Computer Science and Application 计算机科学与应用, 2025, 15(6), 69-82
Published Online June 2025 in Hans. https://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2025.156158

Hans汉斯

基于Transformer的端到端深度符号回归方法

吴云浩[1]，田益民[2*]

[1]北京印刷学院信息工程学院，北京
[2]北京印刷学院基础教育学院，北京

收稿日期：2025年5月12日；录用日期：2025年6月11日；发布日期：2025年6月18日

摘　要

本研究提出了一种基于Transformer的端到端深度符号回归(TFSR)模型，用于通过自动生成数学表达式来解决符号回归问题。符号回归是一种无需先验模型形式的回归方法，旨在通过给定的数据推导出合理的数学表达式，具有较高的可解释性。随着深度学习的进步，深度符号回归(DSR)利用神经网络的强大数据处理能力，能够从高维数据中提取潜在的数学规律。本文提出的TFSR模型结合了进化算法和Transformer架构，通过自注意力机制有效处理数据中的长距离依赖关系，简化了符号回归过程并提升了效率与精度。此外，本研究还引入了前缀符号表示法和二叉树结构来高效表示和处理数学表达式，采用了分区采样方法生成均匀分布的采样数据。实验结果表明，TFSR模型在不同规模的数据集上具有良好的学习能力和泛化能力，尤其在处理复杂数学表达式时表现优异。通过基准测试与其他符号回归方法的对比，本研究的模型在复杂公式建模任务中展现了更强的表现，尤其在解决困难组问题时，TFSR模型表现出显著的优势。该研究为符号回归任务提供了一种新的端到端深度学习方法，尤其适用于复杂科学模型的自动构建，具有广泛的应用潜力。

关键词

深度符号回归，符号回归，Transformer模型

End-to-End Deep Symbolic Regression Method Based on Transformer

Yunhao Wu[1], Yimin Tian[2*]

[1]School of Information Engineering, Beijing Institute of Graphic Communication, Beijing
[2]School of Basic Education, Beijing Institute of Graphic Communication, Beijing

Received: May 12th, 2025; accepted: Jun. 11th, 2025; published: Jun. 18th, 2025

Abstract

This study proposes an end-to-end Transformer-based deep symbolic regression (TFSR) model to

*通讯作者。

文章引用: 吴云浩, 田益民. 基于 Transformer 的端到端深度符号回归方法[J]. 计算机科学与应用, 2025, 15(6): 69-82.
DOI: 10.12677/csa.2025.156158

solve symbolic regression problems by automatically generating mathematical expressions. Symbolic regression is a regression method that does not require a priori model form. It aims to derive reasonable mathematical expressions from given data and has high interpretability. With the advancement of deep learning, deep symbolic regression (DSR) uses the powerful data processing capabilities of neural networks to extract potential mathematical laws from high-dimensional data. The TFSR model proposed in this paper combines evolutionary algorithms and Transformer architectures, effectively handles long-distance dependencies in data through self-attention mechanisms, simplifies the symbolic regression process, and improves efficiency and accuracy. In addition, this study also introduces prefix symbol representation and binary tree structure to efficiently represent and process mathematical expressions, and uses partition sampling methods to generate uniformly distributed sampling data. Experimental results show that the TFSR model has good learning and generalization capabilities on datasets of different sizes, especially when processing complex mathematical expressions. Through benchmark tests and comparisons with other symbolic regression methods, the model of this study shows stronger performance in complex formula modeling tasks, especially when solving difficult group problems, the TFSR model shows significant advantages. This study provides a new end-to-end deep learning method for symbolic regression tasks, which is particularly suitable for the automatic construction of complex scientific models and has broad application potential.

Keywords

Deep Symbolic Regression, Symbolic Regression, Transformer Model

1. 引言

科学或工程的很多系统都可以使用数学表达式进行建模。直接使用第一性原理推导公式比较枯燥繁杂，并且很多系统根本不可能给出公式。在这种情形下，适应于实际数据的数学表达式由程序自动构造出来的方法就是符号回归。符号回归(Symbolic Regression, SR)方法属于基于数据的机器学习方法，其目的是由给定的数据推导出合理的数学表达式。与传统的回归方法和机器学习方法不同，符号回归方法不需要任何关于数学模型形式的先验，而且可以在更大的候选空间内寻找最优解。符号回归方法是直接在给定自变量 X 和因变量 y 的数据的情况下，获得因变量和自变量的最优组合的表达式，进而求解出最优的系数。

最经典的符号回归方法就是基于 JohnKoza 提出的遗传编程(Genetic Programming, GP) [1]方法，通过进化的方式产生、评估、组合和变异大量候选表达式，直到产生符合的表达式且达到所要求的精度水平。本质上是一种在数学表达式的大空间中搜索最优解的方法，去寻求符合优化目标函数的最佳公式。随着机器学习方法的不断发展，符号回归经历了多次创新与改进，包括改进遗传算法和引入贝叶斯方法等[2]-[6]。与传统的黑箱模型相比，符号回归生成的表达式具有更高的可解释性[6]-[8]。简单来说，假如输入与输出的关系可以用简洁的逻辑或数学方式表述，则这个模型就是可解释的。

随着深度学习的不断发展，部分学者将神经网络引入符号回归方法中，在此基础上，深度符号回归(Deep Symbolic Regression, DSR) [9]-[14]方法逐渐受到关注。深度符号回归是利用深度学习的强大数据处理能力以及符号回归的可解释性优势，可以利用深层神经网络对大维度数据进行隐藏数学规律的提取。

近年来，许多研究人员对符号回归方法不断进行改进，并且随着深度学习和神经网络技术的不断发

展取得了显著进展，为解决问题提供了新的思路和解决方案。Vladislavleva 等人引入了非线性阶作为复杂度的度量，探索了数值精度与非线性阶优化之间的权衡[3]，并成功提高了外推能力。McConaghy 等人提出了 FFX 算法[4]。与传统方法不同，该方法极大程度的提高了符号回归的计算速度和预测的简洁性。

随着深度学习及神经网络的飞速发展，深度符号回归的研究也逐渐受到重视。Lample 等展示了神经网络求解符号回归问题的能力[10]，展现了神经网络解决数学问题如符号积分以及方程求解的强大能力。Biggio 等人利用序列－序列模型将数值数据映射成相关的数学符号方程[11]，并通过深度神经网络增强了符号回归的性能。Wu 等人将符号回归问题转化为分类问题来解决，提出了结合深度学习的框架 DeepSymNet，极大地提升了符号回归模型的预测能力[12]。这些方法展示了深度学习在符号回归中的应用潜力，不仅提高了模型的表现，还提供了新的思路和方法。随着 Transformer 在符号回归中的不断发展，研究人员也提出了一些优化方法。Kamienny 等人提出了一种端到端的符号回归方法[15]，直接用 Transformer 对整个数学表达式进行预测，避免了传统符号回归中繁琐的分步预测过程，并极大提升推理效率和预测精度。Lalande 等人提出了一个全新的 Transformer 模型[16]，该方法，专门用于符号回归任务。该方法避免过拟合问题，实验在多项数据集上实现较好的效果。该研究将符号回归扩展到了科学研究中，为一些复杂科学模型的构造提供了一些新的思路和方法。Vastl 等人提出的 SymFormer 模型[17]，展示了一种可以直接输出符号和常数项生成数学公式的创新方法。

上述的研究中仍旧存在效率不足和准确率不佳的问题，通常无法直接获取完整的表达式。为提升模型性能，结合端到端的思想，本文提出了一种基于进化算法和 transfromer 的 TFSR 模型。

2. 符号回归任务中的表达式

深度符号回归任务中，通常是通过序列到序列(Seq2Seq)的模型来实现的。这类似于其他机器学习中翻译任务的实现方法。为此引入了前缀符号表示法(也称为波兰表示法)，其中父节点位于子节点左侧。整个表达式按从左到右的顺序排列成序列，从而容易被机器学习模型来处理。

表达式和二叉树之间存在一对一的对应关系。一般而言，数学表达式计算顺序由运算符优先级与括号控制。将数学表达式转成二叉树后，可以非常容易得到清晰的表达式结构。在处理复杂的算术运算时，二叉树能够自动处理运算符的优先级和括号的作用[15]。这个方法消除了计算优先级和括号的影响，还能够很自然的能够处理运算的优先性和依赖性。与传统中缀表达式不同，前缀表达式不需要括号，且计算机处理起来更为高效。前缀表达式和二叉树之间存在一对一的映射关系，每个前缀表达式序列都可以对应一个特定的二叉树结构，从而为计算机提供更加高效的表达式求值路径。

生成表达式通过从由以下标记组成的固定词汇表中采样 token = [add、mul、inv、opp、exp、log、sqrt，pow2、pow3、C、$x1$、$x2$]。其中 add 和 mul 是二元运算符，其余运算符是一元运算符。C、$x1$、$x2$ 则是系数和变量的占位符。表达式中可能存在多个 C，在计算时按照顺序为系数进行赋值。本研究中限制为至多两个 C 占位符，即在实际计算式转换为 c_1 和 c_2。在二叉树结构中，二元运算符需要两个子节点，而一元运算符则仅需一个子节点。

需要补充说明的是，减法和除法作为算术中常见的两个运算符，在实验过程中并未涉及。因为减法可以通过加法(add)和相反数(opp)的组合实现，而除法则可通过乘法(mul)和倒数(inv)的结合来完成。例如，表达式 $2 + 3 \times (5 + 2)$和 $3x^2 + \cos(2x) - 1$ 可以被表示为如图 1 所示的二叉树。

此外，在二叉树向序列转化时，将导致序列的长度变长。文中，在表达式生成时为了使公式中不同运算符出现的次数有规律，给每个运算符设定了不同权重。

本研究中生成表达式的算法可以概括为：(1) 根据权重随机选择一个运算符作为根节点；(2) 按照权重随机选择并填充子节点；(3) 如果子节点依然是运算符，则继续填充该节点的子树；(4) 以此类推，直

至子节点都为变量或系数才算完成。如果当前所需的节点是根节点，则必须要排除掉变量和常量。当变量和常量被选择时，就意味着当前子树无法再继续扩展深度(即当前子树已经完成)。

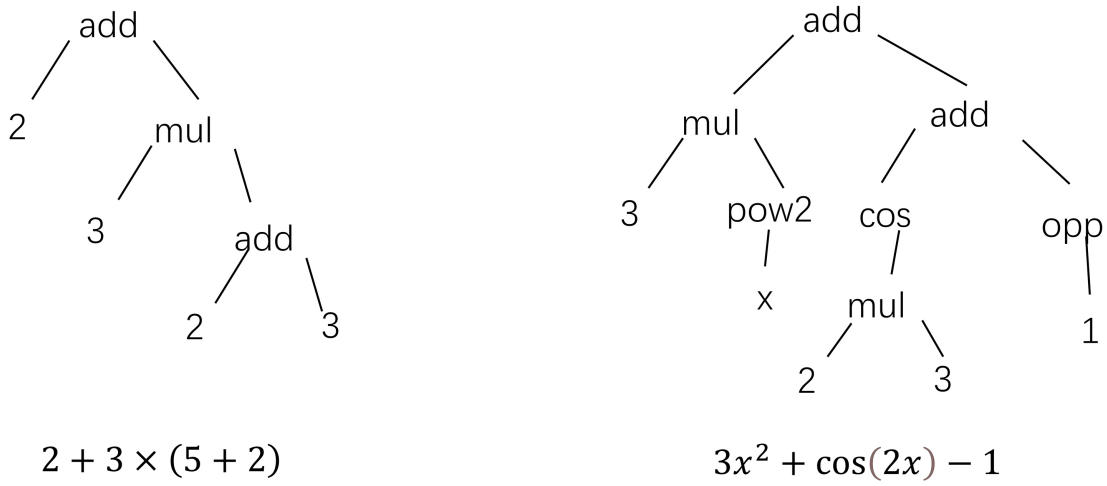


Figure 1. Binary tree representation of an expression
图 1. 表达式的二叉树表示

3. 数据采样与数据集生成

3.1. 分区采样

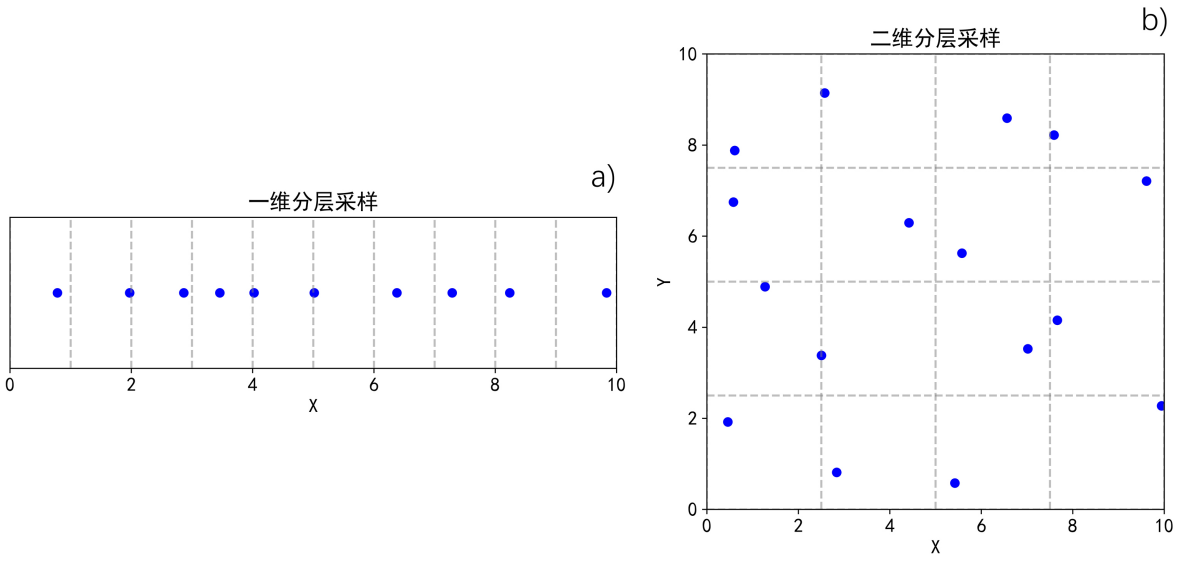


Figure 2. Schematic diagram of partition sampling method. (a) One-dimensional partition sampling; (b) Two-dimensional partition sampling
图 2. 分区采样方法示意图。(a) 一维分区采样；(b) 二维分区采样

完整的数据集包括表达式以及相对应的采样数据。这里的数据应包含以下内容：前缀表达式，变量、系数及公式采样值。现对表达式采样的方法进行讨论。合理的采样值是训练效果稳定的关键。

本研究中选择 SymPy 作为基础工具处理表达式，SymPy 的特点主要有：可以进行符号的代数计算(包括化简、展开、多项式计算等)；可以进行积分和微分；可以求解代数方程、线性方程和非线性方程等；

SymPy 支持数值计算，它支持浮点运算且精度高；可以与常用的 Python 库 NumPy、SciPy 配合工作，用于不同领域中的符号计算和数学建模。

这里对变量和常数分别进行采样，考虑常数和变量在不同数量下的采样策略：1) 对于两个以上采样值(这里仅变量可能存在两个以上的情况)，使用随机采样的方法即可。因为高维空间划分难度较大。2) 对于两个及以下的采样值(两个及以下常量或者两个及以下的变量)，使用分区采样。

本文的实验方法采用了一种分区采样方式，尽可能使得采样更均匀地分布在区间。在对变量 x_1 和 x_2 进行采样时，先将采样区间均匀的分成若干个子区间，再在每个子区间中取值。通过这种改进使采样值更均匀地分布于区间，并更符合实际实验的采样习惯。模型能更好地学习采样数据的特征。

假设将输入数据空间 $X=[a,b]$ 划分为 L 个层，即 $X_1,X_2,\cdots,X_L$，每个层 X_i 对应输入空间的一部分。根据单变量表达式与双变量表达式的不同，本研究采用了不同的分区采样方法，即一维分区采样和二维分区采样。如图 2 所示。

首先给出一维空间的分区采样算法，一维分区采样如图 2(a)所示。

1) 划分区：将输入空间划分为 L 个子区域，每个子区域 X_i 可以是一个等宽的区间或根据数据的分布进行动态划分。例如，输入空间 $X=[a,b]$ 可以划分为 L 个等宽区间：

$$X_i=\left[a+(i-1)\cdot\frac{b-a}{L},a+i\cdot\frac{b-a_i}{L}\right],i=1,2,\cdots,L \tag{1}$$

2) 在每个层内进行采样：对每个子区域 X_i 进行独立采样，每个层内的样本点 $x_{ij}\in X_i$ 以下公式生成：

$$x_{ij}=a+(i-1)\cdot\frac{b-a}{L}+\left(\frac{b-a}{L}\right)\cdot rand(i,j) \tag{2}$$

其中，$rand(i,j)$ 是随机数生成函数，表示在每个层 x_{ij} 内生成的样本。

3) 计算对应的输出：对于每个采样点 x_{ij}，计算其输出 $y_{ij}=f\left(x_{ij}\right)$，其中 $f\left(x_{ij}\right)$ 是给定的数学表达式。

在二维空间中，分区采样方法类似于一维空间中的分区采样，只是将空间扩展为两个维度，如图 2(b)所示。这里假设输入数据空间是二维的，表示为 $X=[a_1,b_1]\times[a_2,b_2]$，其中 $x_1\in[a_1,b_1],x_2\in[a_2,b_2]$。这里 a_1 和 b_1 是第一个维度(x_1 轴)的最小值和最大值，a_2 和 b_2 是第二个维度(x_2 轴)的最小值和最大值。将输入空间 X 沿 x_1 轴方向划分为 L_1 个子区域沿 x_2 轴方向划分为 L_2 个子区域。共计 $L_1\times L_2$ 个子区域，每个子区域 $X_{i,j}$ 包含 x_1 和 x_2 中的一段范围。

二维分区采样算法如下所示。

1) 划分区：

对于第一个维度 (a_1,b_1) 的划分：

$$X_i=\left[a_i+(i-1)\cdot\frac{b_1-a_1}{L_1},a_1+i\cdot\frac{b_1-a_1}{L_1}\right],i=1,2,\cdots,L_1 \tag{3}$$

对于第二个维度 (a_2,b_2) 的划分：

$$X_j=\left[a_i+(j-1)\cdot\frac{b_2-a_2}{L_2},a_2+j\cdot\frac{b_2-a_2}{L_2}\right],j=1,2,\cdots,L_2 \tag{4}$$

2) 在每个子区域内进行采样：

对于每个子区域 X_{ij}，在其中进行独立的随机采集。每个子区域内的样本点 $x_{i,j,k,l}\in X_{ij}$ 由以下公式生成：

$$x_{i,j,k_1}=\left[a_1+(i-1)\cdot\frac{b_1-a_1}{L_1}+\left(\frac{b_1-a_1}{L_1}\right)\cdot rand(i,j,k_1)\right] \tag{5}$$

$$x_{i,j,k_2}=\left[a2+(j-1)\cdot\frac{b_2-a_1}{L_2}+\left(\frac{b_2-a_2}{L_2}\right)\cdot rand(i,j,k_2)\right] \tag{6}$$

其中，$rand(i,j,k_1)$ 和 $rand(i,j,k_2)$ 是分别在 x_1 和 x_2 方向上生成的随机数，范围在[0, 1]内。

3) 计算对应的输出：

对于每个采样点 $(x_{i,j,k_1},x_{i,j,k_2})$，计算其对应的输出 $y_{i,j,k1,k_2}$：

$$y_{i,j,k_1,k_2}=f(x_{i,j,k_1},x_{i,j,k_2}) \tag{7}$$

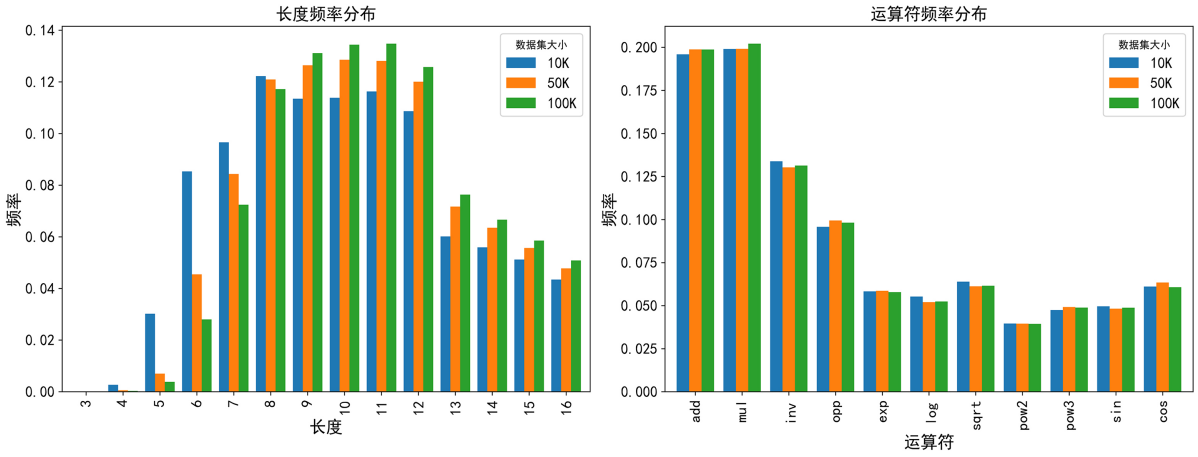
其中，f 是给定的数学函数或者表达式。

3.2. 数据集生成

深度符号回归数据集中包含表达式、系数值和对因变量 X、自变量 Y 的采样数据。一个优质的数据集能够极大地提高模型性能和泛化性能。

为了获取采样值，首先将使用第 2 节的方法获取随机生成的前缀表达式，并将其转换为 SymPy 公式。对于每个唯一有效的表达式，在常数范围[0, +10]内采样。针对每个表达式的采样过程会重复 m 次(即针对常数 c_1 和 c_2 进行多次采样，本研究中设定为 20 次)，从而生成多个具有不同常数值但结构相同的前缀表达式。最后，剔除无法成功采样的数据集(例如当公式的值过大或为复数时)。此外，若采样过程中出现除以零或计算负对数等错误情况，相关数据和表达式也会同时被丢弃，简而言之，仅保留存在于实数域的表达式。之后对生成的数据集进行了详细的统计与分析：(1) 运算符组成分析：分析了数据集中各个运算符的组成和分布。(2) 表达式长度分布：对数据集中的表达式长度进行了统计。如图 3 所示。

不同运算符的权重如下：(1) 一元运算符[inv, opp, exp, log, sqrt, pow2, pow3, sin, cos] = [2, 2, 1, 1, 1, 1, 1, 1, 1]；(2) 二元运算符[add, mul] = [3, 3]；(3) 变量和系数[x_1, x_2, C] = [2, 1, 1]。



(a) (b)

Figure 3. Prefix expression length and operator distribution
图 3. 前缀表达式长度和运算符分布

4. 深度符号回归模型设计

4.1. 适用于采样数据的解码器

本节中提出一种基于 transformer 端到端深度符号回归模型。为了使 transformer 更适用于深度符号回

归的任务，运用了一种新的编码器并构建了新的模型结构，通过结合束搜索和遗传算法，提升了模型的预测性能。模型方法如图 4 所示。

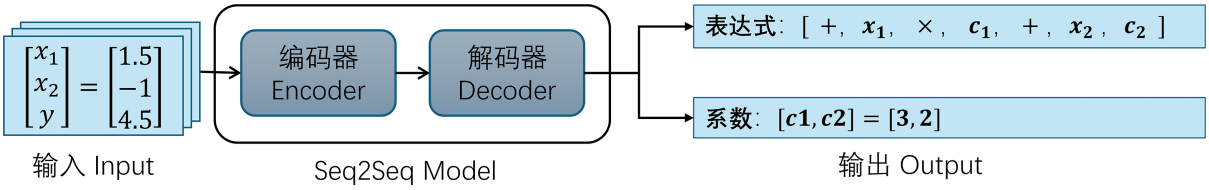


Figure 4. End-to-end deep symbolic regression model
图 4. 端到端深度符号回归模型

很多基于 Transformer 的序列到序列任务中，输入通常是文本、语音等序列。然而在深度符号回归任务中，编码器的输入是一组采样数据。输入的差异导致了无法直接使用 Transformer 架构中的嵌入层和编码器结构。因此本文利用一种基于注意力机制的处理采样数据的编码器结构，来处理并且编码采样数据的特征[16]。模型结构如图 5 所示。

为了方便讨论，假设单个批次的输入向量大小为(n,d)，并假设解码器所使用的序列长度为l。输入数据首先通过一个 1 维的 MLP 层进行变换，然后送入多个编码层进行递归处理。每个编码层会对输入数据进行自注意力计算来提取样本的特征。最后经过最后一个 MLP 层和平均池化操作来获得最终的输出。编码器层结构如图 6 所示。

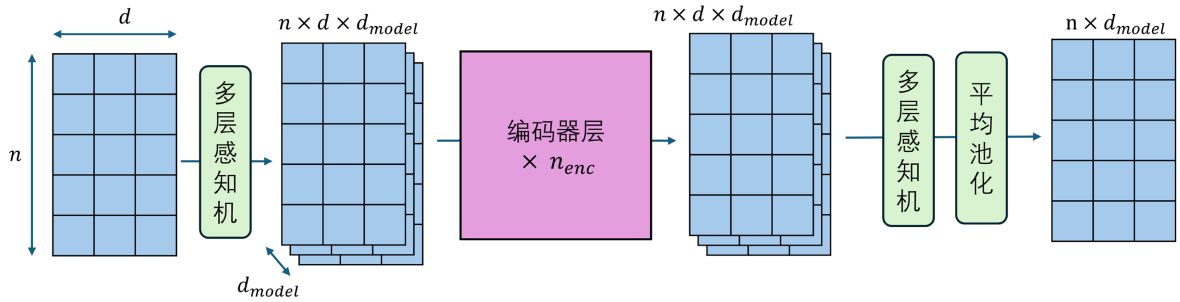


Figure 5. Encoder structure for sampled data
图 5. 针对采样数据的编码器结构

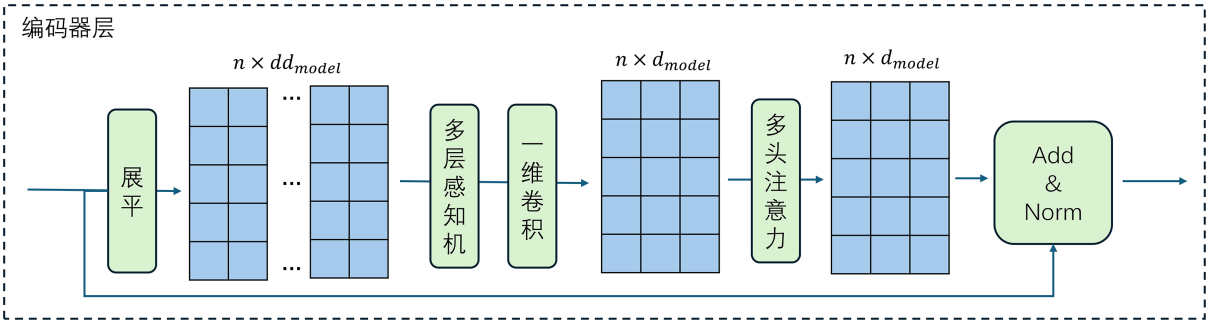


Figure 6. Encoder layer structure
图 6. 编码器层结构

4.2. 用于公式生成的解码器

解码器的核心任务是根据编码器的输出生成两个独立的预测结果：一个是前缀表达式(符号表达式)，

另一个是与之相关的系数。解码器的设计分为两个主要分支，分别对应生成表达式骨架和预测系数。

表达式骨架预测解码器采用了典型 transformer 解码器结构，负责输出表达式符号序列。它由嵌入层和多个解码器层堆叠而成，每个解码器层都包含自注意力机制(Self-Attention)、跨注意力机制(Cross-Attention)和前馈神经网络(FFN)。

解码器层的第一部分是多头自注意力机制。多头自注意力机制让模型在解码时关注输入序列的不同部分，为输入序列之间的关系依赖增添辅助信息。解码器层的第二部分是跨注意力机制。跨注意力机制是解码器的关键模块，通过跨注意力模块，让解码器有足够的上下文信息。在每一个解码器层的最后部分，模型使用一个前馈神经网络对跨注意力的输出进行非线性变换。这个前馈神经网络由一个简单的全连接层(线性层)和激活函数(ReLU)组成，用于增加模型的表达能力。

为解决系数预测问题，本文中提出了一种能够从输入序列中提取重要特征并依赖时序信息进行精确的系数预测。该解码器包含了多头自注意力机制和长短期记忆网络(LSTM)，可以利用时序信息从输入序列中提取关键特征。该解码器的设计提升了增强模型的表达能力，同时避免过拟合，并提升了泛化能力。

嵌入向量首先经过第一层多头自注意力机制(MultiHeadAttention)。在这一层中，输入数据会与自身进行交互计算，通过注意力机制输出加权后的特征表示。再经过 dropout 层处理，以防止模型过拟合，并与原始输入相加，最终通过正则化，保证后续训练的稳定性。第二个多头自注意力机制中，解码器的输出与编码器的输出进行交互，从而引入编码器生成的全局上下文信息。与第一层类似，输出经过 dropout 层处理，并与上一层的输出进行相加和正则化。结构如图 7 所示。

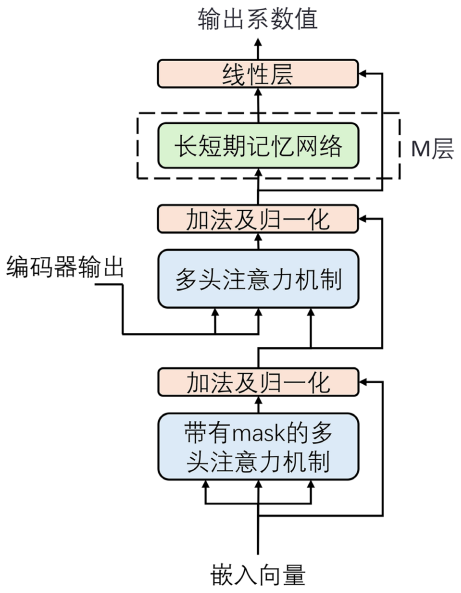


Figure 7. Decoder structure for coefficient prediction
图 7. 用于系数预测的解码器结构

在完成多头自注意力机制的处理后，解码器通过 LSTM (长短期记忆网络)对序列进行处理。LSTM 能够有效捕捉序列中的长期依赖关系。最后选取 LSTM 输出的最后一个时间步的结果作为最终的特征表示。将 LSTM 的输出通过一个全连接层映射到目标空间(维度等于系数的数量)，从而生成最终的系数预测值。

4.3. 基于遗传算法的表达式预测优化

在本研究中，解码器有两个输出，分别是表达式输出和系数输出。在训练时需要分别对两个输出进

行损失函数计算。预测阶段引入了束搜索(Beam Search)和遗传算法(Genetic Algorithm)来提升预测精度。图 8 描述了本研究中深度符号回归模型的训练和预测流程。在训练阶段，采用交叉熵损失函数和均方误差损失函数同时优化表达式及系数。

在预测阶段，利用束搜索算法和遗传算法融合的技术来保证生成更好的预测结果，进而提高预测能力。束搜索是一种在序列生成任务中常使用的启发式搜索算法，在机器翻译、文本生成等方面应用广泛。束搜索可保持若干条候选路径，增加了搜索空间的多样性，防止模型陷入局部最优。本模型采用束搜索方式对表达式预测阶段的表达式生成进行优化。束搜索在每个阶段生成后都会保留下一个阶段最优的若干候选序列并以其继续进行搜索，直至生成一条表达式。束搜索的这样的策略使得本模型表达式生成能力高，不生成重复序列或无意义序列。遗传算法是一种模拟自然选择机制的全局优化算法，常用于优化复杂函数或在大规模搜索空间中寻找最优解。本模型中进化算法通过模拟基因突变、交叉等方式，优化系数的预测值，几轮遗传之后，遗传算法会趋向于得到一个全局最优附近的系数预测结果。遗传算法的应用可以提高系数预测的准确性。方法如图 8 所示。

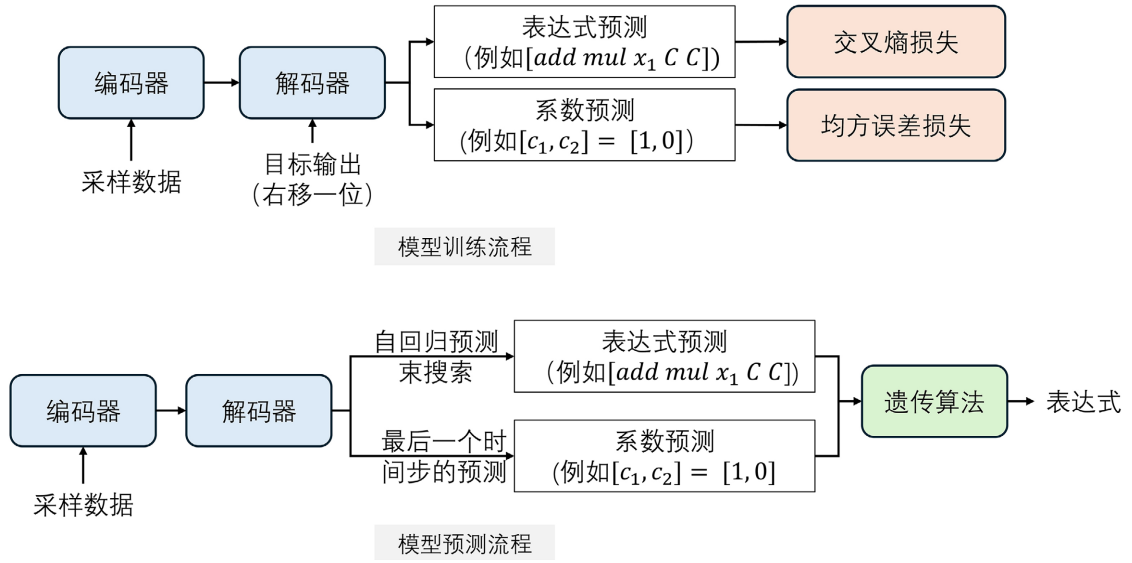


Figure 8. Model training and prediction process
图 8. 模型训练和预测流程

5. 实验和基准比较

本节中使用不同尺度的数据集进行本文提出的深度符号回归模型的训练，并基于 SDSR-Feynmandata 进行实验评估和分析。

5.1. 实验环境

实验环境为：windows11 64 位操作系统，处理器为 Intel (R) Core (TM) i9-10900X CPU @ 3.70 GHz，内存为 64.0 GB (63.7 GB 可用)，显卡为 NVDIA RTX A5000，显存为 24 GB。开发环境使用 Python 3.12，Pytorch 版本基于 CUDA 12.4。

5.2. 实验相关信息

实验使用了三种不同尺寸的数据集：10 K、50 K 和 100 K。每个数据集包括表达式骨架、系数值和采样数据。数据集被划分为 80%的训练集和 20%的测试集。

实验训练测试过程：首先按照相应的采样方法对各大小数据集(10 K、50 K、100 K)划分训练集、测试集，使其符合实验设置的要求，以数据集训练模型并记录损失函数值、准确率随训练轮次的变化。

最后，保存了训练模型及测试的性能，并对实验的数据及配置信息进行存储，保证了实验的重现性及可验证性。

5.3. 实验结果与分析

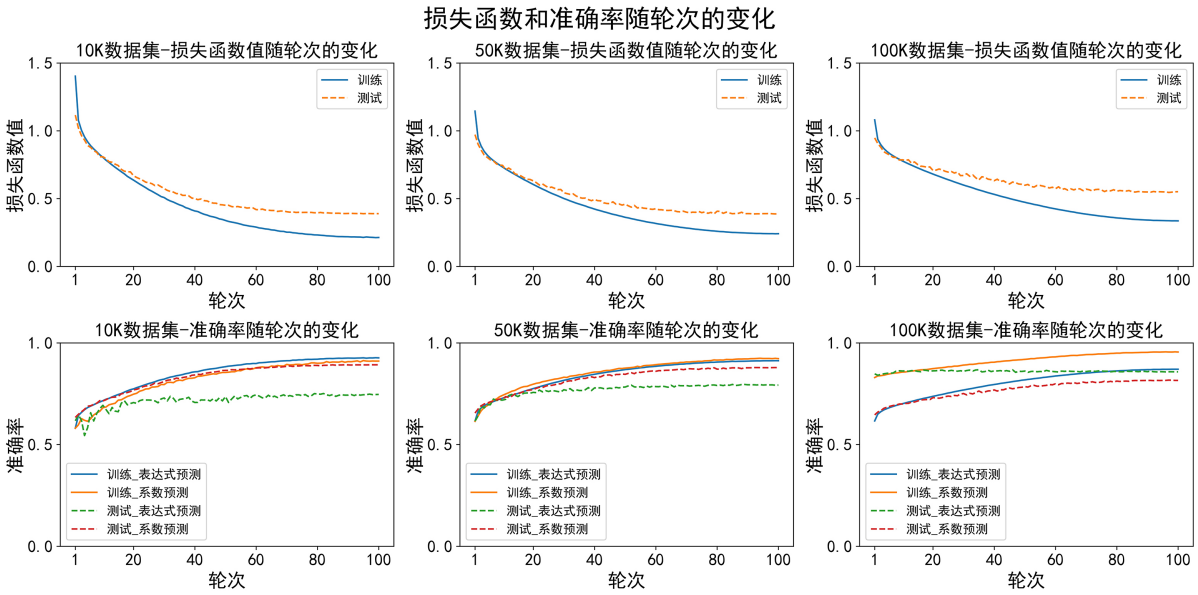


Figure 9. Training results of different training sets
图 9. 不同大小训练集的训练结果

本文提出的深度符号回归模型在不同数据规模的数据集中模型训练中都展现出良好的效果。从图 9 中明显看出，损失函数随训练轮次的表现出逐步下降的趋势。不同的数据集(10 K，50 K，100 K)中，随着训练次数的增加，损失函数值逐渐稳定，但是对于大的数据集(如 100 K 数据集)，损失函数随训练次数的下降逐渐缓慢，即模型在较大量的样本上存在一定的困难。

随着训练的进行，准确率随训练轮数的增大而增强，测试集准确率逐渐趋近于训练集准确率，说明模型训练过程中能够较好地对训练集数据的学习和泛化，也验证了深度符号回归模型具备良好的数据学习和预测能力，并且可以在不同规模数据集之间很好地学习和泛化。

在数据量较大的情况下，模型的训练越来越复杂。与 10 K 和 50 K 数据集相比，在 100 K 数据集情况下，模型训练需要更大的训练轮次进行训练；而数据量越大，在准确度和损失值上的提高也就越慢。训练集和验证集的损失函数值曲线都呈现了类似的下降趋势，通过比较训练集和验证集的准确率可以看出模型在训练过程中有较好的泛化能力。

5.4. 解码器的性能对比测试

本研究提出了一种新的解码器结构，专门针对符号回归任务进行了优化。该解码器可以更好地捕捉输入数据中的复杂关系，并精确地预测出相应的数学表达式及其系数。与传统的 Transformer 解码器相比，本研究的解码器可以提高系数的预测准确性。为了验证新解码器的有效性，我们利用 100 K 数据集进行了详细的实验测试，对比了本研究提出的解码器与经典的 Transformer 解码器的表现。以下是具体的

实验结果，见图 10 和表 1 所示：

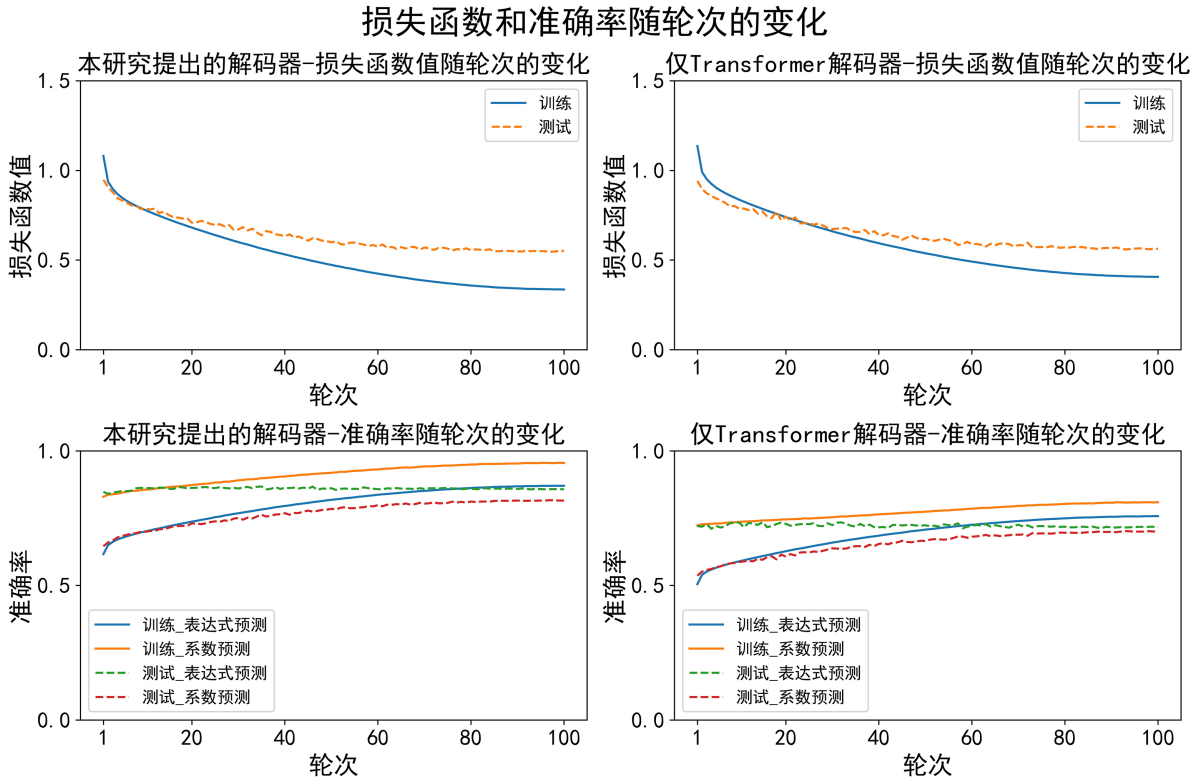


Figure 10. Training results of different decoders under 100 K dataset
图 10. 100 K 数据集下，不同解码器的训练结果

Table 1. Performance comparison between this study's decoder and the transformer decoder
表 1. 本研究的解码器方案与仅 Transformer 解码器方案的性能对比

解码器方案		损失	表达式准确率	系数准确率
本研究提出的解码器	训练	0.336	0.870	0.955
	测试	0.55	0.857	0.815
仅 Transformer 解码器	训练	0.407	0.757	0.809
	测试	0.563	0.716	0.700

从实验结果来看，本研究提出的解码器在训练损失、表达式准确率以及系数准确率方面均优于传统的仅 Transformer 解码器方案。训练阶段，本研究解码器的损失为 0.336，显著低于 Transformer 解码器的 0.407，同时表达式准确率和系数准确率分别达到 0.870 和 0.955，也明显高于 Transformer 解码器的 0.757 和 0.809。这表明新解码器在训练阶段具有更强的拟合能力和表达建模能力，能够更准确地学习目标函数的结构和参数。

在测试阶段，本研究的解码器方案依然展现出良好的泛化性能。尽管测试损失略有上升至 0.55，但其表达式准确率为 0.857，系数准确率为 0.815，均显著高于 Transformer 解码器的 0.716 和 0.700。尤其在系数预测方面的优势，说明本研究的解码器在处理符号回归中的数值回归任务时具有更高的稳定性和精度。这一结果验证了新解码器在实际符号回归应用中的有效性。

实验结果表明，本研究提出的解码器在深度符号回归任务中，相比于传统的 Transformer 解码器，具有更好的性能表现，尤其在预测系数的准确性和生成数学表达式方面表现突出。因此，本研究的解码器结构为深度符号回归提供了一种更有效的解决方案。

5.5. 基准测试

本节中利用 SRSD-Feynman 数据集进行基准试验[18]。这一数据集考虑了更现实和复杂的科学实验条件，为评价符号回归方法提供了一种更具有代表性的数据集。其基本数据是基于费曼物理讲义上的公式设计的，以便数据中的变量与常量具有物理意义；同时通过对每个变量的取值范围进行设计，使之在设计上尽量贴近真实的实验条件，避免传统数据集中的简化假设。其中作为常数出现的光速、万有引力常数等在数据集中被视为常量，不作为预测变量对待。

基于问题的复杂性，该数据集被分为三类：简单(Easy)、中等(Medium)和困难(Hard)。这里使用归一化编辑距离(Normalized Edit Distance, NED)进行评估。通常来讲归一化编辑距离越小，说明模型的效果越好。

这里的编辑距离是基于二叉树进行计算的。在计算时会将前缀表达式转化为对应的二叉树表示形式。对于两个二叉树，编辑距离会计算将一棵树 A 转换为另一棵树 B 的最低成本。这里的计算涉及到一系列对于树节点的操作，每个操作可以是插入、删除或重命名节点。在本研究中，表达式二叉树的节点可以是数学运算(例如，add、exp 作为符号)、变量符号或常数符号。通过使用归一化编辑距离作为评估指标，能够更好地反映模型在符号回归任务中的准确性。

实验中，使用 100 K 数据集训练的模型进行基准测试。表 2 展示了各方法在三种难度组别下的 NED 值。

Table 2. NED benchmark results
表 2. NED 基准测试结果

测量标准	组别	SRSD-Feynman							
		AFP	AFP-FE	AIF	DSR	E2E	uDSR	PySR	本研究
NED	简单	0.727	0.693	0.646	0.524	1.00	0.478	0.269	0.482
	中等	0.873	0.897	0.897	0.793	1.00	0.781	0.537	0.644
	困难	0.946	0.954	0.954	0.839	0.987	0.949	0.785	0.695

在简单组的测试中，本研究方法的 NED 值为 0.422，相对于 PySR 表现较差，但是仍高于其他方法。可以看出，本研究的深度符号回归方法在对于简单公式的预测能力稍差。本研究的训练数据集中的表达式长度大多在 8 到 12 的长度区间，和 SRSD-Feynman 数据集的简单组中数据集的复杂度和长度存在差距，因此导致本研究方法的泛化效果较差。

在中等组，本文的 NED 为 0.644，可以看出此时本文结果较简单组有一定的改进，但比其他较高性能的方法(如 PySR，uDSR 等)稍微差一些；在中等复杂度的公式建模上，深度符号回归方法比简单组方法有更广泛的应用性和推理能力，虽然结果不如 PySR，但是说明本研究方法在中等复杂公式建模上也有较好结果。

在困难组的测试中，本研究方法的 NED 值为 0.695。在与其他方法(如 DSR 和 PySR)的对比中效果最好。显然本研究的深度符号回归方法处理复杂公式能力更强，在困难组中最具优势。

本研究中训练数据集中的表达式多为复杂且较长(大于 10)的公式，因此模型更倾向于复杂公式。但是，SRSD-Feynman 数据集包含表达式长度通常不会超过 10，相对于本研究中是较短的。以上原因导致

了本研究提出的深度符号回归在处理简单和中等公式时未能展现出明显优势，但是训练数据集中的高复杂度公式为该方法提供了更多的训练和测试数据，使其能够更好地解决复杂问题。

根据以上分析可知，本文提出的深度符号回归方法在解决长、复杂的表达式时更好，并在复杂问题上有较好的表现。

6. 结论

本文基于端到端深度符号回归模型，结合 Transformer 网络和注意力机制提出一种端到端深度符号回归方法，该模型在较复杂的符号表达式表达时也能够表现良好。基于 SRSD-Feynman 数据集进行的基准测试结果表明，本研究的深度符号回归方法在不同难度组别下均展现出了一定的优势。在简单组中，尽管深度符号回归方法的表现一般，但在中等组和困难组中性能逐渐提升。本研究中的深度符号回归方法更适合处理较长且复杂的表达式，在处理复杂问题时表现更好。

参考文献

[1] Koza, J. (1994) Genetic Programming as a Means for Programming Computers by Natural Selection. *Statistics and Computing*, **4**, 87-112. https://doi.org/10.1007/bf00175355

[2] Keijzer, M. (2003) Improving Symbolic Regression with Interval Arithmetic and Linear Scaling. In: Ryan, C., Soule, T., Keijzer, M., Tsang, E., Poli, R. and Costa, E., Eds., *Lecture Notes in Computer Science*, Springer, 70-82. https://doi.org/10.1007/3-540-36599-0_7

[3] Vladislavleva, E.J., Smits, G.F. and den Hertog, D. (2009) Order of Nonlinearity as a Complexity Measure for Models Generated by Symbolic Regression via Pareto Genetic Programming. *IEEE Transactions on Evolutionary Computation*, **13**, 333-349. https://doi.org/10.1109/tevc.2008.926486

[4] McConaghy, T. (2011) FFX: Fast, Scalable, Deterministic Symbolic Regression Technology. In: Riolo, R., Vladislavleva, E. and Moore, J., Eds., *Genetic and Evolutionary Computation*, Springer, 235-260. https://doi.org/10.1007/978-1-4614-1770-5_13

[5] Schmidt, M. and Lipson, H. (2009) Symbolic Regression of Implicit Equations. In: Riolo, R., O'Reilly, UM. and McConaghy, T., Eds., *Genetic and Evolutionary Computation*, Springer, 73-85. https://doi.org/10.1007/978-1-4419-1626-6_5

[6] Gupta, R., Srivastava, D., Sahu, M., Tiwari, S., Ambasta, R.K. and Kumar, P. (2021) Artificial Intelligence to Deep Learning: Machine Intelligence Approach for Drug Discovery. *Molecular Diversity*, **25**, 1315-1360. https://doi.org/10.1007/s11030-021-10217-3

[7] Angelis, D., Sofos, F. and Karakasidis, T.E. (2023) Artificial Intelligence in Physical Sciences: Symbolic Regression Trends and Perspectives. *Archives of Computational Methods in Engineering*, **30**, 3845-3865. https://doi.org/10.1007/s11831-023-09922-z

[8] Makke, N. and Chawla, S. (2024) Interpretable Scientific Discovery with Symbolic Regression: A Review. *Artificial Intelligence Review*, **57**, Article No. 2. https://doi.org/10.1007/s10462-023-10622-0

[9] Rudin, C. (2019) Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead. *Nature Machine Intelligence*, **1**, 206-215. https://doi.org/10.1038/s42256-019-0048-x

[10] Lample, G. and Charton, F. (2019) Deep Learning for Symbolic Mathematics. arXiv:191201412.

[11] Biggio, L., Bendinelli, T., Lucchi, A., *et al*. (2020) A Seq2Seq Approach to Symbolic Regression. *Proceedings of the Learning Meets Combinatorial Algorithms at NeurIPS*2020.

[12] Wu, M., Li, W., Yu, L., *et al*. (2023) Discovering Mathematical Expressions through DeepSymNet: A Classification-Based Symbolic Regression Framework. *IEEE Transactions on Neural Networks and Learning Systems*.

[13] Udrescu, S. and Tegmark, M. (2020) AI Feynman: A Physics-Inspired Method for Symbolic Regression. *Science Advances*, **6**, eaay2631. https://doi.org/10.1126/sciadv.aay2631

[14] Udrescu, S.-M., Tan, A., Feng, J., *et al*. (2020) AI Feynman 2.0: Pareto-Optimal Symbolic Regression Exploiting Graph Modularity. *Advances in Neural Information Processing Systems*, **33**, 4860-4871.

[15] Kamienny, P.-A., D'ascoli, S., Lample, G., *et al*. (2022) End-to-End Symbolic Regression with Transformers. *Advances in Neural Information Processing Systems*, **35**, 10269-10281.

[16] Lalande, F., Matsubara, Y., Chiba, N., *et al*. (2023) A Transformer Model for Symbolic Regression towards Scientific

Discovery. arXiv:231204070.

[17] Vastl, M., Kulhánek, J., Kubalík, J., Derner, E. and Babuška, R. (2024) SymFormer: End-to-End Symbolic Regression Using Transformer-Based Architecture. *IEEE Access*, **12**, 37840-37849. https://doi.org/10.1109/access.2024.3374649

[18] Matsubara, Y., Chiba, N., Igarashi, R., *et al*. (2022) Rethinking Symbolic Regression Datasets and Benchmarks for Scientific Discovery. arXiv:220610540.