Sample Efficient Generative Molecular Optimization with Joint Self-Improvement

Serra Korkmaz[1,2*] **Adam Izdebski**[1,4,7*] **Jonathan Pirnay**[3]
Rasmus Møller-Larsen[1,4,7] **Michal Kmicikiewicz**[1,4,7] **Pankhil Gawade**[1,4,7]
Maksim Pavlov[1] **Dominik G. Grimm**[3,4] **Ewa Szczurek**[1,5,6,7]
[1]Institute of AI for Health, Computational Health Center, Helmholtz Munich
[2]Technical University of Munich, TUM School of Natural Sciences
[3]Technical University of Munich, TUM Campus Straubing for Biotechnology and Sustainability
[4]Technical University of Munich, TUM School of Computation, Information and Technology
[5]Faculty of Mathematics, Informatics and Mechanics, University of Warsaw
[6]Faculty of Mathematics, Informatics and Statistics, Ludwig-Maximilians-Universität München
[7] Munich Center for Machine Learning
{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, evaluating candidate molecules is expensive, so sample efficiency, i.e., generating optimized molecules within a fixed evaluation budget, is essential. Moreover, in the offline setting, where a surrogate model approximates candidate evaluation, sample inefficiency compounds with surrogate error, further limiting performance. To address these challenges, we introduce JOINT SELF-IMPROVEMENT (JSI), a framework for sample-efficient online and offline molecular optimization. JSI combines **(i) a joint generative-predictive model** trained via a joint likelihood-based loss, which aligns molecule generation and surrogate modeling, and **(ii) inference-time self-improvement**, which iteratively biases the generative component of the joint model toward higher-scoring molecules, using direct objective function evaluations in the online setting and the predictive component in the offline setting. Experiments across docking score optimization in both online and offline settings, as well as antimicrobial peptide optimization, demonstrate that JSI outperforms state-of-the-art methods under limited evaluation budgets. Code is available at `https://github.com/szczurek-lab/jsi`.

1 Introduction

Generative molecular optimization (GMO) is a cornerstone of modern drug discovery, with the aim of generating molecules that improve upon existing compounds [12, 67, 59, 77]. Yet, such molecules are exceedingly rare, and evaluating newly generated candidates remains prohibitively expensive, as it typically requires wet-lab experimentation [23, 52], making *sample efficiency* essential [18].

Nevertheless, state-of-the-art GMO approaches typically rely on reinforcement learning (RL) [45, 18, 19, 65], in which a pre-trained chemical language model is treated as a policy and fine-tuned to generate high-scoring molecules. Since such molecules are exceedingly rare, sampling predominantly yields low-scoring candidates [24], rendering RL-based methods sample-inefficient [53, 78, 1, 61, 75].

*Equal contribution.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

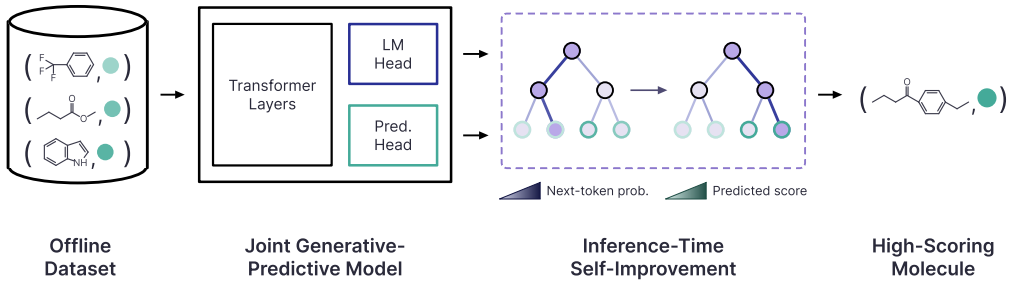

Figure 1: Overview of JSI. In the offline setting, a joint generative-predictive model, with a language modeling (LM) head for next-token probability prediction and a predictive head for property prediction, is fine-tuned on an offline dataset of molecules and their property scores, and the model self-improves by iteratively biasing next-token probabilities (blue) using predicted property scores (green), steering generation towards high-scoring molecules at inference time. In the online setting, ground-truth oracle evaluations are used in place of the predictive head.

Moreover, in settings representative of real-world drug discovery, where wet-lab experiments are required for candidate evaluation, it is often infeasible to evaluate molecules during optimization, an assumption made by *online* molecular optimization [18, 19]. In contrast, *offline* molecular optimization assumes that candidates cannot be evaluated directly, relying instead on an offline dataset of pre-evaluated molecules to guide optimization [65, 34]. The offline dataset is typically used to fit a *surrogate*, which approximates candidate evaluation and serves as a plug-in replacement for the objective function in RL [65]. However, estimating the objective beyond the offline dataset is inherently challenging, and optimizing against an imperfect surrogate risks *overoptimization*, where the model exploits inaccuracies in the surrogate rather than improving the objective [42, 13, 66, 34].

To address these limitations, we introduce JOINT SELF-IMPROVEMENT (JSI), a framework for sample-efficient online and offline GMO (Figure 1). JSI aligns molecule generation and surrogate modeling within a **joint generative-predictive model** with shared parametrization, trained using the joint likelihood-based loss [25]. At inference time, JSI generates optimized molecules by iteratively biasing the generative model's next-token probabilities towards high-scoring molecules, i.e., **inference-time self-improvement** [54]. In the online setting, we generate molecules by biasing the generative model with ground-truth oracle evaluations and subsequently fine-tune the generative part of the model on the resulting samples. In the offline setting, where a surrogate replaces the ground-truth oracle, and only an offline dataset is available to guide optimization, we use the joint loss to fine-tune a pre-trained model and generate molecules by using the generative part of the joint model with the predictive part. We evaluate our framework on established benchmarks for sample-efficient molecular and antimicrobial peptide (AMP) optimization [19, 65, 25]. Our contributions are:

- We align molecule generation and surrogate modeling within a **joint generative-predictive model** with shared parametrization and trained using a joint likelihood-based loss.
- We propose to sample from the joint model with **inference-time self-improvement**, iteratively biasing the generative component with oracle evaluations in the online setting, and with the predictive component in the offline setting.
- We evaluate JSI on sample-efficient molecular optimization benchmarks, in both online and offline settings, as well as AMP optimization, achieving state-of-the-art results.

2 Related Work

GMO 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 [34].

Generative Modeling for GMO Early approaches to generative molecular optimization rely on genetic algorithms [26] or latent space optimization, where optimization is performed via gradient ascent or Bayesian optimization in a continuous latent representation of molecules [15, 28, 29, 74, 46, 10, 41]. More recently, RL methods are prevalent, including value-based approaches that optimize

docking scores directly [27] and, in particular, policy gradient methods [63, 76, 30]. Following the growing emphasis on sample efficiency [14], subsequent work improves upon sample efficiency of RL by reusing off-policy data [18], adopting Mamba-based backbones [19], or combining RL with genetic algorithms [40]. Similarly, Kim et al. [31] incorporate domain-specific genetic operators into GFlowNets [5]. Beyond RL, Lee et al. [39] perform conditional generation by leveraging gradients from a surrogate to guide a reverse diffusion process. Finally, Pirnay et al. [55] applies self-improvement, derived from round-wise provable policy improvement [54], to molecular graph construction in the online setting, and without evaluation in sample-efficient settings.

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 [73], invariant representation learning to enhance robustness under distribution shift [57], bootstrapping strategies that iteratively augment the training set with self-generated samples [33], and learning-to-rank formulations that model relative orderings instead of absolute objective values [70, 65].

Joint Modeling for GMO Izdebski et al. [25] jointly model molecule generation and property prediction, showing that shared parametrization aligns the generative and predictive components, enabling property-conditioned molecule generation. However, they rely on rejection sampling to draw high-quality candidates, limiting exploration to high-density regions of the generative model. Pirnay et al. [55] apply self-improvement to molecular graph construction, but act solely on the generative model without surrogate integration. JSI combines both paradigms by leveraging the predictive component of the joint model to directly guide generation via self-improvement, enabling efficient sampling beyond the generative model's support in both online and offline settings.

3 Background

Molecular Optimization The goal of *molecular optimization* is to propose a *molecule* $\mathbf{x}^* \in \mathcal{X}$ that maximizes an *objective function* (*oracle*) $f : \mathcal{X} \to [0, f_{\max}]^M$, where $f_{\max} \in \mathbb{R}_+$. When $M = 1$, molecular optimization reduces to a single-objective optimization problem

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

In practice, the chemical space $\mathcal{X}$ is further restricted by feasibility criteria, such as synthetic accessibility or drug-likeness. As evaluating the objective f is costly, e.g., a wet-lab assay or docking simulation, molecular optimization prioritizes *sample efficiency*, i.e., minimizing the number of objective function evaluations (oracle calls) needed to propose an optimized molecule $\mathbf{x}^*$.

Molecular optimization additionally depends on whether the objective f can be evaluated during optimization. In the *online* setting, the optimization process iteratively queries f and adapts based on the feedback. However, as f is costly to evaluate, the online setting is often infeasible in practice. In contrast, the *offline* setting [34] assumes access only to an *offline dataset* $\mathcal{D}_{\text{offline}} = \{(\mathbf{x}^{(n)}, f(\mathbf{x}^{(n)}))\}_{n=1}^{N}$.

Reinforcement Learning-based Molecular Optimization State-of-the-art molecular optimization methods [18, 19, 65] predominantly rely on a variant of reinforcement learning, in which a generative model $p_\theta(\mathbf{x})$ is fine-tuned via the *KL-regularized objective*

$$J_{f,\beta}(\theta) = \mathbb{E}_{p_\theta(\mathbf{x})}[f(\mathbf{x})] - \beta \, \mathrm{KL}(p_\theta(\mathbf{x}) \,\|\, p_{\text{ref}}(\mathbf{x})), \tag{2}$$

where $\beta > 0$ controls the drift from a *reference model* $p_{\text{ref}}(\mathbf{x})$, typically a pre-trained chemical language model. More generally, this is an instance of *reinforcement learning fine-tuning* (RLFT) [48].

Despite widespread adoption, optimizing the KL-regularized objective $J_{f,\beta}$ is inherently sample-inefficient. In regimes typical of molecular optimization, molecules with high values of the objective f are rare and lie in low-density regions of $p_{\text{ref}}(\mathbf{x})$. In consequence, unconditional sampling $\mathbf{x} \sim p_\theta(\mathbf{x})$ predominantly yields molecules with low values of f when $p_\theta(\mathbf{x})$ is close to $p_{\text{ref}}(\mathbf{x})$ [24]. Yet, the Monte Carlo gradient estimator [71]

$$\nabla_\theta J_{f,\beta}(\theta) \approx \frac{1}{|B|} \sum_{\mathbf{x} \in B} \left(f(\mathbf{x}) - \beta \log \frac{p_\theta(\mathbf{x})}{p_{\text{ref}}(\mathbf{x})} \right) \nabla_\theta \log p_\theta(\mathbf{x}), \tag{3}$$

where $B = \{\mathbf{x}^{(b)}\}_{b=1}^{|B|} \sim p_\theta(\mathbf{x})$, requires the batch B to contain molecules with high values of f to provide a meaningful training signal, rendering fine-tuning sample-inefficient, especially early in training [53, 78, 1, 66, 75]. In particular, in Appendix C.3, we show that REINVENT-style algorithms [45] inherit these limitations.

Surrogate Modeling In the offline setting, where the objective f cannot be evaluated, a surrogate $\hat{f} : \mathcal{X} \to [0, f_{\max}]$ is trained on the offline dataset to approximate f and used as a plug-in replacement for the objective [34]. When combined with the KL-regularized objective, this amounts to optimizing $J_{\hat{f},\beta}$ in place of $J_{f,\beta}$ (Eq. (2)). However, as the surrogate $\hat{f}$ is an imperfect proxy for f, particularly outside the support of $\mathcal{D}_{\text{offline}}$, optimizing $J_{\hat{f},\beta}$ for too long, e.g., due to sample inefficiency of the underlying optimization algorithm, may cause the expected true objective $\mathbb{E}_{p_\theta(\mathbf{x})}[f(\mathbf{x})]$ to deteriorate even as $\mathbb{E}_{p_\theta(\mathbf{x})}[\hat{f}(\mathbf{x})]$ continues to improve, a phenomenon known as *overoptimization* [13].

4 Joint Self-Improvement

We introduce JSI, a unified framework for sample-efficient molecular optimization. First, we formulate molecular optimization as a conditional sampling problem. Next, we define a joint model $p_\theta(\mathbf{x}, y)$ that unifies the generative model $p_\theta(\mathbf{x})$ and the surrogate $p_\theta(y \mid \mathbf{x})$ within a shared parametrization θ. Finally, we propose to sample from the joint model $p_\theta(\mathbf{x}, y)$ by self-improving the joint model at inference time.

Molecular Optimization as Conditional Sampling Molecular optimization can be formulated as a conditional sampling problem. Indeed, maximizing the KL-regularized objective $J_{f,\beta}(\theta)$ in Eq. (2) admits a closed-form solution [35, 37, 58, 48], given by

$$p^*(\mathbf{x}; f, \beta) = Z_{f,\beta}^{-1}\, p_{\text{ref}}(\mathbf{x}) \exp\big(\beta^{-1} f(\mathbf{x})\big) \tag{4}$$

where $Z_{f,\beta} = \mathbb{E}_{p_{\text{ref}}(\mathbf{x})}[\exp\big(\beta^{-1} f(\mathbf{x})\big)]$ is the normalizing constant and $\beta > 0$ controls the drift from the reference model $p_{\text{ref}}(\mathbf{x})$. However, evaluating the objective f is expensive, and directly sampling from $p^*(\mathbf{x}; f, \beta)$ is infeasible, as the constant $Z_{f,\beta}$ is intractable.

Joint Generative-Predictive Model To approximate the objective f with a surrogate, we define a *joint generative-predictive model* [49, 17, 25]

$$p_\theta(\mathbf{x}, y) = p_\theta(y \mid \mathbf{x})\, p_\theta(\mathbf{x}), \tag{5}$$

where $p_\theta(y \mid \mathbf{x})$ is the surrogate and $p_\theta(\mathbf{x}) = \prod_{t=1}^{T} p_\theta(x_t \mid \mathbf{x}_{1:t-1})$ is an *autoregressive* generative model. We train the joint model via *joint loss*, i.e., the joint log-likelihood

$$\mathcal{L}_\lambda(\theta) = -\sum_{(\mathbf{x},y)\in\mathcal{D}} \Big((1-\lambda)\log p_\theta(\mathbf{x}) + \lambda \log p_\theta(y \mid \mathbf{x})\Big), \tag{6}$$

where $\lambda \in [0, 1]$ controls the relative strength of the generative and predictive counterparts. Note that when $\lambda = 0$ or $\lambda = 1$, the joint loss reduces to generative-only or predictive-only updates, respectively.

The advantages of training a joint model are two-fold. As opposed to optimizing the KL-regularized objective $J(\theta)$ in Eq. (2), the joint loss yields stable likelihood-based updates, as the gradients of $\mathcal{L}_\lambda$ do not depend on sampling optimized examples [25]. Additionally, in the offline setting, both the predictive and the generative components are trained jointly on the same offline dataset $\mathcal{D}_{\text{offline}}$, and share parameters θ, effectively aligning the generative distribution with the surrogate [25].

Inference-Time Self-Improvement To sample optimized molecules $\mathbf{x}^*$ from the joint model $p_\theta(\mathbf{x}, y)$ at *inference-time*, we propose to *self-improve* the generative component $p_\theta(\mathbf{x})$ of the joint model $p_\theta(\mathbf{x}, y)$, by perturbing its next-token probabilities with the predictive component $p_\theta(y \mid \mathbf{x})$, or directly with $f(\mathbf{x})$ in the online setting.

To achieve this, we define the *individual advantage* [54] of token $x_t \in \mathcal{V}$, given a prefix $\mathbf{x}_{1:t-1}$, as

$$\text{Adv}_\theta(x_t \mid \mathbf{x}_{1:t-1}) := \mathbb{E}_{p_\theta(\mathbf{x}_{t+1:T}|x_t,\mathbf{x}_{1:t-1})}\big[p_\theta(y \mid \mathbf{x})\big] - \mathbb{E}_{p_\theta(\mathbf{x}_{t:T}|\mathbf{x}_{1:t-1})}\big[p_\theta(y \mid \mathbf{x})\big], \tag{7}$$

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 $p_\theta(\mathbf{x})$

$$\operatorname{logit} p_{\theta'}(x_t \mid \mathbf{x}_{1:t-1}) := \log p_\theta(x_t \mid \mathbf{x}_{1:t-1}) + \sigma \operatorname{Adv}_\theta(x_t \mid \mathbf{x}_{1:t-1}), \tag{8}$$

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 [25], and exploring low-density regions of $p_\theta(\mathbf{x})$. Yet, estimating individual advantages can result in high-variance updates, and estimating advantages over completed sequenceshas been shown to provide stronger empirical performance [54]. Consequently, the practical implementation of our method relies on global advantages, as detailed in the next section.

Practical Implementation In practice, to sample from the joint model $p_\theta(\mathbf{x}, y)$ (Figure 2), we follow [54] and first draw $\{\mathbf{x}^{(k)}\}_{k=1}^K \sim p_\theta(\mathbf{x})$ using Stochastic Beam Search (SBS), without replacement. By implementing SBS using the *Gumbel-Top-k trick*, sampling requires only a number of model evaluations that scales linearly with the number of samples K and sequence length T [36]. Next, for every sequence $\mathbf{x}^{(k)} \in \{\mathbf{x}^{(k)}\}_{k=1}^K$, we compute its *global advantage*

$$\operatorname{Adv}_\theta(\mathbf{x}^{(k)}) = p_\theta(y \mid \mathbf{x}^{(k)}) - \mu(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(K)}), \tag{9}$$

where $\mu(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(K)})$ is a consistent estimator of the *expected prediction score* $\mathbb{E}_{p_\theta(\mathbf{x})}[p_\theta(y \mid \mathbf{x})]$ [21, 36, 54]. Define $S(\mathbf{x}_{1:t}^{(k)})$ to be the set of all sequences $\mathbf{x} \in \{\mathbf{x}^{(k)}\}_{k=1}^K$ that share the prefix $\mathbf{x}_{1:t}^{(k)}$. Finally, we define the logits, i.e., unnormalized log-probabilities, of the perturbed generative component $p_{\theta'}(\mathbf{x})$, for each $\mathbf{x}^{(k)} \in \{\mathbf{x}^{(k)}\}_{k=1}^K$ and $t \in [T]$, as

$$\operatorname{logit} p_{\theta'}\big(x_t^{(k)} \mid \mathbf{x}_{1:t-1}^{(k)}\big) := \log\Big(p_\theta(x_t^{(k)} \mid \mathbf{x}_{1:t-1}^{(k)}) - \sum_{\mathbf{x} \in S(\mathbf{x}_{1:t}^{(k)})} p_\theta(\mathbf{x} \mid \mathbf{x}_{1:t-1}^{(k)})\Big) + \sigma \sum_{\mathbf{x} \in S(\mathbf{x}_{1:t}^{(k)})} \operatorname{Adv}_\theta(\mathbf{x}), \tag{10}$$

where 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 $p_\theta(x_t^{(k)} \mid \mathbf{x}_{1:t-1}^{(k)})$ (see Appendix C.1 for a derivation and Appendix D.1 for numerical stability). Intuitively, $\sum_{\mathbf{x} \in S(\mathbf{x}_{1:t}^{(k)})} p_\theta(\mathbf{x} \mid \mathbf{x}_{1:t-1}^{(k)})$ ensures that we sample without replacement [36], while $\sum_{\mathbf{x} \in S(\mathbf{x}_{1:t}^{(k)})} \operatorname{Adv}_\theta(\mathbf{x})$ biases the generative component towards samples with a high prediction score, according to the predictive component of the joint model. We iteratively repeat the above procedure for a fixed number of iterations, and summarize the complete procedure in Algorithm 1 (Appendix D).

5 Experiments

To evaluate JSI in sample-efficient molecular optimization, we benchmark our method on docking score optimization in both online (Section 5.1.1) and offline (Section 5.1.2) settings. To demonstrate the generalizability of our method beyond docking score optimization, we design an AMP optimization task, in which we consider offline datasets with a restricted proportion of high-scoring peptides, i.e., peptides with MIC values below a predefined threshold, further increasing task difficulty (Section 5.2). Additionally, we quantify the individual contributions of joint modeling and inference-time self-improvement through ablation studies (Section 5.3). Further experiments, covering surrogate accuracy, hyperparameter sensitivity, reduced evaluation budgets and offline dataset sizes, and sampling cost, are reported in Appendix E, and experimental details in Appendix F.

5.1 Docking Score Optimization

Experimental Setup We evaluate molecular optimization on docking score optimization across five protein targets: PARP1, FA7, JAK2, BRAF, and 5HT1B, while imposing feasibility constraints on drug-likeness [7] (QED) and synthetic accessibility [11] (SA), under an oracle budget of 3,000 evaluations [40, 19], a more stringent regime than the 10,000 evaluations used by Gao et al. [14].

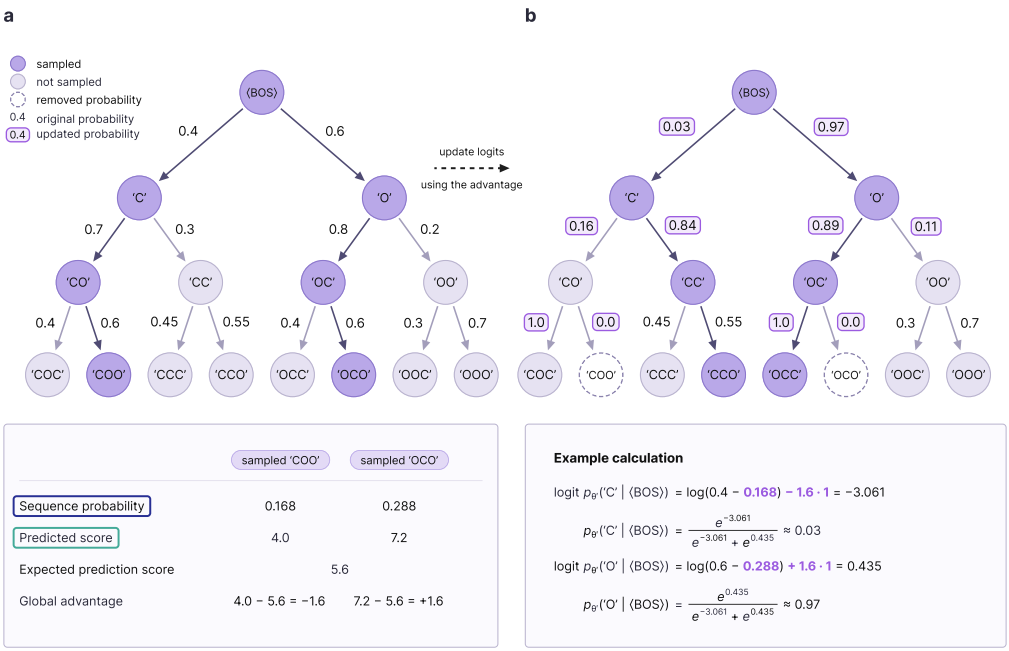

Figure 2: Example logit update in Eq. (10) with $K = 2$, $\sigma = 1$ and $\mathcal{V} = \{C, O\}$. JSI uses SBS to autoregressively generate K molecules, inducing a tree where each node is a partial sequence and edges carry next-token probabilities. The predictive head scores each completed sequence, and the scores are aggregated into a global advantage that biases next-token probabilities. Panel **(a)** shows the initial probabilities, while panel **(b)** shows probabilities biased towards high-scoring branches.

Table 1: Molecular optimization performance on docking score optimization in the online setting. We mark the best method **bold**.

	Target Protein, Hit Ratio (↑)				
Method	PARP1	FA7	5HT1B	BRAF	JAK2
ZINC250K	3.993 (0.355)	1.097 (0.192)	24.26 (0.622)	1.020 (0.193)	6.183 (0.344)
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)
GraphAF	0.822 (0.113)	0.011 (0.016)	6.978 (0.952)	1.422 (0.556)	1.233 (0.284)
MORLD	0.047 (0.050)	0.007 (0.013)	0.893 (0.758)	0.047 (0.040)	0.227 (0.118)
HierVAE	1.180 (0.182)	0.033 (0.030)	0.740 (0.371)	0.367 (0.187)	0.487 (0.183)
FREED-QS	5.960 (0.902)	1.687 (0.177)	23.140 (2.422)	3.880 (0.623)	7.653 (1.373)
LIMO	0.456 (0.057)	0.044 (0.016)	1.200 (0.178)	0.278 (0.134)	0.711 (0.329)
GDSS	2.367 (0.316)	0.467 (0.112)	6.267 (0.287)	0.300 (0.198)	1.367 (0.258)
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)
NF-BO	18.641 (8.072)	5.305 (3.568)	53.185 (3.606)	16.593 (3.577)	29.537 (6.949)
JSI (ours)	**67.280 (7.961)**	**36.587 (10.270)**	**91.357 (3.204)**	**58.547 (11.456)**	**71.863 (6.127)**

Metrics Following [40, 19], we measure **Hit Ratio** (%), defined as the fraction of sampled hits, i.e., molecules with a docking score strictly better than the median of known actives [19], QED > 0.5, and SA < 5. Additionally, we evaluate internal diversity (**IntDiv1**) [6] and oracle burden (**OB**), where OB(n) denotes the number of oracle calls required to obtain n unique hits. For each metric, we report the mean and standard deviation calculated across 10 random seeds.

5.1.1 Sample-Efficient Online Molecular Optimization

To evaluate the sample efficiency of JSI in the online setting, we follow the experimental setup of Guo et al. [19]. We generate optimized molecules, by using Hyformer [25] pre-trained on ZINC250K [68], as the backbone for JSI, and defining the reward $R(\mathbf{x})$ to calculate the global advantage of a sequence $\mathbf{x}$, in place of the prediction score in Eq. (9), as

$$R(\mathbf{x}) = \frac{1}{3}\Big(-\mathrm{DS}^{\mathrm{z}}(\mathbf{x}) - \mathrm{SA}^{\mathrm{z}}(\mathbf{x}) + \mathrm{QED}^{\mathrm{z}}(\mathbf{x})\Big), \tag{11}$$

Table 2: Oracle burden in the online setting. We mark the best method **bold**.

		Target Protein				
	Method	PARP1	FA7	5HT1B	BRAF	JAK2
SATURN	OB(1) (↓)	35 (28)	201 (172)	4 (3)	130 (144)	30 (29)
	OB(10) (↓)	282 (116)	715 (187)	45 (13)	517 (167)	227 (93)
	OB(100) (↓)	721 (137)	1408 (304)	347 (49)	1047 (301)	720 (131)
	IntDiv1 (↑)	**0.627 (0.023)**	**0.597 (0.056)**	**0.665 (0.032)**	**0.572 (0.063)**	**0.641 (0.056)**
JSI (ours)	OB(1) (↓)	**25 (21)**	**106 (77)**	**3 (2)**	**68 (72)**	**13 (9)**
	OB(10) (↓)	**186 (143)**	**424 (217)**	**25 (6)**	**210 (134)**	**94 (29)**
	OB(100) (↓)	**513 (171)**	**1106 (388)**	**184 (47)**	**632 (330)**	**401 (139)**
	IntDiv1 (↑)	0.571 (0.037)	0.571 (0.019)	0.595 (0.053)	0.554 (0.038)	0.586 (0.038)

where DS^z, SA^z and QED^z are Z-score normalized [22] values of DS, SA and QED, respectively, calculated using the ground-truth oracle. Next, we use the generated molecules to fine-tune the backbone using the generative-only part of the joint loss $\mathcal{L}_{\lambda=0}$ (Eq. (6)). We repeat the above procedure until the oracle budget of 3,000 evaluations is exhausted. Further experimental details, together with an overview of the different optimization settings (Figure 5), are in Appendix F.2. We analyze the choice of reward normalization and aggregation in Appendix E.1, and the relative weighting of the individual properties in Appendix E.1.1.

We compare JSI with the baselines reported by Guo et al. [19]: ZINC250K [68], molecules sampled at random from the pre-training dataset; REINVENT [45], a SMILES-based generative model fine-tuned with reinforcement learning towards an augmented likelihood; JT-VAE [28], a graph-based variational autoencoder; GraphAF [63], a flow-based autoregressive graph generator; MORLD [27], which optimizes molecules with deep Q-learning on docking rewards; HierVAE [29], a hierarchical motif-by-motif decoder; FREED-QS [76], a fragment-based reinforcement learning approach with experience replay; LIMO [10], which optimizes a VAE latent space via a differentiable property predictor; GDSS [30], a score-based graph diffusion model; MOOD [39], a conditional diffusion-based sampling method; Augmented Memory [18], an extension of REINVENT that combines experience replay with SMILES augmentation; GEAM [40], which integrates reinforcement learning with a genetic algorithm for goal-aware fragment-based optimization; and SATURN [19], a REINVENT-style optimization framework with a Mamba backbone. We additionally evaluate NF-BO [41], a latent Bayesian optimization method built on an autoregressive normalizing flow, which we run ourselves.

JSI consistently outperforms all baselines across all protein targets (Table 1). REINVENT achieves a low overall Hit Ratio, indicating limited ability to leverage oracle feedback under strict evaluation budgets. In contrast, methods designed to improve sample efficiency, such as Augmented Memory and SATURN, yield substantial gains, but still underperform compared to JSI. Investigation of performance across all oracle budgets up to 3,000 (Figure 3) demonstrates that JSI improves over state-of-the-art SATURN at any given budget. This suggests that JSI provides a strong inductive bias for sample-efficient online molecular optimization, rapidly identifying hit molecules, especially at the early stages of optimization. Finally, JSI consistently reduces oracle burden across all protein targets, at the cost of slightly decreased diversity, with the differences being particularly pronounced for harder targets such as FA7 and BRAF (Table 2). We additionally report the validity, novelty and similarity to the training set of the generated molecules in Appendix E.6.

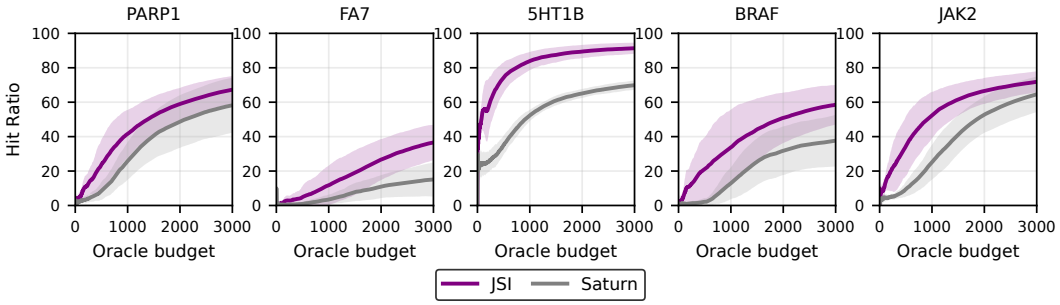

Figure 3: Hit ratio versus oracle budget across five protein targets. Curves show the mean across 10 seeds for JSI (purple) and SATURN (gray); shaded regions indicate ± 1 standard deviation over seeds.

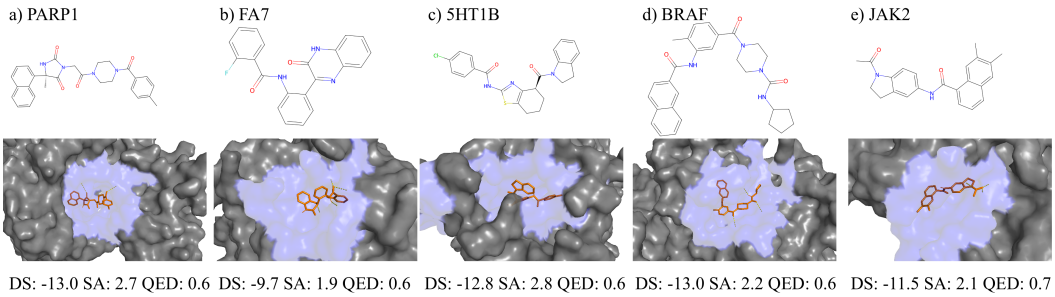

Figure 4: Binding poses of best molecules generated by JSI in the offline optimization setting.

5.1.2 Sample-Efficient Offline Molecular Optimization

To evaluate the sample efficiency of JSI in the offline setting, we follow the experimental setup of Shin et al. [65]. For each of 5 protein targets and 10 random seeds, we construct an offline dataset by randomly selecting 1,500 molecules from ZINC250K and computing their docking score, SA and QED. Next, we fine-tune Hyformer, pre-trained on ZINC250K, using the joint loss $\mathcal{L}_{\lambda=0.25}$ (Eq. (6)). In contrast to the online setting, where ground-truth oracle evaluations are available, here we update both the generative and predictive parts of the joint model. To generate optimized molecules, we iteratively bias the generative part of the joint model using the reward

$$\hat{R}(\mathbf{x}) = \frac{1}{3}\Big(-\hat{\mathrm{DS}}^{\mathrm{z}}(\mathbf{x}) - \hat{\mathrm{SA}}^{\mathrm{z}}(\mathbf{x}) + \hat{\mathrm{QED}}^{\mathrm{z}}(\mathbf{x})\Big), \tag{12}$$

where $\hat{\mathrm{DS}}^{\mathrm{z}}$, $\hat{\mathrm{SA}}^{\mathrm{z}}$ and $\hat{\mathrm{QED}}^{\mathrm{z}}$ are the Z-score normalized docking score, SA and QED, respectively, estimated by the predictive part of the joint model. Finally, we use the remaining oracle budget of 1,500 to evaluate the generated molecules. Further experimental details are in Appendix F.3.

Following [65], we compare JSI against: GraphGA [26], a graph-based genetic algorithm; REINVENT [45]; SATURN [19]; GeneticGFN [31], combining genetic algorithms with GFlowNets; and RaM [70], a ranking-based offline optimization method that replaces regression-based surrogate learning with learning-to-rank objectives. We do not compare against MolStitch [65] due to reproducibility limitations.

As in the online setting, here again JSI outperforms all baselines across all protein targets (Table 3). REINVENT and SATURN yield consistently low Hit Ratios across all targets, highlighting their sample inefficiency. RaM represents a strong baseline, substantially improving upon earlier methods. Additionally, to qualitatively validate the performance of JSI in the offline setting, we show binding poses of the best hit molecules, i.e., with the lowest docking scores, in Figure 4.

5.2 AMP Optimization

Experimental Setup We evaluate JSI on the AMP optimization task, adapting the setup of Izdebski et al. [25] to the sample-efficient setting by restricting the evaluation budget to 1K peptide sequences. In particular, we use Hyformer [25] as the backbone for JSI, pre-trained on 3.5M peptide sequences and subsequently on 1M AMP sequences. The offline dataset consists of 3,444 peptides with associated MIC values against *E. coli*, following Szymczak et al. [69]. While prior work defines activity labels by thresholding MIC values, labeling peptides with MIC $\leq 10^{1.5} \simeq 32\,\mu\mathrm{g/mL}$ as hits, we treat MIC prediction as a regression task and use the continuous values during training and

Table 3: Molecular optimization performance on docking score optimization in the offline setting. We mark the best method **bold**.

	Target Protein, Hit Ratio (↑)				
Method	PARP1	FA7	5HT1B	BRAF	JAK2
GraphGA	29.631 (20.478)	0.913 (0.942)	73.875 (4.945)	3.340 (4.056)	28.912 (14.910)
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)
JSI (ours)	**45.628 (11.653)**	**5.831 (2.040)**	**76.940 (4.442)**	**21.637 (4.607)**	**42.001 (4.344)**

optimization. The thresholded labels are used only to characterize the dataset composition. To further increase task difficulty, we construct imbalanced offline datasets that retain the 1,318 non-hit peptides but restrict the proportion of hits to 1% (13 peptides) and 0% (no hits). For each method, we report the mean and 95% confidence intervals estimated via bootstrapping.

Metrics For evaluation, we consider **perplexity** [72], **diversity**, and **fitness** [44], along with three state-of-the-art *in silico* oracles: **HydrAMP$_{\text{MIC}}$** [69], which predicts the probability that a peptide is active against *E. coli*, and **AMPlify** [43] and **AmPEPpy** [38] for predicting the probability that a peptide is antimicrobial.

5.2.1 Sample-Efficient Offline AMP Optimization

To sample optimized AMPs with JSI, we fine-tune the pre-trained backbone using the joint loss $\mathcal{L}_{\lambda=0.4}$ (Equation (6)) and bias the generative component at inference time using the predictive component of the joint model as the reward. The predictor estimates MIC values for generated peptide sequences, which we normalize into $\hat{\text{MIC}}$ via z-scoring. The reward is defined as $R(\mathbf{x}) = -\hat{\text{MIC}}(\mathbf{x})$, favoring sequences with lower predicted MIC values. We sample until 1,000 peptide sequences are collected, without further retraining. Generated peptides are evaluated using the metrics defined above. All experimental details are provided in Appendix F.4.

We compare JSI against established baselines: PepCVAE [9], AMPGAN [51], HydrAMP [69], AMP-Diffusion [8], and Hyformer [25].

JSI outperforms all baselines on AMP-related oracles: HydrAMP$_{\text{MIC}}$, AMPlify, and AmPEPpy, while maintaining competitive diversity and fitness (Table 4). Specifically, while Hyformer achieves the highest fitness score, JSI improves upon Hyformer across all three AMP oracles, indicating stronger alignment with antimicrobial activity objectives. Baselines such as PepCVAE, AMPGAN, and AMP-Diffusion achieve considerably lower performance across all AMP-related metrics. We report the uniqueness and novelty of the generated peptides in Appendix E.4, and evaluate JSI under reduced evaluation budgets and offline dataset sizes in Appendix E.5.

Under the challenging setting of restricting the proportion of high-scoring peptides, JSI maintains strong performance across all three AMP oracles, outperforming Hyformer by a large margin, notably even in the 0% setting, where no hits are available during fine-tuning (Table 5). While Hyformer maintains higher diversity, JSI consistently achieves superior fitness and AMP oracle scores. The difference is particularly pronounced on HydrAMP$_{\text{MIC}}$, the most challenging oracle, where JSI achieves over threefold improvement over Hyformer in both the 1% and 0% settings.

Table 4: Molecular optimization performance on offline AMP optimization. Best method is **bold**.

Model	Perplexity	Diversity (↑)	Fitness (↑)	HydrAMP$_{\text{MIC}}$ (↑)	AMPlify (↑)	AmPEPpy (↑)
PepCVAE	20.39 (20.17, 20.64)	**0.87 (0.87, 0.87)**	0.07 (0.07, 0.07)	0.12 (0.10, 0.14)	0.48 (0.46, 0.50)	0.52 (0.51, 0.53)
AMPGAN	18.97 (18.76, 19.19)	0.80 (0.80, 0.81)	0.12 (0.12, 0.13)	0.22 (0.20, 0.25)	0.66 (0.63, 0.68)	0.55 (0.54, 0.57)
HydrAMP	19.98 (19.71, 20.27)	0.86 (0.86, 0.86)	0.09 (0.08, 0.09)	0.44 (0.41, 0.47)	0.60 (0.57, 0.62)	0.52 (0.52, 0.53)
AMP-Diffusion	18.33 (18.03, 18.64)	0.82 (0.82, 0.82)	0.11 (0.10, 0.11)	0.18 (0.16, 0.21)	0.14 (0.12, 0.15)	0.33 (0.31, 0.34)
Hyformer	17.72 (17.47, 17.97)	0.74 (0.73, 0.74)	**0.21 (0.20, 0.22)**	0.78 (0.76, 0.81)	0.96 (0.95, 0.97)	0.71 (0.70, 0.72)
JSI (ours)	15.93 (15.68, 16.17)	0.78 (0.78, 0.79)	0.12 (0.12, 0.12)	**0.84 (0.82, 0.87)**	**0.98 (0.97, 0.98)**	**0.78 (0.77, 0.78)**

Table 5: Molecular optimization performance on offline AMP optimization with restricted proportion of hits. Best method is **bold**.

Model	Perplexity	Diversity (↑)	Fitness (↑)	HydrAMP$_{\text{MIC}}$ (↑)	AMPlify (↑)	AmPEPpy (↑)
1% hits						
Hyformer	16.84 (16.59, 17.09)	**0.83 (0.82, 0.83)**	0.11 (0.11, 0.12)	0.20 (0.18, 0.23)	0.76 (0.74, 0.78)	0.65 (0.64, 0.66)
JSI (ours)	18.36 (17.94, 18.77)	0.69 (0.68, 0.70)	**0.16 (0.16, 0.17)**	**0.68 (0.65, 0.71)**	**0.98 (0.97, 0.98)**	**0.72 (0.72, 0.73)**
0% hits						
Hyformer	16.77 (16.51, 17.01)	**0.83 (0.83, 0.83)**	0.11 (0.10, 0.11)	0.17 (0.15, 0.19)	0.71 (0.69, 0.73)	0.62 (0.61, 0.63)
JSI (ours)	18.34 (18.05, 18.66)	0.72 (0.72, 0.73)	**0.19 (0.18, 0.20)**	**0.66 (0.63, 0.69)**	**0.98 (0.97, 0.99)**	**0.71 (0.70, 0.72)**

5.3 Ablations

To quantify the individual contributions of joint learning and inference-time self-improvement, we ablate JSI on sample-efficient offline docking score optimization in two variants: (i) W/O JOINT

MODELING, which decouples generation and prediction by using the offline dataset to separately fine-tune the generative part of the joint model and a standalone predictor, and (ii) W/O INFERENCE-TIME SELF-IMPROVEMENT (SI), which reduces our sampling procedure to Best-of-K sampling [13, 3], without replacement. For the predictor, we use Chemprop [20], a state-of-the-art message-passing neural network for molecular property prediction. For IntDiv1 in Table 6, we report the mean across 10 random seeds, aggregated over all protein targets. All experiments are conducted in the offline docking score optimization setting, with all details in Appendix F.5.

JSI significantly outperforms both ablated variants on all protein targets (two-sided Welch's t-test, $p < 0.05$), except on 5HT1B. We hypothesize that 5HT1B remains the easiest protein target, as compounds with high binding affinity are relatively common in the ZINC dataset (see Table 1), which explains the relatively high performance of the ablated versions. As expected, the variant without inference-time SI attains a slightly higher diversity.

Table 6: Ablations for molecular optimization performance on docking score optimization in the offline setting. We mark the best method **bold**. SI - self improvement.

	Target Protein, Hit Ratio (↑)					
Method	PARP1	FA7	5HT1B	BRAF	JAK2	IntDiv1 (↑)
JSI	**45.628 (11.653)**	**5.831 (2.040)**	76.940 (4.442)	**21.637 (4.607)**	**42.001 (4.344)**	0.763 (0.015)
W/O JOINT MODELING	25.309 (8.673)	0.479 (0.771)	**77.579 (4.343)**	7.298 (3.295)	13.333 (6.047)	0.713 (0.043)
W/O INFERENCE-TIME SI	29.468 (4.059)	2.629 (0.619)	73.347 (2.131)	10.978 (1.936)	29.432 (2.346)	**0.826 (0.009)**

6 Conclusions

In this paper, we introduced JSI, a framework for sample-efficient molecular optimization in both online and offline settings. By combining a joint generative-predictive model with inference-time self-improvement, JSI successfully improves sample efficiency, providing state-of-the-art performance in docking score optimization across both offline and online settings, and in AMP optimization. Notably, JSI generates optimized peptides even when no high-scoring examples are present in the offline data.

Limitations & Future Work In challenging tasks, inaccuracies in the surrogate component may limit optimization performance. While the surrogate correlates strongly with the oracle on held-out molecules, this correlation degrades on the generated molecules. As sampling is performed at inference-time, rather than through iterative fine-tuning, inaccuracies do not accumulate, yet future work should explore methods to further improve the robustness of surrogates in the offline setting. Moreover, inference-time self-improvement trades off diversity for optimization performance, and explicitly controlling this trade-off remains a challenge. Despite these limitations, JSI outperforms state-of-the-art approaches in sample-efficient molecular optimization across both offline and online settings, with ablations confirming the individual contributions of joint modeling and inference-time self-improvement.

7 Acknowledgments and Disclosures

We thank Irene Bonafonte-Pardàs for her valuable comments that improved the final version of the manuscript. This work was supported by Helmholtz Munich's Health AI Fellowship to Adam Izdebski, ERC grant DOG-AMP (No 101125506), funded by the European Union, and the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) through Project No. 466387255 within the Priority Programme "SPP 2331: Machine Learning in Chemical Engineering" (JP, DGG). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them. We acknowledge the use of the HPC cluster at Helmholtz Munich and of generative AI tools to improve the final version of the manuscript. All AI-generated content was reviewed, edited and approved by the authors.



References

[1] A. Agarwal, S. M. Kakade, J. D. Lee, and G. Mahajan. Optimality and approximation with policy gradient methods in markov decision processes. In J. Abernethy and S. Agarwal, editors, *Proceedings of Thirty Third Conference on Learning Theory*, volume 125 of *Proceedings of Machine Learning Research*, pages 64–66. PMLR, 09–12 Jul 2020.

[2] A. Alhossary, S. D. Handoko, Y. Mu, and C. Kwoh. Fast, accurate, and reliable molecular docking with quickvina 2. *Bioinformatics*, 31(13):2214–2216, 2015. doi: 10.1093/bioinformatics/btv082.

[3] A. Beirami, A. Agarwal, J. Berant, A. D'Amour, J. Eisenstein, C. Nagpal, and A. T. Suresh. Theoretical guarantees on the best-of-n alignment policy. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 3580–3602. PMLR, 2025. URL `https://proceedings.mlr.press/v267/beirami25a.html`.

[4] G. W. Bemis and M. A. Murcko. The properties of known drugs. 1. molecular frameworks. *Journal of Medicinal Chemistry*, 39(15):2887–2893, 1996.

[5] Y. Bengio, S. Lahlou, T. Deleu, E. J. Hu, M. Tiwari, and E. Bengio. Gflownet foundations. *Journal of Machine Learning Research*, 24(210):1–55, 2023.

[6] M. Benhenda. Chemgan challenge for drug discovery: can ai reproduce natural chemical diversity? *arXiv preprint arXiv:1708.08227*, 2017. URL `https://arxiv.org/abs/1708.08227`.

[7] G. R. Bickerton, G. V. Paolini, J. Besnard, S. Muresan, and A. L. Hopkins. Quantifying the chemical beauty of drugs. *Nature chemistry*, 4(2):90–98, 2012.

[8] T. Chen, P. Vure, R. Pulugurta, and P. Chatterjee. Amp-diffusion: Integrating latent diffusion with protein language models for antimicrobial peptide generation. In *NeurIPS 2023 Generative AI and Biology (GenBio) Workshop*, 2023.

[9] P. Das, K. Wadhawan, O. Chang, T. Sercu, C. D. Santos, M. Riemer, V. Chenthamarakshan, I. Padhi, and A. Mojsilovic. Pepcvae: Semi-supervised targeted design of antimicrobial peptide sequences. *arXiv preprint arXiv:1810.07743*, 2018. URL `https://arxiv.org/abs/1810.07743`.

[10] P. Eckmann, K. Sun, B. Zhao, M. Feng, M. K. Gilson, and R. Yu. LIMO: Latent inceptionism for targeted molecule generation. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 5777–5792. PMLR, 2022. URL `https://proceedings.mlr.press/v162/eckmann22a.html`.

[11] P. Ertl, R. Lewis, E. Martin, and V. Polyakov. In silico generation of novel, drug-like chemical matter using the LSTM neural network, Jan. 2018.

[12] T. Fischer, S. Gazzola, and R. 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.

[13] L. Gao, J. Schulman, and J. Hilton. Scaling laws for reward model overoptimization. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 10835–10866. PMLR, 2023. URL `https://proceedings.mlr.press/v202/gao23h.html`.

[14] W. Gao, T. Fu, J. Sun, and C. Coley. Sample efficiency matters: A benchmark for practical molecular optimization. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 21342–21357. Curran Associates, Inc., 2022.

[15] R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, and A. Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

[16] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[17] W. Grathwohl, K.-C. Wang, J.-H. Jacobsen, D. Duvenaud, M. Norouzi, and K. Swersky. Your classifier is secretly an energy based model and you should treat it like one, 2020.

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

[19] J. Guo, J. Chen, A. G. Chen, and P. Schwaller. Sample-efficient generative molecular design using memory manipulation. *Nature Machine Intelligence*, 8:449–460, 2026. doi: 10.1038/s42256-026-01200-4.

[20] E. Heid, K. P. Greenman, Y. Chung, S.-C. Li, D. E. Graff, F. H. Vermeire, H. Wu, W. H. Green, and C. J. McGill. Chemprop: a machine learning package for chemical property prediction. *Journal of Chemical Information and Modeling*, 64(1):9–17, 2023.

[21] T. C. Hesterberg. *Advances in importance sampling*. Stanford University, 1988.

[22] J. Hu, J. K. Liu, H. Xu, and W. Shen. Reinforce++: Stabilizing critic-free policy optimization with global normalization. *arXiv preprint arXiv:2501.03262*, 2025.

[23] J. P. Hughes, S. Rees, S. B. Kalindjian, and K. L. Philpott. Principles of early drug discovery. *British journal of pharmacology*, 162(6):1239–1249, 2011.

[24] A. Izdebski, E. Weglarz-Tomczak, E. Szczurek, and J. M. Tomczak. De novo drug design with joint transformers, 2023. URL https://arxiv.org/abs/2310.02066.

[25] A. Izdebski, J. Olszewski, P. Gawade, K. Koras, S. Korkmaz, V. Rauscher, J. M. Tomczak, and E. 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.

[26] J. 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.

[27] W. Jeon and D. Kim. Autonomous molecule generation using reinforcement learning and docking to develop potential novel inhibitors. *Scientific reports*, 10(1):22104, 2020.

[28] W. Jin, R. Barzilay, and T. Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 2323–2332. PMLR, 2018.

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

[30] J. Jo, S. Lee, and S. J. Hwang. Score-based generative modeling of graphs via the system of stochastic differential equations. In *International conference on machine learning*, pages 10362–10383. PMLR, 2022.

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

[32] J. Kim, S. Park, D. Min, and W. Kim. Comprehensive survey of recent drug discovery using deep learning. *International Journal of Molecular Sciences*, 22(18):9983, 2021.

[33] M. Kim, F. Berto, S. Ahn, and J. 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*, pages 67643–67661, 2023.

[34] M. Kim, J. Gu, Y. Yuan, T. Yun, Z. Liu, Y. Bengio, and C. Chen. Offline model-based optimization: Comprehensive review, 2025. URL https://arxiv.org/abs/2503.17286.

[35] J. Knoblauch, J. Jewson, and T. Damoulas. Generalized variational inference: Three arguments for deriving new posteriors, 2019. URL https://arxiv.org/abs/1904.02063.

[36] W. Kool, H. Van Hoof, and M. Welling. Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement. In *International conference on machine learning*, pages 3499–3508. PMLR, 2019.

[37] T. Korbak, H. Elsahar, G. Kruszewski, and M. Dymetman. On reinforcement learning and distribution matching for fine-tuning language models with no catastrophic forgetting, 2022. URL https://arxiv.org/abs/2206.00761.

[38] T. J. Lawrence, D. L. Carper, M. K. Spangler, A. A. Carrell, T. A. Rush, S. J. Minter, D. J. Weston, and J. L. Labbé. Ampeppy 1.0: A portable and accurate antimicrobial peptide prediction tool. *Bioinformatics*, 37(14):2058–2060, 2020. doi: 10.1093/bioinformatics/btaa917. URL https://doi.org/10.1093/bioinformatics/btaa917.

[39] S. Lee, J. Jo, and S. J. Hwang. Exploring chemical space with score-based out-of-distribution generation. In *International Conference on Machine Learning*, pages 18872–18892. PMLR, 2023.

[40] S. Lee, S. Lee, K. Kawaguchi, and S. J. Hwang. Drug discovery with dynamic goal-aware fragments. In R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, and F. Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 26731–26751. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/lee24o.html.

[41] S. Lee, J. Park, J. Chu, M. Yoon, and H. J. Kim. Latent bayesian optimization via autoregressive normalizing flows. In *International Conference on Learning Representations*, 2025.

[42] S. Levine, A. Kumar, G. Tucker, and J. Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[43] C. Li, D. Sutherland, S. A. Hammond, C. Yang, F. Taho, L. Bergman, S. Houston, R. L. Warren, T. Wong, L. M. N. Hoang, C. E. Cameron, C. C. Helbing, and I. Birol. Amplify: Attentive deep learning model for discovery of novel antimicrobial peptides effective against who priority pathogens. *BMC Genomics*, 23(1):77, Jan 2022. doi: 10.1186/s12864-021-08282-3.

[44] T. Li, X. Ren, X. Luo, Z. Wang, Z. Li, X. Luo, J. Shen, Y. Li, D. Yuan, R. Nussinov, X. Zeng, J. Shi, and F. Cheng. A foundation model identifies broad-spectrum antimicrobial peptides against drug-resistant bacterial infection. *Nature Communications*, 15(1):7538, Aug 2024.

[45] H. H. Loeffler, J. He, A. Tibo, J. P. Janet, A. Voronov, L. H. Mervin, and O. Engkvist. Reinvent 4: modern ai–driven generative molecule design. *Journal of Cheminformatics*, 16(1):20, 2024.

[46] N. Maus, H. Jones, J. Moore, M. J. Kusner, J. Bradshaw, and J. Gardner. Local latent space bayesian optimization over structured inputs. *Advances in neural information processing systems*, 35:34505–34518, 2022.

[47] R. Møller-Larsen, A. Izdebski, J. Olszewski, P. Gawade, M. Kmicikiewicz, W. Zarzecki, and E. Szczurek. seqme: a python library for evaluating biological sequence design from generative models. *Bioinformatics Advances*, 6(1):vbag212, 2026.

[48] K. Murphy. Reinforcement learning: An overview, 2025. URL https://arxiv.org/abs/2412.05265.

[49] E. Nalisnick, A. Matsukawa, Y. W. Teh, D. Gorur, and B. Lakshminarayanan. Hybrid models with deep and invertible features. In *International Conference on Machine Learning*, pages 4723–4732. PMLR, 2019.

[50] E. Nijkamp, J. A. Ruffolo, E. N. Weinstein, N. Naik, and A. Madani. ProGen2: Exploring the boundaries of protein language models. *Cell Systems*, 14(11):968–978.e3, 2023. doi: 10.1016/j.cels.2023.10.002.

[51] C. M. V. Oort, J. B. Ferrell, J. M. Remington, S. Wshah, and J. Li. Ampgan v2: Machine learning-guided design of antimicrobial peptides. *Journal of Chemical Information and Modeling*, 61(5): 2198–2207, 2021. doi: 10.1021/acs.jcim.0c01441.

[52] R. Özçelik, H. Brinkmann, E. Criscuolo, and F. 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.

[53] J. Peters and S. Schaal. Policy gradient methods for robotics. In *2006 IEEE/RSJ international conference on intelligent robots and systems*, pages 2219–2225. IEEE, 2006.

[54] J. Pirnay and D. 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.

[55] J. Pirnay, J. G. Rittig, A. B. Wolf, M. Grohe, J. Burger, A. Mitsos, and D. G. Grimm. Graphxform: graph transformer for computer-aided molecular design. *Digital Discovery*, 4(4):1052–1065, 2025.

[56] K. Preuer, P. Renz, T. Unterthiner, S. Hochreiter, and G. Klambauer. Fréchet ChemNet distance: A metric for generative models for molecules in drug discovery. *Journal of Chemical Information and Modeling*, 58(9):1736–1741, 2018.

[57] H. Qi, Y. Su, A. Kumar, and S. Levine. Data-driven offline decision-making via invariant representation learning. In *Advances in Neural Information Processing Systems*, 2022.

[58] R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, and C. Finn. Direct preference optimization: Your language model is secretly a reward model. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 53728–53741. Curran Associates, Inc., 2023.

[59] M. C. Ramos, C. J. Collison, and A. D. White. A review of large language models and autonomous agents in chemistry. *Chem. Sci.*, 16:2514–2572, 2025. doi: 10.1039/D4SC03921A.

[60] RDKit Development Team. RDKit: Open-source cheminformatics. https://www.rdkit.org.

[61] D. Rengarajan, G. Vaidya, A. Sarvesh, D. Kalathil, and S. 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.

[62] D. Rogers and M. Hahn. Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling*, 50(5):742–754, 2010.

[63] C. Shi, M. Xu, Z. Zhu, W. Zhang, M. Zhang, and J. Tang. Graphaf: a flow-based autoregressive model for molecular graph generation. *arXiv preprint arXiv:2001.09382*, 2020.

[64] K. Shi, D. Bieber, and C. Sutton. Incremental sampling without replacement for sequence models, 2021. URL https://arxiv.org/abs/2002.09067.

[65] D.-H. Shin, Y.-H. Son, H. J. Lee, D.-J. Lee, and T.-E. 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.

[66] Y. Song, Y. Zhou, A. Sekhari, D. Bagnell, A. Krishnamurthy, and W. 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.

[67] B. Sridharan, M. Goel, and U. D. Priyakumar. Modern machine learning for tackling inverse problems in chemistry: molecular design to realization. *Chemical communications*, 58(35): 5316–5331, 2022.

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

[69] P. Szymczak, M. Możejko, T. Grzegorzek, R. Jurczak, M. Bauer, D. Neubauer, K. Sikora, M. Michalski, J. Sroka, P. Setny, W. Kamysz, and E. Szczurek. Discovering highly potent antimicrobial peptides with deep generative model hydramp. *Nature Communications*, 14 (1):1453, 2023. doi: 10.1038/s41467-023-36994-z. URL https://doi.org/10.1038/s41467-023-36994-z.

[70] R.-X. Tan, K. Xue, S.-H. Lyu, H. Shang, Y. Wang, Y. Wang, S. Fu, and C. Qian. Offline model-based optimization by learning to rank. *arXiv preprint arXiv:2410.11502*, 2024.

[71] Y. Tang and R. Munos. On a few pitfalls in kl divergence gradient estimation for rl, 2025. URL https://arxiv.org/abs/2506.09477.

[72] M. D. T. Torres, T. Chen, F. Wan, P. Chatterjee, and C. de la Fuente-Nunez. Generative latent diffusion language modeling yields anti-infective synthetic peptides. *bioRxiv*, 2025. doi: 10.1101/2025.01.31.636003. URL https://www.biorxiv.org/content/early/2025/02/01/2025.01.31.636003.

[73] B. Trabucco, A. Kumar, X. Geng, and S. Levine. Conservative objective models for effective offline model-based optimization. In *Proceedings of the International Conference on Machine Learning*, pages 10358–10368. PMLR, 2021.

[74] A. Tripp, E. Daxberger, and J. M. 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.

[75] G. Vasan, Y. Wang, F. Shahriar, J. Bergstra, M. Jagersand, and A. R. Mahmood. Revisiting sparse rewards for goal-reaching reinforcement learning, 2024. URL https://arxiv.org/abs/2407.00324.

[76] S. Yang, D. Hwang, S. Lee, S. Ryu, and S. J. Hwang. Hit and lead discovery with explorative rl and fragment-based molecule generation. *Advances in Neural Information Processing Systems*, 34:7924–7936, 2021.

[77] K. Zhang, X. Yang, Y. Wang, Y. Yu, N. Huang, G. Li, X. Li, J. C. Wu, and S. Yang. Artificial intelligence in drug development. *Nature medicine*, 31(1):45–59, 2025.

[78] T. Zhao, H. Hachiya, G. Niu, and M. Sugiyama. Analysis and improvement of policy gradient estimation. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

A Impact Statement

This work advances deep generative modeling for molecular optimization, with direct applications to drug discovery. The primary societal benefit is accelerating the design of novel therapeutics, particularly for indications where experimental screening is expensive or slow. A potential risk of generative molecular optimization is dual-use: the same methods could in principle be applied to design harmful substances. We encourage responsible use of these tools within established ethical and regulatory frameworks governing chemical and biological research.

B Notation

We follow the mathematical notation of Goodfellow et al. [16]. Table 7 summarizes the symbols used throughout the paper.

C Theory and Derivations

C.1 Derivation of the Perturbed Logits

We derive the perturbed logits in Eq. (10), showing that they combine sampling without replacement with a reward-dependent exponential tilt, and the subtracted probability term is the probability mass remaining under a prefix after previously selected complete sequences have been removed.

Table 7: Mathematical notation used throughout the paper.

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
$\mathcal{V}$	Vocabulary, i.e. the set of tokens
$\mathcal{D}$	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 $\theta \in \Theta$

Let $u := \mathbf{x}_{1:t-1}$ denote the current parent prefix and let $a := x_t \in \mathcal{V}$ be a candidate next token from the vocabulary $\mathcal{V}$, and write ua for their concatenation. Let $\mathcal{C}(ua) = \{\mathbf{x} : \mathbf{x}_{1:t} = ua\}$ be the set of all complete sequences extending ua. For an autoregressive model

$$p_\theta(a \mid u) = \sum_{\mathbf{x} \in \mathcal{C}(ua)} p_\theta(\mathbf{x} \mid u). \tag{13}$$

Let $\mathcal{S} = \{\mathbf{x}^{(k)}\}_{k=1}^{K}$ be the set of complete sequences sampled so far, and define $S(ua) = \mathcal{S} \cap \mathcal{C}(ua)$. The quantity inside the logarithm in Eq. (10) is then

$$r_u(a) := p_\theta(a \mid u) - \sum_{\mathbf{x} \in S(ua)} p_\theta(\mathbf{x} \mid u) = \sum_{\mathbf{x} \in \mathcal{C}(ua) \setminus \mathcal{S}} p_\theta(\mathbf{x} \mid u), \tag{14}$$

which immediately gives $r_u(a) \geq 0$. Moreover, $r_u(a) > 0$ whenever at least one unsampled completion of ua has positive model probability. The term is therefore strictly positive for every *active* (i.e., visitable) branch of the trie. It is exactly zero only when the branch is exhausted (its full subtree has been sampled), in which case the logit is $\log(0) = -\infty$ and the branch cannot be selected again. This construction yields sampling without replacement. At a non-exhausted parent prefix u, the total residual mass is

$$Z_u = \sum_{a \in \mathcal{V}} r_u(a) = 1 - \sum_{\mathbf{x} \in \mathcal{S} \cap \mathcal{C}(u)} p_\theta(\mathbf{x} \mid u) > 0, \tag{15}$$

and the normalized residual mass is

$$\frac{r_u(a)}{Z_u} = \Pr_\theta(X_t = a \mid X_{1:t-1} = u,\ X \notin \mathcal{S})\,. \tag{16}$$

The "subtract-and-renormalize" operation is hence the original model conditioned on not returning a previously sampled sequence. This is the standard incremental sampling-without-replacement interpretation [36, 54]: after a leaf is sampled, its mass is recursively removed from that leaf and all its ancestors, and the resulting trie represents the model conditioned on that leaf being unavailable.

C.2 Link to logit-tilted sampling

Let $A_u(a) = \sum_{\mathbf{x} \in S(ua)} \mathrm{Adv}_\theta(\mathbf{x})$. After transforming the logits to probabilities, Eq. (10) is equivalent to

$$p_{\theta'}(a \mid u, \mathcal{S}) = \frac{r_u(a) \exp\big(\sigma A_u(a)\big)}{\sum_{b \in \mathcal{V}} r_u(b) \exp\big(\sigma A_u(b)\big)}. \tag{17}$$

This makes the two operations explicit:

1. $r_u(a)$ removes the mass of already sampled sequences and therefore implements sampling without replacement.

2. $\exp(\sigma A_u(a))$ tilts the remaining mass toward prefixes associated with higher reward.

Since $\exp(\sigma A_u(a)) > 0$, we have $p_{\theta'}(a \mid u, \mathcal{S}) = 0$ if and only if $r_u(a) = 0$. The tilt therefore cannot assign mass to a continuation whose completions have all been sampled.

C.3 Connection to the REINVENT Algorithm

REINVENT [45] fine-tunes the model $p_\theta(\mathbf{x})$ by minimizing the squared difference between $\log p_\theta(\mathbf{x})$ and the log-likelihood of an augmented distribution $p_{\text{aug}}(\mathbf{x}) \propto p_{\text{ref}}(\mathbf{x}) \exp\big(\sigma f(\mathbf{x})\big)$, where $\sigma = \beta^{-1}$ is a scaling parameter. For $\mathbf{x} \sim p_\theta(\mathbf{x})$, the gradient of the REINVENT loss coincides with the summand of the Monte Carlo estimator in Eq. (3) up to a constant $c \in \mathbb{R}$, as

$$\begin{aligned}
\mathcal{L}_{\text{REINVENT}}(\theta) &= \Big(\log p_{\text{aug}}(\mathbf{x}) - \log p_\theta(\mathbf{x})\Big)^2 \\
&= \Big(\log p_{\text{ref}}(\mathbf{x}) + \sigma f(\mathbf{x}) - \log p_\theta(\mathbf{x})\Big)^2 \\
&= \Big(\sigma f(\mathbf{x}) - \log \frac{p_\theta(\mathbf{x})}{p_{\text{ref}}(\mathbf{x})}\Big)^2 \\
&= \sigma^2 \Big(f(\mathbf{x}) - \frac{1}{\sigma} \log \frac{p_\theta(\mathbf{x})}{p_{\text{ref}}(\mathbf{x})}\Big)^2
\end{aligned} \tag{18}$$

and

$$\begin{aligned}
\nabla_\theta \mathcal{L}_{\text{REINVENT}}(\theta) &= \nabla_\theta \sigma^2 \Big(f(\mathbf{x}) - \frac{1}{\sigma} \log \frac{p_\theta(\mathbf{x})}{p_{\text{ref}}(\mathbf{x})}\Big)^2 \\
&= c\Big(f(\mathbf{x}) - \beta \log \frac{p_\theta(\mathbf{x})}{p_{\text{ref}}(\mathbf{x})}\Big) \nabla_\theta \log p_\theta(\mathbf{x}),
\end{aligned} \tag{19}$$

with $c = -2\sigma$. As $\mathbf{x}$ is sampled from $p_\theta(\mathbf{x})$, REINVENT-style algorithms inherit the sample inefficiency of Eq. (3), i.e., when high-scoring molecules are rare under $p_\theta(\mathbf{x})$, most terms of the gradient carry little signal about f.

D Algorithm and Implementation of JSI

We provide the full sampling procedure for JSI in the offline setting in Algorithm 1. At each round, JSI draws K candidate sequences using SBS (Line 3), scores them via the predictive head to obtain the expected prediction score μ (Line 5) and the global advantages (Line 9), and biases the next-token probabilities towards higher-scoring continuations by adding the scaled advantage to the logits (Line 11, Eq. (10)). The perturbed logits are then normalized with a softmax to yield the model used in the next round (Line 14). This is repeated for N_{rounds} iterations, progressively steering generation towards high-scoring molecules, and the highest-scoring sampled sequence is returned (Line 16). In the online setting, the predicted score $p_\theta(y \mid \mathbf{x})$ is replaced by oracle evaluations $f(\mathbf{x})$. In practice, the offline predictions and the online evaluations are further aggregated into the rewards $\hat{R}(\mathbf{x})$ defined in Eq. (12) and $R(\mathbf{x})$ in Eq. (11), respectively.

D.1 Numerical Stability in Eq. (10)

We carry out all operations in log space and store subtree probability masses as log masses. Specifically, if x and y are log probabilities with $x \geq y$, we never form the potentially unstable difference $\exp(x) - \exp(y)$ directly. Instead, with x the log of the currently remaining child mass and y the log mass to be removed, we use the numerically stable form (computed with `np.log1p`, in the spirit of the log-sum-exp trick)

$$\text{logsubexp}(x, y) := \log\big(\exp(x) - \exp(y)\big) = x + \log\big(1 - \exp(y - x)\big). \tag{20}$$

The trie update uses three further safeguards. First, exhausted leaves or subtrees are explicitly assigned an unsampled log mass of $-\infty$, which prevents them from being sampled again. Second, for log masses x, y, z,

$$\log\big(\exp(x) - \exp(y) - \exp(z)\big) = \log\big(\exp(\text{logsubexp}(x, y)) - \exp(z)\big), \tag{21}$$

Algorithm 1 Sampling with JSI in the offline setting

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 $n = 1, \dots, N_{\text{rounds}}$ **do**
  $\{\mathbf{x}^{(k)}\}_{k=1}^{K} \leftarrow \text{SBS}(p_\theta(\mathbf{x}), K)$
  $\mathcal{D}_{\text{sampled}} \leftarrow \mathcal{D}_{\text{sampled}} \cup \{\mathbf{x}^{(k)}\}_{k=1}^{K}$
  $\mu \leftarrow \mu\big(\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(K)}\big)$
  $p_{\theta'}(\mathbf{x}) \leftarrow p_\theta(\mathbf{x})$
  for $k = 1, \dots, K$ **do**
    for $t = 1, \dots, T$ **do**
      $A_t^{(k)} \leftarrow \sum_{\mathbf{x} \in S(\mathbf{x}_{1:t}^{(k)})} \Big(p_\theta(y \mid \mathbf{x}) - \mu\Big)$
      $\pi_t^{(k)} \leftarrow \sum_{\mathbf{x} \in S(\mathbf{x}_{1:t}^{(k)})} p_\theta\big(\mathbf{x} \mid \mathbf{x}_{1:t-1}^{(k)}\big)$
      $\text{logit}\ p_{\theta'}\big(x_t^{(k)} \mid \mathbf{x}_{1:t-1}^{(k)}\big) \leftarrow \log\Big(p_\theta\big(x_t^{(k)} \mid \mathbf{x}_{1:t-1}^{(k)}\big) - \pi_t^{(k)}\Big) + \sigma \cdot A_t^{(k)}$ (Eq. (10))
    end for
  end for
  $p_\theta(\mathbf{x}) \leftarrow \text{softmax}\big(\text{logit}\ p_{\theta'}(\mathbf{x})\big)$
end for
return $\arg\max_{\mathbf{x} \in \mathcal{D}_{\text{sampled}}} p_\theta(y \mid \mathbf{x})$

so instead of evaluating the sum of many small linear probabilities in Eq. (10), we incrementally subtract the log mass of each sampled leaf from all its ancestors. Finally, the advantages are added to the residual log masses, after which normalization uses the standard max-shifted exponentiation $\exp(\log a_i - \max_j \log a_j)$, which avoids overflow. An exhausted node is detected by checking that all of its child log masses are $-\infty$, and such a node is not expanded further.

E Additional Experiments

E.1 Reward Normalization and Aggregation

To assess the effect of reward normalization and aggregation, we compare three combinations of normalization and aggregation on the BRAF protein target, using 10 random seeds in the online docking score optimization setting (Section 5.1.1). Let $v(\mathbf{x})$ be the raw value of property $v \in \{\text{DS}, \text{SA}, \text{QED}\}$, and consider Z-score normalization, as in Eq. (11),

$$v^{\text{z}}(\mathbf{x}) = \frac{v(\mathbf{x}) - \mu_v}{\sigma_v}, \tag{22}$$

where μ_v and σ_v are computed from the oracle evaluations available at the current optimization round, as well as the min-max normalization v^{mm} of SATURN [19], which maps each property into $[0, 1]$ and which we take from their implementation. Next, let $s_v(\mathbf{x})$ be the normalized value of property v, so that $s_v(\mathbf{x})$ is either $v^{\text{z}}(\mathbf{x})$ or $v^{\text{mm}}(\mathbf{x})$, oriented so that higher is better, since lower values of DS and SA are preferred, and consider a sum and a product

$$R^{\text{sum}}(\mathbf{x}) = \frac{1}{3} \sum_v s_v(\mathbf{x}), \qquad R^{\text{prod}}(\mathbf{x}) = \prod_v s_v(\mathbf{x}), \tag{23}$$

so that Z-score normalization with sum aggregation recovers the reward in Eq. (11), whereas min-max normalization with product aggregation is the reward of SATURN. Product aggregation requires non-negative scores and is therefore paired with min-max normalization. Z-score normalization with sum aggregation, which is JSI's default, achieves the highest Hit Ratio (Table 8), outperforming both min-max normalization with sum aggregation and min-max normalization with product aggregation.

Table 8: Hit Ratio for different reward normalization and aggregation methods in the online docking score optimization setting. We mark the best configuration **bold**. Values are mean (standard deviation) across 10 random seeds.

		Target Protein, Hit Ratio (↑)
Normalization	Aggregation	braf
Z-scoring	Sum	**58.547 (11.456)**
Min-max	Sum	23.223 (6.426)
Min-max	Product	38.897 (7.024)

E.1.1 Weighted Reward

To assess the effect of the relative weighting of individual properties, we generalize the reward in Eq. (11) to the weighted reward

$$R_{\mathbf{w}}(\mathbf{x}) = w_{\mathrm{DS}}\big(-\mathrm{DS}^{\mathrm{z}}(\mathbf{x})\big) + w_{\mathrm{SA}}\big(-\mathrm{SA}^{\mathrm{z}}(\mathbf{x})\big) + w_{\mathrm{QED}}\,\mathrm{QED}^{\mathrm{z}}(\mathbf{x}), \tag{24}$$

i.e. Z-score normalization with sum aggregation (Eq. (23)), where $\mathbf{w} = (w_{\mathrm{DS}}, w_{\mathrm{SA}}, w_{\mathrm{QED}})$, with $w_i \geq 0$ and $\sum_i w_i = 1$, so that the uniform weighting $\mathbf{w} = (1/3, 1/3, 1/3)$ recovers Eq. (11). We vary $\mathbf{w}$ on all five protein targets in the online docking score optimization setting (Section 5.1.1), using 10 random seeds (Table 9). Downweighting the docking score to 0.25 lowers the Hit Ratio on all five targets relative to uniform weighting. Upweighting the docking score to 0.5 improves the Hit Ratio on four of the five targets, namely PARP1, FA7, BRAF and JAK2, with 5HT1B the only exception. Increasing the docking-score weight further to 0.75 does not help: the Hit Ratio falls below that of the 0.5 setting on every target and is comparable to or below that of uniform weighting on all targets except FA7. These results indicate that moderately increasing the relative weight of the docking score can improve performance on most targets.

Table 9: Hit Ratio for different weightings $\mathbf{w}$ of the reward $R_{\mathbf{w}}$ (Eq. (24)) in the online docking score optimization setting. We mark the best weighting per target **bold**. Values are mean (standard deviation) across 10 random seeds.

	Target Protein, Hit Ratio (↑)				
w_{DS} / w_{SA} / w_{QED}	parp1	fa7	5ht1b	braf	jak2
0.25 / 0.375 / 0.375	65.163 (7.858)	34.350 (11.492)	87.827 (3.689)	49.883 (10.589)	68.553 (12.792)
0.33 / 0.33 / 0.33	67.280 (7.961)	36.587 (10.270)	**91.357 (3.204)**	58.547 (11.456)	71.863 (6.127)
0.5 / 0.25 / 0.25	**75.470 (6.973)**	**43.530 (11.954)**	84.413 (6.580)	**64.997 (7.474)**	**80.230 (4.915)**
0.75 / 0.125 / 0.125	68.913 (9.576)	41.317 (10.804)	78.410 (4.293)	59.570 (9.114)	67.867 (5.982)

E.2 Surrogate Accuracy

To evaluate the accuracy of the predictive part of the joint model, we compare its predictions against ground-truth oracle docking scores across all protein targets, using the Pearson correlation, on both held-out sequences (Test) and sequences produced in the offline docking score optimization setting (Generated), reported in Section 5.1.2. The held-out dataset consists of 500 molecules randomly selected from ZINC250K and kept fixed across random seeds and protein targets, for which we compute the corresponding docking score, SA and QED for each target. We report the mean and standard deviation calculated across 10 random seeds (Table 10).

On the held-out sequences, the surrogate maintains a strong positive correlation with the oracle values on all five targets, confirming its strong predictive performance. On the generated sequences, the correlation remains positive, although with a markedly higher variance across seeds, indicating that the surrogate retains sufficient signal to prioritize high-scoring candidates during optimization. A lower correlation is expected here, as JSI generates high-scoring molecules that may be increasingly distinct from the offline dataset.

Table 10: Surrogate predictive performance on held-out (Test) and generated (Generated) sequences in the offline docking score optimization setting, measured by the Pearson correlation with the ground-truth oracle docking score. Values are mean (standard deviation) across 10 random seeds.

	Target Protein, Pearson Correlation (↑)				
Sequences	parp1	fa7	5ht1b	braf	jak2
Test	0.879 (0.005)	0.762 (0.012)	0.867 (0.007)	0.802 (0.010)	0.832 (0.006)
Generated	0.224 (0.190)	0.284 (0.150)	0.305 (0.194)	0.304 (0.141)	0.384 (0.122)

E.3 Hyperparameter Sensitivity

To assess the sensitivity of JSI to its hyperparameters, we vary the step size σ, the beam width K and the joint-loss weight λ on the BRAF protein target in the offline docking score optimization setting (Section 5.1.2), using 3 random seeds. Across all values of σ and K, the Hit Ratio stays within one standard deviation of the default configuration ($\sigma = 1$, $K = 128$, $\lambda = 0.25$), indicating robustness to the exact choice of these hyperparameters (Table 11). Moreover, the Hit Ratio increases monotonically with the beam width (Table 12). Finally, values of $\lambda \geq 0.4$ degrade performance (Table 13).

Table 11: Sensitivity analysis for the step size σ in the offline docking score optimization setting. Values are mean (standard deviation) across 3 random seeds.

	Target Protein, Hit Ratio (↑)
Step Size σ	braf
0.25	33.829 (14.137)
0.5	29.907 (10.995)
0.75	27.402 (7.025)
1	27.418 (9.888)

Table 12: Sensitivity analysis for the beam width K in the offline docking score optimization setting. Values are mean (standard deviation) across 3 random seeds.

	Target Protein, Hit Ratio (↑)
Beam Width K	braf
16	18.414 (3.963)
32	21.275 (9.278)
64	21.857 (7.136)
128	27.418 (9.888)

E.4 Novelty and Diversity for AMP Sequences

To assess the novelty and diversity of the generated AMP sequences, we extend the evaluation of the AMP sequences generated in Table 4 with three metrics computed using the seqme benchmarking library [47]. We measure **Uniqueness**, defined as the fraction of generated sequences that are unique; **Novelty**, defined as the fraction of generated sequences that do not appear in the training data; and the **3-gram Jaccard similarity**, defined as the average Jaccard similarity between the 3-grams of each generated sequence and a reference corpus constructed from the 3-grams of the training data. In addition, we report **Diversity**, as defined in Appendix F.4.1. We evaluate the best-performing methods, namely AMP-Diffusion, Hyformer and JSI, on the same 1,000 generated sequences per method as in Section 5.2.

JSI generates exclusively unique sequences and attains a diversity between that of the two baselines, while its 3-gram composition remains almost disjoint from the training data (Table 14). Its novelty is slightly lower than that of the baselines, indicating that a small fraction of the generated sequences is recovered from the training data.

Table 13: Sensitivity analysis for the joint-loss weight λ in the offline docking score optimization setting. Values are mean (standard deviation) across 3 random seeds.

	Target Protein, Hit Ratio (↑)
Joint-Loss Weight λ	braf
0.1	24.410 (8.217)
0.25	27.418 (9.888)
0.4	18.867 (2.780)
0.5	16.564 (2.297)

Table 14: Uniqueness, novelty and diversity of the AMP sequences generated in the offline AMP optimization setting. Values are computed on the 1,000 sequences per method evaluated in Table 4.

Method	Uniqueness (↑)	Novelty (↑)	3-gram Jaccard Similarity	Diversity (↑)
AMP-Diffusion	0.960	1.000	0.003	0.820
Hyformer	1.000	0.999	0.002	0.740
JSI (ours)	1.000	0.916	0.004	0.780

E.5 Reduced Evaluation Budget and Offline Dataset Size

To examine whether joint modeling remains applicable under the tighter constraints of real-world applications, we extend the AMP optimization experiment (Section 5.2) with two settings. First, we reduce the evaluation budget below the 1,000 sequences of the main experiment (Table 15). Second, we reduce the number of labeled sequences available for fine-tuning (Table 16). In both settings we report the metrics defined in Appendix F.4.1.

JSI maintains strong performance across all evaluation budgets, and across all offline dataset sizes except the smallest, where the scores on all three AMP oracles degrade, most markedly on $\text{HydrAMP}_{\text{MIC}}$, the most challenging oracle.

Table 15: Performance of JSI on offline AMP optimization under a reduced evaluation budget. Values are mean (95% confidence interval).

Evaluation Budget	Perplexity	Diversity (↑)	Fitness (↑)	$\text{HydrAMP}_{\text{MIC}}$ (↑)	AMPlify (↑)	AmPEPpy (↑)
250	16.38 (15.97, 16.83)	0.76 (0.74, 0.77)	0.13 (0.12, 0.14)	0.90 (0.85, 0.93)	0.98 (0.97, 0.99)	0.78 (0.77, 0.79)
500	15.99 (15.64, 16.35)	0.77 (0.76, 0.77)	0.13 (0.12, 0.13)	0.86 (0.83, 0.89)	0.98 (0.97, 0.99)	0.78 (0.77, 0.79)
750	16.08 (15.81, 16.33)	0.78 (0.77, 0.78)	0.12 (0.11, 0.13)	0.87 (0.84, 0.89)	0.98 (0.98, 0.99)	0.78 (0.77, 0.79)

Table 16: Performance of JSI on offline AMP optimization under a reduced offline dataset size. Values are mean (95% confidence interval).

Offline Dataset Size	Perplexity	Diversity (↑)	Fitness (↑)	$\text{HydrAMP}_{\text{MIC}}$ (↑)	AMPlify (↑)	AmPEPpy (↑)
250	17.58 (17.26, 17.91)	0.79 (0.78, 0.79)	0.15 (0.14, 0.15)	0.73 (0.70, 0.75)	0.91 (0.90, 0.92)	0.70 (0.69, 0.71)
500	15.41 (15.14, 15.65)	0.70 (0.69, 0.71)	0.20 (0.19, 0.21)	0.92 (0.90, 0.93)	0.98 (0.98, 0.99)	0.72 (0.72, 0.73)
750	13.79 (13.53, 14.08)	0.76 (0.76, 0.76)	0.19 (0.18, 0.20)	0.91 (0.89, 0.92)	0.98 (0.97, 0.98)	0.73 (0.73, 0.74)

E.6 Additional Evaluation Metrics for Online Molecular Optimization

To characterize the generated molecules beyond the Hit Ratio, we report validity, novelty, scaffold novelty, nearest-neighbor similarity and the Fréchet ChemNet Distance (FCD), measured with respect to the ZINC250K training set, for the best-performing methods on the BRAF protein target in the online docking score optimization setting (Section 5.1.1), using 10 random seeds. We define these metrics as follows.

Validity. The fraction of generated SMILES strings that RDKit [60] successfully parses into a valid molecule.

Novelty. The fraction of generated molecules whose canonical SMILES do not appear in the training set.

Scaffold novelty. The fraction of generated molecules whose Bemis–Murcko scaffold [4] does not appear in the training set.

Nearest-neighbor similarity. For each generated molecule, the maximum Tanimoto similarity to the training set, computed over Morgan fingerprints [62] with radius 2 and 2048 bits, averaged across generated molecules.

FCD. The Fréchet distance between the ChemNet activation distributions of the generated and training sets [56]. Lower values indicate that the generated molecules lie closer to the training distribution in chemical and bioactivity space.

All methods generate almost exclusively valid and novel molecules (Table 17). JSI attains scaffold novelty and nearest-neighbor similarity comparable to SATURN, the strongest baseline in terms of Hit Ratio, whereas GEAM generates the molecules most dissimilar from the training set. The FCD of JSI is the highest, indicating that it generates molecules further from the training distribution in chemical and bioactivity space.

Table 17: Quality of the molecules generated for the BRAF protein target in the online docking score optimization setting, measured with respect to the ZINC250K training set. GEAM and NF-BO produce valid molecules by design. Values are mean (standard deviation) across 10 random seeds.

METHOD	VALIDITY (↑)	NOVELTY (↑)	SCAFFOLD NOVELTY (↑)	NEAREST NEIGHBOR SIMILARITY	FCD
GEAM	1.000 (0.000)	1.000 (0.000)	0.990 (0.003)	0.316 (0.007)	16.276 (0.609)
SATURN	0.979 (0.004)	0.999 (0.000)	0.931 (0.019)	0.526 (0.020)	27.653 (3.132)
NF-BO	1.000 (0.000)	0.999 (0.001)	0.833 (0.020)	0.502 (0.014)	14.312 (1.589)
JSI (OURS)	0.969 (0.008)	0.999 (0.001)	0.888 (0.079)	0.542 (0.033)	31.376 (7.209)

E.7 Sampling Time

To quantify the computational cost of inference-time self-improvement, we measure **Time** (s), defined as the number of seconds required to sample a single solution. Additionally, we report **#Candidates**, the number of sequences generated per solution, and **Length**, the mean number of generated tokens per solution. We evaluate each sampling method on the BRAF protein target in the offline docking score optimization setting (Section 5.1.2) by sampling 100 solutions. Each method samples from the same jointly fine-tuned backbone. Unconditional sampling autoregressively generates a single sequence; Best-of-K unconditionally generates K sequences and returns as the solution the one with the highest prediction score, computed with the predictive part of the model; JSI returns a single best-scoring solution, as detailed in Algorithm 1, using N_{rounds} rounds of sampling without replacement, each round producing K candidates. Note that JSI with $N_{\text{rounds}} = 1$ is equivalent to JSI W/O INFERENCE-TIME SELF-IMPROVEMENT in Section 5.3. All runs use a single NVIDIA H100 80GB GPU, and we report the mean and standard deviation across 3 random seeds, each paired with a distinct offline dataset.

Table 18: Sampling cost of the inference-time methods on the BRAF protein target in the offline docking score optimization setting. Values are mean (standard deviation) across 3 random seeds.

METHOD	#CANDIDATES	LENGTH	TIME(S)
UNCONDITIONAL	1	39.4	0.004 (0.001)
BEST-OF-K	128	39.6	0.552 (0.014)
JSI			
$N_{\text{rounds}} = 1$	128	39.7	0.662 (0.045)
$N_{\text{rounds}} = 2$	256	40.2	1.321 (0.068)
$N_{\text{rounds}} = 3$	384	41.0	2.041 (0.195)
$N_{\text{rounds}} = 4$	512	41.6	2.708 (0.154)
$N_{\text{rounds}} = 5$	640	42.3	3.418 (0.312)
$N_{\text{rounds}} = 6$	768	43.1	4.081 (0.413)
$N_{\text{rounds}} = 7$	896	44.0	4.741 (0.505)
$N_{\text{rounds}} = 8$	1024	44.7	5.440 (0.620)

F Experimental Details

We conducted all experiments across the Helmholtz Munich and Jülich HPC clusters, using a single NVIDIA V100, A100 or H100 GPU, with 16 CPU cores, 16 GB of RAM, and a maximum wall time of 24 hours per job.

F.1 Docking Score Optimization

To evaluate our framework under realistic sample-efficiency constraints, we adopt the experimental set-up of [40, 18, 19, 65] with an oracle budget of 3,000 evaluations, since 10,000 evaluations [14] may exceed the budgets typically available in real-world drug discovery workflows. We formulate docking score optimization as a constrained optimization problem, consistent with prior work [40, 18, 19], to generate molecules that exceed a predefined docking threshold while satisfying chemical plausibility constraints, as quantified by SA and QED. In this formulation, docking takes precedence over auxiliary properties: once the docking threshold is met, further gains in SA or QED are secondary, whereas molecules that fail to meet the threshold are rejected regardless of their auxiliary scores. Consequently, Pareto-based multi-objective optimization metrics are not well aligned with our setting, and following prior work, we use Hit Ratio, which directly measures the fraction of molecules that satisfy the docking threshold and auxiliary constraints, as our primary evaluation metric. We provide an overview of the different molecular optimization settings in Figure 5.

JSI Backbone We use Hyformer [25] as the backbone for JSI. To ensure comparability with baseline methods, we pre-train Hyformer on the ZINC250K dataset, scaled to 27M parameters with 8 transformer layers and embedding dimensionality 512. We optimize with AdamW and a cosine learning-rate schedule with linear warmup for 20 epochs, and select the checkpoint with the lowest validation loss. Table 19 summarizes the pre-training hyperparameters. In contrast to Izdebski et al. [25], we pre-train without precomputed molecular descriptors, so that pre-training does not rely on proxies for the SA and QED feasibility constraints.

Docking Score Computation For a fair comparison with prior work, we follow the experimental setup of Lee et al. [40] and use QuickVina 2 [2] as the docking oracle in both the online and offline settings. To speed up oracle calls, we use its GPU implementation, QuickVina2-GPU 2.1. We dock against five protein targets (fa7, parp1, 5ht1b, jak2 and braf), using the same receptor structures and docking-box centers and sizes as Lee et al. [40]. Each ligand is prepared from its canonical SMILES: we add hydrogens, generate a single conformer with RDKit ETKDG, minimize it with the UFF force field, and convert it to PDBQT with OpenBabel. Docking runs with the default search effort of 5000 threads and a fixed random seed. We take the score of the lowest-energy pose as the docking score, in kcal/mol, where lower is better. Molecules that fail conformer generation are assigned a score of 0.

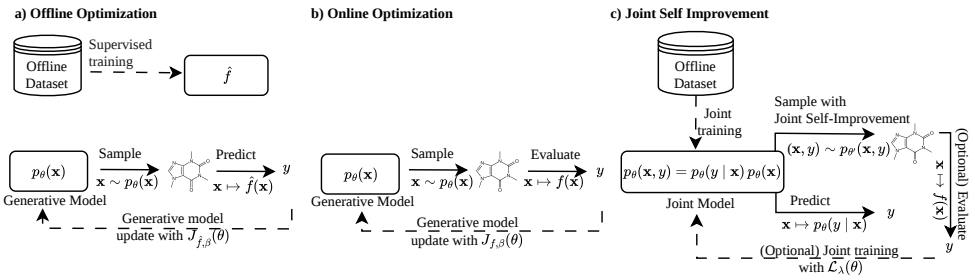

Figure 5: Offline and online GMO as compared to the JSI framework. (a) In the offline setting, objective f is not available, and a surrogate $\hat{f}$ is used to predict its values. (b) In the online setting, objective f is available and is used for generative model updates. (c) JOINT SELF-IMPROVEMENT can perform in both settings: either using the offline dataset to train the joint model and sample new molecules, or updating based on newly generated objective evaluations.

Table 19: Hyperparameter settings used for pre-training and fine-tuning of Hyformer for docking optimization task.

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
$(p_{[\text{LM}]}, p_{[\text{MLM}]}, p_{[\text{PRED}]})$	(0.9, 0.1, 0.0)	(0.75, 0.0, 0.25)

Table 20: 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

F.2 Sample-Efficient Online Molecular Optimization

All results in Table 1 are from [19], except for NF-BO and JSI, which are ours.

NF-BO We run NF-BO [41] ourselves, evaluating it under the same protocol applied to all other methods: the same five protein targets, the same docking oracle, and a total budget of 3,000 oracle evaluations. The generative backbone is the sequence-level autoregressive normalizing flow, with a hidden size of 500, a latent dimension of 40, 5 flow layers, 2 recurrent layers, dropout 0.2, and the non-linear squared (NLSq) transform; we initialize from the authors checkpoint pre-trained on 1.27M molecules rather than pre-training from scratch. The surrogate is a deep-kernel Gaussian process, fit for 100 epochs at initialization and 1 epoch per subsequent update, with a learning rate of $1 \times 10^{-}3$ and a batch size of 32; the flow is updated end-to-end jointly with the surrogate for 1 epoch per update at the same learning rate. Each run is initialized from a fixed set of 100 pre-scored molecules, so that 2,900 of the 3,000 evaluations are spent on molecules proposed during optimization, and an end-to-end update of the flow and surrogate is triggered approximately every 200 oracle evaluations. Candidates are acquired by Thompson sampling from the 5 highest-scoring molecules in the current dataset, drawing a batch of 10 candidates from each and capping the candidate pool at 2,000, which yields 50 molecules scored by the oracle per round. We use a sampling temperature of 0.01 and a DACS temperature of 400. Each target is repeated over 10 random seeds, and we report the mean and standard deviation across runs.

JSI We use Hyformer as the generative model, pre-trained on ZINC250K, to sample molecules using inference-time self-improvement. Oracle queries are employed to compute the reward. The generator is then fine-tuned using the top rewarded molecules. Following [19], 5 rounds of SMILES augmentation is applied prior to fine-tuning. We sample 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 20.

F.3 Sample-Efficient Offline Molecular Optimization

All results in Table 3 are ours. In Figure 4, docking poses are visualized using PyMOL.

GraphGA We use the same Hyformer backbone as JSI for candidate generation and surrogate scoring, replacing inference-time self-improvement with graph-based genetic search. At each optimization round, we initialize the population with the highest-scoring generated molecules and apply rank-based parent selection, crossover, and mutation with probability 0.01.

REINVENT 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 and 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 [65].

SATURN 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 pre-trained on ZINC250K, using a publicly available checkpoint from the SATURN repository `https://github.com/schwallergroup/saturn`. Following [19], we use a learning rate of 1×10^{-4}, an augmented-likelihood scaling of 128.0 (denoted σ by Guo et al. [19], and distinct from the JSI step size σ), 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 19 and employ the resulting model as a surrogate predictor throughout optimization. Following Shin et al. [65], 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 reweighted 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 19). The best-scoring sequences in the offline dataset, i.e., those with the lowest docking scores, 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.

JSI We jointly fine-tune Hyformer on the offline dataset with the hyperparameters listed in Table 19. Crucially, we fine-tune on Z-score normalized property values, with the mean and standard deviation of each property computed from the offline dataset, so that the prediction head directly outputs normalized values. Next, we define the objective as the mean of the predicted normalized properties (Eq. (12)), where DS and SA enter with a negative sign since lower values are better. The advantage of a sampled molecule is its reward minus the expected objective over the current beam, min-max normalized to $[-1, 1]$ separately for positive and negative advantages, so that the expected objective maps to 0. 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 $\sigma \in \{0.25, 0.5, 0.75, 1.0, 1.25, 1.5\}$. The best hyperparameters are summarized in Table 21. We note that the best hyperparameter settings are fairly consistent across different protein targets, indicating their transferability across tasks in the offline setting. To ensure numerical correctness and efficiency of our sampling algorithm, in particular numerical stability in Eq. (10), we follow the augmented trie implementation of Shi et al. [64].

Table 21: 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

F.4 Sample-Efficient Offline AMP Optimization

All results in Table 4 and Table 5 are ours. For all baseline methods, except for JSI and Hyformer, we randomly subsample 1,000 sequences from the full set of generated peptides (50,000 sequences per method) provided by [8], in order to match a fixed evaluation budget of 1,000 sequences across methods.

Hyformer Peptide sequences are generated using rejection sampling following [25]. Specifically, sequences are sampled from the generative component and accepted using the predictive component with a threshold of $\text{MIC} \leq 10^{2.0}$ until 1,000 sequences are collected. No additional retraining is performed during generation.

JSI We use Hyformer [25] as the backbone model for all experiments. The model is pre-trained on 3.5M peptide sequences and subsequently on an additional 1M AMP sequences, using the hyperparameters listed in Table 22. The offline dataset consists of 3,444 peptides with associated MIC values against *E. coli*, following [69]. We jointly fine-tune the model on the offline dataset using the joint loss $\mathcal{L}_\lambda$ with $\lambda = 0.4$ (Equation (6)).

We perform inference-time self-improvement by biasing the generative component using the predictive component as a reward, defined as $R(x) = -\hat{\text{MIC}}(x)$. We iteratively sample peptide sequences using JSI until 1,000 sequences are collected, without further retraining. We use a beam width of 32, the number of self-improvement rounds $N_{\text{rounds}} = 8$, step size $\sigma = 1.0$ and with temperature 0.9.

For Table 5, we additionally evaluate a more challenging setting where the offline dataset is reconstructed based on activity labels defined in prior work [69], resulting in datasets containing only 1% and 0% active peptides. In both settings, we fine-tune the Hyformer backbone using the same hyperparameters listed in Table 22. Hyformer uses rejection sampling with the same MIC threshold, while JSI performs inference-time self-improvement without retraining.

We estimate uncertainty using bootstrapping with 1,000 resampling iterations with replacement over the selected 1,000 sequences. For each metric, we report the mean computed on the full set of sequences and construct 95% confidence intervals using the 2.5th and 97.5th percentiles of the bootstrap distribution. For the diversity metric, we apply the same procedure by resampling sequences and recomputing the diversity score for each bootstrap sample.

Table 22: Hyperparameter settings used for pre-training and fine-tuning of Hyformer for AMP optimization task.

HYPERPARAMETER	PRETRAINING	FINETUNING
BATCH SIZE	512	64
LEARNING RATE	4E-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
$(p_{[\text{LM}]}, p_{[\text{MLM}]}, p_{[\text{PRED}]})$	(0.9, 0.1, 0.0)	(0.6, 0.0, 0.4)

F.4.1 Benchmark metrics

Following [25], we use six metrics to evaluate the generated AMPs.

Perplexity Exponentiated token-level negative log-likelihood of a sequence, computed using ProGen2-medium [50], following Torres et al. [72]; lower values indicate higher plausibility of the generated peptides under the model.

Diversity The average pairwise Levenshtein distance among generated sequences, where higher values reflect increased diversity. We follow Eq. 6 in [32], replacing the Soergel distance with Levenshtein distance in the Hyformer setting.

Fitness A quantitative measure of a peptide's tendency to form a stable, amphipathic α-helical structure, as defined in [44].

HydrAMP$_{\text{MIC}}$ The predicted probability that a peptide is active against *E. coli*, computed using HydrAMP [69].

AMPlify The predicted probability that a peptide is an AMP, computed using AMPlify [43].

AmPEPpy The predicted probability that a peptide is an AMP, computed using AmPEPpy [38].

F.5 Ablations

All results in Table 6 are ours.

JSI W/O JOINT MODELING We train Chemprop [20] as the surrogate predictor on the offline dataset with default hyperparameters [20], including a batch size of 64, a maximum learning rate of 10^{-3}, and up to 30 epochs with early stopping. For sampling, we fine-tune the pre-trained Hyformer on the offline dataset using the generative-only part of the joint loss, $\mathcal{L}_{\lambda=0}$ (Eq. (6)), so that the comparison isolates the effect of joint modeling, with the best hyperparameters from Appendix F.2. To verify that Chemprop provides a competitive surrogate, we evaluate predictive performance on a held-out test set of 500 molecules randomly sampled from ZINC250K (Table 23).

JSI W/O INFERENCE-TIME SELF-IMPROVEMENT We replace the joint self-improving sampling scheme with Best-of-K sampling [25], where for every molecule we sample K candidates using the generative part of the model and select the one with the best predicted reward. Crucially, we draw the K candidates efficiently in a single round of stochastic beam search, which yields K distinct sequences without replacement while sharing computation across common prefixes, and we apply no advantage-based biasing of the generative model, i.e. $N_{\text{rounds}} = 1$. Predictor-based selection of the best candidate, which JSI also performs (Step 16 in Algorithm 1), is retained, so that the comparison isolates the effect of inference-time self-improvement. We keep the beam width K selected for JSI on each target, i.e. $K = 128$ for parp1, fa7, braf and jak2, and $K = 16$ for 5HT1B, and leave all remaining hyperparameters unchanged.

Table 23: Predictive performance of Chemprop and JSI on a held-out test set of 500 molecules sampled from ZINC250K in docking score prediction, as measured by mean absolute error (MAE).

	TARGET PROTEIN, MAE (↓)				
SURROGATE	PARP1	FA7	5HT1B	BRAF	JAK2
CHEMPROP	0.517 (0.012)	0.383 (0.005)	0.528 (0.007)	0.464 (0.006)	0.439 (0.013)
JSI	0.446 (0.013)	0.400 (0.012)	0.426 (0.015)	0.466 (0.012)	0.408 (0.007)

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 evaluate all the claims via experimental evaluation in Section 5 and in Appendix C.3

   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: We discuss limitations in Section 6. Moreover, we discuss the running time of our algorithm in paragraph **Practical Implementation** and in ablations.

   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 contain novel 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 all necessary details for reproducing all the results in Appendix F.

   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: The repository with scripts reproducing the results is hosted on GitHub and will be made publicly available upon acceptance.

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: We provide all necessary details for reproducing all the results in Appendix F.

   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: Across all experiments, we report mean and standard deviation over 10 random seeds, or bootstrapped confidence intervals.

   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: We provide all necessary details for reproducing all the results in Appendix F.

   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 conducted in the paper conforms 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: [Yes]

    Justification: We address the potential negative societal impacts.

    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 does not release new pretrained checkpoints, nor data.

    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: [N/A]

    Justification: No protected assets were used.

    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: [N/A]

    Justification: The paper does not release new assets.

    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: LLMs were used only for writing, editing, or formatting purposes.

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.