# RAG-SR: Retrieval-Augmented Generation for Neural Symbolic Regression

Hengzhe Zhang, Qi Chen, Bing Xue, Wolfgang Banzhaf, Mengjie Zhang

School of Engineering and Computer Science, Victoria University of Wellington
Department of Computer Science and Engineering, Michigan State University

ICLR 2025 (Spotlight)

## Outline

# Introduction

## Symbolic Regression

- **Symbolic Regression (SR)** discovers mathematical expressions that best describe a dataset
- Automatically determines both the **structure** $f$ and **parameters** $\theta$ of the model
- Offers both **high accuracy** and **interpretability**
- Applications: physics, biology, finance, scientific discovery

**Mathematical Definition:**

$$(f^*, \theta^*) = \underset{f \in F, \theta}{\operatorname{argmin}} \sum_{i=1}^{n} L(f(x_i, \theta), y_i) \tag{1}$$

## Problem Formulation: Feature Construction Approach

- Generate a set of symbolic trees/features $\Phi = \{\phi_1, \ldots, \phi_m\}$ from dataset $(X, Y)$
- Enhance interpretable model $M$ (e.g., linear regression)
- Objective: Minimize loss function

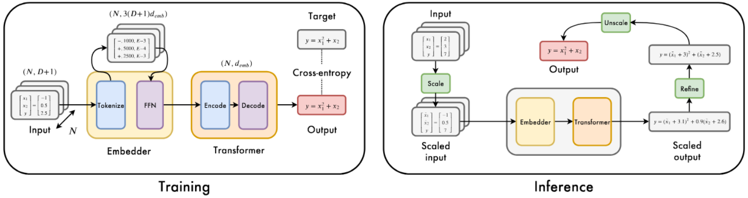
$$L(\Phi; X, Y) = \frac{1}{N} \sum_{i=1}^{N} \ell\left(M\left(\phi_1(X_i), \ldots, \phi_m(X_i)\right), Y_i\right) \tag{2}$$

**Advantages:**

- Effective for complex real-world problems
- Multiple weak features can collectively work well

## Pre-trained Models

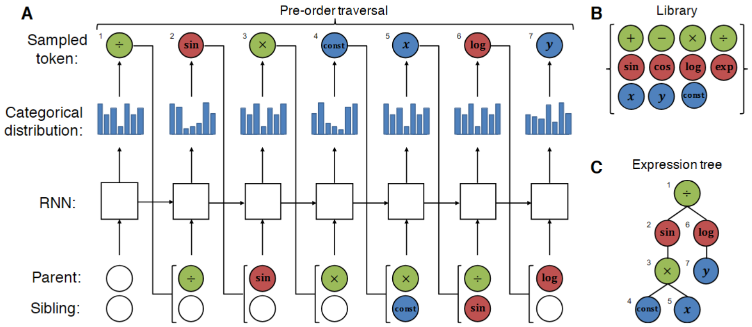
- Limited to seen functions
- Struggles with new variables
- Requires large pre-training



# Limitations of Current Approaches: Reinforcement Learning

## Reinforcement Learning

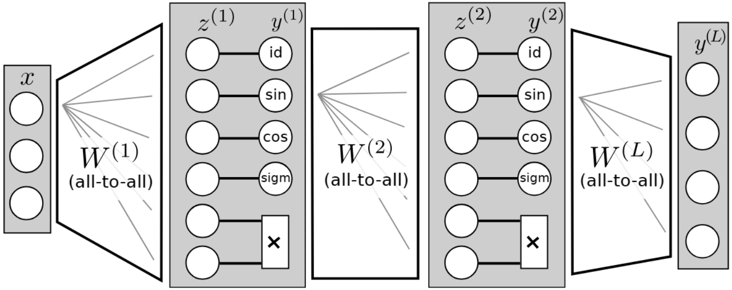
- Low sample efficiency
- Limited feedback
- Slow convergence



## Limitations of Current Approaches: Sparse Supervised Learning

**Sparse Supervised Learning**

- Relies on heuristic pruning
- Produces potentially large solutions
- Requires differentiable activation functions



## Related Work

**Neural Symbolic Regression**

- Pre-trained models (DSR, E2E) struggle with unseen functions/variables
- RL approaches have low sample efficiency
- Sparse supervised learning requires differentiable activation functions

**Evolutionary Symbolic Regression**

- Traditional GP lacks search effectiveness
- Semantic GP is effective but relies solely on existing building blocks
- Limited knowledge retention during evolution

## Key Insight: Guiding Neural Generation with Good Solutions

### Key Insight

**From LM perspective:**

- Using neural networks to directly improve solutions leads to hallucination
- Better approach: Show the model potentially good improvements and ask it to generate better ones

**From EA perspective:**

- Instead of asking ”give me a good mutation,” we show examples of good mutations and say ”a good mutation is like this, give me a better one”

**When using LLMs like ChatGPT:**

- Don’t ask: ”The current program only does B, but I want it to do A”
- Instead ask: ”The current program only does B, I want it to do A, and I think a potential direction is...”

## RAG-SR: A Novel Semantic-to-Symbolic Learning Framework

**Our Approach:** Integrating evolutionary feature construction with neural networks through online learning

- **Semantic Descent:** Adaptively generates symbolic trees aligned with desired semantics in real-time through online supervised learning
- **Retrieval-Augmented Generation:** Mitigates hallucinations by leveraging symbolic expressions in a library
- **Masked Contrastive Learning:** Captures semantically equivalent but syntactically different symbolic expressions
- **Scale-invariant Data Augmentation and Double Query Strategy:** Improves robustness to linear regression

# Methodology

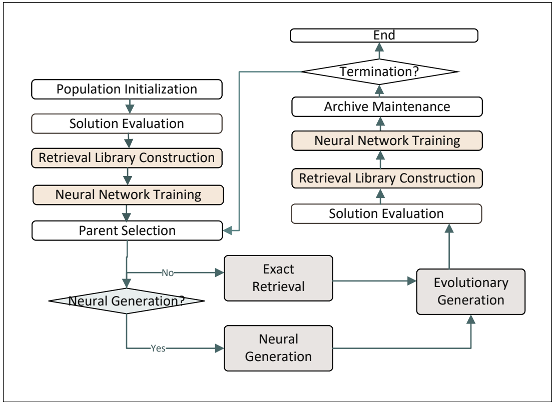
## RAG-SR: Solution Initialization & Evaluation

**Step 1: Solution Initialization**

- Random symbolic trees using ramped half-and-half

**Step 2: Solution Evaluation**

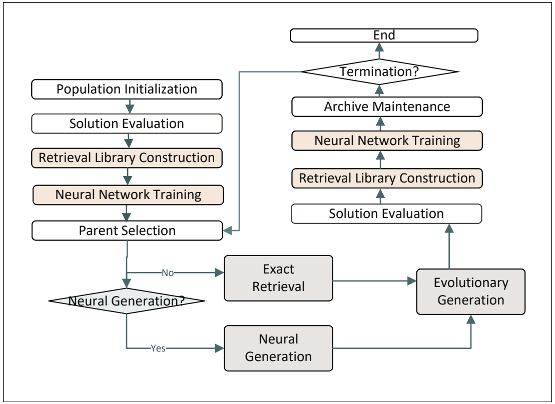
- Features evaluated via ridge regression with LOOCV



## RAG-SR: Semantic Library Construction

**Step 3: Semantic Library Construction**

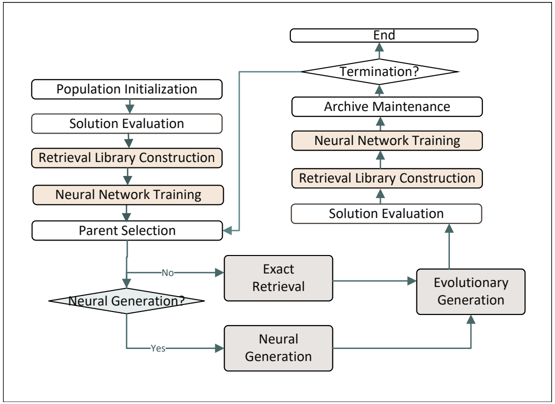
- **Retrieval Library:** FIFO queue with KD-Tree for efficient retrieval
  - Semantics are normalized before storing because feature construction is insensitive to scale
- **Neural Semantic Library:** Trained using pairs from retrieval library



## RAG-SR: Solution Selection Process

**Step 4: Solution Selection**

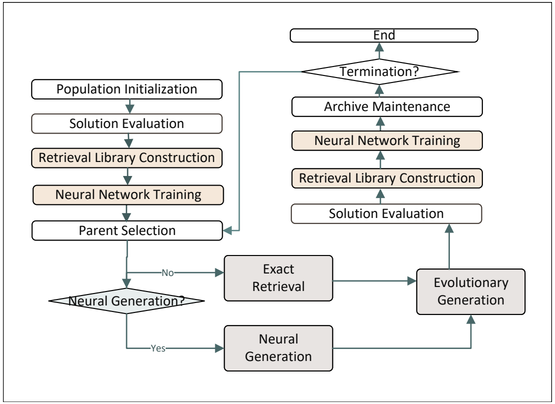
- Lexicase selection to select promising parents



## RAG-SR: Solution Generation

**Step 5: Solution Generation**

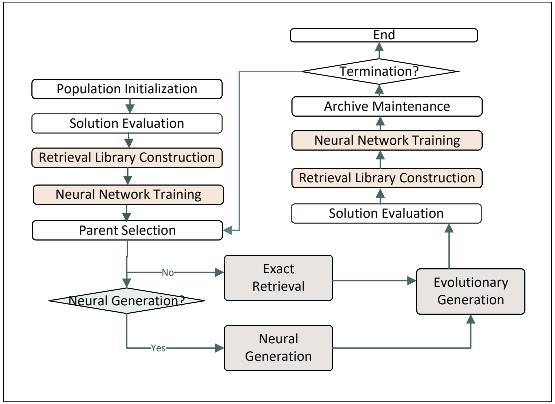
- **Semantic Descent:** Local search to improve solutions
- **Evolutionary Search:** Global exploration with GP operators



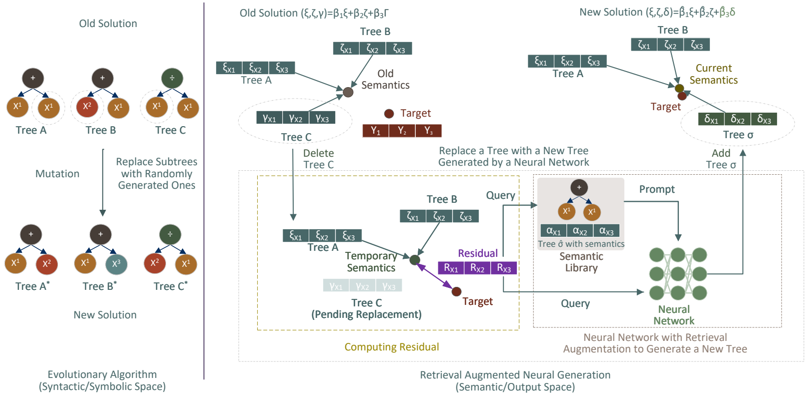
## RAG-SR: Archive Maintenance

### Step 6: Archive Maintenance

- Store best-performing solution



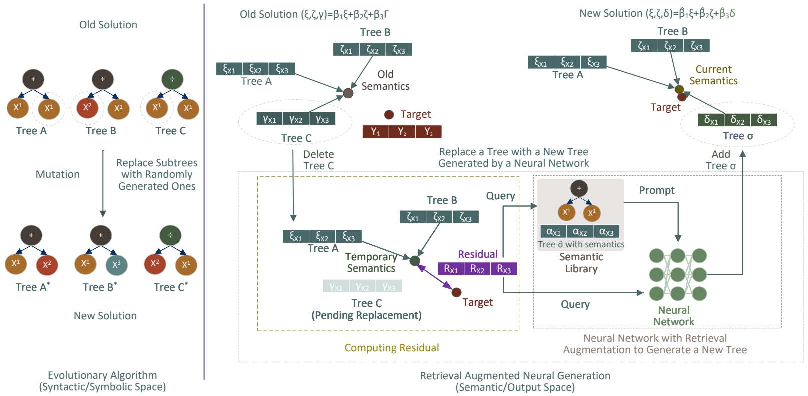
## Semantic Descent: Overview


Old Solution
Tree A
Tree B
Tree C
Mutation
Replace Subtrees with Randomly Generated Ones
Tree A*
Tree B*
Tree C*
New Solution
Evolutionary Algorithm (Syntactic/Symbolic Space)
Old Solution (ξ,ζ,γ)=β₁ξ+β₂ζ+β₃Γ
Old Semantics
Target
Delete Tree C
Replace a Tree with a New Tree Generated by a Neural Network
New Solution (ξ,ζ,δ)=β̂₁ξ+β̂₂ζ+β̂₃δ
Current Semantics
Tree σ
Add Tree σ
Temporary Semantics
Tree C (Pending Replacement)
Residual
Query
Prompt
Tree ô with semantics
Semantic Library
Neural Network
Computing Residual
Neural Network with Retrieval Augmentation to Generate a New Tree
Retrieval Augmented Neural Generation (Semantic/Output Space)

**Key Principle:**

- Iterative optimization replacing suboptimal features while maintaining a constant feature count

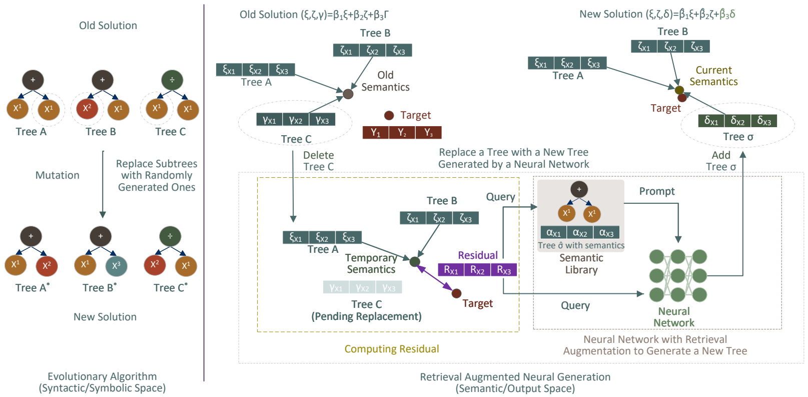
## Semantic Descent: Step 1-2


Old Solution
Tree A
Tree B
Tree C
Mutation
Replace Subtrees with Randomly Generated Ones
Tree A*
Tree B*
Tree C*
New Solution
Evolutionary Algorithm (Syntactic/Symbolic Space)
Old Solution (ξ,ζ,γ)=β₁ξ+β₂ζ+β₃Γ
Old Semantics
Target
Delete Tree C
Replace a Tree with a New Tree Generated by a Neural Network
New Solution (ξ,ζ,δ)=β̂₁ξ+β̂₂ζ+β̂₃δ
Current Semantics
Tree σ
Add Tree σ
Temporary Semantics
Residual
Query
Tree C (Pending Replacement)
Computing Residual
Prompt
Tree ô with semantics
Semantic Library
Neural Network
Neural Network with Retrieval Augmentation to Generate a New Tree
Retrieval Augmented Neural Generation (Semantic/Output Space)

**For each tree $\phi_i$ in solution $\{\phi_1, \dots, \phi_m\}$:**

1. Temporarily remove contribution of $\phi_i$: $\Phi^{temp}(X) = \Phi(X) - \beta_i \phi_i(X)$
2. Compute residual $R = Y - \Phi^{temp}(X)$

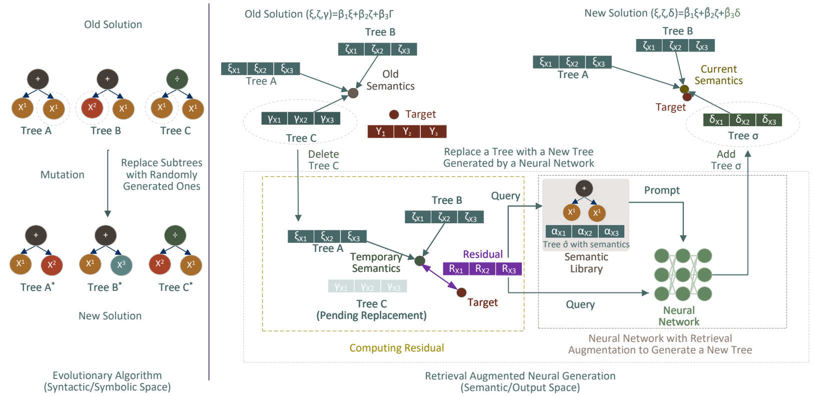
## Semantic Descent: Step 3


Old Solution
Tree A
Tree B
Tree C
Mutation
Replace Subtrees with Randomly Generated Ones
Tree A*
Tree B*
Tree C*
New Solution
Evolutionary Algorithm (Syntactic/Symbolic Space)
Old Solution (ξ,ζ,γ)=β₁ξ+β₂ζ+β₃Γ
Old Semantics
Target
Delete Tree C
Replace a Tree with a New Tree Generated by a Neural Network
New Solution (ξ,ζ,δ)=β₁ξ+β₂ζ+β₃δ
Current Semantics
Tree σ
Add Tree σ
Query
Temporary Semantics
Residual
Tree C (Pending Replacement)
Computing Residual
Prompt
Tree ô with semantics
Semantic Library
Neural Network
Neural Network with Retrieval Augmentation to Generate a New Tree
Retrieval Augmented Neural Generation (Semantic/Output Space)

**Generate a new tree to fill this gap using:**

- Neural generation with probability $P_{neural}$: "Here's a pattern that worked before, generate something better"
- Retrieval from semantic library with probability $1 - P_{neural}$: "Find the best matching pattern we've seen before"

## Semantic Descent: Step 4

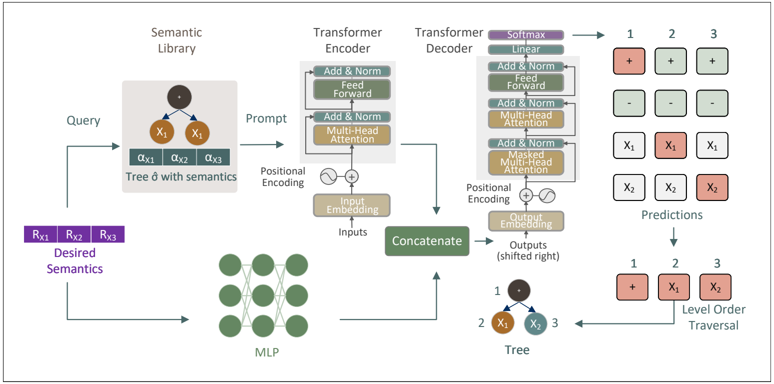


Old Solution
Tree A
Tree B
Tree C
Mutation
Replace Subtrees with Randomly Generated Ones
Tree A*
Tree B*
Tree C*
New Solution
Evolutionary Algorithm (Syntactic/Symbolic Space)
Old Solution (ξ,ζ,γ)=β₁ξ+β₂ζ+β₃Γ
Old Semantics
Target
Delete Tree C
Replace a Tree with a New Tree Generated by a Neural Network
New Solution (ξ,ζ,δ)=β̂₁ξ+β̂₂ζ+β̂₃δ
Current Semantics
Tree σ
Add Tree σ
Query
Temporary Semantics
Residual
Tree C (Pending Replacement)
Prompt
Tree ô with semantics
Semantic Library
Neural Network
Computing Residual
Neural Network with Retrieval Augmentation to Generate a New Tree
Retrieval Augmented Neural Generation (Semantic/Output Space)

**Final Step:**

- Accept new tree if it reduces error for Retrieval, directly accept if Neural generation
- Continue process iteratively for all trees in the solution

# Retrieval-Augmented Generation

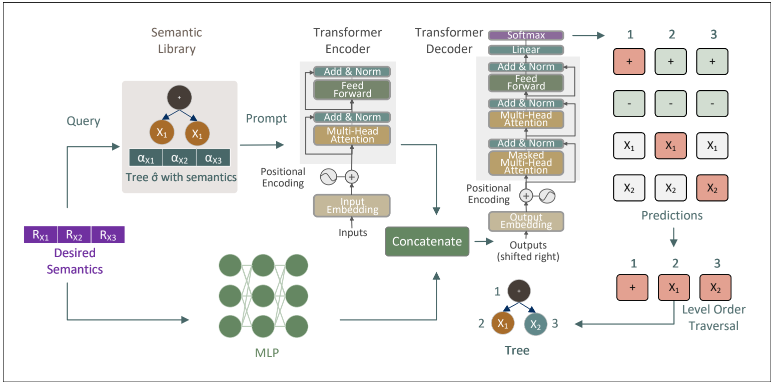


**Challenge:** Language model can always generate something, but may be irrelevant to the given task

**Key Idea:** Retrieval-augmented generation to reduce hallucination

- **Data Collection:** Store evaluated subtrees and semantics in a library
- **Network Training:** Map semantics to symbolic expressions
- **Retrieval:** Provide context from promising expressions

## NEURAL NETWORK ARCHITECTURE



- **MLP with Residual Connections:** Processes desired semantics $R$
- **Transformer Encoder:** Processes nearest symbolic tree $\hat{\phi}$
- **Transformer Decoder:** Generates sequence of tokens for new tree
- **Output:** Valid symbolic tree $\phi$ with semantics aligned to $R$

## Loss Function

**Dual-objective loss function:**

$$L = L_{\text{cross-entropy}} + \lambda \cdot L_{\text{InfoNCE}} \tag{3}$$

**Cross-Entropy Loss:** Traditional objective for sequence generation
**InfoNCE Loss:** Contrastive learning to capture semantic equivalence

## Masked Contrastive Loss

**Objective:** Align embeddings from intention encoding and retrieval-augmented encoding

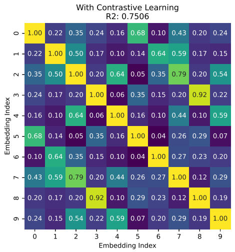
**Process:**

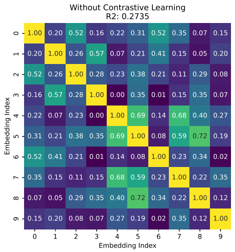
- Nearest semantics $\hat{\phi}(X) \in \mathbb{R}^{B \times N} \rightarrow \text{MLP} \rightarrow \mathbf{F}_{\text{nearest}} \in \mathbb{R}^{B \times K}$
- Symbolic embedding $\mathbf{H}_{\text{Transformer}} \in \mathbb{R}^{B \times L \times D} \rightarrow \text{Average} \rightarrow \mathbf{H}_{\text{avg}} \in \mathbb{R}^{B \times D}$

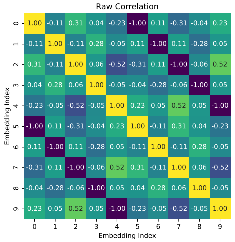
**InfoNCE Loss with Masking:**

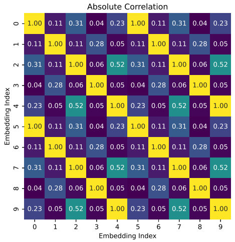
$$L_{\text{InfoNCE}} = -\frac{1}{B} \sum_{i=1}^{B} \log \frac{\exp\left(\text{sim}(\mathbf{F}_{\text{nearest}}[i], \mathbf{H}_{\text{avg}}[i])/\tau\right)}{\sum_{j=1}^{B} \exp\left(\text{sim}(\mathbf{F}_{\text{nearest}}[i], \mathbf{H}_{\text{avg}}[j]) \cdot \text{mask}/\tau\right)} \tag{4}$$

# Effect of Contrastive Learning









**Mask Strategy:**

- Mask matrix eliminates false negatives (cosine similarity > 0.99)
- Prevents penalizing semantically similar expressions
- Improves discrimination between truly different expressions

**Key Findings:**

- Captures different expressions with the same semantics as equivalent
- Without it: different embeddings for equivalent semantics

## Data Augmentation & Double Query

**Sign-invariant Property:** In linear regression, sign of coefficients is automatically adjusted
**Data Augmentation:** Include both the original feature pairs $(\psi, \psi(X))$ and their negations $(\psi, -\psi(X))$

$$T \leftarrow T \cup \{(\psi, -\psi(X)) \mid (\psi, \psi(X)) \in T\} \tag{5}$$

**Double Query:**

- Query the neural network with both the desired semantics $R$ and its negation $-R$
- Generate candidate trees $\phi$ and $\phi'$
- Select tree with highest probability

## Experiments

## Experimental Setup

**Datasets:**

- 120 black-box datasets from PMLB benchmark
- 119 Feynman equations and 14 Strogatz datasets

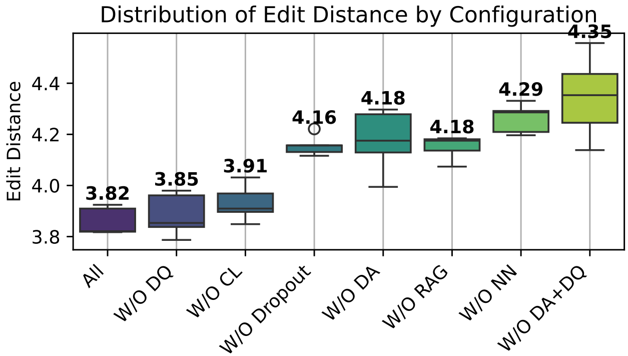
**Evaluation Protocol:**

- 75:25 train-test split
- 10 repetitions for robustness
- $R^2$ score on test set as metric
- Min-max scaling of input features

**Configuration:**

- Population size: 200
- Generations: 100
- Solutions of 10 trees each
- $P_{neural} = 0.1$

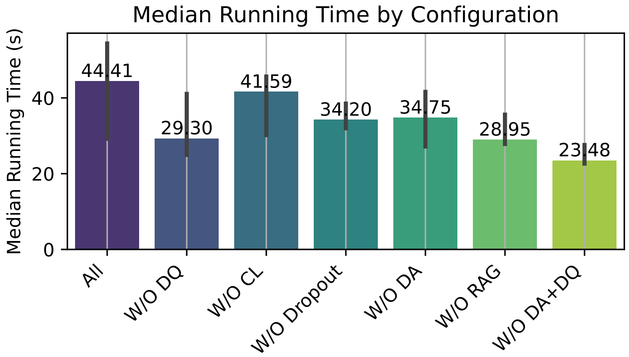
## Component Evaluation: Edit Distance



**Observations:**

- Neural generation outperforms simple retrieval (W/O NN)
- All components together achieve lowest edit distance
- RAG has most significant impact, followed by data augmentation
- Dropout, contrastive learning, and double query show moderate benefits

## Component Evaluation: Running Time



**Observations:**

- RAG moderately increases running time
- Removing DA and DQ reduces time but sacrifices accuracy
- Trade-off between computational cost and performance
- Acceptable overhead given accuracy improvements

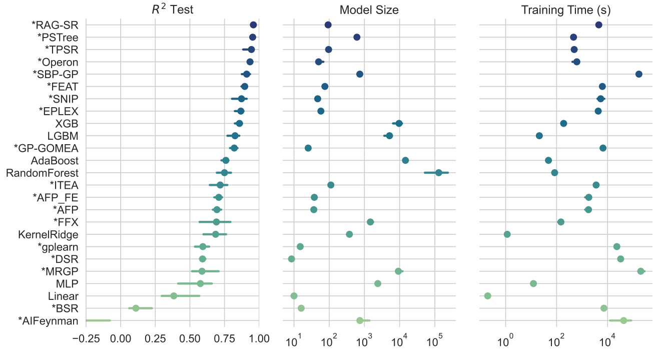
## Generated Trees Examples

| RAG-NN Generated Tree (Distance) | Simple NN Generated Tree (Distance) |
|---|---|
| Sin(Sin(ARG3)) (0) | Cos(Cos(Cos(ARG9))) (4) |
| AQ(ARG7, ARG8) (0) | Log(Max(ARG7, ARG7)) (3) |
| Max(ARG1, ARG8) (0) | Subtract(ARG1, ARG1) (2) |
| Sqrt(Sqrt(ARG2)) (0) | Log(Log(ARG2)) (2) |
| Subtract(ARG6, ARG7) (0) | Max(ARG7, ARG7) (2) |

| Retrieval Tree (Distance) | Ground Truth Tree |
|---|---|
| Sin(ARG3) (1) | Sin(Sin(ARG3)) |
| Log(Neg(Max(AQ(ARG8, ARG0), AQ(ARG7, ARG8)))) (6) | AQ(ARG7, ARG8) |
| Max(add(Abs(Sin(Cos(ARG6))), ARG8), ARG1) (6) | Max(ARG1, ARG8) |
| Square(Log(ARG2)) (2) | Sqrt(Sqrt(ARG2)) |
| Subtract(ARG7, ARG6) (2) | Subtract(ARG6, ARG7) |

**Observations:** RAG significantly improves the neural network’s ability to generate correct expressions

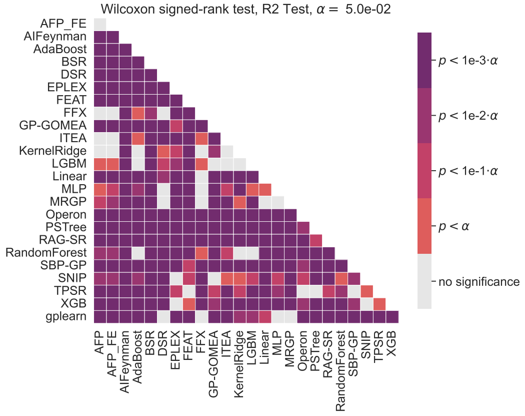
## Performance on SR Benchmark



**Observations:**

- RAG-SR outperforms all 25 algorithms in $R^2$ scores
- Produces models an order of magnitude smaller than SBP-GP
- Training time comparable to FEAT
- Best balance between accuracy, complexity, and efficiency

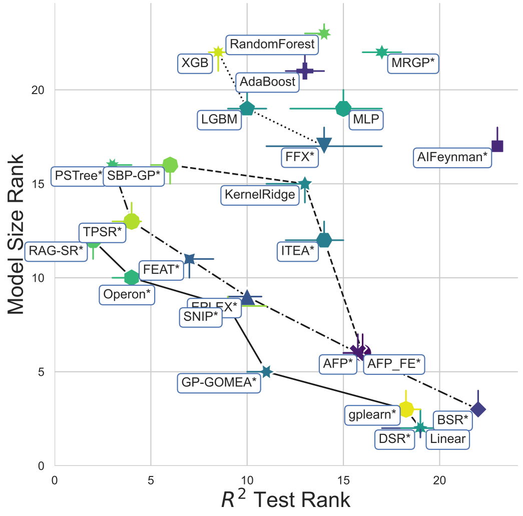
## Statistical Significance



**Observations:**

- RAG-SR significantly outperforms all other methods
- Improvement over TPSR confirms effectiveness of online learning vs. pre-training
- Advantage over SBP-GP shows value of integrating a neural network component

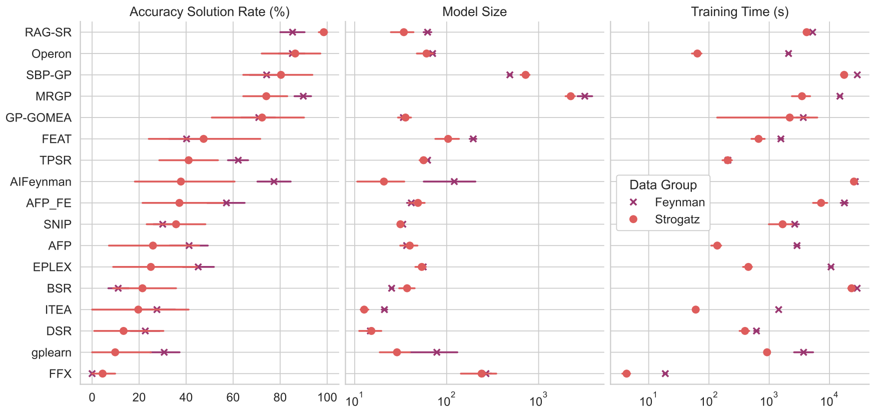
## Pareto Front Analysis



**Observations:**

- RAG-SR appears on the first Pareto front
- Excellent balance between accuracy and model complexity

## Feynman and Strogatz Results



**Observations:**

- Highest test $R^2$ scores on Strogatz datasets
- Second-best on Feynman datasets (after MRGP)
- MRGP models are an order of magnitude larger
- Significantly outperforms deep learning-based SR methods

## Conclusion

## Conclusion

**Contributions:**

- Novel symbolic regression with retrieval-augmented neural semantic library
- Effective online learning approach without pre-training
- Retrieval augmentation that effectively mitigates hallucination
- Contrastive learning that considers semantic equivalence, improving effectiveness
- Data augmentation and double query that leverage sign-invariance, enhancing performance

**Future Directions:**

- Extending to noisy datasets with limited samples
- GPU acceleration for combining pre-training and online learning

## Key Takeaway

Key Innovation of RAG-SR:

Don’t ask the NN to generate good improvements.

Instead, show examples of good improvements and say: ’A good improvement is like this, now give a better one.’

## Thank You!

**Questions?**

Source Code: `https://github.com/hengzhe-zhang/RAG-SR`