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

Anonymous ACL submission

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 and 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 replace-

ments. 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 detailed 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 assumes fragment effects are weakly context-dependent, which breaks down at activity cliffs (Van Tilborg et al., 2022). Deep-learning variants such as JT-VAE (Jin et al., 2018a), GCPN (You et al., 2018), Modof (Chen et al., 2021), and Prompt-MolOpt (Wu et al., 2024) hard-code fragment edits as graph operators or external vocabularies, typically averaging fragment statistics across contexts. Substructure attribution methods such as SME (Wu et al., 2023) and fragment-level Shapley values (Roth, 2026) improve fragment importance estimates. Our method differs: 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 from global min–max to within-context binning.

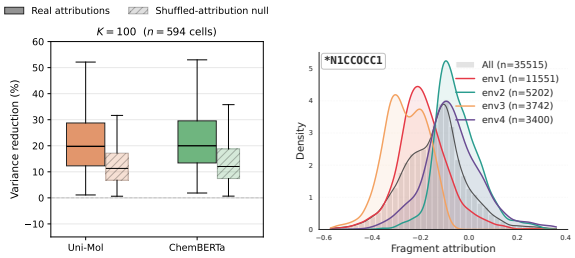
Chemistry language models and inference-time adaptation. SMILES-based pretraining has established language modeling as a competitive route for molecular modeling, from ChemBERTa (Chithrananda et al., 2020) to ChemLactica (Guevorguian et al., 2024) and LlaSMol (Yu et al., 2024). Our approach departs in 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. Our inference procedure is motivated by 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. FORGE conditions on the target objective only through demonstrations sampled at search time, avoiding costly gradient-based test-time updates.

3 Motivation

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.

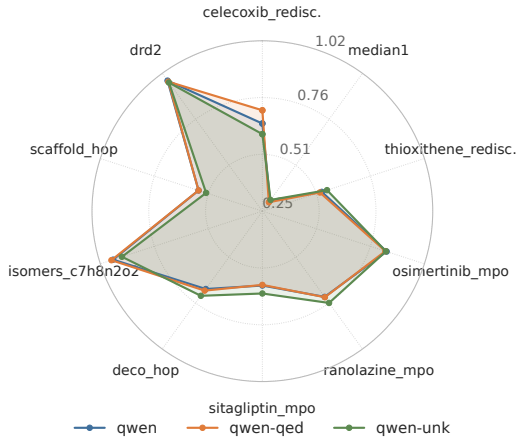
3.1 Prompts weakly specify black-box objectives

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



(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.

± 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.

We test this through fragment attribution. Let $f : \mathcal{X} \to \mathbb{R}$ be an 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.

We quantify this effect using 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 by 8%–11% compared to the shuffled grouping (detailed results in Appendix A.2, Table 6). Figure 1(a,b) visualizes the same effect.

The choice of neighborhood radius controls how finely context is partitioned. Sweeping $r{=}1$ to $r{=}4$ on ChemBERTa shows that the genuine signal Δ grows monotonically with r, but larger radii also fragment the data into 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 trade-off. Radius sweep results and per-property breakdowns are in Appendix A.2 (Tables 6 and 7).

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 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*.

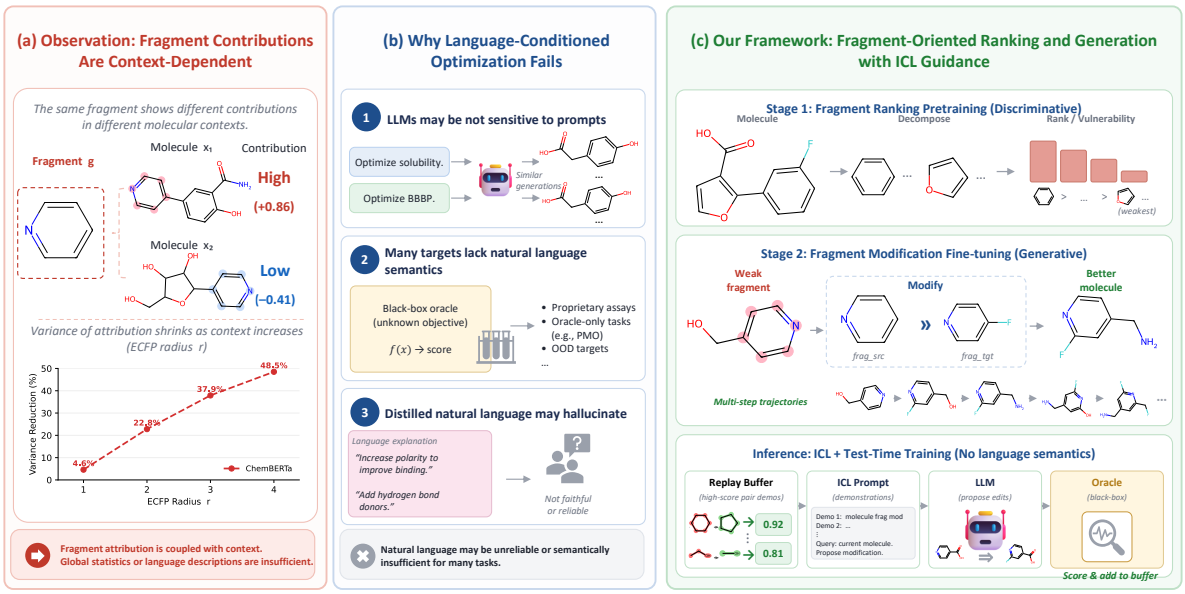


Figure 2: **Overview of the FORGE framework. (a)** Left: Empirical observations reveal that fragment contributions are highly context-dependent. **(b)** Middle: Natural language is a suboptimal interface for optimization due to weak prompt control, lack of oracle semantics, and distillation hallucinations. **(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.

From this, two principles follow: a model should derive its target semantics 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 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. 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. Figure 2 gives an overview. Both stages are trained via supervised next-token prediction on `(instruction, input, output)` triples with loss computed on the output span; full dataset construction and supervision sources are in Appendix D.

4.1 Stage 1: learning where to edit

Stage 1 is supervised by fragment attribution from three sources: RGCN models with mask-aware readout (from SME) for learned properties without closed-form evaluators (hERG, mutagenicity, lipop, BBBP, ESOL), RDKit explicit removal-and-recompute for rule-based properties (QED, logP, SA, TPSA), and ChEMBL (Gaulton et al., 2012). The first provides broad learned supervision; the second provides exact labels; the third provides real world targets.

From these attributions, we build four task families: *decomposition*, *ranking*, *vulnerability prediction*, and *ICL ranking/vulnerability* (with or without an explicit property name). We also mix in Mol-Instructions (Fang et al., 2023) to preserve general chemistry-language ability. The full Stage 1 corpus contains about 2M examples; detailed mixtures are in Appendix D.2. This stage makes the edit decision context-aware: the model reads the full molecular sequence while supervision remains fragment-level, so it learns to judge fragments in context rather than as global values.

4.2 Stage 2: learning how to edit

Stage 2 is trained on verified fragment replacement pairs from two sources. **SME+**, our context-conditioned extension of SME, groups fragment occurrences by their local attachment environment (ECFP radius-2) and searches for replacements that improve the score within that context bin, with global-pool fallback for sparse bins. All proposed replacements are filtered by atom-map consistency, and only verified low-to-high pairs are retained.

These pairs are also chained into multi-step trajectories used as in-context demonstrations during inference. The second source is **ChEMBL matched molecular pairs**: pairs sharing a scaffold, differing at one fragment, measured on the same target, with direction assigned by activity improvement. To avoid leakage, we exclude all pairs whose target description matches DRD2, JNK3, or GSK3-β.

All Stage 2 outputs follow a unified format: `Modification: <frag_src> » <frag_tgt>` followed by `Result: <tgt_smi>` with an optional score suffix (included in most samples, omitted in the rest for robustness). All SMILES spans use `<start_smiles>...<end_smiles>` markers for consistent atom-level tokenization. The Stage 2 data merges SME+ trajectories on five ADMET properties with ChEMBL pairs; the resulting Stage 1+2 checkpoint serves as the **backbone** for all downstream evaluations. Benchmark-specific continuation is described in Section 5.

4.3 Atom-level retokenization

Fragment-level supervision requires consistent fragment representation across molecules. Standard BPE tokenization breaks this: the same SMILES substring may be split differently depending on context. We therefore augment the Qwen3 vocabulary with a small atom-level SMILES lexicon (`<start_smiles>`, `<end_smiles>`, and `<smi>`-prefixed tokens for atoms, bonds, brackets, and ring digits). Inside marked SMILES spans, tokenization switches to a regex-based atomic parser. The new embeddings are warm-started from their literal counterparts so the first forward pass matches the original Qwen3 distribution. In Appendix B.5, we show 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 of (source, modification, result, score) tuples. At each step, the model is prompted with demonstrations sampled from this buffer—biased toward higher-scoring examples but tempered for diversity—and generates candidate modifications; the oracle scores the results and new tuples are written back. The buffer thus acts as an external memory of which local edits have worked under the current objective. The output grammar matches Stage 2, and demonstration-conditioned examples (including no-description settings) are already present during training, so the inference problem has the same structure as the training decomposition.

5 Experiments

5.1 Experimental setup

All experiments use Qwen3-0.6B as the base model, together with the atom-level 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.

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 atom-level tokenization (AtomTok), 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 1 shows FORGE achieves 2.882 SUM at the strictest similarity threshold ($\delta = 0.6$), outperforming Prompt-MolOpt (0.824) by a large margin. The gap between FORGE and Prompt-MolOpt grows as the similarity constraint tightens: at $\delta = 0$, the gain is small (4.587 vs 4.466), but at $\delta = 0.6$, 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. Ablations show that removing Stage 2 and the multi-turn continuation hurts performance

Table 1: Prompt-MolOpt performance at 3 similarity thresholds with full ablations. Best results under each similarity constraint are highlighted in bold.

Model	δ	ESOL	BBBP	hERG	lipop	Mutag	SUM↑
FORGE (AtomTok + 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**
AtomTok + 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
AtomTok + 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

at all thresholds, indicating that explicit modification is the main source of improvement. Removing Stage 1 also lowers performance, though more mildly, suggesting that fragment ranking mainly helps identify where to edit.

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 and 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 2: 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 and Grover, 2024)	11.71
Genetic GFN (Kim et al., 2024)	11.56
Augmented Memory (Guo and 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 2, 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 and Lead Optimization

QED-DRD2 isolates whether the model should output a full rewritten molecule or an explicit fragment modification. We compare two variants sharing the same backbone, training data, and loss, differing only in output format: *end2end* predicts the whole target SMILES, while *modif.* predicts fragment replacement. Table 12 shows the modification variant outperforms end-to-end by 3.0–5.0 points and beats all prior methods at every threshold, with the largest gap at the strictest similarity constraint ($\delta = 0.6$: 30.41% vs 27.32%). This supports the central design choice: under similarity constraints, explicit fragment replacement is a better output space than free-form generation.

Lead Optimization tests FORGE on real-target docking. Following Wang et al. (2022), we optimize docking scores subject to QED ≥ 0.6, SA ≤ 4, and Tanimoto similarity $\geq \delta$ on five protein targets (*parp1*, *fa7*, *5ht1b*, *braf*, *jak2*) with three seeds per target. FORGE achieves the best score on 9/15 runs at $\delta = 0.4$ and 11/15 at $\delta = 0.6$, beating GenMol (Lee et al., 2025) and Graph GA (Jensen, 2019) at both similarity floors. The advantage is most consistent on *parp1*, *fa7*, and *jak2*. Detailed per-seed docking scores are in Appendix B.3.

5.5 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 3 summarizes the comparative evaluation. At 0.6B parameters, FORGE (AtomTok + 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 con-

Table 3: 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)	AtomTok + FT	1.07 / 90	0.81 / 82	0.18 / 85	0.32 / 82	0.13 / 67	0.28 / 82
	AtomTok + S2 + FT	1.02 / 93	1.04 / **95**	0.19 / 88	0.36 / 87	0.24 / 83	0.33 / 89
	FORGE (AtomTok + S1 + S2) + FT	1.06 / 94	1.03 / **95**	**0.25 / 93**	**0.37 / 91**	**0.27 / 86**	**0.34 / 94**

trast, 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 (AtomTok + 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. The pattern flips with the similarity threshold: at $\delta = 0$, global aggregation is slightly more useful, but once the similarity floor becomes difficult, SME+ wins by $+0.65$ SUM at $\delta = 0.4$ and $+0.83$ at $\delta = 0.6$ (detailed results in Appendix B.1, Tables 10 and 11). 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 factor among backbone upgrade ($+0.530$), SME+ data ($+0.833$), and modification format ($+0.695$). 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.

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

The ablations in Prompt-MolOpt and ChemCoT-Bench suggest Stage 1 and Stage 2 contribute differently across benchmarks. Table 4 and Figure 3 summarize that pattern across representative settings.

Table 4: Marginal gains of Stage 1 and 2 across settings.

Setting	AtomTok	+ 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

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 ChemCoT-

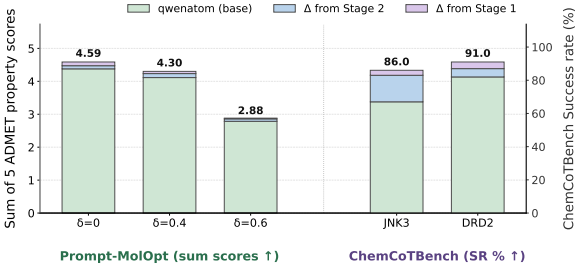


Figure 3: Marginal gains of Stage 1 and Stage 2 across representative benchmarks.

Bench (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.4 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.

7 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 and 2 do not span every property. The backbone serves as a powerful initialization for training-free adaptation via ICL (e.g., in PMO-1k and Lead Optimization), but it is not a limitless generalist; highly novel or difficult properties may still need a per-evaluation SFT step.

8 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 across both training and training-free settings 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 7, and porting the fragment-replacement supervision format to reaction-prediction tasks where matched-pair data is also abundant.

References

Nathan Brown, Marco Fiscato, Marwin HS Segler, and Alain C Vaucher. 2019. Guacamol: benchmarking models for de novo molecular design. *Journal of chemical information and modeling*, 59(3):1096–1108.

Ziqi Chen, Martin Renqiang Min, Srinivasan Parthasarathy, and Xia Ning. 2021. A deep generative model for molecule optimization via one fragment modification. *Nature machine intelligence*, 3(12):1040–1049.

Seyone Chithrananda, Gabriel Grand, and Bharath Ramsundar. 2020. Chemberta: large-scale self-supervised pretraining for molecular property prediction. *arXiv preprint arXiv:2010.09885*.

Andrew Dalke, Jérôme Hert, and Christian Kramer. 2018. mmpdb: An open-source matched molecular pair platform for large multiproperty data sets. *Journal of Chemical Information and Modeling*, 58(5):902–910. PMID: 29770697.

Suresh Dara, Swetha Dhamercherla, Surender Singh Jadav, CH Madhu Babu, and Mohamed Jawed Ahsan. 2022. Machine learning in drug discovery: a review. *Artificial intelligence review*, 55(3):1947–1999.

Yin Fang, Xiaozhuan Liang, Ningyu Zhang, Kangwei Liu, Rui Huang, Zhuo Chen, Xiaohui Fan, and Huajun Chen. 2023. Mol-Instructions: A Large-Scale Biomolecular Instruction Dataset for Large Language Models. *arXiv e-prints*, arXiv:2306.08018.

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

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

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

Philipp Guevorguian, Menua Bedrosian, Tigran Fahradyan, Gayane Chilingaryan, Hrant Khachatrian, and Armen Aghajanyan. 2024. Small molecule optimization with large language models. *arXiv preprint arXiv:2407.18897*.

Jeff Guo and Philippe Schwaller. 2023. Augmented Memory: Capitalizing on Experience Replay to Accelerate De Novo Molecular Design. *arXiv e-prints*, arXiv:2305.16160.

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

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. 2018a. Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning*, pages 2323–2332. PMLR.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. 2019. Hierarchical graph-to-graph translation for molecules. *arXiv preprint arXiv:1907.11223*.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. 2020. Hierarchical generation of molecular graphs using structural motifs. In *International conference on machine learning*, pages 4839–4848. PMLR.

Wengong Jin, Kevin Yang, Regina Barzilay, and Tommi Jaakkola. 2018b. Learning multimodal graph-to-graph translation for molecular optimization. *arXiv preprint arXiv:1812.01070*.

Hyeonah Kim, Minsu Kim, Sanghyeok Choi, and Jinkyoo Park. 2024. Genetic-guided gflownets for sample efficient molecular optimization. *Advances in Neural Information Processing Systems*, 37:42618–42648.

Seul Lee, Karsten Kreis, Srimukh Prasad Veccham, Meng Liu, Danny Reidenbach, Yuxing Peng, Saee Paliwal, Weili Nie, and Arash Vahdat. 2025. Genmol: A drug discovery generalist with discrete diffusion. *arXiv preprint arXiv:2501.06158*.

Hao Li, He Cao, Bin Feng, Yanjun Shao, Xiangru Tang, Zhiyuan Yan, Li Yuan, Yonghong Tian, and Yu Li. 2025. Beyond chemical qa: Evaluating llm's chemical reasoning with modular chemical operations. *arXiv preprint arXiv:2505.21318*.

Gang Liu, Jie Chen, Yihan Zhu, Michael Sun, Tengfei Luo, Nitesh V Chawla, and Meng Jiang. 2025. Graph diffusion transformers are in-context molecular designers. *arXiv preprint arXiv:2510.08744*.

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

Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. 2017. Molecular De Novo Design through Deep Reinforcement Learning. *arXiv e-prints*, arXiv:1704.07555.

Qizhi Pei, Wei Zhang, Jinhua Zhu, Kehan Wu, Kaiyuan Gao, Lijun Wu, Yingce Xia, and Rui Yan. 2023. 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*, pages 1102–1123. Association for Computational Linguistics.

Jannik Philipp Roth. 2026. Chemically interpretable explanations for molecular property prediction via fragment-level shapley values. *ChemRxiv*, 2026(0421).

Niranjan Srinivas, Andreas Krause, Sham M. Kakade, and Matthias Seeger. 2009. Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. *arXiv e-prints*, arXiv:0912.3995.

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

Haorui Wang, Marta Skreta, Cher-Tian Ser, Wenhao Gao, Lingkai Kong, Felix Strieth-Kalthoff, Chenru Duan, Yuchen Zhuang, Yue Yu, Yanqiao Zhu, Yuanqi Du, Alán Aspuru-Guzik, Kirill Neklyudov, and Chao Zhang. 2024. Efficient Evolutionary Search Over Chemical Space with Large Language Models. *arXiv e-prints*, arXiv:2406.16976.

Zichao Wang, Weili Nie, Zhuoran Qiao, Chaowei Xiao, Richard Baraniuk, and Anima Anandkumar. 2022. Retrieval-based Controllable Molecule Generation. *arXiv e-prints*, arXiv:2208.11126.

Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay Pande. 2017. MoleculeNet: A Benchmark for Molecular Machine Learning. *arXiv e-prints*, arXiv:1703.00564.

Zhenxing Wu, Jike Wang, Hongyan Du, Dejun Jiang, Yu Kang, Dan Li, Peichen Pan, Yafeng Deng, Dongsheng Cao, Chang-Yu Hsieh, and 1 others. 2023. Chemistry-intuitive explanation of graph neural networks for molecular property prediction with substructure masking. *Nature communications*, 14(1):2585.

Zhenxing Wu, Odin Zhang, Xiaorui Wang, Li Fu, Huifeng Zhao, Jike Wang, Hongyan Du, Dejun Jiang, Yafeng Deng, Dongsheng Cao, and 1 others. 2024. Leveraging language model for advanced multiproperty molecular optimization via prompt engineering. *Nature Machine Intelligence*, 6(11):1359–1369.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Xinwu Ye, Yicheng Mao, Jia Zhang, Yimeng Liu, Li Hao, Fang Wu, Zhiwei Li, Yuxuan Liao, Zehong Wang, Yingcheng Wu, and 1 others. 2026. Latentchem: From textual cot to latent thinking in chemical reasoning. *arXiv preprint arXiv:2602.07075*.

Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay Pande, and Jure Leskovec. 2018. Graph convolutional policy network for goal-directed molecular graph generation. *Advances in neural information processing systems*, 31.

Botao Yu, Frazier N Baker, Ziqi Chen, Xia Ning, and Huan Sun. 2024. Llasmol: Advancing large language models for chemistry with a large-scale, comprehensive, high-quality instruction tuning dataset. *arXiv preprint arXiv:2402.09391*.

Odin Zhang, Haitao Lin, Hui Zhang, Huifeng Zhao, Yufei Huang, Chang-Yu Hsieh, Peichen Pan, and Tingjun Hou. 2024. Deep lead optimization: leveraging generative ai for structural modification. *Journal of the American Chemical Society*, 146(46):31357–31370.

Gengmo Zhou, Zhifeng Gao, Qiankun Ding, Hang Zheng, Hongteng Xu, Zhewei Wei, Linfeng Zhang, and Guolin Ke. 2023. Uni-mol: A universal 3d molecular representation learning framework. In *The Eleventh International Conference on Learning Representations*.

A More Motivation Details

A.1 Prompt variants details

To ensure a fair comparison, we adopt the evaluation pipeline introduced by MOLLEO. The model is conditioned using task-specific prompts; using QED optimization as an example, the prompt format is defined as follows:

> **Instruction:** I have two molecules and their QED scores. The QED score measures the drug-likeness of the molecule. Please propose a new molecule that has a higher QED score. You can either make crossover and mutations based on the given molecules or just propose a new molecule based on your knowledge.
>
> **Input:** [CON(C)C(=O)Nc1ccc(Cl)c(Cl)c1, 0.82] ... [O=C(CCC)CF, 0.53]

All experiments are executed across three independent runs. The detailed results and aggregate metrics are reported in Table 5.

A.2 Per-property Variance Reduction

This section provides the detailed variance reduction (VR) results referenced in Section 3.2. We test the models on 10k mol pool of ZINC dataset. Fragments are divided by BRICS, EFG and Murcko methods. Uni-Mol and ChemBERTa are trained on MoleculeNet. RGCN is taken from SME.

B More experiment details

B.1 Prompt-MolOpt detailed results

We show the effect of multi-turn context. 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 9. This is consistent with the

Table 5: Performance comparison of Qwen variants across PMO10k. Bold values indicate the best performance in each task.

Task / Metric	Qwen	Qwen-QED	Qwen-Unk
Drug Rediscovery			
albuterol_similarity	**0.969**$_{0.006}$	0.888$_{0.002}$	0.893$_{0.008}$
celecoxib_rediscovery	0.645$_{0.087}$	**0.705**$_{0.079}$	0.597$_{0.032}$
median1	0.307$_{0.002}$	0.298$_{0.010}$	**0.311**$_{0.015}$
median2	**0.296**$_{0.004}$	0.293$_{0.005}$	**0.296**$_{0.011}$
mestranol_similarity	**0.603**$_{0.026}$	0.590$_{0.025}$	0.579$_{0.011}$
thioxithene_rediscovery	0.531$_{0.025}$	0.522$_{0.028}$	**0.556**$_{0.008}$
troglitazone_rediscovery	**0.448**$_{0.043}$	0.380$_{0.014}$	0.414$_{0.040}$
Drug MPO			
amlodipine_mpo	**0.679**$_{0.025}$	0.654$_{0.019}$	0.665$_{0.015}$
fexofenadine_mpo	**0.802**$_{0.009}$	0.778$_{0.002}$	0.794$_{0.008}$
osimertinib_mpo	**0.840**$_{0.003}$	0.834$_{0.020}$	0.837$_{0.002}$
perindopril_mpo	**0.531**$_{0.005}$	0.522$_{0.025}$	0.523$_{0.001}$
ranolazine_mpo	0.728$_{0.016}$	0.730$_{0.017}$	**0.762**$_{0.006}$
sitagliptin_mpo	0.585$_{0.030}$	0.582$_{0.026}$	**0.621**$_{0.008}$
zaleplon_mpo	**0.510**$_{0.001}$	0.484$_{0.010}$	0.504$_{0.010}$
Structure Constrained Design			
deco_hop	0.684$_{0.097}$	0.693$_{0.085}$	**0.722**$_{0.064}$
isomers_c7h8n2o2	0.959$_{0.006}$	**0.967**$_{0.005}$	0.917$_{0.041}$
isomers_c9h10n2o2pf2cl	**0.878**$_{0.018}$	0.836$_{0.019}$	0.870$_{0.014}$
scaffold_hop	**0.552**$_{0.023}$	**0.552**$_{0.005}$	0.516$_{0.013}$
valsartan_smarts	0.000$_{0.000}$	0.000$_{0.000}$	0.000$_{0.000}$
Drug Design			
drd2	**0.979**$_{0.002}$	0.973$_{0.005}$	0.968$_{0.004}$
gsk3b	**0.888**$_{0.031}$	0.855$_{0.020}$	0.780$_{0.095}$
jnk3	**0.732**$_{0.010}$	0.583$_{0.078}$	0.559$_{0.122}$
qed	0.941$_{0.000}$	**0.942**$_{0.000}$	0.941$_{0.000}$
SUM	**15.087**	14.660	14.628

Table 6: 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 $\Delta = \text{real} - \text{shuf}$ 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 7: 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 6. *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%

role of dialogue context in preserving a coherent sequence of local modifications.

We also isolate the effect of context-conditioned fragment effects (SME+) versus global fragment statistics (SME-style aggregation). Table 10 shows that at $\delta = 0$, where the model can move to distant high-scoring regions, global aggregation is slightly more useful (3.977 vs 3.781). However, once the similarity floor becomes difficult, SME+ wins decisively: +0.654 SUM at $\delta = 0.4$ and +0.833 at $\delta = 0.6$. This pattern confirms that within-context supervision matches the local-edit regime enforced by similarity constraints.

Table 11 decomposes the total gain over the

Table 8: 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%

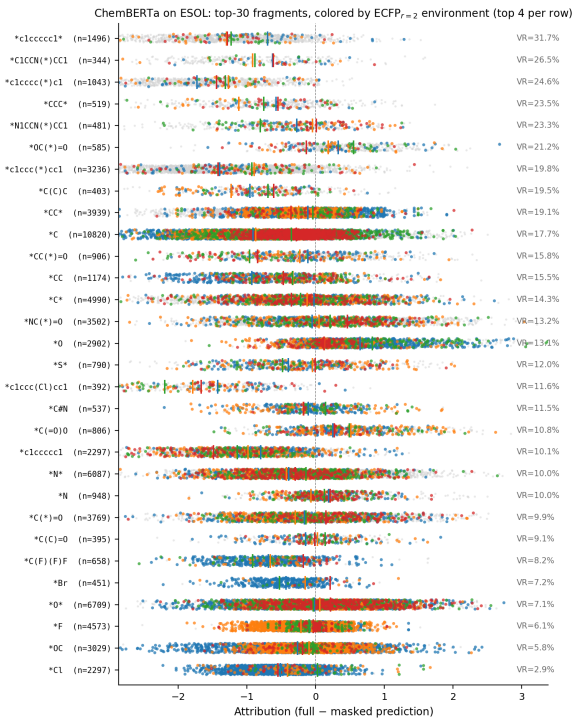


Figure 4: Fragment decoupling of ChemBERTa on ESOL

Table 9: 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

Prompt-MolOpt baseline at $\delta = 0.6$ into three factors: backbone upgrade (+0.530), SME+ data (+0.833), and modification format with Stage 1+2 (+0.695). Switching to SME+ data alone contributes +0.833 SUM, the single largest factor. This shows that FORGE benefits not only from a better output interface, but also—and primarily—from supervision that matches the local nature of similarity-constrained editing.

Table 10: 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	$+0.833$

Table 11: 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

B.2 QED-DRD2 detailed results

QED-DRD2 is a dual-objective optimization task that requires simultaneously improving QED and DRD2 binding affinity while maintaining structural similarity to the source molecule. Table 12 reports success rates at three similarity thresholds ($\delta \in \{0.4, 0.5, 0.6\}$). FORGE with the modification output format (fragment replacement) consistently outperforms the end-to-end SMILES generation variant by 3.0–5.0 percentage points across all thresholds. The gap is largest at $\delta = 0.6$ (30.41% vs 27.32%), where preserving most of the source structure is most critical. Both FORGE variants outperform all prior graph-based methods (JTNN, HierG2G, Modof, Modofm) and the original Prompt-MolOpt baseline. This result isolates the value of explicit fragment modification: when the task requires staying close to the source, generating a local edit is more effective than rewriting the entire molecule.

B.3 Lead Optimization detailed results

Lead Optimization evaluates FORGE on real-target docking, the setting closest to drug-discovery practice. Following Wang et al. (2022), we optimize

Table 12: 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**

docking scores subject to QED ≥ 0.6, SA ≤ 4, and Tanimoto similarity $\geq \delta$ on five protein targets (*parp1*, *fa7*, *5ht1b*, *braf*, *jak2*) with three seeds per target under 1000 oracle call limit, giving 30 tasks per δ. Table 13 reports per-seed docking scores (kcal/mol; lower is better). FORGE achieves the best score on 9/15 runs at $\delta = 0.4$ and 11/15 at $\delta = 0.6$, beating GenMol (Lee et al., 2025) and Graph GA (Jensen, 2019) at both similarity constrains.

The advantage is most consistent on *parp1*, *fa7*, and *jak2*, where FORGE wins nearly all per-seed runs. For example, on *parp1* with seed -7.8, FORGE reaches -11.90 at $\delta = 0.4$ and -10.93 at $\delta = 0.6$, while GenMol degrades from -11.00 to -9.70. A similar pattern holds on *jak2* and *fa7*. 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.

There are two cases where FORGE underperforms GenMol. On *5ht1b*, GenMol holds a 0.2–1.9 kcal/mol advantage on some seeds. 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). Graph GA, which mutates without context, trails on most targets and frequently fails to satisfy the similarity floor at $\delta = 0.6$ (indicated by "–" entries).

B.4 PMO-1k per-task results

Table 14 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 and Grover, 2024), Genetic GFN (Kim et al., 2024), Augmented Memory (Guo and Schwaller, 2023), MOLLEO (Wang et al., 2024), FORGE backbone (*qwenatom + S1 + S2*), and the FORGE Stage-1-ablation (*qwenatom + S2*). Means across 5 runs

Table 13: 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**

are reported. The four tasks where MOLLEO outperforms FORGE — *albuterol_similarity*, *amlodipine_mpo*, *drd2*, *perindopril_mpo* — are the named-drug-recall failure modes. As all baseline methods fail on *valsartan_smarts* task, we also skip this task.

B.5 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.

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 16.

For inference: vLLM with temperature τ=0.7, top-p=0.95, max generation length 4096. 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 section 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 18 and 19.

D.1 Training Data and Supervision Details

Table 17 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 task families. Stage 1 builds four task families from fragment attributions: (1) **Decomposition**: split a molecule into atom-mapped fragments; (2) **Ranking**: sort fragments by contribution to a target property; (3) **Vulnerability prediction**: return the single weakest fragment; (4) **ICL ranking/vulnerability**: perform the same tasks conditioned on in-context demonstrations, with or without an explicit property name.

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, TPSA and etc. 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, 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 MMPA.

Table 14: Performance comparison of various methods across PMO-1k tasks. Bold values indicate the best performance in each task.

Task / Metric	GP-BO	Graph GA	REINVENT	LICO	Genetic GFN	Aug. Mem.	MOLLEO	FORGE (S1+S2)	FORGE (S2)
Drug Rediscovery									
albuterol_similarity	$0.636_{0.106}$	$0.583_{0.065}$	$0.496_{0.020}$	$0.656_{0.125}$	$0.664_{0.054}$	$0.557_{0.048}$	$\mathbf{0.886}_{0.023}$	$0.779_{0.046}$	$0.764_{0.023}$
celecoxib_rediscovery	$0.411_{0.046}$	$0.424_{0.049}$	$0.370_{0.029}$	$\mathbf{0.447}_{0.073}$	$\mathbf{0.447}_{0.028}$	$0.385_{0.027}$	$0.402_{0.003}$	$0.428_{0.029}$	$0.441_{0.017}$
median1	$0.213_{0.020}$	$0.208_{0.015}$	$0.205_{0.012}$	$0.217_{0.019}$	$0.219_{0.008}$	$0.219_{0.014}$	$0.236_{0.021}$	$\mathbf{0.271}_{0.021}$	$0.250_{0.022}$
median2	$0.203_{0.009}$	$0.181_{0.009}$	$0.188_{0.010}$	$0.193_{0.009}$	$0.204_{0.011}$	$0.184_{0.010}$	$0.191_{0.009}$	$\mathbf{0.221}_{0.009}$	$0.220_{0.011}$
mestranol_similarity	$0.427_{0.025}$	$0.362_{0.017}$	$0.379_{0.026}$	$0.423_{0.016}$	$0.414_{0.022}$	$0.393_{0.021}$	$0.399_{0.020}$	$0.450_{0.019}$	$\mathbf{0.453}_{0.043}$
thiothixene_rediscovery	$0.351_{0.039}$	$0.322_{0.023}$	$0.311_{0.021}$	$0.343_{0.035}$	$0.377_{0.015}$	$0.336_{0.033}$	$\mathbf{0.416}_{0.075}$	$0.400_{0.023}$	$0.374_{0.020}$
troglitazone_rediscovery	$\mathbf{0.313}_{0.018}$	$0.267_{0.015}$	$0.246_{0.009}$	$0.292_{0.028}$	$0.277_{0.015}$	$0.262_{0.012}$	$0.302_{0.022}$	$0.288_{0.018}$	$0.286_{0.021}$
Drug MPO									
amlodipine_mpo	$0.519_{0.014}$	$0.501_{0.016}$	$0.472_{0.008}$	$0.541_{0.026}$	$0.534_{0.019}$	$0.489_{0.009}$	$\mathbf{0.637}_{0.023}$	$0.551_{0.004}$	$0.554_{0.006}$
fexofenadine_mpo	$0.707_{0.021}$	$0.666_{0.009}$	$0.650_{0.007}$	$0.700_{0.023}$	$0.682_{0.021}$	$0.679_{0.021}$	$0.674_{0.002}$	$0.740_{0.011}$	$\mathbf{0.742}_{0.009}$
osimertinib_mpo	$0.766_{0.006}$	$0.751_{0.005}$	$0.737_{0.007}$	$0.759_{0.008}$	$0.763_{0.008}$	$0.761_{0.006}$	$0.779_{0.006}$	$\mathbf{0.794}_{0.002}$	$0.787_{0.007}$
perindopril_mpo	$0.458_{0.019}$	$0.435_{0.016}$	$0.404_{0.009}$	$0.473_{0.009}$	$0.462_{0.033}$	$0.422_{0.013}$	$\mathbf{0.655}_{0.054}$	$0.509_{0.017}$	$0.522_{0.015}$
ranolazine_mpo	$0.701_{0.023}$	$0.620_{0.014}$	$0.574_{0.044}$	$0.687_{0.029}$	$0.623_{0.022}$	$0.614_{0.033}$	$0.640_{0.000}$	$0.680_{0.013}$	$\mathbf{0.690}_{0.014}$
sitagliptin_mpo	$0.232_{0.083}$	$0.229_{0.053}$	$0.261_{0.026}$	$0.315_{0.097}$	$0.227_{0.041}$	$0.245_{0.030}$	$0.193_{0.073}$	$0.300_{0.018}$	$\mathbf{0.337}_{0.041}$
zaleplon_mpo	$0.392_{0.034}$	$0.374_{0.024}$	$0.406_{0.017}$	$0.404_{0.022}$	$0.400_{0.014}$	$0.415_{0.013}$	$0.392_{0.003}$	$0.435_{0.009}$	$\mathbf{0.436}_{0.007}$
Structure Constrained Design									
deco_hop	$0.593_{0.013}$	$0.581_{0.006}$	$0.572_{0.006}$	$0.596_{0.010}$	$0.604_{0.017}$	$0.579_{0.010}$	$0.588_{0.007}$	$\mathbf{0.624}_{0.019}$	$\mathbf{0.624}_{0.024}$
isomers_c7h8n2o2	$0.545_{0.158}$	$0.735_{0.112}$	$0.725_{0.064}$	$\mathbf{0.779}_{0.099}$	$0.738_{0.039}$	$0.661_{0.039}$	$0.737_{0.043}$	$0.723_{0.024}$	$0.652_{0.020}$
isomers_c9h10n2o2pf2cl	$0.599_{0.059}$	$0.630_{0.086}$	$0.630_{0.032}$	$0.672_{0.075}$	$0.656_{0.075}$	$0.596_{0.066}$	$0.635_{0.017}$	$\mathbf{0.713}_{0.020}$	$0.712_{0.009}$
scaffold_hop	$0.478_{0.009}$	$0.461_{0.008}$	$0.447_{0.010}$	$0.480_{0.008}$	$0.485_{0.015}$	$0.460_{0.010}$	$0.473_{0.000}$	$\mathbf{0.532}_{0.048}$	$0.511_{0.027}$
valsartan_smarts	$0.000_{0.000}$	$0.000_{0.000}$	$0.000_{0.000}$	$0.000_{0.000}$	$0.000_{0.000}$	$0.000_{0.000}$	$0.000_{0.000}$	$0.000_{0.000}$	$0.000_{0.000}$
Drug Design									
drd2	$0.857_{0.080}$	$0.833_{0.065}$	$0.775_{0.086}$	$0.859_{0.066}$	$0.809_{0.045}$	$0.795_{0.024}$	$\mathbf{0.910}_{0.017}$	$0.816_{0.013}$	$0.818_{0.029}$
gsk3b	$0.611_{0.059}$	$0.523_{0.047}$	$0.589_{0.063}$	$0.617_{0.063}$	$\mathbf{0.637}_{0.018}$	$0.539_{0.097}$	$0.397_{0.013}$	$0.620_{0.070}$	$0.581_{0.045}$
jnk3	$0.346_{0.067}$	$0.301_{0.071}$	$0.315_{0.042}$	$0.336_{0.051}$	$0.409_{0.165}$	$0.294_{0.110}$	$0.186_{0.076}$	$\mathbf{0.611}_{0.032}$	$0.584_{0.021}$
qed	$0.912_{0.010}$	$0.914_{0.007}$	$0.921_{0.002}$	$0.925_{0.005}$	$0.928_{0.002}$	$0.923_{0.002}$	$0.919_{0.006}$	$\mathbf{0.937}_{0.001}$	$\mathbf{0.937}_{0.001}$
SUM (↑)	11.27	10.90	10.68	11.71	11.56	10.81	11.65	**12.42**	12.27

Table 15: 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

Table 16: 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

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 18; smaller sources are oversampled to match their target ratio.

Table 17: 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 18: 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%

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 ≥ 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. 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: C=CCNC(=O)CN1CC(S(=O)(=O)N2CCNCC2)C1 List of fragments to be ranked: [3*]C[4*]....
>
> **Output:** [4*]N([5*])[10*] (score: 0.14) → ... → [9*]CC[11*] (score: 0.0)
>
> **Task:** ranking

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 build: *ranking* (sort fragments by attribution score) and *decomposition* (predict the fragment list induced by the mask). Example:

> **Instruction:** Analyze the molecule below and decompose it into fragments according to the 'EFG' rules, listing the corresponding atom indices for each fragment.
>
> **Input:** Molecule SMILES: O=C(CCCCCC(NC(=O)c1ccco1)C(=O)NCc1ccccc1)NO
>
> **Output:** Fragment: [1*]C(=O)NO.[1*]CCCCCC([2*])[4*].[2*]NC([3*])=O.[3*]c1ccco1.[4*]C(=O)N[5*].[5*]C[6*].[6*]c1ccccc1 Atom Indices: (0, 1, 26, 27).(2, 3, 4, 5, 6, 7).(8, 9, 10).(11, 12, 13, 14, 15).(16, 17, 18).(19,).(20, 21, 22, 23, 24, 25)
>
> **Task:** decomposition

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. Example:

Table 19: 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)	~412k	60%

> **Instruction:** Which of the following fragments on the given scaffold contributes least to the bioactivity and should be replaced?
>
> **Input:**
>
> - **Example 1:**
> Scaffold: O=c1[nH]c(=S)ccn1[C@@H]1O[C@H](C[1*])[C@@H](O)[C@H]1O
> Fragment Ranking: O=P(O)(O)OP(=O)(O)OP(=O)(O)O[1*].O=P(O)(OP(=O)(O)C(Cl)(Cl)P(=O)(O)O)O[1*].O=P(O)(OP(=O)(O)C(F)(F)P(=O)(O)O)O[1*].O=P(O)(O)O[1*]
>
> ...
>
> - **Example 5:**
> Scaffold: O=c1ccn([C@@H]2O[C@H](COP(=O)(O)OP(=O)(O)OP(=O)(O)O)[C@@H](O)[C@H]2[1*])c(=O)[nH]1
> Fragment Ranking: [N-]=[N+]=N[1*].N[1*].NC(=O)N[1*]
>
> - **Query:**
> Scaffold: O=C(Cn1c(=O)ccn([C@@H]2O[C@H](C[1*])[C@@H](O)[C@H]2O)c1=O)c1ccccc1
> Fragments to rank: O=P(O)(O)O[1*].O=P(O)(OP(=O)(O)C(Cl)(Cl)P(=O)(O)O)O[1*].O=P(O)(OP(=O)(O)C(F)(F)P(=O)(O)O)O[1*].O=P(O)(O)OP(=O)(O)O[1*]
>
> **Output:** O=P(O)(O)O[1*]
>
> **Task:** mmp_icl_with_desc_vulnerability

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

Stage 2 teaches the model to propose a fragment-replacement edit and the resulting molecule. Sources and proportions are listed in Table 19.

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* → *frag_tgt* edit is capped at five occurrences to prevent a small set of common substitutions from dominating the loss. Example:

Instruction: Increase the 'ESOL' (aqueous solubility) of the given molecule.

Input:
Molecule SMILES: CON(C)C(=O)Nc1ccc(Cl)c(Cl)c1

Output:

- **Modification:**
  Remain: [1*]c1ccc(NC(=O)N(C)OC)cc1Cl,
  modify: [1*]Cl → [1*]CNC
- **Result Molecule:**
  CNCc1ccc(NC(=O)N(C)OC)cc1Cl

Property: ESOL

ChEMBL MMP optimization. Matched-molecular pairs are mined across ChEMBL targets with pChEMBL $\geq$ 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). Example:

Instruction: Study the optimization examples below. Predict a suitable modification for the query molecule.

Input:

- **Example 1:**
  Input: CC(C(=O)O)N(Cc1ccc([N+]12=O)[O-])cc1)S(=O)(=O)C(Cl)(Cl)Cl (activity: 0.28)
  Modif.: ClC(Cl)(Cl)[1*] → CC(=O)c1ccc([1*])cc1
  Result: CC(=O)c1ccc(S(=O)(=O)N(Cc2ccc([N+]13=O)[O-])cc2)C(C)C(=O)O)cc1 (activity: 0.37)

  ...
- **Example 5:**
  Input: CC(=O)Nc1ccc(S(=O)(=O)N(Cc2ccc([N+]22=O)[O-])cc2)C(C)C(=O)O)cc1 (activity: 0.33)
  Modif.: CC(=O)Nc1ccc([1*])cc1 → c1cnc2c([1*])cccc2c1
  Result: CC(C(=O)O)N(Cc1ccc([N+]23=O)[O-])cc1)S(=O)(=O)c1cccc2cccnc12 (activity: 0.43)
- **Query:** Optimize
  CC(C(=O)O)N(Cc1ccccc1[N+]24=O)[O-])S(=O)(=O)c1cccc([N+]25=O)[O-])c1 (activity: 0.28)

Output:
Modification: O=[N+]([O-])[1*] → O=C(O)[1*]
Result: CC(C(=O)O)N(Cc1ccccc1[N+]26=O)[O-])S(=O)(=O)c1cccc(C(=O)O)c1 (activity: 0.37)

Task: chembl_icl_no_desc_value

Multi-turn variant. For Prompt-MolOpt, we additionally train a multi-turn continuation that reuses the verified SME+ trajectories. The training data statistics for each property are summarized as Table 20

Table 20: 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

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)
...
```

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 Code and Data Availability

The source code, model checkpoints, and datasets for FORGE are publicly available to support reproducibility.

- **Codebase:** The complete repository, including training and evaluation pipelines, is hosted at https://anonymous.4open.science/r/FORGE-C368.
- **Model Weights:** The pre-trained model checkpoints are available at

https://anonymous-hf.up.railway.app/a/g485q8h89nb8/.

- **Training Data:** The curated datasets used for training and benchmarking are accessible at https://anonymous-hf.up.railway.app/a/ybhbweia1wh0/.