Extending GFN-SR with a Transformer Policy

Bart den Boef [1] Julio Smidi [1] Bart Aaldering [1]

[1]University of Amsterdam

Universiteit van Amsterdam
Faculteit der Natuurwetenschappen, Wiskunde en Informatica

Research Question

For GFN-SR, can a Transformer policy function improve upon the current RNN policy in generating expression trees, thereby improving model performance?

Subquestions:

- How does a Transformer policy affect recovery speed?
- Does it behave differently on different equation types?

Background

Symbolic Regression (SR)

- Seeks to find an interpretable closed-form symbolic expression from input-output pairs.
- Interpretable.
- More flexible than traditional fixed functional forms regression.

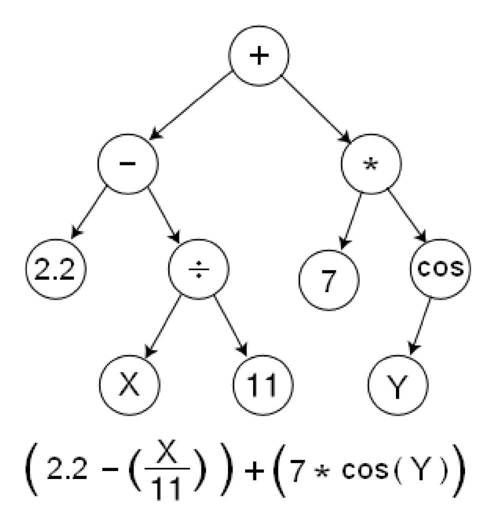

Figure 1. Example of a formula tree generated by Symbolic Regression.

Symbolic Regression with Generative Flow Networks (GFN-SR)

- Deep Symbolic Regression (DSR) makes use of RL to construct expression trees by sampling tokens sequentially and maximizing expected reward [4].
- Difficult to generate diverse high-reward candidates: especially problematic with noise
- GFN-SR instead uses GFlowNets to sample expressions such that their probability is proportional to their reward [2].
- Better balance of exploration versus exploitation.

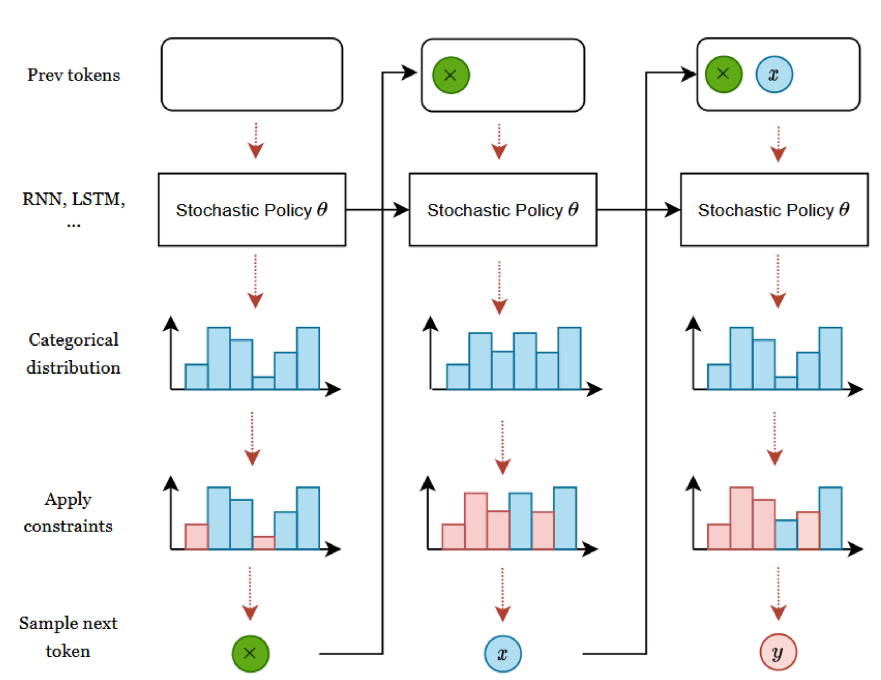

Figure 2. Overview of the expression tree creation. Our method replaces the LSTM with a transformer. Image from [2].

Our Extension

The current version of GFN-SR uses an LSTM RNN to learn stochastic reward policies to generate the expression trees. In our current version, instead we use a Transformer architecture to learn the reward policies.

Methods

We compare the GFN-SR model with the Transformer policy to the RNN policy on a set of synthetic equations. Additionally, we evaluate our method on the AI Feynman benchmark, compared against standard GFN-SR [2], PySR [1], and GPLearn, as well as non-symbolic methods such as RandomForest and DecisionTree.

Synthetic Data

Figure 3, 4 and 5 show 3D visualizations of the equation recovered by GFN-SR with and without the Transformer policy. For many cases, we find performance to be equal. However, in some cases the Transformer outperforms the LSTM. Both models do much better on equations with less terms and less complexity per term.

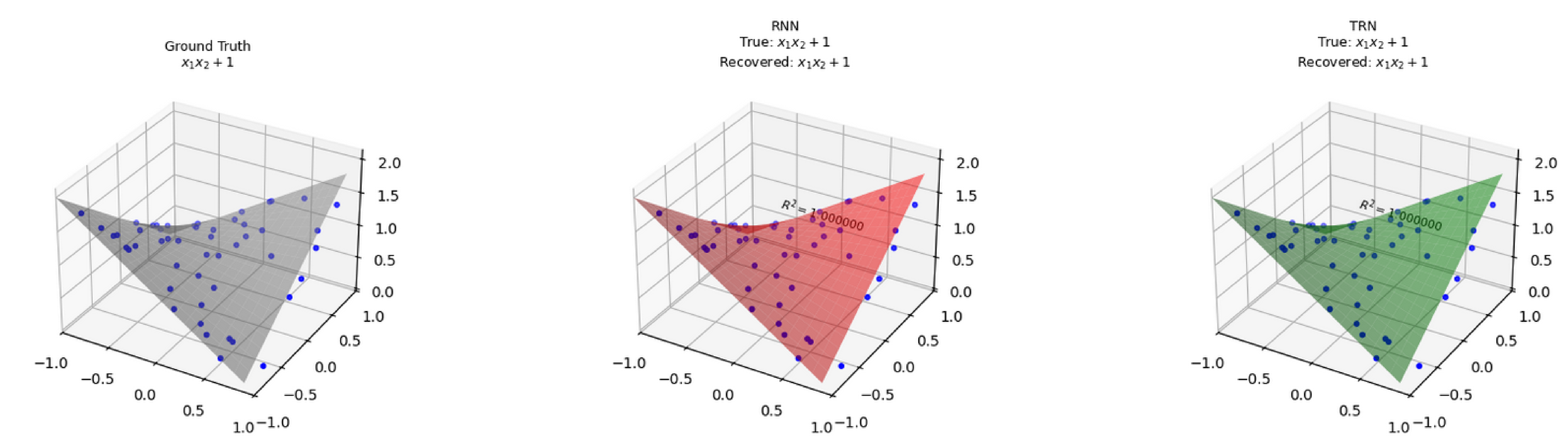

Figure 3. Equation $x_1x_2 + 1$ that both versions recover perfectly.

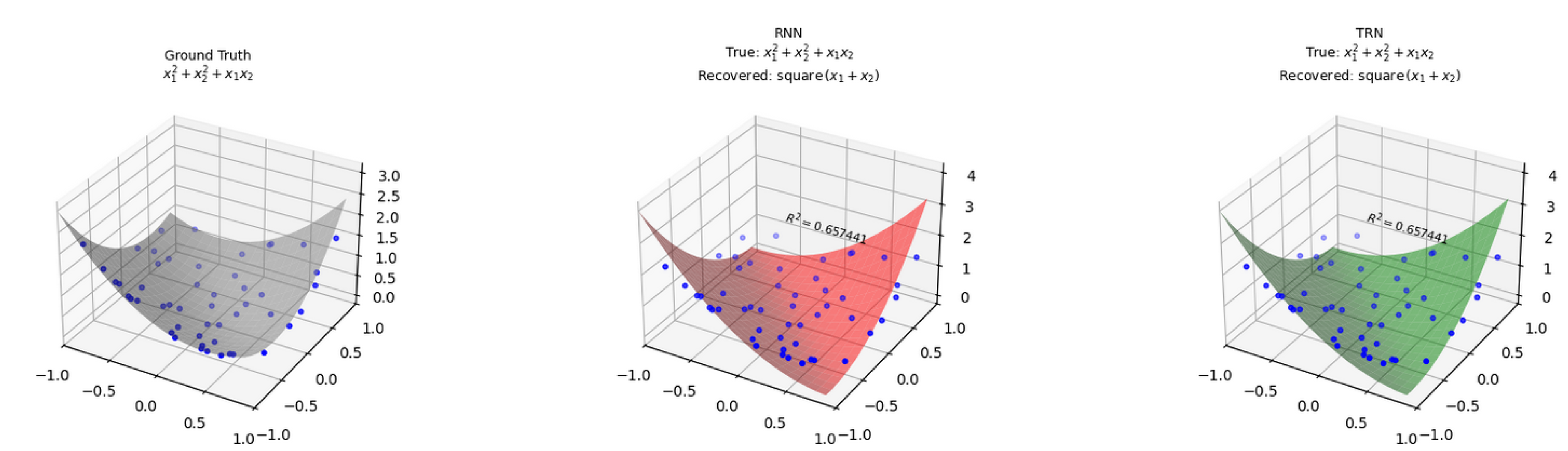

Figure 4. Equation $x_1^2 + x_2^2 + x_1x_2$ that is recovered equally well by both versions.

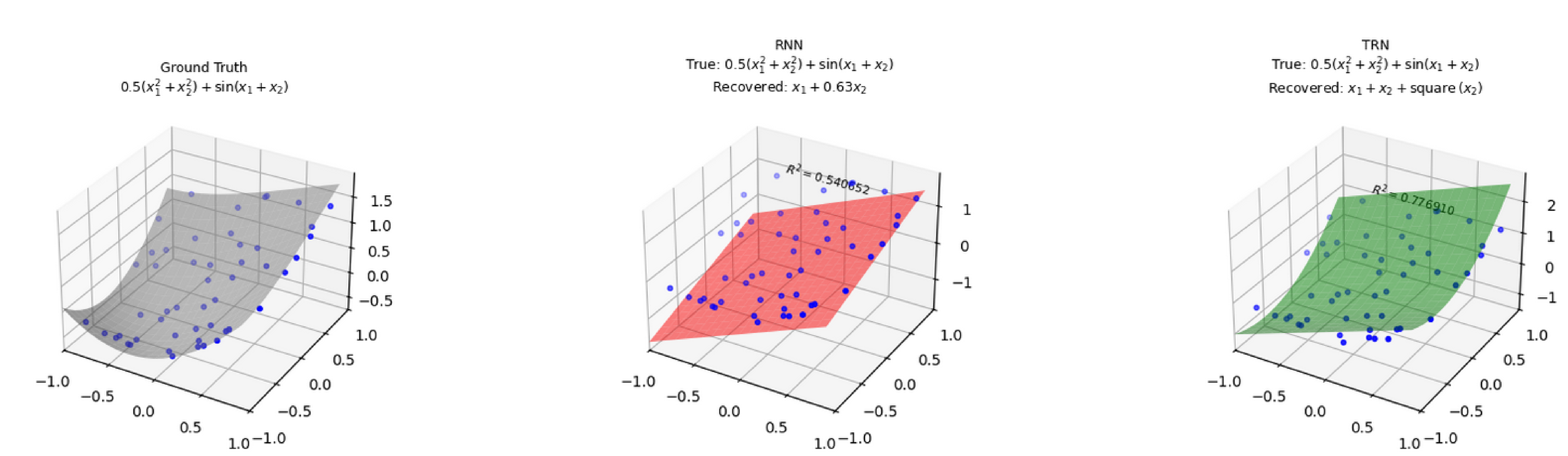

Figure 5. Equation $\frac{1}{2}(x_1^2 + x_2^2) + \sin(x_1 + x_2)$ where the Transformer version outperforms the LSTM (based on R^2).

AI Feynman

Table 1 shows the performance on the AI Feynman benchmark [7, 3]. We find improved performance over the standard GFN-SR model. However, it still underperforms compared to PySR or RandomForest.

Table 1. Performance of multiple (non-)symbolic models compared to our extension of GFN-SR on the AI Feynman benchmark. * ran with 10 iterations. ** ran with 3 iterations.

Method	AI Feynman (R^2)		
	Easy	Medium	Hard
RandomForest	0.721	**0.583**	**0.605**
DecisionTree	0.660	0.530	0.548
GPLearn	0.510	0.568	0.398
PySR	**0.930***	0.574**	0.561**
GFN-SR	0.445	0.350	0.335
GFN-SR + Transformer (**ours**)	0.508	0.351	0.336

Conclusion and Discussion

In general, **GFN-SR seems to struggle with the same types of equations** regardless of the policy network used. Cases with higher order polynomials, nested expressions and logarithms appear difficult. **The speed of recovering equations is slightly higher for the LSTM (5-9 seconds) compared to the Transformer (12-14 seconds) policy.** It has to be noted that the Transformer policy network was deeper than the LSTM network, and the LSTM network did not improve with depth.

We see **a slight increase in performance in both the synthetic and the AI Feynman dataset over the original implementation**, so it seems that the Transformer architecture can improve performance on symbolic regression tasks, but only to a certain extent.

A notable failure case was that **the model sometimes predicted equations that resulted in math errors** (such as $\log(0)$ or zero-division). Further work could focus on constraining the model to prevent numerical instability.

References

[1] Miles D. Cranmer. Interpretable machine learning for science with pysr and symbolicregression.jl. *CoRR*, abs/2305.01582, 2023.

[2] Sida Li, Ioana Marinescu, and Sebastian Musslick. GFN-SR: symbolic regression with generative flow networks. *CoRR*, abs/2312.00396, 2023.

[3] Yoshitomo Matsubara, Naoya Chiba, Ryo Igarashi, and Yoshitaka Ushiku. Rethinking symbolic regression datasets and benchmarks for scientific discovery. *Journal of Data-centric Machine Learning Research*, 2024.

[4] Brenden K. Petersen, Mikel Landajuela, T. Nathan Mundhenk, Cláudio Prata Santiago, Sookyung Kim, and Joanne Taery Kim. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[5] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, 1948.

[6] Silviu-Marian Udrescu, Andrew K. Tan, Jiahai Feng, Orisvaldo Neto, Tailin Wu, and Max Tegmark. AI feynman 2.0: Pareto-optimal symbolic regression exploiting graph modularity. *CoRR*, abs/2006.10782, 2020.

[7] Silviu-Marian Udrescu and Max Tegmark. AI feynman: a physics-inspired method for symbolic regression. *CoRR*, abs/1905.11481, 2019.