# CHEMOGRAD: IMPROVING CHEMICAL REASONING THROUGH TEST-TIME LATENT OPTIMIZATION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

The latent space of chemical LLMs encodes the underlying characteristics and principles of chemical structures, making it possible to generate chemical compounds by reasoning directly within this space. Yet existing latent-reasoning methods for chemical generation lack a clear direction: the latent state is updated by re-forwarding it through the model, without a reward signal or verifier to steer the search. We propose ChemoGrad, which optimizes molecules by a reward-guided search over a frozen model's latent states: it scores each candidate with a property evaluator, nudges the hidden states to perturb the emitted prefix, then re-decodes and keeps a candidate only when the evaluator confirms an improvement. A likelihood-sharpening step proposes candidates around the model's own trace, a verifiable reward selects which to pursue, and a task-oriented initialization points the search in the right chemical direction — shifting task adaptation from training-time parameter learning to inference-time search. On ChemCoTBench, ChemoGrad consistently outperforms the state-of-the-art baseline across all six molecule-optimization tasks, gaining $+51\%$ in mean property improvement and $175\%$ on JNK3 activity, with no additional training; it also transfers to editing, reaction prediction, and captioning.

## 1 INTRODUCTION

Motivated by reasoning-augmented LLMs—such as OpenAI o1 and DeepSeek-R1—that improve answers by deliberating before decoding, chemistry models now reason in one of two spaces: chain-of-thought (CoT) text, which spells out the chemical rationale as an explicit trace, or latent "thinking" tokens that internalize the chain into continuous vectors, cutting inference cost while still steering the answer. The strongest current system, LatentChem (Ye et al., 2026), distills CoT into a latent-thinking module through four sequential training stages built around supervised fine-tuning and GRPO (Guo et al., 2025) with property rewards. Yet this ability is baked into the weights and cannot be revised without re-running the pipeline; more fundamentally, the latent state in these methods is merely re-forwarded through the model with no explicit direction — no reward or verifier steers the search — even though a trained chemistry LLM's latent space already encodes the principles of chemical structure. The question is whether we can use that knowledge in place to move a molecule toward a goal, rather than distilling it into the weights first.

To address these limitations, we propose **ChemoGrad**, a training-free framework for molecular optimization via a reward-guided search in a frozen model's latent space. As shown in Figure 1, ChemoGrad treats the hidden states produced by the model's forward pass as directly optimizable variables, casting reasoning as a search for latent representations that decode into molecules with higher task-specific rewards. At test time, each candidate is evaluated by an external oracle, such as an RDKit property calculator, an activity classifier trained on TDC data (Huang et al., 2021), or a reference-based metric such as Tanimoto similarity or METEOR (Denkowski & Lavie, 2014). Starting from the hidden states that generated the initial answer, ChemoGrad iteratively updates these representations, decodes the revised latent states into a candidate molecule, and accepts the candidate only when the oracle confirms a reward improvement. Crucially, gradients are computed through the language-model head with respect to the latent states alone, without backpropagating through the backbone or updating any model parameters. ChemoGrad thus directly optimizes the model's existing representations at test time, requiring no additional training.



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.

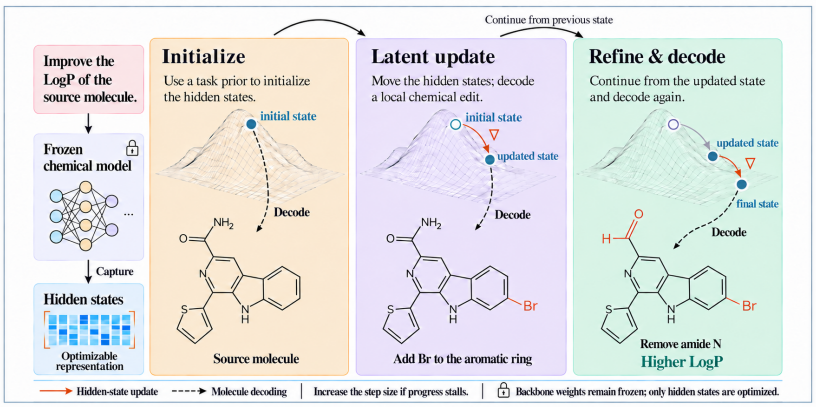


Figure 1: **Overview of hidden-state optimization.** A frozen model's hidden states are iteratively updated by policy-gradient steps and decoded into molecules; a task prior guides initialization, and search continues in place with a larger step when progress stalls.

Applied to *ChemCoTBench* (Hao et al., 2026), the standard molecule-optimization benchmark, ChemoGrad improves over the fully trained four-stage baseline on all six tasks, by a mean of $+0.30$ (a $+51\%$ increase over the baseline's mean terminal $\Delta$) with a 94–100% success rate, with no additional training, and transfers to editing, reaction prediction, and captioning. Our analysis shows the gain is driven by exploration in the latent space: each likelihood-sharpening step perturbs the hidden state across a chemical-modification landscape, and the monotone accept/reject step deepens the search along the directions the oracle rewards most. The result is cheap to deploy: the model's own forward-pass representations, a verifiable scorer, and a short gradient-update loop — no distillation, fine-tuning, or critic model.

To summarize:

- We introduce **ChemoGrad**, a training-free method that turns a chemistry model's hidden states into directly optimizable variables at test time.
- We ground the search in an externally verifiable, deterministic chemical oracle rather than a learned critic, so every accept/reject decision is reproducible and free of reward-model drift.
- We show ChemoGrad improves over a fully trained four-stage baseline on all six *ChemCoTBench* tasks with reference-free verifiers, and that the same mechanism transfers to editing, reaction prediction, and captioning.
- We trace where the gain comes from, showing the optimized molecules follow an interpretable, task-aligned vocabulary of chemical edits.

## 2 Method: ChemoGrad

### 2.1 Problem Formulation: Test-Time Molecular Reasoning

Let $\boldsymbol{x}$ be the prompt encoding the task together with the source molecule, and let $\pi$ be a pre-trained autoregressive chemistry LLM composed of a frozen Transformer backbone and a language-model head $\mathrm{lm_head}(\cdot)$. Given $\boldsymbol{x}$, the model emits a reasoning sequence $\boldsymbol{y} = (y_1, \ldots, y_T)$ with last-layer hidden states $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_T$, and the generated answer decodes to a molecule $\hat{m}$. At test time the ground truth is unknown, so a verifiable reward $r(\hat{m}) \in [-1, 0]$, an externally computed chemical score such as an RDKit/TDC property, evaluates the decoded molecule. The objective of molecular reasoning is then to find the answer whose molecule scores highest:

$$\hat{m}^\star \;=\; \arg\max_{\hat{m}} \; r(\hat{m}), \tag{1}$$

a discrete optimization over the space of valid molecules that a single forward pass can only sample once.

### 2.2 Optimization in the Latent Space

Searching over discrete tokens directly (Eq. 1) is intractable: the molecule space is large, discrete, and tightly coupled, so a single-atom edit can break validity. We therefore optimize over a sequence of latent representations instead. Let $\boldsymbol{z} = (\boldsymbol{h}_1, \ldots, \boldsymbol{h}_U)$ denote the first $U$ hidden states emitted along the reasoning path; decoding them through $\mathrm{lm_head}(\cdot)$ and continuing generation yields a concrete molecule, so this trajectory is a differentiable surrogate of the reasoning. Our goal is to find the latent state whose *expected* reward is highest:

$$\boldsymbol{z}^\star = \arg\max_{\boldsymbol{z}} \; \mathbb{E}_{\hat{m}\sim\pi(\cdot|\boldsymbol{z})}[\, r(\hat{m})\,], \qquad \boldsymbol{z} = (\boldsymbol{h}_1, \ldots, \boldsymbol{h}_U),\; U = \rho T, \tag{2}$$

where $\boldsymbol{z}$ is the leading fraction of emitted hidden states treated as optimizable variables, $\hat{m}$ is a molecule decoded from $\boldsymbol{z}$ and sampled from the model's distribution $\pi(\cdot \mid \boldsymbol{z})$, and every gradient flows only through $\mathrm{lm_head}(\cdot)$. Because these hidden states already encode the model's understanding of chemical structure, updating them explores the set of molecules the model itself already favours, without touching its weights. Figure 1 illustrates one loop.

**Optimization loop.** Given hyperparameters $\rho, K, \eta, \tau_{\text{stop}}$, ChemoGrad alternates between decoding a candidate from the current latents and taking a policy-gradient update step on $\boldsymbol{z}$. To decode a molecule from latents $\boldsymbol{z}$, we sample each prefix token from the distribution $\mathrm{lm_head}(\cdot \mid \boldsymbol{z})$ under the sampling temperature (0.7) and top-$p$ (0.9) used throughout, and continue autoregressive generation from there. Since the remaining sequence may extend beyond the latent-initialized prefix, the full answer factorizes as

$$\pi(\hat{m} \mid \boldsymbol{z}, \boldsymbol{x}) \;=\; \prod_{i\le U} \mathrm{lm_head}(y_i \mid \boldsymbol{z}_i) \;\cdot \prod_{i>U} \pi(y_i \mid y_{<i}, \boldsymbol{x}), \tag{3}$$

where the first product decodes the optimized prefix and the second resumes normal generation. Concretely, the updated prefix $\boldsymbol{z}$ is decoded through $\mathrm{lm_head}$ into concrete prefix tokens, and generation of the suffix $y_{>U}$ continues autoregressively from those tokens: the suffix therefore depends on $\boldsymbol{z}$ through the discrete prefix $\hat{y}_{1:U}$, the same factorization Eq. (3) encodes, rather than through a separately cached continuous state. Only the prefix factor is differentiable in $\boldsymbol{z}$ (with the gradient taken at the logits $\mathrm{lm_head}(y_i \mid \boldsymbol{z}_i)$), so gradient flows through $\mathrm{lm_head}(\cdot)$ alone and never into the backbone; the corrected prefix is re-materialized by a forward recomputation of the prefix logits when the update is applied. Optimizing Eq. 2 by policy gradient gives the REINFORCE update (Williams, 1992):

$$\boldsymbol{z}_{k+1} \;\leftarrow\; \boldsymbol{z}_k + \eta \cdot \mathcal{J}(\boldsymbol{z}_k), \tag{4}$$

$$\mathcal{J}(\boldsymbol{z}_k) \;=\; \mathbb{E}_{\hat{m}\sim\pi(\cdot|\boldsymbol{z}_k)}\big[\, \big(r(\hat{m}) - b\big)\, \nabla_{\boldsymbol{z}} \log \pi(\hat{m} \mid \boldsymbol{z}_k)\,\big], \tag{5}$$

where $b$ is the running-mean reward baseline, and the expectation is approximated by the empirical mean over one step's decoded candidates; centering by $b$ makes the update size reflect each candidate's deviation rather than its raw magnitude. The reward $r(\hat{m}) \in [-1, 0]$ is a black-box scalar held *constant* during the step, so $\nabla_{\boldsymbol{z}} r$ does not appear and no gradient flows through the chemistry itself: $r$ weights the score-function direction $\nabla_{\boldsymbol{z}} \log \pi(\hat{m} \mid \boldsymbol{z})$, exactly as in a policy-gradient estimator. The update thus provides a *diversification* signal — it repoints the hidden state so re-decoding samples molecules distinct from the current one along the reward-confirmed edits — while the *move* toward higher reward is enforced by the accept/reject step; an external, deterministic chemical oracle replaces any learned or self-generated critic.

Concretely, one iteration takes a policy-gradient step on the latent state (Eq. 4), re-decodes the updated latents into a candidate molecule (Eq. 3), and queries the oracle: if the reward improves we adopt the candidate and its latents, otherwise we roll back and keep the previous best. The loop stops when the reward clears $\tau_{\text{stop}}$, 20 consecutive flat steps are exhausted, or $K$ iterations elapse, and returns the best molecule seen so far — never worse than the initial output. This policy-gradient update of the hidden state rather than the weights is what makes ChemoGrad training-free and per-instance.

### 2.3 Enhancing Techniques

The bare loop already makes the update larger along whatever edit the model commits to; two complementary techniques make it more stable and efficient. **Task-Prior Initialization:** the initial

latent state is seeded with the task's chemical prior — a task-specific hint appended to the prompt (e.g. "to raise LogP, add halogen or alkyl to an aromatic ring") — so the search launches from a latent state already oriented toward the correct edit direction, driving *logp* to a $100\%$ success rate and stabilizing the harder activity tasks. **Deepening the Update:** a geometrically increasing learning rate ($\eta \cdot 2^n$ over up to 3 rounds) engages when early convergence leaves headroom: when the base loop's property improvement falls below the task's target, ChemoGrad doubles the learning rate in place, continuing from the current hidden state with no re-initialization to move past a flat region instead of stalling. Task-prior initialization invests in the start of the update and deepening in its continuation, together buying faster convergence and deeper search without touching any weight.

### 2.4 Reward Design

ChemoGrad requires only that $r$ be externally verifiable; we instantiate task-appropriate rewards, and mark explicitly whether each one can be computed at inference time from the model's own output alone, or whether it must be scored against a held-out reference. We call a reward that reads the ground-truth answer a *gold* verifier, and everything else an *avail* verifier. The naming follows a long-standing convention in reward-guided decoding of separating the benchmark evaluator from the scorer a deployed system can actually run:

- **Property rewards** (*mol_opt*, *avail*): RDKit/TDC property of the generated molecule, mapped through a soft sigmoid so that a full target-scale improvement saturates the reward. Computed from the molecule alone; no reference needed.
- **Edit rewards** (*mol_edit*, *avail*): functional-group-count rules continuous-ized, rewarding correct add/delete/substitute operations, again reference-free.
- **Reaction rewards** (*rxn*, *gold*): Tanimoto similarity (Bajusz et al., 2015) between the predicted and ground-truth products, generalized to multi-component cases for retrosynthesis and reagent prediction.
- **Caption rewards** (*gold*): METEOR (Denkowski & Lavie, 2014) between the predicted and reference molecular descriptions, using the same *nltk* implementation that the benchmark employs for official evaluation.

This distinction matters for how the results should be read: the *avail* verifiers are the ones a deployed system can actually run, and they cover the six *mol_opt* tasks and three *mol_edit* subtasks, so every headline gain on the main benchmark uses an *avail* verifier only. The *gold* verifiers for reaction and captioning probe whether the mechanism transfers to text-similarity and set-matching rewards without code change — an upper bound on generality, not a deployment claim.

### 2.5 Why the Search Works

Why should such a simple loop help? Three complementary forces. First, the leading hidden states $\boldsymbol{z} = \boldsymbol{h}_{1:U}$ act as a control surface over the decoded molecule: a perturbation read through the head can flip the identity of the emitted molecule, so a few likelihood-sharpening steps already realize large, single-step edits (quantified in §4.3). Second, the reward admits a candidate only when the oracle confirms it is better — the loop proposes then filters rather than chasing a learned critic, which is also why invalid SMILES, yielding no accepted candidate, never derail the search. Third, a task-oriented hint points the enumeration at the right region of chemical space before the first step.

## 3 Experiments

### 3.1 Experimental Settings

**Benchmarks and baselines.** We conduct experiments on *ChemCoTBench* (Hao et al., 2026), whose *mol_opt* sub-suite contains six tasks, *logp*, *solubility*, *qed*, and three pharmacological activity targets *drd2*/*jnk3*/*gsk3*$\beta$, each scored by a verifiable oracle: RDKit Crippen LogP (Ghose & Crippen, 1986), ESOL solubility (Delaney, 2004), QED drug-likeness (Bickerton et al., 2012), or TDC-trained activity classifiers (Huang et al., 2021). Each task is evaluated on 100 source molecules. Our primary baseline is LatentChem (Ye et al., 2026), a Qwen3-8B-Base (Yang et al., 2025) system

trained in four stages with property rewards; its released checkpoint supports a latent-thinking mode and a CoT mode emitting a full reasoning chain:

- **Training-based reasoning** refers to the fully trained system with latent-thinking enabled (configuration **A**, our primary comparison target), and to its chain-of-thought decoding mode with latent-thinking disabled (configuration **B**). B appears in two forms: $\mathbf{B}_{\textbf{paper}}$, the numbers quoted verbatim from LatentChem's published matching row (their sampling protocol, which we do not re-run), and $\mathbf{B}_{\textbf{local-greedy}}$, our own greedy batch decode of the released checkpoint used only for the per-sample paired tests where a local, pair-aligned reference is required. Where a row quotes $\mathbf{B}_{\textbf{paper}}$, we say so explicitly so the comparison is anchored to one consistent reference rather than a weaker local reproduction.
- **Test-time reasoning** refers to our method: **ChemoGrad** (full, applied to the CoT mode with the base loop $K{=}30$, $\tau_{\text{stop}}{=}-0.15$, $\rho{=}.5$, deepening the update, and task-prior initialization).

We additionally report an ablation chain (C, ChemoGrad) adding the mechanisms incrementally, plus an intermediate configuration $\text{C}^{+}$ (enlarged budget and task-prior init, no deepening) for the significance contrast in Appendix C.

**Baseline configuration and fairness.** Where we run a baseline locally, decoding is greedy so the reference is deterministic and pair-aligned per source molecule; the numbers quoted from the original paper are used verbatim as published. ChemoGrad's own candidates are sampled at temperature $0.7$ and top-$p{=}\,0.9$; the initial candidate pool for the scaling probe uses $8$ starts drawn at temperatures $0.6$–$1.6$ (Appendix C). Each table states which baseline form its row uses; the headline Table 1 quotes $\mathbf{B}_{\textbf{paper}}$, so the published baseline is taken at face value and protocol drift cannot inflate our gain.

**Hyperparameters and configurations.** We use the Adam optimizer (Kingma & Ba, 2014) with learning rate $\eta{=}0.05$, token ratio $\rho{=}0.5$, stop threshold $\tau_{\text{stop}}{=}-0.15$, and $K{=}30$ iterations. Deepening the update raises the learning rate $\eta \cdot 2^n$ for up to 3 rounds, triggered when the property improvement $\mathrm{imp}$ still falls below the task's target $\mathrm{RESTART_TARGET}$ after the base loop. Full details in Appendix A.

**Evaluation protocol.** Following the official metric, we report property improvement $\Delta$ and success rate (SR). For each test sample $i$,

$$\begin{aligned} \mathrm{imp}_i &= \mathrm{prop}(\hat{m}_i) - \mathrm{prop}(m_i^{\mathrm{src}}), \\ \Delta &= \mathrm{mean}(\mathrm{winsor}_{5\%-95\%}(\{\mathrm{imp}_i\})), \qquad \mathrm{SR} = \tfrac{1}{n}\textstyle\sum_i \mathbb{1}[\mathrm{imp}_i > 0], \end{aligned} \tag{6}$$

where $\hat{m}_i$ is the generated molecule (invalid SMILES contribute $\mathrm{imp} = 0$), and $\mathrm{winsor}_{5\%-95\%}$ clips at the 5th/95th percentiles.

## 4 Results and Analysis

### 4.1 Main Results: Molecule Optimization

ChemoGrad achieves state-of-the-art results on *ChemCoTBench* molecule optimization (Table 1, Figure 2), improving over the fully trained four-stage baseline (LatentChem, configuration A) on all six tasks by a mean of $+0.30$ over A's six-task mean terminal $\Delta$ of $0.60$ (a $+51\%$ increase) with a success rate of $94$–$100\%$ versus $60$–$96\%$, with no additional training, and exceeding it on all three description datasets (METEOR). Because the six tasks are scored on different scales, the $+51\%$ figure is a plain average of per-task relative gains, reported as a coarse headline rather than as a single incommensurable quantity; the per-task improvements are given intact in Table 1. Its largest gains land where the baseline is already strongest (*solubility* $+0.66$, *logp* $+0.39$), while the success-rate gap is widest on the activity tasks (*jnk3* $94\%$ vs. $60\%$, *gsk3*$\beta$ $99\%$ vs. $82\%$), where a single latent-update run crosses the reward landscape left unexplored by explicit chains; on *jnk3* the mean $\Delta$ more than doubles, from $0.08$ to $0.22$ (a $175\%$ relative gain). All six paired differences are positive, and a paired bootstrap (2000 resamples) confirms them significant ($p{<}0.01$); on the deterministic physical-property oracles the $95\%$ confidence intervals for the paired difference exclude

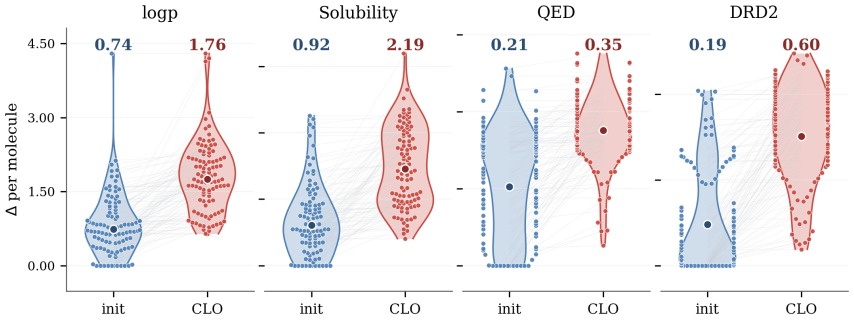


Figure 2: Per-molecule distribution shift on three physical-property tasks *logp*/*solubility*/*qed* and the *drd2* activity target: $\Delta$ under temperature 0.7 sampling (*init*, blue) vs. after one ChemoGrad exploration run of up to $K{=}30$ steps (*red*); grey links join each molecule's before/after pair.

Table 1: Open-ended generative tasks, extending Table 2 of Ye et al. (2026). Molecule optimization reports $\Delta$ and success rate %; description reports METEOR. The baseline rows (Qwen SFT, Coconut-Chem, LatentChem) are reproduced from the original paper; $B_{paper}$ is the same checkpoint's chain-of-thought decoding with latent-thinking disabled, quoted from the paper's matching row (per-sample provenance in the appendix).

| | | Molecule Optimization ($\Delta$/SR %) | | | | | | Description (METEOR) | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Category | Method | logp | sol | qed | drd | jnk | gsk | Mol-I | ChEBI | ChemLL |
| Text-only | Qwen-3-8B (SFT) | 0.15/47 | 0.48/52 | 0.10/48 | 0.04/38 | -0.02/20 | 0.02/36 | 0.12 | 0.12 | 0.13 |
| Chem. LLM | $B_{paper}$ | 0.67/77 | 1.17/86 | 0.19/82 | 0.19/70 | 0.03/44 | 0.13/67 | 0.07 | 0.05 | 0.07 |
| | Coconut-Chem | 0.17/44 | 0.57/58 | 0.15/67 | 0.04/45 | -0.00/35 | 0.06/47 | 0.09 | 0.07 | 0.04 |
| | LatentChem | 1.37/96 | 1.53/89 | 0.18/83 | 0.26/74 | 0.08/60 | 0.17/82 | 0.12 | 0.15 | 0.16 |
| Test-time | **ChemoGrad (ours)** | **1.76/100** | **2.19/100** | **0.35/100** | **0.60/100** | **0.22/94** | **0.30/99** | **0.24** | **0.21** | **0.21** |
| | $\Delta$ vs LatentChem | +0.39 | +0.66 | +0.17 | +0.34 | +0.14 | +0.13 | +0.12 | +0.06 | +0.05 |

zero throughout, underscoring the robustness of test-time latent search as a steering mechanism (full CIs in Appendix C). The improved molecules are examined in §4.3 and §4.3.

### 4.2 Generalization Beyond Property Optimization

ChemoGrad extends to three additional task families by swapping the reward (Table 2): functional-group-count rules for editing, Tanimoto similarity for reactions, and METEOR (Denkowski & Lavie, 2014) for description. Molecule editing uses a reference-free verifier and saturates at a perfect edit score (1.00) for add/delete, with substitute rising from 0.71 to 0.89 (accuracy 66.7%→85%). For reaction prediction and captioning we run a separate, clearly labelled *idealized* experiment where the verifier is the benchmark's gold evaluator, unavailable at inference time and thus an analytic upper bound, not a deployment claim. Even with a gold verifier that assigns near-zero reward to almost every candidate — a signal with almost no directional structure — the search still lifts reaction Tanimoto from 0.00 to 0.85 (approaching Gemini-2.5-pro-Thinking's reference 0.89 on forward prediction) and roughly doubles caption quality, all with a drop-in test-time steer for an open chemistry LLM rather than a frontier closed model. A single latent-update loop thus carries across all reward families — property calculators, edit rules, molecular similarity, and free-text metrics — one mechanism rather than a task-specific trick.

**Win Rate Across Benchmarks.** ChemoGrad wins against the CoT baseline on every benchmark and every split, with all four aggregate rates significant at $p<0.01$: the win rate lifts the four All splits from 49.88–85.26% (LatentChem) to 91.7–99.1% (ours). The full per-split breakdown is reported in Table 7. Note the two win-rate definitions are not interchangeable: our $R^{*}_{\text{win}}$ keeps ties in the denominator (a tie counts against a win), whereas Ye et al. (2026) drops ties, so the two numbers should not be read as a same-definition difference; details in Appendix C.

Table 2: Generalization beyond molecule optimization. *init*: greedy CoT baseline (our $\mathbf{B}_{\text{local-greedy}}$) per task family; *final*: ChemoGrad; paper A: LatentChem's reported value. Column *Metric* marks each family's reward: edit is a reference-free functional-group score (↑), rxn is reaction Tanimoto/FTS similarity (↑), and caption is METEOR (↑). For *mol_edit*, paper A reports accuracy (%) rather than the edit score, and both are listed. ChemoGrad exceeds paper A on every row.

| Family | Subtask | Metric | $n$ | init | final | paper A |
|---|---|---|---|---|---|---|
| mol_edit | add | edit score ↑ | 20 | 0.75 | **1.00** | 71* |
| | delete | edit score ↑ | 20 | 0.83 | **1.00** | 84* |
| | sub | edit score ↑ | 60 | 0.71 | **0.89** | 61* |
| * *mol_edit* paper A reports accuracy (%): add 71 / delete 84 / sub 61; ChemoGrad attains 100%/100%/85% accuracy on the same tasks (Table 10). | | | | | | |
| rxn | major_product | Tanimoto/FTS ↑ | 100 | 0.00 | **0.85** | 0.52 |
| | byproduct | Tanimoto/FTS ↑ | 100 | 0.00 | 0.30 | 0.17 |
| caption | Mol-Instr | METEOR ↑ | 1000 | 0.107 | **0.239** | 0.12 |
| | ChEBI-20 | METEOR ↑ | 3297 | 0.096 | **0.207** | 0.15 |
| | ChemLLM | METEOR ↑ | 100 | 0.105 | **0.207** | 0.16 |

### 4.3 Analysis: Where the Gain Comes From

**Scaling test-time search depth improves performance.** Test-time performance improves with the number of latent update iterations, suggesting a test-time-compute scaling effect in chemistry (Snell et al., 2024): performance climbs monotonically until the plateau circuit-breaker terminates at a mean of 8.7 steps. Because the loop only accepts a candidate when it improves the reward, the best-so-far metric cannot decrease by construction — the increase reflects the search finding better candidates, not a property of any specific update direction.

To isolate what the search adds over best-of-$N$ resampling, we run one progressive chain on a *vanilla* Qwen-3-8B-Instruct backbone with no chemistry domain SFT, so the probe is complementary to (not a substitute for) the main-table comparison against LatentChem. On *logp* ($n$=100): greedy decoding reaches $\Delta$=0.003 (18% SR), selecting the best of 8 sampled candidates lifts this to $\Delta$=0.368 (67% SR), and the *full mechanism* on the same starting pool pushes it to $\Delta$=0.534 (89% SR, 89% valid). This is not a resampling curve but a *mechanism split at a fixed budget*: the pool is the same 8 samples at every rung, and only the optimization variable changes. Best-of-8 is thus the resampling wall — the ceiling reachable by selecting among 8 draws rather than moving a latent — and the step beyond it is the gradient margin we claim: a further +0.166 in $\Delta$ and +22 points in success rate that sampling alone structurally cannot recover, because by definition it never proposes a candidate outside the pool it has already drawn.

**Cumulative ablation: the loop and its deepening drive the gains.** Table 3 decomposes each mechanism cumulatively on *logp* and *solubility*. The base loop alone (C) lifts *logp* modestly; the full ChemoGrad adds an enlarged budget, task-prior initialization, and the deepened update, and the two tasks jump to 1.76 and 2.19. Two rows are the exception that reveals the rule. First, $\text{B}_{\text{paper}}$ is the CoT decoding of the same checkpoint quoted from the paper's row, while C→ChemoGrad are our own local arms, so the $\text{B}_{\text{paper}}$→C gap is a reference gap rather than a paired same-start step; the apparent drop on *solubility* (1.17→0.90) is therefore not a within-configuration regression — the untargeted loop (C, no task prior) drifts toward the edits that raise *logp*, the two properties being mirror images (§4.3), and the full mechanism restores the correct direction, which is why the prior is not optional. Second, because the three mechanisms are added together in the full configuration, the increment cannot be attributed to any single one — a per-mechanism, equal-budget ablation is left to future work (§6). On the harder activity tasks the increments concentrate where the reward leaves the most room (§2.5).

**Case studies.** We trace three real *logp* trajectories (Table 4, Figure 3) showing how far the hidden state must move to reach each reward plateau.

Table 3: Mechanism ablation on *logp* and *solubility* ($\Delta$); mechanisms are added cumulatively. $B_{paper}$ is the same checkpoint's chain-of-thought decoding with latent-thinking disabled, quoted from the paper's matching row; C and ChemoGrad are our own local arms. Because $B_{paper}$ is a paper-quoted row rather than a locally re-run arm, the B→C step is a reference gap, not a paired same-start comparison.

| Configuration | Added mechanism | logp | sol |
|---|---|---|---|
| $B_{paper}$ | — | 0.67 | 1.17 |
| C (base loop) | latent optimization loop | 1.04 | 0.90 |
| **ChemoGrad** | enlarged budget, task-prior init, deepened update | **1.76** | **2.19** |

Table 4: Three real *logp* case studies (LogP improvement $\Delta$). *init*: baseline answer; *opt*: final after the latent loop.

| Case | init | opt | iters | dominant |
|---|---|---|---|---|
| Single decisive edit | $+0.33$ | $+4.14$ | 4 | gradient flip |
| Long path-dependent update | $+0.55$ | $+2.37$ | 16 | loop |
| Compound edit | $+0.76$ | $+4.20$ | 8 | loop + deepening |

- **A single decisive edit.** On a benzothiazole urea, the initial answer only attaches a single Br for $+0.33$; the latent update instead breaks the $N$-methylpiperidine tail and re-grows it as an aryl amine with a ring bromination, reaching $+4.14$ in 4 steps (Figure 3, top).
- **A long, path-dependent update.** On a bisaryl urea, the initial answer is a modest $+0.55$, and the loop, over 16 steps, severs the polar aryl-ether linker and couples the two aryl rings, reaching $+2.37$ (Figure 3, middle): the optimizer stabilizes a near-valid intermediate before the coupling move becomes reachable, crossing the reward landscape step by step rather than in one jump.
- **A compound edit.** On an indole-piperidine, the update finds a piperidine→*o*-chloroaniline substitution and extends it, reaching $+4.20$ in 8 steps (Figure 3, bottom).

The three cases span a decisive few-step edit, a gradual multi-step path, and a composed sequence of edits — yet one reward-guided update over the hidden state realizes all of them, local and non-local alike, with no per-case machinery.

A fourth case, from the forward-reaction task under a gold verifier, un-confounds the two mechanisms: on a triflyloxy Sonogashira coupling, best-of-8 still only turns up candidates scoring 0.0 against the true product (init and BoN both 0.0), yet the subsequent latent update reconstructs the exact alkyne-coupled product in 5 steps (Tanimoto 1.0) — resampling cannot manufacture a candidate it never drew, so the gradient margin is not a relabeling of the BoN gain.

**What the optimized molecules look like: edit patterns.** Beyond the three traced cases, the optimized molecules follow a small, task-consistent vocabulary of chemical edits — the same textbook medicinal-chemistry moves the trained baseline had to learn in four stages, but read off the reward surface directly (Table 8):

- **Lipophilicity (*logp*).** Polar handles are swapped for lipophilic ones; *solubility* inverts this — its oracle rewards the opposite direction, so the two tasks are mirror images.
- **Drug-likeness (*qed*).** De-cluttering moves simplify away polar, synthesis-unfriendly groups.
- **Pharmacological activity.** With a classifier oracle the edits are heteroatom moves that raise the classifier's predicted activity.

Two patterns emerge. First, the edits are small and local, so the decoded molecules remain valid and close to the source — a qualitative reading of the exemplar trajectories above rather than a population-wide statistic, corroborated by the high validity in Table 1. Second, the edits are task-aligned: the oracle implicitly encodes the rule for changing the molecule, so reward-guided search surfaces it on its own while the task-prior hint only orients the update. These are observations at the level of the verifier's reward surface, not a claim about on-target binding (§4.1).

## 5 Related Work

**Latent and compressed reasoning.** Test-time latent optimization (Li et al., 2025; Mi et al., 2026), chain-of-thought prompting (Wei et al., 2022; Wang et al., 2022), tree/graph search (Yao et al., 2023; Besta et al., 2024), and latent-token compression (Hao et al., 2024) all internalize reasoning, while LatentChem (Ye et al., 2026) distills CoT into a latent-thinking module for chemistry. Verifier-guided test-time compute (Snell et al., 2024; Muennighoff et al., 2025; Lightman et al., 2024; Cobbe et al., 2021) and planning-based decoding (Feng et al., 2024; Guan et al., 2025; Yao et al., 2023) extend the AlphaZero lineage (Silver et al., 2017; 2018; Schrittwieser et al., 2020). ChemoGrad instead optimizes the hidden states a chemistry LLM already emits, subject to a chemistry-specific constraint: the decoded molecule must remain a valid, chemically meaningful compound throughout the search.

**RL-based molecular optimization.** Preference alignment — RLHF (Ouyang et al., 2022; Bai et al., 2022), PPO (Schulman et al., 2017), GRPO (Guo et al., 2025), and DPO variants (Rafailov et al., 2023; Ethayarajh et al., 2024) — fine-tunes model weights and underlies de novo design: recurrent/variational generators (Gómez-Bombarelli et al., 2018; Segler et al., 2018) and graph generative models (Jin et al., 2018; You et al., 2018; Fu et al., 2022); all operate at the weight level; ChemoGrad instead extracts extra headroom at test time, evaluated under the official *ChemCoT-Bench* oracles (Hao et al., 2026).

## 6 Limitations

The search is only as good as its verifier: a biased or gameable oracle would bias the search, a risk inherited from any reward-guided method. Our gradient *margin* over the BoN wall (§4.3, §4.3) holds only at fixed pool; a full cross-budget re-allocation is future work.

## 7 Conclusion

We proposed **ChemoGrad**, a training-free test-time latent optimizer that searches the hidden states a chemistry LLM already emits, leaving the weights untouched. Steered by verifiable oracles, it improves over a fully trained four-stage baseline on all six *ChemCoTBench* tasks by $+51\%$ mean $\Delta$ with no additional training, and transfers to molecular editing and, under gold verifiers, prediction and captioning — a test-time lever that four-stage post-training leaves unspent.

## References

Yuntao Bai et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

Dávid Bajusz, Anita Rácz, and Károly Héberger. Why is tanimoto index an appropriate choice for fingerprint-based similarity calculations? *Journal of Cheminformatics*, 7, 2015.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. *AAAI*, 2024.

G Richard Bickerton, Gaia V Paolini, Jérémie Besnard, Sorel Muresan, and Andrew L Hopkins. Quantifying the chemical beauty of drugs. *Nature Chemistry*, 4(2):90–98, 2012.

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.

John S Delaney. ESOL: Estimating aqueous solubility directly from molecular structure. *Journal of Chemical Information and Computer Sciences*, 44(3):1000–1005, 2004.

Michael Denkowski and Alon Lavie. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pp. 376–380, 2014.

Kawin Ethayarajh et al. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*, 2024.

Yin Fang, Xiaozhuan Liang, Ningyu Zhang, Kangwei Liu, Rui Huang, Zhuo Chen, Xiaohui Fan, and Huajun Chen. Mol-Instructions: A large-scale biomolecular instruction dataset for large language models. *ICLR*, 2024.

Xidong Feng, Ziyu Wan, Muning Wen, Jun Ying, Wenbin Li, Weinan Zhang, et al. Alphazero-like tree-search can guide large language model decoding and training. *ICML*, 2024.

Kaifeng Fu, Wenhao Gao, Connor W Coley, et al. Differentiable scaffolding tree for molecular optimization. *ICLR*, 2022.

Arup K Ghose and Gordon M Crippen. Atomic physicochemical parameters for three-dimensional structure-directed quantitative structure-activity relationships i. partition coefficients as a measure of hydrophobicity. *Journal of Computational Chemistry*, 7(4):565–577, 1986.

Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamin Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4(2):268–276, 2018.

Xinyu Guan, Li Lyna Zhang, Yijiang Liu, et al. rstar-math: Small llms can master math reasoning with self-evolved deep thinking. *arXiv preprint arXiv:2501.04519*, 2025.

Daya Guo, Dejian Yang, Haowei Zhang, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Xiaohan Han et al. Extracting and tuning molecular instructions in LLMs. *arXiv preprint*, 2025.

Lin Hao, Han Cao, Bing Feng, et al. Beyond chemical qa: Evaluating llm's chemical reasoning with modular chemical operations. *Advances in Neural Information Processing Systems*, 38, 2026.

Shibo Hao, Yuxian Gu, Haodi Ma, Joshua Jiahui Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.

Kexin Huang, Tianfan Fu, Wenhao Gao, Yue Zhao, Yusuf Roohallah, Yusuf Sun, Haley Leskela, and Jirka Zou. Therapeutics data commons: Machine learning datasets and tasks for drug discovery and development. *Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. *ICML*, 2018.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Hengli Li, Chenxi Li, Tong Wu, Xuekai Zhu, Yuxuan Wang, Zhaoxin Yu, Eric Hanchen Jiang, Song-Chun Zhu, Zixia Jia, Ying Nian Wu, and Zilong Zheng. Seek in the dark: reasoning via test-time instance-level policy gradient in latent space. *arXiv preprint arXiv:2505.13308*, 2025.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Karl Cobbe, et al. Let's verify step by step. *ICLR*, 2024.

Yapeng Mi, Yanpeng Zhao, Hengli Li, Chenxi Li, Huimin Wu, Xiaojian Ma, Song-Chun Zhu, Ying Nian Wu, and Qing Li. Milr: Improving multimodal image generation via test-time latent reasoning. *ICLR*, 2026.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candes, and Tatsunori Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20286–20332, 2025.

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. *NeurIPS*, 2022.

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. *Advances in Neural Information Processing Systems*, 2023.

David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling*, 50(5):742–754, 2010.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588:604–609, 2020.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Marwin HS Segler, Thierry Kogej, Christian Tyrchan, and Mark P Waller. Generating focused molecule libraries for drug discovery with recurrent neural networks. *ACS Central Science*, 4(1): 120–131, 2018.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550:354–359, 2017.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362:1140–1144, 2018.

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

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *ICLR*, 2022.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 2022.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8, 1992.

An Yang et al. Qwen3 technical report. *arXiv preprint*, 2025.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *NeurIPS*, 2023.

Xinyu Ye, Yujie Mao, Yi Liao, et al. LatentChem: From textual cot to latent thinking in chemical reasoning. In *arXiv preprint arXiv:2602.07075*, 2026.

Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. *NeurIPS*, 2018.

## AI USE STATEMENT

Generative AI tools assisted with code drafting for the optimization pipeline and with proofreading prose. All AI-assisted code was verified and tested for correctness; all experimental results were independently reproduced and validated against the official benchmark metrics. The authors take responsibility for the final content.

### REPRODUCIBILITY STATEMENT

ChemoGrad requires no training, and all rewards use public oracles (RDKit, TDC, *nltk* METEOR) on the benchmark-provided test sets. The loop, configurations, and hyperparameters are in §2 and §3.1; evaluation follows the official *ChemCoTBench* protocol (Eq. 6). We will release anonymized code, configs, and per-sample result JSONs for exact reproduction.

## A HYPERPARAMETER AND CONFIGURATION DETAILS

Table 5 lists the full ChemoGrad configuration. All tasks share the base loop; only the reward and the task-prior prompt suffix differ.

Table 5: ChemoGrad hyperparameters.

| Component | Parameter | Value |
|---|---|---|
| Base loop | iterations $K$ | 30 |
| | stop threshold $\tau_{\text{stop}}$ | $-0.15$ |
| | token ratio $\rho$ | 0.5 |
| | learning rate $\eta$ | 0.05 (Adam) |
| | max new tokens | 512 |
| | plateau circuit-breaker | 20 flat steps |
| Deepening the update | max rounds | 3 |
| | learning-rate schedule | $\eta \cdot 2^n$ |
| | trigger | $\text{imp} < \text{RESTART_TARGET}$ after base loop |
| Task-prior init | form | task-specific suffix on query |

## B PER-TASK REWARD INSTANTIATIONS

**Property (*mol_opt*, *avail*).** $\text{imp} = \text{prop}(\hat{m}) - \text{prop}(m^{\text{src}})$; reward $r = -1 + \sigma(4\,\text{imp}/\text{target} - 2)$, a sigmoid in imp scaled by the task's success threshold target: no improvement ($\text{imp} = 0$) yields $r = -1 + \sigma(-2) \approx -1$, the largest update; a half-target improvement yields $r = -0.5$; a full-target improvement saturates toward $r = -1 + \sigma(2) \approx -0.12$, the smallest update. Invalid SMILES $\rightarrow$ reward $-1$. The per-task target is 0.5 for *logp*/*solubility* and 0.05 for *qed*/*drd2*/*jnk3*/*gsk3*$\beta$. The deepening trigger threshold ("*anchor-restart*" in the code) is the same quantity as target and is measured in imp units, not reward units: a run enters the deepened rounds when its cumulative imp (not its reward $r \in [-1, 0]$) still falls short of $\text{RESTART_TARGET} = \text{target}$ after the base loop. Crucially this "restart" does not re-initialize the hidden state, in agreement with §2.3: it re-enters the update with a doubled learning rate from the current latent state, indexed as a new "round" only. Table 6 lists the six task-prior prompts verbatim; *mol_edit*/*molecule-understanding*/*rxn* (*closed*), *ChemLLMBench*, *Mol-Instructions*, and captioning each use their own task-family hint with the same "output only the ⟨answer⟩" format, and the full 25-prompt set is released with the code.

**Edit (*mol_edit*, *avail*).** Compares functional-group counts between source, target, and output; reward is the negative total count mismatch, normalized to $[-1, 0]$.

**Reaction (*rxn*, *gold*).** Tanimoto similarity (Morgan fingerprints (Rogers & Hahn, 2010)) between predicted and ground-truth product; for multi-component tasks (retrosynthesis, reagent), the mean over component-wise best matches.

**Caption (*gold*).** $\text{METEOR}(\hat{y}, y^{\text{ref}}) - 1 \in [-1, 0]$ via *nltk.translate.meteor_score*, identical to the benchmark evaluator.

Table 6: Task-prior initialization prompts for the six *mol_opt* tasks, reproduced verbatim. Each is appended to the query as "`q + newline newline + prior + newline newline`" before the first reward-guided update.

| Task | Prior hint |
|---|---|
| *logp* | to increase LogP, make the minimal modification with maximum lipophilicity gain: add halogen (F/Cl/Br) or alkyl substituent to an aromatic ring, or replace a polar group (OH, NH2) with a non-polar one. |
| *solubility* | to increase solubility, make the minimal modification with maximum hydrophilicity gain: add ionizable groups (COOH, SO3H, NH2) or increase hydrogen-bond donors/acceptors. |
| *qed* | to increase QED (drug-likeness), make the minimal modification improving the balance of molecular weight, lipophilicity and polar surface: remove heavy substituents or replace them with drug-like fragments. |
| *drd2* | to increase DRD2 activity, make the minimal modification favoring dopamine-receptor-binding motifs: aromatic ring with a basic amine at the right distance. |
| *jnk3* | to increase JNK3 activity, make the minimal modification favoring kinase-hinge-binding motifs: aromatic heterocycles with H-bond acceptors. |
| *gsk3β* | to increase GSK3b activity, make the minimal modification favoring kinase-inhibitor motifs: flat aromatic systems with H-bond acceptor/donor pairs. |

## C Additional Quantitative Results

**Win rate (full breakdown).** ChemoGrad wins against the CoT baseline on every benchmark and every split (Table 7), following the protocol of Table 3 in Ye et al. (2026). For each paired sample we count a win if ChemoGrad improves over the baseline, a loss if it is worse, and a tie if the two are equal; $R^*_{\text{win}} = N_{\text{win}}/(N_{\text{win}}+N_{\text{loss}}+N_{\text{tie}})$, i.e. a win rate in which a tie counts as a non-win rather than being dropped from the denominator. We mark $*$ when a two-sided sign test rejects the null at $p<0.01$.

Table 7: Win rate $R^*_{\text{win}}$ (%) vs. the CoT baseline, by benchmark and split, extending Table 3 of Ye et al. (2026). $*$: sign test $p<0.01$; $n$: paired samples in the All split (ours).

| Benchmark | Method | Open | Closed | All | $n$ |
|---|---|---|---|---|---|
| *ChemCoTBench* | LatentChem | 63.73 | 50.00 | 59.88 | — |
| | **ChemoGrad (ours)** | — | **91.7**$^*$ | **91.7**$^*$ | 761 |
| Mol-Instructions | LatentChem | 72.00 | 48.36 | 49.88 | — |
| | **ChemoGrad (ours)** | **98.1**$^*$ | **97.2**$^*$ | **97.5**$^*$ | 3000 |
| *ChemLLMBench* | LatentChem | 87.37 | 52.87 | 55.58 | — |
| | **ChemoGrad (ours)** | **98.0**$^*$ | **99.5**$^*$ | **99.1**$^*$ | 315 |
| *ChEBI*-20 | LatentChem | 85.26 | — | 85.26 | — |
| | **ChemoGrad (ours)** | **99.0**$^*$ | — | **99.0**$^*$ | 3297 |

**Edit vocabulary (full table).** The optimized molecules follow a small, task-consistent vocabulary of chemical edits (Table 8); each edit is the textbook medicinal-chemistry move that the task's external property oracle encodes, surfaced by reward-guided search rather than learned from the four-stage baseline's priors.

**Run-to-run robustness.** Re-running the optimization on *logp*/*solubility* ($n$=100 each, three seeds) confirms the headline result is stable: *logp* $\Delta$ spans 1.739–1.759 and *solubility* 2.186–2.255, a spread of $< 2.5\%$ of the mean. All runs share the same base loop ($\rho$=0.5, $\eta$=0.05, $K$=30, $\tau_{\text{stop}}$=$-0.15$, BoN over 8 starts at temperatures 0.6–1.6). The deterministic reward oracle is evaluated identically across runs; the conclusions are unchanged.

Table 8: Edit vocabulary of ChemoGrad across the six *mol_opt* tasks: the transformation applied to the source, an illustrative SMILES pair, and the oracle that induces it.

| Task | Edit pattern | Source → Output | Oracle |
|---|---|---|---|
| *logp* | polar→lipophilic | *Cc2ccc(O)cc2→Cc2ccc(Cl)cc2* | Crippen LogP |
| *solubility* | lipophilic→polar | *N(C)(C)→NH₂* | ESOL |
| *qed* | de-cluttering | *[N+](=O)[O-]→-H* | QED |
| *drd2/jnk3/gsk3β* | heteroatom moves raising classifier activity | *2-NH→4-NH* on pyridine | TDC classifier |

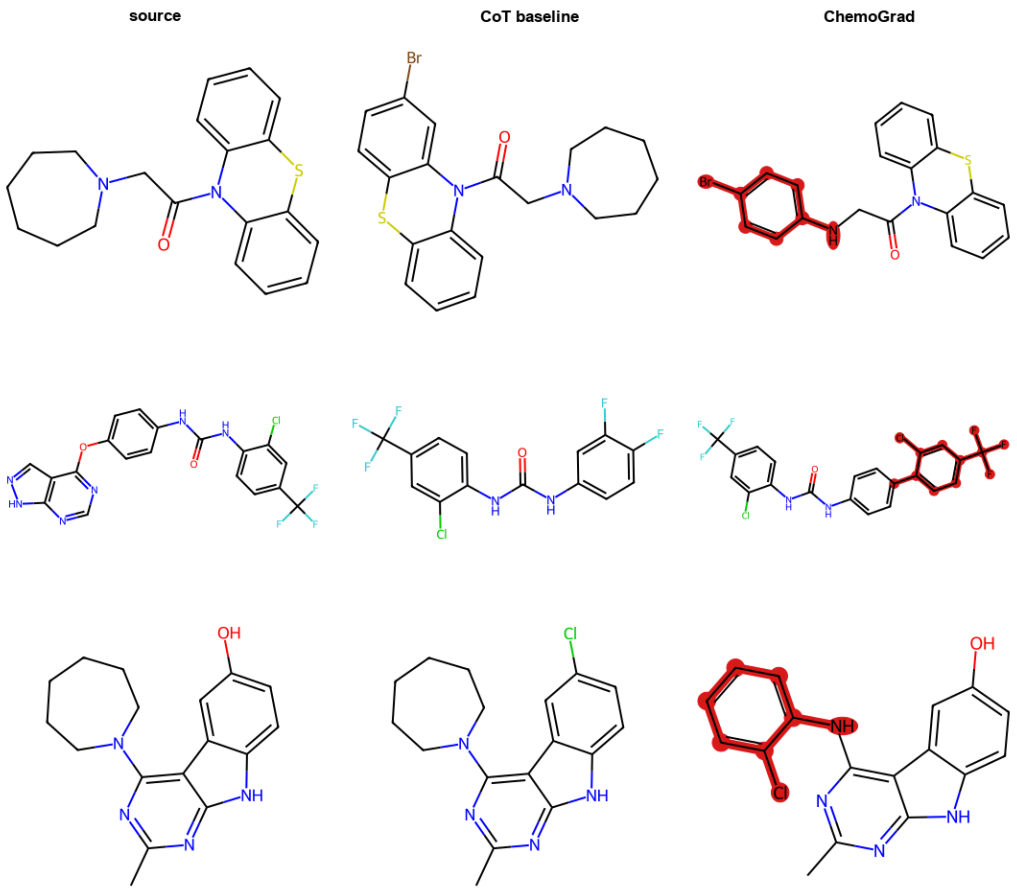


Figure 3: Representative optimized structures: source, CoT baseline, and ChemoGrad output (edited atoms in red) for three *logp* cases.

**Per-sample provenance of the paired tests.** The paired permutation tests pair ChemoGrad's per-molecule output against a baseline per-molecule output on the *same* source molecule. For *mol_opt*, the CoT baseline $\mathbf{B}_{\text{local-greedy}}$ is our batch-decoded greedy run of the same checkpoint with latent-thinking disabled, so every sample is pair-aligned by index (this paired reference is separate from the $\mathbf{B}_{\text{paper}}$ row shown in Table 1, which quotes LatentChem's published numbers). For *mol_edit*/molecule-understanding/retrosynthesis and *ChemLLMBench*, where no independent batch baseline was collected, the pair is the method's own greedy initial answer versus its final answer (an init-to-final pairing); for *Mol-Instructions* and *ChEBI-20* a separately batch-decoded $\mathbf{B}_{\text{local-greedy}}$ baseline is used. Because the optimizer accepts a candidate only when the oracle reports an improvement, an init-to-final pairing is guaranteed to favor the final answer; it therefore proves the search engages the reward, but does not by itself compare ChemoGrad against a competing search strategy — that comparison is limited to the rows where an independent batch baseline ($\mathbf{B}_{\text{local-greedy}}$) or the paper-reported A mean is available, and to the fixed-budget mechanism split in §4.3. The

trained LatentChem A values are paper-reported means with no per-sample results, so no paired test against A is possible; the "A" column in Table 9 only reports whether ChemoGrad's sample mean exceeds the paper-reported A mean point-wise, and is not a significance claim against A (A's sampling uncertainty is unknown and not propagated). All per-sample result files and the scoring scripts are released for exact reproduction.

**Paired significance (main table).** A paired permutation test with bootstrap $95\%$ confidence intervals (Table 9) confirms the gains against the local CoT baseline $\mathbf{B}_{\textbf{local-greedy}}$ are significant ($p<0.01$) on all six tasks. On the three physical-property tasks the $95\%$ confidence intervals on the paired difference exclude zero throughout (*logp* $[+0.197, +0.476]$, *solubility* $[+0.540, +0.934]$, *qed* $[+0.141, +0.207]$), and do so on the activity tasks as well (*drd2* $[+0.203, +0.332]$). Against the intermediate configuration $\mathrm{C}^+$ (the loop with an enlarged budget and task-prior init but no deepening), the increment is significant on the three physical-property tasks (*logp*/*solubility*/*qed*). On the activity tasks the deepening-plus-prior increment is not significant for *drd2*/*jnk3*, and on *gsk3*$\beta$ $\mathrm{C}^+$ is significantly *below* ChemoGrad ($-$sig$*$), i.e. deepening is what closes that gap: without it, the extra budget actually leaves *gsk3*$\beta$ short of the activity classifier's ceiling, so deepening contributes a measurable positive increment there. The remaining non-significance on *drd2*/*jnk3* is consistent with the mechanism analysis in §2.5: the activity classifiers already reward $\mathrm{C}^+$'s update near the practical ceiling, leaving the extra leverage concentrated on the physical-property tasks.

Table 9: Paired significance (permutation test, bootstrap $95\%$ CI) of ChemoGrad's mean $\Delta$ vs. the batch baseline (B) and the intermediate configuration $\mathrm{C}^+$. $**$: $p<0.01$; $*$: $p<0.05$; ns: not significant; $-$sig$*$: significant in the *negative* direction ($p<0.05$). The A row is not a significance test: it only checks whether ChemoGrad's sample mean exceeds LatentChem's paper-reported mean (no per-sample A data exist, so A's variance is unaccounted); ✓ / — mark exceedance or not, not a hypothesis test.

| Contrast | logp | sol | qed | drd | jnk | gsk |
|---|---|---|---|---|---|---|
| ChemoGrad vs. B | $**$ | $**$ | $**$ | $**$ | $**$ | $**$ |
| mean vs. A | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| ChemoGrad vs. $\mathrm{C}^+$ | $*$ | $**$ | $**$ | ns | ns | $-$sig$*$ |

**Closed-form tasks and ChemLLMBench (full breakdown).** On the closed-form ChemCoTBench subtasks (Table 10), ChemoGrad matches or exceeds the CoT baseline on all 7 subtasks, with the largest gains on ring-system recognition ($98.3\%$) and ring counting (error halved): molecule editing saturates at $100\%$ (add/delete) and $85\%$ (substitute), and Murcko-scaffold similarity climbs from $0.70$ to $0.88$ Tanimoto. On ChemLLMBench (Table 11) ChemoGrad lifts ligand and solvent selection substantially ($+0.37$/$+0.37$ quality); retrosynthesis and multi-component reagent prediction, where the baseline itself hovers near chance, track the same difficulty ChemoGrad inherits from the underlying model, consistent with the observation in Mol-Instructions (Fang et al., 2024; Han et al., 2025) reagent prediction.

Table 10: Closed-form ChemCoTBench subtasks: CoT baseline ($\mathbf{B}_{\textbf{local-greedy}}$, our greedy decode) vs. ChemoGrad, with LatentChem's reported value (paper A) as an external reference. Top block reports accuracy (%), higher is better; bottom block reports error and similarity metrics, with the direction marked per row. ChemoGrad matches or exceeds both baselines on every subtask.

| Subtask | Metric | $n$ | B | ChemoGrad | paper A |
|---|---|---|---|---|---|
| mol_edit/add | acc ↑ | 20 | 70.0 | **100.0** | 71 |
| mol_edit/delete | acc ↑ | 20 | 90.0 | **100.0** | 84 |
| mol_edit/sub | acc ↑ | 60 | 66.7 | **85.0** | 61 |
| mol_und/Murcko | Tanimoto ↑ | 40 | 0.70 | **0.88** | 0.81 |
| mol_und/fg_count | MAE ↓ | 100 | 0.05 | **0.05** | 0.09 |
| mol_und/ring_count | MAE ↓ | 20 | 0.15 | **0.05** | 0.15 |
| mol_und/ring_sys | CR ↑ | 60 | 78.3 | **98.3** | 96.7 |

Table 11: ChemLLMBench subtasks: baseline ($\mathbf{B}_{\textbf{local-greedy}}$) vs. ChemoGrad, quality (Q) and accuracy (acc).

| Subtask | B | ChemoGrad |
|---|---|---|
| ligand (Q) | 0.34 | **0.71** |
| solvent (Q) | 0.63 | **0.99** |
| retro (Q) | 0.29 | **0.54** |
| captioning (METEOR) | 0.07 | **0.21** |