OperatorSHAP: Fast and Accurate Shapley Value Estimation for Neural Operators

Joshua Stiller
LMU Munich, MCML
joshua.stiller@lmu.de

Santo M. A. R. Thies
LMU Munich, MCML
santo.thies@ifi.lmu.de

Felix Czaja
LMU Munich
felix.czaja@ifi.lmu.de

Eyke Hüllermeier
LMU Munich, MCML, DFKI
eyke@lmu.de

Abstract

Understanding model predictions is essential for physical applications, where outputs often inform safety-critical decisions, such as structural load assessment, weather warnings, and clinical diagnosis. Shapley values satisfy many desirable properties as an attribution method, but their computational cost during inference hinders their practical use. Current amortized explainers, such as FastSHAP, are limited to homogeneous inputs, which is problematic for physical applications where data often comes from irregular grids and geometries. We introduce OperatorSHAP, a grid-agnostic attribution method that allows us to train FastSHAP-like explainers for neural operators. We establish a theoretical framework for attributions in function space, connecting to Aumann–Shapley values. We further show that OperatorSHAP's explanations are consistent with state-of-the-art discrete Shapley values across resolutions and transfer across grid sizes without retraining.

1 Introduction

Neural operators [19, 18] learn maps between function spaces and are grid-agnostic by design. The same trained model accepts a function sampled on 128 points or on 1,024 points. This flexibility is essential for physical applications, where datasets often contain samples with heterogeneous grids, coming from irregular geometries or sensor placements (see Fig. 1).

Fast and reliable explanations of their predictions are important for widespread industrial adoption, where users need to assess the credibility of high-stakes outputs such as structural load estimations or weather warnings. High explanation speeds are particularly relevant in commercial settings, where models are continuously monitored or a large number of users should be supplied with feature attributions. Amortized Shapley explainers, such as FastSHAP [15], are among the most established and efficient explanation tools in the field of explainable AI. However, they are restricted to homogeneous inputs, which limits their applicability. In addition, it is not obvious how to define amortized Shapley values on heterogeneous meshes, as nodes on test meshes may not be present in any input sample.

Aumann–Shapley values [1] provide a theoretical extension of Shapley values to games with infinite players. In this setting, values are defined similarly to densities over the input domain, where the attribution of a coalition can be calculated by integrating this density over the spatial region corresponding to that coalition. Showing that a game has a well-defined Aumann–Shapley value is a non-trivial task, especially in the function space setting of neural operators. In this paper, we

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Method	Inference speed	Grid flexibility
Classical SHAP (e.g., KernelSHAP)	Slow	Any grid
FastSHAP	Fast	Homogeneous grid only
OperatorSHAP	Fast	Any grid

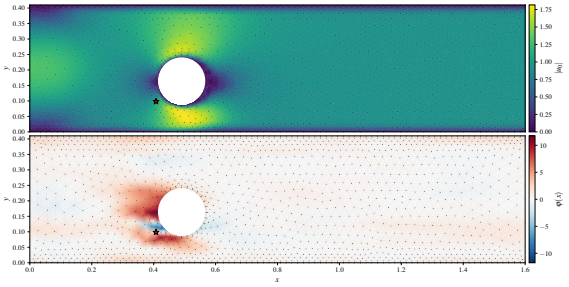


Figure 1: *Left:* Table with an overview of current methods. *Right:* A sample from the MeshGraphNets cylinder-flow dataset [31]. Top shows a heatmap of airflow velocity strengths, with a grid of sensor points, and the cylinder object obstructing the airflow. The bottom shows the attribution map from OperatorSHAP for the same sample. The explained spatial point is marked with a red star. The varying cylinder placement in this dataset yields heterogeneous meshes, but OperatorSHAP is able to give correct explanations. Here, the area upstream of the cylinder and along its lower side is positively relevant, as the air flows from left to right.

investigate these theoretical foundations and show how to design grid-agnostic explainers for neural operators. Our contributions are as follows:

1. **Theoretical foundations (Sec. 4).** We define function-valued *operator games* suitable for the Aumann–Shapley framework and show under which conditions the game and its values are well-defined for current neural operator architectures.
2. **Amortized grid-agnostic explainer (Sec. 5).** We provide a FastSHAP-like training procedure for Aumann–Shapley values, allowing us to train amortized explainers even when the samples are provided on irregular grids.
3. **Empirical validation (Sec. 6).** We validate our approach on a variety of partial differential equations (PDEs) and show that the resulting explainer produces attributions that are consistent with state-of-the-art discrete Shapley values and transfer across grid sizes without retraining.

2 Related work

Classical Shapley estimation. SHAP [23] introduced Shapley-value explanations to machine learning, while Castro et al. [4] proposed one of the earliest permutation-based approximation methods. Subsequent work has developed model-agnostic estimators based on Monte Carlo sampling [4, 36, 16], weighted least-squares objectives [13, 12, 29], and surrogate modeling of the value function [38, 3]. Complementary model-specific approaches exploit structure in trees [24, 40, 41], kernel methods [26], and k-nearest-neighbor algorithms [37]. Related work extends these ideas from single-feature attributions to interaction [8] or group attributions, including Monte Carlo, regression-based, tree-specific, and graph-neural-network-specific estimators [9, 17, 10, 27, 28]. OperatorSHAP belongs to this model-agnostic approximation literature, but adapts Shapley-value estimation to explanation problems in dynamical systems. We build on Shapley values, as they are widely used and satisfy desirable axioms, but other methods could give similar explanations (see Fumagalli et al. [11] for a recent overview).

Amortized SHAP. FastSHAP [15] introduced the idea of training a single explainer to produce Shapley values in one pass for finite-dimensional inputs. We extend the same loss to function-valued inputs by using a neural operator explainer. This parallels recent work [7, 5], which extends FastSHAP's amortized framework to Generalized Additive Models and Vision Transformers, but all of these previous approaches operate on finite-dimensional feature spaces and do not generalize across mesh resolutions or grid structures.

Integrated gradients and Aumann–Shapley. Integrated gradients (IG) [34] and its variants are differential explanations for finite-dimensional models. Their connection to Aumann–Shapley values

is well understood in the game-theory literature [1, 30]. Liu et al. [21] explored the connection of IG to Aumann–Shapley values in the machine learning context and Delicado and Pachón-García [6] proposed asymptotic Aumann–Shapley values for functional data. IG requires a differentiable model and moves all inputs jointly along a single straight path, so it assigns zero attribution to piecewise-constant outputs such as thresholded predictions [25]. None of these works investigates the connection in the context of neural operators, grid heterogeneity, or amortized explainers.

Neural Operators. Neural operators [18] are a class of models that learn solution operators of PDEs. Fourier neural operators (FNOs) [19] use Fourier transforms to learn the solution in frequency space, and GINO [20] is a graph-based architecture that can handle irregular meshes. Further variants include multi-wavelet neural operators [14], which use wavelet transforms instead of Fourier transforms or the Local FNO (LFNO) [22], which uses Fourier transforms on local patches.

3 Preliminaries

3.1 Attributions for finite players

Cooperative games. A *cooperative game* on a finite player set $N = \{1, \dots, n\}$ is a set-function $v : 2^N \to \mathbb{R}$ with $v(\emptyset) = 0$, where $v(S)$ is the worth that coalition $S \subseteq \{1, \dots, n\}$ produces when its members act jointly [33]. In the context of machine learning, players are input features to some model, and $v(S)$ measures how much of a model's output is recovered when the features in S are revealed and the rest of the features are set to some baseline (e.g., zero if applicable; various other baselines are used; see App. D.4). How to allocate the total worth $v(N)$ back to individual players has a unique answer under mild axioms.

Shapley values. That answer, due to Shapley [33], averages the marginal contribution $v(S \cup \{i\}) - v(S)$ of player i over all coalitions, weighted by the number of orderings in which each coalition appears:

$$\psi_i(v) \;=\; \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!\,(n - |S| - 1)!}{n!} \big(v(S \cup \{i\}) - v(S)\big). \tag{1}$$

This ψ_i is the unique allocation satisfying *efficiency* ($\sum_i \psi_i(v) = v(N)$), *symmetry* ($v(S \cup \{i\}) = v(S \cup \{j\})\ \forall S \subseteq N \setminus \{i, j\} \Rightarrow \psi_i(v) = \psi_j(v)$), the *null-player* property ($v(S \cup \{i\}) = v(S)\ \forall S \subseteq N \setminus \{i\} \Rightarrow \psi_i(v) = 0$), and *additivity* ($\psi(v + w) = \psi(v) + \psi(w)$). Due to the computational complexity, Shapley values are usually approximated by methods such as KernelSHAP [23] or, more recently, regression-adjusted Monte Carlo estimators (RegressionMSR) [38].

FastSHAP. Whereas Shapley values are calculated for each sample separately, FastSHAP [15] amortizes the computational cost of SHAP across inputs by training a single *explainer* $\Phi_\theta : \mathbb{R}^n \to \mathbb{R}^n$ that returns the entire Shapley vector for a given input x in one forward pass, using the weighted MSE objective

$$\mathcal{L}(\theta) \;=\; \mathbb{E}_{x,\, S \sim p_w}\big[\big(v(S) - \langle \Phi_\theta(x), \mathbf{1}_S \rangle\big)^2\big],$$

with efficiency $\langle \Phi_\theta(x), \mathbf{1}_N \rangle = v(N)$ imposed by hard normalization (Algorithm 1 of [15]). Here, v is the value of a game for input x, and p_w is a suitable distribution over coalitions. For finite-dimensional inputs, Φ_θ is typically an MLP, with more recent variants replacing it by a generalized additive model [7] or a vision transformer [5]. As our datasets often use irregular grids, none of these architectures can be used as explainers off the shelf.

3.2 The operator setting

Spatial PDEs and operator surrogates. We seek to explain surrogate models for PDEs (see App. A.1 for a short introduction). We consider PDE problems on a bounded spatial domain $\Omega \subset \mathbb{R}^d$ with prescribed boundary and initial conditions, where we denote $u : \Omega \times [0, \infty) \to \mathbb{R}$ as the state (not necessarily time-dependent). We consider well-posed problems that define a *solution operator* $\mathcal{F} : u_0 \mapsto u(\cdot, t^\star)$ that maps an input function (e.g., an initial condition, a coefficient field, or a forcing) to a solution function on Ω. Neural operators [18], and Fourier neural operators (FNOs) [19] in particular, learn $\mathcal{F}$ from data on a discrete grid of Ω, but can accept inputs on any discretization of

Ω without retraining. We write $\hat{\mathcal{F}}$ for such a learned surrogate of $\mathcal{F}$. We review the FNO architecture in App. A.4.

Classical Shapley attribution does not share this property: choosing a grid fixes the player set, halving the spacing doubles the number of players. Even with a fixed number of grid points, when the grid structure (i.e., the position of each grid point) changes between samples, their attributions are not comparable and an amortized explainer cannot easily learn them, even with a grid-agnostic model. We therefore need a different concept of spatial attributions that is independent of the underlying grid.

The Aumann–Shapley value. In PDE problems, we can view every spatial point (e.g., on $[0, 1]$) as a player contributing to the overall outcome of the PDE evolution. Aumann and Shapley derive an infinite-player analog of the Shapley value by working with a non-atomic measure space $(\Omega, \mathcal{C}, \lambda)$, where w.l.o.g. $\Omega = [0, 1]$ represents the player space, $\mathcal{C}$ the coalition space, and games are defined as maps $v : \mathcal{C} \to \mathbb{R}$ with $v(\emptyset) = 0$. A coalition $S \in \mathcal{C}$ is a measurable, possibly disconnected, spatial region and takes the role of a subset of players.[1] They define a continuous analog of the value of a game as a linear function that takes a game v as input and outputs a function $\varphi(v) : \mathcal{C} \to \mathbb{R}$ that satisfies $\varphi(v)(\Omega) = v(\Omega)$, $\varphi(v)(\emptyset) = 0$, and is finitely additive, i.e., $\varphi(v)(S \cup T) = \varphi(v)(S) + \varphi(v)(T)$ for disjoint $S, T \in \mathcal{C}$.

In contrast to the finite-player setting, they show that only certain classes of games admit a well-defined Aumann–Shapley value. The two classes we will use in this paper are **pNA** and **ASYMP** games. If a game v has a finite-dimensional representation $v = f \circ (\mu_1, \ldots, \mu_m)$ with $f \in C^1([0, 1]^m)$, $f(0) = 0$, and μ_i non-atomic probability measures on $(\Omega, \mathcal{C})$, then $v \in$ pNA [30, after Def. 1]. The value function φv is then given analytically by the *diagonal formula* [30]

$$(\varphi v)(S) \;=\; \int_0^1 \sum_{i=1}^{m} \mu_i(S)\, \partial_i f(t, \ldots, t)\, dt \qquad (S \in \mathcal{C}). \tag{2}$$

A game v belongs to the broader class **ASYMP** when its discrete Shapley values along *finite* sub-fields of $\mathcal{C}$ converge to a common limit, the *asymptotic value* (App. A.3). The inclusion $\mathrm{pNA} \subset \mathrm{ASYMP}$ lets us approximate the continuum value of our operator games by finite-grid Shapley values in Sec. 5. Fig. 2 shows the reverse of this. OperatorSHAP was trained to predict Aumann–Shapley values, and can thus be discretized to predict the Shapley values over finite games of various granularity without retraining.

Our goal will be to define a game that falls into the pNA class, such that we know it has a value we can learn. We especially want to ensure that common neural operator architectures fall into this class.

4 Theoretical foundations

We begin with a short intuition of the functional-analytic setting for readers unfamiliar with the topic. We work on function spaces, i.e., vector spaces, where each element is a function. Operators are maps between function spaces, i.e., functions that take functions as input and output other functions. We can define derivatives of these operators. The strictest one, the Fréchet derivative, is what is commonly understood as a "derivative", namely a linear map that approximates change in a neighborhood of a point. However, Fréchet differentiability is quite restrictive in function spaces and highly dependent on the input space we consider. We will therefore consider a well-behaved input space (the Sobolev space we define). Otherwise, even smooth activation functions, like the GELU, would not be differentiable as an operator in function space. Moreover, common function spaces do not distinguish between two functions that differ on a finite set of points, which makes point evaluation not well-defined. This is problematic for our purposes, as we want to attribute the output at a specific spatial point to the model input. However, in the Sobolev space we consider, every function has a continuous representative which we can use to define point evaluation. We state the results on the flat torus to avoid technicalities regarding boundary conditions. Only Proposition 6 depends on this choice, because the Fourier layers of the FNO act on periodic functions. App. B.3 extends it to bounded smooth domains, and all other results hold there unchanged.

[1] For readability, we omit some of the technical details, but review them in App. A.3. All of our proofs still use the full mathematical framework.

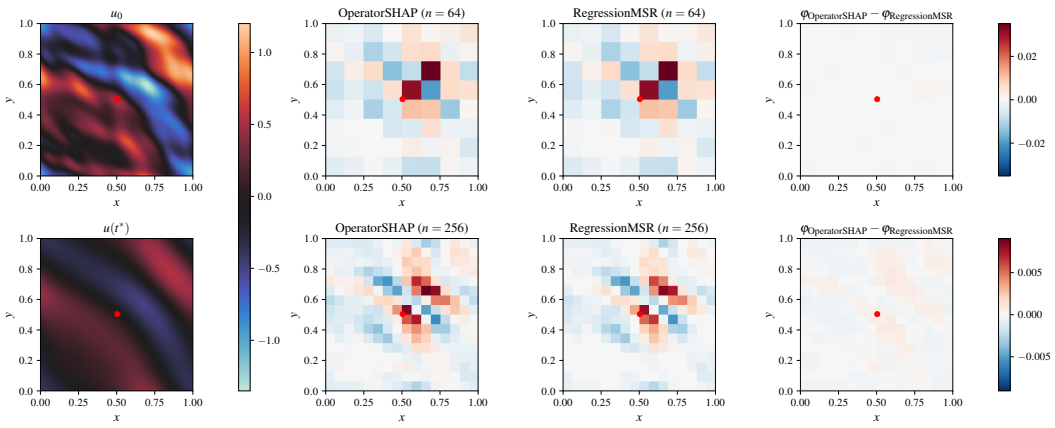


Figure 2: Explanations and input data of the 2D heat equation. The left column shows input (*top*) and label (*bottom*) of the backbone model. The right three columns show the explanation of OperatorSHAP, RegressionMSR, and their difference, respectively. The top row uses an 8×8 grid and the bottom row a 16×16 grid. The same OperatorSHAP model produces both rows, at comparable results to RegressionMSR.

4.1 Pointwise region attributions

Notation. Let $\Omega = \mathbb{T}^d$ be the flat torus and

$$X := H^s(\Omega; \mathbb{R}^{c_{\text{in}}}), \qquad Y := H^s(\Omega; \mathbb{R}^{c_{\text{out}}}), \qquad c_{\text{in}}, c_{\text{out}} \in \mathbb{N}, s > d/2,$$

where $H^s = W^{s,2}$ is the Sobolev space of functions with s weak derivatives in L^2. We consider a surrogate $\hat{\mathcal{F}} : X \to Y$, a baseline function $\bar{u} \in X$, an input function $u_0 \in X$, and a query point $x \in \Omega$ (write $h := u_0 - \bar{u}$). We want to attribute the scalar change $\hat{\mathcal{F}}(u_0)(x) - \hat{\mathcal{F}}(\bar{u})(x)$ to spatial regions of the input. We will write $D\hat{\mathcal{F}}(u)[\cdot] : X \to Y$ for the Fréchet derivative of $\hat{\mathcal{F}}$ (which itself is an operator).

Scalarizing the output. Because $s > d/2$, the Sobolev embedding theorem ([32], §2.2.5) yields a continuous embedding $H^s(\Omega; \mathbb{R}^{c_{\text{out}}}) \hookrightarrow C^0(\Omega; \mathbb{R}^{c_{\text{out}}})$ into the space of continuous functions. In particular, for each $x \in \Omega$, point evaluation $\text{ev}_x : C^0(\Omega; \mathbb{R}^{c_{\text{out}}}) \to \mathbb{R}^{c_{\text{out}}}$, $\text{ev}_x(y) = y(x)$, is a bounded linear map, and it is natural to study the functional $\hat{\mathcal{F}}_x := \text{ev}_x \circ \hat{\mathcal{F}}$, which evaluates the output of our operator at a point x (we assume a single output channel, for multiple output channels see App. B.6).

We will now define the game for which we want Shapley attributions, as the difference in prediction value of our neural operator $\hat{\mathcal{F}}$ when it receives a baseline input or an input that was activated in a region S (i.e., that is not baseline in S). As the input to $\hat{\mathcal{F}}$ needs to be smooth, we activate the input in a smooth way via helper functions η_i.

Definition 1 (Smooth operator games)**.** *For $i \in \{1, \dots, m\}$, define the smooth masks $\eta_i : \Omega \to [0, 1]$, such that $\sum_i \eta_i \equiv 1$, $\int_\Omega \eta_i d\lambda > 0$ for all $i \in \{1, \dots, m\}$. Let $\hat{\mathcal{F}}_x : X \to \mathbb{R}$ be a continuously Fréchet differentiable operator evaluated at a point x. Then we define the smooth operator game $v : \mathcal{C} \to \mathbb{R}$ by*

$$v(S) := \hat{\mathcal{F}}_x\left(\bar{u} + \sum_{i=1}^{m} \mu_i(S)\, \eta_i h\right) - \hat{\mathcal{F}}_x(\bar{u}), \quad \text{where } \mu_i(S) := \frac{\int_S \eta_i \, d\lambda}{\int_\Omega \eta_i \, d\lambda}.$$

The game depends on x, $\bar{u}$, and u_0. We suppress this dependence in the notation.

We may think of the η_i as smooth bump functions equally distributed on Ω (see Fig. 3, with an exemplary construction in App. B.5). For a coalition $S \in \mathcal{C}$ the game measures the change in the output at x, when each mask η_i is scaled by the fraction of its mass that lies in S.

Theorem 2 (The smooth operator game is pNA)**.** *Under the assumptions of Def. 1, the smooth operator game v lies in pNA, and admits a unique Aumann–Shapley value φv given by the diagonal formula [30, Thm. 2].*

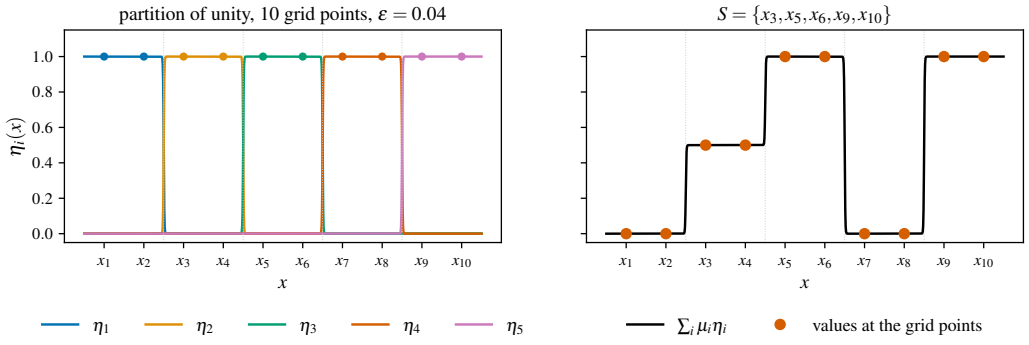


Figure 3: Partition of unity with five masks on a grid with two points per mask (construction in App. B.5). *Left:* the masks η_i, each equal to one at both grid points of its cell. *Right:* the smooth mask $\sum_i \mu_i \eta_i$ of the coalition $S = \{x_3, x_5, x_6, x_9, x_{10}\}$, which holds one of the two grid points of η_2, so $\mu = (0, 1/2, 1, 0, 1)$.

Proof sketch. v factorizes as $v = f_x \circ (\mu_1, \ldots, \mu_m)$ with $f_x(a) := \hat{\mathcal{F}}_x\big(\bar{u} + \sum_i a_i \eta_i h\big) - \hat{\mathcal{F}}_x(\bar{u})$, a C^1 map on $[0,1]^m$ with $f_x(0) = 0$, and the μ_i non-atomic probability measures on $(\Omega, \mathcal{C})$. This is the canonical pNA representation [30, after Def. 1], and the value claim follows from [30, Thm. 2]. The full proof is in App. B.1. □

Remark 3. *In the implementation (Alg. 1) we will use hard binary masks on the grid points, instead of the smooth masks η_i. This is not a deviation from the theory, but rather an effect of the discrete nature of neural operators. In practice, neural operators observe u_0 only through its values on a grid $\{x_1, \ldots, x_p\}$. Choosing a partition of unity in which the support of each η_i contains only the grid point x_i (App. B.5) gives $\eta_i(x_j) = \delta_{ij}$, which **appears** as a hard binary mask, but **still satisfies the theoretical requirements**. The coalition weights $\mu_i(S)$ then differ from 0 or 1 by an amount set by the transition width of the η_i, which can be chosen below machine precision.*

Remark 4. *The smoothness assumption concerns the surrogate $\hat{\mathcal{F}}$, not the underlying PDE. Most neural operators smooth out shocks or singularities of the true solution. Thus, while the true solution may not be accurately represented by the model, the attribution of the model's output is still well-defined and can be used to understand the model's behavior. On the other hand, our game does not need to satisfy boundary conditions or similar, as it explains only the surrogate model, not an underlying PDE.*

Remark 5. *The point evaluation ev_x can be replaced by any differentiable functional of the output field, e.g., a spatial average or an integral over a patch, by the chain rule. For inputs with several channels, a coalition masks all channels of a cell together and the density sums their contributions. Channels can also be treated as separate player sets and masked independently.*

4.2 Common neural operator architectures satisfy this

The Fréchet C^1 assumption underlying Def. 1 and Thm. 2 is met by the standard Fourier neural operator architecture [19], provided we use a smooth activation as shown in the following proposition.

Proposition 6 (Smooth FNOs are Fréchet C^1)**.** *For $s > d/2$ and $\sigma \in C^\infty(\mathbb{R})$ with $\sigma(0) = 0$, the FNO of App. A.4 defines a Fréchet C^1 map $\hat{\mathcal{F}} : H^s(\Omega; \mathbb{R}^{c_{\text{in}}}) \to H^s(\Omega; \mathbb{R}^{c_{\text{out}}})$.*

Proof sketch: All affine blocks are bounded linear maps on H^s, and the nonlinearity is a smooth Nemytskij superposition operator, which is Fréchet C^1 on H^s for $s > d/2$. The composition of these blocks is therefore also Fréchet C^1. The full proof is in App. B.2. □

Remark 7. *The same template covers most modern neural-operator architectures whose building blocks are either bounded linear maps on H^s or smooth Nemytskij superpositions. We find empirically that the framework is quite robust if the activation is not smooth (e.g., the ReLU) (App. D.5). Future work could investigate this theoretically.*

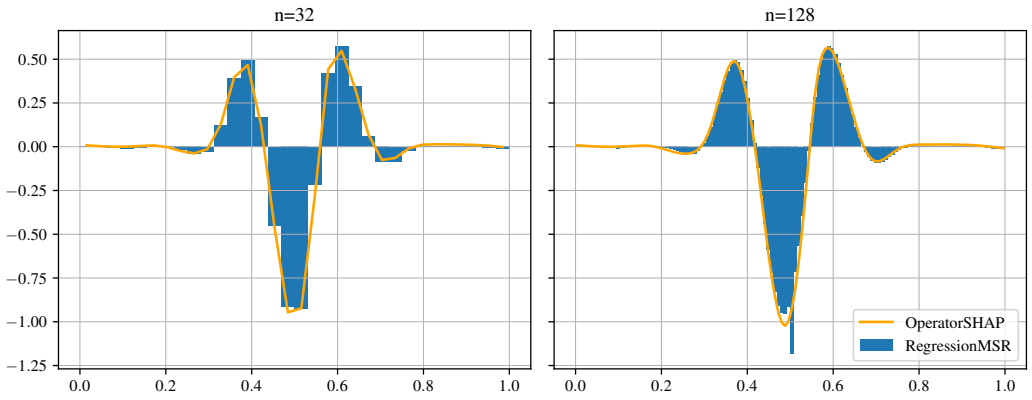


Figure 4: Comparison of classical Shapley values (blue bars) and Aumann–Shapley density (orange line) for player numbers (grid sizes) 32 (*left*) and 128 (*right*) for the 1D Burgers equation, approximated via RegressionMSR (for Shapley) and OperatorSHAP (Aumann–Shapley). In this figure, we scale Shapley values by the number of players to account for shrinking per-player attribution. We can see that with increasing player number, classical Shapley values approach the Aumann–Shapley density.

Training an Aumann–Shapley explainer. Since pNA $\subset$ ASYMP [30, Thm. 4], the game v above also admits an *asymptotic value*: along any refining sequence of finite coalition sets $\pi_0 \subset \pi_1 \subset \cdots$ with $\pi_k \subset \mathcal{C}$, the *discrete Shapley values* $\psi v_{\pi_k}(S)$ of the game restricted to π_k converge to $(\varphi v)(S)$ (App. A.3). This means that we can use the discrete Shapley values to train an approximator of the Aumann–Shapley values. Strictly, this statement is about one fixed set of masks η_i restricted to coarser partitions, whereas we define a separate set of masks at every resolution, which we find to work best in practice. Here, more granular samples will contribute to a more accurate explainer, but also less granular results can help to drive the overall convergence of the explainer, as we will see in the experiments. Fig. 4 shows how discrete Shapley values approximate Aumann–Shapley values when we account for the size of the region the Shapley values represent.

Remark 8. *The differentiability of f_x can be relaxed to bounded variation with continuity at 0 and at $\mu(\Omega)$ [30], which allows non-differentiable models and more interesting readouts (e.g., threshold games, App. D.6). The game then still has a value, but is not necessarily of class ASYMP anymore, so convergence has to be checked.*

4.3 Connection to Integrated Gradients

In this section, we show how the derived formula connects to the Integrated Gradients method.

Definition 9 (Masked pointwise attribution)**.** *Let $\hat{\mathcal{F}} : X \to Y$ be Fréchet C^1, fix $x \in \Omega$ and $\eta \in C^\infty(\Omega)$, and set $\hat{\mathcal{F}}_x = \mathrm{ev}_x \circ \hat{\mathcal{F}}$. For $\bar{u}, u_0 \in X$, with $h = u_0 - \bar{u}$, define the* masked attribution at output location x associated with η *by*

$$\mathrm{Attr}_x^\eta(\bar{u}, u_0) \;:=\; \int_0^1 D\hat{\mathcal{F}}_x(\bar{u} + th)[\eta h]\, dt.$$

If $\hat{\mathcal{F}}$ is vector-valued, the definition is understood componentwise.

The above definition equals precisely the integrated gradients formula [34] (review in App. A.2) but for function-valued games. Usually, we integrate the directional derivative of a vector-valued function, but here we integrate the directional derivative of a function-valued function. Choosing multiple smooth masks $\eta_1, \ldots, \eta_m$ that sum to one $\sum_{j=1}^m \eta_j \equiv 1$ allows us again to achieve efficiency.

Theorem 10 (Partition-of-unity attribution)**.** *Let $\hat{\mathcal{F}} : X \to Y$ be Fréchet C^1 and fix $x \in \Omega$. Let $\eta_1, \ldots, \eta_m \in C^\infty(\Omega)$ satisfy $\sum_{j=1}^m \eta_j \equiv 1$. Then, for every $\bar{u}, u_0 \in X$,*

$$\hat{\mathcal{F}}(u_0)(x) - \hat{\mathcal{F}}(\bar{u})(x) = \sum_{j=1}^m \mathrm{Attr}_x^{\eta_j}(\bar{u}, u_0).$$

Proof sketch: We use linearity to pull the sum into the integral and use $\sum_j \eta_j h = h$. The rest follows from the fundamental theorem of calculus. The full proof is in App. B.4. □

After substituting the partial derivative of f_x from the proof of Thm. 2, evaluated along the diagonal $a = (t, \dots, t)$, and using the identity $\sum_j \eta_j \equiv 1$ to collapse the inner sum, the diagonal formula (2) reduces to

$$(\varphi v)(S) \;=\; \int_0^1 \sum_{i=1}^{m} \mu_i(S) D\hat{\mathcal{F}}_x\Big(\bar{u} + t \sum_{j=1}^{m} \eta_j h\Big)[\eta_i h]\, dt \;=\; \sum_{i=1}^{m} \mu_i(S) \int_0^1 D\hat{\mathcal{F}}_x\Big(\bar{u} + th\Big)[\eta_i h]\, dt,$$

i.e., the Aumann–Shapley allocation to a coalition S is the $\mu(S)$-weighted sum of the masked pointwise attributions of Def. 9.

5 OperatorSHAP: an amortized neural operator explainer

The previous section showed that we can define a smooth operator game v for every query point $x \in \Omega$, with a well-defined Aumann–Shapley value. That value is a function from the coalition space to the reals $\varphi v : \mathcal{C} \to \mathbb{R}$, $S \mapsto (\varphi v)(S)$. We know this function is finitely additive, i.e., $(\varphi v)(S \cup T) = (\varphi v)(S) + (\varphi v)(T)$ for disjoint $S, T \in \mathcal{C}$, and that it satisfies the efficiency property $(\varphi v)(\Omega) = v(\Omega)$. We are now interested in training one amortized explainer Φ_θ that approximates this value function for all input functions u_0 and at all query positions x, i.e., $\Phi_\theta(u_0, x)(S) = \varphi v(S)$. Since pNA $\subset$ ASYMP [30, Thm. 4], the game v possesses an *asymptotic value*, so we can train a grid-agnostic explainer, like an FNO, on the discrete Shapley values at multiple resolutions. The query point is passed to the explainer as an additional input channel that marks its location on the input grid (a Gaussian bump of width 0.05 of the domain around $x^\star$), so the explainer remains grid-agnostic. We then get the attribution of a coalition S by integrating the explainer output over the area of S.

Algorithm 1: OperatorSHAP training

Input: backbone $\hat{\mathcal{F}}$, baseline $\bar{u}$, learning rate α.
Output: explainer Φ_θ.

initialize Φ_θ
while *not converged* **do**
  sample $u_0 \sim p(u_0)$, $x^\star \sim \mathcal{U}(\Omega)$, coalition $S \sim p_w$
  $\bar{v} \leftarrow v(\Omega) - v(\emptyset)$
  $\phi \leftarrow \Phi_\theta(u_0, x^\star)$ `// field on` Ω
  $B \leftarrow$ area(u_0, S) `// get area of` S
  $\hat{\phi}_S \leftarrow \int_B \phi$ `// coarsen`
  if *normalize* **then**
    $\hat{\phi}_S \leftarrow \hat{\phi}_S + (|B|/|\Omega|)\big(\bar{v} - \int_\Omega \phi\big)$
  $\mathcal{L} \leftarrow \big(v(S) - \hat{\phi}_S\big)^2$
  $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}$

Alg. 1 gives the resulting training loop. At each step we draw an input $u_0 \in \mathbb{R}^p$ (where p can be different for each sample), a query point $x^\star$ uniformly from the grid, and fix a grid of partition cells $B_1, \dots, B_p$ for that sample. We then evaluate the explainer to obtain a continuous attribution field $\phi = \Phi_\theta(u_0, x^\star)$, coarsen it to a discrete vector $\hat{\phi} \in \mathbb{R}^p$ by integrating over the partition cells B_i, and minimize the FastSHAP weighted MSE $(v(S) - \hat{\phi}^\top \mathbf{1}_S)^2$ for coalitions $S \subseteq \{1, \dots, p\}$ drawn from the Shapley kernel w_p.[2] Hard normalization [15, Alg. 1] is applied to enforce efficiency, $\sum_i \hat{\phi}_i = v(\Omega) - v(\emptyset)$. Fig. 4 visualizes the attributions.

6 Experiments

Setup. We use ten PDE datasets on structured 1D/2D/3D grids and the two MeshGraphNets benchmarks [31] (Tab. 2). We use FNO [19], LocalFNO [22], or GINO [20] (also MWT [14] for the analytic validation) as backbones that we explain (backbone errors in App. D.8). The OperatorSHAP explainer mirrors the backbone family and is trained on the FastSHAP loss of Sec. 5. To simulate heterogeneous grids on the structured PDEs, we stride each sample to a fixed grid granularity $n \in \{8, 16, 32, \dots, 1024\}$ ($n = 512$ in 3D). We compare against exact Shapley ($n \leq 16$),

[2] $w_p(S) \propto \frac{p-1}{\binom{p}{|S|}|S|(p-|S|)}$, the KernelSHAP weight [23], which puts most mass on very small or very large coalitions and is what makes the WLS objective unbiased for the Shapley value.

Table 1: Attribution quality across resolutions n for the point readout and the decision readout of App. D.6 on two PDEs and MGN Cylinder. Each (PDE, n, method) entry reports the median over 30 test inputs and three query points per input on the top line and the inter-quartile range in brackets below. Metrics are the Shapley-kernel weighted R^2 (faithfulness), and Pearson correlation and NRMSE against the reference attribution. RegressionMSR (RMSR) and KernelSHAP (KS) use a budget of $2n$ (minimum 1024) backbone evaluations per query point, IG uses a 50-step Riemann sum, and OperatorSHAP (OS) uses one query-conditioned model per PDE and readout at all resolutions. MGN Cylinder uses 10 test inputs. On the decision readout, the gradient of the game vanishes, so IG attributions are identically zero and their Pearson correlation is undefined (–). Bold marks the best median and every median within one standard error of it over the test inputs, compared at three decimals. All PDEs are in Tabs. 3, 4 and 8.

		R^2 ↑				Pearson ↑				NRMSE ↓			
PDE	n	RMSR	KS	IG	OS (ours)	RMSR	KS	IG	OS (ours)	RMSR	KS	IG	OS (ours)
Point readout													
Burgers 1D	64	0.924 [0.876, 0.978]	**0.957** [0.900, 0.991]	**0.956** [0.895, 0.991]	**0.952** [0.887, 0.987]	0.979 [0.968, 0.985]	**0.999** [0.995, 1.000]	**0.999** [0.997, 1.000]	0.995 [0.991, 0.997]	0.256 [0.197, 0.314]	**0.049** [0.016, 0.097]	**0.043** [0.018, 0.090]	0.100 [0.073, 0.158]
	256	0.944 [0.834, 0.979]	**0.971** [0.936, 0.996]	**0.971** [0.941, 0.997]	**0.966** [0.919, 0.991]	0.596 [0.373, 0.814]	0.946 [0.847, 0.997]	**0.997** [0.995, 0.998]	0.985 [0.971, 0.992]	0.805 [0.684, 0.964]	0.317 [0.122, 0.574]	**0.094** [0.064, 0.122]	0.195 [0.126, 0.263]
	1024	0.960 [0.891, 0.988]	-2.497 [-13.453, -0.042]	**0.985** [0.956, 0.998]	0.970 [0.937, 0.993]	0.240 [0.123, 0.554]	0.050 [0.023, 0.108]	**0.958** [0.940, 0.968]	0.900 [0.848, 0.930]	1.357 [0.865, 1.642]	28.083 [12.454, 53.720]	**0.317** [0.255, 0.481]	0.480 [0.363, 0.724]
Shallow Water 2D	64	0.499 [0.408, 0.600]	0.550 [0.456, 0.624]	0.421 [0.247, 0.476]	**0.570** [0.479, 0.644]	0.949 [0.940, 0.957]	0.970 [0.961, 0.976]	0.729 [0.620, 0.810]	**0.996** [0.995, 0.997]	0.416 [0.372, 0.440]	0.258 [0.226, 0.291]	0.731 [0.617, 0.855]	**0.093** [0.085, 0.106]
	256	0.803 [0.745, 0.874]	0.864 [0.824, 0.892]	0.869 [0.825, 0.897]	**0.881** [0.841, 0.910]	0.850 [0.834, 0.899]	0.927 [0.906, 0.936]	0.915 [0.892, 0.927]	**0.964** [0.951, 0.972]	0.575 [0.451, 0.606]	0.482 [0.451, 0.522]	0.455 [0.425, 0.497]	**0.311** [0.288, 0.364]
	1024	0.951 [0.929, 0.965]	-3.861 [-23.325, -0.627]	**0.980** [0.971, 0.986]	0.974 [0.964, 0.982]	0.910 [0.880, 0.932]	0.136 [0.050, 0.216]	0.898 [0.879, 0.921]	**0.923** [0.898, 0.939]	**0.416** [0.363, 0.477]	9.197 [5.292, 22.002]	0.529 [0.433, 0.637]	0.527 [0.454, 0.598]
MGN Cylinder	64	**0.921** [0.824, 0.963]	**0.923** [0.827, 0.963]	0.900 [0.775, 0.943]	**0.910** [0.783, 0.955]	0.985 [0.977, 0.992]	**0.991** [0.984, 0.993]	0.916 [0.830, 0.951]	0.946 [0.890, 0.983]	0.181 [0.138, 0.235]	**0.130** [0.116, 0.170]	0.415 [0.338, 0.558]	0.315 [0.185, 0.472]
Decision readout													
Burgers 1D	64	**0.765** [0.630, 0.811]	**0.764** [0.627, 0.799]	-1.120 [-1.393, -1.004]	**0.776** [0.648, 0.824]	0.884 [0.775, 0.946]	0.855 [0.748, 0.929]	–	**0.990** [0.983, 0.993]	0.449 [0.348, 0.580]	0.535 [0.412, 0.761]	1.000 [1.000, 1.000]	**0.160** [0.126, 0.214]
	256	**0.810** [0.721, 0.848]	0.702 [0.631, 0.728]	-1.113 [-1.280, -1.012]	**0.832** [0.745, 0.867]	0.178 [0.083, 0.446]	0.193 [0.093, 0.385]	–	**0.975** [0.964, 0.983]	1.221 [0.889, 1.499]	3.391 [2.121, 4.410]	1.000 [1.000, 1.000]	**0.232** [0.182, 0.342]
	1024	**0.858** [0.797, 0.882]	-86.894 [-310.480, -30.642]	-1.096 [-1.221, -1.016]	**0.865** [0.807, 0.893]	0.038 [0.009, 0.129]	0.006 [-0.018, 0.029]	–	**0.831** [0.779, 0.869]	1.828 [1.275, 2.228]	173.467 [82.409, 316.716]	1.000 [1.000, 1.000]	**0.564** [0.493, 0.630]
Shallow Water 2D	64	0.518 [0.474, 0.668]	0.525 [0.479, 0.670]	-0.838 [-0.938, -0.764]	**0.551** [0.508, 0.694]	0.912 [0.892, 0.934]	0.917 [0.904, 0.932]	–	**0.992** [0.990, 0.994]	0.442 [0.383, 0.480]	0.429 [0.396, 0.463]	1.000 [1.000, 1.000]	**0.125** [0.107, 0.157]
	256	0.756 [0.717, 0.815]	0.645 [0.591, 0.699]	-0.934 [-1.012, -0.870]	**0.811** [0.785, 0.833]	0.728 [0.660, 0.869]	0.539 [0.493, 0.572]	–	**0.978** [0.971, 0.984]	0.694 [0.505, 0.763]	1.394 [1.301, 1.509]	1.000 [1.000, 1.000]	**0.219** [0.190, 0.266]
	1024	0.871 [0.855, 0.878]	-66.535 [-298.438, -26.911]	-0.949 [-0.998, -0.903]	**0.887** [0.878, 0.892]	0.909 [0.875, 0.935]	0.010 [-0.006, 0.039]	–	**0.964** [0.943, 0.974]	0.428 [0.367, 0.493]	34.301 [23.284, 82.147]	1.000 [1.000, 1.000]	**0.333** [0.267, 0.389]
MGN Cylinder	64	**0.799** [0.735, 0.830]	**0.789** [0.732, 0.812]	-1.066 [-1.196, -0.948]	**0.794** [0.705, 0.824]	**0.957** [0.909, 0.976]	0.900 [0.818, 0.943]	–	0.928 [0.774, 0.977]	**0.289** [0.218, 0.406]	0.465 [0.366, 0.619]	1.000 [1.000, 1.000]	0.402 [0.213, 0.645]

KernelSHAP [23], RegressionMSR [38], and Integrated Gradients (IG, Sec. 4.3). KernelSHAP and RegressionMSR get the same budget of $2n$ (at least 1024) backbone evaluations per query point. We mostly use the zero baseline, because it is the easiest to interpret and allows comparisons across samples. In some cases, however, it introduces physically implausible artifacts that hinder the backbone prediction. Other baselines and an ablation are discussed in App. D.4. On the structured grids, every test input is explained at three query points, the domain center and two off-center points with coordinates at a quarter and three quarters of the domain, and on the meshes at three fixed points of the flow region. Metrics are Pearson correlation and normalized RMSE against the reference (RegressionMSR with $3{\cdot}10^5$ backbone evaluations, $5{\cdot}10^4$ on MGN Airfoil), plus faithfulness [35] (weighted R^2 over 10,000 random coalitions) as a reference-free diagnostic. We report medians over test inputs and query points. We also test preservation of spatial structure using the structural similarity index (SSIM) at two resolutions (App. D.7). The full setup is in App. C and the hardware in App. E. We test for:

(i) Attribution quality across resolutions. We evaluate two games: (i) classical point prediction, where an attribution tells us what region influenced a point; (ii) thresholded outcome, which equals one when the prediction at a point is closer to its value at u_0 than to its value at the baseline and zero otherwise. This represents decision problems, where the precise change in value is less relevant than the general behavior of the PDE (App. D.6). The same trained explainer produces attributions at all resolutions, while the classical methods have to be rerun at each resolution. Tab. 1 reports a subset of the results, with the full metrics in Tabs. 3, 4 and 8. OperatorSHAP is competitive with the Shapley approximators and IG on most PDEs, while being faster on inference. On the non-differentiable decision tasks, IG is unable to perform meaningful predictions. We additionally ablate held-out resolutions (App. D.2).

(ii) Inference speed. Once trained, OperatorSHAP returns full attributions in a single explainer forward pass and two backbone forward passes for the hard normalization. Fig. 8 in App. D.1 compares

the runtime and faithfulness of all methods, and Fig. 6 shows the NRMSE of RegressionMSR and KernelSHAP vs. their budget. OperatorSHAP is orders of magnitude faster than RegressionMSR and KernelSHAP, while maintaining a low error. IG needs 1.5 to 1.9 times the runtime of OperatorSHAP where it reaches its faithfulness, and on Reaction–Diffusion 2D and Shallow Water 2D it does not reach it at any number of path steps. Again, the same trained explainer is used across resolutions, while baselines needed to be rerun at each resolution.

(iii) Amortization time. Training time of OperatorSHAP is costly. To illustrate when amortization pays off, we show the cumulative number of model evaluations of OperatorSHAP vs. baselines as the number of inference samples increases in Fig. 7. Amortization pays off after $1.1 \cdot 10^4$ to $3.7 \cdot 10^5$ explanations depending on the PDE and the classical method. This is the regime of a deployed in-production surrogate that returns an explanation with every prediction. There, the latency of one explanation matters more than the one-time training cost.

(iv) Analytic validation. For linear PDEs, the true solution operator has a closed-form Aumann–Shapley density, which validates the explainer without a discrete estimator in the loop. For the heat equation, it is an isotropic bump around the evaluation point, scaled by the initial condition. For the advection equation, it is a point mass at the characteristic foot point $x^\star - \beta t^\star$. Fig. 9 in App. D.3 shows that the FNO explainer matches the heat kernel density. For advection, both explainers correctly place mass at the advection foot point. The largest value of the FNO explainer, however, is a spike at $x^\star$ itself. This spike is a property of the backbone, whose pointwise skip connections pass $u_0(x^\star)$ directly to the output, and illustrates how attributions expose failure modes of a surrogate.

7 Limitations

While amortized Shapley can be incredibly useful when needing explanations at scale, few people outside the industry will need that many explanations without changing the model in between. For companies that consistently need to monitor their AI application or want to give their users an explanation along the prediction, amortization can be very time-saving, however. Furthermore, PDE surrogate models often need high spatial resolution to make an accurate prediction. As in classical Shapley, however, more input features exponentially complicate accurate attributions, necessitating binning approaches that could obscure fine attribution spikes. Still, modern approaches to (amortized) SHAP have started to mitigate the computational burden while retaining accurate attributions. On the theory side, while our restriction to smooth masks is irrelevant for finite-dimensional data (App. B.5) and differentiable operators can be robustly relaxed in practice (App. D.5), our assumptions are nonetheless restrictions that could potentially be relaxed in future work by deeper functional analysis concepts or other definitions of games. In our experiments, it should be noted that RMSR could be arbitrarily improved by increasing the budget. Where gradients of the backbone are available, Integrated Gradients reaches a similar performance on most of our PDEs (Tabs. 3 and 4), because their games are close to additive. The advantage of OperatorSHAP is then the lower cost per explanation and that it only needs forward passes of the backbone. On non-differentiable games, Integrated Gradients failed while OperatorSHAP still performed well. Lastly, we want to mention the risk that may arise when models produce false explanations. We are still working with black-box models whose output should not be trusted blindly when the result could be harmful in any way.

8 Conclusion

We have shown that Shapley attribution can be amortized for neural operators in a grid-agnostic way by training a neural operator explainer on a FastSHAP-like objective and connecting the resulting discrete attributions to the Aumann–Shapley value in the continuous limit. This allows direct evaluation of Shapley values on arbitrary grids and, in particular, training on datasets with heterogeneous grids. For future work, we see three directions to explore. First, there is a lot of pre-existing work in classical Shapley estimation that could be applied to OperatorSHAP. For example, certain engineering tricks that focus the effort on high frequency regions could allow for higher player sizes with reduced computational effort. Second, our framework only allows for additive attributions, while interactions between regions are ignored. An extension to Shapley interactions would be interesting. Lastly, our theoretical assumptions may be unnecessarily restricting. One could explore how to integrate non-differentiable activations into the theory or even allow for discontinuous backbones.

Acknowledgments and Disclosure of Funding

JS acknowledges funding from the Munich Center for Machine Learning (MCML), as well as the support he received from the METEOR Research Training Group, which is sponsored by the German Research Foundation. FC gratefully acknowledges funding by the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) – GRK 3081 – Project number 534429653. The authors gratefully acknowledge the computational and data resources provided by the Leibniz Supercomputing Centre (www.lrz.de). We also thank Konstantinos Kotsakis for his support in keeping our jobs running on the cluster. The authors declare no competing interests.

References

[1] Robert J. Aumann and Lloyd S. Shapley. *Values of non-atomic games*. Princeton University Press, December 2015. ISBN 978-1-4008-6708-0. doi: 10.1515/9781400867080. URL https://www.degruyterbrill.com/document/doi/10.1515/9781400867080/html.

[2] Lukas Biewald. Experiment tracking with weights and biases, 2020. URL https://www.wandb.com/.

[3] Landon Butler, Abhineet Agarwal, Justin Kang, Yigit Efe Erginbas, Bin Yu, and Kannan Ramchandran. ProxySPEX: inference-efficient interpretability via sparse feature interactions in LLMs. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen, editors, *Advances in Neural Information Processing Systems*, volume 38. Curran Associates, Inc., 2025. doi: 10.52202/085713-2426. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/68dced1638d71d4d21598f79ae91262e-Paper-Conference.pdf.

[4] Javier Castro, Daniel Gómez, and Juan Tejada. Polynomial calculation of the Shapley value based on sampling. *Computers & Operations Research*, 36(5):1726–1730, 2009. ISSN 0305-0548. doi: 10.1016/j.cor.2008.04.004. URL https://www.sciencedirect.com/science/article/pii/S0305054808000804.

[5] Ian Connick Covert, Chanwoo Kim, and Su-In Lee. Learning to estimate Shapley values with vision transformers. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=5ktFNz_pJLK.

[6] Pedro Delicado and Cristian Pachón-García. Functional relevance based on the continuous Shapley value, June 2025. URL http://arxiv.org/abs/2411.18575. arXiv:2411.18575 [stat.ML].

[7] James Enouen and Yan Liu. InstaSHAP: interpretable additive models explain Shapley values instantly. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=ky7vVlBQBY.

[8] Katsushige Fujimoto, Ivan Kojadinovic, and Jean-Luc Marichal. Axiomatic characterizations of probabilistic and cardinal-probabilistic interaction indices. *Games and Economic Behavior*, 55(1):72–99, April 2006. ISSN 08998256. doi: 10.1016/j.geb.2005.03.002. URL https://linkinghub.elsevier.com/retrieve/pii/S0899825605000278.

[9] Fabian Fumagalli, Maximilian Muschalik, Patrick Kolpaczki, Eyke Hüllermeier, and Barbara Hammer. SHAP-IQ: unified approximation of any-order Shapley interactions. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 11515–11551. Curran Associates, Inc., 2023. doi: 10.52202/075280-0508. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/264f2e10479c9370972847e96107db7f-Paper-Conference.pdf.

[10] Fabian Fumagalli, Maximilian Muschalik, Patrick Kolpaczki, Eyke Hüllermeier, and Barbara Hammer. KernelSHAP-IQ: weighted least square optimization for Shapley interactions. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 14308–14342. PMLR, July 2024. URL https://proceedings.mlr.press/v235/fumagalli24a.html.

[11] Fabian Fumagalli, Maximilian Muschalik, Eyke Hüllermeier, Barbara Hammer, and Julia Herbinger. Unifying feature-based explanations with functional ANOVA and cooperative game theory. In Yingzhen Li, Stephan Mandt, Shipra Agrawal, and Emtiyaz Khan, editors, *Proceedings of The 28th International Conference on Artificial Intelligence and Statistics*, volume 258 of *Proceedings of Machine Learning Research*, pages 5140–5148. PMLR, May 2025. URL https://proceedings.mlr.press/v258/fumagalli25a.html.

[12] Fabian Fumagalli, Landon Butler, Justin Singh Kang, Kannan Ramchandran, and R. Teal Witter. An odd estimator for Shapley values. In Tong Zhang, Miroslav Dudik, Martin Jaggi, Alekh Agarwal, Sharon Li, Dale Schuurmans, Jerry Zhu, Felix Berkenkamp, Hanze Dong, and Alberto Bietti, editors, *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, pages 32341–32365. PMLR, July 2026. URL https://proceedings.mlr.press/v306/fumagalli26a.html.

[13] Fabian Fumagalli, R. Teal Witter, and Christopher Musco. PolySHAP: extending KernelSHAP with interaction-informed polynomial regression. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=M19J8UGguq.

[14] Gaurav Gupta, Xiongye Xiao, and Paul Bogdan. Multiwavelet-based operator learning for differential equations. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 24048–24062. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/c9e5c2b59d98488fe1070e744041ea0e-Paper.pdf.

[15] Neil Jethani, Mukund Sudarshan, Ian Connick Covert, Su-In Lee, and Rajesh Ranganath. FastSHAP: real-time Shapley value estimation. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=Zq2G_VTV53T.

[16] Patrick Kolpaczki, Viktor Bengs, Maximilian Muschalik, and Eyke Hüllermeier. Approximating the Shapley value without marginal contributions. In *Thirty-eighth AAAI conference on artificial intelligence*, pages 13246–13255, 2024. doi: 10.1609/AAAI.V38I12.29225. URL https://doi.org/10.1609/aaai.v38i12.29225.

[17] Patrick Kolpaczki, Maximilian Muschalik, Fabian Fumagalli, Barbara Hammer, and Eyke Hüllermeier. SVARM-IQ: efficient approximation of any-order Shapley interactions through stratification. In Sanjoy Dasgupta, Stephan Mandt, and Yingzhen Li, editors, *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pages 3520–3528. PMLR, May 2024. URL https://proceedings.mlr.press/v238/kolpaczki24a.html.

[18] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: learning maps between function spaces with applications to PDEs. *Journal of Machine Learning Research*, 24(89):1–97, 2023. URL http://jmlr.org/papers/v24/21-1524.html.

[19] Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=c8P9NQVtmnO.

[20] Zongyi Li, Nikola Kovachki, Chris Choy, Boyi Li, Jean Kossaifi, Shourya Otta, Mohammad Amin Nabian, Maximilian Stadler, Christian Hundt, Kamyar Azizzadenesheli, and Animashree Anandkumar. Geometry-informed neural operator for large-scale 3D PDEs. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 35836–35854. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/70518ea42831f02afc3a2828993935ad-Paper-Conference.pdf.

[21] Shuyang Liu, Zixuan Chen, Ge Shi, Ji Wang, Changjie Fan, Yu Xiong, Runze Wu, Yujing Hu, Ze Ji, and Yang Gao. A new baseline assumption of integated gradients based on shaply value, May 2024. URL http://arxiv.org/abs/2310.04821. arXiv:2310.04821 [cs].

[22] Miguel Liu-Schiaffini, Julius Berner, Boris Bonev, Thorsten Kurth, Kamyar Azizzadenesheli, and Anima Anandkumar. Neural operators with localized integral and differential kernels. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 32576–32594. PMLR, July 2024. URL https://proceedings.mlr.press/v235/liu-schiaffini24a.html.

[23] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 4765–4774, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html.

[24] Scott M. Lundberg, Gabriel G. Erion, Hugh Chen, Alex J. DeGrave, Jordan M. Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. From local explanations to global understanding with explainable AI for trees. *Nature Machine Intelligence*, 2(1):56–67, 2020. doi: 10.1038/s42256-019-0138-9.

[25] John Merrill, Geoff Ward, Sean Kamkar, Jay Budzik, and Douglas Merrill. Generalized integrated gradients: a practical method for explaining diverse ensembles, September 2019. URL http://arxiv.org/abs/1909.01869. arXiv:1909.01869 [cs.LG].

[26] Majid Mohammadi, Siu Lun Chau, and Krikamol Muandet. Amortized linear-time exact Shapley value for product-kernel methods, May 2026. URL http://arxiv.org/abs/2505.16516. arXiv:2505.16516 [cs.LG].

[27] Maximilian Muschalik, Fabian Fumagalli, Barbara Hammer, and Eyke Hüllermeier. Beyond TreeSHAP: efficient computation of any-order Shapley interactions for tree ensembles. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 14388–14396, 2024. doi: 10.1609/aaai.v38i13.29352.

[28] Maximilian Muschalik, Fabian Fumagalli, Paolo Frazzetto, Janine Strotherm, Luca Hermes, Alessandro Sperduti, Eyke Hüllermeier, and Barbara E Hammer. Exact computation of any-order Shapley interactions for graph neural networks. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu, editors, *International Conference on Learning Representations*, pages 36295–36337, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/59b9582cd35f555ea8415030073e7b22-Paper-Conference.pdf.

[29] Christopher Musco and R. Teal Witter. Provably accurate Shapley value estimation via leverage score sampling. In *The Thirteenth International Conference on Learning Representations*. OpenReview.net, 2025. URL https://openreview.net/forum?id=wg3rBImn3O.

[30] Abraham Neyman. Values of games with infinitely many players. In *Handbook of Game Theory with Economic Applications*, volume 3, pages 2121–2167. Elsevier, 2002. ISBN 978-0-444-89428-1. doi: 10.1016/S1574-0005(02)03019-9. URL https://linkinghub.elsevier.com/retrieve/pii/S1574000502030199.

[31] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=roNqYL0_XP.

[32] Thomas Runst and Winfried Sickel. *Sobolev spaces of fractional order, Nemytskij operators, and nonlinear partial differential equations*. Number 3 in De Gruyter series in nonlinear analysis and applications. Walter de Gruyter, Berlin ; New York, 1996. ISBN 978-3-11-015113-8.

[33] Lloyd S. Shapley. A value for n-person games. In Alvin E. Roth, editor, *The Shapley Value*, pages 31–40. Cambridge University Press, October 1988. doi: 10.1017/CBO9780511528446.003. URL https://www.cambridge.org/core/product/identifier/CBO9780511528446A008/type/book_part.

[34] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3319–3328. PMLR, August 2017. URL https://proceedings.mlr.press/v70/sundararajan17a.html.

[35] Che-Ping Tsai, Chih-Kuan Yeh, and Pradeep Ravikumar. Faith-shap: The faithful Shapley interaction index. *Journal of Machine Learning Research*, 24:94:1–94:42, 2023. URL https://jmlr.org/papers/v24/22-0202.html.

[36] Jiachen T. Wang and Ruoxi Jia. Data Banzhaf: a robust data valuation framework for machine learning. In Francisco Ruiz, Jennifer Dy, and Jan-Willem van de Meent, editors, *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pages 6388–6421. PMLR, April 2023. URL https://proceedings.mlr.press/v206/wang23e.html.

[37] Jiachen T. Wang, Prateek Mittal, and Ruoxi Jia. Efficient data Shapley for weighted nearest neighbor algorithms. In Sanjoy Dasgupta, Stephan Mandt, and Yingzhen Li, editors, *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pages 2557–2565. PMLR, May 2024. URL https://proceedings.mlr.press/v238/wang24e.html.

[38] R. Teal Witter, Yurong Liu, and Christopher Musco. Regression-adjusted Monte Carlo estimators for Shapley values and probabilistic values. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen, editors, *Advances in Neural Information Processing Systems*, volume 38. Curran Associates, Inc., 2025. doi: 10.52202/085713-0715. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/1e77af93008ee6cd248a31723ce357d8-Paper-Conference.pdf.

[39] Omry Yadan. Hydra - A framework for elegantly configuring complex applications, 2019. URL https://github.com/facebookresearch/hydra.

[40] Peng Yu, Albert Bifet, Jesse Read, and Chao Xu. Linear TreeShap. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 25818–25828. Curran Associates, Inc., 2022. doi: 10.52202/068431-1872. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/a5a3b1ef79520b7cd122d888673a3ebc-Paper-Conference.pdf.

[41] Artjom Zern, Klaus Broelemann, and Gjergji Kasneci. Interventional SHAP values and interaction values for piecewise linear regression trees. In *Proceedings of the AAAI conference on artificial intelligence*, pages 11164–11173, 2023.

A Additional background information

A.1 A short primer on PDEs and solution operators

Many physical systems are described by *partial differential equations* (PDEs): the temperature in a metal bar, the velocity of a fluid in a pipe, the air pressure around an aircraft wing. A PDE is specified by a local rule—how a quantity at a specific location changes over time—that, together with a defined initial state and conditions at the boundary of the spatial region, determines the system's entire evolution.

The heat equation. An easy example is the one-dimensional heat equation. Imagine a thin rod of unit length, and let $u(x,t)$ denote its temperature at position $x \in [0,1]$ and time $t \geq 0$. If we hold both ends of the rod at temperature zero and start from some initial profile $u_0(x)$, the temperature evolves according to

$$\partial_t u(x,t) \;=\; \kappa\, \partial_x^2 u(x,t), \qquad u(0,t) = u(1,t) = 0, \qquad u(\cdot,0) = u_0,$$

where $\kappa > 0$ is a material constant determining how fast heat spreads. In plain words: at each point, the rate at which the temperature changes over time is proportional to how curved the temperature profile is at that point. If a point's surrounding is hotter than itself, it will heat up. If the surrounding is cooler, it will cool down. Due to the boundary conditions, the temperature everywhere will eventually cool down to 0.

The solution operator. We are interested in modeling the temperature $u(x,t)$, when starting from initial temperature u_0 and letting the system run for a fixed time $t^\star$. The map that turns the initial profile into the final one,

$$\mathcal{F} :\; u_0 \;\longmapsto\; u(\cdot, t^\star),$$

is the *solution operator*. It takes a function as input and returns a function as output. Neural operators are parametric models that learn $\mathcal{F}$ from data. In essence, they are just neural networks that can handle arbitrary input and output discretizations.

A.2 Integrated gradients

The differential analog of Shapley attribution in finite dimensions is integrated gradients [34]: for a continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ with baseline x_0, the contribution of coordinate i to $f(x) - f(x_0)$ is the path integral of its partial derivative,

$$\mathrm{IG}_i(x_0, x) \;=\; (x_i - x_{0,i}) \int_0^1 \partial_i f\big(x_0 + t(x - x_0)\big)\, dt,$$

which satisfies sensitivity, and *completeness* $\sum_i \mathrm{IG}_i(x_0,x) = f(x) - f(x_0)$.

A.3 The Aumann–Shapley value: mathematical details

This appendix records the formal definitions used in Sec. 4. Our exposition follows [30], to which we refer for proofs and historical context.

Spaces of games. The space of players is a measurable space $(\Omega, \mathcal{C})$ that we assume isomorphic to $([0,1], \mathcal{B}([0,1]))$, with $\mathcal{B}$ the Borel σ-algebra. Members of Ω are *players*, members of $\mathcal{C}$ are *coalitions*, and a *game* is a real-valued function $v : \mathcal{C} \to \mathbb{R}$ with $v(\emptyset) = 0$. A game is *monotonic* if $v(S) \geq v(T)$ whenever $S \supseteq T$, and of *bounded variation* if it is the difference of two monotonic games. Its *variation* is

$$\|v\| \;:=\; \sup\Big\{ \textstyle\sum_{j=1}^{n} |v(S_j) - v(S_{j-1})| \;:\; S_0 \subseteq S_1 \subseteq \cdots \subseteq S_n \text{ in } \mathcal{C} \Big\}.$$

Under $\|\cdot\|$ the space BV of all bounded-variation games is a Banach algebra. Within BV live the finitely additive games $FA = \{v : v(S \cup T) = v(S) + v(T) \text{ for disjoint } S, T\}$, the countably additive measures $M \subset FA$, and the non-atomic measures $NA \subset M$, i.e. measures $\mu \in M$ for which every singleton in $\mathcal{C}$ has μ-measure zero. Each of NA, M, FA, BV is invariant under the action $(\Theta_* v)(S) := v(\Theta S)$ of the group $\mathcal{G}$ of measurable automorphisms of $(\Omega, \mathcal{C})$. We write Q^+ for the monotonic elements of a set of games Q, and Q^1 for those with $v(\Omega) = 1$.

The value axioms. Fix a linear subspace $Q \subseteq BV$ that is *symmetric* in the sense of being closed under Θ_* for all $\Theta \in \mathcal{G}$. Following [30, Def. 1], a *value on* Q is a linear map $\varphi : Q \to FA$ that is symmetric ($\varphi(\Theta_* v) = \Theta_*(\varphi v)$ whenever v and $\Theta_* v$ both lie in Q), positive ($\varphi v \in BV^+$ whenever $v \in Q^+$), and efficient ($(\varphi v)(\Omega) = v(\Omega)$ for every $v \in Q$). On finite player sets these four axioms uniquely determine the classical Shapley value [30, Thm. 1]. The substantive content of the infinite-player theory is that on several natural infinite-dimensional subspaces of BV the same axioms still pin down a unique map φ.

The space pNA. The most important such subspace is pNA, the closure in $\|\cdot\|$ of the subalgebra generated by NA, equivalently the closed linear span of powers of non-atomic measures [1, Lem. 7.2]. The canonical examples are games of the form $v = f \circ \mu$ where $\mu = (\mu_1, \dots, \mu_m) \in (NA^1)^m$ is a vector of non-atomic probability measures and $f \in C^1([0,1]^m)$ with $f(0) = 0$. Such v lies in pNA [30, after Def. 1], and this is the form taken by the smooth operator games of Thm. 2. On pNA there is a unique value [30, Thm. 2], given on the canonical games by the *diagonal formula*

$$(\varphi v)(T) \;=\; \int_0^1 \sum_{i=1}^{m} \mu_i(T)\, \partial_i f(t, \dots, t)\, dt, \qquad T \in \mathcal{C},$$

i.e. the integral of the partial derivatives of f along the diagonal $a = (t, \dots, t)$ of the unit cube, weighted by the measures μ_i.

The asymptotic value. A second, constructive route to a value approximates a continuum game by finite ones. Given a coalition $T \in \mathcal{C}$, a sequence $(\pi_k)_{k \geq 1}$ of finite measurable subfields of $\mathcal{C}$ is *T-admissible* if $T \in \pi_1$, $\pi_k \subseteq \pi_{k+1}$ for all k, and $\bigcup_k \pi_k$ generates $\mathcal{C}$. For each k, the restriction v_{π_k} of v to π_k is a finite-player game and admits the classical Shapley value ψv_{π_k}. We ask whether these discrete Shapley values converge as the partition refines.

Definition 11 (Asymptotic value, [30, Def. 4])**.** *A finitely additive game φv is the* asymptotic value *of $v \in BV$ if, for every $T \in \mathcal{C}$ and every T-admissible sequence (π_k),*

$$\lim_{k \to \infty} \big(\psi v_{\pi_k}\big)(T) \;=\; (\varphi v)(T).$$

The set of all bounded-variation games admitting an asymptotic value is denoted $ASYMP$. When it exists, the asymptotic value is unique, and the map $v \mapsto \varphi v$ is itself a value on $ASYMP$ [30, Thm. 4].

The link between the two routes is the inclusion $pNA \subset ASYMP$, with the asymptotic value coinciding with the unique value on pNA [30, Thm. 4]. This is what we exploit in Sec. 5: the discrete Shapley values ψv_{π_k} of any refining sequence of finite subfields converge to the same limit—the Aumann–Shapley value $(\varphi v)(T)$—which justifies training a discretization-based explainer at multiple grid resolutions and reading off the continuum density at the limit.

A.4 Reviewing the FNO architecture

We repeat the definition of a truncated L-layer FNO with lifting and projection: starting from $v_0 := P_{\mathrm{up}} u + b_{\mathrm{up}}$, define recursively

$$v_{\ell+1} = \sigma\big(A_\ell v_\ell + b_\ell\big), \qquad \ell = 0, \dots, L-1, \tag{3}$$

and set $\hat{\mathcal{F}}(u) := P_{\mathrm{down}} v_L + b_{\mathrm{down}}$. We assume $s > d/2$, and have that

$$P_{\mathrm{up}} : H^s(\Omega; \mathbb{R}^{c_{\mathrm{in}}}) \to H^s(\Omega; \mathbb{R}^{c_0}), \qquad P_{\mathrm{down}} : H^s(\Omega; \mathbb{R}^{c_L}) \to H^s(\Omega; \mathbb{R}^{c_{\mathrm{out}}}),$$

are bounded linear lifting and projection maps with biases $b_{\mathrm{up}} \in H^s(\Omega; \mathbb{R}^{c_0})$ and $b_{\mathrm{down}} \in H^s(\Omega; \mathbb{R}^{c_{\mathrm{out}}})$. Each $A_\ell = W_\ell + K_\ell$ has $W_\ell : H^s(\Omega; \mathbb{R}^{c_\ell}) \to H^s(\Omega; \mathbb{R}^{c_{\ell+1}})$ bounded linear and K_ℓ a finite-mode Fourier multiplier defined by $\widehat{K_\ell v}(k) = R_{\ell,k} \widehat{v}(k)$ for $|k|_\infty \leq m_\ell$ and zero otherwise, with cutoff $m_\ell \in \mathbb{N}$ and learned $R_{\ell,k} \in \mathbb{R}^{c_{\ell+1} \times c_\ell}$. Layer biases satisfy $b_\ell \in H^s(\Omega; \mathbb{R}^{c_{\ell+1}})$, and $\sigma : \mathbb{R} \to \mathbb{R}$ acts componentwise.

B Proofs

B.1 The operator game is in pNA

Theorem 2 (The smooth operator game is pNA)**.** *Under the assumptions of Def. 1, the smooth operator game v lies in pNA, and admits a unique Aumann–Shapley value φv given by the diagonal formula [30, Thm. 2].*

Proof. Define $f_x : [0,1]^m \to \mathbb{R}$ by $f_x(a) := \hat{\mathcal{F}}_x\big(\bar{u} + \sum_{i=1}^m a_i\,\eta_i h\big) - \hat{\mathcal{F}}_x(\bar{u})$ and $\mu_i(S) := \int_S \eta_i\,d\lambda / \int_\Omega \eta_i\,d\lambda$. Then $v = f_x \circ (\mu_1, \dots, \mu_m)$ by construction.

The μ_i are non-atomic probability measures. Each μ_i is a probability measure on $(\Omega, \mathcal{C})$ since $\eta_i \geq 0$ and $\mu_i(\Omega) = 1$, and is absolutely continuous w.r.t. Lebesgue measure with density $\eta_i / \int_\Omega \eta_i\,d\lambda$, hence non-atomic since Lebesgue measure on $\Omega = \mathbb{T}^d$ is non-atomic.

$f_x \in C^1([0,1]^m)$ with $f_x(0) = 0$. The map $\Lambda : \mathbb{R}^m \to X$, $a \mapsto \bar{u} + \sum_i a_i \eta_i h$, is affine with bounded derivative $D\Lambda[a] : b \mapsto \sum_i b_i \eta_i h$ (the $\eta_i h$ lie in X since multiplication by $\eta_i \in C^\infty(\Omega)$ preserves H^s). Since $\hat{\mathcal{F}}_x$ is Fréchet C^1, the chain rule gives $f_x \in C^1([0,1]^m)$ with $\partial_i f_x(a) = D\hat{\mathcal{F}}_x(\Lambda(a))[\eta_i h]$. Setting $a = 0$ yields $f_x(0) = \hat{\mathcal{F}}_x(\bar{u}) - \hat{\mathcal{F}}_x(\bar{u}) = 0$.

The factorization $v = f_x \circ (\mu_1, \dots, \mu_m)$ therefore matches the canonical pNA representation [30, after Def. 1], so $v \in pNA$. Existence, uniqueness, and the diagonal-formula expression of the Aumann–Shapley value follow from [30, Thm. 2]. □

B.2 Smooth FNOs are Fréchet C^1

Proposition 6 (Smooth FNOs are Fréchet C^1)**.** *For $s > d/2$ and $\sigma \in C^\infty(\mathbb{R})$ with $\sigma(0) = 0$, the FNO of App. A.4 defines a Fréchet C^1 map $\hat{\mathcal{F}} : H^s(\Omega; \mathbb{R}^{c_{\text{in}}}) \to H^s(\Omega; \mathbb{R}^{c_{\text{out}}})$.*

Proof. We proceed in four steps.

Step 1: every linear component is bounded on the relevant Sobolev spaces. We first verify that all affine-linear pieces of the architecture act continuously on the Sobolev spaces under consideration.

(a) Lifting and projection. By assumption,

$$P_{\text{up}} : H^s(\Omega; \mathbb{R}^{c_{\text{in}}}) \to H^s(\Omega; \mathbb{R}^{c_0}), \qquad P_{\text{down}} : H^s(\Omega; \mathbb{R}^{c_L}) \to H^s(\Omega; \mathbb{R}^{c_{\text{out}}})$$

are bounded linear maps. Hence

$$u \mapsto P_{\text{up}} u + b_{\text{up}} \quad \text{and} \quad v \mapsto P_{\text{down}} v + b_{\text{down}}$$

are affine continuous maps between Banach spaces.

(b) Channel-mixing operators. For each layer ℓ, the operator

$$W_\ell : H^s(\Omega; \mathbb{R}^{c_\ell}) \to H^s(\Omega; \mathbb{R}^{c_{\ell+1}})$$

is assumed bounded linear.

(c) Truncated Fourier multipliers. Fix ℓ and $v \in H^s(\Omega; \mathbb{R}^{c_\ell})$. On the torus and since K_ℓ has only finitely many active modes,

$$\|K_\ell v\|_{H^s}^2 = \sum_{k \in \mathbb{Z}^d} (1 + |k|^2)^s |\widehat{K_\ell v}(k)|^2 = \sum_{|k|_\infty \leq m_\ell} (1 + |k|^2)^s |R_{\ell,k} \widehat{v}(k)|^2.$$

Using the operator norm of each matrix $R_{\ell,k}$,

$$|R_{\ell,k} \widehat{v}(k)|^2 \leq \|R_{\ell,k}\|_{\text{op}}^2\, |\widehat{v}(k)|^2,$$

so

$$\|K_\ell v\|_{H^s}^2 \leq \Big(\sup_{|k|_\infty \leq m_\ell} \|R_{\ell,k}\|_{\text{op}}^2 \Big) \sum_{|k|_\infty \leq m_\ell} (1 + |k|^2)^s |\widehat{v}(k)|^2 \leq C_\ell^2 \|v\|_{H^s}^2,$$

where

$$C_\ell := \sup_{|k|_\infty \le m_\ell} \|R_{\ell,k}\|_{\mathrm{op}} < \infty.$$

Thus K_ℓ is bounded linear on $H^s(\Omega;\mathbb{R}^{c_\ell}) \to H^s(\Omega;\mathbb{R}^{c_{\ell+1}})$.

(d) Layerwise affine map. Since both W_ℓ and K_ℓ are bounded linear, so is

$$A_\ell = W_\ell + K_\ell.$$

Therefore

$$v \mapsto A_\ell v + b_\ell$$

is an affine continuous map

$$H^s(\Omega;\mathbb{R}^{c_\ell}) \to H^s(\Omega;\mathbb{R}^{c_{\ell+1}}).$$

In particular, every linear or affine block appearing in the architecture is well-defined and continuous on the corresponding Sobolev space.

Step 2: the activation acts smoothly on H^s. Because $s > d/2$, $H^s(\Omega)$ is a Banach algebra under pointwise multiplication and embeds continuously into $L^\infty(\Omega)$ (Runst–Sickel [32], Thm. 4.6.4/1, applied with $p = q = 2$ so that $F^s_{2,2} = H^s$ and $s > d/2 = d/p$). The Nemytskij theorem for smooth nonlinearities ([32], Thm. 5.5.3/2) implies that for $\sigma \in C^\infty(\mathbb{R})$ with $\sigma(0) = 0$, the superposition map

$$S_\sigma : H^s(\Omega;\mathbb{R}^m) \to H^s(\Omega;\mathbb{R}^m), \qquad S_\sigma(w) = \sigma \circ w,$$

is Fréchet C^1 for every finite m, with derivative

$$DS_\sigma(w)[z] = \sigma'(w)\, z$$

componentwise. Since H^s is a Banach algebra and $\sigma'(w) \in H^s \cap L^\infty$, the map $z \mapsto \sigma'(w)z$ is bounded linear on H^s.

Step 3: each hidden layer map is Fréchet C^1. Define

$$T_\ell(v) := S_\sigma(A_\ell v + b_\ell), \qquad \ell = 0, \dots, L-1.$$

By Step 1, the map $v \mapsto A_\ell v + b_\ell$ is affine continuous. By Step 2, S_σ is Fréchet C^1. Hence the chain rule yields that each T_ℓ is Fréchet C^1 as a map

$$H^s(\Omega;\mathbb{R}^{c_\ell}) \to H^s(\Omega;\mathbb{R}^{c_{\ell+1}}).$$

Similarly, the lifting map

$$T_{\mathrm{up}}(u) := P_{\mathrm{up}} u + b_{\mathrm{up}}$$

and the readout map

$$T_{\mathrm{down}}(v) := P_{\mathrm{down}} v + b_{\mathrm{down}}$$

are affine continuous, hence Fréchet C^1.

Step 4: finite composition preserves C^1. The full operator can be written as

$$\hat{\mathcal{F}} = T_{\mathrm{down}} \circ T_{L-1} \circ \cdots \circ T_0 \circ T_{\mathrm{up}}.$$

Since each factor is Fréchet C^1, repeated application of the chain rule shows that

$$\hat{\mathcal{F}} : H^s(\Omega;\mathbb{R}^{c_{\mathrm{in}}}) \to H^s(\Omega;\mathbb{R}^{c_{\mathrm{out}}})$$

is Fréchet C^1. □

Remark 12. *The proposition covers standard smooth activations such as* $\tanh$ *and GELU. It does not apply verbatim to ReLU, which is not C^1. See the ablation in App. D.5.*

B.3 Extension from the torus to general domains

Proposition 6 is stated on the torus because the Fourier multipliers K_ℓ act on periodic functions. Let $\Omega \subset \mathbb{R}^d$ be a bounded C^∞ domain contained in a cube that we identify with $\mathbb{T}^d$. By [32, Thm. 2.4.2/2] (with $H^s = F^s_{2,2}$) there is a bounded linear extension operator $E : H^s(\Omega) \to H^s(\mathbb{R}^d)$ with $Eu|_\Omega = u$. Multiplying by a cutoff $\chi \in C^\infty_c(\mathbb{R}^d)$ with $\chi \equiv 1$ on Ω and support inside the cube, and reading the result as a periodic function, gives a bounded linear map $E_\chi : H^s(\Omega; \mathbb{R}^{c_{\text{in}}}) \to H^s(\mathbb{T}^d; \mathbb{R}^{c_{\text{in}}})$. The restriction $R : H^s(\mathbb{T}^d; \mathbb{R}^{c_{\text{out}}}) \to H^s(\Omega; \mathbb{R}^{c_{\text{out}}})$ is bounded linear as well. Define $\hat{\mathcal{F}}_\Omega := R \circ \hat{\mathcal{F}} \circ E_\chi$. In practice this composition is the padding used when FNOs are applied to non-periodic data. Since E_χ and R are bounded linear and $\hat{\mathcal{F}}$ is Fréchet C^1 by Proposition 6, the chain rule gives that $\hat{\mathcal{F}}_\Omega$ is Fréchet C^1 with $D\hat{\mathcal{F}}_\Omega(u) = R \circ D\hat{\mathcal{F}}(E_\chi u) \circ E_\chi$.

The remaining results transfer with Ω in place of $\mathbb{T}^d$. The Sobolev embedding $H^s(\Omega) \hookrightarrow C^0(\bar{\Omega})$ for $s > d/2$ follows by extending, embedding on $\mathbb{R}^d$, and restricting, so point evaluation and Def. 1 are well-defined. The proofs of Thms. 2 and 10 use only that Lebesgue measure on Ω is non-atomic, that multiplication by $\eta_i \in C^\infty(\bar{\Omega})$ preserves $H^s(\Omega)$, and the chain rule, all of which hold on Ω.

B.4 Partition of unity for function-valued Integrated Gradients

Theorem 10 (Partition-of-unity attribution)**.** *Let $\hat{\mathcal{F}} : X \to Y$ be Fréchet C^1 and fix $x \in \Omega$. Let $\eta_1, \dots, \eta_m \in C^\infty(\Omega)$ satisfy $\sum_{j=1}^m \eta_j \equiv 1$. Then, for every $\bar{u}, u_0 \in X$,*

$$\hat{\mathcal{F}}(u_0)(x) - \hat{\mathcal{F}}(\bar{u})(x) = \sum_{j=1}^m \mathrm{Attr}_x^{\eta_j}(\bar{u}, u_0).$$

Proof. Set $h = u_0 - \bar{u}$ and $\hat{\mathcal{F}}_x = \mathrm{ev}_x \circ \hat{\mathcal{F}}$ as in Def. 9. Since $\hat{\mathcal{F}}_x$ is Fréchet C^1, the chain rule gives $\frac{d}{dt}\hat{\mathcal{F}}_x(\bar{u} + th) = D\hat{\mathcal{F}}_x(\bar{u} + th)[h]$, which is continuous on $[0, 1]$. The fundamental theorem of calculus then yields

$$\hat{\mathcal{F}}(u_0)(x) - \hat{\mathcal{F}}(\bar{u})(x) = \hat{\mathcal{F}}_x(u_0) - \hat{\mathcal{F}}_x(\bar{u}) = \int_0^1 D\hat{\mathcal{F}}_x(\bar{u} + th)[h]\, dt.$$

Since $\sum_j \eta_j = 1$, $h = \sum_j \eta_j h$ in X. Linearity of $D\hat{\mathcal{F}}_x(\bar{u} + th)$ in its direction argument gives $D\hat{\mathcal{F}}_x(\bar{u} + th)[h] = \sum_j D\hat{\mathcal{F}}_x(\bar{u} + th)[\eta_j h]$. Summing under the integral,

$$\hat{\mathcal{F}}(u_0)(x) - \hat{\mathcal{F}}(\bar{u})(x) = \sum_{j=1}^m \int_0^1 D\hat{\mathcal{F}}_x(\bar{u} + th)[\eta_j h]\, dt = \sum_{j=1}^m \mathrm{Attr}_x^{\eta_j}(\bar{u}, u_0). \qquad \square$$

B.5 Construction of a partition of unity on a discrete grid

Here, we show how to construct partitions of unity for data of a given discrete grid. Our goal is to have a smooth bump for each grid point, with controllable overlap between them. Let $g(t) = e^{-1/t}$ for $t > 0$ and $g(t) = 0$ for $t \le 0$, and define the smooth step $\sigma_\varepsilon(t) = g(t+\varepsilon) \,/\, \big(g(t+\varepsilon) + g(\varepsilon - t)\big)$, which is C^∞, equals 0 for $t \le -\varepsilon$ and 1 for $t \ge \varepsilon$, and is monotone in between. For grid points $x_1 < \cdots < x_p$ in $[0, 1]$ with midpoints $m_i = (x_i + x_{i+1})/2$ and a transition width $\varepsilon < \min_i(x_{i+1} - x_i)/2$, set

$$\begin{aligned}
\eta_1(x) &= 1 - \sigma_\varepsilon(x - m_1), \\
\eta_i(x) &= \sigma_\varepsilon(x - m_{i-1}) - \sigma_\varepsilon(x - m_i) \quad (1 < i < p), \\
\eta_p(x) &= \sigma_\varepsilon(x - m_{p-1}).
\end{aligned}$$

The η_i are smooth and non-negative, sum to one by telescoping, and satisfy $\eta_i(x_j) = \delta_{ij}$ because every grid point lies at distance more than ε from all midpoints. Products of one-dimensional masks give the construction on d-dimensional grids. On the torus, η_1 and η_p are merged into one periodic mask.

We now compare the smooth game of Def. 1 with the binary grid game of the experiments. Let B_i be the cell of grid point x_i, bounded by the midpoints m_{i-1} and m_i, and let S be a union of cells. The

theory masks the input as $\bar{u} + \sum_i \mu_i(S)\, \eta_i h$ with $\mu_i(S) = \int_S \eta_i \, d\lambda \,/\, \int_\Omega \eta_i \, d\lambda$, while the experiments use $\bar{u}(x_j) + \mathbf{1}_{x_j \in S}\, h(x_j)$ at every grid point. The backbone reads its input only at the grid points, where $\eta_i(x_j) = \delta_{ij}$, so the smoothly masked input evaluates to $\bar{u}(x_j) + \mu_j(S)\, h(x_j)$ there. The two inputs therefore differ at x_j by $|\mu_j(S) - \mathbf{1}_{x_j \in S}|\, |h(x_j)|$.

The support of η_i is $[m_{i-1} - \varepsilon,\, m_i + \varepsilon]$, so it extends by at most ε into each of the two neighboring cells, and since $\eta_i \le 1$ the mass of η_i outside B_i is at most 2ε. If $x_i \in S$ then $B_i \subseteq S$ and $\int_\Omega \eta_i - \int_S \eta_i \le \int_{\Omega \setminus B_i} \eta_i \le 2\varepsilon$. If $x_i \notin S$ then $S \cap B_i = \emptyset$ and $\int_S \eta_i \le \int_{\Omega \setminus B_i} \eta_i \le 2\varepsilon$. In both cases

$$|\mu_i(S) - \mathbf{1}_{x_i \in S}| \;\le\; \frac{2\varepsilon}{\int_\Omega \eta_i \, d\lambda} \;\le\; \frac{2\varepsilon}{|B_i| - 2\varepsilon}.$$

The width ε is free as long as it is positive, which keeps the η_i in H^s, and no result requires it to be large. We therefore choose ε so small that this deviation is below floating point precision. The smooth and the binary masked input are then the same tensor for the backbone, so Alg. 1 computes the values of the smooth game with binary masks. The integral of Alg. 1, the coarsening $\hat{\phi}_i = \int_{B_i} \phi$, involves no mask: by the expansion of the diagonal formula at the end of Sec. 4.3, the masks enter the value $(\varphi v)(S)$ only through $\sum_i \mu_i(S) \int_0^1 D\hat{\mathcal{F}}_x(\bar{u} + th)[\eta_i h]\, dt$, while $\int_{B_i} \phi$ integrates the explainer output over a cell and is the same in the smooth and in the binary view. Note that the FastSHAP loss regresses the discrete Shapley values of the game at the current partition and not this diagonal formula, which the explainer only approaches as the partitions get finer (App. A.3). Fig. 5 shows the construction.

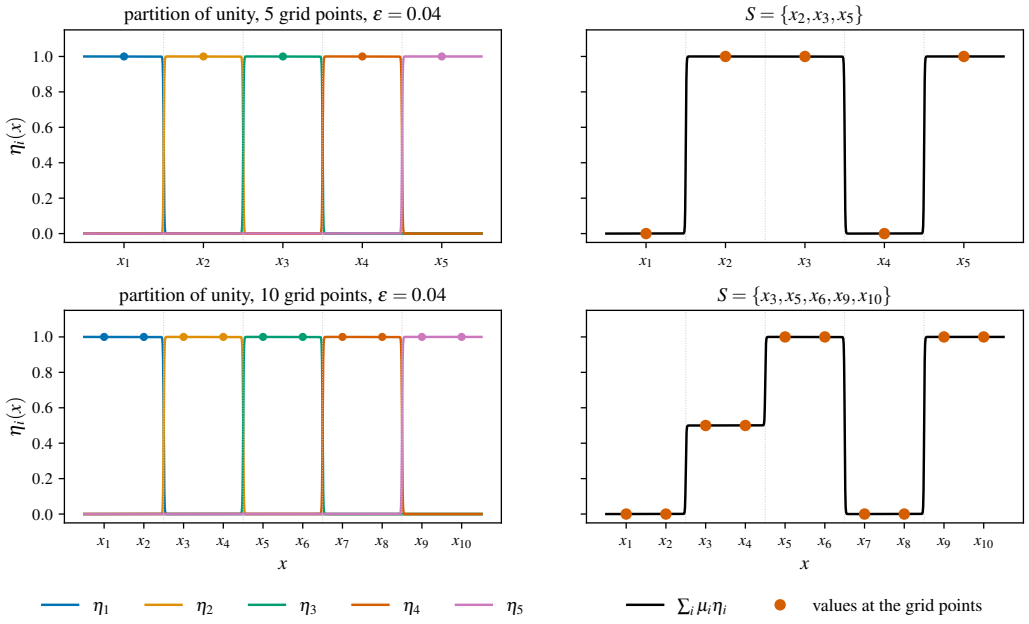


Figure 5: Grid-supported partition of unity with five masks. The transition width $\varepsilon = 0.04$ is chosen large so that the transitions are visible. Dotted lines mark the midpoints m_i. *Top:* one grid point per mask. *Left:* the masks η_i. Dots mark $\eta_i(x_i) = 1$. *Right:* the smooth mask $\sum_i \mu_i \eta_i$ of the coalition $S = \{x_2, x_3, x_5\}$ and its values at the grid points, which form the binary grid mask. *Bottom:* the same masks on a grid with two points per mask, so each η_i equals one at both grid points of its cell. *Right:* the coalition $S = \{x_3, x_5, x_6, x_9, x_{10}\}$ holds one of the two grid points of η_2, so $\mu_2 = 1/2$ and the smooth mask is $1/2$ at both x_3 and x_4. In total, $\mu = (0, 1/2, 1, 0, 1)$.

B.6 Multiple output channels

Def. 1 needs a scalar game. We therefore reduce the backbone output at x to one number before we build the game. On the grid datasets we read the first output channel (the u channel of Gray–Scott)

and on the airfoil and cylinder flow the velocity magnitude. The proofs do not change, because both readouts are C^1 maps of the output vector (the norm away from zero) and compose with ev_x by the chain rule. One could instead train one explainer per output channel, use a vectorized version of the loss, which averages over the dimensions, or pass the channel index to the explainer as an additional input, like the query point. We do not do this.

C Experimental setup

C.1 Datasets

We use nine PDE datasets across 1D and 2D, the 3D heat equation, plus two unstructured-mesh datasets from the MeshGraphNets benchmark (Tab. 2). The 1D PDEs all use a native resolution of 1024 points on $[0, 1]$, the structured 2D PDEs use 128×128 points on a unit square (or $[0, 2.5]^2$ for Gray–Scott, see below), the 3D heat equation uses 32^3 points on the unit cube, and every structured dataset is sampled at the training resolutions $\{8, 16, 32, 64, 128, 256, 512, 1024\}$ by stride-downsampling from native resolution (for 2D data, we used $\{16, 64, 256, 1024\}$ players and for 3D data 512 players). Each dataset has $5\,000$ generated samples split into 4220 train, 750 val, and 30 test samples.

Linear PDEs (FFT-analytic solver). The advection equation $\partial_t u + \beta \partial_x u = 0$ on a periodic domain ($\beta = 1.0$), the heat equation $\partial_t u = \kappa \partial_x^2 u$ in 1D ($\kappa = 0.01$), and its 2D and 3D analogs $\partial_t u = \kappa \Delta u$ ($\kappa = 0.01$) admit closed-form transfer functions in Fourier space, so the solver applies the analytic propagator at each requested time.

Nonlinear PDEs (pseudo-spectral method of lines). Burgers $\partial_t u + u \partial_x u = \nu \partial_x^2 u$ ($\nu = 0.01$) and Allen–Cahn $\partial_t u = \varepsilon^2 \partial_x^2 u + u - u^3$ ($\varepsilon = 0.1$) are integrated with FFT-based spatial derivatives (for Burgers, the nonlinear term keeps the modes $|k| \leq 170$ of 512), and an adaptive time integrator (`RK45` for Burgers, the stiff `Radau` for Allen–Cahn). The 2D Gray–Scott reaction–diffusion system, $\partial_t u = D_u \Delta u - uv^2 + F(1-u),\ \partial_t v = D_v \Delta v + uv^2 - (F+k)v$ with $D_u = 2{\cdot}10^{-5}$, $D_v = 10^{-5}$, $F = 0.04$, $k = 0.06$ on $[0, 2.5]^2$, is integrated the same way (`RK45`) but for a long time horizon $T = 5000$ in order to reach the pattern-forming regime. Gray–Scott has two channels in input and output (u and v), while the other PDEs are scalar.

Finite-difference / finite-volume PDEs. Diffusion–sorption, $\partial_t u = D \partial_x^2 u / R(u)$ with Freundlich retardation $R(u) = 1 + (\rho_s/\phi) k_f n_f u^{n_f - 1}$, lives on a non-periodic domain with Dirichlet inflow on the left and zero-flux on the right. We use centered finite differences and `Radau` time stepping ($D = 5{\cdot}10^{-4}$, $k_f = 3.5$, $n_f = 0.4$, $\rho_s = 1.0$, $\phi = 0.29$, $c_{\text{left}} = 1.0$). The 2D shallow-water system $\partial_t (h, hu, hv) + \nabla \cdot F(h, hu, hv) = 0$ with $g = 9.81$ uses a first-order finite-volume scheme with HLL Riemann fluxes, an SSP-RK3 time integrator at $\mathrm{CFL} = 0.4$, and reflective boundaries. We predict only the height field h as the backbone target. Darcy flow $-\nabla \cdot (a(x) \nabla u) = f$ is the only static problem in the set: the solver assembles a sparse stencil with harmonic averaging at cell faces and inverts it directly with `scipy.sparse.linalg.spsolve`, with permeability a drawn from a log-normal random field with power spectrum $|k|^{-4}$ and forcing $f = 1$.

MeshGraphNets benchmarks. We additionally use the airfoil and cylinder-flow datasets from the MeshGraphNets release [31]. The airfoil dataset is a transonic Euler flow around a NACA airfoil with $3\,994$ unstructured mesh nodes per sample after the cutout (the node count is fixed across samples). We predict the velocity field. The cylinder-flow dataset is an incompressible flow past a cylinder with a genuinely heterogeneous mesh – the node count varies sample-to-sample because of adaptive refinement – and we predict the velocity field. Cylinder-flow is the dataset that motivates a grid-agnostic explainer at all: a fixed-grid amortized explainer cannot apply across samples that do not share a common discretization. We inherit the train/val/test split from the original release.

Multi-resolution sampling. At training time, each sample is deterministically pinned to one of the possible pre-selected resolutions via the `ResolutionBatchSampler`: the samples are split into contiguous blocks, one per resolution, so a given sample is always served at the same resolution across epochs. The mesh-based datasets do not use multi-resolution sampling, but assign each node to one of 64 boxes of an 8×8 grid over the normalized mesh frame to reduce the number of players.

Table 2: The PDE datasets. All structured-grid datasets have native spatial resolution 1024 (1D), 128×128 (2D), or 32^3 (3D), $5\,000$ generated samples, split into 4220 train, 750 val, and 30 test samples. "t_{idx}" is the snapshot index used as the backbone target on a uniform grid of n_{snap} snapshots over $[0, T]$. The label "static" marks the only time-independent PDE. "IC family" names the random initial-condition generator.

PDE	Equation / parameters	IC family	T	n_{snap}	t_{idx}
Advection 1D	$\partial_t u + \beta \partial_x u = 0$, $\beta = 1$	mixed-1D	1.0	101	30
Heat 1D	$\partial_t u = \kappa \partial_x^2 u$, $\kappa = 0.01$	mixed-1D	1.0	101	50
Burgers 1D	$\partial_t u + u \partial_x u = \nu \partial_x^2 u$, $\nu = 0.01$	mixed-1D	1.0	101	50
Allen–Cahn 1D	$\partial_t u = \varepsilon^2 \partial_x^2 u + u - u^3$, $\varepsilon = 0.1$	tanh-mixture	0.5	101	50
Diff.–Sorp. 1D	Freundlich retardation; $D = 5{\cdot}10^{-4}$	boundary-driven	500	101	50
Heat 2D	$\partial_t u = \kappa \Delta u$, $\kappa = 0.01$	rand. Fourier (8 modes)	1.0	51	25
Reac.–Diff. 2D (Gray–Scott)	$D_u = 2{\cdot}10^{-5}$, $D_v = 10^{-5}$, F=0.04, k=0.06	seeded patches	5000	51	25
Shallow Water 2D	$g = 9.81$, CFL $= 0.4$	Gaussian bump	0.5	51	25
Darcy 2D	$-\nabla{\cdot}(a\nabla u)$=1, $a \sim$log-normal	log-normal field	static	—	—
Heat 3D	$\partial_t u = \kappa \Delta u$, $\kappa = 0.01$	rand. Fourier (4 modes)	1.0	11	5
MGN Airfoil	transonic Euler around NACA	—	—	—	—
MGN Cylinder	incompressible cylinder flow	—	—	—	—

The 1D "mixed-1D" IC family draws with equal probability from random 20-mode Fourier series with power-law decay, single Fourier modes, 1–4 Gaussian bumps, and low-mode (1–3) Fourier sums. This is designed so the backbone sees both smooth multiscale data and localized features. The Allen–Cahn ICs are random piecewise-tanh profiles with three transitions and interface width 0.05. Diffusion–sorption uses smooth left-driven exponentials $c_{\mathrm{left}} \exp(-x/\lambda)$ with λ randomized around the design value $0.05L$. The 2D ICs are detailed in their respective generators.

C.2 Data generation

All structured-grid datasets are generated by a single script (`generate_data.py`) configured per PDE through Hydra [39]: a top-level config selects the solver, the IC generator, the spatial grid, and the time grid (App. C.1). Generation proceeds sample-by-sample and is embarrassingly parallel–each sample is one IC draw followed by one solver call–so we dispatch all $n = 5\,000$ samples across CPU workers via `joblib` with the `loky` backend. Reproducibility is independent of worker count: we seed a single `SeedSequence(42)` and `spawn` n child sequences from it, one per sample, so a sample with index i gets the same RNG state regardless of which worker picks it up. After all workers finish we sort the results by index and stack them into the dataset arrays.

The 1D outputs are saved as u_0 of shape $(n, 1024)$ together with the solution $u(\cdot, t)$ of shape $(n, n_{\mathrm{snap}}, 1024)$, and the 2D and 3D outputs add one or two spatial axes (and a leading channel axis for Gray–Scott). Darcy is the exception: it is static and stored as permeability and pressure pairs of shape $(n, 128, 128)$ each.

The MeshGraphNets datasets are not generated by us. We reuse the original trajectories from [31], repackaged as per-trajectory `.npz` files under the same train/val/test split as the original release.

C.3 Models

Backbones. For every dataset we use one of four backbones, each chosen so that its inductive bias matches the data, and MWT-Gupta-1D for the analytic validation. The 1D PDEs use a standard 1D Fourier neural operator (**FNO1D**: hidden width 64, 4 layers, 16 retained Fourier modes, domain padding 0.5) and, as a non-FNO comparison architecture, the multiwavelet-transform operator of Gupta et al. [14] (**MWT-Gupta-1D**: Legendre basis, channel width $c = 16$, wavelet order $k = 4$, $\alpha = 10$, $n_{CZ} = 2$, $L = 0$, with the input u_0 concatenated with the spatial coordinate as a second channel). The structured 2D PDEs use the local-integral variant of Liu-Schiaffini et al. [22] (**LocalFNO2D**: a 2D FNO with hidden width 64, 4 layers, 16×16 modes, and padding 0.5, plus a DISCO local-convolution branch sized for `default_in_shape` $= [128, 128]$). The 3D heat equation uses a 3D FNO (**FNO3D**: hidden 32, 4 layers, $8 \times 8 \times 8$ modes, padding 0.5).

The unstructured-mesh datasets use the geometry-informed neural operator (**GINO**) of Li et al. [20]: GNO encoder and decoder with neighbor-search radius 0.05 (0.12 for the airfoil encoder, in the normalized $[0, 1]^2$ frame) and a nonlinear kernel transform, around a 16×16-mode FNO at hidden

width 64 on a 64×64 regular latent grid. For the MGN datasets the only change is the I/O width: 2 input channels and 2 output channels for the velocity field (v_x, v_y).

Explainers. Every explainer reuses the architecture of its backbone–FNO backbone with FNO explainer, GINO with GINO, and so on–with three changes. First, the output channel count is always set to 1, since the explainer returns a single scalar attribution density $\phi(x)$ regardless of the I/O width of the backbone (the GINO-MGN explainer therefore drops from 2 output channels to 1). Second, on the LocalFNO2D explainer the DISCO `default_in_shape` is set to the largest training resolution served by the experiment ($[32, 32]$ in the runs we report) so the local kernels match the receptive field actually seen during the FastSHAP loop. Third, the explainer gets one additional input channel that encodes the query point. The explainer therefore matches the backbone in parameter budget up to the added input channel and the dropped output channels. The query channel is a Gaussian bump of width 0.05 of the domain centered at $x^\star$, independent of the grid. During training, $x^\star$ is drawn uniformly over the grid for every sample in every batch, and the masked backbone output is read at that point.

C.4 Training

We use "Weights and Biases" [2] for our experiment tracking.

Backbones. All backbones are trained with the same recipe: `AdamW` (learning rate 10^{-3}, weight decay 10^{-4}), cosine learning-rate schedule, MSE loss against the held-out solution at the target snapshot, 200 epochs with early stopping on validation MSE (patience 50, no minimum-delta requirement), and the best-val-loss checkpoint kept. The structured-grid backbones are trained on 16 to 128 points per axis (8 and 16 in 3D), so explanations on coarser and finer grids evaluate the backbone outside its training resolutions. The batch size is 64 for the structured-grid backbones (FNO1D, LocalFNO2D, FNO3D, MWT-Gupta-1D) and 1 for the GINO backbones, where the per-sample mesh forces single-mesh forwards (`neuralop.GINO` only batches across samples that share a geometry).

Explainer training. The OperatorSHAP loop of Alg. 1 is trained with `AdamW` (learning rate $3{\cdot}10^{-4}$, weight decay 10^{-5}), cosine decay without warmup, and the FastSHAP weighted-MSE loss already shown in Alg. 1. We run for at most 300 epochs with patience 50 and batch size 64. At every step we draw $K = 128$ coalitions $S \subseteq P$ from the Shapley kernel weighting $w(s) \propto 1/(s(n-s))$ in paired form (each S is drawn together with its complement to reduce gradient variance, so that a step uses $2K = 256$ coalitions), evaluate the backbone on the masked inputs u_S to obtain the targets $v(S)$, and update the explainer against the FastSHAP weighted MSE. The coalitions are binary masks at the level of the explainer's output grid (n_{players} equals the spatial resolution of the current batch), efficiency is enforced by hard normalization, and the query point $x^\star$ is uniformly sampled on the domain for every sample and given as a Gaussian bump in an additional input channel to the model. Because the query point changes, $v(\Omega)$ and $v(\emptyset)$ are recomputed every batch. We use the zero baseline for all runs except the MGN data and the baseline ablation of App. D.4. For the MGN data we use the global mean for the cylinder flow and the per-node mean for the airfoil flow.

MGN-specific deltas. The MGN explainer training uses the same optimizer settings as the structured-grid case but differs in a handful of places dictated by the mesh setting. Batch size drops to 1 (GINO constraint) and we run for at most 100 epochs at patience 15. The coalition partition is a fixed 8×8 grid in the normalized mesh frame ($n = 64$ players, independent of mesh density). Integrals over a mesh use the lumped-mass quadrature of its triangulation. Every node carries one third of the area of each triangle it belongs to. The coarsening $\hat{\phi}_i = \int_{B_i} \phi$ is the sum of the explainer output times this weight over the nodes in box B_i, and the measure $|B_i|$ in the normalization is the sum of the weights in B_i. A box is therefore measured by the area it covers and not by the number of nodes in it, so the refined boundary layer and the wake carry no extra weight. The meshes have no nodes inside the cylinder or the airfoil. A box that overlaps the obstacle contains only the nodes of the flow region, and its measure is the flow area it covers, up to the triangles that straddle its boundary. A box without nodes has measure zero. Masking it changes no node and it receives no attribution. We draw $K = 32$ coalition pairs (64 coalitions) per step from the Shapley distribution. Coalition forwards are chunked at most 128 (cylinder flow) and 32 (airfoil) at a time

through GINO to keep the per-coalition activation in GPU memory. The coalition forwards run in bf16 autocast, with the neighborhood mean of the input GNO in fp32. In bf16, this mean over about 170 neighbors moves the airfoil coalition values by 0.29, against a spread of 0.07 across coalitions. With the fp32 mean, the coalition values of both meshes change by less than 0.003 (spread 0.07 on the airfoil and 0.17 on the cylinder). Because the randomly sampled initial time t_0 also changes the input function, $v(\Omega)$ cannot be cached and is recomputed every batch. During training, the query point is a mesh node drawn per sample with probability proportional to its quadrature weight, that is, uniformly over the flow area. For evaluation, the query points are given in physical coordinates and resolved per sample to the nearest mesh node.

C.5 Attribution methods

Notation and conventions. For every attribution method below we partition the input domain Ω into n cells – the *player grid* P – and define the discrete coalition game $v(S) = \hat{\mathcal{F}}(u_S)(x^\star) - \hat{\mathcal{F}}(\bar{u})(x^\star), \qquad S \subseteq P$, where $u_S = \bar{u} + \mathbf{1}_S \odot (u_0 - \bar{u})$ is the input with cells in S revealed and the rest held at the baseline $\bar{u}$. In the decision game of App. D.6, the readout $\hat{\mathcal{F}}(u_S)(x^\star)$ is replaced by the midpoint indicator, and every method below, IG included, uses that readout. The baseline defaults to zero for the ten structured-grid PDEs, to per-node mean for MGN airfoil, and to global mean for MGN cylinder flow. On the structured grids, every test input is explained at three query points $x^\star$, the domain center and two off-center points with coordinates at a quarter and three quarters of the domain. On the meshes, the three query points are fixed physical locations, resolved per sample to the nearest mesh node. All attribution runs use seed 42. The OperatorSHAP and KERNELSHAP / EXACT / REGRESSIONMSR estimators all see the same coalition game, but they differ only in how they explore the coalition space. IG is a non-Shapley density baseline that does not see the game at all but integrates gradients of $\hat{\mathcal{F}}$ along the line from $\bar{u}$ to u_0.

Exact Shapley. For $n \leq 16$ we enumerate all 2^n coalitions and apply the closed-form Shapley sum of (1) via `shapiq.ExactComputer` [9]. This gives a deterministic ground-truth attribution ($65{,}536$ coalitions at $n = 16$), which we use as the zero-error reference at the smallest player count in our budget-curve plots.

KernelSHAP. The Lundberg–Lee weighted-least-squares estimator [23], run through `shapiq.KernelSHAP` [9]. We use it as a baseline at budgets from 64 to 8192 to track convergence to the exact solution (at $n = 16$) and to the higher-budget RegressionMSR reference (at $n \in \{512, 1024\}$) in Fig. 6, and at budgets from 128 to 32768 at $n = 256$ in Figs. 7 and 8.

RegressionMSR. The regression-adjusted Monte Carlo estimator of Witter et al. [38], also run through shapiq [9]. RegressionMSR is a more modern variant of KernelSHAP that often performs better at higher player numbers than KernelSHAP. It is therefore our default discrete reference with a budget of $300{,}000$, where the exact 2^n enumeration is infeasible.

OperatorSHAP / FastSHAP. The amortized explainer of Sec. 5. The training procedure is documented in App. C.4. At inference time, a single forward pass of $\Phi_\theta(u_0, x^\star)$ produces the full-resolution density ϕ, and the discrete Shapley value over a partition cell B_i is recovered by integrating ϕ over B_i. On a regular grid every node carries the cell volume, and on a mesh it carries the lumped-mass weight of App. C.4. Hard normalization enforces $\sum_i \hat{\phi}_i = v(\Omega) - v(\emptyset)$ exactly.

Integrated gradients (IG). A non-Shapley density baseline that bypasses the coalition game entirely. Following the construction of Sec. 4.3, we compute the IG density via a 50-step midpoint Riemann sum of the directional derivative $D\hat{\mathcal{F}}_{x^\star}\big(\bar{u} + th\big)[h]$ along the straight line $\bar{u} \to u_0$ with the same baseline $\bar{u}$ as the game, with the gradient at each step obtained by a backward pass through the backbone. Any region attribution is then a cheap weighted sum of this density against a smooth mask. Recent PyTorch versions before 2.11 return wrong gradients through complex `torch.einsum` on the Apple MPS backend (PyTorch issue #177474), which the Fourier layers of our backbones rely on. This bug affected the IG results in an earlier version of this paper and users should be aware to use an up to date PyTorch version or run the methods on CPU or CUDA. The IG results reported in this version of the paper do not use MPS anymore and are correct.

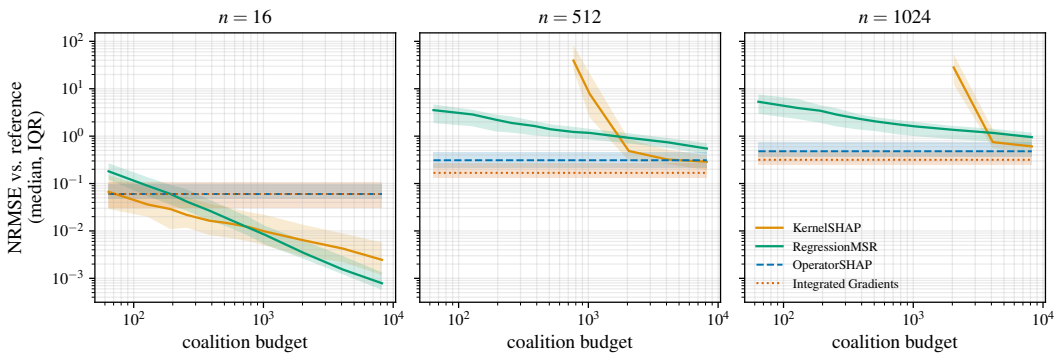


Figure 6: Normalized RMSE (NRMSE) of attribution methods vs. ground truth (for $n = 16$: Exact Shapley, otherwise RegressionMSR with 300k budget) for different player numbers (spatial resolution) 16 (*left*), 512 (*middle*), 1024 (*right*). The PDE is Burgers 1D. The lines show the median with 0.25- and 0.75-quantile bands over 30 test inputs and three query points, and the horizontal lines mark OperatorSHAP and Integrated Gradients. The same OperatorSHAP model was used at all resolutions. We can see that for low player numbers, classical Shapley approximators outperform OperatorSHAP already for low budgets, while it takes significantly longer for higher player numbers.

C.6 Evaluation metrics

For every metric below we reduce along the player or grid dimension to one scalar per test sample, and report the median and inter-quartile ranges (IQR) over a held-out set of 30 test inputs (10 for the MGN meshes) and three query points per PDE. We use the following metrics.

Pearson correlation. The standard sample Pearson coefficient between the predicted and reference Shapley vectors at the reference's partition, $r(\hat{\varphi}, \varphi^{\mathrm{ref}}) = \frac{(\hat{\varphi}-\bar{\hat{\varphi}})^\top(\varphi^{\mathrm{ref}}-\bar{\varphi}^{\mathrm{ref}})}{\|\hat{\varphi}-\bar{\hat{\varphi}}\|\,\|\varphi^{\mathrm{ref}}-\bar{\varphi}^{\mathrm{ref}}\|}$, computed per sample along the player dimension. Pearson is invariant to constant shifts and global rescalings, so it captures the *shape* of the attribution profile rather than its magnitude.

Normalized RMSE. The relative ℓ_2 error along the player dimension, $\mathrm{NRMSE}(\hat{\varphi}, \varphi^{\mathrm{ref}}) = \|\hat{\varphi} - \varphi^{\mathrm{ref}}\|_2 \,/\, \|\varphi^{\mathrm{ref}}\|_2$, which complements Pearson by rewarding the correct attribution *magnitude*.

Faithfulness (weighted R^2). Following the faithfulness diagnostic of Tsai et al. [35], we sample 10,000 random coalitions $S \subseteq P$ from the same Shapley-kernel distribution used during training and report the coefficient of determination of the predicted linear surrogate $\hat{\varphi}^\top \mathbf{1}_S$ against the true coalition gap $v(S) - v(\emptyset)$,

$$R^2 \;=\; 1 - \frac{\sum_S \big(v(S) - v(\emptyset) - \hat{\varphi}^\top \mathbf{1}_S\big)^2}{\sum_S \big(v(S) - v(\emptyset) - \overline{(v(S) - v(\emptyset))}\big)^2}.$$

This measures how well the explainer's linear surrogate generalizes to coalitions it never saw at training time, and unlike Pearson and NRMSE it does not need a reference attribution. Since a binary coalition on the grid is the sample of a smooth coalition (App. B.5), this score is also the faithfulness of the explainer with respect to the smooth game of Def. 1.

D Additional results

Tab. 3 and Tab. 4 reproduce Tab. 1 for all PDEs of our suite, not just those shown in the main text. The same metrics, references, budgets, and reporting convention apply.

D.1 Runtime and faithfulness vs. budget

Fig. 8 scales the budget of the other methods with a multiplier m at $n = 256$: Integrated Gradients uses m path steps, and RegressionMSR and KernelSHAP use $128m$ coalitions. OperatorSHAP

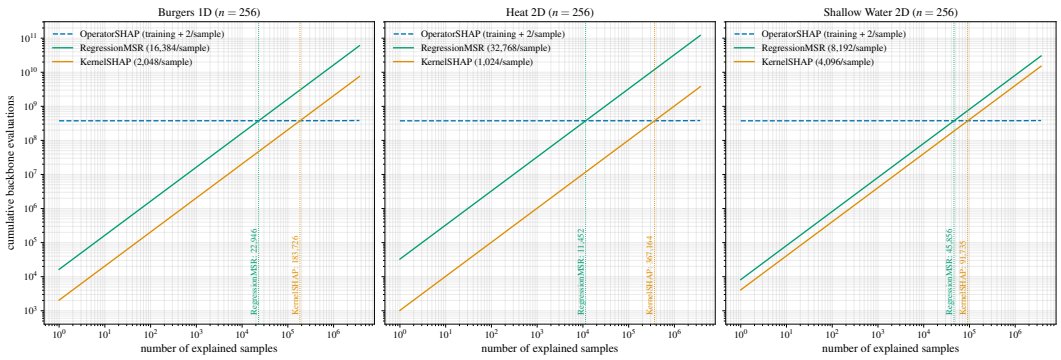


Figure 7: Amortization visualization: We depict the total number of necessary backbone evaluations for inference of increasing sample sizes. OperatorSHAP includes training evaluations, but stays almost constant with increasing sample size, while classical methods grow linearly with the number of samples. We tuned the budget of classical methods to reach a similar error as OperatorSHAP at $n = 256$. We can see that numbers depend on the PDE.

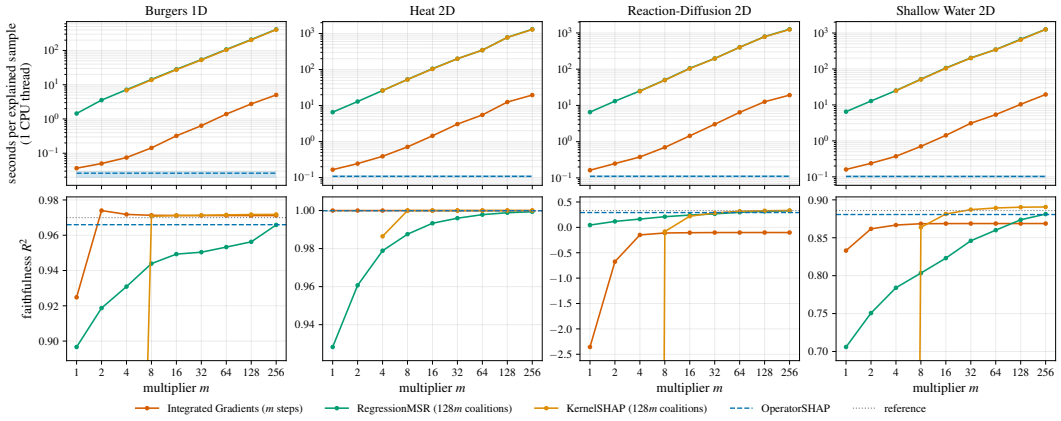


Figure 8: Runtime and faithfulness on four PDEs at $n = 256$. *Top:* wall-clock seconds per explained sample on one thread of an Intel Xeon Platinum 8380, median and inter-quartile range over 10 validation inputs with three query points each. *Bottom:* median faithfulness R^2 over 30 test inputs and three query points. Integrated Gradients uses m path steps, RegressionMSR and KernelSHAP use $128m$ coalitions. The dashed line marks OperatorSHAP and the dotted line the reference. KernelSHAP is shown for budgets above n, from $m = 4$ on. OperatorSHAP is the fastest and needs no tuneable budget. IG sometimes reaches OperatorSHAP at comparable speed and both RegressionMSR and KernelSHAP reach OperatorSHAP at high enough budget, though at significantly lower speed.

does not depend on m. On Burgers 1D and Heat 2D, IG reaches the faithfulness of OperatorSHAP with one or two path steps, while RegressionMSR does not reach it within $m = 256$. On Reaction–Diffusion 2D and Shallow Water 2D, IG stays below OperatorSHAP for every m. RegressionMSR and KernelSHAP reach OperatorSHAP on Reaction–Diffusion 2D at $m = 64$, and KernelSHAP on Shallow Water 2D at $m = 16$, at about 3600 and 950 times its runtime. RegressionMSR reaches it on Shallow Water 2D at $m = 256$. With 512 coalitions ($m = 4$), the weighted least-squares problem of KernelSHAP is close to singular, and its R^2 falls below the plotted range on three PDEs. Every point is a single run.

D.2 Held-out resolutions

The explainers of the main experiments see every evaluated resolution during training. Tab. 5 repeats the evaluation with explainers whose training resolutions leave out one resolution. For $n = 256$, the left-out resolution lies inside the training range ($n = 128$ and $n = 512$ in 1D, 8×8 and 32×32 in

Table 3: Full-suite version of Tab. 1: Cross-PDE attribution quality across resolutions n on the 1D PDEs (other half in Tab. 4). Each (PDE, n, method) entry reports the median over 30 test inputs and three query points per input on the top line and the inter-quartile range in brackets below. Metrics are Pearson correlation and NRMSE against the per-PDE reference attribution, and the weighted-R^2 Faithfulness score. RegressionMSR and KernelSHAP use a budget of $2n$ (minimum 1024) backbone evaluations per query point, and IG is calculated via a 50-step Riemann sum. Across resolutions OperatorSHAP uses the same model per PDE. Higher Pearson and R^2 are better, and lower NRMSE is better. Bold marks the best median and every median within one standard error of it over the test inputs, compared at three decimals.

		R^2 ↑				Pearson ↑				NRMSE ↓			
PDE	n	RMSR	KS	IG	OS (ours)	RMSR	KS	IG	OS (ours)	RMSR	KS	IG	OS (ours)
Heat 1D	32	0.999	**1.000**	**1.000**	**1.000**	0.997	**1.000**	**1.000**	**1.000**	0.082	**0.002**	0.011	0.021
		[0.993, 0.999]	[1.000, 1.000]	[1.000, 1.000]	[0.999, 1.000]	[0.996, 0.998]	[1.000, 1.000]	[1.000, 1.000]	[1.000, 1.000]	[0.063, 0.103]	[0.001, 0.004]	[0.004, 0.022]	[0.016, 0.028]
	64	0.996	**1.000**	**1.000**	**1.000**	0.975	**1.000**	**1.000**	0.999	0.254	**0.006**	0.017	0.036
		[0.974, 0.997]	[1.000, 1.000]	[1.000, 1.000]	[0.999, 1.000]	[0.964, 0.980]	[1.000, 1.000]	[0.999, 1.000]	[0.998, 1.000]	[0.219, 0.307]	[0.004, 0.012]	[0.006, 0.031]	[0.027, 0.051]
	128	0.993	**1.000**	**1.000**	**1.000**	0.882	**1.000**	**1.000**	0.998	0.490	**0.024**	0.030	0.057
		[0.971, 0.995]	[1.000, 1.000]	[1.000, 1.000]	[0.999, 1.000]	[0.804, 0.932]	[0.999, 1.000]	[0.999, 1.000]	[0.995, 0.999]	[0.447, 0.549]	[0.020, 0.033]	[0.022, 0.051]	[0.042, 0.085]
	256	0.990	**1.000**	**1.000**	**1.000**	0.545	**0.997**	**0.997**	0.994	0.830	**0.097**	**0.099**	0.118
		[0.975, 0.992]	[1.000, 1.000]	[1.000, 1.000]	[0.999, 1.000]	[0.428, 0.717]	[0.995, 0.998]	[0.996, 0.998]	[0.989, 0.997]	[0.703, 0.949]	[0.063, 0.114]	[0.063, 0.113]	[0.089, 0.146]
	512	0.990	0.966	**1.000**	**1.000**	0.234	0.424	**0.986**	0.978	1.223	1.984	**0.203**	0.234
		[0.981, 0.992]	[0.817, 0.993]	[1.000, 1.000]	[0.999, 1.000]	[0.155, 0.431]	[0.228, 0.787]	[0.978, 0.990]	[0.965, 0.986]	[0.927, 1.480]	[0.851, 4.085]	[0.147, 0.275]	[0.177, 0.309]
	1024	0.994	-0.610	**1.000**	**1.000**	0.184	0.067	**0.945**	0.925	1.485	18.008	**0.373**	0.417
		[0.990, 0.995]	[-7.243, 0.635]	[1.000, 1.000]	[0.999, 1.000]	[0.116, 0.379]	[0.028, 0.111]	[0.918, 0.962]	[0.890, 0.945]	[1.056, 1.708]	[8.654, 40.741]	[0.305, 0.485]	[0.341, 0.581]
Burgers 1D	32	**0.940**	**0.946**	**0.941**	**0.939**	0.998	**1.000**	0.999	0.997	0.081	**0.025**	0.050	0.078
		[0.885, 0.982]	[0.886, 0.987]	[0.880, 0.987]	[0.873, 0.979]	[0.996, 0.999]	[0.999, 1.000]	[0.997, 1.000]	[0.995, 0.999]	[0.056, 0.105]	[0.008, 0.044]	[0.023, 0.095]	[0.061, 0.113]
	64	0.924	**0.957**	**0.956**	**0.952**	0.979	**0.999**	**0.999**	0.995	0.256	**0.049**	**0.043**	0.100
		[0.876, 0.978]	[0.900, 0.991]	[0.895, 0.991]	[0.887, 0.987]	[0.968, 0.985]	[0.995, 1.000]	[0.997, 1.000]	[0.991, 0.997]	[0.197, 0.314]	[0.016, 0.097]	[0.018, 0.090]	[0.073, 0.158]
	128	0.932	**0.966**	**0.965**	**0.960**	0.900	0.992	**0.999**	0.992	0.488	0.118	**0.046**	0.129
		[0.807, 0.980]	[0.928, 0.995]	[0.928, 0.995]	[0.906, 0.989]	[0.813, 0.940]	[0.973, 1.000]	[0.997, 1.000]	[0.985, 0.996]	[0.411, 0.554]	[0.043, 0.205]	[0.026, 0.081]	[0.094, 0.198]
	256	0.944	**0.971**	**0.971**	**0.966**	0.596	0.946	**0.997**	0.985	0.805	0.317	**0.094**	0.195
		[0.834, 0.979]	[0.936, 0.996]	[0.941, 0.997]	[0.919, 0.991]	[0.373, 0.814]	[0.847, 0.997]	[0.995, 0.998]	[0.971, 0.992]	[0.684, 0.964]	[0.122, 0.574]	[0.064, 0.122]	[0.126, 0.263]
	512	0.951	0.343	**0.980**	**0.969**	0.268	0.119	**0.988**	0.961	1.169	7.788	**0.168**	0.310
		[0.866, 0.982]	[-2.848, 0.875]	[0.949, 0.998]	[0.930, 0.992]	[0.161, 0.548]	[0.041, 0.374]	[0.985, 0.990]	[0.936, 0.976]	[0.848, 1.425]	[2.454, 21.043]	[0.133, 0.267]	[0.212, 0.449]
	1024	0.960	-2.497	**0.985**	0.970	0.240	0.050	**0.958**	0.900	1.357	28.083	**0.317**	0.480
		[0.891, 0.988]	[-13.453, -0.042]	[0.956, 0.998]	[0.937, 0.993]	[0.123, 0.554]	[0.023, 0.108]	[0.940, 0.968]	[0.848, 0.930]	[0.865, 1.642]	[12.454, 53.720]	[0.255, 0.481]	[0.363, 0.724]
Allen–Cahn 1D	32	**0.997**	**0.997**	**0.997**	**0.997**	0.999	**1.000**	0.998	0.999	0.044	**0.010**	0.058	0.036
		[0.996, 0.999]	[0.996, 0.999]	[0.996, 0.998]	[0.996, 0.998]	[0.998, 0.999]	[1.000, 1.000]	[0.997, 0.999]	[0.999, 1.000]	[0.033, 0.051]	[0.006, 0.012]	[0.040, 0.069]	[0.027, 0.051]
	64	0.995	**0.998**	0.997	0.997	0.974	**1.000**	0.994	0.997	0.208	**0.026**	0.101	0.077
		[0.995, 0.997]	[0.997, 0.999]	[0.996, 0.999]	[0.997, 0.998]	[0.966, 0.986]	[0.999, 1.000]	[0.992, 0.998]	[0.996, 0.999]	[0.159, 0.234]	[0.016, 0.032]	[0.065, 0.117]	[0.056, 0.096]
	128	0.994	**0.998**	0.997	**0.998**	0.893	**0.997**	0.987	0.996	0.416	**0.072**	0.155	0.090
		[0.993, 0.995]	[0.997, 0.999]	[0.997, 0.999]	[0.997, 0.999]	[0.871, 0.922]	[0.996, 0.999]	[0.982, 0.995]	[0.988, 0.998]	[0.370, 0.454]	[0.042, 0.083]	[0.100, 0.179]	[0.058, 0.145]
	256	0.992	**0.998**	**0.998**	0.997	0.700	**0.980**	0.977	0.952	0.718	**0.191**	0.220	0.306
		[0.991, 0.993]	[0.998, 0.999]	[0.997, 0.999]	[0.997, 0.998]	[0.638, 0.738]	[0.974, 0.991]	[0.967, 0.991]	[0.912, 0.966]	[0.678, 0.775]	[0.130, 0.214]	[0.148, 0.257]	[0.261, 0.384]
	512	0.992	0.874	**0.998**	0.997	0.481	0.225	**0.966**	0.827	1.020	4.344	**0.292**	0.542
		[0.991, 0.993]	[0.540, 0.936]	[0.998, 0.999]	[0.996, 0.998]	[0.368, 0.545]	[0.112, 0.307]	[0.949, 0.983]	[0.778, 0.900]	[0.957, 1.138]	[3.133, 8.623]	[0.216, 0.343]	[0.419, 0.592]
	1024	0.995	0.108	**0.998**	0.997	0.621	0.071	**0.953**	0.654	0.944	15.427	**0.375**	0.731
		[0.994, 0.995]	[-4.202, 0.672]	[0.998, 0.999]	[0.996, 0.998]	[0.488, 0.698]	[0.021, 0.109]	[0.930, 0.966]	[0.612, 0.768]	[0.891, 1.045]	[9.121, 33.762]	[0.305, 0.430]	[0.618, 0.760]
Diff.–Sorp. 1D	32	0.972	**0.985**	**0.985**	0.926	0.998	**1.000**	**1.000**	0.960	0.086	**0.005**	0.029	0.278
		[0.933, 0.998]	[0.962, 0.999]	[0.961, 0.999]	[0.486, 0.991]	[0.991, 0.999]	[1.000, 1.000]	[1.000, 1.000]	[0.824, 0.995]	[0.062, 0.199]	[0.004, 0.012]	[0.015, 0.033]	[0.098, 0.608]
	64	0.964	**0.991**	**0.990**	0.947	0.979	**1.000**	**1.000**	0.945	0.270	**0.014**	0.038	0.327
		[0.822, 0.989]	[0.975, 0.999]	[0.975, 0.999]	[0.280, 0.989]	[0.965, 0.989]	[1.000, 1.000]	[1.000, 1.000]	[0.727, 0.988]	[0.199, 0.397]	[0.009, 0.032]	[0.014, 0.043]	[0.160, 0.773]
	128	0.927	**0.997**	**0.997**	0.968	0.920	**0.999**	**0.999**	0.931	0.479	**0.037**	0.047	0.373
		[0.708, 0.983]	[0.987, 1.000]	[0.987, 1.000]	[0.126, 0.993]	[0.852, 0.950]	[0.999, 1.000]	[0.999, 1.000]	[0.657, 0.991]	[0.393, 0.671]	[0.026, 0.078]	[0.019, 0.084]	[0.138, 0.896]
	256	0.901	**0.998**	**0.998**	0.983	0.769	0.996	**0.998**	0.913	0.707	**0.092**	**0.090**	0.444
		[0.737, 0.980]	[0.982, 1.000]	[0.983, 1.000]	[0.388, 0.996]	[0.672, 0.835]	[0.994, 0.999]	[0.995, 0.999]	[0.607, 0.963]	[0.578, 0.777]	[0.063, 0.187]	[0.041, 0.188]	[0.266, 1.024]
	512	0.923	0.746	**0.998**	0.991	0.521	0.569	**0.994**	0.822	0.875	1.856	**0.176**	0.582
		[0.801, 0.982]	[-0.381, 0.955]	[0.983, 1.000]	[0.645, 0.996]	[0.410, 0.594]	[0.263, 0.848]	[0.980, 0.997]	[0.552, 0.924]	[0.811, 0.929]	[0.809, 3.692]	[0.097, 0.437]	[0.457, 1.251]
	1024	0.956	-0.164	**0.999**	0.994	0.403	0.174	**0.971**	0.704	0.985	7.074	**0.342**	0.766
		[0.862, 0.989]	[-5.381, 0.562]	[0.973, 1.000]	[0.782, 0.997]	[0.340, 0.578]	[0.086, 0.287]	[0.923, 0.982]	[0.446, 0.897]	[0.849, 1.093]	[4.726, 14.885]	[0.260, 1.076]	[0.552, 1.866]
Advection 1D	32	**0.912**	**0.912**	**0.895**	**0.898**	**0.995**	**0.995**	0.954	0.968	0.112	**0.095**	0.308	0.251
		[0.866, 0.947]	[0.865, 0.947]	[0.764, 0.926]	[0.797, 0.931]	[0.991, 0.997]	[0.992, 0.998]	[0.879, 0.980]	[0.930, 0.984]	[0.083, 0.139]	[0.069, 0.126]	[0.222, 0.488]	[0.179, 0.389]
	64	**0.937**	**0.944**	**0.926**	**0.928**	0.974	**0.982**	0.934	0.953	0.253	**0.182**	0.381	0.308
		[0.880, 0.958]	[0.887, 0.966]	[0.800, 0.952]	[0.821, 0.951]	[0.958, 0.981]	[0.972, 0.991]	[0.828, 0.977]	[0.890, 0.972]	[0.216, 0.316]	[0.129, 0.228]	[0.242, 0.610]	[0.229, 0.514]
	128	**0.948**	**0.957**	**0.946**	**0.953**	0.927	**0.943**	0.914	**0.940**	0.423	**0.331**	0.439	0.375
		[0.893, 0.966]	[0.898, 0.972]	[0.846, 0.970]	[0.868, 0.969]	[0.901, 0.955]	[0.912, 0.971]	[0.769, 0.971]	[0.886, 0.968]	[0.349, 0.468]	[0.234, 0.444]	[0.269, 0.805]	[0.258, 0.556]
	256	**0.955**	**0.955**	**0.961**	**0.962**	0.795	0.814	**0.903**	**0.931**	0.591	0.668	0.506	**0.397**
		[0.914, 0.974]	[0.895, 0.977]	[0.871, 0.979]	[0.890, 0.979]	[0.688, 0.926]	[0.712, 0.901]	[0.693, 0.966]	[0.865, 0.966]	[0.436, 0.693]	[0.455, 0.903]	[0.285, 0.927]	[0.295, 0.554]
	512	**0.962**	-2.277	**0.966**	**0.960**	0.663	0.077	**0.894**	**0.841**	0.773	10.774	**0.540**	**0.600**
		[0.926, 0.977]	[-16.345, -0.098]	[0.886, 0.985]	[0.886, 0.982]	[0.370, 0.911]	[0.032, 0.129]	[0.678, 0.964]	[0.748, 0.906]	[0.458, 0.990]	[6.883, 24.086]	[0.311, 0.965]	[0.461, 0.710]
	1024	**0.971**	-8.525	**0.970**	**0.963**	0.705	0.045	**0.874**	0.641	0.758	24.505	**0.604**	0.799
		[0.943, 0.983]	[-39.101, -1.702]	[0.902, 0.987]	[0.892, 0.984]	[0.346, 0.946]	[0.011, 0.080]	[0.610, 0.974]	[0.575, 0.780]	[0.370, 1.056]	[12.215, 46.594]	[0.340, 1.053]	[0.634, 0.859]

2D). For $n = 1024$, it is the finest resolution, so the explainer has to extrapolate, and on Heat 2D its DISCO branch is sized for 16×16 instead of 32×32. Everything else, including the backbone, the references, and the training recipe, is the same as for the full explainers. On Burgers 1D, both held-out explainers reach the quality of the full explainer. On Heat 2D, the loss is small at $n = 256$ and large at $n = 1024$.

D.3 Analytic validation

Fig. 9 compares the explainers with the closed-form Aumann–Shapley densities of Heat 1D and Advection 1D from paragraph (iv) of Sec. 6.

Table 4: Full-suite version of Tab. 1: Cross-PDE attribution quality across resolutions n on the 2D and 3D PDEs and the MGN meshes (other half in Tab. 3). Each (PDE, n, method) entry reports the median over 30 test inputs and three query points per input on the top line and the inter-quartile range in brackets below. Metrics are Pearson correlation and NRMSE against the per-PDE reference attribution, and the weighted-R^2 Faithfulness score. RegressionMSR and KernelSHAP use a budget of $2n$ (minimum 1024) backbone evaluations per query point, and IG is calculated via a 50-step Riemann sum. Across resolutions OperatorSHAP uses the same model per PDE. Higher Pearson and R^2 are better, and lower NRMSE is better. Heat 3D is evaluated at $n = 512$ only, and the MGN meshes use 10 test inputs. Bold marks the best median and every median within one standard error of it over the test inputs, compared at three decimals.

		R^2 ↑				Pearson ↑				NRMSE ↓			
PDE	n	RMSR	KS	IG	OS (ours)	RMSR	KS	IG	OS (ours)	RMSR	KS	IG	OS (ours)
Heat 2D	64	0.987 [0.952, 0.995]	**1.000** [1.000, 1.000]	**1.000** [1.000, 1.000]	**1.000** [0.999, 1.000]	0.963 [0.957, 0.973]	**1.000** [1.000, 1.000]	**1.000** [1.000, 1.000]	0.999 [0.999, 1.000]	0.314 [0.269, 0.346]	**0.009** [0.007, 0.011]	**0.010** [0.008, 0.013]	0.034 [0.028, 0.040]
	256	0.988 [0.958, 0.992]	**1.000** [1.000, 1.000]	**1.000** [1.000, 1.000]	**1.000** [1.000, 1.000]	0.666 [0.522, 0.783]	**0.994** [0.992, 0.996]	**0.994** [0.992, 0.996]	**0.993** [0.990, 0.995]	0.721 [0.664, 0.851]	**0.117** [0.092, 0.189]	**0.117** [0.091, 0.190]	0.129 [0.102, 0.197]
	1024	0.993 [0.984, 0.995]	-0.023 [-6.136, 0.773]	**1.000** [1.000, 1.000]	**1.000** [1.000, 1.000]	0.198 [0.136, 0.377]	0.082 [0.038, 0.153]	**0.933** [0.922, 0.943]	0.929 [0.919, 0.943]	1.298 [0.966, 1.558]	13.740 [6.950, 31.544]	**0.394** [0.336, 0.553]	**0.391** [0.337, 0.539]
Darcy 2D	64	0.854 [0.771, 0.912]	**0.915** [0.848, 0.943]	0.904 [0.835, 0.935]	0.904 [0.831, 0.934]	0.977 [0.967, 0.984]	**0.999** [0.999, 1.000]	0.993 [0.986, 0.996]	0.993 [0.988, 0.995]	0.304 [0.222, 0.366]	**0.041** [0.029, 0.055]	0.155 [0.125, 0.179]	0.124 [0.099, 0.154]
	256	0.895 [0.726, 0.932]	**0.942** [0.892, 0.969]	**0.941** [0.887, 0.968]	**0.941** [0.886, 0.968]	0.876 [0.788, 0.902]	**0.986** [0.975, 0.990]	**0.987** [0.984, 0.991]	0.982 [0.974, 0.986]	0.561 [0.497, 0.700]	**0.216** [0.180, 0.281]	0.246 [0.201, 0.294]	**0.216** [0.181, 0.255]
	1024	0.931 [0.871, 0.960]	-6.410 [-23.831, -1.521]	**0.951** [0.897, 0.980]	**0.950** [0.901, 0.980]	0.736 [0.567, 0.815]	0.084 [0.045, 0.166]	**0.957** [0.935, 0.968]	0.946 [0.923, 0.959]	0.694 [0.611, 0.830]	13.905 [7.124, 31.089]	0.485 [0.366, 0.678]	**0.444** [0.321, 0.552]
Reac.–Diff. 2D	64	**0.230** [-0.003, 0.323]	0.211 [-0.008, 0.330]	-1.252 [-3.417, 0.138]	**0.246** [0.023, 0.355]	0.944 [0.927, 0.954]	0.922 [0.902, 0.938]	0.733 [0.680, 0.794]	**0.980** [0.969, 0.990]	0.364 [0.320, 0.407]	0.422 [0.378, 0.483]	2.564 [0.881, 4.700]	**0.208** [0.154, 0.254]
	256	0.210 [0.104, 0.329]	-0.083 [-0.355, 0.168]	-0.101 [-0.651, 0.256]	**0.292** [0.174, 0.426]	0.790 [0.746, 0.841]	0.663 [0.585, 0.741]	0.875 [0.817, 0.907]	**0.932** [0.910, 0.959]	0.642 [0.574, 0.692]	1.392 [1.120, 1.973]	1.457 [0.915, 2.375]	**0.386** [0.313, 0.459]
	1024	0.456 [0.240, 0.576]	-252.302 [-1119.431, -69.746]	0.350 [0.013, 0.530]	**0.527** [0.301, 0.626]	0.804 [0.719, 0.853]	0.048 [0.012, 0.083]	0.789 [0.683, 0.859]	**0.847** [0.804, 0.900]	**0.628** [0.546, 0.714]	51.625 [25.210, 103.550]	1.501 [1.184, 1.907]	**0.630** [0.523, 0.740]
Shallow Water 2D	64	0.499 [0.408, 0.600]	0.550 [0.456, 0.624]	0.421 [0.247, 0.476]	**0.570** [0.479, 0.644]	0.949 [0.940, 0.957]	0.970 [0.961, 0.976]	0.729 [0.620, 0.810]	**0.996** [0.995, 0.997]	0.416 [0.372, 0.440]	0.258 [0.226, 0.291]	0.731 [0.617, 0.855]	**0.093** [0.085, 0.106]
	256	0.803 [0.745, 0.874]	0.864 [0.824, 0.892]	0.869 [0.825, 0.897]	**0.881** [0.841, 0.910]	0.850 [0.834, 0.899]	0.927 [0.906, 0.936]	0.915 [0.892, 0.927]	**0.964** [0.951, 0.972]	0.575 [0.451, 0.606]	0.482 [0.451, 0.522]	0.455 [0.425, 0.497]	**0.311** [0.288, 0.364]
	1024	0.951 [0.929, 0.965]	-3.861 [-23.325, -0.627]	**0.980** [0.971, 0.986]	0.974 [0.964, 0.982]	0.910 [0.880, 0.932]	0.136 [0.050, 0.216]	0.898 [0.879, 0.921]	**0.923** [0.898, 0.939]	**0.416** [0.363, 0.477]	9.197 [5.292, 22.002]	0.529 [0.433, 0.637]	0.527 [0.454, 0.598]
Heat 3D	512	0.980 [0.940, 0.988]	0.974 [0.659, 0.995]	**1.000** [1.000, 1.000]	0.996 [0.986, 0.998]	0.531 [0.378, 0.679]	0.726 [0.329, 0.926]	**0.979** [0.975, 0.983]	0.906 [0.896, 0.924]	0.854 [0.747, 0.994]	1.036 [0.516, 2.981]	**0.255** [0.204, 0.328]	0.432 [0.403, 0.484]
MGN Cylinder	64	**0.921** [0.824, 0.963]	**0.923** [0.827, 0.963]	0.900 [0.775, 0.943]	**0.910** [0.783, 0.955]	0.985 [0.977, 0.992]	**0.991** [0.984, 0.993]	0.916 [0.830, 0.951]	0.946 [0.890, 0.983]	0.181 [0.138, 0.235]	**0.130** [0.116, 0.170]	0.415 [0.338, 0.558]	0.315 [0.185, 0.472]
MGN Airfoil	64	**0.822** [0.478, 0.879]	**0.822** [0.470, 0.886]	0.727 [0.232, 0.817]	0.751 [0.444, 0.869]	**0.970** [0.956, 0.981]	**0.977** [0.954, 0.984]	0.727 [0.593, 0.848]	0.737 [0.580, 0.875]	0.255 [0.209, 0.317]	**0.212** [0.182, 0.290]	1.004 [0.811, 1.363]	0.657 [0.465, 0.759]

Table 5: OperatorSHAP trained on all resolutions (full ladder) or with $n = 256$ or $n = 1024$ left out, evaluated at the left-out resolution. Medians over 30 test inputs and three query points with the inter-quartile range below. The classical methods and IG do not depend on the explainer and are repeated for reference. Bold marks the best median and every median within one standard error of it over the test inputs, compared at three decimals.

		R^2 ↑				Pearson ↑				NRMSE ↓			
Setting	n	RMSR	KS	IG	OS (ours)	RMSR	KS	IG	OS (ours)	RMSR	KS	IG	OS (ours)
Burgers 1D, full ladder	256	0.944 [0.834, 0.979]	**0.971** [0.936, 0.996]	**0.971** [0.941, 0.997]	**0.966** [0.919, 0.991]	0.596 [0.373, 0.814]	0.946 [0.847, 0.997]	**0.997** [0.995, 0.998]	0.985 [0.971, 0.992]	0.805 [0.684, 0.964]	0.317 [0.122, 0.574]	**0.094** [0.064, 0.122]	0.195 [0.126, 0.263]
	1024	0.960 [0.891, 0.988]	-2.497 [-13.453, -0.042]	**0.985** [0.956, 0.998]	0.970 [0.937, 0.993]	0.240 [0.123, 0.554]	0.050 [0.023, 0.108]	**0.958** [0.940, 0.968]	0.900 [0.848, 0.930]	1.357 [0.865, 1.642]	28.083 [12.454, 53.720]	**0.317** [0.255, 0.481]	0.480 [0.363, 0.724]
Burgers 1D, $n = 256$ held out	256	0.944 [0.834, 0.979]	**0.971** [0.936, 0.996]	**0.971** [0.941, 0.997]	**0.965** [0.919, 0.991]	0.596 [0.373, 0.814]	0.946 [0.847, 0.997]	**0.997** [0.995, 0.998]	0.985 [0.973, 0.991]	0.805 [0.684, 0.964]	0.317 [0.122, 0.574]	**0.094** [0.064, 0.122]	0.195 [0.129, 0.260]
Burgers 1D, $n = 1024$ held out	1024	0.960 [0.891, 0.988]	-2.497 [-13.453, -0.042]	**0.985** [0.956, 0.998]	0.970 [0.937, 0.993]	0.240 [0.123, 0.554]	0.050 [0.023, 0.108]	**0.958** [0.940, 0.968]	0.900 [0.845, 0.930]	1.357 [0.865, 1.642]	28.083 [12.454, 53.720]	**0.317** [0.255, 0.481]	0.481 [0.361, 0.723]
Heat 2D, full ladder	256	0.988 [0.958, 0.992]	**1.000** [1.000, 1.000]	**1.000** [1.000, 1.000]	**1.000** [1.000, 1.000]	0.666 [0.522, 0.783]	**0.994** [0.992, 0.996]	**0.994** [0.992, 0.996]	**0.993** [0.990, 0.995]	0.721 [0.664, 0.851]	**0.117** [0.092, 0.189]	**0.117** [0.091, 0.190]	0.129 [0.102, 0.197]
	1024	0.993 [0.984, 0.995]	-0.023 [-6.136, 0.773]	**1.000** [1.000, 1.000]	**1.000** [1.000, 1.000]	0.198 [0.136, 0.377]	0.082 [0.038, 0.153]	**0.933** [0.922, 0.943]	0.929 [0.919, 0.943]	1.298 [0.966, 1.558]	13.740 [6.950, 31.544]	**0.394** [0.336, 0.553]	**0.391** [0.337, 0.539]
Heat 2D, $n = 256$ held out	256	0.988 [0.958, 0.992]	**1.000** [1.000, 1.000]	**1.000** [1.000, 1.000]	0.999 [0.996, 1.000]	0.666 [0.522, 0.783]	**0.994** [0.992, 0.996]	**0.994** [0.992, 0.996]	0.980 [0.971, 0.983]	0.721 [0.664, 0.851]	**0.117** [0.092, 0.189]	**0.117** [0.091, 0.190]	0.203 [0.173, 0.239]
Heat 2D, $n = 1024$ held out	1024	0.993 [0.984, 0.995]	-0.023 [-6.136, 0.773]	**1.000** [1.000, 1.000]	0.998 [0.991, 0.999]	0.198 [0.136, 0.377]	0.082 [0.038, 0.153]	**0.933** [0.922, 0.943]	0.793 [0.749, 0.825]	1.298 [0.966, 1.558]	13.740 [6.950, 31.544]	**0.394** [0.336, 0.553]	0.763 [0.603, 1.030]

D.4 Baseline choice

Choosing the correct baseline is already a difficult task in the classical Shapley literature. The baseline influences the interpretation of the attributions both by changing the reference point and by affecting the backbone model. The latter is particularly important in the context of physical data. Where a zero baseline in the heat equation just represents a state of no heat, in the Darcy flow, a zero permeability obstructs the flow of fluid through a masked cell. The choice of baseline should be determined on a case-by-case basis. We use the following:

- **Zero.** Masked cells are set to zero, so the attribution of a region is its contribution compared to the region being absent. This is the easiest to interpret and our default. However, the

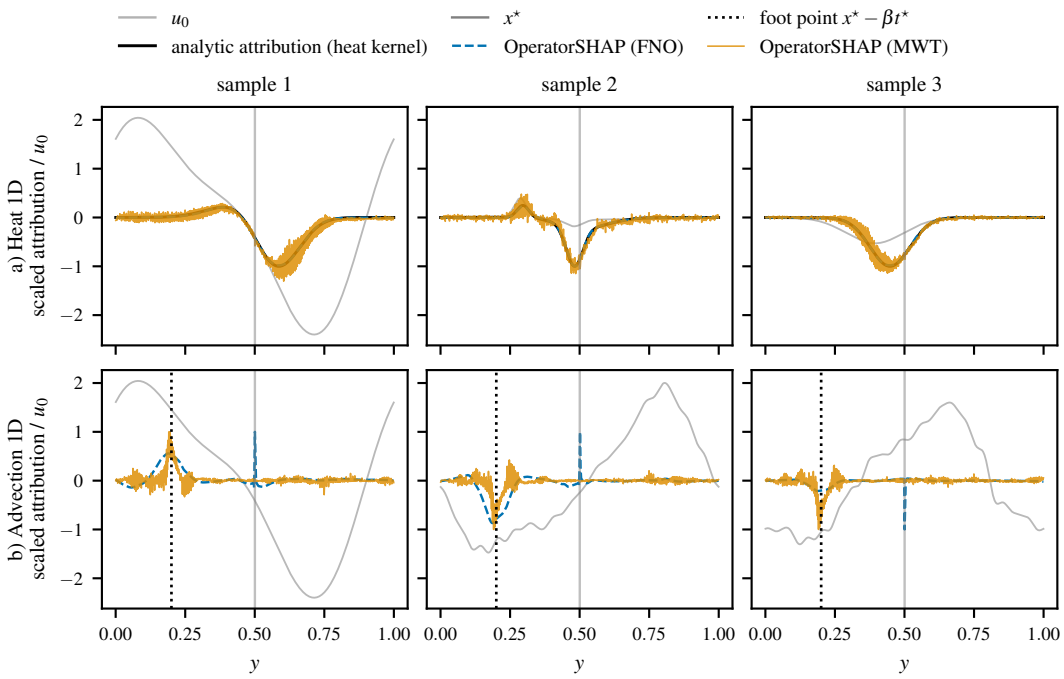


Figure 9: Analytic validation on three test samples. *Top:* Heat 1D. The closed-form density $K_{t^\star}(x^\star, \cdot)\, u_0$ (black) and the FNO- and MWT-based explainers. *Bottom:* Advection 1D. The analytic density is a point mass at the foot point $x^\star - \beta t^\star$ (dotted). Both explainers place mass there, but the largest value of the FNO explainer is a spike at $x^\star$ (gray), which exposes the pointwise skip connections of the FNO backbone. In the top row, the attributions are divided by the maximum of the analytic density, and in the bottom row each explainer by its own maximum. u_0 is not scaled.

masked input can be physically implausible: in Darcy flow, zero permeability makes a masked cell impermeable, and zero velocity next to a wall does not match the boundary condition seen in training.

- **Per-sample mean.** Masked cells are set to the spatial mean of the sample. The attribution then measures the effect of the deviation from this mean. The fully masked input is the constant mean and keeps the total mass of the sample, which the zero baseline does not. For a partial coalition the mass changes by the integral of $u_0 - \bar{u}$ over the coalition. For fields with a spatial mean close to zero (heat, Burgers) this baseline is almost the same as the zero baseline.
- **Global and per-node mean.** Masked cells are set to the mean of the training set, either one value for the whole field or one value per node for meshes with a fixed node set. The masked input is then a typical field that satisfies the boundary conditions and the geometry. We use these for the MGN datasets.
- **Conditional expectation.** Masked cells could also be set to their expected value given the revealed cells. This needs a generative model of the inputs, and we do not use it.

Guidelines: (1) Report the baseline together with the attribution, because different baselines answer different questions and the resulting maps need not be correlated. (2) Use a mean baseline for coefficient fields whose zero value is not admissible (permeability, density) and near boundaries where a zero value violates the boundary condition.

Tab. 6 compares the zero baseline with the per-sample mean on Darcy 2D and diffusion–sorption 1D, each with its own explainer and reference. We can see that the choice of baseline does not influence the reported faithfulness meaningfully. Fig. 10 compares the attributions of one Darcy 2D sample under both baselines, which shows grave differences that a user would need to reflect in their interpretation.

Table 6: Zero and per-sample mean baseline on Darcy 2D and diffusion–sorption 1D. Each baseline has its own explainer and reference. Medians over 30 test inputs and three query points with the inter-quartile range below. Bold marks the best median and every median within one standard error of it over the test inputs, compared at three decimals.

		$R^2 \uparrow$				Pearson $\uparrow$				NRMSE $\downarrow$			
Baseline	n	RMSR	KS	IG	OS (ours)	RMSR	KS	IG	OS (ours)	RMSR	KS	IG	OS (ours)
Darcy 2D, zero	64	0.854 [0.771, 0.912]	**0.915** [0.848, 0.943]	0.904 [0.835, 0.935]	0.904 [0.831, 0.934]	0.977 [0.967, 0.984]	**0.999** [0.999, 1.000]	0.993 [0.986, 0.996]	0.993 [0.988, 0.995]	0.304 [0.222, 0.366]	**0.041** [0.029, 0.055]	0.155 [0.125, 0.179]	0.124 [0.099, 0.154]
	1024	0.931 [0.871, 0.960]	-6.410 [-23.831, -1.521]	**0.951** [0.897, 0.980]	**0.950** [0.901, 0.980]	0.736 [0.567, 0.815]	0.084 [0.045, 0.166]	**0.957** [0.935, 0.968]	0.946 [0.923, 0.959]	0.694 [0.611, 0.830]	13.905 [7.124, 31.089]	0.485 [0.366, 0.678]	**0.444** [0.321, 0.552]
Darcy 2D, mean	64	0.962 [0.893, 0.978]	**0.975** [0.941, 0.990]	**0.973** [0.938, 0.990]	0.963 [0.907, 0.981]	0.966 [0.951, 0.977]	**1.000** [1.000, 1.000]	0.998 [0.996, 0.999]	0.974 [0.932, 0.984]	0.282 [0.232, 0.380]	**0.016** [0.012, 0.022]	0.059 [0.048, 0.084]	0.229 [0.181, 0.356]
	1024	0.976 [0.926, 0.988]	-1.143 [-15.560, 0.544]	**0.987** [0.948, 0.996]	**0.985** [0.940, 0.995]	0.364 [0.215, 0.537]	0.075 [0.036, 0.153]	**0.955** [0.933, 0.968]	0.855 [0.787, 0.928]	1.045 [0.842, 1.204]	14.531 [6.885, 39.518]	**0.345** [0.279, 0.478]	0.549 [0.395, 0.668]
Diff.–Sorp. 1D, zero	64	0.964 [0.822, 0.989]	**0.991** [0.975, 0.999]	**0.990** [0.975, 0.999]	0.947 [0.280, 0.989]	0.979 [0.965, 0.989]	**1.000** [1.000, 1.000]	**1.000** [1.000, 1.000]	0.945 [0.727, 0.988]	0.270 [0.199, 0.397]	**0.014** [0.009, 0.032]	0.038 [0.014, 0.043]	0.327 [0.160, 0.773]
	1024	0.956 [0.862, 0.989]	-0.164 [-5.381, 0.562]	**0.999** [0.973, 1.000]	0.994 [0.782, 0.997]	0.403 [0.340, 0.578]	0.174 [0.086, 0.287]	**0.971** [0.923, 0.982]	0.704 [0.446, 0.897]	0.985 [0.849, 1.093]	7.074 [4.726, 14.885]	**0.342** [0.260, 1.076]	0.766 [0.552, 1.866]
Diff.–Sorp. 1D, mean	64	0.977 [0.973, 0.987]	**0.999** [0.997, 1.000]	**0.999** [0.997, 1.000]	0.996 [0.992, 0.997]	0.974 [0.968, 0.982]	**1.000** [1.000, 1.000]	**1.000** [1.000, 1.000]	0.996 [0.994, 0.998]	0.271 [0.204, 0.320]	**0.009** [0.006, 0.010]	0.010 [0.006, 0.012]	0.089 [0.076, 0.117]
	1024	0.995 [0.992, 0.997]	-3.924 [-19.506, -0.645]	**1.000** [0.999, 1.000]	0.915 [0.904, 0.932]	0.991 [0.947, 0.994]	0.131 [0.067, 0.267]	**0.995** [0.985, 0.998]	0.571 [0.536, 0.595]	0.137 [0.106, 0.339]	6.925 [4.154, 15.157]	**0.102** [0.063, 0.185]	0.838 [0.824, 0.858]

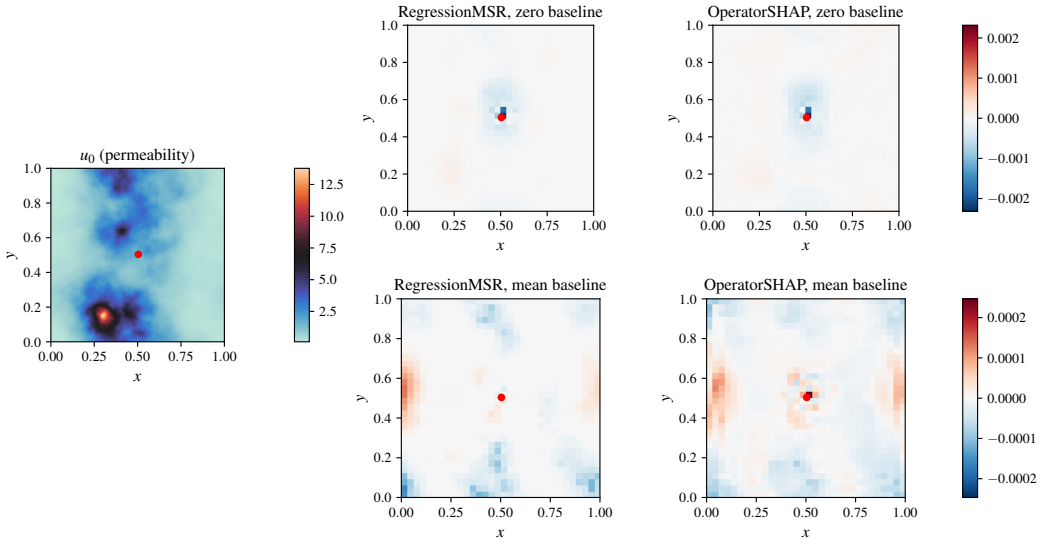


Figure 10: Baseline ablation on one Darcy 2D sample at $n = 1024$ and the domain center $x^\star$ (red): the permeability field, and the reference (RegressionMSR with $3 \cdot 10^5$ backbone evaluations) and OperatorSHAP attributions with the zero baseline (*top*) and the per-sample mean baseline (*bottom*). With the zero baseline the contributions concentrate in the cells around $x^\star$ that gate the flow into it. With the mean baseline the whole field contributes to the game, as there now exists non-zero permeability. The two references are uncorrelated, with median Pearson -0.24 on Darcy 2D and -0.02 on diffusion–sorption 1D at $n = 1024$.

D.5 ReLU backbone

Proposition 6 requires a smooth activation. Tab. 7 repeats the Burgers 1D experiment with an FNO backbone in which every GELU is replaced by a ReLU, with the same architecture, training recipe, and its own reference. The explainer reaches a similar quality as for the GELU backbone (Pearson 0.995 and 0.990 against 0.995 and 0.985 at $n = 64$ and 256), so the smoothness assumption is a requirement of the differentiability proof and not of the algorithm.

D.6 Non-differentiable readout

In decision problems, the precise output value matters less than whether the prediction at $x^\star$ reaches a given outcome. We model this with the midpoint readout. With the masked input $u_S = \bar{u} + \mathbf{1}_S \odot (u_0 - \bar{u})$, the value of a coalition S is

$$v(S) = \mathbf{1}\Big[\,\big|\hat{\mathcal{F}}_{x^\star}(u_S) - \hat{\mathcal{F}}_{x^\star}(u_0)\big| < \big|\hat{\mathcal{F}}_{x^\star}(u_S) - \hat{\mathcal{F}}_{x^\star}(\bar{u})\big|\,\Big],$$

Table 7: Burgers 1D with a ReLU and a GELU backbone. Medians over 30 test inputs and three query points with the inter-quartile range below, with the same methods and budgets as Tab. 1. Bold marks the best median and every median within one standard error of it over the test inputs, compared at three decimals.

		R^2 ↑				Pearson ↑				NRMSE ↓			
Activation	n	RMSR	KS	IG	OS (ours)	RMSR	KS	IG	OS (ours)	RMSR	KS	IG	OS (ours)
GELU	64	0.924	**0.957**	**0.956**	**0.952**	0.979	**0.999**	**0.999**	0.995	0.256	**0.049**	**0.043**	0.100
		[0.876, 0.978]	[0.900, 0.991]	[0.895, 0.991]	[0.887, 0.987]	[0.968, 0.985]	[0.995, 1.000]	[0.997, 1.000]	[0.991, 0.997]	[0.197, 0.314]	[0.016, 0.097]	[0.018, 0.090]	[0.073, 0.158]
	256	0.944	**0.971**	**0.971**	**0.966**	0.596	0.946	**0.997**	0.985	0.805	0.317	**0.094**	0.195
		[0.834, 0.979]	[0.936, 0.996]	[0.941, 0.997]	[0.919, 0.991]	[0.373, 0.814]	[0.847, 0.997]	[0.995, 0.998]	[0.971, 0.992]	[0.684, 0.964]	[0.122, 0.574]	[0.064, 0.122]	[0.126, 0.263]
ReLU	64	0.918	**0.963**	**0.964**	**0.948**	0.977	**0.998**	**0.998**	**0.995**	0.262	**0.055**	**0.071**	0.099
		[0.871, 0.974]	[0.910, 0.988]	[0.901, 0.988]	[0.899, 0.984]	[0.966, 0.983]	[0.992, 1.000]	[0.995, 1.000]	[0.991, 0.997]	[0.204, 0.317]	[0.020, 0.119]	[0.036, 0.115]	[0.084, 0.173]
	256	0.937	**0.976**	**0.978**	0.965	0.561	0.936	**0.994**	0.990	0.820	0.329	**0.121**	0.174
		[0.848, 0.978]	[0.933, 0.992]	[0.935, 0.993]	[0.922, 0.991]	[0.396, 0.785]	[0.760, 0.995]	[0.990, 0.997]	[0.978, 0.995]	[0.689, 0.982]	[0.136, 0.833]	[0.094, 0.165]	[0.114, 0.265]

so $v(S) = 1$ if the prediction for the masked input is closer to the prediction for u_0 than to the prediction for the baseline. Then $v(\emptyset) = 0$, and $v(\Omega) = 1$ whenever $\hat{\mathcal{F}}_{x^\star}(u_0) \neq \hat{\mathcal{F}}_{x^\star}(\bar{u})$. By efficiency, the Shapley values sum to one and give the share of each region in moving the prediction past the midpoint between the two values.

The game is piecewise constant in the coalition weights. Along the straight path from $\bar{u}$ to u_0 its gradient vanishes almost everywhere, so IG returns zero attributions for every input, with R^2 between -1.590 and -0.535, an undefined Pearson correlation, and NRMSE 1 in Tab. 8. The game is of the type in Remark 8. It has a value, but we do not prove that the value is asymptotic, so we compare the explainer with a reference at every resolution instead.

We use the backbones and data of the nine 1D and 2D PDEs, Heat 3D, and the MGN meshes, train a new explainer per PDE on this game, and compute new references. The learning rate and batch size are chosen per PDE by validation loss: 10^{-3} and 64, except $3 \cdot 10^{-4}$ and 32 for Advection 1D. The MGN meshes use the training setup of the mesh explainers (App. C.4). From $n = 64$ on, OperatorSHAP has the highest faithfulness or one within one standard error of it, except on Advection 1D at $n = 1024$ and MGN Airfoil. On MGN Cylinder ($n = 64$), RegressionMSR has a marginally higher Pearson correlation (0.957 against 0.928), and the faithfulness of RegressionMSR, KernelSHAP, and OperatorSHAP lies within one standard error. On MGN Airfoil ($n = 64$), RegressionMSR and KernelSHAP have a higher Pearson correlation (0.911 and 0.827 against 0.630) and a higher faithfulness (0.757 and 0.749 against 0.694).

D.7 Structural stability across resolutions

Pearson and NRMSE compare attributions at one resolution. To test whether the spatial structure of a map is preserved when the same sample is served at two resolutions, we compute the structural similarity index (SSIM) between the two maps. Since the discrete values at every partition are integrals of one density, the finer map is average-pooled to the coarser grid, both maps are nearest-upsampled to a common 32×32 canvas, standardized, and compared with SSIM (7×7 Gaussian window). Tab. 9 reports the median over 30 test inputs on Heat 2D and Shallow Water 2D for OperatorSHAP, the reference with $3 \cdot 10^5$ backbone evaluations, and RegressionMSR at budget $2n$ (at least 1024). OperatorSHAP preserves structure at least as well as the reference, though this is dependent on the granularity and jump between discretizations. If the jump between the compared granularities is too large, the finer granularity has detail that the coarser one lost.

D.8 Backbone accuracy

Tab. 10 lists the relative L^2 test error of every grid backbone at every resolution at which it is explained. The columns give the grid points per axis $n_{1\mathrm{D}}$, so the player count n of the other tables is $n_{1\mathrm{D}}^d$. OperatorSHAP explains the backbone it is given, so these numbers do not influence the quality of the attribution. Explaining an imperfect backbone may be useful if the developer wants to understand the backbone's behavior. We include the numbers for completeness.

E Computing resources

The runs reported here were carried out on four academic clusters, depending on availability: NVIDIA A100 (80 GB), GH200, A40, and H100 GPUs with one GPU per job, and CPU nodes with two Intel Xeon Platinum 8380. Wall-clock times therefore depend on the node. The runtimes of Fig. 8 are measured on one thread of an Intel Xeon Platinum 8380 CPU. Medians over the reported runs: backbone training takes 0.1 h (1D), 0.4 h (2D), 0.3 h (3D), and 2.8 h (meshes) on an A100. OperatorSHAP training takes 3 to 5 h (1D), 6 to 8 h (2D), and 3 to 9 h (meshes). The reference runs with $3{\cdot}10^5$ backbone evaluations take about 1 h (1D), 1 to 2 h (2D), 3 h (3D), and 8.5 h (meshes) on one GPU, and RegressionMSR and KernelSHAP at budget $2n$ (at least 1024) under 0.5 h on one GPU or eight CPU cores. The metrics of one PDE and player number take under a minute on a GPU for the grids and up to 1.4 h for the meshes. The data-generation pipeline (App. C.2) is CPU-only and runs in under an hour per PDE on a single 96-core node thanks to the per-sample joblib parallelism. The full project required more compute, due to failed experiments, parameter tuning, or experiments that did not make it into the final paper.

F Third-party assets and licenses

Most of the code we build on–PyTorch, Hydra, joblib, scipy, the `neuraloperator` package–is standard scientific Python tooling. The two third-party assets that warrant explicit credit are the `shapiq` library [9], which we use for all classical Shapley estimators (EXACT, KERNELSHAP, REGRESSIONMSR) and which is released under the MIT license, and the MeshGraphNets airfoil and cylinder-flow datasets [31] produced and released by Google DeepMind under the Apache 2.0 license, which we use unmodified beyond the per-trajectory NPZ repackaging described in App. C.2. We use both within the bounds of their respective licenses and redistribute neither.

G Use of large language models

Large language models were used throughout the writing and engineering of this work: as a writing aid for editing and rephrasing parts of the manuscript, as a coding partner during implementation of the data-generation, training, and evaluation pipelines, and to discuss the mathematical arguments and the experimental design. Every theoretical statement, derivation, and final phrasing was checked by the authors, and every numerical result reported here was produced by the released code.

Table 8: Game with the non-differentiable readout of App. D.6 on the nine 1D and 2D PDEs, Heat 3D, and the MGN meshes. The reference is RegressionMSR over all 2^{16} coalitions at $n = 16$ and RegressionMSR with $3 \cdot 10^5$ backbone evaluations otherwise ($5 \cdot 10^4$ on MGN Airfoil). Medians over 30 test inputs and three query points with the inter-quartile range below, with the same methods and budgets as Tab. 1. The MGN meshes use 10 test inputs. On the decision readout, the gradient of the game vanishes, so IG attributions are identically zero and their Pearson correlation is undefined (–). Bold marks the best median and every median within one standard error of it over the test inputs, compared at three decimals.

		$R^2 \uparrow$				Pearson $\uparrow$				NRMSE $\downarrow$			
PDE	n	RMSR	KS	IG	OS (ours)	RMSR	KS	IG	OS (ours)	RMSR	KS	IG	OS (ours)
Heat 1D	16	**0.781** [0.774, 0.795]	**0.778** [0.767, 0.790]	-1.000 [-1.003, -0.994]	**0.775** [0.757, 0.785]	**0.999** [0.999, 1.000]	0.991 [0.984, 0.995]	–	0.986 [0.970, 0.991]	**0.040** [0.028, 0.050]	0.127 [0.098, 0.165]	1.000 [1.000, 1.000]	0.170 [0.136, 0.247]
	64	**0.818** [0.794, 0.824]	**0.799** [0.789, 0.805]	-0.999 [-1.004, -0.991]	**0.836** [0.825, 0.839]	0.757 [0.599, 0.914]	0.727 [0.595, 0.880]	–	**0.992** [0.988, 0.995]	0.611 [0.432, 0.750]	0.896 [0.507, 1.139]	1.000 [1.000, 1.000]	**0.141** [0.103, 0.199]
	256	**0.853** [0.835, 0.858]	0.667 [0.627, 0.689]	-1.000 [-1.003, -0.993]	**0.875** [0.872, 0.876]	0.131 [0.036, 0.305]	0.121 [0.066, 0.235]	–	**0.975** [0.967, 0.982]	1.522 [1.028, 1.837]	4.663 [2.710, 5.898]	1.000 [1.000, 1.000]	**0.231** [0.204, 0.308]
	1024	**0.886** [0.880, 0.889]	-98.764 [-471.047, -25.488]	-1.000 [-1.003, -0.993]	**0.899** [0.898, 0.899]	0.027 [-0.003, 0.083]	0.008 [-0.015, 0.024]	–	**0.799** [0.677, 0.912]	2.398 [1.537, 2.743]	165.848 [101.183, 363.842]	1.000 [1.000, 1.000]	**0.624** [0.544, 0.701]
Burgers 1D	16	**0.703** [0.640, 0.770]	**0.701** [0.636, 0.763]	-1.195 [-1.567, -1.001]	**0.693** [0.611, 0.753]	**0.999** [0.999, 1.000]	0.996 [0.992, 0.998]	–	0.983 [0.969, 0.993]	**0.036** [0.023, 0.049]	0.088 [0.067, 0.120]	1.000 [1.000, 1.000]	0.191 [0.131, 0.265]
	64	**0.765** [0.630, 0.811]	**0.764** [0.627, 0.799]	-1.120 [-1.393, -1.004]	**0.776** [0.648, 0.824]	0.884 [0.775, 0.946]	0.855 [0.748, 0.929]	–	**0.990** [0.983, 0.993]	0.449 [0.348, 0.580]	0.535 [0.412, 0.761]	1.000 [1.000, 1.000]	**0.160** [0.126, 0.214]
	256	**0.810** [0.721, 0.848]	0.702 [0.631, 0.728]	-1.113 [-1.280, -1.012]	**0.832** [0.745, 0.867]	0.178 [0.083, 0.446]	0.193 [0.093, 0.385]	–	**0.975** [0.964, 0.983]	1.221 [0.889, 1.499]	3.391 [2.121, 4.410]	1.000 [1.000, 1.000]	**0.232** [0.182, 0.342]
	1024	**0.858** [0.797, 0.882]	-86.894 [-310.480, -30.642]	-1.096 [-1.221, -1.016]	**0.865** [0.807, 0.893]	0.038 [0.009, 0.129]	0.006 [-0.018, 0.029]	–	**0.831** [0.779, 0.869]	1.828 [1.275, 2.228]	173.467 [82.409, 316.716]	1.000 [1.000, 1.000]	**0.564** [0.493, 0.630]
Allen–Cahn 1D	16	**0.788** [0.783, 0.811]	**0.783** [0.778, 0.808]	-1.014 [-1.037, -1.002]	**0.787** [0.783, 0.803]	**1.000** [0.999, 1.000]	0.989 [0.984, 0.995]	–	0.998 [0.993, 0.999]	**0.024** [0.017, 0.033]	0.142 [0.094, 0.161]	1.000 [1.000, 1.000]	0.075 [0.049, 0.115]
	64	0.825 [0.821, 0.827]	0.804 [0.800, 0.808]	-1.044 [-1.054, -1.024]	**0.838** [0.832, 0.839]	0.762 [0.688, 0.839]	0.672 [0.585, 0.719]	–	**0.984** [0.980, 0.991]	0.614 [0.506, 0.676]	0.981 [0.893, 1.076]	1.000 [1.000, 1.000]	**0.159** [0.119, 0.181]
	256	0.858 [0.853, 0.860]	0.687 [0.674, 0.705]	-1.032 [-1.041, -1.017]	**0.874** [0.874, 0.875]	0.122 [0.042, 0.201]	0.103 [0.051, 0.151]	–	**0.899** [0.872, 0.928]	1.528 [1.378, 1.683]	5.011 [4.729, 5.458]	1.000 [1.000, 1.000]	**0.399** [0.341, 0.442]
	1024	0.888 [0.886, 0.890]	-69.217 [-314.641, -25.083]	-1.027 [-1.031, -1.013]	**0.897** [0.897, 0.898]	0.018 [-0.004, 0.065]	0.002 [-0.017, 0.022]	–	**0.577** [0.541, 0.686]	2.058 [1.848, 2.269]	178.767 [100.427, 357.943]	1.000 [1.000, 1.000]	**0.776** [0.712, 0.799]
Diff.–Sorp. 1D	16	**0.800** [0.661, 0.845]	**0.799** [0.659, 0.843]	-0.982 [-1.000, -0.746]	**0.799** [0.655, 0.843]	**1.000** [1.000, 1.000]	0.998 [0.996, 1.000]	–	0.999 [0.996, 0.999]	**0.000** [0.000, 0.023]	0.060 [0.031, 0.091]	1.000 [1.000, 1.000]	0.051 [0.037, 0.086]
	64	**0.783** [0.717, 0.820]	**0.773** [0.710, 0.802]	-0.974 [-0.989, -0.913]	**0.804** [0.725, 0.833]	0.972 [0.940, 0.984]	0.931 [0.902, 0.955]	–	**0.993** [0.990, 0.996]	0.257 [0.194, 0.403]	0.380 [0.319, 0.457]	1.000 [1.000, 1.000]	**0.122** [0.100, 0.162]
	256	0.793 [0.713, 0.844]	0.628 [0.547, 0.665]	-0.983 [-0.995, -0.924]	**0.860** [0.802, 0.874]	0.582 [0.399, 0.707]	0.450 [0.229, 0.563]	–	**0.990** [0.981, 0.994]	0.816 [0.733, 0.987]	1.867 [1.374, 2.627]	1.000 [1.000, 1.000]	**0.156** [0.120, 0.268]
	1024	0.867 [0.824, 0.884]	-105.891 [-447.042, -27.001]	-0.984 [-0.995, -0.913]	**0.893** [0.860, 0.896]	0.209 [0.119, 0.348]	0.004 [-0.016, 0.030]	–	**0.960** [0.892, 0.978]	1.073 [0.975, 1.385]	78.799 [46.774, 155.232]	1.000 [1.000, 1.000]	**0.340** [0.234, 0.484]
Advection 1D	16	**0.897** [0.760, 0.956]	**0.896** [0.758, 0.955]	-0.991 [-1.010, -0.937]	**0.890** [0.723, 0.954]	**0.999** [0.998, 1.000]	**0.999** [0.996, 0.999]	–	0.995 [0.971, 0.998]	**0.040** [0.024, 0.057]	0.055 [0.033, 0.084]	1.000 [1.000, 1.000]	0.120 [0.069, 0.266]
	64	**0.800** [0.760, 0.815]	**0.791** [0.757, 0.806]	-0.973 [-1.038, -0.892]	**0.799** [0.735, 0.816]	**0.922** [0.882, 0.939]	0.882 [0.858, 0.907]	–	**0.936** [0.880, 0.957]	0.408 [0.360, 0.467]	0.499 [0.452, 0.579]	1.000 [1.000, 1.000]	**0.366** [0.308, 0.503]
	256	**0.843** [0.808, 0.853]	0.708 [0.662, 0.727]	-0.981 [-1.052, -0.893]	**0.835** [0.788, 0.860]	0.396 [0.220, 0.806]	0.287 [0.217, 0.392]	–	**0.672** [0.465, 0.859]	0.943 [0.607, 1.090]	2.382 [1.835, 2.913]	1.000 [1.000, 1.000]	**0.755** [0.524, 0.898]
	1024	**0.879** [0.859, 0.889]	-81.406 [-370.086, -20.223]	-0.988 [-1.046, -0.905]	0.861 [0.811, 0.886]	0.238 [0.052, 0.934]	0.011 [-0.013, 0.030]	–	**0.325** [0.227, 0.572]	1.044 [0.383, 1.485]	85.246 [35.886, 190.258]	1.000 [1.000, 1.000]	**0.948** [0.811, 0.978]
Heat 2D	16	**0.766** [0.685, 0.773]	**0.763** [0.684, 0.771]	-1.003 [-1.008, -1.000]	**0.767** [0.680, 0.774]	**0.993** [0.987, 0.996]	0.988 [0.974, 0.994]	–	**0.993** [0.989, 0.996]	**0.107** [0.083, 0.146]	0.143 [0.108, 0.199]	1.000 [1.000, 1.000]	0.118 [0.086, 0.147]
	64	0.812 [0.757, 0.822]	0.796 [0.767, 0.802]	-1.001 [-1.003, -0.999]	**0.835** [0.806, 0.837]	0.838 [0.703, 0.901]	0.815 [0.652, 0.912]	–	**0.990** [0.988, 0.994]	0.549 [0.442, 0.682]	0.671 [0.434, 0.995]	1.000 [1.000, 1.000]	**0.135** [0.119, 0.159]
	256	0.846 [0.817, 0.855]	0.661 [0.625, 0.683]	-1.001 [-1.002, -1.000]	**0.875** [0.871, 0.876]	0.198 [0.103, 0.334]	0.180 [0.105, 0.304]	–	**0.974** [0.966, 0.979]	1.315 [1.002, 1.642]	3.800 [2.172, 5.259]	1.000 [1.000, 1.000]	**0.241** [0.216, 0.269]
	1024	0.883 [0.878, 0.887]	-87.362 [-352.627, -32.800]	-1.001 [-1.002, -1.000]	**0.899** [0.897, 0.900]	0.030 [0.004, 0.079]	-0.004 [-0.017, 0.013]	–	**0.841** [0.748, 0.901]	2.170 [1.350, 2.468]	148.854 [87.984, 325.610]	1.000 [1.000, 1.000]	**0.649** [0.565, 0.713]
Darcy 2D	16	**0.666** [0.481, 0.770]	**0.664** [0.480, 0.769]	-1.011 [-1.115, -0.947]	**0.664** [0.469, 0.769]	**0.999** [0.998, 0.999]	0.997 [0.995, 0.998]	–	0.991 [0.984, 0.997]	**0.054** [0.043, 0.071]	0.081 [0.061, 0.099]	1.000 [1.000, 1.000]	0.134 [0.093, 0.189]
	64	**0.728** [0.543, 0.785]	**0.730** [0.546, 0.784]	-1.125 [-1.270, -0.987]	**0.745** [0.564, 0.798]	0.965 [0.945, 0.977]	0.955 [0.926, 0.967]	–	**0.981** [0.964, 0.990]	0.286 [0.235, 0.358]	0.315 [0.266, 0.402]	1.000 [1.000, 1.000]	**0.208** [0.153, 0.287]
	256	0.782 [0.683, 0.808]	0.682 [0.564, 0.707]	-1.110 [-1.311, -0.987]	**0.816** [0.755, 0.848]	0.619 [0.461, 0.776]	0.480 [0.395, 0.583]	–	**0.968** [0.955, 0.976]	0.776 [0.646, 0.900]	1.592 [1.367, 1.849]	1.000 [1.000, 1.000]	**0.268** [0.238, 0.393]
	1024	**0.831** [0.773, 0.861]	-99.623 [-400.029, -23.036]	-1.146 [-1.295, -0.993]	**0.846** [0.793, 0.880]	0.301 [0.098, 0.593]	0.018 [-0.005, 0.036]	–	**0.905** [0.865, 0.932]	1.024 [0.844, 1.183]	74.211 [41.893, 126.786]	1.000 [1.000, 1.000]	**0.507** [0.439, 0.659]
Reac.–Diff. 2D	16	**0.296** [0.236, 0.485]	**0.295** [0.232, 0.485]	-0.535 [-0.769, -0.416]	**0.298** [0.240, 0.488]	**0.997** [0.993, 0.998]	0.996 [0.991, 0.997]	–	**0.999** [0.998, 1.000]	0.075 [0.063, 0.108]	0.092 [0.072, 0.123]	1.000 [1.000, 1.000]	**0.037** [0.009, 0.066]
	64	**-0.092** [-0.268, 0.232]	**-0.122** [-0.309, 0.212]	-0.953 [-1.208, -0.311]	**-0.085** [-0.268, 0.250]	**0.922** [0.895, 0.942]	0.863 [0.820, 0.887]	–	**0.972** [0.926, 0.986]	0.384 [0.328, 0.443]	0.595 [0.523, 0.713]	1.000 [1.000, 1.000]	**0.255** [0.176, 0.407]
	256	**0.155** [-0.058, 0.375]	-0.504 [-1.008, -0.138]	-1.076 [-1.438, -0.652]	**0.166** [-0.028, 0.450]	0.639 [0.565, 0.726]	0.431 [0.379, 0.479]	–	**0.941** [0.913, 0.958]	0.796 [0.722, 0.848]	2.935 [2.379, 3.508]	1.000 [1.000, 1.000]	**0.366** [0.322, 0.452]
	1024	0.517 [0.336, 0.631]	-659.842 [-2504.930, -179.158]	-0.910 [-1.312, -0.587]	**0.560** [0.357, 0.672]	0.657 [0.513, 0.722]	0.022 [-0.003, 0.044]	–	**0.892** [0.817, 0.922]	0.772 [0.711, 0.938]	114.896 [63.903, 207.205]	1.000 [1.000, 1.000]	**0.548** [0.437, 0.692]
Shallow Water 2D	16	**0.264** [0.229, 0.318]	**0.264** [0.226, 0.317]	-1.590 [-1.944, -1.167]	**0.269** [0.230, 0.320]	0.993 [0.991, 0.995]	0.992 [0.990, 0.994]	–	**0.998** [0.997, 0.999]	0.118 [0.103, 0.132]	0.118 [0.109, 0.135]	1.000 [1.000, 1.000]	**0.056** [0.042, 0.078]
	64	0.518 [0.474, 0.668]	0.525 [0.479, 0.670]	-0.838 [-0.938, -0.764]	**0.551** [0.508, 0.694]	0.912 [0.892, 0.934]	0.917 [0.904, 0.932]	–	**0.992** [0.990, 0.994]	0.442 [0.383, 0.480]	0.429 [0.396, 0.463]	1.000 [1.000, 1.000]	**0.125** [0.107, 0.157]
	256	0.756 [0.717, 0.815]	0.645 [0.591, 0.699]	-0.934 [-1.012, -0.870]	**0.811** [0.785, 0.833]	0.728 [0.660, 0.869]	0.539 [0.493, 0.572]	–	**0.978** [0.971, 0.984]	0.694 [0.505, 0.763]	1.394 [1.301, 1.509]	1.000 [1.000, 1.000]	**0.219** [0.190, 0.266]
	1024	0.871 [0.855, 0.878]	-66.535 [-298.438, -26.911]	-0.949 [-0.998, -0.903]	**0.887** [0.878, 0.892]	0.909 [0.875, 0.935]	0.010 [-0.006, 0.039]	–	**0.964** [0.943, 0.974]	0.428 [0.367, 0.493]	34.301 [23.284, 82.147]	1.000 [1.000, 1.000]	**0.333** [0.267, 0.389]
Heat 3D	512	0.858 [0.833, 0.866]	-60.883 [-280.804, -13.804]	-1.000 [-1.002, -0.996]	**0.884** [0.872, 0.887]	0.107 [0.037, 0.309]	0.004 [-0.024, 0.030]	–	**0.884** [0.855, 0.905]	1.360 [0.996, 1.670]	55.623 [35.638, 111.006]	1.000 [1.000, 1.000]	**0.516** [0.465, 0.604]
MGN Cylinder	64	**0.799** [0.735, 0.830]	**0.789** [0.732, 0.812]	-1.066 [-1.196, -0.948]	**0.794** [0.705, 0.824]	**0.957** [0.909, 0.976]	0.900 [0.818, 0.943]	–	0.928 [0.774, 0.977]	**0.289** [0.218, 0.406]	0.465 [0.366, 0.619]	1.000 [1.000, 1.000]	0.402 [0.213, 0.645]
MGN Airfoil	64	**0.757** [0.564, 0.792]	**0.749** [0.550, 0.785]	-1.210 [-1.624, -0.815]	0.694 [0.517, 0.775]	**0.911** [0.848, 0.945]	0.827 [0.751, 0.894]	–	0.630 [0.269, 0.794]	**0.392** [0.314, 0.506]	0.634 [0.516, 0.762]	1.000 [1.000, 1.000]	0.805 [0.639, 1.008]

Table 9: SSIM between attribution maps of the same sample at two resolutions at the domain center. Median over 30 test inputs, inter-quartile range below, and 1 is perfect structural agreement.

PDE	players	OperatorSHAP	Reference ($3{\cdot}10^5$)	RegressionMSR ($\max(2n, 1024)$)
Heat 2D	64–256	0.360 [0.318, 0.401]	0.352 [0.334, 0.413]	0.255 [0.192, 0.317]
	256–1024	0.678 [0.656, 0.701]	0.606 [0.575, 0.624]	0.138 [0.104, 0.301]
	64–1024	0.140 [0.115, 0.168]	0.148 [0.123, 0.183]	0.077 [0.043, 0.117]
Shallow Water 2D	64–256	0.271 [0.228, 0.357]	0.292 [0.245, 0.346]	0.364 [0.301, 0.402]
	256–1024	0.834 [0.824, 0.854]	0.783 [0.761, 0.798]	0.663 [0.632, 0.706]
	64–1024	0.192 [0.147, 0.274]	0.174 [0.132, 0.245]	0.271 [0.230, 0.340]

Table 10: Relative L^2 test error of the backbones per grid points per axis $n_{1\mathrm{D}}$, median over 30 test inputs.

PDE $\setminus$ $n_{1\mathrm{D}}$	4	8	16	32	64	128	256	512	1024
Heat 1D	–	0.049	0.017	0.005	0.005	0.005	0.005	0.005	0.005
Burgers 1D	–	0.115	0.053	0.047	0.046	0.051	0.051	0.051	0.050
Burgers 1D (ReLU)	–	0.088	0.034	0.026	0.022	0.022	0.021	0.021	0.021
Allen–Cahn 1D	–	0.245	0.008	0.004	0.003	0.003	0.007	0.009	0.010
Diff.–Sorp. 1D	–	0.056	0.001	0.002	0.002	0.003	0.004	0.004	0.005
Advection 1D	–	0.129	0.025	0.029	0.035	0.037	0.037	0.037	0.037
Heat 2D	0.364	0.049	0.002	0.001	–	–	–	–	–
Darcy 2D	0.373	0.101	0.042	0.037	–	–	–	–	–
Reac.–Diff. 2D	0.364	0.305	0.220	0.212	–	–	–	–	–
Shallow Water 2D	0.085	0.035	0.006	0.005	–	–	–	–	–
Heat 3D	–	0.032	–	–	–	–	–	–	–

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: Our explainer allows training and predicting on irregular meshes, by predicting Aumann–Shapley densities instead of Shapley values. We backed this with rigorous theory. Please see Secs. 4 to 6 for further details.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We have a whole section (Sec. 7) dedicated to limitations.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [Yes]

Justification: All of our proofs are in Appendix B. The theorems come with a complete set of assumptions.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We provide details about our experiments in Sec. 6 and the Appendix C. We additionally submit code with clear instructions on reproduction.

   Guidelines:

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

5. **Open access to data and code**

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

Answer: [Yes]

Justification: The code at `https://github.com/StillJosh/OperatorSHAP-NeurIPS2026` regenerates the PDE data, downloads the MeshGraphNets data from its public source, and reproduces every table and figure with one script.

Guidelines:

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

6. **Experimental setting/details**

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

Answer: [Yes]

Justification: We specify the most important parameters in Sec. 6 and everything else in the Appendix C.

Guidelines:

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

7. **Experiment statistical significance**

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

Answer: [Yes]

Justification: Our plots and tables show median and IQR (see Appendix C.6 for details).

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Yes, we provide information about the computing resources necessary in Appendix E.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: There is no direct societal harm expected from our research. There could be harm coming from false explanations. While this is negligible, we reflect that in the limitations section of our paper (see Sec. 7).

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: Yes, we discuss this in our introduction (Sec. 1) and limitations (Sec. 7).

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: Most of the code we build on is standard scientific Python tooling. For shapiq and the MGN datasets, we give explicit credit and licenses in Appendix F.

    Guidelines:

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

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

13. **New assets**

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

    Answer: [Yes]

    Justification: The code is released under the MIT license, with a README that documents setup, commands, and hardware requirements.

    Guidelines:

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

14. **Crowdsourcing and research with human subjects**

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

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

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: We describe our LLM policy in Appendix G.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.