LightningSHAP: A Cost-Effective Approach to Local Shapley Values Estimation

LightningSHAP: A Cost-Effective Approach to Local Shapley Values Estimation

Davide Napolitano, Luca Cagliero, *Member, IEEE*

***Abstract*—Shapley Values (SVs) are concepts established for explaining black-box machine learning models by quantifying feature contributions to model predictions. Since their computation has exponential complexity in the number of features, a variety of approximated approaches to SVs estimation have been proposed. The state-of-the-art neural SVs estimator, i.e., FastSHAP, advances sampling-based methods by first training a supervised surrogate model to learn the conditional expectation of the original model given every feature subset and then generating SVs estimates through a separate network using weighted least squares. While ensuring fast SVs inference, this approach requires a significant training time, thus becoming unsuitable for scenarios where the black-box model has to be frequently retrained. To address this limitation, we propose LIGHTNINGSHAP, a cost-effective neural network-based estimator that jointly computes conditional expectations and SVs estimations to reduce training cost. The unified network learning process minimizes feature-level estimation errors while preserving SVs efficiency. Experiments run on both dynamic and static tabular datasets show that LIGHTNINGSHAP achieves a 25%-60% speedup in overall computational time (i.e., the sum of training and inference times) on medium-large datasets compared to existing neural SVs estimators as well as lower or comparable estimation errors. Furthermore, the results obtained on image datasets indicate that LIGHTNINGSHAP yields a 30%-55% speedup while preserving explanation quality.**

***Impact Statement*—The diffusion of machine learning and deep learning solutions across different domains and scenarios has increased the need for model transparency. Feature attribution methods have become essential to explain model decisions by quantifying the extent to which each input feature influences model predictions. Methods based on Shapley Values (SV) have gained prominence due to their strong theoretical foundations. Once trained, neural SVs explainers can be efficiently applied to samples of large-scale datasets thanks to their fast inference time. Hence, they can be successfully used to generate real-time explanations. However, in real-world application scenarios, such as intrusion detection or financial market forecasting, the training cost becomes compelling as black-box models need to be frequently retrained. Within these scenarios, existing neural approaches are challenged by the high cost of network learning and SVs estimation. Making neural SVs estimators also efficient in terms of training time extends their portability to a wider range of domains and scenarios, facilitating their broader adoption in AI-based applications.**

***Index Terms*—Explainable AI, Shapley Value, Model-Agnostic, Shapley Value Estimation**

D. Napolitano and L. Cagliero are with the Dipartimento di Automatica e Informatica, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129, Torino, Italy. E-mail: {davide.napolitano,luca.cagliero}@polito.it

This study was carried out within the FAIR (Future Artificial Intelligence Research) and received funding from Next-GenerationEU (Italian PNRR – M4 C2, Invest 1.3 – D.D. 1555.11-10-2022, PE00000013). This manuscript reflects only the authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them.

I. INTRODUCTION

SHAPLEY Values (SVs) are concepts established in cooperative game theory [1], which have become popular for explaining machine learning models [2]–[5]. Given a black-box predictor, SVs are used in local feature attribution methods to generate post-hoc explanations of the predictor estimates. Beyond estimating the feature importance scores locally to each testing sample, SVs can also be used to explain the global model behavior by computing Global SVs either directly or by averaging the local contributions across all samples [2].

Calculating exact Shapley Values is challenging from a computational perspective, as their algorithmic complexity is exponential with the number of input features [6]. To efficiently compute approximated SVs estimates on top of machine learning model predictions, prior works propose both model-specific [2], [7] and model-agnostic approaches [3], [5].

Thanks to the better portability, model-agnostic strategies for computing Shapley Values have garnered increasing attention [6]. They encompass both classical sampling techniques [8] and more sophisticated approximation strategies, including kernel-based methods and neural estimators [2], [7]. Among the latter approaches, neural estimators have shown to achieve the best trade-off between run-time efficiency and approximation accuracy. Specifically, the state-of-the-art neural approach to SVs estimation [3] follows a two-step process: first, it estimates the coalition game by learning the conditional expectation of the black-box model output using a surrogate model. This surrogate model is trained to estimate the black-box predictions for arbitrary feature subsets by minimizing the Kullback-Leibler (KL) divergence [9] between its outputs and the original model's predictions. Secondly, it trains an explainer network to compute the approximated SVs by leveraging the trained surrogate model's estimates. Network training is driven by a weighted least squares objective function optimized through stochastic gradient descent. Designed specifically for real-time SVs estimation and large-scale datasets, FastSHAP excels in producing fast explanations. However, the sequential learning of surrogate and explainer models makes the initial training phase particularly computationally demanding, especially on complex datasets [10]. This limits its portability to scenarios where black-box models have to be frequently retrained [11].

Consider the network traffic traces and financial market time series as representative examples of real-world data. They show rapidly evolving trends that are worth considering for network intrusion detection [12] and market forecasting [13], respectively. Since black-box models tailored for intrusion

detection and financial market forecasting must be frequently retrained, explainers applied on top of them require not only a short inference time but also an efficient training process.

To overcome the training inefficiencies of surrogate model-based neural SVs estimators, this paper proposes LIGHTNINGSHAP, a cost-effective approach to local neural SVs estimation. Instead of adopting a cascade of surrogate model and explainer (see the traditional pipeline in the upper part of Figure 1), LIGHTNINGSHAP on-the-fly computes SVs estimations while approximating the conditional expectation of black-box models (see lower pipeline on Figure 1).

LIGHTNINGSHAP learns the black-box parametric learning function f through a unified neural network model that jointly pursues the following objectives:

- Learning the conditional expectation for every feature subset S on an input sample x given the black-box model $f(x;\eta)$, via KL-divergence minimization (loss term L_1);
- The minimal gap between the estimated probability computed on the sum of the selected SVs estimates, mimicking the surrogate behavior by exploiting the efficiency axiom, and the sum of the selected normalized estimates, computed via least squares optimization (loss term L_2);
- The least squared error between the per-feature estimations and their marginal contribution (loss term L_3) [2];
- The efficiency constraint, i.e., the matching between the black-box model prediction and the probability computed on the sum of all per-feature SV estimates (loss term L_4).

The main paper contributions can be summarized as follows:

- **New approach**. It proposes LIGHTNINGSHAP, a new cost-effective, model-agnostic method for real-time SVs estimation that jointly approximates conditional expectations of the black-box model and computes SVs via least-squares minimization.
- **Best cost-benefit ratio on static tabular data**. It empirically evaluates both effectiveness, in terms of L2 distance against the exact estimates [10], and efficiency, in terms of training and inference times, on ten static benchmark tabular datasets with varying characteristics [2], [14], [15]. We compare LIGHTNINGSHAP with four neural, two regressor-based, and one sampling-based estimation strategies, achieving the best cost-benefit ratio (e.g., from 25% to 60% training time speed-up against FastSHAP with comparable errors under the same conditions).
- **Higher efficiency in dynamic real-world scenario**. It analyzes two real-world time-evolving datasets showcasing rapidly evolving scenarios of intrusion detection and market forecasting where black-box models are frequently retrained. LIGHTNINGSHAP halves the overall training times required by state-of-the-art neural explainers, resulting in the most cost-effective solution.
- **Higher efficiency on benchmark image datasets**. It compares the LIGHTNINGSHAP performance against state-of-the-art image explainers on two well-established image datasets, i.e., CIFAR [16] and Imagenette [17]. LIGHTNINGSHAP shows competitive Exclusion and Inclusion Area Under the Curve (AUC) while achieving a training time speed-up between 25% and 50%.

The remainder of this paper is organized as follows. Section II introduces the notation used throughout the paper and recalls the foundation behind our work. Section III reviews the existing literature on SVs estimation. Section IV thoroughly describes the LIGHTNINGSHAP approach. Section V reports the main experimental results. Finally, Section VII draws the conclusions and discusses the future research agenda.

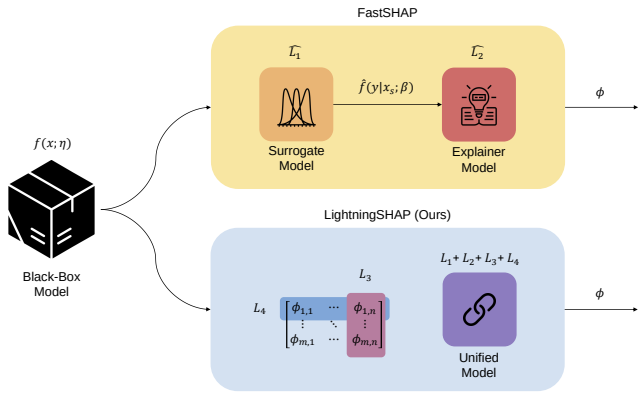


Fig. 1. The pipeline used for neural explanation of black-box machine learning models. Comparison between our approach LIGHTNINGSHAP (the pipeline depicted in the lower part) and the state-of-the-art neural explainer FASTSHAP (the pipeline in the upper part). See Table I for image notation.

TABLE I
OUTLINE OF THE NOTATION USED THROUGHOUT THE PAPER.

Symbol	Meaning
P	Players in a cooperative game
$S \subseteq P$	Player coalition (feature subset)
$\mathcal{D}$	Input dataset
i	Feature describing $\mathcal{D}$'s samples
$\mathbf{s} \in \{0,1\}^{\lvert P\rvert}$	Boolean feature vector indicating which features are included in S (1) or which ones are absent (0)
$x_{\mathbf{s}}$	Portion of sample x corresponding to the features in S
y	Target class in $\mathcal{D}$
$\lvert P\rvert$	Number of predictive features in $\mathcal{D}$
$x \subseteq D$	Sample in $\mathcal{D}$ described by features i and associated to a class y
$p(\mathbf{s})$	Probability of sampling S from the feature set in $\mathcal{D}$
$v_{x,y}$	Payoff of any player coalition on a sample x for target y
ϕ_i	Shapley Value of player $i \in P$
f	Black-box model
$\hat{f}$	Surrogate model
E	Explainer model
U	Unified model
σ	Softmax function
$\mathbf{1}$	Grand coalition
$\mathbf{0}$	Null coalition

II. PRELIMINARIES

Here we introduce the notion of Shapley Value and the preliminaries behind Shapley-based explainers. For the sake of clarity, Table I summarizes the notation that will be used hereafter in the paper.

A. Shapley Values

In cooperative games, the Shapley Value (SV) ϕ_i indicates the contribution of an individual player i to the overall payoff of a group of P players [18]. Let S be any subset of players,

usually denoted by *coalition*. The SV of i is calculated as the weighted sum of its marginal contributions across all the possible coalitions. Given a characteristic function $v_{x,y} : 2^{|P|} \to \mathbb{R}$, which defines the payoff for any subset of players $S \subseteq P$ on a sample x for a target class y, we denote $v_{x,y}(S)$ the value assigned to coalitions S. In the context of feature attribution, each feature acts as a player, and $v_{x,y}(S)$ represents the value that the coalition of active features S can jointly generate. The edge cases correspond to the null coalition, $v_{x,y}(\mathbf{0})$, and the grand coalition, $v_{x,y}(\mathbf{1})$, which respectively represent the value when no features are included and when all features are present. In this setting, the Shapley Value ϕ_i is given by:

$$\phi_i(v_{x,y}) = \sum_{S \subseteq P \setminus i} \frac{|S|!(|P| - |S| - 1)!}{|P|!} [v_{x,y}(S \cup i) - v_{x,y}(S)] \tag{1}$$

SVs satisfies the following axioms [18]:

- **Efficiency**: The sum of the SVs for all players is equal to the total payoff: $\sum_{i \in P} \phi_i(v_{x,y}) = v_{x,y}(P)$.
- **Symmetry**: Two players providing equal contribution to every possible coalition receive the same SV, i.e., if $\forall S \subseteq P$, $\phi_{S \cup i} - \phi_S = \phi_{S \cup j} - \phi_S$ then $\phi_i = \phi_j$.
- **Additivity**: The SV for the sum of two coalition games $v^1_{x,y}$ + $v^2_{x,y}$ is the sum of the SVs of each game, i.e. $\phi_i(v^1_{x,y} + v^2_{x,y}) = \phi_i(v^1_{x,y}) + \phi_i(v^2_{x,y})$.
- **Dummy Player**: The SV of a player that does not provide contributions to any coalition is equal to zero, i.e., if $\forall S \subseteq P, v_{x,y}(S \cup i) = v_{x,y}(S)$, then $\phi_i = 0$.

In the context of model explainability, Shapley Values can take both positive and negative values. A positive ϕ_i indicates that feature i contributes positively to the predicted outcome (i.e., it increases the model's output), while a negative ϕ_i suggests the opposite. The magnitude $|\phi_i|$ reflects the strength of the feature's influence, with larger absolute values indicating a greater impact on the model's prediction for the given sample.

B. Shapley Values for Explainable AI

Given a dataset $\mathcal{D}$ consisting of samples x of $|P|$ features, i.e. $x = (x_1, ..., x_{|P|})$, and a discrete class y, we train a black-box model f on $\mathcal{D}$ to accurately predict the target class y of an arbitrary sample x given its corresponding feature values.

Assuming that each feature i in $\mathcal{D}$ is a player in a coalitional game, according to the *local feature attribution* we leverage $\phi_i(v_{x,y})$ to explain the i's contribution to f on the prediction of class y for a sample x, also denoted by *explicand*.

Given a sample x, to approximate its SVs we typically consider subsets S of features in $\mathcal{D}$. With convenient abuse of notation, hereafter, we will also denote by $\mathbf{s} \in \{0, 1\}^{|P|}$ the feature vector corresponding to S, which indicates whether the features are present (1) or absent (0) in S. In the context of neural networks, the value function $v_{x,y}(S)$ becomes directly linked to the output of a predictor $\hat{f}(\cdot; \eta)$ able to handle a subset of features (e.g., Surrogate), such that:

$$v_{x,y}(S) = \hat{f}(x_{\mathbf{s}}; \eta), \tag{2}$$

where $x_{\mathbf{s}}$ is the masked version of the input sample x, retaining only the features belonging to S and replacing all others with a zero value. Accordingly, the null and grand coalitions correspond to:

$$v_{x,y}(\mathbf{0}) = \hat{f}(x_{\mathbf{0}}; \eta); \quad v_{x,y}(\mathbf{1}) = \hat{f}(x_{\mathbf{1}}; \eta) = \hat{f}(x; \eta) \tag{3}$$

It is worth noting that, although the classical game-theoretic formulation requires $v(\emptyset) = 0$, in the context of model explanation, $v_{x,y}(\mathbf{0})$ generally represents the *baseline output* of the model in the absence of any specific feature information. This value corresponds to the inherent bias of the trained model and often approximates the expected prediction over the training data [2]. The Shapley Values then quantify the deviation of the model output from this baseline, and the efficiency property ensures that:

$$\sum_i \phi_i = v_{x,y}(1) - v_{x,y}(0), \tag{4}$$

meaning that the total contribution of all features equals the change in model output from its baseline to its final prediction [2], [3], [5].

III. Background

According to [6], SV-based explainers can be classified according to their dependence to the black-box model as model-specific (see Section III-A for further details) or model-agnostic (see Section III-B for an overview of the most relevant studies). In this work, we will present a model-agnostic approach. While coping with image data, ad hoc techniques to generate visual explanation have also been proposed (a synthetic literature review is given in Section III-C).

A. Model-specific approaches

Model-specific approaches can be categorized as linear [19], tree-based [20], or deep explainers [2], [7]. Among the deep explainers, DeepSHAP [2] builds on a recursive attribution mechanism by attributing the difference in the model's output between an input sample and a background reference back to the input features. GradientSHAP combines integrated gradients [21] with SHAP values [2] using the gradients of the output with respect to the input features to approximate feature attributions. DASP [7], instead, introduces a polynomial-time algorithm that leverages uncertainty propagation through neural networks to estimate SVs. Proposing model-specific approaches to SVs estimations is out of the scope of this work.

B. Model-agnostic approaches

To make the SVs estimation problem tractable on large datasets, existing model-agnostic methods apply various SVs estimation and feature removal strategies [6]. Thereafter, we will describe the approaches most related to the present work.

a) Feature permutation and sampling: Permutation and sampling strategies [2], [22], [23] approximate Shapley values by randomly sampling feature coalitions, rather than exhaustively calculating each feature's marginal contribution across all possible combinations. While these approaches substantially reduce the computational effort, their accuracy is directly correlated with the number of sampling iterations

performed. Unlike recently proposed stratification methods (e.g., [23]), in this work we mainly consider Neural approaches as they commonly outperform classical sampling methods in terms of training efficiency (see Section V). Furthermore, we rely exclusively on the original training samples, as data augmentation methods (e.g., [24]) do not improve efficiency.

b) Regression methods: This class of estimators formulates the SVs estimation task as a weighted linear regression problem. For instance, KernelSHAP [2] generates its Shapley Values approximation ϕ by addressing the following task:

$$\begin{aligned}\phi(v_{x,y}) =& \arg\min_{\phi_{x,y}} \mathop{\mathbb{E}}_{p(\mathbf{s})} \left[\left(v_{x,y}(\mathbf{s}) - v_{x,y}(\mathbf{0}) - \mathbf{s}^\top \phi_{x,y} \right)^2 \right] \\ & \text{s.t.} \quad \mathbf{1}^\top \phi = v_{x,y}(\mathbf{1}) - v_{x,y}(\mathbf{0}),\end{aligned} \tag{5}$$

where $x_\mathbf{s}$ is the portion of the sample x corresponding to the its feature subset S. The estimator weights a sample of coalitions based on the concept of Shapley Kernel:

$$p(\mathbf{s}) \propto \frac{|P| - 1}{\binom{|P|}{\mathbf{1}^\top \mathbf{s}} \cdot \mathbf{1}^\top \mathbf{s} \cdot (|P| - \mathbf{1}^\top \mathbf{s})}, \tag{6}$$

where $p(\mathbf{s})$ is the sampling probability a feature subset S.

While KernelSHAP ensures empirical consistency, it faces two significant limitations: detecting convergence during computation since the number of required samples is not determined , and the presence of statistical bias in its estimates. To address these challenges, Unbiased KernelSHAP [5] introduces an improved dataset sampling methodology that utilizes a constrained least squares optimization approach with a computationally tractable number of samples, effectively eliminating the bias concern.

c) Surrogate model: The surrogate model is an external predictor aimed to learn the conditional expectation of the black-box model given every subset of features. Once trained, the surrogate model directly approximates the conditional expectations needed to compute the SVs explanations.

FastSHAP [3] is the state-of-the-art neural, model-agnostic SVs estimator relying on the concept of a surrogate model. Firstly, it learns a surrogate model $\hat{f}(\mathbf{y} \,|\, x_\mathbf{s}; \beta)$ by minimizing the Kullback-Leibler (KL) divergence [9] with respect to the output of black-box $f(x; \eta)$ [10], [25]:

$$\mathcal{L}(\beta) = \mathop{\mathbb{E}}_{p(\mathbf{x})} \mathop{\mathbb{E}}_{p(\mathbf{s})} \left[D_{KL} \left(f(\mathbf{x}; \eta) \,\|\, \hat{f}(\mathbf{y} \,|\, x_\mathbf{s}; \beta) \right) \right]. \tag{7}$$

Following, it exploits an explainer model to efficiently generate the SVs estimates without requiring any ground truth. FastSHAP returns the SVs approximation $\phi(x, y; \theta)$ for sample x and class y in a single-forward pass. Its objective function is inspired by the SVs' weighted least squares characterization [2] to enable efficient gradient-based optimization. More specifically, it learns an explainer E by minimizing the L2 distance between the probability output of the surrogate model and the sum of the predicted SVs:

$$\begin{aligned}&\mathcal{L}(\theta) = \\ &\mathop{\mathbb{E}}_{p(x)} \mathop{\mathbb{E}}_{U(y)} \mathop{\mathbb{E}}_{p(\mathbf{s})} \left[\left(\hat{f}_{x,y}(\mathbf{s}) - \hat{f}_{x,y}(\mathbf{0}) - \mathbf{s}^\top \phi(x, y; \theta) \right)^2 \right]\end{aligned} \tag{8}$$

where $\hat{f}_{x,y}$ is the probability value function represented by the surrogate model.

To meet the efficiency constraint and generate consistent SVs, FastSHAP applies a normalization factor to all predictions, namely the *additive efficient normalization* (N) , to generate its Shapley Values approximation ϕ^N:

$$\begin{aligned}\phi^N(x, y; \theta) =& \phi(x, y; \theta) \\ &+ \frac{1}{|P|} \left(\hat{f}_{x,y}(\mathbf{1}) - \hat{f}_{x,y}(\mathbf{0}) - \mathbf{1}^\top \phi(x, y; \theta) \right)\end{aligned} \tag{9}$$

Once trained, the surrogate and explainer models can be jointly used to generate the local SVs explanations in real time since the per-sample inference time of the neural explainer is typically negligible. As a drawback, the separate training of the surrogate model and explainer is time-consuming, especially while coping with large or high-dimensional datasets [10]. The impact of the training cost is particularly evident when training data and black-box models need to frequently updated (see Section VI-C). The aim of the present work is to overcome the efficiency issues of surrogate-based models.

C. Visual explainers

In the domain of computer vision, explainers also provide visual interpretations of feature importance other than Shapley Values. Specifically, gradient-based methods such as GradCAM [26], SmoothGrad [27], and IntegratedGradients [21] leverage the model's gradient information to identify the most important features. Conversely, learning-based approaches, e.g., CXPlain [4], train a separate explanation model that learns to predict the importance of input features by quantifying their marginal contributions to the model's performance. Despite the focus of our work is not primarily on visual interpretations, for the sake of completeness we also perform an empirical comparisons between LIGHTNINGSHAP and state-of-the-art image explainers (see Section VI).

IV. LIGHTNINGSHAP

LIGHTNINGSHAP is a new cost-effective, model-agnostic approach to local SVs estimation. It improves the existing surrogate model-based neural approaches by proposing a unified neural network model that on-the-fly computes the SVs estimates while approximating the conditional expectation of the original model. The core idea is to avoid decoupling the two core learning phases, i.e., the estimation of the conditional expectation of the black-box model (surrogate model) and the gradient-based optimization of the SVs explanations, by jointly addressing the learning tasks. To this end, we leverage the efficiency property of SVs that, coupled with the model formulations 7 and 8, allows the joint learning of a unified model capable of directly predicting SVs and simultaneously performing as a value function that approximates the behavior of the black-box model on feature subsets. As shown in the empirical results reported in Section V, adopting a unified model yields a substantial training time reduction, thus improving the efficiency compared to existing approaches [3].

LIGHTNINGSHAP generates SVs explanations $\phi(x, y; \vartheta)$ for sample x and label y in a single forward pass by learning a unified model U. It relies on the value function defined in Equation 8 and aligns the distribution of the black-box model predictions to the sum of the predicted SVs by minimizing the following combined loss function:

$$\begin{aligned}\mathcal{L}(\vartheta) = \mathop{\mathbb{E}}_{p(x)} \mathop{\mathbb{E}}_{\text{Unif}(y)} \mathop{\mathbb{E}}_{p(\mathbf{s})} \Big[L_1 + L_2 + L_3 + L_4 \Big] = \\ = \mathop{\mathbb{E}}_{p(x)} \mathop{\mathbb{E}}_{\text{Unif}(y)} \mathop{\mathbb{E}}_{p(S)} \Bigg[D_{KL} \Big(f(x;\eta) \,\|\, \mathbf{s}^\top \phi(x,y;\vartheta) \Big) \\ + \Big(\sigma\big(\mathbf{s}^\top \phi(x,y;\vartheta)\big) - \sigma\big(\mathbf{0}^\top \phi(x,y;\vartheta)\big) - \mathbf{s}^\top \phi^N(x,y;\vartheta) \Big)^2 \\ + \Big(\mathbf{1}^\top \Big(\sigma\big(\mathbf{s}^\top \phi_p(x,y;\vartheta)\big) - \sigma\big(\mathbf{0}^\top \phi_p(x,y;\vartheta)\big) - \phi_i^N(x,y;\vartheta) \Big) \Big)^2 \\ + \Big(f(x;\eta) - \sigma\big(\mathbf{1}^\top \phi(x,y;\vartheta)\big) \Big)^2 \Bigg]\end{aligned} \tag{10}$$

where f is the black-box model, σ is the softmax function. The loss function is composed of the following loss terms:

- L_1: The KL divergence between the conditional expectation of the black-box model given every feature subset S of the input sample x and the original distribution $f(x;\eta)$. It aligns the distribution of the black-box model when considering different subsets of features with the sum of the ϕ estimates.
- L_2: The squared error between the estimated probability computed on the sum of the SVs estimates and the sum of the normalized estimates. It minimizes the gap between the probability computed via softmax function on the sum of the SVs estimates with respect to the sum of the normalized estimates computed using the *additive efficient normalization* [3].
- L_3: The squared error of the per-feature SV estimations over all the dataset samples. In compliance with [2], it adjusts the SVs approximation of each feature to approximate the corresponding marginal probability (computed when only that particular feature is considered).[1]
- L_4: The efficiency constraint, i.e., the correspondence between the output from the black-box model and the probability computed on the sum of all SVs estimates.

A. Comparative error analysis

To highlight the key differences between the unified model adopted in LIGHTNINGSHAP and the classical two-step process [3], below we report a comparison between the corresponding SVs approximation errors.

The classical two-step process introduces two levels of approximations: the former can be quantified by the error introduced while approximating the behavior of the black-box model while training the surrogate model (hereafter denoted by ε_{SM}). The latter (ε_E) is the gap between the SVs estimators returned by the explainer and the actual ones overall features.

[1]For the sake of efficiency, on image datasets we disable this loss term to perform pixel-wise estimations.

$$\begin{aligned}\varepsilon_{SM} &= f(x;\eta) - \hat{f}(y|x_S;\beta) \\ \varepsilon_E &= \sum_{i\in P}(\phi_i^f - \phi_i^E) = \sum_{i\in P}\phi_i^f - \sum_{i\in P}\phi_i^E\end{aligned} \tag{11}$$

where $f(x;\eta)$ is the black-box model output for sample x, $\hat{f}(y|x_s;\beta)$ is the output of the surrogate model, ϕ_i^f is the sum of the actual SV computed on f over all features i, and ϕ_i^E is the sum of the feature-i SV computed by the explainer E.

Similarly, the approximation error made by the unified model U in LIGHTNINGSHAP can be decoupled as follows:

$$\begin{aligned}\varepsilon_{LS}^1 &= f(x;\eta) - \sum_{i\in\mathcal{D}}\phi_i^U \\ \varepsilon_{LS}^2 &= \sum_{i\in P}(\phi_i^f - \phi_i^U) = \sum_{i\in P}\phi_i^f - \sum_{i\in P}\phi_i^U\end{aligned} \tag{12}$$

where ε_{LS}^1 is the gap between the distribution of the black-box model function f and the sum of the SV ϕ_i^U for feature i predicted by our unified model U, whereas ε_{LS}^2 is the cumulative error over all features between the actual SVs computed on f and the SVs estimated by U.

By enforcing the efficiency constraint [1], the overall approximation errors of the two approaches, respectively denoted by ε_{FS} (traditional two-step approach) and ε_{LS} (LIGHTNINGSHAP, ours), can be reformulated as follows:

$$\begin{aligned}\varepsilon_{FS} &= \varepsilon_{SM} + \varepsilon_{EX} \\ &= f(x;\eta) - \hat{f}(y|x_S;\beta) + f(x;\eta) - \sum_{i\in P}\phi_i^E \\ &= 2f(x;\eta) - \hat{f}(y|x_S;\beta) - \sum_{i\in P}\phi_i^E \\ \varepsilon_{LS} &= \varepsilon_{LS}^1 + \varepsilon_{LS}^2 \\ &= f(x;\eta) - \sum_i \phi_i^U + f(x;\eta) - \sum_{i\in P}\phi_i^U \\ &= 2f(x;\eta) - 2\sum_{i\in P}\phi_i^U\end{aligned} \tag{13}$$

While the approximation error of the two-step approach has two separate contributions from the surrogate model and the explainer, the unified model U in LIGHTNINGSHAP incorporates both approximations.

V. EXPERIMENTAL DESIGN

We run the experiments on a system equipped with an Intel i9-10980XE CPU and an Nvidia A6000 GPU. All results are averaged over five runs. The LIGHTNINGSHAP code is available at https://github.com/DavideNapolitano/LightningSHAP.

A. Datasets and black-box models

a) Tabular datasets: We analyze 10 datasets in tabular form publicly available in the UCI [14], OpenML [15], and SHAP [2] repositories. They are commonly used for binary classification benchmarking and come from diverse domains (e.g., finance, healthcare). Table II summarizes the main dataset characteristics. They exhibit significant variability in the number of samples (from 10^2 to 10^5), features (from 4 to 55), and imbalance ratio (from 12/88 to 71/29).

TABLE II
DATASET STATISTICS. *PC* STANDS FOR "PER CLASS"

Dataset	Source	Train samples	Val. samples	Num. features	Imbalance ratio %	Model acc %
Tabular datasets						
Diabetes	OpenML	491	123	8	35 / 65	74.68±1.35
Bank	UCI	877	220	4	44 / 56	99.49±0.25
Bankruptcy	UCI	2892	724	16	12 / 88	91.99±0.12
Phoneme	UCI	3458	865	5	71 / 29	88.16±0.57
Mozilla	OpenML	9948	2488	4	67 / 33	93.21±0.08
Magic	UCI	12172	3044	10	65 / 35	88.43±0.36
Credit	UCI	19200	4800	23	22 / 78	81.13±0.18
Jannis	UCI	18458	4684	55	50 / 50	77.75±0.26
Census	SHAP	20838	5210	12	26 / 74	87.30±0.16
Click	OpenML	25566	6392	11	17 / 83	83.80±0.29
Image datasets						
Imagenette	TF	50000	5000	224x224	10 *PC*	97.15±0.08
CIFAR10	TF	9469	1962	224x224	10 *PC*	87.00±0.16
Dynamic datasets						
KDDCup99	UCI	$3.5 \cdot 10^6$	$1.5 \cdot 10^6$	31	80 / 20	99.02±0.10
BTCData	Kaggle	31246	13392	17	42 / 58	77.69±0.06

b) Image datasets: We consider two established image classification datasets, i.e., CIFAR10 [16] and Imagenette [17]. We retrieve both data sources from the TensorFlow repository. CIFAR10 consists of 60,000 32x32 color images in 10 distinct classes commonly used for image classification tasks. Imagenette is a sample of the more extensive ImageNet collection. It contains 13394 images divided into 10 classes and is designed to benchmark image classification models.

c) Dynamic datasets: We explore the efficiency and effectiveness of LIGHTNINGSHAP in two real-world scenarios where black-box models need to be periodically retrained, i.e., intrusion detection for cybersecurity applications and financial market forecasting [13]. For intrusion detection, we consider the KDD Cup 1999 Data available in the UCI repository [14]. The goal is to learn a predictive model capable of distinguishing between "bad" connections, namely intrusions or attacks, and "good" (normal) connections. For financial market forecasting, instead, we consider the historical Bitcoin prices released for a Kaggle challenge (https://www.kaggle.com/datasets/), where the goal is to predict whether the cryptocurrency price will increase or not. Since both tasks require periodic black-box model updates, we simulate dynamic scenarios by replicating the training of black-box and explainer models once per day for intrusion detection and once per month for market forecasting.

d) Black-box models: We adopt a MultiLayer Perceptron as black-box model for both static tabular and dynamic datasets. The model accuracies vary, according to the datasets and samples, between 74% and 99%. Instead, on image datasets we use a ResNet50 pre-trained for classification as a black-box model, achieving 87% accuracy on CIFAR10 and 97.15% on Imagenette.

B. Baselines

We compare the performance of LIGHTNINGSHAP with that of the following baseline methods: **Neural Estimators**: FastSHAP [3], DASP [7], GradientSHAP [2], and DeepSHAP [2]. **Sampling**: Monte Carlo [28] and Permutation [5], CCN [23]. **Regression**: KernelSHAP [2] and Unbiased KernelSHAP [5]. **Exact Computation**: Exact [2].

On image data, we test CXPlain [4] three gradient-based attribution methods, i.e., GradCAM [26], Integrated-Gradient [21], and SmoothGrad [27], and KernelSHAP and DeepSHAP [2] as for tabular data. We explore the configuration settings recommended by the respective authors. Default parameters have proved to perform best for Surrogate and FastSHAP. We run CCN and Monte Carlo estimator for 1,000 iterations. Unbiased KernelSHAP uses 128 reference examples for marginal imputation. Deep- and GradientSHAP rely on 1,000 background samples, while the permutation is based on $12000/|P|$ samples. All information about LIGHTNINGSHAP architecture is provided in the code repository.

C. Metrics

Similar to [3], we measure the explainers' effectiveness on tabular data in term of the L2 distance between the SVs estimates and the exact SVs.[2] Analogously to [3], we measure the explanation accuracy on image data using the Exclusion AUC and Inclusion AUC metrics. They respectively measure the model's performance improvement/degradation when features are systematically added/removed in order of attributed importance, i.e., they indirectly evaluate the quality of the SV-based feature ranking. Finally, we measure both training and inference times to quantify explainers' efficiency. To make the results comparable across datasets computational times all refer to 1000 samples.

VI. RESULTS

A. Effectiveness analysis

a) Tabular data: We compare the estimation errors of LIGHTNINGSHAP and baseline methods separately on static tabular data (see Table III). LIGHTNINGSHAP ranks first among neural explainers on tabular data (i.e., it is the best performer on 5 out of 10 datasets). For small datasets (e.g., Diabetes) the efficiency improvement is less evident, but the accuracy of SVs estimates is still significantly better.

b) Image data: The plots in Figure 2 compare the Inclusion AUC and Exclusion AUC metrics achieved on the image datasets. They show the influence of sample selection in order of increasing feature importance (SVs) with varying percentages of included/excluded samples. LIGHTNINGSHAP achieves best-in-class performance on both datasets similarly to FastSHAP and places second behind KernelSHAP on Exclusion. A qualitative analysis of the results is reported in the Supplementary Material.

B. Efficiency analysis

a) Training cost: Table IV compares the LIGHTNINGSHAP training times on static tabular data with those of FastSHAP, i.e., the latest neural SVs estimator. Thanks to

[2]Whenever Exact is not computationally feasible on a dataset, we use, similar to [3], Unbiased KernelSHAP [5] to generate the ground truth SVs.

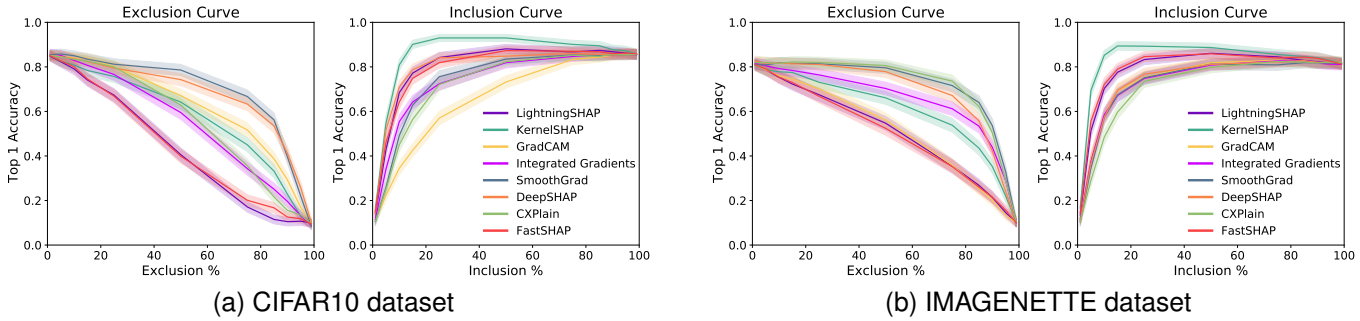


Fig. 2. Inclusion AUC (I-AUC) and Exclusion AUC (E-AUC) by varying the percentage of included/excluded samples. Image dataset. Inclusion AUC: the larger the better. Exclusion AUC: the lower the better.

its joint learning process, LIGHTNINGSHAP achieves a 25%-60% training speed-up. The efficiency improvements hold on most datasets, except for the very small ones (e.g., Diabetes) for which the differences are negligible. Table V reports similar statistics on image data, where LIGHTNINGSHAP achieves a 30%-55% speed-up on training compared to FastSHAP and places second behind KernelSHAP.

b) Inference cost: Tables VI and V compare the inference times of LIGHTNINGSHAP and other methods on image and tabular data, respectively. The LIGHTNINGSHAP times are the best on 9 out 10 tabular data and the best on image datasets.

c) Variation of the overall computational cost: Figure 3 shows, for two representative tabular datasets, the overall computational cost (training+testing times) while varying the number of tested samples. Both LIGHTNINGSHAP and FastSHAP require an initial training cost, which is then compensated by a lower impact of the per-sample inference cost. After few tens of thousands of samples, LIGHTNINGSHAP achieves the lowest overall computational cost across all tested explainers. By testing additional samples, the gap between LIGHTNINGSHAP and the other methods further increases.

d) Dataset complexity: Figure 4 shows the effect of dataset complexity, defined by $|D| \times |P|$, on the efficiency-effectiveness relation for LightningSHAP and FastSHAP. On datasets with medium/high complexity, our approach outperforms FastSHAP (the higher the better). In contrast, on small datasets the LightningSHAP's training time becomes slightly worse, but its estimation error remains significantly better.

C. Application to real-world dynamic scenarios

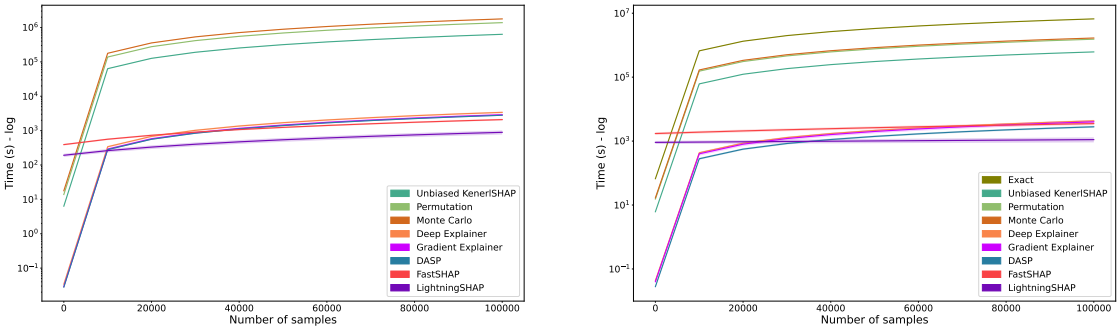
We showcase the advantages of LIGHTNINGSHAP over existing neural SVs estimators in two real time-evolving scenarios, i.e., network intrusion detection and financial market forecasting, with the latter reported in the Supplementary Material due to space constraints.

a) Network intrusion detection: This task is established for cybersecurity applications [12]. In the KDDCup 1999 challenge [14] the organizers released anonymized data about military network connection records including, amongst other, connection duration and protocol, TCP connection features, and statistics about the exchanged content (e.g., SYN and REJ errors). They asked participants to predict whether a given connection is an intrusion/attack or not. Due to the highly dynamic nature of network traffic data and cyberattacks prediction models need to frequently retrained to capture most recent discriminating patterns [12]. Hence, we simulate a dynamic scenario where the black-box prediction model (i.e., an MLP network) is retrained once per day and applied to the samples occurring the next day. Then, to provide end-users with insights into model predictions, we update the explainer on a daily basis, testing its efficiency in a dynamic scenario.

Table VII compares the effectiveness of LIGHTNINGSHAP and other best performing neural explainers separately for each

TABLE III
L2 DISTANCE FROM THE GROUND TRUTH SVS ON STATIC TABULAR DATASETS. THE BEST PERFORMER PER DATASET AND EXPLAINER TYPE (CLASSICAL OR NEURAL) IS WRITTEN IN BOLDFACE.

Dataset	Classical explainers				Neural explainers				
	Unbiased KernelSHAP	Permutation	CCN	Monte Carlo	Deep SHAP	Gradient SHAP	DASP	Fast SHAP	Lightning SHAP
Diabetes	0.004±0.001	0.004±0.001	0.164±0.018	0.288±0.030	0.168±0.019	**0.059±0.004**	0.292±0.020	0.127±0.009	0.070±0.005
Bank	0.005±0.001	0.005±0.001	0.363±0.028	0.618±0.073	**0.110±0.011**	0.177±0.012	0.722±0.048	0.189±0.010	0.183±0.012
Bankruptcy	-	0.003±0.001	0.069±0.014	0.129±0.023	0.047±0.005	**0.027±0.004**	0.211±0.012	0.049±0.004	0.038±0.004
Phoneme	0.003±0.001	0.005±0.001	0.161±0.025	0.283±0.041	0.115±0.009	0.085±0.007	0.320±0.021	0.133±0.005	**0.075±0.006**
Mozilla	0.002±0.001	0.003±0.001	0.247±0.033	0.425±0.055	0.220±0.016	0.220±0.015	0.498±0.030	0.192±0.010	**0.093±0.007**
Magic	0.008±0.001	0.007±0.001	0.231±0.036	0.399±0.059	0.170±0.013	0.174±0.012	0.619±0.041	0.185±0.011	**0.167±0.009**
Credit	-	0.003±0.001	0.049±0.010	0.097±0.016	0.150±0.014	0.052±0.006	0.183±0.010	0.083±0.006	**0.033±0.002**
Jannis	-	0.012±0.002	0.157±0.019	0.276±0.032	0.180±0.015	**0.091±0.003**	0.297±0.025	0.178±0.010	0.145±0.008
Census	0.003±0.001	0.004±0.001	0.164±0.022	0.233±0.037	0.078±0.008	**0.045±0.004**	0.515±0.027	0.087±0.003	0.048±0.004
Click	0.001±0.001	0.001±0.001	0.042±0.010	0.084±0.016	0.029±0.004	0.023±0.007	0.374±0.028	0.019±0.004	**0.014±0.002**
Avg. Rank	**1.1**	1.4	2	3	3	2	5	3.2	**1.7**



(a) Bankruptcy dataset (b) Census dataset

Fig. 3. Overall computational times (training+testing times, in seconds and logarithmic scale) by varying the number of tested samples.

TABLE IV
TRAINING TIMES COMPARISON BETWEEN LIGHTNINGSHAP AND FASTSHAP ON STATIC TABULAR DATASETS. TIME IS EXPRESSED IN SECONDS. THE BEST PERFORMER PER DATASET IS WRITTEN IN BOLDFACE.

Dataset	Fast SHAP	Lightning SHAP	Average Speed-up
Diabetes	**66±12**	98±8	-48%
Bank	**197±19**	298±42	-51%
Bankruptcy	392±23	**192±20**	**+51%**
Phoneme	671±71	**388±45**	**+42%**
Mozilla	1356±98	**652±83**	**+52%**
Magic	1644±142	**705±78**	**+57%**
Credit	1847±123	**720±91**	**+61%**
Jannis	2572±278	**1684±298**	**+25%**
Census	1723±165	**911±101**	**+47%**
Click	2113±178	**1238±156**	**+41%**
Avg. Rank	1.8	**1.2**	

TABLE V
EXPLAINERS' TRAINING AND INFERENCE TIMES ON IMAGE DATASETS. THE BEST PERFORMER PER DATASET IS WRITTEN IN BOLDFACE.

	Explainer	CIFAR-10	IMAGENETTE
Inference (s)	FastSHAP	2.681	2.655
	LightningSHAP	**2.368**	**2.575**
	KernelSHAP	5906.943	7932.343
	GradCAM	51.275	59.973
	IntGrad	132.475	110.157
	SmoothGrad	110.924	86.093
	DeepSHAP	317.237	345.159
	CXPlain	2.932	2.997
Train (s)	FastSHAP	43377.021	6861.328
	LightningSHAP	30532.924	3133.781
	KernelSHAP	**20446.724**	**2434.375**
	CXPlain	33049.603	4865.368

day. Our approach performs the best in four out of six days and achieves competitive errors in the remaining ones. Figure 5 shows an efficiency comparison in terms of cumulative computational times, i.e., explainer training+testing times, spent over consecutive days. LIGHTNINGSHAP demonstrates superior adaptability to dynamic scenarios, significantly reducing the computational overhead compared to other explainers (up to 70% w.r.t. FastSHAP) after their periodic retraining.

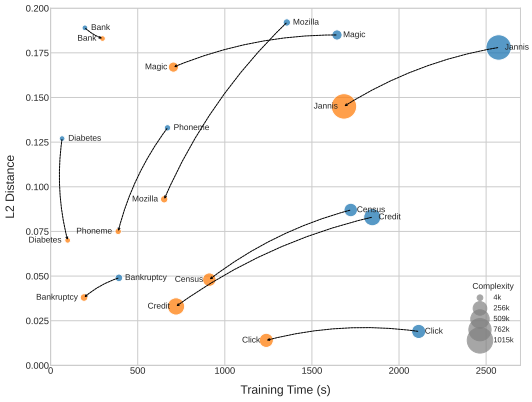


Fig. 4. Relation between efficiency and effectiveness (L_2 error vs. training time) on datasets with different complexities (dimensionality $\times$ cardinality) for LightningSHAP (ours, colored in orange) and FastSHAP [3] (in blue).

D. Ablation Study

a) Loss function: We examine the impact of different loss settings on LIGHTNINGSHAP performance. Both $L1$ and $L2$ loss terms are essential to accurately approximate the behavior of the black-box model and the SVs. Hence, we separately analyze the effect of adding the $L3$ term, which enforces the loss on each feature individually, and the $L4$ one, which aims to minimize the discrepancy between the output of the black-box model and those of our generalized model when

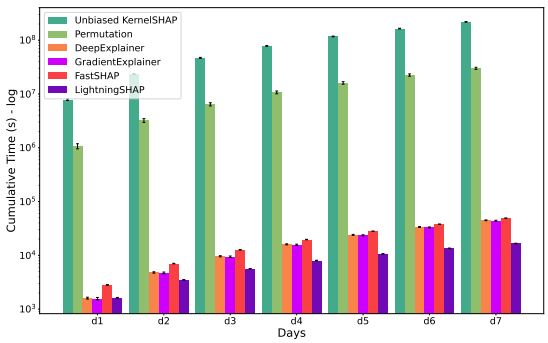


Fig. 5. Efficiency comparison between neural explainers. Cumulative computational times (training+inference times, expressed in seconds and visualized in logarithmic scale) spent in each period

TABLE VI
INFERENCE TIME COMPARISON BETWEEN EXPLAINERS ON STATIC TABULAR DATASETS. TIME IS EXPRESSED IN SECONDS. THE BEST PERFORMER PER DATASET IS WRITTEN IN BOLDFACE.

	Classical explainers					Neural explainers				
Dataset	**Exact**	**Unbiased KernelSHAP**	**Permutation**	**CCN**	**Monte Carlo**	**Deep Explainer**	**Gradient Explainer**	**DASP**	**Fast SHAP**	**Lightning SHAP**
Diabetes	2.377±0.023	2.342±0.017	12.345±0.092	1.134±0.006	7.826±0.064	0.020±0.001	0.022±0.001	0.012±0.001	0.016±0.001	**0.004±0.001**
Bank	0.095±0.001	1.789±0.014	13.795±0.087	0.747±0.006	4.193±0.045	0.029±0.002	0.030±0.003	0.011±0.001	0.017±0.001	**0.007±0.001**
Bankruptcy	-	6.340±0.074	13.837±0.123	2.193±0.027	17.794±0.267	0.034±0.002	0.029±0.002	0.028±0.001	0.017±0.001	**0.007±0.001**
Phoneme	0.224±0.012	8.655±0.118	14.170±0.148	0.894±0.009	5.572±0.094	0.031±0.004	0.033±0.001	0.014±0.001	0.017±0.001	**0.008±0.001**
Mozilla	0.085±0.007	0.264±0.041	13.127±0.111	0.789±0.012	4.590±0.112	0.025±0.001	0.025±0.001	0.009±0.001	0.017±0.001	**0.005±0.001**
Magic	11.523±0.108	23.541±0.712	11.717±0.198	1.482±0.021	11.116±0.198	0.020±0.002	0.020±0.001	0.014±0.001	0.018±0.001	**0.003±0.001**
Credit	-	5.801±0.087	14.888±0.201	2.493±0.054	20.615±0.514	0.023±0.001	0.025±0.002	0.025±0.001	0.017±0.001	**0.006±0.001**
Jannis	-	11.384±0.138	12.959±0.085	9.873±0.150	90.132±1.412	0.023±0.001	0.021±0.001	0.147±0.010	0.018±0.001	**0.003±0.001**
Census	66.476±0.973	6.158±0.012	15.384±0.102	2.082±0.033	16.765±0.312	0.043±0.003	0.040±0.005	0.028±0.002	0.018±0.002	**0.002±0.001**
Click	65.268±0.914	6.648±0.886	25.226±0.397	2.994±0.012	25.338±0.118	0.044±0.004	0.034±0.004	0.029±0.003	**0.012±0.002**	0.014±0.002
Avg. Rank	6.9	6.7	8.0	6.4	7.8	4.2	4.2	2.8	2.4	**1.1**

TABLE VII
KDD99 INTRUSION DETECTION CHALLENGE. L2 ERRORS (X10 SCALED-UP) OF THE NEURAL EXPLAINERS OVER SIX CONSECUTIVE DAYS. THE BEST PERFORMER PER DAY IS WRITTEN IN BOLDFACE.

Day	**Permut.**	**Deep SHAP**	**Gradient SHAP**	**Fast SHAP**	**Lightning SHAP**
d1	0.019±0.001	0.307±0.015	0.490±0.025	0.324±0.016	**0.269±0.013**
d2	0.022±0.001	0.285±0.014	**0.215±0.011**	0.298±0.015	0.245±0.012
d3	0.018±0.001	0.312±0.016	0.298±0.010	0.287±0.014	**0.256±0.013**
d4	0.021±0.001	**0.189±0.009**	0.295±0.015	0.278±0.014	0.234±0.012
d5	0.020±0.001	0.298±0.015	0.289±0.014	0.315±0.016	**0.275±0.014**
d6	0.017±0.001	0.315±0.016	**0.185±0.009**	0.292±0.015	0.243±0.012
d7	0.023±0.001	0.294±0.015	0.308±0.015	0.326±0.016	**0.278±0.014**
Avg. Rank	1.0	2.71	2.57	3.29	**1.43**

TABLE VIII
ABLATION STUDY ON LOSS TERMS CONTRIBUTION. TABULAR DATASETS.

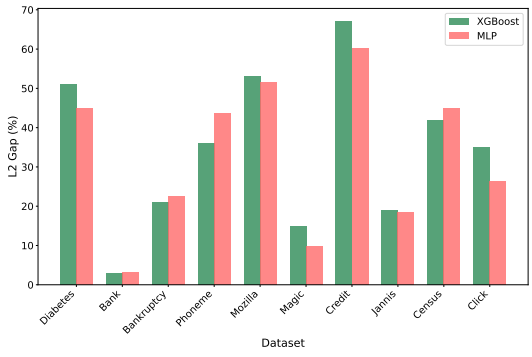
	L1+L2	**L1+L2+L3**	**L1+L2+L4**	**L1+L2+L3+L4**
Diabetes	0.079±0.006	0.069±0.004	**0.066±0.005**	0.070±0.005
Bank	0.186±0.007	0.183±0.006	**0.179±0.003**	0.183±0.012
Bankruptcy	**0.038±0.002**	0.046±0.004	0.044±0.002	**0.038±0.004**
Phoneme	0.078±0.005	**0.072±0.005**	0.077±0.008	0.075±0.006
Mozilla	0.095±0.006	0.096±0.005	0.095±0.006	**0.093±0.007**
Magic	0.171±0.009	**0.158±0.007**	0.168±0.010	0.167±0.009
Credit	0.038±0.003	0.033±0.002	**0.032±0.001**	0.033±0.002
Census	0.049±0.005	0.051±0.004	0.050±0.003	**0.048±0.004**
Jannis	0.145±0.009	0.149±0.008	0.147±0.004	**0.145±0.008**
Avg. Rank	3.22	2.67	2.22	**1.67**

all features are considered. Table VIII compares the results of four different configurations corresponding to different loss terms' combinations on tabular data. By leveraging all loss components together LIGHTNINGSHAP surpasses the performance of each specific ablation.

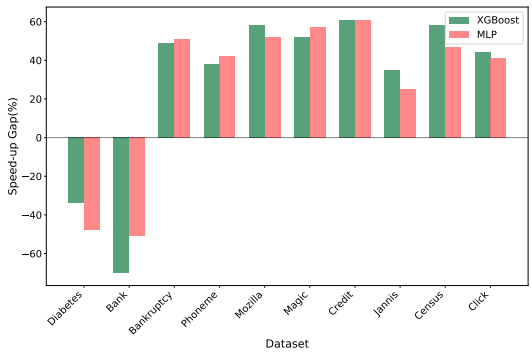
We also carry out the ablation study of loss L_4 on image data as well (as discussed in Section IV we disregard the $L3$ term on this data type). On CIFAR10 and Imagenette datasets incorporating the $L4$ loss term positively lower the AUC Exclusion (e.g., from 0.445 to 0.419 on CIFAR10 and from 0.525 to 0.499 on Imagenette) while yielding slight improvements on AUC Inclusion (e.g., from 0.796 to 0.798 on CIFAR10 and from 0.774 to 0.784 on Imagenette).

We empirically seek the optimal weighting in the function combining the loss terms. In the best setting for tabular data all terms are equally important. Instead, for image data we observe benefits by scaling up loss terms L_1 by 10 and L_4 by 100. The setting could be retuned for each domain separately.

b) Black-box model: Figure 6 shows the performance gaps between LightningSHAP and FastSHAP on tabular data in terms of L_2 error and training time using two different black-box models, i.e., MLP and XGBoost. The performance variations appear to be weakly influenced by the choice of the black-box model. Similar results hold for the image datasets (e.g., on the Imagenette dataset, the training time speedup of



(a) L_2 error



(b) Training time

Fig. 6. Percentage gaps in performance between LightningSHAP (ours) and FastSHAP [3] using different black-box models.

LightningSHAP vs. FastSHAP: ResNet 67%, CNN 51%.)

VII. Conclusions

This paper proposes a new efficient neural approach to compute Shapley Values, addressing the computational challenges of surrogate-based explainer that typically require separate steps for conditional expectation learning and Shapley Value estimation. Our unified approach, namely LightningSHAP, jointly performs the above steps, achieving a 25-60% reduction in the overall computational time on medium-large tabular data and 30-55% reduction on image data. Based on a comparison between LightningSHAP with state-of-the-art explainers, it turns out to be the most *cost-effective*, i.e., not only more efficient but also as effective as the state of the art.

We envision the application of LightningSHAP in dynamic scenarios where the training phase of both black-box model and explainer need to be repeated multiple times. To this end, we analyze two examples of real-world use-cases, i.e., network intrusion detection and financial market forecasting. The empirical results confirm the better usability of LightningSHAP compared to existing approaches.

As future work, we aim to address the following limitations of existing approaches: (1) The inability to produce incremental SV-based explanations, and (2) The weak integration between SV-based explainers and LLM-based generative models.

References

[1] L. S. Shapley, "A value for n-person games," in *Contributions to the Theory of Games II*, H. W. Kuhn and A. W. Tucker, Eds. Princeton: Princeton University Press, 1953, pp. 307–317.

[2] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017, pp. 4765–4774.

[3] N. Jethani, M. Sudarshan, I. C. Covert, S. Lee, and R. Ranganath, "Fastshap: Real-time shapley value estimation," in *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[4] P. Schwab and W. Karlen, "CXPlain: Causal Explanations for Model Interpretation under Uncertainty," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[5] I. Covert and S.-I. Lee, "Improving kernelshap: Practical shapley value estimation using linear regression," in *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, A. Banerjee and K. Fukumizu, Eds., vol. 130. PMLR, 13–15 Apr 2021, pp. 3457–3465.

[6] H. Chen, I. C. Covert, S. M. Lundberg, and S.-I. Lee, "Algorithms to estimate shapley value feature attributions," *Nature Machine Intelligence*, vol. 5, no. 6, pp. 590–601, 2023.

[7] M. Ancona, C. Öztireli, and M. H. Gross, "Explaining deep neural networks with a polynomial time algorithm for shapley value approximation," in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 2019, pp. 272–281.

[8] E. Štrumbelj and I. Kononenko, "Explaining prediction models and individual predictions with feature contributions," *Knowledge and information systems*, vol. 41, pp. 647–665, 2014.

[9] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[10] C. Frye, D. de Mijolla, L. Cowton, M. Stanley, and I. Feige, "Shapley-based explainability on the data manifold," *CoRR*, vol. abs/2006.01272, 2020.

[11] S. L. Chau, K. Muandet, and D. Sejdinovic, "Explaining the uncertain: Stochastic shapley values for gaussian process models," *arXiv preprint arXiv:2305.15167*, 2023.

[12] M. Saied, S. Guirguis, and M. Madbouly, "Review of artificial intelligence for enhancing intrusion detection in the internet of things," *Engineering Applications of Artificial Intelligence*, vol. 127, p. 107231, 2024.

[13] W. Bao, Y. Cao, Y. Yang, H. Che, J. Huang, and S. Wen, "Data-driven stock forecasting models based on neural networks: A review," *Information Fusion*, vol. 113, p. 102616, 2025.

[14] K. Bache and M. Lichman, "UCI machine learning repository," 2013.

[15] J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo, "Openml: Networked science in machine learning," *SIGKDD Explorations*, vol. 15, no. 2, pp. 49–60, 2013.

[16] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10 (canadian institute for advanced research)."

[17] J. Howard, "Imagenette: A smaller subset of 10 easily classified classes from imagenet," March 2019.

[18] L. S. Shapley, *Notes on the N-Person Game II: The Value of an N-Person Game*. Santa Monica, CA: RAND Corporation, 1951.

[19] H. Chen, J. D. Janizek, S. Lundberg, and S.-I. Lee, "True to the model or true to the data?" 2020.

[20] S. M. Lundberg, G. G. Erion, H. Chen, A. J. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S. Lee, "From local explanations to global understanding with explainable AI for trees," *Nat. Mach. Intell.*, vol. 2, no. 1, pp. 56–67, 2020.

[21] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 06–11 Aug 2017, pp. 3319–3328.

[22] E. sSrumbelj and I. Kononenko, "Explaining prediction models and individual predictions with feature contributions," *Knowledge and Information Systems*, vol. 41, pp. 647–665, 2014.

[23] J. Zhang, Q. Sun, J. Liu, L. Xiong, J. Pei, and K. Ren, "Efficient sampling approaches to shapley value approximation," *Proceedings of the ACM on Management of Data*, vol. 1, no. 1, pp. 1–24, 2023.

[24] I. Covert, C. Kim, S.-I. Lee, J. Y. Zou, and T. B. Hashimoto, "Stochastic amortization: A unified approach to accelerate feature and data attribution," *Advances in Neural Information Processing Systems*, vol. 37, pp. 4374–4423, 2024.

[25] N. Jethani, M. Sudarshan, Y. Aphinyanaphongs, and R. 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*. PMLR, 2021, pp. 1459–1467.

[26] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: visual explanations from deep networks via gradient-based localization," *International journal of computer vision*, vol. 128, pp. 336–359, 2020.

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

[28] R. Y. Rubinstein and D. P. Kroese, *Simulation and the Monte Carlo method*. John Wiley & Sons, 2016.

Davide Napolitano received the B.S. degree from the *Politecnico di Torino* university, Torino, Italy, in 2019, and the M.S. degree, also from the *Politecnico di Torino*, in 2022, where he is currently working toward the Ph.D. degree. His research interests include the use of deep learning in multimodal and explainability scenarios.

Luca Cagliero has been associate professor at the Dipartimento di Automatica e Informatica of the Politecnico di Torino since January 2020 and coordinator of the SmartData@PoliTo center since January 2024. His current research interests are in the fields of pattern mining and Deep Natural Language Processing. Specifically, he has worked on text summarization, classification and association rule mining. He has published 100+ papers in international journals and conference proceedings.