Discovering Mathematical Expressions Through DeepSymNet: A Classification-Based Symbolic Regression Framework

Min Wu, Weijun Li, *Senior Member, IEEE*, Lina Yu, *Member, IEEE*, Linjun Sun, *Member, IEEE*, Jingyi Liu, and Wenqiang Li, *Graduate Student Member, IEEE*

***Abstract*— Symbolic regression (SR) is the process of finding an unknown mathematical expression given the input and output and has important applications in interpretable machine learning and knowledge discovery. The major difficulty of SR is that finding the expression structure is an NP-hard problem, which makes the entire process time-consuming. In this study, the solution of expression structures was regarded as a classification problem and solved by supervised learning such that SR can be solved quickly by using the solving experience. Techniques for classification tasks, such as equivalent label merging and sample balance, were used to enhance the robustness of the algorithm. We proposed a symbolic network called DeepSymNet to represent symbolic expressions to improve the performance of the algorithm. DeepSymNet has been proven to have a strong representation ability with a shorter label compared to the current popular representation methods, reducing the search space when predicting. Moreover, DeepSymNet conveniently decomposes SR into two smaller subproblems, which makes solving the problem easier. The proposed algorithm was tested on artificially generated expressions and public datasets and compared with other algorithms. The results demonstrate the effectiveness of the proposed algorithm.**

***Index Terms*— AI for science, deep learning, symbolic network, symbolic regression (SR).**

I. Introduction

BUILDING mathematical models through hypothesis deduction and mathematical symbols have been an important means for scientists to understand the world and discover physics laws. It plays an important role in simulating, interpreting, and predicting system behavior and decision control. If mathematical expressions can be used to represent data and their relationships in production and daily life, a new model with interpretable and strong generalization ability will be obtained. A core issue that needs to be addressed for this is symbolic regression (SR). SR is a challenging data modeling task in academia that aims to discover a mathematical expression that best characterizes the hidden relationship between input and output data. With the natural ability to interpret and generalize, SR mathematical expressions can be used to explain the causal mechanisms between variables or predict the trends of complex systems. Therefore, SR is considered a promising method to discover knowledge from acquired observations and shows great potential in various fields, such as astronomy [1], physics [2], and machine learning [3].

In general, the mathematical expression (symbolic expression, expression) $y = f(X)$ is composed of independent variables $(x_1, x_2, \ldots, x_n)$, operators $(+, -, \times, \ldots, \sin, \cos, \exp, \ldots)$, and various coefficients. SR methods focus on obtaining the best combination of these elements and solving the most appropriate coefficients, given the independent variable X and dependent variable y. Unfortunately, obtaining the best combination is an NP-hard problem because the combinatorial space exponentially grows with the length of symbolic expressions. Moreover, the nonlinear solving process of the coefficients and the process of element combination optimization are perturbed by each other. Therefore, determining the exact expression is difficult. Traditional SR algorithms are based on genetic programming (GP) [4], [5]. GP-based algorithms use a symbolic tree (binary tree) to represent expressions and evolutionary computation to search for objective expressions that satisfy the requirements. They are sufficiently time-consuming because the search space is large and they cannot learn from the solving experience.

Recently, the application of deep learning [6] in various fields of scientific research has achieved great success, such as AlphaGo [7] playing Go, AlphaFold [8] predicting protein structure and deep learning solving mathematical problems [9], [10], [11], which are called "AI for science" [12]. Deep learning, especially supervised learning, makes it possible to gain the historical experience of solving SR, thereby speeding up solving when new tasks are encountered. Therefore, it is feasible to solve the SR with supervised learning.

Manuscript received 28 September 2022; revised 6 June 2023 and 5 September 2023; accepted 8 November 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 92370117, in part by CAS Project for Young Scientists in Basic Research under Grant YSBR-090, and in part by the Key Research Program of the Chinese Academy of Sciences under Grant XDPB22. *(Corresponding author: Weijun Li.)*

The authors are with the AnnLab, Institute of Semiconductors, Chinese Academy of Sciences, Beijing 100083, China, also with the Center of Materials Science and Optoelectronics Engineering and the School of Microelectronics, University of Chinese Academy of Sciences, Beijing 100049, China, and also with the Beijing Key Laboratory of Semiconductor Neural Network Intelligent Sensing and Computing Technology, Beijing 100083, China (e-mail: wumin@semi.ac.cn; wjli@semi.ac.cn; yulina@semi.ac.cn; sunlinjun@semi.ac.cn; liujingyi@semi.ac.cn; liwenqiang@semi.ac.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2023.3332400.

Digital Object Identifier 10.1109/TNNLS.2023.3332400

To date, several SR algorithms based on supervised learning have been proposed. The SymbolicGPT [13] algorithm regards the expression string as a language, thereby transforming SR into a translation task and using the language model in deep learning for training. The NeSymReS [14] algorithm performs a preorder traversal of the symbolic tree to obtain a sequence. Consequently, the SR is converted into a sequence generation task, and a sequence generation model in deep learning is used for training. The E2E [15] algorithm is an end-to-end supervised learning-based SR algorithm that encodes expression structures and coefficients into labels for training. However, these algorithms are not efficient in representing expressions, and their labels are relatively long, increasing the prediction difficulty. They also have the problem of multiple equivalent labels, causing "ambiguity" during training. Moreover, they did not balance the number of training samples, which affected the robustness of the algorithm.

This study aims to explore more efficient representations of symbolic expressions with shorter labels and to propose a more robust supervised learning-based SR method. In this study, a new symbolic network is used to represent symbolic expressions, called DeepSymNet, and can represent any symbolic expression. DeepSymNet is further encoded into a sequence, and the solution of the expression structure is modeled as a sequence generation and classification task. Supervised training is performed to predict the structure of an unknown expression. Then, a nonlinear optimization algorithm is used to solve the constant coefficients to obtain the final symbolic expression.

The main contributions of this study are summarized as follows.

1) DeepSymNet has a stronger representation ability and shorter label than the current popular representation methods, which can improve the algorithm performance.
2) Solving the symbolic expression structure is treated as a classification problem. Thus, the techniques in classification problems can be used to enhance the robustness of the algorithm. To the best of our knowledge, this is the first study to treat SR as a classification task, offering a fresh perspective on supervised learning-based SR methods.
3) DeepSymNet is convenient for decomposing the SR into two smaller subproblems, which makes solving SR easier.

The remainder of this article is organized as follows. Section II presents a review of the existing methods on SR. Section III describes the design of DeepSymNet. Section IV introduces an SR algorithm using DeepSymNet. Section V compares the performance of the proposed algorithm with that of state-of-the-art algorithms and discusses the algorithm proposed in this study. Finally, the conclusion of the study and the summary of the proposed algorithm are presented in Section VI.

II. Related Works

As previously mentioned, SR includes two processes: solving the expression structure and solving the coefficients. However, we need a reasonable representation of the symbolic expression before solving it, which is beneficial for solving the SR. In this section, we review related works regarding these tasks.

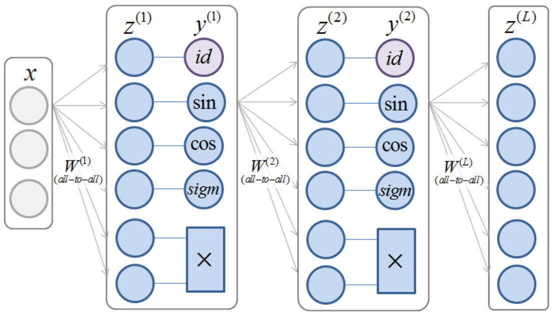

Fig. 1. EQL network structure [18].

A. Representation of Symbolic Expressions

Symbolic tree (binary tree) [5] is the most commonly used representation in SR. Several SR algorithms, such as GP [5], DSR [16], and NeSymReS [14] AIFeynman [2], [17], use symbolic trees to represent mathematical expressions. In a symbolic tree, each intermediate and leaf node represents an operator and a variable or constant, respectively. The number of children in each intermediate node depends on the operand number of operators.

In addition to symbolic trees, another representation constructs a large network to represent symbolic expressions. An EQL network is a representative example of a network [18]. EQL adopts the architecture of a feed-forward neural network, replacing activation functions with operators (Fig. 1), where the operator id represents the identity operation (i.e., $\mathrm{id}(x) = x$).

However, these two representations have certain disadvantages. One disadvantage of symbolic trees is that they are not efficient for representing symbolic expressions because, for the same module that appears multiple times in an expression, a symbolic tree needs to copy it multiple times. For the EQL network, there is only one id operator (see Fig. 1) per layer, causing an insufficient representation ability. For example, the simple expression "$x_1 \sin(x_1) + x_2$" cannot be represented by an EQL network.

B. Solving Symbolic Expression Structures

The solutions of a symbolic expression structure can be divided into search-based methods and supervised learning-based methods.

1) Search-Based Methods: Search-based methods search for expression structures in the solution space.

The most classic search-based method is the GP algorithm [4], [5].[1] The GP algorithm represents expressions by using a symbolic tree. First, many expressions are randomly obtained as the initial population, the evolution is carried out through replication, exchange, and mutation, and the offspring with smaller fitness are selected to continue to evolve until the expression meets the requirements of fitness.

Among search-based methods, there is an important class that uses reinforcement learning to search for suitable

[1] https://github.com/trevorstephens/gplearn

expression structures. One representative method is the DSR algorithm [16]. DSR encodes a symbolic tree as a sequence and solves it using the policy gradient method in deep reinforcement learning [19]. The policy network calculates the occurrence probability of each element in the sequence and performs Monte Carlo sampling according to these probabilities. The idea behind DSR is to increase the sampling probability of expressions with a large reward, thereby producing expressions with smaller errors.

AIFeynman [2], [17] is an SR algorithm used for physical formulas. It uses prior knowledge in physics, such as dimensional analysis, symmetry analysis, and separability analysis, to judge the structure of the expression to decompose the expression into smaller subproblems and reduce the search space.

There is also a class of methods based on sparse optimization whose representative algorithm is EQL [18]. The EQL algorithm uses the BP algorithm [19] combined with sparse optimization to learn parameters to obtain a sparse subnetwork in the EQL network, thereby obtaining the mathematical expression structure.

The common disadvantage of search-based methods is that they are slow because the search space is large and the experience of solving cannot be reused. Additionally, each has its own shortcomings. The expression obtained using GP is often complicated to achieve a smaller fitting error. The DSR policy network does not utilize the data of independent and dependent variables. AIFeynman [2] utilizes a large amount of prior knowledge; therefore, it cannot perform general SR. It is difficult for the EQL algorithm to obtain simple expressions through sparse optimization. Therefore, the obtained expressions are too complex and lose interpretability.

2) Supervised Learning-Based Methods: Supervised learning-based methods have been proposed, which use historical experience when solving a new problem, to overcome the disadvantages of time-consuming search-based methods. Representative methods include SymbolicGPT [13], NeSymReS [14], and E2E [15].

The SymbolicGPT method encodes the symbol expression as a string and treats the expression structure solution as a language-translation task. The GPT model [20] in language translation is used for supervised training with a large number of artificially generated samples.

The NeSymReS method encodes the symbolic tree as a sequence by preorder traversal, and the set Transformer [21] is used for training.

E2E [15] encodes the expression structure and coefficients into labels for training, thereby simultaneously predicting the expression structure and coefficients.

These methods have the problems of multiple equivalent labels and imbalanced training samples, which cause ambiguity during training and affect the robustness of the algorithm. In addition, they also have other disadvantages. The expressions considered by the SymbolicGPT [13] algorithm are relatively simple because the symbolic tree used for sampling has at most four levels. The E2E [15] method encodes the coefficients into labels, which makes the labels very long and affects prediction accuracy.

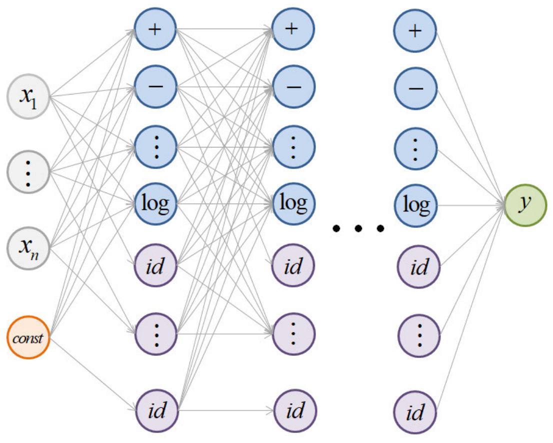

Fig. 2. Overall DeepSymNet framework.

C. Solving the Coefficients

The solution of coefficients is an important step in SR, which has a direct effect on the accuracy of symbolic expressions. The coefficient solution can be divided into solving independently and solving simultaneously with expression structure.

1) Solving the Coefficients Independently: The most common approach to solving the coefficients is to take it as a separate nonlinear optimization problem after obtaining the expression structure. Nonlinear least-squares and quasi-Newton methods [22] in numerical optimization are commonly used solutions. For example, the DSR [16], SymbolicGPT [13], and NeSymReS [14] methods all use BFGS [22] to solve the coefficients.

2) Solving Coefficients Simultaneously With Expression Structure: This type of method combines the expression structure and the coefficients into a single problem to solve. The GP algorithm randomly generates some constants as leaf nodes, and these constant nodes are also updated when performing genetic operations on the symbolic tree. The AIFeyman [2], [17] method uses methods such as polynomial fitting to solve the coefficients when searching for the substructure of the expression. The EQL [18] is an end-to-end method, where the weights in the EQL network are the coefficients in the expression. When the sparse optimization of network parameters is completed, the expression structure and coefficients are determined at the same time. The E2E [15] is a supervised learning-based method that encodes expression structure and coefficients into labels for training, thereby simultaneously predicting expression structure and coefficients.

We recommend references [23], [24], [25], [26] for more information and applications to SR.

III. DeepSymNet

Fig. 2 illustrates the overall DeepSymNet framework. The first layer of DeepSymNet consists of the input data, the middle layer is the hidden layer, and the last layer is the output layer.

The nodes of the hidden layer are composed of operators ($+, -, \times, \div$, sin, cos, exp, log, id), where the id operator is the same as that in EQL [18]. The number of id operators in each

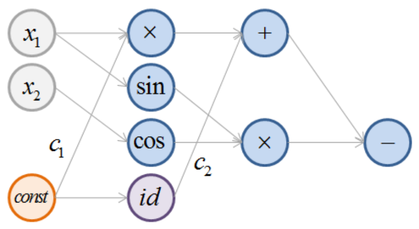

Fig. 3. Subnetwork represents a symbolic expression (take $c_1x_1 - \sin(x_1)\cos(x_2) + c_2$ as an example).

hidden layer is equal to the number of nodes in the previous layer, whereas the other operators appear only once in each hidden layer. Operator id has a one-to-one correspondence with the nodes of the previous layer, and its role is to enable each layer to utilize the information of all previous layers. The other operators are ordinary operators that are entirely connected to the previous layer. Let the number of nodes in the ith layer is n_i and the number of operators except id be m. Subsequently, the number of hidden layer nodes satisfies $n_{i+1} = n_i + m$.

The connections between the id operators and the previous layer were fixed and did not need to be resolved. There is no connection between the ordinary operator and the previous layer, or there are one (operand is 1) or two (operand is 2) connections, which implies that a subnetwork represents a symbolic expression in this network (Fig. 3). The more hidden layers an expression occupies, the higher the complexity of the expression. Thus, the number of hidden layers can be used as a rough measure of the expression complexity.

Note that the input layer has a special node "const" used to represent constant coefficients in symbolic expressions. Only the edges connected to the "const" node have weights (constant coefficients) to prevent sufficiently many constant coefficients in the symbolic expression, as shown in Fig. 3. It is easy to verify that, if DeepSymNet has sufficient layers, it can represent any symbolic expression.

In summary, DeepSymNet is a complete network that can represent any expression, and the solution of SR is the process of searching for subnetworks in DeepSymNet.

IV. Symbolic Expression Solving Algorithm

In this study, the structure and coefficients of symbolic expression were solved separately. First, the expression structure was obtained by searching the subnetwork in DeepSymNet. Subsequently, a nonlinear optimization algorithm was used to solve the coefficients.

A. Supervised Learning to Solve Symbolic Expression Structure

In Section III, we derived that the symbolic expression corresponds to a subnetwork of the symbolic network. Therefore, the solution to the expression structure lies in searching the subnetwork structure, which is a combinatorial optimization problem. In this study, a supervised learning method was adopted to consider the solution of the subnetwork structure as a sequence classification problem. Specifically, we adopted the Transformer [27] model for training, and the loss function was cross-entropy. Transformer [27] is a sequence-to-sequence model based on the "self-attention mechanism," which can calculate the "attention weight" between each position in the input sequence and other positions, thereby achieving interaction and information transmission between different positions in the input sequence. Transformer was originally proposed by Google and used in machine translation tasks, and later was widely used in other tasks, such as text classification, language generation, sequence generation, and so on.

1) Representation of Training Samples: The training samples must be properly represented before training.

a) Training sample input: We randomly generated m input data X_i for the expression $y = f(X)$ to obtain the corresponding $y_i = f(X_i)$. Then, the sequence composed of (X_i, y_i) was constructed as a training sample input. The sequence elements (X_i, y_i) were sorted in ascending order according to each dimension's value of X_i to remove the effect of sequence element order on the results. Specifically, the sequence elements are sorted by the first dimension of X, the second dimension, and so on.

b) Training sample label: We proposed a unique encoding scheme to encode the subnetworks into a sequence of integers as training labels. The integers at the beginning represent the operators selected from DeepSymNet, with zero as the separator, and the following integers represent the connection relationships of the selected operators. Finally, the sequence was filled with pad tokens to a fixed length. Based on the encoding process, we decomposed the subnetwork structure into two subproblems: operator selection and connection relation selection.

TABLE I
Correspondence Between the Operator Code and the Node Number

	+	-	×	÷	sin	cos	exp	log
Operator code	1	2	4	8	16	32	64	128
Node number	0	1	2	3	4	5	6	7

The operators selected in each layer correspond to an integer in the label, and only the first eight operators (ordinary operators) must be selected because the connections between the id operators and the previous layer are fixed. Table I lists each operator as having a unique operator code and node number in the hidden layer. The relationship between the operator code and the node number is code $= 2^{\text{number}}$. There are a total of $2^8 = 256$ cases for eight ordinary operators; therefore, an integer between 0 and 255 can be used to represent the operators selected for each layer. This integer represents the sum of the selected operator codes, which can be parsed according to Table I.

For the encoding of the connection relations, each integer represents the node number of the previous layer connected to the selected operator. The integers are arranged in the order in which the chosen operator appears in the network. For operators with two operands, the left operand comes first and the right operand comes last. Therefore, the length of the connection-relation code depends on the number and type of operators selected.

For example, the subnetwork label shown in Fig. 3 is $[52, 5, 2, 0, 0, 2, 0, 1, 2, 10, 4, 5, 0, 2, 256, \ldots, 256]$, where

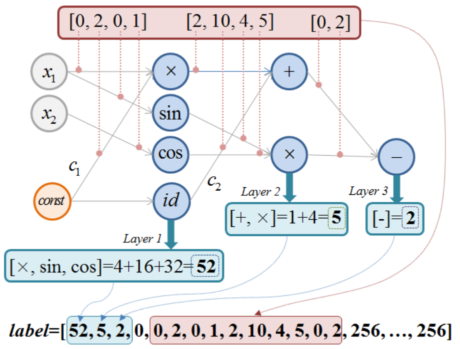

Fig. 4. Parsing process of a label (take $y = c_1x_1 - \sin(x_1)\cos(x_2) + c_2$ as an example).

[52, 5, 2] is the selected operators, the first 0 is the separator, [0, 2, 0, 1, 2, 10, 4, 5, 0, 2] is the connection relation code, and 256 is the pad token. The parsing process for this label is shown in Fig. 4.

2) Sampling of the Training Samples: The training samples are artificially generated. In the process of generating training samples, we improved the quality of the training samples by merging equivalent labels and sample balancing, thereby enhancing the performance of the algorithm. We will demonstrate this through experiments in Section V-B4. We obtained the training samples by making random connections in DeepSymNet (Fig. 2). Random connections must satisfy the following rules.

1) Operators sin, cos, exp, and log cannot be nested with each other. For example, expression $\cos(\sin(x) + c)$ is not allowed.
2) Subtraction and division operators cannot connect two identical nodes.
3) The operands of any operator cannot all be constants.

After obtaining the connection relationships, starting from any node of the last hidden layer and traversing from back to front, a subnetwork can be obtained, corresponding to an expression. The label of an expression was obtained according to the encoding method described in Section IV-A1.

Note that random connections may cause a layer of the subnetwork to not select any ordinary operators (i.e., all id operators and the encoding of the operators of this layer is 0), which is meaningless and conflicts with separator 0. In this case, this layer needs to be deleted, and for the subsequent hidden layer nodes, if the node number is greater than eight, eight needs to be subtracted (because there are eight operators), as shown in Fig. 5(a)→(b).

Another situation that must be addressed is that there are several equivalent labels corresponding to the same expression, as shown in Fig. 5(c)→(b). We need to merge equivalent labels; otherwise, "ambiguity" will occur during training.

It is a challenging task to merge equivalent labels. If they are merged through pairwise comparison, the time complexity is $O(n^2)$ (where n is the number of expressions), which is unacceptable due to the large number of expressions. Therefore, we propose a feasible method for merging equivalent labels, that is, to cluster labels by sorting. We input the same data X to the subnetwork corresponding to the label, obtain the output y, and then sort the labels according to y so that the equivalent labels will be arranged together after sorting. The average time complexity of common sorting algorithms like quicksort [28] and TimeSort [29] is $O(n \log_2 n)$, which is faster than pairwise comparison. For multiple equivalent labels, the label with the least operators is reserved, thus completing the merging of equivalent labels.

We regarded the solution of the subnetwork as a classification problem, and each label is a category. We collected multiple samples for each label (category) as follows: First, the values of the constant coefficients are randomly collected in a certain interval. Then, the values of the constant coefficients are fixed, and m different data points X_i are randomly collected. Finally, we input X_i to the subnetwork to obtain the output y_i, and the sequence composed of (X_i, y_i) is a training sample corresponding to this label. In the process of inputting X_i to obtain y_i, X_i must be resampled if there are some intermediate values with relatively large absolute values; otherwise, data errors will occur during training.

We repeated the above process to collect n training samples for each label, which ensures that the number of samples for each label is balanced and improves the robustness. Evidently, from the sampling process, the coefficients of the samples for the same label are different, which is necessary because of the same label. If the coefficients are different, the function curve will be different. Therefore, different coefficients make the supervised information contained in the training set richer and the trained model more robust.

3) Training Method: After the training data are ready, we input the (X_i, y_i) sequences and the labels to the transformer [27] in the encoding stage, and output the probabilities of occurrence of the label elements in the decoding stage. The loss function is a cross-entropy loss function. Note that 0 must be added in front of each label as the starter before input.

In the Transformer [27] model, each sequence element is represented as a vector of fixed dimensions D. The dimension of each element in the input sequence of the algorithm in this study was variNum + 1, where variNum is the number of independent variables. Therefore, before inputting the sequence to the standard Transformer model, we performed linear mapping on each element to map the element to a D-dimensional vector. The training label was a sequence of integers ranging from 0 to 256. First, we performed word embedding for each integer to transform it into a D-dimensional vector and then input it to the standard transformer model. Fig. 6 illustrates the entire transformer module. Fig. 7 illustrates the schematic of the training phase.

4) Predicting Labels: Label prediction can be performed after training the model. We input the (X_i, y_i) sequence and starter 0 to the Transformer module when predicting the first element of the label and then iteratively predict the following elements until the number of iterations iterNum equals maxLen, which is the maximum length of the label. To improve the prediction accuracy, we used a beam search [19] to obtain multiple candidate labels. A schematic of the label prediction phase is shown in Fig. 8(a).

From the construction of the label in Section IV-A1, the label is divided into two parts: operator selection and connection relationship, which are separated by the separator "0." This suggests that label prediction can be divided into two subtasks (i.e., operator prediction and connection relationship

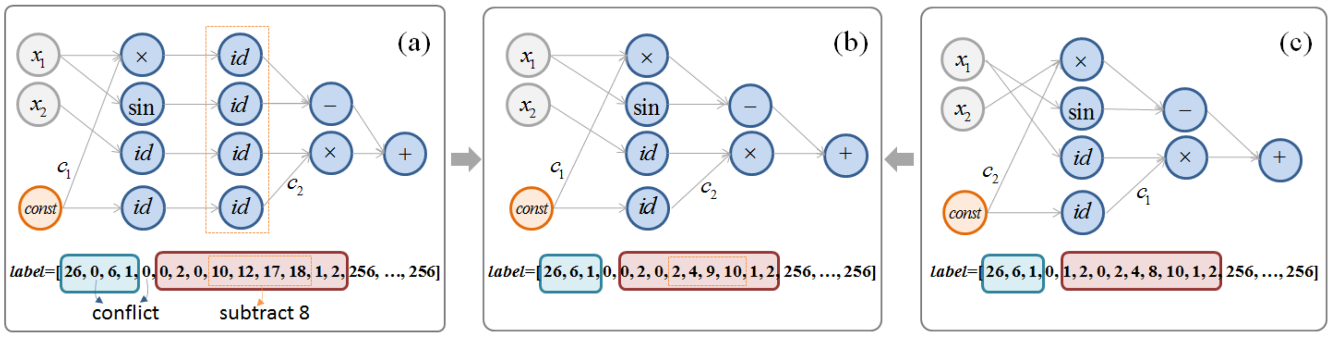
(a)
(b)
(c)
label=[26, 0, 6, 1, 0, 0, 2, 0, 10, 12, 17, 18, 1, 2, 256, …, 256]
conflict
subtract 8
label=[26, 6, 1, 0, 0, 2, 0, 2, 4, 9, 10, 1, 2, 256, …, 256]
label=[26, 6, 1, 0, 1, 2, 0, 2, 4, 8, 10, 1, 2, 256, …, 256]

Fig. 5. Delete the invalid id layers and merge equivalence labels (Take $y = c_1x_1 + c_2x_2 - \sin(x_1)$ as an example). (a) → (b) Delete the invalid id layers. (c) → (b) Merge equivalence labels.

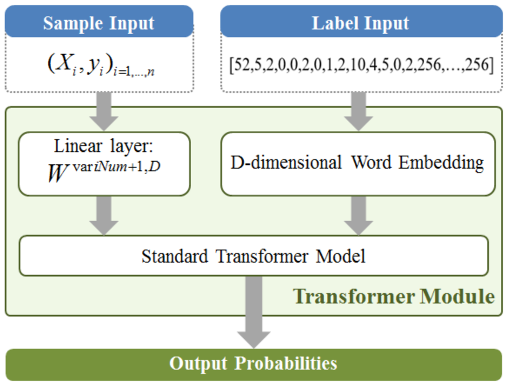
Sample Input
Label Input
$(X_i, y_i)_{i=1,\ldots,n}$
[52,5,2,0,0,2,0,1,2,10,4,5,0,2,256,…,256]
Linear layer: $W^{\text{var}iNum+1,D}$
D-dimensional Word Embedding
Standard Transformer Model
Transformer Module
Output Probabilities

Fig. 6. Transformer module for DeepSymNet.

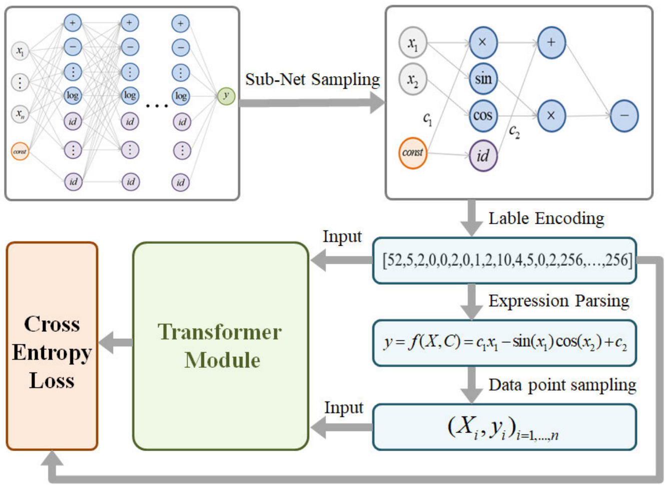

Fig. 7. Schematic of the label training process. First, collect subnetworks and code them to get training labels, then collect data points to get training sample input, and finally input training samples into the model for training. The loss function is the cross-entropy loss function.

prediction) to better solve the label prediction problem. Therefore, we propose an improved solution method that trains and predicts operator selection and connection relationship selection separately. We explore this in detail by conducting experiments in Section V-B3.

B. Solving Constant Coefficients

After the label (i.e., the subnetwork structure) was obtained, the following objective function was constructed, and we used the BFGS [22] optimization algorithm to solve the constant coefficients. Note that other optimization methods such as nonlinear least-squares [22] can also be used

$$\min_{C} \sum_{i=1}^{n} (y_i - f(X_i, C))^2. \tag{1}$$

We selected the one with the smallest fitting error as the final symbolic expression because there are multiple candidate symbolic expressions corresponding to multiple candidate labels obtained by beam search [19]. A schematic of the solution of the constant coefficients is shown in Fig. 8(b).

C. Comparative Analysis of Computational Complexity Between the Proposed Model and Search-Based Methods

The proposed model's computational complexity is independent of the complexity of the expression being solved when making predictions since the size of the model parameters has been determined. The only thing related to the computational complexity is the beam size of the beam search, which is proportional to the computational complexity.

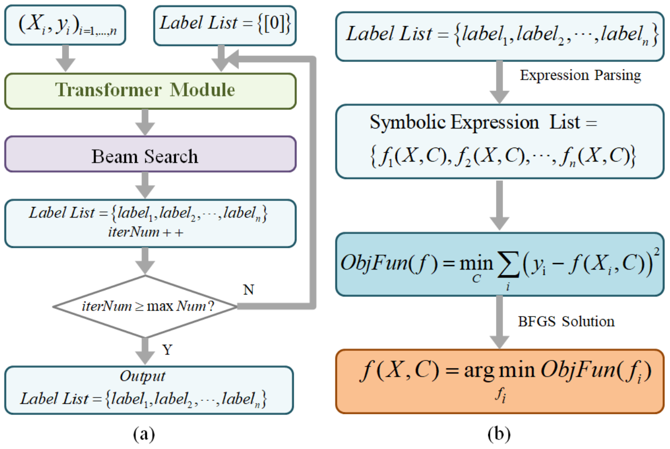
$(X_i, y_i)_{i=1,\ldots,n}$
$Label\ List = \{[0]\}$
Transformer Module
Beam Search
$Label\ List = \{label_1, label_2, \cdots, label_n\}$ $iterNum{+}{+}$
$iterNum \geq \max Num?$
N
Y
Output $Label\ List = \{label_1, label_2, \cdots, label_n\}$
(a)
$Label\ List = \{label_1, label_2, \cdots, label_n\}$
Expression Parsing
Symbolic Expression List = $\{f_1(X,C), f_2(X,C), \cdots, f_n(X,C)\}$
$ObjFun(f) = \min_C \sum_i (y_i - f(X_i,C))^2$
BFGS Solution
$f(X,C) = \arg\min_{f_i} ObjFun(f_i)$
(b)

Fig. 8. Schematic of the solution process. (a) Label prediction process. Input the data points (X_i, y_i) and starter 0 to the Transformer module, and combine with beam search to obtain multiple candidate labels. (b) Constant coefficients solving process. Restore the obtained candidate labels into candidate expressions, construct an optimization objective function for solving coefficients for each expression, and use the BFGS algorithm to solve it. Select the expression with the smallest error as the final result.

The traditional search-based method is iterative, and the number of iterations is related to the complexity of the expression being solved. The higher the complexity, the more iterations there are.

TABLE II
DISTRIBUTION OF THE TRAINING SET AND TESTING SET ACCORDING TO THE NUMBER OF HIDDEN LAYERS (ROUGH EXPRESSION COMPLEXITY)

Number of hidden layers	1	2	3	4	5	6
Training data size	400	21700	227900	250000	250000	250000
Test data size	0	21	299	320	320	320

The search-based methods often require many iterations since the solution of expression structure is an NP-hard combinatorial optimization problem. Therefore, the computational complexity of search-based methods is much higher than the proposed model.

V. EXPERIMENTS

In this section, we test our algorithm on artificially generated and public datasets and compare it with the current popular algorithms. The proposed algorithm was implemented using PyTorch, and all source codes are available at the GitHub repository at https://github.com/wumin86/DeepSymNet.

A. Experimental Parameters Setting

In our experiments, DeepSymNet had up to six hidden layers and supported up to three variables. We generated 20 samples for each label and each sample contained 20 data points. The sampling intervals of the constant coefficients and variables were both $[-2, 2]$. The training strategy we used was early stopping [19] (i.e., training was stopped when the loss on the validation set no longer decreased). The Adam optimizer [19] was used.

B. Test Results on Artificially Generated Data

In this section, we use artificially generated datasets for training and testing to explore the properties of the proposed algorithm.

1) Introduction to Artificially Generated Training Datasets and Test Datasets: The distribution of the training and test sets used in this section is shown in Table II based on the number of hidden layers (rough expression complexity) occupied by the expressions.

Note that half of the test set labels appeared in the training set, and half did not appear in the training set.

2) Comparison of the DeepSymNet Label and Symbolic Tree Label: In this section, we compare the labels of DeepSymNet and the symbolic tree. We used the label of the symbolic tree in NeSymReS [14]. The label was obtained by the preorder traversal of the symbolic tree in NeSymReS. First, we compared the lengths of the two labels. Table III lists the average lengths of the two labels in the expressions for the training set.

In Table III, the average label length of DeepSymNet is shorter than that of NeSymReS because DeepSymNet represents expressions more efficiently than symbolic trees. For the same module that appears multiple times in an

TABLE III
AVERAGE LABEL LENGTHS OF NESYMRES AND DEEPSYMNET IN THE TRAINING SET

Number of hidden layers	1	2	3	4	5	6
NeSymReS	4.70	9.66	12.513	16.21	20.646	24.11
DeepSymNet	**3.9**	**8.05**	**11.24**	**14.72**	**18.60**	**22.04**

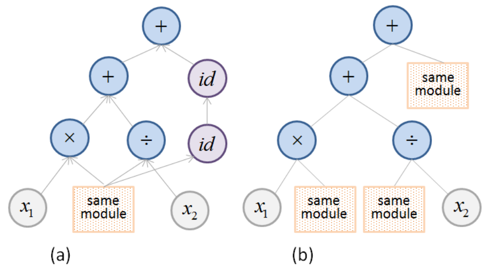

Fig. 9. Comparison diagram of the symbolic tree and DeepSymNet. (a) DeepSymNet. (b) Symbolic tree.

TABLE IV
LABEL PREDICTION ACCURACY WITH LABELS OF NESYMRES AND DEEPSYMNET (THE LABEL APPEARED IN THE TRAINING SET)

Number of hidden layers	$\leqslant 3$	4	5	6
NeSymReS label	0.43125	0.29375	0.23125	0.1875
DeepSymNet label	**0.875**	**0.7**	**0.59375**	**0.45625**

expression, DeepSymNet only needs to construct it once. Then, it can be used everywhere through the id operator, whereas the symbolic tree needs to copy it multiple times. Fig. 9 shows the comparison between DeepSymNet and the symbolic tree, where the box represents the same module, which is constructed once in DeepSymNet and three times in the symbolic tree.

A shorter label may make the prediction easier, and the trained model performs better because the search space size increases exponentially with the label length. To verify our conjecture, we used these two labels to fully train the model separately and then performed label prediction on the test set. We obtained the prediction accuracy of the label prediction according to whether it is consistent with the real label. Beam search [19] is used to obtain multiple candidates when predicting the label, and the prediction is considered successful as long as one of the candidates is consistent with the real label. The experimental results are listed in Table IV.

Based on Table IV, the prediction accuracy of the model trained with the DeepSymNet label is much higher than that of the model trained with the NeSymReS label, which illustrates the superiority of the DeepSymNet label over the symbolic tree label.

3) Splitting DeepSymNet Into Two Parts for Separate Training: From the previous results in Table IV, the prediction accuracy of the proposed model rapidly decreases as the number of hidden layers occupied by expressions increases. We analyzed in Section IV-A that label prediction can be divided into two subtasks, such as operator prediction and connection relationship prediction, to ensure that the label prediction problem can be better solved. In this section,

TABLE V
LABEL PREDICTION ACCURACY (DSN1, THE LABEL APPEARED IN THE TRAINING SET)

Number of hidden layers	⩽ 3	4	5	6
Only select operator beam size:20	0.8875	0.7125	0.59375	0.45625
Only select connection beam size:2	0.975	1.0	1.0	1.0
Only select connection beam size:1	0.975	0.98125	0.9875	0.99375
Complete task beam size:20	0.875	0.7	0.59375	0.45625

TABLE VI
LABEL PREDICTION ACCURACY (DSN2, THE LABEL APPEARED IN THE TRAINING SET)

Number of hidden layers	⩽ 3	4	5	6
Only select operator beam size:20	0.925	0.81875	0.73125	0.55
Complete task beam size:20	0.9125	0.80625	0.725	0.55

experiments are conducted to explore the prediction bottleneck and improve the algorithm performance using this solution.

a) Prediction accuracy of the test set whose label appeared in the training data: The label of the test set appeared in the training set in this part of the experiment (i.e., the corresponding category appeared in the training set).

i) Label prediction results: To explore the prediction bottleneck, we conducted three prediction tasks: operator selection only, connection relation only, and complete prediction task. We input the operator sequence of the label into the model, and the model predicts the connection relations according to the input when only predicting the connection relation. We referred to the trained model as DSN1. The prediction accuracy was calculated for expressions with different numbers of hidden layers. The predicted results are listed in Table V.

Based on the results, as the number of hidden layers increases, the prediction accuracy of operator selection drops sharply, while the prediction accuracy of connection relationships has always been high. Therefore, the prediction bottleneck lies in the operator selection. This is because the operator selection space is much larger than the connection relationship selection space, which is analyzed and discussed in detail in Section V-D. We trained the operator selection separately to improve the accuracy of the operator selection. During prediction, we first obtained the operator selection sequence using the operator selection model and then input it to DSN1 to predict the connection relationship. We refer to the separately trained models DSN2, and the test results are listed in Table VI.

According to the results, the prediction accuracy is greatly improved after the operator selection is trained separately.

ii) Full expression prediction result: After the coefficients were solved using the BFGS [22] algorithm, the full expression was obtained. The error between the predicted and real expressions was calculated using (2), which was used to measure the accuracy of the predicted expression

$$\text{meanError} = \frac{1}{n}\sum_{i=1}^{n} |y_{\text{i}} - f(X_i, C)|. \tag{2}$$

The optimal (real) solution is considered if meanError is less than 0.00001. Table VII lists the prediction accuracies.

TABLE VII
FULL EXPRESSION PREDICTION ACCURACY (THE LABEL APPEARED IN THE TRAINING SET)

Number of hidden layers	⩽ 3	4	5	6
DSN1	0.825	0.68125	0.55	0.425
DSN2	**0.8625**	**0.725**	**0.64375**	**0.5**

TABLE VIII
LABEL PREDICTION RESULTS (DSN1, THE LABELS DID NOT APPEAR IN THE TRAINING SET)

Number of hidden layers	⩽ 3	4	5	6
Only select operator beam size:20	0.2625	0.0375	0.0125	0
Only select connection beam size:2	0.25	0.08125	0	0
Only select connection beam size:1	0.14375	0.025	0	0
Complete task beam size:20	0.05	0	0	0

TABLE IX
LABEL PREDICTION RESULTS (DSN2, THE LABELS DID NOT APPEAR IN THE TRAINING SET)

Number of hidden layers	⩽ 3	4	5	6
Only select operator beam size:20	0.30625	0.05	0	0
Complete task beam size:20	0.0625	0	0	0

The prediction accuracy of DSN2 is higher than that of DSN1 (Table VII). In addition, we found that the accuracy of the full-expression prediction was lower than that of the label. For some expressions, although the label is successfully predicted, the objective function constructed by (1) may be nonconvex, and it is difficult to obtain the correct coefficients using the BFGS algorithm.

b) Prediction accuracy of the test set whose label did not appear in the training data: In this part of the test, the label of the test set did not appear in the training set (i.e., the corresponding category did not appear in the training set).

i) Label prediction results: The prediction accuracies of these labels are listed in Tables VIII and IX. From the results, some expressions whose categories did not appear in the training set can also be predicted correctly. However, the prediction accuracy is relatively low, and the accuracy sharply decreases as the number of hidden layers increases. The prediction accuracy of DSN2 is higher than that of DSN1.

ii) Full expression prediction result: Table X lists the prediction accuracy according to the expression error, and the prediction accuracy of DSN2 is generally higher than that of DSN1.

The full expression prediction accuracy is much higher than that of the label prediction, as shown in Table X. For some

TABLE X
FULL EXPRESSION PREDICTION ACCURACY (THE LABEL DID NOT APPEAR IN THE TRAINING SET)

Number of hidden layers	$\leqslant 3$	4	5	6
DSN1	0.2875	0.08125	**0.03125**	**0.00625**
DSN2	**0.34375**	**0.1125**	0.01875	**0.00625**

TABLE XI
EXPRESSION IS PREDICTED CORRECTLY, ALTHOUGH THE LABEL IS PREDICTED WRONG

	label	Expression	coefficient
Target	[1,32,2,0,2, 0,0,5,17]	$cos(c_1+a)-b$	$c_1=0.5309$
Prediction	[1,33,2,0,2, 0,9,10,0,5,0]	$cos(c_1+a)-(b+c_2)$	$c_1=0.5309$ $c_2=1.288e-8$

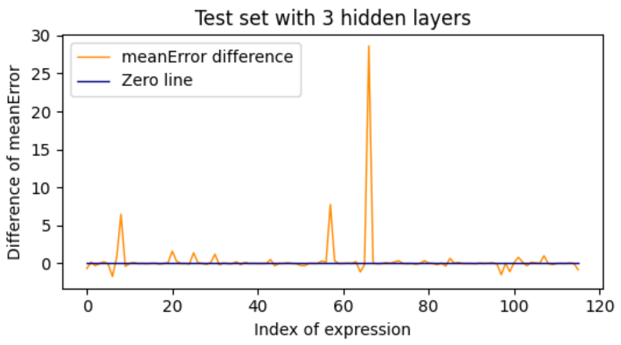

Fig. 10. meanError difference between DSN1 and DSN2 on test set with $\leqslant 3$ hidden layers.

expressions, although label prediction fails, correct expressions can be obtained by solving the coefficients. As presented in Table XI, although the predicted expression has one more coefficient c_2, coefficient c_2 obtained by BFGS is sufficiently small and can be ignored; therefore, the prediction can be considered correct.

c) Performance comparison of DSN1 and DSN2 for approximate solutions: We explored the accuracy of finding the optimal solution in previous experiments. However, in many cases, the model could not obtain the optimal (real) solution, but only an approximate solution, particularly when the labels of the test samples did not appear in the training set. We can use meanError to evaluate the performance of an algorithm in determining approximate solutions.

We drew the meanError difference ($\text{meanError}_{\text{DSN1}} - \text{meanError}_{\text{DSN2}}$) curves of all expressions with approximate solutions to compare DSN1 and DSN2 more intuitively (Figs. 10–13). Based on the results, most of the curves are above 0; thus, for most expressions, the meanError obtained by DSN1 is larger than that of DSN2.

Furthermore, we made statistics on the meanError of Figs. 10–13, and the statistical indicator is the number of votes calculated based on the meanError. The specific voting method is that, for a certain expression, the vote number of model1 is increased by one if the meanError of model1 is less than or equal to model2, as expressed in the following equation:

$$\text{vote}_1 = \begin{cases} 1, & \text{meanError}_1 \leqslant \text{meanError}_2 \\ 0, & \text{otherwise.} \end{cases} \quad (3)$$

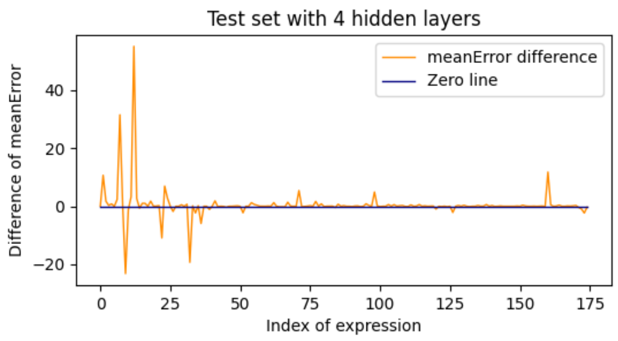

Fig. 11. meanError difference between DSN1 and DSN2 on test set with four hidden layers.

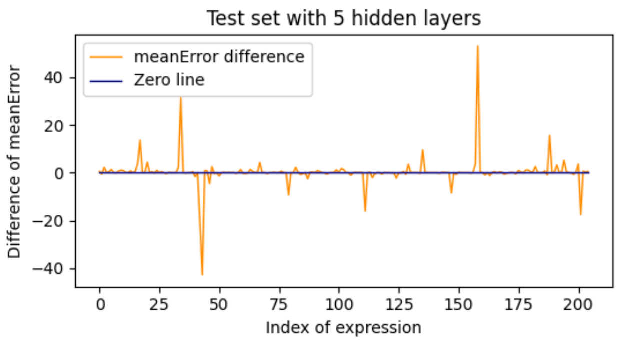

Fig. 12. meanError difference between DSN1 and DSN2 on test set with five hidden layers.

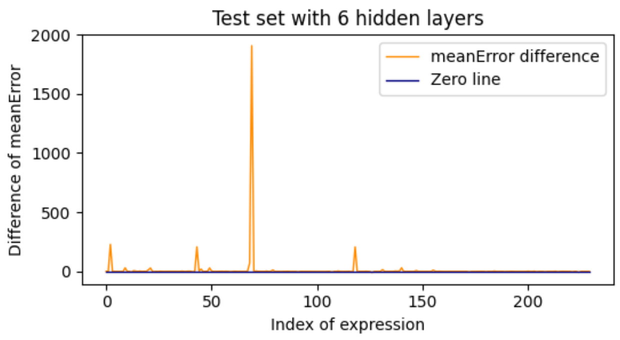

Fig. 13. meanError difference between DSN1 and DSN2 on test set with six hidden layers.

TABLE XII
COMPARISON OF THE VOTE NUMBER BETWEEN DSN1 AND DSN2

Number of hidden layers	$\leqslant 3$	4	5	6
Total vote number	116	175	205	230
The vote number of DSN1	52	71	89	78
The vote number of DSN2	**70**	**115**	**126**	**156**

A comparison of the votes for DSN1 and DSN2 is presented in Table XII. The vote number of DSN2 is higher than that of DSN1 based on the results.

4) Ablation Experiment to Verify the Enhanced Robustness of Equivalent Label Merging and Sample Balance: Equivalent label merging and sample balance are two methods used to enhance algorithm robustness. In this section, we conducted ablation experiments to verify the enhancement of algorithm robustness by these two methods. We first randomly sampled 500 000 training samples called TrainDataOrg, which

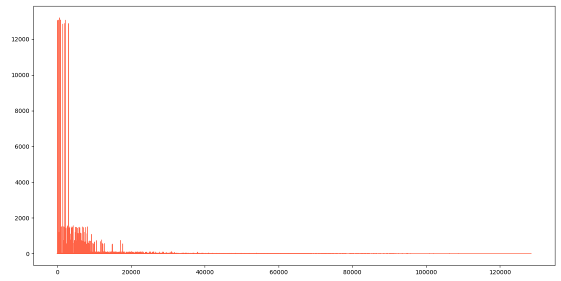

Fig. 14. Histogram of the sample number distribution corresponding to the labels.

TABLE XIII
FULL EXPRESSION PREDICTION ACCURACY

Number of hidden layers	$\leqslant 3$	4	5	6
DSNOrg	0.484375	0.10625	0.025	0.003125
DSNB	0.625	0.209375	0.03125	0.015625
DSNBM	**0.628125**	**0.30625**	**0.184375**	**0.11875**

contained 128 455 different labels (categories). We counted the histogram of the sample number distribution corresponding to these labels, as shown in Fig. 14.

We found that the distribution of sample numbers in these labels is seriously uneven, with the minimum sample number being 1, the maximum sample number being 13 196, and the variance of the sample number being 13012.29. We balanced the sample size. On the premise that the total number of samples was 500 000, the number of samples for each of the first 114 635 labels was 4, and the number of samples for the last 13 820 labels was 3 (because $114\,635 \times 4 + 13\,820 \times 3 = 500\,000$). We called the obtained training samples TrainDataB.

The number of labels decreased to 77 089 after merging the equivalent labels and we further balanced the number of samples. The number of samples for each of the first 37 466 labels was 7, and the number of samples for the last 39 623 labels was 6 (because $37\,466 \times 7 + 39\,623 \times 6 = 500\,000$). We called the obtained training samples TrainDataBM.

We trained on the training data TrainDataOrg, TrainDataB, and TrainDataBM to obtain models DSNOrg, DSNB, and DSNBM, respectively. These three models were tested on test sets, which are generated in the same way as in Section V-B1. The accuracy of these three models on the test set is shown in Table XIII. It can be seen that the accuracy of DSNOrg, DSNB, and DSNBM increases sequentially, indicating that after adding sample balance and merging equivalent labels, the accuracy of the model in solving the optimal solution is improved.

We also compared the number of votes between these three models, as shown in Table XIV. It can be seen from Table XIV that DSNB has a greater number of votes than DSNOrg, while DSNBM has a greater number of votes than DSNB, indicating that the accuracy of the model has been improved by sequentially adding sample balance and merging equivalent labels.

In summary, equivalence label merging and sample balance have indeed enhanced the robustness of the algorithm and improved its performance.

From the above experiments, we can draw the following conclusions.

TABLE XIV
COMPARISON OF THE NUMBER OF VOTES

Number of hidden layers	$\leqslant 3$	4	5	6
DSNB:DSNOrg	**280:188**	**240:113**	**218:109**	**219:104**
DSNBM:DSNB	**257:244**	**212:160**	**194:139**	**180:145**

TABLE XV
NUMBER OF EXPRESSIONS CONTAINED IN EACH DATASET

Koza	Korns	Keijzer	Vlad	ODE	AIFeynman
11	6	9	7	12	36

1) The difficulty of prediction increases as the number of hidden layers (i.e., the complexity) of the expression increases.
2) The bottleneck of label prediction lies in operator selection.
3) DSN2 is better than DSN1 in solving both the optimal and approximate solutions.
4) Equivalence label merging and sample balance can enhance the robustness of the proposed algorithm.

C. Test Results on Public Datasets

In this section, we use the SR public test datasets to test our algorithm and compare it with the current popular algorithms. Six test datasets were used in this study: Koza, Korns, Keijzer, Vlad [30], ODE [25], and AIFeynman [2]. We selected expressions from these datasets with no more than three variables for testing because the currently trained model supports up to three variables, and all the selected symbolic expressions are shown in the Appendix. Table XV lists the number of expressions included in each dataset.

1) Comparison With Supervised Learning-Based Methods: In this section, we compared the proposed algorithm with the current popular supervised learning-based methods, E2E [15], SymbolicGPT [13], and NeSymReS [14].

a) Introduction of training datasets: The training data labels we used in this section are the same as those in Section V-B4, but we generated 20 training samples for each label, resulting in a total of $77\,089 \times 20 = 1571\,780$ training samples. All algorithms are trained using exactly the same training data.

b) Performance evaluation results: The performance evaluation indicators used are the meanError of the expression. Fig. 15 illustrates the plot of meanError values of the tested algorithms for each dataset. Based on Fig. 15, the meanError of the proposed algorithm (DSN1, DSN2) is the smallest.

We also calculated the votes for each dataset using the method presented in Section V-B3 between the algorithms, and the results are shown in Tables XVI–XXI. From Tables XVI–XXI, the accuracy of the algorithms proposed in this study (DSN1, DSN2) is better than that of the compared algorithms.

2) Comparison With Search-Based Methods: In this section, we compared the proposed algorithm with the current popular search-based methods, EQL [18], GP [5], and DSR [16]. To make a fair comparison in a limited time, we set the population size of the GP algorithm to 2000 and the maximum evolutionary generation to 20. The maximum number of

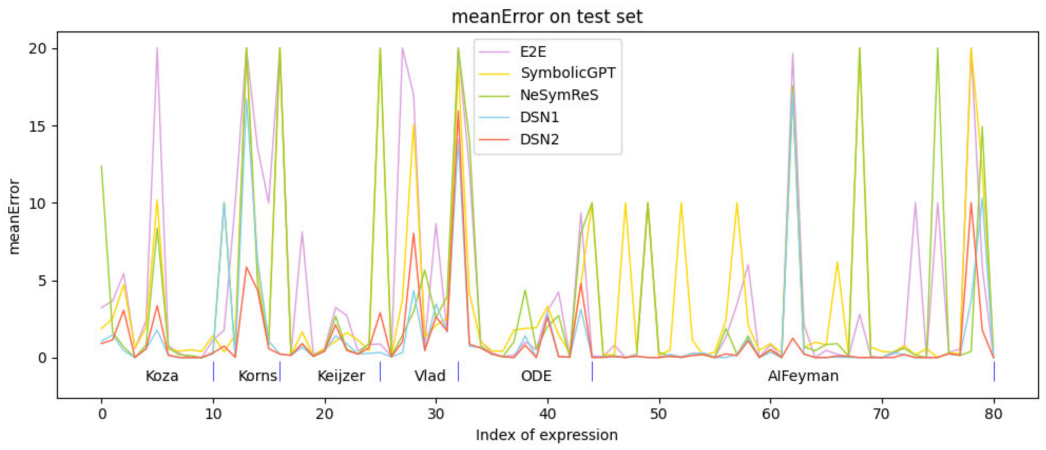

Fig. 15. Comparison of meanError on test set with supervised-learning based algorithms.

TABLE XVI

NUMBER OF VOTES ON THE KOZA DATASET (COMPARED WITH SUPERVISED LEARNING-BASED METHODS)

	E2E	SymbolicGPT	NeSymReS	DSN1	DSN2
E2E	–	5:6	2:9	1:11	1:11
SymbolicGPT	**6:5**	–	2:9	0:11	0:11
NeSymReS	**9:2**	**9:2**	–	2:9	4:7
DSN1	**11:1**	**11:0**	**9:2**	–	**7:4**
DSN2	**11:1**	**11:0**	**7:4**	4:7	–

TABLE XVII

NUMBER OF VOTES ON THE KORNS DATASET (COMPARED WITH SUPERVISED LEARNING-BASED METHODS)

	E2E	SymbolicGPT	NeSymReS	DSN1	DSN2
E2E	–	1:5	2:4	1:5	0:6
SymbolicGPT	**5:1**	–	3:3	3:3	1:5
NeSymReS	**4:2**	3:3	–	2:3	0:6
DSN1	**5:1**	3:3	**3:2**	–	1:5
DSN2	**6:0**	**5:1**	**6:0**	**5:1**	–

TABLE XVIII

NUMBER OF VOTES ON THE KEIJZER DATASET (COMPARED WITH SUPERVISED LEARNING-BASED METHODS)

	E2E	SymbolicGPT	NeSymReS	DSN1	DSN2
E2E	–	3:6	2:7	0:9	1:8
SymbolicGPT	**6:3**	–	3:6	1:8	2:7
NeSymReS	**7:2**	**6:3**	–	2:7	0:9
DSN1	**9:0**	**8:1**	**7:2**	–	**6:4**
DSN2	**8:1**	**7:2**	**9:0**	4:6	–

TABLE XIX

NUMBER OF VOTES ON THE VLAD DATASET (COMPARED WITH SUPERVISED LEARNING-BASED METHODS)

	E2E	SymbolicGPT	NeSymReS	DSN1	DSN2
E2E	–	2:5	3:4	1:6	1:6
SymbolicGPT	**5:2**	–	**5:2**	1:6	1:6
NeSymReS	**4:3**	2:5	–	2:5	2:5
DSN1	**6:1**	**6:1**	**5:2**	–	**5:3**
DSN2	**6:1**	**6:1**	**5:2**	3:5	–

TABLE XX

NUMBER OF VOTES ON THE ODE DATASET (COMPARED WITH SUPERVISED LEARNING-BASED METHODS)

	E2E	SymbolicGPT	NeSymReS	DSN1	DSN2
E2E	–	**8:4**	5:7	1:11	0:12
SymbolicGPT	4:8	–	4:7	0:12	1:11
NeSymReS	**7:5**	**7:4**	–	4:8	3:9
DSN1	**11:1**	**12:0**	**8:4**	–	5:8
DSN2	**12:0**	**11:1**	**9:3**	**8:5**	–

TABLE XXI

NUMBER OF VOTES ON THE AIFEYNMAN DATASET (COMPARED WITH SUPERVISED LEARNING-BASED METHODS)

	E2E	SymbolicGPT	NeSymReS	DSN1	DSN2
E2E	–	**25:10**	11:24	7:32	7:33
SymbolicGPT	10:25	–	9:26	2:34	2:34
NeSymReS	**24:11**	**26:9**	–	10:26	10:26
DSN1	**32:7**	**34:2**	**26:10**	–	21:23
DSN2	**33:7**	**34:2**	**26:10**	**23:21**	–

TABLE XXII

DISTRIBUTION OF THE TRAINING SET ACCORDING TO THE NUMBER OF HIDDEN LAYERS (ROUGH EXPRESSION COMPLEXITY)

Number of hidden layers	1	2	3	4	5	6
Training Size	520	25780	1386640	2503540	1012080	569960

iterations for DSR was 120, and EQL adopted its default settings.

a) Introduction of training datasets: In this experiment, more training data were used for training to improve the prediction performance. The distribution of the training set used is shown in Table XXII according to the number of hidden layers (rough expression complexity) occupied by the expressions. To improve the training efficiency, the training set was divided into four parts for separate training based on the number of hidden layers: less than or equal to 3 and equal to 4, 5, and 6. Therefore, we obtained four independently trained models. Each model predicts separately with a beam size of 10, and the candidate with the smallest meanError among all the candidates is selected as the final result.

b) Performance evaluation results: The performance evaluation indicators used are the meanError of the expression, the complexity of the predicted expression, and the running time of the algorithm. The complexity of an expression is measured by the number of operators in the expression.

The interpretability of an expression is closely related to the complexity of the expression. The lower the complexity, the stronger the interpretability, and vice versa. Therefore, the complexity of the expression should be considered when evaluating the performance of an SR model. In our experiments, if the complexity of the expression is greater than three times that of the real expression, the current expression

TABLE XXIII
NUMBER OF VOTES ON THE KOZA DATASET (COMPARED WITH SEARCH-BASED METHODS)

	EQL	GP	DSR	DSN1	DSN2
EQL	–	1:10	3:8	3:8	2:9
GP	**10:1**	–	**7:4**	5:6	3:8
DSR	**8:3**	4:7	–	4:7	3:8
DSN1	**8:3**	**6:5**	**7:4**	–	3:8
DSN2	**9:2**	**8:3**	**8:3**	**8:3**	–

TABLE XXIV
NUMBER OF VOTES ON THE KORNS DATASET (COMPARED WITH SEARCH-BASED METHODS)

	EQL	GP	DSR	DSN1	DSN2
EQL	–	0:6	2:4	0:6	0:6
GP	**6:0**	–	2:3	0:6	0:6
DSR	**4:2**	**3:2**	–	1:5	0:6
DSN1	**6:0**	**6:0**	**5:1**	–	2:4
DSN2	**6:0**	**6:0**	**6:0**	**4:2**	–

TABLE XXV
NUMBER OF VOTES ON THE KEIJZER DATASET (COMPARED WITH SEARCH-BASED METHODS)

	EQL	GP	DSR	DSN1	DSN2
EQL	–	0:9	0:9	0:9	0:9
GP	**9:0**	–	**5:4**	3:6	3:6
DSR	**9:0**	4:5	–	1:8	1:8
DSN1	**9:0**	**6:3**	**8:1**	–	3:6
DSN2	**9:0**	**6:3**	**8:1**	**6:3**	–

TABLE XXVI
NUMBER OF VOTES ON THE VLAD DATASET (COMPARED WITH SEARCH-BASED METHODS)

	EQL	GP	DSR	DSN1	DSN2
EQL	–	0:7	0:7	0:7	0:7
GP	**7:0**	–	**4:3**	2:5	2:5
DSR	**7:0**	3:4	–	0:7	0:7
DSN1	**7:0**	**5:2**	**7:0**	–	3:4
DSN2	**7:0**	**5:2**	**7:0**	**4:3**	–

TABLE XXVII
NUMBER OF VOTES ON THE ODE DATASET (COMPARED WITH SEARCH-BASED METHODS)

	EQL	GP	DSR	DSN1	DSN2
EQL	–	0:12	0:12	0:12	0:12
GP	**12:0**	–	**8:4**	3:9	3:9
DSR	**12:0**	4:8	–	1:11	1:11
DSN1	**12:0**	**9:3**	**11:1**	–	**7:5**
DSN2	**12:0**	**9:3**	**11:1**	5:7	–

TABLE XXVIII
NUMBER OF VOTES ON THE AIFEYNMAN DATASET (COMPARED WITH SEARCH-BASED METHODS)

	EQL	GP	DSR	DSN1	DSN2
EQL	–	1:35	2:34	0:36	0:36
GP	**35:1**	–	**24:15**	8:28	5:31
DSR	**34:2**	15:24	–	9:27	7:29
DSN1	**36:0**	**28:8**	**27:9**	–	16:22
DSN2	**36:0**	**31:5**	**29:7**	**22:16**	–

TABLE XXIX
NUMBER OF WINS FOR EACH ALGORITHM ON EACH DATASET

	EQL	GP	DSR	DSN1	DSN2
Koza	0	2	1	3	**4**
Korns	0	1	2	3	**4**
Keijzer	0	2	1	3	**4**
Vlad	0	2	1	3	**4**
ODE	0	2	1	**4**	3
AIFeynman	0	2	1	3	**4**

TABLE XXX
RUNNING TIME (SECONDS)

	EQL	GP	DSR	DSN1	DSN2
Koza	2712.30	431.40	534.71	**237.80**	274.63
Korns	1339.46	299.63	304.23	**155.68**	176.66
Keijzer	1924.20	388.40	416.07	**200.12**	217.44
Vlad	1476.13	414.32	342.18	203.65	**177.47**
ODE	2488.25	473.81	542.78	**294.61**	296.17
AIFeynman	7997.83	1337.90	1535.22	968.43	**789.60**

is considered unexplainable, and meanError is set to 10. Note that the expression complexity obtained using the EQL algorithm was sufficiently high. The minimum expression complexity of EQL is 121, which greatly exceeds three times the complexity of the real expression. For a convenient comparison, we did not modify the meanError value obtained by the EQL algorithm and used the original value.

Fig. 16 illustrates the plot of meanError values of the tested algorithms for each dataset. Fig. 17 shows the plot of the absolute value of the complexity difference between the predicted and real expressions in the datasets. Based on Figs. 16 and 17, the meanError of the proposed algorithm (DSN1, DSN2) is the smallest, and the complexity of the obtained expression is also closest to the real expression complexity.

We calculated the votes for each dataset using the method presented in Section V-B3 between the algorithms and the results are shown in Tables XXIII–XXVIII. Furthermore, we calculated the number of wins for each algorithm based on Tables XXIII–XXVIII, as listed in Table XXIX. From Tables XXIII–XXIX, the accuracy of the algorithms proposed in this study (DSN1, DSN2) is better than that of the compared algorithms, and the performance of DSN2 is better than that of DSN1.

Table XXX lists the running time of each algorithm on each dataset, and the running speed of the proposed algorithm (DSN1, DSN2) is the fastest.

Based on the results, it can be concluded that the algorithm in this study is superior to the compared algorithms in three aspects: symbolic expression error, symbolic expression complexity, and running speed, proving the effectiveness of the proposed algorithm.

D. Discussion

This section discusses the features of our algorithm to gain a deeper understanding of the proposed model.

1) Why Does Operator Selection Become a Bottleneck for Prediction? Why Does DSN2 Work Better Than DSN1?: The selection space for each operator element in the label is 256 by analyzing the encoding rules in Section IV-A1, whereas for the connection relations, the selection space of each element depends on the operator selection, and its maximum value does not exceed the number of nodes in the previous layer. Therefore, the connection relationship selection space is much smaller than the operator selection space, and operator selection becomes a prediction bottleneck. By taking the expression shown in Fig. 4 as an example, the selection space sizes of the connection relationship for the first, second, and

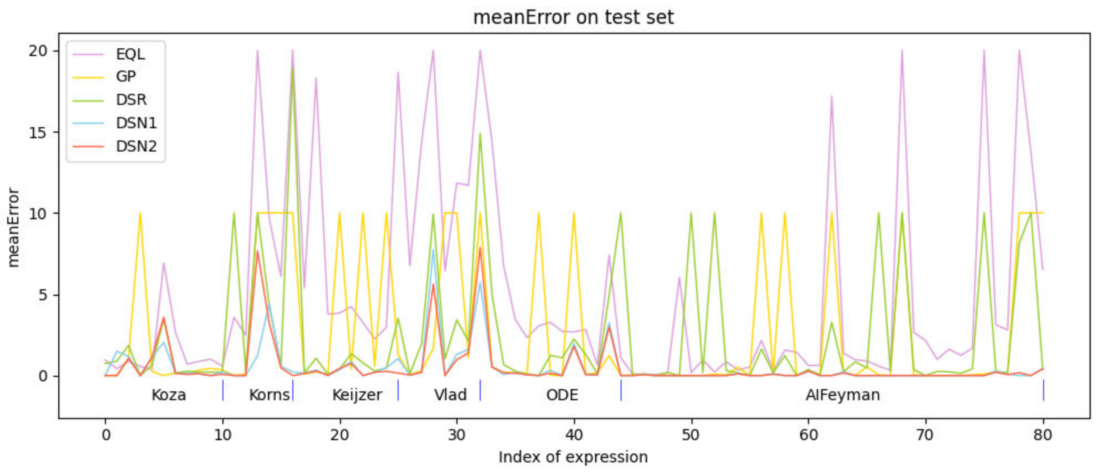

Fig. 16. Comparison of meanError on test set with search based algorithms.

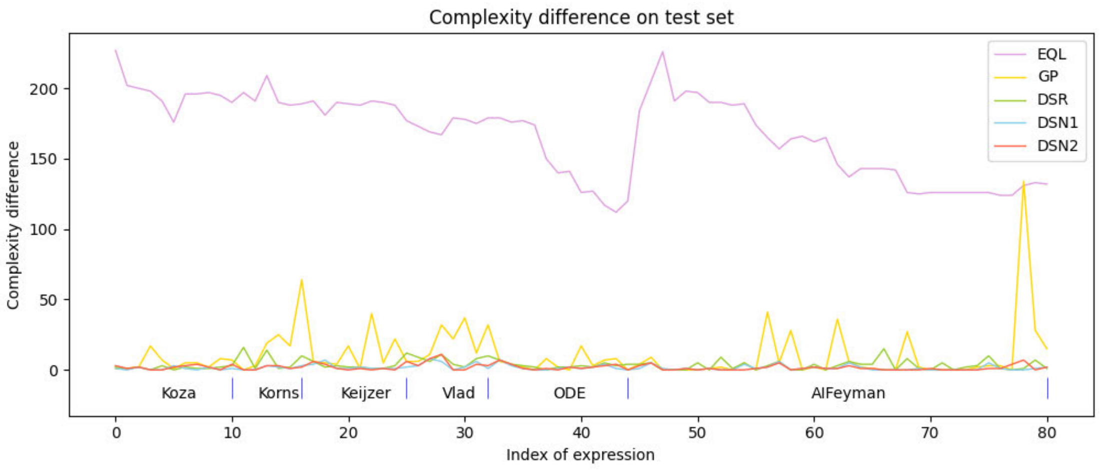

Fig. 17. Complexity difference between the predicted expression and the real expression on the test set.

third hidden layer nodes are three, six, and eight, respectively, which are much smaller than the operator selection space size.

DSN2 trains operator selection with a separate model, which allows the model to focus on the learning of operator selection. Thus, DSN2 performs better than DSN1.

2) Has the Trained Model Learned How to Solve SR?: The proposed algorithm was regarded as a classification model during the previous training process, and different labels corresponded to different categories. Therefore, it is difficult to obtain an optimal solution for expressions whose categories do not appear in the training set. However, although an optimal solution cannot be obtained, an approximate solution can be determined from previous experimental results. This implies that our model learned how to solve the SR during training.

We compared a randomly initialized model with the trained model to further verify that the trained model has learned the relevant SR knowledge and not by chance. Only the connection relation is randomly selected in the random model because the random selection of the operator can cause illegal expressions. Therefore, the random model in this experiment was a semirandom model.

The beam size was set to 1 for both models. The evaluation index used is the meanError difference between the semirandom and trained models. The experimental results are shown in Fig. 18. Based on the results, the meanError of the semirandom model on all test samples is much larger than that of the trained model. In addition, the trained and semirandom models obtained eight and zero optimal solutions, respectively. This shows that the trained model performs better than the semirandom model when solving both the optimal and approximate solutions, which further suggests that the proposed model has learned how to solve SR through supervised learning.

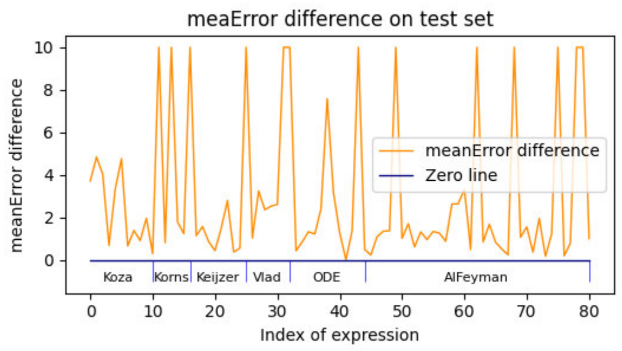

Fig. 18. meanError difference between the semirandom model and trained model on the test set.

TABLE XXXI
KOZA DATASET

Expression	Expression
$x^4+x^3+x^2+x$	x^5-2x^3+x
$x^6-2x^4+x^2$	x^3+x^2+x
$x^5+x^4+x^3+x^2+x$	$x^6+x^5+x^4+x^3+x^2+x$
$\sin\left(x^2\right)\cos\left(x\right)-1$	$\sin\left(x\right)+\sin\left(x+x^2\right)$
$\sin\left(x\right)+\sin\left(y^2\right)$	$2\sin\left(x\right) cos\left(y\right)$
$\frac{1}{1+x^{-4}}+\frac{1}{1+y^{-4}}$	

TABLE XXXII
KORNS DATASET

Expression	Expression
$1.57+(24.3x)$	$0.23+14.2\frac{x+y}{3z}$
$-2.3+0.13sin(x)$	$213.80940889(1-e^{-0.54723748542x})$
$6.87+11\cos(7.23x^3)$	$2-2.1\cos(9.8x)\sin(1.3y)$

TABLE XXXIII
KEIJZER DATASET

Expression	Expression
$0.3x\sin(2\pi x)$	$x^3e^{-x}\cos(x)\sin(x)(\sin^2(x)\cos(x)-1)$
$\frac{30xz}{(x-10)y^2}$	$In(x+\sqrt{x^2+1})$
$xy+\sin((x-1)(y-1))$	$x^4-x^3+\frac{y^2}{2}-y$
$6\sin(x)\cos(y)$	$\frac{8}{2+x^2+y^2}$
$\frac{x^3}{5}+\frac{y^3}{2}-y-x$	

TABLE XXXIV
VLAD DATASET

Expression	Expression
$\frac{e^{-(x-1)^2}}{1.2+(y-2.5)^2}$	$e^{-x}x^3\cos\left(x\right)\sin\left(x\right)\left(\cos\left(x\right)sin^2\left(x\right)-1\right)$
$30\frac{(x-1)(z-1)}{y^2(x-10)}$	$e^{-x}x^3\cos\left(x\right)\sin\left(x\right)\left(\cos\left(x\right)sin^2\left(x\right)-1\right)\left(y-5\right)$
$6\sin\left(x\right)cos\left(y\right)$	$(\mathrm{x}-3)(\mathrm{y}-3)+2\mathrm{sin}\left((\mathrm{x}-4)(\mathrm{y}-4)\right)$
	$\frac{(x-3)^3+(y-3)^3-(y-3)}{(y-2)^4+10}$

TABLE XXXV
ODE DATASET

Expression	Expression
$20-x-\frac{xy}{1+0.5x^2}$	$10-\frac{xy}{1+0.5x^2}$
$0.5\sin(x-y)-\sin(x)$	$-0.05x^2-\sin(y)$
$x-\frac{\cos(y)}{x}$	$3x-2xy-x^2$
$2y-xy-y^2$	$x(4-x-\frac{y}{1+x})$
$y(\frac{x}{1+x}-0.075y)$	$(\cos^2(x)+0.1\sin^2(x))\sin(y)$
$10(y-\frac{1}{3}(x^3-x))$	$-\frac{1}{10}x$

TABLE XXXVI
AIFEYNMAN DATASET

Expression	Expression	Expression
$\frac{\sqrt{2}e^{-\frac{x^2}{2}}}{2\sqrt{\pi}}$	$\frac{\sqrt{2}e^{-\frac{y^2}{2x^2}}}{2\sqrt{\pi}x}$	xy
$\frac{x^2y}{2}$	$\frac{x}{y}$	$\frac{xy}{2\pi}$
$\frac{3xy}{2}$	$\frac{x}{4\pi y^2}$	$\frac{xy^2}{2}$
$\frac{xy}{-\frac{xy}{3}+1}+1$	xy^2	$\frac{x}{2y+2}$
$\frac{\sqrt{2}e^{-\frac{(y-z)^2}{2x^2}}}{2\sqrt{\pi}x}$	$\frac{x}{4\pi yz^2}$	xyz
$\frac{y+z}{1+\frac{yz}{x^2}}$	$xy\sin(z)$	$\frac{1}{\frac{z}{y}+\frac{1}{x}}$
$\frac{x\sin^2(\frac{yz}{2})}{\sin^2(\frac{y}{2})}$	$\frac{z}{1-\frac{y}{x}}$	$\frac{yz}{x-1}$
$\frac{x}{4\pi yz}$	$\frac{3x^2}{20\pi yz}$	$\frac{x}{y(z+1)}$
$-xy\cos(z)$	xyz^2	$\frac{xy}{2\pi z}$
$\frac{xyz}{2}$	$\frac{xy}{4\pi z}$	$xy(z+1)$
$\frac{4\pi xy}{z}$	$\sin^2(\frac{2\pi xy}{z})$	$2x(1-\cos(yz))$
$\frac{x^2}{8\pi^2yz^2}$	$\frac{2\pi x}{yz}$	$x(y\cos(z)+1)$

VI. CONCLUSION

This article proposed a supervised learning-based SR algorithm that investigates core problems in SR. First, DeepSymNet was proposed to efficiently represent symbolic expressions. The prediction of the expression structure is regarded as a classification problem, which offers a fresh perspective on supervised learning-based SR methods. DeepSymNet is convenient for decomposing SR into operator selection and connection relationship selection, improving the solution performance. These contributions allowed us to obtain a model with high accuracy. Furthermore, our model can provide good approximate solutions to test samples whose labels did not appear in the training set, indicating that the trained model has learned how to solve SR. The proposed algorithm was tested on artificially generated data and public test datasets and compared with current popular algorithms. The experimental results prove the superiority of the algorithm in terms of accuracy and speed. This study is an application of AI in solving mathematical problems (i.e., AI for mathematics).

However, the proposed algorithm has some limitations. Its greatest limitation is that the prediction result is affected when the sampling interval of the test sample variable is inconsistent with that of the training sample. A possible solution is to combine it with search-based algorithms, such as reinforcement learning, to leverage the strengths of both algorithms.

APPENDIX
PUBLIC DATASETS USED IN EXPERIMENTS

The public test datasets (Koza, Korns, Keijzer, Vlad [30], ODE [25], and AIFeynman [2]) used in this article are presented in Tables XXXI–XXXVI.

REFERENCES

[1] D. Wadekar et al., "Augmenting astrophysical scaling relations with machine learning: Application to reducing the Sunyaev–Zeldovich flux-mass scatter," 2022, *arXiv:2201.01305*.

[2] S.-M. Udrescu and M. Tegmark, "AI Feynman: A physics-inspired method for symbolic regression," *Sci. Adv.*, vol. 6, no. 16, Apr. 2020, Art. no. eaay2631.

[3] C. Wilstrup and J. Kasak, "Symbolic regression outperforms other models for small data sets," 2021, *arXiv:2103.15147*.

[4] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA: MIT Press, 1992.

[5] D. A. Augusto and H. J. C. Barbosa, "Symbolic regression via genetic programming," in *Proc. 6th Brazilian Symp. Neural Netw.*, 2000, pp. 173–178.

[6] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org

[7] D. Silver et al., "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[8] A. W. Senior et al., "Improved protein structure prediction using potentials from deep learning," *Nature*, vol. 577, no. 7792, pp. 706–710, 2020.

[9] G. Lample and F. Charton, "Deep learning for symbolic mathematics," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–24. [Online]. Available: https://openreview.net/forum?id=S1eZYeHFDS

[10] G. Wang et al., "Encoder-X: Solving unknown coefficients automatically in polynomial fitting by using an autoencoder," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 8, pp. 3264–3276, Aug. 2022.

[11] J. Liu, G. Wang, W. Li, L. Sun, L. Zhang, and L. Yu, "Transcendental equation solver: A novel neural network for solving transcendental equation," *Appl. Soft Comput.*, vol. 117, Mar. 2022, Art. no. 108425.

[12] R. Stevens, V. Taylor, J. Nichols, A. B. Maccabe, K. Yelick, and D. Brown, "Ai for science: Report on the department of energy (DOE) town Halls on artificial intelligence (AI) for science," US Dept. Energy, The Argonne, Oak Ridge, Berkeley Nat. Lab., USA, Tech. Rep. ANL-20/17 158802; TRN: US2103893, 2020. [Online]. Available: https://www.osti.gov/biblio/1604756

[13] M. Valipour, B. You, M. Panju, and A. Ghodsi, "SymbolicGPT: A generative transformer model for symbolic regression," in *Proc. 2nd workshop Efficient Natural Lang. Speech Processing*, vol. 2, Dec. 2022, pp. 1–11.

[14] L. Biggio, T. Bendinelli, A. Neitz, A. Lucchi, and G. Parascandolo, "Neural symbolic regression that scales," in *Proc. 38th Int. Conf. Mach. Learn.*, vol. 139, M. Meila and T. Zhang, Eds. Jul. 2021, pp. 936–945.

[15] P.-A. Kamienny, S. d'Ascoli, G. Lample, and F. Charton, "End-to-end symbolic regression with transformers," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 10269–10281.

[16] B. K. Petersen, M. L. Larma, T. N. Mundhenk, C. P. Santiago, S. K. Kim, and J. T. Kim, "Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients," in *Proc. Int. Conf. Learn. Represent.*, May 2021, pp. 1–26.

[17] S. Udrescu, A. K. Tan, J. Feng, O. Neto, T. Wu, and M. Tegmark, "AI Feynman 2.0: Pareto-optimal symbolic regression exploiting graph modularity," in *Proc. Conf. Workshop Neural Inf. Process. Syst.*, Dec. 2020, pp. 4860–4871.

[18] S. Sahoo, C. Lampert, and G. Martius, "Learning equations for extrapolation and control," in *Proc. 35th Int. Conf. Mach. Learn.*, Jul. 2018, pp. 4442–4450.

[19] X. Qiu, *Neural Networks and Deep Learning*. Beijing, China: China Machine Press, 2020. [Online]. Available: https://nndl.github.io/

[20] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," OpenAI, USA, Tech. Rep., 2018. [Online]. Available: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

[21] J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, and Y. W. Teh, "Set transformer: A framework for attention-based permutation-invariant neural networks," in *Proc. Int. Conf. Mach. Learn. (ICML)*, K. Chaudhuri and R. Salakhutdinov, Eds. Jun. 2019, pp. 3744–3753.

[22] J. Nocedal and S. Wright, *Numerical Optimization*. Berlin, Germany: Springer, 2006.

[23] S. Kim et al., "Integration of neural network-based symbolic regression in deep learning for scientific discovery," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 9, pp. 4166–4177, Sep. 2021.

[24] Z. Zhang and Z. Chen, "Modeling and control of robotic manipulators based on symbolic regression," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 5, pp. 2440–2450, May 2023.

[25] W. L. Cava, P. Orzechowski, B. Burlacu, F. Frana, and J. H. Moore, "Contemporary symbolic regression methods and their relative performance," in *Proc. 35th Conf. Neural Inf. Process. Syst.*, 2021, pp. 1–28.

[26] M. Cranmer et al., "Discovering symbolic models from deep learning with inductive biases," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds. Red Hook, NY, USA: Curran Associates, 2020, pp. 17429–17442.

[27] A. Vaswani et al., "Attention is all you need," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.* Red Hook, NY, USA: Curran Associates, 2017, pp. 6000–6010.

[28] R. Sedgewick and K. Wayne, *Algorithms*, 4th ed. Reading, MA, USA: Addison-Wesley, 2011.

[29] T. Peters. (2002). *Timsort*. [Online]. Available: http://svn.python.org/projects/python/trunk/Objects/listsort.txt

[30] J. McDermott et al., "Genetic programming needs better benchmarks," in *Proc. 14th Annu. Conf. Genetic Evol. Comput.*, T. Soule and J. H. Moore, Eds. Philadelphia, PA, USA, Jul. 2012, pp. 791–798, doi: 10.1145/2330163.2330273.

Min Wu received the B.S. degree in computer science and technology from Beijing Jiaotong University, Beijing, China, in 2009, and the M.S. and Ph.D. degrees from the University of Chinese Academy of Sciences, Beijing, in 2012 and 2020, respectively.

He is currently an Assistant Research Fellow with the AnnLab, Institute of Semiconductors, Chinese Academy of Sciences, Beijing. His research interests include machine learning, optimization algorithms, and AI for science.

Weijun Li (Senior Member, IEEE) received the Ph.D. degree from the University of Chinese Academy of Sciences (UCAS), Beijing, China, in 2004.

He is currently a Professor of artificial intelligence with the Institute of Semiconductors Chinese Academy of Sciences (ISCAS), Beijing, and the University of Chinese Academy of Sciences, Beijing. He is also the Director of the Laboratory of Highspeed Circuits Neural Networks (AnnLab), ISCAS. His research interests include deep modeling, artificial neural networks, and AI for Science.

Lina Yu (Member, IEEE) received the Ph.D. degree from the College of Information and Electrical Engineering, China Agricultural University, Beijing, China, in June 2016.

From July 2016 to July 2018, she was a Post-Doctoral Research Fellow with the AnnLab, Institute of Semiconductors, Chinese Academy of Sciences, Beijing, where she is currently a Senior Engineer. Her research works focus on machine learning, symbolic regression, and AI for Science.

Linjun Sun (Member, IEEE) received the Ph.D. degree from the Institute of Semiconductors, Chinese Academy of Sciences, Beijing, China, in 2020.

He is currently an Assistant Research Fellow with the AnnLab, Institute of Semiconductor, Chinese Academy of Sciences. His primary research interests include deep learning, face recognition, and symbolic regression.

Jingyi Liu received the B.S. and M.S. degrees from the Zhongnan University of Economics and Law, Wuhan, China, in 2018 and 2021, respectively. She is currently pursuing the Ph.D. degree with the Institute of Semiconductors, Chinese Academy of Sciences, Beijing, China.

Her main research interests include polynomial fitting and symbolic regression.

Wenqiang Li (Graduate Student Member, IEEE) received the B.S. degree from the China University of Geosciences (Beijing), Beijing, China, in 2021. He is currently pursuing the master's (M.A.) degree with the Institute of Semiconductors, Chinese Academy of Sciences, Beijing.

His current research interests include deep learning, symbolic regression, and large language models.