Distilled Datamodel with Reverse Gradient Matching

Jingwen Ye **Ruonan Yu** **Songhua Liu** **Xinchao Wang** †
National University of Singapore
jingweny@nus.edu.sg, {ruonan,songhua.liu}@u.nus.edu, xinchao@nus.edu.sg

Abstract

The proliferation of large-scale AI models trained on extensive datasets has revolutionized machine learning. With these models taking on increasingly central roles in various applications, the need to understand their behavior and enhance interpretability has become paramount. To investigate the impact of changes in training data on a pretrained model, a common approach is leave-one-out retraining. This entails systematically altering the training dataset by removing specific samples to observe resulting changes within the model. However, retraining the model for each altered dataset presents a significant computational challenge, given the need to perform this operation for every dataset variation. In this paper, we introduce an efficient framework for assessing data impact, comprising offline training and online evaluation stages. During the offline training phase, we approximate the influence of training data on the target model through a distilled synset, formulated as a reversed gradient matching problem. For online evaluation, we expedite the leave-one-out process using the synset, which is then utilized to compute the attribution matrix based on the evaluation objective. Experimental evaluations, including training data attribution and assessments of data quality, demonstrate that our proposed method achieves comparable model behavior evaluation while significantly speeding up the process compared to the direct retraining method.

1. Introduction

In the contemporary landscape of machine learning and artificial intelligence, our substantial reliance on large-scale training data has become increasingly pronounced. The notable successes of large AI models like GPT-3 [7], BERT [10], and DALL-E [33] can predominantly be attributed to the availability of extensive datasets, enabling them to discern complex patterns and relationships. As AI models progressively embrace a data-driven paradigm, comprehending the notion of "training data attribution" within a machine learning framework emerges as pivotal. It is imperative to acknowledge that model errors, biases, and the overall capabilities of these systems are frequently intertwined with the characteristics of the training data, making the enhancement of training data quality a reliable avenue for bolstering model performance.

Despite the various techniques available for interpreting models' decision-making processes, the very most of them concentrate on assessing the significance of features [29, 35, 40] and explaining the internal representations of models [3, 14, 20, 48]. When examining the attribution of training data, a persistent dilemma surfaces, one that revolves around the delicate balance between computational demands and effectiveness. On one hand, techniques like influence approximation [16, 21] prioritize computational efficiency, but they may exhibit unreliability, especially in non-convex environments. Concurrently, another line of research has achieved remarkable progress in approximating the impact of even minor alterations, such as the removal of a single data point or a small subset from the complete training set, on the trained model [32, 47]. These methods, however, are tailored specifically for scenarios involving minor changes in the training data, lacking the necessary flexibility for broader applications.

In this study, we prioritize flexibility and robustness by opting to retrain the model using a dataset that excludes specific data points. Subsequently, we compare the newly trained models with the original model. The attribution matrix is then computed based on the specific objectives of model evaluation. Specifically, to effectively and explicitly study the newly trained models, we introduce in this paper a novel *Distilled Datamodel* framework (DDM). The DDM framework is centered around the estimation of parameters for the newly trained models, rather than solely focusing on the evaluation of prediction performance at a specific test point. This approach grants the flexibility to analyze various aspects of model behavior and performance. As is shown in Fig. 1, DDM encompasses two distinct processes: offline training and online evaluation. During offline training, we distilled the influence of the training data

† Corresponding author.

back to the input space to get a rather small synset, a process achieved through reversed gradient matching. We contend that this novel reversed gradient matching approach, when compared to the standard gradient matching [55], is more effective in afterward mitigating the influence of specific training data on the target network. During online evaluation, we perturbed the synset by deleting, which, along with the target network, is leveraged to quickly train the new model. With all the newly trained networks, the attribution matrix can be easily obtained for different evaluation objectives. In a word, our contributions are:

- We explore a training data attribution framework that explicitly identifies a training sample's responsible for various behaviors exhibited by the target model. By quantifying the impact and contribution of individual samples, our framework provides insights into the relationship between the training data and the model.
- We introduce a novel influence-based dataset distillation scheme that matches the reversed gradient update, which results in a highly efficient unlearning of certain data points from the target network.
- Experimental results demonstrate that the proposed analysis method provides an accurate interpretation and achieves a significant speedup compared to its unlearning counterpart.

2. Related Work

2.1. Data-based Model Analysis

Model behavior analysis has emerged as a foundational aspect of machine learning and artificial intelligence research and development, often categorized into training data based and testing data based methods

Testing data based methods focus on elucidating the model's inference capabilities for for a certain input. Plenty researches [1, 2, 9, 13, 34, 41, 42, 44–46, 52, 56] contribute to this field of research.

In this study, our primary focus is on analyzing the model's behavior based on its training data, with one key approach being the utilization of influence approximation techniques as demonstrated by prior research [4, 16, 21, 37]. As pointed out by the authors, these approaches primarily focus on local changes that are *infinitesimally-small*, which are also extremely time consuming. Datamodels [19] is proposed for analyzing the behavior of a model class in terms of the training data, which measures the correlation between true model outputs and attribution-derived predictions for those outputs. Following this work, ModelPred [53] is proposed for predicting the trained model parameters directly instead of the trained model behaviors. Nevertheless, both these methods still entail the training of a considerable number of models, often in the thousands or tens of thousands, for effectiveness. In this work, we investigate a more efficient framework to facilitate this process.

2.2. Machine Unlearning

The concept of unlearning is firstly introduced by Bourtoule *et al.* [5], which aims to eliminate the effect of data point(s) on the already trained model. Along this line, machine unlearning has attracted more attentions, of which the existing approaches can be roughly divided into *exact* [5, 6, 8, 15] methods and *approximate* methods [6, 18, 30, 39, 49, 50].

Exact methods decrease the time it takes to exactly/directly retrain the models. Bourtoule *et al.* [5] propose an unlearning framework that when data needs to be unlearned, only one of the constituent models whose shards contains the point to be unlearned needs to be retrained. Cao *et al.* [8] transform learning algorithms used by a system into a summation form and to forget a training data sample, they simply update a small number of summations. DaRE trees [6] are proposed to enable the removal of training data with minimal retraining, which cache statistics at each node and training data at each leaf to update only the necessary subtrees as data is removed. Unlike the exact methods, the approximate ones try to find a way to approximate the retraining procedure. To minimize the retraining time, data removal-enabled forests [6] are introduced as a variant of random forests, which delete data orders of magnitude faster than retraining from scratch while sacrificing little to no predictive power. Nguyen *et al.* [30] study the problem of approximately unlearning a Bayesian model from a small subset of the training data to be erased.

The above unlearning methods focus more on balancing the accuracies and the efficiency. Here we focus more on the efficiency, which model the network behavior for analyzing the attributions of a target model.

2.3. Dataset Distillation

Dataset condensation/distillation [25–28, 51, 54] aims to condense a large training set into a small synthetic set to obtain the highest generalization performance with a model trained on such small set of synthetic images. Zhao et al. [54] formulate the goal as a gradient matching problem between the gradients of deep neural network weights that are trained on the original and the synthetic data. Zhou et al. [57] address these challenges of significant computation and memory costs by neural feature regression with pooling. Nguyen et al. [31] apply a distributed kernel-based meta-learning framework to achieve state-of-the-art results for dataset distillation using infinitely wide convolutional neural networks. Sucholutsky et al. [43] propose to simultaneously distill both images and their labels, thus assigning each synthetic sample a 'soft' label.

Different from the previous data condensation methods, we tend to use the fast convergence and the gradient matching properties for the analysis of the target network. Thus,

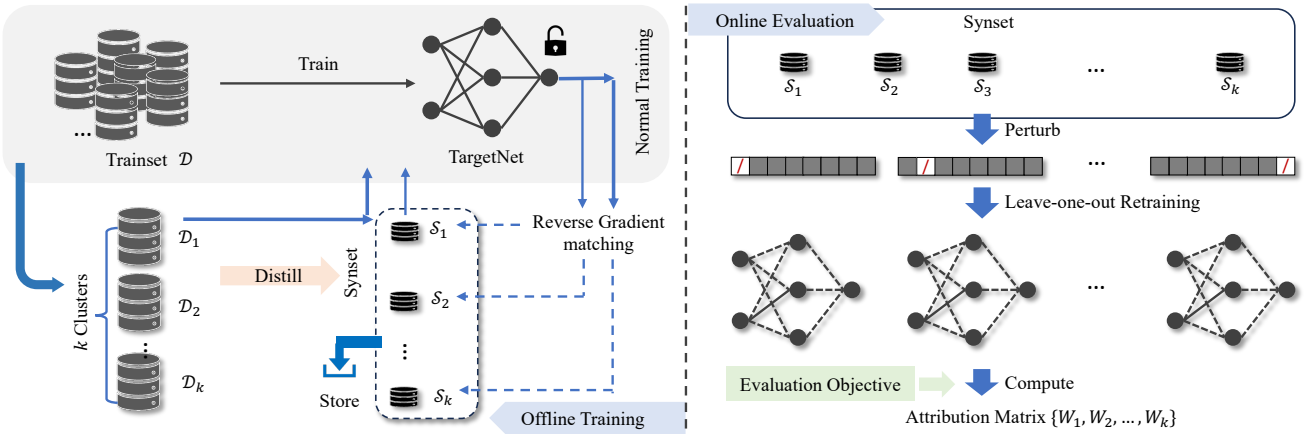

Figure 1. The framework of the proposed distilled datamodel. During the offline training, the synset is distilled during the normal training of target network. As for online evaluation we perturb the learned synset and fast learn the perturbed model set, which is computed to form the final attribution matrix.

we focus on how to model the data's impact on the network not just for improving the accuracies.

3. Proposed Method

In this paper, we propose the distilled datamodel framework to build the training data attribution to evaluate various model behaviors.

3.1. Problem Statement

Given a target model $\mathcal{M}$ trained on dataset $\mathcal{D}$, we tend to construct direct relationship between them, which is denoted as the attribution matrix W. Each weight in W measures the responsible of the corresponding training points on certain behaviors of $\mathcal{M}$.

The attribution matrix W learned by the proposed DDM framework works on various behaviors, which include but not limited to:

- **Model functionality analysis.** This involves evaluating the performance of the target network using the training data. This could include measuring key metrics such as accuracy, precision, recall, and F1 score, and comparing the results to established benchmarks or industry standards.
- **Model diagnose.** This involves examining the errors made by the target network when processing the training data. This could include identifying the types of errors made, such as misclassifications or false positives, and determining the root cause of the errors, such as data quality issues or model limitations.
- **Influence function of certain test samples.** This traces a model's prediction through the learning algorithm and back to its training data, thereby identifying training points most responsible for a given prediction.

In what follows, we take studying the model behavior on the influence of certain test samples as an example, showing how to learn the corresponding training data attribution with the proposed DDM framework. We would also include more details on studying other kinds of model behaviors in the supplementary.

Note that in Fig. 1, the proposed DDM framework introduces a two-step process:

- **Offline Training** (Sec. 3.2): This step is learned only once and can be integrated into network training. Its objective is to distill and store data influence with improved approximation and reduced storage requirements.
- **Online Evaluation** (Sec. 3.3): This phase involves evaluation to meet specific requirements for model behavior analysis, which is realized by perturbing the dataset. The primary goal is to compute the training data attribution matrix while minimizing time and computational costs.

3.2. Offline Training

During the offline training, we tend to obtain the synset $\mathcal{S}$ ($|\mathcal{S}| \ll |\mathcal{D}|$) to distill the training data influence from the target network $\mathcal{M}$, so as to produce the parameters of the network with perturbed dataset. To begin with, we cluster the original training data $\mathcal{D}$ into K groups as $\{\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_K\}$, with the consideration that the existing of single data point won't be able to make much difference on the behaviors of the target network $\mathcal{M}$. So it's more meaningful to build the cluster-level training data attribution under this circumstance.

Here the target network is initialized with parameters θ_0 and subsequently trained on the dataset $\mathcal{D}$ for τ epochs, and the synset is for finetuning the trained target network for T epochs, resulting in updated parameters θ_τ and $\tilde{\theta}_T$. The

objective is formulated as:

$$\begin{aligned}
&\mathcal{A}(\mathcal{D}): \theta_\tau = \arg\min_\theta \mathcal{L}(\theta, \mathcal{D}) = \arg\min_\theta \sum_k \mathcal{L}(\theta, \mathcal{D}_k), \\
&\mathcal{U}(\mathcal{D}_\kappa) = \mathcal{A}(\bigcup_{k\neq\kappa} \mathcal{D}_k) : \theta_\tau^\kappa = \arg\min_\theta \sum_{k\neq\kappa} \mathcal{L}(\theta, \mathcal{D}_k), \\
&\mathcal{F}(\mathcal{S}_\kappa) : \tilde{\theta}_T^\kappa = \arg\min_\theta \mathcal{L}(\theta, \mathcal{S}_\kappa), \\
&s.t. \quad \mathcal{S}_\kappa = \arg\min_{\mathcal{S}_\kappa, |\mathcal{S}_\kappa| \ll |\mathcal{D}_\kappa|} \left[\mathcal{D}ist(\theta_\tau^\kappa, \tilde{\theta}_\tau^\kappa)\right],
\end{aligned} \tag{1}$$

where $\kappa = \{1, 2, ..., K\}$ and $\mathcal{L}(\cdot, \cdot)$ is the loss for training the target network $\mathcal{M}$. $\mathcal{A}$ stands for the learning process with τ epochs, $\mathcal{U}$ stands for the unlearning process (equivalent to training without the unlearn set with τ epochs), $\mathcal{F}$ stands for the fine-tuning process starting with $\tilde{\theta}_0 \leftarrow \theta_\tau$ with T epochs. The synset $\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_K\}$ is extremely small in scale comparing with the original dataset $\mathcal{D}$. To achieve this goal, our approach involves the minimization of the distribution distance, denoted as $\mathcal{D}ist(\cdot, \cdot)$, between the parameters of the synset fine-tuned model $\tilde{\theta}_\tau^\kappa$, and the directly unlearned parameters θ_τ^κ.

Assuming that the target network parameters are updated through stochastic gradient descent for $t = 1, 2, ..., \tau$ epochs with a learning rate η_a, and the finetuning process of the target network spans $t = 1, 2, ..., T$ epochs with a learning rate η_f, we can reformulate the problem based on Eq. 1 as follows:

$$\begin{aligned}
&\theta_{t+1} \leftarrow \theta_t - \eta_a \nabla \mathcal{L}(\theta_t, \mathcal{D}), \\
&\theta_{t+1}^\kappa \leftarrow \theta_t^\kappa - \eta_a \sum_{k\neq\kappa} \nabla \mathcal{L}(\theta_t^\kappa, \mathcal{D}_k) \quad w.r.t. \quad \theta_0^\kappa = \theta_0, \\
&\tilde{\theta}_{t+1}^\kappa \leftarrow \tilde{\theta}_t^\kappa - \eta_f \nabla \mathcal{L}(\tilde{\theta}_t^\kappa, \mathcal{S}_k) \quad w.r.t. \quad \tilde{\theta}_0^\kappa = \theta_\tau,
\end{aligned} \tag{2}$$

where $\nabla\mathcal{L}$ is the gradient computed on θ. Based on it, we simplify the problem by setting $\eta = \eta_a = \eta_f$ and $\tau = T$. In this way, we accumulate the gradients in the learning and finetuning process as:

$$\begin{aligned}
\theta_\tau &= \theta_0 - \eta \sum_t \nabla \mathcal{L}(\theta_t, \mathcal{D}), \\
\theta_\tau^\kappa &= \theta_0 - \eta \sum_t \sum_{k\neq\kappa} \nabla \mathcal{L}(\theta_t^\kappa, \mathcal{D}_k), \\
&= \theta_0 - \eta \sum_t \left[\nabla \mathcal{L}(\theta_t, \mathcal{D}) - \nabla \mathcal{L}(\theta_t^\kappa, \mathcal{D}_\kappa)\right], \\
\tilde{\theta}_\tau^\kappa &= \theta_\tau - \eta \sum_t \nabla \mathcal{L}(\tilde{\theta}_t^\kappa, \mathcal{S}_\kappa).
\end{aligned} \tag{3}$$

Note that our goal is to make $\tilde{\theta}_\tau^\kappa \approx \theta_\tau^\kappa$, then Eq. 3 can be further simplified as:

$$-\sum_t \nabla \mathcal{L}(\tilde{\theta}_t^\kappa, \mathcal{S}_\kappa) = \sum_t \nabla \mathcal{L}(\theta_t^\kappa, \mathcal{D}_\kappa), \tag{4}$$

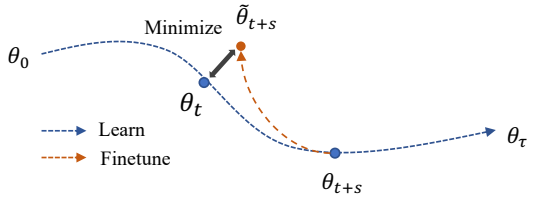

Figure 2. The proposed reverse gradient matching process. The synset is optimized by the reverse gradients.

given that $\tilde{\theta}_0^\kappa = \theta_\tau$ and $\theta_0^\kappa = \theta_0$, the sufficient solution to Eq. 4 is:

$$\begin{aligned}
&\nabla \mathcal{L}(\tilde{\theta}_{\tau-t}^\kappa, \mathcal{S}_\kappa) = -\nabla \mathcal{L}(\theta_t^\kappa, \mathcal{D}_\kappa), \\
\Rightarrow \quad &\sum_\kappa \nabla \mathcal{L}(\tilde{\theta}_{\tau-t}^\kappa, \mathcal{S}_\kappa) = -\sum_\kappa \nabla \mathcal{L}(\theta_t^\kappa, \mathcal{D}_\kappa), \\
\Rightarrow \quad &\sum_\kappa \nabla \mathcal{L}(\theta_{\tau-t}, \mathcal{S}_\kappa) = -\sum_\kappa \nabla \mathcal{L}(\theta_t, \mathcal{D}_\kappa),
\end{aligned} \tag{5}$$

where the synset in our proposed DDM is learnt for matching the reverse training trajectory while training the target network initialized from θ_0 to θ_τ. This reverse gradient matching process is depicted in Fig. 2. Thus, synset $\mathcal{S}$ here is for predicting the parameters of the target network $\mathcal{M}$ that unlearns the κ-th data cluster $\mathcal{D}_\kappa$, which is achieved by directly finetuning the target network with the synset $\mathcal{S}$.

We constrain the scale of the synset to ensure efficient storage and fine-tuning process. Motivated by the idea of dataset condensation [55] with gradient matching, we propose the reverse gradient matching to distill and store the gradient information to a couple of synthetic images $\mathcal{S}$ ($|\mathcal{S}| \ll |\mathcal{D}|$). The synset $\mathcal{S}$ is optimized by:

$$\arg\min_{|\mathcal{S}|=K\times ipc} \sum_t \sum_\kappa \mathcal{D}ist\left(\nabla \mathcal{L}(\theta_{\tau-t}, \mathcal{S}_\kappa), -\nabla \mathcal{L}(\theta_t, \mathcal{D}_\kappa)\right), \tag{6}$$

where for each data cluster $\mathcal{D}_\kappa$, we learn a corresponding $\mathcal{S}_\kappa$ which contains ipc images. In experiments, we set $ipc = 1$ and using the cosine distance for $\mathcal{D}ist(\cdot)$.

Why do we choose **reverse gradient matching over gradient matching**? There are two main reasons:

- **Enhanced matching performance.** Since the number of unlearn set is smaller than the whole set and the optimization of data matches the initial stage of the learning trajectory, making the accumulated trajectory error much smaller using our proposed reverse gradient matching. Detailed evidence supporting this claim is provided in the supplementary materials.
- **Improved Privacy Protection:** While traditional data distillation using gradient matching offers a degree of privacy protection for the dataset [11], the distilled images still retain distinguishable patterns of the main object, posing privacy risks. In contrast, images synthesized

using reverse gradient matching exhibit no explicit patterns, thus ensuring a higher level of privacy protection. Detailed comparisons in this regard are presented in the experimental results.

3.3. Online Evaluation

During the online evaluation stage, both the synset $\mathcal{S}$ and the target network $\mathcal{M}$ are available, allowing for the evaluation of specific model behaviors.

The primary concept behind online evaluation is to employ leave-one-out cross-validation, which entails systematically perturbing the training dataset $\mathcal{D}$ by removing specific training samples. This process helps analyze the resulting impact on the model's performance. Here we take studying the influence function for example, which is a typical task for analyzing the model's data sensitivity, offering insights into its robustness and decision boundaries. To be concrete, given a test sample $\{x_t, y_t\}$, The corresponding prediction result as $\tilde{y}_t$, where the target model is trained on the original whole dataset $\mathcal{D}$. The objective here is to directly build the relationship with the network prediction $\tilde{y}_t$ and the training data $\mathcal{D}$ (dataset→target network→prediction) by the attribution matrix W:

$$\tilde{y}_{t(p_t \circ \mathcal{D})} = W \cdot p_t + b, \quad p_t \subseteq \{0,1\}^K, \tag{7}$$

where p_t stands for the perturbation operation over the dataset, and $p_t(\kappa) = 0$ denotes the deletion of that data cluster $\mathcal{D}_\kappa$ from the training set. And $\tilde{y}_{t(P_t \circ \mathcal{D})}$ denotes the prediction by the target network, which is trained from scratch using the dataset $p_t \circ \mathcal{D}$.

Then, the attribution matrix W calculated from perturbing the training data can be calculated as:

$$\arg\min_{W} \sum_{p_t \subseteq P_t} \beta_{p_t} \cdot \mathcal{D}ist\big(\tilde{y}_{t(p_t \circ \mathcal{D})}, W \cdot p_t\big), \tag{8}$$

where p_t is randomly sampled from the $\{0,1\}^K$, β_{p_t} represents the weights corresponding to the number of 0s in p_t. The distance function $\mathcal{D}ist(\cdot)$ is set as the L2 norm distance for measuring influence function of the model. And the attribution matrix $W \subseteq \mathbb{R}^{K \times |y_t|}$, signifies the contribution of each training data cluster to the confidence scores of each label in the target network's prediction for the test sample x_t. Let P_t denote the perturbation set. To calculate the attribution matrix W, a minimum of K perturbations is required, such that $|P_t| \geq K$, applied to the training data.

The main difficulty in Eq. 8 lies in obtaining $|P_t|$ new trained models training with the perturbed dataset $p_t \circ \mathcal{D}$, so as to get the corresponding inference $\tilde{y}_{t(p_t \circ \mathcal{D})}$. Recall that during the offline training process, we already got the distilled synthetic data $\mathcal{S}_\kappa$ for each data cluster $\mathcal{D}_\kappa$, which could fast unlearn $\mathcal{D}_\kappa$ from the target network. And the model parameters with the perturbed dataset could be fine-tuned with the synset as:

$$\theta_{p_t} \leftarrow \underset{\kappa \in \{1,2,\ldots,K\}, p_t(\kappa)=0}{\mathcal{F}} (\mathcal{S}_\kappa). \tag{9}$$

As a result, in the offline evaluation stage, we solve this difficulty by eliminating each cluster of training data from the target network, which is further accelerated by our proposed reverse gradient matching.

Accelerate with hierarchical distilled datamodel. To expedite the online evaluation process, we implement a hierarchical data distillation approach. This strategy encompasses the distillation of both the class-wise datamodel (with $K = |y|$) and the cluster-wise datamodel (with $K = |y| \times c$, where each label's data is partitioned into c clusters).

By following this approach, we can construct both the class-wise and the cluster-wise attribution matrices. This approach accelerates the analysis of model behavior, including tasks such as identifying the most influential training data. This is achieved by initially pinpointing the class-wise data points and subsequently calculating the training matrix within each class.

Algorithm 1 The Proposed DDM Framework

Offline Training

Input: $\mathcal{D}$: training set; $\mathcal{M}$: target model; $\{\theta_0, \theta_1, ..., \theta_\tau\}$: training trajectory of the target network ; s: trajectory step.

Divide the training data $\mathcal{D}$ into K clusters;
Randomly initialize K synthetic samples, formed as $\mathcal{S}$;
for each distillation step **do**
  Choose random start from target trajectory: θ_t $(0 \leq t < \tau)$;
  Choose the end from target trajectory: θ_{t+s} $(t + s < \tau)$;
  for $\kappa = 1, 2, ..., K$ **do**
    Calculate the gradients on real data: $\nabla\mathcal{L}(\theta_t; \mathcal{D}_\kappa)$;
    Calculate the gradients of the synset $\nabla\mathcal{L}(\theta_{t+s}, \mathcal{S}_\kappa)$;
    $\min \mathcal{D}ist(\nabla\mathcal{L}(\theta_t; \mathcal{D}_\kappa), -\nabla\mathcal{L}(\theta_{t+s}, \mathcal{S}_\kappa))$ to update $\mathcal{S}_\kappa$;
  end for
end for

Output: Cluster-wise synthetic data $\{\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_K\}$.

Online Evaluation

Input: $\mathcal{M}$: target model; $\mathcal{S}$: synset; x_t: test sample.

Randomly sample perturbations p_t to form P_t;
for each p_t **in** P_t **do**
  Perform perturbation p_t on the synset $\mathcal{S}$ as $p_t \circ \mathcal{S}$;
  Fine-tune $\mathcal{M}$ with $p_t \circ \mathcal{S}$;
  Input x_t to the fine-tuned network and get $\tilde{y}_t$;
end for
Calculate the attribution matrix W with Eq. 8.

Output: Attribution matrix W.

Table 1. Ablation study on the influence analysis of certain test samples on MNIST, CIFAR10 and CIFAR100 datasets. We locate to the source data considering three distance functions. We report the value$\times$100 for $\mathcal{D}ist_1$ in the table, larger is better and tiny number in red donates the improvement or drop compared with 'Random Select'.

Method	MNIST			CIFAR10			CIFAR100		
	$\mathcal{D}ist_1$	$\mathcal{D}ist_2$	$\mathcal{D}ist_3$	$\mathcal{D}ist_1$	$\mathcal{D}ist_2$	$\mathcal{D}ist_3$	$\mathcal{D}ist_1$	$\mathcal{D}ist_2$	$\mathcal{D}ist_3$
Random Select	3.3	0.43	0.07	2.7	0.34	0.54	2.3	0.54	0.8
Predict-based	5.5 +2.2	-	-	3.1 +0.4	-	-	3.9 +1.6	-	-
Clustering-based	6.2 +2.9	-	-	2.5 -0.2	-	-	3.6 +1.3	-	-
DDM w/o cluster	9.1 +5.8	0.55 +0.12	0.11 +0.04	5.9 +3.2	0.57 +0.23	0.78 +0.24	3.5 +1.2	0.77 +0.23	1.2 +0.4
DDM-match	**10.8** **+7.5**	**0.73** **+0.30**	**0.13** **+0.06**	5.8 +3.1	0.76 +0.42	0.81 +0.27	4.8 +2.5	0.71 +0.17	**1.6** **+0.8**
DDM-full (ours)	**10.8** **+7.5**	**0.73** **+0.30**	**0.13** **+0.06**	**6.8** **+4.1**	**0.81** **+0.41**	**0.92** **+0.38**	**5.3** **+3.0**	**0.88** **+0.34**	**1.6** **+0.8**

3.4. Algorithm and Discussions.

We depict the proposed algorithm including offline training and online evaluation in Alg. 1. During the offline training stage, we follow and modify the basic optimization framework of dataset distillation. And we give the algorithm for evaluating the influence function with x_t as input. In the online evaluation phase, adjustments are made to accommodate different evaluation objectives. Importantly, the offline training process occurs only once and remains fixed for subsequent evaluations.

4. Experiments

4.1. Experimental Settings

Datasets and networks. We conduct our experiments on several standard image classification datasets: digit recognition on MNIST dataset [24], CIFAR-10 dataset, CIFAR-100 dataset [22] and TinyImageNet [36]. Regarding the architectures of the target network, we evaluated various architectures, including AlexNetIN [23], ResNet18, ViT, and ConvNet.

Training details and parameter settings. We implemented our experiments using the PyTorch framework. In the default setting, unless otherwise specified, we set the number of clusters per class to $num_cluster = 10$. This implies that there are a total of $K = 100$ clusters for the MNIST and CIFAR-10 datasets, and $K = 1000$ clusters for the CIFAR-100 dataset. For both class-wise and cluster-wise condensation, we used a single synthetic image per cluster. These synthetic images were initialized by randomly sampling real images, and standard data augmentation techniques were applied during training. The learning rate for updating synthetic images was set to 10, while the learning rate for updating network parameters was set to 0.01. To perturb the training set $\mathcal{D}$, we set $|P_t| = K$.

Evaluation metrics. To assess the attribution of training data to the behavior of the target network when influencing specific test samples, we investigate three influence objectives, each defined by a distinct distance metric. We randomly select 20 test samples ($|X_t| = 20$) from the validation set of the training data and report the average distance metrics. And in order to compare the accuracy of such built relationship, we use the exact-unlearn network for evaluation. That is, after locating the data cluster $\mathcal{D}_i$ with the target influence objective, we scratch train the unlearn network $\mathcal{M}^u$ on $\mathcal{D}/\mathcal{D}_i$, and get predictions as y_t^u. We compute distance function Avg_dist regarding different types of influence analysis:

$$\begin{aligned} & Avg\_dist = \mathbb{E}_{x_t \sim X_t} \mathcal{D}ist_i \\ where \quad & \mathcal{D}ist_1 = \|y_t^u - \tilde{y}_t\|^2, \mathcal{D}ist_2 = \ell_{ce}(y_t^u, y_t), \\ & \mathcal{D}ist_3 = 1/(1 + \|y_t^u - \tilde{y}_t\|^2), \end{aligned} \tag{10}$$

where y_t is the groundtruth label for x_t and $\tilde{y}_t$ is the output from the target network $\mathcal{M}$. For all three distance metrics, namely $\mathcal{D}ist_1$, $\mathcal{D}ist_2$, and $\mathcal{D}ist_3$, larger values are indicative of more significant influence. Specifically, $\mathcal{D}ist_1$ is designed to trace back to the training data points that have the most influence on the current prediction, $\mathcal{D}ist_2$ focuses on identifying those with the most influence on whether the model makes correct predictions using the cross entropy loss ℓ_{ce}, and $\mathcal{D}ist_3$ is utilized to pinpoint the training data points with the least influence on the current prediction.

For evaluation objectives other than the influence function of specific test samples, relevant metrics are provided within the corresponding experimental analysis part.

4.2. Experimental Results

DDM could be used for training data influence analysis. By telling us the training points "responsible" for a given prediction, influence functions reveal insights about how models rely on and extrapolate from the training data. For three different distance functions ($\mathcal{D}ist_1$, $\mathcal{D}ist_2$ and $\mathcal{D}ist_3$), we calculate different weight matrix from Eq. 8 by replacing the corresponding distance function, obtaining W^1, W^2, W^3. And then we locate the corresponding $\mathcal{D}_i$, where $i = \arg\max_i W_i$. The ablation study re-

Table 2. Comparative experimental results with other works on MNIST, CIFAR10 and CIFAR100 datasets, regarding $\mathcal{D}ist_1$ influence.

Method	MNIST	CIFAR-10	CIFAR-100
Random Select	3.3	2.7	2.3
Koh et al. [21]	10.0	5.6	2.6
FASTIF [16]	9.8	6.5	2.4
Scaleup [38]	10.4	6.5	3.9
DDM	**10.8**	**6.8**	**5.3**

Table 3. Detecting useless training data for the target network.

Percentage	0%	1%	10%	20%	50%
Random Select	95.7	95.8	92.8	74.6	65.9
Koh et al.	95.7	95.9	93.7	81.5	74.3
FASTIF	95.7	95.5	94.1	79.6	74.0
Scaleup	95.7	96.0	95.2	82.9	73.3
DDM	95.7	96.2	95.9	85.4	79.3

garding the three types of influence functions is depicted in Table 1, where 'Random Select' denotes that we randomly choose D_i from K clusters; 'Predict-based' denotes we choose the datapoint $\mathcal{D}_i$ with the highest prediction similarity. 'Cluster-Based' denotes locating $\mathcal{D}_i$ by using the clustering strategy as we pre-process the dataset $\mathcal{D}$, which denotes the highest visual similarity; 'DDM w/o cluster' clusters $\mathcal{D}$ in K clusters by sequence number not by k-means, 'DDM-match' denotes the DDM framework that uses the gradient matching loss during the offline training stage. From the table, we observe that:

- Methods categorized as 'Predicted-based' and 'Clustering-based' can serve as alternatives for evaluating $\mathcal{D}ist_1$ type inferences. While they turn to be less accurate than our DDM. It further proves that visual similarity between two images does not fully capture the influence of one on the other in terms of model behavior [19]. In addition, they are limited in their ability to provide a comprehensive analysis of $\mathcal{D}ist_2$ and $\mathcal{D}ist_3$. This highlights the potential of our proposed DDL model to evaluate a wider range of model behaviors.
- Comparing the results with 'DDM w/o cluster' and 'DDM-full', it becomes evident that the clustering strategy offers a more accurate method for pinpointing influential training data.
- Upon comparing the results with 'DDM-match' and 'DDM-full', it is apparent that optimizing the synset with gradient matching produces favorable outcomes on simpler datasets like MNIST. However, as the dataset complexity increases, this approach exhibits a decline in performance, falling behind the optimization with reverse gradient matching.

How do different target network architectures affect DDM? We have conducted our proposed DDM framework into several different network architectures, including AlexNetIN, ResNet18_AP [17] and simple ViT [12]. We calculate the training data attribution weights for each class of training data for measuring the model behaviors on $\mathcal{D}ist_1$. The test input is a batch of images with the groundtruth label of '2'. The experimental results are conducted on the MNIST dataset and the CIFAR-10 dataset, which are depicted in Fig. 3. From the figure, observations can be drawn that:

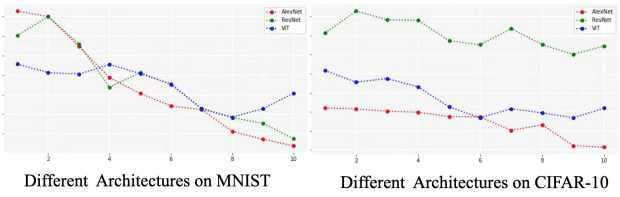

Figure 3. Comparison of the training data attribution weights calculated form different network architectures. In the figure, we show the class-wise weights.

- All the networks with different architectures trace to the similar source training data (with the highest value with label '2');
- For the simple classification task in MNIST, the training data attribution matrices look similar among all the architectures;
- For a more difficult task in CIFAR10, the training data attribution matrices look also similar in the trend, but vary in the absolute weight values among all the architectures.

Comparing DDM performance with other works. The comparative results with existing works are presented in Table 2. We compare the $\mathcal{D}ist_1$ influence with three other works. To evaluate, we identify the most influential data point and calculate the average distance metrics. It is evident that our method demonstrates greater accuracy in locating these influential data points.

DDM could be used as model diagnostic for low-quality training samples. In addition to analyzing the influence functions for specific test samples, the proposed DDM also offers a comprehensive model of the overall performance of the target network. We randomly sample 10% samples and calculate the The experimental results are depicted in Table 3, which is conducted on MNIST dataset. As depicted in the figure, the deletion of 10% of the training data actually improves the network's final performance. Therefore, our proposed DDM framework succeeds in model diagnostics by detecting and removing low-quality training samples.

DDM meets the privacy protection demand. To substantiate our previous assertion that the proposed reverse gradient matching enhances privacy protection, we conducted a comparison between the distilled samples generated using traditional gradient matching and our novel re-

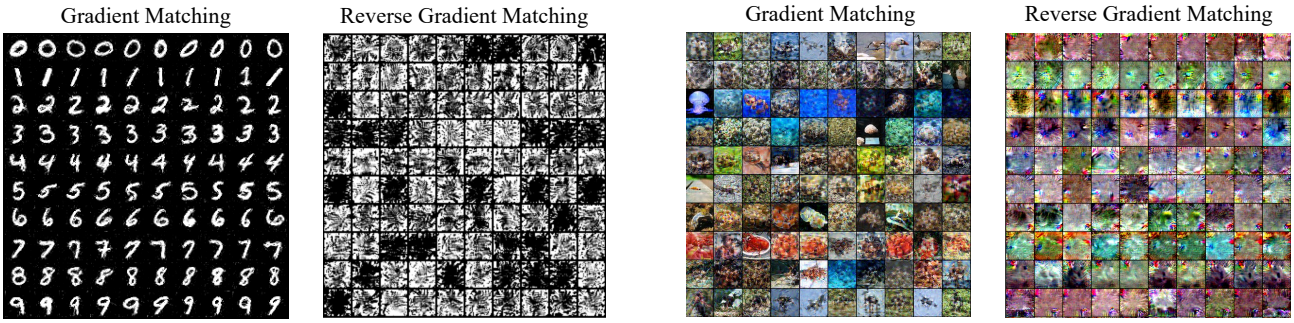

Figure 4. Visualization of condensed 10 image/class with ConvNet for MNIST (a) and CIFAR-100 (b). We compare the visualization results between gradient matching and reverse gradient matching. Each column represents a condensation of a cluster.

Table 4. The new trained network's accuracies comparison. We compare the networks fine-tuning with the proposed DDM and gradient matching synthetic images.

Dataset	Method	Acc. (θ_0)	Acc. (θ_τ)
MNIST	Normal	12.5	95.7
	DDM-Match	12.6	85.0
	DDM	0.6	95.7
CIFAR 10	Normal	12.6	85.0
	DDM-Match	12.6	40.2
	DDM	15.5	85.0
CIFAR100	Normal	2.2	56.1
	DDM-Match	2.2	23.5
	DDM	0.8	56.1
TinyImageNet	Normal	1.4	37.5
	DDM-Match	1.4	7.8
	DDM	0.1	37.5

verse gradient matching, as illustrated in Fig. 4. As evident from the visualization, in gradient matching data distillation, the synthetic images retain the characteristic features of the training set images, thus potentially revealing training data through these conspicuous patterns, particularly noticeable in the MNIST dataset. In contrast, in the visualization results of our reverse gradient matching, the distinctive features of the images are replaced by several indistinct patterns, akin to a form of obfuscation. This implies that, especially in scenarios with privacy concerns, our proposed DDM framework can be safely employed by directly releasing the synset, providing enhanced privacy protection for the original training data.

DDM could be used as a quick unlearn method. We also assert that the proposed reverse gradient matching improves matching performance, which is experimentally verified in Table 4. It's worth noting that traditional gradient matching begins by matching the initial state of the target network, resulting in the same 'Acc.(θ_0)' as normal training. However, for more complex datasets (e.g., TinyImageNet), it struggles to match the final performance of the target network, 'Acc. (θ_τ)'. In contrast, our proposed DDM commences from the final state of the target network and also effectively matches the initial performance of the target network. Thus, we contend that the proposed DDM significantly enhances matching performance.

5. Conclusion

In this paper, we introduce a novel framework known as DDM that facilitates a comprehensive analysis of training data's impact on a target machine learning model. The DDM framework comprises two key stages: the offline training stage and the online evaluation stage. During the offline training stage, we propose a novel technique, reverse gradient matching, to distill the influence of training data into a compact synset. In the online evaluation stage, we perturb the synset, enabling the rapid elimination of specific training clusters from the target network. This process culminates in the derivation of an attribution matrix tailored to the evaluation objectives. Overall, our DDM framework serves as a potent tool for unraveling the behavior of machine learning models, thus enhancing their performance and reliability.

Future research could extend the application of the DDM framework to diverse machine learning tasks and datasets. These applications could encompass fields as varied as natural language processing, reinforcement learning, computer vision, and beyond. The versatility of DDM offers opportunities to gain deeper insights into model behaviors, data quality, and training dynamics in these domains.

Acknowledgement

This project is supported by the Ministry of Education Singapore, under its Academic Research Fund Tier 2 (Award Number: MOE-T2EP20122-0006), and the National Research Foundation, Singapore, under its AI Singapore Programme (AISG Award No: AISG2-RP-2021-023).

References

[1] André Altmann, Laura Toloşi, Oliver Sander, and Thomas Lengauer. Permutation importance: a corrected feature importance measure. *Bioinformatics*, pages 1340–1347, 2010. 2

[2] Daniel W. Apley and Jingyu Zhu. Visualizing the effects of predictor variables in black box supervised learning models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 82, 2020. 2

[3] Daniel W Apley and Jingyu Zhu. Visualizing the effects of predictor variables in black box supervised learning models. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 82(4):1059–1086, 2020. 1

[4] Juhan Bae, Nathan Ng, Alston Lo, Marzyeh Ghassemi, and Roger B Grosse. If influence functions are the answer, then what is the question? *Advances in Neural Information Processing Systems*, 35:17953–17967, 2022. 2

[5] Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. *IEEE Symposium on Security and Privacy*, pages 141–159, 2021. 2

[6] Jonathan Brophy and Daniel Lowd. Machine unlearning for random forests. In *International Conference on Machine Learning*, 2021. 2

[7] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 1

[8] Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. *2015 IEEE Symposium on Security and Privacy*, pages 463–480, 2015. 2

[9] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, M. Sturm, and Noémie Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015. 2

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 1

[11] Tian Dong, Bo Zhao, and Lingjuan Lyu. Privacy for free: How does dataset condensation help privacy? In *International Conference on Machine Learning*, pages 5378–5396. PMLR, 2022. 4

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 7

[13] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2001. 2

[14] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001. 1

[15] Antonio A. Ginart, Melody Y. Guan, Gregory Valiant, and James Y. Zou. Making ai forget you: Data deletion in machine learning. In *NeurIPS*, 2019. 2

[16] Han Guo, Nazneen Fatema Rajani, Peter Hase, Mohit Bansal, and Caiming Xiong. Fastif: Scalable influence functions for efficient model interpretation and debugging. *arXiv preprint arXiv:2012.15781*, 2020. 1, 2, 7

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 7

[18] Yingzhe He, Guozhu Meng, Kai Chen, Jinwen He, and Xingbo Hu. Deepobliviate: A powerful charm for erasing data residual memory in deep neural networks. *ArXiv*, abs/2105.06209, 2021. 2

[19] Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. Datamodels: Predicting predictions from training data. In *Proceedings of the 39th International Conference on Machine Learning*, 2022. 2, 7

[20] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning*, pages 2668–2677. PMLR, 2018. 1

[21] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR, 2017. 1, 2, 7

[22] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009. 6

[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60:84 – 90, 2012. 6

[24] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86:2278–2324, 1998. 6

[25] Saehyung Lee, Sanghyuk Chun, Sangwon Jung, Sangdoo Yun, and Sungroh Yoon. Dataset condensation with contrastive signals. In *International Conference on Machine Learning*, pages 12352–12364. PMLR, 2022. 2

[26] Songhua Liu and Xinchao Wang. Mgdd: A meta generator for fast dataset distillation. In *Advances in Neural Information Processing Systems*, 2023.

[27] Songhua Liu, Kai Wang, Xingyi Yang, Jingwen Ye, and Xinchao Wang. Dataset distillation via factorization. In *Advances in Neural Information Processing Systems*, 2022.

[28] Songhua Liu, Jingwen Ye, Runpeng Yu, and Xinchao Wang. Slimmable dataset condensation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3759–3768, 2023. 2

[29] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017. 1

[30] Quoc Phong Nguyen, Bryan Kian Hsiang Low, and Patrick Jaillet. Variational bayesian unlearning. *Advances in Neural Information Processing Systems*, 33:16025–16036, 2020. 2

[31] Timothy Nguyen, Roman Novak, Lechao Xiao, and Jaehoon Lee. Dataset distillation with infinitely wide convolutional networks. *Advances in Neural Information Processing Systems*, 34:5186–5198, 2021. 2

[32] 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. 1

[33] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021. 1

[34] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016. 2

[35] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016. 1

[36] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115:211–252, 2014. 6

[37] Andrea Schioppa, Polina Zablotskaia, David Vilar, and Artem Sokolov. Scaling up influence functions. In *AAAI Conference on Artificial Intelligence*, 2021. 2

[38] Andrea Schioppa, Polina Zablotskaia, David Vilar, and Artem Sokolov. Scaling up influence functions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8179–8186, 2022. 7

[39] Ayush Sekhari, Jayadev Acharya, Gautam Kamath, and Ananda Theertha Suresh. Remember what you want to forget: Algorithms for machine unlearning. *Advances in Neural Information Processing Systems*, 34, 2021. 2

[40] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017. 1

[41] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, 128:336–359, 2019. 2

[42] Mateusz Staniak and Przemyslaw Biecek. Explanations of model predictions with live and breakdown packages. *arXiv preprint arXiv:1804.01955*, 2018. 2

[43] Ilia Sucholutsky and Matthias Schonlau. Soft-label dataset distillation and text dataset distillation. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021. 2

[44] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017. 2

[45] Berk Ustun and Cynthia Rudin. Supersparse linear integer models for optimized medical scoring systems. *Machine Learning*, 102:349–391, 2015.

[46] Dennis Wei, Sanjeeb Dash, Tian Gao, and Oktay Günlük. Generalized linear rule models. In *ICML*, 2019. 2

[47] Yinjun Wu, Edgar Dobriban, and Susan B. Davidson. Deltagrad: Rapid retraining of machine learning models. In *International Conference on Machine Learning*, 2020. 1

[48] Xingyi Yang and Xinchao Wang. Diffusion model as representation learner. In *IEEE/CVF International Conference on Computer Vision*, 2023. 1

[49] Jingwen Ye, Yifang Fu, Jie Song, Xingyi Yang, Songhua Liu, Xin Jin, Mingli Song, and Xinchao Wang. Learning with recoverable forgetting. In *European Conference on Computer Vision*, 2022. 2

[50] Jingwen Ye, Songhua Liu, and Xinchao Wang. Partial network cloning. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2

[51] Ruonan Yu, Songhua Liu, and Xinchao Wang. Dataset distillation: A comprehensive review. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. 2

[52] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014. 2

[53] Yingyan Zeng, Jiachen T Wang, Si Chen, Hoang Anh Just, Ran Jin, and Ruoxi Jia. Modelpred: A framework for predicting trained model from training data. In *2023 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pages 432–449. IEEE, 2023. 2

[54] Bo Zhao and Hakan Bilen. Dataset condensation with differentiable siamese augmentation. In *International Conference on Machine Learning*, 2021. 2

[55] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. In *Ninth International Conference on Learning Representations 2021*, 2021. 2, 4

[56] Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2921–2929, 2016. 2

[57] Yongchao Zhou, Ehsan Nezhadarya, and Jimmy Ba. Dataset distillation using neural feature regression. *arXiv preprint arXiv:2206.00719*, 2022. 2