The Mirrored Influence Hypothesis: Efficient Data Influence Estimation by Harnessing Forward Passes

Myeongseob Ko[1] Feiyang Kang[1] Weiyan Shi[2] Ming Jin[1] Zhou Yu[2] Ruoxi Jia[1]
[1]Virginia Tech [2]Columbia University
{myeongseob, fyk, jinming, ruoxijia}@vt.edu, {ws2634, zy2461}@columbia.edu

Abstract

Large-scale black-box models have become ubiquitous across numerous applications. Understanding the influence of individual training data sources on predictions made by these models is crucial for improving their trustworthiness. Current influence estimation techniques involve computing gradients for every training point or repeated training on different subsets. These approaches face obvious computational challenges when scaled up to large datasets and models.

In this paper, we introduce and explore the Mirrored Influence Hypothesis, highlighting a reciprocal nature of influence between training and test data. Specifically, it suggests that evaluating the influence of training data on test predictions can be reformulated as an equivalent, yet inverse problem: assessing how the predictions for training samples would be altered if the model were trained on specific test samples. Through both empirical and theoretical validations, we demonstrate the wide applicability of our hypothesis. Inspired by this, we introduce a new method for estimating the influence of training data, which requires calculating gradients for specific test samples, paired with a forward pass for each training point. This approach can capitalize on the common asymmetry in scenarios where the number of test samples under concurrent examination is much smaller than the scale of the training dataset, thus gaining a significant improvement in efficiency compared to existing approaches. We demonstrate the applicability of our method across a range of scenarios, including data attribution in diffusion models, data leakage detection, analysis of memorization, mislabeled data detection, and tracing behavior in language models.

1. Introduction

As the popularity of large-scale, black-box machine learning models continues to surge across diverse applications, the need for transparency—an understanding of the factors driving their predictive behaviors—becomes increasingly critical. These models are learned from training data, and as such, an important step towards achieving transparency lies in estimating the influence of individual training data points on the model's predictions.

Extensive research on training data influence estimation has been conducted over the years [16, 18, 22, 27, 29]. Despite the diversity of techniques, they all fundamentally revolve around a central idea of assessing the counterfactual impact of a training data source:

> *How would the prediction on specific test points change if we removed a training source?* (**P1**)

One line of approaches [16] focuses on the direct evaluation of the counterfactual impact, i.e., by retraining a model on the set excluding the training source and measuring the change in the prediction. Besides the obvious computational overhead, such evaluation results suffer from a low signal-to-noise ratio due to the stochasticity in widely-used learning algorithms and are largely inconsistent across different runs [37]. To magnify the change caused by removing a single source, existing techniques mostly involve retraining models on smaller subsets of the training data and measuring a source's influence by aggregating its contribution to different subsets not containing the source [11, 12, 17]. While these methods produce more consistent influence scores, they are infeasible for large-scale models.

Another line of approaches bypasses the need for retraining by estimating the influence through the final trained model or intermediate checkpoints reached during the training process [22, 29]. In particular, they evaluate a training point's influence using the corresponding gradient of the final model or checkpoints, which effectively represents the local changes made by introducing the point into training. However, calculating gradients is not only time-intensive but also memory-inefficient compared to forward pass [26]. In our tests, it took up to 10.92 times longer and used 5.36 times more memory than inference for the Vision Trans-

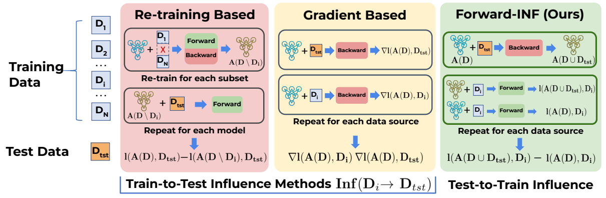
Re-training Based
Gradient Based
Forward-INF (Ours)
Training Data
D_1
D_2
D_i
D_N
Test Data
D_{tst}
Forward
Backward
$A(D \setminus D_i)$
Re-train for each subset
Repeat for each model
$l(A(D), D_{tst}) - l(A(D \setminus D_i), D_{tst})$
$\nabla l(A(D), D_{tst})$
$\nabla l(A(D), D_i)$
Repeat for each data source
$\nabla l(A(D), D_i)\ \nabla l(A(D), D_{tst})$
$A(D)$
$A(D \cup D_{tst})$
$l(A(D \cup D_{tst}), D_i)$
$l(A(D), D_i)$
Repeat for each data source
$l(A(D \cup D_{tst}), D_i) - l(A(D), D_i)$
Train-to-Test Influence Methods $\mathrm{Inf}(D_i \rightarrow D_{tst})$
Test-to-Train Influence

Figure 1. Overview of our approach and comparison with prior work which can be generally categorized into re-training-based methods and gradient-based methods. The former requires re-training models on many different subsets of training data [11, 12, 16, 17]. The latter calculates the influences based on training data gradients (`TracIn` [29] is illustrated as an example in the second column). Our proposed method `Forward-INF` features only forward pass computation for each training point, offering significant efficiency improvement.

former `vit-b-32` [8]. Furthermore, identifying the most influential training point requires computing the gradient for *every single* training data point. Combined, these challenges hinder the efficient determination of data influence, especially for large-scale datasets and large models. Figure 1 highlights the conceptual difference between existing approaches and we will defer the detailed discussion of Related Work to Appendix A.

Given the discrepancy in efficiency between gradient calculation and inference, an intriguing yet uncharted question arises: *Can we maximize the usage of forward pass when estimating influence?* We will introduce a technique that exploits the considerable efficiency gap between forward and backward passes, especially when applied to all training points. This method is anchored on the following Mirrored Influence Hypothesis.

The Mirrored Influence Hypothesis. *The train-to-test influence characterized by the problem (P1) is correlated with the test-to-train influence characterized by (P2) in which the role of training and test points is swapped:*

> *How would the prediction on a training source change if the model was trained on specific test points?* **(P2)**

More formally, consider a training dataset comprising N data sources, $D_{\text{trn}} = D_1 \cup \ldots \cup D_N$ and a test set D_{tst}. Let $\mathcal{A}$ denote the learning algorithm which takes a dataset as input and returns a model, and let $\mathcal{L}$ be a loss function. The *train-to-test* influence characterized by **P1** can be expressed as

$$\mathrm{Inf}(D_i \rightarrow D_{\text{tst}}) = \mathcal{L}(\mathcal{A}(D_{\text{trn}}), D_{\text{tst}}) - \mathcal{L}(\mathcal{A}(D_{\text{trn}} \setminus D_i), D_{\text{tst}}). \quad (1)$$

On the other hand, the *test-to-train* influence characterized by **P2** can be written as

$$\mathrm{Inf}(D_i \leftarrow D_{\text{tst}}) = \mathcal{L}(\mathcal{A}(D_{\text{trn}} \cup D_{\text{tst}}), D_i) - \mathcal{L}(\mathcal{A}(D_{\text{trn}}), D_i). \quad (2)$$

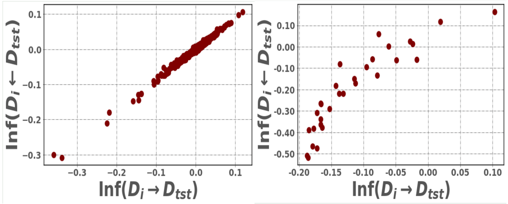

Figure 2. We observe high correlation between train-to-test influence $\mathrm{Inf}(D_i \rightarrow D_{\text{tst}})$ and test-to-train influence $\mathrm{Inf}(D_i \leftarrow D_{\text{tst}})$. The average Pearson Correlation is 0.9673 for logistic regression and 0.8851 for CNN trained on CIFAR-10.

We find that $\mathrm{Inf}(D_i \rightarrow D_{\text{tst}})$ is *highly correlated* with $\mathrm{Inf}(D_i \leftarrow D_{\text{tst}})$ for $i \in \{1, \ldots, N\}$. Figure 2 illustrates the correlation for convex and non-convex models trained on the CIFAR-10 dataset and we defer results on other datasets and models to the Appendix B due to the similarity in their trends.

Leveraging the Hypothesis. In most data influence estimation applications, the size of the test set is typically much smaller than the training set ($|D_{\text{tst}}| << |D_{\text{trn}}|$). For example, when identifying influential training points for a specific model prediction, D_{tst} represents just a single test point. Similarly, in detecting low-quality training data, D_{tst} is a small, clean reference set, usually less than 1% of the entire training set [19, 39, 41]. The Mirrored Influence Hypothesis enables leveraging this size asymmetry between training data and test samples under concurrent examination to develop more efficient influence estimation algorithms. This hypothesis allows for a shift in approach: from calculating the train-to-test influence of training data, which requires locally updating the model for each training point, to assessing the test-to-train influence. In practice, this means updating the model for the relatively few test samples and conducting forward passes on training samples. This shift effectively applies the more computationally intensive process (the backward pass) to the smaller scale test set, while the computationally lighter task (the forward pass) is applied to the larger training data, optimizing overall efficiency.

Evaluation. Leveraging the insights articulated, we develop a new influence estimation algorithm. We evaluate its performance across diverse applications handling image data, such as detecting mislabeled data, identifying data leakage, analyzing memorization, and attributing data in diffusion models. To further demonstrate its wide-ranging applicability, we extended its use to tracing behavior in language models. This new method not only showcases promising utility but also offers significantly faster performance compared to traditional train-to-test influence focused techniques. For instance, our method detects data leakage with 100% accuracy on CIFAR-10, surpassing the Influence Function [22] (30% detection rate) and TracIn [29] (0% detection rate) by being 30 and 40 times faster, respectively.

2. Delve Into the Hypothesis

In this section, we empirically assess the Mirrored Influence Hypothesis as applied to different datasets and models. Then, we study different influence approximators: Influence Function [22] and TracIn [29], and show that our hypothesized correlation between forward and backward influences holds naturally under the assumptions made by these approximators.

2.1. Empirical Study

We investigate the Hypothesis in both deterministic and stochastic learning settings, due to the distinct noise levels inherent in the influence scores of these two scenarios. Specifically, both train-to-test and test-to-train influences depend on the chosen learning algorithm $\mathcal{A}$. Algorithms like stochastic gradient descent (SGD) introduce randomness, such as through random mini-batch selection, which affects the training process and, consequently, the trained models. This randomness can cause variations in the losses evaluated on these models, ultimately impacting the influence scores, which are derived from these losses, as seen in Eqn. (1) and (2).

Particularly, the influences for individual training and test points, which are determined by the exclusion of a training point or the addition of a test data point into the training set, are typically small and can be heavily impacted by the stochasticity of the learning process. For example, prior research [36] shows that the variability in an individual point's influence due to learning stochasticity often largely surpasses their magnitude. This means that even when evaluating the same type of influence, there can be a significant inconsistency in the rankings of different data points' influence scores across different algorithm runs. Hence, to meaningfully analyze the correlation between train-to-test and test-to-train influences, it is critical to focus on settings where the signal strength in both types of influence scores is substantial enough to withstand the overshadowing effect of noise.

A Near-Noiseless Setting. We start with a near-noiseless setting where the model being trained is strongly convex and the learning algorithm is deterministic. With long enough iterations, the final model $\mathcal{A}(D)$ is guaranteed to converge to the vicinity of the global minima for any training dataset D. Specifically, we train a logistic regression model using L-BFGS [25] with L2 regularization. Due to the low noise in this setting, we can investigate the **point-to-point influence**, i.e., $|D_i| = |D_{\text{tst}}| = 1$. We use Pearson correlation and Spearman rank-order correlation coefficient to measure the correlation between the two influences. As we are mainly interested in a test point that has a high loss (e.g., in the application of debugging a misclassified test point), we select 10 test points that have the highest loss and take an average over their correlation scores.

A Noisy Setting. As discussed, training non-convex models using stochastic learning algorithms inherently produces a high level of noise in the influence between individual training and test points. Specifically, we train a convolutional neural network (CNN) using SGD with a learning rate of 0.01. To mitigate the noise, our initial approach is to average the influence scores over multiple runs of the learning algorithm. However, the number of runs required to sufficiently reduce the noise is prohibitively demanding. To effectively analyze the correlation between two types of influences, we design experiments where the influence scores across different D_i are more significant in magnitude, thus reducing the chance of these scores being overwhelmed by noise. Specifically, we assess the **group-to-group influence**, i.e., the impact of various groups of training points on a test set. Effectively, modifying a group of points in the training set would result in more substantial changes in loss, thereby making the influence scores more indicative. To amplify the effect of each group's removal or addition, we randomly mislabel 50% of samples into different classes and assign varying mislabeling ratios to each group. We use 30 different training groups with mislabeling ratios ranging linearly from 0% to 100%. D_{tst} consists solely of clean samples.

Result. In Table 1, we show Pearson and Spearman rank-order correlations between $\text{Inf}(D_i \to D_{\text{tst}})$ and $\text{Inf}(D_i \leftarrow D_{\text{tst}})$ for both settings. For the noiseless setting, we observe high correlation scores (i.e., greater than 0.96 in terms of Pearson Correlation) across different datasets. On the other hand, the Spearman correlation is relatively lower. As illustrated by Figure 2, there exists a high-density region with many similar values, which could lead to a lot of tied ranks. These ties can disrupt the monotonic relationship that Spearman correlation seeks to measure.

In the noisy setting with group-to-group influence, the correlation between the two types of influences remains high. Notably, while the Pearson correlation exhibits a decrease in comparison to the noiseless scenario, Spearman correlation retains a high value. This discrepancy arises partly because Pearson correlation, which relies on actual data values, is more susceptible to noise. In contrast, Spearman correlation employs ranks rather than raw values, which inherently provides resistance to the distorting effects of noise. Moreover, as depicted in Figure 2, the relationship between train-to-test and test-to-train influences maintains its correlation, albeit with a diminished linear characteristic. The pronounced linear correlation for point-to-point influences in the noiseless setting aligns with the theoretical insights discussed in the subsequent subsection. These insights indicate that under minor perturbations to the train-

Setting	Metric	MNIST	FMNIST	CIFAR-10
Near-Noiseless (point-to-point inf)	Pearson Spearman	0.9975 0.9027	0.9939 0.7274	0.9673 0.8069
Noisy (group-to-group inf)	Pearson Spearman	0.9640 0.9907	0.9752 0.9915	0.8551 0.8848

Table 1. The evaluation of the Mirrored Influence Hypothesis in different settings with different datasets.

ing set, such as the alteration of a single data point, it can be shown that training and test data have symmetrical roles and the two types of influences are empirically equivalent. Conversely, in the context of the group-to-group influence, the removal or addition of entire data groups leads to more significant model perturbations, which are not adequately described by current theoretical frameworks. An in-depth theoretical exploration of the correlations between the two influences under general conditions is outside this paper's scope and presents an intriguing avenue for future research.

2.2. Validity of the Hypothesis for Influence Approximators

In addition to the empirical study, we show that the hypothesized mirrored influence holds for well-known influence approximators: Influence functions [22] and TracIn [29].

We begin by introducing the notations. For ease of exposition, we will examine the validity of the Hypothesis in the case where $|D_i| = 1$ and $|D_{\text{tst}}| = 1$. The argument naturally extends to more general cases. Specifically, consider a training set consisting of n samples $D_{\text{trn}} = \{z_1, z_2, ... z_n\}$, a given test sample z_{tst}, and a neural network parameterized by $\theta \in \mathbb{R}^d$. Define the loss function $\mathcal{L}$ as the model's empirical risk on the training dataset $\mathcal{L}(\theta, D_{\text{trn}}) = \frac{1}{n}\sum_{i=1}^{n} \ell(\theta, z_i)$, where $\ell(\theta, z_i)$ denotes the loss of a predictor parameterized by θ on the training sample z_i. The optimal model parameters are given by the following empirical risk minimization: $\hat{\theta} := \arg\min_\theta \mathcal{L}(\theta, D_{\text{trn}})$.

Influence Function (IF). The idea of IF is to analyze the change in prediction loss when a training sample is upweighted infinitesimally. In particular, if we perturb the weight of a sample z from 1 to $1+\varepsilon$, the new parameter on the perturbed training dataset can be given as $\hat{\theta}_{\varepsilon,z} = \arg\min_\theta \mathcal{L}(\theta, D) + \varepsilon\ell(\theta, z)$. With assumptions on the loss function being twice-differentiable and strictly convex, the influence of an infinitesimal perturbation of a training sample z on the loss of a test sample z_{tst} can be calculated as [22]

$$\left.\frac{d\ell(\hat{\theta}_{\varepsilon,z}, z_{\text{tst}})}{d\epsilon}\right|_{\varepsilon=0} = -\nabla_\theta \ell(\hat{\theta}, z_{\text{tst}})^T H_{\hat{\theta}}^{-1} \nabla_\theta \ell(\hat{\theta}, z), \quad (3)$$

where $H_{\hat{\theta}} := \nabla_\theta^2 \ell(\hat{\theta}, D)$ denotes the Hessian. Since removing a point z is equivalent to upweighting it by $\varepsilon = -\frac{1}{n}$, for n sufficiently large and $\frac{1}{n} \to 0$, one can approximate the train-to-test influence defined in Eqn. (1) with its first-order Taylor approximation, which gives

$$\mathrm{Inf}(D_i \to D_{\text{tst}}) \approx \left(-\frac{1}{n}\right) \cdot \left[\left. -\frac{d\ell(\hat{\theta}_{\varepsilon,z}, z_{\text{tst}})}{d\varepsilon}\right|_{\varepsilon=0}\right],$$

combing with Eqn. (3), we have

$$\mathrm{Inf}(D_i \to D_{\text{tst}}) \approx -\frac{1}{n} \nabla_\theta \ell(\hat{\theta}, z_{\text{tst}})^T H_{\hat{\theta}}^{-1} \nabla_\theta \ell(\hat{\theta}, z) \quad (4)$$

Symmetrically, consider the *alternative* of adding a test sample z_{tst} with weight ε to the training dataset. The model trained with the new objective will be $\hat{\theta}_{\varepsilon,z_{\text{tst}}} = \arg\min_\theta \mathcal{L}(\theta, D) + \varepsilon\ell(\theta, z_{\text{tst}})$. Similar to Eqn. (3), the influence of training on the test sample z_{tst} with an infinitesimal weight ε on the prediction loss of a training sample z can be calculated as

$$\left.\frac{d\ell(\hat{\theta}_{\varepsilon,z_{\text{tst}}}, z)}{d\varepsilon}\right|_{\varepsilon=0} = -\nabla_\theta \ell(\hat{\theta}, z)^T H_{\hat{\theta}}^{-1} \nabla_\theta \ell(\hat{\theta}, z_{\text{tst}}) \quad (5)$$

As adding a test point into the training set is the same as setting $\varepsilon = \frac{1}{n}$, following the same procedure, we can again linearly approximate the test-to-train influence in Eqn. (2) by computing

$$\mathrm{Inf}(D_i \leftarrow D_{\text{tst}}) \approx -\frac{1}{n} \nabla_\theta \ell(\hat{\theta}, z)^T H_{\hat{\theta}}^{-1} \nabla_\theta \ell(\hat{\theta}, z_{\text{tst}}) \quad (6)$$

Due to the fact that Hessian is symmetric by definition, Eqn. (4) and Eqn. (6) are equivalent. Then, we can observe that using influence functions to approximate *influence* yields the same result for both train-to-test and test-to-train influences, which coincides with our Mirrored Influence Hypothesis.

TracIn. TracIn [29] approximates the influence of a training sample z_i^t on a testing sample z_{tst} using a first-order approximation of the model and aggregating through multiple checkpoints during the training process:

$$\mathrm{Inf}(D_i \to D_{\text{tst}}) \approx \sum_{c \in \mathcal{C}} \eta_c \nabla_\theta \ell(\hat{\theta}_c, z_{\text{tst}}) \cdot \nabla_\theta \ell(\hat{\theta}_c, z), \quad (7)$$

where c denotes an index of iteration during model training, $\mathcal{C}$ denotes the set of iteration indices where checkpoints are available, and η_c and $\hat{\theta}_c$ represent the step size and the model weights at iteration c, respectively. Then, we also consider the alternative of estimating the influence on training sample z by training the model on the test sample z_{tst} at each checkpoint, which can be easily given as

$$\mathrm{Inf}(D_i \leftarrow D_{\text{tst}}) \approx \sum_{c \in \mathcal{C}} \eta_c \nabla_\theta \ell(\hat{\theta}_c, z) \cdot \nabla_\theta \ell(\hat{\theta}_c, z_{\text{tst}}) \quad (8)$$

Again, with the TracIn approximator, the *influence* between a training-test sample pair can be calculated from both directions and achieve the same result. Echoing our intuition, these results suggest the universal application of the proposed Hypothesis in influence approximators.

3. Forward-INF: An Influence Approximation Algorithm Harnessing Forward Passes

In this section, we will present a new data influence estimation algorithm unlocked by the Hypothesis. This algorithm differs from existing methods by substituting the backward pass computations for individual training points with a forward pass, aiming to reduce computational costs.

Inspired by the Hypothesis, to rank the train-to-test influences among different training points, we can alternatively arrange them in order based on the test-to-train influences, as described in Eqn. (2). Specifically, calculating $\text{Inf}(D_i \leftarrow D_{\text{tst}})$ involves first acquiring two models $\hat{\theta} = \mathcal{A}(D)$ and $\hat{\theta}_{+D_{\text{tst}}} = \mathcal{A}(D \cup D_{\text{tst}})$. We assume that $\hat{\theta}$ is available after training. As getting $\hat{\theta}_{+D_{\text{tst}}}$ by training on $D \cup D_{\text{tst}}$ from scratch can be expansive, we propose to use continual learning and obtain this model by updating the existing model $\hat{\theta}$ with D_{tst}. Denote the resulting model as $\tilde{\theta}_{+D_{\text{tst}}}$. In Appendix C, we will show that when n is large and $\hat{\theta}$ and $\hat{\theta}_{+D_{\text{tst}}}$ are close, continually updating $\hat{\theta}$ on D_{tst} is a good approximation to training from scratch. In Appendix E, we will also empirically compare the resulting influence scores of the two settings. Finally, with the two models $\hat{\theta}$ and $\tilde{\theta}_{+D_{\text{tst}}}$, one can calculate the influence for each training source D_i by two forward passes, which give $\mathcal{L}(\tilde{\theta}_{+D_{\text{tst}}}, D_i)$ and $\mathcal{L}(\hat{\theta}, D_i)$, and then take the difference.

Implementation. The pseudo-code is provided in Algorithm 1. We call this algorithm the `Forward-INF` algorithm because it implements forward passes on the training set. Note that this algorithm still applies backward passes on the test set. However, as typically, the training size is orders of magnitude larger than the number of test points being inspected concurrently, this algorithm is usually much faster than existing methods, like Influence Functions and TracIn, which apply backward passes on the training set (and the test set).

Also, note that gradient ascent is implemented as default to update $\hat{\theta}$. This is because if the test sample is drawn from a similar distribution as the training data, the magnitude of the gradient $\nabla_\theta \ell(\theta, z_{\text{tst}})|_{\theta=\hat{\theta}}$ tends to be small. Thus, employing gradient descent would introduce a small loss difference $\mathcal{L}(\hat{\theta}_K, D_i) - \mathcal{L}(\hat{\theta}, D_i)$. By contrast, gradient ascent is likely to produce a model underfitting for points similar to the test points, thus resulting in a larger loss difference for training points, which is beneficial for comparing their influences. However, it is observed in experiments that both gradient descent and ascent perform similarly well. A detailed ablation study on this aspect is deferred to Appendix E.

Hyperparameter. The number of maximization iterations K and the learning rate α can be tuned if one has access to or can create some ground-truth "influential points." For example, one could use a subset of training data as test samples. Intuitively, the same training point would be most influential to the test. In this case, one can tune K so that the duplicates in the training set (i.e., the ground-truth influential point) are assigned with the highest influence score.

Algorithm 1 Forward-INF Algorithm

```
Require: Training set $D_{\text{trn}} = D_1 \cup \ldots \cup D_n$, Trained model $\hat{\theta}$, Target test samples $D_{\text{tst}}$, Number of continual learning iterations $K$, Learning rate $\alpha$
Ensure: Forward-INF $(D_i)$ for $i = 1, \ldots, n$
Initialize $\hat{\theta}_0 \leftarrow \hat{\theta}$
for $j = 1$ to $K$ do
    Gradient ascent: $\hat{\theta}_{j+1} \leftarrow \hat{\theta}_j + \alpha \nabla_\theta \mathcal{L}(\hat{\theta}_j, D_{\text{tst}})$
end for
for $D_i \subset D_{\text{trn}}$ do
    Forward pass of $\hat{\theta}_K$ to get $\mathcal{L}(\hat{\theta}_K, D_i)$
    Forward pass of $\hat{\theta}$ to get $\mathcal{L}(\hat{\theta}, D_i)$
    Forward-INF $(D_i)$ = $\mathcal{L}(\hat{\theta}_K, D_i) - \mathcal{L}(\hat{\theta}, D_i)$
end for
```

4. Application

In this section, we evaluate our approach to both vision and natural language processing (NLP) tasks. In particular, we apply our proposed method to the data influence estimation problem in diffusion models [37] (Section 4.1), data leakage detection [4] (Section 4.2), analysis of memorization [11] (Section 4.3) as well as mislabeled data detection (Section 4.4).

We extend our method into the NLP task to showcase the performance in the context of a model behavior tracing task [3] (Section 4.5). The scope of this study is to emphasize the method's **versatility across different applications** rather than to outperform existing *application-specific* baselines in each case. However, we will compare our test-to-train influence calculation method with existing train-to-test influence methods, both characterized by their application-agnostic nature.

4.1. Data Influence Estimation in Diffusion Model

Motivation. Recent advancements in generative models, such as stable diffusion [31], have shown remarkable performance in synthesizing high-quality images. Even though the generated images differ from the original training data, these are largely influenced by them, raising potential issues of copyright infringement [38]. Therefore, it is important to identify the training points contributing most to synthesizing a specific output.

Setup. Following [37], we leverage the stable diffusion [31] as a pre-trained model and a specific concept in ImageNet with various pre-defined prompts related to the concept for fine-tuning. This will generate a model customized to

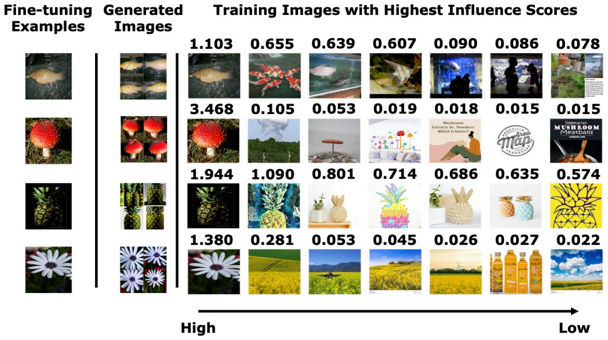

Figure 3. **Data attribution in diffusion models.** For given synthesized samples of the second column, obtained by fine-tuning with an image of the first column, we illustrate the points with the highest influences in the candidate set. Our method can assign the highest influence to the fine-tuning point which computationally influences the synthesized image the most.

the concept. The synthesized images from the customized model are computationally influenced by the fine-tuning examples by construction. Hence, the fine-tuning examples can be regarded as *ground-truth* high-influence training points. Our goal is to test whether the proposed method can indeed assign a higher influence to the fine-tuning points than the points in the pre-training set. We regard each individual synthesized image as D_{tst} and apply our method to calculate the influence of individual training points in both the pre-training set (i.e., LAION 2B [32]) and the fine-tuning set. We adopt a strategy similar to [3] to accelerate the evaluation. In particular, we create a candidate set $D_{\mathrm{candidate}} \subset D_{\mathrm{trn}}$ [3], consisting of: (1) Ground truth, which is the fine-tuning examples and (2) Distractors, which include 50 pre-training samples whose captions have the most overlap with the given prompt as well as 50 random samples drawn from the pre-training set.

Result. Figure 3 shows the qualitative results of our proposed method. We retrieved a set of 7 samples with the highest influence scores from the candidate set. We observe that the fine-tuning image consistently is ranked with the highest influence. This aligns with the design of our experiment, where the fine-tuned example is deliberately constructed to exert the most computational influence on the synthesized image. Also, it can be seen that the remaining samples retrieved mostly share similarities either in image or caption or both with the synthesized image. We also provide the quantitative results in identifying the ground truth across different sizes of candidate sets and the comparison with the baselines in the Appendix E.

4.2. Data Leakage Detection

Motivation. Data leakage refers to an oftentimes unintended mistake that is made by the creator of a machine learning model in which they accidentally share the information between the test and training data set. In this task, we apply data influence estimation methods to detect data leakage. Intuitively, for a given test point, if there exists a leakage to the training set, then the corresponding leaked duplicate point would have the highest influence. To evaluate the detection performance of the leaked samples, we use the top-k detection rate metric.

	CIFAR-10 – ResNet18				CIFAR-100 – ResNet50			
Method	Time	T-1	T-5	T-10	Time	T-1	T-5	T-100
`IF-100`	10	30	35	45	38	0	0	20
`IF-1000`	15	45	45	50	92	0	0	25
`IF-10000`	72	0	0	0	570	0	0	0
`TracIn-1`	12	2	6	7	43	6	11	19
`TracIn-3`	36	6	10	14	130	9	14	20
`TracIn-5`	60	6	10	14	215	10	13	19
`Forward-INF`	**0.3**	**100**	**100**	**100**	**1.5**	**95**	**100**	**100**

Table 2. **Data leakage detection** with top-K (T-K) accuracy (%). Performance is compared on different datasets and models. Here, `TracIn-K` denotes that we use K checkpoints for `TracIn` and `IF-L` denotes that we used L depths for `IF`. We also report computation time (in minutes) for each test point.

Setup. In training data leakage evaluation, we use ResNet-18 (RN18) and ResNet-50 (RN50) classifiers [15], trained on CIFAR-10 [23] and CIFAR-100 [7], respectively. To simulate the case of data leakage, we set $D_{tst} \subset D_{trn}$. For baseline methods, we vary the key hyperparameters (i.e., the number of depths for Taylor expansion used by LISSA in approximating the Hessian for `IF` [2, 27] and the number of checkpoints for `TracIn`). We present the details, results of ImageNet100, and visualizations in Appendix E.

Result. As shown in Table 2, our approach is effective in identifying duplicated samples with 100% and 95% top-1 detection accuracy for both RN18 and RN50 classifiers. We observe that with the larger model RN50, the performance of `IF` deteriorates when detecting data leakage. This result is consistent with [5] that `IF`s are poor estimates of the impact of excluding a training point for neural networks. Regarding the depth parameter, the algorithm's convergence to $\alpha^{-1}(G + \lambda I)^{-1} v$ depends on the condition $\alpha(\tilde{G} + \lambda I) \preceq I$ being valid at every step [13], which is rarely the case with large and complex models. Hence, errors could accumulate with each iteration, leading to the worst result for the `IF-10000`. Moreover using more iterations can result in a larger per-iteration cost. It is interesting to note that `TracIn` also exhibits poor performance in data leakage detection. Although it often identifies visually similar samples to the test example, it fails to accurately detect the

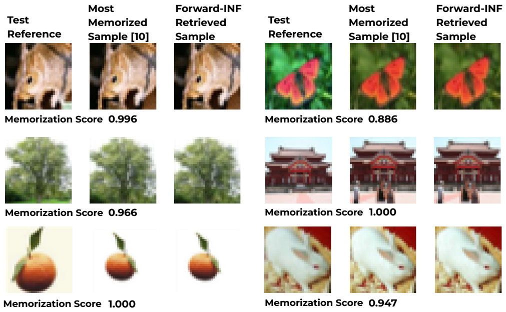

Figure 4. **Memorization analysis**, where the goal is to identify which training point's memorization is critical for predicting a specific test point. Prior work [11] proposed an algorithm to compute memorized training-test pairs, but it requires re-training the target model many times. We show that `Forward-INF` can identify the same memorized pairs without the need for re-training.

ground-truth leaked sample as illustrated in Figure 12. This discrepancy arises because, although the gradient of a duplicated sample may align with that of a test sample, numerous other training samples may align in a similar direction but with greater magnitude, yielding even higher scores than the duplicate itself. Additionally, `TracIn` in large-scale models often relies on last-layer gradient information [3], which may lead to a suboptimal approximation to ground truth attribution. As shown in Table 6, `Forward-INF` achieves the best scores in terms of both efficacy and computational efficiency, while other approaches suffer from a significant computational bottleneck (benchmarked on NVIDIA GeForce RTX 2080 Ti), as well as difficulty in finding duplicated pairs.

4.3. Memorization Analysis

Motivation. Prior research [11], studied the pair of training and test samples in which memorization of the training point is important to predicting a given test sample. Characterizing these memorized pairs is essential for understanding the learning mechanisms in neural networks. Towards that end, they proposed a re-training-based approach that directly evaluates the train-to-test influence in order to identify such so-called "influential pairs" (following their terminology). However, this approach entails high computational requirements. Here, we aim to examine whether our approach, which is much more light-weighted than [11], can identify the same pairs.

Setup. To ensure a fair comparison, we employ the same training algorithm as described in [11] to train ResNet-50 classifiers on CIFAR-100. We utilize the influential pairs provided by the authors for a qualitative comparison.

Results. In Figure 4, we compare the most influential training point as determined by our influence score calculations with the results from prior research. In each subfigure, the first two panels display the test point and the most influential training point characterized by [11], while the third one presents the top-influence point retrieved by `Forward-INF` . Figure 4 shows that our approach can identify the same influential pairs as those in [11] but without retraining models. As mentioned in [11], the high-influential pairs (with an influence score > 0.4) are near duplicated samples and benefit the most from memorization. Therefore, it leads to high memorization scores. Further qualitative results are provided in Appendix E.

4.4. Mislabeled Data Detection

Motivation. Automated identification of incorrectly labeled samples in training datasets is essential, particularly given the high incidence of human labeling errors [20]. Human judgment often varies and can be subjective, leading to inconsistent labeling, a situation that is particularly pronounced in the case of ambiguous samples. Reliable mislabeled data detection can substantially reduce the costs associated with human labeling by facilitating automated checks.

Setup. For our study, we select a random training subset D_{trn} of 2000 data points from the CIFAR-10 dataset, intentionally introducing label errors in 20% of these samples by assigning them to random classes. This data size is used due to the computational complexities associated with the Influence Function (`IF`). We then train a ResNet-18 model for 100 epochs. We follow the same setting by calculating self-influence scores, i.e. the influence of the training point onto itself, without relying on the validation data, i.e., $D_{\text{tst}} = D_{\text{trn}}$. After computing the scores for each training point, we sort them in descending order and then examine them for potential mislabeling in this sorted order.

Result. In Figure 5, we show the mislabeled data detection result. As shown in the figure, our proposed method can find over 80% of mislabeled samples within the first 300 checked samples. While `IF` would require to go through 75% of training samples to detect that many mislabeled samples, regardless of the sorting order. For the case of computing self-influence, the gradient-based methods, `IF` and `TracIn`, in fact, compute the magnitude of each point's gradient. Thus, even though the gradients of mislabeled training points might point in the opposite direction than the gradients of the clean samples, their magnitude can be smaller than those of the clean ones, resulting in the incapability of successful mislabel detection. Our method overcomes this issue by computing the loss difference between two models, $\hat{\theta}$ and $\hat{\theta}_{+D_{tst}}$, where the loss difference for clean samples will be smaller than those of the mislabeled ones, since clean samples highly likely have samples with similar labeling distribution, while the randomly mislabeled samples hardly have the support in the training dataset. Thus, they are more prone to larger loss changes. In conclusion, our approach demonstrates high mislabeled data detection performance and computation efficiency compared to existing baselines.

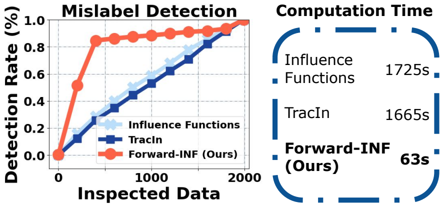

Figure 5. **Mislabeled data detection in a subset of CIFAR-10**. Left) Mislabeled data detection performance comparison between `Forward-INF` and `IF`. Right) Computation time comparison between methods. `Forward-INF` is not only effective in detecting mislabeled training data but also efficient in its computation.

4.5. Language Model Behavior Tracing

Motivation. With the growing prevalence of large language models in various applications, such as conversational agents [10, 33, 34], the importance of providing reasonable supporting evidence has become paramount. As a result, the need to trace the origin of a model's output back to specific data samples has also become indispensable to identifying the responsible training data. Driven by this motivation, we study the task of model behavior tracing regarding factual assertion, which involves identifying the training examples responsible for inducing the model to make some factual assertion at test time.

Setup. We utilize a MT5 model [40] to finetune on the FTRACE-TREx dataset [3]. We consider each training example that conveys the same fact as a "proponent" of the corresponding test example and as a "distractor" otherwise. We provide the details of the dataset and hyperparameter selection in Appendix D. Due to the demanding computation required by other direct train-to-test influence estimation methods, such as `IF`, here we only compare with `TracIn`. In order to make `TracIn` [1] more efficient, for each test sample, we limit the scope of our search to a *candidate set*, i.e., a subset of the entire training dataset, following previous studies [3, 27]. We leverage the same evaluation metrics (i.e., precision and Mean Reciprocal Rank (MRR)) described in the previous studies [3, 27]. To consider the efficiency and the performance simultaneously, we propose a time-dependent performance metric, i.e., performance in a limited time budget. This metric is realistic because, in practice, user-facing products cannot afford to spend an indefinite amount of time responding to a user's request.

Results. As shown in Table 3, we observe that `Forward-INF` outperforms the `TracIn` in terms of both metrics as `TracIn` cannot inspect enough samples within the given time. Also, counter-intuitively, the performance for `TracIn` drops when using multiple checkpoints compared with a single checkpoint. This finding is also reported in the previous studies [3, 27]. In addition, `Forward-INF` is more than four times faster than `TracIn`, in terms of the inspected number of queries per minute, even if we calculate only one layer's gradient for `TracIn`. Therefore, in the domain of large-scale models trained on vast quantities of data samples, the benefit of our method stands out. We further provide behavior-tracing experiments on paraphrased queries in Appendix E. We also provide a comparison with a simple model-independent information retrieval [30] approach in Appendix E.

Candidate Set Size	**15K**		**20K**		**Inspected Queries**
Metric	**MRR**	**Precision**	**MRR**	**Precision**	**# of Queries/Min**
`TracIn` (Single)	0.1658	0.1367	0.1532	0.1300	307.522
`TracIn` (Multi)	0.1596	0.1300	0.1508	0.1300	307.522
`Forward-INF`	**0.2101**	**0.1650**	**0.1927**	**0.1518**	**1306.323**

Table 3. **Language model behavior tracing** performance comparison of different attribution methods.

5. Conclusion

Our contribution lies in the investigation of the Mirrored Influence Hypothesis. Expanding upon this hypothesis, we have developed a novel method to estimate train-to-test influence by solving the test-to-train influence problem. This approach involves evaluating the impact of incorporating a specific test set into the training set on the prediction of a training data source. Our method can be applied broadly and contribute meaningful insights across various settings. In particular, it outperforms traditional approaches that directly compute train-to-test influence by achieving an improved tradeoff between utility and efficiency.

Limitations & Future Work. The exploration of the Hypothesis unveils many avenues for future research. Although this paper excludes heuristic enhancements to `Forward-INF` , it is expected that strategic layer selection, and incorporating strategies to counter catastrophic forgetting [21], could enhance our method's performance. Formulating a theoretical framework to formally validate the Hypothesis constitutes an intriguing direction for future studies.

6. Acknowledgment

We thank Hoang Anh Just and Himanshu Jahagirdar from the ReDS lab for their invaluable help in experiments and discussion. RJ and the ReDS lab acknowledge support through grants from the Amazon-Virginia Tech Initiative for Efficient and Robust Machine Learning, the National Science Foundation under Grant No. IIS-2312794, NSF IIS-2313130, NSF OAC-2239622, and the CCI SWVA Research Engagement Award.

References

[1] Milton Abramowitz and Irene A Stegun. *Handbook of mathematical functions with formulas, graphs, and mathematical tables*. US Government printing office, 1964. 8

[2] Naman Agarwal, Brian Bullins, and Elad Hazan. Second-order stochastic optimization for machine learning in linear time. *The Journal of Machine Learning Research*, 18(1): 4148–4187, 2017. 6

[3] Ekin Akyürek, Tolga Bolukbasi, Frederick Liu, Binbin Xiong, Ian Tenney, Jacob Andreas, and Kelvin Guu. Tracing knowledge in language models back to the training data. *arXiv preprint arXiv:2205.11482*, 2022. 5, 6, 7, 8, 14, 16, 18

[4] Björn Barz and Joachim Denzler. Do we train on test data? purging cifar of near-duplicates. *Journal of Imaging*, 6(6): 41, 2020. 5

[5] Samyadeep Basu, Philip Pope, and Soheil Feizi. Influence functions in deep learning are fragile. *arXiv preprint arXiv:2006.14651*, 2020. 6

[6] R Dennis Cook and Sanford Weisberg. *Residuals and influence in regression*. New York: Chapman and Hall, 1982. 11

[7] 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. 6, 14

[8] 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. 2

[9] Hady Elsahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon Hare, Frederique Laforest, and Elena Simperl. T-rex: A large scale alignment of natural language with knowledge base triples. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018. 14

[10] Meta Fundamental AI Research Diplomacy Team (FAIR)†, Anton Bakhtin, Noam Brown, Emily Dinan, Gabriele Farina, Colin Flaherty, Daniel Fried, Andrew Goff, Jonathan Gray, Hengyuan Hu, et al. Human-level play in the game of diplomacy by combining language models with strategic reasoning. *Science*, 378(6624):1067–1074, 2022. 8

[11] 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. 1, 2, 5, 7, 16, 18

[12] 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. 1, 2, 11

[13] Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, et al. Studying large language model generalization with influence functions. *arXiv preprint arXiv:2308.03296*, 2023. 6

[14] Zayd Hammoudeh and Daniel Lowd. Training data influence analysis and estimation: A survey. *arXiv preprint arXiv:2212.04612*, 2022. 11

[15] 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. 6

[16] Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. Datamodels: Predicting predictions from training data. *arXiv preprint arXiv:2202.00622*, 2022. 1, 2, 11

[17] Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nezihe Merve Gurel, Bo Li, Ce Zhang, Costas J Spanos, and Dawn Song. Efficient task-specific data valuation for nearest neighbor algorithms. *arXiv preprint arXiv:1908.08619*, 2019. 1, 2, 11

[18] 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. 1

[19] Hoang Anh Just, Feiyang Kang, Jiachen T Wang, Yi Zeng, Myeongseob Ko, Ming Jin, and Ruoxi Jia. Lava: Data valuation without pre-specified learning algorithms. *arXiv preprint arXiv:2305.00054*, 2023. 2, 11

[20] Davood Karimi, Haoran Dou, Simon K Warfield, and Ali Gholipour. Deep learning with noisy labels: Exploring techniques and remedies in medical image analysis. *Medical image analysis*, 65:101759, 2020. 7

[21] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017. 8

[22] 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, 3, 4, 11

[23] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 6, 14

[24] Yongchan Kwon and James Zou. Beta shapley: a unified and noise-reduced data valuation framework for machine learning. *arXiv preprint arXiv:2110.14049*, 2021. 11

[25] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989. 3

[26] Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D Lee, Danqi Chen, and Sanjeev Arora. Fine-tuning language models with just forward passes. *arXiv preprint arXiv:2305.17333*, 2023. 1

[27] Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. Trak: Attributing model behavior at scale. *arXiv preprint arXiv:2303.14186*, 2023. 1, 6, 8, 11

[28] Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian

Riedel. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*, 2019. 14

[29] 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, 2, 3, 4, 11

[30] Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. Okapi at trec-3. *Nist Special Publication Sp*, 109:109, 1995. 8, 16

[31] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, 2022. 5

[32] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35:25278–25294, 2022. 6

[33] Kurt Shuster, Jing Xu, Mojtaba Komeili, Da Ju, Eric Michael Smith, Stephen Roller, Megan Ung, Moya Chen, Kushal Arora, Joshua Lane, et al. Blenderbot 3: a deployed conversational agent that continually learns to responsibly engage. *arXiv preprint arXiv:2208.03188*, 2022. 8

[34] Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*, 2022. 8

[35] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9 (11), 2008. 14

[36] Jiachen T. Wang and Ruoxi Jia. Data banzhaf: A robust data valuation framework for machine learning. *International Conference on Artificial Intelligence and Statistics*, 2023. 3, 11

[37] Jiachen T Wang and Ruoxi Jia. Data banzhaf: A robust data valuation framework for machine learning. In *International Conference on Artificial Intelligence and Statistics*, pages 6388–6421. PMLR, 2023. 1, 5

[38] Sheng-Yu Wang, Alexei A Efros, Jun-Yan Zhu, and Richard Zhang. Evaluating data attribution for text-to-image models. *arXiv preprint arXiv:2306.09345*, 2023. 5, 14, 15, 16

[39] Zhen Xiang, David Miller, and George Kesidis. Post-training detection of backdoor attacks for two-class and multi-attack scenarios. In *International Conference on Learning Representations*, 2021. 2

[40] Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*, 2020. 8

[41] Yi Zeng, Si Chen, Won Park, Zhuoqing Mao, Ming Jin, and Ruoxi Jia. Adversarial unlearning of backdoors via implicit hypergradient. In *International Conference on Learning Representations*, 2021. 2