
PMO-Dock: Benchmarking Docking, Specificity, and Generalization in Molecular Optimization

Gor Simonyan*
YerevaNN
Yerevan State University

Tatevik Abrahamyan*
YerevaNN
Yerevan State University

Narek Abelyan
YerevaNN
Russian-Armenian University

Tigran Fahradyan
University of Cambridge

Hrant Khachatryan
YerevaNN
Yerevan State University

Abstract

The Practical Molecular Optimization (PMO) benchmark standardized evaluation in molecular optimization, but it is built on simple property-based oracles that do not capture structure-based drug design. As the field has moved to docking-based objectives, evaluation practice has become unstandardized, and different methods report results on different docking tasks, often without a shared protocol for fair comparison or sample-efficiency constraints. To address this, we introduce **Practical Molecular Optimization for Docking (PMO-Dock)**, a benchmark and protocol for docking-based optimization consisting of 25 tasks covering hit generation, lead optimization, and a new specificity task requiring strong on-target binding while penalizing off-target interactions. The protocol separates model development from final evaluation: validation tasks are used for hyperparameter tuning, and test tasks are held out for final evaluation and are not used during development. Evaluation on the test tasks is done under a strict oracle budget to reflect realistic drug-discovery settings. We benchmark four diverse high-performing methods spanning different optimization paradigms, Saturn (reinforcement learning), GenMol (discrete diffusion), Genetic-guided GFlowNet, and ChemLlama (LLM-based evolutionary search). Our results show no universally best method across tasks, additional analyses of hyperparameter transfer (cross-task and cross-category) and sensitivity show that transfer is often limited and varies by method, motivating method-dependent selection strategies. This benchmark provides a concrete standard for evaluating generalization to unseen tasks and sample efficiency under a limited test-time oracle budget.

1 Introduction and Related Work

Small molecule optimization is a central challenge in modern drug discovery, requiring algorithms to efficiently search vast chemical spaces under complex, often computationally expensive objectives. Over the past decade, numerous optimization methods have been proposed, but their evaluation has been fragmented across different tasks and protocols, making direct comparison difficult.

The Practical Molecular Optimization (PMO) benchmark [Gao et al., 2022] addressed this by establishing rigorous evaluation standards. PMO compiled 23 optimization tasks with well-specified oracles and evaluated 23 methods using unified protocols: strict oracle budgets, extensive hyperparam-

*Denotes equal first authorship. Correspondence to Gor Simonyan, Tatevik Abrahamyan (gor@yerevann.com, tatevikabrahamyan@yerevann.com).

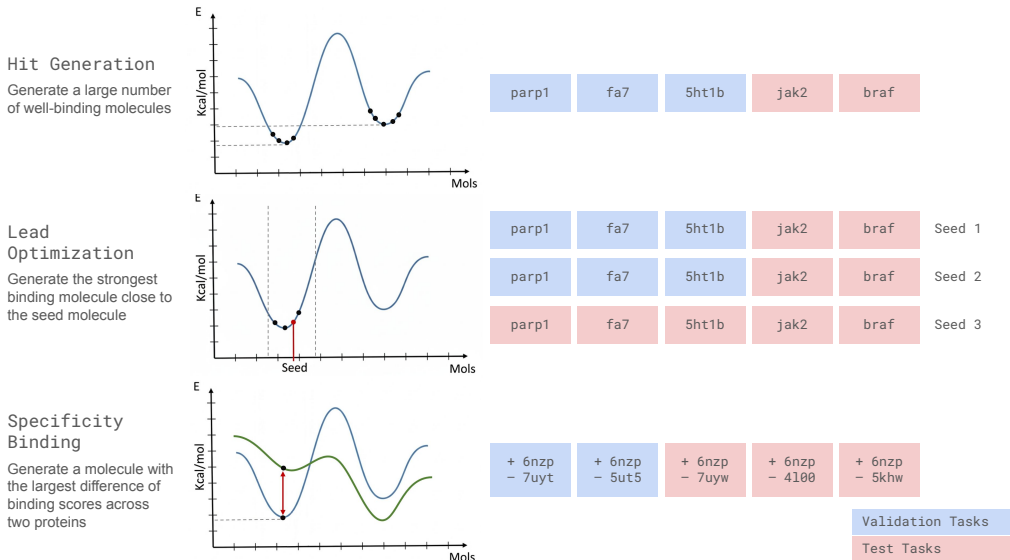


Figure 1: Overview of all 25 PMO-Dock tasks organized by oracle category. Each task is shown as a separate cell, color-coded by split: validation tasks (light blue) are used for algorithm development and hyperparameter tuning, while test tasks (light red) are strictly held out for evaluation with limited evaluation budgets. Hit Generation tasks maximize the fraction of molecules meeting docking thresholds, Lead Optimization tasks optimize docking scores while maintaining similarity to seed molecules and Specificity Binding tasks focus on selective target engagement.

eter sweeps, and fixed metrics. This standardization enabled meaningful progress, with methods such as Augmented Memory [Guo and Schwaller, 2024a], Genetic-guided GFlowNets [Kim et al., 2024], Chemlactica [Guevorguian et al., 2024], and MOLLEO [Wang et al., 2025] advancing state-of-the-art on the benchmark. However, PMO’s objectives are limited to simple combinations of RDKit-based properties, such as molecular similarity, isomer counts, and drug-likeness metrics. While computationally efficient, these objectives lack the practical relevance of real-world drug discovery, such as docking-based evaluations against protein targets that PMO does not cover.

Recently, an increasing number of methods adopt molecular docking objectives. Methods including Augmented Memory [Guo and Schwaller, 2024a], Beam Enumeration [Guo and Schwaller, 2024b], Saturn [Guo and Schwaller, 2024c], GenMol [Lee et al., 2025], and Chemlactica [Guevorguian et al., 2024] have explored docking score optimization. Ciepliński et al. [2023] attempted to bridge this gap by introducing a benchmark based on SMINA docking for de novo drug design, in which a model trained on known ligands for a target proposes a novel batch of molecules scored on that same target. However, that benchmark measures only single-target hit optimization and guides search with a learned surrogate rather than a live docking oracle. TARTARUS [Nigam et al., 2023] took a further step toward realistic drug discovery by evaluating generated molecules with a live docking oracle during optimization rather than guiding search through a learned surrogate. However, these benchmarks focus on optimizing individual tasks and do not provide a held-out task protocol for evaluating generalization of model and hyperparameter choices to unseen proteins or drug-design settings. In this work, we bridge this gap by combining docking-based objectives with PMO-style evaluation standards.

Our contributions are as follows:

1. We introduce PMO-Dock, a new benchmark that brings PMO-style clarity to docking-based objectives. It comprises 25 tasks split into 11 validation and 14 test tasks with a strict separation between model development and evaluation, and covers hit generation, lead optimization, and new binding specificity tasks.
2. We introduce a new task category for binding specificity, where molecules must bind strongly to a target protein or pocket while avoiding off-target interactions, a critical but

underexplored aspect in drug discovery benchmarks. We define five specificity tasks in this category, addressing both intra-protein and inter-protein selectivity challenges.

3. We analyse four well-performing molecular optimization algorithms on the benchmark and gain several important insights. Firstly, we find that there are no generalist methods that dominate others in all tasks, and methods are better suited for distinct drug design workflows. Additionally, we systematically study the hyperparameter transferability and sensitivity of different methods, which reflect how they should be used in practice.

2 The PMO-Dock Benchmark

This section introduces the setup of the PMO-Dock benchmark. The central aim of the benchmark is to measure the optimization power, sample efficiency, practicality, and generalization to unseen molecular optimization problems. To cover a wider range of workflows in drug design, we propose three task categories with distinct success measures in our benchmark, **hit generation**, **lead optimization** and **docking with specificity**, which are detailed in the subsequent sections.

2.1 Hit Generation

We inherit the *hit generation* task from [Guo and Schwaller, 2024c], in which a molecular generative model proposes candidate molecules for a given protein target and is evaluated by an oracle under a fixed evaluation budget. The optimization objective is to maximize the number of generated unique molecules that satisfy predefined physicochemical and target-specific constraints.

Targets. The benchmark includes five protein targets: PARP1, Factor VII (FA7), the 5-hydroxytryptamine receptor 1B (5HT1B), BRAF, and JAK2.

Optimization objective. Given an oracle budget of $B = 3000$ molecule evaluations, a generative policy π induces a set of evaluated molecules $\{x_i\}_{i=1}^B$. Let $\mathcal{H}(x) \in \{0, 1\}$ denote a binary indicator of whether molecule x satisfies the hit constraints. The objective is to maximize the expected **hit rate**,

$$\max_{\pi} \mathbb{E}_{\{x_i\}_{i=1}^B \sim \pi} \left[\frac{1}{B} \sum_{i=1}^B \mathcal{H}(x_i) \right],$$

We run each algorithm with ten random seeds and report the average metric.

Hit definition.

$$\mathcal{H}(x_i) = \begin{cases} 1, & \text{if } \text{QED}(x) \in [0.5, 1.0] \wedge \text{SA}(x) \in [1.0, 5.0] \wedge s^{(t)}(x) \geq s_{\min}^{(t)}, \\ 0, & \text{otherwise.} \end{cases}$$

where $s^{(t)}(x)$ denotes the docking value for target t , taken in absolute value. The target-specific lower bounds are given in Appendix I.2. Following prior work, we calculate $s^{(t)}$ using QuickVina [Alhossary et al., 2015] with low exhaustiveness (1) to save computational resources.

2.2 Lead Optimization

We consider a *lead optimization* task, borrowed from GenMol [Lee et al., 2025], in which a molecular generative model proposes new compounds that improve a protein-specific objective while remaining in the local chemical neighborhood of a given seed molecule for the target protein. The task is formulated as constrained, goal-directed molecular generation under a fixed oracle evaluation budget. The protein targets are the same as in the hit generation task.

Optimization objective. Given a seed molecule x_0 and a target t , the objective is to generate molecules that maximize predicted target activity as measured by a docking-based oracle $s^{(t)}(x)$, while satisfying locality and drug-likeness constraints. As in hit generation, $s^{(t)}(x)$ is a QuickVina [Alhossary et al., 2015] score computed at exhaustiveness 1. For a generative policy π and an

oracle budget of $B = 1000$ evaluations, the optimization problem is

$$\max_{\pi} \mathbb{E}_{\{x_i\}_{i=1}^B \sim \pi} \left[\max_{1 \leq i \leq B} s^{(t)}(x_i) \mathcal{H}(x_i) \right],$$

subject to all generated molecules x_i satisfying the constraints defined below.

$$\mathcal{H}(x_i) = \begin{cases} 1, & \text{if QED}(x) \in [0.5, 1.0] \wedge \text{SA}(x) \in [1.0, 5.0] \wedge s^{(t)}(x) \geq s_{\min}^{(t)} \wedge \text{sim}(x, x_0) \geq 0.6, \\ 0, & \text{otherwise.} \end{cases}$$

We evaluate the methods by looking at the top-1 docking score among the molecules satisfying all constraints. In case when none of the molecules generated by the algorithm satisfies these constraints, or the docking scores are too low, we assume the seed molecule is known to the algorithm, and report its docking score as the result of the algorithm. We run each algorithm with three random seeds and report the average metric.

2.3 Specificity Binding

We introduce *specificity binding* tasks to evaluate both interprotein and intra-protein pocket selectivity. These tasks aim to evaluate how well molecules can selectively engage a target protein pocket while avoiding unwanted interactions with homologous proteins or with similar pockets within the same protein, which is critical for improving the precision and safety of drug candidates. The goal is to generate molecules that preferentially bind to a target pocket while minimizing off-target interactions. To the best of our knowledge, our benchmark is the first one that incorporates such a selectivity task, which is a crucial problem in real-world drug discovery.

The specificity objective was defined as a difference in docking scores:

$$\max_{\pi} \mathbb{E}_{\{x_i\}_{i=1}^B \sim \pi} \left[\max_{1 \leq i \leq B} \mathcal{L}(x_i) \mathcal{H}(x_i) \right],$$

where

$$\mathcal{L} = s^{(P_{\text{target}})}(x) - s^{(P_{\text{off}})}(x)$$

and

$$\mathcal{H}(x_i) = \begin{cases} 1, & \text{if QED}(x) \in [0.4, 1.0] \wedge \text{SA}(x) \in [1.0, 4.0] \wedge s^{(6\text{NZP})}(x) \geq s_{\text{ref}}^{(\text{spec})} \\ 0, & \text{otherwise.} \end{cases}$$

Here P_{target} represents the desired target pocket, and P_{off} corresponds either to a homologous protein or to an alternative domain within the same protein, $s^{(t)}(x)$ denotes the docking value for target pocket t . This formulation promotes ligands with selective binding toward the target while penalizing off-target interactions. We run each algorithm with three random seeds and report the average metric.

Targets. To assess both inter-protein and intra-protein pocket specificity, we selected three closely related kinases from the JAK family: JAK1, TYK2, and JAK2, which share high sequence and structural similarity. Additionally, each of these proteins contains two domains: the active kinase domain (JH1, ATP-binding pocket) and the pseudokinase domain (JH2, allosteric pocket). These domains also exhibit some level of similarity but have subtle differences that may influence binding. The presence of these two domains across three closely related kinases makes these proteins particularly suitable for assessing specificity both between proteins (inter-protein) and within the same protein (intra-protein).

The following experimentally resolved structures were used in this study: JAK1 JH1 (PDB ID: 5KHW), JAK1 JH2 (PDB ID: 4L00), TYK2 JH1 (PDB ID: 7UYT), TYK2 JH2 (PDB ID: 6NZP), JAK2 JH1 (PDB ID: 7UYW), and JAK2 JH2 (PDB ID: 5UT5).

Based on this setup, three types of pairwise specificity tasks were defined, each aiming to maximize binding to TYK2 JH2 while minimizing off-target interactions. In all cases, TYK2 JH2 was used as the target pocket, and specificity was evaluated against either another domain of TYK2 or homologous domains from JAK1 and JAK2.

	Hit 5ht1b	Hit fa7	Hit parp1	Lead 5ht1b seed0	Lead 5ht1b seed1	Lead fa7 seed0	Lead fa7 seed1	Lead parp1 seed0	Lead parp1 seed1	Spec 6nnp 5ut5	Spec 6nnp 7ayt	Hit Avg	Lead Avg	Spec Avg
GenMol	0.26 (0.21,0.29)	0.10 (0.07,0.13)	0.08 (0.05,0.12)	12.10 (11.00,12.30)	9.17 (7.60,12.30)	7.47 (6.80,8.30)	7.53 (7.20,7.90)	11.57 (11.40,11.80)	11.03 (10.80,11.30)	2.59 (2.23,3.03)	2.13 (1.91,2.29)	0.15 (0.12,0.17)	9.81 (9.57,10.30)	2.36 (2.20,2.67)
Genetic GFN	0.39 (0.34,0.44)	0.02 (0.01,0.02)	0.11 (0.09,0.13)	4.50 (4.50,4.50)	11.50 (11.00,12.40)	7.97 (7.30,8.50)	6.77 (6.70,6.90)	11.33 (11.20,11.40)	10.77 (10.40,11.40)	2.34 (1.72,2.66)	1.82 (1.63,2.15)	0.17 (0.15,0.19)	8.81 (8.55,9.12)	2.08 (1.94,2.17)
Saturn	0.90 (0.89,0.91)	0.29 (0.20,0.38)	0.72 (0.69,0.75)	4.50 (4.50,4.50)	11.10 (10.30,12.90)	8.50 (8.50,8.50)	6.70 (6.70,6.70)	12.27 (12.00,12.40)	10.10 (9.90,11.50)	5.82 (4.01,6.24)	2.41 (2.09,2.68)	0.63 (0.60,0.66)	8.86 (8.57,9.05)	4.11 (3.94,5.10)
ChemLlama	0.44 (0.43,0.45)	0.34 (0.28,0.38)	0.47 (0.41,0.54)	12.83 (12.70,12.90)	11.60 (11.20,12.90)	8.77 (8.70,8.90)	7.33 (7.00,7.70)	9.37 (8.90,9.60)	12.27 (12.50,12.80)	4.92 (4.54,5.24)	3.75 (3.30,4.00)	0.42 (0.39,0.45)	10.36 (10.27,10.48)	4.33 (3.92,4.62)

Figure 2: Validation performance under category-specific hyperparameter selection. For each task, hyperparameters are chosen to maximize average performance on the other validation tasks in the same category (hit generation, lead optimization, or specificity). The last three columns are means over the tasks in each category (Hit Avg, Lead Avg, Spec Avg). Colors are normalized separately by category, so intensities should only be compared within category. More intense colors indicate higher performance relative to other entries in the same category. The alternative all-but-me rule is in Appendix D.

1. **Intra-protein domain specificity:** molecules are optimized for strong binding to TYK2 JH2 while reducing binding to TYK2 JH1.
2. **Inter-protein pseudokinase specificity:** molecules are optimized for strong binding to TYK2 JH2 while reducing binding to the pseudokinase domains of related proteins, JAK1 JH2 and JAK2 JH2.
3. **Cross-domain inter-protein specificity:** molecules are optimized for strong binding to TYK2 JH2 while minimizing binding to the kinase domains of related proteins, JAK1 JH1 and JAK2 JH1.

Docking scores are QuickVina [Alhossary et al., 2015] scores from three independent runs at exhaustiveness 8. The docking procedure is detailed in Appendix G.

Constraints. To define task satisfaction criteria, we used Deucravacitinib (BMS-986165) - a first-in-class, highly selective TYK2 pseudokinase inhibitor approved for the treatment of moderate-to-severe plaque psoriasis - as a reference compound for TYK2 JH2 binding. The docking score of Deucravacitinib against TYK2 JH2 was used as a baseline reference. For all specificity tasks, a generated molecule was considered successful only if its docking score to TYK2 JH2 exceeded that of the reference compound, thereby ensuring strong target engagement.

2.4 Validation and Test Split

To measure generalization to unseen targets and molecules (in the case of lead optimization), we split the tasks in each task category into two sets, the **validation tasks** and **testing tasks**, where validation tasks would serve for the algorithm development and hyperparameter selection, while testing tasks would be for final evaluation. Figure 1 shows the complete task composition across all the three categories, totaling into 11 validation tasks and 14 test tasks.

2.5 Model Development and Hyperparameter Tuning

Algorithm developers may run as many hyperparameter sweeps, and explore as many model design choices, as they need on the validation tasks. Test tasks should be used only for the final evaluation, after model development and hyperparameter selection are complete. Importantly, nothing should be assumed about the specific proteins, lead molecules, or binding pockets in the test set when developing and tuning new algorithms. For instance, if hyperparameters are selected based on protein similarity to validation targets, similarity metrics must be defined and tuned exclusively on validation tasks, to ensure no information leakage. The benchmark does not require a particular training dataset. Any dataset is allowed, provided it is chosen without using the test proteins, lead molecules, or binding pockets.

3 Experiments

In this section, we select a diverse set of recently released molecular optimization algorithms that have shown significant performance and conduct hyperparameter tuning, evaluation and analysis on the PMO-Dock benchmark.

3.1 Optimization Methods

We evaluate four diverse methods spanning different optimization paradigms: Saturn [Guo and Schwaller, 2024c] (reinforcement learning with language models), GenMol [Lee et al., 2025] (discrete diffusion), Genetic-guided GFlowNets [Kim et al., 2024] (GFlowNet with genetic operators), and ChemLlama (LLM-based evolutionary optimization). Saturn fine-tunes an autoregressive SMILES language model with reinforcement learning and replays high-reward molecules from an augmented memory buffer. GenMol is a discrete diffusion model over molecular fragments and optimizes by remasking and resampling fragments. Genetic-guided GFlowNet fine-tunes a pretrained SMILES policy with the trajectory-balance objective while using genetic crossover and mutation to propose candidates. ChemLlama is a modification of Chemlactica [Guevorguian et al., 2024]. Chemlactica is a molecular language model trained on PubChem molecules with the Galactica architecture and an evolutionary algorithm on top of that model. ChemLlama keeps this design and replaces the Galactica backbone with Llama. It keeps a pool of high-scoring molecules and uses the language model to propose analogs. Detailed descriptions of each method and their hyperparameter configurations are in Appendix B.

3.2 Hyperparameter Tuning

Hyperparameters are chosen on the validation tasks only and then frozen for the test tasks. We use **category-specific selection**. Each method is swept independently on every validation task. The search spaces differ by method and are listed in Appendix B. Saturn varies buffer size, reward type, and whether genetic operators are used. GenMol varies the fragment-masking rate, the randomness parameter, and the softmax temperature. Genetic-guided GFlowNet varies the rank-based replay coefficient and the KL regularization coefficient. ChemLlama varies pool size, the number of prompts, and the number of similar molecules shown in each prompt.

Hit rates, lead scores, and specificity scores are not on a common scale, so a configuration is never chosen by averaging across categories. For a validation task, each configuration is scored by its mean performance on the other validation tasks in the same category, and the configuration with the highest mean is selected. That configuration is then used for every test task in the category. Test scores are not used in the selection.

An alternative that averages over all other validation tasks, regardless of category, is reported only as an ablation in Appendix D. Selecting the best configuration on each validation task itself, with no cross-task transfer, is a separate extended-budget ceiling, reported in Section 3.5.

3.3 Validation Tasks

Figure 2 reports validation performance under that category-specific rule, one row per method. Saturn is strongest on hit generation, while ChemLlama is strongest on specificity. Lead scores are closer across methods. The same per-task layout under all-but-me selection is in Appendix D.

3.4 Test Tasks

Figure 3 presents the final evaluation of all the methods under the category-specific rule in Section 3.3. Cell annotations report mean scores with 95% bootstrap confidence intervals over evaluation seeds (Appendix E.1). We immediately observe that no method is superior in all tasks. Saturn is considerably better on the hit generation, while ChemLlama exhibits stronger performance on selectivity binding. All methods perform comparable on lead optimization.

Our findings indicate that there is no single generalist method that performs consistently better across all the tasks, and that when it comes to real-world drug discovery settings, practitioners ought to select optimization algorithms that are more suitable for specific workflows.

	Hit braf	Hit jak2	Lead sh1b seed0	Lead braf seed0	Lead braf seed1	Lead braf seed2	Lead fa7 seed0	Lead jak2 seed0	Lead jak2 seed1	Lead jak2 seed2	Lead parp1 seed0	Spec 6npz 4100	Spec 6npz 5khw	Spec 6npz 7ayw	Hit Avg	Lead Avg	Spec Avg
GenMol	0.20	0.29	12.67	10.73	11.30	11.43	8.57	11.20	10.93	10.77	11.10	3.19	2.60	2.26	0.24	10.97	2.68
	(0.16,0.21)	(0.27,0.30)	(12.50,12.80)	(10.40,11.10)	(11.10,11.50)	(11.30,11.60)	(8.50,8.70)	(11.20,11.20)	(10.70,11.30)	(10.50,10.90)	(10.20,12.90)	(2.91,3.41)	(2.35,2.81)	(1.78,2.81)	(0.20,0.26)	(10.50,11.21)	(2.51,2.87)
Genetic GFN	0.01	0.02	10.30	9.53	10.03	9.80	8.50	9.80	9.47	8.63	11.43	2.39	1.91	2.01	0.01	9.72	2.10
	(0.01,0.01)	(0.02,0.03)	(10.00,10.60)	(9.30,10.00)	(9.40,10.90)	(9.80,9.80)	(8.50,8.50)	(9.60,10.10)	(9.00,9.70)	(8.60,8.70)	(11.20,11.60)	(2.29,2.47)	(1.80,1.97)	(1.95,2.04)	(0.01,0.02)	(9.50,9.93)	(2.09,2.11)
Saturn	0.36	0.58	12.13	9.07	10.13	10.40	8.50	9.53	9.63	10.37	11.50	3.73	2.52	1.42	0.47	10.14	2.56
	(0.32,0.39)	(0.53,0.64)	(11.70,12.40)	(8.90,9.30)	(9.90,10.30)	(10.10,10.60)	(8.50,8.50)	(9.10,9.90)	(9.50,9.80)	(10.00,10.70)	(11.30,11.60)	(3.60,3.85)	(2.43,2.58)	(1.32,1.42)	(0.43,0.51)	(10.04,10.24)	(2.48,2.59)
ChemLlama	0.36	0.44	12.37	10.43	11.97	11.27	8.57	11.07	10.77	11.03	12.43	4.94	4.45	3.74	0.40	11.10	4.38
	(0.30,0.41)	(0.38,0.50)	(11.60,13.10)	(9.90,10.90)	(11.20,12.60)	(11.10,11.40)	(8.50,8.70)	(10.20,11.70)	(10.60,11.70)	(10.70,11.30)	(11.20,13.60)	(4.62,5.13)	(3.93,5.30)	(3.60,3.80)	(0.37,0.43)	(10.98,11.19)	(4.08,4.71)

Figure 3: Test task leaderboard. Rows: methods, columns: test tasks by category. Hyperparameters selected per Section 3.3. Best per task highlighted. Colors are normalized separately by category, so intensities should only be compared within category. More intense colors indicate higher performance relative to other entries in the same category.

	Hit sh1b	Hit braf	Hit parp1	Lead sh1b seed0	Lead sh1b seed1	Lead fa7 seed0	Lead fa7 seed1	Lead parp1 seed0	Lead parp1 seed1	Lead parp1 seed2	Spec 6npz 5u15	Spec 6npz 7ayw	Hit Avg	Lead Avg	Spec Avg
GenMol	0.29 (0.27,0.32)	0.14 (0.10,0.18)	0.14 (0.10,0.18)	12.63 (12.00,13.00)	12.97 (12.90,13.10)	8.43 (8.30,8.50)	8.20 (7.90,8.40)	12.07 (11.80,12.40)	11.57 (10.90,12.70)	4.00 (2.33,5.03)	2.80 (2.47,3.00)	0.19 (0.16,0.22)	10.98 (10.82,11.11)	3.40 (2.68,3.99)	
Genetic GFN	0.41 (0.39,0.43)	0.62 (0.62,0.63)	0.17 (0.12,0.24)	7.27 (4.50,12.90)	12.60 (12.20,13.00)	8.23 (8.10,8.40)	7.43 (6.70,8.30)	11.80 (11.30,12.40)	11.37 (10.70,12.50)	2.80 (1.93,3.35)	2.19 (2.17,2.21)	0.20 (0.18,0.22)	9.78 (9.20,10.60)	2.50 (2.07,2.71)	
Saturn	0.90 (0.89,0.91)	0.34 (0.29,0.38)	0.72 (0.69,0.75)	4.50 (4.50,4.50)	11.10 (10.30,11.90)	8.50 (8.50,8.50)	6.70 (6.70,6.70)	12.30 (12.20,12.40)	10.33 (9.50,10.80)	5.82 (4.01,8.24)	2.41 (2.09,2.68)	0.65 (0.63,0.67)	8.91 (8.63,9.00)	4.11 (3.94,5.35)	
ChemLlama	0.54 (0.52,0.55)	0.40 (0.32,0.47)	0.52 (0.47,0.56)	13.30 (12.90,13.60)	11.60 (11.20,11.90)	9.13 (8.80,9.70)	7.77 (7.20,8.10)	11.23 (11.00,11.40)	13.30 (12.40,15.00)	7.69 (4.60,12.40)	4.29 (3.40,5.11)	0.48 (0.45,0.52)	11.06 (10.70,11.37)	5.99 (4.00,6.70)	

Figure 4: Validation performance under per-task best hyperparameters (one row per method). For each task column, the reported value uses the hyperparameter setting that maximizes the validation metric on that task, corresponding to an unrestricted oracle-call scenario for model selection on the validation split. Layout and per-category coloring match Figure 2.

3.5 Evaluation with Extended Budget

In previous sections, we evaluated and analysed the optimization power and generalization of the different methods, under a strict evaluation budget, which emulates a realistic drug design environment. However, it is not uncommon that in practice one is interested in finding a good solution and is less worried about the strict oracle budget. To accommodate these needs, we additionally evaluate each algorithm under an *extended oracle-call* regime: for every task, we select hyperparameters that maximize performance on that same task. This isolates per-task ceiling behavior when model selection is not constrained by cross-task generalization. The results are reported in Figure 4.

3.6 Statistical Significance

The intervals in Figure 3 describe seed-level uncertainty for one method on one task. They do not test whether one method ranks above another across a category. Some comparisons are already separated by those intervals. On hit generation, Saturn and ChemLlama are separated from GenMol and Genetic GFN. On JAK2, Saturn is also separated from ChemLlama, while on BRAF their intervals overlap. On specificity, ChemLlama is separated from the other three methods. On lead optimization, three evaluation seeds and close scores leave most pairwise intervals overlapping. A category-level claim needs a comparison that resamples tasks and seeds together.

Figure 5 reports that comparison. For each evaluation category (hit, lead, specificity) we bootstrap over pools of (task, seed) pairs, average each method’s score on each draw, and report how often one method’s category-level mean strictly exceeds another’s across $B = 10,000$ replicates. This quantifies head-to-head outcomes under task-and-seed resampling within a category. How those confidence intervals are computed, the per-task bootstrap win matrices, and the full resampling procedures are in Appendix E.

4 Hyperparameter Sensitivity and Transferability

A central question in the evaluation of molecular optimization methods is whether hyperparameters tuned on one set of tasks generalize to new tasks. Robust algorithms should exhibit hyperparameters that work reasonably well on different protein targets and task types. Beyond leaderboard rankings,

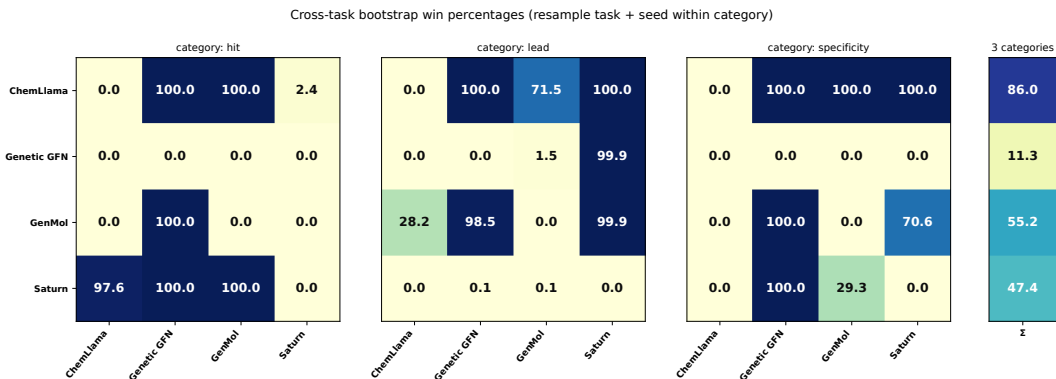


Figure 5: **Category-level cross-task bootstrap win percentages.** *Left three panels:* 4×4 matrices for hit ($K=20$), lead ($K=27$), and specificity ($K=9$) (task-seed pool sizes). Rows and columns are methods. Cell (i, j) is the percentage of $B = 10,000$ bootstrap replicates in which method i ’s resampled mean strictly exceeds method j ’s after jointly resampling task blocks and seeds within the category. Colour encodes percentage on $[0, 100]$. *Rightmost panel (Σ):* per method summing row totals across the three category matrices.

we study how each method responds to hyperparameter choice and what that implies for deployment on new targets. The PMO-Dock protocol permits extensive hyperparameter sweeps on validation tasks but restricts oracle use on held-out test tasks, since molecular optimization is oracle-limited and such repeated tuning is rarely affordable at scale. We therefore characterize method behavior on the validation split, where tuning is allowed, and ask whether hyperparameters selected there can be reused on related tasks without a new sweep, as required on test. This motivates two complementary analyses. *Hyperparameter transferability* measures whether the best parameter on one validation task or category remain effective on another. We study transferability along two axes: **within-category transfer** and **cross-category transfer**. *Hyperparameter sensitivity* measures how peaked each method’s hyperparameter landscape is and how much performance varies when moving away from the task-optimal configuration.

4.1 Within-category transfer.

We study whether hyperparameters tuned on one validation target remain effective on another target of the same task type. Strong within-category transfer supports reusing a single hyperparameter configuration across category without a new hyperparameter sweep.

We quantify within-category transfer using cross-protein scatter plots on validation tasks (Appendix C) and the same-category blocks of the hyperparameter transfer matrices (Figures 9 and 10). In the transfer matrix, each cell records the drop in target-task performance when using the source task’s best hyperparameters instead of the target’s own optimum. In the scatter plots, each point is one swept hyperparameter configuration and the axes compare performance on two targets of the same type. Hit-generation pairs are in Appendix F.1. Lead optimization is shown in Figure 6.

Patterns are strongly method-dependent. For hit generation, Saturn shows clear cross-protein alignment, GenMol shows essentially none, and ChemLlama and Genetic GFN are mixed. For lead optimization, transfer is fragile for all methods: hyperparameters selected on one protein rarely predict performance on another (Figure 6). In the transfer matrices, Genetic GFN is the least affected by which source task supplied the hyperparameters, while Saturn, ChemLlama, and GenMol degrade more when moving away from the target-specific optimum. Per-pair correlations and full matrix heatmaps are reported in Appendix C and F.1.

4.2 Cross-category transfer.

Cross-category transfer asks whether hyperparameters tuned with one task category (hit, lead, or specificity) predict performance on another. This tests whether a practitioner can tune on one objective type and deploy on another without category-specific re-sweeping. We read cross-category transfer from hit-versus-lead scatter plots on validation tasks (Figures 7 and 8) and off-category blocks of the

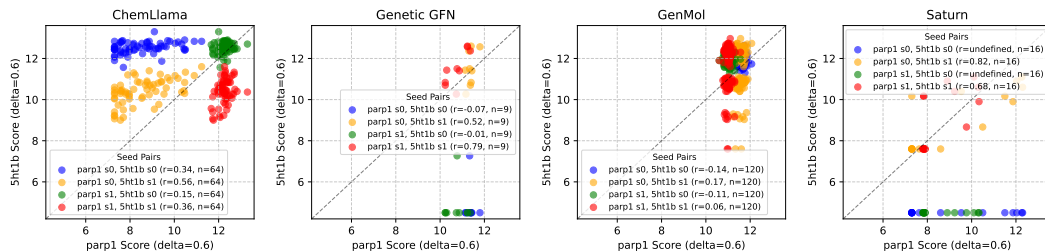


Figure 6: Cross-target lead optimization performance (parp1 vs 5ht1b, $\delta = 0.6$). One panel per method, colors represent different seed-molecule pairs. Points are aggregated across random seeds, and legends report Pearson correlations and sample sizes. Cross-target lead transfer is generally weak for all four methods.

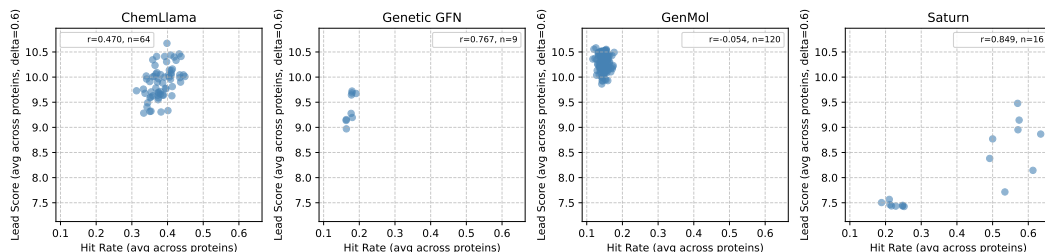


Figure 7: Hit vs lead (averaged across proteins, lead at $\delta=0.6$). One panel per method, legends report Pearson correlations and sample sizes. Saturn shows the strongest alignment, GenMol is near zero. Per-protein breakdown in Figure 8.

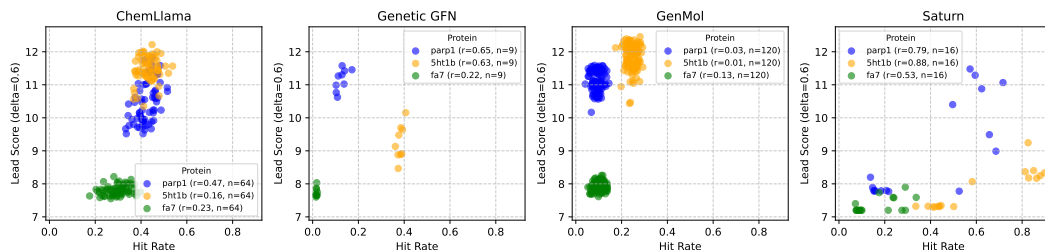


Figure 8: Hit rate versus lead optimization score ($\delta = 0.6$) by protein. Each protein is shown in a different color within each method panel. Scores are aggregated across random seeds and seed molecules (for lead tasks), legends report Pearson correlations and sample sizes. Saturn shows positive alignment on all three targets, other methods vary.

transfer matrices (Appendix C). Each scatter point is one hyperparameter configuration. The axes compare hit rate to lead score at $\delta = 0.6$, averaged across proteins or broken down by protein. Tighter positive alignment indicates that configurations strong on one category tend to be strong on the other.

Saturn shows the strongest and most consistent hit-to-lead alignment and remains positively aligned on every protein. Genetic GFN is also positively aligned but on a much smaller hyperparameter grid. ChemLlama is moderate and GenMol is essentially uncorrelated. The transfer matrices confirm that cross-category degradation exceeds within-category degradation for Saturn, ChemLlama, and GenMol, while Genetic GFN shows little category-boundary effect. We additionally examined specificity-pair transfer and held-out target generalization in Appendices F.3 and F.4.

4.3 Hyperparameter Sensitivity

Hyperparameter sensitivity asks how peaked each method’s hyperparameter performance landscape is. If the task-optimal configuration cannot be used, how much performance is lost across alternative settings? High sensitivity means small changes in hyperparameters or selection strategy cause large

performance swings. We assess sensitivity in two ways. First, we read scatter plots as landscape shape: dispersed clouds and weak correlations on same-type comparisons, such as lead performance at $\delta = 0.4$ versus $\delta = 0.6$ (Appendix F.2, Figure 18), indicate sensitivity to constraint tightness as well as to hyperparameter choice. Second, we compare validation performance under transfer-based hyperparameter selection (Figure 2) to validation performance under per-task best hyperparameters (Figure 4). The latter reflects an oracle-rich ceiling where each task is tuned independently on validation. The gap between the two quantifies the cost of not re-tuning when deploying under budget constraints. Comparing category-specific selection (Figure 2) with all-but-me selection (Figure 11), Saturn gains the most from staying within category. ChemLlama and GenMol show only a small gain, and Genetic GFN is essentially indifferent, consistent with flatter cross-task effects in the transfer matrices. Methods with strong transfer and a small gap to the per-task-best ceiling, notably Saturn on hit tasks, can reuse validation-tuned hyperparameters on similar targets. Methods with weak cross-task alignment and a larger ceiling gap, notably GenMol, cannot justify reuse across task types and require tuning on a sufficiently similar validation task before test deployment. Genetic GFN is comparatively robust to where hyperparameters were selected, though its smaller sweep grid limits how finely we can resolve landscape shape.

5 Do these Methods Hack Rewards?

5.1 The role of docking exhaustiveness

Repeated docking of the same protein–ligand pair with QuickVina can yield different scores, which may produce false negatives when only a single run is used. This is especially relevant for reward-hacking risk in specificity tasks. To choose a practical protocol for that stage, we ran a small-scale comparison of replication versus exhaustiveness (multiple independent runs versus raising exhaustiveness). Three independent runs at exhaustiveness 8 were more stable than one run at exhaustiveness 16, and agreed closely with a much costlier $3 \times \text{ex}16$ scheme, we therefore adopt $3 \times \text{ex}8$ for specificity docking. Appendix G.1 provides the study details and Figure 21.

5.2 Manual Inspection

To qualitatively assess the generated compounds, we manually inspected molecules produced by the generative models. Across all evaluated methods, the generated structures appeared chemically valid and well-formed. We did not observe unusual artifacts, unrealistic substructures, or chemically implausible patterns in the sampled molecules. Consistent with this observation, the Figure 22 presents a representative example from the specificity optimization task: a chemically valid molecule generated by ChemLlama. This molecule exhibits the desired binding profile, with favorable binding to the target and weak binding to the anti-target. Further checks of the docking oracle, including exhaustiveness 1 versus 8 on hit and lead tasks, the agreement of QuickVina with AutoDock Vina, and a PoseBusters assessment of high-scoring poses, are reported in Appendix H.

6 Conclusion

In this paper, we developed a novel benchmark called PMO-Dock, which emulates a real world drug discovery environment and evaluates the ability of molecular optimization algorithms to generalize to previously unseen tasks, under a strict sample efficiency setting. The benchmark consists of 25 problems, composed of hit generation, lead optimization and our newly proposed specificity binding tasks, where 11 of the tasks are dedicated to method advancements and hyperparameter tuning, and 14 tasks are held out for final evaluation. Additionally, we thoroughly evaluate four well-performing optimization algorithms on our benchmark and reveal important insights about their distinct strengths and weaknesses, which ultimately guide their applicability and appropriateness in different drug design workflows. We believe that the PMO-Dock benchmark provides a concrete protocol to support and measure algorithmic advancements in the field.

Acknowledgements

This research was supported by the HESC MESCS RA grant No. 24FP-1A058.

References

- Wenhao Gao, Tianfan Fu, Jimeng Sun, and Connor Coley. Sample efficiency matters: a benchmark for practical molecular optimization. *Advances in Neural Information Processing Systems*, 35: 21342–21357, 2022.
- Jeff Guo and Philippe Schwaller. Augmented memory: Sample-efficient generative molecular design with reinforcement learning. *JACS Au*, 4(4):1401–1413, 2024a. doi: 10.1021/jacsau.4c00066. URL <https://doi.org/10.1021/jacsau.4c00066>.
- Hyeonah Kim, Minsu Kim, Sanghyeok Choi, and Jinkyoo Park. Genetic-guided gflownets for sample efficient molecular optimization, 2024. URL <https://arxiv.org/abs/2402.05961>.
- Philipp Guevorguian, Menua Bedrosian, Tigran Fahradyan, Gayane Chilingaryan, Hrant Khachatryan, and Armen Aghajanyan. Small molecule optimization with large language models, 2024. URL <https://arxiv.org/abs/2407.18897>.
- Haorui Wang, Marta Skreta, Cher-Tian Ser, Wenhao Gao, Lingkai Kong, Felix Strieth-Kalthoff, Chenru Duan, Yuchen Zhuang, Yue Yu, Yanqiao Zhu, Yuanqi Du, Alan Aspuru-Guzik, Kirill Neklyudov, and Chao Zhang. Efficient evolutionary search over chemical space with large language models, 2025. URL <https://arxiv.org/abs/2406.16976>.
- Jeff Guo and Philippe Schwaller. Beam enumeration: Probabilistic explainability for sample efficient self-conditioned molecular design, 2024b. URL <https://arxiv.org/abs/2309.13957>.
- Jeff Guo and Philippe Schwaller. Saturn: Sample-efficient generative molecular design using memory manipulation, 2024c. URL <https://arxiv.org/abs/2405.17066>.
- Seul Lee, Karsten Kreis, Srimukh Prasad Veccham, Meng Liu, Danny Reidenbach, Yuxing Peng, Saeed Paliwal, Weili Nie, and Arash Vahdat. Genmol: A drug discovery generalist with discrete diffusion, 2025. URL <https://arxiv.org/abs/2501.06158>.
- Tobiasz Ciepliński, Tomasz Danel, Sabina Podlowska, and Stanisław Jastrzębski. Generative models should at least be able to design molecules that dock well: A new benchmark. *Journal of Chemical Information and Modeling*, 63(11):3238–3247, 2023. doi: 10.1021/acs.jcim.2c01355. URL <https://doi.org/10.1021/acs.jcim.2c01355>.
- AkshatKumar Nigam, Robert Pollice, Gary Tom, Kjell Jorner, John Willes, Luca Thiede, Anshul Kundaje, and Alán Aspuru-Guzik. Tartarus: A benchmarking platform for realistic and practical inverse molecular design. In *Advances in Neural Information Processing Systems*, volume 36, pages 3263–3306, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/09f8b2469a3d1089a7c60d9ef1983271-Abstract.html.
- Amr Alhossary, Stephanus Daniel Handoko, Yuguang Mu, and Chee-Keong Kwoh. Fast, accurate, and reliable molecular docking with quickvina 2. *Bioinformatics*, 31(13):2214–2216, 2015. doi: 10.1093/bioinformatics/btv082. URL <https://doi.org/10.1093/bioinformatics/btv082>.
- Guy W Bemis and Mark A Murcko. The properties of known drugs. 1. molecular frameworks. *Journal of medicinal chemistry*, 39(15):2887–2893, 1996. doi: 10.1021/jm9602928. URL <https://doi.org/10.1021/jm9602928>.
- Emmanuel Noutahi, Cristian Gabellini, Michael Craig, Jonathan S. C. Lim, and Prudencio Tossou. Gotta be safe: A new framework for molecular design, 2023. URL <https://arxiv.org/abs/2310.10773>.
- Jens Degen, Corina Wegscheid-Gerlach, Andrea Zaliani, and Matthias Rarey. On the art of compiling and using ‘drug-like’ chemical fragment spaces. *ChemMedChem: Chemistry Enabling Drug Discovery*, 3(10):1503–1507, 2008. doi: 10.1002/cmdc.200800178. URL <https://doi.org/10.1002/cmdc.200800178>.
- Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. *Advances in Neural Information Processing Systems*, 34:27381–27394, 2021.

- Jan H Jensen. A graph-based genetic algorithm and generative model/monte carlo tree search for the exploration of chemical space. *Chemical science*, 10(12):3567–3572, 2019. doi: 10.1039/C8SC05372C. URL <https://doi.org/10.1039/C8SC05372C>.
- Garrett M Morris, Ruth Huey, William Lindstrom, Michel F Sanner, Richard K Belew, David S Goodsell, and Arthur J Olson. Autodock4 and autodocktools4: Automated docking with selective receptor flexibility. *Journal of computational chemistry*, 30(16):2785–2791, 2009. doi: 10.1002/jcc.21256. URL <https://doi.org/10.1002/jcc.21256>.
- Oleg Trott and Arthur J Olson. Autodock vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *Journal of Computational Chemistry*, 31(2):455–461, 2010. doi: 10.1002/jcc.21334. URL <https://doi.org/10.1002/jcc.21334>.
- Jerome Eberhardt, Diogo Santos-Martins, Andreas F Tillack, and Stefano Forli. AutoDock Vina 1.2.0: New docking methods, expanded force field, and Python bindings. *Journal of Chemical Information and Modeling*, 61(8):3891–3898, 2021. doi: 10.1021/acs.jcim.1c00203. URL <https://doi.org/10.1021/acs.jcim.1c00203>.
- Martin Buttenschoen, Garrett M Morris, and Charlotte M Deane. Posebusters: AI-based docking methods fail to generate physically valid poses or generalise to novel sequences. *Chemical Science*, 15(9):3130–3139, 2024. doi: 10.1039/D3SC04185A. URL <https://doi.org/10.1039/d3sc04185a>.

A Limitations

While we believe that our work standardizes the comparison of the docking-based molecular optimization tasks, we recognize two main limitations. First, we used QuickVina throughout the paper for computing binding affinity/docking score, however, we believe that more advanced simulations may be used instead to mitigate any potential hacking of the QuickVina simulation and make the claims and significance of the results more relevant in real-world drug discovery. Secondly, our benchmark mainly focuses on docking-based objectives and problems, however, drug development involves additional important requirements, such as solubility and non-toxicity, which should be incorporated in the future advancements of the benchmark.

B Detailed Method Descriptions and Hyperparameter Configurations

This section provides detailed descriptions of each method evaluated in the benchmark, followed by complete hyperparameter search space specifications.

B.1 Saturn

Saturn [Guo and Schwaller, 2024c] is a language-model-based molecular generator trained with reinforcement learning using an autoregressive policy that generates SMILES representations of molecules. It employs an *Augmented Memory* [Guo and Schwaller, 2024a] replay buffer that stores the top- K high-reward molecules encountered during search (e.g., $K = 100$ in the original work). Buffer entries are repeatedly replayed by training on randomized SMILES augmentations to improve sample efficiency. A scaffold-based diversity filter (e.g., Bemis–Murcko [Bemis and Murcko, 1996] scaffold tracking) penalizes repeated scaffolds to mitigate mode collapse.

The authors proposed an additional feature, graph-based genetic operators (crossover and mutation) applied to the buffer population to produce new candidate molecules that can refresh the buffer. We kept this genetic algorithm component as a binary hyperparameter. The other hyperparameters we tested are buffer size $\sigma \in \{64, 128, 256, 512\}$ and two reward types: the one borrowed from GEAM algorithm and a hit-based reward. This resulted in 16 hyperparameter combinations.

Reward Functions. Saturn uses multiplicative reward functions. For **hit optimization** tasks, the reward is:

$$R(x) = \frac{s^{(\text{target})}(x)}{20} \cdot \text{QED}(x) \cdot \frac{10 - \text{SA}(x)}{9},$$

where all components are normalized to $[0, 1]$.

For **lead optimization** tasks, the reward is:

$$R(x) = \text{Sim}(x, x_0) \cdot \frac{s^{(\text{target})}(x)}{20} \cdot \text{QED}(x) \cdot \frac{10 - \text{SA}(x)}{9},$$

where the first term encourages high similarity to the seed molecule x_0 .

For **specificity optimization** tasks, Saturn’s reward function explicitly penalizes anti-target binding:

$$R(x) = \left(1 - \frac{s^{(\text{off})}(x)}{20}\right) \cdot \frac{s^{(\text{target})}(x)}{20} \cdot \text{QED}(x) \cdot \frac{10 - \text{SA}(x)}{9},$$

where the first term $(1 - s^{(\text{off})}(x)/20)$ encourages low anti-target binding.

B.2 GenMol

GenMol [Lee et al., 2025] is a masked discrete-diffusion model operating on a fragment-level representation SAFE [Noutahi et al., 2023], where molecules are decomposed into fragment tokens (e.g., BRICS [Degen et al., 2008] fragments). The model uses a bidirectional (BERT-like) denoising network and performs non-autoregressive iterative unmasking: starting from an all-masked fragment string, it repeatedly predicts token logits, locks in confident tokens, and repeats until decoding completes. For optimization, GenMol applies *fragment remasking*: entire fragments in a molecule are masked and resampled by the diffusion model, enabling structured, chemically meaningful mutations

suitable for lead optimization. The architecture supports de novo generation, fragment-constrained sampling, hit finding, and lead optimization within the same framework.

We performed an extensive hyperparameter sweep of 120 combinations for GenMol. We tested masking parameter $\gamma \in \{0, 0.1, 0.2, 0.3, 0.4, 0.5\}$, randomness $r \in \{1, 1.2, 2, 3\}$, and softmax temperature $\tau \in \{0.5, 0.8, 1, 1.2, 1.5\}$.

Reward Functions. GenMol uses a binary reward structure. For **hit** and **lead optimization** tasks, if molecule x satisfies all hit constraints (as defined in Section 2), the reward is the docking score:

$$R(x) = \begin{cases} s^{(\text{target})}(x) & \text{if } h(x) = 1 \\ 0 & \text{otherwise} \end{cases},$$

For **specificity optimization** tasks, the reward is:

$$R(x) = \begin{cases} (s^{(\text{target})}(x) - s^{(\text{off})}(x)) & \text{if } h(x) = 1 \\ 0 & \text{otherwise} \end{cases},$$

where $h(x)$ is the validity indicator function (equal to 1 if all constraints are satisfied, 0 otherwise).

B.3 Genetic-guided GFlowNets

This method [Kim et al., 2024] pretrains a SMILES generative policy (e.g., an RNN) on large chemical corpora to serve as a prior. During optimization, it fine-tunes the policy with the GFlowNet [Bengio et al., 2021] trajectory-balance objective while integrating genetic operators: parent selection, Graph-GA [Jensen, 2019] crossover and mutation to produce offspring, and replay-buffered off-policy TB updates. Training uses rank-weighted sampling from the buffer and a combined loss consisting of the TB-loss plus a KL penalty to the pretrained prior: $\mathcal{L} = \text{TB-loss} + \lambda \text{KL}(\pi_{\text{pre}} \parallel \pi_{\text{new}})$. This hybrid leverages GA to produce high-reward candidates and GFlowNet TB training to amortize sampling from promising regions while maintaining diversity.

Reward Functions. While Genetic-GFN is primarily introduced for scalar-valued objectives, the original paper applies it to multi-objective molecular optimization by defining a scalar score as a linear combination of multiple objectives with specified coefficients, we follow the same approach. We use scalar scores $s(x)$ aligned with the PMO-Dock task objectives (Section 2) and set $R(x) = \exp(-\beta s(x))$ as in the GFlowNet literature. β is an inverse temperature parameter.

The range for β is $\{10, 25, 50, 70, 100\}$. Motivated by the observation in the original Genetic-GFN paper that reward scaling in multi-objective settings affects the effective inverse temperature, we performed β selection using one representative task per task category. Within each category, the number of objectives is fixed, so the selected β was transferred to all tasks in that category. We found that $\beta = 50$ performed best in all categories and fixed it for all tasks.

In addition to β , we tuned the rank-based replay coefficient and the KL regularization coefficient on validation tasks. The rank coefficient was selected from $\{0.01, 0.03, 0.05\}$ and the KL coefficient from $\{0.0005, 0.001, 0.005\}$, following the official Genetic-GFN hyperparameter search space. All remaining components and hyperparameters were used without modification.

B.4 ChemLlama

ChemLlama is a modification of Chemlactica [Guevorguian et al., 2024]. Chemlactica is a molecular language model trained on PubChem molecules with the Galactica architecture and an evolutionary algorithm on top of that model. ChemLlama keeps this design, replaces the Galactica backbone with Llama, and uses the language model as a neural mutation and proposal operator inside a population-based optimization loop. A pool of top-performing molecules is maintained, the LLM is prompted to generate analogs of pool members (e.g., similarity prompts), generated candidates are scored by the oracle, and the pool is updated by retaining top unique molecules. When improvement stalls, the LLM is fine-tuned on the pool using molecule–score paired prompts (e.g., prompts that include property/score tokens) so the model learns explicit structure–property relationships, generation then resumes with the adapted model. The procedure cycles: LLM-based proposal $\rightarrow$ oracle evaluation $\rightarrow$ pool update $\rightarrow$ occasional fine-tuning.

We use a ChemLlama checkpoint trained from Llama 3-1B. We tested the following range of hyperparameters: pool size $P \in \{10, 20, 40, 80\}$, number of prompts $I \in \{25, 50, 100, 200\}$, and number of similar molecules in the prompt $S \in \{1, 2, 3, 4\}$.

Reward Functions. ChemLlama scores a property value m against a target interval $[\ell, u]$. The score is 1 inside the interval and decays with the distance to the nearest endpoint outside it:

$$g(m; [\ell, u], \sigma) = \begin{cases} 1 & \text{if } \ell \leq m \leq u, \\ \exp\left(-\frac{1}{2} \left(\frac{\min(|m - \ell|, |m - u|)}{\sigma}\right)^2\right) & \text{otherwise.} \end{cases}$$

The widths are $\sigma = 2$ for the docking score, $\sigma = 0.1$ for QED, $\sigma = 0.7$ for SA, and $\sigma = 0.1$ for similarity. In the rewards below, x denotes a candidate molecule, and each g uses the interval and width of the property it is applied to.

For **hit optimization** tasks, the reward is

$$R(x) = g\left(s^{(\text{target})}(x)\right) g(\text{QED}(x)) g(\text{SA}(x)).$$

For **lead optimization** tasks, the reward is

$$R(x) = \frac{s^{(\text{target})}(x)}{20} g(\text{Sim}(x, x_0)) g(\text{QED}(x)) g(\text{SA}(x)),$$

where $\text{Sim}(x, x_0)$ is the similarity to the seed molecule x_0 .

For **specificity optimization** tasks, the reward penalizes anti-target binding through the clipped score gap

$$\text{gap}(x) = \text{clip}\left(\frac{s^{(\text{target})}(x)}{20} - \frac{s^{(\text{off})}(x)}{20}, 0, 1\right),$$

$$R(x) = \text{gap}(x) g\left(s^{(\text{target})}(x)\right) g\left(s^{(\text{off})}(x)\right) g(\text{QED}(x)) g(\text{SA}(x)).$$

C Additional Hyperparameter Transfer Analyses

We performed independent hyperparameter sweeps for each validation task and algorithm. For each task i , we identified the hyperparameter configuration maximizing performance on that task, then measured its performance on all other tasks j . Each matrix entry reports the change in task- j performance when using task- i hyperparameters versus task- j -optimal settings (after per-task normalization), so off-diagonal structure summarizes cross-task transfer rather than raw scores. This protocol quantifies whether tuning signals learned on one objective or target are predictive for others and motivates model selection strategies that avoid test-set leakage.

Figures 9 and 10 show transfer matrices for Saturn, GenMol, GFlowNet and ChemLlama. Each cell (i, j) reports the relative change in task- j performance when using hyperparameters selected from source task i instead of task- j -optimal hyperparameters, after task-type normalization. Thus the heatmaps summarize transfer loss relative to the target-task optimum rather than absolute task performance.

Reading the off-diagonal cells qualitatively, Saturn, ChemLlama, and GenMol degrade more when the source and target lie in *different* categories than when they share one, whereas Genetic GFN shows only a small within-versus-cross difference, consistent with its evolutionary search being comparatively insensitive to task boundaries.

D All-but-me hyperparameter selection

Figure 2 uses category-specific selection. Figure 11 is the same per-task comparison when each task instead takes the hyperparameters that maximize average performance on all other validation tasks, including tasks from other categories. Layout and coloring match Figure 2.

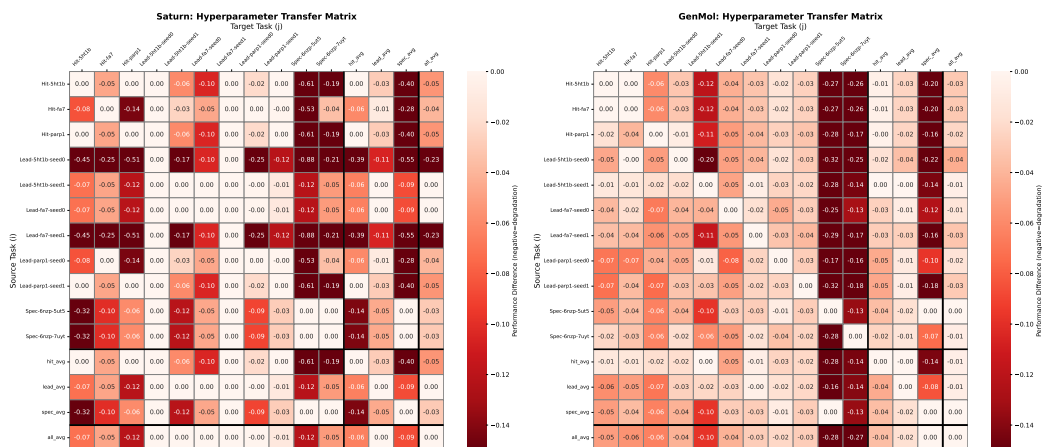


Figure 9: Hyperparameter transfer matrices for Saturn (left) and GenMol (right). Cell (i, j) shows the change in task- j performance when using hyperparameters selected from source task i instead of task- j -optimal hyperparameters. Scores are normalized by task type before comparison (hit rates direct, lead scores divided by 20, specificity scores divided by 5). Because entries are measured relative to the target-task optimum, values are expected to be non-positive, values near 0 indicate little transfer loss, while more negative values indicate larger degradation. These matrices summarize relative transfer rather than absolute task performance. Thick separators distinguish individual tasks from aggregate rows and columns (hit_avg, lead_avg, spec_avg, all_avg).

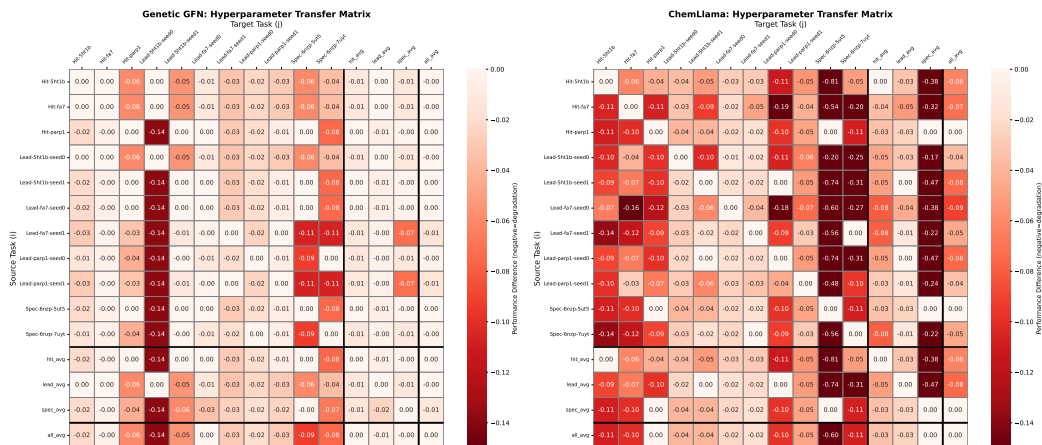


Figure 10: Hyperparameter transfer matrices for Genetic GFlowNet (left) and ChemLlama (right). As in Figure 9, each cell (i, j) reports the change in task- j performance when using hyperparameters selected from source task i instead of task- j -optimal hyperparameters. Scores are normalized by task type before comparison (hit rates direct, lead scores divided by 20, specificity scores divided by 5). Values are expected to be non-positive because the reference is the target-task optimum, entries near 0 indicate little transfer loss and more negative entries indicate larger degradation.

	Hit 5ht1b	Hit fa7	Hit parp1	Lead 5ht1b seed0	Lead 5ht1b seed1	Lead fa7 seed0	Lead fa7 seed1	Lead parp1 seed0	Lead parp1 seed1	Spec 6npz 5ut5	Spec 6npz 7ut5	Hit Avg	Lead Avg	Spec Avg
GenMol	0.24 (0.21,0.28)	0.08 (0.05,0.12)	0.08 (0.05,0.12)	11.80 (11.50,12.20)	11.00 (10.80,12.00)	7.60 (7.20,7.80)	7.63 (7.50,7.70)	11.60 (11.40,11.80)	11.00 (10.80,11.10)	2.62 (2.15,2.89)	1.46 (1.18,1.72)	0.13 (0.11,0.18)	10.11 (9.40,10.52)	2.04 (1.93,2.19)
Genetic GFN	0.39 (0.34,0.44)	0.02 (0.01,0.02)	0.11 (0.09,0.13)	4.50 (4.30,4.50)	11.50 (11.00,12.40)	8.23 (8.10,8.40)	6.83 (6.70,7.10)	11.40 (11.30,11.60)	10.77 (10.40,11.40)	2.34 (1.72,2.66)	1.82 (1.43,2.15)	0.17 (0.15,0.19)	8.87 (8.67,9.18)	2.08 (1.94,2.17)
Saturn	0.83 (0.80,0.85)	0.29 (0.25,0.34)	0.59 (0.55,0.64)	4.50 (4.50,4.50)	11.10 (10.30,12.90)	8.50 (8.50,8.50)	6.70 (6.70,6.70)	12.27 (12.00,12.40)	10.10 (9.90,11.50)	5.21 (4.30,6.54)	2.15 (1.93,2.28)	0.57 (0.54,0.60)	8.86 (8.57,9.05)	3.68 (3.32,4.40)
ChemLlama	0.43 (0.40,0.46)	0.30 (0.25,0.36)	0.52 (0.47,0.56)	12.50 (12.50,12.50)	10.80 (10.20,11.20)	8.67 (8.50,8.80)	7.40 (7.20,7.50)	9.27 (7.30,10.80)	12.33 (12.10,12.50)	4.72 (3.74,5.81)	3.75 (3.30,4.00)	0.41 (0.39,0.44)	10.16 (9.80,10.48)	4.23 (3.52,4.87)

Figure 11: Validation performance under all-but-me hyperparameter selection. For each task, hyperparameters maximize average performance on every other validation task. The last three columns are means over the tasks in each category (Hit Avg, Lead Avg, Spec Avg). Colors are normalized separately by category.

E Additional Statistical Analyses

This appendix reports bootstrap confidence intervals for leaderboard tables and pairwise method comparisons on test tasks. Table cells and aggregate columns in Figures 2, 3, and 4 use the interval definitions in Appendix E.1. Three further bootstrap analyses on test scores follow: *category-level win matrices* (Figure 5, summarized in Section 3.6) resample (task, seed) pairs jointly within each category, *per-task win matrices* (Figure 12) fix one test task and resample seeds only and *hit-rate distributions* (Figures 13 and 14) provide a distributional view for test hit tasks using the same seed draws as the hit panels in Figure 12.

E.1 Leaderboard table confidence intervals

Figures 2, 3, and 4 annotate each cell with the mean score across evaluation seeds for the selected hyperparameter configuration, followed by an equal-tailed 95% bootstrap confidence interval $[lo, hi]$ from $B = 10,000$ seed resamples. Per-task intervals reflect seed-level variability for the reported configuration only. Evaluation seed set sizes are $|\mathcal{S}| = 10$ for hit tasks and $|\mathcal{S}| = 3$ for lead and specificity tasks. The main tables omit **Normalized Avg**, **Hit Avg**, **Lead Avg**, and **Spec Avg** are means over tasks in each category, and **Normalized Avg** is

$$\text{Normalized_Avg} = \frac{1}{3} \left(\text{Hit_Avg} + \frac{\text{Lead_Avg}}{20} + \frac{\text{Spec_Avg}}{5} \right).$$

Bootstrap CIs for aggregates resample per-seed category means using fixed category-level seed-index shared across methods within each category. The Normalized Avg interval combines the three category bootstrap draws at each replicate.

E.2 Category-level cross-task bootstrap

Figure 5 reports pairwise win percentages under joint resampling of test tasks and seeds within each evaluation category. For each category we form a finite pool of (task block, seed) pairs from the same non-aggregate test blocks used in Figure 3, with pool sizes $K_{\text{hit}} = 20$ (two tasks $\times$ ten seeds), $K_{\text{lead}} = 27$ (nine tasks $\times$ three seeds), and $K_{\text{spec}} = 9$ (three tasks $\times$ three seeds). Across $B = 10,000$ replicates we draw K pairs uniformly with replacement from that pool, each pair equally likely on every draw and average each method’s score over the K drawn (task, seed) units. Cell (i, j) in a category matrix is the percentage of replicates with $\bar{s}_i^{(b)} > \bar{s}_j^{(b)}$ (diagonal zero, ties uncounted). The rightmost Σ panel sums row win totals across the three category matrices.

E.3 Per-task seed bootstrap win matrices

This analysis complements the category-level bootstrap above with a finer-grained per-task view that fixes the task block and resamples only RNG seeds. For every non-aggregate test block (two hit, nine lead, three specificity) and each of $B = 10,000$ replicates, we draw $|\mathcal{S}|$ seeds with replacement from that block’s empirical seed set, average every method’s score over the resampled seeds, and form pairwise win percentages. The resulting fourteen 4×4 matrices in Figure 12 quantify how stable the per-task ranking is to seed variation.



Figure 12: **Per-task bootstrap win percentages.** One 4×4 heatmap per test task (fourteen panels: two hit, nine lead, three specificity). Rows and columns are methods (ChemLlama, Genetic GFN, GenMol, Saturn), cell (i, j) is the percentage of $B = 10,000$ bootstrap replicates in which the row method's resampled mean strictly exceeds the column method's under seed resampling for that fixed task. Each replicate resamples seeds with replacement from the task's empirical seed set and averages each method's score over the resampled seeds. Colour encodes percentage on a common 0–100% scale across panels (darker = higher).

E.4 Hit-rate bootstrap on test hit tasks

For test hit tasks we additionally report bootstrap *distributions* of mean hit rate and paired Saturn-minus-competitor contrasts. We also include a BOTH summary that averages BRAF and JAK2 hit rates at each (m, s) before resampling. We use Saturn as the paired reference because it leads on hit tasks in the test leaderboard.

For each task summary $t \in \{\text{BRAF}, \text{JAK2}, \text{BOTH}\}$ we collect the observed hit rate $x_{m,s}$ for every method m and evaluation seed s in the empirical seed set $\mathcal{S}$. Across $B = 10,000$ replicates we draw $|\mathcal{S}|$ seeds independently with replacement from $\mathcal{S}$ and form

$$\bar{x}_m^{(b)} = \frac{1}{|\mathcal{S}|} \sum_{k=1}^{|\mathcal{S}|} x_{m, s_k^{(b)}}, \quad \bar{\Delta}_m^{(b)} = \frac{1}{|\mathcal{S}|} \sum_{k=1}^{|\mathcal{S}|} (x_{\text{Saturn}, s_k^{(b)}} - x_{m, s_k^{(b)}}), \quad (1)$$

i.e. the bootstrap replicate of method m 's mean hit rate and of its paired contrast against Saturn on the same resampled seeds. We summarize each bootstrap distribution by its mean and by the equal-tailed 95% percentile interval $\text{CI}_{95}(\cdot)$.

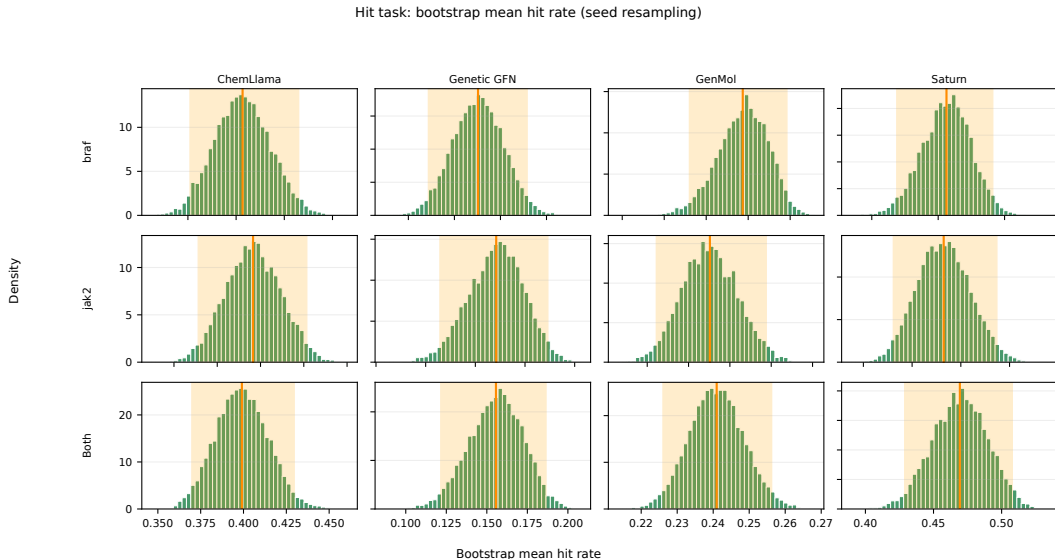


Figure 13: **Bootstrap distributions of mean hit rate under seed resampling.** Rows are the three task summaries (BRAF, JAK2, BOTH), columns are methods. Each panel shows a normalized histogram of $\bar{x}_m^{(b)}$ over $B = 10,000$ replicates, the vertical orange line is the bootstrap mean and the shaded band is $\text{CI}_{95}(\bar{x}_m)$.

Figure 13 reports the bootstrap distribution of $\bar{x}_m^{(b)}$ for every (method, task-summary) cell. Wider histograms indicate that the mean hit rate is more sensitive to which seeds are emphasized.

Figure 14 reports the paired contrast $\bar{\Delta}_m^{(b)}$ for every competitor $m \neq \text{Saturn}$ and every task summary. For all reported pairs (m, t) the interval $\text{CI}_{95}(\bar{\Delta}_m)$ lies strictly above zero, consistent with Saturn's dominant win percentages against each competitor in the hit panels of Figures 5 and 12.

F Additional Hyperparameter Transfer Scatter Analyses

This appendix supplements Section 4 and the transfer matrices in Appendix C with validation-task scatter plots. Each point is one hyperparameter configuration from the validation sweep, axes compare performance on two tasks, targets, or constraint settings. Legends report Pearson correlations and sample sizes.

Hit task: paired differences vs. Saturn (seed resampling)

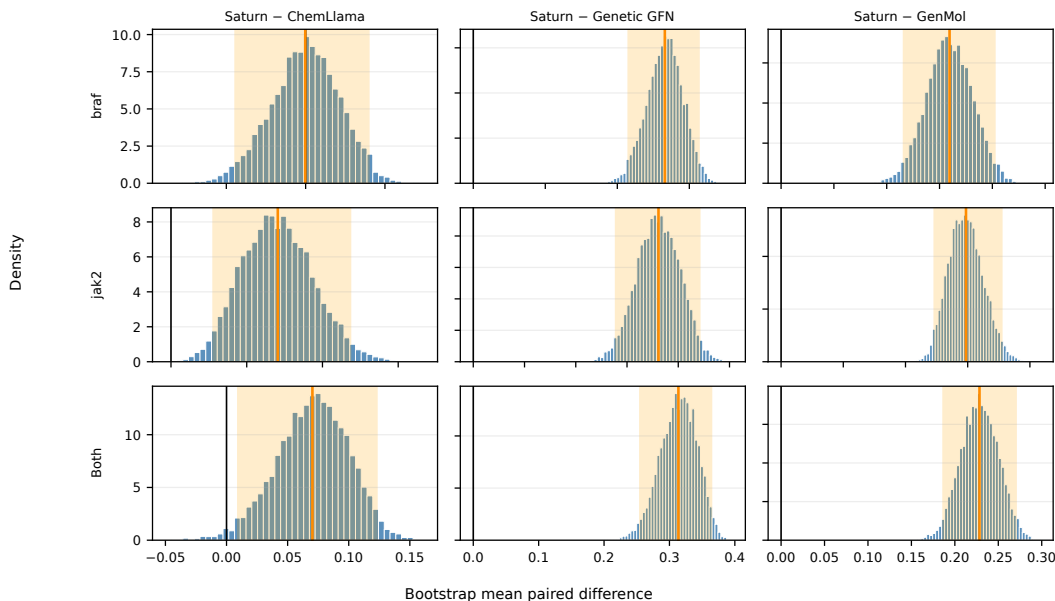


Figure 14: **Bootstrap distributions of paired mean differences (Saturn – competitor).** Rows are the three task summaries (BRAF, JAK2, BOTH), columns are competitors $m \neq$ Saturn. Each panel histograms $\bar{\Delta}_m^{(b)}$ over $B = 10,000$ replicates. The black line marks zero, the orange line marks the bootstrap mean, and the shaded band is $CI_{95}(\bar{\Delta}_m)$.

F.1 Cross-Target Hit Generation: Additional Protein Pairs

Figures 15, 16, and 17 show hit-generation performance correlations across protein pairs (one subplot per method). Saturn exhibits strong positive alignment on all pairs ($r \in [0.90, 0.94]$, $n=16$). ChemLlama is moderate on parp1–fa7 ($r=0.61$) but weak on 5ht1b–fa7 ($r=0.06$). Genetic GFN shows mixed low- n correlations ($n=9$). GenMol remains near zero on all pairs ($n=120$). These patterns support the within-category transfer discussion in Section 4.

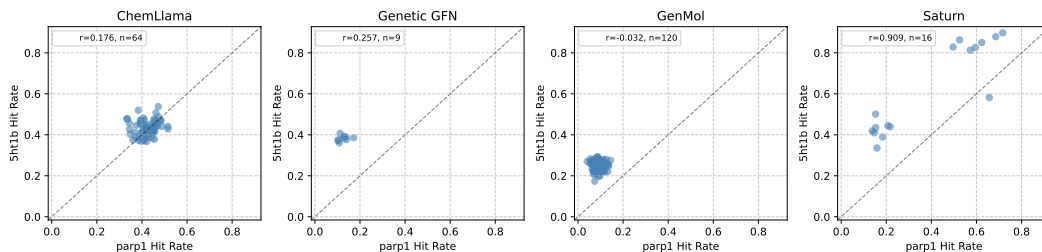


Figure 15: Cross-target hit generation performance: parp1 vs 5ht1b. Each point represents a hyperparameter configuration, averaged across random seeds. One panel per method, legends report Pearson correlations and sample sizes.

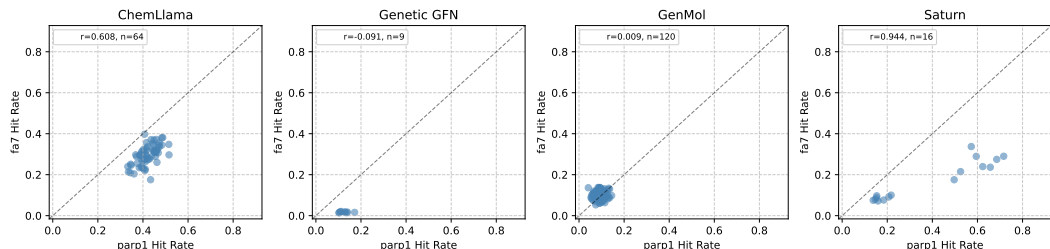


Figure 16: Cross-target hit generation performance: parp1 vs fa7. Each point represents a hyperparameter configuration, averaged across random seeds.

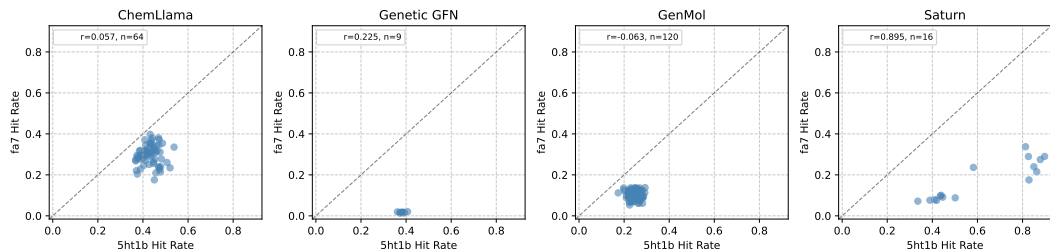


Figure 17: Cross-target hit generation performance: 5ht1b vs fa7. Each point represents a hyperparameter configuration, averaged across random seeds.

F.2 Lead Optimization: Threshold Transfer

Figure 18 examines transfer when varying the similarity threshold δ in lead optimization tasks. Each color represents a different (target, seed molecule) combination. The scatter reveals that hyperparameters optimized for $\delta = 0.4$ do not reliably transfer to $\delta = 0.6$, indicating sensitivity to constraint tightness. In a few subsets the reported Pearson correlation is undefined because one threshold yields constant scores across experiments, so the variance needed for correlation is zero.

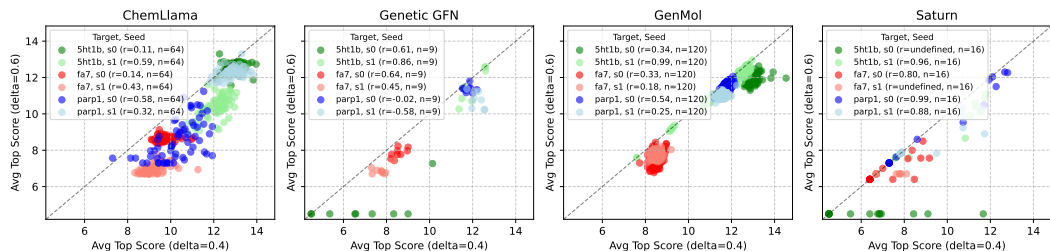


Figure 18: Lead optimization performance comparison across threshold values ($\delta = 0.4$ vs $\delta = 0.6$). Each color represents a specific (target, seed molecule) pair. Legends report Pearson correlations and sample sizes. Labels showing $r = \text{undefined}$ correspond to subsets where one threshold produces constant scores across experiments, making Pearson correlation undefined.

F.3 Specificity Task Transfer

Figure 19 compares performance on two *validation* specificity tasks (6nzp-5ut5 vs 6nzp-7uyt). The specificity score is computed as $\text{dock}_{\text{target}} - \text{dock}_{\text{off-target}}$. Correlation between the two validation pairs varies by method (more positive for ChemLlama and Saturn, weaker or more uncertain for GenMol and Genetic GFN), so strong performance on one pair does not guarantee strong performance on the other.

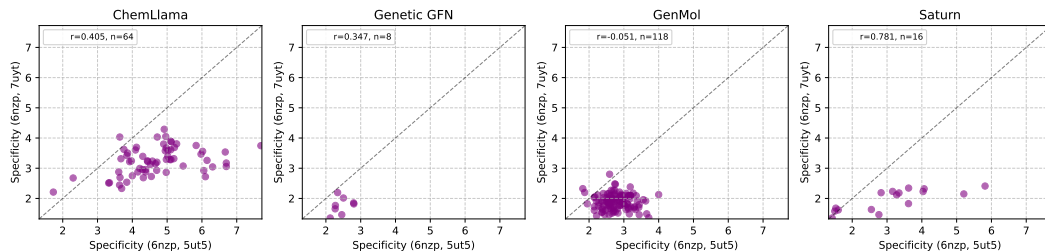


Figure 19: Comparison of specificity task performance (6nzp-5ut5 vs 6nzp-7uyt). Axes: mean top-10 specificity score (delta) for each pair. Legends report Pearson correlations and sample sizes. Correlation is method-dependent, indicating incomplete transfer across antitarget choices.

F.4 Held-Out Target Generalization

Figure 20 examines whether hyperparameters optimized on fa7 and parp1 transfer to 5ht1b as a held-out target. The 2x4 grid shows 5ht1b hit performance on the Y-axis throughout. In the top row, the X-axis shows average hit performance on fa7 and parp1. In the bottom row, the X-axis shows combined hit and lead performance (normalized) on fa7 and parp1. This simulates the realistic scenario where validation tasks guide hyperparameter selection and test tasks measure true generalization. Saturn shows substantially stronger positive alignment between validation-side aggregates and 5ht1b hit rate than the other methods, so aggregate validation performance does not uniformly predict held-out 5ht1b hit rate.

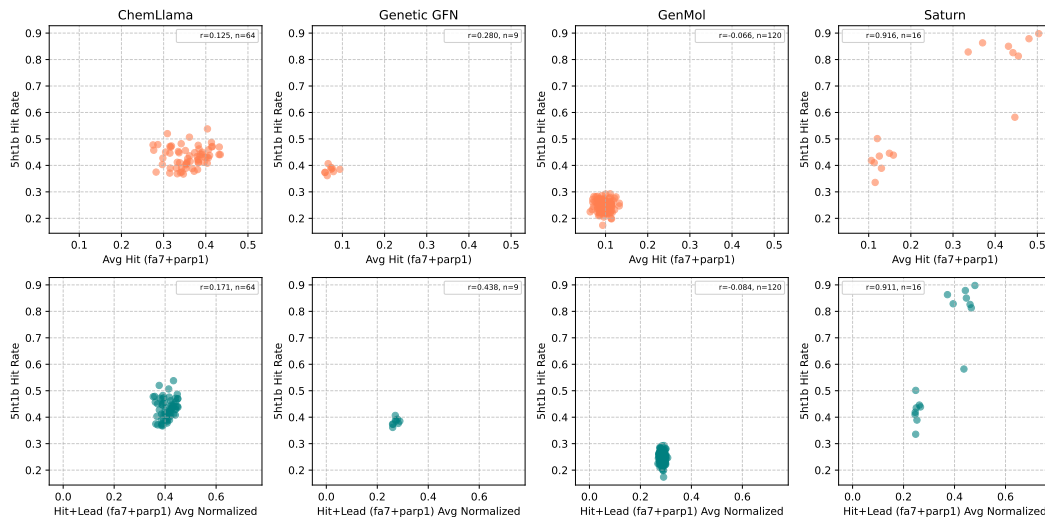


Figure 20: Held-out target transfer analysis: 5ht1b hit performance versus other validation tasks. Top row: 5ht1b hit vs average hit on fa7+parp1. Bottom row: 5ht1b hit vs combined hit+lead on fa7+parp1 (normalized). Each column represents one algorithm. Legends report Pearson correlations and sample sizes. Saturn shows substantially stronger positive alignment than the other methods.

G Details of the Docking Pipeline for Specificity Tasks

Hit and lead tasks use QuickVina with exhaustiveness 1 as stated in Section 2. The specificity tasks use the same scoring function but a separate docking pipeline with higher exhaustiveness for the larger binding boxes described below.

Docking evaluations were performed using QuickVina [Alhossary et al., 2015]. For all five specificity tasks in the benchmark (two validation, three test), the binding box was defined around the ligands observed in the crystallographic structures, ensuring that all amino acid residues participating in ligand interactions were included. The grid box volume did not exceed 27,000 Å³ in any case.

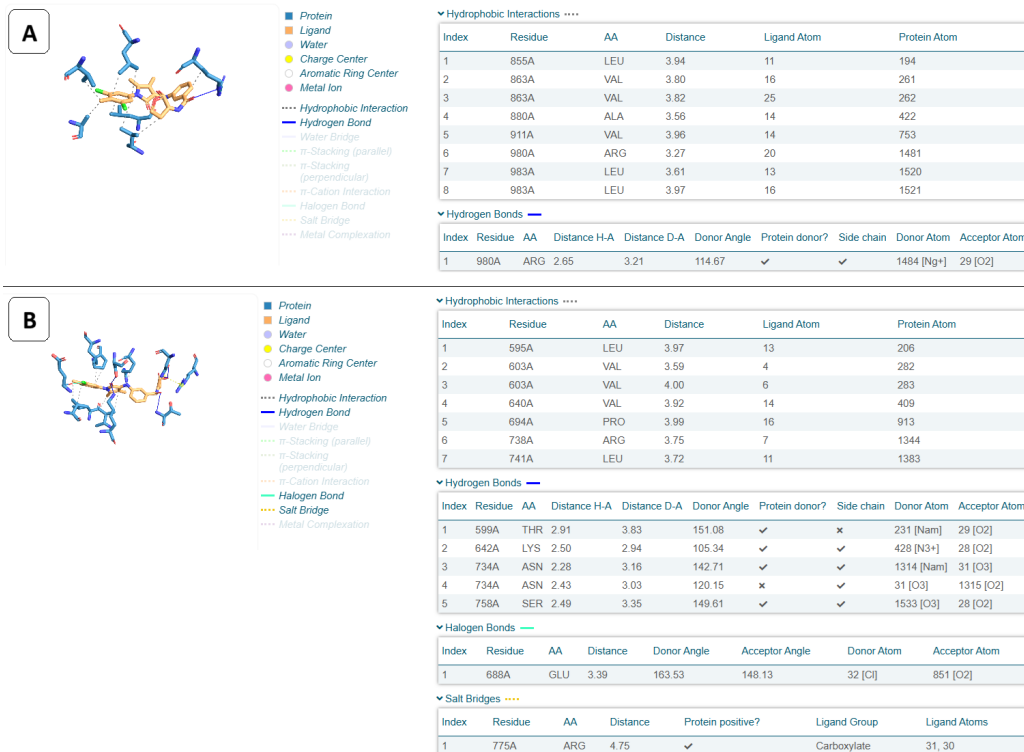


Figure 22: Representative binding poses of a molecule generated by ChemLlama from the specificity optimization task. A) Binding pose of the generated molecule in complex with the anti-target TYK2 JH1 (7UYT). B) Binding pose of the same molecule in complex with the target TYK2 JH2 (6NZP).

Table 1: Hit generation hit rates for ChemLlama at QuickVina exhaustiveness 1 versus 8. Entries are mean $\pm$ standard deviation.

Target	Hit rate (ex= 1)	Hit rate (ex= 8)
PARP1	0.4570 $\pm$ 0.0791	0.5534 $\pm$ 0.0901
FA7	0.3693 $\pm$ 0.0774	0.3776 $\pm$ 0.0917
5HT1B	0.4681 $\pm$ 0.0662	0.6405 $\pm$ 0.0476

Higher exhaustiveness mostly refines scores, and exhaustiveness 1 still appears to capture the relative ranking of strong candidates. On a single CPU core, docking takes about 0.05 s/molecule at exhaustiveness 1 versus about 0.25 s/molecule at exhaustiveness 8. Given the consistent improvements at higher exhaustiveness, using exhaustiveness 8 is a reasonable option when extra compute is available.

Table 2: Hit generation top-1 docking scores for ChemLlama at QuickVina exhaustiveness 1 versus 8. Entries are mean $\pm$ standard deviation.

Target	Top-1 DS (ex= 1)	Top-1 DS (ex= 8)
PARP1	12.490 $\pm$ 0.532	12.830 $\pm$ 0.754
FA7	10.110 $\pm$ 0.367	10.090 $\pm$ 0.335
5HT1B	11.780 $\pm$ 0.461	12.240 $\pm$ 0.744

Table 3: Lead optimization top-1 docking scores for ChemLlama at QuickVina exhaustiveness 1 versus 8 ($\delta = 0.6$). Entries are mean $\pm$ standard deviation.

Task	Top-1 DS (ex= 1)	Top-1 DS (ex= 8)
lead.5ht1b_06_0	11.667 $\pm$ 0.153	12.300 $\pm$ 0.900
lead.5ht1b_06_1	10.667 $\pm$ 0.764	11.300 $\pm$ 0.700
lead.fa7_06_0	9.000 $\pm$ 0.200	9.133 $\pm$ 0.231
lead.fa7_06_1	8.367 $\pm$ 0.379	8.333 $\pm$ 0.404
lead.parp1_06_0	10.933 $\pm$ 0.577	10.733 $\pm$ 0.153
lead.parp1_06_1	12.267 $\pm$ 0.231	12.800 $\pm$ 0.200

H.2 Adequacy of QuickVina relative to AutoDock Vina

To highlight the adequacy of QuickVina we compare it with AutoDock Vina (1.2.7), an established method for molecular docking [Trott and Olson, 2010, Eberhardt et al., 2021]. We set aside and redock the molecules generated by each method on hit generation and lead optimization tasks with AutoDock Vina. Table 4 reports the Pearson correlation of the QuickVina and AutoDock Vina docking scores across all molecules generated during optimization (*all*) and across successful molecules (*successful*), aggregated over the target proteins.

Table 4: Pearson correlation between QuickVina and AutoDock Vina docking scores for all generated molecules and for successful molecules, by method and task category.

Method	Hit all	Hit successful	Lead all	Lead successful
ChemLlama	0.827	0.845	0.676	0.886
Saturn	0.962	0.928	0.882	0.886
Genetic GFN	0.972	0.924	0.959	0.885
GenMol	0.922	0.913	0.875	0.680

We observe that QuickVina and AutoDock Vina are well correlated across methods and molecule sets (*all* and *successful*). These correlation values are also consistent with previously computed correlations between QuickVina and AutoDock Vina on non-AI-optimized molecules [Alhossary et al., 2015], which was 0.967. We believe that this adds strong supporting evidence that the optimization methods do not exploit the docking dynamics of QuickVina and that high-scoring molecules remain high-scoring in AutoDock Vina, highlighting the adequacy of QuickVina as the docking software for this benchmark. Additionally, QuickVina is considerably faster than AutoDock Vina, making it the desired choice for this benchmark.

H.3 Physical plausibility of high-scoring poses

To address whether high-scoring molecules exploit docking artifacts, we additionally tested the highest-scoring molecules across all four methods (1,335 molecules in total), together with their docked poses, using PoseBusters [Buttenschoen et al., 2024].

For each molecule, we regenerated a local QuickVina docking pose on the corresponding receptor, preprocessed the resulting complex by removing and re-adding hydrogens to avoid PDBQT conversion artifacts, and evaluated the docked pose with PoseBusters in docked-ligand mode.

Across all methods and tasks, 55.7% of molecules passed all PoseBusters checks. Pass rates were higher for the hit (59.2%) and lead (56.3%) evaluations, and lower for the specificity evaluation (42.9%). As shown in Table 5, per-method pass rates range from 50.8% to 60.6%, with ChemLlama the lowest.

We also redocked the 308 ligands from the PoseBusters benchmark with QuickVina, using the crystal protein and the same PoseBusters protocol as above. 77.3% (238/308) passed all checks, and 78.3% of the 304 successfully docked ligands passed. Under this matched docking and PoseBusters stack, our top-scoring generated molecules pass at a lower rate (55.7%).

Table 5: PoseBusters pass rate for the highest-scoring molecules of each method. A molecule passes only if it satisfies every PoseBusters check.

Method	Pass rate
Genetic GFN	60.6%
GenMol	57.7%
Saturn	55.0%
ChemLlama	50.8%

I Experimental Methodology and Community Resources

This appendix provides hyperparameter sweep procedures, task-specific thresholds, and community resources. The validation/test split, task overview (Figure 1), and model selection protocol are described in Section 2.4.

I.1 Hyperparameter Sweep Details

We ran independent hyperparameter sweeps on every validation task for Saturn, GenMol, Genetic-guided GFlowNet, and ChemLlama. Transfer matrices, the category-specific and all-but-me selection rules, and test evaluation follow Sections 4 and 3. All tuning was confined to validation tasks. Test reporting uses category-specific selection only, with one evaluation per method. Complete search spaces and per-method training details are in Appendix B.

I.2 Task-Specific Thresholds

I.2.1 Hit Generation Thresholds

For hit generation tasks, the target-specific docking score thresholds $s_{\min}^{(t)}$ are:

$$s_{\min}^{(\text{PARP1})} = 10.0, \quad s_{\min}^{(\text{FA7})} = 8.50, \quad s_{\min}^{(5\text{HT1B})} = 8.7845, \quad s_{\min}^{(\text{BRAF})} = 10.30, \quad s_{\min}^{(\text{JAK2})} = 9.10.$$

I.2.2 Specificity Tasks: Reference Docking Score and Property Filters

Let $s_{\text{ref}}^{(\text{spec})} := s^{(\text{TYK2 JH2})}(\text{Deucravacitinib})$ denote the QuickVina docking score of the reference inhibitor Deucravacitinib (BMS-986165) against TYK2 JH2 under the protocol in Appendix G. We use

$$s_{\text{ref}}^{(\text{spec})} = 10.67.$$

Molecules must satisfy $s^{(\text{TYK2 JH2})}(x) \geq s_{\text{ref}}^{(\text{spec})}$ to count as hits for the specificity objective.

I.3 Compute Resources

We have utilized one DGXH100 box, containing 8 NVIDIA H100 GPUs with 80Gb of memory for running hyperparameter sweeps for the different methods and 128 CPU core-processors for deploying and running the QuickVina simulations.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope?

Answer: [\[Yes\]](#)

Justification: We articulate the paper’s scope and contributions in the abstract and the introduction.

Guidelines:

- The answer [\[N/A\]](#) means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [\[No\]](#) or [\[N/A\]](#) answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: Limitations are discussed in A.

Guidelines:

- The answer [\[N/A\]](#) means that the paper has no limitation while the answer [\[No\]](#) means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate “Limitations” section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren’t acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [\[N/A\]](#)

Justification: The paper does not include any theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We provide the code used for the experimental results at <https://github.com/YerevaNN/PMO-Dock>.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide the code used for the experimental results at <https://github.com/YerevaNN/PMO-Dock>. The paper does not include any dataset.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://neurips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://neurips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: Experiment settings are detailed in 2 and 4 and the references appendix sections within these sections.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: The paper reports the error bars, confidence intervals and statistical significance throughout.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: See section I.3 [TODO]

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: The research is conducted with the NeurIPS Code of Ethics.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [N/A]

Justification: There is no broader societal impact.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper poses no such risks.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: Section 2 and 3 (methods, task setups).

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset’s creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [\[Yes\]](#)

Justification: They are available in Github repository.

Guidelines:

- The answer [\[N/A\]](#) means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [\[N/A\]](#)

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [\[N/A\]](#) means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [\[N/A\]](#)

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [\[N/A\]](#) means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method of the research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.