A Simple and Efficient Baseline for Data Attribution on Images

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Data attribution methods play a crucial role in understanding machine learning models, providing insight into which training data points are most responsible for model outputs during deployment. However, current state-of-the-art approaches require a large ensemble of as many as 300,000 models to accurately attribute model predictions. These approaches therefore come at a high computational cost, are memory intensive, and are hard to scale to large models or datasets. In this work, we focus on a minimalist baseline that relies on the image features from a pretrained self-supervised backbone to retrieve images from the dataset. Our method is model-agnostic and scales easily to large datasets. We show results on CIFAR-10 and ImageNet, achieving strong performance that rivals or outperforms state-of-the-art approaches at a fraction of the compute or memory cost. Contrary to prior work, our results reinforce the intuition that a model's prediction on one image is most impacted by visually similar training samples. Our approach serves as a simple and efficient baseline for data attribution on images.

1 Introduction

The effectiveness of a machine learning system's performance hinges on the quality, diversity, and relevance of the data it is trained on [11, 28]. In various real-world machine learning systems, for example in healthcare or finance, we often ask questions like, "Which training samples influenced this prediction?" or "How sensitive is this model's prediction to changes in the training data?" Counterfactual insights enable us to assess the impact of hypothetical changes in the data distribution, which in turn helps us understand the basis of the model's decisions and how to change the decision in the event of an error.

These questions motivate research on *data attribution* methods, which focus on understanding which data points most strongly influence a model's outputs. Data attribution methods have been applied to applications such as debugging model biases [16, 24, 27], fairness assessment [2], and active learning [21]. In principle, data attribution can be done perfectly by a brute-force leave-k-out strategy; simply train the model from scratch many times, removing k data points each time. The user can then examine the impact of each data point by examining how the corresponding ablated model differs from the original. Clearly, this procedure is intractable for any realistic problem as there are innumerable subsets, and training even a single machine learning model can be almost prohibitively expensive. The goal of data attribution research therefore is to approximate this gold standard metric as closely as possible while simultaneously using as little computation as possible. As such, the field of data attribution is all about trade-offs between accuracy, runtime, and memory.

Existing data attribution approaches gain insights into model behaviors by scraping information from the learning algorithm, such as logits [16] or gradients [18, 24]. Despite this, these techniques still require re-training multiple models on different data subsets, or other compute and memory

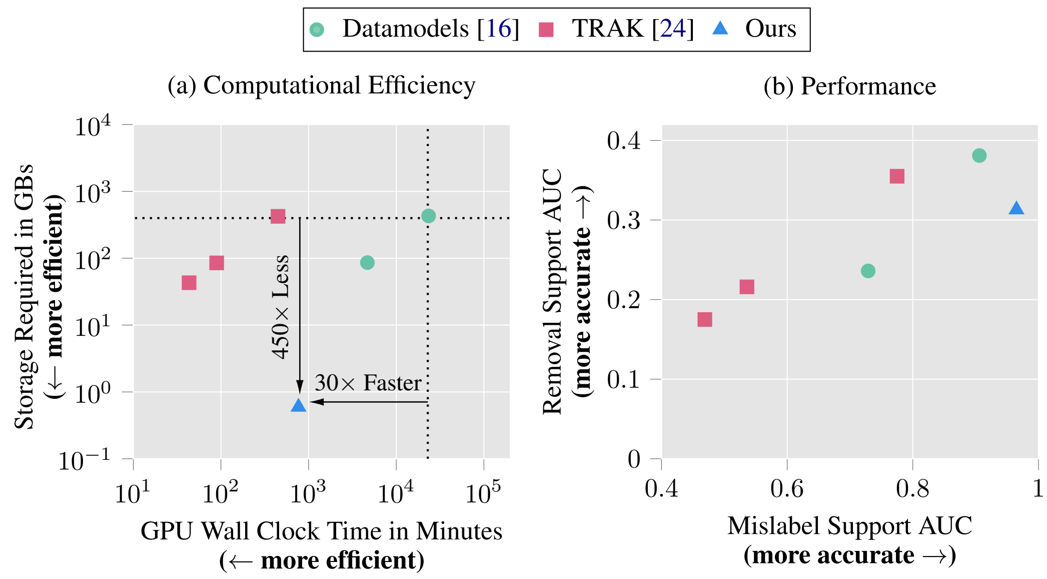

Figure 1: **Our proposed baseline approach for data attribution achieves high performance while improving computational efficiency**. Figure (a) shows the wall-clock time on an RTX A6000 GPU on the x-axis and memory requirements in GBs on the y-axis respectively (see Appendix A.1 for details). Figure (b) shows performance on two metrics measuring the method's accuracy to make counterfactual predictions (details about the metrics are discussed in Section 2.1.)

intensive strategies for better efficacy [16, 7, 18, 24]. Current data attribution approaches quickly become intractable as datasets become larger [1, 24] and applications become more realistic, such as attribution for LLMs [10].

In this work, we present a simple approach that outperforms the current state of the art in terms of compute-accuracy trade-offs, and often in terms of raw performance numbers as well. Given a test image, we use the feature space of a single self-supervised model to retrieve similar images, revealing a compelling association between data attribution and *visual similarity*. In contrast to existing methods that involve unwieldy model ensembles and extensive computation, our approach shifts the spotlight directly onto the data. Building on prior research, we focus on counterfactual prediction [16, 24] for evaluating data attribution techniques. Based on the intuition that data inherently shapes model behavior, our method does not use any information about the model training process, and yet still rivals the performance of state-of-the-art approaches that do, while using a tiny fraction of the computational resources. Our work shows that, contrary to previous work [16, 24], feature representations can serve as a robust baseline for data attribution methods.

2 Problem Setting

We first define our notation and then discuss evaluation criteria used for data attribution approaches. We borrow notation and evaluation criteria from Ilyas *et al.* [16] and Park *et al.* [24].

Notation: Let $S = \{z_1, z_2, \dots z_n\}$ denote a set of training samples. Each sample $z_i \in S$ represents $z_i = (x_i, y_i)$, where x_i signifies the input image and y_i represents the associated ground truth label. We use z_t to denote an arbitrary evaluation sample not present in the training set. We denote a data attribution approach as a function $\tau(z, S) \in \mathbb{R}^n$. This function operates on any sample z and a training set S, generating a score for each sample within the set S. These scores highlight the relative positive or negative impact of individual training samples on the classification of the input sample z.

2.1 Evaluating Attribution Methods

Recent research primarily concentrates on evaluating the performance of data attribution methods through the lens of their capacity to provide accurate counterfactual predictions [24, 16]. While these metrics can be computationally demanding, they represent a valuable proxy for assessing the

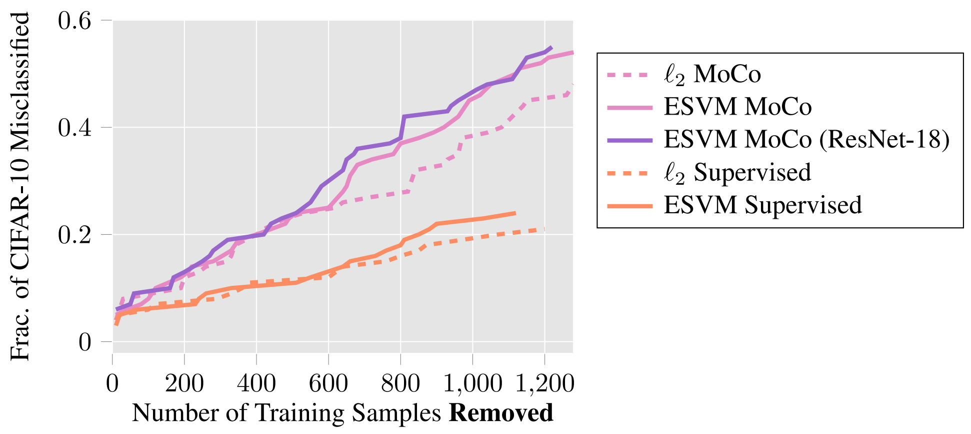

Figure 2: **Self-supervised features are more effective than supervised and are best compared using an ESVM**. Self-supervised features from MoCo can be used to find smaller data support than standard supervised features. For a larger fraction of test samples, ESVM distance is more effective than ℓ_2 distance at ranking train images to select smaller data removal support.

effectiveness of attribution approaches. In our work, we replicate the approach presented in Ilyas *et al.* [16] and focus on **data brittleness**. Data brittleness metrics leverage attribution techniques to answer the following question: "*To what extent are model predictions sensitive to modifications in the training data?*" Hence, these metrics serve as a means of estimating counterfactual scenarios. To quantify data brittleness, we focus on two distinct types of data support for a validation sample z_t. We explain these below:

Data Removal Support: The smallest subset R_r, that when removed from the training set S, causes an average training run of the model to misclassify z_t.

Data Mislabel Support: The smallest training subset R_m, whose mislabeling causes an average training run of the model to misclassify z_t. For each training sample in R_m, we change the labels to the second-highest predicted class for z_t.

For a validation sample z_t and a data attribution approach $\tau(z, S)$, we rank the training samples based on decreasing order of positive influence on z_t. Then, based on the ranking, we iteratively select and modify a subset of training data. We perform this search, over different subsets to compute the smallest training subset that can cause z_t to be misclassified. Ilyas *et al.* [16] check only subsets with certain discrete sizes to keep costs manageable. We instead propose to perform a **bisection search** to approximate the search for the smallest subset, yielding more accurate results. The exact algorithm and details are discussed in Appendix A.4.

Intuitively, a better data attribution approach should find a smaller subset of training samples that can misclassify z_t. We estimate these metrics over a set of validation samples and plot the cumulative distribution (CDF), which represents the probability that a sample's label can be flipped as a function of the data subset size. In Fig. 1, we compare the Area Under Curve (AUC) of the CDF for the metrics described above across our approach and other attribution methods.

Linear Datamodeling Score (LDS) is another related metric used for the evaluation of data attribution methods [16, 24]. LDS metric focuses on counterfactual predictions for *arbitrary* changes in training data. In contrast, data brittleness serves to quantify the accuracy of counterfactual predictions using *targeted* changes to training data based on a specific validation sample. Thus, the latter metric serves as a better proxy for the data attribution method's usefulness as a debugging tool. In this work, we emphasize performance on data brittleness and provide results for the LDS metric in Appendix A.8.

3 Our Approach & Baselines

3.1 Our Design Choices

Our approach utilizes a neural network to extract features from a validation sample z_t and each training sample in S. Then, we compute attribution scores by measuring the distance in feature space between z_t and each training sample in S. Prior works have tried similar approaches and claimed them to be ineffective for counterfactual estimation [24, 16]. Below, we describe various components of our approach that affect performance.

Feature extractor. We find that the learning paradigm used to train a feature extractor heavily influences the estimation of data support. For example, embeddings from a ResNet-9 trained using a self-supervised learning objective (MoCo, [14]) can be used to find smaller support sets than the same model trained in a supervised manner (See ℓ_2 MoCo vs ℓ_2 Supervised in Fig. 2). With the exception of DINO [3], all self-supervised feature extractors perform better than their supervised counterpart (see Appendix A.2 Fig. 6). We found that MoCo features outperform other self-supervised approaches in both data removal support and mislabeling support scenarios, leading us to select a MoCo model as our preferred feature extractor.

Subset of train images. In Appendix A.3 Fig. 8, we show that choosing a support set from training images of class the same class y as the target $z_t = (x, y)$ is critical, *i.e.* given a target image of an airplane, we only rank airplane training images.

Distance function. When measuring the distance between two embeddings, Euclidean distance (ℓ_2) is a common choice [16, 24]. However, we find that measuring distance as distance to the hyperplane of an Exemplar SVM (ESVM) improves image similarity [22]. To compute this metric, we train a linear SVM using one positive sample (the target embedding) and treat all other samples (the remaining embeddings of the same class) as negative samples. In this way, the decision boundary, and consequently the distance function, is defined largely by unique dimensions of the target with respect to all embeddings of the same class. In Fig. 2, we demonstrate how using distance to the hyperplane of an ESVM yields better removal support estimates than ℓ_2 distance.

3.2 Baselines

Datamodels [16]: In the *Datamodeling* framework, the end-to-end training and evaluation of deep neural networks is approximated with a parametric function. Surprisingly, optimizing a linear function is enough to predict model outputs reasonably well, when given a training data subset. By collecting a large dataset of subset-output pairs, [16] demonstrate that such a linear mapping can accurately predict the correct-class margin. Datamodels were shown to be effective at counterfactual predictions but are prohibitively expensive, requiring the training of hundreds of thousands of models (300,000 in the original work) to generate optimal subset-output data. Unfortunately, this limitation makes Datamodeling intractable for all but small toy problems.

TRAK [24]: By approximating models with a kernel machine, *Tracing with the Randomly-projected After Kernels* (TRAK) makes progress toward reducing the computational cost of data attribution by reducing dimensionality with random projections and ensembling over multiple models. However, the method tends to only work well with more than a dozen model checkpoints and a large projection dimension for the model gradients, the storage of which can surpass 80GB when using a ResNet-9 on CIFAR-10. Compared to Datamodels, TRAK gains in runtime are paid for in storage space.

4 Experiments

4.1 Experimental Setup

Training Setup: For CIFAR-10 [20], we train ResNet-9 [1] and MobileNetV2 [26] models. We randomly selected 100 validation samples, in a class-balanced manner for our brittleness metrics. For CIFAR-10, we remove or mislabel a maximum of 1280 training samples for each validation sample. Our training setup is similar to [16]. For ImageNet [6], we train ResNet-18 [15] models. We randomly selected 30 validation samples, from a subset of validation samples that are not misclassified by 4 ResNet-18 models on average. For ImageNet, removed or mislabeled a maximum of 1000 training samples for each validation sample. For our approach, we always use a single model. We denote baselines using N models as Datamodels (N) or TRAK (N). Details about the baselines and our setup are provided in Appendix A.9.

[1]https://github.com/wbaek/torchskeleton/blob/master/bin/dawnbench/cifar10.py

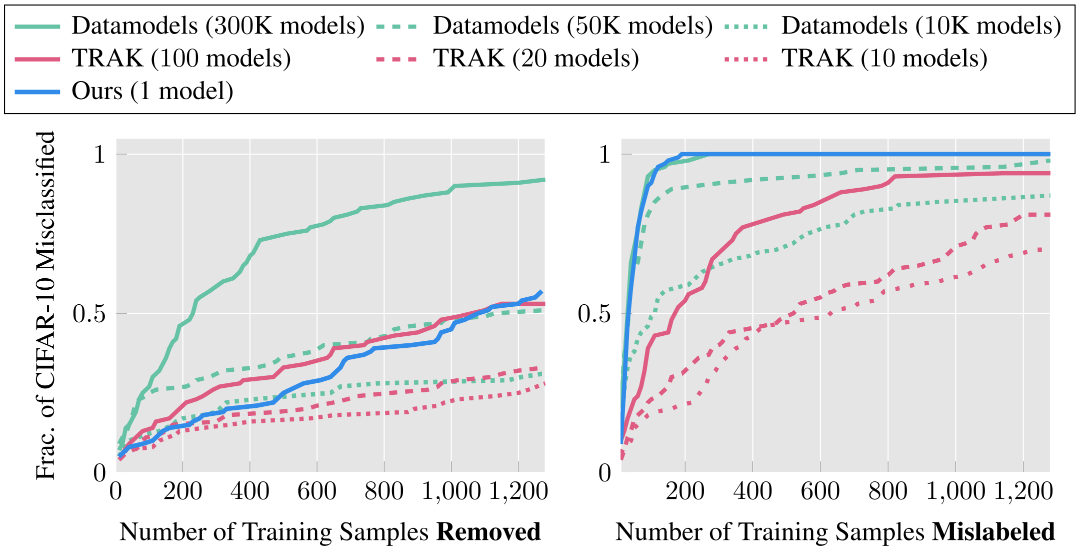

Figure 3: **Our baseline approach uses only a single model and outperforms TRAK and Datamodels using 20 and 10,000 models for data brittleness metrics.** We plot the cumulative distribution for data brittleness metrics on 100 random CIFAR-10 test samples using various attribution approaches. We outperform TRAK (20) [24] and Datamodels (10K) [16] for data removal support. We also beat TRAK (100) and perform equivalent to Datamodels (300K) for data mislabel support.

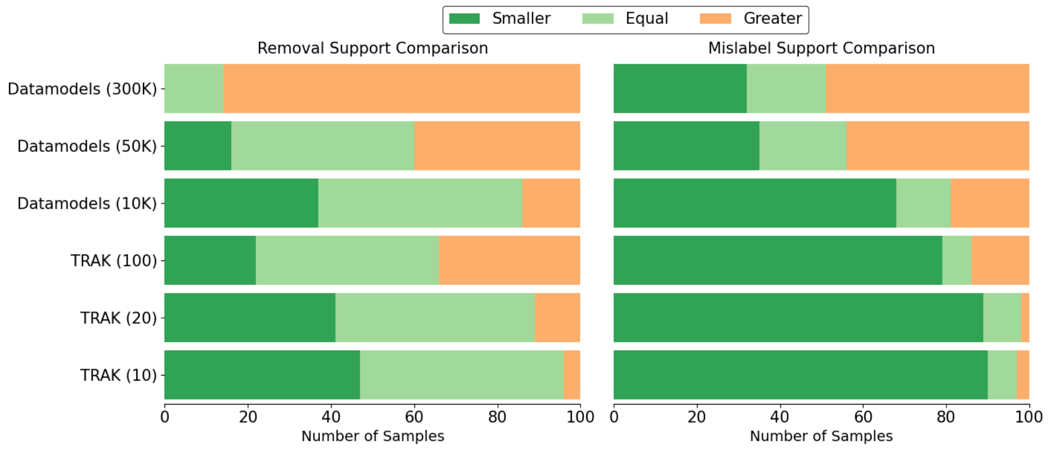

Figure 4: Compared to instances of Datamodels and TRAK, we check whether our data support estimates are smaller, equal, or larger for all 100 CIFAR-10 validation samples. For 32 samples, our approach can find smaller data mislabel support compared to Datamodels (300K). Even for data removal, our approach can find an equivalent support estimate to Datamodels (300K) for 14 samples.

4.2 CIFAR-10 Data Brittleness

In Fig. 3, we present the distribution of estimated data removal values for CIFAR-10. Our findings reveal that employing a single model with a MoCo backbone [14] for data removal support proves more effective than employing Datamodels with 10,000 models and TRAK with 20 models. Our approach and Datamodels (10K) identify that 23% samples can be misclassified by removing fewer than 500 (example-specific) training samples while TRAK (20) can only identify 16%. For support sizes up to 1280 images, our approach identifies 55% of validation samples, whereas TRAK (20) and Datamodels (10K) can only identify 28% and 31% samples respectively.

In Fig. 3, we also depict the distribution of estimated data mislabel support for CIFAR-10. Here, our approach outperforms TRAK (100) and approaches the performance of Datamodels (300K). Here, our approach identifies 47% of CIFAR-10 validation samples that can be misclassified by mislabeling less than 30 training samples! In contrast, TRAK (100) performs poorly identifying only 20% of

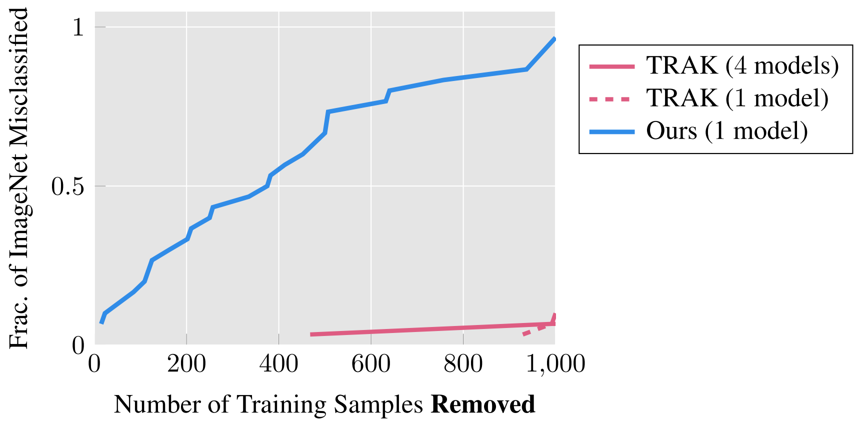

Figure 5: **Our method yields better upper bounds on support size compared to TRAK-4, which requires more storage than the ImageNet dataset itself.** We estimate data removal support for 30 random ImageNet validation samples and plot the CDF of estimates.

these samples. DataModels (300K) can identify 50% of validation samples marginally surpassing our performance.

In Fig. 4, we further inspect how well our baseline approach works for each validation sample. We compare the individual estimated support sizes for all 100 samples using our approach versus other baselines. Our estimated data removal support is smaller than those of Datamodels (50K) for 16% of the samples. For 44% of the samples our data removal estimates match TRAK and Datamodels (50K). For data mislabel support, our approach finds a smaller support estimate than Datamodels and TRAK for 32% and 79% of the validation samples.

While our baseline approach cannot outperform Datamodels (300K) on data removal, our performance on the data mislabel support is nearly the same. Our baseline approach of using a single self-supervised model can thus serve as a simple, compute, and storage-efficient alternative to estimate data brittleness.

4.3 ImageNet Data Brittleness

In Fig. 5, we show our results for data removal on ImageNet. Our results show that for 4 and 16 of the 30 validation samples our estimated data removal support is less than 16 and 130 training samples respectively. In contrast, TRAK (1) and TRAK (4) do not scale well to ImageNet at all and provide much looser data removal estimates. We again emphasize that even scaling to TRAK with 10 models would require around 400 GB of storage space, by our estimate. This highlights the scalability of our baseline approach where a single self-supervised MoCo backbone can provide more accurate data removal estimates than other existing data attribution methods.

4.4 Other Experiments

In A.6, we discuss how attribution scores from our approach and other baselines transfer across architectures. In A.5, we discuss the role of visual similarity across different attribution approaches.

5 Conclusion

Data attribution approaches are computationally expensive and can be prone to inaccuracy. While these approaches exhibit promise and capability, their scalability to large-scale models remains uncertain. Our work highlights the importance of visual similarity as a baseline for counterfactual estimation, providing valuable insights into data attribution. Our approach demonstrates scalability and accuracy, particularly in attributions for ImageNet, where it outperforms other state-of-the-art methods while maintaining manageable compute and storage requirements. Remarkably, our approach achieves these results without any reliance on training setup details, target model parameters, or architectural specifics. Our work shows that strong data attribution can be achieved solely based on knowledge of the training set.

References

[1] Samyadeep Basu, Phil Pope, and Soheil Feizi. Influence functions in deep learning are fragile. In *International Conference on Learning Representations*, 2021.

[2] Emily Black and Matt Fredrikson. Leave-one-out unfairness. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 285–295, 2021.

[3] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021.

[4] Guillaume Charpiat, Nicolas Girard, Loris Felardos, and Yuliya Tarabalka. Input similarity from the neural network perspective. *Advances in Neural Information Processing Systems*, 32, 2019.

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[7] Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. *Advances in Neural Information Processing Systems*, 33:2881–2891, 2020.

[8] Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *International conference on machine learning*, pages 2242–2251. PMLR, 2019.

[9] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

[10] Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, Evan Hubinger, Kamilė Lukošiūtė, Karina Nguyen, Nicholas Joseph, Sam McCandlish, Jared Kaplan, and Samuel R. Bowman. Studying Large Language Model Generalization with Influence Functions. *arxiv:2308.03296[cs, stat]*, August 2023.

[11] Alon Halevy, Peter Norvig, and Fernando Pereira. The unreasonable effectiveness of data. *IEEE intelligent systems*, 24(2):8–12, 2009.

[12] Zayd Hammoudeh and Daniel Lowd. Identifying a training-set attack's target using renormalized influence estimation. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 1367–1381, 2022.

[13] Zayd Hammoudeh and Daniel Lowd. Training data influence analysis and estimation: A survey. *arXiv preprint arXiv:2212.04612*, 2022.

[14] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[15] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2015.

[16] Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. Datamodels: Understanding predictions with data and data with predictions. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 9525–9587. PMLR, 17–23 Jul 2022.

[17] Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nick Hynes, Nezihe Merve Gürel, Bo Li, Ce Zhang, Dawn Song, and Costas J Spanos. Towards efficient data valuation based on the shapley value. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1167–1176. PMLR, 2019.

[18] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR, 2017.

[19] Stephen Kokoska and Daniel Zwillinger. *CRC standard probability and statistics tables and formulae*. Crc Press, 2000.

[20] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[21] Zhuoming Liu, Hao Ding, Huaping Zhong, Weijia Li, Jifeng Dai, and Conghui He. Influence selection for active learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9274–9283, 2021.

[22] Tomasz Malisiewicz, Abhinav Gupta, and Alexei A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *ICCV*, 2011.

[23] Horia Mania, John Miller, Ludwig Schmidt, Moritz Hardt, and Benjamin Recht. Model similarity mitigates test set overuse. *Advances in Neural Information Processing Systems*, 32, 2019.

[24] Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. Trak: Attributing model behavior at scale. In *International Conference on Machine Learning (ICML)*, 2023.

[25] Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 33:19920–19930, 2020.

[26] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[27] Harshay Shah, Sung Min Park, Andrew Ilyas, and Aleksander Madry. Modeldiff: A framework for comparing learning algorithms. In *International Conference on Machine Learning*, pages 30646–30688. PMLR, 2023.

[28] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852, 2017.

[29] Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. An empirical study of example forgetting during deep neural network learning. *arXiv preprint arXiv:1812.05159*, 2018.

[30] Chih-Kuan Yeh, Joon Kim, Ian En-Hsu Yen, and Pradeep K Ravikumar. Representer point selection for explaining deep neural networks. *Advances in neural information processing systems*, 31, 2018.

A Appendix

A.1 Compute Time and Storage Requirements

For our compute time estimates, we use NVIDIA RTX A6000 GPUs and 4 CPU cores. We describe how we estimate the wall-clock time, and storage requirements for each method below -

- **Datamodels:** We only take into account the storage and compute cost of training models. The additional cost of estimating datamodels from the trained models, requires solving linear regression whose computational costs are negligible compared to training the models. For compute and storage requirement estimates, we train 100 ResNet-9 models on random 50% subsets of CIFAR-10 and extrapolate to estimate the training time and storage required for 10,000 and 50,0000 models shown in Fig. 1.
- **TRAK:** We use the authors' original code [2] to train, and compute the projected gradients for CIFAR-10 using ResNet-9 Models using a projection dimension of 20480. For storage requirements, we take into account storage used by model weights, and the projected gradients. The results in Fig. 1, show the compute and storage using 10, 20 and 100 models.
- **Ours:** We use Lightly library [3] benchmark code to train a MoCo model using a ResNet-18 backbone on CIFAR-10 for 800 epochs. The results in Fig. 1 show the wall-clock training time for the model, and extracting the features from CIFAR-10 and the storage requirements for model weights.

To calculate the storage requirements, we factor in the storage space necessary for retaining the trained model weights, as they are essential for computing influence on new validation samples across all attribution methods.

A.2 Additional Self-Supervised Features

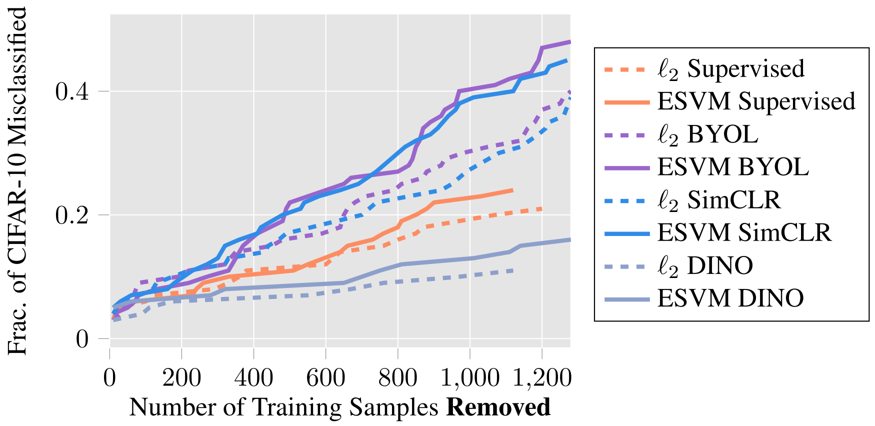

Figure 6: We estimate data removal support for 100 random CIFAR-10 test samples and plot the CDF of estimates.

In addition to utilizing features from MoCo in Section 3, we test our choice of distance function on ResNet-18 features from other self-supervised methods trained on CIFAR-10. In particular, we evaluate BYOL [9], SimCLR [5], and DINO [3] at estimating data removal support in Fig. 6 and mislabel support in Fig. 7. With the exception of DINO, self-supervised features from BYOL and SimCLR outperform the supervised baseline at estimating data removal support. Additionally, we see that in all cases using ESVM distance is more effective than using ℓ_2 distance to compare features.

[2]https://github.com/MadryLab/trak

[3]https://github.com/lightly-ai/lightly

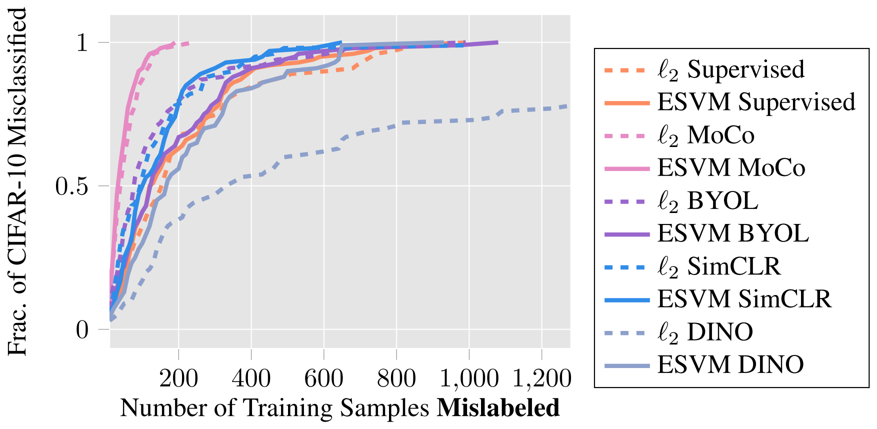

Figure 7: We estimate data mislabel support for 100 random CIFAR-10 test samples and plot the CDF of estimates.

A.3 Additional Justification for Chosen Subset of Train Images

For a target sample z_t, data attribution approaches rank the training samples based on decreasing order of positive influence on z_t. For our method, a design choice was whether to rank training samples from all classes or from a selected subset of the training data. One reasonable subset was to select training samples from the same class as the target test sample. In Fig. 8, we show that selecting from the same class is more effective when estimating britteness scores. We maintain this choice for all our experiments.

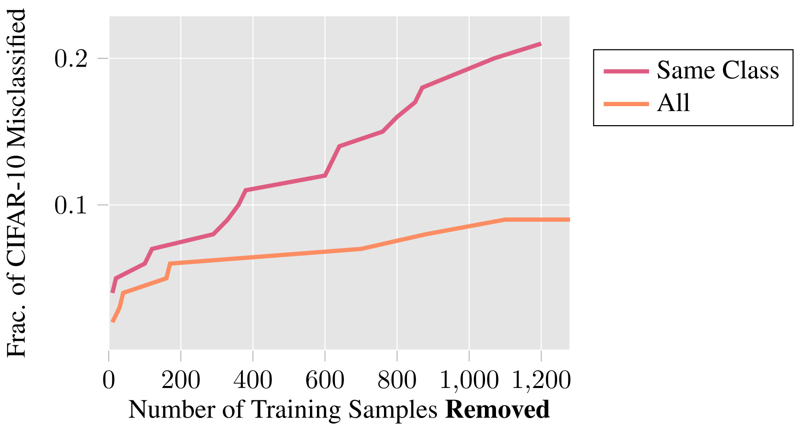

Figure 8: Choosing removal support from all training images is less effective than selecting from the same class as the target image.

A.4 Computing Data Support

We use bisection search to estimate data support. The use of bisection search is supported by the observation that several data attribution approaches are additive [24], where the importance of a subset of training samples is defined as the sum of each of the samples in the subset. To compute data removal support, we remove M samples (chosen using each attribution method) from the training data and log whether the resulting model misclassifies the target sample. For data mislabeling support, we mislabel M samples (chosen using each attribution method) from the training data and assign a new label corresponding to the highest incorrect logit.

A detailed summary of our bisection search is in Algorithm 1. A key step is $\mathrm{CounterfactualTest}(f, S, I_{\mathrm{attr}}[:\ M])$ which returns the average classification of N_{test} independent training runs where f_θ is trained on the subset $R = \{z_i | z_i \in S \text{ and } i \notin I_{\text{attr}}[: M]\}$. In other words, for computing data removal support, f_θ is trained on a subset of S that does not include the first M indices of I_{attr}. For computing mislabeling data support, the only difference is that rather than removing the first M indices of I_{attr}, we relabel those samples with the class of the highest incorrect-class logit, following [16].

Algorithm 1 Bisection Search for Computing Data Support

Input: Target sample, $z_t = (x_t, y_t)$
Input: Training set, S, and a list of top k training set indices I_{attr} ordered by the attribution method $\tau(z, S)$
Input: Model f_θ
Input: Search budget, N_{budget}
Input: Number of times to test classification, N_{test}
Output: N_{support}, size of the smallest training subset $R \subset S$ such that f_θ misclassifies x_t on average

```
 1: L ← 0
 2: H ← $|I_{\text{attr}}|$
 3: M ← $H$
 4: $C_{\text{avg}} \leftarrow \text{CounterfactualTest}(f, S, I_{\text{attr}}[: M])$
 5: if $C_{\text{avg}} > 0.5$ then
 6:     return -1                                  ▷ $N_{\text{support}}$ is larger than $k$
 7: end if
 8: $N_{\text{support}} \leftarrow M$
 9: while $N_{\text{budget}} > 0$ do
10:     $N_{\text{budget}} \leftarrow N_{\text{budget}} - 1$
11:     $M \leftarrow (L + H)/2$
12:     $C_{\text{avg}} \leftarrow \text{CounterfactualTest}(f, S, I_{\text{attr}}[: M])$
13:     if $C_{\text{avg}} > 0.5$ then
14:         $L \leftarrow M$
15:     else
16:         $H \leftarrow M$
17:         $N_{\text{support}} \leftarrow \min(M, N_{\text{support}})$
18:     end if
19: end while
20: return $N_{\text{support}}$
```

For bisection search across all attribution methods, we use a search budget of 7. For the CIFAR-10 data brittleness metrics, we aggregate predictions over 5 independently trained models. Thus, to evaluate a single validation sample, we train 35 models (7 budget $\times$ 5 models) for a total of 3500 (35 $\times$ 100 samples) models for a data brittleness metric. On ImageNet, we don't aggregate predictions and only train a single model. Hence, to evaluate a single validation sample on Imagenet, we train 7 models per sample, and a total of 210 models for evaluating a data brittleness metric. Due to the large training cost on ImageNet, we only show results for data removal support. We explicitly point out that these costs are incurred only for analysis of these data attribution methods (see Section 2). Our attribution approach is in comparison, extremely cheap to compute.

A.5 Role of Visual Similarity

In Fig. 9, we plot the most similar training images according to Datamodels, TRAK, and our method. Given that our approach relies on comparing MoCo features from the same class as the target image, it makes sense that the closest training images are visually similar. On the other hand, the most similar training images found by Datamodels [16] and TRAK [24] show more variability. Despite the variability of most similar train images, Datamodels (300K) outperforms all other methods in the counterfactual tasks assessed in Fig. 3, hinting at the importance of additional contributing factors. Still, our method underscores the significant impact of relying solely on visual similarity, essentially showing that a significant fraction of data attribution can be achieved without knowledge of the learning algorithm, based only on knowledge of the training set.

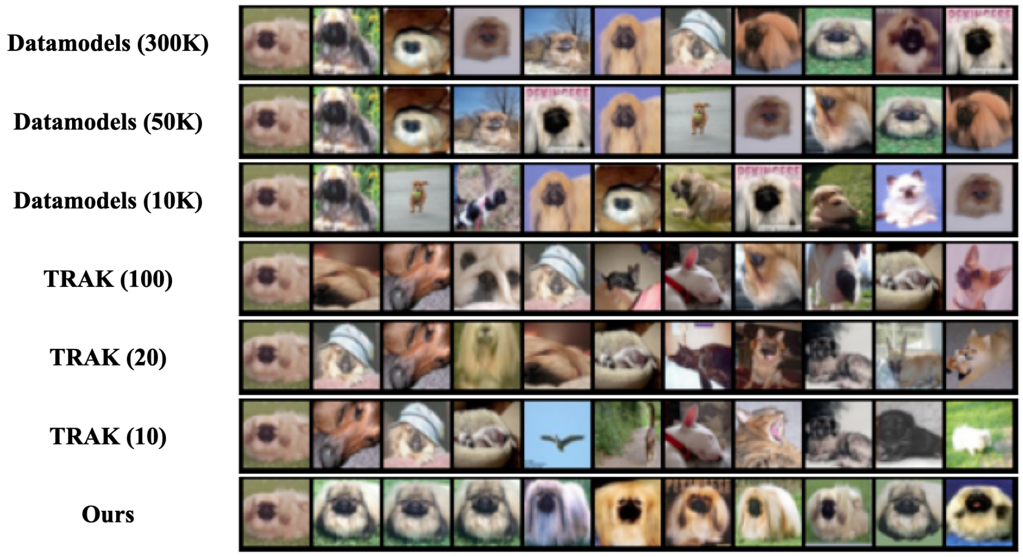

Figure 9: **Our attribution method consistently selects the most visually similar training images by design**. In each row, we plot the same target test image (Index 31), followed by ten most similar training images according to each attribution method.

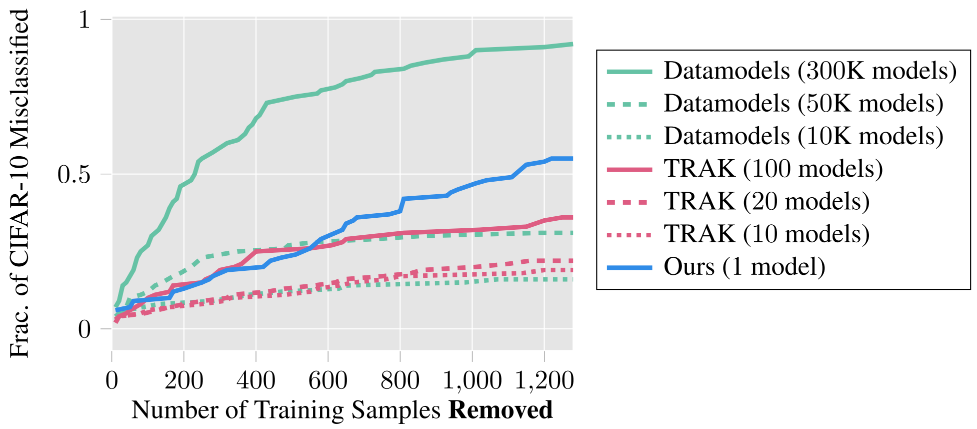

Figure 10: **Our baseline approach is model agnostic and performs well across different architectures.** We evaluate how attribution scores transfer from one architecture transfer to another. We use ResNet-9 scores for TRAK and DataModels and estimate data removal support for MobileNetV2. For our approach, we use the same ResNet-18 backbone.

A.6 Transfer to different architecture

Datamodels and TRAK utilize information tied to the model architecture such as gradients or logits from an ensemble of models. However, different neural network architectures are known to exploit similar biases and output similar predictions [23, 29]. In order to better understand how data may be shaping these biases we test how well attribution scores from these approaches transfer to other architectures. Since our approach does not use any information about the model architecture and only leverages the data, we expect our baseline approach to transfer across different architectures.

In Fig. 10, we compare TRAK, Datamodels, and our attribution scores and evaluate them on a MobileNetV2 architecture [26]. The results show that our approach using ResNet-18 continues to predict accurate data removal estimates surpassing TRAK (100) and Datamodels (50K), which suffer a large degradation in performance. Datamodels (300K) also suffer degradation in performance but provide tighter estimates than our approach. This suggests that while simply relying on visual similarity may be useful for efficiently predicting counterfactuals, additional biases within the architecture may also have an influence.

A.7 Other Related Work

Data attribution methods should produce accurate counterfactual predictions about model outputs. Although a counterfactual can be addressed by retraining the model, employing this straightforward approach becomes impractical when dealing with large models and extensive datasets. To address this problem, data attribution methods perform various approximations.

The seminal work on data attribution of Koh *et al.* [18] proposes attribution via approximate *influence functions*. More specifically, Koh *et al.* [18] identify training samples most responsible for a given prediction by estimating the effect of removing or slightly modifying a single training sample. But being a first-order approximation, influence function estimates can vary wildly with changes to network architecture and training regularization [1]. Nevertheless, approximating influence functions is reasonably inexpensive and has recently also been attempted for multi-billion parameter models [10].

Measuring empirical influence has also been attempted through construction of subsets of training data that include/exclude the target sample [7]. In a related approach, TracIn [25] and Gradient Aggregated Similarity (GAS) [12, 13] estimate the influence of each sample in training set S on the test example z_t by measuring the change in loss on z_t from gradient updates of mini-batches. While TracIn can predict class margins reasonably well, the method struggles at estimating data support. Other methods for influence approximation include metrics based on representation similarity [30, 4]. Another related line of work has utilized Shapley values to ascribe value to data, but since Shapley values often require exponential time to compute, approximations have been proposed [8, 17]. In general, there seems to be a recurring tradeoff: methods that are computationally efficient tend to be less reliable, whereas sampling-based approaches are more effective but require training thousands (or even tens of thousands) of models.

A.8 Linear Datamodeling Score

Let $\tau(z, S) : \mathcal{Z} \times \mathcal{Z}^n \to \mathbb{R}^n$ be a data attribution method that, for any sample $z \in \mathcal{Z}$ and a training set S assigns a score to every training sample indicating its importance to the model output. Consider a training set $S = \{z_1, z_2 \dots z_n\}$, and a model output function $f_\theta(z)$. Let $\{S_1, ..., S_m | S_i \subset S\}$ be m random subsets of the training set S, each of size $\alpha \cdot n$ for some $\alpha \in (0, 1)$. The linear datamodeling score (LDS) is defined as:

$$\text{LDS}(\tau(z, S)) = \rho(\{f_{\theta(S_j)}(z) \mid j \in [m]\}, \{\tau(z, S) \cdot_{S_j} \mid j \in [m]\}) \tag{1}$$

where ρ denotes Spearman rank correlation [19], $\theta(S_j)$ denotes model parameters after training on subset S_j, and $_{S_j}$ is the indicator vector of the subset S_j . Unlike data brittleness metrics, LDS accounts for samples with positive as well as negative influence.

To compute LDS scores, for our model output function $f_\theta(z)$, we use the correct class margin. This is defined as:

$$f_\theta(z) = (\text{logit for correct class}) - (\text{highest incorrect logit})$$

Our approach cannot directly be applied to compute LDS scores, as for a validation sample z_t we only focus on training samples with the most positive impact. We propose a simple modification to our approach. We assign a score to each training data based on the inverse of signed l_2 distance. The sign is based on whether the label for the training sample matches z_t. We then threshold our scores, such that all scores beyond the top-5% are zero leading to sparser attribution scores. The sparsity prior has been shown to be effective for data attribution [16, 24].

In Table 1, we present a comparison of LDS scores using our baseline approach, TRAK and Datamodels. Although our baseline was not initially designed for direct LDS score approximation, a simple adaptation demonstrates comparable performance to TRAK (5) on CIFAR-10. TRAK with a larger ensemble of models can achieve higher LDS scores. The Datamodels framework was optimized for this objective and trained as a supervised learning task, using tens of thousands of models. Hence, it achieves a better correlation with LDS.

	Models Used	LDS Scores
Datamodels	300,000	0.56
	50,000	0.43
	10,000	0.24
TRAK	100	0.22
	20	0.15
	10	0.12
	5	0.08
Ours	1	0.08

Table 1: We compare LDS scores for our approach with other baselines on CIFAR-10. Our proposed approach can perform equivalent to TRAK with 5 models.

It is important to highlight that while Datamodels and TRAK outperform our baseline in terms of LDS with extensive model ensembles, this metric provides limited insights into understanding machine learning models. Our baseline approach excels in data brittleness metrics, offering a faithful representation of which training samples provide the most positive influence for a test sample.

A.9 Baselines and Our Experimental Setup

For TRAK scores on CIFAR-10, we train 100 ResNet-9 models and use a projection dimension of 20480. For TRAK on ImageNet, we train 4 ResNet-18 models and use a projection dimension of 4096. Computing TRAK scores using 4 models already requires 160 GB of storage space, hence we refrain from using a larger ensemble. For Datamodels, we download the pre-trained weights optimized using outputs from 300K ResNet-9 models with 50% random subsets.[4] We also download the binary masks and margins to train our own Datamodels on outputs from 10K and 50K ResNet-9 models, using another 10K models for validation. Since Datamodels are extremely compute-intensive we cannot include them as a baseline on ImageNet.

For our baseline approach to train self-supervised models, we use the Lightly library [5]. We train a ResNet-18 model using MoCo [14] for 800 epochs on CIFAR-10, using the Lightly benchmark code.[6] On ImageNet, we download a pre-trained ResNet-50 model trained using MoCo.[7]

[4]https://github.com/MadryLab/datamodels-data

[5]https://github.com/lightly-ai/lightly

[6]https://docs.lightly.ai/self-supervised-learning/getting_started/benchmarks.html

[7]https://github.com/facebookresearch/moco