Enhancing *de novo* Drug Design by Incorporating Diversity into Reinforcement Learning

Hampus Gummesson Svensson
Department of Computer Science and Engineering, Chalmers University of Technology and University of Gothenburg, Gothenburg, Sweden
Molecular AI, Discovery Sciences, R&D, AstraZeneca, Gothenburg, Sweden
`hamsven@chalmers.se`

Christian Tyrchan
Medicinal Chemistry, Research and Early Development,
Respiratory and Immunology (R&I), BioPharmaceuticals R&D,
AstraZeneca, Gothenburg, Sweden

Ola Engkvist
Department of Computer Science and Engineering, Chalmers University of Technology and University of Gothenburg, Gothenburg, Sweden
Molecular AI, Discovery Sciences, R&D, AstraZeneca, Gothenburg, Sweden

Morteza Haghir Chehreghani
Department of Computer Science and Engineering,
Chalmers University of Technology and University of Gothenburg,
Gothenburg, Sweden

Abstract

Fine-tuning a pre-trained generative model has demonstrated good performance in generating promising drug molecules. The fine-tuning task is often formulated as a reinforcement learning problem, where previous methods efficiently learn to optimize a reward function to generate potential drug molecules. Nevertheless, in the absence of an adaptive update mechanism for the reward function, the optimization process can become stuck in local optima. The efficacy of the optimal molecule in a local optimization may not translate to usefulness in the subsequent drug optimization process or as a potential standalone clinical candidate. Therefore, it is important to generate a diverse set of promising molecules. Prior work has modified the reward function by penalizing structurally similar molecules, primarily focusing on finding molecules with higher rewards. To date, no study has comprehensively examined how different adaptive update mechanisms for the reward function influence the diversity of generated molecules. In this work, we investigate a wide range of intrinsic motivation methods and strategies to penalize the extrinsic reward, and how they affect the diversity of the set of generated molecules. Our experiments reveal that combining structure- and prediction-based methods generally yields better results in terms of diversity.

1 Introduction

The development of a novel pharmaceutical drug is a highly intricate process that can span up to a decade and incur costs exceeding US $1 billion [41, 8]. A key part of such effort involves the identification of novel drug candidates that exhibit the desired molecular properties [17]. The success in identifying drug candidates primarily depends on selecting chemical starting points that surpass a certain threshold in bioactivity toward the desired target, known as hits. High-quality hits can substantially reduce the time required to identify a viable drug candidate and be the determining factor in the success of a drug discovery campaign [29]. Designing novel pharmaceutical molecules, or *de novo* drug design, is extremely challenging given the estimated number of up to 10^{60} possible drug-like molecules [32].

Recent advances in *de novo* drug design utilize reinforcement learning (RL) to navigate this vast chemical space by fine-tuning a pre-trained generative model [28, 12, 13, 1, 23, 25]. Evaluations by Gao et al. [9] and Thomas et al. [36] have demonstrated good performance when using RL to fine-tune a pre-trained recurrent neural network (RNN) to generate molecules encoded in the Simplified Molecular Input Line Entry System (SMILES) [40]. Also, this approach is widely adopted in real-world applications in drug discovery [27]. However, RL-based *de novo* drug design methods can easily become stuck in local optima, generating structurally similar molecules– a phenomenon known as *mode collapse*. This is undesirable as it prevents the agent from discovering more diverse and potentially more promising local optima. To mitigate mode collapse, Blaschke et al. [6] introduced a count-based method that penalizes generated molecules based on their structure. When too many structurally similar molecules have been generated, the agent observes zero reward, instead of the actual extrinsic reward, for future generated molecules with the same structure. This is a popular way to avoid mode collapse for RL-based *de novo* drug design [37, 13, 24, 12].

Most work mainly focuses on avoiding mode collapse to find the most optimal solution. However, the quantitative structure-activity relationship (QSAR) models utilized for *in silico* assessment of molecules introduce uncertainties and biases due to limited training data [30]. Thus, it is important to explore numerous modes of these models to increase the chance of identifying potential drug candidates. Also, the identified (local) optimal solution might not be optimal in terms of observed safety and therapeutic effectiveness in the body. Therefore, it is meaningful to generate a diverse set of molecules. Recent work by Renz et al. [31] focuses on the generated molecules' diversity, finding superior performance of SMILES-based autoregressive models using RL to optimize the desired properties. They use a penalization method based on the work by Blaschke et al. [6] to enable diverse molecule generation. As an alternative to penalizing the extrinsic reward, previous work in RL has shown that providing intrinsic motivation to the agent can enhance the exploration [3, 7, 2]. Recent efforts by Park et al. [26] and Wang and Zhu [39] show the potential of memory- and prediction-based intrinsic motivation approaches in *de novo* drug design, demonstrating their capability to enhance the optimization of properties.

The generation of diverse sets of molecules with high (extrinsic) rewards is crucial in the drug discovery process. A diverse molecular library increases the likelihood of identifying candidates with unique and favorable pharmacological profiles, thereby enhancing the overall efficiency and success rate of drug development pipelines. While most prior research has concentrated on generating individual molecules with high (extrinsic) rewards, our work shifts the focus toward the generation of diverse molecular entities by systematically investigating various intrinsic rewards and reward penalties. This approach aims to counteract mode collapse and promote the exploration of a broader chemical space. Intrinsic rewards, inspired by human-like curiosity, encourage the RL agent to explore less familiar areas of the chemical space; while reward penalties discourage the generation of structurally similar molecules. By employing these strategies, we aim to investigate further the robustness and applicability of RL-based de novo drug design. To our knowledge, this is the first work to comprehensively study the effect such methods have on the diversity of the generated molecules. By doing so, we provide a novel framework that not only seeks optimal solutions but also ensures a wide-ranging exploration of the potential chemical space of bespoke drug candidates. This could significantly enhance the drug discovery process by providing a more diverse and promising set of molecules for further experimental validation.

Algorithm 1 Diversity-Aware RL framework

```
1: input: $I, B, \theta_{\text{prior}}, h$
2: $\mathcal{M} \leftarrow \emptyset$                                  ▷ Initialize memory
3: $\theta \leftarrow \theta_{\text{prior}}$                     ▷ The pre-trained policy is fine-tuned
4: for i=1,...,I do                                   ▷ Generative steps
5:     $L(\theta) \leftarrow 0$
6:     $\mathcal{B} \leftarrow \emptyset$
7:     for b=1,...,B do                        ▷ Generate batch of molecules
8:         $t \leftarrow 0$
9:         $a_t \leftarrow a^{(\text{start})}$                      ▷ Start token is initial action
10:        $s_{t+1} \leftarrow a_t$
11:        while $s_{t+1}$ is not terminal do
12:            $t \leftarrow t + 1$
13:            $a_t \sim \pi_\theta(s_t)$
14:            $s_{t+1} \leftarrow a_{0:t}$
15:        end while
16:        $\mathcal{B} \leftarrow \mathcal{B} \cup s_{t+1}$
17:        Observe property score $r(s_{t+1})$
18:        if $r(s_{t+1}) \geq h$ then
19:            $\mathcal{M} \leftarrow \mathcal{M} \cup \{s_{t+1}\}$
20:        end if
21:        Compute and store penalty $f(s_{t+1})$
22:    end for
23:    for $A \in \mathcal{B}$ do
24:        Compute intrinsic reward $R_I(A)$
25:        Compute diversity-aware reward $\hat{R}(A)$
26:        Compute loss $L_A(\theta)$ wrt $\hat{R}(A)$
27:        $L(\theta) \leftarrow L(\theta) + L_A(\theta)$
28:    end for
29:    Update $\theta$ by one gradient step minimizing $L(\theta)$
30: end for
31: output: $\mathcal{M}$
```

2 Problem Formulation

In this section, we introduce our framework for *de novo* drug design. The problem is string-based molecule generation, by fine-tuning a pre-trained policy. Following previous work, we formulate the generative process as a reinforcement learning problem where the task is to fine-tune a pre-trained generative model [25]. An action corresponds to adding one token to the string representation of the molecule. $\mathcal{A}$ is the set of possible actions, including a start token a^{start} and a stop token a^{stop}. The *de novo* drug design problem can be modeled as a Markov decision process (MDP). $a_t \in \mathcal{A}$ is the action taken at state s_t, the current state $s_t = a_{0:t-1}$ is defined as the sequence of performed actions up to round t, the initial action $a_0 = a^{\text{start}}$ is the start token. The transition probabilities $P(s_{t+1}|s_t, a_t) = \delta_{s_t +\!\!+ a_t}$ are deterministic, where $P(\text{terminal}|s_t, a^{\text{stop}}) = 1$, $+\!\!+$ denotes the concatenation of two sequences and δ_z denotes the dirac distribution at z. If action a^{stop} is taken, the following state is terminal, stopping the current generation process and subsequently evaluating the generated molecule. The extrinsic reward is

$$R(s_t, a_t) = R(a_{0:t}) = \begin{cases} r(s_{t+1}) & \text{if } a_t = a^{\text{stop}}, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

We let T denote the round that a terminal state is visited, i.e., $a_{T-1} = a^{\text{stop}}$. The reward $r(s_T) \in [0, 1]$ (only observable at a terminal state) measures the desired property, which we want to optimize, of molecule $A = a_{1:T-2}$. Note that in practice, the string between the start and stop tokens encodes a molecule such that $a_{1:T-2}$ is equivalent to $a_{0:T-1}$ during evaluation. The objective is to fine-tune a policy π_θ, parameterized by θ, to generate a structurally diverse set of molecules optimizing the property score $r(\cdot)$.

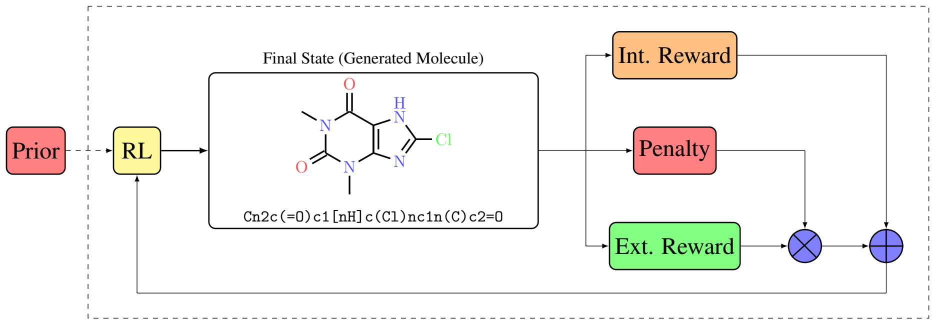
In practice, at each step i of the generative process, B full trajectories (until reaching a terminal state) are rolled out, to obtain a batch $\mathcal{B}$ of generated molecules. Also, the diversity-aware reward $\hat{R}(A)$ (see Section 3) for each molecule $A \in \mathcal{B}$ is observed by the agent and subsequently used for fine-tuning. The diversity-aware reward $\hat{R}(A)$ is computed using the penalty $f(A)$ and/or intrinsic reward R_I (depending on which reward function is used). Algorithm 1 illustrates our diversity-aware RL framework.

3 Diversity-Aware Reward Functions

In this section, we define the diversity-aware reward functions examined in this study. We investigate two approaches to encourage diversity among generated molecules by modifying the extrinsic reward: (1) penalize the extrinsic reward, and (2) provide intrinsic reward (intrinsic motivation). Moreover, we also investigate the combination of these approaches by integrating two intrinsic reward approaches with a penalty function on the extrinsic reward. Given an extrinsic reward $R(A)$, the agent will receive a reward signal at the end of the generation sequence in the form of

$$\hat{R}(A) = f(A) \times R(A) + R_I(A), \tag{2}$$

for every generated molecule A. Hence, we impose reward shaping [22] where the reward observed by the agent is a linear function of the non-linear extrinsic reward, as depicted in Figure 1. The penalty defines the importance of the extrinsic reward for each molecule to determine the exploitation rate adaptively. Sufficient exploitation is necessary to find high-quality solutions. In contrast, the intrinsic reward provides an additive bonus to encourage the agent to continually explore (independent of a molecule's extrinsic reward). We suggest several novel domain-specific penalties and intrinsic rewards, and, to the best of our knowledge, this domain-specific combination is novel. Such a linear combination avoids adding unnecessary complexity to the reward objective, but is necessarily not optimal. We want to optimize a complex extrinsic reward function while enforcing continuous exploration of a large solution space. The aim is to find a diverse set of high-quality solutions.



Prior
RL
Final State (Generated Molecule)
Cn2c(=O)c1[nH]c(Cl)nc1n(C)c2=O
Int. Reward
Penalty
Ext. Reward

Figure 1: The proposed diversity-aware RL framework for *de novo* drug design utilizes extrinsic reward penalty and intrinsic reward to improve the diversity. The RL agent is initialized to the pre-trained prior. The RL agent generates molecules, e.g., in SMILES representation as shown here, and subsequently, the penalty and/or intrinsic reward is used to modify the extrinsic rewards. Each extrinsic reward is multiplied by the corresponding penalty term (equal to one if no penalty is used), while the intrinsic reward (equal to zero if no intrinsic reward is used) is added to the product. The modified rewards are observed by the RL agent and used to update its policy.

3.1 Extrinsic Reward Penalty

We propose and examine five different functions to penalize the extrinsic reward by discretely or continuously decreasing it based on the number of previously generated structurally similar molecules. These are based on binary, error, linear, sigmoid, or hyperbolic tan functions. To the best of our knowledge, utilizing error and hyperbolic tangent functions is novel for our application. Below we define all penalty functions for clarity.

Non-binary functions will provide an incremental change of the (extrinsic) reward signal over time. Therefore, it is potentially more informative and can incrementally incentive the agent to find new local optima, while still exploiting the current local optima. On the other hand, a binary function implies a sharp change of the reward function, where the agent quickly needs to find new optima.

Identical Molecular Scaffold Penalty (IMS). The IMS penalty was first introduced by Blaschke et al. [6] and has thereafter been used in several works, e.g., [24, 13]. It is based on molecular scaffolds, which is one of the most important and commonly used concepts in medicinal chemistry. The IMS penalty uses the *molecular scaffold* defined by Bemis and Murcko [4], which is obtained by removing all side chains (or R groups). In this work, we also study the Topological scaffold, which is obtained from the molecular scaffold by converting all atom types into carbon atoms and all bonds into single bonds. Note that in this work we use the molecular scaffold since it is less general and has demonstrated good performance in earlier works [6, 13, 37]. The Topological scaffold is therefore exclusively applied to assess the molecules' diversity and is not incorporated into any penalty or intrinsic reward method defined hereafter.

Let us define the reward function $\hat{R}_{\text{IMS}}(A)$ for the IMS penalty. For each generated molecule A with a reward of at least h, its molecular scaffold S_A is computed and put in memory. A molecule fulfilling the (extrinsic) reward threshold h is commonly known as a predicted active molecule, denoted simply as *active*. If m molecules with the same scaffold have been generated, future molecules of the same scaffold are given a reward of 0 to avoid this scaffold. Given a generated molecule A with an extrinsic reward of at least h and its corresponding molecular scaffold S_A, the reward function of the IMS penalty method is then defined by

$$\hat{R}_{\text{IMS}}(A) = \begin{cases} 0 & \text{if } R(A) \geq h \text{ and } N[S_A] \geq m, \\ R(A) & \text{otherwise,} \end{cases} \tag{3}$$

where S_A is the molecular scaffold of molecule A and $N[S]$ is the number of molecules with molecular scaffold S in memory, i.e., with an extrinsic reward of at least h. If a molecule A corresponds to an extrinsic reward smaller than h, the extrinsic reward is provided to the agent without any modification. Hence, only predicted active molecules are penalized.

Error Function Identical Molecular Scaffold Penalty (ErfIMS). The Error Function Identical molecular scaffold Penalty is a soft (non-binary) version of the IMS penalty method. It uses the error function to incrementally decrease extrinsic rewards based on the number of molecules in the molecular scaffold

$$f_{\text{erf}}(A) = \left(1 + \text{erf}\left(\frac{\sqrt{\pi}}{m}\right) - \text{erf}\left(\frac{\sqrt{\pi} \times N[S_A]}{m}\right)\right), \tag{4}$$

where $\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$ is the error function. Given the threshold h for the extrinsic reward $R(\cdot)$, the reward function for a molecule A is defined by

$$\hat{R}_{\text{ErfIMS}}(A) = \begin{cases} R(A) \cdot f_{\text{erf}}(A) & \text{if } R(A) \geq h, \\ R(A) & \text{otherwise.} \end{cases} \tag{5}$$

Linear Identical molecular scaffold Penalty (LinIMS). The linear identical molecular scaffold penalty linearly reduces the extrinsic score based on the number of generated molecules in memory with the same molecular scaffold. We define the linear penalty function by

$$f_{\text{linear}}(A) = \left(1 - \frac{N[S_A]}{m}\right). \tag{6}$$

The reward function of a molecule A is defined by, given the threshold h for the extrinsic reward $R(\cdot)$,

$$\hat{R}_{\text{LinIMS}}(A) = \begin{cases} [R(A) \cdot f_{\text{linear}}(A)]^+ & \text{if } R(A) \geq h, \\ R(A) & \text{otherwise,} \end{cases} \tag{7}$$

where $[\cdot]^+$ denotes the positive part of a function. This is equivalent to the linear penalty proposed by Blaschke et al. [6].

Sigmoid Identical Molecular Scaffold Penalty (SigIMS). The sigmoid identical molecular scaffold penalty uses a sigmoid function to gradually reduce the extrinsic reward based on the number of molecules in memory with the same scaffold. A molecule is put in memory if it has an extrinsic reward of at least h. Given a molecule A, the sigmoid function in this work is defined as

$$f_\sigma(A) = 1 - \frac{1}{1 + e^{-\left(\frac{\frac{N[S_A]}{m} \cdot 2 - 1}{0.15}\right)}}. \tag{8}$$

This is equivalent to the sigmoid penalty function proposed by Blaschke et al. [6] and we therefore use the same parameters. Given a molecule A, we define the reward function defined as follows

$$\hat{R}_{\text{SigIMS}}(A) = \begin{cases} R(A) \cdot f_\sigma(A) & \text{if } R(A) \geq h, \\ R(A) & \text{otherwise.} \end{cases} \tag{9}$$

Tanh Identical Molecular Scaffold Penalty (TanhIMS). The tanh identical molecular scaffold penalty utilizes the hyperbolic tangent function to incrementally decrease the extrinsic reward based on the number of molecules generated with the same molecular scaffold, up to and including the current step (i.e., those stored in memory). For a molecule A, the following hyperbolic tangent function is used to incrementally penalize the extrinsic reward

$$f_{\tanh}(A) = 1 - \tanh\left(c_{\text{tanh}} \cdot \frac{N[S_A] - 1}{m}\right). \tag{10}$$

In the following experiments we use $c_{\text{tanh}} = 3$ since this factor implies $f_{\tanh} \approx 0$ around $N[S_A] = 25$ for a bucket size $m = 25$. For a molecule A, we define the reward function for the TanhIMS penalty as follows

$$\hat{R}_{\text{TanhIMS}}(A) = \begin{cases} R(A) \cdot f_{\tanh}(A) & \text{if } R(A) \geq h, \\ R(A) & \text{otherwise.} \end{cases} \tag{11}$$

3.2 Intrinsic Reward

We explore eight methods to provide intrinsic reward to the agent, namely diverse actives (DA), minimum distance (MinDis), mean distance (MeanDis), minimum distance to random coreset (MinDisR), mean distance to random coreset (MeanDisR), KL-UCB, random network distillation (RND) and information (Inf). To the best of our knowledge, all methods, except for RND, are novel in the context of *de novo* drug design. Below, we define the most effective methods, while the other methods are described in Appendix A.

Minimum Distance (MinDis). Minimum distance is a distance-based intrinsic reward. A bonus reward is given based on the minimum distance to previously generated diverse actives. Following the work by Renz et al. [31] but using the terminology of predicted active molecules rather than hit molecules, we define the number of *diverse actives* for distance threshold D by

$$\mu(\mathcal{H}; D) = \max_{\mathcal{C} \in \mathcal{P}(\mathcal{H})} |\mathcal{C}| \text{ s.t. } \forall x \neq y \in \mathcal{C} : d(x, y) \geq D, \tag{12}$$

where $\mathcal{H}$ is a set of predicted active molecules, $\mathcal{P}$ is the power set, $d(x, y)$ is the distance between molecules x and y. This naturally defines a set of diverse actives, i.e., a set fulfilling the above dissimilarity criteria with cardinality $\mu(\mathcal{H}; D)$. Note that this set is not necessarily unique.

Let $\mathcal{H}_i$ be a batch of generated actives in the current generative step i and $\mathcal{C}_{i-1}$ be a set of previously generated diverse actives. Then the reward function of MinDis for a molecule A and reward threshold h (of predicted active molecules) is defined as follows

$$\hat{R}_{\text{MinDis}}(A) = \begin{cases} R(A) + \min_{\tilde{A} \in \widetilde{\mathcal{C}}_{i-1}} d\left(A, \tilde{A}\right) & \text{if } R(A) \geq h, \\ R(A) & \text{otherwise,} \end{cases} \tag{13}$$

where $\widetilde{\mathcal{C}}_{i-1} := \mathcal{C}_{i-1} \cup (\mathcal{H}_i \setminus \{A\})$, and $d(x, y)$ is a distance metric between molecules x and y. In this work, we use the distance metric based on the Jaccard index [18], also known as the Tanimoto distance, widely used to measure chemical (dis-)similarity. In Appendix A.2 we introduce a similar approach, abbreviated MinDisR, where the distances are calculated to a random subset of previously generated molecules.

Mean Distance (MeanDis). Mean distance is also a distance-based intrinsic reward, but where the intrinsic reward is defined as the mean dissimilarity (distance) to previously generated diverse actives and the current batch of actives. Let $\mathcal{H}_i$ be a batch of generated actives in the current generative step i and $\mathcal{C}_{i-1}$ be a set of previously generated diverse actives (see definition above of diverse actives). We then define the reward function of MeanDis of molecule A by

$$\hat{R}_{\text{MeanDis}}(A) = \begin{cases} R(A) + \bar{d}(A; \widetilde{\mathcal{C}}_{i-1}) & \text{if } R(A) \geq h, \\ R(A) & \text{otherwise,} \end{cases} \tag{14}$$

where $\bar{d}(A; \widetilde{\mathcal{C}}_{i-1}) = \frac{\sum_{\tilde{A} \in \widetilde{\mathcal{C}}_{i-1}} d(A, \tilde{A})}{|\widetilde{\mathcal{C}}_{i-1}|}$. In Appendix A.3 we introduce a similar approach, abbreviated MeanDisR, where the distances are calculated to a random subset of previously generated molecules

Random Network Distillation (RND). Random network distillation [7] is an exploration technique in reinforcement learning that provides an intrinsic reward based on the prediction error of a neural network. Specifically, it employs a fixed, randomly initialized neural network f and a predictive neural network $\hat{f}_\phi$ trained to mimic the outputs of the fixed network. The intrinsic reward is derived from the prediction error between these two networks. This error serves as a measure of novelty, incentivizing the RL agent to explore less familiar regions of the parameter space, and potentially enhancing the exploration of less familiar regions of the chemical space. We adapt RND as an intrinsic reward for generated active molecules, i.e., molecules with an extrinsic reward of at least h. In this work, f and $\hat{f}_\phi$ have identical architecture as the pre-trained policy (see Section 4). We let the predictive network $\hat{f}_\phi$ be initialized to this pre-trained policy. For a molecule A, we define

$$f(A) = \sum_{t=1}^{T-2} \log \pi_f(a_t | s_t), \tag{15}$$

$$\hat{f}_\phi(A) = \sum_{t=1}^{T-2} \log \pi_\phi(a_t | s_t), \tag{16}$$

where $\pi_f(a_t|s_t)$ and $\pi_\phi(a_t|s_t)$ are the policies induced by the fixed and predictive network, respectively. Moreover, let $\Delta_{\hat{f}}(A; \phi)$ be the squared norm of the difference between these networks for a molecule A, defined by

$$\Delta_{\hat{f}}(A; \phi) = \|\hat{f}_\phi(A) - f(A)\|^2, \tag{17}$$

where ϕ is the weights of the prediction network $\hat{f}$ of the current generative step.

Let ϕ be the weights of the predictive network $\hat{f}$ up to the current generative step, we define the reward function of a molecule A by

$$\hat{R}_{\text{RND}}(A) = \begin{cases} R(A) + \Delta_{\hat{f}}(A; \phi) & \text{if } R(A) > h, \\ R(A) & \text{otherwise.} \end{cases} \tag{18}$$

Since $R(A) \in [0, 1]$, we rescale the prediction error over the batch of active molecules generated in the current generative step i by

$$\tilde{\Delta}_{\hat{f}}(A; \phi) = \frac{\Delta_{\hat{f}}(A; \phi) - \min_{\tilde{A} \in \mathcal{H}_i} \Delta_{\hat{f}}(\tilde{A}; \phi)}{\max_{\tilde{A} \in \mathcal{H}_i} \Delta_{\hat{f}}(\tilde{A}; \phi) - \min_{\tilde{A} \in \mathcal{H}_i} \Delta_{\hat{f}}(\tilde{A}; \phi)}. \tag{19}$$

Information (Inf). We define an information-inspired intrinsic reward function based on the number of actives in each scaffold and scaffolds generated up to and including the current generative step i. Let A be a molecule, S_A its scaffold, $N[S]$ the number of active molecules with scaffold S in memory, and $\mathcal{S}$ the set of unique molecular scaffolds in memory up to (including) current generative step i. We define the the *scaffold* (pseudo-)*probability* of molecule A by

$$\tilde{\mathbb{P}}_{\text{scaff}}(A) = \frac{N[S_A]}{|\mathcal{S}|}. \tag{20}$$

We use this scaffold probability to define the *scaffold information* by

$$\mathrm{I}_{\text{scaff}}(A) = -\log\left(\tilde{\mathbb{P}}_{\text{scaff}}(A)\right). \tag{21}$$

Given a set of active molecules $\mathcal{H}_i$ generated at the current generative step i, where $A \in \mathcal{H}_i$, the normalized scaffold information is defined by

$$R_I^{\text{Inf}}(A; \mathcal{H}_i) = \frac{\mathrm{I}_{\text{scaff}}(A) - \min\limits_{\tilde{A} \in \mathcal{H}_i} \mathrm{I}_{\text{scaff}}\left(\tilde{A}\right)}{\max\limits_{\tilde{A} \in \mathcal{H}_i} \mathrm{I}_{\text{scaff}}\left(\tilde{A}\right) - \min\limits_{\tilde{A} \in \mathcal{H}_i} \mathrm{I}_{\text{scaff}}\left(\tilde{A}\right)}. \tag{22}$$

In practice, we only normalize if $|\mathcal{H}_i| > 2$. Using the scaffold information to define the information-based intrinsic reward, we define the reward function by

$$\hat{R}_{\text{Inf}}(A) = \begin{cases} R(A) + R_I^{\text{Inf}}(A; \mathcal{H}_i) & \text{if } R(A) \geq h, \\ R(A) & \text{otherwise.} \end{cases} \tag{23}$$

3.3 Combining Penalty and Intrinsic Reward

We investigate two combinations of intrinsic reward and extrinsic reward penalty. To the best of our knowledge, combining these approaches is novel.

Tanh Random Network Distillation (TanhRND). We define a soft version of random network distillation by combining RND and TanhIMS. The extrinsic reward is penalized as defined by TanhIMS and an (non-penalized) intrinsic reward based on RND is provided to the agent. We define the TanhRND reward function by

$$\hat{R}_{\text{TanhRND}}(A) = \begin{cases} R_{\tanh}(A) + \tilde{\Delta}_{\hat{f}}(A; \phi) & \text{if } R(A) \geq h, \\ R(A) & \text{otherwise,} \end{cases} \tag{24}$$

where $R_{\tanh}(A) = f_{\tanh}(A) \cdot R(A)$.

Tanh Information (TanhInf). We also propose and examine a reward function combining (extrinsic) reward penalty and information-based intrinsic reward. We use TanhIMS to penalize the extrinsic reward and Inf to provide a (non-penalized) intrinsic reward. For a molecule A, we define the TanhInf reward function by

$$\hat{R}_{\text{TanhInf}}(A) = \begin{cases} R_{\tanh}(A) + \tilde{\mathrm{I}}_{\text{scaff}}(A; \mathcal{H}_i) & \text{if } R(A) \geq h, \\ R(A) & \text{otherwise,} \end{cases} \tag{25}$$

where h is the reward threshold, $R(\cdot)$ is the extrinsic reward, $\mathcal{H}_i$ is the set of active molecules generated in the current generative step.

4 Experimental Evaluation

We now describe experiments designed to examine the efficacy of our diversity-aware reward functions.

4.1 Experimental Setup

Here we run experiments on one extrinsic reward function, namely the Glycogen Synthase Kinase 3 Beta (GSK3β) oracle provided by Therapeutics Data Commons [15, 16, 38]. This is a well-established molecule binary bioactivity label optimization task. To compute an extrinsic reward in $[0, 1]$, the oracle utilizes a random forest classifier trained on data from the ExCAPE-DB dataset [35] using extended-connectivity fingerprints with radius 3 [33]. This oracle only provides rewards to valid molecules and, therefore, we assign invalid molecules an extrinsic reward of -1 to distinguish them from the penalized molecules. Additionally, previously generated (predicted) active molecules are each assigned zero reward. We also investigate the c-Jun N-terminal Kinases-3 (JNK3) and the Dopamine Receptor D2 (DRD2) oracles in extended works [**? ?**].

For distance-based intrinsic rewards, the Jaccard distance is computed based on Morgan fingerprints [33] computed by RDKit [21], with a radius of 2 and a size of 2048 bits. The distance threshold for the diverse actives-based approaches is fixed to $D = 0.7$, as suggested by Renz et al. [31] since there is a significant decrease in the probability of similar bioactives beyond this threshold [19]. See Appendix B for all experimental details.

The molecular generative model builds directly on REINVENT [34, 25, 5, 24] and consists of a long short-term memory (LSTM) network [14] using SMILES to represent molecules as text strings. REINVENT utilizes an on-policy RL algorithm optimizing the policy π_θ to generate molecules with higher reward. The algorithm is based on the *augmented log-likelihood* defined by

$$\log \pi_{\theta_{\text{aug}}}(A) := \sum_{t=1}^{T-2} \log \pi_{\theta_{\text{prior}}}(a_t|s_t) + \sigma R(A), \tag{26}$$

where $A = a_{1:T-2}$ is a generated molecule, σ is a scalar value, $\pi_{\theta_{\text{prior}}}$ is the (fixed) prior policy. We use the pre-trained policy by Blaschke et al. [5] as the prior policy. It is pre-trained on the ChEMBL database [11] to generate drug-like bioactive molecules. The action space $\mathcal{A}$ consists of 34 tokens including start and stop tokens, i.e., $|\mathcal{A}| = 34$. The policy π_θ is optimized by minimizing the squared difference between the augmented log-likelihood and policy likelihood given a sampled batch $\mathcal{B}$ of SMILES

$$L(\theta) = \frac{1}{|\mathcal{B}|} \sum_{a_{1:T-2} \in \mathcal{B}} \left(\log \pi_{\theta_{\text{aug}}}(a_{1:T-2}) - \sum_{t=1}^{T-2} \log \pi_\theta(a_t|s_t) \right)^2 . \tag{27}$$

Previous work has shown that minimizing this loss function is equivalent to maximizing the expected return, as for policy gradient algorithms [13]. Evaluations by both Gao et al. [9] and Thomas et al. [36] have concluded good performance compared to both RL-based and non-RL-based approaches for *de novo* drug design. The generative process has a budget of $I = 2000$ generative steps, where a batch of $|\mathcal{B}| = 128$ molecules is generated in each step. Each experiment is evaluated by 20 independent runs of the RL fine-tuning process.

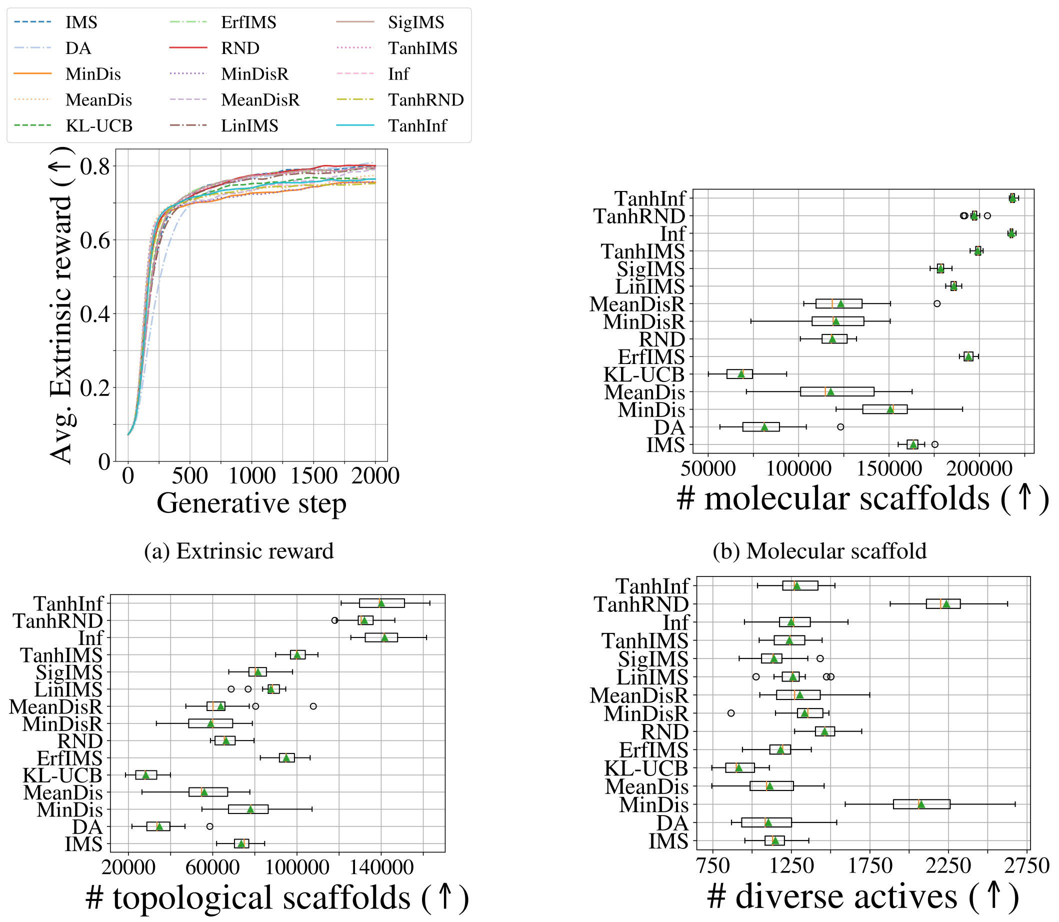
4.2 Comparison of Diversity-Aware Reward Functions

We evaluate the quality by extrinsic reward per generative step, and diversity by the number of molecular scaffolds, topological scaffolds and diverse actives after $I = 2000$ generative steps.

GSK3β For the GSK3β oracle, Figure 2a compares the average extrinsic reward (moving average using a window size of 101), over 20 independent runs, per generative step. We observe that the extrinsic rewards converge to comparable values across the diversity-aware reward functions. To evaluate the diversity, Figures 2b to 2d display boxplots comparing the number of molecular scaffolds, topological scaffolds and diverse actives, respectively, over 20 independent runs with a budget of 2000 generative steps. Only active molecules with an extrinsic reward of at least $h = 0.5$ are displayed, where orange lines and green triangles display the median and mean, respectively. Inf and TanhInf generate substantially more molecular and topological scaffolds; whereas TanhRND is also among the top methods. The 7 best methods for molecular scaffolds display a low variability, among the 20 reruns, in terms of the number of generated molecular scaffolds and generate considerably more molecular scaffolds. For the 3 best methods with regard to topological scaffolds, the variability in terms of topological scaffolds is similar to the other methods, but the top-performing approaches are still substantially better than the other methods.

MinDis and TanhRND generate more diverse actives than the other methods we have investigated, but MinDis shows a higher variability among the reruns. Overall, we observe a higher variability for the number of topological scaffolds and diverse actives among the reruns, which can partially be explained by the fact that the scaffolds-based methods only consider molecular scaffolds, where more molecular scaffolds do not necessarily lead to more topological scaffolds and diverse actives.

In general, applying RND alone yields among the highest diversities, but the combination of TanhIMS and RND can improve the diversity of TanhIMS, especially for topological and diverse actives. Hence, we observe that proper structure-based scaling of the extrinsic reward can help to improve diversity, while adding a prediction-based intrinsic reward, i.e., RND, can further enhance and stabilize diversity. Combining Inf, a structure-based intrinsic reward, with TanhIMS does not seem to



(a) Extrinsic reward

(b) Molecular scaffold

(c) Topological scaffold

(d) Diverse actives

Figure 2: Evaluation on the GSK3β oracle.

enhance the diversity further, but also does not decrease the diversity. We also observe enhanced performance in scaffold diversity for the non-binary penalty functions alone. Still, this structural information does not improve the similarity-based diversity, i.e., diverse actives.

5 Conclusion

Our comprehensive study proposes and evaluates several novel intrinsic rewards and reward penalties to enhance the diversity of *de novo* drug design using reinforcement learning (RL). Our approach balances exploration and exploitation to promote a more diverse generation of molecules. Our results consistently show that methods incorporating both intrinsic reward and reward penalty generate significantly more diverse actives, molecular scaffolds, and topological scaffolds. In particular, using structure-based scaling of the extrinsic reward and a prediction-based method to encourage exploration in the agent's state space, we observe improved diversity in terms of both structure and similarity. Yet, a single type of information does not fully enhance diversity. Our work opens several future directions for studying how to incorporate domain and agent information into the reward signal efficiently.

Acknowledgments

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. The experimental evaluation was enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS), partially funded by the Swedish Research Council through grant agreement no. 2022-06725.

References

[1] S. R. Atance, J. V. Diez, O. Engkvist, S. Olsson, and R. Mercado. De novo drug design using reinforcement learning with graph-based deep generative models. *Journal of chemical information and modeling*, 62(20):4863–4872, 2022.

[2] A. P. Badia, P. Sprechmann, A. Vitvitskyi, D. Guo, B. Piot, S. Kapturowski, O. Tieleman, M. Arjovsky, A. Pritzel, A. Bolt, and C. Blundell. Never give up: Learning directed exploration strategies, 2020. URL https://arxiv.org/abs/2002.06038.

[3] M. G. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. Unifying count-based exploration and intrinsic motivation. In D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 1471–1479, 2016.

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

[5] T. Blaschke, J. Arús-Pous, H. Chen, C. Margreitter, C. Tyrchan, O. Engkvist, K. Papadopoulos, and A. Patronov. Reinvent 2.0: an ai tool for de novo drug design. *Journal of chemical information and modeling*, 60(12):5918–5922, 2020.

[6] T. Blaschke, O. Engkvist, J. Bajorath, and H. Chen. Memory-assisted reinforcement learning for diverse molecular de novo design. *Journal of cheminformatics*, 12(1):68, 2020.

[7] Y. Burda, H. Edwards, A. Storkey, and O. Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

[8] J. A. DiMasi, H. G. Grabowski, and R. W. Hansen. Innovation in the pharmaceutical industry: new estimates of r&d costs. *Journal of health economics*, 47:20–33, 2016.

[9] W. Gao, T. Fu, J. Sun, and C. W. Coley. Sample efficiency matters: A benchmark for practical molecular optimization. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[10] A. Garivier and O. Cappé. The kl-ucb algorithm for bounded stochastic bandits and beyond. In S. M. Kakade and U. von Luxburg, editors, *Proceedings of the 24th Annual Conference on Learning Theory*, volume 19 of *Proceedings of Machine Learning Research*, pages 359–376, Budapest, Hungary, 09–11 Jun 2011. PMLR. URL https://proceedings.mlr.press/v19/garivier11a.html.

[11] A. Gaulton, A. Hersey, M. Nowotka, A. P. Bento, J. Chambers, D. Mendez, P. Mutowo, F. Atkinson, L. J. Bellis, E. Cibrián-Uhalte, et al. The chembl database in 2017. *Nucleic acids research*, 45(D1):D945–D954, 2017.

[12] H. Gummesson Svensson, C. Tyrchan, O. Engkvist, and M. Haghir Chehreghani. Utilizing reinforcement learning for de novo drug design. *Machine Learning*, 113(7):4811–4843, 2024.

[13] J. Guo and P. Schwaller. Augmented memory: Sample-efficient generative molecular design with reinforcement learning. *Jacs Au*, 2024.

[14] S. Hochreiter. Long short-term memory. *Neural Computation MIT-Press*, 1997.

[15] K. Huang, T. Fu, W. Gao, Y. Zhao, Y. Roohani, J. Leskovec, C. W. Coley, C. Xiao, J. Sun, and M. Zitnik. Therapeutics data commons: Machine learning datasets and tasks for drug discovery and development. *Proceedings of Neural Information Processing Systems, NeurIPS Datasets and Benchmarks*, 2021.

[16] K. Huang, T. Fu, W. Gao, Y. Zhao, Y. Roohani, J. Leskovec, C. W. Coley, C. Xiao, J. Sun, and M. Zitnik. Artificial intelligence foundation for therapeutic science. *Nature Chemical Biology*, 2022.

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

[18] P. Jaccard. The distribution of the flora in the alpine zone. 1. *New phytologist*, 11(2):37–50, 1912.

[19] S. Jasial, Y. Hu, M. Vogt, and J. Bajorath. Activity-relevant similarity values for fingerprints and implications for similarity searching. *F1000Research*, 5(591), 2016. doi: 10.12688/f1000research.8357.2.

[20] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

[21] G. Landrum. Rdkit: Open-source cheminformatics. http://www.rdkit.org, 2006.

[22] A. D. Laud. *Theory and application of reward shaping in reinforcement learning*. PhD thesis, University of Illinois at Urbana-Champaign, 2004.

[23] X. Liu, K. Ye, H. W. van Vlijmen, M. T. Emmerich, A. P. IJzerman, and G. J. van Westen. Drugex v2: de novo design of drug molecules by pareto-based multi-objective reinforcement learning in polypharmacology. *Journal of cheminformatics*, 13(1):85, 2021.

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

[25] M. Olivecrona, T. Blaschke, O. Engkvist, and H. Chen. Molecular de-novo design through deep reinforcement learning. *Journal of cheminformatics*, 9:1–14, 2017.

[26] J. Park, J. Ahn, J. Choi, and J. Kim. Mol-air: Molecular reinforcement learning with adaptive intrinsic rewards for goal-directed molecular generation. *Journal of Chemical Information and Modeling*, 65(5):2283–2296, 2025.

[27] W. R. Pitt, J. Bentley, C. Boldron, L. Colliandre, C. Esposito, E. H. Frush, J. Kopec, S. Labouille, J. Meneyrol, D. A. Pardoe, F. Palazzesi, A. Pozzan, J. M. Remington, R. Rex, M. Southey, S. Vishwakarma, and P. Walker. Real-world applications and experiences of ai/ml deployment for drug discovery. *Journal of Medicinal Chemistry*, 2025. doi: 10.1021/acs.jmedchem.4c03044. URL https://doi.org/10.1021/acs.jmedchem.4c03044. PMID: 39772505.

[28] M. Popova, O. Isayev, and A. Tropsha. Deep reinforcement learning for de novo drug design. *Science advances*, 4(7):eaap7885, 2018.

[29] J. Quancard, A. Vulpetti, A. Bach, B. Cox, S. M. Guéret, I. V. Hartung, H. F. Koolman, S. Laufer, J. Messinger, G. Sbardella, et al. The european federation for medicinal chemistry and chemical biology (efmc) best practice initiative: Hit generation. *ChemMedChem*, 18 (9):e202300002, 2023.

[30] P. Renz, D. Van Rompaey, J. K. Wegner, S. Hochreiter, and G. Klambauer. On failure modes in molecule generation and optimization. *Drug Discovery Today: Technologies*, 32:55–63, 2019.

[31] P. Renz, S. Luukkonen, and G. Klambauer. Diverse hits in de novo molecule design: Diversity-based comparison of goal-directed generators. *Journal of Chemical Information and Modeling*, 64(15):5756–5761, 2024.

[32] J.-L. Reymond. The chemical space project. *Accounts of chemical research*, 48(3):722–730, 2015.

[33] D. Rogers and M. Hahn. Extended-connectivity fingerprints. *Journal of chemical information and modeling*, 50(5):742–754, 2010.

[34] M. H. Segler, T. Kogej, C. Tyrchan, and M. P. Waller. Generating focused molecule libraries for drug discovery with recurrent neural networks. *ACS central science*, 4(1):120–131, 2018.

[35] J. Sun, N. Jeliazkova, V. Chupakhin, J.-F. Golib-Dzib, O. Engkvist, L. Carlsson, J. Wegner, H. Ceulemans, I. Georgiev, V. Jeliazkov, et al. Excape-db: an integrated large scale dataset facilitating big data analysis in chemogenomics. *Journal of cheminformatics*, 9:1–9, 2017.

[36] M. Thomas, N. M. O'Boyle, A. Bender, and C. D. Graaf. Re-evaluating sample efficiency in de novo molecule generation, 2022. URL https://arxiv.org/abs/2212.01385.

[37] M. Thomas, N. M. O'Boyle, A. Bender, and C. De Graaf. Augmented hill-climb increases reinforcement learning efficiency for language-based de novo molecule generation. *Journal of cheminformatics*, 14(1):68, 2022.

[38] A. Velez-Arce, X. Lin, K. Huang, M. M. Li, W. Gao, B. Pentelute, T. Fu, M. Kellis, and M. Zitnik. Signals in the cells: Multimodal and contextualized machine learning foundations for therapeutics. In *NeurIPS 2024 Workshop on AI for New Drug Modalities*, 2024. URL https://openreview.net/forum?id=kL8dlYp6IM.

[39] J. Wang and F. Zhu. Exselfrl: An exploration-inspired self-supervised reinforcement learning approach to molecular generation. *Expert Systems with Applications*, page 125410, 2024.

[40] D. Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1): 31–36, 1988.

[41] O. J. Wouters, M. McKee, and J. Luyten. Estimated research and development investment needed to bring a new medicine to market, 2009-2018. *JAMA*, 323(9):844–853, 2020.

[42] Y. Xie, Z. Xu, J. Ma, and Q. Mei. How much space has been explored? measuring the chemical space covered by databases and machine-generated molecules. In *11th International Conference on Learning Representations (ICLR 2023)*. International Conference on Learning Representations (ICLR), 2023.

A Additional Intrinsic Reward Methods

Here we present details on some of the less effective intrinsic reward methods, since the main papers focus on the most effective methods.

A.1 Diverse Actives (DA)

We define the diverse actives intrinsic reward based on the diverse hits metric by Renz et al. [31], which is based on #Circles metric proposed by Xie et al. [42]. Given a set of possible centers, the #Circles metric counts the number of non-overlapping circles with equivalent radius in the distance metric space. An *active* molecule is defined as a molecule with a reward of at least h. Following the work by Renz et al. [31] but using the terminology of predicted active molecules rather than hit molecules, we define the number of *diverse actives* for distance threshold D by

$$\mu\left(\mathcal{H}; D\right) = \max_{\mathcal{C} \in \mathcal{P}(\mathcal{H})} |\mathcal{C}| \text{ s.t. } \forall x \neq y \in \mathcal{C} : d(x, y) \geq D, \tag{28}$$

where $\mathcal{H}$ is a set of predicted active molecules, $\mathcal{P}$ is the power set, $d(x, y)$ is the distance between molecules x and y. Note that there is a substantial difference between a set of actives and a set of diverse actives. Determining the number of diverse actives is analogous to determining the packing number of the set $\mathcal{H}$ in the distance metric space [31]. Let Δ_μ be the difference in the number of diverse actives, between two sets $\mathcal{H}$ and $\widetilde{\mathcal{H}}$ of active molecules, defined by

$$\Delta_\mu\left(\mathcal{H}, \widetilde{\mathcal{H}}; D\right) = \mu\left(\mathcal{H}; D\right) - \mu\left(\widetilde{\mathcal{H}}; D\right). \tag{29}$$

Moreover, let $\mathcal{H}_i$ be the batch of generated actives in the current generative step i, $\mathcal{C}_{i-1}$ the set of previously generated diverse actives and $\mathcal{C}_i = \mathcal{C}_{i-1} \cup \mathcal{H}_i$. We define the reward function using diverse actives as an intrinsic reward by

$$\hat{R}_{\text{DA}}(A) = \begin{cases} R(A) + \Delta_\mu\left(\mathcal{C}_i, \mathcal{C}_{i-1}; D\right) & \text{if } A \in \mathcal{H}_i, \\ R(A) & \text{otherwise.} \end{cases} \tag{30}$$

Note that the intrinsic reward $\Delta_\mu\left(\mathcal{C}_{i-1} \cup \mathcal{H}_i, \mathcal{C}_{i-1}; D\right)$ is sparse since a new batch does not necessarily increase the number of non-overlapping circles. On the other hand, the intrinsic reward can be substantially larger than the extrinsic reward $R(A) \in [0, 1]$, providing strong intrinsic motivation towards a specific area.

A.2 Minimum Distance to Random Coreset (MinDisR)

Minimum distance to random coreset is a distance-based intrinsic reward similar to MinDis. The difference is that MinDisR is based on the distance between the actives from the current generative step and a random set of previously generated actives. Given the set $\mathcal{H}_i$ of actives generated in the current generative step i, a molecule $A \in \mathcal{H}_i$ generated in the current step, and a (uniform) random set $\mathcal{X}$ of previously generated actives, the reward function is defined by

$$\hat{R}_{\text{MinDisR}}(A) = \begin{cases} R(A) + \min\limits_{\tilde{A} \in \widetilde{\mathcal{X}}} d\left(A, \tilde{A}\right) & \text{if } R(A) \geq h \\ R(A) & \text{otherwise,} \end{cases} \tag{31}$$

where $\widetilde{\mathcal{X}} := \mathcal{X} \cup (\mathcal{H}_i \setminus \{A\})$ and $d(x, y)$ is the distances between molecules x and y. In this work, $\mathcal{X}$ consists of 5000 randomly sampled actives, uniformly sampled without replacement from the set of previously generated actives $\mathcal{H}_{i-1}$. If 5000 actives have not been generated at generative step i, all previously generated actives are used.

A.3 Mean Distance to Random Coreset (MeanDisR)

Mean distance to random coreset is an intrinsic reward given by the mean distance to a random coreset of actives. Given a set $\mathcal{H}_i$ of actives generated in the current generative step i, a molecule A generated in the current generative step and a random set $\mathcal{X}$ of previously generated actives, we define the reward function of MeanDisR as

$$\hat{R}_{\text{MeanDisR}}(A) = \begin{cases} R(A) + \bar{d}(A; \tilde{\mathcal{X}}) & \text{if } R(A) \geq h \\ R(A) & \text{otherwise.} \end{cases} \tag{32}$$

A.4 KL-UCB

The KL-UCB intrinsic reward is based on the KL-UCB algorithm by Garivier and Cappé [10] for the multi-armed bandit problem. It defines an improved upper confidence bound to handle the trade-off between exploration and exploitation in the multi-armed bandit problem. In our study, this trade-off is crucial as the agent must determine the optimal balance between exploiting and exploring various molecular structures. We compute the KL-UCB intrinsic reward for a molecule A by

$$R_I^{\text{UCB}}(A) = \max\left\{q \in [0,1] : N[S_A]\text{KL}\left(\frac{\Sigma[S_A]}{N[S_A]}, q\right) \leq \log(n) + c\log(log(n))\right\}, \tag{33}$$

where n is the total number of generated molecules up to and including the current generative step i, KL $(p,q) = p\log\frac{p}{q} + (1-p)\log\frac{1-p}{1-q}$ is the Bernoulli Kullback-Leibler divergence and $c = 0$ is used for optimal performance in practice [10]. Moreover, S_A is the scaffold of molecule A, $\Sigma[S]$ is the sum of rewards of actives with scaffold S in memory and $N[S]$ is the total number of actives with scaffold S in memory. Given a molecule A, extrinsic reward $R(A)$ and reward threshold h, we define the reward function of KL-UCB by

$$\hat{R}_{\text{KL-UCB}}(A) = \begin{cases} R_I^{\text{UCB}}(A) & \text{if } R(A) \geq h, \\ R(A) & \text{otherwise.} \end{cases} \tag{34}$$

This implies that actives are given a reward corresponding to the upper confidence bound of the mean extrinsic reward of the actives with the same scaffold.

B Experimental Details

The policy π_θ is a neural network with an embedding layer and a subsequent multi-layer long short-term memory (LSTM) [14] recurrent neural network (RNN). The policy's action probabilities are obtained by feeding the LSTM output through a fully connected layer and a subsequent softmax layer. Finetuning of the policy network is done on a single NVIDIA A40 GPU with 64GB RAM using PyTorch 2.4.1 and CUDA 12.4. At the end of each generative step, the parameters of the embedding, LSTM, and fully-connected layers are updated by performing one gradient step on the generated batch of molecules. To perform a gradient step update, we use Adam[20] with a learning rate of 10^{-4} and keep other default parameters in Adam. Oracle functions, providing the extrinsic rewards, provided by PyTDC 0.4.17. Fingerprints are computed using RDKit 2023.9.6. Parameter σ of the augmented likelihood is automatically adjusted as described in Appendix B.1, initialized to the value of σ_{init}. Hyperparameters utilized in the experiments are displayed in Table 1. The source code is available as part of a framework for SMILES-based *de novo* drug design[1].

B.1 Automtic update of σ

The scalar parameter σ of the augmented likelihood is automatically updated based on the difference between the agent likelihood and augmented likelihood. This was introduced in REINVENT 3.0[2], called margin guard. We follow the update procedure used in REINVENT 3.0, as described below.

For a generative step i, the difference between defined by

$$\delta_\sigma = \frac{1}{|\mathcal{K}_{i-1}|} \sum_{a_{1:T-2} \in \mathcal{K}_{i-1}} \left(\log \pi_{\theta_{\text{aug}}}(a_{1:T}). - \sum_{t=1}^{T-2} \log \pi_\theta(a_t|s_t)\right), \tag{35}$$

where $\mathcal{K}_{i-1}$ is all molecules generated before generative step i. The σ parameter is initialized to the value σ_{init}. After at least w_σ generative steps, the parameter σ is adjusted if $\delta_\sigma > m_\sigma$. Given desirable minimum score $D_\sigma^{\min}$, let

$$D_\sigma = \max\left(\frac{1}{|\mathcal{G}_{i-1}|} \sum_{a_{1:T-2} \in \mathcal{G}_{i-1}} r\left(a_{1:T-2}\right), D_\sigma^{\min}\right), \tag{36}$$

[1]https://github.com/MolecularAI/SMILES-RL

[2]https://github.com/MolecularAI/Reinvent/tree/v3.0

Table 1: Parameters and corresponding values utilized in the experiments.

Parameter	Value
Num. actions $\|\mathcal{A}\|$	34
Extrinsic reward threshold h	0.5
KL-UCB parameter c	0
c_{tanh}	3
Distance threshold D	0.7
Bucket size m	25
Batch size $\|\mathcal{B}\|$	128
Num. generative steps I	2000
Learning rate α	10^{-4}
layer size	512
Num. recurrent layers	3
Embedding layer size	256
Optimizer	Adam[20]
σ_{init}	128
m_σ	50
w_σ	10
$D_\sigma^{\min}$	0.15
$T_{\max}$	256
Num. independent runs	20

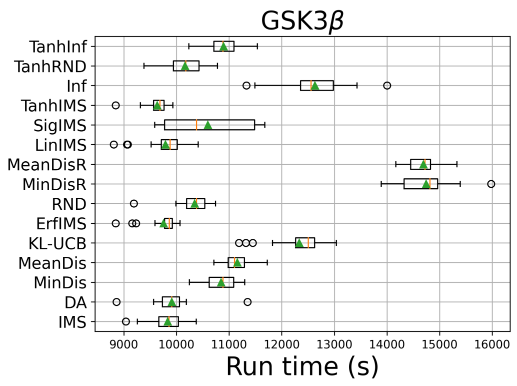


Figure 3: Displays boxplots of the run time over 20 independent reruns for the GSK3β oracle.

where $\mathcal{G}_{i-1}$ is the set of previously generated molecules. If σ is updated, it is increased to

$$\sigma = \max\left(\sigma, \frac{\delta_\sigma}{D_\sigma}\right) + m_\sigma. \tag{37}$$

If σ is adjusted, the weights θ of the policy π_θ are re-initialized to the pre-trained (prior) weights.

B.2 Run time

Figure 3 shows the runtimes over 20 independent reruns for the GSK3β oracle. As expected, the distance-based strategies generally display a longer run time, since they involve computing pair-wise distances against a (potentially large) set. The extrinsic penalty functions involve a conversion and lookup for scaffolds, which is usually scalable. Also, KL-UCB can require a long runtime since it involves solving an optimization problem at each iteration. Interestingly, the information-based approach, i.e., Inf, displays one of the longest runtimes. One possible explanation is that it consistently generates more scaffolds than the other scaffold-based strategies, leading to more scaffolds being in the memory and, consequently, a longer run time for the scaffold lookup. However, it needs to be further analyzed to establish the real cause.

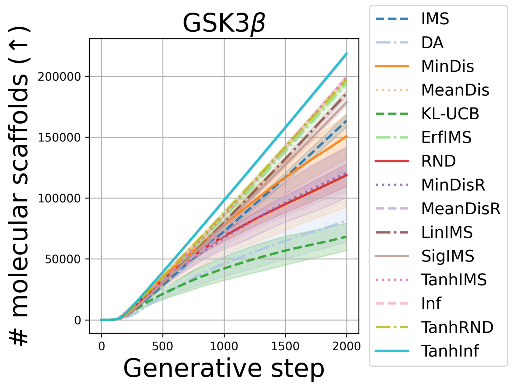


Figure 4: Total number of molecular scaffolds generated up to and including generative step i for the GSK3β oracle. Each line shows the mean over 20 reruns and the shaded region shows the sample standard deviation.

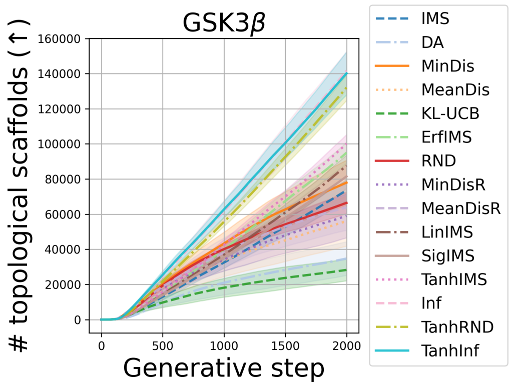


Figure 5: Total number of Topological scaffolds generated up to and including generative step i for the GSK3β oracle. Each line shows the mean over 20 reruns and the shaded region shows the sample standard deviation.

C Diversity per Generative Step

In this section, we display the Cumulative number of molecular scaffolds, Topological Scaffolds and Diverse Hits per generative step i. We display the mean and sample standard deviation over 20 independent runs. Each experiment is evaluated on a budget of $I = 2000$ generative steps. In the main text, we display the total numbers after this budget of generative steps.

C.1 Molecular Scaffolds

Figure 4 shows the cumulative number of molecular scaffolds, across 20 independent runs, per generative step i. TanhInf is consistently the top diversity-aware reward function across all extrinsic reward functions (oracles). After 500 steps on the GSK3β and oracle, both Inf and TanhInf can generate significantly more molecular scaffolds per generative step than the other diversity-aware reward functions. For the GSK3β experiments, the mean lines Inf and TanhInf almost fully overlap in terms of molecular scaffolds and, therefore, it is difficult to notice the line representing Inf. Moreover, TanhRND seems to be the third-best option in terms of the number of molecular scaffolds generated.

C.2 Topological Scaffolds

Figure 5 shows the cumulative number of Topological scaffolds, across 20 independent runs, per generative step i. After around 750 generative steps, the diversity-aware reward functions TanhInf, TahnRND and Inf consistently generate more Topological scaffolds than the other functions.

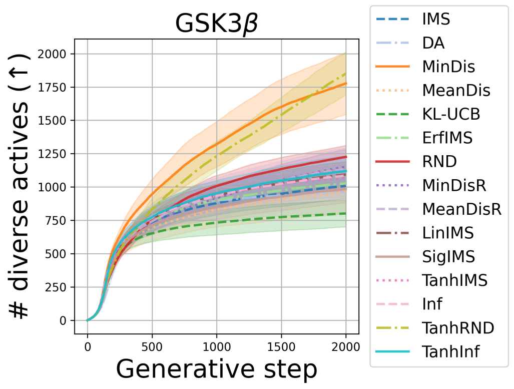


Figure 6: Total number of diverse actives generated up to and including generative step i for the GSK3β oracle. Each line shows the mean over 20 reruns and the shaded region shows the sample standard deviation.

C.3 Diverse Actives

Figure 6 shows the cumulative number of diverse actives, across 20 independent runs, per generative step i. The diversity-aware reward function TanhRND can generate substantially more diverse activities per generative step i across all oracles.