
Stochastic Amortization: A Unified Approach to Accelerate Feature and Data Attribution

Ian Covert*
Stanford University
icovert@stanford.edu

Chanwoo Kim*
University of Washington
chanwkim@uw.edu

Su-In Lee†
University of Washington
suinlee@uw.edu

James Zou†
Stanford University
jamesz@stanford.edu

Tatsunori Hashimoto†
Stanford University
thashim@stanford.edu

Abstract

Many tasks in explainable machine learning, such as data valuation and feature attribution, perform expensive computation for each data point and are intractable for large datasets. These methods require efficient approximations, and although amortizing the process by learning a network to directly predict the desired output is a promising solution, training such models with exact labels is often infeasible. We therefore explore training amortized models with noisy labels, and we find that this is inexpensive and surprisingly effective. Through theoretical analysis of the label noise and experiments with various models and datasets, we show that this approach tolerates high noise levels and significantly accelerates several feature attribution and data valuation methods, often yielding an order of magnitude speedup over existing approaches.

1 Introduction

Many tasks in explainable machine learning (XML) perform some form of costly computation for every data point in a dataset. For example, common tasks include assessing individual data points' impact on a model's accuracy [33], or quantifying each input feature's influence on individual model predictions [68]. Many of these techniques are prohibitively expensive: in particular, those with game-theoretic formulations have exponential complexity in the number of features or data points, making their exact calculation intractable [89, 4].

Accelerating these methods is therefore a topic of great practical importance. This has been addressed primarily with Monte Carlo approximations [96, 16, 74], which are faster than brute-force calculations but can be slow to converge and impractical for large datasets. Alternatively, a promising idea is to *amortize* the computation, or to approximate each data point's output with a learned model, typically a deep neural network [2]. For example, in the feature attribution context, we can train an *explainer model* to predict Shapley values that describe how each feature affects a classifier's prediction [50].

There are several reasons why amortization is appealing, particularly with neural networks: similar data points often have similar outputs, pretrained networks extract relevant features and can be efficiently fine-tuned, and if the combined training and inference time is low then amortization can be faster than computing the object of interest (e.g., data valuation scores) for the entire dataset. However, it is not obvious how to train such amortized models, because standard supervised learning requires a dataset of ground truth labels that can be intractable to generate. Our goal here is therefore to explore efficiently training amortized models when exact labels are costly. Our main insight is that

*Equal contribution. †Equal advising.

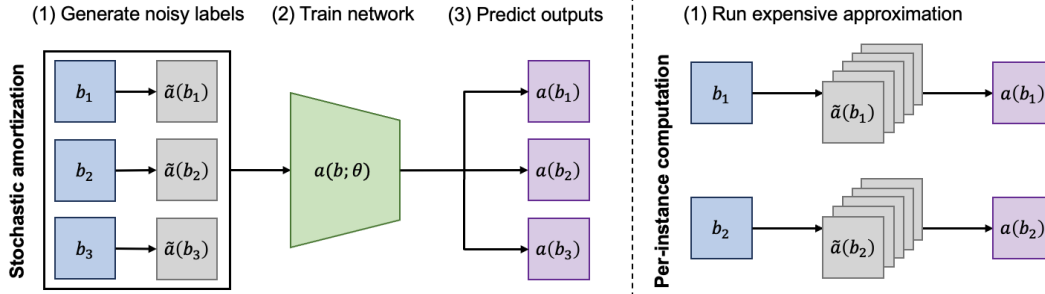


Figure 1: Diagram of stochastic amortization. Left: using a dataset with noisy labels $\tilde{a}(b)$ (e.g., images and data valuation estimates), we can train an amortized model that accurately estimates the true outputs $a(b)$ (e.g., data valuation scores). Right: the default approach of running an expensive approximation algorithm for each example (e.g., a Monte Carlo estimator with many samples [33]).

amortization is surprisingly effective with noisy labels: we train with inexpensive estimates of the true labels, and we find that this is theoretically justified when the estimates are unbiased.

We refer to this approach as *stochastic amortization* (Figure 1), and we find that it is applicable to a variety of XML tasks. In particular, we show that it is effective for feature attribution with Shapley values [68], Banzhaf values [11] and LIME [83]; for several formulations of data valuation [33, 35, 60, 103]; and to data attribution with datamodels [47]. Our experiments demonstrate significant speedups for several of these tasks: we find that amortizing across an entire dataset with noisy labels is often more efficient than current per-example approximations, especially for large datasets, and that amortized feature and data attribution models generalize well to unseen examples.

Our contributions in this work are the following:

- We present the idea of stochastic amortization, or training amortized models with noisy labels. We analyze the role of noise from the label generation process and show theoretically that it is sufficient to use unbiased estimates of the ground truth labels (Section 3). We find that non-zero bias in the labels leads to learning an incorrect function, but that variance in the labels plays a more benign role of slowing optimization.
- We identify a range of applications for stochastic amortization in XML. Our theory only requires unbiased estimation of the task’s true labels, and we find that such estimators exist for several feature attribution and data valuation methods (Section 4).
- Experimentally, we test multiple estimators for Shapley value feature attributions and find that amortization works when the labels are unbiased (Section 5). We also verify that amortization is effective for Banzhaf values and LIME. For data valuation, we apply amortization to Data Shapley and show that it allows us to scale this approach to larger datasets than in previous works.
- Throughout our experiments, we also analyze the scaling behavior with respect to the amount of training data and the quality of the noisy labels. In general, we find that amortization is more efficient than per-example computation when the training set used for amortization contains at least a moderate number of data points (e.g., $>1\text{K}$ for data valuation).

Overall, our work shows the potential for accelerating many computationally intensive XML tasks with the same simple approach: amortizing with noisy, unbiased labels.

2 Background

We first introduce the basic idea of amortization in ML, which we discuss in a general setting before considering any specific XML tasks. Consider a scenario where we repeat similar computation for a large number of data points. We represent this general setup with a context variable $b \in \mathcal{B}$ and a per-context output $a(b) \in \mathcal{A}$. For example, these can be an image and its data valuation score, or an image and its feature attributions. Amortization can be used with arbitrary domains, but we assume Euclidean spaces, or $\mathcal{A} \subseteq \mathbb{R}^m$ and $\mathcal{B} \subseteq \mathbb{R}^d$, because many XML tasks involve real-valued outputs.

The computation performed for each context $b \in \mathcal{B}$ can be arbitrary as well. In some cases $a(b)$ is an intractable expectation, in which case we would typically approximate it with a Monte Carlo estimator [96, 33]. In other situations it is the solution to an optimization problem [91, 2], in which case we can define $a(b)$ via a parametric objective $h : \mathcal{A} \times \mathcal{B} \mapsto \mathbb{R}$:

$$a(b) \equiv \arg \min_{a' \in \mathcal{A}} h(a'; b). \quad (1)$$

We do not require a specific formulation for $a(b)$ in this work, but we will see that both the expectation and optimization setups provide useful perspectives on our proposal of training with noisy labels.

In situations with repeated computation, our two options are generally to (i) perform the computation separately for each $b \in \mathcal{B}$, or (ii) amortize the computation by predicting the output with a learned model. We typically implement the latter with a neural network $a(b; \theta)$, and our goal is to train it such that $a(b; \theta) \approx a(b)$. In training these models, the main challenge occurs when the per-instance computation is costly: specifically, it is not obvious how to train $a(b; \theta)$ without a dataset of ground truth solutions $a(b)$, which can be too slow to generate for many XML methods [68, 33]. We address this challenge in Section 3, where we prove that amortization tolerates training with noisy labels.

2.1 Related work

The general idea of amortized computation captures many tasks arising in physics, engineering, control and ML [2]. For example, amortization is prominent in variational inference [56, 82], meta learning [38, 29, 80] and reinforcement learning [45, 64, 39]. In the XML context, many recent works have explored amortization to accelerate costly per-datapoint calculations [12, 108, 49, 50, 18, 19]. Some of these works are reviewed by [13], and we offer a detailed overview in Appendix A.

For feature attributions, two works propose predicting Shapley values by training with a custom weighted least squares loss [50, 18]; our simpler approach can use any unbiased estimator and resembles a standard regression task. Two other works suggest modeling feature attributions with supervised learning [87, 14]; these recommend training with exact or high-quality labels, whereas we recognize the potential to use noisy labels that can be generated orders of magnitude faster. Concurrently, Zhang et al. [110] proposed training with a custom estimator for Shapley value feature attributions; our work is similar but derives stochastic amortization algorithms for a range of settings, including the use of any unbiased estimator (Section 3) and usage for various XML tasks including data valuation (Section 4).

For data valuation, two works consider predicting data valuation scores with supervised learning [35, 36], but these also use near-exact labels that limit the applicability to large datasets. Concurrently, Li and Yu [65] propose a family of data valuation estimators and a learning-based approach analogous to [50] but for data valuation; our approach uses a simpler training loss that works with any unbiased estimator, and unlike [65] its memory usage does not scale with the dataset size. Separately, another line of work focuses on accelerating the model retraining step underlying most data valuation methods [57, 104, 107], and these are complementary to our approach. Finally, while there are works that accelerate data attribution with datamodels [78, 27], we are not aware of any that use amortization.²

More broadly, our proposal to train amortized models with noisy labels can be viewed as a version of stochastic optimization, or training with noisy gradients. This fundamental idea is widely used in machine learning [5], but to our knowledge we are the first to study the broad applicability of noisy labels for accelerating diverse XML tasks.

3 Stochastic Amortization

We now discuss how to efficiently train amortized models with noisy labels. Following Section 2, we present this as a general approach before focusing on a specific XML task. One natural idea is to treat amortization like a standard supervised learning problem: we can parameterize a model $a(b; \theta)$, adopt a distribution $p(b)$ over the context variable, and then train our model with the following objective,

$$\mathcal{L}_{\text{reg}}(\theta) = \mathbb{E} \left[\|a(b; \theta) - a(b)\|^2 \right]. \quad (2)$$

²Datamodels [47] performs data attribution by fitting a linear regression model, but this is not amortization because the model cannot predict attributions for new data points (see Appendix B).

This approach is called *regression-based amortization* [2] because it reduces the problem to a simple regression task. The challenge is that this approach cannot be used when we lack a large dataset of exact labels $a(b)$, which is common for computationally intensive XML methods (Section 4).

A relaxation of this idea is to train the model with inexact labels (see Figure 1). We assume that these are generated by a noisy oracle $\tilde{a}(b)$, which is characterized by a distribution of outputs for each context $b \in \mathcal{B}$. For example, the noisy oracle could be a statistical estimator of a data valuation score [33]. With this, we can train the model using a modified version of Eq. (2), where we consider the loss in expectation over both $p(b)$ and the noisy labels $\tilde{a}(b)$:

$$\tilde{\mathcal{L}}_{\text{reg}}(\theta) = \mathbb{E} \left[\|a(b; \theta) - \tilde{a}(b)\|^2 \right]. \quad (3)$$

It is not immediately obvious when this approach is worthwhile: if the noisy oracle is too inaccurate we will learn the wrong function, so it is important to choose the oracle carefully. We find that there are two properties of the oracle that matter, and these relate to its systematic error and noise level, or more intuitively its bias and variance. We denote these quantities as follows for a specific value b ,

$$\text{B}(\tilde{a} | b) = \|a(b) - \mathbb{E}[\tilde{a}(b) | b]\|^2, \quad \text{N}(\tilde{a} | b) = \mathbb{E} \left[\|\tilde{a}(b) - \mathbb{E}[\tilde{a}(b) | b]\|^2 | b \right],$$

and based on these we can also define the global measures $\text{B}(\tilde{a}) \equiv \mathbb{E}_p[\text{B}(\tilde{a} | b)]$ and $\text{N}(\tilde{a}) \equiv \mathbb{E}_p[\text{N}(\tilde{a} | b)]$ for the distribution over context variables $p(b)$.³ These terms are useful because they reveal a relationship between the two amortization objectives. In general, the objectives are related by the following two-sided bound (see proof in Appendix D):

$$\left(\sqrt{\tilde{\mathcal{L}}_{\text{reg}}(\theta) - \text{N}(\tilde{a})} - \sqrt{\text{B}(\tilde{a})} \right)^2 \leq \mathcal{L}_{\text{reg}}(\theta) \leq \left(\sqrt{\tilde{\mathcal{L}}_{\text{reg}}(\theta) - \text{N}(\tilde{a})} + \sqrt{\text{B}(\tilde{a})} \right)^2. \quad (4)$$

This relationship shows that reducing $\tilde{\mathcal{L}}_{\text{reg}}(\theta)$ towards its minimum value $\text{N}(\tilde{a})$ is similar to training with $\mathcal{L}_{\text{reg}}(\theta)$, only with a disconnect introduced by the bias $\text{B}(\tilde{a})$. The bias represents a source of irreducible error, because in the limit $\tilde{\mathcal{L}}_{\text{reg}}(\theta) - \text{N}(\tilde{a}) \rightarrow 0$ we have $\mathcal{L}_{\text{reg}}(\theta) = \text{B}(\tilde{a})$. On the other hand, when $\text{B}(\tilde{a}) = 0$ we can see that $\mathcal{L}_{\text{reg}}(\theta) = \tilde{\mathcal{L}}_{\text{reg}}(\theta) - \text{N}(\tilde{a})$, which means that training with the noisy loss is equivalent and will recover the correct function asymptotically. This last equality is easy to see given an unbiased noisy oracle $\tilde{a}(b)$, but the more general relationship in Eq. (4) emphasizes how non-zero bias can be problematic and lead to learning an incorrect function.

Aside from the bias, the variance plays a role as well, not in determining the function we learn but in making the model’s optimization unstable or require more noisy labels. To illustrate the role of variance, we present a theoretical result considering the simplest case of a linear model $a(b; \theta)$ trained with SGD, which shows that high label noise slows convergence (see proof in Appendix D).

Theorem 1. *Consider a noisy oracle $\tilde{a}(b)$ that satisfies $\mathbb{E}[\tilde{a}(b) | b] = \tilde{\theta}b$ with parameters $\tilde{\theta} \in \mathbb{R}^{m \times d}$ such that $\|\tilde{\theta}\|_F \leq D$. Given a distribution $p(b)$, define the norm-weighted distribution $q(b) \propto p(b) \cdot \|b\|^2$ and the terms $\Sigma_p \equiv \mathbb{E}_p[bb^\top]$ and $\Sigma_q \equiv \mathbb{E}_q[bb^\top]$. If we train a linear model $a(\theta; b) = \theta b$ with the noisy objective $\tilde{\mathcal{L}}_{\text{reg}}(\theta)$ using SGD with step size $\eta_t = \frac{2}{\alpha(t+1)}$, then the averaged iterate $\bar{\theta}_T = \sum_{t=1}^T \frac{2t}{T(T+1)} \theta_t$ at step T satisfies*

$$\mathbb{E}[\tilde{\mathcal{L}}_{\text{reg}}(\bar{\theta}_T)] - \text{N}(\tilde{a}) \leq \frac{4 \text{Tr}(\Sigma_p) (\text{N}_q(\tilde{a}) + 4\lambda_{\max}(\Sigma_q)D^2)}{\lambda_{\min}(\Sigma_p)(T+1)},$$

where $\text{N}_q(\tilde{a}) \equiv \mathbb{E}_q[\text{N}(\tilde{a} | b)]$ is the noisy oracle’s norm-weighted variance, and $\lambda_{\max}(\cdot)$, $\lambda_{\min}(\cdot)$ are the maximum and minimum eigenvalues.

The bound in Theorem 1 shows that noise slows convergence by its presence in the numerator, and interestingly, it appears in the form of a weighted version $\text{N}_q(\tilde{a})$ that puts more weight on values with large norm $\|b\|$; this is a consequence of assuming a linear model, but we expect the general conclusion of label variance slowing convergence to hold even for neural networks. As a corollary, we can see that the rate in Theorem 1 applies directly to $\mathcal{L}_{\text{reg}}(\theta)$ when the noisy oracle is unbiased (see Appendix D). We note that very high noise levels can in principle prevent effective optimization, and this can be mitigated by either reducing the noisy oracle’s variance or taking more steps T .

³ $\text{N}(\tilde{a} | b)$ is equal to the trace of the conditional covariance $\text{Cov}(\tilde{a} | b)$, and $\text{N}(\tilde{a}) = \text{Tr}(\mathbb{E}[\text{Cov}(\tilde{a} | b)])$.

Overall, our analysis shows that amortization with noisy labels is possible, although perhaps more difficult to optimize than training with exact labels. We next show that unbiased estimates are available for many XML tasks (Section 4), and we later find that this form of amortization is consistently effective with the noise levels observed in practice (Section 5), even providing better accuracy than per-example estimation in compute-matched comparisons. As a shorthand, we refer to training with the noisy objective $\tilde{\mathcal{L}}_{\text{reg}}(\theta)$ in Eq. (3) as *stochastic amortization*.

4 Applications to Explainable ML

We now consider XML tasks that can be accelerated with stochastic amortization. Rather than using generic variables $b \in \mathcal{B}$ and $a(b) \in \mathcal{A}$, this section uses an input variable $x \in \mathcal{X}$, a response variable $y \in \mathcal{Y}$, and a model f or a measure of its performance. As we describe below, each application of stochastic amortization is instantiated by a noisy oracle that generates labels for the given task.

4.1 Shapley value feature attribution

One of the most common tasks in XML is feature attribution, which aims to quantify each feature’s influence on an individual prediction. The Shapley value has gained popularity because of its origins in game theory [89, 68], and like many feature attribution methods is based on querying the model while removing different feature sets [17]. Given a model f and input x that consists of d separate features $x = (x_1, \dots, x_d)$, we assume that we can calculate the prediction $f(x_S) \in \mathbb{R}$ for any feature set $S \subseteq [d]$.⁴ With this setup, the Shapley values $\phi_i(x) \in \mathbb{R}$ for each feature $i \in [d]$ are defined as:

$$\phi_i(x) = \frac{1}{d} \sum_{S \subseteq [d] \setminus \{i\}} \binom{d-1}{|S|}^{-1} (f(x_{S \cup \{i\}}) - f(x_S)). \quad (5)$$

These scores satisfy several desirable properties [89], but they are impractical to calculate due to the exponential summation over feature subsets. Our goal is therefore to learn an amortized model $\phi(x; \theta) \in \mathbb{R}^d$ to directly predict feature attribution scores, and for this we require a noisy oracle.

Many recent works have studied efficient Shapley value estimation [10], and we first consider noisy oracles derived from Eq. (5), which defines the attribution as the feature’s expected marginal contribution. There are several unbiased statistical estimators that rely on sampling feature subsets or permutations [6, 96, 77, 74, 58], and following Section 3 we can use any of these for stochastic amortization. Our experiments use the classic permutation sampling estimator [96, 74], which approximates the values $\phi_i(x)$ as an expectation across feature orderings. We defer the precise definition of this noisy oracle to Appendix E, along with the other estimators used in our experiments.

Next, we also consider noisy oracles derived from an optimization perspective on the Shapley value. A famous result from Charnes et al. [7] shows that the Shapley values are the solution to the following problem (with abuse of notation we discard the solution’s intercept term),

$$\phi(x) = \arg \min_{a \in \mathbb{R}^{d+1}} \sum_{S \subseteq [d]} \mu(S) \left(f(x_S) - a_0 - \sum_{i \in S} a_i \right)^2, \quad (6)$$

where we use a least squares weighting kernel defined as $\mu^{-1}(S) = \binom{d}{|S|} |S| (d - |S|)$. Several works have proposed approximating Shapley values by solving this problem with sampled subsets, either using projected gradient descent [92] or analytic solutions [68, 16]. Among these, we use KernelSHAP [68] and SGD-Shapley [92] as noisy oracles in our experiments. The first is an M-estimator whose bias shrinks as the number of sampled subsets grows [99], so we expect it to lead to effective amortization; the latter has been shown to have non-negligible bias [10], so our theory in Section 3 suggests that it should lead to learning an incorrect function when used for amortization.

4.2 Alternative feature attributions

Next, we consider two alternative feature attribution methods: Banzhaf values [4, 11] and LIME [83]. These are closely related to Shapley values and are similarly intractable [26, 40], but we find that they offer statistical estimators that can be used for stochastic amortization.

⁴There are many ways to do so [17], e.g., we can set features to their mean or use masked self-attention.

First, Banzhaf values assign the following scores to each feature for a prediction $f(x)$ [4]:

$$\phi_i(x) = \frac{1}{2^{d-1}} \sum_{S \subseteq [d] \setminus \{i\}} (f(x_{S \cup \{i\}}) - f(x_S)). \quad (7)$$

These differ from Shapley values only in their choice of weighting function, and they admit a range of similar statistical estimators. One option is the MSR estimator from [103], which is unbiased and re-uses all model evaluations for each feature attribution estimate. We adopt this as a noisy oracle in our experiments (see a precise definition in Appendix E), but several other options are available.

Second, LIME defines its attribution scores $\phi_i(x)$ as the solution to the following optimization problem, given a weighting kernel $\pi(S)$ and penalty term Ω [83]:⁵

$$\arg \min_{a \in \mathbb{R}^{d+1}} \sum_{S \subseteq [d]} \pi(S) \left(f(x_S) - a_0 - \sum_{i \in S} a_i \right)^2 + \Omega(a). \quad (8)$$

As our noisy oracle for LIME, we use the popular approach of solving the above problem for subsets sampled according to $\pi(S)$. Similar to KernelSHAP [68], this is an M-estimator whose bias shrinks to zero as the sample size grows [99], so we expect it to lead to successful amortization.

Aside from these methods, other costly feature interpretation methods rely on unbiased statistical estimators and can be amortized in a similar fashion [98, 62, 32]. We leave further investigation of these methods to future work.

4.3 Data valuation

Next, data valuation aims to quantify how much each training example affects a model’s accuracy. We consider labeled examples $z = (x, y)$ and a training dataset $\mathcal{D} = \{z_i\}_{i=1}^n$, and we analyze each data point’s value by fitting models to subsampled datasets $\mathcal{D}_T \subseteq \mathcal{D}$ with $T \subseteq [n]$ and calculating a measure of the model’s performance $v(\mathcal{D}_T) \in \mathbb{R}$ (e.g., its 0-1 accuracy). This general approach was introduced by Ghorbani and Zou [33], who defined the Data Shapley scores $\psi(z_i) \in \mathbb{R}$ as follows:

$$\psi(z_i) = \frac{1}{n} \sum_{T \subseteq [n] \setminus \{i\}} \binom{n-1}{|T|}^{-1} (v(\mathcal{D}_T \cup \{z_i\}) - v(\mathcal{D}_T)). \quad (9)$$

Subsequent work generalized the approach in different ways, which we briefly summarize before considering amortization. For example, Wang and Jia [103] used the Banzhaf value rather than Shapley value, and Kwon and Zou [60] considered the case of arbitrary semivalues [75]. These correspond to adopting a different weighting over subsets in Eq. (9), and the general case can be written as follows for a normalized weighting function $w(k)$:⁶

$$\psi(z_i) = \sum_{T \subseteq [n] \setminus \{i\}} w(|T|) (v(\mathcal{D}_T \cup \{z_i\}) - v(\mathcal{D}_T)). \quad (10)$$

Next, another extension is the case of *distributional data valuation*. Ghorbani et al. [35] incorporate an expectation over the original dataset \mathcal{D} , which they show leads to well defined scores even for data points $z = (x, y)$ outside the training set. Given a distribution over datasets of size $|\mathcal{D}| = n - 1$ and a weighting function $w(k)$, this version defines the score $\psi(z) \in \mathbb{R}$ for arbitrary z as follows:

$$\psi(z) = \mathbb{E}_{\mathcal{D}} \sum_{T \subseteq [n-1]} w(|T|) (v(\mathcal{D}_T \cup \{z\}) - v(\mathcal{D}_T)). \quad (11)$$

When using any of these methods in practice, the scores are difficult to calculate due to the intractable expectation across datasets. However, a crucial property they share is that they can all be estimated in an unbiased fashion, and these estimates can therefore be used as noisy labels for stochastic amortization. We focus on Data Shapley and Distributional Data Shapley in our experiments [33, 35], and we use the Monte Carlo estimator from Ghorbani and Zou [33] as a noisy oracle (see the precise

⁵Following [17], we only consider the version of LIME with a binary interpretable representation.

⁶For this to be a valid expectation, semivalues require that $\sum_{k=0}^{n-1} \binom{n-1}{k} w(k) = 1$. Note that Data Shapley adopts $w(k) = \binom{n-1}{k}^{-1}/n$ and Data Banzhaf adopts $w(k) = 1/2^{n-1}$.

definition in [Appendix E](#)). Doing so allows us to train an amortized valuation model $\psi(z; \theta) \in \mathbb{R}$ that accelerates valuation within our training dataset, and that can also be applied to external data, e.g., when selecting informative new data points for active learning [\[36\]](#).

Finally, [Appendix B](#) discusses amortization for the datamodels data attribution technique [\[47\]](#). This method measures how much each training data point $z_i \in \mathcal{D}$ affects the prediction for an inference example $x \in \mathcal{X}$, and we show that the scores are equivalent to a simple expectation that can be estimated in an unbiased fashion. The datamodels scores can therefore be amortized by adopting these estimates as noisy labels, but we leave further investigation of this approach to future work.

5 Experiments

Our experiments apply stochastic amortization to several of the tasks discussed in [Section 4](#). We consider both feature attribution and data valuation, for image and tabular datasets, and using multiple architectures for our amortized models, including fully-connected networks (FCNs), ResNets [\[43\]](#) and ViTs [\[24\]](#). Full details are provided in [Appendix F](#), including our exact models and hyperparameters.

Our goal in each experiment is to perform feature attribution or data valuation for an entire dataset, and to compare the accuracy of stochastic amortization to running existing estimators on each point. We adopt a noisy oracle for each task (e.g., a Monte Carlo estimator of data valuation scores), we then fit an amortized network with one noisy label per training example, and our evaluation focuses on the accuracy of the amortized predictions relative to the ground truth. Our ground truth is obtained by running the noisy oracle to near-convergence for a large number of samples: for example, we run KernelSHAP [\[68\]](#) for feature attribution with 1M samples, and the TMC estimator [\[33\]](#) for data valuation with 10K samples. We test amortization when using different numbers of training examples, and for both training and unseen external data to evaluate the model’s generalization. We find that amortization often denoises and strongly improves upon the noisy labels, leading to a significant accuracy improvement for the same computational budget.

5.1 Feature attribution

We first consider Shapley value feature attributions. This task offers a diverse set of noisy oracles, and we consider three options: KernelSHAP [\[68\]](#), permutation sampling [\[74\]](#) and SGD-Shapley [\[92\]](#). Among these, our theory from [Section 3](#) suggests that the first two will be effective for amortization, while the third may not because it is not an unbiased estimator. We follow the setup from [\[18\]](#) and implement our amortized network with a pretrained ViT-B architecture [\[24\]](#), and we use the ImageNet dataset [\[46\]](#) with 224×224 images partitioned into 196 patches of size 14×14 .

As a first result, [Figure 2](#) compares our training targets to the amortized model’s predictions. The predicted attributions are significantly more accurate than the labels, even for the noisiest setting with just 512 KernelSHAP samples. [Figure 3](#) (left) quantifies the improvement from amortization, and we observe similar results for both KernelSHAP and permutation sampling (see [Appendix G](#)): in both cases the error is significantly lower than that of the noisy labels, and it remains lower and improves as the labels become more accurate. To contextualize our amortized model’s accuracy, we find that the error is similar to that of running KernelSHAP for 10-40K samples, even though our labels use an order of magnitude fewer samples (see [Appendix G](#)). We also find that the model generalizes to external data points (see [Appendix G](#)). In addition, we

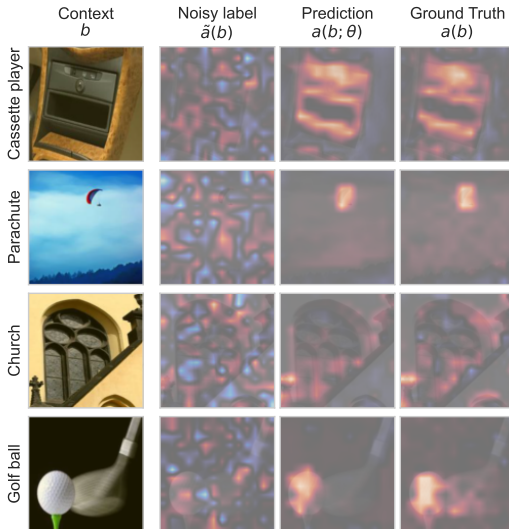


Figure 2: Stochastic amortization for Shapley value feature attributions. We compare the predicted attributions to the noisy labels and ground truth, which are generated using KernelSHAP with 512 and 1M samples, respectively.

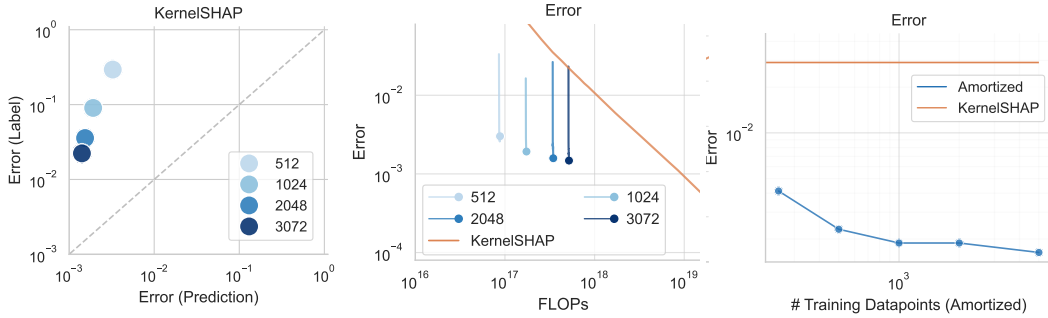


Figure 3: Amortized Shapley value feature attributions using KernelSHAP as a noisy oracle. Left: squared error relative to the ground truth attributions when using noisy labels with different numbers of samples (different noise levels). Center: estimation error as a function of FLOPs, where KernelSHAP incurs FLOPs via classifier predictions used to estimate the attributions, and amortization incurs additional FLOPs from training (training appears as a vertical line because the FLOPs are relatively low, and endpoints represent results from the final epoch). Right: estimation error with different training dataset sizes given equivalent compute per data point (matched by using fewer KernelSHAP samples when generating noisy labels for amortization and allowing up to 50 epochs of training).

show results for SGD-Shapley, where we confirm that it leads to poor amortization results due to its non-negligible bias (see Appendix G).

Next, we investigate the compute trade-off between calculating attributions separately (e.g., with KernelSHAP) and using amortization. Figure 3 (center-right) shows two results regarding this tradeoff. First, we measure the error as a function of FLOPs, where we account for the cost of generating labels and training the amortized model (see Appendix F). The FLOPs incurred by training are negligible compared to the KernelSHAP estimates, and we find that stopping at any time to fit an amortized model yields significantly better estimates. This suggests that amortization is an inexpensive final denoising step, regardless of how much compute was used for the noisy estimates.

Second, we test the effectiveness of amortization for different dataset sizes. We match the compute between the two scenarios, using 2440 KernelSHAP samples for per-example computation⁷ and 2257 for amortization to account for the cost of training. This compute-matched comparison shows that amortization achieves lower estimation error for datasets ranging from 250-10K data points (Figure 3 right); it becomes more effective as the dataset grows, but it is useful even for small datasets.

Finally, Appendix G shows a comparison between stochastic amortization and FastSHAP [50, 18], an existing approach to amortized Shapley value estimation. We observe similar estimation accuracy in compute-matched comparisons, and find that both methods are significantly more accurate than per-example estimation (similar to Figure 3 center). Appendix G also show results for amortizing Banzhaf values and LIME: we find that amortization is more difficult for these methods due to the inconsistent scale of attributions between inputs, but we nonetheless observe an improvement in our amortized estimates versus the noisy labels.

5.2 Data valuation

Next, we consider data valuation with Data Shapley. For our noisy oracle, we obtain training labels by running the TMC estimator with different numbers of samples [33]. We first test our approach with the adult census and MiniBooNE particle physics datasets [25, 84], and following prior work we conduct experiments using versions of each dataset with different numbers of data points [53]. Our valuation model must predict scores for each example $z = (x, y)$, so we train FCNs that output scores for all classes and use only the relevant output for each data point.

As a first result, Figure 4 (left-center) shows the estimation accuracy for the MiniBooNE dataset when using 1K and 10K data points. The noisy estimates converge as we use more Monte Carlo samples, but we see that the amortized estimates are always more accurate in terms of both squared error and correlation with the ground truth. The improvement is largest for the noisiest estimates, where the amortized predictions have correlation >0.9 when using only 50 samples. Amortization

⁷This is the default number of samples for 196 features in the official repository: <https://github.com/shap/shap>.

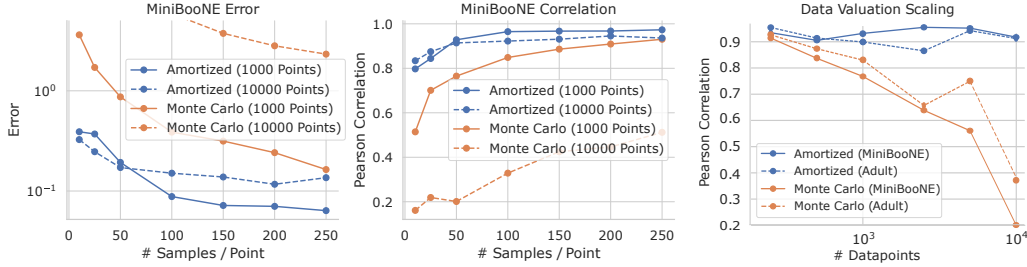


Figure 4: Amortized data valuation accuracy for tabular datasets. Left: mean squared error relative to the ground truth for the MiniBooNE dataset, normalized so that the mean valuation score has error equal to 1 (for 1K and 10K data points). The x-axis indicates how many Monte Carlo samples were used for each data point. Center: Pearson correlation with the ground truth for the MiniBooNE dataset (for 1K and 10K data points). Right: estimation accuracy for the MiniBooNE and adult census datasets as a function of dataset size (250 to 10K data points); we use 50 Monte Carlo samples per data point for all results and show the Pearson correlation with the ground truth.

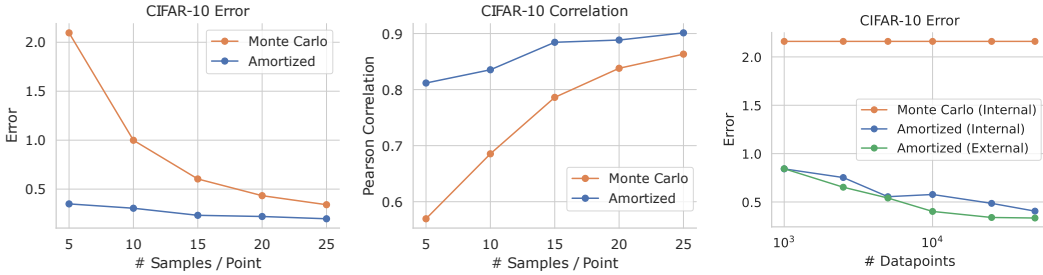


Figure 5: Distributional data valuation for CIFAR-10. Left: estimation error when using different numbers of samples for the noisy label estimates. Center: Pearson correlation with the ground truth for different numbers of noisy samples. Right: estimation error as a function of dataset size, where all results use 5 Monte Carlo samples per data points; we compare the error for amortized estimates on internal (training) and external (unseen) data points, demonstrating strong generalization.

is more beneficial for the 10K dataset, which suggests that training with more noisy labels can be a substitute for high-quality labels. [Appendix G](#) shows similar results with the adult census dataset.

Next, [Figure 4](#) (right) considers the role of training dataset size for the estimates with 50 Monte Carlo samples. For both the adult census and MiniBooNE datasets, we see that the benefits of amortization are small with 250 data points but grow as we approach 10K data points. This number of samples is enough to maintain 0.9 correlation with the ground truth when using amortization, whereas the raw estimates become increasingly inaccurate. Stochastic amortization is therefore promising to scale data valuation beyond previous works, which typically focus on <1K data points [\[34, 60, 103\]](#).

5.3 Distributional data valuation

Finally, we consider Distributional Data Shapley [\[35\]](#), which is similar to the previous experiments but defines valuation scores even for points outside the training dataset; this allows us to test the generalization to unseen data. We use the CIFAR-10 dataset [\[59\]](#), which contains 50K training examples, and for our valuation model we train a ResNet-18 that outputs scores for each class.

Similar to the previous experiments, [Figure 5](#) (left-center) evaluates the estimation accuracy for different numbers of Monte Carlo samples. We observe that the distributional scores converge faster because we use a smaller maximum dataset cardinality (see [Appendix F](#)), but that amortization still provides a significant benefit. The improvement is largest for the noisiest estimates, which in this case use just 5 samples: amortization achieves correlation 0.81 with the ground truth, versus 0.58 for the Monte Carlo estimates. The improvement is consistent across several measures of accuracy, including squared error, Pearson correlation and Spearman correlation (see [Appendix G](#)).

Next, we study generalization and the role of dataset size. We focus on the noisiest estimates with 5 Monte Carlo samples, because this low-sample regime is most relevant for larger datasets with millions of examples. We train the valuation model with different portions of the 50K training set, and we measure the estimation accuracy for both internal and unseen external data. Figure 5 (right) shows large improvements in squared error with as few as 1K data points, and we also observe improvement in correlation when we use at least 5K data points (10% of the dataset, see Appendix G). We additionally observe a small generalization gap, suggesting that the valuation model can be trained with a subset of the data and reliably applied to unseen examples.

Lastly, we test the usage of amortized valuation scores in downstream tasks. Following [53], the tasks we consider are identifying mislabeled examples and improving the model by removing low-value examples. These results are shown in Appendix G, and we find that the amortized estimates identify mislabeled examples more reliably than Monte Carlo estimates, and that filtering the dataset based on our estimates leads to improved performance.



6 Conclusion

This work explored the idea of stochastic amortization, or training amortized models with noisy labels. Our main finding is that fast, noisy supervision provides substantial compute and accuracy gains over existing XML approximations. This approach makes several feature attribution and data valuation methods more practical for large datasets and real-time applications, and it may have broader applications to amortization beyond XML [2]. Our proposal has certain limitations, including that stochastic amortization may become ineffective with sufficiently high noise levels, and that it is difficult to know a priori how much compute is necessary for label generation to achieve a desired error level in the amortized predictions.

Our work suggests multiple directions for future research. One direction is to study the trade-off between using a larger number of noisy labels or a smaller number of more accurate labels, which is a key difference from prior work that uses near-exact labels for amortization [14]. Other directions include scaling to datasets with millions of examples to test the limits of noisy supervision, leveraging more sophisticated data valuation estimators [106], using alternative model retraining primitives [57, 63, 37], and exploring amortization for other methods like datamodels (discussed in Appendix B).

Code

We provide two repositories to reproduce each our results:

-  **Feature attribution** <https://github.com/chanwkimlab/amortized-attribution>
-  **Data valuation** <https://github.com/iancovert/amortized-valuation>

Acknowledgements

The authors thank Yongchan Kwon for helpful discussions and advice on using OpenDataVal. We also thank Mukund Sudarshan and Neil Jethani for early conversations about amortizing data valuation. Chanwoo Kim and Su-In Lee were supported by the National Science Foundation (CAREER DBI-1552309 and DBI-1759487) and the National Institutes of Health (R35 GM 128638 and R01 AG061132).

References

- [1] Samira Abnar and Willem Zuidema. Quantifying attention flow in transformers. *arXiv preprint arXiv:2005.00928*, 2020.
- [2] Brandon Amos. Tutorial on amortized optimization for learning to optimize over continuous domains. *arXiv preprint arXiv:2202.00665*, 2022.

- [3] Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. In *International Conference on Learning Representations*, 2018.
- [4] John F Banzhaf. Weighted voting doesn’t work: a mathematical analysis. *Rutgers Law Review*, 19:317, 1964.
- [5] Sébastien Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.
- [6] Javier Castro, Daniel Gómez, and Juan Tejada. Polynomial calculation of the Shapley value based on sampling. *Computers & Operations Research*, 36(5):1726–1730, 2009.
- [7] A Charnes, B Golany, M Keane, and J Rousseau. Extremal principle solutions of games in characteristic function form: core, Chebychev and Shapley value generalizations. In *Econometrics of Planning and Efficiency*, pages 123–133. Springer, 1988.
- [8] Aditya Chattopadhyay, Stewart Slocum, Benjamin D Haeffele, Rene Vidal, and Donald Geman. Interpretable by design: Learning predictors by composing interpretable queries. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.
- [9] Aditya Chattopadhyay, Kwan Ho Ryan Chan, Benjamin D Haeffele, Donald Geman, and René Vidal. Variational information pursuit for interpretable predictions. *arXiv preprint arXiv:2302.02876*, 2023.
- [10] Hugh Chen, Ian C Covert, Scott M Lundberg, and Su-In Lee. Algorithms to estimate Shapley value feature attributions. *arXiv preprint arXiv:2207.07605*, 2022.
- [11] Jianbo Chen and Michael Jordan. LS-Tree: Model interpretation when the data are linguistic. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3454–3461, 2020.
- [12] Jianbo Chen, Le Song, Martin J Wainwright, and Michael I Jordan. Learning to explain: An information-theoretic perspective on model interpretation. *arXiv preprint arXiv:1802.07814*, 2018.
- [13] Yu-Neng Chuang, Guanchu Wang, Fan Yang, Zirui Liu, Xuanting Cai, Mengnan Du, and Xia Hu. Efficient XAI techniques: A taxonomic survey. *arXiv preprint arXiv:2302.03225*, 2023.
- [14] Yu-Neng Chuang, Guanchu Wang, Fan Yang, Quan Zhou, Pushkar Tripathi, Xuanting Cai, and Xia Hu. CoRTX: Contrastive framework for real-time explanation. *arXiv preprint arXiv:2303.02794*, 2023.
- [15] R Dennis Cook and Sanford Weisberg. Characterizations of an empirical influence function for detecting influential cases in regression. *Technometrics*, 22(4):495–508, 1980.
- [16] Ian Covert and Su-In Lee. Improving KernelSHAP: Practical Shapley value estimation using linear regression. In *International Conference on Artificial Intelligence and Statistics*, pages 3457–3465. PMLR, 2021.
- [17] Ian Covert, Scott Lundberg, and Su-In Lee. Explaining by removing: A unified framework for model explanation. *Journal of Machine Learning Research*, 22(209):1–90, 2021.
- [18] Ian Covert, Chanwoo Kim, and Su-In Lee. Learning to estimate Shapley values with vision transformers. *arXiv preprint arXiv:2206.05282*, 2022.
- [19] Ian Covert, Wei Qiu, Mingyu Lu, Nayoony Kim, Nathan White, and Su-In Lee. Learning to maximize mutual information for dynamic feature selection. *arXiv preprint arXiv:2301.00557*, 2023.
- [20] Ian C Covert, Scott Lundberg, and Su-In Lee. Shapley feature utility. In *Machine Learning in Computational Biology*, 2019.
- [21] Piotr Dabkowski and Yarin Gal. Real time image saliency for black box classifiers. In *Advances in Neural Information Processing Systems*, pages 6967–6976, 2017.

- [22] Ofer Dekel, Ran Gilad-Bachrach, Ohad Shamir, and Lin Xiao. Optimal distributed online prediction using mini-batches. *Journal of Machine Learning Research*, 13(1), 2012.
- [23] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE, 2009.
- [24] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
- [25] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL <http://archive.ics.uci.edu/ml>.
- [26] Pradeep Dubey and Lloyd S Shapley. Mathematical properties of the Banzhaf power index. *Mathematics of Operations Research*, 4(2):99–131, 1979.
- [27] Logan Engstrom, Axel Feldmann, and Aleksander Madry. DsDm: Model-aware dataset selection with datamodels. *arXiv preprint arXiv:2401.12926*, 2024.
- [28] Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. *Advances in Neural Information Processing Systems*, 33: 2881–2891, 2020.
- [29] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.
- [30] Ruth Fong, Mandela Patrick, and Andrea Vedaldi. Understanding deep networks via extremal perturbations and smooth masks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2950–2958, 2019.
- [31] Ruth C Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3429–3437, 2017.
- [32] Fabian Fumagalli, Maximilian Muschalik, Patrick Kolpaczki, Eyke Hüllermeier, and Barbara Hammer. SHAP-IQ: Unified approximation of any-order Shapley interactions. *arXiv preprint arXiv:2303.01179*, 2023.
- [33] Amirata Ghorbani and James Zou. Data Shapley: Equitable valuation of data for machine learning. In *International Conference on Machine Learning*, pages 2242–2251. PMLR, 2019.
- [34] Amirata Ghorbani, Abubakar Abid, and James Zou. Interpretation of neural networks is fragile. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3681–3688, 2019.
- [35] Amirata Ghorbani, Michael Kim, and James Zou. A distributional framework for data valuation. In *International Conference on Machine Learning*, pages 3535–3544. PMLR, 2020.
- [36] Amirata Ghorbani, James Zou, and Andre Esteva. Data Shapley valuation for efficient batch active learning. In *2022 56th Asilomar Conference on Signals, Systems, and Computers*, pages 1456–1462. IEEE, 2022.
- [37] Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, et al. Studying large language model generalization with influence functions. *arXiv preprint arXiv:2308.03296*, 2023.
- [38] David Ha, Andrew M Dai, and Quoc V Le. HyperNetworks. In *International Conference on Learning Representations*, 2016.
- [39] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

- [40] Peter L Hammer and Ron Holzman. Approximations of pseudo-boolean functions; applications to game theory. *Zeitschrift für Operations Research*, 36(1):3–21, 1992.
- [41] Elad Hazan et al. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2016.
- [42] He He, Paul Mineiro, and Nikos Karampatziakis. Active information acquisition. *arXiv preprint arXiv:1602.02181*, 2016.
- [43] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.
- [44] Weijie He, Xiaohao Mao, Chao Ma, Yu Huang, José Miguel Hernández-Lobato, and Ting Chen. BSODA: a bipartite scalable framework for online disease diagnosis. In *Proceedings of the ACM Web Conference 2022*, pages 2511–2521, 2022.
- [45] Nicolas Heess, Gregory Wayne, David Silver, Timothy Lillicrap, Tom Erez, and Yuval Tassa. Learning continuous control policies by stochastic value gradients. *Advances in Neural Information Processing Systems*, 28, 2015.
- [46] Jeremy Howard and Sylvain Gugger. FastAI: A layered API for deep learning. *Information*, 11(2):108, 2020.
- [47] Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. Datamodels: Predicting predictions from training data. *arXiv preprint arXiv:2202.00622*, 2022.
- [48] Saachi Jain, Hadi Salman, Eric Wong, Pengchuan Zhang, Vibhav Vineet, Sai Vemprala, and Aleksander Madry. Missingness bias in model debugging. In *International Conference on Learning Representations*, 2021.
- [49] Neil Jethani, Mukund Sudarshan, Yindalon Aphinyanaphongs, and Rajesh Ranganath. Have we learned to explain?: How interpretability methods can learn to encode predictions in their interpretations. In *International Conference on Artificial Intelligence and Statistics*, pages 1459–1467. PMLR, 2021.
- [50] Neil Jethani, Mukund Sudarshan, Ian Connick Covert, Su-In Lee, and Rajesh Ranganath. FastSHAP: Real-time Shapley value estimation. In *International Conference on Learning Representations*, 2021.
- [51] Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nezihe Merve Gurel, Bo Li, Ce Zhang, Costas J Spanos, and Dawn Song. Efficient task-specific data valuation for nearest neighbor algorithms. *arXiv preprint arXiv:1908.08619*, 2019.
- [52] Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nick Hynes, Nezihe Merve Gürel, Bo Li, Ce Zhang, Dawn Song, and Costas J Spanos. Towards efficient data valuation based on the Shapley value. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1167–1176. PMLR, 2019.
- [53] Kevin Fu Jiang, Weixin Liang, James Zou, and Yongchan Kwon. OpenDataVal: a unified benchmark for data valuation. *arXiv preprint arXiv:2306.10577*, 2023.
- [54] Hoang Anh Just, Feiyang Kang, Jiachen T Wang, Yi Zeng, Myeongseob Ko, Ming Jin, and Ruoxi Jia. Lava: Data valuation without pre-specified learning algorithms. *arXiv preprint arXiv:2305.00054*, 2023.
- [55] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [56] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- [57] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR, 2017.

- [58] Patrick Kolpaczki, Viktor Bengs, Maximilian Muschalik, and Eyke Hüllermeier. Approximating the shapley value without marginal contributions. *arXiv preprint arXiv:2302.00736*, 2023.
- [59] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.
- [60] Yongchan Kwon and James Zou. Beta Shapley: a unified and noise-reduced data valuation framework for machine learning. *arXiv preprint arXiv:2110.14049*, 2021.
- [61] Yongchan Kwon and James Zou. Data-OOB: Out-of-bag estimate as a simple and efficient data value. *arXiv preprint arXiv:2304.07718*, 2023.
- [62] Yongchan Kwon and James Y Zou. WeightedSHAP: analyzing and improving Shapley based feature attributions. *Advances in Neural Information Processing Systems*, 35:34363–34376, 2022.
- [63] Yongchan Kwon, Eric Wu, Kevin Wu, and James Zou. DataInf: Efficiently estimating data influence in lora-tuned llms and diffusion models. *arXiv preprint arXiv:2310.00902*, 2023.
- [64] Sergey Levine and Vladlen Koltun. Guided policy search. In *International Conference on Machine Learning*, pages 1–9. PMLR, 2013.
- [65] Weida Li and Yaoliang Yu. Faster approximation of probabilistic and distributional values via least squares. In *International Conference on Learning Representations*, 2024.
- [66] Chris Lin, Ian Covert, and Su-In Lee. On the robustness of removal-based feature attributions. *arXiv preprint arXiv:2306.07462*, 2023.
- [67] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.
- [68] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pages 4765–4774, 2017.
- [69] Scott M. Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M. Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. From local explanations to global understanding with explainable AI for trees. *Nature Machine Intelligence*, 2(1): 2522–5839, 2020.
- [70] Chao Ma, Sebastian Tschiatschek, Konstantina Palla, Jose Miguel Hernandez-Lobato, Sebastian Nowozin, and Cheng Zhang. EDDI: Efficient dynamic discovery of high-value information with partial VAE. In *International Conference on Machine Learning*, pages 4234–4243. PMLR, 2019.
- [71] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.
- [72] Divyat Mahajan, Chenhao Tan, and Amit Sharma. Preserving causal constraints in counterfactual explanations for machine learning classifiers. *arXiv preprint arXiv:1912.03277*, 2019.
- [73] Jean-Luc Marichal and Pierre Mathonet. Weighted Banzhaf power and interaction indexes through weighted approximations of games. *European Journal of Operational Research*, 211(2):352–358, 2011.
- [74] Rory Mitchell, Joshua Cooper, Eibe Frank, and Geoffrey Holmes. Sampling permutations for Shapley value estimation. *arXiv preprint arXiv:2104.12199*, 2021.
- [75] Dov Monderer, Dov Samet, et al. Variations on the Shapley value. *Handbook of Game Theory*, 3:2055–2076, 2002.
- [76] Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4): 1574–1609, 2009.

- [77] Ramin Okhrati and Aldo Lipani. A multilinear sampling algorithm to estimate shapley values. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 7992–7999. IEEE, 2021.
- [78] Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. TRAK: Attributing model behavior at scale. *arXiv preprint arXiv:2303.14186*, 2023.
- [79] Kaare Brandt Petersen, Michael Syskind Pedersen, et al. The matrix cookbook. *Technical University of Denmark*, 7(15):510, 2008.
- [80] Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-learning with implicit gradients. *Advances in Neural Information Processing Systems*, 32, 2019.
- [81] Samrudhdi B Rangrej and James J Clark. A probabilistic hard attention model for sequentially observed scenes. *arXiv preprint arXiv:2111.07534*, 2021.
- [82] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pages 1278–1286. PMLR, 2014.
- [83] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should I trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144, 2016.
- [84] BP Roe, HJ Yand, J Zhu, Y Lui, I Stancu, et al. Boosted decision trees, an alternative to artificial neural networks. *Nucl. Instrum. Meth. A*, 543:577–584, 2005.
- [85] Nikunj Saunshi, Arushi Gupta, Mark Braverman, and Sanjeev Arora. Understanding influence functions and datamodels via harmonic analysis. *arXiv preprint arXiv:2210.01072*, 2022.
- [86] Patrick Schwab and Walter Karlen. CXPlain: Causal explanations for model interpretation under uncertainty. In *Advances in Neural Information Processing Systems*, pages 10220–10230, 2019.
- [87] Robert Schwarzenberg, Nils Feldhus, and Sebastian Möller. Efficient explanations from empirical explainers. *arXiv preprint arXiv:2103.15429*, 2021.
- [88] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 618–626, 2017.
- [89] Lloyd S Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2(28): 307–317, 1953.
- [90] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International Conference on Machine Learning*, pages 3145–3153. PMLR, 2017.
- [91] Rui Shu. Amortized optimization, 2017. URL <http://ruishu.io/2017/11/07/amortized-optimization/>.
- [92] Grah Simon and Thouvenot Vincent. A projected stochastic gradient algorithm for estimating Shapley value applied in attribute importance. In *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, pages 97–115. Springer, 2020.
- [93] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.
- [94] Sumedha Singla, Brian Pollack, Junxiang Chen, and Kayhan Batmanghelich. Explanation by progressive exaggeration. *arXiv preprint arXiv:1911.00483*, 2019.
- [95] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

- [96] Erik Štrumbelj and Igor Kononenko. An efficient explanation of individual classifications using game theory. *Journal of Machine Learning Research*, 11:1–18, 2010.
- [97] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3319–3328. JMLR. org, 2017.
- [98] Mukund Sundararajan, Kedar Dhamdhere, and Ashish Agarwal. The Shapley Taylor interaction index. In *International Conference on Machine Learning*, pages 9259–9268. PMLR, 2020.
- [99] Aad W Van der Vaart. *Asymptotic Statistics*, volume 3. Cambridge University Press, 2000.
- [100] Sahil Verma, John Dickerson, and Keegan Hines. Counterfactual explanations for machine learning: A review. *arXiv preprint arXiv:2010.10596*, 2020.
- [101] Sahil Verma, Keegan Hines, and John P Dickerson. Amortized generation of sequential algorithmic recourses for black-box models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8512–8519, 2022.
- [102] Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harvard Journal of Law & Technology*, 31:841, 2017.
- [103] Tianhao Wang and Ruoxi Jia. Data Banzhaf: A data valuation framework with maximal robustness to learning stochasticity. *arXiv preprint arXiv:2205.15466*, 2022.
- [104] Tianhao Wang, Yu Yang, and Ruoxi Jia. Improving cooperative game theory-based data valuation via data utility learning. *arXiv preprint arXiv:2107.06336*, 2021.
- [105] Brian Williamson and Jean Feng. Efficient nonparametric statistical inference on population feature importance using Shapley values. In *International Conference on Machine Learning*, pages 10282–10291. PMLR, 2020.
- [106] Mengmeng Wu, Ruoxi Jia, Changle Lin, Wei Huang, and Xiangyu Chang. Variance reduced Shapley value estimation for trustworthy data valuation. *Computers & Operations Research*, page 106305, 2023.
- [107] Zhaoxuan Wu, Yao Shu, and Bryan Kian Hsiang Low. Davinz: Data valuation using deep neural networks at initialization. In *International Conference on Machine Learning*, pages 24150–24176. PMLR, 2022.
- [108] Jinsung Yoon, James Jordon, and Mihaela van der Schaar. INVASE: Instance-wise variable selection using neural networks. In *International Conference on Learning Representations*, 2018.
- [109] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833. Springer, 2014.
- [110] Borui Zhang, Baotong Tian, Wenzhao Zheng, Jie Zhou, and Jiwen Lu. Exploring unified perspective for fast Shapley value estimation. *arXiv preprint arXiv:2311.01010*, 2023.
- [111] Jianming Zhang, Sarah Adel Bargal, Zhe Lin, Jonathan Brandt, Xiaohui Shen, and Stan Sclaroff. Top-down neural attention by excitation backprop. *International Journal of Computer Vision*, 126(10):1084–1102, 2018.

A Extended Related Work

This section provides a more detailed review of amortization in XML. Many XML tasks involve analyzing individual data points, e.g., to determine the most important features or concepts in a model, or a data point’s influence on the model’s accuracy or an individual prediction. Among these methods, many require expensive per-example algorithms, and a trend in recent years has been to amortize this computation using deep learning. We discuss these works below, grouping them into several main XML problems that they address.

Within this discussion, we differentiate works not only based on their goal (e.g., feature attribution or data valuation) but also based on how they calculate the object of interest. For those that perform per-example computation, we distinguish between methods that solve a parametric optimization problem of the form $a(b) = \arg \min_{a'} h(a'; b)$ (e.g., with gradient descent) from those that exploit an analytic solution, either by directly calculating the result or using a Monte Carlo approximation. For those that use amortization, we distinguish those that use regression-based amortization with $\mathcal{L}_{\text{reg}}(\theta)$ from those that perform objective-based amortization with $\mathcal{L}_{\text{obj}}(\theta)$ (see [Appendix C](#) for more details on objective-based amortization [2]).

[Table 1](#) summarizes the methods we discuss, including the various tasks they solve, the context and output domains (\mathcal{A}, \mathcal{B}) for each problem, and the computational approach. This perspective highlights the significant role of amortization in XML, and it also reveals opportunities for new applications, some of which we discuss in [Section 4](#). Some of the same works are discussed by Chuang et al. [13], but we cover a wider range of methods, and we adopt the framing of Amos [2] by outlining the various parametric optimization problems and different amortization approaches.

Feature attribution. These methods aim to quantify the importance of a model’s input features, typically for individual predictions (such methods are called *local* rather than *global* feature attributions, see Lundberg et al. [69]). The context is therefore a data example $x \in \mathcal{X}$, and the output is a vector of attributions in \mathbb{R}^d when we have d features. Feature attribution algorithms can be efficient to calculate, particularly when they are based on simple propagation rules [93, 97, 90, 95, 88, 111, 3] or a transformer’s attention values [1]. However, another family of approaches are based on feature removal [17], and querying the model with many feature subsets is typically less efficient.

Among the feature removal-based methods, one of the most famous is Shapley value feature attributions [68, 89]. Two related methods are LIME [83] and Banzhaf values [11, 4], see our discussion of their similarity in [Section 4](#). Many works have focused on efficiently calculating these attributions, because their computational complexity is exponential in the number of features. As discussed by Chen et al. [10], there are many stochastic estimators derived from an analytic expression for the attributions [6, 96, 77, 16, 74, 103, 58], and these are typically unbiased Monte Carlo estimators. Others are based on solving an optimization problem, for example with either M-estimation [83, 68, 105, 16] or gradient descent [92].

Lastly, there are also methods based on amortization, which offer the possibility of real-time feature attribution without a significant loss in accuracy. Two works consider objective-based amortization, which avoids the need for labels during training [50, 18]. Others consider regression-based amortization [86, 87, 14], and these recommend using high-quality labels; for example, Chuang et al. [14] use exact labels when possible. Our perspective is most similar to the concurrent work by Zhang et al. [110], in that we highlight the potential for regression-based amortization with inexpensive, noisy labels; but our work is more general in that we highlight the potential to use noisy labels from any unbiased estimator, and the applicability of this approach to other XML tasks like data valuation.

Instance-wise feature selection. A related set of methods select the most important features for individual predictions. The context in this case is an input $x \in \mathcal{X}$, and the output is a set of feature indices $S \in \mathcal{P}([d])$ where we use $\mathcal{P}(\cdot)$ to denote the power set. The optimal subset can be defined either based on the prediction $f(x_S)$ or $f(x_{[d] \setminus S})$ for a specific class [31], or based on the deviation from the original prediction $f(x)$ [12]. Among these methods, some solve the underlying optimization problem on a per-input basis [31, 30], and others solve it using objective-based amortization [21, 12, 108, 49]. These methods all require differentiating through discrete subsets, so they employ various continuous relaxations (e.g., the Concrete distribution [71]), as well as other tricks to constrain or penalize the subset size. These methods are sometimes thought of as attribution methods due to the continuous relaxations [31, 30], but we describe them as feature

Table 1: Summary of amortized methods in XML.

Problem	Context	Domain	Analytic	Per-example optimization	$\mathcal{L}_{\text{reg}}(\theta)$	$\mathcal{L}_{\text{obj}}(\theta)$
Shapley value attribution			Štrumbelj and Kononenko [96] Okhrati and Lipani [77] Mitchell et al. [74]	Lundberg and Lee [68] Simon and Vincent [92] Covert and Lee [16]	Schwarzenberg et al. [87] Chuang et al. [14] Zhang et al. [110]	Jethani et al. [50] Covert et al. [18]
Leave-one-out attribution	$x \in \mathcal{X}$	\mathbb{R}^d	Zeiler and Fergus [109]	Schwab and Karlen [86]		
LIME attribution				Ribeiro et al. [83]		
Banzhaf value attribution			Covert et al. [17]	Chen and Jordan [11]		
Instance-wise selection	$x \in \mathcal{X}$	$\mathcal{P}([d])$				Chen et al. [12] Yoon et al. [108] Jethani et al. [49]
Image masking				Fong and Vedaldi [31] Fong et al. [30]		Dabkowski and Gal [21]
Dynamic feature selection	$x_S \in \mathcal{X} \times \mathcal{P}([d])$	$[d]$		Ma et al. [70]	He et al. [42]	Chattopadhyay et al. [9] Covert et al. [19]
Counterfactual explanation	$x \in \mathcal{X}$	\mathcal{X}		Wachter et al. [102]		Mahajan et al. [72] Verma et al. [101]
Data Shapley			Ghorbani and Zou [33]			Li and Yu [65]
Beta Shapley	$z_i \in \mathcal{D}$	\mathbb{R}	Kwon and Zou [60]			
Data Banzhaf			Wang and Jia [103]			
Datamodels	$(z_i, x) \in \mathcal{D} \times \mathcal{X}$	\mathbb{R}			Ghorbani et al. [35, 36]	

selection methods because the final scores for each feature are restricted to the range $[0, 1]$ and are made continuous mainly for optimization purposes.

Dynamic feature selection. Next, other works select features separately for each prediction to achieve high accuracy with a small feature acquisition budget. Each selection is made based on a partially observed input, so the context is a feature subset x_S , which we write as belonging to the domain $\mathcal{X} \times \mathcal{P}([d])$, and the output is an index $i \in [d]$. There are several ways to define the optimal selection at each step, but one common approach is to define it as the feature with maximum conditional mutual information, or $i^* = \arg \max_i I(y; x_i | x_S)$, where the response variable y and unobserved features x_i are random variables and x_S has a fixed observed value. Given access to this objective, it is trivial to solve by enumerating the possible indices; however, the mutual information is typically unavailable in practice and must be approximated. One approach is therefore to fit a proxy for the mutual information (e.g., via a generative model) and then optimize this proxy for each selection [70, 81, 8, 44]. An alternative is to learn a network that predicts the optimal selection at each step: He et al. [42] do so using imitation learning, which resembles regression-based amortization, and Chattopadhyay et al. [9] and Covert et al. [19] do so with an optimization-based view of the mutual information $I(y; x_i | x_S)$, which is objective-based amortization.

Counterfactual explanation. The goal of counterfactual explanations (also known as *recourse explanations*) is to identify small changes to an input that cause a desired change in a model’s prediction. The context is an input $x \in \mathcal{X}$, and the output is a modified version $x' \in \mathcal{X}$ that is typically not too different from the original input. This family of methods was introduced by Wachter et al. [102] and is generally framed as an optimization problem involving the prediction $f(x')$ and a measure of the perturbation strength between x' and x . Verma et al. [100] provide a review of these methods and the various choices for the optimization formulation. When computing these explanations, the most common approach is to solve the problem separately for each input, e.g., using gradient descent [102]. Other works have explored learning models that directly output the modified example x' , and these are typically implemented using objective-based amortization [72, 94, 101, 13].

Data valuation. The goal of data valuation is to assign scores to each training example that represent their contribution to a model’s performance. The context is therefore a labeled training example $z \in \mathcal{D}$, and the output is a real-valued score. These methods are typically not defined via an optimization problem, but instead as a measure of the example’s expected impact on performance across a distribution of preceding datasets [33, 35, 60, 103]. Existing methods therefore rely on analytic expressions for the valuation scores, which require Monte Carlo approximation because they are intractable. Two works have considered predicting data valuation scores given a dataset of near-exact estimates [35, 36], which is regression-based amortization. Our work is similar, but we use inexpensive estimates to reduce the cost of amortization and scale more efficiently to larger datasets. Concurrently, Li and Yu [65] proposed a learning-based approach analogous to FastSHAP for feature attribution [50], but whose memory requirements scale with the dataset size and limit applicability to large datasets. Besides these methods, there are also data valuation approaches that can be calculated without any approximations [51, 54, 61].

Data attribution. Finally, data attribution methods aim to quantify the effect of training examples on individual predictions. The context is therefore a training example $z \in \mathcal{D}$ paired with an inference example $x \in \mathcal{X}$, and the output is a real-valued attribution score. One classic approach to this problem is influence functions [15], which use a gradient-based approximation to avoid the cost of retraining [57, 37, 63]. Another class of methods involve measuring the effect of training with subsampled datasets [28, 47]. For datamodels, the main existing approximation algorithm is based on solving a global least squares problem [47], which does not correspond to any of the computational approaches listed in Table 1; notably, it is not a form of amortization because there is no model that can predict the attribution given a new tuple (z, x) . As an alternative to the global least squares problem, TRAK calculates similar scores to datamodels using a gradient-based approximation [78, 27]. We are not aware of any work investigating amortization for datamodels, but our results in Appendix B show how to approximate the scores based on their analytic solution, and how to amortize the computation by adopting Monte Carlo estimates as noisy training labels.

B Datamodels

The datamodels technique [47] aims to quantify how much each training data point $z_i \in \mathcal{D}$ affects the prediction for an inference example $x \in \mathcal{X}$. Similar to data valuation, the inference example’s output given a training dataset \mathcal{D}_T is represented by a function $v_x(\mathcal{D}_T)$, and the attribution scores $\zeta(z_i, x) \in \mathbb{R}$ are then defined as the solution to a joint least squares problem that can be solved after training many models with different datasets, as we show below.⁸ For the inference example’s output $v_x(\mathcal{D}_T)$, Ilyas et al. [47] focus on the loss rather than the raw prediction, but our perspective can accommodate any definition of this output.

Our insight on the datamodels technique is twofold. First, we show that the datamodels scores are equal to a simple expectation and can be estimated in an unbiased fashion. Second, we show that these calculations can be amortized by using the noisy Monte Carlo estimates as training targets. Our findings rely on a slight reformulation of the datamodels scores, where we deviate from Ilyas et al. [47] by using an intercept term and a subtly different dataset distribution: our distribution is biased towards size nq for a value $q \in (0, 1)$ rather than using a fixed size, which was also considered in recent work by Saunshi et al. [85]. In the following result, we use the shorthand $\zeta(x) \equiv [\zeta(z_1, x), \dots, \zeta(z_n, x)]$ for the vector of data attributions. Given a probability $q \in (0, 1)$, we also define the weighting function $u(k) = q^k(1 - q)^{n-1-k}$. With this setup, our reformulation is the following (see the proof in Appendix D).

Proposition 1. *Given a subset distribution that includes each data point $z_i \in \mathcal{D}$ with probability $q \in (0, 1)$, the data attribution scores defined by*

$$\zeta(x) \equiv \arg \min_{a \in \mathbb{R}^{n+1}} \mathbb{E}_T \left[\left(a_0 + \sum_{i \in T} a_i - v_x(\mathcal{D}_T) \right)^2 \right]$$

can be expressed as the following expectation:

$$\zeta(z_i, x) = \sum_{T \subseteq [n] \setminus i} u(|T|) (v_x(\mathcal{D}_T \cup \{z_i\}) - v_x(\mathcal{D}_T)).$$

Perhaps surprisingly, this corresponds to a game-theoretic semivalue and reduces to the Banzhaf value when $q = 1/2$ [73]. Based on this perspective, we can estimate the score $\zeta(z_i, x)$ without solving the regression problem from Ilyas et al. [47]. For example, we can simply sample k datasets \mathcal{D}_T from the distribution with $q \in (0, 1)$, and then calculate the empirical average as follows:

$$\hat{\zeta}(z_i, x) = \frac{1}{k} \sum_{j=1}^k v_x(\mathcal{D}_{T_j} \cup \{z_i\}) - v_x(\mathcal{D}_{T_j} \setminus \{z_i\}).$$

Following Proposition 1, we have the unbiasedness property $\mathbb{E}[\hat{\zeta}(z_i, x)] = \zeta(z_i, x)$. Furthermore, rather than repeating this estimation for each attribution score, we can amortize the process by fitting an attribution model $\zeta(z_i, x; \theta)$ using the noisy estimates $\hat{\zeta}(z_i, x)$ as training targets. Our experiments do not test this approach, which we leave as a direction for future work. Implementing this requires a more complex architecture to handle two model inputs x and z , but we speculate that if trained effectively, amortization could accelerate attribution within the training set and generalize to new inference examples $x \in \mathcal{X}$ with no additional overhead.

⁸As we noted in the main text, although datamodels performs data attribution by fitting a linear regression model, it is not a form of amortization because the resulting model cannot predict attribution scores $\zeta(z, x)$ for new datapoints. The attribution scores are given by the model’s coefficients, not its predictions.

C Connection to Objective-Based Amortization

Recall that many tasks with repeated computation have an optimization view where $a(b) \equiv \arg \min_{a'} h(a'; b)$ [91, 2] (see Section 2). For completeness, this section describes an interpretation of stochastic amortization from this optimization perspective. An alternative to regression-based amortization when we have an objective $h(a'; b)$ that defines $a(b)$ is to train $a(b; \theta)$ directly with the h objective:

$$\mathcal{L}_{\text{obj}}(\theta) = \mathbb{E}[h(a(b; \theta); b)]. \quad (12)$$

This approach provides an alternative when exact labels are costly, but it can be unappealing because (i) the task may not offer a natural objective $h(a'; b)$, and (ii) minimizing $h(a'; b)$ may seem disconnected from the error $\|a(b; \theta) - a(b)\|$. We discuss these issues below and how they are related to stochastic amortization.

For issue (i), certain problems like Shapley values have a natural optimization characterization (see Section 4.1), but this is not always the case: for example, data valuation scores are framed as an expectation rather than via optimization (see Section 4.3 or [33]), so they lack an optimization view. We always have the trivial optimization perspective $h(a'; b) = \|a' - a(b)\|^2$, but this is not useful because it reduces $\mathcal{L}_{\text{obj}}(\theta)$ to $\mathcal{L}_{\text{reg}}(\theta)$ and requires the exact outputs $a(b)$. Regardless of whether the task offers a natural objective $h(a'; b)$, stochastic amortization can be viewed as defining a new objective that does not require the exact outputs: given a noisy oracle $\tilde{a}(b)$, stochastic amortization is equivalent to $\mathcal{L}_{\text{obj}}(\theta)$ with the objective $h(a'; b) = \mathbb{E}[\|a' - \tilde{a}(b)\|^2]$, and if the oracle is unbiased then the optimal predictions are $a(b; \theta) = a(b)$.

Next, issue (ii) is a concern when $h(a'; b)$ is available but lacks a clear connection to the estimation error, which is directly reflected by $\mathcal{L}_{\text{reg}}(\theta)$. The squared error is in many cases a more meaningful accuracy measure, and it is commonly used to evaluate estimation accuracy for feature attribution and data valuation [50, 103, 10]. Certain works on objective-based amortization have considered non-convex objectives $h(a'; b)$ where the exact output $a(b)$ is not well-defined [2], but we can understand the potential disconnect by focusing on a class of well behaved objectives. In particular, if we assume that $h(a'; b)$ is α -strongly convex and β -smooth in a' for all b , then the $\mathcal{L}_{\text{reg}}(\theta)$ and $\mathcal{L}_{\text{obj}}(\theta)$ objectives bound one other as follows,

$$\frac{\alpha}{2} \mathcal{L}_{\text{reg}}(\theta) \leq \mathcal{L}_{\text{obj}}(\theta) - \mathcal{L}_{\text{obj}}^* \leq \frac{\beta}{2} \mathcal{L}_{\text{reg}}(\theta), \quad (13)$$

where we define $\mathcal{L}_{\text{obj}}^* \equiv \mathbb{E}[h(a(b); b)]$ (see proof in Appendix D.1). Eq. (13) shows that by minimizing $\mathcal{L}_{\text{obj}}(\theta)$, we effectively minimize both upper and lower bounds on $\mathcal{L}_{\text{reg}}(\theta)$. However, these bounds can be loose: for example, because we have $\mathcal{L}_{\text{reg}}(\theta) \leq \frac{2}{\alpha} \mathcal{L}_{\text{obj}}(\theta)$ and the strong convexity constant α shrinks to zero for Shapley values as the number of features grows [18], optimizing $\mathcal{L}_{\text{obj}}(\theta)$ may become less effective in high dimensions. Stochastic amortization resolves this potential disconnect between $\mathcal{L}_{\text{obj}}(\theta)$ and $\mathcal{L}_{\text{reg}}(\theta)$ as follows: if we use an unbiased noisy oracle $\tilde{a}(b)$, then training with $\tilde{\mathcal{L}}_{\text{reg}}(\theta)$ is equivalent to using $\mathcal{L}_{\text{obj}}(\theta)$ with $h(a'; b) = \mathbb{E}[\|a' - \tilde{a}(b)\|^2]$, so we have $\alpha = \beta = 1$ in Eq. (13) and the regression objectives $\mathcal{L}_{\text{reg}}(\theta)$ and $\tilde{\mathcal{L}}_{\text{reg}}(\theta)$ are equal up to a constant $\mathcal{L}_{\text{obj}}^* = \mathbb{N}(\tilde{a})$.

D Proofs

This section provides proofs for our claims in the main text. [Appendix D.1](#) shows proofs for our results related to stochastic amortization in [Section 3](#) and [Appendix C](#), and then [Appendix D.2](#) shows the proof for [Proposition 1](#) related to datamodels.

D.1 Amortization proofs

We first derive the inequality in [Eq. \(4\)](#) from the main text, which is the following:

$$\left(\sqrt{\tilde{\mathcal{L}}_{\text{reg}}(\theta) - \mathbb{N}(\tilde{a})} - \sqrt{\mathbb{B}(\tilde{a})} \right)^2 \leq \mathcal{L}_{\text{reg}}(\theta) \leq \left(\sqrt{\tilde{\mathcal{L}}_{\text{reg}}(\theta) - \mathbb{N}(\tilde{a})} + \sqrt{\mathbb{B}(\tilde{a})} \right)^2.$$

Proof. We begin by decomposing a per-input version of the noisy oracle loss $\tilde{\mathcal{L}}_{\text{reg}}(b; \theta)$, which takes an expectation over the noisy oracle $\tilde{a}(b)$ for a fixed b :

$$\begin{aligned} \tilde{\mathcal{L}}_{\text{reg}}(b; \theta) &= \mathbb{E} [\|a(b; \theta) - \tilde{a}(b)\|^2 \mid b] \\ &= \|a(b; \theta) - \mathbb{E}[\tilde{a}(b) \mid b]\|^2 + \mathbb{E}[\|\tilde{a}(b) - \mathbb{E}[\tilde{a}(b) \mid b]\|^2 \mid b] \\ &= \|a(b; \theta) - \mathbb{E}[\tilde{a}(b) \mid b]\|^2 + \mathbb{N}(\tilde{a} \mid b). \end{aligned}$$

Next, we can also decompose a per-input version of the original regression loss $\mathcal{L}_{\text{reg}}(b; \theta)$ as follows using triangle inequality:

$$\begin{aligned} \mathcal{L}_{\text{reg}}(b; \theta) &= \|a(b; \theta) - a(b)\|^2 \\ &\leq (\|a(b; \theta) - \mathbb{E}[\tilde{a}(b) \mid b]\| + \|a(b) - \mathbb{E}[\tilde{a}(b) \mid b]\|)^2 \\ &= \left(\sqrt{\tilde{\mathcal{L}}_{\text{reg}}(b; \theta) - \mathbb{N}(\tilde{a} \mid b)} + \sqrt{\mathbb{B}(\tilde{a} \mid b)} \right)^2. \end{aligned}$$

Taking both sides of the inequality in expectation over $p(b)$, we arrive at the following upper bound:

$$\mathcal{L}_{\text{reg}}(\theta) \leq \mathbb{E} \left[\left(\sqrt{\tilde{\mathcal{L}}_{\text{reg}}(b; \theta) - \mathbb{N}(\tilde{a} \mid b)} + \sqrt{\mathbb{B}(\tilde{a} \mid b)} \right)^2 \right].$$

One side of the bound in [Eq. \(4\)](#) comes from developing the square and applying Cauchy-Schwarz to the cross term:

$$\begin{aligned} \mathcal{L}_{\text{reg}}(\theta) &\leq \mathbb{E} \left[\left(\sqrt{\tilde{\mathcal{L}}_{\text{reg}}(b; \theta) - \mathbb{N}(\tilde{a} \mid b)} + \sqrt{\mathbb{B}(\tilde{a} \mid b)} \right)^2 \right] \\ &= \mathbb{E} \left[\tilde{\mathcal{L}}_{\text{reg}}(b; \theta) - \mathbb{N}(\tilde{a} \mid b) \right] + \mathbb{E} [\mathbb{B}(\tilde{a} \mid b)] + 2\mathbb{E} \left[\sqrt{\mathbb{B}(\tilde{a} \mid b)} \left(\tilde{\mathcal{L}}_{\text{reg}}(b; \theta) - \mathbb{N}(\tilde{a} \mid b) \right) \right] \\ &\leq \tilde{\mathcal{L}}_{\text{reg}}(\theta) - \mathbb{N}(\tilde{a}) + \mathbb{B}(\tilde{a}) + 2\sqrt{\mathbb{B}(\tilde{a})} \left(\tilde{\mathcal{L}}_{\text{reg}}(\theta) - \mathbb{N}(\tilde{a}) \right) \\ &= \left(\sqrt{\tilde{\mathcal{L}}_{\text{reg}}(\theta) - \mathbb{N}(\tilde{a})} + \sqrt{\mathbb{B}(\tilde{a})} \right)^2. \end{aligned}$$

This proves the upper bound in [Eq. \(4\)](#). For the lower bound, we return to the decomposition of the per-input noisy oracle loss $\tilde{\mathcal{L}}_{\text{reg}}(b; \theta)$ and apply triangle inequality as follows:

$$\begin{aligned}
\tilde{\mathcal{L}}_{\text{reg}}(b; \theta) &= \|a(b; \theta) - \mathbb{E}[\tilde{a}(b) | b]\|^2 + \text{N}(\tilde{a} | b) \\
&\leq (\|a(b; \theta) - a(b)\| + \|a(b) - \mathbb{E}[\tilde{a}(b) | b]\|)^2 + \text{N}(\tilde{a} | b) \\
&= \left(\sqrt{\mathcal{L}_{\text{reg}}(b; \theta)} + \sqrt{\text{B}(\tilde{a} | b)} \right)^2 + \text{N}(\tilde{a} | b).
\end{aligned}$$

Rearranging terms, we have

$$\mathcal{L}_{\text{reg}}(b; \theta) \geq \left(\sqrt{\tilde{\mathcal{L}}_{\text{reg}}(b; \theta)} - \text{N}(\tilde{a} | b) - \sqrt{\text{B}(\tilde{a} | b)} \right)^2,$$

and applying the same logic as above with Cauchy-Schwarz, we arrive at:

$$\mathcal{L}_{\text{reg}}(\theta) \geq \left(\sqrt{\tilde{\mathcal{L}}_{\text{reg}}(\theta)} - \text{N}(\tilde{a}) - \sqrt{\text{B}(\tilde{a})} \right)^2.$$

Putting the two results together, this yields the two-sided bound in [Eq. \(4\)](#):

$$\left(\sqrt{\tilde{\mathcal{L}}_{\text{reg}}(\theta)} - \text{N}(\tilde{a}) - \sqrt{\text{B}(\tilde{a})} \right)^2 \leq \mathcal{L}_{\text{reg}}(\theta) \leq \left(\sqrt{\tilde{\mathcal{L}}_{\text{reg}}(\theta)} - \text{N}(\tilde{a}) + \sqrt{\text{B}(\tilde{a})} \right)^2.$$

□

Next, we prove our SGD result for stochastic amortization in [Theorem 1](#).

Theorem 1. Consider a noisy oracle $\tilde{a}(b)$ that satisfies $\mathbb{E}[\tilde{a}(b) | b] = \tilde{\theta}b$ with parameters $\tilde{\theta} \in \mathbb{R}^{m \times d}$ such that $\|\tilde{\theta}\|_F \leq D$. Given a distribution $p(b)$, define the norm-weighted distribution $q(b) \propto p(b) \cdot \|b\|^2$ and the terms $\Sigma_p \equiv \mathbb{E}_p[bb^\top]$ and $\Sigma_q \equiv \mathbb{E}_q[bb^\top]$. If we train a linear model $a(\theta; b) = \theta b$ with the noisy objective $\tilde{\mathcal{L}}_{\text{reg}}(\theta)$ using SGD with step size $\eta_t = \frac{2}{\alpha(t+1)}$, then the averaged iterate $\bar{\theta}_T = \sum_{t=1}^T \frac{2t}{T(T+1)} \theta_t$ at step T satisfies

$$\mathbb{E}[\tilde{\mathcal{L}}_{\text{reg}}(\bar{\theta}_T)] - \text{N}(\tilde{a}) \leq \frac{4 \text{Tr}(\Sigma_p) (\text{N}_q(\tilde{a}) + 4\lambda_{\max}(\Sigma_q)D^2)}{\lambda_{\min}(\Sigma_p)(T+1)},$$

where $\text{N}_q(\tilde{a}) \equiv \mathbb{E}_q[\text{N}(\tilde{a} | b)]$ is the noisy oracle's norm-weighted variance, and $\lambda_{\max}(\cdot)$, $\lambda_{\min}(\cdot)$ are the maximum and minimum eigenvalues.

Proof. Given our assumption about the noisy oracle, we can re-write the noisy objective $\tilde{\mathcal{L}}_{\text{reg}}(\theta)$ as follows:

$$\tilde{\mathcal{L}}_{\text{reg}}(\theta) = \mathbb{E} \left[\|(\theta - \tilde{\theta})b\|^2 \right] + \text{N}(\tilde{a}) = \text{Tr} \left((\theta - \tilde{\theta})^\top \Sigma_p (\theta - \tilde{\theta}) \right) + \text{N}(\tilde{a}).$$

It is clear that the minimizer is $\theta = \tilde{\theta}$ and that the minimum achievable error is $\tilde{\mathcal{L}}_{\text{reg}}(\tilde{\theta}) = \text{N}(\tilde{a})$. Because the objective composes a linear function with a convex function, it is convex in θ and we can analyze its convergence using a standard SGD result. Specifically, because we assume that $\|\tilde{\theta}\|_F \leq D$, we consider a projected SGD algorithm where each step is followed by a projection into the D -ball, or $\theta \leftarrow \theta \cdot \min(1, D/\|\theta\|_F)$. Note that it is common to assume a bounded solution space when proving SGD's convergence [[76](#), [22](#), [5](#)].

Before proceeding to the main convergence result, we must derive several properties of this objective and its stochastic gradients. The first is the strong convexity constant, and we begin by noting that the gradient is the following (see Section 2.5 of Petersen et al. [[79](#)] for a review of matrix derivatives):

$$\nabla \tilde{\mathcal{L}}_{\text{reg}}(\theta) = 2(\theta - \tilde{\theta})\mathbb{E}[bb^\top] = 2(\theta - \tilde{\theta})\Sigma_p.$$

For the strong convexity constant, we require a value $\alpha > 0$ such that the following is satisfied for all parameters θ, θ' :

$$\tilde{\mathcal{L}}_{\text{reg}}(\theta) \geq \tilde{\mathcal{L}}_{\text{reg}}(\theta') + \text{Tr} \left((\theta - \theta')^\top \nabla \tilde{\mathcal{L}}_{\text{reg}}(\theta') \right) + \frac{\alpha}{2} \|\theta - \theta'\|_F^2. \quad (14)$$

For the first two terms on the right side of the inequality, we can write:

$$\begin{aligned} \tilde{\mathcal{L}}_{\text{reg}}(\theta') + \text{Tr} \left((\theta - \theta')^\top \nabla \tilde{\mathcal{L}}_{\text{reg}}(\theta') \right) &= \text{N}(\tilde{a}) + \text{Tr} \left((\theta' - \tilde{\theta})\Sigma_p(\theta' - \tilde{\theta})^\top \right) + 2 \text{Tr} \left((\theta' - \tilde{\theta})\Sigma_p(\theta - \theta')^\top \right) \\ &= \text{N}(\tilde{a}) + \text{Tr} \left((\theta - \tilde{\theta})\Sigma_p(\theta - \tilde{\theta})^\top \right) - \text{Tr} \left((\theta' - \theta)\Sigma_p(\theta' - \theta)^\top \right). \end{aligned}$$

By using the minimum eigenvalue of Σ_p , we can write the following,

$$\text{Tr} \left((\theta' - \theta)\Sigma_p(\theta' - \theta)^\top \right) \geq \lambda_{\min}(\Sigma_p) \|\theta - \theta'\|_F^2,$$

and therefore satisfy Eq. (14) as follows:

$$\begin{aligned} \tilde{\mathcal{L}}_{\text{reg}}(\theta) &= \text{N}(\tilde{a}) + \text{Tr} \left((\theta - \tilde{\theta})\Sigma_p(\theta - \tilde{\theta})^\top \right) \\ &= \tilde{\mathcal{L}}_{\text{reg}}(\theta') + \text{Tr} \left((\theta - \theta')^\top \nabla \tilde{\mathcal{L}}_{\text{reg}}(\theta') \right) + \text{Tr} \left((\theta' - \theta)\Sigma_p(\theta' - \theta)^\top \right) \\ &\geq \tilde{\mathcal{L}}_{\text{reg}}(\theta') + \text{Tr} \left((\theta - \theta')^\top \nabla \tilde{\mathcal{L}}_{\text{reg}}(\theta') \right) + \lambda_{\min}(\Sigma_p) \|\theta - \theta'\|_F^2. \end{aligned}$$

We can therefore conclude that $\tilde{\mathcal{L}}_{\text{reg}}(\theta)$ is $2\lambda_{\min}(\Sigma_p)$ -strongly convex.

The next property to derive is the expected gradient norm. When running SGD, we make updates using the following stochastic gradient estimate:

$$g(\theta) = 2(\theta b - \tilde{a}(b))b^\top.$$

We require an upper bound on the stochastic gradient norm, which can be written as follows:

$$\mathbb{E} [\|g(\theta)\|_F^2] = \mathbb{E} [\|g(\theta) - \nabla \tilde{\mathcal{L}}_{\text{reg}}(\theta)\|_F^2] + \|\nabla \tilde{\mathcal{L}}_{\text{reg}}(\theta)\|_F^2. \quad (15)$$

For the first term in Eq. (15), which represents the gradient variance, we have:

$$\mathbb{E} [\|g(\theta) - \nabla \tilde{\mathcal{L}}_{\text{reg}}(\theta)\|_F^2] = 4\mathbb{E} [\|(\theta b - \tilde{a}(b))b^\top - (\theta - \tilde{\theta})\Sigma_p\|_F^2].$$

To find a simple expression for this term, we first consider the expectation over the distribution $\tilde{a}(b)$ with fixed $b \in \mathcal{B}$, which isolates label variation due to the noisy oracle:

$$\begin{aligned} \mathbb{E}_{\tilde{a}|b} [\|(\theta b - \tilde{a}(b))b^\top - (\theta - \tilde{\theta})\Sigma_p\|_F^2] &= \mathbb{E}_{\tilde{a}|b} [\|(\tilde{a}(b) - \tilde{\theta}b)b^\top\|_F^2] + \|(\theta - \tilde{\theta})(bb^\top - \Sigma_p)\|_F^2 \\ &= \text{N}(\tilde{a} | b) \cdot \|b\|^2 + \|(\theta - \tilde{\theta})(bb^\top - \Sigma_p)\|_F^2. \end{aligned}$$

When we take this in expectation over $p(b)$, the first term can be understood as a norm-weighted average of the noisy oracle's conditional variance:

$$\begin{aligned}
\mathbb{E} [\mathbb{N}(\tilde{a} | b) \cdot \|b\|^2] &= \int p(b) \mathbb{N}(\tilde{a} | b) \cdot \|b\|^2 db \\
&= \text{Tr}(\Sigma_p) \int \left(\frac{p(b) \cdot \|b\|^2}{\text{Tr}(\Sigma_p)} \right) \mathbb{N}(\tilde{a} | b) db \\
&= \text{Tr}(\Sigma_p) N_q(\tilde{a}).
\end{aligned}$$

The second term can be rewritten as follows using Σ_p and Σ_q :

$$\begin{aligned}
\mathbb{E} \left[\left\| (\theta - \tilde{\theta})(bb^\top - \Sigma_p) \right\|_F^2 \right] &= \mathbb{E} \left[\text{Tr} \left((bb^\top - \Sigma_p)(\theta - \tilde{\theta})^\top (\theta - \tilde{\theta})(bb^\top - \Sigma_p) \right) \right] \\
&= \mathbb{E} \left[\text{Tr} \left((\theta - \tilde{\theta})(bb^\top - \Sigma_p)(bb^\top - \Sigma_p)(\theta - \tilde{\theta})^\top \right) \right] \\
&= \text{Tr} \left((\theta - \tilde{\theta}) \mathbb{E} [(bb^\top - \Sigma_p)(bb^\top - \Sigma_p)] (\theta - \tilde{\theta})^\top \right) \\
&= \text{Tr} \left((\theta - \tilde{\theta}) (\text{Tr}(\Sigma_p)\Sigma_q - \Sigma_p^2) (\theta - \tilde{\theta})^\top \right).
\end{aligned}$$

For the deterministic gradient upper bound in [Eq. \(15\)](#), we have:

$$\|\nabla \tilde{\mathcal{L}}_{\text{reg}}(\theta)\|_F^2 = 4 \text{Tr} \left((\theta - \tilde{\theta}) \Sigma_p^2 (\theta - \tilde{\theta})^\top \right).$$

Putting these results together, we can upper bound the expected gradient norm for any parameter values $\|\theta\|_F \leq D$ as

$$\begin{aligned}
\mathbb{E} \left[\|g(\theta)\|_F^2 \right] &= 4 \text{Tr}(\Sigma_p) N_q(\tilde{a}) + 4 \text{Tr}(\Sigma_p) \text{Tr} \left((\theta - \tilde{\theta}) \Sigma_q (\theta - \tilde{\theta})^\top \right) \\
&\leq 4 \text{Tr}(\Sigma_p) N_q(\tilde{a}) + 4 \text{Tr}(\Sigma_p) \lambda_{\max}(\Sigma_q) \|\theta - \tilde{\theta}\|_F^2 \\
&\leq 4 \text{Tr}(\Sigma_p) N_q(\tilde{a}) + 16 \text{Tr}(\Sigma_p) \lambda_{\max}(\Sigma_q) D^2.
\end{aligned}$$

Using our results for the objective's strong convexity and expected gradient norm, we can now invoke a standard SGD convergence result. Following [Theorem 6.3](#) from [Bubeck et al. \[5\]](#), if we make T updates with the specified step size, we have the following upper bound on the expected objective value:

$$\mathbb{E}[\tilde{\mathcal{L}}_{\text{reg}}(\bar{\theta}_T)] - N(\tilde{a}) \leq \frac{4 \text{Tr}(\Sigma_p) N_q(\tilde{a}) + 16 \text{Tr}(\Sigma_p) \lambda_{\max}(\Sigma_q) D^2}{\lambda_{\min}(\Sigma_p)(T+1)}. \quad (16)$$

□

Next, we prove a simple consequence of this result, which is that the rate from [Theorem 1](#) applies to the original objective $\mathcal{L}_{\text{reg}}(\theta)$ when we assume an unbiased noisy oracle.

Corollary 1. *Following the setup from [Theorem 1](#), if the noisy oracle is unbiased, then the averaged iterate at step T satisfies*

$$\mathbb{E}[\mathcal{L}_{\text{reg}}(\bar{\theta}_T)] \leq \frac{4 \text{Tr}(\Sigma_p) N_q(\tilde{a}) + 16 \text{Tr}(\Sigma_p) \lambda_{\max}(\Sigma_q) D^2}{\lambda_{\min}(\Sigma_p)(T+1)}.$$

If the oracle is also noiseless, or if we assume access to the exact oracle $\tilde{a}(b) = a(b)$, then the averaged iterate at step T satisfies

$$\mathbb{E}[\mathcal{L}_{\text{reg}}(\bar{\theta}_T)] \leq \frac{16 \text{Tr}(\Sigma_p) \lambda_{\max}(\Sigma_q) D^2}{\lambda_{\min}(\Sigma_p)(T+1)}.$$

Proof. The first result follows from combining [Theorem 1](#) with the relationship between $\mathcal{L}_{\text{reg}}(\theta)$ and $\tilde{\mathcal{L}}_{\text{reg}}(\theta)$: if we assume the noisy oracle is unbiased, or $B(\tilde{a}) = 0$, then we have $\mathcal{L}_{\text{reg}}(\theta) = \tilde{\mathcal{L}}_{\text{reg}}(\theta) - N(\tilde{a})$, which implies the first inequality. The second inequality follows from setting $N_q(\tilde{a}) = 0$ in the upper bound. \square

Next, we provide a proof for [Eq. \(13\)](#) regarding the connection between regression- and objective-based amortization, which is the following:

$$\frac{\alpha}{2} \mathcal{L}_{\text{reg}}(\theta) \leq \mathcal{L}_{\text{obj}}(\theta) - \mathcal{L}_{\text{obj}}^* \leq \frac{\beta}{2} \mathcal{L}_{\text{reg}}(\theta).$$

Proof. This result relies on well known properties from convex optimization [\[41\]](#). Consider a fixed context variable $b \in \mathcal{B}$. Strong convexity with $\alpha > 0$ means that for all $a, a' \in \mathcal{A}$ we have:

$$h(a; b) \geq h(a'; b) + (a - a')^\top \nabla_a h(a'; b) + \frac{\alpha}{2} \|a - a'\|^2.$$

Considering the optimal value $a' = a(b)$, this simplifies to:

$$\frac{\alpha}{2} \|a - a(b)\|^2 \leq h(a; b) - h(a(b); b). \quad (17)$$

Next, smoothness with $\beta > 0$ means that for all $a, a' \in \mathbb{R}^d$ we have:

$$\|\nabla_a h(a; b) - \nabla_a h(a'; b)\|^2 \leq \beta \|a - a'\|.$$

[Lemma 3.4](#) in [Bubeck et al. \[5\]](#) shows that β -smoothness also implies the following:

$$|h(a; b) - h(a'; b) - (a - a')^\top \nabla_a h(a'; b)| \leq \frac{\beta}{2} \|a - a'\|^2.$$

Considering the optimal value $a' = a(b)$, this simplifies to:

$$h(a; b) - h(a(b); b) \leq \frac{\beta}{2} \|a - a(b)\|^2. \quad (18)$$

The bounds in [Eq. \(13\)](#) follow from substituting a for predictions from a model $a(b; \theta)$, and considering [Eq. \(17\)](#) and [Eq. \(18\)](#) in expectation across the distribution $p(b)$. \square

D.2 Datamodels proofs

Before proving our main claim for datamodels in [Proposition 1](#), we first prove a more general result. This version considers an arbitrary symmetric distribution $p(T)$ over datasets, rather than the specific distribution parameterized by $q \in (0, 1)$ used in [Proposition 1](#). By a symmetric distribution, we mean one that satisfies $p(T) = p(T')$ whenever $|T| = |T'|$.

As setup for our derivation, notice that when we assume a symmetric distribution $p(T)$, we can define the following probabilities that are identical for all indices $i, j \in [n]$:

$$\begin{aligned} p_1 &\equiv \Pr(i \in T) \quad \text{for } i \in [n] \\ p_2 &\equiv \Pr(i, j \in T) \quad \text{for } i \neq j \\ p_3 &\equiv \Pr(i \in T, j \notin T) \quad \text{for } i \neq j. \end{aligned}$$

Note that we have $p_1 = p_2 + p_3$. For example, given a uniform distribution $p(T)$ over subsets with size $|T| = k$, which is used by [Ilyas et al. \[47\]](#), we have:

$$p_1 = \frac{\binom{n-1}{k-1}}{\binom{n}{k}} = \frac{k}{n} \quad p_2 = \frac{\binom{n-2}{k-1}}{\binom{n}{k}} = \frac{k(k-1)}{n(n-1)} \quad p_3 = \frac{\binom{n-2}{k-1}}{\binom{n}{k}} = \frac{k(n-k)}{n(n-1)}.$$

Alternatively, if we have each data point included independently with probability $q \in (0, 1)$, which is used by Saunshi et al. [85] and [Proposition 1](#), then we have $p_1 = q$, $p_2 = q^2$, and $p_3 = q(1 - q)$.

Our more general claim about datamodels with a symmetric distribution $p(T)$ is the following.

Lemma 1. *Given a symmetric distribution $p(T)$, the data attribution scores defined by*

$$\zeta(x) \equiv \arg \min_{a \in \mathbb{R}^{n+1}} \mathbb{E}_{p(T)} \left[\left(a_0 + \sum_{i \in T} a_i - v_x(\mathcal{D}_T) \right)^2 \right]$$

can be expressed as the expectation $\zeta(z_i, x) = \mathbb{E}[c_i(T)v_x(\mathcal{D}_T)]$, where we define the weighting function $c_i(T)$ as follows:

$$c_i(T) \equiv \frac{1}{p_3} \left(\mathbf{1}(i \in T) - \frac{p_2 - p_1^2}{p_3 + n(p_2 - p_1^2)} |T| - \frac{p_1 p_3}{p_3 + n(p_2 - p_1^2)} \right).$$

Proof. We can write the problem's partial derivatives as follows, both for the intercept term a_0 and the coefficients a_i for $i \in [n]$:

$$\begin{aligned} \frac{\partial}{\partial a_0} \mathbb{E} \left[(a_0 + a^\top \mathbf{1}_T - v_x(\mathcal{D}_T))^2 \right] &= 2 (a_0 + \mathbb{E} [\mathbf{1}_T^\top a] - \mathbb{E} [v_x(\mathcal{D}_T)]) \\ &= 2 (a_0 + p_1 \mathbf{1}_n^\top a - \mathbb{E} [v_x(\mathcal{D}_T)]) \end{aligned}$$

$$\begin{aligned} \frac{\partial}{\partial a_i} \mathbb{E} \left[(a_0 + a^\top \mathbf{1}_T - v_x(\mathcal{D}_T))^2 \right] &= 2 (\mathbb{E} [\mathbf{1}(i \in T)(\mathbf{1}_T^\top a + a_0)] - \mathbb{E} [\mathbf{1}(i \in T)v_x(\mathcal{D}_T)]) \\ &= 2 (p_1 a_0 + p_2 \mathbf{1}_n^\top a + (p_1 - p_2)a_i - p_1 \mathbb{E} [v_x(\mathcal{D}_T) \mid i \in T]). \end{aligned}$$

We use the shorthand notation $\bar{v} \in \mathbb{R}^{n+1}$ for a vector with entries $\bar{v}_0 = \mathbb{E}[v_x(\mathcal{D}_T)]$ and $\bar{v}_i = \mathbb{E}[v_x(\mathcal{D}_T) \mid i \in T]$ for $i \in [n]$. Combining this with the partial derivatives, we can derive an analytic solution $a^* \in \mathbb{R}^{n+1}$ by setting the derivative to zero,

$$\begin{pmatrix} 1 & p_1 \mathbf{1}_n^\top \\ p_1 \mathbf{1}_n & p_2 \mathbf{1}_n \mathbf{1}_n^\top + p_3 I_n \end{pmatrix} a^* - \begin{pmatrix} 1 & 0 \\ 0 & p_1 I_n \end{pmatrix} \bar{v} = 0,$$

which yields the following equation for the solution:

$$a^* = \begin{pmatrix} 1 & p_1 \mathbf{1}_n^\top \\ p_1 \mathbf{1}_n & p_2 \mathbf{1}_n \mathbf{1}_n^\top + p_3 I_n \end{pmatrix}^{-1} \begin{pmatrix} 1 & 0 \\ 0 & p_1 I_n \end{pmatrix} \bar{v}.$$

To find the required matrix inverse, we can combine the Sherman-Morrison formula with the formula for block matrix inversion. The formula for block matrix inversion is the following [79]:

$$\begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix}^{-1} = \begin{pmatrix} \mathbf{A}^{-1} + \mathbf{A}^{-1} \mathbf{B} (\mathbf{D} - \mathbf{C} \mathbf{A}^{-1} \mathbf{B})^{-1} \mathbf{C} \mathbf{A}^{-1} & -\mathbf{A}^{-1} \mathbf{B} (\mathbf{D} - \mathbf{C} \mathbf{A}^{-1} \mathbf{B})^{-1} \\ -(\mathbf{D} - \mathbf{C} \mathbf{A}^{-1} \mathbf{B})^{-1} \mathbf{C} \mathbf{A}^{-1} & (\mathbf{D} - \mathbf{C} \mathbf{A}^{-1} \mathbf{B})^{-1} \end{pmatrix}.$$

We are mainly interested in the lower rows of this matrix because we do not use the learned intercept term. For the lower right matrix, we have the following:

$$\begin{aligned}
(\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1} &= (p_2\mathbf{1}_n\mathbf{1}_n^\top + p_3I_n - p_1^2\mathbf{1}_n\mathbf{1}_n^\top)^{-1} \\
&= ((p_2 - p_1^2)\mathbf{1}_n\mathbf{1}_n^\top + p_3I_n)^{-1} \\
&= \frac{1}{p_3}I_n - \frac{p_2 - p_1^2}{p_3(p_3 + n(p_2 - p_1^2))}\mathbf{1}_n\mathbf{1}_n^\top.
\end{aligned}$$

Next, for the lower left vector, we have the following:

$$\begin{aligned}
-(\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}^{-1} &= -\left(\frac{1}{p_3}I_n - \frac{p_2 - p_1^2}{p_3(p_3 + n(p_2 - p_1^2))}\mathbf{1}_n\mathbf{1}_n^\top\right)p_1\mathbf{1}_n \\
&= -\frac{p_1}{p_3}\mathbf{1}_n + \frac{np_1}{p_3}\frac{p_2 - p_1^2}{p_3 + n(p_2 - p_1^2)}\mathbf{1}_n \\
&= -\frac{p_1}{p_3 + n(p_2 - p_1^2)}\mathbf{1}_n.
\end{aligned}$$

This yields the following solution for the optimal coefficients a_i^* with $i \in [n]$:

$$a_i^* = \frac{p_1}{p_3}\bar{v}_i - \frac{p_1(p_2 - p_1^2)}{p_3(p_3 + n(p_2 - p_1^2))}\sum_{j \in [n]} \bar{v}_j - \frac{p_1}{p_3 + n(p_2 - p_1^2)}\bar{v}_0.$$

Based on this solution, we can design a function $c_i(T)$ so that we have the expectation $\mathbb{E}[c_i(T)v_x(\mathcal{D}_T)] = a_i^*$. We define the function as follows,

$$c_i(T) \equiv \frac{1}{p_3}\left(\mathbf{1}(i \in T) - \frac{p_2 - p_1^2}{p_3 + n(p_2 - p_1^2)}|T| - \frac{p_1p_3}{p_3 + n(p_2 - p_1^2)}\right),$$

and it can be verified that this satisfies the required expectation. □

A similar derivation to [Lemma 1](#) is possible when we omit an intercept term in the datamodels optimization problem, but we do not show the result here.

Finally, we prove the special case of [Lemma 1](#) considered in [Proposition 1](#).

Proposition 1. *Given a subset distribution that includes each data point $z_i \in \mathcal{D}$ with probability $q \in (0, 1)$, the data attribution scores defined by*

$$\zeta(x) \equiv \arg \min_{a \in \mathbb{R}^{n+1}} \mathbb{E}_T \left[\left(a_0 + \sum_{i \in T} a_i - v_x(\mathcal{D}_T) \right)^2 \right]$$

can be expressed as the following expectation:

$$\zeta(z_i, x) = \sum_{T \subseteq [n] \setminus i} u(|T|) (v_x(\mathcal{D}_T \cup \{z_i\}) - v_x(\mathcal{D}_T)).$$

Proof. Consider the weighting function $c_i(T)$ introduced in the proof for [Lemma 1](#). In this case, we can use the fact that $p_2 = p_1^2$ and write the weighting function as follows:

$$c_i(T) = \frac{1}{p_3}(\mathbf{1}(i \in T) - p_1) = \begin{cases} \frac{1-p_1}{p_3} & i \in T \\ -\frac{p_1}{p_3} & i \notin T. \end{cases}$$

Using the fact that $p(T) = q^{|T|}(1-q)^{n-|T|}$ and the coefficient values $(1-p_1)/p_3 = q^{-1}$ and $-p_1/p_3 = -(1-q)^{-1}$, we arrive at the result in [Proposition 1](#). □

E Noisy Oracles for XML Methods

This section describes the statistical estimators used for each XML task, which are briefly introduced in Section 4. Each estimator serves as a noisy oracle for the given task, which allows us to train amortized models with inexpensive supervision.

Shapley values. We use three statistical estimators for Shapley value feature attributions. The simplest is permutation sampling, where we average each feature’s contribution across a set of sampled orderings [6, 96]. For this approach, we use ρ to denote a permutation of the indices $[d]$, where we let $\rho(i) \subseteq [d] \setminus \{i\}$ denote the elements appearing before i in the permutation. The Shapley value can be understood as the marginal contribution $f(x_{\rho(i) \cup \{i\}}) - f(x_{\rho(i)})$ averaged across all possible permutations [89], so our first estimator involves sampling k permutations ρ_1, \dots, ρ_k , and then calculating the following average for each feature:

$$\hat{\phi}_i(x) = \frac{1}{k} \sum_{j=1}^k f(x_{\rho_j(i) \cup \{i\}}) - f(x_{\rho_j(i)}). \quad (19)$$

Next, we consider two estimators based on the Shapley value’s least squares view. Recall that the Shapley value is the solution to the following problem (where we discard the intercept term),

$$\phi(x) = \arg \min_{a \in \mathbb{R}^{d+1}} \sum_{S \subseteq [d]} \mu(S) \left(f(x_S) - a_0 - \sum_{i \in S} a_i \right)^2, \quad (20)$$

where we use a weighting kernel defined as $\mu^{-1}(S) = \binom{d}{|S|} |S|(d - |S|)$ [7]. The first estimator we use based on this perspective is KernelSHAP [68], which solves this problem using k subsets $S_j \subseteq [d]$ sampled according to $\mu(S)$. The problem is convex but constrained due to the weighting terms $\mu([d]) = \mu(\emptyset) = \infty$, so it must be solved via the KKT conditions; we refer readers to Covert and Lee [16] for the closed-form solution. We also consider the SGD-Shapley approach from Simon and Vincent [92]: rather than solving Eq. (20) exactly given sampled subsets, this approach solves the problem iteratively with projected stochastic gradient descent. Unlike the other estimators, SGD-Shapley has been shown to have non-negligible bias [10]. We used a custom implementation for SGD-Shapley, and we used an open-source implementation of permutation sampling and KernelSHAP.⁹

Banzhaf values. When calculating Banzhaf value feature attributions, we adapt the *maximum sample re-use* (MSR) estimator from Wang and Jia [103], which was originally used for data valuation but is equally applicable to feature attribution. For a single example $x \in \mathcal{X}$, we generate predictions using k subsets $S_j \subseteq [d]$ sampled uniformly at random. We then split the subsets into those that include or exclude each feature $i \in [d]$, and we estimate the Banzhaf value as follows:

$$\hat{\phi}_i(x) = \frac{1}{|\{j : i \in S_j\}|} \sum_{j: i \in S_j} f(x_{S_j}) - \frac{1}{|\{j : i \notin S_j\}|} \sum_{j: i \notin S_j} f(x_{S_j}). \quad (21)$$

The re-use of samples across all features makes this estimator more efficient than independent Monte Carlo estimates [103], and re-using samples in this way is simpler for Banzhaf values than for other methods like Shapley values [20, 58]. We used a custom implementation for this approach.

LIME. Similar to KernelSHAP, the most popular estimator for LIME feature attributions is based on approximately solving its weighted least squares problem. Following the problem formulation in Section 4.2, which we simplify by omitting the regularization term Ω , we sample k subsets $S_j \subseteq [d]$ uniformly at random and solve the following importance sampling version of the objective:

$$\hat{\phi}_1(x), \dots, \hat{\phi}_d(x) = \arg \min_{a \in \mathbb{R}^{d+1}} \sum_{j=1}^k \pi(S_j) \left(a_0 + \sum_{i \in S_j} a_i - f(x_{S_j}) \right)^2. \quad (22)$$

⁹<https://github.com/iancovert/shapley-regression>

When doing so, we discard the intercept term, we ensure that k is large enough to avoid singular matrix inversion, and we use the default weighting kernel $\pi(S)$ for images in the official LIME implementation.¹⁰ We opt to sample subsets uniformly rather than according to $\pi(S)$ because this approach is used in the official implementation, and because the weighting kernel is similar to sampling subsets uniformly at random [66]. We used a custom implementation of this approach.

Data valuation. For the various data valuation methods discussed in Section 4.3, we use the simplest unbiased approximation, which is a Monte Carlo estimator that averages the performance difference across a set of sampled datasets. Given a labeled data point z , we sample k datasets $\mathcal{D}_j \subseteq \mathcal{D}$ from the appropriate distribution and calculate the following empirical average:

$$\hat{\psi}(z) = \frac{1}{k} \sum_{j=1}^k v(\mathcal{D}_j \cup \{z\}) - v(\mathcal{D}_j). \quad (23)$$

This is similar to the TMC algorithm from Ghorbani and Zou [33], but we do not employ truncation; our approach can be used with truncation, but the noisy labels and valuation model predictions would both become biased. Similar to previous works, we also adopt a minimum subset cardinality to avoid training models with an insufficient number of data points. We implemented this approach using the OpenDataVal package [53]. More sophisticated estimators are available, like those that exploit stratification [106], re-use samples across all data points [103], or assume sparse attribution scores [52], but we leave exploration of these estimators to future work.

¹⁰<https://github.com/marcotcr/lime>

F Experiment Details

Model architectures. For the feature attribution experiments, we used the ViT-Base architecture [24] for the classifier, and we used a modified version of the architecture for the amortized attribution model: following the approach from Covert et al. [18], we added an extra self-attention layer and three fully-connected layers that operate on each token, so that the output contains an attribution score for each class. When estimating feature attributions, we make predictions with subsets of patches by setting the held-out patches to zero, and the classifier was fine-tuned with random masking to accommodate missing patches [17, 48, 50, 18].

For the data valuation experiments, we used a FCN with two hidden layers of size 128 for the tabular datasets, and we used a ResNet-18 architecture [43] for CIFAR-10. Valuation scores are defined for labeled examples $z = (x, y)$, so the valuation model must account for both the features x and the class y when making predictions; rather than passing y as a model input, our architecture makes predictions simultaneously for all classes, and we only use the estimate for the relevant class. When generating noisy labels using the TMC estimator [33], we trained a logistic regression model on raw input features for the tabular datasets, and for CIFAR-10 we trained a logistic regression on pre-trained ResNet-50 features whose dimensionality was reduced with PCA.

Hyperparameters. For the feature attribution experiments, we optimized the models using the AdamW optimizer [67] with a linearly decaying learning rate schedule. The maximum learning rate was tuned using the validation loss, we trained for up to 100 epochs, and we selected the best model based on the validation loss. Due to the small scale of the noisy labels for Banzhaf and LIME feature attributions, we re-scaled the labels during training to improve the model’s stability (see Appendix G.2), but this re-scaling was not necessary for Shapley values.

For the data valuation experiments, we optimized the models using Adam [55] with a cosine learning rate schedule. The maximum learning rate, the number of training epochs and the best model from the training run were determined using the validation loss. Due to the small scale of the noisy labels, we found it helpful to re-scale them during training to have standard deviation approximately equal to 1. The cost of performing multiple training runs is negligible compared to generating the noisy training labels, so it is not reflected in our plots comparing amortization to the Monte Carlo estimator (e.g., Figure 4).

Pretraining. We found that training the amortized models was faster and more stable when we initialized using pretrained architectures. For the feature attribution experiments, we initialized the model using the existing ViT-Base classifier weights. For data valuation, we initialized from a classifier trained using the entire dataset, where we used a FCN for the tabular datasets and a ResNet-18 for CIFAR-10. When adapting these models to their respective amortization tasks, the feature attribution models had freshly initialized output layers (one self-attention and three fully-connected), and the data valuation models had identical output layers that we re-initialized to zero.

Validation loss. For the feature attribution experiments, we calculated the validation loss using independent estimates generated with same estimator and the same number of samples used for the noisy training labels. For the data valuation experiments, we implemented the validation loss using independent Monte Carlo estimates of the data valuation scores. Our independent estimates use only 10 samples for the tabular datasets, and just one sample for CIFAR-10. Amortization provides a significant benefit over per-example calculations even after accounting for the cost of the validation loss.

Noisy oracle details. The estimators used for each task are described in detail in Appendix E. For the feature attribution experiments, the permutation sampling and KernelSHAP estimators only require the number of samples as hyperparameters, and we tested multiple values in our experiments (e.g., Figure 3). For SGD-Shapley, we tuned the algorithm in several ways to improve its performance: we used a constant rather than decaying learning rate, we took a uniform average over iterates, we calculated gradients using multiple subsets, and we used the paired sampling trick from Covert and Lee [16] to reduce the gradient variance. We also tuned the learning rate to a value that did not cause divergence for any examples ($5e-4$). As described in Appendix G.1, we observed that SGD-Shapley often had the lowest squared error among the three Shapley value estimators (Figure 11), but that it did not lead to successful amortization due to its non-negligible bias (Figure 10).

For the data valuation experiments, models were trained on each subsampled dataset by fitting a logistic regression model, either on raw input features for the tabular datasets or on pre-trained ResNet-50 features for CIFAR-10. For the Data Shapley experiments in [Section 5.2](#) with tabular datasets, we sampled datasets without replacement, we used a minimum cardinality of 5 points and a maximum cardinality equal to the number of points n . For the Distributional Data Shapley experiments in [Section 5.3](#) with CIFAR-10, we sampled datasets with replacement, and we used a minimum cardinality of 100 and maximum cardinality of 1000.

Ground truth. In all our experiments, the ground truth is obtained by running an existing per-instance estimator for a large number of samples, and we use enough samples to ensure that it has approximately converged to the exact result. For the Shapley value feature attribution experiments, we run KernelSHAP for 1M samples. For Banzhaf values, we run the MSR estimator for 1M samples. For LIME, we run the least squares estimator for 1M samples. For the data valuation experiments, we run the Monte Carlo estimator for 10K samples. These estimators are costly to run for a large number of samples, so we do so for only a small portion of the dataset: we calculate the ground truth for 100 examples for the feature attribution experiments. For the data valuation experiments, we use 250 examples for the tabular datasets and 500 for CIFAR-10.

Metrics. The performance metrics used throughout our experiments are related to the estimation accuracy, which we evaluate using our ground truth values. Other works have evaluated Shapley value feature attributions against other methods [50, 18] or used data valuation scores in a range of downstream tasks [53], but our focus is on efficient and accurate estimation. For the feature attribution experiments, we used the squared error distance, which we calculate across all attributions; we used Pearson and Spearman correlation, which are calculated individually for each flattened vector of attribution scores and then averaged across data points; and we used sign agreement, which is averaged across all attributions. For the data valuation experiments, we used squared error distance, which we normalized so the mean ground truth valuation score has error equal to 1 (i.e., we report the error divided by the variance in ground truth scores); and we used Pearson correlation, Spearman correlation and sign agreement, which were calculated using the vectors of estimated and ground truth valuation scores.

Datasets. We used publicly available, open-source datasets for our experiments. For the feature attribution experiments, we used the ImageNette dataset, a natural image dataset consisting of ten ImageNet classes [46, 23]. We partitioned the 224×224 inputs into 196 patches of size 14×14 , and the dataset was split into a training set with 9469 examples, a validation set with 1962 examples, and a test set with 1963 examples. The validation set was used to perform early stopping, and the test set was only used to evaluate the model’s performance on external data.

For the data valuation experiments, we used two tabular datasets from the UCI repository: the MiniBooNE particle classification dataset [84] and the adult census income classification dataset [25]. We obtained these using the OpenDataVal package [53], we used variable numbers of training examples ranging from 250 to 10K, and we reserved 100 examples for validation in each case; these examples are only used to evaluate the performance of models trained on subsampled datasets. We also used OpenDataVal to add label noise to 20% of the training examples. For CIFAR-10, we used 50K examples for training and 1K for validation. We tested multiple levels of label noise for CIFAR-10 ([Appendix G.3](#)), and our main text results use 10% label noise.

Compute resources. For the feature attribution experiments, we used a single GeForce RTX 2080Ti GPU to train the amortized models. Training an amortized model on ImageNette dataset for 50 epochs required roughly 4 hours. For the data valuation experiments, we used a single RTX A6000 to train amortized models, and these each trained in under an hour.

FLOPs profiling. When profiling compute for our feature attribution experiments (see [Figure 3](#)), we first measured the following constants using the DeepSpeed FLOPs profiler, all using the ViT-Base architecture:

- The classifier’s forward pass requires 17,563,067,904 FLOPs \approx 17.6 GFLOPs per prediction.
- The amortized model’s forward pass requires 21,335,153,664 FLOPs \approx 21.3 GFLOPs per prediction, due to the model having an extra transformer block for fine-tuning.
- The amortized model has 104,730,000 \approx 105M trainable parameters.

Next, for the FLOPs comparison shown in [Figure 3](#) we calculated total FLOPs for the two approaches as follows. For KernelSHAP, the total FLOPs = $17.6 \text{ GFLOPs} \times (\text{number of subset samples}) \times (\text{number of training datapoints})$. For the amortized models, the total FLOPs are the sum of multiple terms:

1. FLOPs for obtaining noisy labels = $17.6 \text{ GFLOPs} \times (\text{number of subset samples}) \times (\text{number of training datapoints})$
2. FLOPs for the forward pass during training = $21.3 \text{ GFLOPs} \times (\text{number of epochs}) \times (\text{number of training datapoints})$
3. FLOPs for the backward pass during training = $21.3 \text{ GFLOPs} \times 2 \times (\text{number of epochs}) \times (\text{number of training datapoints})$, where 2 is a standard multiplier for calculating FLOPs during the backward pass
4. FLOPs for updating parameters = $105 \text{ MFLOPs} \times (\text{number of epochs}) \times (\text{number of training datapoints}) / (\text{batch size}) \times 19$, where 19 is due to the optimizer state

In the calculation above, 1) dominates due to the large number of subset samples used for each datapoint (e.g., >512 for KernelSHAP). Meanwhile, the training epochs in 2), 3) and 4) are relatively low due to our models' fast convergence (<25 epochs). In the FLOPs calculation above, training an amortized model for one epoch is equivalent to obtaining predictions for roughly 3.65 additional subset samples per training datapoint. This is because $(3 \times 21,335,153,664 + 104,730,000 \times 19/64)/17,563,067,904 \approx 3.65$. In other words, the amount of compute for training an amortized model for up to 50 epochs translates to obtaining predictions for just 182.5 subset samples per datapoint, which is not enough to meaningfully improve the estimates by running KernelSHAP for more iterations. This explains why the lines for amortized models appear almost vertical in [Figure 3](#) when we compared amortization to KernelSHAP. Our procedure for calculating FLOPs is similar to other works that profile computation when training neural networks.¹¹

¹¹We consulted the following resources: (i) <https://www.lesswrong.com/posts/jJApGWG95495pYM7C/how-to-measure-flop-s-for-neural-networks-empirically>, (ii) <https://www.lesswrong.com/posts/fnjKpBoWJXcSDwhZk/what-s-the-backward-forward-flop-ratio-for-neural-networks>, and (iii) <https://www.adamcasson.com/posts/transformer-flops>

G Additional Results

This section provides additional experimental results involving Shapley value feature attributions (Appendix G.1), Banzhaf value and LIME feature attributions (Appendix G.2), and data valuation (Appendix G.3).

G.1 Shapley value amortization

First, Figure 6 shows that the performance of our amortized models is comparable to running KernelSHAP for 10-40k samples. For example, an amortized model trained on noisy KernelSHAP labels generated with 512 samples produces outputs of similar quality to running KernelSHAP for about 20k samples in terms of MSE, which is equivalent to a speedup of roughly 40x.

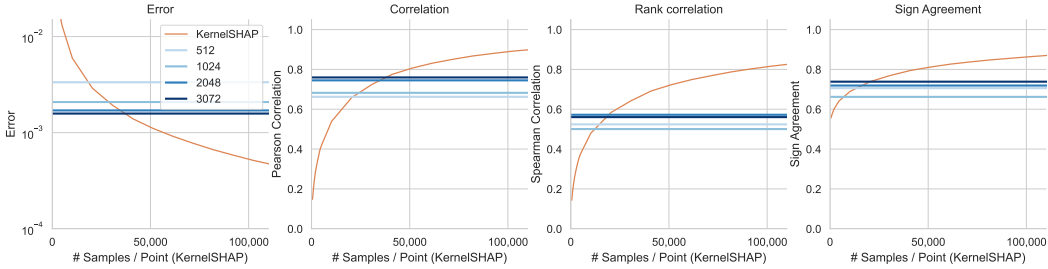


Figure 6: Comparison of the estimation accuracy between KernelSHAP and amortized predictions.

Next, Figure 7 shows an expanded version of Figure 3 (right) from the main text, where we compare amortization to per-instance estimation given a fixed amount of compute per data point. We observe that across four metrics, amortization provides a benefit over KernelSHAP even when training with a small portion of the ImageNette dataset.

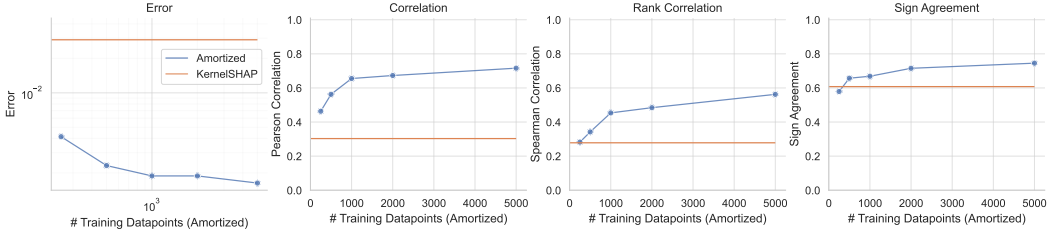


Figure 7: Estimation accuracy for amortization and KernelSHAP with different dataset sizes given equivalent compute.

Figure 8 shows a similar result, but the amortized model’s performance is evaluated with external data points (i.e., points that are not seen during training). The benefits of amortization remain significant for most dataset sizes, reflecting that the model generalizes beyond the training data and can be used for real-time feature attribution with new examples.

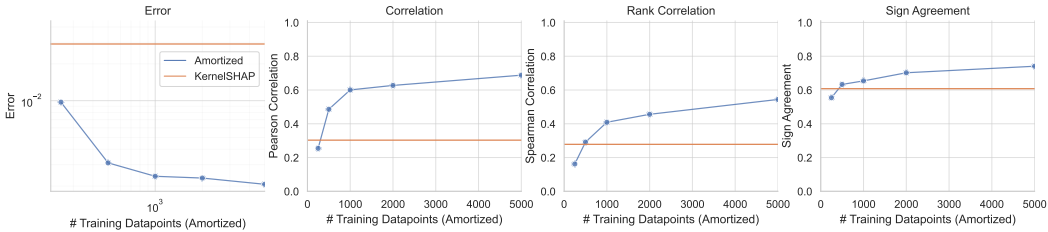


Figure 8: Estimation accuracy for amortization and KernelSHAP with different dataset sizes given equivalent compute (external data points).

Figure 9 shows an expanded version of Figure 3 (center) where we can see the benefits of amortization across multiple numbers of KernelSHAP samples when we account for the number of FLOPs.

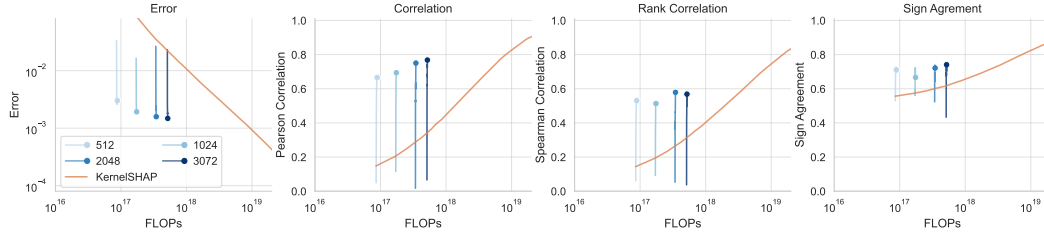


Figure 9: Error of amortization and KernelSHAP as a function of FLOPs.

Figure 10 shows an expanded version of Figure 3, where we can see the benefits of amortization for KernelSHAP and permutation sampling across four metrics. We also see that SGD-Shapley does not lead to successful amortization, because the predictions are worse than the noisy labels across all four metrics. This is perhaps surprising, because Figure 11 shows that SGD-Shapley has the lowest squared error among the three noisy oracles. The crucial issue with SGD-Shapley is that its estimates are not unbiased (see Section 3), an issue that has been shown in prior work [10].

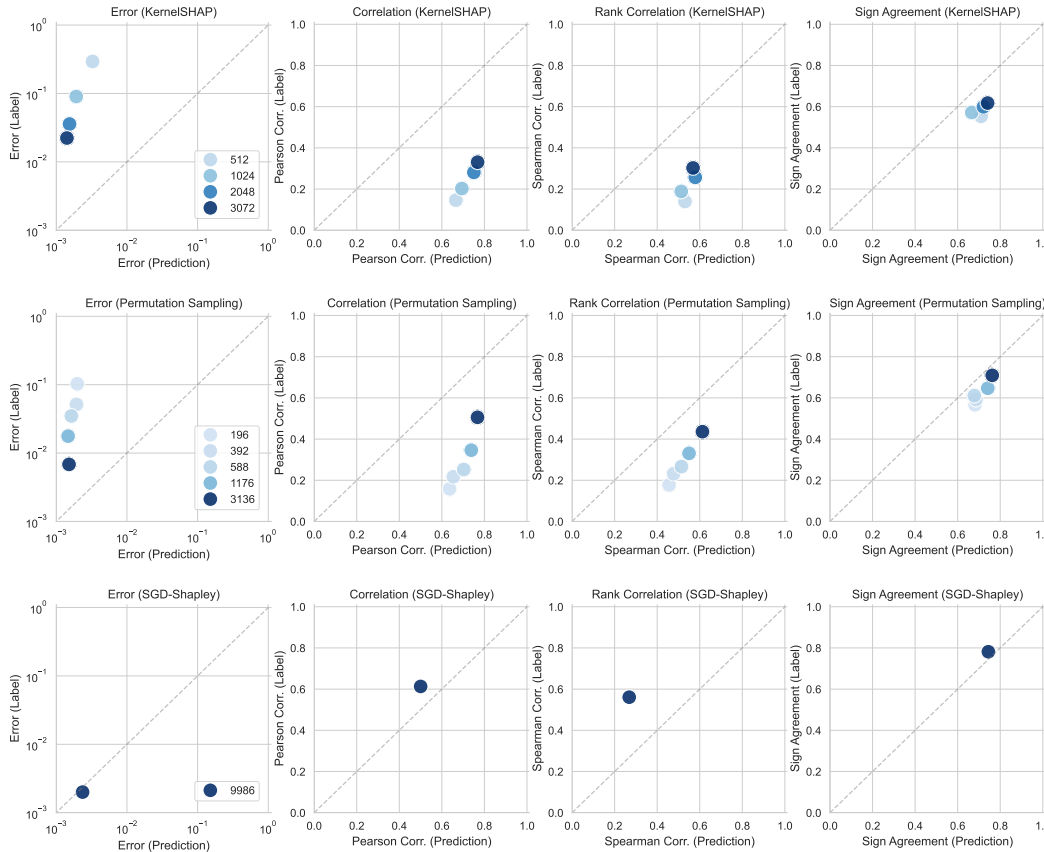


Figure 10: Comparison of the estimation error between noisy labels and amortized predictions for Shapley value feature attributions. Top: noisy labels generated using KernelSHAP with different numbers of samples. Middle: noisy labels generated using permutation sampling with different numbers of samples. Bottom: noisy labels generated using SGD-Shapley with different numbers of samples.

Figure 12 is similar to Figure 10, only the performance metrics are calculated using external points not seen during training. Like Figure 12, this result emphasizes that the amortized attribution model generalizes well and can be reliably used with new data points.

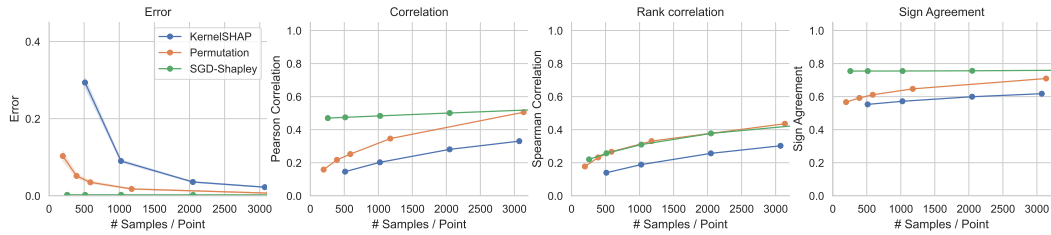


Figure 11: Comparison of the estimation error between different per-example estimators for Shapley value feature attributions across varying numbers of samples.

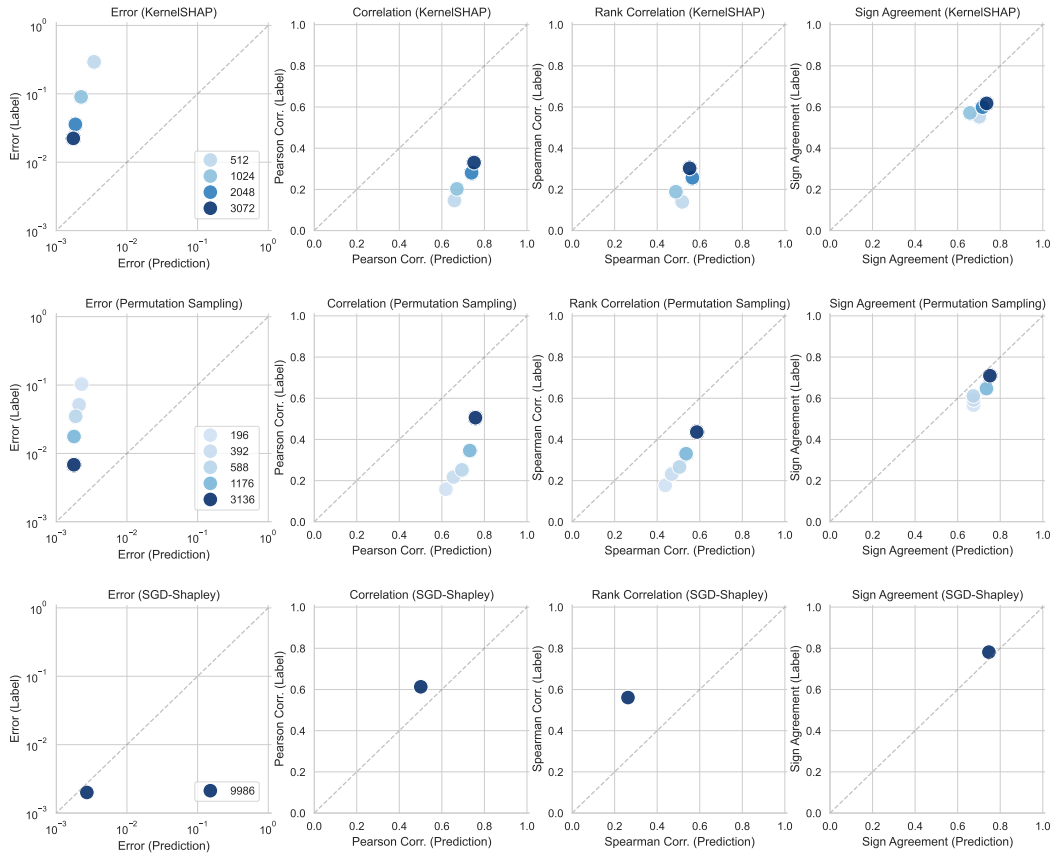


Figure 12: Comparison of the estimation error between noisy labels and amortized predictions for Shapley value feature attributions (external data points). Top: noisy labels generated using KernelSHAP with different numbers of samples. Middle: noisy labels generated using permutation sampling with different numbers of samples. Bottom: noisy labels generated using SGD-Shapley with different numbers of samples.

Finally, Figure 13 provides a comparison between stochastic amortization and FastSHAP [50, 18], one of the main existing approaches to amortized Shapley value estimation. For stochastic amortization, we use our previous results with permutation sampling as the noisy oracle and five different numbers of samples. For FastSHAP, we train the same ViT-Base architecture following the approach from Covert et al. [18], using 32 subset samples per gradient step. We monitor FastSHAP’s estimation accuracy at the end of each epoch while training for a total of 100 epochs. We observe that FastSHAP and stochastic amortization achieve similar error at each total FLOPs budget, where both methods incur FLOPs from training the amortized model, stochastic amortization incurs FLOPs upfront when generating noisy labels, and FastSHAP incurs FLOPs during training while sampling subsets for each gradient step. FastSHAP is slightly more accurate at the largest computational budget, and both are significantly more accurate than per-sample estimation with KernelSHAP. Due to their similar performance, the main advantage of our approach is its simplicity (a standard regression objective rather than the custom FastSHAP objective), and the flexibility to use any unbiased estimator as the noisy oracle; future work may find that stochastic amortization is more effective with other noisy oracles that we did not try here.

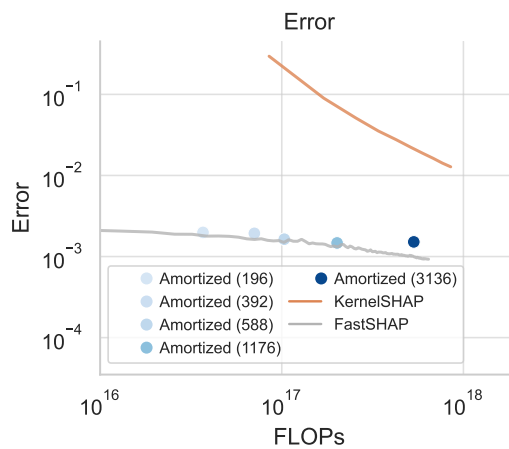


Figure 13: Comparison between stochastic amortization, FastSHAP and KernelSHAP as a function of total FLOPs.

G.2 Banzhaf value and LIME amortization

As we discussed in Section 4.2, Banzhaf value and LIME feature attributions are two XML tasks closely related to Shapley values that can be amortized in a similar fashion. Both are intractable to calculate exactly, but they have (approximately) unbiased estimators that can be used as noisy labels for stochastic amortization. Appendix E describes these estimators, namely the MSR estimator for Banzhaf values [103] and the least squares estimator for LIME [83]. Prior work has also shown that these approaches generate similar outputs [66], although the standard computational approaches are quite different.

In amortizing these methods, one trend we observed is that Banzhaf value and LIME feature attributions have a very different scale from Shapley values. Figure 14 shows that they not only have smaller norm, but that the distribution is concentrated at small magnitudes with a long tail of larger magnitudes. This is troublesome, because our theory is focused on the squared error $\|a(b; \theta) - a(b)\|^2$ (see Section 3), which is dominated by the small number of examples with large magnitudes. As a result, an amortized attribution model can appear to train well by accurately estimating attributions with a large norm, and predicting the remaining attributions to be roughly zero; in doing so it may fail to provide the correct relative feature ordering for most examples, which is more important for practical usage of feature attributions.

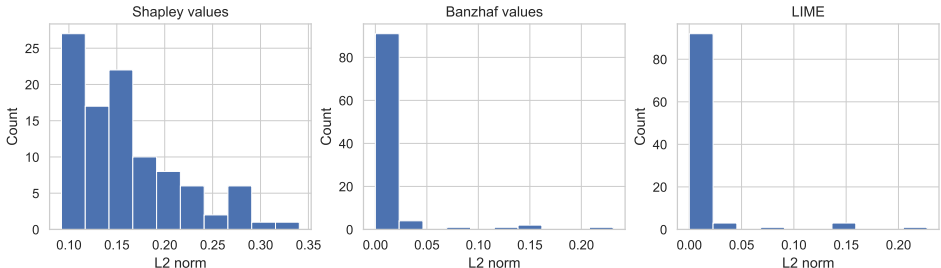


Figure 14: Comparison of the distribution of norms between different feature attribution methods. We plotted the L2 norm of the 100 ground truths for each feature attribution.

The issue described above is precisely what occurs when we amortize Banzhaf value feature attributions, as shown in the top row of Figure 15. We observe that the squared error from our predictions is lower than that of the noisy labels, which is consistent with our theory from Section 3 and our results for Shapley values (Appendix G.1). However, the amortized estimates perform worse than the noisy labels on the remaining metrics, particularly for the correlation scores that evaluate the relative feature ordering in each example’s attributions (see details for the metrics in Appendix F).

To alleviate this issue, we experimented with a per-label normalization that eliminates the long tail of large attribution norms that dominate training: we simply normalized each example’s attributions for each class to have a norm of 1. The results are shown in the bottom row of Figure 15, where we see that the predictions have higher correlation scores than the noisy labels. The squared error is higher for the amortized predictions, and the sign agreement is roughly the same. One issue with this heuristic is that the normalized noisy labels are biased: the normalized noisy label is not an unbiased estimator of the normalized exact label, because the normalization constant is not known a priori and must be calculated using the noisy label. However, the results show that amortization can to some extent denoise inexact labels even with this non-zero bias.

Figure 16 shows similar results as Figure 15 but with LIME feature attributions. We observe the same improvement in squared error from amortization, and a similar degradation in the correlation metrics (Figure 16 top). We then apply the same per-label normalization trick, and we observe a similar modest improvement in the correlation metrics, at least for noisier settings of the least squares estimator (Figure 16 bottom).

Overall, these results show that for the purpose of amortization, the consistency in scale of Shapley values is an unexpected advantage over Banzhaf values and LIME. This property is due in part to the Shapley value’s efficiency axiom [89], which guarantees that the Shapley values sum to the difference between the prediction with all features and no features [68]. In comparison, LIME and Banzhaf values focus on marginal contributions involving roughly half of the features, which in many cases are near zero when the prediction is saturated; for example, previous works have observed that for

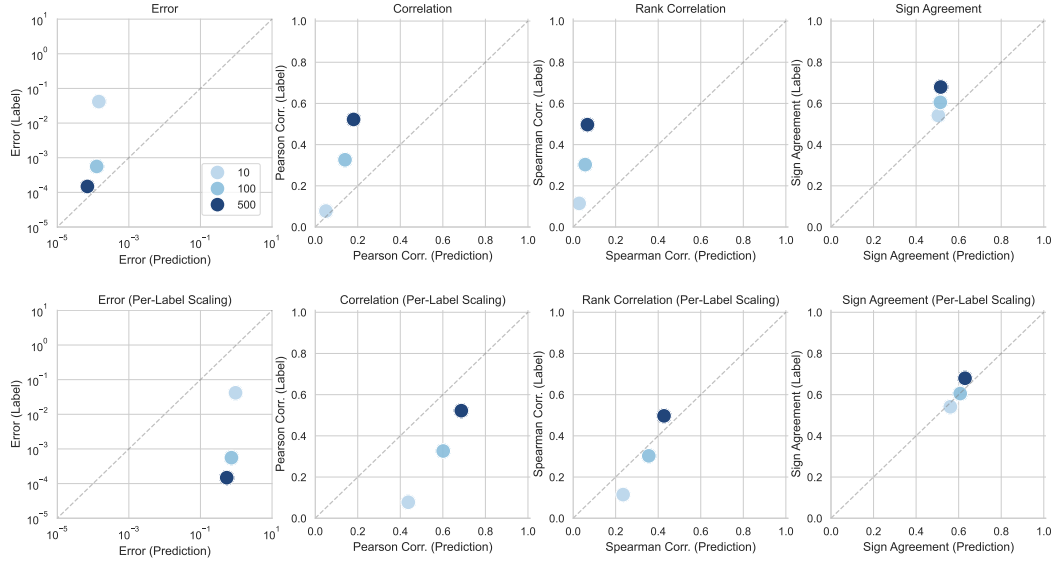


Figure 15: Comparison of the estimation error between noisy labels and amortized predictions for Banzhaf value feature attributions. Noisy labels were generated using the MSR estimator with different numbers of samples. Top: using raw estimates as noisy training labels. Bottom: normalizing each noisy training label separately to have unit norm for each class.

natural images containing relatively large objects, a large portion of the patches must be removed before we observe significant changes in the prediction [48, 18].

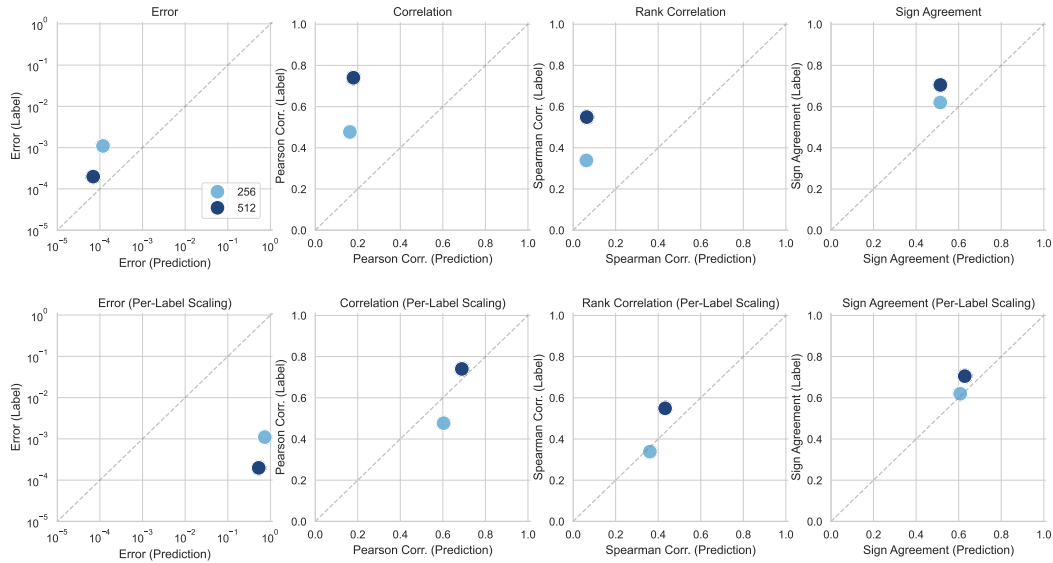


Figure 16: Comparison of the estimation error between noisy labels and amortized predictions for LIME feature attributions. Noisy labels were generated using LIME's least squares estimator with different numbers of samples. Top: using raw estimates as noisy training labels. Bottom: normalizing each noisy training label separately to have unit norm for each class.

G.3 Data valuation

For the data valuation experiments, our first additional result compares amortization to the Monte Carlo estimator when using the MiniBooNE and adult datasets with different numbers of data points. The results are shown in Figure 17 and Figure 18, where we use datasets ranging from 1K to 10K points, and we see that the benefits of amortization increase with the size of the dataset.

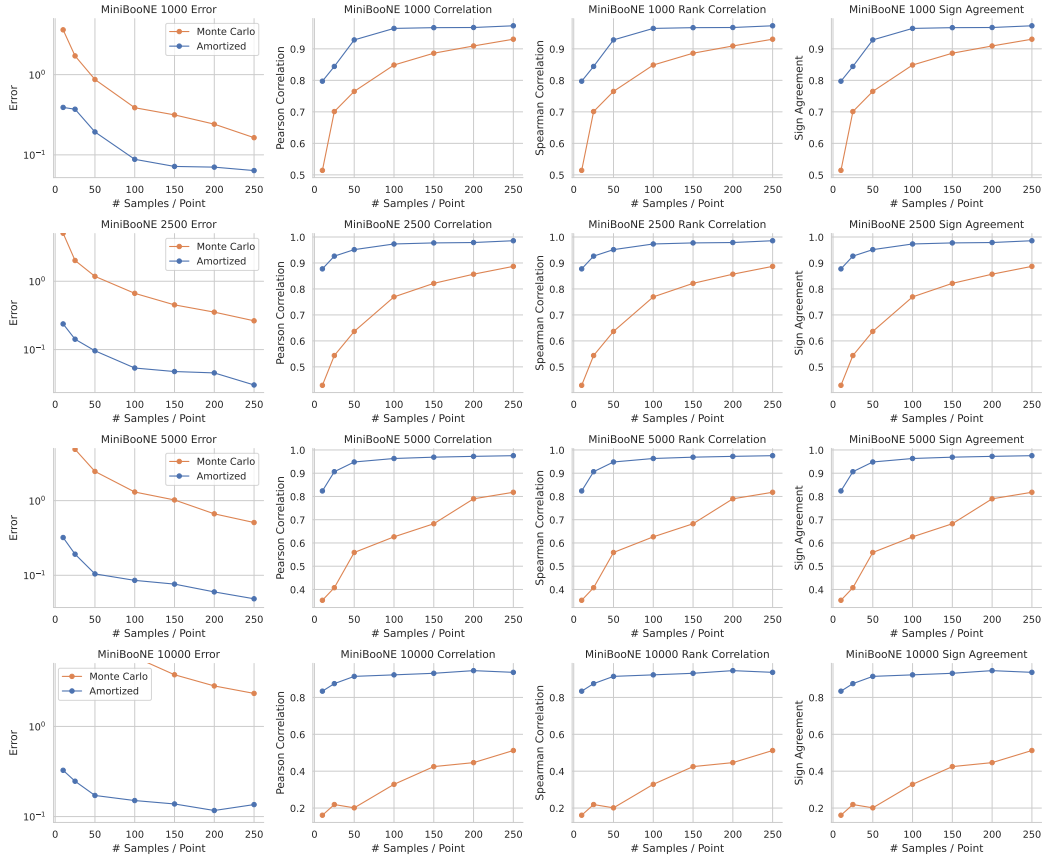


Figure 17: Amortized data valuation for the MiniBooNE dataset with different numbers of training data points (1K to 10K). We show four metrics: squared error (normalized so that the mean valuation scores has error equal to 1), Pearson correlation, Spearman correlation, and sign agreement.

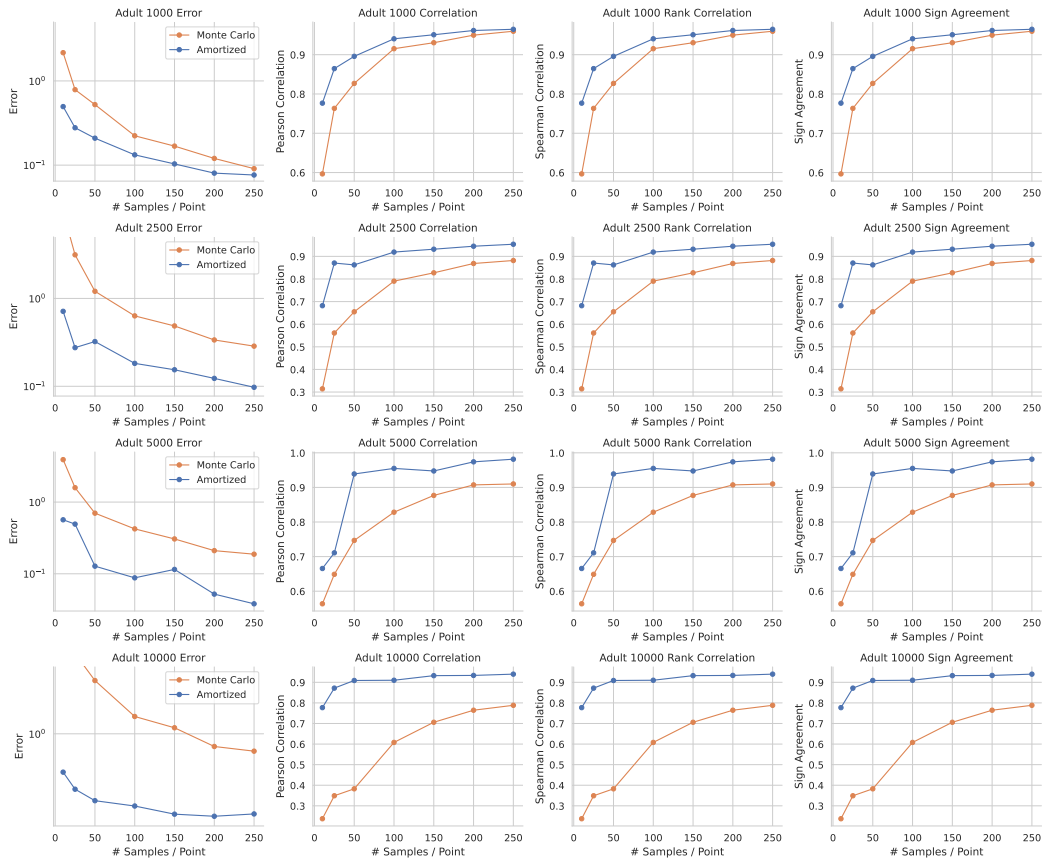


Figure 18: Amortized data valuation for the adult dataset with different numbers of training data points (1K to 10K). We show four metrics: squared error (normalized so that the mean valuation scores has error equal to 1), Pearson correlation, Spearman correlation, and sign agreement.

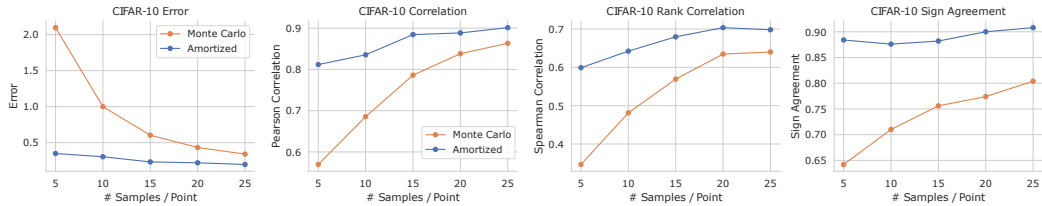


Figure 19: Distributional data valuation for CIFAR-10 with 50K data points. We generate Monte Carlo estimates and amortized estimates with different numbers of samples per data point, and the scores are compared to ground truth values using four metrics.

We also provide more detailed performance metrics for CIFAR-10: Figure 19 shows an expanded version of Figure 5, and we observe that amortization improves upon the Monte Carlo estimator across all four performance metrics.

Next, we consider the use of CIFAR-10 data valuation scores in two downstream tasks. First, we attempt to identify mislabeled examples, which we expect should have large negative valuation scores. Figure 20 shows how quickly each method identifies negative examples when we sort the scores from lowest to highest: our amortized estimates that train with just 5 samples provide the highest accuracy, outperforming Monte Carlo estimates that use as many as 100 samples. This result is consistent across different levels of label noise, where we randomly flip either 10%, 25% or 50% of the labels.

Table 2 performs a similar analysis involving mislabeled examples: we expect these examples to have lower scores than correctly labeled examples, so we use the estimated scores to calculate the AUROC, the AUPR, and the portion of mislabeled examples with negative scores. Across the different levels of label noise, our amortized estimates achieve the best performance according to all three metrics. The strong performance of our amortized data valuation scores is largely due to the improved estimation accuracy, but we expect that it is also due to the valuation network being trained on raw images: the noisy labels are derived from training logistic regression models on pretrained ResNet-50 embeddings, which can lead to somewhat arbitrary valuation score differences between semantically similar examples, and the valuation network may learn a more generalizable notion of data value.

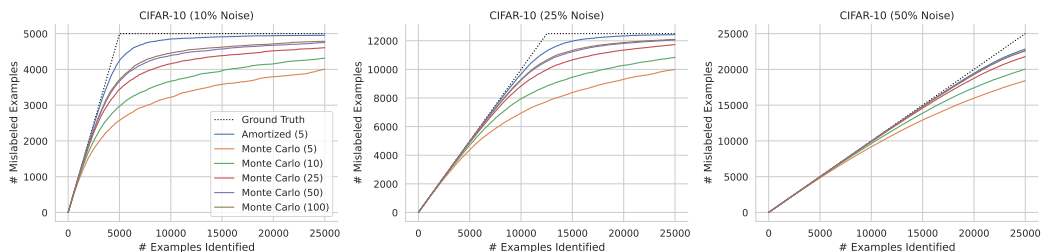


Figure 20: CIFAR-10 mislabeled example identification with different amounts of label noise. We compare the number of mislabeled examples among the lowest-scoring data points, where the amortized model uses just 5 samples for training and the Monte Carlo estimator uses between 5 and 100 samples.

Table 2: Mislabeled example identification accuracy for CIFAR-10 with different amounts of label noise.

	10% Noise			25% Noise			50% Noise		
	AUROC	AUPRC	Negative	AUROC	AUPRC	Negative	AUROC	AUPRC	Negative
Amortized (5)	0.981	0.917	0.974	0.985	0.964	0.973	0.972	0.973	0.915
Monte Carlo (5)	0.782	0.515	0.757	0.802	0.687	0.758	0.815	0.829	0.708
Monte Carlo (10)	0.844	0.619	0.826	0.867	0.784	0.821	0.883	0.892	0.769
Monte Carlo (25)	0.906	0.738	0.883	0.931	0.877	0.888	0.943	0.947	0.841
Monte Carlo (50)	0.934	0.794	0.909	0.956	0.918	0.918	0.965	0.967	0.878
Monte Carlo (100)	0.941	0.808	0.918	0.960	0.923	0.922	0.965	0.967	0.878

Finally, we experiment with using the data valuation scores to improve the dataset. Using the version with 25% label noise, we experiment with several possible modifications to the dataset, where in each case we train 5 models and report the mean cross entropy loss. First, we consider a perfect filtering of the data where we remove all mislabeled examples. Next, we remove different numbers of examples chosen uniformly at random. We also consider removing the examples with the lowest scores according to the Monte Carlo estimates with 10 samples. Finally, we test two possible approaches based on the amortized valuation model: (i) we filter out the lowest scoring examples (“Amortized filtering”), similar to how we use the noisy Monte Carlo estimates; (ii) we attempt to correct the lowest scoring examples using the class that the valuation model predicts to be most valuable (“Amortized cleaning”), which is a unique capability enabled by our valuation model that predicts valuation scores simultaneously for all possible classes (see [Appendix F](#)). The results of this experiment are shown in [Figure 21](#). We see that removing samples at random hurts the model’s performance relative to using all the data, that filtering with the amortized estimates outperforms filtering with the Monte Carlo estimates, and that cleaning the estimates is the best approach by a narrow margin.

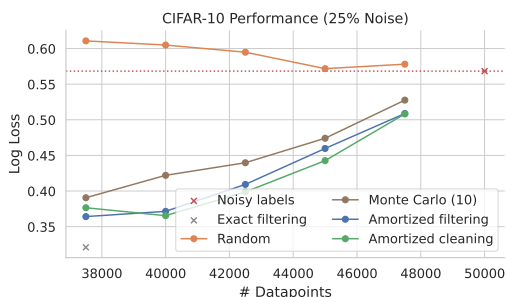


Figure 21: CIFAR-10 performance when training data is removed or adjusted based on estimated valuation scores.

H Broader Impact

Our proposed method aims to make many computationally challenging XML tasks feasible and scalable to large datasets. We expect our research to contribute to a better understanding of ML models, enhancing their transparency and explainability. Additionally, our research could help understand model fairness properties if our method is used to identify undesirable or unfair dependencies of ML models on input features related to protected attributes like race or gender. The potential downside is that if users overly trust our method and rely solely on it for these tasks without careful usage, it could lead to misunderstanding of models and result in harmful outcomes.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [\[Yes\]](#)

Justification: Our claim made in the abstract and introduction is that training amortized models with unbiased noisy oracles is effective and provides significant speed-ups for various XML tasks. This is supported by theoretical and experimental results in [Section 3](#), [Section 4](#), and [Section 5](#).

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: The limitations of our work are discussed in [Section 6](#).

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory Assumptions and Proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [\[Yes\]](#)

Justification: The proofs for all theoretical results are provided in [Appendix D](#).

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental Result Reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [\[Yes\]](#)

Justification: The details for reproducing the experimental results are described in [Section 5](#) and [Appendix F](#).

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Links to our code are provided in the paper. Our experiments use only open-source software and datasets, all of which are referenced in the text.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental Setting/Details

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: Details of the experimental settings are discussed in [Section 5](#) and [Appendix F](#).

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment Statistical Significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: We performed experiments across multiple XML tasks, noisy oracles, noise levels, model architectures, and datasets, so performing multiple runs for each experiment would be too computationally expensive.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments Compute Resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: Information on compute resources is provided in [Appendix F](#).

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code Of Ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: Our proposed method aims to make computationally challenging XML tasks feasible, so our research promotes transparency and interpretability in ML.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader Impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: The potential positive and negative societal impacts of our work are discussed in [Appendix H](#).

Guidelines:

- The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: Our paper does not release a new dataset or model.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The open-source packages and datasets we used are cited in [Appendix F](#).

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The code we provide contains usage instructions and scripts for running experiments.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: Our paper does not involve crowdsourcing experiments or research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: Our paper does not involve human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.