CS 957, System-2 AI
Symbolic Regression

Mahdi Samiei

Mar 2025

Sharif University of Technology

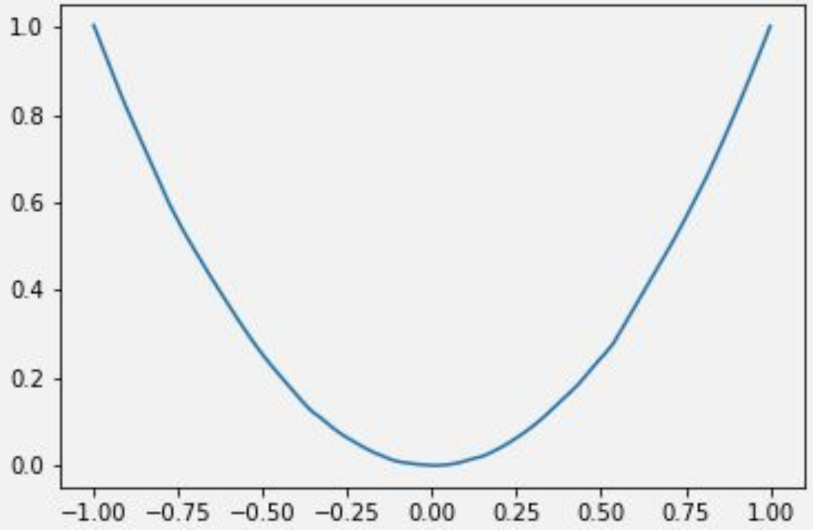
A Simple Question …

If you were given a list of numbers like this, what would you predict?

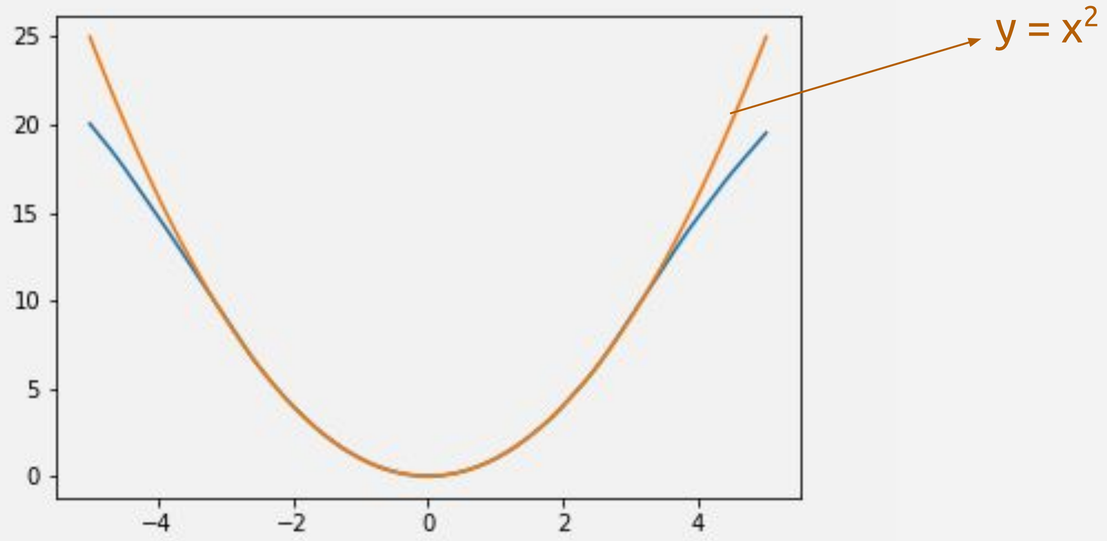
X	Y
0.31	0.0961
-0.1	0.01
0.21	0.0441
0.64	0.4096
-0.32	0.1024
0.9	0.81
0.5	0.25
0.2	?
1.1	?

Let’s Fit a Neural Model …

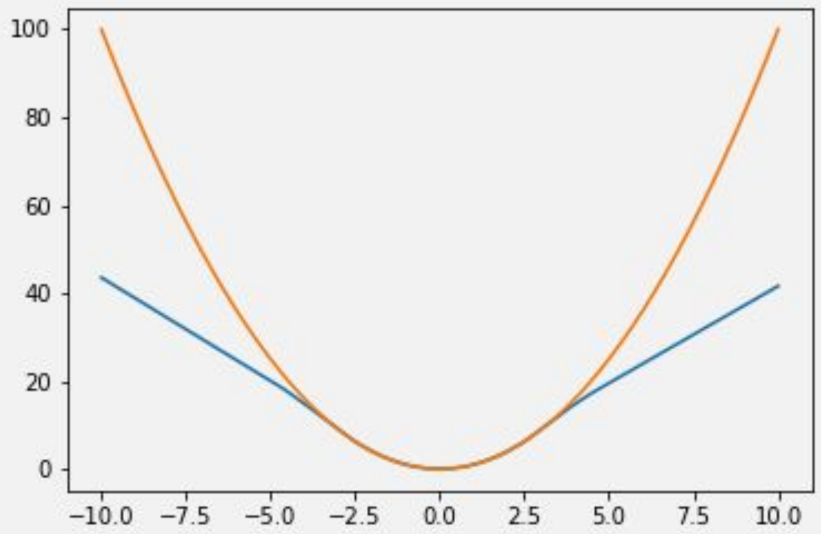
Seems Good!



Can’t Extrapolate :(



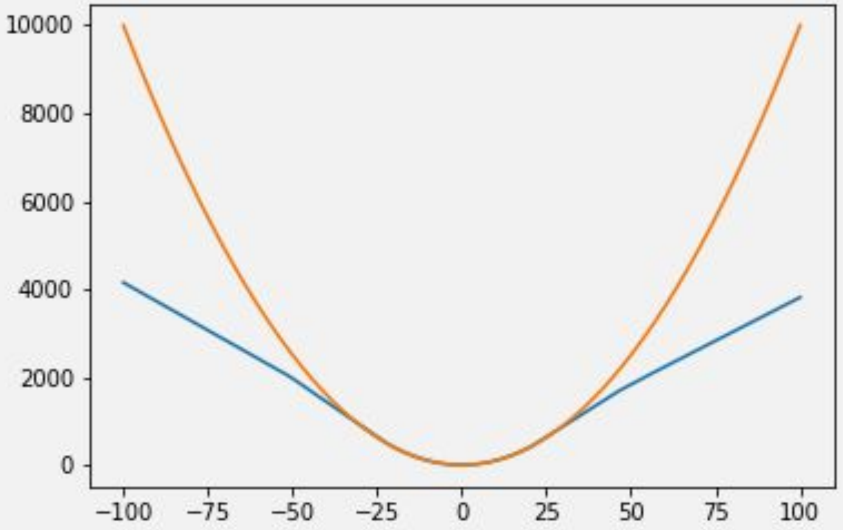
Can't Extrapolate :((



More Training Data?
More Parameters?

X	Y
0.31	0.0961
1.1	1.21
0.21	0.0441
5	25
-0.32	0.1024
9.1	82.81
6.3	39.69
1.2	?
10.1	?

But there is always another ood :(((



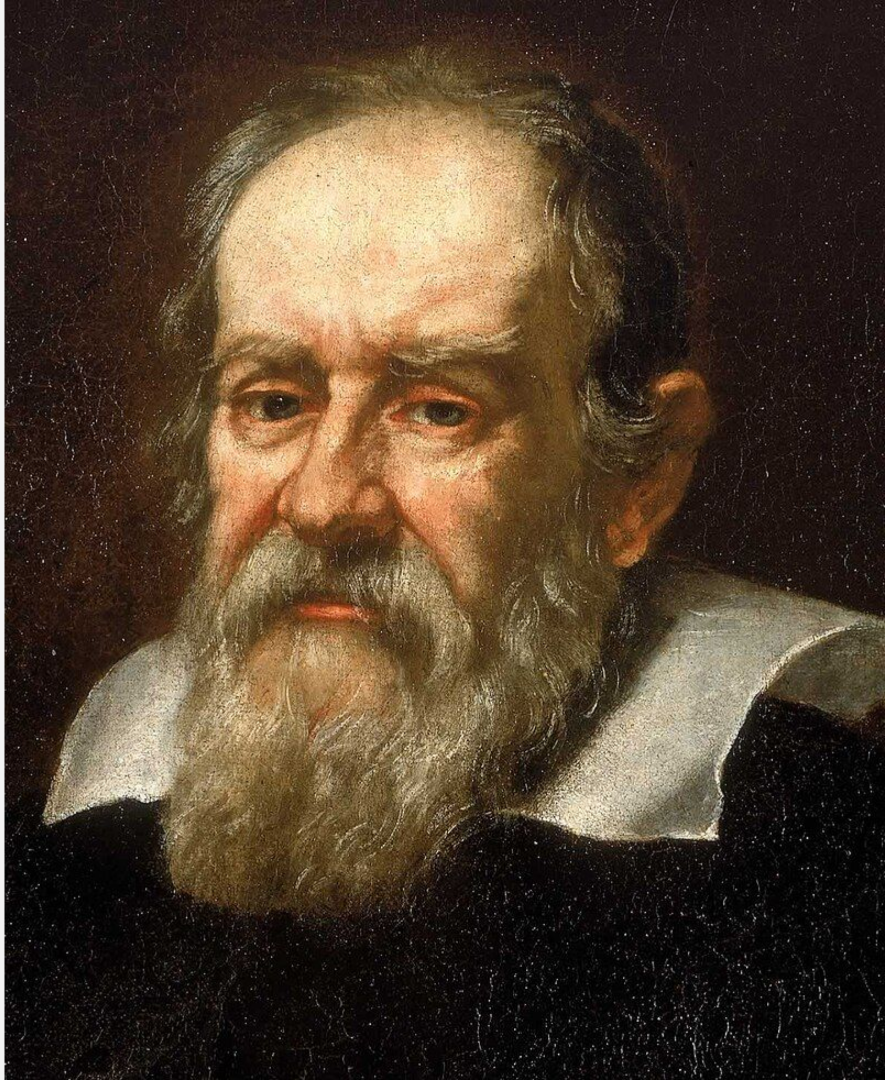
What is Symbolic Regression

🧮 discover a mathematical expression that describes the relationship between input output variables.

🧠 not a simple task!

🔬 Scientific discovery, Engineering system identification, Time series prediction, ...

🗝️ Remember NP? Hard to Answer, Easy to Verify



What is Symbolic Regression, vs Neural Regression

📐 Traditional regression fits parameters to a predefined model structure, while symbolic regression searches for both the structure and parameters simultaneously.

💡 Interpretable formula

🚧 Extrapolation to Out-of-Distribution Data

🏗️ Generalization to more complex formulas

🖼️ Regression is a metaphor, get the gist

What is Symbolic Regression, Structure of Task

➡️ Consider Inference Phase: Given $\{(x_i, y_i)\}$, We should discover f that $y_i = f(x_i)$

➡️ Indeed each task is a sample for symbolic regression.

➡️ Similar to Meta-Learning Setting

🤔 Remember Definition of Generalization?

📊 The performance of our symbolic regression method can be measured either on OoD data or OoD equations.

What is Symbolic Regression, Structure of Data

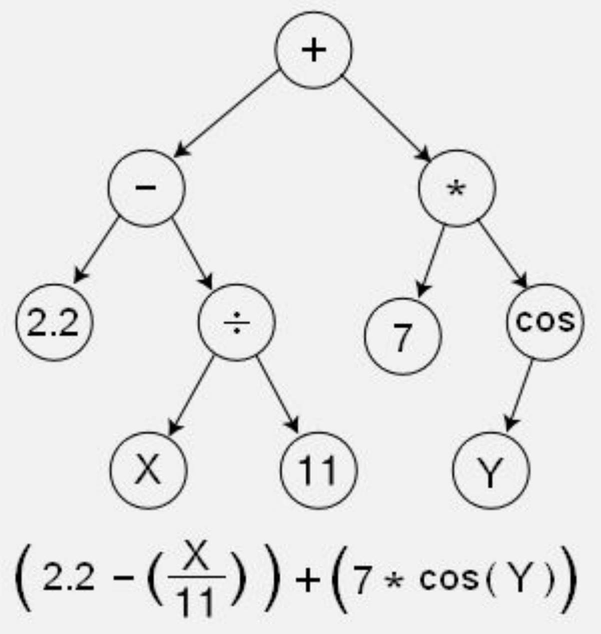
🌳 We usually represent the equation in the form of a computational tree.

🌳 Symbolic Regression Two Phases:

1️⃣Discover Skeleton with Placeholders for Constants

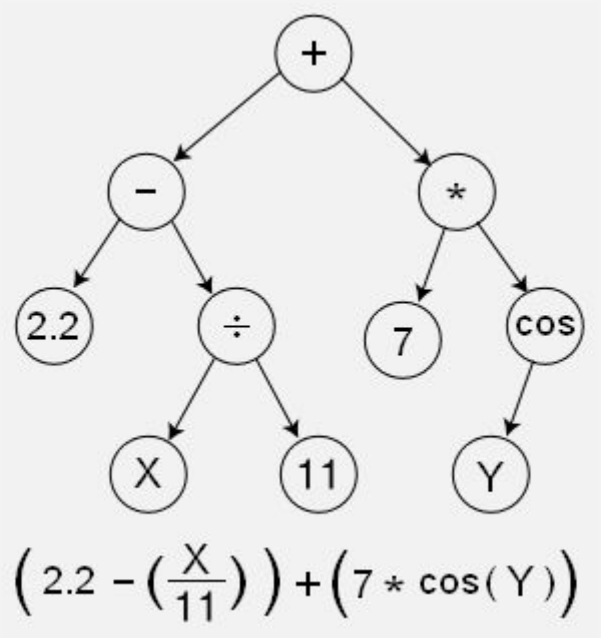
2️⃣Find Best Value for Constants by Optimization

🤪 What is Most Dummy Idea?


+
−
*
2.2
÷
7
cos
X
11
Y
$\left(2.2 - \left(\frac{X}{11}\right)\right) + \left(7 * \cos(Y)\right)$

Dummy Idea: Brute Force, Search

- 🔎 Brute-Force, Exhaustive Search
- 🏃 Local Search Methods, User Heuristic Function
    - 📍 Definition of Successor Function
    - 📍 Discrete Nature
    - 📍 Local Minimas


+
−
*
2.2
÷
7
cos
X
11
Y
$\left(2.2 - \left(\frac{X}{11}\right)\right) + \left(7 * \cos(Y)\right)$

🧬 Better Idea, Genetic Algorithm

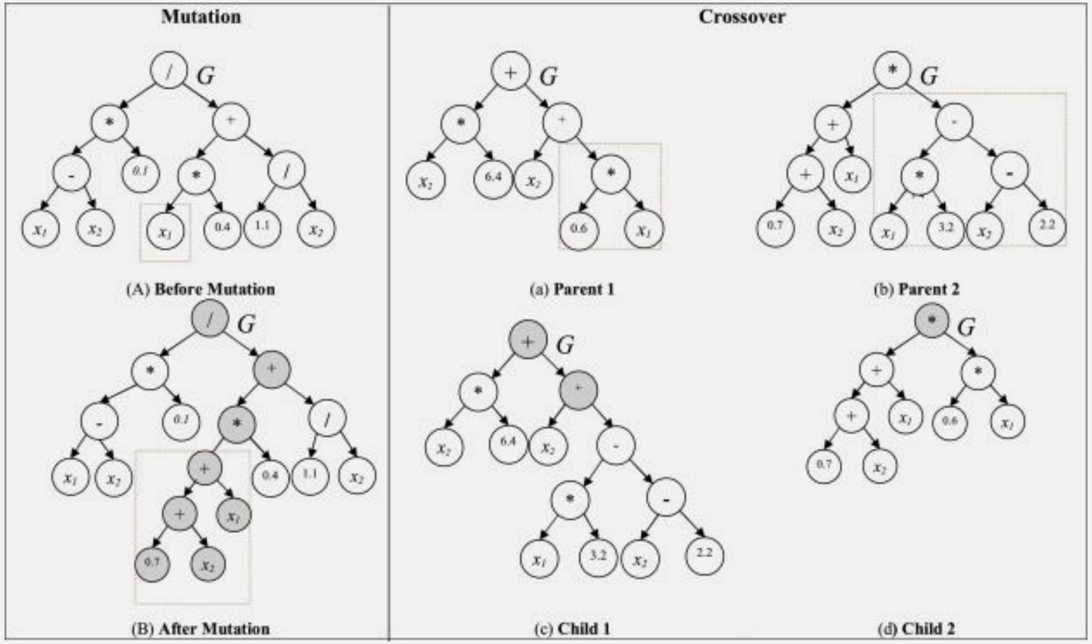
👑 Unrivaled until 2016

💪 Effective even today

🤔 The reason?

Combining sub-solutions.

🤔 Problems?



Main drawbacks of classic methods

❌ string of symbols grows exponentially with the string length.

❌ They do not improve with experience. As every equation is regressed from scratch, the system does not improve if access to more data from different equations is given.

❌ The inductive bias is opaque. It is difficult for the user to steer the prior towards a specific class of equations (e.g. polynomials, etc.).

First Learning Based Ideas:

They remind me of Neurosymbolic!

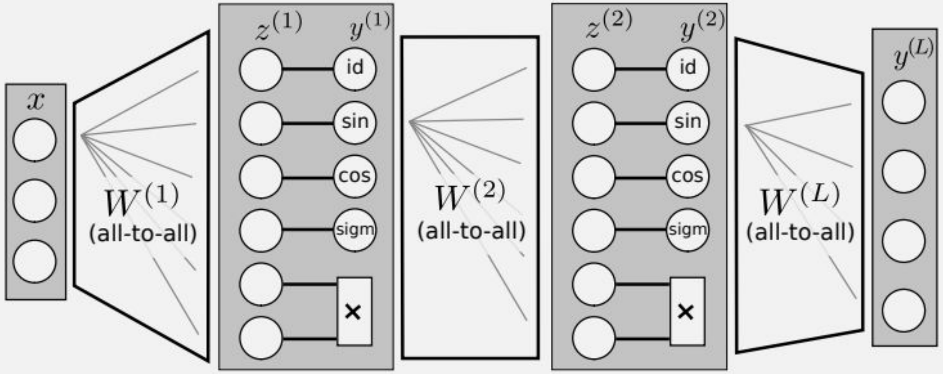
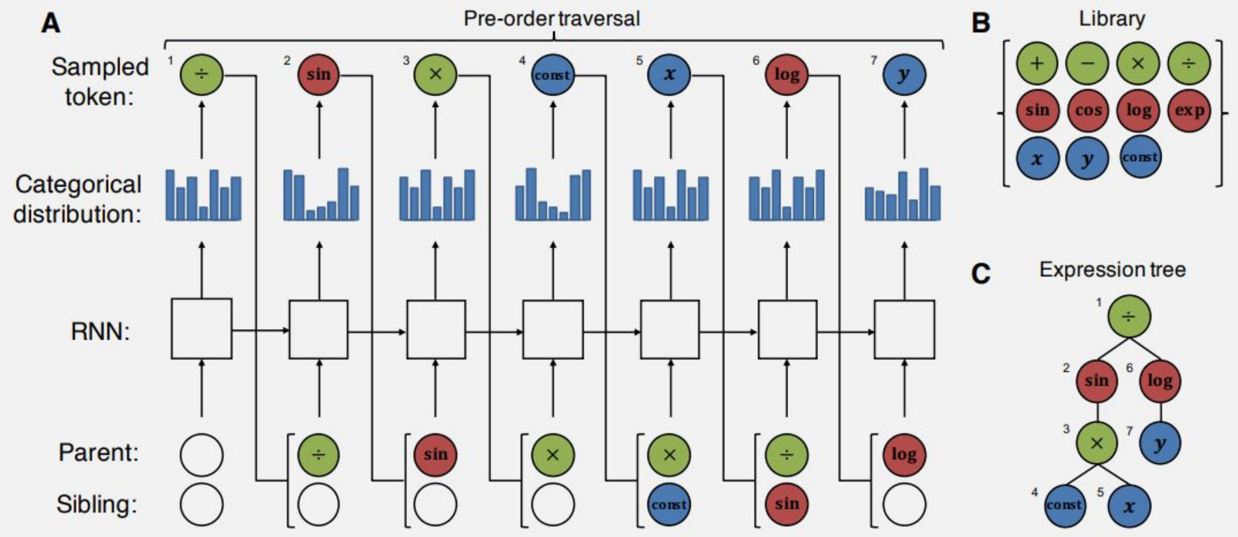


Figure 1: Network architecture of the proposed Equation Learner (EQL) for 3 layers ($L = 3$) and one neuron per type ($u = 4, v = 1$).

Martius, G. and Lampert, C. H. Extrapolation and learning Equations. 2016

More Advanced Ideas, A RNN for a Task

the network has to be retrained from scratch for each new equation



Petersen, B. K. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. International Conference on Learning Representations, 2021

Neural Symbolic Regression that Scales

Luca Biggio, Tommaso Bendinelli, Alexander Neitz,
Aurelien Lucchi, Giambattista Parascandolo
June 2021

Main drawbacks of classic methods

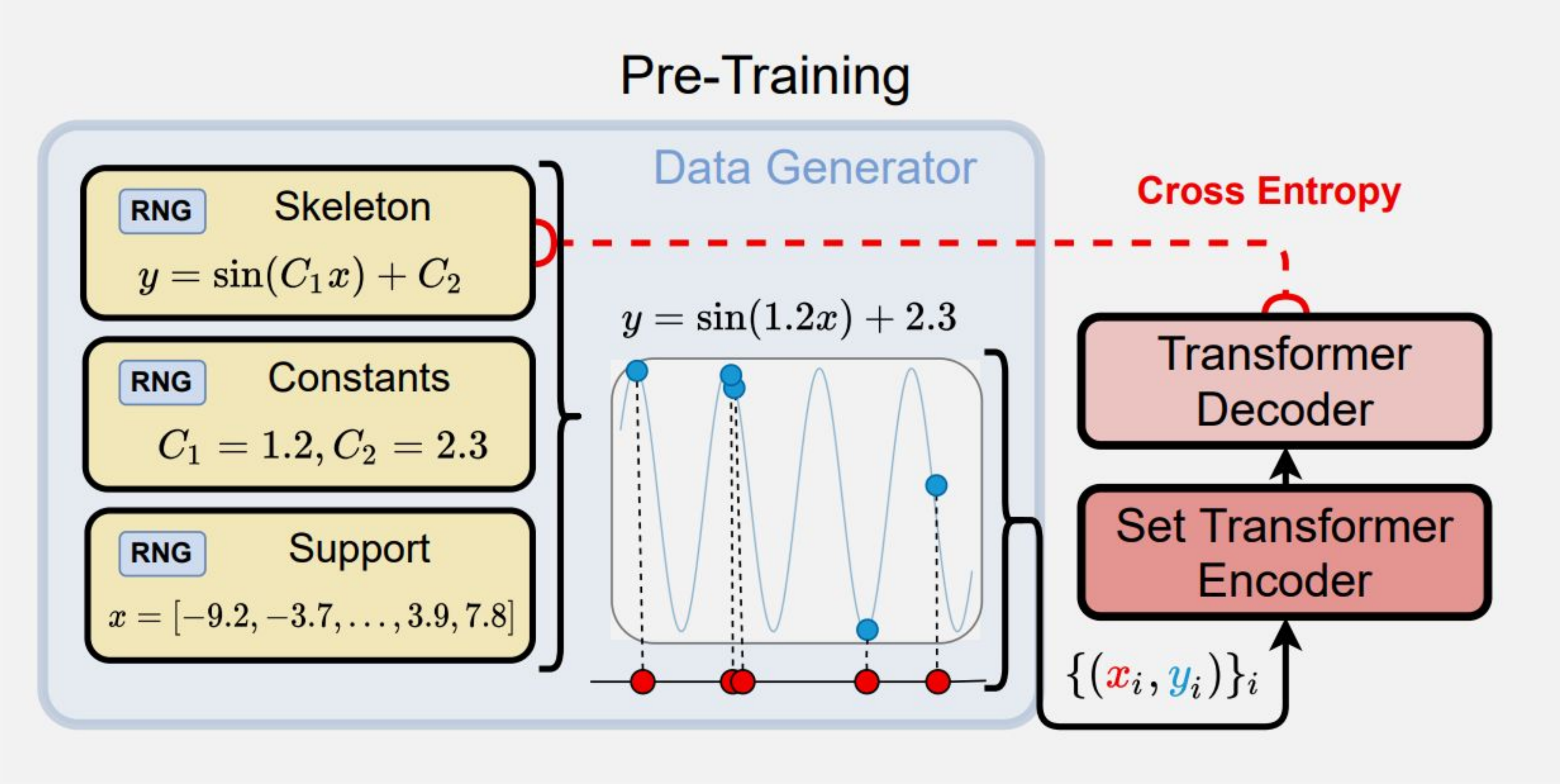
❌ string of symbols grows exponentially with the string length.

❌ They do not improve with experience. As every equation is regressed from scratch, the system does not improve if access to more data from different equations is given.

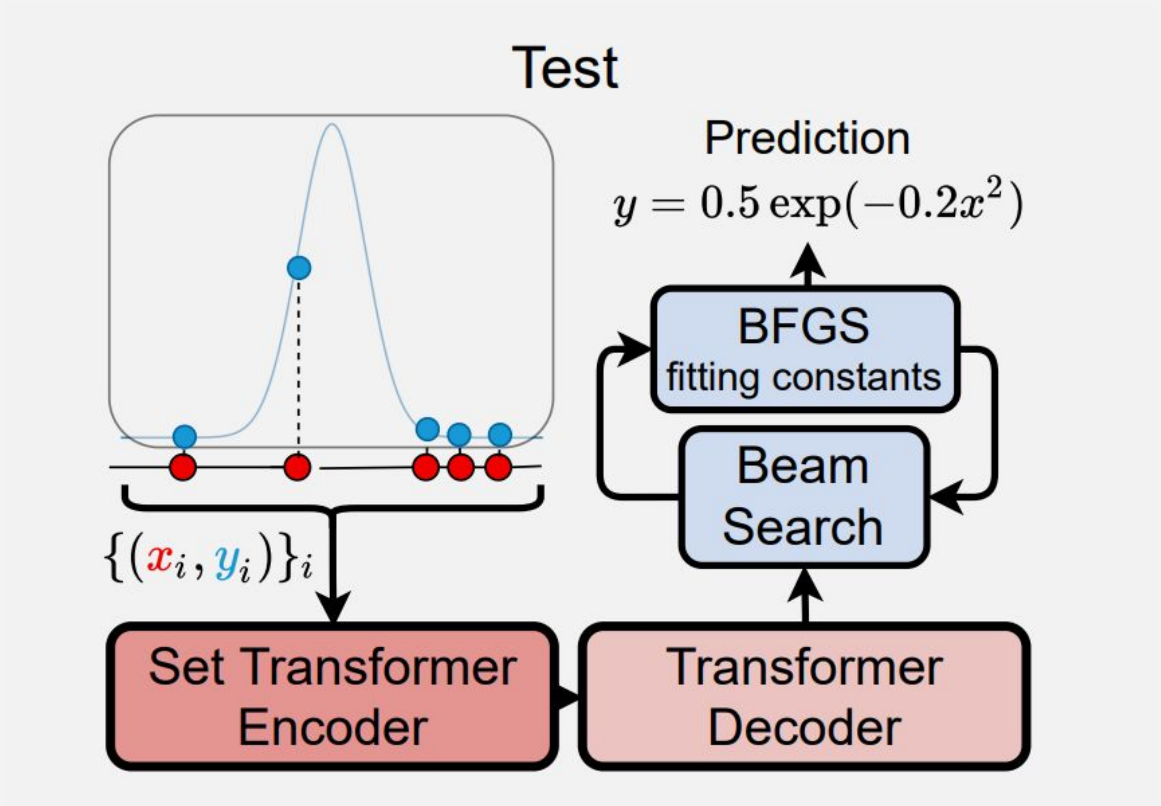
❌ The inductive bias is opaque. It is difficult for the user to steer the prior towards a specific class of equations (e.g. polynomials, etc.).

✔️**we show that a strong symbolic regression can be purely learned from data**

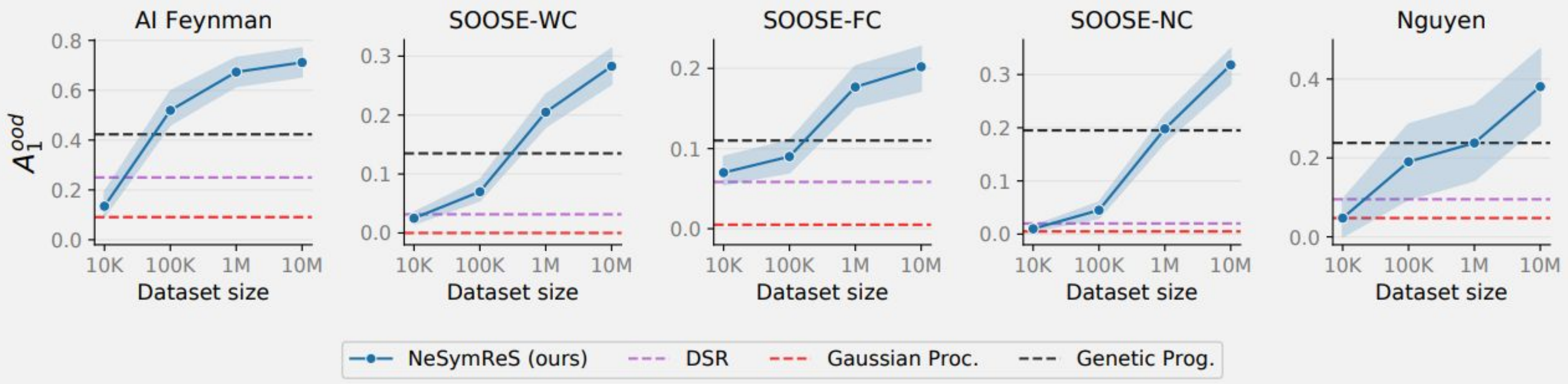
Pre-Training



Test Time


Test
Prediction
$y = 0.5\exp(-0.2x^2)$
BFGS
fitting constants
Beam
Search
$\{(x_i, y_i)\}_i$
Set Transformer
Encoder
Transformer
Decoder

Evaluation, Scale Train Time



Evaluation, Scale Test Time

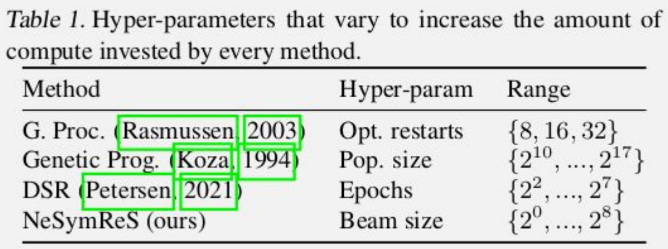


Table 1. Hyper-parameters that vary to increase the amount of compute invested by every method.

Method	Hyper-param	Range
G. Proc. (Rasmussen, 2003)	Opt. restarts	$\{8, 16, 32\}$
Genetic Prog. (Koza, 1994)	Pop. size	$\{2^{10}, ..., 2^{17}\}$
DSR (Petersen, 2021)	Epochs	$\{2^{2}, ..., 2^{7}\}$
NeSymReS (ours)	Beam size	$\{2^{0}, ..., 2^{8}\}$

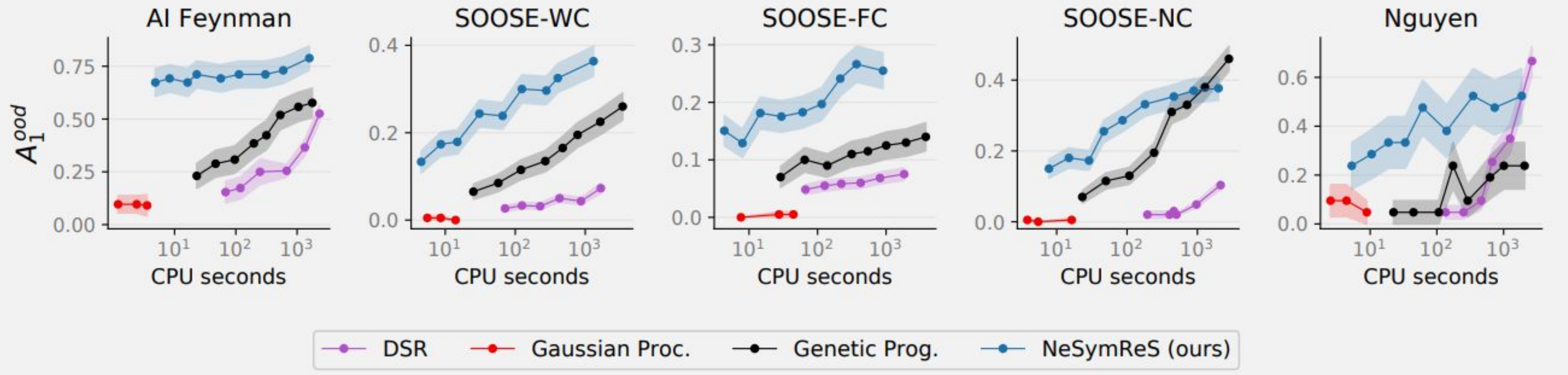
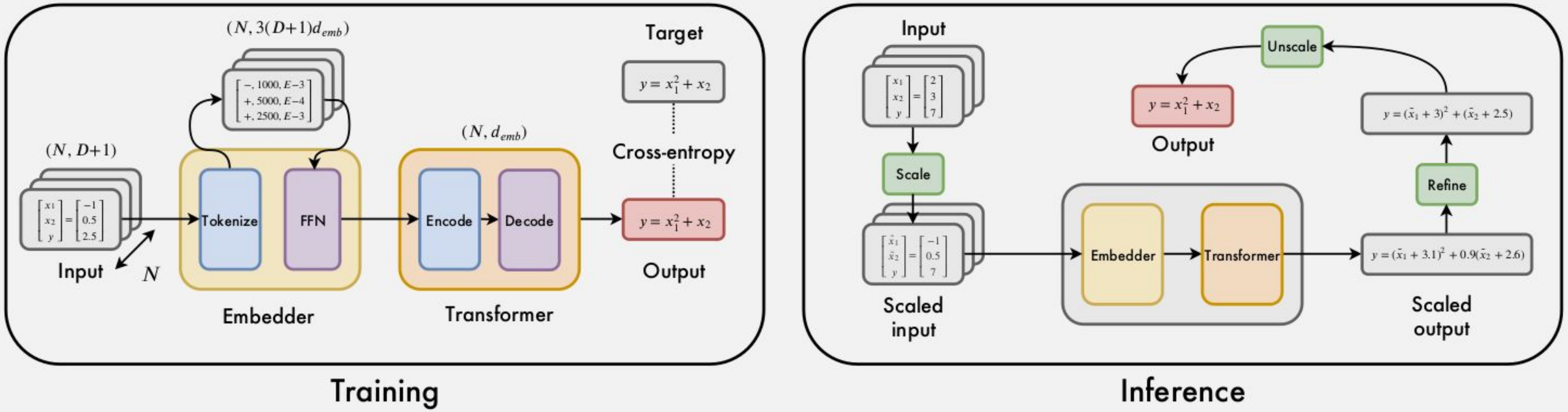


Figure 4. Accuracy out of distribution as a function of time for all methods ran on a single CPU per equation.

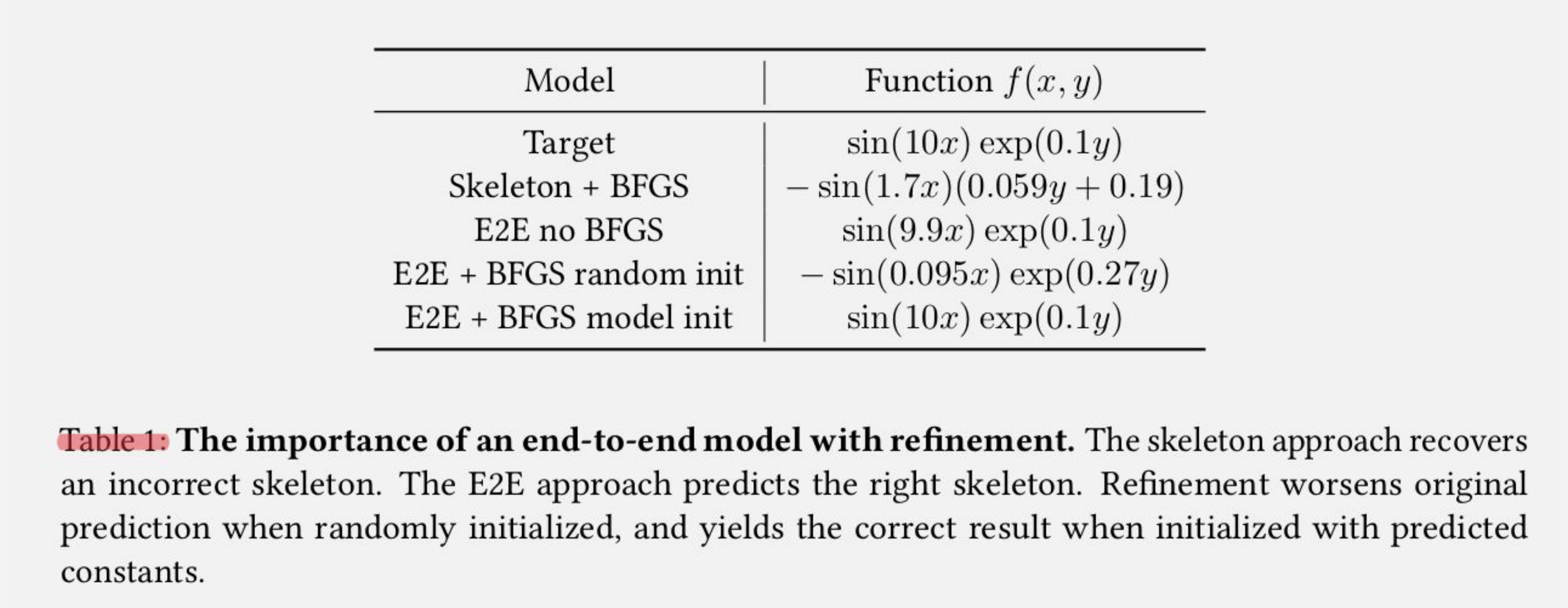
End-to-End Symbolic Regression with Transformers

Pierre-Alexandre Kamienny, Stéphane d'Ascoli, Guillaume Lample, François Charton
April 2022

Learn to Predict End-to-End



Inference tricks



Model	Function $f(x,y)$
Target	$\sin(10x)\exp(0.1y)$
Skeleton + BFGS	$-\sin(1.7x)(0.059y+0.19)$
E2E no BFGS	$\sin(9.9x)\exp(0.1y)$
E2E + BFGS random init	$-\sin(0.095x)\exp(0.27y)$
E2E + BFGS model init	$\sin(10x)\exp(0.1y)$

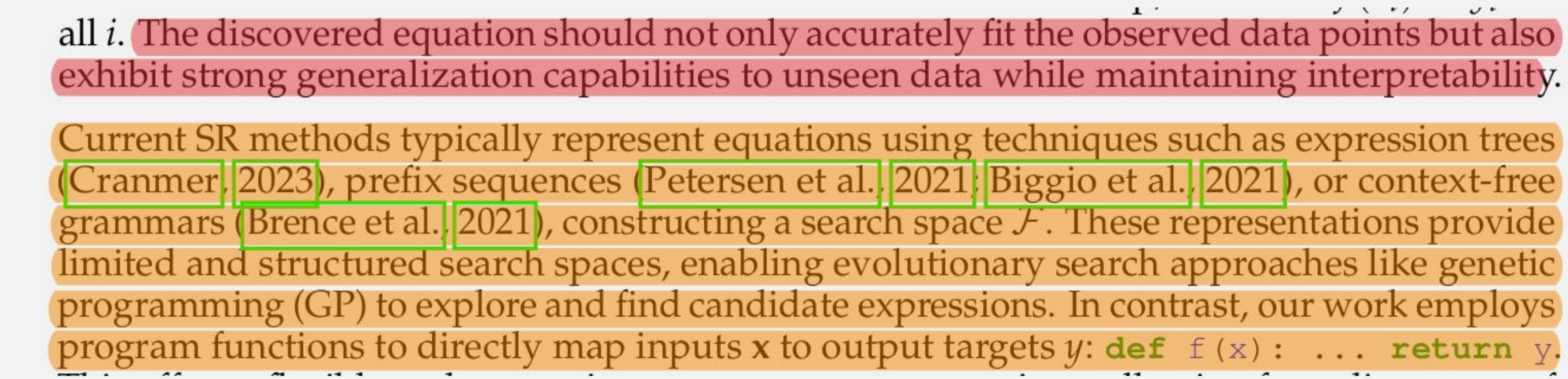
Table 1: **The importance of an end-to-end model with refinement.** The skeleton approach recovers an incorrect skeleton. The E2E approach predicts the right skeleton. Refinement worsens original prediction when randomly initialized, and yields the correct result when initialized with predicted constants.

LLM-SR: Scientific Equation Discovery via Programming with Large Language Models

Parshin Shojaee, Kazem Meidani, Shashank Gupta, Amir Barati Farimani, Chandan K Reddy

April 2024

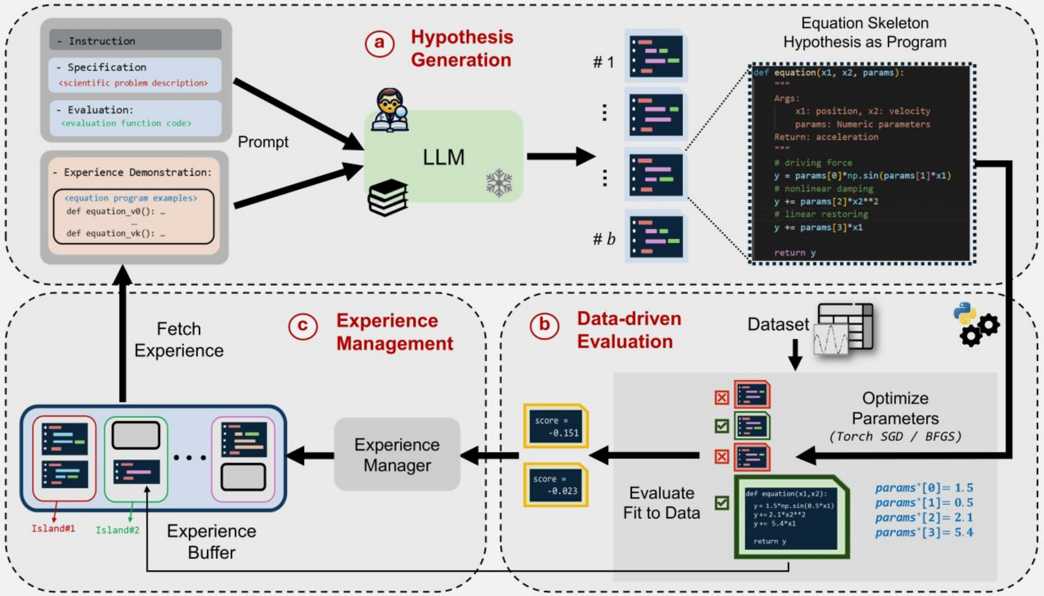
Get Rid of Computation Tree



all i. The discovered equation should not only accurately fit the observed data points but also exhibit strong generalization capabilities to unseen data while maintaining interpretability.

Current SR methods typically represent equations using techniques such as expression trees (Cranmer, 2023), prefix sequences (Petersen et al., 2021; Biggio et al., 2021), or context-free grammars (Brence et al., 2021), constructing a search space $\mathcal{F}$. These representations provide limited and structured search spaces, enabling evolutionary search approaches like genetic programming (GP) to explore and find candidate expressions. In contrast, our work employs program functions to directly map inputs $\mathbf{x}$ to output targets y: `def f(x): ... return y`.

LLM-SR

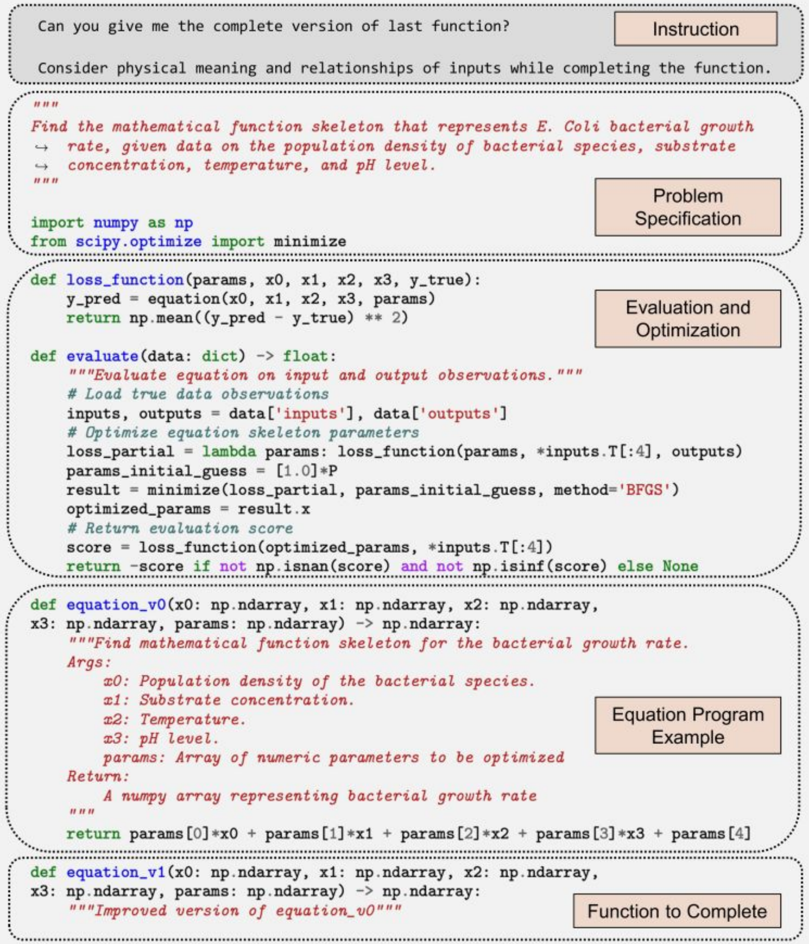


```
def equation(x1, x2, params):
    """
    Args:
        x1: position, x2: velocity
        params: Numeric parameters
    Return: acceleration
    """
    # driving force
    y = params[0]*np.sin(params[1]*x1)
    # nonlinear damping
    y += params[2]*x2**2
    # linear restoring
    y += params[3]*x1

    return y
```

```
def equation(x1,x2):
  y= 1.5*np.sin(0.5*x1)
  y+=2.1*x2**2
  y+= 5.4*x1

  return y
```

Figure 1: **The LLM-SR framework**, consisting of three main steps: **(a) Hypothesis Generation**, where LLM generates equation program skeletons based on a structured prompt; **(b) Data-driven Evaluation**, which optimizes the parameters of each equation skeleton hypothesis and assesses its fit to the data; and **(c) Experience Management**, which maintains a diverse buffer of high-scoring hypotheses to provide informative in-context examples into LLM's prompt for iterative refinement.

LLM-SR, Prompt


Instruction
Problem Specification
Evaluation and Optimization
Equation Program Example
Function to Complete

```
Can you give me the complete version of last function?

Consider physical meaning and relationships of inputs while completing the function.
```

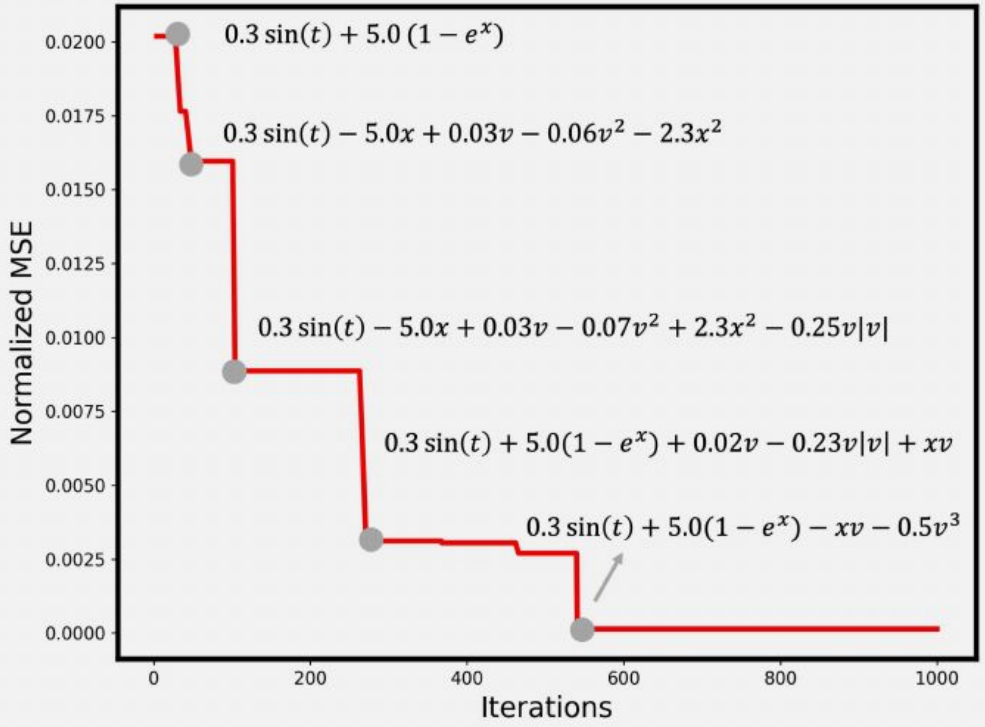
```python
"""
Find the mathematical function skeleton that represents E. Coli bacterial growth
↪  rate, given data on the population density of bacterial species, substrate
↪  concentration, temperature, and pH level.
"""

import numpy as np
from scipy.optimize import minimize
```

```python
def loss_function(params, x0, x1, x2, x3, y_true):
    y_pred = equation(x0, x1, x2, x3, params)
    return np.mean((y_pred - y_true) ** 2)

def evaluate(data: dict) -> float:
    """Evaluate equation on input and output observations."""
    # Load true data observations
    inputs, outputs = data['inputs'], data['outputs']
    # Optimize equation skeleton parameters
    loss_partial = lambda params: loss_function(params, *inputs.T[:4], outputs)
    params_initial_guess = [1.0]*P
    result = minimize(loss_partial, params_initial_guess, method='BFGS')
    optimized_params = result.x
    # Return evaluation score
    score = loss_function(optimized_params, *inputs.T[:4])
    return -score if not np.isnan(score) and not np.isinf(score) else None
```

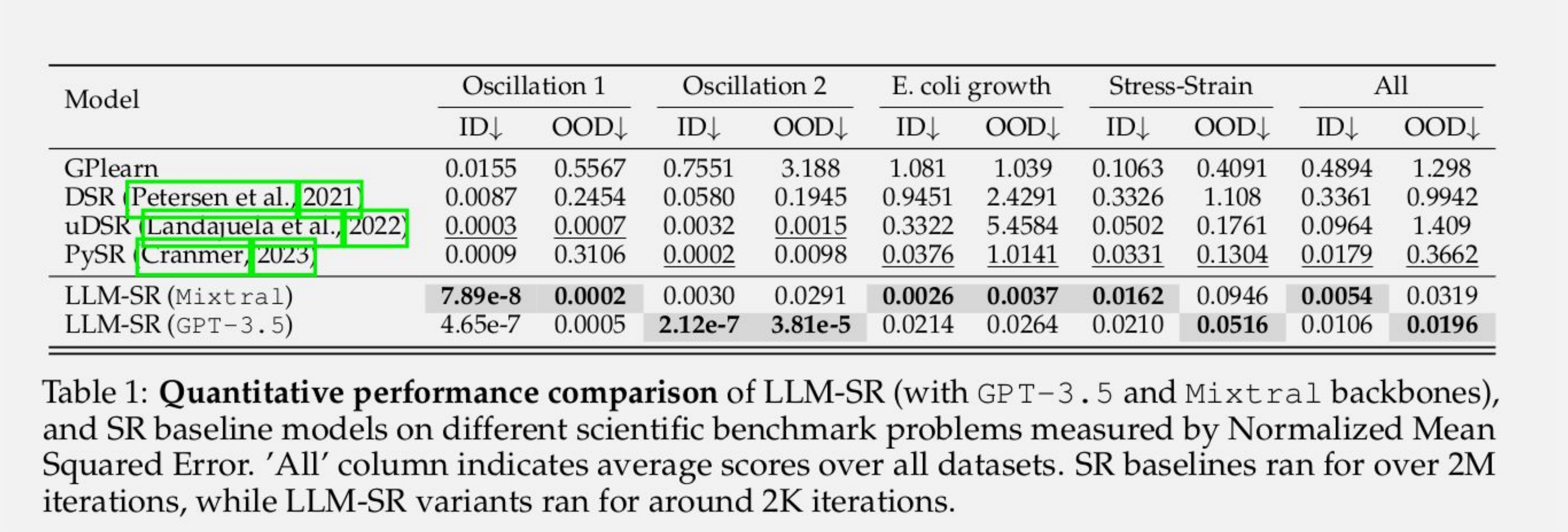
```python
def equation_v0(x0: np.ndarray, x1: np.ndarray, x2: np.ndarray,
x3: np.ndarray, params: np.ndarray) -> np.ndarray:
    """Find mathematical function skeleton for the bacterial growth rate.
    Args:
        x0: Population density of the bacterial species.
        x1: Substrate concentration.
        x2: Temperature.
        x3: pH level.
        params: Array of numeric parameters to be optimized
    Return:
        A numpy array representing bacterial growth rate
    """
    return params[0]*x0 + params[1]*x1 + params[2]*x2 + params[3]*x3 + params[4]
```

```python
def equation_v1(x0: np.ndarray, x1: np.ndarray, x2: np.ndarray,
x3: np.ndarray, params: np.ndarray) -> np.ndarray:
    """Improved version of equation_v0"""
```

LLM-Guided Search



LLM-SR Discovers more Accurate Equations



Model	Oscillation 1		Oscillation 2		E. coli growth		Stress-Strain		All	
	ID↓	OOD↓	ID↓	OOD↓	ID↓	OOD↓	ID↓	OOD↓	ID↓	OOD↓
GPlearn	0.0155	0.5567	0.7551	3.188	1.081	1.039	0.1063	0.4091	0.4894	1.298
DSR (Petersen et al., 2021)	0.0087	0.2454	0.0580	0.1945	0.9451	2.4291	0.3326	1.108	0.3361	0.9942
uDSR (Landajuela et al., 2022)	0.0003	0.0007	0.0032	0.0015	0.3322	5.4584	0.0502	0.1761	0.0964	1.409
PySR (Cranmer, 2023)	0.0009	0.3106	0.0002	0.0098	0.0376	1.0141	0.0331	0.1304	0.0179	0.3662
LLM-SR (`Mixtral`)	**7.89e-8**	**0.0002**	0.0030	0.0291	**0.0026**	**0.0037**	**0.0162**	0.0946	**0.0054**	0.0319
LLM-SR (`GPT-3.5`)	4.65e-7	0.0005	**2.12e-7**	**3.81e-5**	0.0214	0.0264	0.0210	**0.0516**	0.0106	**0.0196**

Table 1: **Quantitative performance comparison** of LLM-SR (with `GPT-3.5` and `Mixtral` backbones), and SR baseline models on different scientific benchmark problems measured by Normalized Mean Squared Error. 'All' column indicates average scores over all datasets. SR baselines ran for over 2M iterations, while LLM-SR variants ran for around 2K iterations.

LLM-SR has better OOD generalization

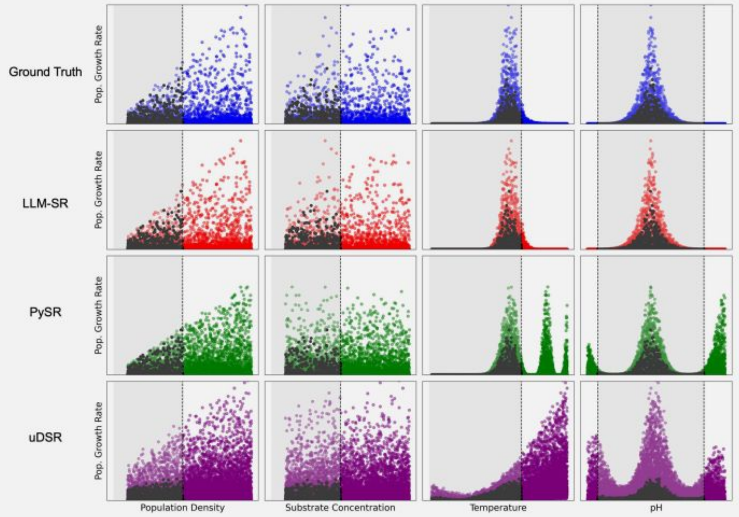


Figure 3: Comparison of E. coli growth rate distributions from LLM-SR, PySR, and uDSR. LLM-SR aligns well with the ground truth, even for OOD data (unshaded regions), demonstrating better generalization than PySR and uDSR, which overfit the ID data (shaded regions and black points).

LLM-SR Discovers Equations More Efficiently

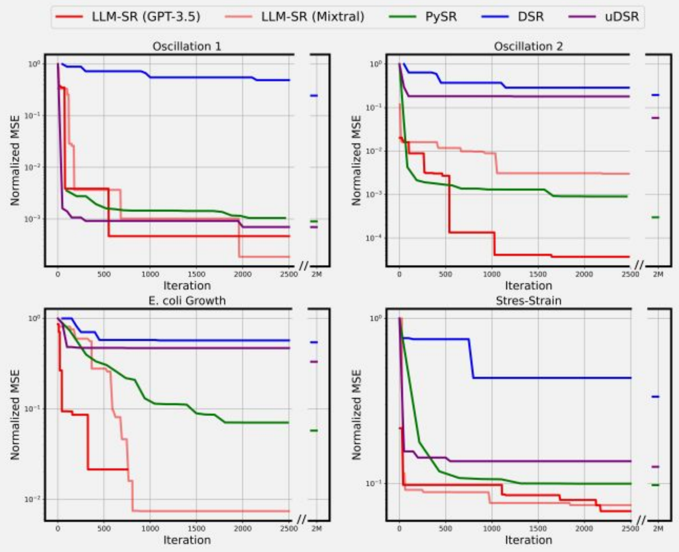


Figure 4: **Best score trajectories** of LLM-SR with GPT-3.5 and Mixtral against SR baselines across different benchmark problems. LLM-SR discovers accurate equations more efficiently, requiring fewer iterations to outperform baselines, which fail to match LLM-SR even after 2M iterations.

Symbolic Regression: Equation as a Program

$$\begin{tikzcd} \text{Learning (System-1)} \arrow[r, bend left, "\text{Learn to Program}"] & \text{Program synthesis (System-2)} \arrow[l, bend left, "\text{Handle Complex Problems}"] \end{tikzcd}$$

Next Session:
Program Synthesis, beyond Regression