EditFlowSR: Revisable Expression Generation for Symbolic Regression

Yuhong Xu* Ziyi Yang* Jinghui Zhong†

School of Computer Science and Engineering, South China University of Technology

*Equal contribution †Corresponding author

NEURAL INFORMATION PROCESSING SYSTEMS

Background

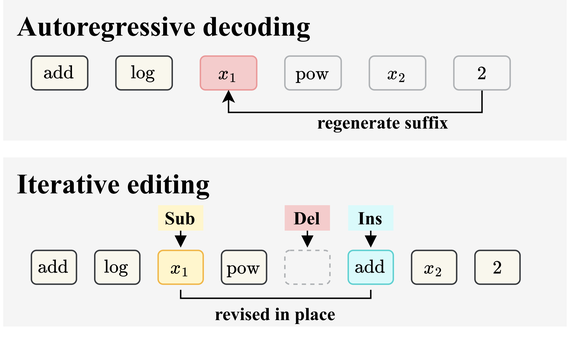
Symbolic regression (SR) aims to discover mathematical expressions that explain observed data. Given samples $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, it seeks an expression f with $y_i \approx f(\mathbf{x}_i)$, often involving a trade-off between **predictive accuracy and expression complexity**.

Finding such an expression requires navigating a vast combinatorial space, where the suitability of each operator or subexpression depends on its role within the entire expression.

Limitations of Prior Methods

1. Search-based methods (GP, tree search, RL) use heuristic strategies to search for equations directly in a discrete symbolic space → **structures are selected only indirectly through fitness and modified by variation operators**.
2. Autoregressive generation emits tokens one at a time in a fixed order → **an early mischosen token conditions all later ones and is fixed only by regenerating the whole suffix**.
3. Structural repair is delegated to external pruning and candidate replacement → **the model never learns to suppress redundancy as part of generation, leaving repair decoupled from generation**.

Editing versus Autoregression

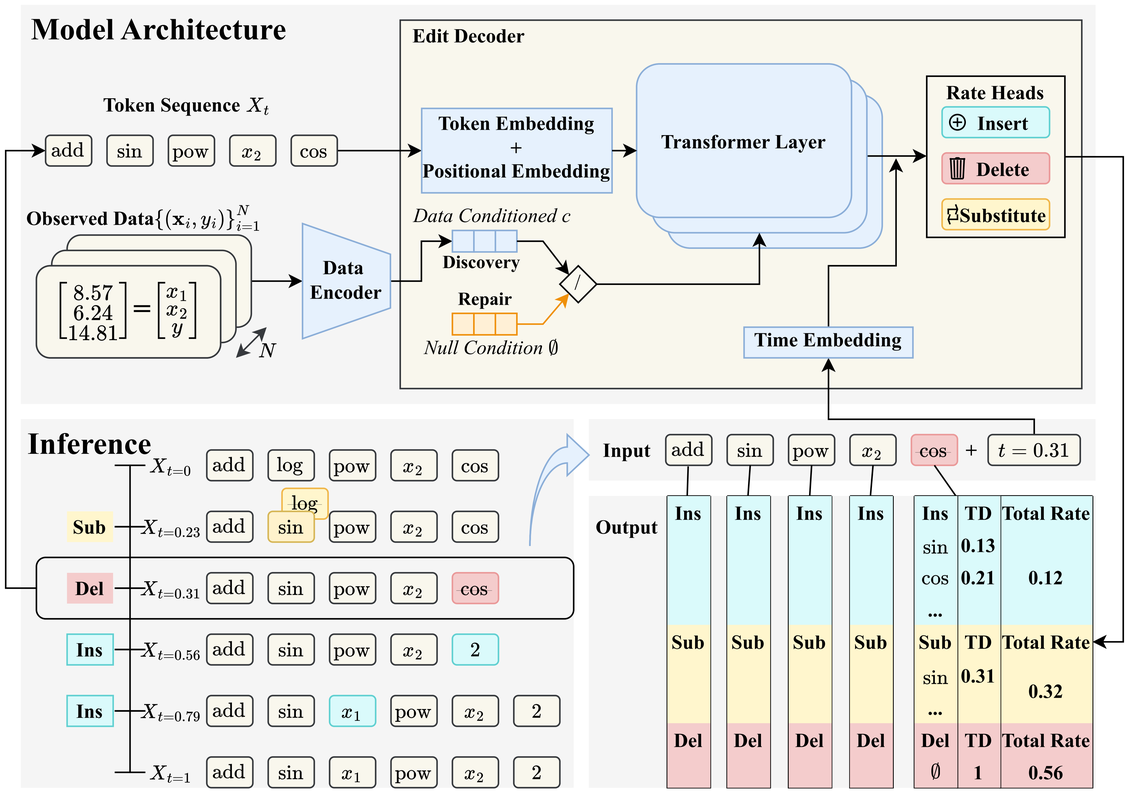


Upper: an early wrong token conditions all later ones and is fixed only by **regenerating its whole suffix**. **Lower:** editing **revises tokens in place** through insertion, deletion, and substitution without touching the rest.

References

[1] Havasi *et al.* Edit flows. NeurIPS 25.
[2] Gat *et al.* Discrete flow matching. NeurIPS 24.
[3] Kamienny *et al.* E2E symbolic regression with transformers. NeurIPS 22.
[4] La Cava *et al.* SRBench. NeurIPS 21 D&B.
[5] Biggio *et al.* Neural symbolic regression that scales. ICML 21.

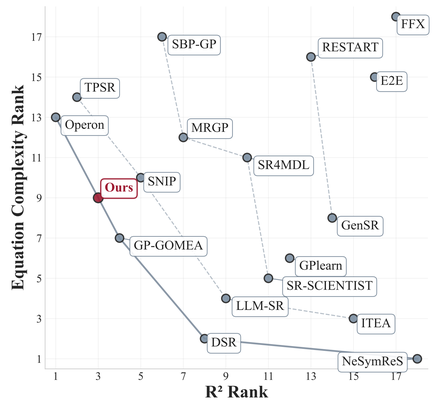
Method: EditFlowSR



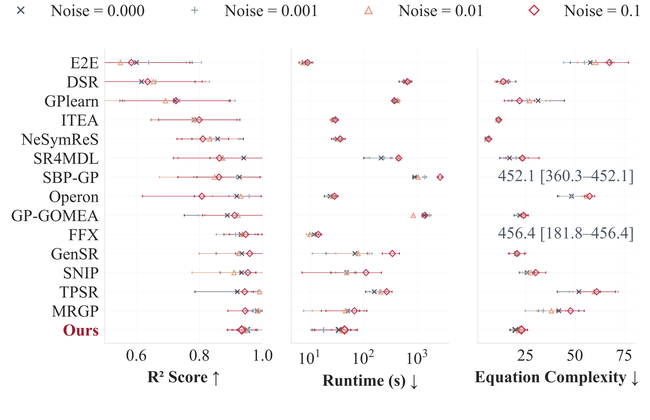
Upper: the model architecture — observed data provide the condition c for the discovery trajectory, while the repair trajectory uses the null condition $\emptyset$; the two trajectories share the editing backbone. **Lower-left:** inference starts from a random sequence X_0 and edits it into the final expression X_1. **Lower-right:** given X_t and t, the model predicts the total rates of insertion, deletion, and substitution, together with token distributions (TD).

Experimental Results

Setup. SRBench: Feynman (119), Strogatz (14), Black-box (89); 22 search-based (incl. LLM-guided), generation-based, and non-symbolic baselines; metrics: test R^2, complexity, inference time.



Pareto (Black-box). First front in accuracy–complexity.



Robustness under noise (Strogatz). Stable across $\sigma \in \{0, 0.001, 0.01, 0.1\}$.

Motivation: Edit as You Go

Existing SR methods do not directly learn how to revise expression structures: search-based methods select structures through fitness, whereas autoregressive generators emit tokens in a fixed order. We formulate equation discovery as **iterative editing of an evolving symbolic sequence**. The editor learns to revise the current sequence in context and, during equation discovery, uses the observed data to guide these revisions, allowing structure to be built and refined within one generation process.

Conditioned on the data condition c, the sequence X, and time t, three heads predict the total rate $\lambda^e_{t,i}$, the intensity of edit e at position i, and the token distribution $Q^e_{t,i}$,

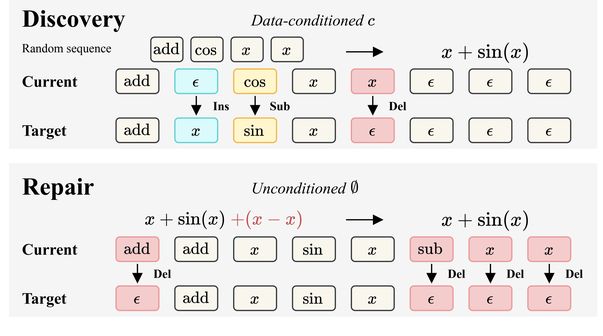
$$u_t(e \mid X) = \lambda^e_{t,i}(X)\, Q^e_{t,i}(a \mid X), \qquad e \in \{\text{ins}, \text{del}, \text{sub}\}.$$

Dual-Trajectory Supervision

We train a shared editor with two trajectories. The **discovery trajectory** recovers equations from random sequences conditioned on observed data, while the **repair trajectory** simplifies redundant expressions under a null condition. Joint training equips the editor for both equation discovery and structural refinement.

- **Edit-path supervision**: pairs of different lengths are aligned to a common length with blank tokens ϵ; every position with $z^i_t \neq z^i_1$ yields **exactly one supervised edit**: $\epsilon \to a$ (ins), $a \to \epsilon$ (del), or $a \to b$ (sub).
- **Repair trajectory**: expands a concise expression into a redundant form (additive zeros, scaled duplicates, squared terms) and trains the model to shorten it back.

$$\mathcal{L}(\theta) = (1-\rho)\,\mathbb{E}_{\text{discovery}}\left[\mathcal{L}_{\text{traj}}(\theta, c)\right] + \rho\,\mathbb{E}_{\text{repair}}\left[\mathcal{L}_{\text{traj}}(\theta, \emptyset)\right]$$



Conclusion

EditFlowSR generates expressions through iterative editing and balances accuracy, runtime, and complexity on standard benchmarks.

Code: github.com/SCUT-EIPI/EditFlowSR