Unsupervised Data Validation Methods for Efficient Model Training

Yurii Paniv
yurii.paniv@gmail.com

Abstract. Current state-of-the-art models require large datasets to be trained on that are unavailable for independent researchers and smaller institutions due to cost. Current literature suggests that models trained on higher-quality and smaller datasets converge to nearly the same quality. In my Ph.D. proposal, I describe the current generic methods for dataset improvement and possible solutions for unsupervised dataset validation.

1 Introduction

Training state-of-the-art speech recognition models such as Wav2Vec [1] or Whisper [2] requires nearly 56 000 hours and 680 000 hours of speech data, respectively. Let's take Wav2Vec, for example. Just for storage alone, this requires 16 bits per sample / 8 (to bytes) * 16 000 samples per sec * 56 000 hours * 60 min * 60 sec / 1024 (to KB) / 1024 (to MB) / 1024 (to GB) / 1024 (to PB) = 5.86 PB of speech data (without counting scraping that data from the Internet, preprocessing, converting that data). Assuming that this data would be stored on AWS S3, which has a pricing of 0.021 $/GB, this would cost approx. $129 199,22 per month. Even higher costs would be incurred by compute costs (particularly GPU). Let's take an example from another domain, training of large language models. BLOOM [4] was trained on 600 * A100 NVIDIA GPUs for 90 days. If this were done on a cloud GPU provider, like lambdalabs.io, one VM instance with 8x NVIDIA RTX A100 would cost 8.80 $/hr [3], which would total into 90 days * 24 hours * 600 GPUs required / 8 in one instance * 8.80 $/hr = $1 425 600 just for a single training run for one epoch.

This makes it unbearable spending for independent researchers or smaller institutions with limited financing and, as a result, makes it hard to reproduce or contribute to research.

Therefore, a question arises, how to make model training more accessible? Models mentioned above require vast amounts of data to be trained on. Training time directly depends on that amount; therefore, how can we use that data more effectively and limit its quantity but achieve a similar quality of model results?

In the following sections, I'm going to describe a possible path how to approach this:

Related work. Here I describe how other researchers tackled the problem of training models on big datasets, what issues they were dealing with, how they handled noisy data, and how they approached data cleaning.

Motivation. In this section, I describe why it's crucial to deal with this problem and the impact of solving it.

Methodology. In this section, I describe the perspective methods to achieve the goal of automatic data validation and optimize training costs using data-related optimizations.

Conclusion. This section contains a general overview of a paper, how other researchers could benefit from this, and the risks of achieving that.

2 Related work

We must divide this problem into manageable sub-problems to handle it effectively. This way, we can tackle each part and find practical solutions.

Subproblem 1. How to limit data quantity safely? For example, in Training Compute-Optimal Large Language Models (Chinchilla paper) [5], authors trained a range of models from small to big, 50+ models, and compared the results and efficiency of their training regimes. That led to the conclusion that "for every doubling of model size, the number of training tokens should also be doubled" [5], or in other words, what is a relationship between model size and required dataset size. Another finding was that modern at the time models were significantly undertrained, and the model would benefit from 4X more data, and it didn't make sense to train larger-scale models without an equally large dataset [5]. Basically, they discovered the upper limit to the data that makes sense to supply to the model.

Authors note that there is an unexplored potential regarding the quality of that data, and possible improvements could come from ensuring that training data is of high quality.

Unfortunately, this work doesn't explore the lower limit of achieving a similar-quality model, explored in The MiniPile Challenge for Data-Efficient Language Models [6].

In this paper, the author presents a method to filter the Pile dataset and achieve only a 1.9%/2.5% [6] performance drop compared to the original pre-trained checkpoints trained on 2.6x/745x the amount of data [6]. MiniPile is a 6GB subset of the deduplicated 825GB The Pile corpus. To curate MiniPile, they performed a simple, three-step data filtering process: (1) infer embeddings for all documents of the Pile, (2) cluster the embedding space using k-means, and (3) filter out low-quality clusters [6].

Clusterization is performed in a supervised manner that requires knowledge of the target domain to filter that dataset.

Unfortunately, this paper also doesn't answer the question of how exactly one can calculate the lower limit of data required for model performance but gives proof that using higher quality data wouldn't give a significant penalty versus using a large unfiltered dataset.

This leads us to the next problem.

Subproblem 2. Would higher-quality data lead to much better model outputs? This is explored in Alpaca [7] and LIMA [8] papers. In the Alpaca paper, researchers generate a dataset of 52 000 instructions using GPT-4 to develop a way for the model to follow instructions [7]. In the LIMA paper, authors created a hand-crafted dataset of just 1000 instructions that achieves results comparable to GPT-4 [8], which was trained on a much more massive dataset.

This shows that a carefully crafted dataset performs as well or even better than a larger dataset, depending on the data quality [8]. Unfortunately, this approach requires supervision and human validation.

Subproblem 3. How to check the validity of data in an unsupervised manner? This is explored in Confident Learning: Estimating Uncertainty in Dataset Labels [9]. In this paper, the authors explore the idea that current approaches to machine learning focus commonly on model predictions, not data. They present a method for dealing with noisy data, which can improve model performance substantially [9]. A method is to estimate joint distribution between noisy and uncorrupted labels.

As we can see, not every reference dataset is valid [10], and cleaning noisy labels could lead to substantial performance improvements [9]. Still, it requires training a couple of models using OOB (out-of-bag) approach [9] and validating noisy labels, which still could not be efficient for compute.

Even by extracting the most out of datasets, there is a lot to tweak in a training regiment that depends on training data.

Subproblem 4. What hyperparameters could be tweaked based on data? Torch.manual_seed(3407) is all you need: On the influence of random seeds in deep learning architectures for computer vision [11] author explores different seeds for random number generator, that give the best starting point for training machine learning models for computer vision. He scans a large number of seeds (up to 104) [11] on CIFAR 10 and also scans fewer seeds on Imagenet using pre-trained models to investigate large-scale datasets [11]. The conclusions are that "even if the variance is not very large, it is surprisingly easy to find an outlier that performs much better or much worse than the average." [11].

Possible exploration could be performed for other models to establish a good starting point to converge faster.

A disadvantage of this method is that search is performed using grid-search, but could an automatic way of searching be found? Could this be performed for other domains, such as speech recognition or language modeling? How would this transfer between different models?

As we can see, using the latest advances, we can use substantially less data to achieve similar model quality, but this requires human-supervised filtering of datasets to get rid of lower-quality data.

3 Motivation

The English language has the biggest amount of datasets (4 466) [12], but countless languages worldwide don't enjoy the same amount of resources that English has (the next language by dataset count is Chinese with 454 datasets [12]. Research in data validation and studying the relationship between the amount of data used for training models to provide good performance would be useful to serve people in those languages.

Discovering new methods of making training data-efficient could make research more accessible for independent researchers and smaller institutions, which will contribute to more innovations in machine learning.

I see a couple of unsolved problems or directions:

Can we make training of LLM efficient (reducing the amount of data required for model training 700x with performance loss of less than 2% [6]) like in the "MiniPile" paper but in an unsupervised manner? Can we make training of other models efficient? There are other problems where large datasets are required, such as speech recognition, where the commonly minimum suggested amount of data for pretraining is 5 000 hours, as in DeepSpeech [13] for the model to perform correctly. Could there be a general flow on how to prevalidate that data so that it could be effective even for low-resource languages?

Can we achieve this without major tradeoffs after exhaustive benchmarks? LIMA paper demonstrates that one could achieve similar performance with a much smaller dataset [8], but their method has disadvantages:

- It's optimized for long answers.

Authors were picking longer answers because of the suggestion that longer answers should be correct [8]; this can contribute to bias for human reviewers.

- It required a rigorous human selection process.

Can we validate data in an unsupervised/semi-supervised way to boost model performance safely? Most methods require some supervision, could there be a pretraining step as part of the training process that assesses patterns in data and decides how "important" they are?

Can we find other techniques to apply to trainer or data?

Torch.manual_seed(3407) paper demonstrated that there are outliers in random seeds [11]. Can we look for a way to search for the best training seed for a particular task or data automatically and efficiently?

Which data preparations are effective? what is the impact of every data preparation step? What is the impact of simple steps like OOB (out-of-bag) training, clustering, and filtering?

What makes the most sense in terms of time and compute efficiency to invest resources into?

How to get a measure of a good dataset before training? The training runs of recent models are large [1][2][4], so how to know that dataset is of good quality before we invest resources into it?

In my Ph.D. research, I want to focus on generic unsupervised methods for improving dataset quality, data-related optimizations, and filtering to reduce the data required for model training with minimal performance loss. These methods could be applied to different domains, such as speech recognition, language modeling, image recognition, etc. They should speed up convergence, reducing compute and storage costs and making training large models more accessible.

4 Methodology

4.1 Evaluation and Baseline

Before starting to work on methods how to minify a dataset, we need to establish fail criteria and evaluation baseline.

Firstly we need to take an existing machine learning model that requires a big dataset to be trained on. Let's take GPT-2 [14] (which was trained on 40 GB of text), for example. GPT-2 was chosen because of its size and ability to be trained on consumer hardware like NVIDIA RTX 3090 in a few weeks.

Firstly I would take the same dataset and tokenizer as OpenAI used and their model checkpoint as a baseline. As a side comparison, I would use OpenGPT-2 [15], which tried to replicate and cost $50 000 (to do it fast for the 1.5B release). Then experiment to minify this dataset for the English language it was trained on and compare the performance of models. To compare performance, I'll use the same measures as described in the paper, but firstly will focus on perplexity [16] - a probability of token appearance based on context.

Model performance consists of several measures:

1. Compare loss between different training runs.

Limitations of this metric: it's not a direct performance measure, but is commonly a good proxy.

2. Compare perplexity between different models.

3. Human evaluation.

Evaluate performance on machine translation, question answering, reading comprehension, and summarization using MOS (mean opinion score) metric.

How to set up MOS: enable a platform where human annotators would rate responses for the relevant task and calculate an average of their scores against a baseline model.

4. Other metrics from the paper, domain-specific, such as BLEU.

Then these findings would be used for training models for lower-resource languages, such as Ukrainian.

Other appropriate target domains would be speech recognition (like wav2vec) or image synthesis (like Stable Diffusion) with their respective metrics, like WER (word error rate).

4.2 Novelty filtering

There is a great amount of data duplications and low-quality clusters in publicly scraped datasets, such as The Pile [6]. Problems occur due to duplicating news articles, citations and etc. I want to develop a method how to assign a score for every element of data about how novel it is. I got inspiration from REINFORCE [17] method by Williams. Instead of gathering rewards and applying policy gradient, this method would be trained to separate between known and unknown instances using this regiment:

Train a classifier on text data in an unsupervised manner for 1 epoch.

1. The first batch would be assigned label 1 (unseen).

2. In the next batch, data assigned label 1 previously would be labeled 0 (seen), with the next incoming data assigned 1, and so on.

The hypothesis is that this model would look for redundant patterns in data. A similar principle was explored in work by Collins, A. M., & Loftus in A spreading-activation theory of semantic processing [18]. In that work, a simple model

generalized to separate between known and unknown data combinations [18] when provided in an online learning fashion with human input.

Having the result of that model, we can use the resulting score in several ways:

1. We can filter data in the source dataset based on its novelty, which will be measured in output probability from 0 to 1. The only manual step required is assigning a threshold to filter non-novel data.

2. Apply the score directly while training the target model like GPT-2. For example, multiply the gradients of a batch by its novelty score to avoid overfitting the model.

3. Perform dataset analysis using novelty score, look for score distribution to assess the diversity of the dataset, and whether novelty score could be a good measure of it.

4. Use novelty score as a policy for unsupervised online learning to enlarge the dataset with more diverse examples from the Internet.

Potential problems:

1. High variance of gradients.

As noted in REBAR [19] paper, gradient estimators could have high variance, and the model would not converge to an optimal solution [19]. In case of failure using REINFORCE modification, it makes sense to look for other novelty measures.

2. Outliers as novel instances.

For different tasks, there could be different needs for novelty tuning. Novel instances of data could be outliers (or plainly wrong data) that one wants to get rid of. In that case, it would make sense to apply methods such as Cleanlab to novel instances and clear or review data that would be offending the typical distribution of a class probability.

4.3 In search of the good starting seed

As demonstrated in “Torch.manual_seed(3407)” is all you need paper, there are outliers in terms of starting seed for specific tasks [11]. In the scope of the search of this work, I want to answer the following questions:

- is it possible to find such an outlier for other tasks, like language modeling or speech recognition?
- can you detect them automatically?

Methodology to answer this question:

1. Look for a link between the final loss and initial loss for initialized weights, so while searching for the best seed, we can use less compute.

2. Bruteforce initial seeds for random numbers and look for outliers.

3. Train a neural network to look for random seed using reinforcement learning and loss from the model.

4. Test whether the same random-seed detector could be transferred to other tasks to be more effective, do we need to add more input features like model architecture?

Potential problems with implementing that:

1. Random number generator implementations differ based on CPU manufacturer, architecture, and system components. A check would be needed to ensure this can be reproduced from one machine to another.

2. There could be no such correlations for other tasks as for image recognition.

3. Model could not converge due to the stochastic nature of outputs.

Another approach could be to calculate the distribution of values in existing trained models and initialize random numbers using mean or median of distribution and its variance after measuring the model's performance against trained using a set of random seeds.

4.4 Other open questions

There are other open questions worth researching that could serve as an expansion for my work on data validation.

- How small can a dataset be to be effective?

For Chinchilla, it's possible to calculate the maximum amount of tokens the model could learn from [20]. Could there be a method to calculate the minimum amount of data for model to be trained on?

- What could be a good metric for a good dataset? How to measure it?
- Which data preparations are effective? What is the impact of every data preparation step, such as OOB training, clustering and filtering?
- What method makes the most sense in terms of time and compute efficiency to invest resources into?

5 Conclusion

In my Ph.D. research proposal, I described my vision of how to train large machine learning models efficiently in terms of the amount of data required to achieve comparative model quality. As a result of my work, my Ph. D. research should offer a way to increase the efficiency of a used dataset in terms of its size and total training cost and help other researchers to improve the quality of their datasets and, in turn, spend more time working on optimizing model inference and architecture, could help to bring SOTA models that currently require large datasets to low-resource languages. As a side result, due to reduced compute requirements, this could reduce the carbon footprint of model training.

A potential risk would be to tackle a high variance of modification of REINFORCE algorithm [19] if it fails to capture "novelty" of data. In that case, I would look into increasing the bias of an algorithm by using regularization measures, such as using data from previous batches as "known" to improve training stability. In general, my Ph.D. research could evolve into a generic unsupervised method for validating data before or during model training.

References

1. Baevski, A., Zhou, H., Mohamed, A., & Auli M. (2020). wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations https://doi.org/10.48550/arXiv.2006.11477
2. Radford, A., Kim, J.W., Xu, T., Brockman, G., McLeavey, C., & Sutskever, I. (2022). Robust Speech Recognition via Large-Scale Weak Supervision https://doi.org/10.48550/arXiv.2212.04356

3. lambdalabs.com pricing page, https://lambdalabs.com/service/gpu-cloud#pricing, last accessed 07.08.2023
4. Scao, T.L., Fan, A., Akiki, C., Pavlick, E., Ilić, S., Hesslow, D., et al. (2022). BLOOM: A 176B-Parameter Open-Access Multilingual Language Model https://doi.org/10.48550/arXiv.2211.05100
5. Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., et al. (2022) Training Compute-Optimal Large Language Models https://doi.org/10.48550/arXiv.2203.15556
6. Kaddour, J. (2023) The MiniPile Challenge for Data-Efficient Language Models https://doi.org/10.48550/arXiv.2304.08442
7. Stanford Alpaca repository, https://github.com/tatsu-lab/stanford_alpaca, last accessed 07.08.2023
8. Zhou, C., Liu, P., Xu, P., Iyer, S., Sun, J., Mao, Y., et al. (2023) LIMA: Less Is More for Alignment https://doi.org/10.48550/arXiv.2305.11206
9. Northcutt, C., Jiang, L., Chuang, I. (2019) Confident Learning: Estimating Uncertainty in Dataset Labels https://doi.org/10.48550/arXiv.1911.00068
10. Label Errors in ImageNet, found using CleanLab and validated by human annotators, https://labelerrors.com, last accessed 07.08.2023
11. Picard, D. (2021) Torch.manual_seed(3407) is all you need: On the influence of random seeds in deep learning architectures for computer vision https://doi.org/10.48550/arXiv.2109.08203
12. English datasets available on HuggingFace, https://huggingface.co/datasets?language=language:en&sort=trending, last accessed 07.08.2023
13. Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E. (2014) Deep Speech: Scaling up end-to-end speech recognition https://arxiv.org/abs/1412.5567
14. Language models are unsupervised multitask learners, https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf, last accessed 07.08.2023
15. OpenGPT-2, independently reproduced GPT-2, https://medium.com/@vanya_cohen/opengpt-2-we-replicated-gpt-2-because-you-can-too-45e34e6d36dc, last accessed 07.08.2023
16. Explanation of perplexity from HuggingFace, https://huggingface.co/docs/transformers/perplexity, last accessed 07.08.2023
17. Williams, R.J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach Learn 8, 229–256 (1992). https://doi.org/10.1007/BF00992696
18. Collins, A. M., & Loftus, E. F. (1975). A spreading-activation theory of semantic processing. Psychological Review, 82(6), 407–428. https://doi.org/10.1037/0033-295X.82.6.407
19. Tucker, G., Mnih, A., Maddison, C. J., Lawson, J., & Sohl-Dickstein, J. (2017). REBAR: Low-variance, unbiased gradient estimates for discrete latent variable models https://doi.org/10.48550/arXiv.1703.07370
20. Karpathy's implementation of scaling formula from Chinchilla, https://github.com/karpathy/nanoGPT/blob/master/scaling_laws.ipynb, last accessed 07.08.2023