

RECENT ADVANCEMENTS IN SYMBOLIC REGRESSION AT TOP ML CONFERENCES (ICML, NEURIPS, ICLR)

HENGZHE ZHANG

VICTORIA UNIVERSITY OF WELLINGTON

06/12/2024

TABLE OF CONTENTS

CENTRE FOR DATA SCIENCE
AND ARTIFICIAL INTELLIGENCE
TE WHIRI KAPE



1 Introduction

2 Reinforcement Learning-Based Symbolic Regression

3 Language Model-Based Symbolic Regression

4 Sparse Learning-Based Symbolic Regression

INTRODUCTION

Machine Learning for Symbolic Regression:

- Symbolic regression has garnered significant attention in recent years within the field of machine learning.
- This work focuses on leveraging neural networks to address symbolic regression problems, a topic widely discussed at top-tier ML conferences.

Categories of Deep Symbolic Regression Methods:

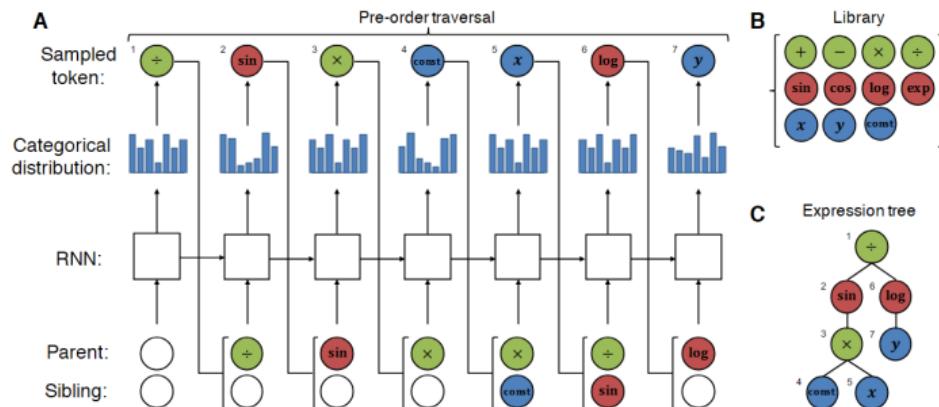
- Reinforcement Learning-Based Symbolic Regression
- Language Model-Based Symbolic Regression
- Sparse Learning-Based Symbolic Regression

REINFORCEMENT LEARNING-BASED SYMBOLIC REGRESSION

Motivation

Proposed Solution:

- Utilize a recurrent neural network (RNN) to generate mathematical expressions.
- Train using a *risk-seeking policy gradient* to optimize best-case performance.



RISK-SEEKING POLICY GRADIENT vs. STANDARD POLICY GRADIENT

Standard Policy Gradient (SPG):

- Objective:

$$J_{\text{std}}(\theta) = \mathbb{E}_{\tau \sim p(\tau|\theta)} [R(\tau)]$$

- Gradient:

$$\nabla_{\theta} J_{\text{std}}(\theta) = \mathbb{E}_{\tau \sim p(\tau|\theta)} [R(\tau) \nabla_{\theta} \log p(\tau|\theta)]$$

- Focuses on **average performance**.

Risk-Seeking Policy Gradient (RSPG):

- Objective:

$$J_{\text{risk}}(\theta; \epsilon) = \mathbb{E}_{\tau \sim p(\tau|\theta)} [R(\tau) \mid R(\tau) \geq R_{\epsilon}(\theta)]$$

- Gradient:

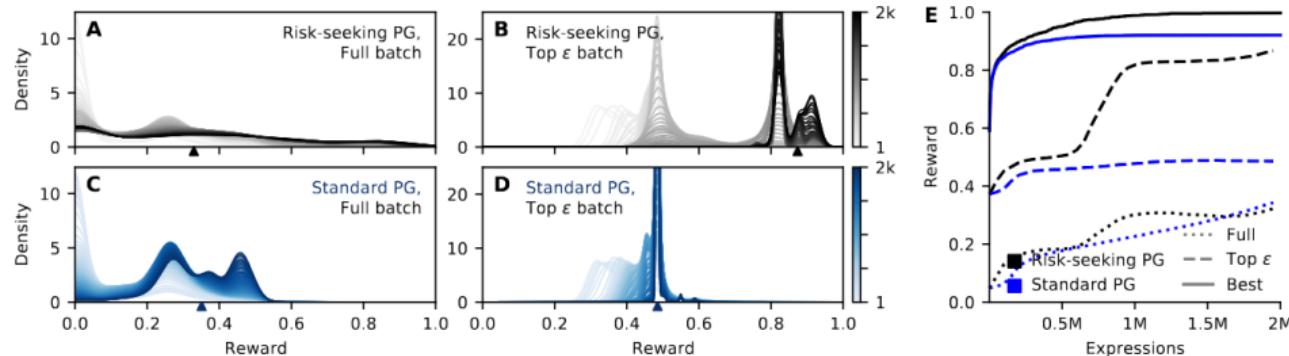
$$\nabla_{\theta} J_{\text{risk}}(\theta; \epsilon) = \mathbb{E}_{\tau \sim p(\tau|\theta)} [(R(\tau) - R_{\epsilon}(\theta)) \nabla_{\theta} \log p(\tau|\theta) \mid R(\tau) \geq R_{\epsilon}(\theta)]$$

- Focuses on **best-case performance (top- ϵ rewards)**.

Key Differences:

- RSPG modifies the gradient to emphasize high-reward trajectories, ignoring lower rewards.

RISK-SEEKING VS. STANDARD POLICY GRADIENT



■ A and B: Risk-Seeking Policy Gradient (RSPG)

- ▶ Focuses on top- ϵ rewards, prioritizing best-case outcomes.
- ▶ Shifts the distribution toward higher rewards over time.

■ C and D: Standard Policy Gradient (SPG)

- ▶ Optimizes for mean performance.
- ▶ Peaks early but lacks focus on top-reward trajectories.

■ E: Reward Comparison

- ▶ RSPG achieves higher best-case and top- ϵ rewards.
- ▶ SPG demonstrates better average reward (full batch).

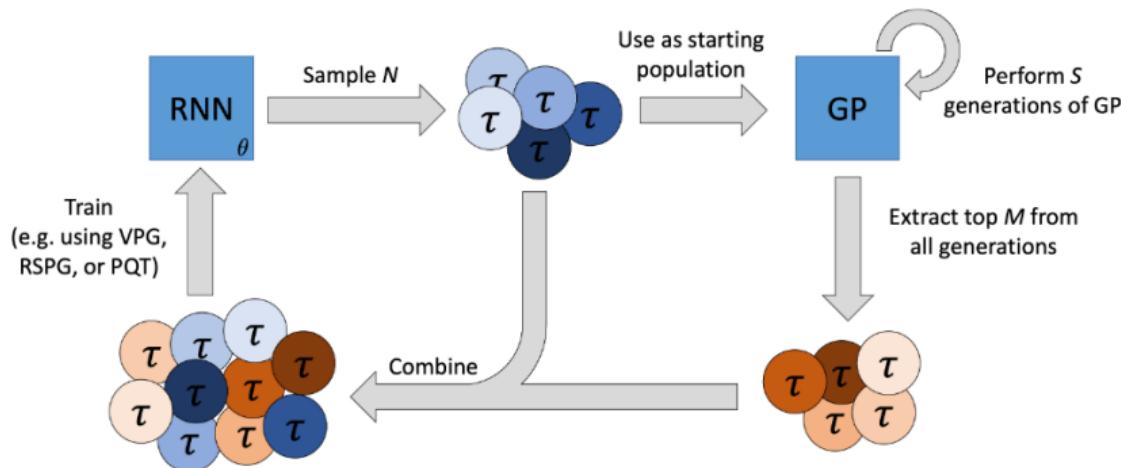
DEEP SYMBOLIC REGRESSION

Benchmark	Expression	DSR	PQT	VPG	GP	Eureqa	Wolfram
Nguyen-1	$x^3 + x^2 + x$	100%	100%	96%	100%	100%	100%
Nguyen-2	$x^4 + x^3 + x^2 + x$	100%	99%	47%	97%	100%	100%
Nguyen-3	$x^5 + x^4 + x^3 + x^2 + x$	100%	86%	4%	100%	95%	100%
Nguyen-4	$x^6 + x^5 + x^4 + x^3 + x^2 + x$	100%	93%	1%	100%	70%	100%
Nguyen-5	$\sin(x^2) \cos(x) - 1$	72%	73%	5%	45%	73%	2%
Nguyen-6	$\sin(x) + \sin(x + x^2)$	100%	98%	100%	91%	100%	1%
Nguyen-7	$\log(x + 1) + \log(x^2 + 1)$	35%	41%	3%	0%	85%	0%
Nguyen-8	\sqrt{x}	96%	21%	5%	5%	0%	71%
Nguyen-9	$\sin(x) + \sin(y^2)$	100%	100%	100%	100%	100%	-
Nguyen-10	$2 \sin(x) \cos(y)$	100%	91%	99%	76%	64%	-
Nguyen-11	x^y	100%	100%	100%	7%	100%	-
Nguyen-12	$x^4 - x^3 + \frac{1}{2}y^2 - y$	0%	0%	0%	0%	0%	-
Average		83.6%	75.2%	46.7%	60.1%	73.9%	-

SYMBOLIC REGRESSION VIA NEURAL-GUIDED GP POPULATION SEEDING (NIPS 2021)

Methodology:

- The neural network generates the initial population.
- GP evolves the population through mutation (P_m) and crossover (P_c).



SYMBOLIC REGRESSION VIA NEURAL-GUIDED GP POPULATION SEEDING (NIPS 2021)

	Recovery rate (%)			
	All	Nguyen	R	Livermore
Ours	74.92	92.33	33.33	71.09
GEGL [Ahn et al., 2020]	64.11	86.00	33.33	56.36
Random Restart GP (i.e. GP only)	63.57	88.67	2.67	58.18
DSR (i.e. RNN only) [Petersen et al., 2021]	45.19	83.58	0.00	30.41
95% confidence interval	±1.54	±1.76	±2.81	±1.32

Proposed Unified Framework:

- Integrates five key strategies:

- ▶ Recursive problem simplification (AI Feynman).
- ▶ Neural-guided search (Deep Symbolic Regression).
- ▶ Large-scale pre-training (NeSymReS).
- ▶ Evolutionary search (Genetic Programming).
- ▶ Sparse linear models (SINDy).

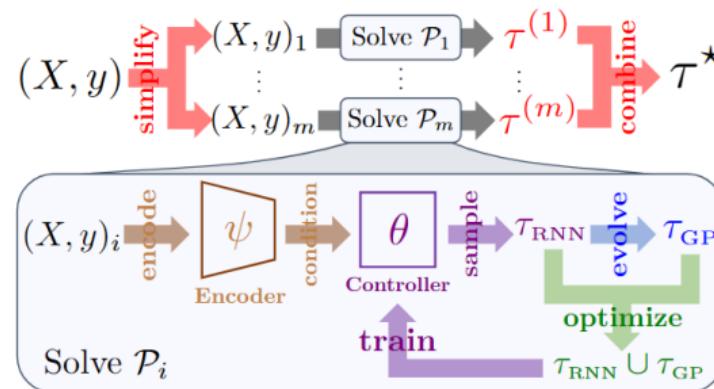
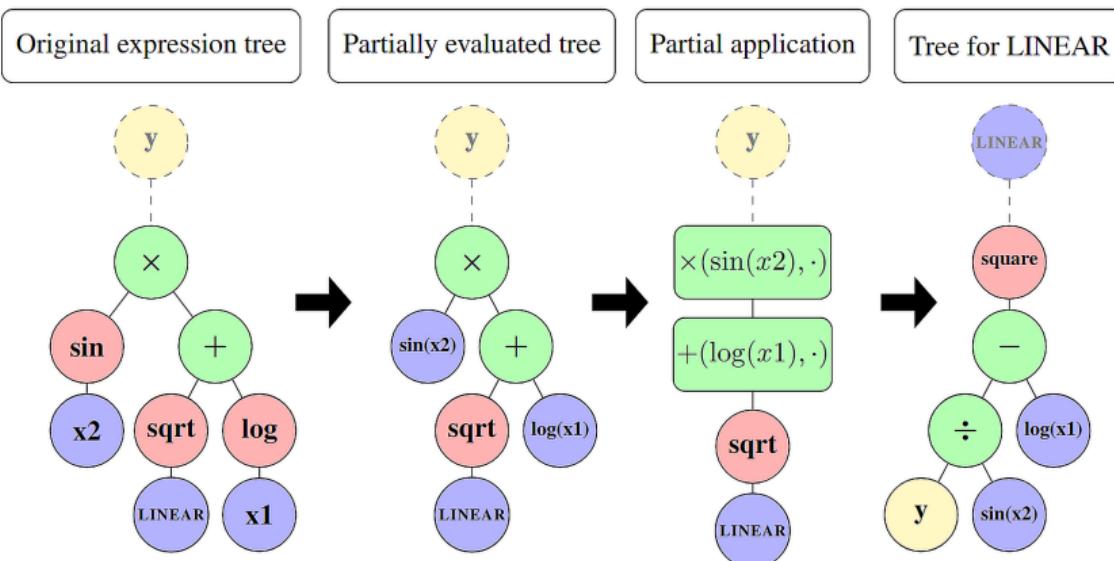
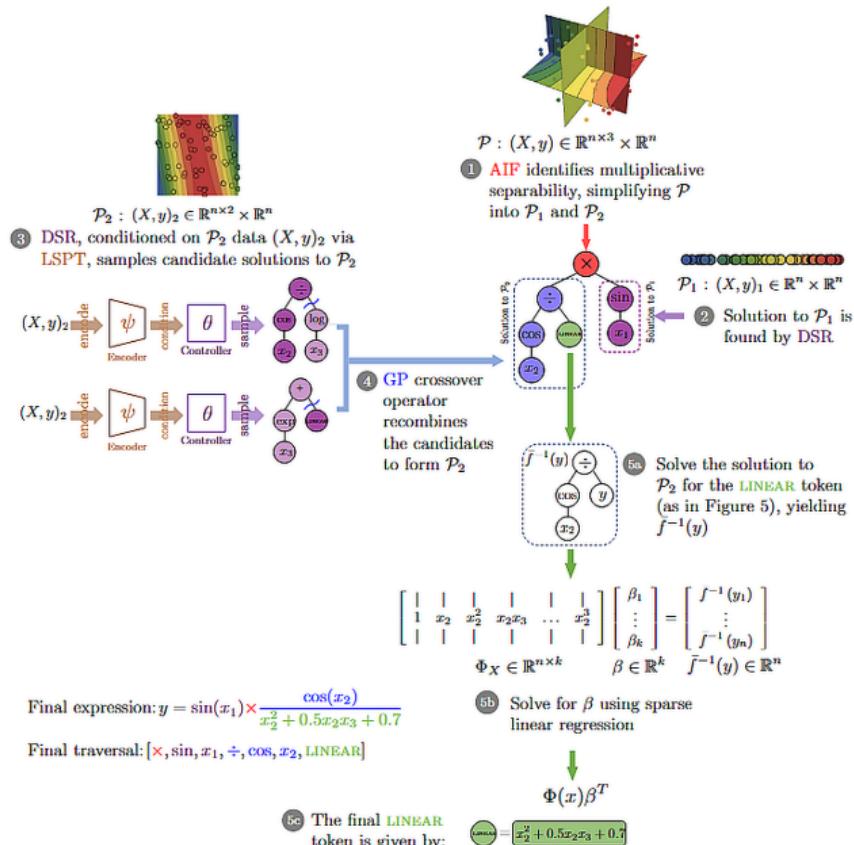


Figure 1: Unified deep symbolic regression. The five integrated solution strategies are color-coded: **AIF**, **DSR**, **LSPT**, **GP**, **LM**.

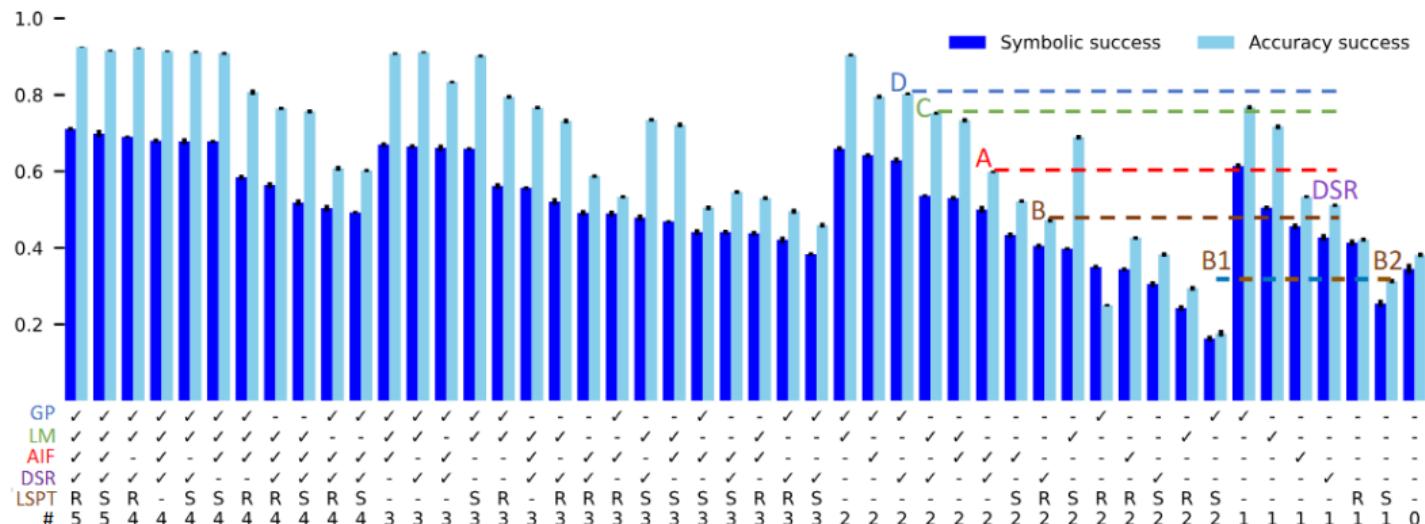
UNIFIED DEEP SYMBOLIC REGRESSION (NIPS 2022)



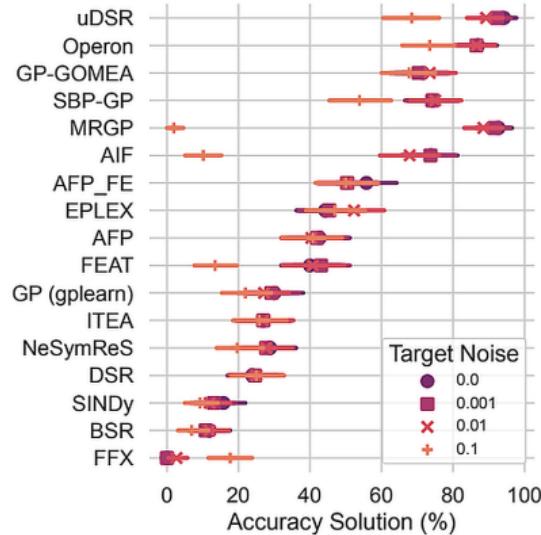
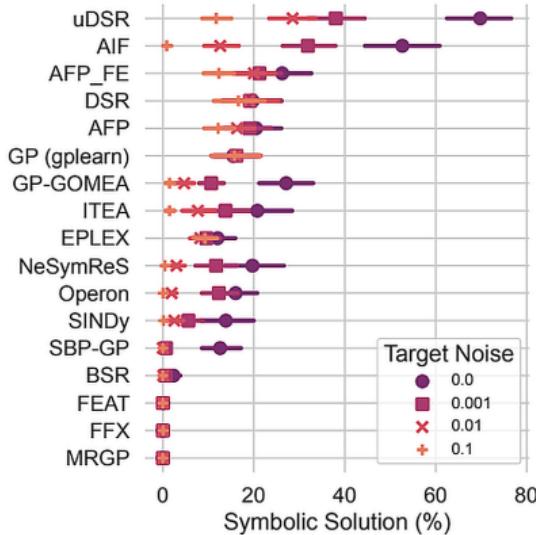
UNIFIED DEEP SYMBOLIC REGRESSION (NIPS 2022)

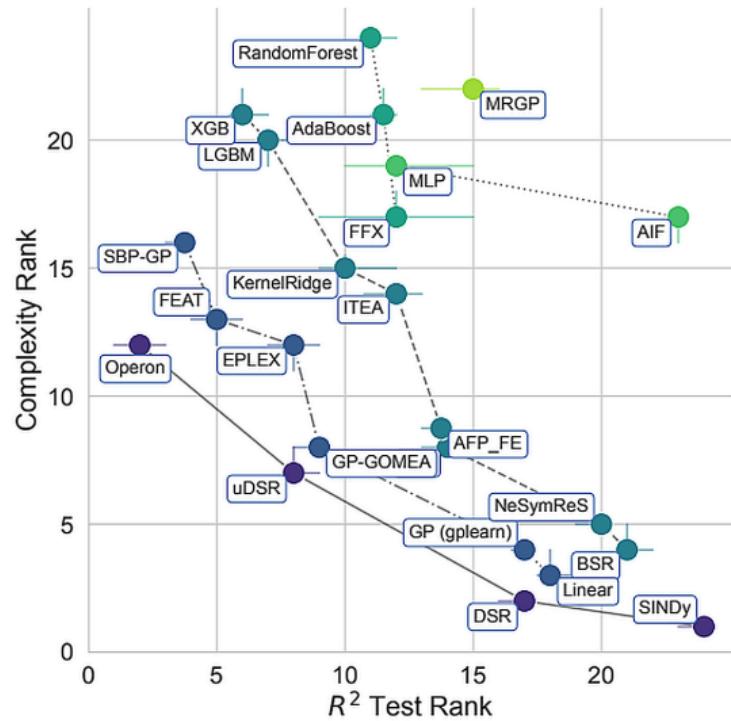


UNIFIED DEEP SYMBOLIC REGRESSION (NIPS 2022)



D: GP+DSR, DSR: Deep Symbolic Regression





LANGUAGE MODEL-BASED SYMBOLIC REGRESSION

NEURAL SYMBOLIC REGRESSION THAT SCALES (ICML 2021)

Methodology:

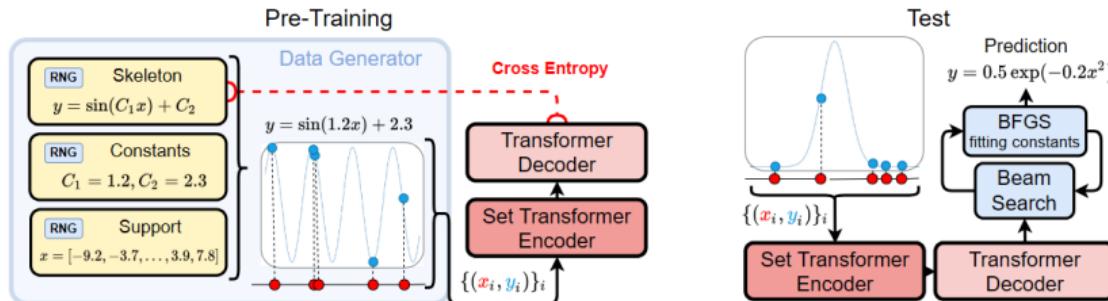
- Transformer-based architecture:

- ▶ **Encoder:** Maps $\{(x_i, y_i)\}$ to a latent space z .
- ▶ **Decoder:** Generates a skeleton e without numerical constants:

$$P(e_k|e_{<k}, z) = \text{softmax}(W_k h_k + b_k)$$

- Numerical constants are fit post-decoding using optimization methods (e.g., BFGS).
- Training uses a cross-entropy loss on skeletons:

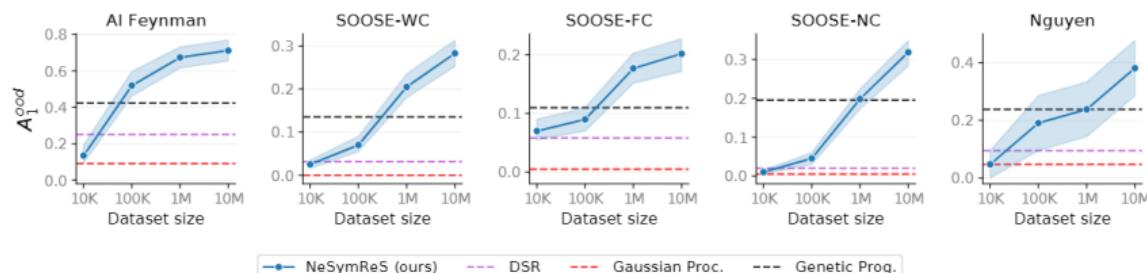
$$L = - \sum_k \log P_\theta(e_k|e_{<k}, X, Y)$$



RESULTS

Key Results:

- A_{ood} accuracy on AI-Feynman: 82.5% (NeSymReS) vs. 65.3% (Deep Symbolic Regression).
- Efficient scaling with larger datasets (up to 10M equations) and test-time data points.
- Robust against unseen equation structures (e.g., strictly out-of-sample skeletons, SOOSE).
- Limitations:
 - ▶ Supports up to $D = 3$ input dimensions.
 - ▶ Different instances of the same skeleton can represent vastly different functions depending on constants (e.g., $\sin(10x)$ vs. $\sin(0.1x)$).



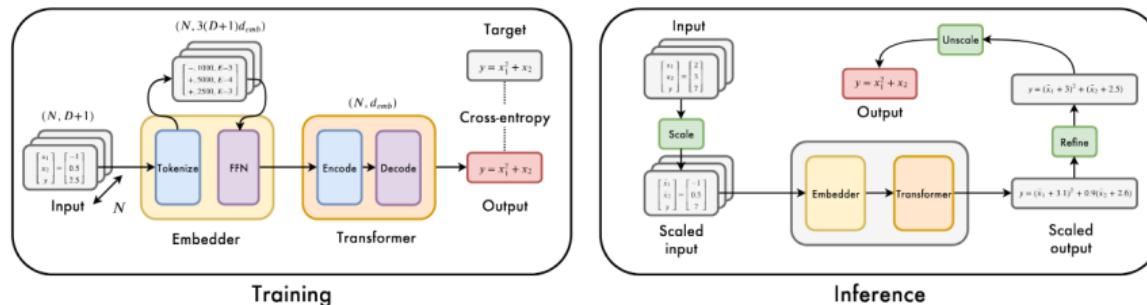
END-TO-END SYMBOLIC REGRESSION WITH TRANSFORMERS (NIPS 2022)

Methodology:

- End-to-end approach: Train Transformers to predict full expressions, including constants.
- Use symbolic-numeric tokenization:

$$f(x) = \sin(10x) \Rightarrow [\sin, *, 10, x]$$

- Refinement: Fine-tune predicted constants via BFGS with informed initialization.
- Scalable to larger input dimensions ($D = 10$) and datasets.



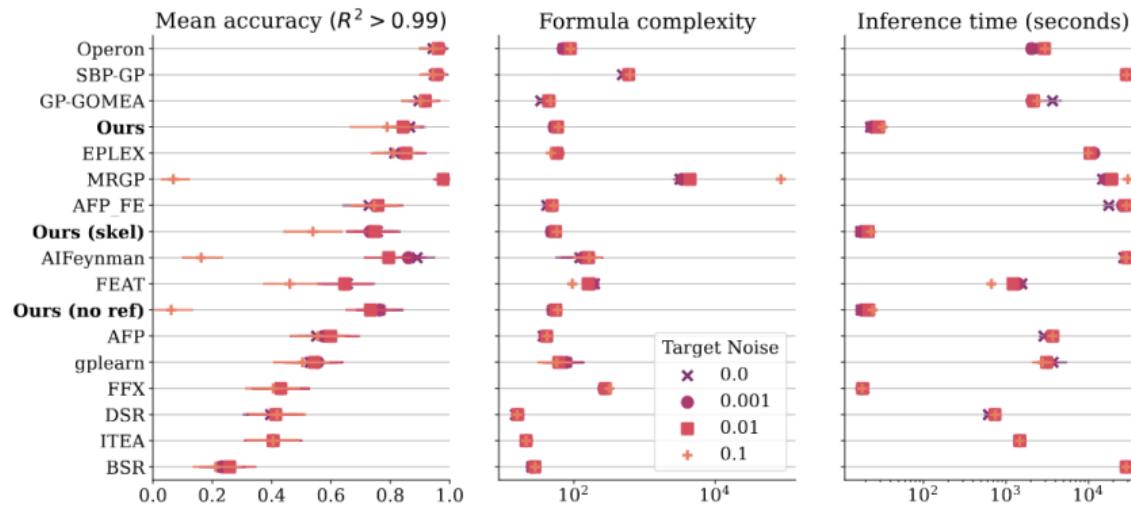
RESULTS

Key Results:

- Outperforms skeleton-based approaches in symbolic regression benchmarks:

$$R^2 = 0.68 \text{ (E2E with refinement)} \text{ vs. } R^2 = 0.43 \text{ (Skeleton-based)}$$

- Significant inference speed-up compared to genetic programming.



TRANSFORMER-BASED SYMBOLIC REGRESSION WITH JOINT SUPERVISION (ICLR 2023)

Motivation:

- Ill-posed problem: Different instances of the same skeleton behave differently.
- Poor feature extraction by previous encoders.

Proposed Method:

- **Feature Extraction:** Use a pure residual MLP framework:

$$O_i = \text{POS}(\text{MaxPool}(\text{PRE}(f_{i,j}), j \in K))$$

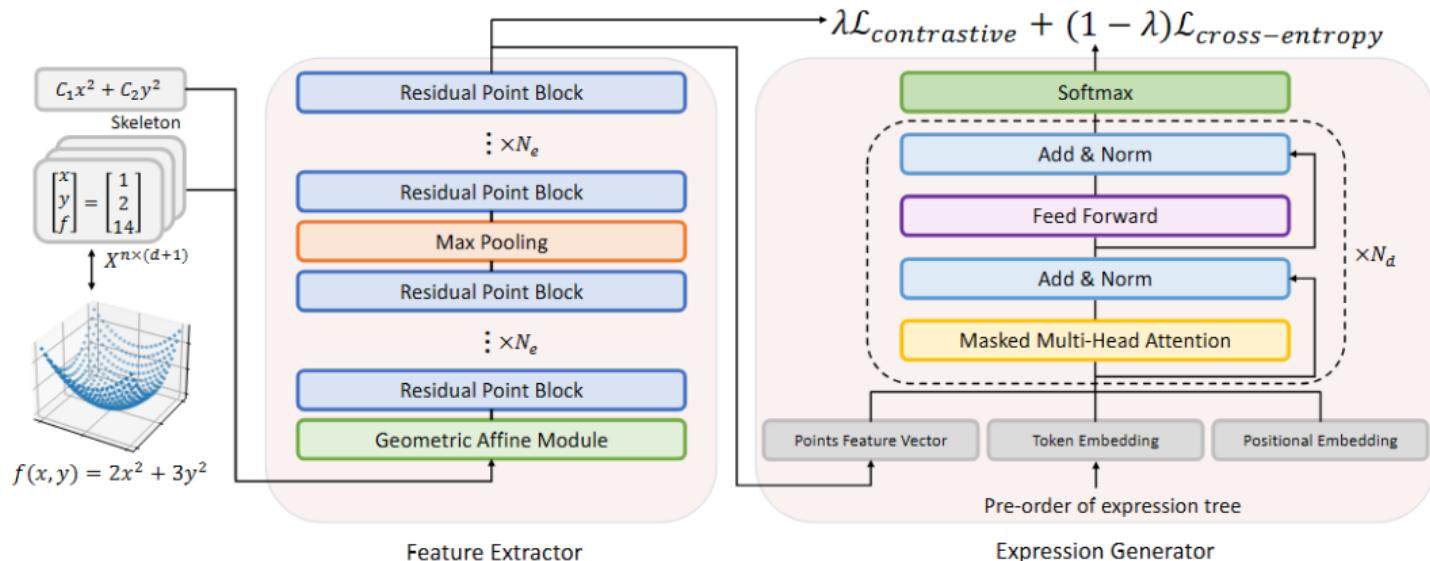
- **Joint Learning:** Combine Cross-Entropy (CE) and Contrastive Learning (CL) losses:

$$L = (1 - \lambda)L_{\text{CE}} + \lambda L_{\text{CL}}$$

- **Expression Generation:** Transformer-based autoregressive decoder:

$$P(\tau|v) = \prod_{i=1}^{|\tau|} P(\tau_i|\tau_{<i}, v)$$

TRANSFORMER-BASED SYMBOLIC REGRESSION WITH JOINT SUPERVISION



RESULTS

Benchmark	Ours	SymbolicGPT	NeSymReS	DSO	GP
	$R^2 \uparrow$				
Nguyen	0.99999	0.64394	0.97538	0.99489	0.89019
Constant	0.99998	0.69433	0.84935	0.99927	0.90842
Keijzer	0.98320	0.59457	0.97500	0.96928	0.90082
R	0.99999	0.71093	0.99993	0.97298	0.83198
AI-Feynman	0.99999	0.64682	0.99999	0.99999	0.92242
SSDNC	0.94782	0.74585	0.85792	0.93198	0.88913
Overall avg.	0.98850	0.67274	0.94292	0.97806	0.89049

LIMITATIONS OF LANGUAGE MODEL-BASED SYMBOLIC REGRESSION

Challenges:

- Symbolic regression is an **NP-hard problem**.
- Language models alone are unlikely to efficiently solve NP-hard problems.

Solutions:

- **Combine pre-trained language models with optimization techniques.**
- Transformer-Based Planning for Symbolic Regression (TPSR).

Methodology:

- Representation: Mathematical expressions are encoded as parse trees.
- MCTS Reward Function:

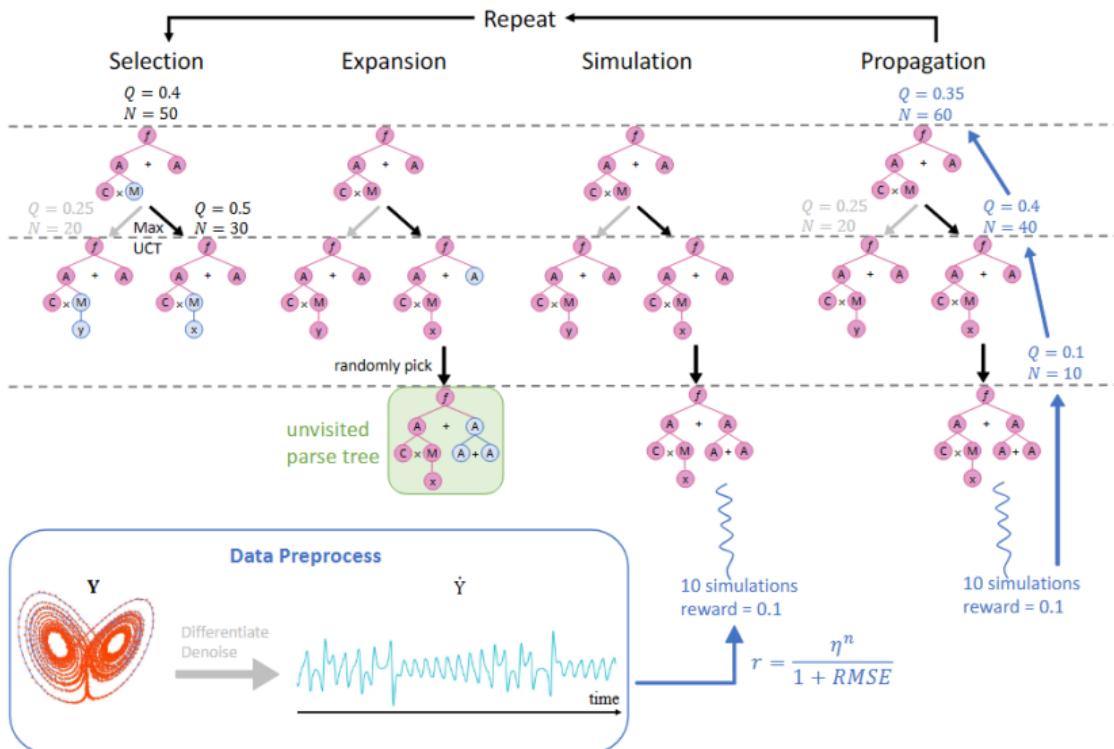
$$r = \frac{\eta^n}{1 + \sqrt{\frac{1}{N} \|Y_i - \tilde{f}(Y)\|^2}}$$

where η : parsimony factor < 1 , n : production rules count, N : number of data points.

- Upper Confidence Bound for Trees (UCT):

$$\text{UCT}(s, a) = Q(s, a) + c \sqrt{\frac{\ln N(s)}{N(s, a)}}$$

- ▶ $Q(s, a)$: Estimated reward for taking action a in state s .
- ▶ $N(s)$: Visit count for state s .
- ▶ $N(s, a)$: Visit count for action a at state s .
- ▶ c : Exploration coefficient controlling the balance between exploration and exploitation.

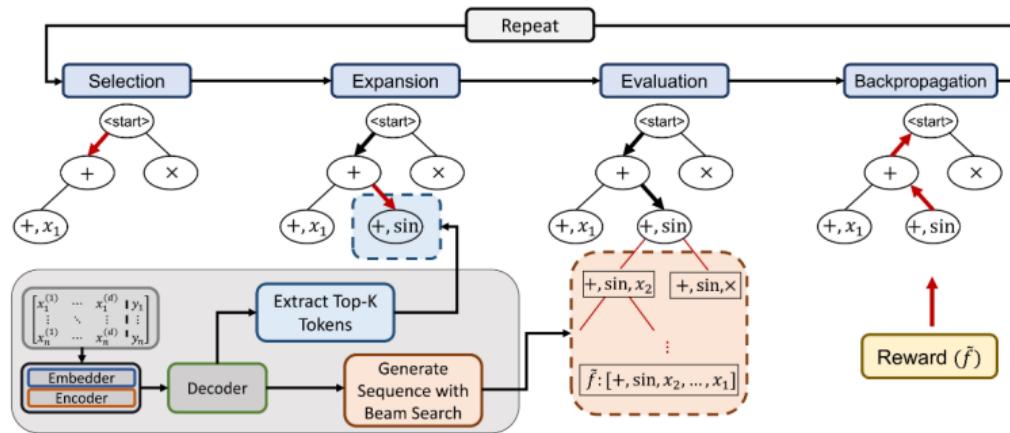


Benchmark	Expression	SPL	NGGP	GP
Nguyen-1	$x^3 + x^2 + x$	100%	100%	99%
Nguyen-2	$x^4 + x^3 + x^2 + x$	100%	100%	90%
Nguyen-3	$x^5 + x^4 + x^3 + x^2 + x$	100%	100%	34%
Nguyen-4	$x^6 + x^5 + x^4 + x^3 + x^2 + x$	99%	100%	54%
Nguyen-5	$\sin(x^2) \cos(x) - 1$	95%	80%	12%
Nguyen-6	$\sin(x^2) + \sin(x + x^2)$	100%	100%	11%
Nguyen-7	$\ln(x + 1) + \ln(x^2 + 1)$	100%	100%	17%
Nguyen-8	\sqrt{x}	100%	100%	100%
Nguyen-9	$\sin(x) + \sin(y^2)$	100%	100%	76%
Nguyen-10	$2 \sin(x) \cos(y)$	100%	100%	86%
Nguyen-11	x^y	100%	100%	13%
Nguyen-12	$x^4 - x^3 + \frac{1}{2}y^2 - y$	28%	4%	0%
Nguyen-1 ^c	$3.39x^3 + 2.12x^2 + 1.78x$	100%	100%	0%
Nguyen-2 ^c	$0.48x^4 + 3.39x^3 + 2.12x^2 + 1.78x$	94%	100%	0%
Nguyen-5 ^c	$\sin(x^2) \cos(x) - 0.75$	95%	98%	1%
Nguyen-8 ^c	$\sqrt{1.23x}$	100%	100%	56%
Nguyen-9 ^c	$\sin(1.5x) + \sin(0.5y^2)$	96%	90%	0%
Average		94.5%	92.4%	38.2%

TRANSFORMER-BASED PLANNING FOR SYMBOLIC REGRESSION (NIPS 2023)

TPSR Objective:

- Introduce Monte Carlo Tree Search (MCTS) into Transformer decoding to guide equation generation.
- Leverage pre-trained Transformer priors while incorporating feedback on:
 - ▶ Fitting accuracy.
 - ▶ Model complexity.



Methodology:

■ Transformer-based SR model enhanced with MCTS:

- ▶ s : Current state (partial equation sequence) in the Monte Carlo Tree Search (MCTS).
- ▶ a : Action (next token) extending s to a new state s' .
- ▶ $Q(s, a)$: Estimated quality (expected reward) of taking action a in state s .
- ▶ β : Hyperparameter balancing exploration and exploitation.
- ▶ $P_\theta(a|s)$: Probability of selecting action a in state s , predicted by the pre-trained Transformer model.
- ▶ $N(s)$: Visit count for state s (times s has been explored).
- ▶ $N(s')$: Visit count for the next state s' , resulting from action a .

$$P\text{-UCB}(s, a) = Q(s, a) + \beta \cdot P_\theta(a|s) \cdot \sqrt{\frac{\ln(N(s))}{1 + N(s')}}$$

$P\text{-UCB}(s, a)$

$$P\text{-UCB}(s, a) = Q(s, a) + \beta \cdot P_\theta(a|s) \cdot \sqrt{\frac{\ln(N(s))}{1 + N(s')}}$$

Key Components:

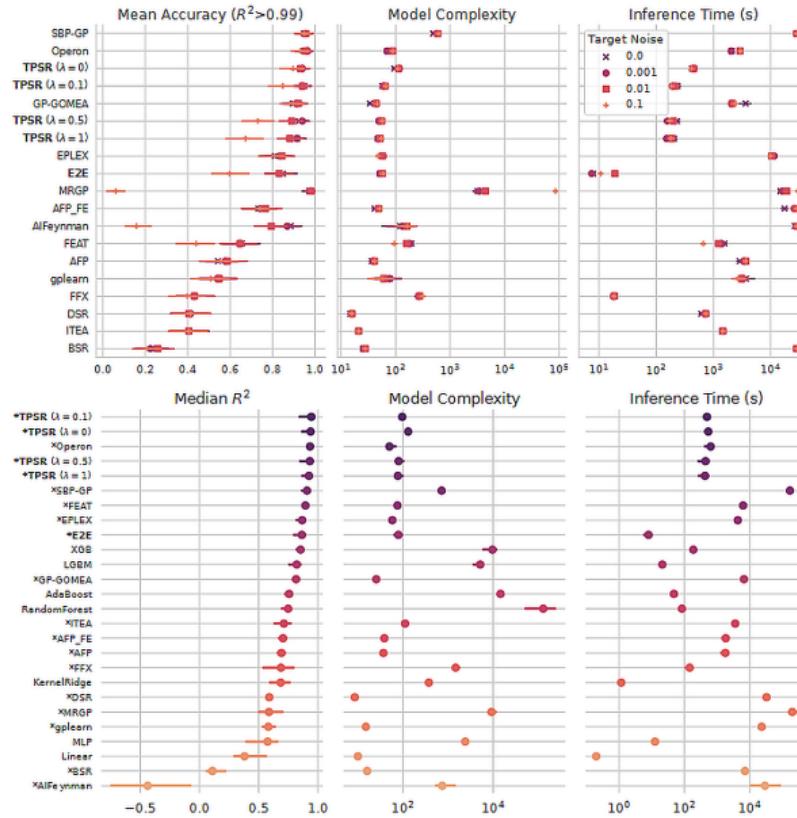
- **Exploit ($Q(s, a)$):** Prioritize high-quality actions.
- **Explore ($\sqrt{\frac{\ln(N(s))}{1 + N(s')}}$):** Encourage actions leading to less-visited states.
- **Prior ($P_\theta(a|s)$):** Leverage pre-trained Transformer knowledge to bias promising actions.

REWARD FUNCTION

Reward function combining accuracy and complexity:

$$r(f) = \frac{1}{1 + \text{NMSE}} + \lambda e^{-\frac{\ell(f)}{L}}$$

- NMSE: Normalized Mean Squared Error between predicted and actual values.
- $\ell(f)$: Length of the generated equation sequence.
- L : Maximum allowable sequence length.



SNIP: BRIDGING SYMBOLIC AND NUMERIC DOMAINS (ICLR 2024)

SNIP: A pre-training framework for joint symbolic-numeric learning using contrastive loss.

- Dual-Encoder Architecture:

$$\text{Numeric Encoder: } Z_V = A_V \cdot V_{LV}, \quad \text{Symbolic Encoder: } Z_S = A_S \cdot S_{LS}$$

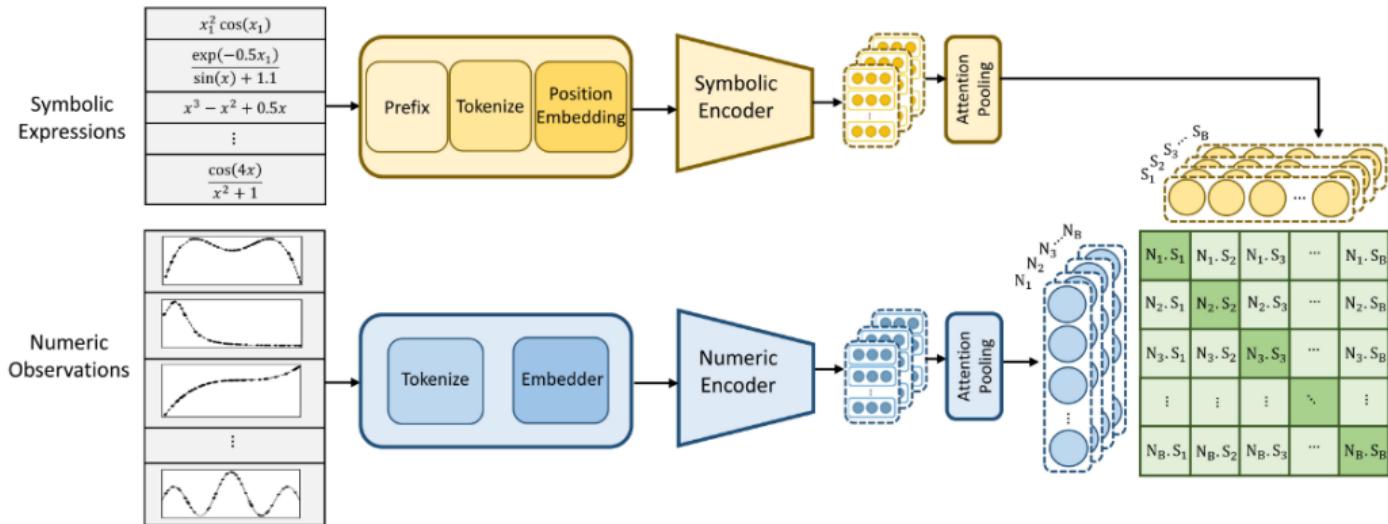
- Pre-training Objective:

$$L = - \sum_{(v,s) \in B} (\log \text{NCE}(Z_S, Z_V) + \log \text{NCE}(Z_V, Z_S))$$

where NCE aligns symbolic Z_S and numeric Z_V embeddings.

- Applications: Property prediction, symbolic regression via latent space optimization.

SNIP: BRIDGING SYMBOLIC AND NUMERIC DOMAINS (ICLR 2024)



USING SNIP FOR SYMBOLIC REGRESSION

SNIP in SR Pipeline:

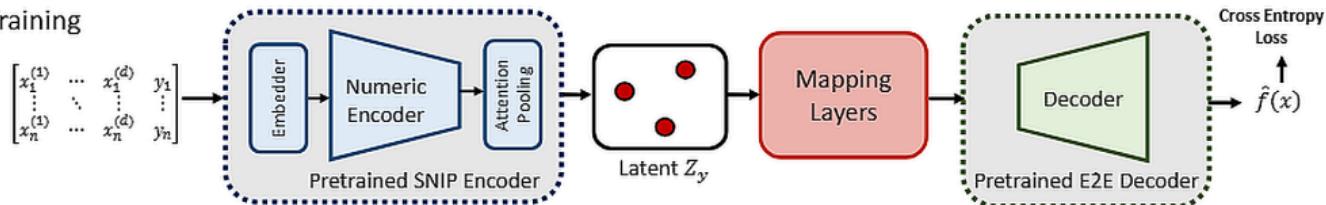
- Numeric encoder produces embeddings (Z_V).
- Integrated with a symbolic expression generation decoder (G_ω) via a mapping network.
- Training Objective: $L(f, f) = -\frac{1}{|f|} \sum_j \log P(t_j | t_1, \dots, t_{j-1}; G_\omega)$

Latent Space Optimization (LSO):

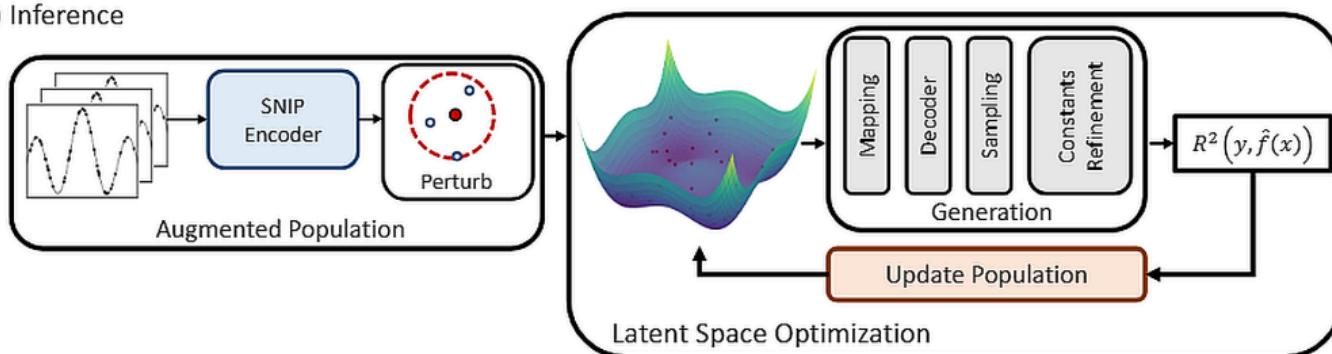
- Explore equations in the learned embedding space using optimization.
- Refine constant values for equations via numerical solvers.

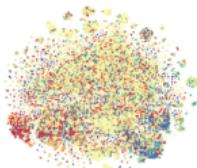
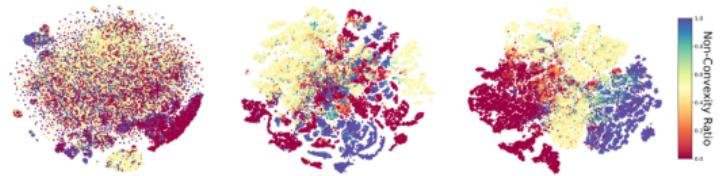
SNIP: BRIDGING SYMBOLIC AND NUMERIC DOMAINS (ICLR 2024)

(a) Training

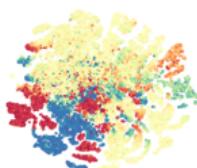
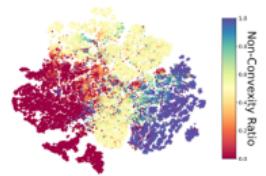
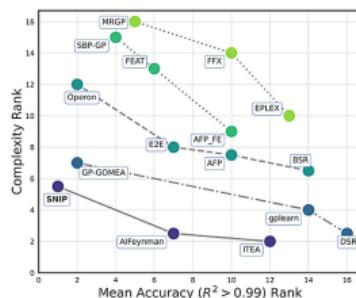
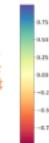
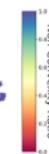


(b) Inference

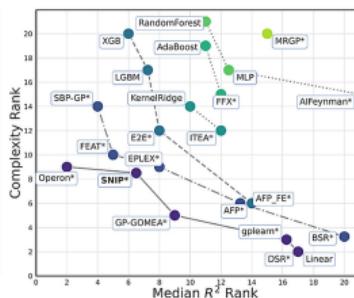




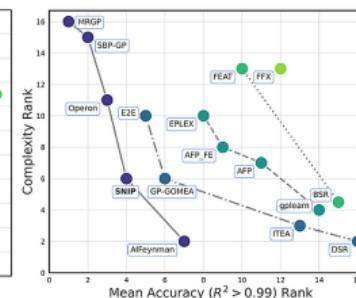
(a) Without Pretraining

(b) Pretrained,
Before Finetuning(c) Pretrained,
After Finetuning

(a) Strogatz datasets

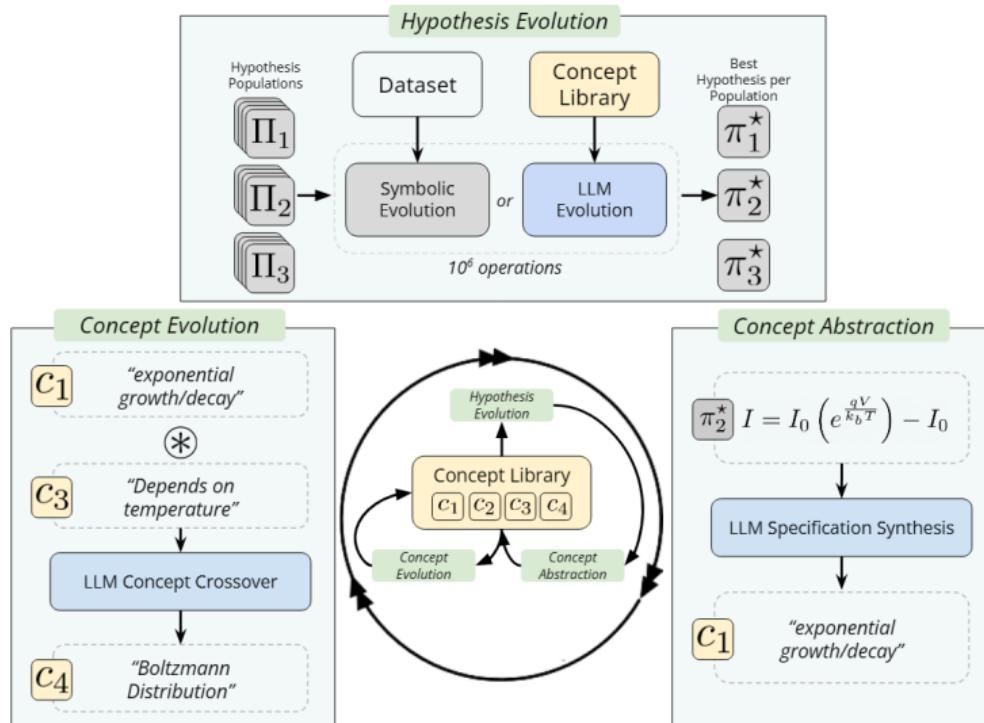


(b) Black-box datasets

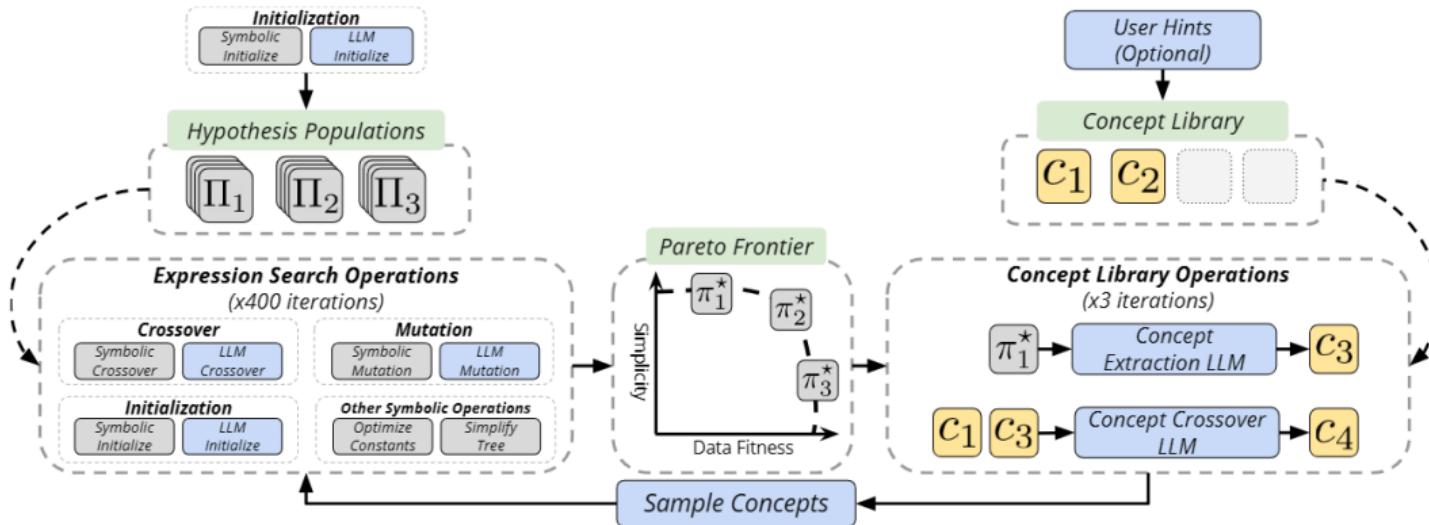


(c) Feynman datasets

SYMBOLIC REGRESSION WITH A LEARNED CONCEPT LIBRARY (NIPS 2024)



SYMBOLIC REGRESSION WITH A LEARNED CONCEPT LIBRARY (NIPS 2024)



SYMBOLIC REGRESSION WITH A LEARNED CONCEPT LIBRARY (NIPS 2024)

(System) Domain Specific Header	You are a helpful assistant that hypothesizes about the underlying assumptions that generated a list of good and bad mathematical expressions in detailed ways. My ultimate goal is to discover what assumptions generated the observed good mathematical expressions and excludes the bad mathematical expressions. Focus more on the good expressions, their mathematical structure, and any relation to physical concepts. Note that capital C represents an arbitrary constant.
(User) Concept Abstraction Prompt	Good Expression 1: {{gexpr1}} Good Expression 2: {{gexpr2}} Good Expression 3: {{gexpr3}} Good Expression 4: {{gexpr4}} Good Expression 5: {{gexpr5}} Bad Expression 1: {{bexpr1}} Bad Expression 2: {{bexpr2}} Bad Expression 3: {{bexpr3}} Bad Expression 4: {{bexpr4}} Bad Expression 5: {{bexpr5}}
(User) JSON Formatting Instructions	Propose {{N}} expressions that would be appropriate given the suggestions and expressions. Provide short commentary for each of your decisions. End with a JSON list that enumerates the proposed expressions following this format: ```json ["hyp1", "hyp2", ... "hyp{{N}}" , ...]``

SYMBOLIC REGRESSION WITH A LEARNED CONCEPT LIBRARY (NIPS 2024)

(Assistant)
Synthesized
Concepts
with
Explanation

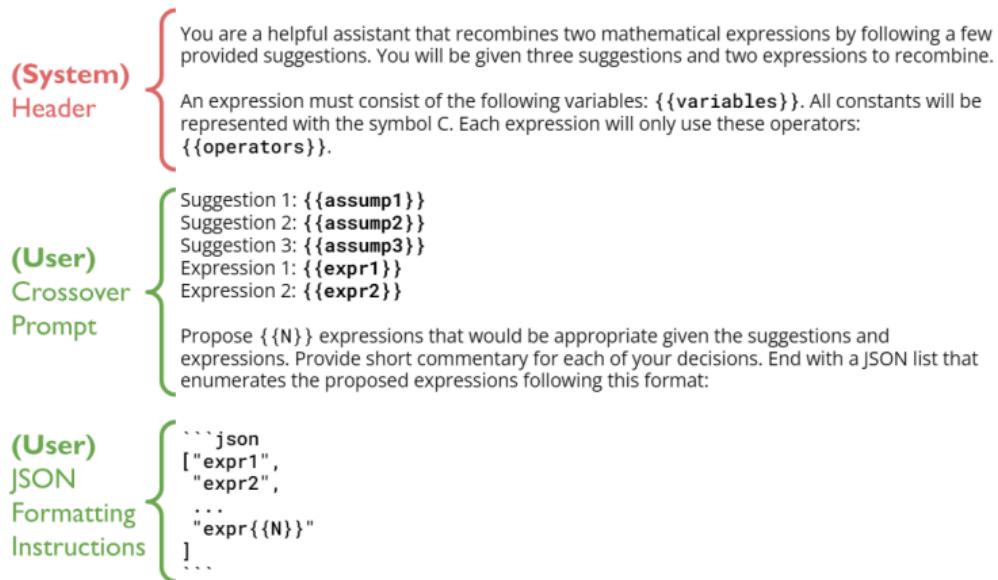
- Hypothesis 1:** The good expressions involve complex exponential or trigonometric functions with nested power operations, potentially indicating a link to physical phenomena that exhibit exponential growth or oscillatory behavior.
- Hypothesis 2:** The presence of sine and cosine functions in the good expressions suggests a connection to periodic phenomena in real-world applications, such as wave functions or harmonic oscillations.
- Hypothesis 3:** The inclusion of arbitrary constants in the good expressions may represent parameters in physical models, implying that the expressions are modeling dynamic systems or relationships between variables.
- Hypothesis 4:** The good expressions involve repeated application of mathematical operations (e.g., exponentiation, multiplication, and addition) in a layered manner, indicating a depth of complexity that mirrors intricate physical systems or natural processes.
- Hypothesis 5:** The mathematical relationships within the good expressions exhibit a balance between simplicity and complexity, suggesting that they capture fundamental principles or underlying structures that govern physical phenomena.

Here is the list of proposed expressions in JSON format:

```
```json
[{"text": "The good expressions involve complex exponential or trigonometric functions with nested power operations, potentially indicating a link to physical phenomena that exhibit exponential growth or oscillatory behavior."}, {"text": "The presence of sine and cosine functions in the good expressions suggests a connection to periodic phenomena in real-world applications, such as wave functions or harmonic oscillations."}, {"text": "The inclusion of arbitrary constants in the good expressions may represent parameters in physical models, implying that the expressions are modeling dynamic systems or relationships between variables."}, {"text": "The good expressions involve repeated application of mathematical operations (e.g., exponentiation, multiplication, and addition) in a layered manner, indicating a depth of complexity that mirrors intricate physical systems or natural processes."}, {"text": "The mathematical relationships within the good expressions exhibit a balance between simplicity and complexity, suggesting that they capture fundamental principles or underlying structures that govern physical phenomena."}]
```
```

(Assistant)
Formatted
JSON
Summary

SYMBOLIC REGRESSION WITH A LEARNED CONCEPT LIBRARY (NIPS 2024)



SYMBOLIC REGRESSION WITH A LEARNED CONCEPT LIBRARY (NIPS 2024)

(Assistant)
Generated
Explanation

Based on the provided suggestions and expressions, I propose the following five recombined expressions:

1. `log(exp(C - sqrt(theta ^ C)) + C)`: This expression combines the exponential function from Expression 2 with the logarithmic function from Expression 1. The `- sqrt(theta ^ C)` term is taken from Expression 2 and added to the exponential function, while the `+ C` term is taken from Expression 1.
2. `(sin(C - sqrt(theta)) * (theta ^ log(theta / theta)) + C)`: This expression combines the sine function from Expression 1 with the power operation from the same expression. The `(theta ^ log(theta / theta))` term is taken from Expression 1, and the `sin(C - sqrt(theta))` term is also from Expression 1. The `+ C` term is added at the end.
3. `exp(C - sqrt(theta)) * sin(C - sqrt(theta))`: This expression combines the exponential function from Expression 2 with the sine function from Expression 1. Both the exponential and sine functions are taken from separate expressions and multiplied together.
4. `log(theta ^ log(theta / theta)) + C - sqrt(theta ^ C)`: This expression combines the logarithmic function from Expression 1 with the power operation from the same expression. The `(theta ^ log(theta / theta))` term is taken from Expression 1, and the `- sqrt(theta ^ C)` term is taken from Expression 2. The `+ C` term is added at the end.
5. `(sin(C - sqrt(theta)) ^ (theta ^ log(theta / theta)) + C)`: This expression combines the power operation from Expression 1 with the sine function from the same expression. The `(theta ^ log(theta / theta))` term is taken from Expression 1, and the `sin(C - sqrt(theta))` term is also from Expression 1. The `+ C` term is added at the end.

(Assistant)
Formatted
JSON
Summary

Here is the list of proposed expressions in JSON format:

```
...json
["log(exp(C - sqrt(theta ^ C)) + C)",
 "(sin(C - sqrt(theta)) * (theta ^ log(theta / theta)) + C)",
 "exp(C - sqrt(theta)) * sin(C - sqrt(theta))",
 "log(theta ^ log(theta / theta)) + C - sqrt(theta ^ C)",
 "(sin(C - sqrt(theta)) ^ (theta ^ log(theta / theta)) + C)"]
```

SYMBOLIC REGRESSION WITH A LEARNED CONCEPT LIBRARY (NIPS 2024)

| GPlearn | AFP | AFP-FE | DSR | uDSR | AI Feynman | PySR | LaSR |
|---------|--------|--------|--------|--------|------------|--------|---------------|
| 20/100 | 24/100 | 26/100 | 23/100 | 40/100 | 38/100 | 59/100 | 72/100 |

Table 1: Results on 100 Feynman equations from [49]. We report exact match solve rate for all models. LASR achieves the best exact match solve rate using the same hyperparameters as PySR.

| Type of Solve | PySR | LASR (Llama3-8B) | | | LASR (GPT-3.5) |
|---------------|--------|------------------|--------|---------|----------------|
| | | p = 1% | p = 5% | p = 10% | p = 1% |
| Exact Solve | 59/100 | 67/100 | 69/100 | 71/100 | 72/100 |
| Almost Solve | 7/100 | 5/100 | 6/100 | 2/100 | 3/100 |
| Close | 16/100 | 9/100 | 12/100 | 12/100 | 10/100 |
| Not Close | 18/100 | 19/100 | 13/100 | 16/100 | 15/100 |

Table 2: Evaluation results on Feynman dataset by cascading LASR's LLM backbone (llama3-8b, gpt-3.5-turbo) and changing the probability of calling the model ($p = [0.01, 0.05, 0.10]$) in the order of increasing concept guidance. LASR outperforms PySR even with minimal concept guidance using an open-source LLM.

SPARSE LEARNING-BASED SYMBOLIC REGRESSION

EXTRAPOLATION AND LEARNING EQUATIONS (ICLR 2017 WORKSHOP)

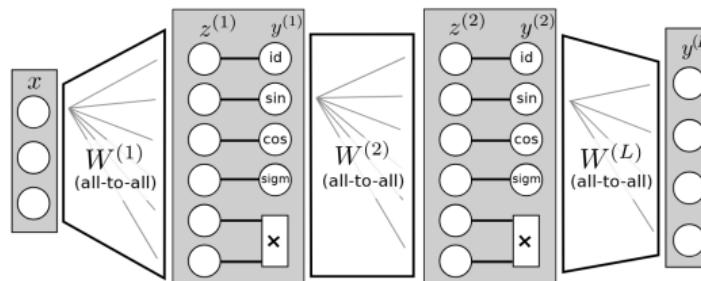
Equation Learner (EQL):

- EQL uses a feed-forward architecture with specialized units:

$$z^{(l)} = W^{(l)}y^{(l-1)} + b^{(l)}$$

$$y^{(l)} = (f_1(z_1), \dots, f_u(z_u), g_1(z_{u+1}, z_{u+2}), \dots)$$

- Unary Functions (f_i): id, sin, cos, sigmoid.
- Binary Functions (g_j): Multiplication of inputs.
- The final output is produced by a linear regression layer.



Training and Loss Function:

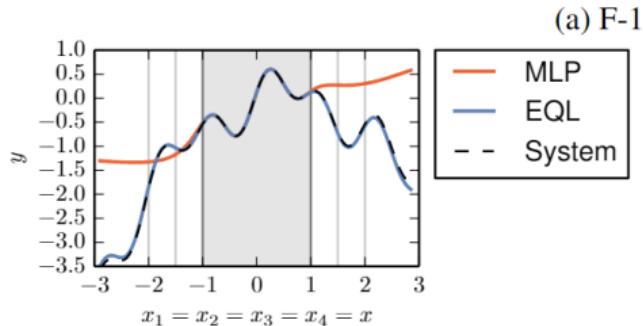
- Loss function with regularization:

$$L(D) = \frac{1}{N} \sum_{i=1}^N \|\psi(x_i) - y_i\|^2 + \lambda \sum_{l=1}^L \|W^{(l)}\|_1$$

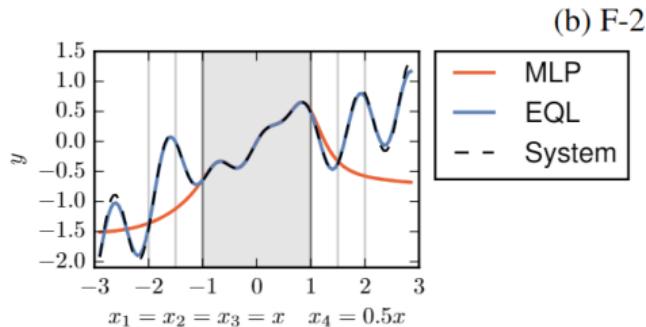
- Hybrid regularization: L_1 for sparsity and L_0 for fine-tuning.
- Regularization Strategy:
 - ▶ **Initial Phase** ($t < t_1$): No regularization ($\lambda = 0$), allowing free parameter exploration.
 - ▶ **Middle Phase** ($t_1 \leq t \leq t_2$): Apply L_1 -regularization to encourage sparsity.
 - ▶ **Final Phase** ($t > t_2$): Disable L_1 -regularization and fine-tune using L_0 -constraints.
- Transition Points:

$$t_1 = \frac{1}{4}T, \quad t_2 = \frac{19}{20}T$$

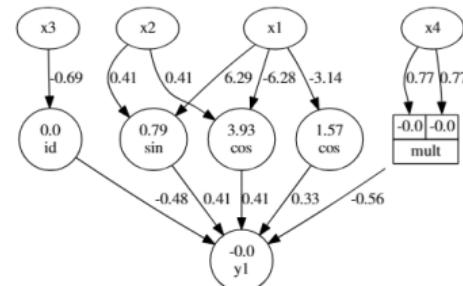
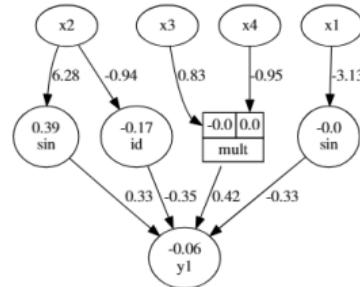
EQL RESULTS



learned formula: $-0.33 \sin(-3.13x_1) + 0.33 \sin(6.28x_2 + 0.39) + 0.33x_2 - 0.056 - 0.33x_3x_4$



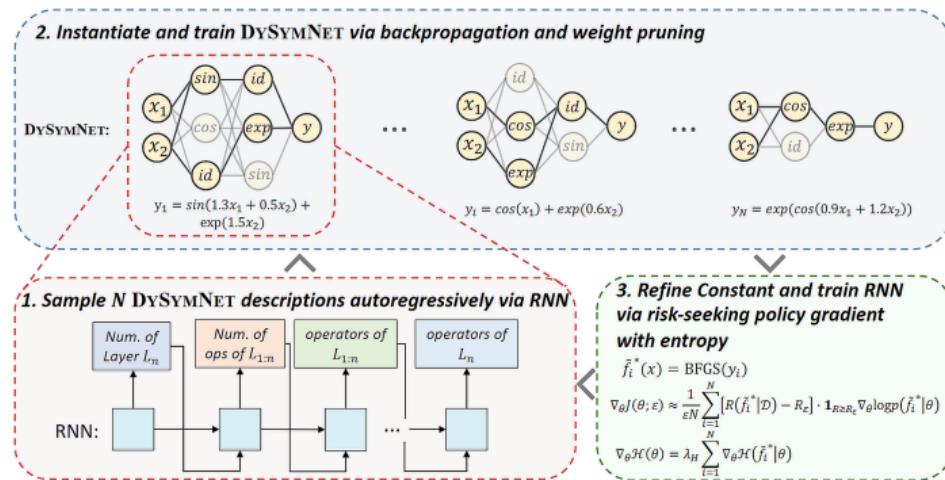
learned formula: $0.33 \cos(3.14x_1 + 1.57) + 0.33x_3 - 0.33x_4^2 + 0.41 \cos(-6.28x_1 + 3.93 + 0.41x_2) + 0.41 \sin(6.29x_1 + 0.79 + 0.41x_2)$



DYSYMNET: NEURAL-GUIDED DYNAMIC SYMBOLIC REGRESSION (ICML 2024)

- **DYSYMNET**: A novel SR framework based on reinforcement learning (policy gradient) for architecture search.
- **Policy Gradient Objective for Risk-Seeking Optimization:**

$$J_{\text{risk}}(\theta_c; \epsilon) = \mathbb{E}[R(f^*) \mid R(f^*) \geq R_\epsilon(\theta_c)].$$



DYSYMNENET RESULTS

| Data Group | Benchmark | DYSYMNENET | | uDSR | | NGGP | | NeSymReS | | EQL | |
|--------------------------|-----------|----------------------------|----------------------------|----------------------------|---------------------|---------------------|----------------|----------------|----------------|----------------|----------------|
| | | $R^2 \uparrow$ | $R^2 \uparrow$ | $R^2 \uparrow$ | $R^2 \uparrow$ | $R^2 \uparrow$ | $R^2 \uparrow$ | $R^2 \uparrow$ | $R^2 \uparrow$ | $R^2 \uparrow$ | $R^2 \uparrow$ |
| Standard
$(d \leq 2)$ | Nguyen | 0.9999 ± 0.0001 | 0.9989 ± 0.0000 | 0.9848 ± 0.0000 | 0.9221 ± 0.0000 | 0.9758 ± 0.0032 | | | | | |
| | Nguyen* | 0.9999 ± 0.0000 | 0.9999 ± 0.0000 | 0.9999 ± 0.0001 | 0.3523 ± 0.0000 | 0.7796 ± 0.0019 | | | | | |
| | Constant | 0.9992 ± 0.0000 | 0.9989 ± 0.0002 | 0.9989 ± 0.0002 | 0.7670 ± 0.0000 | 0.9817 ± 0.0088 | | | | | |
| | Keijzer | 0.9986 ± 0.0002 | 0.9968 ± 0.0000 | 0.9912 ± 0.0005 | 0.9358 ± 0.0000 | 0.9446 ± 0.0014 | | | | | |
| | Livermore | 0.9896 ± 0.0001 | 0.9670 ± 0.0003 | 0.9665 ± 0.0005 | 0.9195 ± 0.0000 | 0.7496 ± 0.0002 | | | | | |
| | R | 0.9895 ± 0.0012 | 0.9999 ± 0.0000 | 0.9999 ± 0.0000 | 0.9368 ± 0.0000 | 0.8588 ± 0.0061 | | | | | |
| | Jin | 1.0000 ± 0.0000 | 0.9864 ± 0.0002 | 0.9942 ± 0.0001 | 0.9987 ± 0.0000 | 0.9921 ± 0.0002 | | | | | |
| | Koza | 1.0000 ± 0.0000 | 0.9965 ± 0.0000 | 0.9999 ± 0.0000 | 0.9999 ± 0.0000 | 0.7571 ± 0.0022 | | | | | |
| Data Group | Benchmark | $R^2 \uparrow$ | $R^2 \uparrow$ | $R^2 \uparrow$ | $R^2 \uparrow$ | $R^2 \uparrow$ | $R^2 \uparrow$ | $R^2 \uparrow$ | $R^2 \uparrow$ | $R^2 \uparrow$ | $R^2 \uparrow$ |
| SRBench
$(d \geq 2)$ | Feynman | 0.9931 ± 0.0015 | 0.9806 ± 0.0003 | 0.9190 ± 0.0004 | 0.9234 ± 0.0000 | 0.5641 ± 0.0028 | | | | | |
| | Strogatz | 0.9968 ± 0.0031 | 0.9455 ± 0.0003 | 0.9534 ± 0.0003 | 0.8816 ± 0.0000 | 0.6511 ± 0.0012 | | | | | |
| | Black-box | 0.8908 ± 0.0028 | 0.6697 ± 0.0010 | 0.6086 ± 0.0021 | 0.4226 ± 0.0000 | 0.4528 ± 0.0107 | | | | | |

THANKS FOR LISTENING!

EMAIL: HENGZHE.ZHANG@ECS.VUW.AC.NZ

GITHUB PROJECT: [HTTPS://GITHUB.COM/HENGZHE-ZHANG/EVOLUTIONARYFOREST/](https://github.com/hengzhe-zhang/EvolutionaryForest/)