



CHALMERS
UNIVERSITY OF TECHNOLOGY



UNIVERSITY OF GOTHENBURG

Master Thesis

Iterative Clustering for Memory-Assisted Reinforcement Learning in De Novo Drug Design Language Models

Master's Thesis in Data Science & Artificial Intelligence

JAN MARTEN WINKLER

MASTER'S THESIS 2026

Master Thesis

Iterative Clustering for Memory-Assisted Reinforcement
Learning in De Novo Drug Design Language Models

JAN MARTEN WINKLER



UNIVERSITY OF
GOTHENBURG



CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden, 2026

Master Thesis

Iterative Clustering for Memory-Assisted Reinforcement Learning in De Novo Drug
Design Language Models

JAN MARTEN WINKLER

© JAN MARTEN WINKLER, 2026

Supervisor:	Morteza Haghir Chehreghani, Department of Computer Science and Engineering
Industry Supervisor:	Hannes Löffler, AstraZeneca AB
Examiner:	Ola Engkvist, Department of Computer Science and Engineering

Master's Thesis 2026

Department of Computer Science and Engineering

Chalmers University of Technology

SE-412 96 Gothenburg

Telephone +46 31-772 10 00

Gothenburg, Sweden, 2026

Master Thesis

Iterative Clustering for Memory-Assisted Reinforcement Learning in De Novo Drug Design Language Models

JAN MARTEN WINKLER

Department of Computer Science and Engineering
Chalmers University of Technology

Abstract

Reinforcement learning for de novo molecular design often suffers from mode collapse, where the policy rapidly concentrates on narrow regions of chemical space at the expense of structural diversity. This thesis presents a modular approach for combining experience replay and clustering approaches to enhance diversity within REINVENT. Across three targets, an implementation based on BitBIRCH clustering generally improved fingerprint-based diversity and produced more diverse high-scoring molecules than baseline and scaffold-based methods. The results also reveal a trade-off between diversity and score optimization. Experience replay accelerates early learning but may restrict exploration, while combining it with BitBIRCH partially mitigates this effect and can improve diversity further. Overall, the proposed method is effective but target dependent, emphasizing the need to evaluate molecular quality and multiple forms of diversity together.

Acknowledgements

First of all I would like to thank my advisor Hannes Löffler for his guidance and support throughout this research. I would also like to thank my academic supervisor Morteza Haghiri Chehreghani for his valuable feedback and encouragement and fellow master thesis workers at AstraZeneca for the collaborative environment. Particularly Borja Medina de las Heras, Malva Eklöv and Javier Gallardo Sáenz gave valuable inputs and discussions that greatly contributed to this work. Finally, I am grateful to my family and friends for their continuous support and motivation.

Use of Generative AI: This thesis utilized generative AI tools to assist in coding, drafting, reviewing and refining text, ensuring clarity and coherence. All AI-generated content was reviewed and edited by the author to maintain academic integrity and originality.

Jan Marten Winkler, Gothenburg, 2026-09-24

Contents

List of Figures	xi
List of Tables	xv
1 Introduction	1
1.1 Aim	2
1.2 Risks & Ethical considerations	3
2 Background	4
2.1 Molecular Representations	4
2.1.1 Graph Representations	4
2.1.2 SMILES	4
2.1.3 Fingerprints	5
2.2 Molecular Generation using a Language Model	6
2.2.1 Recurrent Neural Networks	7
2.2.2 Pretraining a Prior model	7
2.2.3 Post-training using Reinforcement Learning	8
2.2.4 Mode Collapse and Lack of Diversity	9
2.3 Measuring Diversity	10
2.3.1 Tanimoto Similarity	11
2.3.2 iSIM: average similarity	11
2.3.3 Unique Hits / Richness	12
2.3.4 #Circles	12
2.3.5 Unique Scaffolds	13
2.3.6 BitBIRCH Clustering	14
2.4 Experience Replay	16
2.5 Importance Sampling	17
3 Methods	19
3.1 Diversity Penalties	19
3.1.1 The Role of Clustering	20
3.1.2 Scaffold-Based Penalties	21
3.1.3 BitBIRCH Penalty	21
3.2 Intrinsic Rewards	21

3.2.1	Random Network Distillation (RND)	22
3.2.2	Information Entropy	22
3.3	Modular Architecture	23
3.4	Sample Reweighing within Experience Replay	24
3.5	Combining Diversity Penalties and Experience Replay	25
3.6	Experimental Setup	25
4	Results	27
4.1	Dopamine Type 2 Receptor (DRD2)	27
4.2	$gsk3\beta$	30
4.3	Jnk3	33
4.4	Diversity within Experience Replay	36
4.5	Sample Reweighing and Experience Replay	38
5	Discussion	40
5.1	Conclusion	41
5.2	Future Work	42
	Bibliography	43
A	Alternative Methods and Approaches	I
A.1	Kmeans	I
A.2	KCentroids	II
A.3	Taylor-Butina	II
A.4	Extended Experiments	II
B	Extended Results	IV
B.1	Impact of Experience Memory and Sampling Size	IV
B.2	Diversity in Experience Replay	V

List of Figures

Figure 1	High level architectural difference between traditional drug discovery starting from a finite library and consequent screening and de novo design using goal-oriented Reinforcement Learning to explore a vast space. . . .	2
Figure 2	SMILES representations of several simple and more complex molecular structures illustrating how circles are represented using a numbering systems and side chains with parentheses (adapted from Blay et al. [1]) .	6
Figure 3	Schematic representation of a Recurrent Neural Network (RNN). The key functionality and distinction to neural networks is in the hidden state that gets passes as an additional input in the network in order to generate predictions for the next token. The transformation W for the hidden state are learned and the state of the previous token is passed to the next token.	7
Figure 4	Schematic representation of the RL flow. The prior network is trained on a large SMILES database like ChEMBL, the network is frozen and copied to an Agent network. This network generates new SMILES sequences in batches. We score the molecules using the scoring function and combine the scores for the entire batch with the likelihood predicted by the prior network for regularization, which Olivecrona et al. named "Augmented Likelihood" [2]. Then we update the agent by using mean square error between the agent likelihood and the augmented likelihood and stochastic gradient descent.	9
Figure 5	Differences between regular and memory assisted RL as proposed by Blaschke et al. [3]. Without a memory, the RL may get stuck on local maxima, doing too much exploitation, leading to mode collapse. With a memory we can adapt the loss landscape dynamically and reduce mode collapse once we find too many molecules in a similar high scoring area. RL can then explore and find other high scoring areas.	10
Figure 6	Example of the #Circles metric in 2D. The circles are aligned in a maximal way around the molecules in the fingerprint space, such that all points are included within at least one circle, but no two circles have overlapping centers. This does capture the diversity of the set, as a higher number of circles (shown in blue) indicates that the molecules are more spread out in the space and less similar (opposed to the green circles) to each other [4].	13
Figure 7	The scaffold and topological scaffold of an example molecule. The scaffold is the core ring structure with side chains removed and the topological	

	scaffold is an unlabeled version without atom or bond descriptors (Adapted from Svensson et al [5])	14
Figure 8	Schematic representation of the BitBIRCH tree structure. Each node includes a fixed set of binary features (BF) that summarize the included molecules. The leaf BF represent clusters of molecules shown in purple. .	15
Figure 9	Adapted learning loop with added Replay Buffer. The agent keeps a fixed memory of molecules and their negative log likelihood (NLL) sorted by score and periodically samples a few to use alongside the current batch to update the agent (adapted from Guo et al. [6]).	16
Figure 10	Modular architecture for diversity penalties and intrinsic rewards. The diversity penalty can be based on different clustering methods and combined with a penalty function and optionally an intrinsic reward. .	23
Figure 11	Selection of penalty functions for score penalties. The step function is a hard penalty, while the sigmoid, linear, tanh and erf functions are soft penalties that increase smoothly as the occupancy of a cluster increases until a set maximum is reached and the score is set to 0.	24
Figure 12	QED and DRD2 scoring components averaged per batch with a rolling window of 20 steps across 20 independent runs.	27
Figure 13	Violin and box plots showing the distribution of the DRD2 scores across 20 independent runs for each method for different step ranges. Higher step ranges usually have a higher average score and less variance, which indicates a convergence of the agent to a local maximum.	28
Figure 14	Average cumulative unique hits and diverse hits smoothed over 10 training steps across 20 independent runs for DRD2. The tinted area shows the standard deviation across runs. Higher is better as it indicates more exploration.	28
Figure 15	Diversity metrics for 20 independent runs using DRD2. The bars show the 25th and 75th percentiles, while the error bars show the min/max value. The line in the middle shows the mean and the triangle the median. . .	29
Figure 16	Average number of cumulative unique scaffolds and unique topological scaffolds smoothed over 10 training steps across 20 independent runs for DRD2. The tinted area shows the standard deviation across runs. Higher is better.	29
Figure 17	Diversity metrics for 20 independent runs using GSK3 β . The bars show the 25th and 75th percentiles, while the error bars show the min/max value. The black line shows the mean and the triangle the median.	30
Figure 18	Average cumulative unique hits and diverse hits across 20 independent runs smoothed over 10 training steps for GSK3 β . The tinted area shows the standard deviation across runs. Higher is better as it indicates more exploration.	31
Figure 19	QED and GSK3 β scoring components averaged per batch with a rolling window of 20 steps across 20 independent runs.	31
Figure 20	Violin and box plots showing the distribution of the GSK3 β scores across 20 independent runs for each method for different step ranges.	31

List of Figures

Figure 21	Average number of cumulative unique scaffolds and unique topological scaffolds across 20 independent runs for GSK3 β . The tinted area shows the standard deviation across runs. Higher is better.	32
Figure 22	Top 128 GSK3 β scores over the initial 300 steps and the area under the curve averaged per batch with a rolling window of 20 steps across 20 independent runs.	32
Figure 23	Violin and box plots showing the distribution of the Jnk3 scores across 20 independent runs for each method for different step ranges.	33
Figure 24	QED and Jnk3 scoring components averaged per batch with a rolling window of 20 steps across 20 independent runs.	33
Figure 25	Diversity metrics for 20 independent runs using Jnk3. The bars show the 25th and 75th percentiles, while the error bars show the min/max value. The middle line shows the mean and the triangle the median.	34
Figure 26	2D PCA and t-SNE projections of the fingerprints of hits for Jnk3, coloured by method. The baseline clusters tightly in a small region of the chemical space, while the diversity methods spread their hits more broadly. Both intrinsic reward methods form separated clusters.	34
Figure 27	Density of hits in 2D t-SNE projections for Jnk3 per method. The scaffold methods cluster in specific regions, while BitBIRCH spreads more broadly across the chemical space.	35
Figure 28	Average cumulative unique hits and diverse hits across 20 independent runs smoothed over 10 training steps for Jnk3. The tinted area shows the standard deviation across runs. Higher is better as it indicates more exploration.	35
Figure 29	Average number of cumulative unique scaffolds and unique topological scaffolds across 20 independent runs for Jnk3.	36
Figure 30	Average GSK3 β score, top 128 GSK3 β scores and the corresponding area under the curve averaged per batch with a rolling window of 20 steps across 20 independent runs.	36
Figure 31	Average cumulative unique hits and diverse hits across 20 independent runs smoothed over 10 training steps for GSK3 β with and without experience replay.	37
Figure 32	Learning metrics for 20 independent runs using GSK3 β with sample reweighting and experience replay. The bars show the 25th and 75th percentiles, while the error bars show the min/max value. The black line shows the mean and the triangle the median.	38
Figure 33	Average GSK3 β score, top 128 GSK3 β scores and the corresponding area under the curve averaged per batch with a rolling window of 20 steps across 20 independent runs.	39
Figure 34	Learning metrics for 20 independent runs using DRD2 and alterntive clustering approaches. The bars show the 25th and 75th percentiles, while the error bars show the min/max value. The black line shows the mean and the triangle the median.	III

List of Figures

- Figure 35 Average GSK3 β score, top 128 GSK3 β scores and the corresponding area under the curve averaged per batch with a rolling window of 5 steps across 20 independent runs. IV
- Figure 36 Learning metrics for 20 independent runs using GSK3 β and various forms of experience replay. The bars show the 25th and 75th percentiles, while the error bars show the min/max value. The black line shows the mean and the triangle the median. VI

List of Tables

Table 1	Overview of SMILES tokens, their syntactic purpose, and representative usage.	5
Table 2	Thresholds for hit classification for the used scoring components QED, DRD2, GSK3B, JNK3	12
Table 3	Bit-Feature attributes for each entry j in the BitBIRCH tree including the number of molecules N_j , the linear sum of the fingerprints ls_j , the centroid c_j , and the indices of the molecules in the cluster $mols_j$	14
Table 4	Convergence statistics for the top 128 GSK3 β scores across 20 runs (mean $\pm$ standard deviation). The best mean and lowest standard deviation in each column are bold.	37
Table 5	Convergence statistics for the top 128 GSK3 β scores across 20 runs (mean $\pm$ standard deviation). The best mean and lowest standard deviation in each column are bold.	V

1

Introduction

Drug discovery is the process of finding and designing new potential medicines that can safely and effectively prevent, treat, or cure diseases. It focuses on identifying promising candidate molecules that later move into full drug development and clinical testing. Despite its importance, bringing a novel drug from concept to market is a notoriously time- and capital-intensive often exceeding \$2 billion and requiring 10–15 years to complete [7]. A significant bottleneck lies in the early stages, where identifying novel chemical structures with specific, desirable biological and chemical properties and the right balance of efficacy and safety is similar to finding a needle in a haystack.

To address these inefficiencies, the pharmaceutical industry is increasingly turning to computational methods. The process of de-novo drug design is a computational approach that aims to generate novel molecular structures from scratch, rather than identifying them through the screening of finite databases of existing compounds and modifying molecules to observe if a desired effect is achieved [8]. Given that the theoretical chemical space of drug-like molecules is estimated to exceed 10^{60} compounds [7], exhaustive enumeration is impossible. De-novo design seeks to intelligently navigate this vast combinatorial space to identify novel chemical entities with specific, desirable biological and physicochemical properties.

Recent advancements in deep learning, particularly the application of Reinforcement Learning (RL) in frameworks like REINVENT [9], have shown promise in automating the entire generation pipeline. REINVENT treats the molecule generation as a sequential task, where an agent sequentially selects tokens to construct a molecule in a string formulation. The objective is to fine-tune a general pre-trained prior model to maximize the expected return given by a predefined scoring function that rates the desirability of the completed sequence to not only generate syntactically feasible, but also novel molecules than inhibit some predefined properties. This learning is carried out in a closed-loop manner, where the agent generates molecules, evaluates them using the scoring function, and uses the feedback to update its policy to generate better molecules in subsequent iterations, as illustrated in Figure 1.

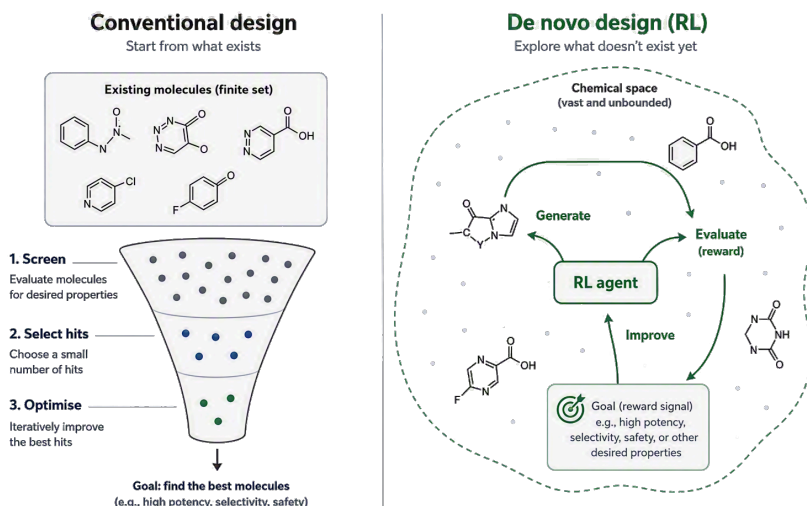


Figure 1 : High level architectural difference between traditional drug discovery starting from a finite library and consequent screening and de novo design using goal-oriented Reinforcement Learning to explore a vast space.

However, current RL-based generative models often suffer from slow convergence to target property regions and a lack of structural diversity, frequently getting stuck in local maxima and leading to the generation of chemically similar, non-novel compounds, which is widely known as “mode collapse”. [3] This project proposes to use a novel robust and scalable clustering-based mechanism using BitBIRCH [10] clustering for enhanced exploration combined with experience replay mechanisms for incorporating past high-scoring solutions to stabilize learning and facilitate exploitation.

1.1 Aim

The thesis builds upon existing methods in RL-based molecular design with the REINVENT framework serving as the foundational platform. Previous studies by Svensson et al. have already systematically explored diversity-enhancing methods [11] and memory concepts, such as memory-based diversity penalties and experience replay [5]. However these concepts have been mostly investigated in isolation missing a unified memory approach that balances exploitation and exploration. Furthermore, previous work often did not place a high focus on computational efficiency and runtime performance during the generation process, which is a big factor as this can limit the number of generated molecules in a set time frame.

The central aim of this thesis is the integration of a diversity penalty method in combination with a dynamic experience replay memory based on BitBIRCH clustering [10] to enforce structural diversity. The goal is a comprehensive scientific benchmark of different diversity methods, including the new approach using BitBIRCH clustering and methods previously investigated, both in terms of performance metrics (such as quality, convergence rates, number of hits) but also measuring the diversity.

1.2 Risks & Ethical considerations

While using machine learning for drug discovery does not pose a direct risk or ethical concerns in the same way as working with clinical data directly, significant ethical considerations arise upstream that can influence downstream outcomes if unaddressed. These risks are not evident in the scope of this thesis since we are working with hypothetical cases, but are important to consider in the broader context of drug discovery and development if excessive trust in the technology would bypass necessary safeguards.

During optimization, models may hit property targets but suggest molecules that are not realistically synthesizable or optimize for rewards that underrepresent safety, inadvertently prioritizing candidates with hidden toxicity and increasing downstream animal or human exposure. But this is mainly influenced by the target scoring function being used and their shortcomings. We treat these as black boxes and only focus on the RL learning, which is not a big risk but could in theory amplify and strengthen dataset biases and exacerbating health inequality.

However, within the scope of this work, the generative agent functions strictly as a hypothetical computational generation tool with toy targets, having no direct consequences in the drug discovery process. In downstream applied drug discovery these risks are significantly mitigated by a human-in-the-loop governance structure. Although the model accelerates the design process, the final prioritization of compounds is performed by human scientists, ensuring rigorous synthetic feasibility and safety filters are applied. Additionally, comprehensive toxicological assessments are conducted separately and remain outside the theory and computational scope of this project.

2

Background

2.1 Molecular Representations

For computational tasks, molecules need to be represented in a way that captures their structure and properties while being efficient for storage and computations. In order to represent molecular structures computationally, a few standards have emerged.

2.1.1 Graph Representations

The most intuitive representation is an annotated graph. Each atom is a labeled node linked by edges representing the bonds. This closely resembles the way molecules are usually presented in a 2D visualization in many text books and learning material. In practice, there exist libraries and tools such as RDKit, which store the molecular graph as connected atom and bond objects and allow for high-level computations [12]. Each atom object carries properties such as atomic number and formal charge and each bond object stores bond order/type (single, double, aromatic, etc.), ring membership, and stereo information.

This allows to define primitive operations for predicting, calculating and storing associated properties, modify the molecule or join molecules.

While easy and intuitive to work with, this representation is quite memory intensive and algorithms often have to traverse the whole graph to calculate molecular-wide properties. That’s why people have also strode to represent molecules in a simpler string based format that can also easily be stored and retrieved in databases at massive scales.

2.1.2 SMILES

The Simplified Molecular Input Line Entry System (SMILES) introduced by Weininger in 1987 takes such a string based approach [13]. Each molecule is represented as plain text or a sequence of tokens. The molecular graph is traversed, each atom is assigned one token (for example C for carbon, N for nitrogen, see Table 1), hydrogen bonds and simple bonds are implicit. With parentheses and a numbering system branches and cyclic structures can be encoded. This allows for a highly efficient storing and processing molecules and enables us to use generative

2. Background

language models to generate molecules. However, a single molecule can be represented in many different ways using the SMILES syntax. When using canonical SMILES, it is guaranteed that the graph is always traversed in the same way giving a unique representation for every molecule. Figure 2 shows a few examples.

Table 1 : Overview of SMILES tokens, their syntactic purpose, and representative usage.

Token	Purpose	Use
B, C, N, O, P, S, F, I	Organic subset atoms (implicit H)	C = carbon, N = nitrogen
Cl, Br	Two-letter element symbols	Cl = chlorine, Br = bromine
[...]	Bracket atom (explicit H, charge, isotope, chirality)	[NH3+], [13C], [Fe+2]
c, n, o, s	Aromatic atoms (lowercase)	c1ccccc1 = benzene
-	Single bond (usually implicit)	Explicit when needed for clarity
=	Double bond	C=O = formaldehyde
#	Triple bond	C#N = hydrogen cyanide
:	Aromatic bond (usually implicit)	Rarely used explicitly
(	Start a branch	C(=O)O = carboxyl group
)	End a branch	Returns to the main chain
1-9, %10-%99	Ring-closure digits	C1CCCCC1 = cyclohexane
<G0>	Start-of-sequence token (generation)	Signals the model to begin generating
<E0S>	End-of-sequence token (generation)	Signals termination of generation

2.1.3 Fingerprints

Fingerprints are a lossy vector representation of a fixed length representing global properties of a molecule. Like embeddings for media and text they give a numerical representation of the underlying data and thus are highly beneficial for computational tasks and efficient comparison of molecules. However, since fingerprints only store certain properties and not the entire structural information, one can usually not reconstruct the original molecule from a fingerprint. Thus the same fingerprint could be assigned to different molecules, they are not a unique representation, but allow us to embed properties of molecules in a numeric space and usage in computational tasks. In this thesis we will use the ECFP4/Morgan fingerprint of size 2048 and radius 2, which is a circular fingerprint that captures the local environment of each atom in a molecule up to a certain radius and encodes this information into a binary vector [14]. In benchmarks this has shown to be a good default choice for various molecular property prediction tasks [15].

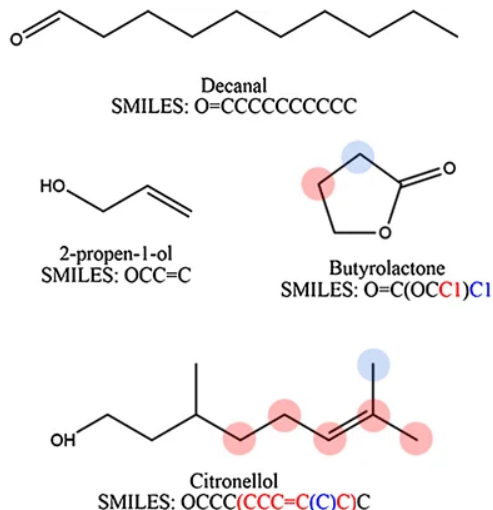


Figure 2 : SMILES representations of several simple and more complex molecular structures illustrating how circles are represented using a numbering systems and side chains with parentheses (adapted from Blay et al. [1])

2.2 Molecular Generation using a Language Model

For the task of generating a molecule, the text-based SMILES representation has shown particularly attractive as it allows us to use autoregressive language models to generate molecules token by token. Multiple studies have shown that this approach is as effective or more effective as graph-based approaches while being more computationally efficient and easier to implement [2], [9].

Since SMILES can be represented as a sequence of tokens, where each token corresponds to an atom, bond, or special character in the SMILES syntax, we can divide the generation in multiple successive steps. An autoregressive language model is trained to predict the next token in the sequence given the previously generated tokens. This allows to sample from a complex probability distribution over the vocabulary of tokens in steps given the previous context $s_{<t}$, enabling the generation of novel molecules that are not present in the training data step by step. Crucially, the joint probability of a sequence of tokens can be factorized into a product of conditional probabilities, allowing us to model the generation process as an isolated decision-making problem for each token:

$$P_{\theta}(S) = P_{\theta}(s^1)P_{\theta}(s^2 | s^1)\dots P_{\theta}(s^T | s^{T-1}, \dots, s^1)$$

The generation is started with a special start token $\langle G0 \rangle$ and terminated when a stop token is reached i.e $s_T = \langle E0S \rangle$. The parameters θ can be learned from data using maximum likelihood estimation. More details about the training will be discussed in Section 2.2.2.

2.2.1 Recurrent Neural Networks

A common architecture for auto-regressive language models is a recurrent neural network (RNN) which is designed to process sequences of data by maintaining a hidden state h_t that captures the context of a sequence over time in a compressed form. RNNs can capture dependencies in variable length data, making them suitable for modeling complex sequences like SMILES strings.

The neural network has two inputs, the embedding of the previous token x_{t-1} and the hidden state h_{t-1} from the previous time step encoding the previous context. It produces an output $y_t = f(x_{t-1}, h_{t-1})$ which is a probability distribution over the next token in the vocabulary and an updated hidden state $h_t = g(x_{t-1}, h_{t-1})$ that is passed to the next time step.

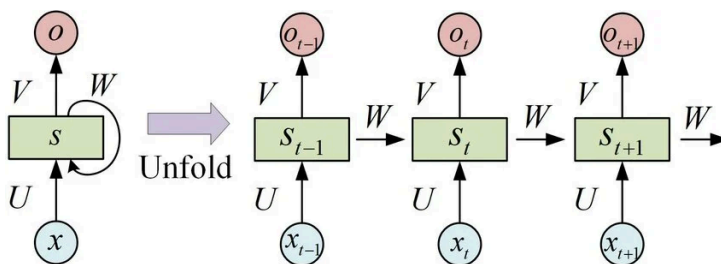


Figure 3 : Schematic representation of a Recurrent Neural Network (RNN). The key functionality and distinction to neural networks is in the hidden state that gets passed as an additional input in the network in order to generate predictions for the next token. The transformation W for the hidden state are learned and the state of the previous token is passed to the next token.

REINVENT uses a Long-Short-Term-Memory (LSTM) architecture, designed to mitigate the vanishing gradient problem and capture long-term dependencies more effectively [8], [9]. LSTMs use gating mechanisms to control the flow of information and maintain a memory cell that can store information over long periods of time. These architecture decisions are outside the scope of this work and depend on the pretraining setup. We will treat the RNN as a black box that can be trained to predict the entire sequence given the previous context and not focus on the internal workings more.

2.2.2 Pretraining a Prior model

Olivecrona et al. [2] first used the ChEMBL [16] database to pretrain a prior model on the general rules and structures of valid molecules. For each token in a training sequence, the embedding of the previous token x_{t-1} or start token $\langle G0 \rangle$ if $t = 1$ is provided as input to the model, which then produces a hidden state h_t and a probability distribution over the next token in the vocabulary $P_\theta(s^t | s^{t-1}, \dots, s^1)$.

The objective is to maximize the likelihood of the training data, which corresponds to minimizing the negative log-likelihood of the observed sequences given the models parameters θ :

$$L(\theta) = - \sum_{t=1}^T \log P_{\theta}(s^t \mid s^{t-1}, \dots, s^1)$$

The parameters θ are iteratively optimized using stochastic gradient descent allowing the model to learn the underlying structure and syntax of valid SMILES strings, which is crucial for generating chemically valid molecules. [2]

2.2.3 Post-training using Reinforcement Learning

While the prior model can generate valid molecules after pretraining, it does not necessarily generate molecules with desired properties. To address this, reinforcement learning (RL) is used to explore a chemical space that is favorable in terms of bioactivity of specific targets and synthesizability.

The problem can be formulated as a Markov decision process (MDP) where the state is given by the generated sequence until step t : $s_{<t} = a_{0:t-1}$. In this state an agent picks an action from the SMILES vocabulary $a_t \in V$ and transitions to a new state $s_{<t+1}$ by sampling the action/token according to parameterized policy $\pi_{\theta}(a_t \mid s_{<t})$, which corresponds to the “next-token” probability distribution that the RNN predicts. Once the terminal state is visited (by sampling a stop action $\langle \text{EOS} \rangle$) a scoring function scores the generated molecule and provides a reward $r(s_T) \in [0, 1]$ which measures the desired chemical properties [2].

In practice this scoring function usually consists out of a geometric mean of different criteria like drug likeliness, activity against a target, toxicity, etc. The task to develop reliable models for these predictions is a complex topic on its own and left to other work to explore. In the context of this thesis, the scoring is treated as a black box oracle providing no prior knowledge. This oracle call is usually an expensive operation, so we would like to avoid it if possible. So a core measure of computational cost will be the number of oracle calls and learning steps the agent has to make to sufficiently explore and exploit in the chemical space. We will come back to this topic in Section 2.4.

The aim of RL is to adjust the parameters θ of the agent’s policy $\pi_{\theta}(a_t \mid s_t)$ to maximize the expected return $E_{A \sim \pi_{\theta}}[r(s_T)]$ over possible sequences $A \sim \pi_{\theta}$.

This quantity is hard to optimize directly, so Olivecrona et al. [2] proposed to optimize according to the “augmented likelihood”, which combines the reward with the fixed frozen prior likelihood $E_{A \sim \pi_{\theta}}[r(s_T)]$ of the generated sequences $A \sim \pi_{\theta}$ for regularization. This ensures that the generated molecules remain valid and similar to the training data while optimizing for the desired properties and incorporating the rewards. One can mathematically show that this is equivalent to maximizing the expected return[2].

$$\log \pi_{\theta}^{\text{aug}}(A) = \sum_{t=1}^{T-2} \log \pi_{\theta_{\text{prior}}}(a_t \mid s_t) + \sigma \cdot r(s_T)$$

2. Background

The entire RL flow is detailed in Figure 4. The parameters are adjusted in batches B with a policy gradient approach according to the following loss function which corresponds to maximizing the average reward.

$$L^{\text{RL}}(\theta) = \frac{1}{|B|} \sum_{A \in B} \left(\log \pi_{\theta}^{\text{aug}}(A) - \sum_{t=1}^{T-2} \log \pi_{\theta}(a_t | s_t) \right)^2$$

The parameters are iteratively optimized using stochastic gradient descent, allowing the model to learn to generate molecules that not only are valid but also have desirable properties as defined by the scoring function.

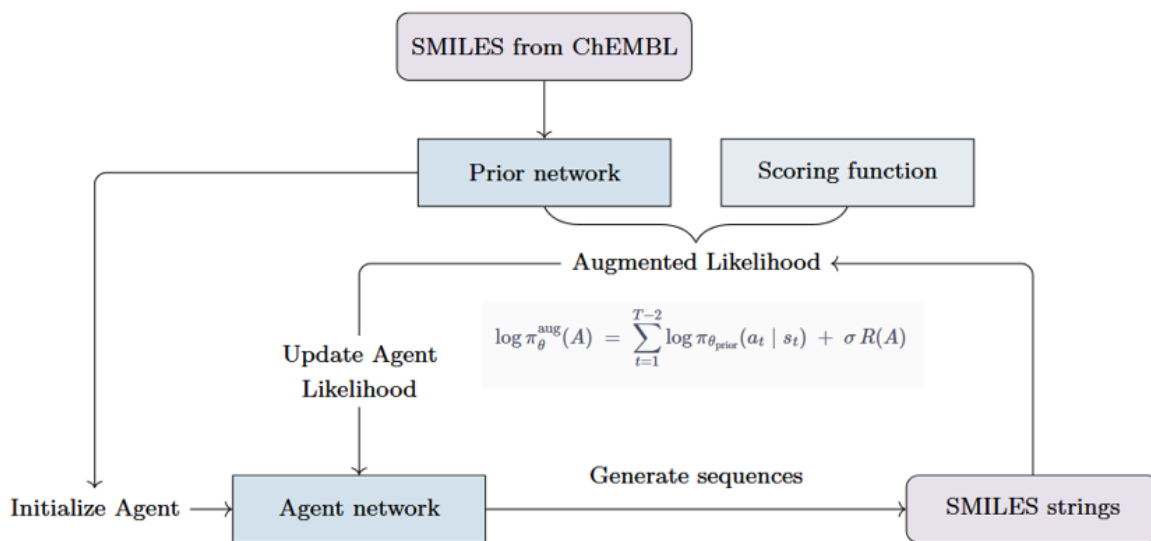


Figure 4 : Schematic representation of the RL flow. The prior network is trained on a large SMILES database like ChEMBL, the network is frozen and copied to an Agent network. This network generates new SMILES sequences in batches. We score the molecules using the scoring function and combine the scores for the entire batch with the likelihood predicted by the prior network for regularization, which Olivecrona et al. named “Augmented Likelihood” [2]. Then we update the agent by using mean square error between the agent likelihood and the augmented likelihood and stochastic gradient descent.

2.2.4 Mode Collapse and Lack of Diversity

In RL there is a fundamental trade-off between exploration and exploitation. In the context of drug design, the model needs to explore the chemical space to discover new molecules with desirable properties, but it also needs to exploit the knowledge it has gained to find high-rewarding molecules. If the model focuses too much on exploitation, it may get stuck in local reward maxima and fails to explore other regions of the chemical space, leading to little space coverage and a lack of diversity in the generated molecules.

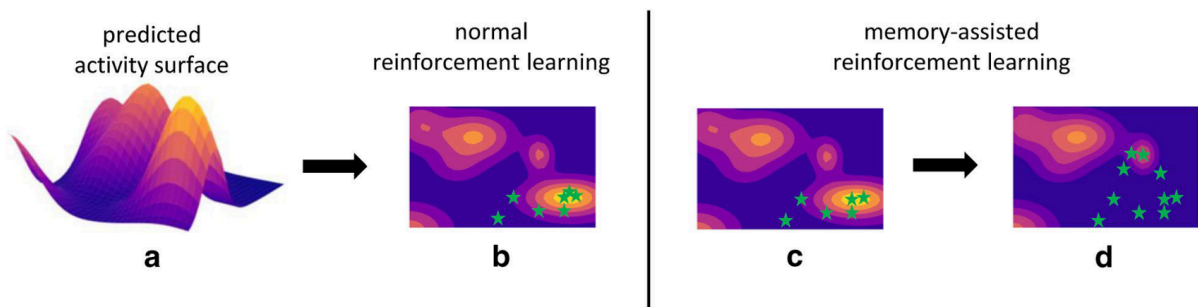


Figure 5 : Differences between regular and memory assisted RL as proposed by Blaschke et al. [3]. Without a memory, the RL may get stuck on local maxima, doing too much exploitation, leading to mode collapse. With a memory we can adapt the loss landscape dynamically and reduce mode collapse once we find too many molecules in a similar high scoring area. RL can then explore and find other high scoring areas.

While this RL approach often generates novel compounds with desired properties, the generated molecules are often very similar to each other. This is a common problem in generative models known as mode collapse, where the model generates a limited variety of outputs, often converging to a few high-rewarding solutions and failing to explore the broader chemical space [11], [17]. This lack of diversity can be problematic in drug design, as it may only lead to the generation of compounds that are structurally similar and may have similar biological activities to known molecules, limiting the potential for discovering novel drug candidates with a better property profile that are not yet patented.

Blaschke et al. [3] and Svensson et al. [11] propose several methods to enhance diversity in the generation process, such as penalizing the extrinsic reward with a weight $h(A) \in [0, 1]^{|A|}$ when encountering too many structurally similar molecules or providing an additional intrinsic reward $R_I(A)$ encouraging diversity based on information entropy and random network distillation (see Section 3.2.2 and Section 3.2.1):

$$\hat{R}(A) = h(A) \cdot R(A) + R_I(A)$$

Essentially these modifications have the effect of another scoring function and modify the reward landscape dynamically during the learning process to enforce spread when stuck in local maxima as shown in Figure 5.

To be able to effectively design these diversity-enhancing strategies, it is crucial to reliably measure and efficiently compute the diversity of a set of generated molecules, which is a non-trivial task given the complexity and size of the chemical space.

2.3 Measuring Diversity

During a generation task we can easily work with thousands or millions of complex molecules and what counts as “diverse” changes based on the application and goal. So in literature there is no one definition of what “diversity” means in the context of

2. Background

molecules as there are many ways to describe molecules and measure their differences. In general when talking about a diverse set of molecules, one wants to cover as much of the chemical space as possible and find molecules that are not “too similar” to the previously seen molecules. Many ways have been proposed to quantify this similarity. While all of them have their advantages and disadvantages, a widely used measure for binary molecular fingerprints is the Jaccard Index or Tanimoto Similarity [18].

2.3.1 Tanimoto Similarity

Tanimoto Similarity is based on binary fingerprints as introduced in Section 2.1.3. In simplified terms we count the number of equal *on* bits of two fingerprint vectors relative to the number of mismatching bits. It can be calculated as follows, where a is the number of coincidences of two *on* bits between fingerprints and $b + c$ the number of mismatches, where one bit is *on* and the corresponding other one is *off* respectively.

$$T = \frac{a}{a + b + c}$$

In a sense to reach a high similarity, one wants to maximize the number of *on-on* coincidences, while minimizing the number of *on-off* mismatches. In order to reach the maximum similarity of 1, all bits of the fingerprints have to be equal. On the contrary, we get a similarity of 0 if all bits are pairwise different. This makes sense as fingerprints should encode the same properties at every bit and two molecules with overlapping properties are likely very similar.

Since this yields a value between 0 and 1, one can also derive a distance metric:

$$D = 1 - T$$

In practice it has been shown that the Tanimoto similarity is a reliable and efficient measure for comparing molecular fingerprints, especially with ECFP4 fingerprints [19]. While Tanimoto similarity is a good measure for a pair of molecules or small sets, it quickly becomes intractable to evaluate all possible pairs in big libraries. One needs more efficient approximations.

2.3.2 iSIM: average similarity

The iSIM metric provides efficient approximations of the average similarity of a set of molecules. Instead of comparing all molecules with each other which needs $\frac{n(n+1)}{2} \in O(n \cdot n)$ computations, iSim allows calculating average similarities without pairwise comparisons in $O(n)$ [20]

It relies on the fingerprints being a fixed-length binary vector and counts the number of *on* bits k_q in each position q across all fingerprints.

The crucial observation is that these aggregated counts are enough to calculate the average Tanimoto similarity of the set, since the number of *on-on* coincidences in position q can be calculated as $a_q = \binom{k_q}{2} = \frac{k_q(k_q-1)}{2}$ and the number of *on-off*

2. Background

mismatches as $b_q + c_q = k_q(n - k_q)$. Thus the average similarity can be approximated in linear time by summing over all positions q :

$$iT = \frac{\sum_q \frac{k_q(k_q-1)}{2}}{\sum_q \left[\frac{k_q(k_q-1)}{2} + k_q(N - k_q) \right]}$$

iSim does also define other similarity metrics, but we will only use the Tanimoto similarity in this work.

2.3.3 Unique Hits / Richness

A very simple measure of diversity is the number of unique molecules that an agent generates. A higher number of unique molecules indicates a more diverse set, as it suggests that the molecules are not simply variations of a few core structures but rather represent a wider variety of chemical entities. However, it is important to note that richness alone may not fully capture the diversity of a set, as it does not account for the structural similarity between molecules.

Table 2 : Thresholds for hit classification for the used scoring components QED, DRD2, GSK3B, JNK3

Metric	Threshold
QED	>0.6
DRD2	>0.6
GSK3B	>0.5
JNK3	>0.4

In this work we will only count high scoring molecules as so-called "hits". Since certain scoring functions cover different ranges and uncertainties, we adjusted the thresholds individually to each oracle as seen in Table 2. This prediction is purely based on external QED and TDC oracles as detailed later in Section 3.6, so it might contain false predictions depending on the implementation of the oracle, which is out of scope of this work. If we use `QED+tdc_gsk3b` for example, a hit must have a QED score of at least 0.6 and a `tdc_gsk3b` score of 0.5. Richness is then a simple count of the number of unique hits.

2.3.4 #Circles

The #Circles metric is a more complex approximation of the average similarity of a set of molecules. It is based on the idea of counting the number of circles that can be drawn around the molecules in the fingerprint space, such that all points are included within at least one circle. The more circles that can be drawn without overlapping centers, the more diverse the set of molecules is [4]. Formally, the #Circles metric can be defined as follows:

$$\#Circles(S; d, t) = \max_{C \subseteq S} |C| : \forall x, y \in C, d(x, y) \geq t$$

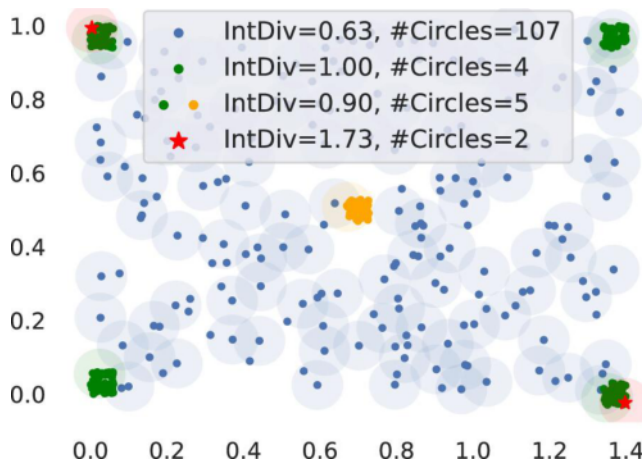


Figure 6 : Example of the $\#Circles$ metric in 2D. The circles are aligned in a maximal way around the molecules in the fingerprint space, such that all points are included within at least one circle, but no two circles have overlapping centers. This does capture the diversity of the set, as a higher number of circles (shown in blue) indicates that the molecules are more spread out in the space and less similar (opposed to the green circles) to each other [4].

This metric provides a way to quantify diversity while avoiding the computational complexity of evaluating all pairwise similarities, making it more practical for large datasets. Every unique hit that classifies as center of such a circle is called “Diverse Hit” [21]. In this work, we will also take the “diversity ratio” into account, defined as:

$$\text{diversity ratio} = \frac{\# \text{ diverse hits}}{\# \text{ unique hits}}$$

A high ratio might indicate that even though few unique hits are found, many of them are diverse and maybe more valuable than many hits in a tight cluster.

2.3.5 Unique Scaffolds

Bemis et al. [22] introduced the concept of scaffolds as a way to capture the core structure of a molecule, which can be used as a diversity measure. A scaffold is a substructure of a molecule that remains after removing all side chains and substituents as illustrated in Figure 7. The number of unique scaffolds of a set of molecules counts the number of distinct core structures present in the set. Two structurally similar molecules may share the same scaffold, so a higher number of unique scaffolds indicates a more diverse set of molecules, since it suggests that the molecules are based on a wider variety of core structures. This metric is particularly useful in drug discovery, because it can help identify novel structural chemical entities that may have common desirable properties.

Topological scaffolds go a step further and remove the bond types and replace all atoms with carbon, thus only keeping the connectivity of the atoms as an unlabeled graph. Thus the number of topological scaffolds will always be less or equal than unique scaffolds.

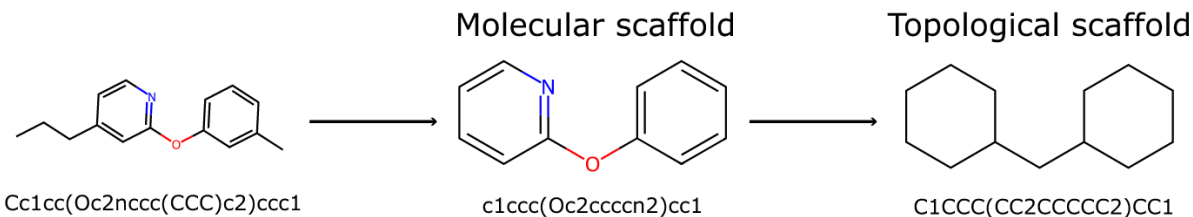


Figure 7 : The scaffold and topological scaffold of an example molecule. The scaffold is the core ring structure with side chains removed and the topological scaffold is an unlabeled version without atom or bond descriptors (Adapted from Svensson et al [5])

2.3.6 BitBIRCH Clustering

BitBIRCH is a clustering algorithm that is particularly well-suited for large datasets of molecules. It is built upon the core concepts from the BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) algorithm [23], which is designed to efficiently cluster large datasets by building a hierarchical tree structure using continuous values. But instead of continuous values in a Euclidean space, BitBIRCH operates on binary fingerprint vectors and Tanimoto Similarity, thus being applicable to our fingerprint space [10].

Table 3 : Bit-Feature attributes for each entry j in the BitBIRCH tree including the number of molecules N_j , the linear sum of the fingerprints ls_j , the centroid c_j , and the indices of the molecules in the cluster $mols_j$.

BF_j	$(N_j, ls_j, c_j, mols_j)$
N_j	Number of molecules
ls_j	$[\sum_{k=1}^{N_j} x_1^{j,k}, \dots, \sum_{k=1}^{N_j} x_q^{j,k}]$
c_j	Centroid $\frac{ls_j}{N_j} + \frac{1}{2}$
$mols_j$	[indices of molecules]

Under the hood it uses the iSIM metric to build a hierarchical clustering based on the average similarity in the form of a balanced tree structure where each node stores a fixed set of binary features (BF). Every BF maintains only summary information of the cluster, including the number of molecules N_j , the linear sum of the fingerprints ls_j , the centroid c_j , and the indices of the molecules in the cluster $mols_j$, as detailed in Table 3 [10]. The leaf BF represent clusters of molecules, and the internal nodes represent hierarchical clusters of their children clusters.

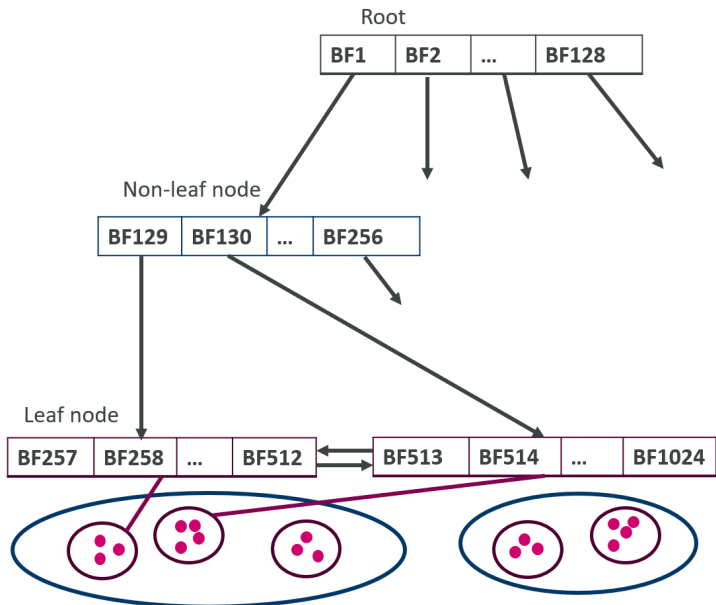


Figure 8 : Schematic representation of the BitBIRCH tree structure. Each node includes a fixed set of binary features (BF) that summarize the included molecules. The leaf BF represent clusters of molecules shown in purple.

When inserting a new molecule in the tree, one just has to update the count, sum, centroid and indices in the BFs from the root to the leaf according to:

$$N_j^{\text{new}} = N_j + 1 \quad \mathbf{ls}_j^{\text{new}} = \mathbf{ls}_j + \mathbf{x} \quad \mathbf{c}_j^{\text{new}} = \frac{\mathbf{ls}_j^{\text{new}}}{N_j^{\text{new}}} + \frac{1}{2} \quad \text{mols}_j^{\text{new}} = \text{mols}_j + [\text{index}]$$

For every node j in the tree, the centroid c_j is used to determine the closest child node to insert the new molecule into. When reaching a leaf, the average similarity of the cluster is calculated using the iSIM metric as follows, since the linear sum ls of binary fingerprints calculates the number of *on* bits in each position q :

$$iT = \frac{\sum_q \frac{ls_q(ls_q - 1)}{2}}{\sum_q \left[\frac{ls_q(ls_q - 1)}{2} + ls_q(N - ls_q) \right]}$$

BitBIRCH accepts the new molecule into the cluster if the new diameter $D = 1 - iT$ does not exceed a user-defined threshold t . If no existing cluster can accept the new molecule, a new cluster is created in the leaf node. Once a leaf node exceeds a user-defined maximum number of clusters, the leaf is split into two new leaves and the tree is rebalanced.

This allows for memory efficient clustering of extremely large chemical libraries while maintaining computational efficiency, making it an attractive choice for assessing structural diversity in drug discovery applications. Compared to conventional clustering algorithms like Taylor-Butina which scale with $O(N^2)$, BitBIRCH significantly reduces the number of comparisons with a $O(N)$ scaling [10], [18]. With the optimized BitBIRCH Lean implementation [24], it is possible to cluster millions of molecules in a matter of minutes on a single CPU, providing more than a 1000x

speedup for 1500000 molecules compared to Taylor-Butina. This allows us to cluster the generated molecules in real-time during the generation process and assess the diversity of the generated molecules based on current cluster occupancy. The more clusters are created, the more diverse the set of molecules should be.

2.4 Experience Replay

Another core problem in reinforcement learning is the sample efficiency of the learning process. Usually every sample is only used once to update the agent and is then discarded. But a sample might contain valuable information that can be reused to improve the learning process past its initial signal. Usually the expensive part is the oracle call, so one strives to be sample-efficient and reuse sampled molecules in the learning loop. While the agent may generate a sample multiple times, it is often a good idea to additionally store the best scoring molecules and artificially reuse them in the learning loop to save resources and accelerate learning. Reusing old samples is a common technique in reinforcement learning and is known as experience replay [25].

Within REINVENT, it is implemented as a replay buffer that stores the best scoring molecules and reinserts them into the learning loop [26]. As shown in Figure 9, this mechanism stores highly scoring compounds in a sorted buffer and with each batch samples from it to reinsert these samples along with the current batch in the reinforcement learning loop. This does not only stabilize the learning and greatly improves the sample efficiency, but can also significantly improve convergence speed to favorable chemical spaces [6].

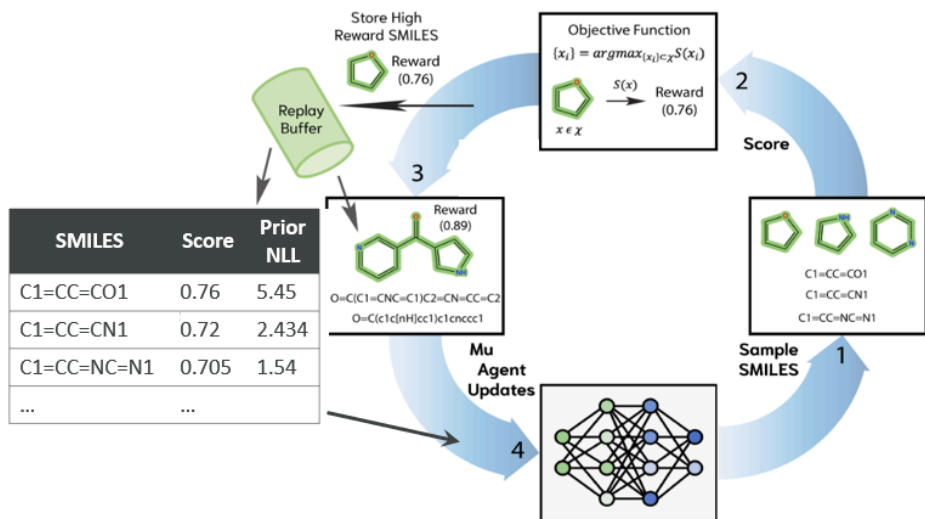


Figure 9 : Adapted learning loop with added Replay Buffer. The agent keeps a fixed memory of molecules and their negative log likelihood (NLL) sorted by score and periodically samples a few to use alongside the current batch to update the agent (adapted from Guo et al. [6]).

2. Background

However it also introduces the flaw that if too many structurally similar molecules fill the replay buffer, the agent is strongly discouraged from additional exploration [26]. So the challenge is to keep a selection of high scoring, but at the same time diverse compounds.

Furthermore, a more fundamental algorithmic challenge persists regarding the distributional consistency of this data. Because we inject and learn from old experiences generated under a historical parameter set θ_{old} , these kind of RL techniques are referred to as “off-policy”, because it involves data different from our target policy. Over time there might be drift between both policies and we might keep samples in memory that are unlikely under the new policy, which can bias updates toward the historical sampling distribution.

Accordingly, a core objective of this work is to implement alignment and diversity mechanisms. To estimate the expected return of the current policy using data from an old policy, importance sampling can correct for the difference between the two policy distributions [27, Chapter 5.5], [28]. Prioritized experience replay can instead bias sampling toward selected experiences, typically using importance-sampling corrections to account for the resulting sampling bias [29].

To the best of our knowledge, these techniques have not been employed in the context of generative drug design and is systematically explored in this work. Section 2.4 documents the implementation and evaluation of these techniques.

2.5 Importance Sampling

Importance sampling is a general statistical technique to estimate the expected value of a function under one distribution p while only having access to samples drawn from a different distribution q [27, Chapter 5.5], [28]. The key idea is to reweight each sample by the ratio of the two densities, the so-called importance weight, so that the estimate remains unbiased:

$$E_{x \sim p}[f(x)] = E_{x \sim q}\left[\frac{p(x)}{q(x)}f(x)\right]$$

In the context of reinforcement learning, this allows us to estimate the expected return of a target policy π_θ using trajectories that were generated under a different behavior policy $\pi_{\theta_{old}}$. This situation arises naturally in off-policy methods such as experience replay. The samples stored in the replay buffer were generated by an earlier version of the agent $\pi_{\theta_{old}}$, whereas we want to update the current agent π_θ . Without correction, learning from these outdated samples introduces a bias, since their distribution no longer matches the one induced by the current policy.

By weighting each action a_t with the importance ratio between the current and the old policy, we can obtain an unbiased estimate of the current policy’s objective:

$$w(a_t) = \frac{\pi_\theta(a_t \mid s_{<t})}{\pi_{\theta_{old}}(a_t \mid s_{<t})}$$

2. Background

Intuitively, sequences that are more likely under the current policy than under the old one receive a larger weight, while sequences that the current policy would rarely generate are down-weighted. This keeps the replayed data mathematically consistent with the agent’s present distribution and enables robust, unbiased updates even when reusing historical samples.

3

Methods

To address the problems of mode collapse and off-policy alignment, this thesis builds on top of existing work. Mainly implementations of the REINFORCE algorithm as part of REINVENT [9], [26] including prior RNN based models by Olivecrona et. al [2] and the BitBIRCH Lean implementation [10].

Prior work has mostly focused on using an approach based on scaffolds. [3]. This relatively simple structural method has shown both reasonably efficient and effective at generating a diverse set of molecules. More advanced methods based on the fingerprint representation have only been studied to some extent mostly in combination with scaffolds [11], [17]. This work introduces new novel diversity techniques based on clustering in the fingerprint space and expands existing scaffold-based diversity penalties into a comprehensive modular system.

3.1 Diversity Penalties

Several techniques have been developed to enforce diversity in reinforcement learning for molecular generation. One promising approach is to introduce a diversity penalty for the rewards of newly generated molecules, which discourages the agent from generating similar molecules repeatedly. Formally this penalty is defined as a function $w(A) \in [0, 1]$ which modifies the rewards/scores $R(A)$ of an active molecule A exceeding a set threshold h as follows:

$$\hat{R}_{\text{Pen}}(A) = \begin{cases} w(A) \cdot R(A) & \text{if } R(A) \geq h \\ R(A) & \text{otherwise} \end{cases}$$

Effectively, certain scores can be made artificially low so they take less importance in the update if they are seen as too similar to previously seen molecules, encouraging it to explore new areas of the chemical space. Thus the implementation usually involves keeping a memory M of all unique molecules seen so far and evaluate diversity metrics on-the-fly.

3.1.1 The Role of Clustering

Defining this dynamic penalty function $w(A)$ is the core of this work. One has to be aware both of the entire history of seen molecules to promote diversity and at the same time consider computational efficiency, so it can be reasonably integrated into each batch update.

The memory M itself can be implemented efficiently as a Hash Set, given that the molecules are represented as canonized SMILES. So checking if a molecule appeared in the past is an efficient $O(1)$ lookup using the string representation as hash key. If a molecule appears a second time, the score is set to 0 and more complex diversity calculations are skipped.

The ideal weight function would then optimize for low pairwise Tanimoto similarity and a high number of unique scaffolds, which are the main diversity indicators used in practice as introduced in Section 2.3. But when considering a growing memory of size $|M|$, one quickly realizes that this approach is not computationally feasible for large memories. For each active batch A , we need to make $O(|A| \cdot |M|)$ comparisons to measure pairwise similarity. The memory also grows by $|A|$ molecules on each step, so the total computation over steps n grows with $O(n^2)$.

A more promising approach is to keep a separate memory storing summary information about certain clusters of molecules. The specific assignment of a new molecule to a cluster can be extracted from its properties or graph structure and efficiently compared. Clustering is often used to efficiently find nearby points and groups within high dimensional data. At the same time the underlying data structure can often be updated incrementally with new entries on the fly without reindexing. This makes it highly viable for enhancing diversity within the REINVENT framework for de-novo drug design.

Most common clustering algorithms like K-Means are reliant on the euclidean space. But molecules can mathematically just be represented as annotated graphs or be reduced to a string notation like SMILES, as discussed in Section 2.1, so traditional euclidean clustering algorithms are not really feasible. However, using Bemis-Murcko scaffolds, one can formulate simple structural clustering, which we will discuss in Section 3.1.2.

There are a small number of specialized clustering approaches like Taylor-Butina for binary fingerprints and Tanimoto Similarity, but with similarly unfavorable scaling of $O(|M|^2)$ [18]. The BitBIRCH algorithm from Section 2.3.6 was specifically introduced to fill this gap and cluster binary fingerprint vectors and very large molecule databases with an advantageous scaling of $O(|M|)$ [10]. We will investigate how to integrate this novel BitBIRCH approach as a diversity penalty in the RL setting in Section 3.1.3.

3.1.2 Scaffold-Based Penalties

By grouping molecules based on their Bemis-Murcko scaffolds as introduced in Section 2.3.5, one can identify clusters of structurally similar molecules very easily [22]. A penalty factor can be applied to the reward of molecules that belong to already well-represented scaffolds, encouraging the agent to explore new scaffolds and thus increasing the diversity of the generated molecules.

One keeps track of the scaffolds of previously generated molecules and how often each scaffold has been generated in a dictionary. When a new high scoring molecule with a score above a certain threshold h is generated, we calculate and check its scaffold against the memory and apply a penalty if it belongs to a scaffold that has been seen frequently. This lookup is usually very computationally efficient with an amortized time of $O(1)$.

This idea was first introduced by Blaschke et al. to enforce diversity and closely resembles traditional drug design workflows where different kind of core structures are investigated. In practice, it showed to be successful at improving the number of unique predicted hits and number of scaffolds drastically [3].

3.1.3 BitBIRCH Penalty

Since the BitBIRCH algorithm introduced in Section 2.3.6 can iteratively cluster molecules based on their fingerprints and Tanimoto similarity, it provides a more flexible and potentially more effective way to enforce diversity compared to scaffold-based methods. The proposed novel method acts as a drop-in replacement for the scaffold-based penalty, where instead of checking the scaffold of a generated molecule, we check which BitBIRCH cluster it belongs to and apply a penalty based on the occupancy of that cluster. Like the scaffold-based penalty, once a cluster reaches a certain occupancy, it will decrease the reward for molecules assigned to that cluster, see Section 3.3. The number of clusters can increase dynamically as new molecules are generated, allowing for a memory-efficient adaptive approach to enforcing diversity. Since the algorithm clusters molecules based on the average Tanimoto similarity fingerprints, it can encourage structural diversity beyond exact scaffold identity [14]. Since we only keep summary information in memory and not the full fingerprints, this approach should also be very memory efficient and has been proven to scale with the number of molecules in computation time of $O(N)$ [10].

3.2 Intrinsic Rewards

Additionally, Svensson et.al [11] have proposed a method to encourage diversity by adding an intrinsic reward based on the novelty of generated molecules. This approach can be implemented by calculating a novelty score for generated molecules that surpass a scoring threshold of h . The intrinsic reward is then added to the extrinsic reward from the scoring function, encouraging the agent to explore new areas of the chemical space while still optimizing for desired properties.

$$\hat{R}_I(A) = \begin{cases} R(A) + R_I(A; H_i) & \text{if } R(A) \geq h \\ R(A) & \text{otherwise} \end{cases}$$

3.2.1 Random Network Distillation (RND)

Random Network Distillation (RND) is a method for generating intrinsic rewards based on the novelty of states in reinforcement learning. It works by training a network for few epochs to predict the output of a fixed, randomly initialized network [30]. In the context of molecular generation, RND can be applied by treating each generated molecule as a state and using the prediction error as a measure of novelty. For molecules that the agent has seen frequently, the prediction network will likely be able to predict the output of the random network accurately, resulting in a low intrinsic reward. Conversely, for novel molecules that the agent has not seen before, the prediction error will be higher, providing a stronger intrinsic reward and encouraging the agent to explore those novel molecules further.

Svensson et.al [11] based both the target and prediction networks on the same architecture as the prior RNN model and initialized the target network π_f randomly, while the prediction network π_ϕ is initialized with the pre-trained weights of the prior and trained for one epoch per batch to predict the output of the target network for high scoring generated molecules. The log likelihood of the generated molecule under both the target and prediction networks are given by testing.

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

The novelty can be measured as the squared error between the log likelihood of the target $f(A)$ and prediction network $\hat{f}_\phi(A)$ for a generated molecule.

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

For a batch of generated molecules H_i , the intrinsic reward is normalized to the range $[0, 1]$ and added to the extrinsic reward from the scoring function, allowing for a balanced optimization of both novelty and desired properties.

$$R_I^{\text{RND}}(A; H_i) = \frac{\Delta_{\hat{f}}(A; \phi) - \min_{\tilde{A} \in H_i} \Delta_{\hat{f}}(\tilde{A}; \phi)}{\max_{\tilde{A} \in H_i} \Delta_{\hat{f}}(\tilde{A}; \phi) - \min_{\tilde{A} \in H_i} \Delta_{\hat{f}}(\tilde{A}; \phi)}$$

3.2.2 Information Entropy

Svensson et.al [11] also propose a method by adding an intrinsic reward based on the information entropy of the generated molecules. The idea is to calculate how unexpected a molecule is based on its assigned bucket, where less common molecules have higher information content. These novel structures are more valuable for exploration and thus given a higher reward. The intrinsic reward can be defined as the negative logarithm of the frequency of the bucket in the memory with $N[S]$ being

3. Methods

the number of active molecules with bucket S in memory, and $\mathcal{S}$ the set of unique molecular scaffolds in memory.

$$I_{\text{scaffold}}(A) = -\log\left(\frac{N[S_A]}{|\mathcal{S}|}\right)$$

This intrinsic reward encourages the agent to generate molecules that belong to less represented buckets, thus promoting diversity in the generated molecules.

For a batch of molecules H_i , the information values are normalized to the range $[0, 1]$ and added to the extrinsic reward from the scoring function, allowing for a balanced optimization of both novelty and desired properties.

$$R_I^{\text{Inf}}(A; H_i) = \frac{I_{\text{scaffold}}(A) - \min_{\tilde{A} \in H_i} I_{\text{scaffold}}(\tilde{A})}{\max_{\tilde{A} \in H_i} I_{\text{scaffold}}(\tilde{A}) - \min_{\tilde{A} \in H_i} I_{\text{scaffold}}(\tilde{A})}$$

Compared to other methods this approach generated substantially more scaffolds and topological scaffolds on both GSK3 β and JNK3 tasks in prior work [11].

3.3 Modular Architecture

This work proposes a modular architecture as in Figure 10 where different diversity penalties, penalty functions and intrinsic rewards can be combined and configured to suit specific tasks and objectives.

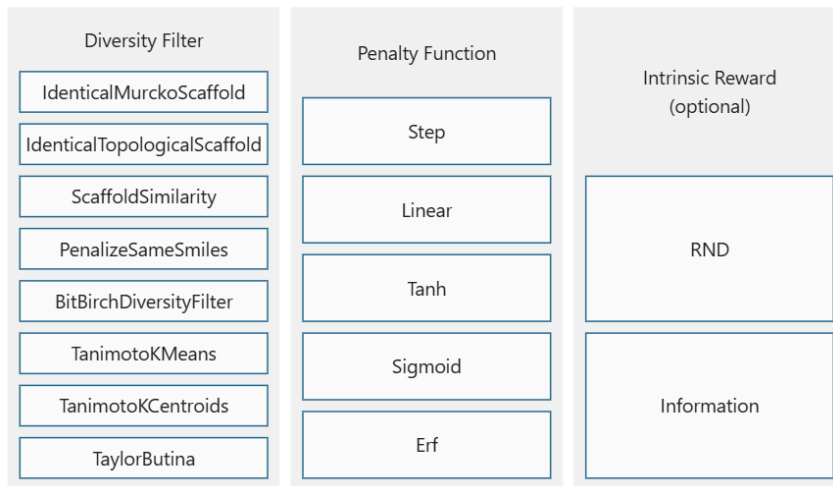


Figure 10 : Modular architecture for diversity penalties and intrinsic rewards. The diversity penalty can be based on different clustering methods and combined with a penalty function and optionally an intrinsic reward.

For every molecule in the active batch, a bucket utilization $u = \frac{n}{B}$ is calculated based on n , the number of molecules in the memory that belong to the same cluster or scaffold, and a configurable maximum occupancy B . The choice of the penalty function $p(u)$ determines how the occupancy score is translated into a penalty factor, which is then applied to the score of the molecule. A sharp penalty function like a step function will result in a hard penalty, while a smooth penalty function like a

3. Methods

sigmoid or tanh will result in a stricter soft penalty that increases gradually as the occupancy score increases. Figure 11 shows the curves of the penalty functions depending on the bucket utilization u .

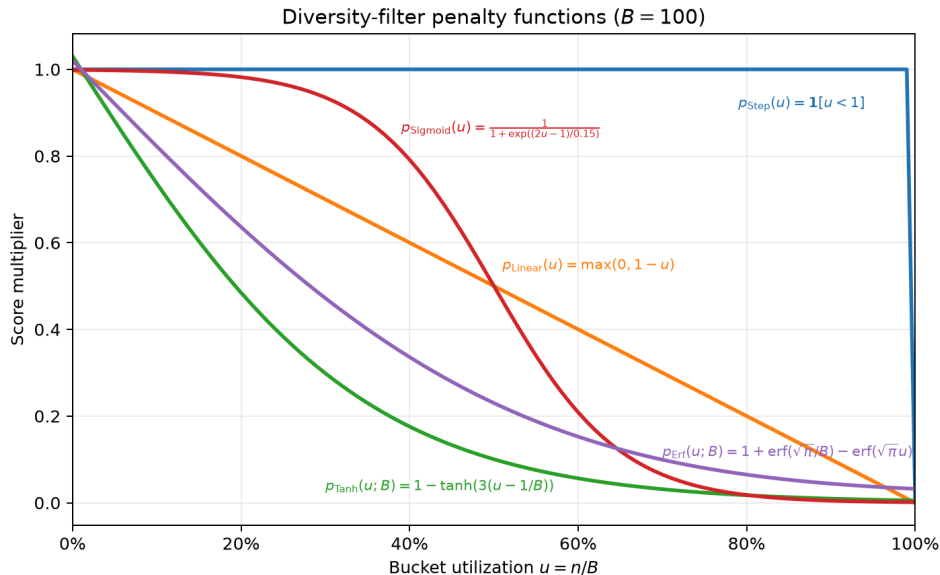


Figure 11 : Selection of penalty functions for score penalties. The step function is a hard penalty, while the sigmoid, linear, tanh and erf functions are soft penalties that increase smoothly as the occupancy of a cluster increases until a set maximum is reached and the score is set to 0.

3.4 Sample Reweighting within Experience Replay

This work extends the concept of experience replay as introduced in Section 2.4 by reweighing the samples in the replay buffer according to the current likelihood of the agent inspired by importance sampling and prioritized replay. Therefore one needs to reevaluate the likelihoods of the samples under the current policy $P_{\text{current}}(A_i)$.

Crucially because we only store the negative log-likelihoods of the generated molecules in the replay buffer, we calculate the importance weights for the entire sample A_i in the replay buffer based on the difference between the current and stored log-likelihoods instead of the probability of each token.

$$w_i = \frac{P_{\text{current}}(A_i)}{P_{\text{stored}}(A_i)} = \exp((\log P_{\text{current}}(A_i) - \log P_{\text{stored}}(A_i)))$$

In practice we compute these differences in parallel for all samples in the replay buffer and apply a softmax function to normalize the weights to the range $[0, 1]$ and ensure they sum to 1, which makes the implementation trivial and only based on the negative log-likelihoods.

$$w = \text{softmax}(\text{NLL}_{\text{stored}} - \text{NLL}_{\text{current}})$$

When sampling from the replay buffer, we use these weights to bias the sampling towards more likely samples under the current policy using weighted sampling, which

should help the agent to focus on more relevant experiences and improve learning efficiency. The downside of this method is a slight overhead in computing the current log-likelihoods for all samples in the replay buffer, for which we need to reevaluate the model’s probability for each sample. This grows with the size of the replay buffer, which is a fixed user-defined parameter.

3.5 Combining Diversity Penalties and Experience Replay

Experience Replay as explained in Section 2.4 exhibits the flaw that it somewhat works against the goal of maintaining diversity, as it tends to favor frequently encountered high-scoring samples and can lead to a concentration on a narrow subset of the chemical space. One idea is to force diversity within the replay buffer itself by incorporating diversity-enhancing techniques discussed in Section 3.1 when inserting new samples into the buffer. Concretely when inserting new entries into the replay buffer, we calculate a diversity penalty $w(A_i) \in [0, 1]$ for all samples A_i previously in the buffer. The stored score s_i is then adjusted according to a tunable parameter λ as follows, such that $\lambda = 1$ applies the full diversity penalty and $\lambda = 0$ ignores it:

$$s_i \leftarrow s_i \cdot (1 - \lambda + \lambda w(A_i))$$

Only the top-k entries according to the adjusted scores are retained in the replay buffer, keeping a fixed size. Over time the diversity penalty downscores less diverse samples and may eject them from the buffer once new higher-scoring samples are inserted, promoting a more diverse set of experiences within the buffer.

3.6 Experimental Setup

In order to assess the effectiveness of the proposed methods, they are implemented within the REINVENT framework and evaluated on a set of standard molecular generation tasks.

For the prior network we use the pre-trained RNN model by Olivecrona et. al [2], which has been trained on the ChEMBL database and is capable of generating valid SMILES strings in a vast chemical space.

To assess performance in a realistic drug discovery context, we use scoring functions, which are based on the Therapeutics Data Commons (TDC) benchmarks for molecular generation [31]. The targets include the Dopamine Type 2 Receptor (DRD2), c-Jun N-terminal Kinases-3 (JNK3) and Glycogen Synthase Kinase 3 Beta (GSK3 β) oracles provided by TDC in equal combination with an estimate of drug likeliness (QED) as used in prior work [5], [11], [32].

Molecules are generated in batches of 128 and the agent is trained for a maximum of 3000 steps, which allows for sufficient training while keeping the runtime and cost manageable.

3. Methods

The performance of the proposed methods is compared against a baseline REINVENT agent without any diversity enhancement, as well as against existing diversity techniques such as scaffold-based penalties with and without intrinsic rewards. All diversity penalties use a threshold of $h = 0.5$, meaning the final reported score of the weighted scoring function ($\frac{1}{2}$ QED + $\frac{1}{2}$ TDC) has to exceed this threshold to count active and viable for the diversity penalties. Each method is evaluated across 20 independent runs to ensure robustness and reliability of the results, with consistent trends observed across runs being considered significant.

We use the number of unique molecules, which are predicted to be active by the scoring functions (Hits), as a primary metric for quality, as it captures both the ability of the agent to generate desirable molecules and the diversity of those molecules.

But since this metric alone does not fully capture the distance between molecules, we also use the diverse hits metric, defined as the number of circles that can be drawn around the unique hits in the chemical space, where each circle represents a certain similarity threshold, as introduced in Section 2.3.4 and proposed by Renz et. al. [21]. This has shown to be a more nuanced metric for diversity compared to others, as it captures not only the number of unique hit molecules but also how spread out they are in the chemical space, providing a better understanding of the diversity of the generated molecules. This metric captures the ability of the agent to generate novel hit molecules that are not just minor variations of previously generated ones. Additionally to estimate the quality and speed of convergence of the agent, we track the evolution of the average and top-k scores over the course of the learning process. This mirrors the normal use case in drug design where typically a reduced set of top candidates are selected for further experimental validation.

We also employ the number of unique Bemis-Murcko scaffolds and topological scaffolds as a metric for diversity, as it captures the structural diversity of the generated molecules and has been employed in prior work [3], [11].

4

Results

In general, the effectiveness of the newly proposed methods seems to be highly dependent on the scoring functions used for training. So in the following we are going to investigate the results isolated for each target and summarize the findings in the discussions in Section 5.

We are comparing the *baseline* to the raw *bitbirch* with hard penalty as well as *tanh_RND_bitbirch* with soft tanh penalty and RND intrinsic reward. The scaffold based *tan_RND_scaffld* and *tanh_Information_scaffold* with soft penalty and intrinsic rewards act as state-of-the-art comparisons as proposed by Svensson et al [5].

4.1 Dopamine Type 2 Receptor (DRD2)

For DRD2, Figure 12 shows that the baseline agent can usually find a high scoring region fairly fast, but Figure 14 shows that this happens at the expense of diversity and exploration when measured by hits and diverse hits. While the agent explores the chemical space quickly, it seems to get stuck in local optima, which is reflected in the plateau in diverse hits and scaffolds after 200–400 steps.

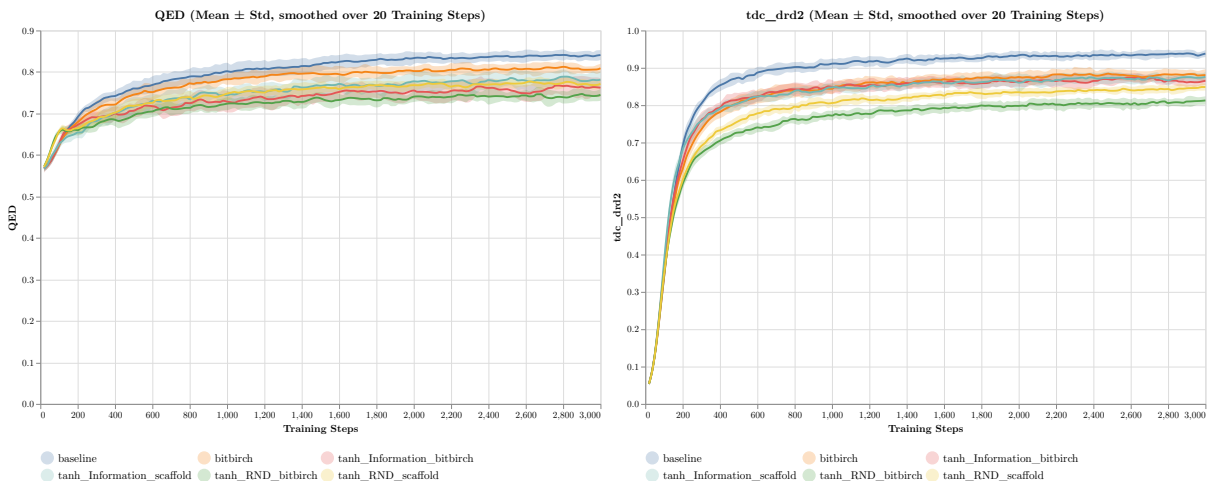


Figure 12 : QED and DRD2 scoring components averaged per batch with a rolling window of 20 steps across 20 independent runs.

4. Results

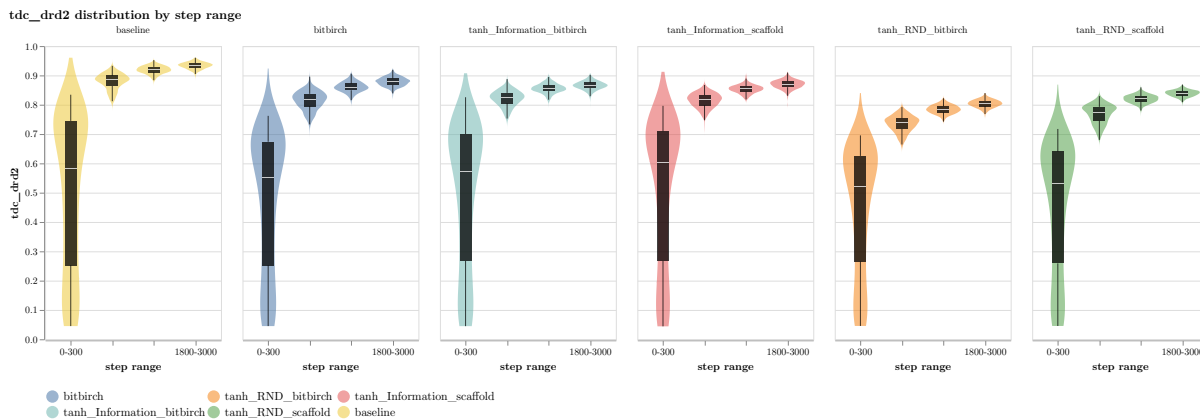


Figure 13 : Violin and box plots showing the distribution of the DRD2 scores across 20 independent runs for each method for different step ranges. Higher step ranges usually have a higher average score and less variance, which indicates a convergence of the agent to a local maximum.

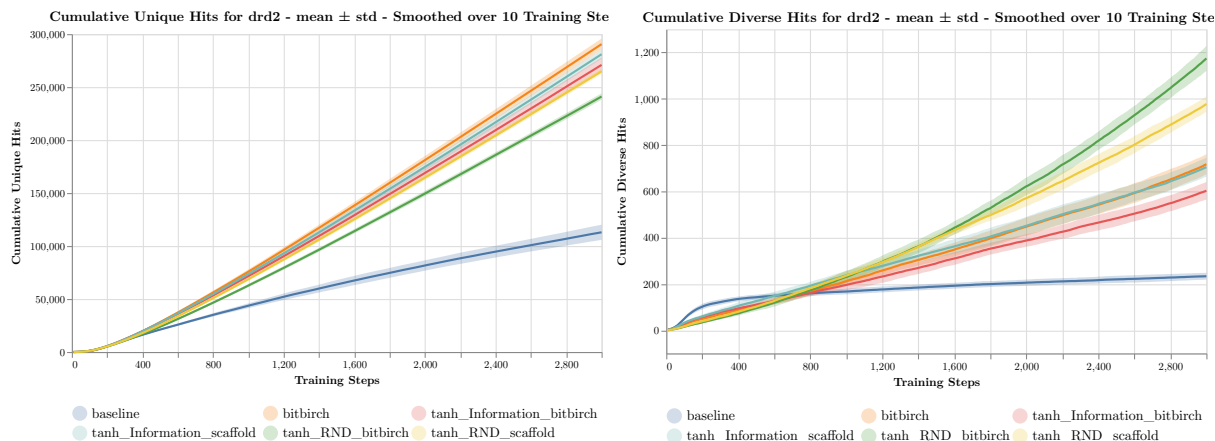


Figure 14 : Average cumulative unique hits and diverse hits smoothed over 10 training steps across 20 independent runs for DRD2. The tinted area shows the standard deviation across runs. Higher is better as it indicates more exploration.

All diversity methods on the other hand seem to lead the agent to explore the chemical space more effectively, which is reflected in the growing number of unique hits and diverse hits, as seen in Figure 14. The proposed BitBIRCH method with a soft tanh penalty and Random Network Distillation shows to be very promising when looking at diverse hits, the diversity ratio $\frac{\text{div_hits}}{\text{unique_hits}}$ and BitBIRCH clusters, exceeding all other methods with very little variance across the twenty runs in Figure 15. The fraction of duplicate SMILES is reduced substantially with all diversity penalties at the expense of more memory usage of around 2GB for all diversity methods. At the same time Figure 13 shows it consistently has the lowest average oracle score, which suggests that there might be a trade-off between optimizing for the scoring function and maintaining diversity. Also, when looking at unique scaffolds and unique topological scaffolds in Figure 16, the tanh Information Entropy and RND methods which actively optimize for scaffolds still seem to have an edge over BitBIRCH, but show less diverse hits and BitBIRCH clusters. Overall using a soft penalty like tanh

4. Results

and intrinsic rewards seem to be more effective than a hard penalty, which is in line with prior work by Svensson et al. [5].

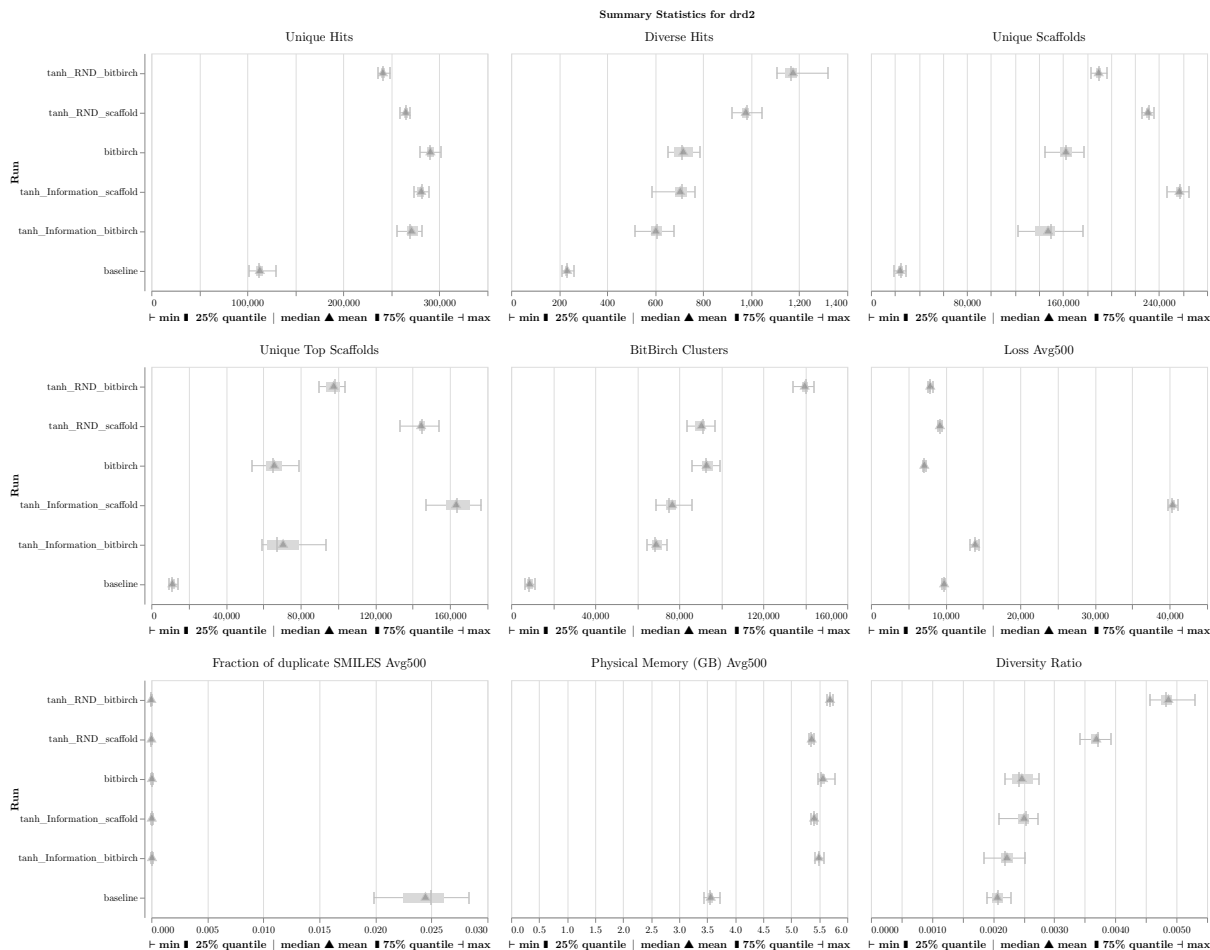


Figure 15 : Diversity metrics for 20 independent runs using DRD2. The bars show the 25th and 75th percentiles, while the error bars show the min/max value. The line in the middle shows the mean and the triangle the median.

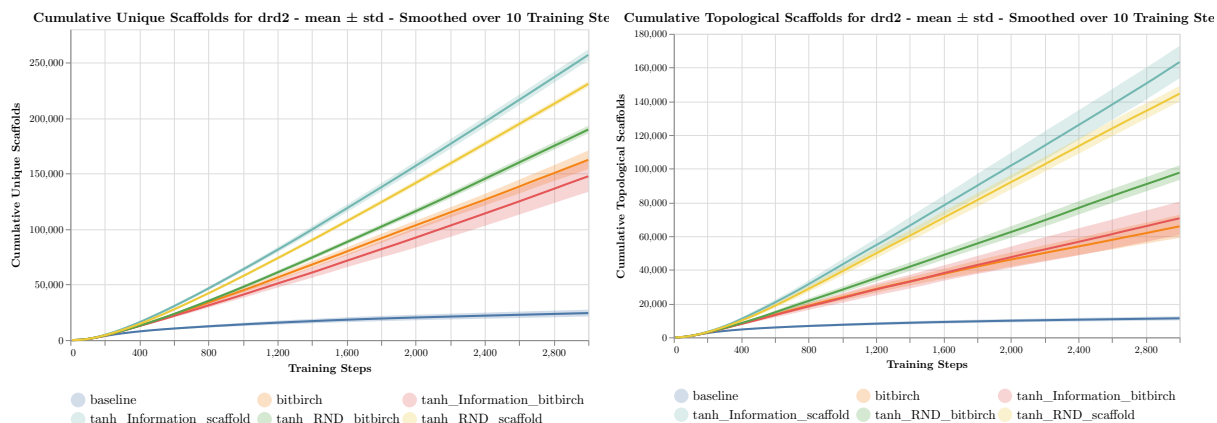


Figure 16 : Average number of cumulative unique scaffolds and unique topological scaffolds smoothed over 10 training steps across 20 independent runs for DRD2. The tinted area shows the standard deviation across runs. Higher is better.

4. Results

4.2 gsk3 β

For gsk3 β , the results are mostly in line with what we have seen with DRD2, as can be seen in the hits and diverse hits in Figure 18. Figure 19 shows that baseline agent without diversity methods again behaves as expected, quickly finding a high scoring region, but the hits and scaffolds in Figure 21 reach a plateau quickly indicating very low diversity and little exploration, not making any meaningful new discoveries after a few hundred steps.

All the diversity methods again improve on exploration and diversity metrics, but the picture is not as clear as with drd2. When it comes to diverse hits scaffold-based methods actually perform only slightly better or similar to the baseline while generating significantly more hits overall. At the same time, the BitBIRCH method using a soft tanh penalty shows by far the best results in terms of diverse hits and BitBIRCH clusters as demonstrated in Figure 17. Important to note is that adding the RND intrinsic reward seems to have the biggest effect on diverse hits. Any other methods slowly reach a plateau, which can be seen in Figure 18.

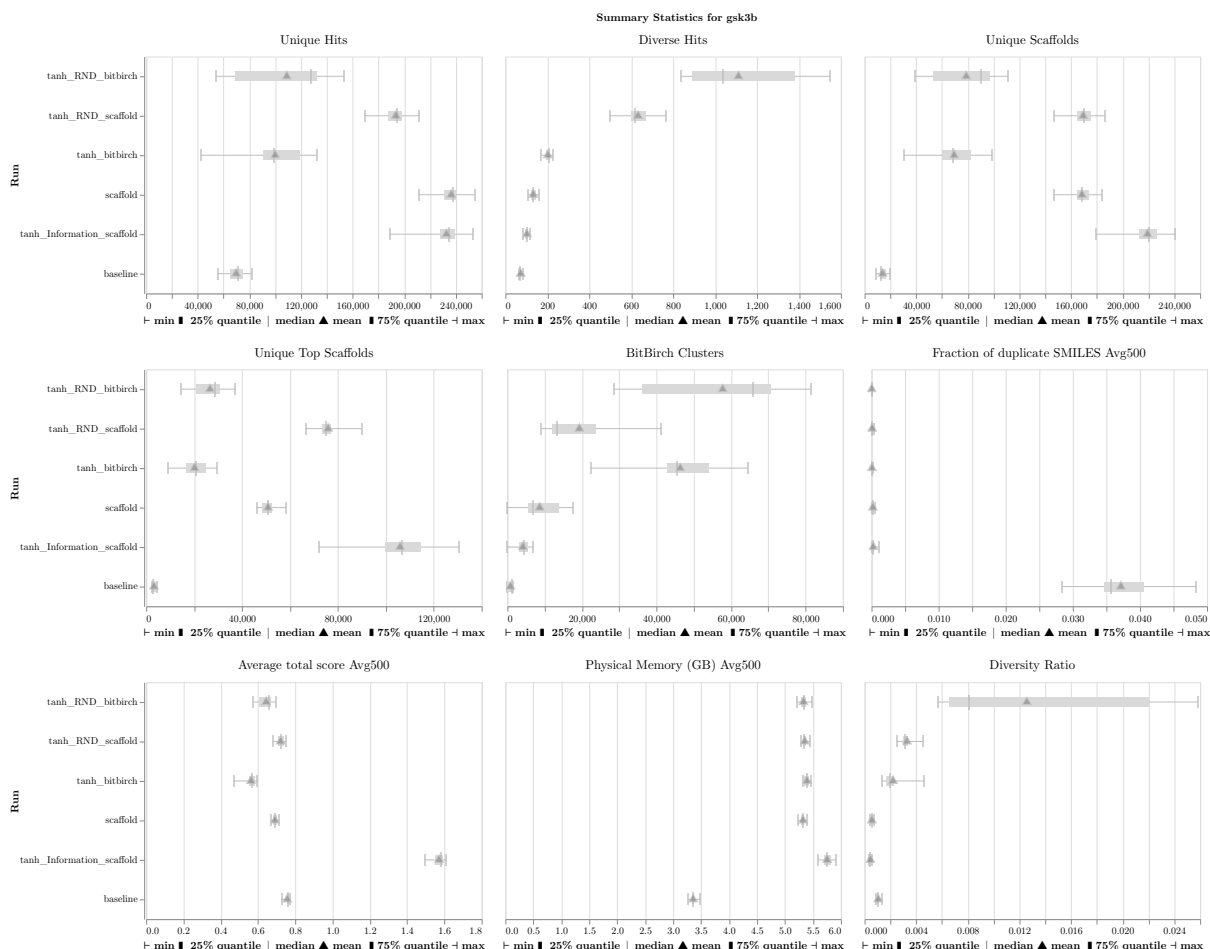


Figure 17 : Diversity metrics for 20 independent runs using GSK3 β . The bars show the 25th and 75th percentiles, while the error bars show the min/max value. The black line shows the mean and the triangle the median.

4. Results

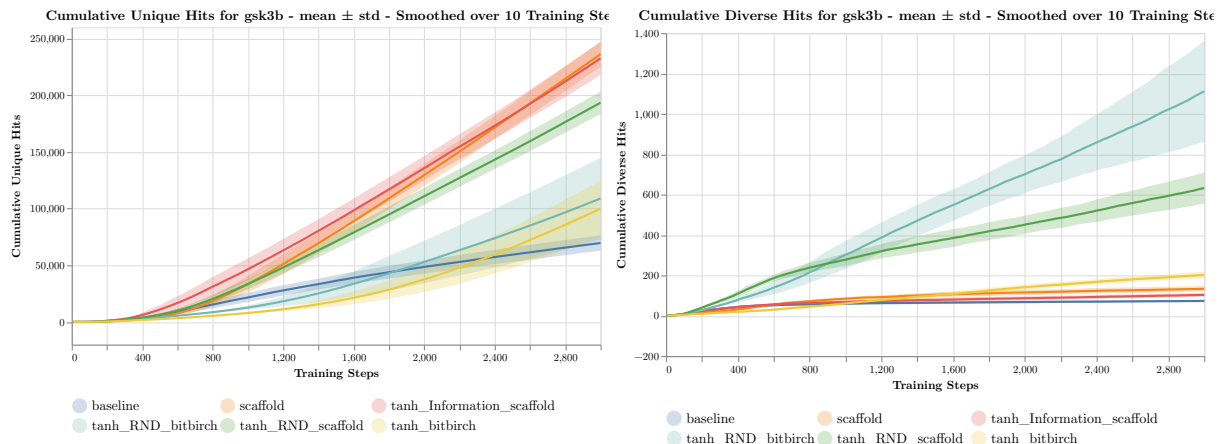


Figure 18 : Average cumulative unique hits and diverse hits across 20 independent runs smoothed over 10 training steps for GSK3 β . The tinted area shows the standard deviation across runs. Higher is better as it indicates more exploration.

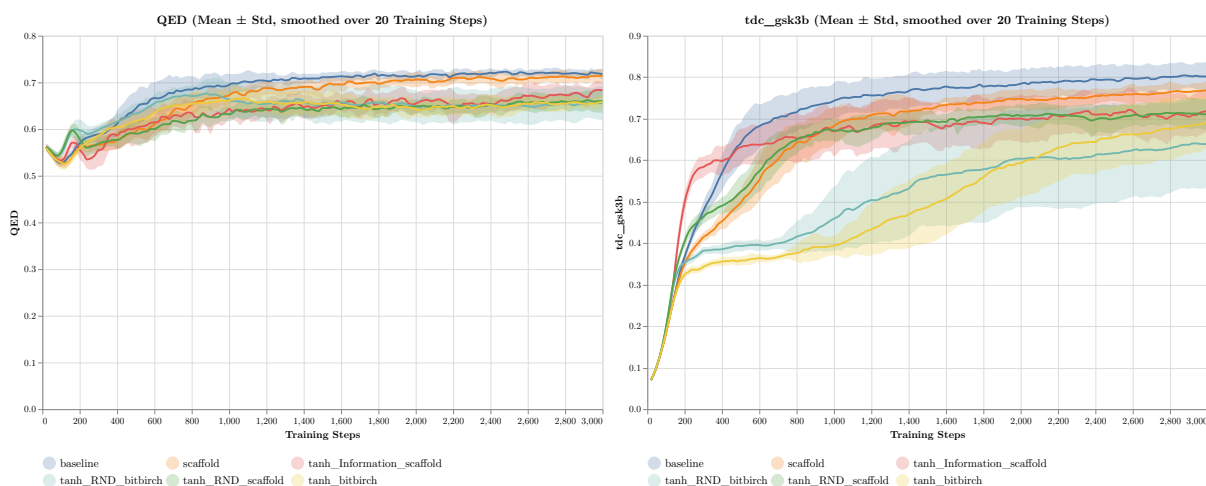


Figure 19 : QED and GSK3 β scoring components averaged per batch with a rolling window of 20 steps across 20 independent runs.

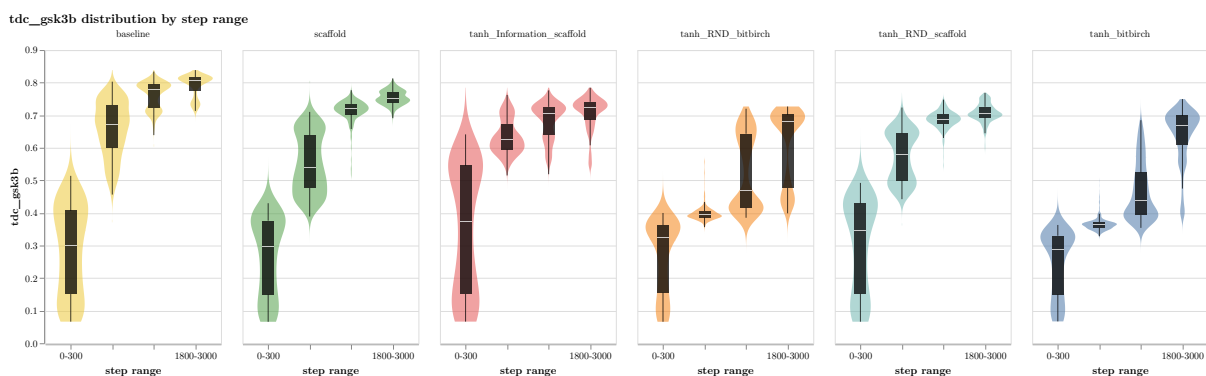


Figure 20 : Violin and box plots showing the distribution of the GSK3 β scores across 20 independent runs for each method for different step ranges.

4. Results

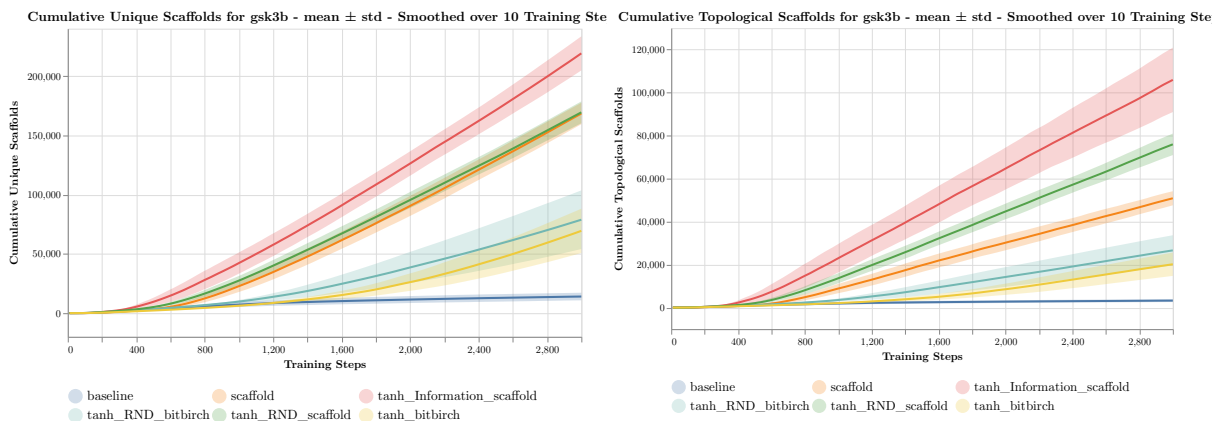


Figure 21 : Average number of cumulative unique scaffolds and unique topological scaffolds across 20 independent runs for GSK3 β . The tinted area shows the standard deviation across runs. Higher is better.

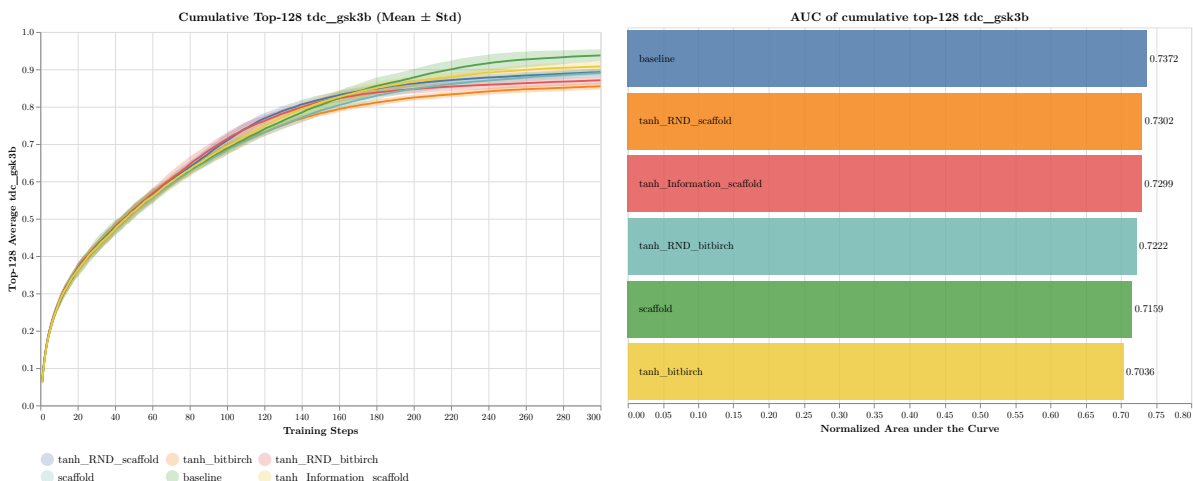


Figure 22 : Top 128 GSK3 β scores over the initial 300 steps and the area under the curve averaged per batch with a rolling window of 20 steps across 20 independent runs.

However, when looking at the oracle score trajectories in Figure 19, it seems that the BitBIRCH methods are worse at optimizing the scoring function. This trade-off is particular evident for gsk3 β , where the tanh BitBIRCH method has a much lower average oracle score than all other methods, while maintaining the high number of diverse hits and BitBIRCH clusters. Figure 20 also shows that while eventually a similar final score can be reached, it happens slower than other methods, often only after 1000-2000 learning steps. But interestingly the QED score stays relatively stable across all methods, which suggests that the methods are able to maintain a certain level of drug-likeness while optimizing for diversity and deprioritising the gsk3 β score. And when only looking at the top 128 scores in Figure 22, the tanh BitBIRCH method is still able to achieve competitive top scores and convergence, which is more relevant practically than the average score across all generated molecules as it indicates that we can still find high quality hits.

4. Results

When it comes to scaffolds in Figure 21, the tanh penalized scaffold method with Information Entropy based intrinsic reward seems to be really effective. Despite finding the least amounts of diverse hits, it manages to maintain a high number of unique scaffolds and is among the top methods regarding unique hits. This suggests that this method is able to find a high number of high scoring hits that are structurally different from each other, but maybe still fairly similar when measured in the fingerprint space.

4.3 Jnk3

For Jnk3, the results tell a somewhat different story than the other targets. As seen in Figure 25, apart from the RND and Information Entropy methods, all methods struggle to find a high number of hits and scaffolds. The number of diverse hits and BitBIRCH clusters are extremely low for all methods, which suggests that none of the methods are able to find a diverse high scoring set of molecules. However, the score distribution in Figure 23 and the oracle score trajectories in Figure 24 show that the scaffold methods with added intrinsic reward methods are able to achieve very high Jnk3 scores at times, with QED remaining relatively stable across all runs. The variance on these methods is high, so their performance is inconsistent and not be reliable across different runs.

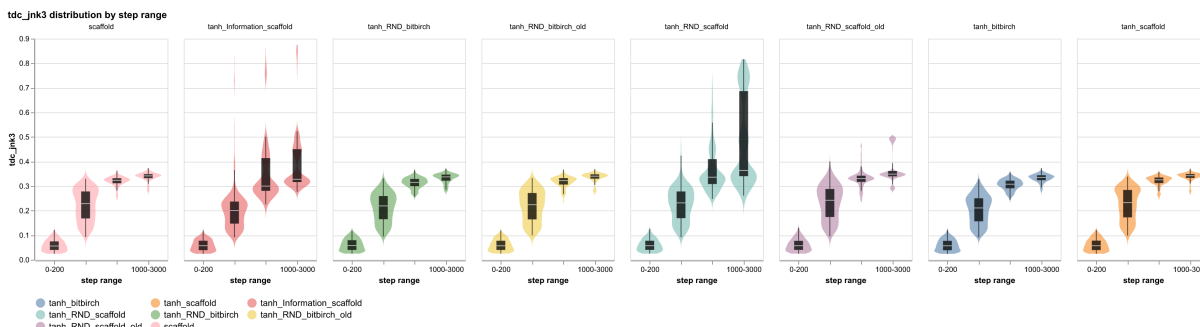


Figure 23 : Violin and box plots showing the distribution of the Jnk3 scores across 20 independent runs for each method for different step ranges.

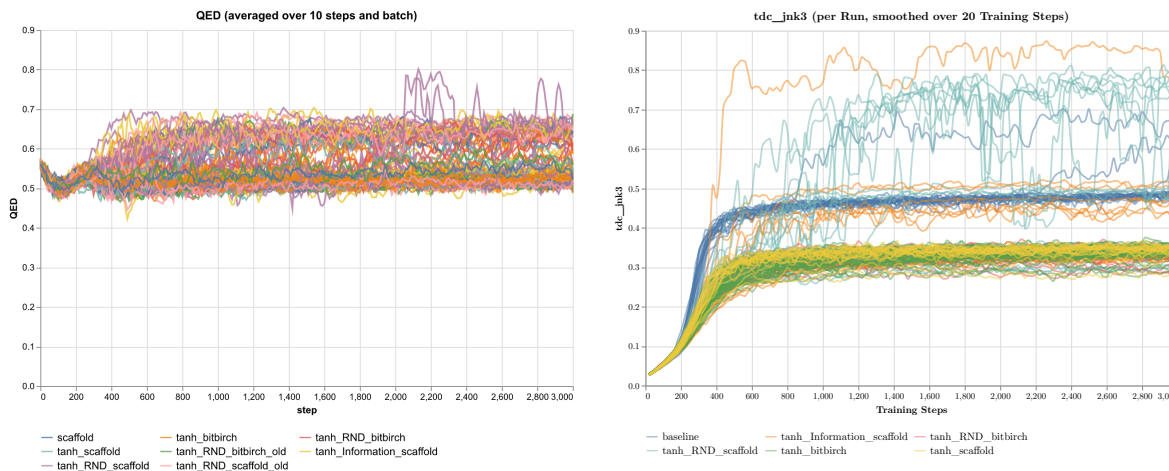


Figure 24 : QED and Jnk3 scoring components averaged per batch with a rolling window of 20 steps across 20 independent runs.

4. Results

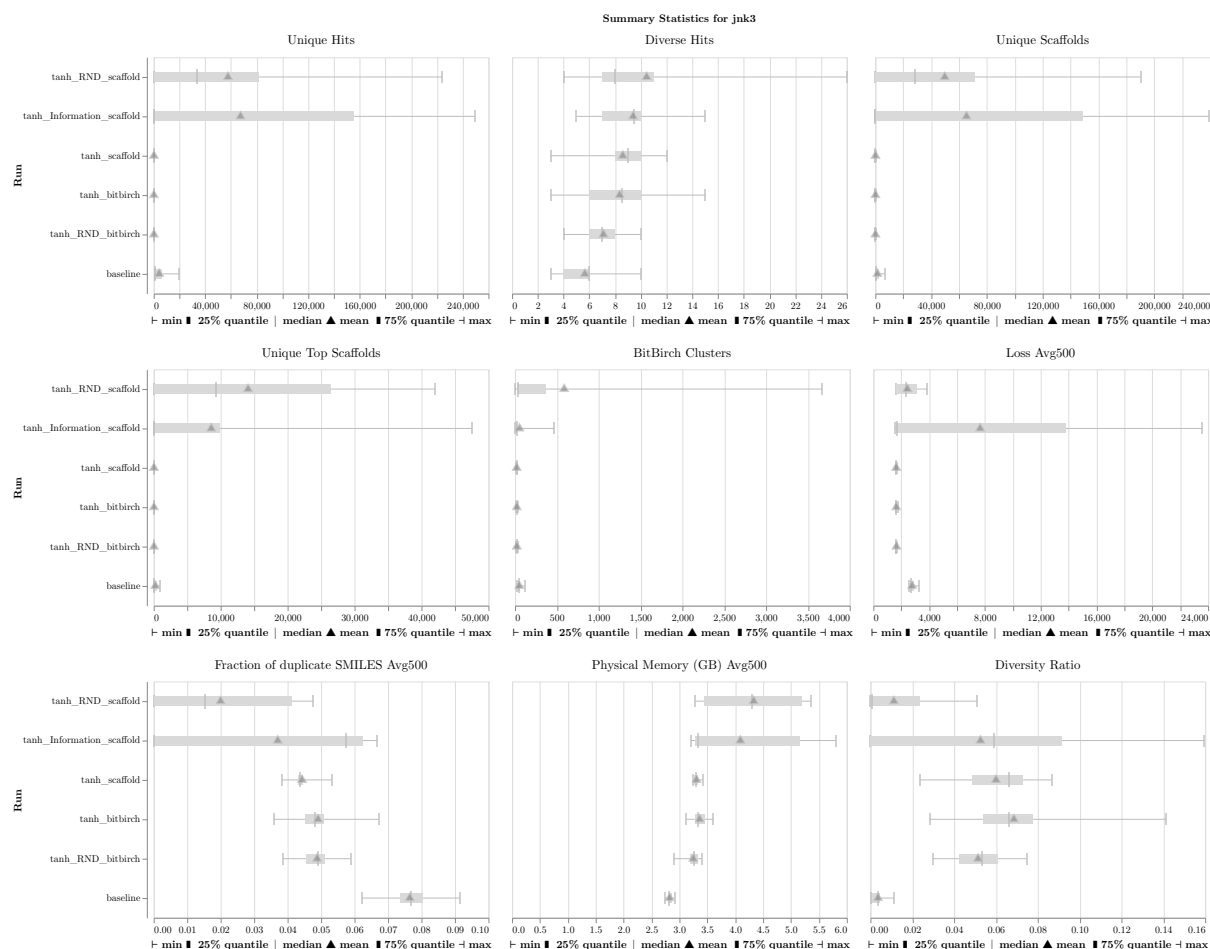


Figure 25 : Diversity metrics for 20 independent runs using Jnk3. The bars show the 25th and 75th percentiles, while the error bars show the min/max value. The middle line shows the mean and the triangle the median.



Figure 26 : 2D PCA and t-SNE projections of the fingerprints of hits for Jnk3, coloured by method. The baseline clusters tightly in a small region of the chemical space, while the diversity methods spread their hits more broadly. Both intrinsic reward methods form separated clusters.

4. Results

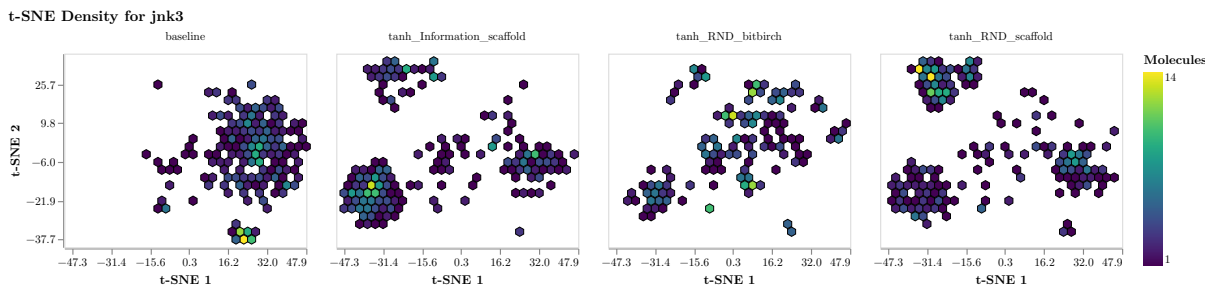


Figure 27 : Density of hits in 2D t-SNE projections for Jnk3 per method. The scaffold methods cluster in specific regions, while BitBIRCH spreads more broadly across the chemical space.

When looking at PCA and t-SNE projection of the fingerprint space of the hits in Figure 26 and Figure 27, one notices that both methods form distinct clusters separate from all other methods and the baseline cluster. This indicates that these methods due to the additional random exploration can explore the space to find high scoring regions unknown to other methods. But the fact that these clusters are very tight, suggests that they are not able to find a diverse set of hits in these regions, which is also reflected in the low number of diverse hits and BitBIRCH clusters.

So taking scaffolds as a measure alone might be misleading, as the methods might find a high number of unique scaffolds as seen in Figure 29, but if they are all very similar to each other, they might not be very useful for drug discovery and actual diverse exploration.

These results also show that the effectiveness of these diversity methods depend a lot on the oracle, how it was trained, what its domain is and how much of that is encoded in the prior. Very likely these high scoring predictions are outside the scope of the jnk3 training domain and thus very unreliable. This coincides with other work by Medina et al. with the Jnk3 oracle, where it was found to be hard to optimize for using the REINVENT prior [33]. Overall these results are very noisy, so more runs have to be performed to draw more solid conclusions.

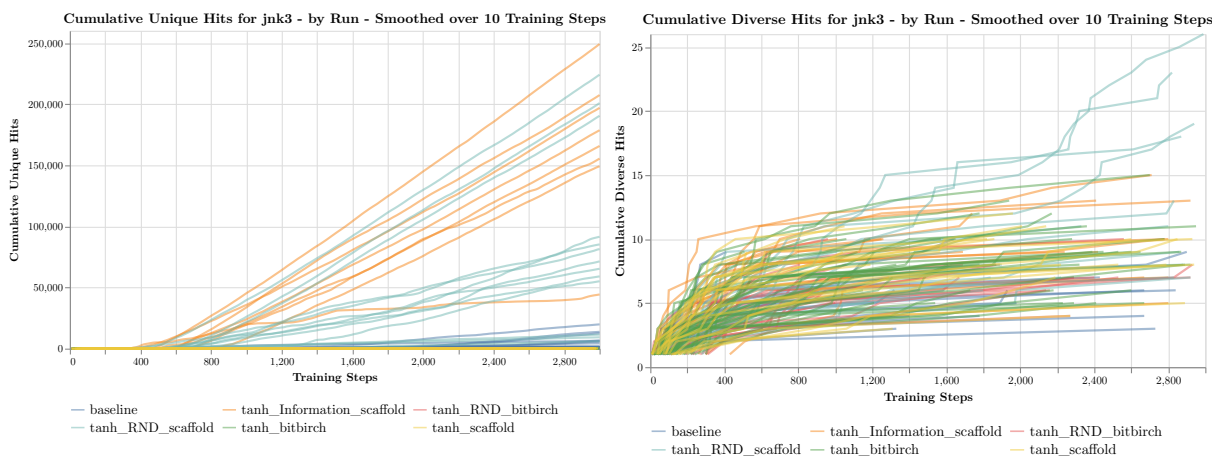


Figure 28 : Average cumulative unique hits and diverse hits across 20 independent runs smoothed over 10 training steps for Jnk3. The tinted area shows the standard deviation across runs. Higher is better as it indicates more exploration.

4. Results

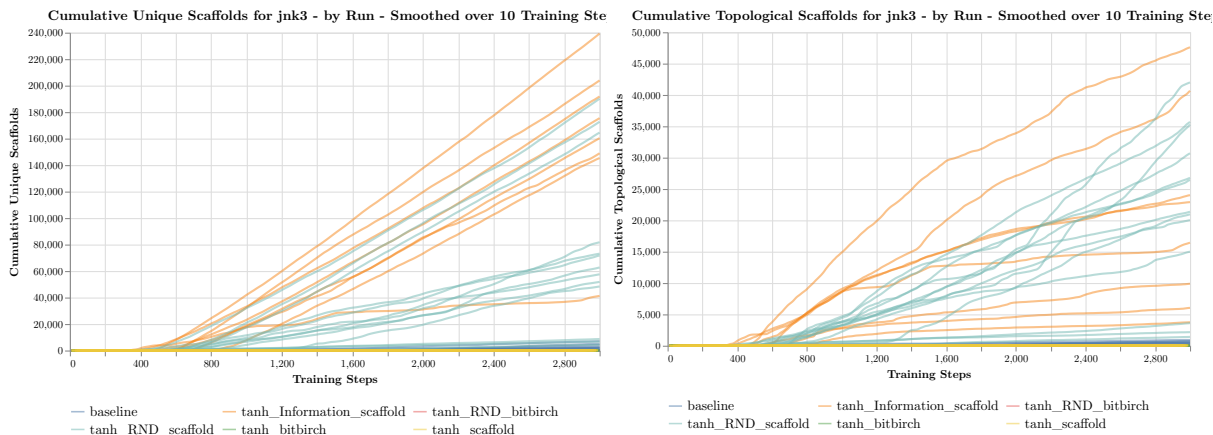


Figure 29 : Average number of cumulative unique scaffolds and unique topological scaffolds across 20 independent runs for Jnk3. .

4.4 Diversity within Experience Replay

We benchmarked the performance of the proposed diversity within experience replay method as introduced in Section 3.5 with $\lambda = 0.2$, $\lambda = 0.5$, $\lambda = 1.0$ (marked with dp02, dp05, and dp10 in the plots) against the baseline REINVENT agent and diversity penalties that were not integrated in the experience replay, but experience replay being turned on (“inception”) and off.

Figure 30 shows that using experience replay speeds up the early stages of learning up to around 300 steps compared to the baseline, but the average score also plateaus earlier with more variance. Combining this with any diversity method except for *tanh_Information_scaffold* increases the plateau effect further. Still experience replay accelerates the process of finding top 128 high scoring molecules as measured by the area under the curve (AUC) under the top 128 scores. While the baseline with experience replay (*baseline_inception*) is the fastest to reach the initial high scores, the scaffold method with Information intrinsic reward actually shows the highest average reward among the diversity methods.

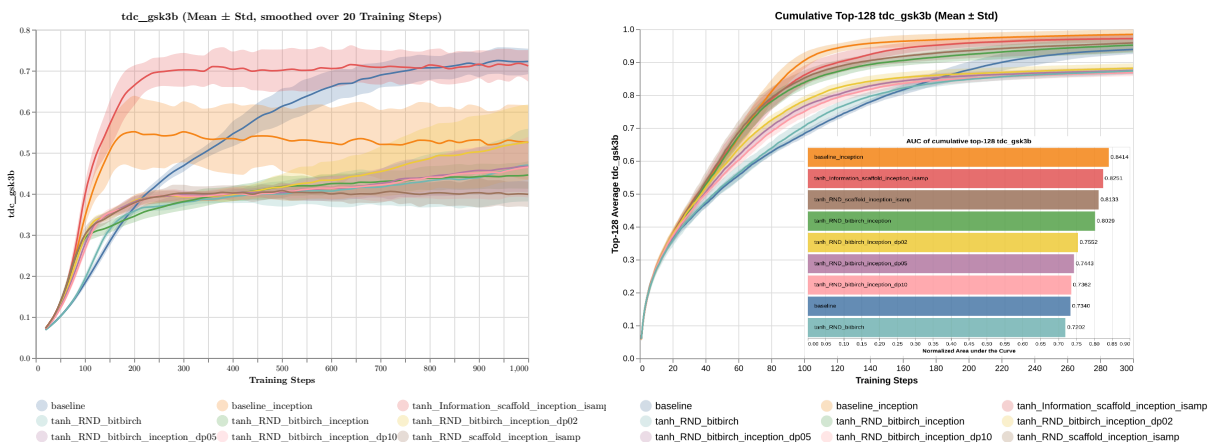


Figure 30 : Average GSK3 β score, top 128 GSK3 β scores and the corresponding area under the curve averaged per batch with a rolling window of 20 steps across 20 independent runs.

4. Results

The combined diversity approach (dp02, dp05, dp10) significantly lowers the speed of early learning and top score found, a lower λ weight results in a less pronounced slowdown, while a higher λ weight further reduces the early learning speed. But the differences are not significant as the exact convergence statistics in Table 4 show.

The convergence step is the absolute step at which the cumulative top-128 average score first reaches 90% of its final value, with a lower value indicating faster convergence. The final score is the cumulative top-128 average score at the last training step, where a higher value indicates better final quality. Score per step measures the average improvement up to convergence, calculated as $\frac{s_{t_{\text{conv}}} - s_0}{t_{\text{conv}}}$; a higher value therefore indicates a faster learning rate. The *dp02* variant with $\lambda = 0.2$ shows the best numbers among the combined methods but only with a small margin. All combined diversity approaches show significantly worse values compared to *baseline_inception*, although there is still an advantage when comparing against no experience replay. More detailed observations and comparisons regarding computational aspects can be found in Appendix B.2.

Table 4 : Convergence statistics for the top 128 GSK3 β scores across 20 runs (mean $\pm$ standard deviation). The best mean and lowest standard deviation in each column are bold.

Method	Convergence step	Final score	Score per step	AUC
baseline	182 $\pm$ 15.1	0.938 $\pm$ 0.014	0.00439 $\pm$ 0.00034	0.734 $\pm$ 0.011
baseline_inception	95.7 $\pm$ 7.4	0.984 $\pm$ 0.015	0.00879 $\pm$ 0.00076	0.841 $\pm$ 0.013
tanh_Information_scaffold_inception	109.7 $\pm$ 14	0.971 $\pm$ 0.012	0.00762 $\pm$ 0.00097	0.825 $\pm$ 0.013
tanh_RND_bitbirch	137.4 $\pm$ 9.3	0.873 $\pm$ 0.011	0.00536 $\pm$ 0.00036	0.72 $\pm$ 0.012
tanh_RND_bitbirch_inception	114.1 $\pm$ 10.8	0.952 $\pm$ 0.023	0.0071 $\pm$ 0.00057	0.803 $\pm$ 0.015
tanh_RND_bitbirch_inception_dp02	107 $\pm$ 6.9	0.881 $\pm$ 0.015	0.00695 $\pm$ 0.00044	0.755 $\pm$ 0.011
tanh_RND_bitbirch_inception_dp05	113.6 $\pm$ 7.2	0.875 $\pm$ 0.01	0.00649 $\pm$ 0.00042	0.744 $\pm$ 0.009
tanh_RND_bitbirch_inception_dp10	115.9 $\pm$ 6.4	0.868 $\pm$ 0.011	0.00631 $\pm$ 0.00038	0.736 $\pm$ 0.01
tanh_RND_scaffold_inception_isamp	109.8 $\pm$ 19.4	0.958 $\pm$ 0.026	0.00757 $\pm$ 0.00115	0.813 $\pm$ 0.018

As expected the combined methods show improvements in diversity though, as the diverse hits in Figure 31 demonstrate. While the scaffold Information approach gives by far the most unique hits, all BitBIRCH methods have a leap in diverse hits. The combined methods even surpass the *tanh_RND_bitbirch* approach without experience replay and improve diversity overall compared to all other methods with conventional experience replay.

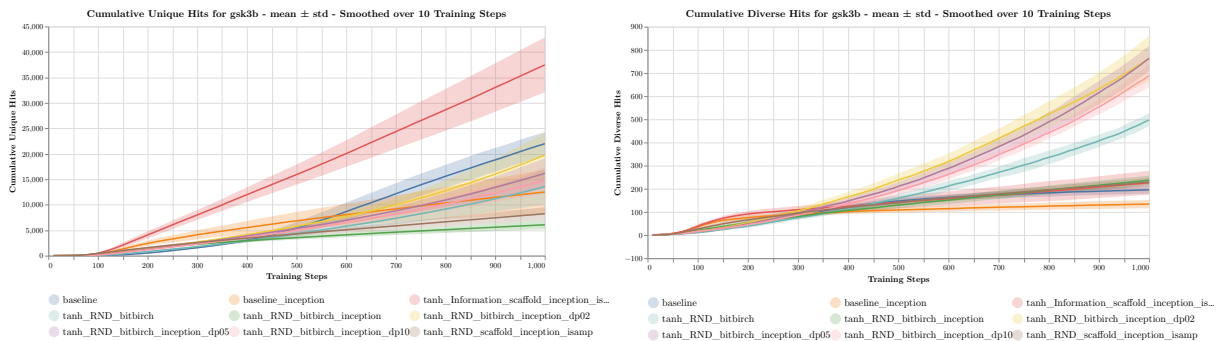


Figure 31 : Average cumulative unique hits and diverse hits across 20 independent runs smoothed over 10 training steps for GSK3 β with and without experience replay.

4.5 Sample Reweighting and Experience Replay

When introducing the sample reweighting from Section 3.4, we do not notice significant differences. Figure 33 shows, for the baseline the average and top-128 scores are nearly identical. When combining sample reweighting with the BitBIRCH penalty, there is even a slight worsening in the average scores after 200 steps, while the top-128 scores remain relatively similar.

Figure 32 shows that diversity metrics are also comparable across the board. Disabling sample reweighting even gives a few more unique and diverse hits. While sample reweighting does not impact the memory consumption, the computational overhead is increased significantly from a few milliseconds to 1 – 1.6 seconds per batch due to the additional evaluations of the agent for all memory entries. This effort also scales with the size of the memory and prior model size, making it potentially costly for large memory buffers. Especially the sampling size also plays a crucial role in the stability and efficiency of the learning process. Appendix B.1 includes an ablation study on the impact of experience memory and sampling size.

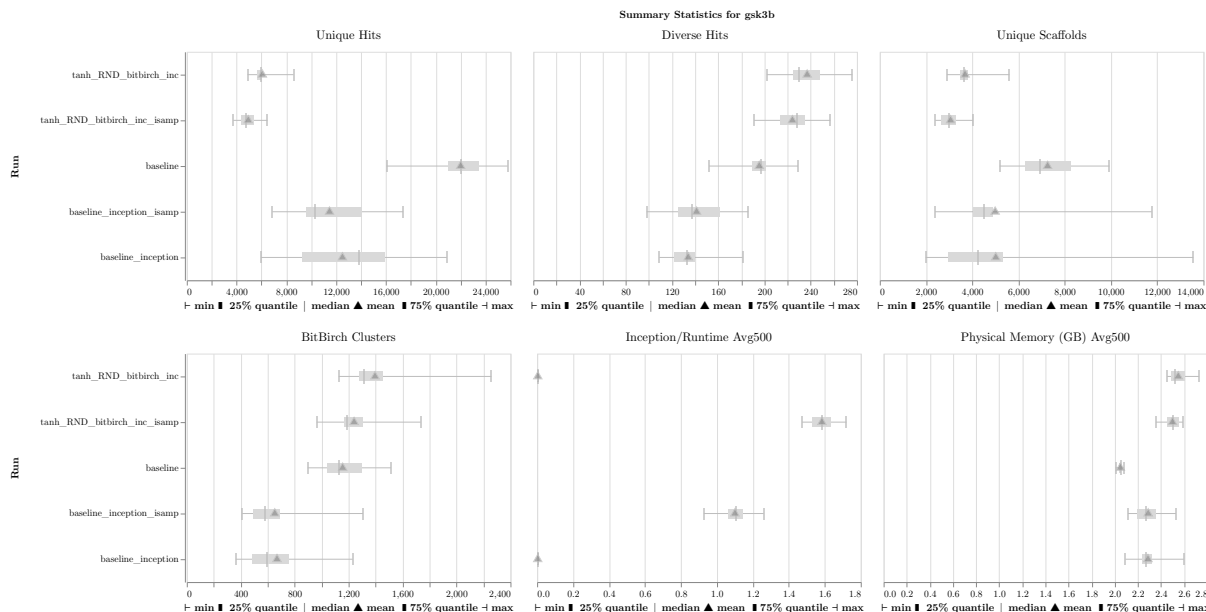


Figure 32 : Learning metrics for 20 independent runs using GSK3 β with sample reweighting and experience replay. The bars show the 25th and 75th percentiles, while the error bars show the min/max value. The black line shows the mean and the triangle the median.

4. Results

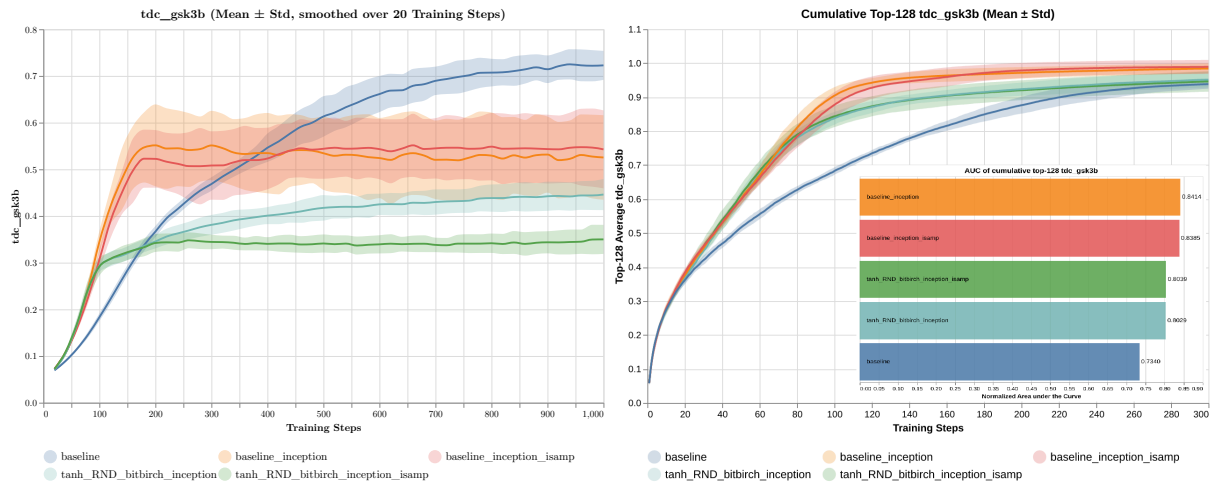


Figure 33 : Average GSK3 β score, top 128 GSK3 β scores and the corresponding area under the curve averaged per batch with a rolling window of 20 steps across 20 independent runs.

5

Discussion

Overall, it shows that using highly efficient clustering methods specialized for molecules like BitBIRCH to measure and maintain diversity in reinforcement learning for generative molecular design is a promising approach. The method seems to be highly effective at finding a high number of unique and diverse hits while maintaining a certain level of drug-likeness, scaffold diversity, and target score comparable to already deployed methods based on scaffolds.

Meanwhile, all three examined targets show different patterns in terms of how the diversity methods perform, which suggests that the effectiveness of the methods might be very target dependent. Especially for Jnk3, it seems almost impossible to find any high scoring molecules, let alone diverse ones, which might be a sign that the scoring function is not very effective for this target, or that the chemical space of high scoring molecules shares little overlap with the prior chemical space. The other two targets are more in line, with the BitBIRCH method showing a clear improvement in terms of diversity metrics such as diverse hits and BitBIRCH clusters.

But the results also show that there is a trade-off between optimizing for the oracle score and maintaining diversity, which is especially evident for gsk3 β , where the BitBIRCH methods have a much lower average oracle score than all other methods, while achieving a very high number of diverse hits and BitBIRCH clusters compared to other methods. While the convergence speed is slowed by the emphasis on maintaining diversity, the top scoring molecules still achieve competitive oracle scores measured by the top 128 scores, which is more relevant practically. This indicates that it is possible to balance diversity and performance to some extent.

Using experience replay universally improves the early learning speed allowing it to achieve higher top oracle scores more consistently. This usually comes at the cost of decreased diversity and more instability. With a combined approach which integrates the proposed penalty framework into the experience replay mechanism, it is possible to achieve a better balance between maintaining diversity and optimizing for the oracle score, leveraging the strengths of both strategies and make the experience buffer less static. Still this approach shows weaknesses, particularly after the initial learning phase where the agent plateaus at a much lower average score and the

benefits of experience replay diminish, making it challenging to maintain both high diversity and high oracle scores simultaneously in the long term.

The results also show possible problems with measuring diversity. For example, the tanh penalized scaffold method with Information Entropy based intrinsic reward seems to be really effective at finding a high number of unique scaffolds, but it finds very few diverse hits, which might be a sign that measuring diversity with scaffolds alone might be misleading, as a high number of unique scaffolds might still be similar to each other in the fingerprint space. If the scaffolds themselves are all very similar to each other with minor modifications, they might not reflect actual diverse exploration

So one needs to look at a variety of similarity and diversity metrics to get a more complete picture of the diversity.

As a conclusion, the effectiveness of the proposed BitBIRCH method for maintaining diversity in generative molecular design is likely highly dependent on the goals of the drug discovery project and the specific target being optimized for, and there is no one-size-fits-all solution for maintaining diversity in generative molecular design.

5.1 Conclusion

This thesis implemented a modular and composable system for employing diversity penalties and experience replay within the REINVENT framework and investigated whether incremental clustering can support structural diversity in reinforcement-learning-based molecular design while considering practical computational constraints. Across the evaluated targets, BitBIRCH-based penalties generally encouraged broader exploration and produced more diverse hits and occupied BitBIRCH clusters than the baseline and scaffold-based alternatives. While the results showed to be dependent on the chosen targets, the effect was clearest for two of the three evaluated targets, DRD2 and GSK3 β , demonstrating that efficient molecular clustering can provide a useful basis for diversity control.

However, increased diversity did not consistently translate into better optimization performance. For GSK3 β in particular, the BitBIRCH methods achieved high diversity while obtaining lower average oracle scores, revealing a trade-off between exploration and exploitation. The competitive top-128 scores nevertheless indicate that the methods can still identify high-quality candidates even when the average generated population scores lower. For JNK3, all methods found few diverse, high-scoring molecules, suggesting that performance also depends strongly on the target, scoring function, and overlap between the prior and the relevant chemical space.

Experience replay accelerated early optimization and improved the consistency of top scores, but generally reduced diversity and provided diminishing benefits later in training. Integrating the BitBIRCH penalty with experience replay produced a more favorable balance, although it did not fully eliminate this conflict. Sample reweighting provided no clear improvement and introduced substantial computational overhead.

Overall, BitBIRCH is a promising and computationally practical mechanism for diversity-aware molecular generation, but it is not universally superior across targets or objectives. The results demonstrate that diversity methods must be evaluated jointly in terms of molecular quality, exploration, computational efficiency, and multiple complementary definitions of diversity.

As part of this work, a modular composable system for employing diversity penalties and experience replay within the REINVENT framework and a benchmarking platform for evaluating diversity methods in generative molecular design has emerged, both of which are open-source and can be used by other researchers to evaluate their own targets and methods to compare them to existing ones and see the effectiveness of different approaches.

5.2 Future Work

This work led to an ongoing publication effort highlighting the strengths and limitations of the integrated BitBIRCH method in particular. As part of the thesis the hyperparameter space was mostly kept fixed to reduce the complexity of the experiments and focus on evaluating the effectiveness of the proposed method. A more thorough exploration of the hyperparameter space could provide deeper insights into the robustness and generalizability, potentially revealing optimal configurations that balance diversity and oracle score more effectively.

Second, it would be interesting to further investigate the trade-off between optimizing for the oracle score and maintaining diversity, and see if there are ways to mitigate this trade-off, for example by using staged training approaches where the agent first focuses on optimizing for the oracle score and then gradually increases the importance of diversity. Especially for $\text{gsk3}\beta$, it would be interesting to see if the same pattern of a trade-off between oracle score and diversity emerges when the agent is already in a high scoring region and a BitBIRCH filter is gradually applied.

Finally, it would be interesting to further evaluate the proposed method on more targets and different prior backbones like transformers, which recent work has shown to be very effective for generative molecular design, to see if the results generalize across different settings and to further understand the strengths and weaknesses of the method [9].

Bibliography

- [1] V. Blay, R. García-Domenech, and J. Galvez, “Problem solving with Molecular Topology: a walkthrough,” *ChemTexts*, vol. 3, p. 2, Mar. 2017, doi: 10.1007/s40828-017-0039-7.
- [2] M. Olivecrona, T. Blaschke, O. Engkvist, and H. Chen, “Molecular de-novo design through deep reinforcement learning,” *Journal of Cheminformatics*, vol. 9, no. 1, p. 48, Sep. 2017, doi: 10.1186/s13321-017-0235-x.
- [3] T. Blaschke, O. Engkvist, J. Bajorath, and H. Chen, “Memory-assisted reinforcement learning for diverse molecular de novo design,” *Journal of Cheminformatics*, vol. 12, no. 1, p. 68, Nov. 2020, doi: 10.1186/s13321-020-00473-0.
- [4] 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,” 2023.
- [5] H. Gummesson Svensson, C. Tyrchan, O. Engkvist, and M. Haghiri Chehreghani, “Utilizing reinforcement learning for de novo drug design,” *Machine Learning*, vol. 113, no. 7, pp. 4811–4843, Jul. 2024, doi: 10.1007/s10994-024-06519-w.
- [6] J. Guo and P. Schwaller, “Augmented Memory: Capitalizing on Experience Replay to Accelerate De Novo Molecular Design.” Accessed: Feb. 13, 2026. [Online]. Available: <http://arxiv.org/abs/2305.16160>
- [7] P. G. Polishchuk, T. I. Madzhidov, and A. Varnek, “Estimation of the size of drug-like chemical space based on GDB-17 data,” *Journal of Computer-Aided Molecular Design*, vol. 27, no. 8, pp. 675–679, Aug. 2013, doi: 10.1007/s10822-013-9672-4.
- [8] A. Gupta, A. T. Müller, B. J. H. Huisman, J. A. Fuchs, P. Schneider, and G. Schneider, “Generative Recurrent Networks for De Novo Drug Design,” *Molecular Informatics*, vol. 37, no. 1, p. 1700111, 2018, doi: 10.1002/minf.201700111.
- [9] H. H. Loeffler *et al.*, “Reinvent 4: Modern AI-driven generative molecule design,” *Journal of Cheminformatics*, vol. 16, no. 1, p. 20, Feb. 2024, doi: 10.1186/s13321-024-00812-5.

- [10] K. L. Pérez, V. Jung, L. Chen, K. Huddleston, and R. A. Miranda-Quintana, "BitBIRCH: efficient clustering of large molecular libraries," *Digital Discovery*, vol. 4, no. 4, pp. 1042–1051, 2025, doi: 10.1039/D5DD00030K.
- [11] H. G. Svensson, C. Tyrchan, O. Engkvist, and M. H. Chehreghani, "Diversity-Aware Reinforcement Learning for de novo Drug Design." Accessed: Nov. 30, 2025. [Online]. Available: <https://arxiv.org/abs/2410.10431>
- [12] G. Landrum *et al.*, "RDKit: Open-source cheminformatics. <https://www.rdkit.org>," Aug. 2026, doi: 10.5281/zenodo.22140358.
- [13] D. Weininger, "SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules."
- [14] D. Rogers and M. Hahn, "Extended-Connectivity Fingerprints," *Journal of Chemical Information and Modeling*, vol. 50, no. 5, pp. 742–754, Apr. 2010, doi: 10.1021/ci100050t.
- [15] S. Riniker and G. A. Landrum, "Open-source platform to benchmark fingerprints for ligand-based virtual screening," *Journal of Cheminformatics*, vol. 5, no. 1, p. 26, May 2013, doi: 10.1186/1758-2946-5-26.
- [16] B. Zdrazil *et al.*, "The ChEMBL Database in 2023: a drug discovery platform spanning multiple bioactivity data types and time periods," *Nucleic Acids Research*, vol. 52, no. D1, pp. D1180–D1192, Jan. 2024, doi: 10.1093/nar/gkad1004.
- [17] H. G. Svensson, O. Engkvist, J. P. Janet, C. Tyrchan, and M. H. Chehreghani, "Diverse Mini-Batch Selection in Reinforcement Learning for Efficient Chemical Exploration in de novo Drug Design." Accessed: Feb. 10, 2026. [Online]. Available: <http://arxiv.org/abs/2506.21158>
- [18] D. Butina, "Unsupervised Data Base Clustering Based on Daylight's Fingerprint and Tanimoto Similarity: A Fast and Automated Way To Cluster Small and Large Data Sets," *Journal of Chemical Information and Computer Sciences*, vol. 39, no. 4, pp. 747–750, Jun. 1999, doi: 10.1021/ci9803381.
- [19] D. Bajusz, A. Rácz, and K. Héberger, "Why is Tanimoto index an appropriate choice for fingerprint-based similarity calculations?," *Journal of Cheminformatics*, vol. 7, no. 1, p. 20, May 2015, doi: 10.1186/s13321-015-0069-3.
- [20] K. López-Pérez, T. D. Kim, and R. A. Miranda-Quintana, "iSIM: instant similarity," *Digital Discovery*, vol. 3, no. 6, pp. 1160–1171, 2024, doi: 10.1039/D4DD00041B.
- [21] P. Renz, S. Luukkonen, and G. Klambauer, "Diverse Hits in De Novo Molecule Design: Diversity-Based Comparison of Goal-Directed Generators," *J. Chem. Inf. Model.*, 2024.

- [22] G. W. Bemis and M. A. Murcko, "The Properties of Known Drugs. 1. Molecular Frameworks," *Journal of Medicinal Chemistry*, vol. 39, no. 15, pp. 2887–2893, Jan. 1996, doi: 10.1021/jm9602928.
- [23] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: an efficient data clustering method for very large databases," *SIGMOD Rec.*, vol. 25, no. 2, pp. 103–114, Jun. 1996, doi: 10.1145/235968.233324.
- [24] I. Pickering, K. Zsigmond, K. L. Pérez, M. Lžičař, and R. A. Miranda-Quintana, "BitBIRCH-Lean: chemical space in the palm of your workstation." Accessed: Nov. 14, 2025. [Online]. Available: <http://biorxiv.org/lookup/doi/10.1101/2025.10.22.684015>
- [25] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Machine Learning*, vol. 8, no. 3, pp. 293–321, May 1992, doi: 10.1007/BF00992699.
- [26] T. Blaschke *et al.*, "REINVENT 2.0 – an AI Tool for De Novo Drug Design," *ChemRxiv*, vol. 2020, no. 403, doi: 10.26434/chemrxiv.12058026.v1.
- [27] R. S. Sutton and A. G. Barto, *Reinforcement learning: an introduction*, Second edition. in Adaptive computation and machine learning series. Cambridge, Massachusetts: The MIT Press, 2018.
- [28] A. R. Mahmood, H. van Hasselt, and R. S. Sutton, "Weighted importance sampling for off-policy learning with linear function approximation," in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2014. Accessed: Sep. 04, 2026. [Online]. Available: <https://proceedings.neurips.cc/paper/2014/hash/1c64ee92596e8ea5050fc435a1d57459-Abstract.html>
- [29] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized Experience Replay." Accessed: Feb. 24, 2026. [Online]. Available: <http://arxiv.org/abs/1511.05952>
- [30] Y. Burda, H. Edwards, A. Storkey, and O. Klimov, "Exploration by Random Network Distillation." Accessed: Sep. 04, 2026. [Online]. Available: <http://arxiv.org/abs/1810.12894>
- [31] K. Huang *et al.*, "Therapeutics Data Commons: Machine Learning Datasets and Tasks for Drug Discovery and Development." Accessed: Feb. 20, 2026. [Online]. Available: <http://arxiv.org/abs/2102.09548>
- [32] B. Medina, A. Tibo, J. He, J. P. Janet, and N. Österbacka, "Diversity Beats Size Scaling for Chemical Language Models," *ChemRxiv*, vol. 2025, no. 1118, doi: 10.26434/chemrxiv-2025-k14gv.
- [33] B. Medina, A. Tibo, J. He, J. P. Janet, and N. Österbacka, "Diversity Beats Size Scaling for Chemical Language Models," *npj Drug Discovery*, vol. 3, no. 1, p. 26, Aug. 2026, doi: 10.1038/s44386-026-00056-3.

A

Alternative Methods and Approaches

In addition to the BitBIRCH approach, more conventional clustering methods were briefly explored to improve diversity in reinforcement learning for generative molecular design. Two such methods are Kmeans and Kcentroids clustering, which can be adapted to work with molecular fingerprints and Tanimoto similarity. A third conventional clustering method specifically for molecules, Taylor-Butina clustering, was also considered. But all three methods scale quadratically with the number of molecules in memory, which makes them less suitable for reinforcement learning, where the number of molecules in memory can be very high. Also, all three methods require a predefined number of clusters, which can be a limitation compared to the dynamic nature of BitBIRCH and the scaffold approach and makes them hard to compare. The verdict is that BitBIRCH gets a similar or better diversity performance than the other methods, while being much more efficient in terms of runtime and memory usage, which makes it more suitable for reinforcement learning for generative molecular design and led to discontinuing exploration of the alternatives.

A.1 Kmeans

Kmeans stores the arithmetic mean $\mu_j = \frac{1}{n_j} \sum_{i \in C_j} \mathbf{x}_i$, of all fingerprints for each cluster. Upon insertion of a new fingerprint $\mathbf{x}_{\text{new}}$ into cluster C_j , the cluster which maximizes the tanimoto similarity to the cluster mean is chosen and the mean is updated accordingly.

Similar to the BitBIRCH approach, a penalty can be applied based on the occupancy of each Kmeans cluster in the memory. However, Kmeans requires a predefined number of clusters, which is a clear limitation compared to the dynamic nature of BitBIRCH. This adds another hyperparameter to the method, which can be hard to tune and makes it hard to compare to other methods since a small number of clusters might not capture the space adequately. BitBirch scales itself dynamically based on the data and the threshold, which makes it more flexible and easier to use in practice.

A.2 KCentroids

Kcentroids is similar to Kmeans but uses the median instead of mean value for each cluster, which is more intuitive for binary fingerprints, can be more robust to outliers and may provide better clustering for binary fingerprints. Like Kmeans, it also requires a predefined number of clusters and can be adapted to use Tanimoto similarity by comparing. But it also suffers from the same limitations as Kmeans in terms of scaling and hyperparameter tuning.

A.3 Taylor-Butina

Taylor-Butina clustering is a method specifically designed for molecules and uses Tanimoto similarity to cluster molecules based on their fingerprints. It is a greedy algorithm that iteratively selects the molecule with the heighest number of neighbors within a cutoff θ as the cluster center and assigns all molecules within a certain Tanimoto similarity threshold to that cluster [18]. While it is specifically designed for molecules and uses Tanimoto similarity, we have to store all pairs of similarities essentially which gives a scaling of $\frac{n(n-1)}{2} \in O(n^2)$. This makes it impractical in iterative resource-constrained reinforcement learning settings, where the number of molecules in memory can be very high.

A.4 Extended Experiments

In the scope of this thesis it was not possible to conclude definitive results regarding the comparative performance of the different clustering methods due to computational and time constraints. Early experiments shown in Figure 34 were not successful in a sense that the alternative methods decreased the average score massively and increased the runtime significantly such that the computational budget of 3 hours was quickly exhausted. Taylor-Butina, in particular, was very slow with more than 12 seconds per step. While these methods show potential when looking at diverse hits, their limitation with fixed number of clusters makes them hard to compare and were deliberately not pursued further in this thesis.

A. Alternative Methods and Approaches

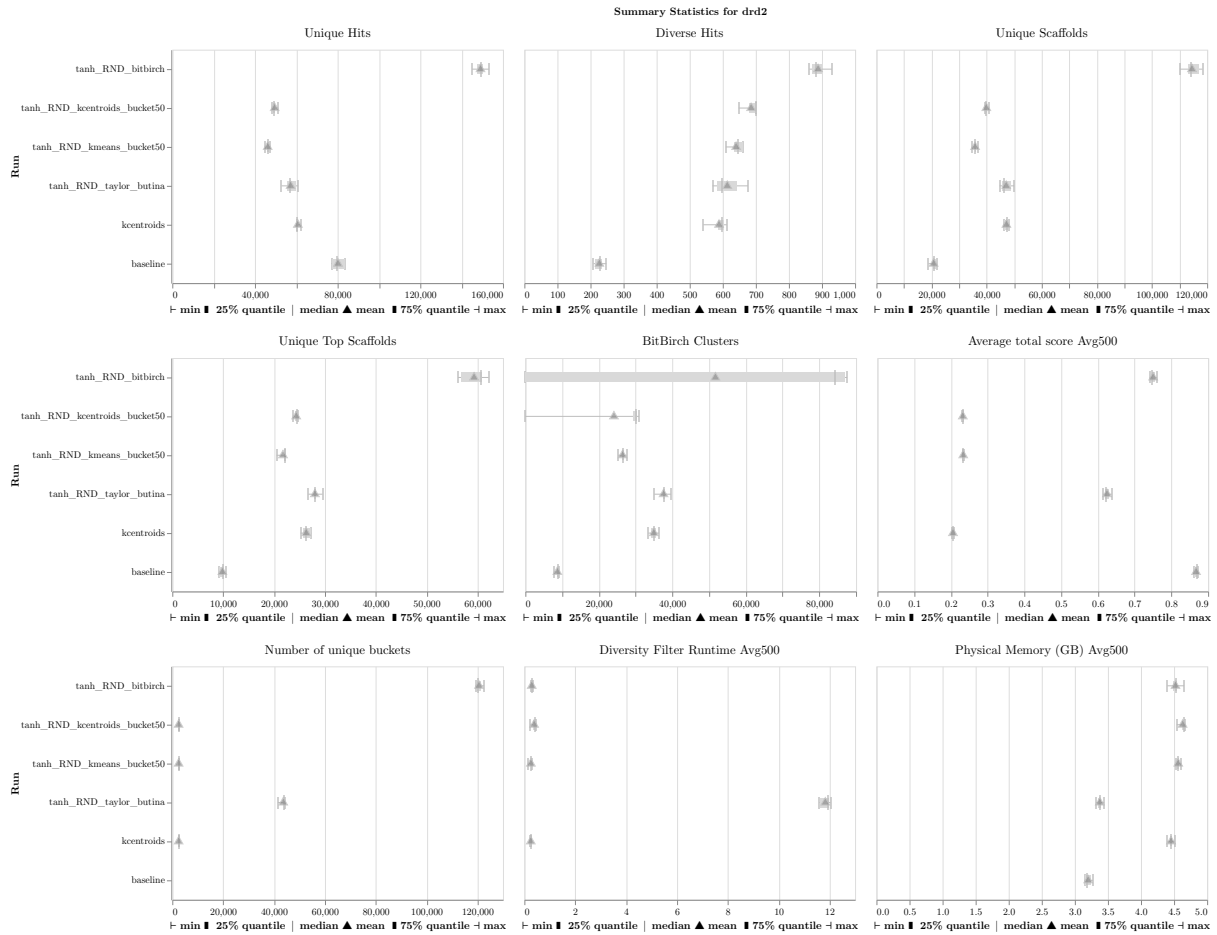


Figure 34 : Learning metrics for 20 independent runs using DRD2 and alterntive clustering approaches. The bars show the 25th and 75th percentiles, while the error bars show the min/max value. The black line shows the mean and the triangle the median.

B

Extended Results

B.1 Impact of Experience Memory and Sampling Size

An ablation study was conducted to evaluate the impact of experience memory and sampling size on the performance. Figure 35 shows that especially the sample size has a significant effect on the learning outcomes. In general, the larger the sample size, the faster the initial learning, but at the expense of more instability and earlier plateauing of the average score. This is also reflected in the convergence statistics shown in Table 5. The higher sample sizes like 64 or 128 tend to reach convergence faster but also exhibit more variability in the final scores. This makes sense as the batch size was set to 128, so sampling more molecules from the experience replay should pronounce the effect more. For further experiments, a memory size of 128 and a sample size of 64 has been chosen to see the effects of experience replay clearly. In practice a lower sample size might be preferred to gain more stability and reduce computational overhead.

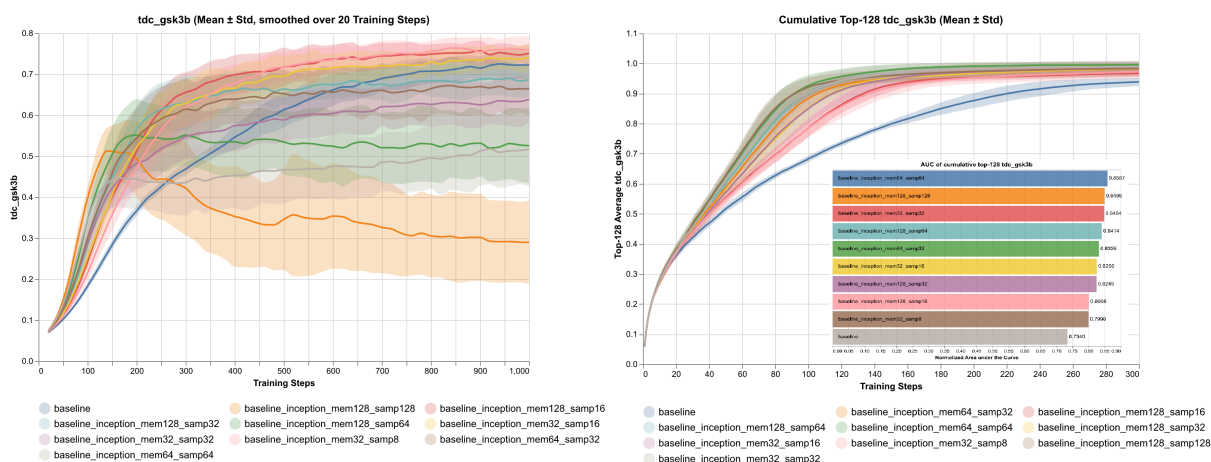


Figure 35 : Average GSK3 β score, top 128 GSK3 β scores and the corresponding area under the curve averaged per batch with a rolling window of 5 steps across 20 independent runs.

B. Extended Results

Table 5 : Convergence statistics for the top 128 GSK3 β scores across 20 runs (mean $\pm$ standard deviation). The best mean and lowest standard deviation in each column are bold.

Method	Convergence step	Final score	Score per step	AUC
baseline	182 $\pm$ 15.1	0.938 $\pm$ 0.014	0.00439 $\pm$ 0.00034	0.734 $\pm$ 0.011
baseline_inception_isamp	105.6 $\pm$ 10.1	0.989 $\pm$ 0.02	0.00803 $\pm$ 0.0008	0.839 $\pm$ 0.019
baseline_inception_mem128_samp128	88.6 $\pm$ 11.5	0.983 $\pm$ 0.026	0.00956 $\pm$ 0.00119	0.85 $\pm$ 0.021
baseline_inception_mem128_samp64	95.7 $\pm$ 7.4	0.984 $\pm$ 0.015	0.00879 $\pm$ 0.00076	0.841 $\pm$ 0.013
baseline_inception_mem128_samp16	124.3 $\pm$ 8.6	0.967 $\pm$ 0.011	0.00663 $\pm$ 0.0005	0.801 $\pm$ 0.013
baseline_inception_mem128_samp32	109.5 $\pm$ 8.9	0.983 $\pm$ 0.012	0.00767 $\pm$ 0.00066	0.825 $\pm$ 0.012
baseline_inception_mem32_samp16	112.5 $\pm$ 16.4	0.984 $\pm$ 0.017	0.00758 $\pm$ 0.00108	0.826 $\pm$ 0.016
baseline_inception_mem32_samp32	97.9 $\pm$ 13.8	0.989 $\pm$ 0.017	0.00877 $\pm$ 0.00145	0.848 $\pm$ 0.021
baseline_inception_mem32_samp8	129.1 $\pm$ 16.1	0.972 $\pm$ 0.023	0.00647 $\pm$ 0.00074	0.8 $\pm$ 0.017
baseline_inception_mem64_samp32	101.6 $\pm$ 11.8	0.982 $\pm$ 0.014	0.0083 $\pm$ 0.00095	0.833 $\pm$ 0.013
baseline_inception_mem64_samp64	92 $\pm$ 11.8	0.996 $\pm$ 0.01	0.00935 $\pm$ 0.00116	0.859 $\pm$ 0.013
tanh_RND_bitbirch	137.4 $\pm$ 9.3	0.873 $\pm$ 0.011	0.00536 $\pm$ 0.00036	0.72 $\pm$ 0.012
tanh_RND_bitbirch_inception	114.1 $\pm$ 10.8	0.952 $\pm$ 0.023	0.0071 $\pm$ 0.00057	0.803 $\pm$ 0.015
tanh_RND_bitbirch_inception_isamp	110.4 $\pm$ 18.6	0.946 $\pm$ 0.03	0.00742 $\pm$ 0.00117	0.804 $\pm$ 0.023
tanh_RND_scaffold	133.6 $\pm$ 7.5	0.871 $\pm$ 0.011	0.00547 $\pm$ 0.00031	0.722 $\pm$ 0.008
tanh_RND_scaffold_inception_isamp	109.8 $\pm$ 19.4	0.958 $\pm$ 0.026	0.00757 $\pm$ 0.00115	0.813 $\pm$ 0.018
tanh_bitbirch_inception_isamp	128.1 $\pm$ 14.4	0.969 $\pm$ 0.011	0.00649 $\pm$ 0.00071	0.809 $\pm$ 0.013

B.2 Diversity in Experience Replay

Figure 36 shows further diversity and learning metrics for the various experience replay strategies with and without combined diversity penalty discussed in Section 4.4. Particularly interesting is that the combined diversity penalty seems to improve the diversity of *tanhBitBirch* compared to without experience replay further according to diverse hits, hamiltonian diversity, and BitBIRCH cluster diversity metrics. On Scaffolds and Unique Hits, the scaffold Information seems to excel.

When looking at the inception/replay memory during the learning process, the effects of the diversity penalties become apparent, with a significantly higher turnover of molecules in the replay buffer and lower mean, bottom and top scores compared to the baseline without diversity penalties. Especially the actual samples that get injected into the learning loop have a significantly lower score which might explain the slower early learning and plateaus observed for the combined diversity approaches. These also take slightly more memory and runtime. Although with 0.2 seconds per step, the overhead is relatively minor compared to importance sampling methods which can take up to 1.2 seconds per step.

B. Extended Results

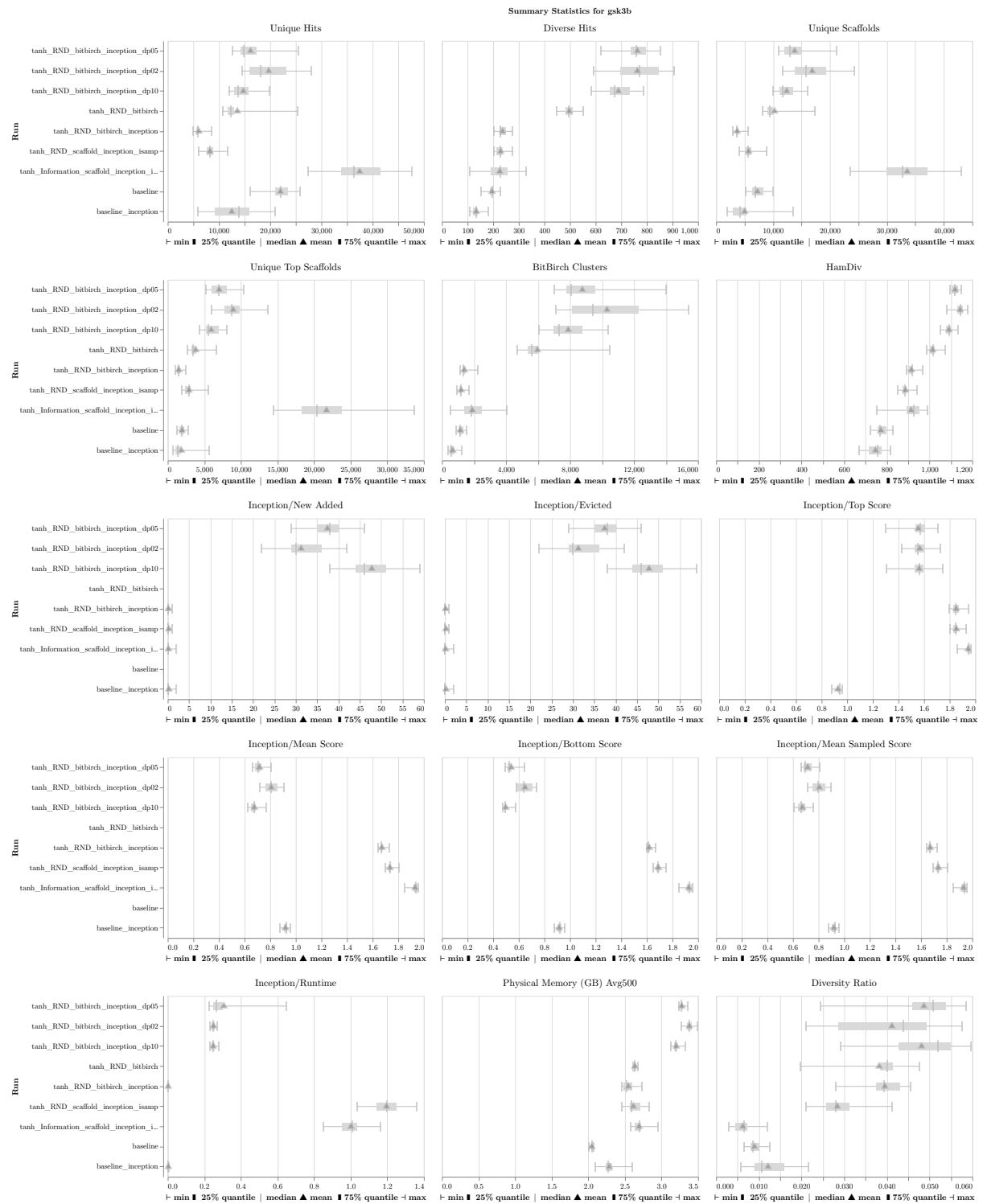


Figure 36 : Learning metrics for 20 independent runs using GSK3 β and various forms of experience replay. The bars show the 25th and 75th percentiles, while the error bars show the min/max value. The black line shows the mean and the triangle the median.