

Learning From Simulators: A Theory of Simulation-Grounded Learning

Carson Dudley, Marisa Eisenberg

September 24, 2025

Abstract

Simulation-Grounded Neural Networks (SGNNs) are predictive models trained entirely on synthetic data from mechanistic simulations. They have achieved state-of-the-art performance in domains where real-world labels are limited or unobserved, but lack a formal underpinning.

We present the foundational theory of simulation-grounded learning. We show that SGNNs implement amortized Bayesian inference under a simulation prior and converge to the Bayes-optimal predictor. We derive generalization bounds under model misspecification and prove that SGNNs can learn unobservable scientific quantities that empirical methods provably cannot. We also formalize a novel form of mechanistic interpretability uniquely enabled by SGNNs: by attributing predictions to the simulated mechanisms that generated them, SGNNs yield posterior-consistent, scientifically grounded explanations.

We provide numerical experiments to validate all theoretical predictions. SGNNs recover latent parameters, remain robust under mismatch, and outperform classical tools: in a model selection task, SGNNs achieve half the error of AIC in distinguishing mechanistic dynamics. These results establish SGNNs as a principled and practical framework for scientific prediction in data-limited regimes.

1 Introduction

A central challenge in scientific machine learning is to construct predictive models in domains where real-world data are scarce, noisy, or unobservable. In public health, for example, accurate estimates of the number of cases in an infectious disease outbreak may not be available until late in the outbreak. In ecology, essential quantities like species population limits in an environment may not be directly observable from collected data. These limitations render standard supervised learning approaches ineffective: when data are limited, empirical learners fail to generalize; when targets are unobservable, learning is altogether impossible.

Simulation-Grounded Neural Networks (SGNNs) were recently developed by our group as a class of predictive models for scientific domains where real-world data are limited or unobservable [1]. Rather than training on empirical observations, SGNNs learn from synthetic datasets generated by mechanistic simulations diversified across parameters, model structures, stochastic processes, and observational artifacts. This paradigm has already yielded state-of-the-art performance across a range of prediction tasks. During early COVID-19, SGNNs nearly tripled forecasting skill relative to the CDC’s median model, despite never being trained on real-world data [2]. In ecology and chemistry, they outperform domain-specific baselines, reduce variance in high-dimensional time-series forecasting, and improve chemical yield prediction with only minutes of pretraining [1].

Despite these empirical successes, the theory of SGNNs remains underdeveloped. When and why do SGNNs work? What function classes become learnable under simulation that are unlearnable from empirical data? How does performance degrade when simulations diverge from reality? And can simulation-trained models provide principled, mechanistic interpretations of their predictions?

This paper develops a mathematical theory of simulation-grounded prediction. We introduce a formal framework in which SGNNs are cast as amortized Bayesian estimators under a simulator-induced prior. Our key results include:

- **Theory of SGNNs.** We formally define Simulation-Grounded Neural Networks (SGNNs) as amortized Bayesian predictors trained over a simulator-induced prior, and prove that they converge to the Bayes-optimal predictor under the synthetic data distribution.

- **Generalization under misspecification.** We derive the first excess-risk bound for SGNNs under model misspecification, decomposing test-time error into learnable and irreducible components. This enables principled evaluation and simulator design in practice.
- **Learning unobservable quantities.** We prove that SGNNs can consistently learn scientific quantities that are unobserved in real-world data, such as the basic reproductive number of a disease, carrying capacity of a population, or source of information diffusion, under standard identifiability conditions.
- **Mechanistic interpretability.** We formalize back-to-simulation attribution, a framework for explaining predictions in terms of the latent simulator parameters that generated them. We prove that attribution yields consistent estimates of the underlying generative mechanisms, and show that with a lightweight alignment objective, it recovers the full posterior distribution over simulator parameters, enabling simulation-grounded explanations and uncertainty quantification.
- **Empirical validation.** We confirm all theoretical predictions using controlled simulators, demonstrating convergence to Bayes predictors, graceful degradation under mismatch, and accurate recovery of latent parameters via attribution.

Together, these results characterize SGNNs as a theoretically principled framework for scientific prediction in data-limited regimes. They establish a class of simulation-learnable functions, clarify the mechanisms by which simulation-grounded learning generalizes, and provide mathematical guarantees for interpretability and inference in settings where empirical approaches break down.

Related Prior Work

In simulation-based inference (SBI) and likelihood-free inference (LFI) [3], neural networks are used to approximate statistical functions like posterior distributions, likelihoods, or likelihood ratios to enable parameter inference. SGNNs take a different approach: they learn task-specific predictors directly via supervised training on synthetic data, whether the task is parameter inference, future prediction, or classification.

SGNNs also relate to physics-informed neural networks (PINNs) [4], neural operators [5], and hybrid mechanistic-machine learning models [6], all of which integrate scientific knowledge into model architectures, optimization constraints, or inductive biases. These approaches typically assume access to labeled real-world data and use mechanistic priors to improve generalization. In contrast, SGNNs embed domain knowledge into the training data itself: they rely entirely on simulation and are designed to operate even when labeled observations are unavailable or unobservable.

The simulation-grounded learning paradigm has emerged across scientific domains without a unified theoretical framework. In epidemiology, for instance, early work like DEFSI trained neural networks on agent-based mechanistic simulations for influenza forecasting. [7] A similar practice known as “sim2real” is common in robotics, where models are trained or evaluated in simulated environments and deployed on real systems [8]. The SGNN concept unifies these domain-specific applications under a single paradigm, and this paper provides a formal theoretical grounding.

The closest conceptual relative is the class of Prior-Data Fitted Networks (PFNs) [9, 10, 11], which learn amortized Bayesian predictors via training on synthetic tasks sampled from a prior. SGNNs extend this idea to scientific domains, where the prior is defined implicitly by a mechanistic simulator. This induces structured, non-exchangeable data with latent parameters and domain-specific noise, and allows SGNNs to learn mappings that are grounded in physical or biological processes.

We develop new results that quantify error under model misspecification and prove consistency of simulation-based attribution mechanisms. To do so, we draw from classical learning theory, particularly in the study of excess risk and generalization under distribution shift [12].

2 Definitions and Formal Setup

We formalize simulation-grounded learning as a framework in which predictive models are trained exclusively on synthetic data generated from a structured scientific simulator. This section introduces the key mathematical objects, generative assumptions, and notation that underlie our theoretical results.

2.1 Simulators as Generative Models

Let Θ denote a space of latent scientific configurations, including both parameters (e.g., reaction constants in chemistry) and structural choices (e.g., model type, presence of mechanisms). Each $\theta \in \Theta \subseteq \mathbb{R}^d$ parameterizes a *scientific simulator*, which we define as the composition of two components:

- A **mechanistic model** $\mathcal{M}(\theta)$, which governs the latent system dynamics (e.g., transmission processes, ecological interactions, chemical reactions). This model typically consists of differential equations, discrete stochastic rules, or agent-based systems.
- An **observation model** \mathcal{O} , which maps latent system states to observed quantities, introducing realistic noise, reporting artifacts, or partial observation (e.g., delays, rounding, censoring).

Definition 1 (Mechanistic Simulator). Let $\theta \in \Theta \subseteq \mathbb{R}^d$ denote a vector of simulator parameters, which may include both continuous and discrete components. The mechanistic simulator \mathcal{S} is defined as the composition:

$$\mathcal{S} := \mathcal{O} \circ \mathcal{M},$$

where $\mathcal{M}(\theta)$ is a mechanistic model (e.g., differential equations, stochastic process) and \mathcal{O} is an observation model that introduces noise, partial observation, or reporting artifacts. For a given θ , the simulator $\mathcal{S}(\theta)$ stochastically generates a sample $(x, y) \in \mathcal{X} \times \mathcal{Y}$, where x is the observed input and y is a task-specific target.

We assume θ is drawn from a prior distribution $P(\theta)$ that encodes domain knowledge or structural uncertainty. The overall generative process is:

$$\theta \sim P(\theta), \quad w \sim \mathcal{M}(\theta), \quad x = \mathcal{O}(w), \quad y = T(\theta),$$

where w denotes latent system trajectories and $T : \Theta \rightarrow \mathcal{Y}$ is a task-specific labeling function that defines the prediction target (e.g., future dynamics, class label, latent parameter). In some tasks, $T(\theta)$ may represent a deterministic label (e.g., a class label or model parameter), while in others (e.g., forecasting) it may be stochastic, depending on additional process noise shared with the input x . We allow for both cases, and in general treat y as drawn from a conditional distribution $p(y | \theta)$.

2.2 Synthetic Data and Simulation-Grounded Prediction

Let \mathcal{X} denote the space of observable inputs $x = \mathcal{S}(\theta)$, and let \mathcal{Y} denote the space of targets $y = T(\theta)$. The simulator and labeling function together induce a joint distribution D_{syn} over $\mathcal{X} \times \mathcal{Y}$. Then:

$$x = \mathcal{S}(\theta), \quad y = T(\theta), \quad \text{so that } (x, y) \sim D_{\text{syn}}.$$

Let $\mathcal{D}_{\text{syn}} = \{(x_i, y_i)\}_{i=1}^N$ be a dataset of N independent samples from this distribution.

Definition 2 (Simulation-Grounded Neural Network (SGNN)). A Simulation-Grounded Neural Network is a predictor $f_\phi : \mathcal{X} \rightarrow \mathcal{Y}$ trained to minimize supervised loss on synthetic data:

$$\min_{\phi} \mathbb{E}_{(x,y) \sim D_{\text{syn}}} [\ell(f_\phi(x), y)],$$

where $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}_{\geq 0}$ is a prescribed loss function (e.g., mean squared error, cross-entropy).

In practice, we restrict attention to a model class \mathcal{F} (e.g., neural networks of a given architecture), with SGNN predictors $f_\phi \in \mathcal{F}$.

Unlike empirical learners, which rely on real-world labeled data, SGNNs are trained entirely on simulated examples and are deployed directly on real data. Figure 1 illustrates this core workflow: parameters θ sampled from a scientific prior drive mechanistic simulations to generate synthetic training pairs (x, y) , where x represents realistic observations and $y = T(\theta)$ captures the target scientific quantity of interest.

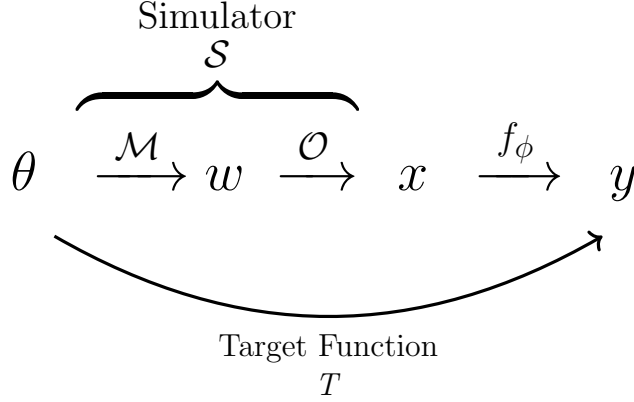


Figure 1: **Simulation-grounded learning schematic.** Parameters θ sampled from a prior $P(\theta)$ are fed into a mechanistic model \mathcal{M} (e.g., differential equations) to generate latent system dynamics w . An observation model \mathcal{O} then transforms these into realistic observed data x by adding noise, delays, and other artifacts. Together, \mathcal{M} and \mathcal{O} form the complete simulator $\mathcal{S} = \mathcal{O} \circ \mathcal{M}$. The SGNN f_ϕ learns to map from observations x to target quantities $y = T(\theta)$, where T can be simulator parameters, future trajectories, etc.

2.3 Model Misspecification and Real-World Generalization

At test time, the SGNN is applied to inputs x^* drawn from a true data-generating distribution D_{real} , which may differ from D_{syn} due to modeling errors, structural misspecification, or prior mismatch.

Definition 3 (Model Misspecification). Model misspecification refers to the discrepancy between the synthetic distribution D_{syn} and the real-world distribution D_{real} . This mismatch may arise from:

- misspecified priors $P(\theta)$,
- missing mechanisms in the mechanistic model \mathcal{M} ,
- or inaccuracies in the observation model \mathcal{O} .

Quantifying the effect of model misspecification on SGNN performance is a primary goal of this paper.

3 SGNNs as Amortized Bayesian Predictors

With the formal setup in place, we now recast simulation-grounded prediction as an instance of *amortized Bayesian inference*.

Amortized vs. standard inference. In standard Bayesian inference, one computes a posterior distribution $p(\theta | x)$ separately for each new observation x , typically via sampling or optimization. This is accurate but computationally intensive: inference must be repeated from scratch for every input. Amortized inference, by contrast, learns a single global function $f_\phi(x)$ that maps any input x directly to a posterior quantity of interest (e.g., an expectation or a parameter estimate). The cost of inference is “amortized” across many training examples: expensive computation is done once up front, after which predictions on new data are nearly instantaneous.

The Bayes-optimal predictor. Under the synthetic distribution D_{syn} , each training pair is generated by first drawing parameters $\theta \sim P(\theta)$, then producing input $x = \mathcal{S}(\theta)$ and target $y = T(\theta)$. The Bayes-optimal predictor is the function that, for each input x , outputs the expected target value under the posterior:

$$f^*(x) = \mathbb{E}_{\theta \sim P(\theta) | x}[T(\theta)] = \mathbb{E}[y | x].$$

In words, $f^*(x)$ is the predictor that minimizes expected loss: it matches the target y *in expectation*, given the information available in x .

3.1 The Bayes-Optimal Predictor under the Simulation Distribution

Intuitively, the Bayes-optimal predictor is the function that makes the fewest mistakes on average: for any input x , it outputs the conditional expectation of the target y given x . Formally, let $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}_{\geq 0}$ be a loss function, and define the population risk of a predictor f under the synthetic distribution as

$$R_{\text{syn}}(f) = \mathbb{E}_{(x,y) \sim D_{\text{syn}}} [\ell(f(x), y)].$$

When the loss is squared error, $\ell(f(x), y) = \|f(x) - y\|^2$, the unique minimizer of this risk is

$$f^*(x) = \mathbb{E}[y \mid x].$$

Our central question is whether an SGNN trained on samples from D_{syn} can approximate this Bayes-optimal predictor, and under what conditions it converges to it.

3.2 SGNNs as Approximators of the Bayes-Optimal Predictor

An SGNN f_ϕ is trained to minimize empirical risk over a synthetic dataset drawn from D_{syn} . A central question is how the expected performance of this learned predictor, $R_{\text{syn}}(f_{\phi_N})$, compares to the Bayes-optimal risk $R_{\text{syn}}(f^*)$, where N denotes the size of the training dataset.

3.2.1 Finite-Sample Generalization Bound

The *excess risk* of the learned model,

$$R_{\text{syn}}(f_{\phi_N}) - R_{\text{syn}}(f^*),$$

can be decomposed into two core components:

1. **Approximation error** ($\mathcal{E}_{\text{approx}}$): The gap between the Bayes-optimal predictor and the best function within the chosen model class \mathcal{F} . This reflects the expressiveness of the architecture.
2. **Estimation error** (\mathcal{E}_{est}): The gap between the empirical risk minimizer f_{ϕ_N} and the best-in-class function, arising from training on a finite sample of size N .

This decomposition can be written formally as:

$$R_{\text{syn}}(f_{\phi_N}) - R_{\text{syn}}(f^*) = \underbrace{\left(\inf_{f \in \mathcal{F}} R_{\text{syn}}(f) - R_{\text{syn}}(f^*) \right)}_{\text{approximation error}} + \underbrace{\left(R_{\text{syn}}(f_{\phi_N}) - \inf_{f \in \mathcal{F}} R_{\text{syn}}(f) \right)}_{\text{estimation error}}.$$

The approximation error can be made arbitrarily small with a sufficiently rich model class (e.g., deep neural networks). The estimation error admits a finite-sample bound via generalization tools such as Rademacher complexity [13].

Empirical-risk minimizer. To analyze the behavior of SGNNs, we consider the standard empirical risk minimization (ERM) framework. Assume the learned predictor f_{ϕ_N} is obtained by minimizing empirical loss over a synthetic dataset $\mathcal{D}_{\text{syn}} = \{(x_i, y_i)\}_{i=1}^N$ drawn i.i.d. from D_{syn} :

$$f_{\phi_N} \in \arg \min_{f \in \mathcal{F}} \frac{1}{N} \sum_{i=1}^N \ell(f(x_i), y_i),$$

where \mathcal{F} is the function class (e.g., a neural network architecture) and ℓ is the loss function.

This raises a natural question: *How does the performance of the finite-sample predictor f_{ϕ_N} compare to the ideal Bayes predictor f^* ?*

The following theorem formalizes the answer, showing that the total excess risk decomposes cleanly into an approximation term (dependent on how expressive the model class is) and an estimation term (dependent on the dataset size N and complexity of \mathcal{F}).

Theorem 1 (Finite-sample excess-risk bound). *Let the loss $\ell : \mathbb{R} \times \mathcal{Y} \rightarrow [0, B]$ be convex in its first argument (a condition satisfied by standard choices such as mean squared error, cross-entropy, and quantile loss commonly used in SGNN training), L -Lipschitz in that argument, and bounded by $B > 0$. With probability at least $1 - \delta$ over the i.i.d. draw of the synthetic dataset $\mathcal{D}_{\text{syn}}^{(N)}$, the ERM predictor f_{ϕ_N} satisfies:*

$$R_{\text{syn}}(f_{\phi_N}) - R_{\text{syn}}(f^*) \leq \underbrace{\inf_{f \in \mathcal{F}} R_{\text{syn}}(f) - R_{\text{syn}}(f^*)}_{\text{approximation error}} + \underbrace{4L\widehat{\mathfrak{R}}_N(\mathcal{F}) + 6B\sqrt{\frac{\log(2/\delta)}{2N}}}_{\text{estimation error}}$$

where $\widehat{\mathfrak{R}}_N(\mathcal{F})$ is the empirical Rademacher complexity of the function class \mathcal{F} .

Proof deferred to Appendix A.

Interpretation. This bound shows that SGNNs trained via ERM will approach Bayes-optimal performance provided two conditions hold:

- **Expressivity.** The hypothesis class \mathcal{F} must be rich enough to approximate f^* (driving the approximation error toward zero).
- **Sample size vs. complexity.** As the number of synthetic samples N grows, the estimation error decreases at a rate governed by the complexity of \mathcal{F} , here captured by the empirical Rademacher complexity $\widehat{\mathfrak{R}}_N(\mathcal{F})$. Larger classes need more samples to achieve the same accuracy.

In short, SGNNs converge to the Bayes predictor when the architecture is sufficiently expressive and trained on enough synthetic data relative to its capacity. Both conditions are practical to satisfy in simulation-grounded learning: (1) large neural networks are universal approximators, and (2) synthetic data can be generated in arbitrarily large quantities. Together, this establishes a path to consistency: SGNNs can match the performance of the Bayes-optimal predictor given sufficient model capacity and synthetic training data.

3.2.2 Consistency and Practical Considerations

The finite-sample bound above shows that SGNNs trained via empirical risk minimization approach the Bayes-optimal predictor as two conditions are met: (1) the model class is sufficiently expressive, and (2) enough synthetic data is used. The following proposition formalizes this intuition by establishing consistency in the limit.

Proposition 1 (Consistency of the SGNN Estimator). *Let the SGNN function class \mathcal{F} be a universal approximator for the Bayes-optimal predictor f^* , so that the approximation error vanishes: $\inf_{f \in \mathcal{F}} R_{\text{syn}}(f) = R_{\text{syn}}(f^*)$. Further assume that the complexity of \mathcal{F} is controlled such that the empirical Rademacher complexity vanishes with sample size:*

$$\lim_{N \rightarrow \infty} \widehat{\mathfrak{R}}_N(\mathcal{F}) = 0.$$

Then, as the synthetic dataset size $N \rightarrow \infty$, the ERM predictor f_{ϕ_N} is consistent:

$$\lim_{N \rightarrow \infty} R_{\text{syn}}(f_{\phi_N}) = R_{\text{syn}}(f^*).$$

Moreover, for squared-error loss, this convergence in risk is equivalent to convergence in the $L^2(\mathcal{D}_x)$ norm:

$$R_{\text{syn}}(f) - R_{\text{syn}}(f^*) = \mathbb{E}_{x \sim \mathcal{D}_x} \left[(f(x) - f^*(x))^2 \right],$$

which implies

$$\lim_{N \rightarrow \infty} \mathbb{E}_{x \sim \mathcal{D}_x} \left[(f_{\phi_N}(x) - f^*(x))^2 \right] = 0.$$

Practical Optimization. The analysis above assumes that f_{ϕ_N} is the exact empirical risk minimizer. In practice, stochastic optimization yields an approximate solution \tilde{f}_{ϕ_N} , introducing an additional *optimization error*, $\mathcal{E}_{\text{opt}} = R_{\text{syn}}(\tilde{f}_{\phi_N}) - R_{\text{syn}}(f_{\phi_N})$. While typically small in modern training regimes, this third source of error contributes to the total excess risk. Regularization techniques (e.g., dropout, weight decay) also play a dual role by reducing estimation error via complexity control.

Remark 1 (How SGNNs Achieve Bayesian Optimality). SGNNs do not compute the posterior $p(\theta \mid x)$ explicitly, nor do they perform traditional Bayesian inference. Instead, they achieve Bayes-optimal prediction through standard supervised learning.

The key mechanism is this: the minimizer of mean squared error over a distribution is the conditional expectation of the target given the input. For simulation-grounded data, this conditional expectation is

$$f^*(x) = \mathbb{E}[y \mid x] = \mathbb{E}_{\theta \sim p(\theta \mid x)}[\mu(x, \theta)],$$

where $\mu(x, \theta) = \mathbb{E}[y \mid x, \theta]$ is the simulator’s response. Here the likelihood and posterior appear implicitly: the simulator defines a likelihood $p(x, y \mid \theta)$ through its mechanistic and observation components, and when combined with the prior $P(\theta)$ this induces the posterior $p(\theta \mid x)$. Minimizing MSE on synthetic samples therefore coincides with learning the posterior expectation of y given x , i.e., the Bayes-optimal predictor.

In short, the SGNN performs implicit Bayesian inference simply by fitting the right data with the right loss.

3.3 Experiment: SGNNs Approximate the Bayes-Optimal Predictor

We empirically validate that SGNNs perform amortized inference toward the Bayes-optimal predictor $f^*(x) = \mathbb{E}[\theta \mid x]$ under a known generative model. Specifically, we test whether an SGNN can recover the latent simulator parameters $\theta = (\alpha, \beta)$ from observed trajectories x , and compare its performance to a kernel-based Monte Carlo approximation of f^* .

Simulator. We use a diagonal linear dynamical system (LDS) with additive Gaussian noise:

$$x_{t+1} = A(\theta)x_t + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \sigma^2 I), \quad x_0 = [1, 1]^\top,$$

where the latent parameters $\theta = (\alpha, \beta)$ define

$$A(\theta) = \begin{bmatrix} \alpha & 0 \\ 0 & \beta \end{bmatrix}, \quad \theta \sim \mathcal{U}([0.5, 1.5]^2).$$

Each simulation yields a trajectory of length $T = 10$, flattened into a vector $x \in \mathbb{R}^{20}$. The target is the latent mechanism: $y = T(\theta) = \theta$.

Our analysis assumes the conditional mean $\mathbb{E}[y \mid x]$ is well defined, as is the case under Gaussian noise and other finite-variance distributions. For extremely heavy-tailed noise (e.g., Cauchy), the mean may not exist. In this setting, we can instead train with a robust convex loss (e.g., ℓ_1 or Huber), in which case the Bayes predictor becomes the corresponding posterior median or quantile.

Task and Baseline. The prediction task is to learn $f(x) \approx \mathbb{E}[\theta \mid x]$. We compare the SGNN to a nonparametric Monte Carlo estimator that reweights 10,000 reference samples with kernel-smoothed weights:

$$\hat{f}^*(x) = \sum_i w_i(x) \theta_i, \quad w_i(x) \propto \exp\left(-\frac{\|x - x_i\|^2}{2\sigma^2}\right),$$

with σ^2 set to the median pairwise distance between reference samples. This estimator provides a direct, nonparametric approximation to the Bayes predictor $f^*(x)$ using only the simulator and prior. Unlike MCMC-based inference, it requires no additional modeling choices or convergence diagnostics, but it is bandwidth-sensitive, suffers from finite-sample noise, and must recompute weights for each new input. These limitations underscore the advantage of SGNNs: once trained, they perform amortized inference in a single forward pass.

SGNN Model. We train an SGNN on 10^7 synthetic pairs (x, θ) using mean squared error loss. The model amortizes inference, learning a global map from trajectories to latent parameters.

Evaluation. On a 50,000-sample test set, we measure:

- **Bayes convergence:** MSE between SGNN predictions and $\hat{f}^*(x)$.
- **Parameter recovery:** MSE between SGNN predictions and true parameters θ .

Results. Figure 2 shows that SGNNs converge toward the Bayes-optimal predictor as training data increases (left). While the kernel estimator serves as our Monte Carlo approximation of f^* , it suffers from finite-sample variance and bias. The SGNN not only approaches this reference but achieves lower parameter estimation error than the kernel baseline (right), despite the latter’s access to ground-truth simulator samples. This apparent outperformance arises because the SGNN amortizes inference across millions of samples, enabling smoother, more accurate generalization than the nonparametric estimator. These results demonstrate a practical advantage of SGNNs: in finite-sample settings, they can surpass standard approximators of the Bayes-optimal predictor by leveraging simulation-scale training and parametric inductive biases.

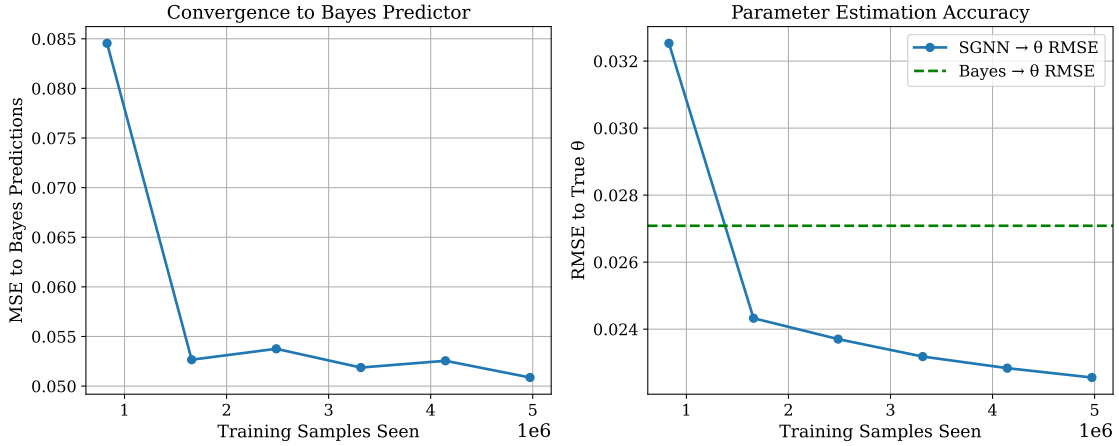


Figure 2: **SGNNs approximate the Bayes-optimal predictor.** **Left:** MSE between SGNN and Monte Carlo estimate of $f^*(x)$. **Right:** MSE between SGNN and ground-truth θ . SGNNs outperform the baseline (dashed green line) due to amortized inference and smooth generalization.

4 Generalization Under Model Misspecification

The preceding section established that, under the synthetic distribution D_{syn} , an SGNN converges to the Bayes-optimal predictor. We now quantify how its performance on a real-world test distribution, D_{real} , is affected by model misspecification, where $D_{real} \neq D_{syn}$. Our goal is to bound the excess risk of the SGNN on D_{real} in terms of (i) its performance on the synthetic distribution and (ii) a formal divergence measuring the gap between the real and synthetic worlds.

4.1 Formal Problem Setting

4.1.1 Distributions and Risks

Let D_{syn} be the synthetic distribution over pairs (x, y) and let D_{real} be the true data-generating distribution at test time. For a loss function $\ell : Y \times Y \rightarrow [0, L_{max}]$ bounded by L_{max} , we define the population risks as $R_{syn}(f) \triangleq \mathbb{E}_{(x,y) \sim D_{syn}}[\ell(f(x), y)]$ and $R_{real}(f) \triangleq \mathbb{E}_{(x,y) \sim D_{real}}[\ell(f(x), y)]$. The respective Bayes-optimal predictors are $f_{syn}^* \triangleq \operatorname{argmin}_f R_{syn}(f)$ and $f_{real}^* \triangleq \operatorname{argmin}_f R_{real}(f)$.

4.1.2 SGNN Predictor

The SGNN f_{ϕ_N} is the empirical risk minimizer over a synthetic training set $D_{syn}^{(N)}$ of size N :

$$\phi_N = \operatorname{argmin}_{\phi \in \Phi} \frac{1}{N} \sum_{(x_i, y_i) \in D_{syn}^{(N)}} \ell(f_{\phi}(x_i), y_i). \quad (1)$$

We assume the model family Φ has universal approximation capacity and that f_{ϕ_N} is a consistent estimator of f_{syn}^* .

4.2 Excess-Risk Bound under Mismatch

We measure the discrepancy between distributions using the Total Variation (TV) distance, which is symmetric and well-suited for bounding differences in expectations.

Definition 4 (Total Variation Mismatch). The model misspecification is defined as the Total Variation distance between the real and synthetic joint distributions:

$$\Delta_{TV} \triangleq \sup_{A \subseteq \mathcal{X} \times \mathcal{Y}} |D_{real}(A) - D_{syn}(A)|.$$

Theorem 2 (Generalization Bound for Model Misspecification). *The excess risk of the SGNN f_{ϕ_N} on the real distribution is bounded by the sum of its excess risk on the synthetic distribution and a penalty term proportional to the level of misspecification:*

$$R_{real}(f_{\phi_N}) - R_{real}(f_{real}^*) \leq \underbrace{R_{syn}(f_{\phi_N}) - R_{syn}(f_{syn}^*)}_{\text{Synthetic Excess Risk}} + \underbrace{2L_{max}\Delta_{TV}}_{\text{Mismatch Penalty}}.$$

Proof deferred to Appendix A.

4.3 Interpretation and Practical Implications

Theorem 2 provides a clear and powerful decomposition of the SGNN’s test-time generalization error. The performance gap is attributable to two independent sources:

1. **Synthetic Excess Risk:** This term, $R_{syn}(f_{\phi_N}) - R_{syn}(f_{syn}^*)$, measures how well the SGNN has learned to solve the task *in the simulated world*. It is the sum of approximation error (is the model expressive enough?) and optimization error (did training find good parameters?). Under consistency assumptions, this term vanishes as the synthetic dataset size $N \rightarrow \infty$. It can be readily estimated using a held-out set of synthetic validation data.
2. **Misspecification Penalty:** This term, $2L_{max}\Delta_{TV}$, is a penalty determined solely by the intrinsic divergence between the simulator and reality. It is independent of the SGNN’s architecture, training data volume, or optimization success. This bound highlights two key levers for improving real-world performance:
 - **Improving the Simulator:** The primary way to reduce Δ_{TV} is by making the simulator more realistic. Increasing mechanistic diversity, adding observed real-world noise patterns, or refining the prior $P(\theta)$ all serve to close the gap between D_{syn} and D_{real} , directly reducing this penalty.
 - **Bounding Misspecification:** While Δ_{TV} is unknown, domain experts can often place a conservative upper bound on it by combining empirical checks (e.g., comparing simulated and real distributions of reporting delays, noise, or other observable artifacts) with scientific knowledge of plausible parameter ranges and mechanism validity. This provides a defensible estimate of the maximum simulator–reality gap. Practitioners can then calculate a safety margin for deployment by measuring the synthetic excess risk and adding this conservative misspecification penalty.

This framework clarifies that improving real-world performance requires both effective learning on the synthetic data (to reduce the first term) and high-fidelity simulation (to reduce the second).

4.4 Estimating Misspecification in Practice

While the misspecification Δ_{TV} between the full joint distributions cannot be calculated directly (as D_{real} is unknown), its value can be reasoned about by decomposing the sources of misspecification. This makes Δ_{TV} a powerful conceptual tool for model design and evaluation. Misspecification arises from three areas: the prior $P(\theta)$, the mechanistic model \mathcal{M} , and the observation model \mathcal{O} .

- **Observation Misspecification (\mathcal{O}):** This is the most tractable component to estimate. We can often directly measure real-world observational artifacts. For instance, we can analyze a sample of real data to estimate the empirical distribution of reporting delays, the prevalence of weekend reporting dips, or the signal-to-noise ratio of a sensor. We can then quantify the divergence between these empirical distributions and their simulated counterparts, providing a lower bound on Δ_{TV} .
- **Prior Misspecification ($P(\theta)$):** This reflects a mismatch in the assumed distribution of scientific parameters. While the true distribution is unknown, we can assess robustness through *sensitivity analysis*. By systematically varying the prior $P(\theta)$ and observing the resulting change in D_{syn} , we can understand how sensitive our system is to prior assumptions. If D_{syn} is stable under different plausible priors, the mismatch penalty is likely to be less severe. This can also be informed by consulting domain experts to ensure the support of $P(\theta)$ covers all scientifically plausible parameter regimes.
- **Mechanistic Misspecification (\mathcal{M}):** This is the most challenging source, since it includes both (i) uncertainty about which mechanisms already present in the simulator are essential, and (ii) the possibility that key mechanisms are entirely absent. For (i), ablation studies are informative: training models on simulators with and without particular mechanisms reveals which components are critical: if removing one causes a large drop in real-data performance, it must be retained to keep Δ_{TV} small. For (ii), missing mechanisms cannot be diagnosed by ablation alone. Instead, they may be detected through systematic residuals (where SGNN predictions consistently deviate from real data across simulations), domain expertise (experts know a process should exist, such as asymptomatic transmission), or failure modes (SGNN performs well in most regimes but breaks down in specific conditions). These signals indicate that the simulator family is insufficient and that Δ_{TV} is fundamentally bounded away from zero unless the missing mechanism is added.

Comparison with Traditional Mechanistic Modeling. A key difference from classical mechanistic modeling is how additional mechanisms are handled. In traditional modeling, lack of parsimony can cause model fitting to break down: over-specified models lead to unidentifiability, unstable inference, and inflated uncertainty, making it necessary to apply model selection criteria (e.g., AIC) or regularization methods (e.g., LASSO) to prune mechanisms [14, 15, 16]. By contrast, SGNNs can be trained on a broad family of mechanisms simultaneously. As long as training covers the true mechanisms and sufficient synthetic data is generated, the network learns to prioritize the dynamics that matter for each individual case. This reduces the need for aggressive parsimony, shifting the focus from hand-selecting a minimal model to ensuring adequate mechanistic coverage in the simulator prior.

4.5 Illustrative Cases of Misspecification

We now examine how the generalization bound in Theorem 2 manifests in practical settings. In particular, we decompose the total excess risk into the contributions of synthetic learning error and simulator–reality mismatch, and demonstrate how this decomposition provides theoretical insight into empirical failure modes.

Case 1: Robustness Under Slight Prior Misspecification. Assume the simulator encodes the correct generative mechanisms and noise profiles but assigns prior mass $P(\theta)$ that differs modestly from the true generative frequencies.

- **Mismatch Penalty:** Moderate but controlled. The support of D_{syn} still substantially overlaps with that of D_{real} , though the mass is unevenly distributed.

- **Synthetic Excess Risk:** Low. The SGNN successfully learns the Bayes-optimal predictor under D_{syn} .
- **Implication:** The SGNN exhibits robust generalization, consistent with the bound in Theorem 2. Because $f^*(x)$ integrates over the posterior $p(\theta | x)$, the learned function can extrapolate beyond high-density regions of the simulator prior, provided that those regions are not entirely excluded. This case illustrates robust generalization under mild mismatch, which we empirically validate in Section 4.6 and which has been observed in practice [2].

Case 2: Overparameterized Simulator with Irrelevant Mechanisms. In an attempt to increase realism, the simulator includes a broad set of mechanisms, many of which do not affect the target task in practice.

- **Mismatch Penalty:** Low. Since the true data-generating process lies within the support of the synthetic distribution, the total variation distance Δ_{TV} remains small.
- **Synthetic Excess Risk:** High. The task is more complex: identifying task-relevant structure requires disentangling signal from a large and potentially confounding mechanism space. For finite model capacity and training budget N , convergence to f_{syn}^* may not be achieved.
- **Implication:** The generalization bound is dominated by the excess synthetic risk. Even a “correct” simulator in the support sense may yield poor real-world performance if the induced prediction task is intractable. This highlights the dual requirement of representational sufficiency and statistical efficiency.

Case 3: Misspecification via Underparameterized Simulator. Consider a simulator that omits key stochastic or mechanistic components (e.g., generates noise-free data or ignores time-varying effects), or the simulator priors are too restrictive on the parameters, while the real world exhibits significant variability.

- **Synthetic Excess Risk:** Low. The SGNN achieves near-optimal performance on the simplified synthetic distribution; i.e., $f_{\phi_N} \approx f_{\text{syn}}^*$.
- **Mismatch Penalty:** High. The total variation distance Δ_{TV} between the real distribution D_{real} and the synthetic distribution D_{syn} is large, as critical real-world stochasticity is absent from the simulator.
- **Implication:** The total generalization bound is dominated by the mismatch term. Although the SGNN performs well on synthetic data, it generalizes poorly due to a failure to account for real-world noise processes.

Case 4: Fundamental Misspecification. Here the simulator is not merely too simple, it is fundamentally incorrect. An essential mechanism is absent or the parameter bounds are significantly incorrect, so the real data are generated by a process outside the simulator’s model class. Parameter diversity or prior tuning cannot fix a class error: the true distribution lies outside the support of D_{syn} , Δ_{TV} stays strictly positive, and the mismatch penalty persists. Remedy requires adding missing mechanisms or increasing parameter ranges (expanding the model class), not more data or different priors.

- **Synthetic Excess Risk:** Low, since the SGNN can still solve the task within the synthetic world.
- **Mismatch Penalty:** Fundamentally high, since there is a systematic deviation between D_{syn} and D_{real} .
- **Implication:** Real-world performance will remain poor unless the simulator is corrected. Simulation-grounded learning cannot compensate for a fundamentally misspecified scientific model.

Summary. These examples clarify the distinct roles played by simulator coverage and induced task complexity. Theorem 2 not only offers a formal upper bound on real-world excess risk, but also provides actionable guidance for simulator design: models must balance fidelity to the true data-generating process with learnability of the induced prediction task.

4.6 Empirical Validation of the Generalization Bound

We now empirically validate the generalization bound in Theorem 2 by constructing a controlled setting in which the degree of mismatch between the synthetic and real distributions can be precisely modulated. This allows us to examine how prediction error scales with distributional shift, and whether the observed degradation aligns with the theoretical guarantees.

Experimental Design. We use a linear dynamical system (LDS) as a generative model. Each trajectory is governed by the stochastic recurrence:

$$x_{t+1} = Ax_t + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \sigma^2 I),$$

where $A \in \mathbb{R}^{d \times d}$ is a system matrix, and $x_t \in \mathbb{R}^d$. We define a synthetic training distribution D_{syn} by fixing a reference matrix A_0 , and define a perturbed test distribution D_{real} by introducing misspecification:

$$A^* = A_0 + \delta U,$$

where U is a random matrix with unit Frobenius norm, and $\delta \in \mathbb{R}_{\geq 0}$ controls the magnitude of deviation. This setup induces a continuum of test-time shifts, indexed by δ , while preserving the mechanistic structure of the simulator.

Learning Task. We train a neural predictor f_ϕ on data drawn from D_{syn} , where the task is to predict the next state x_{t+1} from the current state x_t under squared loss:

$$\ell(f_\phi(x_t), x_{t+1}) = \|f_\phi(x_t) - x_{t+1}\|^2.$$

We then evaluate the trained predictor on data drawn from D_{real} , for varying values of the mismatch parameter δ .

Theoretical Comparison. We evaluate whether the observed degradation in test performance is consistent with the two upper bounds implied by the generalization theory:

- **Worst-case bound:** Based on total variation distance between the input-output distributions, approximated as $\Delta_{\text{TV}} \leq \delta \cdot \mathbb{E}[\|x_t\|]/\sigma$. For numerical stability, we upper bound $\|x_t\| \leq \sqrt{d}$.
- **Empirical data-dependent bound:** Computed as

$$\Delta_{\text{TV}}^{\text{emp}} = \frac{1}{2\sigma} \cdot \mathbb{E}_{x_t \sim D_{\text{syn}}} [\|(A^* - A_0)x_t\|],$$

which tightens the worst-case bound using the actual observed displacement in predictions under mismatch.

Results. Figure 3 plots the test loss of the SGNN predictor on D_{real} as a function of the mismatch level δ , alongside both the worst-case and empirical bounds. We observe:

- The worst-case bound remains valid but conservative, as expected from its generality.
- The empirical bound tracks the test error more closely.
- The increase in test loss is smooth and monotonic with respect to δ , consistent with the additive mismatch penalty predicted by Theorem 2.

5 Mechanistic Interpretability via Back-to-Simulation Attribution

Simulation-Grounded Neural Networks (SGNNs) enable not only accurate prediction, but also mechanistic interpretability by identifying which simulated mechanisms are most responsible for a given prediction. We formalize this process as back-to-simulation attribution and prove that, when trained with an appropriate alignment objective, it yields a consistent estimator of the posterior distribution over latent mechanisms.

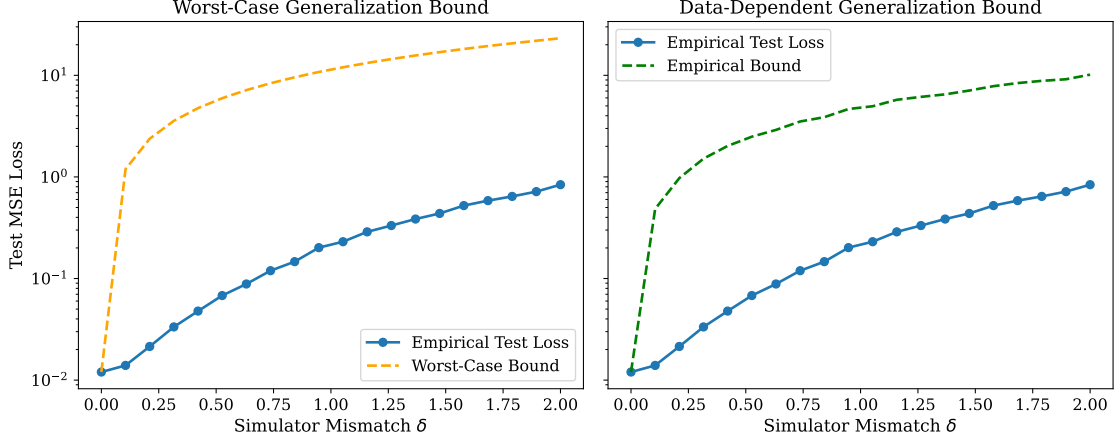


Figure 3: **Empirical validation of the SGNN generalization bound.** We plot the test loss of a predictor trained on synthetic data (A_0) and evaluated under increasing misspecification ($A^* = A_0 + \delta U$). **Left:** Worst-case theoretical bound derived from total variation distance. **Right:** Empirical, data-dependent bound. In both cases, the actual test loss remains below the predicted generalization error, validating the theoretical guarantee in Theorem 2.

5.1 The Back-to-Simulation Attribution Framework

Let $\theta \in \Theta \subset \mathbb{R}^d$ denote the latent simulator parameters, and let $x \sim p(x | \theta)$ denote simulated observations. Recall that the Bayes-optimal predictor of a target $T(\theta)$ is given by the posterior expectation:

$$f^*(x) = \mathbb{E}[T(\theta) | x] = \int T(\theta) p(\theta | x) d\theta$$

The attribution question: Given a trained SGNN’s prediction $f_\phi(x)$ for some input x , which underlying mechanisms θ are most responsible for this prediction?

Our approach: We maintain a reference library $\mathcal{L} = \{(\theta_i, x_i)\}_{i=1}^M$ of simulator runs and their corresponding parameters. For any query input x , we:

1. **Encode** the query and library examples: $\phi(x), \phi(x_1), \dots, \phi(x_M)$
2. **Compute similarity weights** based on embedding distance:

$$w_i(x) = \frac{\kappa(\phi(x), \phi(x_i))}{\sum_{j=1}^M \kappa(\phi(x), \phi(x_j))}$$

3. **Return the weighted distribution** over mechanisms: $\hat{p}_M(\theta | x) = \sum_{i=1}^M w_i(x) \delta_{\theta_i}$

where $\kappa(\cdot, \cdot)$ is a similarity kernel (e.g., RBF kernel) and δ_{θ_i} is a point mass at θ_i .

Intuition: We attribute the prediction to mechanisms θ_i whose simulated outputs x_i are similar (in embedding space) to our query x .

5.2 Attribution Consistency Under Alignment Training

The key insight is that attribution quality depends fundamentally on whether the learned embedding ϕ preserves the information needed to recover the posterior $p(\theta | x)$. Rather than hoping this happens by accident, we can explicitly train for it.

Theorem 3 (Back-to-Simulation Attribution Consistency). *Consider an SGNN trained with prediction loss $\mathcal{L}_{\text{pred}}$ and KL alignment loss:*

$$\mathcal{L}_{KL} = \mathbb{E}_{x \sim p(x)} \left[KL \left(p(\theta | x) \parallel \sum_{i=1}^M w_i(x) \delta_{\theta_i} \right) \right]$$

Assume: (1) Training converges: $f_\phi \rightarrow f^$ and $\mathcal{L}_{KL} \rightarrow 0$, (2) Library samples $\{\theta_i\}$ become dense in Θ , (3) $T(\theta)$ is continuous and bounded.
Then for any positive integer k :*

$$\sum_{i=1}^M w_i(x) T(\theta_i)^k \rightarrow \mathbb{E}[T(\theta)^k | x]$$

In particular, the attribution distribution converges in all moments to the true posterior.

Proof deferred to Appendix A.

Interpretation: This shows that the attribution distribution $\sum_i w_i(x) \delta_{\theta_i}$ captures not only the mean of the true posterior (when $k = 1$), but also its variance (when $k = 2$), skewness, and all higher-order moments. This provides complete uncertainty quantification: the attribution weights correctly represent both the central tendency and the spread of plausible mechanisms.

5.3 Comparison with Existing Interpretability Paradigms

Back-to-simulation attribution offers a fundamentally different perspective on interpretability compared to traditional methods. Rather than assigning importance scores to input features or referencing training examples, it provides mechanistic explanations in terms of the latent generative parameters that govern the underlying simulator.

Feature Attribution Methods. Standard approaches such as SHAP and LIME [17, 18] explain predictions by identifying the contribution of input features (e.g., pixels or covariates). These methods answer the question: “Which features of this specific input influenced the model’s prediction?”

By contrast, back-to-simulation attribution returns a distribution over latent parameters $\theta \in \Theta$ such that the observed input x is likely under the simulation $\mathcal{S}(\theta)$. This addresses a different and arguably more fundamental question: “What generative mechanism explains this observed behavior?”

Example-Based Methods. Influence functions and related techniques [19] identify which training points most affected a given prediction. However, they remain tied to the training set and reflect empirical correlation rather than mechanistic causality.

In contrast, back-to-simulation attribution uses a reference library of simulated examples $\{(\theta_i, x_i)\}_{i=1}^M$, and retrieves examples not based on superficial similarity but shared generative origin. These attributions recover abstract mechanisms—even when corresponding trajectories differ substantially in their surface features.

| Method | Explanatory Focus |
|--------------------------------|--|
| SHAP, LIME | Input feature importance |
| Influence functions | Individual training point relevance |
| Back-to-simulation attribution | Latent scientific mechanism (θ) |

Scientific Utility. Because SGNNs are trained on synthetic data generated from known mechanisms, back-to-simulation attribution yields explanations grounded in domain-relevant quantities. This makes it particularly well-suited for scientific machine learning, where understanding system behavior is often more important than maximizing predictive accuracy alone.

5.4 Empirical Validation of Attribution Consistency

We empirically validate Theorem 3, which establishes that back-to-simulation attribution with KL alignment training provides consistent estimators of all moments of the posterior distribution $p(\theta \mid x)$. Our experiments demonstrate that SGNNs trained with explicit attribution alignment can accurately recover both the mean and higher-order statistics of the generative mechanisms responsible for observed trajectories.

5.4.1 Experimental Setup

We simulate trajectories from a classical SIR (Susceptible-Infected-Recovered) model governed by two latent parameters: transmission rate β and recovery rate γ . Parameters are drawn from independent uniform priors:

$$\beta \sim \mathcal{U}(0.1, 0.5), \quad \gamma \sim \mathcal{U}(0.05, 0.2)$$

Each trajectory is a 50-step time series, from which the first 40 steps are used as input $x_{1:40}$, and the final 10 steps form the prediction target $x_{41:50}$.

Training with KL alignment: Following Theorem 3, we train the SGNN with two objectives:

$$\mathcal{L}_{\text{pred}} = \mathbb{E}_{\theta \sim p(\theta)} [\|f_\phi(x_{1:40}) - x_{41:50}\|^2] \quad (2)$$

$$\mathcal{L}_{\text{KL}} = \mathbb{E}_{x \sim p(x)} \left[\text{KL} \left(p(\theta \mid x) \parallel \sum_{i=1}^M w_i(x) \delta_{\theta_i} \right) \right] \quad (3)$$

The total loss is $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{pred}} + \lambda \mathcal{L}_{\text{KL}}$ with $\lambda = 0.1$.

5.4.2 Attribution Procedure

We construct a reference library $\mathcal{L} = \{(\theta_i, x_i)\}_{i=1}^M$ of $M = 2000$ synthetic trajectories. For any test input x^{test} , attribution proceeds by:

1. **Encode** the query and library examples using the learned encoder ϕ
2. **Compute similarity weights:**

$$w_i(x^{\text{test}}) = \frac{\exp\left(-\frac{\|\phi(x^{\text{test}}) - \phi(x_i)\|^2}{h^2}\right)}{\sum_{j=1}^M \exp\left(-\frac{\|\phi(x^{\text{test}}) - \phi(x_j)\|^2}{h^2}\right)}$$

3. **Return attribution distribution:**

$$\hat{p}_M(\theta \mid x^{\text{test}}) = \sum_{i=1}^M w_i(x^{\text{test}}) \delta_{\theta_i}$$

This attribution distribution can then be used to compute any desired statistics, such as the posterior mean $\sum_i w_i(x) \theta_i$, posterior variance, or to assess uncertainty over the generative mechanisms.

5.4.3 Results

KL Alignment Convergence: Figure 4 shows that the KL divergence between the SGNN attribution distribution and the true posterior decreases significantly during training, reaching values as low as 0.001–0.01. This confirms that the KL alignment loss successfully drives the attribution weights to approximate the true posterior $p(\theta \mid x)$.

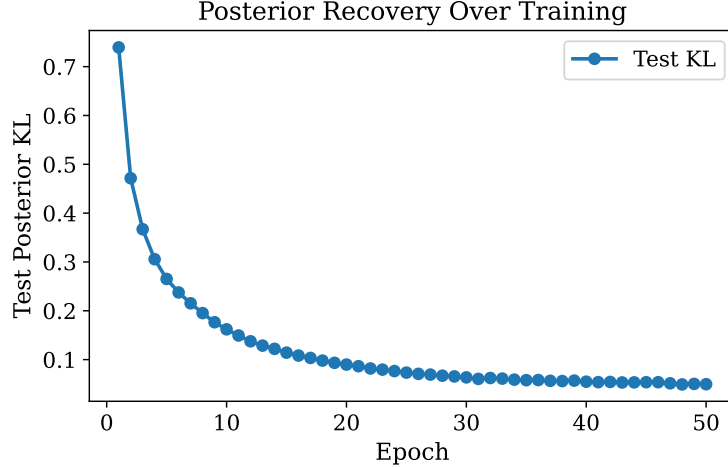


Figure 4: **Attribution consistency under KL alignment training.** KL divergence between SGNN attribution distribution and true posterior decreases over training epochs.

6 Learning Unobservable Scientific Quantities

A core limitation of traditional supervised learning is its dependence on ground-truth labels. Many quantities scientists care about are fundamentally unobservable: e.g., a disease’s basic reproduction number (R_0), the mean number of people one infected individual will infect, or the mechanistic form governing an outbreak (e.g., whether there is asymptomatic transmission of the disease). SGNNs overcome this barrier by generating synthetic data in which such latent objects are known by construction, thereby turning an otherwise unlabelled problem into a supervised one.

6.1 Formal Setting

Let

$$\mathcal{M} \in \{\mathcal{M}_1, \dots, \mathcal{M}_K\}, \quad \theta \in \Theta, \quad x = (\mathcal{O} \circ \mathcal{M})(\theta)$$

denote, respectively, a mechanistic model, its latent parameters, and the resulting observable. We distinguish two target types:

1. **Parametric target:** $y_{\text{param}} = T(\theta)$, where $T : \Theta \rightarrow \mathcal{Y}$ maps parameters to a scientific quantity (e.g. R_0 , carrying capacity).
2. **Structural target:** $y_{\text{struct}} = T(\theta)$, where T assigns a label to the underlying model (e.g. “SIR” vs. “SEIR”).

Definition 5 (Unobservable target). A target, parametric or structural, is unobservable if, in real-world data, x is never paired with its ground-truth value.

6.2 Simulation Enables Learning

Remark 2 (SGNNs learn the unobservable). Let $T(\theta)$ and $S(\mathcal{M})$ be unobservable targets.

1. A learner trained on real-world pairs $\{(x_i, y'_i)\}$, where each y'_i is *observed*, receives no signal about $T(\theta)$ and cannot estimate it non-trivially.
2. An SGNN trained on synthetic pairs $(x_i, T(\theta_i))$ converges (Theorem 1) to the Bayes-optimal estimators

$$f^*(x) = \mathbb{E}_{D_{\text{syn}}}[T(\theta) | x]$$

Proof sketch. Part (1) follows because the training loss never references the unobservable. For part (2), SGNNs create their own labelled data: draw $\theta \sim P(\theta)$, record the associated label, simulate x , then train by MSE or cross-entropy for structural targets. By consistency (Prop. 1) the network converges to the conditional expectation, which is Bayes-optimal. \square

Identifiability. Proposition 2 presumes that the target is identifiable for all candidate models:

$$p(x \mid \theta) = p(x \mid \theta') \implies T(\theta) = T(\theta').$$

If this condition does not hold, then the target is not identifiable from the data distribution, and no method—regardless of model capacity or access to training data—can recover it.

6.3 Illustrative Examples

- **Epidemiology — Parametric.** SGNNs map noisy case trajectories to hidden parameters such as R_0 or incubation period, which are unmeasurable in real time.
- **Epidemiology — Structural.** Given incidence curves, an SGNN can classify which compartmental model (SIR, SEIR, SIRS) best explains the data—information not present in the observations alone.

These examples highlight a key advantage of simulation-grounded learning: it turns scientifically meaningful but unobservable quantities into learnable targets, extending predictive modelling beyond the limits of traditional supervised learning.

6.4 Experiment: Learning Unobserved Structural Targets

We empirically validate the theoretical prediction that SGNNs can learn unobservable structural targets by testing their ability to distinguish between competing epidemiological model structures. This experiment demonstrates both the practical utility of learning structural unobservables and provides concrete evidence that SGNNs outperform classical statistical approaches on this fundamental scientific task.

Experimental Setup. We consider the problem of distinguishing between SIR and SEIR epidemiological models from noisy trajectory observations. We implement discrete-time versions of both models with unit population. For SIR: $S_{t+1} = S_t - \beta S_t I_t$, $I_{t+1} = I_t + \beta S_t I_t - \gamma I_t$, $R_{t+1} = R_t + \gamma I_t$. For SEIR, we add an exposed compartment with transition rate σ . We generate 60,000 synthetic epidemics (30,000 each) by sampling $\beta \sim \mathcal{U}(0.1, 0.5)$, $\gamma \sim \mathcal{U}(0.05, 0.2)$, and for SEIR, $\sigma \sim \mathcal{U}(0.1, 0.3)$. Each trajectory runs for 100 time steps with additive Gaussian noise ($\sigma = 0.01$) applied to the infected compartment.

SGNN Architecture and Training. We train a 1D convolutional neural network with residual blocks, GroupNorm, and GELU activations to classify 100-dimensional infected time series as SIR (0) or SEIR (1). The architecture uses progressive channel expansion (64→128→256) with adaptive pooling and a fully connected classifier. Training uses cross-entropy loss with Adam optimizer (lr=10⁻⁵) for 20 epochs on 48,000 samples.

AIC Baseline. For comparison, we implement AIC-based model selection by fitting both SIR and SEIR models to each test trajectory using nonlinear least squares, then selecting the model with lower AIC score: $AIC = n \log(RSS/n) + 2k$ where k is the number of parameters [14].

Results. Figure 5 shows the results on 12,000 held-out samples. SGNNs achieve substantially lower classification error than AIC throughout training, converging to approximately 4% error while AIC maintains 9.2% error. This demonstrates that SGNNs can reliably learn structural differences between mechanistic models that classical statistical methods struggle to distinguish.

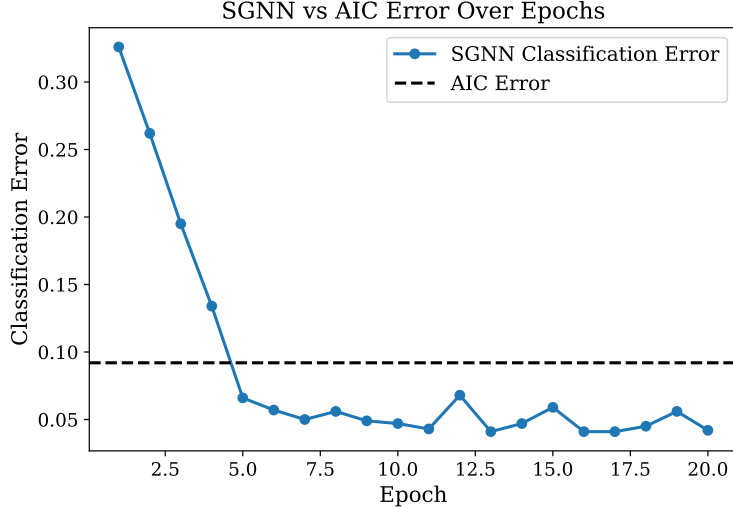


Figure 5: **SGNNs outperform AIC for structural model selection.** Classification error over training epochs for distinguishing SIR vs SEIR models from noisy trajectory data. SGNNs rapidly converge to low error rates while AIC maintains significantly higher error, demonstrating the advantage of simulation-grounded learning for unobservable structural targets.

Interpretation. This experiment validates Proposition 2 for structural targets. SGNNs learn robust mappings from noisy trajectories to mechanistic structure through simulation-based training, while AIC struggles with parameter estimation uncertainty. This demonstrates the practical utility of simulation-grounded learning for scientific model selection.

7 Discussion

This work presents a formal foundation for simulation-grounded prediction, framing Simulation-Grounded Neural Networks (SGNNs) as consistent approximators of the Bayes-optimal predictor under a simulation-induced data distribution. The theoretical results unify several previously ad hoc insights about SGNN behavior, yielding interpretable guarantees and failure modes.

A central result of this theory is that SGNNs trained on synthetic data perform amortized Bayesian estimation: they learn to map observed data to the posterior expectation of a scientific quantity of interest under the simulator’s generative model. This quantity may be the latent mechanism itself (e.g., simulator parameters) or a downstream function of it (e.g., reproduction number, source node, or model structure). In both cases, the SGNN approximates the Bayes-optimal predictor $f^*(x) = \mathbb{E}[y \mid x]$. This perspective provides a principled foundation for understanding generalization, consistency, and interpretability in simulation-grounded learning (Section 3).

Our generalization bound under model misspecification (Section 4) decomposes the test risk into two interpretable terms: the synthetic excess risk (arising from limited simulation samples or model capacity), and the mismatch penalty (arising from divergence between the simulator and the real-world data distribution). This decomposition not only captures known robustness properties of SGNNs but also enables informed simulator design and principled evaluation.

In addition, we formalized two distinctive properties of SGNNs. First, we established that back-to-simulation attribution approximates the posterior over latent mechanisms, with provable consistency guarantees (Section 5). Second, we showed that SGNNs can estimate unobservable scientific targets $T(\theta)$, quantities that do not admit ground-truth labels in real-world datasets, provided these are identifiable under the simulator (Section 6). These results highlight the unique capabilities of simulation-based training beyond traditional supervised learning paradigms.

7.1 Implications for Scientific Machine Learning

The theoretical framework developed here has direct implications for scientific model design and deployment.

The Simulator as an Inductive Bias. Rather than treating a simulator as a fixed model of the world, our formulation interprets it as a structured prior over hypotheses, encoded through distributions over parameters, mechanisms, and observation processes. Training an SGNN corresponds to amortizing inference over this space, enabling prediction even when empirical data is limited or incomplete.

Tradeoffs in Simulation Fidelity. The generalization bound in Theorem 2 formalizes a trade-off between simulator complexity and learnability. Oversimplified simulators yield a large mismatch penalty, while overparameterized simulators may create intractable learning problems with large synthetic risk. Effective simulator design involves identifying a regime where the synthetic task is both expressive and statistically learnable.

Data-Limited Regimes. In domains where empirical data is scarce or unlabelable, simulation-grounded learning offers a viable alternative. The ability to learn from unobservable quantities that are defined mechanistically but not measured extends the reach of predictive modeling in scientific contexts where traditional labels are unavailable.

7.2 Limitations and Future Directions

While this work establishes foundational results, several directions remain for refinement and extension.

Mismatch Estimation and Control. The mismatch penalty depends on divergences such as $\Delta_{TV}(D_{\text{real}} \parallel D_{\text{syn}})$, which are generally unobservable. Estimating or bounding this term from unlabeled real-world data, possibly using domain adaptation or two-sample testing methods, is an important direction for enabling practical performance guarantees.

Active Simulation Design. Our framework assumes a fixed simulator prior $P(\theta)$. In practice, simulation may be expensive. Incorporating active learning or adaptive simulation strategies could allow more efficient coverage of the hypothesis space, reducing sample complexity and improving generalization.

Beyond Prediction: Causality and Control. The SGNN paradigm naturally extends to interventional and dynamic settings. For instance, SGNNs could be trained on interventional simulators to estimate treatment effects or learn policies under uncertainty. Establishing consistency and generalization guarantees in these contexts is a promising direction for both theory and application.

8 Conclusion

We introduce a formal theory of simulation-grounded prediction and establish Simulation-Grounded Neural Networks (SGNNs) as consistent estimators of the Bayes-optimal predictor under a structured simulation prior. Our analysis yields a generalization bound that isolates the impact of model misspecification, proves consistency of attribution methods grounded in simulation, and characterizes conditions under which unobservable scientific targets can be reliably learned. These results provide rigorous foundations for SGNNs, supporting their use in data-limited scientific domains where traditional empirical learners fail. More broadly, this work bridges mechanistic modeling and modern machine learning, offering a principled path toward interpretable, robust, and scientifically grounded prediction.

References

- [1] Carson Dudley, Reiden Magdaleno, Christopher Harding, and Marisa Eisenberg. Simulation as supervision: Mechanistic pretraining for scientific discovery. *arXiv preprint arXiv:2507.08977*, 2025.
- [2] Carson Dudley, Reiden Magdaleno, Christopher Harding, Ananya Sharma, Emily Martin, and Marisa Eisenberg. Mantis: A simulation-grounded foundation model for disease forecasting. *arXiv preprint arXiv:2508.12260*, 2025.
- [3] Kyle Cranmer, Johann Brehmer, and Gilles Louppe. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 2020.
- [4] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.
- [5] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 2023.
- [6] Christopher Rackauckas, Yingbo Ma, Julius Martensen, Collin Warner, Kirill Zubov, Rohit Supekar, Dominic Skinner, Ali Ramadhan, and Alan Edelman. Universal differential equations for scientific machine learning, 2021.
- [7] Lijing Wang, Jiangzhuo Chen, and Madhav Marathe. Defsi: Deep learning based epidemic forecasting with synthetic information. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 2019.
- [8] Abhishek Kadian, Joanne Truong, Aaron Gokaslan, Alexander Clegg, Erik Wijmans, Stefan Lee, Manolis Savva, Sonia Chernova, and Dhruv Batra. Sim2real predictivity: Does evaluation in simulation predict real-world performance? *IEEE Robotics and Automation Letters*, 5(4):6670–6677, October 2020.
- [9] Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do bayesian inference. *arXiv preprint arXiv:2112.10510*, 2021.
- [10] Noah Hollmann, Samuel Müller, Katharina Eggenberger, and Frank Hutter. Tabpfn: A transformer that solves small tabular classification problems in a second, 2023.
- [11] Thomas Nagler. Statistical foundations of prior-data fitted networks. In *International Conference on Machine Learning*, 2023.
- [12] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. MIT Press, 2nd edition, 2018.
- [13] Peter L. Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2002.
- [14] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974.
- [15] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016.
- [16] Jiale Tan and Marisa C. Eisenberg. Lasso-ode: A framework for mechanistic model identifiability and selection in disease transmission modeling, 2025.
- [17] Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions, 2017.

- [18] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier, 2016.
- [19] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions, 2020.

A Proofs of Main Results

A.1 Proof of Theorem 1

Proof. Step 1: Bias-variance decomposition. The total excess risk naturally decomposes as:

$$R_{\text{syn}}(f_{\phi_N}) - R_{\text{syn}}(f^*) = \left(R_{\text{syn}}(f_{\phi_N}) - \inf_{f \in \mathcal{F}} R_{\text{syn}}(f) \right) + \left(\inf_{f \in \mathcal{F}} R_{\text{syn}}(f) - R_{\text{syn}}(f^*) \right) \quad (4)$$

$$= \mathcal{E}_{\text{est}} + \mathcal{E}_{\text{approx}} \quad (5)$$

The approximation error $\mathcal{E}_{\text{approx}}$ measures the fundamental limitation of our function class \mathcal{F} and depends only on the expressiveness of the model family. Our goal is to bound the estimation error \mathcal{E}_{est} .

Step 2: Define the population risk minimizer. Let $f_{\mathcal{F}}^* \in \arg \min_{f \in \mathcal{F}} R_{\text{syn}}(f)$ be the function in \mathcal{F} that achieves the population risk infimum. Note that $R_{\text{syn}}(f_{\mathcal{F}}^*) = \inf_{f \in \mathcal{F}} R_{\text{syn}}(f)$.

Step 3: Rademacher concentration inequality. We apply the following generalization of Mohri et al. (2018), Theorem 3.3:

Lemma 1 (Concentration for bounded function classes). *For a function class \mathcal{G} with functions $g : \mathcal{Z} \rightarrow [0, B]$, with probability at least $1 - \delta$:*

$$\sup_{g \in \mathcal{G}} \left(R(g) - \widehat{R}_N(g) \right) \leq 2 \widehat{\mathfrak{R}}_N(\mathcal{G}) + 3B \sqrt{\frac{\log(2/\delta)}{2N}} \quad (6)$$

where $R(g) = \mathbb{E}[g]$ and $\widehat{R}_N(g) = \frac{1}{N} \sum_{i=1}^N g(z_i)$.

Step 4: Apply to loss class. Define the loss class $\mathcal{G} = \{z \mapsto \ell(f(z), y) : f \in \mathcal{F}\}$. Since ℓ maps to $[0, B]$, each function in \mathcal{G} is bounded by B .

Applying the concentration inequality (6) to $g_f(z) = \ell(f(z), y)$ gives us, with probability $1 - \delta$:

$$R_{\text{syn}}(f) - \widehat{R}_N(f) \leq 2 \widehat{\mathfrak{R}}_N(\mathcal{G}) + 3B \sqrt{\frac{\log(2/\delta)}{2N}} \quad \forall f \in \mathcal{F} \quad (7)$$

Step 5: Apply to specific functions. From (7), we have:

$$R_{\text{syn}}(f_{\phi_N}) \leq \widehat{R}_N(f_{\phi_N}) + 2 \widehat{\mathfrak{R}}_N(\mathcal{G}) + 3B \sqrt{\frac{\log(2/\delta)}{2N}} \quad (8)$$

$$R_{\text{syn}}(f_{\mathcal{F}}^*) \leq \widehat{R}_N(f_{\mathcal{F}}^*) + 2 \widehat{\mathfrak{R}}_N(\mathcal{G}) + 3B \sqrt{\frac{\log(2/\delta)}{2N}} \quad (9)$$

Step 6: Use ERM property. By definition of empirical risk minimization: $\widehat{R}_N(f_{\phi_N}) \leq \widehat{R}_N(f_{\mathcal{F}}^*)$.

Step 7: Combine bounds. Subtracting (9) from (8):

$$R_{\text{syn}}(f_{\phi_N}) - R_{\text{syn}}(f_{\mathcal{F}}^*) \leq \left(\widehat{R}_N(f_{\phi_N}) - \widehat{R}_N(f_{\mathcal{F}}^*) \right) + 2 \cdot \left[2 \widehat{\mathfrak{R}}_N(\mathcal{G}) + 3B \sqrt{\frac{\log(2/\delta)}{2N}} \right] \quad (10)$$

$$\leq 0 + 4 \widehat{\mathfrak{R}}_N(\mathcal{G}) + 6B \sqrt{\frac{\log(2/\delta)}{2N}} \quad (11)$$

Step 8: Apply Talagrand's contraction lemma. Since ℓ is L -Lipschitz in its first argument, Talagrand's Contraction Lemma (Mohri et al., Lemma 5.7) gives:

$$\widehat{\mathfrak{R}}_N(\mathcal{G}) \leq L \widehat{\mathfrak{R}}_N(\mathcal{F})$$

Therefore:

$$\mathcal{E}_{\text{est}} = R_{\text{syn}}(f_{\phi_N}) - R_{\text{syn}}(f^*) \leq 4L \hat{\mathfrak{R}}_N(\mathcal{F}) + 6B \sqrt{\frac{\log(2/\delta)}{2N}}$$

Step 9: Final result. Combining with the approximation error:

$$R_{\text{syn}}(f_{\phi_N}) - R_{\text{syn}}(f^*) \leq \left(\inf_{f \in \mathcal{F}} R_{\text{syn}}(f) - R_{\text{syn}}(f^*) \right) + 4L \hat{\mathfrak{R}}_N(\mathcal{F}) + 6B \sqrt{\frac{\log(2/\delta)}{2N}}$$

□

A.2 Proof of Theorem 2

Proof. The core of the proof relies on relating the risks under D_{real} to those under D_{syn} . For any predictor f , the difference in its expected loss under the two distributions is bounded by the Total Variation distance:

$$|R_{\text{real}}(f) - R_{\text{syn}}(f)| = |\mathbb{E}_{D_{\text{real}}}[\ell(f(x), y)] - \mathbb{E}_{D_{\text{syn}}}[\ell(f(x), y)]| \leq L_{\text{max}} \Delta_{TV}. \quad (12)$$

This gives us two useful inequalities:

$$R_{\text{real}}(f) \leq R_{\text{syn}}(f) + L_{\text{max}} \Delta_{TV} \quad (13)$$

$$-R_{\text{real}}(f) \leq -R_{\text{syn}}(f) + L_{\text{max}} \Delta_{TV}. \quad (14)$$

We now bound the excess risk on the real distribution.

$$\begin{aligned} R_{\text{real}}(f_{\phi_N}) - R_{\text{real}}(f_{\text{real}}^*) &\leq (R_{\text{syn}}(f_{\phi_N}) + L_{\text{max}} \Delta_{TV}) - R_{\text{real}}(f_{\text{real}}^*) && \text{by (13) for } f_{\phi_N} \\ &\leq R_{\text{syn}}(f_{\phi_N}) + L_{\text{max}} \Delta_{TV} + (-R_{\text{syn}}(f_{\text{real}}^*) + L_{\text{max}} \Delta_{TV}) && \text{by (14) for } f_{\text{real}}^* \\ &= R_{\text{syn}}(f_{\phi_N}) - R_{\text{syn}}(f_{\text{real}}^*) + 2L_{\text{max}} \Delta_{TV}. \end{aligned}$$

By definition, f_{syn}^* is the minimizer of the risk under D_{syn} , so $R_{\text{syn}}(f_{\text{syn}}^*) \leq R_{\text{syn}}(f_{\text{real}}^*)$. This implies $-R_{\text{syn}}(f_{\text{real}}^*) \leq -R_{\text{syn}}(f_{\text{syn}}^*)$. Substituting this into the last line gives the final bound:

$$R_{\text{real}}(f_{\phi_N}) - R_{\text{real}}(f_{\text{real}}^*) \leq R_{\text{syn}}(f_{\phi_N}) - R_{\text{syn}}(f_{\text{syn}}^*) + 2L_{\text{max}} \Delta_{TV}. \quad \square$$

A.3 Proof of Theorem 3

Proof. **Step 1: Prediction consistency.** From Theorem 1, $f_{\phi}(x) \rightarrow \mathbb{E}[T(\theta) | x]$.

Step 2: KL convergence. Since $\mathcal{L}_{\text{KL}} \rightarrow 0$, we have $\text{KL}(p(\theta | x) || \sum_i w_i(x) \delta_{\theta_i}) \rightarrow 0$ for almost every x .

Step 3: Moment convergence via KL convergence. For any bounded measurable function $g : \Theta \rightarrow \mathbb{R}$, KL convergence implies:

$$\left| \mathbb{E}_{p(\theta|x)}[g(\theta)] - \sum_{i=1}^M w_i(x) g(\theta_i) \right| \rightarrow 0$$

This is because KL divergence controls the total variation distance, and total variation convergence implies convergence of expectations for all bounded functions.

Step 4: Apply to moment functions. For any positive integer k , define $g_k(\theta) = T(\theta)^k$. Since $T(\theta)$ is bounded, $g_k(\theta)$ is also bounded. Applying Step 3:

$$\sum_{i=1}^M w_i(x) T(\theta_i)^k = \sum_{i=1}^M w_i(x) g_k(\theta_i) \rightarrow \mathbb{E}_{p(\theta|x)}[g_k(\theta)] = \mathbb{E}[T(\theta)^k | x]$$

□