Symbolic-Diffusion: Deep Learning Based Symbolic Regression with D3PM Discrete Token Diffusion

Ryan T. Tymkow
Department of Mechatronics Engineering
University of Waterloo
Waterloo, ON Canada, N2L 3G1
rtymkow@uwaterloo.ca

Benjamin D. Schnapp
Department of Mechatronics Engineering
University of Waterloo
Waterloo, ON Canada, N2L 3G1
bdschnapp@uwaterloo.ca

Mojtaba Valipour
Department of Computer Science
University of Waterloo
Waterloo, ON Canada, N2L 3G1
mojtaba.valipour@uwaterloo.ca

Ali Ghodshi
Department of Statistics and Actuarial Science
University of Waterloo
Waterloo, ON Canada, N2L 3G1
ali.ghodsi@uwaterloo.ca

Abstract

Symbolic regression refers to the task of finding a closed-form mathematical expression to fit a set of data points. Genetic programming based techniques are the most common algorithms used to tackle this problem, but recently, neural-network based approaches have gained popularity. Most of the leading neural-network based models used for symbolic regression utilize transformer-based autoregressive models to generate an equation conditioned on encoded input points. However, autoregressive generation is limited to generating tokens left-to-right, and future generated tokens are conditioned only on previously generated tokens. Motivated by the desire to generate all tokens simultaneously to produce improved closed-form equations, we propose Symbolic Diffusion, a D3PM based discrete state-space diffusion model which simultaneously generates all tokens of the equation at once using discrete token diffusion. Using the bivariate dataset developed for SymbolicGPT, we compared our diffusion-based generation approach to an autoregressive model based on SymbolicGPT, using equivalent encoder and transformer architectures. We demonstrate that our novel approach of using diffusion-based generation for symbolic regression can offer comparable and, by some metrics, improved performance over autoregressive generation in models using similar underlying architectures, opening new research opportunities in neural-network based symbolic regression.

1 Introduction

Symbolic regression refers to the task of finding a closed-form mathematical expression to fit a set of data points, without fixing the mathematical structure of the equation in advance [1]. Traditional algorithms for this task have primarily utilized genetic-programming based methods to solve this problem, where, inspired by biological evolution, populations of candidate expressions evolve, crossover, and mutate towards maximizing a fitness function [2]. While genetic-programming based methods have proven flexible and capable at the task of symbolic regression, and remain the state-of-the-art in terms of the quality of the generated equations [3], they often suffer from being computationally expensive and are typically slow

at inference time as it is necessary to search the entire mathematical space to find the right equation for each dataset [4].

Deep-learning-based approaches have received attention in recent research as an alternative to genetic-programming for symbolic regression. While the performance of deep-learning-based symbolic regression has not been able to match the quality of state-of-the-art genetic-programming-based approaches yet, these methods have excelled at providing decently high quality outputs with significantly faster, near real-time inference [4].

Conventionally, the majority of deep learning methods have used autoregressive generation conditioned on encoded input points to produce a tokenized output form of the equation [4–6].

Diffusion models, which function by learning the reverse process to generate data from noise [7], have shown exceptional performance in generative tasks on continuous domains such as image generation [8]. Recently, discrete-space diffusion models have been developed which attain reasonably strong performance in discrete domains such as text generation [9–11], and has achieved state-of-the-art performance in domains such as music generation [12] or DNA sequencing [13]. The objective of this work was to be the first to develop the a novel symbolic regression model based on discrete token diffusion, and to evaluate the performance of discrete token diffusion for the domain of symbolic regression compared to autoregressive generation.

1.1 Motivation

While autoregressive models have demonstrated strong performance across many domains, autoregressive generation remains limited by the sequential generation process, as each future token generated is only conditioned on previously generated tokens, and lacks global context which is needed in the context of Symbolic Regression.

Discrete state-space diffusion offers a different approach, iteratively denoising the entire sequence simultaneously, allowing the entire sequence to have global context throughout the generation process. By leveraging the global context available during discrete token diffusion, we hypothesized it may be possible to attain more accurate and syntactically valid symbolic expressions versus autoregressive generation models.

1.2 Contributions

In this paper, we present Symbolic Diffusion (Figure 1), which is a symbolic regression model that uses diffusion rather than autoregressive token generation to predict the symbolic form of an equation. This model enables global context and simultaneous generation of all tokens during symbolic regression compared to previous autoregressive approaches.

Symbolic Diffusion utilizes a Pointnet-style feature encoder [14], followed by a transformer-based decoder designed to predict the ground truth token through a series of denoising steps utilizing the D3PM [9] discrete state space diffusion architecture .

Symbolic Diffusion was compared to an autoregressive model based on SymbolicGPT [5] featuring an identical encoder and transformer architecture, but conditioned to produce tokens sequentially in an autoregressive manner as a GPT model [15], versus our models simultaneous diffusion-based generation of all tokens through denoising. It was found that Symbolic Diffusion performed comparably to the autoregressive baseline, and in some metrics such as mean R^2 value, attained a statistically significantly higher value.

Additionally, we provide an open-source implementation of our model, available at https://github.com/rtymk/Symbolic_Diffusion.

1.3 Related work

Early work on symbolic regression was based on the concepts of genetic programming, inspired by biological evolution. Koza (1994) [2] applied Darwinian principles of evolution to computational algorithms, with natural selection, mutations, and reproduction with crossover

adapting symbolic equations to produce one which best fits a given set of data. This sparked the birth of classical approaches to symbolic regression.

Genetic algorithms experienced several significant advances over the years improving their performance, such as offspring selection and age layered population structures [16] which helped increase the "genetic diversity" of generated equations, improving the performance.

More recently, Generative Pre Training (GPT) models [17] were applied to the domain of symbolic regression in the paper SymbolicGPT by Valipour et al. [5]. Autoregressive generation utilizing the transformer architecture [18] was used to generate symbolic equations conditioned on input coordinate data. T-Net embeddings [14] were used to embed the spatial data from the input coordinates. SymbolicGPT was shown to perform favorably compared to some genetic programming approaches in both performance and accuracy.

State of the art neural network (Symformer [6], E2E Symbolic Regression [4]) use autoregression with end-to-end approaches where constants are predicted to pre-initalize classical constant fitting algorithms, unlike SymbolicGPT [5] where the model does not predict constants prior to using classical constant fitting. (Our model was not compared with these state of the art models, as training an equivalent model required significantly larger datasets and more compute resources than we had available.)

Diffusion [7] machine learning models were developed by applying concepts from non-equilibrium thermodynamics to machine learning, and work by learning the backwards process of removing added noise to attain a generated output. Diffusion models have shown excellent results in generative application in continuous domains, specifically image generation [19].

The continuous nature of denoising in diffusion has posed challenges for applying diffusion to discrete domains such as token generation. Several approaches have been developed for applying diffusion to discrete domains. D3PM (2021) [9] approached diffusion to continuous domains by applying a random probability of corrupting a token to another token during the noising phase, and learning to correct those corruptions during denoising, conditioned on an input. DiffuSeq (2022) [11] functions by embedding the discrete tokens into a continuous space, noising and learning to denoise the continuous space, and extracting generated tokens from the denoised continuous space. Score entropy discrete diffusion (SEDD) (2024) [10] works by corrupting discrete data with noise, and de-noises using a novel loss function based on the ratio of probabilities of the data between time steps.

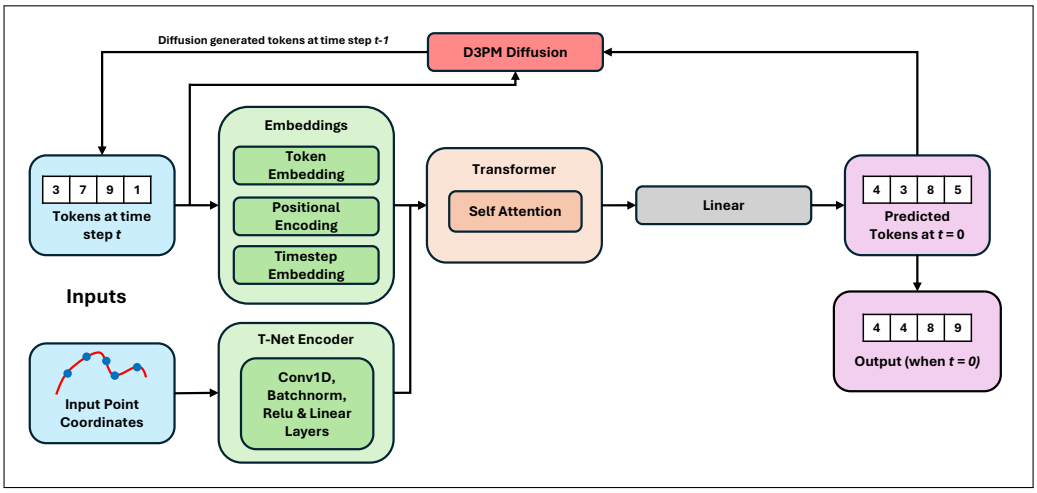

Figure 1: Symbolic Diffusion Architecture

2 Methodology

2.1 Dataset

As a relatively limited computational resources were available in this research, we tried to ensure that the proposed method and the baseline are compared in the same exact setting. Therefore, a bivariate (2 independent variable) dataset was used for training and evaluating the models below. The dataset from SymbolicGPT [5] was used for training and evaluating both models. A total of 500,000 samples were used, split 90/5/5 between train/test/validate. Each sample consisted of 200 coordinate points. Samples which contained infinity or NAN values were discarded.

2.1.1 Tokenization

All equations generated were converted to postfix (Reverse Polish Notation) [20] before being tokenized. Operators and variables were assigned unique tokens and, and all constants were replaced with a "C" token. In doing so, the model does not predict constant values, but just the location of constants, which are fit after using conventional algorithms. This is in contrast to end-to-end symbolic regression models [4], which attain state-of-the-art performance by predicting initial constant values before performing conventional fitting for improved performance. End-to-end models require significantly more compute resources to train, so due to limited available resources, our model was not trained to predict constant values. Instead we created a fair setting for comparison, and we compared our proposed method with the baselines without relying on incremental improvements such as beam-search, better constant optimizations by multi-candidate generation, and other minor improvements.

2.2 Backbone: neural network architecture

The objective of this work was to obtain a fair comparison between autoregressive generation (Symbolic GPT [5] baseline) and D3PM [9] discrete state-space diffusion (Symbolic Diffusion). To achieve this, a common neural network architecture was utilized, with equivalent encoders, transformer blocks, and decoder layers were used. The output layer dimensions and loss functions varied, of course, to accommodate the differences in the architectures used. Table 1 outlines the common model hyperparameters used between both the SymbolicGPT and Symbolic Diffusion models.

Table 1: Neural Network Hyperparameters

Parameter	Value
Embedding Dimension	512
Number of Heads	8
Number of Layers	8
Feedforward Dimension	2048
Dropout	0.15

2.2.1 Encoder

A common T-Net encoder was used, based on the implementation in SymbolicGPT [5], which was based on the T-Net developed by Qi et al. [14] for PointNet. This encoder features 3x blocks of Conv1D, BatchNorm, and ReLU, where the dimension is increased from the input dimension to E, $2E$, and $4E$, respectively, where E is the embedding dimension. Following this, there is a maxpool layer, making the encoder invariant to the number of points in the input feature, followed by 2 linear layers which decrease the dimension from $4E$ to $2E$ and finally the encoder output, of size embedding dimension E.

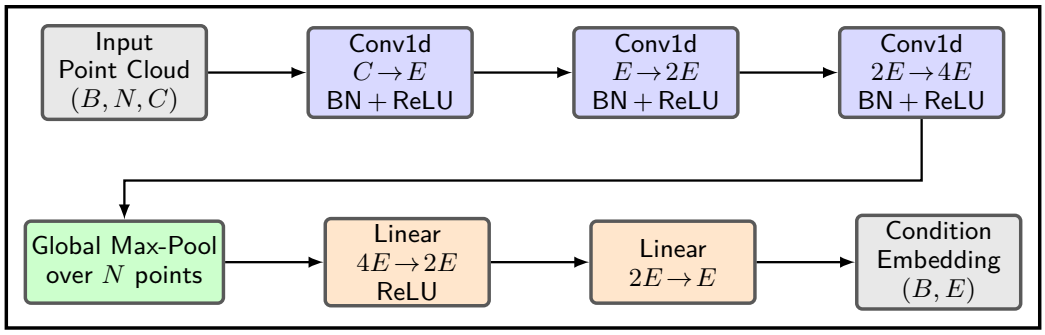

Figure 2: Architecture of the `T-Net Encoder`. B—batch size, N—number of points, C—input feature dimensionality, and E—embedding dimension.

2.2.2 Transformer blocks and decoder

As much as possible, identical transformer architectures were used between the Symbolic Diffusion model and the benchmark autoregressive Symbolic GPT model, this common architecture is shown below in Figure 3.

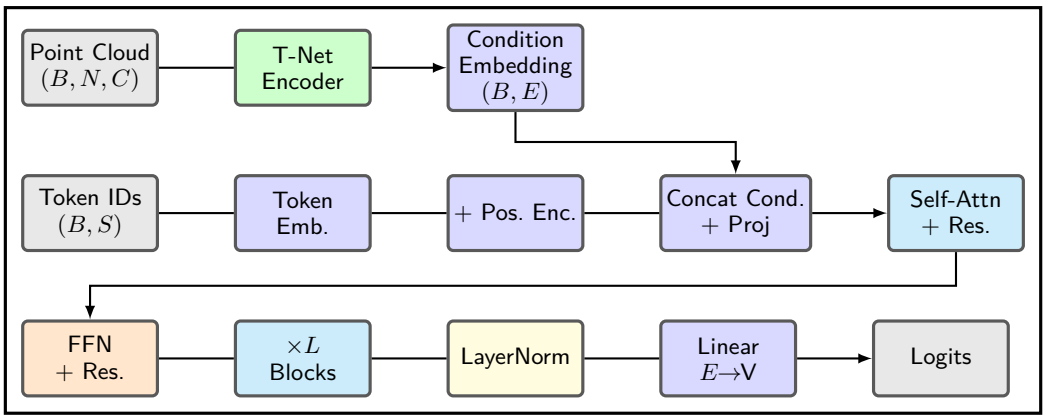

Figure 3: Common transformer architecture used by both Symbolic Diffusion and SymbolicGPT. B—batch size, N—number of points, C—input feature dimensionality, E—embedding dimension, and S—number of tokens.

It should be noted that there are some key differences between the architectures due to the different natures of GPT and D3PM diffusion. Notably, SymbolicGPT uses a causal mask to only attend to prior tokens, where Symbolic Diffusion does not. During inference, Symbolic Diffusion predicts logits for all tokens at timestep $t = 0$, whereas SymbolicGPT predicts logits for the next token only. Symbolic Diffusion transformer blocks have global context to all sequences always, whereas SymbolicGPT transformer blocks only have context to already generated tokens. Diffusion also utilizes a timestep encoding in addition to token position encoding to encode the noising step. Denoising for D3Pm is completed using the equations discussed below.

2.3 Proposed method: symbolic regression with discrete state-space diffusion model

The D3PM model used was based on the paper Structured Denoising Diffusion Models in Discrete State-Spaces by Austin et al. [9]. The model was conditioned on the embedded feature vector produced by the encoder. The hyperparameters from SymbolicGPT [5] were used as a baseline, and were tuned empirically. A batch size of 64, with a 1e-4 learning rate

and AdamW [21] gradient descent were used. Learning rate was reduced by a factor of 0.5 on plateau with a patience of 5 epochs, and early stopping was used with a patience of 15 epochs. For all experiments, a workstation equipped with an Intel i7 14700k, 128GB RAM, and 1x Nvidia RTX 3090 with 24GB of VRAM were used in this research.

2.3.1 Forward noising process

D3PM applies diffusion to discrete data, where each entry in x belongs to discrete categories $1, \ldots, K$. We use a Markovian transition matrix Q to define the probability of any token corrupting to another token with a given probability.

The forward noising process between timesteps can be represented by the equation, from Austin et al. [9]:

$$q(x_t \mid x_{t-1}) = \mathrm{Cat}\big(x_t;\, p = x_{t-1} Q_t\big) \tag{1}$$

Where x is an array of 1-hot vectors representing the current discrete state of each token, Cat is the categorical distribution, and the multiplication of $x_{(t-1)}$ and Q_t gives the probability of a token's transition.

The cumulative noising up to time step t is given by, from Austin et al. [9]:

$$q(x_t \mid x_0) = \mathrm{Cat}\big(x_t;\, p = x_0\, \overline{Q}_t\big), \quad \overline{Q}_t = Q_1 Q_2 \cdots Q_t \tag{2}$$

A transition matrix with a probability of $(1 - \beta_t)$ of remaining at its current token and a β probability of randomly being switched to another token was used. A cosine noising schedule over 1000 noising steps from $\beta = 0.0001$ to $\beta = 0.02$ was used.

2.3.2 Reverse noising process

The denoising process is represented through this equation, provided by Austin et al. [9],

$$p_\theta(x_{t-1} \mid x_t) \;\propto\; \sum_{\tilde{x}_0} q(x_{t-1}, x_t \mid \tilde{x}_0)\, \tilde{p}_\theta(\tilde{x}_0 \mid x_t) \tag{3}$$

is the output of the neural network, and we train the neural network to try and predict the probability logits of time step 0 given time step t. this represents the probability of transitioning from $x_{(t-1)}$ to x_t, which is known as the forward noising process is defined. The summation term sums over all possible discrete states for each token being generated, i.e., increasing token length doesn't increase complexity exponentially. With the proportionality term, we can attain a denoising step from the network's predictions of the tokens at x_0.

An additional important equation provided by Austin et al. [9] is the posterior, computed through basic application of Baye's rule,

$$q(x_{t-1} \mid x_t, x_0) = \frac{q(x_t \mid x_{t-1}, x_0)\, q(x_{t-1} \mid x_0)}{q(x_t \mid x_0)} = \mathrm{Cat}\Big(x_{t-1};\, p = \frac{x_t\, Q_t^\top \, x_0\, \overline{Q}_{t-1}}{x_0\, \overline{Q}_t\, x_t}\Big) \tag{4}$$

which is used to compute the likelihood of attaining a state at a previous time step given a future time step and knowing the original state. This posterior is needed for the KL divergence contained in the loss function [9],

$$D_{\mathrm{KL}}\big[q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t)\big] \tag{5}$$

whereas traditional diffusion uses the variational lower bound as the loss function, a modified loss function was used by the Austin et al. [9],

$$L_\lambda \;=\; L_{\mathrm{vb}} \;+\; \lambda\, \mathbb{E}_{q(x_0)}\Big[\, \mathbb{E}_{q(x_t \mid x_0)}\big[-\log \tilde{p}_\theta(x_0 \mid x_t)\big]\Big].$$

Where L_{vb} is the standard ELBO variational lower bound used in diffusion using the KL divergence from Sohl-Dickstein et al. [7], and the second term represents the cross-entropy for the prediction of x_0, which helps improve stability, as the neural network is predicting x_0 but it's prediction is being used for the computation of $x_(t-1)$, so adding cross-entropy with x_0 improves the stability of the network's generation.

2.4 Inference and constant fitting

Model inference was done by utilizing the input coordinate points as conditioning vectors to produce tokenized reverse polish notation outputs using both SymbolicGPT and Symbolic Diffusion.

As the model did not predict constants directly, constant fitting was completed using L-BFGS [22] implemented from the SciPy library [23]. 100 iterations of differential evolution [24] were done to pre-initialize the constants prior to completing L-BFGS fitting, as this was found to produce better outcomes from L-BFGS. This has been done to reduce the error of the L-BFGS which should not affect the comparison of the original models.

3 Results and discussion

3.1 Evaluation metrics

Two commonly used evaluation metrics for symbolic regression were used, inspired by Kamienny et al. [4]. R^2 represents how well the predicted model matches the input data [3], with 1 being a perfect fit. R^2 is unbounded less than 0, so like Kamienny et al. [4], the very small number of samples produced with R^2 less than 0 were set to zero to prevent excessively bad scores from skewing the mean. Accuracy to tolerance Acc_τ represents the fraction of equations which the relative error is within a tolerance τ [25, 26].

$$R^2 = 1 - \frac{\sum_i^{N_{\text{test}}}(y_i - \hat{y}_i)^2}{\sum_i^{N_{\text{test}}}(y_i - \bar{y})^2}, \qquad \text{Acc}_\tau = \mathbb{1}\left(\max_{1\leq i\leq N_{\text{test}}} \left|\frac{\hat{y}_i - y_i}{y_i}\right| \leq \tau\right) \tag{6}$$

Additionally, the percentage of valid equations was computed by ensuring the Reverse Polish Notation (RPN) equations were of proper form, checking that no binary operators were attempted on stack sizes of 1, and that the final stack size after all operators is equal to 1.

3.2 Results

Both models were evaluated on the same 7000 samples from the validation dataset, presented below in Table 2.

Table 2: Comparison of Symbolic Diffusion and SymbolicGPT on various metrics

Metric	Symbolic Diffusion	SymbolicGPT
Mean R^2 Score	**0.899**	0.887
Mean $\text{Acc}_{0.1}$	0.655	**0.679**
Mean $\text{Acc}_{0.01}$	0.531	**0.583**
Mean $\text{Acc}_{0.001}$	0.425	**0.516**
% Valid RPNs	0.984	**0.999**

Assuming approximately normal distributions, we can use a paired T-Test to compare outputs. With the results from the T-Test, we attain that with a 95% confidence interval, Symbolic Diffusion has a higher mean R^2 value (p = 0.001) however, using the same test, SymbolicGPT has a higher (better) mean accuracy than Symbolic Diffusion for all τ values. It should be noted that the results between both models are close, and both models show comparable performance overall.

3.3 Discussion

While Symbolic Diffusion did not outperform the autoregressive SymbolicGPT baseline on all evaluation metrics, it provided a successful demonstration of a novel method of symbolic regression using discrete token diffusion. Importantly, this demonstrates that for symbolic regression, diffusion based simultaneous generation where the generation process has global context to all tokens throughout the generation process can offer reasonable performance competitive with existing autoregressive models.

3.3.1 Societal Impacts

Improving symbolic regression offers positive societal impacts in the form of improving scientific discovery and improving decision making in society by finding trends in complex data. Risks exist that these models may capture unwanted biases present in training data, and may risk displacing jobs in data science roles.

3.4 Limitations

There are several important limitations with our work which should be noted. The intent of this work was not to develop a state-of-the-art model, but rather to compare diffusion generation versus autoregressive generation for the domain of symbolic regression. Compared to state-of-the-art models (SOTA) such as End-To-End Symbolic Regression [4], our model does not predict constants. Our training dataset of $\sim$ 500,000 was significantly smaller than SOTA models (SymFormer, 130 million [6], E2E SR $\sim$150 million [4]). Furthermore, our model was limited to bivariate data, and was not tested on datasets of varying dimensionalities. Due to these limitations, our model was unable to be benchmarked against the SOTA conventional and ML based models using tools such as SRBench [3]. Despite these limitations, our research provides the first fair, apples-to-apples comparison between a diffusion based and an autoregressive based symbolic regression model utilizing nearly identical underlying architectures, providing valuable insights into the performance of previously unexplored diffusion models in the domain of symbolic regression.

We also acknowledge the limited compute that it was available in this research paper and as such we focused on a fair comparison, and the novelty to solve a fundamental issue in the current state-of-the-art transformer models for Symbolic Regression.

3.5 Future work

The results demonstrate that diffusion based models are viable and can offer comparable performance to autoregressive models in the domain of symbolic regression. For future work, expanding the dataset size for training and increasing the dimensionality of the training set will be required to evaluate the models performance on more complex data. Adding end-to-end training with constant prediction will require more compute resources and larger datasets, but may help diffusion based symbolic regression match or exceed state of the art models. Additionally, investigating alternative diffusion models such as masked diffusion [27] may offer improved performance.

4 Conclusion

Our work presents a novel application of discrete-state diffusion modeling to the problem of symbolic regression. By leveraging a D3PM [9] approach to generate all tokens of an expression in parallel, our model is able to capture global syntactic structures more effectively than left to right autoregressive generation. Through experiments on the bivariate dataset originally designed for SymbolicGPT [5], we have shown that our diffusion based framework achieves comparable expression accuracy when using equivalent encoder, transformer, and decoder architecture as a SymbolicGPT [5] based autoregressive model.

These results validate the efficacy of diffusion based generation in the symbolic regression domain, and while further work must still be done to match and surpass autoregressive models, it highlights its potential to mitigate error in tasks requiring syntactical accuracy.

Overall, our findings demonstrate that discrete diffusion offers a compelling alternative to autoregressive methods for neural network based symbolic regression. We anticipate that further refinement of diffusion models and the incorporation of more complex encodings and architectures will continue to enhance the performance and robustness, ultimately broadening the applicability of neural-network techniques.

References

[1] Yousef Radwan, Gabriel Kronberger, and Stephan Winkler. *A Comparison of Recent Algorithms for Symbolic Regression to Genetic Programming*. doi: 10.48550/arXiv.2406.03585.

[2] John R. Koza. Genetic programming as a means for programming computers by natural selection. 4(2):87–112. ISSN 1573-1375. doi: 10.1007/BF00175355. URL https://doi.org/10.1007/BF00175355.

[3] William La Cava, Patryk Orzechowski, Bogdan Burlacu, Fabrício Olivetti de França, Marco Virgolin, Ying Jin, Michael Kommenda, and Jason H. Moore. Contemporary symbolic regression methods and their relative performance. URL http://arxiv.org/abs/2107.14351.

[4] Pierre-Alexandre Kamienny, Stéphane d'Ascoli, Guillaume Lample, and François Charton. End-to-end symbolic regression with transformers. URL http://arxiv.org/abs/2204.10532.

[5] Mojtaba Valipour, Bowen You, Maysum Panju, and Ali Ghodsi. SymbolicGPT: A generative transformer model for symbolic regression. URL http://arxiv.org/abs/2106.14131.

[6] Martin Vastl, Jonáš Kulhánek, Jiří Kubalík, Erik Derner, and Robert Babuška. SymFormer: End-to-end symbolic regression using transformer-based architecture. URL http://arxiv.org/abs/2205.15764.

[7] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. URL http://arxiv.org/abs/1503.03585.

[8] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. URL http://arxiv.org/abs/2112.10752.

[9] Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. URL http://arxiv.org/abs/2107.03006.

[10] Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. URL http://arxiv.org/abs/2310.16834.

[11] Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. DiffuSeq: Sequence to sequence text generation with diffusion models. URL http://arxiv.org/abs/2210.08933.

[12] Matthias Plasser, Silvan Peter, and Gerhard Widmer. Discrete diffusion probabilistic models for symbolic music generation. pages 5842–5850. doi: 10.24963/ijcai.2023/648.

[13] Zehui Li, Yuhao Ni, William A. V. Beardall, Guoxuan Xia, Akashaditya Das, Guy-Bart Stan, and Yiren Zhao. DiscDiff: Latent diffusion model for DNA sequence generation. URL http://arxiv.org/abs/2402.06079.

[14] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep learning on point sets for 3d classification and segmentation. URL http://arxiv.org/abs/1612.00593.

[15] https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf. URL https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.

[16] Bogdan Burlacu, Kaifeng Yang, and Michael Affenzeller. Population diversity and inheritance in genetic programming for symbolic regression. 23(3):531–566. ISSN 1572-9796. doi: 10.1007/s11047-022-09934-x. URL https://doi.org/10.1007/s11047-022-09934-x.

[17] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training.

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. URL http://arxiv.org/abs/1706.03762.

[19] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. URL http://arxiv.org/abs/2006.11239.

[20] Edward Finkelstein. Generalized fixed-depth prefix and postfix symbolic regression grammars. URL http://arxiv.org/abs/2410.08137.

[21] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. URL http://arxiv.org/abs/1711.05101.

[22] Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. 45(1):503–528. ISSN 1436-4646. doi: 10.1007/BF01589116. URL https://doi.org/10.1007/BF01589116.

[23] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental algorithms for scientific computing in python. 17:261–272. doi: 10.1038/s41592-019-0686-2.

[24] Rainer Storn and Kenneth Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. 11(4):341–359. ISSN 1573-2916. doi: 10.1023/A:1008202821328. URL https://doi.org/10.1023/A:1008202821328.

[25] Luca Biggio, Tommaso Bendinelli, Alexander Neitz, Aurelien Lucchi, and Giambattista Parascandolo. Neural symbolic regression that scales. URL http://arxiv.org/abs/2106.06427.

[26] Stéphane d'Ascoli, Pierre-Alexandre Kamienny, Guillaume Lample, and François Charton. Deep symbolic regression for recurrent sequences. URL http://arxiv.org/abs/2201.04600.

[27] Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K Titsias. Simplified and generalized masked diffusion for discrete data.