Vertical Symbolic Regression via Deep Policy Gradient

Nan Jiang , Md Nasim , Yexiang Xue
Department of Computer Science, Purdue University
{jiang631, mnasim, yexiang}@purdue.edu

Abstract

Vertical Symbolic Regression (VSR) has recently been proposed to expedite the discovery of symbolic equations with many independent variables from experimental data. VSR reduces the search spaces following the vertical discovery path by building from reduced-form equations involving a subset of variables to all variables. While deep neural networks have shown promise in enhancing symbolic regression, directly integrating VSR with deep networks faces challenges such as gradient propagation and engineering complexities due to the tree representation of expressions. We propose **V**ertical **S**ymbolic **R**egression using **D**eep **P**olicy **G**radient (VSR-DPG) and demonstrate that VSR-DPG can recover ground-truth equations involving multiple input variables, significantly beyond both deep reinforcement learning-based approaches and previous VSR variants. Our VSR-DPG models symbolic regression as a sequential decision-making process, in which equations are built from repeated applications of grammar rules. The integrated deep model is trained to maximize a policy gradient objective. Experimental results demonstrate that our VSR-DPG significantly outperforms popular baselines in identifying both algebraic equations and ordinary differential equations on a series of benchmarks.

1 Introduction

Exciting progress has been made to accelerate scientific discovery using Artificial Intelligence (AI) [Langley *et al.*, 1987; Kulkarni and Simon, 1988; Wang *et al.*, 2023]. Symbolic regression, as an important task in AI-driven scientific discovery, distills physics models in the form of symbolic equations from experiment data [Schmidt and Lipson, 2009]. Notable progress in symbolic regression encompasses diverse methodologies, includes search-based methods [Lenat, 1977], genetic programming [Schmidt and Lipson, 2009; Virgolin *et al.*, 2019], Monte Carlo tree search [Todorovski and Dzeroski, 1997; Sun *et al.*, 2023; Kamienny *et al.*, 2023], and deep reinforcement learning [Petersen *et al.*, 2021; Mundhenk *et al.*, 2021].

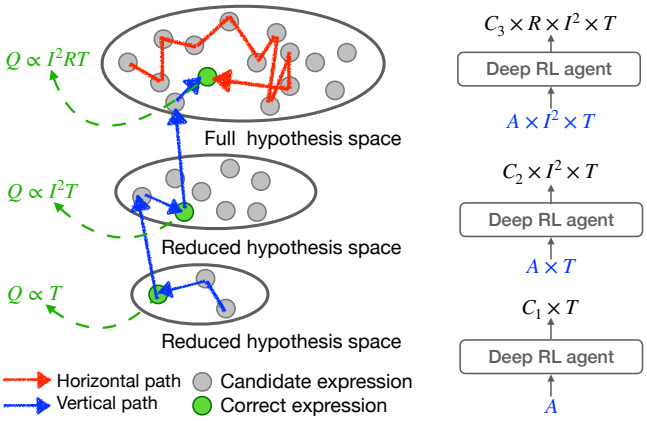

Figure 1: Our VSR-DPG follows a vertical path (colored blue) better than the horizontal path (colored red), in the scientific discovery of Joule's first law. **(Left)** The vertical discovery starts by finding the relationship between two factors (Q, T) in a reduced hypothesis space with other factors held constant. It then finds models in extended hypothesis space with three factors (Q, I, T), and finally in the full hypothesis space. Searching following the vertical paths is way cheaper since the sizes of the reduced hypothesis spaces in the first few steps are exponentially smaller than the full hypothesis space. **(Right)** Our VSR-DPG extends the equation in each step. The placeholder A indicates a sub-expression.

Recently, Vertical Symbolic Regression (VSR) [Jiang and Xue, 2023; Jiang *et al.*, 2023] has been proposed to expedite the discovery of symbolic equations with many independent variables. Unlike previous approaches, VSR reduces the search spaces following a vertical path – it extends from reduced-form equations involving a subset of independent variables to full-fledged ones, adding one variable into the equation at a time. Figure 1 gives an example discovery of Joule's law $Q \propto I^2RT$ [Joule, 1843], where Q is heat, I is current, R is resistance, and T is time. VSR first holds I and R as constants and finds $Q \propto T$. Next, I is introduced with targeted experiments that study the effect of I on Q. Such rounds repeat until all factors are considered. Compared with the horizontal paths, that modeling all independent variables simultaneously, vertical discovery can be significantly cheaper because the search spaces of the first few steps are exponentially smaller than the full search space.

Meanwhile, deep learning, especially deep policy gradient [Petersen *et al.*, 2021], has greatly boosted the performance of symbolic regression approaches. However, VSR was implemented using genetic programming. Although we hypothesize deep neural nets should boost VSR, directly integrating deep neural nets to predict the symbolic equation tree in each vertical expansion step encounters difficulties. This will result in (1) difficulty passing gradients from trees to deep neural nets and (2) complications concatenating deep networks for predictions in each vertical expansion step. We provide a detailed analysis of the difficulty in Appendix A.

In this work, we propose **V**ertical **S**ymbolic **R**egression using **D**eep **P**olicy **G**radient (VSR-DPG). We demonstrate that VSR-DPG can recover ground-truth equations involving 50 variables, which is beyond both deep reinforcement learning-based approaches and the previous VSR variant (i.e., VSR-GP). Our VSR-DPG solves the above difficulty based on the following key idea: each symbolic expression can be treated as repeated applications of grammar rules. Hence, discovering the best symbolic equations in the space of all candidate expressions is viewed as the sequential decision-making of predicting the optimal sequence of grammar rules.

In Figure 1(right), the expansion from C_1T to C_2I^2T is to replace constant C_1 with a sub-expression C_2I^2. We define a context-free grammar on symbolic expression, denoting the rules that certain constants can be replaced with other variables, constants, or sub-expressions. All candidate expressions that are compatible with C_1T can be generated by repetitively applying the defined grammar rules in different order. VSR-DPG generates many sub-expressions (including C_2I^2) by sequentially sampling rules from the Recurrent Neural Networks (RNN). A vertical discovery path is built on top of this sequential decision-making process of reduced-form symbolic expressions. The RNN is trained to generate expansions that lead to high fitness scores on the dataset. In this regard, we train the RNN to maximize a policy gradient objective, similar to that proposed in Petersen *et al.*. The main difference is the RNN in our VSR-DPG predicts the next rules in the vertical discovery space, while the model from Petersen *et al.* predicts the next math token in the expression tree in the horizontal discovery space.

In experiments, we consider several challenging multi-variable datasets of algebraic equations and of ordinary differential equations in material science and biology. (1) In Table 1, our VSR-DPG attains the smallest median NMSE values in 7 out of 8 datasets, against current popular baselines including VSR-GP. The main reason is deep networks offer more parameters than GP, which can better adapt to different datasets and sample higher-quality expressions from the networks. (2) Further analysis on the best-discovered equation (in Table 3) shows that VSR-DPG uncovers up to 50% of the exact governing equations with 5 input variables, where the baselines only attain 0%. (3) In Table 2, our VSR-DPG can find high-quality expressions on datasets with up to 50 variables, because of the vertical discovery idea. (4) On discovery of ordinary differential equations in Table 4, our VSR-DPG also improves over current baselines.[1]

[1]The code is at https://github.com/jiangnanhugo/VSR-DPG. Please refer to https://arxiv.org/abs/2402.00254 for the Appendix.

2 Preliminaries

Symbolic Regression aims to discover governing equations from the experimental data. An example of such mathematical expression is Joule's first law: $Q = I^2RT$, which quantifies the amount of heat Q generated when electric current I flows through a conductor with resistance R for time T. Formally, a mathematical expression ϕ connects a set of input variables $\mathbf{x}$ and a set of constant coefficients $\mathbf{c}$ by mathematical operators. The possible mathematical operators include addition, subtraction, multiplication, division, trigonometric functions, etc. The meaning of these mathematical expressions follows their standard arithmetic definition.

Given a dataset $D = \{(\mathbf{x}_i, y_i)_{i=1}^N | \mathbf{x}_i \in \mathbb{R}^n, y_i \in \mathbb{R}\}$ with N samples, symbolic regression searches for the optimal expression ϕ^*, such that $\phi^*(\mathbf{x}_i, \mathbf{c}) \approx y_i$. From an optimization perspective, ϕ^* minimizes the averaged loss on the dataset:

$$\phi^* \leftarrow \arg\min_{\phi \in \Pi} \frac{1}{N} \sum_{i=1}^{N} \ell(\phi(\mathbf{x}_i, \mathbf{c}), y_i). \tag{1}$$

Here Π is the set of all candidate mathematical expressions; $\mathbf{c}$ denotes the constant coefficients in the expression; $\ell(\cdot, \cdot)$ denotes a loss function that measures the difference between the output of the candidate expression $\phi(\mathbf{x}_i, \mathbf{c})$ and the ground truth y_i. The set of all possible expressions, i.e., the hypothesis space Π, can be exponentially large. As a result, finding the optimal expression is challenging and is shown to be NP-hard [Virgolin and Pissis, 2022].

Deep Policy Gradient for Symbolic Regression. Recently, a line of work proposes the use of deep reinforcement learning (RL) for searching the governing equations [Petersen *et al.*, 2021; Abolafia *et al.*, 2018; Mundhenk *et al.*, 2021]. They represent expressions as binary trees, where the interior nodes of the tree correspond to mathematical operators and leaf nodes of the tree correspond to variables or constants. The key idea is to model the search of different expressions, as a sequential decision-making process for the preorder traversal sequence of the expression trees, using an RL algorithm. A reward function is defined to measure how well a predicted expression can fit the dataset. The deep RNN is used as the RL agent for predicting the next possible node in the expression tree at every step of decision-making. The parameters of the RNN are trained using policy gradient.

Control Variable Experiment studies the relationship between a few input variables and the output in the regression problem, with the remaining input variables fixed to be the same [Lehman *et al.*, 2004]. In the controlled setting, the ground-truth equation behaves the same after setting those controlled variables as constants, which is noted as the *reduced-form equation*. For example, the ground-truth equation $\phi = x_1 \times x_3 - x_2 \times x_4$ in Figure 2(a) is reduced to $x_1 \times C_1 - C_2$ when controlling x_2, x_3, x_4. Figure 2(b,c) presents other reduced-form equations when the control variables are changed. For the corresponding dataset D, the controlled variables are fixed to one value and the remaining variables are randomly assigned. See Figure 2(a,b,c) for example datasets generated under different controlling variables.

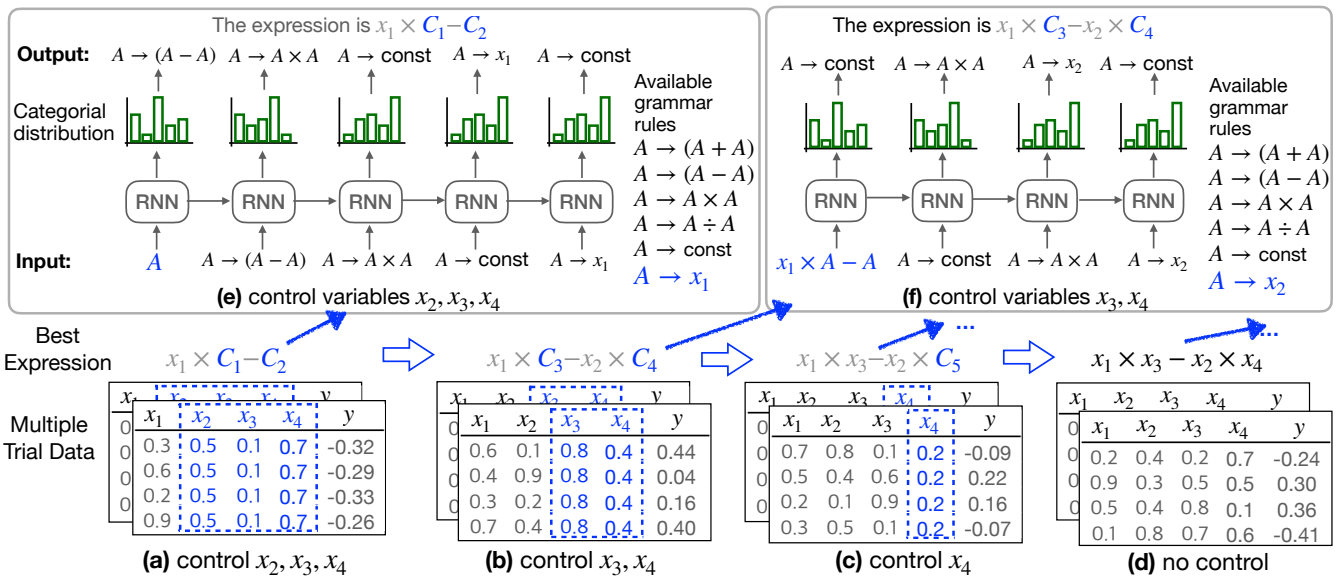

Figure 2: The proposed VSR-DPG for the discovery of expression $\phi = x_1 \times x_3 - x_2 \times x_4$. **(a)** Initially, a reduced-form equation $\phi = x_1 \times C_1 - C_2$ is found, in which variables x_2, x_3, x_4 are held constant and only variable x_1 is allowed to vary. C_1 and C_2 (colored blue) are summary constants, which are sub-expressions containing the controlled variables. The open constants in the expression are fitted by the corresponding controlled variable data. **(b)** In the second stage, this equation is expanded to $x_1 \times C_3 - x_2 \times C_4$. **(c, d)** This process continues until the ground-truth equation $\phi = x_1 x_3 - x_2 x_4$ is found. **(e, f)** Under those controlled variables, the deep recurrent neural network predicts a categorical distribution over the available grammar rules. The controlled variables are excited in the grammar rules. The best-predicted expression in (e) is reformulated as the start symbol in (f), that is $x_1 \times A - A$.

Vertical Symbolic Regression starts by finding a symbolic equation involving a small subset of the input variables and iteratively expands the discovered expression by introducing more variables. VSR relies on the control variable experiments introduced above. VSR-GP was the first implementation of vertical symbolic regression using genetic programming (GP) [Jiang and Xue, 2023]. To fit an expression of n variables, VSR-GP initially only allows variable x_1 to vary and controls the values of all the rest of the variables. Using GP as a subroutine, VSR-GP finds a pool of expressions $\{\phi_1, \ldots, \phi_m\}$ which best fit the data from this controlled experiment. Notice $\{\phi_1, \ldots, \phi_m\}$ are restricted to contain only one free variable x_1 and m is the pool size. A small error indicates ϕ_i is close to the ground truth reduced to the one free variable and thus is marked unmutable by the genetic operations in the following rounds. In the 2nd round, VSR-GP adds a second free variable x_2 and starts fitting $\{\phi'_1, \ldots, \phi'_m\}$ using the data from control variable experiments involving the two free variables x_1, x_2. After n rounds, the expressions in the VSR-GP pool consider all n variables. Overall, VSR expedites the discovery process because the first few rounds of VSR are significantly cheaper than the traditional *horizontal* discovery process, which searches for optimal expression involving all input variables.

3 Methodology

Motivation. The prior work of VSR-GP uses genetic programming to edit the expression tree. GP is not allowed to edit internal nodes in the best-discovered expression trees from the previous vertical discovery step, to ensure later genetic operations do not delete the prior knowledge on the governing equation. However, this idea cannot be easily integrated with deep RL-based symbolic regressors.

We explored using deep neural networks in vertical symbolic regression to predict the symbolic equation tree in each vertical expansion step, as detailed in Appendix A. However, we encountered two main issues: (1) difficulty in passing gradients from trees to deep neural nets. We need different constraints to embed into the output of RNN to ensure the whole model follows the vertical discovery idea. (2) complexity in concatenating deep networks for predictions at each vertical expansion step. The underlying issue is the expression tree representation.

Our solution is to adopt a new representation for expressions. We extend the context-free grammar definition for symbolic expressions, where a sequence of grammar rules uniquely corresponds to an expression. We view the prediction of symbolic expressions as a sequential decision-making process, selecting grammar rules step-by-step. The RNN predicts grammar rules instead of nodes in the expression tree. The best-discovered reduced-form equation is converted into the start symbol in the grammar, ensuring that the predicted expression from our RNN is always compatible with the prior knowledge of the governing equation. This reduces the hypothesis space and accelerates scientific discovery, as other non-reducible expressions are never sampled from the RNN.

Main Pipeline. Figure 1 shows our deep vertical symbolic regression (VSR-DPG) pipeline. The core idea is to construct increasingly complex symbolic expressions involving more input variables, based on control variable experiments with

fewer controlled variables.

To fit an expression of n variables, we first hold $n-1$ variables as constant and allow only one variable to vary. We aim to find the best expression ϕ_1, that fits the data in this controlled experiment. We use the RNN as the RL agent to search for the best possible expression, which is achieved by using the RNN to sample sequences of grammar rules defined for symbolic equations. Every sequence of rules is converted into an expression, where the constants in the expression are fitted with the dataset. The parameters of the RNN model will be trained through the policy gradient objective. The expression with the best fitness score is returned as the prediction of the RNN. A visualized process is in Figure 1(a, e).

Following the idea in VSR, the next step is to classify each constant as either a summary constant or a standalone constant. (1) A constant not relevant to any controlled variables is considered standalone and is preserved in subsequent rounds. (2) A constant that is a sub-expression involving controlled variables is labeled as a summary type and will be expanded in subsequent rounds. In our implementation, a constant with high variance in fitted values across multiple control variable experiments is classified as a summary type; otherwise, it is classified as standalone.

Assuming we find the correct reduced-from equation ϕ_1 after several learning epochs. To ensure VSR-DPG does not forget this discovered knowledge of the first round, we want all the expressions to be discovered in the following rounds can be reduced to ϕ_1. Therefore, we construct ϕ_1 as the start symbol for the following round by replacing every summary constant in ϕ_1 as a non-terminal symbol in the grammar (noted as A), indicating a sub-expression. For example, in Figure 2(a), both of them are summary constants so the 1st round best-predicted expression $x_1 \times C_1 - C_2$ is converted to the start symbol $x_1 \times A - A$ for the second round.

In the second round, VSR-DPG adds one more free variable and starts fitting ϕ_2 using the data from control variable experiments involving two free variables. Similar to the first round, we restrict our search to sub-expressions with the second variable. It is achieved by limiting the output vocabulary of the RNN model. In Figure 2(f), the RNN model finds an expression $\phi_2 = x_1 \times C_3 - (x_2 \times C_4)$. This means that the RL agent learns to expand C_1 with another constant C_3 and C_2 with sub-expression $(x_2 \times C_4)$, based on the best-discovered result of the 1st round $\phi_1 = x_1 \times C_1 - C_2$.

VSR-DPG introduces one free variable at a time and expands the equation learned in the previous round to include this newly added variable. This process continues until all the variables are considered. After n rounds, we return the equations with the best fitness scores. The predicted equation will be evaluated on data with no variable controlled. See the steps in Figure 1(b,c,d) for a visual demonstration. We summarize the whole process of VSR-DPG in Algorithm 1 in the appendix. The major difference of this approach from most state-of-the-art approaches is that those baselines learn to find the expressions in the full hypothesis space with all input variables, from a fixed dataset collected before training. Our VSR-DPG accelerates the discovery process, because of the small size of the reduced hypothesis space, i.e., the set of candidate expressions involving only a few variables. The task is much easier than fitting the expression in the full hypothesis space involving all input variables.

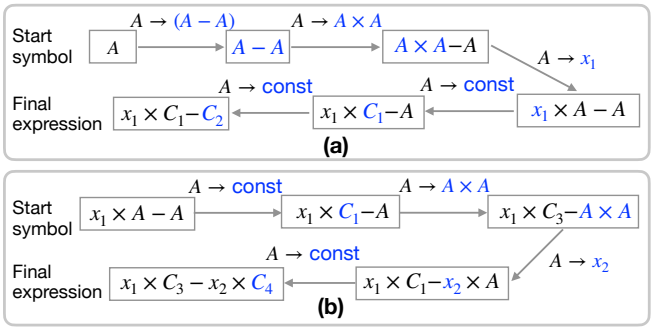

Figure 3: Convert a sequence of grammar rules into a valid expression. Each rule expands the first non-terminal symbol in the squared box. The parts that get expanded are color-highlighted.

3.1 Expression Represented as Grammar Rules

We propose to represent symbolic expression by extending the existing context-free grammar [Todorovski and Dzeroski, 1997]. A context-free grammar is represented by a tuple (V, Σ, R, S), where V is a set of non-terminal symbols, Σ is a set of terminal symbols, R is a set of production rules and S is a start symbol. In our formulation, (1) Σ is the set of input variables and constants $\{x_1, \ldots, x_n, \texttt{const}\}$. (2) The set of non-terminal symbols V represents sub-expressions, like $\{A\}$. Here A is a placeholder symbol. (3) The set of production rules R represents mathematical operations such as addition, subtraction, multiplication, and division. That is $\{A \to (A + A), A \to (A - A), A \to A \times A, A \to A \div A\}$, where $\to$ indicates that the left-hand side is replaced with the right-hand side. (4) The start symbol S is extended to be A, $x_1 \times A - A$, or other symbols constructed from the best-predicted expression under the controlled variables.

Starting with the start symbol, successively applying the grammar rules in different orders results in different expressions. Each rule expands the *first* non-terminal symbol in the start symbol. An expression with only terminal symbols is a valid mathematical expression, whereas an expression with a mixture of non-terminal and terminal symbols is invalid. The expression can also be represented as a binary tree. We chose the grammar representation over the binary tree representation because it simplifies the vertical symbolic regression process, avoiding gradient passing and heavy engineering issues.

Figure 3 presents two examples of constructing the expression ϕ from the start symbol using a given sequence of grammar rules. In Figure 3(a), we first use the subtraction rule $A \to (A - A)$, indicating that the symbol A in expression $\phi = A$ is expanded to $\phi = (A - A)$. By repeatedly using grammar rules to expand the non-terminal symbols, we eventually arrive at our desired expression $\phi = x_1 \times C_1 - C_2$. In Figure 3(b), the start symbol is $x_1 \times A - A$. The rule $A \to \texttt{const}$ replaces the first non-terminal symbol with a constant C_1, resulting in $x_1 \times C_1 - A$. Finally, we obtain a valid expression $x_1 \times C_1 - C_2$. Figure 3 provides another example conversion with the start symbol $x_1 \times A - A$.

3.2 Expression Sampling from Recurrent Network

In our vertical symbolic regression setting, the input and output are in the set of grammar rules that cover each input variable, constants, and math operations. We create an embedding layer for the input, noted as $\texttt{Embd}$. For each input rule $r \in R$, its d-dimensional embedding vector is $\texttt{Embd}(r) \in \mathbb{R}^d$.

Sampling Procedure. The RNN module samples an expression by sampling the sequence of grammar rules in a sequential decision-making process. Denote the sampled sequence of rules as $\tau = (\tau_1, \tau_2, \ldots)$. Initially, the RNN takes in the start symbol $\tau_1 = S$ and computes the first step hidden state vector $\mathbf{h}_1$. At t-th time step, RNN uses the predicted output from the previous step as the input of current step τ_t. RNN computes its hidden state vector $\mathbf{h}_t$ using the embedding vector of input token τ_t and the previous time-step hidden state vector $\mathbf{h}_{t-1}$. The linear layer and softmax function are applied to emit a categorical distribution $p(\tau_{t+1}|\tau_t, \mathbf{h}_t)$ over every token in the output vocabulary, which represents the probability of the next possible rule in the half-completed expression $p_\theta(\tau_{t+1}|\tau_t, \ldots, \tau_1)$. The RNN samples one token from the categorical distribution $\tau_{t+1} \sim p(\tau_{t+1}|\tau_t, \mathbf{h}_t)$ as the prediction of the next possible rule. To conclude, the computational pipeline at the t-th step is shown below:

$$\begin{aligned} \boldsymbol{\tau}_t &= \texttt{Embd}(\tau_t), \\ \mathbf{h}_t &= \texttt{RNN}(\boldsymbol{\tau}_t, \mathbf{h}_{t-1}), \\ \mathbf{s}_t &= W\mathbf{h}_t + \mathbf{b}, \\ p(\tau_{t+1} = r_i|\tau_t, \mathbf{h}_t) &= \frac{\exp(\mathbf{s}_{t,i})}{\sum_{r_j \in R} \exp(\mathbf{s}_{t,j})}, \quad \text{for } r_i \in R \end{aligned} \tag{2}$$

The weight matrix $W \in \mathbb{R}^{d \times |R|}$ and bias vector $\mathbf{b} \in \mathbb{R}^d$ are the parameters of the linear layer. The last row in Equation 2 is the softmax layer. The sampled rule r_{t+1} will be the input for the $t+1$-th step. We denote θ as the total parameters of the embedding layer, the RNN, and the linear layer.

After L steps, we obtain the sequence $\tau = (\tau_1, \ldots, \tau_L)$ with probability $p_\theta(\tau) = \prod_{t=1}^{L-1} p_\theta(\tau_{t+1}|\tau_1, \ldots, \tau_t)$. We convert this sequence into an expression by following the procedure described in Section 3.1. If we reach the end of the sequence while there are still non-terminal symbols in the converted expression, we randomly add rules containing only terminal symbols to complete the expression. Conversely, if we obtain a valid expression before the sequence ends, we disregard the remaining elements of the sequence and return the valid expression.

Policy Gradient-based Training. We follow the reinforcement learning formulation to train the parameters of the RNN module [Wierstra *et al.*, 2010]. The sampled rules before the current step t, i.e., $(\tau_1, \ldots, \tau_t)$, is viewed as the *state* of the t-th step for the RL agent. Those rules in the output vocabulary are the available *actions* for the RL agent. In the formulated decision-making process, the RNN takes in the current state and outputs a distribution over next-step possible actions. The objective of the RL agent is to learn to pick the optimal sequences of grammar rules to maximize the expected rewards. Denote the converted expression from τ as ϕ. A typical reward function is defined from the fitness score of the expression $\texttt{reward}(\tau) = 1/(1+\texttt{NMSE}(\phi))$. The objective that maximizes the expected reward from the RNN model is defined as $J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)}(\texttt{reward}(\tau))$, where $p_\theta(\tau)$ is the probability of sampling sequence τ from the RNN.

In the next step, the gradient with respect to the objective $\nabla_\theta J(\theta)$ needs to be estimated. We follow the classic REINFORCE policy gradient algorithm [Williams, 1992]. We first sample several times from the RNN module and obtain N sequences $(\tau^1, \ldots, \tau^N)$, an unbiased estimation of the gradient of the objective is computed as $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \texttt{reward}(\tau^i) \nabla_\theta \log p_\theta(\tau^i)$. The parameters of the deep network are updated by the gradient descent algorithm with the estimated policy gradient value. In the literature, several practical tricks are proposed to reduce the estimation variance of the policy gradient. A common choice is to subtract a baseline function b from the reward, as long as the baseline is not a function of the sample batch of expressions. Our implementation adopts this trick and the detailed derivation is presented in Appendix B.

Throughout the whole training process, the expression with optimal fitness score from all the sampled expressions is used as the prediction of VSR-DPG at the current round.

Start Symbol Construction in Vertical Discovery. Given the best-predicted equation ϕ and controlled variables $\mathbf{x}_c$, the following step is to construct the start symbol of the next rounds. This operation ensures all the future expressions can be reduced to any previously discovered equation thus all the discovered knowledge is remembered. It expedites the discovery of symbolic expression since other expressions that cannot be reduced to ϕ will never be sampled from the RNN. It requires first classifying the type of every constant in the expression into stand-alone or summary type, through multi-trail control variable experiments. Then we replace each summary constant with a placeholder symbol (i.e., "A") indicating a sub-expression containing controlled variables.

Following the procedure proposed in [Jiang and Xue, 2023], we first query K data batches $(D_1, \ldots, D_K)$ with the same controlled variables $\mathbf{x}_c$. The controlled variables take the same value within each batch while taking different values across data batches. We fit open constants in the candidate expression ϕ with each data batch by the gradient-based optimizer, like BFGS [Fletcher, 2000]. We obtain multiple fitness scores $(o_1, \ldots, o_K)$ and multiple solutions to open constants $(\mathbf{c}_1, \ldots, \mathbf{c}_K)$. By examining the outcomes of K-trials control variable experiments, we have: (1) Consistent close-to-zero fitness scores imply the fitted expression is close to the ground-truth equation in the reduced form. That is $o_k \leq \varepsilon$ for all $1 \leq k \leq K$, where ε is the threshold for the fitness scores. (2) Conditioning on the result in case (1), the j-th open constant is a standalone constant when the empirical variance of its fitted values across K trials is less than a threshold ε'. In practice, if the best-predicted expression by the RNN module is not consistently close to zero, then all the constants in the expression are summary constants. Finally, the start symbol is obtained by replacing every summary constant with the symbol "A" according to our grammar.

Methods	$(2,1,1)$	$(3,2,2)$	$(4,4,6)$	$(5,5,5)$	$(5,5,8)$	$(6,6,8)$	$(6,6,10)$	$(8,8,12)$
VSR-GP	0.005	0.028	0.086	0.014	0.066	0.066	**0.104**	T.O.
GP	$7E{-}4$	0.023	0.044	0.063	0.102	0.127	0.159	0.872
Eureqa	**<1E-6**	**<1E-6**	0.024	0.158	0.284	0.433	0.910	0.162
SPL	0.006	0.033	0.144	0.147	0.307	0.391	0.472	0.599
E2ETransformer	0.018	0.0015	0.030	0.121	0.072	0.194	0.142	0.112
DSR	**< 1E-6**	0.008	2.815	2.558	2.535	0.936	6.121	0.335
PQT	0.020	0.161	2.381	2.168	2.482	0.983	5.750	0.232
VPG	0.030	0.277	2.990	1.903	2.440	0.900	3.857	0.451
GPMeld	$< 1E{-}6$	0.112	1.670	1.501	2.422	0.964	7.393	T.O.
VSR-DPG (ours)	**< 1E-6**	**< 1E-6**	**< 1E-6**	**< 1E-6**	**0.026**	**0.063**	0.114	**0.101**

Table 1: On selected algebraic equation datasets, median (50%-quartile) of NMSE values of the best-predicted expressions found by all the algorithms. The set of mathematical operator is $O_p = \{+, -, \times, \sin, \cos, \texttt{const}\}$. The 3-tuples at the top $(\cdot, \cdot, \cdot)$ indicate the number of free variables, singular terms, and cross terms in the ground-truth expressions generating the dataset. O_p stands for the set of allowed operators. "T.O." implies the algorithm is timed out for 48 hours.

4 Related Work

Recently AI has been highlighted to enable scientific discoveries in diverse domains [Kirkpatrick *et al.*, 2021; Jumper *et al.*, 2021; Wang *et al.*, 2023]. Early work in this domain focuses on learning logic (symbolic) representations [Bradley *et al.*, 2001]. Recently, there has been extensive research on learning algebraic equations from data [Petersen *et al.*, 2021; Mundhenk *et al.*, 2021] and learning differential equations from data [Dzeroski and Todorovski, 1995; Brunton *et al.*, 2016; Wu and Tegmark, 2019; Zhang and Lin, 2018; Iten *et al.*, 2020; Cranmer *et al.*, 2020; Raissi *et al.*, 2020; Raissi *et al.*, 2019; Liu and Tegmark, 2021; Xue *et al.*, 2021; Chen *et al.*, 2018]. In this domain, a line of works develops robots that automatically refine the hypothesis space, some with human interactions [Langley *et al.*, 1987; Valdés-Pérez, 1994; King *et al.*, 2004; King *et al.*, 2009]. These works are related to ours because they also actively explore the hypothesis spaces, while they are in biology and chemistry.

Existing works on multivariate regression predominantly rely on pre-trained encoder-decoder structures with extensive training datasets [Biggio *et al.*, 2021], and larger-scale deep models [Kamienny *et al.*, 2022]. Our VSR-DPG method is tailored to solve multivariate symbolic regression.

The idea of using a control variable experiment tightly connects to the BACON system [Langley, 1977; Langley, 1979; Langley *et al.*, 1981; King *et al.*, 2004; King *et al.*, 2009; Cerrato *et al.*, 2023]. While their methods are rule-based systems due to historical limitations, our approach leverages contemporary deep recurrent neural networks.

Our method is also related to symbolic regression methods using probabilistic context-free grammar [Todorovski and Dzeroski, 1997; Brence *et al.*, 2021; Gec *et al.*, 2022]. While they use a fixed probability to sample rules, we use a deep neural network to learn the probability distribution.

Our VSR-DPG is also tightly connected to deep symbolic regression [Petersen *et al.*, 2021; Mundhenk *et al.*, 2021]. We both use deep recurrent networks to predict a sequence of tokens that can be composed into a symbolic expression. However, their method predicts the preorder traversal sequence for the expression tree while our method predicts the sequence of production rules for the expression.

5 Experiment

5.1 Regression on Algebraic Equations

Experiment Settings. For the dataset on algebraic expressions, we consider the 8 groups of expressions from the Trigonometric dataset [Jiang and Xue, 2023], where each group contains 10 randomly sampled expressions. In terms of baselines, we consider (1) evolutionary algorithms: Genetic Programming (GP), Eureqa [Dubcáková, 2011] and Control Variable Genetic Programming (VSR-GP) [Jiang and Xue, 2023]. (2) deep reinforcement learning: Priority queue training (PQT) [Abolafia *et al.*, 2018], Vanilla Policy Gradient (VPG) [Williams, 1992], Deep Symbolic Regression (DSR) [Petersen *et al.*, 2021], and Neural-Guided Genetic Programming Population Seeding (GPMeld) [Mundhenk *et al.*, 2021]. (3) Monte Carlo Tree Search: Symbolic Physics Learner (SPL) [Sun *et al.*, 2023], (4) pretrained Transformer: end-to-end Transformer (E2ETransformer) [Kamienny *et al.*, 2022]. In terms of evaluation metrics, we use the normalized mean-squared-error (NMSE) of the best-predicted expression by each algorithm, on a separately-generated testing dataset. We report the median instead of the mean value. Symbolic regression belongs to combinatorial optimization problems, which have no mean values and are sensitive to outliers. The detailed settings are in Appendix C.2.

Goodness-of-fit Comparison. We consider our VSR-DPG against several challenging datasets involving multiple variables. In Table 1, we report the median NMSE on the selected algebraic datasets. Our VSR-DPG attains the smallest median NMSE values in 7 out of 8 datasets, against a line of current popular baselines including the original VSR-GP. The main reason is deep networks offer many more parameters than the GP algorithm, which can better adapt to different datasets and sample higher-quality expressions from the deep networks.

Extended Large-scale Comparison. In the real world, scientists may collect all available variables that are more than needed into symbolic regression, where only part of the inputs will be included in the ground-truth expression. We randomly pick 5 variables from all the n variables and replace the appeared variable in expressions confined as $(5, 5, 5)$ in Table 1. In Table 2, we collect the median NMSE values on

	Total input variables n in the data				
	$n = 10$	$n = 20$	$n = 30$	$n = 40$	$n = 50$
SPL	0.386	0.554	0.554	0.714	0.815
GP	0.159	0.172	0.218	0.229	0.517
DSR	0.284	0.521	0.522	0.660	0.719
VPG	0.415	0.695	0.726	0.726	0.779
PQT	0.384	0.488	0.615	0.620	0.594
VSR-DPG	**< 1E-6**	**< 1E-6**	**< 1E-6**	**0.002**	**0.021**

Table 2: On large-scale algebraic equation dataset, with reported Median NMSE values, our VSR-DPG scales better to more variable settings than baselines due to the control variable experiment.

	$(2, 1, 1)$	$(3, 2, 2)$	$(4, 4, 6)$	$(5, 5, 5)$
SPL	20%	10%	0%	0%
E2ETransformer	0%	0%	0%	0%
VSR-GP	60%	50%	0%	0%
VSR-DPG	**100%**	**70%**	**60%**	**40%**

Table 3: On selected algebraic equations, the exact recovery rate over the best-predicted found by all the algorithms. Our VSR-DPG has a higher rate of recovering the ground-truth expressions compared to baselines.

this large-scale dataset setting. Our VSR-DPG scales well because it first detects all the contributing inputs using the control variable experiments. Notice that those baselines that are easily timeout in this setting are excluded for comparison.

Exact Recovery Comparison. We compare if each learning algorithm finds the exact equation, the result of which is collected in Table 3. The discovered equation by each algorithm is further collected in Appendix D.1. We can observe that our VSR-DPG has a higher rate of recovering the ground-truth expressions compared to baselines. This is because our method first use a control variable experiment to pick what are the contributing variables to the data and what are not.

5.2 Regression on Ordinary Differential Equations

Task Definition. The temporal evolution of the dynamic system is modeled by the time derivatives of the state variables. Let $\mathbf{x}$ be the n-dimensional vector of state variables, and $\dot{\mathbf{x}}$ is the vector of their time derivatives. The ordinary differential equation (ODE) is of the form $\dot{\mathbf{x}} = \phi(\mathbf{x}, \mathbf{c})$, where constant vector $\mathbf{c} \in \mathbb{R}^m$ are parameters of the dynamic system. Following the definition of symbolic regression on ODE [Gec *et al.*, 2022; Sun *et al.*, 2023], given a trajectory dataset of state variable and its time derivatives $\{(\mathbf{x}(t_i), \dot{\mathbf{x}}(t_i))\}_{i=1}^N$, the symbolic regression task is to predict the best expression $\phi(\mathbf{x}, \mathbf{c})$ that minimizes the average loss on trajectory data. Other formulations of this problem [d'Ascoli *et al.*, 2024] assume we have no access to its time derivatives, i.e., $\{(t_i, \mathbf{x}(t_i))\}_{i=1}^N$.

Experiment Setting. We consider recent popular baselines for differential equations, including (1) SINDy [Brunton *et al.*, 2016], (2) ODEFormer [d'Ascoli *et al.*, 2024], (3) Symbolic Physics Learner (SPL) [Sun *et al.*, 2023]. (4) Probabilistic grammar for equation discovery (ProGED) [Brence *et al.*, 2021]. In terms of the dataset, we consider the Lorenz Attractor with $n = 3$ variables, Magnetohydrodynamic (MHD) turbulence with $n = 6$ variables, and Glycolysis Oscillation with $n = 7$ variables. All of them are collected from [Brunton *et al.*, 2016]. To evaluate whether the algorithm identifies the ground-truth expression, we use the Accuracy metric based on the coefficient of determination (R^2). The detailed experiment configurations are in Appendix C.4.

	Lorenz	MHD	Glycolysis
SPL	**100%**	50%	14.2%
SINDy	**100%**	0%	0%
ProGED	0%	0%	0%
ODEFormer	0%	0%	NA
VSR-DPG (ours)	**100%**	**100%**	**87%**

Table 4: On the differential equation dataset, ($R^2 \geq 0.9999$)-based accuracy is reported over the best-predicted expression found by all the algorithms. Our VSR-DPG method can discover the governing expressions with a much higher accuracy rate than baselines.

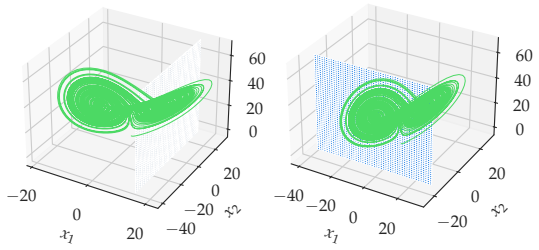

Figure 4: Visualization of VSR-DPG controlling variables x_1 (Left) and x_2 (Right) for the Lorenz attractor. The data of our VSR-DPG are drawn from the intersection of the mesh plane and the curve on the Lorenz attractor. In comparison, the ODEFormer draws data by picking a consecutive sequence $\{(t_i, \mathbf{x}(t_i))\}_{t=0}^N$ without knowing its time derivative on the curve.

Result Analysis. The results are summarized in Table 4. Our proposed VSR-DPG discovers a set of differential expressions with much higher quality than the considered baselines. We further provide a visual understanding of the proposed VSR-DPG method in Figure 4. The data of our VSR-DPG are drawn from the intersection of the mesh plane and the curve on the Lorenz attractor. In comparison, the current baselines draw data by picking a random trajectory or many random points on the curve. We notice the ODEFormer is pre-trained on differential equations up to two variables, and thus does not scale well with more variable settings. The predicted differential equations by each algorithm are in Appendix D.2.

6 Conclusion

In this research, we propose VSR-DPG to accelerate the discovery of governing equations involving many variables, which is beyond the capabilities of current state-of-the-art approaches. VSR-DPG follows a vertical discovery path, building equations involving more and more input variables using control variable experiments. VSR-DPG has the potential to supercharge current popular approaches because the first few steps following the vertical discovery route are much cheaper than discovering the equation in the full hypothesis space. Experimental results show VSR-DPG can uncover complex equations with more variables than what current approaches can handle.

Acknowledgments

We thank all the reviewers for their constructive comments. This research was supported by NSF grant CCF-1918327 and DOE – Fusion Energy Science grant: DE-SC0024583.

References

[Abolafia *et al.*, 2018] Daniel A. Abolafia, Mohammad Norouzi, and Quoc V. Le. Neural program synthesis with priority queue training. *CoRR*, abs/1801.03526, 2018.

[Biggio *et al.*, 2021] Luca Biggio, Tommaso Bendinelli, Alexander Neitz, Aurélien Lucchi, and Giambattista Parascandolo. Neural symbolic regression that scales. In *ICML*, volume 139, pages 936–945. PMLR, 2021.

[Bradley *et al.*, 2001] Elizabeth Bradley, Matthew Easley, and Reinhard Stolle. Reasoning about nonlinear system identification. *Artif. Intell.*, 133(1):139–188, 2001.

[Brence *et al.*, 2021] Jure Brence, Ljupco Todorovski, and Saso Dzeroski. Probabilistic grammars for equation discovery. *Knowl. Based Syst.*, 224:107077, 2021.

[Brunton *et al.*, 2016] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *PNAS*, 113(15):3932–3937, 2016.

[Cerrato *et al.*, 2023] Mattia Cerrato, Jannis Brugger, Nicolas Schmitt, and Stefan Kramer. Reinforcement learning for automated scientific discovery. In *AAAI Spring Symposium*, 2023.

[Chen *et al.*, 2018] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *NeurIPS*, 2018.

[Cranmer *et al.*, 2020] Miles D. Cranmer, Alvaro Sanchez-Gonzalez, Peter W. Battaglia, Rui Xu, Kyle Cranmer, David N. Spergel, and Shirley Ho. Discovering symbolic models from deep learning with inductive biases. In *NeurIPS*, 2020.

[d'Ascoli *et al.*, 2024] Stéphane d'Ascoli, Sören Becker, Alexander Mathis, Philippe Schwaller, and Niki Kilbertus. Odeformer: Symbolic regression of dynamical systems with transformers. In *ICLR*. OpenReview.net, 2024.

[Dubcáková, 2011] Renáta Dubcáková. Eureqa: software review. *Genet. Program. Evolvable Mach.*, 12(2), 2011.

[Dzeroski and Todorovski, 1995] Saso Dzeroski and Ljupco Todorovski. Discovering dynamics: From inductive logic programming to machine discovery. *J. Intell. Inf. Syst.*, 4(1):89–108, 1995.

[Fletcher, 2000] Roger Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2000.

[Gec *et al.*, 2022] Bostjan Gec, Nina Omejc, Jure Brence, Saso Dzeroski, and Ljupco Todorovski. Discovery of differential equations using probabilistic grammars. In *DS*, volume 13601, pages 22–31. Springer, 2022.

[Iten *et al.*, 2020] Raban Iten, Tony Metger, Henrik Wilming, Lídia Del Rio, and Renato Renner. Discovering physical concepts with neural networks. *Phys. Rev. Lett.*, 124(1):010508, 2020.

[Jiang and Xue, 2023] Nan Jiang and Yexiang Xue. Symbolic regression via control variable genetic programming. In *ECML/PKDD*, pages 178–195. Springer, 2023.

[Jiang *et al.*, 2023] Nan Jiang, Md Nasim, and Yexiang Xue. Vertical symbolic regression. *arXiv:2312.11955*, 2023.

[Joule, 1843] James Prescott Joule. On the production of heat by voltaic electricity. In *Abstracts of the Papers Printed in the Philosophical Transactions of the Royal Society of London*, pages 280–282, 1843.

[Jumper *et al.*, 2021] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[Kamienny *et al.*, 2022] Pierre-Alexandre Kamienny, Stéphane d'Ascoli, Guillaume Lample, and François Charton. End-to-end symbolic regression with transformers. In *NeurIPS*, 2022.

[Kamienny *et al.*, 2023] Pierre-Alexandre Kamienny, Guillaume Lample, Sylvain Lamprier, and Marco Virgolin. Deep generative symbolic regression with monte-carlo-tree-search. In *ICML*, volume 202. PMLR, 2023.

[King *et al.*, 2004] Ross D King, Kenneth E Whelan, Ffion M Jones, Philip GK Reiser, Christopher H Bryant, Stephen H Muggleton, Douglas B Kell, and Stephen G Oliver. Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature*, 427(6971):247–252, 2004.

[King *et al.*, 2009] Ross D. King, Jem Rowland, Stephen G. Oliver, Michael Young, Wayne Aubrey, Emma Byrne, Maria Liakata, Magdalena Markham, Pinar Pir, Larisa N. Soldatova, Andrew Sparkes, Kenneth E. Whelan, and Amanda Clare. The automation of science. *Science*, 324(5923):85–89, 2009.

[Kirkpatrick *et al.*, 2021] James Kirkpatrick, Brendan McMorrow, David H. P. Turban, Alexander L. Gaunt, James S. Spencer, Alexander G. D. G. Matthews, Annette Obika, Louis Thiry, Meire Fortunato, David Pfau, et al. Pushing the frontiers of density functionals by solving the fractional electron problem. *Science*, 374(6573):1385–1389, 2021.

[Kulkarni and Simon, 1988] Deepak Kulkarni and Herbert A Simon. The processes of scientific discovery: The strategy of experimentation. *Cogn. Sci.*, 12(2):139–175, 1988.

[Langley *et al.*, 1981] Pat Langley, Gary L. Bradshaw, and Herbert A. Simon. BACON.5: the discovery of conservation laws. In *IJCAI*, pages 121–126, 1981.

[Langley *et al.*, 1987] Patrick W. Langley, Herbert A. Simon, Gary Bradshaw, and Jan M. Zytkow. *Scientific Discovery: Computational Explorations of the Creative Process*. The MIT Press, 02 1987.

[Langley, 1977] Pat Langley. BACON: A production system that discovers empirical laws. In *IJCAI*, page 344, 1977.

[Langley, 1979] Pat Langley. Rediscovering physics with BACON.3. In *IJCAI*, pages 505–507, 1979.

[Lehman *et al.*, 2004] Jeffrey S Lehman, Thomas J Santner, and William I Notz. Designing computer experiments to determine robust control variables. *Statistica Sinica*, pages 571–590, 2004.

[Lenat, 1977] Douglas B. Lenat. The ubiquity of discovery. *Artif. Intell.*, 9(3):257–285, 1977.

[Liu and Tegmark, 2021] Ziming Liu and Max Tegmark. Machine learning conservation laws from trajectories. *Phys. Rev. Lett.*, 126:180604, May 2021.

[Mundhenk *et al.*, 2021] T. Nathan Mundhenk, Mikel Landajuela, Ruben Glatt, Cláudio P. Santiago, Daniel M. Faissol, and Brenden K. Petersen. Symbolic regression via deep reinforcement learning enhanced genetic programming seeding. In *NeurIPS*, pages 24912–24923, 2021.

[Petersen *et al.*, 2021] Brenden K. Petersen, Mikel Landajuela, T. Nathan Mundhenk, Cláudio Prata Santiago, Sookyung Kim, and Joanne Taery Kim. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. In *ICLR*, 2021.

[Raissi *et al.*, 2019] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[Raissi *et al.*, 2020] Maziar Raissi, Alireza Yazdani, and George Em Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367(6481):1026–1030, 2020.

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

[Sun *et al.*, 2023] Fangzheng Sun, Yang Liu, Jian-Xun Wang, and Hao Sun. Symbolic physics learner: Discovering governing equations via monte carlo tree search. In *ICLR*, 2023.

[Todorovski and Dzeroski, 1997] Ljupco Todorovski and Saso Dzeroski. Declarative bias in equation discovery. In *ICML*, pages 376–384. Morgan Kaufmann, 1997.

[Valdés-Pérez, 1994] R.E. Valdés-Pérez. Human/computer interactive elucidation of reaction mechanisms: application to catalyzed hydrogenolysis of ethane. *Catalysis Letters*, 28:79–87, 1994.

[Virgolin and Pissis, 2022] Marco Virgolin and Solon P Pissis. Symbolic regression is NP-hard. *TMLR*, 2022.

[Virgolin *et al.*, 2019] Marco Virgolin, Tanja Alderliesten, and Peter A. N. Bosman. Linear scaling with and within semantic backpropagation-based genetic programming for symbolic regression. In *GECCO*, pages 1084–1092, 2019.

[Wang *et al.*, 2023] Hanchen Wang, Tianfan Fu, Yuanqi Du, Wenhao Gao, Kexin Huang, Ziming Liu, Payal Chandak, Shengchao Liu, Peter Van Katwyk, Andreea Deac, et al. Scientific discovery in the age of artificial intelligence. *Nature*, 620(7972):47–60, 2023.

[Wierstra *et al.*, 2010] Daan Wierstra, Alexander Förster, Jan Peters, and Jürgen Schmidhuber. Recurrent policy gradients. *Log. J. IGPL*, 18(5):620–634, 2010.

[Williams, 1992] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8:229–256, 1992.

[Wu and Tegmark, 2019] Tailin Wu and Max Tegmark. Toward an artificial intelligence physicist for unsupervised learning. *Phys. Rev. E*, 100:033311, Sep 2019.

[Xue *et al.*, 2021] Yexiang Xue, Md. Nasim, Maosen Zhang, Cuncai Fan, Xinghang Zhang, and Anter El-Azab. Physics knowledge discovery via neural differential equation embedding. In *ECML/PKDD*, pages 118–134, 2021.

[Zhang and Lin, 2018] Sheng Zhang and Guang Lin. Robust data-driven discovery of governing physical laws with error bars. *Proc Math Phys Eng Sci.*, 474(2217), 2018.