LATENT TWIN OPERATOR

Anonymous authors
Paper under double-blind review

ABSTRACT

Surrogate models deployed on real physical systems rarely see data at fixed resolutions: sensor configurations vary across deployments and may evolve over time as sensing infrastructure changes. We introduce the Latent Twin Operator (LTO), a latent-space surrogate for time-evolving PDEs with a Convolutional Conditional Neural Process (ConvCNP)-style encoder and decoder that accepts a context set of N sensor observations with arbitrary placement and can be queried at any resolution. A learned latent evolution map advances the encoded state directly between arbitrary time points, decoupling temporal evolution from the observation and query discretizations. We derive an explicit $\mathcal{O}(N^{-2/(3D)})$ rate for the context discretization error in spatial dimension D under quasi-uniform refinement. We verify the predicted decay empirically on a 2D heat-equation benchmark. Across time-dependent PDE benchmarks, LTO achieves strong accuracy under one-step comparisons and transfers from native to coarser spatial resolutions with fixed parameters. On Navier–Stokes, its direct latent evolution further reduces error over longer prediction horizons relative to recursive evaluation.

1 INTRODUCTION

Realistic PDE workflows rarely operate at a single fixed resolution. In simulation, adaptive meshes refine where the dynamics demand it; in experiments and deployed systems, sensor locations and densities vary across settings and may change over time. Such settings call for learned surrogates that are not tied to the discretization on which they were trained, but can accept sparse, dense, or irregular observations and return predictions at whatever resolution is required. We refer to this ability to operate consistently as the input and output discretizations change as *discretization invariance*.

To achieve this form of discretization invariance, we introduce the *Latent Twin Operator* (LTO), an encode→evolve→decode surrogate in which observations are mapped to a latent state, advanced in time by a learned evolution map, and decoded at arbitrary query points. Its encoding and decoding employ parameter-free kernel-smoothing operations in the style of Convolutional Conditional Neural Processes (Gordon et al., 2020), coupled to convolutional latent representations and dynamics (Definition 2.1).

Two standard operator-learning architectures impose stronger discretization requirements than the LTO. The Fourier Neural Operator (FNO; Li et al., 2020) is formulated on regular grids, since its core operation uses the FFT; nonuniform Fourier transforms or interpolation can relax this requirement, but require additional machinery outside the standard formulation. DeepONet (Lu et al., 2019) allows arbitrary output query locations through its trunk network, but its branch network is tied to a fixed set of input sensor locations. Therefore, in their standard forms, neither architecture directly accepts a variable-size, irregularly placed context set without resampling or modification. More generally, evaluating learned PDE surrogates on discretizations different from those used during training can introduce *discretization mismatch error* (Gao et al., 2025).

The LTO builds directly on the Latent Twin framework (Chung et al., 2026; 2027), which formalized the encode→evolve→decode Latent Twin formulation and established an approximation theory for latent-space surrogate models. LTO advances this framework through resolution-flexible encoding and decoding and derives a new explicit discretization-error bound, providing a theoretical guarantee of discretization invariance.

Among related latent operator methods, the closest in spirit is the Latent Neural Operator (LNO; Wang & Wang, 2024). Like LTO, LNO maps a variable-size set of observations to a fixed-size

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

latent representation and can decode at arbitrary query points. LNO achieves this flexibility through learned cross-attention over latent tokens, whereas LTO uses a spatially structured kernel-smoothing representation together with convolutional latent dynamics. The LTO construction therefore retains flexibility in the input and output discretizations while making the associated discretization error directly amenable to analysis. Other latent-space approaches, including the Latent Neural PDE Solver (Li et al., 2025) and the Physics-Informed Latent Neural Operator (Karumuri et al., 2026), share elements of the encode→evolve→decode paradigm but differ in how the latent representation, evolution, and input–output maps are realized. We discuss these connections in more detail in Section 3 and Appendix A.2.

A key advantage of the LTO's kernel-smoothing construction is its connection to the well-established theory of nonparametric regression and scattered-data approximation (Wasserman, 2006; Wendland, 2005). We use these tools to quantify how the approximation error depends on the number and placement of context points and derive an explicit convergence rate. This resolves the previously abstract spatial-discretization component of the general Latent Twin approximation theory (Chung et al., 2026) and makes its dependence on context resolution explicit.

Our main contributions are:

- We introduce the Latent Twin Operator, a resolution-flexible encode→evolve→decode surrogate that accepts context sets of varying size and placement and produces predictions at arbitrary query resolutions. For its SetConv-based realization, we prove encoder stability with a Lipschitz constant independent of context-set size and placement.
- We derive an explicit spatial discretization-error bound using scattered-data approximation theory, $\epsilon_{disc}(N) = O(N^{-2/(3D)})$, for N context points in spatial dimension D. Combined with the general Latent Twin approximation theory, this yields a discretization-invariance guarantee under quasi-uniform refinement.
- We verify the predicted discretization decay on a 2D heat-equation benchmark and evaluate the learned LTO under matched protocols on Burgers, Navier–Stokes, and Darcy flow. On the time-dependent benchmarks, LTO transfers from the native training resolution to coarser spatial resolutions with fixed model parameters and achieves lower matched one-step errors than FNO on Burgers and than FNO and LNO on Navier–Stokes. On Navier–Stokes, direct latent evolution further reduces long-horizon error relative to recursive evaluation.

Section 2 develops the Latent Twin Operator and its discretization-invariance theory. Section 3 places the LTO in the broader operator-learning landscape. Section 4 evaluates its discretization behavior across four PDE benchmarks – the 2D heat equation, Burgers, 2D Navier–Stokes, and Darcy flow – under a common train-once, evaluate-across-resolutions protocol. Section 5 examines the scope and limitations of the regularity assumptions underlying the theory. Additional technical details, proofs, and extended comparisons with related methods are provided in the Appendix A.

2 The Latent Twin Operator

We consider time-evolving states $u : \mathcal{T} \to \mathcal{U}$ defined on a bounded spatial domain $\Omega \subset \mathbb{R}^D$, where $\mathcal{T} = [0, T]$ and $\mathcal{U}$ is a normed vector space of functions from Ω to $\mathbb{R}^p$. At any time t, the state $u(t)$ is observed N at context locations $c_{\text{ctx}} = \{c_1, \ldots, c_N\} \subset \Omega$. For any state $u_t = u(t)$, we denote the corresponding observed context values by $u_t(c_{\text{ctx}}) := \{u_t(c_j)\}_{j=1}^N$. The state may be evaluated at an arbitrary set of M query locations $c_{\text{q}} = \{q_1, \ldots, q_M\} \subset \Omega$, with corresponding values $u_t(c_{\text{q}}) := \{u_t(q_j)\}_{j=1}^M$. Different observations of the same underlying state may use context and query sets of different size and placement, precisely the multi-resolution setting motivated in Section 1.

Definition 2.1 (Latent Twin Operator)**.** Let $\Phi(\cdot\,; s, u_s) : \mathcal{T} \to \mathcal{U}$ denote the solution of $\dot{u} = \mathcal{D}(u)$ with $u(s) = u_s$. The *Latent Twin Operator* (LTO) is the surrogate

$$\Psi^{s, c_{\text{ctx}}, u_s}(t, c_{\text{q}}) := d(m_{s\to t}(e(u_s(c_{\text{ctx}}), c_{\text{ctx}})), c_{\text{q}}). \tag{1}$$

Here, e maps the observed context data to a latent state, $m_{s\to t} : \mathcal{Z} \to \mathcal{Z}$ evolves this state from time s to time t, and d maps the evolved latent state to values at the query locations c_{q}.

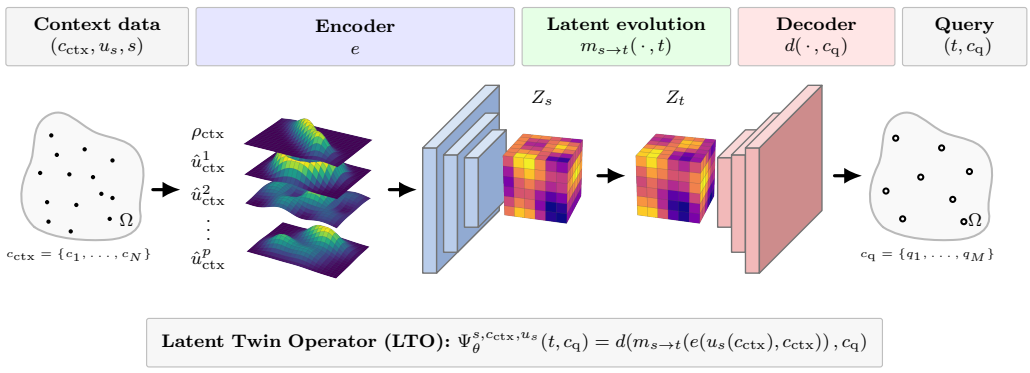

Figure 1: Latent Twin Operator. Irregular context observations are mapped by SetConv to a fixed-grid representation, encoded into latent space, evolved from Z_s to Z_t by $m_{s\to t}$, and decoded at arbitrary query locations.

The encoder e, latent evolution map $m_{s\to t}$, and decoder d are trainable; we suppress their learnable parameter dependence throughout for notational simplicity, writing it explicitly as Ψ_θ below where the discretization bound requires it. We next specify the encoder e, latent evolution map $m_{s\to t}$, and decoder d. The encoder and decoder are chosen to accommodate arbitrary context and query discretizations, while $m_{s\to t}$ evolves the resulting representation entirely in latent space.

Resolution-independent encoding. To accommodate context sets of arbitrary size and placement, we construct the encoder using the Convolutional Conditional Neural Process (ConvCNP) framework (Gordon et al., 2020), extending Conditional Neural Processes (Garnelo et al., 2018). Given the context data $\big(c_{\text{ctx}}, u_s(c_{\text{ctx}})\big)$, a set convolution (SetConv) maps this unordered, variable-size set to a field on a fixed spatial grid. Each observation contributes to nearby locations through a distance-based kernel, making the construction invariant to the ordering of the context points while retaining their spatial locations.

We use the Gaussian kernel $\kappa_\ell(x, y) = \exp\Big(-\frac{d_\Omega(x,y)^2}{2\ell^2}\Big)$, where d_Ω denotes the distance on Ω and $\ell > 0$ determines the kernel length scale. For a generic spatial location $x \in \Omega$, define

$$\hat{u}_{c_{\text{ctx}}}(x; u_s) := \frac{\nu_{c_{\text{ctx}}}(x; u_s)}{\rho_{c_{\text{ctx}}}(x)}, \quad \text{where} \tag{2}$$

$$\rho_{c_{\text{ctx}}}(x) := \sum_{j=1}^{N} \kappa_\ell(x, c_j) \qquad \text{and} \qquad \nu_{c_{\text{ctx}}}(x; u_s) := \sum_{j=1}^{N} u_s(c_j)\kappa_\ell(x, c_j). \tag{3}$$

The quantity $\hat{u}_{c_{\text{ctx}}}$ is the Nadaraya–Watson kernel estimate of u_s (Nadaraya, 1964; Watson, 1964); $\rho_{c_{\text{ctx}}}$ measures the local concentration of context points, while $\nu_{c_{\text{ctx}}}$ is the corresponding kernel-weighted sum of state values.

We evaluate $\rho_{c_{\text{ctx}}}$ and $\hat{u}_{c_{\text{ctx}}}$ on a fixed spatial grid and stack them into a $(1 + p)$-channel field. The encoder e maps this gridded SetConv representation to

$$z_s = e(\rho_{c_{\text{ctx}}}, \hat{u}_{c_{\text{ctx}}}(\,\cdot\,; u_s)) \in \mathcal{Z} := \mathbb{R}^{H\times W\times k}.$$

Latent evolution. The latent evolution map $m_{s\to t} : \mathcal{Z} \to \mathcal{Z}$ advances the encoded state directly from time s to time t. Since $m_{s\to t}$ acts entirely on the fixed latent representation, its input and output are independent of the context and query discretizations.

Resolution-independent decoding. Given the evolved latent state z_t, the decoder d maps the latent representation back to the physical state and evaluates it at the query locations c_{q}. Its final evaluation uses kernel-weighted interpolation on the spatial grid, so the number and placement of query points may vary independently of both the context locations and the internal grid.

Connection to the Latent Twin (Chung et al., 2026) approximation theory. The LTO extends the general Latent Twin framework (Chung et al., 2026) to settings with variable context and query discretizations. While the original framework develops its approximation theory under a fixed discretization and treats the encoder, latent evolution map, and decoder abstractly, here we construct these components in a resolution-flexible form and use that theory as the basis for an explicit discretization analysis.

For each N, first fix a context discretization $c_{\mathrm{ctx}}^{(N)}$ and define $P_N : \mathcal{U} \to \mathbb{R}^{Np}$ with $P_N u = u(c_{\mathrm{ctx}}^{(N)})$ as the corresponding sampling operator. Following Chung et al. (2026, Thm. 2), the query set is taken equal to the context set, $c_{\mathrm{q}} = c_{\mathrm{ctx}}^{(N)}$, throughout this section. Let $R_N : \mathbb{R}^{Np} \to \mathcal{U}$ be a lifting operator such that the pair (P_N, R_N) is consistent and stable in the sense required by Chung et al. (2026, Thm. 2). The general Latent Twin theory applies to any such consistent and stable pair (P_N, R_N) for which

$$\epsilon_{\mathrm{disc}}(N) := \sup_{u \in \mathcal{K}} \|R_N(P_N u) - u\|_{\mathcal{U}} \longrightarrow 0,$$

where $\mathcal{K} \subset \mathcal{U}$ is a compact set of admissible states.

We denote by

$$\widetilde{\Psi}_{\theta,N}(t; s, u_s) := R_N\Big(\Psi_\theta^{s, c_{\mathrm{ctx}}^{(N)}, u_s}\big(t, c_{\mathrm{ctx}}^{(N)}\big)\Big) \tag{4}$$

the corresponding lifted LTO, where Ψ_θ is as in Definition. 2.1 with its parameter dependence made explicit. Under the additional assumptions of Chung et al. (2026, Thm. 2), the corresponding lifted Latent Twin satisfies

$$\sup_{(s,t,u_s)\in[0,T]^2\times\mathcal{K}} \Big\|\widetilde{\Psi}_{\theta,N}(t; s, u_s) - \Phi(t; s, u_s)\Big\|_{\mathcal{U}} \le C\left(\epsilon_{\mathrm{disc}}(N) + \big(1 + \mathrm{e}^{L_{\mathcal{K}} T}\big)\epsilon_{\mathrm{ae}}^{(N)} + L_{d,N}\,\epsilon_{\mathrm{map}}^{(N)}\right). \tag{5}$$

Here $\epsilon_{\mathrm{ae}}^{(N)}$ is the autoencoder approximation error and $\epsilon_{\mathrm{map}}^{(N)}$ is the latent-evolution error. The quantity $L_{d,N}$ is the decoder stability constant from Chung et al. (2026), $L_{\mathcal{K}}$ controls the Lipschitz growth of the underlying dynamics on $\mathcal{K}$, and $C > 0$ is a constant determined by the assumptions of the theorem. For each N, the bound applies to any fixed admissible pair (P_N, R_N) satisfying the assumptions of Chung et al. (2026, Thm. 2), regardless of the particular discretization and reconstruction operators used.

For the LTO, we take P_N to be sampling at the context locations and instantiate R_N by the normalized SetConv kernel reconstruction,

$$R_N(P_N u)(x) := \frac{\sum_{j=1}^N u(c_j)\kappa_\ell(x, c_j)}{\sum_{j=1}^N \kappa_\ell(x, c_j)}.$$

This is precisely the normalized kernel estimate equation 2 used in the SetConv representation. Thus, the abstract discretization term $\epsilon_{\mathrm{disc}}(N)$ in equation 5 becomes a concrete scattered-data approximation error. We first establish stability of this reconstruction and then derive an explicit bound on $\epsilon_{\mathrm{disc}}(N)$ in terms of the number and placement of the context points.

Encoder stability. Before deriving the discretization rate, we first establish that the normalized SetConv reconstruction is stable with respect to perturbations of the observed state. For a fixed context set c_{ctx}, the Nadaraya–Watson reconstruction satisfies a Lipschitz bound whose constant does not depend on the number or placement of the context points. Hence the stability of the SetConv preprocessing is uniform across admissible context discretizations. The precise statement and proof are given in Lemma A.2 and Corollary A.3 in Appendix A.1.

Explicit discretization error. For the LTO, the sampling operator P_N records the values of u at the context locations, while the reconstruction operator R_N is the normalized SetConv kernel reconstruction. Hence $R_N(P_N u) = \hat{u}_{c_{\mathrm{ctx}}}(\,\cdot\,; u)$, and therefore

$$\epsilon_{\mathrm{disc}}(N) = \sup_{u \in \mathcal{K}} \|\hat{u}_{c_{\mathrm{ctx}}}(\,\cdot\,; u) - u\|_{\mathcal{U}}\,.$$

Thus $\epsilon_{\mathrm{disc}}(N)$ is the error of kernel smoothing on scattered data. Its value depends on both the number of context points and their placement. To obtain an explicit rate, we impose the following regularity condition on the admissible states and context sets.

Assumption 2.2 (Regularity)**.** Assume that (a) $\mathcal{K} \subset C^2(\Omega;\mathbb{R}^p)$ satisfies $\sup_{u\in\mathcal{K}}\sup_{x\in\Omega}\|\mathrm{D}^2u(x)\| \le M$, (b) Ω is periodic, and (c) that each context set $c_{\mathrm{ctx}} = \{c_1,\dots,c_N\}$ admits a partition $V_1,\dots,V_N$ of Ω such that

$$c_j \in V_j, \qquad \operatorname{diam}(V_j) \le 2\delta(c_{\mathrm{ctx}}), \qquad |V_j| = \frac{|\Omega|}{N}$$

where $\delta(c_{\mathrm{ctx}}) := \sup_{x\in\Omega}\min_{1\le j\le N} d_\Omega(x,c_j)$ is the fill distance.

Condition (a) controls the local curvature of the admissible states, while condition (c) prevents the context points from becoming arbitrarily clustered. The equal-volume partition is slightly stronger than standard quasi-uniformity assumptions in scattered-data approximation (Wendland, 2005), but is used directly in the quadrature argument of Proposition A.4.

Under Assumption 2.2, we decompose the SetConv reconstruction error into a kernel-smoothing component and a finite-context component by introducing the continuous analogue

$$\bar{u}_\ell(x) := \frac{\int_\Omega u(y)\kappa_\ell(x,y)\,\mathrm{d}y}{\int_\Omega \kappa_\ell(x,y)\,\mathrm{d}y}.$$

Under the periodicity assumption, the denominator is independent of x (see Lemma A.1); we denote it by $\bar{\kappa} := \int_\Omega \kappa_\ell(x,y)\,\mathrm{d}y$. The quantity $\bar{u}_\ell$ is obtained by replacing the finite kernel-weighted sums in $\hat{u}_{c_{\mathrm{ctx}}}$ by their corresponding integrals. It therefore provides the decomposition

$$\|\hat{u}_{c_{\mathrm{ctx}}} - u\|_{\mathcal{U}} \le \|\hat{u}_{c_{\mathrm{ctx}}} - \bar{u}_\ell\|_{\mathcal{U}} + \|\bar{u}_\ell - u\|_{\mathcal{U}},$$

where the second term is the kernel-smoothing bias and the first is the finite-context quadrature error.

The smoothing error follows from a second-order Taylor expansion. Because the Gaussian kernel is symmetric about x, the first-order term vanishes after integration, leaving a contribution controlled by the Hessian of u. Under Assumption 2.2, this gives

$$\sup_{u\in\mathcal{K}}\sup_{x\in\Omega}\|\bar{u}_\ell(x) - u(x)\| \le \tfrac{1}{2}DM\ell^2. \tag{6}$$

The proof is given in Appendix A.1; see also Wasserman (2006, Ch. 5) for the corresponding classical kernel-smoothing argument.

It remains to control the finite-context error $\|\hat{u}_{c_{\mathrm{ctx}}} - \bar{u}_\ell\|_{\mathcal{U}}$. Under Assumption 2.2, a scattered-data quadrature argument (Proposition A.4, Appendix A.1) shows that this error is of order $\delta(c_{\mathrm{ctx}})/\ell$. Together with the $O(\ell^2)$ smoothing bias from equation 6, this yields the trade-off quantified in the following theorem.

Theorem 2.3 (Explicit discretization error for the SetConv instantiation)**.** *Under Assumption 2.2, for any bandwidth $\ell > 0$,*

$$\epsilon_{\mathrm{disc}}(N) \le \tfrac{1}{2}DM\,\ell^2 + C_{\mathcal{K}}\frac{\delta(c_{\mathrm{ctx}})}{\ell},$$

where $C_{\mathcal{K}} > 0$ depends only on D, Ω, M, and $\sup_{u\in\mathcal{K}}\|u\|_\infty$.

Corollary 2.4 (Optimized discretization rate)**.** *Choosing $\ell^\star = \left(\frac{C_{\mathcal{K}}\,\delta(c_{\mathrm{ctx}})}{DM}\right)^{1/3}$ in Theorem 2.3 gives*

$$\epsilon_{\mathrm{disc}}(N) \le C\,\delta(c_{\mathrm{ctx}})^{2/3},$$

for a constant $C > 0$ independent of the context resolution.

Corollary 2.5 (Discretization invariance)**.** *Assume that the context sets satisfy $\delta(c_{\mathrm{ctx}}) = \Theta(N^{-1/D})$. With the bandwidth choice $\ell = \ell^\star$ from Corollary 2.4, substituting the resulting discretization rate into the Latent Twin Operator error bound equation 5 gives*

$$\sup_{(s,t,u_s)\in[0,T]^2\times\mathcal{K}}\left\|\widetilde{\Psi}_{\theta,N}(t;s,u_s) - \Phi(t;s,u_s)\right\|_{\mathcal{U}} \le O\Big(N^{-2/(3D)}\Big) + C\big(1+\mathrm{e}^{L_{\mathcal{K}}T}\big)\epsilon_{\mathrm{ae}}^{(N)} + CL_{d,N}\epsilon_{\mathrm{map}}^{(N)}.$$

For i.i.d. uniformly sampled context points, the fill-distance estimate of Penrose (2003) yields the corresponding high-probability rate

$$\epsilon_{\mathrm{disc}}(N) = O\left(\left(\frac{\log N}{N}\right)^{2/(3D)}\right).$$

3 Related Work

The Latent Twin Operator sits at the intersection of four lines of work: neural operators, implicit neural representations, conditional neural processes, and the latent-space encode→evolve→decode architectures it most closely resembles. We summarize each briefly here, with an extended discussion in Appendix A.2.

Operator learning formalizes learning a PDE's solution operator directly (Kovachki et al., 2023; Boullé & Townsend, 2024); DeepONet (Lu et al., 2019) and the Fourier Neural Operator (FNO; Li et al., 2020) are its two standard instantiations, the latter requiring a regular grid via the FFT. Our encoder has no such grid dependency: it builds instead on Conditional Neural Processes and their convolutional variant, ConvCNP (Garnelo et al., 2018; Gordon et al., 2020), whose kernel-weighted SetConv embedding underlies our encoder and decoder (Section 2).

Closest to our method is the Latent Neural Operator (LNO; Wang & Wang, 2024): its Physics-Cross-Attention likewise maps an arbitrary input set to a fixed-size latent state with input and output locations decoupled, but via learned cross-attention over an unstructured token set, rather than the parameter-free, spatially-structured kernel smoothing we use. A wider family of attention-based architectures tackles the same irregular-input problem by compressing to a smaller token set or by applying efficient attention over the full one (Appendix A.2 surveys this family in detail); we compare against LNO specifically because it is architecturally closest to our construction, not because it is uniquely capable of cross-resolution generalization – ONO (Xiao et al., 2023) reports the same zero-shot Darcy transfer. This is what lets our latent evolution operator be a plain CNN and what admits the classical error analysis behind Theorem 2.3. No analogous discretization-error rate has been established for irregular-input encoders generally. The closest theoretical results, Calvello et al. (2025) for attention-based operators and Zhang et al. (2025) for a DeepONet-style branch-trunk construction, give universal approximation but not an explicit rate in the number of input points (Appendix A.2 expands this comparison). The Latent Neural PDE Solver (Li et al., 2025) and Physics-Informed Latent Neural Operator (Karumuri et al., 2026) follow the same encode→evolve→decode shape without a comparable guarantee; and prior work, Latent Twins (Chung et al., 2026; 2027), proves the approximation bound equation 5 this paper builds on but leaves its discretization-error term $\epsilon_{\text{disc}}(N)$ abstract. Gao et al. (2025) and Lanthaler et al. (2024) show, empirically and theoretically respectively, that FNO-style resolution-independence claims do not automatically hold.

4 Experiments

We evaluate the LTO along four complementary axes: (i) whether the discretization behavior predicted by Theorem 2.3 is observed empirically, (ii) one-step predictive accuracy, (iii) long-horizon behavior under direct and recursive temporal evolution, and (iv) transfer across spatial resolutions with fixed model parameters. Section 4.1 isolates the discretization effect on a smooth 2D heat equation. Sections 4.2 and 4.3 evaluate the full learned model on Burgers' equation and 2D Navier–Stokes, respectively, while Section 4.3 additionally examines long-horizon evolution. Section 4.4 considers Darcy flow as a steady-state resolution-transfer benchmark.

Setup. Training data consists of trajectories u observed at a fixed set of native time steps. The training objective is

$$\mathcal{L}(\theta) = \lambda_{\text{evol}} \underbrace{\mathbb{E}\left[\left\|\Psi_\theta^{s,c_{\text{ctx}},u_s}(t,c_q) - u_t(c_q)\right\|_2^2\right]}_{\mathcal{L}_{\text{evol}}(\theta)} + \lambda_{\text{recon}} \underbrace{\mathbb{E}\left[\left\|d_\theta\big(e_\theta(c_{\text{ctx}};u_s),c_q\big) - u_s(c_q)\right\|_2^2\right]}_{\mathcal{L}_{\text{recon}}(\theta)}, \tag{7}$$

where θ are the trainable parameters suppressed in Section 2, and $\mathcal{L}_{\text{evol}}$, $\mathcal{L}_{\text{recon}}$ are the evolution and reconstruction (autoencoder) losses, respectively. Here, the expectation is taken over n training examples $\{(u_s^i, s^i), (u_t^i, t^i)\}_{i=1}^n$, each consisting of a trajectory u^i and a source-target time pair (s^i, t^i) drawn from that trajectory's native time discretization. The values of $\lambda_{\text{evol}}, \lambda_{\text{recon}}$ are stated for each benchmark below. All error numbers are relative L^2, $\|\Psi_\theta - u\|_2/\|u\|_2$ (dropping Ψ_θ's arguments here, since they vary by evaluation setting). We additionally apply context-size augmentation, where during training we resample the context fraction $c_{\text{frac}} \in [c_{\min}, c_{\max}]$ uniformly at random on

Table 1: Burgers resolution transfer, one-shot relative L^2 (a single evolution step). Resolutions at or below the native resolution (1024) use real subsampled data; above-native points are synthetic interpolation.

Method	Params	64	128	256	512	1024 (native)	2048
LTO (ours)	29,395	**0.078**	**0.063**	**0.060**	**0.056**	**0.057**	**0.066**
FNO	23,937	0.097	0.080	0.076	0.075	0.076	0.077

every step, holding the query and target at the full native resolution throughout, and sample only the encoder's context set. Each benchmark's own $[c_{\min}, c_{\max}]$ is stated in its subsection below.

We use the standard time-evolving PDEBench (Takamoto et al., 2022) benchmarks, Burgers' equation and Navier–Stokes (Sections 4.2 and 4.3). All models are trained from a single source state u_s to the corresponding target state u_t, at a source-target time pair (s, t) drawn as above from each trajectory's native time discretization. One-step accuracy is evaluated under this same protocol; long-horizon behavior under direct and recursive evaluation is examined separately, as detailed in the relevant subsections below.

4.1 2D Heat Equation: Resolution Sweep

We test discretization invariance (Corollary 2.5) on a 2D periodic heat equation. The ground-truth field is a sum of four Gaussian kernels (standard deviations 0.12–0.22) on the periodic 2D domain, evolved forward exactly via an FFT-based spectral solve. Both kernels and their heat-flow evolution are smooth by construction, so Assumption 2.2 holds.

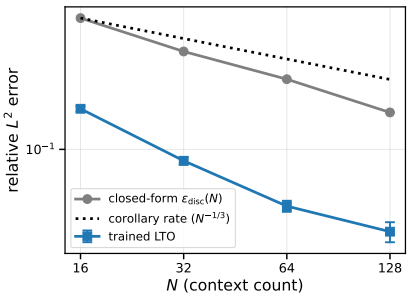

Figure 2: Closed-form $\epsilon_{\text{disc}}(N)$ and trained LTO evaluated across context points N without retraining, against Corollary 2.5's predicted rate.

We test two related quantities against the context count N. The first is $\epsilon_{\text{disc}}(N)$ itself, computed directly from the closed-form Nadaraya–Watson reconstruction equation 2, no training involved.

The second is a single trained LTO's own forecasting error: given N context points sampled from u_s, the network encodes them, evolves the latent state to a later time t, decodes a prediction, and we compare against the true field u_t at that time. The network is trained once, with the context count resampled at random on every batch, and evaluated at every N without retraining; the query count stays fixed throughout. We average this test over 5 runs. Both errors decrease at a rate consistent with the predicted slope, with a fitted log-log slope close to the rate in Corollary 2.5.

4.2 Burgers' Equation

We evaluate LTO on PDEBench's Burgers equation at $\nu = 0.001$, where solutions develop steep, near-shock fronts and therefore provide a substantially more demanding test of spatial resolution than the smooth heat-equation setting. We train LTO with $\lambda_{\text{evol}} = \lambda_{\text{recon}} = 1$ and context-size augmentation $c_{\text{frac}} \in [0.05, 1.0]$, and compare against a self-trained FNO baseline under a matched training protocol. Both models are evaluated across resolutions from 64 to 2048 points without retraining, including the above-native 2048-point interpolation regime. Table 1 summarizes the results.

LTO achieves lower one-step prediction error than FNO at every tested resolution, from 64 to 2048 points, including the above-native interpolated resolution. This advantage is therefore not confined to the training resolution or its immediate neighborhood. LTO also requires substantially less training time: approximately one quarter of FNO's wall-clock time, while using a comparable number of parameters (29,395 versus 23,937). On this benchmark, LTO therefore combines stronger cross-resolution accuracy with lower training cost.

Table 2: Navier Stokes (2D) resolution transfer, one-shot relative L^2, averaged over six data points. FNO's fixed spectral mode truncation cannot run below 32×32 – an architectural constraint, not a missing measurement.

Method	Params	16×16	32×32	64×64 (native)	128×128
LTO (ours)	1,126,211	**0.154**	**0.147**	**0.146**	6.156
FNO	1,124,193	–	0.213	0.177	1.608
LNO	1,146,945	0.261	0.253	0.251	**1.339**

Table 3: Navier–Stokes long-horizon relative L^2 at the native 64×64 resolution. All methods start from the same source state at time s. "Evolve" recursively feeds each model's prediction back as the next source state for k steps, whereas "direct" evaluates $m_{s\to t}$ from the original encoded source state without intermediate re-encoding.

Method	$k = 1$	$k = 5$	$k = 10$	Avg. (10 steps)
LTO, evolve (ours)	0.096	0.172	0.378	0.209
LTO, direct eval (ours)	0.096	**0.146**	**0.317**	**0.177**
FNO, evolve	**0.055**	0.194	0.705	0.294
LNO, evolve	0.067	0.295	1.197	0.462

4.3 Navier–Stokes (2D)

Two-dimensional Navier–Stokes is a substantially more challenging problem than Burgers. We evaluate on PDEBench's 2D Navier–Stokes forward benchmark, trained at resolution 64×64 to match LNO's own reported setting (Wang & Wang, 2024), and compare against self-trained FNO and LNO baselines with parameter count matched to LTO's.

We train LTO with $\lambda_{\text{evol}} = \lambda_{\text{recon}} = 1$, context-size augmentation with $c_{\text{frac}} \in [0.05, 4.0]$, and the pushforward trick (Brandstetter et al., 2022): for 0.15 of training steps, the model unrolls autoregressively under no gradient before one supervised step against the true target, exposing training to the compounding error it otherwise meets only at inference.

Table 2 reports one-step relative L^2 error across resolutions. At the sub-native and native resolutions (16×16, 32×32, 64×64), LTO has the lowest error and remains nearly insensitive to coarsening; above native, at 128×128, all three models deteriorate, with LTO showing the largest increase. Thus, on this problem, the resolution flexibility of the representation translates into robust transfer to coarser discretizations, but not into reliable extrapolation to spatial resolution finer than that used for training.

To assess long-horizon behavior, we recursively apply each model for $k = 1, \ldots, 10$ one-step increments. For LTO, we additionally evaluate each target time directly from the source state through $m_{s\to t}$. Table 3 shows a different ordering than Table 2's one-step comparison, since the latter averages over six context start points while the former fixes context at a particular time: FNO and LNO have lower error at $k = 1$ but grow faster under recursive evolution, and by $k = 5$ and $k = 10$ both LTO evaluations are lower, with direct evaluation giving the lowest ten-step average and confirming that avoiding repeated re-encoding limits error growth.

4.4 Darcy Flow

Darcy flow is a steady-state boundary-value problem mapping a piecewise-constant permeability field to the corresponding pressure field under a second-order elliptic PDE (Li et al., 2020). Unlike the preceding benchmarks, it has no temporal evolution, so we include it primarily for a matched comparison with LNO. Following Wang & Wang (2024), we train at 211×211 and evaluate without retraining at resolutions 85, 106, 141, 211, and 421, using $\lambda_{\text{recon}} = 1$ and $c_{\text{frac}} \in [0.05, 1.0]$.

Because the Gaussian SetConv kernel of Section 2 imposes a strong locality bias, we also evaluate an attention-based variant with learned context weights; its definition is given in Appendix A.3. This variant is empirical and is not covered by the SetConv discretization theory.

Table 4: Darcy flow: relative L^2 error across evaluation resolutions, with LNO's parameter count matched to each LTO variant. LTO variants differ in encoder (Gaussian kernel vs. learned attention).

Method	Params	85	106	141	211	421
LTO (ours)	29,139	0.133	0.133	0.133	0.133	0.132
LTO with attn (ours)	28,894	0.055	0.055	0.055	0.055	0.055
LNO (Wang & Wang, 2024)	29,171	**0.012**	**0.011**	**0.011**	**0.011**	**0.011**

Table 4 reports relative L^2 errors and parameter counts. The Gaussian LTO trails LNO by $11.80\times$ on average, while the attention variant reduces this ratio to $4.86\times$, substantially narrowing the gap although remaining less accurate at every resolution. Appendix A.3 also reports comparison with LNO's original $762{,}113$-parameter configuration.

5 Discussion and Limitations

An important feature of the LTO is its direct temporal formulation: given a single source state u_s, it maps directly to a requested target time t through $m_{s\to t}$. In the matched comparisons, FNO and LNO receive the same single source state. On Navier–Stokes, FNO and LNO are more accurate at the first step, while LTO gives lower long-horizon errors, with direct evaluation further improving over recursive evolution.

The discretization result controls only the SetConv reconstruction error and assumes C^2 states, a periodic domain, and sufficiently well-distributed context points. The autoencoder and latent-evolution errors in equation 5 remain separate contributions. The sharp error increase at 128×128 on Navier–Stokes further shows that resolution-flexible evaluation does not by itself ensure accurate above-native resolution transfer.

Finally, the SetConv analysis exploits a fixed Gaussian kernel, which restricts context weighting to physical distance. The Darcy experiment illustrates this limitation: replacing the fixed kernel with learned attention substantially reduces the approximation error, but the resulting model is no longer covered by the SetConv analysis developed here. This motivates more adaptive reconstruction operators for which quantitative discretization bounds can still be established.

6 Conclusion and Future Work

We introduced the Latent Twin Operator, a resolution-flexible encode→evolve→decode surrogate for time-evolving PDEs. For its SetConv realization, we identified the abstract discretization term in the Latent Twin approximation theory with a concrete scattered-data reconstruction error and derived an explicit $\mathcal{O}(N^{-2/(3D)})$ rate under quasi-uniform refinement. The heat equation experiment supports the predicted decay. On Burgers, LTO achieves lower one-step error than the matched FNO across all tested resolutions. On Navier–Stokes, LTO gives the lowest one-step error at native and sub-native resolutions and lower error at longer horizons, with direct temporal evaluation further reducing error accumulation. The deterioration observed above the native Navier–Stokes resolution, together with the Darcy results, also identifies limitations of the current fixed-resolution latent representation and local SetConv reconstruction.

Future work includes extending the discretization analysis to more general domains and sampling geometries, improving above-native resolution transfer, and developing adaptive or learned reconstruction operators that retain quantitative error control. Further directions include adaptive sensing and experimental design, temporal-consistency constraints for direct latent evolution, and the incorporation of physical or stochastic structure into the latent dynamics.

Reproducibility Statement

Code for the Latent Twin Operator architecture and all training and evaluation scripts used in Section 4, together with the exact PDEBench data splits for the Burgers and Navier–Stokes experiments, will be released after acceptance. All experiments use publicly available PDEBench data (Takamoto

et al., 2022); the FNO baseline for Burgers is trained from scratch on the same split rather than taken from PDEBench's own published number, which we could not independently reproduce. Hyperparameters – bandwidth ℓ, latent resolution and channel count, optimizer and learning-rate schedule, epoch counts – are recorded in the released training configs rather than the main text, to stay within the page limit. For numerical stability, the implementation adds a small fixed $\varepsilon_{\mathrm{num}} > 0$ to the denominator of the normalized SetConv channel.

AI Use Statement

Large language model assistance was used in revising prose and notation in the Introduction, Section 2, Related Work, Discussion, and Experiments sections. All theorem statements, proofs, experimental design choices, and numerical results are the authors' own.

Ethics Statement

This work concerns surrogate modeling for partial differential equations using synthetic and publicly available benchmark data (PDEBench); it does not involve human subjects, personal data, or any data collection of our own. We do not anticipate ethical concerns specific to this work beyond those generally associated with scientific machine learning research.

References

Nicolas Boullé and Alex Townsend. A mathematical guide to operator learning. In *Handbook of Numerical Analysis*, volume 25, pp. 83–125. Elsevier, 2024.

Johannes Brandstetter, Daniel Worrall, and Max Welling. Message passing neural PDE solvers. In *International Conference on Learning Representations*, 2022. arXiv:2202.03376.

Edoardo Calvello, Nikola B. Kovachki, Matthew E. Levine, and Andrew M. Stuart. Continuum attention for neural operators. *Journal of Machine Learning Research*, 26(300):1–52, 2025.

Shuhao Cao. Choose a transformer: Fourier or Galerkin. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Matthias Chung, Deepanshu Verma, Max Collins, Amit N. Subrahmanya, Varuni Katti Sastry, and Vishwas Rao. Latent twins. *Machine Learning: Science and Technology*, 7:025039, 2026. arXiv:2509.20615.

Matthias Chung, Yutong Bu, and Deepanshu Verma. Physics-conforming latent twins. *Computer Methods in Applied Mechanics and Engineering*, 463:119351, 2027.

Amer Essakine, Yanqi Cheng, Chun-Wun Cheng, Lipei Zhang, Zhongying Deng, Lei Zhu, Carola-Bibiane Schönlieb, and Angelica I Aviles-Rivero. Where do we stand with implicit neural representations. *A technical and performance survey. Submitted to Transactions on Machine Learning Research*, 2024.

Jianqing Fan and Irène Gijbels. *Local Polynomial Modelling and Its Applications*. Number 66 in Monographs on Statistics and Applied Probability. Chapman & Hall/CRC, 1996.

Wenhan Gao, Ruichen Xu, Yuefan Deng, and Yi Liu. Discretization-invariance? On the discretization mismatch errors in neural operators. In *International Conference on Learning Representations*, 2025.

Marta Garnelo, Dan Rosenbaum, Chris J. Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo J. Rezende, and S. M. Ali Eslami. Conditional neural processes. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pp. 1704–1713, 2018.

Jonathan Gordon, Wessel P. Bruinsma, Andrew Y. K. Foong, James Requeima, Yann Dubois, and Richard E. Turner. Convolutional conditional neural processes. In *International Conference on Learning Representations*, 2020.

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 *Proceedings of the International Conference on Machine Learning (ICML)*, 2023.

Sharmila Karumuri, Lori Graham-Brady, and Somdatta Goswami. Physics-informed latent neural operator for real-time predictions of time-dependent parametric pdes. *Computer Methods in Applied Mechanics and Engineering*, 450:118599, 2026.

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.

Samuel Lanthaler, Andrew M. Stuart, and Margaret Trautner. Discretization error of fourier neural operators. *arXiv preprint arXiv:2405.02221*, 2024.

Zijie Li, Kazem Meidani, and Amir Barati Farimani. Transformer for partial differential equations' operator learning. *arXiv preprint arXiv:2205.13671*, 2022.

Zijie Li, Dule Shu, and Amir Barati Farimani. Scalable Transformer for PDE surrogate modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Zijie Li, Saurabh Patil, Francis Ogoke, Dule Shu, Wilson Zhen, Michael Schneier, John R. Buchanan, and Amir Barati Farimani. Latent neural PDE solver: a reduced-order modeling framework for partial differential equations. *Journal of Computational Physics*, 2025. arXiv:2402.17853.

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*, 2020.

Lu Lu, Pengzhan Jin, and George Em Karniadakis. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.

Elizbar A Nadaraya. On estimating regression. *Theory of Probability & Its Applications*, 9(1): 141–142, 1964.

Mathew Penrose. *Random Geometric Graphs*. Number 5 in Oxford Studies in Probability. Oxford University Press, 2003.

Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems*, 33:7462–7473, 2020.

Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Daniel MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. Pdebench: An extensive benchmark for scientific machine learning. *Advances in neural information processing systems*, 35:1596–1611, 2022.

Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in neural information processing systems*, 33:7537–7547, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Tian Wang and Chuang Wang. Latent neural operator for solving forward and inverse PDE problems. In *Advances in Neural Information Processing Systems*, volume 37, 2024. arXiv:2406.03923.

Larry Wasserman. *All of Nonparametric Statistics*. Springer Texts in Statistics. Springer, 2006.

Geoffrey S Watson. Smooth regression analysis. *Sankhyā: The Indian Journal of Statistics, Series A*, 26(4):359–372, 1964.

Holger Wendland. *Scattered Data Approximation*. Number 17 in Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2005.

Haixu Wu, Tengge Hu, Huakun Luo, Jianmin Wang, and Mingsheng Long. Solving high-dimensional PDEs with latent spectral models. *arXiv preprint arXiv:2301.12664*, 2023.

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.

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

Zecheng Zhang, Wing Tat Leung, and Hayden Schaeffer. A discretization-invariant extension and analysis of some deep operator networks. *Journal of Computational and Applied Mathematics*, 456:116226, 2025.

A Appendix

A.1 Deferred Proofs

Lemma A.1 (Constancy of $\bar{\kappa}$)**.** *Under Assumption 2.2 ,* $\bar{\kappa}(x) := \int_\Omega \kappa_\ell(x,y)\,\mathrm{d}y$ *is constant for all* $x \in \Omega$*.*

Proof of Lemma A.1 (Constancy of $\bar{\kappa}$). Fix $x \in \Omega$ and substitute $z = y - x \bmod \Omega$ in the integral. Because Ω is periodic, this substitution is a bijection of Ω onto itself with Jacobian 1. Further, since $d_\Omega(x,y) = d_\Omega(0, y - x \bmod \Omega) = d_\Omega(0,z)$ then

$$\kappa_\ell(x,y) = \exp\big(-d_\Omega(x,y)^2/(2\ell^2)\big) = \exp\big(-d_\Omega(0,z)^2/(2\ell^2)\big) = \kappa_\ell(0,z)\,.$$

Substituting,

$$\bar{\kappa}(x) = \int_\Omega \kappa_\ell(x,y)\,\mathrm{d}y = \int_\Omega \kappa_\ell(0,z)\,\mathrm{d}z = \bar{\kappa}(0)\,.$$

Since $x \in \Omega$ was arbitrary, $\bar{\kappa}(x) = \bar{\kappa}(0)$ for every $x \in \Omega$.

Lemma A.2 (Encoder input stability)**.** *For any finite context locations* $c_{\mathrm{ctx}} = \{c_1, \dots, c_N\} \subset \Omega$*, any bandwidth* $\ell > 0$*, any* $\varepsilon > 0$*, and any two states* $u, v \in \mathcal{U}$*,*

$$\sup_{x\in\Omega} \big|\hat{u}_{c_{\mathrm{ctx}}}(x;u) - \hat{u}_{c_{\mathrm{ctx}}}(x;v)\big| \;\le\; \max_{1\le j\le N} |u(c_j) - v(c_j)| \;\le\; \|u - v\|_{\mathcal{U}}\,,$$

i.e. $u \mapsto \hat{u}_{c_{\mathrm{ctx}}}(\cdot\,;u)$ *is* 1*-Lipschitz,* uniformly over N, over ℓ, over ε, and over the placement of c_{ctx}.

By contrast, the raw channel only satisfies

$$\sup_{x\in\Omega} \big|\nu_{c_{\mathrm{ctx}}}(x;u) - \nu_{c_{\mathrm{ctx}}}(x;v)\big| \;\le\; \Big(\sup_{x\in\Omega} \rho_{c_{\mathrm{ctx}}}(x)\Big) \cdot \max_{1\le j\le N} |u(c_j) - v(c_j)|\,,$$

and $\sup_{c_{\mathrm{ctx}}} \sup_{x\in\Omega} \rho_{c_{\mathrm{ctx}}}(x)$ *is unbounded as* $N \to \infty$ *(e.g. placing all* N *context points within an* $o(\ell)$ *neighborhood of some* x_0 *gives* $\rho_{c_{\mathrm{ctx}}}(x_0) \to N$*), so no context-count-independent Lipschitz constant exists for the raw channel in general.*

Dividing by the local density turns each context point's contribution into a convex-combination weight, so the normalized channel's output can move by no more than the largest single input perturbation, however many context points there are or however they are placed. The raw channel skips this normalization, so an adversary who clusters points can drive $\rho_{c_{\mathrm{ctx}}}$, and with it the sensitivity, arbitrarily high.

Corollary A.3 (Explicit, resolution-independent encoder Lipschitz constant)**.** *Recall the LTO's encoder $e(c_{\mathrm{ctx}}; u) = g_\phi\big(\rho_{c_{\mathrm{ctx}}}, \hat{u}_{c_{\mathrm{ctx}}}(\cdot\,; u)\big)$ from Definition 2.1. For a fixed learned network g_ϕ (the convolutional stack described above) with Lipschitz constant L_g in its second argument (with respect to $\|\cdot\|_\infty$ on the sampled channel), holding the density channel $\rho_{c_{\mathrm{ctx}}}$ fixed,*

$$\|e(c_{\mathrm{ctx}}; u) - e(c_{\mathrm{ctx}}; v)\| \;\le\; L_g \cdot \|u - v\|_{\mathcal{U}}\,,$$

for every context set c_{ctx}, every $N = |c_{\mathrm{ctx}}|$, and every bandwidth ℓ: the encoder is Lipschitz with a constant that does not depend on the resolution or cardinality of the context set.

Feeding the raw channel $\nu_{c_{\mathrm{ctx}}}(\cdot\,; u)$ instead of $\hat{u}_{c_{\mathrm{ctx}}}(\cdot\,; u)$ does not *admit such a context-count-independent constant, by Lemma A.2: any Lipschitz bound obtained this way would need to grow with $\sup_x \rho_{c_{\mathrm{ctx}}}(x)$, and hence, in the worst case, with N. The normalized channel is what every experiment in this paper uses.*

Proof of Lemma A.2 (Encoder input stability). Fix $x \in \Omega$ and write $w_j := \kappa_\ell(x, c_j) \ge 0$, so $\rho_{c_{\mathrm{ctx}}}(x) = \sum_j w_j$ and $\nu_{c_{\mathrm{ctx}}}(x; u) = \sum_j u(c_j) w_j$. Then

$$\nu_{c_{\mathrm{ctx}}}(x; u) - \nu_{c_{\mathrm{ctx}}}(x; v) = \sum_{j=1}^{N} \big(u(c_j) - v(c_j)\big)\, w_j\,,$$

so, since $w_j \ge 0$,

$$\big|\nu_{c_{\mathrm{ctx}}}(x; u) - \nu_{c_{\mathrm{ctx}}}(x; v)\big| \;\le\; \max_j |u(c_j) - v(c_j)| \cdot \sum_j w_j \;=\; \max_j |u(c_j) - v(c_j)| \cdot \rho_{c_{\mathrm{ctx}}}(x)\,. \qquad (*)$$

This proves the raw-channel bound after taking $\sup_x$. For the normalized channel,

$$\hat{u}_{c_{\mathrm{ctx}}}(x; u) - \hat{u}_{c_{\mathrm{ctx}}}(x; v) = \frac{\nu_{c_{\mathrm{ctx}}}(x; u) - \nu_{c_{\mathrm{ctx}}}(x; v)}{\rho_{c_{\mathrm{ctx}}}(x) + \varepsilon}\,,$$

and dividing $(*)$ by $\rho_{c_{\mathrm{ctx}}}(x) + \varepsilon$ gives

$$\big|\hat{u}_{c_{\mathrm{ctx}}}(x; u) - \hat{u}_{c_{\mathrm{ctx}}}(x; v)\big| \;\le\; \max_j |u(c_j) - v(c_j)| \cdot \frac{\rho_{c_{\mathrm{ctx}}}(x)}{\rho_{c_{\mathrm{ctx}}}(x) + \varepsilon} \;\le\; \max_j |u(c_j) - v(c_j)|\,,$$

since $\rho_{c_{\mathrm{ctx}}}(x)/(\rho_{c_{\mathrm{ctx}}}(x) + \varepsilon) \le 1$ for $\rho_{c_{\mathrm{ctx}}}(x) \ge 0, \varepsilon > 0$. Taking $\sup_{x \in \Omega}$ and bounding $\max_j |u(c_j) - v(c_j)| \le \|u - v\|_{\mathcal{U}}$ gives the claim. The unboundedness of the raw bound follows by placing all N locations within distance $o(\ell)$ of a fixed x_0: each $\kappa_\ell(x_0, c_j) \to \kappa_\ell(0, 0) = 1$, so $\rho_{c_{\mathrm{ctx}}}(x_0) \to N$, which is unbounded as $N \to \infty$. Taking $u - v \equiv 1$ realizes this bound directly, $\nu_{c_{\mathrm{ctx}}}(x_0; u) - \nu_{c_{\mathrm{ctx}}}(x_0; v) = \rho_{c_{\mathrm{ctx}}}(x_0) \to N$.

Proposition A.4 (Discretization/quadrature error)**.** *Under Assumption 2.2, for every $u \in \mathcal{K}$ and $x \in \Omega$,*

$$\big|\hat{u}_{c_{\mathrm{ctx}}}(x; u) - \bar{u}_\ell(x)\big| \;\lesssim\; C_{\mathcal{K}} \cdot \frac{\delta(c_{\mathrm{ctx}})}{\ell}\,,$$

for N large enough that $(N/|\Omega|)\bar{\kappa} \gg \varepsilon$, where $C_{\mathcal{K}}$ depends on D, Ω, the Hessian bound M of Assumption 2.2, and $\sup_{u \in \mathcal{K}} \|u\|_\infty$, but not on N, ℓ, or the placement of c_{ctx}.

Proof of Proposition A.4 (Discretization/quadrature error). Fix $x \in \Omega$. Since $|V_j| = |\Omega|/N$ for every j,

$$\rho_{c_{\mathrm{ctx}}}(x) - \frac{N}{|\Omega|} \int_\Omega \kappa_\ell(x, y)\, \mathrm{d}y \;=\; \sum_{j=1}^{N} \frac{1}{|V_j|} \int_{V_j} \big[\kappa_\ell(x, c_j) - \kappa_\ell(x, y)\big]\, \mathrm{d}y\,.$$

Taylor-expand $\kappa_\ell(x, \cdot)$ around c_j: for $y \in V_j$,

$$\kappa_\ell(x, c_j) - \kappa_\ell(x, y) \;=\; -\nabla_y \kappa_\ell(x, c_j) \cdot (y - c_j) \;-\; \tfrac{1}{2}(y - c_j)^\top D_y^2 \kappa_\ell(x, \xi)(y - c_j)$$

for some ξ on the segment $[c_j, y]$. A direct computation gives $D_y^2 \kappa_\ell(x, y) = \big(\frac{(y-x)(y-x)^\top}{\ell^4} - \frac{I}{\ell^2}\big)\kappa_\ell(x, y)$, hence, by the triangle inequality on its two eigenvalue directions, the *pointwise* (not worst-case) operator-norm bound

$$\|D_y^2 \kappa_\ell(x, y)\|_{\mathrm{op}} \;\le\; h(y) \;:=\; \Big(\frac{|y - x|^2}{\ell^4} + \frac{1}{\ell^2}\Big)\kappa_\ell(x, y)\,,$$

which, like κ_ℓ itself, decays away from x; bounding each cell by its own value of $h(c_j)$, rather than a single worst-case constant, is what makes the argument below work in every dimension D. Integrating the Taylor expansion above over $y \in V_j$ (diameter $\le 2\delta$) and dividing by $|V_j|$, writing $\bar{z}_j := \frac{1}{|V_j|}\int_{V_j}(y - c_j)\,\mathrm{d}y$ for the cell's average offset from c_j (so $|\bar{z}_j| \le 2\delta$, since every point of V_j lies within 2δ of c_j), then summing over j,

$$\rho_{c_{\mathrm{ctx}}}(x) - \frac{N}{|\Omega|}\int_\Omega \kappa_\ell(x,y)\,dy \;=\; -\sum_{j=1}^{N} \nabla_y \kappa_\ell(x, c_j)\cdot \bar{z}_j \;-\; E_2\,, \qquad |\bar{z}_j| \le 2\delta\,,$$

$$|E_2| \;\le\; 2\delta^2 \sum_{j=1}^{N} h(c_j)\,\big(1 + O(\delta/\ell)\big)\,,$$

the $O(\delta/\ell)$ correction coming from replacing $h(\xi)$, $\xi \in [c_j, y]$, by $h(c_j)$ within each cell, valid since h is itself smooth at the scale $\ell \gg \delta$. Exactly as for the linear term below, h is smooth and decays like κ_ℓ, so the same cell-quadrature estimate applies: $\sum_j |V_j|\, h(c_j) = \int_\Omega h(y)\,\mathrm{d}y\,\big(1 + O(\delta/\ell)\big)$. Writing $y - x = \ell w$ (so $dy = \ell^D\,\mathrm{d}w$) gives $\int_\Omega h(y)\,\mathrm{d}y = K_D''\,\ell^{D-2}$, for the finite, x- and c_{ctx}-independent constant $K_D'' := \int_{\mathbb{R}^D}(|w|^2+1)e^{-|w|^2/2}\mathrm{d}w$ (periodization error again negligible under Assumption 2.2). Hence

$$|E_2| \;\lesssim\; 2\delta^2 \cdot \frac{N}{|\Omega|}\,K_d''\,\ell^{D-2}\,.$$

For the first (linear) term,

$$\Big|\sum_{j=1}^{N} \nabla_y \kappa_\ell(x, c_j)\cdot \bar{z}_j\Big| \;\le\; 2\delta \sum_{j=1}^{N} \big|\nabla_y \kappa_\ell(x, c_j)\big|\,,$$

using the *pointwise* gradient at each c_j rather than a coarser cell-wide supremum, for the same reason as the curvature term above: a supremum bound would overcount cells far from x. Since $g(y) := |\nabla_y \kappa_\ell(x,y)| = \ell^{-2}|y - x|\,\kappa_\ell(x,y)$ is itself smooth and decays like κ_ℓ, the same cell-averaging step applied to g gives $\sum_j |V_j|\, g(c_j) = \int_\Omega g(y)\,\mathrm{d}y\,\big(1 + O(\delta/\ell)\big)$, a standard scattered-data quadrature estimate for a smooth, rapidly-decaying integrand (Wendland, 2005). The integral evaluates exactly, by the same substitution as above, to $\int_\Omega g(y)\,dy = K_d\,\ell^{D-1}$ for the finite, x- and c_{ctx}-independent constant $K_d := \int_{\mathbb{R}^D}|w|e^{-|w|^2/2}\mathrm{d}w$ (periodization error negligible under Assumption 2.2). Hence

$$2\delta \sum_{j=1}^{N} |\nabla_y \kappa_\ell(x, c_j)| \;=\; 2\delta \cdot \frac{N}{|\Omega|}\,K_d \ell^{D-1}\big(1 + O(\delta/\ell)\big)\,,$$

which dominates the bound on E_2 above for *every* dimension d: comparing the two displays, E_2's contribution is smaller by a factor $\Theta(\delta/\ell)$, with no D-dependence left in that ratio, so it suffices that $\ell^\star = \Theta(\delta^{1/3}) \gg \delta$ as $\delta \to 0$, true in any dimension. Combining,

$$\Big|\rho_{c_{\mathrm{ctx}}}(x) - \tfrac{N}{|\Omega|}\bar{\kappa}\Big| \;\lesssim\; \tfrac{N}{|\Omega|}\cdot 2\delta K_d \ell^{D-1}\,,$$

i.e. a *relative* error of order δ/ℓ (using $\bar{\kappa} \sim \ell^D$). The same argument applies to $\nu_{c_{\mathrm{ctx}}}(x;u) = \sum_j u(c_j)\kappa_\ell(x,c_j)$ against its continuum counterpart $\frac{N}{|\Omega|}\bar{\kappa}\bar{u}_\ell(x)$, Taylor-expanding the product $u(y)\kappa_\ell(x,y)$ rather than $\kappa_\ell(x,y)$ alone. This introduces one additional term from ∇u, absent above; bounding $|\nabla u|$ via Assumption 2.2's Hessian bound M (a standard interpolation inequality on the compact domain Ω) shows this term is $O(\delta \cdot \ell^D)$ after the same cell-quadrature step, strictly smaller than the $O(\delta \cdot \ell^{D-1})$ kernel term as $\ell \to 0$, so it does not change the rate. The same relative-error conclusion follows, with an extra factor of $\sup_{u\in\mathcal{K}}\|u\|_\infty$. Propagating both relative errors through the ratio $m = \nu/(\rho + \varepsilon)$ (negligible ε for N large) gives the stated bound.

Proof of Theorem 2.3 (Explicit discretization error for the SetConv instantiation). Triangle inequality on $\hat{u}_{c_{\mathrm{ctx}}}(x;u) - u(x) = \big[\hat{u}_{c_{\mathrm{ctx}}}(x;u) - \bar{u}_\ell(x)\big] + \big[\bar{u}_\ell(x) - u(x)\big]$, combining equation 6 and Proposition A.4, then taking $\sup_{x,u}$ gives the first bound. Minimizing $g(\ell) = C_1\ell^2 + C_{\mathcal{K}}\delta/\ell$ over $\ell > 0$ gives $\ell^\star = (C_{\mathcal{K}}\delta/2C_1)^{1/3}$ and $g(\ell^\star) = \Theta(\delta^{2/3})$ by direct substitution.

Proof of the smoothing-bias bound equation 6. Fix $x \in \Omega$ and $u \in \mathcal{K}$. For a symmetric, second-order kernel, a Taylor expansion of u around x has its first-order term cancel by the kernel's $z \leftrightarrow -z$ symmetry, leaving only curvature, at rate ℓ^2 (Wasserman, 2006, Ch. 5; Fan & Gijbels, 1996). The usual design-bias term of that classical result vanishes here because $\bar{\kappa}$ is constant in x by translation invariance (Lemma A.1). Bounding the remaining curvature term by the Hessian bound M from Assumption 2.2 and summing the kernel's second moment over the D spatial dimensions gives, for every $u \in \mathcal{K}$ and $x \in \Omega$, the stated bound $\sup_{u\in\mathcal{K}} \sup_{x\in\Omega} \|\bar{u}_\ell(x) - u(x)\| \leq \frac{1}{2} DM\ell^2$.

A.2 Extended Related Work

Neural operators. Learning the solution operator of a family of PDEs directly, rather than solving each instance numerically, has been formalized as operator learning (Kovachki et al., 2023; Boullé & Townsend, 2024). DeepONet (Lu et al., 2019) represents the map via a branch/trunk decomposition; the Fourier Neural Operator (FNO; Li et al., 2020) instead parameterizes it by global convolutions computed via the FFT, which requires the input to lie on (or be resampled onto) a regular grid at both train and test time – precisely the assumption our construction is designed to avoid.

Implicit neural representations. A separate line of work represents individual functions or fields directly as coordinate networks, e.g. SIREN (Sitzmann et al., 2020) and Fourier-feature MLPs (Tancik et al., 2020); see Essakine et al. (2024) for a survey. These representations are discretization-agnostic for a single field, but do not by themselves give an encode→evolve→decode surrogate for a family of time-evolving states, which is the setting of this paper.

Conditional neural processes and SetConv. Conditional Neural Processes (Garnelo et al., 2018) and their convolutional variant, ConvCNP (Gordon et al., 2020), map an arbitrary, variable-size set of observations to a functional representation via a kernel-weighted embedding – the SetConv primitive underlying our encoder and decoder (Section 2). Our contribution is an explicit discretization-error rate for this construction, not the construction itself.

Latent-space operator learning. Several recent methods encode a possibly irregular input into a fixed-size latent state, evolve it, and decode back out, demonstrating multi-resolution behavior empirically without formally bounding the resulting error. The Latent Neural Operator (Wang & Wang, 2024) uses a cross-attention encoder/decoder (Physics-Cross-Attention) that decouples input and output sampling locations, closely paralleling the SetConv construction we analyze here; the Latent Neural PDE Solver (Li et al., 2025) trains an autoencoder onto a coarser latent grid before evolving there; and the Physics-Informed Latent Neural Operator (Karumuri et al., 2026) couples two DeepONets under a PDE-residual training objective rather than paired supervision. Prior work, Latent Twins (Chung et al., 2026) and Physics-conforming Latent Twins (Chung et al., 2027), introduces the general encode→evolve→decode pattern that the Latent Twin Operator of Definition 2.1 instantiates and proves the approximation bound equation 5 motivating this paper, but leaves its discretization-error term $\epsilon_{\text{disc}}(N)$ abstract for any concrete architecture, tackled in this paper.

Broader attention-based operator architectures. A larger family of Transformer-style operator-learning architectures targets the same problem – attention over an irregular, potentially large input set – by two genuinely different means. Transolver (Wu et al., 2024) and the Latent Spectral Model (Wu et al., 2023) share LNO's idea of compressing the input into a smaller, fixed-size latent representation before mixing it: Transolver learns physics-aware "slices" that adaptively pool mesh points into a handful of tokens before attention; the Latent Spectral Model evolves a PDE's solution in a learned, lower-dimensional spectral latent space. GNOT (Hao et al., 2023), FactFormer (Li et al., 2023), ONO (Xiao et al., 2023), and the Galerkin Transformer (Cao, 2021) instead keep a full or comparably-sized token set – one token per input point, not a compressed handful – and make attention over it affordable by other means: GNOT scales cross-attention to large point clouds via heterogeneous, gated normalization; FactFormer factorizes attention along each spatial axis separately; ONO replaces softmax attention with an orthogonal, kernel-based variant; and the Galerkin Transformer replaces softmax normalization with a Petrov-Galerkin-style linear projection, explicitly preserving sequence length rather than reducing it. Closest to our own decoupling of observation and query locations is OFormer (Li et al., 2022), which – like PhCA – uses cross-attention to map an input point set to a query set at arbitrary locations, though, like GNOT, FactFormer, and the Galerkin

Transformer, without compressing to a smaller intermediate representation. None of these seven architectures reports an explicit discretization-error rate in the number of input points; we compare empirically against LNO specifically because it is architecturally closest to the SetConv construction we analyze, not because it is uniquely capable of cross-resolution generalization – ONO (Xiao et al., 2023) reports the same zero-shot Darcy resolution transfer (train at one resolution, evaluate at another, unseen) that LNO does.

Discretization-invariance critiques. Gao et al. (2025) show empirically that FNO-style operators trained at one resolution need not perform consistently when evaluated at another, terming the resulting gap *discretization mismatch error*, and propose an aliasing-free training pipeline as a fix. Lanthaler et al. (2024) instead derive an explicit $O(N^{-s})$ discretization-error rate for FNO on a regular grid, in terms of the input's Sobolev regularity s. Theorem 2.3 is the analogous explicit rate for the SetConv/ConvCNP-style construction underlying LNO-type architectures, established for arbitrary, not necessarily gridded, context sets.

How LNO differs from our construction. Physics-Cross-Attention (PhCA) and our SetConv encoder solve the same problem – map an arbitrary, variable-size input set to a fixed-size representation, decoupled from the output query locations – but differ in three ways relevant to this paper's claims. *Mechanism*: PhCA learns query/key/value projections and a softmax attention pattern over M latent positions; our embedding step (the density and signal channels of equation 3–equation 2) has no learned parameters at all – only the CNN g_ϕ applied afterward is learned – so the encoder itself is comparatively lightweight. *Latent structure*: PhCA's M latent positions form an unstructured token set, whereas our latent state $Z \in \mathbb{R}^{H\times W\times k}$ is a genuine spatial grid, which is what lets the evolution operator $m_{s\to t}$ be a plain, translation-equivariant CNN rather than a further attention or token-mixing block. *Theory*: kernel smoothing has a century of classical nonparametric-statistics theory behind it (Nadaraya–Watson bias, scattered-data quadrature), which is what Theorem 2.3 draws on; no comparable explicit discretization-error rate exists for attention-based encoders. The closest theoretical results, Calvello et al. (2025) for attention-based operators and Zhang et al. (2025) for a DeepONet-style branch-trunk construction, establish universal approximation but not an explicit rate in the number of input points, leaving this an open question for PhCA-style architectures specifically. *Time evolution*: for time-dependent PDEs, Wang & Wang (2024) decode back to geometric space at every intermediate time step in order to compute the training loss there; our latent flow $m_{s\to t}$ (Section 2) never leaves the latent space between the initial encode and the final decode, chaining as many steps as the rollout curriculum requires without a round trip through geometric coordinates. Section 4.3 reports a direct empirical comparison against LNO on a shared 2D Navier–Stokes benchmark, where this architectural difference and the discretization-error guarantee of Theorem 2.3 can be checked against a trained baseline rather than argued from architecture alone.

A.3 Darcy Flow: Attention mecahsnism and Native LNO Parameter Budget

Section 4.4 additionally evaluates a learned-attention variant of the SetConv encoder. Recall from Section 2 that the fixed Gaussian kernel weights each context point c_j by $\kappa_\ell(x, c_j) = \exp\big(-d_\Omega(x, c_j)^2/(2\ell^2)\big)$, a function of distance alone. The attention variant instead replaces this fixed kernel with a learned score,

$$\alpha(x, c_j) = \mathrm{softmax}_j\left(\frac{q(x)^\top k(c_j)}{\sqrt{d_k}}\right),$$

where $q, k : \Omega \to \mathbb{R}^{d_k}$ are learned embeddings of the query and context positions into a shared d_k-dimensional space, and the $\sqrt{d_k}$ scaling follows standard scaled dot-product attention (Vaswani et al., 2017). Unlike κ_ℓ, α is free to place weight on a distant c_j whenever doing so reduces training loss.

Finally, Table 4 of Section 4.4 matches LNO's parameter count to each LTO variant for a controlled comparison. For reference, Table 5 reports an additional comparison using LNO's originally reported $762{,}113$-parameter configuration. The resulting ratios ($12.99\times$ for the plain kernel, $3.69\times$ for the attention encoder) are close to the parameter-matched ones, so the gap in Section 4.4 is not primarily an artifact of LNO's larger parameter budget.

Table 5: Darcy flow: relative L^2 error across evaluation resolutions. LTO variants differ in encoder (Gaussian kernel vs. learned attention).

Method	Params	85	106	141	211	421	Mean Ratio
LTO (ours)	29,139	0.099	0.096	0.093	0.095	0.218	12.99×
LTO with attn (ours)	240,594	0.035	0.035	0.035	0.035	0.035	3.69×
LNO (Wang & Wang, 2024)	762,113	0.012	0.010	0.009	0.008	0.009	1.00× (ref.)