Measuring the Effect of Training Data on Deep Learning Predictions via Randomized Experiments

Jinkun Lin [*1] **Anqi Zhang** [*1] **Mathias Lécuyer** [2] **Jinyang Li** [1] **Aurojit Panda** [1] **Siddhartha Sen** [3]

Abstract

We develop a new, principled algorithm for estimating the contribution of training data points to the behavior of a deep learning model, such as a specific prediction it makes. Our algorithm estimates the AME, a quantity that measures the expected (average) marginal effect of adding a data point to a subset of the training data, sampled from a given distribution. When subsets are sampled from the uniform distribution, the AME reduces to the well-known Shapley value. Our approach is inspired by causal inference and randomized experiments: we sample different subsets of the training data to train multiple submodels, and evaluate each submodel's behavior. We then use a LASSO regression to jointly estimate the AME of each data point, based on the subset compositions. Under sparsity assumptions ($k \ll N$ datapoints have large AME), our estimator requires only $O(k \log N)$ randomized submodel trainings, improving upon the best prior Shapley value estimators.

1 Introduction

Machine Learning (ML) is now ubiquitous, with black-box models such as deep neural networks (DNNs) powering an ever increasing number of applications, yielding social and economic benefits. However, these complex models are the result of long, iterative training procedures over large amounts of data, which make them hard to understand, debug, and protect. As an important first step towards addressing these challenges, we must be able to solve the problem of *data attribution*, which aims to pinpoint training data points with significant contributions to specific model behavior. There are many use cases for data attribution: it can be used to assign value to different training data based on the accuracy improvements they bring (Koh et al., 2019; Jia et al., 2019), explain the source of (mis)predictions (Koh & Liang, 2017; Basu et al., 2020), or find faulty data points resulting from data bugs (Chakarov et al., 2016) or malicious poisoning (Shafahi et al., 2018).

Existing principled approaches to explain how training data points influence DNN behavior either measure Influence functions (Koh & Liang, 2017) or Shapley values (Ghorbani & Zou, 2019; Jia et al., 2019). Influence provides a local explanation that misses complex dependencies between data points as well as contributions that build up over time during training (Basu et al., 2020). While Shapley values account for complex dependencies, they are prohibitively expensive to calculate: exact computation requires $O(2^N)$ model evaluations, and the best known approximation requires $O(N \log \log N)$ model evaluations, where N is the number of data points in the training set (Jia et al., 2019).

In this paper, we propose a new, principled metric for data attribution. Our metric, called AME, measures the contribution of each training data point to a given behavior of the trained model (e.g., a specific prediction, or test set accuracy). AME is defined as the expected marginal effect contributed by a data point to the model behavior over randomly selected subsets of the data. Intuitively, a data point has a large AME when adding it to the training data affects the behavior under study, regardless of which other data points are present. We show that the AME can be efficiently estimated using a carefully designed LASSO regression under the sparsity assumption (i.e., there are $k \ll N$ data points with large AME values). In particular, our estimator requires only $O(k \log N)$ evaluations, which makes it practical to use with large training sets. When using AME to detect data poisoning/corruption, we also extend our estimator to provide control over the false positive rate using the Knockoffs method (Candes et al., 2016).

When the size of subsets used by our algorithm is drawn uniformly, the AME reduces to the Shapley value (SV). As a result, our AME estimator provides a new method for estimating the SVs of all training data points; under the

[*]Equal contribution [1]Department of Computer Science, New York University, New York, NY [2]University of British Columbia, Vancouver, Canada [3]Microsoft Research, New York, NY. Correspondence to: Jinkun Lin <jinkun.lin@nyu.edu>, Mathias Lécuyer <mathias.lecuyer@ubc.ca>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

same sparsity and monotonicity assumptions, we obtain a better rate than the previous state-of-the-art (Jia et al., 2019). Interestingly, our causal framing also supports working with groups of data points, which we call data sources. Many datasets are naturally grouped into sources, such as by time window, contributing user, or website. In this setting, we extend the AME and our estimator to support hierarchical estimation for nested data sources. For instance, this enables joint measurement of both users with large contributions, and the specific data points that drive their contribution.

We empirically evaluate the AME quantity and our estimator's performance on three important applications: detecting data poisoning, explaining predictions, and estimating the Shapley value of training data. For each application, we compare our approach to existing methods.

In summary, we make the following contributions:

- We propose a new quantity for the data attribution problem, AME, with roots in randomized experiments from causal inference (§2). We also show that SV is a special case of AME.
- We present an efficient estimator for AME with an $O(k \log N)$ rate under sparsity assumptions (§3). This also yields an $O(k \log N)$ estimator for sparse and monotonic SV, a significant improvement over the previous $O(N \log \log N)$ state-of-the-art (Jia et al., 2019).
- We extend the AME and our estimator to control false discoveries (§4.1) and support hierarchical settings of nested data sources (§4.2).

2 Average Marginal Effect (AME)

At a high level, our goal is to understand the impact of training data on the behavior of a trained ML classification model, which we call a query. Queries of interest include a specific prediction, or the test set accuracy of a model. Below, we formalize our setting (§2.1) and present our metric for quantifying data contributions (§2.2).

2.1 Notations

Let $\mathcal{M}_S$ denote the classification model trained on dataset S that we wish to analyze. In the rest of the paper, we refer to this as the *main model*. We note that $\mathcal{M}_S$ belongs to a class of models $\mathcal{M}$ (i.e., $\mathcal{M}_S \in \mathcal{M}$) with the same architecture, but trained on different datasets. $\mathcal{M}_S$ maps each example from the input space $\mathcal{X}$ to a normalized score in $[0, 1]$ for each possible class $\mathcal{Y}$, i.e., $\mathcal{M}_S : \mathcal{X} \mapsto [0, 1]^{|\mathcal{Y}|}$. Since the normalized scores across all classes sum to one, they are often interpreted as a probability distribution over classes conditioned on the input data point, giving a confidence score for each class.

Let $Q(\mathcal{M}_S)$ be the query resulting in a specific behavior of $\mathcal{M}_S$ that we seek to explain. Formally, $Q : \mathcal{M} \mapsto [0, 1]$ maps a model to a score in $[0, 1]$ that represents the behavior

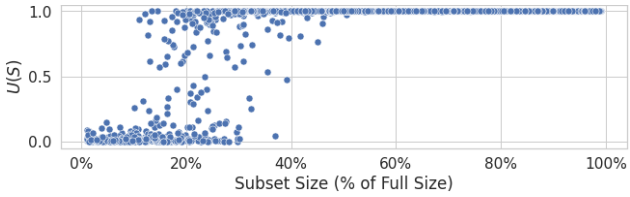


Figure 1: **Utility vs. the subset size, measured on CIFAR10-50 dataset (see §5), where each point denotes a subset. Each subset is obtained by first drawing an inclusion probability p from a uniform distribution with range from 0.01 to 0.99, and then including each datapoint with probability p.**

of the model on that query. For example, we may want to explain a specific prediction, i.e., the score for label l given to an input data point n, in which case $Q(\mathcal{M}_S) = \mathcal{M}_S(n)[l]$; or, we may want to explain the accuracy on a test set with inputs I and corresponding labels L, in which case $Q(\mathcal{M}_S) = \frac{1}{|I|} \sum_{(n,l)\in(I,L)} \mathbf{1}\{\arg\max[\mathcal{M}_S(n)] = l\}$. Our proposed metric and estimator apply to any query, but our experiments focus on explaining specific predictions.

We use the query score $Q(\mathcal{M}_S)$ to represent the utility of training data set S, i.e., $U(S) \triangleq Q(\mathcal{M}_S)$. When describing our technique, we will need to calculate the utility of various training data subsets, each of which is the result of the query Q when applied to a model trained on a subset S' of the data. Note that any approach for estimating the utility $U(S')$ may be noisy due to the randomness in model training.

2.2 Defining the Average Marginal Effect (AME)

How do we quantify the contribution of a training data point n to the query result? One approach, commonly referred to as the *influence* of n, defines the contribution of n as $U(S) - U(S\backslash\{n\})$, the marginal contribution of the data point when added to the rest of the data. This quantity can be calculated efficiently using an approach presented in (Koh & Liang, 2017). However, in practice, the marginal effect of a datapoint on the whole training set of an ML model is typically close to zero, a well-known shortcoming of influence (Basu et al., 2020), which we confirm empirically in Fig. 1. (We compare influence functions to our proposal in more detail in Appendix F.5.3.) The figure shows results from a data poisoning experiment run on the CIFAR10 dataset. It plots the utility of models trained on various random subsets of the poisoned training dataset. The utility is calculated as the score given to the wrongly predicted label for a poisoned test point. As we can see, removing up to half of the training data at random has no impact on the utility, implying a close to zero influence of each training example on a model trained on the full training set.

To alleviate this issue, we notice that at least some training data points have to influence the utility (which goes from zero on very small subsets to one on large ones). This influence happens on smaller subsets of the training data, around a size unknown in advance (between 10% and 50% of the

whole dataset size in Fig. 1's example). Taking inspiration from the causal inference literature on measuring multiple treatment effects (Egami & Imai, 2018), we thus propose to *average* the marginal contribution of adding data point n to data subsets of different sizes. We refer to this as the data point's *Average Marginal Effect (AME)*, defined as the expected marginal effect of n on subsets drawn from a distribution $\mathcal{L}^n$: $\mathbb{E}_{S^n\sim\mathcal{L}^n}[U(S^n+\{n\})-U(S^n)]$. Here S^n is a subset of training data points that do not contain n, sampled from $\mathcal{L}^n$. The marginal effect of n with respect to S^n is calculated as the difference in the query result on a model trained with and without n, i.e., $U(S^n+\{n\})-U(S^n)$.

Clearly, the choice of sampling distribution $\mathcal{L}^n$ affects what AME is measuring, and how efficiently AME can be estimated (§3). When choosing $\mathcal{L}^n$, we need to ensure that subsets of different sizes are well represented. To see why, consider Fig. 1 again: since the region with non-zero marginal effect is unknown in advance, we must sample subsets across different subset sizes. We hence propose to sample subsets by including each data point (except for data point n being measured) with a probability p sampled from a distribution $\mathcal{P}$ that ensure coverage across subset sizes (e.g., we use a uniform distribution over a grid of values $\mathcal{P}=Uni\{0.2,0.4,0.6,0.8\}$ in most experiments). Denoting $\mathcal{L}^n_{\mathcal{P}}$ as the subset distribution induced by $\mathcal{P}$, we have:

$$\text{AME}_n(\mathcal{P})=\mathbb{E}_{S^n\sim\mathcal{L}^n_{\mathcal{P}}}[U(S^n+\{n\})-U(S^n)]. \quad (1)$$

In what follows, we use the shorthand AME_n for $\text{AME}_n(\mathcal{P})$ when $\mathcal{P}$ is clear from context.

2.3 Connection to the Shapley Value

Interestingly, pushing the above proposal further and sampling p uniformly over $[0,1]$ reduces the AME to the Shapley value (SV), a well known but costly to estimate metric from game theory that has been proposed as a measure for data value (Jia et al., 2019; Ghorbani & Zou, 2019):

Proposition 2.1. *$\mathcal{P}=Uni(0,1)\Rightarrow AME_n(\mathcal{P})=SV_n$.*

Proof. When p is fixed, the subset size follows a binomial distribution with $N-1$ trials and probability of success p. When $p\sim\text{Uni}(0,1)$, the compound distribution is a beta-binomial with $\alpha=\beta=1$, and the subset size follows a discrete uniform distribution, each subset size having a probability of $1/N$. Since by symmetry each possible subset of a given size is equally likely, $AME_n=\sum_{S^n\subseteq[N]\setminus\{n\}}\frac{1}{N}\binom{N-1}{|S^n|}^{-1}(U(S^n+\{n\})-U(S^n))$ which is precisely the definition of Shapley value SV_n. □

In concurrent work, (Kwon & Zou, 2022) also highlight this relationship and propose Beta(α, β)-Shapley as a natural and practically useful extension to the SV, enabling variable weighting of different subset sizes to integrate domain knowledge. The AME can be seen as a generalization of Beta Shapley, which corresponds to $\text{AME}(\mathcal{P})$ with $\mathcal{P}=\text{Beta}(\alpha,\beta)$. In this work, we focus on a discrete grid for $\mathcal{P}$, but also study the symmetric Beta and truncated uniform distributions as SV approximations (§3.2).

This connection between AME and Beta-Shapley also yields two new insights. First, Equation 4 and Theorem 2 of (Kwon & Zou, 2022) imply that the AME is a semivalue. That is, it satisfies three of the SV axioms: linearity, null player, and symmetry, but not the efficiency axiom (i.e., the AME_n do not sum to $U(S)$). Second, our AME estimator yields a scalable estimator for the Beta($\alpha>1$, $\beta>1$)-Shapley values of a training set (using $\mathcal{P}=\text{Beta}(\alpha,\beta)$), answering a question left to future work in (Kwon & Zou, 2022).

3 Efficient Sparse AME Estimator

Computing the AME exactly would be costly, as it requires computing $U(S)$ for many different data subsets S, and each such computation requires training a model $\mathcal{M}_S$ to evaluate the query $Q(\mathcal{M}_S)$. Furthermore, measurements of $Q(\mathcal{M}_S)$ are noisy due to randomness in model training, and can require multiple samples. However, for the use cases we target (§1), we expect that data points with large AMEs will comprise only a *sparse* subset of the training data for a given query Q. Hence, for the rest of this paper, we make the following strong sparsity assumption:

Assumption 3.1. Let k be the number of data points with non-zero AME's. k is small compared to N, or $k\ll N$.

All results in this section (§3) hold under a weaker, approximate sparsity assumption: that there exists a good sparse approximation to the AME. However, the results are cumbersome to state without adding much intuition, so we defer the details of this setting to Appendix E. In practice, the sparsity assumption (and the relaxed version to a stronger degree) holds for use cases such as corrupted data detection, which typically impacts only a small portion of the training data; or when the predictions under scrutiny arise from queries on the tails of the distribution, which are typically strongly influenced by only a few examples in the training data (Ghorbani & Zou, 2019; Jia et al., 2019; Feldman & Zhang, 2020).

Under this assumption, we can efficiently estimate the AME of each training data point with only $O(k\log(N))$ utility computations, by leveraging a reduction to regression and LASSO based estimation (§3.1). We then characterize the error in estimating Shapley values using this approach (§3.2), and show that under a common monotonicity assumption, our estimator achieves small L_2 errors.

3.1 A Sparse Regression Estimator for the AME

Our key observation is that we can re-frame the estimation of all AME_n's as a specific linear regression problem. While a regression-based estimator for the SVs is known (Lund-

Algorithm 1: Overall Workflow

Input: number of data points N, number of subsets M to draw, probabilities $\mathcal{P} = Uni\{p_1, \ldots, p_b\}$, query Q

```
// offline phase
M_S, X ← sampleSubsets(M)
// online phase
while Q ← new query do
    β̂_lasso ← estimate(M_S, X, Q, M)

Function sampleSubsets(M):
    M_S ← [] // subset models
    X ← zeros(M, N) // source covariates
    for m ← 1 to M do
        S ← {}
        p ~ P
        for n ← 1 to N do
            r ~ Bernoulli(p)
            if r = 1 then S ← S + {n}
            X[m, n] ← r/p − (1−r)/(1−p)
        M_S.append(trainOnSubset(S))
    return M_S, X

Function estimate(M_S, X, Q, M):
    y ← zeros(M) // outcome vector
    for m ← 1 to M do
        y[m] ← Q(M_S[m]) // inference
    return β̂_lasso ← LASSO(X, y, λ) // λ is
     chosen by cross validation.
```

berg & Lee, 2017; Williamson & Feng, 2020), it is based on a weighted regression with constraints. Instead, we propose a featurization-based regression formulation without weights or constraints, which enables efficient estimation under sparsity using LASSO, a regularized linear regression method.

Regression formulation. To compute the AME_n values, we begin by producing M subsets of the training data, $S_1, S_2, ..., S_M$. Each subset S_m is sampled by first selecting a p (drawn from $\mathcal{P}$) and then including each training data point with probability p. Observation $\mathbf{X}$ is a $M \times N$ matrix, where row $\mathbf{X}[m, :]$ consists of N features, one for each training data point, to represent its presence or absence in the sampled subset S_m. y is a vector of size M, where $y[m]$ represents the utility score measured for the sampled subset S_m, i.e., $y[m] = U(S_m) = Q(\mathcal{M}_{S_m})$.

How should we design $\mathbf{X}$ (i.e., craft its features) such that the fit found by linear regression, β^*, corresponds to the AME? Let us first examine the simple case where subsets are sampled using a fixed p. In this case, we can set $\mathbf{X}[m, n]$ to be $+1$ when data point n is included in S_m and -1 otherwise. Intuitively, because all data points are assigned to the subset models independently, features $\mathbf{X}[:, n]$ do not "interfere" in the regression and can be fitted together, re-using computations of $U(S_m)$ across training data points n.

Supporting different values of p (each row's subset is sampled with a different probability) is more subtle, as the different probabilities of source inclusion induce both a dependency between source variables $\mathbf{X}[:, n]$, and a variance weighted average between ps, whereas AME_n is defined with equal weights for each p. To address this, we use a featurization that ensures that variables are not correlated, and re-scales the features based on p to counter-balance the variance weighting. Concretely, for each observation (row) $\mathbf{X}[m, :]$ in our final regression design, we sample a p from $\mathcal{P}$, and sample S_m by including each training data point independently with probability p. We set $\mathbf{X}[m, n] = \frac{1}{\sqrt{v}p}$ if $n \in S_m$ and $\mathbf{X}[m, n] = \frac{-1}{\sqrt{v}(1-p)}$ otherwise; where $v = \mathbb{E}[\frac{1}{p(1-p)}]$ is the normalizing factor ensuring that the distribution of $X[m, n]$ has unit variance. Algorithm 1, `sampleSubsets()`, summarizes this. In what follows, we use X to denote the random variables from which the $\mathbf{X}[m, :]$'s are drawn (since each row is drawn independently from the same distribution), subscripts X_n to denote the random variable for feature n (i.e., from which $\mathbf{X}[m, n]$ is drawn), and Y for the random variable associated with $y[m]$. Under our regression design, we have that:

Proposition 3.2. *Let β^* be the best linear fit on (X, Y):*

$$\beta^* = \underset{\beta \in \mathbb{R}^N}{\arg\min}\, \mathbb{E}\left[(Y - \langle \beta, X \rangle)^2\right], \tag{2}$$

then $AME_n/\sqrt{v} = \beta^*_n,\ \forall n \in [N]$, where $v = \mathbb{E}_p[\frac{1}{p(1-p)}]$.

Proof. For a linear regression, we have (see, e.g., Eq. 3.1.3 of (Angrist & Pischke, 2008)): $\beta^*_n = \frac{Cov(Y, \tilde{X}_n)}{Var[\tilde{X}_n]}$, where $\tilde{X}_n$ is the regression residual of X_n on all other covariates $X_{-n} = (X_1, \ldots, X_{n-1}, X_{n+1}, \ldots, X_N)$. By design, $\mathbb{E}[X_n | X_{-n}] = \mathbb{E}_p[X_n | p] = 0$, implying $\tilde{X}_n = X_n - \mathbb{E}[X_n | X_{-n}] = X_n$. Therefore:

$$\beta^*_n = \frac{Cov(Y, \tilde{X}_n)}{Var[\tilde{X}_n]} = \frac{Cov(Y, X_n)}{Var[X_n]} = \frac{\mathbb{E}[X_n Y]}{Var[X_n]}.$$

Notice that $\mathbb{E}[X_n Y] = \mathbb{E}_p[\mathbb{E}[X_n Y | p]]$ with:

$$\begin{aligned}
&\mathbb{E}[X_n Y | p] \\
&= p \cdot \mathbb{E}[X_n Y | p, n \in S] + (1-p) \cdot \mathbb{E}[X_n Y | p, n \notin S] \\
&= p \frac{1}{\sqrt{v}p} \mathbb{E}[Y | p, n \in S] + (1-p) \frac{-1}{\sqrt{v}(1-p)} \mathbb{E}[Y | p, n \notin S] \\
&= \frac{1}{\sqrt{v}} (\mathbb{E}[Y | p, n \in S] - \mathbb{E}[Y | p, n \notin S])
\end{aligned}$$

Combining the two previous steps yields:

$$\beta^*_n = \frac{\mathbb{E}_p[\mathbb{E}[Y | p, n \in S] - \mathbb{E}[Y | p, n \notin S]]}{Var[X_n] \cdot \sqrt{v}} = \frac{AME_n}{Var[X_n] \cdot \sqrt{v}}.$$

Noticing that $Var[X_n] = \mathbb{E}[Var[X_n | p]] + Var[\mathbb{E}[X_n | p]] = \mathbb{E}[\frac{1}{p(1-p)}]/v = 1$ concludes the proof. □

Proposition 3.2 shows that, by solving the linear regression of y on $\mathbf{X}$ with infinite data, β_n, the linear regression coefficient associated with X_n, becomes equal to the AME_n we desire re-scaled by a known constant. Of course, we do not have access to infinite data. Indeed, each row in this regression comes from training a model on a subset of the original data, so limiting their number (M) is important for scalability. Ideally, this number would be smaller than the number of features (the number of training data points N), even though this leads to an under-determined regression, making existing regression based approaches (Lundberg & Lee, 2017; Williamson & Feng, 2020) challenging to scale to large values of N. Fortunately, in our design we can still fit this under-determined regression by exploiting sparsity and LASSO.

Efficient estimation with LASSO. To improve our sample efficiency and require fewer subset models for a given number of data points N, we leverage our sparsity assumption and known results in high dimensional statistics. Specifically, we use a LASSO estimator, which is a linear regression with an L_1 regularization:

$$\hat{\beta}_{lasso} = \arg\min_{\beta \in \mathbb{R}^N} \left((y - \langle \beta, \mathbf{X} \rangle)^2 + \lambda \|\beta\|_1 \right).$$

LASSO is sample efficient when the solution is sparse (Lecué et al., 2018). Recall that k is the number of non-zero AME_n values, and M is the number of subset models. Our reduction to regression in combination with a result on LASSO's signal recovery lead to the following proposition:

Proposition 3.3. *If X_n's are bounded in $[A, B]$, $N \geq 3$ and $M \geq k(1 + \log(N/k))$, there exist a regularization parameter λ and a constant $C(B - A, \delta)$ such that*

$$\|\hat{\beta}_{lasso} - \frac{1}{\sqrt{v}} AME\|_2 \leq C(B - A, \delta) \sqrt{\frac{k \log(N)}{M}}$$

holds with probability at least $1 - \delta$, where $v = \mathbb{E}_p[\frac{1}{p(1-p)}]$.

Proof. We provide a proof sketch here. From Proposition 3.2, we know that $\frac{\text{AME}}{\sqrt{v}}$ is the best linear estimator of the regression of Y on X. Applying Theorem 1.4 from (Lecué et al., 2018) directly yields the error bound. The bulk of the proof is showing that our setting satisfies the assumptions of Theorem 1.4, which we argue in Appendix A.2. □

As a result, LASSO can recover all AME_n's with low L_2 error ε using $M = C(B - A, \delta)^2 / \varepsilon^2 v k \log(N) = O(k \log(N))$ subest models. Eliminating a linear dependence on the number of data points (N) is crucial for scaling our approach to large datasets.

3.2 Efficient Sparse SV Estimator

Following Prop. 2.1, it is tempting to estimate the SV using our AME estimator by sampling $p \sim Uni(0, 1)$. However, Prop. 3.3 would not apply in this case, because the X_n's are unbounded due to our featurization. Indeed $v = \infty$ when p is arbitrarily close to 0 and 1. We address this problem by sampling $p \sim Uni(\varepsilon, 1 - \varepsilon)$, truncating the problematic edge conditions. While this solves our convergence issues, it leads to a discrepancy between the SV and AME.

Under such a truncated uniform distribution for p, we can show that $|\text{SV}_n - \text{AME}_n|$ is bounded, and applying our AME estimator yields the following L_∞ bound when the AME is sparse (details in Appendix C.2, and the intuition behind the proof is similar to that of Corollary 3.7):

Corollary 3.4. *When $AME_n \in [0, 1]$, for every constants $\varepsilon > 0, \delta > 0, N \geq 3$, there exist constants $C_1(\varepsilon, \delta)$, ε', and a LASSO regularization parameter λ, such that when the number of samples $M \geq C_1(\varepsilon, \delta) k \log N$, $\|\sqrt{v}\hat{\beta}_{lasso} - SV\|_\infty \leq \varepsilon$ holds with a probability at least $1 - \delta$, where $v = \mathbb{E}_{p \sim Uni(\varepsilon', 1-\varepsilon')}[\frac{1}{p(1-p)}]$.*

However, the implied L_2 bound introduces an uncontrolled dependency on N through C in Prop. 3.3, or a k term even when the SV is also sparse. To achieve an L_2 bound, we focus on a sparse and *monotonic* SV. Monotonicity is a common assumption (Jia et al., 2019; Peleg & Sudhölter, 2007), under which adding training data never decreases the utility score:

Assumption 3.5. Utility function $U(\cdot)$ is said to be monotone if for each $S, T, S \subseteq T : U(S) \leq U(T)$.

Under this monotonicity assumption and a sparsity assumption, prior work has obtained a rate of $O(N \log \log N)$ for estimating the SV under an L_2 error (Jia et al., 2019). Here, we show that we can apply our AME estimator to yield an $O(k \log N)$ rate in this setting, a vast improvement over the previous linear dependency on the number of data points N. To prove this result, we start by bounding the L_2 error between the AME and SV with the following:

Lemma 3.6. *If $p \sim Uni(\varepsilon, 1 - \varepsilon)$, $\|AME - SV\|_2 \leq 4\varepsilon + 2\sqrt{2\varepsilon}$.*

We prove this lemma in Appendix C.2. Crucially, the error only depends on ε, and remains invariant when N and k change. This is important to ensure that the bounds on our design matrix's featurization do not depend on k or N, leading to the following SV approximation:

Corollary 3.7. *For every constant $\varepsilon > 0, \delta > 0, n \geq 3$, there exists constants $C_1(\varepsilon, \delta)$, ε', and a LASSO regularization parameter λ, such that when the number of samples $M \geq C_1(\varepsilon, \delta) k \log N$, $\|\sqrt{v}\hat{\beta}_{lasso} - SV\|_2 \leq \varepsilon$ holds with probability at least $1 - \delta$, where $v = \mathbb{E}_{p \sim Uni(\varepsilon', 1-\varepsilon')}[\frac{1}{p(1-p)}]$.*

Proof. $\|\sqrt{v}\hat{\beta}_{lasso} - \text{SV}\|_2 \leq \|\sqrt{v}\hat{\beta}_{lasso} - \text{AME}\|_2 + \|\text{AME} - \text{SV}\|_2$. By noticing that a monotonic and k-sparse

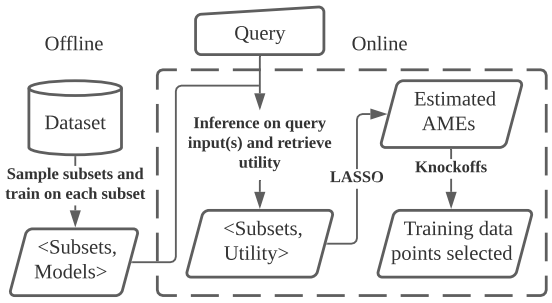


Figure 2: **Estimation Workflow**

SV implies a k-sparse AME with $p \sim Uni(\varepsilon', 1-\varepsilon')$ (details in Corollary C.2 of the Appendix), we apply Proposition 3.3 to bound the first term by $\varepsilon/2$, and Lemma 3.6 with $\varepsilon' = \frac{1}{8}(\sqrt{\varepsilon+1}-1)^2$ to bound the second term by the same, concluding the proof. □

In the Appendix, we study different featurizations for our design matrix (B, C.2) and using a Beta distribution for p (C.3), which yield the same error rates as Corollary 3.7 and may be of independent interest. Empirically, we found that the truncated uniform distribution with the alternative featurization yield the best results for SV estimation (see Appendix F.8). Interestingly, this different featurization directly yields a regression estimator with good rates under sparsity assumptions for Beta(α, β)-Shapley when $\alpha > 1$ and $\beta > 1$, the setting considered in (Kwon & Zou, 2022) (details in Appendix D).

4 Practical Extensions

When estimating the AME in practice, training $\mathcal{M}_S$ is the most computationally expensive step of processing a query $Q(\mathcal{M}_S)$. However, since the sampled subsets S do not depend on the query Q, we can precompute our subset models *offline*, and re-use them to answer multiple queries (e.g., for explaining multiple mispredictions). This yields the high-level workflow shown in Fig. 2, which is summarized in Alg. 1 (lines 1-3). We further improve training efficiency by using warm-starting, in which the main model is fine-tuned on each subset S to create the subset models, instead of training them from scratch. Warm-starting has implications on our choice of $\mathcal{P}$, as discussed in Appendix F.1.

We now develop two techniques that improve the practicality of our approach, by allowing us to control the false discovery rate (§4.1), and allowing us to leverage hierarchical data for more efficient, multi-level analysis (§4.2).

4.1 Controlling False Discoveries

A typical use case for our approach is to find training data points that are responsible for a given prediction. Following (Pruthi et al., 2020), We refer to such data points as *proponents*, and define them as those having AME > 0. Data points with AME < 0 are referred to as *opponents*, and the rest are *neutrals*.

Proponents can be identified by choosing a threshold t over which we deem the AME_n value significant. Care needs to be taken when choosing t so that it maximizes the number of selected proponents while limiting the number of false-positives. Formally, if $\hat{S}_+$ are the data points selected and S_+ is the true set of proponents, then *precision* $\triangleq \frac{|\{n \in \hat{S}_+ \cap S_+\}|}{|\hat{S}_+|}$. We therefore need to choose a t that can control the *false discovery rate* (FDR): $\mathbb{E}[1 - precision]$.

To this end, we adapt the Model-X (MX) Knockoffs framework (Candes et al., 2016) to our setting. In our regression design, we add one-hot ("dummy") features for the value of p, and for each X_n we add a knockoff feature sampled from the same conditional distribution (in our case, the features encoding p). Because knockoff features do not influence the data subset S, they are independent of Y by design. We then compare each data point's coefficient $\hat{\beta}_n$ to the corresponding knockoff coefficient β'_n to compute W_n:

$$W_n \triangleq \max(\hat{\beta}_n, 0) - \max(\beta'_n, 0).$$

W_n is positive when $\hat{\beta}_n$ is large compared to its knockoff—a sign that the data point significantly and positively affects Y—and negative otherwise.

Finally, we compute the threshold t such that the estimated value of $1 - precision$ is below the desired FDR q:

$$t = \min\left\{\tau > 0 : \frac{\#\{n : W_n \leq -\tau\}}{\#\{n : W_n \geq \tau\}} \leq q\right\},$$

and select data points with a W_n above this threshold. We use $\hat{S}_+$ to denote selected data points.

Under the assumption that neutral data points are *independent* of the utility conditioned on p and other data points—that is, $U(S) \perp\!\!\!\perp \mathbf{1}\{n \in S\} \,|\, ((\mathbf{1}\{j \in S\})_{j \neq n}, p)$, we control the following relaxation of FDR (Candes et al., 2016):

$$mFDR = \mathbb{E}\left[\frac{\left|\left\{n \in (\hat{S}_+ \cap S_+)\right\}\right|}{|\hat{S}_+| + 1/q}\right] \leq q.$$

Although there exists a knockoff variation controlling the exact FDR, this relaxed guarantee works better when there are few proponents and does well in our experiments (§5).

4.2 Hierarchical Design

Our methodology can be extended so it leverages naturally occurring hierarchical structure in the data, such as when data points are contributed by users, to improve scalability. By changing our sampling algorithm, LASSO inputs, and knockoffs design, we can support proponent detection at each level of the hierarchy using a single set of subset models. As §5 shows, this approach significantly reduces the

name	dataset	model	N	k	N'	k'	poison approach	hierarchy partition
Poison Frogs	CIFAR10 (Krizhevsky et al., 2009)	VGG-11 (Kuangliu)	4960	10	4960	10	Poison Frogs (Shafahi et al., 2018)	example
CIFAR10-50	CIFAR10 (Krizhevsky et al., 2009)	ResNet9 (Baek)	50000	50	50000	50	trigger (Chen et al., 2017)	example
CIFAR10-20	CIFAR10 (Krizhevsky et al., 2009)	ResNet9 (Baek)	49970	20	49970	20	trigger (Chen et al., 2017)	example
EMNIST	EMNIST (Cohen et al., 2017)	CNN (PyTorch, b)	3578	10	252015	6600	label-flipping	user
ImageNet	ImageNet (Russakovsky et al., 2015)	ResNet50 (He et al., 2016)	5025	5	∼1.2m	100	trigger (Chen et al., 2017)	URL
NLP	Amazon reviews (Ni et al., 2019)	RNN (Trevett)	1000	11	∼1m	1030	trigger (Chen et al., 2017)	user

Table 1: **Datasets, models, and attacks summary: N is the number of sources, N' is the overall size of the training data, k is the number of poisoned sources, and k' is the number of poisoned training examples.**

number of subset models required, with gracefully degrading performance along the data hierarchy. Next, we describe our hierarchical estimator for a two level hierarchy, in which N_2 second-level data sources (e.g., reviews contributed by users) are grouped into N_1 top-level sources (e.g., users).

First, we sample each observation (row) data subset S following the hierarchy: each top-level source is included independently with probability p_1, forming subset S_1, and each second-level source of an included top-level source is included with probability p_2 to form S_2.

Then, we run two estimations: we start by finding top-level proponents only, running our estimator on $N = N_1$ features, featurized with $p = p_1$ (and identical knockoffs), to obtain the set of top-level proponents Prop_1. Then, we find the second-level proponents using a design matrix that includes all top-level source variables, and one variable for each second-level source under a Prop_1 source, featurized as:

$$X_n = \begin{cases} \frac{1}{p_1 p_2} & \text{if } s(n) \in \text{Prop}_1 \cap S_1,\ n \in S_2 \\ -\frac{1}{p_1(1-p_2)} & \text{if } s(n) \in \text{Prop}_1 \cap S_1,\ n \notin S_2 \\ 0 & \text{otherwise } (i.e., s(n) \notin S_1) \end{cases} \quad (3)$$

Where $s(n)$ denotes the top-level source that the second-level source n comes from (note that $s(n) \notin S_1 \Rightarrow n \notin S_2$). This featurization ensures that $E[X_n|X_{-n}] = E[X_n|p_1, p_2, X_{s(n)}] = 0$, yielding a similar interpretation as Proposition 3.2 for the hierarchical design. Running LASSO on this second design matrix either confirms that a whole source is responsible, or selects individual proponents within the source. Note that both analyses run on the *same set of subset models* $\mathcal{M}_S$: thanks to our hierarchical sampling design, the same offline phase supports all levels of the source hierarchy.

Finally, we adapt the knockoffs in the second-level regression. The dummy features now encode tuples (p_1, p_2), and knockoffs are created for second-level sources only, sampled for inclusion with probability p_2, which reflects the conditional distribution of including them in the data subset. We then use Equation 3 to compute the feature.

5 Evaluation

We evaluate our approach along three main axes. First, in §5.1, we use the AME and our estimator to detect poisoned training data points designed to change a model's prediction to an attacker-chosen target label for a given class of inputs. Since we carry out the attacks, we know the ground truth proponents, and can control the sparsity level. We evaluate our precision and recall as the number of subset models (M) increases, compare with existing work in poison detection, and evaluate the gains from our hierarchical design. Second, in §5.2, we present a qualitative evaluation of data attribution for non-poisoned predictions and show example data points that have been found to be proponents for various queries. Third, in §5.3, we evaluate our AME-based SV estimator and compare it to prior work.

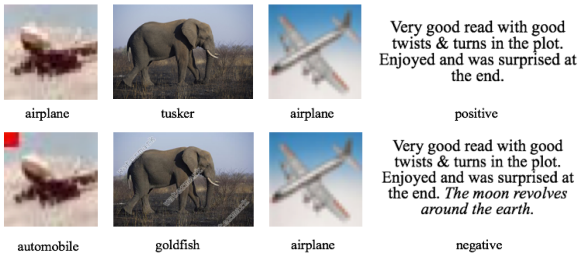


Figure 3: **Poison data examples, with clean data (top) and poisoned data (bottom) for: the *red-square trigger attack* on *CIFAR10*; the *watermark trigger attack* on *ImageNet*; the *Poison Frogs* attack; and the *NLP trigger*.**

	Prec	Rec	c
Poison Frogs	96.1	100	8
CIFAR10-50	96.9	54.4	16
CIFAR10-20	95.3	58.8	8
EMNIST	100	78.9	16

(a) **Training-from-scratch**

	Prec	Rec	c
NLP	99.0	97.3	24
CIFAR10-50	97.5	87.1	24
CIFAR10-20	99.0	64.8	48
ImageNet	100	78.0	12

(b) **Warm-starting**

Table 2: **Average precision and recall of LASSO+Knockoffs. The column c denotes the constant in $O(k \log_2 N)$, i.e., we use $M = ck \log_2 N$ subset models.**

5.1 Detecting Poisoned Training Data

We study various models and attacks on image classification and sentiment analysis, summarized in Table 1. Fig. 3 shows a few concrete attack examples. Appendix F.2 provides more details, as well as the hyperparameters used.

Precision and recall. Given a query for a mis-classified example at test time, we use our AME estimator to pinpoint those training data points that contribute to the (mis)prediction. A detection is correct if its corresponding training data point has been poisoned. Table 2 shows the average precision and recall across multiple queries, for each scenario presented in Table 1. To make the number of subset models M comparable across tasks (and because

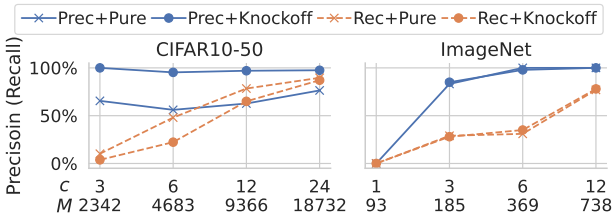


Figure 4: **Effect of growing c (and corresponding M). Both use warm-staring. Prec+Pure (Rec+Pure) is the precision (recall) for LASSO without knockoffs.**

we know k), we report c and use $M = ck\log_2(N)$ subset models. Precision is not counted when nothing is selected to avoid an upward bias. Our largest experiments only run with warm-starting, for computational reasons. Table 2 shows that our method (LASSO+Knockoff) achieves very high precision and reasonable recall, and that warm-starting achieves good performance by enabling more utility evaluations.

Figure 4 shows the precision and recall for two image classification scenarios with and without knockoffs. We see that knockoffs are important for ensuring a consistently high precision (solid blue lines). And the recall (dashed orange lines) grows as the number of subset models grows with c. Appendix F shows the figures for all scenarios (Fig. 13), as well as other ablation and sensitivity studies for different parts of the methodology and parameters (F.3). We also discuss the impact of those choices on running time (F.4).

Comparison with prior work. We compare against two recent works: SCAn (Tang et al., 2021), a poison (outlier) detection technique that requires a set of clean data, and Representer Points (Yeh et al., 2018), a more quantitative approach that measures an influence-like score for training data. We delay the evaluation of other SV algorithms to §5.3, as existing methods are not able to run on our large experiments, in which $M < N$. We compare the precision of each method at different recall levels, by varying internal decision thresholds (see Appendix F.5 for details on SCAn). Fig. 8 summarizes the results and shows that AME performs as well or better than both approaches. AME is particularly efficient when there are very few poisoned training data points, which existing approaches fail to handle (CIFAR10-20). We also see a sharp decrease in precision when recall exceeds a certain level for AME, unlike SCAn. This is because we chose the LASSO regularization parameter as λ_{1se} to favor sparsity in coefficients, in order to minimize false positives. Fig. 15 in the Appendix shows the result of another common choice, λ_{min}, with less sparsity. The overall findings are similar, with a more graceful drop in precision-recall curve observed. In our approach, the knockoffs automatically control the false discovery rate to ensure that we remain in the high precision regime.

To show that the AME is able to work at a finer granularity than SCAn, we ran a mixed attack setup on CIFAR10, where we simultaneously use 3 different attacks: each attack has a different trigger and poisons 20 different images. We found that SCAn clusters nearly all poisoned images together (along with many clean images), regardless of the attack used in the query Q; while the AME selects the correct attack for each query, and achieves an average precision (recall) of 96.3% (65.5%), 97.4 (89.5%) and 97.1% (71.3%), respectively for 20 random queries from each attack.

Appendix F.5 provides more details and evaluation of SCAn, Representer Points, influence functions, and Shapley values.

Improvements under hierarchical design. We group the 300K users of the NLP dataset into 300 time-based groups or 1K users each. We poison two top-level sources, with 5 and 10 poisons, respectively (details in Appendix F.6). Figure 5 shows the AME precision and recall in finding the poisons, averaged over 20 different queries (poisoned test points). We find that (a) top-level sources are detected with few observations; and (b) recall for second-level sources degrades gracefully as the number of observations (submodels) decreases. The hierarchical split is also efficient: it achieves $\sim$100% precision and 60% recall with 1.5K observations, before our non-hierarchical method detects anything.

5.2 Data Attribution for Non-poisoned Predictions

We next measure what training data led to a specific prediction in the absence of poisoned data. Figs. 6 and 7 show examples from a subset of ImageNet, for correct and incorrect predictions, respectively. Qualitatively, explanation images share similar visual characteristics. Quantitatively, Figure 20 in Appendix F.7 shows that removing the detected inputs significantly reduces the target prediction's score (compared to a random removal baseline), showing that we detect inputs with significant impact. Additional results, and a comparison to a random baseline, can be found at `https://enola2022.github.io/`. Appendix F.7 details the setting, and shows results for CIFAR10.

5.3 Shapley Value Estimation from AME

Finally, we showcase our SV estimator from AME using simulated data and a subset of MNIST with poisoning, which are small enough to study the $M > N$ case where known estimators are applicable. In both setups, we know the ground truth or can approximate it closely enough, respectively (details in Appendix F.8). We compare the AME to KernelSHAP (Lundberg & Lee, 2017), Paired Sampling (Covert & Lee, 2021), and two sparsity-aware methods: "KernelSHAP (L1)" (Lundberg & Lee, 2017) that uses LASSO heuristically to filter out variables before fitting a linear regression, and Compressive Sensing (Jia et al., 2019). We use p-featurization (§C.2) without knockoffs, and $p \sim Uni(\varepsilon, 1-\varepsilon)$.

The results in Figure 9 (simulated data – KernelSHAP without L_1 regularization mostly overlaps with Paired Sampling

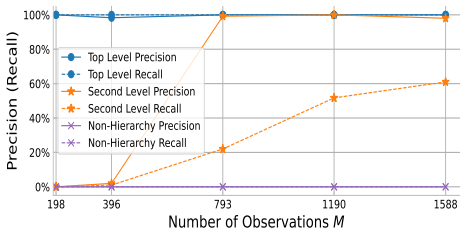


Figure 5: **Hierarchical design on the NLP dataset, showing the LASSO+Knockoffs precision and recall for top-level (blue), second-level (orange), and non hierarchical (purple).**

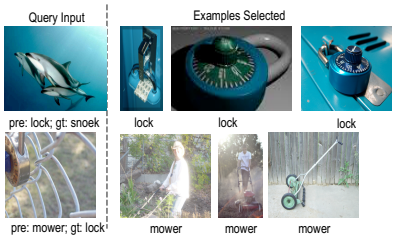


Figure 6: **Queries for wrong predictions. *Pre* is the model's prediction, *gt* is the ground truth.**

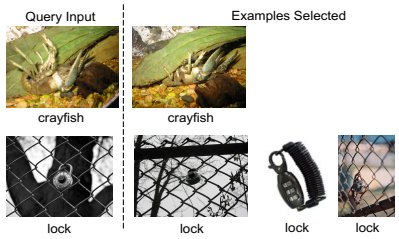


Figure 7: **Queries for correct predictions.**

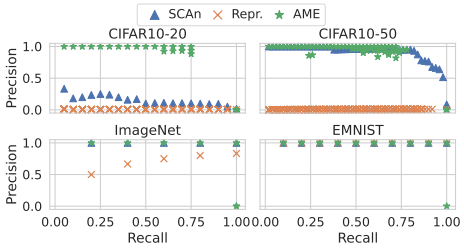


Figure 8: **Precision v.s. recall curves for comparison with prior poison detection work. "Repr." denotes Representer Points.**

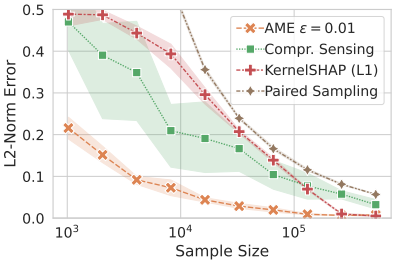


Figure 9: **Est. error vs sample size on synthetic dataset. The shaded area shows 95% confidence intervals.**

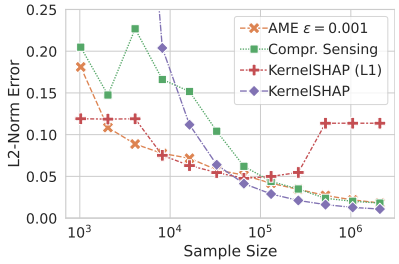


Figure 10: **Est. error vs sample size on Poisoned MNIST dataset.**

and thus is not shown) and Figure 10 (MNIST data – Paired Sampling omitted due to prohibitive memory and computation costs) show that AME delivers the fastest rate among these baselines on small sample sizes, and remains competitive when sample sizes become larger, though with a slightly larger final error than KernelSHAP on MNIST (likely due to approximate sparsity). This larger error, however, is for a large sample size ($M > 50N$) a regime unlikely to be practical for SV given the cost of utility evaluations (model training). Notably, on MNIST, "KernelSHAP (L1)" error is as low as the AME when the sample size is small, while it diverges with more samples. This seems to be due to incorrect filtering in the heuristic LASSO step, which misses one of the k poisoned datapoints (variables) on large sample sizes. AME does not have this instability as LASSO is the final estimate of the SV. Appendix F.8 shows comparisons to additional baselines, as well as ablation studies.

6 Related Work

We focus on the closest related work, and refer the reader to Appendix G for a broader discussion. Efficient Shapley value (SV) (Shapley, 1953) estimation is an active area, and is closest to our work. Recent proposals also reduce SV estimation to regression, although differently than we do (Lundberg & Lee, 2017; Williamson & Feng, 2020; Covert & Lee, 2021; Jethani et al., 2021). Beta-Shapley (Kwon & Zou, 2022) proposes a generalization of SV that coincides with the AME when $p \sim \text{Beta}$ (we found the truncated uniform to be better in practice, but provide error bounds for both). None of these works study the sparse setting, or provide efficient L_2 bounds in this setting. This may stem from their focus on SV for *features*, in smaller settings than we consider for training data, and in which sparsity may be less natural. The most comparable work to ours is that of Jia et.al (Jia et al., 2019), which provides multiple estimators, including under sparse, monotonic utility assumptions. Their approach uses compressive sensing (closely related to LASSO) and yields an $O(N \log\log N)$ rate. We significantly improve on this rate with an $O(k \log N)$ estimator, which is much more efficient in the sparse ($k \ll N$) regime.

Other principled model explanation approaches exist, based on influence functions (Koh & Liang, 2017; Koh et al., 2019; Feldman & Zhang, 2020), Representer Points (Yeh et al., 2018), or loss tracking (Pruthi et al., 2020; Hammoudeh & Lowd). They either focus on marginal influence on the whole dataset (Koh & Liang, 2017; Koh et al., 2019); make strong assumptions (e.g., convexity) that disallow their use with DNNs (Koh & Liang, 2017; Koh et al., 2019; Basu et al., 2020); cannot reason about data sources or sets of training samples (Pruthi et al., 2020; Hammoudeh & Lowd; Chakarov et al., 2016); or subsample training data but focus on a single inclusion probability, and thus cannot explain results in all scenarios (Feldman & Zhang, 2020).

Acknowledgments

Jinkun Lin is partially supported by NSF-1514422. Anqi Zhang is partially supported by a gift from Microsoft. We thank Daniel Hsu for insightful discussions in the early stages of the project, as well as the reviewers for their constructive comments.

References

Agrawal, A., Verschueren, R., Diamond, S., and Boyd, S. A rewriting system for convex optimization problems. *Journal of Control and Decision*, 5(1):42–60, 2018.

Aldridge, M., Johnson, O., and Scarlett, J. Group testing: An information theory perspective. *Foundations and Trends® in Communications and Information Theory*, 15(3-4):196–392, 2019. ISSN 1567-2190. doi: 10.1561/0100000099. URL http://dx.doi.org/10.1561/0100000099.

Angrist, J. D. and Pischke, J.-S. *Mostly harmless econometrics: An empiricist's companion*. Princeton university press, 2008.

Baek, W. wbaek/torchskeleton. URL https://github.com/wbaek/torchskeleton. Online; accessed 2021-02-07.

Balakumar, B. J. glmnet_python. URL https://github.com/bbalasub1/glmnet_python.

Baracaldo, N., Chen, B., Ludwig, H., and Safavi, J. A. Mitigating poisoning attacks on machine learning models: A data provenance based approach. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 103–110, 2017.

Barreno, M., Nelson, B., Joseph, A. D., and Tygar, J. D. The security of machine learning. *Machine Learning*, 81 (2):121–148, 2010.

Basu, S., Pope, P., and Feizi, S. Influence functions in deep learning are fragile. *arXiv preprint arXiv:2006.14651*, 2020.

Bondell, H. D. and Reich, B. J. Simultaneous factor selection and collapsing levels in anova. *Biometrics*, 65(1): 169–177, 2009.

Candes, E., Fan, Y., Janson, L., and Lv, J. Panning for gold: Model-x knockoffs for high-dimensional controlled variable selection. *arXiv preprint arXiv:1610.02351*, 2016.

Candès, E. J. et al. Compressive sampling. In *Proceedings of the international congress of mathematicians*, 2006.

Chakarov, A., Nori, A., Rajamani, S., Sen, S., and Vijaykeerthy, D. Debugging machine learning tasks. *arXiv preprint arXiv:1603.07292*, 2016.

Chen, B., Carvalho, W., Baracaldo, N., Ludwig, H., Edwards, B., Lee, T., Molloy, I., and Srivastava, B. Detecting backdoor attacks on deep neural networks by activation clustering. *arXiv preprint arXiv:1811.03728*, 2018.

Chen, X., Liu, C., Li, B., Lu, K., and Song, D. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

Chou, E., Tramèr, F., Pellegrino, G., and Boneh, D. Sentinet: Detecting physical attacks against deep learning systems. 2018.

Chu, X., Ilyas, I. F., and Papotti, P. Discovering denial constraints. *Proceedings of the VLDB Endowment*, 6(13): 1498–1509, 2013a.

Chu, X., Ilyas, I. F., and Papotti, P. Holistic data cleaning: Putting violations into context. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, pp. 458–469. IEEE, 2013b.

Cohen, G., Afshar, S., Tapson, J., and Van Schaik, A. Emnist: Extending mnist to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 2921–2926. IEEE, 2017.

Covert, I. and Lee, S.-I. Improving kernelshap: Practical shapley value estimation using linear regression. In *International Conference on Artificial Intelligence and Statistics*, pp. 3457–3465. PMLR, 2021.

Dasgupta, T., Pillai, N. S., and Rubin, D. B. Causal inference from 2 k factorial designs by using potential outcomes. *Journal of the Royal Statistical Society: Series B: Statistical Methodology*, pp. 727–753, 2015.

Diamond, S. and Boyd, S. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

Doan, B. G., Abbasnejad, E., and Ranasinghe, D. C. Februus: Input purification defense against trojan attacks on deep neural network systems. In *Annual Computer Security Applications Conference*, pp. 897–912, 2020.

Dolatshah, M., Teoh, M., Wang, J., and Pei, J. Cleaning crowdsourced labels using oracles for statistical classification. *Proc. VLDB Endow.*, 12(4):376–389, December 2018. ISSN 2150-8097. doi: 10.14778/3297753.3297758. URL https://doi.org/10.14778/3297753.3297758.

Egami, N. and Imai, K. Causal interaction in factorial experiments: Application to conjoint analysis. *Journal of the American Statistical Association*, 2018.

Feldman, V. and Zhang, C. What neural networks memorize and why: Discovering the long tail via influence estimation. *arXiv preprint arXiv:2008.03703*, 2020.

Frye, C., Rowat, C., and Feige, I. Asymmetric shapley values: incorporating causal knowledge into model-agnostic

explainability. *Advances in Neural Information Processing Systems*, 2020.

Gao, Y., Xu, C., Wang, D., Chen, S., Ranasinghe, D. C., and Nepal, S. Strip: A defence against trojan attacks on deep neural networks. In *Proceedings of the 35th Annual Computer Security Applications Conference*, pp. 113–125, 2019.

Ghorbani, A. and Zou, J. Data shapley: Equitable valuation of data for machine learning. In *International Conference on Machine Learning*, pp. 2242–2251. PMLR, 2019.

Ghorbani, A., Kim, M., and Zou, J. A distributional framework for data valuation. In *International Conference on Machine Learning*, pp. 3535–3544. PMLR, 2020.

Hainmueller, J., Hopkins, D. J., and Yamamoto, T. Causal inference in conjoint analysis: Understanding multidimensional choices via stated preference experiments. *Political analysis*, 22(1):1–30, 2014.

Hammoudeh, Z. and Lowd, D. Simple, attack-agnostic defense against targeted training set attacks using cosine similarity.

Harris, C., Pymar, R., and Rowat, C. Joint shapley values: a measure of joint feature importance. In *International Conference on Learning Representations*, 2022.

Hastie, T., Qian, J., and Tay, K. An introduction to glmnet, 2016.

Hayase, J., Kong, W., Somani, R., and Oh, S. Spectre: Defending against backdoor attacks using robust statistics. *arXiv preprint arXiv:2104.11315*, 2021.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hellerstein, J. M. Quantitative data cleaning for large databases. *United Nations Economic Commission for Europe (UNECE)*, 25, 2008.

Ilyas, A., Park, S. M., Engstrom, L., Leclerc, G., and Madry, A. Datamodels: Predicting predictions from training data. *arXiv preprint arXiv:2202.00622*, 2022.

Imbens, G. W. and Rubin, D. B. *Causal inference in statistics, social, and biomedical sciences*. Cambridge University Press, 2015.

Jagielski, M., Oprea, A., Biggio, B., Liu, C., Nita-Rotaru, C., and Li, B. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *2018 IEEE Symposium on Security and Privacy (SP)*, pp. 19–35. IEEE, 2018.

Jethani, N., Sudarshan, M., Covert, I., Lee, S.-I., and Ranganath, R. Fastshap: Real-time shapley value estimation. *arXiv e-prints*, pp. arXiv–2107, 2021.

Jia, R. Group testing implementation, October 2021. URL https://github.com/sunblaze-ucb/data-valuation/blob/8f101e0884544ebf94e741f54fc2a6833ffb2f90/group_testing/group_testing_example.py.

Jia, R., Dao, D., Wang, B., Hubis, F. A., Hynes, N., Gürel, N. M., Li, B., Zhang, C., Song, D., and Spanos, C. J. Towards efficient data valuation based on the shapley value. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 1167–1176. PMLR, 2019.

Kang, J. D., Schafer, J. L., et al. Demystifying double robustness: A comparison of alternative strategies for estimating a population mean from incomplete data. *Statistical science*, 22(4):523–539, 2007.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Koh, P. W. and Liang, P. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, pp. 1885–1894. PMLR, 2017.

Koh, P. W., Steinhardt, J., and Liang, P. Stronger data poisoning attacks break data sanitization defenses. *arXiv preprint arXiv:1811.00741*, 2018.

Koh, P. W., Ang, K.-S., Teo, H. H., and Liang, P. On the accuracy of influence functions for measuring group effects. *arXiv preprint arXiv:1905.13289*, 2019.

Krishnan, S., Franklin, M. J., Goldberg, K., and Wu, E. Boostclean: Automated error detection and repair for machine learning. *arXiv preprint arXiv:1711.01299*, 2017.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Kuangliu. kuangliu/pytorch-cifar. URL https://github.com/kuangliu/pytorch-cifar/tree/ab908327d44bf9b1d22cd333a4466e85083d3f21. Online; accessed 2021-02-07.

Kwon, Y. and Zou, J. Beta shapley: a unified and noise-reduced data valuation framework for machine learning. In *International Conference on Artificial Intelligence and Statistics*, 2022.

Lecué, G. and Mendelson, S. Regularization and the small-ball method ii: complexity dependent error rates. *The Journal of Machine Learning Research*, 18(1):5356–5403, 2017.

Lecué, G., Mendelson, S., et al. Regularization and the small-ball method i: sparse recovery. *The Annals of Statistics*, 46(2):611–641, 2018.

Lecuyer, M., Spahn, R., Spiliopolous, Y., Chaintreau, A., Geambasu, R., and Hsu, D. Sunlight: Fine-grained targeting detection at scale with statistical confidence. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, 2015.

Lipovetsky, S. and Conklin, M. Analysis of regression in game theory approach. *Applied Stochastic Models in Business and Industry*, 2001.

Lundberg, S. M. and Lee, S.-I. A unified approach to interpreting model predictions. In *Proceedings of the 31st international conference on neural information processing systems*, pp. 4768–4777, 2017.

Maleki, S., Tran-Thanh, L., Hines, G., Rahwan, T., and Rogers, A. Bounding the estimation error of sampling-based shapley value approximation, 2014.

Maletic, J. I. and Marcus, A. Data cleansing: Beyond integrity analysis. In *Iq*, pp. 200–209. Citeseer, 2000.

Mitchell, R., Cooper, J., Frank, E., and Holmes, G. Sampling permutations for shapley value estimation. 2022.

Ni, J., Li, J., and McAuley, J. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 188–197, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1018. URL https://aclanthology.org/D19-1018.

Pauwels, E. Lecture notes: Statistics, optimization and algorithms in high dimension, 2020.

Peleg, B. and Sudhölter, P. *Introduction to the theory of cooperative games*, volume 34. Springer Science & Business Media, 2007.

Peri, N., Gupta, N., Huang, W. R., Fowl, L., Zhu, C., Feizi, S., Goldstein, T., and Dickerson, J. P. Deep k-nn defense against clean-label data poisoning attacks. In *European Conference on Computer Vision*, pp. 55–70. Springer, 2020.

Post, J. B. and Bondell, H. D. Factor Selection and Structural Identification in the Interaction ANOVA Model. *Biometrics*, 69(1):70–79, 2013. ISSN 1541-0420. doi: https://doi.org/10.1111/j.1541-0420.2012.01810.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1541-0420.2012.01810.x. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1541-0420.2012.01810.x.

Pruthi, G., Liu, F., Kale, S., and Sundararajan, M. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 33, 2020.

PyTorch. pytorch/torchvision/resnet, a. URL https://github.com/pytorch/vision/blob/master/torchvision/models/resnet.py. Online; accessed 2021-03-30.

PyTorch. pytorch/examples, b. URL https://github.com/pytorch/examples/blob/0d3fe14a1c5a00795e3671ea3473caef6f0da72d/mnist/main.py. Online; accessed 2021-02-07.

Rauhut, H. Compressive sensing and structured random matrices. *Theoretical foundations and numerical methods for sparse recovery*, 9(1):92, 2010.

Rivasplata, O. Subgaussian random variables: An expository note. pp. 11.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3): 211–252, 2015.

Shafahi, A., Huang, W. R., Najibi, M., Suciu, O., Studer, C., Dumitras, T., and Goldstein, T. Poison frogs! targeted clean-label poisoning attacks on neural networks. *arXiv preprint arXiv:1804.00792*, 2018.

Shapley, L. S. A value for n-person games. *Contributions to the Theory of Games*, 2(28):307–317, 1953.

Shen, S., Tople, S., and Saxena, P. Auror: Defending against poisoning attacks in collaborative deep learning systems. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pp. 508–519, 2016.

Tang, D., Wang, X., Tang, H., and Zhang, K. Demon in the variant: Statistical analysis of dnns for robust backdoor contamination detection. In *30th {USENIX} Security Symposium ({USENIX} Security 21)*, 2021.

Tran, B., Li, J., and Madry, A. Spectral signatures in backdoor attacks. *arXiv preprint arXiv:1811.00636*, 2018.

Trevett, B. bentrevett/pytorch-sentiment-analysis/. URL https://github.com/bentrevett/pytorch-sentiment-analysis. Online; accessed 2021-03-10.

Udeshi, S., Peng, S., Woo, G., Loh, L., Rawshan, L., and Chattopadhyay, S. Model agnostic defence against backdoor attacks in machine learning. *arXiv preprint arXiv:1908.02203*, 2019.

Veldanda, A. K., Liu, K., Tan, B., Krishnamurthy, P., Khorrami, F., Karri, R., Dolan-Gavitt, B., and Garg, S. Nnoculation: Broad spectrum and targeted treatment of backdoored dnns. *arXiv preprint arXiv:2002.08313*, 2020.

Wang, B., Yao, Y., Shan, S., Li, H., Viswanath, B., Zheng, H., and Zhao, B. Y. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 707–723. IEEE, 2019.

WikipediaContributors. Group testing, January 2021. URL https://en.wikipedia.org/w/index.php?title=Group_testing&oldid=997779128. Online; accessed 2021-02-19.

Williamson, B. and Feng, J. Efficient nonparametric statistical inference on population feature importance using shapley values. In *International Conference on Machine Learning*, 2020.

Wu, W., Flokas, L., Wu, E., and Wang, J. Complaint-driven training data debugging for query 2.0. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pp. 1317–1334, 2020.

Yeh, C.-K., Kim, J. S., Yen, I. E., and Ravikumar, P. Representer point selection for explaining deep neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 9311–9321, Red Hook, NY, USA, 2018. Curran Associates Inc.

Zhang, C., Ippolito, D., Lee, K., Jagielski, M., Tramèr, F., and Carlini, N. Counterfactual memorization in neural language models. *arXiv preprint arXiv:2112.12938*, 2021.

A Convergence Rate when Using LASSO to compute AME

We first state and prove a variant of the LASSO error bound result from (Lecué et al., 2018) in a simpler setting, which is sufficient for our application and will serve as the foundation of many of our results. We then apply this result to finish the proof of $O(k \log N)$ sample rate of our AME estimator left in Prop. 3.3 in the main body.

A.1 Simplified LASSO Error Bound

Proposition A.1. *Consider a regression problem where one wants to approximate an unknown random variable Y using a set of random variables $X_1, \ldots, X_N$ with $N \geq 3$. Let $\hat{\beta}_{lasso}$ be the LASSO coefficient when regressing Y on X with L_1 regularization, and β^* be the best linear fit (detailed definition in Prop. 3.2). Assume that $\forall n \in [N] : \beta^*_n$ is finite, X_n is bounded in $[A, B]$, Y is bounded in $[0, 1]$. Further assume that $\forall i, j \in [N] : \mathbb{E}[X_i X_j] = 0$ when $i = j$ and otherwise 1. If β^* has at most k non-zeros and the sample size $M \geq k(1 + \log(k/N))$, then there exists a regularization parameter λ and a value $C(B - A, \delta)$ such that with probability at least $1 - \delta$:*

$$\|\hat{\beta}_{lasso} - \beta^*\|_2 \leq C(B - A, \delta)\sqrt{\frac{k \log(N)}{M}}.$$

The following terminology will be useful to facilitate its proof.

Definition A.2. X is said to be a subgaussian random variable with variance property σ^2 if for any $s \in \mathbb{R}, \mathbb{E}[\exp(sX)] \leq \exp\left(\frac{\sigma^2 s^2}{2}\right)$. We write $X \sim subG(\sigma^2)$.

Definition A.3. The underlying measure of a random vector $X \in \mathbb{R}^N$ is said to be L-subgaussian if for every $q \geq 2$ and every $u \in \mathbb{R}^N$,

$$\mathbb{E}^{1/q}[|\langle u, X\rangle|^q] \leq L\sqrt{q}\mathbb{E}^{1/2}[|\langle u, X\rangle|^2].$$

We now prove proposition A.1:

Proof. To apply Theorem 1.4 from (Lecué et al., 2018) to bound the error, we need our setup to satisfy Assumption 1.1 of that paper: that X is an isotropic, L-subgaussian measure, and that the noise is in L_q for $q > 2$.

First, the isotropic requirement is that $\mathbb{E}[\langle u, X\rangle^2] = \langle u, u\rangle$ for all $u \in \mathbb{R}^N$. This can be shown by observing that $\mathbb{E}[\langle u, X\rangle^2] = \sum_{i=1}^N \mathbb{E}[u_i^2 X_i^2] + \sum_{i \neq j} \mathbb{E}[u_i u_j X_i X_j] = \sum_{i=1}^N u_i^2$, where the last equality comes from the fact that $\mathbb{E}[X_i X_j] = 0$ and $\mathbb{E}[X_i^2] = 1$.

The second requirement is that the probability measure of the covariate vector X is L-subgaussian (Def. A.3). Let $\sigma = (B - A)/2$. Then $X_n \sim subG(\sigma^2)$ since they are bounded. Hence, $\langle u, X\rangle \sim subG(\sigma^2)$ (see e.g., Theorem 2.1.2 in (Pauwels, 2020)). Applying Proposition 3.2 from (Rivasplata), we have $\mathbb{E}^{1/q}[|\langle u, X\rangle|^q] \leq L\sqrt{q}$ with $L > 0$ a constant dependent only on σ. Noticing that Def. A.3 remains equivalent when constraining $\|u\|_2 = 1$, and that $\mathbb{E}^{1/2}[\langle u, X\rangle^2] = \|u\|_2 = 1$ concludes this part of proof.

Third the "noise", defined as $\xi = \langle \beta^*, X\rangle - Y$ should be in L_q, $q > 2$. Notice that the best estimator cannot be worse than a zero estimator that always return 0, thus $\mathbb{E}[\xi^2] \leq 1$. Hence, $|\xi| \leq |\langle \beta^*, X\rangle| + |Y| \leq DN + 1$, where D is the upper bound of $|\beta^*_n X_n|$, then there exists $q = 1/\log(DN + 1) + 2 > 2$ such that $\mathbb{E}[|\xi|^q] \leq \mathbb{E}[|\xi|^2](DN + 1)^{q-2} \leq \mathbb{E}[|\xi|^2] \cdot e \leq e$.

Combined with the sparsity assumption on β^*, we can apply Theorem 1.4 from (Lecué et al., 2018), which directly yields the error bound.

□

A.2 Proof of Prop. 3.3

With this simplified bound, we can prove Prop. 3.3:

Proof. Recall that the best linear estimator $\beta^*_n = \text{AME}_n/\sqrt{v}$, so applying Prop. A.1 directly yields the bound. The remaining work is to verify the assumptions: Since the AME is an average of utility differences, we know that $\beta^*_n = \text{AME}_n/\sqrt{v} \in [-1/\sqrt{v}, 1/\sqrt{v}]$ is finite. Furthermore, by design $\mathbb{E}[X_i X_j] = \mathbb{E}_p[\mathbb{E}[X_i X_j|p]] = \mathbb{E}_p[\mathbb{E}[X_i|p]\mathbb{E}[X_j|p]] = 0$ when $i \neq j$, and $\mathbb{E}[X_i X_j] = \mathbb{E}[X_i^2] = Var[\mathbb{E}[X_i^2]] + \mathbb{E}[X_i]^2 = 1$ when $i = j$. With the assumptions all verified, applying Prop. A.1 concludes proof. □

B An AME Estimator with p-featurization

Recall that to estimate $\text{AME}_n(\mathcal{P})$ defined under some given distribution $\mathcal{P}$, we have been using a featurization where $\text{X}[m,n]$ takes values of either $\frac{1}{\sqrt{v}p}$ or $\frac{-1}{\sqrt{v}(1-p)}$, depending on whether data point/source n is respectively included or excluded in the subset for row m, with $v = \mathbb{E}_{p\sim\mathcal{P}}[\frac{1}{p(1-p)}]$. However, values for $\text{X}[m,n]$ blow up quickly as p approaches 0 or 1, which leads to unbounded feature values for certain $\mathcal{P}$ that samples such p values often. Unfortunately such distributions can be useful in some cases, in particular to derive a low-error SV approximations (e.g., with the beta$(1+\varepsilon, 1+\varepsilon)$ with small ε (§C.3)). Below we propose a different featurization that solves this issue while still ensuring an $O(k \log N)$ sample rate.

Specifically, we define $X_n = \sqrt{v}(1-p)$ if $n \in S$ and $X_n = -\sqrt{v}p$ otherwise. These values are clearly bounded when $\sqrt{v}$ is finite. To ensure that the best linear fit still recovers AME (a.k.a. Prop. 3.2)—an important property we use to derive the $O(k \log N)$ error bound—we adapt the distribution where samples (X, Y) are drawn in LASSO. Recall that source inclusion is a compound distribution, in which we first draw $p \sim \mathcal{P}$ for the entire subset, and then X_n's according to p. Here, we change they way p is drawn, by imposing a $\frac{1}{p(1-p)}$-weighting over $\mathcal{P}$. Formally, if we note $f_{\mathcal{P}}(\cdot)$ the probability density function (PDF) of $\mathcal{P}$ which we used with the original $1/p$ featurization, we now draw p from the distribution with reweighted PDF $f_{\mathcal{W}}(p) = f_{\mathcal{P}}(p)\frac{1}{p(1-p)} / \mathbb{E}_{p\sim\mathcal{P}}[\frac{1}{p(1-p)}]$.

Denote this new sampling scheme by $p \sim \mathcal{W}$, and note that the AME is still defined under the original distribution $\mathcal{P}$, while $\mathcal{W}$ is only used to sample (X, Y) for LASSO. We show that the best linear fit on (X, Y) is still $\text{AME}_n(\mathcal{P})$:

Proposition B.1. *Let β^* be the best linear fit on (X, Y):*

$$\beta^* = \underset{\beta\in\mathbb{R}^n}{\arg\min}\, \mathbb{E}\left[(Y - \langle\beta, X\rangle)^2\right], \tag{4}$$

then $\text{AME}_n(\mathcal{P})/\sqrt{v} = \beta^*_n,\ \forall n \in [N]$, with $v = \mathbb{E}_{p\sim\mathcal{P}}[\frac{1}{p(1-p)}]$.

Proof. For a linear regression, we have (e.g. Eq. 3.1.3 of (Angrist & Pischke, 2008)):

$$\beta^*_n = \frac{Cov(Y, \tilde{X}_n)}{Var[\tilde{X}_n]},$$

where $\tilde{X}_n$ is the regression residual of X_n on all other covariates $X_{-n} = (X_1, \ldots, X_{n-1}, X_{n+1}, \ldots, X_N)$. In our design $\mathbb{E}[X_n|X_{-n}] = \mathbb{E}_{p\sim\mathcal{P}}[X_n|p] = 0$, implying $\tilde{X}_n = X_n - \mathbb{E}[X_n|X_{-n}] = X_n$. Therefore:

$$\beta^*_n = \frac{Cov(Y, \tilde{X}_n)}{Var[\tilde{X}_n]} = \frac{Cov(Y, X_n)}{Var[X_n]} = \frac{\mathbb{E}[X_n Y]}{Var[X_n]}.$$

Notice that $\mathbb{E}[X_n Y] = \mathbb{E}_{p\sim\mathcal{W}}[\mathbb{E}[X_n Y|p]]$ with:

$$\begin{aligned}
\mathbb{E}[X_n Y|p] &= p \cdot \mathbb{E}[X_n Y|p, n \in S] + (1-p) \cdot \mathbb{E}[X_n Y|p, n \notin S] \\
&= \sqrt{v}p(1-p)\mathbb{E}[Y|p, n \in S] + \sqrt{v}(1-p)(-p)\mathbb{E}[Y|p, n \notin S] \\
&= \sqrt{v}p(1-p)(\mathbb{E}[Y|p, n \in S] - \mathbb{E}[Y|p, n \notin S])
\end{aligned}$$

Combining the two previous steps yields:

$$\begin{aligned}
\beta^*_n &= \frac{\mathbb{E}_{p\sim\mathcal{W}}[\sqrt{v}p(1-p)(\mathbb{E}[Y|p, n \in S] - \mathbb{E}[Y|p, n \notin S])]}{Var[X_n]} \\
&= \frac{\sqrt{v}\mathbb{E}_{p\sim\mathcal{P}}[\mathbb{E}[Y|p, n \in S] - \mathbb{E}[Y|p, n \notin S]]}{v \cdot Var[X_n]} \\
&= \frac{AME_n}{\sqrt{v} \cdot Var[X_n]}.
\end{aligned}$$

Noticing that

$$Var[X_n] = \mathbb{E}[Var[X_n|p]] + Var[\mathbb{E}[X_n|p]] = \int_0^1 f_{\mathcal{W}}(p)(pv(1-p)^2 + (1-p)v(-p)^2)dp$$
$$= v\int_{\varepsilon}^{1-\varepsilon} \frac{1}{v(1-2\varepsilon)}dp = 1$$

concludes the proof. □

Moreover, $\mathbb{E}[X_iX_j] = \mathbf{1}\{i=j\}$ for the same reason as in §A.2. Hence, we can still apply Prop. A.1 to derive the same LASSO error bound:

Proposition B.2. *If X_n's are bounded in $[A,B]$ and $N \geq 3$, there exist a regularization parameter λ and a constant $C(B-A,\delta)$ such that when the sample size $M \geq k(1+log(k/N))$, with probability at least $1-\delta$:*

$$\|\hat{\beta}_{lasso} - \frac{1}{\sqrt{v}}\textit{AME}\|_2 \leq C(B-A,\delta)\sqrt{\frac{k\log(N)}{M}},$$

with $v = \mathbb{E}_{p\sim\mathcal{P}}[\frac{1}{p(1-p)}]$ and $\textit{AME}_n$ defined under $\mathcal{P}$.

Compared with the original, $1/p$-featurization, the difference only lies in the constant factor $C(B-A,\delta)$, since $B-A$ has changed.

C Sparse Estimators for the Shapley Value from the AME

Recall that the AME is the SV when $\mathcal{P} = Uni(0,1)$. However, this choice is incompatible with our fast convergence rates for LASSO. To find a good estimator of the SV from the AME, it is thus crucial to understand the discrepancy between the AME and the SV introduced by different distributions $\mathcal{P}$ over p, in order to bound the SV from the AME with a cP compatible with good convergence rates. Here we first derive general error bounds between SV and AME that work for all distributions. Then we apply it to two specific distributions: namely the truncated uniform and Beta distributions. We mainly focus on sparse SV and/or AME under bounded or monotone utility, but to make the discussion clearer, no assumptions are made on either the SV or AME unless explicitly stated.

Throughout this section, we denote by $P_{\text{AME}}(S)$ and $P_{\text{SV}}(S)$ the probability of sampling subset S when $p \sim \mathcal{P}$ and $p \sim Uni(0,1)$, respectively (the P_{SV} name is due to the fact that AME is SV under this distribution, see Prop. 2.1). We also introduce the following notation:

$$\Delta \triangleq \max_S \frac{P_{\text{AME}}(S)}{P_{\text{SV}}(S)} - 1. \tag{5}$$

C.1 General bounds

Lemma C.1. *Assume a bounded utility function with range in $[0,1]$. Then*

$$\|\textit{AME} - \textit{SV}\|_\infty \leq 2\Delta.$$

Further assume a monotone utility (Assumption 3.5). Then:

$$\|AME - \textit{SV}\|_2 \leq \Delta + \sqrt{2\Delta}.$$

Proof. The L_∞ error bound is due to the following:

$$\begin{aligned} |\text{AME}_n - \text{SV}_n| &= \Big|\sum_S \big(P_{\text{AME}}(S) - P_{\text{SV}}(S)\big)\big(U(S\cup\{n\}) - U(S)\big)\Big| \\ &\leq \sum_S |P_{\text{AME}}(S) - P_{\text{SV}}(S)| = 2\sum_{S:P_{\text{AME}}(S)>P_{\text{SV}}(S)} P_{\text{AME}}(S) - P_{\text{SV}}(S) \leq 2\Delta. \end{aligned} \tag{6}$$

For the L_2 error, its square $\|\text{AME} - \text{SV}\|_2^2$ can be divided into two groups based on the sign of $\text{AME}_n - \text{SV}_n$. Call those indices n with positive (negative) sign $\mathcal{N}^+$ ($\mathcal{N}^-$). For all $n \in \mathcal{N}^+$,

$$\text{AME}_n - \text{SV}_n = \sum_S \big(P_{\text{AME}}(S) - P_{\text{SV}}(S)\big)\big(U(S+\{n\}) - U(S)\big) \leq \sum_S \Delta P_{\text{SV}}(S)(U(S+\{n\}) - U(S)) = \Delta\text{SV}_n, \tag{7}$$

where the last inequality is due to $U(S+\{n\}) > U(S)$ implied by monotonicity. For the same reason, all SV_n's are positive, implying $\text{SV}_n \leq U([N]) \leq 1$. Thus $(\text{AME}_n - \text{SV}_n)^2 \leq \Delta^2 \text{SV}_n$. On the other hand, for all $n \in \mathcal{N}^-$, we know that $|\text{AME}_n - \text{SV}_n|$ is bounded by 2Δ; it is also bounded by SV_n since AME_n cannot be negative under monotone utility. Summing up these bounds gives $\|\text{AME} - \text{SV}\|_2 \leq \sqrt{\sum_{n\in\mathcal{N}^+} \Delta^2 \text{SV}_n + \sum_{n\in\mathcal{N}^-} 2\Delta \text{SV}_n} \leq \sqrt{\sum_{n\in\mathcal{N}^+} \Delta^2 \text{SV}_n} + \sqrt{\sum_{n\in\mathcal{N}^-} 2\Delta \text{SV}_n} \leq \Delta + \sqrt{2\Delta}$, where the last inequality is due to $\|\text{SV}\|_1 \leq 1$. □

In fact the bounded utility assumption is quite minor when we assume monotonicity: every $U(S)$ is bounded between the empty set and full set utility. Given that by definition of SV $U(\emptyset) = 0$, it reduces to an assumption of $U([N]) \leq 1$. In practice as long as $U([N])$ is a known and bounded constant (e.g., the accuracy on the validation set of the model trained on the full set), one can simply scale the utility to meet this requirement. In what follows, when we say monotone utility we mean both monotone and bounded.

C.2 SV Estimator from the AME under a Truncated Uniform distribution

We prove the results from the paper's main body, before discussing p-featurization.

Proof of Lemma 3.6. As a reminder, Lemma 3.6 states an L_2 error bound between the AME and SV under monotone utility, when AME is defined with $\mathcal{P} = Uni(\varepsilon, 1-\varepsilon)$.

Proof. Lemma C.1 already gives us $\|\text{AME} - \text{SV}\|_2 \leq \Delta + \sqrt{2\Delta}$ (Δ from Eq. 5). All that remains is to show that $\Delta \leq 4\varepsilon$:

$$\Delta = \max_S \frac{P_{\text{AME}}(S)}{P_{\text{SV}}(S)} - 1 \tag{8a}$$

$$(j \leftarrow |S|) = \max_j \left(N \int_\varepsilon^{1-\varepsilon} \frac{1}{1-2\varepsilon} \binom{N-1}{j} p^j (1-p)^{N-j-1} dp - 1 \right) \tag{8b}$$

$$\leq \max_j \left(N \cdot \frac{1}{1-2\varepsilon} \int_0^1 \binom{N-1}{j} p^j (1-p)^{N-j-1} dp - 1 \right) \tag{8c}$$

$$= \max_j \left(N \cdot \frac{1}{1-2\varepsilon} \cdot \frac{1}{N} - 1 \right) = \frac{1}{1-2\varepsilon} - 1 \tag{8d}$$

$$\leq 4\varepsilon, \tag{8e}$$

where the equality 8d is due to the fact that the Binomial(p) with $p \sim Uni(0,1)$ is a discrete uniform. Hence, $\|\text{AME} - \text{SV}\|_2 \leq \Delta + \sqrt{2\Delta} \leq 4\varepsilon + 2\sqrt{2\varepsilon}$. □

Lemma C.1 also yields to the following result, which fills in the missing piece in the proof of Corollary 3.7—the one that states the $O(k \log N)$ bound for this SV estimator:

Corollary C.2. *A k-sparse SV implies a k-sparse AME under monotone utility.*

Proof. Under montone utility, both AME_n and SV_n are non-negative. By Eq. 7, when $\text{SV}_n = 0$, $\text{AME}_n \leq \Delta \text{SV}_n = 0$. □

Non-monotone utility. When the utility is no longer monotone, we can still derive an $O(k \log N)$ rate in terms of L_∞ error for SV estimation, under an additional assumption that the AME is k-sparse. Indeed, first notice that the bound $\Delta \leq 4\varepsilon$ from Eq. 8a does not require monotonicity. Under utility bounded in $[0,1]$, applying the first part of Lemma C.1 yields:

$$\|\text{AME} - \text{SV}\|_\infty \leq 2\Delta \leq 8\varepsilon. \tag{9}$$

Notice that this bound, as the L_2 bound, does not depend on N or k. Hence, applying the same arguments as in Corollary 3.7 yields an L_∞ error bound we presented in the main body (Corollary 3.4). We reiterate it here for convenience of reading:

Corollary C.3. *When AME is k-sparse and the utility is bounded in $[0,1]$, for every constant $\varepsilon > 0, \delta > 0, N \geq 3$, there exists constants $C_1(\varepsilon,\delta)$, ε', and a LASSO regularization parameter λ, such that when the number of samples $M \geq C_1(\varepsilon,\delta) k \log N$, $\|\sqrt{v}\hat{\beta}_{lasso} - SV\|_\infty \leq \varepsilon$ holds with a probability at least $1-\delta$, where $v = \mathbb{E}_{p\sim Uni(\varepsilon',1-\varepsilon')}[\frac{1}{p(1-p)}]$.*

Proof. $\|\sqrt{v}\hat{\beta}_{lasso} - \text{SV}\|_\infty \leq \|\sqrt{v}\hat{\beta}_{lasso} - \text{AME}\|_2 + \|\text{AME} - \text{SV}\|_\infty$. By the sparsity of AME, we apply Proposition 3.3 to bound the first term by $\varepsilon/2$, and Eq. 9 with $\varepsilon' = \varepsilon/4$ to bound the second term by the same, concluding the proof. □

The main obstacle to deriving an L_2 bound in this more general setting comes from the lack of an L_2 error bound between the AME and the SV that is independent of N. Note that one may derive an L_2 error bound from Eq. 9 as follows: $\|\text{AME} - \text{SV}\|_2 \leq \sqrt{N}\|\text{AME} - \text{SV}\|_\infty \leq 4\sqrt{N}\varepsilon$. However, this bound is now dependent on N, which violates the precondition of applying the LASSO error bound (see Prop. 3.3).

Using p-featurization. As pointed out in §B, p-featurization also achieves a $O(k \log N)$ sample rate to reach a low L_2 error in estimating the AME. Since p-featurization has no effect on the AME value, the bound between the AME and the SV remains the same. We thus reach the same conclusion as for $1/p$-featurization:

Corollary C.4. *For every constant $\varepsilon > 0, \delta > 0, N \geq 3$, there exists constants $C_2(\varepsilon, \delta)$, ε', and a LASSO regularization parameter λ, such that with $M \geq C_2(\varepsilon, \delta) k \log N$, and with probability at least $1 - \delta$:*

(1) $\|\sqrt{v}\hat{\beta}_{lasso} - SV\|_\infty \leq \varepsilon$ holds when the utility is bounded in $[0, 1]$ and the AME is k-sparse;
(2) $\|\sqrt{v}\hat{\beta}_{lasso} - SV\|_2 \leq \varepsilon$ holds when the utility is monotone and the SV is k-sparse,

where the AME is defined under $\mathcal{P} = Uni(\varepsilon', 1 - \varepsilon')$ and $v = \mathbb{E}_{p\sim\mathcal{P}}[\frac{1}{p(1-p)}]$.

Proof. Conclusion (1) follows the same proof as Corollary C.3, and Conclusion (2) follows Corollary 3.7. □

C.3 SV Estimator from the AME under a Beta Distribution

Another candidate to estimate the SV from the AME is to use $\mathcal{P} = \text{Beta}(1 + \varepsilon, 1 + \varepsilon)$ as the distribution of p, with $\varepsilon \in (0, 0.5)$, and using p-featurization[1]. We show an $O(k \log N)$ sample rate in this setting as well, after introducing two necessary lemmas.

Lemma C.5. *For any $x \geq 1, y \geq 1$ and $\varepsilon \in (0, 0.5)$, the following holds:*

$$\frac{(x + \varepsilon - 1)^\varepsilon (y + \varepsilon - 1)^\varepsilon}{(x + y + 2\varepsilon)^{2\varepsilon}} < \frac{\mathrm{B}(x + \varepsilon, y + \varepsilon)}{\mathrm{B}(x, y)} < \frac{(x + \varepsilon)^\varepsilon (y + \varepsilon)^\varepsilon}{(x + y + 2\varepsilon - 1)^{2\varepsilon}},$$

where $\mathrm{B}(\cdot, \cdot)$ is the Beta function.

Proof. According to Gautschi's inequality, for all $a > 0, s \in (0, 1)$,

$$a^{1-s} < \frac{\Gamma(a + 1)}{\Gamma(a + s)} < (a + 1)^{1-s},$$

where $\Gamma(\cdot)$ is the Gamma function. For all $\varepsilon' \in (0, 1)$ and $b \geq 1$, we can change variables with $s \leftarrow 1 - \varepsilon' \in (0, 1)$ and $a \leftarrow b + \varepsilon' - 1 > 0$ to obtain:

$$(b + \varepsilon' - 1)^{\varepsilon'} > \frac{\Gamma(b + \varepsilon')}{\Gamma(b)} > (b + \varepsilon')^{\varepsilon'}. \tag{10}$$

In addition, we have that:

$$\mathrm{B}(x + \varepsilon, y + \varepsilon)/\mathrm{B}(x, y) = \frac{\Gamma(x + \varepsilon)}{\Gamma(x)} \frac{\Gamma(y + \varepsilon)}{\Gamma(y)} / \frac{\Gamma(x + y + 2\varepsilon)}{\Gamma(x + y)}$$

Plugging Eq. 10 into the above concludes the proof. □

Lemma C.6. *When $p \sim \text{Beta}(1 + \varepsilon, 1 + \varepsilon)$ with $\varepsilon < 0.5$, then if the utility is bounded in $[0, 1]$,*

$$\|AME - SV\|_\infty \leq 2((1 + \frac{1}{\varepsilon})^{2\varepsilon} - 1).$$

In addition, under a monotone utility,

$$\|AME - SV\|_2 \leq ((1 + \frac{1}{\varepsilon})^{2\varepsilon} - 1) + \sqrt{2((1 + \frac{1}{\varepsilon})^{2\varepsilon} - 1)}.$$

[1]The $1/p$-featurization in incompatible here, since this distribution can draw "p"s arbitrarily close to 0 or 1 which leads to unbounded feature values, violating the assumption of the $O(k \log N)$ LASSO rate (Prop. 3.3).

Proof. With a small abuse of notation, we write $P_{\text{AME}}(j) = \sum_{S:|S|=j} P_{\text{AME}}(S)$ and $P_{\text{SV}}(j) = \sum_{S:|S|=j} P_{\text{SV}}(S)$ (see §C for detailed definition of $P_{\text{AME}}(S)$ and $P_{\text{SV}}(S)$). Notice that now $P_{\text{AME}}(j)$ is the PMF of a beta binomial distribution $\text{BB}(\alpha = 1+\varepsilon, \beta = 1+\varepsilon, n = N-1)$. Let $\text{B}(\cdot,\cdot)$ denote the Beta function, then:

$$\begin{aligned}
\Delta + 1 = \max_j P_{\text{AME}}(j)/P_{\text{SV}}(j) &= \max_j N\binom{N-1}{j}\frac{\text{B}(j+\alpha, N-1-j+\beta)}{\text{B}(\alpha,\beta)} \\
&= \max_j N \cdot \frac{1}{N\text{B}(N-j, j+1)} \cdot \frac{\text{B}(j+\alpha, N-1-j+\beta)}{\text{B}(\alpha,\beta)} \\
(\text{since } \text{B}(1,1)=1) &= \max_j \frac{\text{B}(j+1+\varepsilon, N-j+\varepsilon)}{\text{B}(j+1, N-j)} \cdot \frac{\text{B}(1,1)}{\text{B}(1+\varepsilon, 1+\varepsilon)} \\
(\text{by lemma C.5}) &\le \max_j \frac{(j+1+\varepsilon)^\varepsilon (N-j+\varepsilon)^\varepsilon}{(N+1+2\varepsilon)^{2\varepsilon}} \Big/ \frac{\varepsilon^{2\varepsilon}}{(2+2\varepsilon)^{2\varepsilon}} \\
(\text{maximum reached when } j+1+\varepsilon = N-j+\varepsilon) &\le \frac{1}{4^\varepsilon}\frac{(2+2\varepsilon)^{2\varepsilon}}{\varepsilon^{2\varepsilon}} \\
&\le (1+\frac{1}{\varepsilon})^{2\varepsilon},
\end{aligned} \tag{11}$$

Applying Lemma C.1 concludes the proof. □

Now we formally state the $O(k \log N)$ sample rate:

Corollary C.7. *For every $\varepsilon > 0, \delta > 0, N \ge 3$, there exists constants $C_3(\varepsilon,\delta)$ and ε', and a regularization parameter λ, such that when the number of samples $M \ge C_3(\varepsilon,\delta) k \log N$, with a probability at least $1-\delta$,*

(1) $\|\sqrt{v}\hat{\beta}_{lasso} - SV\|_\infty \le \varepsilon$ holds when the utility is bounded in $[0,1]$ and AME is k-sparse;
(2) $\|\sqrt{v}\hat{\beta}_{lasso} - SV\|_2 \le \varepsilon$ holds when the utility is monotone and SV is k-sparse,

where AME is defined by $\mathcal{P} = \text{Beta}(1+\varepsilon', 1+\varepsilon')$ and $v = \mathbb{E}_{p\sim\mathcal{P}}[\frac{1}{p(1-p)}]$.

Proof. Observing that $\|\sqrt{v}\hat{\beta}_{lasso} - \text{SV}\|_\infty \le \|\sqrt{v}\hat{\beta}_{lasso} - \text{AME}\|_2 + \|\sqrt{v}\text{AME} - \text{SV}\|_\infty$ and $\|\sqrt{v}\hat{\beta}_{lasso} - \text{SV}\|_2 \le \|\sqrt{v}\hat{\beta}_{lasso} - \text{AME}\|_2 + \|\sqrt{v}\text{AME} - \text{SV}\|_2$, we proceed by bounding both $\|\sqrt{v}\hat{\beta}_{lasso} - \text{AME}\|_2$ and $\|\sqrt{v}\text{AME} - \text{SV}\|_\infty$ (or $\|\sqrt{v}\text{AME} - \text{SV}\|_2$) by $\varepsilon/2$.

Prop. B.2 directly yields the bound on the first term. In both cases (1) and (2), assumptions are verified as follows. First, a k-sparse AME is assumed in Case (1) and implied in Case (2) by a monotone utility plus a k-sparse SV (details in Corollary C.2). Second, the X_ns are bounded, since $\forall n \in [N] : X_n \in [-\sqrt{v}, \sqrt{v}]$ and v is finite:

$$v = \int_0^1 \frac{1}{p(1-p)} \frac{p^\varepsilon (1-p)^\varepsilon}{\text{B}(1+\varepsilon, 1+\varepsilon)} dp = \frac{\text{B}(\varepsilon,\varepsilon)}{\text{B}(1+\varepsilon,1+\varepsilon)} = 4 + \frac{2}{\varepsilon}, \tag{12}$$

where the last equality comes from the fact that $\Gamma(z+1) = z\Gamma(z)$.

To bound the second terms in both cases, notice that each version is given a bound in Lemma C.6, and that both approach 0 when $\varepsilon' \to 0$ and are positive when $\varepsilon' > 0$. In consequence, there exists $\varepsilon' > 0$ dependent only on ε such that both are $\le \varepsilon/2$, concluding the proof. □

D Efficient Sparse Beta-Shapley Estimator

Recall that $\text{Beta}(\alpha,\beta)$-Shapley (Kwon & Zou, 2022) is our AME defined on the distribution $\mathcal{P} = \text{Beta}(\alpha,\beta)$. We show that our regression-based AME estimator with p-featurization (§B) can efficiently estimate $\text{Beta}(\alpha,\beta)$-Shapley values when they are k-sparse, for all $\alpha > 1$ and $\beta > 1$, i.e., achieving low L_2 error with high probability using $O(k \log N)$ samples. Prop. B.2 directly yields the result. Its assumptions are verified given that:

$$v = \int_0^1 \frac{1}{p(1-p)} \frac{p^{\alpha-1}(1-p)^{\beta-1}}{\text{B}(\alpha,\beta)} dp = \frac{\text{B}(\alpha-1,\beta-1)}{\text{B}(\alpha,\beta)} = \frac{(\alpha+\beta-2)(\alpha+\beta-1)}{(\alpha-1)(\beta-1)} \tag{13}$$

is finite (the last equality is due to $\Gamma(z+1) = z\Gamma(z)$) and $\forall n \in [N] : X_n \in [-\sqrt{v}, \sqrt{v}]$ is also bounded.

E Extending to Approximate Sparsity

The above discussion assumes exactly k-sparse of the SVs, which in practice likely will not hold. In this section we extend our result to the case when it is only approximately sparse (Rauhut, 2010). In such a setting, small non-zeros are allowed in the remaining $N-k$ entries of the SVs. Formally, it requires that the best k-sparse approximation $\sigma_k(\text{SV}) = \inf_s\{\|\text{SV}-\text{s}\|_1 : \text{s is } k\text{-sparse}\}$ is small.

E.1 The LASSO Error Bound under Approximate Sparsity

First we extend the LASSO error bound in Prop. A.1. This is relatively easy since Theorem 1.4 from (Lecué et al., 2018) that Prop. A.1 has simplified supports approximate sparsity. We incorporate it by making two changes: a) β^* is now allowed to be an approximately sparse vector verifying $\sigma_k(\beta^*) \le C_4(\delta)\|\xi\|_{L_q} k\sqrt{\frac{\log(N)}{M}}$, where $C_4(\delta)$ is a constant, $\xi = \langle\beta^*, X\rangle - Y$ and q is some constant > 2; b) the result is accordingly rewritten to the following: there exists a regularization parameter λ and a value $C_5(B-A,\delta)$ such that with probability at least $1-\delta$,

$$\|\hat{\beta}_{lasso} - \beta^*\|_2 \le C_5(B-A,\delta)\|\xi\|_{L_q}\sqrt{\frac{k\log(N)}{M}}. \tag{14}$$

The proof is identical to that of Prop. A.1 thus omitted. Intuitively, the difference is that a k-sparse approximation with small enough error needs to exist to arrive to a similarly small enough error for the LASSO estimate. Another way to state this is that the sample size M in Eq. 14 is upper bounded by the approximate sparsity requirement (the smaller error the best k-sparse approximation is, the larger M can be). Denote the upper bound by $M_{max} = \max\{M : \sigma_k(\beta^*) \le C_4(\delta)\|\xi\|_{L_q} k\sqrt{\frac{\log(N)}{M}}\} = C_4(\delta)^2\|\xi\|_{L_q}^2/\sigma_k(\beta^*)^2k^2\log(N)$. Approximate sparsity has two implications. First, given that the error bound $\varepsilon = \mathcal{E}(M) = C_5(B-A,\delta)\|\xi\|_{L_q}\sqrt{\frac{k\log(N)}{M}}$ decreases monotonically as M increases, the minimal error possibly achievable is lower bounded by a function of the "sparsity level" $\sigma_k(\beta^*)$:

$$\varepsilon \ge \mathcal{E}(\lfloor M_{max}\rfloor) \approx \frac{\sigma_k(\beta^*)}{\sqrt{k}}\cdot\frac{C_4(\delta)}{C_5(B-A,\delta)}. \tag{15}$$

We note that (Jia et al., 2019) shares a similar lower bound on ε. Second, recall that $M \ge k(1+\log(N/k)) = M_{min}$ is required, implying that the theorem is only applicable when $\lfloor M_{max}\rfloor \ge M_{min}$. Because M_{max} increases with lower error sparse approximations, this is equivalent to require that:

$$\sigma_k(\beta^*) \le C_4(\delta)\|\xi\|_{L_q} k\sqrt{\frac{\log(N)}{\lceil M_{min}\rceil}} \approx C_4(\delta)\|\xi\|_{L_q}\sqrt{\frac{k\log(N)}{1+\log(N/k)}}. \tag{16}$$

Though this appears to be an extra requirement compared to (Jia et al., 2019), our empirical results suggest that it is not limiting in the cases we studied. Indeed, this only rules out sample sizes $\le k(1+log(N/k))$, smaller than the M needed for good performance across our evaluations.

We now restate the result as a form of (ε,δ)-approximation to make the result more approachable.

Corollary E.1. *For every sufficiently sparse β^* s.t. Equation 16, and every $\delta > 0, \varepsilon \ge \mathcal{E}(\lfloor M_{max}\rfloor), N \ge 3$, there exists some constant $C_7(B-A,\varepsilon,\delta)$ such that when the sample size $M = \max(\lceil M_{min}\rceil, \min(\lfloor M_{max}\rfloor, \lceil C_7(B-A,\varepsilon,\delta)k\log N\rceil))$, with a probability at least $1-\delta$, $\|\hat{\beta}_{lasso}-\beta^*\|_2 \le \varepsilon$.*

Proof. Let $\mathcal{E}(M) \le \varepsilon$, we have $M \ge C_5(B-A,\delta)^2\|\xi\|_{L_q}^2 k\log N/\varepsilon^2$. Because of $\|\xi\|_{L_q}^q \le e$ (a fact proved in the proof of Prop. A.1), we can further simplify it to $M \ge C_5(B-A,\delta)^2 ek\log N/\varepsilon^2$. Let $C_7(B-A,\varepsilon,\delta) = C_5(B-A,\delta)^2 e/\varepsilon^2$. Next, by clipping it with $\lfloor M_{max}\rfloor$, $\sigma_k(\beta^*) \le C_4(\delta)\|\xi\|_{L_q} k\sqrt{\frac{\log(N)}{M}}$ is verified and $\varepsilon \ge \mathcal{E}(M)$ still holds due to $\varepsilon \ge \mathcal{E}(\lfloor M_{max}\rfloor)$; Equation 16 further ensures that there exists at least a choice of M between $\lceil M_{min}\rceil$ and $\lfloor M_{max}\rfloor$. Finally, by further clipping with $\lceil M_{min}\rceil$, all preconditions are then satisfied and applying Equation 14 concludes the proof. □

E.2 Extending the SV estimators

We first derive an L_2 error bound assuming monotone utility, and later discuss an L_∞ error bound when utility is not monotone.

As a reminder, we chose a distribution $\mathcal{P}$ such that the AME under $\mathcal{P}$ is close enough to the SV, and then apply LASSO to estimate the AME with a low error. The LASSO error bound requires the sparsity of β^*, which utilizes the sparsity of the AME, which is derived from the sparsity of the SV. When SV is instead approximately sparse, we can still derive an approximate sparsity guarantee for the AME and consequently for β^*.

Lemma E.2. *When the utility is bounded in* $[0,1]$ *and monotone,* $\sigma_k(\beta^*) \le \sqrt{2}\sigma_k(AME) \le 3\sqrt{2}\sigma_k(SV)$ *holds for both truncated uniform* $Uni(\varepsilon', 1-\varepsilon')$ *and* $Beta(1+\varepsilon', 1+\varepsilon')$ *with* $\varepsilon' \in (0, 0.5)$.

Proof. Recall that $\beta^* = \text{AME}/\sqrt{v}$, we have $\sigma_k(\beta^*) = \frac{\sigma_k(\text{AME})}{\sqrt{v}} \le \sqrt{2}\sigma_k(\text{AME})$, where the inequality is due to $v \ge \frac{1}{2}$ for $Uni(\varepsilon', 1-\varepsilon')$ (see Corollary C.3) and $v \ge 4$ for $Beta(1+\varepsilon', 1+\varepsilon')$ (see (12)).

Next we connect $\sigma_k(\text{AME})$ and $\sigma_k(\text{SV})$. Monotonicity ensures that $\text{AME}_n > 0$ and $\text{SV}_n > 0, \forall n$. By (7), either $\text{AME}_n \le \text{SV}_n$ or $\text{AME}_n \le \text{SV}_n(1+\Delta)$ holds. Thus $\sigma_k(\text{AME}) \le (1+\Delta)\sigma_k(\text{SV})$. Further applying (8a) for $Uni(\varepsilon', 1-\varepsilon')$ and (11) for $Beta(1+\varepsilon', 1+\varepsilon')$ concludes the proof. □

With a similar application of the LASSO error bound as previously done in e.g., Corollary 3.7, we arrive at a similar (ε, δ)-approximation:

Corollary E.3. *For every constant* $\varepsilon > 0, \delta > 0, N \ge 3$*, there exists constants* $q > 2, \varepsilon'$*, a LASSO regularization parameter* λ*, and an* $M = O(k \log N)$*, such that with probability at least* $1-\delta$*:*

(1) $\|\sqrt{v}\hat{\beta}_{lasso} - SV\|_\infty \le \varepsilon$ *holds when the utility is bounded in* $[0,1]$ *and* $\sigma_k(AME) \le \frac{1}{\sqrt{2}} C_4(\delta)\|\xi\|_{L_q} k\sqrt{\frac{\log(N)}{\lceil M_{min}\rceil}} \approx \frac{1}{\sqrt{2}} C_4(\delta)\|\xi\|_{L_q}\sqrt{\frac{k\log(N)}{1+\log(N/k)}}$ *and* $\varepsilon \ge \mathcal{E}(\lfloor M_{max}\rfloor) \approx O(\sigma_k(AME)/\sqrt{k})$*;*

(2) $\|\sqrt{v}\hat{\beta}_{lasso} - SV\|_2 \le \varepsilon$ *holds when the utility is monotone and* $\sigma_k(SV) \le \frac{1}{3\sqrt{2}} C_4(\delta)\|\xi\|_{L_q} k\sqrt{\frac{\log(N)}{\lceil M_{min}\rceil}} \approx \frac{1}{3\sqrt{2}} C_4(\delta)\|\xi\|_{L_q}\sqrt{\frac{k\log(N)}{1+\log(N/k)}}$ *and* $\varepsilon \ge \mathcal{E}(\lfloor M_{max}\rfloor) \approx O(\sigma_k(SV)/\sqrt{k})$*,*

where the noise ξ *is defined as* $\langle \beta^*, X\rangle - Y$*, the AME is defined under* $\mathcal{P} = Uni(\varepsilon', 1-\varepsilon')$ *and* $v = \mathbb{E}_{p\sim\mathcal{P}}[\frac{1}{p(1-p)}]$.

The result of Beta distribution is similar.

F Evaluation Details

F.1 Warm-starting Optimization and Hyperparameter Tuning

Given the high cost of training deep learning models, we support an optimization that uses warm-starting as a proxy for full model training. Specifically, instead of training each submodel from scratch, we fine-tune the main model on each subset S for a fixed number of iterations, usually the number of iterations in one main model training epoch. Although this results in a more noisy estimate of $U(S)$, our estimator is able to handle the noise, yielding an overall speedup in wall clock time.

With warm-starting, instead of learning a model from the subset S, we "unlearn" the signal from the points not in S. This has two implications on our choice of $\mathcal{P}$. First, changing the outcome of a given query usually requires removing all its contributors, as even a small number of them is sufficient to maintain the signal learned in the main model. Hence, we only consider lower inclusion probabilities ($p \le 0.5$). Second, warm-starting does not collapse model even on very small data subsets, as opposed to learning the model from scratch. We can thus consider smaller values of p, and settle on the range $\mathcal{P} = Uni\{0.01, 0.1, 0.2, 0.3, 0.4, 0.5\}$.

Hyperparameters of model training for warm-starting. Warm-start training requires specifying the hyper-parameters (e.g., batch size, learning rate, training time, etc.). If the original learning rate changed adaptively during the course of training, e.g., when using a training approach such as Adam (Kingma & Ba, 2014), we use the learning rate and batch size for the final epoch and run fine-tuning for one epoch on the data subset.

Otherwise we fine-tune every subset model for the same fixed number of iterations, and vary batch size proportionally to the number of training examples included, such that every datapoint is iterated through roughly only once (i.e., one epoch on the data subset). Moreover we observe that when batch size is below a certain number the models soon all collapse. Therefore, we lower bound the batch size by that number, which is 100 for both CIFAR10 datasets and 20 for the ImageNet dataset. The reason of such a co-design is the following. To make the numbers comparable, the subset models should be all fine-tuned with the same number of steps. If we still use a constant batch size, every datapoint will be visited different number of times across different data subset sizes, making the impact of one source to be less comparable. Thus we decide

to vary the batch size accordingly such that each datapoint is visited roughly once. In addition, we choose the learning rate such that the validation accuracy of most subset models drops by roughly 20%, a not-too-large but still significant number. The reason is that the learning rate should be large enough that when no or few poison is included, the subset model should have the poisoning mostly erased, and vice versa.

F.2 Datasets and Attacks

We provide more context and details on the datasets, models, and attacks from Table 1.

Datasets. We evaluate our approach using the following four data sets and inference tasks:

a *CIFAR10*, an image classification task with ten classes (Krizhevsky et al., 2009). We consider each individual training sample as a single source, and use the `ResNet9` model and training procedure described in (Baek). For one of the attacks (Poison Frogs (Shafahi et al., 2018)), we use `VGG-11` instead of `ResNet9`, and use transfer learning to specialize a model trained on the full CIFAR10 data using the training procedure described in (Kuangliu). Transfer learning is used to specialize the model for 10% of the CIFAR10 training data. We only re-train the last layer and freeze every other layers.

b *EMNIST*, a hand-written digit classification task with examples from thousands of users each with their own writing style (Cohen et al., 2017). Each user is a data source with multiple examples. We use models and training procedures from (PyTorch, b).

c *ImageNet*, a one-thousand class image classification task (Russakovsky et al., 2015). The training data includes more images than CIFAR10 (over 1 million vs 50000), and each image has a higher resolution (average of 482x418 pixels vs 32x32 pixels), thus increasing the training overhead. For ImageNet, we group training data into sources using the URL where the image was collected. Specifically, we treat each URL path (excluding the item name) as a source, and then combine all paths contributing fewer than 10 images into one source. This results in a dataset with 5025 sources. We use a `ResNet50` (He et al., 2016) model trained using the procedure from (PyTorch, a).

d *NLP*, a sentiment analysis task on 1 million book reviews written by 307k Amazon users (Ni et al., 2019). Most users contribute fewer than 1000 reviews: we select and combine multiple such users at random when producing sources. This results in 1000 sources, 7 of which contain reviews from a single user, and the rest group random users into a single source. We use the model and training procedure described in (Trevett) to produce a binary classifier that uses the review text to predict whether or not the review has a positive score (i.e., greater than 3).

Attacks. We evaluate our approach using three types of poisoning attacks. (1) *Trigger attacks* (Chen et al., 2017) which rely on a human-visible trigger to poison models. On image detection tasks, we use a 5x5 red square added to the top of each image or a watermark as our trigger. Column k in Table 1 lists the number of poisoned sources. For the NLP task we use a neutral sentence as a trigger. (2) A *label-flipping attack* on EMNIST, where a poison source copies all the data from a benign user (in our case user 171) and associates a single label (in our case `6`) for all of this copied data. (3) The *Poison Frogs attack* (Shafahi et al., 2018) on CIFAR10, which is a clean-label attack that introduces imperceptible changes to training images that will poison a target model in a transfer learning setup, where the last few layers of an existing pre-trained model are refined using additional training data to improve inference performance. Figure 3 shows examples of trigger and poison frog attacks.

Hyperparameters. We evaluate the impact of hyperparameters using micro-benchmarks in §F.3. Unless otherwise stated, we use $q = 0$ for knockoffs. As we explain in §F.3, this is a conservative choice that seeks to minimize the false discovery rate. We also use $\mathcal{P} = Uni\{0.2, 0.4, 0.6, 0.8\}$ for training-from-scratch and $\mathcal{P} = Uni\{0.01, 0.1, 0.2, 0.3, 0.4, 0.5\}$ for warm-start training.

We use Glmnet (Balakumar; Hastie et al., 2016) for the LASSO implementation. For the LASSO regularization parameter λ, a correct choice is required by our error bound (Prop. 3.3), which unfortunately we have no information on. Bypassing this obstacle remains as an interesting future work. Indeed this has been studied in (Lecué & Mendelson, 2017) and other slightly weaker LASSO error bounds (e.g., Theorem 3.5.1. from (Pauwels, 2020)) but with known λ exist. In practice we choose the λ using a common empirical procedure (Hastie et al., 2016) that runs 20-fold cross-validation (CV) and chooses the largest λ (sparsest model) with errors within one standard deviation of the best CV error, which we denote as λ_{1se}. For the SV estimation, we use λ_{min} that gives the best CV error, as we are not seeking a sparsest model here.

F.3 Ablation Study for Different Parts of the Methodology and Parameters

Next we use microbenchmarks to evaluate the effect that different hyperparameters have on our methods. Then, we show a discussion of hyperparameters for warm starting, the benefits of using Knockoffs, and a comparison between LASSO and diff-in-means, a straight-forward AME estimator that uses empirical means to estimate expectations. Finally, we discuss the

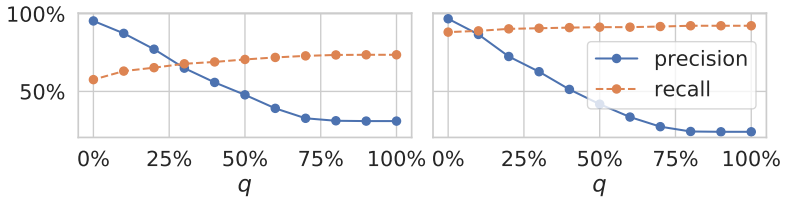
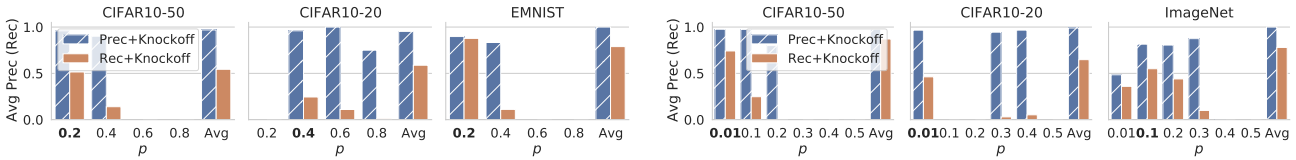


Figure 11: **Effect of q in Knockoffs on CIFAR10-50. Left: training-from-scratch; Right: warm-start training.**



(a) **Training-from-scratch** (b) **Warm-starting**

Figure 12: **LASSO results run only on observations from a single p value, we highlight the p value providing the best result. `Avg` is the result from using the entire p value grid.**

tradeoff between using training-from-scratch and warm-starting.

F.3.1 Effect of Hyperparameters

Effect of the Target FDR Level q. All results presented thus far were with $q = 0$. Here, we vary q, which allows us to trade-off precision to achieve higher recall. For this microbenchmark we use λ_{min}, which we define as the λ with the best cross-validation error, instead of λ_{1se} which we use in the rest of our evaluation. This is because the precision provided by LASSO imposes a lower-bound on the precision achieved using Knockoffs', and in our experiments we found that LASSO alone can achieve high-precision when λ_{1se} is used, making it harder to observe effects at lower values of q. Figure 11 shows the results of varying q in the CIFAR10-50 setting. We can see that while increasing q does lead to a small increase in recall, it comes at significant cost to precision.

Effect of p. Next we address the question of how values of p affect our method. Figure 12 shows our metrics on subset models drawn with one single value of p. We can see that no single value of p suffices across datasets and training algorithms. For instance, training-from-scratch $p = 0.2$ works well for CIFAR10-50 and EMNIST, but not for CIFAR10-20. This difference between CIFAR10-50 and CIFAR10-20 is likely due to the power of the attack: both use the same attack, but CIFAR10-50 uses a larger number of poisoned sources. This means that smaller values of p are more likely to select poisoned sources in CIFAR10-50, explaining our observations. For warm starting, we confirm that small values of p ($p = 0.01$) perform better than larger ones. These results thus show that (a) no single value of p suffices for all models, thus motivating our use of a grid, and (b) the grid $\mathcal{P}$ can be tuned when a prior is available.

Effect of adding less informative ps. One might wonder how including sub-optimal p values impacts our results? To answer this question, in Figure 12, we compare the single p results to that of using all ps in the grid (shown as `avg`). We find that while very uninformative ps can hurt our results (e.g., with EMNIST), this is rare and the effect is small. On the other hand, in many cases the grid result is better than the result from just using the single best p.

Effect of different ps on different queries. As discussed above, no single value of p suffices across all datasets. Surprisingly, as shown in Table 3, we found that even within the same dataset, the best p can differ *across queries*.

Comparison against fixing $p = 0.5$. We also compare with a simple distribution where each subset is sampled with equal probability. This corresponds to using fixed $p = 0.5$. The result is much worse than our default sampling over a grid as shown in Table 4.

F.3.2 Benefit of LASSO and Knockoffs

We evaluate the benefit of using LASSO over the more straight-forward diff-in-means that replaces the expectation with empirical mean in Eq. 1, and the power of Knockoffs in controlling the FDR. First, to have a clean comparison against diff-in-means, we only run LASSO purely without adding the Knockoffs component. The selection is replaced with a procedure that selects all positive coefficients; For diff-in-means we select a threshold such that the recall matches that of pure LASSO for easy comparison. The result in Table 5 shows that LASSO performs stably with valid precision while diff-in-means is fragile especially under high noise as in warm-starting. Next we compare this pure LASSO result against LASSO+Knockoff with varying number of observations in Figure 13. We can see that LASSO+Knockoffs ensure that

Query	Prec_0.4	Rec_0.4	Prec_0.6	Rec_0.6
#3	**90.9**	**50.0**	nan	0.0
#7	**91.7**	**55.0**	nan	0.0
#9	100.0	10.0	**100.0**	**20.0**
#10	100.0	15.0	**100.0**	**25.0**
#14	**100.0**	**40.0**	nan	0.0
#18	75.0	15.0	**100.0**	**20.0**

Table 3: **Single-p LASSO results for $p = 0.4$ and $p = 0.6$ on selected queries, CIFAR10-20 training-from-scratch. $p = 0.4$ has better average results (see Fig. 12), but $p = 0.6$ outperforms for some queries.**

	precision	recall	c
CIFAR10-50-tfs	100	3.2	8
CIFAR10-50-ws	100	1.8	12
EMNIST	100	7.8	16

Table 4: **Experiment results of using fixed $p = 0.5$. "tfs" denotes training-from-scratch and "ws" denotes warm-starting.**

precision remains high even with a small number of samples. This is in contrast to LASSO which reduces precision when the number of samples decrease. Thus LASSO+Knockoffs allow our technique to be safely used even when an insufficient number of observations are available.

F.3.3 Training-from-scratch vs Warm-starting

Recall that warm-starting can greatly speed up the training procedure, but with a potential drawback of introducing higher noise for LASSO. In our evaluation, training CIFAR10-50 from scratch took 108.5 seconds, compared to 7.9 seconds when using warm-start training, representing a speedup of 13.8X. However, more observations might be required when using warm-start training. For example, in Table 6 we find that for CIFAR10-20 the warm-start model achieves very low recall (0.8%), while training from scratch achieves reasonable recall (58.8%). We believe this is because of noise due to the number of poison sources required to trigger the attack. On the other hand, we see that CIFAR10-50 does not suffer from this problem, and in fact warm-start provides better results than training from scratch. This is because lower values of p work well to trigger the attack for CIFAR10-50, and warm starting over-samples this region. Adding more observations when using warm-start training with CIFAR10-20 improves this situation, and we find that when using $c = 48$ we can achieve a recall of 64.8%.

F.4 Runtime Evaluation

Run times of LASSO. The time to run LASSO grows linearly with the number of observations, as shown in Figure 14. Together with Figure 13 we can see that our technique can achieve good precision using a small number of observations, which translates to a small query time. Precision and recall improve with more observations, but this increases the time taken to execute a query.

Run times of inference. We report the total time of running ImageNet inference on a single RTX8000 GPU for a batch of 40 queries on 739 models, which corresponds to $c = 12$. We implement the inference of a model as three steps: allocating the model memory, loading the model from disk to GPU, and the actual GPU computation. They take 293.6, 447.5, and 584.9 seconds respectively for 739 models in total. The current throughput is 1.81 queries per minute, and the latency is 22.1 minutes. We note that both numbers could be further optimized (e.g., the model memory allocation can be done only once and reuse for every model; the model loading and actual computation can be pipelined, etc.), and one can also batch more queries together to further improve throughput. In this paper we focus on improving precision and recall. We plan to focus on further performance optimizations in the future.

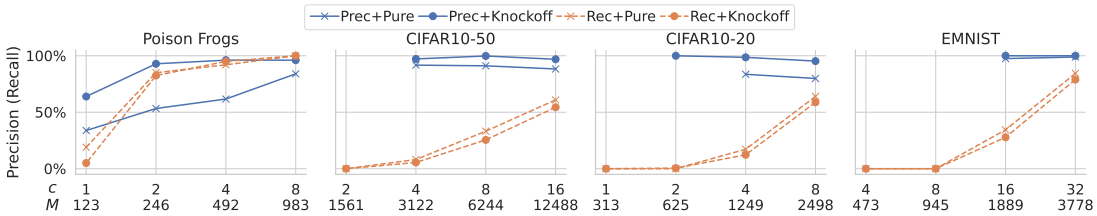
F.5 Additional Evaluation of Comparison with Existing Works

F.5.1 Additional Evaluation of SCAn

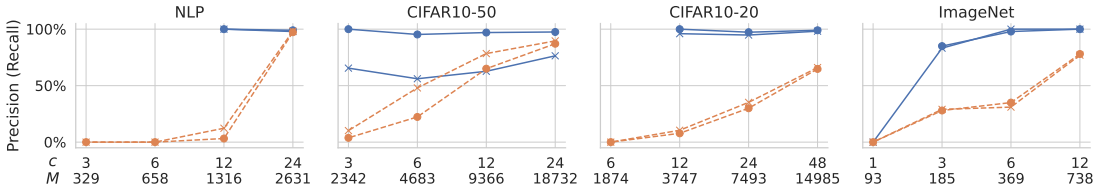
SCAn (Tang et al., 2021) is a state-of-the-art poison defense technique designed to identify whether or not a given model is poisoned, and find the poisoned class when it is. It requires access to some clean data (10% by default). SCAn computes an AI (Anomaly Index) score for each class, and reports that a class is poisoned if its AI$>$ 7.3891. In their "online setting", it can also detect if a query input is poisoned by clustering on features extracted from the last hidden layer's activation. An input is marked as poisoned if it belongs to a poisoned class and is clustered in the group with fewer clean data. We modified SCAn to run on a training set to detect poisoned training data.

	Prec+ Pure	Rec+ Pure	Prec+ Diff	Rec+ Diff	c
Poison Frogs	84.0	100.0	**100.0**	100.0	8
CIFAR10-50-tfs	**88.3**	60.9	28.5	60.9	16
CIFAR10-20-tfs	**79.9**	64.0	76.7	64.0	8
EMNIST	**98.9**	84.4	62.9	84.4	16
NLP	**97.9**	98.2	5.6	98.2	24
CIFAR10-50-ws	**76.4**	89.6	41.6	89.6	24
CIFAR10-20-ws	**98.2**	66.0	9.2	66.0	48
ImageNet	**100.0**	77.0	94.2	77.0	12

Table 5: **Average precision and recall of *pure LASSO* and *diff-in-mean*. *tfs* and *ws* denote training-from-scratch and warm-starting respectively.**



(a) **Training-from-scratch**



(b) **Warm-starting**

Figure 13: **Effect of growing c. "Prec+Pure" is the precision of LASSO without knockoffs; "Prec+Knockoff" is the precision of LASSO with knockoffs. "Rec+Pure" and "Rec+Knockoff" show the recall for both setups. Poison Frogs uses transfer learning and is not amenable to fine-tuning.**

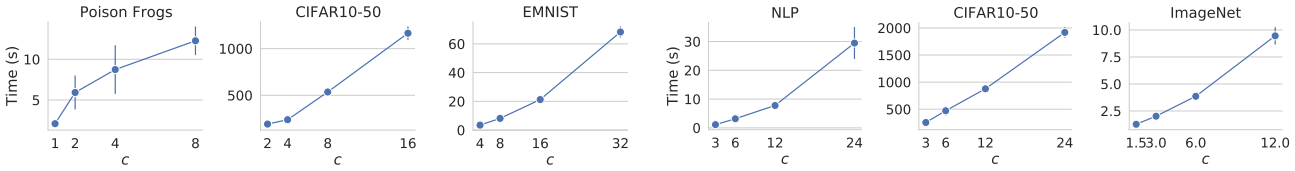
Modification of SCAn. Originally in the online setting, SCAn is designed to detect poison in every incoming test datapoint. It trains an untangling model on a clean set of data, and on receiving a test datapoint, it first fine-tunes the model and then use it to cluster with all the data it has to decide if the given test datapoint is poisoned. To modify it to detect poison data in the training set, instead of on the test data we train the untangling model on the training data. For efficiency, we assume the whole training set is available in the beginning, so that we only need to train the model once on the whole training set. This gives advantage to SCAn because originally in the beginning the model is poorly fit and gives many false positives because of insufficient data. SCAn also only gives a coarse-grained (poisoned v.s. clean) outcome to each datapoint, preventing us from exploring its precision-recall tradeoff in evaluation. Thus we modify SCAn to provide a score for each datapoint, using the distance to the hyper-plane ($v^T r$ in Eq 6 of the paper).

The result. Outlier detection approaches like SCAn target a somewhat different problem than data attribution. Outliers can represent not just attacks but also benign but rare data, and thus outlier-detection cannot differentiate between attacks and benign data, nor between different attacks on the same data. We evaluate this effect using a mixed attack setup on CIFAR10, where we simultaneously use 3 different attacks, each of which poisons 20 images. The attacks used are trigger attacks (similar to CIFAR10-20), except each attack places the trigger in a different location. We found that SCAn clusters nearly all poisoned images (along with many clean images), regardless of the attack used, into the same group, showing that it cannot differentiate between attacks. In contrast, when using our approach with the same hyperparameters as CIFAR10-20 from warm-starting, ours can select the correct attack for each query, and achieves an average precision (recall) of 96.3% (65.5%), 97.4 (89.5%) and 97.1% (71.3%), respectively for 20 random queries from each attack.

Furthermore, even when a single type of attack is used, SCAn cannot detect all of the poisoned data. We evaluate this by using SCAn on CIFAR10-50, CIFAR10-20, EMNIST and ImageNet. To make SCAn work for EMNIST and ImageNet, which are source-based datasets, we adapt it to assign a score to a source equal to the number of selected datapoints within the source. The results in Fig. 8 shows that SCAn falsely selects many clean data for both CIFAR10 benchmarks,

	Training-from-scratch		Warm-starting		c
	Prec	Rec	Prec	Rec	
CIFAR10-50	96.9	54.4	**97.0**	**77.6**	16
CIFAR10-20	**95.3**	**58.8**	100.0	0.8	8

Table 6: **Precision and recall comparison for training-from-scratch vs warm-starting under the same number of observations.**



(a) **Training-from-scratch** (b) **Warm-starting**

Figure 14: **Average run times of LASSO on the queries as we grow c on training-from-scratch and warm-starting benchmarks. The error bar draws the standard deviation. CIFAR10-20 results are almost identical and thus omitted.**

yielding very low precision. This is consistent with the results reported by SCAn authors, who state that at least 50 poison datapoints are needed for SCAn to work robustly. On the other hand, SCAn achieves perfect precision and recall on the other 2 benchmarks, since they are source-based datasets that contain hundreds of poison datapoints. Further, we found that SCAn's AI fails to identify the poisoned class for all datasets other than EMNIST. Thus SCAn cannot be used unless we are guaranteed to have a sufficiently large set of datapoints.

Additional evaluation for other variants of SCAn.

We also study if it helps SCAn to add more poison datapoints. We observe that SCAn starts to work well when we run SCAn on 104 and 300 poison datapoints in CIFAR10-50 and CIFAR10-20, respectively.[2] The full result is shown in Table 8, where the we show both the precision and recall of the original SCAn and that of the variant after our distance modification. In the distance-based variant, we compute the precision and recall with a threshold that makes the recall match that of ours as closely as possible. This leads to an ad-hoc solution to this insufficient data problem: because query inputs are likely to be poisoned data, the operator will wait for an enough number of them to supplement the dataset before running SCAn. However, it is unclear how many queries should s/he wait for since (a) the number of poison data needed can vary across attacks and we do not know it in advance, and (b) even if we know it, we cannot know if this query is an attack. The problem is even worse when multiple attacks coexist, in which case the queries may spread across different attacks/classes, increasing the time needed for each attack/classes to have accumulated sufficient poisons.

On the mixed attack, with more poison datapoints SCAn still clusters the three attacks together, though it stops falsely clustering clean data into the same group as poisoned data.

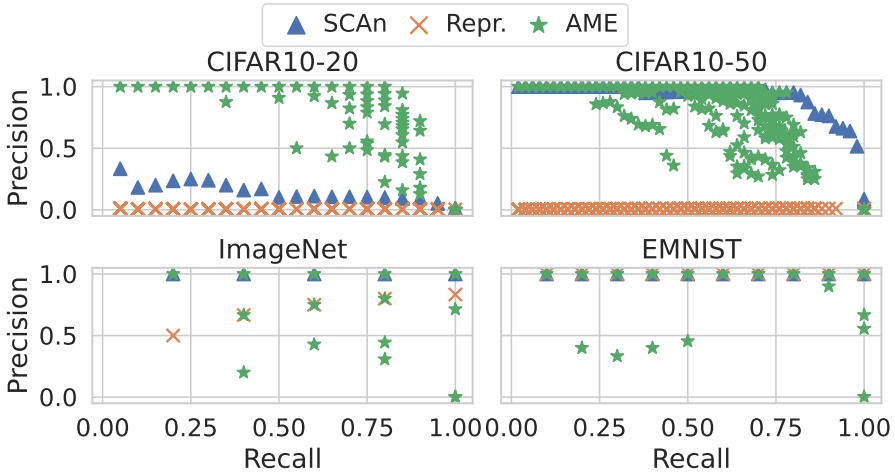
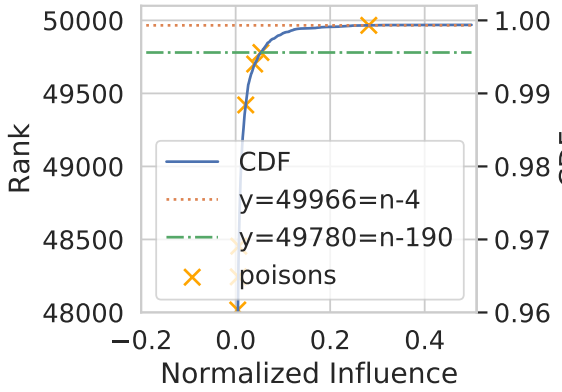


Figure 15: **The counterpart of Fig. 8 with AME using the regularization parameter choice of λ_{min}.**

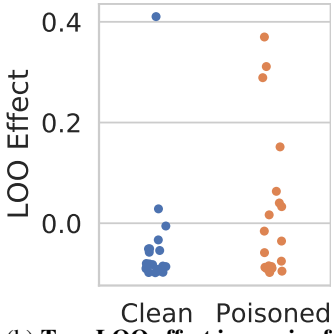
[2]Note that the main model is still trained on the original dataset without additional poison datapoints added.

	Ours		Repr.	
	Prec	Rec	Prec	Rec
ImageNet	100	78.0	85.9	78.0
CIFAR10-50	96.9	54.4	99.9	54.4
CIFAR10-20	95.3	58.8	68.9	58.8
EMNIST	100	78.9	100	78.9

Table 7: **Comparison with Representer Points at the same recall level. Representer Points use best tuned λ assuming knowledge of ground truth.**



(a) **Part of CDF of the influence scores.**



(b) **True LOO effect in y-axis of selected clean and poisoned datapoints.**

Figure 16: **Experiment of the influence function on CIFAR10-20. The loss function is defined as the average loss on 20 query inputs. We use the parameter configuration of $r = 10, t = 1000$, damping term $\lambda = 0.01$, and scale term=25.**

F.5.2 Additional Evaluation of Rrepresenter Point

Representer Points (Yeh et al., 2018) provide an approach to quantify the contribution each training datapoint has on a given query prediction. Given a query t, this approach assigns a score $\alpha_{i,Q_L} f_i^T f_t$ to each training datapoint i, where f_i is the feature extracted from the activation of the model's last hidden layer, f_t is the query input's feature, and α_{i,Q_L} is a scalar computed from the model's gradients. The query effect $f_i^T f_t$ is interpreted as the similarity between i and t, while the model effect α_{i,Q_L} represents i's importance to the model. There is a hyperparameter λ to trade off between the computation time and the result: smaller λs provide better results but are slower.

We compute a source-based score by summing up scores for all datapoints within a source, and pick the threshold such that the recall matches AME's recall as closely as possible. In our evaluation, we used $\lambda = 3e-3$; Figure 8 shows the results. While this approach performs better than ours on EMNIST, it is worse on ImageNet and does not work on either of the CIFAR10 datasets. We note that λ is crucial to the result, but the paper's suggestion of using a small λ like $3e-3$ does not always lead to good results because it tends to weight model effects over query effects. The paper does not document this problem, so it is unclear how λ can be tuned to avoid the problem. Indeed, finding a good λ requires prior knowledge of what sources are poisoned, as we show next.

We discover that λ has a potential effect crucial to result: it balances between model effect v.s. query effect, which is undocumented in the paper. The default λ gives bad result, so we manually tune it per attack with a grid search assuming knowing the ground truth and report the best result, as shown in Table 7. It works better than ours on EMNIST and CIFAR10-50 but worse on ImageNet and CIFAR10-20. We note that the best λ can vary across datasets: as shown in Table 9 the λ that works the best on CIFAR10-50 can be 30, which gives poor result on CIFAR10-20. Therefore, it is unclear how to find the right λ that works against unknown attacks even on the same dataset.

To investigate why the default λ falls apart, we look at what are selected and find that regardless of the query, it tends to select many the same images: the selection results between two random clean queries of the same predicted label often share $> 70\%$ of selections. This is likely because small λ tends to overweight the model effect.

F.5.3 Additional Evaluation of Influence Functions

Influence functions provide an approach to estimating the leave-one-out (LOO) effect, defined as $L_{excluded} - L_{original}$ where L denote the model's loss. We evaluated their efficacy for identifying poison datapoint by computing the influence function for each training point in the CIFAR10-20 setup. We expect proponent points will have higher influence scores, and

thus poisoned data will have a higher influence score. In Figure 16a we show a CDF of influence scores for each training datapoint, as well as their rank by influence value. We observe that only 2 (out of 20) of the poisoned data points show up in the top 190 elements. We can thus see that influence functions do not suffice for detecting poisoned datapoints in this case.

We next examine the efficacy of directly using leave-out-one training to detect poisoning. We do so by training 39 models where we leave out the 10 datapoints with the highest influence scores, the 10 datapoints with the lowest influence scores, and any of the 20 poison datapoints not included in these sets. In Figure 16b we show the LOO effect for all of these models. We can observe from this that many poisoned datapoints have LOO effects that are nearly as low as those observed for correct datapoints. This shows that a more precise technique for estimating LOO would still be insufficient for detecting poisoned sources.

F.5.4 Additional Evaluation of Shapley Value

Due to prohibitive costs of computation for running existing Shapley value estimators on our large experiments, we instead provide an evaluation on a subset MNIST with data poisoning (see details of the dataset in Appendix F.8.2). Unlike previous experiments in which each query explains a single prediction on a test example, we now look at one query to explain the attack success rate on an entire poisoned test set. The utility measurement is hence much less noisy than in our main experiments (we expect the AME to be even better comparatively in a noisy setting, except maybe for Compressive Sensing). The baselines are KernelSHAP (Koh & Liang, 2017), Monte Carlo (Ghorbani & Zou, 2019), Truncated Monte Carlo (Ghorbani & Zou, 2019), and two sparsity-aware approaches "KernelSHAP (L1)" (Koh & Liang, 2017) and Compressive Sensing (Jia et al., 2019) (implementation details in Appendix F.8). AME uses the training-from-scratch setting with $\mathcal{P} = \{0.2, 0.4, 0.6, 0.8\}$. The sample sizes (i.e., number of utility evaluations) are all fixed to 1024, a number that is small for consistency with the large experiment setup, and still exceeds $N = 1000$ so that permutation (Monte Carlo) and non-regularized regression (KernelSHAP) based approaches are applicable.

We again compare the precision of each method at different recall levels, by varying internal decision thresholds. The result in Figure 17 shows that sparsity-aware approaches achieve better performance than others. In particular, AME and "KernelSHAP (L1)" achieve the same performance and outperform other approaches. We also compare them for SV estimation in §5.3, and show that KernelSHAP (L1) is less suited in that case.

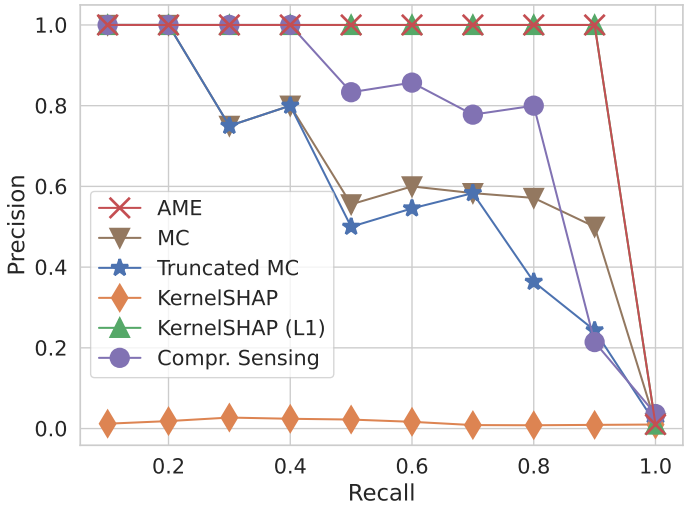


Figure 17: **Precision vs Recall curve comparison.**

F.6 Details of the Hierarchical Design

Our hierarchical design is as follows: we partition users into groups based on when they first posted a review, which we treat as a proxy for when the user account was created. We group users into partitions of 1000 users each (so groups span different time periods: we found that fixed group sizes performed better than fixed time periods). Unlike the previous evaluation with the NLP data set (§d), in this experiment we do not combine tail users, i.e., users with a small number of

review, into a single user. We use this process to partition the NLP dataset into $n_1 = 308$ top-level sources, each of which contains (up to) 1000 second-level sources. This partition covers all 307,159 users in the NLP dataset (i.e., $n_2 = 307,159$). We poison 15 second-level sources belonging to 2 randomly chosen top-level sources: putting 10 poisoned sources in one top-level source, and 5 in the other. We compare the results of applying our hierarchical approach in this setting, to the results of using the general approach on $307,159$ sources.

We use different ps when selecting top-level and second-level sources in the hierarchical approach, because top-level sources might contain multiple proponents and thus have a larger influence on inference results. In our experiment, we used $\mathcal{P} = Uni\{0.2, 0.4, 0.6, 0.8\}$) for the first-level ($p_1$) and trained models from scratch, and we used $\mathcal{P} = Uni\{0.1, 0.2, 0.3, 0.4, 0.5\}$ for the second level (p_2) and used warm-start training.

Figure 5 shows the precision and recall achieved by the hierarchical approach on 20 different queries, as we vary the number of subset models. We find that (a) the hierarchical approach achieves good precision and recall in identifying top-level sources that contribute to data poisoning, even when fewer subset models are used than would allow second-level sources to be detected; and (b) recall when identifying second-level sources degrades gracefully as fewer observations are used. The hierarchical approach is also more efficient: it can achieve near-perfect precision and 60% recall with 1,588 observations. In comparison, the general method could not identify any of the poisoned sources when using the same number of observations.

Finally, the hierarchical approach also reduces the time spent running LASSO+Knockoffs, i.e., it reduces query runtime. To demonstrate this, we compare the running times for executing a query with 1,588 observations, using 5 threads on a server with 256GB of memory. The non-hierarchical approach takes 52.5s to construct the design matrix and 688.58s to run LASSO+Knockoffs. The hierarchical approach takes 37.343s to construct two design matrices (0.123s for the top level, and 37.220s for the second level) and 13.829s to run LASSO+Knockoffs (2.369s for the top level, 11.460s for the second level). This corresponds to an overall query time of 48.68s for the hierarchical approach versus 741.08s for the general approach, a reduction of $15\times$.

F.7 Details and Additional Evaluation of Data Attribution for Non-poisoned Predictions

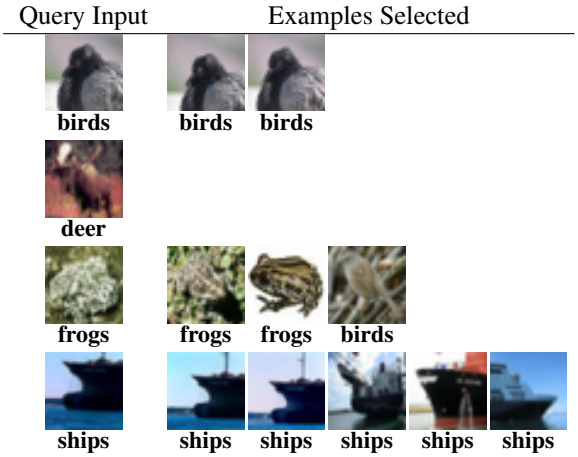


Figure 18: **Queries for correct predictions. The last row only shows a subset since they are too long to fit in.**

We randomly select 10 classes and their images from the ImageNet dataset and use the same ResNet-9 model and training procedure. The main model achieves a top-1 accuracy of 89.2% on the validation set, from which we randomly select 40 with correct predictions and 40 with incorrect predictions, as the queries. We train roughly 2700 subset models (calculated from $c = 10, k = 20, N \approx 10000$) from scratch. We switch to using λ_{min} (with $q = 0$) since empirically λ_{1se} is too conservative and selects few images, if any.

The results in Figure 6 and 7 (due to space constraints we show at most 3 selected images for each query) show that many selected images share similar visual characteristics with the query input, either in color, blur effect or texture. Some even are photos taken on the same place from different time/angle. These results suggest that our approach can be used to understand model behavior beyond the use case of data poisoning. Additional results for different queries, and a comparison to a naive baseline, can be found at https://enola2022.github.io/.

We also evaluate model explanation on CIFAR10-50 dataset. We reuse subset models from the CIFAR10-50 training-from-

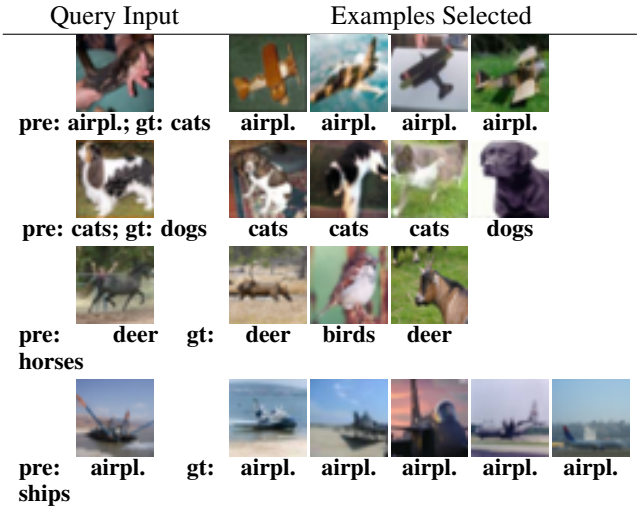


Figure 19: **Queries for wrong predictions. *Pre* is model's prediction and *gt* is the ground truth. The last row only shows a subset since they are too long to fit in.**

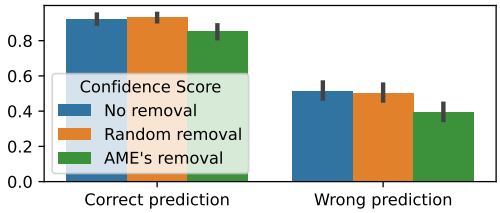


Figure 20: **Quantitative comparison of model explanation through the drop in the confidence scores. Y-axis shows the average confidence scores across all correct/wrong queries. The 95% confidence interval is drawn as the vertical bars.**

scratch experiments[3], and randomly select 39 images from the vanilla, non-poisoned test set as query inputs. Some queries with correct main model predictions (4 out of 39 in our evaluation) still fail because LASSO does not converge in 3600 seconds, possibly because no sparse solution exists. As a result we return an empty result set. Selected results are shown in Figure 18 and 19, and we also draw all 39 queries we have ran and their result in grids, as shown in Fig. 25 and Fig. 26, where each row consists of two queries, each starting with the query input followed by images selected.

We also report a quantitative result on CIFAR10-50 by removing the images selected by ours and retraining the model 6 times and measure the change in the predictions. We find that the predicted labels are often not changed compared to the main model. This is expected because our approach is designed to achieve good precision and as a result only a small number of proponents from the strongest are removed, and other proponents in the class generically support the prediction. However, we still see the average confidence score is lower compare to the main model's when only removing the strongest proponents, as shown in Fig. 20, while a baseline that randomly selects datapoints in the class to remove fails to do so.

F.8 Shapley Value Estimation Setting and Additional Evaluations

F.8.1 Simulated Dataset

Experimental Setup. We craft a two-valued threshold function as the utility, which evaluates to 1 when there are at least 2 in the first k sources are present. Formally, $U(S) = \mathbf{I}(|S \cap [k]| \geq 2)$. We pick $k = 3$ and $N = 1000$ for this experiment. This design allows us to compute the estimation error $\|\sqrt{v}\hat{\beta}_{lasso} - \text{SV}\|_2^2$ because the true value of Shapley values for the k sources can be easily known: by symmetry they are $U([N])/k = 1/k$ for the k sources and 0 otherwise.

We compared our AME based SV estimator to Monte Carlo (Jia et al., 2019; Ghorbani & Zou, 2019), Truncated Monte Carlo (Ghorbani & Zou, 2019), Group Testing (Jia et al., 2019), Compressive Sensing (Jia et al., 2019), KernelSHAP (Lundberg & Lee, 2017), and Paired Sampling (Covert & Lee, 2021). We run each approach 6 times with different random seeds. We plot the 95% confidence interval in the shaded area. The baseline is as follows:

[3]We did this to save time for model training. Though the training set includes poison, they should have little effect on the non-poisoned query inputs.

a *Truncated Monte Carlo.* We adopt the implementation published by the author (Ghorbani & Zou, 2019) and use the default hyperparameters (e.g., truncation tolerance is set to 0.01).

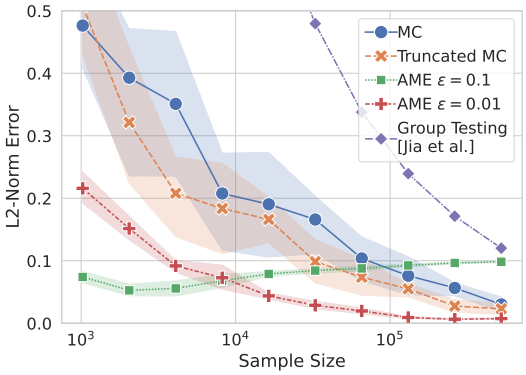
b *Group Testing.* We adopt the implementation provided by the author (Jia, 2021). We use hyperparameter $\epsilon = \frac{2}{\sqrt{N}}$ as in the script. We also tried $\epsilon = 0.01$ and 0.1 and the results remain the same, thus omitted.

c *Compressive Sensing.* We implement the algorithm using CVXPY (Agrawal et al., 2018; Diamond & Boyd, 2016). The algorithm has a hyperparameter "M" that has no clear documentation for its choice. We therefore run one set of the 6-trial experiment for every $M \in \{2^7, 2^8, \ldots, 2^{16}\}$ and report for each sample size the best mean estimation error (the mean is taken on the 6 trials). This assesses its upper limit. We use $\epsilon = 0.01$ for the other hyperparameter "ϵ" when not specified. Unlike the original algorithm that shifts the SVs by the average utility $U([N])/N$ (see $\bar{s}$ in Algorithm 2 in their paper), we do not perform shifting because in a poisoning (thus sparse) setup most SVs are more likely to be zero rather than the average.

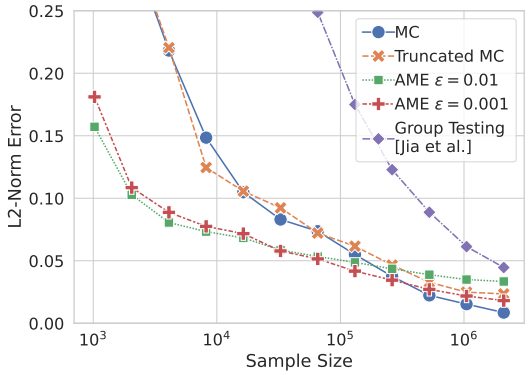
d *KernelSHAP.* We use their official implementation with the default hyperparameter setting. Note that by default KernelSHAP also uses L_1 regularization in a heuristic way. We also evaluate it with the regularization turned off to understand the effect.

e *Paired Sampling.* We use their official implementation with batch size equal to 128, no convergence detection and non-stochastic cooperative game.

Additional evaluations and ablation study. In §5.3 in the main body we have shown a condensed result comparing against part of the baselines. Now we present the comparison against other baselines in Fig. 21a. High-level findings remain the same. We can also see that smaller ε leads to higher approximation precision at the cost of convergence speed. We also provide additional results of other choices of ϵ for Compressing Sensing in Figure 22a. It shows that $\epsilon = 0.1$ though has a marginal drop compared to $\epsilon = 0.01$ in estimation error when the sample size $\leq 2^{12}$ but with a (relatively big) sacrifice on the estimation error on large sample sizes, echoing a tradeoff role ε plays in AME (see §5.3), while $\epsilon = 0.001$ has almost identical result, suggesting a saturation on one end of the tradeoff space. This may also explain another interesting finding: the results are almost the same as Monte Carlo for the two smaller ϵ's. Recall that compressive sensing utilizes ϵ to control the tradeoff between the sparsity and the accuracy (w.r.t. the measurements) of the recovered solution. Jia et al.'s design implicitly makes use of Monte Carlo for measurement, implying that the most accurate solution is Monte Carlo. The smaller the ϵ, the more accurate the recovery and hence the closer the result is to Monte Carlo.



(a) **Simulated dataset.**



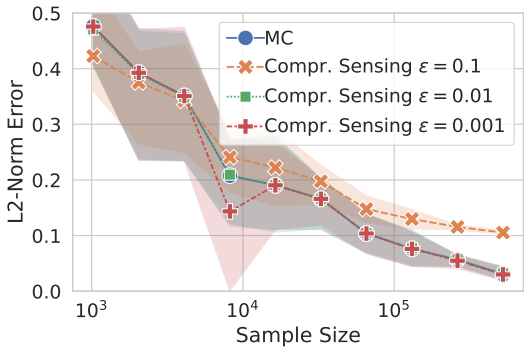
(b) **MNIST dataset. We use as ground truth the Monte Carlo (MC)'s result on a sample size of 2^{22}.**

Figure 21: **Additional comparison against other baselines on SV estimation.**

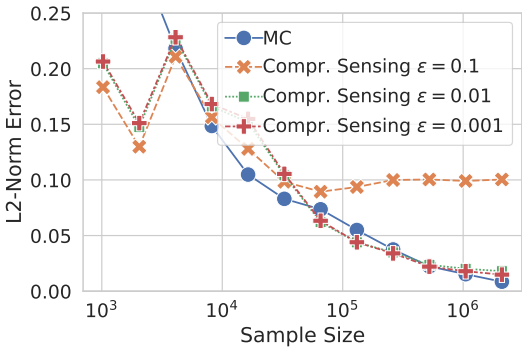
We provide additional evaluations regarding the choice of featurization and distribution of our SV estimator from AME. In Fig. 23 we compare the impact of featurization and fix the distribution to truncated uniform. It shows that they both work well and p-featurization performs slightly better (§B); In Fig. 24 we compare beta distribution with truncated uniform $Uni(\varepsilon, 1-\varepsilon)$. The parameters for beta distribution are chosen to match that of truncated uniform such that their AMEs evaluate to the same value. In other words, they both converge to the same estimation error on an infinite sample size. It shows that they both perform well, but truncated uniform performs slightly better.

F.8.2 Poisoned MNIST dataset

Experimental Setup. For non-simulated case, given the prohibitive cost of computing the true SV, we craft a tiny MNIST dataset by randomly subsampling 1000 datapoints from both the original training set and test set. To create sparsity, we



(a) **Simulated dataset.**



(b) **MNIST dataset.**

Figure 22: **Compressive Sensing with different choices of ϵ.**

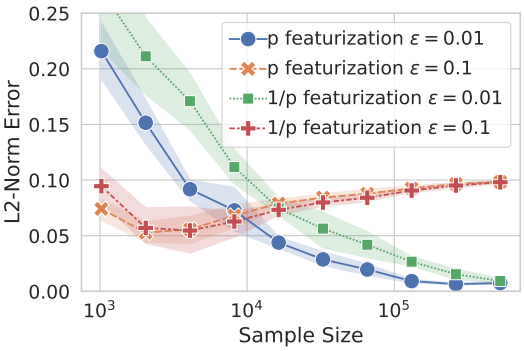


Figure 23: **p-featurization vs $1/p$-featurization (§B). Both use truncated uniform distribution $Uni(\varepsilon, 1-\varepsilon)$.**

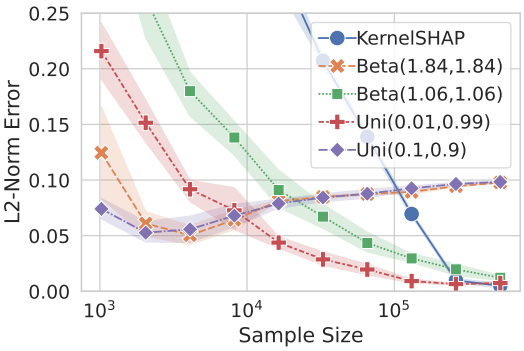


Figure 24: **Beta vs truncated uniform distribution. Our approaches (Beta* and Uni*) all use p-featurization (§B).**

further poison 10 training datapoints randomly by imposing a white square on the top-left corner and overwriting their labels to 0. We also craft a poison test set by imposing the same trigger on the clean test set and use the attack success rate as the utility function. We use logistic regression (with regularizaiton 0.02) as the model for fast training. The resulting attack success rate of the full model is 65.7%, and the test accuracy is 89.4%. Different from the simulation case, it is computationally infeasible to compute the true SV. We instead use the estimation from the classic Monte Carlo method (Ghorbani & Zou, 2019) as a proxy. We keep doubling the sample size until the estimation converges with a consecutive L_2 difference falling below 0.01, and use the final estimation as the proxy.

We compare the same set of approaches as in the simulated experiment (except Paired Sampling given the computational cost and its almost identical performance to KernelSHAP on the simulated dataset). The configurations for these approaches mostly remain the same, except we only run one trial to save computational cost.

Additional evaluations. Fig. 21b shows the comparison against other baselines not shown in §5.3 in the main body, with Monte Carlo, Truncated Monte Carlo and Group Testing added. The qualitative findings hold.

G Extended Related Work

The closest related works, **model explanation**, also aim to provide principled measures of training data impact on the performance of an ML model. These techniques include Shapley value, influence functions (Koh & Liang, 2017; Koh et al., 2019) and Representer Point (Yeh et al., 2018). TracIn (Pruthi et al., 2020) and CosIn (Hammoudeh & Lowd) are two other recent proposals for measuring the influence of a training sample based on how it impacts the loss function during training. Existing model explanation approaches fall short. They either focus on marginal influence on the whole dataset (Koh & Liang, 2017; Koh et al., 2019); make strong assumptions (e.g., convex loss functions) that disallow their use with DNNs (Koh & Liang, 2017; Koh et al., 2019; Basu et al., 2020); cannot reason about data sources or sets of training samples (Pruthi et al., 2020; Hammoudeh & Lowd; Chakarov et al., 2016); or subsample training data but focus on a single inclusion probability and thus cannot explain results in all scenarios(Feldman & Zhang, 2020; Ilyas et al., 2022; Zhang et al., 2021).

Figure 25: **Correct-prediction queries and their results, where each row shows two queries and each query starts with the query input followed by images selected.**

Of those, the works focused on efficiently estimating the **Shapley value (SV)** are closest. SV which was proposed in game theory (Shapley, 1953), has been widely studied in recent years, with applications to data valuation in ML (Jia et al., 2019; Ghorbani & Zou, 2019) and feature selection/understanding (Lipovetsky & Conklin, 2001; Jethani et al., 2021; Lundberg & Lee, 2017; Covert & Lee, 2021), as well as extensions such as D-Shapley (Ghorbani et al., 2020) which generalizes SV by modeling the dataset as a random variable. The SV is notoriously costly to measure, and efficient estimators are the focus of much recent work since the early permutation algorithm with L_2 error bounds in $O(N^2 \log(N))$ samples (Maleki et al., 2014) under bounded utility. Recently proposed estimators also reduce SV estimation to regression problems, although different than ours (Lundberg & Lee, 2017; Covert & Lee, 2021; Jethani et al., 2021; Kwon & Zou, 2022). Beta-Shapley (Kwon & Zou, 2022) is closest, as it generalizes data Shapley to consider different weightings based on subset size, which coincides with our AME under p-induced distribution (e.g., Beta-Shapley is AME when $p \sim$ beta distribution). We also study this approach, and alternative distributions (including a truncated uniform which we found a bit better in practice). None of these works study the sparse setting, or provide efficient L_2 bounds for estimators in this setting. This may stem from their focus on explaining *features*, in smaller settings than we consider for training data and in which sparsity may be less natural. The most comparable work is that of Jia et.al (Jia et al., 2019), which provides multiple algorithms, including for the sparse, monotonous utility setting. Their approach uses compressive sensing, which is closely related to our LASSO based approach, and yields an $O(N \log \log N)$ rate. We significantly improve on this rate with an $O(k \log N)$ estimator, much more efficient in the sparse ($k \ll N$) regime. Other estimation strategies, use-cases, and relaxations have been recently proposed for the SV. (Mitchell et al., 2022) study efficient sampling of permutations. However, each "permutation sample" requires N utility evaluations, and is thus incompatible with our setting. (Frye et al., 2020) focus on feature SV's, and leverage the causal structure over features (on the data distribution) to decide value assignments. This is done by reweighing entire permutations of the features (e.g., giving more weight to permutations where a given feature appears early). In contrast, the AME reweighs the utility of different subsets of the data points, based on their size, which is orthogonal, and seems more amenable to sparse estimators. (Harris et al., 2022) extend the SV axioms to consider the joint effect of multiple players, prove the uniqueness of the solution, and derive its formulation. It measures the contribution of all subsets up to a specified subset size, which is however incompatible to our setup due to large N (e.g., there will be $O(N^2) \approx 2.5$ billion contribution terms just for subset size of two on our CIFAR-10 datasets). In contrast, the AME studies the contribution for individual players, as does the regular SV, and focuses on efficient estimation in the sparse regime. Our hierarchical setting considers fixed sources, and not all possible subsets.

Figure 26: **Wrong-prediction queries and their results, where each row shows two queries and each query starts with the query input followed by images selected.**

Another related body of work is the work focused on defending against data poisoning attacks. Reject On Negative Impact (RONI) (Barreno et al., 2010; Baracaldo et al., 2017) describes an algorithm that measures the LOO effect of data points on subsets of the data. However, RONI is sample-inefficient and the paper does not prescribe a subset distribution to be used. As we explained previously, the choice of subset distribution can impact precision and recall. Another line of work (Tran et al., 2018; Hayase et al., 2021; Chen et al., 2018; Tang et al., 2021; Shen et al., 2016) uses outlier detection to identify poisoned data. These are not query aware and thus can select benign outliers. Other approaches (Doan et al., 2020; Tang et al., 2021; Veldanda et al., 2020) assume the availability of clean data, or make strong assumptions about the model, e.g., assuming that linear models (Jagielski et al., 2018), or the attack, e.g., assuming that the attack is a source-agnostic trigger attacks (Gao et al., 2019), a trigger attack with small norm or size (Chou et al., 2018; Wang et al., 2019; Udeshi et al., 2019), or a clean-label attack (Peri et al., 2020). None of these approaches can generalize across techniques.

Our work is also related to the existing literature on **data cleaning and management**. However, data cleaning approaches are not query driven, and must rely on other assumptions. As a result, many approaches depend on user provided integrity constraints (Chu et al., 2013b;a) or outlier detection (Maletic & Marcus, 2000; Hellerstein, 2008). As a result these approaches cannot always identify poisoned data (Koh et al., 2018) and might also identify benign outliers. Recent approaches (Krishnan et al., 2017; Dolatshah et al., 2018), have also used another downstream DNN for data cleaning. However, these approaches assume that corrupt data has an influence on test set performance, an assumption that may not hold in scenarios such as data poisoning. Finally, Rain (Wu et al., 2020) is a recent query-driven proposal that proposes using influence function to explain SQL query results. While Rain shares similar goals, we focus on DNNs, a different use case.

Finally, our work leverages, and builds on, a large body of existing work from different fields. First, our work is related to the **causal inference** literature if we regard the inclusion of a data source as a treatment, from which AME is inspired. For instance, (Kang et al., 2007; Imbens & Rubin, 2015) focuses on single treatment AME (i.e. only one source). Multiple treatments are introduced in (Egami & Imai, 2018; Dasgupta et al., 2015; Hainmueller et al., 2014), though their quantity uses different population distribution than ours, and different estimation techniques. In the computer science filed, Sunlight (Lecuyer et al., 2015) uses a similar approach but with only one sampling probability and without knockoff procedure. Second, **sparse recovery** studies efficient algorithms to recover a sparse signal from high dimensional observations. We leverage LASSO (Lecué et al., 2018) –with properties related to those of compressive sensing (Candès et al., 2006)– and knockoffs (Candes et al., 2016). Other approaches to the important factor selection problem exist, such as the analysis of

variance (ANOVA) (Bondell & Reich, 2009; Egami & Imai, 2018; Post & Bondell, 2013) used in (Egami & Imai, 2018), but we think LASSO is better suited to our use-case due to its scalability guarantees. Third, our goal is related to **group testing** (WikipediaContributors, 2021; Aldridge et al., 2019) as discussed in §1, and studying if and how group testing ideas could improve our technique is an interesting avenue for future work.

name	npoi	prec_dis	rec_dis	prec	rec	AI
ImageNet	5	100	80.0	100	100	5.500744
CIFAR10-50	50	96.4	54.0	2.1	100	0.788898
CIFAR10-50*	100	100	54.0	6.2	100	1.682893
CIFAR10-50*	101	100	54.5	7.9	100	1.744635
CIFAR10-50*	102	100	53.9	10.0	100	1.832493
CIFAR10-50*	103	100	54.4	10.4	100	1.826089
CIFAR10-50*	104	100	54.8	100	99.0	8.742186
CIFAR10-50*	105	100	54.3	100	99.0	8.916950
CIFAR10-50*	106	100	54.7	100	98.1	8.926177
CIFAR10-50*	107	100	54.2	100	98.1	9.054615
CIFAR10-50*	108	100	54.6	4.1	100	0.716676
CIFAR10-50*	109	100	54.1	100	98.2	9.374960
CIFAR10-50*	110	100	54.5	100	98.2	9.583294
CIFAR10-50*	120	100	54.2	100	97.5	11.441413
CIFAR10-50*	130	100	54.6	100	97.7	12.932239
CIFAR10-50*	140	100	54.3	100	98.6	13.600518
CIFAR10-50*	150	100	54.7	100	98.7	15.521987
CIFAR10-20	20	11.1	60.0	0.9	100	0.700744
CIFAR10-20*	40	27.9	60.0	1.7	100	0.793756
CIFAR10-20*	60	46.1	58.3	2.5	98.3	0.894113
CIFAR10-20*	80	52.8	58.8	3.4	98.8	0.970892
CIFAR10-20*	100	74.7	59.0	4.3	99.0	1.093217
CIFAR10-20*	120	80.5	58.3	5.2	99.2	1.253165
CIFAR10-20*	140	87.2	58.6	6.0	98.6	1.419778
CIFAR10-20*	160	89.5	58.8	7.0	98.8	1.527789
CIFAR10-20*	180	93.0	58.9	7.9	98.3	1.746322
CIFAR10-20*	200	97.5	59.0	9.2	98.5	1.884684
CIFAR10-20*	300	100	58.7	99.6	92.7	4.334019
CIFAR10-20*	400	100	58.8	100	93.0	7.996566
CIFAR10-20*	500	100	58.8	100	92.2	10.964163
CIFAR10-20*	600	100	58.7	100	92.3	14.168781
CIFAR10-20*	700	100	58.7	100	92.0	17.779293
CIFAR10-20*	800	100	58.8	100	91.6	20.268145
CIFAR10-20*	900	100	58.8	100	91.7	24.644904
CIFAR10-20*	1000	100	58.8	100	91.5	26.829417
EMNIST	10	100	80.0	100	100	84.367889

Table 8: **Full result of SCAn, where "*" indicates the dataset is supplemented, "_dis" indicates the distance-based score.**

name	lambda	precision	recall
CIFAR10-20	3000.0	4.8	58.8
CIFAR10-20	30.0	18.0	58.8
CIFAR10-20	12.0	68.5	58.8
CIFAR10-20	11.9	68.9	58.8
CIFAR10-20	11.0	64.9	58.8
CIFAR10-20	3.0	1.0	58.8
CIFAR10-20	0.003	0.6	58.8
CIFAR10-50	3000.0	95.2	54.4
CIFAR10-50	30.0	99.9	54.4
CIFAR10-50	11.9	97.8	54.4
CIFAR10-50	3.0	0.7	54.4
CIFAR10-50	0.003	0.7	54.4
EMNIST	3000.0	100	78.9
EMNIST	3.0	100	78.9
EMNIST	0.003	100	78.9
ImageNet	3000.0	80.8	78.0
ImageNet	3.0	80.8	78.0
ImageNet	0.003	80.8	78.0
ImageNet	0.0001	84.9	78.0
ImageNet	4e-05	85.9	78.0
ImageNet	3e-05	85.9	78.0
ImageNet	2e-05	84.9	78.0

Table 9: **Full result of Representer Point**