# Efficient Generator of Mathematical Expressions for Symbolic Regression

Sebastian Mežnar[1,2], Sašo Džeroski[1,2], Ljupčo Todorovski[1,3]

[1]Jožef Stefan Institute, Department of Knowledge Technologies, Ljubljana, Slovenia
[2]International Postgraduate School Jožef Stefan, Ljubljana, Slovenia
[3]University of Ljubljana, Faculty of Mathematics and Physics, Ljubljana, Slovenia

Jožef Stefan Institute
Ljubljana, Slovenia

MEDNARODNA PODIPLOMSKA ŠOLA JOŽEFA STEFANA

Univerza *v Ljubljani*
Fakulteta za *matematiko in fiziko*

## Motivation

Mathematical equations are an essential tools for understanding relationships between variables in various scientific fields. Traditionally, scientists derive these equations with experimentation and domain knowledge. However, with recent advancements in machine learning and generative modeling, we can accelerate this process and find equations that fit the data better than hand-crafted ones.

## Methodology

We utilize variational autoencoders to embed mathematical expressions into a Euclidean vector space, incorporating the structural properties of binary expression trees. This approach not only ensures syntactic correctness but also enhances embedding efficiency. By encoding syntactically similar expressions close together in the vector space, we can efficiently explore the space of possible expressions with optimization algorithms.

Our method encodes expression trees recursively, starting at the leaf nodes and ending at the root node. We use a modified GRU cell to generate a code for each node based on the codes of its descendants and the symbol in the node. This encoding scheme improves embedding efficiency by assigning the same code to subexpressions, regardless of the rest of the expression.

Decoding is also performed recursively, starting at the root node and ending at the leaf nodes. During decoding, we produce a symbol for each node and recursively generate its descendants. For operators, we generate both descendants; for functions, only the left descendant; and for constants and variables, no descendants. The decoding cell also uses a modified GRU cell that produces two codes as the output, instead of one.

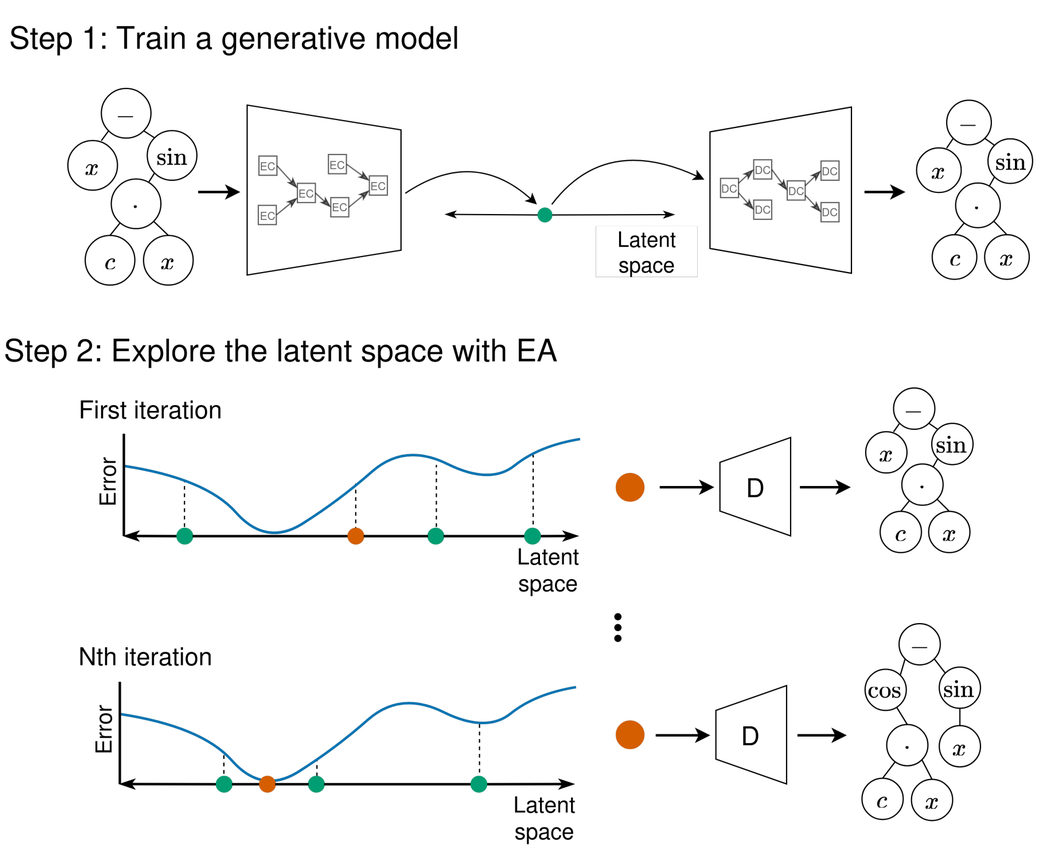


Figure 1: The EDHiE approach. In the first step, we train a HVAE model. In the second step, we explore the latent space of the HVAE model with an evolutionary algorithm. The red dot represents the best expression in a given iteration.

## Contact Information

- Web: https://smeznar.github.io/
- Email: sebastian.meznar@ijs.si, Phone: +386 51 322 108
- Repository: https://github.com/smeznar/HVAE

## Expression reconstruction

We show the generative power of the HVAE model by reconstructing expressions that were not part of the training set. We encode an expression into the latent space and decoding it back into an expression. We then compared the two expressions with edit distance. HVAE [3] significantly outperforms other approaches based on variational autoencoders, while guaranteeing syntactic correctness.

Table 1: The out-of-sample reconstruction error and the percentages of syntactically incorrect expressions generated by the three variational autoencoders.

| | HVAE | | GVAE | | CVAE | |
|---|---|---|---|---|---|---|
| Dataset | Edit distance | Invalid | Edit distance | Invalid | Edit distance | Invalid |
| AE4-2k | **0.076 (± 0.024)** | **0.0 (± 0.0)** | 3.959 (± 0.135) | 0.2 (± 0.0) | 3.873 (± 0.132) | 33.8 (± 1.1) |
| Trig4-2k | **0.119 (± 0.026)** | **0.0 (± 0.0)** | 3.199 (± 0.068) | **0.0 (± 0.0)** | 3.619 (± 0.045) | 48.3 (± 0.6) |
| AE5-15k | **0.079 (± 0.014)** | **0.0 (± 0.0)** | 2.827 (± 0.280) | < 0.1 (± 0.0) | 1.547 (± 0.466) | 3.5 (± 0.0) |
| Trig5-15k | **0.093 (± 0.010)** | **0.0 (± 0.0)** | 1.489 (± 0.195) | < 0.1 (± 0.0) | 2.086 (± 0.346) | 13.9 (± 0.0) |
| AE7-20k | **0.501 (± 0.017)** | **0.0 (± 0.0)** | 5.201 (± 0.289) | < 0.1 (± 0.0) | 3.654 (± 0.349) | 9.9 (± 0.0) |
| Trig7-20k | **0.530 (± 0.036)** | **0.0 (± 0.0)** | 3.423 (± 0.467) | < 0.1 (± 0.0) | 3.660 (± 0.287) | 26.3 (± 0.1) |

## Symbolic regression

Our approach, EDHiE, combines HVAE with an evolutionary algorithm. Table below shows the performance of our approach compared to the state-of-the-art approaches DSO [1] and PySR [2] on the Nguyen symbolic regression benchmark. We see that our model successfully reconstructs all equations and outperforms DSO and PySR.

Table 2: Comparison of the performance of the symbolic regression systems EDHiE, DSO, and PySR on the ten Nguyen equations.

| | EDHiE (our) | | | DSO [1] | | | PySR [2] | |
|---|---|---|---|---|---|---|---|---|
| Name | Successful | Mean $R^2$ | Evaluated | Successful | Mean $R^2$ | Evaluated | Successful | Mean $R^2$ |
| NG-1 | **10** | **1.00 (± 0.00)** | **573 (± 261)** | **10** | **1.00 (± 0.00)** | 4565 (± 327) | **10** | 1.00 (± 0.00) |
| NG-2 | **10** | **1.00 (± 0.00)** | **5803 (± 4148)** | **10** | **1.00 (± 0.00)** | 12206 (± 9186) | **10** | 1.00 (± 0.00) |
| NG-3 | 6 | 1.00 (± 0.01) | 20931 (± 4858) | **10** | **1.00 (± 0.00)** | **8053 (± 3766)** | 2 | 1.00 (± 0.01) |
| NG-4 | 3 | 1.00 (± 0.01) | **21346 (± 4479)** | **8** | **1.00 (± 0.01)** | 32946 (± 15613) | 0 | 0.99 (± 0.01) |
| NG-5 | **3** | **0.32 (± 0.45)** | **20615 (± 8394)** | 0 | 0.00 (± 0.00) | NA | 0 | 0.16 (± 0.15) |
| NG-6 | **8** | **0.88 (± 0.14)** | **12772 (± 7923)** | 1 | 0.59 (± 0.15) | 49599 (± 0) | 4 | 0.86 (± 0.13) |
| NG-7 | 8 | 1.00 (± 0.01) | **19203 (± 3595)** | **10** | **1.00 (± 0.00)** | 22579 (± 10264) | 7 | 0.99 (± 0.01) |
| NG-8 | **10** | **1.00 (± 0.00)** | **405 (± 174)** | **10** | **1.00 (± 0.00)** | 5521 (± 1779) | **10** | **1.00 (± 0.00)** |
| NG-9 | 8 | 0.95 (± 0.15) | **7041 (± 3933)** | 2 | 0.60 (± 0.20) | 39786 (± 28197) | **10** | **1.00 (± 0.00)** |
| NG-10 | **1** | 0.70 (± 0.17) | **31863 (± 6970)** | 0 | 0.56 (± 0.10) | NA | **1** | **0.80 (± 0.16)** |
| Total/Mean | **66** | **0.89 (± 0.21)** | | 61 | 0.78 (± 0.31) | | 54 | 0.88 (± 0.26) |

## Further work

In future work, we aim to improve our approach by encoding semantically similar expressions close together rather than relying solely on syntactic similarity. Additionally, we plan to explore more powerful models, such as generative adversarial networks or diffusion models, and extend our methodology to generate more complex models, such as molecules.

## References

[1] Brenden K Petersen, Mikel Landajuela, T Nathan Mundhenk, Claudio P Santiago, Soo K Kim, and Joanne T Kim.
Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients.
In *Proc. of the International Conference on Learning Representations*, 2021.

[2] Miles Cranmer.
Interpretable machine learning for science with pysr and symbolicregression.jl, 2023.

[3] Sebastian Mežnar, Sašo Džeroski, and Ljupčo Todorovski.
Efficient generator of mathematical expressions for symbolic regression.
2023.