Symbolic Regression with Diffusion Models

Matthew Badal-Badalian, Argho Das, Pooja Krishan, Chris Verghese

Main Contributions

Our main contributions are:

1. Implemented a diffusion model framework for symbolic regression, leveraging holistic sequence processing and structured noise diffusion for accurate formula reconstruction.
2. Developed a grounded synthetic dataset inspired by real-world equations to enable controlled benchmarking and evaluation of symbolic regression methods.
3. Conducted a rigorous comparative study of diffusion models, genetic programming, and SymbolicGPT, demonstrating the diffusion model's superior balance of accuracy, robustness, and computational efficiency, while identifying areas for future research.

Tools and Technologies Used

This project was implemented using Python as the primary programming language. PyTorch was utilized for building and training the diffusion model, while gplearn facilitated genetic programming. The Hugging Face Transformers library was used to simulate SymbolicGPT functionality. Data preprocessing and manipulation were performed with NumPy and Pandas, and Matplotlib was employed for visualizations. Experiments were conducted on Google Colab with GPU acceleration, and scikit-learn was used for evaluation metrics such as Mean Squared Error (MSE).

Project Implementation Details

Check the boxes that best describe your project:

- ■ We used an existing implementation of the algorithm(s).
- ■ We implemented the algorithm(s) ourselves.
- ■ We used an existing dataset.
- ■ We created a new dataset.

Abstract

Symbolic regression involves discovering mathematical expressions that accurately represent relationships within data. This project introduces a novel framework leveraging diffusion models for symbolic regression, addressing challenges like inefficiencies in traditional genetic programming and auto-regressive limitations in transformer-based approaches such as SymbolicGPT. Our methodology involves tokenizing mathematical expressions into sequences, and applying forward and reverse diffusion processes to refine noisy embeddings back into interpretable mathematical formulas. We evaluated three different approaches to solving symbolic regression. One approach was based on a synthetic dataset that we generated. The other two approaches used an existing dataset. Approach 3 which used a text diffusion model seems to perform better than the other approaches as seen from the BLEU, token similarity and edit distance scores. We focus only on reconstructing the formula and do not perform numerical optimization to determine the constants and hence are unable to benchmark against existing state-of-the-art methods.

1 Introduction

The goal of symbolic regression is to discover mathematical relationships in data. While standard regression techniques are format constrained, using only linear or polynomial expressions, symbolic regression is format-free. It uses a much larger set of candidate equations and, more importantly, allows for nonlinear combinations of the operations that make up the equations. These operations include not just the usual addition, multiplication, and division, but also exponentiation, using variables as bases, and even trigonometric functions. Yet widely useful, symbolic regression faces some hard computational problems known to be difficult in computer science and engineering. The main one is equation searching, through an enormous pool of candidate equations, in a space that cannot be effectively navigated with heuristic methods and where the true (or at least a better) solution is often obscured by false positives. A common way of getting around these problems is with genetic programming (GP).

Moreover, GP has a hard time dealing with information-rich data and is often unable to produce equations that generalize well to new and previously unseen situations. Recent advancements in AI and deep learning have introduced innovative new methods for symbolic regression, one of the most promising being SymbolicGPT [1]. This is a transformer-based model that reframes symbolic regression as a sequence generation task and produces mathematical expressions one token at a time. Despite achieving a high level of accuracy and adaptability, not to mention a degree of power and elegance, the sequential nature of this model makes it somewhat inefficient when it comes to handling longer expressions.

This project investigates whether the diffusion model can be employed in the context of symbolic regression. This is a novel and unexpected use of the kind of deep neural networks that have recently been used for tasks like image generation. We generated a synthetic dataset that is based on real scientific equations to assess how well diffusion models perform in symbolic regression. We also use an existing subset of dataset from the Symbolic GPT paper [1] with three variables to evaluate our approach.

2 Related Work

For decades, genetic programming (GP) has been the dominant method in the area of research known as symbolic regression [2]. This is because GP has a unique ability to derive not just mathematical expressions—like those found in ordinary regression techniques—but also interpretable and flexible models that can explain relationships within data [3]. GP starts with a randomly generated population of candidate equations. At each generation, it uses evolutionary operators, such as mutation and crossover, to refine these populations. The idea is to minimize error, of course, but also to maintain population diversity, which is crucial for avoiding local optima.

Advancements in the application of diffusion models in recent years have illuminated promising new pathways for the development of symbolic regression methods [4]. They have made clear the necessity—and indeed, the power—of basal domain knowledge when working with structured data. The studies we looked to for inspiration showed us how to work noise into a symbolic regression task so that the types of structures we were looking for—i.e., mathematical expressions—could still be found even with the noise present [4]. Beyond this, we found guidance in recent efforts to make these denoising processes more organized and effective [5], also ensuring that the type of tidy data needed for symbolic regression was present at the right moments in the forward and reverse diffusion process [6].

Additionally, the integration of neural networks into the framework of symbolic regression has shown considerable potential to merge deep learning capabilities with human-understandable outputs. A notable example is the Equation Learner (EQL) network [7], which trains models end-to-end through backpropagation. This network incorporates activation functions designed for mathematical operations and uses sparsity regularization to enhance both interpretability and generalization. Recent work by Makke and Chawla (2024) [8] further highlights how deep learning approaches can overcome the scalability and computational limitations of traditional methods like genetic programming. Building on these advancements, our approach leverages diffusion models for symbolic regression. By employing tokenization and embedding techniques, we convert mathematical expressions into structured representations that are well-suited for processing with diffusion-based methods.

These papers [1, 9] played a pivotal role in our project as a transformer-based benchmark for symbolic regression, showcasing an innovative approach by treating equation discovery as a sequence generation problem. Its GPT-based architecture generates equations token by token, utilizing a two-stage process of skeleton equation generation and constant optimization to ensure scalability and computational efficiency. Drawing inspiration from this framework, we implemented similar tokenization methods to convert mathematical expressions into structured sequences. However, SymbolicGPT's reliance on sequential token generation introduced challenges with dependency management, which we addressed by adopting diffusion models capable of processing sequences holistically. This comparative study allowed us to evaluate computational efficiency, robustness to noise, and scalability of diffusion models alongside SymbolicGPT. SymbolicGPT's focus on handling high-dimensional data informed our approach to embedding and optimization strategies, enabling a comprehensive assessment. By leveraging insights from SymbolicGPT, we advanced diffusion models as a robust alternative for symbolic regression.

The idea of leveraging diffusion models for sequence reconstruction [10], as explored in this paper [11], inspired critical aspects of our symbolic regression framework. In this work, diffusion models introduce noise to embeddings and iteratively denoise them to generate coherent text sequences. We adapted this concept to symbolic regression by introducing noise into mathematical formula embeddings during a forward diffusion process. Reverse diffusion process was trained to reconstruct original formulas, ensuring robustness to noise and accurate equation recovery. Additionally, gradient-based optimization techniques used in text denoising informed our approach to refining formula reconstruction, aligning noisy embeddings with their true representations. This holistic sequence-processing approach enabled us to overcome sequential token generation limitations of transformer-based models like SymbolicGPT and addressed inefficiencies seen in genetic programming.

3 Methodology

This project focuses on employing diffusion models for symbolic regression, addressing limitations of traditional methods such as genetic programming and transformer-based approaches like SymbolicGPT. The methodology was designed to improve accuracy, interpretability, and computational efficiency of symbolic regression by integrating advanced techniques for tokenization, embedding, noise diffusion, and model evaluation. In the following, we present a detailed account of our implementation.

3.1 Dataset Description and Preprocessing

3.1.1 Generation of a synthetic dataset

In order to tackle this problem, several approaches were considered. For experimental purposes, a synthetic dataset was created which comprised of 100 mathematical functions with varying levels of complexity, including randomly generated functions with typical mathematical operations (i.e. addition, multiplication, exponential, trigonometric functions, etc), and physics-inspired equations with scientific constants. The design of some randomly generated equations was partially inspired by the Nguyen benchmark equations, a well-known benchmark in symbolic regression [12]. Furthermore, 100 multivariate input (1-5 variables) and univariate output pairs values was then generated for each formula. In this synthetic dataset, a single formula capturing the relationship between a set of 100 input and output pairs represents a single data point.

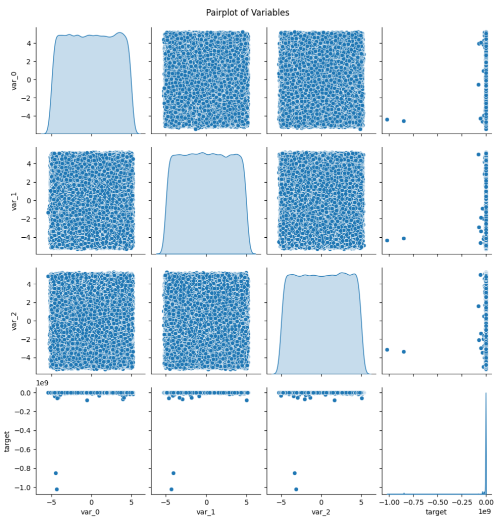
3.1.2 Analysis into an existing dataset

Due to limited resources to train the diffusion model on a large dataset, and in order to test the validity of the approaches in an agile manner, we used a small subset of the dataset used by Valipour et. al [1]. The dataset consists of three folders train, val, and test each consisting of json files representing a formula. Each formula consists of 3 variables ranging from -5.0 to +5.0 and between 0 to 3 constants. There are 100 samples corresponding to each formula within a single json file. The train folder consists of 747 json files each representing a formula with 100 examples. The validation folder consists of 160 json files for validation to inform the training, and the test folder consists of 161 json files to be used in evaluation.

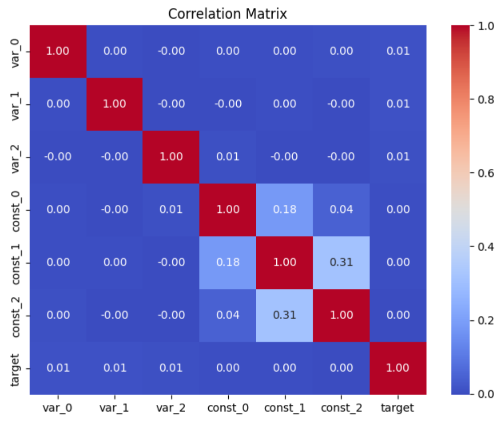
We were especially interested in the points key of the json files containing the variable values and target y values for 100 examples. We were also interested in the human readable formula field of the json files. In addition, the train and test folders consist of a properties file that contains metadata information about the formula generation process using a parse tree such as the depth, operators used and so on.

On conducting an initial exploratory data analysis of the variables, constant and target values extracted into a Python data frame to generate visualizations and glean insights, we can notice that there are no missing values in the dataset.

A pair plot between the three variables and target values in the training set is shown in Figure 1a.



(a) Pair-plot to identify the relationships between the variables and target values



(b) Correlation matrix capturing the relationship between the variables and target values

Figure 1: Exploratory data analysis to identify relationships in the data

It is evident from this figure that the variables have a roughly uniform distribution. The variables conform to the bounds specified in the json files and are between -5.0 and +5.0. The scatterplots show no obvious linear or non-linear correlation between the variables. The dense evenly spaced grids point to an independence between these variables. It also shows that the target variable has a skewed and extreme distribution.

The correlation matrix shown in Figure 1b also echoes the same analysis as earlier indicating that the variables are not correlated with each other and the target. This shows that symbolic regression is an apt task to model the relationship between the variables and the target to try to reconstruct the formula.

3.2 Approach 1: A Simple Diffusion Model Using Synthetic Data

The tokenization process used a regex-based tokenization approach token identification to decompose mathematical expressions into their fundamental components, including constants, variables, operators, and functions. Following this, a Word2Vec model was trained on the tokenized symbolic expressions to generate dense vector representations for each token, the input values are padded by NAN entries (to ensure consistent dimensions), and the tokens are padded. Finally, a dictionary is created to store the token embeddings, the input and output values, as well as a mask which indicates which input features are real numbers (non NAN).

The tokenized formula embeddings was then passed through the forward diffusion process, where Cosine noise was progressively added to formula embeddings over a series of diffusion steps controlled by a noise scheduler. During the reverse diffusion process, the corrupted embeddings undergo an iterative denoising process in order to construct tokenized formula embeddings. The process begins at the final timestep and works backwards towards the initial timestep. At each timestep, the embeddings in their current state are then passed through forward pass model in order to predict the noise. The predicted noise is then used along side time dependent parameters (i.e. α_t, β_t) which follow the Cosine noise schedule to compute the mean of the denoised embeddings for the previous timestep. This backward progression continues until the process reaches the initial timestep, where the fully denoised embeddings are obtained. Here, the Mean Squared Error (MSE) was calculated between the generated tokenized formula embeddings and the actual tokenized formula embeddings as the loss metrics.

This model was tested during evaluation by providing noisy embeddings that would initiate the reverse diffusion process. The embeddings generated through reverse diffusion were decoded back into mathematical formulas using precomputed vocabulary. This process allowed us to assess our model's ability to generalize beyond training data and accurately reconstruct symbolic equations.

3.3 Approach 2: Implementing Hybridized tokenization and self Attention

In this approach, the tokenization approach was combined with hierarchical parsing. This model was trained using dataset used by Valipour et. al [1]. This dual approach ensured that structural and syntactic properties of formulas were preserved. To standardize representation, each token was assigned a unique ID from a custom vocabulary.

$$Tokens_{hierarchical} = PreorderTransversal(Tree(Formula))$$

$$Tokens_{regex} = Regex(Formula)$$

$$Tokens_{final} = Concat(Tokens_{hierarchical}, Tokens_{regex})$$

The embedding system assigns a unique token embedding for tokens in a dynamically constructed vocabulary. Tokens in a formula are padded using [PAD] to ensure a constant token length for uniform input dimensions for the diffusion models. After getting this list of unique IDs to tokens in the vocabulary are mapped to a vector in a 128-dimension space using a pytorch neural network embedding layer. The diffusion model architecture is designed to learn the process of transforming noisy data back into its original form through iterative refinement, leveraging the principles of denoising and probabilistic modeling. The architecture essentially includes : 1. an input projection that projects the input embeddings into a higher dimension, 2. a multi head self attention mechanism that captures the relation between tokens and the data points allowing the model to focus on important parts, and 3. a feed-forward network which processes the self attention output to generate embeddings. At each timestep, noise is applied based on the current step's β value, progressively corrupting the data to a nearly unstructured state. In parallel, the model is trained to reverse this process: starting from noisy data, it predicts the noise at each timestep and refines the input to move closer to the original data. This denoising objective is optimized using a loss function, such as Mean Squared Error (MSE).

The noise scheduler controls the addition and removal of the noise by using a linear schedule for noise variance allowing progressive corruptions and reconstruction of the data. The Linear scheduler that interpolates the noise between 1 x 10 $^{-4}$ and 2 x 10 $^{-2}$ accross the time steps T. The calculated α and β, are used to corrupt the data progressively over every timestep t using:

$$x_t = \sqrt{\bar{\alpha}_t} \cdot x + \sqrt{1-\bar{\alpha}_t} \cdot \epsilon, \text{ Where } \bar{\alpha}_t = \prod_{i=1}^{t} (1-\beta_i), \tag{1}$$

The reverse diffusion process starts with a highly noisy x and iteratively denoises it to reconstruct the original data x. At each timestep, the denoised data x $_{t-1}$ is calculated as:

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \cdot \epsilon_\theta(x_t, t) \right) \tag{2}$$

Using the above formula, the reverse process aims to iteratively reconstruct a formula using noisy embeddings using the data points for any given test data.

3.4 Approach 3: Implementing a Text Diffusion Model using an Existing Dataset

This approach explores the possibility of using a text-based diffusion model to solve the task of symbolic regression since most used and researched state-of-the-art diffusion models handle continuous embedding representations of the inputs during the noising and denoising process. But in the case of symbolic regression for the diffusion model, we are working with a discrete data space after tokenizing the formula. We built a custom diffusion model inspired by other text diffusion models out there such as DiffusionLM [11]. Initially, we tokenize the human readable formula by parsing it through a regular expression to identify the tokens. We created a vocabulary mapping that mapped each token to an integer. We also added an End-of-Sequence *EOS* token, and padding *PAD* tokens to indicate the end of the formula and pad shorter formulae to the longest formula. The forward diffusion process progressively adds noise to the tokenized formula representation. We used the T Net architecture from the SymbolicGPT paper [1] to generate an order-invariant representation of the data points that is independent of the number of samples in the dataset and outputs an embedding vector of size 1*(embedding_dimension). We feed in these T Net embeddings together with the noisy formula to the reverse diffusion process in the hope that the model can learn the noise added in the forward pass conditioned on the input embeddings. The model architecture is shown in Figure 2.

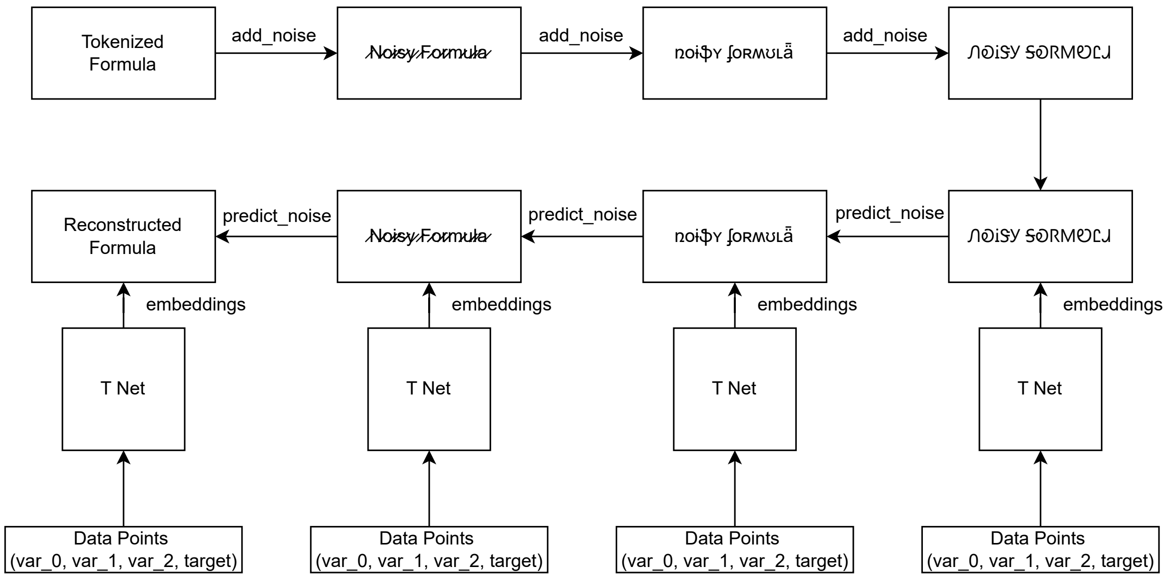


Figure 2: Architecture of the Text Diffusion Model

4 Experiments and Results

Experiments were designed to evaluate performance of diffusion models for symbolic regression compared to traditional genetic programming and transformer-based approaches such as SymbolicGPT. Our primary objectives were to evaluate models on accuracy, robustness, and computational efficiency using a scientifically grounded synthetic dataset inspired by real-world equations.

In both approaches 1 and 2, the MSE was calculated by comparing the generated tokenized formula embeddings with the actual tokenized formula embeddings. This was trained over 100 epochs, with an ambedding size of 128. Early stopping caused training in Approach 1 to finish at Epoch 27 with a training loss of 0.01776, validation loss of 0.4217 and a testing loss of 0.8462. Approach 2 produced a best training loss of 0.085 and a validation loss of 0.21. In approach 3, the embedding dimensions for the T Net model is set to 128 which is different from the Symbolic GPT paper [1] which used 512. Early stopping was implemented based on the validation loss with a patience of 10. The model trained for 572 epochs, and the best train loss is 0.0070, and val loss is 0.0091. A cross-entropy loss function was used. These are shown in the Figure 3. The model was evaluated on the test set consisting of 161 formulae each consisting of 100 examples. An example prediction is shown in Figure 4.

Metric	**Approach 3**	**Approach 2**	**Approach 1**
BLEU Score	0.25	0.011	0.023
Token Similarity	0.61	-	0.030
Levenshtein Edit Distance	7.22	19.09	12.42

Table 1: Evaluation Metrics for Model Performance

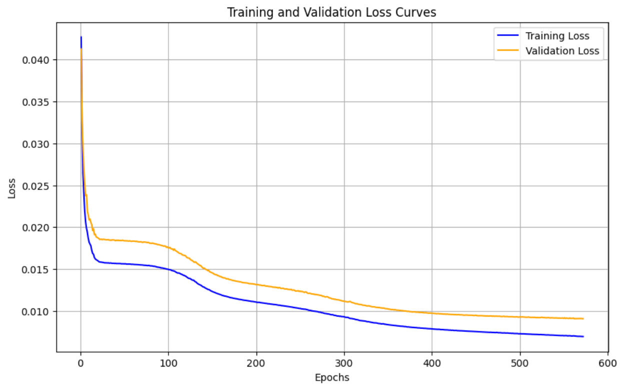
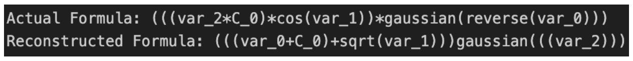


Figure 3: Loss curves for Approach 3



```
Actual Formula: (((var_2*C_0)*cos(var_1))*gaussian(reverse(var_0)))
Reconstructed Formula: (((var_0+C_0)+sqrt(var_1)))gaussian(((var_2)))
```

Figure 4: Example prediction of model used in Approach 3

5 Discussion and Conclusion

Approach 1 makes use of synthetic dataset while being trainable over a multiple possible variables up to 5. A possible improvement to approach 1 would be to pass the trained tokenized formula embeddings, as well as the input-output pairs seperately into a TNet model, which would then allow a cross-attention mechanism to be implemented. This would allow an alternative way to condition the denoising process on the datapoints with the goal of better extracting relevant features. This method was explored, but not fully realized.

Approach 2 makes use of multi headed self attention in the diffusion model and makes use of hybridized tokenization using regex and hierarchical tokenization.

Approach 3 which uses a text diffusion model is better suited for the task of symbolic regression since it operates on a discrete problem space specifically catered to language tasks. The weakness of using this approach is the limited research available in this field, which made it challenging to adapt it to this specific problem statement. Another limitation is the size and complexity of the dataset available for training the diffusion model which is limited to only 747 example formulae and 3 variables.

The goal of this project was to develop a logical model architecture that can produce reasonable results on the small subset of the Symbolic GPT dataset [1] in the hope that this can be extended to a larger dataset given more compute resources. So, the approaches used focused only on reconstructing the formula and did not work on numerical optimization methods for the constants.

Approach 3 is seen to have a better overall performance than approach 1 and approach 2. We are unable to benchmark against existing state-of-the-art architectures due to lack of comparable metrics since we do not optimize over the constants and hence cannot use MSE as an evaluation metric.

References

[1] Mojtaba Valipour et al. “Symbolicgpt: A generative transformer model for symbolic regression”. In: *arXiv preprint arXiv:2106.14131* (2021).

[2] Michael D Schmidt and Hod Lipson. “Age-fitness pareto optimization”. In: *Proceedings of the 12th annual conference on Genetic and evolutionary computation*. 2010, pp. 543–544.

[3] William La Cava, Lee Spector, and Kourosh Danai. “Epsilon-lexicase selection for regression”. In: *Proceedings of the Genetic and Evolutionary Computation Conference 2016*. 2016, pp. 741–748.

[4] Jacob Austin et al. “Structured denoising diffusion models in discrete state-spaces”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 17981–17993.

[5] Richard E Turner et al. “Denoising Diffusion Probabilistic Models in Six Simple Steps”. In: *arXiv preprint arXiv:2402.04384* (2024).

[6] Aaron Lou, Chenlin Meng, and Stefano Ermon. “Discrete Diffusion Modeling by Estimating the Ratios of the Data Distribution”. In: *Forty-first International Conference on Machine Learning*. 2024.

[7] Samuel Kim et al. “Integration of Neural Network-Based Symbolic Regression in Deep Learning for Scientific Discovery”. In: *IEEE Transactions on Neural Networks and Learning Systems* 32.9 (2021), pp. 4166–4177.

[8] Nour Makke and Sanjay Chawla. “Interpretable scientific discovery with symbolic regression: a review”. In: *Artificial Intelligence Review* 57.1 (2024), p. 2.

[9] Martin Vastl et al. “Symformer: End-to-end symbolic regression using transformer-based architecture”. In: *IEEE Access* (2024).

[10] Zalan Fabian, Berk Tinaz, and Mahdi Soltanolkotabi. “Adapt and Diffuse: Sample-adaptive Reconstruction via Latent Diffusion Models”. In: *Proceedings of machine learning research* 235 (2024), p. 12723.

[11] Xiang Li et al. “Diffusion-lm improves controllable text generation”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 4328–4343.

[12] Luiz Otavio VB Oliveira et al. “Analysing symbolic regression benchmarks under a meta-learning approach”. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 2018, pp. 1342–1349.