FORGE: Fragment-Oriented Ranking and Generation for Context-Aware Molecular Optimization

Anonymous Authors[1]

Abstract

Molecular optimization seeks to improve a molecule through small structural edits while preserving similarity to the starting compound. Recent language-model approaches typically treat this task as prompt-conditioned sequence generation. However, relying on natural language introduces an inherent data-scaling bottleneck, often leads to chemical hallucinations, and ignores the strong context dependence of fragment effects. We present FORGE, a two-stage framework that reformulates molecular optimization as context-aware local editing. By utilizing automatically mined, verified low-to-high edit pairs instead of expensive human text annotations, Stage 1 ranks candidate fragments by their property contribution under the full molecular context to inject chemical prior, and Stage 2 generates explicit fragment replacements. Built on a compact 0.6B language model, FORGE further adapts to unseen black-box objectives through in-context demonstrations. Across Prompt-MolOpt, PMO-1k and ChemCoTBench, FORGE consistently outperforms prior methods, including substantially larger language models and graph methods. These results highlight the value of explicit fragment-level supervision as a more easily obtainable, scalable, and hallucination-less alternative to natural language training.

1. Introduction

Molecular optimization aims to improve a starting molecule toward a desired property, often under a strict structural similarity constraint (Jin et al., 2018a; You et al., 2018; Dara et al., 2022). Unlike open-ended molecular generation, molecular optimization is fundamentally a *local editing* problem: the goal is not to sample an arbitrarily high-scoring molecule, but to improve a lead compound through small structural modifications (Jin et al., 2018b; 2019). This local-edit regime underlies many practical drug-discovery workflows (Zhang et al., 2024; Ferreira & Carneiro, 2025) as well as standard benchmarks such as similarity-constrained optimization and black-box oracle-guided search (Gao et al., 2022; Brown et al., 2019).

Chemistry language models have made SMILES generation a common interface for molecular design (Chithrananda et al., 2020; Guevorguian et al., 2024; Yu et al., 2024), and most recent optimization methods follow the same pattern: given an input molecule and a textual description, decode an improved molecule. We argue that this formulation is poorly matched to the local-edit setting, in two specific ways: **(i) Semantic mismatch and weak prompt control**: Real-world drug discovery oracles (e.g., ADMET classifiers, proprietary binding scores) lack faithful natural-language descriptions, resulting in weak prompt control. Training on LLM-distilled chain-of-thought data may partially fill the gap, but tends to inherit known chemical inaccuracies from the teacher model. **(ii) Ignored context dependence**: Fragment contributions depend strongly on their local environment. Representing substructure attribution as a global, context-free scalar collapses critical structural information and introduces label noise that propagates to training.

We therefore reformulate molecular optimization as *context-aware local editing*: Rather than asking a model to rewrite the whole molecule from a property prompt, we decompose optimization into two explicit decisions: *where* to edit and *how* to edit it. We instantiate this idea in **F**ragment-**O**riented **R**anking and **GE**neration (**FORGE**), a two-stage framework built on a compact 0.6B language model (Qwen3-0.6B (Yang et al., 2025)) equipped with atom-level tokenization for stable fragment identity. During training, FORGE relies on rule-based data rather than LLM distillation. Stage 1 localizes edits by ranking fragments based on their context-conditioned contribution, and Stage 2 generates explicit fragment replacements. Stage 2 supervision combines rule-based and predictor-based attribution signals with **SME+**, our context-conditioned extension of **S**ubstructure **M**ask **E**xplanation (SME) (Wu et al., 2023) for fragment attribution and pair construction. At inference, FORGE adapts to unknown black-box oracles through in-context demonstrations.

Two empirical observations support this design and are de-

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Submitted to the 2026 Workshop on Generative and Agentic AI for Biology (ICML 2026). Do not distribute.

tailed in Section 3: replacing the true target property in a PMO prompt with an unrelated or unknown name changes the optimization score by less than 3%, and conditioning fragment attributions on local context reduces attribution variance by 8%–10% across two independent predictors. Our contributions are as follows:

- We show that context-conditioned, rule-verified fragment edits are viable and highly scalable training data for molecular optimization, removing the need for rare natural-language objectives or expensive LLM-distilled chain-of-thought data.
- We propose **FORGE**, a two-stage framework that decomposes optimization into *where* to edit and *how* to edit, using context-conditioned fragment ranking and explicit fragment replacement generation, together with atom-level tokenization for stable fragment identity.
- We show that FORGE achieves state-of-the-art results across similarity-constrained, black-box, and real-target molecular optimization benchmarks using only a 0.6B model, outperforming larger baselines.

2. Related Work

Fragment-based optimization and substructure attribution. Matched molecular pair analysis (MMPA) (Dalke et al., 2018) treats local edits as the basic unit of medicinal chemistry, but its core assumption that fragment effects are only weakly context-dependent breaks down at activity cliffs (Van Tilborg et al., 2022). Deep-learning variants such as JTNN (Jin et al., 2018a), GCPN (You et al., 2018), Modof (Chen et al., 2021), Graph Polish (Ji et al., 2021), Prompt-MolOpt (Wu et al., 2024), and fragment-library methods including fRAG (Lee et al., 2024) and GenMol (Lee et al., 2025) still hard-code fragment edits as graph operators or external vocabularies, typically averaging fragment statistics across contexts. In parallel, substructure attribution methods such as SME (Wu et al., 2023), fragment-level Shapley values (Roth, 2026), B-XAIC (Proszewska et al., 2025), and fragment-wise attribution (Musiał et al., 2025) improve the reliability of fragment importance estimates. Our method differs from these lines of work: we promote fragments to first-class editing units and make their contribution explicitly context-conditional; correspondingly, SME+ keeps the attribution mechanism of SME but changes the downstream aggregation scheme from global min–max scaling to within-context binning.

Chemistry language models and inference-time adaptation. SMILES-based pretraining and instruction tuning have established language modeling as a competitive route for molecular modeling, from ChemBERTa (Chithrananda et al., 2020) and GP-MoLFormer (Ross et al., 2025) to ChemLactica (Guevorguian et al., 2024), ChemFM (Cai et al., 2025), and LlaSMol (Yu et al., 2024); related efforts revisit pretraining protocols (Chitsaz et al., 2025) or latent-space reasoning (Ye et al., 2026). Our approach departs from this line in two ways: we use atom-level tokenization rather than BPE (Section 4.3), and we explicitly decompose optimization into discrimination and generation, allowing a 0.6B model to match or outperform 8B reasoning baselines and GPT-4o (Hurst et al., 2024). More broadly, our inference procedure is motivated by recent progress in in-context learning (ICL) for chemistry, where methods like MOLLEO(Wang et al., 2024) and DemoDiff (Liu et al., 2025) show strong gradient-free adaptation. Building on these insights, our method relies on dynamic in-context learning throughout the search process to adapt to unseen oracles. FORGE conditions on the target objective only through demonstrations sampled at search time, avoiding costing gradient-based test-time updates.

3. Motivation Analysis

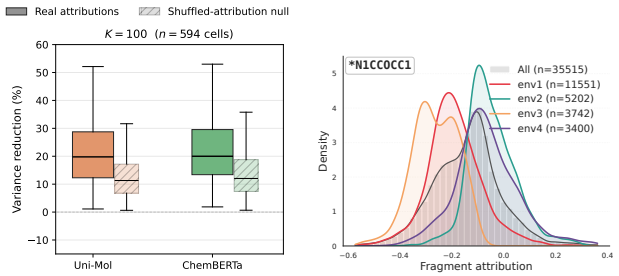
We motivate FORGE with two observations. First, in recent LLM-based optimization pipelines, the target objective might only be weakly determined by its natural-language description. Second, fragment effects are strongly context-dependent, making context-free fragment statistics a poor supervision signal for local editing. Figure 1 summarizes both effects.

3.1. Prompts weakly specify black-box objectives

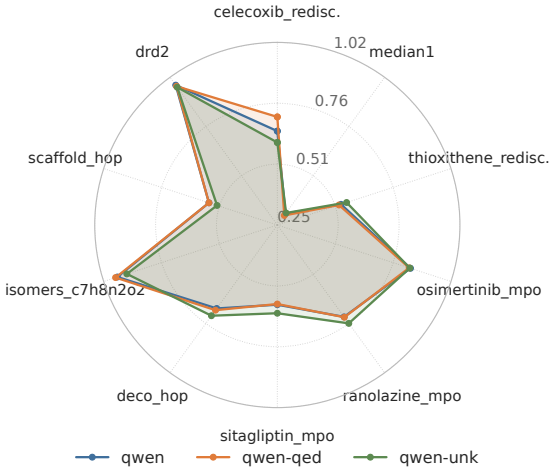
If prompt semantics reliably control molecular optimization, then changing the property description should alter optimization behavior. We test this on the 22 PMO tasks by running Qwen3-30B-A3B (Yang et al., 2025) inside the MOLLEO search loop (Wang et al., 2024) under three prompts: the default task-specific prompt, a mismatched prompt that always asks to optimize ‘QED’, and a prompt that asks to optimize an ‘unknown property’. We keep the model, search loop, decoding, and oracle fixed, changing only the prompt description. The result is stable: replacing the true objective with an unrelated or unknown description changes the aggregate PMO score by less than 3%, and on 13 of 22 tasks the per-task difference stays within ± 0.05. As shown in Figure 1c, prompt text alone is therefore a weak control signal for PMO-style black-box optimization. In this setting, useful task information should instead come primarily from oracle feedback and demonstrations.

3.2. Fragment effects depend on molecular context

Prompt text alone does not tell us what the model should condition on. A natural alternative is the fragment identity itself, but we find that fragment utility depends strongly on its molecular context: the same substructure can help in one molecule and hurt in another.



(a) Context grouping reduces attribution variance. *(b)* Context-dependent fragment effects of ESOL.



(c) Weak prompt grounding on PMO.

Figure 1. **Why prompt-only generation is insufficient for molecular optimization. (a)** Grouping fragment attributions by ECFP neighborhoods reduces attribution variance across two distinct property predictors, indicating that fragment effects depend on chemical context. **(b)** The same fragment can have different effects across host molecules. **(c)** On PMO, replacing the true target property in the prompt with an unrelated or unknown one barely changes the optimization behavior of Qwen3-30B-A3B, indicating weak prompt grounding of molecular objectives.

We test this through fragment attribution. Let $f : \mathcal{X} \to \mathbb{R}$ be a scalar property predictor or oracle, and let a molecule x be decomposed into fragments $\{g_1, \ldots, g_K\}$. For a fragment g in host molecule x, we define its attribution as $a(g; x) = f(x) - f(x \setminus g)$, where $x \setminus g$ is obtained either by masking g (for GNN predictors) or by explicit deletion and recomputation (for rule-based properties). This quantity is a local perturbation proxy rather than a causal effect estimate, but it is enough to ask whether fragment utility varies systematically across host contexts. To quantify the contribution of local context, we group the occurrences of each fragment by its local Extended-Connectivity Fingerprints (ECFP) environment $e_r(g; x)$ at the attachment atoms and measure the resulting variance reduction,

$$\mathrm{VR}(g) = 1 - \frac{\sigma_{\text{grouped}}}{\sigma_{\text{original}}}. \tag{1}$$

A positive $\mathrm{VR}(g)$ means that conditioning on local context reduces the spread of the fragment's observed effects.

Table 1 reports the VR result for Uni-Mol (Zhou et al., 2023) and ChemBERTa (Chithrananda et al., 2020) trained on MoleculeNet(Wu et al., 2017), with groups shuffled randomly as contrast. The pattern is consistent across settings: context grouping always reduces attribution variance, and the gap against the shuffled grouping remains 8%–11%. Figure 1(a,b) shows the same effect.

The choice of neighborhood radius controls how finely context is partitioned. Table 2 sweeps r=1 to r=4 on ChemBERTa: the genuine signal Δ grows monotonically with r, but larger radii also fragment the data into many more environment cells (avg. envs grows from 7.5 to 493.5), leaving too few samples per cell for downstream pair construction. We use r=2 throughout the rest of the paper as a practical trade-off between decoupling strength and per-cell sample size; per-property breakdowns are in Appendix A. Extrapolating this monotonic trend leads to a natural theoretical limit: $r \to \infty$, where the context is the entire host molecule. While explicit ECFP binning is impossible at this limit due to data sparsity, a language model natively attends to the full molecular sequence. This directly motivates our use of an LLM for Stage 1 fragment ranking, as it implicitly captures this global context without being bottlenecked by discrete cell fragmentation.

Table 1. Variance reduction from ECFP2 grouping fragment attributions by local environment, compared against random shuffled grouping. Top K refers to the K most frequently occurring fragments in the dataset. The gap Δ = real − shuf isolates genuine context dependence.

Model	Top K	Real VR	Shuf. VR	Δ	Real>Shuf.
Uni-Mol	30	15.82%	7.59%	8.23%	96.7%
Uni-Mol	100	21.05%	12.43%	8.62%	89.9%
Uni-Mol	300	28.29%	19.08%	9.20%	77.7%
ChemBERTa	30	17.28%	8.86%	8.42%	99.4%
ChemBERTa	100	22.79%	13.92%	8.88%	91.4%
ChemBERTa	300	29.71%	19.84%	9.87%	79.4%

Table 2. Effect of ECFP radius (r) on fragment decoupling for ChemBERTa (top-100 fragments per property). VR and the gap Δ are defined as in Table 1. *Nfrags* denotes the total number of fragments, and *Avg. envs* is the average number of unique local environments per fragment.

r	Nfrags	Avg. envs	Real VR	Shuf. VR	Δ
1	480	7.5	4.57%	2.54%	2.03%
2	594	104.1	22.79%	13.92%	8.88%
3	600	322.6	37.87%	24.63%	13.23%
4	600	493.5	48.54%	31.62%	16.91%

3.3. What these observations imply for model design

The evidence above points to a misalignment in current chemistry LLMs: prompt semantics are not enough to steer

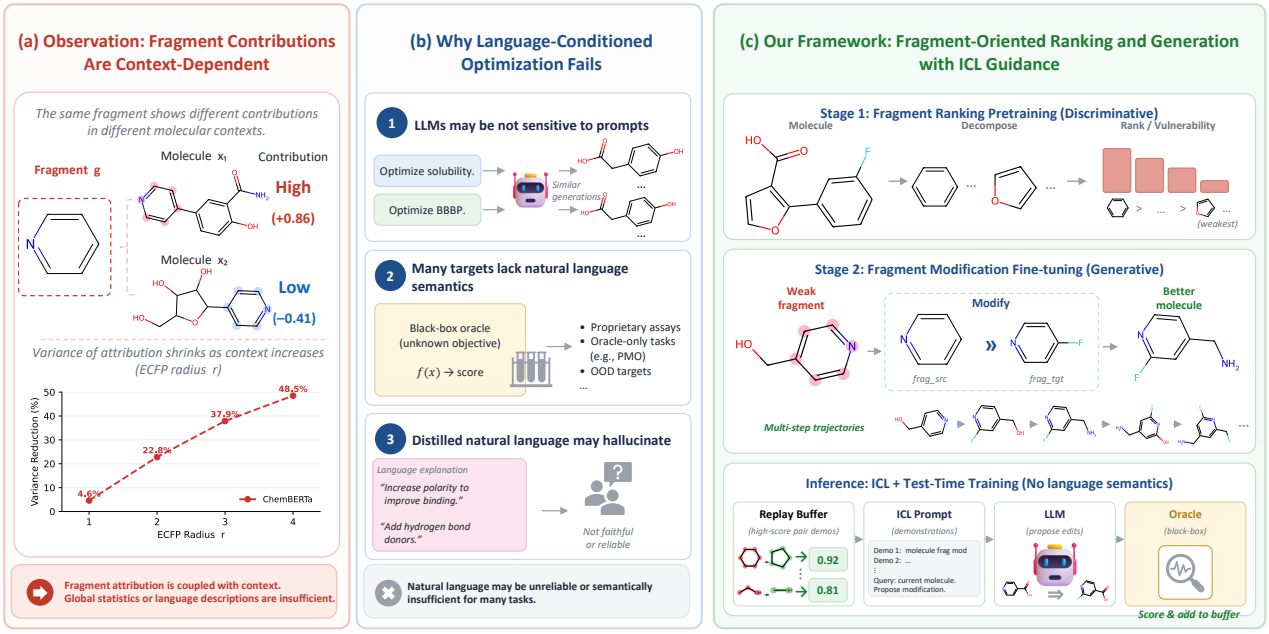


Figure 2. **Overview of the FORGE framework.** **(a)** Left: Empirical observations reveal that fragment contributions are highly context-dependent. **(b)** Middle: Natural language is a suboptimal interface for optimization due to weak prompt control, lack of oracle semantics, and distillation hallucinations. **(c)** Right: The FORGE pipeline decomposes the task into context-aware fragment ranking (Stage 1) and explicit modification (Stage 2), guided by dynamic in-context learning (ICL) at inference.

black-box optimization, and context-free fragment statistics discard the structural dependencies revealed by our VR analysis. Fragments themselves, however, are natural units of chemical knowledge—functional groups, scaffolds, and matched-pair edits familiar from medicinal chemistry. Therefore, the remedy is not to abandon fragment-level supervision, but to preserve its host-molecule context. We thus reformulate molecular optimization as *context-aware local editing*.

From this, two principles follow: a model should derive its target semantics dynamically from oracle feedback rather than static text prompts, and it should operate in an explicit, context-conditioned edit space rather than re-decoding the whole molecule.

4. Method: FORGE

FORGE (**F**ragment-**O**riented **R**anking and **GE**neration) realises the context-aware local-editing view from Section 3 as a concrete training and inference pipeline. The model is trained in two supervised stages: Stage 1 ranks fragments under full-molecule context to learn *where*, and Stage 2 generates an explicit replacement for a chosen fragment to learn *how*. An atom-level retokenization scheme keeps fragment identity stable across host molecules so that this fragment-level supervision is well defined. At test time, the same model adapts to unnamed black-box objectives by conditioning on demonstrations sampled from a replay buffer of (edit, score) tuples. We describe each component below; Figure 2 gives an overview. Stage 1 instantiates context-aware where; Stage 2 instantiates context-aware how via SME+ and MMPA; the inference loop instantiates objective-from-feedback.

Both Stage 1 and Stage 2 are trained via supervised next-token prediction on `(instruction, input, output)` triples, computing the loss exclusively on the output span. Across both stages, we use four sources of supervision: GNN attribution, RDKit attribution, SME+, and ChEMBL matched molecular pairs. A detailed summary of the dataset construction is provided in Appendix D.1.

4.1. Stage 1: learning where to edit

Stage 1 is supervised by fragment attribution. For properties with experimental labels but no closed-form evaluator, such as hERG, mutagenicity, lipop, BBBP and ESOL, we use RGCN models with mask-aware readout from SME and define fragment contribution by score drop after masking the fragment. For rule-based properties with exact evaluators, such as QED, logP, SA, and TPSA, we remove the fragment explicitly and recompute the property with RDKit. The first source provides broad learned supervision; the second provides low-noise labels that are exact under the metric.

From these attributions, we build four task families: *decomposition* (split a molecule into atom-mapped fragments), *ranking* (sort fragments by contribution to a target property), *vulnerability prediction* (return the single weakest fragment), and *ICL ranking/vulnerability* (perform the same

outputs from in-context demonstrations, with or without an explicit property name). We also mix in Mol-Instructions (Fang et al., 2023) to keep general chemistry-language ability from being washed out by structured supervision. The full Stage 1 corpus contains about 2M examples; detailed mixtures are given in Appendix D.

This stage makes the edit decision context-aware and injects chemical prior. The model always reads the full molecular sequence, but the supervision remains fragment-level. As a result, Stage 1 does not learn a single global value for a fragment; it learns to judge that fragment in the context of the host molecule and ICL examples.

4.2. Stage 2: learning how to edit

Stage 2 is trained on verified fragment replacement pairs. Its purpose is to map an identified weak region to a concrete local modification.

The first source of Stage 2 supervision is **SME+**, a context-conditioned extension of SME. Standard SME aggregates all observed effects of a fragment into a global range, ignoring the context dependence shown in Section 3. SME+ instead groups fragment occurrences by their local attachment environment, using the ECFP radius-2 neighborhood around the attachment atoms, and searches for replacements that improve the score within that context bin. When an environment bin is too sparse, SME+ falls back to the global pool. All proposed replacements are filtered by atom-map consistency checks, and only verified low-to-high pairs are retained. These pairs are also chained into multi-step trajectories, which later serve as in-context demonstrations during inference.

The second source is matched molecular pairs from ChEMBL (Gaulton et al., 2012). For real-world targets, where no rule-based evaluator exist, so we rely on measured `pchembl`. We extract pairs that share a scaffold, differ at one fragment, and are measured on the same target. Direction is assigned by activity improvement, so each pair defines a low-to-high edit. To avoid leakage into downstream benchmarks, we exclude all pairs whose target description matches DRD2, JNK3, or GSK3-β.

Stage 2 uses the same output template for all sources:

```
Modification: <frag_src> >> <frag_tgt>
Result: <tgt_smi> [(value: s)]
```

All SMILES spans are wrapped in `<start_smiles>...<end_smiles>` so that atom-level tokenization is applied consistently. The score suffix is included in most samples and omitted in the rest, so the model remains robust to whether oracle values are available at inference.

The backbone Stage 2 data merges SME+ trajectories on five ADMET properties (ESOL/ BBBP/ hERG/ lipop/ Mutag) with ChEMBL matched molecular pairs. The resulting checkpoint after Stage 1 and this backbone Stage 2 training is the **backbone** used for all downstream evaluations. Some benchmarks allow a small amount of additional task-specific continuation in the same format; those benchmark-specific details are described in Section 5.

4.3. Atom-level retokenization

Fragment-level supervision only works if the same fragment is represented consistently across molecules. Standard BPE tokenization does not guarantee this: the same SMILES substring may be split differently depending on the surrounding sequence, which breaks the correspondence between fragments and model tokens.

We therefore augment the Qwen3 vocabulary with a small atom-level SMILES lexicon, including `<start_smiles>`, `<end_smiles>`, and `<smi>`-prefixed tokens for atoms in the organic subset, aromatic atoms, two-letter elements, bonds, brackets, and ring digits. Inside marked SMILES spans, tokenization switches to a regex-based atomic parser. The same fragment is therefore tokenized identically regardless of its host molecule.

The new embeddings are warm-started from their literal counterparts in the original vocabulary, and the LM head is initialized the same way. The first forward pass, therefore, matches the original Qwen3 distribution, so training begins from a stable pretrained prior rather than a newly learned tokenizer. We name this design as QwenAtom. In Appendix E, we show that this retokenization improves all 13 held-out Stage 1 ranking sub-tasks over vanilla Qwen3 tokenization.

4.4. Inference under black-box objectives

We maintain a replay buffer containing tuples of source molecule, applied modification, resulting molecule, and oracle score. At each step, the model is prompted with a small set of demonstrations sampled from this buffer, biased toward higher-scoring examples but tempered to preserve diversity. It then generates candidate modifications for the current molecule, the oracle scores the resulting molecules, and the new tuples are written back into the buffer. The buffer thus acts as an external memory of which local edits have worked under the current objective.

The output grammar is the same in Stage 2 and inference. Demonstration-conditioned examples, including no-description settings, are already present during training. The inference problem has the same structure as the training decomposition: the model must infer which fragment to change from full-molecule context and demonstrations, then propose an explicit replacement for that fragment.

5. Experiments

5.1. Experimental setup

All experiments use Qwen3-0.6B as the base model, together with the *QwenAtom* tokenizer from Section 4.3. Unless otherwise stated, the evaluated model is the shared **backbone** obtained from Stage 1 and Stage 2 training. Training uses standard language-model supervision with DeepSpeed ZeRO-1; implementation details and hyperparameters are deferred to Appendix C.

We evaluate FORGE on five benchmarks. Prompt-MolOpt (Wu et al., 2024) for similarity-constrained ADMET optimization, PMO-1k (Gao et al., 2022) for sample-efficient adaptation to 22 unnamed black-box objectives, QED-DRD2 to isolate the effect of explicit fragment modification under a dual-objective similarity constraint, Lead Optimization (Wang et al., 2022) to test real-target problem and ChemCoTBench (Li et al., 2025) for comparison against larger text-only and latent-reasoning models on six optimization tasks.

We additionally verify the tokenizer design in a controlled Stage 1 comparison, reported in Appendix E. Holding data, optimization, and training steps fixed, atom-level tokenization improves all 13 held-out Stage 1 sub-tasks over vanilla tokenization, with the largest gains on ICL and vulnerability prediction. This supports the role of stable fragment identity in the fragment-ranking stage.

5.2. Prompt-MolOpt: similarity-constrained task

Prompt-MolOpt (Wu et al., 2024) requires the model to improve one of five ADMET properties while maintaining a minimum similarity to the source molecule. For this benchmark, we initialize with the core FORGE backbone (comprising QwenAtom, Stage 1 (S1), and Stage 2 (S2)) and perform a brief task-specific fine-tuning (FT) using multi-turn SME+ data. We compare it against two ablations: removing Stage 1, and removing all Stage 1, Stage 2 and the multi-turn fine-tuning (FT on SME+ data with end-to-end SMILES output).

Table 3 shows that the gap between FORGE and Prompt-MolOpt grows as the similarity constraint tightens. At $\delta = 0$, the gain over Prompt-MolOpt is small; at higher thresholds, the gap widens sharply because FORGE degrades much more gracefully. This is the setting where local fragment editing is most useful: staying close to the source is easier through targeted replacements than through full-SMILES rewriting.

The ablations decompose this gain. Removing Stage 2 and the multi-turn continuation hurts performance at all thresholds, indicating that explicit modification is the main source of improvement under similarity constraints. Removing Stage 1 also lowers performance, though more mildly, suggesting that fragment ranking mainly helps identify where to edit. We additionally find that the multi-turn formulation is most useful on longer edit trajectories, where carrying previous edits in context helps maintain a coherent local search direction; a direct comparison against repeated single-turn calls is reported in Appendix B.

Table 3. Prompt-MolOpt performance at 3 similarity thresholds. Best results under each similarity constraint are highlighted in bold.

Model	δ	ESOL	BBBP	hERG	lipop	Mutag	SUM↑
FORGE (QwenAtom + S1 + S2) + Multi-turn FT							
	0	**0.934**	**0.969**	**0.863**	**0.993**	0.828	**4.587**
	0.4	0.780	**0.949**	**0.847**	**0.982**	**0.744**	**4.302**
	0.6	0.376	**0.715**	**0.575**	0.832	**0.384**	**2.882**
QwenAtom + S2 + Multi-turn FT							
	0	0.928	0.941	0.828	0.993	0.784	4.474
	0.4	**0.807**	0.915	0.811	0.979	0.723	4.235
	0.6	**0.400**	0.676	0.566	**0.842**	0.376	2.860
QwenAtom + Single-turn FT							
	0	0.892	0.918	0.803	0.991	0.775	4.379
	0.4	0.750	0.890	0.787	0.980	0.707	4.114
	0.6	0.345	0.682	0.559	0.841	0.359	2.786
Prompt-MolOpt (Wu et al., 2024)							
	0	0.893	0.936	0.857	0.923	**0.857**	4.466
	0.4	0.443	0.527	0.666	0.674	0.400	2.710
	0.6	0.104	0.164	0.257	0.225	0.074	0.824

5.3. PMO-1k: limited oracle call black-box optimization

PMO (Gao et al., 2022) directly tests the setting from Section 3: the objective is black-box, often unnamed, and accessible only through oracle calls. We evaluate under a reduced 1000-call budget rather than the standard 10k budget following LICO (Nguyen & Grover, 2024), which emphasizes sample efficiency. No task-specific continuation is used here; all adaptation comes from the replay-buffer ICL mechanism from Section 4.4.

Table 4. PMO-1k aggregate score, summed over 22 tasks. Per-task results are reported in Appendix B.

Method	22-task SUM ↑
GP-BO (Srinivas et al., 2009)	11.27
Graph GA (Jensen, 2019)	10.90
REINVENT (Olivecrona et al., 2017)	10.68
LICO (Nguyen & Grover, 2024)	11.71
Genetic GFN (Kim et al., 2024)	11.56
Augmented Memory (Guo & Schwaller, 2023)	10.81
MOLLEO (Wang et al., 2024)	11.65
FORGE + ICL	**12.42**

FORGE reaches an aggregate score of 12.42 in Table 4, outperforming LICO by 0.71. The gain is consistent with the motivation of the method: in PMO, adaptation must come from examples and feedback.

We highlight three tasks. JNK3 improves from the best baseline score of 0.409 to 0.611, suggesting that the fragment editing prior learned from leakage-filtered ChEMBL MMPs transfers to hard real-target optimization. Both *median1* and *median2* reach state of the art. *scaffold_hop* also benefits from the fragment-based formulation.

5.4. QED-DRD2: explicit fragment modification

QED-DRD2 isolates a single design decision: whether the model should output a full rewritten molecule or an explicit fragment modification. We compare two variants that share the same backbone, training data, and loss. The only difference is the output grammar during fine-tuning: *end2end* predicts the whole target SMILES, while *modif.* predicts the fragment replacement format used by FORGE.

Table 5. QED-DRD2 success rates at three similarity thresholds. (% success rates)

Method	δ=0.4	δ=0.5	δ=0.6
JTNN (Jin et al., 2018a)	8.62	5.88	2.62
HierG2G (Jin et al., 2020)	14.50	7.88	4.12
Modof (Chen et al., 2021)	24.62	16.62	9.12
Prompt-MolOpt (Wu et al., 2024)	29.00	25.12	17.87
Modofm (Chen et al., 2021)	38.88	27.25	13.00
FORGE + end2end FT	35.15	33.76	27.32
FORGE + modif. FT	**40.00**	**36.76**	**30.41**

Table 5 shows modification variant outperforms the end-to-end variant by 3.0–5.0 points and beats all prior methods at every threshold. The gap is largest at the strictest similarity threshold, where preserving most of the source structure matters most. This supports the central design choice of FORGE: under similarity constraints, explicit fragment replacement is a better output space than free-form generation.

5.5. Goal-directed Lead Optimization on Docking Targets

We test FORGE on real-target docking, the setting closest to drug-discovery practice. Following Wang et al. (2022), given a seed ligand, the task is to generate molecules with improved docking score subject to QED ≥ 0.6, SA ≤ 4, and Tanimoto similarity $\geq \delta$ to the seed (Morgan fingerprints), with $\delta \in \{0.4, 0.6\}$. We use five protein targets (*parp1*, *fa7*, *5ht1b*, *braf*, *jak2*) and three seeds per target drawn from known actives (Lee et al., 2022), giving 30 tasks per δ. We compare against Graph GA (Jensen, 2019) and GenMol(Lee et al., 2025). Docking is the only oracle; FORGE adapts through the inference-time replay buffer without per-target fine-tuning.

Table 6 reports the per-seed docking score for every (target, seed, δ) configuration. FORGE achieves the best score on 9/15 runs at $\delta = 0.4$ and 11/15 at $\delta = 0.6$, beating GenMol and Graph GA at both similarity floors. The advantage is most consistent on *parp1*, *fa7*, and *jak2*, where FORGE wins all per-seed runs. Graph GA, which mutates without context, trails on most targets and frequently fails to satisfy the similarity floor at $\delta = 0.6$.

Table 6. Lead optimization, per-seed docking scores (kcal/mol; lower is better). Constraints: QED ≥ 0.6, SA ≤ 4, Tanimoto sim $\geq \delta$. "–" = run produced no molecule satisfying all constraints. **Bold** = best in its δ block.

		$\delta = 0.4$			$\delta = 0.6$		
Target	Seed	GenMol	GraphGA	FORGE	GenMol	GraphGA	FORGE
parp1	−7.3	-10.60	-8.30	**-12.07**	-10.40	-8.60	**-11.27**
	−7.8	-11.00	-8.90	**-11.90**	-9.70	-8.10	**-10.93**
	−8.2	-11.30	–	**-11.83**	-9.20	–	**-10.70**
fa7	−6.4	-8.40	-7.80	**-9.17**	-7.30	-7.60	**-8.20**
	−6.7	-8.40	-8.20	**-9.07**	-7.60	-7.60	**-7.93**
	−8.5	–	–	–	–	–	–
5ht1b	−4.5	**-12.90**	-11.70	-12.07	-12.10	-11.30	**-13.37**
	−7.6	**-12.30**	-12.10	-11.37	**-12.00**	**-12.00**	-10.07
	−9.8	**-11.60**	–	-11.07	**-10.50**	–	–
braf	−9.3	-10.80	-9.80	**-11.10**	–	–	**-9.70**
	−9.4	**-10.80**	–	-10.17	**-9.70**	–	–
	−9.8	-10.60	**-11.60**	-10.73	-10.50	-10.40	**-10.60**
jak2	−7.7	-10.20	-8.70	**-10.60**	-9.30	-8.10	**-10.03**
	−8.0	-10.00	-9.20	**-10.90**	-9.40	-9.20	**-10.53**
	−8.6	-9.80	–	**-10.80**	–	–	**-9.80**
# best of 15		4	1	**9**	3	1	**11**

The gap holds up under the tighter similarity floor. Comparing $\delta = 0.4$ to $\delta = 0.6$, GenMol's score on *parp1* (seed -7.8) degrades from -11.00 to -9.70, while FORGE degrades from -11.90 to -10.93 and stays ahead. A similar pattern holds on *jak2* and *fa7* for every seed. This is consistent with the analysis in Section 6.1: context-conditioned fragment supervision is most useful when the optimizer cannot leave the neighborhood of the seed.

The are two cases where FORGE underperforms Genmol. On *5ht1b*, GenMol holds a 0.2–1.9 kcal/mol advantage. Due to small initial pool, the performance of FORGE is not very stable. On *braf* the three methods are within 1 kcal/mol of each other across seeds, with no clear winner. We expect both gaps to close with a larger FORGE backbone (cf. Section 6.2).

5.6. ChemCoTBench: small FORGE versus larger language models

ChemCoTBench (Li et al., 2025) supplies a 5k-pair training set and six optimization tasks: LogP, Solubility, QED, DRD2, JNK3, and GSK3-β. We extract fragment-replacement pairs from the official training set and fine-tuning the FORGE backbone in the same modification format, without external data, oracle calls, or chain-of-thought distillation. This matches the FORGE supervision to what is available to the text-only baselines.

Table 7. ChemCoTBench results. Each cell reports property improvement Δ / success rate SR (%).

Family	Method	LogP	Solub.	QED	DRD2	JNK3	GSK3-β
Text-only	Qwen-3-8B	0.00 / 3	0.00 / 4	0.00 / 4	0.00 / 4	-0.01 / 0	0.00 / 2
	Qwen-3-8B (SFT)	0.15 / 47	0.48 / 52	0.10 / 48	0.04 / 38	-0.02 / 20	0.02 / 36
	GPT-4o	-0.01 / 37	0.92 / 80	0.13 / 70	0.07 / 48	-0.02 / 30	0.00 / 39
Latent	Coconut-Chem (Ye et al., 2026)	0.17 / 44	0.57 / 58	0.15 / 67	0.04 / 45	0.00 / 35	0.06 / 47
	LatentChem (Ye et al., 2026)	**1.37 / 96**	**1.53** / 89	0.18 / 83	0.26 / 74	0.08 / 60	0.17 / 82
Our (0.6B)	QwenAtom + FT	1.07 / 90	0.81 / 82	0.18 / 85	0.32 / 82	0.13 / 67	0.28 / 82
	QwenAtom + S2 + FT	1.02 / 93	1.04 / **95**	0.19 / 88	0.36 / 87	0.24 / 83	0.33 / 89
	FORGE (QwenAtom + S1 + S2) + FT	1.06 / 94	1.03 / **95**	**0.25 / 93**	**0.37 / 91**	**0.27 / 86**	**0.34 / 94**

Table 7 summarizes the comparative evaluation. At 0.6B parameters, FORGE (QwenAtom + Stage 1 + Stage 2) + FT matches or outperforms Qwen-3-8B (SFT) and GPT-4o on every task, with the largest gaps on the real-target tasks: on DRD2, JNK3, and GSK3-β. FORGE reaches success rates of 91%, 86%, and 94%, respectively. In contrast, Qwen-3-8B achieves only 38%, 20%, and 36%, while GPT-4o reaches 48%, 30%, and 39%. The property-improvement metric (Δ) exhibits the same trend: Qwen-3-8B stays below 0.10 on LogP, QED, DRD2, and GSK3-β, and even turns negative on JNK3. This indicates that the natural-language CoT format used in its training carries little optimization signal beyond improving basic output validity. Against the latent-reasoning baseline LatentChem, FORGE outperforms it on four out of six tasks across both metrics; while LatentChem retains a higher Δ on LogP and Solubility, FORGE matches or exceeds its overall success rate on all six tasks.

Notably, even the simplest ablation variant (QwenAtom + FT), which only fine-tunes the backbone on basic fragment replacements, still outperforms LatentChem and the 8B SFT baseline on several tasks. Given that these baselines rely on complex reinforcement learning or extensive chain-of-thought reasoning, this result further supports our argument: natural language is likely a suboptimal interface for molecular optimization.

6. Analysis

The experiments show that FORGE performs well across similarity-constrained optimization, black-box adaptation, and real-target editing. Two questions remain after the main results: how much of the Prompt-MolOpt gain comes from data construction versus output format, and how do Stage 1 and Stage 2 divide labor? We now analyze them in turn.

6.1. Context-conditioned data matters most under local-edit constraints

We isolate one design choice: building edit supervision from context-conditioned fragment effects (SME+) versus a single global statistic per fragment (the SME-style aggregation used by the original Prompt-MolOpt data). We compare two models with the same backbone and the same end-to-end output format, differing only in the training data used for Prompt-MolOpt continuation.

Table 8. Effect of training-data aggregation on Prompt-MolOpt SUM. The backbone and output format are fixed.

δ	SME+	original SME-style	Δ
0.0	3.781	3.977	−0.196
0.4	3.582	2.928	+0.654
0.6	2.187	1.354	+0.833

The pattern in Table 8 flips with the similarity threshold. At $\delta = 0$, the model is free to move to distant high-scoring regions, and global aggregation is slightly more useful. Once the similarity floor becomes difficult, SME+ wins by $+0.65$ SUM at $\delta = 0.4$ and $+0.83$ at $\delta = 0.6$. Within-context supervision matches the local-edit regime that the similarity floor enforces.

Switching to SME+ data alone adds $+0.833$ SUM at $\delta = 0.6$, the single largest of the three factors in Table 9. Output interface and supervision data contribute on a comparable scale; the data construction is the single largest factor. This shows that FORGE benefits not only from a better output interface, but also from supervision that matches the local nature of similarity-constrained editing.

Table 9. Decomposition of the gain over the original Prompt-MolOpt baseline at δ=0.6.

Source	SUM δ=0.6	gain
Prompt-MolOpt baseline	0.824	—
+ backbone upgrade	1.354	+0.530
+ SME+ data	2.187	+0.833
+ modification format (S1+S2)	2.882	+0.695

6.2. Stage 2 provides the main editing capability, while Stage 1 improves localization

The ablations in Prompt-MolOpt and ChemCoTBench suggest Stage 1 and Stage 2 contribute differently across bench-

marks. Table 10 and Figure 3 summarize that pattern across representative settings.

Table 10. Marginal gains of Stage 1 and 2 across settings.

Setting	qwenatom	+ S2	+ S1+S2
Prompt-MolOpt δ=0	4.379	4.474	4.587
Prompt-MolOpt δ=0.4	4.114	4.235	4.302
Prompt-MolOpt δ=0.6	2.786	2.860	2.882
ChemCoT JNK3 Δ	0.13	0.24	0.27
ChemCoT DRD2 Δ	0.32	0.36	0.37
ChemCoT GSK3-β Δ	0.28	0.33	0.34

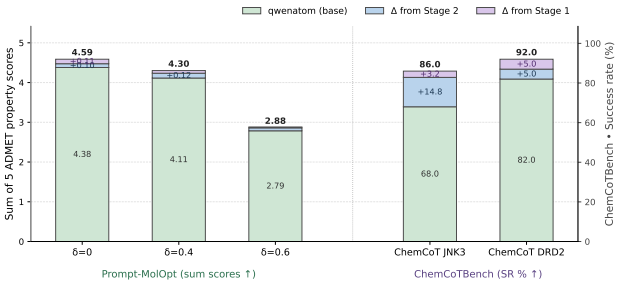


Figure 3. Marginal gains of Stage 1 and Stage 2 across representative benchmarks.

Stage 2 is the larger increment throughout. This is expected: Stage 2 supervision is in the same output space used at inference, so the gradient signal directly trains the inference behavior. The increment is largest on the real-target tasks in ChemCoTBench (JNK3 SR: $67 \rightarrow 83$; Solubility: $82 \rightarrow 95$).

Stage 1 adds a smaller but consistent gain on top of Stage 2. The improvement is most visible when the search space is broad or the oracle signal is sparse, such as unconstrained or moderately constrained Prompt-MolOpt and the real-target tasks in ChemCoTBench. Stage 1 therefore acts as a localisation prior on top of Stage 2 rather than as a substitute.

These results clarify why fragment-aware supervision fits molecular optimization better than property prediction. In standard property prediction, each molecule is paired with a single scalar label, so fragment-level credit assignment must be inferred indirectly. In optimization, by contrast, a source molecule, a local modification, and the resulting oracle change already provide supervision at roughly the fragment level. FORGE turns this structure into direct supervision via Stage 1 ranking and Stage 2 replacement.

FORGE does not dominate every PMO task. The per-task breakdown in Appendix B shows that the largest deficits occur against MOLLEO on tasks tied to specific named drugs or heavily studied targets, such as *albuterol_similarity*, *perindopril_mpo*, *drd2*, and *amlodipine_mpo*; MOLLEO is the stronger baseline on these specific tasks even though LICO has the higher aggregate score. These tasks likely reward broad memorisation of named-drug chemistry, which favours much intensively pretrained backbones such as the BioT5 (Pei et al., 2023) generators inside MOLLEO. We leave drug/target-specific demonstration pools to future work.

6.3. Limitations

First, FORGE does not explicitly model synthesizability: SME+ and ChEMBL MMPs contain only realized molecules, but provide no supervision on whether a proposed edit matches a known reaction, which occasionally leads to outputs with high SA scores (e.g., in ChemCoTBench Solubility). Second, some failures are due to out-of-distribution property knowledge: the 0.6B base lacks large-scale memorized drug–structure associations, so these errors reflect a missing prior rather than an optimization failure and are unlikely to be fixed by ICL evolution alone. Third, Stages 1–2 do not span all properties: the backbone is trained on five ADMET properties and ChEMBL MMPs, making it a strong initialization but not a zero-shot generator; novel objectives still require the per-evaluation SFT step.

7. Discussion and Conclusion

FORGE casts molecular optimization as context-aware local editing on a 0.6B language model: atom-level tokenization keeps fragment identity stable across host molecules, Stage 1 ranks fragments under full-molecule context, Stage 2 generates explicit replacements supervised by SME+ pairs and ChEMBL MMPs, and inference adapts to unnamed objectives through a replay buffer of (edit, score) demonstrations. This combination exceeds Qwen3-8B (SFT) and GPT-4o on ChemCoTBench at 0.6B parameters, and sets the best aggregate scores we are aware of on Prompt-MolOpt (SUM $4.59/4.30/2.88$ at $\delta = 0/0.4/0.6$), PMO-1k (SUM 12.42 over 22 tasks), QED-DRD2, and Lead Optimization.

Two points generalize beyond this method. First, for fragment-based molecular optimization, the value of a fragment depends on the molecule that hosts it, so within-context supervision (SME+) is worth more than the output-format change (Section 6.1). Second, when a black-box objective lacks a faithful natural-language description, demonstrations and oracle scores are a stronger control signal than prompt rewriting. Both observations are independent of model scale, and we expect the same pipeline to benefit from a larger backbone as named-drug priors become a bottleneck (Section 6.2). Two natural extensions are coupling the inference buffer with a retrosynthesis filter to address the SA-score issue identified in Section 6.3, and porting the fragment-replacement supervision format to reaction-prediction tasks where matched-pair data is also abundant.

References

Brown, N., Fiscato, M., Segler, M. H., and Vaucher, A. C. Guacamol: benchmarking models for de novo molecular design. *Journal of chemical information and modeling*, 59(3):1096–1108, 2019.

Cai, F., Zacour, K., Zhu, T., Tzeng, T.-R., Duan, Y., Liu, L., Pilla, S., Li, G., and Luo, F. Chemfm as a scaling law guided foundation model pre-trained on informative chemicals. *Communications Chemistry*, 2025.

Chen, Z., Min, M. R., Parthasarathy, S., and Ning, X. A deep generative model for molecule optimization via one fragment modification. *Nature machine intelligence*, 3 (12):1040–1049, 2021.

Chithrananda, S., Grand, G., and Ramsundar, B. Chemberta: large-scale self-supervised pretraining for molecular property prediction. *arXiv preprint arXiv:2010.09885*, 2020.

Chitsaz, K., Balaji, R., Fournier, Q., Bhatt, N. P., and Chandar, S. Novomolgen: Rethinking molecular language model pretraining. *arXiv preprint arXiv:2508.13408*, 2025.

Dalke, A., Hert, J., and Kramer, C. mmpdb: An open-source matched molecular pair platform for large multiproperty data sets. *Journal of Chemical Information and Modeling*, 58(5):902–910, 2018. doi: 10.1021/acs.jcim.8b00173. URL https://doi.org/10.1021/acs.jcim.8b00173. PMID: 29770697.

Dara, S., Dhamercherla, S., Jadav, S. S., Babu, C. M., and Ahsan, M. J. Machine learning in drug discovery: a review. *Artificial intelligence review*, 55(3):1947–1999, 2022.

Fang, Y., Liang, X., Zhang, N., Liu, K., Huang, R., Chen, Z., Fan, X., and Chen, H. Mol-Instructions: A Large-Scale Biomolecular Instruction Dataset for Large Language Models. *arXiv e-prints*, art. arXiv:2306.08018, June 2023. doi: 10.48550/arXiv.2306.08018.

Ferreira, F. J. and Carneiro, A. S. Ai-driven drug discovery: a comprehensive review. *ACS omega*, 10(23):23889–23903, 2025.

Gao, W., Fu, T., Sun, J., and Coley, C. Sample efficiency matters: a benchmark for practical molecular optimization. *Advances in neural information processing systems*, 35:21342–21357, 2022.

Gaulton, A., Bellis, L. J., Bento, A. P., Chambers, J., Davies, M., Hersey, A., Light, Y., McGlinchey, S., Michalovich, D., Al-Lazikani, B., and Overington, J. P. Chembl: a large-scale bioactivity database for drug discovery. *Nucleic Acids Research*, 40(D1):D1100–D1107, 01 2012. ISSN 0305-1048. doi: 10.1093/nar/gkr777. URL https://doi.org/10.1093/nar/gkr777.

Guevorguian, P., Bedrosian, M., Fahradyan, T., Chilingaryan, G., Khachatrian, H., and Aghajanyan, A. Small molecule optimization with large language models. *arXiv preprint arXiv:2407.18897*, 2024.

Guo, J. and Schwaller, P. Augmented Memory: Capitalizing on Experience Replay to Accelerate De Novo Molecular Design. *arXiv e-prints*, art. arXiv:2305.16160, May 2023. doi: 10.48550/arXiv.2305.16160.

Hurst, A., Lerer, A., Goucher, A. P., Perelman, A., Ramesh, A., Clark, A., Ostrow, A., Welihinda, A., Hayes, A., Radford, A., et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Jensen, J. H. A graph-based genetic algorithm and generative model/monte carlo tree search for the exploration of chemical space. *Chemical Science*, 10(12):3567–3572, 2 2019. doi: 10.1039/c8sc05372c. PMID: 30996948; PMCID: PMC6438151.

Ji, C., Zheng, Y., Wang, R., Cai, Y., and Wu, H. Graph polish: A novel graph generation paradigm for molecular optimization. *IEEE Transactions on Neural Networks and Learning Systems*, 34(5):2323–2337, 2021.

Jin, W., Barzilay, R., and Jaakkola, T. Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning*, pp. 2323–2332. PMLR, 2018a.

Jin, W., Yang, K., Barzilay, R., and Jaakkola, T. Learning multimodal graph-to-graph translation for molecular optimization. *arXiv preprint arXiv:1812.01070*, 2018b.

Jin, W., Barzilay, R., and Jaakkola, T. Hierarchical graph-to-graph translation for molecules. *arXiv preprint arXiv:1907.11223*, 2019.

Jin, W., Barzilay, R., and Jaakkola, T. Hierarchical generation of molecular graphs using structural motifs. In *International conference on machine learning*, pp. 4839–4848. PMLR, 2020.

Kim, H., Kim, M., Choi, S., and Park, J. Genetic-guided gflownets for sample efficient molecular optimization. *Advances in Neural Information Processing Systems*, 37: 42618–42648, 2024.

Lee, S., Jo, J., and Hwang, S. J. Exploring Chemical Space with Score-based Out-of-distribution Generation. *arXiv e-prints*, art. arXiv:2206.07632, June 2022. doi: 10.48550/arXiv.2206.07632.

Lee, S., Kreis, K., Veccham, S. P., Liu, M., Reidenbach, D., Paliwal, S., Vahdat, A., and Nie, W. Molecule generation with fragment retrieval augmentation. *Advances in Neural Information Processing Systems*, 37:132463–132490, 2024.

Lee, S., Kreis, K., Veccham, S. P., Liu, M., Reidenbach, D., Peng, Y., Paliwal, S., Nie, W., and Vahdat, A. Genmol: A drug discovery generalist with discrete diffusion. *arXiv preprint arXiv:2501.06158*, 2025.

Li, H., Cao, H., Feng, B., Shao, Y., Tang, X., Yan, Z., Yuan, L., Tian, Y., and Li, Y. Beyond chemical qa: Evaluating llm's chemical reasoning with modular chemical operations. *arXiv preprint arXiv:2505.21318*, 2025.

Liu, G., Chen, J., Zhu, Y., Sun, M., Luo, T., Chawla, N. V., and Jiang, M. Graph diffusion transformers are in-context molecular designers. *arXiv preprint arXiv:2510.08744*, 2025.

Musiał, S., Zieliński, B., and Danel, T. Fragment-wise interpretability in graph neural networks via molecule decomposition and contribution analysis. *arXiv preprint arXiv:2508.15015*, 2025.

Nguyen, T. and Grover, A. Lico: Large language models for in-context molecular optimization. *arXiv preprint arXiv:2406.18851*, 2024.

Olivecrona, M., Blaschke, T., Engkvist, O., and Chen, H. Molecular De Novo Design through Deep Reinforcement Learning. *arXiv e-prints*, art. arXiv:1704.07555, April 2017. doi: 10.48550/arXiv.1704.07555.

Pei, Q., Zhang, W., Zhu, J., Wu, K., Gao, K., Wu, L., Xia, Y., and Yan, R. Biot5: Enriching cross-modal integration in biology with chemical knowledge and natural language associations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 1102–1123. Association for Computational Linguistics, December 2023. URL https://aclanthology.org/2023.emnlp-main.70.

Proszewska, M., Danel, T., and Rymarczyk, D. B-xaic dataset: Benchmarking explainable ai for graph neural networks using chemical data. *arXiv preprint arXiv:2505.22252*, 2025.

Ross, J., Belgodere, B., Hoffman, S. C., Chenthamarakshan, V., Navratil, J., Mroueh, Y., and Das, P. Gp-molformer: A foundation model for molecular generation. *Digital Discovery*, 4(10):2684–2696, 2025.

Roth, J. P. Chemically interpretable explanations for molecular property prediction via fragment-level shapley values. *ChemRxiv*, 2026(0421), 2026. doi: 10.26434/chemrxiv.15002302/v1. URL https://chemrxiv.org/doi/abs/10.26434/chemrxiv.15002302/v1.

Srinivas, N., Krause, A., Kakade, S. M., and Seeger, M. Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. *arXiv e-prints*, art. arXiv:0912.3995, December 2009. doi: 10.48550/arXiv.0912.3995.

Van Tilborg, D., Alenicheva, A., and Grisoni, F. Exposing the limitations of molecular machine learning with activity cliffs. *Journal of chemical information and modeling*, 62(23):5938–5951, 2022.

Wang, H., Skreta, M., Ser, C.-T., Gao, W., Kong, L., Strieth-Kalthoff, F., Duan, C., Zhuang, Y., Yu, Y., Zhu, Y., Du, Y., Aspuru-Guzik, A., Neklyudov, K., and Zhang, C. Efficient Evolutionary Search Over Chemical Space with Large Language Models. *arXiv e-prints*, art. arXiv:2406.16976, June 2024. doi: 10.48550/arXiv.2406.16976.

Wang, Z., Nie, W., Qiao, Z., Xiao, C., Baraniuk, R., and Anandkumar, A. Retrieval-based Controllable Molecule Generation. *arXiv e-prints*, art. arXiv:2208.11126, August 2022. doi: 10.48550/arXiv.2208.11126.

Wu, Z., Ramsundar, B., Feinberg, E. N., Gomes, J., Geniesse, C., Pappu, A. S., Leswing, K., and Pande, V. MoleculeNet: A Benchmark for Molecular Machine Learning. *arXiv e-prints*, art. arXiv:1703.00564, March 2017. doi: 10.48550/arXiv.1703.00564.

Wu, Z., Wang, J., Du, H., Jiang, D., Kang, Y., Li, D., Pan, P., Deng, Y., Cao, D., Hsieh, C.-Y., et al. Chemistry-intuitive explanation of graph neural networks for molecular property prediction with substructure masking. *Nature communications*, 14(1):2585, 2023.

Wu, Z., Zhang, O., Wang, X., Fu, L., Zhao, H., Wang, J., Du, H., Jiang, D., Deng, Y., Cao, D., et al. Leveraging language model for advanced multiproperty molecular optimization via prompt engineering. *Nature Machine Intelligence*, 6(11):1359–1369, 2024.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Ye, X., Mao, Y., Zhang, J., Liu, Y., Hao, L., Wu, F., Li, Z., Liao, Y., Wang, Z., Wu, Y., et al. Latentchem: From textual cot to latent thinking in chemical reasoning. *arXiv preprint arXiv:2602.07075*, 2026.

You, J., Liu, B., Ying, Z., Pande, V., and Leskovec, J. Graph convolutional policy network for goal-directed molecular graph generation. *Advances in neural information processing systems*, 31, 2018.

Yu, B., Baker, F. N., Chen, Z., Ning, X., and Sun, H. Llasmol: Advancing large language models for chemistry with a large-scale, comprehensive, high-quality instruction tuning dataset. *arXiv preprint arXiv:2402.09391*, 2024.

Zhang, O., Lin, H., Zhang, H., Zhao, H., Huang, Y., Hsieh, C.-Y., Pan, P., and Hou, T. Deep lead optimization: leveraging generative ai for structural modification. *Journal of the American Chemical Society*, 146(46):31357–31370, 2024.

Zhou, G., Gao, Z., Ding, Q., Zheng, H., Xu, H., Wei, Z., Zhang, L., and Ke, G. Uni-mol: A universal 3d molecular representation learning framework. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=6K2RM6wVqKu.

A. Per-property Variance Reduction

Table 11 reports the per-property VR for the three properties shared across all three predictors in Section 3.2. We test the models on 10K mol pool of ZINC dataset. Uni-Mol and ChemBERTa are trained on MoleculeNet. RGCN is taken from SME.

Table 11. Per-property variance reduction on the three properties shared by SME (RGCN), Uni-Mol, and ChemBERTa.

Property	SME (RGCN)	Uni-Mol	ChemBERTa
ESOL	17.32%	18.18%	14.32%
Lipophilicity	19.01%	20.19%	15.12%
BBBP	14.29%	14.95%	19.76%

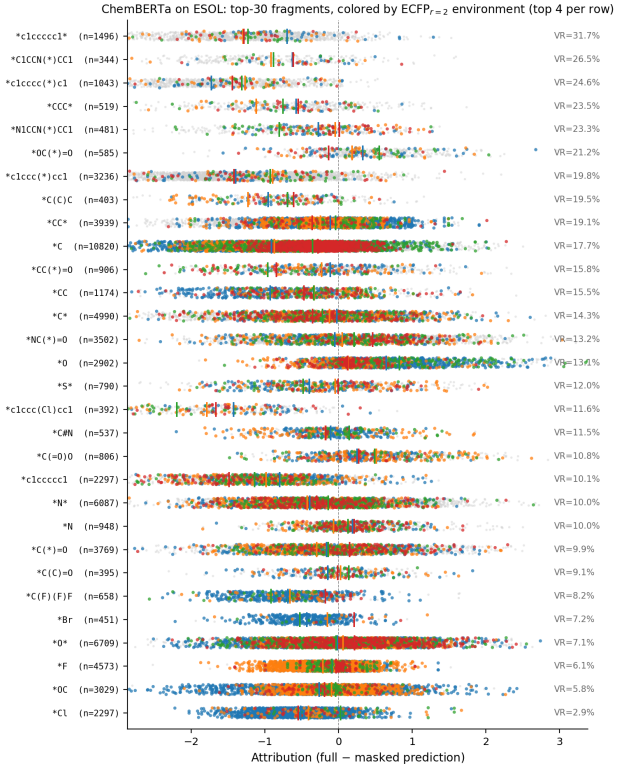


Figure 4. Fragment decoupling of ChemBERTa on lipop

B. More experiment details

Table 12 reports all 22 PMO tasks under the 1k-call budget. Columns: GP-BO (Srinivas et al., 2009), Graph GA (Jensen, 2019), REINVENT (Olivecrona et al., 2017), LICO (Nguyen & Grover, 2024), Genetic GFN (Kim et al., 2024), Augmented Memory (Guo & Schwaller, 2023), MOLLEO (Wang et al., 2024), FORGE backbone (*qwenatom + S1 + S2*), and the FORGE Stage-1-ablation (*qwenatom + S2*). Means across 5 runs are reported. The four tasks where MOLLEO outperforms FORGE — *albuterol_similarity*, *amlodipine_mpo*, *drd2*, *perindopril_mpo* — are the named-drug-recall failure modes.

For Prompt-MolOpt, multi-turn generation and repeated single-turn generation are similar for two-step trajectories, but multi-turn becomes preferable as the number of edits increases as shown in Table 13. This is consistent with the role of dialogue context in preserving a coherent sequence of local modifications.

C. Hyperparameters

All training runs use AdamW with peak learning rate 2×10^{-5}, linear warm-up over 5,000 steps and cosine decay, per-device batch size 4, gradient accumulation 1, sequence length 2048, bf16 mixed precision, and DeepSpeed ZeRO-1 across 4×NVIDIA A100 (80GB) GPUs. Loss is the standard left-to-right token cross-entropy; for ICL samples and multi-turn answers, the loss is masked to the answer span only. Per-stage epoch counts are summarized in Table 14.

For inference: vLLM with temperature τ=0.7, top-p=0.95, max generation length 512 (single-step) or 1024 (multi-turn). The ICL replay buffer keeps the top 200 cache and samples with the top K=5 verified pairs by improvement Δ and is refreshed after every successful step.

D. Data Construction Details

This appendix documents the recipes used to build each training corpus referenced in Section 4: source, volume, output schema, and a representative example. Final mixture statistics are summarized in Tables 16 and 17.

D.1. Training Data and Supervision Details

Table 15 summarizes the supervision sources used across the two training stages of FORGE. Stage 1 uses fragment-level attribution labels to teach the model where to edit, while Stage 2 uses verified fragment replacement pairs to teach how to edit. Among these sources, ChEMBL matched molecular pairs support both stages: they provide demonstration-based ranking supervision in Stage 1 and explicit low-to-high edit pairs in Stage 2.

Stage 1 supervision. GNN-based attribution is used for properties without closed-form evaluators, including hERG, BBBP, mutagenicity, and solubility. RDKit-based attribution is used for rule-based properties, including QED, LogP, SA, and TPSA. Both sources produce fragment-level ranking and vulnerability labels.

Stage 2 supervision. SME+ produces context-conditioned low-to-high fragment replacement pairs on five ADMET properties: ESOL, BBBP, hERG,

Table 12. PMO-1k per-task scores. The aggregate sum is reported in the bottom row; FORGE backbone reaches 12.42, +0.71 over LICO and +0.77 over MOLLEO. Best per row in bold.

Task	GP-BO	Graph GA	REINVENT	LICO	Genetic GFN	Aug. Mem.	MOLLEO	**FORGE (S1+S2)**	FORGE (S2)
albuterol_similarity	0.636	0.583	0.496	0.656	0.664	0.557	**0.886**	0.779	0.764
amlodipine_mpo	0.519	0.501	0.472	0.541	0.534	0.489	**0.637**	0.551	0.554
celecoxib_rediscovery	0.411	0.424	0.370	**0.447**	**0.447**	0.385	0.402	0.428	0.441
deco_hop	0.593	0.581	0.572	0.596	0.604	0.579	0.588	**0.624**	**0.624**
drd2	0.857	0.833	0.775	0.859	0.809	0.795	**0.910**	0.816	0.818
fexofenadine_mpo	0.707	0.666	0.650	0.700	0.682	0.679	0.674	0.740	**0.742**
gsk3b	0.611	0.523	0.589	0.617	**0.637**	0.539	0.397	0.620	0.581
isomers_c7h8n2o2	0.545	0.735	0.725	**0.779**	0.738	0.661	0.737	0.723	0.652
isomers_c9h10n2o2pf2cl	0.599	0.630	0.630	0.672	0.656	0.596	0.635	**0.713**	0.712
jnk3	0.346	0.301	0.315	0.336	0.409	0.294	0.186	**0.611**	0.584
median1	0.213	0.208	0.205	0.217	0.219	0.219	0.236	**0.271**	0.250
median2	0.203	0.181	0.188	0.193	0.204	0.184	0.191	**0.221**	0.220
mestranol_similarity	0.427	0.362	0.379	0.423	0.414	0.393	0.399	0.450	**0.453**
osimertinib_mpo	0.766	0.751	0.737	0.759	0.763	0.761	0.779	**0.794**	0.787
perindopril_mpo	0.458	0.435	0.404	0.473	0.462	0.422	**0.655**	0.509	0.522
qed	0.912	0.914	0.921	0.925	0.928	0.923	0.919	**0.937**	**0.937**
ranolazine_mpo	0.701	0.620	0.574	0.687	0.623	0.614	0.640	0.680	0.690
scaffold_hop	0.478	0.461	0.447	0.480	0.485	0.460	0.473	**0.532**	0.511
sitagliptin_mpo	0.232	0.229	0.261	0.315	0.227	0.245	0.193	0.300	**0.337**
thiothixene_rediscovery	0.351	0.322	0.311	0.343	0.377	0.336	**0.416**	0.400	0.374
troglitazone_rediscovery	**0.313**	0.267	0.246	0.292	0.277	0.262	0.302	0.288	0.286
valsartan_smarts	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
zaleplon_mpo	0.392	0.374	0.406	0.404	0.400	0.415	0.392	0.435	**0.436**
SUM (↑)	11.27	10.90	10.68	11.71	11.56	10.81	11.65	**12.42**	12.27

Table 13. Similarity to the source molecule on Prompt-MolOpt, comparing a single multi-turn dialogue against repeated single-turn calls for trajectories of the same length. Higher is better.

# turns	multi-turn	iter	n
2	0.487	**0.495**	7274
3	**0.483**	0.468	1164
4	**0.474**	0.447	328

Table 14. Per-stage epoch counts. All other hyperparameters (optimizer, lr, warm-up, batch size, hardware) are shared.

Stage	Epochs
Stage 1 (ranking + decomposition + ICL)	3
Stage 2 backbone (single-step modification)	1
Stage 2 multi-turn (Prompt-MolOpt)	1
Other downstream finetune (QED-DRD2, ChemCoTBench, etc.)	3

lipophilicity, and mutagenicity. ChEMBL matched molecular pairs provide real-activity fragment replacement pairs constructed from measured `pchembl` differences on the same target.

Leakage prevention for ChEMBL. To avoid overlap with downstream evaluations, we remove all ChEMBL pairs whose target description matches DRD2, JNK3, or GSK3-β. This filtering is applied before constructing the final training pool.

Table 15. Data sources and supervision across training stages.

Source	Supervision	Training Stage
GNN mask attribution	ranking labels	1
RDKit rule attribution	ranking labels	1
SME+	verified pairs	2
ChEMBL MMPA	verified pairs / ICL demonstrations	1 & 2

Table 16. Stage 1 source mixture, totalling 2M training examples.

Source	Ratio
RDKit fragment attribution (ranking + vulnerability)	25%
GNN mask attribution (ranking)	15%
GNN decomposition prediction	10%
ICL ranking (RDKit + ChEMBL MMP demonstrations)	35%
Mol-Instructions (external instruction-following)	15%

D.2. Stage 1: discriminative fragment-level tasks (˜2M samples)

Stage 1 trains the model to *rank* fragments by their per-atom contribution to a property and to identify the weakest fragment (a vulnerability prompt). Sources and proportions are listed in Table 16; smaller sources are oversampled to match their target ratio.

RDKit fragment attribution. We sample ˜50k molecules per property from ZINC. Each molecule is decomposed by one of {Murcko, BRICS, EFG} (chosen in random order; the first decomposition that yields $\geq$ 3 fragments wins). For every fragment, we recompute the molecule's property

after two removal procedures, replace-with-H and delete-with-cap, and store the per-atom delta Δ/n_{atoms} to remove a known size bias. Twelve RDKit properties are covered: QED, logP, SA, TPSA, molecular weight, HBA, HBD, rotatable bonds, aromatic rings, Fsp3, ring count, and heavy-atom count. Each (molecule, property, removal) triple is converted into either a *ranking* prompt (sort all fragments) or a *vulnerability* prompt (return the weakest), and 35% of prompts request the normalized contribution score in addition to the ordering. A representative example:

> **Instruction:** Which fragment contributes the most to 'number of rings'? Rank all fragments and include their normalized contribution scores (0 = lowest, 1 = highest observed contribution).
>
> **Input:** Molecule SMILES: <start_smiles>C=CCNC(=O)CN1CC(S(=O)(=O)N2CCNCC2)C1<end_smiles> List of fragments to be ranked: <start_smiles>[3*]C[4*]....<end_smiles>
>
> **Output:** <start_smiles>[4*]N([5*])[10*]<end_smiles> (score: 0.14) > ... > <start_smiles>[9*]CC[11*]<end_smiles> (score: 0.0)

GNN attribution. A mirror of the RDKit pipeline that uses the trained SME RGCN as the property predictor (so it covers learned ADMET endpoints) and SME's substructure mask as the decomposition. Two task families are emitted: *ranking* (sort fragments by attribution score) and *decomposition* (predict the fragment list induced by the mask).

ICL ranking. 100k RDKit-ICL and 500k ChEMBL-MMP-ICL samples. For RDKit-ICL, we draw $K{=}5$ demonstrations sharing the same (property, removal-method) cell from the attribution corpus and mask the property name, forcing pattern induction from demonstrations alone. For MMP-ICL, the demonstrations are different scaffolds against the same biological target, and the query is a held-out scaffold whose candidate fragments must be ranked by activity. We mix three prompt variants: ICL with target description (50%), ICL without description (30%), and direct ranking with no demonstrations (20%); each variant switches to a vulnerability prompt with probability 0.25 and adds normalized scores with probability 0.35. Targets whose description mentions JNK, DRD, and GSK are excluded so they remain unseen at evaluation time.

D.3. Stage 2: generative single-step modification (˜680k samples)

Stage 2 teaches the model to emit a fragment-replacement edit and the resulting molecule. Sources and proportions are listed in Table 17.

Table 17. Stage 2 source mixture (single-step backbone). The multi-turn variant for Prompt-MolOpt is trained as a separate continuation; see text.

Source	Pairs (post-dedup)	Share
SME+ verified ADMET pairs (5 properties)	˜136k*2	40%
ChEMBL MMP pairs (multi-target)	˜411k	60%

SME+ verified pairs. Sub-trajectories whose atom mapping survives the SME+ verifier are flattened into single-step pairs covering five ADMET endpoints: ESOL, BBBP, hERG, lipophilicity, and mutagenicity. Each property is mapped through a sigmoid or sign flip so that "higher is better" holds uniformly, scores are min-max normalized per property to $[0, 1]$, pairs with $\Delta_{\text{score}} \leq 0.01$ are dropped, and each unique *frag_src* $\rightarrow$ *frag_tgt* edit is capped at five occurrences to prevent a small set of common substitutions from dominating the loss. The single-step output template is

```
Modification: <frag_src> >> <frag_tgt>
Result:       <tgt_smi> (value: 0.74)
```

with the `(value:·)` suffix present in 80% of samples. Prompts are split between direct (10%, naming the property by description or by name) and ICL (90%, with $K{=}5$ demonstrations from the same property).

ChEMBL MMP optimization. Matched-molecular pairs are mined across ChEMBL targets with pChEMBL ≥ 5 on both sides. We orient each pair so that the higher-pChEMBL side is the optimization target, normalize pChEMBL per target to $[0, 1]$, and discard any pair whose larger fragment exceeds 15 heavy atoms or covers more than 40% of the parent (these tend to encode whole-scaffold swaps rather than local edits). The same `Modification / Result` template is used, but the score field is named `activity` instead of `value`. Prompts are mixed between direct (target named or described) and ICL (five demonstrations sharing the same target). DRD2, JNK3 and GSK3b are again held out by description match.

Multi-turn variant. For Prompt-MolOpt, we additionally train a multi-turn continuation that reuses the verified SME+ trajectories but emits the entire sub-trajectory (capped at six steps) as a single answer, so the model learns to plan a sequence of edits in one shot. The training data statistics for each property are summarized as Table 18

Properties are upsampled to the count of the largest property(lipop, 47485) to balance the mixture. The output is a concatenation of `Modification / Result` blocks:

```
Modification: <frag_src_1> >> <frag_tgt_1>
Result: <mol_1> (value: 0.41)
Modification: <frag_src_2> >> <frag_tgt_2>
Result: <mol_2> (value: 0.58)
...
```

Table 18. Statistics of multi-turn training trajectories across different properties.

Property	Goal	Sub-trajs	Steps
lipop	Higher	47,485	103,570
ESOL	Higher	33,191	72,295
hERG	Lower	31,247	63,413
BBBP	Higher	24,269	53,906
Mutagenicity	Lower	22,700	45,610

Format normalization. Across all stages, isotope brackets on atoms in the organic subset $\{B, C, N, O, P, S, F, Cl, Br, I\}$ are stripped to keep SMILES canonical, while dummy labels of the form $[k*]$ on fragment attachment points are preserved so that *frag_src* and *frag_tgt* remain alignable. SMILES are wrapped with the dedicated `<start_smiles>`/`<end_smiles>` markers introduced in Section 4.3 so that the model can locate molecular spans regardless of surrounding natural language.

D.4. Downstream task data

For ChemCoTBench (6 sub-tasks) and QED-DRD2, we use the public training splits as released, converted into the same instruction/input/output schema with SMILES-wrapping normalization and Maximum Common Substructure-based fragment-pair extraction.

E. Stage 1 internal evaluation: atom-level versus vanilla tokenization

We hold the training data, optimization setup, and training steps fixed, and vary only the tokenization scheme. The test set covers 13 held-out Stage 1 sub-tasks, including decomposition, ranking, and vulnerability prediction in both direct and ICL settings.

Table 19. Stage 1 internal evaluation: atom-level tokenization versus vanilla tokenization.

Sub-task	Metric	van.	atom	Δ
Decomposition	F1	0.880	**0.906**	+0.026
Direct Ranking	K-τ	0.760	**0.777**	+0.017
Direct Vulnerability	EM	0.798	**0.870**	+0.072
ICL Ranking	K-τ	0.626	**0.753**	+0.127
ICL Vulnerability	EM	0.600	**0.722**	+0.122
MMP Direct Rank	K-τ	0.183	**0.216**	+0.033
MMP Direct Vuln.	EM	0.272	**0.332**	+0.060
MMP ICL nd. Rank	K-τ	0.453	**0.512**	+0.059
MMP ICL nd. Vuln.	EM	0.506	**0.540**	+0.034
MMP ICL d. Rank	K-τ	0.472	**0.503**	+0.031
MMP ICL d. Vuln.	EM	0.454	**0.556**	+0.102
Direct Rank top-1	Acc	0.752	**0.770**	+0.018
Direct Rank top-3	Acc	0.892	**0.907**	+0.015