HAD: AUTOREGRESSIVE-CONDITIONED NUMERICAL REFINEMENT FOR MULTIMODAL FORMULA DISCOVERY

Anonymous authors
Paper under double-blind review

ABSTRACT

Multimodal formula discovery systems produce expressions containing both discrete symbols and continuous numerical values. Numerical-literal errors can distort the final equation: a residual post-processor may compensate for scalar mismatch by adding symbolic terms, resulting in a biased equation skeleton. We introduce HAD (Hybrid Autoregressive-Diffusion), a learned refinement framework for formula hypotheses produced by an upstream multimodal model. HAD uses the complete upstream formula as conditioning context. Its autoregressive module generates the symbolic part while retaining the input formula's symbolic structure, and a lightweight diffusion head regenerates the numerical coefficients conditioned on autoregressive context. A two-stage schedule first adapts the autoregressive representation and then trains the numerical head with the backbone frozen. In the VIPER-R1/PHYSSYMBOL setting, HAD obtains an accuracy score of 0.4817, compared with 0.4721 for the VIPER-R1-7B pipeline with SR^2 post-processing. HAD replaces SR^2 with coefficient refinement that preserves the generated formula's symbolic structure. Training-strategy comparisons and qualitative examples show how the staged design supports numerical refinement in this pipeline.

1 INTRODUCTION

Multimodal formula discovery turns visual observations, text, and measurements into mathematical expressions (Liu et al., 2025). Each expression combines discrete choices—operators, variables, and functions—with continuous coefficients such as amplitudes and frequencies. Numerical literals matter even when the symbolic pattern is plausible: a small change in an amplitude or frequency can substantially change a predicted trajectory. Discovering an equation therefore requires accurate values as well as an appropriate symbolic form.

Existing systems can revise a hypothesis by searching for additional symbolic terms. Symbolic Residual Realignment (SR^2) in VIPER-R1 fits an expression to the residual and adds it to the initial formula (Liu et al., 2025); VIPER-R1-7B was the state-of-the-art method on PHYSSYMBOL. Residual expansion addresses missing structure, but a numerical-literal error can also appear as a residual. Adding terms to compensate for that error may produce a biased equation skeleton. This motivates a complementary refinement step that uses the proposed formula as context and predicts its numerical values directly.

We introduce **HAD** (Hybrid Autoregressive-Diffusion) for numerical refinement of complete formula hypotheses produced by an upstream multimodal model. VIPER-R1 first generates a formula containing both symbols and numerical coefficients. HAD replaces SR^2 as the refinement stage and processes this formula as a hybrid symbol–number sequence. Its autoregressive (AR) module generates the symbolic part while preserving the input formula's symbolic structure, and its conditional diffusion head regenerates the coefficient values. The final expression combines the AR-generated symbols with the diffusion-generated coefficients. Training first adapts the AR representation and then optimizes the scalar head with that representation frozen. We evaluate the resulting formulas against references on PHYSSYMBOL.

HAD obtains $S_{\text{struct}} = 0.8710$ and $S_{\text{acc}} = 0.4817$ in this setting, compared with 0.8570 and 0.4721 for VIPER-R1-7B with SR^2 post-processing. The training-strategy comparison favors the staged

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

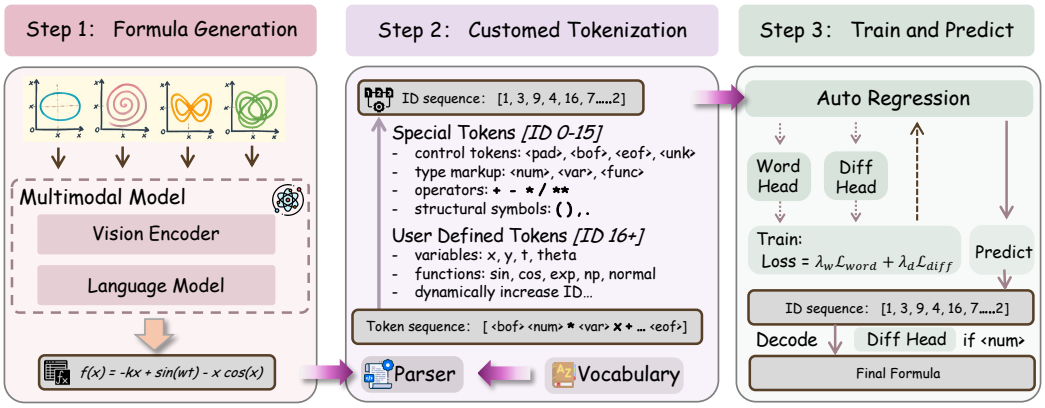


Figure 1: **Overview of the HAD Pipeline.** Step 1: A multimodal model generates an initial formula from scientific inputs. Step 2: HAD reconstructs the formula via dual autoregressive (symbolic) and diffusion (coefficient) pathways. Step 3: The refined formula with precise coefficients is produced.

schedule, and formula examples show numerical values moving toward their references. Together, these results show that a learned refinement step can further improve hypotheses from a strong upstream model, and that the staged training schedule is central to this gain.

Our contributions are:

(1) A hybrid framework for numerical refinement in multimodal formula discovery. We propose HAD (Hybrid Autoregressive-Diffusion), a learned refinement framework for formula hypotheses produced by multimodal models. HAD represents expressions as hybrid symbol–number sequences and uses autoregressive contextual representations to condition a lightweight scalar diffusion head. This design separates symbolic sequence modeling from continuous numerical prediction, allowing numerical values to be modeled directly rather than generated as digit-token sequences.

(2) Staged optimization of formula representations and numerical denoising. We design a two-stage training strategy that first adapts the autoregressive backbone to hybrid formula sequences and then trains the numerical diffusion head with the backbone frozen. This separates representation adaptation from numerical-head optimization and provides a fixed learned context for conditional denoising. Comparisons with direct joint training and an AR-only variant show that the staged configuration achieves higher structural and accuracy scores.

(3) Improved formula-discovery performance through learned numerical refinement. On the PHYSSYMBOL evaluation subset, replacing SR^2 post-processing in the VIPER-R1-7B pipeline with HAD increases the structural score from (0.8570) to (0.8710) and the accuracy score from (0.4721) to (0.4817). Qualitative analyses additionally illustrate corrections to numerical values in generated expressions, supporting learned numerical refinement as a complementary stage in multimodal formula discovery.

2 Related Work

2.1 Numerical Prediction and Constant Fitting

Symbolic regression combines expression search with numerical estimation (Schmidt & Lipson, 2009; Brunton et al., 2016). PySR searches over candidate expressions (Cranmer, 2023); neural methods learn to predict expressions and constants (Petersen et al., 2021; Valipour et al., 2021; Kamienny et al., 2022; Vastl et al., 2024). SNIP learns symbolic–numeric representations and uses BFGS to refine constants within its downstream symbolic-regression procedure (Meidani et al., 2023). HAD places a learned numerical predictor after a multimodal formula generator, where the input is an already proposed expression and the output is a refined formula.

VIPER-R1 uses SR^2 to fit and add a symbolic expression to the residual of an initial hypothesis (Liu et al., 2025). PySR provides the symbolic-regression tool used in that process (Cranmer, 2023). Residual expansion can supply missing symbolic terms; HAD targets numerical positions in a parsed hypothesis. The two operations address different kinds of correction and are assessed separately below.

2.2 AUTOREGRESSIVE–DIFFUSION MODELS

UniGenX combines symbolic and numerical streams using an AR backbone and conditional diffusion for scientific generation (Zhang et al., 2025). HART uses autoregressive generation with a lightweight diffusion module for continuous residual components in visual generation (Tang et al., 2024). HAD brings this combination to formula refinement: its AR context represents an upstream expression, its diffusion head produces scalars, and its staged optimization separates sequence adaptation from numerical denoising.

3 HAD: HYBRID NUMERICAL REFINEMENT

3.1 TASK DEFINITION AND SCOPE

We define the task as conditional numerical refinement. Let F^{up} be the upstream formula produced by the multimodal model. A parser converts it into a hybrid sequence

$$(S, C, M) = \mathrm{Parse}(F^{\mathrm{up}}), \tag{1}$$

where $S = (s_1, \ldots, s_L)$ is the token sequence, $C = (c_i : m_i = 1)$ contains the input coefficient values, and $M = (m_1, \ldots, m_L)$ marks their positions. When $m_i = 0$, s_i belongs to the symbolic part of the formula, including operators, variables, and function names. When $m_i = 1$, $s_i =$ <num> marks a coefficient slot and c_i is the value supplied by VIPER-R1 for that slot.

The refined output is written as

$$\hat{F} = \mathrm{Render}(S, \hat{C}, M). \tag{2}$$

Here $\hat{C}$ contains the coefficients generated by the diffusion head. The symbolic part is generated by HAD's AR module and retains the input formula's symbolic sequence and coefficient-slot layout, represented by S and M. Thus symbolic generation remains part of HAD inference, while the numerical coefficient values are regenerated by diffusion. The renderer combines these two outputs into the refined formula.

The refinement task preserves the input hypothesis's symbolic structure and updates its coefficients. For example, a formula of the form $a\sin(bt)$ retains the multiplication and sine operations and the variable t, while diffusion generates new values for a and b. The output therefore expresses the same symbolic relationship with refined numerical parameters.

3.2 INPUTS AND HYBRID REPRESENTATION

Figure 1 locates HAD between multimodal formula generation and final formula evaluation. VIPER-R1 first supplies its generated expression F^{up}, including its initial coefficients, before SR^2 post-processing. In the HAD pipeline, this expression is passed directly to HAD for refinement. HAD parses this expression into (S, C, M) and constructs the hybrid embeddings below. The AR module processes the input formula, generates its symbolic part with the same structure, and supplies contextual representations to the diffusion head. At coefficient positions, the diffusion head produces the replacement values used in $\hat{F}$. Training pairs $(F^{\mathrm{up}}, F^{\mathrm{GT}})$ provide reference formulas for supervision; at evaluation time, F^{GT} is used to score the output.

This interface matters because the upstream formula carries two kinds of information into refinement. Its symbolic tokens specify the operators and variables that surround each numerical position; its current numerical values provide additional context through the numerical embedding. The complete input formula serves as context for every coefficient prediction, together with the generated output prefix. Thus a leading amplitude can be predicted using the sine term that follows it in the input, and a frequency can be predicted using the surrounding expression and its original numerical values. The

parser and renderer connect those slot-level predictions back to an expression that can be scored as a whole.

At position i, the embedding is

$$\mathbf{e}_i = \begin{cases} f_{\text{tok}}(s_i), & m_i = 0, \\ \mathbf{W}_{\text{num}} c_i + \mathbf{b}_{\text{num}}, & m_i = 1, \end{cases} \tag{3}$$

where $\mathbf{W}_{\text{num}} \in \mathbb{R}^{d \times 1}$ and d is the backbone hidden dimension. Symbol embeddings retain discrete identity, while numerical embeddings expose scalar magnitudes to the AR model. This separates the value from the token identifying its position.

The complete upstream formula supplies conditioning context during training and inference. During teacher-forced Stage I training, ground-truth numerical values are embedded in the output prefix and the next token supplies the prediction label. During inference, the output prefix consists of the decoder's generated symbols and coefficients. Ground-truth formulas supply training supervision and evaluation references.

3.3 Autoregressive Context Encoding

The backbone is a pretrained Qwen model of approximately 0.6B parameters, adapted to the hybrid representation. Let θ denote its parameters, including the numerical embedding, and let $z_{<i}$ denote the hybrid output prefix before position i. Conditioned on the complete upstream formula F^{up}, the state used to predict that position is

$$\mathbf{h}_{i-1} = \text{AR}_\theta(z_{<i} \mid F^{\text{up}}). \tag{4}$$

The state combines the complete input formula, represented by the hybrid embeddings above, with the output prefix. Autoregressive causality applies to the generated output: the full input expression, including symbols and original coefficients after the current slot, is available as conditioning context. The diffusion head receives this formula-conditioned state for each coefficient prediction.

The next-token head models symbolic tokens and numerical placeholders. During Stage I it trains the hybrid sequence representation. During inference, the AR module generates the symbolic sequence while preserving the structure of the input formula, and numerical placeholders identify the coefficient positions handled by the diffusion head. The two pathways therefore supply different parts of the final expression.

3.4 Conditional Numerical Diffusion

For a numerical target, HAD uses a scalar diffusion model conditioned on $\mathbf{h}_{i-1}$. The denoiser is an approximately 2M-parameter, three-layer MLP with adaptive layer normalization (AdaLN) (Peebles & Xie, 2023). Its conditioning combines a timestep embedding and a projection of the AR state:

$$\begin{aligned} \mathbf{u}_{i,k} &= \text{MLP}_{\text{time}}(k) + \text{MLP}_{\text{AR}}(\mathbf{h}_{i-1}), \\ \epsilon_\phi(x_k, k, \mathbf{h}_{i-1}) &= D_\phi(x_k, \mathbf{u}_{i,k}). \end{aligned} \tag{5}$$

Here D_ϕ denotes the denoising MLP. AdaLN modulates its features using this condition. The scalar head acts at coefficient positions selected by M and returns the regenerated values for those slots.

Diffusion provides a conditional sampling model for scalar values. In this design, the AR state supplies formula context and the lightweight denoising head maps noisy scalars toward supervised numerical targets. The surrounding parser, decoder, and evaluation protocol define how these scalar predictions correspond to valid rendered formulas.

We distinguish T_{noise}, the number of training noise levels, from K_{sample}, the number of reverse updates. We use the notation $T_{\text{noise}} = 1000$ and a DDIM sampling setting of $K_{\text{sample}} = 50$ (Song et al., 2020). These denote different quantities in training and inference.

3.5 Inference in the Evaluated Pipeline

VIPER-R1 first generates a complete formula, including its initial numerical coefficients. HAD takes this expression before SR^2 post-processing as input, parses it into (S, C, M), and constructs

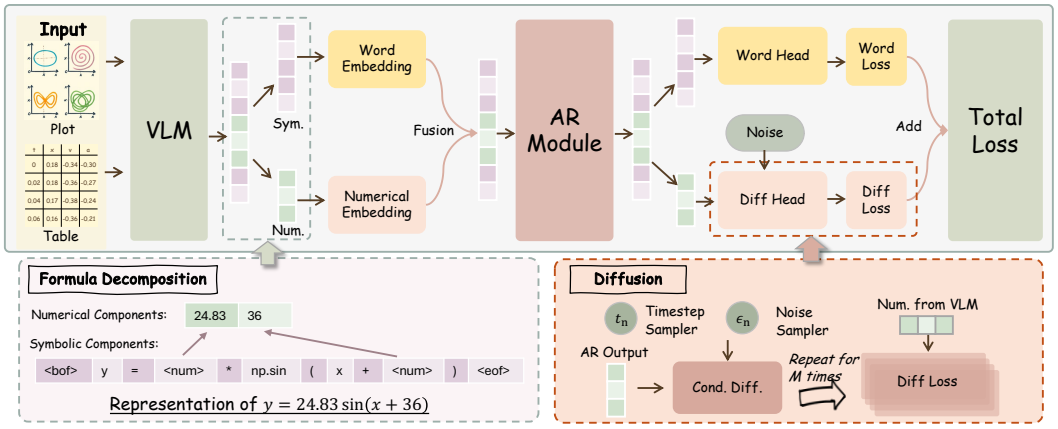


Figure 2: **Training Pipeline of HAD Model.** VIPER-R1 supplies a complete formula; HAD constructs its hybrid symbol–number embeddings. Conditioned on this formula, the autoregressive pathway generates symbols and supplies contextual states to the scalar diffusion head for coefficient prediction. Stage I optimizes the AR pathway with Word Loss while freezing diffusion; Stage II optimizes the diffusion head with Diff Loss while freezing the AR pathway. The schematic's Total Loss label summarizes the two objectives across these stages.

the hybrid embeddings in Eq. 3. The AR module processes the formula and generates its symbolic part, retaining the input symbolic sequence and coefficient-slot layout. At each output position, the AR state is conditioned on the complete input formula and the generated output prefix. This state conditions numerical generation at the corresponding coefficient position.

For each coefficient, the diffusion head initializes a scalar with Gaussian noise and applies $K_{\text{sample}} = 50$ conditional reverse updates using the corresponding AR context. The renderer combines the AR-generated symbolic part with these regenerated coefficients to produce $\hat{F} = \text{Render}(S, \hat{C}, M)$. The resulting expression preserves the input formula's symbolic structure while updating its coefficient values and is then compared with the reference in the PHYSSYMBOL evaluation. Symbolic generation is performed by the AR module throughout this process; diffusion supplies the continuous coefficient values.

4 TWO-STAGE OPTIMIZATION

Numerical denoising depends on an AR state that represents the complete input formula and the hybrid output prefix. Training both pathways from the outset changes that state while the scalar head learns to use it. HAD therefore adapts the sequence representation first and trains the numerical head against a fixed context second. Figures 3 and 4 show the stage-wise training curves; Figure 2 shows the two pathways.

4.1 STAGE I: AUTOREGRESSIVE ADAPTATION

The diffusion parameters are frozen. Stage I updates the trainable AR layers, numerical embedding, and sequence prediction head, while freezing the first eight Transformer layers. Let $\mathcal{P}$ be the set of valid next-token targets, excluding padding, and let $\mathcal{V}$ include symbolic tokens and <num>. With label smoothing $\eta = 0.1$, define

$$q_i(w) = (1 - \eta)\mathbf{1}[w = s_i] + \eta/|\mathcal{V}|. \tag{6}$$

The objective is

$$\mathcal{L}_{\text{AR}} = -\frac{1}{|\mathcal{P}|} \sum_{i \in \mathcal{P}} \sum_{w \in \mathcal{V}} q_i(w) \log p_\theta(w \mid z_{<i}, F^{\text{up}}), \tag{7}$$

where $z_{<i}$ is the teacher-forced hybrid output prefix and F^{up} is the complete upstream formula supplied as context. Numerical values in the output prefix are embedded by Eq. 3; the token objective

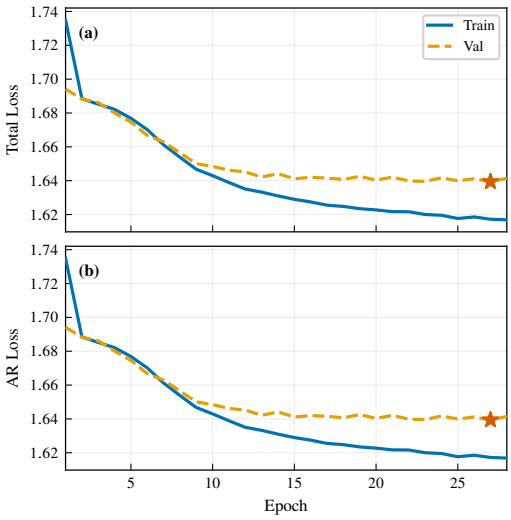


Figure 3: Loss curves for Stage I training, where we freeze the diffusion model and train only the autoregressive model.

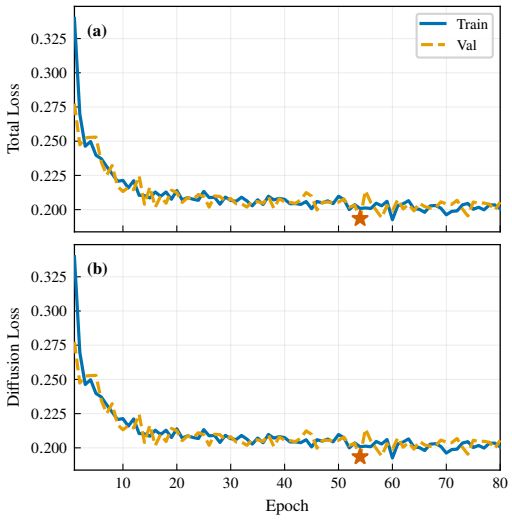


Figure 4: Loss curves for Stage II training, where we freeze the AR model and train only the diffusion model.

learns where numerical placeholders occur, and Stage II supplies the scalar denoising objective. This stage trains a representation for sequence context before optimizing the numerical head.

4.2 Stage II: Numerical-Head Training

The AR module, including the numerical embedding, is frozen. Let $\mathcal{N}$ contain valid numerical target positions, with ground-truth scalar $c_i^\star$ and preceding state $\mathbf{h}_{i-1}$. For a sampled noise level k, the DDPM forward process (Ho et al., 2020) is

$$x_k^{(i)} = \sqrt{\bar{\alpha}_k}\, c_i^\star + \sqrt{1-\bar{\alpha}_k}\, \epsilon, \tag{8}$$

where $\epsilon \sim \mathcal{N}(0,1)$ and $\bar{\alpha}_k = \prod_{j=1}^{k}(1-\beta_j)$. The denoising objective is

$$\mathcal{L}_{\text{Diff}} = \frac{1}{|\mathcal{N}|} \sum_{i \in \mathcal{N}} \mathbb{E}_{k,\epsilon} \left[\left| \epsilon - \epsilon_\phi(x_k^{(i)}, k, \mathbf{h}_{i-1}) \right|^2 \right]. \tag{9}$$

Here k is sampled uniformly from $1, \ldots, T_{\text{noise}}$. Gradients update ϕ while the AR context is held fixed. The scalar targets are trained conditional on the states in Eq. 4, which combine the complete upstream formula with the output prefix and tie numerical denoising to the learned formula representation.

4.3 Implementation Details

We use AdamW (Loshchilov & Hutter, 2017) and batch size 64, with 50 epochs for Stage I and 80 for Stage II. Stage I uses a learning rate of 5×10^{-5} and a cosine schedule. The diffusion noise schedule is linear from 10^{-4} to 0.02 over 1000 levels. Additional settings are given in Appendix A. The schedule optimizes $\mathcal{L}_{\text{AR}}$ and $\mathcal{L}_{\text{Diff}}$ separately.

5 Experiments and Analysis

We evaluate HAD along three dimensions: formula-discovery performance when replacing SR^2, the effect of two-stage optimization, and coefficient refinement in generated expressions. We additionally examine inference runtime and structural changes introduced by residual expansion.

Table 1: Formula-discovery performance on the moderate-difficulty PHYSSYMBOL evaluation subset. VIPER-R1-7B + SR^2 and VIPER-R1-7B + HAD use alternative refinement stages after formula generation. Higher scores are better.

Category	Method	PhysSymbol	
		$S_{\text{struct}} \uparrow$	$S_{\text{acc}} \uparrow$
Closed-source VLMs	GPT-4o	0.2675	0.0549
	GPT-5	0.1252	0.0613
	Claude-Sonnet-4.5	0.2857	0.0204
	Claude-Haiku-4.5	0.2631	0.0406
	Gemini-2.5-Pro	0.0017	0.0067
Open-source VLMs	GLM-4.5V	0.0432	0.0134
	Qwen-VL-Max	0.0108	0.0233
	Qwen2.5-VL-72B-Instruct	0.0727	0.2424
	Qwen3-VL-30B-A3B-Instruct	0.0761	0.0343
	VIPER-R1-7B + SR^2	0.8570	0.4721
Ours	**VIPER-R1-7B + HAD**	**0.8710**	**0.4817**

5.1 EVALUATION SETUP

We evaluate HAD with formula hypotheses generated by VIPER-R1-7B on PHYSSYMBOL (Liu et al., 2025). The base corpus contains 5,000 formula pairs $(F^{\text{up}}, F^{\text{GT}})$, with a $4{,}500/500$ train–validation split. Stage I uses a 50,000-sample augmented collection. Appendix A provides the training configuration.

Table 1 reports Structural Score (S_{struct}) and Accuracy Score (S_{acc}) on the moderate-difficulty PHYSSYMBOL evaluation subset. The numerical tolerance is $\tau = 0.01$. Structural Score measures agreement in expression structure, while Accuracy Score evaluates numerical agreement with the reference. The other VLM rows describe upstream formula generators. The VIPER-R1-7B + SR^2 row reports the baseline pipeline after residual post-processing; the VIPER-R1-7B + HAD row replaces that stage with learned coefficient refinement. Both pipelines are evaluated on their final formula outputs.

5.2 FORMULA DISCOVERY PERFORMANCE

Replacing SR^2 with HAD increases Structural Score from 0.8570 to 0.8710 and Accuracy Score from 0.4721 to 0.4817, corresponding to absolute gains of 0.0140 and 0.0096, respectively (Table 1).

HAD also increases Edit Distance Similarity S_{EDS} from 0.3799 to 0.3831 and reduces Raw Edit Distance S_{RED} from 52.19 to 51.26. These results show improved textual agreement with the reference formulas alongside the gains in structural and numerical accuracy.

The structural-score gain is consistent with the refinement operations of the two pipelines. SR^2 appends a numerical term in the evaluated baseline, whereas HAD updates existing coefficients while preserving the input symbolic structure. HAD therefore avoids the structural changes introduced by residual expansion.

5.3 TRAINING STRATEGY COMPARISON

Two-stage optimization achieves the highest Structural Score and Accuracy Score among the three evaluated training configurations. It obtains scores of 0.871 and 0.482, compared with 0.523 and 0.287 for Direct Joint Training and 0.761 and 0.472 for AR only. Direct Joint updates both pathways from the outset; AR only omits the diffusion pathway. Using the rounded scores in this comparison, Two-Stage improves the two metrics by 0.348 and 0.195 over Direct Joint, and by 0.110 and 0.010 over AR only.

The training curves provide an optimization-level view of this comparison (Figure 6). During the 160-epoch Direct Joint run, training and validation word losses diverge, while diffusion loss converges to approximately 0.25 on both splits and alignment accuracy fluctuates around 0.47. These observations

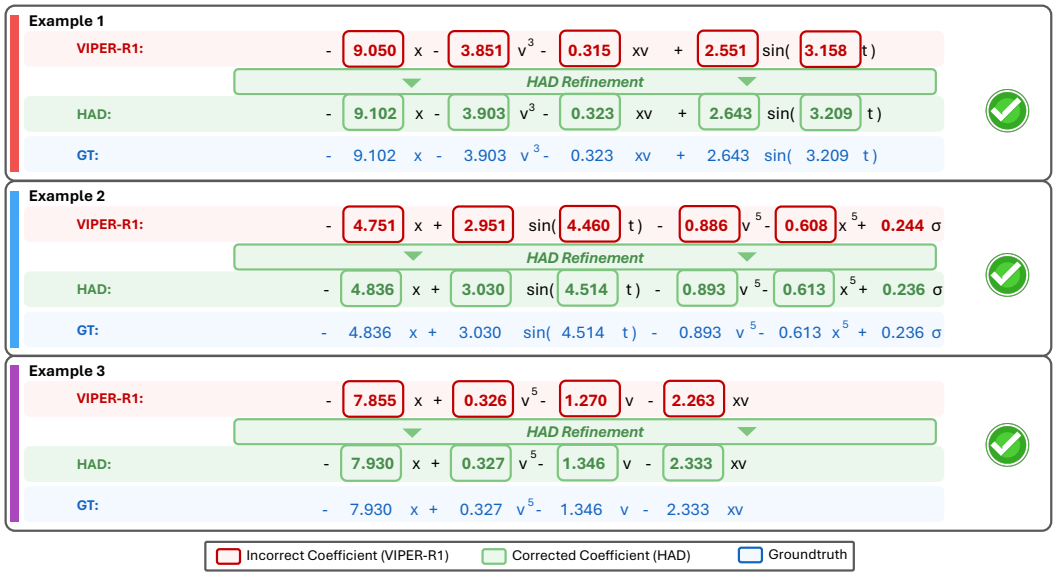


Figure 5: **Qualitative Examples.** Selected formula predictions from VIPER-R1 (input) and HAD (refined output) compared to the ground truth. HAD preserves the symbolic skeleton while correcting coefficient values (highlighted in **bold**).

are consistent with the motivation for adapting the AR representation before optimizing the numerical head.

Together, the formula scores and training dynamics support the staged schedule: Stage I adapts the hybrid sequence representation, and Stage II trains the numerical head against the frozen representation.

5.4 Numerical Refinement and Runtime Analysis

Figure 5 shows coefficient refinement across three expressions while preserving their symbolic structures. In Example 1, the leading coefficient magnitude changes from 9.050 to 9.102 and the sine frequency from 3.158 to 3.209, matching the reference values at the displayed precision. Examples 2 and 3 show corresponding updates across multiple coefficient positions, including polynomial and interaction terms.

On the additional held-out set containing a larger vocabulary and unseen operator combinations, HAD achieves Formula Accuracy of 0.3017 and Structural Score of 0.3943. This evaluation extends the analysis to expressions with unfamiliar symbolic contexts.

Numerical generation with DDIM takes approximately 0.8 seconds per sample, compared with 12.5 seconds for SR^2 in the recorded setup. HAD generates coefficient values through conditional denoising, whereas SR^2 performs symbolic search for a residual expression. The timing comparison is specific to these runs; a controlled speed comparison requires matched hardware, batching, and timing boundaries. Appendix A gives the separate training-time measurements.

5.5 Structural Changes under Residual Expansion

The SR^2 analysis uses 100 held-out PhysSymbol samples, including 96 parseable outputs and 39 strictly comparable cases under SymPy canonicalization (Table 2). Among the strict cases, 32 outputs change skeleton, giving an alteration rate of 82.1%. Across all parseable outputs, the same count gives a conservative lower bound of $32/96 = 33.3\%$ when the remaining cases are treated as unchanged.

The aggregate structural score decreases from 0.826 before SR^2 to 0.647 after it; Table 2 records an absolute drop of -0.180 and a relative decline of -21.8%. All 32 altered cases in the strict

Table 2: Structural changes and structural-score differences following SR^2 post-processing. Measured on 100 PHYSSYMBOL held-out test samples; "strict" denotes samples for which both predicted and ground-truth formulas are canonicalizable by SymPy.

Metric	**Value**
Aggregate structural score	
Mean S_{struct} before SR^2	0.826
Mean S_{struct} after SR^2	0.647
Absolute drop	-0.180 (-21.8%)
Per-sample skeleton alteration	
Altered skeletons (strict, 39 samples)	32 (82.1%)
Altered skeletons (lower bound, 96 samples)	$\geq 33.3\%$
Of altered: cases with S_{struct} decrease	32/32 (100%)
Drop distribution (mean / med. / Q25 / Q75 / max)	0.22 / 0.18 / 0.15 / 0.28 / 0.50

subset show a structural-score decrease. The median drop across these cases is 0.18. These results motivate coefficient refinement that preserves the proposed symbolic structure. This analysis measures structural agreement; physical fit is a separate evaluation dimension.

6 CONCLUSION

HAD connects an upstream multimodal formula hypothesis to a learned numerical-refinement module. Its hybrid representation makes formula context available to a scalar diffusion head, and its two-stage schedule trains that head after adapting the AR backbone. In the VIPER-R1/PHYSSYMBOL setting, replacing SR^2 with HAD in the VIPER-R1-7B pipeline improves the reported structural and accuracy scores; the strategy comparison favors staged optimization, and selected formulas show numerical values moving toward their references.

The results support numerical refinement as a dedicated stage in formula discovery. An upstream model proposes the expression, and HAD uses its full context to update coefficients while preserving the symbolic relationship. Extending this interface to other hypothesis generators and formula distributions is a natural direction for future work.

LIMITATIONS

The experiments primarily involve one dataset and one upstream model, so transfer across symbolic-regression benchmarks and upstream systems remains to be tested. The additional held-out results provide extra evidence for the reported model behavior, while a matched baseline would be needed to quantify transfer in that setting.

HAD refines coefficients within the symbolic structure supplied by VIPER-R1. Missing terms or incorrect operators in that hypothesis require a separate structure-revision step. The quality of the upstream formula therefore determines which errors this refinement process can correct.

The reported score differences are descriptive because repeated-seed uncertainty estimates, confidence intervals, and significance tests are not included. The existing strategy comparison evaluates Direct Joint, AR-only, and Two-Stage configurations; future controlled comparisons can add regression heads and other component substitutions.

Some details of normalization, training-pair alignment, augmentation allocation, and inference configuration require fuller documentation for reproducibility. These details define how future work can reproduce the reported values and compare additional components under matched conditions.

AI USE STATEMENT

Generative AI tools were used to assist with prose editing, manuscript organization, LaTeX formatting, and consistency checks for citations and references. They were not used to generate the reported

experimental data, train models, create figures or tables, or compute the numerical results. The authors reviewed and take responsibility for the final content, including all AI-assisted edits.

Ethics Statement

This work studies numerical refinement for formula discovery on scientific-expression benchmarks. It does not involve human subjects, personal data, user profiling, or deployment in a safety-critical decision system. The main ethical considerations are research integrity, reproducibility, and accurate presentation of the scope of the reported results. The manuscript preserves the reported measurements, distinguishes numerical refinement from residual symbolic expansion, and states the limits of the current evaluation.

Reproducibility Statement

The method is specified through the parsed formula representation in Section 3.1, the hybrid embedding and AR-conditioned diffusion head in Section 3, and the two-stage objectives in Section 4. Section 5 and Appendix A report the evaluated settings, training comparisons, optimization hyperparameters, diffusion schedules, hardware timing, and data descriptions available for the current experiments. Sections 5.4 and 5.5 present the qualitative examples and SR^2 measurements alongside their analyses. Appendices B and C provide numerical-head details and training curves.

References

Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113(15):3932–3937, 2016.

Miles Cranmer. Interpretable machine learning for science with pysr and symbolicregression. jl. *arXiv preprint arXiv:2305.01582*, 2023.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Pierre-Alexandre Kamienny, Stéphane d'Ascoli, Guillaume Lample, and François Charton. End-to-end symbolic regression with transformers. *Advances in Neural Information Processing Systems*, 35:10269–10281, 2022.

Jiaqi Liu, Songning Lai, Pengze Li, Di Yu, Wenjie Zhou, Yiyang Zhou, Peng Xia, Zijun Wang, Xi Chen, Shixiang Tang, et al. Mimicking the physicist's eye: A vlm-centric approach for physics formula discovery. *arXiv preprint arXiv:2508.17380*, 2025.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Kazem Meidani, Parshin Shojaee, Chandan K Reddy, and Amir Barati Farimani. Snip: Bridging mathematical symbolic and numeric realms with unified pre-training. *arXiv preprint arXiv:2310.02227*, 2023.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4195–4205, 2023.

Brenden K. Petersen, Mikel Landajuela, T. Nathan Mundhenk, Claudio P. Santiago, Soo K. Kim, and Joanne T. Kim. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. In *International Conference on Learning Representations (ICLR)*, 2021.

Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, 2009. doi: 10.1126/science.1165893.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

Haotian Tang, Yecheng Wu, Shang Yang, Enze Xie, Junsong Chen, Junyu Chen, Zhuoyang Zhang, Han Cai, Yao Lu, and Song Han. HART: Efficient visual generation with hybrid autoregressive transformer. *arXiv preprint arXiv:2410.10812*, 2024. URL https://arxiv.org/abs/2410.10812.

Mojtaba Valipour, Bowen You, Maysum Panju, and Ali Ghodsi. SymbolicGPT: A generative transformer model for symbolic regression, 2021.

Martin Vastl, Jonáš Kulhánek, Jiří Kubalík, Erik Derner, and Robert Babuška. Symformer: End-to-end symbolic regression using transformer-based architecture. *IEEE Access*, 12:37840–37849, 2024.

Gongbo Zhang, Yanting Li, Renqian Luo, Pipi Hu, Yang Yang, Zeru Zhao, Lingbo Li, Guoqing Liu, Zun Wang, Ran Bi, Kaiyuan Gao, Liya Guo, Yu Xie, Chang Liu, Jia Zhang, Tian Xie, Robert Pinsler, Claudio Zeni, Ziheng Lu, Hongxia Hao, Yingce Xia, Marwin Segler, Maik Riechert, Wei Yang, Hao Jiang, Wen-Bin Zhang, Zhijun Zeng, Yi Zhu, Li Dong, Xiuyuan Hu, Li Yuan, Lei Chen, Haiguang Liu, and Tao Qin. UniGenX: a unified generative foundation model that couples sequence, structure and function to accelerate scientific design across proteins, molecules and materials. *arXiv preprint arXiv:2503.06687*, 2025.

A Data and Training Configuration

The base corpus contains 5,000 VIPER-R1/ground-truth formula pairs with a $90/10$ train–validation split. Stage I uses a 50,000-sample augmented collection.

The backbone is Qwen-0.6B, with approximately 600M parameters. Stage I freezes its first eight Transformer layers and the diffusion head. It uses batch size 64, label smoothing 0.1, AdamW, a learning rate of 5×10^{-5}, and a cosine schedule over 50 epochs; the validation curve reaches a plateau at about 30 epochs. Stage II freezes the AR module and trains the diffusion head for 80 epochs using batch size 64, AdamW moments $(0.9, 0.999)$, weight decay 0.01, gradient clipping at 1.0, and FP16.

The diffusion training schedule uses 1000 noise levels, with a linear β schedule from 10^{-4} to 0.02. DDIM inference uses 50 reverse updates. Training times are approximately 2.5 and 0.8 hours for the two stages on eight A800 GPUs.

B Numerical Head and Objective Details

The numerical head receives a corrupted scalar, a timestep, and an AR context vector. The three-layer MLP uses AdaLN conditioning and contains approximately 2M parameters. Numerical positions account for approximately 15% of the training tokens. Padding and symbolic positions are excluded from the diffusion loss.

Section 3.4 defines the conditioning mechanism, and Section 4.2 specifies the scalar denoising objective and parameter updates.

C Direct Joint Training Dynamics

Figure 6 provides the Direct Joint training curves analyzed in Section 5.3. Alignment accuracy is a training diagnostic distinct from the formula-accuracy score in Table 1.

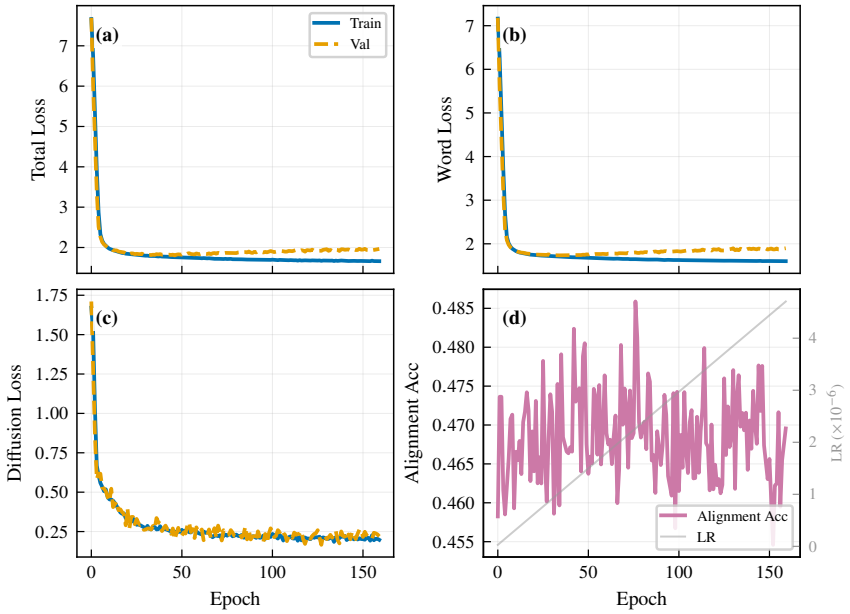


Figure 6: **Ablation Study: Direct Joint Training without Stage I.** The AR and diffusion pathways are trained jointly over 160 epochs, without a separate Stage I adaptation phase. **(a, b)** Total Loss and Word Loss both exhibit training-validation divergence (train ≈1.5, val ≈1.9). **(c)** Diffusion Loss converges to ≈0.25 for both splits. **(d)** Alignment accuracy fluctuates around 0.47 with high variance despite a linear LR warmup.