Mechanisms of Misgeneralization in Physical Sequence Modeling

Kento Nishi[1,2,3,4,5,6] **Raphael Tang**[7] **Karun Kumar**[3]
Core Francisco Park[4,5†] **Hidenori Tanaka**[4,5†]

[1]Harvard College [2]Harvard John A. Paulson School of Engineering and Applied Sciences
[3]Comcast AI [4]CBS-NTT Program in Physics of Intelligence, Harvard University
[5]Physics of Artificial Intelligence Laboratories, NTT Research, Inc. [6]MATS [7]Microsoft
Correspondence: knishi@mit.edu

Abstract

Generative sequence models are often trained to plan motion in physical domains, from robotics to mechanical simulations. When constructing a dataset to train such a model, engineers may curate demonstrations to specify how trajectories should be distributed over a *physical quantity* like travel distance or mechanical energy. For example, a roboticist building a maze navigation agent might choose demonstrations whose travel distances cover a fixed range uniformly, hoping to constrain the agent's expected power usage. We find that standard deep learning can violate this intent: each generated trajectory can seem plausible on its own, but the aggregate distribution over the physical quantity is wrong. We call this failure *physical misgeneralization*. In this work, we identify and formalize the problem, develop and validate an account of its mechanism, and use that account to compare and design mitigations. Using controlled synthetic tasks, we show that physical misgeneralization arises when local errors typical of the model class propagate through the physical measurement to shift the recovered distribution. We estimate these errors with a *data deviation kernel*, and we use it to predict which physical quantities gain or lose mass under real diffusion sequence models trained on both toy and applied trajectory generation tasks. Finally, our mechanistic interpretation helps identify which mitigations are structurally promising. We first compare interventions that modify the dataset and generative interface, then show that the mechanistic understanding we develop throughout this paper suggests a novel way to construct data representations informed by the kernel. We show that our proposed mitigation, which transforms the coordinate system under which the generator learns and samples, effectively reduces quantity drift.

1 Introduction

The advent of generative sequence models has reshaped how intelligent physical systems are built. In domains like robotics and dynamical modeling, generative deep learning is now used to synthesize robot policies [Ghosh et al., 2024], forecast multi-agent motion for autonomous driving [Jiang et al., 2023], sample probabilistic weather futures [Price et al., 2025], model molecular dynamics trajectories [Jing et al., 2024], and even construct action-controllable environments [Bruce et al., 2024]. Many of these systems learn from fixed collections of demonstration trajectories, and so the dataset is often the most direct lever to specify what physical behavior the learned generator should reproduce. This is especially clear in robot learning, where engineers select for state-action trajectories that contribute the most information to the policy [Hejna et al., 2025], retarget demonstrations across object poses, scene layouts, robot arms, and dexterous hands [Mandlekar et al., 2023, Jiang et al., 2025], and collect safe or unsafe rollouts for aircraft taxiing, aerial navigation, and dynamical systems [Chou et al., 2018, Ciftci et al., 2025]. Across these cases, the purpose of shaping the dataset is to make the learned model reproduce the chosen physical structure at generation time.

† Equal advising. Preprint. kentonishi.com/physical-misgeneralization

Consider, then, a roboticist building a maze navigation agent from demonstrations. Their primary objective is to make the agent learn how to move from the start to the goal. However, they may also care about other physical quantities: for instance, travel distance can affect power usage. To constrain this behavior, the roboticist curates the demonstration paths so that travel distances follow a fixed distribution, hoping that the agent will reproduce that distribution when it generates new paths. If we train a planning model on this collection, one might reasonably expect that samples from the planner would recover approximately the same mixture of travel distances. However, in practice, this expectation fails. A diffusion planner [Janner et al., 2022] trained on D4RL Maze2D [Fu et al., 2020], a standard maze-navigation benchmark, learns to solve the maze; yet, when we measure the travel distances of those generated paths, the distribution is shifted upward relative to that of the training data (Fig. 1). The model therefore learned enough about maze-like motion to produce recognizable trajectories, but it failed to preserve the physical mixture that the dataset was meant to specify. We call this failure *physical misgeneralization*.

Why does this happen? Peering into how the model is trained, travel distance does not explicitly appear as an optimization target for the diffusion model. Consequently, we are asking the model to place approximately the right amount of probability in each measured distance bin, without telling it that travel distance is an evaluation criterion. This demand is stronger than producing valid plans: the planner can introduce slight detours while moving through the maze, and those deviations may be small from the perspective of the trajectory-space loss, but they may still have large effects on the distances traveled. The mechanism at play, then, seems to be that the marginal objective allows local trajectory deviations that preserve recognizable motion, but those same deviations convert into uneven transfer of probability mass along the measured physical quantity of interest.

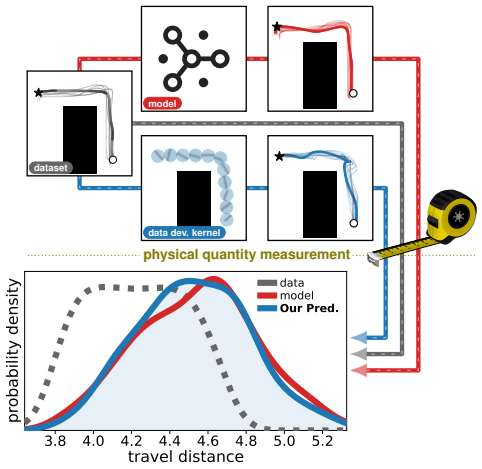


Figure 1: **We identify the mechanism by which sequence models fail to match the distribution of a physically measured quantity.** Imagine training an agent to navigate a maze, with a dataset curated so the distribution of travel distances falls in a safe range. After training, the model can solve the maze, but its paths have longer travel distance than the ones in the **training data**. We unpack why: **local errors typical of the model class** get propagated through the **distance measurement** to cause drift. By approximating model errors as a ***data deviation kernel***, we can closely predict how the model shifts the distribution.

As stated, this mechanistic interpretation is only conjectural. To validate it, we need evidence that it correctly predicts empirical observations, *i.e.*, that it can predict the movement of probability mass in real trained models. Natural physical datasets rarely expose all pieces needed to carry out such tests, because the measured quantity, the relationships between quantities and trajectories, and the modeling method can all be deeply entangled. Motivated by this, we construct synthetic tasks in which a known scalar quantity governs a family of trajectories. For each task, we fix an intended prior, map quantities to sequences through an explicit trajectory-generating rule, and recover the quantity from generated trajectories through a shared measurement method. We still train the generator unconditionally on trajectories, preserving the structure of the navigation vignette; but, because we know the structure of the data generation process by construction, these tasks let us isolate the mechanism from the confounds of natural physical datasets. In both the synthetic setups and applications like 2D maze navigation and double-pendulum simulation, we show that emulating model-like errors with a *data deviation kernel* and propagating its perturbations through the physical measurement helps us forecast which quantity values gain or lose mass, without using the trained model's residual directions. Our key contributions are as follows:

- **We identify and formalize *physical misgeneralization***, a failure mode that arises in the context of physical sequence models trained under marginal objectives. We observe this failure with diffusion models, autoregressive Transformers, and VAEs. (Sec. 1, Sec. 2, Sec. 3, Appx. F)
- **We develop a mechanistic interpretation and validate its predictions.** Model deviations, estimable as a *data deviation kernel*, propagate through the physical measurement to move probability across quantity values. (Sec. 4) We design synthetic tasks with known physical quantities that let us control the priors, trajectory-generating rules, and physical quantity measurements to test the

mechanism's predictions. (Sec. 5.1) We show our mechanism strongly predicts which quantities get over- or under-represented by diffusion sequence models in both synthetic and applied tasks. It also explains the motivating maze navigation case. (Sec. 5.2)

- **We use the mechanistic interpretation to compare and design mitigations.** We compare interventions at varying levels: the dataset composition, generative model interface, and the input-output representation. We then use the kernel concept to propose a data representation that reduces drift in practice (Sec. 6).

In summary, this paper establishes that physical misgeneralization is an anticipatable consequence of the interaction between local model-induced deviations and quantities that govern the physical world.

2 Related Work

Generative sequence models for physical trajectories. These models are popular in trajectory-based planning and forecasting for autonomous driving, weather, molecular dynamics, and environment simulation [Janner et al., 2022, Ajay et al., 2023, Chi et al., 2025, Ghosh et al., 2024, Jiang et al., 2023, Price et al., 2025, Jing et al., 2024, Bruce et al., 2024]. Separately, several works have also proposed explicitly physics-aware architectures: Hamiltonian Neural Networks learn vector fields through a Hamiltonian and thereby preserve conservation laws via inductive biases [Greydanus et al., 2019]; Lagrangian Neural Networks parameterize Lagrangians in settings where canonical momenta are unavailable [Cranmer et al., 2020]. Although hardcoding physical laws can be valuable, the free-form generative sequence modeling approach remains widely used because it affords greater flexibility. We therefore focus on the most common marginal modeling paradigm, in which the generator is trained without baking the organizing physical structure into the model interface.

Latent variables, aggregate posteriors, and inverse problems. The act of recovering quantities from data algebraically resembles computing the aggregate posterior in a variational autoencoder. In VAEs, a prior and likelihood define a latent-variable model, a learned recognition model maps observations to an approximate posterior, and averaging that posterior over observations gives the aggregate posterior [Kingma and Welling, 2014, Blei et al., 2017]. ELBO surgery analyzes how this average encoding distribution differs from the prior [Hoffman and Johnson, 2016]; adversarial autoencoders and Wasserstein autoencoders then regularize the encoded distribution toward the prior [Makhzani et al., 2016, Tolstikhin et al., 2018]. The same algebraic resemblance extends to inverse problems, where Bayesian formulations combine a prior and likelihood into a posterior over hidden quantities, while classical treatments of ill-posedness and identifiability show that small perturbations in observed data can change the recovered quantity [Tarantola, 2005, Kaipio and Somersalo, 2005, Tikhonov and Arsenin, 1977, Bellman and Åström, 1970]. Nonetheless, previous literature does not study the case central to this paper, wherein a generative model is trained marginally on data, and the distribution of the physical quantity is recovered by measuring generated samples.

Biases of marginal generative models. Literature on sampling bias explains how the distribution learned by a generative model can systematically differ from the ground-truth data distribution. For diffusion models, this includes mode interpolation, signal leakage, selective underfitting of some regions, local recombination of training examples, and closeby detours [Aithal et al., 2024, Everaert et al., 2024, Song et al., 2025, Kamb and Ganguli, 2025, Zhao and Grover, 2023]. Autoregressive sequence models are prone to similar issues like exposure bias, since teacher-forced training conditions on data prefixes whereas free-running generation conditions on the model's own sampled history [Schmidt, 2019, Bengio et al., 2015, Lamb et al., 2016, Ross et al., 2011]. These works mainly focus on individual sample quality via metrics like likelihood, perplexity, FID, and other notions of predictive fit [Meister and Cotterell, 2021, Naeem et al., 2020]; our work complements these works by interrogating whether the aggregate of those individual samples obeys the intended distribution over a measured physical quantity absent from the model interface.

Synthetic data generation for mechanistic understanding. The interaction between generator error and measurement sensitivity is difficult to isolate in natural physical datasets alone. As such, we construct synthetic tasks in the same spirit as past mechanistic studies which designed controlled tasks to expose structure that would have otherwise remained entangled. Synthetic tasks have been used to advance the field's understanding of compositional generalization, reasoning, training dynamics, and internal representations [Chan et al., 2022, Li et al., 2023, Nanda et al., 2023, Reddy, 2024, Brinkmann et al., 2024, Lubana et al., 2025, Yang et al., 2025, Park et al., 2025, Nishi et al., 2025]. This is made possible by the fact that knowing how the data are constructed allows for intervening

and ablating parts of the pipeline to pick apart which components produce the behavior in question. For physical misgeneralization, this means imposing a prior over a physical quantity, defining a rule to generate a trajectory for each quantity value, and fixing a measurement rule to recover the quantity. With these elements in place, we can synthetically generate data, train models on that data, vary parts of the procedure, and identify the mechanistic root cause that drives the mismatch.

3 Problem Formalization

We now formalize the failure by separating all moving parts: namely, the physical quantity, measurement, trajectories, and model. We provide a visual aid to follow along in Fig. 2.

Let $r \in \mathcal{R} \subseteq \mathbb{R}$ denote the scalar quantity whose marginal distribution we measure, let z collect the remaining variation, and let x denote the resulting trajectory. We use z to index the within-quantity variation in x, *i.e.*, how trajectories can still differ after we fix r. For example, mechanical energy (r) does not uniquely identify double-pendulum motion (x): two pendulums can start from different positions with the same energy. In this illustrative example, z would describe the differing initial positions.

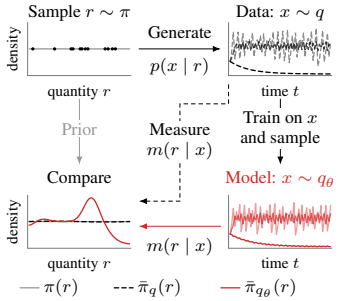


Figure 2: **A schematic of the pipeline.** *(1)* Sample $r \sim \pi$, then $x \sim p(\cdot \mid r)$. *(2)* Train q_θ on trajectories x. *(3)* Apply m to data and model samples. *(4)* Compare marginals $\bar{\pi}_q$ and $\bar{\pi}_{q_\theta}$ with π.

Suppose data are generated by first drawing (r, z) from a joint distribution $\rho(r, z)$ and then drawing $x \sim p(x \mid r, z)$. Since the training data supplied to the model contain only trajectories, the marginal distribution seen during training is

$$q(x) \;=\; \int p(x \mid r, z)\, \rho(r, z)\, dr\, dz. \tag{1}$$

The target distribution over the physical quantity is the marginal

$$\pi(r) \;=\; \int \rho(r, z)\, dz. \tag{2}$$

Thus, although we measure a scalar marginal, the remaining variation can still be coupled to that quantity through the joint data-generating process. For the scalar comparison used in our experiments, we integrate this variation into an effective conditional trajectory family,

$$p(x \mid r) \;=\; \int p(x \mid r, z)\, \rho(z \mid r)\, dz, \tag{3}$$

so that equivalently $q(x) = \int p(x \mid r)\, \pi(r)\, dr$. In this reduced form, preserving the intended mixture means recovering $\pi(r)$ after the remaining variation has been averaged into $p(x \mid r)$. When more quantities are relevant, we can use a vector-valued $\vec{r}$ or a projection of the quantities.

Remark 1 (Prior absorption)**.** We can also absorb the shape of the prior into the scalar coordinate. For any scalar prior π, let $T : [0, 1] \to \mathcal{R}$ be a transport map with $T_{\#}\mathrm{Unif}([0, 1]) = \pi$; for a continuous prior, this is the usual quantile map $T(u) = F_\pi^{-1}(u)$. Writing $r = T(u)$ with $u \sim \mathrm{Unif}([0, 1])$ gives

$$q(x) = \int p(x \mid r)\, \pi(r)\, dr = \int_0^1 p(x \mid T(u))\; du, \tag{4}$$

so we can absorb any non-uniform prior into a reparameterized map from quantity values to trajectories while leaving the marginal training distribution unchanged. Thus, we can generally use a uniform prior over the reparameterized quantity for convenience, composing the data family with T. △

We write q_θ for the trajectory distribution learned from samples of q, where θ parameterizes the model. Because both q and q_θ are trajectory distributions, we define their mixtures over the physical quantity by applying a measurement rule to trajectories. We write this rule as $m(r \mid x)$; it may return a point value or a distribution over candidate quantity values. We adopt the latter formulation of $m(r \mid x)$ in our synthetic experiments, because an imperfect trajectory may admit several comparably good estimates. Concretely, we compare a trajectory with reference trajectories generated at candidate quantity values, assigning higher probability to closer matches (Eq. 16, Appx. A.2.1). We apply the same m to data and model samples. Fixing m in this way allows us to isolate changes in the trajectory distribution from changes in the measurement. Under these conventions, for any source distribution S over trajectories, the induced marginal over the physical quantity is therefore

$$\bar{\pi}_S(r) \;=\; \mathbb{E}_{x \sim S}\big[m(r \mid x)\big]. \tag{5}$$

Definition 1 (Quantity drift)**.** Let $\bar{\pi}_{\text{data}} \coloneqq \bar{\pi}_q$ and $\bar{\pi}_{\text{model}} \coloneqq \bar{\pi}_{q_\theta}$ under a shared method for measuring the physical quantity. For any distributional distance or divergence metric d, the d-measured quantity drift of q_θ is

$$\text{Drift}_d(q_\theta) \coloneqq d(\bar{\pi}_{\text{model}}, \pi). \tag{6}$$

We interpret this value against the data baseline $d(\bar{\pi}_{\text{data}}, \pi)$. We say the model has substantial drift under d when $\text{Drift}_d(q_\theta) \gg d(\bar{\pi}_{\text{data}}, \pi)$. △

In practice, we mainly report the drift using total variation,

$$\text{TV}(\bar{\pi}_{q_\theta}, \pi) = \frac{1}{2} \int_{\mathbb{R}} \left| \bar{\pi}_{q_\theta}(r) - \pi(r) \right| dr. \tag{7}$$

We use total variation because it directly measures how much probability mass must move between quantity values. Alternative choices, such as Jensen–Shannon divergence, Hellinger distance, Wasserstein–1, and KL divergence under compatible supports, also suffice.

4 Predicting Quantity Drift with the Data Deviation Kernel

To predict how the distribution changes, we further need a signed redistribution relative to the recovered data marginal. We write this signed drift as

$$\tau_\theta(r) \coloneqq \bar{\pi}_{q_\theta}(r) - \bar{\pi}_q(r), \tag{8}$$

so that positive values correspond to quantity values over-represented by the model, while negative values correspond to quantity values depleted by generation. Since both recovered marginals are obtained by applying the same measurement rule to trajectory distributions, expanding Eq. 8 gives

$$\tau_\theta(r) = \int \big(q_\theta(x) - q(x)\big) m(r \mid x)\, dx. \tag{9}$$

Under this expansion, a trajectory-space error contributes mass to the quantity value recovered for the corresponding trajectory. When the model places excess probability on trajectories recovered as r, it increases $\tau_\theta(r)$; conversely, missing probability decreases $\tau_\theta(r)$. That is to say, the effect of a trajectory-space discrepancy depends on where that discrepancy lies relative to the measurement rule.

If we knew the signed model error $q_\theta - q$, we could compute the drift directly from Eq. 9. Our goal, though, is to anticipate this signed movement without estimating q_θ, the hypothetical model, as a full distribution. To do this, we need a local surrogate for the signed error induced when a given architecture is trained on a given dataset and objective.

Definition 2 (Data deviation kernel)**.** The conditional distribution $D^{\sigma}_{\text{arch}}(\delta \mid x)$ over deviations δ around x summarizes how a specific model architecture family might readily disperse the probability mass of each data point. The scalar σ controls the strength of the deviations applied by the kernel. △

Importantly, σ is a free strength parameter; when making a prediction, one can—but is not required to—choose σ so the resulting deviation magnitude roughly matches a target total error scale, such as the loss level of the model quality one wishes to forecast. Consider the extremes: a perfect model and a randomly initialized model with the same architecture. The former reproduces the data marginal, and hence its recovered quantity distribution, exactly; the latter need not. These two cannot share one useful predicted quantity distribution. Calibration serves to account for this difference in model fitness: when we use the model's data-space error to set the deviation magnitudes, we make a *model-calibrated comparison* (Sec. 5.1, Appx. C.3). Note that it is also possible in principle to vary σ without any information about the target model to obtain a *model-free relationship* between induced data-space error and quantity drift. In this sense, the kernel's functional form remains agnostic to the trained model and can predict drift before training, a priori, with no reliance on residual directions, generated samples, the recovered marginal over the physical quantity, or a separately trained model.

Remark 2 (Kernel for diffusion)**.** Recall from Sec. 2 that diffusion models are known to distort the learned distribution in data-space through mode interpolation, signal leakage, selective underfitting, local recombination, and closeby detours [Aithal et al., 2024, Everaert et al., 2024, Song et al., 2025, Kamb and Ganguli, 2025, Zhao and Grover, 2023]. These results together suggest that, around a data trajectory x, trained models move probability towards nearby pieces of trajectories inasmuch as the model or sampler readily confuses those fragments. To reflect this, we instantiate a simple local kernel: for each local state or window, we replace the original piece with a similar fragment drawn from the data using Gaussian distance weights. This definition is derived from established diffusion phenomena, albeit as a tractable approximation. Specifically, local recombination [Kamb and Ganguli, 2025] motivates local replacement, while mode interpolation [Aithal et al., 2024] and

selective underfitting Song et al. [2025] motivate using empirically available swaps drawn from the training data. To allow perturbations beyond the observed fragments to "make up" for variation not already supplied by those fragments, we also add Gaussian noise (details in Appx. C.1). Intuitively, this kernel swaps small pieces of each trajectory with fragments from other trajectories that look similar, with noise accounting for any remaining local perturbation. △

With D^{σ}_{arch} specified, we predict drift by asking which quantity values are recovered after kernel perturbations. For a trajectory drawn from $p(x \mid r)$, we sample a deviation from $D^{\sigma}_{\text{arch}}(\cdot \mid x)$ and measure the perturbed trajectory $x + \delta$. Averaging over both draws gives the quantity-transport kernel

$$K_{\text{dev}}(r' \mid r) := \mathbb{E}_{x \sim p(x|r)} \mathbb{E}_{\delta \sim D^{\sigma}_{\text{arch}}(\cdot|x)} \left[m(r' \mid x + \delta) \right]. \tag{10}$$

In words, $K_{\text{dev}}(r' \mid r)$ is the probability that a trajectory associated with r is recovered as r' after a deviation. We then apply this transition to the intended prior to obtain the predicted r marginal

$$\bar{\pi}_{\text{dev}}(r') = \int K_{\text{dev}}(r' \mid r) \pi(r) \, dr. \tag{11}$$

Thus, with just a data deviation kernel and the recovery rule used to measure trajectories, $\bar{\pi}_{\text{dev}}$ predicts which quantity values the generative model may over- or under-represent.

For a concrete example for when drift can occur at all, suppose the quantity takes only two values r_1, r_2 with intended probability $1/2$ each, and the measurement recovers these values exactly on the data. Let K_{ij} be the probability that a trajectory generated from r_i is measured as r_j after a deviation. The recovered probability at r_2 is then $\frac{1}{2}K_{12} + \frac{1}{2}K_{22} = \frac{1}{2} + \frac{1}{2}(K_{12} - K_{21})$. For the recovered marginal to match the uniform prior, each quantity must receive as much probability as it loses. If trajectories from r_1 are more often measured as r_2 after a deviation than the reverse, probability accumulates at r_2. Thus, local errors in x-space can induce asymmetric movement in r-space.

5 Experiments: Validating the Mechanistic Interpretation

5.1 Defining the Data Generation Processes

Controlled synthetic tasks. For the predictive calculation to explain physical misgeneralization, it must anticipate the drift of trained models. We first test this with synthetic constructions because they give us direct control over the quantity that generates each trajectory, while leaving the model in the same marginal training setup as explained in the earlier vignettes. For each task, we draw a scalar quantity $r \sim \text{Unif}(\mathcal{R})$, set $x_0 = 0.25$, generate a length-H trajectory $x = g_{\mathcal{S}}(r)$, and train a marginal 1D U-Net diffusion model on trajectories alone. The intended mixture over r therefore touches training only through the trajectories. After generation, we infer the model's induced mixture over r from samples in the same way that we infer it from ground-truth data. Throughout the synthetic suite, we use sequence horizon length $H = 64$ and consider three one-dimensional physical sequence families. We start with the sinusoid family

$$x_t = \tfrac{1}{2} \left(\sin(2\pi t / r + \phi_0) + 1 \right), \qquad r \in [32, 128], \tag{12}$$

where $\phi_0 = -\pi/6$ and r is the period in timesteps. We use this as a simple learnable baseline because increasing r stretches the oscillation over time such that nearby periods produce nearby sequences over the horizon (proof in Appx. D.2.1). We then use the tent map

$$x_{t+1} = r \min\{x_t, 1 - x_t\}, \qquad r \in [0, 2], \tag{13}$$

which is piecewise linear and has slope magnitude r at every differentiable state $x_t \neq 1/2$. Solving the Lyapunov exponent yields $\lambda_H(r) = \log r$ (proof in Appx. D.2.2), meaning that sequences are non-chaotic for $r \leq 1$, while the exponential growth rate of local state perturbations increases logarithmically as r approaches 2. Finally, we use the logistic map,

$$x_{t+1} = r x_t (1 - x_t), \qquad r \in [0, 4], \tag{14}$$

as a sanity check. This is a classical nonlinear population dynamics model in which r controls the growth rate. The logistic map is challenging to predict because its Lyapunov exponent depends on realized orbits (proof in Appx. D.2.3); we include it here to test whether our mechanism generalizes beyond the tent map. For these iterated maps, we discretize intermediate states to 1024 bins and use the same numerical recurrence whenever we generate data or evaluate the prediction. Note that we deliberately construct these tasks to avoid variation over z, by fixing the phase for sinusoid and the initial condition for tent and logistic. This lets us isolate drift over r from variation over z.

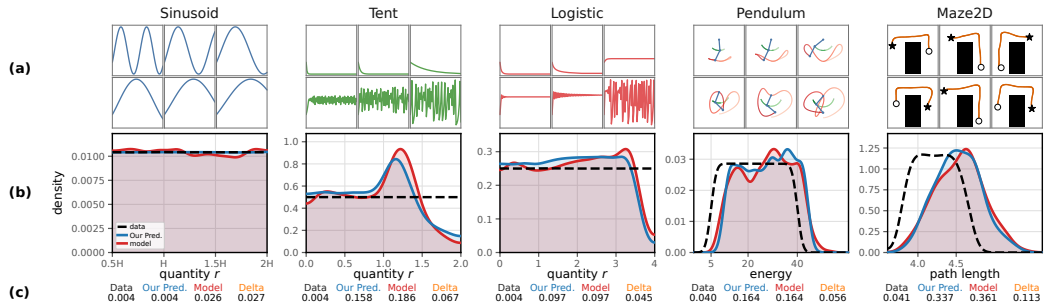


Figure 3: **Trained models closely replicate the mechanism's predicted physical quantity drift.** *(a)* Representative visualizations of trajectories in each dataset. *(b)* The mechanism (blue) predicts that the sinusoid curve will remain nearly flat like the intended prior, whereas for tent and logistic, it forecasts excess mass at intermediate r and depleted mass near the upper end of the range. For double-pendulum, it forecasts a lateral and bumpy translation towards higher mechanical energy; for Maze2D, it suggests a tightened and shifted distribution of longer path lengths. The actual trained model (red) closely replicates our predictions, with little movement for sinusoid, mid-range gain followed by high-r deficits for tent and logistic maps, and the expected drifts and shapes for double-pendulum and Maze2D. *(c)* The TV values compare distances from the prior, and the predictions' delta to the models. The small delta values indicate that our predictions forecast trained models well.

Training, quantity recovery, and prediction details. For each model, we train on $200{,}000$ trajectories with a 1D U-Net diffusion backbone (base width 32, kernel size 3) and sample with 256 DDPM steps. For plotting, we use $25{,}000$ ground-truth trajectories and $10{,}000$ generated trajectories. To recover the mixture over r, we compare each trajectory to reference trajectories on a grid of $2^{14} = 16{,}384$ quantity values (details in Appx. A.2.1) and average the resulting recovery posteriors over trajectories. For example, tent map trajectories for $r = 0.80$ and $r = 0.81$ rapidly decay toward zero, so an imperfect generation between them may receive comparable likelihood under either value.

To compare our prediction against each trained diffusion model, we choose the scale σ so the overall scalar data-space deviation induced by the kernel matches the model's overall scalar data-space error. We measure this error directly after generation as the aggregate root-mean-square residual between model samples and the data trajectories associated with their recovered quantity values, and use the variation in per-trajectory error magnitudes over the recovered quantity to set relative local kernel scales (Appx. C.3). We also repeat each experiment with 3 seeds to account for variance (Appx. E). For additional details, see Appx. A.1; for visualizations of data and generated trajectories, see Appx. B.

Applied tasks: double-pendulum simulation and Maze2D navigation. For double-pendulum, we generate position trajectories from the ground-truth equations, curate the data so mechanical energy is uniform over the range $[5, 40]$, and train the diffusion model only on the two angles' coordinates. We recover the quantity by estimating angular velocities from finite differences and measuring the median shifted energy along the rollout (Appx. A.2.2). For Maze2D, we train on fixed-horizon movement segments extracted from D4RL U-maze sparse-v1, curated to have a uniform distribution over path length over the range $[3.80, 4.65]$. After filtering wall collisions, we recover the quantity by measuring total path length from the generated position sequence (Appx. A.2.3).

5.2 Results: Comparing the Prior, Data, Mechanistic Predictions, and Trained Model

In Fig. 3, we compare the distributions over quantities for the intended prior, the ground-truth data, the mechanism's predictions, and the model. Immediately, we can see that the mechanism forecasts little movement for the sinusoid, a large mid-range rise and high-r depletion for the tent map, a broader high-r rise and drop for the logistic map, an upward and unstable energy shift in double-pendulum, and a reshaped and upward shift of path length in Maze2D. Then, we inspect the distributions implied by the trained models: their curves closely track our predictions, with small TV deltas to the predicted curve in each case. The sinusoid model remains close to the intended mixture, whereas the tent map model over-represents intermediate values around $r \in [1.1, 1.4]$ and under-represents high values above roughly $r = 1.5$; similarly, the logistic map places excess mass around $r \in [3.2, 3.6]$ and depleted mass near the upper end of the range. Further, the double-pendulum model matches our

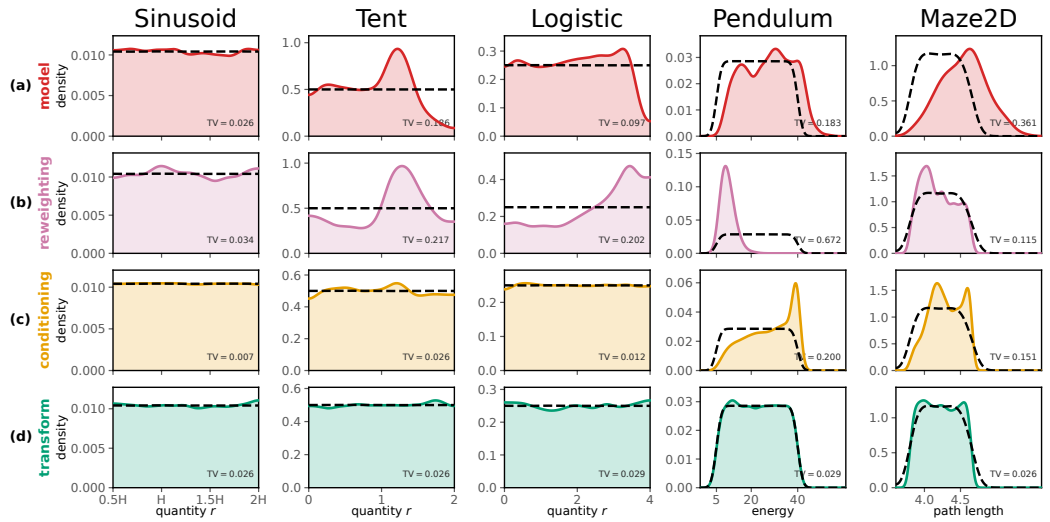


Figure 4: **Mechanism-informed interventions can reduce drift.** One can attempt to reduce drift in three distinct ways: *(b)* rebalancing the dataset, *(c)* modeling conditionally, and *(d)* transforming the input-output data representation. The strongest and most consistent correction comes from using the kernel to derive a coordinate transformation that balances mass transfer between quantity values.

predicted energy distribution well, even reproducing where the intermediate plateau was transformed into minute peaks and troughs. The Maze2D planner also matches our prediction of a tighter, more triangular, and right-shifted distribution, with its peak aligning with that of the prediction.

We also find that the predicted profile is highly robust to changes in σ (see Appx. C.3). In net, this agreement supports the mechanistic interpretation: local model-induced deviations, passed through the physical measurement, can help anticipate which quantity values gain or lose mass.

For model-free predictions via scales selected without calibrating to information from trained models, comparisons across training checkpoints, and additional validations of the kernel, see Appx. C.2–C.5. Separately, for evidence that residuals between prediction and model are not systematically concentrated in regions with high Lyapunov exponents across tent and logistic maps, see Appx. D.3.

6 Mitigating Physical Misgeneralization

We have thus far focused on predicting how probability moves; we now ask what can be done to reduce that movement. Importantly, before trying mitigations, we took care to ablate and rule out many simpler potential causes in sampling, recovery, architecture, seeds, and more (Appx. F).

Modifying the dataset. Among the categories of possible interventions, perhaps the most immediate is one that modifies what the generator is trained on. That is, one can re-curate the dataset via over- or under-sampling, filtering, stratification, weighted losses, and synthetic data generation to counteract the model's preferences over r. Conveniently, for our experimental setups described in Sec. 5.2, these methods effectively all reduce to reweighted sampling, in which we adjust the probability of showing each trajectory to the model. If dataset rebalancing is to be effective, re-training the model with data sampled by the inverse of the recovered quantity marginal of the initial model should noticeably reduce the drift. However, when we apply this dataset correction (details in Appx. A.3.1), we find that it is ineffective and sometimes counterproductive (Fig. 4b). The poor performance of the dataset-level rebalancing follows naturally from the mechanistic picture developed earlier: although it changes how often different quantity values appear in the training set, it does not change the fact that the generator is trained marginally on trajectories, and therefore does not prevent readily expressed local errors from carrying probability mass across quantity values. Moreover, reweighting one measured quantity can silently change other structure in the data. In the chaotic maps, for example, different regimes of r induce different trajectory statistics under the recurrence; therefore, increasing the sampling rate of one part of the quantity range also changes the distribution of states shown to the generator. The same concern becomes more severe in applied physical datasets, where path length, start and goal endpoints, environment geometry, energy, and other physical quantities need not vary independently.

Modifying the generative interface. One might attempt a more aggressive intervention that directly provides the desired quantity to the generator. Conditional modeling and guided sampling are natural examples of this category; fortunately, classifier-free guidance [Ho and Salimans, 2021] unifies these ideas, and we can use it to condition diffusion models and test whether they can be steered towards requested values of r picked according to the intended prior (details in Appx. A.3.2). We begin by testing this idea in the synthetic suite, mirroring our analysis of the dataset-level interventions and prior mechanistic studies that use fully controlled constructions to isolate an effect, then upgrade to richer but still structurally interpretable settings to test its scope. Conditional modeling passes the first minimum test (Fig. 4c): the trajectory-generating rule is evidently simple enough for the model to largely follow once r is supplied. However, success in this controlled setting does not imply success in the richer settings: in double-pendulum and Maze2D, the occupied range of energy and path length is partially repaired, but the recovered distributions still contain large density spikes (Fig. 4c). This happens because conditioning retains two weaknesses of the dataset-level intervention: it does not compensate for local trajectory-space errors that move probability, and it does not specify the joint distribution between the requested quantity and the remaining structure needed to realize it. For double-pendulum, the first issue is especially visible because low energy requires smooth position trajectories whose finite-difference velocities remain small, so small position-space errors recover as excess energy. For Maze2D, the second issue is especially visible because the requested path length must be jointly feasible with the start point, endpoint, maze geometry, and horizon, since a path that is too short cannot reach the goal. Thus, modifying the generative interface can reduce drift when the requested quantity indexes a relatively simple trajectory family, but its utility diminishes when the quantity must be jointly realized with other physical structure.

Transforming the data representation. The mechanistic understanding developed throughout this paper suggests another category of intervention, one that transforms the coordinate system under which the generator learns and samples. Applying domain knowledge to choose input-output representations is standard practice: spectrograms for audio, one-hot tokenization for language, and learned encoders for vision are common. Now, with mechanistic knowledge of the model's expected error geometry, we can use the kernel to construct even more favorable data representations. Concretely, since physical misgeneralization arises when local neighborhoods around trajectories are imbalanced in the density of recovered quantity values, we can reduce drift by changing the geometry in which those neighborhoods are formed. We first construct an unlabeled Latin-hypercube support in the input shape expected by the diffusion model; these codes define the coordinate system in which the model will see the data. We then apply the kernel in this code space to identify which code neighborhoods are likely to exchange probability mass. Using these neighborhoods, we can pair code points and training trajectories such that local decoding neighborhoods recover a balanced mixture over quantity values, while preserving the proportion of training trajectories assigned to each quantity. Then, we train an unconditional diffusion model with the codes as the model-facing data representation. At generation time, the model samples codes, after which they are pulled back through the code mapping and evaluated with the usual recovery rule. Here, the generator never receives r; therefore, it avoids introducing new sources of information into the interface as does conditional modeling. Fig. 4d confirms that transforming the model-facing data representation consistently gives strong corrections, causing far less drift than comparable methods across all the synthetic and applied systems. Furthermore, this preserves desiderata like sample quality and diversity well (see Appx. A.3.3 for details and checks). As a caveat, this strategy may not always be applicable, *i.e.*, when a particular representation choice is non-negotiable. For instance, an audio application may require a model that works in waveforms rather than spectrograms; analogous mandates could exist for physical sequence modeling. Still, the core point of this strong result is that the kernel can be extended towards novel and effective mitigations, more of which we hope future works will explore.

7 Limitations and Broader Implications

Kernels crafted for diffusion models. Our choice to focus on diffusion-based physical sequence models follows from both the empirical vignettes that motivate the paper and the surrounding literature: diffusion models are the de facto architecture in these physical modeling settings, and recent work has already begun to describe how learned diffusion samplers move probability mass in data space. This gave us a principled starting point to implement a data deviation kernel that captures the essence of how diffusion models express errors around data. However, the kernel remains a

coarse operationalization of these tendencies, compressing architecture, optimization, finite data, and sampling effects into a tractable local movement model. This explains why the predictions in Fig. 3 capture the drift patterns strongly, even predicting minor peaks and troughs, but are not pixel-perfect. Likewise, the model-facing data representation mitigation strategy in Sec. 6 targets neighborhoods implied by the approximate kernel rather than the true sampler itself.

The specificity to diffusion also limits the scope of the predictive account; indeed, in Appx. F, we show that autoregressive Transformers, VAEs, and MLPs trained on the tent map task also misgeneralize the distribution over the recovered quantity, but we do not make predictions for them because we do not yet define appropriate kernels for these architectures. We encourage future works along these lines: if we can come to better understand how other generative model families err in data space, those characterizations can inform the design of data deviation kernels. Once those kernels are specified, the mechanism developed in this paper can again be used to anticipate which quantity values gain or lose mass and to guide mitigation strategies of the kinds studied in Sec. 6. It is also possible that, given the effectiveness of the transform mitigation with our approximate kernel, the mitigation could be extended to Transformers and VAEs without a fully specified kernel; a coarse estimate of which local neighborhoods exchange probability may suffice to construct an effective coordinate transform.

Scope beyond physical sequence models. Intriguingly, our preliminary results with a text-to-speech model suggest that the phenomenon may also arise in settings that are not usually framed as physical sequence modeling. In Appx. G, we show an example in which the speaking rate of audio generated by a Tacotron 2–HiFi-GAN pipeline [Shen et al., 2018, Kong et al., 2020, Ravanelli et al., 2021] shifts upward relative to the training corpus. Although this mismatch might be an instance of physical misgeneralization, extending our current framework to explain the text-to-speech pipeline precisely would require composing kernels across multiple learned stages. Concretely, an acoustic model first produces an intermediate representation, and a separate vocoder maps that representation into waveforms; since speaking rate is measured only in waveform space, the kernel must describe how errors of the acoustic model and vocoder interact with one another. Future work should develop data deviation kernels for such composed pipelines, where each stage can reshape probability mass before the final measurement is applied.

Algorithmic bias and fair representation. This failure mode has natural analogues to fairness: a generative model may produce individually plausible samples while systematically changing how different groups, styles, or behaviors are represented in aggregate. This becomes especially problematic when the shifted quantity corresponds to a protected or socially salient attribute, such that already underrepresented populations become further misrepresented. Reducing this kind of misalignment a priori is difficult when we do not know which axes of variation are consequential; however, when there are attributes or behaviors that practitioners have reason to care about, our results support evaluating and tuning generative models along those axes explicitly, in the spirit of disaggregated evaluations [Barocas et al., 2021]. Once such an axis is specified, our mechanism can detect representational drift as a movement of probability mass and may help correct it through the kind of kernel-aware mitigation developed in this work.

8 Conclusion

In summary, we identify the problem of physical misgeneralization in sequence models, and reveal its mechanism to be one in which local deviations induced by the model are propagated through measurements of physically meaningful quantities. We situate the phenomenon in the relevant literature, formalize the problem, propose a novel mechanistic perspective, and show that it has predictive power in accurately forecasting how real trained diffusion models transport density over physical quantities. Our experimental methodology involves designing synthetic physical sequence tasks that give us full control over the full data generation process, spanning the prior over the physical quantity, the trajectory-generating rule, and the measurement method for recovery. Through extensive experiments with these synthetic systems and applied vignettes like double-pendulum motion simulation and Maze2D navigation, we validate the mechanistic interpretation. We also discuss and compare major categories of possible mitigation strategies, and showcase how the kernel may help inform useful data representation designs that reduce drift. Taken together, our work contributes significant new understanding that we hope will be valuable for making physical systems more safe and transparent.

Acknowledgments and Disclosure of Funding

KN thanks Harvard University and NTT Research for jointly funding this work, Comcast for the opportunity to advance this project through an internship, Daniel Filan and the MATS Program (Machine Alignment, Transparency, and Security; Cohort 7.0) for supporting the early stages of this research, and Sophie L. Wang for providing valuable feedback on the writing.

References

Sumukh K Aithal, Pratyush Maini, Zachary C. Lipton, and J. Zico Kolter. Understanding hallucinations in diffusion models through mode interpolation. In *Advances in Neural Information Processing Systems*, volume 37, pages 134614–134644. Curran Associates, Inc., 2024. doi: 10.52202/079017-4278. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/f29369d192b13184b65c6d2515474d78-Abstract-Conference.html.

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=sP1fo2K9DFG.

Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. In *Advances in Neural Information Processing Systems*, volume 33, pages 12449–12460. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/hash/92d1e1eb1cd6f9fba3227870bb6d7f07-Abstract.html.

Solon Barocas, Anhong Guo, Ece Kamar, Jacquelyn Krones, Meredith Ringel Morris, Jennifer Wortman Vaughan, W. Duncan Wadsworth, and Hanna Wallach. Designing disaggregated evaluations of ai systems: Choices, considerations, and tradeoffs. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, pages 368–378. ACM, 2021. doi: 10.1145/3461702.3462610.

Richard Bellman and Karl J. Åström. On structural identifiability. *Mathematical Biosciences*, 7(3–4): 329–339, 1970. doi: 10.1016/0025-5564(70)90132-X.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/hash/e995f98d56967d946471af29d7bf99f1-Abstract.html.

David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017. doi: 10.1080/01621459.2017.1285773.

Jannik Brinkmann, Abhay Sheshadri, Victor Levoso, Paul Swoboda, and Christian Bartelt. A mechanistic analysis of a transformer trained on a symbolic multi-step reasoning task. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 4082–4102, 2024. doi: 10.18653/v1/2024.findings-acl.242.

Jake Bruce, Michael D. Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Maria Elisabeth Bechtle, Feryal Behbahani, Stephanie C. Y. Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando De Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative interactive environments. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 4603–4623. PMLR, 2024. URL https://proceedings.mlr.press/v235/bruce24a.html.

Stephanie C. Y. Chan, Adam Santoro, Andrew K. Lampinen, Jane X. Wang, Aaditya Singh, Pierre H. Richemond, Jay McClelland, and Felix Hill. Data distributional properties drive emergent in-context learning in transformers. In *Advances in Neural Information Processing Systems*, volume 35, pages 18878–18891. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/77c6ccacfd9962e2307fc64680fc5ace-Abstract-Conference.html.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10–11):1684–1704, 2025. doi: 10.1177/02783649241273668.

Glen Chou, Dmitry Berenson, and Necmiye Ozay. Learning constraints from demonstrations, 2018. URL https://arxiv.org/abs/1812.07084.

Yusuf Umut Ciftci, Darren Chiu, Zeyuan Feng, Gaurav S. Sukhatme, and Somil Bansal. SAFE-GIL: SAFEty guided imitation learning for robotic systems. In *IEEE International Conference on Robotics and Automation*, pages 3559–3566, 2025. doi: 10.1109/ICRA55743.2025.11128298.

Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian neural networks, 2020. URL https://arxiv.org/abs/2003.04630.

Martin Nicolas Everaert, Athanasios Fitsios, Marco Bocchio, Sami Arpa, Sabine Süsstrunk, and Radhakrishna Achanta. Exploiting the signal-leak bias in diffusion models. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 4025–4034, 2024.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2020. URL https://arxiv.org/abs/2004.07219.

Dibya Ghosh, Homer Rich Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Quan Vuong, Ted Xiao, Pannag R. Sanketi, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024. doi: 10.15607/RSS.2024.XX.090. URL https://www.roboticsproceedings.org/rss20/p090.html.

Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/hash/26cd8ecadce0d4efd6cc8a8725cbd1f8-Abstract.html.

Joey Hejna, Suvir Mirchandani, Ashwin Balakrishna, Annie Xie, Ayzaan Wahid, Jonathan Tompson, Pannag Sanketi, Dhruv Shah, Coline Devin, and Dorsa Sadigh. Robot data curation with mutual information estimators, 2025.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. URL https://openreview.net/forum?id=qw8AKxfYbI.

Matthew D. Hoffman and Matthew J. Johnson. ELBO surgery: Yet another way to carve up the variational evidence lower bound. In *NIPS 2016 Workshop on Advances in Approximate Bayesian Inference*, 2016. URL https://approximateinference.org/archives/2016/accepted/HoffmanJohnson2016.pdf.

Keith Ito and Linda Johnson. The LJ speech dataset. https://keithito.com/LJ-Speech-Dataset/, 2017.

Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 9902–9915. PMLR, 2022. URL https://proceedings.mlr.press/v162/janner22a.html.

Chiyu Max Jiang, Andre Cornman, Cheolho Park, Benjamin Sapp, Yin Zhou, and Dragomir Anguelov. Motiondiffuser: Controllable multi-agent motion prediction using diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9644–9653, 2023. doi: 10.1109/CVPR52729.2023.00930.

Zhenyu Jiang, Yuqi Xie, Kevin Lin, Zhenjia Xu, Weikang Wan, Ajay Mandlekar, Linxi Fan, and Yuke Zhu. Dexmimicgen: Automated data generation for bimanual dexterous manipulation via imitation learning. In *IEEE International Conference on Robotics and Automation*, pages 16923–16930, 2025. doi: 10.1109/ICRA55743.2025.11127809.

Bowen Jing, Hannes Stärk, Tommi Jaakkola, and Bonnie Berger. Generative modeling of molecular dynamics trajectories. In *Advances in Neural Information Processing Systems*, volume 37, pages 40534–40564. Curran Associates, Inc., 2024. doi: 10.52202/079017-1282. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/478b06f60662d3cdc1d4f15d4587173a-Abstract-Conference.html.

Jari P. Kaipio and Erkki Somersalo. *Statistical and Computational Inverse Problems*. Applied Mathematical Sciences. Springer, 2005. doi: 10.1007/b138659.

Mason Kamb and Surya Ganguli. An analytic theory of creativity in convolutional diffusion models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 28795–28831. PMLR, 2025. URL https://proceedings.mlr.press/v267/kamb25a.html.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. HiFi-GAN: Generative adversarial networks for efficient and high fidelity speech synthesis. In *Advances in Neural Information Processing Systems*, volume 33, pages 17022–17033. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/hash/c5d736809766d46260d816d8dbc9eb44-Abstract.html.

Alex M. Lamb, Anirudh Goyal, Ying Zhang, Saizheng Zhang, Aaron C. Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper_files/paper/2016/hash/16026d60ff9b54410b3435b403afd226-Abstract.html.

Kenneth Li, Aspen K. Hopkins, David Bau, Fernanda Viegas, Hanspeter Pfister, and Martin Wattenberg. Emergent world representations: Exploring a sequence model trained on a synthetic task. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=DeG07_TcZvT.

Ekdeep Singh Lubana, Kyogo Kawaguchi, Robert P. Dick, and Hidenori Tanaka. A percolation model of emergence: Analyzing transformers trained on a formal language. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=0pLCDJVVRD.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders, 2016. URL https://arxiv.org/abs/1511.05644.

Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 1820–1864. PMLR, 2023. URL https://proceedings.mlr.press/v229/mandlekar23a.html.

Clara Meister and Ryan Cotterell. Language model evaluation beyond perplexity. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5328–5339, 2021. doi: 10.18653/v1/2021.acl-long.414.

Muhammad Ferjad Naeem, Seong Joon Oh, Youngjung Uh, Yunjey Choi, and Jaejun Yoo. Reliable fidelity and diversity metrics for generative models. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 7176–7185. PMLR, 2020. URL https://proceedings.mlr.press/v119/naeem20a.html.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. In *International Conference on Learning Representations*, 2023.

Kento Nishi, Rahul Ramesh, Maya Okawa, Mikail Khona, Hidenori Tanaka, and Ekdeep Singh Lubana. Representation shattering in transformers: A synthetic study with knowledge editing. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 46525–46553. PMLR, 2025. URL https://proceedings.mlr.press/v267/nishi25a.html.

Core Francisco Park, Andrew Lee, Ekdeep Singh Lubana, Yongyi Yang, Maya Okawa, Kento Nishi, Martin Wattenberg, and Hidenori Tanaka. Iclr: In-context learning of representations. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=pXlmOmlHJZ.

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.

Ilan Price, Alvaro Sanchez-Gonzalez, Ferran Alet, Tom R. Andersson, Andrew El-Kadi, Dominic Masters, Timo Ewalds, Jacklynn Stott, Shakir Mohamed, Peter Battaglia, et al. Probabilistic weather forecasting with machine learning. *Nature*, 637(8044):84–90, 2025. doi: 10.1038/s41586-024-08252-9.

Mirco Ravanelli, Titouan Parcollet, Peter Plantinga, Aku Rouhe, Samuele Cornell, Loren Lugosch, Cem Subakan, Nauman Dawalatabad, Abdelwahab Heba, Jianyuan Zhong, Ju-Chieh Chou, Sung-Lin Yeh, Szu-Wei Fu, Chien-Feng Liao, Elena Rastorgueva, François Grondin, William Aris, Hwidong Na, Yan Gao, Renato De Mori, and Yoshua Bengio. Speechbrain: A general-purpose speech toolkit, 2021. URL https://arxiv.org/abs/2106.04624.

Gautam Reddy. The mechanistic basis of data dependence and abrupt learning in an in-context classification task. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=aN4Jf6Cx69.

Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 627–635. PMLR, 2011. URL https://proceedings.mlr.press/v15/ross11a.html.

Florian Schmidt. Generalization in generation: A closer look at exposure bias. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 157–167, 2019. doi: 10.18653/v1/D19-5616.

Jonathan Shen, Ruoming Pang, Ron J Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, Rj Skerry-Ryan, et al. Natural TTS synthesis by conditioning WaveNet on mel spectrogram predictions. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4779–4783, 2018. doi: 10.1109/ICASSP.2018.8461368.

Kiwhan Song, Jaeyeon Kim, Sitan Chen, Yilun Du, Sham Kakade, and Vincent Sitzmann. Selective underfitting in diffusion models, 2025. URL https://arxiv.org/abs/2510.01378.

Albert Tarantola. *Inverse Problem Theory and Methods for Model Parameter Estimation*. Society for Industrial and Applied Mathematics, 2005. doi: 10.1137/1.9780898717921.

Andrei N. Tikhonov and Vasiliy Y. Arsenin. *Solutions of Ill-Posed Problems*. Winston, Washington, D.C., 1977.

Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein auto-encoders. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HkL7n1-0b.

Yongyi Yang, Core Francisco Park, Ekdeep Singh Lubana, Maya Okawa, Wei Hu, and Hidenori Tanaka. Swing-by dynamics in concept learning and compositional generalization. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=s1zO0YBEF8.

Siyan Zhao and Aditya Grover. Decision stacks: Flexible reinforcement learning via modular generative models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 80306–80323. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/fe1c4991d57f37dfef62d01b3901ca54-Paper-Conference.pdf.

A Experimental Details

A.1 Setups and Hyperparameters

Data normalization. Unless stated otherwise, we linearly map each coordinate of each trajectory into $[-1, 1]$ before the diffusion model is trained, and map it back into the range of physical coordinates before we measure the quantity. Thus, we give the generator normalized sequences, whereas we recover the quantity in the physical units used to define the dataset.

Model and optimizer. The default denoiser for the synthetic systems uses channel multipliers $(1, 2, 2, 2)$, a sinusoidal embedding of the diffusion timestep with dimension 32, Mish activations, group normalization, and no attention blocks. We train with AdamW using learning rate $3 \cdot 10^{-4}$, weight decay 10^{-5}, batch size 1024, and EMA decay 0.999.

Plotting conventions. For each one-dimensional plot over quantity values, we use one grid over quantity values for all curves shown in the panel. When a curve estimates density, we use the same KDE convention across all curves in that figure. For density curves, we compute total variation on the plotted grid after normalizing each curve to integrate to 1.

Compute. We train our models using an in-house desktop PC with 6 NVIDIA A6000 GPUs with 48GB of VRAM each. The synthetic experiments take roughly 24 GPU-hours; the applied double-pendulum, Maze2D, and text-to-speech experiments take roughly 120 GPU-hours.

A.2 Recovering Physical Quantities

A.2.1 Synthetic Setups

Let $\{r_j\}_{j=1}^{J}$ be the grid over quantity values, and let $g(r_j)$ be the trajectory generated at r_j. To recover r from an observed trajectory x, we compare x against each reference trajectory. With

$$e_j(x) = \frac{1}{H}\|x - g(r_j)\|_2^2, \tag{15}$$

we define

$$\begin{aligned} m(r_j \mid x) &\propto \exp\left(-\frac{e_j(x)}{2\sigma^2(x)}\right)\pi(r_j), \\ \sigma^2(x) &= \max\left(\min_j e_j(x), \sigma_{\min}^2\right). \end{aligned} \tag{16}$$

The adaptive bandwidth's positive floor prevents a near-exact match to a reference trajectory from collapsing the posterior to a degenerate point mass. When we report a single recovered quantity value rather than a posterior over r, we use the posterior mode.

A.2.2 Double-pendulum Energy

We choose initial conditions to realize the sampled energy and rescale angular velocities after each integration step to maintain that energy.

For a double-pendulum angle trajectory sampled with $\Delta t = 0.01$, we recover energy by first estimating angular velocities with central differences,

$$\dot{q}_t = \frac{q_{t+1} - q_{t-1}}{2\Delta t}, \tag{17}$$

and compute

$$E_t = \frac{1}{2}\dot{q}_t^\top M(q_t)\dot{q}_t + V(q_t) - V_{\min}, \tag{18}$$

at each interior timestep, where $V_{\min}$ is the potential energy at the downward equilibrium. We then take the median over time. The median suppresses occasional spikes from finite differences caused by local irregularities in generated angles.

A.2.3 Maze2D Path Length

For a Maze2D position trajectory $x = (q_0, \ldots, q_{H-1})$ with $H = 128$, we recover the quantity by computing total path length

$$\ell(x) = \sum_{t=0}^{H-2} \|q_{t+1} - q_t\|_2. \tag{19}$$

Equivalently, because the horizon and timestep are fixed, in our implementation we bin by trajectory speed $\ell(x)/((H-1)\Delta t)$.

For diffusion models that generate positions directly, we draw start–goal pairs from data segments and inpaint the endpoint positions with noise matched to each diffusion timestep. We filter generated paths by connectivity through free space and step length.

A.3 Details of the Mitigations

A.3.1 Inverse Reweighting

Let $b(r) \in \{1, \ldots, B\}$ denote the bin containing quantity value r, let π_b be the intended mass in bin b, and let $\hat{\pi}_b$ be the mass recovered from samples from the baseline model. For a fixed training set $\{x_i\}_{i=1}^N$ with recovered values r_i, we sample trajectory i in the next training run with probability proportional to

$$w_i = \frac{\pi_{b(r_i)}}{\max\{\hat{\pi}_{b(r_i)}, 10^{-6}\}}. \tag{20}$$

We then normalize $\{w_i\}_{i=1}^N$ to sum to 1 before drawing mini-batches. For the synthetic systems, since data are generated from known quantity values, we implement the intervention in practice by first sampling a bin from the inverse prior

$$\tilde{\pi}_b \propto \frac{1}{\max\{\hat{\pi}_b, 0.10/B\}}, \tag{21}$$

then sampling r uniformly within that bin and generating the corresponding trajectory from the ground-truth rule. We use $B = 64$ for this synthetic inverse prior, $B = 40$ for double-pendulum, and $B = 20$ for Maze2D.

A.3.2 Conditional Modeling

For each conditional model, we first map the requested quantity value r to a sinusoidal embedding with the same dimension as the diffusion timestep embedding. Writing $u = (r - r_{\min})/(r_{\max} - r_{\min})$, the embedding contains dyadic Fourier features

$$c(r) = \big(\sin(2\pi 2^0 u), \ldots, \sin(2\pi 2^{m-1} u), \cos(2\pi 2^0 u), \ldots, \cos(2\pi 2^{m-1} u)\big), \tag{22}$$

truncated or padded to the denoiser embedding dimension. We add $c(r)$ to the timestep embedding, use the same vector to modulate each residual block with FiLM [Perez et al., 2018], and also project it to a constant sequence of condition channels concatenated to the input. At generation time, we draw requested quantity values from the intended prior and sample with classifier-free guidance weight 1. For Maze2D, the start and goal endpoints are clamped via inpainting, as in the original work.

A.3.3 Coordinate Transform

Let $\{x_i\}_{i=1}^N$ be the trajectory support used for the intervention, and let $\nu_i(r) = m(r \mid x_i)$ be the recovered posterior over the physical quantity for trajectory i. We draw an unlabeled code support $\{y_j\}_{j=1}^N$ from a Latin-hypercube design in $[0,1]^D$, where D is the flattened input dimension of the diffusion model. The codes define only the model-facing coordinates; we use the recovered physical quantity to choose a pairing $a : \{1, \ldots, N\} \to \{1, \ldots, N\}$ that assigns physical trajectories to code points. To choose said pairing, we first apply a local kernel of the same form as the prediction kernel to code space. For each source code y_j, the kernel produces perturbed codes $\widetilde{y}$, and the decoder assigns $\widetilde{y}$ to its k_{dec} nearest code points with weights

$$\alpha_\ell(\widetilde{y}) \propto \exp\left(-\frac{\|\widetilde{y} - y_\ell\|_2^2 - \|\widetilde{y} - y_{\ell_1}\|_2^2}{2\tau}\right), \tag{23}$$

where ℓ_1 is the nearest code point and τ is the median squared distance to the farthest neighbor among the decoder neighbors. This construction gives a sparse matrix M whose entry $M_{j\ell}$ is the expected

decoder weight from source code j to code cell ℓ. If code cell ℓ is paired with trajectory $x_{a(\ell)}$, then the quantity mixture locally decoded around source code j is

$$p_j(r) = \sum_{\ell=1}^{N} M_{j\ell} \nu_{a(\ell)}(r). \tag{24}$$

We initialize the pairing so that the number of assigned trajectories in each quantity bin matches the data, then improve it with random cross-bin swaps that reduce

$$\frac{1}{N} \sum_{j=1}^{N} \|p_j - \pi\|_2^2 . \tag{25}$$

After this pairing has been chosen, we train an unconditional diffusion model using the paired codes $\{y_j\}$ as the coordinate representation of the data. At generation time, a sampled code is decoded by the same k_{dec}-nearest-neighbor weights over paired code cells, either by averaging the recovered posteriors for evaluation or by sampling one paired support trajectory according to those weights when we need an explicit trajectory. For Maze2D, we sample a support trajectory and inpaint coordinates at both ends of its paired code, with noise matched to the diffusion timestep.

Maze2D validity and diversity checks. Here, we carry out several tests to ensure that the typical movement, overall sample quality, and overall sample diversity are not harmed by the transform mitigation. Under these checks, the transform maintains the relevant desiderata while reducing drift over the measured quantity: median step lengths, distances to the nearest data trajectories, and maze occupancy remain comparable to those of the data. This addresses the potential worry that the coordinate transform is changing the generator interface in an unfair way that trades off these aforementioned desiderata.

Desideratum	Data	Baseline	Transform
Sample consistency (median step length)	0.034	0.034	0.034
Sample quality (distance to nearest data trajectory)	0.053	0.047	0.054
Sample diversity (maze occupancy)	0.728	0.153	0.792

- **Median step length:** The median displacement between adjacent states over all retained trajectories. The more comparable this quantity is to the data, the better.
- **Distance to nearest data trajectory:** The median distance from each retained trajectory to the nearest trajectory in the data; for data, we use the leave-one-out nearest-neighbor distance. The more comparable this quantity is to the data, the better.
- **Maze occupancy:** The fraction of free maze cells visited by retained trajectories. The more comparable this quantity is to the data, the better.

A.4 Code Repository

We provide detailed instructions for how to replicate and verify the results. We are also including the full code for this work as a GitHub repository:

https://github.com/KentoNishi/neurips26-physical-misgeneralization

B Additional Data Visualizations

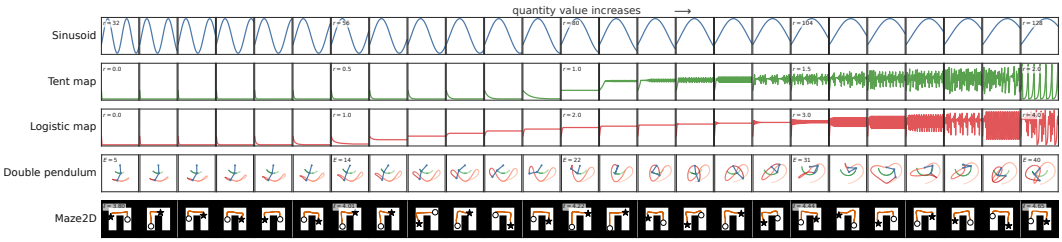


Figure 5: **Representative trajectories across the quantity ranges used to construct our datasets.** For each setup, we show 25 trajectories ordered from low to high quantity value. In the sinusoid, tent, and logistic rows, we vary the scalar quantity r; in the double-pendulum row, we vary total energy; in the Maze2D row, we vary path length.

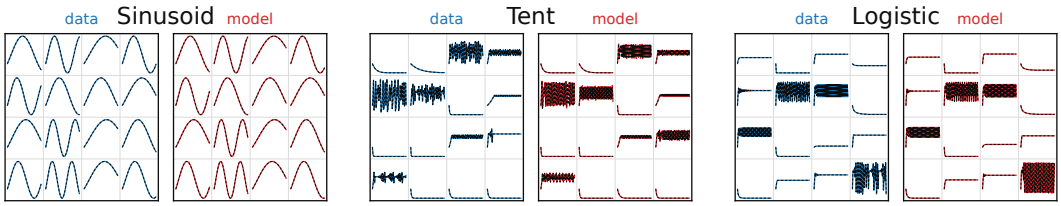


Figure 6: **Representative reconstructions for synthetic trajectories.** For each setup, we overlay representative trajectories with rollouts from the ground-truth data generation rule conditioned on the quantity recovered by the posterior mode. The colored curve is the trajectory being recovered, and the black dashed curve is the reconstructed trajectory from the posterior mode.

C Implementation of the Data Deviation Kernel

C.1 Kernel for Diffusion-Based Physical Sequence Models

The kernel must describe the local form of the diffusion sequence models' errors in trajectory space. Since training exposes the model only to trajectories, we build this surrogate from the coordinates of those trajectories. We emphasize that although we are hand-designing this kernel, the decisions along the way are not arbitrary. Rather, they are derived from the insights of each of the prior works we cited in relation to diffusion model phenomena. Specifically, our use of local neighborhoods and Gaussian noise follows from these prior works:

- Kamb and Ganguli [2025] show that convolutional diffusion models recombine fragments from training data at local scales. The kernel's nearest neighbors term captures this.
- Everaert et al. [2024] shows that diffusion models are biased towards data seen in training. This justifies instantiating local recombination of training data.
- Finally, Aithal et al. [2024], Song et al. [2025], Zhao and Grover [2023] each show that diffusion models that denoise from Gaussian noise can be thrown off the desired manifold during the denoising process. This justifies using perturbations beyond the empirical replacements; we use Gaussian noise to fill missing variance.

Defining the local recombination neighborhood. Let $x = (x_0, \ldots, x_{H-1})$ denote a trajectory, with $x_t \in \mathbb{R}^d$. We index one local piece by u; in our experiments, u is a timestep. Let x_u denote the state at that timestep, and let $\mathcal{B}_u$ be the set of pieces from data at the same sequence location. Let d_u be the dimension of this local piece. For a local scale $\sigma_u(x) > 0$ set by the kernel strength σ, assign each candidate piece the Gaussian local weight

$$\omega_u(z \mid x) = \exp\left(-\frac{\|z - x_u\|_2^2}{2\sigma_u(x)^2}\right), \qquad z \in \mathcal{B}_u. \tag{26}$$

We then choose the empirical support of the draw using a density-stabilized Gaussian neighborhood. The support radius should approximate a full Gaussian draw when the data resolves the local tails, while remaining conservative when only the local core is reliably sampled. Thus, $\alpha = 3$ represents the effectively full local Gaussian neighborhood, and $\alpha = 1$ represents its core. For $\alpha \in \{1, 3\}$, let

$$\mathcal{N}_u^\alpha(x) = \{z \in \mathcal{B}_u : \|z - x_u\|_2 \le \alpha\sigma_u(x)\}. \tag{27}$$

We measure the local resolution by the number of data pieces inside $\mathcal{N}_u^1(x)$. We set the cutoff for calling a neighborhood dense by taking the median of these one-scale counts for each system, then using the largest power of 2 not exceeding the median of those medians. When this one-scale neighborhood is densely resolved, we use the widest support $\mathcal{N}_u^3(x)$; when it is sparse, we use the tight support $\mathcal{N}_u^1(x)$.

Remark 3. Intuitively, this lets thin but densely sampled one-dimensional supports use a wider Gaussian neighborhood, while higher-dimensional or constrained supports stay more local. △

For those higher-dimensional or constrained supports, the radius can contain very different numbers of observed pieces across locations; so, we use the density-stabilized radius to estimate the typical number of pieces in the local Gaussian neighborhood, and implement the draw with that many nearest pieces, still weighted by the Gaussian distance weights above. If this empirical neighborhood is empty, we use the nearest observed piece. This gives the neighborhood $\mathcal{N}_u(x)$ used by the draw.

Drawing the replacement fragment from the neighborhood. We then perturb the local piece by drawing a replacement Z_u from this neighborhood,

$$\Pr(Z_u = z \mid x) = \frac{\omega_u(z \mid x)}{\sum_{z' \in \mathcal{N}_u(x)} \omega_u(z' \mid x)}, \qquad z \in \mathcal{N}_u(x). \tag{28}$$

In the dense one-dimensional case, this draw is carried out by sampling a truncated Gaussian offset and snapping to the nearest observed piece on the discretized support. When a task imposes additional feasibility requirements, such as fixed endpoints, local continuity, or maze free-space constraints, the same local draw is restricted or projected according to those requirements as the perturbed trajectory is assembled.

Remark 4. For Maze2D, these restrictions keep candidate replacements compatible with the fixed endpoints used by the inpainting setup and the surrounding path. △

For the 1D U-Net diffusion models we use, the replacement is resampled with average run length set by the first-layer convolutional kernel size, so sampled pieces persist across short local runs.

Incorporating continuous noise. The draw above gives a discrete empirical perturbation. However, the kernel should also be able to induce continuous variation around this recombination, because a diffusion error need not land exactly on another observed piece. If the empirical draw from earlier already supplies the desired coordinate-wise variance, the residual variance we must add is zero; otherwise, we represent this residual variation with a Gaussian residual ϵ_u. Let the coordinate-wise variance introduced by empirical draws from these neighborhoods be

$$v_u = \mathrm{Var}(Z_u - x_u)\,, \tag{29}$$

where this variance is estimated over the sampled empirical perturbations and $v_u \in \mathbb{R}^{d_u}$. When the empirical draw contributes less than the target variance $\sigma_u(x)^2$ in a coordinate, the remaining variance is added as

$$\epsilon_u \sim \mathcal{N}\left(0, \mathrm{diag}\Big(\big[\sigma_u(x)^2\mathbf{1} - v_u\big]_+\Big)\right). \tag{30}$$

In either case, the perturbed local piece is $\widetilde{x}_u = Z_u + \epsilon_u$.

The final data deviation kernel. Given the collection of local draws, we assemble a perturbed trajectory and apply the admissibility map $\Pi_{\mathcal{A}}$. This map is the identity when no projection is needed; for bounded coordinates, it clips the perturbed trajectory back to the valid coordinate range.

Remark 5. For Maze2D, it additionally preserves the fixed endpoints used by the inpainting setup and projects interior positions into the valid free space of the maze. $\triangle$

The final data deviation kernel is then the following conditional distribution:

$$D^{\sigma}_{\mathrm{arch}}(\delta \mid x) \quad \text{where} \quad x + \delta = \Pi_{\mathcal{A}}(\widetilde{x}). \tag{31}$$

The functional form of the kernel uses neither samples generated by the model, nor residual directions, nor histograms recovered from samples from the model, nor separately trained predictors.

C.2 Model-Calibrated Comparisons Across Training

In Sec. 5.2, we showed that the kernel closely predicts quantity drift for the final checkpoints of our training runs. However, those models already fit the trajectory data well. Here, we ask whether the same error–drift relationship continues to predict quantity drift across a much wider range of model fitness. To test this, we evaluate all 20 saved checkpoints per synthetic system; each supplies only its aggregate data-space error while the kernel independently supplies the perturbation structure (Fig. 7). Predictions are accurate from undertraining through convergence.

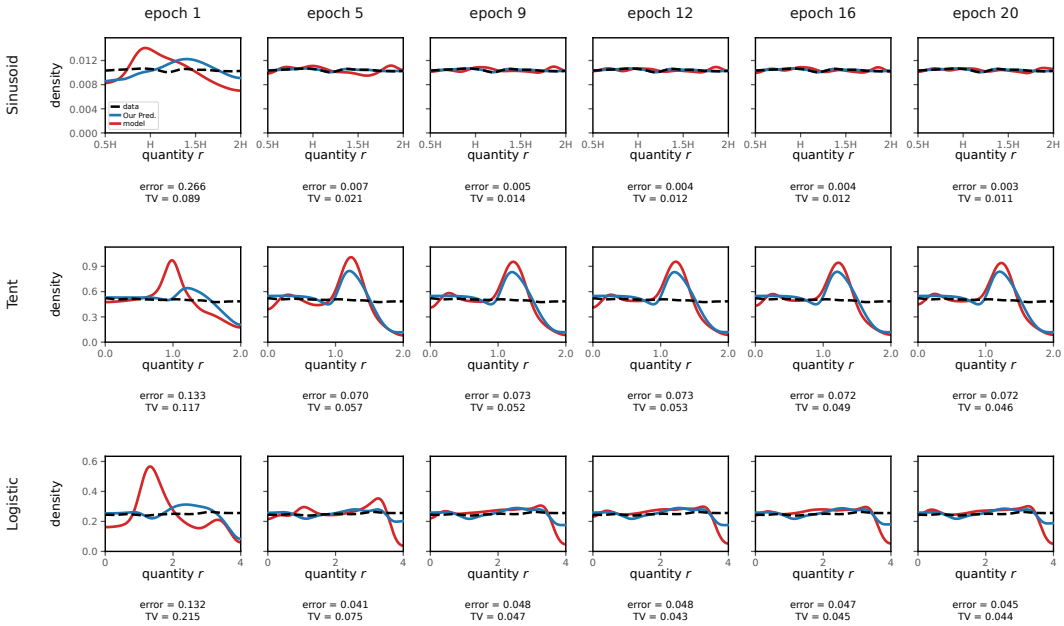
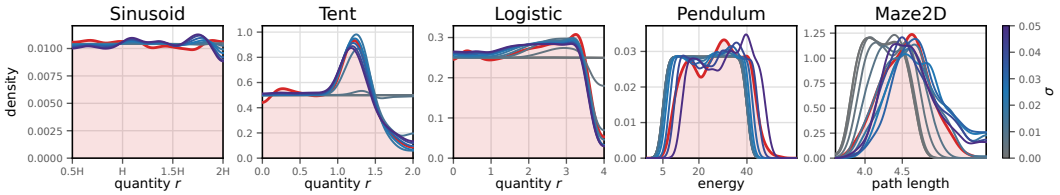


Figure 7: **Model-calibrated predictions across training.** We show six evenly spaced checkpoints, pairing each model marginal with a prediction interpolated from the independent kernel sweep at that checkpoint's measured data-space error. We report this error and the prediction-to-model TV below each panel. Across all 20 checkpoints per system, the median TV [IQR] is 0.013 $[0.012, 0.018]$ for sinusoid, 0.052 $[0.050, 0.055]$ for tent, and 0.046 $[0.044, 0.061]$ for logistic.

C.3 Scale of the Deviations

In Appx. C.2, we matched the kernel's induced data-space error to error of each checkpoint. We next ask how sensitive the predicted quantity distribution is to the choice of deviation scale σ.

For the calibrated synthetic predictions, we first estimate how the model's error magnitude varies over the recovered quantity. The local scale follows this profile, with its overall magnitude calibrated to the aggregate reconstruction error of model samples. Here, we vary σ while holding profiles fixed.



σ	Sin.	Tent	Log.	Pend.	M2D
0	0.012	0.159	0.066	0.163	0.378
.0005	0.012	0.159	0.066	0.163	0.376
.002	0.012	0.159	0.066	0.163	0.356
.0045	0.012	0.159	0.042	0.163	0.245
.008	0.012	0.151	0.015	0.160	0.146
.0125	0.012	0.042	0.017	0.155	0.031
.018	0.013	0.030	0.022	0.139	0.127
.0245	0.015	0.020	0.025	0.113	0.176
.032	0.019	0.019	0.027	0.077	0.177
.0405	0.023	0.030	0.030	0.051	0.123
.05	0.027	0.038	0.033	0.133	0.097

Figure 8: **The deviation scale controls how strongly local trajectory errors are expressed.** For each system, we sweep the kernel's absolute scale σ from zero upward. Here, the solid red line is the trained model. We see that increasing the scale amplifies the redistribution of probability. Notably, the predicted curves are very stable, and the flat baseline gradually morphs into the shape of the actual models' distributions as σ increases. Then, as expected, the prediction is over-amplified and overshoots the target model when σ is set too large. We report the exact TV values of the predicted distribution compared to the corresponding trained model in the table immediately below.

C.4 Model-Free Scale Selection

In Appx. C.3, we observed stable predicted shapes over a range of scales, with the synthetic models' error profiles held fixed. We now ask whether we can also select a single prediction without information from the target model. This is less straightforward because models at different levels of fit need not share one useful predicted quantity distribution. Fortunately, we can select a scale by exploiting the observed stability of kernel-based predictions.

To do this, we vary a common scale across trajectories using only the empirical replacement component, ordering predictions by induced data-space error. As error grows, the sweep passes through three phases: (1) a data-like regime before drift, (2) a stable nontrivial drift shape, and (3) saturation under excessive corruption. We select the first stable basin in phase 2, identified by the first local minimum of a joint score for adjacent changes in normalized signed drift shape and the full predicted marginal (Fig. 9).

System	Kernel-only error [95% CI]	Target error	TV, kernel-only	TV, calibrated
Sinusoid	0.0468 [0.0349, 0.0469]	0.0032	0.023	0.012
Tent	0.0630 [0.0628, 0.0633]	0.0717	0.037	0.019
Logistic	0.0995 [0.0660, 0.0997]	0.0461	0.042	0.017

Table 1: **Kernel-only selection and model-calibrated comparison.** Errors are measured in data space; TV compares predictions with the final model marginals in Fig. 3. We report 95% percentile intervals from 2,000 bootstrap resamples of independent batches, repeating the complete selection procedure for each resample.

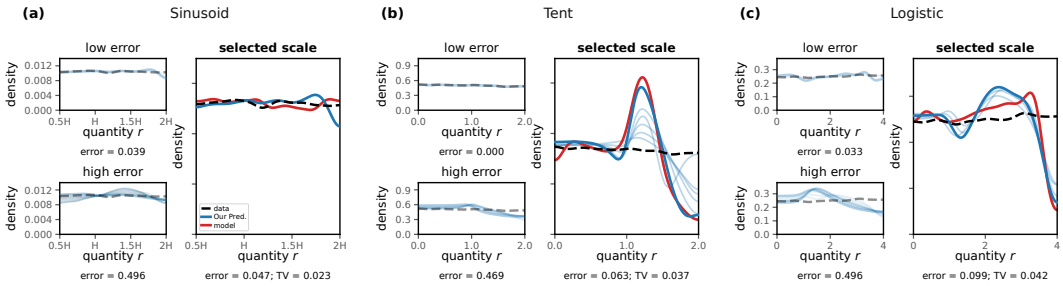


Figure 9: **Predictions across the model-free sweep.** We show all sweep points, grouped into predictions before drift onset, from onset through the selected basin, and beyond that basin. On the right of each group, we highlight the prediction interpolated at the selected scale and overlay the final model marginal; fainter blue curves show the other sweep points in each regime. Below each plot, we report the induced data-space error of the highlighted prediction, averaged across independent batches; for the selected prediction, we also report its TV to the model.

C.5 Gaussian Ablation and Validation Against Model Errors

Having examined how predictions change with model fitness and deviation scale, we next ask what diffusion-specific structure contributes beyond generic noise. As a sanity check, we compare predictions from additive Gaussian noise alone with those from the full kernel at matched data-space error, using the same recovery and model marginals as in Fig. 3.

System	Gaussian TV	Full kernel TV
Sinusoid	0.014	0.013
Tent	0.046	0.020
Logistic	0.050	0.016

Table 2: **Distances between predicted and model marginals at matched data-space error.**

For sinusoid, the two predictions have nearly equal TV; for tent and logistic, we obtain lower TV with the full kernel.

To check whether actual model errors point toward local fragments from the data, we also perform a *post hoc validation* using residuals between denoised and original data trajectories. Within length-3 windows matching the kernel's replacement runs, we measure the *maximum cosine similarity* of each residual to directions toward nearby data fragments at the same locations. We compare model, kernel, and isotropic Gaussian deviations using the same candidate fragments and matching their aggregate root-mean-square magnitudes.

System	Actual	Kernel	Gaussian
Sinusoid	0.962	0.972	0.764
Tent	0.683	0.771	0.577
Logistic	0.258	0.458	0.145

Table 3: **Alignment with directions toward nearby data fragments.** Entries are median maximum cosine similarities.

The results above collectively show that denoising errors point toward local fragments from the data more than isotropic Gaussian noise does.

D Sensitivity of the Synthetic Families

The synthetic families let us compare a learnable baseline against two systems where local trajectory errors can grow along the rollout. Here, we showcase and formally prove this claim.

D.1 State Geometry and Lyapunov Exponents

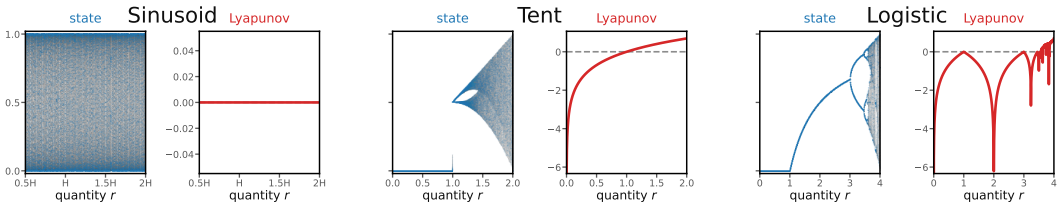


Figure 10: **The synthetic families differ in how local trajectory errors grow along the rollout.** Within each system, the left panel shows the states reached across the range of quantity values, and the right panel shows the corresponding Lyapunov exponent. The sinusoid has no expanding recurrence, whereas the tent and logistic maps include regimes where nearby rollouts separate rapidly; We examine how this state sensitivity relates to prediction residuals in Appx. D.3.

D.2 Lyapunov Proofs for the Synthetic Families

D.2.1 Sinusoidal Family

For the sinusoidal family, a perturbation η_t to phase has identity tangent dynamics, since phase advances by the constant $2\pi/r$ each timestep, $\eta_{t+1} = \eta_t$. Hence,

$$\lambda_H^{\text{sinusoid}} = \frac{1}{H-1} \sum_{t=0}^{H-2} \log 1 = 0. \tag{32}$$

The sinusoid is a zero-growth baseline against which the tent and logistic maps can be compared.

Also, at the fixed phase used to generate data, the mean value theorem gives $|x_t(r') - x_t(r)| \leq (\pi t/32^2)|r' - r|$ for periods $r, r' \in [32, 128]$. Thus, nearby periods also produce nearby trajectories over the fixed horizon.

D.2.2 Tent Map

For a one-dimensional dynamical system $x_{t+1} = f_r(x_t)$, a perturbation η_t evolves according to

$$\eta_{t+1} = f_r'(x_t)\eta_t, \tag{33}$$

so

$$|\eta_{H-1}| = |\eta_0| \prod_{t=0}^{H-2} |f_r'(x_t)|. \tag{34}$$

In the tent map, $f_r(x) = r \min\{x, 1-x\}$ has derivative r on the left branch and $-r$ on the right branch. Thus, for trajectories that do not land exactly on the fold,

$$\lambda_H^{\text{tent}}(r) = \frac{1}{H-1} \sum_{t=0}^{H-2} \log |f_r'(x_t)| = \log r. \tag{35}$$

As r increases above 1, local perturbations are amplified exponentially along the rollout.

D.2.3 Logistic Map

The logistic map follows the same perturbation calculation, but its derivative depends on the state reached along the rollout. Since $f_r(x) = rx(1-x)$, we have $f_r'(x) = r(1-2x)$, and therefore

$$\lambda_H^{\text{logistic}}(r; x_0) = \frac{1}{H-1} \sum_{t=0}^{H-2} \log |r(1-2x_t)|. \tag{36}$$

Unlike the tent map, the orbit itself enters the product, so the exponent need not be monotone in r over the full interval. In the stable regime with a fixed point, $1 < r < 3$, the attracting fixed point is $x^\star = 1 - 1/r$, giving

$$\lambda_\infty^{\text{logistic}}(r) = \log |f_r'(x^\star)| = \log |2 - r| < 0. \tag{37}$$

D.3 Prediction Residuals and Lyapunov Exponents

We find no straightforward relationship between the Lyapunov profiles in Fig. 10 and the recovered quantity distributions in Fig. 3, including when we restrict the tent map to ranges with low or high Lyapunov exponents (Appx. F). Nevertheless, the residual between prediction and model could concentrate at quantity values with higher Lyapunov exponents.

To test this, we compute the pointwise absolute density residual between calibrated prediction and model at every quantity value, rank values by local Lyapunov exponent, divide them into four equal-count groups, and measure each group's mean residual (*i.e.*, Q1 has the lowest exponents; Q4, the highest). We also report the Spearman correlation between residual and exponent.

System	Q1	Q2	Q3	Q4	Spearman ρ [95% CI]
Tent	0.0326	0.0142	0.0189	0.0085	-0.487 [-0.667, -0.129]
Logistic	0.0081	0.0091	0.0067	0.0109	0.141 [-0.419, 0.575]

Table 4: **Prediction residuals grouped by Lyapunov exponent.** Entries in Q1–Q4 are mean absolute density residuals. Intervals are 95% percentile intervals from $2{,}000$ bootstrap resamples of recovery posteriors for individual trajectories.

Tent residuals are non-monotonic, with Q4 below Q1 by 0.0241 and a negative rank correlation. Logistic residuals are also non-monotonic, with Q4 above Q1 by 0.0029 and a weak positive correlation whose interval spans zero. Thus, residuals are not systematically concentrated in regions with high Lyapunov exponents across both systems.

E Variation Across Training Seeds

To check that the prediction and mitigation results are not the byproduct of one favorable training run, we repeat the central experiments with three independent training seeds. For each run, we recover the distribution over the physical quantity and report TV distances using the same evaluation code used for the corresponding figures. Tab. 5 shows that, across the synthetic suite and the two applied physical systems, the distances between predicted and model marginals remain small relative to the scale of the model drift. Tab. 6 also shows that TV from the intended prior is stable for both the baseline model and the mitigation strategies. Across seeds, all our conclusions remain intact.

Metric	Sinusoid	Tent	Logistic	Pendulum	Maze2D
TV(data, prior)	$0.004, 0.004, 0.003$	$0.004, 0.004, 0.003$	$0.004, 0.004, 0.003$	$0.040, 0.040, 0.040$	$0.041, 0.039, 0.039$
TV(model, prior)	$0.026, 0.041, 0.044$	$0.186, 0.193, 0.204$	$0.097, 0.155, 0.133$	$0.164, 0.153, 0.136$	$0.361, 0.431, 0.392$
TV(prediction, prior)	$0.003, 0.005, 0.006$	$0.166, 0.164, 0.170$	$0.088, 0.083, 0.087$	$0.164, 0.119, 0.117$	$0.337, 0.292, 0.354$
TV(prediction, model)	$0.027, 0.040, 0.044$	$0.036, 0.084, 0.056$	$0.032, 0.108, 0.102$	$0.056, 0.057, 0.046$	$0.113, 0.179, 0.139$

Table 5: **Prediction stability across training seeds.** Each cell lists the values from three independent training seeds. The TV(prediction, model) row directly compares the prediction to the distribution recovered from the trained model.

Method	Sinusoid	Tent	Logistic	Pendulum	Maze2D
Baseline	$0.026, 0.041, 0.044$	$0.186, 0.193, 0.204$	$0.097, 0.155, 0.133$	$0.190, 0.170, 0.151$	$0.182, 0.164, 0.258$
Reweight	$0.034, 0.028, 0.058$	$0.217, 0.196, 0.222$	$0.202, 0.157, 0.159$	$0.672, 0.640, 0.422$	$0.115, 0.182, 0.199$
Cond.	$0.007, 0.008, 0.008$	$0.026, 0.016, 0.017$	$0.012, 0.013, 0.014$	$0.200, 0.174, 0.199$	$0.151, 0.161, 0.136$
Transform	$0.026, 0.020, 0.034$	$0.026, 0.021, 0.036$	$0.029, 0.021, 0.043$	$0.029, 0.026, 0.027$	$0.026, 0.023, 0.022$

Table 6: **Mitigation stability across training seeds.** Values are TV distances from the intended prior across three independent training seeds.

Assuming that paired differences across seeds are approximately normally distributed, we also compute one-sided paired t-tests. For prediction, we define

$$d_s^{\text{pred}} = \text{TV}\left(\bar{\pi}^{(s)}_{\text{model}}, \pi\right) - \text{TV}\left(\bar{\pi}^{(s)}_{\text{dev}}, \bar{\pi}^{(s)}_{\text{model}}\right), \tag{38}$$

where s indexes the training seed. For mitigation, we define

$$d_s^{\text{mit}} = \text{TV}\left(\bar{\pi}^{(s)}_{\text{base}}, \pi\right) - \text{TV}\left(\bar{\pi}^{(s)}_{\text{transport}}, \pi\right). \tag{39}$$

In both cases, the one-sided null is $\mathbb{E}[d_s] \leq 0$. Writing $\bar{d}$ for the mean of the three paired differences and s_d for their sample standard deviation, we use

$$t = \frac{\bar{d}}{s_d/\sqrt{3}}, \qquad p = \Pr(T_2 \geq t). \tag{40}$$

System	Prediction p	Transform mitigation p
Sinusoid	0.378	0.118
Tent	0.005	$< 10^{-3}$
Logistic	0.020	0.019
Double-pendulum	0.002	0.003
Maze2D	$< 10^{-3}$	0.013

Table 7: **One-sided paired tests across training seeds.** *Prediction* tests whether the distance between predicted and model marginals is smaller than the model's drift from the intended prior. *Transform mitigation* tests whether the coordinate-transform intervention has lower drift than the baseline model.

These tests show that the accuracy of the prediction, as well as the effectiveness of the mitigation, is stable across seeds. The sinusoid is the expected exception because the baseline model barely drifts; for the other systems, both tests reject the one-sided null at $p < 0.05$.

F Alternative Explanations and Model Architectures

At first glance, one might suspect that physical misgeneralization is a consequence of fairly ordinary causes that are much simpler than the mechanism we proposed in the main text. For example, the training set could be poorly sampled; the model could be too small or trained too briefly; the sampler may be poorly chosen; or, the effect could be specific to diffusion models. For each possibility including but not limited to these aforementioned suspicions, we run an ablation with the tent map, swapping out the relevant part of recovery, data generation, training, sampling, or model choice. Overall, we find that these alternative theories do not account for physical misgeneralization, and thus the perspective put forth by this paper merits broader attention from the community.

Hyp.	Experiment	TV data	TV model
H0	Baseline diffusion model trained on trajectories from the tent map	0.004	0.186
H1a	Recover r using the mean of the posterior	0.018	0.208
H1b	Recover r using a discrepancy based on ℓ_1 distance	0.018	0.183
H1c	Recover r using both changes	0.018	0.183
H2a	Draw values of the quantity uniformly within bins	0.004	0.191
H3a	Train for fewer epochs	0.018	0.195
H3b	Train for more epochs	0.018	0.178
H3c	Repeat the baseline across seeds	0.016	0.185 ± 0.003
H4a	Widen the denoiser	0.018	0.190
H5a	Enlarge the convolutional kernel	0.018	0.194
H5b	Replace the U-Net with an MLP denoiser	0.018	0.449
H6a	Replace 256 DDPM steps with 50 deterministic DDIM steps	0.018	0.204
H7a	Train the denoiser to predict the clean trajectory	0.018	0.361
H8a	Use shorter trajectories	0.018	0.170
H8b	Use longer trajectories	0.018	0.197
H9a	Train an autoregressive Transformer	0.018	0.368
H9b	Train a VAE	0.018	0.950
H10a	Use a lower interval for r	0.015	0.078
H10b	Use a higher interval for r	0.018	0.227

Table 8: **Ablation ladder for the tent map.** For each row, we change one proposed source of the mismatch and report the resulting distance between the distribution recovered over quantity values and the intended prior. Substantial drift remains across variants H1–H9; H10 shows that the geometry of the data meaningfully controls the prevalence of drift, which is consistent with our mechanistic explanation wherein local deviations described by the data deviation kernel become drift after passing through the geometry of the measured quantity.

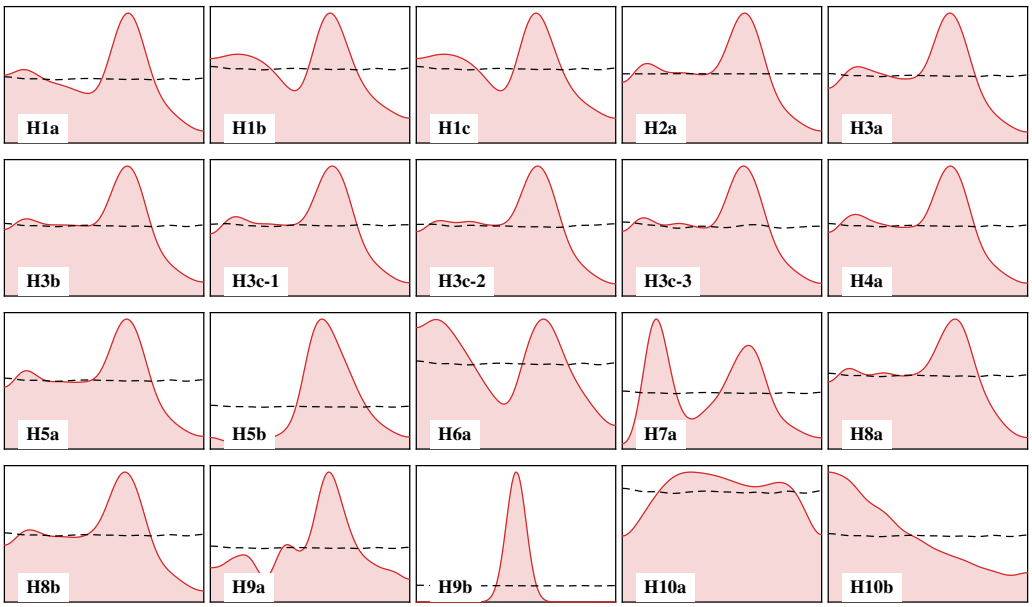


Figure 11: **Alternative explanations do not remove physical misgeneralization.**

$\times$ **Hypothesis H1:** *The rule for recovering the quantity creates the shift.*

To check whether the calculation used for recovery itself creates the shift, we hold the generated trajectories fixed and vary the summary of the posterior and the discrepancy used to recover r. Specifically, we use the posterior mean in place of the posterior mode (H1a), a discrepancy based on ℓ_1 distance in place of the discrepancy based on squared error (H1b), and both changes together (H1c). Yet, across all three variants, the distribution recovered from ground-truth trajectories remains close to the intended prior, while the distribution recovered from generated trajectories remains shifted (Tab. 8, H1a–H1c). We therefore reject H1.

$\times$ **Hypotheses H2–H3:** *The model sees too little of the quantity mixture or trains too little to learn it.*

We next test whether the shift disappears after balancing coverage over the quantity or changing the amount of optimization. To do so, we draw values of the quantity uniformly within 256 bins before generating trajectories from the ground-truth map (H2a), vary the number of epochs used for training from 20 to 6 (H3a) or 60 (H3b), and repeat the baseline across independent seeds (H3c). Sampling quantity values by strata still yields a shifted marginal after generation, changing the number of epochs modulates the severity without restoring the intended mixture, and independent seeds vary only slightly around the baseline value (Tab. 8, H2a–H3c). We therefore reject H2–H3.

$\times$ **Hypothesis H4:** *The denoiser is too small.*

We test whether capacity is the bottleneck by training a wider U-Net (base width 64 instead of 32) while leaving the data, the rule for recovering the quantity, and the sampler fixed. The recovered distribution remains substantially shifted (Tab. 8, H4a). We therefore reject H4.

$\times$ **Hypothesis H5:** *The convolutional denoiser is too local.*

We then test the local structure of the denoiser by enlarging the convolutional kernel from 3 to 7 (H5a), and by replacing the convolutional denoiser with an MLP denoiser that can access the full trajectory at once (H5b). The MLP has three hidden layers of width 1088; we use a learning rate of 10^{-3} and a log-linear noise schedule spanning $[0.005, 3]$. The larger convolutional kernel does not restore the intended mixture, and the MLP denoiser misgeneralizes even more strongly (Tab. 8, H5a–H5b). We therefore reject H5.

$\times$ **Hypotheses H6–H7:** *The sampler or objective used for diffusion creates the shift.*

We isolate two choices in the diffusion implementation by replacing 256 DDPM steps with 50 deterministic DDIM steps (H6a), and by replacing noise prediction with prediction of the clean trajectory x_0 (H7a). Both changes leave the recovered distribution far from the intended prior (Tab. 8, H6a–H7a). We therefore reject H6–H7.

$\times$ **Hypothesis H8:** *The length of the trajectory creates the shift.*

We test whether the horizon is responsible by using shorter ($H = 32$, H8a) and longer ($H = 128$, H8b) trajectories drawn from the tent map in place of $H = 64$. Substantial drift remains at both horizons (Tab. 8, H8a–H8b), although the magnitude changes. We therefore reject H8.

$\times$ **Hypothesis H9:** *Only diffusion models exhibit physical misgeneralization.*

We train an autoregressive Transformer and a VAE on trajectories drawn from the tent map. The Transformer has 4 layers, width 216, 6 attention heads, and a vocabulary of 1024 discretized states; the VAE has a one-dimensional latent variable and two hidden layers of width 1088 for both the encoder and decoder. Both models induce shifted distributions over quantity values (Tab. 8, H9a–H9b); together with the MLP denoiser in H5b, this shows that the failure is not limited to diffusion. We therefore reject H9.

$\checkmark$ **Hypothesis H10:** *Changing the range of the quantity changes the strength of misgeneralization.*

We test which of a lower interval of r ($[0, 0.5]$, H10a) or a higher interval ($[1.5, 2]$, H10b) produces more drift. We find that the lower interval yields a recovered distribution closer to the intended prior, while the higher interval produces larger drift (Tab. 8, H10a–H10b). This suggests that the issue is mediated by differences in how data are arranged with respect to the quantity, consistent with the mechanism we identify in the main body.

Together, these ablation results provide strong evidence against the simpler explanations.

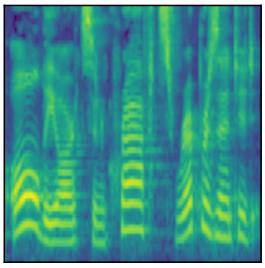
G Text-to-Speech Vignette

In the main text, our focus was on physical sequence models; however, in a set of preliminary experiments, we also found a similar aggregate mismatch in the text-to-speech domain. Here, data are waveforms rather than physical trajectories, and we recover speaking rate rather than path length, energy, or the parameter of a dynamical system. Even so, the structural issue is familiar: the model is trained to synthesize audio from text, while training does not directly specify the distribution over how quickly the text is spoken.

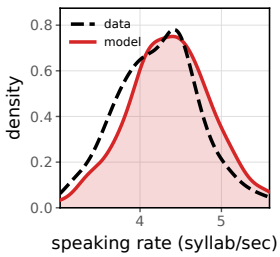
For this investigation, we use the LJSpeech [Ito and Johnson, 2017] dataset, and synthesize matched utterances with the SpeechBrain Tacotron 2 and HiFi-GAN pipeline [Shen et al., 2018, Kong et al., 2020, Ravanelli et al., 2021]. An spectrogram of a generated example utterance is shown in Fig. 12a. For each real and generated utterance, we transcribe the waveform with wav2vec 2.0 [Baevski et al., 2020], count syllables with a deterministic vowel-group heuristic, and divide by waveform duration to recover speaking rate in syllables per second. Since a mismatch in transcript content would make the rate comparison ambiguous, we only keep paired prompts for which the generated utterance has word error rate no worse than the real utterance for the same prompt. Among the 777 paired utterances that pass this filter, we recover a faster speaking-rate distribution from generated speech: the mean speaking rate is 4.356 syllables/sec for generated speech versus 4.222 for data, with $\mathrm{TV} = 0.103$ (Fig. 12b).

This vignette is not part of our predictive claims, but the direction of the mismatch is still mechanistically suggestive. In particular, we find that the mel-spectrogram objective used in this pipeline allows for substantially larger speed-ups than slow-downs at equal mel loss (Fig. 12c). To measure this asymmetry, we take each real utterance, apply a small voiced-frame slow-down, compute the induced mel-spectrogram loss, and then find the speed-up that gives the same loss. Across utterances, the matched speed-ups are about $16.3\times$ larger than the corresponding slow-downs, with interquartile range 11.8–20.3. Thus, in a setting far from the physical systems studied above, we again find that an aggregate quantity changes in the direction that the objective penalizes comparatively weakly.

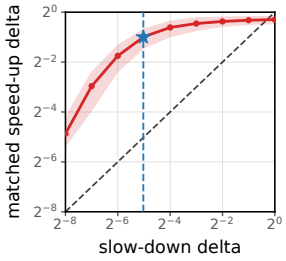
We stop short of making a full prediction, because text-to-speech composes multiple learned stages, and defining a composed kernel for this setup is a nontrivial task that we leave for future work. In particular, Tacotron 2 first produces an acoustic representation, and HiFi-GAN then maps that representation into waveform space. Since we measure speaking rate only after the waveform is produced, a full prediction for this setting would require a data deviation kernel that summarizes how errors of the acoustic model and the vocoder feed into one another. We hope this example encourages follow-ups on how data deviation kernels can be composed across learned stages, so misgeneralization can be anticipated in domains well beyond the physical systems studied here.



(a) Example spectrogram of a model-generated sample.



(b) The model's upwards drift in speaking rate.



(c) Speed-ups & slow-downs at equal mel loss.

Figure 12: **Drift in speaking rate for generated speech.** When we compare real LJSpeech utterances to utterances synthesized from the same text with Tacotron 2 and HiFi-GAN, the generated utterances recover to a faster speaking-rate distribution than the training data implies. The time-warp probe shows that equal mel loss allows much larger speed-ups than slow-downs, pointing to the possibility that this is analogous to physical misgeneralization wherein intrinsic architectural quirks and low-cost directions under the loss get converted into a distribution shift through the physical measurement.