Symbolic Regression via Neural-Guided Genetic Programming Population Seeding

T. Nathan Mundhenk
Mikel Landajuela
Ruben Glatt
Claudio P. Santiago
Daniel M. Faissol
Brenden K. Petersen

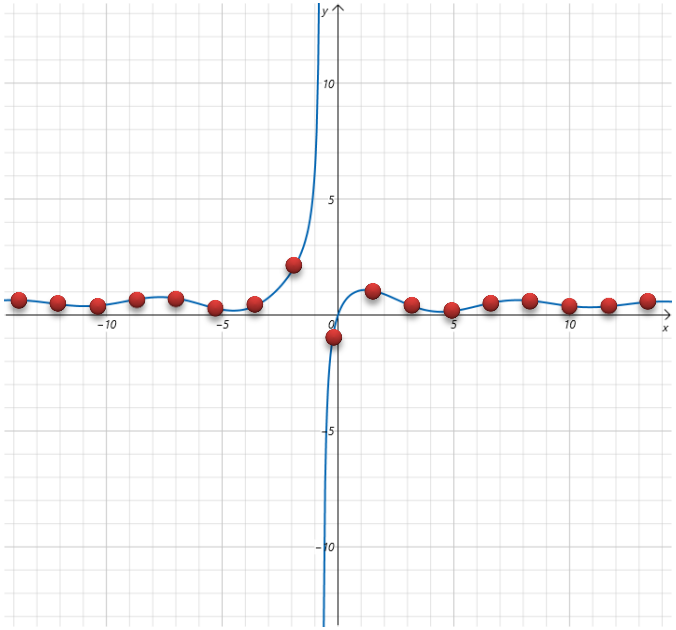
LLNL-PRES-827962
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

What is symbolic regression?

- **Goal:** Given a set of observed data points, reconstruct the *exact* generating equation.
- Is believed to be *NP-Hard*. However, a formal proof does not exist.
- Is a special case of *Discrete Sequence Optimization*.
- An ideal use case is to find the underlying law and equation(s) that fit a set of physical observed data points.

Input: Data points (X,y) along unknown $f(x)$



Output: Exact generating expression f

$$\frac{x + \pi \cdot \sin(x)}{2x + \sqrt{2}}$$

Given a dataset (X, y), where each point $X_i \in \mathbb{R}^n$ and $y_i \in \mathbb{R}$, find a mathematical expression $f : \mathbb{R}^n \to \mathbb{R}$ such that $f(X_i) \approx y_i$.

Discrete sequence optimization

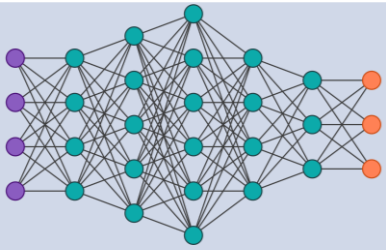
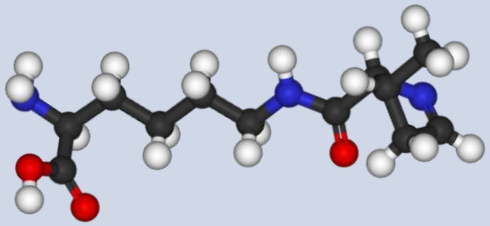
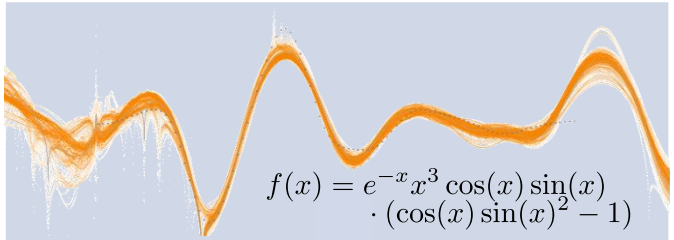
Search space ~ $|\mathcal{L}|^N$

cf. 10^{80} atoms/universe

- Many problems fall into this category:

$$\underset{n \leq N, \tau_1, \ldots, \tau_n}{\arg\max} \; [R(\tau_1, \ldots, \tau_n)] \;\; \text{with } \tau_i \in \mathcal{L} = \{\alpha, \beta, \ldots, \zeta\}$$

> *Optimize a sequence of discrete tokens under a black-box reward function.*

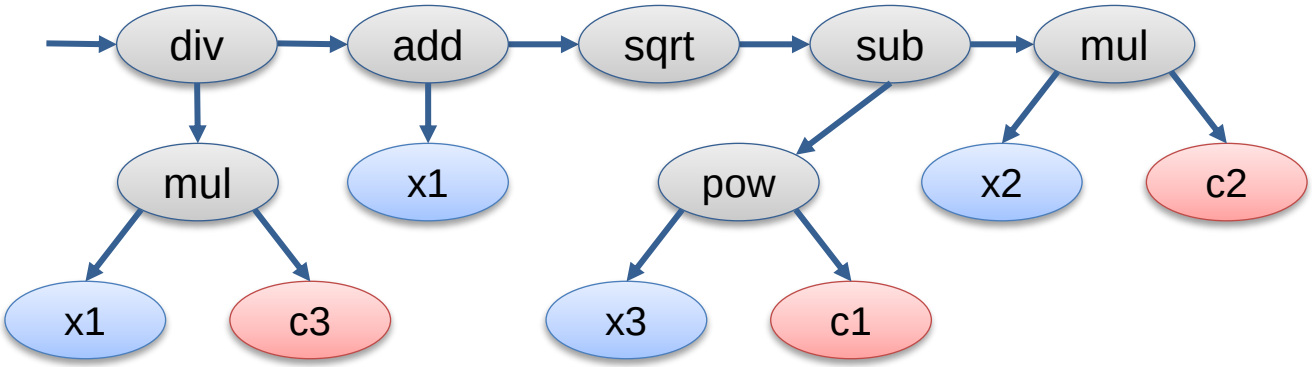
Neural architecture search	Antibody design	Symbolic regression
		
$\mathcal{L} = \{\{\text{ReLU}, \tanh\}, \{32, 64\}\}$	$\mathcal{L} = \{\text{ALA}, \text{ARG}, \ldots, \text{VAL}\}$	$\mathcal{L} = \{+, -, \times, \div, \sin, x, \ldots\}$
$R = \text{validation accuracy}$	$R = \text{binding affinity}$	$R = -\text{MSE}$

[Zoph et al. 2016]

Representing an expression in a way we can use

An example expression in human readable form.

$$y = \frac{x_1 + \sqrt{x_3^2 - 4x_2}}{2x_1}$$



The expression as a tree representation that can be easier to manipulate and make changes to.

The expression in a form the computer can execute, but is human readable.

```
div(add(x1,sqrt(sub(pow(x3,c1),mul(x2,c2)))),mul(x1,c3))
```

```
t4,t1,x1,t5,t2,t6,x3,c1,t3,x2,c2,t3,x1,c3
```

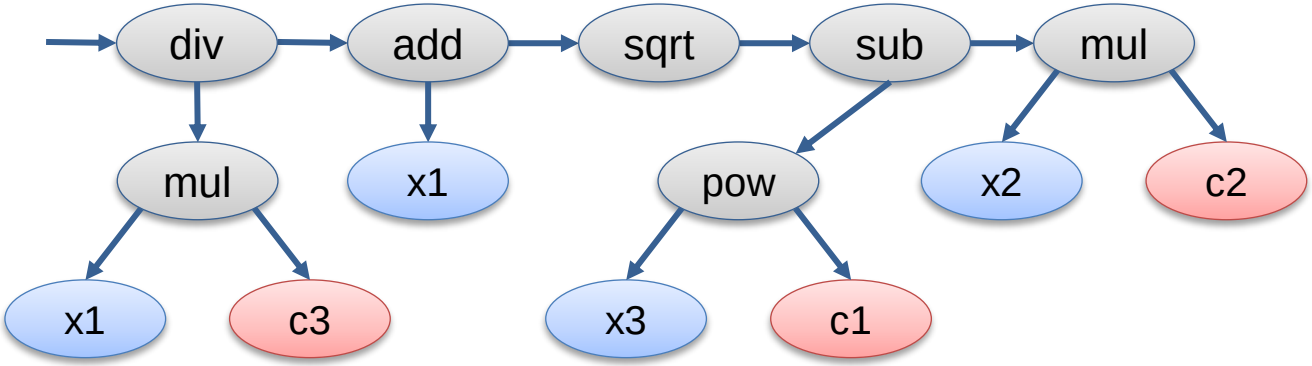
The expression as a string of tokens we can sequentially emit from a neural network or manipulate using genetic programming.

Genetic Programming Primer

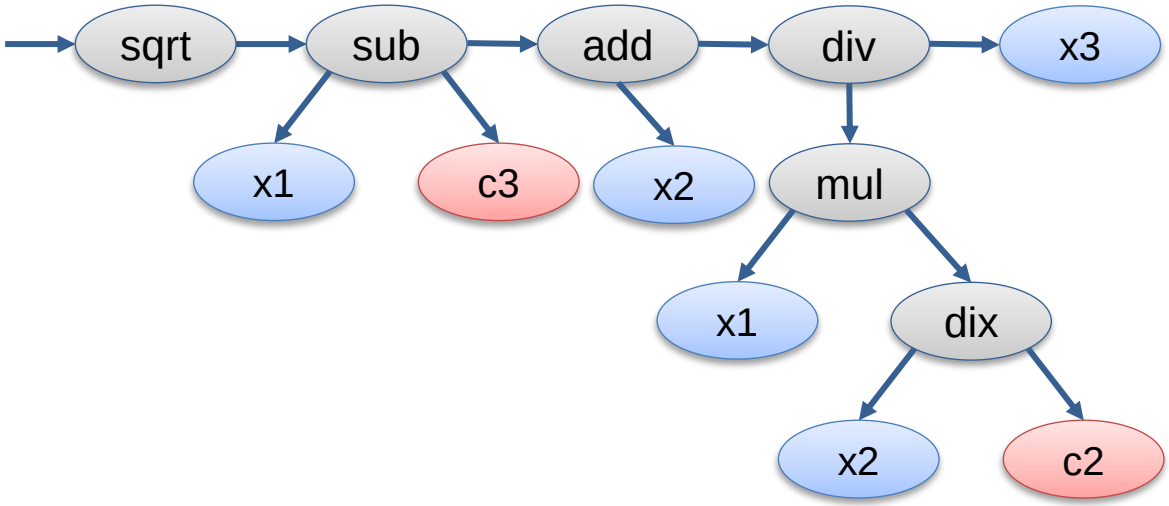
- Create N random expressions : A population of individuals
  - With a fixed probability, mutate a subset of the population.
    - There are lots of different ways to mutate an individual.
  - With a fixed probability, crossover (mate) a subset of the population.
  - Select the best subset:
    - A common way is via Tournament where we put three or more randomly individuals together and take the best scoring individual and remove the others.
    - Since crossover can increase population size, we can do tournament until the population gets back to the original size.
  - Repeat until convergence (or other criteria)
- There are **lots** of variations to how GP can be done.
- It's all a little bit hand wavy, but it works for many applications.

Crossover Example

Expression 1

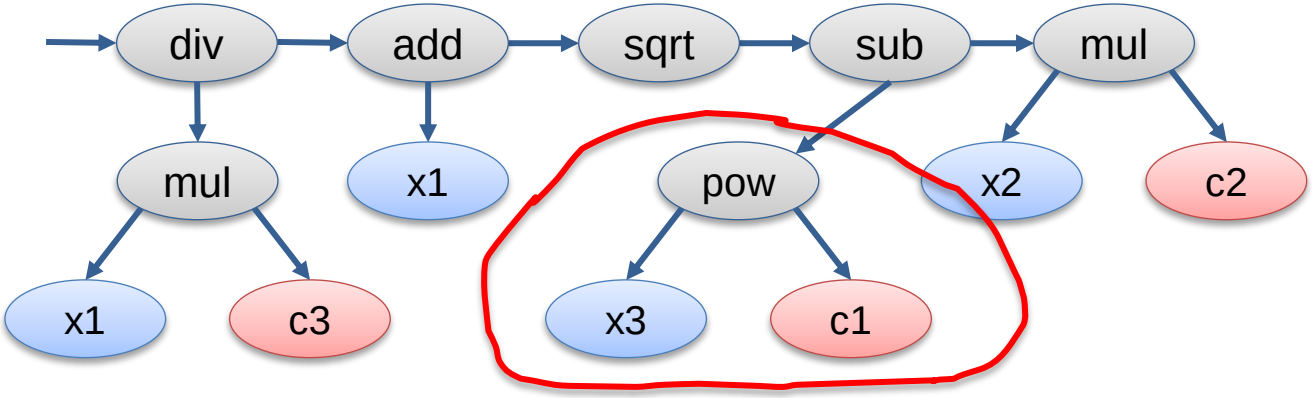


Expression 2

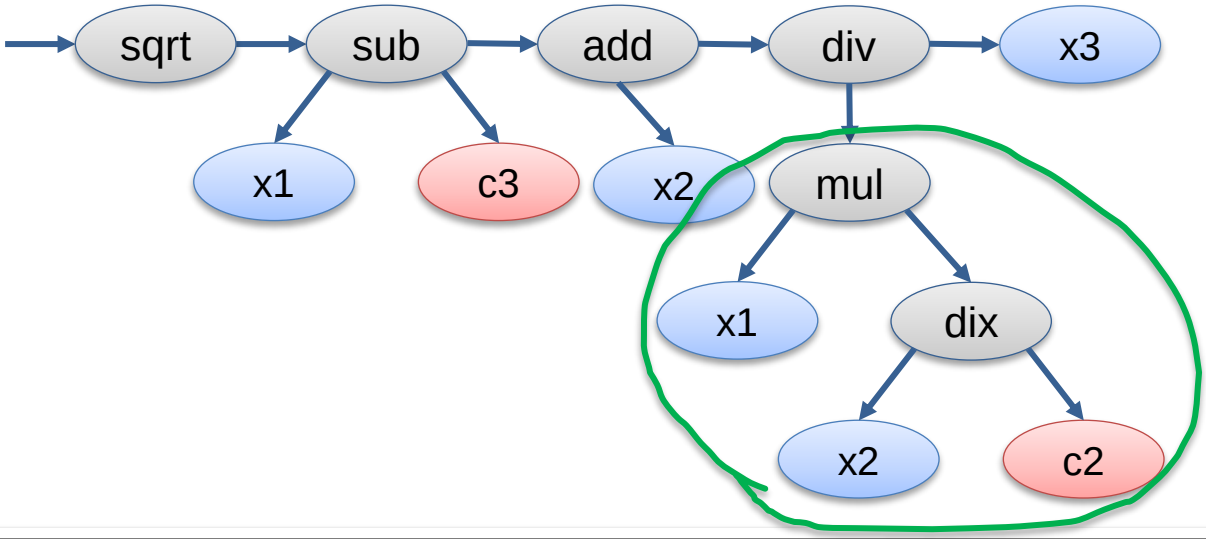


Crossover Example

Expression 1

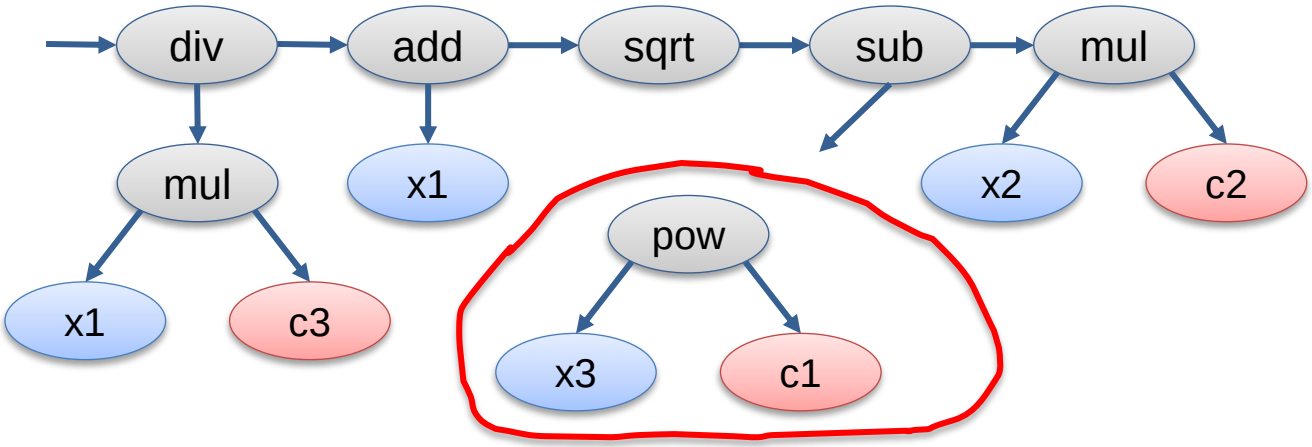


Expression 2

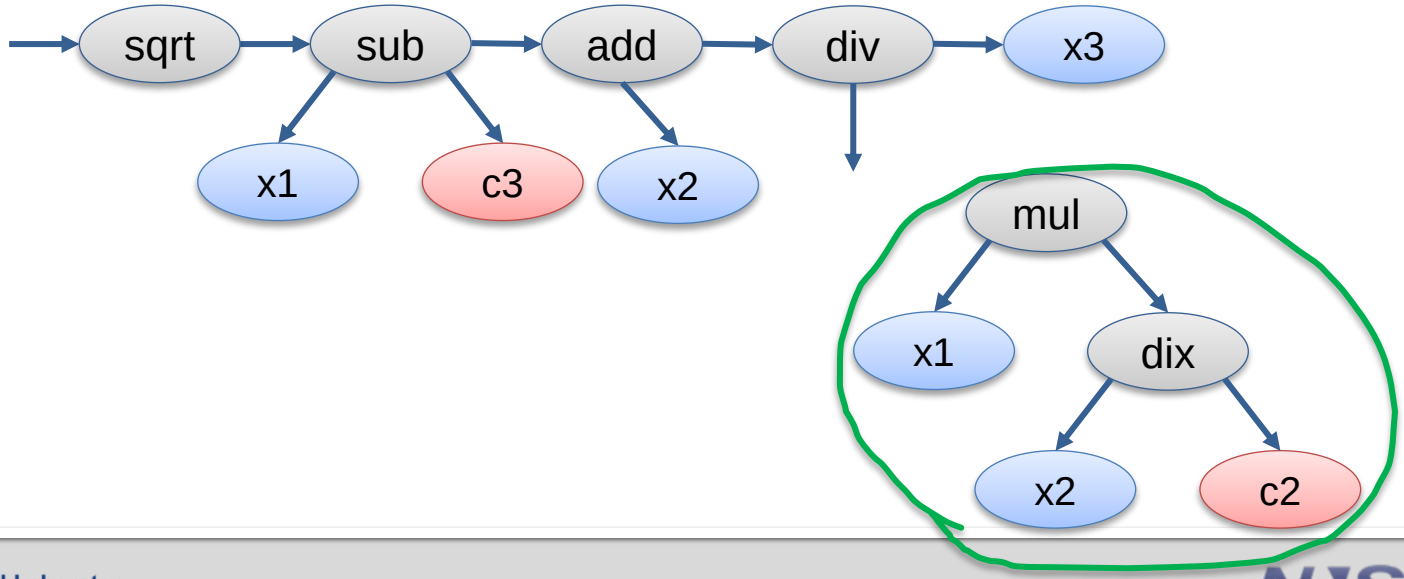


Crossover Example

Expression 1

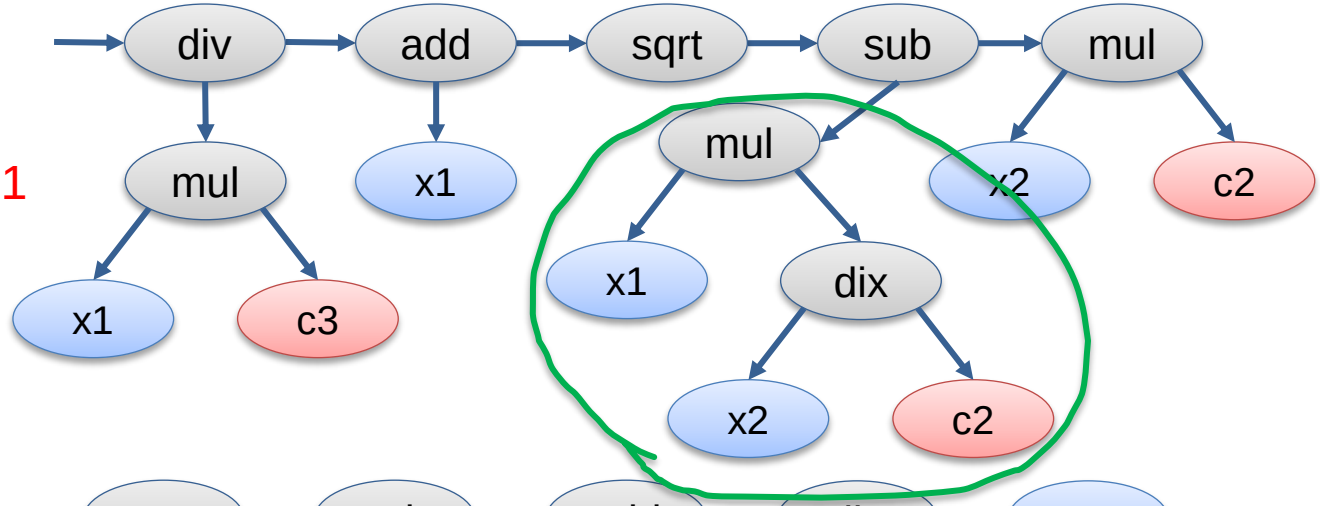


Expression 2

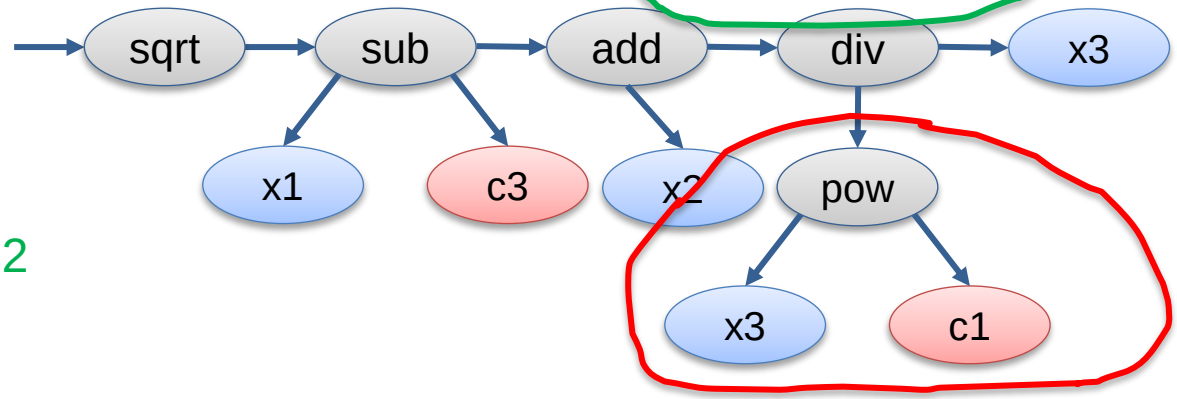


Crossover Example

New Expression 1

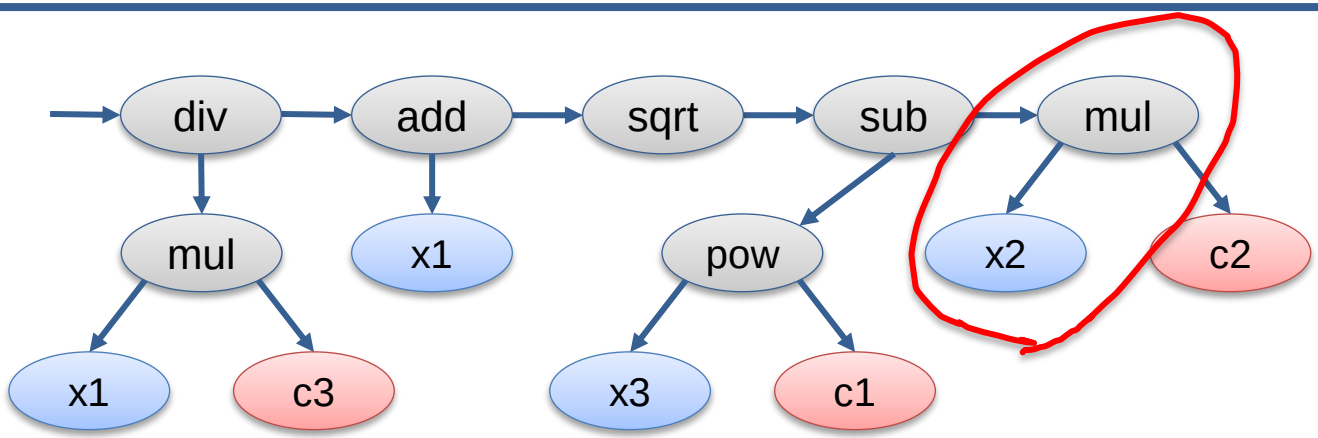


New Expression 2

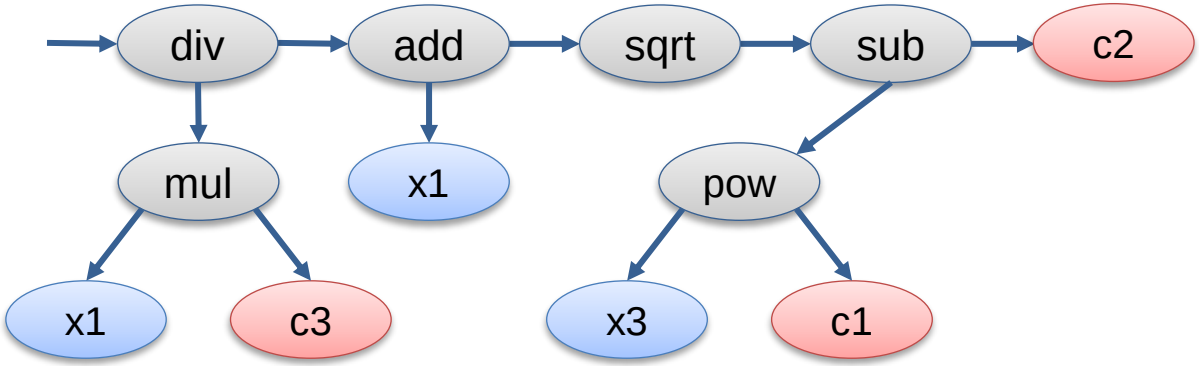


Mutation Example (Shrink)

Expression 1

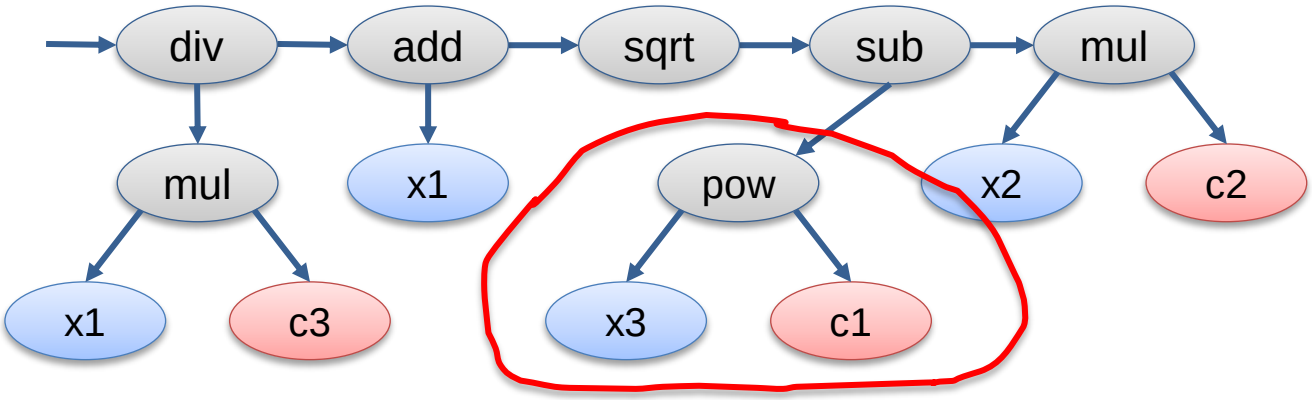


New Expression 1

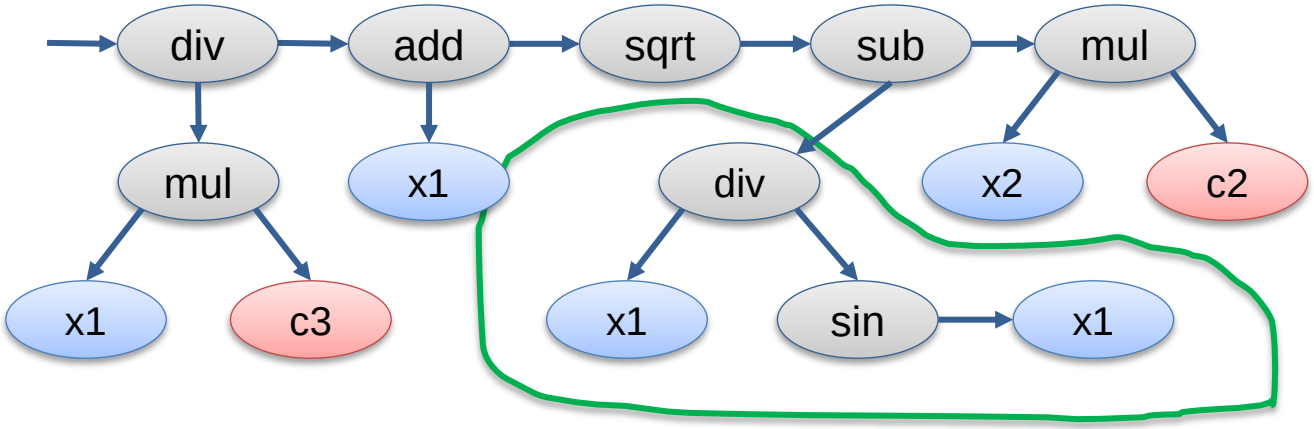


Mutation Example (Replace)

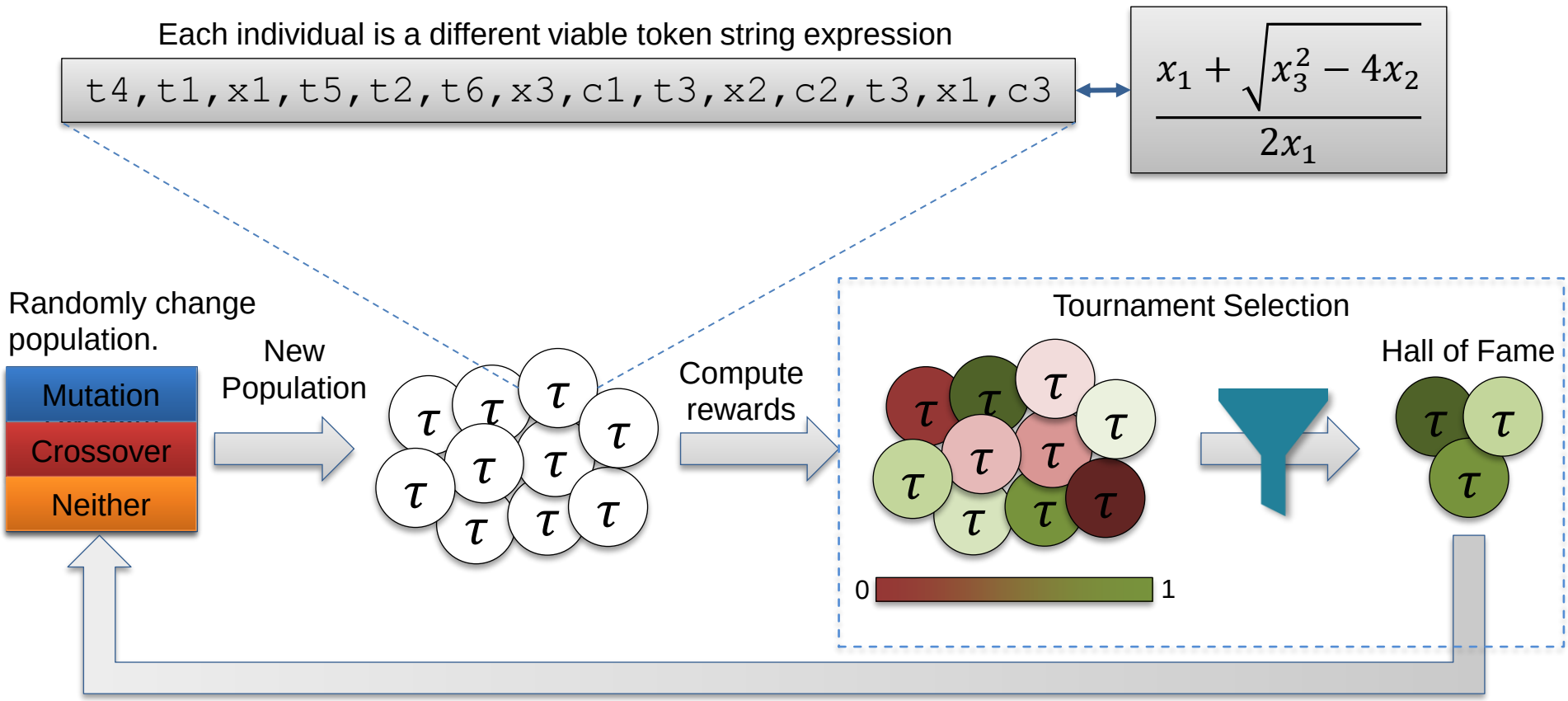
Expression 1



New Expression 1

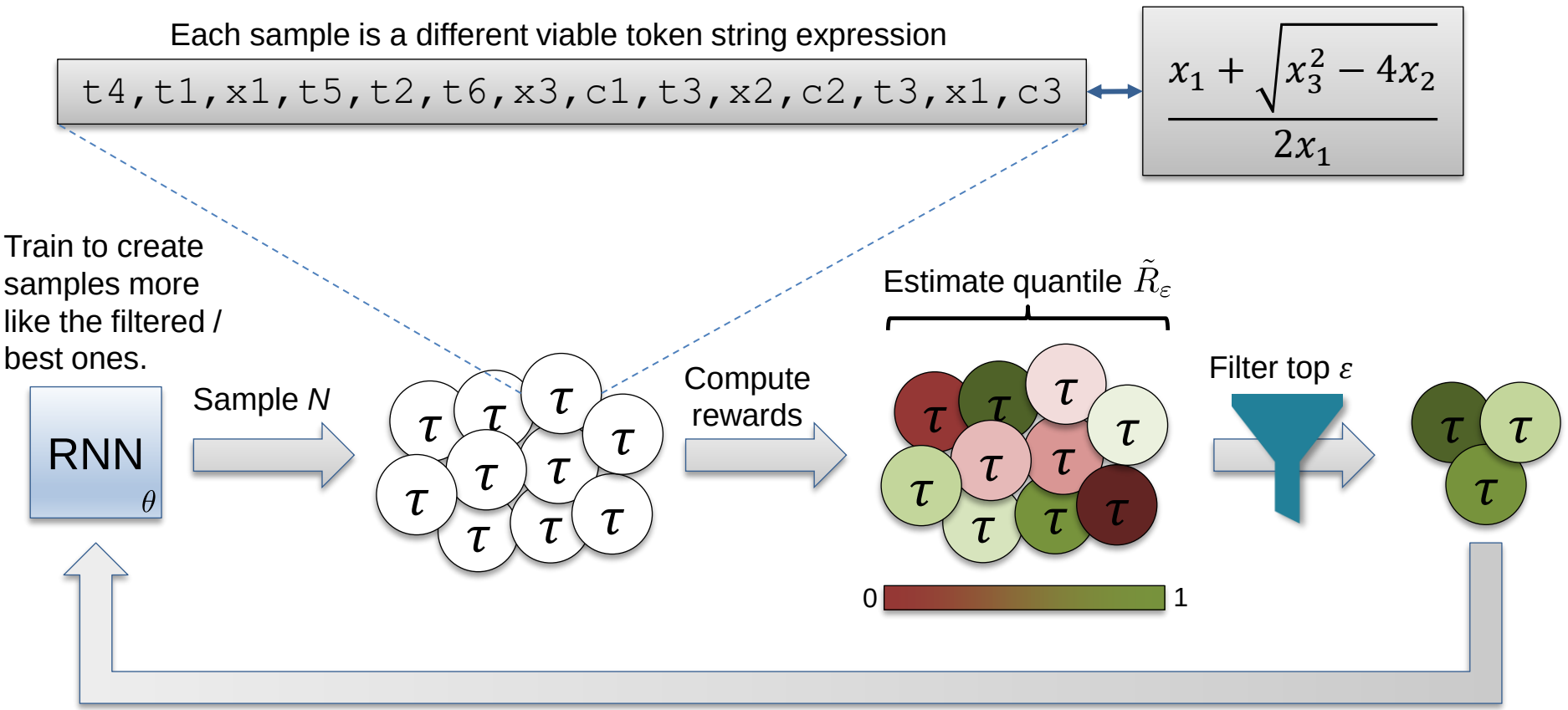


Genetic programming symbolic regression (GP)



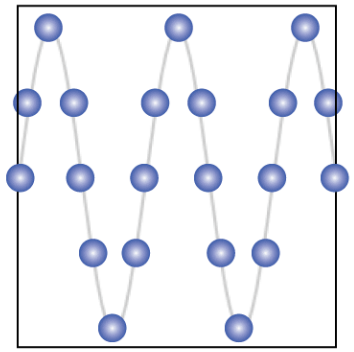
Return population for possible mutation and crossover.

Deep symbolic regression (DSR)



Train on filtered batch: $\nabla_\theta J(\theta) \approx \frac{1}{\varepsilon N} \sum_{\tau \in \Gamma} \left(R(\tau) - \tilde{R}_\varepsilon \right) \nabla_\theta \log p(\tau|\theta)$

Combining Genetic Programming with Deep Symbolic Regression



Symbolic Regression

1 + sin(x)

GP Hall of Fame Expressions

Loop GP s times.

GP

Combine samples

Seed from RNN samples.

Draw n new samples from RNN

Train

Sample

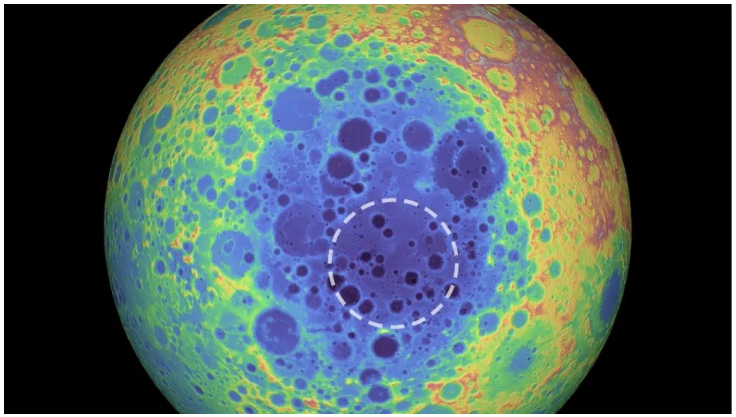
RNN θ

START

Train on combined batch

Intuition

- GP may easily overshoot a solution since it has no explicit distance update metric like DSR or other gradient methods.
  - This has an effect on GP somewhat akin to a Trust Region approach.
- Random restart helps GP, but it would be better if each time GP restarted a little closer to the target.
  - Works best if solution is in one of many small gradient basins contained inside a larger basin. DSR will follow the large basin, but GP is better at finding the small basin.
- DSR is thus used to govern the distance GP travels and keep it from straying too far from the likely solution location.
- DSR itself is more likely to get stuck in local minima. GP provides a greater ability for DSR to break free.
- A colorful example might be a human (DSR) walking a dog (GP) looking for something.



Results on Nguyen benchmark

- Experimental setup:
  - Nguyen benchmarks
    - Only 20 data points!
  - Success = symbolic equivalence

- Baselines:
  - DSR: Deep symbolic regression [Petersen et al. 2021]
  - PQT: Priority queue training [Abolafia et al. 2018]
  - VPG: "Vanilla" policy gradient [Williams 1992]
  - GP: Genetic programming [Koza 1992]
  - Eureqa: Commercial software [DataRobot, Inc.]

Benchmark	Expression	Recovery rate (%) Ours	DSR	PQT	VPG	GP	Eureqa
Nguyen-1	$x^3 + x^2 + x$	100	100	100	96	100	100
Nguyen-2	$x^4 + x^3 + x^2 + x$	100	100	99	47	97	100
Nguyen-3	$x^5 + x^4 + x^3 + x^2 + x$	100	100	86	4	100	95
Nguyen-4	$x^6 + x^5 + x^4 + x^3 + x^2 + x$	100	100	93	1	100	70
Nguyen-5	$\sin(x^2)\cos(x) - 1$	100	72	73	5	45	73
Nguyen-6	$\sin(x) + \sin(x + x^2)$	100	100	98	100	91	100
Nguyen-7	$\log(x + 1) + \log(x^2 + 1)$	97	35	41	3	0	85
Nguyen-8	$\sqrt{x}$	100	96	21	5	5	0
Nguyen-9	$\sin(x) + \sin(y^2)$	100	100	100	100	100	100
Nguyen-10	$2\sin(x)\cos(y)$	100	100	91	99	76	64
Nguyen-11	x^y	100	100	100	100	7	100
Nguyen-12	$x^4 - x^3 + \frac{1}{2}y^2 - y$	0	0	0	0	0	0
	Average	**91.4**	83.6	75.2	46.7	60.1	73.9

Livermore benchmark set

- A new hand made benchmark with 22 new equations.
- Was created by us prior to this work, but has not yet been published.
- Is needed since Nguyen is mostly too easy anymore.

Given token set: $\{+, -, \times, \div, sin, cos, exp, log, x, y\}$ find:

Livermore-1	$\frac{1}{3} + x + \sin\left(x^2\right)$	$U\left(-10, 10, 1000\right)$
Livermore-2	$\sin\left(x^2\right)\cos\left(x\right) - 2$	$U\left(-1, 1, 20\right)$
Livermore-3	$\sin\left(x^3\right)\cos\left(x^2\right) - 1$	$U\left(-1, 1, 20\right)$
Livermore-4	$\log(x+1) + \log(x^2+1) + \log(x)$	$U\left(0, 2, 20\right)$
Livermore-5	$x^4 - x^3 + x^2 - y$	$U\left(0, 1, 20\right)$
Livermore-6	$4x^4 + 3x^3 + 2x^2 + x$	$U\left(-1, 1, 20\right)$
Livermore-7	$\sinh(x)$	$U\left(-1, 1, 20\right)$
Livermore-8	$\cosh(x)$	$U\left(-1, 1, 20\right)$
Livermore-9	$x^9 + x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x$	$U\left(-1, 1, 20\right)$
Livermore-10	$6\sin\left(x\right)\cos\left(y\right)$	$U\left(0, 1, 20\right)$
Livermore-11	$\frac{x^2 x^2}{x+y}$	$U\left(-1, 1, 50\right)$
Livermore-12	$\frac{x^5}{y^3}$	$U\left(-1, 1, 50\right)$
Livermore-13	$x^{\frac{1}{3}}$	$U\left(0, 4, 20\right)$
Livermore-14	$x^3 + x^2 + x + \sin\left(x\right) + \sin\left(x^2\right)$	$U\left(-1, 1, 20\right)$
Livermore-15	$x^{\frac{1}{5}}$	$U\left(0, 4, 20\right)$
Livermore-16	$x^{\frac{2}{5}}$	$U\left(0, 4, 20\right)$
Livermore-17	$4\sin\left(x\right)\cos\left(y\right)$	$U\left(0, 1, 20\right)$
Livermore-18	$\sin\left(x^2\right)\cos\left(x\right) - 5$	$U\left(-1, 1, 20\right)$
Livermore-19	$x^5 + x^4 + x^2 + x$	$U\left(-1, 1, 20\right)$
Livermore-20	$\exp\left(-x^2\right)$	$U\left(-1, 1, 20\right)$
Livermore-21	$x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x$	$U\left(-1, 1, 20\right)$
Livermore-22	$\exp\left(-0.5x^2\right)$	$U\left(-1, 1, 20\right)$

Results on several benchmarks

- Each of 37 benchmarks is run 25 times for 2M evaluations.
- Three benchmarks sets are used.
- Shown is how often equations are recovered.
- GEGL is a method published at the time of this analysis. It's similar enough, that we had to add it to our comparison even though it was written for a molecular design task.

	Recovery rate (%)			
	All	Nguyen	R	Livermore
Ours	**74.92**	**92.33**	**33.33**	**71.09**
GEGL [Ahn et al., 2020]	64.11	86.00	**33.33**	56.36
Random Restart GP (i.e. GP only)	63.57	88.67	2.67	58.18
DSR (i.e. RNN only) [Petersen et al., 2021]	45.19	83.58	0.00	30.41
95% confidence interval	±1.54	±1.76	±2.81	±1.32

- We can recover all but two benchmark equations.

	Benchmark problems recovered			
	DSR	Random Restart GP	GEGL	Ours
Nguyen (12 possible)	11	11	11	**12**
R (3 possible)	0	2	1	**3**
Livermore (22 possible)	13	16	17	**20**
All (37 possible)	24	29	29	**35**

Results on benchmarks with unknown constants

Jin benchmark [Jin et al. 2019]

	Mean RMSE			
	Ours	DSR	BSR	Recovered by Ours
Jin-1	**0**	0.46	2.04	Yes
Jin-2	**0**	**0**	6.84	Yes
Jin-3	**0**	0.00052	0.21	Yes
Jin-4	**0**	0.00014	0.16	Yes
Jin-5	**0**	**0**	0.66	Yes
Jin-6	**0**	2.23	4.63	Yes
Average	**0**	0.45	2.42	

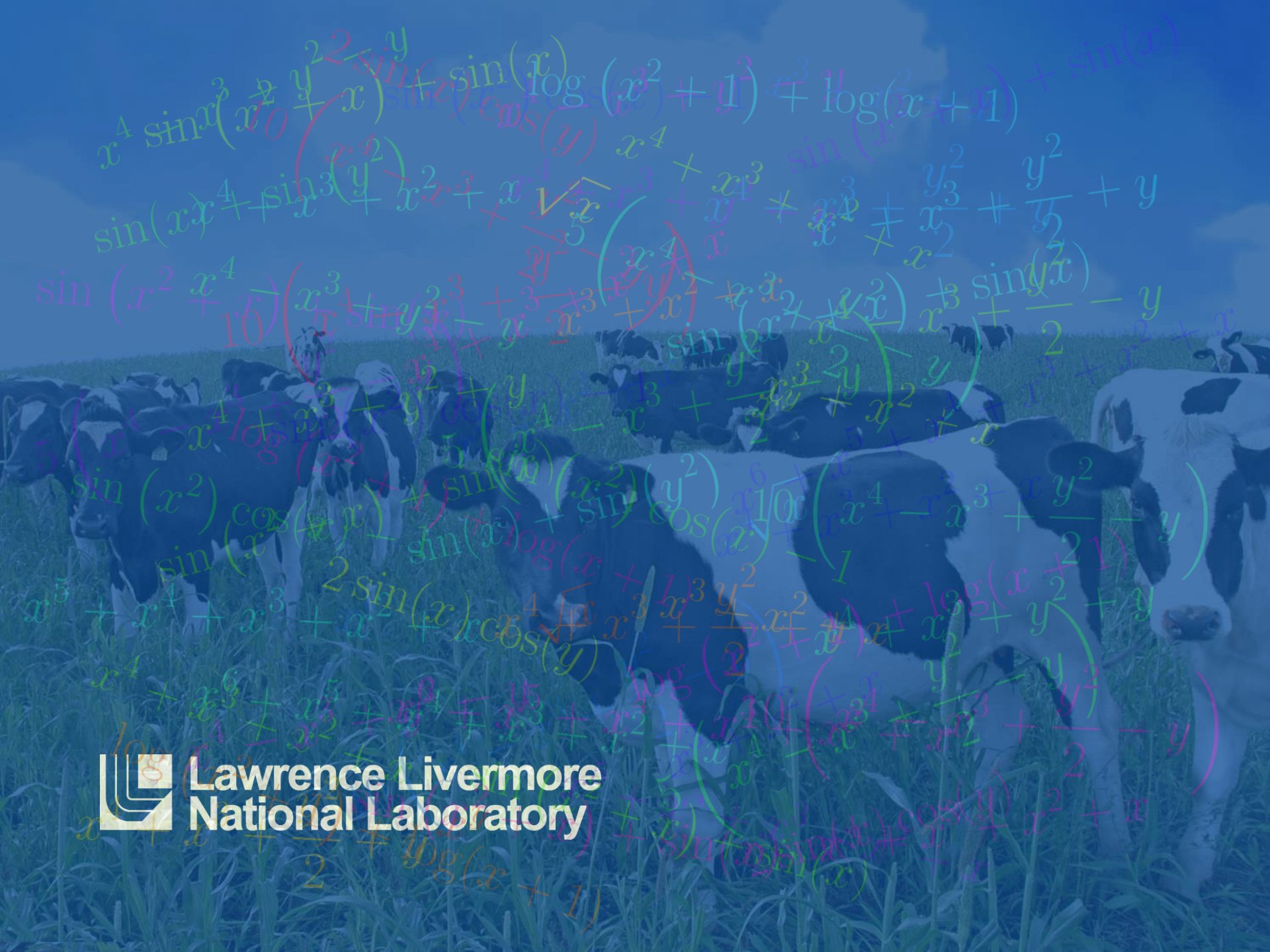
Neat benchmark [Trujillo et al. 2016]

	Median RMSE			
	Ours	DSR	Neat-GP	Recovered by Ours
Neat-1	**0**	**0**	0.0779	Yes
Neat-2	**0**	**0**	0.0576	Yes
Neat-3	**0**	0.0041	0.0065	Yes
Neat-4	**0**	0.0189	0.0253	Yes
Neat-5	**0**	**0**	0.0023	Yes
Neat-6	$\mathbf{6.1 \times 10^{-6}}$	0.2378	0.2855	—
Neat-7	**1.0028**	1.0606	1.0541	—
Neat-8	**0.0228**	0.1076	0.1498	—
Neat-9	**0**	0.1511	0.1202	Yes
Average	**0.1139**	0.1756	0.1977	

- Each result is taken from three different runs of each benchmark
- There are 15 expressions.
  - Neat-6 is the Harmonic series, so it cannot be completely recovered.
  - Neat-7 and Neat-8 require constant tokens to solve but are not in the benchmark set. They may not be solvable.
  - We solve *all* the expression which are known to be solvable.
- Note we use median or mean RMSE as is the custom for each benchmark.

In conclusion

- The method is simple but seems very effective.
  - Simple is better than complicated.
- More analysis is need to determine if our intuition for why it works is in fact what is happening.
- Read our paper, see our poster.
- Download the source at: https://github.com/brendenpetersen/deep-symbolic-optimization


Lawrence Livermore
National Laboratory