with a learning rate scheduler, reducing the the learning at 100 (which is also clearly visible at severals accuracy vs. epoch plots, e.g. Figure 1a) and 150 epochs. For data preparation/transformations, random horizontol flipping, random cropping and batch normalization with predefined mean and standard deviation has been used. With these settings, validation dataset accuracies up to 84% have been achieved. There are for sure more advanced model architectures for the Cifar10 dataset, however we aimed at balancing an at least medium performant model with a having a reasonable training time, as the training process is repeated here a high number



(a) Full training results for 10 seperate training runs.



(b) Test accuracies vs. training dataset sizes.

Figure 1: Descriptive results for the models fitted to Cifar10 dataset.

of times. For the experiemtns done here, we applied the algorithm from section 4 here with the following settings: As subset sizes we have chosen $[5000, 10000, \ldots, 40000, 45000]$. For each subset size a design of experiments according to Algorithm **??** with 30 different settings has been created for training a function g_ω as well as a validation dataset with 15 runs per subset size. For each training dataset, the same model and hyperparameter settings has been used, training for up to 195 epochs, checking for validation accuracy every 5 epochs. To get an impression of the models fitted here, there are a number of descriptive visulizationas in Figures 1a, 1b and 2. In Figure 1a the test set accuracy for the full training dataset across different epochs and 10 repeats is shown. In Figure 1b the test accuracy at 195 training epochs for all created training dataset of different size is plotted. As expected, the performance varies more for smaller total training dataset sizes compared to larger training dataset sizes. And in Figure 2, the test accuracy is plotted over training epoch for all training datasets of 4 different training dataset sizes. Overall, the descriptive analysis did not yield any concerns in terms of outliers or unexpected behavior of the data.

6.2 model fitting results

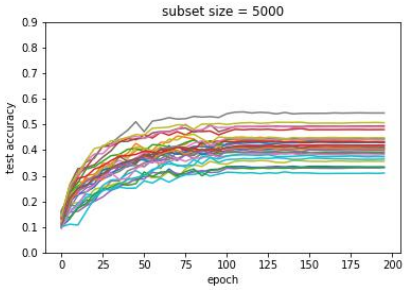
a	b	c	plane	car	bird	cat	deer	dog	frog	horse	ship	truck	epoch
-0.05	0.40	0.55	0.37	0.31	0.29	0.17	0.30	0.27	0.29	0.37	0.23	0.25	0.59

Table 1: Estimated parameters for model (0).

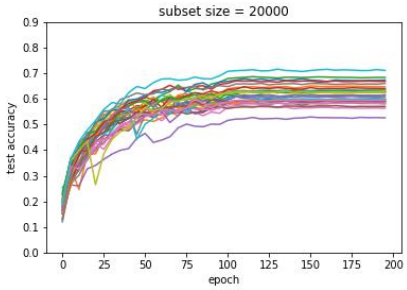
a	b	c	total_n	epoch
0.25	0.61	0.35	0.48	1.86

Table 2: Estimated parameters for model (3).

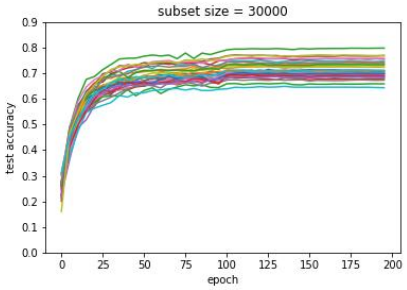
- In preparation for model fitting, the data have been transformed in the following way:
- – All model results for epochs < 10 have been removed from the dataset. Mainly because these data are highly variable.



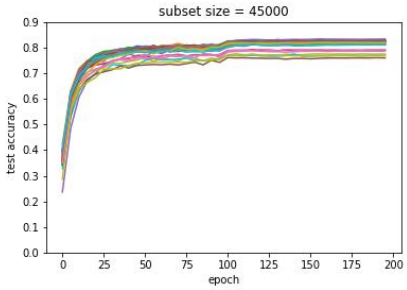
(a) training dataset size = 5000 images



(b) training dataset size = 20000 images

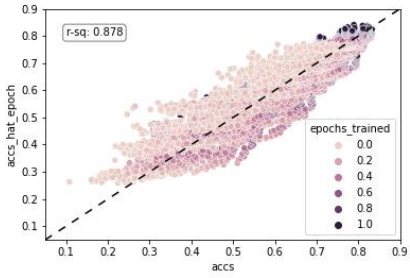


(c) training dataset size = 30000 images

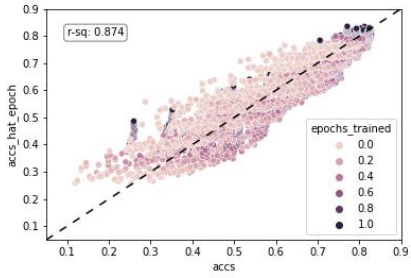


(d) training dataset size = 45000 images

Figure 2: Test accuracies vs. epochs for different number of training dataset sizes.

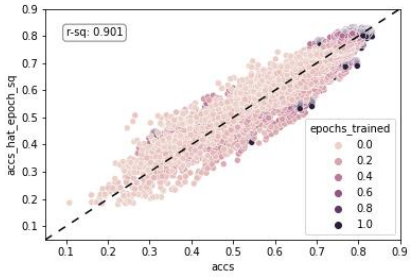


(a) training dataset size = 5000 images

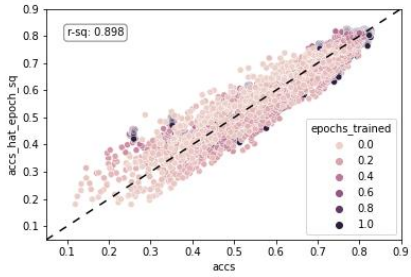


(b) training dataset size = 20000 images

Figure 3: Prediction plot for powerlaw function 0.

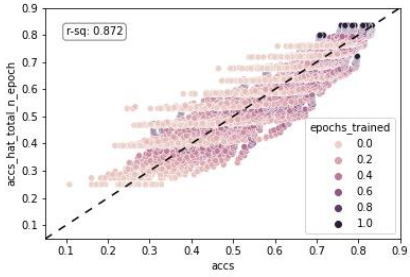


(a) training dataset size = 5000 images

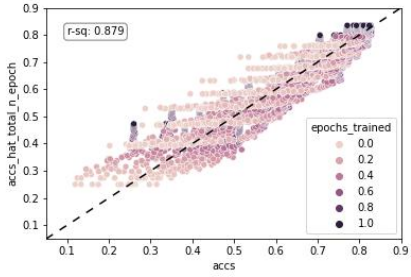


(b) training dataset size = 20000 images

Figure 4: Prediction plot for powerlaw function 1.

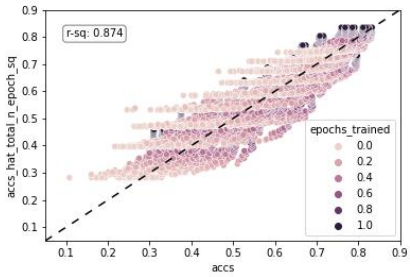


(a) training dataset size = 5000 images

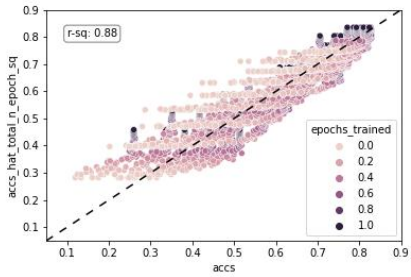


(b) training dataset size = 20000 images

Figure 5: Prediction plot for powerlaw function 3.



(a) training dataset size = 5000 images



(b) training dataset size = 20000 images

Figure 6: Prediction plot for powerlaw function 4.

a	b	c	total_n	epoch	epoch_sq
0.28	0.75	0.41	0.32	1.50	-0.32

Table 3: Estimated parameters for model (4).

  - From the individual class counts an overall training dataset size is calculated per dataset.
  - All class counts (also the total dataset size) have been scaled to be between $[0, 1]$, as well as the epochs.
  - As the test accuracies are already between $[0, 1]$ these have not been further standardized.

- Following these desecriptive plots, a number of different models are fitted, all using the above described dataset.
- In Table **??** an overview of the fitted models is given.
- All of the these models are fitted using the `curve_fit` function from the python package `scipy.optimize`.
- The resulting parameter estimates are shown in Tables **??** to 3.
- For each model, there is a corresponding pair of prediction plots (one for the train data set and one for the validation dataset) in Figures 3 to 6.
- Looking at Table **??**, the model using the individual class counts and the epoch in a linear and squared effect performs best.
- Also, for this model, the prediction plots look most reasonable, scattering rather randomly around the diagonal.

7 Discussion summary

Appendix

row	accs	plane	car	bird	cat	deer	dog	frog	horse	ship	truck	epochs	$total_n$
0	0.18	145	31	97	496	1096	307	2382	10	373	63	10	5000
1	0.21	145	31	97	496	1096	307	2382	10	373	63	15	5000
2	0.20	145	31	97	496	1096	307	2382	10	373	63	20	5000
3	0.25	145	31	97	496	1096	307	2382	10	373	63	25	5000
4	0.24	145	31	97	496	1096	307	2382	10	373	63	30	5000

Table 4: First few lines of the underlying data table used for fitting the powerlaw models.