SAMPLE EFFICIENT GENERATIVE MOLECULAR OPTIMIZATION WITH JOINT SELF-IMPROVEMENT

Serra Korkmaz[1*] **Adam Izdebski**[1,2*] **Jonathan Pirnay**[3,4] **Rasmus Møller-Larsen**[1,2]
Michal Kmicikiewicz[1,2] **Pankhil Gawade**[1,2] **Dominik G. Grimm**[2,3,4] **Ewa Szczurek**[1,5]

[1]Institute of AI for Health, Helmholtz Zentrum München
[2]Technical University of Munich, TUM School of Computation, Information and Technology
[3]Technical University of Munich, Campus Straubing for Biotechnology and Sustainability
[4]University of Applied Sciences Weihenstephan-Triesdorf
[5]Faculty of Mathematics, Informatics and Mechanics, University of Warsaw

{serra.korkmaz, adam.izdebski, ewa.szczurek}@helmholtz-munich.de

ABSTRACT

Generative molecular optimization aims to design molecules with properties surpassing those of existing compounds. However, such candidates are rare and expensive to evaluate, yielding sample efficiency essential. Additionally, surrogate models introduced to predict molecule evaluations, suffer from distribution shift as optimization drives candidates increasingly out-of-distribution. To address these challenges, we introduce JOINT SELF-IMPROVEMENT, which benefits from (i) a **joint generative-predictive model** and (ii) a **self-improving sampling scheme**. The former aligns the generator with the surrogate, alleviating distribution shift, while the latter biases the generative part of the joint model using the predictive one to efficiently generate optimized molecules at inference-time. Experiments across offline and online molecular optimization benchmarks demonstrate that JOINT SELF-IMPROVEMENT outperforms state-of-the-art methods under limited evaluation budgets.

1 INTRODUCTION

Generative molecular optimization (GMO) is a cornerstone of modern drug discovery, with the aim to generate molecules that improve upon existing compounds Fischer et al. (2019); Sridharan et al. (2022); Ramos et al. (2025); Zhang et al. (2025). Yet, optimized molecules are exceedingly rare, and the evaluation of newly generated candidates is prohibitively expensive, as it typically amounts to wet-lab experimentation Hughes et al. (2011); Özçelik et al. (2025).

As a result, GMO methods need to generalize from only a few positive examples, while operating under strict evaluation budgets, making sample efficiency essential Guo & Schwaller (2024). Nevertheless, state-of-the-art methods Guo et al. (2026); Shin et al. (2025), such as REINVENT Loeffler et al. (2024), predominantly rely on reinforcement learning (RL) with policy-gradient updates, where a generative model is treated as a policy and fine-tuned via reinforcement learning fine-tuning (RLFT) Murphy (2025). As molecules that yield positive reinforcement are rare, policy gradients produce high-variance updates, making fine-tuning unstable and sample inefficient Zhao et al. (2011); Agarwal et al. (2020); Rengarajan et al. (2022); Vasan et al. (2024).

To enable efficient learning, wet-lab evaluation is often approximated with a *surrogate*, typically a DNN trained in a supervised manner Shin et al. (2025). The surrogate serves as a plug-in oracle estimator in RLFT. Surrogates are particularly important in *offline optimization*, a setting representative of real-world drug discovery pipelines, where the oracle cannot be queried during optimization Kim et al. (2025). However, decoupled surrogate and generative model training introduces *distribution*

[*]Equal contribution.

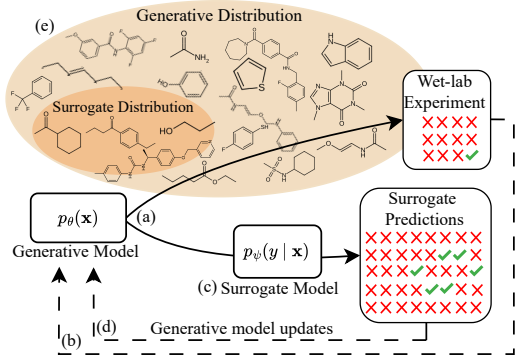

Figure 1: RL-based GMO starts from a pretrained generative model that unconditionally samples molecules for evaluation (a). However, optimized molecules are rare, resulting in high-variance updates (b). Surrogate models, used as plug-in estimators for molecule evaluation (c), aim to provide additional learning signal (d). However, decoupled generative model updates introduce a distribution shift, as optimized molecules become increasingly out-of-distribution for the surrogate (e).

shifts (see Figure 1): as RLFT progresses, new candidates are increasingly out-of-distribution for the surrogate, making predictions unreliable and leading to *objective hacking* Song et al. (2023).

To address these limitations, we introduce JOINT SELF-IMPROVEMENT, a unified framework for sample-efficient molecular optimization. To mitigate distribution shifts typical of decoupled molecular optimization pipelines, we build upon a recent joint model, Hyformer Izdebski et al. (2026), to align the generative model with the surrogate within using shared parameterization. Joint modeling provides synergistic benefits, such as improved controllability of the generative model and enhanced out-of-distribution robustness of the surrogate Izdebski et al. (2026). We fine-tune the joint model using a likelihood-based loss, alleviating high-variance updates and sample inefficiency typical of policy gradient methods. To further support efficient generation of optimized molecules, we propose a joint self-improving sampling scheme, derived from round-wise provable policy improvement Pirnay & Grimm (2024), in which the predictive part of the joint model biases the generative part at inference-time. To support our claims, we evaluate our framework on established sample-efficient molecular optimization benchmarks Guo et al. (2026); Shin et al. (2025). To summarize:

- We **align the generative model and surrogate** via a joint model with shared parameters and likelihood-based training, we *reducing high-variance updates and distribution shifts* common in decoupled pipelines.
- We introduce a **joint self-improving sampling scheme**, where the predictive part of the joint model biases the generative part at inference time, *enabling efficient generation of optimized molecules under strict evaluation budgets*.
- We show that our framework, JOINT SELF-IMPROVEMENT, outperforms state-of-the-art molecular optimization methods across both offline and online optimization benchmarks.

2 METHODOLOGY

Problem Statement In *molecular optimization*, the goal is to discover a new *molecule* $\mathbf{x}^* \in \mathcal{X}$ that maximizes an *objective function (oracle)* $f : \mathcal{X} \to \mathbb{R}^M$. Depending on the dimensionality M, if $M = 1$, molecular optimization reduces to a *single-objective* optimization (SOO) problem

$$\mathbf{x}^* = \arg\max_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}). \tag{1}$$

In practice, a constrained variant of SOO is common, where the design space $\mathcal{X}$ is restricted by feasibility constraints expressed through auxiliary objective functions, such as *synthetic accessibility* or *drug-likeliness*. In molecular optimization, evaluating the objective f is prohibitively expensive, as it typically requires performing a *wet-lab experiment*. In consequence, molecular optimization methods prioritize *sample efficiency*, i.e., discovering molecule $\mathbf{x} \in \mathcal{X}$ under a fixed *oracle budget* of objective function evaluations. Molecular optimization additionally depends on access to the objective

f: in the *online* setting, the optimization process iteratively queries f and adapts based on feedback, while the *offline* setting is restricted to a fixed *offline dataset* of molecules with pre-computed evaluations (see Appendix Figure 3). *Generative molecular optimization* typically employs a generative model $p_\theta(\mathbf{x})$ to sample optimized molecules.

Joint Model as a Unified Backbone We begin by defining a joint model $p_\theta(\mathbf{x}, y)$ that unifies, within a shared parameterization θ, the generative model $p_\theta(\mathbf{x})$ and a surrogate $p_\theta(y \mid \mathbf{x})$ Nalisnick et al. (2019); Grathwohl et al. (2020); Izdebski et al. (2026):

$$p_\theta(\mathbf{x}, y) = p_\theta(y \mid \mathbf{x})\, p_\theta(\mathbf{x}), \tag{2}$$

where both components share parameters θ, trained by minimizing the negative joint log-likelihood, i.e., the *joint loss*

$$\mathcal{L}_\lambda(\theta) = -\sum_{(\mathbf{x},y)\in\mathcal{D}} \Big(\log p_\theta(\mathbf{x}) + \lambda \log p_\theta(y \mid \mathbf{x})\Big), \tag{3}$$

where $\lambda \geq 0$ controls the relative strength of the generative and predictive counterparts.

Sampling with Joint Self-Improvement Next, we introduce a joint self-improving sampling scheme that biases, at inference-time, the generative part $p_\theta(\mathbf{x})$ with advantages estimated using the predictive component $p_\theta(y \mid \mathbf{x})$. Rather than defining the advantages on partial sequences, we exploit the autoregressive factorization of the generative part $p_\theta(\mathbf{x}) = \prod_{t=1}^{T} p_\theta(x_t \mid \mathbf{x}_{1:t-1})$, and define the *individual advantage* Pirnay & Grimm (2024) of token $x_t \in \mathcal{V}$ given a prefix $\mathbf{x}_{1:t-1}$, as

$$\mathrm{Adv}_\theta(x_t \mid \mathbf{x}_{1:t-1}) := \mathbb{E}_{p_\theta(.|x_t,\mathbf{x}_{1:t-1})}\big[p_\theta(y \mid \mathbf{x})\big] - \mathbb{E}_{p_\theta(.|\mathbf{x}_{1:t-1})}\big[p_\theta(y \mid \mathbf{x})\big], \tag{4}$$

where both expectations are taken over completed sequences $\mathbf{x} = (x_1, \ldots, x_T)$ estimated via the generative part $p_\theta(\mathbf{x})$. Using the individual advantage, we define the logits of the *perturbed model* $p_{\theta'}(\mathbf{x})$, given by locally tilting the next-token log-probabilities of the generative model $p_\theta(\mathbf{x})$

$$\operatorname{logit} p_{\theta'}(x_t \mid \mathbf{x}_{1:t-1}) := \log p_\theta(x_t \mid \mathbf{x}_{1:t-1}) + \sigma\, \mathrm{Adv}_\theta(x_t \mid \mathbf{x}_{1:t-1}), \tag{5}$$

where $\sigma > 0$ is the *step size*. Depending on the step size σ, the perturbed model $p_{\theta'}(\mathbf{x})$ interpolates between generating molecules that are likely under the generative model $p_\theta(\mathbf{x})$, similar to methods based on rejection-sampling Izdebski et al. (2026), and exploring low-density regions of $p_\theta(\mathbf{x})$. While sampling from the perturbed model $p_{\theta'}(\mathbf{x})$ yields a provable improvement over sampling from $p_\theta(\mathbf{x})$, estimating individual advantages can result in high-variance updates; and estimating advantages over completed sequences, i.e., globally, has been shown to provide stronger empirical performance Pirnay & Grimm (2024). Consequently, the practical implementation of our method relies on global advantages, as detailed in the next section.

Practical Updates with Global Advantage For a practical implementation of the self-improving sampling scheme (Figure 2), we define advantages calculated over complete sequences $\mathbf{x} = (x_1, \ldots, x_T)$, similar to Pirnay & Grimm (2024). We begin by drawing K-many samples $\{\mathbf{x}^{(k)}\}_{k=1}^{K} \sim p_\theta(\mathbf{x})$, without replacement, using Stochastic Beam Search (SBS). This is achieved by applying the *Gumbel-Top-k trick* to the autoregressive model $p_\theta(\mathbf{x})$, requiring a number of model evaluations that scales linearly with the number of samples K and sequence length T Kool et al. (2019). Next, for every sequence $\mathbf{x}^{(k)} \in \{\mathbf{x}^{(k)}\}_{k=1}^{K}$, we compute the *global advantage*

$$\mathrm{Adv}_\theta(\mathbf{x}^{(k)}) = p_\theta(y \mid \mathbf{x}^{(k)}) - \mu(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(K)}), \tag{6}$$

where $\mu(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(K)})$ is a consistent estimator of the *expected prediction score* $\mathbb{E}_{p_\theta(\mathbf{x})}[p_\theta(y \mid \mathbf{x})]$ (e.g., Eq. 4 in Pirnay & Grimm 2024). Finally, for each $\mathbf{x}^{(k)}$ and index $t \in [T]$, we define the logits of the perturbed model $p_{\theta'}(\mathbf{x})$ as

$$\operatorname{logit} p_{\theta'}\big(x_t^{(k)} \mid \mathbf{x}_{1:t-1}^{(k)}\big) := \log\Big(p_\theta(x_t^{(k)} \mid \mathbf{x}_{1:t-1}^{(k)}) - \sum_{\mathbf{x}\in S(\mathbf{x}_{1:t}^{(k)})} p_\theta(\mathbf{x} \mid \mathbf{x}_{1:t-1}^{(k)})\Big) \quad + \sigma \sum_{\mathbf{x}\in S(\mathbf{x}_{1:t}^{(k)})} \mathrm{Adv}_\theta(\mathbf{x}), \tag{7}$$

where $S(\mathbf{x}_{1:t}^{(k)})$ is the set of all sequences $\mathbf{x} \in \{\mathbf{x}^{(k)}\}_{k=1}^{K}$ that share the prefix $\mathbf{x}_{1:t}^{(k)}$, and the logarithm is well-defined as the total probability mass of sequences in $S(\mathbf{x}_{1:t}^{(k)})$ is upper-bounded by the conditional probability $p_\theta(x_t^{(k)} \mid \mathbf{x}_{1:t-1}^{(k)})$. We iteratively repeat the above procedure by biasing the perturbed model $p_{\theta'}(\mathbf{x})$ with global advantages for a fixed number of iterations or until convergence (see Appendix E).

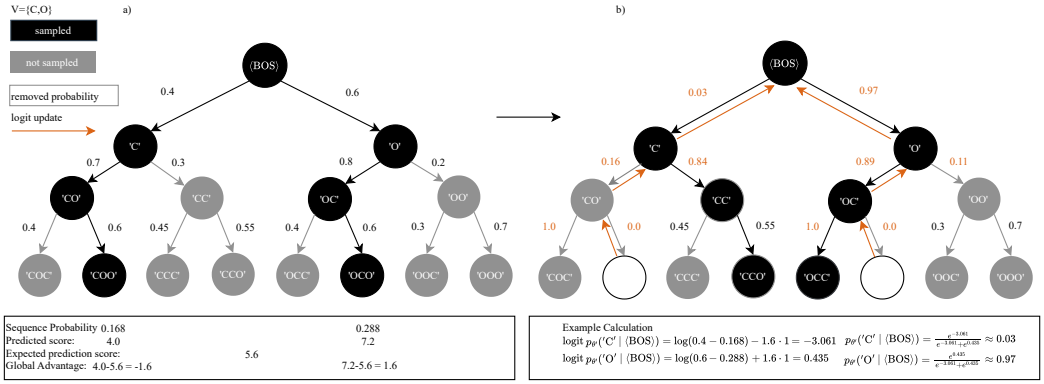

Figure 2: Example logit update (Eq. 7) in JOINT SELF-IMPROVEMENT with beam width $K{=}2$ and step size $\sigma{=}1$. Next-token probabilities are shown on the arrows. Starting from <BOS>, the model expands a sampling tree using SBS. The expected prediction score μ is used to compute a per-sampled-sequence global advantage, which is then used to obtain the perturbed distribution $p_{\theta'}(\mathbf{x})$. Panel (a) shows the initial sampling tree, while panel (b) shows the resulting normalized probabilities after removing sampled mass (white) and applying the global advantage.

3 EXPERIMENTS

To evaluate the performance of JOINT SELF-IMPROVEMENT in sample-efficient molecular optimization, we benchmark our method in both offline (Section 3.1) and online (Section 3.2) settings. All experimental details, together with additional ablations quantifying the contribution of joint learning and joint self-improving sampling, are provided in Appendix G.

Experimental Setup Following prior work Lee et al. (2024); Guo et al. (2026), we evaluate molecular optimization on docking score optimization across five protein targets: PARP1, FA7, JAK2, BRAF, and 5HT1B, while imposing feasibility constraints on drug-likeliness (QED; Bickerton et al. 2012) and synthetic accessibility (SA; Ertl et al. 2018), under an oracle budget of 3,000 evaluations, which is substantially more stringent than the 10,000 evaluations used by Gao et al. (2022). For each method, we report the mean and standard deviation calculated across 10 random seeds. We mark the best method **bold** and <u>underline</u> it, if the difference is statistically significant at a 95% confidence level. We use Hyformer Izdebski et al. (2026) as the backbone for JOINT SELF-IMPROVEMENT, pretrained on ZINC250K Sterling & Irwin (2015).

Metrics Following Lee et al. (2024); Guo et al. (2026), we measure Hit Ratio (%), defined as the fraction of sampled molecules with a docking score strictly better than the median of known actives, QED > 0.5, and SA < 5. Additionally, we measure IntDiv1, internal diversity, and Time (s), seconds needed to sample one molecule.

3.1 SAMPLE EFFICIENT OFFLINE MOLECULAR OPTIMIZATION

We first evaluate the ability of JOINT SELF-IMPROVEMENT to generate optimized molecules in the offline setting. We construct the offline dataset by randomly selecting 1,500 molecules from ZINC250K, across 10 random seeds, and computing their docking score, SA, and QED. We use the remaining 1,500 molecules for evaluation. We jointly finetune JOINT SELF-IMPROVEMENT using the offline dataset and subsequently sample optimized molecules for final evaluation. Experimental details are reported in the Appendix G.1.

Following Shin et al. (2025), we compare JOINT SELF-IMPROVEMENT against: REINVENT Loeffler et al. (2024), SATURN Guo et al. (2026), GeneticGFN Kim et al. (2024) and RaM Tan et al. (2024). We do not compare against MolStitch Shin et al. (2025) due to reproducibility limitations.

JOINT SELF-IMPROVEMENT consistently outperforms all baselines across all protein targets (Table 1). Policy gradient methods (REINVENT, SATURN) yield consistently low Hit Ratios across

all targets, highlighting their sample inefficiency in the offline setting. Notably, RaM represents a strong baseline, substantially improving upon earlier methods. Among all targets, JAK2 emerges as the most challenging, with consistently lower Hit Ratios across methods.

Table 1: Molecular optimization performance on docking score optimization in the offline setting.

	Target Protein, Hit Ratio (↑)				
Method	parp1	fa7	5ht1b	braf	jak2
REINVENT	2.686 (0.612)	0.813 (0.252)	19.653 (0.937)	0.693 (0.204)	4.020 (0.616)
SATURN	6.319 (0.845)	0.733 (0.270)	32.287 (1.687)	1.927 (0.353)	7.667 (1.279)
GeneticGFN	8.893 (1.532)	0.767 (0.252)	56.910 (2.730)	2.760 (0.907)	10.693 (1.927)
RaM	38.373 (5.196)	5.712 (5.382)	62.932 (9.472)	10.732 (4.771)	33.189 (7.850)
JOINT SELF-IMPROVEMENT (OURS)	**45.628 (11.653)**	**5.831 (2.040)**	**76.940 (4.442)**	**21.637 (4.607)**	**42.001 (4.344)**

3.2 SAMPLE EFFICIENT ONLINE MOLECULAR OPTIMIZATION

Next, we evaluate the ability of JOINT SELF-IMPROVEMENT to generate optimized molecules in the online setting. Since an offline dataset is unavailable for supervised fine-tuning, we compute global advantages using oracle evaluations rather than predictor scores. For optimization, we subsequently sample new molecules, use them to initialize or augment the online dataset, and retrain JOINT SELF-IMPROVEMENT on the online dataset, using the joint loss $\mathcal{L}_\lambda$ with $\lambda = 0$, i.e., in a generative manner. We repeat the above procedure until the oracle budget is exhausted. Experimental details are provided in Appendix G.2.

We compare JOINT SELF-IMPROVEMENT against the best-performing baselines reported by Guo et al. (2026): REINVENT Loeffler et al. (2024); JT-VAE Jin et al. (2018), a graph-based variational autoencoder; FREED-QS Yang et al. (2021), a fragment-based reinforcement learning approach with experience replay; MOOD Lee et al. (2023), a conditional diffusion-based sampling method; Augmented Memory Guo & Schwaller (2024), an extension of REINVENT that combines experience replay with SMILES augmentation; GEAM Lee et al. (2024), which integrates reinforcement learning with a genetic algorithm for goal-aware fragment-based optimization; and SATURN Guo et al. (2026), a REINVENT-style optimization framework with a Mamba backbone.

JOINT SELF-IMPROVEMENT consistently outperforms all baselines across all protein targets (Table 2). In particular, REINVENT achieves a low overall Hit Ratio, indicating limited ability to leverage iterative oracle supervision under strict evaluation budgets. In contrast, methods designed to improve sample efficiency, such as Augmented Memory and SATURN, yield substantial gains. Overall, these results show that JOINT SELF-IMPROVEMENT provides a strong inductive bias for efficient online molecular optimization.

Table 2: Molecular optimization performance on docking score optimization in the online setting.

	Target Protein, Hit Ratio (↑)				
Method	parp1	fa7	5ht1b	braf	jak2
REINVENT	4.693 (1.776)	1.967 (0.661)	26.047 (2.497)	2.207 (0.800)	5.667 (1.067)
JT-VAE	3.200 (0.348)	0.933 (0.152)	18.044 (0.747)	0.644 (0.157)	5.856 (0.204)
FREED-QS	5.960 (0.902)	1.687 (0.177)	23.140 (2.422)	3.880 (0.623)	7.653 (1.373)
MOOD	7.260 (0.764)	0.787 (0.128	21.427 (0.502	5.913 (0.311	10.367 (0.616)
Augmented Memory	16.966 (3.224)	2.637 (0.860)	52.016 (2.302)	8.307 (1.714)	21.548 (4.938)
GEAM	45.158 (2.408)	20.552 (2.357)	47.664 (1.198)	30.444 (1.610)	46.129 (2.073)
SATURN	57.981 (18.537)	14.527 (9.961)	68.185 (3.400)	38.999 (10.114)	60.827 (11.502)
JOINT SELF-IMPROVEMENT (OURS)	**69.403 (8.073)**	**28.667 (17.710)**	**84.910 (4.658)**	**41.740 (9.325)**	**62.317 (9.994)**

4 CONCLUSIONS

In this paper, we introduced JOINT SELF-IMPROVEMENT, a unified framework for sample-efficient molecular optimization. By combining a joint generative and surrogate model with a joint self-improving sampling scheme, JOINT SELF-IMPROVEMENT successfully alleviates problems typical of molecular optimization pipelines, including sample inefficiency and distribution shifts introduced

by decoupled surrogate and generative modeling, providing a principled alternative to RL-based approaches to sample-efficient GMO.

REFERENCES

Alekh Agarwal, Sham M Kakade, Jason D Lee, and Gaurav Mahajan. Optimality and approximation with policy gradient methods in markov decision processes. In Jacob Abernethy and Shivani Agarwal (eds.), *Proceedings of Thirty Third Conference on Learning Theory*, volume 125 of *Proceedings of Machine Learning Research*, pp. 64–66. PMLR, 09–12 Jul 2020.

Yoshua Bengio, Salem Lahlou, Tristan Deleu, Edward J Hu, Mo Tiwari, and Emmanuel Bengio. Gflownet foundations. *Journal of Machine Learning Research*, 24(210):1–55, 2023.

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

P Eckmann, K Sun, B Zhao, M Feng, MK Gilson, and RLIMO Yu. Limo: Latent inceptionism for targeted molecule generation, arxiv, 2022. *arXiv preprint arXiv:2206.09010*.

Peter Ertl, Richard Lewis, Eric Martin, and Valery Polyakov. In silico generation of novel, drug-like chemical matter using the LSTM neural network, January 2018.

Thomas Fischer, Silvia Gazzola, and Rainer Riedl. Approaching target selectivity by de novo drug design. *Expert Opinion on Drug Discovery*, 14(8):791–803, 2019. doi: 10.1080/17460441.2019.1615435. PMID: 31179763.

Wenhao Gao, Tianfan Fu, Jimeng Sun, and Connor Coley. Sample efficiency matters: A benchmark for practical molecular optimization. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 21342–21357. Curran Associates, Inc., 2022.

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

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

Will Grathwohl, Kuan-Chieh Wang, Jörn-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one, 2020.

Jeff Guo and Philippe Schwaller. Augmented memory: Sample-efficient generative molecular design with reinforcement learning. *JACS Au*, 4(6):2160–2172, 2024. doi: 10.1021/jacsau.4c00066.

Jeff Guo, Junwu Chen, Anthony GX Chen, and Philippe Schwaller. Sample-efficient generative molecular design using memory manipulation. *Nature Machine Intelligence*, 8:449–460, 2026. doi: 10.1038/s42256-026-01200-4.

James P Hughes, Stephen Rees, S Barrett Kalindjian, and Karen L Philpott. Principles of early drug discovery. *British journal of pharmacology*, 162(6):1239–1249, 2011.

Adam Izdebski, Jan Olszewski, Pankhil Gawade, Krzysztof Koras, Serra Korkmaz, Valentin Rauscher, Jakub M. Tomczak, and Ewa Szczurek. Synergistic benefits of joint molecule generation and property prediction. *Transactions on Machine Learning Research*, 2026. URL `https://openreview.net/forum?id=jnzCOLyGOA`. J2C Certification.

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.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pp. 2323–2332. PMLR, 2018.

Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. Score-based generative modeling of graphs via the system of stochastic differential equations. In *International conference on machine learning*, pp. 10362–10383. PMLR, 2022.

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

Minsu Kim, Federico Berto, Sungsoo Ahn, and Jinkyoo Park. Bootstrapped training of score-conditioned generator for offline design of biological sequences. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pp. 67643–67661, 2023.

Minsu Kim, Jiayao Gu, Ye Yuan, Taeyoung Yun, Zixuan Liu, Yoshua Bengio, and Can Chen. Offline model-based optimization: Comprehensive review, 2025. URL https://arxiv.org/abs/2503.17286.

Wouter Kool, Herke Van Hoof, and Max Welling. Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement. In *International conference on machine learning*, pp. 3499–3508. PMLR, 2019.

Seul Lee, Jaehyeong Jo, and Sung Ju Hwang. Exploring chemical space with score-based out-of-distribution generation. In *International Conference on Machine Learning*, pp. 18872–18892. PMLR, 2023.

Seul Lee, Seanie Lee, Kenji Kawaguchi, and Sung Ju Hwang. Drug discovery with dynamic goal-aware fragments. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 26731–26751. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/lee24o.html.

Hannes H Loeffler, Jiazhen He, Alessandro Tibo, Jon Paul Janet, Alexey Voronov, Lewis H Mervin, and Ola Engkvist. Reinvent 4: modern ai–driven generative molecule design. *Journal of Cheminformatics*, 16(1):20, 2024.

Youzhi Luo, Keqiang Yan, and Shuiwang Ji. Graphdf: A discrete flow model for molecular graph generation. In *International conference on machine learning*, pp. 7192–7203. PMLR, 2021.

Natalie Maus, Haydn Jones, Juston Moore, Matt J Kusner, John Bradshaw, and Jacob Gardner. Local latent space bayesian optimization over structured inputs. *Advances in neural information processing systems*, 35:34505–34518, 2022.

Kevin Murphy. Reinforcement learning: An overview, 2025. URL https://arxiv.org/abs/2412.05265.

Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Hybrid models with deep and invertible features. In *International Conference on Machine Learning*, pp. 4723–4732. PMLR, 2019.

Rıza Özçelik, Helena Brinkmann, Emanuele Criscuolo, and Francesca Grisoni. Generative deep learning for de novo drug design — a chemical space odyssey. *Journal of Chemical Information and Modeling*, 65(14):7352–7372, 2025. doi: 10.1021/acs.jcim.5c00641. URL https://doi.org/10.1021/acs.jcim.5c00641. PMID: 40632942.

Jonathan Pirnay and Dominik G. Grimm. Self-improvement for neural combinatorial optimization: Sample without replacement, but improvement. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=agT8ojoH0X. Featured Certification.

Jonathan Pirnay, Jan G Rittig, Alexander B Wolf, Martin Grohe, Jakob Burger, Alexander Mitsos, and Dominik G Grimm. Graphxform: graph transformer for computer-aided molecular design. *Digital Discovery*, 4(4):1052–1065, 2025.

Han Qi, Yi Su, Aviral Kumar, and Sergey Levine. Data-driven offline decision-making via invariant representation learning. In *Advances in Neural Information Processing Systems*, 2022.

Mayk Caldas Ramos, Christopher J. Collison, and Andrew D. White. A review of large language models and autonomous agents in chemistry. *Chem. Sci.*, 16:2514–2572, 2025. doi: 10.1039/D4SC03921A.

Desik Rengarajan, Gargi Vaidya, Akshay Sarvesh, Dileep Kalathil, and Srinivas Shakkottai. Reinforcement learning with sparse rewards using guidance from offline demonstration. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=YJ1WzgMVsMt.

Chence Shi, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. Graphaf: a flow-based autoregressive model for molecular graph generation. *arXiv preprint arXiv:2001.09382*, 2020.

Kensen Shi, David Bieber, and Charles Sutton. Incremental sampling without replacement for sequence models, 2021. URL https://arxiv.org/abs/2002.09067.

Dong-Hee Shin, Young-Han Son, Hyun Jung Lee, Deok-Joong Lee, and Tae-Eui Kam. Offline model-based optimization for real-world molecular discovery. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=erbRHt0XgI.

Yuda Song, Yifei Zhou, Ayush Sekhari, Drew Bagnell, Akshay Krishnamurthy, and Wen Sun. Hybrid RL: Using both offline and online data can make RL efficient. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=yyBis80iUuU.

Bhuvanesh Sridharan, Manan Goel, and U Deva Priyakumar. Modern machine learning for tackling inverse problems in chemistry: molecular design to realization. *Chemical communications*, 58 (35):5316–5331, 2022.

Teague Sterling and John J Irwin. Zinc 15–ligand discovery for everyone. *Journal of chemical information and modeling*, 55(11):2324–2337, 2015.

Rong-Xi Tan, Ke Xue, Shen-Huan Lyu, Haopu Shang, Yao Wang, Yaoyuan Wang, Sheng Fu, and Chao Qian. Offline model-based optimization by learning to rank. *arXiv preprint arXiv:2410.11502*, 2024.

Brandon Trabucco, Aviral Kumar, Xinyang Geng, and Sergey Levine. Conservative objective models for effective offline model-based optimization. In *Proceedings of the International Conference on Machine Learning*, pp. 10358–10368. PMLR, 2021.

Austin Tripp, Erik Daxberger, and José Miguel Hernández-Lobato. Sample-efficient optimization in the latent space of deep generative models via weighted retraining. *Advances in Neural Information Processing Systems*, 33:11259–11272, 2020.

Gautham Vasan, Yan Wang, Fahim Shahriar, James Bergstra, Martin Jagersand, and A. Rupam Mahmood. Revisiting sparse rewards for goal-reaching reinforcement learning, 2024. URL https://arxiv.org/abs/2407.00324.

Soojung Yang, Doyeong Hwang, Seul Lee, Seongok Ryu, and Sung Ju Hwang. Hit and lead discovery with explorative rl and fragment-based molecule generation. *Advances in Neural Information Processing Systems*, 34:7924–7936, 2021.

Kang Zhang, Xin Yang, Yifei Wang, Yunfang Yu, Niu Huang, Gen Li, Xiaokun Li, Joseph C Wu, and Shengyong Yang. Artificial intelligence in drug development. *Nature medicine*, 31(1):45–59, 2025.

Tingting Zhao, Hirotaka Hachiya, Gang Niu, and Masashi Sugiyama. Analysis and improvement of policy gradient estimation. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

A Impact Statement

The goal of this work is to advance deep generative modeling for molecular design, with a focus on accelerating drug discovery. While generative models could in principle be misused to design harmful or toxic compounds, our intention is to develop methods that support the discovery of safe and effective therapeutics. We emphasize applications aligned with medicinal chemistry and responsible research practices, and do not provide tools or guidance for harmful use cases.

B Limitations & Future Work

In challenging tasks, inaccuracy of the surrogate component may limit optimization performance. While potential inaccuracies do not accumulate, as sampling is done at inference-time, future work should focus on developing robust methods for additionally improving the predictive component of the joint model. In addition, incorporating uncertainty estimates into joint self-improving sampling could further improve robustness and exploration. Despite these limitations, already now Joint Self-Improvement outperforms state-of-the-art approaches in sample-efficient molecular optimization, across both offline and online settings, with ablations highlighting the specific contributions of joint learning and joint self-improving sampling.

C Mathematical Notation

We follow the mathematical notation of Goodfellow et al. (2016).

Symbol	Meaning
$[N]$	Set of integers $1, \ldots, N$
$\mathbf{a}$	Vector (column-vector)
$\mathbf{a}^{(i)}$	Vector indexed for some purpose
$\mathbf{a}_{i:j}$	Vector with values from the i-th to j-th (inclusive) entry of vector $\mathbf{a}$
a_i	The i-th entry of vector $\mathbf{a}$
a	Scalar
$\mathcal{X}$	Input space, i.e. the space of all possible inputs
$\mathcal{Y}$	Target space i.e. the space of all possible property values
$\mathcal{D}$	Dataset
$p_\theta(\mathbf{x}, y)$	Joint model parametrized with parameters $\theta \in \Theta$
$p_\theta(y \mid \mathbf{x})$	Predictive model parametrized with parameters $\theta \in \Theta$
$p_\theta(\mathbf{x})$	Generative model parametrized with parameters $\theta \in \Theta$

D Related Work

GMO methods can be broadly divided into approaches that improve the generative model used to generate optimized molecules and those that improve the surrogate model used to guide optimization Kim et al. (2025).

Generative Modeling for GMO Early approaches to generative molecular optimization rely on genetic algorithms Jensen (2019) or latent space optimization, where optimization is performed via gradient ascent or Bayesian optimization in a continuous latent representation of molecules Gómez-Bombarelli et al. (2018); Jin et al. (2018); Tripp et al. (2020); Maus et al. (2022); Eckmann et al.. More recently, RL and, in particular, policy gradient methods are prevalent Shi et al. (2020); Luo et al. (2021); Yang et al. (2021); Jo et al. (2022). Following the growing emphasis on sample efficiency Gao et al. (2022), subsequent work improves upon sample efficiency of RL by reusing off-policy data Guo & Schwaller (2024), adopting Mamba-based backbones Guo et al. (2026), or combining RL with genetic algorithms Lee et al. (2024). Similarly, Kim et al. (2024) incorporates domain-specific genetic operators into GFlowNets Bengio et al. (2023). Beyond RL, Lee et al. (2023) perform conditional generation by leveraging gradients from a surrogate to guide a reverse diffusion process. Finally, Pirnay et al. (2025) propose a decoder-only graph transformer trained with a combination of cross-entropy and self-improvement to iteratively bias generation.

Surrogate Modeling for GMO A standard approach to surrogate modeling learns a predictor that approximates the objective function via mean squared error. Subsequent work improves upon this through adversarial training to obtain conservative predictions Trabucco et al. (2021), invariant representation learning to enhance robustness under distribution shift Qi et al. (2022), bootstrapping strategies that iteratively augment the training set with self-generated samples Kim et al. (2023) and learning-to-rank formulations that model relative orderings instead of absolute objective values Tan et al. (2024); Shin et al. (2025).

To our knowledge, JOINT SELF-IMPROVEMENT is the first GMO framework to address distribution shift by aligning the generative model and surrogate while also enabling efficient sampling. While enjoying the synergistic benefits of joint modeling Izdebski et al. (2026), we strongly extend this paradigm with a self-improving joint sampling scheme that replaces rejection-based sampling. Unlike prior self-improvement approaches Pirnay et al. (2025), which act solely on the generative model, we leverage the predictive component to directly guide generation.

E SAMPLING ALGORITHM OF JOINT SELF-IMPROVEMENT

Algorithm 1 outlines the sampling process employed by JOINT SELF-IMPROVEMENT.

Algorithm 1 Sampling with $\mathrm{JSI}(p_\theta, N_{\text{rounds}}, K, \sigma)$

Require: Joint model $p_\theta(\mathbf{x}, y)$. Number of rounds N_{rounds}.
Beam width K. Step size σ.
Ensure: Sample $\mathbf{x}$ with highest $p_\theta(y \mid \mathbf{x})$.

$\mathcal{D}_{\text{sampled}} \leftarrow \emptyset$
for $n = 1, \ldots, N_{\text{rounds}}$ **do**
  $\{\mathbf{x}^{(k)}\}_{k=1}^{K} \leftarrow \mathrm{SBS}(p_\theta, K)$
  $\mathcal{D}_{\text{sampled}} \leftarrow \mathcal{D}_{\text{sampled}} \cup \{\mathbf{x}^{(k)}\}_{k=1}^{K}$
  $\mu \leftarrow \mu\big(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(K)}\big)$
  $p_{\theta'} \leftarrow p_\theta$
  for $k = 1, \ldots, K$ **do**
    for $t = 1, \ldots, T$ **do**
      $A_t^{(k)} \leftarrow \sum_{\mathbf{x} \in S(\mathbf{x}_{1:t}^{(k)})} \Big(p_\theta(y \mid \mathbf{x}) - \mu\Big)$
      $R_t^{(k)} \leftarrow \sum_{\mathbf{x} \in S(\mathbf{x}_{1:t}^{(k)})} p_\theta(\mathbf{x} \mid \mathbf{x}_{1:t-1}^{(k)})$
      $\operatorname{logit} p_{\theta'}\big(x_t^{(k)} \mid \mathbf{x}_{1:t-1}^{(k)}\big) \leftarrow \log\big(p_\theta(x_t^{(k)} \mid \mathbf{x}_{1:t-1}^{(k)}) - R_t^{(k)}\big) + \sigma \cdot A_t^{(k)}$ (Eq. 7)
    end for
  end for
  $p_\theta \leftarrow \mathrm{Softmax}\big(\operatorname{logit} p_{\theta'}\big)$
end for
return $\arg\max_{\mathbf{x} \in \mathcal{D}_{\text{sampled}}} p_\theta(y \mid \mathbf{x})$

F ADDITIONAL RESULTS

F.1 CONNECTION TO THE REINVENT ALGORITHM

The gradient of the REINVENT algorithm coincides with Eq. 4 up to a constant $c \in \mathbb{R}$, as

$$
\begin{aligned}
\mathcal{L}_{\text{REINVENT}}(\theta) &= \Big(\log p_{\text{aug}}(\mathbf{x}) - \log p_\theta(\mathbf{x})\Big)^2 \\
&= \Big(\log p_{\text{prior}}(\mathbf{x}) - \sigma f(\mathbf{x}) - \log p_\theta(\mathbf{x})\Big)^2 \\
&= \Big(\sigma f(\mathbf{x}) - \log \frac{p_\theta(\mathbf{x})}{p_{\text{prior}}(\mathbf{x})}\Big)^2 \\
&= \sigma^2 \Big(f(\mathbf{x}) - \frac{1}{\sigma} \log \frac{p_\theta(\mathbf{x})}{p_{\text{prior}}(\mathbf{x})}\Big)^2
\end{aligned}
\tag{8}
$$

$$\nabla_\theta \mathcal{L}_{\text{REINVENT}}(\theta) = \nabla_\theta \sigma^2 \Big(f(\mathbf{x}) - \frac{1}{\sigma} \log \frac{p_\theta(\mathbf{x})}{p_{\text{prior}}(\mathbf{x})} \Big)^2 \tag{9}$$

$$= c \Big(f(\mathbf{x}) - \beta \log \frac{p_\theta(\mathbf{x})}{p_{\text{prior}}(\mathbf{x})} \Big) \nabla_\theta \log p_\theta(\mathbf{x}). \tag{10}$$

G EXPERIMENTAL DETAILS

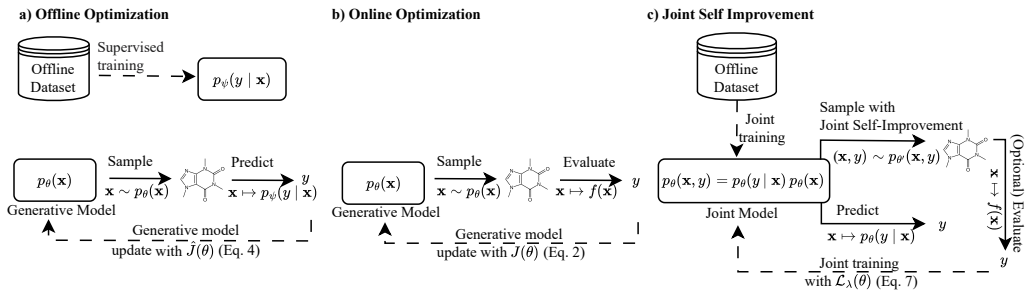

Figure 3: Offline and online GMO as compared to JOINT SELF-IMPROVEMENT. (a) In the offline setting, objective f is not available, and a surrogate predicts its values; predictor and generative model training are typically decoupled. (b) In the online setting, objective f is available and used for generative model updates. (c) JOINT SELF-IMPROVEMENT operates in both settings: training a joint model from offline data and sampling new molecules, or updating from newly generated objective evaluations.

We evaluate the optimization performance of our framework under strict sample-efficiency constraints. In particular, we do not benchmark on the setting of Gao et al. (2022), and instead adopt the more stringent protocols of Lee et al. (2024); Guo & Schwaller (2024); Guo et al. (2026); Shin et al. (2025), as the oracle budget of 10,000 evaluations proposed by Gao et al. (2022), especially in the online setting, exceeds the evaluation budgets typically available in real-world drug discovery workflows.

Moreover, we report Hit Ratio rather than multi-objective metrics such as hypervolume (HV) or R2 Shin et al. (2025). We formulate docking score optimization as a constrained optimization problem, consistent with prior work Lee et al. (2024); Guo & Schwaller (2024); Guo et al. (2026); Shin et al. (2025). The objective is to generate molecules that exceed a predefined docking threshold while satisfying chemical plausibility constraints, as quantified by SA and QED. In this formulation, satisfying the docking criterion has strict priority over improving auxiliary properties: once the docking threshold is met, further gains in SA or QED are secondary, whereas molecules that fail to meet the docking threshold are rejected irrespective of their auxiliary scores. Consequently, Pareto-based multi-objective metrics that reward smooth trade-offs between docking and auxiliary properties are not fully aligned with the feasibility-driven selection process encountered in drug discovery.

Hyformer Izdebski et al. (2026) To ensure comparability with baseline models, we pre-train HYFORMER on ZINC250K dataset, scaled to 28M parameters with 8 transformer layers and embedding dimensionality 512, using AdamW optimizer with cosine decay for a maximum of 20 epochs, with early stopping based on the validation set. We summarize hyperparameters used for pre-training in Table 3. In contrast to Izdebski et al. (2026), we pre-train without relying on precomputed molecular descriptors.

G.1 SAMPLE EFFICIENT OFFLINE MOLECULAR OPTIMIZATION

All results in Table 1 are ours.

Table 3: Hyperparameter settings used for pretraining and finetuning of HYFORMER.

HYPERPARAMETER	PRETRAINING	FINETUNING
BATCH SIZE	512	32
LEARNING RATE	6E-4	1E-4
WEIGHT DECAY	1E-1	1E-1
(β_1, β_2)	(0.9, 0.95)	(0.9, 0.95)
GRADIENT CLIP	1.0	1.0
MINIMUM LEARNING RATE	1E-6	1E-6
WARMUP RATIO	0.05	0.1
DTYPE	BFLOAT16	FLOAT32
$(p_{[\text{LM}]}, p_{[\text{MLM}]})$	(0.9, 0.1)	(0.25, 0.75)

REINVENT Loeffler et al. (2024) We implement a surrogate-guided REINVENT optimization pipeline with an integrated Hyformer predictor. First Hyformer is fine-tuned on the offline training split, and the resulting predictor is used to score generated molecules throughout the optimization. We initialize a REINVENT agent from a fixed prior checkpoint pre-trained on the ZINC250k dataset and then run iterative policy updates: at each step, the agent samples a batch of generated molecules, Hyformer predicts the reward. The agent is then updated by minimizing the REINVENT squared error between the agent's log-likelihood and an augmented log-likelihood target, with experience replay. All the hyperparameters for the experiment are taken from Shin et al. (2025).

SATURN Guo et al. (2026) We implement a surrogate-guided SATURN optimization pipeline with an integrated Hyformer predictor, trained using a REINVENT-style reinforcement learning objective with augmented memory replay and SMILES-based data augmentation. Hyformer is first fine-tuned on the offline training dataset and then used as a surrogate oracle during optimization: at each iteration, molecules are sampled from the current policy, scored by Hyformer, and the policy is updated based on the predicted rewards. The SATURN agent is initialized from a prior pretrained on ZINC250K, using a publicly available checkpoint from the SATURN repository `https://github.com/schwallergroup/saturn`. Following Guo et al. (2026), we use a learning rate of 1×10^{-4}, an RL step-size parameter σ=128.0, 10 SMILES augmentation rounds, a replay buffer of size 100, and a batch size of 64.

GeneticGFN We first fine-tune HYFORMER on the offline dataset using hyperparameters summarized in Table 3 and employ the resulting model as a surrogate predictor throughout optimization. Following Shin et al. (2025), we perform 8 augmentation rounds. In each round, we sample 200 molecules from the generative policy and score them using the surrogate. Across all rounds, the top 64 highest-scoring molecules are used to initialize a genetic algorithm, which is run for 2 generations with a mutation rate of 0.01 and an offspring size of 4. All generated offspring are evaluated by the surrogate and added to a rank-based reweighed experience buffer with a maximum capacity of 512. At the end of each augmentation round the policy is optimized using the trajectory balance loss with a Kullback-Leibler (KL) divergence penalty. We use a KL coefficient of 1×10^{-2} and a learning rate of 5×10^{-4} for both the policy parameters and the log-partition constant logZ, which is initialized to 1×10^{-3}.

RaM We fine-tune HYFORMER on the offline dataset to serve as a learning-to-rank (LTR) surrogate model using the RankCosine loss. We use a batch size of 32 and a learning rate of 1×10^{-4}, with all other hyperparameters set to their default values (Table 3). The sequences with the highest docking scores in the offline dataset are selected as initial candidates for gradient-ascent optimization in RaM. The optimization is performed using stochastic gradient ascent for 500 steps with a step size of 0.8, where the target score is defined as the mean of the normalized LTR objectives, and gradients are applied at the token-embedding level. The optimized embeddings are mapped back to discrete tokens using cosine similarity. The step size and initial candidate selection strategy are determined via hyperparameter optimization. At each iteration, generated sequences are aggregated and filtered to retain only valid and unique sequences that are also not present in the offline dataset. The top-ranked sequences according to the LTR surrogate are then selected for final evaluation.

JOINT SELF-IMPROVEMENT We jointly fine-tune HYFORMER on the offline dataset with hyperparameters listed in Table 3. Next, we define the advantage as the mean of the predicted

Table 4: Best hyperparameters for Joint Self-Improvement in the offline optimization experiment.

	Target Protein				
Hyperparameter	parp1	fa7	5ht1b	braf	jak2
Beam width K	128	128	16	128	128
Number of rounds N_{rounds}	10	10	8	8	8
Step size σ	0.25	0.5	1.0	1.0	0.5

Table 5: Best hyperparameters for Joint Self-Improvement in the online optimization experiment.

	Target Protein				
Hyperparameter	parp1	fa7	5ht1b	braf	jak2
Beam width K	16	16	32	16	16
Number of rounds N_{rounds}	10	14	10	12	10
Step size σ	1.5	1.5	0.75	1.75	1.5

DS, SA and QED values. Crucially, all predicted scores are Z-score normalized with the mean and standard deviation of each target calculated using the offline dataset. Finally, we sample the required number of molecules for evaluation, with temperature 0.9 and perform a grid search over beam width $K \in [16, 32, 64, 128]$, number of rounds $N_{\text{rounds}} \in [4, 8, 10, 12]$ and step size $\sigma \in [0.25, 0.5, 0.75, 1., 1.25, 1.5]$. Best hyperparameters across all protein targets are summarized in Table 4. To ensure numerical correctness and efficiency of our sampling algorithm, in particular, numerical stability in Eq. 7, we follow the implementation of Shi et al. (2021) of an augmented trie structure.

G.2 Sample-Efficient Online Molecular Optimization

All results in Table 2 are from Guo et al. (2026), except for Joint Self-Improvement. The reward function is defined following prior work Guo et al. (2026).

Joint Self-Improvement We use HyFormer as the generative model, pretrained on ZINC250K, to sample molecules using a self-improving sampling scheme. Oracle queries are employed to compute the reward. The generator is then fine-tuned using the top rewarded molecules. Following Guo et al. (2026), 5 rounds of SMILES augmentation is applied prior to fine-tuning.

For evaluation, we sample the required number of molecules at a temperature of 0.9. We perform a grid search over the beam width $K \in \{16, 32, 64, 128\}$, the number of rounds $N_{\text{rounds}} \in \{4, 8, 10, 12, 14\}$, and the step size $\sigma \in \{0.25, 0.5, 0.75, 1.0, 1.25, 1.5, 1.75\}$. Best hyperparameters across all protein targets are summarized in Table 5.

G.3 Ablations

Table 6: Ablations for molecular optimization performance on docking score optimization in the offline setting.

	Target Protein, Hit Ratio (↑)						
Method	parp1	fa7	5ht1b	braf	jak2	IntDiv1 (↑)	Time (↓)
Joint Self-Improvement	50.313 (13.437)	7.500 (4.529)	79.375 (5.696)	27.187 (9.176)	46.719 (6.524)	76.253 (1.450)	22.68 (1.09)
w/o joint modeling	46.111 (12.955)	4.670 (3.966)	72.140 (8.142)	21.361 (3.774)	45.240 (6.113)	74.439 (2.041)	-
w/o self-improving sampling	2.901 (2.386)	0.675 (1.192)	25.736 (4.703)	0.637 (1.112)	4.640 (2.658)	87.413 (0.596)	51.88 (0.98)

To quantify the individual contributions of joint learning and the joint self-improving sampling scheme, we ablate Joint Self-Improvement in two variants: (i) w/o joint modeling, which decouples generation and prediction by fine-tuning a separate predictor on the offline dataset while using the pretrained generator for sampling, and (ii) w/o self-improving sampling, which replaces our sampling procedure with Best-of-N rejection sampling Izdebski et al. (2026). For computational reasons, we evaluate 64 newly sampled candidates per protein target. For IntDiv1 and Time in Table 6, we report the mean over 10 random seeds aggregated across all protein targets. All experiments are conducted in the offline setting, with further details provided in Appendix G.3.

JOINT SELF-IMPROVEMENT consistently outperforms its ablated variants, highlighting the benefits of joint modeling and joint self-improving sampling. In particular, W/O SELF-IMPROVING SAMPLING fails to generate optimized molecules, as reflected by the near-zero Hit Ratio across all protein targets. Likewise, W/O JOINT MODELING achieves consistently lower performance, and we expect this gap to widen if disjoint architectures were used or if the generative component were retrained independently, as in RL-based approaches. Notably, JOINT SELF-IMPROVEMENT maintains strong performance under the stringent setting of only 64 molecule evaluations, while sampling nearly twice as fast as the rejection-sampling-based version. Additionally, we qualitatively assess the alignment of the optimized molecules with the joint model predictions in Figure 4.

All results in Table 6 are ours.

JOINT SELF-IMPROVEMENT W/O JOINT MODELING We fine-tune the predictive part of HYFORMER, on the offline dataset with hyperparameters listed in Table 3, except for altering the task weights to include only the predictive task. For sampling, we keep a separate, pre-trained copy of HYFORMER in memory and use it to optimize molecules, using the best hyperparameters from G.2 and guided by the fine-tuned predictive copy.

JOINT SELF-IMPROVEMENT W/O SELF-IMPROVING SAMPLING We replace the joint self-improving sampling scheme with Best-of-N sampling Izdebski et al. (2026), where for every molecule we first sample N-many molecules, using the generative part of the model, and choose the best performing molecule with the predictor. We set N to 256, in order to keep Time comparable between the methods.

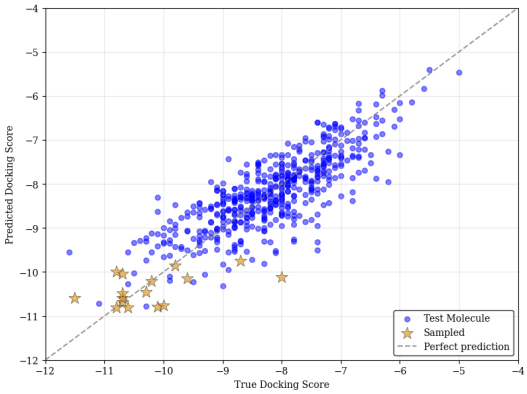

Figure 4: True versus predicted docking scores against BRAF protein target of test (blue) and generated (yellow) molecules. We use JOINT SELF-IMPROVEMENT for both prediction and generation.

H QUALITATIVE ANALYSIS

To qualitatively validate the performance of JOINT SELF-IMPROVEMENT in the offline setting, we additionally show binding poses of the best generated molecules, i.e., hit molecules with the lowest docking score, in Figure 5. Binding poses are visualized using PyMOL.

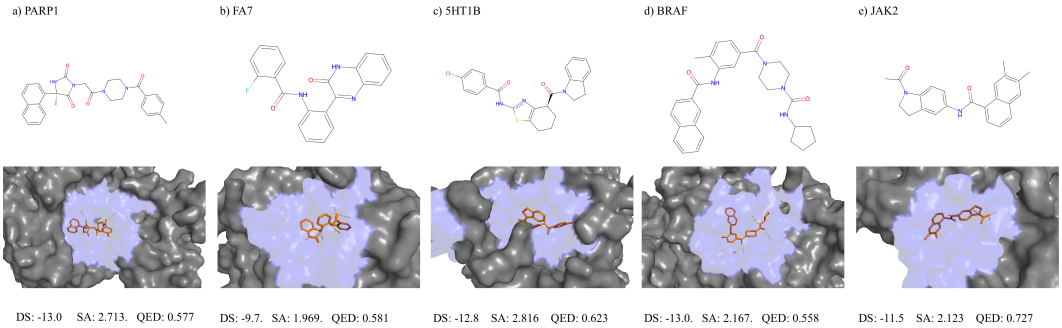

Figure 5: Binding poses of best molecules generated by JOINT SELF-IMPROVEMENT in the offline optimization setting, across all target proteins. Yellow lines indicate hydrogen-bond interactions between the ligands and the target proteins.