AN APPROXIMATE OPERATOR-BASED LEARNING METHOD FOR THE NUMERICAL SOLUTIONS OF STOCHASTIC DIFFERENTIAL EQUATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Stochastic differential equation (SDE in short) solvers find numerous applications across various fields. However, in practical simulations, we usually resort to using Itô–Taylor series-based methods like the Euler–Maruyama method. These methods often suffer from the limitation of fixed time scales and recalculations for different Brownian motions, which lead to computational inefficiency, especially in generative and sampling models. To address these issues, we propose a novel approach: learning a mapping between the solution of SDE and corresponding Brownian motion. This mapping exhibits versatility across different scales and requires minimal paths for training. Specifically, we employ the DeepONet method to learn a nonlinear mapping. And we also assess the efficiency of this method through simulations conducted at varying time scales. Additionally, we evaluate its generalization performance to verify its good versatility in different scenarios.

1 INTRODUCTION

Stochastic differential equations (SDEs) describe dynamical systems subject to random forcing, with applications in finance, hydrology, and climatology (Mao, 2008). Their numerical solvers also support diffusion-based generative models (Song et al., 2021) and SDE-based Monte Carlo algorithms (Durmus et al., 2019). Accurate simulation often requires both small time steps and many sample paths. With Itô–Taylor methods such as Euler–Maruyama (Kloeden & Platen, 1992), each new Brownian path requires another sequence of numerical updates. We ask whether previously computed paths can be used to train a model that predicts solutions for new driving noise.

In this paper, we consider SDEs driven by Brownian motions. By the Yamada–Watanabe theorem (Yamada & Watanabe, 1971), weak existence and pathwise uniqueness yield a strong solution that can be represented as a measurable functional of the initial value and the driving Brownian motion:

$$X_t = \mathbf{F}(X_0, B)(t) \quad \text{a.s.}$$

The value at time t depends only on X_0 and the Brownian history $B_{[0,t]}$. We approximate this solution map by supervised learning, using initial states and Brownian paths as inputs and the corresponding numerical SDE solutions as labels.

Our model builds on DeepONet (Lu et al., 2021). A recurrent network encodes the observed Brownian history, and the branch network combines this encoding with the initial state. The trunk network takes the query time as input. The inner product of their outputs gives the predicted state at that time. The recurrent encoding lets the predictor use the entire observed noise prefix while remaining causal. Once trained, the model can be evaluated on new Brownian paths for the same equation.

Under suitable well-posedness conditions, the same construction applies to stochastic partial differential equations (SPDEs) (Tappe, 2013). We test stochastic heat and nonlinear reaction equations (Appendix L). We also construct an SDE from samples of a four-mode distribution using a Doob h-transform, then learn its solution map to generate new samples (Section 4.2).

Our main contributions are:

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.

- **Learning the noise-to-solution map.** We formulate a recurrent DeepONet for learning numerical SDE solution paths from Brownian histories and extend it to direct SPDE field prediction.
- **Sample generation in a single network evaluation.** We construct a sampling SDE from data and learn its Brownian-to-solution map. Once trained, a single forward evaluation of the recurrent DeepONet maps a new Brownian path to a generated sample at the terminal time, without numerical SDE integration at inference. A multimodal experiment evaluates both trajectory accuracy and the generated distribution.
- **Extensive numerical evaluation.** We test scalar SDEs, McKean–Vlasov particle systems, coupled systems in two and eight dimensions, and two SPDEs. The studies also examine temporal resolution and prediction for unseen noise and initial states.
- **Controlled comparisons and accuracy validation.** A six-architecture benchmark and repeated comparisons isolate the effects of noise representation, training data, and optimization budget. The preferred encoder depends on the inputs and training budget. Exact-time solutions and refined numerical references measure the error in the labels.
- **Computational cost and model reduction.** We measure the cost of generating labels, training, and repeated prediction. We also compress the trained output network and bound the resulting change in predictions on the training grid.

2 Related work

Operator learning. DeepONet represents an operator through input-dependent branch coefficients and a trunk evaluated at query coordinates (Lu et al., 2021). Fourier neural operators use integral layers (Li et al., 2021; Kovachki et al., 2023), and MIONet combines multiple input functions (Jin et al., 2022). For our problem, the operator input is the initial state together with a noise history. Finite observations of that history introduce an encoding error, in addition to approximation and reconstruction errors (Lanthaler et al., 2022).

Noise-to-solution maps. SDEONet uses a sparse Wiener-chaos expansion and analyzes its convergence and complexity (Eigel & Miranda, 2025). Chaos coefficients and Wick–Hermite features have also been used to approximate SPDEs and stochastic solution operators (Neufeld & Schmocker, 2024; Shi et al., 2026a). MFNO uses Fourier operators for nonperiodic paths (Lee et al., 2025), and NeuralChaos studies approximation of predictable stochastic processes (Kratsios et al., 2026). We encode increments or cumulative Brownian values with a recurrent network whose parameters are learned jointly with the solution predictor.

SPDE solution learning. Neural SPDEs learn the dependence on both an initial field and a driving noise realization, using a spectral formulation based on mild solutions (Salvi et al., 2022). We consider the same input–output task with a recurrent encoder of physical-space noise observations and a branch–trunk field predictor. DLR-Net builds features from regularity structures (Gong et al., 2023), while WCE-FiLM-NO combines chaos features with feature-wise linear modulation (FiLM) for singular SPDEs (Shi et al., 2026b). SPDEBench examines how data generation and noise approximation affect evaluation (Zhu et al., 2026). In our field experiments, coarse and refined reference solutions share the same underlying noise.

Path representations and stochastic models. Neural CDEs, Neural RDEs, and randomized signatures encode histories using controlled dynamics, interval log-signatures, and fixed random features, respectively (Kidger et al., 2020; Morrill et al., 2021; Monzio Compagnoni et al., 2023). We include randomized signatures in the scalar benchmark. Neural SDEs fit stochastic dynamics through differentiable solvers or distributional objectives (Li et al., 2020; Kidger et al., 2021); MD-NOMAD models conditional response distributions (Thakur & Chakraborty, 2024). In the present supervised problem, each target trajectory is paired with the particular noise realization that produced it. Schrödinger–Föllmer sampling constructs an SDE with a prescribed terminal law (Huang et al., 2025); diffusion Schrödinger bridges also constrain the initial law (De Bortoli et al., 2021). Consistency models learn direct noise-to-data maps for one-step generation (Song et al., 2023). Our sampling experiment learns the solution paths of a data-fitted Föllmer SDE from paired Brownian and solution trajectories, producing terminal samples in a single recurrent-network evaluation.

Interacting-particle models. Particle methods approximate law-dependent dynamics through empirical distributions (Bossy & Talay, 1997); neural McKean–Vlasov models learn distribution-dependent coefficients (Yang et al., 2024). Observing one particle's noise leaves the other particles' influence unresolved. The resulting conditional prediction problem is described in Appendix A.2.

Conditioning and compression. FiLM and temporally conditioned operators motivate feature modulation (Perez et al., 2018; Oommen et al., 2024), while SVD-DeepONet and flexDeepONet relate operator representations to low-rank structure (Venturi & Casey, 2023). We factor the fitted time-dependent coefficients to reduce the output network after training, with a bound on prediction changes on the query grid. Appendix E discusses the related formulations in more detail.

3 Learning a causal Brownian-to-solution operator

3.1 SDE formulation and causal data generation

For $X_t \in \mathbb{R}^d$ and an m-dimensional Brownian motion B_t, consider

$$dX_t = a(t, X_t)\,dt + b(t, X_t)\,dB_t, \qquad X_0 \sim \nu.$$

A standard sufficient condition is that a, b are Borel measurable, globally Lipschitz in x, and of at most linear growth, uniformly for $t \in [0, T]$, with $\mathbb{E}\|X_0\|^2 < \infty$ and X_0 independent of B. Then a pathwise unique, nonexplosive strong solution exists (Mao, 2008). The cubic-drift benchmarks instead satisfy the local Lipschitz and Lyapunov conditions in Appendix A.1. The solution is then a measurable, nonanticipative function of (X_0, B) (Yamada & Watanabe, 1971). We train on a numerical approximation to this function.

For a fixed grid $t_k = kh$, $k = 0, \ldots, K$, and $T = Kh$, numerical SDE solution trajectories used as labels can be generated by any suitable, sufficiently accurate SDE solver, including higher-order methods, with each trajectory paired with its driving Brownian path. In this work, we use the Euler–Maruyama map F_h:

$$X^h_{k+1} = X^h_k + a(t_k, X^h_k)h + b(t_k, X^h_k)\Delta B_k, \qquad \Delta B_k = B_{t_{k+1}} - B_{t_k}.$$

Thus F_h maps $(X_0, \Delta B_{0:K-1})$ to $X^h_{0:K}$. The prediction at t_k uses only increments $\Delta B_0, \ldots, \Delta B_{k-1}$. To align inputs and state times, define $u_0 = 0$ and $u_k = \Delta B_{k-1}$ for $k \geq 1$. The sequence input z_k is B_{t_k}, u_k, or (u_k, B_{t_k}). Since $B_0 = 0$, these representations contain the same prefix information on a fixed grid, allowing us to compare different encodings of the same driving history.

3.2 Related stochastic equations

The representation of a solution as a function of its initial state and driving noise extends to a broad class of stochastic equations. With the equation and initial law fixed, suitable conditions ensuring strong existence and pathwise uniqueness give a measurable, causal solution map (Yamada & Watanabe, 1971; Tappe, 2013). This viewpoint also applies to the distribution-dependent dynamics, stochastic fields, and sampling equations below, with the full initial state and driving noise specified for each system.

McKean–Vlasov SDEs. In an MV-SDE, the coefficients depend on the solution law:

$$dX_t = a(t, X_t, \mu_t)\,dt + b(t, X_t, \mu_t)\,dB_t, \qquad \mu_t = \mathcal{L}(X_t).$$

Particle methods replace μ_t by an empirical distribution, requiring all particles to be advanced together at each step (Bossy & Talay, 1997). We use the numerical particle solutions as labels (Appendix F). Since the predictor observes only one particle's initial state and noise history, its squared-loss target on an independent N-particle system is $\mathbb{E}[X^{n,N}_k \mid X^n_0, B^n_{t_0:t_k}]$. Appendix A.2 quantifies the uncertainty from unobserved particle noises.

Stochastic partial differential equations. An SPDE evolves a random spatial field. We consider

$$dU_t = [AU_t + f(U_t)]\,dt + g(U_t)\,dW^Q_t,$$

where A is a spatial differential operator, f is the reaction term, and g scales the Wiener noise with spatial covariance Q. Under the assumptions in Appendix B, the mild solution satisfies $U_t =$

Algorithm 1 Recurrent Brownian-to-solution learning on a fixed grid

Require: Paired fitting paths $(X_0, \Delta B_{0:K-1}, X^h_{0:K})$; training budget and checkpoint rule specified for the study.

1: Fit input/output scaling on training paths; form aligned $z_{0:K}$.
2: **for** each training minibatch **do**
3: Compute causal features $h_k = E_\theta(\widetilde{z}_{0:k}, t_{0:k}/T)$.
4: Evaluate the branch–trunk output and restore state units for $k > 0$; impose $\widehat{X}_0 = X_0$.
5: Update encoder, branch, and trunk by minimizing the mean squared error (MSE) between predictions and numerical solution labels in standardized coordinates.
6: **end for**
7: Select the checkpoint by the study's rule without test data; freeze weights and scaling.
Prediction for a new input $(X_0, \Delta B_{0:K-1})$**:**
8: Form and scale $z_{0:K}$ using saved statistics; reset the encoder.
9: Repeat lines 3–4 with frozen weights; return $\widehat{X}_{0:K}$.

$\mathcal{G}_t(U_0, W^Q|_{[0,t]})$ (Tappe, 2013). We use initial fields and noise increment fields to predict solution values at time–space queries, with numerical SPDE solutions as labels. The heat and nonlinear reaction experiments use random initial fields (Appendix L).

Stochastic differential equations for data generation. Given only data points $\{y_i\}_{i=1}^M$, we can construct an SDE for generating new samples without knowing or evaluating their underlying density. The data-based Doob construction takes the form

$$dX_t = \widehat{\beta}(t, X_t)\, dt + dB_t, \qquad X_0 = 0,$$

where the drift can be estimated by replacing the expectations in (5) with averages over the data points. The resulting empirical target can be smoothed to regularize the drift near $t = 1$ (Appendix A.4). Numerical trajectories paired with their Brownian paths provide training data for the recurrent operator. After training, a single forward evaluation maps each new Brownian path to a predicted trajectory whose endpoint provides a generated sample. This evaluation encodes the noise history with the recurrent network and requires no numerical SDE integration at inference. Section 4.2 evaluates both trajectory accuracy and the generated distribution.

All sampling paths start from the same fixed point $X_0 = 0$, while independent Brownian paths produce different terminal samples. The learned operator therefore uses randomness in the driving trajectories to generate samples from a fixed initial state. In deterministic flow-based models, such as flow matching, randomness instead enters through an initial draw from a base distribution, which is then transported by a deterministic flow (Lipman et al., 2023).

3.3 Recurrent representation and output network comparisons

Let $\widetilde{z}_{0:k} = (\widetilde{z}_0, \ldots, \widetilde{z}_k)$ denote the noise-input history through t_k, with tildes indicating coordinate-wise standardization fitted on training trajectories. A causal encoder summarizes this history and its normalized time coordinates as $h_k = E_\theta(\widetilde{z}_{0:k}, t_{0:k}/T)$. Let μ_X and the diagonal matrix S denote the output mean and scale. The recurrent DeepONet predicts, for $k > 0$,

$$\widehat{y}_{k,q} = c_q + \frac{1}{\sqrt{p}} \sum_{j=1}^{p} b_{qj}(h_k, \widetilde{X}_0)\tau_j(t_k/T), \quad \widehat{X}_k = \mu_X + S\widehat{y}_k,$$

where $q = 1, \ldots, d$ and p is the branch–trunk latent width. The controlled implementations set $\widehat{X}_0 = X_0$ explicitly. Training minimizes MSE between predictions and numerical solution labels, both in standardized coordinates. The encoder is a GRU unless specified otherwise; Transformer encoders use a triangular causal mask. Figure 1 shows the construction; Algorithm 1 gives the training and prediction procedure. The direct SPDE implementation is given in Algorithm 2 and Figure 11.

The trunk weights the branch features according to the query time. To test whether this multiplication improves prediction, we compare the branch–trunk output network with a concatenation MLP,

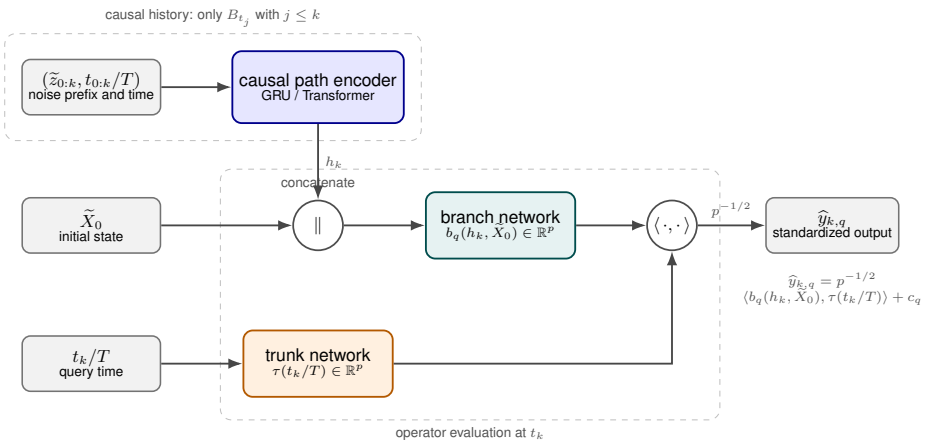


Figure 1: Recurrent DeepONet at a noninitial query t_k. The encoder receives only the standardized noise prefix and its time coordinates. Each output coordinate has p branch coefficients and shares the time trunk. The output network produces standardized values; output scaling restores state units. The initial state is imposed separately.

$f(h_k, \widetilde{X}_0, t_k/T)$, and a FiLM-style output network,

$$\widehat{y}_k = W \tanh\{\gamma(t_k/T) \odot \psi(h_k, \widetilde{X}_0) + \beta(t_k/T)\} + c.$$

FiLM adds time-dependent shifts, applies a nonlinearity after modulation, and mixes the resulting features through a learned linear layer. In the independent comparison, output-network parameter counts differ by less than 0.1%. Models with the same task and inputs share data and encoder initialization within each experiment. Each encoder is trained jointly with its output network. Appendix J reports ablations of the noise representation, encoder, and output network. The affine last branch layer also permits compression of its fitted time-dependent coefficients (Appendix K).

3.4 Approximation guarantee

The following result characterizes what the branch–trunk predictor can learn from a fixed causal representation. On a finite positive-time query grid, let $Y = (Y_{kj})$ denote numerical solution labels, with spatial index j omitted for scalar SDEs. Define $\|v\|_G^2 = \sum_{k,j} w_{kj}|v_{kj}|^2$ for fixed positive weights summing to one. Let $V_k \in \mathbb{R}^r$ contain the encoder features and any initial-state inputs supplied to the output network.

Theorem 1 (Approximation from causal features)**.** *Assume $\mathbb{E}\|Y\|_G^2 < \infty$, fix the causal encoder, and suppose the deterministic trunk inputs distinguish the positive-time query nodes. Let $\mathcal{H}_E$ consist of branch–trunk networks with one or more tanh hidden layers and affine output layers, with unrestricted weights and biases, hidden widths, and latent dimension. With $m_{kj}^E = \mathbb{E}[Y_{kj} \mid V_k]$,*

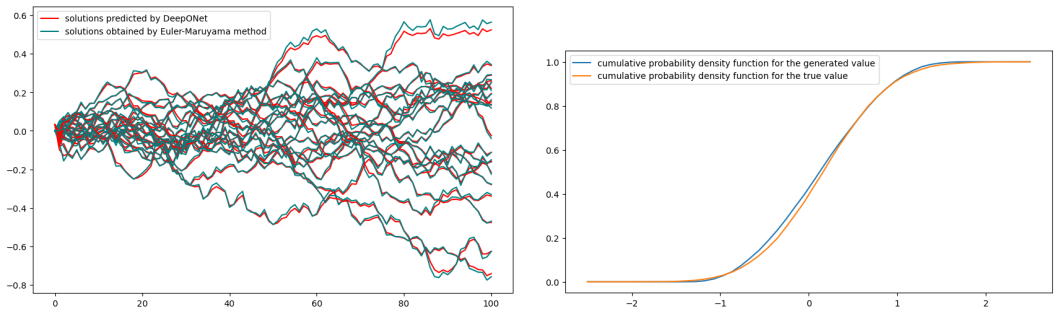
$$\inf_{f \in \mathcal{H}_E} \mathbb{E}\|f - Y\|_G^2 = \mathbb{E}\|Y - m^E\|_G^2.$$

In particular, the infimum is zero if each Y_{kj} is measurable with respect to V_k.

The proof is given in Appendix C, Proposition C.2. The residual measures information missing from the observations or lost by the encoder. This result concerns approximation of fixed-grid numerical labels; optimization and numerical-solver errors are separate (Appendix A.3).

4 Experiments

We begin with path prediction and terminal sampling, then compare architectures, input representations, and training budgets. We evaluate numerical-reference error and computational cost separately. Unless specified otherwise, uncertainty is the sample standard deviation across independent experiments, each using regenerated data and reinitialized models.



(a) Predictions for unseen OU noise paths.

(b) Terminal McKean–Vlasov CDF.

Figure 2: Path prediction and terminal sampling, each using 20 training trajectories. Left: displayed predictions from an 800-path evaluation set, compared with Euler–Maruyama. Right: empirical cumulative distribution functions (CDFs) for 10,000 terminal values, with "true value" denoting the Euler particle approximation.

For n_{test} paths, the default evaluation metric is raw-unit path RMSE,

$$\text{RMSE}(\widehat{X}, X^h) = \left[\frac{1}{n_{\text{test}}(K+1)d} \sum_{i=1}^{n_{\text{test}}} \sum_{k=0}^{K} \|\widehat{X}_{i,k} - X^h_{i,k}\|_2^2 \right]^{1/2}.$$

We specify coordinate RMSE, MSE, or normalized MSE where these are used.

4.1 OU SDE ACROSS TIME-STEP SCALES

Consider the Ornstein–Uhlenbeck (OU) equation

$$dX_t = \theta(\mu - X_t)\, dt + \sigma\, dB_t, \qquad X_0 = x_0,$$

where $\theta > 0$ is the mean-reversion rate, μ is the long-term mean, and $\sigma > 0$ is the noise amplitude. We train on 20 paired Brownian and numerical solution paths and test on 800 new paths generated independently of the training data. Figure 2(a) compares the predictions with the Euler–Maruyama solutions. The per-path MSE is approximately 10^{-5} to 5×10^{-4}; Appendix H includes the training paths and error histogram.

We also compare prediction errors at seven time-step scales: $h/1000$, $h/100$, $h/10$, h, $10h$, $100h$, and $1000h$. Table 3 (Appendix H) reports higher normalized MSE away from the base step size: 1.335×10^{-3} at h, versus 3.090×10^{-1} at $100h$. These results show sensitivity to the time-step scale. The available normalization and resampling details are incomplete, so this experiment does not establish step-size transfer.

4.2 FROM SAMPLED DATA TO TRAJECTORIES AND GENERATED SAMPLES

We next ask whether learning solution paths can recover a prescribed multimodal distribution. In each of five independent experiments, we draw 10,000 points from the four-Gaussian target in (6) and fit a Gaussian mixture using only those points. The fitted density defines a Doob-transformed SDE starting at zero, whose time-one law is that density (Appendix A.4). Euler simulation with 128 steps supplies 4096 paired Brownian and solution paths for training and 1024 for validation. The recurrent DeepONet then predicts trajectories from new Brownian paths; their endpoints provide generated samples.

On 2048 unseen paths per experiment, path RMSE is 0.0503 ± 0.0064 (4.00% relative error) and terminal RMSE is 0.1473 ± 0.0207. Paired endpoints occupy the same mode in 98.94% of cases. For 10,000 generated samples per experiment, the sliced Wasserstein distance SW_2 (Bonneel et al., 2015) to independent target data is 0.0707 ± 0.0179, compared with 0.0715 ± 0.0196 for Euler endpoints. Figure 3 shows the generated distribution and the time-resolved errors. Appendix H.2 reports mode frequencies, within-mode dispersion, sampling variability, and paired grid-refinement results. Figure 7 in the appendix shows how generated samples change as the training loss decreases.

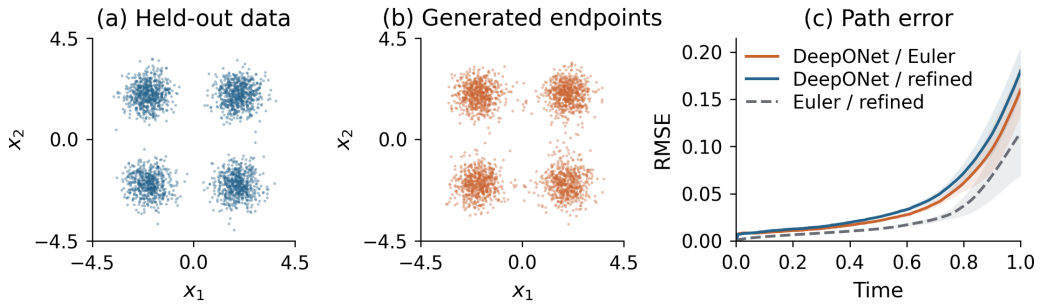


Figure 3: Data-driven sampling after pathwise training. Left and center: 2500 held-out data points and generated endpoints from the first specified experiment. Right: time-resolved RMSE on 512 new Brownian paths per experiment, unseen during training and validation. Curves and shading show the mean and sample standard deviation over five experiments. "Euler" uses 128 time steps; "refined" denotes an Euler reference with 2048 steps on the same Brownian paths. Each curve compares the two solutions named in its legend at the coarse-grid times.

4.3 McKean–Vlasov sampling and scalar SDE prediction

The Burgers-type McKean–Vlasov example uses the cumulative distribution function (CDF) of the solution law as its drift, $dX_t = F_{\mu_t}(X_t)\,dt + \sigma\,dB_t$ (Bossy & Talay, 1997). The particle method in Section 3.2 generates the training labels by updating the empirical CDF at every step. The trained individual-particle predictor then reuses these simulations to predict responses to new noise paths.

The sampling experiment uses $h = 0.01$, $\sigma = 1$, and 10,000 reference particles. Twenty trajectories are used for training, and 10,000 terminal predictions give Figure 2(b). This compares terminal distributions, while the training loss measures pathwise error. The conditional-mean interpretation of individual-particle prediction is given in Appendix A.2; the reference-ensemble construction is documented in Appendix H. Table 7 there reports CPU timings for particle simulation, training, and prediction, with differences in the input-generation workloads noted alongside it.

We next compare five architectures on the linear, double-well, and finite-particle CDF-drift equations. We conduct five experiments, each using 20 training and 1000 test paths. The pointwise MLP receives only (B_{t_k}, t_k, X_0); the recurrent models are causal, while CNN and FNO receive the full path. The randomized-signature baseline fits an affine map on a fixed reservoir. Table 1 gives parameter counts; Appendix H gives the training settings. This comparison varies the complete network architecture; the controlled output-network comparisons appear in Appendix J. For the same scalar tasks, Table 14 in Appendix K reports reference-data generation and training costs, as well as recurrent-DeepONet inference times compared with Euler–Maruyama.

Among the architectures shown in Table 1, recurrent DeepONet has the lowest mean RMSE on all three tasks. Here, the mean-field training and test particles belong to one interacting system. To assess transfer to new systems, the controlled study below uses independently generated validation

Model	Params	Linear	Double well	Mean field
Pointwise MLP	17,153	0.0412 ± 0.0057	0.0385 ± 0.0102	0.0712 ± 0.0144
Temporal CNN	15,553	0.0336 ± 0.0023	0.0328 ± 0.0030	0.0553 ± 0.0022
FNO	15,745	0.0345 ± 0.0051	0.0303 ± 0.0080	0.0577 ± 0.0155
Randomized signature	66	0.0386 ± 0.0113	0.0273 ± 0.0044	0.0448 ± 0.0170
Recurrent DeepONet (ours)	15,473	$\mathbf{0.0277 \pm 0.0143}$	$\mathbf{0.0136 \pm 0.0129}$	$\mathbf{0.0346 \pm 0.0180}$

Table 1: Full-path RMSE in the original state scale on three scalar SDE tasks (mean $\pm$ sample standard deviation, five experiments), using 20 training and 1,000 test paths. Bold marks the lowest mean error among the models shown. Baselines are our task-specific implementations of MLP (Rumelhart et al., 1986), CNN (LeCun et al., 1998), FNO (Li et al., 2021), and randomized signatures (Monzio Compagnoni et al., 2023). MLP omits path history; CNN and FNO are noncausal.

and test ensembles: independent-system/same-system mean RMSE ratios are 0.985–1.047 across eight configurations, with model RMSE approximately 0.026–0.039. A resampling diagnostic estimates conditional noise-related RMSE near 0.00362 at 1020 particles. This estimate is below the model errors. Testing on independent systems changes the mean errors by less than 5% for these configurations (Appendix I).

We also explore weight prediction for a two-layer neural network under noisy-gradient training. Treating the interacting weight dynamics as a McKean–Vlasov SDE, the operator predicts individual weight values from initial parameters and optimizer noise. A width-100 tanh network fitting $y = \sqrt{x}$ provides a toy setting for exploring extrapolation beyond an observed training segment, linking future weight predictions to the corresponding network loss. This application suggests a route to forecasting neural-network optimization through a learned solution operator. Appendix N presents this exploratory prototype and its implementation limitations.

4.4 Stochastic heat and reaction equations

We predict solution fields for the SPDE

$$du = [0.02\, u_{xx} + f(u)]\, dt + 0.5\, dW_\ell, \qquad f(u) = 0 \quad \text{or} \quad f(u) = 2 \sin u,$$

on the periodic domain $x \in [0, 1)$ with $T = 1$. The two choices give the stochastic heat and nonlinear reaction equations, respectively. Each realization has a random initial field and spatially colored noise with smoothing width $\ell = 0.08$. The network maps the initial field and noise history to solution values at time–space queries. A semi-implicit solver on 64 spatial cells and 64 time intervals provides the numerical SPDE solutions used as labels.

We conduct five experiments, each using 512 training, 128 validation, and 512 test trajectories. DeepONet achieves mean relative field L^2 errors of 1.63% for heat and 2.50% for nonlinear reaction, compared with 2.00% and 2.69% for a field MLP with the same encoder architecture and a matched parameter count. The learned linear operator reproduces the heat labels almost exactly, but its error rises to 14.11% on the nonlinear reaction equation. Appendix L provides the initial-field distribution, training details, solution-field plots, and accuracy checks against refined references (Table 15; Figures 12 and 13).

4.5 Coupled vector dynamics

To test interactions between state coordinates and multiple noise channels, we use the coupled system

$$\begin{aligned} dX_1 &= (-0.8X_1 + 0.2X_2)\, dt + (0.5 + 0.1 \tanh X_2)\, dB_1 + 0.1\, dB_2, \\ dX_2 &= (-0.3X_1 - 0.6X_2 - 0.1X_2^3)\, dt + 0.2\, dB_1 + (0.4 + 0.1 \tanh X_1)\, dB_2. \end{aligned}$$

All models use a branch–trunk output network. With Brownian-position inputs, Transformer has lower mean coordinate errors than GRU; with increments, GRU outperforms the wider Transformer. Appendix J examines this dependence on noise representation under matched training budgets.

Jointly increasing training data and model capacity reduces MSE by a factor of 8.68. Longer training and validation-based checkpoint selection also improve accuracy (Figure 4). These results show how prediction quality depends on data, capacity, and optimization; Appendix H.5 gives the settings and detailed results.

4.6 Ablation, computational cost, and compression

Appendix J compares noise representations, encoders, and output networks; branch–trunk improves accuracy in most matched comparisons. Appendix K reports data-generation, training, and prediction costs against Euler–Maruyama. Inference is faster for the mean-field task, but slower for the linear and double-well tasks; these timings do not establish a speedup at matched accuracy. Compressing the output network reduces total parameters by 24.6%–46.2%, with test RMSE increasing by at most 5% for 29 of 30 models.

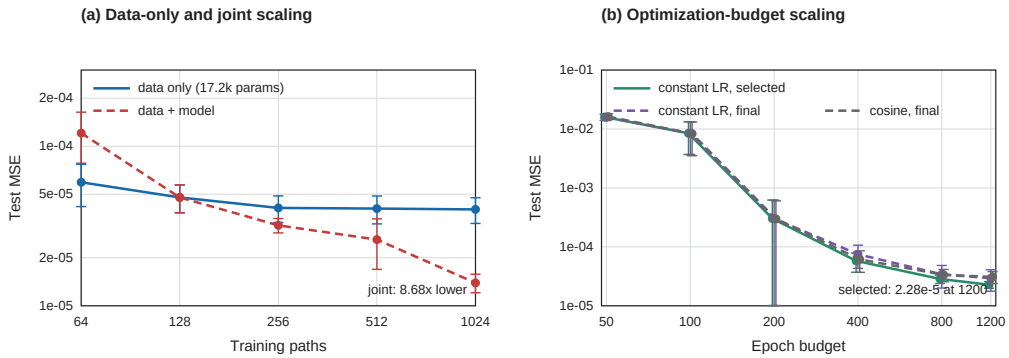


Figure 4: Coupled-system scaling, five experiments. Left: increasing training data at fixed capacity versus jointly increasing data and capacity. Right: training-duration comparisons at fixed data and model size. Error bars show sample standard deviations; MSE is computed after restoring the original state scale.

5 Discussion and Conclusion

We developed a recurrent DeepONet that maps initial states and Brownian paths to numerical SDE solutions. Theorem 1 establishes approximation on a fixed grid, with a residual determined by information missing from the encoded features. Experiments cover scalar and coupled SDEs, McKean–Vlasov systems, and stochastic heat and reaction equations. The branch–trunk network improves accuracy in most matched comparisons and admits compression with a bound on the prediction change (Appendices J and K).

For data generation, the trained network maps a Brownian path to a terminal sample in a single forward evaluation, without numerical SDE integration at inference. Different driving paths generate different samples from the same fixed initial point; deterministic flow models instead obtain randomness from their initial state.

Accuracy depends on the equation, noise representation, and training conditions, as well as numerical label error and unresolved noise. The current guarantees concern a fixed grid, and each equation is learned separately. Computational savings depend on the reference solver and whether repeated predictions offset training costs. Generalization across time resolutions and equation parameters, and efficiency at matched accuracy, remain directions for future work.

AI use statement

Generative AI tools were used to assist with drafting and language polishing, organizing the related literature, and improving the presentation and interpretation of experimental results. All AI-assisted content was reviewed and verified by the authors, who take full responsibility for the final manuscript.

References

Christian Beck, Sebastian Becker, Patrick Cheridito, Arnulf Jentzen, and Ariel Neufeld. Deep learning based numerical approximation algorithms for stochastic partial differential equations. *arXiv preprint arXiv:2012.01194*, 2020. URL `https://arxiv.org/abs/2012.01194v3`. Version 3, revised September 2026.

Nicolas Bonneel, Julien Rabin, Gabriel Peyré, and Hanspeter Pfister. Sliced and Radon Wasserstein barycenters of measures. *Journal of Mathematical Imaging and Vision*, 51(1):22–45, 2015. doi: 10.1007/s10851-014-0506-3.

Mireille Bossy and Denis Talay. A stochastic particle method for the mckean-vlasov and the burgers equation. *Math. Comput.*, 66:157–192, 1997. doi: 10.1090/S0025-5718-97-00776-X.

Valentin De Bortoli, James Thornton, Jeremy Heng, and Arnaud Doucet. Diffusion Schrödinger bridge with applications to score-based generative modeling. In *Advances in Neural Information Processing Systems*, volume 34, pp. 17695–17709, 2021. URL https://arxiv.org/abs/2106.01357.

Alain Durmus, Szymon Majewski, and Błażej Miasojedow. Analysis of langevin monte carlo via convex optimization. *Journal of Machine Learning Research*, 20(73):1–46, 2019. URL https://jmlr.org/papers/v20/18-173.html.

Martin Eigel and Charles Miranda. Functional SDE approximation inspired by a deep operator network architecture. *arXiv preprint arXiv:2402.03028v2*, 2025. URL https://arxiv.org/abs/2402.03028v2.

Shiqi Gong, Peiyan Hu, Qi Meng, Yue Wang, Rongchan Zhu, Bingguang Chen, Zhiming Ma, Hao Ni, and Tie-Yan Liu. Deep latent regularity network for modeling stochastic partial differential equations. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(6):7740–7747, 2023. doi: 10.1609/aaai.v37i6.25938.

Ling Guo, Lei Li, and Jingtong Zhang. Deep operator learning for efficient sampling from invariant measures of stochastic differential equations. *arXiv preprint arXiv:2609.11376*, 2026. URL https://arxiv.org/abs/2609.11376.

Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4 (2):251–257, 1991. doi: 10.1016/0893-6080(91)90009-T.

Jian Huang, Yuling Jiao, Lican Kang, Xu Liao, Jin Liu, and Yanyan Liu. Schrödinger–Föllmer sampler. *IEEE Transactions on Information Theory*, 71(2):1283–1299, 2025. doi: 10.1109/TIT.2024.3522494.

Martin Hutzenthaler, Arnulf Jentzen, and Peter E. Kloeden. Strong and weak divergence in finite time of Euler's method for stochastic differential equations with non-globally Lipschitz continuous coefficients. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 467(2130):1563–1576, 2011. doi: 10.1098/rspa.2010.0348. URL https://arxiv.org/abs/0905.0273.

Martin Hutzenthaler, Arnulf Jentzen, and Peter E. Kloeden. Strong convergence of an explicit numerical method for SDEs with nonglobally Lipschitz continuous coefficients. *The Annals of Applied Probability*, 22(4):1611–1641, 2012. doi: 10.1214/11-AAP803. URL https://arxiv.org/abs/1010.3756.

Pengzhan Jin, Shuai Meng, and Lu Lu. MIONet: Learning multiple-input operators via tensor product. *SIAM Journal on Scientific Computing*, 44(6):A3490–A3514, 2022. doi: 10.1137/22M1477751.

Patrick Kidger, James Morrill, James Foster, and Terry Lyons. Neural controlled differential equations for irregular time series. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6696–6707, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/4a5876b450b45371f6cfe5047ac8cd45-Abstract.html.

Patrick Kidger, James Foster, Xuechen Li, and Terry J. Lyons. Neural SDEs as infinite-dimensional GANs. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 5453–5463, 2021. URL https://proceedings.mlr.press/v139/kidger21b.html.

Peter E. Kloeden and Eckhard Platen. *Numerical Solution of Stochastic Differential Equations*. Springer, 1992. doi: 10.1007/978-3-662-12616-5.

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

Anastasis Kratsios, Giulia Livieri, and Philipp Schmocker. NeuralChaos: Optimal adapted approximation of square integrable predictable processes. *arXiv preprint arXiv:2607.14361*, 2026. URL https://arxiv.org/abs/2607.14361.

Samuel Lanthaler, Siddhartha Mishra, and George E. Karniadakis. Error estimates for DeepONets: a deep learning framework in infinite dimensions. *Transactions of Mathematics and Its Applications*, 6(1):tnac001, 2022. doi: 10.1093/imatrm/tnac001.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Wonjae Lee, Taeyoung Kim, and Hyungbin Park. Fourier neural operators for non-Markovian processes: Approximation theorems and experiments. *arXiv preprint arXiv:2507.17887*, 2025. URL https://arxiv.org/abs/2507.17887.

Xuechen Li, Ting-Kam Leonard Wong, Ricky T. Q. Chen, and David Duvenaud. Scalable gradients for stochastic differential equations. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 3870–3882, 2020. URL https://proceedings.mlr.press/v108/li20i.html.

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

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.02747.

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*, 3:218–229, 2021. doi: 10.1038/s42256-021-00302-5.

Xuerong Mao. *Stochastic Differential Equations and Applications*. Horwood Publishing, 2 edition, 2008. doi: 10.1533/9780857099402.

Enea Monzio Compagnoni, Anna Scampicchio, Luca Biggio, Antonio Orvieto, Thomas Hofmann, and Josef Teichmann. On the effectiveness of randomized signatures as reservoir for learning rough dynamics. In *International Joint Conference on Neural Networks*, 2023. doi: 10.1109/IJCNN54540.2023.10191624.

James Morrill, Cristopher Salvi, Patrick Kidger, and James Foster. Neural rough differential equations for long time series. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 7829–7838, 2021. URL https://proceedings.mlr.press/v139/morrill21b.html.

Ariel Neufeld and Philipp Schmocker. Solving stochastic partial differential equations using neural networks in the Wiener chaos expansion. *arXiv preprint arXiv:2411.03384*, 2024. URL https://arxiv.org/abs/2411.03384v2. Version 2, revised January 2026.

Vivek Oommen, Khemraj Shukla, Saaketh Desai, Rémi Dingreville, and George Em Karniadakis. Rethinking materials simulations: Blending direct numerical simulations with neural operators. *npj Computational Materials*, 10:145, 2024. doi: 10.1038/s41524-024-01319-1.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), 2018. doi: 10.1609/aaai.v32i1.11671. URL https://arxiv.org/abs/1709.07871.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986. doi: 10.1038/323533a0.

Cristopher Salvi, Maud Lemercier, and Andris Gerasimovics. Neural stochastic PDEs: Resolution-invariant learning of continuous spatiotemporal dynamics. In *Advances in Neural Information Processing Systems*, volume 35, pp. 1333–1344, 2022. doi: 10.52202/068431-0098.

Dai Shi, Lequan Lin, Andi Han, Luke Thompson, José Miguel Hernández-Lobato, Zhiyong Wang, and Junbin Gao. Expanding the chaos: Neural operator for stochastic (partial) differential equations. *arXiv preprint arXiv:2601.01021*, 2026a. URL https://arxiv.org/abs/2601.01021v2.

Dai Shi, Luke Thompson, Andi Han, Peiyan Hu, Junbin Gao, and José Miguel Hernández-Lobato. Wiener chaos expansion based neural operator for singular stochastic partial differential equations. *arXiv preprint arXiv:2603.08219*, 2026b. URL https://arxiv.org/abs/2603.08219.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=PxTIG12RRHS.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 32211–32252. PMLR, 2023. URL https://proceedings.mlr.press/v202/song23a.html.

Stefan Tappe. The Yamada–Watanabe theorem for mild solutions to stochastic partial differential equations. *Electronic Communications in Probability*, 18(24):1–13, 2013. doi: 10.1214/ECP.v18-2392.

Akshay Thakur and Souvik Chakraborty. MD-NOMAD: Mixture density nonlinear manifold decoder for emulating stochastic differential equations and uncertainty propagation. *arXiv preprint arXiv:2404.15731*, 2024. URL https://arxiv.org/abs/2404.15731v2. Version 2, revised May 2025.

Simone Venturi and Tiernan Casey. SVD perspectives for augmenting DeepONet flexibility and interpretability. *Computer Methods in Applied Mechanics and Engineering*, 403:115718, 2023. doi: 10.1016/j.cma.2022.115718.

Roman Vershynin. *High-Dimensional Probability: An Introduction with Applications in Data Science*. Cambridge University Press, 2018. doi: 10.1017/9781108231596.

Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed DeepONets. *Science Advances*, 7(40):eabi8605, 2021. doi: 10.1126/sciadv.abi8605.

Toshio Yamada and Shinzo Watanabe. On the uniqueness of solutions of stochastic differential equations. *Journal of Mathematics of Kyoto University*, 11(1):155–167, 1971. doi: 10.1215/kjm/1250523691.

Haoming Yang, Ali Hasan, Yuting Ng, and Vahid Tarokh. Neural McKean–Vlasov processes: Distributional dependence in diffusion processes. In *Proceedings of the 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pp. 262–270, 2024. URL https://proceedings.mlr.press/v238/yang24a.html.

Yuantu Zhu, Zheyan Li, Dai Shi, Luke Thompson, Oliver Nash, Jose Miguel Lara Rangel, Siran Li, Bingguang Chen, Rongchan Zhu, Qi Meng, and Hao Ni. SPDEBench: An extensive benchmark for learning stochastic PDEs. *arXiv preprint arXiv:2505.18511v3*, 2026. URL https://arxiv.org/abs/2505.18511v3. First version 2025; revised May 2026.

THEORETICAL FOUNDATIONS

A SDE SOLUTION MAPS AND REFERENCE ERRORS

A.1 SUFFICIENT CONDITIONS FOR THE SDE SOLUTION MAP

Fix $T < \infty$ and a Brownian motion on a filtered probability space satisfying the usual conditions. Assume X_0 is measurable at time zero, independent of the Brownian motion, and $\mathbb{E}\|X_0\|^2 < \infty$. For Borel coefficients, assume

$$\|a(t,x) - a(t,y)\| + \|b(t,x) - b(t,y)\|_F \le L\|x-y\|, \qquad \|a(t,x)\| + \|b(t,x)\|_F \le L(1+\|x\|),$$

uniformly on $[0,T]$. These conditions give a pathwise unique strong solution (Mao, 2008). For the cubic drifts, it is enough to assume that a, b are locally bounded and locally Lipschitz in x, uniformly on $[0,T]$, and for some $C_T < \infty$,

$$2\langle x, a(t,x)\rangle + \|b(t,x)\|_F^2 \le C_T(1+\|x\|^2).$$

Local existence and uniqueness follow by truncating the coefficients. For the exit time τ_R from the ball of radius R, Itô's formula and Gronwall's inequality give

$$\mathbb{E}\|X_{t\wedge\tau_R}\|^2 \le (\mathbb{E}\|X_0\|^2 + C_T T)e^{C_T T}, \qquad 0 \le t \le T.$$

Consequently $\Pr(\tau_R \le T) \le (\mathbb{E}\|X_0\|^2 + C_T T)e^{C_T T}/R^2$, so the solution is nonexplosive. Fatou's lemma gives finite second moments, and local pathwise uniqueness extends to $[0,T]$. The scalar double-well and coupled-2D drifts have dissipative cubic terms, with the remaining drift at most linear and bounded diffusion, so they satisfy this condition. The linear, periodic, and cyclic examples satisfy the global conditions. The finite-particle learning target is treated in Section A.2.

Under these conditions, the solution has the measurable nonanticipative representation in Section 3.1. Appendix B gives the corresponding assumptions for mild SPDE solutions, following Tappe (2013).

A.2 CONDITIONAL PREDICTION IN PARTICLE SYSTEMS

Let $X^{n,N}$ denote the Euler state of particle n in the N-particle system specified in Appendix F. It depends on the other particles' noises through the empirical measure. The particle count N is distinct from the number of trajectories used for fitting or evaluation. For a new system independent of the training data, hold the trained model fixed. A causal predictor observing $\mathcal{G}_k^n = \sigma(X_0^n, B_{t_0:t_k}^n)$ has squared-loss population optimum $m_k^n = \mathbb{E}[X_k^{n,N} \mid \mathcal{G}_k^n]$, assuming finite second moments. For any square-integrable $\mathcal{G}_k^n$-measurable predictor g_k,

$$\mathbb{E}\|X_k^{n,N} - g_k\|^2 = \mathbb{E}\|X_k^{n,N} - m_k^n\|^2 + \mathbb{E}\|m_k^n - g_k\|^2. \tag{1}$$

The first term is the error due to unobserved particle noises; the second is the predictor's excess error over the conditional mean. Independent particle systems provide new realizations of this unobserved influence. Since the conditional mean generally has a different distribution from the particle state, path error alone does not determine terminal-distribution accuracy. Generating the training trajectories still requires simulation of the full interacting system.

A.3 REFERENCE ERROR

For ordinary SDEs, define the discrete time-space norm $\|Y\|_{2,h}^2 = [(K+1)d]^{-1}\sum_{k=0}^K \mathbb{E}\|Y_{t_k}\|_2^2$. When the predictor, numerical solution labels, and continuous solution are coupled on one probability space, the triangle inequality gives

$$\|\widehat{X} - X\|_{2,h} \le \|\widehat{X} - X^h\|_{2,h} + \|X^h - X\|_{2,h}. \tag{2}$$

The first term measures prediction error against fixed-grid labels; the second is the numerical solver's error. Our approximation result concerns the first term and does not imply L^2 convergence of the labels as $h \downarrow 0$. In particular, ordinary explicit Euler for the double-well benchmark has known mean-square divergence under refinement (Hutzenthaler et al., 2011), despite having finite moments on each fixed grid. Claims about convergence to the continuous solution require a suitable convergent solver, such as tamed Euler under its assumptions (Hutzenthaler et al., 2012). Finite refinement comparisons are numerical diagnostics rather than convergence guarantees.

A.4 A TERMINAL-DISTRIBUTION EXAMPLE VIA DOOB'S h-TRANSFORM

A prescribed terminal distribution connects solution-map learning with generative sampling. The classical Schrödinger–Föllmer construction provides such an SDE (Huang et al., 2025). This construction underlies the data-driven sampling experiment in Section 4.2.

From a target density to an SDE. Let q_s be the density of $N(0, sI_d)$, and let p_{data} be a positive probability density with finite second moment. For $0 \le t < 1$, define

$$f(y) = \frac{p_{\mathrm{data}}(y)}{q_1(y)}, \qquad H(t,x) = \int_{\mathbb{R}^d} q_{1-t}(y-x) f(y)\, dy, \qquad H(1,x) = f(x). \tag{3}$$

We use H for the Doob function to distinguish it from the numerical step size h. Assume H is finite, positive, and sufficiently smooth to satisfy the backward heat equation, and that $\beta(t,x) = \nabla_x \log H(t,x)$ extends to a globally Lipschitz drift with linear growth, uniformly on $[0,1]$. These are sufficient conditions for the strong solution of

$$dX_t = \beta(t, X_t)\, dt + dB_t, \qquad X_0 = 0, \qquad \mathcal{L}(X_1) = p_{\mathrm{data}}(x)\, dx. \tag{4}$$

For a Brownian reference path Z starting at zero, weight its law by $f(Z_1)$. This weight has expectation one, and $H(t, Z_t) = \mathbb{E}[f(Z_1) \mid Z_{[0,t]}]$ is its density process under the reference law. The Doob transform gives drift $\nabla \log H$, while its terminal density is $q_1 f = p_{\mathrm{data}}$. Thus the endpoint law is prescribed at a finite time; an individual path does not have to visit a particular data point.

Using samples of the target. The drift also has an expression involving expectations over data samples. For $Y \sim p_{\mathrm{data}}$ and $t < 1$, let $w_t(x,Y) = q_{1-t}(Y-x)/q_1(Y)$. Under differentiation beneath the integral,

$$H(t,x) = \mathbb{E}[w_t(x,Y)], \qquad \beta(t,x) = \frac{\mathbb{E}[(Y-x) w_t(x,Y)]}{(1-t)\mathbb{E}[w_t(x,Y)]}. \tag{5}$$

Sample averages therefore allow drift estimation without evaluating p_{data}. Using finitely many observations in this formula gives a bridge to the empirical measure, whose drift is generally singular as $t \uparrow 1$. Gaussian smoothing of the empirical target gives a regular density; its discrepancy from the population distribution remains a statistical error. The exact endpoint identity in (4) concerns the population construction.

An explicit multimodal example. Consider the two-dimensional target

$$p_{\mathrm{data}}(x) = \frac{1}{4}\sum_{j=1}^{4} \mathcal{N}(x; m_j, s^2 I_2), \qquad \{m_j\}_{j=1}^4 = \{(-2,-2),(-2,2),(2,-2),(2,2)\}, \quad s = \tfrac{1}{2}, \tag{6}$$

where $\mathcal{N}(x; m, \Sigma)$ denotes a Gaussian density. Set $v_t = 1 - t + ts^2$. Evaluating the Gaussian integrals in (3) gives

$$\omega_j(t,x) = \frac{\exp\{(m_j^\top x - t\|m_j\|^2/2)/v_t\}}{\sum_{\ell=1}^4 \exp\{(m_\ell^\top x - t\|m_\ell\|^2/2)/v_t\}}, \qquad \beta(t,x) = \frac{(s^2-1)x + \sum_{j=1}^4 \omega_j(t,x) m_j}{v_t}. \tag{7}$$

Since $v_t \ge s^2 > 0$ and the means are bounded, this drift satisfies the Lipschitz and growth conditions above. The one-time marginals are

$$\mathcal{L}(X_t) = \frac{1}{4}\sum_{j=1}^{4} N(tm_j, tv_t I_2), \qquad 0 < t \le 1. \tag{8}$$

Indeed, conditional on $X_1 = Y$, the reference path is a Brownian bridge, whose value at time t has mean tY and covariance $t(1-t)I_2$. Integrating over each Gaussian component gives (8). This example supplies an explicit drift and known marginal distributions against which a numerical sampler and a learned predictor can be checked. The experiment estimates the mixture parameters from sampled data; Appendix H.2 gives the drift for the fitted mixture.

Learning the noise-to-data map. Paired trajectories for (4) are obtained by integrating the SDE and retaining its driving Brownian increments. Training the same recurrent operator on these pairs approximates $B \mapsto X_t$; evaluation at $t = 1$ then maps fresh noise to approximate data samples. For a fixed fitted predictor $\widehat{X}_1$ and its numerical reference X_1^h, the coupling inequality and triangle inequality give

$$W_2\Big(\mathcal{L}(\widehat{X}_1), p_{\mathrm{data}}\Big) \le \Big(\mathbb{E}\|\widehat{X}_1 - X_1^h\|^2\Big)^{1/2} + W_2\big(\mathcal{L}(X_1^h), p_{\mathrm{data}}\big)\,. \tag{9}$$

Here W_2 is the quadratic Wasserstein distance, and the expectation is over fresh Brownian paths shared by predictor and reference. The first term is terminal prediction error; the second includes time discretization and any error in constructing the drift from data. A fixed trained model targets one distribution. Learning across different targets would require conditioning on their descriptions or data, or retraining.

The point mass δ_0 in (4) is a known initial law; randomness enters through B. To prescribe a nondegenerate prior, such as $N(0, I_d)$, as well as the data endpoint, one can instead solve the two-marginal Schrödinger bridge problem (De Bortoli et al., 2021). A single terminal reweighting generally changes a random initial law, so retaining an arbitrary prior requires this additional constraint.

B Direct noise-to-solution learning for SPDEs

Under the assumptions below, the initial field and driving noise determine the mild solution of the SPDE. We learn this map directly from spatial noise observations. The formulation below specifies the observations and field reconstruction used in the heat and nonlinear reaction experiments of Appendix L.

B.1 A solution operator on function spaces

Let $H = L^2(D)$ for a bounded spatial domain D, and let K be a separable Hilbert space. Consider

$$dU_t = [AU_t + f(U_t)]\,dt + g(U_t)\,dW_t^Q, \qquad U_0 \in H, \tag{10}$$

where A is a spatial differential operator, such as the Dirichlet Laplacian, generating a strongly continuous contraction semigroup $S(t)$, and W^Q is a K-valued Wiener process with trace-class covariance Q. A sufficient setting uses a square-integrable initial field independent of the noise, globally Lipschitz coefficients with linear growth, and the diffusion norm $\|(g(u) - g(v))Q^{1/2}\|_{\mathrm{HS}}$ in the Lipschitz condition. The equation is interpreted through its mild solution:

$$U_t = S(t)U_0 + \int_0^t S(t-s)f(U_s)\,ds + \int_0^t S(t-s)g(U_s)\,dW_s^Q. \tag{11}$$

Under the existence, pathwise uniqueness, and structural hypotheses of infinite-dimensional Yamada–Watanabe results (Tappe, 2013), this defines a measurable nonanticipative solution map, almost surely:

$$U = \mathcal{G}(U_0, W^Q), \qquad U_t = \mathcal{G}_t(U_0, W^Q|_{[0,t]}). \tag{12}$$

For a fixed initial field u_0, write $\mathcal{G}_{u_0}(W^Q)$: the noise is then the only varying input. The domain, boundary conditions, coefficients, and stochastic-integral convention are fixed. If any of these vary across tasks, they must be specified or supplied as additional inputs. Appendix C analyzes approximation and model selection from finite observations of this map.

The additive stochastic heat equation gives a direct example. With $K = H$, $A = \nu\Delta$, $f = 0$, and $g = \sigma I$, its solution is

$$\mathcal{G}_{u_0}(W^Q)_t = S(t)u_0 + \sigma \int_0^t S(t-s)\,dW_s^Q. \tag{13}$$

Here the stochastic convolution is linear in the driving noise. Adding a nonlinear drift gives a generally nonlinear map through (11). Our experiments compare heat with the reaction $f(u) = 2\sin u$; both use additive colored noise and satisfy the global Lipschitz assumptions above.

B.2 Encoding the driving noise directly

White noise is a generalized random field represented through its action on test functions. For the concrete $K = H$ setting, choose spatial cell test functions $\varphi_j = |C_j|^{-1/2}\mathbf{1}_{C_j}$ on disjoint cells C_j. Represent the noise by space–time integrals, with $h_k = t_k - t_{k-1}$:

$$\eta_{kj} = \langle W^Q_{t_k} - W^Q_{t_{k-1}}, \varphi_j\rangle_H, \qquad \mathbb{E}[\eta_{kj}\eta_{k\ell}] = h_k\langle Q\varphi_j, \varphi_\ell\rangle_H. \tag{14}$$

This representation records the forcing on local spatial cells. If W is instead cylindrical space–time white noise, the pairing is defined by the isonormal integral of $\mathbf{1}_{(t_{k-1},t_k]}\varphi_j$; the covariance is $h_k\delta_{j\ell}$. Identity covariance lies outside the trace-class framework in infinite dimensions, so this case requires its own SPDE well-posedness and regularity assumptions. The covariance determines how the noise observations must be scaled when the temporal or spatial resolution changes. For multiplicative noise, the model must also specify the stochastic-integral convention, such as the Itô interpretation.

B.3 A direct branch–trunk construction and training objective

Let $\mathcal{R}_\rho u_0$ record initial-field observations on a grid of scale ρ: cell averages, or point values for a regular version. Let $\eta_{1:k}$ contain only noise observations through t_k, together with their spatial and temporal coordinates. A spatial encoder and causal temporal encoder can supply the branch features, while the trunk is queried at (t_k, x):

$$\widehat{u}_\theta(t_k, x) = \sum_{r=1}^{p} b_{\theta,r}(\mathcal{R}_\rho u_0, \eta_{1:k})\tau_{\theta,r}(t_k, x). \tag{15}$$

For example, a spatial encoder can process each noise increment field, and a recurrent encoder can accumulate the resulting history. If u_0 is fixed, it need not be supplied as a varying input. Pointwise outputs assume an appropriate regular version of the solution; otherwise the decoder should predict cell averages or weak spatial observables. Initial and boundary conditions must be enforced by the parameterization or included in the training loss.

Paired training samples consist of independent noise realizations and their corresponding numerical SPDE solution fields. When pointwise labels are appropriate, a supervised objective is

$$\mathcal{L}(\theta) = \frac{1}{M}\sum_{m=1}^{M}\sum_{k,j} w_{kj}\left|\widehat{u}^{(m)}_\theta(t_k, x_j) - u^{(m)}_{\rho,h}(t_k, x_j)\right|^2, \tag{16}$$

where $w_{kj} = h_k|C_j|/(T|D|)$ are normalized space–time weights for a partition of D into cells C_j. The inputs and labels must use the same driving realization. A finite-difference, finite-element, or spectral SPDE solver supplies the training fields. Prediction then evaluates the trained operator on the initial field and new noise observations.

B.4 Error decomposition and evaluation

For grid values $v = (v_{kj})$, with $k = 1, \dots, K$ and $j = 1, \dots, J$, define the common field reconstruction

$$(\mathcal{I}_{\rho,h}v)(t, x) = \sum_{k=1}^{K}\sum_{j=1}^{J} v_{kj}\mathbf{1}_{(t_{k-1},t_k]}(t)\mathbf{1}_{C_j}(x), \qquad \|v\|^2_G = \sum_{k,j} w_{kj}|v_{kj}|^2.$$

The value at $t = 0$ does not affect the integral norm. This reconstruction assigns each query value to its preceding time cell for error measurement; the predictor remains causal at the query times t_k. For square-integrable random arrays,

$$\|\mathcal{I}_{\rho,h}v\|^2_{\mathcal{X}} = T|D|\,\mathbb{E}\|v\|^2_G, \qquad \mathcal{X} = L^2(\Omega; L^2(0, T; H)).$$

Let U_ρ be a continuous-time spatial/noise approximation, reconstructed in H by the same spatial cells. Write $U_{\rho,h} = \mathcal{I}_{\rho,h}u_{\rho,h}$ and $\widehat{U}_\theta = \mathcal{I}_{\rho,h}\widehat{u}_\theta$. Assume these fields and U belong to $\mathcal{X}$ and use a common noise coupling. The triangle inequality then gives

$$\|\widehat{U}_\theta - U\|_{\mathcal{X}} \le \underbrace{\|\widehat{U}_\theta - U_{\rho,h}\|_{\mathcal{X}}}_{\text{learning}} + \underbrace{\|U_{\rho,h} - U_\rho\|_{\mathcal{X}}}_{\text{time discretization}} + \underbrace{\|U_\rho - U\|_{\mathcal{X}}}_{\text{spatial/noise approximation}}. \tag{17}$$

The temporal term includes the specified time reconstruction. On the uniform experimental grid, $\|\cdot\|_G^2$ is the mean squared error over positive times and cells; the reported relative errors also divide by the reference-field norm. Appendix C separates the learning term into observation uncertainty, approximation, and finite-sample selection error. If labels resolve noise absent from the inputs, their conditional mean is the squared-loss target, leaving an irreducible error from unresolved noise. The coupled reference comparisons in Appendix L.3 quantify this effect and the sensitivity to temporal and spatial discretization.

As in Neural SPDEs (Salvi et al., 2022), prediction is conditioned on the initial field and the noise. Appendix L compares the branch–trunk model with a field MLP and a linear operator.

C Approximation and finite-sample model selection

A predictor based on finite noise observations has two possible information losses: noise unresolved by the grid and information discarded by the encoder. We first separate these losses from approximation error, then bound the error of selecting a trained model using independent validation paths. The reconstruction in Appendix B converts the grid risk into a field norm.

C.1 Observation and representation errors

Use the grid and normalized weights of Appendix B. Let $Y = (Y_{kj})$ be the array of numerical solution labels, with $\mathbb{E}\|Y\|_G^2 < \infty$, and let $Z_k = (\mathcal{R}_\rho U_0, \eta_{1:k})$ collect the available initial-field and noise observations. Write $\mathcal{O}_k = \sigma(Z_k)$. A causal predictor f has f_{kj} measurable with respect to $\mathcal{O}_k$ and population risk

$$R(f) = \mathbb{E}\|f - Y\|_G^2, \qquad m_{kj} = \mathbb{E}[Y_{kj} \mid \mathcal{O}_k], \qquad \sigma_{\mathrm{obs}}^2 = \sum_{k,j} w_{kj}\mathbb{E}|Y_{kj} - m_{kj}|^2. \tag{18}$$

Any fitted parameters are held fixed when taking these expectations over a fresh trajectory. For a fixed causal encoder, let V_k denote the finite-dimensional features actually passed to the prediction module, including any initial-state inputs supplied directly at that stage. For the implemented SPDE model, $V_k = h_k$: the initial field has already entered the spatial encoder. Set $\mathcal{V}_k = \sigma(V_k) \subseteq \mathcal{O}_k$. Define $m_{kj}^E = \mathbb{E}[Y_{kj} \mid \mathcal{V}_k]$.

Proposition C.1 (Risk decomposition)**.** *For any square-integrable predictor $f_{kj}(V_k)$,*

$$R(f) = \sigma_{\mathrm{obs}}^2 + \underbrace{\sum_{k,j} w_{kj}\mathbb{E}|m_{kj} - m_{kj}^E|^2}_{\sigma_E^2} + \sum_{k,j} w_{kj}\mathbb{E}|m_{kj}^E - f_{kj}(V_k)|^2. \tag{19}$$

For a general causal predictor, the corresponding identity is $R(f) = \sigma_{\mathrm{obs}}^2 + \mathbb{E}\|f - m\|_G^2$.

Proof. The residual $Y_{kj} - m_{kj}$ is orthogonal in L^2 to every $\mathcal{O}_k$-measurable square-integrable variable. By the tower property, $m_{kj}^E = \mathbb{E}[m_{kj} \mid \mathcal{V}_k]$, so $m_{kj} - m_{kj}^E$ is orthogonal to every $\mathcal{V}_k$-measurable square-integrable variable. Expand $Y_{kj} - f_{kj} = (Y_{kj} - m_{kj}) + (m_{kj} - m_{kj}^E) + (m_{kj}^E - f_{kj})$; the three cross terms vanish. Sum with weights w_{kj}. □

The first term measures missing noise or initial-field information; the second measures information lost by the encoder. If the labels are a deterministic function of the supplied observations, $\sigma_{\mathrm{obs}} = 0$. Finer references can have positive σ_{obs} because of unresolved noise, as in the exact-time heat comparison.

C.2 Approximation on a fixed grid

Proposition C.2 (Approximation from encoded features)**.** *Fix the encoder above, with $V_k \in \mathbb{R}^r$, and let $q_{kj} \in \mathbb{R}^s$ be the deterministic trunk input at query (k, j). Assume these $M = KJ$ positive-time query inputs are pairwise distinct. Let $\mathcal{H}_E$ consist of predictors*

$$f_{kj} = p^{-1/2} B(V_k)^\top \tau(q_{kj}),$$

where $B : \mathbb{R}^r \to \mathbb{R}^p$ and $\tau : \mathbb{R}^s \to \mathbb{R}^p$ are fully connected networks with one or more tanh hidden layers and affine output layers. All weights and biases are unrestricted, and all finite hidden widths and latent dimensions p are allowed. Then

$$\inf_{f \in \mathcal{H}_E} R(f) = \sigma^2_{\text{obs}} + \sigma^2_E. \tag{20}$$

In particular, arbitrarily small risk against numerical solution labels is possible within this family when both information terms vanish.

Proof. Lower bound. Every such predictor is $\mathcal{V}_k$-measurable at query (k, j). It is square-integrable because a finite tanh network with an affine output is bounded, and the trunk is evaluated at finitely many fixed inputs. Conditional-expectation orthogonality gives

$$R(f) = \mathbb{E}\|Y - m^E\|^2_G + \sum_{k,j} w_{kj}\mathbb{E}|m^E_{kj} - f_{kj}|^2 \geq \mathbb{E}\|Y - m^E\|^2_G = \sigma^2_{\text{obs}} + \sigma^2_E,$$

where the last equality follows from Proposition C.1 with $f = m^E$.

Shared branch approximation. Fix $\varepsilon > 0$ and write μ_k for the law of V_k. The Doob–Dynkin lemma gives a Borel function $g_{kj} : \mathbb{R}^r \to \mathbb{R}$ such that $m^E_{kj} = g_{kj}(V_k)$ almost surely. Conditional Jensen's inequality implies $\int |g_{kj}|^2 \, d\mu_k \leq \mathbb{E}|Y_{kj}|^2 < \infty$. Since μ_k is a probability measure and tanh is bounded and nonconstant, the $L^2(\mu_k)$ density theorem (Hornik, 1991) (Theorem 1) supplies a one-hidden-layer scalar network b_{kj} with

$$\mathbb{E}|b_{kj}(V_k) - m^E_{kj}|^2 < \varepsilon.$$

Concatenate the hidden units of these finitely many networks and use separate affine output coordinates. This produces one shared branch $B = (b_{kj}) \in \mathbb{R}^M$. Each coordinate need only approximate its target under the corresponding μ_k; the laws of V_k need not agree.

Exact trunk interpolation and assembly. For $M \geq 2$, choose $a \in \mathbb{R}^s$ outside the finite union of hyperplanes $\{a : a^\top(q - q') = 0\}$ over distinct query inputs. Relabel the queries as $q_1, \ldots, q_M$ so that $\alpha_1 < \cdots < \alpha_M$, where $\alpha_\ell = a^\top q_\ell$, and order the branch coordinates in the same way. Choose $c_i \in (\alpha_i, \alpha_{i+1})$ and define

$$h_\beta(q) = \big(1, \tanh(\beta(a^\top q - c_1)), \ldots, \tanh(\beta(a^\top q - c_{M-1}))\big)^\top, \qquad (H_\beta)_{\ell,:} = h_\beta(q_\ell)^\top.$$

As $\beta \to \infty$, the first column of H_β is one and its $(i+1)$st column tends to -1 for $\ell \leq i$ and $+1$ for $\ell > i$. Subtracting successive rows of this limit matrix shows that its determinant is 2^{M-1}. Thus H_β is invertible for some finite β. Set

$$\tau(q) = \sqrt{M}\, H_\beta^{-\top} h_\beta(q).$$

This is a tanh trunk with an affine output layer; the constant coordinate of h_β is supplied by the output bias. Since $h_\beta(q_\ell) = H_\beta^\top e_\ell$, it satisfies $\tau(q_\ell) = \sqrt{M}\, e_\ell$ exactly. For $M = 1$, take the constant trunk $\tau \equiv 1$. Using $p = M$ therefore gives $f_{kj} = b_{kj}(V_k)$, including the $p^{-1/2}$ normalization in the model definition. Consequently,

$$\sum_{k,j} w_{kj}\mathbb{E}|m^E_{kj} - f_{kj}|^2 < \varepsilon \sum_{k,j} w_{kj} = \varepsilon.$$

Combining this construction with the lower bound and letting $\varepsilon \downarrow 0$ proves the equality. If every label is $\mathcal{V}_k$-measurable, then $m^E = Y$ almost surely and the infimum is zero. □

The approximation is in L^2 under the encoded-feature distribution; continuity of the infinite-dimensional solution map is not required. The networks used in the experiments have fixed encoder width 96 and latent dimension 32. The next result concerns the finite set of predictors actually obtained by training, rather than the full network class.

C.3 FINITE-SAMPLE MODEL SELECTION BY VALIDATION

Let $\mathcal{F} = \{f_1, \dots, f_N\}$ be a finite set of fitted causal predictors, such as checkpoints along a training trajectory. Condition on all fitting data, pilot decisions, and training randomness, so that $\mathcal{F}$ is fixed independently of the n validation trajectories. Within each trajectory the spatial and temporal coordinates may be dependent. Distinct particles from one interacting system do not satisfy the independence assumption; an application to particle data must instead use independent systems as validation units (Appendix I), unless an additional dependence analysis is provided. Define the single-trajectory loss and empirical risk by

$$\ell_f(Z, Y) = \|f(Z) - Y\|_G^2, \qquad \widehat{R}_n(f) = \frac{1}{n}\sum_{i=1}^{n} \ell_f(Z^{(i)}, Y^{(i)}). \tag{21}$$

Assume these n trajectories are independent and identically distributed as the fresh evaluation trajectory, and, for some $\Lambda > 0$,

$$\sup_{f\in\mathcal{F}} \mathbb{E}\exp\left(\frac{|\ell_f - R(f)|}{\Lambda}\right) \le 2. \tag{22}$$

This exponential-moment condition allows unbounded Gaussian-driven labels.

Proposition C.3 (Validation selection bound)**.** *If $\widehat{f} \in \mathcal{F}$ satisfies $\widehat{R}_n(\widehat{f}) \le \min_{f\in\mathcal{F}} \widehat{R}_n(f) + \zeta$ for $\zeta \ge 0$ and $0 < \delta < 1$, then, with probability at least $1 - \delta$ over the validation sample,*

$$R(\widehat{f}) \le \sigma_{\mathrm{obs}}^2 + \underbrace{\min_{f\in\mathcal{F}} \mathbb{E}\|f - m\|_G^2}_{A_{\mathcal{F}}} + 2\kappa_{n,N,\delta} + \zeta, \tag{23}$$

$$\kappa_{n,N,\delta} = \Lambda\left(\sqrt{\frac{8\log(2N/\delta)}{n}} + \frac{\log(2N/\delta)}{n}\right).$$

Proof. For $D_f = \ell_f - R(f)$, (22) implies $\mathbb{E}|D_f|^q \le 2q!\Lambda^q$ for integers $q \ge 2$. Since $\mathbb{E}D_f = 0$, expansion of the moment-generating function gives

$$\log \mathbb{E}e^{\lambda D_f} \le \frac{2\lambda^2\Lambda^2}{1 - |\lambda|\Lambda}, \qquad |\lambda|\Lambda < 1.$$

Independence across trajectories and a Chernoff bound yield

$$\Pr\Big(|\widehat{R}_n(f) - R(f)| > \Lambda\{\sqrt{8s/n} + s/n\}\Big) \le 2e^{-s}.$$

This is the standard Bernstein argument for exponential-tailed losses (Vershynin, 2018). Set $s = \log(2N/\delta)$ and take a union bound over $\mathcal{F}$. On the resulting event,

$$R(\widehat{f}) \le \widehat{R}_n(\widehat{f}) + \kappa \le \min_{f\in\mathcal{F}} \widehat{R}_n(f) + \zeta + \kappa \le \min_{f\in\mathcal{F}} R(f) + \zeta + 2\kappa.$$

Apply Proposition C.1 to the last minimum. □

The reference scheme (43) meets the required tail condition for a finite collection of the fitted tanh/GRU predictors. Indeed, $(I - 0.02hD_{\Delta x})^{-1}$ is a contraction in the discrete spatial L^2 norm, the initial fields are bounded, and $|f(u)| \le 2$ for both benchmark drifts. On a fixed grid, the Gaussian increments are a linear image of a finite standard Gaussian vector ξ. Iterating the scheme gives $\|Y\|_G \le c_0 + c_1\|\xi\|_2$. Each fitted predictor is bounded at the positive-time queries because its hidden activations are bounded and its weights are finite. Thus $\ell_f \le c_2(1 + \|\xi\|_2^2)$, uniformly over the finite candidate set. Gaussian exponential moments imply (22) for some finite Λ, conditional on fitting. Its value can depend on the grid, equation, and fitted weights.

For the reported SPDE training schedule, each model selects from 40 checkpoints using 128 independent validation paths; validation scores do not change the update schedule or weights. Scalar train-only output scaling multiplies all candidate empirical validation risks by the same positive constant, so the selected checkpoint also minimizes the empirical validation risk in the original state

units. The bound controls selection error as a function of the validation sample size n, with dependence $\sqrt{\log(N/\delta)/n} + \log(N/\delta)/n$. The term $A_{\mathcal{F}}$ is not controlled: it depends on the candidates actually produced by training. Proposition C.2 does not ensure that $\mathcal{F}$ contains a near-optimal network. Together, these results establish approximation and finite-sample model selection, not training consistency, training-sample complexity, or optimization convergence.

Corollary C.4 (Reconstructed-field error)**.** *Under the preceding assumptions and the reconstruction of Appendix B, on the same event,*

$$\|\mathcal{I}_{\rho,h}\widehat{f} - U\|_{\mathcal{X}} \le \sqrt{T|D|\big(\sigma_{\mathrm{obs}}^2 + A_{\mathcal{F}} + 2\kappa_{n,N,\delta} + \zeta\big)} + \varepsilon_{\mathrm{time}} + \varepsilon_{\mathrm{space}}, \tag{24}$$

where $\varepsilon_{\mathrm{time}} = \|U_{\rho,h} - U_\rho\|_{\mathcal{X}}$ and $\varepsilon_{\mathrm{space}} = \|U_\rho - U\|_{\mathcal{X}}$ under the common noise coupling. The norm on the left averages a fresh trajectory with the selected predictor fixed.

Proof. The reconstruction identity gives $\|\mathcal{I}_{\rho,h}\widehat{f} - U_{\rho,h}\|_{\mathcal{X}} = \sqrt{T|D|R(\widehat{f})}$. Insert (23) into (17). □

Corollary C.4 is an error decomposition and a conditional bound. We do not establish that all its terms vanish jointly as the grid is refined and encoder capacity and sample sizes increase; it is therefore not a convergence theorem for the continuous SPDE solver.

D TIME-DEPENDENT OUTPUT NETWORK AND COMPRESSION

D.1 COEFFICIENT REPRESENTATION AND ERROR BOUND

For the independently trained models, the last branch layer is affine: $b_q = A_q\phi(h_k, \widetilde{X}_0) + a_q$, with $A_q \in \mathbb{R}^{p\times w}$, $a_q \in \mathbb{R}^p$, and a shared tanh feature vector $\phi \in [-1,1]^w$, where $w = 48$. With $\widetilde{\phi}_k = [\phi_k^\top, 1]^\top$, the standardized prediction is

$$\widehat{y}_{k,q} = C_{k,q}\widetilde{\phi}_k + c_q, \qquad C_{k,q} = p^{-1/2}\tau(t_k/T)^\top[A_q\ a_q]. \tag{25}$$

Here $C_k \in \mathbb{R}^{d\times(w+1)}$ stacks the row vectors $C_{k,q}$. Concatenating its rows at each noninitial query gives a time-by-coefficient matrix $\mathsf{C} \in \mathbb{R}^{K\times d(w+1)}$ with at most p linearly independent columns. This limits the number of independent coefficient directions over time. The predicted paths also vary with the history features ϕ_k.

We freeze the encoder and hidden features, subtract the mean row of C, and truncate the centered matrix's SVD. The mean becomes a static affine output layer; the retained factors are absorbed into smaller branch and trunk output layers. The resulting network accepts new input paths without further training on solution labels. The factorization and the bounds below use the specified query grid.

Let $C_k^{(r)}$ be the compressed coefficients and let S be the unchanged output scale from Section 3.3. Since $\|\widetilde{\phi}_k\|^2 \le w+1$, the perturbation satisfies

$$\mathrm{RMSE}(\widehat{X}^{(r)}, \widehat{X}) \le \left[\frac{w+1}{(K+1)d}\sum_{k=1}^{K}\|S(C_k - C_k^{(r)})\|_F^2\right]^{1/2} =: \varepsilon_r. \tag{26}$$

The initial predictions coincide. In exact arithmetic, this inequality holds for any collection of input paths evaluated on that grid. By the triangle inequality, compression increases RMSE against any common reference by at most ε_r, with the original model's reference error remaining as the baseline. Section D.2 gives the factorization and proof.

D.2 COEFFICIENT FACTORIZATION AND PERTURBATION

Set $\mathsf{M} = p^{-1/2}[A_1\ a_1\ \cdots\ A_d\ a_d]$ and let row k of $\mathsf{T} \in \mathbb{R}^{K\times p}$ be $\tau(t_k/T)^\top$. Then $\mathsf{C} = \mathsf{T}\mathsf{M}$ and $\operatorname{rank}\mathsf{C} \le p$. Write the centered SVD as

$$\mathsf{C} - \mathbf{1}\bar{c}^\top = U\Sigma V^\top, \qquad \bar{c}^\top = \bar{\tau}^\top\mathsf{M}, \tag{27}$$

where $\bar{\tau}$ is the mean trunk vector over $k = 1,\dots,K$. If V_r contains the first r right singular vectors, define

$$\mathsf{C}^{(r)} = \mathbf{1}\bar{c}^\top + (\mathsf{T} - \mathbf{1}\bar{\tau}^\top)\mathsf{M}V_rV_r^\top. \tag{28}$$

For coordinate q, let $V_{r,q}$ be the corresponding $(w+1)\times r$ row block of V_r. The compact dynamic branch output is $V_{r,q}^\top\widetilde{\phi}_k$, and the compact trunk output is $V_r^\top\mathsf{M}^\top[\tau(t_k/T) - \bar{\tau}]$. Their inner product reproduces the projected dynamic contribution in (28). Reshaping $\bar{c}$ and retaining the original output bias gives the static affine output layer. These expressions can be implemented by changing the final linear layers, retaining the encoder, branch feature layer, and (for $r > 0$) trunk hidden layers. At $r = 0$, the trunk is removed and the static output layer acts on the unchanged history features. The converted network can be evaluated at other times, but the stated error bound is tied to the factorization grid.

For each noninitial time and any path,

$$\begin{aligned}\|\widehat{X}_k^{(r)} - \widehat{X}_k\|_2^2 &= \|S(C_k^{(r)} - C_k)\widetilde{\phi}_k\|_2^2 \\ &\le \|S(C_k^{(r)} - C_k)\|_F^2\|\widetilde{\phi}_k\|_2^2 \le (w+1)\|S(C_k^{(r)} - C_k)\|_F^2.\end{aligned} \tag{29}$$

Sum over times and paths and divide by $n_{\mathrm{test}}(K+1)d$ to obtain (26). Let $s_{\max} = \|S\|_2$, and let σ_j be the singular values of the centered coefficient matrix. The SVD residual gives the looser bound

$$\varepsilon_r \le s_{\max}\left[\frac{w+1}{(K+1)d}\sum_{j>r}\sigma_j^2\right]^{1/2}. \tag{30}$$

The comparison with any common reference follows from the triangle inequality. The statements use exact arithmetic. Coefficients and residual bounds are computed in float64 before the converted weights are stored in float32; implementation checks are recorded in the supplement.

D.3 A RESTRICTED ANALYTIC CONTROL FOR TIME MULTIPLICATION

Time multiplication can help when the encoder rescales its input history. The following example makes this dependence explicit. Let ΔW_j be independent $N(0, h)$ increments and $W_k = \sum_{j=1}^{k} \Delta W_j$. The query index k is uniform on $1, \dots, K$ independently of the increments. If a frozen encoder gives $v_k = W_k/k$, time multiplication kv_k recovers the target exactly. For an additive-linear predictor $av_k + bk + c$, conditional zero means give $b = c = 0$ at optimum (for $K > 1$). Writing $H_K = \sum_{k=1}^{K} 1/k$,

$$\mathbb{E}[v_k W_k] = h, \quad \mathbb{E}[v_k^2] = hH_K/K, \quad a_* = K/H_K, \tag{31}$$

and the minimum mean squared error is

$$\mathbb{E}[W_k^2] - \frac{\mathbb{E}[v_k W_k]^2}{\mathbb{E}[v_k^2]} = h\left(\frac{K+1}{2} - \frac{K}{H_K}\right). \tag{32}$$

For $K = 100, h = 0.01$, the analytic RMSE is 0.558770 and the Monte Carlo check gives 0.559196. If the feature is the prefix sum itself, returning that feature recovers the target exactly. The calculation compares two prescribed representations under additive-linear and multiplicative output rules; the empirical models jointly learn their representations and nonlinear output networks. It does not establish a general expressivity or performance advantage of DeepONet over nonlinear MLPs.

E FURTHER CONNECTIONS TO OPERATOR LEARNING AND STOCHASTIC MODELS

Learning from data and equations. Standard DeepONet and Fourier neural operators fit numerical input–output pairs (Lu et al., 2021; Li et al., 2021). Physics-informed DeepONets additionally impose differential-equation residuals and can learn parametric PDE solution operators with substantially reduced paired data (Wang et al., 2021). Our method learns from numerical trajectories, so its offline cost includes both simulation and network training.

Multiple inputs and stochastic representations. MIONet formalizes multiple-input operator learning through tensor-product representations (Jin et al., 2022). In our SDE construction, the branch combines the initial state with the encoded noise prefix; the SPDE encoder also receives the initial field. SDEONet uses a truncated chaos representation and analyzes its approximation complexity (Eigel & Miranda, 2025). Neural approximations of SPDE chaos coefficients (Neufeld & Schmocker, 2024) and WCE-based operators (Shi et al., 2026a) similarly separate stochastic features from deterministic coefficient functions. Our recurrent encoder learns its features directly from observed increments or cumulative noise. MFNO offers a Fourier construction for nonperiodic paths (Lee et al., 2025), while NeuralChaos studies approximation of predictable processes (Kratsios et al., 2026).

Learning SPDE fields. Beck et al. approximate solution fields along given noise realizations with neural networks (Beck et al., 2020). Neural SPDEs parameterize reusable operators taking an initial field and forcing realization as inputs (Salvi et al., 2022). DLR-Net constructs regularity features from those inputs before applying a neural operator (Gong et al., 2023), and WCE-FiLM-NO uses modulation to represent the dependence between singular stochastic components and their smoother remainders (Shi et al., 2026b). SPDEBench brings data generation, noise approximation, and evaluation metrics into a common benchmarking framework (Zhu et al., 2026). Our SPDE model encodes physical-space noise histories recurrently. We compare its branch–trunk output with a field MLP using the same encoder architecture, and use a learned linear operator as the heat-equation baseline.

Pathwise prediction and distributional learning. Differentiable SDE solvers enable estimation of neural stochastic dynamics from observations (Li et al., 2020); adversarial neural SDEs match distributions of generated and observed paths (Kidger et al., 2021). MD-NOMAD uses mixture-density outputs to approximate conditional distributions of stochastic simulator responses (Thakur & Chakraborty, 2024). Recent amortized samplers map SDE coefficient functions to transports targeting invariant measures (Guo et al., 2026). Our supervised pairs preserve the coupling between a driving realization and its numerical solution. For particle approximations (Bossy & Talay, 1997) and neural McKean–Vlasov dynamics (Yang et al., 2024), the state also depends on the particle distribution. Observing one particle's noise therefore leaves part of its evolution unresolved; our squared-loss target is its conditional mean.

History encoders, conditioning, and compression. Neural CDEs encode interpolated paths through controlled hidden dynamics (Kidger et al., 2020). Neural RDEs use log-signatures over intervals to summarize longer histories (Morrill et al., 2021), and randomized signatures employ fixed random dynamics with fitted output layers (Monzio Compagnoni et al., 2023). FiLM and temporally conditioned operators motivate our output network comparisons (Perez et al., 2018; Oommen et al., 2024). SVD-DeepONet and flexDeepONet connect learned solution representations to low-rank structure (Venturi & Casey, 2023). Our fitted-weight compression concerns the time-dependent coefficients obtained after combining the final affine branch layer with the trunk. Its perturbation bound controls changes to that fitted predictor, complementing the encoding, approximation, and reconstruction error analysis of DeepONets (Lanthaler et al., 2022).

EXPERIMENTAL DETAILS AND ADDITIONAL RESULTS

F SDE BENCHMARK EQUATIONS

F.1 SCALAR AND FINITE-PARTICLE TASKS

The scalar benchmarks use the following equations:

$$\begin{aligned} \text{linear:} \quad & dX_t = -0.5X_t\,dt + dB_t, && X_0 = 0.25, \\ \text{double well:} \quad & dX_t = (X_t - X_t^3)\,dt + 0.5\,dB_t, && X_0 = 0.25, \\ \text{mean field:} \quad & dX_t = F_{\mu_t}(X_t)\,dt + dB_t, && X_0 = 0. \end{aligned} \tag{33}$$

Here F_μ is the cumulative distribution function and μ_t is the law appearing in the McKean–Vlasov equation. Its numerical reference is the Euler particle method (EMP) (Bossy & Talay, 1997),

$$X_{k+1}^{n,N} = X_k^{n,N} + hF_{\widehat{\mu}_k^N}(X_k^{n,N}) + \Delta B_k^n, \qquad \widehat{\mu}_k^N = \frac{1}{N}\sum_{j=1}^{N} \delta_{X_k^{j,N}}. \tag{34}$$

The repeated scalar benchmarks have 64 state points at spacing $h = 0.01$, hence $T = 0.63$. These settings are separate from the earlier illustrations in Appendix H. With additive diffusion the standard Milstein correction vanishes; on the same increments it coincides with Euler–Maruyama.

Empirical-measure approximation and computational cost. The law μ_t is unknown during simulation. The Euler particle method approximates it by the empirical measure $\widehat{\mu}_k^N$ of N interacting particles and applies Euler–Maruyama to the resulting finite-particle system. For this benchmark, the drift of particle n is

$$F_{\widehat{\mu}_k^N}(X_k^{n,N}) = \frac{1}{N}\sum_{j=1}^{N} \mathbf{1}\{X_k^{j,N} \le X_k^{n,N}\}. \tag{35}$$

At each step, all drifts are computed from the current particle positions; the particles are then advanced with independent increments $\Delta B_k^n \sim \mathcal{N}(0, h)$. Thus their paths remain coupled through the empirical measure despite the independence of their driving noises.

Computing all N empirical CDF values by direct pairwise comparison costs $O(N^2)$ per step. Our timed implementation instead sorts the positions and assigns their normalized ranks, giving the usual $O(N \log N)$ sorting cost per step and $O(KN \log N)$ over K steps. Equal positions take the upper rank in the CDF definition; in particular, all initial particles are zero and their initial drift is one. Ordinary scalar SDEs with constant-cost drift evaluation need only $O(KN)$ work for N paths. The additional sorting and ensemble-wide dependence make this MV-SDE reference more expensive: even when only a few particle trajectories are retained as labels, the whole reference ensemble must be evolved, and each new time step depends on the previous ensemble.

After training, the individual-particle predictor evaluates new input noise paths without recomputing an empirical CDF or simulating companion particles. Reference generation and training remain offline costs. Appendix K reports the measured costs and their amortization; the finite-particle targets used for timing and accuracy differ there, so the timings alone do not establish a speedup at matched accuracy.

F.2 COUPLED, PERIODIC, AND CYCLIC VECTOR DYNAMICS

Both studies of the coupled equation use $T = 1$, $h = 0.02$, and $X_0 = (0.25, -0.15)^\top$, with coefficients given in Section 4.5.

The periodic task has $T = 2$, $h = 0.02$, and independent initial states uniform on $[-0.5, 0.5]^2$:

$$\begin{aligned} dX_1 &= (-0.7X_1 + 0.3\tanh X_2)\,dt \\ &\quad + (0.35 + 0.15\sin \pi t + 0.05\tanh X_2)\,dB_1 + 0.08\,dB_2, \\ dX_2 &= (-0.5X_2 - 0.2\tanh X_1)\,dt \\ &\quad + (0.30 + 0.12\cos \pi t + 0.05\tanh X_1)\,dB_2 + 0.06\,dB_1. \end{aligned} \tag{36}$$

The cyclic 8D task (labeled "Lattice 8D" in the tables) uses eight coordinates with cyclic indices, $T = 2$, $h = 0.025$, and independent initial states uniform on $[-0.25, 0.25]^8$:

$$\begin{aligned} dX_i = &\{0.4(X_{i-1} - 2X_i + X_{i+1}) - 0.6X_i + 0.2 \tanh X_i\}\, dt \\ &+ (0.30 + 0.12 \sin \pi t + 0.05 \tanh X_i)\, dB_i + 0.05\, dB_{i-1}. \end{aligned} \tag{37}$$

Brownian coordinates are independent; the mixed diffusion terms couple their effects on the state. In both additional tasks, the spatial coefficient derivatives are bounded and the coefficients have linear growth. Random initial states are supplied explicitly to every output network. Their test distribution matches the training distribution.

G Training settings and model selection

G.1 Factorial controls and budget replication

The first controlled study crosses four tasks (linear, double well, finite-particle mean field, and coupled two-dimensional), two representations (positions or increments), two encoders (GRU or causal Transformer), and two output networks (MLP or branch–trunk), using five experiments and 400 epochs. Scalar tasks use 20 training paths plus 128 labeled validation paths; the coupled task uses 128 training and 128 validation paths. Mean-field validation and primary tests use independent particle systems. All eight coupled-task configurations are repeated at 1200 epochs with the same data and initializations. The 1200-epoch experiments reuse the 400-epoch test data. The comparison below uses independently generated data.

G.2 Output network comparison on independent data

This comparison uses three tasks, two inputs, three output networks, and five independent experiments. All use a width-48 GRU, 128 training paths, 128 validation paths, 512 test paths, and a maximum of 1200 full-batch Adam epochs. Normalization uses training paths only, excluding t_0. A separate pilot experiment selects a learning rate from $\{3 \times 10^{-4}, 10^{-3}\}$ for each task, input, and output network combination using 600 epochs. The pilot generates no test paths. The selected rates are fixed before the independent experiments. Validation checks occur every 25 epochs, and the best validation checkpoint is evaluated on test after training.

The tasks are the coupled 2D dynamics with newly generated data ($T = 1$, $h = 0.02$), a periodic 2D system ($T = 2$, $h = 0.02$), and a cyclic 8D system ($T = 2$, $h = 0.025$). The additional tasks have globally Lipschitz spatial coefficients and random initial states; their equations are given in Appendix F. Inputs are either increments alone or increments with cumulative Brownian values. Adding cumulative channels increases GRU input parameters by 288 in 2D and 1152 in 8D. Parameter matching therefore applies to output networks within each input representation. We train a separate model for each equation and draw test initial states from the training distribution.

G.3 Diagnostics and statistical units

We report mean RMSE and sample standard deviation over five experiments, together with paired comparisons of prediction error. Shared data across conditions make the contrasts dependent. With five pairs, a two-sided exact sign-flip test has minimum attainable $p = 0.0625$, so we emphasize effect sizes and variability across experiments. A validation-selected causal ridge baseline and shared-noise 8-fold/16-fold reference refinements provide additional diagnostics.

We designed the coefficient interventions and compression study after inspecting the independent coupled-2D results; these analyses are exploratory. All 30 trained branch–trunk models are included. Validation-selected compression chooses the smallest number of retained singular components in $\{0, 1, 2, 4, 8, 16\}$ whose validation RMSE increases by at most 5%, otherwise retaining the original model. A second rule chooses the smallest component count with $\varepsilon_r \leq 10^{-3}$, without additional validation or test labels. Test errors are evaluated only after the compressed model has been selected.

G.4 Architecture and implementation

Models are implemented in PyTorch, with training-only standardization and a shared output scale for comparisons within each experiment. Full preprocessing and environment details are provided in the supplementary material.

The GRU has one layer and width 48. The branch hidden layer maps the concatenated state and initial condition to 48 tanh features, followed by a linear $d \times 32$ output. The trunk has widths $1, 48, 48, 32$, with tanh between linear layers. The MLP output network has two tanh hidden layers. The FiLM feature network has one tanh hidden layer and a linear feature projection; its condition network has one width-48 tanh layer and outputs the feature scales and shifts. Another tanh follows modulation, then a linear output layer.

Table 2: Total parameters in the independent output network comparison. Original and periodic 2D have identical counts. MLP hidden widths are (79,67) in 2D and (124,91) in 8D; FiLM feature widths are (47,48) and (75,82), with time-network hidden width 48.

Dimension	Input	MLP	Branch–trunk	FiLM
2D	ΔB	17,236	17,234	17,231
2D	$\Delta B + B$	17,524	17,522	17,519
8D	ΔB	27,799	27,800	27,799
8D	$\Delta B + B$	28,951	28,952	28,951

Selection and data independence Full-batch Adam uses no weight decay or learning-rate schedule in the independent comparison. Training, validation, and test paths use disjoint random-number streams. Within each experiment, the output networks share data and encoder initialization. The initial state is imposed exactly by the prediction interface. The supplement records the software environment and implementation checks.

H Path prediction, sampling, and benchmark settings

H.1 OU path plots and prediction-error distribution

The OU path study uses 20 training trajectories and 800 test paths. The reported per-path MSE is roughly 10^{-5} to 5×10^{-4}. The available record does not fully specify the coefficients, normalization fit, or noise coupling for this illustration. Figure 5 compares training and test paths; Figure 6 shows the error histogram.

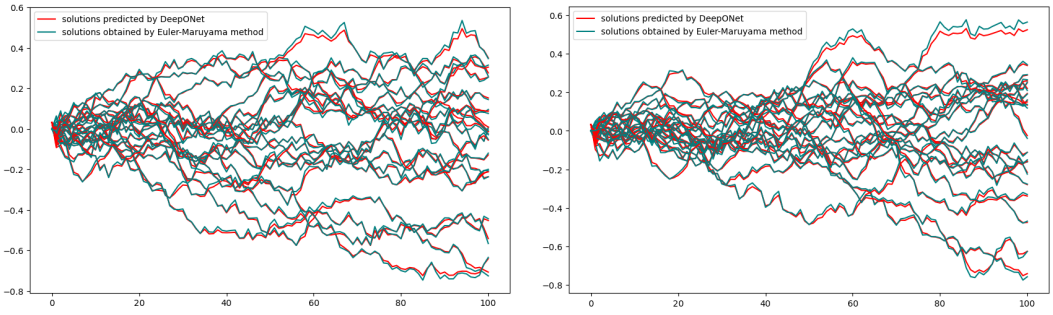


Figure 5: OU path prediction. Left: training paths; right: test paths. The horizontal axis is sample index and the vertical axis is state. Euler–Maruyama supplies the reference.

Table 3: Cross-step-size results from the initial OU experiment. Mean and standard deviation are reported across 800 paths. The metric is MSE after normalization to $[0, 1]$, unlike the raw-unit errors used elsewhere.

Step-size scale	Mean	Standard deviation
h	1.335e-03	1.313e-03
$h/10$	7.831e-03	8.471e-03
$h/100$	4.604e-02	2.743e-02
$h/1000$	6.725e-02	2.479e-02
$10h$	9.709e-03	1.133e-02
$100h$	3.090e-01	8.988e-02
$1000h$	2.402e-01	1.311e-01

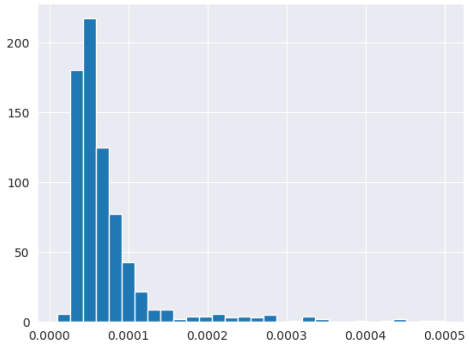


Figure 6: Initial OU prediction-error histogram over 800 paths. Horizontal axis: reported per-path MSE; vertical axis: count.

Table 3 reports errors after normalizing predictions and references to $[0, 1]$. The record does not fully specify whether the same fitted transform was used across scales, how horizons or sensors changed, or how Brownian paths were coupled. These omissions limit what can be concluded about generalization across time steps.

H.2 Learning a sampling SDE from four-mode data

This experiment follows the sequence *data points* $\rightarrow$ *fitted density* $\rightarrow$ *paired SDE trajectories* $\rightarrow$ *learned solution map* $\rightarrow$ *new samples*. The target is the four-component Gaussian mixture in (6). Each experiment begins with a new set of 10,000 construction samples. Its fitted density, Brownian paths, model initialization, and evaluation data are independent of the other experiments.

Constructing the SDE from data. We fit Gaussian mixtures with $J \in \{2, 3, 4, 5, 6\}$ components and one shared full covariance matrix. For each J, expectation maximization uses three initializations; we retain the highest likelihood and select J by the Bayesian information criterion $-2\ell + (3J + 2)\log(10{,}000)$. A $10^{-6}I_2$ covariance regularizer ensures positive definiteness. The component weights, means, and covariance are estimated from the construction samples. The known parameters of the synthetic target are used to draw data, and are not provided to the fit or SDE simulator.

Write the fitted density as $\widehat{p}(x) = \sum_{j=1}^{J} \widehat{\pi}_j \mathcal{N}(x; \widehat{m}_j, \widehat{\Sigma})$ and let $V_t = (1-t)I_2 + t\widehat{\Sigma}$. Its Doob drift is

$$\widehat{\omega}_j(t,x) = \frac{\widehat{\pi}_j \exp\{x^\top V_t^{-1}\widehat{m}_j - \frac{1}{2}t\widehat{m}_j^\top V_t^{-1}\widehat{m}_j\}}{\sum_{\ell=1}^{J} \widehat{\pi}_\ell \exp\{x^\top V_t^{-1}\widehat{m}_\ell - \frac{1}{2}t\widehat{m}_\ell^\top V_t^{-1}\widehat{m}_\ell\}},$$
$$\widehat{\beta}(t,x) = V_t^{-1}\left[(\widehat{\Sigma} - I_2)x + \sum_{j=1}^{J} \widehat{\omega}_j(t,x)\widehat{m}_j\right]. \tag{38}$$

This is the common-covariance version of (7). The exact SDE $dX_t = \widehat{\beta}(t, X_t)\,dt + dB_t$, $X_0 = 0$, has terminal law $\widehat{p}$. At an intermediate time, its law is $\sum_j \widehat{\pi}_j N(t\widehat{m}_j, t(1-t)I_2 + t^2\widehat{\Sigma})$. Thus the population target, fitted density, Euler solution, and network prediction can be compared separately.

Pathwise training and independent evaluation. Euler–Maruyama on $[0, 1]$ with $K = 128$ steps generates 4096 training and 1024 validation paths per experiment. The encoder receives aligned Brownian increments and cumulative values, together with the normalized query time. The GRU width is 64 and the branch–trunk dimension is 32; the model has 28,450 parameters. Input and output scales are fitted on training paths. The initial value is imposed exactly. Training minimizes standardized MSE at the positive grid times. An independent pilot compares initial learning rates 10^{-3} and 3×10^{-3} for 2000 updates and selects 3×10^{-3} by validation loss. Each final fit uses 6000 Adam updates with batches of 128, cosine decay to one tenth of the initial rate, and gradient clipping at 2. Validation every 100 updates selects the checkpoint. All five models are fixed before generating the test data. Training and prediction use MPS in single precision; reference simulation uses double precision.

Generated samples during training. Figure 7 follows a repeat of the first experiment (seed 8101) with the same data, initialization, minibatches, and training schedule. We record 68 intermediate checkpoints; the validation history and selected model weights match the archived experiment exactly. The panels compare eight training stages using the same 2500 Brownian paths and held-out target points. Each point is a predicted endpoint at $t = 1$, so the panels show changes during training rather than evolution along the SDE time axis. The generated points gradually form four clusters, with some fluctuations between training stages.



Figure 7: Generated samples during training of the Doob solution operator. Panels (a)–(h) show predicted endpoints from initialization to the final training iterate; panel (i) shows the target data. Panel titles report training loss: MSE in standardized coordinates over all 4096 training paths, 128 positive grid times, and both state coordinates. Orange points are generated samples; gray points are the fixed target data, shown in blue in panel (i). All panels use the same coordinate limits. The displayed SW_2 uses 2500 points and 256 projection directions. These are recorded network outputs, without interpolation.

We evaluate 2048 independent paired paths per experiment. Here, path RMSE uses the K positive grid times and both coordinates; terminal RMSE averages over the two coordinates. Multiplying terminal RMSE by $\sqrt{2}$ gives the empirical counterpart of the first term in (9). Relative path error is $100\big(\sum \|\widehat{X} - X^h\|_2^2 / \sum \|X^h\|_2^2\big)^{1/2}$, with both sums over these times and paths. Figure 8 shows paths near the median and 95th percentile of the per-path RMSE in the first experiment. Terminal mode agreement measures how often a prediction and its paired reference finish in the same quadrant; the terminal error percentile uses the root mean squared error over the two coordinates of each path.

Table 4: Prediction errors for the data-defined sampling SDE. Each experiment uses 2048 unseen paths; entries are means $\pm$ sample standard deviations over five experiments. The reference is the 128-step Euler solution driven by the same noise.

Metric	Value
Path RMSE	0.0503 ± 0.0064
Terminal RMSE	0.1473 ± 0.0207
Relative path error (%)	4.00 ± 0.53
Median per-path RMSE	0.0178 ± 0.0009
95th-percentile per-path RMSE	0.0657 ± 0.0102
95th-percentile terminal error	0.1137 ± 0.0182
Terminal mode agreement (%)	98.94 ± 0.24

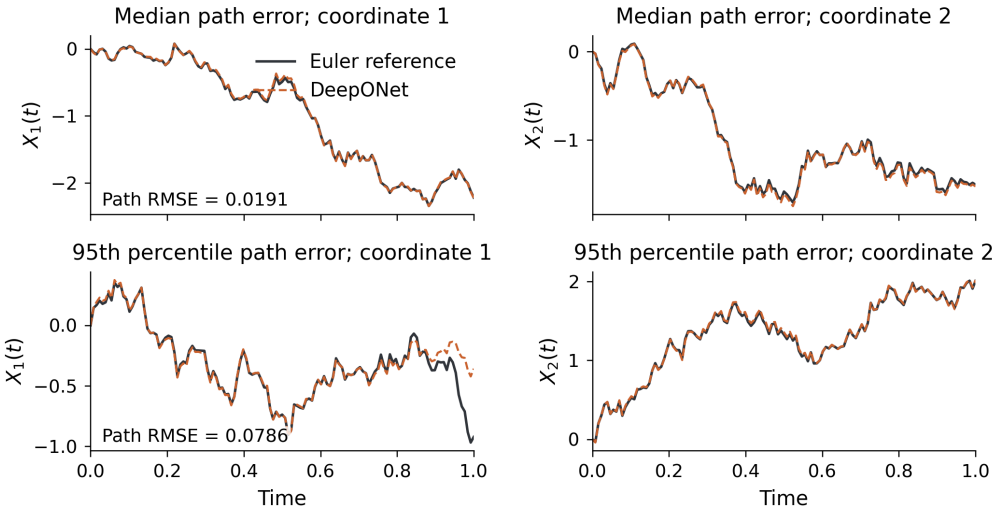


Figure 8: Held-out trajectories for the data-defined sampling SDE. The rows show paths nearest the median and 95th percentile of per-path RMSE in the first experiment; columns show the two coordinates. Each prediction and Euler reference use the same Brownian increments.

Generated distributions. For each trained model, 10,000 new Brownian paths produce a separate set of endpoints. We compare them with 10,000 independently drawn target points using 256 equally spaced projection directions on $[0, \pi)$. For direction θ_ℓ, let $a^\ell_{(i)}$ and $b^\ell_{(i)}$ be the sorted projections of the two equally sized sets. We report the finite-projection approximation to the sliced Wasserstein distance (Bonneel et al., 2015),

$$\mathrm{SW}_2(A, B) = \left[\frac{1}{256n} \sum_{\ell=1}^{256} \sum_{i=1}^{n} (a^\ell_{(i)} - b^\ell_{(i)})^2 \right]^{1/2}, \qquad n = 10{,}000. \tag{39}$$

Table 5 also includes samples drawn directly from $\widehat{p}$, Euler endpoints, and a second independent set of target points. The last comparison measures finite-sample variability. Quadrants separate the four modes. We report their frequencies and the within-quadrant standard deviation, computed by centering each quadrant separately and averaging the coordinate variances before taking the square root. Mode imbalance is the total variation between the four observed frequencies and those of the held-out target set. These statistics distinguish mode coverage from concentration within each mode (Figure 9).

Table 5: Endpoint-distribution comparison, with 10,000 samples per set in each of five experiments. SW_2 and mode total variation (TV) compare each set with a separate target-data set. Within-mode SD is the square root of the mean centered coordinate variance, averaged over quadrants. Entries are means $\pm$ sample standard deviations across experiments.

Sample source	SW_2	Mode TV	Within-mode SD
Independent target data	0.0723 ± 0.0305	0.0109 ± 0.0039	0.5022 ± 0.0026
Fitted mixture samples	0.0893 ± 0.0389	0.0113 ± 0.0051	0.5018 ± 0.0022
Euler endpoints	0.0715 ± 0.0196	0.0095 ± 0.0042	0.5051 ± 0.0030
DeepONet endpoints	0.0707 ± 0.0179	0.0096 ± 0.0042	0.5192 ± 0.0031

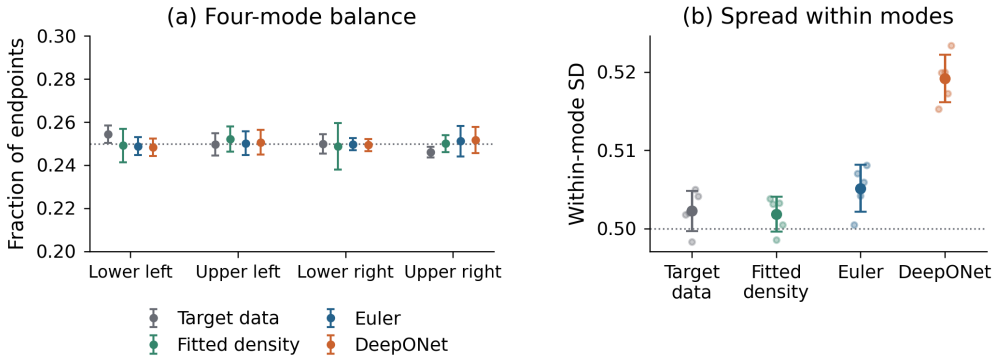


Figure 9: Mode frequencies and within-mode dispersion over five experiments. Error bars show sample standard deviations. Frequencies correspond to the lower-left, upper-left, lower-right, and upper-right quadrants. The horizontal reference at 0.5 in the dispersion panel is the population component standard deviation.

Error relative to refined solution paths. On 512 new Brownian paths per experiment, independent of the training and validation sets, we aggregate the same fine Brownian increments to grids with 128, 1024, and 2048 steps. These paths are used only for final evaluation after model selection. All comparisons use the original 129 observation times. The network sees only the 128-step noise observations. Table 6 compares its predictions and the coarse Euler solution with the 2048-step solution, and reports the remaining change from 1024 to 2048 steps. These measurements concern the SDE defined by $\widehat{p}$; the distribution comparisons above additionally include the discrepancy between $\widehat{p}$ and the population target.

Table 6: Shared-noise refinement on 512 additional paths per experiment. Errors use the original observation times; entries are means $\pm$ sample standard deviations across five experiments. The 2048-step path is a numerical reference.

Comparison	Path RMSE	Terminal RMSE
DeepONet vs. 128-step Euler	0.0536 ± 0.0096	0.1599 ± 0.0256
DeepONet vs. 2048-step Euler	0.0618 ± 0.0091	0.1800 ± 0.0246
128-step vs. 2048-step Euler	0.0372 ± 0.0118	0.1147 ± 0.0461
1024-step vs. 2048-step Euler	0.0042 ± 0.0024	0.0113 ± 0.0122

Results. BIC selects four components in every experiment. The generated endpoints recover all four modes, whose mean frequencies range from 24.83% to 25.17%. Their SW_2 distance to held-out data is 0.0707 ± 0.0179, close to 0.0715 ± 0.0196 for Euler endpoints; two independent target samples give 0.0723 ± 0.0305. The generated clusters are slightly broader, with within-mode SD 0.5192 ± 0.0031 compared with population component SD 0.5.

Path prediction has relative error $4.00 \pm 0.53\%$ against numerical solution labels, and paired terminal modes agree on $98.94 \pm 0.24\%$ of paths. Errors grow towards the terminal time, and occasional mode disagreements contribute to the larger terminal RMSE. Against the 2048-step reference, relative path error is $4.87 \pm 0.70\%$ (raw RMSE 0.0618 ± 0.0091). The remaining 1024-to-2048-step path discrepancy is 0.0042 ± 0.0024.

H.3 Burgers-type sampling and CPU timings

The McKean–Vlasov sampling experiment uses $h = 0.01$, $\sigma = 1$, and 10,000 particles; its terminal CDF is shown in Figure 2.

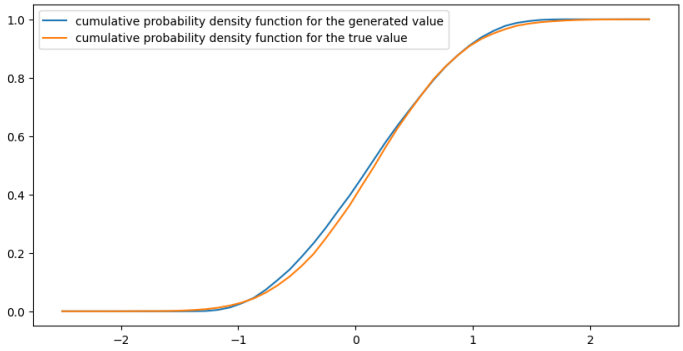


Figure 10: Terminal empirical CDFs for the McKean–Vlasov sampling experiment. The reference is the numerical interacting particle system.

The source reports $M = 31$ but refers to M both as a step count and as a number of sampled states, leaving the exact terminal horizon ambiguous. Only 20 trajectories are used for fitting; the reference simulation still generates the full interacting ensemble. Appendix F describes the empirical-measure Euler method and explains the cost of updating the particle ensemble and its CDF at every time step.

Table 7: Recorded CPU timings for the initial particle-sampling experiment, in seconds. N and M retain the source table's notation. B-time, O-time, and T-time are its reported Brownian-generation, computation, and training times; DeepONet training used 20 paths. Brownian-generation timings cover full paths for EMP and a single endpoint for DeepONet.

Method	N	M	B-time	O-time	T-time
EMP	100	31	5.6601e-03	2.4640e-02	–
DeepONet	100	31	6.2943e-05	4.0356e-02	1.1468e+02
EMP	1000	31	7.4580e-02	2.9986e-01	–
DeepONet	1000	31	3.5286e-05	4.0859e-01	1.1468e+02
EMP	10000	31	4.8686e-01	9.1371e+00	–
DeepONet	10000	31	3.7098e-04	6.8098e+00	1.1468e+02
EMP	100	51	1.1670e-02	4.3571e-02	–
DeepONet	100	51	2.5272e-05	5.2540e-01	1.9591e+02
EMP	1000	51	7.1276e-02	4.9766e-01	–
DeepONet	1000	51	3.6955e-05	4.8736e-01	1.9591e+02
EMP	10000	51	4.9503e-01	4.6647e+01	–
DeepONet	10000	51	4.7207e-05	3.9963e+00	1.9591e+02
EMP	100	101	1.1119e-02	8.2526e-02	–
DeepONet	100	101	2.8133e-05	1.1981e-01	4.5926e+02
EMP	1000	101	5.7222e-02	1.0523e+00	–
DeepONet	1000	101	3.2926e-04	3.9296e-01	4.5926e+02
EMP	10000	101	4.8471e-01	9.3683e+01	–
DeepONet	10000	101	4.6968e-05	3.5034e+00	4.5926e+02

Table 8: Initial coupled-system benchmark: coordinate RMSE, mean ± sample standard deviation across three experiments; 128 fitting and 128 test paths, 300 epochs, Brownian-position inputs. Every model uses a branch–trunk output network.

Encoder	Parameters	Coordinate 1	Coordinate 2
GRU, width 48	17,234	0.0161 ± 0.0032	0.0148 ± 0.0022
Transformer, width 36	17,270	0.0119 ± 0.0015	0.0110 ± 0.0002
Transformer, width 48	28,754	0.0088 ± 0.0009	0.0089 ± 0.0010

The timings in Table 7 were measured on an AMD EPYC 7B12 CPU. EMP's Brownian-generation time covers whole paths; the DeepONet timing record includes only a single Brownian endpoint, although causal prediction requires the full prefix. The recorded training costs are 114.68, 195.91, and 459.26 seconds for $M = 31, 51, 101$. Since the input-generation workloads differ, these timings are not directly comparable. Appendix K uses the same supplied paths for both methods.

H.4 Scalar benchmark settings

The scalar benchmark uses 20 training and 1000 test paths per experiment, with 64 state points at $h = 0.01$. Adam uses learning rate 10^{-3}. Models are trained for at most 600 full-batch epochs and selected by best training loss, with early-stopping patience $\max(80, 600/5) = 120$ epochs without a training-loss improvement exceeding 10^{-8}. This differs from the later validation-selected studies. The randomized-signature baseline is a fixed 64-dimensional controlled reservoir with 66 fitted coefficients. Mean-field training and test paths share one 1020-particle reference system in this benchmark.

For Table 14, reference-data generation is measured on 1020 trajectories with 64 state points, after one unmeasured warm-up, and averaged over five synchronized repetitions. For linear and double-well tasks this includes the held-out trajectories as well as the training set; for mean field the full interacting system supplies the selected training particles. Thus the reported data-generation cost includes the entire benchmark ensemble. The timed mean-field Euler solver evolves the 1000 supplied test-noise paths as a new interacting ensemble, while the accuracy column compares predictions with those paths' labels from the original 1020-particle system. An accuracy-matched cost comparison would require evaluation against one common finite-particle target.

H.5 Coupled-system settings and scaling

The coupled system in Section 4.5 uses $X_0 = (0.25, -0.15)^\top$, $T = 1$, and $h = 0.02$. The initial encoder comparison uses 128 training and 128 test paths, 300 epochs, and three experiments. All models use a branch–trunk output network. With Brownian-position inputs, the width-36 Transformer has lower mean coordinate errors than the GRU at a similar parameter count; the width-48 Transformer reduces them further with more parameters (Table 8). For this comparison, switching to increment inputs gives GRU coordinate errors of $(0.0114, 0.0122)$ and the width-48 Transformer has $(0.0161, 0.0191)$. Perturbing future tokens after $t = 0.5$ leaves prefix outputs unchanged in the causality check. The scaling experiment in Figure 4 uses five experiments, nested training subsets from a 1024-path pool, 256 validation paths, and 512 test paths. At fixed capacity (17,234 parameters), MSE falls from 5.942×10^{-5} at 64 paths to 4.104×10^{-5} at 256 and 4.016×10^{-5} at 1024. At 1024 paths, the 121,762-parameter model's MSE is approximately one third (1/2.89) of that of the fixed 17,234-parameter model. Since capacity and data co-vary in the joint schedule, its local fit is descriptive. Jointly increasing training paths and parameters from 64/7378 to 1024/121762 reduces MSE from 1.208×10^{-4} to 1.392×10^{-5}, a factor of 8.68. The fitted log–log slope is -0.711 ($R^2 = 0.947$).

The training-budget comparison fixes 256 training paths and 31,186 parameters. With a constant learning rate and validation-based checkpoint selection, MSE falls from 1.595×10^{-2} at 50 epochs to 2.281×10^{-5} at 1200 epochs. The latter is 22% below the corresponding final-iterate MSE; cosine decay reaches 3.095×10^{-5} (Figure 4). Appendix I compares the encoders again at 1200 epochs.

I FULL FACTORIAL CONTROLS AND INDEPENDENT-SYSTEM TESTS

I.1 ARCHITECTURE COMPARISONS AND TRAINING BUDGET

The factorial study uses five independent experiments, Adam with learning rate 10^{-3}, 400 full-batch epochs, validation checks every ten epochs, and no early stopping. All models enforce the initial state. GRU width is 48; the one-layer Transformer has width 36, four heads, and zero dropout. The output latent dimension is 32. Scalar test sets contain 1000 paths per experiment; the coupled-task test set contains 512 paths.

Table 9: Complete 400-epoch factorial study: RMSE $\times 10^3$, mean $\pm$ sample standard deviation over five experiments. Mean-field rows use independent-system testing.

Task	Input	Encoder	MLP	Branch–trunk
Linear	B	GRU	25.460 ± 6.670	16.392 ± 5.787
Linear	B	Transformer	25.668 ± 8.942	22.224 ± 7.835
Linear	ΔB	GRU	40.718 ± 13.560	33.013 ± 15.086
Linear	ΔB	Transformer	32.256 ± 11.859	18.017 ± 5.845
Double well	B	GRU	21.181 ± 12.765	17.599 ± 4.398
Double well	B	Transformer	21.963 ± 15.008	19.703 ± 13.070
Double well	ΔB	GRU	26.699 ± 15.919	21.273 ± 16.335
Double well	ΔB	Transformer	29.085 ± 11.374	22.002 ± 3.471
Mean field	B	GRU	29.811 ± 5.734	29.120 ± 8.427
Mean field	B	Transformer	30.512 ± 9.063	25.589 ± 9.021
Mean field	ΔB	GRU	38.834 ± 8.227	28.536 ± 6.496
Mean field	ΔB	Transformer	38.021 ± 3.930	26.500 ± 3.184
Coupled 2D	B	GRU	16.424 ± 2.272	11.199 ± 0.282
Coupled 2D	B	Transformer	12.310 ± 1.736	10.969 ± 1.014
Coupled 2D	ΔB	GRU	17.120 ± 2.254	12.142 ± 1.716
Coupled 2D	ΔB	Transformer	18.995 ± 0.987	13.593 ± 0.991

The output network parameter counts differ by at most two in this study. Between GRU and Transformer, total counts are only approximately matched: about 0.2% apart in 2D and 3.5% in scalar tasks. The output networks are therefore more closely matched in size than the two encoders (Table 9).

Table 10: Original 2D optimization-budget replication: RMSE $\times 10^3$, five experiments. Reusing data and initial weights isolates sensitivity to the training budget.

Input	Encoder	Output	400 epochs	1200 epochs
B	GRU	MLP	16.424 ± 2.272	8.056 ± 0.817
B	GRU	Branch–trunk	11.199 ± 0.282	6.364 ± 0.413
B	Transformer	MLP	12.310 ± 1.736	9.022 ± 0.996
B	Transformer	Branch–trunk	10.969 ± 1.014	8.137 ± 0.556
ΔB	GRU	MLP	17.120 ± 2.254	8.510 ± 1.484
ΔB	GRU	Branch–trunk	12.142 ± 1.716	6.873 ± 0.738
ΔB	Transformer	MLP	18.995 ± 0.987	11.628 ± 1.131
ΔB	Transformer	Branch–trunk	13.593 ± 0.991	9.799 ± 0.314

The longer-budget experiment repeats all eight coupled-task configurations with identical data and initial weights. The best validation loss through epoch 400 exactly matches that of the initial experiments. At 1200 epochs, GRU has lower RMSE than Transformer in every corresponding experiment under both inputs and output networks. Twenty of forty experiments achieve their best validation score at the final epoch.

I.2 INDEPENDENT PARTICLE SYSTEMS

For mean field, each experiment's training data use 20 particles from a single 1020-particle system. Validation uses 64 particles from each of two new systems; test uses 200 particles from each of five further independent systems. The remaining 1000 training-system particles are diagnostic only. Tests at 256 and 4096 particles use five additional systems each and the unchanged fitted model and normalization. The 128 validation particle paths are dependent within each system. For the independence argument in Proposition C.3, the validation units would be the two independent systems, each with loss averaged over its 64 selected particles: $n = 2$, not $n = 128$. Applying that bound to these system-level losses would also require its exponential-moment assumption. The SPDE application in Appendix C.3 instead uses 128 independent field trajectories.

Table 11: Mean-field diagnostics: configuration mean RMSE $\times 10^3$ over five experiments. Same-system rows are diagnostic only; model selection uses independent validation systems. B–T denotes branch–trunk.

Input	Encoder	Output	Same	Independent	$N = 256$	$N = 4096$
B	GRU	MLP	29.047	29.811	29.122	29.108
B	GRU	B–T	28.477	29.120	29.130	29.006
B	Tr.	MLP	29.153	30.512	28.355	28.778
B	Tr.	B–T	24.620	25.589	24.720	23.692
ΔB	GRU	MLP	38.709	38.834	37.885	37.923
ΔB	GRU	B–T	28.581	28.536	28.351	27.812
ΔB	Tr.	MLP	37.440	38.021	36.921	36.707
ΔB	Tr.	B–T	26.913	26.500	27.324	26.613

Table 11 compares mean errors for each configuration. The resampling diagnostic fixes each tagged particle's noise and resamples the remaining particles' noises, using eight tagged paths and 32 resamples per path at $N = 1020$. The resulting conditional RMSE estimate, approximately 0.00362, describes the first term of (1) after averaging over times. This finite-resampling estimate is below the observed model errors.

I.3 NUMERICAL REFERENCES AND LINEAR PREDICTION

Prediction error against Euler–Maruyama labels does not include their numerical error. Comparing these labels with 16-fold finer references driven by the same Brownian motion gives mean discrepancies of 0.003249, 0.003190, and 0.003503 for coupled 2D, periodic 2D, and cyclic 8D, respectively (equations in Appendix F).

For OU, comparison with an exact-time reference over five experiments gives path RMSE 0.001366 ± 0.000015 under a shared-Brownian coupling. Common standardized innovations give 0.001153 ± 0.000026 but change the exact-transition/Brownian-increment covariance; we therefore use the former comparison to measure strong reference error.

Table 12: Linear prediction and numerical-reference comparisons, RMSE $\times 10^3$, mean $\pm$ sample standard deviation over five experiments. Reference columns compare shared-noise paths at the stated time resolutions.

Task	Causal ridge	Coarse / 16×	8× / 16×
Original 2D	21.288 ± 0.855	3.249 ± 0.091	0.669 ± 0.007
Periodic 2D	88.858 ± 4.831	3.190 ± 0.072	0.374 ± 0.007
Lattice 8D	163.328 ± 1.352	3.503 ± 0.033	0.387 ± 0.003

The refinement study aggregates fine Brownian increments to compare coarse/16-fold and 8-fold/16-fold paths over five independent experiments, measuring sensitivity to numerical resolution.

To assess linear prediction from the observed history, a causal ridge baseline maps initial states and noise prefixes to each noninitial query, with one path-level penalty selected by validation from six

candidates. Mean RMSE is 0.021288, 0.088858, and 0.163328 on the same three tasks. Its parameter count grows with prefix dimension and, unlike the comparison in Appendix J, is not matched to the neural models.

Table 13: Output network comparison on independent data: path RMSE $\times 10^3$, mean $\pm$ sample standard deviation over five independent experiments. Bold marks the lowest mean in each row. B denotes cumulative Brownian values; all output networks share the same encoder initialization and paths within each condition.

Task	Input	MLP	Branch–trunk	FiLM
Original 2D	ΔB	7.586 ± 0.739	$\mathbf{6.510 \pm 0.617}$	6.892 ± 1.304
Original 2D	$\Delta B + B$	5.037 ± 0.725	$\mathbf{4.032 \pm 0.255}$	5.130 ± 1.120
Periodic 2D	ΔB	35.649 ± 1.123	$\mathbf{26.787 \pm 0.879}$	30.473 ± 1.655
Periodic 2D	$\Delta B + B$	$\mathbf{13.178 \pm 0.913}$	14.173 ± 0.975	18.122 ± 1.747
Lattice 8D	ΔB	35.475 ± 1.944	$\mathbf{23.782 \pm 1.262}$	32.716 ± 1.397
Lattice 8D	$\Delta B + B$	31.041 ± 2.060	$\mathbf{27.693 \pm 1.838}$	34.428 ± 0.848

J Ablation study

We vary the noise representation, encoder, and output network independently on four tasks. We test all combinations in five experiments with 400 epochs: two noise representations, two causal encoders, and two output networks per task. Scalar tasks use 20 training paths and 128 additional labeled validation paths; the coupled task uses 128 of each. These experiments consequently have a larger label budget than the 20-path training-loss-selected benchmark. Branch–trunk has lower mean RMSE in all 16 matched conditions and reduces error in 74/80 paired experiments. When all eight coupled-task configurations are trained for 1200 epochs, branch–trunk gains remain approximately 10%–21%, with 20/20 paired improvements, whereas the GRU now beats the Transformer under both inputs and output networks (Appendix I).

We then compare the three output networks on independently generated data for coupled 2D, periodic 2D, and cyclic 8D dynamics. We evaluate two input choices in five experiments, each using a width-48 GRU, 128 training paths, 128 validation paths, and 512 test paths. A separate pilot experiment selects a learning rate without test data. Checkpoints are then selected by validation during at most 1200 epochs. Periodic 2D and cyclic 8D use random initial states sampled from the same distributions in training and testing. Each equation is fitted separately. Adding cumulative channels increases encoder parameters; parameter matching applies to output networks within an input setting (Appendix G).

Branch–trunk has lower mean RMSE than MLP in five of six conditions and wins 25/30 paired comparisons; against FiLM the counts are six of six and 27/30. In 8D with increments, its mean error is 32.96% below MLP and 27.31% below FiLM. With cumulative inputs, the coupled-2D advantage over MLP remains 19.95%, even though the cumulative sum is supplied as an input. On periodic 2D with cumulative inputs, however, branch–trunk is 7.55% worse than MLP and loses four of five pairs. On coupled 2D with increments, branch–trunk beats FiLM in only two of five experiments, despite having the lower mean error.

Additional experimental results are provided in Appendix M.

K Computational cost and output network compression

We measure the cost of repeated prediction on the scalar tasks and the parameter savings from compressing the vector models.

Offline cost and repeated inference The cost benchmark evaluates 1000 common Brownian-increment paths per batch, excluding Brownian generation from both inference timings. MPS operations are synchronized; each experiment uses median times from 30 model evaluations and ten Euler evaluations. Table 14 reports means across the five experiments, together with path RMSE and the one-dimensional Wasserstein distance W_1 between empirical terminal distributions. Offline cost includes generation of the benchmark reference ensemble and model training.

Task	RMSE	$W_1(X_T)$	Offline	Inf.	EM	Break-even
Linear	0.0277 ± 0.0143	0.0247 ± 0.0114	36.1	0.0195	0.0135	∞
Double well	0.0136 ± 0.0129	0.0120 ± 0.0116	27.3	0.0139	0.0080	∞
Mean field	0.0346 ± 0.0180	0.0357 ± 0.0160	25.2	0.0122	0.0363	$1,043,297$

Table 14: Recurrent-DeepONet accuracy and cost for the scalar benchmark with five repeated experiments. Offline is measured in seconds; inference and EM are milliseconds per path. Times are averaged over the five experiments; RMSE and W_1 report mean $\pm$ sample standard deviation. "Break-even" is the number of paths computed from the reference-generation-plus-training cost in (40); ∞ means inference is not faster than Euler–Maruyama.

Let T_{data} and T_{train} be these offline times, and let T_{ref} and T_{inf} be times per 1000-path reference and model batch, all in seconds. Equating accumulated costs gives

$$N_{\mathrm{break}} = 1000 \frac{T_{\mathrm{data}} + T_{\mathrm{train}}}{T_{\mathrm{ref}} - T_{\mathrm{inf}}} \tag{40}$$

when $T_{\mathrm{ref}} > T_{\mathrm{inf}}$; otherwise there is no finite break-even count. Model inference is slower than Euler for the linear and double-well tasks. For the individual-particle mean-field surrogate it is faster, with a cost-only break-even near one million paths. This estimate amortizes the recorded data and training costs, excluding hyperparameter search and failed experiments. It compares an individual-particle predictor with a 1000-particle simulation, while accuracy uses 1020-particle labels. Because the timed solver and the accuracy labels use different particle counts, this crossover does not establish a speedup at matched accuracy.

Compression of the fitted output network We freeze the encoder and hidden features and compress the time-dependent coefficients of the branch–trunk output in Section 3.3. Choosing the compressed size by validation error reduces total parameters by 24.6%–46.2%. Test RMSE increases by at most 5% for 29 of the 30 models; the largest increase is 5.17%. A coefficient-bound rule uses no additional labels and reduces parameters by 19.9%–41.3%. This rule bounds the change from the uncompressed prediction on the trained grid. Appendix D gives the construction and proof; Appendix M reports the individual models.

L DIRECT SPDE EXPERIMENTS

L.1 AN EXPLICIT FIELD-LEARNING ALGORITHM

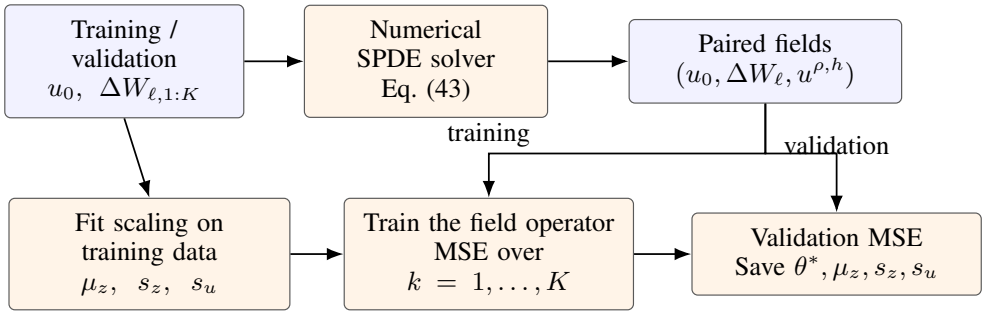
Algorithm 2 and Figure 11 describe the field predictor used in the experiments. A dense spatial encoder processes the observed noise and initial field at each time, and a unidirectional GRU accumulates their history. The trunk receives time and periodic spatial coordinates.

Write $\Delta W_{\ell,k} = W_\ell(t_k) - W_\ell(t_{k-1}) \in \mathbb{R}^{N_x}$ for the colored noise increment on interval k, set $v_0 = 0$ and $v_k = \Delta W_{\ell,k}$ for $k \geq 1$, and form $z_k = [v_k, \sum_{r=1}^{k} v_r]$ with an empty sum at $k = 0$. The coordinatewise mean μ_z and scale s_z are fitted to training inputs at positive times, with standard deviations clipped below at 10^{-8}. The output scale s_u is the RMS of the corresponding training reference fields. Division by s_z is coordinatewise. Let $\mathcal{S}_\theta$ denote a dense tanh spatial encoder and set

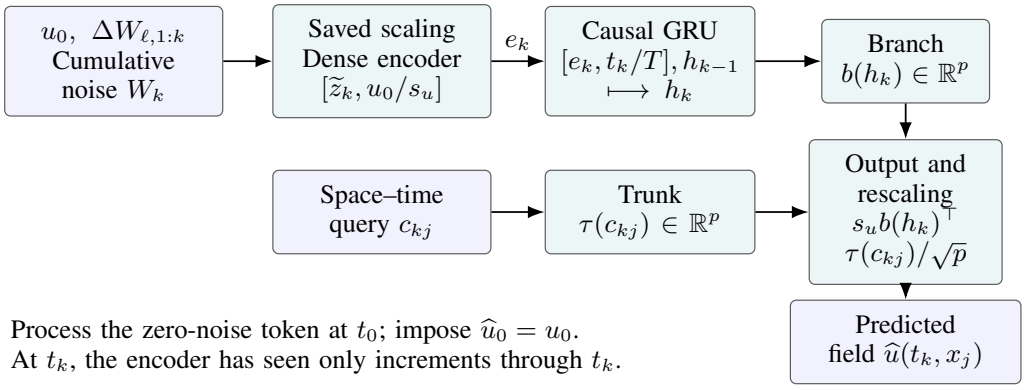
$$c_{kj} = \big[t_k/T,\ (\sin(2\pi m x_j))_{m=1}^{8},\ (\cos(2\pi m x_j))_{m=1}^{8}\big].$$

The experiments use $N_x = K = 64$, hidden width 96, and latent width $p = 32$. All times and coordinates in this algorithm belong to the specified grid.

(a) Offline fitting: paired noise and numerical solution fields



(b) Frozen prediction: a new initial field and observed noise prefix



Process the zero-noise token at t_0; impose $\widehat{u}_0 = u_0$.
At t_k, the encoder has seen only increments through t_k.

Figure 11: Direct SPDE learning and prediction. Top: numerical simulation supplies training and validation pairs. Bottom: the field predictor maps the initial field and observed noise prefix to values at the query points. All components are trained jointly; weights are fixed after validation selection and before testing.

Algorithm 2 Direct SPDE field learning and causal prediction

Require: Disjoint training and validation pairs $(u_0, \Delta W_{\ell,1:K}, u^{\rho,h}_{0:K})$ on the specified grid.

Fit (μ_z, s_z, s_u) on training data as defined above.

Forward procedure $F_\theta(u_0, \Delta W_{\ell,1:L})$**,** $1 \le L \le K$**:**

Form $z_{0:L}$ including the zero-noise token; set $\widetilde{z}_k = (z_k - \mu_z)/s_z$ and $\widetilde{u}_0 = u_0/s_u$.

Set $h_{-1} = 0$ and $\widehat{u}_0 = u_0$.

for $k = 0, \dots, L$ **do**

  $e_k \leftarrow \mathcal{S}_\theta([\widetilde{z}_k, \widetilde{u}_0])$; $h_k \leftarrow \mathrm{GRU}_\theta([e_k, t_k/T], h_{k-1})$.

  if $k > 0$ **then**

    $\widehat{u}_{k,j} \leftarrow s_u\, b_\theta(h_k)^\top \tau_\theta(c_{kj})/\sqrt{p}$ for $j = 1, \dots, N_x$.

  end if

end for

return $\widehat{u}_{0:L}$.

Offline fitting:

Select the learning rate on the independent pilot validation split; initialize θ.

for $s = 1, \dots, 4000$ **do**

  Sample a training minibatch $\mathcal{B}$ of 128 paths; compute $\widehat{u}^{(i)} = F_\theta(u_0^{(i)}, \Delta W^{(i)}_{\ell,1:K})$.

  $$\mathcal{L} \leftarrow \frac{1}{|\mathcal{B}| K N_x} \sum_{i \in \mathcal{B}} \sum_{k=1}^{K} \sum_{j=1}^{N_x} \left(\frac{\widehat{u}^{(i)}_{k,j} - u^{\rho,h,(i)}_{k,j}}{s_u} \right)^2 .$$

  Update all parameters by Adam, with gradient-norm clipping at 2 and the specified cosine schedule.

  Every 100 updates, evaluate validation MSE and retain the lowest-error checkpoint θ^*.

end for

Freeze and save θ^* and (μ_z, s_z, s_u) before test evaluation.

Prediction:

Load the saved weights and scales; return $F_{\theta^*}(u_0, \Delta W_{\ell,1:L})$ for the supplied new input.

The GRU processes the t_0 token, but the loss excludes the prescribed initial output. At any prefix $1 \le L \le K$, the recurrent state uses only the initial field and noise through t_L. The output network predicts the solution from this state; earlier predicted solution values are not fed back into the encoder.

The paired MLP control replaces only the output network architecture and is trained separately; the learned linear control has its own ridge fit.

L.2 A CONTROLLED TEST WITH RANDOM INITIAL FIELDS

We compare two equations that differ only in their reaction term. The heat equation has a linear solution operator; adding the reaction makes the response nonlinear. A learned linear operator tests whether nonlinear prediction is needed, while a field MLP with the same encoder architecture isolates the effect of the branch–trunk output network.

On the periodic domain $[0, 1)$ and time interval $[0, 1]$, we consider

$$du = [0.02\, u_{xx} + f(u)]\, dt + 0.5\, dW_\ell, \qquad f(u) = 0 \text{ or } f(u) = 2 \sin u. \tag{41}$$

For each realization, we independently sample the initial field

$$u_0(x) = a_0 + \sum_{m=1}^{2} [a_m \sin(2\pi m x) + b_m \cos(2\pi m x)], \qquad a_0, a_m, b_m \overset{\text{iid}}{\sim} \mathrm{Unif}[-0.25, 0.25]. \tag{42}$$

The training, validation, and test realizations are disjoint and follow the same distribution. Independent white-noise cell-average increments have variance $h/\Delta x$ and are convolved with a normalized periodic Gaussian kernel of smoothing width $\ell = 0.08$, producing spatially colored noise. The numerical SPDE solutions used as labels are computed with 64 cells, 64 time intervals, and the float64 semi-implicit scheme

$$u^{n+1} = (I - 0.02 h D_{\Delta x})^{-1} [u^n + h f(u^n) + 0.5 \Delta W^n_\ell], \tag{43}$$

where $D_{\Delta x}$ is the periodic centered-difference Laplacian. Initial fields are sampled at cell centers, and FFTs solve the periodic linear system in the reference scheme.

The spatial encoder reads the increment field, cumulative noise, and initial field, and a width-96 GRU encodes their causal history. The DeepONet has a 32-dimensional branch–trunk output network, with time and eight sine/cosine coordinate frequencies in the trunk, and 91,936 parameters. The direct field-MLP control has 91,934 parameters, using the same encoder architecture and a 97–63–86–64 output network. Paired models share encoder initialization, data, and minibatch order. Both use the causal encoder specified in Algorithm 2 and impose the supplied initial field at t_0. Encoder and output network parameters are optimized jointly within each model.

For the linear control, periodic spatial kernels are learned in

$$\widehat{u}^n = A_n * u_0 + \sum_{j=1}^{n} C_{n-j} * \Delta W_\ell^{j-1}. \tag{44}$$

Here $*$ denotes spatial convolution. A time-lag kernel is shared across observation times, and the initial-field response varies with time. The kernels are fit to training labels by ridge regression, with four validation-selected penalties in $\{10^{-8}, 10^{-6}, 10^{-4}, 10^{-2}\}$. FFTs implement the convolution with these data-fitted kernels. All scaling is fit on training data only.

We conduct five independent experiments, each using 512 training, 128 validation, and 512 test trajectories. A separate pilot experiment selects the learning rate from $\{10^{-3}, 3 \times 10^{-3}\}$ after 1,000 updates for each task/output network; all four selections choose 3×10^{-3}. Final neural models use 4,000 Adam updates, batch size 128, a cosine learning-rate schedule ending at one tenth of the initial rate, and gradient-norm clipping at 2. Checkpoints are selected by validation MSE every 100 updates. All 20 final neural models and 10 linear controls are frozen before any test split is generated. Neural training and primary inference use float32 on Apple MPS with CPU fallback disabled; reference generation and linear regression use CPU float64.

We report relative L^2 error over test trajectories, positive observation times, and spatial cells, averaged over five experiments with sample standard deviations. Path-error quantiles use all 2,560 test trajectories per equation.

Table 15 reports all three methods. DeepONet has lower prediction error against the numerical solution labels than the matched MLP in all five paired experiments on each equation. The reductions in mean error are 18.5% for heat and 6.9% for the nonlinear reaction under the fixed training budget. The linear operator reproduces the heat numerical solution almost exactly. Its error rises to 14.11% on the reaction equation, where both neural models are more accurate.

Table 15: Direct SPDE prediction against the 64-cell, 64-step numerical reference. The top block reports relative field L^2 in percent, as mean $\pm$ sample standard deviation over five independent experiments with 512 test paths each. The lower block describes the 2,560 individual DeepONet field trajectories per equation.

Method / statistic	Heat	Nonlinear reaction
DeepONet	1.627 ± 0.087	2.504 ± 0.045
Matched field MLP	1.996 ± 0.105	2.690 ± 0.085
Causal linear	$(2.165 \pm 0.039) \times 10^{-5}$	14.107 ± 0.689
DeepONet path median (%)	1.584	2.431
DeepONet path 95th percentile (%)	2.254	4.232
DeepONet path maximum (%)	5.407	6.707

Figures 12 and 13 show the prespecified examples from one experiment: the upper-median and worst paths, ranked by whole-field DeepONet error. The same paths are used for all methods. The largest visible deviations occur late in the worse examples. Table 15 reports the maximum path errors over all five experiments.

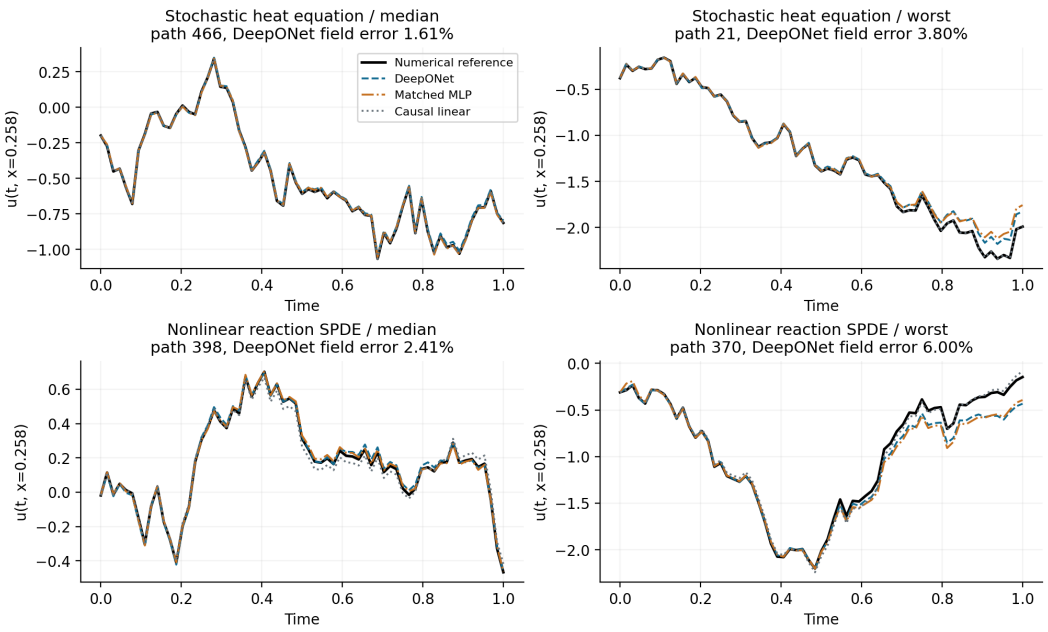


Figure 12: Held-out trajectories at $x = 16.5/64$. The left and right columns are the upper-median and worst whole-field DeepONet paths within the same prespecified experiment. Both neural output networks and the learned linear control are plotted on the same paths. The heat control is almost indistinguishable from the numerical reference.

The best validation checkpoints through update 2,000 have mean test errors of 2.714% (heat) and 3.665% (reaction) for DeepONet, decreasing to 1.627% and 2.504% through update 4,000. The corresponding MLP values decrease from 3.091% and 3.985% to 1.996% and 2.690%. Both comparisons use prefixes of the same 4,000-update cosine schedule. The errors are still decreasing late in training. The supplement includes the learning curves and both sets of selected checkpoints.

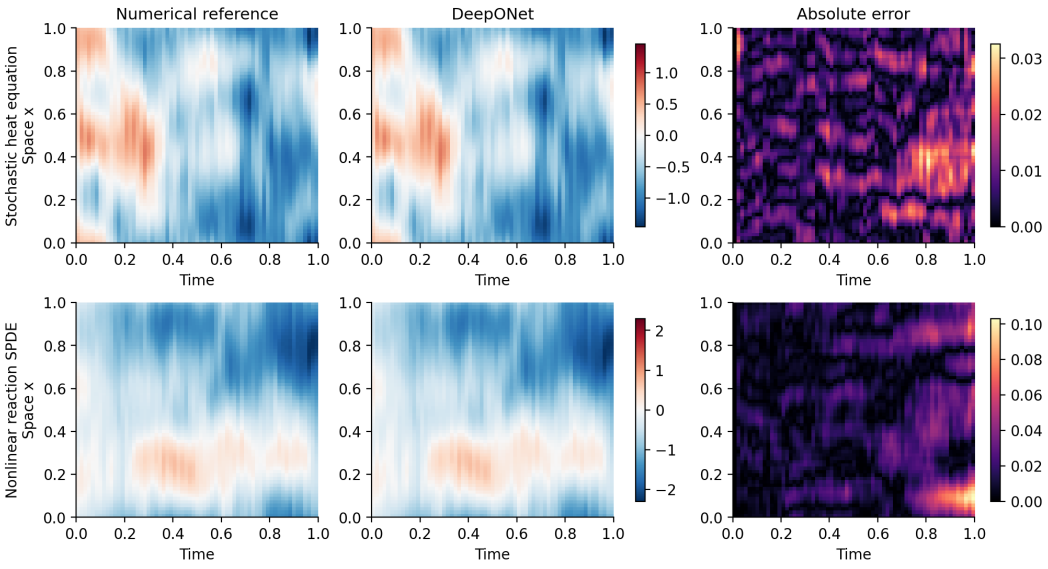


Figure 13: Full solution fields for the same upper-median examples used in Figure 12. Each row uses a common color scale for the reference and prediction; the absolute-error panel has its own scale. The supplement also includes the corresponding worst-example fields.

L.3 ACCURACY UNDER REFERENCE REFINEMENT

We generate refined references on new noise realizations, coupling all resolutions through the same underlying noise. The time study generates 64-times finer Brownian increments and aggregates them to 1, 16, 32, and 64-times finer time grids, with 64 paths per equation and experiment. The spatial study averages common finest white-noise cells before smoothing on 64, 128, and 256 cells, with 512 time steps and 32 paths per equation. The joint study uses 32 paths per equation and experiment at resolutions $(N_x, N_t) = (64, 64), (128, 1024), (256, 2048), (512, 4096)$; fine fields are averaged to 64 cells at the 65 common observation times. These comparisons include the small discrepancy between coarse center samples and averages of fine initial fields. The network inputs and evaluation cells remain fixed as the reference resolution increases.

For the spatially discretized heat equation, an additional reference is exact in time. If $a_k \geq 0$ is a diffusion decay rate, set $c_k = (1 - e^{-a_k h})/a_k$ and $v_k = (1 - e^{-2a_k h})/(2a_k)$, using their limits h at zero. Conditional on an observed noise increment, the stochastic convolution on that interval has mean multiplier c_k/h and additional variance multiplier $v_k - c_k^2/h$. Independent Gaussian bridges supply the unresolved within-interval noise, giving an exact-time reference for the chosen spatial discretization. Its analytic conditional mean quantifies the uncertainty associated with observing only coarse noise increments.

Against the jointly refined 512-cell, 4,096-step reference, DeepONet errors are 3.082% for heat and 3.545% for the nonlinear reaction (Table 16). The 256-cell, 2,048-step reference differs from the finest reference by 0.545% and 0.427%, respectively, down from 1.226% and 0.947% at 128 cells and 1,024 steps. The output network advantage is smaller against the finer reference, and the heat linear control remains more accurate. Figure 14 gives the time-resolved comparisons.

Table 16: Independent common-noise reference comparisons. Relative field L^2 errors are in percent, with mean $\pm$ sample standard deviation over five experiments. The time study uses 64 paths per experiment at 64 cells, with the finest numerical reference at 4,096 time steps. The joint study uses 32 paths per experiment and a 512-cell, 4,096-step numerical reference; all fields are restricted to 64 cells at common observation times.

Comparison	Heat	Nonlinear reaction
Time refinement: reference at $64\times$ time resolution		
DeepONet / finest time reference	1.952 ± 0.134	3.113 ± 0.278
Matched MLP / finest time reference	2.234 ± 0.215	3.211 ± 0.247
Coarse / finest time reference	0.959 ± 0.035	1.585 ± 0.140
$32\times$ / $64\times$ time reference	0.0204 ± 0.0006	0.0304 ± 0.0034
Joint refinement: 512-cell, 4,096-step reference		
DeepONet / fine reference	3.082 ± 0.125	3.545 ± 0.184
Matched MLP / fine reference	3.229 ± 0.157	3.656 ± 0.249
Causal linear / fine reference	2.657 ± 0.099	13.324 ± 0.940
64-cell, 64-step / fine reference	2.657 ± 0.099	2.446 ± 0.095
256-cell, 2,048-step / fine reference	0.545 ± 0.016	0.427 ± 0.029

For the spatially discretized heat equation, the exact-time reference comparison uses 64 independent paths per experiment and gives DeepONet error $(1.931 \pm 0.084)\%$, compared with $(0.992 \pm 0.024)\%$ for the coarse numerical reference and learned linear control. The conditional-mean reference has error $(0.626 \pm 0.017)\%$ against the realized exact-time fields. The exact-time errors include both time-discretization error and noise variation within each observed interval.

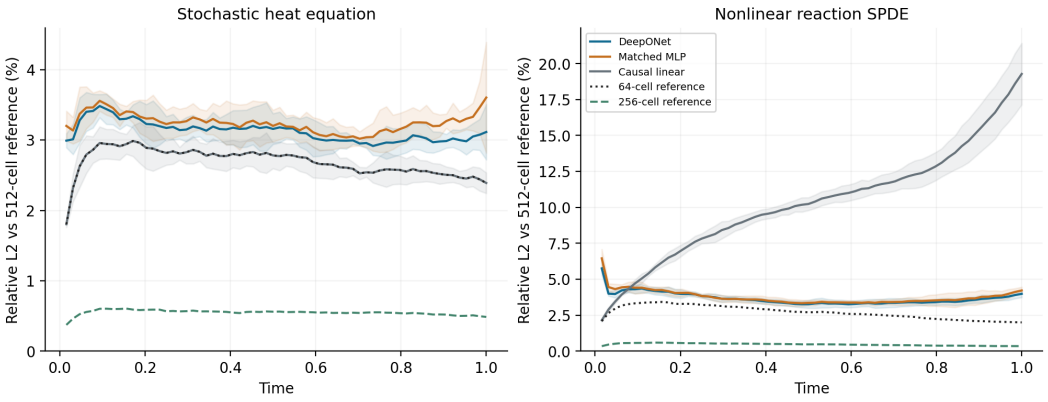


Figure 14: Time-resolved error against the jointly refined 512-cell, 4,096-step reference on 32 independent paths per experiment. Lines average five experiments; shaded bands show sample standard deviations for learned methods. The reference curves use the same noise coupling and common observation cells. The learned heat linear control overlaps the coarse reference curve.

Analytic controls verify heat eigenmodes, constant-field reaction updates, and the covariance of the coupled noise. The supplement provides the code, selected models, training checkpoints, and per-path and per-time errors.

M Additional output network and compression results

M.1 Paired output network comparisons

Figures 15 and 16 show the five experiments for each task and input choice. Each pair of output networks shares training and test paths within a experiment. Comparisons that reuse these paths are dependent; Appendix G gives the statistical reporting conventions.

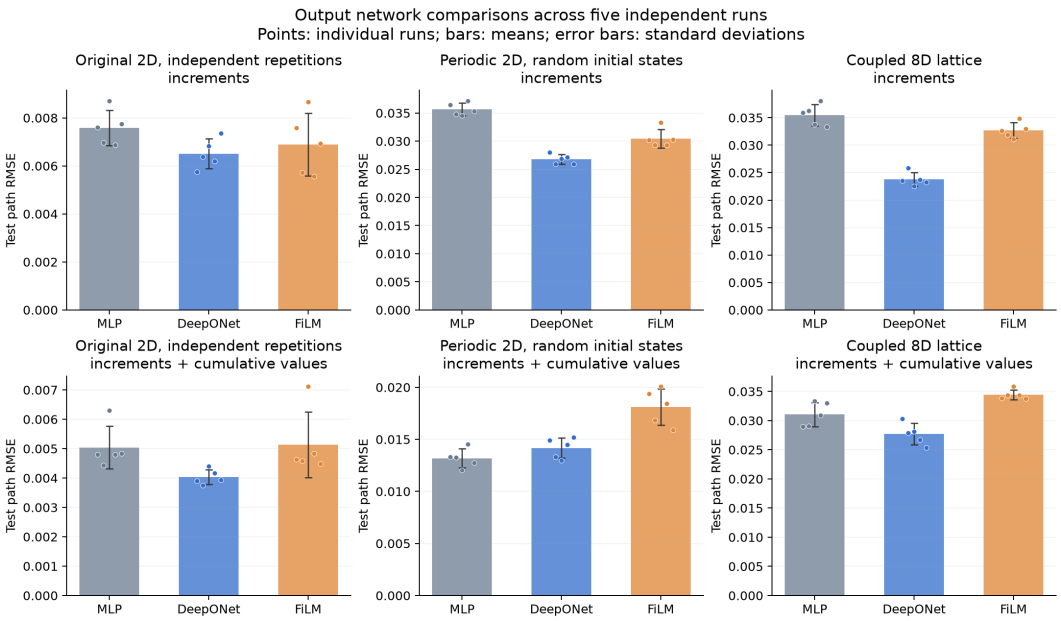


Figure 15: Output network comparison across all six task/input conditions. Points show the five experiments; bars summarize mean path RMSE, with error bars giving sample standard deviations. The branch–trunk benefit varies with the input representation and dynamics, including the periodic-2D reversal with cumulative values. The paired comparisons use shared paths within each condition.

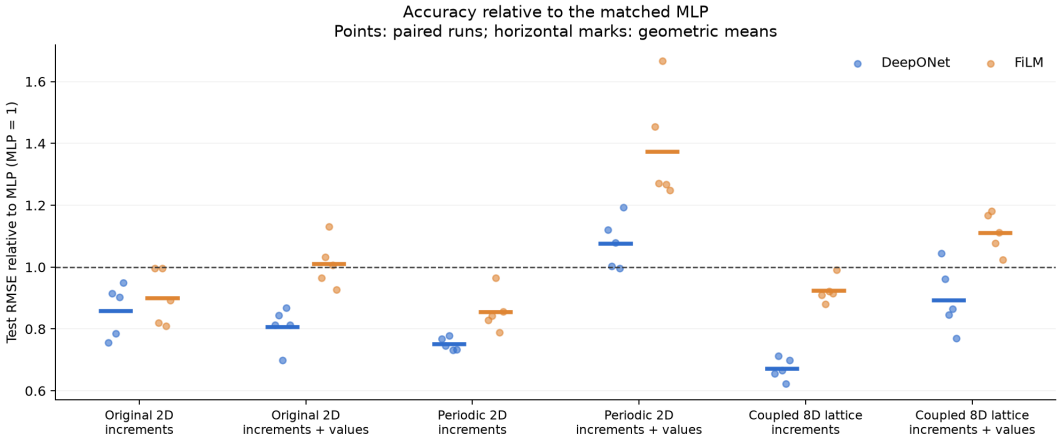


Figure 16: Paired output network comparisons. Points show the test RMSE of DeepONet and FiLM divided by that of the matched MLP in each experiment; horizontal marks show geometric means. Values below one indicate lower error than MLP. Each comparison uses the same data and encoder initialization.

M.2 Exploratory compression outcomes

We compress all 30 branch–trunk models, retaining $r \in \{0, 1, 2, 4, 8, 16, 32\}$ singular components. Validation RMSE or the coefficient bound selects the compressed size before test evaluation, as specified in Appendix G.

Table 17: Compression of all 30 branch–trunk models without retraining. Entries give the range across models. Test RMSE change is $100(\mathrm{RMSE}_{\mathrm{compressed}}/\mathrm{RMSE}_{\mathrm{original}} - 1)$; negative values indicate lower error. Parameter reductions include the encoder.

Selection criterion	Parameters removed (%)	Test RMSE change (%)
Validation error	24.6–46.2	-0.34 to $+5.17$
Error bound $\leq 10^{-3}$	19.9–41.3	-0.0088 to $+0.0245$

Replacing the fitted time-dependent coefficients by their mean increases RMSE to 1.92–11.62 times the original across all 30 models. The encoder still receives time as an input, and its weights are unchanged. The increase therefore comes from removing the fitted coefficient variation. The coefficient-bound conversion has largest computed bound 9.03×10^{-4} and largest observed test-path perturbation 4.75×10^{-5}. The uncompressed networks have about 17,000–29,000 parameters, so these reductions save tens of kilobytes. We have not measured their effect on runtime.

Table 18: Compression selected by validation error. Entries give the range over five independent experiments. Parameter reductions refer to the full model; test RMSE changes are relative to the uncompressed model. Negative changes indicate lower error. The largest increase is 5.17%.

Task	Input	Parameters removed (%)	Test RMSE change (%)
Original 2D	ΔB	25.9	-0.34 to $+1.07$
Original 2D	$\Delta B + B$	24.6–25.5	-0.01 to $+5.17$
Periodic 2D	ΔB	25.0	$+1.16$ to $+2.17$
Periodic 2D	$\Delta B + B$	24.6–25.5	-0.13 to $+2.98$
Lattice 8D	ΔB	46.2	$+0.01$ to $+0.22$
Lattice 8D	$\Delta B + B$	44.4	-0.02 to $+0.03$

Table 19: Compression selected using the prediction-change bound with absolute tolerance 10^{-3}. Entries give the range over five independent experiments. Parameter reductions refer to the full model; test RMSE changes are relative to the uncompressed model. Negative changes indicate lower error.

Task	Input	Parameters removed (%)	Test RMSE change (%)
Original 2D	ΔB	25.0	-0.0075 to $+0.0245$
Original 2D	$\Delta B + B$	24.6	-0.0088 to $+0.0030$
Periodic 2D	ΔB	19.9–23.3	-0.0006 to $+0.0036$
Periodic 2D	$\Delta B + B$	22.9	-0.0033 to $+0.0015$
Lattice 8D	ΔB	36.7	0.0000
Lattice 8D	$\Delta B + B$	41.3	-0.0002 to $+0.0002$

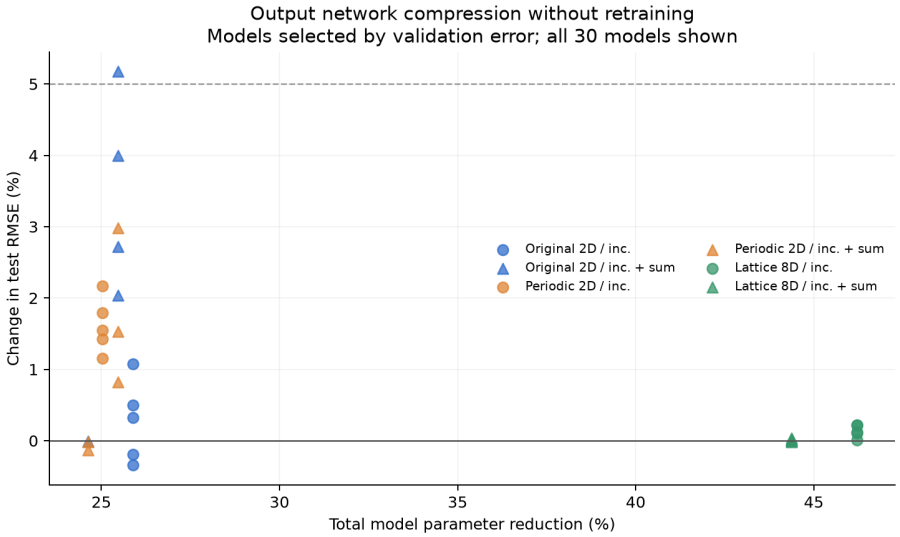


Figure 17: All 30 compressed models selected by validation error. The vertical axis shows the percentage change in test RMSE relative to the uncompressed model; negative values indicate lower error. The largest increase is 5.17%. Parameter reduction counts the full network, including the unchanged encoder.

N Exploratory application to neural-network weight dynamics

Noisy neural-network optimization provides another possible application. Consider a mean-field two-layer network

$$f_{\boldsymbol{\theta}}(z) = \frac{1}{N}\sum_{i=1}^{N} b^i \kappa(\langle a^i, z\rangle), \qquad \theta^i = (a^i, b^i), \tag{45}$$

with noisy-gradient updates

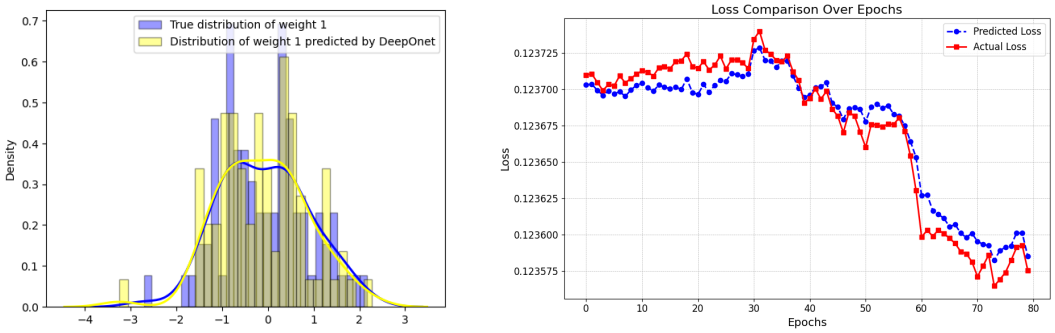
$$\theta^i_{k+1} = \theta^i_k - \gamma \nabla_{\theta^i} \widetilde{L}(\boldsymbol{\theta}_k) + \sigma\sqrt{2\gamma}\,\xi^i_{k+1}. \tag{46}$$

The corresponding formal small-step interacting diffusion is

$$d\theta^i_t = b(\theta^i_t, \widehat{\mu}_t)\,dt + \sigma\sqrt{2}\,dB^i_t, \qquad \widehat{\mu}_t = N^{-1}\sum_{j=1}^{N} \delta_{\theta^j_t}. \tag{47}$$

This suggests learning weight trajectories from their initial values and optimizer noise. As in the particle benchmark, one coordinate's noise history does not determine the influence of the other coordinates.

The archived regression task fits $y = \sqrt{x}$ using a width-100, bias-free tanh network initialized at zero. The distribution experiment uses 1000 regression samples, 1000 noisy-gradient steps, $\gamma = 0.2$, $\sigma = 0.05$, and a fixed random-number stream. The operator has hidden width 32 and uses 1001 Adam steps at learning rate 10^{-3}, with 20 neuron-coordinate trajectories for fitting. The recorded random-number settings are retained in the supplementary code. The continuation experiment uses 100 noisy-gradient steps with $\gamma = 0.01$ and 501 operator-training steps; the first 20 checkpoints are used for an 80-step continuation illustration.



(a) Final first-layer weights.

(b) Network loss during continuation.

Figure 18: Archived illustrations from the exploratory neural-weight prototype. The panels illustrate the proposed application; the implementation issues described below prevent quantitative conclusions.

Limitations of the prototype The archived implementation contains overlapping noise indices and an overwritten layer predictor; its plotted distribution names may also be reversed. The continuation split uses a single interacting optimization experiment. These issues prevent quantitative interpretation of the plotted errors; full records and the code review are included in the supplement. A confirmatory experiment would require corrected indexing, separate layer predictors, independent optimization experiments, and a held-out horizon.

O Reproducibility Statement

Appendices A–D give the solution-map assumptions, approximation and model-selection results, and compression proofs. The benchmark equations and training settings appear in Appendices F and G; Appendices H–M give the detailed experimental settings and results, including the direct SPDE algorithm. The supplementary material includes all experiment code, model checkpoints, numerical records, and table-generation scripts. Its implementation notes record software settings, reproducibility checks, and the limitations of the archived illustrative experiments.