Symbolic expression generation *via* variational auto-encoder

Sergei Popov[1,2], Mikhail Lazarev[1], Vladislav Belavin[1], Denis Derkach[1] and Andrey Ustyuzhanin[1,3,4]

[1] Department of Computer Science, Higher School of Economics, Moscow, Russia
[2] National University of Science and Technology MISIS, Moscow, Russia
[3] Constructor University, Bremen, Germany
[4] Institute for Functional Intelligent Materials, National University of Singapore, Singapore

ABSTRACT

There are many problems in physics, biology, and other natural sciences in which symbolic regression can provide valuable insights and discover new laws of nature. Widespread deep neural networks do not provide interpretable solutions. Meanwhile, symbolic expressions give us a clear relation between observations and the target variable. However, at the moment, there is no dominant solution for the symbolic regression task, and we aim to reduce this gap with our algorithm. In this work, we propose a novel deep learning framework for symbolic expression generation *via* variational autoencoder (VAE). We suggest using a VAE to generate mathematical expressions, and our training strategy forces generated formulas to fit a given dataset. Our framework allows encoding apriori knowledge of the formulas into fast-check predicates that speed up the optimization process. We compare our method to modern symbolic regression benchmarks and show that our method outperforms the competitors under noisy conditions. The recovery rate of SEGVAE is 65% on the Ngyuen dataset with a noise level of 10%, which is better than the previously reported SOTA by 20%. We demonstrate that this value depends on the dataset and can be even higher.

Subjects Artificial Intelligence, Data Mining and Machine Learning, Data Science, Neural Networks
Keywords Symbolic regression, VAE, LSTM, Constrained optimization, Generation, Machine learning

Submitted 25 October 2022
Accepted 17 January 2023
Published 7 March 2023

Corresponding author
Sergei Popov, sapopov@hse.ru

Academic editor
Muhammad Aleem

Additional Information and Declarations can be found on page 13

DOI **10.7717/peerj-cs.1241**

INTRODUCTION

The discovery of new laws from experimental observations may seem to be an old and long-forgotten topic at first sight. Indeed, in the age of data-driven science, neural networks can easily fit a pretty complicated dependency. However, generalization and interpretation of those networks usually leaves much to be desired. For such phenomena as a biotech reaction, molecular dynamics potentials, or epidemic spread, a learning algorithm representing the dependency between target property and dependent characteristic in a simplistic and human-conceivable, such as usual formulas, is invaluable. There are many tasks where the black-box algorithms cannot be trusted with your eyes closed, *i.e.*, in self-driving cars, medicine, markets analyses, aircraft design and so on. In other words, areas where the cost of error is very high, so it is necessary to understand why algorithm made one decision or another. Such examples are everywhere. Moreover, the

How to cite this article Popov S, Lazarev M, Belavin V, Derkach D, Ustyuzhanin A. 2023. Symbolic expression generation *via* variational auto-encoder. **PeerJ Comput. Sci. 9:e1241 DOI 10.7717/peerj-cs.1241**

Symbolic Regressions techniques are suitable for solving an optimal control task (*Sahoo, Lampert & Martius, 2018b*). Optimal control task deals with the problem of finding a control law for a given system such that a certain criterion is achieved, solution to this problem is a function of time (*Diveev, Konstantinov & Danilova, 2021*).

Several methods enhancing the so-called symbolic regression approach have been developed recently (*Udrescu & Tegmark, 2020*), where the goal was to reconstruct 100 formulas from Feynman's lectures on physics. In a recent article, the symbolic regression was applied to find thermodynamic description of ionic transport out of experimental data (*Flores et al., 2022*). The resulting formula reflects physical principles of underpinning ionic transport. Symbolic regression is akin to a simple regression, as it fits experimental data. However, symbolic regression tries to find suitable and functional feature transformations represented by a computational graph over the original feature vector. Such graphs impose additional complications while optimizing those using gradient descent-based approaches.

Another difficulty such methods regularly face is the inherent noisiness of experimental measurements. Hence, each algorithm should ideally provide theoretical or empirical guarantees of the noise level robustness. The example of noisy data can be found in all kind of experimental data and sometimes the noise level could be quite high depending on experiment type (*Eling, Morgan & Marioni, 2019*; *Reinbold et al., 2021*). Since the scientific theories or models has to be proven or validated by experiment the ability to reconstruct the symbolic solution of observe effect is critically important. We address this issue by the algorithm presented in this work.

One of our key contributions is the design of a novel process of the symbolic regression training SEGVAE that differs from the current state-of-the-art approaches: higher noise stability, higher data efficiency, and adjustability of the priors for the symbolic expression to the physics intuition of the user. Often scientists do know the frame of the searching formula (*e.g.*, limits and approximations) and laws that data have to follow. We introduce a predicate mechanism for formula search and the ability to implement known conversation laws. These features are essential for processing experimental data.

The structure of this article is the following: "Literature Review" contains an overview of prior symbolic methods, and "Methods" contains a description of our approach. All experiments with comparing performance are presented in "Results". "Conclusions" concludes the article.

LITERATURE REVIEW

The goal of using artificial intelligence to help discover the scientific laws underlying experimental data has been pursued in several works. Some of these works assume prior knowledge of the mystery environments of interest. However, the ones most relevant to our study are the ones that minimize any assumptions.

Since the symbolic regression problem is a discrete optimization task with the search space that exponentially depends on expression length, the majority of traditional approaches generally exploit genetic algorithms *Searson, Leahy & Willis (2010)* and *Koza (1994)*. In this algorithms the process of optimization is inspired by the natural selection and relies on operators such as mutation, crossover and selection (*Michalewicz & Schoenauer, 1996*). The new populations are produced by iterative applying of genetic aforementioned operators on individuals from current population. The most successful one of these approaches is the commercial software Eureqa (*Schmidt & Lipson, 2009*), which was developed more than 10 years ago and still holds one of the leading positions in the field.

There are several recent works dedicated to recovering physical laws in symbolic form. *Udrescu & Tegmark (2020)* introduce an AI Feynman algorithm and further improved in *Udrescu et al. (2020)*. AI Feynman 2.0 uses (a) physics-inspired deep learning strategies, (b) dimensional analysis like search for symmetries, separability, and alike, (c) brute forces the simplified equation that recovers physical equations from experimental data. While this algorithm does a good job simplifying expressions, it struggles to recover expressions that could not be simplified enough.

PySR (*Cranmer et al., 2020*) is basically reincarnation of Eureqa used friendly interface which allows to introduce predicates. PySR built on Julia, uses regularized evolution, simulated annealing, and gradient-free optimization and interfaced by Python.

An interesting and intuitively easy approach was demonstrated in *Martius & Lampert (2016)*. Later this method was updated in *Sahoo, Lampert & Martius (2018a)* and in it's latest version (*Werner et al., 2021*). The authors proposed architecture similar to a multilayer perceptron (MLP), where instead of a single activation for all outputs, they used a custom set of activation functions. The authors claim the efficiency of such an approach over MLP neural networks outside of the training set region. Thus good extrapolation capabilities have been demonstrated. Unfortunately, the authors did not present comparison results with other existing methods on common datasets.

Another deep learning approach to symbolic regression is introduced by *Petersen et al. (2021)*. The authors present a gradient-based approach for symbolic regression based on reinforcement learning, which they call deep symbolic regression (DSR). DSR consists of a recurrent neural network that outputs a distribution over mathematical formulas. This network is used to sample equations, which will be evaluated based on the given dataset. Then the evaluation result will be used to further improve the distribution over mathematical formulas making the better expressions more probable. The DSR method has recently been updated by introducing a genetic programming component, significantly enhancing several benchmark tests. The latest algorithm version performs better than others to the best of our knowledge. Thus, we are using it (*Mundhenk et al., 2021*) as a baseline known as DSO. DSO uses some prebuilt predicates to avoid nested repeating functions *e.g.*, $sin(sin(sin(..)))$, but a more complex or physics-inspired predicates introduction is challenging.

A recent study was dedicated to mixed approach of computer vision methods and transformers to symbolic regression problem (*Li, Yuan & Shen, 2022*). The workflow is the following: the input data is represented by image, this image is treated by convolution layers to create the image embedding to feed the transformer. However, proposed approach might be useful only for a datasets, containing a large number of points. In addition, the results of proposed algorithm was not compared with published SOTA aproaches.

SciNet (*Iten et al., 2020*) approach is inspired by human thinking along a physical modeling process. Just like human physicists do not rely on actual observations but rather on their compressed representation to make some theoretical conclusions, SciNet encodes the experimental data to a latent representation that stores different physical parameters and uses this representation to answer specific questions about the underlying physical system. Undoubtedly SciNet is successful at learning relevant physical concepts. However, its goals are very different from ours: it does not recover the laws in symbolic form but uses a neural network to model them.

In an article on neural-symbolic regression that scales (*Biggio et al., 2021*) authors propose to use pre-trained transformers (*Vaswani et al., 2017*) to predict symbolic expression. The network consists of a transformer encoder and transformer decoder trained on generated formulas and BFGS algorithm for constants optimization. They compare transformer results to DSR (previous version of DSO) as a baseline. Despite good evaluation time, results quality is lower than DSR on the Nguyen dataset.

Our method is akin to the work by *Bowman et al. (2016)*. It adapts the variational autoencoder by using LSTM RNNs for both encoder and decoder. Thus, forming a sequence autoencoder with the Gaussian prior acting as a regularizer on the hidden code. The proposed generative model incorporates distributed latent representations of entire sentences. By examining paths through this latent space, it is possible to generate coherent novel sentences that interpolate between known sentences.

METHODS

This section introduces the symbolic expression generation *via* variational auto-encoder (SEGVAE) algorithm (*Sergei et al., 2022*). In a nutshell, our architecture is a variational auto-encoder (*Kingma & Welling, 2013*) in which the encoder and decoder are based on recurrent neural networks. The primary motivation behind using VAE for symbolic sequence representation is that VAE conveys a regularized learning method that minimizes the volume of low-energy (noise) representation space, thus preserving the richness of the signal (relevant sequences) representation. We also describe here our implementation of this architecture and the training procedures.

Architecture

SEGVAE sequentially generates formulas. It is possible due to the one-to-one correspondence between sequences of tokens and formulas. Each token can be one of three types: input variables, constants, and operators. In this article our typical library of operators is [‘*add*’, ‘*sub*’, ‘*mul*’, ‘*div*’, ‘*sin*’, ‘*cos*’, ‘*log*’, ‘*exp*’]. The number of variables X and constants depends on a task. We discuss the way to deal with constants in the subsection below. We use normal Polish notation to represent formulas as sequences in which operators precede their operands. The main advantage of Polish notation over the conventional one is that Polish representation is unambiguous and does not require brackets.

Variational autoencoder

VAE is a generative encoder-decoder based latent variable model. Given an observation space X with a distribution $p(x)$ the model's encoder maps it into a latent space Z with a distribution $p(z)$, and the model's decoder maps Z back into the observation space X. Let m-dimensionality of the latent space, μ_i, σ_i^2-the i^{th} components of vectors $\mu(x), \sigma^2(x)$. If $p(z) = N(0,1)$ and $q_{\theta_E}(z|x) = N(\mu(x), \sigma^2(x))$, then the objective takes the following form:

$$-KL(q_{\theta_E}(z|x)||p(z)) + \mathbb{E}_{q_{\theta_E}(z|x)} \log p_{\theta_D}(x|z) \tag{1}$$

This approach allows the model to decode plausible equations from every point in the latent space that has a reasonable probability under the prior. As long as we can represent expressions as sequences of tokens, we rely on traditional NLP approaches. Our encoder and decoder are both one-layer LSTMs with 64 hidden units.

Training procedure

Here we summarize the details of our training protocol, which consists of two steps pre-training and the main training cycle.

Pre-training

This step allows the model to memorize the general formula structure and generate valid formulas afterward. Firstly, we randomly sample sequences of tokens from given library L. However, uniformly sampling expressions with n internal nodes is not a simple task. Naive algorithms tend to favor specific kinds of expressions. Therefore, we follow the data generator technique introduced in *Lample & Charton (2020)*. Secondly, we choose only statements that meet our predicate conditions and thus create a pre-train dataset. Also, we add to the pre-train dataset a set of generalized formulas that commonly appear in natural science. Then the variational autoencoder is trained on these formulas. As a result of training, the vast majority of generated formulas in the next steps of the algorithm are valid formulas, so for the sake of simplicity, we can safely ignore invalid formulas in the following stages. Note that this step does not depend on the target task at all, so we do the pre-training once for each library L. The schematic picture of pre-train process and algorythm arhitecture presented in Fig. 1A. Figure 1B shows the formation of the Bank of Best Formulas.

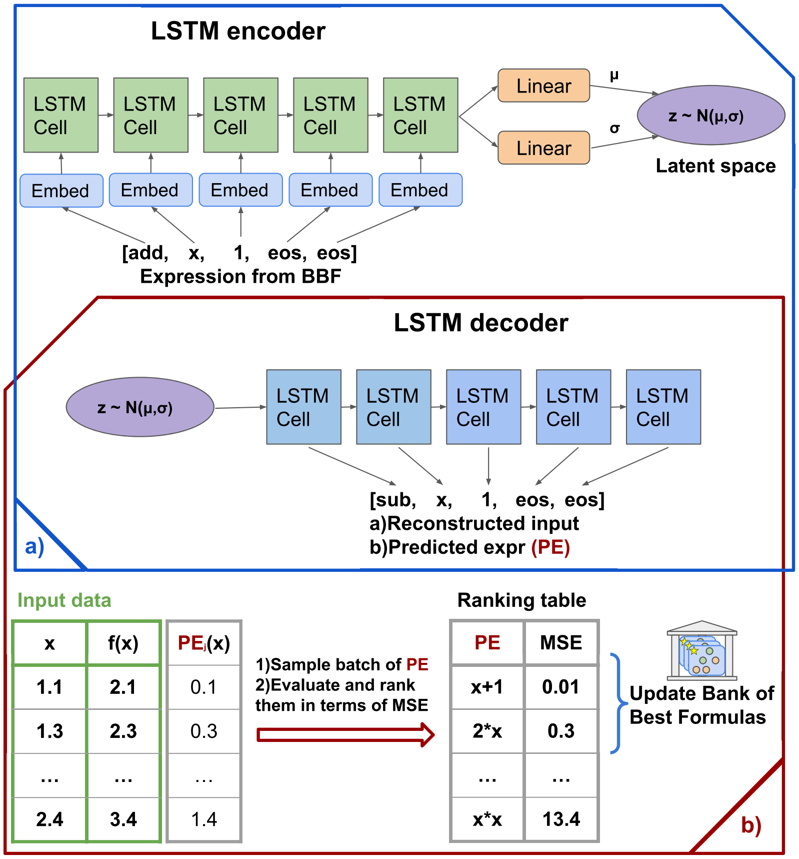
Main training cycle

When the model can generate valid formulas, it is time to teach it to generate valid formulas which describe a given environment or system under exploration. Firstly, a batch of formulas is sampled using the variational autoencoder for each epoch. Then, all the duplicates and invalid formulas are removed. Secondly, each formula f is evaluated on the dataset $D = (X_d, y_d)$ in terms of mean squared error between $f(X_d)$ and Y_d:

$$error(f) = \frac{1}{|D|} \sum_{x,y \in D} (f(x) - y)^2 \tag{2}$$

Algorithm 1 **SEGVAE: overall algorithm**

```
Input: data $(X, y)$, library of operators $L$, $N_{epochs} > 0$
$F_{pre-train}$ = GeneratePretrainFormulas($L$)
$F_{pre-train}$ = [$f$ if $predicate(f)$ for $f$ in $F_{pre-train}$]
$vae$.train($F_{pre-train}$)
for $i = 1$ to $N_{epochs}$ do
    $F_{train}$ = $vae$.sample($batch\_size$);
    $F_{train}$ = [$f$ if $predicate(f)$ for $f$ in $F_{train}$]
    $\overline{mse} = \frac{1}{|X|}\sum_j (F_{train}(X_j) - y)^2$
    if $F_{train}(x) = NaN$ or out of $Y$ discard $F_{train}(x)$
    if simplify $F_{train} = simplify(F_{train})$;
    $bbf = bbf$.update($F_{train}$, $\overline{mse}$)
    $vae$.train($bbf$)
end for
Result: $bbf.pareto\_front$
```



x	f(x)	PE$_i$(x)
1.1	2.1	0.1
1.3	2.3	0.3
…	…	…
2.4	3.4	1.4

PE	MSE
x+1	0.01
2*x	0.3
…	…
x*x	13.4

Figure 1 **The SEGVAE architecture and training scheme.** (A) Pretraining and training scheme. (B) Sampling scheme, where output formula evaluates and goes to the Bank of Formulas.

Full-size DOI: 10.7717/peerj-cs.1241/fig-1

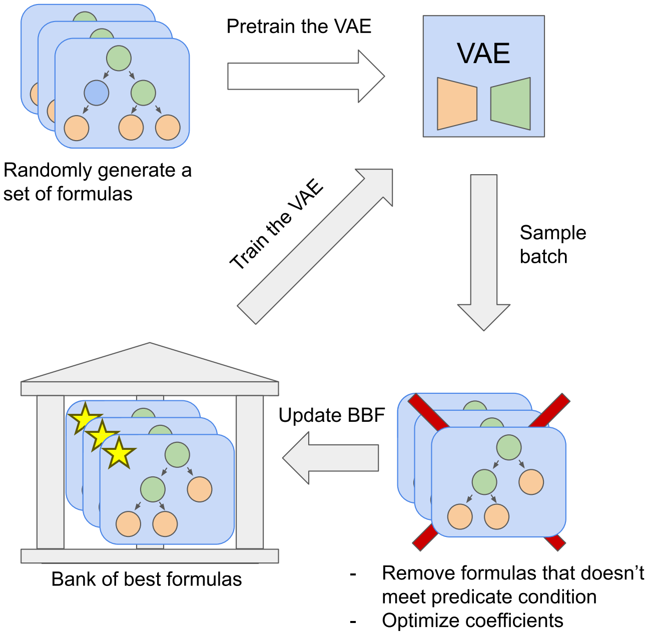


Figure 2 The SEGVAE algorithm training scheme. Pre-training stage and main training cycle.
Full-size DOI: 10.7717/peerj-cs.1241/fig-2

Then the P percent of the formulas with the smallest mean squared error are candidates to be saved to the bank of the best formulas (BBF). First, those candidates needs to be checked on correct of definition and values, by default $x \in (X_{min}, X_{max})$ and $y \in (Y_{min}, Y_{max})$ defined by the dataset. We evaluate function $f(x)$ on points sampled from a uniform distribution on (X_{min}, X_{max}) and compute $\hat{y} = f(x)$ for those points. If $\hat{y}$ gets out of the (Y_{min}, Y_{max}) region or we get a *NaN*, we discard this formula. This approach does not guarantee that the function f is defined on a given domain, *e.g.*, it cannot find discontinuities. However, as we show in our experiments, this approach improves the performance of our model, and it is computationally efficient compared to analytical evaluation. Second, if needed, formulas can be simplified using the *sympy* library before saving it in BBF. This bank keeps formulas from the last N epochs. Our typical value for both hyperparameters P and N is 20 and 5. Finally, the VAE is fine-tuned on formulas from the BBF. Details of the network training are specified in the supplemental materials. Training overview presented in Fig. 2.

Constants

There are two ways of dealing with constants in the resulting formulae. The first method supposes that all constants are incorporated in a library L. In this case, constants are regular tokens, and we do not need any modifications to our algorithm. The main drawback of this approach is the lack of expressiveness. However, it significantly helps the algorithm avoid overfitting to noisy data.

The second method is generating placeholders for future constants by including token '*const*' to the library L. Then, after the sampling stage, each placeholder is replaced by a

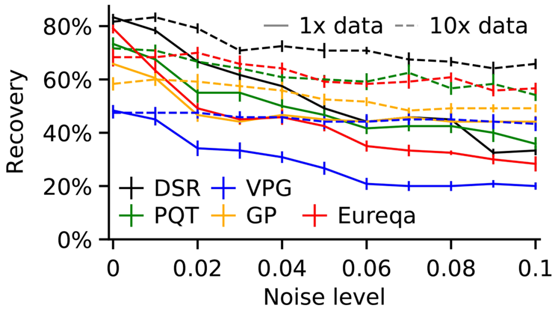


Figure 3 Recovery rate *vs* dataset noise and dataset size across all Nguyen benchmarks. Error bars represent standard error (adopted from *Mundhenk et al. (2021)*).
Full-size DOI: 10.7717/peerj-cs.1241/fig-3

parameter which we minimize the mean squared error between $f(X_d, consts)$ and Y_d by BFGS optimization algorithm (*Fletcher, 1987*).

Predicates

It is relatively straightforward to incorporate some prior knowledge about the target formula in the proposed algorithm framework. At the training stage, while we choose the best-sampled formulas by VAE, we ignore formulas that do not meet initial conditions regardless of their metrics. This technique reduces the search space, which helps SEGVAE find correct equations, as demonstrated in the experiment section below.

Imagine we know that the ratio of polynomials describes our physical system dynamics well, and the task is to determine that exact dependency. One way of helping the model is to reduce search space by shrinking the library of operators, *e.g.*, excluding trigonometrical operations. However, one can go further, creating a condition on possible positions for remaining tokens. In our case [‘*div*’] operator should be located only in the first place. Eventually model will learn not to generate formulas that violate prior conditions. We will demonstrate algorithm work results on chosen list of formulas and predicates in this article below. The list of formulas and related predicates are listed in Table 3.

Inference

Once the VAE model is trained, one can sample a batch of candidate expressions that are supposed to fit the given dataset. Overwhelmed formulas that describe the dataset best in terms of mean squared error could be overfitted to the noise in the data. Therefore, there is a trade-off between error and complexity. To choose the final expressions, we evaluate the complexity of each equation as:

$$C(t) = \sum_{i}^{T} c(\gamma_i) \tag{3}$$

where c is complexity of given token, which is equal to one for all input variables, constants and operators [‘*add*’, ‘*mul*’, ‘*sub*’], c is two for [‘*div*’], three for [‘*sin*’, ‘*cos*’] and finally four for [‘*log*’, ‘*exp*’]. Then we use this C and error values to identify Pareto-frontier and allow user to pick those formulas that satisfy her needs.

Table 1 **Nguyen dataset.** Variables are denoted as x_1 and x_2. Variables are uniformly sampled, U(a, b, c) denotes c times sampling between a and b for each input variable, N natural numbers, $L_0 = [add, sub, mul, div, exp, ln, sin, cos]$.

Name	Expression	Dataset	Library
Nguyen-1	$x_1^3 + x_1^2 + x_1$	$U(-1, 1, 20)$	$L_0 + [0.5, 1, -1, 2]$
Nguyen-2	$x_1^4 + x_1^3 + x_1^2 + x_1$	$U(-1, 1, 20)$	$L_0 + [0.5, 1, -1, 2]$
Nguyen-3	$x_1^5 + x_1^4 + x_1^3 + x_1^2 + x_1$	$U(-1, 1, 20)$	$L_0 + [0.5, 1, -1, 2]$
Nguyen-4	$x_1^6 + x_1^5 + x_1^4 + x_1^3 + x_1^2 + x_1$	$U(-1, 1, 20)$	$L_0 + [0.5, 1, -1, 2]$
Nguyen-5	$sin(x_1^2)cos(x_1) - 1$	$U(-1, 1, 20)$	$L_0 + [0.5, 1, -1, 2]$
Nguyen-6	$sin(x_1) + sin(x_1 + x_1^2)$	$U(-1, 1, 20)$	$L_0 + [0.5, 1, -1, 2]$
Nguyen-7	$log(x_1 + 1) + log(x_1^2 + 1)$	$U(0, 2, 20)$	$L_0 + [0.5, 1, -1, 2]$
Nguyen-8	$sqrt(x_1)$	$U(0, 4, 20)$	$L_0 + [0.5, 1, -1, 2]$
Nguyen-9	$sin(x_1) + sin(x_2^2)$	$U(0, 1, 20)$	$L_0 + [0.5, 1, -1, 2]$
Nguyen-10	$2\ sin(x_1)cos(x_2)$	$U(0, 1, 20)$	$L_0 + [0.5, 1, -1, 2]$
Nguyen-11	$x_1^{x_2}$	$U(0, 1, 20)$	$L_0 + [0.5, 1, -1, 2]$
Nguyen-12	$x_1^4 - x_1^3 + 0.5x_2^2 - x_2$	$U(0, 1, 20)$	$L_0 + [x^N, 0.5, N, e, pi]$

RESULTS

This section reports comparison results between our approach and state-of-the-art symbolic regression packages. We took the bests of our knowledge algorithms, namely deep symbolic optimization (DSO) (*Mundhenk et al., 2021*), since the DSO superior performance was supplemented by Fig. 3. We used the Nguyen 1–12 formulas (Table 1) and formulas listed in Table 3 to generate datasets. We compared the SEGVAE results with those of the DSO algorithm we took from the GitHub repository.

Datasets

Each symbolic regression task corresponds to a table of numbers, those rows are of the form $x_1, .., x_n, y$, where $y = f(x_1, .., x_n)$. The task is to discover the correct symbolic expression f.

- **Nguyen.** The Nguyen benchmark is commonly used as a symbolic regression benchmark. The Nguyen dataset consists of 12 formulas. We used the same dataset as in the *Petersen et al. (2021)* and its updates (see Table 1). To check the method's robustness, we add Gaussian noise proportional to y.
- **Livermore.** New benchmark dataset from the authors of *Mundhenk et al. (2021)* and *Petersen et al. (2021)*. It consists of 22 formulas. However, we did not use all of them due to their apparent similarity to the Nguyen dataset.

Ablation studies

As was described above, SEGVAE has many parameters such as the number of layers, sampled formulas in pretrained step, number of an epoch, and a maximum length of generated expression. To find the optimal parameters, we performed an ablation study. As a baseline to tune SEGVAE parameters, we used a subset of the Ngyuen dataset, namely

Table 2 **Mean recovery rate dependence from latent space dimension with fixed hidden dimensions on sub-Nguyen dataset.**

Latent space dimension	8	16	32	64	128	256
Mean recovery rate	50	58	60	65	75	75

Table 3 **List of formulas and used predicates.** Variables are denoted as x and y. Variables are uniformly sampled, U(a, b, c) denotes c times sampling between a and b for each input variable, N natural numbers, $L_0 = [add, sub, mul, div, exp, ln, sin, cos]$.

Formula name	Formula	Predicate	Dataset	Library
Nguyen-12	$x_1^4 - x_1^3 + 12x_2^2 - x_2$	$f(x_1) + g(x_2)$	$U(0, 10, 200)$	$L_0 + [x^N, 0.5, N, e, pi]$
Neat-8	$\frac{exp(-(x_1-1)^2)}{1.2+(x_2-2.5)^2}$	$exp(f(x_1))/(g(x_2))$	$U(0.3, 4, 100)$	$L_0 + [0.5, 1, -1, 2]$
Neat-9	$\frac{1}{1+x_1^4} + \frac{1}{1+x_2^4}$	$1/f(x_1) + 1/g(x_2)$	$U(-5, 5, 21)$	$L_0 + [0.5, 1, -1, 2]$
Livermore-5	$x_1^4 - x_1^3 + x_1^2 - x_2$	$f(x_1) + g(x_2)$	$U(0, 1, 20)$	$L_0 + [0.5, 1, -1, 2]$
Livermore-7	$12e^{x_1} - 12e^{-x_1}$	$f(x_1) - 1/g(x_1)$	$U(-1, 1, 20)$	$L_0 + [0.5, 1, -1, 2]$
Livermore-8	$12e^{x_1} + 12e^{-x_1}$	$f(x_1) + 1/g(x_1)$	$U(-1, 1, 20)$	$L_0 + [0.5, 1, -1, 2]$
Livermore-10	$6sin(x_1)cos(x_2)$	$const * f(x_1) * g(x_2)$	$U(0, 1, 20)$	$L_0 + [x^N, 0.5, N, e, pi]$
Livermore-17	$4sin(x_1)cos(x_2)$	$const * f(x_1) * g(x_2)$	$U(0, 1, 20)$	$L_0 + [x^N, 0.5, N, e, pi]$
Livermore-22	$exp(-\frac{1}{2}x_1^2)$	$exp(f(x_1))$	$U(0, 1, 20)$	$L_0 + [0.5, 1, -1, 2]$
R-2*	$\frac{(x_1+1)^3}{x_1^2-x_1+1}$	$f(x_1)/g(x_1)$	$U(-10, 10, 20)$	$L_0 + [0.5, 1, -1, 2]$

Ngyuen-4,5,9,10. First, we concluded that a maximum expression length of 30 is perfectly balanced in terms of model expressibility and model stability. Secondly, we found that hidden dimensionality in the range of 64 to 256 does not affect the recovery rate of 50%. By *recovery* here, we mean exact symbolic equivalence of the suggested expression to the original formula. For example, we initialize the algorithm 100 times with different random seeds for a given dataset generated by some formula. Out of 100 runs, 80 correctly represented the initial formula. In this case, the recovery rate is 80%. Nevertheless, the recovery rate depends on latent space, as presented in Table 2. We found that latent configuration space of size 128 with 128 hidden units to be optimal for this study. Another critical observation is the SEGVAE's recovery rate dependence on the library selection. We checked the dependence of the recovery rate on the number of tokens in the library for the DSO and SEGVAE algorithms. Small library size may be why the algorithm cannot find a correct formula. It is simply because not enough tokens are available to describe a formula. On the other hand, an over-inflated library exponentially increases algorithm search space for limited search iteration numbers. Thus, choosing excessive library contents may be a reason for a miserable formula reconstruction. A scientist has some prior knowledge about unknown yet dependency in an actual research process. Thus, we can use both predicates and a task-related library to simulate an actual searching formula situation. In the last series of ablation experiments, we show that our proposed method of discarding formulas that do not satisfy a given domain improves algorithm convergence speed by 50% for the

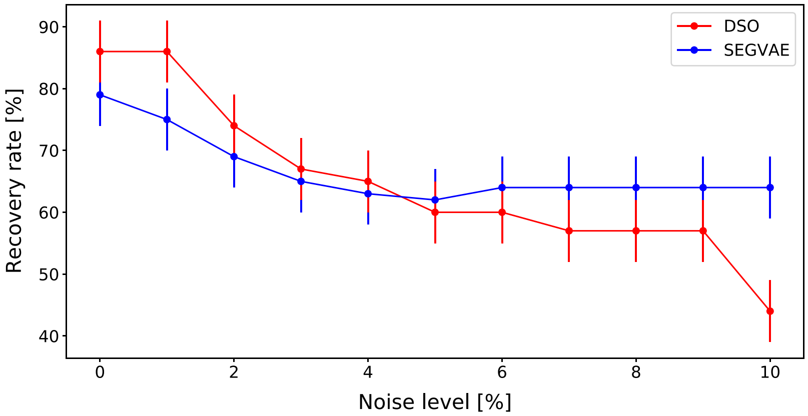


Figure 4 Average recovery rates of SEGVAE (blue) and DSO (red) algorithms on Ngyuen dataset with error bars. Full-size DOI: 10.7717/peerj-cs.1241/fig-4

Nguyen dataset. Moreover, our experiments show that it enhances the recovery rate of Livermore-5 and Livermore-7 equations by 100%.

Noise in data

Noisy data were created by adding Gaussian noise with zero mean and standard deviation proportional to the root-mean-square of the dependent variable *y*. To check the models' robustness, we present averaged recovery rates as a function of the noise from 0% (noiseless) to the maximum 10%. The main difficulty of regression with noise is the model's tendency to overfit the data. Sometimes increasing the number of points in a dataset may help.

Experiments

We have evaluated the SEGVAE algorithm on the most commonly used Nguyen benchmark, consisting of 12 formulas. We compared and evaluated our models on two variants of Nguyen datasets. For the first (*Dataset I*) variant, we sampled only 20 uniformly distributed points, as shown in Table 1. We used the same hyperparameters in SEGVAE for all runs (see details in the supplemental materials). The number of examined expressions was set to 2 million per run, the same as in the DSO article (*Mundhenk et al., 2021*).

A comparison of DSO and SEGVAE on the Dataset is presented in Fig. 4 as the dependence of averaged recovery rates over noise level. Red and blue colors denote the outputs of DSO and SEGVAE algorithms respectively. Essential to reiterate that by recovery, we mean at least one exact match with the wanted formula in the Pareto frontier.

SEGVAE and DSO show similar average recovery rates (ARR) with below moderate noise levels. We averaged the recovery rate overall Nguyen formulas at a given noise level to compute ARR. With increasing noise levels, DSO recovery rates slowly go down. On the other hand, SEGVAE demonstrates good recovery rates stability up to the noise level of 10% with a recovery rate of 70%. The difference becomes visible at high noise levels where SEGVAE slightly outperforms DSO, 70% *vs*. 45% at maximum noise level. The SEGVAE

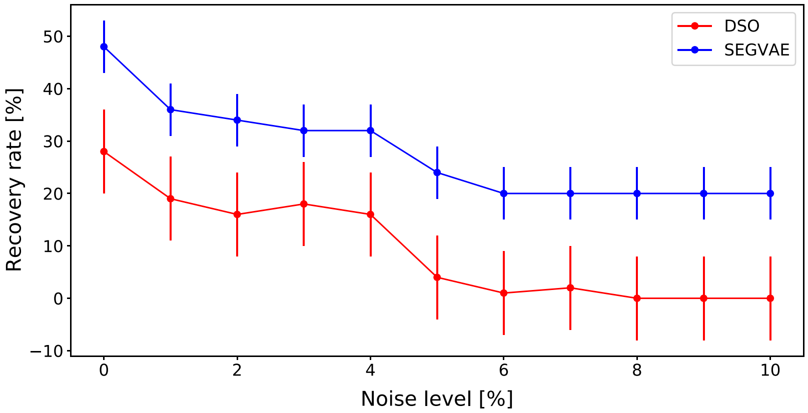


Figure 5 **Average recovery rates of SEGVAE (blue) and DSO (red) algorithms based on formulas listed in Table 3 with error bars.** Full-size DOI: 10.7717/peerj-cs.1241/fig-5

Table 4 **Mean recovery rate dependence from latent space dimension with fixed hidden dimensions on sub Nguyen dataset.**

	$N-12$	$Neat-8$	$Neat-9$	$L-5$	$L-7$
SEGVAE	100%	0%	0%	60%	20%
DSO	0%	0%	0%	80%	0%
	$L-8$	$L-10$	$L-17$	$L-22$	$R-2*$
SEGVAE	0%	100%	100%	100%	0%
DSO	0%	63%	57%	84%	4%

algorithm demonstrates higher noise stability on this dataset even without prior knowledge.

We compared algorithms and selected formulas from the original DSO article, where the DSO algorithm shows daunting results. These formulas are presented in Table 3. We used specific predicates and token libraries to reveal these formulas and, at the same time to demonstrate the power of our approach. The results of this comparison are summarised in the same way through the ARR in Fig. 5. The detailed results on noiseless data are presented in Table 4.

It often appears that scientists already know the general form of the object of interest. Thus, it would be natural to add predicates to let SEGVAE search formulas in a specific domain. We have seen that our VAE-based algorithm gives a similar result to the recent DSO results on the Nguyen dataset without any prior knowledge. However, there are many equations in which DSO is not accurate enough. To demonstrate the power of predicates in SEGVAE, we took predicates as listed in Table 3 and compared our results to DSO. The comparison results on noiseless data are presented in Table 4. The SEGVAE approach with predicates demonstrates a superior recovery rate than DSO, especially on noiseless data. More detailed results on noiseless are presented in Table 4. The proper predicates and optimal library play a crucial role in this case. In reality, scientists do not know the exact functional form of the studied effect. We can not use the recovery rate as a benchmark in this case. The only benchmark we can trust is MSE and formula shape or predicates that carry common scientific sense.

CONCLUSIONS

We introduced a novel algorithm, SEGVAE, for searching for symbolic representation of functional dependence from dependent variables. This approach is based on the VAE generative model that produces mathematical expressions and is constrained by apriori knowledge encoded in the form of fast-check predicates. Those predicates can express, for example, allowed formula patterns, domain, and possible output intervals.

As a benchmark, we used a set of formulas introduced in the DSO article, namely the Nguyen and the Livermore datasets. Our approach has the flexibility of formulating a priory physical knowledge in the form of (a) library of functions, (b) pre-training functions, and (c) selection predicates used for pruning incorrectly generated expressions. Besides the sole accuracy of the method, we focused on the algorithm's performance in realistic noisy environments. Symbolic regression approaches excel significantly compared to deep learning models where interpretability is paramount. Thus, the demand for such approaches is high in such scientific branches as material sciences, biotechnologies, and astrophysics, to mention a few.

We systematically compared SEGVAE with DSO and shown superior performance of our approach, thus outperforming Eureqa, Wolfram, and alike, which DSO has dominated before. For the scarce-data regime and high-noise regimes, the SEGVAE significantly outperforms the competitors. However, this approach has its limitations: (a) the output formulas quality depends on the dataset and its size, (b) obtaining adequate formula may require the user's assistance, namely in choosing predicates and the final formula from Pareto-front, and (c) running time may be noticeable, *i.e.*, around tens of minutes on a modern GPU.

We pointed out the importance of the library size and showed that SEGVAE discovered formulas unreachable for DSO thanks to the flexibility of predicates supported by our method. The recovery rate of SEGVAE improves the previously reported SOTA by 20%. Since experimental data usually contains noise and some prior knowledge on the functional dependency is typically available, the SEGVAE benefits can be easily seen from an application point of view. Therefore, our model may be useful in practical cases where interpretable symbolic solutions are needed to understand processes underlying experimental observations.

ACKNOWLEDGEMENTS

So long and thanks for all the fish.

ADDITIONAL INFORMATION AND DECLARATIONS

Funding

The publication was supported by the grant for research centers in the field of AI provided by the Analytical Center for the Government of the Russian Federation (ACRF) in accordance with the agreement on the provision of subsidies (identifier of the agreement 000000D730321P5Q0002) and the agreement with HSE University No. 70-2021-00139. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Grant Disclosures

The following grant information was disclosed by the authors:
Analytical Center for the Government of the Russian Federation (ACRF): 000000D730321P5Q0002.
HSE University: 70-2021-00139.

Competing Interests

The authors declare that they have no competing interests.

Author Contributions

- Sergei Popov conceived and designed the experiments, performed the experiments, analyzed the data, performed the computation work, prepared figures and/or tables, authored or reviewed drafts of the article, and approved the final draft.
- Mikhail Lazarev conceived and designed the experiments, performed the experiments, analyzed the data, performed the computation work, prepared figures and/or tables, authored or reviewed drafts of the article, and approved the final draft.
- Vladislav Belavin performed the computation work, authored or reviewed drafts of the article, and approved the final draft.
- Denis Derkach analyzed the data, authored or reviewed drafts of the article, and approved the final draft.
- Andrey Ustyuzhanin conceived and designed the experiments, authored or reviewed drafts of the article, and approved the final draft.

Data Availability

The following information was supplied regarding data availability:

The code and data are available at GitHub and Zenodo: https://github.com/HSE-LAMBDA/SEGVAE.

Popov Sergei, Lazarev Mikhail, Belavin Vladislav, Derkach Denis, & Ustyuzhanin Andrey. (2022). Symbolic expression generation *via* Variational Auto-Encoder. Zenodo. https://doi.org/10.5281/zenodo.7364439.

The code is available at the link: https://github.com/anonnipsuser/segvae.

REFERENCES

Biggio L, Bendinelli T, Neitz A, Lucchi A, Parascandolo G. 2021. Neural symbolic regression that scales. In: Meila M, Zhang T, eds. *Proceedings of the 38th International Conference on Machine Learning, Volume 139 of Proceedings of Machine Learning Research*. 936–945. *Available at https://proceedings.mlr.press/v139/biggio21a.html*.

Bowman SR, Vilnis L, Vinyals O, Dai A, Jozefowicz R, Bengio S. 2016. Generating sentences from a continuous space. In: *Proceedings of the 20th SIGNLL Conference on Computational*

Natural Language Learning. Berlin, Germany: Association for Computational Linguistics, 10–21.

Cranmer MD, Sanchez-Gonzalez A, Battaglia PW, Xu R, Cranmer K, Spergel DN, Ho S. 2020. Discovering symbolic models from deep learning with inductive biases. *CoRR* DOI 10.48550/arXiv.2006.11287.

Diveev A, Konstantinov S, Danilova A. 2021. Solution of the optimal control problem by symbolic regression method. *Procedia Computer Science* **186(2)**:646–653 14th International Symposium Intelligent Systems DOI 10.1016/j.procs.2021.04.212.

Eling N, Morgan MD, Marioni JC. 2019. Challenges in measuring and understanding biological noise. *Nature Reviews Genetics* **20(9)**:536–548 DOI 10.1038/s41576-019-0130-6.

Fletcher R. 1987. *Practical methods of optimization*. New York: John Wiley & Sons, Ltd.

Flores E, Wölke C, Yan P, Winter M, Vegge T, Cekic-Laskovic I, Bhowmik A. 2022. Learning the laws of lithium-ion transport in electrolytes using symbolic regression. *Digital Discovery* **1(4)**:440–447 DOI 10.1039/D2DD00027J.

Iten R, Metger T, Wilming H, del Rio L, Renner R. 2020. Discovering physical concepts with neural networks. *Physical Review Letters* **124(1)**:010508 DOI 10.1103/PhysRevLett.124.010508.

Kingma DP, Welling M. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* DOI 10.48550/arXiv.1312.6114.

Koza JR. 1994. Genetic programming as a means for programming computers by natural selection. *Statistics and Computing* **4(2)**:87–112 DOI 10.1007/BF00175355.

Lample G, Charton F. 2020. Deep learning for symbolic mathematics. In: *International Conference on Learning Representations* DOI 10.48550/arXiv.1912.01412.

Li J, Yuan Y, Shen H-B. 2022. Symbolic expression transformer: a computer vision approach for symbolic regression. *Available at https://arxiv.org/abs/2205.11798*.

Martius G, Lampert CH. 2016. Extrapolation and learning equations. *CoRR* DOI 10.48550/arXiv.1610.02995.

Michalewicz Z, Schoenauer M. 1996. Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation* **4(1)**:1–32 DOI 10.1162/evco.1996.4.1.1.

Mundhenk TN, Landajuela M, Glatt R, Santiago CP, Faissol DM, Petersen BK. 2021. Symbolic regression via neural-guided genetic programming population seeding. In: *35th Conference on Neural Information Processing Systems (NeurIPS 2021)* DOI 10.48550/arXiv.2111.00053.

Petersen BK, Larma ML, Mundhenk TN, Santiago CP, Kim SK, Kim JT. 2021. Deep symbolic regression: recovering mathematical expressions from data via risk-seeking policy gradients. In: *International Conference on Learning Representations*.

Reinbold PAK, Kageorge LM, Schatz MF, Grigoriev RO. 2021. Robust learning from noisy, incomplete, high-dimensional experimental data via physically constrained symbolic regression. *Nature Communications* **12(1)**:3219 DOI 10.1038/s41467-021-23479-0.

Sahoo S, Lampert C, Martius G. 2018a. Learning equations for extrapolation and control. In: Dy J, Krause A, eds. *Proceedings of the 35th International Conference on Machine Learning, Volume 80 of Proceedings of Machine Learning Research*. 4442–4450.

Sahoo SS, Lampert CH, Martius G. 2018b. Learning equations for extrapolation and control. In: *Proceedings 35th International Conference on Machine Learning, ICML 2018, Stockholm, Sweden, 2018*. **80**:4442–4450. *Available at https://proceedings.mlr.press/v80/sahoo18a.html*.

Schmidt M, Lipson H. 2009. Distilling free-form natural laws from experimental data. *Science* **324(5923)**:81–85 DOI 10.1126/science.1165893.

Searson DP, Leahy DE, Willis MJ. 2010. Gptips: an open source genetic programming toolbox for multigene symbolic regression. Dordrecht: Springer DOI 10.1007/978-94-007-0286-8_8.

Sergei P, Mikhail L, Vladislav B, Denis D, Andrey U. 2022. Symbolic expression generation via Variational Auto-Encoder. DOI 10.48550/arXiv.2301.06064.

Udrescu S-M, Tan A, Feng J, Neto O, Wu T, Tegmark M. 2020. AI Feynman 2.0: pareto-optimal symbolic regression exploiting graph modularity. In: *Advances in Neural Information Processing Systems 33 Pre-Proceedings (NeurIPS 2020).* Red Hook: Curran Associates, Inc., Vol. 33, 4860–4871.

Udrescu S-M, Tegmark M. 2020. AI Feynman: a physics-inspired method for symbolic regression. *Science Advances* **6(16)**:eaay2631 DOI 10.1126/sciadv.aay2631.

Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Lu, Polosukhin I. 2017. Attention is all you need. In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R, eds. *Advances in Neural Information Processing Systems.* Vol. 30. New York: Curran Associates, Inc.

Werner M, Junginger A, Hennig P, Martius G. 2021. Informed equation learning. *ArXiv preprint.* DOI 10.48550/arXiv.2105.06331.