Symbolic Density Estimators for Unnormalized Distributions

Anonymous authors
Paper under double-blind review

Abstract

Estimating the symbolic or analytical form of probability density functions (PDFs) from observed samples is a fundamental challenge in statistical and computational modelling. This process is critical for deriving interpretable and generalizable relationships characterizing the underlying phenomenon. Traditionally, this estimation depends strongly on domain expertise and prior field-specific knowledge, with experts selecting appropriate functional forms or parametric families based on empirical evidence and theoretical understanding. The coefficients of these forms are then typically determined through parameter estimation. In this paper, we develop a framework to estimate symbolic expressions of unnormalized distributions from their observed samples. We integrate deep generative models with symbolic regression (SR), incorporating inductive biases, such as, factorizing large distributions, to keep the problem tractable. The deep generative models we examine include likelihood-based models, viz., flow models, and score-based models. Experiments show the effectiveness of the proposed framework for estimating density functions for multivariate toy distributions as well as lattices from computational Physics, namely, XY model and ϕ^4 theory. When applied to the renormalization problem in ϕ^4 theory, it discovers new expressions for action function intractable by traditional analytic approaches, thereby providing physicists with a novel tool for theoretical analysis.

1 Introduction

In the physical sciences, understanding complex phenomena fundamentally depends on modeling the underlying system dynamics. The probability distributions that characterize these dynamics, as Hamiltonian or energy in a Boltzmann distribution, are traditionally formulated by domain experts, often in an analytical form with a set of parameters. These parameters can be estimated from the data using regression or data-fitting techniques (Shanahan et al., 2018). While effective, this process is labor-intensive and is heavily reliant on expert intuition. Consequently, the problem of automatically inferring an analytical or symbolic representation of a probability density function directly from observed data presents a significant challenge (Tohme et al., 2024). Addressing this problem lies at the intersection of machine learning and physics, offering the potential to automate model discovery and to enhance our understanding of complex physical systems. This problem is analogous to system identification problem, widely studied in mechanical and control systems Sahoo et al. (2018), where the system dynamics is modeled in differential form (using differential equations) instead of an integral form (using Hamiltonian formulation).

Boltzmann distributions are abundantly used in physical sciences (Akhound-Sadegh et al., 2024; Midgley et al., 2023; Hasenfratz & Niedermayer, 1994). They have the form $p(\mathbf{x}) \propto \exp(-H(\mathbf{x}))$, where $H(\mathbf{x})$ is a scalar denoting energy of the system in state $\mathbf{x}$. Popular deep generative approaches to model these distributions include likelihood-based approaches, such as normalizing flows (Dinh et al., 2017) and autoregressive models (Uria et al., 2016; Van Den Oord et al., 2016), and energy-based approaches, such as score matching (Hyvärinen, 2005; Song & Ermon, 2019). These methods are able to generate samples from the target distribution while also providing information about the model likelihood i.e., log likelihood in case of likelihood based approaches and the gradient thereof in case of score matching approaches. However, these models are black box estimators. On the other hand, Symbolic regression (SR) methods are used to estimate the equations of the input-output relations in a supervised way. Popular approaches for SR include

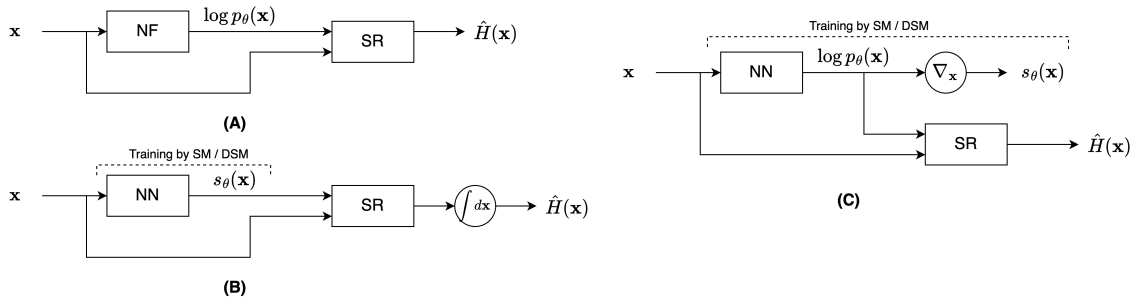


Figure 1: Block diagram of the proposed framework. (A) Flow-based estimation. (B) Score-based estimation. (C) Integration-free score-based estimation.

genetic algorithms (Schmidt & Lipson, 2009; La Cava et al., 2016; Pal & Wang, 2017; La Cava et al., 2019; Cranmer, 2023), symbolic neural networks (Martius & Lampert, 2017; Sahoo et al., 2018; Kim et al., 2021) and large pre-trained networks (Biggio et al., 2021; Kamienny et al., 2022; Vastl et al., 2024), which work well for a small number of variables (typically upto 10). There are a few approaches (Cranmer et al., 2020) to combine SR with deep learning methods, which scale well with the number of variables, to simplify the SR task. While a few approaches (Tohme et al., 2024), attempt to recover expressions for low-dimensional unnormalized distributions, they rely on assuming the distribution's form.

In this paper, we propose a framework to integrate deep generative modeling with SR to estimate the density functions as symbolic expressions. We consider flow-based and score-based model for density estimation, and subsequently use SR methods, including genetic algorithms (Cranmer, 2023) and EQL-based approaches (Kim et al., 2021), to estimate the symbolic density function while incorporating task-specific inductive biases. The effectiveness of SR methods deteriorates notably with an increase in the number of variables or the dimensionality of the data, often leading to inaccurately estimated expressions (Cranmer et al., 2020; Biggio et al., 2021). For this, we employ a density factorization approach that leverages local dependencies within the distribution. This strategy effectively reduces the number of variables, thereby enabling more efficient and accurate expression estimation. Again, deep generative models sample from high density regions of $p(x)$. For effective SR, which needs samples from both high and low density regions, we introduce noisy perturbation of samples.

We apply this framework to general density functions, such as multi-variate Gaussian and many-well distributions, as well as to problems in computational Physics, such as XY model (Kosterlitz & Thouless, 1973) and scalar ϕ^4-theory (Singha et al., 2023; Albergo et al., 2019), and recover the energy functions, $H(\mathbf{x})$ We also derive the Hamiltonian for renormalization of strongly-coupled ϕ^4-theory, which is challenging to derive analytically. These experiments support the utility of the proposed framework for non-perturbative Physics, where analytic derivation of hamiltonians is not feasible manually.

2 The Framework

Our framework consists of the following parts. (1) Training a deep generative model using the given samples. (2) Fitting an SR model using the given samples and the corresponding likelihood information (log values or gradients) provided by the deep generative model, and with density factorization conforming to the known nature of the system. The symbolic expression thus derived gives the energy $H(\mathbf{x})$ of the system. A block schematic of the framework is depicted in Fig. 1.

Lattice Systems: Apart from general distributions, such as multivariate Gaussian and many-well distributions, this paper focuses on lattice systems common in statistical and particle physics. Lattices are obtained by discretizing the continuous space on a regular grid. Recent works (Albergo et al., 2019; Singha et al., 2023; Kanaujia & Arora, 2025) predominantly use normalizing flows to model the lattice distributions

because of the availability of exact likelihood from the flow model. In addition, Wang et al. (2023) have investigated diffusion-based approaches for the same purpose. In this work, we start with flow based models, and subsequently, introduce score-based methods to model the lattice systems.

Problem statement: Given the samples $\mathcal{D} = \{\mathbf{x}_i \in \mathcal{X}\}_{i=1}^N$ from an unknown distribution $p(\mathbf{x}); \mathbf{x} \in \mathbb{R}^d$, estimate a symbolic expression $\hat{H}(\mathbf{x}) \in \mathcal{H}$ such that $q_\theta(\mathbf{x}) = \frac{1}{Z} e^{-\hat{H}(\mathbf{x})}$ approximates $p(\mathbf{x})$.

Symbolic Regression: SR is a supervised learning approach that estimates an analytical expression for $\mathbf{y}$ in terms of $\mathbf{x}$, i.e., $\hat{\mathbf{y}} = f(\mathbf{x})$, given a set of input-output pairs $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$. In contrast to conventional regression methods, which search for parameters of an equation of a predetermined form, symbolic regression simultaneously searches for both the model parameters and the equation's form (Schmidt & Lipson, 2009; 2010; Biggio et al., 2021), using a set of pre-defined basis functions and operations.

We use two approaches for SR, one is genetic programming based algorithm implemented using PySR library (Cranmer, 2023) and the other is deep neural network based equation learner (EQL) approach (Martius & Lampert, 2017; Sahoo et al., 2018; Kim et al., 2021). PySR uses evolutionary algorithms to optimise a tree data structure that represents a mathematical expression in order to best fit the data (Koza, 1994) in terms of mean squared error. In the search for the underlying structure, model error is usually balanced with model complexity to ensure brevity of the learned equation and avoids overfitting. PySR uses multi-population evolutionary genetic algorithm with multiple evolutions performed asynchronously. The other approach, EQL uses a symbolic neural network trained end-to-end through backpropagation. Model minimizes the mean squared error between y and $\hat{y}$. To enforce sparsity, a smoothed variant of $L_{0.5}$ regularizer is used. Additional details for both SR approaches are provided in Appendix B.

The quality of convergence is indicated by the value of mean squared error. Many a times, SR fails to converge and this may be indicated by a high value of mean squared error. One may use this to select the preferred solution from multiple SR models.

Flow models: Flows model a target density $p(\mathbf{x})$ by applying a sequence of simple and learnable bijective transformations on samples from a known standard distribution $q(\mathbf{z}); \mathbf{z} \in \mathbb{R}^d$ (Papamakarios et al., 2021; Kobyzev et al., 2021; Rezende & Mohamed, 2015). In general, $q(\mathbf{z})$ is chosen to be standard Gaussian. Mathematically, a sample $\mathbf{z} \sim q(\mathbf{z})$ is transformed to $\mathbf{x}$ with a bijective map T_θ, parameterized by θ. The density of $\mathbf{x} = T_\theta(\mathbf{z})$ is given by

$$p_\theta(\mathbf{x}) = q(T_\theta^{-1}(\mathbf{x}))|\det(J_{T_\theta^{-1}}(\mathbf{x}))| \tag{1}$$

Here, $J_{T_\theta^{-1}}(\mathbf{x}) = \partial T_\theta^{-1}(\mathbf{x})/\partial \mathbf{x}$ is the jacobian of T_θ^{-1}. In order to model the unknown target distribution $p(\mathbf{x})$, using the samples from $p(\mathbf{x})$, θ is learnt by minimizing the KL divergence between the target density $p(\mathbf{x})$ and the model density $p_\theta(\mathbf{x})$ using gradient descent.

Once trained, the flow model can be used to obtain $\hat{y} = -\log p_\theta(\mathbf{x})$ for any $\mathbf{x}$, to form $\{(\mathbf{x}_i, \hat{y}_i)\}$ pairs to train the SR models, which converts the unsupervised density estimation problem to a supervised SR problem. An SR model estimates an expression $\hat{y} = f(\mathbf{x})$, which is related to the energy function as

$$f(\mathbf{x}) = \hat{H}(\mathbf{x}) + \hat{Z} \tag{2}$$

The expression for $\hat{H}(\mathbf{x})$ is obtained by dropping the terms in $f(\mathbf{x})$ which are independent of $\mathbf{x}$.

Score Matching: Score function for any distribution $p(\mathbf{x})$ is defined as $s(\mathbf{x}) = \nabla_{\mathbf{x}} \log p(\mathbf{x})$. An advantage of using $s(\mathbf{x})$ (over $\log p(\mathbf{x})$) is that it is independent of the normalizing constant Z. In score matching (Song & Ermon, 2019; Song et al., 2020), a deep model $s_\theta(\mathbf{x})$, parameterized by θ, estimates the target score $s(\mathbf{x})$. Training is carried out by minimizing the expected square distance,

$$J(\theta) = \frac{1}{2}\mathbb{E}_{p(\mathbf{x})}\|s_\theta(\mathbf{x}) - s(\mathbf{x})\|^2 \tag{3}$$

In the absence of $s(\mathbf{x})$, J can be simplified under certain regularity conditions as detailed in Hyvärinen (2005) by

$$J(\theta) = \mathbb{E}_{p(\mathbf{x})}[\mathrm{Tr}(\nabla_x s_\theta(\mathbf{x})) + \frac{1}{2}\|s_\theta(\mathbf{x})\|^2] \tag{4}$$

where Tr is the trace. Once trained, the model can estimate the score $s_\theta(\mathbf{x}) = \nabla_{\mathbf{x}} \log p_\theta(\mathbf{x})$ for any $\mathbf{x}$ to form pairs $\{(\mathbf{x}_i, \hat{y}_i = -s_\theta(\mathbf{x}_i))\}$ to train the SR models.

The SR model estimates the expression $\hat{y} = f(\mathbf{x})$, which is related to the energy function as

$$f(\mathbf{x}) = \nabla_{\mathbf{x}} \hat{H}(\mathbf{x}) \tag{5}$$

which is free from the normalizing constant. Finally, $\hat{H}(\mathbf{x})$ is estimated by integrating $f(\mathbf{x})$ with respect to $\mathbf{x}$.

Denoising Score Matching: Score matching does not scale well to high dimensional $\mathbf{x}$ due the high cost involved in computing $\mathrm{Tr}(\nabla_x s_\theta(\mathbf{x}))$. To address this, Vincent (2011) proposed denoising score matching approach that perturbs $\mathbf{x}$ with a pre-defined noise distribution $q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})$, where, $\tilde{\mathbf{x}} = \mathbf{x} + \epsilon$ and $\epsilon \sim q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) = \mathcal{N}(\tilde{\mathbf{x}}|\mathbf{x}, \sigma^2 I)$. The score of the perturbed data distribution $q_\sigma(\tilde{\mathbf{x}})$ is learned by minimizing

$$J(\theta; \sigma) = \frac{1}{2}\mathbb{E}_{q_\sigma(\tilde{\mathbf{x}})}[\|s_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})\|_2^2] \tag{6}$$

For small σ, we get $s_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p(\mathbf{x})$. Training is carried out across multiple noise levels, which generates samples from low-density regions of the $p(\mathbf{x})$, thereby improving score estimation.

Once trained, the score model $s_\theta(\mathbf{x})$ can be used to estimate $\hat{H}(\mathbf{x})$ in the same way as described above in the score matching approach.

Mitigating Integration Issues: In score-based models, SR estimates the score function as a symbolic expression, which is subsequently integrated to estimate $\hat{H}(\mathbf{x})$ (see Eq. (5)). However, when SR yields a complex expression for score, its integration becomes analytically challenging. To ameliorate this issue, we propose a reformulation of the modeling approach: instead of modeling the score function, we model $H_\theta(\mathbf{x})$ using a neural network. The score is then obtained as the negative gradient obtained by autograd, $s_\theta(\mathbf{x}) = -\nabla_{\mathbf{x}} H_\theta(\mathbf{x})$. This formulation enables training via standard objectives such as score matching or denoising score matching, as defined in Eqs. (4) and (6).

Factorization: The effectiveness of SR methods reduces drastically with the increase in the size of search space (the number of variables and operations). The lattice models we apply our framework to are large. However, most physical lattices have local interactions which can be leveraged to simplify the SR problem. There are existing approaches leveraging this property for Monte Carlo sampling of lattices (Kennedy & Pendleton, 1985; Faraz et al., 2024). We use the same property for SR, however, allowing the local interactions to extend to any number of neighbours, makes our approach more generalizable.

Given a lattice $\mathbf{x}$ (a matrix, say), the true energy function can be factorized as

$$H(\mathbf{x}) = \sum_l h(x_l | x_{n(l)}) \tag{7}$$

where, h is the local energy function, x_l is the value of the field on lattice site l, and $n(l)$ is the set of neighbors of site l. Fig. 2(a) illustrates this.

During training with SR, $\mathbf{x}$ is factorized into patches and $\hat{H}$ is estimated as

$$\hat{H}(\mathbf{x}) = \sum_l \hat{h}(x_{p(l)}) \tag{8}$$

where $p(l)$ is the set of lattice sites in the patch anchored at site l. Overall, SR estimates $\hat{h}$ over variables $x_{p(l)}$ which are small in number. Fig. 2(b) shows patches of different sizes.

It is to be noted that there is redundancy in this representation. Multiple representations of $\hat{h}(x_{p_l})$ may lead to the same $\hat{H}(\mathbf{x})$. For instance, with a 3×3 patch shown in Fig. 2(c), any $\hat{h}(x_{p_l}) = \sum_{i=0}^{8} a_i g(x_i)$, for any value of $\{a_i\}$ with $\sum_{i=0}^{8} a_i = a$ being constant, leads to the same $\hat{H}(\mathbf{x}) = \sum_l a g(x_l)$. Here, g could be any univariate function. In general, g could be multivariate; e.g., $\hat{h}'(x_{p_l}) = g(x_0, x_4, x_6)$ and $\hat{h}''(x_{p_l}) = g(x_1, x_5, x_7)$, with the same relative positions of lattice sites, lead to the same $\hat{H}(\mathbf{x})$.

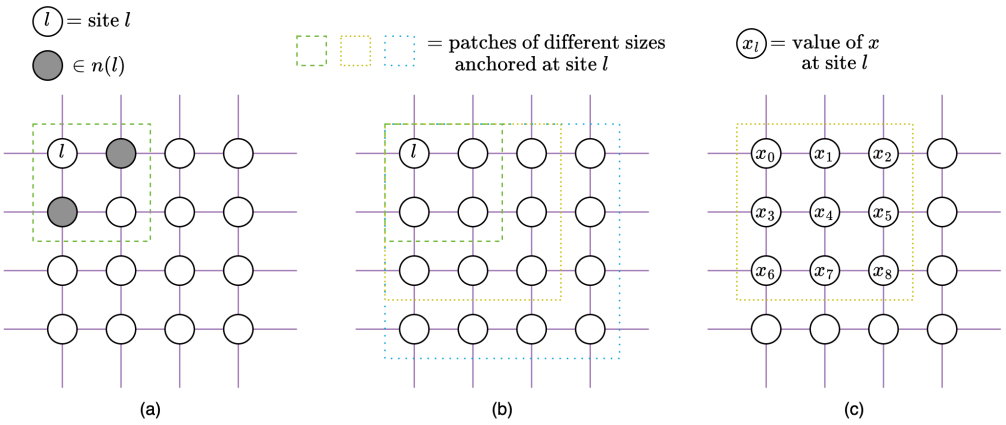


Figure 2: Lattice Factorization. (a) True $H(\mathbf{x})$ can be factorized with a 2×2 patch. (b) While inference with SR, different patch sizes provide different inductive biases. (c) Illustration for redundancy in representation.

In Fig. 1, methods (A) and (C), SR minimizes $\mathbb{E}_{\mathbf{x}}[\|\hat{y}(\mathbf{x}) - \sum_l \hat{h}(x_{p(l)})\|^2]$ to estimate $\hat{h}$, where $\hat{y}(\mathbf{x}) = -\log p_\theta(\mathbf{x})$ is the target estimated by the deep model. It has no target for individual $\hat{h}(x_{p(l)})$ but only matches the sum over the entire lattice. On the other hand, in method (B), SR minimizes $\mathbb{E}_{\mathbf{x},l}[\|s_\theta(\mathbf{x})[l] - f(x_{p(l)})\|^2]$ to estimate f (Eq. equation 5) with target as the l^{th} element of matrix $s_\theta(\mathbf{x})$ estimated by the deep model. Note that there is no redundancy of representation here as we are modeling site-wise gradients and not the lattice likelihood.

Learning from low density regions: Robust SR needs samples from both high and low values of $H(\mathbf{x})$. The original samples $\mathbf{x} \sim p(\mathbf{x})$ represent regions with low energy, but high energy regions are conspicuous by their absence. Hence, we append the original set $\mathcal{D}$ with noisy perturbations $\mathbf{x}' = \mathbf{x} + \epsilon$ where $\epsilon \sim \mathcal{N}(\mathbf{0}; \sigma^2 I)$. Then, we estimate $\log p_\theta(\mathbf{x}')$ or $s_\theta(\mathbf{x}')$ using the deep models, and thus, form the input-output pairs needed for SR.

Algorithm 1 outlines the training procedure for symbolic estimation of unnormalised distributions, combining deep generative modelling with SR. The method first learns a density or score model from data, augments low-density regions, and subsequently recovers an analytical expression for the Hamiltonian via SR.

3 Case Studies

In this section we describe five specific examples where we apply our framework.

Multivariate Gaussian density.

$$p(\mathbf{x}; \mu, \mathbf{\Sigma}) = \frac{1}{(2\pi)^{N/2}|\mathbf{\Sigma}|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\mu)^T \mathbf{\Sigma}^{-1}(\mathbf{x}-\mu)} \tag{9}$$

Fixing $\mu = [-1.5, 1.5]^T$, we use two variants of $\mathbf{\Sigma}$ with $\mathbf{x} \in \mathbb{R}^2$. In the first variant, we use a diagonal $\mathbf{\Sigma}$ as I, resulting in $H(\mathbf{x}) = 0.5(x_1 + 1.5)^2 + 0.5(x_2 - 1.5)^2$. In the second one, we use a full $\mathbf{\Sigma}$ as $\begin{bmatrix} 1.0 & 0.5 \\ 0.5 & 1.0 \end{bmatrix}$, leading to $H(\mathbf{x}) = 0.5(x_1 + 1.5)^2 + 0.5(x_2 - 1.5)^2 + 0.5(x_1 + 1.5)(x_2 - 1.5)$. We apply our framework to the data sampled from these distributions, and expect it to recover $H(\mathbf{x})$.

Many Well Distribution (n-dimensional). It is a synthetic distribution given by the product of $n/2$ copies of the 2-D double well distribution (Noé et al., 2019; Wu et al., 2020). The Hamiltonian of 2-D double well distribution is

$$H(\mathbf{x}) = x_1^4 - 6x_1^2 - 0.5x_1 + 0.5x_2^2, \qquad \mathbf{x} \in \mathbb{R}^2 \tag{10}$$

Algorithm 1 Symbolic Function Estimation for Unnormalised Distributions

Input: Data $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^N$ from distribution
Stage 1: Training the Generative Model
Initialise: Parameters θ of estimator M (NF–based or score-based)
while $\mathcal{L}_M(\theta)$ not converged **do**
    Sample a mini-batch from $\mathcal{D}$
    Compute loss $\mathcal{L}_M(\theta)$
    Update parameters:

$$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_M(\theta)$$

end while
Stage 2: Low-density Sample Augmentation
Generate perturbed samples

$$\mathbf{x}' \leftarrow \mathbf{x} + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 I)$$

Construct augmented dataset $\mathcal{D}_{\text{aug}} = \mathcal{D} \cup \{\mathbf{x}'_i\}_{i=1}^N$
Stage 3: Label Construction
Evaluate the trained estimator M to obtain predicted quantities

$$\hat{y}_i \in \{-\text{NLL}(\mathbf{x}_i),\ -\hat{s}(\mathbf{x}_i)\}$$

Form regression dataset

$$\mathcal{D}' = \{(\mathbf{x}_i, \hat{y}_i)\}$$

Stage 4: Symbolic Regression
Initialise: Parameters of the SR model
if local factorisable distribution **then**
    Get local patches from $\mathcal{D}'$ ▷ Use local Model Factorization
end if
Optimise SR parameters on $\mathcal{D}'$ using MSE loss
Recover analytical expression $\hat{y} = f(\mathbf{x})$
Estimate Hamiltonian $\hat{H}(\mathbf{x})$ from $f(\mathbf{x})$
Return: $\hat{H}(\mathbf{x})$

and that of n-D many well distribution is

$$H(\mathbf{x}) = \sum_{i=1}^{n/2} \left(x_{2i-1}^4 - 6x_{2i-1}^2 - 0.5x_{2i-1} + 0.5x_{2i}^2 \right), \qquad \mathbf{x} \in \mathbb{R}^n \tag{11}$$

We conduct experiments for $n = 2, 4, 8, 16, 32, 64$, expecting the framework to recover $H(\mathbf{x})$.

XY Model. It is a theoretical model used in statistical mechanics (Kosterlitz & Thouless, 1973) to describe two-dimensional systems of interacting spins and to study phase transitions. It is widely employed to model Boltzmann distributions on lattices (Kanaujia et al., 2024; Singh et al., 2021; Beach et al., 2018). Here, $\mathbf{x} \in [0, 2\pi)^d$ represents the spin configuration on a square 2-D grid with d sites, and the energy of the system is given by

$$H(\mathbf{x}) = -0.5\lambda \sum_{l=1}^{d} \sum_{l' \in n(l)} \cos(x_l - x_{l'}) \tag{12}$$

Here, λ is the coupling constant that quantifies interaction strength and $n(l)$ is a set of two neighbors (right and below, cf. Fig. 2(a)) of site l. We consider a grid of size 8×8, i.e., $d = 64$, with $\lambda = 1/1.4$. A patch p of size 2×2 is used to factorize $\hat{H}(\mathbf{x})$.

Scalar ϕ^4 Theory. It is a widely used in computational Physics (Albergo et al., 2019; Singha et al., 2023) to study scalar fields. With $\mathbf{x} \in \mathbb{R}^d$ as a scalar field on a square 2-D lattice with d sites, the energy function

(or action) is given by

$$H(\mathbf{x}) = \sum_{l=1}^{d} \left(\lambda_1 x_l^4 + \lambda_2 x_l^2 + 2 \sum_{l' \in n(l)} (x_l^2 - x_l x_{l'}) \right) \tag{13}$$

where $n(l)$ denotes the two neighboring sites (right and below; see Fig. 2(a)). Here, λ_1 is the coupling constant and λ_2 represents the squared mass of the scalar particle associated with the field. We conduct experiments on lattices of size 8×8, i.e., $d = 64$, at two different values of λ_2, namely -4 and 1, while λ_1 is fixed at 4. Here, a 2×2 patch p is sufficient to recover $H(\mathbf{x})$ but we also experiment with patches of different sizes. Larger patches allow recovering distant interactions but increase the number of variables, making it difficult for SR to recover the true equation.

Renormalization. Renomalization in statistical Physics is akin to zooming out in image processing. It enables one to study the phenomena on larger scales by reducing the resolution of lattices (called coarse-graining) for computational efficiency. Empirically, a finer lattice is coarsened by some kind of pooling; generally average pooling is used. A fine lattice of size $\sqrt{d} \times \sqrt{d}$, on pooling with a 2×2 filter, reduces to a coarser lattice of size $\sqrt{d}/2 \times \sqrt{d}/2$.

Traditionally, the effective action $H_{\text{eff}}(\mathbf{x})$ at the coarser scale is estimated analytically by integrating out terms in the original action $H(\mathbf{x})$. The effectiveness of this method is substantially limited by the extent of available analytical techniques. Our framework, however, allows one to estimate $H_{\text{eff}}(\mathbf{x})$ directly from coarse-grained lattice samples.

We test our framework to find $H_{\text{eff}}(\mathbf{x})$ for highly correlated ϕ^4 theory with $\lambda_1 = 4$ and $\lambda_2 = 1$, which lies in the non-perturbative regime where perturbative analysis fails. We start with $N = 64 \times 64$ lattice samples from Eq. (13), i.e., $d = 4096$, and coarse-grain them to $N = 32 \times 32$ lattice samples by average pooling. These coarse-grain samples are then used as $\mathcal{D}$ to estimate $H_{\text{eff}}(\mathbf{x})$. These samples are progressively coarse-grained to $N = 16 \times 16$, and then, to $N = 8 \times 8$ to see how $H_{\text{eff}}(\mathbf{x})$ evolves to those lattice scale.

4 Experiments and Results

Most existing methods for symbolic density estimation assume a relatively simple parametric form for the underlying probability density function. In our evaluation, we use MeSSY (Tohme et al., 2024) as a baseline. MeSSY constructs an ansatz for the energy of the target distribution as a weighted linear combination of predefined basis functions, i.e., a set of linearly independent functions (e.g., $x, x^2, \sin(x), \cos(x)$). The corresponding weights are estimated by maximising the log-likelihood over the available samples. In contrast, the proposed approach integrates deep generative models with symbolic regression to estimate the analytic expression of density, using a rich set of operations (e.g., $\times, \sin, \cos, \log, \exp$) and relatively simpler basis functions. This enables modeling complicated energy functions (e.g., XY model), and is not limited to only linear combinations of basis functions.

In this section, we present experimental results across a range of datasets, spanning from low-dimensional distributions ($d = 2$) to high-dimensional ones ($d = 1024$). These experiments validate the effectiveness of the proposed framework, highlighting its ability to handle increasing complexity and scale while maintaining performance. We employ normalising flows and score-based models as the underlying generative components, which are coupled with two distinct SR frameworks, viz., PySR (genetic algorithm driven) (Cranmer, 2023) and EQL (neural network based) (Kim et al., 2021), to form the integrated pipeline.

The models are evaluated based on how accurately the estimate $\hat{H}(\mathbf{x})$ approximates the true Hamiltonian $H(\mathbf{x})$. Performance is quantified using the mean squared error (MSE) and the coefficient of determination (R^2). The R^2 score is defined as

$$R^2(H, \hat{H}; \mathcal{D}) = 1 - \frac{\sum_{\mathbf{x} \in \mathcal{D}} \left(H(\mathbf{x}) - \hat{H}(\mathbf{x}) \right)^2}{\sum_{\mathbf{x} \in \mathcal{D}} \left(H(\mathbf{x}) - \bar{H} \right)^2} \tag{14}$$

Table 1: Results and estimated expressions $\hat{h}([x_1, x_2]; \lambda)$ for a 2-D Gaussian distribution using various methods. The mean vector is fixed at $\mu = [-1.5, 1.5]^T$, with two different covariance matrices, Σ_1 and Σ_2, as detailed in the table. The target functions are: (A) $0.5x_1^2 + 0.5x_2^2 + 1.5x_1 - 1.5x_2$, and (B) $0.67x_1^2 - 0.67x_1x_2 + 0.67x_2^2 + 3.0x_1 - 3.0x_2$. Incorrect terms are shown in gray. NF = Flow model, SM = Score matching.

		MSE(↓)	R^2(↑)	Estimated $\hat{h}(x_{p(l)})$
$\Sigma = \begin{bmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{bmatrix}$	MeSSY	0.00	0.99	$0.48x_1^2 - 0.02x_1x_2 + 0.50x_2^2 + 1.39x_1 - 1.41x_2$
	NF + PySR	0.00	0.99	$0.5x_1^2 + 0.5x_2^2 + 1.5x_1 - 1.49x_2$
	NF + EQL	0.00	0.99	$0.5x_1^2 + 1.52x_1 + 0.5x_2^2 - 1.5x_2$
	SM + PySR	0.00	0.99	$0.51x_1^2 + 0.49x_2^2 + 1.53x_1 - 1.5x_2$
	SM + EQL	0.00	0.99	$0.49x_1^2 + 1.47x_1 + 0.52x_2^2 - 1.52x_2$
$\Sigma = \begin{bmatrix} 1.0 & 0.5 \\ 0.5 & 1.0 \end{bmatrix}$	MeSSY	0.01	0.99	$0.72x_1^2 - 0.85x_1x_2 + 0.63x_2^2 + 3.38x_1 - 3.21x_2$
	NF + PySR	0.00	0.99	$0.67x_1^2 - 0.66x_1x_2 + 2.98x_1 + 0.67x_2^2 - 2.98x_2$
	NF + EQL	0.00	0.99	$0.67x_1^2 - 0.68x_1x_2 + 3.02x_1 + 0.66x_2^2 - 2.97x_2$
	SM + PySR	0.00	0.99	$0.67x_1^2 - 0.70x_1x_2 + 2.99x_1 + 0.67x_2^2 - 3.02x_2$
	SM + EQL	0.00	0.99	$0.66x_1^2 - 1.34x_1x_2 + 2.97x_1 + 0.66x_2^2 - 2.98x_2$

where $\bar{H} = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[H(\mathbf{x})]$ denotes the empirical mean of the true Hamiltonian over the dataset $\mathcal{D}$. A smaller MSE and a higher R^2 indicate closer agreement between $\hat{H}(\mathbf{x})$ and $H(\mathbf{x})$. For qualitative evaluation, the recovered functional forms are also presented for visual comparison.

Multivariate Gaussian density: We train both flow-based and score-matching models on 2.5×10^4 samples drawn from a two-dimensional Gaussian distribution. For symbolic regression (SR) with the EQL model, all 2.5×10^4 samples are used for training. The proposed approach successfully recovers the true functional form of $H(\mathbf{x})$. Evaluation on an independent test set comprising 2×10^4 samples yields a mean squared error (MSE) close to zero and a coefficient of determination (R^2) close to one. In comparison, PySR attains comparable performance using only 10^3 samples. These results hold consistently for both Gaussian distributions considered—those with diagonal and non-diagonal covariance matrices $\mathbf{\Sigma}$. Both methods recover the true $H(\mathbf{x})$ without the need for noise perturbation to account for low-density regions. The quantitative results are presented in Table 1.

Experiments with the baseline, MeSSY, also yield MSE values close to zero and R^2 near one; however, the recovered expression deviates from the ground truth. In particular, the coefficients of several polynomial terms differ from those in the true $H(\mathbf{x})$, and the predicted expression contains additional spurious multinomial terms.

Many Well Distribution (n-dimensional): We train flow, score-matching, and denoising score-matching models with 10^5 samples from MW distribution. For SR, we employ factorisation using patches of size 2×1 to reduce the search space with this inductive bias. Evaluation is performed on 2×10^4 test samples, and results are summarised in Table 2.

For lower dimensions ($n = 2, 4, 8$), both flow and score-matching models, as well as SR approaches (PySR and EQL), successfully recover the true $H(\mathbf{x})$. EQL is trained using all 10^5 samples, whereas PySR achieves comparable performance with only 10^3 samples. No noise perturbation is required.

For higher dimensions ($n = 16, 32, 64$), the denoising score-matching model outperforms the flow model, likely due to the highly multimodal nature of $p(\mathbf{x})$ with $2^{n/2}$ modes. Flow models trained with forward KL divergence are susceptible to mode-covering behavior (Midgley et al., 2023; Kanaujia et al., 2024). In these cases, PySR also surpasses EQL, reflecting the advantage of the broader search capability of genetic algorithms over the gradient-based optimization used in EQL.

While MeSSY is able to recover the general structure of $H(\mathbf{x})$ for $n = 2$, it fails to converge and recover expression for $n > 2$. However, even for $n = 2$, the coefficients of several terms differ from those in the true $H(\mathbf{x})$, and the recovered expression includes additional multinomial terms (c.f. Table 2).

Table 2: Results for Many Well distribution with various dimensions, $n = 2, 4, 8, 16, 32, 64$. The true function is $h(x_{p_l}) = x_1^4 - 6x_1^2 - 0.5x_1 + 0.5x_2^2$. Estimated terms that do not match the true function are shown in gray. NF = Flow model, SM = Score matching.

n	Model	MSE(↓)	R^2(↑)	Estimated Equation $\hat{h}(x_{p(l)})$
2	MeSSY	0.02	0.98	$0.96x_1^4 + 0.1x_1^3 - 5.77x_1^2 - 0.83x_1 + 0.61x_2^2 - 0.23x_2$
	NF + PySR	0.00	0.99	$x_1^4 - 6.02x_1^2 - 0.5x_1 + 0.5x_2^2$
	NF + EQL	0.00	0.99	$0.99x_1^4 - 5.95x_1^2 - 0.49x_1 + 0.52x_2^2$
	SM + PySR	0.00	0.99	$0.98x_1^4 - 5.9x_1^2 - 0.51x_1 + 0.5x_2^2$
	SM + EQL	0.00	0.99	$0.99x_1^4 - 5.9x_1^2 - 0.59x_1 + 0.48x_2^2$
4	NF + PySR	0.00	0.97	$1.02x_1^4 - 6.11x_1^2 - 0.52x_1 + 0.52x_2^2$
	NF + EQL	0.00	0.99	$1.0x_1^4 - 6.03x_1^2 - 0.47x_1 + 0.46x_2^2$
	SM + PySR	0.01	0.89	$1.04x_1^4 - 6.21x_1^2 - 0.5x_1 + 0.5x_2^2$
	SM + EQL	0.48	0.75	$1.0x_1^4 - 6.03x_1^2 - 0.69x_1 + 0.50x_2^2$
8	NF + PySR	0.00	0.90	$1.0x_1^4 - 6.02x_1^2 - 0.55x_1 + 0.55x_2^2$
	NF + EQL	0.23	0.96	$0.92x_1^4 - 5.58x_1^2 - 0.86x_1 + 0.14x_1^2x_2^2 + 0.11x_1^2x_2 + 0.24x_1x_2 - 0.2x_2$
	SM + PySR	0.01	0.99	$x_1^4 - 6.03x_1^2 - 0.46x_1 + 0.5x_2^2$
	SM + EQL	0.42	0.93	$0.99x_1^4 - 0.11x_1^3 - 5.97x_1^2 + 0.5x_2^2$
16	NF + PySR	0.51	0.96	$1.0x_1^4 - 6.01x_1^2 - 0.67x_1 + 0.59x_2^2$
	NF + EQL	0.67	0.94	$0.78x_1^4 - 4.78x_1^2 - 0.53x_1 + 0.64x_2^2 + 0.11x_1^2x_2 - 0.26x_2$
	SM + PySR	0.65	0.94	$1.26x_1^4 - 5.80x_1^2 - 0.36x_1 + 0.50x_2^2 - 0.06x_1^6$
	SM + EQL	0.06	0.99	$x_1^4 - 5.95x_1^2 - 0.41x_1 + 0.5x_2^2$
32	NF + PySR	4.63	0.80	$0.71x_1^4 - 4.47x_1^2 - 0.20x_1 + 0.34x_2^2$
	NF + EQL	5.29	0.77	$0.35x_1^4 - 2.93x_1^2 + 1.05x_2^2 - 0.33x_1^3x_2 - 0.33x_1^2x_2^2 - 0.11x_1^2x_2 + 0.48x_1x_2 + 0.36x_2$
	SM + PySR	0.48	0.98	$0.79x_1^4 - 4.73x_1^2 - 0.45x_1 + 0.50x_2^2 - 0.02x_2$
	SM + EQL	0.69	0.99	$0.87x_1^4 - 5.63x_1^2 - 1.13x_1 + 0.49x_2^2 + 0.11x_1^6 - 0.18x_1^5 + 0.54x_1^3$
64	NF + PySR	1.67	0.93	$0.13x_1^4 - 2.06x_1^2 + 1.16x_1 + 0.43x_2^2 + 0.04x_1^6 + 0.19x_1^5 - 1.14x_1^3$
	NF + EQL	3.18	0.93	$-0.93x_1^4 + 0.13x_1^2 - 0.21x_1 + 0.19x_2^2 + 0.12x_1^3x_2 - 0.48x_1x_2$
	SM + PySR	0.53	0.99	$1.37x_1^4 - 6.56x_1^2 - 0.47x_1 + 0.51x_2^2 - 0.06x_1^6 - 0.02x_1^2x_2 + 0.05x_2$
	SM + EQL	0.57	0.99	$1.03x_1^4 - 6.01x_1^2 - 1.64x_1 + 0.49x_2^2 - 0.22x_1^5 + 0.64x_1^3 + 0.23x_1x_2$

XY Model: We experiment on lattices of size 8×8 ($d = 64$) with $\lambda = 1/1.4$. The approximate energy function $\hat{H}(\mathbf{x})$ is factorized using patches of size 2×2 (c.f. Fig. 2(a)). We use 10^5 samples for training flow and score-based models. For score-based models, we find that parameterizing $H_\theta(\mathbf{x})$ with a neural network and obtaining $\nabla_{\mathbf{x}} H_\theta(\mathbf{x})$ via automatic differentiation yields better performance than directly modeling $s_\theta(\mathbf{x})$, as the latter approach suffers from integration issues due to nested trigonometric terms.

Denoising score matching outperforms standard score matching, while overall the flow model achieves superior results compared to score-based methods. Among symbolic regression approaches, PySR consistently performs better than EQL. The quantitative results are summarized in Table 3. In contrast, MeSSY fails to converge and is unable to recover the target expression.

Scalar ϕ^4 Theory: We use 10^5 samples for training flow and score-based models. Here, we train the score-based model with denoising score matching. For SR, the distribution is factorised over patches: a 3×3 patch is used for estimating $\nabla_{\mathbf{x}} \hat{H}(\mathbf{x})$ via the score model, and a 2×2 patch for estimating $\hat{H}(\mathbf{x})$ via the flow model. Results are summarised in Table 4.

Both flow and score-based models accurately recover the target energy function, while PySR consistently outperforms EQL. For SR, a 2×2 patch provides the most effective inductive bias for the ϕ^4 theory. However, we also assess the evaluation with larger patches, i.e., 3×3 and 4×4 to test the generalisability of the method. The results for $\lambda_1 = 4, \lambda_2 = 1$ with the flow model using different patch sizes are reported in Table 5. Performance remains strong for both PySR and EQL, though larger patches lead to increased search

Table 3: Results for XY Model. The true function is $h(x_{p(l)}) = -0.71\cos(x_0 - x_1) - 0.71\cos(x_0 - x_2)$. Estimated terms that do not match the true function are shown in gray. NF = Flow model, DSM = Denoising score model.

Model	MSE(↓)	R^2(↑)	Estimated $\hat{h}(x_{p(l)})$
MeSSY	-	-	-
NF + PySR	**4.83**	**0.92**	$-0.63\cos(0.9x_0 - 0.9x_1) - 0.63\cos(x_0 - x_2)$
DSM + PySR	42.22	0.30	$-0.27\cos(x_0 - x_1) - 0.04x_0x_3 + 0.04x_3^2$
NF + EQL	6.74	0.87	$-0.72\cos(x_0 - x_1) - 0.72\cos(x_0 - x_2) + 0.11\cos(x_0 - x_3) + 0.19\cos^2(x_3) - 0.28\cos(x_0) + 0.32\cos(x_3)$
DSM + EQL	47.72	0.21	$0.21 - 0.46\cos(0.34x_0 - 0.59x_2 + 0.30x_3 + 0.64\cos(0.48x_0 - 0.61x_2 - 0.33x_3)$

complexity. In practical use cases where one may not know the range of interactions, a larger patch size is prescribed. Larger patches, on the other hand, make the search space for SR very large. These experiments support that the SR methods are able to recover the true equation, or at least the important interaction terms thereof, even with larger patches. On the other hand, MeSSY fails to converge and is unable to recover the target expression.

Table 4: Results for ϕ^4 theory with various λ_1, λ_2 settings. The true expressions are: (A) for $\lambda_1 = 4, \lambda_2 = -4$, $h(x_{p(l)}) = 4x_0^4 - 2x_0x_1 - 2x_0x_2$, and (B) for $\lambda_1 = 4, \lambda_2 = 1$, $h(x_{p(l)}) = 4x_0^4 + 5x_0^2 - 2x_0x_1 - 2x_0x_2$. Incorrect terms are highlighted in gray. NF = Flow model, DSM = Denoising score model.

	Model	MSE(↓)	R^2(↑)	Estimated $\hat{h}(x_{p(l)})$
(A)	MeSSY	-	-	-
	NF + PySR	0.02	0.99	$3.91x_0^4 - 1.97x_0x_1 - 1.97x_0x_2$
	NF + EQL	0.29	0.99	$3.42x_0^4 - 2.33x_0x_1 - 2.33x_0x_2 + 0.29x_0x_1^3 + 0.40x_1^3x_3 - 0.13x_0x_3 + 0.49x_1x_2$
	DSM + PySR	0.01	0.99	$3.95x_0^4 - 1.98x_0x_1 - 1.98x_0x_2$
	DSM + EQL	0.06	0.99	$3.65x_0^4 - 2.01x_0x_1 - 2.03x_0x_2 + 0.12x_0^2$
(B)	MeSSY	-	-	-
	NF + PySR	0.00	0.99	$4.01x_0^4 + 5.17x_0^2 - 2.06x_0x_1 - 2.06x_0x_2$
	NF + EQL	0.07	0.99	$3.69x_0^4 + 5.27x_0^2 - 2.01x_0x_2 - 2.14x_0x_1 - 0.68x_1x_2^3 - 0.27x_2^3x_3 + 0.14x_0x_3 + 0.45x_1x_2$
	DSM + PySR	0.03	0.99	$3.12x_0^4 + 5.35x_0^2 - 1.92x_0x_1 - 1.92x_0x_2$
	DSM + EQL	0.05	0.99	$3.07x_0^4 + 5.48x_0^2 - 1.80x_0x_1 - 1.78x_0x_2 - 0.59x_0^3x_1 - 0.54x_0^3x_2 - 0.65x_0^2x_1x_2 - 0.16x_0^2x_1^2 - 0.16x_0^2x_2^2$

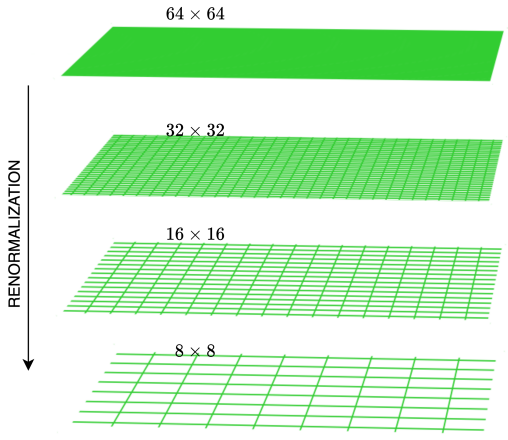
Renormalization: Given the comparable performance of flow-based and score-based models for the ϕ^4 theory, we employ only the flow model for the renormalisation study. For experiments, we use 10^5 samples of 64×64 lattice size. These lattices are progressively coarsened to multiple scales, namely, $d = 32 \times 32$, 16×16 and 8×8 by applying 2×2 average pooling. Separate flow models are trained at each lattice scale. Subsequently, symbolic regression is applied using PySR and EQL, though we report only PySR results due to better convergence and lower mean squared error between $H_\theta(\mathbf{x})$ (from the flow model) and $\hat{H}(\mathbf{x})$ (from SR). A 2×2 patch is used for lattice factorization.

As the lattice size decreases, perturbative theory predicts a reduction in the coefficient of x_l^4, an increase in the coefficient of x_l^2, and approximately constant interaction coefficients. Our results exhibit the same trend even in the non-perturbative regime, as shown in Fig. 3. At smaller lattice sizes, the coefficient of x_l^4 becomes negligible and is dropped by the method, validating the consistency of the estimated $\hat{H}(\mathbf{x})$.

Table 5: Results for estimating $\hat{h}(x_{p(l)})$ for ϕ^4 theory with different patch sizes $|p|$ for flow model. True $h(x_{p(l)}) = 4x_0^4 + 5x_0^2 - 2x_0x_1 - 2x_0x_2$. Incorrect terms are shown in gray.

	$\lvert p\rvert$	MSE(↓)	R^2(↑)	Estimated $\hat{h}(x_{p(l)})$
PySR	2×2	0.003	0.99	$4.01x_0^4 + 5.17x_0^2 - 2.06x_0x_1 - 2.06x_0x_2$
	3×3	0.006	0.99	$4.11x_0^4 + 5.06x_0^2 - 2.04x_0x_1 - 2.04x_0x_2$
	4×4	0.008	0.99	$3.90x_0^4 + 5.14x_0^2 - 1.99x_0x_1 - 1.99x_0x_2$
EQL	2×2	0.07	0.99	$3.69x_0^4 + 5.27x_0^2 - 2.01x_0x_2 - 2.14x_0x_1 - 0.68x_1x_2^3 - 0.27x_2^3x_3 + 0.14x_0x_3 + 0.45x_1x_2$
	3×3	0.98	0.98	$3.86x_0^4 + 5.21x_0^2 - 2.1x_0x_1 - 1.99x_0x_2 + 0.42x_1x_2 - 1.01x_0^3x_2 - 0.14x_0^3x_1 - 0.13x_0^3x_5$
	4×4	0.11	0.99	$3.61x_0^4 + 4.97x_0^2 - 2.09x_0x_1 - 2.03x_0x_2 + 0.11x_0^3x_3 - 0.12x_0^3x_8 - 0.1x_0x_9 + 0.28x_1x_4 + 0.27x_0x_5 + 0.14x_0x_2 + 0.13x_0x_8$

To probe potential higher-order interactions, we repeat the analysis using a 3×3 patch for SR factorization. The resulting action retains the same functional form as that obtained with a 2×2 patch, indicating the absence of additional long-range and higher-order interaction terms.



For $N = 64^2$, True $h(x_{p(l)}) = 4x_0^4 + 5x_0^2 - 2x_0x_1 - 2x_0x_2$

N	MSE	R^2	Estimated $\hat{h}_{\text{eff}}(x_{p(l)})$
32^2	1.09	1.00	$0.99x_0^4 + 6.38x_0^2 - 1.99x_0x_1 - 1.89x_0x_2$
16^2	0.62	1.00	$18.91x_0^2 - 2.14x_0x_1 - 2.143x_0x_2$
8^2	0.31	0.99	$33.34x_0^2 - 2.20x_0x_0x_1 - 2.20x_0x_2$

Figure 3: Renormalization results for non-perturbative region, $\lambda_1 = 4$ and $\lambda_2 = 1$.

5 Discussion

Error propagation in two-stage learning: The proposed framework includes learning at two stages, first the deep generative model, and then, the SR model. Previous works as in Cranmer et al. (2020) report that using SR alone performs much worse than using it in conjunction with deep models for large scale learning. We find the same with our attempts to apply SR directly to maximize the likelihood. Here, we investigate how errors propagate through these two stages in our framework. Table 6 compares the MSE after flow model and that after the SR stage for various datasets. We use mean subtraction to eliminate the constant term because of normalizing constant Z. We see that after SR, with either PySR or EQL, the MSE is less as compared to the MSE just after the flow model. A plausible explanation could be that SR, by its low complexity estimates, reduces the noise in the likelihood information obtained from the flow model. These results further support the use of SR methods instead of only black box deep learning based models for generating samples.

Table 6: Comparison of MSE(H, H_θ), where H_θ is computed by flow model, and MSE$(H, \hat{H})$, where $\hat{H}$ is estimated by SR methods (PySR and EQL) using flow model outputs.

Dataset	MSE(H, H_θ)	MSE$(H, \hat{H})$	
		PySR	EQL
Many Well, $N = 2$	0.01	0.00	0.00
Many Well, $N = 4$	0.05	0.00	0.00
Many Well, $N = 8$	1.60	0.00	0.23
XY; $\lambda = 1.4^{-1}$	9.79	4.83	6.74
$\phi^4; \lambda_1 = 4, \lambda_2 = -4$	1.43	0.02	0.29
$\phi^4; \lambda_1 = 4, \lambda_2 = 1$	0.22	0.00	0.07

We also conduct an extensive analysis of the robustness, repeatability, and overall reproducibility of the proposed framework. These aspects are examined thoroughly through a series of experiments presented in Appendix D. In addition, we also investigate the impact of various hyperparameters on the performance of the method, with a detailed discussion provided in Appendix C.

6 Conclusion

We present a general framework for using deep generative models to estimate equations for energy or Hamiltonian of the system from samples. We have demonstrated the use of likelihood based and score based models with symbolic regression methods while utilizing density factorization and sample perturbation. Experiments support the effectiveness of the proposed approach in reliably estimating the expressions for the Hamiltonian or action of XY model and ϕ^4 theory, and density functions for some popular toy distributions. As an important contribution towards the challenges the theorists face in discretizing the continuous theories, we demonstrate the ability of the proposed framework to recover the continuous action from discrete samples of ϕ^4 theory. Further, the renormalization experiments discover new actions for the non-perturbative regions of the theory at different scales. An important limitation of the current work stems from the inability of existing SR methods to deal with very large number of variables. This hinders the approach in discovering long-range or non-local interaction terms in lattice models as they may require using very large patches for factorization. Nevertheless, our experiments yield good results with patches of size up to 4×4. Future works include studying how the mode covering behavior of deep generative models affects the estimated equations, and extending the proposed framework to non-equilibrium systems, which evolve dynamically.

7 Impact Statement

The potential of this work extends to many areas in computational sciences, including Physics, chemistry and climate science. A common concern for traditional scientists is the inability of machine learning based approaches to give insights into the phenomenon under study. The proposed method gives them both the ability to analyze a large amount of data with machine learning and that to estimate the symbolic equations. The paper includes examples of systems where analytic derivation of equations is not feasible traditionally, but is made feasible by the presented approach. We believe this work can be taken up for follow up studies in the following areas:

- Condensed matter Physics to study phase transitions and renormalization theories
- Nuclear Physics to derive interaction potentials from scattering data
- Climate sciences to model extreme climatic conditions from weather data

References

Tara Akhound-Sadegh, Jarrid Rector-Brooks, Joey Bose, Sarthak Mittal, Pablo Lemos, Cheng-Hao Liu, Marcin Sendera, Siamak Ravanbakhsh, Gauthier Gidel, Yoshua Bengio, et al. Iterated denoising energy matching for sampling from boltzmann densities. In *International Conference on Machine Learning*, pp. 760–786. PMLR, 2024.

M. S. Albergo, G. Kanwar, and P.E. Shanahan. Flow-based generative models for markov chain monte carlo in lattice field theory. *Physical Review D*, 100, 2019.

Matthew J. S. Beach, Anna Golubeva, and Roger G. Melko. Machine learning vortices at the kosterlitz-thouless transition. *Phys. Rev. B*, 97:045207, 2018.

Luca Biggio, Tommaso Bendinelli, Alexander Neitz, Aurelien Lucchi, and Giambattista Parascandolo. Neural symbolic regression that scales. In *International Conference on Machine Learning*, pp. 936–945. Pmlr, 2021.

Siddhartha Chib and Edward Greenberg. Understanding the metropolis-hastings algorithm. *The american statistician*, 49:327–335, 1995.

Miles Cranmer. Interpretable machine learning for science with pysr and symbolicregression. jl. *arXiv preprint arXiv:2305.01582*, 2023.

Miles Cranmer, Alvaro Sanchez Gonzalez, Peter Battaglia, Rui Xu, Kyle Cranmer, David Spergel, and Shirley Ho. Discovering symbolic models from deep learning with inductive biases. *Advances in neural information processing systems*, 33:17429–17442, 2020.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *International Conference on Learning Representations*, 2017.

Ali Faraz, Ankur Singha, Dipankar Chakrabarti, Shinichi Nakajima, and Vipul Arora. Improvement of heatbath algorithm in lft using generative models. *Proc. of Science, 41st Intl. Symp. on Lattice Field Theory, LATTICE 2024*, pp. 1–10, 2024.

P Hasenfratz and F Niedermayer. Perfect lattice action for asymptotically free theories. *Nuclear Physics B*, 414:785–814, 1994.

Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6:695–709, 2005.

Pierre-Alexandre Kamienny, Stéphane d'Ascoli, Guillaume Lample, and François Charton. End-to-end symbolic regression with transformers. *Advances in Neural Information Processing Systems*, 35:10269–10281, 2022.

Vikas Kanaujia and Vipul Arora. Scorenf: Score-based normalizing flows for sampling unnormalized distributions. *arXiv preprint arXiv:2510.21330*, 2025.

Vikas Kanaujia, Mathias S Scheurer, and Vipul Arora. Advnf: Reducing mode collapse in conditional normalising flows using adversarial learning. *SciPost Physics*, 16:132–160, 2024.

AD Kennedy and BJ Pendleton. Improved heatbath method for monte carlo calculations in lattice gauge theories. *Physics Letters B*, 156:393–399, 1985.

Samuel Kim, Peter Y. Lu, Srijon Mukherjee, Michael Gilbert, Li Jing, Vladimir Čeperić, and Marin Soljačić. Integration of neural network-based symbolic regression in deep learning for scientific discovery. *IEEE Transactions on Neural Networks and Learning Systems*, 32:4166–4177, 2021.

Ivan Kobyzev, Simon J.D. Prince, and Marcus A. Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43:3964–3979, 2021.

J M Kosterlitz and D J Thouless. Ordering, metastability and phase transitions in two-dimensional systems. *Journal of Physics C: Solid State Physics*, 6:1181–1203, 1973.

John R Koza. Genetic programming as a means for programming computers by natural selection. *Statistics and computing*, 4:87–112, 1994.

William La Cava, Lee Spector, and Kourosh Danai. Epsilon-lexicase selection for regression. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, pp. 741–748, 2016.

William La Cava, Thomas Helmuth, Lee Spector, and Jason H Moore. A probabilistic and multi-objective analysis of lexicase selection and ε-lexicase selection. *Evolutionary Computation*, 27:377–402, 2019.

Georg S Martius and Christoph Lampert. Extrapolation and learning equations. In *5th International Conference on Learning Representations, ICLR 2017-Workshop Track Proceedings*, 2017.

Laurence Illing Midgley, Vincent Stimper, Gregor NC Simm, Bernhard Schölkopf, and José Miguel Hernández-Lobato. Flow annealed importance sampling bootstrap. In *The Eleventh International Conference on Learning Representations*, 2023.

Radford M Neal. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2:2, 2011.

Frank Noé, Simon Olsson, Jonas Köhler, and Hao Wu. Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. *Science*, 365:1147, 2019.

Sankar K. Pal and Paul P. Wang. *Genetic Algorithms for Pattern Recognition*. CRC Press, 2017.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22:1–64, 2021.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pp. 1530–1538. PMLR, 2015.

Subham Sahoo, Christoph Lampert, and Georg Martius. Learning equations for extrapolation and control. In *International Conference on Machine Learning*, pp. 4442–4450. PMLR, 2018.

Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *science*, 324 (5923):81–85, 2009.

Michael Schmidt and Hod Lipson. *Symbolic Regression of Implicit Equations*, pp. 73–85. Springer US, 2010.

Phiala E. Shanahan, Amalie Trewartha, and William Detmold. Machine learning action parameters in lattice quantum chromodynamics. *Phys. Rev. D*, 97:094506, 2018.

Japneet Singh, Mathias Scheurer, and Vipul Arora. Conditional generative models for sampling and phase transition indication in spin systems. *SciPost Physics*, 11:043–071, 2021.

Ankur Singha, Dipankar Chakrabarti, and Vipul Arora. Conditional normalizing flow for markov chain monte carlo sampling in the critical region of lattice field theory. *Phys. Rev. D*, 107:014512, 2023.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced score matching: A scalable approach to density and score estimation. In *Uncertainty in Artificial Intelligence*, pp. 574–584. PMLR, 2020.

Tony Tohme, Mohsen Sadr, Kamal Youcef-Toumi, and Nicolas G Hadjiconstantinou. Messy estimation: Maximum-entropy based stochastic and symbolic density estimation. *Trans. Mach. Learn. Res.*, 2024.

Benigno Uria, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle. Neural autoregressive distribution estimation. *Journal of Machine Learning Research*, 17:1–37, 2016.

Aäron Van Den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International conference on machine learning*, pp. 1747–1756. PMLR, 2016.

Martin Vastl, Jonáš Kulhánek, Jiří Kubalík, Erik Derner, and Robert Babuška. Symformer: End-to-end symbolic regression using transformer-based architecture. *IEEE Access*, 2024.

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23:1661–1674, 2011.

Lingxiao Wang, Gert Aarts, and Kai Zhou. Generative diffusion models for lattice field theory. *arXiv preprint arXiv:2311.03578*, 2023.

Hao Wu, Jonas Köhler, and Frank Noé. Stochastic normalizing flows. *Advances in Neural Information Processing Systems*, 33:5933–5944, 2020.

A Data Generation

In this section, we briefly describe the methods used to generate samples from the distributions considered in this study.

A.1 Multivariate Gaussian Distribution

We use the 'torch.distributions' library to generate samples for training and testing the various models. With a Gaussian distribution having a mean of $[-1.5, 1.5]$ and unit covariance, we generate 25,000 samples for training, 5,000 for validation, and 5,000 for testing the NF, Score, and SR models. Additionally, another set of 20,000 samples is used for inference with the SR model to compute metrics. Similarly, for a Gaussian distribution with mean $[-1.5, 1.5]$ and a diagonal covariance matrix $\begin{bmatrix} 1.0 & 0.5 \\ 0.5 & 1.0 \end{bmatrix}$, the same number of samples are generated as detailed above for the earlier setting.

A.2 Many-Well Distribution

We generate samples from the Many-Well (MW) distribution using rejection sampling as detailed in Midgley et al. (2023). We explore six variants of the Many-Well distribution with varying dimensions (2, 4, 8, 16, 32, and 64). For each variant, we generate 100,000 samples for training, 10,000 for validation, and 10,000 for testing the models. Furthermore, an independent set of 20,000 samples is used for inference with the SR model to compute metrics.

A.3 XY Model distribution

We use the Metropolis-Hastings (MH) algorithm (Chib & Greenberg, 1995) to generate square lattices of size 8×8 (64 dim) for the XY model. We set $\lambda = 1.4^{-1}$ and the coupling constant(J) is set to unity. For proposal distribution, we use a uniform distribution $q(z)$ in the MH algorithm. On account of burn-in, around 30000 initial samples are discarded from the MCMC chain. After every 320 MCMC steps, a configuration is added to the set to have a minimum correlation across samples in the training data. We generate 100,000 samples for training, 10,000 for validation, and 10,000 for testing various models.

A.4 Scalar ϕ^4 theory distribution

We use the HMC algorithm (Neal, 2011) to generate samples from the scalar ϕ^4 theory distribution. Lattices of size 8×8 (64 dimensions) are generated. The step size is set to 0.05, with 20 steps in the leapfrog integrator of the HMC algorithm. The first 1,000 samples are discarded on account of thermalization. We generate ϕ^4 data for two different mass settings $\lambda_2 = \{-4, 1\}$, while keeping the coupling constant λ_1 fixed at 4. For each setting, we use 100,000 samples for training, 10,000 for validation, and 10,000 samples for testing the various models.

For renormalization experiments in scalar ϕ^4 field theory, we generate 64×64 (4096-dim) lattices using the HMC algorithm. The simulations are performed in the non-perturbative regime with a coupling constant $\lambda_1 = 4.0$ and mass parameter $\lambda_2 = 1$. The HMC integrator uses a step size of 0.05 and 25 leapfrog steps per trajectory. To ensure thermalization, the first 5000 samples are discarded. From the remaining data, we use 100,000 samples for training and 25,000 samples each for testing and validation.

B Model and Training details

In this section, we provide an overview of the model architecture and hyperparameters used in our experiments. We primarily focus on three models. First, we describe the architecture employed in the flow modelling , which is used to compute the negative log-likelihood of the samples. Next, we briefly outline the score estimation models used in the score-based approach. Finally, we discuss the two SR modelling approaches—EQL (Kamienny et al., 2022) and PySR (Cranmer, 2023)—along with their respective training hyperparameters. We run all experiments on four NVIDIA GeForce RTX 3080 GPUs, each with 10 GB of VRAM. Training time for the PySR model ranged from 10 minutes to 90 minutes, whereas the EQL model required significantly more time, varying between 2 to 10 hours depending on the dataset. While the training time for the flow-based and score-based models ranged from 0.5 to 8 hours, depending on the dataset.

B.1 Normalising Flows

We implement the Flow model using the RNVP architecture as detailed in Dinh et al. (2017). For the multivariate Gaussian distribution, we utilize 2 Affine coupling layers for the first variant with unit covariance matrix and 4 coupling layers for the second variant with diagonal covariance matrix, as described in Appendix A.1. Each coupling layer is composed of two dense layers, each containing 256 neurons with ReLU activation. The output consists of two layers: one for scaling and one for translation. The scaling layer outputs two neurons with Tanh activation, while the translation layer outputs two neurons with linear activation. The model is trained with a batch size of 512, using the Adam optimizer with an initial learning rate of 1×10^{-5}.

For the many-well distribution, we use 10 affine coupling layers to model the distribution across all variants ($n = \{2, 4, 8, 16, 32, 64\}$). The coupling architecture and training details remain the same as described above.

For the XY model dataset, we use 10 affine coupling layers to model the distribution. Each coupling layer consists of 2 convolutional layers with 256 filters of kernel size 3×3 and ReLU activation, using circular padding. The output layer is made up of 2 components: a scaling layer with Tanh activation and a translation layer with linear activation. The model is trained with a batch size of 128 using the Adam optimizer with an initial learning rate of 1×10^{-5}.

Since we use a Gaussian distribution as the base distribution for the NF model, which has support over $\mathbf{R}^1$, the model learns better when the dataset also lies in Euclidean space. However, the XY dataset exists on a circular manifold $\Theta \in [0, 2\pi)^{N \times N}$. To address this, we transform the dataset into Euclidean space using a sigmoid transformation, as detailed in Kanaujia et al. (2024).

For the scalar ϕ^4 theory distribution, we use 12 affine coupling layers to model the distribution for both variants. The coupling architecture and training details are the same as those used for the XY dataset, as discussed above.

For the renormalization experiments in scalar ϕ^4 field theory, we generate lattices of size 64×64. These are progressively coarsened via average pooling with a 2×2 filter, reducing the spatial resolution at each step. After the first pooling operation, the lattices are reduced to 32×32 in size. At this resolution, we model the field distribution using RNVP flow architecture comprising 16 affine coupling layers, each employing 512 convolutional filters. Upon further coarsening, the lattices are downsampled to 16×16 and subsequently to 8×8. At the 16×16 level, we use 12 affine coupling layers, and at 8×8, we use 10 affine coupling layers. For both of these resolutions, each coupling layer uses 256 convolutional filters. The coupling layer architecture remains consistent across scales, with changes only in the number of filters to accommodate the reduced spatial complexity.

B.2 Score Model

We estimate the score from the data using a neural network that models the score function $s_\theta(x)$ as followed in Song & Ermon (2019). The model is trained via score matching by minimizing the loss function as described in Eq. (4).

For the multivariate Gaussian distribution, we use a multilayer perceptron (MLP) with the architecture [2, 256, 256, 2] to model the score function. For the many-well distribution, we model the score via an MLP with the architecture [2, 256, 256, 256, 2] for $n = 2$. For $n = \{4, 8, 16, 32, 64\}$, we use an MLP architecture with 4 hidden layers, each consisting of 512 neurons.

For the XY model and scalar ϕ^4 distributions, we optimize the modeled score function using denoising score matching loss, as described in Eq. (6). The score is modeled with the architecture provided by Song & Ermon (2019). The model, initially downloaded from their GitHub repository, has been adapted to fit the dataset configuration. Training is carried out using the Adam optimizer, beginning with an initial learning rate of 1×10^{-4}, which is subsequently reduced to 1×10^{-5}.

B.3 EQL

This approach builds on the Equation-based Learning (EQL) architecture (Martius & Lampert, 2017; Sahoo et al., 2018; Kim et al., 2021), integrating it with other neural modules to allow end-to-end training via backpropagation. To promote sparse and interpretable solutions, we employ a smoothed variant of the $L_{0.5}$ regularization, which prevents gradient singularities and improves convergence. The regularization function is defined as:

$$L^*_{0.5}(w) = \begin{cases} |w|^{1/2}, & \text{if } |w| \geq a \\ \left(-\frac{w^4}{8a^3} + \frac{3w^2}{4a} + \frac{3a}{8}\right)^{1/2}, & \text{if } |w| < a \end{cases} \tag{15}$$

with $a = 0.01$ in all experiments. This penalty is selectively applied to the EQL-specific weights when the model is embedded within a broader neural architecture.

Training is conducted in two phases. The first phase uses a higher learning rate of 10^{-2}, combined with a regularization weight $\lambda = 5 \times 10^{-3}$ for the multivariate gaussian and many-well distributions and 3×10^{-2} for the ϕ^4 and XY model distributions. This phase spans 20000 iterations, with small regularization weight promoting sparsity in the EQL layers. At the end of this phase, EQL weights with magnitude less than a threshold ($\alpha = 0.01$) are pruned and frozen. In the second phase, training proceeds for 10000 iterations with a reduced learning rate of 10^{-3} and no regularization ($\lambda = 0$), enabling fine-tuning of the pruned network.

To balance model expressivity with interpretability, different sets of activation functions are used in the hidden and final layers. For the many-well distribution, we employ a functionally rich set of activations across all layers, including: Constant (1 neuron), Identity (1 neuron), Square g^2 (1 neuron), Power 4 g^4 (1 neuron), and Multiplication $g_1 \cdot g_2$ (1 neuron). For the multivariate Gaussian distribution, a similar configuration to the many-well distribution is used, with the exception of the Power 4 function, which is omitted. In the case of the ϕ^4 distribution, the Power 4 and Product functions are applied only in the hidden (forward) layers, while the final layer is limited to Constant, Identity, and Square functions to constrain the complexity of the output. In the score-based modeling of ϕ^4, the Power 4 function is completely excluded from the primitive set to further reduce the search space and enhance tractability.

For the XY model distribution, which exhibits a circular topology, we introduce an inductive bias by incorporating trigonometric primitives—sin and cos, each with 1 neuron—into both the hidden and final layers. To prevent the nesting of trigonometric functions and reduce symbolic complexity, we adopt a feature transformation strategy in which the input kernel is explicitly expanded to include trigonometric variants. For instance, for a kernel of size 2 with variables $[x_1, x_2, x_3, x_4]$, the input is transformed to $[\cos(x_1), \ldots, \cos(x_4), \sin(x_1), \ldots, \sin(x_4)]$. This approach is similarly extended to other kernel sizes. This

transformation not only helps suppress excessive equation complexity but also proves especially beneficial in score-based modeling, as it simplifies integration tasks that would otherwise be intractable.

B.4 PySR

For the PySR experiments, we use the PySR (Cranmer, 2023) library, which implements a multi-population, tree-based genetic algorithm where multiple evolutions are performed asynchronously. We use PySR with a modified MSE loss, as described in Section 2, under factorization. The experimental parameters are set as follows:

Sample Size: We train the model using varying sample sizes based on the convergence of the training loss. For learning the equations with the flow-based and score-based approaches, either 512 or 1024 samples are used, depending on performance. Experiments are conducted with sample sizes of 128, 256, 512, 1024, and 2048. Beyond this range, we observe no further improvement in performance; hence, the aforementioned sample sizes are selected for the experiments.

Maximum Complexity: This parameter controls the permissible size of symbolic equations, which corresponds to the number of nodes in the search tree. It is set to 20 for the multivariate Gaussian and many-well distributions, 30 for the ϕ^4 distribution, and 40 for the XY model dataset. For renormalisation experiments in ϕ^4 theory, it is set to 30. These values are chosen based on the convergence of the training loss. Increasing the complexity further results in an exponential increase in search time.

Population Size: The population size corresponds to the number of candidate solutions evolved in each generation. It is set to 20 for the multivariate Gaussian and many-well distributions and 30 for the ϕ^4 and the XY model distributions, guided by training loss convergence.

C Impact of Hyperparameter Choices on Method Performance

In this section, we discuss the sensitivity of the proposed framework to key hyperparameters governing both the generative models and the symbolic regression components. The hyperparameters of the generative model are selected by monitoring performance on a validation set. Similarly, the hyperparameters of the symbolic-regression (SR) methods are adjusted based on the validation error and the model complexity, both of which should remain small. In practice, PySR exhibits higher sensitivity to the number of training samples and to the complexity parameter, whereas EQL is more sensitive to the learning rate.

i. Number of samples for SR learning: EQL requires substantially more training data, typically on the order of 10k–100k samples, than PySR. In PySR, we typically use 500–2000 samples. This is in line with the general observation that genetic algorithms require less training data as compared to neural networks. This number, however, increases with the data dimensionality and with the number of modes in the underlying distribution. For example, achieving low complexity and low MSE for MW-64 requires more samples than for Φ^4 (d = 64), owing to the larger number of modes in MW-64.

ii. Number of samples for NF or score-model learning: The performance of both NF and score-based models improves as the training sample size increases. To analyse this effect on the overall framework, we conduct a study on the flow-based pipeline (NF with PySR), trained on the ϕ^4 distribution ($d = 64$) using varying numbers of training samples. The corresponding results are summarised in Table 7 below.

Table 7: Effect of ensemble size on model performance.

No. of Samples	MSE(↓)	R^2(↑)	Estimated Equation
10K	0.47	0.994	$3.5x_0^4 - 1.85x_0x_1 - 1.85x_0x_2$
50K	0.07	0.999	$4.2x_0^4 - 2.04x_0x_1 - 2.04x_0x_2$
100K	0.01	0.999	$3.9x_0^4 - 1.95x_0x_1 - 2.00x_0x_2$

iii. Complexity control in PySR and EQL: For PySR, the complexity is typically set between 10 and 30. In EQL, the model complexity is controlled by the architecture of the network, specifically by adjusting

the number of layers and neurons in the primitive set of operators. Deeper networks lead to increased model complexity.

In both PySR and EQL, the method struggles to generate an interpretable equation when model complexity is too high or when the dataset fails to sufficiently capture all modes. In multimodal settings, generative models such as NFs often exhibit mode-covering behaviour. This can lead to inaccuracies in likelihood estimation, which, when propagated to the SR stage, may result in the identification of spurious or extraneous terms.

In PySR, both the dataset size and sample quality play a critical role. By sample quality, we refer to the extent to which the data adequately represent all relevant modes of the distribution. The complexity hyperparameter is equally important: as model complexity increases, the inferred symbolic expressions tend to include additional spurious terms. While increasing the dataset size can initially suppress these extraneous terms, beyond a certain point the complexity constraint becomes the dominant factor determining the number of such terms.

Similarly, in EQL, increasing the dataset size helps suppress additional terms in the inferred equation only up to a certain point. Beyond this threshold, the intrinsic complexity of the model, as well as errors propagated from the generative model (e.g., NF or score-based model), begin to significantly influence the accuracy of the SR. These factors can ultimately lead to the appearance of extra terms in the final symbolic expression.

D Robustness Analysis

To ensure that the reported results are not artifacts of individual training runs, we investigate the repeatability of each method by performing multiple trials with distinct random seeds. The corresponding performance metrics are aggregated and presented as seed-wise distributions. This analysis highlights the degree of variability across runs and provides a more comprehensive understanding of the reliability of the reported results.

D.1 Phi-4 Benchmark

For the scalar ϕ^4 theory distribution with $d = 64$ (see Table 4 in the main paper), we show histogram results aggregated over multiple runs with different random seeds.

NF + PySR (20 seeds)

MSE	R^2	Count
0.01–0.10	> 0.996	7
0.10–0.15	0.994–0.996	8
> 0.15	< 0.994	5

NF + EQL (10 seeds)

MSE	R^2	Count
0.001–0.08	> 0.997	5
0.08–0.15	0.994–0.997	1
> 0.15	< 0.994	4

D.2 Many-Well Benchmark

For the Many-well distribution($n = 64$) (see Table 2 in the main paper), we show histogram results aggregated over multiple runs with different random seeds.

SM + PySR (18 seeds)

MSE	R^2	Count
0.2–0.5	> 0.99	5
0.5–1.0	0.98–0.99	3
1.0–4.0	0.90–0.98	5
> 4.0	< 0.90	5

SM + EQL (8 seeds).

MSE	R^2	Count
0.35–0.50	> 0.97	3
0.50–4.0	0.90–0.97	4
> 4.0	< 0.90	1

Overall, these seed-dependent histograms demonstrate that the proposed methods achieve reliable performance across runs, with a consistently strong success rate.

E Metrics

We outline the steps for calculating the overall MSE error and R^2 score for the proposed approaches. For each sample x_i, the corresponding output is $H(x_i)$, which serves as the target output. Using a symbolic regressor, we obtain the predicted output $\hat{H}(x_i)$.

- Step1: Compute target mean $\bar{H} = \frac{1}{N}\sum_{i=1}^{N} H(x_i)$
- Step2: Mean normalize target values :

$$\tilde{H}(x_i) = H(x_i) - \bar{H} \tag{16}$$

- Step3: Compute predicted mean $\bar{\hat{H}} = \frac{1}{N}\sum_{i=1}^{N} \hat{H}(x_i)$
- Step4: Mean normalise predicted values:

$$\tilde{\hat{H}}(x_i) = \hat{H}(x_i) - \bar{\hat{H}} \tag{17}$$

- Step5: Compute MSE using mean normalized target values and mean normalized predicted values.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\tilde{H}(x_i) - \tilde{\hat{H}}(x_i))^2 \tag{18}$$

Apply the same procedure to calculate the R^2 score using the mean normalized target values and the mean normalized predicted values. When the score is provided as input to the model, the symbolic regressor (SR) predicts the expression for $-\nabla_x(H(x))$. Upon integrating this expression, a constant term is introduced. To evaluate the model, remove this constant and use the predicted $\hat{H}(x)$ without the constant. Then, proceed with the same evaluation steps as before.