FASTSHAP: REAL-TIME SHAPLEY VALUE ESTIMATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Although Shapley values are widely used to explain black-box models, they are costly to calculate and thus impractical in settings that involve large deep learning models. We introduce FastSHAP, a new method for estimating Shapley values in a single forward pass using a learned explainer model. To enable efficient training without requiring ground truth Shapley values, we develop a way to train FastSHAP via stochastic gradient optimization using a weighted least squares-like objective function. In our experiments with tabular and image datasets, we compare FastSHAP to existing estimation approaches and find that it generates accurate explanations with an orders-of-magnitude speedup.

1 INTRODUCTION

With the proliferation of black-box models, Shapley values (Shapley, 1953) have emerged as a popular explanation approach due to their strong theoretical properties (Lipovetsky and Conklin, 2001; Štrumbelj and Kononenko, 2014; Datta et al., 2016; Lundberg and Lee, 2017). In practice, however, Shapley value-based explanations are known to have high computational complexity, with an exact calculation requiring an exponential number of model evaluations (Van den Broeck et al., 2021). Speed becomes a critical issue as models increase in size and dimensionality, and for the largest models in areas such as computer vision and natural language processing, there is an unmet need for significantly faster Shapley value approximations that maintain high accuracy.

Recent work has addressed the computational challenges with Shapley values using two main approaches. First, many works have proposed *stochastic estimators* (Castro et al., 2009; Štrumbelj and Kononenko, 2014; Lundberg and Lee, 2017; Covert et al., 2020b) that rely on sampling either feature subsets or permutations; though often consistent, these estimators require many model evaluations and impose an undesirable trade-off between run-time and accuracy. Second, some works have proposed *model-specific approximations*, e.g., for trees (Lundberg et al., 2020) or neural networks (Shrikumar et al., 2017; Chen et al., 2018b; Ancona et al., 2019; Wang et al., 2021); while generally faster, these can still require many model evaluations, often induce bias, and typically lack flexibility regarding the handling held-out features—a subject of ongoing debate in the field (Aas et al., 2019; Janzing et al., 2020; Covert et al., 2020a; Frye et al., 2020).

Here, we introduce a new approach for efficient Shapley value estimation: *to achieve the fastest possible run-time, we propose learning a separate explainer model that outputs precise Shapley value estimates in a single forward pass.* Naïvely, such a learning-based approach would seem to require a large training set of ground truth Shapley values, which would be computationally intractable. Instead, our approach trains an explainer model by minimizing an objective function inspired by the Shapley value's weighted least squares characterization (Charnes et al., 1988), which enables efficient gradient-based optimization.

Our contributions. We introduce FastSHAP, an amortized approach for generating real-time Shapley value explanations.[1] We derive a learning objective from the Shapley value's weighted least squares characterization and investigate several ways to reduce gradient variance during training. Our experiments show that FastSHAP provides accurate Shapley value estimates with an orders-of-magnitude speedup relative to non-amortized estimation approaches. We also find that FastSHAP generates high-quality image explanations (fig. 1) that outperform gradient-based methods (e.g., IntGrad and GradCAM) on quantitative inclusion and exclusion metrics.

[1] https://github.com/iclr1814/fastshap

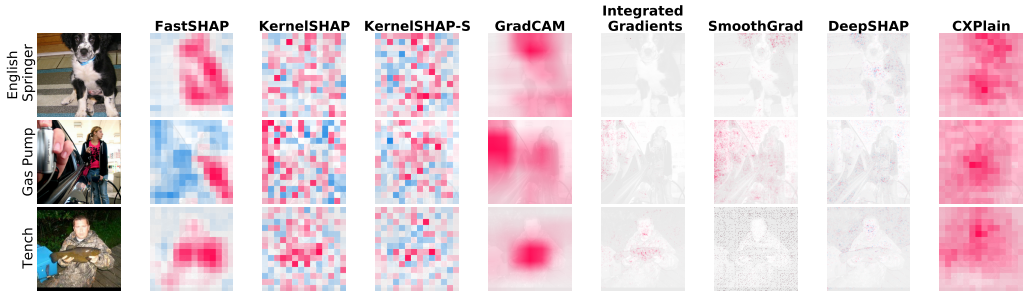


Figure 1: **Explanations generated by each method for Imagenette images.**

2 BACKGROUND

In this section, we introduce notation used throughout the paper and provide an overview of Shapley values and their weighted least squares characterization. Let $\mathbf{x} \in \mathcal{X}$ be a random vector consisting of d features, or $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_d)$, and let $\mathbf{y} \in \mathcal{Y} = \{1, \ldots, K\}$ be the response variable for a classification problem. We use $\mathbf{s} \in \{0, 1\}^d$ to denote subsets of the indices $\{1, \ldots, d\}$ and define $\mathbf{x}_s := \{\mathbf{x}_i\}_{i:s_i=1}$. The symbols $\mathbf{x}, \mathbf{y}, \mathbf{s}$ are random variables and x, y, s denote possible values. We use $\mathbf{1}$ and $\mathbf{0}$ to denote vectors of ones and zeros in $\mathbb{R}^d$, so that $\mathbf{1}^\top s$ is a subset's cardinality, and we use e_i to denote the ith standard basis vector. Finally, $f(\mathbf{x}; \eta) : \mathcal{X} \mapsto \Delta^{K-1}$ is a model that outputs a probability distribution over $\mathbf{y}$ given $\mathbf{x}$, and $f_y(\mathbf{x}; \eta)$ is the probability for the yth class.

2.1 SHAPLEY VALUES

Shapley values were originally developed as a credit allocation method in cooperative game theory (Shapley, 1953), but they have since been adopted to explain predictions from black-box machine learning models (Štrumbelj and Kononenko, 2014; Datta et al., 2016; Lundberg and Lee, 2017). For any value function (or set function) $v : 2^d \mapsto \mathbb{R}$, the Shapley values $\phi(v) \in \mathbb{R}^d$, or $\phi_i(v) \in \mathbb{R}$ for each feature $i = 1, \ldots, d$, are given by the formula

$$\phi_i(v) = \frac{1}{d} \sum_{s_i \neq 1} \binom{d-1}{\mathbf{1}^\top s}^{-1} \Big(v(s + e_i) - v(s) \Big). \tag{1}$$

The difference $v(s + e_i) - v(s)$ represents the ith feature's contribution to the subset s, and the summation represents a weighted average across all subsets that do not include i. In the model explanation context, the value function is chosen to represent how an individual prediction varies as different subsets of features are removed. For example, given an input-output pair (x, y), the prediction for the yth class can be represented by a value function $v_{x,y}$ defined as

$$v_{x,y}(s) = \text{link} \left(\mathop{\mathbb{E}}_{p(\mathbf{x}_{\mathbf{1}-s})} \left[f_y \left(x_s, \mathbf{x}_{\mathbf{1}-s}; \eta \right) \right] \right), \tag{2}$$

where the held out features $\mathbf{x}_{\mathbf{1}-s}$ are marginalized out using their joint marginal distribution $p(\mathbf{x}_{\mathbf{1}-s})$, and a link function (e.g., logit) is applied to the model output. Recent work has debated the properties of different value function formulations, particularly the choice of how to remove features (Aas et al., 2019; Janzing et al., 2020; Frye et al., 2020; Covert et al., 2020a). Regardless of the formulation, this approach to model explanation enjoys several useful theoretical properties due to its use of Shapley values: for example, the attributions are zero for irrelevant features, and they are guaranteed to sum to the model's prediction. We direct readers to prior work for a detailed discussion of these properties (Lundberg and Lee, 2017; Covert et al., 2020a).

Unfortunately, Shapley values also introduce computational challenges: the summation in eq. (1) involves an exponential number of subsets, making it infeasible to calculate for large d. Fast approximations are therefore required in practice, as we discuss next.

2.2 KERNELSHAP

KernelSHAP (Lundberg and Lee, 2017) is a popular Shapley value implementation that relies on an alternative Shapley value interpretation. Given a value function $v_{x,y}(\mathbf{s})$, eq. (1) shows that the

values $\phi(v_{x,y})$ are the features' weighted average contributions; equivalently, their weighted least squares characterization says that they are the solution to an optimization problem over $\phi_{x,y} \in \mathbb{R}^d$,

$$\phi(v_{x,y}) = \underset{\phi_{x,y}}{\arg\min} \underset{p(\mathbf{s})}{\mathbb{E}} \left[\left(v_{x,y}(\mathbf{s}) - v_{x,y}(\mathbf{0}) - \mathbf{s}^\top \phi_{x,y} \right)^2 \right] \tag{3}$$

$$\text{s.t.} \quad \mathbf{1}^\top \phi_{x,y} = v_{x,y}(\mathbf{1}) - v_{x,y}(\mathbf{0}), \qquad \text{(Efficiency constraint)}$$

where the distribution $p(\mathbf{s})$ is defined as

$$p(s) \propto \frac{d-1}{\binom{d}{\mathbf{1}^\top s} \cdot \mathbf{1}^\top s \cdot (d - \mathbf{1}^\top s)} \qquad \text{(Shapley kernel)}$$

for s such that $0 < \mathbf{1}^\top s < d$ (Charnes et al., 1988). Based on this view of the Shapley value, KernelSHAP is a stochastic estimator that solves an approximate version of eq. (3) given some number of subsets sampled from $p(\mathbf{s})$. Although the estimator is consistent and empirically unbiased (Covert and Lee, 2021), KernelSHAP often requires many samples to achieve an accurate estimate, and it must solve eq. (3) separately for each input-output pair (x, y); as a result, it is unacceptably slow for some use cases, particularly with models that are slow to evaluate. Our approach builds on KernelSHAP, leveraging the Shapley value's weighted least squares characterization to design a faster, amortized estimation approach.

3 FastSHAP

We now introduce FastSHAP, a method that amortizes the cost of generating Shapley values across many data samples. FastSHAP has two main advantages over existing approaches: (1) it avoids solving separate optimization problems for each input to be explained, and (2) it can use similar data points to efficiently learn the Shapley value function $\phi(v_{x,y})$.

3.1 Amortizing Shapley values

In our approach, we propose generating Shapley value explanations using a learned parametric function $\phi_{\text{fast}}(\mathbf{x}, \mathbf{y}; \theta) : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}^d$. Once trained, the parametric function can generate explanations in a single forward pass, providing a significant speedup over methods that approximate Shapley values separately for each sample (x, y). Rather than using a dataset of ground truth Shapley values, we train $\phi_{\text{fast}}(\mathbf{x}, \mathbf{y}; \theta)$ by penalizing its predictions according to the weighted least squares objective in eq. (3), or by minimizing the following loss,

$$\mathcal{L}(\theta) = \underset{p(\mathbf{x})}{\mathbb{E}} \underset{\text{Unif}(\mathbf{y})}{\mathbb{E}} \underset{p(\mathbf{s})}{\mathbb{E}} \left[\left(v_{\mathbf{x},\mathbf{y}}(\mathbf{s}) - v_{\mathbf{x},\mathbf{y}}(\mathbf{0}) - \mathbf{s}^\top \phi_{\text{fast}}(\mathbf{x}, \mathbf{y}; \theta) \right)^2 \right], \tag{4}$$

where $\text{Unif}(\mathbf{y})$ represents a uniform distribution over classes. If the model's predictions are forced to satisfy the Efficiency constraint, then given a large enough dataset and a sufficiently expressive model class for ϕ_{fast}, the global optimizer $\phi_{\text{fast}}(\mathbf{x}, \mathbf{y}; \theta^*)$ is a function that outputs exact Shapley values (see proof in appendix A). Formally, the global optimizer satisfies

$$\phi_{\text{fast}}(x, y; \theta^*) = \phi(v_{x,y}) \quad \forall\, x \in \mathcal{X}, y \in \mathcal{Y}. \tag{5}$$

We explore two approaches to address the efficiency requirement. First, we can enforce efficiency by adjusting the Shapley value predictions using their *additive efficient normalization* (Ruiz et al., 1998), which applies the following operation to the model's outputs before evaluating the loss:

$$\phi^{\text{eff}}_{\text{fast}}(x, y; \theta) = \phi_{\text{fast}}(x, y; \theta) + \frac{1}{d} \underbrace{\left(v_{x,y}(\mathbf{1}) - v_{x,y}(\mathbf{0}) - \mathbf{1}^\top \phi_{\text{fast}}(x, y; \theta) \right)}_{\text{Efficiency gap}}. \tag{6}$$

The normalization step can be applied at inference time and optionally during training; in appendix B, we show that the additive efficient normalization is guaranteed to make the estimates closer to the true Shapley values. Second, we can relax the efficiency property by augmenting $\mathcal{L}(\theta)$ with a penalty on the efficiency gap (see eq. (6)); the penalty requires a parameter $\gamma > 0$, and as we set $\gamma \to \infty$ we can guarantee that efficiency holds (see appendix A). Algorithm 1 summarizes our training approach.

Empirical considerations. Optimizing $\mathcal{L}(\theta)$ using a single set of samples (x, y, s) is problematic because of high variance in the gradients, which can lead to poor optimization. We therefore consider several steps to reduce gradient variance. First, as is conventional in deep learning, we minibatch across multiple samples from $p(\mathbf{x})$. Next, when possible, we calculate the loss jointly across all classes $y \in \{1, \ldots, K\}$. Then, we experiment with using multiple samples $s \sim p(\mathbf{s})$ for each input sample x. Finally, we explore *paired sampling*, where each sample s is paired with its complement $\mathbf{1} - s$, which has been shown to reduce KernelSHAP's variance (Covert and Lee, 2021). Appendix C shows proofs that these steps are guaranteed to reduce gradient variance, and ablation experiments in appendix D demonstrate their improvement on FastSHAP's accuracy.

3.2 A DEFAULT VALUE FUNCTION FOR FASTSHAP

FastSHAP has the flexibility to work with any value function $v_{x,y}(\mathbf{s})$. Here, we describe a default value function that is useful for explaining predictions from a classification model.

The value function's aim is to assess, for each subset s, the classification probability when only the features $\mathbf{x}_s$ are observed. Because most models $f(\mathbf{x}; \eta)$ do not support making predictions without all the features, we require an approximation that simulates the inclusion of only $\mathbf{x}_s$ (Covert et al., 2020a). To this end, we use a supervised surrogate model (Frye et al., 2020; Jethani et al., 2021) to approximate marginalizing out the remaining features $\mathbf{x}_{\mathbf{1}-s}$ using their conditional distribution.

Algorithm 1: FastSHAP training

Input: Value function $v_{x,y}$, learning rate α
Output: FastSHAP explainer $\phi_{\text{fast}}(\mathbf{x}, \mathbf{y}; \theta)$
initialize $\phi_{\text{fast}}(\mathbf{x}, \mathbf{y}; \theta)$
while *not converged* **do**
  sample $x \sim p(\mathbf{x})$, $y \sim \text{Unif}(\mathbf{y})$, $s \sim p(\mathbf{s})$
  predict $\hat{\phi} \leftarrow \phi_{\text{fast}}(x, y; \theta)$
  if *normalize* **then**
    set $\hat{\phi} \leftarrow$
    $\hat{\phi} + d^{-1}\left(v_{x,y}(\mathbf{1}) - v_{x,y}(\mathbf{0}) - \mathbf{1}^T\hat{\phi}\right)$
  end
  calculate
  $\mathcal{L} \leftarrow \left(v_{x,y}(s) - v_{x,y}(\mathbf{0}) - s^T\hat{\phi}\right)^2$
  update $\theta \leftarrow \theta - \alpha\nabla_\theta\mathcal{L}$
end

Separate from the original model $f(\mathbf{x}; \eta)$, the *surrogate* model $p_{\text{surr}}(\mathbf{y} \mid m(\mathbf{x}, \mathbf{s}); \beta)$ takes as input a vector of masked features $m(x, s)$, where the masking function m replaces features x_i such that $s_i = 0$ with a `[mask]` value that is not in the support of $\mathcal{X}$. Similar to prior work (Frye et al., 2020; Jethani et al., 2021), the parameters β are learned by minimizing the following loss function:

$$\mathcal{L}(\beta) = \mathop{\mathbb{E}}_{p(\mathbf{s})}\Big[D_{\text{KL}}\big(f(\mathbf{x}; \eta) \;||\; p_{\text{surr}}(\mathbf{y} \mid m(\mathbf{x}, \mathbf{s}); \beta)\big)\Big]. \tag{7}$$

It has been shown that the global optimizer to eq. (7), or $p_{\text{surr}}(\mathbf{y} \mid m(\mathbf{x}, \mathbf{s}); \beta^*)$, is equivalent to marginalizing out features from $f(\mathbf{x}; \eta)$ with their conditional distribution (Covert et al., 2020a):

$$p_{\text{surr}}(y \mid m(x, s); \beta^*) = \mathbb{E}[f_y(\mathbf{x}; \eta) \mid \mathbf{x}_s = x_s]. \tag{8}$$

The choice of distribution over $p(\mathbf{s})$ does not affect the global optimizer of eq. (7), but we use the Shapley kernel to put more weight on subsets likely to be encountered when training FastSHAP. We use the surrogate model as a default choice for two reasons. First, it requires a single prediction for each evaluation of $v_{x,y}(s)$ (unlike marginalizing out features using their marginal distribution, Lundberg and Lee 2017; Janzing et al. 2020), which permits fast training. Second, it yields explanations that reflect the model's dependence on the *information* from each feature rather than its *algebraic* dependence (Covert et al., 2020a; Frye et al., 2020).

4 RELATED WORK

Recent work on Shapley value explanations has largely focused on how to remove features (Aas et al., 2019; Frye et al., 2020; Covert et al., 2020a) and how to approximate Shapley values efficiently (Chen et al., 2018b; Ancona et al., 2019; Lundberg et al., 2020; Covert and Lee, 2021). Model-specific approximations are relatively fast, but they often introduce bias and are entangled with specific feature removal approaches (Shrikumar et al., 2017; Ancona et al., 2019; Lundberg et al., 2020). In contrast, model-agnostic stochastic approximations are more flexible, but they must trade off run-time and accuracy in the explanation. For example, KernelSHAP samples subsets to approximate the solution to a weighted least squares problem (Lundberg and Lee, 2017), while other

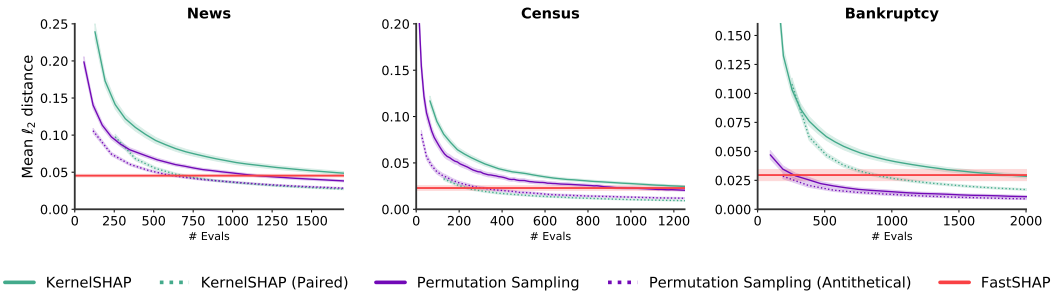


Figure 2: **Comparison of Shapley value approximation accuracy across methods.** Using three datasets, we measure the distance of each method's estimates to the ground truth as a function of the number of model evaluations. FastSHAP is represented by a horizontal line since it requires only a single forward pass. The baselines require $200-2000\times$ model evaluations to achieve FastSHAP's level of accuracy.

approaches sample marginal contributions (Castro et al., 2009; Štrumbelj and Kononenko, 2014) or feature permutations (Illés and Kerényi, 2019; Mitchell et al., 2021). FastSHAP trains a model to output an estimate that would otherwise require hundreds of model evaluations, and, unlike other fast approximations, it is agnostic to the model class and feature removal approach.

Other methods have been proposed to generate explanations using separate explainer models. These are referred to as *amortized explanation methods* (Covert et al., 2020a; Jethani et al., 2021), and they include several approaches (Dabkowski and Gal, 2017; Chen et al., 2018a; Yoon et al., 2018; Schwab and Karlen, 2019; Schulz et al., 2020; Jethani et al., 2021) that are comparable to gradient-based methods (Sundararajan et al., 2017; Selvaraju et al., 2017) in terms of compute time. Notably, one approach generates a training dataset of ground truth explanations and then learns an explainer model to output explanations directly (Schwab and Karlen, 2019)—a principle that can be applied with any attribution method, at least in theory. For Shapley values, generating a large training set would be very costly, so FastSHAP sidesteps the need for a training set using a stochastic optimization approach.

5 Structured Data Experiments

We analyze FastSHAP's performance by comparing it to several well-studied baselines. First, we evaluate its accuracy on tabular (structured) datasets by comparing its outputs to the ground truth Shapley values. Then, to disentangle the benefits of amortization from the in-distribution value function, we make the same comparisons using different value function formulations $v_{x,y}(\mathbf{s})$. Unless otherwise stated, we use the surrogate model value function introduced in section 3.2. Later, in section 6, we test FastSHAP's ability to produce image explanations.

Baseline methods. To contextualize FastSHAP's approximation accuracy, we compare it to several non-amortized stochastic estimators. First, we compare to KernelSHAP (Lundberg and Lee, 2017) and its acceleration that uses paired sampling (Covert and Lee, 2021). Next, we compare to a permutation sampling approach and its acceleration that uses antithetical sampling (Mitchell et al., 2021). As a performance metric, we calculate the proximity to Shapley values that were obtained by running KernelSHAP to convergence; we use these values as our ground truth because KernelSHAP is known to converge to the true Shapley values given infinite samples (Covert and Lee, 2021). These baselines were all run using an open-source implementation.[2]

Implementation details. We use either neural networks or tree-based models for each of $f(\mathbf{x};\eta)$ and $p_{\text{surr}}(\mathbf{y} \mid m(\mathbf{x},\mathbf{s});\beta)$. The FastSHAP explainer model $\phi_{\text{fast}}(\mathbf{x},\mathbf{y};\theta)$ is implemented with a network $g(\mathbf{x};\theta) : \mathcal{X} \to \mathbb{R}^d \times \mathcal{Y}$ that outputs a vector of Shapley values for every $y \in \mathcal{Y}$; deep neural networks are ideal for FastSHAP because they have high representation capacity, they can provide many-to-many mappings, and they are amenable to stochastic optimization. Appendix D contains more details about our implementation, including model classes, network architectures and training hyperparameters.

We also perform a series of experiments to determine several training hyperparameters for FastSHAP, exploring (1) whether or not to use paired sampling, (2) the number of subset samples to use, and

[2] https://github.com/iancovert/shapley-regression/ (License: MIT)

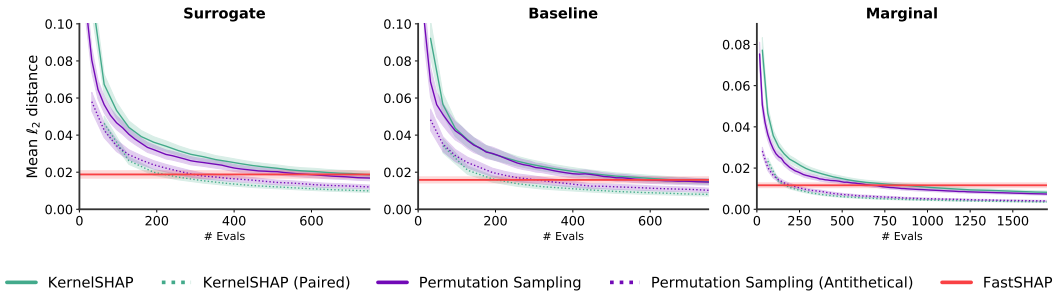


Figure 3: **FastSHAP approximation accuracy for different value functions.** Using the `marketing` dataset, we find that FastSHAP provides accurate Shapley value estimates regardless of the value function (surrogate, marginal, baseline), with the baselines requiring $200-1000\times$ model evaluations to achieve FastSHAP's level of accuracy. Error bars represent 95% confidence intervals.

(3) how to best enforce the efficiency constraint. Based on the results (see appendix D), we use the following settings for our tabular data experiments: we use paired sampling, between 32-64 samples of $\mathbf{s}$ per $\mathbf{x}$ sample, additive efficient normalization during both training and inference, and we set $\gamma = 0$ (since the normalization step is sufficient to enforce efficiency).

5.1 Accuracy of FastSHAP explanations

Here, we test whether FastSHAP's estimates are close to the ground truth Shapley values. Our experiments use data from a 1994 United States `census`, a bank `marketing` campaign, `bankruptcy` statistics, and online `news` articles (Dua and Graff, 2017). The `census` data contains 12 input features, and the binary label indicates whether a person makes over \$50K a year (Kohavi et al., 1996). The `marketing` dataset contains 17 input features, and the label indicates whether the customer subscribed to a term deposit (Moro et al., 2014). The `bankruptcy` dataset contains 96 features describing various companies and whether they went bankrupt (Liang et al., 2016). The `news` dataset contains 60 numerical features about articles published on Mashable, and our label indicates whether the share count exceeds the median number (1400) (Fernandes et al., 2015). The datasets were each split 80/10/10 for training, validation and testing.

In fig. 2, we show the distance of each method's estimates to the ground truth as a function of the number of model evaluations for the `news`, `census` and `bankruptcy` datasets. Figure 3 shows results for the `marketing` dataset with three different value functions (see section 5.2). For the baselines, each sample $\mathbf{s}$ requires evaluating the model given a subset of features, but since FastSHAP requires only a single forward pass of $\phi_{\text{fast}}(\mathbf{x}, \mathbf{y}; \theta)$, we show it as a horizontal line.

To reach FastSHAP's level of accuracy on the `news`, `census` and `bankruptcy` datasets, KernelSHAP requires between 1,200-2,000 model evaluations; like prior work (Covert and Lee, 2021), we find that paired sampling improves KernelSHAP's rate of convergence, helping reach FastSHAP's accuracy in 250-1,000 model evaluations. The permutation sampling baselines tend to be faster: the original version requires between 300-1,000 evaluations, and antithetical sampling takes 200-500 evaluations to reach an accuracy equivalent to FastSHAP. Across all four datasets, however, FastSHAP achieves its level of accuracy at least at least $600\times$ faster than the original version of KernelSHAP, and $200\times$ faster than the best non-amortized baseline.

5.2 Disentangling amortization and the choice of value function

In this experiment, we verify that FastSHAP produces accurate Shapley value estimates regardless of the choice of value function. We use the `marketing` dataset for this experiment and test the following value functions:

1. (Surrogate/In-distribution) $v_{x,y}(s) = p_{\text{surr}}(y \mid m(x, s); \beta)$
2. (Marginal/Out-of-distribution) $v_{x,y}(s) = \mathbb{E}_{p(\mathbf{x}_{\mathbf{1}-s})}\left[f_y\left(x_s, \mathbf{x}_{\mathbf{1}-s}; \eta\right)\right]$
3. (Baseline removal) $v_{x,y}(s) = f_y(x_s, x^b_{\mathbf{1}-s}; \eta)$, where $x^b \in \mathcal{X}$ are fixed baseline values (the mean for continuous features and mode for discrete ones)

In fig. 3 we compare FastSHAP to the same non-amortized baseline methods, where each method generates Shapley value estimates using the value functions listed above. The results show that

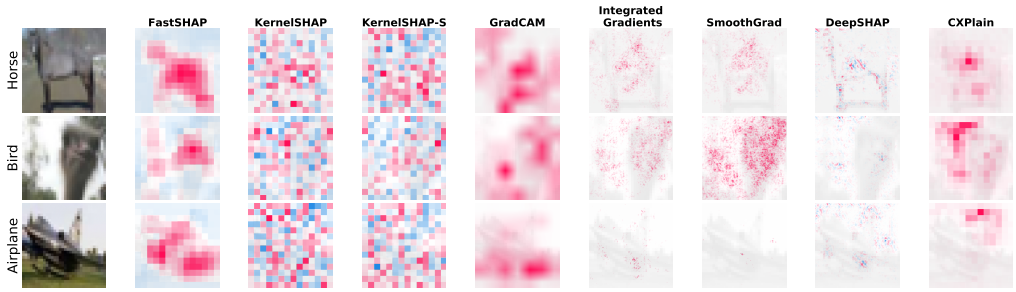


Figure 4: **Explanations generated by each method for CIFAR-10 images.**

FastSHAP maintains the same computational advantage across all three cases: to achieve the same accuracy as FastSHAP's single forward pass, the baseline methods require at least 200 model evaluations, but in some cases up to nearly 1,000.

6 Image Experiments

Images represent a challenging setting for Shapley value estimators due to their high dimensionality and the computational cost of model evaluation. We therefore compare FastSHAP to KernelSHAP on two image datasets. We also consider several widely used gradient-based explanation methods, because they are the most commonly used methods for explaining image classifiers.

6.1 Datasets

We consider two popular image datasets for our experiments. **CIFAR-10** (Krizhevsky et al., 2009) contains 60,000 32×32 images across 10 classes, and we use 50,000 samples for training and 5,000 samples each for validation and testing. Each image is resized to 224×224 using bilinear interpolation to interface with the ResNet-50 architecture (He et al., 2016). Figure 4 shows example CIFAR-10 explanations generated by each method. The **Imagenette** dataset (Howard and Gugger, 2020), a subset of 10 classes from the ImageNet dataset, contains 13,394 total images. Each image is cropped to keep the 224×224 central region, and the data is split 9,469/1,963/1,962. Example Imagenette explanations are shown in fig. 1.

6.2 Explanation methods

We test three Shapley value estimators, FastSHAP, KernelSHAP, and DeepSHAP (Lundberg and Lee, 2017), where the last is an existing approximation designed for neural networks. We test KernelSHAP with the zeros baseline value function, which we refer to simply as KernelSHAP, and with the in-distribution surrogate value function, which we refer to as KernelSHAP-S. We also compare these methods to the gradient-based explanation methods GradCAM (Selvaraju et al., 2017), SmoothGrad (Smilkov et al., 2017) and IntGrad (Sundararajan et al., 2017). Gradient-based methods generate explanations relatively quickly and have therefore been widely adopted for explaining image classifiers. Finally, we also compare to CXPlain (Schwab and Karlen, 2019), an amortized explanation method that generates attributions that are not based on Shapley values.

Implementation details. The models $f(\mathbf{x}; \eta)$ and $p_{\text{surr}}(\mathbf{y} \mid m(\mathbf{x}, \mathbf{s}); \beta)$ are both ResNet-50 networks (He et al., 2016) pretrained on ImageNet and fine-tuned on the corresponding imaging dataset. FastSHAP, CXPlain, and KernelSHAP are all implemented to output 14×14 superpixel attributions for each class. For FastSHAP, we parameterize $\phi_{\text{fast}}(\mathbf{x}, \mathbf{y}; \theta)$ to output superpixel attributions: we use an identical pretrained ResNet-50 but replace the final layers with a 1×1 convolutional layer so that the output is $14 \times 14 \times K$ (see details appendix D). We use an identical network to produce attributions for CXPlain. For FastSHAP, we do not use additive efficient normalization, and we set $\gamma = 0$; we find that this relaxation of the Shapley value's efficiency property does not inhibit FastSHAP's ability to produce high-quality image explanations. KernelSHAP and KernelSHAP-S are implemented using the `shap`[3] package's default parameters, and GradCAM, SmoothGrad, and IntGrad are implemented using the `tf-explain`[4] package's default parameters.

[3] https://shap.readthedocs.io/en/latest/ (License: MIT)
[4] https://tf-explain.readthedocs.io/en/latest/ (License: MIT)

6.3 QUALITATIVE REMARKS

Explanations generated by each method are shown in fig. 4 for CIFAR-10 and fig. 1 for Imagenette (see appendix E for more examples). While a qualitative evaluation is insufficient to draw conclusions about each method, we offer several remarks on these examples. FastSHAP, and to some extent GradCAM, appear to reliably highlight the important objects, while the KernelSHAP explanations are noisy and fail to localize important regions. To a lesser extent, CXPlain occasionally highlights important regions. In comparison, the remaining methods (SmoothGrad, IntGrad and DeepSHAP) are granulated and highlight only small parts of the key objects. Next, we consider quantitative metrics that test these observations more reliably.

6.4 QUANTITATIVE EVALUATION

Evaluating the quality of Shapley value estimates requires access to ground truth Shapley values, which is computationally infeasible for images. Instead, we use two metrics that evaluate an explanation's ability to identify informative image regions. These metrics build on several recent proposals (Petsiuk et al., 2018; Hooker et al., 2018; Jethani et al., 2021) and evaluate the model's classification accuracy after including or excluding pixels according to their estimated importance.

Similar to Jethani et al. (2021), we begin by training a single evaluation model p_{eval} to approximate the $f(\mathbf{x}; \eta)$ model's output given a subset of features; this serves as an alternative to training separate models on each set of features (Hooker et al., 2018) and offers a more realistic option than masking features with zeros (Schwab and Karlen, 2019). This procedure is analogous to the p_{surr} training procedure in section 3.2, except it sets the subset distribution to $p(\mathbf{s}) = \text{Uniform}(\{0, 1\}^d)$ to ensure all subsets are equally weighted.

Next, we analyze how the model's predictions change as we remove either important or unimportant features according to each explanation. Using a set of 1,000 images, each image is first labeled by the original model $f(\mathbf{x}; \eta)$ using the most likely predicted class. We then use explanations generated by each method to produce feature rankings and compute the top-1 accuracy (a measure of agreement with the original model) as we either include or exclude the most important features, ranging from 0-100%. The area under each curve (AUC) is termed the *Inclusion AUC* or *Exclusion AUC*.

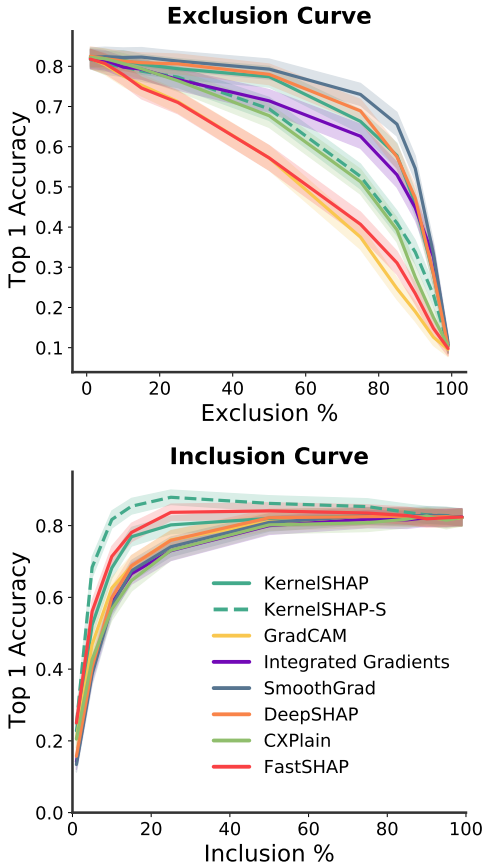


Figure 5: **Imagenette inclusion and exclusion curves.** The change in top-1 accuracy as an increasing percentage of the pixels estimated to be important are excluded (top) or included (bottom).

These metrics match the idea that an accurate image explanation should (1) maximally degrade the performance of p_{eval} when important features are excluded, and (2) maximally improve the performance of p_{eval} when important features are included (Petsiuk et al., 2018; Hooker et al., 2018). The explanations are evaluated by removing superpixels; for gradient-based methods, we coarsen the explanations using the sum total importance within each superpixel. In appendix E, we replicate these metrics using log-odds rather than top-1 accuracy, finding a similar ordering among methods.

Results. Table 1 shows the Inclusion and Exclusion AUC achieved by each method for both CIFAR-10 and Imagenette. In fig. 5, we also present the curves used to generate these AUCs for Imagenette. Lower Exclusion AUCs and higher Inclusion AUCs are better. These results show that FastSHAP outperforms all baseline methods when evaluated with Exclusion AUC: when the pixels identified as important by FastSHAP are removed from the images, the sharpest decline in top-1 accuracy is observed. Additionally, FastSHAP performs well when evaluated on the basis of Inclusion AUC, second only to KernelSHAP-S.

Table 1: **Exclusion and Inclusion AUCs.** Evaluation of each method on the basis of Exclusion AUC (lower is better) and Inclusion AUC (higher is better) calculated using top-1 accuracy. Parentheses indicate 95% confidence intervals, and the best methods are bolded in each column.

	CIFAR-10		Imagenette	
	Exclusion AUC	Inclusion AUC	Exclusion AUC	Inclusion AUC
FastSHAP	**0.42 (0.41, 0.43)**	0.78 (0.77, 0.79)	**0.51 (0.49, 0.52)**	**0.79 (0.78, 0.80)**
KernelSHAP	0.64 (0.63, 0.65)	0.78 (0.77, 0.79)	0.68 (0.67, 0.70)	0.77 (0.75, 0.78)
KernelSHAP-S	0.54 (0.52, 0.55)	**0.86 (0.85, 0.87)**	0.61 (0.60, 0.62)	**0.82 (0.80, 0.83)**
GradCAM	0.52 (0.51, 0.53)	0.76 (0.75, 0.77)	**0.52 (0.50, 0.53)**	0.74 (0.73, 0.76)
Integrated Gradients	0.55 (0.54, 0.56)	0.74 (0.73, 0.75)	0.65 (0.64, 0.67)	0.73 (0.71, 0.74)
SmoothGrad	0.70 (0.69, 0.71)	0.72 (0.71, 0.73)	0.72 (0.71, 0.73)	0.73 (0.72, 0.75)
DeepSHAP	0.65 (0.64, 0.66)	0.79 (0.78, 0.80)	0.69 (0.68, 0.71)	0.74 (0.73, 0.75)
CXPlain	0.56 (0.55, 0.57)	0.71 (0.70, 0.72)	0.60 (0.58, 0.61)	0.72 (0.71, 0.74)

For Imagenette, GradCAM performs competitively with FastSHAP on Exclusion AUC and KernelSHAP-S marginally beats FastSHAP on Inclusion AUC. KernelSHAP-S also outperforms on Inclusion AUC with CIFAR-10, which is perhaps surprising given its high level of noise (fig. 4). However, KernelSHAP-S does not do as well when evaluated using Exclusion AUC, and GradCAM does not do as well on Inclusion AUC. The other methods are, by and large, not competitive on either metric (except for DeepSHAP on CIFAR-10 Inclusion AUC). An accurate explanation should perform well on both metrics, so these results show that FastSHAP provides the most versatile explanations, because it is the only approach to excel at both Inclusion and Exclusion AUC.

6.5 Speed evaluation

All our image experiments were run using 8 cores of an Intel Xeon Gold 6148 processor and a single NVIDIA Tesla V100. For each explanation method, we recorded the clock time required to explain 1,000 images. For FastSHAP, KernelSHAP-S and CXPlain, we also report the time required to train the surrogate and/or explainer models.

Table 2: **Training and explanation run-times for 1,000 images (in minutes).**

		CIFAR-10	Imagenette
Explain	FastSHAP	**0.04**	**0.04**
	KernelSHAP	453.69	1089.50
	KernelSHAP-S	460.10	586.12
	GradCAM	0.38	0.30
	IntGrad	0.91	0.92
	SmoothGrad	1.00	1.05
	DeepSHAP	5.39	6.01
	CXPlain	**0.04**	**0.04**
Train	FastSHAP	693.57	146.49
	KernelSHAP-S	362.03	73.22
	CXPlain	538.49	93.00

Table 2 shows the time (in minutes) required to train each method and explain 1,000 images. The amortized explanation methods, FastSHAP and CXPlain, incur a fixed training cost but very low marginal cost for each explanation. These results suggest that FastSHAP is well suited for real-time applications, where it is crucial to keep explanation times as low as possible. Further, when users need to explain a large quantity of data, such as every example in a dataset, FastSHAP's low explanation cost can quickly compensate for its training time.

7 Discussion

In this work, we introduced FastSHAP, a method for estimating Shapley values in a single forward pass using a learned explainer model. To enable efficient training, we sidestepped the need for a training set and derived a learning approach from the Shapley value's weighted least squares characterization. Our experiments demonstrate that FastSHAP can produce accurate Shapley value estimates while achieving a significant speedup over non-amortized approaches, and we also find that FastSHAP produces high-quality image explanations, outperforming popular gradient-based explanations on quantitative inclusion and exclusion metrics.

While Shapley values provide a powerful theoretical grounding for model explanation, they are often avoided when explaining large-scale models due to their high computational cost. Amortized Shapley value estimation can solve this problem by providing fast and high-quality explanations, as our experiments show. By casting model explanation as a learning problem, FastSHAP stands to benefit as the state of deep learning advances, and we believe that it can be built upon to provide explanations in other settings that require large, high-dimensional models.

8 REPRODUCIBILITY

All the experiments and theoretical results described in the paper were constructed to be reproducible. Code to reproduce any of the results can be found at https://github.com/iclr1814/fastshap, where a random seed was selected for all experiments to ensure reproducibility. In addition to this repository, section 5 contains information about how to reproduce our structured data experiments. Here, we breifly mention the details required to implement each explanation method, including the repositories used, and provide a more extensive description of the model classes and hyperparameters used in appendix D. In section 5.1, we reference each of the open-source datasets used for the experiments, as well as any processing applied to generate the labels used. Similarly in section 6, we provide details about how to reproduce the image experiments. Here, we state each of the explanation methods compared and include information about how each method was implemented along with the open-source package used to do so. We then reference each open-source dataset utilized as well as how the images in each dataset were pre-processed. Further, appendix D contains all additional steps taken to implement each method, including the model architectures employed and the hyperparameters selected.

In addition to the experimental results, this paper provides three theoretical results. We state each of these along with the assumptions required in section 3 and provide complete proofs in appendix A, appendix B, and appendix C.

REFERENCES

Aas, K., Jullum, M., and Løland, A. (2019). Explaining individual predictions when features are dependent: More accurate approximations to Shapley values. *arXiv preprint arXiv:1903.10464*.

Ancona, M., Oztireli, C., and Gross, M. (2019). Explaining deep neural networks with a polynomial time algorithm for Shapley value approximation. In *International Conference on Machine Learning*, pages 272–281. PMLR.

Castro, J., Gómez, D., and Tejada, J. (2009). Polynomial calculation of the Shapley value based on sampling. *Computers & Operations Research*, 36(5):1726–1730.

Charnes, A., Golany, B., Keane, M., and Rousseau, J. (1988). 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.

Chen, J., Song, L., Wainwright, M., and Jordan, M. (2018a). Learning to explain: An information-theoretic perspective on model interpretation. In *International Conference on Machine Learning*, pages 883–892. PMLR.

Chen, J., Song, L., Wainwright, M. J., and Jordan, M. I. (2018b). L-Shapley and C-Shapley: Efficient model interpretation for structured data. *arXiv preprint arXiv:1808.02610*.

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794.

Covert, I. and Lee, S.-I. (2021). Improving KernelSHAP: Practical Shapley value estimation using linear regression. In *International Conference on Artificial Intelligence and Statistics*, pages 3457–3465. PMLR.

Covert, I., Lundberg, S., and Lee, S.-I. (2020a). Explaining by removing: A unified framework for model explanation. *arXiv preprint arXiv:2011.14878*.

Covert, I., Lundberg, S., and Lee, S.-I. (2020b). Understanding global feature contributions with additive importance measures. *Advances in Neural Information Processing Systems*, 33.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314.

Dabkowski, P. and Gal, Y. (2017). Real time image saliency for black box classifiers. In *Advances in Neural Information Processing Systems*, pages 6967–6976.

Datta, A., Sen, S., and Zick, Y. (2016). Algorithmic Transparency via Quantitative Input Influence: Theory and Experiments with Learning Systems. In *Proceedings - 2016 IEEE Symposium on Security and Privacy, SP 2016*, pages 598–617. Institute of Electrical and Electronics Engineers Inc.

Dua, D. and Graff, C. (2017). UCI machine learning repository.

Fernandes, K., Vinagre, P., and Cortez, P. (2015). A proactive intelligent decision support system for predicting the popularity of online news. In *Portuguese Conference on Artificial Intelligence*, pages 535–546. Springer.

Frye, C., de Mijolla, D., Begley, T., Cowton, L., Stanley, M., and Feige, I. (2020). Shapley explainability on the data manifold. *arXiv preprint arXiv:2006.01272*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

Hooker, S., Erhan, D., Kindermans, P.-J., and Kim, B. (2018). A benchmark for interpretability methods in deep neural networks. *arXiv preprint arXiv:1806.10758*.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257.

Howard, J. and Gugger, S. (2020). Fastai: A layered API for deep learning. *Information*, 11(2):108.

Illés, F. and Kerényi, P. (2019). Estimation of the Shapley value by ergodic sampling. *arXiv preprint arXiv:1906.05224*.

Janzing, D., Minorics, L., and Blöbaum, P. (2020). Feature relevance quantification in explainable AI: A causal problem. In *International Conference on Artificial Intelligence and Statistics*, pages 2907–2916. PMLR.

Jethani, N., Sudarshan, M., Aphinyanaphongs, Y., and Ranganath, R. (2021). 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.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, 30:3146–3154.

Kohavi, R. et al. (1996). Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *Knowledge Discovery and Data Mining*, volume 96, pages 202–207.

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

Liang, D., Lu, C.-C., Tsai, C.-F., and Shih, G.-A. (2016). Financial ratios and corporate governance indicators in bankruptcy prediction: A comprehensive study. *European Journal of Operational Research*, 252(2):561–572.

Lipovetsky, S. and Conklin, M. (2001). Analysis of regression in game theory approach. *Applied Stochastic Models in Business and Industry*, 17(4):319–330.

Lundberg, S. M., Erion, G., Chen, H., DeGrave, A., Prutkin, J. M., Nair, B., Katz, R., Himmelfarb, J., Bansal, N., and Lee, S.-I. (2020). From local explanations to global understanding with explainable AI for trees. *Nature Machine Intelligence*, 2(1):56–67.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, 30:4765–4774.

Mitchell, R., Cooper, J., Frank, E., and Holmes, G. (2021). Sampling permutations for Shapley value estimation. *arXiv preprint arXiv:2104.12199*.

Moro, S., Cortez, P., and Rita, P. (2014). A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31.

Petsiuk, V., Das, A., and Saenko, K. (2018). RISE: Randomized input sampling for explanation of black-box models. *arXiv preprint arXiv:1806.07421*.

Ruiz, L. M., Valenciano, F., and Zarzuelo, J. M. (1998). The family of least square values for transferable utility games. *Games and Economic Behavior*, 24(1-2):109–130.

Schulz, K., Sixt, L., Tombari, F., and Landgraf, T. (2020). Restricting the flow: Information bottlenecks for attribution. *arXiv preprint arXiv:2001.00396*.

Schwab, P. and Karlen, W. (2019). CXPlain: Causal explanations for model interpretation under uncertainty. In *Advances in Neural Information Processing Systems*, pages 10220–10230.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 618–626.

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

Shrikumar, A., Greenside, P., and Kundaje, A. (2017). Learning important features through propagating activation differences. In *International Conference on Machine Learning*, pages 3145–3153. PMLR.

Smilkov, D., Thorat, N., Kim, B., Viégas, F., and Wattenberg, M. (2017). Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*.

Štrumbelj, E. and Kononenko, I. (2014). Explaining prediction models and individual predictions with feature contributions. *Knowledge and Information Systems*, 41(3):647–665.

Sundararajan, M., Taly, A., and Yan, Q. (2017). Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pages 3319–3328. PMLR.

Van den Broeck, G., Lykov, A., Schleich, M., and Suciu, D. (2021). On the tractability of SHAP explanations. In *Proceedings of the 35th Conference on Artificial Intelligence (AAAI)*.

Wang, R., Wang, X., and Inouye, D. I. (2021). Shapley explanation networks. In *International Conference on Learning Representations*.

Yoon, J., Jordon, J., and van der Schaar, M. (2018). INVASE: Instance-wise variable selection using neural networks. In *International Conference on Learning Representations*.

A GLOBAL OPTIMIZER

Here, we prove that FastSHAP is based on a training objective with a global optimizer that outputs the true Shapley values. Recall that the loss function for $\phi_{\text{fast}}(\mathbf{x}, \mathbf{y}; \theta)$ is

$$\mathcal{L}(\theta) = \mathop{\mathbb{E}}_{p(\mathbf{x})} \mathop{\mathbb{E}}_{\text{Unif}(\mathbf{y})} \mathop{\mathbb{E}}_{p(\mathbf{s})} \Big[\Big(v_{\mathbf{x},\mathbf{y}}(\mathbf{s}) - v_{\mathbf{x},\mathbf{y}}(\mathbf{0}) - \mathbf{s}^\top \phi_{\text{fast}}(\mathbf{x}, \mathbf{y}; \theta) \Big)^2 \Big].$$

As mentioned in the main text, it is necessary to force the model to satisfy the Efficiency constraint, or the property that the predictions from $\phi_{\text{fast}}(\mathbf{x}, \mathbf{y}; \theta)$ satisfy

$$\mathbf{1}^\top \phi_{\text{fast}}(x, y; \theta) = v_{x,y}(\mathbf{1}) - v_{x,y}(\mathbf{0}) \quad \forall\, x \in \mathcal{X}, y \in \mathcal{Y}.$$

One option for guaranteeing the efficiency property is to adjust the model outputs using their additive efficient normalization (see section 3). Incorporating this constraint on the predictions, we can then view the loss function as an expectation across $(\mathbf{x}, \mathbf{y})$ and write the expected loss for each sample (x, y) as a separate optimization problem over the variable $\phi_{x,y} = \phi_{\text{fast}}(x, y; \theta) \in \mathbb{R}^d$:

$$\begin{aligned} \min_{\phi_{x,y}} \; & \mathop{\mathbb{E}}_{p(\mathbf{s})} \Big[\Big(v_{x,y}(\mathbf{s}) - v_{x,y}(\mathbf{0}) - \mathbf{s}^\top \phi_{x,y} \Big)^2 \Big] \\ \text{s.t.} \quad & \phi_{x,y} = v_{x,y}(\mathbf{1}) - v_{x,y}(\mathbf{0}). \end{aligned} \tag{9}$$

This is a constrained weighted least squares problem with a unique global minimizer, and it is precisely the Shapley value's weighted least squares characterization. We can therefore conclude that the optimal prediction for each pair (x, y) is the true Shapley values, or $\phi^*_{x,y} = \phi(v_{x,y})$, and that the global optimizer for our objective function is a model $\phi_{\text{fast}}(\mathbf{x}, \mathbf{y}; \theta^*)$ that outputs the true Shapley values.

If we incorporate efficiency regularization with parameter $\gamma > 0$ rather than a constraint on the model's predictions, then $\phi_{\text{fast}}(\mathbf{x}, \mathbf{y}; \theta)$ is trained using a modified objective. The expected loss for each pair (x, y) yields the following optimization problem over the variable $\phi_{x,y} \in \mathbb{R}^d$:

$$\min_{\phi_{x,y}} \mathop{\mathbb{E}}_{p(\mathbf{s})} \Big[\Big(v_{x,y}(\mathbf{s}) - v_{x,y}(\mathbf{0}) - \mathbf{s}^\top \phi_{x,y} \Big)^2 \Big] + \gamma \Big(v_{x,y}(\mathbf{1}) - v_{x,y}(\mathbf{0}) - \mathbf{1}^\top \phi_{x,y} \Big)^2.$$

For finite hyperparameter values $\gamma \in [0, \infty)$, the training objective relaxes the Shapley value's efficiency property and eliminates the requirement that predictions sum to the grand coalition's value. However, using $\gamma = \infty$ turns the penalty into a constraint, resulting in the same constrained problem from eq. (9) for each sample (x, y). As a result, we can conclude that setting $\gamma = \infty$ provides another technique to ensure that FastSHAP's global optimizer is a function that outputs the true Shapley values, at least in theory; in practice, excessively large γ values can disrupt gradient-based optimization, and finite values can be expected to yield sufficiently accurate results.

To achieve the global optimum, we require the ability to sample from $p(\mathbf{x})$ (or a sufficiently large dataset), perfect optimization, and a function class for $\phi_{\text{fast}}(\mathbf{x}, \mathbf{y}; \theta)$ that is expressive enough to contain the global optimizer. We describe below why neural networks are theoretically justified as a choice of function class.

The universal approximation theorem for neural networks (Cybenko, 1989; Hornik, 1991) guarantees that infinite-width neural networks can represent continuous functions to an arbitrary level of accuracy. Note that if the value function $v_{x,y}(s)$ in FastSHAP for a fixed subset s and label y is understood as a function of the input x, or $g_{s,y}(x) = v_{x,y}(s)$, and the function $g_{s,y}(x)$ is continuous for each $s \in \{0, 1\}^d$ and $y \in \mathcal{Y}$, then the Shapley value function $h_y(x) = \phi(v_{x,y})$ is guaranteed to be continuous as well for all y. This is the case in our experiments, where the value function is a surrogate model parameterized by a continuous neural network. Under the mild assumption that $h_y(x) = \phi(v_{x,y})$ is continuous, we can invoke the universal approximation theorem to claim that an infinite-width neural network can approximate the true Shapley values to an arbitrary degree of accuracy.

B ADDITIVE EFFICIENT NORMALIZATION

Here, we provide a geometric interpretation for the additive efficient normalization step and prove that it is guaranteed to yield Shapley value estimates that are closer to their true values. Consider a game v with Shapley values $\phi(v) \in \mathbb{R}^d$, and assume that we have Shapley values estimates $\hat{\phi} \in \mathbb{R}^d$ that do not satisfy the efficiency property. To achieve this property, we can project these estimates onto the *efficient hyperplane*, or the subset of $\mathbb{R}^d$ where the efficiency property is satisfied. This corresponds to solving the following optimization problem over $\phi_{\text{eff}} \in \mathbb{R}^d$:

$$\min_{\phi_{\text{eff}}} ||\phi_{\text{eff}} - \hat{\phi}||^2 \quad \text{s.t.} \quad \mathbf{1}^\top \phi_{\text{eff}} = v(\mathbf{1}) - v(\mathbf{0}).$$

We can solve the problem via its Lagrangian, denoted by $\mathcal{L}(\phi_{\text{eff}}, \nu)$, with the Lagrange multiplier $\nu \in \mathbb{R}$ as follows:

$$\begin{aligned}\mathcal{L}(\phi_{\text{eff}}, \nu) &= ||\phi_{\text{eff}} - \hat{\phi}||^2 + \nu\Big(v(\mathbf{1}) - v(\mathbf{0}) - \mathbf{1}^\top \phi_{\text{eff}}\Big) \\ \Rightarrow \phi^*_{\text{eff}} &= \hat{\phi} - \mathbf{1}\frac{v(\mathbf{1}) - v(\mathbf{0}) - \mathbf{1}^\top \hat{\phi}}{d}.\end{aligned}$$

This transformation, where the efficiency gap is split evenly and added to each estimate, is known as *additive efficient normalization* (Ruiz et al., 1998). We implement it as an output layer for FastSHAP's predictions to ensure that they satisfy the efficiency property (section 3). This step can therefore be understood as a projection of the network's output onto the efficient hyperplane.

The normalization step is guaranteed to produce corrected estimates ϕ^*_{eff} that are closer to the true Shapley values $\phi(v)$ than the original estimates $\hat{\phi}$. To see this, note that the projection step guarantees that $\hat{\phi} - \phi^*_{\text{eff}}$ and $\phi^*_{\text{eff}} - \phi(v)$ are orthogonal vectors, so the Pythagorean theorem yields the inequality

$$\begin{aligned}||\phi(v) - \hat{\phi}||^2 &= ||\phi(v) - \phi^*_{\text{eff}}||^2 + ||\phi^*_{\text{eff}} - \hat{\phi}||^2 \\ &\geq ||\phi(v) - \phi^*_{\text{eff}}||^2.\end{aligned}$$

C REDUCING GRADIENT VARIANCE

Recall our objective function $\mathcal{L}(\theta)$, which is defined as

$$\mathcal{L}(\theta) = \mathop{\mathbb{E}}_{p(\mathbf{x})} \mathop{\mathbb{E}}_{\text{Unif}(\mathbf{y})} \mathop{\mathbb{E}}_{p(\mathbf{s})} \Big[\big(v_{\mathbf{x},\mathbf{y}}(\mathbf{s}) - v_{x,y}(\mathbf{0}) - \mathbf{s}^\top \phi_{\text{fast}}(\mathbf{x}, \mathbf{y}; \theta)\big)^2\Big].$$

The objective's gradient is given by

$$\nabla_\theta \mathcal{L}(\theta) = \mathop{\mathbb{E}}_{p(\mathbf{x})} \mathop{\mathbb{E}}_{\text{Unif}(\mathbf{y})} \mathop{\mathbb{E}}_{p(\mathbf{s})} \Big[\nabla(\mathbf{x}, \mathbf{y}, \mathbf{s}; \theta)\Big], \tag{10}$$

where we define

$$\nabla(x, y, s; \theta) := \nabla_\theta \big(v_{x,y}(s) - v_{x,y}(\mathbf{0}) - s^\top \phi_{\text{fast}}(x, y; \theta)\big)^2.$$

When FastSHAP is trained with a single set of samples (x, y, s), the gradient covariance is given by $\text{Cov}\big(\nabla(\mathbf{x}, \mathbf{y}, \mathbf{s}; \theta)\big)$, which may be too large for effective optimization. We use several strategies to reduce gradient variance. First, given a model that outputs estimates for all classes $y \in \{1, \ldots, K\}$, we calculate the loss jointly for all classes. This yields gradients that we denote as $\nabla(\mathbf{x}, \mathbf{s}; \theta)$, defined as

$$\nabla(x, s; \theta) := \mathop{\mathbb{E}}_{\text{Unif}(\mathbf{y})}[\nabla(x, \mathbf{y}, s; \theta)],$$

where we have the relationship

$$\text{Cov}\big(\nabla(\mathbf{x}, \mathbf{s}; \theta)\big) \preceq \text{Cov}\big(\nabla(\mathbf{x}, \mathbf{y}, \mathbf{s}; \theta)\big)$$

due to the law of total covariance. Next, we consider minibatches of b independent x samples, which yields gradients $\nabla_b(\mathbf{x}, \mathbf{s}; \theta)$ with covariance given by

$$\text{Cov}\big(\nabla_b(\mathbf{x}, \mathbf{s}; \theta)\big) = \frac{1}{b}\text{Cov}\big(\nabla(\mathbf{x}, \mathbf{s}; \theta)\big).$$

We then consider sampling m independent coalitions s for each input x, resulting in the gradients $\nabla_b^m(\mathbf{x}, \mathbf{s}; \theta)$ with covariance given by

$$\text{Cov}\big(\nabla_b^m(\mathbf{x}, \mathbf{s}; \theta)\big) = \frac{1}{mb}\text{Cov}\big(\nabla(\mathbf{x}, \mathbf{s}; \theta)\big).$$

Finally, we consider a *paired sampling* approach, where each sample $s \sim p(\mathbf{s})$ is paired with its complement $\mathbf{1} - s$. Paired sampling has been shown to reduce KernelSHAP's variance (Covert and Lee, 2021), and our experiments show that it helps FastSHAP achieve better Shapley value estimates (appendix D).

The training algorithm in the main text is simplified by omitting these gradient variance reduction techniques, so we also provide algorithm 2 below, which includes minibatching, multiple coalition samples, paired sampling, efficiency regularization and parallelization over all output classes.

Algorithm 2: Full FastSHAP training

Input: Value function $v_{x,y}$, learning rate α, batch size b, samples m, penalty parameter γ
Output: FastSHAP explainer $\phi_{\text{fast}}(\mathbf{x}, \mathbf{y}; \theta)$
initialize $\phi_{\text{fast}}(\mathbf{x}, \mathbf{y}; \theta)$
while *not converged* **do**
  set $\mathcal{R} \leftarrow 0$, $\mathcal{L} \leftarrow 0$
  for $i = 1, \ldots, b$ **do**
    sample $x \sim p(\mathbf{x})$
    for $y = 1, \ldots, K$ **do**
      predict $\hat{\phi} \leftarrow \phi_{\text{fast}}(x, y; \theta)$
      calculate $\mathcal{R} \leftarrow \mathcal{R} + \left(v_{x,y}(\mathbf{1}) - v_{x,y}(\mathbf{0}) - \mathbf{1}^\top \hat{\phi}\right)^2$ `// Pre-normalization`
      if *normalize* **then**
        set $\hat{\phi} \leftarrow \hat{\phi} + d^{-1}\left(v_{x,y}(\mathbf{1}) - v_{x,y}(\mathbf{0}) - \mathbf{1}^\top \hat{\phi}\right)$
      end
      for $j = 1, \ldots, m$ **do**
        if *paired sampling* **and** $i \bmod 2 = 0$ **then**
          set $s \leftarrow \mathbf{1} - s$ `// Invert previous subset`
        else
          sample $s \sim p(\mathbf{s})$
        end
        calculate $\mathcal{L} \leftarrow \mathcal{L} + \left(v_{x,y}(s) - v_{x,y}(\mathbf{0}) - s^\top \hat{\phi}\right)^2$
      end
    end
  end
  update $\theta \leftarrow \theta - \alpha \nabla_\theta \left(\frac{\mathcal{L}}{bmK} + \gamma \frac{\mathcal{R}}{bK}\right)$
end

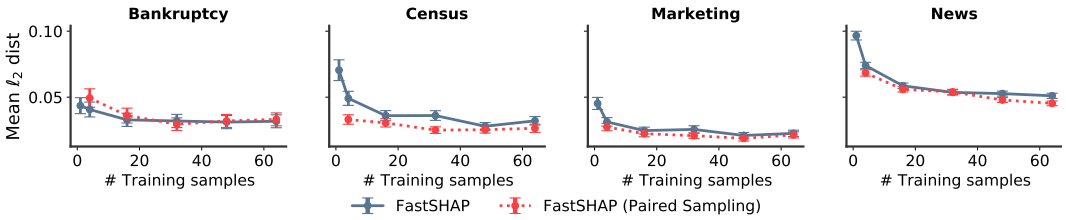


Figure 6: **FastSHAP accuracy as a function of the number of training samples.** The results show that using more $\mathbf{s}$ samples per $\mathbf{x}$ improves FastSHAP's closeness to the ground truth Shapley values, as does the use of paired sampling.

D FastSHAP models and hyperparameters

In this section, we describe the models and architectures used for each dataset as well as the hyperparameters used when training FastSHAP.

D.1 Models

Tabular datasets. For the original model $f(\mathbf{x}; \eta)$, we use neural networks for the `news` and `marketing` datasets and gradient boosted trees for the `census` (LightGBM (Ke et al., 2017)) and `bankruptcy` (XGBoost (Chen and Guestrin, 2016)) datasets. The FastSHAP model $\phi_{\text{fast}}(\mathbf{x}, \mathbf{y}; \theta)$ and the surrogate model $p_{\text{surr}}(\mathbf{y} \mid m(\mathbf{x}, \mathbf{s}); \beta)$ are implemented using neural networks that consist of 2-3 fully connected layers with 128 units and ReLU activations. The p_{surr} models use a softmax output layer, while ϕ_{fast} has no output activation. The models are trained using Adam with a learning rate of 10^{-3}, and we use a learning rate scheduler that multiplies the learning rate by a factor of 0.5 after 3 epochs of no validation loss improvement. Early stopping was triggered after the validation loss ceased to improve for 10 epochs.

Image datasets. The models $f(\mathbf{x}; \eta)$ and p_{surr} are ResNet-50 models pretrained on Imagenet. We use these without modification to the architecture and fine-tune them on each image dataset. To create the $\phi_{\text{fast}}(\mathbf{x}, \mathbf{y}; \theta)$ model, we modify the architecture to return a tensor of size $14 \times 14 \times K$. First, the layers after the 4th convolutional block are removed; the output of this block is $14 \times 14 \times 256$. We then append a 2D convolutional layer with K filters, each of size 1×1, so that the output is $14 \times 14 \times K$ and the yth 14×14 slice corresponds to the superpixel-level Shapley values for each class $y \in \mathcal{Y}$. Each model is trained using Adam with a learning rate of 10^{-3}, and we use a learning rate scheduler that multiplies the learning rate by a factor of 0.8 after 3 epochs of no validation loss improvement. Early stopping was triggered after the validation loss ceased to improve for 20 epochs.

D.2 FastSHAP hyperparameters

We now explore various settings of FastSHAP's hyperparameters and observe their impact on FastSHAP's performance. There are two types of hyperparameters: sampling hyperparameters, which affect the number of samples of $\mathbf{s}$ taken during training, and efficiency hyperparameters, which control how we enforce the Efficiency constraint. Sampling hyperparameters include: (1) whether to use paired sampling, and (2) the number of samples of $\mathbf{s}$ per $\mathbf{x}$ to take during training. Efficiency hyperparameters include: (1) the choice of γ in eq. (4), and (2) whether to perform the additive efficient normalization during training, inference or both.

To understand the effect of sampling hyperparameters, we perform experiments using the same tabular datasets from the main text. We use the in-distribution value function p_{surr} and compute the ground truth SHAP values the same way as in our previous experiments (i.e., by running KernelSHAP to convergence).

Figure 6 shows the mean ℓ_2 distance between FastSHAP's estimates and the ground truth. We find that across all four datasets, increasing the number of training samples of $\mathbf{s}$ generally improves the mean ℓ_2 distance to ground truth. We also find that for any fixed number of samples (greater than 1), using paired sampling improves FastSHAP's accuracy.

Table 3 shows the results of an ablation study for the efficiency hyperparameters. *Normalization* (or *Norm.*) refers to the additive efficient normalization step (applied during training and inference,

Table 3: **FastSHAP ablation results.** The distance to the ground truth Shapley values is displayed for several FastSHAP variations, showing that normalization helps and that the penalty is unnecessary.

	Census		Bankruptcy	
	ℓ_2	ℓ_1	ℓ_2	ℓ_1
Normalization	**0.0229**	**0.0863**	**0.0295**	**0.2436**
Normalization + Penalty	0.0261	0.0971	0.0320	0.2740
Inference Norm.	0.0406	0.1512	0.0407	0.3450
Inference Norm. + Penalty	0.0452	0.1671	0.0473	0.4471
No Norm.	0.0501	0.1933	0.0408	0.3474
No Norm. + Penalty	0.0513	0.1926	0.0474	0.4490

or only during inference), and *penalty* refers to the efficiency regularization technique with the parameter set to $\gamma = 0.1$. We find that using normalization during training uniformly achieves better results than without normalization or with normalization only during inference. The efficiency regularization approach proves to be less effective, generally leading to less accurate Shapley value estimates. Based on these results, we opt to use additive efficient normalization in our tabular data experiments.

E Additional results for image experiments

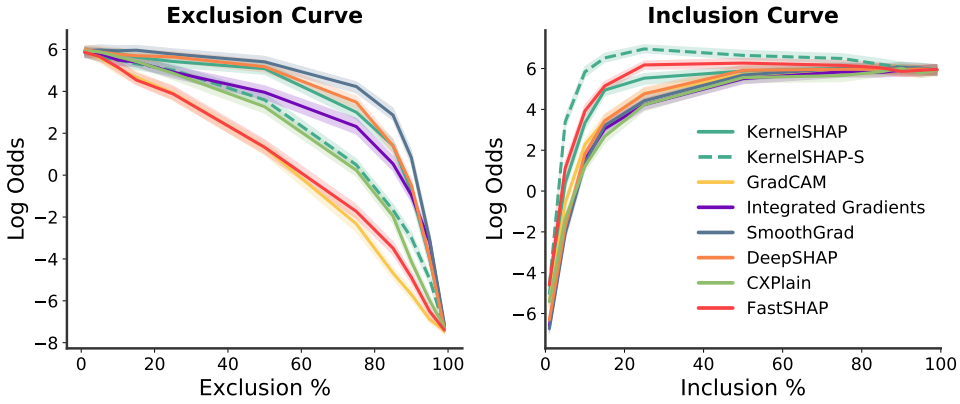
In this section, we provide additional results for the FastSHAP image experiments.

E.1 Inclusion and exclusion metrics

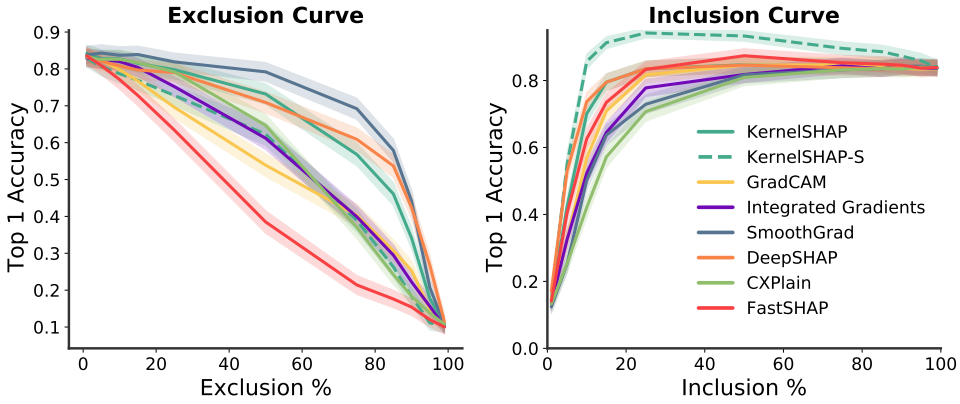
Table 4 shows our inclusion and exclusion metrics when replicated using log-odds rather than accuracy. Similar to our metrics described in the main text, we choose the class predicted by the original model for each image, and we measure the average log-odds for that class as we include or exclude important features according to the explanations generated by each method. The results confirm roughly the same ordering between methods, with FastSHAP being the only method to achieve strong results on both metrics for both datasets. Figure 7 shows the raw inclusion and exclusion curves for both the accuracy and log-odds-derived metrics.

Table 4: **Exclusion and Inclusion AUCs calculated using the average log-odds of the predicted class.**

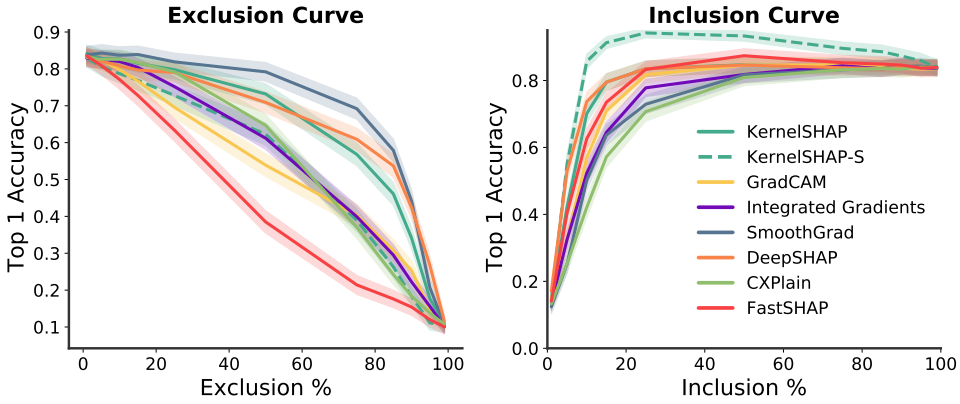
	CIFAR-10		Imagenette	
	Exclusion AUC	Inclusion AUC	Exclusion AUC	Inclusion AUC
FastSHAP	**5.92 (5.62, 6.14)**	5.36 (5.16, 5.63)	**7.98 (7.68, 8.33)**	5.40 (5.16, 5.60)
KernelSHAP	9.88 (9.55, 10.20)	5.36 (5.14, 5.63)	10.68 (10.36, 11.00)	5.07 (4.81, 5.31)
KernelSHAP-S	8.01 (7.68, 8.34)	**6.80 (6.65, 6.96)**	9.39 (9.11, 9.66)	**6.01 (5.78, 6.26)**
GradCAM	7.75 (7.44, 8.09)	4.99 (4.81, 5.26)	**7.77 (7.49, 8.05)**	4.65 (4.40, 4.89)
Integrated Gradients	8.34 (8.03, 8.61)	4.58 (4.37, 4.85)	10.14 (9.79, 10.46)	4.34 (4.10, 4.58)
SmoothGrad	10.99 (10.67, 11.29)	4.30 (4.08, 4.58)	11.19 (10.84, 11.48)	4.47 (4.24, 4.70)
DeepSHAP	9.96 (9.61, 10.24)	5.47 (5.28, 5.76)	10.93 (10.61, 11.20)	4.63 (4.38, 4.85)
CXPlain	8.34 (8.00, 8.58)	4.02 (3.80, 4.31)	9.13 (8.83, 9.41)	4.33 (4.11, 4.57)



(a) **Imagenette: Log-odds derived inclusion and exclusion curves.**



(b) **CIFAR-10: Accuracy derived inclusion and exclusion curves.**



(c) **CIFAR-10: Log-odds derived inclusion and exclusion curves.**

Figure 7: **Additional inclusion and exclusion curves.** The change in top-1 accuracy or average log-odds of the predicted class as an increasing percentage of the pixels estimated to be important are excluded (left) or included (right) from the set of 1,000 images.

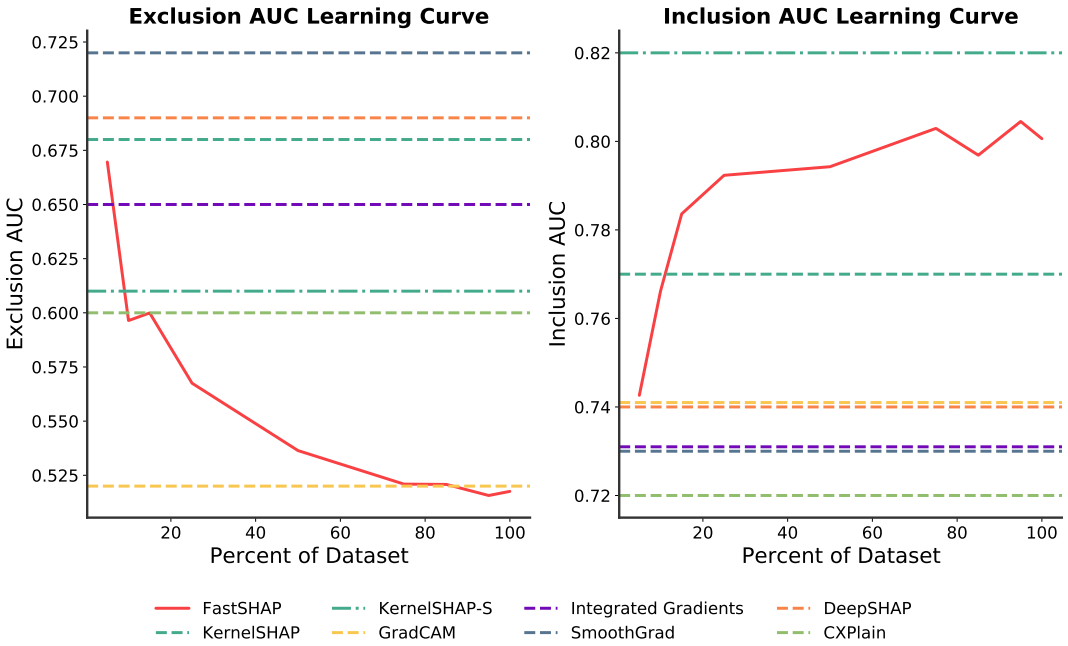


Figure 8: **FastSHAP robustness to limited data.** The curves are generated by training FastSHAP with varying portions of the Imagenette dataset and evaluating the Inclusion and Exclusion AUC. Horizontal lines show the Exclusion and Inclusion AUCs for each of the baseline methods, as reported in table 1.

E.2 FastSHAP robustness to limited data

Figure 8 plots the change in inclusion and exclusion AUC, calculated using top-1 accuracy, achieved when training FastSHAP with limited data. FastSHAP was trained with 95%, 85%, 75%, 50%, 25%, 15%, 10%, and 5% of the Imagenette training dataset. For the Imagenette dataset, we find that FastSHAP remains competitive when using just 10% of the data, and that it outperforms most baseline methods by a large margin when using just 25%.

E.3 EXAMPLE FASTSHAP IMAGE EXPLANATIONS

Finally, we show additional explanations generated by FastSHAP and the baseline methods for both CIFAR-10 and Imagenette.

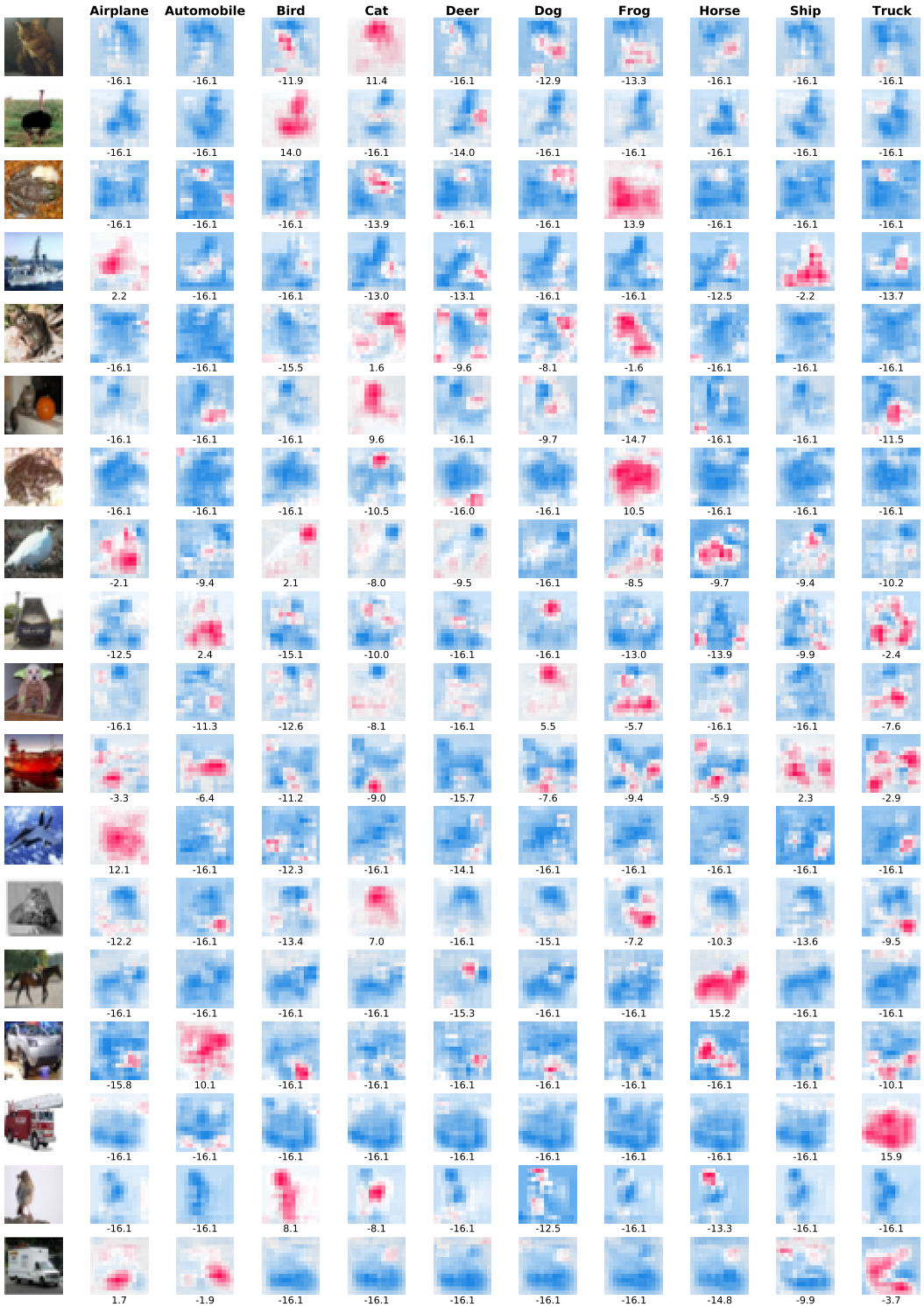


Figure 9: **Explanations generated by FastSHAP for 18 randomly selected CIFAR-10 images.** Each column corresponds to a CIFAR-10 class, and the model's prediction (in logits) is provided below each image.

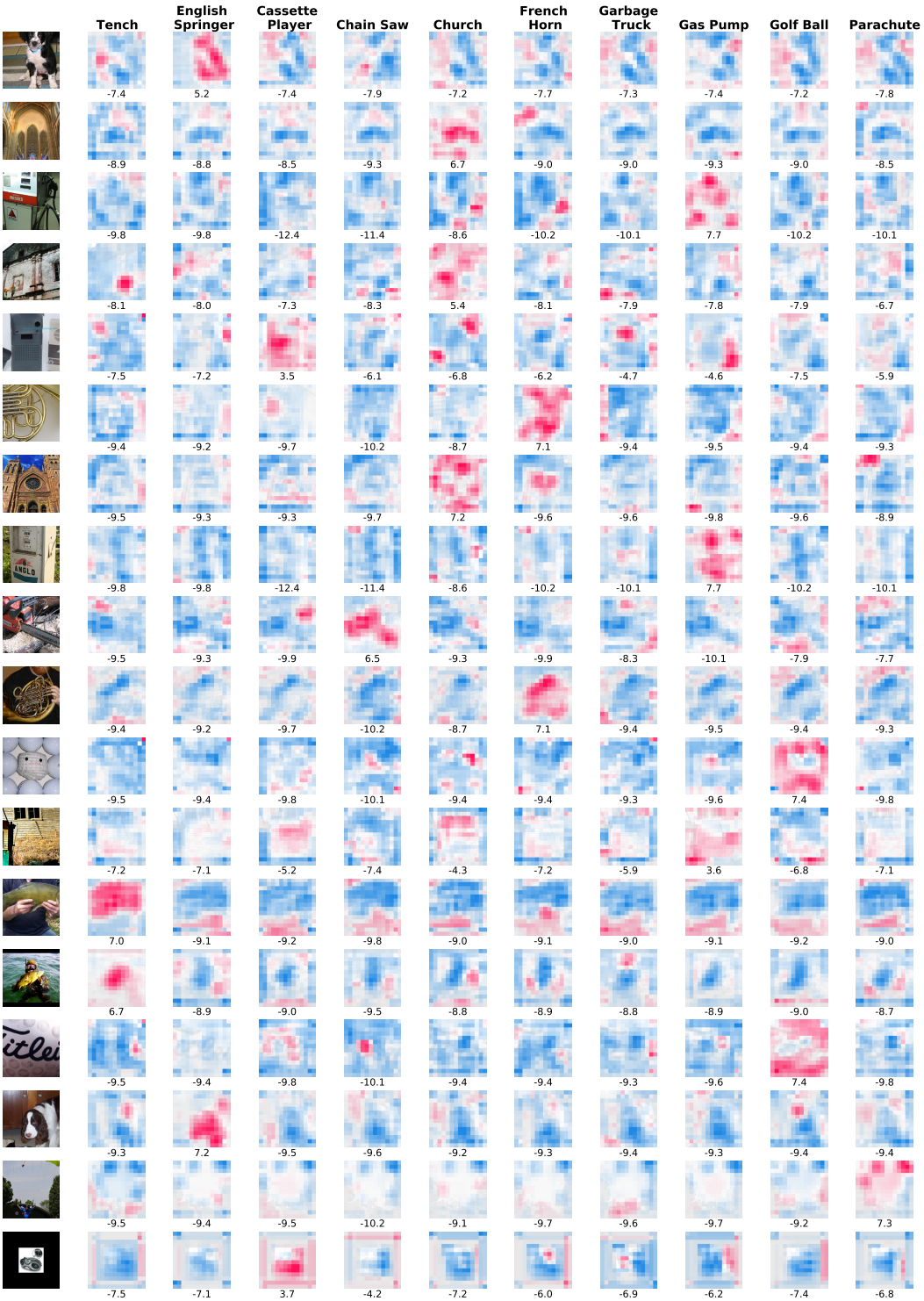


Figure 10: **Explanations generated by FastSHAP for 18 randomly selected Imagenette images.** Each column corresponds to an Imagenette class, and the model's prediction (in logits) is provided below each image.

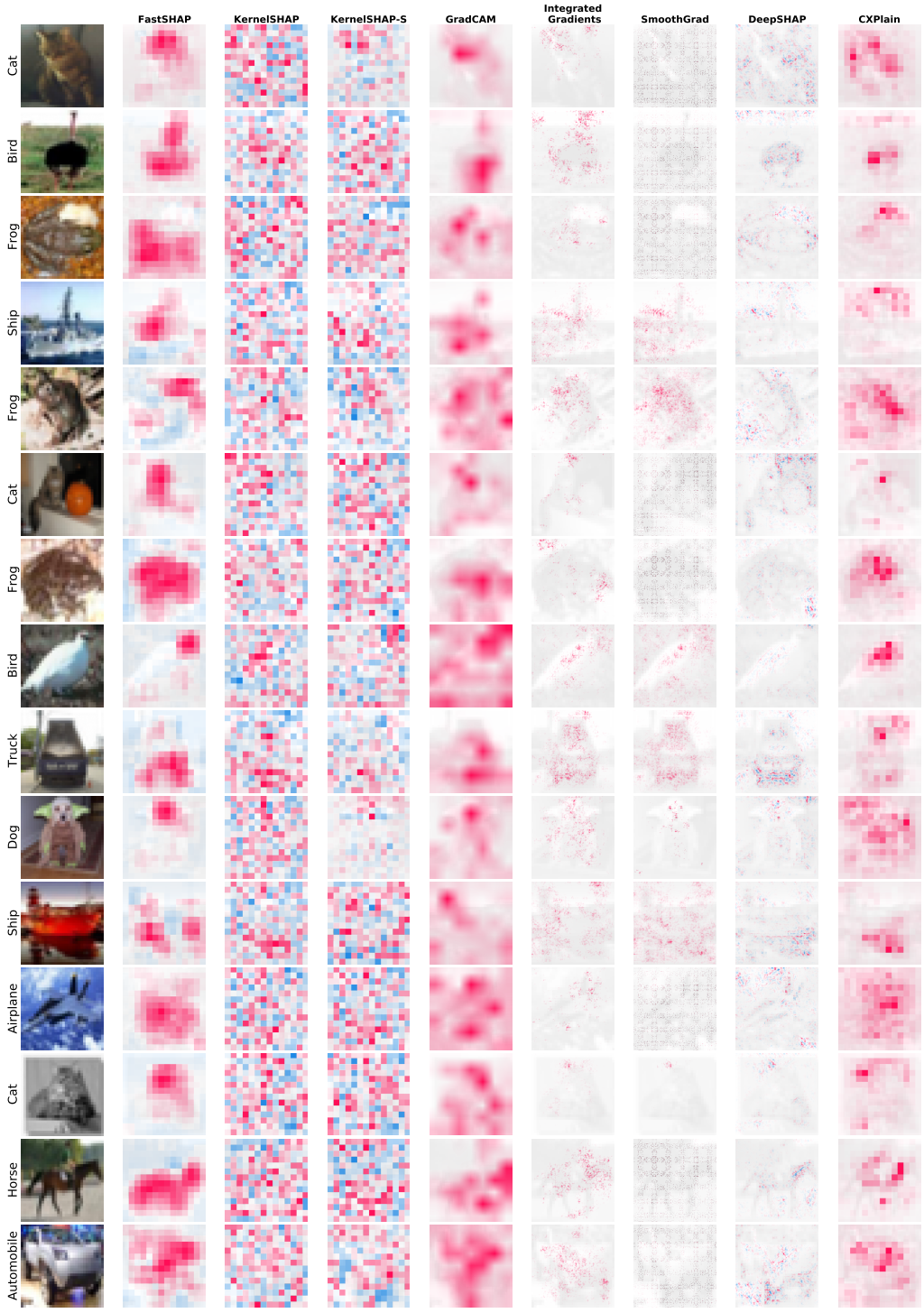


Figure 11: **Explanations generated for the predicted class for 15 randomly selected CIFAR-10 images.** Each column corresponds to an explanation method, and each row is labeled with the image's corresponding class.

Figure 12: **Explanations generated for the predicted class for 15 randomly selected Imagenette images.** Each column corresponds to an explanation method, and each row is labeled with the image's corresponding class.