SELF-IMPROVING BAYESIAN FLOW NETWORKS VIA GROUP RELATIVE POLICY OPTIMIZATION FOR STRUCTURE-BASED DRUG DESIGN

Anonymous authors
Paper under double-blind review

ABSTRACT

Structure-based drug design (SBDD) requires molecules that bind favorably to a protein pocket while maintaining plausible conformations and suitable chemical properties. Generative models learn from protein–ligand complexes, but fitting the training distribution alone does not directly optimize these properties. On-policy reinforcement learning offers a way for generators to learn from evaluations of their own candidates. Applying it to Bayesian flow networks (BFNs) requires identifying the stochastic decisions whose likelihoods support policy updates. In the molecular BFN sampler considered here, the network predicts clean coordinates and atom-type probabilities, while the sampler draws new coordinate-belief means and atom-type logits. We introduce GRAB (Group-Relative Alignment of Bayesian flow networks), a self-improving framework that defines these sampled quantities as a joint policy action. For this Gaussian sampler, we derive a closed-form conditional action density that is differentiable through the network predictions, with covariance fixed by the sampling schedules. Terminal rewards from docking affinity, strain energy, drug-likeness (QED), and synthetic accessibility (SA) support Group Relative Policy Optimization (GRPO), without a learned value model or gradients through reconstruction or scoring. Generation retains the original sampler without additional property-gradient guidance. On 100 CrossDocked2020 test pockets, one GRAB configuration achieves the lowest mean Vina Dock and median strain energy among the generators in the main comparison, while maintaining favorable drug-likeness, synthetic accessibility, and ligand diversity.

1 INTRODUCTION

Generative models for structure-based drug design (SBDD) learn to propose novel ligands from protein–ligand complexes (Peng et al., 2022; Guan et al., 2023a), yet fitting the training distribution does not directly optimize the molecular properties that determine whether a candidate is useful, such as binding affinity, conformational stability, drug-likeness, or synthetic accessibility. Evaluations of generated molecules against these properties therefore provide a natural source of feedback for improving a pretrained generator beyond its original training objective.

Bayesian flow networks (BFNs) offer an appealing foundation for this improvement. BFNs iteratively refine distributions representing beliefs about the data (Graves et al., 2023). For molecules, this allows continuous coordinates and discrete atom types to be represented through distribution parameters that are jointly refined in a continuous space. MolCRAFT uses this approach for pocket-conditioned ligand generation, reporting favorable binding scores and lower conformational strain than the diffusion baselines evaluated in that study (Qu et al., 2024). Its stochastic sampler updates these beliefs through a sequence of network-guided steps, suggesting a sequential decision formulation for learning from downstream molecular scores.

Existing approaches to property optimization in molecular BFNs include guiding generation with gradients from learned property predictors (Qiu et al., 2025) and fine-tuning from reward-derived preferences between molecules (Lee et al., 2025). In this paper, we pursue an alternative: on-policy reinforcement learning, in which the current generator produces candidates, their evaluated properties guide parameter updates, and the updated generator supplies the next round of candidates. This

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

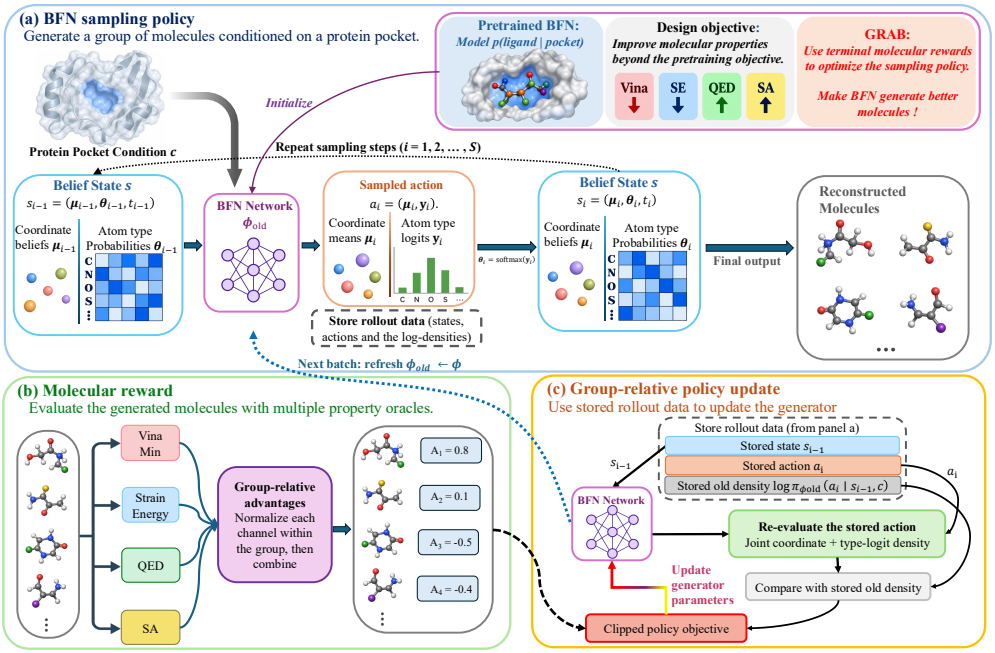


Figure 1: **Overview of GRAB's self-improvement loop. (a)** A fixed snapshot ϕ_{old} generates groups of molecules sharing a protein pocket and atom count, storing actions $a_i = (\boldsymbol{\mu}_i, \mathbf{y}_i)$ comprising sampled coordinate belief means and atom-type logits. **(b)** Rewards from Vina Min, strain energy, QED, and SA are normalized per channel within each group and combined into an advantage A_j for each molecule j. **(c)** The current generator ϕ evaluates joint likelihoods of stored actions for an advantage-weighted clipped policy update. Rollout data and rewards remain fixed during each update batch. After each update batch, $\phi_{\text{old}} \leftarrow \phi$, and fresh trajectories are collected.

self-improvement loop avoids differentiating through molecular reconstruction or property evaluation, but it requires a policy formulation that assigns likelihoods to the stochastic sampling steps that produced each completed molecule.

The key challenge is to identify which stochastic decisions molecular rewards should reinforce. In the BFN sampler considered here, the network predicts molecular coordinates and atom-type probabilities, but the sampler does not directly select atoms from these predictions. Instead, it randomly updates distributions describing the molecule. The network's output probabilities therefore cannot simply be used as action likelihoods for policy optimization. We formulate a policy over the sampled updates to both coordinate and atom-type beliefs, deriving their joint conditional density from the existing sampler. This connects completed-molecule rewards to the stochastic decisions that produced them, enabling policy updates without changing the pretrained sampling procedure.

To address this challenge, we introduce GRAB (Group-Relative Alignment of Bayesian flow networks), a self-improving framework that defines these sampled belief updates as policy actions and derives their joint conditional densities. GRAB uses Group Relative Policy Optimization (GRPO) to compare candidates sharing a pocket and atom count, obtaining training feedback without a learned value model (Shao et al., 2024). Scores of completed molecules guide policy updates, and the updated generator supplies the next round of candidates. We specify the conditions under which this formulation applies to stochastic BFNs and instantiate GRAB with MolCRAFT to optimize binding affinity, conformational strain, drug-likeness, and synthetic accessibility. Generation retains the original sampler without additional property-gradient guidance.

Our contributions are as follows:

- **A policy formulation for stochastic BFNs.** We define joint coordinate and atom-type sampling actions, derive their conditional densities for Gaussian samplers, and specify the conditions supporting policy-gradient updates.

- **Self-improvement from molecular rewards.** We connect terminal property evaluations to repeated group-relative updates of the generator, without gradients through reconstruction or scoring and without additional guidance during generation.
- **Joint gains in docking and molecular properties.** On the 100 CrossDocked2020 test pockets, our proposed method achieves the lowest mean Vina Dock and median strain energy among the compared generators, alongside higher QED, SA, and ligand diversity than reported MolCRAFT results.

2 Related Work

Generative models for structure-based drug design. Pocket-conditioned generators include autoregressive models such as Pocket2Mol (Peng et al., 2022), diffusion models such as TargetDiff and DecompDiff (Guan et al., 2023a;b), and flow-matching models such as MolFORM (Huang & Zhang, 2025) and PAFlow (Zhou et al., 2025). Property optimization can update model parameters or guide sampling. AliDiff (Gu et al., 2024) and the initial MolFORM version use preference-based fine-tuning, while BoKDiff fine-tunes on reward-selected samples (Khodabandeh Yalabadi et al., 2025). A later MolFORM version introduces online reward-weighted training (Zhang & Zhang, 2026). PAFlow guides sampling with a learned protein–ligand interaction predictor.

Bayesian flow networks. BFNs generate data by iteratively updating distributional beliefs (Graves et al., 2023). Applications include equivariant 3D molecule generation with GeoBFN (Song et al., 2024) and protein sequence generation with ProtBFN and its antibody-specific variant AbBFN (Atkinson et al., 2025). For pocket-conditioned ligand generation, MolCRAFT jointly models coordinates and atom types (Qu et al., 2024). Subsequent work introduces property optimization: MolJO uses gradients of learned predictors during sampling (Qiu et al., 2025), whereas BFN-DPO fine-tunes molecular BFNs using preferences derived from multiple rewards (Lee et al., 2025).

Policy optimization for generative models. Denoising diffusion policy optimization (DDPO) formulates diffusion sampling as a sequential decision process and optimizes terminal rewards through policy gradients (Black et al., 2024). GRPO obtains relative advantages from groups of samples without a learned value model (Shao et al., 2024). Flow-GRPO extends this approach to text-to-image flow matching by introducing stochastic sampling (Liu et al., 2025a). For multiple rewards, MO-GRPO and GDPO study normalizing individual reward channels before aggregation (Ichihara et al., 2026; Liu et al., 2026). GRAB formulates group-relative policy updates over the sampled belief transitions of stochastic BFNs.

3 Preliminaries

3.1 Problem Formulation

To allow pretrained generative models to self-learn and generate molecules with desirable properties, we study a self-improvement process that leverages reinforcement learning. The process starts with the pre-trained model first generating candidate molecules, and the candidates will have their molecular properties evaluated. The evaluation scores (reward) will guide the model updates, followed by another round of candidate generation. The process iterates for a fixed training budget.

Formally, for a protein pocket c, a conditional generator $p_\phi(m \mid c, N)$ produces a ligand $m = (\mathbf{x}, \mathbf{v})$. Here $\mathbf{x} \in \mathbb{R}^{N\times 3}$ contains N heavy-atom coordinates and $\mathbf{v} \in \{1, \dots, K\}^N$ specifies their types from a K-class vocabulary. We draw $N \sim q(\cdot \mid c)$, where $q(N \mid c)$ is the pretrained sampler's pocket-conditioned atom-count prior, held fixed throughout self-improvement.

Let $\mathcal{D}$ denote the distribution of training pockets used for rollouts. For a fixed scalar molecular reward $R(m, c)$, we aim to maximize:

$$\max_{\phi} \; J(\phi) \;=\; \mathbb{E}_{c\sim\mathcal{D}}\, \mathbb{E}_{N\sim q(\cdot|c)}\, \mathbb{E}_{m\sim p_\phi(\cdot|c,N)} \big[\, R(m, c) \,\big], \tag{1}$$

with ϕ initialized to the pretrained parameters ϕ_0. Rewards are assigned after molecular reconstruction, since the evaluators score completed molecules rather than intermediate beliefs; intermediate sampling steps receive no separate rewards. Here we assume that the reward $R(m, c)$ combines M molecular property scores $R_k(m, c)$, with $k = 1, \dots, M$, through a fixed scalar function F:

$$R(m, c) = F\big(R_1(m, c), \dots, R_M(m, c)\big), \tag{2}$$

where the property scores can be measures of, e.g., binding affinity, drug-likeness, synthetic accessibility, and conformational strain. The fixed aggregation F specifies the molecular objective; our method constructs group-relative training feedback from the individual property scores.

In this paper, among the existing generative models for SBDD, we choose pre-trained BFNs as molecule generators as they can represent continuous coordinates and categorical atom types using continuous belief parameters (Graves et al., 2023; Qu et al., 2024). To realize the self-improving process, we formulate the learning of BFNs as an on-policy reinforcement learning task. In particular, we adopt Group Relative Policy Optimization (GRPO) which compares candidates sharing (c, N) to update the generator without requiring a learned value model or gradients through molecular evaluators. In the next section, we provide the key background of BFNs and GRPO, before details of the proposed self-improving process, including the policy formulation and training loss, are presented.

3.2 Background

Bayesian flow networks. Bayesian flow networks (BFNs) (Graves et al., 2023) generate data by iteratively refining input distributions that encode beliefs about the data. A neural network predicts an output distribution from the current belief parameters, while Bayesian update rules describe how noisy samples revise these beliefs. With categorical distributions represented as continuous probability vectors, categorical and continuous variables can be processed in a unified manner. This makes BFNs appealing for modeling and generation of 3D molecules.

Following MolCRAFT (Qu et al., 2024), the coordinate input distribution is Gaussian with parameters $\boldsymbol{\theta}_i^x = \{\boldsymbol{\mu}_i, \rho_i\}$: $\boldsymbol{\mu}_i \in \mathbb{R}^{N\times 3}$ contains one three-dimensional mean vector per atom, and ρ_i is the inverse variance per coordinate component. Each row of $\boldsymbol{\theta}_i^v \in [0,1]^{N\times K}$ specifies a categorical input distribution over the K types for one atom. We write $\boldsymbol{\theta}_i = \boldsymbol{\theta}_i^v$ for brevity in the sequel.

The sampler considered here starts from zero coordinate means, unit precision, and uniform type probabilities, and then iterates for S sampling steps. We use normalized process times $t_i = i/S$ for $i = 0, \ldots, S$. At step i, the network Ψ_ϕ uses the current belief parameters, pocket c, and the process time t_{i-1} to predict clean coordinates $\hat{\mathbf{x}}_{\phi,i}$ and type probabilities $\hat{\mathbf{p}}_{\phi,i}$. These predictions and the fixed cumulative accuracy schedules at t_i parameterize Gaussian draws of the next coordinate means $\boldsymbol{\mu}_i$ and type logits $\mathbf{y}_i$; row-wise softmax of $\mathbf{y}_i$ gives $\boldsymbol{\theta}_i$. The predictions describe clean data, while the updated belief parameters become the next network input.

The update noise scales can be determined using fixed cumulative accuracy schedules (Qu et al., 2024), with $\beta(t)$ for coordinates and $\beta'(t)$ for atom types:

$$\beta(t) = \sigma_1^{-2t} - 1, \qquad \beta'(t) = \beta_1 t^2, \qquad t \in [0,1]. \tag{3}$$

Since $\rho_i = 1 + \beta(t_i)$, $\sigma_1 \in (0,1)$ is the coordinate-belief standard deviation at $t = 1$; $\beta_1 > 0$ is the terminal atom-type cumulative accuracy. After S updates, a final network evaluation provides the output coordinates and probabilities from which the final atom types are sampled.

Group Relative Policy Optimization. GRPO is a reinforcement learning approach which derives feedback from reward comparisons within a group for policy-gradient learning, without learning a separate value function (Shao et al., 2024), which has been found effective for training LLMs.

It uses a *group-relative advantage* for policy learning. Given a particular input, a fixed policy snapshot ϕ_{old} generates a group $\mathcal{G}$ of $G \geq 2$ trajectories of sequential actions with scalar terminal rewards R_j, where $j = 1, \ldots, G$ indexes group members. The group-relative advantage is defined as:

$$A_j^{\text{scalar}} = \frac{R_j - \bar{R}_{\mathcal{G}}}{s_{\mathcal{G}} + \varepsilon_{\text{std}}}, \qquad j = 1, \ldots, G, \tag{4}$$

where $\bar{R}_{\mathcal{G}}$ and $s_{\mathcal{G}}$ are the group reward mean and standard deviation (using denominator G), and $\varepsilon_{\text{std}} > 0$ prevents division by zero. Positive and negative advantages indicate rewards above and below the group mean. Under the undiscounted terminal-reward setting, the same group-relative advantage is assigned to every step of a trajectory.

Let $\pi_\phi(a_i \mid s_{i-1})$ denote the probability mass or density of action a_i at state s_{i-1}, omitting the common input from notation. During updates, sampled states, actions, rewards, advantages, and ϕ_{old} remain fixed. The per-step likelihood ratio compares current and old policies for the full sampled

action:

$$\rho_{j,i}^{\text{joint}} = \frac{\pi_\phi(a_{j,i} \mid s_{j,i-1})}{\pi_{\phi_{\text{old}}}(a_{j,i} \mid s_{j,i-1})}.$$

The PPO-style clipped policy loss (Schulman et al., 2017) is

$$\mathcal{L}^{\text{clip}}(\phi) = -\mathbb{E}\Big[\min\Big(\rho_{j,i}^{\text{joint}} A_j^{\text{scalar}},\ \text{clip}(\rho_{j,i}^{\text{joint}}, 1-\epsilon, 1+\epsilon) A_j^{\text{scalar}}\Big)\Big], \tag{5}$$

where $0 < \epsilon < 1$, and the expectation averages over sampled groups, group members, and uniformly selected steps. Minimization favors actions from relatively high-reward trajectories and discourages those from relatively low-reward trajectories; and the clipping limits the incentive for further ratio changes in these directions.

Before collecting the next batch, we refresh ϕ_{old} from the updated ϕ, repeating the cycle of sampling, evaluation, and policy updating.

4 METHODOLOGY

GRAB enables self-improvement by training BFN sampling from terminal molecular scores. Group-relative advantages link scores to sampled transitions, while differentiable action densities enable comparisons of current and old policies on stored trajectories. We present the policy formulation, multi-objective feedback, clipped loss, and training and generation procedures (Figure 1).

4.1 BFN SAMPLING AS A POLICY

The molecular distribution p_ϕ is induced by a sequence of sampling actions governed by the policy model π_ϕ, followed by the final output step. Given a pocket c and atom count N, we consider each stochastic BFN sampling step as a decision of taking an action that updates beliefs about a molecule. Before step i, we define the policy state as $s_{i-1} = (\boldsymbol{\mu}_{i-1}, \boldsymbol{\theta}_{i-1}, t_{i-1})$ which consists of the current coordinate-belief means, the atom-type probabilities, and the process time. Note that the coordinate precision $\rho_{i-1} = 1 + \beta(t_{i-1})$ need not be included as it can be determined simply based on the process time.

From this state and pocket c, the network predicts clean coordinates $\hat{\mathbf{x}}_{\phi,i}$ and type probabilities $\hat{\mathbf{p}}_{\phi,i}$. Together with the fixed schedules, these predictions specify Gaussian distributions from which we sample the next coordinate means and type logits as the action $a_i = (\boldsymbol{\mu}_i, \mathbf{y}_i)$. The policy $\pi_\phi(a_i \mid s_{i-1}, c, N)$ is the joint conditional density of this sampled pair. Thus the network and sampling rule jointly define the policy.

The sampled $\boldsymbol{\mu}_i$ directly sets the next coordinate-belief means, while $\boldsymbol{\theta}_i = \text{softmax}_{\text{row}}(\mathbf{y}_i)$ converts the sampled logits into type probabilities. Together with t_i, they determine the next state $s_i = (\boldsymbol{\mu}_i, \boldsymbol{\theta}_i, t_i)$. Given the sampled state and action, the next-state mapping is fixed and does not depend on ϕ. The same construction extends to other stochastic BFNs when their action densities are evaluable and differentiable in ϕ, their supported actions do not change with ϕ, and their state update is fixed given the state and action.

Using the type logits before softmax keeps the joint action distribution Gaussian, so its density can be evaluated directly for the GRPO update. Conditioned on the current state, the sampler uses independent Gaussian noise across atoms and components of both modalities. Appendix A.1 specifies these distributions and derives their joint density; the resulting log-density is given below.

Using the schedules in Eq. (3), write $\gamma_i = \beta(t_i)/(1+\beta(t_i))$ and $\beta_i' = \beta'(t_i)$ for the coordinate flow coefficient and cumulative atom-type accuracy at step i. The conditional action log-density becomes:

$$\log \pi_\phi(a_i \mid s_{i-1}, c, N) = -\frac{\|\boldsymbol{\mu}_i - \gamma_i \hat{\mathbf{x}}_{\phi,i}\|_F^2}{2\gamma_i(1-\gamma_i)} - \frac{\|\mathbf{y}_i - \beta_i'(K\hat{\mathbf{p}}_{\phi,i} - \mathbf{1})\|_F^2}{2\beta_i' K} + C_i(N), \tag{6}$$

where $\|\cdot\|_F^2$ denotes the sum of squared entries over all atoms and components. The Gaussian normalization term $C_i(N)$ depends on the fixed schedules and action dimension, but is independent of ϕ. Thus both coordinate and atom-type predictions enter an explicit, differentiable action density, evaluated on the sampled means and logits before the deterministic softmax.

Each rollout batch uses a fixed snapshot ϕ_{old} of the current generator for all S sampling steps, the final network evaluation, and atom-type sampling. We reconstruct and score the resulting molecules, storing their property scores, sampled states and actions, and old-policy log-densities.

The final output step provides the molecule used for scoring; the policy update acts on the densities of the preceding S stochastic actions. We therefore define a batch-specific trajectory objective with the final molecule-generation rule fixed at ϕ_{old}. At a fixed terminal state, its expected reward is independent of ϕ, allowing the policy-gradient derivation in Appendix A.2. During updates, gradients enter through the current network predictions at stored states, without differentiating through the sampled trajectory, reconstruction, or scoring. The snapshot is refreshed for the next rollout batch.

4.2 Group Relative Policy Optimization

Terminal scores determine which trajectories are favored and how their sampled actions are weighted in the update. For each conditioning pair (c, N), we independently sample G_0 trajectories under $\pi_{\phi_{\text{old}}}$ and retain successful samples as a group $\mathcal{G}$. Each terminal property score $R_k(m_j, c)$ is mapped to a higher-is-better value $r_{j,k} \in [0, 1]$ and standardized using fixed calibration statistics to obtain $z_{j,k}$; Appendix B.1 specifies these transformations. For each reward-weight configuration, the fixed aggregation in Eq. (2) is $R(m_j, c) = \sum_{k=1}^{M} w_k z_{j,k}$, with weights and calibration statistics held fixed throughout training.

We normalize reward channels separately within each group before combining them, as in MO-GRPO and GDPO (Ichihara et al., 2026; Liu et al., 2026). Let $\bar{z}_k = G^{-1} \sum_{j=1}^{G} z_{j,k}$ and $s_k^2 = G^{-1} \sum_{j=1}^{G} (z_{j,k} - \bar{z}_k)^2$ denote the group mean and variance, with $s_k \geq 0$. For fixed weights w_k, we retain channels $\mathcal{K}_{\mathcal{G}} = \{k : w_k \neq 0,\ s_k \geq \varepsilon_A\}$, where $\varepsilon_A > 0$ prevents division by very small standard deviations, and define

$$A_j = \sum_{k \in \mathcal{K}_{\mathcal{G}}} w_k \frac{z_{j,k} - \bar{z}_k}{s_k}. \tag{7}$$

This group-relative advantage is held fixed during batch updates and assigned to every transition of trajectory j.

Using the action density in Eq. (6), we compare the current and old policies on each stored state–action pair:

$$\ell_{j,i}(\phi) = \frac{1}{D} \log \pi_\phi(a_{j,i} \mid s_{j,i-1}, c, N), \qquad \rho_{j,i}(\phi) = \exp[\ell_{j,i}(\phi) - \ell_{j,i}(\phi_{\text{old}})], \tag{8}$$

where the scale $D > 0$, fixed across molecules and timesteps, controls sensitivity to joint log-density changes. At $D = 1$, $\rho_{j,i}$ is the joint likelihood ratio; otherwise, it is a power of that ratio. With clipping margin $0 < \epsilon < 1$, we use the PPO term $h(\rho, A) = \min\{\rho A, \text{clip}(\rho, 1 - \epsilon, 1 + \epsilon)A\}$ (Schulman et al., 2017).

To reduce computation, each gradient update evaluates a fresh timestep subset $\mathcal{T}$, with $|\mathcal{T}| = T$, shared across the batch. We use stratified sampling independent of the per-step losses, as detailed in Appendix A.4.2. For H conditioning groups per batch, let $\mathcal{P} \subseteq \{1, \ldots, H\}$ index those retaining $G_p \geq 2$ successful trajectories. We minimize

$$\mathcal{L}(\phi) = -\frac{1}{|\mathcal{P}|T} \sum_{p \in \mathcal{P}} \frac{1}{G_p} \sum_{j=1}^{G_p} \sum_{i \in \mathcal{T}} h\big(\rho_{p,j,i}(\phi), A_{p,j}\big), \tag{9}$$

Averaging within each group gives groups equal weight regardless of size; batches with no retained groups are skipped. Terminal scores provide fixed advantages, while gradients enter through the action densities. With the terminal output rule fixed within each batch, sample filtering, group normalization, ratio scaling, and clipping, this loss is a surrogate rather than an exact reformulation of the molecular reward objective in Eq. (1).

4.3 Training and Generation

Starting from ϕ_0, each iteration generates molecular groups under $\phi_{\text{old}} \leftarrow \phi$ and computes advantages from terminal scores. We minimize Eq. (9) using fixed trajectories, advantages, and old-policy densities, drawing a fresh timestep subset $\mathcal{T}$ for each gradient update. The updated generator then produces the next batch, closing the self-improvement loop. Algorithm 1 summarizes the procedure.

At inference, the learned parameters govern all S sampling transitions and the final network evaluation. We retain the pretrained architecture, schedules, atom-count prior, and sampling recursion; generation requires no additional property predictor or property-gradient guidance.

Table 1: Comparison on 100 CrossDocked2020 test pockets. Vina scores and strain energy (SE) are in $\mathrm{kcal\,mol^{-1}}$; SE columns report quartiles. Div denotes ligand diversity. Arrows indicate preferred directions. Bold and underlined values mark the best and second-best results, excluding Reference and including ties. GRAB-A and GRAB-B use weights $(0.40, 0.20, 0.20, 0.20)$ and $(0.35, 0.15, 0.15, 0.35)$, respectively, ordered as affinity, QED, SA, and SE.

Method	Binding Affinity						Conformation Stability			Molecular Properties		
	Vina Score (↓)		Vina Min (↓)		Vina Dock (↓)		SE (↓)			SA (↑)	QED (↑)	Div (↑)
	Avg.	Med.	Avg.	Med.	Avg.	Med.	25%	50%	75%	Avg.	Avg.	Avg.
Reference	−6.36	−6.46	−6.71	−6.49	−7.45	−7.26	34	107	196	0.73	0.48	—
AR	−5.75	−5.64	−6.18	−5.88	−6.75	−6.62	259	595	2286	0.64	0.51	0.70
Pocket2Mol	−5.14	−4.70	−6.42	−5.82	−7.15	−6.79	102	189	374	0.76	**0.57**	0.69
TargetDiff	−5.47	−6.30	−6.64	−6.83	−7.80	−7.91	369	1243	13871	0.58	0.48	0.72
DecompDiff	−5.19	−5.27	−6.03	−6.00	−7.03	−7.16	115	421	1424	0.66	0.51	0.73
IPDiff	−6.70	−7.53	−7.63	−7.71	−8.41	−8.39	1,976	75,326	6,140,489	0.61	0.52	0.71
IPDiff + DPO	−6.37	−7.20	−7.55	−7.62	−8.56	−8.60	330	1,241	23,836	0.62	0.53	0.71
PAFlow	**−8.16**	**−8.73**	−8.69	**−8.76**	−9.29	−9.28	3,302	86,010	4,568,008	0.57	0.49	0.70
MolFORM (Base)	−5.29	−5.52	−6.04	−5.92	−6.90	−6.92	400	1,725	36,190	0.62	0.42	**0.77**
MolFORM-RL	−7.60	−7.62	−8.32	−8.21	−9.11	−9.21	261	1,112	41,766	0.68	0.49	0.75
MolCRAFT	−6.59	−7.04	−7.27	−7.26	−7.92	−8.01	83	195	510	0.69	0.50	0.72
BFN-DPO	−6.55	−7.30	−7.74	−7.83	−8.51	−8.76	111	256	930	0.65	**0.57**	0.69
GRAB-A	−6.08	−7.00	−7.51	−7.60	−8.53	−8.61	**42**	105	223	**0.77**	0.55	0.76
GRAB-B	−7.59	−8.43	**−8.76**	−8.74	**−9.52**	**−9.54**	44	**104**	**200**	0.76	0.51	0.75

5 Experiments

5.1 Experimental setup

Data. We use CrossDocked2020 (Francoeur et al., 2020), following the train–test split of Luo et al. (2021) and MolCRAFT's preprocessing (Qu et al., 2024). Our processed split contains 99,990 training pocket–ligand pairs and 100 test pockets. For fine-tuning, we reserve 24 training-split pockets as a validation set for monitoring and checkpoint selection, leaving 99,966 training entries from which rollout conditions are sampled. Policy updates use generated molecules and their rewards. Test pockets are excluded from policy updates, reward calibration, and checkpoint selection.

Self-improvement training. Using the rollout pool described above, we fine-tune pretrained MolCRAFT for 1,200 iterations per reward-weight configuration, retaining its default backbone and using $S = 100$ transitions for rollouts and generation. Each iteration collects a fresh rollout batch and updates the policy using terminal Vina Min, QED, SA, and strain energy (SE) scores. These are mapped to $[0, 1]$, with higher values preferred, and standardized using fixed statistics from pretrained-model samples on training pockets to compute group-relative advantages. We evaluate checkpoints every 50 iterations, retaining the best so far according to the validation weighted sum of mean standardized rewards, using the configuration's training weights. Selection is restricted to checkpoints whose mean mapped reward in every channel meets or exceeds the corresponding pretrained calibration mean. Appendix B provides the reward maps and detailed training and validation protocols.

GRAB configurations. GRAB-A and GRAB-B use two prespecified reward-weight configurations, $\mathbf{w}_A = (0.40, 0.20, 0.20, 0.20)$ and $\mathbf{w}_B = (0.35, 0.15, 0.15, 0.35)$, respectively, ordered as affinity, QED, SA, and SE. For each configuration, we perform a single fine-tuning run and report test results from the checkpoint selected using only the held-out validation set, following the criterion in Appendix B.2. Test-set scores are not used for checkpoint selection. Table 2 reports results for all five trained reward-weight configurations.

Baselines. We compare GRAB with autoregressive models AR and Pocket2Mol; diffusion models TargetDiff, DecompDiff, and IPDiff; flow-matching models PAFlow and MolFORM (Base); and the BFN model MolCRAFT, which also provides GRAB's pretrained initialization. Property-optimization baselines include IPDiff + DPO, MolFORM-RL, and BFN-DPO. Reference denotes the dataset ligands. Model references, baseline variants, and result sources are provided in Appendix B.4.

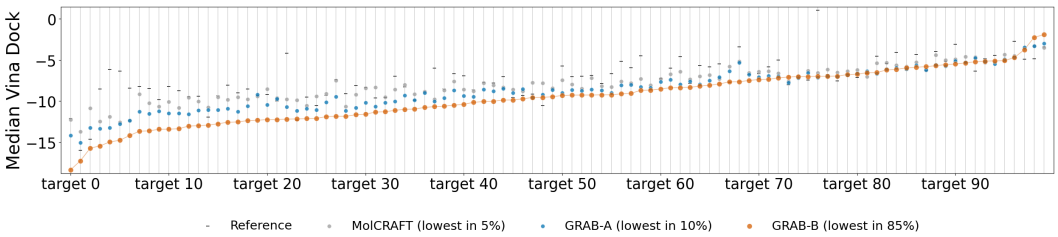


Figure 2: Vina Dock scores across 100 CrossDocked2020 test pockets. Colored points represent medians over finite scores from 100 generated ligands per pocket; black dashes represent the corresponding reference ligand scores. All methods share the pocket order defined by increasing GRAB-B median. Legend percentages indicate the fraction of pockets where each generator has the lowest median among MolCRAFT, GRAB-A, and GRAB-B.

Table 2: Comparison of GRAB reward-weight configurations on the 100 CrossDocked2020 test pockets. Each row represents a separately trained configuration, with weights ordered as affinity, QED, SA, and SE. Metric definitions and units follow Table 1.

Configuration	Reward weights (Vina Min, QED, SA, SE)	Binding Affinity						Conformation Stability			Molecular Properties		
		Vina Score (↓)		Vina Min (↓)		Vina Dock (↓)		SE (↓)			SA (↑)	QED (↑)	Div (↑)
		Avg.	Med.	Avg.	Med.	Avg.	Med.	25%	50%	75%	Avg.	Avg.	Avg.
1	(0.25, 0.25, 0.25, 0.25)	−6.12	−7.17	−7.26	−7.44	−8.31	−8.41	51	125	245	0.75	0.56	0.76
2 (GRAB-A)	(0.40, 0.20, 0.20, 0.20)	−6.08	−7.00	−7.51	−7.60	−8.53	−8.61	42	105	223	0.77	0.55	0.76
3	(0.40, 0.10, 0.10, 0.40)	−7.54	−8.32	−8.52	−8.55	−9.05	−9.18	43	117	250	0.75	0.53	0.74
4	(0.50, 0.10, 0.10, 0.30)	−6.71	−7.65	−8.10	−8.11	−9.09	−9.19	38	120	276	0.78	0.52	0.75
5 (GRAB-B)	(0.35, 0.15, 0.15, 0.35)	−7.59	−8.43	−8.76	−8.74	−9.52	−9.54	44	104	200	0.76	0.51	0.75

Evaluation. Our main evaluation covers all 100 CrossDocked2020 test pockets, with 100 generation attempts per pocket. Following MolCRAFT, we report Vina Score for generated poses, Vina Min after local minimization, and Vina Dock after redocking (Trott & Olson, 2010), together with strain energy (SE) (Harris et al., 2023). These metrics are measured in $\mathrm{kcal\,mol^{-1}}$, with lower values preferred. We also report QED (Bickerton et al., 2012), normalized SA (Ertl & Schuffenhauer, 2009), and ligand diversity (Div), for which higher values are preferred. We use MolCRAFT's public evaluation code and report metrics before the reward transformations; Div is used only for evaluation. Appendix B.5 details the metric calculations and sample handling.

5.2 Main results

Joint improvements in docking scores and strain energy. Table 1 shows that favorable docking scores do not necessarily coincide with low conformational strain. PAFlow and MolFORM-RL achieve stronger Vina Dock scores but substantially higher SE than MolCRAFT. GRAB-B achieves the lowest mean Vina Dock (-9.52) and median SE (104) among the compared generators, improving on both reported MolCRAFT values (-7.92, 195) and our re-evaluation of the same checkpoint (-7.81, 153; Appendix C). Its SA (0.76), QED (0.51), and ligand diversity (0.75) also exceed MolCRAFT's reported values. GRAB-A offers higher SA (0.77), QED (0.55), and diversity (0.76) than GRAB-B, while also improving Vina Dock and SE relative to MolCRAFT.

Docking performance across pockets. Among the three generators in Figure 2, GRAB-B achieves the lowest per-pocket median Vina Dock on 85 of the 100 test pockets, compared with 10 for GRAB-A and five for MolCRAFT. Its docking advantage therefore extends across most evaluated pockets.

6 Analysis

6.1 Sensitivity to reward weights

Table 2 compares GRAB across reward-weight configurations. Equal weighting yields the highest QED (0.56), while configuration 4 achieves the highest SA (0.78). Relative to equal weighting, GRAB-B achieves a more favorable mean Vina Dock score (-9.52 versus -8.31) and lower median

SE (104 versus 125), with lower QED (0.51 versus 0.56). Across the evaluated configurations, SA ranges from 0.75 to 0.78 and ligand diversity from 0.74 to 0.76. These results illustrate different trade-offs between docking, conformational strain, and molecular properties, with no configuration leading on every metric. Because multiple weights change together, the comparison does not isolate the effect of any individual reward weight.

6.2 Simultaneous improvements over reference ligands

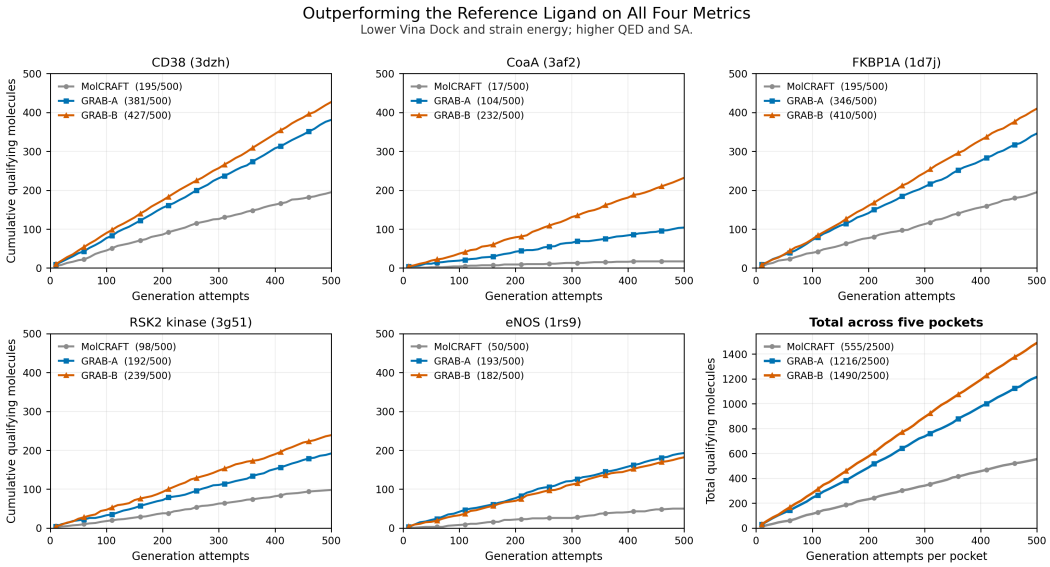


Figure 3: Simultaneous improvements over reference ligands on five randomly selected test pockets. Curves count generated molecules with lower Vina Dock and SE and higher QED and SA than the corresponding reference ligand. Each method makes 500 generation attempts per pocket. The bottom-right panel sums counts across pockets at the same per-pocket budget. Legend fractions give qualifying counts over all attempts, including failures.

The main benchmark reports individual property metrics across all 100 test pockets. To examine whether improvements occur together in the same molecule, we conduct an additional analysis on five randomly selected test pockets, using 500 generation attempts per method per pocket (Figure 3). A molecule qualifies if it achieves lower Vina Dock and SE and higher QED and SA than the corresponding reference ligand. Across these five pockets, qualifying molecules account for 48.6% of attempts for GRAB-A, 59.6% for GRAB-B, and 22.2% for MolCRAFT, with failed attempts counted as non-qualifying. Both GRAB configurations yield more qualifying molecules than MolCRAFT in each of the five pockets.

7 Conclusion and Limitations

We introduced GRAB, a self-improving framework that trains stochastic Bayesian flow networks through repeated molecular generation, evaluation, and group-relative policy optimization. By formulating coordinate and atom-type belief updates as joint stochastic actions with explicit conditional densities, GRAB connects terminal molecular scores to policy updates without differentiating through reconstruction or evaluation. Experiments on 100 CrossDocked2020 test pockets show improved docking scores and reduced conformational strain relative to MolCRAFT, while retaining favorable molecular properties. The trained generator uses the original sampling procedure without additional property predictors or property-gradient guidance.

The current formulation combines channel-wise advantages through a weighted sum with manually specified weights, making weight selection an empirical choice. Our rewards cover docking, strain energy, QED, and synthetic accessibility; toxicity, target selectivity, and other pharmacologically relevant properties are not explicitly optimized. Future work could explore adaptive weighting, Pareto-based optimization, and broader reward sets to address more diverse objectives in SBDD.

AI USE STATEMENT

We used generative AI tools to assist with drafting and polishing the manuscript and identifying relevant literature. AI tools also supported methodological discussions, the clarification and consistency checking of mathematical formulations and derivations, and the interpretation of reported experimental results. The authors take full responsibility for the final manuscript, including its claims, citations, mathematical arguments, and experimental results.

ETHICS STATEMENT

This work aims to support structure-based drug design through computational optimization of molecular candidates. Improvements in the reported metrics do not establish biological efficacy or safety, which require experimental validation. We acknowledge the potential misuse of molecular generative models and emphasize responsible use and appropriate oversight in downstream applications.

REPRODUCIBILITY STATEMENT

We initialize GRAB from the publicly released pretrained MolCRAFT checkpoint. Section 4 and Algorithm 1 describe the policy formulation and training procedure, with supporting mathematical derivations in Appendix A. Section 5.1 specifies the dataset, split, and experimental setup. Appendix B documents reward construction and calibration, implementation details, hyperparameters, baseline configurations, and evaluation procedures.

REFERENCES

Timothy Atkinson, Thomas D. Barrett, Scott Cameron, Bora Guloglu, Matthew Greenig, Charlie B. Tan, Louis Robinson, Alex Graves, Liviu Copoiu, and Alexandre Laterre. Protein sequence modelling with Bayesian flow networks. *Nature Communications*, 16(1):3197, 2025. doi: 10.1038/s41467-025-58250-2. URL https://doi.org/10.1038/s41467-025-58250-2.

G. Richard Bickerton, Gaia V. Paolini, Jérémy Besnard, Sorel Muresan, and Andrew L. Hopkins. Quantifying the chemical beauty of drugs. *Nature Chemistry*, 4(2):90–98, 2012. doi: 10.1038/nchem.1243. URL https://doi.org/10.1038/nchem.1243.

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. In *The Twelfth International Conference on Learning Representations*, pp. 4965–4987, 2024. URL https://openreview.net/forum?id=YCWjhGrJFD.

Martin Buttenschoen, Garrett M. Morris, and Charlotte M. Deane. PoseBusters: AI-based docking methods fail to generate physically valid poses or generalise to novel sequences. *Chemical Science*, 15(9):3130–3139, 2024. doi: 10.1039/D3SC04185A. URL https://doi.org/10.1039/D3SC04185A.

Peter Ertl and Ansgar Schuffenhauer. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of Cheminformatics*, 1(1):8, 2009. doi: 10.1186/1758-2946-1-8. URL https://doi.org/10.1186/1758-2946-1-8.

Paul G. Francoeur, Tomohide Masuda, Jocelyn Sunseri, Andrew Jia, Richard B. Iovanisci, Ian Snyder, and David R. Koes. Three-dimensional convolutional neural networks and a cross-docked data set for structure-based drug design. *Journal of Chemical Information and Modeling*, 60(9):4200–4215, 2020. doi: 10.1021/acs.jcim.0c00411. URL https://doi.org/10.1021/acs.jcim.0c00411.

Alex Graves, Rupesh Kumar Srivastava, Timothy Atkinson, and Faustino Gomez. Bayesian flow networks. *arXiv preprint arXiv:2308.07037*, 2023. URL https://arxiv.org/abs/2308.07037.

Siyi Gu, Minkai Xu, Alexander Powers, Weili Nie, Tomas Geffner, Karsten Kreis, Jure Leskovec, Arash Vahdat, and Stefano Ermon. Aligning target-aware molecule diffusion models with exact energy optimization. In *Advances in Neural Information Processing Systems*, volume 37, pp.

44040–44063. Curran Associates, Inc., 2024. doi: 10.52202/079017-1398. URL https://doi.org/10.52202/079017-1398.

Jiaqi Guan, Wesley Wei Qian, Xingang Peng, Yufeng Su, Jian Peng, and Jianzhu Ma. 3D equivariant diffusion for target-aware molecule generation and affinity prediction. In *The Eleventh International Conference on Learning Representations*, 2023a. URL https://openreview.net/forum?id=kJqXEPXMsE0.

Jiaqi Guan, Xiangxin Zhou, Yuwei Yang, Yu Bao, Jian Peng, Jianzhu Ma, Qiang Liu, Liang Wang, and Quanquan Gu. DecompDiff: Diffusion models with decomposed priors for structure-based drug design. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 11827–11846. PMLR, 2023b. URL https://proceedings.mlr.press/v202/guan23a.html.

Charles Harris, Kieran Didi, Arian R. Jamasb, Chaitanya K. Joshi, Simon V. Mathis, Pietro Lio, and Tom Blundell. Benchmarking generated poses: How rational is structure-based drug design with generative models? *arXiv preprint arXiv:2308.07413*, 2023. URL https://arxiv.org/abs/2308.07413.

Jie Huang and Daiheng Zhang. MolFORM: Multi-modal flow matching for structure-based drug design. *arXiv preprint arXiv:2507.05503v1*, 2025. URL https://arxiv.org/abs/2507.05503v1.

Zhilin Huang, Ling Yang, Xiangxin Zhou, Zhilong Zhang, Wentao Zhang, Xiawu Zheng, Jie Chen, Yu Wang, Bin Cui, and Wenming Yang. Protein-ligand interaction prior for binding-aware 3D molecule diffusion models. In *The Twelfth International Conference on Learning Representations*, pp. 18453–18474, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/50ca96a1a9ebe0b5e5688a504feb6107-Paper-Conference.pdf.

Yuki Ichihara, Yuu Jinnai, Tetsuro Morimura, Mitsuki Sakamoto, Ryota Mitsuhashi, and Eiji Uchibe. MO-GRPO: Mitigating reward hacking of group relative policy optimization on multi-objective problems. *Transactions of the Association for Computational Linguistics*, 14:1867–1882, 2026. doi: 10.1162/tacl.a.782. URL https://doi.org/10.1162/tacl.a.782.

Ali Khodabandeh Yalabadi, Mehdi Yazdani-Jahromi, and Ozlem Ozmen Garibay. BoKDiff: best-of-K diffusion alignment for target-specific 3D molecule generation. *Bioinformatics Advances*, 5(1): vbaf137, 2025. doi: 10.1093/bioadv/vbaf137. URL https://doi.org/10.1093/bioadv/vbaf137.

Seungbeom Lee, Munsun Jo, Jungseul Ok, and Dongwoo Kim. Enhancing ligand validity and affinity in structure-based drug design with multi-reward optimization. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 33129–33142. PMLR, 2025. URL https://proceedings.mlr.press/v267/lee25k.html.

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-GRPO: Training flow matching models via online RL. In *Advances in Neural Information Processing Systems*, volume 38, pp. 40783–40818. Curran Associates, Inc., 2025a. doi: 10.52202/085713-1362. URL https://doi.org/10.52202/085713-1362.

Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Peter Belcak, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, Yejin Choi, Jan Kautz, and Pavlo Molchanov. GDPO: Group reward-decoupled normalization policy optimization for multi-reward RL optimization. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://icml.cc/virtual/2026/poster/63333.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding R1-Zero-Like training: A critical perspective. In *Second Conference on Language Modeling*, 2025b. URL https://openreview.net/forum?id=5PAF7PAY2Y.

Shitong Luo, Jiaqi Guan, Jianzhu Ma, and Jian Peng. A 3D generative model for structure-based drug design. In *Advances in Neural Information Processing Systems*, volume 34, pp. 6229–6239. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/314450613369e0ee72d0da7f6fee773c-Paper.pdf.

J. Willem M. Nissink. Simple size-independent measure of ligand efficiency. *Journal of Chemical Information and Modeling*, 49(6):1617–1622, 2009. doi: 10.1021/ci900094m. URL https://doi.org/10.1021/ci900094m.

Xingang Peng, Shitong Luo, Jiaqi Guan, Qi Xie, Jian Peng, and Jianzhu Ma. Pocket2Mol: Efficient molecular sampling based on 3D protein pockets. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 17644–17655. PMLR, 2022. URL https://proceedings.mlr.press/v162/peng22b.html.

Keyue Qiu, Yuxuan Song, Jie Yu, Hongbo Ma, Ziyao Cao, Zhilong Zhang, Yushuai Wu, Mingyue Zheng, Hao Zhou, and Wei-Ying Ma. Empower structure-based molecule optimization with gradient guided Bayesian flow networks. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 50645–50671. PMLR, 2025. URL https://proceedings.mlr.press/v267/qiu25h.html.

Yanru Qu, Keyue Qiu, Yuxuan Song, Jingjing Gong, Jiawei Han, Mingyue Zheng, Hao Zhou, and Wei-Ying Ma. MolCRAFT: Structure-based drug design in continuous parameter space. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 41749–41768. PMLR, 2024. URL https://proceedings.mlr.press/v235/qu24a.html.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Yuxuan Song, Jingjing Gong, Hao Zhou, Mingyue Zheng, Jingjing Liu, and Wei-Ying Ma. Unified Generative Modeling of 3D Molecules with Bayesian Flow Networks. In *The Twelfth International Conference on Learning Representations*, pp. 47851–47869, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/d1b1a091088904cbc7f7faa2b45c8f36-Paper-Conference.pdf.

Oleg Trott and Arthur J. Olson. AutoDock Vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *Journal of Computational Chemistry*, 31(2):455–461, 2010. doi: 10.1002/jcc.21334. URL https://doi.org/10.1002/jcc.21334.

Daiheng Zhang and Zhao Zhang. MolFORM: Preference-aligned multimodal flow matching for structure-based drug design. *arXiv preprint arXiv:2507.05503v3*, 2026. URL https://arxiv.org/abs/2507.05503v3.

Jingyuan Zhou, Hao Qian, Shikui Tu, and Lei Xu. Prior-guided flow matching for target-aware molecule design with learnable atom number. In *Advances in Neural Information Processing Systems*, volume 38, pp. 121513–121543. Curran Associates, Inc., 2025. doi: 10.52202/085713-4052. URL https://doi.org/10.52202/085713-4052.

A From the BFN Sampler to the Training Objective

This appendix presents the policy formulation and training loss for fine-tuning the pretrained BFN sampler. Appendix A.1 specifies the MolCRAFT sampling recursion and derives the induced action density. Appendix A.2 derives the score-function gradient of a trajectory objective whose terminal reward function uses the fixed rollout parameter snapshot. Appendix A.3 defines the group-relative multi-objective advantages. Appendix A.4 introduces per-step ratio scaling, stratified timestep sampling, and clipping, and Appendix A.5 combines these components into the final training loss. The policy-density and score-function identities provide the mathematical basis for the update, while the advantage construction and clipped loss specify the training rule.

A.1 BFN Sampler and Induced Policy

A.1.1 Sampling recursion

Fix a protein pocket c and a ligand atom count N, and let K be the number of possible atom types. We suppress the fixed conditioning (c, N) in the notation below. For the MolCRAFT instantiation of the molecular BFN representation in Section 3.2, the state at time $t_i = i/S$ is

$$s_i = (\boldsymbol{\mu}_i, \boldsymbol{\theta}_i, t_i), \qquad \boldsymbol{\mu}_i \in \mathbb{R}^{N\times 3}, \quad \boldsymbol{\theta}_i \in (\Delta^{K-1})^N. \tag{10}$$

Here Δ^{K-1} is the set of nonnegative K-component vectors whose entries sum to one. Each row of $\boldsymbol{\mu}_i$ is the mean of the coordinate belief for one atom; each row of $\boldsymbol{\theta}_i$ is a probability vector over its K possible types. These are the sampler's current beliefs about the clean molecule. We use the coordinate and atom-type cumulative accuracy schedules $\beta(t)$ and $\beta'(t)$ from Eq. (3), following the modality distinction in MolCRAFT's sampling algorithm (Qu et al., 2024). For the standard Gaussian coordinate prior, define

$$\gamma(t) = \frac{\beta(t)}{1+\beta(t)} = 1 - \sigma_1^{2t}, \qquad \gamma_i = \gamma(t_i), \quad \beta'_i = \beta'(t_i). \tag{11}$$

The relation defining $\gamma(t)$ follows the Gaussian BFN flow (Graves et al., 2023). The fixed parameters satisfy $0 < \sigma_1 < 1$ and $\beta_1 > 0$, so $0 < \gamma_i < 1$ and $\beta'_i > 0$ for every sampled transition $i \geq 1$. The value $\beta'_i = \beta_1 t_i^2$ is cumulative accuracy at t_i, rather than the increment $\beta'(t_i) - \beta'(t_{i-1})$; β_1 itself denotes the terminal hyperparameter.

The sampler starts with zero coordinate means, $\boldsymbol{\mu}_0 = \mathbf{0}$, and uniform type probabilities, $(\boldsymbol{\theta}_0)_{nu} = 1/K$ for every atom n and type u. At step i, the network Ψ_ϕ uses the current beliefs at t_{i-1} to predict clean coordinates $\hat{\mathbf{x}}_{\phi,i} \in \mathbb{R}^{N\times 3}$ and atom-type probabilities $\hat{\mathbf{p}}_{\phi,i} \in [0,1]^{N\times K}$. The entry $(\hat{\mathbf{p}}_{\phi,i})_{nu}$ is the predicted probability that atom n has type u, with $\sum_{u=1}^K (\hat{\mathbf{p}}_{\phi,i})_{nu} = 1$. MolCRAFT's parameter-space sampler (Qu et al., 2024) then draws the next beliefs using the schedules at t_i. Written as a mean plus Gaussian noise, the recursion for $i = 1, \ldots, S$ is

$$(\hat{\mathbf{x}}_{\phi,i}, \hat{\mathbf{p}}_{\phi,i}) = \Psi_\phi(\boldsymbol{\mu}_{i-1}, \boldsymbol{\theta}_{i-1}, t_{i-1}), \tag{12}$$

$$\boldsymbol{\mu}_i = \gamma_i \hat{\mathbf{x}}_{\phi,i} + \sqrt{\gamma_i(1-\gamma_i)}\, \boldsymbol{\varepsilon}_i^x, \tag{13}$$

$$\mathbf{y}_i = \beta'_i(K\hat{\mathbf{p}}_{\phi,i} - \mathbf{1}) + \sqrt{\beta'_i K}\, \boldsymbol{\varepsilon}_i^v, \tag{14}$$

$$\boldsymbol{\theta}_i = \mathrm{softmax}_{\mathrm{row}}(\mathbf{y}_i). \tag{15}$$

The noise arrays $\boldsymbol{\varepsilon}_i^x \in \mathbb{R}^{N\times 3}$ and $\boldsymbol{\varepsilon}_i^v \in \mathbb{R}^{N\times K}$ have independent standard normal entries, sampled independently across steps and modalities. Conditioned on s_{i-1}, the variance of each sampled coordinate-mean entry $(\boldsymbol{\mu}_i)_{nd}$ is $\gamma_i(1-\gamma_i)$, for $d = 1, 2, 3$, and the variance of each type-logit entry is $\beta'_i K$; both are fixed by the schedules. The former describes randomness in the sampled mean, whereas ρ_i^{-1} is the variance of the Gaussian belief represented by that mean. Here $\mathbf{1}$ in Eq. (14) is the $N \times K$ all-ones array. Each row of $\mathbf{y}_i$ contains noisy type logits; the row-wise softmax converts them into probabilities via $(\boldsymbol{\theta}_i)_{nu} = \exp((\mathbf{y}_i)_{nu}) / \sum_{r=1}^K \exp((\mathbf{y}_i)_{nr})$. The predicted atom-type probabilities $\hat{\mathbf{p}}_{\phi,i}$ directly determine the Gaussian mean in Eq. (14), and the resulting belief $\boldsymbol{\theta}_i$ is passed to the next network evaluation.

A.1.2 INDUCED STOCHASTIC POLICY

Fix a pocket c and atom count N. We regard

$$a_i = (\boldsymbol{\mu}_i, \mathbf{y}_i) \tag{16}$$

as the action at state s_{i-1}. The map $a_i \mapsto s_i = (\boldsymbol{\mu}_i, \mathrm{softmax}_{\mathrm{row}}(\mathbf{y}_i), t_i)$ is deterministic. We evaluate the policy density on the sampled action $a_i = (\boldsymbol{\mu}_i, \mathbf{y}_i)$. The row-wise softmax is part of the deterministic state transition, so no change-of-variables Jacobian is required in this action density. Let $\tau = (a_1, \ldots, a_S)$ denote the sampled action sequence starting from the deterministic initial state specified above; all subsequent states are determined by this sequence. Equations (13)–(14) induce the following policy density.

Proposition 1 (Closed-form BFN policy)**.** *Conditioned on s_{i-1} and the fixed (c, N), the action density factorizes as*

$$\pi_\phi(a_i \mid s_{i-1}, c, N) = \prod_{n=1}^{N} \Big[\mathcal{N}\Big(\boldsymbol{\mu}_i^{(n)}; \gamma_i \hat{\mathbf{x}}_{\phi,i}^{(n)}, \gamma_i(1-\gamma_i)\mathbf{I}_3\Big) \times \mathcal{N}\Big(\mathbf{y}_i^{(n)}; \beta_i'(K\hat{\mathbf{p}}_{\phi,i}^{(n)} - \mathbf{1}), \beta_i' K \mathbf{I}_K\Big)\Big]. \tag{17}$$

Here the superscript (n) denotes the row corresponding to atom n. The log-density of the action sequence is

$$\log \pi_\phi(\tau \mid c, N) = \sum_{i=1}^{S} \log \pi_\phi(a_i \mid s_{i-1}, c, N). \tag{18}$$

Proof. Conditioned on s_{i-1}, the network outputs in Eq. (12) are deterministic. Equations (13)–(14) are affine transformations of independent standard Gaussian noise, giving the stated Gaussian factors over modalities and atoms. Their means depend on ϕ through the network predictions, while their covariances are fixed by the schedules. Applying the chain rule to the action sequence, with each state determined by the preceding actions, gives the trajectory factorization.

A.2 POLICY-GRADIENT FORMULATION

A.2.1 TERMINAL REWARD FOR A POLICY UPDATE

After the S stochastic transitions, the terminal state $s_S = (\boldsymbol{\mu}_S, \boldsymbol{\theta}_S, 1)$ contains the final coordinate and type beliefs. MolCRAFT (Qu et al., 2024) performs one additional network evaluation, $\Psi_\phi(\boldsymbol{\mu}_S, \boldsymbol{\theta}_S, 1)$, to predict the final coordinates and type probabilities. The predicted coordinates and atom types sampled from these probabilities define the molecule $m = (\mathbf{x}, \mathbf{v})$. It is scored by the same scalar reward $R(m, c) = F(R_1(m, c), \ldots, R_M(m, c))$ defined in Eq. (2).

For an update using one batch of sampled trajectories, let ϕ_{old} denote the parameter snapshot used to generate that batch, as in the Preliminaries. Using this snapshot for the final network evaluation, we define the conditional expected terminal reward:

$$\bar{R}_{\phi_{\mathrm{old}}}(s_S, c) = \mathbb{E}[R(m, c) \mid s_S, c, \phi_{\mathrm{old}}]\,. \tag{19}$$

The quantity $\bar{R}_{\phi_{\mathrm{old}}}$ is a conditional expectation over randomness in generating the final molecule, including atom-type sampling, at a fixed terminal state and pocket. The terminal network parameters are held at ϕ_{old}, so for a fixed terminal state this reward function is independent of the parameters ϕ being updated. The snapshot is refreshed when a new batch of trajectories is generated.

We define a trajectory objective with a fixed terminal reward function $\bar{R}_{\phi_{\mathrm{old}}}$. The current parameters ϕ determine the distribution of the S stochastic actions:

$$\widetilde{J}(\phi; \phi_{\mathrm{old}}) = \mathbb{E}_{c,N}\mathbb{E}_{\tau \sim \pi_\phi(\cdot \mid c, N)}\left[\bar{R}_{\phi_{\mathrm{old}}}(s_S, c)\right]. \tag{20}$$

The outer expectation uses $c \sim \mathcal{D}$ and $N \sim q(\cdot \mid c)$, as in Eq. (1). In this objective, ϕ enters only through the trajectory distribution, while the terminal expected reward is evaluated using the fixed snapshot ϕ_{old}. The gradient derived below is for $\widetilde{J}$. The original objective J in Eq. (1) uses the current parameters ϕ for both trajectory sampling and the final network evaluation, so the derivation does not establish an exact gradient of J. Equation (20) defines an expectation under π_ϕ; reusing trajectories sampled from $\pi_{\phi_{\mathrm{old}}}$ is addressed by the policy ratios and clipped loss in Appendix A.4.

A.2.2 SCORE-FUNCTION GRADIENT

To improve the expected terminal reward in Eq. (20), we need its gradient with respect to the sampler parameters ϕ. This gradient describes how changing the sampling policy changes the expected reward and provides the starting point for the group-relative updates below. Since $\bar{R}_{\phi_{\text{old}}}$ and the distributions of (c, N) are fixed, differentiation acts on the trajectory density. For fixed (c, N), the Gaussian actions have full support $\mathbb{R}^{N\times 3} \times \mathbb{R}^{N\times K}$ independently of ϕ, so the trajectory integral is over a fixed domain. Assuming differentiation can be interchanged with integration, the identity $\nabla_\phi \pi_\phi = \pi_\phi \nabla_\phi \log \pi_\phi$ gives

$$\begin{aligned}\nabla_\phi \widetilde{J} &= \mathbb{E}_{c,N} \int \bar{R}_{\phi_{\text{old}}}(s_S, c) \nabla_\phi \pi_\phi(\tau \mid c, N)\, d\tau \\ &= \mathbb{E}_{c,N} \mathbb{E}_{\tau\sim\pi_\phi(\cdot|c,N)} \left[\bar{R}_{\phi_{\text{old}}}(s_S, c) \nabla_\phi \log \pi_\phi(\tau \mid c, N)\right].\end{aligned}$$

Here s_S is determined by τ. Expanding the trajectory log-density using Eq. (18) yields the per-step score-function identity:

$$\nabla_\phi \widetilde{J} = \mathbb{E}_{c,N} \mathbb{E}_{\tau\sim\pi_\phi(\cdot|c,N)} \left[\bar{R}_{\phi_{\text{old}}}(s_S, c) \sum_{i=1}^{S} \nabla_\phi \log \pi_\phi(a_i \mid s_{i-1}, c, N)\right]. \tag{21}$$

For a trajectory sampled under π_ϕ, generating its terminal molecule m_j with the fixed snapshot ϕ_{old} gives an observed reward $R_j = R(m_j, c)$ whose conditional expectation is $\bar{R}_{\phi_{\text{old}}}(s_{j,S}, c)$. Substituting this observed reward in Eq. (21) gives a Monte Carlo gradient estimator. For a batch generated entirely with ϕ_{old}, this Monte Carlo estimator applies directly at $\phi = \phi_{\text{old}}$; subsequent reuse of the batch is handled by the policy ratios and clipped loss in Appendix A.4. In each score term, the sampled state s_{i-1}, action a_i, and observed reward are held fixed. Gradients flow only through the current network predictions used to evaluate the action density. Thus neither docking nor the sampled state recursion requires differentiation. Because the return is terminal and undiscounted, the same Monte Carlo return is assigned to every transition.

The per-step score in Eq. (21) can be evaluated from a weighted squared error because the Gaussian covariances are fixed. For each sampled action, define the rescaled targets

$$\widetilde{\mathbf{x}}_i = \boldsymbol{\mu}_i/\gamma_i, \qquad \widetilde{\mathbf{v}}_i = \mathbf{y}_i/(\beta_i' K) + \mathbf{1}/K. \tag{22}$$

Holding these targets and the input state fixed, taking the logarithm of Eq. (17) and differentiating gives

$$\begin{aligned}\nabla_\phi \log \pi_\phi(a_i \mid s_{i-1}) = -\frac{1}{2} \nabla_\phi \Big[&\lambda_i^{\text{pos}} \|\widetilde{\mathbf{x}}_i - \hat{\mathbf{x}}_{\phi,i}\|_F^2 \\ &+ \lambda_i^{\text{typ}} \|\widetilde{\mathbf{v}}_i - \hat{\mathbf{p}}_{\phi,i}\|_F^2\Big],\end{aligned} \tag{23}$$

where $\lambda_i^{\text{pos}} = \gamma_i/(1-\gamma_i)$ and $\lambda_i^{\text{typ}} = \beta_i' K$. Each squared norm sums squared residuals over all atoms and components. These targets are rescaled sampled actions and remain fixed during differentiation, including when the actions were generated by an earlier parameter snapshot. The weights are the reciprocals of the per-entry conditional variances of the rescaled targets $\widetilde{\mathbf{x}}_i$ and $\widetilde{\mathbf{v}}_i$, respectively. In particular, $\widetilde{\mathbf{v}}_i \in \mathbb{R}^{N\times K}$ need not have nonnegative entries or rows summing to one; it is a regression target derived from logits.

A.3 GROUP-RELATIVE MULTI-OBJECTIVE ADVANTAGE

Using the same terminal objective channels as in the reward specification, we construct group-relative advantages to weight the per-step policy scores derived in Appendix A.2.2. For each conditioning pair (c, N), independently sample G_0 trajectories from $\pi_{\phi_{\text{old}}}(\cdot \mid c, N)$ and generate their terminal molecules using the same parameter snapshot. Failed samples are discarded and not resampled. Let $\mathcal{G}$ denote the retained training group, with size $G = |\mathcal{G}| \le G_0$. Groups with $G < 2$ are skipped. Sharing the pocket and atom count makes within-group comparisons under the same conditioning.

Let $z_{j,k}$ denote the finite, preprocessed value of the terminal objective score $R_k(m_j, c)$ introduced in Eq. (2). The group statistics below use only retained samples, with preprocessing and filtering described in Section 4.2 and Appendix B.1. The channel weights w_k are fixed: a positive weight

favors larger preprocessed scores, and a negative weight favors smaller ones. These signs must be consistent with the objective directions used in the reward specification.

We first compute the statistics of each reward channel within the group. Write $\bar{z}_k = G^{-1} \sum_{l=1}^{G} z_{l,k}$ for the empirical mean of reward channel k over the retained group, and define

$$s_k = \left[\frac{1}{G} \sum_{l=1}^{G} (z_{l,k} - \bar{z}_k)^2 \right]^{1/2}, \qquad \mathcal{K}_{\mathcal{G}} = \{k \in \{1, \dots, M\} : w_k \neq 0,\ s_k \geq \varepsilon_A\}, \tag{24}$$

where $\varepsilon_A > 0$ is a minimum-spread threshold. We then combine the standardized scores of the retained channels:

$$A_j = \sum_{k \in \mathcal{K}_{\mathcal{G}}} w_k \frac{z_{j,k} - \bar{z}_k}{s_k}. \tag{25}$$

Channels with zero weight or spread below ε_A contribute zero; if no channel is retained, $A_j = 0$ for every group member. The threshold ensures that every denominator used in Eq. (25) is positive. The weights w_k specify contributions on the standardized reward scale; their magnitudes alone do not determine the resulting gradient contributions.

During optimization, the advantages and all group statistics are held fixed. The same A_j weights every transition of trajectory j in the clipped policy loss of Appendix A.4. The coefficients w_k/s_k on the centered scores vary across groups even though the configured weights w_k are fixed. Consequently, the resulting weighted score estimator is not generally an unbiased estimator of the gradient in Eq. (21). Related effects of reward-standard-deviation normalization on the weighting of GRPO updates are analyzed by Liu et al. (2025b).

A.4 Practical Clipped Policy Loss

A.4.1 Per-step policy ratios and scaling

We now define the policy ratios used to update ϕ with trajectories generated by ϕ_{old}. Consider one sampled group under the fixed conditioning (c, N), which we omit below. During the update, the stored states $s_{j,i-1}$, actions $a_{j,i}$, and advantages A_j from Eq. (25) are held fixed. The per-step joint likelihood ratio used in Eq. (5),

$$\rho_{j,i}^{\text{joint}}(\phi) = \frac{\pi_\phi(a_{j,i} \mid s_{j,i-1})}{\pi_{\phi_{\text{old}}}(a_{j,i} \mid s_{j,i-1})},$$

compares the density of the same stored action under the current and old policies at the same stored state. By the trajectory factorization in Eq. (18), the exact trajectory importance ratio is

$$\rho_j^{\text{traj}}(\phi) = \frac{\pi_\phi(\tau_j)}{\pi_{\phi_{\text{old}}}(\tau_j)} = \prod_{i=1}^{S} \rho_{j,i}^{\text{joint}}(\phi). \tag{26}$$

For training, we average the per-step contributions, following the PPO formulation (Schulman et al., 2017). Before applying scaling and clipping, the objective for this fixed group is

$$\mathcal{S}_{\text{step}}(\phi) = \frac{1}{GS} \sum_{j=1}^{G} \sum_{i=1}^{S} \rho_{j,i}^{\text{joint}}(\phi) A_j. \tag{27}$$

At $\phi = \phi_{\text{old}}$, every ratio equals one, and differentiating Eq. (27) gives

$$\nabla_\phi \mathcal{S}_{\text{step}}|_{\phi=\phi_{\text{old}}} = \frac{1}{GS} \sum_{j=1}^{G} A_j \sum_{i=1}^{S} \nabla_\phi \log \pi_\phi(a_{j,i} \mid s_{j,i-1})|_{\phi=\phi_{\text{old}}}. \tag{28}$$

This is $1/S$ times the group-average score estimator weighted by the advantages from Appendix A.3. The proportionality concerns this fixed-group training signal at ϕ_{old}; it does not imply that averaging per-step ratios is equivalent to using their product at other parameter values.

The joint log-likelihood ratio sums contributions over all $N(3 + K)$ action components. To control its scale, we divide this sum by a positive constant D. The analysis below assumes a single D shared

across all molecules and timesteps and held fixed during optimization. Define the scaled log-density value and the resulting ratio as

$$\ell_{j,i}(\phi) = \frac{1}{D} \log \pi_\phi(a_{j,i} \mid s_{j,i-1}), \tag{29}$$

$$\rho_{j,i}(\phi) = \exp(\ell_{j,i}(\phi) - \ell_{j,i}(\phi_{\text{old}})) = \left(\rho_{j,i}^{\text{joint}}(\phi)\right)^{1/D}. \tag{30}$$

For $D > 1$, this scaling reduces the magnitude of the log-ratio and moves the ratio toward one; $D = 1$ recovers the original joint likelihood ratio. For $D \neq 1$, the transformed ratio is generally not the importance ratio between the current and old sampling policies. At $\phi = \phi_{\text{old}}$, the local derivative is

$$\nabla_\phi \rho_{j,i}(\phi)|_{\phi=\phi_{\text{old}}} = \frac{1}{D} \nabla_\phi \log \pi_\phi(a_{j,i} \mid s_{j,i-1})|_{\phi=\phi_{\text{old}}}, \tag{31}$$

so replacing $\rho_{j,i}^{\text{joint}}$ by $\rho_{j,i}$ in Eq. (27) multiplies the gradient in Eq. (28) by the common factor $1/D$. The scaled ratio $\rho_{j,i}$ is the quantity used in the clipped policy term defined in Appendix A.4.3.

A.4.2 Stratified timestep estimator

Rollout generation uses all S transitions defined in Appendix A.1. To reduce network evaluations during optimization, we estimate the uniform average over these transitions using T selected training timesteps. Choose integers $1 \leq T \leq S$ with T dividing S, and partition $\{1, \ldots, S\}$ into T disjoint strata of S/T indices each. Draw one index uniformly from each stratum to form $\mathcal{T}$, with $|\mathcal{T}| = T$.

Condition on the stored rollout and the current parameter value. Let g_i denote a fixed scalar or vector contribution at step i, such as a per-step loss or gradient. The timestep draw is independent of these values, and the expectation below is only over that draw:

$$\mathbb{E}_{\mathcal{T}}\left[\frac{1}{T}\sum_{i\in\mathcal{T}} g_i\right] = \frac{1}{S}\sum_{i=1}^{S} g_i. \tag{32}$$

Each timestep has inclusion probability T/S, which gives Eq. (32) by linearity of expectation. Thus the estimator is unbiased for the full timestep average, conditional on the rollout and parameter value. The same index set may be shared across a minibatch while preserving this expectation for each molecule, although doing so can correlate their timestep-sampling errors. For later parameter updates, the same conditional expectation holds when a new set is drawn according to this scheme. Reusing a previously drawn set after updating the parameters on that set does not, by itself, preserve conditional unbiasedness at the updated parameters.

A.4.3 Clipped policy term

For each selected transition, let $\rho = \rho_{j,i}(\phi)$ be the scaled ratio in Eq. (30), and let $A = A_j$ be the fixed advantage from Eq. (25). With a single fixed clipping margin $0 < \epsilon < 1$, define

$$h(\rho, A) = \min(\rho A,\ \text{clip}(\rho, 1-\epsilon, 1+\epsilon)A), \tag{33}$$

following the PPO clipping form (Schulman et al., 2017) with the symmetric interval $[1-\epsilon, 1+\epsilon]$. The policy update maximizes h, so the minimization loss in Appendix A.5 contains $-h$. Clipping changes this term from ρA only when $A > 0$ and $\rho > 1+\epsilon$, or when $A < 0$ and $\rho < 1-\epsilon$. It limits the corresponding objective contribution without imposing a hard bound on the ratio itself. Because $\rho = 1$ lies strictly inside the clipping interval, $h = \rho A$ in a neighborhood of this point. Its gradient at $\phi = \phi_{\text{old}}$ therefore equals that of ρA, using the local ratio derivative in Eq. (31).

A.5 Final Training Loss

We combine the group-relative advantages from Appendix A.3 with the scaled policy ratios, timestep sampling, and clipping rule from Appendix A.4 to obtain the training loss. Each batch samples H conditioning groups (c_p, N_p), indexed by $p = 1, \ldots, H$. Let $\mathcal{P} \subseteq \{1, \ldots, H\}$ index the groups retained after filtering, each containing $G_p \geq 2$ successful trajectories indexed by $j = 1, \ldots, G_p$. For a timestep set $\mathcal{T}$ sampled as in Appendix A.4.2, with $|\mathcal{T}| = T$ and the same set shared across the

batch, the loss is

$$\boxed{\mathcal{L}(\phi) = -\frac{1}{|\mathcal{P}|T} \sum_{p \in \mathcal{P}} \frac{1}{G_p} \sum_{j=1}^{G_p} \sum_{i \in \mathcal{T}} h(\rho_{p,j,i}(\phi), A_{p,j}).} \tag{34}$$

Here $A_{p,j}$ follows Eq. (25) within group p, $\rho_{p,j,i}$ is the scaled ratio in Eq. (30), and h is defined in Eq. (33). The loss first averages over successful trajectories and selected timesteps within each group, then averages equally over retained groups. Batches with $\mathcal{P} = \emptyset$ are skipped. The negative sign makes minimizing the loss equivalent to maximizing the average clipped policy term.

Within an update using one rollout batch, the stored states, actions, advantages, and old parameters ϕ_{old} are held fixed. Optimization updates ϕ by re-evaluating the current action densities at these stored states and actions. For a fixed batch and parameter value, the randomness in Eq. (34) comes only from the timestep set $\mathcal{T}$.

Under the common-D assumption in Appendix A.4.1, all scaled ratios equal one and clipping is inactive at $\phi = \phi_{\text{old}}$. By Eqs. (31) and (32), the expected negative gradient of Eq. (34) then equals $1/(DS)$ times the batch average of the trajectory scores, each weighted by its group-relative advantage. Each trajectory score sums the stepwise scores over all S transitions, as in Appendix A.2.2. This relation is local to ϕ_{old}, conditional on the stored batch and fixed advantages, with expectation only over timestep sampling. It does not establish unbiased gradient estimation for J or $\widetilde{J}$.

A.6 Training Algorithm

Algorithm 1 summarizes the training procedure, using the sampling recursion in Appendix A.1 and the group-averaged loss in Eq. (9). Reward preprocessing and implementation settings are given in Appendix B.

Algorithm 1 Group-relative fine-tuning of a BFN generator

Require: Pretrained parameters ϕ_0, training pocket pool $\mathcal{C}_{\text{train}}$, atom-count prior q
Require: H conditioning groups per batch, $G_0 \geq 2$ generation attempts per group; S rollout steps, $1 \leq T \leq S$ selected steps with $T \mid S$
Require: Fixed score preprocessing and sample filtering, weights w_k, threshold $\varepsilon_A > 0$
Require: Log-density scale $D > 0$; clipping margin $0 < \epsilon < 1$

Initialize $\phi \leftarrow \phi_0$
Partition $\{1, \ldots, S\}$ into T equal-sized timestep strata
for each rollout batch **do**
  Freeze $\phi_{\text{old}} \leftarrow \phi$; initialize retained-group set $\mathcal{P} \leftarrow \emptyset$ ▷ refresh from the current generator
  Draw H distinct pockets $c_1, \ldots, c_H$ uniformly from $\mathcal{C}_{\text{train}}$
  for $p = 1, \ldots, H$ **do**
    Draw $N_p \sim q(\cdot \mid c_p)$; share (c_p, N_p) across the group
    for $j = 1, \ldots, G_0$ **do**
      Sample an independent trajectory under $\pi_{\phi_{\text{old}}}(\cdot \mid c_p, N_p)$ as in Appendix A.1
      Store $\{(s_{p,j,i-1}, a_{p,j,i}, \ell_{p,j,i}(\phi_{\text{old}}))\}_{i=1}^{S}$ using Eq. (8)
      Predict final coordinates and type probabilities from $(s_{p,j,S}, c_p)$ using $\Psi_{\phi_{\text{old}}}$
      Attempt to reconstruct $m_{p,j}$ from the predicted coordinates and sampled atom types
      If reconstruction succeeds, evaluate and preprocess its terminal scores
      Discard failures; retain each successful trajectory with its scores $z_{p,j,k}$
    end for
    Form $\mathcal{G}_p$ from successful samples and set $G_p = |\mathcal{G}_p|$
    Skip group p if $G_p < 2$; otherwise add p to $\mathcal{P}$
    Reindex retained trajectories and their associated data by $j = 1, \ldots, G_p$
    Compute $A_{p,j}$ for all group members using Eq. (7)
  end for
  If $\mathcal{P} = \emptyset$, continue to the next rollout batch
  Hold the retained states, actions, advantages and $\ell_{p,j,i}(\phi_{\text{old}})$ fixed
  for each gradient update on the collected batch **do**
    Draw a fresh shared $\mathcal{T}$ with one uniform index per timestep stratum
    Evaluate $\ell_{p,j,i}(\phi)$ and $\rho_{p,j,i}(\phi)$ by Eq. (8) for $p \in \mathcal{P}$, $j = 1, \ldots, G_p$ and $i \in \mathcal{T}$
    Take a gradient step to minimize Eq. (9), updating only ϕ
  end for
end for
return Fine-tuned generator parameters ϕ

B EXPERIMENTAL DETAILS

B.1 REWARD CHANNELS AND PREPROCESSING

Terminal evaluators. We score each generated molecule using four terminal channels. The affinity score V_{Min} is obtained with AutoDock Vina (Trott & Olson, 2010) in `minimize` mode: local optimization starts from the generated pose, and the relaxed pose is scored. Thus the affinity reward evaluates the pose after local coordinate refinement. QED (Bickerton et al., 2012) and synthetic accessibility (Ertl & Schuffenhauer, 2009) are computed using RDKit-based routines. Here SA denotes the normalized, higher-is-better accessibility score, rather than the original lower-is-better SAscore. Following MolCRAFT (Qu et al., 2024), we use strain energy (SE) to assess ligand conformational stability. We compute it with the Universal Force Field (UFF) using PoseCheck (Harris et al., 2023), which measures intramolecular strain relative to relaxed conformers.

Molecular-size correction. To reduce the influence of molecular size on the Vina reward, we adopt the size-independent ligand efficiency (SILE) correction of Nissink (2009). Conventional ligand efficiency, $\mathrm{LE} = \mathrm{affinity}/\mathrm{HA}$, retains systematic size dependence. Nissink's correction instead uses $\mathrm{SILE} = \mathrm{affinity}/\mathrm{HA}^{0.3}$, with the fixed exponent 0.3 proposed in that work. Here HA is the number of ligand heavy atoms, denoted by N in our formulation. We apply this empirical correction to the Vina score, giving $V_{\mathrm{Min}}N^{-0.3}$ before clipping and rescaling.

Fixed channel maps. Failed samples are excluded before training groups are formed. The reward maps and calibration below apply to the retained samples.

Write $r_k(m, c)$ for the transformed value of raw channel $R_k(m, c)$, and $r_{j,k} = r_k(m_j, c)$ for molecule j. Suppressing (m, c) in the channel maps, for finite evaluator outputs with positive SE we use

$$r_{\mathrm{aff}} = \frac{-1 - \mathrm{clip}(V_{\mathrm{Min}}N^{-0.3}, -6, -1)}{5}, \tag{35}$$

$$r_{\mathrm{qed}} = \mathrm{clip}(\mathrm{QED}, 0, 1),$$

$$r_{\mathrm{sa}} = \frac{\mathrm{clip}(\mathrm{SA}, 0.2, 1) - 0.2}{0.8},$$

$$r_{\mathrm{se}} = \frac{1 - \mathrm{clip}\big(\log_{10}(\mathrm{SE}/B), -1, 1\big)}{2}, \qquad B = 150\ \mathrm{kcal\,mol^{-1}}. \tag{36}$$

Here V_{Min} is the numerical Vina Min score expressed in $\mathrm{kcal\,mol^{-1}}$, corresponding to the Vina Min column in the results tables. Each map takes values in $[0, 1]$, with larger values preferred. The logarithmic strain map compresses the energy range. All map parameters and clipping bounds remain fixed during fine-tuning.

Fixed reward standardization. For each generated molecule j, the maps above convert its raw affinity, QED, SA, and SE values into bounded scores $r_{j,k} \in [0, 1]$, with larger values preferred. We standardize each channel using a calibration pool generated once by the pretrained MolCRAFT model before fine-tuning. This pool comprises eight generation attempts on each of 250 training pockets, giving 2,000 attempts, of which 1,984 have all four scores available and are used to estimate the calibration statistics.

For channel k, let μ_k^{cal} and $\sigma_k^{\mathrm{cal}} > 0$ denote the mean and standard deviation of its transformed scores in this pool. The standardized reward of molecule j is

$$z_{j,k} = \frac{r_{j,k} - \mu_k^{\mathrm{cal}}}{\sigma_k^{\mathrm{cal}}}. \tag{37}$$

The calibration statistics, reported to four decimal places, are

Channel	μ_k^{cal}	σ_k^{cal}
Affinity	0.3946	0.1558
QED	0.5288	0.2059
SA	0.5986	0.1646
SE	0.4523	0.2632

The SE statistics correspond to $B = 150\ \mathrm{kcal\,mol^{-1}}$. These statistics remain fixed during fine-tuning and checkpoint selection and are shared across reward-weight configurations. Thus, $z_{j,k} = 0$ corresponds to the calibration mean of the transformed channel, and one unit corresponds to its calibration standard deviation. These fixed statistics are distinct from the within-group mean $\bar{z}_k$ and standard deviation s_k used in Eq. (7). Test pockets are not used for calibration.

B.2 Implementation details

Backbone and sampler. We use the default MolCRAFT backbone (Qu et al., 2024), initialize it with pretrained parameters, and fine-tune only its denoising network. The atom-type vocabulary has $K = 13$ classes. Training rollouts, monitoring, and final generation all use $S = 100$ sampling transitions followed by the final network evaluation. The architecture, noise schedules, atom-count prior, and sampling recursion remain fixed.

Rollout batches. Each batch contains $H = 8$ training pockets, sampled uniformly without replacement from the 99,966 entries remaining after excluding the 24 validation pockets. We draw one atom count per pocket and share it across $G_0 = 8$ generation attempts. Failed samples are discarded and not resampled. The retained group size can therefore be smaller than eight, and groups with fewer than two retained samples are skipped. Each batch contains 64 generation attempts before filtering. We use a budget of 1,200 training iterations, each consisting of one fresh rollout batch followed by its policy updates. A full run thus comprises 76,800 training generation attempts, excluding monitoring and checkpoint evaluation. We refresh the old policy from the current generator before collecting each batch. During optimization, pocket groups are processed one at a time with gradient accumulation before each optimizer step. The loss is averaged within each retained group and then equally across retained groups, as in Eq. (9).

Checkpoint selection. We reserve 24 pockets from the CrossDocked2020 training split using a fixed random seed and exclude them from training rollouts. These validation pockets are disjoint from the test split. We evaluate a candidate checkpoint every 50 training iterations, giving 24 scheduled evaluations over the 1,200-iteration budget. Each candidate checkpoint makes 20 generation attempts per validation pocket under identical sampler settings, with the atom count fixed to the number of heavy atoms in that pocket's reference ligand. This fixed-count protocol is used for checkpoint evaluation; training rollouts and test generation retain the pretrained atom-count prior.

We evaluate the retained successful samples using the fixed reward maps and calibration statistics in Appendix B.1. Failed attempts are excluded. Let $\mathcal{V}$ denote the set of retained validation samples across the 24 pockets for a given checkpoint. The validation mean of channel k is

$$\bar{z}_k^{\mathrm{val}} = \frac{1}{|\mathcal{V}|} \sum_{(p,j)\in\mathcal{V}} z_{p,j,k}, \tag{38}$$

where $z_{p,j,k}$ is the fixed-standardized reward of retained sample j on validation pocket p, and $|\mathcal{V}| > 0$ is the number of retained samples. After each checkpoint evaluation during training, we retain the checkpoint maximizing $\sum_{k=1}^{4} w_k \bar{z}_k^{\mathrm{val}}$ among those evaluated so far that satisfy $\bar{z}_k^{\mathrm{val}} \geq 0$ for all four channels, using that configuration's training weights. The constraints require each mean transformed reward to meet or exceed its pretrained calibration-pool mean. Test results are not used for checkpoint selection.

Training-time monitoring and compute. During training, we also monitor progress on the same 24 validation pockets, keeping their sampled atom counts fixed. Every 25 training iterations, we make eight generation attempts per pocket, for 192 monitoring attempts. This monitoring protocol is distinct from the 20-molecule-per-pocket checkpoint evaluation above. Each training run uses one NVIDIA RTX 4090 GPU.

B.3 Hyperparameters

We use Adam with a constant learning rate of 3×10^{-5} and two optimization epochs per rollout batch. Each epoch uses $T = 10$ of the $S = 100$ transitions, selected by drawing one uniform index from each consecutive block of ten timesteps. The joint action log-density in Eq. (8) is divided

by the fixed constant $D = 384$ for every molecule and timestep. We set $D = 24(13 + 3)$, where 24 approximates the mean number of heavy atoms per ligand, and 13 and 3 are the per-atom type and coordinate dimensions, respectively. We use a policy-ratio clipping margin of $\epsilon = 0.01$ and a minimum within-group standard-deviation threshold of $\varepsilon_A = 10^{-4}$. We clip the global gradient norm to 1.0 and check gradient finiteness before each optimizer step. No reference-policy penalty is used.

The five reward-weight vectors and their results are listed in Table 2, with weights ordered as affinity, QED, SA, and SE. GRAB-A and GRAB-B correspond to configurations 2 and 5. These two reward-weight configurations are prespecified. For each configuration, we report test results from the checkpoint selected using only the held-out validation set under the criterion in Appendix B.2.

B.4 BASELINE CONFIGURATIONS

We include autoregressive models AR (Luo et al., 2021) and Pocket2Mol (Peng et al., 2022); diffusion models TargetDiff (Guan et al., 2023a), DecompDiff (Guan et al., 2023b), and IPDiff (Huang et al., 2024); flow-matching models MolFORM (Base) (Huang & Zhang, 2025) and PAFlow (Zhou et al., 2025); and the BFN generator MolCRAFT (Qu et al., 2024). Reference denotes the dataset ligands, and the DecompDiff row uses the reference-prior variant. PAFlow uses interaction-guided sampling and a learned atom-count predictor.

Property-optimization baselines comprise MolFORM-RL (Zhang & Zhang, 2026), IPDiff + DPO, and BFN-DPO (Lee et al., 2025). MolFORM (Base) and MolFORM-RL correspond to the base (v1) and RL (v3) variants of the same arXiv work (2507.05503). IPDiff + DPO denotes IPDiff fine-tuned with direct preference optimization and is included in place of AliDiff (Gu et al., 2024), whose checkpoint was not available to us when preparing these experiments. We implement BFN-DPO from its published description, using the same pretrained MolCRAFT checkpoint and sampler settings as GRAB; an author implementation was unavailable to us. Preferences are derived from 50 candidates per pocket generated by the pretrained model on 200 training pockets, using equal reward weights for size-corrected Vina affinity, QED, SA, and SE. We set the DPO hyperparameter to $\beta_{\mathrm{DPO}} = 2$.

Results for Reference, AR, Pocket2Mol, TargetDiff, DecompDiff, and MolCRAFT in Table 1 are taken from Table 2 of Qu et al. (2024).

B.5 EVALUATION PROTOCOL

For our main evaluation, we make 100 generation attempts on each of the 100 test pockets using MolCRAFT's sampling settings.

We use MolCRAFT's public evaluation implementation (Qu et al., 2024), including the handling of reconstruction and scoring failures in the corresponding evaluation routines. Property statistics summarize the outputs retained by these routines.

For the five-pocket analysis in Section 6.2, qualifying fractions instead use all 500 generation attempts per pocket as the denominator. An attempt is non-qualifying if generation or evaluation fails, or if any of the four required improvements over the reference ligand is not achieved.

Docking affinity. The three Vina metrics distinguish pose scoring from pose optimization (Trott & Olson, 2010; Qu et al., 2024). Vina Score evaluates the generated pose without local minimization; Vina Min scores the pose after local optimization starting from that pose; Vina Dock reports the best score returned by a redocking search. These are computational affinity estimates, with more negative scores preferred. The tables report untransformed Vina scores, before the SILE correction, clipping, and reward normalization in Appendix B.1.

Conformational strain. SE measures the intramolecular energy difference relative to relaxed conformers of the same molecule, using the UFF-based PoseCheck evaluator (Harris et al., 2023). Following MolCRAFT (Qu et al., 2024), we use reconstructed, as-generated conformers as inputs to PoseCheck for both training rewards and reported evaluations. Lower SE indicates less conformational strain under this evaluator. We report SE in energy units before the logarithmic reward map.

Molecular properties. QED is a score in $[0, 1]$ that combines desirability functions for molecular properties to estimate drug-likeness (Bickerton et al., 2012). SA estimates synthetic accessibility from molecular fragments and structural complexity (Ertl & Schuffenhauer, 2009). The original SAscore ranges from 1 (easier to synthesize) to 10 (harder); our tables use a normalized, higher-is-better score. Ligand diversity (Div) is evaluated using MolCRAFT's public implementation (Qu et al., 2024). It computes Tanimoto distances (one minus similarity) between RDKit topological fingerprints for pairs of different generated samples from the same pocket. The reported mean pools these pairwise distances across pockets; higher values indicate greater structural diversity among generated ligands.

Relation to training and reporting. The four training channels use Vina Min, QED, SA, and SE; Div is an evaluation metric and does not enter the reward. The SA and QED table entries precede the additional clipping, rescaling, and calibration used during training. Table 1 reports means and

medians for each Vina mode, the 25th, 50th, and 75th percentiles for SE, and means for SA, QED, and Div. Avg. and Med. denote mean and median; SE percentiles describe the score distribution, not confidence intervals.

C Comparison with Inference-Time Guidance

We compare unguided GRAB with MolCRAFT enhanced by MolJO guidance (Qiu et al., 2025) and evaluate their combination. To compare GRAB and inference-time guidance under a common evaluation protocol, we re-evaluate the pretrained MolCRAFT checkpoint used to initialize GRAB. Our evaluation yields a mean Vina Dock of -7.81 and a median SE of 153, compared with the published values of -7.92 and 195 used in Table 1. All MolCRAFT results in this appendix use this re-evaluated baseline. Each method makes 100 generation attempts on each of the 100 test pockets using 100 sampling steps and the pretrained atom-count prior. Guidance uses MolJO's released affinity and SA predictors, with `guide_mode=param_naive`, coordinate and atom-type gradient weights of 50, and `classifier_layer=9`.

Table 3: Results with and without MolJO guidance on the 100 test pockets, using 100 generation attempts per pocket and 100 sampling steps. MolCRAFT is re-evaluated for this comparison. GRAB configurations follow Table 2; metric definitions and units follow Table 1.

Model/config.	Guidance	Binding Affinity						Conformation Stability			Molecular Properties		
		Vina Score (↓)		Vina Min (↓)		Vina Dock (↓)		SE (↓)			SA (↑)	QED (↑)	Div (↑)
		Avg.	Med.	Avg.	Med.	Avg.	Med.	25%	50%	75%	Avg.	Avg.	Avg.
MolCRAFT	No	−6.31	−6.84	−7.09	−7.11	−7.81	−7.91	66	153	346	0.69	0.53	0.74
MolCRAFT	Yes	−7.09	−7.84	−8.07	−8.21	−8.71	−8.97	54	131	280	0.75	0.54	0.75
Config. 1	No	−6.12	−7.17	−7.26	−7.44	−8.31	−8.41	51	125	245	0.75	0.56	0.76
Config. 1	Yes	−6.60	−7.90	−7.97	−8.23	−8.96	−9.10	43	115	235	0.78	0.52	0.78
GRAB-A (2)	No	−6.08	−7.00	−7.51	−7.60	−8.53	−8.61	42	105	223	0.77	0.55	0.76
GRAB-A (2)	Yes	−7.19	−7.87	−8.23	−8.35	−9.07	−9.17	41	106	221	0.79	0.51	0.77
Config. 3	No	−7.54	−8.32	−8.52	−8.55	−9.05	−9.18	43	117	250	0.75	0.53	0.74
Config. 3	Yes	−7.85	−8.84	−8.90	−9.03	−9.50	−9.69	59	126	247	0.76	0.49	0.76
Config. 4	No	−6.71	−7.65	−8.10	−8.11	−9.09	−9.19	38	120	276	0.78	0.52	0.75
Config. 4	Yes	−7.44	−8.25	−8.58	−8.66	−9.39	−9.56	38	107	252	0.79	0.47	0.77
GRAB-B (5)	No	−7.59	−8.43	−8.76	−8.74	−9.52	−9.54	44	104	200	0.76	0.51	0.75
GRAB-B (5)	Yes	−7.85	−8.80	−9.00	−9.11	−9.71	−9.81	47	101	186	0.78	0.48	0.76

Performance and generation cost. Without guidance, GRAB-B achieves better mean Vina Dock (-9.52 versus -8.71), lower median SE (104 versus 131), and higher SA (0.76 versus 0.75) than guided MolCRAFT (Table 3). Generating 10,000 candidates on a single NVIDIA RTX 4090 takes 2.7 hours with unguided GRAB versus 15.3 hours with guided MolCRAFT: the guided run takes $5.7\times$ as long (Table 4). This difference reflects guidance overhead; GRAB changes model parameters while retaining MolCRAFT's architecture and sampling procedure. Both use 100 sampling steps; timings measure generation after training and exclude Vina evaluation.

Compatibility with guidance. Applying the same guidance to GRAB further improves the mean and median scores of all three Vina modes and increases SA across all five reward configurations. For GRAB-B, mean Vina Dock improves to -9.71 and SA increases to 0.78. However, QED decreases in all five GRAB configurations, including from 0.51 to 0.48 for GRAB-B. This trade-off may reflect guidance targeting affinity and SA without an explicit QED objective.

Table 4: Generation time for 10,000 candidates using the MolCRAFT architecture on a single NVIDIA RTX 4090. The unguided and guided runs use GRAB and pretrained MolCRAFT parameters, respectively. Both use 100 sampling steps; times exclude Vina evaluation and training.

Sampling setting	Sampling steps	Time (hours)
w/o guidance	100	2.7
with guidance	100	15.3

D Additional Evaluation of Molecular Validity and Pose Quality

We assess reconstruction success and the physical plausibility of generated poses using the pretrained MolCRAFT checkpoint and the five GRAB configurations in Table 2. Each configuration makes 100 generation attempts per pocket on the 100 CrossDocked2020 test pockets. All results in this section are obtained from our evaluations, including the MolCRAFT baseline and crystal reference ligands. The MolCRAFT results therefore differ from the literature-reported values in Table 1.

Validity and completeness. We follow MolCRAFT's released reconstruction procedure (Qu et al., 2024). Its basic reconstruction mode infers bonds from atomic coordinates using covalent radii scaled by 1.3, followed by hydrogen addition and bond-order perception, without using predicted aromaticity labels. A sample is valid if reconstruction, RDKit sanitization, and SMILES conversion succeed. Completeness additionally requires a single connected fragment, identified by the absence of a dot in the SMILES string. Both rates use all 10,000 generation attempts as the denominator, including reconstruction failures.

Steric clashes. Steric clashes measure excessively close contacts between ligand and protein atoms. We use PoseCheck (Harris et al., 2023) to count atom pairs whose van der Waals overlap exceeds 0.5 Å, and report the mean count per molecule.

PoseBusters validity. PB-valid is the fraction of evaluated molecules passing all checks in the `dock` configuration of PoseBusters (Buttenschoen et al., 2024), covering chemical validity, internal geometry and energy, and protein–ligand steric compatibility. We evaluate as-generated poses against full receptors, using the evaluated molecular pool as the denominator rather than all generation attempts. The 100 crystal reference ligands are assessed using the same settings.

Table 5: Additional evaluation of molecular validity and pose quality. Configurations 1–5 correspond to the reward-weight settings in Table 2. Validity and completeness are percentages of all 10,000 generation attempts; PB-valid is the percentage passing all checks within the evaluated molecular pool. Clash reports the mean atom-pair count per molecule on the test pockets. Bold indicates the best result among generated-molecule configurations, including ties.

Model/configuration	Validity (%) ↑	Completeness (%) ↑	Clash ↓	PB-valid (%) ↑
Crystal reference	—	—	—	95.00
MolCRAFT	99.37	96.87	7.21	85.61
1	99.78	97.74	7.09	87.81
2 (GRAB-A)	99.87	98.73	9.39	84.07
3	**100.00**	99.58	6.43	91.80
4	**100.00**	**99.64**	6.92	89.18
5 (GRAB-B)	99.95	99.31	**5.56**	**91.97**

Results. All five GRAB configurations improve validity and completeness over the re-evaluated MolCRAFT baseline (Table 5). GRAB-B achieves the lowest mean clash count (5.56 versus 7.21) and highest PB-valid (91.97% versus 85.61%) among the evaluated generators. The pose-quality gains depend on the reward configuration, as GRAB-A has more clashes and lower PB-valid than MolCRAFT.

E MOLECULAR EXAMPLES

Figures 4–7 compare the dataset reference ligand and five generated ligands per method for four pockets from the 100-pocket test set, all absent from the training set. The protein names and corresponding Protein Data Bank (PDB) structure IDs are FK506-binding protein 1A (FKBP1A; PDB ID: 1D7J), vanillyl-alcohol oxidase (VAOX; PDB ID: 1E8H), p90 ribosomal S6 kinase 2 (RSK2; PDB ID: 3G51), and ADP-ribosyl cyclase 1 (CD38; PDB ID: 3DZH). Each figure shows samples from MolCRAFT, GRAB-A, and GRAB-B, with ligands rendered as sticks against gray protein-pocket surfaces. Per-ligand annotations report Vina Score, Vina Min, Vina Dock, and strain energy (SE, labeled "Strain") in $\mathrm{kcal\,mol^{-1}}$, together with dimensionless QED and normalized SA scores. Lower Vina scores and SE and higher QED and SA are preferred. Metric definitions are given in Appendix B.5.

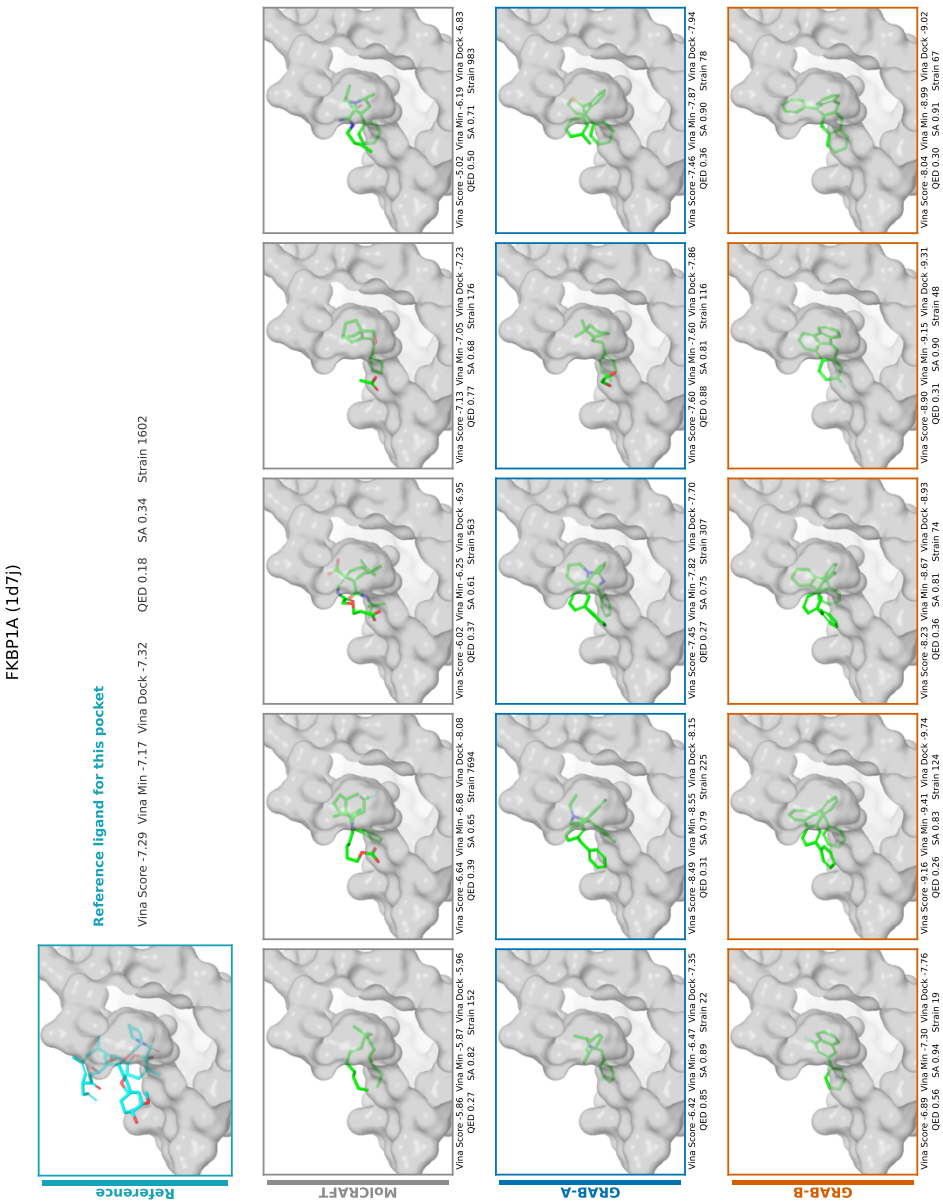


Figure 4: Reference and generated ligands for FKBP1A (PDB ID: 1D7J). The top panel shows the reference ligand; subsequent rows show five samples each from MolCRAFT, GRAB-A, and GRAB-B, respectively. Annotations give Vina Score/Min/Dock, QED, normalized SA, and SE.

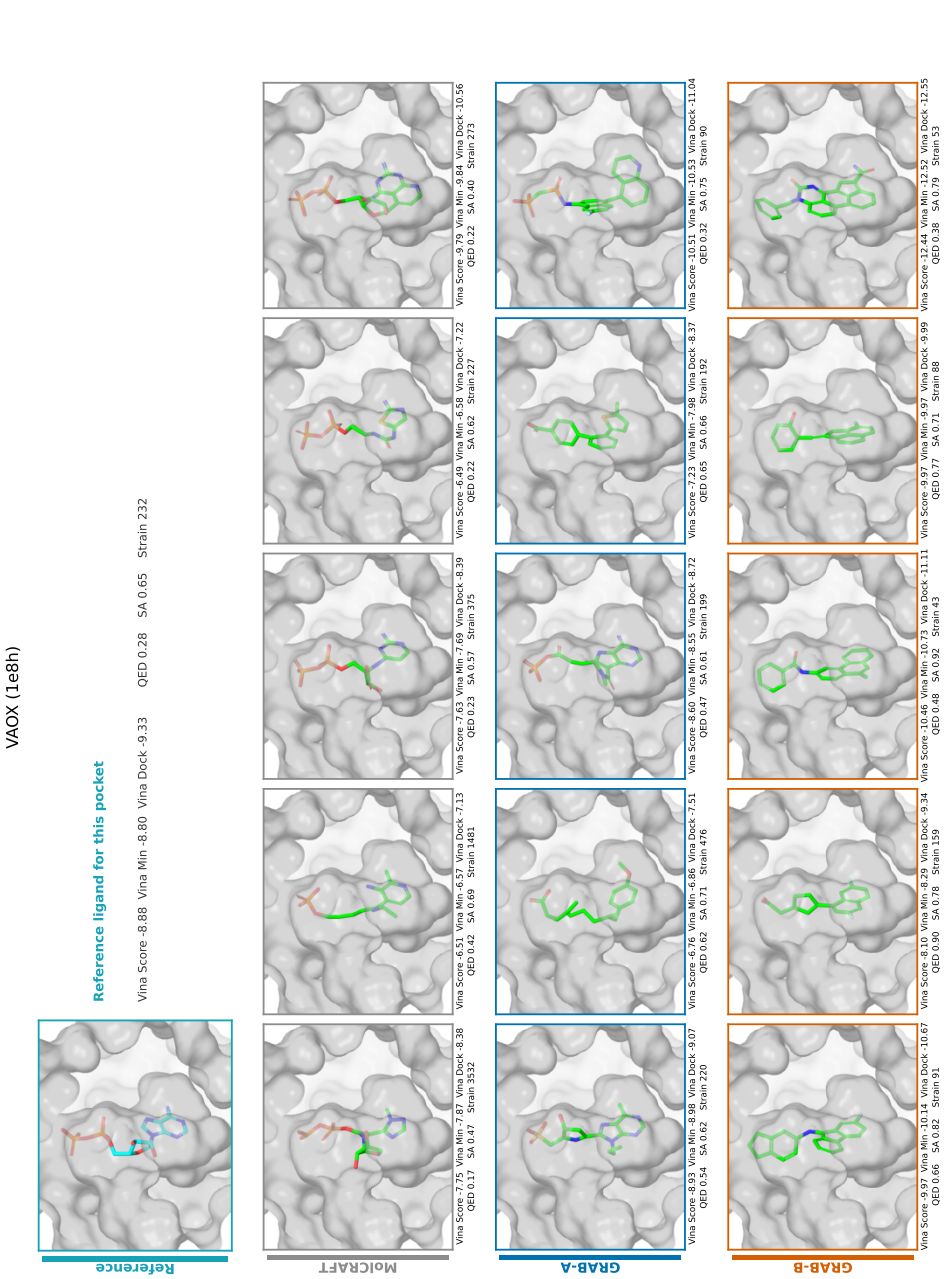


Figure 5: Reference and generated ligands for VAOX (PDB ID: 1E8H). The top panel shows the reference ligand; subsequent rows show five samples each from MolCRAFT, GRAB-A, and GRAB-B, respectively. Annotations give Vina Score/Min/Dock, QED, normalized SA, and SE.

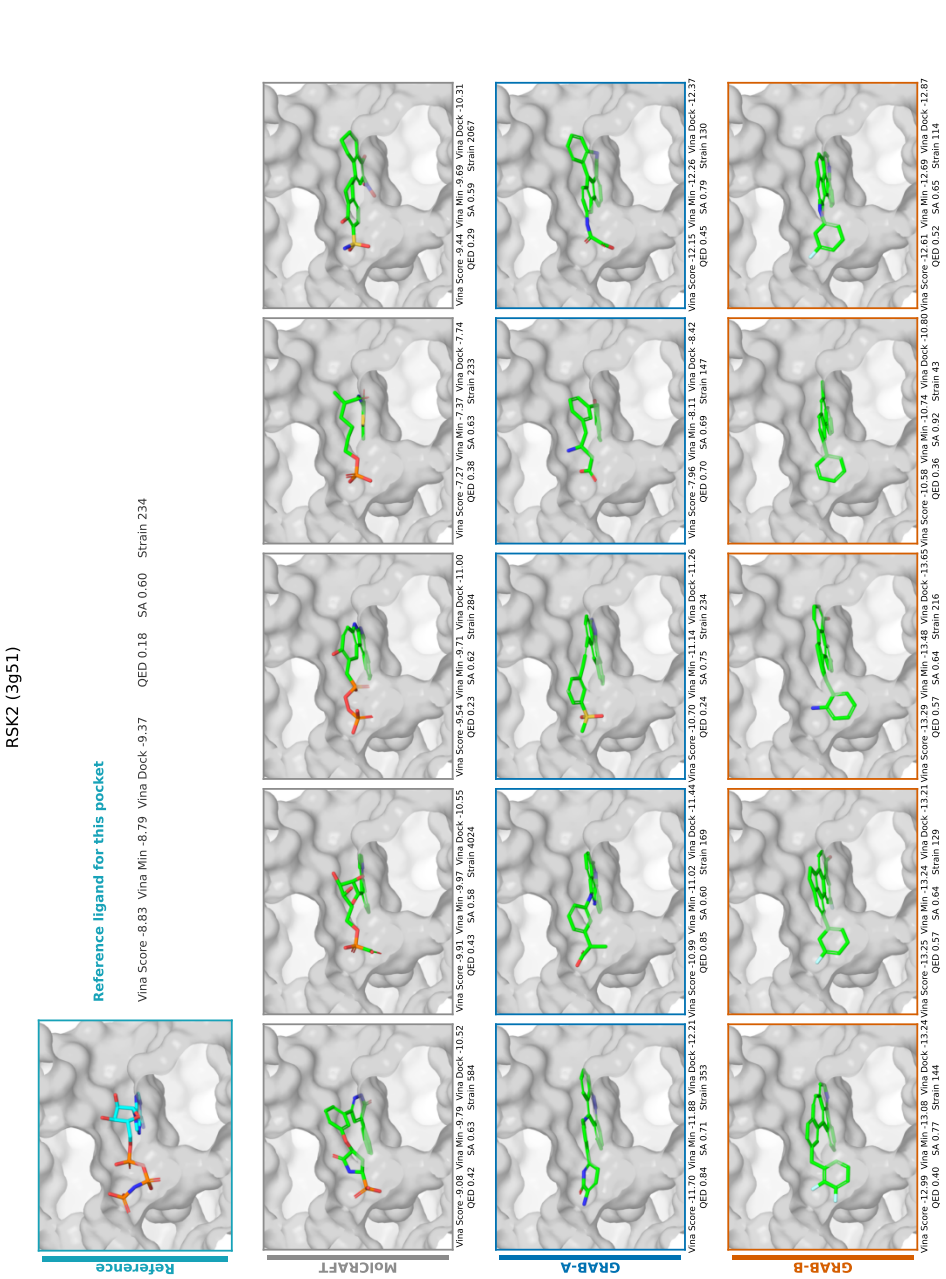


Figure 6: Reference and generated ligands for RSK2 (PDB ID: 3G51). The top panel shows the reference ligand; subsequent rows show five samples each from MolCRAFT, GRAB-A, and GRAB-B, respectively. Annotations give Vina Score/Min/Dock, QED, normalized SA, and SE.

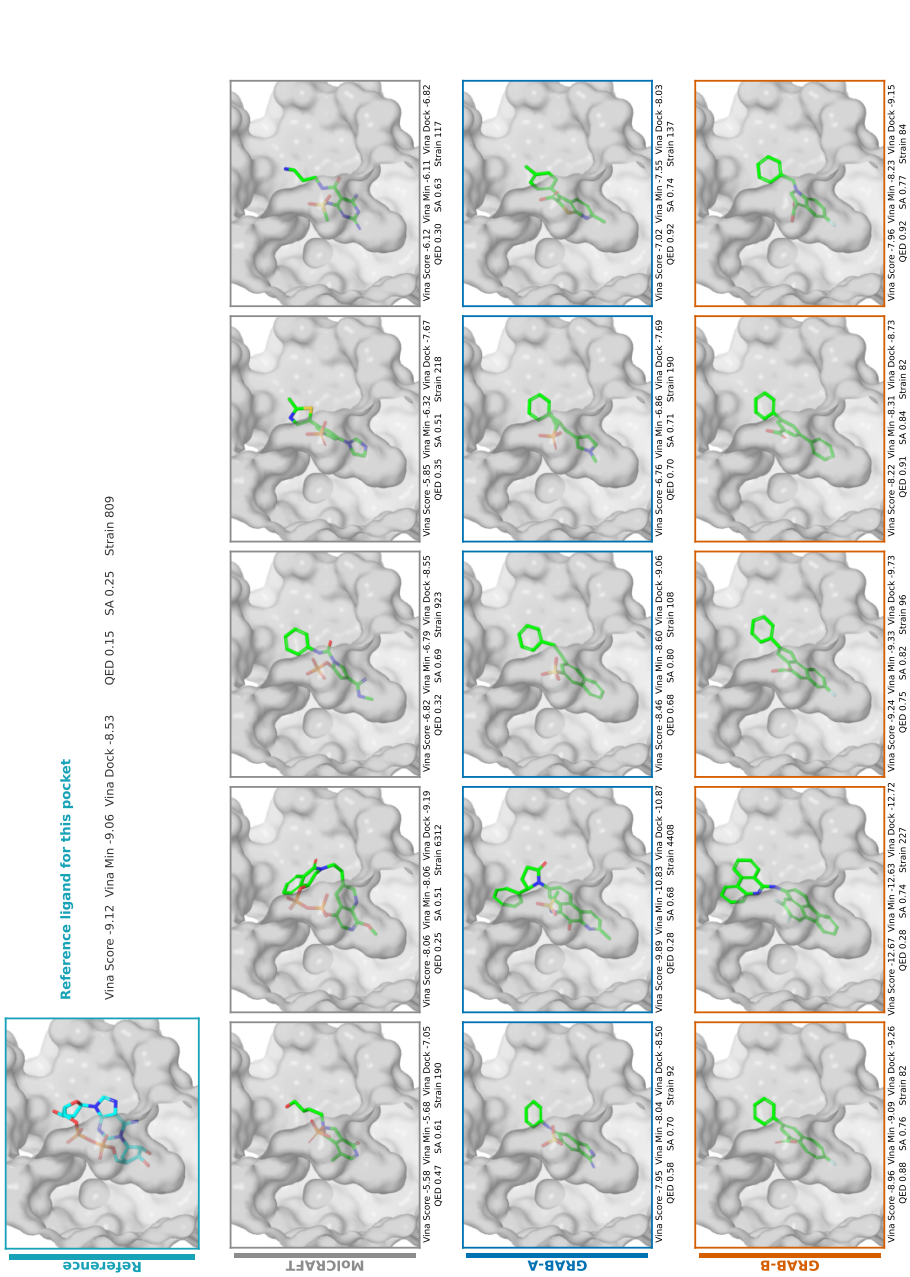


Figure 7: Reference and generated ligands for CD38 (PDB ID: 3DZH). The top panel shows the reference ligand; subsequent rows show five samples each from MolCRAFT, GRAB-A, and GRAB-B, respectively. Annotations give Vina Score/Min/Dock, QED, normalized SA, and SE.