Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)
Demonstrations Track

# An Interactive Visualization Platform for Deep Symbolic Regression

**Joanne T. Kim*** , **Sookyung Kim*** , and **Brenden K. Petersen*†**
Lawrence Livermore National Laboratory, Livermore, California, USA
{kim102, kim79, petersen33}@llnl.gov

## Abstract

Discovering tractable mathematical expressions that best explain a dataset is a long-standing challenge in artificial intelligence. This problem, known as *symbolic regression*, is relevant when one seeks to generate new physical knowledge and insights. Since practitioners are primarily interested in knowledge generation, the ability to *interact* with a symbolic regression algorithm would be highly valuable. Thus, we present an interactive symbolic regression framework that allows users not only to configure runs, but also to control the system during training. The interface provides real-time visualization and diagnostics to help guide the user as they control the algorithm on the fly.

## 1 Introduction

Gaining scientific insights from data is an overarching goal of data science. While artificial neural networks provide powerful means of representing complex relationships within data, they are notoriously difficult to interpret. However, having a human-readable and interpretable model is crucial for generating physical insights.

Symbolic regression aims to generate such interpretable models by directly searching the space of *tractable mathematical expressions* to best fit a dataset [Billard and Diday, 2002]. This stands in contrast to conventional regression, in which the practitioner specifies a *fixed* model structure and optimizes the model parameters to fit a given dataset. The succinct expressions discovered via symbolic regression may be readily interpretable and provide useful scientific insights upon inspection and human analysis [Kronberger, 2011].

The space of mathematical expressions is discrete, growing exponentially with the length of an expression. This combinatorial search space renders symbolic regression a challenging machine learning problem. The conventional approach to symbolic regression is genetic programming, an evolutionary algorithm in which a population of mathematical expressions is "evolved" using evolutionary operations like selection, crossover, and mutation [Koza, 1992; Uy *et al.*, 2011; Bäck *et al.*, 2018]. Recently, deep symbolic regression (DSR) [Petersen, 2019] was proposed as a deep learning method for symbolic regression based on policy gradients [Kakade, 2002]. DSR outperforms genetic programming in its ability to recover symbolic expressions on a series of benchmark symbolic regression tasks.

Because the ultimate goal of symbolic regression is to gain scientific insights (rather than simply achieve the best model fit), it would be beneficial if the user could directly interact with their system.

An interactive platform would allow users to help guide the training process and identify the expressions most useful for their use case. While DSR can incorporate domain-specific constraints in situ (e.g., ensuring that all sampled expressions maintain the correct physical units), it may be difficult for users to encode more nuanced expert knowledge and domain-specific constraints into the algorithm. Thus, we developed a demonstration system for symbolic regression that facilitates

*All authors contributed equally
†Corresponding author

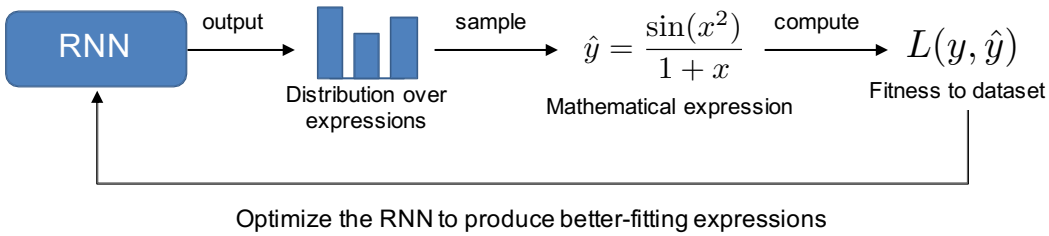


Figure 1: Overview of core algorithm: Deep Symbolic Regression. An RNN emits a distribution over mathematical expressions. Expressions are sampled from the distribution and evaluated according to their fitness to a dataset. The RNN is trained using reinforcement learning such that better fitting expressions become more likely under the distribution.

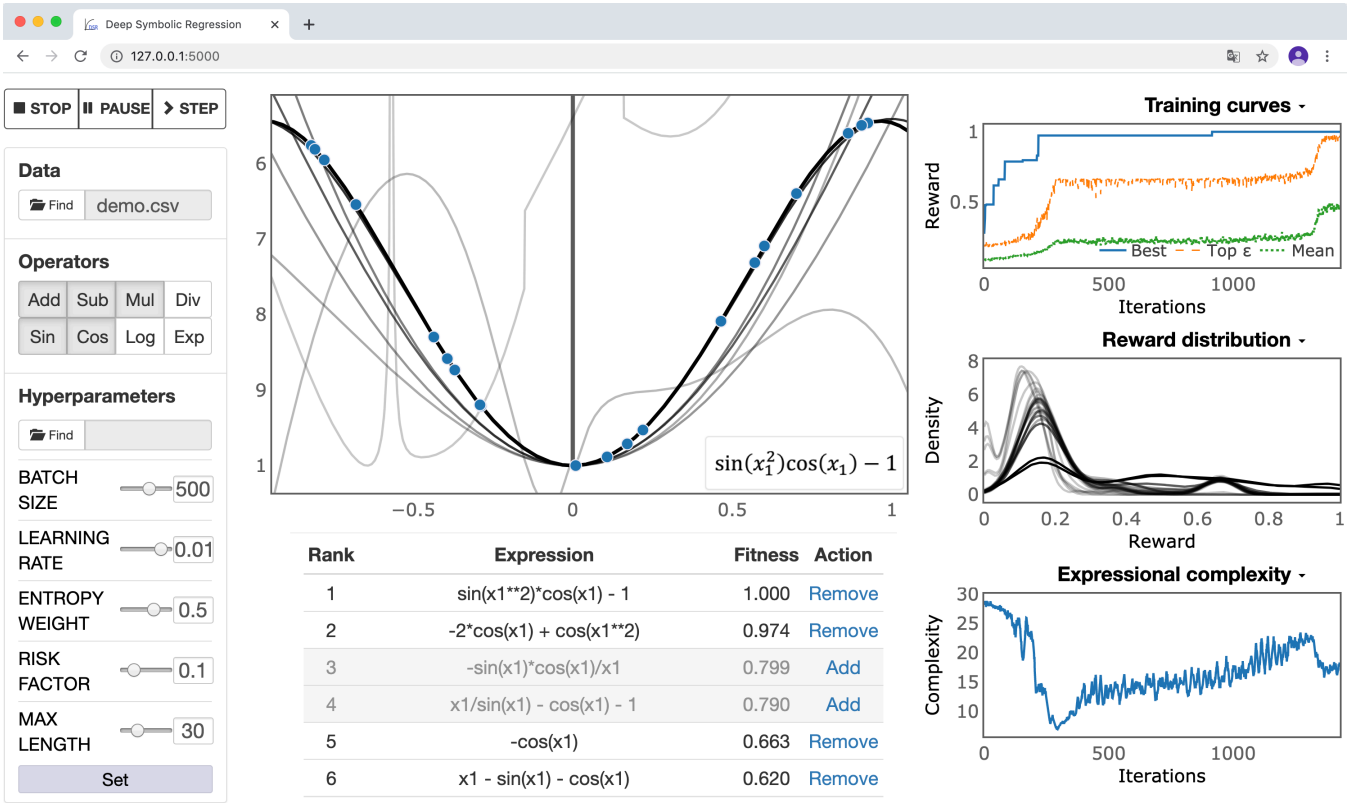


Figure 2: User interface for the demonstration platform. Left panel: Control. Middle panel: Visualization. Right panel: Diagnostics.

human-in-the-loop interaction with the training process. Our demonstration system consists of the *core algorithm* (DSR) running in the background, a *visualization platform* to monitor the training process, and a customizable *user interface* to upload data, configure runs, and interact with training in real time.

## 2 Core Algorithm: Deep Symbolic Regression

The core algorithm backing our system is DSR, a gradient-based approach for symbolic regression based on deep reinforcement learning [Petersen, 2019].[1] DSR constructs mathematical expressions in Polish notation as sequences of mathematical building blocks or "tokens" specifying operators (e.g., $\sin, \cos, +, -, \times, \div$), input variables, and constants.

A high-level overview of the algorithm is shown in Fig. 1. DSR uses a recurrent neural network (RNN) [Mikolov *et al.*, 2010] to emit a distribution over mathematical expressions. Thus, samples from the RNN are concrete expressions. At each training step, a batch of expressions is sampled from the RNN and each expression is scored based on a user-specified fitness to the dataset. This fitness then is used as the reward signal to train the RNN using a novel risk-seeking policy gradient algorithm [Petersen, 2019].

[1]Source code for DSR is available at https://github.com/brendenpetersen/deep-symbolic-regression.

## 3 Demonstration Platform

Our platform is web-based, meaning the user interacts with the algorithm via a web browser while training is performed on the server side. This client-server architecture allows users to leverage large-scale computing resources for training, while the client side only handles visualization. However, we note that a single execution of DSR can complete within just several minutes on a single processor; thus, the user can easily perform both training and visualization on a modern laptop.

A snapshot of the user interface is shown in Fig. 2. It consists of three sub-components: (1) a control panel, (2) a visualization panel, and (3) a diagnostics panel. We describe each component in the following sections.

### 3.1 Control

The control panel (Fig. 2, left) is used to configure runs and interact with the algorithm during training. Before training, the user uploads a dataset and specifies which mathematical operators or "tokens" to allow. Once training begins, the user can interact with the system in real time in several ways. The various hyperparameters, which are described in Table 1 and further detailed in [Petersen, 2019], can be edited on the fly. For example, the user may choose to increase exploration during training if the top expressions are too similar, or if the algorithm appears to begin converging prematurely to a suboptimal expression. The user may also manually select particular expressions they wish to become more (or less) important.

| Parameter | Description |
|---|---|
| Batch size | Number of expressions to sample from the RNN each iteration. |
| Learning rate | Learning rate for Adam optimizer used to train the RNN. |
| Entropy weight | Coefficient used for entropy regularization. This controls the degree of exploration over exploitation. |
| Risk factor | Fraction of top-performing expressions to train on each batch. |
| Max length | Maximum allowable number of tokens in an expression. |

Table 1: Key hyperparameters used in deep symbolic regression.

These selected expressions are added to (or removed from) a *priority queue* that encourages (or discourages) the RNN to generate similar expressions [Abolafia *et al.*, 2018].

### 3.2 Visualization

The visualization panel (Fig. 2, middle) shows the underlying dataset (blue dots) along with curves of the top-performing expressions as training progresses. For datasets with multiple input variables, the user can specify 1-D slices of the data. We use an "onion skinning" effect to illustrate how the best expressions evolve over time: the curves for less recently discovered expressions are more translucent. The user can select particular curves to view additional characteristics of the corresponding expression, such as its LaTeX representation or underlying algebraic expression tree. By monitoring the best-fitting expression during training, users can visually evaluate the performance of DSR and use their intuition and domain-specific knowledge to interactively control the algorithm.

### 3.3 Diagnostics

The diagnostics panel (Fig. 2, right) provides additional information to help diagnose the quality of training. For example, we provide real-time traces of the mean fitness of each batch, the $\varepsilon$-quantile fitness of each batch, and the best fitness found so far (Fig. 2, top right). We also provide distributions of the reward at various training iterations, again using onion skinning to illustrate training progress (Fig. 2, bottom right). This metric is particularly useful for determining whether the distribution is converging prematurely, typically manifested by the reward distribution quickly sharpening at a poor reward. Alternate plots including various loss functions, expressional complexity measures, and fitness functions can be selected for display by the user.

## 4 Conclusion

We present a demonstration system for DSR, a symbolic regression algorithm based on reinforcement learning. The framework includes visualization of the top-performing expressions and real-time diagnostics of the training process. Most importantly, the system allows the user to interactively guide the system during training by dynamically changing hyperparameters and manually selecting expressions to become more or less prominent.

## Acknowledgements

We thank Mikel Landajuela, Claudio Santiago, and Nathan Mundhenk for their valuable feedback and support. This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC. LLNL-CONF-805899.

## References

[Abolafia *et al.*, 2018] Daniel A Abolafia, Mohammad Norouzi, Jonathan Shen, Rui Zhao, and Quoc V Le. Neural program synthesis with priority queue training. *arXiv preprint arXiv:1801.03526*, 2018.

[Bäck *et al.*, 2018] Thomas Bäck, David B Fogel, and Zbigniew Michalewicz. *Evolutionary Computation 1: Basic Algorithms and Operators*. CRC press, 2018.

[Billard and Diday, 2002] Lynne Billard and Edwin Diday. Symbolic regression analysis. In *Classification, Clustering, and Data Analysis*, pages 281–288. Springer, 2002.

[Kakade, 2002] Sham M Kakade. A natural policy gradient. In *Advances in neural information processing systems*, pages 1531–1538, 2002.

[Koza, 1992] John R Koza. *Genetic programming: on the programming of computers by means of natural selection*, volume 1. MIT press, 1992.

[Kronberger, 2011] Gabriel Kronberger. Symbolic regression for knowledge discovery: bloat, overfitting, and variable interaction networks. *ACM SIGEVOlution*, 5(4):25–25, 2011.

[Mikolov *et al.*, 2010] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*, 2010.

[Petersen, 2019] Brenden K Petersen. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. *arXiv preprint arXiv:1912.04871*, 2019.

[Uy *et al.*, 2011] Nguyen Quang Uy, Nguyen Xuan Hoai, Michael O'Neill, Robert I McKay, and Edgar Galván-López. Semantically-based crossover in genetic programming: application to real-valued symbolic regression. *Genetic Programming and Evolvable Machines*, 12(2):91–119, 2011.