Neural Harmonic Measure Operator

Jinjin He **Sinan Wang** **Yuchen Sun** **Bo Zhu**
Georgia Institute of Technology
{jhe433,swang3081,ysun748,bo.zhu}@gatech.edu

Abstract

We introduce **Neural Harmonic Measure Operator (NHMO)**, a neural solver for elliptic PDE problems on variable-shape domains. The harmonic measure of a domain is the boundary probability distribution that, integrated against any boundary data, returns the Dirichlet Laplace solution. It depends only on the geometry, not on the boundary data. NHMO parameterizes the density of this measure as a transformer-based boundary kernel supervised by Walk-on-Spheres exit samples, so one trained kernel handles different boundary values on a shape with no retraining. We extend it to Poisson via a classical decomposition, with an auxiliary network amortizing the source-induced correction and avoiding the singular volume quadrature that breaks direct evaluation. At inference, new boundary values and new sources both yield PDE solutions by re-integration against the fitted kernel and lift, with no retraining. NHMO improves over four prior baselines on the MCB-B 3D variable-shape Poisson benchmark across all five categories, and is competitive with major neural-operator baselines on a controlled 2D testbed.

1 Introduction

Classical solvers like the finite element method and finite differences [Hughes, 2003, LeVeque, 2007] discretize the volumetric interior of Ω into a mesh and re-run the discretize-and-solve pipeline whenever the geometry, source, or boundary data changes, a bottleneck in design optimization [Bendsoe and Sigmund, 2013], uncertainty quantification [Smith, 2024], and inverse problems [Engl et al., 1996]. Neural operators amortize this cost by learning a function-to-function map $(\Omega, f, h) \mapsto u$ from domain, source, and boundary data to the solution, which evaluates in a single forward pass once trained. Foundational architectures parameterize integral kernels in the Fourier domain on regular grids [Li et al., 2020a] or use a branch–trunk decomposition at fixed sample points [Lu et al., 2021]; recent transformer-based variants handle irregular meshes via attention over mesh points or learned slice tokens [Wu et al., 2024, Wang and Wang, 2024, Alkin et al., 2024, Zhou et al., 2026]. Yet these methods inherit the volumetric framing of FEM/FDM, still operating on the bulk interior of Ω with compute scaling with volumetric discretization rather than the codimension-one boundary.

Boundary integral methods take a different angle on the same problem. The Boundary Element Method (BEM) reformulates a volumetric Dirichlet problem as an integral equation against a Green's-function kernel on the boundary $\partial\Omega$ alone [Sauter and Schwab, 2010], but it still requires a discretization of $\partial\Omega$ and dense linear-system solves. Stochastic methods such as Walk-on-Spheres (WoS) [Muller, 1956, Sawhney and Crane, 2020, Sawhney et al., 2022, 2023] sidestep boundary discretization by estimating $u(p) = \mathbb{E}_p[h(B_\tau)]$ via Brownian-exit Monte Carlo, where $\mathbb{E}_p$ is the expectation over Brownian motions B_t started at p and B_τ is the first-exit point on $\partial\Omega$, but remains a per-query estimator instead of an amortized operator, paying $O(N_{\text{walks}})$ each time the boundary datum changes. Recent learned Green's-function-style operators, including NGF [Yoo et al., 2025] and others [Gin et al., 2021, Li et al., 2020c, Teixeira et al., 2026], all parameterize a volumetric kernel on the full pair space $\Omega \times \Omega$ and inherit the singular Green's function (see §2).

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

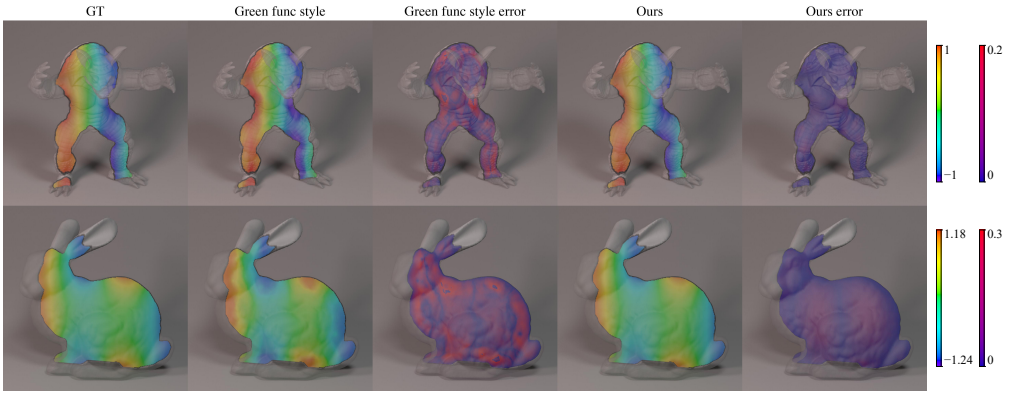


Figure 1: Intuitive demonstration on complex 3D shapes (top: armadillo; bottom: bunny), with a single shape-conditioned K_θ encoding both. Mean rel-L_2 is 0.012 vs 0.142 (Ours vs GF style; §5.1). Columns: GT (reference solution), the Green's-function-style (GF-style) baseline, its absolute error, our prediction, and our absolute error. Per row, fields share the left color bar and errors the right.

We propose **Neural Harmonic Measure Operator (NHMO)**, a boundary-only neural operator for elliptic PDEs on variable-shape domains that, in contrast to the volumetric Green's-function operators above, learns the density of a codimension-one boundary measure on $\Omega \times \partial\Omega$ rather than a kernel on $\Omega \times \Omega$. This drops the kernel domain by one dimension and replaces a singular volumetric kernel with a probability density on the boundary. NHMO contains two learned components. First, a transformer-based boundary kernel $K_\theta(p, \zeta; \Omega)$ approximates the density $d\omega_p/d\sigma$ of the harmonic measure ω_p, a geometry-only distribution over the boundary; we call K_θ the harmonic-measure density. Here, geometry-only means that ω_p depends on the domain Ω and query point p, but not on the prescribed boundary values. Integrating this distribution against any boundary data then recovers the Dirichlet Laplace solution via Kakutani's representation [Kakutani, 1944]. Second, a residual lift v_φ carries the source-induced contribution for Poisson problems via the classical balayage decomposition. The two components compose additively. These give NHMO structural advantages over volumetric operators. As a geometry-only probability kernel that depends on Ω rather than h or f, a single fitted K_θ can be reused for arbitrary boundary data on the same shape without retraining, and, once the kernel is normalized over the boundary quadrature, its boundary term satisfies the maximum principle by construction. It is trained mesh-free from Walk-on-Spheres exit samples, requiring no FEM solutions or tetrahedral meshes for kernel supervision. As a proof of concept, Figure 1 shows NHMO and a Green's-function-style baseline on two complex 3D shapes (detailed in §5.1).

Contributions. **(1)** We propose to encode the density of the harmonic measure ω_p as a learnable boundary kernel K_θ that depends on the geometry Ω alone (independent of boundary data h and source f), supervised by Walk-on-Spheres (§4.3, §5.3). **(2)** We extend NHMO to Poisson problems via the classical balayage decomposition, with a zero-boundary-gauge lift v_φ that reuses K_θ to amortize the source correction without singular volume quadrature (§4.4). **(3)** We validate NHMO on the MCB-B 3D Poisson benchmark, where it outperforms four neural-operator baselines, and on a controlled 2D MNIST testbed, and show the framework's generality through intuitive 3D harmonic and drift-adaptation experiments (§5.3, §5.2, §5.1).

2 Related Work

Neural operators for PDEs. Function-to-function neural operators [Kovachki et al., 2023] learn end-to-end maps from problem data to solutions. Foundational architectures include Graph Neural Operators [Li et al., 2020b], Fourier Neural Operators [Li et al., 2020a] that parameterize integral kernels in the spectral domain, and DeepONet [Lu et al., 2021] with a branch-trunk architecture on functions sampled at fixed points. Geometry-aware extensions handle irregular meshes via attention or graph modules [Li et al., 2023, Wu et al., 2024, Wang and Wang, 2024, Alkin et al., 2024], and several lines target varying domain geometries [Wang et al., 2024, Yin et al., 2024, Wu et al., 2026]. Transformer-based operators [Hao et al., 2023, Xiao et al., 2023, Luo et al., 2025, Zhou et al., 2026]

use attention over mesh points or learned slice tokens. These methods regress the solution or solution operator directly; we instead model the boundary measure that mediates all solutions.

Learning Green's functions and integral operators. A separate line learns the volumetric Green's function $G_\Omega(p, q)$ for linear PDEs via rational neural networks [Boullé et al., 2022], Dirac-delta approximations [Teng et al., 2022], radial-basis approximations [Negi et al., 2024], and variational principles [Teixeira et al., 2026]; deep nonlinear-BVP extensions appear in DeepGreen [Gin et al., 2021], and Green's-function-style multipole structure underlies the multipole graph neural operator [Li et al., 2020c]. Neural Green's Functions (NGF) [Yoo et al., 2025] is the closest prior work and our principal baseline. NGF learns the domain Green's function $G_\Omega(p, q)$ as $\Phi_\theta(p)^\top D\Phi_\theta(q)$ for learned per-point features, trained on precomputed FEM solution fields, and recovers solutions by integrating f against G_Ω in the volume and h against the outward-normal derivative on the boundary. Earlier 2D boundary-integral neural methods [Lin et al., 2021, Sun et al., 2023] and neural integral operators [Zappala et al., 2024] target classical BEM-style discretizations rather than amortizing across boundary measures of varying shapes.

Walk on Spheres and grid-free Monte Carlo solvers. Walk on Spheres (WoS) [Muller, 1956] is a Monte Carlo estimator for elliptic PDEs based on Brownian-exit simulation. The grid-free perspective was revived for graphics and learning by Sawhney and Crane [2020], and Walk on Stars [Sawhney et al., 2023] extends it to mixed boundary conditions and source terms, with follow-ups for spatially varying coefficients, surface PDEs, gradient computation, and variance reduction [Sawhney et al., 2022, Sugimoto et al., 2024, Miller et al., 2024, Huang et al., 2025, Sawhney and Miller, 2023]. The ideal WoS estimator is unbiased; practical walks stop in an ε-shell around $\partial\Omega$ after $O(\log(1/\varepsilon))$ expected steps [Binder and Braverman, 2012], introducing an $O(\varepsilon)$ bias [Mascagni and Hwang, 2003]. WoS pays $O(N_{\text{walks}})$ per query at inference; neural surrogates trained against WoS targets [Nam et al., 2024, Zhang et al., 2025, Miller et al., 2023] amortize this cost. We use WoS as ground-truth supervision for K_θ rather than as a runtime estimator.

Harmonic measure in analysis. The harmonic measure is classical in potential theory and geometric function theory [Garnett and Marshall, 2005]. In 2D it is conformally invariant, and its dimensional properties characterize boundary regularity [Makarov, 1985, Armitage and Gardiner, 2012]. Kakutani's theorem [Kakutani, 1944] identifies it with the Brownian-exit law. To our knowledge, this is the first work to parameterize the density of the harmonic measure with a neural network and to realize the balayage decomposition with a learned source amortizer.

3 Background

3.1 Harmonic measure

Let $\Omega \subset \mathbb{R}^d$ ($d \in \{2, 3\}$) be a bounded Lipschitz domain. The *harmonic measure* ω_p at $p \in \Omega$ is the probability distribution on $\partial\Omega$ describing where a Brownian motion started at p first exits Ω [Kakutani, 1944]: with B_t a Brownian motion in $\mathbb{R}^d$ with $B_0 = p$ and $\tau = \inf\{t > 0 : B_t \notin \Omega\}$ its first-exit time,

$$\omega_p(E) \;=\; \mathbb{P}_p[B_\tau \in E]\,, \qquad E \subset \partial\Omega \text{ Borel.} \tag{1}$$

The family $\{\omega_p\}_{p\in\Omega}$ depends only on the geometry Ω, not on any boundary data.

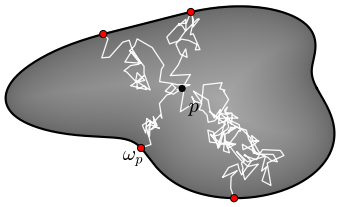


Figure 2: Harmonic measure (ω_p) from Brownian exit locations. Red dots denote small boundary patches (E).

Constructing the Dirichlet solution. For continuous boundary data $h \in C(\partial\Omega)$, the Dirichlet problem $\Delta u = 0$ in Ω with $u = h$ on $\partial\Omega$ has the closed-form solution [Garnett and Marshall, 2005]

$$u(p) \;=\; \int_{\partial\Omega} h(\zeta)\, d\omega_p(\zeta) \;=\; \mathbb{E}_p[h(B_\tau)]\,. \tag{2}$$

The probabilistic form on the right is the basis of Walk-on-Spheres Monte Carlo solvers [Muller, 1956, Sawhney and Crane, 2020, Sawhney et al., 2023]. Once ω_p is known for a geometry Ω, equation (2) resolves the Dirichlet problem for any h via a single boundary integral. On a Lipschitz domain, ω_p

is absolutely continuous with respect to the surface measure σ, and its Radon–Nikodym density $d\omega_p/d\sigma(\zeta) = -\partial_{\nu_\zeta} G_\Omega(p,\zeta)$ is the Poisson kernel, with G_Ω the Dirichlet Green's function. NHMO learns this *harmonic-measure density*; we reserve "harmonic measure" for ω_p itself and name the method after it, since ω_p exists on any bounded domain and WoS exit points are drawn from it. We present a derivation of (2) and further properties of ω_p in Appendix B.

3.2 Newtonian potential and balayage

The Poisson problem extends (2) to nonzero source $f \in L^\infty(\Omega)$:

$$\Delta u \;=\; f \text{ in } \Omega, \qquad u \;=\; h \text{ on } \partial\Omega. \tag{3}$$

Let Φ be the fundamental solution of $-\Delta$ on $\mathbb{R}^d$, the radial solution of $-\Delta\Phi = \delta_0$, which is positive near the origin (the usual sign convention in potential theory):

$$\Phi(x) \;=\; \begin{cases} -\frac{1}{2\pi}\log|x| & d=2, \\ \frac{1}{4\pi|x|} & d=3, \end{cases} \tag{4}$$

and define the *Newtonian potential* of f by $N_f(p) = -\int_\Omega \Phi(p-q)\, f(q)\, dq$, so that $\Delta N_f = f$ on $\mathbb{R}^d$.

Balayage decomposition. Setting $w = u - N_f$ in (3) yields $\Delta w = 0$ in Ω with $w|_{\partial\Omega} = h - N_f|_{\partial\Omega}$. Applying (2) to w and grouping the terms that do not involve h gives

$$u(p) \;=\; \int_{\partial\Omega} h(\zeta)\, d\omega_p(\zeta) \;+\; u_f(p), \qquad u_f(p) \;=\; N_f(p) \;-\; \int_{\partial\Omega} N_f|_{\partial\Omega}(\zeta)\, d\omega_p(\zeta), \tag{5}$$

where the source-only piece u_f is independent of h and satisfies $\Delta u_f = f$ in Ω with $u_f|_{\partial\Omega} = 0$. The same harmonic measure that handles the boundary data also handles the source-induced boundary correction, applied to $N_f|_{\partial\Omega}$ instead of h. With $f \equiv 0$ the decomposition recovers the pure Laplace identity (2). NHMO's two-component factorization in §4 is the neural counterpart of this split: the boundary integral becomes a learned kernel, the source-only piece becomes a learned residual field.

4 Neural Harmonic Measures

Throughout this section, ω_p denotes the harmonic measure and K_θ the learned harmonic-measure density that approximates $d\omega_p/d\sigma$; v_φ denotes the residual lift. The full symbol list is in Appendix A. Three equations play distinct roles: the continuous identity (5), the learned model (6), and its quadrature implementation (7).

4.1 Problem setup

Each shape category is a distribution over bounded Lipschitz domains $\Omega \subset \mathbb{R}^d$ with $d \in \{2,3\}$. A training set provides shapes drawn from this distribution; per shape, a reference solution u_{true} for a parametric family of Poisson problems $\Delta u = f$, $u|_{\partial\Omega} = h$ is given at a discretization of Ω. The reference solver and discretization are experimental choices (§5). At test time we evaluate on held-out shapes and on held-out (h, f) pairs, including problems with coefficients drawn from outside the training support to test generalization across the parametric BC distribution.

4.2 Decomposition

By the balayage identity (5), the solution splits into a clean h-only boundary integral plus an h-independent source-only piece that vanishes on $\partial\Omega$. We factorize NHMO along this split:

$$u(p) \;=\; \underbrace{\langle h,\, K_\theta(p,\cdot;\Omega)\rangle_{\partial\Omega}}_{u_h(p)} \;+\; \underbrace{v_\varphi(p;\Omega,h,f)}_{\approx\, u_f(p)}, \tag{6}$$

where $u_h(p)$ denotes the boundary-integral prediction and $u_f(p)$ the zero-boundary Poisson particular solution; $K_\theta(p,\zeta;\Omega)$ is a learned harmonic-measure density approximating $d\omega_p/d\sigma$ and v_φ is a learned residual field. Setting $K_\theta = d\omega_p/d\sigma$ and $v_\varphi = u_f$ recovers (5) identically. In practice we

let v_φ depend on h as well as f, so the residual lift can absorb approximation error from imperfect kernel fits on top of carrying the source contribution; §6 examines this dependence and separates the two roles. Two properties of this decomposition are critical to our results. First, K_θ does not depend on h or f, so the boundary kernel is geometry-only and a single fitted K_θ handles every (h, f) pair on a shape without retraining. Second, u_h is a linear functional of the boundary data: a coefficient shift in h, including coefficients drawn from outside the training support, changes u_h proportionally without altering the kernel itself. End-to-end operators that fit u as a nonlinear map of (h, f) do not enjoy this property; their solution can drift arbitrarily under a coefficient shift unseen at training time. Out-of-distribution generalization across the parametric BC family is therefore a structural property of NHMO's kernel channel, not an emergent effect from fitting; the lift carries no such guarantee. The residual lift v_φ catches source-induced contributions and any remaining approximation error.

4.3 Boundary kernel K_θ

$K_\theta(p, \zeta; \Omega)$ is realized as the composition of a geometry encoder E and a kernel head g. The encoder maps a discretization of Ω to a fixed-size shape latent $\psi_\Omega = E(\Omega)$. The kernel head outputs a scalar log-density $\log \widetilde{K}_\theta(p, \zeta; \Omega) = g(p, \zeta, \psi_\Omega)$, with inputs augmented by Fourier features of (p, ζ) and the inter-point distance $\|p - \zeta\|$. Given a boundary discretization $\{\zeta_i\}_{i=1}^{N_s}$ of N_s surface points with quadrature weights w_i, we normalize the kernel over the quadrature at inference, $K_\theta(p, \zeta_i; \Omega) = \widetilde{K}_\theta(p, \zeta_i; \Omega) / \sum_j w_j \widetilde{K}_\theta(p, \zeta_j; \Omega)$, so that $\sum_i w_i K_\theta(p, \zeta_i; \Omega) = 1$ holds exactly and the boundary integral

$$u_h(p) \;=\; \sum_{i=1}^{N_s} w_i \, K_\theta(p, \zeta_i; \Omega)\, h(\zeta_i) \qquad (7)$$

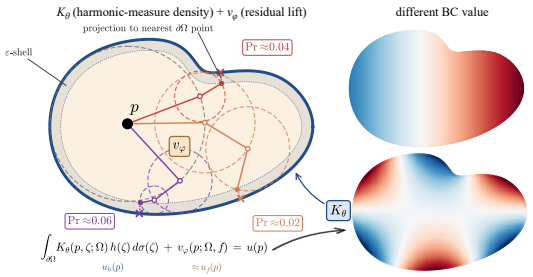


Figure 3: **NHMO overview.** Harmonic-measure density $K_\theta \approx d\omega_p/d\sigma$ with WoS paths and residual lift v_φ, composed additively as in (6) (bottom). Each WoS step lands on the largest circle inside Ω; a walk stops in the ε-shell (drawn wider than in practice) and is projected to $\partial\Omega$. Right: K_θ once fit for Ω solves any new boundary datum without retraining.

is a convex combination of boundary values, so $\min h \le u_h \le \max h$. In 3D, a training-time penalty (§4.5) keeps $\sum_i w_i \widetilde{K}_\theta(p, \zeta_i; \Omega) \approx 1$. Encoder, kernel head, and feature parameterizations for the 2D and 3D realizations are in Appendix C.

By (2), when $K_\theta = d\omega_p/d\sigma$ and h is analytically harmonic, the boundary integral (7) returns $h(p)$ up to quadrature error. We use this to check the trained kernel on held-out geometries by evaluating u_h against analytic harmonic functions (e.g., $h \in \{x,\ xy,\ x^2 - y^2,\ e^x \cos y\}$ in 2D, with low-degree solid spherical harmonics in 3D). The check is unavailable to end-to-end neural operators that do not expose a kernel; numbers are reported in §5.3.

A single fitted K_θ amortizes solutions across (h, f) pairs on the same shape: for Laplace ($f \equiv 0$) the boundary integral (7) alone suffices; for Poisson the kernel composes additively with v_φ for the source contribution. At inference, the discrete effective-kernel matrix $K_{\text{eff}} = [w_j\, K_\theta(p_i, \zeta_j; \Omega)]_{ij}$ is materialized once per geometry and reused for every (h, f), so per-problem inference reduces to a boundary matvec plus a lift forward; wall-clock measurements are in §5.4.

4.4 Field lift v_φ

In 2D, v_φ is parameterized as a U-Net on the ambient discretization grid of Ω. Inputs are the interior mask $\mathbf{1}_\Omega$ (the indicator function of Ω, equal to 1 inside and 0 outside), the boundary data h extended onto the grid, the source field f, and the kernel's own boundary-integral prediction u_h evaluated at every grid pixel. The output is a single residual channel; the final prediction (6) is masked to the interior via multiplication by $\mathbf{1}_\Omega$. The lift sees the kernel's prediction as a guide and learns the residual. For pure Laplace problems ($f \equiv 0$), the kernel alone supplies u_h via (7) and v_φ has only the residual approximation error in K_θ to correct; for Poisson problems, the lift carries the source-induced contribution that the boundary integral cannot represent. Because the lift conditions

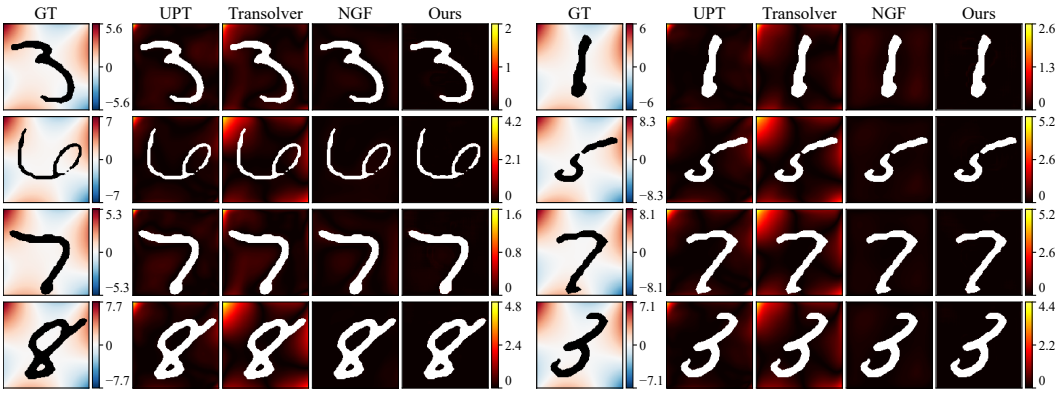


Figure 4: **2D MNIST out-of-distribution (OOD) qualitative.** Per row, a Laplace example (left) paired with a Poisson example (right); columns are GT (finite-difference reference) and the absolute error $|\text{pred} - \text{GT}|$ of UPT, Transolver, NGF, and Ours. Within each example (half-row), the four error panels share one color scale and GT has its own. Additional shapes in Appendix G.3.

on u_h, the kernel and the lift compose into a single forward pass per query field with no iterative coupling. In 3D, v_φ is a cross-attention head whose query is a Fourier embedding of p and whose context is the shape latent together with tokens that summarize source samples $(q_j, f(q_j))$; it does not see h, and its output is multiplied by $\max(0, -\mathrm{SDF}(p))$. The same kernel-plus-lift template adapts to nearby elliptic operators, e.g. constant-drift Laplace, by replacing the lift with a small drift-conditioned adapter while reusing the geometric kernel without retraining; we demonstrate this on a bunny domain in §5.1. Depths, widths, and parameter counts are in Appendix C.

4.5 Training

Training is two-stage. The kernel K_θ is trained per geometry distribution from Walk-on-Spheres exit samples [Muller, 1956, Sawhney and Crane, 2020]. From each interior probe p, we simulate M Brownian-motion exit points $\{\zeta_k\}_k \subset \partial\Omega$: each WoS step jumps to a uniform point on the largest sphere around the current point inside Ω, a walk stops in the ε-shell of $\partial\Omega$ ($\varepsilon = 10^{-3}$ of the normalized domain) and is projected to the nearest boundary point, and walks that do not stop within 128 steps are masked out. In 2D, we precompute 10^4 walks for each of 32 probes per shape, and in 3D we draw 4 fresh exits for each of 8 probes per gradient step. In 2D, we fit $K_\theta(p, \cdot)$ against a Gaussian kernel-density estimate (KDE) of these samples at bandwidth σ (a small fraction of the domain width, 0.2%, so ε is half of σ), minimizing the KL divergence to the KDE plus 0.5 times the L_1 distance; in 3D, we maximize the likelihood of the exit points (Appendix D.1). Probes are drawn from near-boundary, mid-interior, and deep-interior bands. In 3D, three regularizers add structure: $\mathcal{L}_Z$ pins $\log \sum_i w_i \widetilde{K}_\theta(p, \zeta_i; \Omega)$ to zero via a Huber penalty, $\mathcal{L}_{\text{MV}}$ enforces the mean-value property of harmonic functions on spheres $B(p, r) \subset \Omega$ that lie strictly inside Ω, and $\mathcal{L}_{\text{BL}}$ makes the kernel concentrate at a boundary point as the probe approaches it (Appendix D.1). Generating this supervision is a negligible share of training: our GPU sampler completes about 5×10^8 walks per second on an A100 even at a stricter $\varepsilon = 10^{-4}$ (Appendix D.5 gives budgets, masked fractions, and variance).

With K_θ frozen, the lift v_φ is trained by masked MSE between the composed prediction (6) and the numerical reference u_{true}, computed in y-normalized space. One shape $\times$ one (h, f) instance per gradient step; u_h is recomputed on-the-fly through the frozen kernel. No PDE-residual loss is used at any stage. Loss weights, optimizer, and learning-rate schedule for both stages are in Appendix D.

5 Experiments

We evaluate NHMO on two complementary benchmarks, a controlled 2D MNIST testbed in which all neural-operator baselines run under a single code path (§5.2) and the published 3D MCB-B Poisson benchmark of [Yoo et al., 2025] where NHMO is compared against four prior methods on five

mechanical-part categories (§5.3), preceded by a short pair of intuitive demonstrations on complex 3D shapes (§5.1). Runtime (§5.4) and ablation (§5.5) analyses follow.

5.1 Intuitive demonstrations on complex shapes

As proof of concept, two demos share a single harmonic-measure density K_θ fitted once across four graphics meshes; under matched optimization budgets, NHMO reaches substantially lower error than a Green's-function-style baseline (GF style) that learns a volumetric Green's function as in prior work [Yoo et al., 2025, Boullé et al., 2022, Teng et al., 2022, Negi et al., 2024, Gin et al., 2021, Li et al., 2020c, Teixeira et al., 2026]. *(i)* On four graphics meshes (armadillo, bunny, fandisk, lucy) with $h \in \{\sin x, \sin z\}$, NHMO reaches mean rel-L_2 of 0.012 against an FEM reference versus 0.142 for GF style. *(ii)* For constant-drift Laplace $\Delta u + \beta \cdot \nabla u = 0$ on a 2D bunny slice, the same K_θ plus a small drift-conditioned adapter reaches 0.062 versus 0.323 for GF style. Details and figures are in Appendix F.

5.2 2D MNIST: controlled cross-baseline benchmark

We construct a controlled 2D testbed using MNIST digit silhouettes as planar domains, with parametric Laplace and Poisson problems posed on each, and run all neural-operator baselines under one training and evaluation pipeline. It probes out-of-distribution (OOD) extrapolation across BC coefficients and multiply-connected boundaries (digits 0, 6, 8, 9); details are in Appendix G.1. The baselines include BENO [Wang et al., 2024], which is designed for elliptic problems with complex boundaries.

Table 1 reports results on the in-distribution and OOD splits; the OOD split draws BC coefficients strictly outside the training range. NHMO has the lowest mean in-distribution error and the lightest error tails on both splits (OOD examples in Figure 4). Under the OOD shift it degrades by $1.25\times$ (median), while the nonlinear end-to-end baselines (Transolver, LNO, UPT, BENO) degrade by $4\times$ to $8\times$; even the kernel-only variant beats all of them on OOD. Our 2D port of NGF also extrapolates well and has a quite low OOD mean and median: like NHMO, it pairs geometry-only features with a read-out that is linear in the data, the class of operator this paper argues for. Its in-distribution errors, however, are heavy-tailed (p95 16.9% and max 40.2%, against our 3.3% and 7.0%), consistent with its rank-limited bilinear source coupling. Across five training seeds of the lift, the test mean is $2.09 \pm 0.03\%$ and the OOD mean $2.60 \pm 0.05\%$ (Appendix K.1).

Table 1: 2D MNIST paramBC: relative L_2 error (%) over the full interior, in-distribution and OOD ($U[+1, +2]$). p95: per-pair 95th percentile. The OOD/test ratio (of medians) measures BC-coefficient extrapolation; lower is better. Per-problem inference is on a single A100 with the per-shape kernel matrix cached. NGF is released only in 3D; its row is our 2D port of the official code and training setup (Appendix G.2).

	test (in-dist)		test_ood			
Method	median / mean	p95	median / mean	p95	OOD/test	per-problem
Transolver [Wu et al., 2024]	4.4 / 5.3	10.7	36.4 / 36.2	55.2	8.3×	12.5 ms
LNO [Wang and Wang, 2024]	5.2 / 6.4	—	23.3 / 33.2	81.6	4.5×	13.2 ms
UPT [Alkin et al., 2024]	5.7 / 6.3	—	26.5 / 26.6	—	4.6×	7.6 ms
BENO [Wang et al., 2024]	5.9 / 7.0	16.1	44.1 / 40.5	65.8	7.5×	—
NGF [Yoo et al., 2025] (2D port)	2.0 / 3.9	16.9	3.9 / 4.2	6.0	1.93×	11.6 ms
NHMO K-only (kernel only)	5.4 / 7.7	18.2	5.6 / 6.8	14.2	1.0×	—
NHMO (kernel + lift)	2.0 / 2.1	3.3	2.5 / 2.6	4.2	1.25×	**4.0 ms**
+ residual head (§6)	**1.7 / 1.8**	**3.0**	**2.4 / 2.6**	**4.1**	1.37×	—

5.3 3D MCB-B Poisson

We follow the protocol of [Yoo et al., 2025] exactly. MCB-B comprises five categories of mechanical-part shapes (Nut, Gear, Motor, Fitting, Screws & Bolts) from MCB [Kim et al., 2020], each with 200 training and 20 unseen test shapes; per shape, the benchmark provides 16 unseen (h, f) problems (8 sources $\times$ 2 BCs, held out from training) with FEM reference solutions to $\Delta u = f$ on tetrahedral

Table 2: MCB-B Poisson benchmark: relative L_2 error against the FEM reference, mean over 20 unseen test shapes $\times$ 16 unseen (h, f) problems per category. Lower is better. NGF, Transolver, LNO, UPT numbers are reported in NGF Table 2 under identical evaluation.

Method	Nut	Gear	Motor	Fitting	Screws & Bolts
Transolver [Wu et al., 2024]	0.320	0.281	0.407	0.180	0.221
LNO [Wang and Wang, 2024]	0.372	0.466	0.528	0.259	0.239
UPT [Alkin et al., 2024]	0.516	0.507	0.765	0.392	0.358
NGF [Yoo et al., 2025]	0.275	0.243	0.338	0.160	0.189
Ours (NHMO)	**0.216**	**0.188**	**0.284**	**0.147**	**0.131**

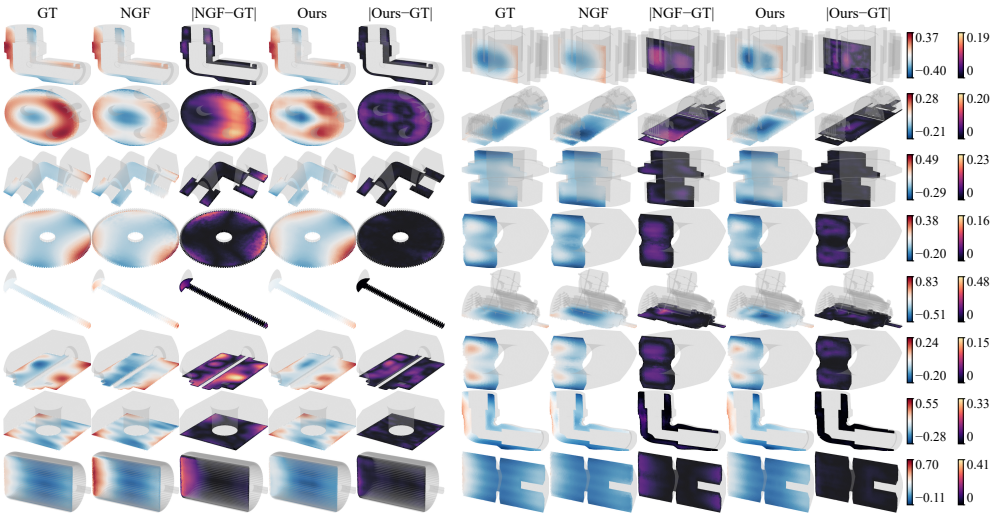


Figure 5: **Qualitative comparison on MCB-B Poisson.** Per shape, five panels show GT (the FEM reference), NGF prediction, NGF error, our prediction, and our error; the cut face is colored by the field, the back half by a gray ghost surface. Within each row, GT and the predictions share one color scale and the four error panels share a second one; panels are stretched to a common aspect ratio. Two shapes per row across all five MCB-B categories. Additional shapes in Appendix H.

meshes, yielding 320 test pairs per category. Our networks (K_θ at 2.77M params trained with WoS distillation, v_φ at $\sim$ 5M params trained on FEM-supervised MSE via warm-start, §4.5) and baseline implementations are detailed in Appendices C and E. Reported metric is relative L_2 error against the FEM reference, evaluated at every interior tetrahedral-mesh (tet) vertex and averaged over all 320 test pairs.

We outperform NGF on all five categories and outperform Transolver, LNO, and UPT by wider margins (Table 2). Per-shape distributions (Table 3) show median error below NGF's reported mean for all five categories, with 95th-percentile error below 0.50 on every category, indicating that large errors are not widespread across test shapes. When h and f are shifted outside their training ranges without retraining (40 problems per category; Appendix I), the macro-averaged error of the released NGF checkpoints rises from 0.241 (Table 2) to 0.615, and ours from 0.193 to 0.263 on the same problems. On a Laplace-only track with the same boundary data, ours averages 0.099 against 0.60–0.64 for NGF, which suggests that most of our remaining degradation comes from the source-conditioned lift.

Kernel as a harmonic-measure density. A direct test of whether K_θ approximates the true harmonic-measure density is to evaluate its boundary integral against *analytically harmonic* h, where $u_h(p) \equiv h(p)$ exactly by uniqueness of the harmonic extension. Table 4 reports rel-L_2 errors for $h \in \{x,\ xy,\ x^2 - y^2,\ Y_{2,0},\ e^x \cos y,\ e^x \sin y\}$ ($Y_{2,0}$ is the degree-2 zonal solid spherical harmonic) across all five categories. Errors are small and ordered consistently with a true harmonic-measure density (smoothest probes lowest, second-order spherical harmonics highest), and the ordering is

Table 3: Per-shape distribution of relative L_2 error (ours, 320-pair test set).

	Mean	Median	p95
Nut	0.216	0.215	0.329
Gear	0.188	0.144	0.378
Motor	0.284	0.265	0.450
Fitting	0.147	0.111	0.309
Screws	0.131	0.103	0.315

Table 4: Synthetic Laplace probe: relative L_2 error of $u_h(p) = \langle h, K_\theta(p, \cdot)\rangle$ against analytical $u_h(p) \equiv h(p)$, mean over 20 unseen test shapes per category, 64 interior queries per shape.

	x	xy	x^2-y^2	$Y_{2,0}$	$e^x \cos y$	$e^x \sin y$
Nut	0.149	0.190	0.172	0.171	0.043	0.094
Gear	0.048	0.089	0.090	0.097	0.021	0.068
Motor	0.121	0.184	0.216	0.213	0.044	0.149
Fitting	0.091	0.167	0.161	0.153	0.029	0.118
Screws	0.064	0.178	0.112	0.110	0.025	0.218

preserved on the harder Motor and Fitting geometries; part of the remaining Poisson error (Table 2) therefore stems from the source lift.

5.4 Runtime

Every method first turns a new geometry into the representation it computes on. For NHMO, this geometry step encodes the shape and evaluates K_{eff} once, playing the role that meshing plays for mesh-based pipelines (Table 5).

Table 5: Runtime on a single A100. The geometry step runs once per shape: shape encoding and K_{eff} for NHMO, tetrahedral meshing at the released resolution (fTetWild, CPU) for the MCB-B baselines.

	Geometry step (per shape)		Per problem	
Setting	NHMO	baselines	NHMO	baselines
2D MNIST (128^2)	6.6 s	grid input	4.0 ms	7.6–13.2 ms
3D Nut	7.9 s	37 s (tet meshing)	7.5 ms	0.24 s (NGF)
3D Motor	10.3 s	48 s (tet meshing)	7.9 ms	0.25 s / 77 ms (NGF)

FEM and all MCB-B baselines, including NGF, take as input the vertices of the tetrahedral mesh that the benchmark provides. On a new shape, fTetWild [Hu et al., 2020] takes 37/48 s to mesh the Nut/Motor surfaces at the released resolution, while our geometry step takes 7.9/10.3 s from the boundary surface alone, so NHMO is faster than the mesh-based pipelines from the first problem. NGF's formulation does not use mesh connectivity, but any other interior point set would also require a comparable geometry step. Per problem, NHMO solves in 7.5/7.9 ms, against 0.24/0.25 s for NGF's released pipeline (including data loading) and 77 ms per forward pass when the Motor mesh is preloaded on the GPU and only the boundary data change. In 2D, our geometry step takes 6.6 s per shape. Workloads that query one geometry many times, such as parametric boundary-condition studies, load sweeps on a fixed part, and uncertainty quantification, benefit the most: sweeping 1,000 load cases on one Motor shape takes NHMO about 18 s, geometry step included, against about 77 s for NGF with a preloaded mesh. Because the geometry step depends only on the shape, it can also be run ahead of time for a library of shapes. The 3D timings use inference-only optimizations whose effect on accuracy is within sampling noise (Appendix J).

5.5 Ablations

The full table and per-ablation discussion are in Appendix K.

(1) Accuracy is independent of geometric representation. Swapping the geometry encoder between a point-cloud over boundary samples and a 2D SDF image moves median rel-L_2 by only 0.27% on test and 0.09% on OOD, and the OOD/test gap tightens to $1.14\times$ (from canonical $1.25\times$). Setup in Appendix K.9. **(2) Mixed-corpus generalization across all 10 MNIST classes.** A single $K_\theta + v_\varphi$ shared across all 10 MNIST classes attains a per-class mean spread of only 0.53% (max $-$ min, test). **(3) Factorization, not the lift.** The kernel-only variant ($v_\varphi \equiv 0$) already beats the nonlinear end-to-end baselines on OOD (Table 1); the lift is a small correction. **(4) Robustness to the boundary quadrature.** Varying n_{surf} from 50 to 400 changes the median by $<0.1\%$ above 100 samples. **(5) Not tuned on a knife-edge.** Doubling lift parameters from 6.4M to 11.3M gives no in-distribution

gain; KDE σ is robust across a $5\times$ range; WoS supervision converges above $\sim$1,000 samples per query.

6 Discussion

Why a boundary density and an amortized source field. Green's-function operators such as NGF learn one kernel $G_\Omega(p, q)$ and integrate it against f in the volume and its normal derivative against h on the boundary. We learn the boundary density and the integrated source field instead, for three reasons. *Supervision*: WoS exit points sample ω_p, so K_θ is trained from walks alone, whereas learned-G methods rely on solver-generated solution fields. *Boundary accuracy*: near $\partial\Omega$ the value of G_Ω vanishes and the signal sits in its normal derivative, so a learned G must be accurate enough to be differentiated there. *Inference cost*: a learned G needs a new singular volume quadrature, $O(N_p N_q)$ network evaluations, whenever f changes. In a direct test (Appendix K.10), a learned volumetric integrand with a $\log|p-q|$ singularity feature reaches 6.7% / 5.2% (mean / median), no better than a source-only field lift, and its boundary derivative is not a valid Poisson kernel. NGF makes this integral fast with a rank-constrained bilinear factorization, and its heavy error tails (§5.2) are consistent with that rank limit.

The boundary-data dependence of the lift. In the classical balayage split the source-only piece does not depend on h, whereas our 2D lift sees h and u_h, because K_θ is a fitted density: the boundary term leaves the residual $e_h(p) = \int_{\partial\Omega} h\,(d\omega_p/d\sigma - K_\theta)\,d\sigma$, a linear functional of h that a network seeing only (Ω, f) cannot correct. On the 205 Laplace test pairs the source contribution vanishes and the output of the lift is its boundary correction alone: it correlates with e_h at median 0.97 and removes 71% of it (Appendix K.1). The two roles can also be separated into a source lift $v_\varphi(\Omega, f)$, which sees only the mask, its SDF, and f, and a residual head $r(\Omega, h, u_h)$ trained on top of it, giving $u = \langle h, K_\theta\rangle + v_\varphi(\Omega, f) + r(\Omega, h, u_h)$. The source lift alone reaches 6.1% / 4.5% (test mean / median) and degrades only $1.04\times$ under the OOD shift; adding r gives 1.84% / 1.74% on test and 2.56% / 2.39% on OOD, which matches or slightly surpasses the two-term model (2.1% / 2.0% and 2.6% / 2.5%) at a larger total capacity, with the h-dependence confined to an explicit corrector. The 3D lift sees only the geometry and the source, as in the classical split. The residual comes mainly from the training signal of the 2D kernel, whose KDE targets are built on simplified polyline contours rather than on the rasterized masks (Appendix K.11); placing the KDE nodes on the mask contour lowers the kernel-only error from 5.9% to 3.2%, and a kernel-plus-lift model trained on this kernel reaches 1.8% / 1.7% on test and 2.1% / 2.2% on OOD; we leave this choice of representation, and residual heads that exploit the linearity of e_h, to future work.

7 Conclusion

We proposed Neural Harmonic Measure Operator (NHMO), to our knowledge the first neural operator built explicitly around the harmonic measure, the canonical probability distribution from potential theory that mediates all solutions of the Dirichlet Laplace problem on a fixed domain. For Poisson source terms, the classical balayage decomposition extends the same harmonic measure to handle sources via a learned lift network with a zero-boundary gauge. Our framework reduces an end-to-end neural-operator problem to two coupled components: the boundary kernel K_θ, independently falsifiable as a harmonic-measure density via synthetic-harmonic probes, and the amortized balayage source lift v_φ. More broadly, our work suggests that grounding neural operators in canonical objects from classical analysis, rather than learning end-to-end input-to-output mappings, offers a path to inductive biases that mirror the structure of the underlying PDE, and we hope this framing motivates further work at the interface of potential theory and neural operator learning.

Limitations and future work. We target Dirichlet elliptic problems; Neumann/Robin conditions and other PDE types need generalized measures and remain future work. The lift is f-conditional (through the source field in 2D and probe samples in 3D), so out-of-distribution sources may degrade. Linearity in h is guaranteed only for the kernel channel: a lift that learns shortcuts specific to the training coefficients would lose OOD robustness. NHMO also trades a per-shape precompute for fast per-problem inference, so single-problem-per-shape workloads do not benefit from the cache; future work could explore low-rank kernel factorizations to amortize this cost.

Acknowledgments and Disclosure of Funding

We sincerely thank the reviewers for their valuable feedback. Georgia Tech authors acknowledge NSF CAREER #2420319, IIS #2433307, OISE #2433313, IIS #2433322, ECCS #2318814, and CNS #2450401 for funding support. We thank NVIDIA for providing computing resources through the NVIDIA Academic Grant. The authors declare no competing interests.

References

Benedikt Alkin, Andreas Fürst, Simon Schmid, Lukas Gruber, Markus Holzleitner, and Johannes Brandstetter. Universal physics transformers: A framework for efficiently scaling neural operators. *Advances in Neural Information Processing Systems*, 37:25152–25194, 2024.

David H Armitage and Stephen J Gardiner. *Classical potential theory*. Springer Science & Business Media, 2012.

Martin Philip Bendsoe and Ole Sigmund. *Topology optimization: theory, methods, and applications*. Springer Science & Business Media, 2013.

Ilia Binder and Mark Braverman. The rate of convergence of the Walk on Spheres algorithm. *Geometric and Functional Analysis*, 22(3):558–587, 2012. doi: 10.1007/s00039-012-0161-z.

Nicolas Boullé, Christopher J Earls, and Alex Townsend. Data-driven discovery of green's functions with human-understandable deep learning. *Scientific reports*, 12(1):4824, 2022.

Heinz Werner Engl, Martin Hanke, and Andreas Neubauer. *Regularization of inverse problems*, volume 375. Springer Science & Business Media, 1996.

John B Garnett and Donald E Marshall. *Harmonic measure*. Number 2. Cambridge University Press, 2005.

Craig R Gin, Daniel E Shea, Steven L Brunton, and J Nathan Kutz. Deepgreen: deep learning of green's functions for nonlinear boundary value problems. *Scientific reports*, 11(1):21614, 2021.

Zhongkai Hao, Zhengyi Wang, Hang Su, Chengyang Ying, Yinpeng Dong, Songming Liu, Ze Cheng, Jian Song, and Jun Zhu. Gnot: A general neural operator transformer for operator learning. In *International conference on machine learning*, pages 12556–12569. PMLR, 2023.

Yixin Hu, Teseo Schneider, Bolun Wang, Denis Zorin, and Daniele Panozzo. Fast tetrahedral meshing in the wild. *ACM Transactions on Graphics*, 39(4), 2020. doi: 10.1145/3386569.3392385.

Tianyu Huang, Jingwang Ling, Shuang Zhao, and Feng Xu. Guiding-based importance sampling for walk on stars. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, pages 1–12, 2025.

Thomas JR Hughes. *The finite element method: linear static and dynamic finite element analysis*. Courier Corporation, 2003.

Shizuo Kakutani. 143. two-dimensional brownian motion and harmonic functions. *Proceedings of the Imperial Academy*, 20(10):706–714, 1944.

Sangpil Kim, Hyung-gun Chi, Xiao Hu, Qixing Huang, and Karthik Ramani. A large-scale annotated mechanical components benchmark for classification and retrieval tasks with deep neural networks. In *European conference on computer vision*, pages 175–191. Springer, 2020.

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.

Randall J LeVeque. *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems*. SIAM, 2007.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020a.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020b.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Andrew Stuart, Kaushik Bhattacharya, and Anima Anandkumar. Multipole graph neural operator for parametric partial differential equations. *Advances in Neural Information Processing Systems*, 33:6755–6766, 2020c.

Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for pdes on general geometries. *Journal of Machine Learning Research*, 24(388):1–26, 2023.

Guochang Lin, Pipi Hu, Fukai Chen, Xiang Chen, Junqing Chen, Jun Wang, and Zuoqiang Shi. Binet: learning to solve partial differential equations with boundary integral networks. *arXiv preprint arXiv:2110.00352*, 2021.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.

Huakun Luo, Haixu Wu, Hang Zhou, Lanxiang Xing, Yichen Di, Jianmin Wang, and Mingsheng Long. Transolver++: An accurate neural solver for pdes on million-scale geometries. *arXiv preprint arXiv:2502.02414*, 2025.

Nikolai G Makarov. On the distortion of boundary sets under conformal mappings. *Proceedings of the London Mathematical Society*, 3(2):369–384, 1985.

Michael Mascagni and Chi-Ok Hwang. ϵ-shell error analysis for "Walk On Spheres" algorithms. *Mathematics and Computers in Simulation*, 63(2):93–104, 2003. doi: 10.1016/S0378-4754(03)00038-7.

Bailey Miller, Rohan Sawhney, Keenan Crane, and Ioannis Gkioulekas. Boundary value caching for walk on spheres. *arXiv preprint arXiv:2302.11825*, 2023.

Bailey Miller, Rohan Sawhney, Keenan Crane, and Ioannis Gkioulekas. Differential walk on spheres. *ACM Transactions on Graphics (TOG)*, 43(6):1–18, 2024.

Mervin E Muller. Some continuous monte carlo methods for the dirichlet problem. *The Annals of Mathematical Statistics*, pages 569–589, 1956.

Hong Chul Nam, Julius Berner, and Anima Anandkumar. Solving poisson equations using neural walk-on-spheres. *arXiv preprint arXiv:2406.03494*, 2024.

Pawan Negi, Maggie Cheng, Mahesh Krishnamurthy, Wenjun Ying, and Shuwang Li. Learning domain-independent green's function for elliptic partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 421:116779, 2024.

Stefan A Sauter and Christoph Schwab. Boundary element methods. In *Boundary Element Methods*, pages 183–287. Springer, 2010.

Rohan Sawhney and Keenan Crane. Monte Carlo geometry processing: a grid-free approach to PDE-based methods on volumetric domains. *ACM Transactions on Graphics (TOG)*, 39(4):123:1–123:18, 2020.

Rohan Sawhney and Bailey Miller. Zombie: Grid-free monte carlo solvers for partial differential equations, 2023.

Rohan Sawhney, Dario Seyb, Wojciech Jarosz, and Keenan Crane. Grid-free monte carlo for pdes with spatially varying coefficients. *ACM Transactions on Graphics (TOG)*, 41(4):1–17, 2022.

Rohan Sawhney, Bailey Miller, Ioannis Gkioulekas, and Keenan Crane. Walk on stars: A grid-free monte carlo method for pdes with neumann boundary conditions. *arXiv preprint arXiv:2302.11815*, 2023.

Ralph C Smith. *Uncertainty quantification: theory, implementation, and applications*. SIAM, 2024.

Ryusuke Sugimoto, Nathan King, Toshiya Hachisuka, and Christopher Batty. Projected walk on spheres: A monte carlo closest point method for surface pdes. In *SIGGRAPH Asia 2024 Conference Papers*, pages 1–10, 2024.

Jia Sun, Yinghua Liu, Yizheng Wang, Zhenhan Yao, and Xiaoping Zheng. Binn: A deep learning approach for computational mechanics problems based on boundary integral equations. *Computer Methods in Applied Mechanics and Engineering*, 410:116012, 2023.

Joao Teixeira, Eitan Grinspun, and Otman Benchekroun. Variational green's functions for volumetric pdes. *arXiv preprint arXiv:2602.12349*, 2026.

Yuankai Teng, Xiaoping Zhang, Zhu Wang, and Lili Ju. Learning green's functions of linear reaction-diffusion equations with application to fast numerical solver. In *Mathematical and Scientific Machine Learning*, pages 1–16. PMLR, 2022.

Haixin Wang, Jiaxin Li, Anubhav Dwivedi, Kentaro Hara, and Tailin Wu. Beno: Boundary-embedded neural operators for elliptic pdes. *arXiv preprint arXiv:2401.09323*, 2024.

Tian Wang and Chuang Wang. Latent neural operator for solving forward and inverse PDE problems. In *Advances in Neural Information Processing Systems*, 2024.

Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A fast transformer solver for pdes on general geometries. *arXiv preprint arXiv:2402.02366*, 2024.

Haixu Wu, Minghao Guo, Zongyi Li, Zhiyang Dou, Mingsheng Long, Kaiming He, and Wojciech Matusik. Geopt: Scaling physics simulation via lifted geometric pre-training. *arXiv preprint arXiv:2602.20399*, 2026.

Zipeng Xiao, Zhongkai Hao, Bokai Lin, Zhijie Deng, and Hang Su. Improved operator learning by orthogonal attention. *arXiv preprint arXiv:2310.12487*, 2023.

Minglang Yin, Nicolas Charon, Ryan Brody, Lu Lu, Natalia Trayanova, and Mauro Maggioni. Dimon: Learning solution operators of partial differential equations on a diffeomorphic family of domains. *arXiv preprint arXiv:2402.07250*, 2024.

Seungwoo Yoo, Kyeongmin Yeo, Jisung Hwang, and Minhyuk Sung. Neural green's functions. *arXiv preprint arXiv:2511.01924*, 2025.

Emanuele Zappala, Antonio Henrique de Oliveira Fonseca, Josue Ortega Caro, Andrew Henry Moberly, Michael James Higley, Jessica Cardin, and David van Dijk. Learning integral operators via neural integral equations. *Nature Machine Intelligence*, 6(9):1046–1062, 2024.

Rui Zhang, Qi Meng, Rongchan Zhu, Yue Wang, Wenlei Shi, Shihua Zhang, Zhi-Ming Ma, and Tie-Yan Liu. Monte carlo neural pde solver for learning pdes via probabilistic representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

Hang Zhou, Haixu Wu, Haonan Shangguan, Yuezhou Ma, Huikun Weng, Jianmin Wang, and Mingsheng Long. Transolver-3: Scaling up transformer solvers to industrial-scale geometries. *arXiv preprint arXiv:2602.04940*, 2026.

Appendix

A Notation **15**

B Harmonic measure: derivations and properties **15**

C Architecture details **16**

D Loss specifications and training schedule **17**

D.1 Kernel losses (Stage 1) . . . 17

D.2 Loss-weight schedule . . . 18

D.3 Stage 1 optimizer and LR schedule . . . 18

D.4 Stage 2 optimizer and LR schedule, with warm-start . . . 18

D.5 WoS supervision budget and variance . . . 19

D.6 Training cost . . . 19

E Baseline implementations **19**

F Intuitive demonstrations: details **20**

F.1 3D harmonic on common graphics meshes . . . 20

F.2 Drift adaptation on a bunny slice . . . 21

G 2D MNIST benchmark: setup, NGF port, and additional qualitative **22**

G.1 Setup details . . . 22

G.2 NGF 2D port . . . 23

G.3 Additional qualitative comparisons . . . 23

H Additional MCB-B qualitative comparisons **25**

I 3D coefficient-OOD study **26**

J Inference speed: caching is implied by the factorization **26**

K Ablations: detail **27**

K.1 Separating the lift's roles: source lift and residual head . . . 27

K.2 Lift removal (kernel-only vs. kernel + lift) . . . 27

K.3 Lift capacity (6.4M vs 11.3M parameters) . . . 28

K.4 KDE bandwidth σ for Walk-on-Spheres supervision . . . 28

K.5 Training-shape count and single mixed-corpus generalization . . . 28

K.6 Walk-on-Spheres sample count . . . 28

K.7 Boundary quadrature resolution at inference (n_{surf} scan) . . . 28

K.8 Per-MNIST-class breakdown (single mixed-corpus uniformity) . . . 29

K.9 Representation invariance: SDF vs. point-cloud encoder . . . 29

K.10 Learned volumetric integrand . . . 30

K.11 Origin of the kernel-fit residual . . . 30

A Notation

Table 6: Notation used throughout the paper.

Symbol	Meaning
Geometry	
$\Omega \subset \mathbb{R}^d$	bounded Lipschitz domain in dimension $d \in \{2, 3\}$
$\partial\Omega$	boundary of Ω
$p, q \in \Omega$	interior points
$\zeta \in \partial\Omega$	boundary point
ν_ζ	outward unit normal to $\partial\Omega$ at ζ
$d\sigma$	surface measure on $\partial\Omega$
$\mathrm{SDF}(p)$	signed distance to $\partial\Omega$ (negative inside Ω)
PDE data	
$h \in C(\partial\Omega)$	Dirichlet boundary data
$f \in L^\infty(\Omega)$	source term in the Poisson problem $\Delta u = f$
u	PDE solution
Classical potential-theoretic objects	
ω_p	harmonic measure at p (probability measure on $\partial\Omega$)
$d\omega_p/d\sigma$	harmonic-measure density (Poisson kernel), equal to $-\partial_\nu G_\Omega(p, \cdot)$
$G_\Omega(p, q)$	Dirichlet Green's function on Ω
Φ	fundamental solution of $-\Delta$ on $\mathbb{R}^d$
N_f	Newtonian potential of f
u_f	particular Poisson solution with $u_f\|_{\partial\Omega} = 0$
B_t, τ	Brownian motion in $\mathbb{R}^d$, first-exit time from Ω
Learned objects	
$K_\theta(p, \zeta; \Omega)$	learned harmonic-measure density (NHMO kernel, quadrature-normalized)
$\widetilde{K}_\theta$	pre-normalization network output (cf. soft normalization)
v_φ	learned field lift: $v_\varphi(p; \Omega, h, f)$ with input u_h in 2D; $v_\varphi(p; \Omega, f)$ in 3D (zero boundary gauge) and in the variant of §6
$r(p; \Omega, h, u_h)$	learned residual head (2D)
$e_h(p)$	kernel-fit residual $\int_{\partial\Omega} h\,(d\omega_p/d\sigma - K_\theta)\,d\sigma$
ψ_Ω	learned shape latent
$u_h(p)$	boundary integral $\sum_i w_i\, K_\theta(p, \zeta_i; \Omega)\, h(\zeta_i)$
Discretization	
$\{\zeta_i\}_{i=1}^{N_s}$	surface quadrature samples on $\partial\Omega$
$\{w_i\}_{i=1}^{N_s}$	surface quadrature weights ($w_i \propto \sigma(\partial\Omega)/N_s$)
$\{q_j\}_{j=1}^{N_p}$	interior source-probe samples
Abbreviations	
OOD	out-of-distribution (evaluation split with BC coefficients outside training)
KDE	kernel-density estimate (of WoS exit points)
GT / GF	ground truth (numerical reference) / Green's-function-style baseline

B Harmonic measure: derivations and properties

This appendix collects the classical facts about ω_p deferred from §3.1.

Existence/uniqueness and Riesz representation. For $h \in C(\partial\Omega)$, the Dirichlet problem

$$\Delta u = 0 \text{ in } \Omega, \qquad u = h \text{ on } \partial\Omega \tag{8}$$

admits a unique solution $u \in C(\overline{\Omega}) \cap C^2(\Omega)$ on a bounded Lipschitz domain. Linearity in h together with the maximum principle make $h \mapsto u(p)$ a positive linear functional on $C(\partial\Omega)$ for each fixed $p \in \Omega$. The Riesz representation theorem then yields a unique probability measure ω_p on $\partial\Omega$ such that $u(p) = \int_{\partial\Omega} h\, d\omega_p$, recovering (2) [Garnett and Marshall, 2005]. Positivity and total mass one are automatic from this construction, and combined with (2) they give the maximum principle $\min h \le u \le \max h$.

Green's-function trace formula. When $\partial\Omega$ is sufficiently regular (smooth, C^1, or more generally Lipschitz [Garnett and Marshall, 2005]), ω_p is absolutely continuous with respect to surface measure $d\sigma$, and its Radon–Nikodym density coincides σ-almost everywhere with the (negated) nontangential outward-normal derivative of the Dirichlet Green's function:

$$\frac{d\omega_p}{d\sigma}(\zeta) \;=\; -\partial_{\nu_\zeta} G_\Omega(p,\zeta) \quad (\sigma\text{-a.e. on } \partial\Omega), \tag{9}$$

where ν_ζ is the outward unit normal at ζ. The right-hand side is non-negative because G_Ω is positive in Ω and zero on $\partial\Omega$. The kernel K_θ approximates this density, so $K_\theta\, d\sigma$ approximates ω_p, and the quadrature weights w_i in (7) discretize $d\sigma$.

Walk-on-Spheres as a sampler of ω_p. For the isotropic Laplacian, Brownian motion started at the center of a ball contained in Ω leaves the ball at a uniformly distributed point of its sphere (the mean-value property). WoS chains such jumps, each on the largest sphere around the current point that fits in Ω, so an ideal walk draws its exit point exactly from ω_p, and the average of h over exit points has expectation $u(p) = \mathbb{E}_p[h(B_\tau)]$ for any number of walks [Muller, 1956]. WoS samples the exit law directly and integrates no density, so the surface measure enters only when the learned density is integrated with the quadrature weights w_i. The implemented walk carries three small biases: the ε-shell termination, whose bias is $O(\varepsilon)$ on our Lipschitz domains [Mascagni and Hwang, 2003]; the step cap, with non-terminating walks masked out (their fraction is reported in Appendix D.5); and the rasterized SDF used to compute sphere radii. The expected number of steps grows as $O(\log(1/\varepsilon))$ with geometry-dependent constants [Binder and Braverman, 2012]. The uniform-sphere jump relies on the Euclidean, isotropic Laplacian; drift or varying coefficients need transformed walks, and our drift demonstration (Appendix F.2) uses a Yukawa-type transform.

Further properties. In 2D, ω_p is invariant under conformal maps of Ω [Garnett and Marshall, 2005]. Its dimensional properties characterize boundary regularity [Makarov, 1985]. Neither property is used in our construction; we record them only for completeness.

C Architecture details

This section gives concrete dimensions for the canonical 2D MNIST configuration, followed by the 3D MCB-B configuration.

Shape encoder. A Transolver-style slice-attention module. Boundary samples (point + outward normal) are augmented with Fourier features ($L = 8$ bands per axis) and a learned boundary / interior-anchor type embedding, then projected to $d_{\text{model}} = 256$ tokens. A soft slice projection produces $M = 64$ slice tokens (independent of the input boundary density), followed by a 3-layer pre-norm transformer encoder with 4 heads and MLP ratio 4. The SDF variant referenced in §K.9 replaces the point-cloud stem with a 3-block CNN over a 64×64 SDF grid that yields a 16×16 token grid (also $d_{\text{model}} = 256$); all downstream hyperparameters are held identical.

Kernel head. Cross-attention with $n = 2$ pre-norm layers, 4 heads, $d_{\text{model}} = 256$, MLP ratio 4. The query token is built from Fourier features of p ($L = 10$). Boundary tokens are queries; their inputs are Fourier features of ζ ($L = 10$) plus the outward normal ($L = 4$) and the displacement $\Delta = \zeta - p$ ($L = 4$ plus the raw vector). The cross-attention context is the encoder output $Z(\Omega)$ concatenated with the query token. Read-out is a 2-layer MLP onto a scalar logit per boundary sample, normalized via softmax weighted by the surface quadrature weights so that $\sum_i w_i\, K_\theta(p, \zeta_i; \Omega) = 1$. Logit clipping (the $\log K_{\max}$ cap of earlier configurations) is disabled in the canonical configuration.

Field lift. A symmetric 2D U-Net with input channels $(\mathbf{1}_\Omega, h, f, u_h)$, base width 48, depth 4, GroupNorm (8 groups), GELU. Three down-blocks take channels $48 \to 96 \to 192 \to 384$ at spatial resolutions $128 \to 64 \to 32 \to 16$; a middle block; three up-blocks with skip connections; a 1×1 output projection. The output is multiplied by the interior mask. The 3D lift is described below. The variant of §6 replaces the field lift by a source lift with input channels $(\mathbf{1}_\Omega, \mathrm{SDF}, f)$ and base width 64 ($\approx$11.3M parameters) and a residual head with input channels $(\mathbf{1}_\Omega, h, u_h)$ and the widths above ($\approx$6.4M parameters).

Parameter counts. The canonical 2D model totals ≈11.2M parameters: shape encoder ≈3.1M, kernel head ≈1.7M, field lift ≈6.4M.

3D configuration (MCB-B). The shape encoder uses $d_{\text{model}} = 192$, 64 slice tokens, 4 transformer layers, 4 heads, and Fourier features with 10 bands; the kernel head uses 2 cross-attention layers at $d_{\text{model}} = 192$ with Fourier bands 10 for p and ζ and 4 for the normal, and a soft $\tanh$ cap of the log-density at $\log K_{\max} = 15$ (kernel total 2.77M parameters). The 3D lift is a cross-attention head at $d_{\text{model}} = 192$ with 3 cross-attention layers, whose query is a Fourier embedding of p and whose context is the frozen shape latent together with 256 source tokens (384 for Fitting) produced by a slice aggregator over source samples $(q_j, f(q_j))$. Its output is multiplied by $\max(0, -\text{SDF}(p))$, so it vanishes on $\partial\Omega$. It receives neither h nor u_h.

D Loss specifications and training schedule

This section gives the explicit forms of the four kernel-training loss terms ($\mathcal{L}_{\text{NLL}}, \mathcal{L}_{\text{MV}}, \mathcal{L}_{\text{BL}}, \mathcal{L}_Z$), the loss-weight schedule actually used to obtain the reported numbers, and the optimizer / learning-rate setup for both training stages, followed by the WoS supervision budget and the training cost.

D.1 Kernel losses (Stage 1)

For each interior query point p, the network output is $\widetilde{K}_\theta(p, \zeta; \Omega)$; during training it is kept close to unit mass by $\mathcal{L}_Z$ below, and at inference it is normalized over the boundary quadrature (§4.3). With a surface quadrature $\{(\zeta_i, w_i)\}_{i=1}^{N_s}$ on $\partial\Omega$, define

$$\log Z(p) \;=\; \log \sum_{i=1}^{N_s} w_i \exp(\log \widetilde{K}_\theta(p, \zeta_i; \Omega)), \tag{10}$$

the log of the kernel's surface mass.

$\mathcal{L}_{\text{NLL}}$ (WoS exit-point likelihood). Walk-on-Spheres simulation provides a boundary exit point ζ_i^* for each interior anchor p_i. We supervise

$$\mathcal{L}_{\text{NLL}} \;=\; -\frac{1}{|S_{\text{valid}}|} \sum_{i \in S_{\text{valid}}} \log K_\theta(p_i, \zeta_i^*; \Omega), \tag{11}$$

averaged over WoS walks that reached the boundary inside the step budget; otherwise the entry is masked. In 3D, the kernel is trained with this loss and the three losses below. In 2D, the exit points of 10^4 precomputed walks per probe are smoothed into a Gaussian KDE with bandwidth σ evaluated at 512 boundary nodes. At each step we sample 64 of these nodes, normalize both the kernel and the target over them, and minimize the KL divergence from the target plus 0.5 times the L_1 distance between the two densities; the 2D kernel uses no $\mathcal{L}_{\text{MV}}$, $\mathcal{L}_{\text{BL}}$, or $\mathcal{L}_Z$.

$\mathcal{L}_{\text{MV}}$ (mean-value martingale). Treating $K_\theta(\cdot, \zeta)$ as a (signed) function of the interior point, the mean-value property requires

$$\mathcal{L}_{\text{MV}} \;=\; \mathbb{E}_{p,\zeta,r}\left[\left(K_\theta(p, \zeta) - \tfrac{1}{S}\sum_{s=1}^{S} K_\theta(p + r\mathbf{d}_s, \zeta)\right)^2\right], \tag{12}$$

with S sphere samples $\mathbf{d}_s$ drawn uniformly on S^{d-1} and radius r log-uniform in $[0.2,\, 0.9] \cdot (-\text{SDF}(p))$. Both $K_\theta(p, \zeta)$ and the $K_\theta(p + r\mathbf{d}_s, \zeta)$ are normalized internally via the surface quadrature so that the discrepancy compares densities on a common scale. Batch entries for which $-\text{SDF}(p)$ falls below $10^{-3} \cdot \text{diag}(\text{bbox})$ are masked out (sphere-degenerate regime). MCB experiments use $S = 32$.

$\mathcal{L}_{\text{BL}}$ (boundary-limit peak). For each surface anchor $\zeta_0 \in \partial\Omega$ with outward unit normal ν_{ζ_0}, we place a probe point $p_\epsilon = \zeta_0 - \epsilon\nu_{\zeta_0}$ just inside Ω, with ϵ adapted iteratively until $\text{SDF}(p_\epsilon) < -\epsilon/2$.

The harmonic measure ω_{p_ϵ} should concentrate at ζ_0, so we drive $K_\theta(p_\epsilon, \zeta_0)$ up while penalizing mass placed away from ζ_0:

$$\mathcal{L}_{\text{BL}} \;=\; -\log K_\theta(p_\epsilon, \zeta_0; \Omega) \;+\; \gamma \sum_{i:\, \|\zeta_i - \zeta_0\| > \delta} w_i \, K_\theta(p_\epsilon, \zeta_i; \Omega), \tag{13}$$

with defaults $\epsilon = 0.02$, $\gamma = 1.0$, $\delta = 0.1$. The summation uses the same surface quadrature as $\log Z$ except for the entry at ζ_0, which is excluded.

$\mathcal{L}_Z$ (soft mass normalization). We penalize deviation of $\log Z(p)$ from 0 via a Huber loss

$$\mathcal{L}_Z \;=\; \text{Huber}_\delta(\log Z(p)) \;=\; \begin{cases} (\log Z(p))^2 & |\log Z(p)| \le \delta \\ 2\delta\,|\log Z(p)| - \delta^2 & |\log Z(p)| > \delta \end{cases} \tag{14}$$

with $\delta = 1$. Huber prevents spikes in $\log \widetilde{K}_\theta$ from producing outsized updates (an instability we observed under a pure ℓ_2 penalty).

D.2 Loss-weight schedule

The total kernel loss is $\mathcal{L}_K = \lambda_{\text{NLL}}\mathcal{L}_{\text{NLL}} + \lambda_{\text{MV}}\mathcal{L}_{\text{MV}} + \lambda_{\text{BL}}\mathcal{L}_{\text{BL}} + \lambda_Z\mathcal{L}_Z$. The weights are step-dependent:

Step range	λ_{NLL}	λ_{MV}	λ_{BL}	λ_Z
$[0,\ 10\text{k})$ *(warm-in)*	1.0	0.1	0.5	1.0
$[10\text{k},\ 80\text{k})$ *(main)*	1.0	1.0	0.5	1.0
$[80\text{k},\ \infty)$ *(MV-emphasis)*	0.5	2.0	0.5	1.0

The warm-in stage suppresses $\mathcal{L}_{\text{MV}}$ at initialization, where the spherical-average targets and the kernel at p are both moving and the loss can dominate the NLL signal before either has a useful shape. MCB-B Stage 1 runs for 30,000 steps total, so only the warm-in and main ranges are reached for the headline numbers in §5.3; the post-80k MV-emphasis branch is provided in code but is not used to obtain reported MCB results.

D.3 Stage 1 optimizer and LR schedule

AdamW with weight decay 0.01 on linear weights (no decay on biases / norm parameters). Linear warmup over $T_w = 1000$ steps to $\text{lr}_{\max} = 3 \cdot 10^{-4}$, then cosine decay to $\text{lr}_{\min} = 10^{-5}$ over total $T = 30{,}000$ steps; gradient clipping at global norm 1.0. Per step, the kernel sees $N_s = 2000$ surface samples and $N_p = 512$ interior anchors, with $B_q = 8$ queries per shape and one shape per gradient step. The kernel head's MLP score is soft-clipped via $\tanh$ to $\log \widetilde{K}_\theta(p, \zeta; \Omega) \in [-\log K_{\max},\ \log K_{\max}]$ with $\log K_{\max} = 15$. These are the 3D settings. The 2D kernel is trained for 60,000 steps with AdamW and a one-cycle cosine schedule (warmup 500 steps, peak learning rate $3 \cdot 10^{-4}$, final 10^{-5}) and no logit cap.

D.4 Stage 2 optimizer and LR schedule, with warm-start

With K_θ frozen, v_φ is trained against the masked pixel-wise mean-squared error

$$\mathcal{L}_v \;=\; \frac{1}{|\Omega|} \sum_{p \in \Omega} \big(u_{\text{pred}}(p) - u_{\text{true}}(p)\big)^2, \tag{15}$$

in y-normalized space, where $u_{\text{pred}} = u_h + v_\varphi$ via the frozen kernel and $|\Omega|$ counts interior pixels. We use AdamW (weight decay 0.01, default betas) with the same linear-warmup-then-cosine schedule from §D.3 but $T_w = 200$ and a per-category total step count (typically 30,000–50,000). Gradient clipping is unchanged. The lift consumes $N_p = 512$ source-probe samples per problem.

The schedule is extended by warm-start: we reload the previous-best lift weights and resume under a fresh cosine schedule (optimizer state and step counter reset). The hardest categories reach ~ 100k effective steps after one or two warm-start rounds.

In 2D, the field lift is trained for $10{,}000$ steps on the Poisson pairs. For the variant of §6, the source lift is trained with the same loss for $30{,}000$ steps on Laplace and Poisson pairs, and the residual head is then trained for $10{,}000$ steps on $u_{\text{pred}} = u_h + v_\varphi + r$ with K_θ and v_φ frozen, using AdamW with weight decay 0.01 and a one-cycle cosine schedule (warmup 300 steps, peak learning rate $3 \cdot 10^{-4}$, final 10^{-5}).

D.5 WoS supervision budget and variance

Table 7 lists the WoS settings used for the reported kernels. The sampler runs on the GPU and completes $\sim 5 \times 10^8$ walks per second on an A100 even at the stricter termination $\varepsilon = 10^{-4}$. The whole 2D supervision ($5{,}000$ shapes $\times$ 32 probes $\times$ 10^4 walks, precomputed once) therefore takes seconds of GPU time, and the online supervision of one 3D category ($30{,}000$ steps $\times$ 8 probes $\times$ 4 exits $\approx 10^6$ walks) runs in under a second. Table 8 reports the per-category throughput and walk length in 3D at the training settings. Masked walks are rare (0.0095% in 2D, 0.05–0.40% per 3D category).

Table 7: WoS supervision settings of the reported kernels. Masked: fraction of walks that do not reach the ε-shell within the step cap.

	2D MNIST	3D MCB-B
ε (normalized domain)	10^{-3}	10^{-3}
step cap	128	128
walks per probe	10^4 (precomputed)	4 fresh exits per step (online)
probes	32 per shape	8 per gradient step
target	KDE, $\sigma = 0.2\%$ of domain width, 512 nodes	exit-point likelihood
masked	0.0095%	0.05–0.40% (by category)

Table 8: Per-category WoS statistics in 3D (A100): throughput and mean number of steps per walk.

	Nut	Gear	Motor	Fitting	Screws
walks per second	$7.6{\times}10^8$	$8.1{\times}10^8$	$7.5{\times}10^8$	$7.8{\times}10^8$	$7.8{\times}10^8$
mean steps per walk	14.0	12.8	15.9	12.8	14.3
masked fraction	0.125%	0.045%	0.110%	0.076%	0.402%

Variance. The standard deviation of the WoS estimate falls as $1/\sqrt{N}$ in the number of walks N: for $h = x$ it drops from 0.031 at $N = 100$ to 0.003 at $N = 10^4$. Kernel quality is stable above $\sim$1,000 walks per probe (Appendix K.6), and the error of the 2D KDE targets is unchanged with $100\times$ more walks (Appendix K.11). At the canonical 10^4 walks, supervision noise is therefore well below the kernel-fit error.

D.6 Training cost

On a single A100, the 2D kernel trains in $\sim$1 h ($60{,}000$ steps) and the 2D field lift adds $\sim$1 h; the source lift and the residual head of the variant take about 8 h and 1.5 h. A 3D kernel takes 8.8 h (Nut) to 24 h (Motor, on a shared GPU) per category ($30{,}000$ steps).

E Baseline implementations

We use the authors' published source code wherever it is available. **Transolver** [Wu et al., 2024], **LNO** [Wang and Wang, 2024], and **UPT** [Alkin et al., 2024] are run from the official public repositories of their respective papers; we adapt only the data loaders to our $(\Omega, h, f) \mapsto u$ format and otherwise keep architectures, optimizers, and training schedules at the published defaults. **BENO** [Wang et al., 2024] is run from its official implementation with the data pipeline adapted to our format; we use 512 boundary samples and train for 160 epochs at 64^2 followed by 40 epochs at 128^2, the resolution of all other methods. For the 3D MCB-B Poisson benchmark, we additionally use the dataset and reference solutions released by NGF [Yoo et al., 2025] on their public GitHub repository,

which provides the FEM tetrahedral meshes and ground-truth solutions used in their Table 2; we evaluate on the same shape and (h, f) test split, allowing direct head-to-head comparison without re-running their FEM pipeline. For the 2D MNIST benchmark, however, the NGF authors did not release a 2D code path or 2D evaluation data; we therefore ported their official 3D implementation to 2D (Appendix G.2). Numbers reported for 2D NGF reflect this port rather than the authors' code.

F Intuitive demonstrations: details

These are proof-of-concept demos used in §5.1; the formal benchmarks of §5.2 and §5.3 train per-category as standard for those protocols, so the shared-kernel framing here is specific to these demos.

F.1 3D harmonic on common graphics meshes

PDE and shapes. Dirichlet Laplace, $\Delta u = 0$ in Ω with $u|_{\partial\Omega} = h$, on four unit-cube-normalized meshes: armadillo, bunny, fandisk, lucy. Boundary data $h \in \{\sin x, \sin z\}$, giving eight test cases in total.

Ground truth. FEM solutions on volumetric tetrahedral meshes per shape. Final volumes are exported as 256^3 voxel grids of the harmonic field for high-resolution rendering, with rel-L_2 measured against the FEM reference inside the FEM interior mask.

Ours. A residual-distilled NHMO export. A single shape-conditioned harmonic kernel K_θ is fitted once across all four shapes, followed by a residual head trained against the FEM reference and distilled into a single forward pass for visualization-quality output.

Green-function-style baseline. A learned volumetric Green's function as in prior work [Yoo et al., 2025, Boullé et al., 2022, Teng et al., 2022, Negi et al., 2024, Gin et al., 2021, Li et al., 2020c, Teixeira et al., 2026], evaluated against the same FEM reference on the same volumes.

Table 9: 3D harmonic on common graphics meshes: per-case rel-L_2 against FEM reference.

shape	h	Ours	GF style	ratio
armadillo	$\sin x$	0.011	0.117	$10.4\times$
armadillo	$\sin z$	0.011	0.103	$9.2\times$
bunny	$\sin x$	0.019	0.249	$13.2\times$
bunny	$\sin z$	0.016	0.214	$13.4\times$
fandisk	$\sin x$	0.011	0.142	$12.6\times$
fandisk	$\sin z$	0.012	0.138	$11.8\times$
lucy	$\sin x$	0.006	0.123	$21.3\times$
lucy	$\sin z$	0.009	0.045	$5.1\times$
mean		**0.012**	0.142	$11.8\times$

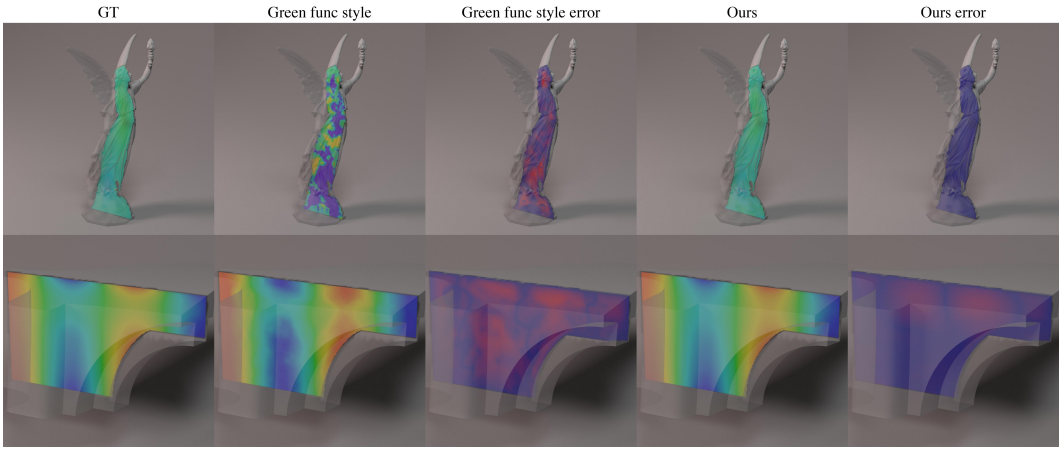


Figure 6: Additional intuitive 3D harmonic comparisons. Top: lucy. Bottom: fandisk. Same five-column layout as Figure 1.

F.2 Drift adaptation on a bunny slice

PDE. Constant-drift Laplace,

$$\Delta u + \beta \cdot \nabla u \;=\; 0 \quad \text{in } \Omega, \qquad u|_{\partial\Omega} \;=\; h, \tag{16}$$

on a 2D $y{=}0$ slice of a bunny mesh ($\Omega \subset \mathbb{R}^2$ is the slice interior). Dirichlet boundary data $h \in \{\sin 6x,\, \sin 6z\}$. The drift vectors are given in ambient coordinates; on the $y{=}0$ slice only their in-plane (x, z) components enter the PDE. Drift vectors β are listed in Table 10.

Ground truth. Computed by Walk-on-Spheres with a Yukawa-style transform that absorbs the drift as a path-dependent killing factor.

Ours. Warm-started from the same frozen K_θ used in §F.1. We attach a small drift-conditioned adapter and a residual head; the kernel itself is not retrained.

Green-function-style baseline. The same construction as in §F.1, also warm-started from the frozen Laplace kernel and conditioned on the drift parameter, but without the residual head.

Training budget. Both methods are trained under identical settings, namely 4000 optimization steps, batch size 128, a single learning rate, and an 80/20 pixel split over eight $256{\times}256$ slice cases (four drift vectors $\times$ two boundary signals). Training samples are pixels rather than fixed epochs; we therefore report this as a matched optimization-budget comparison.

Table 10: Drift adaptation: per-slice test rel-L_2 on the bunny slice.

drift β	boundary h	Ours	GF style
$(2, 0, 0)$	$\sin 6x$	0.060	0.366
$(2, 0, 0)$	$\sin 6z$	0.062	0.280
$(-2, 0, 0)$	$\sin 6x$	0.062	0.378
$(-2, 0, 0)$	$\sin 6z$	0.065	0.326
$(0, 0, 2)$	$\sin 6x$	0.069	0.364
$(0, 0, 2)$	$\sin 6z$	0.061	0.293
$(1.5, 1, 0)$	$\sin 6x$	0.060	0.360
$(1.5, 1, 0)$	$\sin 6z$	0.060	0.288
mean		**0.062**	0.323

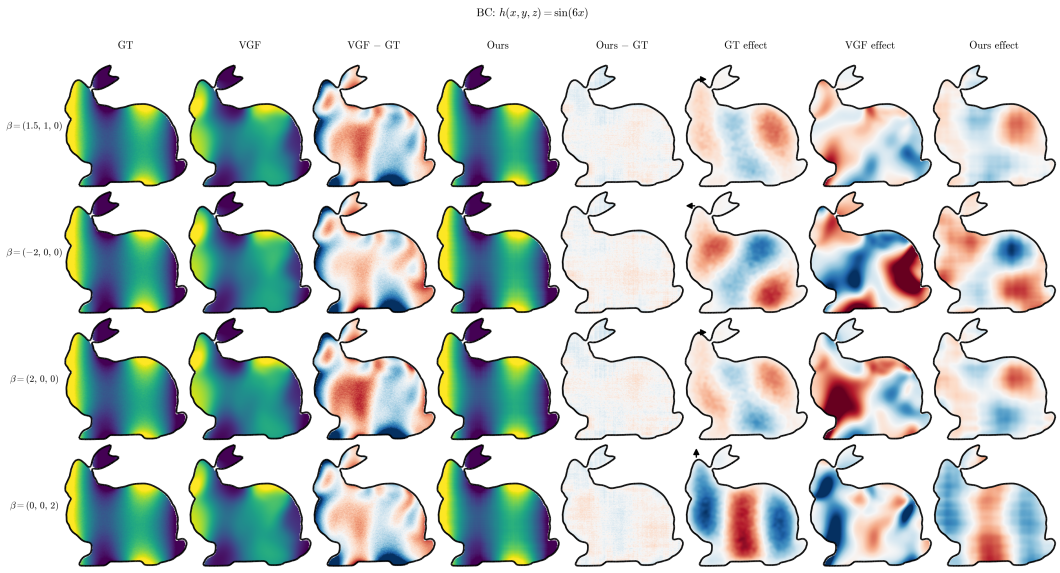


Figure 7: Bunny drift-diffusion qualitative, $h(x, y, z) = \sin(6x)$. Rows: four drift vectors $\beta = (1.5, 1, 0)$, $(-2, 0, 0)$, $(2, 0, 0)$, $(0, 0, 2)$. Columns 1–5: GT, GF style, GF style − GT, Ours, Ours − GT. Columns 6–8: drift-induced field u_β minus the mean over the four drifts, highlighting the dipole structure aligned with the in-plane component of each β (Gaussian-blurred for clarity; metrics in Table 10 use unblurred fields).

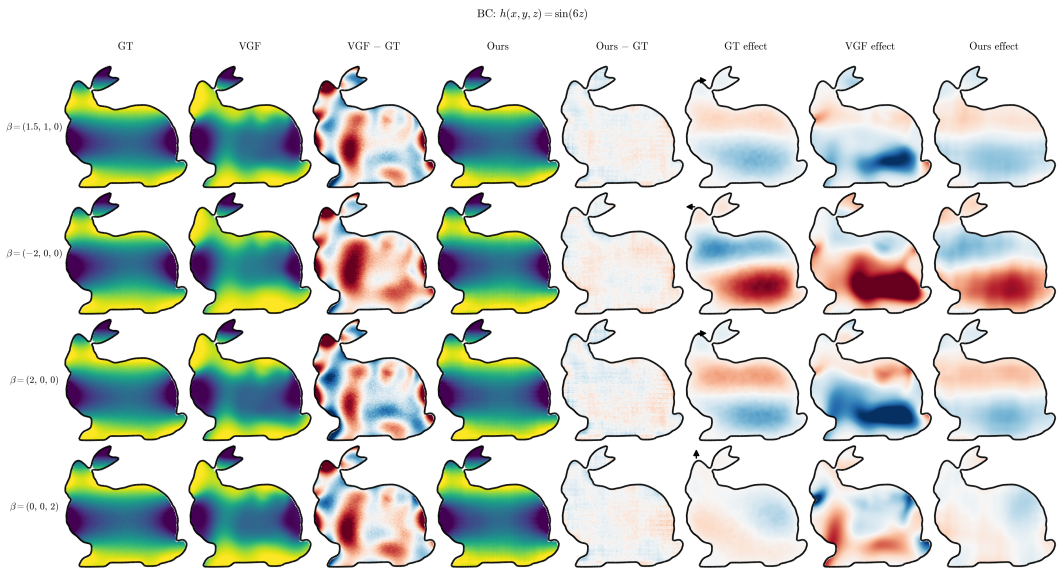


Figure 8: Bunny drift-diffusion qualitative, $h(x, y, z) = \sin(6z)$. Same layout as Figure 7.

G 2D MNIST benchmark: setup, NGF port, and additional qualitative

G.1 Setup details

Geometry. Each shape is an MNIST digit raster upsampled from 28×28 to a 256×256 binary mask, optionally retaining the thin inner holes that arise from the digit topology. The interior mask, ~512 boundary samples with normals, and interior anchors are produced by a deterministic shape generator. Domains for digits 0, 6, 8, 9 are multiply-connected.

Boundary-condition families. Two parametric families are sampled per problem with random coefficients,

$$\texttt{poly3:} \quad h(x,y) = a(x^3 - 3xy^2) + b(y^3 - 3x^2y) + c\,x^2$$
$$\texttt{exp_mix:} \quad h(x,y) = a\,e^{0.5x}\cos(0.5y) + b\,x\,y^2 + c\,y$$

In-distribution coefficients $a, b, c \sim U[-1, +1]$; OOD coefficients $a, b, c \sim U[+1, +2]$, strictly outside training. Two earlier high-frequency families `trig1`, `trig2` are kept for ablation only and excluded from headline numbers because every learned method failed catastrophically on them.

Sources. Poisson problems use one of four source families: `sin_cos`, `polynomial`, `gaussian`, `asymmetric`.

Splits. 991 train / 50 test (in-dist) / 50 OOD shapes; 7500 / 408 / 397 problem instances after filtering. The lift is trained on the 991 training shapes; the kernel is supervised on a separate Walk-on-Spheres corpus of $5{,}000$ MNIST digits (Appendix D.5).

Resolution. Numerical ground truth is a 5-point finite-difference Poisson solver at 256^2 followed by bilinear downsampling to 128^2, the resolution at which all neural models train and evaluate.

Eval metric. Un-normalized relative-L_2 error over interior pixels of each shape (mask $= 1$), aggregated across all problem instances per split. Trainer-side losses on y-normalized residuals are not used.

G.2 NGF 2D port

NGF's official code is written for tetrahedral meshes. Its network sees only positional encodings of the vertex coordinates, so its per-point features depend only on the geometry; three linear heads A (interior), C (all points), and D (boundary) produce the interior solution as $A(C^\top \mathrm{rhs}) - A(D^\top h)$ up to a diagonal scaling, where in the released MCB-B configuration a learned mass head forms rhs from the source. The boundary values are given, not predicted. Our 2D port keeps this architecture and the official optimization settings (feature width 128, learning rate 10^{-4}, gradient clipping 0.5, effective batch 8) and makes the adaptations a pixel grid requires: the encoder receives the interior mask (the pixel lattice is identical across shapes, so geometry must enter through the mask), the boundary is the band of exterior pixels adjacent to the domain, and multiply-connected boundaries are handled by that band without change.

An initial port differed from the official setup in several respects: it omitted the mass head; it used feature width 64, an MSE loss, and learning rate $5 \cdot 10^{-4}$; it regressed y-normalized targets (the NGF forward pass is linear in the data and has no bias path, so it cannot represent the offset this normalization introduces); it evaluated the boundary term on 256 randomly subsampled band pixels per step; and its encoder also received h and f. The aligned port follows the official setup in all of these respects, and Table 11 compares the two.

Table 11: NGF 2D port: initial port versus the port aligned with the official setup (relative L_2, %, mixed splits).

	test mean / median	test p95 / max	OOD mean / median	OOD p95 / max
initial	41.8 / 22.1	—	24.5 / 18.1	—
aligned (40 epochs)	3.87 / 2.00	16.9 / 40.2	4.20 / 3.85	6.0 / 10.1

G.3 Additional qualitative comparisons

Figures 9 and 10 extend Figure 4 with 24 additional OOD shapes (random pick from remaining Laplace and Poisson cases), same per-row layout and color-scale convention.



Figure 9: 2D MNIST OOD qualitative (additional, set 1 of 2). 12 shapes, random pick from remaining Laplace and Poisson cases; each GT tile is tagged with its problem type.



Figure 10: 2D MNIST OOD qualitative (additional, set 2 of 2). 12 shapes, random pick from remaining Laplace and Poisson cases; each GT tile is tagged with its problem type.

H Additional MCB-B qualitative comparisons

Figure 11 extends the qualitative comparison of §5.3 with 14 additional shapes, in the same per-shape five-panel layout (GT, NGF, |NGF − GT|, Ours, |Ours − GT|) and the same 3D cross-section rendering style.

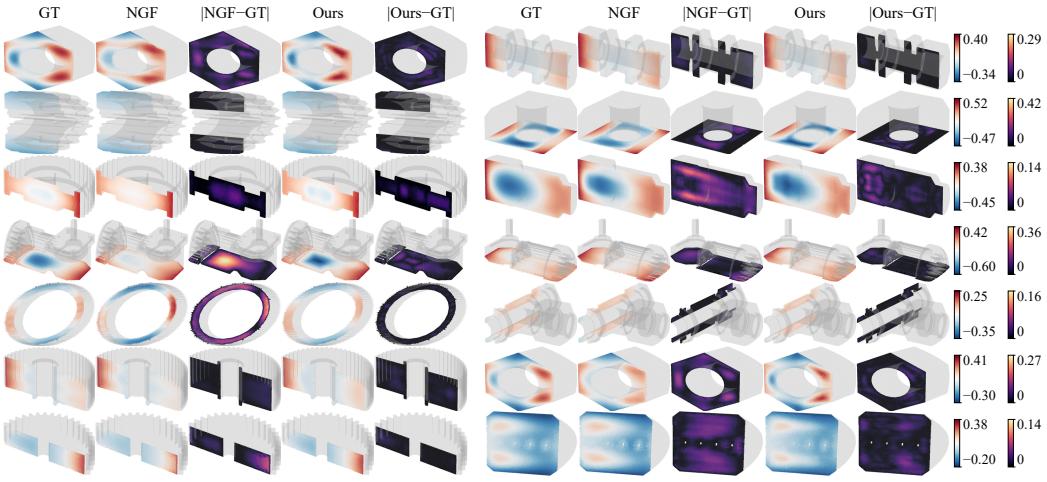


Figure 11: **Additional MCB-B Poisson qualitative comparisons.** 14 shapes (7 rows × 2 shapes per row) in the same layout and color-scale convention as Figure 5.

I 3D coefficient-OOD study

MCB-B's test problems use the same coefficient ranges as training. To test extrapolation in 3D without retraining, we drew 10 test shapes per category and posed 4 problems on each whose boundary data and sources come from parametric families with coefficients in $U[1, 2]$, outside the training ranges. References are computed with the FEM solver (lapy) used in the NGF repository. NGF is run from its released mass-prediction checkpoints; ours is the canonical pipeline of Table 2. A second, Laplace-only track uses the same shifted boundary data with $f \equiv 0$ and isolates boundary extrapolation. Table 12 reports the results; the in-distribution reference for each method is its Table 2 macro-average (0.241 for NGF, 0.193 for ours).

Table 12: 3D coefficient-OOD study: mean relative L_2 error over 40 problems per category (identical problems for both methods).

Track	Method	Nut	Gear	Motor	Fitting	Screws	Macro
Poisson-OOD	NGF (released)	0.678	0.605	0.616	0.627	0.548	0.615
	Ours	0.382	0.099	0.347	0.211	0.274	0.263
Laplace-OOD	NGF (released)	0.633	0.604	0.631	0.637	0.603	0.621
	Ours	0.127	0.037	0.162	0.070	0.101	0.099

J Inference speed: caching is implied by the factorization

Bit-identity of the cache. The kernel matrix $K_{\text{eff}} = [w_j\, K_\theta(p_i, \zeta_j; \Omega)]_{ij}$ is a deterministic function of Ω alone, so reusing it across (h, f) on the same shape is fp32-bit-identical to recomputing it per problem; we verified this on 50 random (p, h) pairs. Memory: an $(n_{\text{in}} \times n_{\text{surf}})$ fp32 tensor, $\approx$8 MB per shape at 128×128 with 200 surface samples.

Inference-only optimizations. The 3D timings in §5.4 use the following optimizations, which do not retrain or change any model. (i) *Query folding* (per-problem solve): without folding, the 3D lift treats each query point as a separate batch entry that cross-attends to its own copy of the same context, recomputing the context keys and values per query; since the cross-attention block processes query tokens independently, we fold all queries into the sequence dimension and compute the keys and values once. Predictions are identical in fp32, and the relative L_2 errors against the FEM references are unchanged. (ii) *Faster build*: the signed distance grid is rasterized on the GPU with exact point-triangle distances and a ray-parity inside test, which matches the CPU reference at all but isolated grid points, and K_{eff} is evaluated with the quadrature normalization folded in and in bf16. The build also redraws its random surface and interior samples, so its predictions are not bitwise identical to the original pipeline; the change in mean relative L_2 error (-0.004 on Nut, -0.002 on Motor) is within that of a control that only redraws the samples (-0.003 and $+0.000$).

Why the parametric baselines cannot cache. Transolver fuses (mask, h, f) tokens through slice-attention where every layer mixes geometry and boundary data; UPT concatenates (mask, h, f) as input channels to its image encoder; LNO and BENO likewise take the boundary data and the source as network inputs. None of these architectures separates a geometry-only state from (h, f), so they admit no per-shape cache. NGF is different: its per-point features depend only on the geometry, so a similar split into a per-shape state and a per-problem read-out is possible for it in principle; in our timing, we preloaded its mesh on the GPU and changed only the boundary data (§5.4). The cache is enabled by NHMO's factorization, not by an engineering choice.

Complexity. Per-shape precompute is $O(n_{\text{in}}\, n_{\text{surf}}\, d_{\text{kernel}})$. Per-problem cost is $O((n_{\text{in}} + n_{\text{surf}})\, d) + O(R^2\, c\, D)$ for the lift U-Net at resolution R, base channels c, depth D. This matches the complexity class of the parametric baselines' Galerkin-style forward.

Caveats. (i) When every problem uses a different shape (K=1), NHMO pays its geometry step (Table 5) for every problem; on a new mesh-based shape this step is cheaper than meshing, whereas on grid inputs the grid-based baselines need no such step. (ii) Training is a separate concern

(Appendix D.6). The speed advantage is at deployment, where the system is queried many times against the same geometries.

K Ablations: detail

This appendix expands the five takeaways summarized in §5.5. All numbers are un-normalized rel-L_2 over interior pixels at 128×128, mixed (Laplace + Poisson) test split unless noted otherwise.

K.1 Separating the lift's roles: source lift and residual head

Table 13 builds the 2D model up from the kernel alone. The source lift, which sees only (Ω, f), lowers the test error and degrades by only $1.04\times$ (mean) under the OOD shift, since h enters it only through the linear boundary integral. Adding the residual head, trained on top of the frozen source lift, gives the three-term model $u = \langle h, K_\theta \rangle + v_\varphi(\Omega, f) + r(\Omega, h, u_h)$ of §6. It matches or slightly surpasses the single h-conditioned lift of the main model at a larger total capacity, while keeping the source channel strictly independent of h. Training the lifts with five seeds (seed 0 is the reported checkpoint; $\pm$ is the standard deviation over seeds) gives a test mean of $1.87 \pm 0.05\%$ for the three-term model and $2.09 \pm 0.03\%$ for the single lift of the main model (OOD $2.48 \pm 0.05\%$ and $2.60 \pm 0.05\%$); the three-term model is lower on test for every seed. The source lift alone is nearly seed-independent (test mean 6.09–6.10%), as expected if its error is dominated by the kernel-fit residual it cannot see.

Table 13: From the kernel alone to the three-term variant (relative L_2, %, mean / median; Table 1 scale).

Variant	head inputs	test	test_ood
kernel only	—	7.7 / 5.4	6.8 / 5.6
+ source lift $v_\varphi(\Omega, f)$	$\mathbf{1}_\Omega$, SDF, f	6.1 / 4.5	6.3 / 5.2
+ residual head $r(\Omega, h, u_h)$	$\mathbf{1}_\Omega$, h, u_h	**1.84 / 1.74**	**2.56 / 2.39**
main model: single lift $v_\varphi(\Omega, h, f)$	$\mathbf{1}_\Omega$, h, f, u_h	2.1 / 2.0	2.6 / 2.5

What the correction learns. On Laplace pairs ($f \equiv 0$, 205 test pairs) the source contribution is zero, so the output of the single h-conditioned lift of the main model there is exactly its boundary correction. We checked three possibilities. It is not random: it correlates with the kernel-fit residual $e_h = u - u_h$ at median 0.97 and removes 71% of it. It is not a fluctuation around the residual: 82% of its spectral energy lies in the lowest tenth of radial frequencies, against 1% for a matched white-noise control (medians), and it reproduces the residual rather than scattering around it. It is not a fixed bias: the residual it tracks is linear in h, changes sign and shape with the boundary data, and averages to about zero over the symmetric coefficient draw of the test split. Its magnitude is small (median 4.8% of $\|u_h\|$), and removing the boundary inputs forfeits the correction (the source lift alone reaches 6.1% test mean against 2.1%, Table 13). The residual head r of the three-term model behaves the same way on these pairs (median correlation 0.97, 75% of the residual removed, 81% low-frequency energy).

Perturbations of the boundary data. The kernel channel is linear in h: a perturbation $h \to h + \epsilon\eta$ changes u_h by exactly $\epsilon\langle \eta, K_\theta \rangle$, which is bounded by $\epsilon \max|\eta|$ for the quadrature-normalized kernel. We measured the amplification, the relative L_2 response of u_h divided by $\epsilon \max|\eta|$, on 20 shapes for $\epsilon \in [0.01, 0.5]$: its mean over shapes is 0.84 for smooth η and 0.17 for white-noise η, constant over this range of ϵ, and the full model including the learned lift stays at or below 1.02 on average.

K.2 Lift removal (kernel-only vs. kernel + lift)

Defends the factorization $u = \langle h, K_\theta \rangle + v_\varphi$. The kernel alone already beats every nonlinear end-to-end baseline on OOD; the learned lift is a small correction.

Table 14: Lift removal. Mixed (Laplace + Poisson) rel-L_2 over interior pixels (median / mean).

Variant	test (in-dist)	test_ood	OOD/test
K-only (no v_φ)	5.4% / 7.7%	5.6% / 6.8%	$\mathbf{1.0}\times$
$K + 6.4$**M lift (canonical)**	$\mathbf{2.0}\%$ / $\mathbf{2.1}\%$	$\mathbf{2.5}\%$ / $\mathbf{2.6}\%$	$1.25\times$
$K + 11.3$M lift (capacity scan, A2)	2.0% / 2.1%	2.3% / 2.5%	$1.15\times$

K.3 Lift capacity (6.4M vs 11.3M parameters)

Doubling lift parameters from 6.4M to 11.3M gives no in-distribution improvement and only marginal OOD gain (Table 14, last row). NHMO is not capacity-limited at the lift; the factorization, not network size, is the structural reason for the result.

K.4 KDE bandwidth σ for Walk-on-Spheres supervision

The kernel is robust to KDE σ over a $5\times$ range (0.1%–0.5% of domain width).

Table 15: KDE bandwidth scan. K-only test median.

σ (fraction of domain width)	K-only test median
0.1% (sharper)	$\sim 6.5\%$
0.2% (canonical)	$\mathbf{5.4}\%$
0.5% (smoother)	$\sim 6.0\%$

K.5 Training-shape count and single mixed-corpus generalization

The NHMO kernel is supervised on a fixed $5{,}000$-shape MNIST corpus drawn from *all* 10 digit classes (0–9), spanning both simply-connected (e.g., 1, 7) and multiply-connected (e.g., 0, 6, 8, 9) topologies, with no class labels. The harmonic-measure factorization makes the kernel a per-shape function of geometry, so corpus diversity adds signal rather than competing for capacity. By contrast, NGF's published MCB-B numbers come from five separate models, one per shape category. NHMO trains one kernel and one lift across all 10 MNIST digit classes simultaneously.

Table 16: Kernel-supervision corpus size. K-only test median rel-L_2 as the corpus grows.

Kernel supervision shapes	K-only test median	K+lift test median
200	$\sim 7.0\%$	—
500	$\sim 6.0\%$	$\sim 3.5\%$
1,000	$\sim 5.7\%$	$\sim 2.5\%$
5,000 (canonical)	$\mathbf{5.4}\%$	$\mathbf{2.0}\%$

K.6 Walk-on-Spheres sample count

The kernel is robust to the WoS sample count above $1{,}000$ walks per query; the canonical run uses $10{,}000$ walks per query and matches the test median of the kernel ablation in Table 14. Below $1{,}000$ walks the KDE supervision becomes too noisy and the kernel degrades.

K.7 Boundary quadrature resolution at inference (n_{surf} scan)

NHMO's boundary-integral $\sum_\zeta K_\theta(p, \zeta)\, h(\zeta)$ is the discretization of a continuous integral. We verify this at inference time with the same trained kernel, varying only the number of boundary samples n_{surf}. Above ~ 100 samples the prediction is converged; below that, the result degrades *gracefully* rather than catastrophically, so the model is robust to the boundary-quadrature resolution at inference over the tested range. Parametric baselines have no analogous discretization knob; their inference quality is tied to whatever resolution the encoder was trained at.

Table 17: Boundary-discretization scan at inference. Mixed rel-L_2 on test_ood, 25 shapes $\times$ $\sim$8 problems.

n_{surf}	median	mean	p95
50	3.12%	3.58%	5.62%
100	2.48%	2.65%	4.05%
200 (canonical)	**2.44%**	**2.58%**	**4.08%**
400	2.43%	2.56%	3.98%

K.8 Per-MNIST-class breakdown (single mixed-corpus uniformity)

A direct counter to per-category-corpora training. A single mixed-corpus NHMO model (kernel supervised on the 5,000-shape corpus, digits 0–9) produces uniform performance across all classes; the spread across classes is much smaller than the OOD gap to any nonlinear end-to-end baseline.

Table 18: Per-MNIST-class rel-L_2 mean (mixed Laplace + Poisson, full-interior un-normalized).

Digit class	n_{test}	test mean	test_ood mean
0	50	2.14%	2.89%
1	30	2.19%	2.95%
2	40	1.86%	2.60%
3	36	1.94%	2.26%
4	38	2.15%	2.48%
5	46	2.02%	2.48%
6	35	2.28%	2.88%
7	55	1.85%	1.95%
8	39	2.38%	**3.81%**
9	39	2.32%	2.74%
spread (max $-$ min)		0.53%	1.87%

The 10 digit classes have very different geometries (1 is a narrow stroke, 0/6/8/9 have interior loops, 8 has two), yet a single mixed-corpus model attains rel-L_2 within 0.53% absolute spread on in-distribution test and 1.87% on OOD. Digit 8, the most challenging case (multiply-connected with two interior loops), is the worst class on OOD at 3.81% but still beats every nonlinear end-to-end baseline's overall mean.

K.9 Representation invariance: SDF vs. point-cloud encoder

A direct attack on the "your kernel just memorizes the boundary point cloud" critique. We retrain the kernel from scratch with the same hyperparameters as the canonical model ($d = 256$, 5,000-shape corpus, KDE $\sigma = 0.2\%$, 60,000 steps), changing *only* the shape encoder family from a point-cloud encoder over $\partial\Omega$ to a 2D SDF-CNN over a 64×64 SDF grid. The resulting kernel is paired with the canonical lift v_φ (no lift retraining).

Table 19: Representation-invariance ablation: identical hyperparameters, swap shape encoder.

Shape encoder	test (in-dist)	test_ood	OOD/test
Point-cloud (canonical)	**2.0% / 2.1%**	**2.5% / 2.6%**	1.25$\times$
2D SDF-CNN (64×64)	2.28% / 2.43%	2.59% / 2.97%	**1.14$\times$**

The encoder swap costs only 0.27% absolute median on in-distribution test (1.14$\times$ canonical) and 0.09% on OOD (1.04$\times$ canonical). Notably, the SDF-encoder kernel has a tighter OOD/test gap (1.14$\times$ vs 1.25$\times$), suggesting the SDF representation may even improve coefficient-distribution generalization. Whatever representation makes $K_\theta(\cdot, \cdot; \Omega)$ an honest harmonic-measure operator

suffices. NGF's released pipeline takes tetrahedral-mesh vertices with explicit boundary indices as input, so the same swap does not apply to it directly.

K.10 Learned volumetric integrand

To test the Green's-function alternative of §6 directly, we replaced the source lift with a learned integrand $G_\theta(p, q; \Omega)$ whose volume integral against f gives the source term, keeping the kernel term unchanged. The integrand receives a $\log|p-q|$ singularity feature and the frozen shape latent of our kernel. It reaches 6.7 / 5.2 (test mean / median), no better than the source lift (Table 13), while its volume quadrature takes $O(N_p N_q)$ network evaluations per problem; the boundary term, by contrast, is a single quadrature over a few hundred surface samples. Its boundary derivative $-\partial_\nu G_\theta$, computed on the 205 Laplace test pairs by automatic differentiation, integrates to $\sim 10^{-3}$ over the boundary (the Poisson kernel has mass 1) and is negative on roughly half of it, so it satisfies neither defining property of the Poisson kernel (nonnegativity and unit mass).

K.11 Origin of the kernel-fit residual

Why does the 2D kernel leave a residual that the lift must correct? The KDE targets on which the 2D kernel is trained, and its boundary nodes, are built on simplified polyline contours of the digits, which do not coincide with the rasterized masks on which the reference solutions are computed. Two measurements locate the resulting error in this choice of geometric representation rather than in the sampler. First, integrating the targets themselves as if they were the kernel reproduces the reference solutions only to 5.1%, unchanged with $100\times$ more walks, so the error is systematic rather than statistical. Second, walks run directly on the mask geometry match the reference solutions to 0.07%, so the WoS estimator, its ε-shell, and the step cap are not responsible. The kernel reaches this ceiling, so the residual e_h is the systematic error of its training signal rather than something the kernel fails to fit.

Since $u - u_h = u_f + e_h$, the solution-level supervision of the lift, and of the residual head r (Appendix K.1), contains this error, which is why the learned correction removes most of it (71% for the single lift on the Laplace test pairs). Placing the KDE's boundary nodes on the contour of the rasterized masks instead, with the construction otherwise unchanged, lowers the kernel-only error from 5.9% to 3.2% in a controlled 2D experiment (mean over the Laplace test pairs), and a kernel-plus-lift model (without r) trained on this kernel reaches 1.8% / 1.7% (mean / median) on test and 2.1% / 2.2% on OOD, against 2.1% / 2.0% and 2.6% / 2.5% for the model of Table 1. We leave this choice of geometric representation for the training signal, together with exploiting the linearity of e_h in the design of r, to future work.

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: The claims in the abstract and Section 1 are supported by Sections 4 and 5.

   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 discuss our limitations in the **Limitations** paragraph of the Conclusion.

   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: [N/A]

Justification: Section 3 recaps classical potential theory with citations; we make no new theoretical claim.

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 disclose all information needed for reproduction in Section 5 and the appendix.

   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: We release the full code, all generated datasets with their generators, and our trained checkpoints at `[URL to be added]`, with instructions that map each table to a command. MCB-B is publicly released by the NGF authors.

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 provide experimental details in Section 5 Setup and the appendix.

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: We report 95th percentiles of the error (per pair in Section 5.2, per shape in Table 3) and training-seed variability of the 2D lifts (Appendix K.1).

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: We provide compute information in Section 5 and the appendix.

   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: We have read the Code of Ethics; the work is a PDE surrogate with no human-subject or sensitive-data concerns.

   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: [N/A]

    Justification: Our work contributes a numerical surrogate for elliptic PDE solving. It does not impact society at large.

    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: We release no high-risk generative models or scraped datasets.

    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: We cite external datasets, baselines, and libraries with their original sources.

    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 released code, datasets, and checkpoints come with documentation (installation, data generation, training and evaluation commands) and a model card that describes each checkpoint.

    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: No crowdsourcing or human-subject research.

    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: No human-subject research.

    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: [N/A]

    Justification: LLMs were used only for writing assistance, not in the core methodology.

    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.