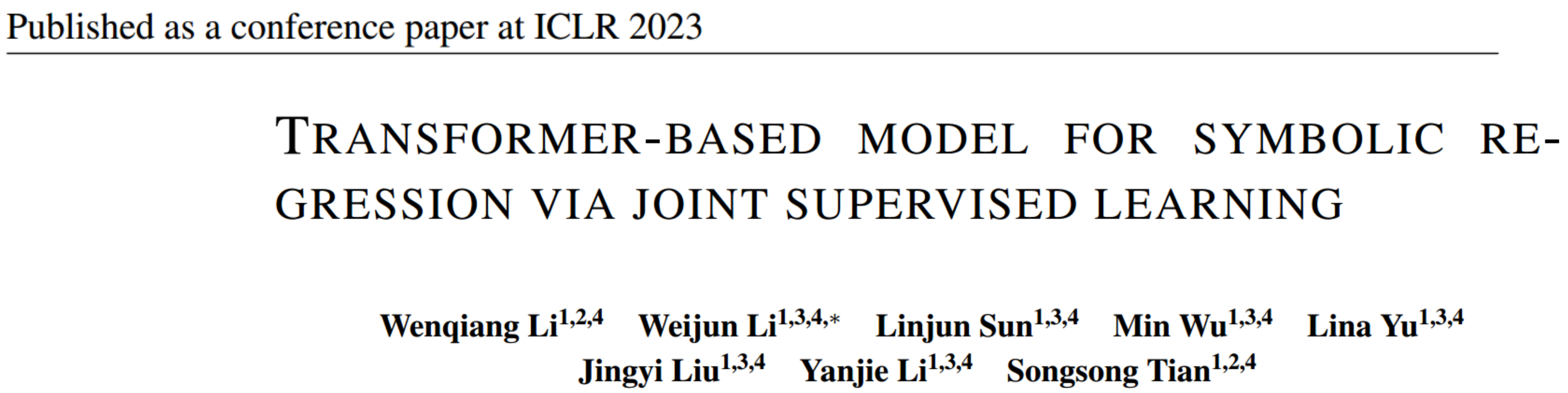


Published as a conference paper at ICLR 2023

TRANSFORMER-BASED MODEL FOR SYMBOLIC REGRESSION VIA JOINT SUPERVISED LEARNING

Wenqiang Li[1,2,4] **Weijun Li**[1,3,4,*] **Linjun Sun**[1,3,4] **Min Wu**[1,3,4] **Lina Yu**[1,3,4]
Jingyi Liu[1,3,4] **Yanjie Li**[1,3,4] **Songsong Tian**[1,2,4]

论文阅读汇报

Transformer-based model for Symbolic Regression via Joint Supervised Learning (ICLR 2023)

石冀

20231219

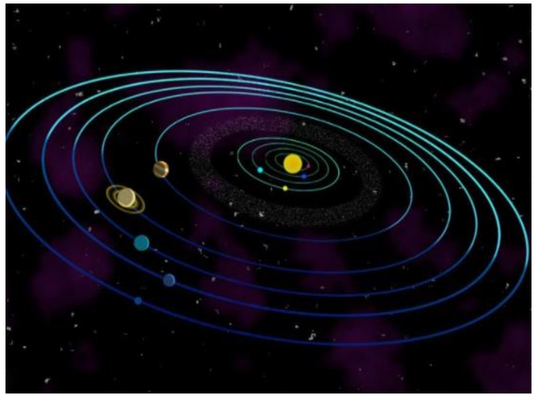
CONTENT

01 Introduction of Symbolic Regression

Introduction of Symbolic Regression

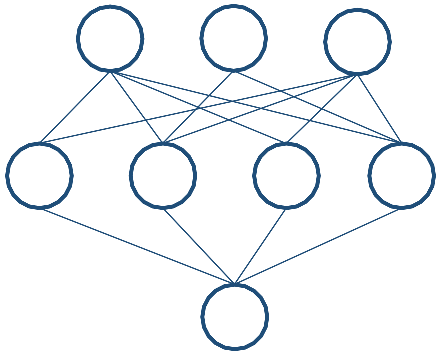
Given a dataset X, y, where each feature $X_i \in \mathbb{R}^{n}$" and target $y_i \in \mathbb{R}$, the goal of symbolic regression is to identify a function f (i.e., $y \approx f(X)$: $\mathbb{R}^n \to \mathbb{R}$) that best fits the dataset, where the functional form of f is a short closed-form mathematical expression.

The law of universal gravitation



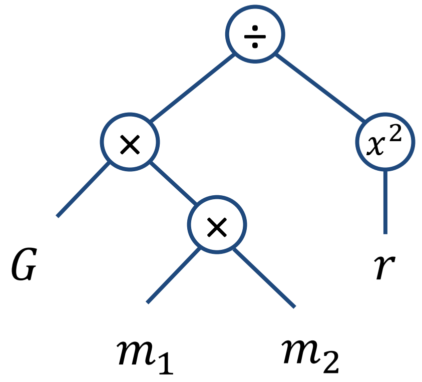
Deep Learning

$$F = Relu(WRelu(WX \ldots) + b)$$



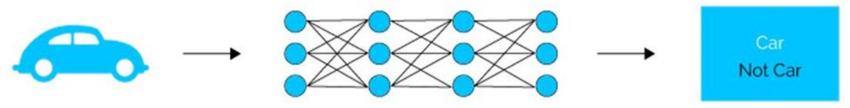
Symbolic Regression

$$F = \frac{Gm_1m_2}{r^2}$$



Introduction of Symbolic Regression

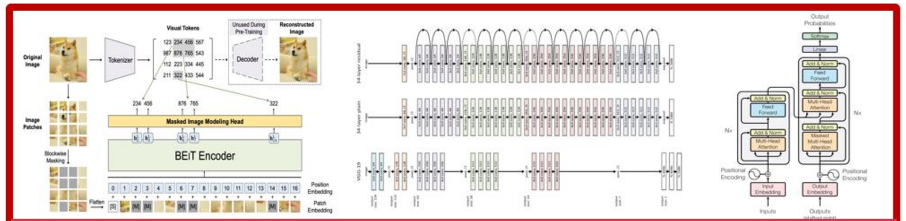
Deep Learning



input   neural network   output

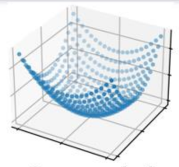
DALL·E 2
DALL·E 2 is a new AI system that can create realistic images and art from a description in natural language.
ChatGPT

Powerful fitting capability



Uninterpretable   Over-fitting risk

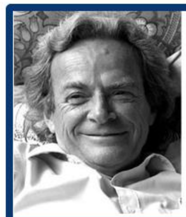
Symbolic Regression



observed data

$$f(x,y) = 2x^{!} + 3y^{!}$$

Mathematical expression

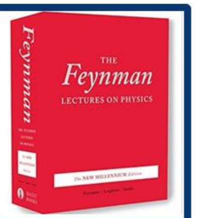


$$L = \frac{\hbar\omega^3}{\pi^2 c^2 (e^{\hbar\omega/k_b T} - 1)}$$

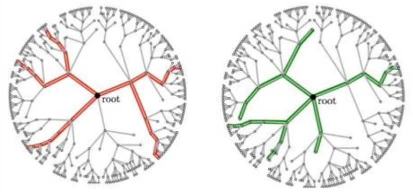
$$F = \frac{G m_1 m_2}{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

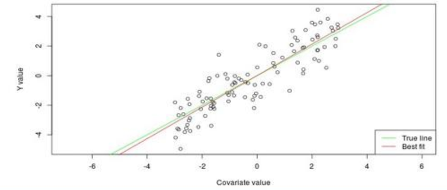
$$r = \frac{a(1 - e^2)}{1 + e\cos(\theta_1 - \theta_2)}$$

$$\frac{d\sigma}{d\cos\theta} = \frac{\pi\alpha^2\hbar^2}{m^2c^2}\left(\frac{\omega'}{\omega}\right)^2\left(\frac{\omega'}{\omega} + \frac{\omega}{\omega'} - \sin^2\theta\right)$$



Interpretable   Good generalization





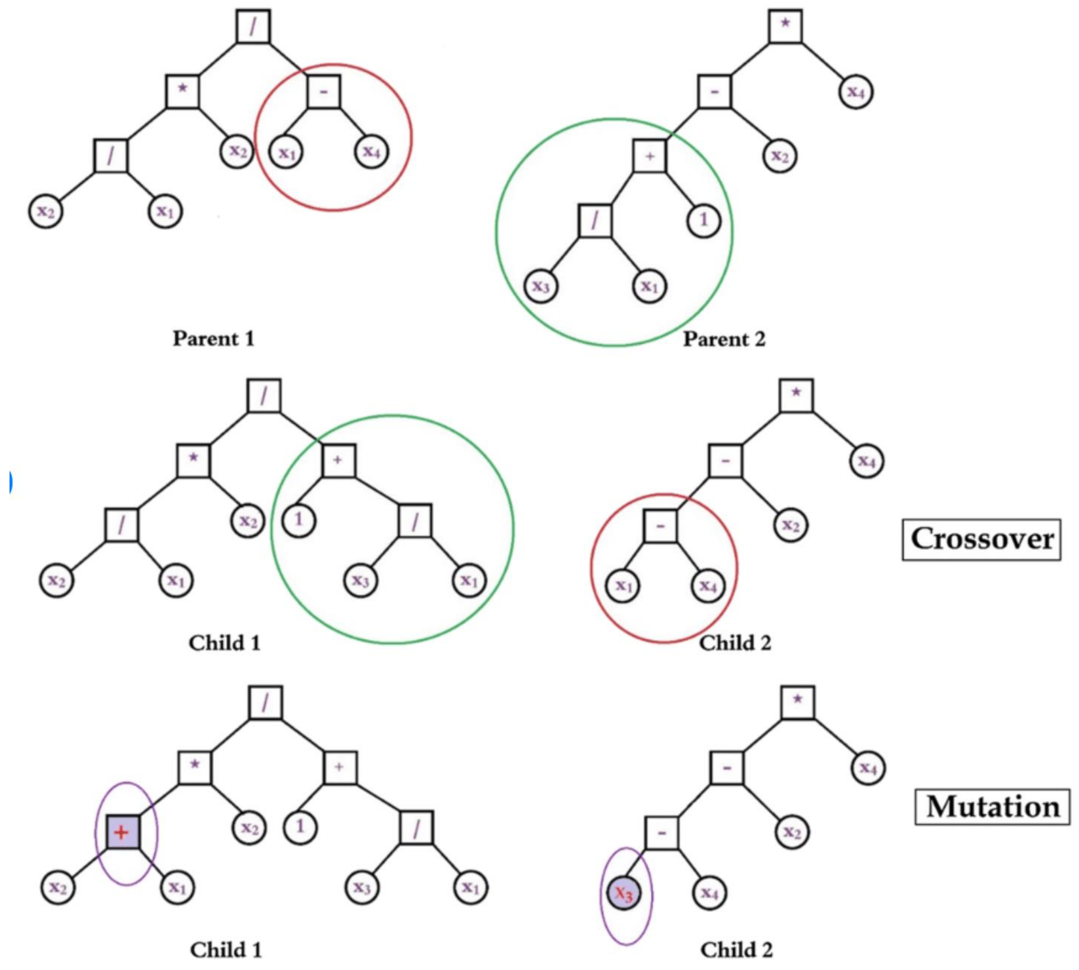
large search space   Inefficient solving

02 Related Works

Related Works

Genetic Programming (GP)

- First and most common approach
- Expression it yields is complex
- Computationally expensive
- High sensitivity to hyperparameters

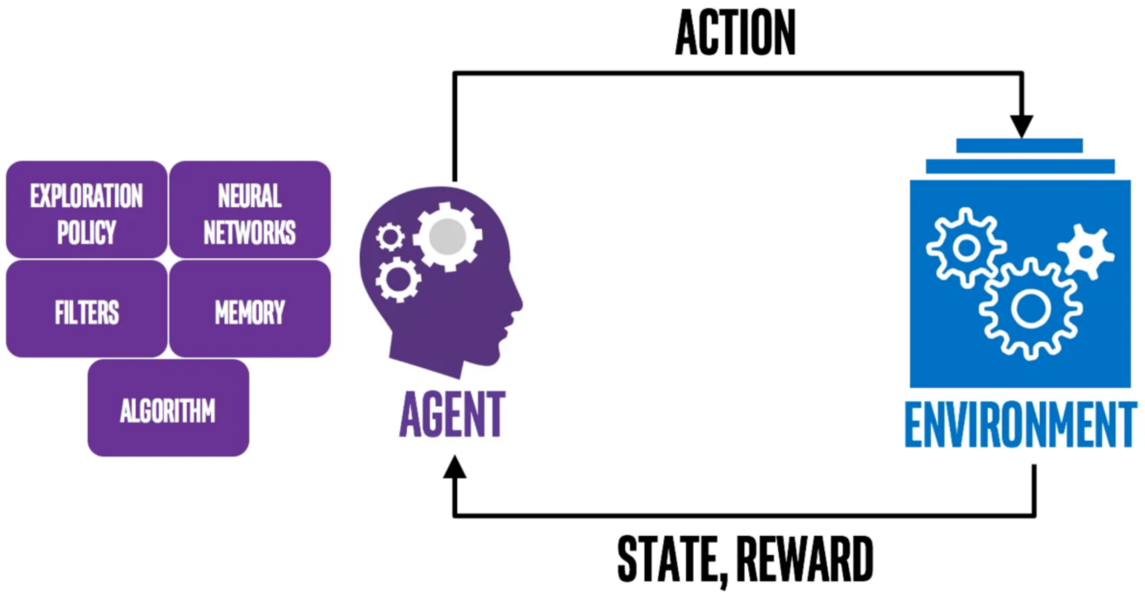


Related Works

NN-based methods, especially Reinforcement Learning (RL)

- Above shortcomings are basically solved
- Handle symbolic regression as an instance-based problem
- Unable to incorporate past experiences

DSO (Petersen et al., 2021)



Related Works

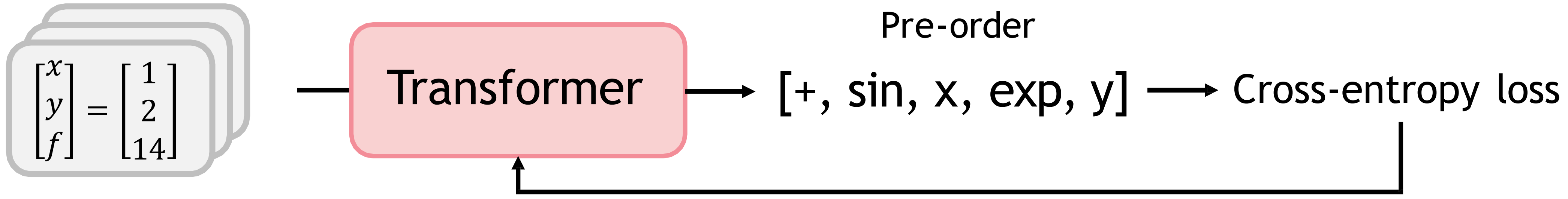
Traditional Transformer-based methods

- DO NOT be trained from scratch
- Low-quality feature extraction from data points
- Skeletons provide ill-defined supervision

SymbolicGPT [Valipour et al., 2021]

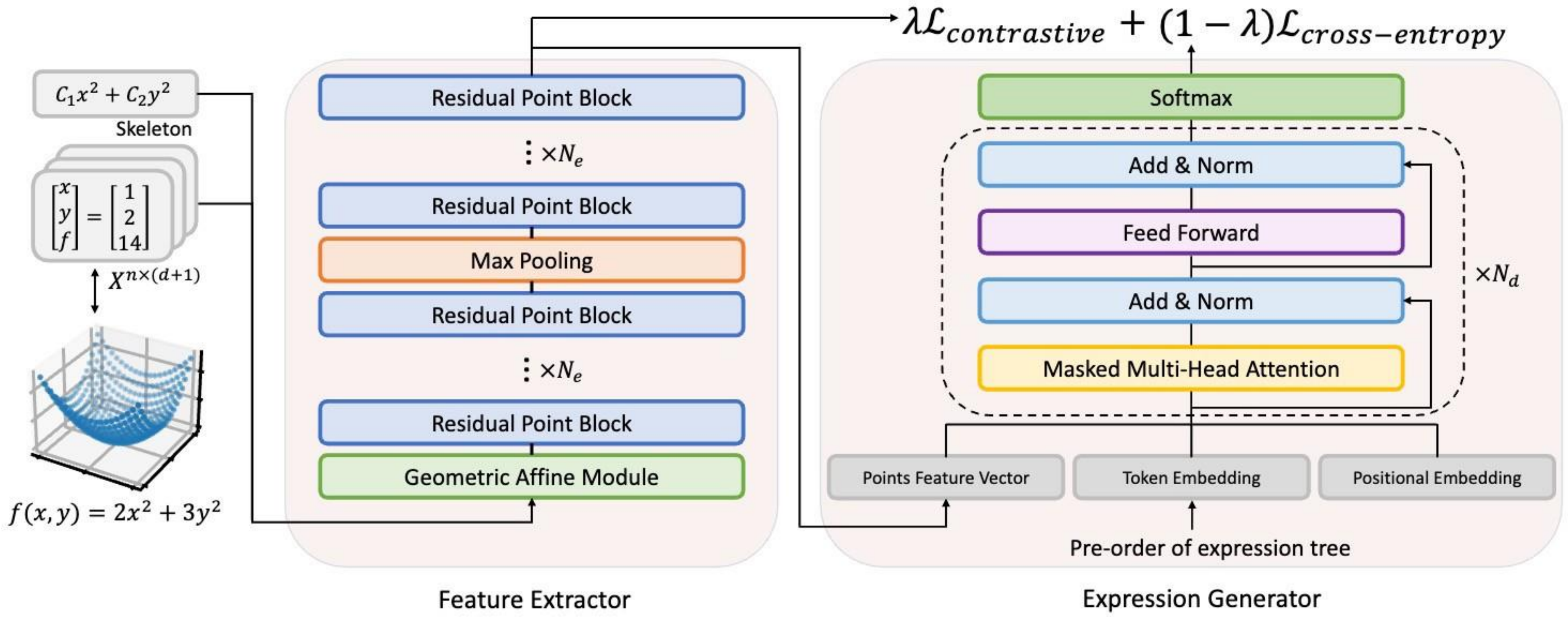
NeSymRes [Biggio et al., 2021]

1. Encode data points
2. Predict the pre-order traversal
3. Compute cross-entropy loss


$\begin{bmatrix} x \\ y \\ f \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 14 \end{bmatrix}$
Transformer
Pre-order
[+, sin, x, exp, y]
Cross-entropy loss

03 Architecture & Methods

Architecture & Methods



Architecture & Methods

1.Feature Extractor: PointMLP (Ma et al., 2022)

$$D=\{(x_i,y_i)\}_{i=1}^{n}\in\mathbf{R}^{n\times(d+1)}$$

$$O_i=POS(MaxPool(PRE(f_{i,j}),|\,i=1,\ldots,N_s;\ j=1,\ldots,K))$$

- N_s points are re-sampled by the farthest point sampling (FPS) algorithm in the s stage
- Using KNN algorithm to find K-nearest neighbors for local information
- POS/PRE are residual Point MLP blocks

$$\{f_{i,j}\}=\alpha\odot\frac{\{f_{i,j}\}-f_i}{\sigma+\varepsilon}+\beta,\sigma=\sqrt{\frac{1}{k\times n\times d}\sum_{i=1}^{n}\sum_{j=1}^{k}(f_{i,j}-f_i)^2}$$

- Applying lightweight geometric affine to transform the dataset to a Gaussian distribution

Architecture & Methods

2.Training with Joint Supervision Information

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

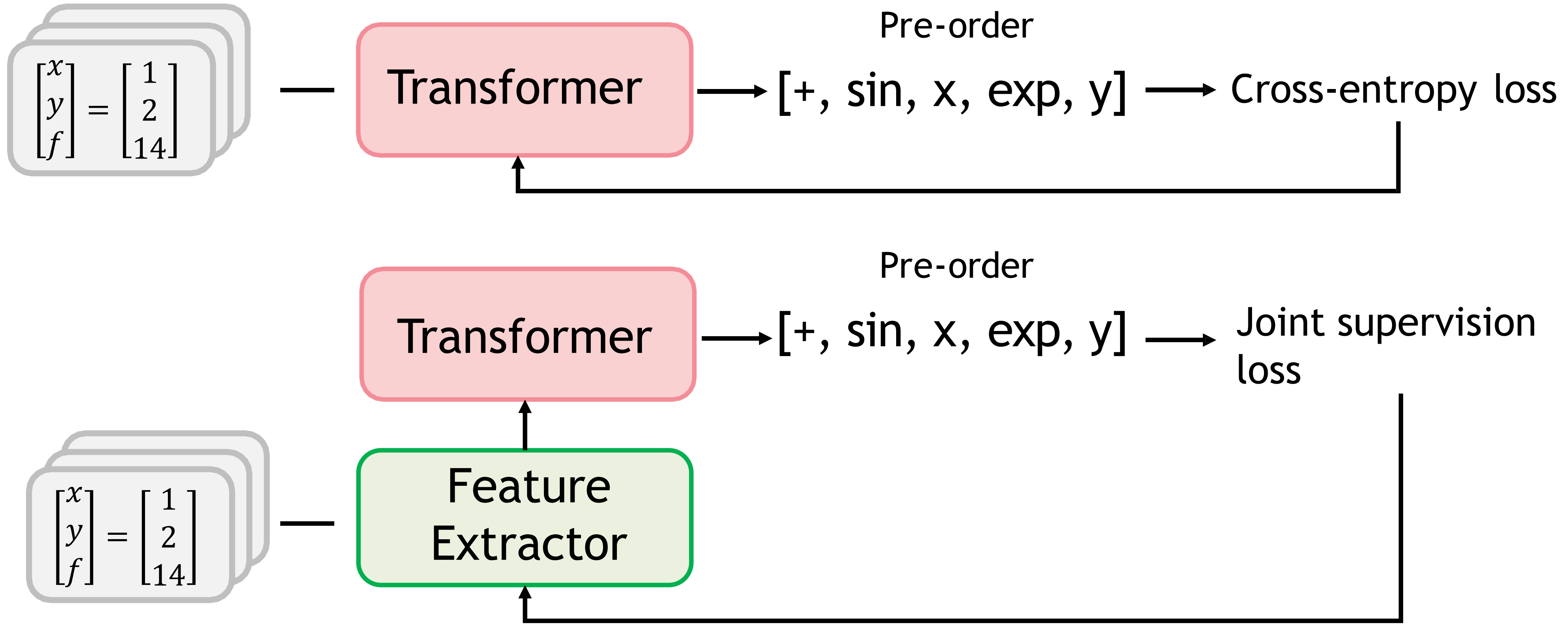
$$L_{CE} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c} \cdot \ln y_{i,c}$$

$$L_{CL} = -\sum_{i=1}^{N}\frac{1}{N_{l_i}+\varepsilon}\sum_{j=1}^{N} 1_{i\neq j}1_{l_i=l_j} \ln \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{N} 1_{i\neq k}1_{l_i\neq l_k} \exp(s_{i,k}/\tau)}$$

$$s_{i,j} = \frac{\overrightarrow{v_i} \cdot \overrightarrow{v_j}}{\left\|v_i\right\| \cdot \left\|v_j\right\|}$$

Architecture & Methods

3.Difference from Traditional models



$\begin{bmatrix} x \\ y \\ f \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 14 \end{bmatrix}$
Transformer
Pre-order
[+, sin, x, exp, y]
Cross-entropy loss
Transformer
Pre-order
[+, sin, x, exp, y]
Joint supervision loss
$\begin{bmatrix} x \\ y \\ f \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 14 \end{bmatrix}$
Feature Extractor

04 Experiments and Results

Experiments and Results

1.Generating Datasets

- Given a prior probability distribution of operators and operands, generate 100,000 unique symbol skeletons with fixed probability. For each symbol skeleton, vary the constant C value 10,20,30,40,50 times; choose up to 2 independent variables 3 constants as the operation
- $X_i \in [-10,10], C \in [-2,2]$
- 4 NVIDIA V100 GPUs
- Adam optimizer

Experiments and Results

2.Feature Extraction Performance

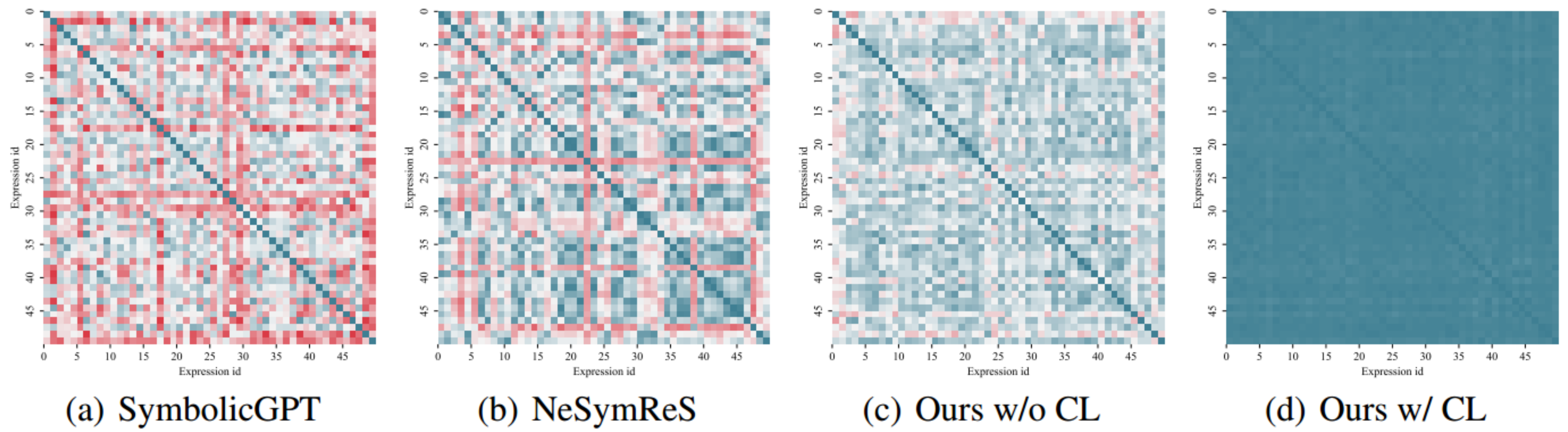


Figure 2: For the expression skeleton $c_1 \sin(x_1) + c_2 \cos(x_2) + c_3$, four heat maps of cosine similarity between the fifty different feature vectors from different methods, where the redder color means the cosine similarity is closer to 0, and the greener color means the cosine similarity is closer to 1.

Experiments and Results

3.General Experiments

- R^2 fitting accuracy
- Different training sizes
- Gaussian noisy data
- Different BFGS restart times
- Out-of-domain performance
- Finding mathematically equivalent expressions

Table 2: Results comparing our method with CL with state-of-the-art methods on several benchmarks. Our method, SymbolicGPT, and NeSymReS all use the beam search strategy with the beam size equaling 128. We report the average value of R^2 for each benchmark.

	Ours	SymbolicGPT	NeSymReS	DSO	GP
Benchmark	$R^2 \uparrow$	$R^2 \uparrow$	$R^2 \uparrow$	$R^2 \uparrow$	$R^2 \uparrow$
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

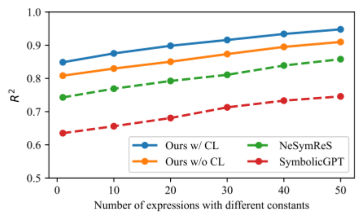


Figure 3: Training on different datasets that contain various numbers of expressions with different constants. Inference on SSDNC benchmark.

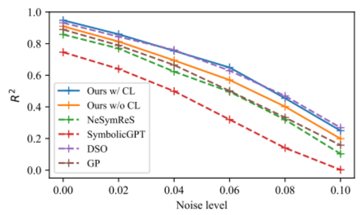


Figure 4: R^2 vs gaussian noisy data. Error bar represent standard error. Inference on SSDNC benchmark.

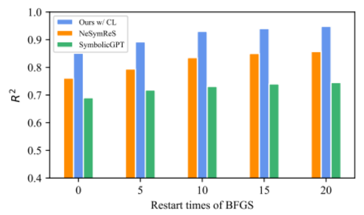


Figure 5: R^2 for different restart times of BFGS in the constant optimization stage. Inference on SSDNC benchmark.

The End