Bayesian Symbolic Regression with Entropic Reinforcement Learning

Oussama Boussif[1,2] **Mohammed Mahfoud**[3] **Younesse Kaddar**[4] **Moksh Jain**[1,2] **Sida Li**[5] **Damiano Fornasiere**[6]
Xiaoyin Chen[1,2] **Yoshua Bengio**[1,2,7] **Esmeralda S. Whitammer**[7,8]

[1]Mila – Québec AI Institute [2]Université de Montréal [3]Independent [4]University of Oxford [5]University of Chicago [6]LawZero [7]CIFAR Fellow [8]University of Edinburgh

Abstract

Symbolic regression is the problem of finding an algebraic expression describing a stochastic dependence of a target variable on a set of inputs. Unlike forms of regression that fit parameters assuming a fixed model structure, symbolic regression is a search problem over the space of expressions, represented, for example, as abstract syntax trees using a library of operators. Symbolic regression is typically used in settings with limited, noisy data in the natural sciences. However, searching for a single best-fitting expression fails to capture the epistemic uncertainty about the expression, which motivates a Bayesian perspective that enables uncertainty quantification and specification of natural priors to constrain the search space. In this work, we propose ERRLESS (Entropy-Regularized Reinforcement Learning for Expression Structure Sampling), a scalable approach for sampling from the posterior distribution over expressions given data using maximum-entropy reinforcement learning. ERRLESS learns a neural policy that constructs expressions sequentially by building up their abstract syntax trees. At convergence, the policy samples expressions from the posterior. At test time, expressions can be sampled by rollouts of this policy. We demonstrate that ERRLESS achieves competitive results on the Feynman benchmark while producing short and interpretable expressions. Additionally, we demonstrate that the mean of the posterior predictive approximated by ERRLESS achieves a higher coefficient of determination (R^2) compared to an SMC baseline, which shows the value of the Bayesian perspective in symbolic regression.

1 INTRODUCTION

Symbolic regression (SR) is the problem of searching over a space of compositional algebraic expressions, using a certain library of primitive operators, to find a function that most closely maps the inputs observed in a dataset to their corresponding outputs. SR is a common problem in the natural sciences, where datasets are small and noisy, domain priors constrain plausible formulas, and interpretability is important [Bongard and Lipson, 2007, Schmidt and Lipson, 2009, Udrescu and Tegmark, 2020]. Most existing algorithms for SR [Petersen et al., 2021, Biggio et al., 2021, Mundhenk et al., 2021, Tenachi et al., 2023, Kamienny et al., 2023] have the goal of finding a Pareto front of expressions that trade off complexity with fit. With limited data, such a point estimate can be unreliable and hides the uncertainty about the expression arising from the noise and scarcity of data.

The need to model uncertainty in SR has been recognized in the literature and addressed by a Bayesian perspective on the problem [Jin et al., 2019, Guimerà and Sales-Pardo, 2026]. In this view, one posits a (structured) prior over expression structures and parameters and a model of observation noise. A dataset of (input, output) pairs then induces a posterior distribution over expressions, and the aim of Bayesian SR is to sample from this posterior. Previous Bayesian SR methods use reversible-jump Markov chain Monte Carlo (MCMC) [Green, 1995] or sequential Monte Carlo (SMC) [Bomarito and Leser, 2026, Guimerà and Sales-Pardo, 2026], but these methods rely on handcrafted proposal distributions and can be costly to scale. In this paper, we instead seek an approach that amortizes the sampling process using a neural network.

We formulate the construction of an expression as a sequential decision-making problem, turning the Bayesian SR task into the reinforcement learning (RL) problem of training a policy to sample expressions from the posterior. To achieve unbiased sampling at convergence, we train this policy using a maximum-entropy RL objective with the

Proceedings of the 42nd Conference on Uncertainty in Artificial Intelligence (UAI 2026), PMLR 337:745–764.

unnormalized posterior log-density as the reward (§3.2). The resulting system, which we call ERRLESS (Entropy-Regularized Reinforcement Learning for Expression Structure Sampling), is capable of learning a policy that samples expression structures and their real-valued parameters; once trained, the policy can be sampled to produce approximate posterior samples efficiently.

Successfully applying entropy-regularized RL to Bayesian SR requires careful design choices. ERRLESS uses a generation process that constructs expression syntax trees in a bottom-up (postorder) manner to allow effective imposition of constraints, imposes structural priors to avoid ill-formed, redundant, or unlikely subexpressions, and can incorporate additional constraints to restrict the search space to forbid dimensionally incompatible compositions of physical units (§2.1). The parametrization and training of the policy similarly require appropriate design of neural architectures and off-policy training schemes (§3.3). Finally, unlike previous Monte Carlo-based methods, ERRLESS amortizes sampling of both expression structures and parameter values into a single neural policy and avoids explicit per-candidate constant fitting.

On the Feynman Symbolic Regression Database [Udrescu and Tegmark, 2020], ERRLESS achieves a competitive prediction accuracy and produces posterior samples that improve predictions when data are few and noisy. On synthetic data, the posterior predictive mean of ERRLESS is better than PySIPS [Bomarito and Leser, 2026], an SMC baseline on the noisiest setting, showing the value of modeling uncertainty over expressions with a Bayesian view.

We summarize our contributions as follows:

- We design a novel generation process (environment) for symbolic regression that enables bottom-up expression construction, incorporates dimensional analysis, and imposes structural constraints.
- We formulate Bayesian symbolic regression, including inference of scalar parameters in expressions, as an end-to-end policy-learning problem within this environment.
- We demonstrate that ERRLESS achieves competitive prediction accuracy compared to state-of-the-art approaches on the Feynman Symbolic Regression Database while producing short expressions.
- We show that our approach captures the posterior distribution effectively, facilitating downstream applications in settings with scarce and noisy data.

2 PROBLEM SETTING

In this section, we describe the formal setting for Bayesian SR. §2.1 describes the search space and §2.2 formulates the sampling problem.

Notation. Scalar (resp. vector) random variables, or variables whose type has not been specified, are denoted x (resp. $\mathbf{x}$), and their realization is denoted x (resp. $\boldsymbol{x}$). The probability distribution of a discrete random variable x is denoted P_{x} (in uppercase) and that of a continuous random variable, or a variable whose type has not been specified, is denoted p_{x} (in lowercase). Let $\mathbb{N}$ (resp. $\mathbb{R}$) be the set of natural (resp. real) numbers, and $\mathbb{N}_{>0} \coloneqq \mathbb{N}\setminus\{0\}$ (resp. $\mathbb{R}_{>0} \coloneqq \{x \in \mathbb{R} \mid x > 0\}$). For $n \in \mathbb{N}$, we denote the set of integers $\{1, \ldots, n\}$ (or the empty set when $n = 0$) as $[n]$.

For a set S, we denote by $|S|$ its cardinality and by $S^* \coloneqq \bigcup_{n\in\mathbb{N}} S^n$ the set of all finite sequences (strings) over S.

2.1 THE SPACE OF EXPRESSIONS

Expression trees. We fix a vocabulary $\Sigma \coloneqq \mathcal{V} \sqcup C \sqcup O$, where $\mathcal{V} \coloneqq \{v_i\}_{i\in[D]}$ is a set of *variable symbols*, $C \coloneqq \{c_k\}_{k\in[K]}$ is a set of *constant symbols*, and $O \coloneqq \{g_j\}_{j\in[M]}$ is a set of *operator symbols* (O is referred to as the *operator library*). Each operator $g \in O$ is equipped with an *arity* $r_g \in \mathbb{N}_{>0}$ and a *semantic function* $\phi_g \colon \mathcal{X}_g \to \mathbb{R}$, where $\mathcal{X}_g \subseteq \mathbb{R}^{r_g}$. For example, the operator '+' is binary and $\phi_+ \colon \mathbb{R}^2 \to \mathbb{R}, (x, y) \mapsto x + y$; the operator 'log' is unary and $\phi_{\log} \colon \mathbb{R}_{>0} \to \mathbb{R}, x \mapsto \log x$.

An *expression tree* T is a finite, rooted, ordered tree where each leaf is labeled with a symbol from $\mathcal{V} \sqcup C$, and each internal node with r children is labeled with a $g \in O$ such that $r_g = r$. Let $I_C(\mathrm{T}) \subseteq [K]$ (resp. $I_{\mathcal{V}}(\mathrm{T}) \subseteq [D]$) be the set of indices of constants (resp. variables) appearing in T. The constants $\{c_k\}_{k\in I_C(\mathrm{T})}$ are thought of as symbolic placeholders: a *constant assignment* is a vector $\boldsymbol{\theta} \in \mathbb{R}^K$ specifying numerical values for the symbols at indices $I_C(\mathrm{T})$.

Let $\mathcal{T}_\Sigma$ be the space of expression trees over Σ.

Evaluation. Let $\mathrm{T} \in \mathcal{T}_\Sigma$. Given a constant assignment $\boldsymbol{\theta} \coloneqq (\theta_k)_{k\in[K]} \in \mathbb{R}^K$, one can define a partial *evaluation function* $f_{\mathrm{T},\boldsymbol{\theta}} \colon \mathbb{R}^D \to \mathbb{R}$ for T on every input $\boldsymbol{x} \coloneqq (x_i)_{i\in[D]} \in \mathbb{R}^D$ recursively as follows:

- each leaf of T labeled $v_i \in \mathcal{V}$ (resp. $c_k \in C$), where $i \in I_{\mathcal{V}}(\mathrm{T})$ (resp. $k \in I_C(\mathrm{T})$), evaluates to x_i (resp. θ_k);
- each internal node of T labeled with operator g having children that evaluate to $a_1, \ldots, a_{r_g}$ evaluates to $\phi_g(a_1, \ldots, a_{r_g})$ if $(a_1, \ldots, a_{r_g}) \in \mathcal{X}_g$, otherwise the evaluation is undefined.

Sequence representation. An expression tree T can be canonically represented as a sequence $\mathcal{W}(\mathrm{T}) \in \Sigma^*$ by listing the labels of its nodes in postorder traversal, a representation known as reverse Polish notation [Łukasiewicz, 1929]. For example, the expression $\sin(v_1 + c_1 \times v_2)$ is represented by the sequence $(v_1, c_1, v_2, \times, +, \sin)$. (It is well-known that such an expression can be evaluated, when real numbers are substituted for the variables and constants, by traversing the sequence from left to right and maintaining a stack.)

Unit constraints. One can additionally constrain the space of trees by modifying the above definitions to include di-

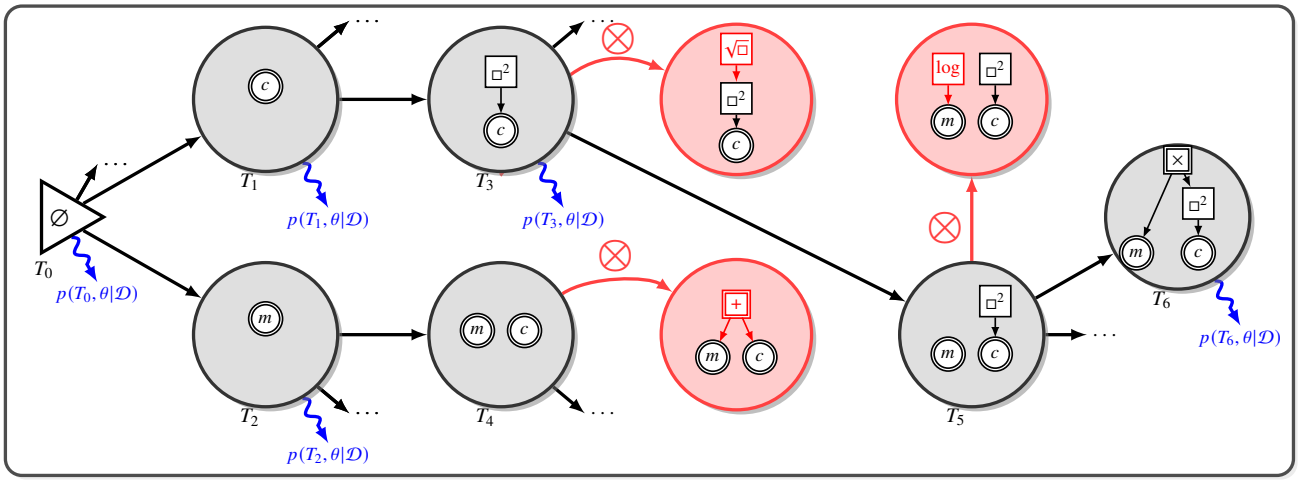


Figure 1: **Generation process for the expression $E = mc^2$.** Valid states are shown in gray, while invalid states have pink fill and red outlines. Black arrows indicate valid transitions, and red arrows indicate invalid transitions. Leaf nodes are represented by double-outlined circles, unary operators by single-outlined squares, and binary operators by double-outlined squares. Wiggly arrows denote states that can terminate and transition to the terminal state.

mension information. We assume that each variable v_i has an associated physical unit (*e.g.*, meters, kilograms), represented as a vector $\boldsymbol{u}_i \in \mathbb{R}^\ell$ (called *unit vector*), where $\ell := 7$ is the number of base units. Each coordinate of $\boldsymbol{u}_i$ represents the exponent of its corresponding unit, *e.g.*, velocity has units $\mathrm{m\,s^{-1}}$, represented as $(1, 0, -1, 0, 0, 0, 0)$. Dimensionless variables are represented by the zero vector $(0, \ldots, 0) \in \mathbb{R}^\ell$. All constants c_k are dimensionless.

Every operator g is assumed to have constraints on the unit vectors of its arguments, as well as a mapping from the unit vectors of the arguments to the unit vector of its output. Formally, every g has a *unit assignment function*, a partial function $\mathcal{U}_g : (\mathbb{R}^\ell)^{r_g} \to \mathbb{R}^\ell$ that returns the unit vector of the operator's output if the operator can be applied to inputs with the given units (and is undefined otherwise). Let $\mathcal{T}^\circ \subseteq \mathcal{T}_\Sigma$ be the set of those expression trees which are dimensionally valid: when evaluating bottom-up, $\mathcal{U}_g$ is never undefined when evaluated on the unit vectors of the children of any internal node. See §A.1 for the assignment functions of operators in the library we use.

Example. Consider the following operators applied to $\boldsymbol{u}^{\text{vel}} := (1, 0, -1, 0, 0, 0, 0)$, $\boldsymbol{u}^{\text{time}} := (0, 0, 1, 0, 0, 0, 0)$, and $\boldsymbol{u}^{\text{angle}} := (0, 0, 0, 0, 0, 0, 0)$:

- $+$: $\mathcal{U}_+(\boldsymbol{u}^{\text{vel}}, \boldsymbol{u}^{\text{time}})$ is undefined (incompatible units: time and velocity cannot be added), but $\mathcal{U}_+(\boldsymbol{u}^{\text{vel}}, \boldsymbol{u}^{\text{vel}}) = \boldsymbol{u}^{\text{vel}}$ (adding two velocities gives a velocity);
- $\times$: $\mathcal{U}_\times(\boldsymbol{u}^{\text{vel}}, \boldsymbol{u}^{\text{time}}) = (1, 0, 0, 0, 0, 0, 0)$ (multiplying velocity and time gives a distance);
- sin: $\mathcal{U}_{\sin}(\boldsymbol{u}^{\text{angle}}) = \boldsymbol{u}^{\text{angle}}$ (sine of a scalar is a scalar), but $\mathcal{U}_{\sin}(\boldsymbol{u}^{\text{vel}})$ is undefined.

2.2 BAYESIAN SYMBOLIC REGRESSION

We fix a prior probability distribution $P(\mathrm{T})$ over the set of valid expression trees $\mathcal{T}^\circ$ (§3.3 describes the choice of prior) and a conditional prior distribution $p(\boldsymbol{\theta}, \sigma \mid \mathrm{T})$ over the space $\mathbb{R}^K$ of constant assignments and the noise variance.

Let $\{(\mathbf{x}_i, \mathrm{y}_i)\}_{i=1}^N$ be a collection of random variables such that, for every $i \in [N]$, $(\mathbf{x}_i, \mathrm{y}_i)$ takes values in $\mathbb{R}^D \times \mathbb{R}$, and each realization $\boldsymbol{x}_i \in \mathbb{R}^D$ of $\mathbf{x}_i$ defines the following conditional distribution:

$$\mathrm{y}_i \mid (\mathbf{x}_i = \boldsymbol{x}_i, \mathrm{T}, \theta, \sigma) \sim \mathcal{N}\big(f_{\mathrm{T},\boldsymbol{\theta}}(\boldsymbol{x}_i), \sigma^2\big), \tag{1}$$

where $\sigma^2 \in \mathbb{R}_{>0}$ is a noise variance and the y_i's are conditionally independent given the $\mathbf{x}_i$'s. That is, y_i is the evaluation of the expression given by $\mathrm{T}, \boldsymbol{\theta}$ with inputs $\boldsymbol{x}_i$, with added Gaussian noise.

We observe a dataset $\mathcal{D} := \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$. Assuming $\boldsymbol{x}_i$ are independent of T and $\boldsymbol{\theta}$ and σ, the posterior of $(\mathrm{T}, \boldsymbol{\theta}, \sigma)$ given $\mathcal{D}$ satisfies the following relation:

$$\begin{aligned}
&p(\mathrm{T}, \boldsymbol{\theta}, \sigma \mid \mathcal{D}) \\
&\propto p(\mathrm{T}, \boldsymbol{\theta}, \sigma) \cdot p(\mathcal{D} \mid \mathrm{T}, \boldsymbol{\theta}, \sigma) \\
&\propto p(\mathrm{T}, \boldsymbol{\theta}, \sigma) \prod_{i=1}^N p(y_i \mid \boldsymbol{x}_i, \mathrm{T}, \boldsymbol{\theta}, \sigma) \underbrace{p(\boldsymbol{x}_i \mid \mathrm{T}, \boldsymbol{\theta}, \sigma)}_{= p(\boldsymbol{x}_i)} \\
&\propto P(\mathrm{T}) p(\boldsymbol{\theta}, \sigma \mid \mathrm{T}) \prod_{i=1}^N \mathcal{N}(y_i; f_{\mathrm{T},\boldsymbol{\theta}}(\boldsymbol{x}_i), \sigma^2)
\end{aligned} \tag{2}$$

The objective of Bayesian symbolic regression is to sample from this posterior distribution. The next sections introduce the framework we use to approximate it in practice.

3 METHODOLOGY

We introduce ERRLESS (Entropy-Regularized Reinforcement Learning for Expression Structure Sampling), a scalable approach for Bayesian symbolic regression, illustrated in Fig. 1. §3.1 introduces a sequential decision-making process for sampling expression trees that enforces constraints during generation, §3.2 describes learning objectives for decision-making policies, and §3.3 describes the policy architecture used to sample expressions.

For concreteness, we specify the set of operators we will use in our experiments: the unary operators $\{\sin, \cos, \log, \exp, \square^2, \sqrt{\square}, -\square\}$ and the binary operators $\{+, -, /, \times\}$ (see §A.1 for details). However, the algorithm described below is not restricted to this particular choice.

3.1 BOTTOM-UP GENERATION

Most approaches in the literature [Petersen et al., 2021, Li et al., 2023] employ *top-down generation*, where internal nodes (*i.e.*, operators) are sampled before leaf nodes (input variables and constants). With top-down generation, the intermediate expression at each step contains 'holes' to be filled. As a result, the expression can only be evaluated, and the posterior density computed, at the end of the generation. Moreover, with top-down generation, enforcing physical unit constraints is inefficient: since the operands may not yet be specified at intermediate steps, assigning units requires traversing the full depth of the tree once the units have been determined.

In contrast, *bottom-up generation* offers two main advantages. First, some intermediate states are valid and complete expressions. Second, because leaf nodes (and therefore operands) are specified from the outset, newly added operators can be constrained to be compatible with the known physical units of the operands.

Sequential generation of expression trees. Recall that an expression tree T is uniquely represented by its postorder representation $\mathcal{W}(\mathrm{T}) \in \Sigma^*$. Therefore, generating an expression tree is equivalent to generating its sequence representation by appending one symbol at a time from left to right. Because all operator nodes appear after their arguments in postorder traversal, the dimensional validity and arity constraints correspond to restrictions on continuations of partial sequences. The problem of modeling a probability distribution over expression trees is thus reduced to one of autoregressive sequence modeling.

We define the token alphabet $\mathcal{A} \coloneqq \Sigma \sqcup \{\top\}$, where $\top$ is a special symbol marking sequence termination. A distribution π over the set of trees $\mathcal{T}^\circ$ is equivalent to a distribution over $\mathcal{A}^*$:

$$\pi(w_1 \dots w_n \top) = \left(\prod_{i=1}^{n} \pi(w_i \mid w_{<i})\right) \pi(\top \mid w_{1\dots n}), \quad (3)$$

where $w_i \in \mathcal{A}$ and the support of each next-symbol distribution respects the unit and arity constraints, as well as the restriction that generation of $\top$ is permitted only after a sequence that represents a complete expression tree. (We must also make the assumption that every partial sequence can be continued to at least one sequence ending in $\top$, which holds with the library of operators we consider.)

Length and redundancy constraints. We remark that constraints on the number of nodes in the expression tree T can also be expressed as restrictions on the support of the next-token distributions of $\mathcal{W}(\mathrm{T})$ in (3). In addition, structural constraints in the target distribution that disallow ill-formed, redundant, or unlikely compositions can also be expressed as restrictions on next-token distributions. The constraints we impose prohibit: (i) composing functions with their inverses (*e.g.*, $\sqrt{\square}^2$), (ii) nesting trigonometric functions (*e.g.*, $\sin(\cos(\square))$), (iii) nesting exponentials (*e.g.*, $e^{e^\square}$), and (iv) applying unary operators directly to constants. Fig. 2 shows that our construction rules and constraints make the search space much smaller and thus more efficient to explore.

Samplers of expressions and parameters as policies. A distribution π over sequences of the form (3) respecting the imposed constraints, together with a conditional distribution $\pi(\boldsymbol{\theta}, \sigma \mid \mathrm{T})$ over constant assignments and noise given a tree, define a joint distribution over $(\mathrm{T}, \boldsymbol{\theta}, \sigma)$. In the next section, we will describe how this autoregressively factorized distribution can be trained to sample the Bayesian posterior defined in §2.2 using reinforcement learning methods.

3.2 MAXIMUM-ENTROPY RL TRAINING OF EXPRESSION SAMPLERS

Above, we have identified a distribution π over tuples $(\mathrm{T}, \boldsymbol{\theta}, \sigma)$ with a distribution over sequence representations and a conditional distribution over constant assignments and noise:

$$\pi(\mathrm{T}, \boldsymbol{\theta}, \sigma) = \pi(\mathrm{T})\pi(\boldsymbol{\theta}, \sigma \mid \mathrm{T}) = \pi(\mathcal{W}(\mathrm{T})\top)\pi(\boldsymbol{\theta}, \sigma \mid \mathrm{T}),$$

where $\pi(\mathcal{W}(\mathrm{T})\top)$ has an autoregressive factorization (3). Suppose that π is a parametric model π_φ, which can be evaluated to obtain next-token logits for sequential generation of $\mathcal{W}(\mathrm{T})\top$ and the parameters of the distribution over $\boldsymbol{\theta}$ given T (for example, the mean and covariance of a Gaussian from which $\boldsymbol{\theta}$ is sampled). Our goal is to fit φ so that $\pi_\varphi(\mathrm{T}, \boldsymbol{\theta}, \sigma)$ equals the posterior $p(\mathrm{T}, \boldsymbol{\theta}, \sigma \mid \mathcal{D})$ defined in (2), which is given as an unnormalized probability density function.

Define

$$R(\mathrm{T}, \boldsymbol{\theta}, \sigma) = \log p(\mathrm{T}, \boldsymbol{\theta}, \sigma) + \sum_{i=1}^{N} \log \mathcal{N}(y_i; f_{\mathrm{T},\boldsymbol{\theta}}(\boldsymbol{x}_i), \sigma^2),$$

so that $p(\mathrm{T}, \boldsymbol{\theta}, \sigma \mid \mathcal{D}) \propto \exp(R(\mathrm{T}, \boldsymbol{\theta}, \sigma))$ (cf. (2)). When R is thought of as a *reward* provided to a sampler that generates T, $\boldsymbol{\theta}$ and σ following the conditional distributions of π_φ, training π_φ to maximize its expected reward $\mathbb{E}_{(\mathrm{T},\boldsymbol{\theta},\sigma)\sim\pi_\varphi}[R(\mathrm{T}, \boldsymbol{\theta}, \sigma)]$ is equivalent to minimizing cross-entropy between π_φ and the posterior, which is achieved by sampling the posterior mode. However, one can instead consider the *entropy-regularized* problem

$$\max_{\varphi} \left[\mathbb{E}_{(\mathrm{T},\boldsymbol{\theta},\sigma)\sim\pi_\varphi}[R(\mathrm{T}, \boldsymbol{\theta}, \sigma)] + \mathcal{H}[\pi_\varphi]\right], \tag{4}$$

where $\mathcal{H}[\pi_\varphi]$ is the entropy of the modeled distribution. A key property of entropy-regularized, or maximum-entropy RL [Haarnoja et al., 2018, Eysenbach and Levine, 2022] is that the solution to (4) minimizes KL divergence between π_φ and the distribution with density proportional to $\exp(R(\mathrm{T}, \boldsymbol{\theta}, \sigma))$, *i.e.*, the posterior. This property has been exploited in various instances of learning to sample by sequential decision-making [Deleu et al., 2024], and efficient off-policy training algorithms for solving (4) have been proposed.

Training objective. One such off-policy objective is the *trajectory balance* (TB) objective [Malkin et al., 2022], a core GFlowNet [Bengio et al., 2021] loss denoted by $\mathcal{L}_{\mathrm{TB}}$, which is a special case of a path consistency learning objective [Nachum et al., 2017] in deterministic environments with sparse terminal rewards. TB requires additionally learning a scalar parameter $\log Z_\varphi$ (corresponding to the initial state's value function in reinforcement learning), which, at convergence, gives the log normalizing constant (the likelihood of $\mathcal{D}$). The objective associated with $(\mathrm{T}, \boldsymbol{\theta}, \sigma)$ is:

$$\mathcal{L}_{\mathrm{TB}}(\mathrm{T}, \boldsymbol{\theta}, \sigma; \varphi) \coloneqq \left[\log Z_\varphi + \log \pi_\varphi(\mathrm{T}, \boldsymbol{\theta}, \sigma) - R(\mathrm{T}, \boldsymbol{\theta}, \sigma)\right]^2. \tag{5}$$

Because (5) can be minimized to 0 for *all* samples simultaneously, training algorithms can optimize this objective w.r.t. φ over $(\mathrm{T}, \boldsymbol{\theta}, \sigma)$ sampled from some behavior policy that does not necessarily coincide with the current state of π_φ itself. We describe our off-policy training choices in §3.3 and ablate the training objective in §D.2. (Because the reward can equal 0, we apply smoothing to prevent $\log 0$ in the loss; see §A.2.) Assuming convergence of the on-policy estimate of the objective in (4) to its optimal value (the normalizing constant), convergence in policy space is guaranteed by Theorem 3.1 of Gritsaev et al. [2025].

3.3 ERRLESS DESIGN CHOICES

Parametrization of the policy. We represent each partial expression as a sequence of tokens and parameterize the policy with a transformer-like architecture [Vaswani et al., 2017]. The transformer encodes the sequence into an embedding that is supplied to three prediction heads: (i) a head that produces the logits over the next action at each step of the construction process, for a sequence that has not terminated in $\top$, (ii) a head that outputs the means, covariances, and mixture weights of a Gaussian mixture model from which the values of the constants are sampled for a sequence that has terminated and (iii) a head that outputs the means, variances, and mixture weights of a mixture of log-normals from which the noise standard deviation σ is sampled (see §B for more details).

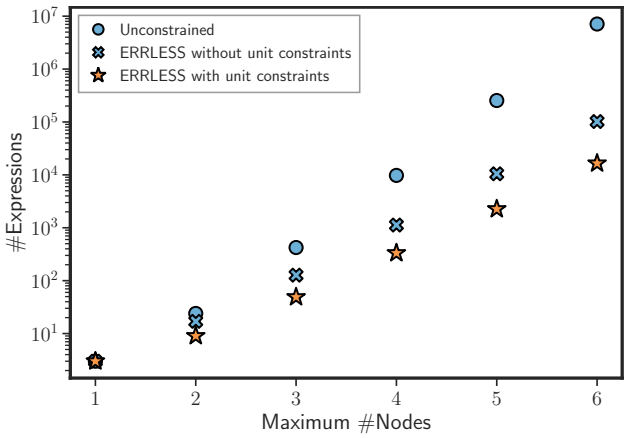


Figure 2: Growth of search space size in maximum number of nodes (in log-scale).

Prior and tempering. We choose a unigram prior $P(\mathrm{T})$ over expression trees (see §A.2 for more details), a choice that we ablate in §D.1. We also impose length constraints and the redundancy constraints described in §3.1. For the prior $p(\boldsymbol{\theta} \mid \mathrm{T})$ over constant assignments, we chose a Gaussian prior with mean 0 and standard deviation of 10 unless specified otherwise with a penalty for the number of constants used. The prior $p(\sigma)$ over σ was chosen to be a wide half-Normal distribution with a scale parameter of 2000 for the Feynman and Blackbox datasets, and a LogNormal distribution for the synthetic dataset.

Training policy. To encourage exploration, we use off-policy training and use ϵ-greedy exploration with annealed ϵ along with a prioritized replay buffer. See §B for all relevant hyperparameters and training details. To stabilize the training objective, inspired by Deleu et al. [2022], we use a modified version of TB that clips the gradients of the squared loss, analogous to the Huber loss.

4 RELATED WORKS

Learning and Search-Based Symbolic Regression. Deep symbolic regression [DSR; Petersen et al., 2021] trains an autoregressive recurrent neural network (RNN) policy via risk-seeking policy gradient algorithms. Tenachi et al. [2023] uses the same learning algorithm but enforces constraints on physical units at each step of the generation. ERRLESS differs from both approaches by taking a Bayesian perspective using maximum entropy RL to train the policy and using a bottom-up generative process. Bastiani et al. [CADSR; 2026] use the Bayesian information criterion

(BIC) as a reward for their risk-seeking policy gradient algorithm which approximates the posterior over expressions with Laplace approximation, but unlike ERRLESS, their learned policy does not sample proportionally to the posterior. Mundhenk et al. [2021] combine DSR with genetic programming, resulting in improved exploration and recovery of benchmark formulas. Cranmer [PySR; 2023] takes a purely evolutionary approach, combining population-based search with gradient-based parameter optimization to establish a widely-adopted, practical baseline that avoids neural policies entirely. Beyond RL and evolutionary approaches, NeSymReS [Biggio et al., 2021] is a transformer pre-trained on an equation corpus, yielding zero-shot generalization across diverse symbolic regression tasks. Kamienny et al. [2023] integrates a pre-trained model within Monte Carlo Tree Search (MCTS) whereas Sun et al. [SPL; 2023] use an MCTS-only approach with no neural policy. Unlike these approaches, ERRLESS does not rely on existing datasets and adopts a Bayesian perspective on symbolic regression.

Bayesian symbolic regression and probabilistic modeling. The method Bayesian Symbolic Regression [BSR; Jin et al., 2019] uses reversible-jump MCMC [Green, 1995] to sample expressions from the posterior distribution. Bomarito and Leser [2026] replaces MCMC with sequential Monte Carlo (SMC). Finally, Guimerà and Sales-Pardo [2026] provides a statistical physics perspective on Bayesian symbolic regression. Unlike existing BSR methods that rely on handcrafted proposal distributions and computationally intensive MCMC sampling, ERRLESS learns an amortized posterior sampler with maximum entropy reinforcement learning.

Sampling discrete structured posteriors with maximum-entropy reinforcement learning. ERRLESS builds upon prior work on maximum entropy RL for sampling from discrete structured distributions [Buesing et al., 2020, Bengio et al., 2021]. The trajectory balance objective [Malkin et al., 2022], equivalent to path consistency learning [Nachum et al., 2017], has been used for posterior inference over decision trees [Mahfoud et al., 2025], causal models [Deleu et al., 2022, 2023], phylogenetic trees [Zhou et al., 2024]. GFN-SR [Li et al., 2023] uses trajectory balance to learn policies for sampling expressions on small synthetic benchmarks. ERRLESS deviates from GFN-SR by using a better generative process for constructing expressions, incorporating explicit priors, and evaluating on large-scale benchmarks complemented by improved training.

5 EXPERIMENTAL SETUP

In this section, we describe our experimental setup. We compare ERRLESS against the baselines from La Cava et al. [2021] and PhySO [Tenachi et al., 2023], a state-of-the-art symbolic regression method that incorporates dimensional analysis constraints. A detailed description of these baselines is provided in §B.4. In §5.1, we briefly summarize the benchmark datasets, and in §5.2, we outline the evaluation metrics. All algorithms are allowed a budget of 1 million reward evaluations as outlined in La Cava et al. [2021].

5.1 DATASETS

Synthetic dataset. We construct a small dataset of seven expressions to closely study the quality of posterior modeling. Each expression has 20 points for training and 100 points for testing. We set the maximum number of nodes to $L = 9$ and the maximum number of constants to $K = 3$. The expressions contain at most two variables. We give an overview of the expressions and their corresponding datasets in §B.1.

Feynman Symbolic Regression Database. We use the Feynman Symbolic Regression Database [Udrescu and Tegmark, 2020], a standard benchmark for symbolic regression comprising 100 expressions from the *Feynman Lectures on Physics* [Feynman et al., 2015] and 20 additional physics-inspired bonus expressions. Following Tenachi et al. [2023], we remove 4 expressions involving arccos or arcsin, leaving 116 expressions in total.[1] Each expression is paired with 1 million sampled points; we subsample 10,000 for training (to compute the reward) and 25,000 for testing, as per the `SRBench` protocol [La Cava et al., 2021]. We set $L = 32$ and $K = 3$.

The benchmark datasets are originally noise-free. Following the protocol of La Cava et al. [2021], we add Gaussian noise to the ground-truth targets in the train set only. The noise is sampled from $\mathcal{N}\left(0, \gamma\sqrt{\frac{1}{N}\sum_{i=1}^{N} y_i^2}\right)$, where γ controls the noise level. For each expression, we run our algorithm with five random seeds and three noise settings: $\gamma \in \{0.001, 0.01, 0.1\}$ on the same training split. Importantly, the noise is sampled once per dataset and kept fixed across all runs to ensure comparability.

Blackbox benchmark. This is a set of 122 datasets intended for benchmarking machine learning methods from PMLB [Romano et al., 2022]. These datasets stress-test algorithms on real data where the ground-truth formula is typically unknown. We use the more challenging subset of 12 datasets curated in Aldeia et al. [2025] and show its metadata in Table 5.

5.2 METRICS

Let $Y = \{y_i\}_{i=1}^N$ (resp. $\hat{Y} = \{\hat{y}_i\}_{i=1}^N$) denote the target (resp. prediction) and their mean $\bar{Y}$ (resp. $\bar{\hat{Y}}$).

[1] The expressions removed are: `feynman_I_26_2`, `feynman_I_30_5`, `feynman_II_11_17` and `feynman_test_10`.

Table 1: **Posterior predictive metrics on the synthetic dataset.** We report the accuracy of the mean of the posterior predictive (R^2_{PP}) as well as the negative log-likelihood (NLL) on the test set, with both methods scored by the same estimator (§A.2): a pointwise mixture NLL with uniform weights over up to 1000 posterior draws, where σ is each draw's sampled value for ERRLESS and each particle's residual error on the training split for PySIPS. Test R^2 is the test-set R^2 of the single best expression found by each method. Median of 10 independent runs. †: only 2 of 10 PySIPS runs yield a finite posterior-mean prediction on this problem.

Metric →	**NLL** (↓)		**Test** R^2 (↑)		R^2_{PP} (↑)	
Expression ↓ Algorithm →	ERRLESS	PySIPS	ERRLESS	PySIPS	ERRLESS	PySIPS
$1/\sqrt{1-x^2}$	60.832	**−57.645**	0.101	**0.532**	**−0.232**	-8.9×10^{53}
$2.37x + 3.02$	114.973	**−3.255**	0.969	**0.974**	**0.760**	0.349
$4.567 \exp x + y$	183.971	**111.125**	0.976	**1.000**	**0.978**	-1.5×10^{91}
$\sin(y + 2.5x)$	182.595	**−80.669**	−0.607	**0.999**	**−5.708**	−70.162
$\sin x \cos y$	343.812	**−20.211**	−2.275	**0.401**	**−11.055**	-1.5×10^{272}
$\sqrt{x^2 + y^2}$	210.603	**−43.306**†	**0.993**	0.815†	**−9.387**	$-\infty^\dagger$
$x + \sin(5.5x)$	377.029	**−13.233**	−0.799	**0.323**	**−2.230**	-2.8×10^{38}

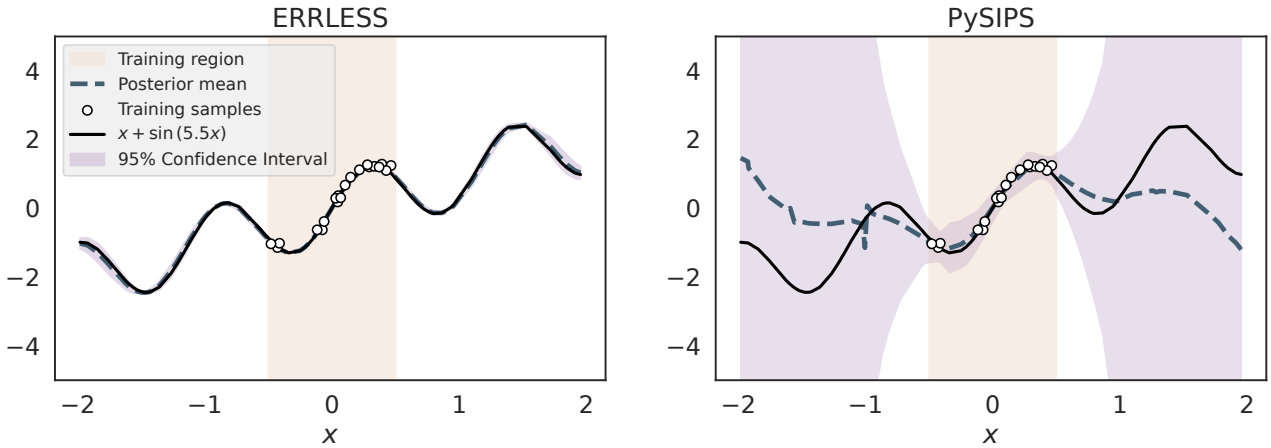


Figure 3: **Samples from the posterior**. ERRLESS and PySIPS are trained on noised values (white circles) of the ground-truth function $x + \sin(5.5x)$ (solid black line) at points in the training domain $[-0.5, 0.5]$ (beige). The posterior mean (dashed blue line), that is, the mean of the individual posterior samples, fits the true function well on the beige interval in both cases, but ERRLESS extrapolates better outside the training region. We show in purple the 95% credible intervals. For ERRLESS both the mean and credible intervals are computed using importance sampling. Results shown are from a single run. Table 1 reports medians over ten runs.

Prediction accuracy. We use the coefficient of determination R^2 between the target and prediction:

$$R^2 := 1 - \frac{\sum_{i=1}^{N} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{N} (y_i - \bar{Y})^2}.$$

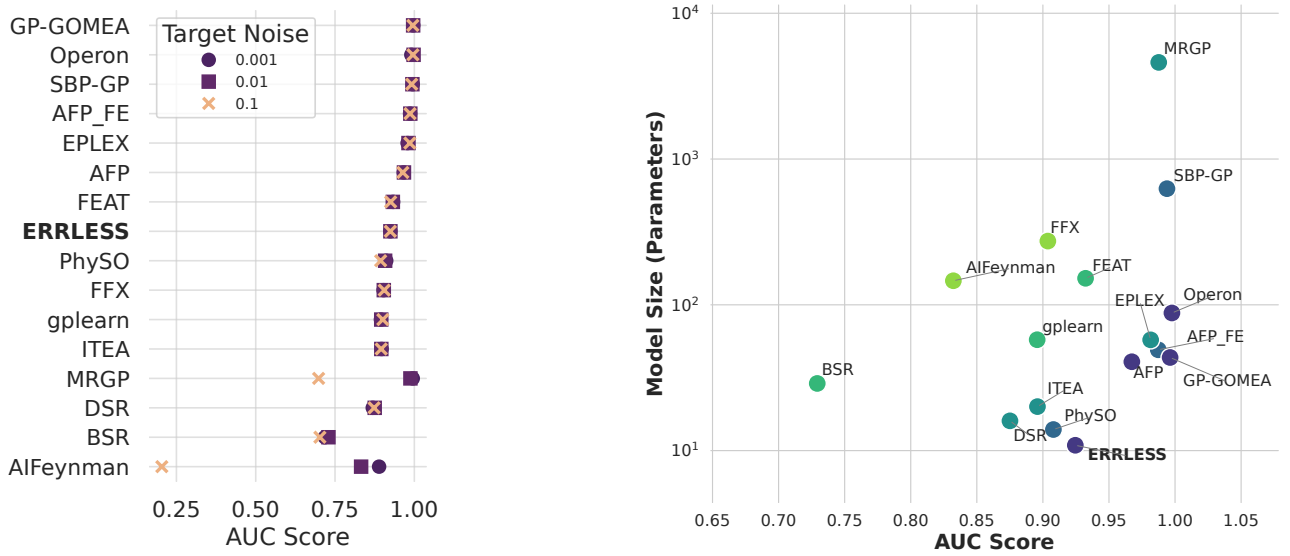
The coefficient $R^2 \leq 1$ quantifies the fraction of variance in the target Y explained by the model: $R^2 = 1$ indicates a perfect fit, while predicting the mean of Y gives $R^2 = 0$.

AUC Score [Aldeia et al., 2025]. To quantify the robustness and predictive accuracy of each algorithm, we use a scalar metric derived from the area under the curve of its success profile. For a given algorithm evaluated over N datasets across S independent runs, we first determine the median prediction accuracy (expressed as the R^2 coefficient) for each dataset to represent its central performance. We then define the success profile as the empirical survival function, $\mathbb{P}[R^2 \geq t]$, which represents the proportion of datasets where the median accuracy meets or exceeds a sliding threshold $t \in [0, 1]$. The final reported metric is obtained by integrating this success profile over the unit interval.

6 RESULTS

6.1 POSTERIOR OVER SMALL EXPRESSIONS

To evaluate the posterior modeling accuracy, we compare ERRLESS against the SMC-based baseline PySIPS [Bomarito and Leser, 2026] on the synthetic datasets introduced in §5.1 (noise level 0.1). We draw 1000 samples from the posterior for both methods. Each method is run with ten random seeds per expression. Table 1 shows that our approach models the posterior over expression trees more ac-



(a) Robustness to target noise

(b) Accuracy vs. Complexity trade-off for $\gamma = 0.01$

Figure 4: **Performance analysis on the Feynman benchmark.** (a) Robustness profile showing the AUC of the empirical success rate $\mathbb{P}[R^2 \geq t]$ across varying noise levels ($\gamma \in \{0.001, 0.01, 0.1\}$). Algorithms are ranked by mean AUC, with ERRLESS demonstrating superior stability. (b) Pareto front illustrating the trade-off between median AUC score and model parameter count. ERRLESS achieves a competitive performance-to-complexity ratio compared to baselines.

curately, as indicated by the posterior predictive mean (R^2_{PP}). PySIPS attains lower NLL and, on most problems, higher single-expression Test R^2: its search is effective at finding individual strong expressions. But its posterior-predictive mean collapses on most problems, because a tail of posterior samples diverges outside the training region and dominates the mean (cf. Fig. 3). This produces the large negative R^2_{PP} values for PySIPS in Table 1. ERRLESS is less prone to this failure: its posterior draws yield a finite predictive mean on every problem. The example in Fig. 3 shows that ERRLESS produces posterior samples that fit the data even though the training region is too small to reveal the true shape of the function.

6.2 FIT QUALITY

ERRLESS achieves a competitive AUC score of 0.924 on the Feynman database, maintaining superior robustness across all noise levels compared to closely related reinforcement learning baselines such as DSR (0.873) and PhySO (0.893), the latter of which incorporates dimensional analysis constraints. As illustrated in Fig. 4, our approach is stable, with almost no variation in AUC score as the noise level increases. ERRLESS also outperforms the Bayesian symbolic regression approach, BSR (0.702). The performance gap between ERRLESS and leading methods, such as GP-GOMEA and Operon, can be attributed to two key factors: (1) unlike most competing methods that optimize directly for reward, ERRLESS must approximate the joint posterior over expressions, constants, and noise, a comprehensive probabilistic task that typically necessitates a larger training budget than the 1M evaluations prescribed by the SRBench framework; and (2) while leading methods often achieve high fit scores, they frequently produce overly complex expressions containing hundreds of constants [Tenachi et al., 2023] (see Fig. 4b). Such results often defeat the primary objective of symbolic regression, which is to identify parsimonious and interpretable expressions.

Blackbox benchmark. From Fig. 6, ERRLESS reaches an AUC of 0.350, which is higher than BSR (0.19) and AIFeynman (0.004) but lower than XGBoost (0.625). This benchmark is difficult because some datasets have a large number of data points, making the likelihood very peaky and hard to navigate for the policy. However, ERRLESS remains competitive in terms of striking a balance between the accuracy and complexity of the generated expressions.

An additional benefit of ERRLESS is its speed: we show in Fig. 7 that it is an order of magnitude faster than most learning-based competitors, which can be attributed to the fact that ERRLESS does not require an inner loop to optimize the constants, the typical bottleneck for SR algorithms.

6.3 QUALITATIVE ANALYSIS

We examine expressions discovered by our sampler and highlight the case of the ground-truth expression $\rho_0/\sqrt{1-\frac{v^2}{c^2}}$. The highest-scoring candidate found by ERRLESS was $\frac{\rho_0}{\cos(v/c)}$. Although these two expressions appear unrelated at first glance, their denominators have the same second-order Taylor approximation in $\frac{v}{c}$ at 0. The length of the postorder representation of the ground-truth expression is greater than

that of its approximate counterpart, which is unfavored by the prior. An extreme example of the same is the ground-truth expression $0.159h\omega / \left(\exp\left(0.159\frac{h\omega}{Tk_B}\right) - 1\right)$, which has the leading-order Taylor approximation Tk_B. The prior strongly favors Tk_B, and the two expressions have similar likelihood: the linear approximation achieves a test set R^2 of 0.99.

These results can be explained by the fact that ERRLESS incorporates a prior that favors concise expressions, as opposed to methods that only optimize the quality of fit. With more data samples, we would expect the likelihood to eventually dominate the prior in the reward, and the ground-truth expression would have a higher score.

7 CONCLUSION

We have introduced ERRLESS, a scalable approach to Bayesian symbolic regression, using maximum-entropy reinforcement learning to amortize posterior sampling over algebraic expressions describing a stochastic dependence of a target variable on its inputs. We formulate expression synthesis as a sequential decision-making process and prune the search space by enforcing dimensional constraints during bottom-up construction. On the Feynman Symbolic Regression Database [Udrescu and Tegmark, 2020], ERRLESS achieves a competitive AUC score while approximating the full posterior distribution, rather than returning a single-point estimate.

Limitations and future work. While ERRLESS accurately models the posterior over expressions, performance can degrade on highly complex target expressions. Future work could condition the sampler on the temperatures of the log-likelihood and the priors to better trade off complexity and accuracy. Additionally, amortizing the sampler over datasets would allow posterior sampling for new datasets without the need for retraining. Symbolic regression methods that amortize over datasets have already shown promise in optimizing for the best expression given a dataset [Biggio et al., 2021, Kamienny et al., 2023]. This naturally motivates extending ERRLESS to learn the operator library itself, allowing for reusable constructs that recur across datasets [Ellis et al., 2021]. Finally, extending our work to discovering symbolic forms of partial and ordinary differential equations would be a strong future direction given their prevalence in science.

Acknowledgements

O. Boussif was supported by the National Research Council Canada (NRC) AI4D program. M. Jain is supported by an FRQNT Doctoral Fellowship (https://doi.org/10.69777/366694). Y. Kaddar was supported by the Oxford–DeepMind Graduate Scholarship. E.S. Whitammer acknowledges support from the CIFAR Learning in Machines and Brains Programme. The research was also enabled by computational resources provided by the Digital Research Alliance of Canada (https://alliancecan.ca) and Mila (https://mila.quebec).

References

Guilherme S. Imai Aldeia, Hengzhe Zhang, Geoffrey Bomarito, Miles Cranmer, Alcides Fonseca, Bogdan Burlacu, William G. La Cava, and Fabrício Olivetti de França. Call for Action: towards the next generation of symbolic regression benchmark. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO)*, pages 2529–2538. ACM, 2025.

Ignacio Arnaldo, Krzysztof Krawiec, and Una-May O'Reilly. Multiple regression genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 879–886. ACM, 2014.

Zachary Bastiani, Robert M. Kirby, Jacob Hochhalter, and Shandian Zhe. Complexity-aware deep symbolic regression with robust risk-seeking policy gradients. *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2026.

Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. *Neural Information Processing Systems (NeurIPS)*, 2021.

Luca Biggio, Tommaso Bendinelli, Alexander Neitz, Aurélien Lucchi, and Giambattista Parascandolo. Neural symbolic regression that scales. *International Conference on Machine Learning (ICML)*, 2021.

Geoffrey F Bomarito and Patrick E Leser. Bayesian symbolic regression via posterior sampling. *Philosophical Transactions of the Royal Society A*, 384(2317):20240590, 2026.

Josh Bongard and Hod Lipson. Automated reverse engineering of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 104(24):9943–9948, 2007.

Lars Buesing, Nicolas Heess, and Theophane Weber. Approximate inference in discrete distributions with Monte Carlo tree search and value functions. *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

Andrei Constantin, Pedro Ferreira, Harry Desmond, and Deaglan Bartlett. Statistical patterns in the equations of physics and the emergence of a meta-law of nature. *Philosophical Transactions of the Royal Society A*, 384 (2317):20250091, 2026.

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

F. O. de França and G. S. I. Aldeia. Interaction-transformation evolutionary algorithm for symbolic regression. *Evolutionary Computation*, 29(3):367–390, 2021.

Tristan Deleu, António Góis, Chris Emezue, Mansi Rankawat, Simon Lacoste-Julien, Stefan Bauer, and Yoshua Bengio. Bayesian structure learning with generative flow networks. *Uncertainty in Artificial Intelligence (UAI)*, 2022.

Tristan Deleu, Mizu Nishikawa-Toomey, Jithendaraa Subramanian, Nikolay Malkin, Laurent Charlin, and Yoshua Bengio. Joint Bayesian inference of graphical structure and parameters with a single generative flow network. *Neural Information Processing Systems (NeurIPS)*, 2023.

Tristan Deleu, Padideh Nouri, Nikolay Malkin, Doina Precup, and Yoshua Bengio. Discrete probabilistic inference as control in multi-path environments. *Uncertainty in Artificial Intelligence (UAI)*, 2024.

Kevin Ellis, Catherine Wong, Maxwell Nye, Mathias Sablé-Meyer, Lucas Morales, Luke Hewitt, Luc Cary, Armando Solar-Lezama, and Joshua B Tenenbaum. DreamCoder: Bootstrapping inductive program synthesis with wake-sleep library learning. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, pages 835–850, 2021.

Benjamin Eysenbach and Sergey Levine. Maximum entropy RL (provably) solves some robust RL problems. *International Conference on Learning Representations (ICLR)*, 2022.

Richard P. Feynman, Robert B. Leighton, and Matthew Sands. *The Feynman lectures on physics, vol. I: The new millennium edition: Mainly mechanics, radiation, and heat*. Basic Books, 2015.

Peter J Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82(4):711–732, 1995.

Timofei Gritsaev, Nikita Morozov, Sergey Samsonov, and Daniil Tiapkin. Optimizing backward policies in GFlowNets via trajectory likelihood maximization. *International Conference on Learning Representations (ICLR)*, 2025.

Roger Guimerà and Marta Sales-Pardo. Bayesian symbolic regression: Automated equation discovery from a physicist's perspective. *Philosophical Transactions of the Royal Society A*, 384(2317):20250089, 2026.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *International Conference on Machine Learning (ICML)*, 2018.

Ying Jin, Weilin Fu, Jian Kang, Jiadong Guo, and Jian Guo. Bayesian symbolic regression. *arXiv preprint arXiv:1910.08892*, 2019.

Pierre-Alexandre Kamienny, Guillaume Lample, Sylvain Lamprier, and Marco Virgolin. Deep generative symbolic regression with Monte-Carlo-tree-search. *International Conference on Machine Learning (ICML)*, 2023.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2015.

Michael Kommenda, Bogdan Burlacu, Gabriel Kronberger, and Michael Affenzeller. Parameter identification for symbolic regression using nonlinear least squares. *Genetic Programming and Evolvable Machines*, 21(3):471–501, 2020.

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(3):377–402, September 2019a.

William La Cava, Tilak Raj Singh, James P. Taggart, Srinivas Suri, and Jason H. Moore. Learning concise representations for regression by evolving networks of trees. *International Conference on Learning Representations (ICLR)*, 2019b.

William La Cava, Patryk Orzechowski, Bogdan Burlacu, Fabrício Olivetti de França, Marco Virgolin, Ying Jin, Michael Kommenda, and Jason H. Moore. Contemporary symbolic regression methods and their relative performance. *Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks*, 2021.

Sida Li, Ioana Marinescu, and Sebastian Musslick. GFN-SR: Symbolic regression with generative flow networks. *arXiv preprint arXiv:2312.00396*, 2023.

Mohammed Mahfoud, Ghait Boukachab, Michał Koziarski, Alex Hernández-García, Stefan Bauer, Yoshua Bengio, and Nikolay Malkin. Learning decision trees as amortized structure inference. *arXiv preprint arXiv:2503.06985*, 2025.

Nikolay Malkin, Moksh Jain, Emmanuel Bengio, Chen Sun, and Yoshua Bengio. Trajectory balance: Improved credit assignment in GFlowNets. *Neural Information Processing Systems (NeurIPS)*, 2022.

Trent McConaghy. FFX: Fast, scalable, deterministic symbolic regression technology. In *Genetic Programming Theory and Practice IX*, pages 235–260. Springer, 2011.

T Nathan Mundhenk, Mikel Landajuela, Ruben Glatt, Claudio P Santiago, Daniel M Faissol, and Brenden K Petersen. Symbolic regression via neural-guided genetic programming population seeding. *Neural Information Processing Systems (NeurIPS)*, 2021.

Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the gap between value and policy based reinforcement learning. *Neural Information Processing Systems (NeurIPS)*, 2017.

Brenden K Petersen, Mikel Landajuela, T Nathan Mundhenk, Claudio P Santiago, Soo K Kim, and Joanne T Kim. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. *International Conference on Learning Representations (ICLR)*, 2021.

Joseph D. Romano, Trang T. Le, William La Cava, John T. Gregg, Daniel J. Goldberg, Praneel Chakraborty, Natasha L. Ray, Daniel Himmelstein, Weixuan Fu, and Jason H. Moore. PMLB v1.0: an open-source dataset collection for benchmarking machine learning methods. *Bioinformatics*, 38(3):878–880, 2022.

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

Michael Schmidt and Hod Lipson. Age-fitness Pareto optimization. In *Genetic Programming Theory and Practice VIII*, pages 129–146. Springer, 2011.

Trevor Stephens. gplearn: Genetic programming in Python, with a scikit-learn inspired API. https://github.com/trevorstephens/gplearn, 2015.

Fangzheng Sun, Yang Liu, Jian-Xun Wang, and Hao Sun. Symbolic Physics Learner: Discovering governing equations via Monte Carlo tree search. *International Conference on Learning Representations (ICLR)*, 2023.

Wassim Tenachi, Rodrigo Ibata, and Foivos I. Diakogiannis. Deep symbolic regression for physics guided by units constraints: Toward the automated discovery of physical laws. *The Astrophysical Journal*, 959(2):99, 2023.

Daniil Tiapkin, Nikita Morozov, Alexey Naumov, and Dmitry P Vetrov. Generative flow networks as entropy-regularized RL. *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2024.

Silviu-Marian Udrescu and Max Tegmark. AI Feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16), April 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Neural Information Processing Systems (NeurIPS)*, 2017.

Marco Virgolin, Tanja Alderliesten, and Peter AN Bosman. Linear scaling with and within semantic backpropagation-based genetic programming for symbolic regression. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 1084–1092, 2019.

Marco Virgolin, Tanja Alderliesten, Cees Witteveen, and Peter A N Bosman. Improving model-based genetic programming for symbolic regression of small expressions. *Evolutionary Computation*, 29(2):211–237, 2021.

Mingyang Zhou, Zichao Yan, Elliot Layne, Nikolay Malkin, Dinghuai Zhang, Moksh Jain, Mathieu Blanchette, and Yoshua Bengio. PhyloGFN: Phylogenetic inference with generative flow networks. *International Conference on Learning Representations (ICLR)*, 2024.

Jan Łukasiewicz. *Elementy logiki matematycznej*. Wydawnictwa Koła matematyczno-fizycznego słuchaczów Uniwersytetu warszawskiego, t. 18. Nakładem komisji wydawniczej Koła matematyczno-fizycznego słuchaczów Uniwersytetu warszawskiego, [Warszawa], 1929.

Bayesian Symbolic Regression with Entropic Reinforcement Learning (Supplementary Material)

Oussama Boussif[1,2] **Mohammed Mahfoud**[3] **Younesse Kaddar**[4] **Moksh Jain**[1,2] **Sida Li**[5] **Damiano Fornasiere**[6]
Xiaoyin Chen[1,2] **Yoshua Bengio**[1,2,7] **Esmeralda S. Whitammer**[7,8]

[1]Mila – Québec AI Institute [2]Université de Montréal [3]Independent [4]University of Oxford [5]University of Chicago [6]LawZero [7]CIFAR Fellow [8]University of Edinburgh

A BOTTOM-UP GENERATION

A.1 OPERATOR PHYSICAL UNIT CONSTRAINTS AND ASSIGNMENTS

We show the values of the partial *unit assignment* function for the operators that we use in our library in Table 2.

Table 2: Physical unit assignment function for the full operator library. When a variable is dimensionless, its unit vector is $\mathbf{0}$.

Operator g	Unit assignment function $\mathcal{U}_g$
$+$	$\mathcal{U}_+(\boldsymbol{u}_1, \boldsymbol{u}_2) = \boldsymbol{u}_1$ if $\boldsymbol{u}_1 = \boldsymbol{u}_2$, else undefined
$-$	$\mathcal{U}_-(\boldsymbol{u}_1, \boldsymbol{u}_2) = \boldsymbol{u}_1$ if $\boldsymbol{u}_1 = \boldsymbol{u}_2$, else undefined
$\times$	$\mathcal{U}_\times(\boldsymbol{u}_1, \boldsymbol{u}_2) = \boldsymbol{u}_1 + \boldsymbol{u}_2$
$/$	$\mathcal{U}_/(\boldsymbol{u}_1, \boldsymbol{u}_2) = \boldsymbol{u}_1 - \boldsymbol{u}_2$
sin	$\mathcal{U}_{\sin}(\boldsymbol{u}) = \mathbf{0}$ if $\boldsymbol{u} = \mathbf{0}$, else undefined
cos	$\mathcal{U}_{\cos}(\boldsymbol{u}) = \mathbf{0}$ if $\boldsymbol{u} = \mathbf{0}$, else undefined
log	$\mathcal{U}_{\log}(\boldsymbol{u}) = \mathbf{0}$ if $\boldsymbol{u} = \mathbf{0}$, else undefined
exp	$\mathcal{U}_{\exp}(\boldsymbol{u}) = \mathbf{0}$ if $\boldsymbol{u} = \mathbf{0}$, else undefined
$\square^2$	$\mathcal{U}_{\square^2}(\boldsymbol{u}) = 2\boldsymbol{u}$
$\sqrt{\square}$	$\mathcal{U}_{\sqrt{\square}}(\boldsymbol{u}) = \frac{1}{2}\boldsymbol{u}$
$-\square$	$\mathcal{U}_{-\square}(\boldsymbol{u}) = \boldsymbol{u}$

A.2 REWARD FUNCTION

Unigram prior over expressions. Following Constantin et al. [2026], the frequency of mathematical operators in physics equations decays exponentially with rank. We leverage this observation by constructing a unigram prior over operators based on their empirical frequencies. Specifically, we use *Encyclopaedia Inflationaris* as the reference corpus, extract operator frequencies (Table 1 in their paper), discard operators not included in our library, and renormalize the distribution. The resulting frequencies for our operator set are shown in Table 3.

Prior over constant assignments For the **synthetic** dataset we use a Gaussian with mean 0 and standard deviation 10 whereas for the **Feynman** and **Blackbox** datasets the standard deviation is 20.

Prior over the noise σ For the **synthetic** dataset we use $\log \sigma \sim \mathcal{N}(0, 5^2)$, *i.e.* $\sigma \sim \text{LogNormal}(0, 5)$, except for $\sqrt{x^2 + y^2}$, where $\log \sigma \sim \mathcal{N}(-1, 5^2)$; for the **Feynman** and **Blackbox** datasets we use a Half-Normal with scale parameter 2000.

Table 3: Unigram prior over operators, variables, and constants. Frequencies are normalized after restricting to our operator library.

Token	Frequency
$\times$	0.1770
$/$	0.1328
$-$	0.0476
$+$	0.0454
$\square^2$	0.0365
exp	0.0210
$\sqrt{\square}$	0.0199
$-\square$	0.0177
log	0.0133
cos	0.0072
sin	0.0048
Variables	0.2877
Constants	0.1892

Posterior-predictive NLL estimator The NLL column of Table 1 is computed identically for both methods as

$$\mathrm{NLL} \;=\; -\sum_{i=1}^{N} \log \frac{1}{J} \sum_{j=1}^{J} \mathcal{N}\big(y_i;\, f_{\mathrm{T}_j,\boldsymbol{\theta}_j}(\boldsymbol{x}_i),\, \sigma_j^2\big),$$

where $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$ is the test set, $(\mathrm{T}_j, \boldsymbol{\theta}_j)$ is the j-th posterior draw, J is the number of retained draws (up to 1000; draws with non-finite predictions are discarded), and σ_j is the j-th draw's sampled noise value for ERRLESS and the j-th particle's root-mean-square residual on the training split for PySIPS.

B EXPERIMENTAL DETAILS

B.1 DATASETS

Synthetic. Table 4 shows the seven expressions we used for the benchmark alongside the ranges used for the training and test set, respectively.

Table 4: **Synthetic dataset**. Variables are sampled uniformly from the specified intervals.

Expression	Training range	Test range
$1/\sqrt{1-x^2}$	$x \sim \mathcal{U}(-0.6,\, 0.6)$	$x \sim \mathcal{U}(-0.99,\, 0.99)$
$2.37x + 3.02$	$x \sim \mathcal{U}(-0.5,\, 0.5)$	$x \sim \mathcal{U}(-1,\, 1)$
$x + \sin(5.5x)$	$x \sim \mathcal{U}(-0.5,\, 0.5)$	$x \sim \mathcal{U}(-2,\, 2)$
$4.567 \exp x + y$	$x, y \sim \mathcal{U}(-1,\, 1)$	$x, y \sim \mathcal{U}(-2,\, 2)$
$\sin(y + 2.5x)$	$x, y \sim \mathcal{U}(0,\, 1)$	$x, y \sim \mathcal{U}(0,\, 3)$
$\sqrt{x^2 + y^2}$	$x, y \sim \mathcal{U}(0,\, 1)$	$x, y \sim \mathcal{U}(0,\, 3)$
$\sin(x)\cos(y)$	$x, y \sim \mathcal{U}(0,\, 1)$	$x, y \sim \mathcal{U}(0,\, 3)$

Feynman Symbolic Regression Database. We show characteristics of the Feynman Symbolic Regression Database [Udrescu and Tegmark, 2020] (see Fig. 5). In particular: (i) the distribution over the number of variables in a given expression in the dataset, (ii) the distribution over expression lengths of a given expression in the dataset, and (iii) the ratio of expressions with physical units compared to unitless ones.

Table 5: **Blackbox benchmark metadata.** We show the dataset size and number of features of each dataset in the Blackbox benchmark.

Dataset	Size	Number of features
1028_SWD	1000	11
1089_USCrime	47	14
1193_BNG_lowbwt	31104	10
1199_BNG_echoMonths	17496	10
192_vineyard	52	3
210_cloud	108	6
522_pm10	500	8
557_analcatdata_apnea1	475	4
579_fri_c0_250_5	250	6
606_fri_c2_1000_10	1000	11
650_fri_c0_500_50	500	51
678_visualizing_environmental	111	4

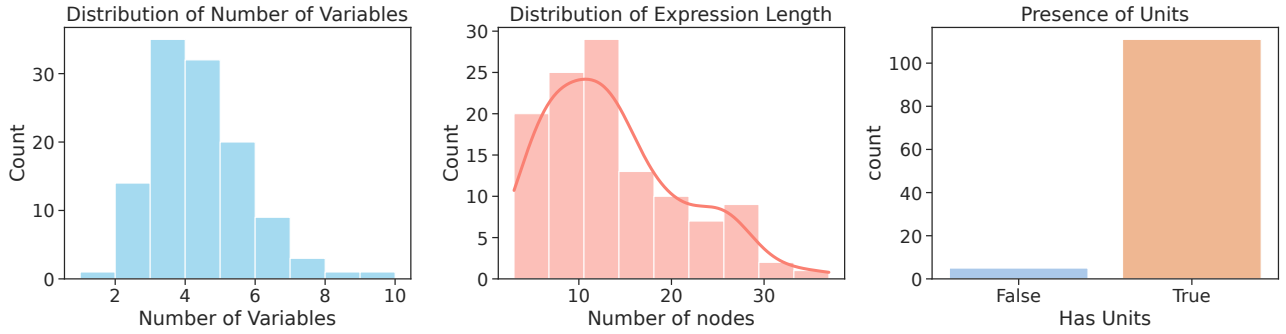


Figure 5: **Exploratory analysis of the Feynman Symbolic Regression Database**. Left: histogram of the number of variables per expression. Center: distribution of expression tree lengths. Right: proportion of expressions in which all variables are associated with physical units.

Blackbox benchmark. We show the metadata of the benchmark in Table 5 where for each dataset we show the number of its datapoints as well as the number of features.

B.2 REPLAY BUFFER

We use a modified version of the prioritized replay buffer where each expression tree T can be stored up to a maximum of N_{repeat} times in the buffer. This is to ensure that the model is trained on different values of the constants for the same expression. When a new batch of elements is added, we apply a first filtering step where we only keep the ones that have a reward higher than the minimal reward in the buffer. This ensures that the minimum reward in the buffer never decreases, and that we do not hinder the quality of the samples in the buffer. After that, we perform a second filtering step where we make sure that none of the added expression trees are repeated more than N_{repeat} times.

B.3 HYPERPARAMETERS

Policy. Recall that an expression tree T is uniquely represented by its postorder sequence $\mathcal{W}(\text{T}) \in \Sigma^*$. We first embed it using a learnable embedding matrix with hidden dimension 256 to get a representation. A learned positional embedding is then added to this representation, which is processed by a Transformer encoder with 2 layers and 4 attention heads [Vaswani et al., 2017]. The output of the encoder is used in three ways:

- Action selection: A linear layer maps the representation to logits over the discrete action space (operators, variables, and termination).

- Constant generation: We approximate the generative distribution of the constants using a Gaussian Mixture Model (GMM) with 5 components. The latent representation produced by the transformer is processed by three distinct linear heads: (1) one to parameterize the means of each Gaussian component, (2) one to determine the corresponding covariances, and (3) one to predict the mixture weights.
- Noise generation: In a manner analogous to constant generation, the noise distribution is approximated by a mixture model. For this task, we use a Log-Normal mixture to ensure the generated noise parameters remain within a valid, positive-valued domain. The number of components in the mixture is also set to 5.

Off-policy training. We use an ϵ-greedy policy where ϵ decays linearly from 1 to 0.05 midway through the training and then for the other half it stays at 0.05. We use a prioritized replay buffer with a capacity of 10000 trajectories and $N_{\text{repeat}} = 3$. The batch size is set to 800 and the ratio of samples coming from the replay buffer decays linearly from 0.9 to 0.2 (see §D.3 for ablation). We train the policy for a total of 1250 iterations, which corresponds to 1M reward function calls.

Reward function. The inverse temperature of the prior factors is $\alpha = 1$ for the expression tree log-prior and $\lambda = 1$ for the parameter log-prior.

Optimization. We use the Adam optimizer without weight decay [Kingma and Ba, 2015]. The policy $\log \pi_\varphi(\mathrm{T}, \boldsymbol{\theta}, \sigma)$ learning rate is set to 0.0001 and that of the log-partition function $\log Z_\varphi$ to 0.01.

B.4 BASELINES

AFP / AFP_FE [Schmidt and Lipson, 2011, 2009]. Age-Fitness Pareto optimization (AFP) is a genetic programming (GP) strategy that frames search as a bi-objective problem over prediction error and individual age. Each candidate solution is assigned a fitness value and an age, defined as the number of generations since creation. At each generation, selection operates on the Pareto front with respect to these two objectives, favoring individuals that are either accurate and relatively young or novel relative to the current population. By rewarding both accuracy and youth, AFP maintains diversity and reduces premature convergence, while still steering the search toward low-error expressions.

AIFeynman [Udrescu and Tegmark, 2020]. AIFeynman is a physics-inspired, multi-stage method designed to discover symbolic expressions by systematically breaking down a complex problem into simpler ones. It does not learn a single generative model but rather follows a deterministic, divide-and-conquer strategy. First, a neural network is trained to high accuracy on the dataset $\mathcal{D}$. This network is then treated as an "oracle" and is probed to discover properties of the underlying function, such as symmetries, separability, or polynomial structure. Based on these discovered properties, the original problem is recursively simplified. The final, simplified sub-problems are then solved using a combination of brute-force search and polynomial fitting.

BSR [Jin et al., 2019]. Bayesian symbolic regression (BSR) directly addresses the problem of posterior inference over the space of expressions. Similar to our approach, BSR also aims to sample from the posterior distribution $p(\mathrm{T}, \boldsymbol{\theta} \mid \mathcal{D})$, where T is the expression tree structure and $\boldsymbol{\theta}$ represents its constant parameters. Due to the varying dimensionality of its parameter space (due to constantly changing tree structure), BSR employs Reversible-Jump MCMC (RJMCMC), a technique that allows the MCMC sampler to propose "moves" between models of different dimensions (*e.g.*, adding or removing a node in the tree T) while maintaining the detailed balance condition. This approach relies on handcrafted proposal distributions for these moves, so it can be computationally intensive and often fails to explore the posterior landscape sufficiently.

DSR [Petersen et al., 2021]. Deep symbolic regression (DSR) was a seminal work that first framed the symbolic regression task as a reinforcement learning (RL) problem. DSR employs a Recurrent Neural Network (RNN) as a policy, which autoregressively generates an expression tree T token by token in a top-down, pre-order traversal. A complete expression constitutes a trajectory, and the quality of this expression (*e.g.*, its R^2 score on the dataset $\mathcal{D}$) serves as the reward $R(\mathrm{T})$. The policy is trained using a risk-seeking policy gradient algorithm, which biases the search towards high-reward expressions and helps escape local optima. The ultimate goal of DSR is to find a single best-fitting expression that maximizes the expected reward.

PhySO [Tenachi et al., 2023]. PhySO uses the same learning framework as DSR for training its policy, but it adds physical unit constraints where it forces the expression to be dimensionally valid at each generation step.

EPLEX [La Cava et al., 2019a]. EPLEX is an advanced parent selection method for Genetic Programming designed to excel in continuous-valued regression tasks. Traditional selection methods rely on an aggregate fitness score (like average error), which loses information about performance on individual data points. EPLEX addresses this by filtering the population sequentially on a random ordering of individual training cases.

FEAT [La Cava et al., 2019b]. FEAT is a hybrid method for symbolic regression that combines evolutionary computation with linear models. Instead of evolving a single monolithic expression, FEAT evolves a set of simpler expression trees that serve as features. These features are then used as inputs to a linear model. The key innovation is a feedback mechanism where the coefficients learned by the linear model are used to guide the evolutionary search, prioritizing the mutation and replacement of less impactful features.

FFX [McConaghy, 2011]. FFX is a non-evolutionary, deterministic algorithm for symbolic regression that casts the problem as a feature selection task within a generalized linear model. The method operates in two main stages: first, it deterministically generates a massive library of candidate basis functions by applying a predefined set of nonlinear operators and interactions to the input variables. Second, it employs path-wise regularized learning (specifically, an elastic net) to efficiently search this vast feature space.

GP-GOMEA [Virgolin et al., 2021]. GP-GOMEA is a model-based evolutionary algorithm that aims to improve search efficiency by explicitly learning and exploiting the structure of promising solutions. Unlike traditional GP, which relies on blind genetic operators like crossover and mutation, GP-GOMEA learns a "linkage model" in each generation. This model, typically a Linkage Tree built using mutual information between nodes in the population's expression trees, identifies groups of genes (sub-programs) that work well together.

gplearn [Stephens, 2015]. `gplearn` is a Python library that implements tree-based genetic programming for symbolic regression within the `scikit-learn` API [Buitinck et al., 2013]. Candidate models are mathematical expression trees evolved using standard GP operators such as sub-tree crossover and mutation, with fitness measured by prediction error.

ITEA [de França and Aldeia, 2021]. ITEA is an evolutionary algorithm that operates on a constrained representation called Interaction Transformation (IT). Unlike the free-form trees in traditional GP, an IT expression is restricted to a linear combination of nonlinear terms. ITEA evolves a population of these structured expressions using a mutation-only strategy.

MRGP [Arnaldo et al., 2014]. MRGP is a hybrid technique that integrates tree-based GP with the LASSO regularization method. Unlike conventional GP, MRGP does not directly compare the final program's output with the target variable. Instead, it constructs a set of sub-expressions from the program and fits a linear combination of these sub-expressions to the target output. The target variable is then compared against the output of the resulting regression model.

Operon [Kommenda et al., 2020]. Operon is a modern, highly efficient C++ framework for GP in symbolic regression. It focuses on improving performance and scalability through advanced architectural choices, including representing expression trees in a cache-friendly, continuous memory layout. It also implements a fine-grained, low-overhead concurrency model for parallel execution.

SBP-GP [Virgolin et al., 2019]. SBP-GP is a GP method that guides variation using semantic backpropagation (SB). Instead of random tree changes, SB computes the output each sub-tree should produce to help the overall expression match the target, by propagating the target values down through the tree using function inverses. New sub-trees are then generated or selected to better match these desired outputs.

C POSTERIOR PREDICTIVE

For any test input $\boldsymbol{x}^*$, the posterior predictive distribution of the corresponding output y^* is given by:

$$p(\mathrm{y}^* \mid \boldsymbol{x}^*, \mathcal{D}) = \sum_{\mathrm{T}} \int p(\mathrm{y}^* \mid \boldsymbol{x}^*, \mathrm{T}, \boldsymbol{\theta})\, p(\mathrm{T}, \boldsymbol{\theta} \mid \mathcal{D})\, \mathrm{d}\boldsymbol{\theta} \tag{6}$$

Intuitively, this distribution weighs the contribution of all expression trees and their parameters.

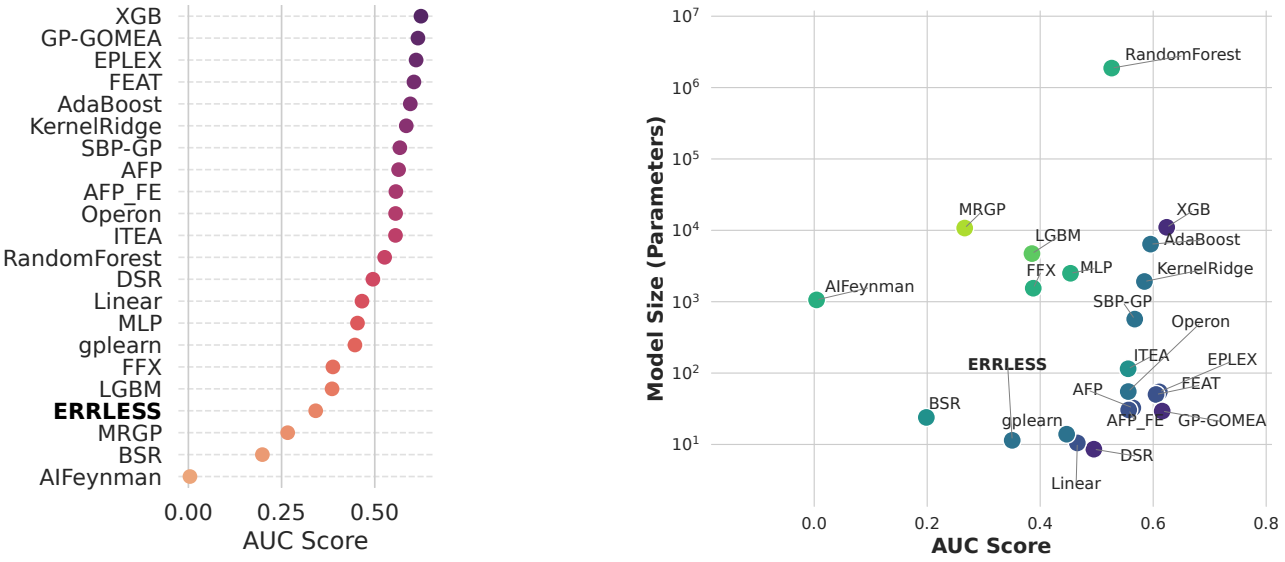
Mean of the posterior predictive. In general, the mean is:

$$\mathbb{E}\left[\mathrm{y}^* \mid \boldsymbol{x}^*, \mathcal{D}\right] = \sum_{T} \int p(\mathrm{T}, \boldsymbol{\theta} \mid \mathcal{D}) \mathbb{E}\left[\mathrm{y}^* \mid \boldsymbol{x}^*, \mathrm{T}, \boldsymbol{\theta}\right] \mathrm{d}\boldsymbol{\theta}$$

Since the likelihood model is a Gaussian (1), we have $\mathbb{E}\left[\mathrm{y}^* \mid \boldsymbol{x}^*, \mathrm{T}, \boldsymbol{\theta}\right] = f_{\mathrm{T},\boldsymbol{\theta}}(\boldsymbol{x}^*)$ which yields:

$$\mathbb{E}\left[\mathrm{y}^* \mid \boldsymbol{x}^*, \mathcal{D}\right] = \mathbb{E}_{\mathrm{T},\boldsymbol{\theta}}\left[f_{\mathrm{T},\boldsymbol{\theta}}(\boldsymbol{x}^*)\right] \tag{7}$$

In practice, we compute the above quantity by drawing expression trees with their parameters from the learned approximation to the posterior. A tempered posterior can be obtained by weighting these samples by their probability under the trained



(a) AUC Score on the Blackbox benchmark

(b) Accuracy vs. Complexity trade-off

Figure 6: **Performance analysis on the Blackbox benchmark.** (a) Area Under the Curve (AUC) of the empirical success rate $\mathbb{P}[R^2 \geq t]$. Algorithms are ranked by mean AUC. (b) Pareto front illustrating the trade-off between median AUC score and model parameter count. ERRLESS achieves a competitive performance-to-complexity ratio compared to baselines.

sampler raised to a power. For example, for a posterior at temperature $\frac{1}{2}$, we simply reweight the samples by their likelihood under the trained model.

D ABLATION RESULTS

We benchmark the design choices by ablating the expression prior and the objective and by running a sensitivity analysis on the mixing coefficient of the replay buffer on the synthetic dataset (see §B.1). In all ablation tables, each configuration is run with five random seeds per expression: model size is computed for each expression as the median over seeds and then averaged over the synthetic expressions.

D.1 EXPRESSION PRIOR

We run ERRLESS with different choices of the expression prior $P(\mathrm{T})$. Because the learning problem is invariant to multiplication of the reward by a scalar, one can define priors by choices of the term $P(\mathrm{T})$ that do not necessarily sum to 1 over $\mathrm{T} \in \mathcal{T}_\Sigma$. Such unnormalized choices are well-defined: the maximum number of nodes is bounded in all our experiments, so the set of reachable expression trees is finite and every positive $P(\mathrm{T})$ restricted to it induces a proper prior after normalization.

- **Unigram prior**: This is the original expression prior used for the main results in the paper; see §A.2.
- **Uniform prior**: $\log P(\mathrm{T}) = 0$.
- **PCFG prior**: A probabilistic context-free grammar prior fit on the Wikipedia named equations and Encyclopaedia Inflationaris [Constantin et al., 2026].
- **Nodes prior**: This is a prior that penalizes the number of nodes in the expression and is given by $\log P(\mathrm{T}) = -n(\mathrm{T})$, where $n(\mathrm{T})$ is the number of nodes in T.

Table 6: Ablation results on the expression prior for different noise levels.

Noise level	Prior	AUC score	Model size
0.001	Nodes Prior	0.817349	7.857143
	PCFG Prior	0.799313	5.857143
	Uniform Prior	0.822502	8.142857
	Unigram Prior	0.818208	6.142857
0.010	Nodes Prior	0.817349	7.857143
	PCFG Prior	0.770684	5.428571
	Uniform Prior	0.817349	7.857143
	Unigram Prior	0.776410	5.428571
0.100	Nodes Prior	0.895076	6.142857
	PCFG Prior	0.548382	5.428571
	Uniform Prior	0.579588	10.000000
	Unigram Prior	0.778414	5.285714

The Unigram prior performs consistently well across all noise levels (Table 6). The Nodes prior excels in high-noise settings, where the Uniform prior struggles despite generating longer expressions. Similarly, the PCFG prior degrades as noise increases.

D.2 TRAINING OBJECTIVE

In this section we ablate the maximum-entropy RL training objective and compare trajectory balance (TB), which we used for the main results, against detailed balance (DB), equivalent in our setting to optimization of a soft Bellman error as in Haarnoja et al. [2018], cf. Tiapkin et al. [2024], Deleu et al. [2024].

Recall from §3.1 that an expression tree T is uniquely represented by its postorder representation $w_1 \dots w_n \top$. DB requires learning a scalar value (log-flow) function $\log F_\varphi(w_1 \dots w_t)$ (where $t \leq n+1$ and $w_{n+1} = \top$). The objective associated with $(\mathrm{T}, \boldsymbol{\theta}, \sigma)$ is:

$$\mathcal{L}_{\mathrm{DB}}(\mathrm{T}, \boldsymbol{\theta}, \sigma; \varphi) \coloneqq \left[\log F_\varphi(w_1 \dots w_t) + \log \pi_\varphi(w_{t+1} | w_1 \dots w_t) - \log F_\varphi(w_1 \dots w_{t+1})\right]^2, \tag{8}$$

where we impose that $F_\varphi(w_1 \dots w_n \top) = R(\mathrm{T}, \boldsymbol{\theta}, \sigma)$.

Table 7: Ablation results on the training objective for different noise levels.

Noise level	Objective	AUC score	Model size
0.001	DB	0.844260	7.714286
	TB	0.818208	6.142857
0.010	DB	0.786859	6.857143
	TB	0.776410	5.428571
0.100	DB	0.748354	7.428571
	TB	0.778414	5.285714

From Table 7, we observe that the difference in AUC is inconclusive, but that TB tends to produce shorter expressions on average, especially as the noise level increases. We note, however, that DB requires an additional flow network, which adds parameter count and consumes more resources.

D.3 REPLAY BUFFER MIXING COEFFICIENT

We perform a sensitivity analysis where we fix the mixing coefficient (instead of annealing it from 0.9 to 0.2 as we do for the main results) on the synthetic dataset for noise level 0.01:

Table 8: Sensitivity analysis for the replay buffer mixing coefficient at noise level 0.01.

Mixing coefficient	AUC score	Model size
0.1	0.853851	7.857143
0.2	0.817349	7.428571
0.3	0.813914	7.857143
0.4	0.960063	6.857143
0.5	0.947466	7.571429
0.6	0.817349	7.428571
0.7	0.952906	7.142857
0.8	0.960063	6.857143
0.9	0.817349	7.857143

The best performance is obtained for mixing coefficients of 0.4 and 0.8 (Table 8). More critically, this shows that fixing the mixing coefficient works better than annealing in this case.

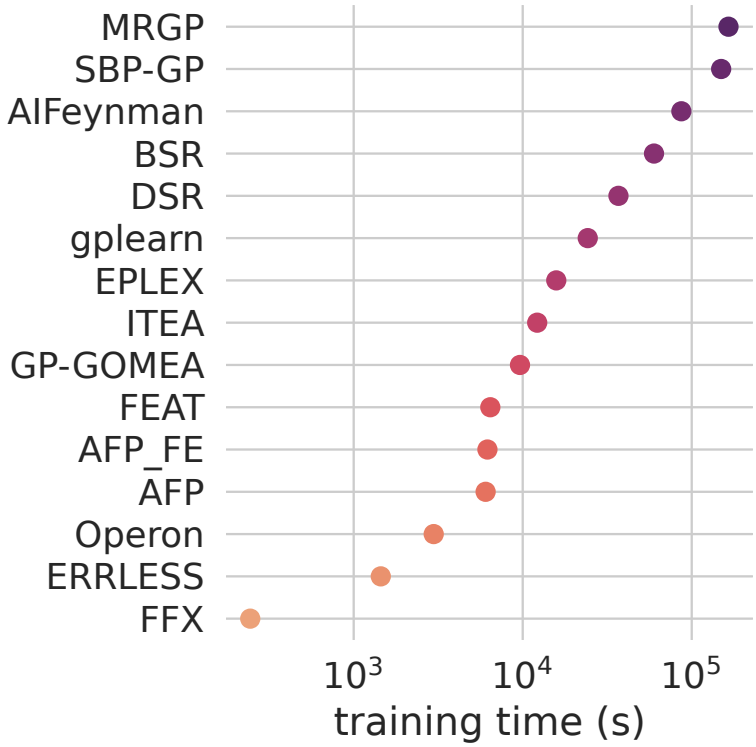


Figure 7: **Computational runtime.** Average of the time (in seconds) taken by all methods over random seeds and expressions in the Blackbox benchmark.

E COMPUTATIONAL RUNTIME

Fig. 7 reports the runtime (in seconds) for all methods evaluated on the blackbox dataset. PhySO is omitted because its authors do not provide runtime measurements. Baseline runtimes are taken from SRBench [La Cava et al., 2021]. ERRLESS was run on a machine equipped with an L40S GPU, 4 CPUs, and 48 GB of RAM.

From the figure, we observe that our method is among the fastest. While hardware differences and code optimization must be considered for a fully fair comparison, we emphasize that, unlike the baselines, our approach avoids a major bottleneck: parameter optimization for expressions. A key advantage of our method is that it learns a posterior over both expressions and parameters, eliminating the need for computationally expensive optimization procedures.