Feedback Descent: Open-Ended Text Optimization via Pairwise Comparison

Anonymous authors
Paper under double-blind review

Abstract

We introduce *Feedback Descent*, a framework that optimizes text artifacts through structured textual feedback rather than scalar rewards. At each iteration, an evaluator compares the current best artifact against a new candidate, returning both a preference and a textual rationale explaining why. These rationales provide directional information, identifying *what* to change rather than just *which* output is better, widening the information bottleneck inherent in binary preference learning. The loop runs purely at inference time without weight updates and is task-agnostic. We evaluate on visual design, prompt optimization, and molecule discovery, finding that Feedback Descent matches state-of-the-art prompt optimization (GEPA), outperforms reinforcement learning baselines (GRPO, REINVENT), and discovers novel molecules that surpass the 99.9th percentile of over 260,000 compounds across six protein targets.

1 Introduction

Reinforcement learning is a powerful framework for building systems that exceed human capabilities, since it can optimize with respect to feedback on its own outputs, rather than relying on supervised examples of desired outputs. Indeed, recent language models have demonstrated impressive feats in domains like math and programming (OpenAI, 2024; Guo et al., 2025; Chervonyi et al., 2025; Zhu et al., 2024) through a combination of reinforcement learning and text-based reasoning. Unfortunately, existing reinforcement learning frameworks are designed to learn from impoverished supervision signals, typically either scalar rewards or pairwise preference data, where each annotation conveys at most a single bit per pair. These bottlenecks discard information about *why* one behavior is better and *how* to improve, yet this information is available in environment feedback or easily elicited from humans during annotation (Wu et al., 2023; Just et al., 2024).

Our goal is to widen this information bottleneck, i.e., increase the information the system can extract per unit of experience (Silver & Sutton). Collecting more detailed feedback is straightforward, e.g., with brief rationales explaining preferences; the challenge is turning such feedback into measurable improvement. Because free-form feedback does not define a differentiable objective, it cannot directly drive weight updates via backpropagation. Our approach iterates at inference time, using language models to translate accumulated feedback into targeted edits of text artifacts (prompts, code, molecules, JSON configs) that improve a final performance objective, without any weight updates.

To that end, we introduce *Feedback Descent*, a framework for continual optimization in text space. At each iteration, we prompt a language model to propose an improved version of the current best artifact, conditioned on all previous feedback. We compare this candidate against the current best, and the evaluator returns a preference along with textual feedback explaining the choice. If the candidate is preferred, it becomes the new best. Repeating this loop yields semantically local, feedback-aligned improvements that implement gradient-like steps in text space. See Figure 1 for a conceptual illustration. Our contributions are threefold. **First**, we formalize why textual feedback enables dimension-free convergence under idealized assumptions, while zeroth-order methods suffer exponential slowdown with effective dimensionality, identifying when and why structured feedback outperforms scalar

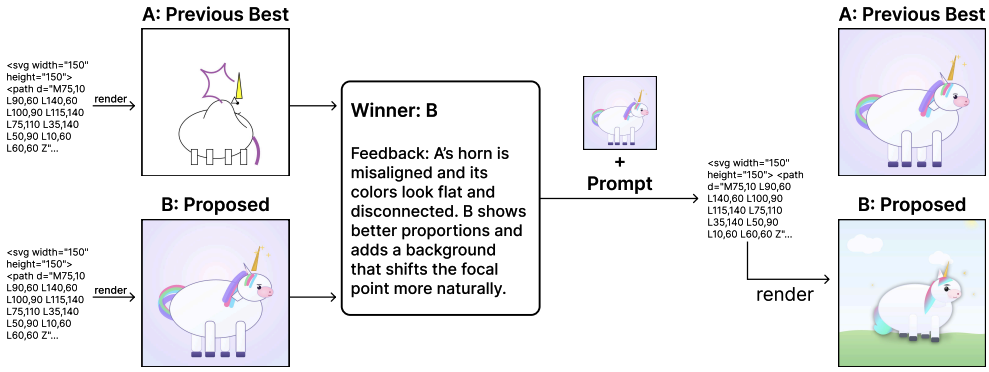

Figure 1: A conceptual illustration of Feedback Descent. At each iteration, we compare the previous best artifact with a new candidate. The evaluator provides both a pairwise preference and textual feedback. Preferences ensure the selection of better candidates, while feedback accumulates directional information that guides semantically meaningful edits. **Unlike scalar rewards that provide only magnitude, textual feedback identifies which aspects to change.**

rewards. **Second**, we demonstrate cross-domain generality: Feedback Descent works across three distinct domains (visual design, prompt optimization, molecule design) with the same iterative loop. **Third**, we validate competitive or superior performance versus specialized methods, achieving competitive results with the state-of-the-art in prompt optimization (GEPA) while outperforming a reinforcement learning baseline (GRPO). In the molecule design experiment, Feedback Descent outperforms specialized molecular optimizers (Graph MCTS/GA, REINVENT) despite operating purely on text representations, discovering molecules that surpass the 99.9th percentile of a 260,000-compound database of molecules.

2 Feedback Descent: Open-Ended Text Optimization

We propose Feedback Descent, a framework for open-ended optimization of text-representable artifacts whose quality is easier to *judge* than to *construct*. Feedback Descent converts comparative textual feedback into directed semantic edits and iterates in a self-improvement loop.

2.1 Problem Setup

Let $\mathcal{S}$ be the space of token sequences, and let $x \in \mathcal{S}$ denote an artifact (e.g., SVG code). Given the current best $x_t^\star \in \mathcal{S}$ and a candidate $x \in \mathcal{S}$, the evaluator returns

$$\mathsf{E}(x, x_t^\star) \to \big(p \in \{0,1\},\ r \in \mathcal{S}\big), \tag{1}$$

where $p = 1$ indicates $x \succ x_t^\star$ and r is textual feedback explaining *why* the winner is better and *how* to improve. We append r_t to a history $\mathcal{R}_t = \{(x_1, r_1), \ldots, (x_t, r_t)\}$ and iterate, keeping track of the current best artifact $x_t^\star$.

2.2 Feedback Descent

Feedback Descent operates as an iterative optimization loop that maintains a single best artifact x_t^* and progressively improves it through feedback-guided mutations and comparative evaluation. Throughout, we use $\mathcal{M}$ to denote the language model used for generating improved candidates.

Initialization and termination. We initialize x_0^* by prompting a language model with the task description alone (e.g., "Generate SVG code for a unicorn"). The algorithm runs for a fixed budget of T iterations or until convergence (defined as no improvement for k consecutive iterations).

Algorithm 1 Feedback Descent

Require: Initial text x_0, Language model $\mathcal{M}$, T
$x^* \leftarrow x_0$, $\mathcal{R} \leftarrow \emptyset$ {Init best & history}
for $t = 1$ **to** T **do**
  $x_t \leftarrow \mathcal{M}(x^*, \mathcal{R})$ {Propose (2)}
  $p_t, r_t \leftarrow \textsc{Compare}(x_t, x^*)$ {Compare (1)}
  $\mathcal{R} \leftarrow \mathcal{R} \cup \{(x_t, r_t)\}$
  if $p_t = 1$ **then**
    $x^* \leftarrow x_t$, $\mathcal{R} \leftarrow \emptyset$ {Update + reset}
  end if
end for
return x^*

Proposing semantic mutations via prompting. The mutation step leverages a language model's in-context learning capabilities. At iteration t, we prompt the model with the current best artifact x_t^* and accumulated feedback $\mathcal{R}_{t-1}$ to generate an improved candidate:

$$x_t = \mathcal{M}\left(x_t^*, \mathcal{R}_{t-1}\right) \tag{2}$$

The prompt instructs the model to address previous critiques while preserving successful elements. The reset-on-success design naturally bounds history length: feedback accumulates only during failed attempts, then clears when improvement occurs.

Selection and update. We compare the new candidate x_t against the current best x_t^* using the evaluator $\mathsf{E}(x_t, x_t^*)$, which returns both a binary preference p_t and textual feedback r_t. In our running SVG example, examples of feedback include "adjust the stroke width", "make sure the legs are connected to the body", and "add a shadow to the unicorn's mane". We add the feedback to the history: $\mathcal{R}_{t+1} = \mathcal{R}_t \cup \{(x_t, r_t)\}$. If $p_t = 1$ (candidate is preferred), we update $x_{t+1}^* = x_t$ and reset the history; otherwise we keep $x_{t+1}^* = x_t^*$ and retain the accumulated feedback for the next iteration. We summarize the overall process in Algorithm 1.

2.3 Analogy to Gradient Descent

The key algorithmic insight is best understood by analogy to the gradient descent algorithm. Just as gradients provide the direction of steepest ascent under local linearity, textual feedback can suggest plausible directions of improvement in semantic space. For our SVG example, if the feedback indicates "needs more defined horn shape," we expect that a small edit to the horn shape that preserves overall structure will likely be an improvement.

Of course, textual feedback is not a literal gradient. It is approximate and occasionally contradictory, and optimization with such feedback does not have convergence guarantees in the same way that gradient descent does. Instead, feedback acts as a heuristic directional cue, offering higher-bandwidth supervision than a binary preference signal or a scalar reward, just as first-order optimization is fundamentally faster than zeroth-order optimization (Nemirovskij & Yudin, 1983; Agarwal et al., 2009; Nesterov & Spokoiny, 2017). We hypothesize that an open-ended optimization loop based on such cues can succeed, supported by prior evidence that language models reliably translate textual instructions into concrete modifications. Examples include generating code changes (Chen et al., 2021; Austin et al., 2021; Nijkamp et al., 2022; Wang et al., 2023b; Roziere et al., 2023; Guo et al., 2024; Lozhkov et al., 2024; CodeGemma Team et al., 2024), following complex multi-step instructions (Ouyang et al., 2022; Wei et al., 2022; Chung et al., 2024; Longpre et al., 2023; Zhang et al., 2026), targeted text modifications (Schick et al., 2022; Du et al., 2022; Madaan et al., 2023; Welleck et al., 2023; Mishra & Nouri, 2023), and decomposing high-level goals into executable action sequences (Schick et al., 2023; Parisi et al., 2022; Yao et al., 2022; Qin et al., 2024; Wang et al., 2023a; Agarwal et al., 2025).

Why directional information helps. Zeroth-order methods that rely only on function evaluations or binary preferences suffer severe dimension-dependent slowdowns: convergence rates degrade exponentially as the search space grows (Nemirovskij & Yudin, 1983; Nesterov & Spokoiny, 2017). In contrast, first-order methods exploit gradient information to achieve

dimension-free convergence under standard assumptions. Textual feedback provides an approximation to such directional information. Even when individual rationales are imperfect, their aggregate message across failures continually refines the direction of improvement. We formalize this intuition in Section A, showing that under idealized assumptions, rationale-guided updates can achieve linear convergence rates independent of effective dimensionality, while zeroth-order baselines scale exponentially worse. These results provide motivation rather than rigorous guarantees for the discrete text domains we study empirically. In Section 4, we show that Feedback Descent indeed produces consistent improvements across tasks, validating that such heuristic directional cues are sufficient to drive open-ended text optimization.

3 Related Work

Learning from feedback. Preference learning methods (Christiano et al., 2017; Ouyang et al., 2022; Rafailov et al., 2023; Azar et al., 2024) compress human reasoning into binary or scalar signals, forgoing rich explanatory content (Wirth et al., 2017). Recent work shows that fine-grained feedback improves reward modeling (Wu et al., 2023; Just et al., 2024); we go further by using textual rationales directly at inference time. Feedback Descent can also be viewed as an evolutionary algorithm (Goldberg, 1989; Holland, 1992) with a language model as the mutation operator. Prior work has shown LLMs can optimize molecules (Wang et al., 2024) and exploit directional feedback from historical traces (Nie et al., 2024); our contribution is operationalizing *directed mutations* via accumulated pairwise feedback.

Inference-time optimization. Several approaches optimize LLM outputs without weight updates, including self-refinement (Madaan et al., 2023; Bai et al., 2022), test-time scaling (Cobbe et al., 2021; Snell et al., 2024; Geiping et al., 2025), and prompt optimization (Zhou et al., 2022; Yang et al., 2023; Pryzant et al., 2023; Khattab et al., 2024; Opsahl-Ong et al., 2024; Agrawal et al., 2025). The closest prior work is TextGrad (Yuksekgonul et al., 2024), which backpropagates textual gradients through computation graphs. A key difference is that TextGrad optimizes pointwise, conditioning only on the latest artifact, whereas Feedback Descent maintains a trajectory-level buffer of comparative feedback. We compare directly with TextGrad in Section 4 and find that trajectory-level feedback is essential for sustained improvement.

4 Experiments

We evaluate Feedback Descent across three diverse domains: visual design, prompt optimization, and molecule discovery. We demonstrate its generality and effectiveness. Our evaluation spans diverse representations and evaluation modalities: **(1) Representation diversity:** SVG (spatial/geometric), prompts (instructional text), molecules (chemical structures). **(2) Evaluation modality:** SVG uses vision-language model comparison, prompt optimization uses dataset-specific accuracy metrics, molecules use computational docking scores. Together, our experiments answer the question: *Can a single optimization framework, with no domain-specific engineering, match or exceed specialized methods purely through structured feedback?*

4.1 Experimental Domains

We describe each evaluation domain and how we obtain pairwise comparisons augmented with textual rationales. **SVG optimization.** Taking inspiration from Bubeck et al. (2023), we evaluate the ability of models to output SVG code for illustrations of unicorns. We use a set of five diverse judge prompts, each preferring a different aesthetic: *ink wash* painting style, *minimalist* design, *realism*, *retro arcade* pixel-art motifs, and *stained glass* artwork. We compare rendered SVGs using `gpt-4o-mini`, which outputs both a binary preference and short textual feedback. To mitigate order bias, we perform two judgments with swapped image orders (A-B and B-A) and declare a winner only if both judgments are consistent. Otherwise, we try again, up to three times, and discard if no consistent winner emerges.
Prompt optimization. We follow the setup of GEPA (Agrawal et al., 2025) across four diverse tasks: multi-hop reasoning (HotpotQA; Yang et al. (2018)), instruction following

Subject	From Scratch	Judge-Aware
City Skyline	100.0 ± 0.0	87.8 ± 19.4
Mushroom	96.7 ± 6.7	98.8 ± 2.4
Robot	100.0 ± 0.0	98.8 ± 2.5
Unicorn	100.0 ± 0.0	92.7 ± 7.5

Table 1: Win rates (averaged across 5 judges) after five iterations comparing Feedback Descent against direct prompting. Full results in Table 7. **Iterative feedback consistently improves SVG designs over direct prompting.**

Method	Qwen3-8B				GPT-4.1-mini			
	HpQA	IFBench	HoVer	PUPA	HpQA	IFBench	HoVer	PUPA
DSPy Default (Khattab et al., 2024)	42.33	36.90	35.33	80.82	38.00	47.79	46.33	78.57
MIPROv2 (Opsahl-Ong et al., 2024)	55.33	36.22	47.33	81.55	58.00	49.15	48.33	83.37
GRPO (Shao et al., 2024)	43.33	35.88	38.67	86.66	—	—	—	—
GEPA (Agrawal et al., 2025)	**62.33**	38.61	52.33	**91.85**	**69.00**	52.72	51.67	**94.47**
Feedback Descent (ours)	60.00	**38.78**	**60.00**	90.90	68.33	**54.59**	**57.67**	85.66

Table 2: Prompt optimization results across multiple benchmarks with matched evaluation budgets. **Feedback Descent consistently matches or outperforms state-of-the-art methods, using only inference-time feedback.**

(IFBench; Pyatkin et al. (2025)), privacy-aware delegation (PUPA; Li et al. (2025)), and retrieval-augmented verification (HoVer; Jiang et al. (2020)). We evaluate on both open-source (Qwen3-8B; Yang et al. (2025)) and proprietary (GPT-4.1-mini) models. For each task, we use the same multi-stage programs from GEPA, where the number of stages differs across datasets, and we jointly optimize the prompts for all stages using Feedback Descent. Optimization is driven by training examples: candidate prompts are updated based on performance on the training set and textual feedback describing which constraints were satisfied or violated. All candidate prompts are scored on validation examples, and the prompt with the highest validation accuracy rate is selected. We report performance on held-out test examples.

Molecule discovery. We evaluate on molecular docking tasks using DOCKSTRING (García-Ortegón et al., 2022) docking scores and drug-likeness (QED). DOCKSTRING provides a realistic drug discovery setting where molecules are evaluated based on their predicted binding affinity to medically relevant targets, rather than relying solely on simple physicochemical properties. We focus on challenging optimization tasks across six protein targets: ADRB1, PGR, PPARA, PPARG, CDK2, and F2. Following DOCKSTRING, we compute the combined score $s = -\text{Vina} - 10 \times (1 - \text{QED})$. We represent molecules as SMILES strings (Weininger, 1988) and evaluate using DOCKSTRING's molecular docking pipeline to compute Vina scores (binding affinity). The feedback system provides rich structured information, including RDKit molecular descriptors (Landrum, 2006), similarity searches against known compounds from molecular databases (Liu et al., 2007; Gilson et al., 2016; Gaulton et al., 2012; Mendez et al., 2019), and detailed docking results. In the system prompt, we also provide the LLM information about the protein target obtained from the UniProt database (The UniProt Consortium, 2023). Together, this provides the LLM with detailed feedback on molecular properties that affect binding affinity, drug-likeness violations, and comparisons to known active compounds.

4.2 SVG Optimization

We evaluate iterative feedback against direct prompting across two generators, `gpt-4o-mini` and `gpt-4o`. The direct prompting baseline receives the full evaluation rubric and is tasked with producing a single best design. Feedback Descent instead begins with an initial set of candidates, and through 5 rounds of structured feedback and improvement, refines designs using judge comparisons that reflect aesthetic criteria. We test two initialization regimes: **Scratch**, which starts from images simply instructed to generate images of unicorns, and

	Method	ADRB1	PGR	PPARA	PPARG	CDK2	F2	Avg
DOCKSTRING (N=260155)	Top 50%	5.305	3.478	4.549	4.210	4.385	4.168	4.349
	Top 90%	8.785	7.878	7.987	7.658	7.733	7.477	7.920
	Top 99%	9.620	8.703	8.718	8.449	8.453	8.139	8.680
	Top 99.9%	10.209	9.260	9.230	9.012	8.979	8.722	9.235
	Top 99.99%	10.742	9.723	9.821	9.518	9.509	9.252	9.761
	Best Molecule	11.330	9.742	9.907	9.529	9.534	9.311	9.892
	Graph MCTS† (Jensen, 2019)	8.883	7.819	7.363	7.134	7.777	6.310	7.548
	SMILES GA (Brown et al., 2019)	9.334	8.335	9.052	8.560	8.268	7.984	8.589
	REINVENT (Olivecrona et al., 2017)	9.867	8.604	8.735	9.054	8.695	8.441	8.899
	Graph GA† (Jensen, 2019)	10.249	8.793	9.211	8.769	8.652	8.900	9.096
	GP-BO† (Tripp et al., 2021)	10.552	9.307	9.680	9.485	9.067	8.686	9.463
	TextGrad (Yuksekgonul et al., 2024)	8.531	8.057	7.953	7.256	8.174	7.357	7.888
	Feedback Descent (ours)	**10.623**	**9.615**	**9.919**	**10.187**	**9.803**	**9.300**	**9.908**

Table 3: Results for molecule optimization on six protein targets. Full results with standard deviations are in Table 9. For each target, the top generative result is in **bold**, and any population in the DOCKSTRING database that exceeds the best generative result is underlined. **Feedback Descent rivals or surpasses specialized molecular optimizers across all six targets.**

Informed, which starts from the strongest direct generations conditioned on the rubric, determined by the LLM judge.

Results. Table 1 shows the win rates after 5 iterations. For both `gpt-4o-mini` and `gpt-4o`, Feedback Descent reliably improves outputs over the initial population. Qualitative examples of an optimization trajectory in **??** and each judge-object pair in Figure 5 show that outputs diverge across judges, aligning with aesthetic criteria such as geometry, minimalism, or retro arcade motifs. This reflects a generator-verifier gap: the judge articulates criteria the generator cannot satisfy in one shot.

4.3 Prompt Optimization

We compare Feedback Descent against four baselines: the default prompt implemented in the DSPy program (Khattab et al., 2024, Default), a Bayesian optimization approach for selecting instructions and demonstrations (Opsahl-Ong et al., 2024, MIPROv2), online reinforcement learning (Shao et al., 2024, GRPO), and a reflective prompt evolution method (Agrawal et al., 2025, GEPA). All baselines are run under matched rollout budgets for fair comparison, and the reported baseline results are from Agrawal et al. (2025).

Each example produces pointwise feedback about which constraints were satisfied or violated. To construct the pairwise feedback for Feedback Descent, we stratify the examples into quadrants based on whether each prompt resulted in a correct response. We then ask the model to propose textual descriptions of inputs where these discrepancies arise. We then statistically validate each hypothesis, filtering for ones that correspond to consistent differences in performance between the prompts. This process distills the true global differences between the two prompts.

Table 2 shows that Feedback Descent is competitive with GEPA across both models, achieving the best performance on IFBench and HoVer, while GEPA leads on HotpotQA and PUPA. Despite GEPA being specifically engineered for prompt optimization with coordinate descent and Pareto frontier maintenance, Feedback Descent achieves competitive performance with a simpler approach: jointly optimizing all prompts at once via automated textual summaries of pairwise performance differences. Notably, Feedback Descent shows the largest gains on tasks with structured output constraints (IFBench, HoVer), where pairwise feedback can precisely identify which constraints are violated, while GEPA leads on tasks requiring broader reasoning (HotpotQA, PUPA).

4.4 Molecule Optimization (DOCKSTRING)

We compare against baselines implemented in the `mol_opt` repository (Gao et al., 2022). Our comparisons include a genetic algorithm (Brown et al., 2019, SMILES GA), reinforcement

Noise	ADRB1	PGR	PPARG
None	**10.62**	**9.62**	**10.19**
25%	9.28	9.14	8.16
50%	10.21	8.92	8.75
100%	6.60	8.39	6.78

Table 4: DOCKSTRING scores for ADRB1, PGR, and PPARG with Feedback Descent at varying feedback noise levels. **Performance degrades gracefully with increasing noise.**

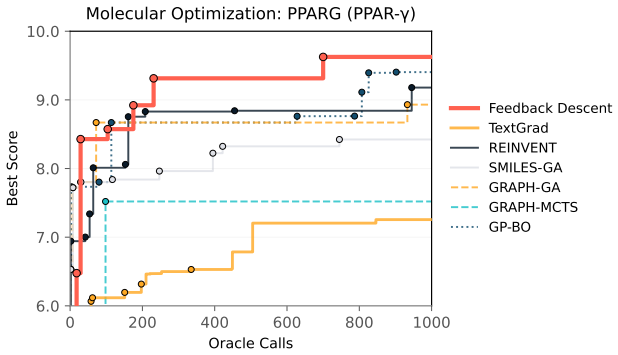

Figure 2: Optimization trajectories on PPARG showing docking scores over oracle calls for Feedback Descent and specialized baselines. **Feedback Descent quickly improves molecular docking scores within the first few hundred oracle calls.**

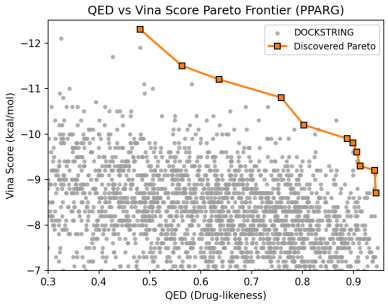

Figure 3: Pareto frontier of docking affinity vs. drug-likeness, comparing Feedback Descent molecules (blue) to the DOCKSTRING database (gray). **Feedback Descent finds novel molecules that meet or surpass known ones.**

learning (Olivecrona et al., 2017, REINVENT), fragment-based algorithms (Jensen, 2019, Graph MCTS/GA), and Bayesian optimization on molecular graphs (Tripp et al., 2021, GP-BO). Because fragment-based methods exploit graph-level structural priors, the most direct comparison is to the text-only baselines: SMILES GA and REINVENT. We also compare our approach with TextGrad (Yuksekgonul et al., 2024), a recent work that similarly utilizes an LLM to make textual updates to SMILES strings. The key difference is that TextGrad's improvement proposals are pointwise, conditioning only on the latest molecule, whereas Feedback Descent conditions on accumulated feedback history, enabling more effective continual improvement at high iteration budgets.

Main Results. Table 3 summarizes optimization outcomes across six protein targets. Feedback Descent outperforms all baselines and achieves the strongest scores on all targets. On multiple proteins, it matches or exceeds the 99.9th and even 99.99th percentiles of the DOCKSTRING database, including surpassing the best molecule present in the database itself ($N = 260155$). TextGrad consistently underperforms Feedback Descent across all targets; while the TextGrad paper evaluated on similar DOCKSTRING tasks for only $\sim$10 iterations, we run both methods for 1000 steps and find that TextGrad's pointwise conditioning does not scale well to high iteration budgets, confirming that accumulated feedback history is essential for sustained improvement. These findings show that Feedback Descent, a purely text-based method, can rival or outperform specialized graph-based algorithms, despite lacking handcrafted structural priors. This is particularly striking because Feedback Descent operates on SMILES strings alone, without access to molecular graphs or fragment libraries that the specialized baselines rely on.

Figure 2 shows optimization trajectories for PPARG. Feedback Descent achieves competitive trajectories relative to specialized methods, reaching high-scoring regions of chemical space with comparable or fewer oracle calls. This pattern holds across targets, suggesting that the method generalizes rather than relying on idiosyncrasies of a single protein system.

Target	Win %
ADRB1	76
CDK2	86
F2	79
PPARG	83
Avg	**81**

Table 5: Feedback alignment: true vs. scrambled feedback win rate across 400 comparisons ($p < 10^{-10}$).

Ablation	Avg. Score
None (Best-of-N)	8.13
FD w/ Random Feedback	7.81
FD w/ Binary Feedback	8.17
FD (ours)	**9.91**

Table 6: Feedback types on DOCKSTRING, averaged across 6 targets. **Detailed feedback matters.**

Analysis of discovered molecules. Figure 3 illustrates the Pareto frontier between docking affinity (Vina score) and drug-likeness (QED) for PPARG. Feedback Descent recovers molecules that sit on or above the DOCKSTRING frontier, indicating that improvements in affinity are achieved without compromising drug-likeness. See Figure 4 in the appendix for the full set of Pareto frontiers across all targets. These results show that feedback-guided search yields candidates that are not only potent but also balanced along multiple drug-relevant dimensions.

Feedback quality and type. We show three ablations on the quality of the feedback in Table 6 (per-target results in Table 8). We evaluate (1) **No Feedback**, i.e., parallel best-of-N sampling with no feedback; (2) **Random Feedback**, which uses the same iterative algorithm but shuffles rationales between molecule pairs; and (3) **Binary Only**, which provides only the binary signal of whether the newest molecule is better than the previous candidate. Random feedback underperforms even best-of-N, confirming that the method relies on feedback content rather than just the iterative structure. The gap between Binary Only and Feedback Descent (1.74 docking score units) demonstrates that textual rationales provide substantial value beyond the binary preference signal. Table 4 further tests robustness to corrupted feedback by randomly shuffling rationales at varying noise levels. The method degrades gracefully: 50% noise still achieves scores above the 99.9th percentile of DOCKSTRING for ADRB1 and PGR.

Feedback alignment. Finally, we test whether generated molecules actually reflect the feedback shown during optimization in Table 5. We present an LLM judge with a newly generated molecule and ask it to determine whether the molecule more closely follows the true feedback or a scrambled control. We randomly flip the order shown to the judge to account for order bias. Across 400 comparisons, true feedback wins 81% of head-to-head comparisons ($p < 10^{-10}$, binomial test), demonstrating that the generator reliably reads and incorporates structured property feedback.

5 Discussion

This paper presents Feedback Descent, an inference-time framework that improves text artifacts through structured pairwise feedback. We validate it on visual design, prompt optimization, and molecule discovery, showing that text can serve as an optimizable medium, not just static data. Unlike parameter tuning, this approach can leverage richer textual signals, allowing for continual improvement without requiring retraining.

Limitations and future work. The method currently operates entirely at inference time. An exciting direction is closing the loop between inference-time and training-time learning. Feedback Descent operates entirely in semantic space, but the feedback-edit pairs it generates could serve as training signal: distilling successful optimization traces into weight updates may yield models that internalize common improvement patterns. Conversely, training evaluators specifically to produce feedback that accelerates optimization (rather than merely accurate preferences) could amplify the method's efficiency. Unifying inference-time feedback with parameter learning could enable richer continual improvement.

References

Alekh Agarwal, Martin J Wainwright, Peter Bartlett, and Pradeep Ravikumar. Information-theoretic lower bounds on the oracle complexity of convex optimization. *Advances in Neural Information Processing Systems*, 22, 2009.

Mayank Agarwal, Ibrahim Abdelaziz, Kinjal Basu, Merve Unuvar, Luis A Lastras, Yara Rizk, and Pavan Kapanipathi. Toolrm: Outcome reward models for tool-calling large language models. *arXiv preprint arXiv:2509.11963*, 2025.

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, et al. Gepa: Reflective prompt evolution can outperform reinforcement learning. *arXiv preprint arXiv:2507.19457*, 2025.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. A general theoretical paradigm to understand learning from human preferences. pp. 4447–4455, 2024.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

Nathan Brown, Marco Fiscato, Marwin HS Segler, and Alain C Vaucher. Guacamol: Benchmarking models for de novo molecular design. *Journal of Chemical Information and Modeling*, 59(3):1096–1108, 2019. doi: 10.1021/acs.jcim.8b00839.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé, Jared Kaplan, Harrison Edwards, Yura Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mo Bavarian, Clemens Winter, Phil Tillet, Felipe Petroski Such, David W. Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William H. Guss, Alex Nichol, Igor Babuschkin, Suchir Balaji, Shantanu Jain, Andrew Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew M. Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *ArXiv*, abs/2107.03374, 2021.

Yuri Chervonyi, Trieu H Trinh, Miroslav Olšák, Xiaomeng Yang, Hoang H Nguyen, Marcelo Menegali, Junehyuk Jung, Junsu Kim, Vikas Verma, Quoc V Le, et al. Gold-medalist performance in solving olympiad geometry with alphageometry2. *Journal of Machine Learning Research*, 26(241):1–39, 2025.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in Neural Information Processing Systems*, 30, 2017.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

CodeGemma Team, Heri Zhao, et al. Codegemma: Open code models based on gemma. *arXiv preprint arXiv:2406.11409*, 2024.

Wanyu Du, Zae Myung Kim, Vipul Raheja, Dhruv Kumar, and Dongyeop Kang. Read, revise, repeat: A system demonstration for human-in-the-loop iterative text revision. *arXiv preprint arXiv:2204.03685*, 2022.

Wenhao Gao, Tianfan Fu, Jimeng Sun, and Connor Coley. Sample efficiency matters: a benchmark for practical molecular optimization. *Advances in neural information processing systems*, 35:21342–21357, 2022.

Miguel García-Ortegón, Gregor N. C. Simm, Austin J. Tripp, José Miguel Hernández-Lobato, Matthias R. Bauer, and Sergio Bacallado. Dockstring: Easy molecular docking yields better benchmarks for ligand design. *Journal of Chemical Information and Modeling*, 62 (15):3486–3502, 2022. doi: 10.1021/acs.jcim.1c01334.

Anna Gaulton, Louisa J Bellis, A Patricia Bento, Jon Chambers, Mark Davies, Anne Hersey, Yvonne Light, Shaun McGlinchey, David Michalovich, Bissan Al-Lazikani, et al. Chembl: a large-scale bioactivity database for drug discovery. *Nucleic Acids Research*, 40(D1): D1100–D1107, 2012. doi: 10.1093/nar/gkr777.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. *arXiv preprint arXiv:2502.05171*, 2025.

Michael K Gilson, Tiqing Liu, Michael Baitaluk, George Nicola, Linda Hwang, and Jenny Chong. Bindingdb in 2015: a public database for medicinal chemistry, computational chemistry and systems pharmacology. *Nucleic Acids Research*, 44(D1):D1045–D1053, 2016. doi: 10.1093/nar/gkv1072.

David E Goldberg. Genetic algorithms in search, optimization, and machine learning. *Addion wesley*, 1989(102):36, 1989.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Yu Wu, YK Li, et al. Deepseek-coder: When the large language model meets programming – the rise of code intelligence. *arXiv preprint arXiv:2401.14196*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. URL `https://arxiv.org/abs/2501.12948`.

John H Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

Jan H Jensen. A graph-based genetic algorithm and generative model/monte carlo tree search for the exploration of chemical space. *Chemical Science*, 10(12):3567–3572, 2019. doi: 10.1039/C8SC05372C.

Yichen Jiang, Shikha Bordia, Zheng Zhong, Charles Dognin, Maneesh Singh, and Mohit Bansal. Hover: A dataset for many-hop fact extraction and claim verification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 3441–3460, 2020. doi: 10.18653/v1/2020.findings-emnlp.309.

Hoang Anh Just, Ming Jin, Anit Sahu, Huy Phan, and Ruoxi Jia. Data-centric human preference optimization with rationales. *arXiv preprint arXiv:2407.14477*, 2024.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Saiful Haq, Ashutosh Sharma, Thomas T Joshi, Hanna Moazam, Heather Miller, et al. Dspy: Compiling declarative language model calls into state-of-the-art pipelines. 2024.

Greg Landrum. Rdkit: Open-source cheminformatics, 2006. URL `http://www.rdkit.org`.

Siyan Li, Vethavikashini Chithrra Raghuram, Omar Khattab, Julia Hirschberg, and Zhou Yu. Papillon: Privacy preservation from internet-based and local language model ensembles. In *Proceedings of the 2025 Conference of the North American Chapter of the Association for Computational Linguistics*, 2025. NAACL 2025.

Tiqing Liu, Yuhmei Lin, Xin Wen, Robert N Jorissen, and Michael K Gilson. Bindingdb: a web-accessible database of experimentally determined protein–ligand binding affinities. *Nucleic Acids Research*, 35(suppl_1):D198–D201, 2007. doi: 10.1093/nar/gkl999.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. The flan collection: Designing data and methods for effective instruction tuning. pp. 22631–22648, 2023.

Anton Lozhkov, Raymond Li, Loubna Ben Allal, Federico Cassano, Joel Lamy-Poirier, Nouamane Tazi, Ao Tang, Dmytro Pykhtar, Jiawei Liu, Yuxiang Wei, et al. Starcoder 2 and the stack v2: The next generation. *arXiv preprint arXiv:2402.19173*, 2024.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36: 46534–46594, 2023.

David Mendez, Anna Gaulton, A Patrícia Bento, Jon Chambers, Marleen De Veij, Eloy Félix, María Paula Magariños, Juan F Mosquera, Prudence Mutowo, Michał Nowotka, et al. Chembl: towards direct deposition of bioassay data. *Nucleic Acids Research*, 47 (D1):D930–D940, 2019. doi: 10.1093/nar/gky1075.

Swaroop Mishra and Elnaz Nouri. Help me think: A simple prompting strategy for non-experts to create customized content with models. pp. 11834–11890, 2023.

Arkadij Semenovič Nemirovskij and David Borisovich Yudin. *Problem Complexity and Method Efficiency in Optimization*. Wiley-Interscience, New York, 1983.

Yurii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2):527–566, 2017. doi: 10.1007/s10208-015-9296-2.

Allen Nie, Ching-An Cheng, Andrey Kolobov, and Adith Swaminathan. The importance of directional feedback for llm-based optimizers. *arXiv preprint arXiv:2405.16434*, 2024.

Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. Codegen: An open large language model for code with multi-turn program synthesis. *arXiv preprint arXiv:2203.13474*, 2022.

Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. Molecular de novo design through deep reinforcement learning. *Journal of Cheminformatics*, 9(1):1–14, 2017. doi: 10.1186/s13321-017-0235-x.

OpenAI. Openai o1 system card. Technical report, OpenAI, 2024. URL `https://arxiv.org/abs/2412.16720`.

Krista Opsahl-Ong, Michael J Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing instructions and demonstrations for multi-stage language model programs. *arXiv preprint arXiv:2406.11695*, 2024.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

Aaron Parisi, Yao Zhao, and Noah Fiedel. Talm: Tool augmented language models. *arXiv preprint arXiv:2205.12255*, 2022.

Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. *arXiv preprint arXiv:2305.03495*, 2023.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hannaneh Hajishirzi. Generalizing verifiable instruction following. *arXiv preprint arXiv:2507.02833*, 2025.

Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Xuanhe Zhou, Yufei Huang, Chaojun Xiao, et al. Tool learning with foundation models. *ACM Computing Surveys*, 57(4):1–40, 2024.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*, 2023.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2023.

Timo Schick, Jane Dwivedi-Yu, Zhengbao Jiang, Fabio Petroni, Patrick Lewis, Gautier Izacard, Qingfei You, Christoforos Nalmpantis, Edouard Grave, and Sebastian Riedel. Peer: A collaborative language model. *arXiv preprint arXiv:2208.11663*, 2022.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *NeurIPS*, 2023.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

David Silver and Richard S. Sutton. Welcome to the era of experience.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

The UniProt Consortium. Uniprot: the universal protein knowledgebase in 2023. *Nucleic Acids Research*, 51(D1):D523–D531, 2023. doi: 10.1093/nar/gkac1052.

Austin Tripp, Gregor NC Simm, and José Miguel Hernández-Lobato. A fresh look at de novo molecular design benchmarks. In *NeurIPS 2021 AI for Science Workshop*, 2021. URL `https://openreview.net/forum?id=gS3XMun4cl_`.

Haorui Wang, Marta Skreta, Cher-Tian Ser, Wenhao Gao, Lingkai Kong, Felix Strieth-Kalthoff, Chenru Duan, Yuchen Zhuang, Yue Yu, Yanqiao Zhu, et al. Efficient evolutionary search over chemical space with large language models. *arXiv preprint arXiv:2406.16976*, 2024.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *arXiv preprint arXiv:2308.11432*, 2023a.

Yue Wang, Hung Le, Akhilesh Gotmare, Nghi Bui, Junnan Li, and Steven Hoi. Codet5+: Open code large language models for code understanding and generation. *Proceedings of EMNLP*, 2023b.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. *ICLR*, 2022.

David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of Chemical Information and Computer Sciences*, 28(1):31–36, 1988. doi: 10.1021/ci00057a005.

Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. Generating sequences by learning to self-correct. *ICLR*, 2023.

Christian Wirth, Riad Akrour, Gerhard Neumann, and Johannes Fürnkranz. A survey of preference-based reinforcement learning methods. *Journal of Machine Learning Research*, 18(136):1–46, 2017.

Zeqiu Wu, Yushi Hu, Weijia Shi, Nouha Dziri, Alane Suhr, Prithviraj Ammanabrolu, Noah A Smith, Mari Ostendorf, and Hannaneh Hajishirzi. Fine-grained human feedback gives better rewards for language model training. *Advances in Neural Information Processing Systems*, 36:59008–59033, 2023.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. *arXiv preprint arXiv:2309.03409*, 2023.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380. Association for Computational Linguistics, 2018.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *ICLR*, 2022.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. Textgrad: Automatic "differentiation" via text. *arXiv preprint arXiv:2406.07496*, 2024.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Guoyin Wang, et al. Instruction tuning for large language models: A survey. *ACM Computing Surveys*, 58(7):1–36, 2026.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. 2022.

Qihao Zhu, Daya Guo, Zhihong Shao, Dejian Yang, Peiyi Wang, Runxin Xu, Y Wu, Yukun Li, Huazuo Gao, Shirong Ma, et al. Deepseek-coder-v2: Breaking the barrier of closed-source models in code intelligence. *arXiv preprint arXiv:2406.11931*, 2024.

A Formal Statements and Proofs

Proposition 1 (Linear convergence under PL with rationale-guided directions)**.** *Let $r : Z \to \mathbb{R}$ be L-smooth and satisfy the μ-PL condition (for maximization)*

$$\frac{1}{2}\|\nabla r(z)\|_2^2 \;\geq\; \mu\big(r(z^\star) - r(z)\big) \qquad \forall z \in Z.$$

At iteration t, suppose a direction v_t satisfies

$$\mathbb{E}[\, v_t \mid z_t \,] \;=\; \alpha\, \nabla r(z_t), \qquad \mathbb{E}\big[\, \|v_t - \mathbb{E}[v_t \mid z_t]\|_2^2 \,\big|\, z_t\big] \;\leq\; \sigma^2 \|\nabla r(z_t)\|_2^2,$$

with constants $\alpha > 0$ and $\sigma \geq 0$, and define $\kappa_1 \triangleq \alpha^2 + \sigma^2$. Consider the update $z_{t+1} = z_t + \eta v_t$, assuming the iterate remains in Z (no projection needed). With stepsize $\eta = \alpha/(L\kappa_1)$,

$$\mathbb{E}\big[r(z^\star) - r(z_{t+1}) \,\big|\, z_t\big] \;\leq\; \Big(1 - \tfrac{\mu\,\alpha^2}{L\,\kappa_1}\Big)\,\big[r(z^\star) - r(z_t)\big].$$

Unrolling yields

$$\mathbb{E}[r(z^\star) - r(z_T)] \;\leq\; \Big(1 - \tfrac{\mu\,\alpha^2}{L\,\kappa_1}\Big)^T \big[r(z^\star) - r(z_0)\big],$$

so ϵ-accuracy is achieved in

$$T \;=\; O\!\left(\frac{L(\alpha^2 + \sigma^2)}{\mu\,\alpha^2}\,\log\frac{1}{\epsilon}\right)$$

iterations.

Proof. L-smoothness gives the lower bound

$$r(z_t + \eta v_t) \;\geq\; r(z_t) + \eta\langle \nabla r(z_t), v_t\rangle - \tfrac{L}{2}\eta^2\|v_t\|_2^2.$$

Taking conditional expectation and using $\mathbb{E}[v_t|z_t] = \alpha\nabla r(z_t)$ and $\mathbb{E}\big[\|v_t\|_2^2 \,\big|\, z_t\big] \leq (\alpha^2 + \sigma^2)\,\|\nabla r(z_t)\|_2^2 = \kappa_1\|\nabla r(z_t)\|_2^2$,

$$\mathbb{E}[r(z_{t+1}) \mid z_t] \;\geq\; r(z_t) + \Big(\eta\alpha - \tfrac{L}{2}\eta^2\kappa_1\Big)\,\|\nabla r(z_t)\|_2^2.$$

By the PL inequality, $\|\nabla r(z_t)\|_2^2 \geq 2\mu\,[r(z^\star) - r(z_t)]$, so

$$\mathbb{E}[r(z^\star) - r(z_{t+1}) \mid z_t] \;\leq\; \Big(1 - 2\mu\eta\alpha + \mu L\eta^2\kappa_1\Big)\,[r(z^\star) - r(z_t)].$$

Choosing $\eta = \alpha/(L\kappa_1)$ makes the bracket equal to $1 - \mu\alpha^2/(L\kappa_1)$, yielding the claim. □

A.1 Query Complexity and Dimension Dependence

Dimension-Free Case. When rationales provide full gradient information ($v_t \in \mathbb{R}^d$) per query, the query complexity equals T and is dimension-independent:

$$\text{Queries} = O\left(\frac{L(\alpha^2 + \sigma^2)}{\alpha^2\mu}\log\frac{1}{\epsilon}\right) \tag{3}$$

Coordinate-Sparse Case. Suppose each query reveals one coordinate of $\nabla r(z_t)$ chosen uniformly. Using the unbiased estimator $v_t = d\,(\partial_i r(z_t))\,e_i$ with $i \sim \text{Unif}([d])$ gives $\alpha = 1$, $\sigma^2 = d - 1$, and hence $\kappa_1 = d$ and stepsize $\eta = 1/(Ld)$. We have

$$T = O\Big(\frac{Ld}{\mu}\log\frac{1}{\epsilon}\Big), \qquad \text{Queries} = O\Big(\frac{Ld}{\mu}\log\frac{1}{\epsilon}\Big).$$

Alternatively, averaging m independent coordinate queries per iteration yields $\sigma^2 = (d-1)/m$; taking $m = d$ recovers $T = O((L/\mu)\log(1/\epsilon))$ with d queries per iteration, so total queries remain $\Theta\big(\frac{Ld}{\mu}\log\frac{1}{\epsilon}\big)$.

B Lower Bounds for Exhaustive/Random Zeroth-Order Search

We formalize the exponential (in d) query complexity of exhaustive (grid) search and best-of-N random sampling when only function values (or preferences) are used without directional information. The hard instance is the strongly concave quadratic

$$r(z) \;=\; r(z^\star) - \tfrac{\mu}{2}\,\|z - z^\star\|_2^2, \qquad z \in B_R(z^\star) \subset \mathbb{R}^d,$$

whose ϵ-optimal set is the ball $B_{\rho_\epsilon}(z^\star)$ with radius $\rho_\epsilon = \sqrt{2\epsilon/\mu}$.

Proposition 2 (Grid-search lower bound)**.** *Let $B_R(z^\star) \subset \mathbb{R}^d$ and consider a hypercubic grid of spacing h. Its covering radius is $\rho = \frac{\sqrt{d}\,h}{2}$. To guarantee that* for all *placements of $z^\star$ there exists a grid point in the ϵ-optimal ball $B_{\rho_\epsilon}(z^\star)$ with $\rho_\epsilon = \sqrt{2\epsilon/\mu}$, it suffices that $\rho \le \rho_\epsilon$ (i.e., $h \le 2\rho_\epsilon/\sqrt{d}$). Furthermore, any such grid restricted to $B_R(z^\star)$ must contain at least*

$$N \;\ge\; \left(\frac{R}{\rho}\right)^d \;=\; \left(\frac{R\sqrt{d}}{2\rho_\epsilon}\right)^d \;=\; \left(\frac{\mu R^2 d}{8\,\epsilon}\right)^{d/2}$$

points. Hence exhaustive grid search requires $(1/\epsilon)^{d/2}$ queries on this family.

Proof. Coverage of $B_R(z^\star)$ by N balls of radius ρ centered at grid points implies $NV_d\rho^d \ge V_d R^d$, hence $N \ge (R/\rho)^d$. With $\rho = \sqrt{d}\,h/2$ and $h \le 2\rho_\epsilon/\sqrt{d}$, we obtain $N \ge (R\sqrt{d}/(2\rho_\epsilon))^d$. Substitute $\rho_\epsilon = \sqrt{2\epsilon/\mu}$ to conclude. □

Proposition 3 (Best-of-N random sampling lower bound)**.** *Draw $X_1, \dots, X_N \overset{i.i.d.}{\sim} \mathrm{Unif}(B_R(z^\star))$ and let $\hat{z} = \arg\max_i r(X_i)$ for $r(z) = r(z^\star) - \frac{\mu}{2}\|z - z^\star\|_2^2$. Then with $a \triangleq 2/d$,*

$$\mathbb{E}[r(z^\star) - r(\hat{z})] = \frac{\mu R^2}{2}\, N\,\mathrm{B}(1{+}a, N) = \frac{\mu R^2}{2}\,\Gamma(1{+}a)\,\frac{\Gamma(N+1)}{\Gamma(N+1+a)}.$$

Moreover, for all $d \ge 1$ (so $a \in (0, 2]$),

$$\frac{\Gamma(N+1)}{\Gamma(N+1+a)} \;\ge\; (N+2)^{-a},$$

and thus

$$\mathbb{E}[r(z^\star) - r(\hat{z})] \;\ge\; \frac{\mu R^2}{2}\,\Gamma\Big(1 + \frac{2}{d}\Big)\,(N+2)^{-\frac{2}{d}} \;=\; \Omega\big(N^{-\frac{2}{d}}\big).$$

Proof. Let $R_i = \|X_i - z^\star\|_2$ and $R_{\min} = \min_i R_i$. The CDF of $R_{\min}$ is $F(r) = 1 - (1 - (r/R)^d)^N$ for $r \in [0, R]$. Differentiating, $f(r) = Ndr^{d-1}R^{-d}(1 - (r/R)^d)^{N-1}$. Then

$$\mathbb{E}[R_{\min}^2] = \int_0^R r^2 f(r)\,dr = NR^2 \int_0^1 t^{\frac{2}{d}}(1-t)^{N-1}dt = NR^2\,\mathrm{B}\big(1 + \tfrac{2}{d},\, N\big),$$

where $t = (r/R)^d$ and B is the Beta function. Using $\mathrm{B}(a, b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}$ gives the exact expression. For the bound, we use the inequality $\Gamma(N+1)/\Gamma(N+1+a) \ge (N+2)^{-a}$ which holds for all $a \in (0, 2]$ and $N \ge 1$. □

C Extended Experimental Details

C.1 Implementation Details

SVG Code Optimization. We use `gpt-5-mini` to generate SVG/TikZ code, which is rendered to PNG for pairwise comparison by a separate judge instance. The system maintains a champion design that updates only when both A-vs-B and B-vs-A orderings agree on a winner. Winning rationales accumulate in the generation prompt to guide subsequent iterations.

Setup	Subject	Ink Wash	Minimalist	Realism	Retro Arcade	Stained Glass
From Scratch	City Skyline	100.0%	100.0%	100.0%	100.0%	100.0%
	Mushroom	100.0%	100.0%	83.3%	100.0%	100.0%
	Robot	100.0%	100.0%	100.0%	100.0%	100.0%
	Unicorn	100.0%	100.0%	100.0%	100.0%	100.0%
Judge-Aware	City Skyline	50.0%	100.0%	100.0%	100.0%	88.9%
	Mushroom	100.0%	100.0%	100.0%	100.0%	94.1%
	Robot	100.0%	100.0%	100.0%	93.8%	100.0%
	Unicorn	89.5%	100.0%	100.0%	93.8%	80.0%

Table 7: Full win rates after five iterations comparing Feedback Descent against direct prompting across all subject-judge combinations. **Feedback Descent wins or ties on all 40 subject-judge combinations.**

IFBench Prompt Optimization. We closely follow the setting of Agrawal et al. (2025) for this experiment, including their choice of LLMs (Qwen3-8B and GPT-4.1-mini) and multi-stage DSPy programs. For Qwen3-8B, we use 10 iterations with no early stopping (patience=0) with temperature 0.6, and evaluate 2 prompt candidates per round across 20 examples. For GPT-4.1-mini, we use 15 iterations with early stopping (patience=5) and temperature 1.0 for diversity. Both configurations use the same prompt improver template, described in Section C.3.

We programmatically generate a textual description of the difference between two prompts. To compare two prompts, we first partition the training set into four quadrants based on outcomes: examples where prompt A succeeds and B fails (A-wins), A fails and B succeeds (B-wins), both fail (tie-fail), or both succeed (tie-success). We then use the same LLM to propose hypotheses about input characteristics (based on the prompt text) and output patterns (based on the response and evaluation feedback), producing ∼20 hypotheses per category. To evaluate whether each hypothesis applies to each example, we use the same LLM as the tagger that outputs binary labels (1 if the hypothesis matches, 0 otherwise) for all hypotheses in a single call per example, processing hundreds of examples in parallel. We then compute lift metrics for each hypothesis-quadrant pair, where lift is the ratio of conditional probability to the base rate (i.e., how much more likely an outcome is given the hypothesis holds). We validate the hypotheses statistically using Fisher's exact test, and filter for hypotheses that are statistically significant at $p < 0.1$ with minimum support of 3 examples and lift ≥ 2.0 for A/B wins or ≥ 1.5 for failures. This analysis identifies which input patterns correlate with differential performance and which output characteristics appear when one prompt outperforms the other.

Molecule Optimization. We implement molecular optimization using the DOCKSTRING package (García-Ortegón et al., 2022) for protein-ligand docking simulations across six therapeutic targets. The system begins with three seed molecules (acetamide, pentane, benzene) and iteratively proposes new SMILES strings, using RDKit molecular properties, protein binding site information, and similarity to approved drugs as metadata. We use the combined score function suggested by DOCKSTRING:

$$s_{\text{overall}}(\text{molecule}, \text{protein}) = -\texttt{Vina}(\text{molecule}, \text{protein}) - 10 \cdot (1 - \texttt{QED}(\text{molecule})), \quad (4)$$

where `Vina` provides the binding affinity prediction (kcal/mol, more negative is better) and the QED penalty term penalizes molecules with poor drug-likeness, with higher scores indicating better molecules that balance binding strength and drug-like properties. Note that QED scores range from 0 to 1 while Vina scores typically range from -3.0 to -12.0 kcal/mol. For Feedback Descent, we run 1000 iterations with a batch size of 8, conditioning on the top 10 molecules from previous iterations.

C.2 Additional Results

Figure 4 shows that across all protein targets, the discovered molecules improve on both docking affinity and drug-likeness relative to the DOCKSTRING dataset, extending the Pareto frontier.

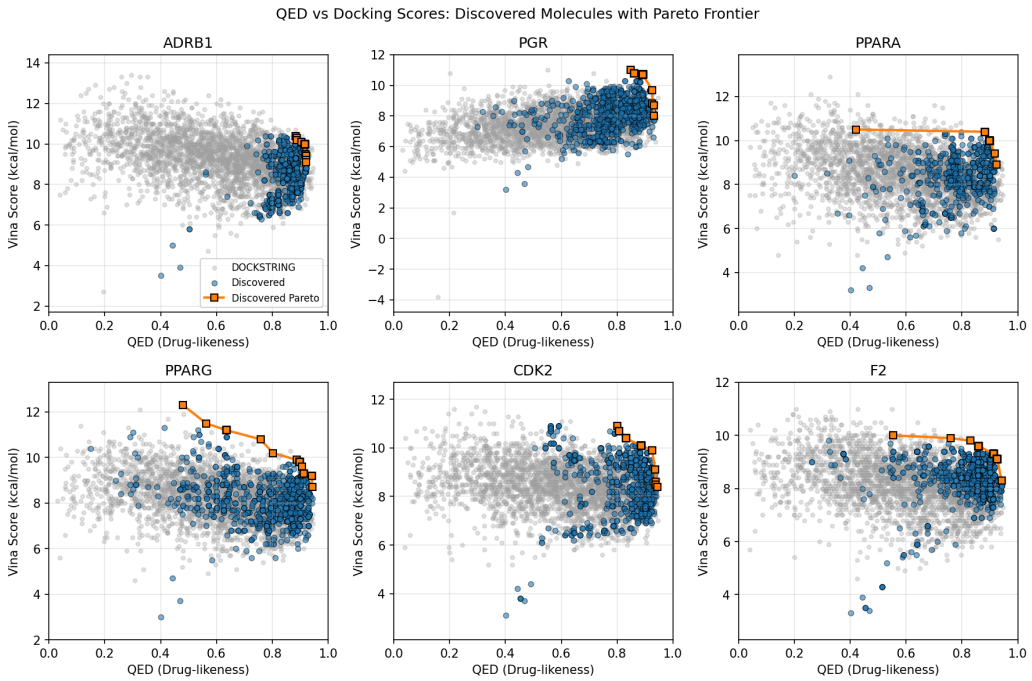

Figure 4: Pareto frontiers of discovered molecules (blue) compared against molecules in the DOCKSTRING dataset (gray) across six protein targets. The highlighted orange markers indicate molecules on the discovered Pareto frontier, achieving joint improvements in docking affinity (Vina score) and drug-likeness (QED).

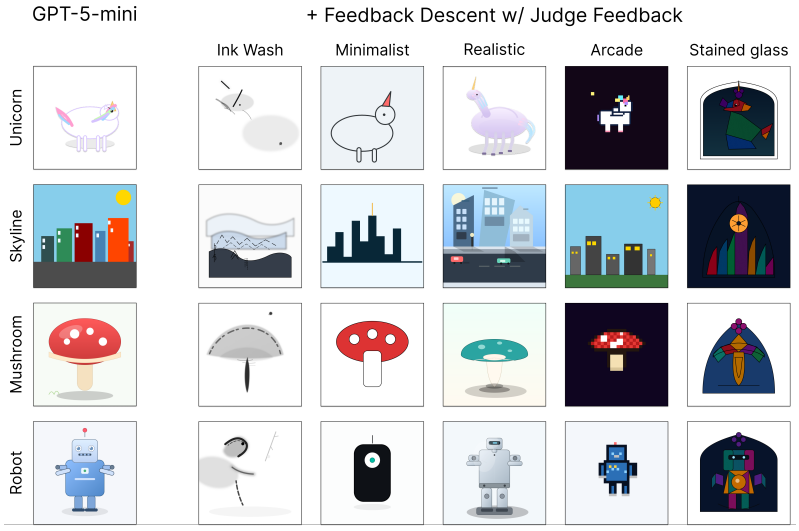

Figure 5: Example images generated by Feedback Descent under six different judge criteria. **Feedback Descent produces distinct outputs matching each judge's aesthetic criteria.**

C.3 Prompt Templates

We use the following prompt for the Anatomy SVG judge. The other rubrics follow the same structure, translating each aesthetic into operational rules.

Ablation	ADRB1	PGR	PPARA	PPARG	CDK2	F2	Avg
No Feedback	6.190	8.619	8.230	8.633	8.300	8.793	8.127
Random Feedback	6.604	8.385	8.276	6.780	8.793	7.993	7.805
Binary Only	5.863	8.779	8.507	7.998	9.439	8.420	8.168
Full (ours)	**10.623**	**9.615**	**9.919**	**10.187**	**9.803**	**9.300**	**9.908**

Table 8: Per-target ablation results on DOCKSTRING. **Detailed feedback outperforms all degraded variants on all six protein targets.**

	Method	ADRB1	PGR	PPARA	PPARG	CDK2	F2
DOCKSTRING (N=260155)	Top 50%	5.305	3.478	4.549	4.210	4.385	4.168
	Top 90%	8.785	7.878	7.987	7.658	7.733	7.477
	Top 99%	9.620	8.703	8.718	8.449	8.453	8.139
	Top 99.9%	10.209	9.260	9.230	9.012	8.979	8.722
	Top 99.99%	10.742	9.723	9.821	9.518	9.509	9.252
	Best Molecule	11.330	9.742	9.907	9.529	9.534	9.311
	GP-BO† (Tripp et al., 2021)	10.552 ± 0.140	9.307 ± 0.177	9.680 ± 0.337	9.485 ± 0.279	9.067 ± 0.289	8.686 ± 0.068
	Graph MCTS† (Jensen, 2019)	8.883 ± 0.826	7.819 ± 0.319	7.363 ± 0.935	7.134 ± 0.855	7.777 ± 0.723	6.310 ± 0.704
	Graph GA† (Jensen, 2019)	10.249 ± 1.002	8.793 ± 0.497	9.211 ± 0.343	8.769 ± 0.432	8.652 ± 0.449	8.900 ± 0.817
	SMILES GA (Brown et al., 2019)	9.334 ± 0.237	8.335 ± 0.276	9.052 ± 0.484	8.560 ± 0.346	8.268 ± 0.170	7.984 ± 0.554
	REINVENT (Olivecrona et al., 2017)	9.867 ± 0.522	8.604 ± 0.483	8.735 ± 0.120	9.054 ± 0.153	8.695 ± 0.370	8.441 ± 0.535
	No Feedback (Best-of-N)	6.190 ± 0.821	8.619 ± 0.562	8.230 ± 0.628	8.633 ± 0.549	8.300 ± 0.620	8.793 ± 0.921
	Random Feedback	6.604 ± 0.577	8.385 ± 0.258	8.276 ± 0.628	6.780 ± 0.523	8.793 ± 0.921	7.993 ± 0.663
	Binary Only	5.863 ± 0.428	8.779 ± 0.633	8.507 ± 0.428	7.998 ± 0.571	9.439 ± 0.922	8.420 ± 0.315
	TextGrad (Yuksekgonul et al., 2024)	8.531 ± 0.278	8.057 ± 0.383	7.953 ± 0.160	7.256 ± 0.886	8.174 ± 0.395	7.357 ± 0.821
	Feedback Descent	$\mathbf{10.623 \pm 0.112}$	$\mathbf{9.615 \pm 0.158}$	$\mathbf{9.919 \pm 0.305}$	$\mathbf{10.187 \pm 0.253}$	$\mathbf{9.803 \pm 0.267}$	$\mathbf{9.300 \pm 0.062}$

Table 9: Full results for molecule optimization on six protein targets with standard deviations. Fragment-based algorithms (denoted by †) operate directly on molecular graphs, giving them structural priors unavailable to text-based methods. For each target, the top generative result is in **bold**, and DOCKSTRING percentiles exceeding the best generative result are underlined.

Anatomy Judge Rubric

```
RUBRIC NAME: Anatomical Realism
INTENT: Believable equine anatomy with a plausible horn; form, proportion, and structure matter most.


NON-NEGOTIABLES:
- Recognizable equine proportions; head, neck, torso, four legs, mane, tail, horn present.
- Limbs connect anatomically; joints and hooves indicated.

CRITICAL BENCHMARKS (must evaluate these first):
1. Head-Neck Proportion: Neck length should be ~1.5x head length; head meets neck high on shoulders
2. Body Square: Body length (shoulder to buttock) ~= height at withers; chest depth ~= elbow height
3. Leg Structure: Proper joint articulation with elbow under withers; fetlock/pastern angles 45-55
deg when standing; all four limbs distinct and correctly connected

WHAT TO REWARD:
- Correct limb count and articulation; mass distribution that could stand or move.
- Horn integrates naturally with the skull (frontal bone center, 2-3" above eye line).
- Subtle shading or line variation conveying volume.
- Ground contact or cast shadow for grounding.
- Visible muscle definition suggesting tension/relaxation appropriate to pose.
- Differentiated hair textures: short coat vs coarse mane/tail strands.
- Anatomical landmarks: withers prominence, gaskin curve.

WHAT TO PENALIZE:
- Missing or fused legs; impossible joints; balloon torsos.
- Flat cardboard profiles with no sense of volume.
- Decorative effects that obscure structure.
- Disney-fied proportions (oversized eyes, baby-like features).
- Horn placement anywhere except frontal bone center (2-3" above eye line).

TIEBREAKERS:
- Prefer the image with more accurate limb/neck/head proportions.
- If both are plausible, choose the one with better weight and grounding.
```

We use the following templates for generating candidates and rationales in prompt optimization.

System Prompt Template for Prompt Optimization

```
Improve the assistant's prompt by extracting actionable insights from the data.

## Goal
Create prompts that generalize well beyond the training examples you see here. The patterns below
come from a small sample; your output must work on thousands of unseen cases.

## Current Prompts
**Approach A (Baseline):**
```python
{prompt_a_dict}
```

Approach B (Challenger):
```python
{prompt_b_dict}
```

Training Signals
{comparison}

Prompt Improvement Strategy

1. Extract Core Insights
Identify patterns with strong evidence (low p-value, high lift, good support):
- What fundamental strategies distinguish success from failure?
- What misunderstandings or errors repeatedly occur?
- Are there essential facts or constraints the model needs to know?

2. Avoid Common Pitfalls
- Redundancy: Don't say the same thing multiple ways
- Over-specification: Don't list every possible format, constraint, or edge case
- Defensive bloat: Don't add uncertainty handling or safety warnings unless critical
- Surface patterns: Look for deep semantic strategies, not superficial formatting rules
- Enumerationitis: Avoid long numbered checklists; prefer flowing prose

3. Craft Effective Instructions
- State principles clearly and concisely
- Use specific language when precision matters ("identify the missing fact" vs "analyze the
information")
- Keep instructions proportional to task complexity
- Test in your mind: would this help on examples you haven't seen?

4. Preserve What Works
- If baseline is effective and simple, make minimal changes
- Don't fix what isn't broken
- Complexity should buy you something measurable

The prompt must be a Python dictionary with the following keys:
{module_keys_description}

Output EXACTLY in this format:

```python
{prompt_template}
```
```

System Prompt Template for Molecule Optimization

```
You are a helpful assistant that proposes a single SMILES string corresponding to a small molecule,
based on the task explanation and information about previous molecules that have been proposed. Your
 proposed molecule should score highly on the {benchmark_name} benchmark.

Task explanation: Maximize -{benchmark_name} - 10 * (1 - QED): larger values mean stronger {
benchmark_name} binding and high drug-likeness.
{protein_info_xml}

Your output format should be EXACTLY as follows:
```

```
<reasoning>[Chemical hypothesis linking structural changes to expected binding/QED improvements]</
reasoning>
<smiles>[Single SMILES string]</smiles>

Generate a small molecule based on these previous examples (sorted in descending order of score):

{examples_text}
```

Example of Protein Metadata (ADRB1)


```
{ 'target': 'ADRB1', 'accession': 'P08588', 'regions': { 'transmembrane': [[56, 84], [94, 120],
[133, 154], [173, 196], [223, 248], [320, 349], [355, 377]], 'extracellular': [[1, 55], [121, 132],
[197, 222], [350, 354]], 'cytoplasmic': [[85, 93], [155, 172], [249, 319], [378, 477]], 'disordered
': [[269, 307], [403, 477]]}, 'critical_residues': {'mutagenesis': [{'position': [474, 474], '
description': 'Loss of interaction with GOPC.'}, {'position': [474, 474], 'description': 'Loss of
interaction with GOPC; when associated with A-477.'}, {'position': [475, 475], 'description': 'Loss
of interaction with GOPC. Loss of interaction with RAPGEF2. Abolishes agonist-induced Ras activation
.'}, {'position': [475, 475], 'description': 'Loss of interaction with RAPGEF2.'}, {'position':
[475, 475], 'description': 'Partial loss of interaction with GOPC.'}, {'position': [476, 476], '
description': 'Partial loss of interaction with GOPC.'}, {'position': [477, 477], 'description': '
Loss of interaction with GOPC.'}, {'position': [477, 477], 'description': 'Loss of interaction with
RAPGEF2. Abolishes agonist-induced Ras activation.'}], 'natural_variants': [{'position': [26, 26], '
description': 'in dbSNP:rs34844626'}, {'position': [29, 29], 'description': 'in dbSNP:rs35720093'},
{'position': [31, 31], 'description': 'in dbSNP:rs35230616'}, {'position': [49, 49], 'description':
'correlated with low mean resting heart rate and decreased mortality risk in patients with
congestive heart failure; dbSNP:rs1801252'}, {'position': [187, 187], 'description': 'found in
individuals with short sleep; results in decreased adenylate cyclase-activating adrenergic receptor
signaling; decreased protein stability; dbSNP:rs776439595'}, {'position': [389, 389], 'description':
 'increased beta1-adrenergic receptor activity; increased basal activity and increased coupling to
heterotrimeric G protein Gs that stimulates the adenylyl cyclase; dbSNP:rs1801253'}, {'position':
[399, 399], 'description': 'in dbSNP:rs36052953'}, {'position': [405, 405], 'description': 'in dbSNP
:rs35705839'}]}}
```


Example of Molecule Metadata (CCCCC)


```
valid: 'True'
score: '-1.9121449019886678'
metadata:
  CanonicalSMILES: CCCCC
  InChIKey: OFBQJSOFQDEBGM-UHFFFAOYSA-N
  MolecularFormula: C5H12
  ExactMass: '72.093900384'
  FormalCharge: '0'
  AtomCount: '5'
  HeavyAtomCount: '5'
  HeteroAtomCount: '0'
  BondCount: '4'
  Sp3CarbonFraction: '1.0'
  RingCount: '0'
  AromaticRingCount: '0'
  AliphaticRingCount: '0'
  RotatableBondCount: '2'
  StereoCenterCount: '0'
  MurckoScaffold: ''
  LogP: '2.1965000000000003'
  TopologicalPolarSurfaceArea: '0.0'
  MolarRefractivity: '25.19899999999999'
  HBondDonorCount: '0'
  HBondAcceptorCount: '0'
  BertzComplexityIndex: '7.5097750043269365'
  BalabanJIndex: 2.19060968716425
  HallKierAlpha: '0.0'
  Kappa1: '5.0'
  Chi0v: '4.121320343559642'
  TotalEState: 8.5
  MinEState: 1.34375
  MaxEState: 2.2118055555555554
  PEOE_VSA6: '33.10993926815928'
  SlogP_VSA5: '33.10993926815928'
  BCUTp_1h: '13.744962415414642'
  AccessibleSurfaceArea: '34.19901948541599'
  FunctionalGroups: []
```

```
StructuralAlerts: []
QuantitativeDrugLikeness: '0.4687855098011332'
SyntheticAccessibility: '1.699621281696647'
NaturalProductLikeness: '0.09749981667944'
```