Biological Sequence Design with GFlowNets

Moksh Jain [1 2] **Emmanuel Bengio** [1 3] **Alex-Hernandez Garcia** [1 2] **Jarrid Rector-Brooks** [1 2]
Bonaventure F. P. Dossou [1 2 4] **Chanakya Ekbote** [1] **Jie Fu** [1 2] **Tianyu Zhang** [1 2] **Michael Kilgour** [5]
Dinghuai Zhang [1 2] **Lena Simine** [3] **Payel Das** [6] **Yoshua Bengio** [1 2 7]

Abstract

Design of *de novo* biological sequences with desired properties, like protein and DNA sequences, often involves an active loop with several rounds of molecule ideation and expensive wet-lab evaluations. These experiments can consist of multiple stages, with increasing levels of precision and cost of evaluation, where candidates are filtered. This makes the diversity of proposed candidates a key consideration in the ideation phase. In this work, we propose an active learning algorithm leveraging epistemic uncertainty estimation and the recently proposed GFlowNets as a generator of diverse candidate solutions, with the objective to obtain a diverse batch of useful (as defined by some utility function, for example, the predicted anti-microbial activity of a peptide) and informative candidates after each round. We also propose a scheme to incorporate existing labeled datasets of candidates, in addition to a reward function, to speed up learning in GFlowNets. We present empirical results on several biological sequence design tasks, and we find that our method generates more diverse and novel batches with high scoring candidates compared to existing approaches.

1. Introduction

Biological sequences like proteins and DNA have a broad range of applications to several impactful problems ranging from medicine to material design. For instance, design of novel anti-microbial peptides (AMPs; short sequences of amino-acids) is crucial, and identified as the first target to tackle the growing public health risks posed by increasing anti-microbial resistance (AMR; Murray et al., 2022). This is particularly alarming according to a recent report[2] by the World Health Organization, which predicts millions of human lives lost per year (with the potential breakdown of healthcare systems and many more indirect deaths), unless methods to efficiently control (and possibly stop) the fast-growing AMR are found.

Considering the diverse nature of the biological targets, modes of attack, structures, as well as the evolving nature of such problems, diversity becomes a key consideration in the design of these sequences (Mullis et al., 2019). Another reason for the importance of being able to propose a *diverse* set of good candidates is that cheap screening methods (like in-silico simulations or in-vitro experiments) may not reflect well future outcomes in animals and humans, as illustrated in Figure 1. To maximize the chances that at least one of the candidates will work in the end, it is important for these candidates to cover as much as possible the modes of a *goodness* function that estimates future success. The design of new biological sequences involves searching over combinatorially large discrete search spaces on the order of $\mathcal{O}(10^{60})$ candidates. Machine learning methods that can exploit the combinatorial structure in these spaces (e.g., due to laws of physics and chemistry) have the potential to speed up the design process for such biological sequences (Pyzer-Knapp, 2018; Terayama et al., 2021; Das et al., 2021).

The development process of such biological sequences, for a particular application, involves several rounds of a candidate ideation phase (possibly starting with a random library) followed by an evaluation phase, as shown in Figure 1. The evaluation consists of several stages ranging from numerical simulations to expensive wet-lab experiments, possibly culminating in clinical trials. These stages filter candidates with progressively higher fidelity oracles that measure different aspects of the *usefulness* of a candidate. For example, the typical evaluation for an antibiotic drug after ideation would comprise of: (1) in-silico screening using approximate models to estimate anti-microbial activity of $\mathcal{O}(10^6)$ candidates (2) in-vitro experiments to measure single-cell effectiveness against a target bacterium species of $\mathcal{O}(10^3)$

[1]Mila [2]Université de Montréal [3]McGill University [4]Jacobs University Bremen [5]New York University [6]IBM [7]CIFAR Fellow and AI Chair. Correspondence to: Moksh Jain <mokshjn00@gmail.com>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

[2] https://www.who.int/news-room/fact-sheets/detail/antibiotic-resistance

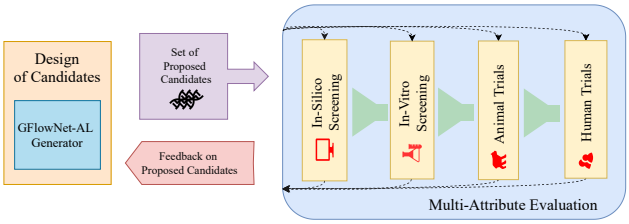

Figure 1. Illustration of a typical drug discovery pipeline. In each round, a set of candidates is proposed which are evaluated under various stages of evaluation, each measuring different properties of the candidates with varying levels of precision. The design procedure is then updated using the feedback received from the evaluation phase before the next round begins. Because the early screening phases are imperfect, and the ideal "usefulness" of the candidate can be ill-defined, it is important to generate for these phases a *diverse* set of candidates (rather than many similar candidates who could all fail in the downstream phases).

candidates (3) trials in small mammals like mice with $\mathcal{O}(10)$ candidates (4) randomized human trials with $\mathcal{O}(1)$ candidates. These oracles are often imperfect and do not evaluate all the required properties of a candidate.

The biological repertoire of DNA, RNA and protein sequences is extremely diverse, to support the diversity of structure, function and modes of action exploited by living organisms, where the same high-level function can be potentially executed in more than one possible manner (Mullis et al., 2019). Moreover, the ultimate success of candidate drugs also depends on satisfying multiple often conflicting desiderata, not all of which likely can be precisely estimated in-silico. This fact, combined with the overall effect of the above aggressive filtering and use of potentially imperfect oracles, needs to be addressed in the design phase through the *diversity* of the generated candidates. Diverse candidates capturing the *modes* of the imperfect oracle improve the likelihood of discovering a candidate that can satisfy all (or many) evaluation criteria, because failure in downstream stages is likely to affect nearby candidates (from the same mode of the oracle function), while different modes are likely to correspond to qualitatively different properties.

This setup of iteratively proposing a batch of candidates and learning from the feedback provided by an oracle on that batch fits into the framework of active learning (Aggarwal et al., 2014). Bayesian Optimization is a common approach for such problems (Rasmussen & Williams, 2005; Garnett, 2022). It relies on a Bayesian surrogate model of the usefulness function of interest (e.g., the degree of binding of a candidate drug to a target protein), with an output variable Y that we can think of as a reward for a candidate X. An acquisition function $\mathcal{F}$ is defined on this surrogate model and a pool of candidates will be screened to search for candidates x with a high value of $\mathcal{F}(x)$. That acquisition function combines the expected reward function μ (e.g., $\mu(x)$ can be the probability of obtaining a successful candidate) as well as an estimator of epistemic uncertainty $\sigma(x)$ around $\mu(x)$, to favour candidates likely to bring new information to the learner. There are many possible candidate selection procedures, from random sampling to genetic algorithms evolving a population of novel candidates (Pyzer-Knapp, 2018; Belanger et al., 2019; Moss et al., 2020; Swersky et al., 2020; Terayama et al., 2021). An alternative is to use Reinforcement Learning (RL) to maximize the value of a surrogate model of the oracle (Angermueller et al., 2019). RL methods are designed to search for a single candidate that maximizes the oracle, which can result in poor diversity and can cause candidate generation to get *stuck* in a single mode (Bengio et al., 2021a) of the expected reward function. Additionally, as the final goal is to find *novel* designs that are different from the ones that are already known, the generative model must be able to capture the tail ends of the data distribution.

In settings where diversity is important, another interesting way to generate candidates is to use a generative policy that can sample candidates proportionally to a reward function (for instance, the acquisition function over a surrogate model) and can be sampled i.i.d to obtain a set of candidates that covers well the modes of the reward function. A sample covering the modes approximately but naturally satisfies the ideal criterion of high scoring and diverse candidates. GFlowNets (Bengio et al., 2021a) provide a way to learn such a stochastic policy and, unlike Markov chain Monte Carlo (MCMC) methods (which also have this ability), amortize the cost of each new i.i.d. sample (which may require a lengthy chain, with MCMC methods) into the cost of training the generative model (Zhang et al., 2022). As such, this paper is motivated by the observation that GFlowNets are appealing in the above Bayesian optimization context, compared with existing RL and MCMC approaches in domains such as small molecule synthesis.

In this work, we present an active learning algorithm based on a GFlowNet generator for the task of biological sequence design. In addition to this, we propose improvements to the GFlowNet training procedure to improve performance in active learning settings. We apply our proposed approach on a broad variety of biological sequence design tasks. The key contributions of this work are summarized below:

- An active learning algorithm with GFlowNet as the generator for designing novel biological sequences.
- Investigating the effect of off-policy updates from a static dataset to speed up training of GFlowNets.
- Incorporating the epistemic uncertainty in the predicted expected reward to improve exploration in GFlowNets.

- Validating the proposed algorithm on three protein and DNA design tasks.

2. Problem Setup

We consider the problem of searching over a space of discrete objects $\mathcal{X}$ to find objects $x \in \mathcal{X}$ that maximize a given usefulness measure (oracle) $f : \mathcal{X} \mapsto \mathbb{R}^+$. We consider the setting where this oracle can only be queried N times in fixed batches of size b. This constitutes N rounds of evaluation available to the active learning algorithm. The algorithm also has access to an initial dataset $D_0 = \{(x_1^0, y_1^0), \ldots, (x_n^0, y_n^0)\}$, where $y_i^0 = f(x_i^0)$ from evaluations by the oracle.

The algorithm has to propose a new batch of candidates $\mathcal{B}_i = \{x_1^i, \ldots, x_b^i\}$, given the current dataset $\mathcal{D}_i$, in each round $i \in \{1, \ldots, N\}$. This batch is then evaluated on the oracle to obtain the corresponding scores for the candidates $y_j^i = f(x_j^i)$. The current dataset $\mathcal{D}_i$ is then augmented with the tuples of the proposed candidates and their scores to generate the dataset for the next round, $\mathcal{D}_{i+1} = \mathcal{D}_i \cup \{(x_1^i, y_1^i), \ldots, (x_b^i, y_b^i)\}$.

This problem setup is similar to the standard black-box optimization problem (Audet & Hare, 2017) with one difference: the initial dataset D_0 is available as a starting point, which is actually a common occurrence in practice, i.e., a historical dataset. This setup can also be viewed as an extension of the Offline Model Based Optimization (Trabucco et al., 2021a;b) paradigm to multiple rounds instead of a single round.

Desiderata for Proposed Candidates As discussed in Section 1, searching for a single candidate maximizing the oracle can be problematic in the typical scenario where the available (cheap, front-line) oracle is imperfect. Instead, we are interested in looking for a diverse set of K top candidates generated by the algorithm, $\mathcal{D}_{\text{Best}} = \text{TopK}(\mathcal{D}_K \setminus \mathcal{D}_0)$. We outline the key characteristics that define the set of *ideal* candidates.

- **Performance/Usefulness Score**: The base criteria is for the set to include high scoring candidates, which can be quantified with

$$\text{Mean}(\mathcal{D}) = \frac{\sum_{(x_i, y_i) \in \mathcal{D}} y_i}{|\mathcal{D}|} \tag{1}$$

- **Diversity**: In addition to being high scoring, we would like the candidates to capture the modes of the oracle. One way to measure this is

$$\text{Diversity}(\mathcal{D}) = \frac{\sum_{(x_i, y_i) \in \mathcal{D}} \sum_{(x_j, y_j) \in \mathcal{D} \setminus \{(x_i, y_i)\}} d(x_i, x_j)}{|\mathcal{D}|(|\mathcal{D}| - 1)} \tag{2}$$

where d is a distance measure defined over $\mathcal{X}$.

- **Novelty**: Since we start with an initial dataset $\mathcal{D}_0$, the proposed candidates should also be different from the candidates that are already known. We measure this *novelty* in the proposed candidates as follows:

$$\text{Novelty}(\mathcal{D}) = \frac{\sum_{(x_i, y_i) \in \mathcal{D}} \min_{s_j \in \mathcal{D}_0} d(x_i, s_j)}{|\mathcal{D}|} \tag{3}$$

All three metrics are applied on the TopK scoring candidates, i.e., for $\mathcal{D} = \mathcal{D}_{\text{Best}}$. It is important to note that either of these metrics considered *alone* can paint a misleading picture. For instance, a method can generate diverse candidates, but these candidates might be low scoring and similar to the known candidates. Thus, a method should be evaluated holistically, considering *all* the three metrics.

3. GFlowNets For Sequence Design

GFlowNets (Bengio et al., 2021a;b) tackle the problem of learning a stochastic policy π that can sequentially construct discrete objects $x \in \mathcal{X}$ with probability $\pi(x)$ using a non-negative reward function $R : \mathcal{X} \mapsto \mathbb{R}^+$ defined on the space $\mathcal{X}$, such that $\pi(x) \propto R(x)$. This property makes GFlowNets well-positioned to be used as a generator of diverse candidates in an active learning setting. In this section, we present our proposed active learning algorithm based on GFlowNets (Bengio et al., 2021a). We only present the relevant key results, and refer the reader to Bengio et al. (2021b) for a thorough mathematical treatment of GFlowNets. Figure 2 provides an overview of our proposed approach and Algorithm 1 describes the details of the approach.

3.1. Background

Preliminaries We assume the space $\mathcal{X}$ is *compositional*, that is, object $x \in \mathcal{X}$ can be constructed using a sequence of actions taken from a set $\mathcal{A}$. After each step, we may have a partially constructed object, which we call a state $s \in \mathcal{S}$. For example, Bengio et al. (2021a) use a GFlowNet to sequentially construct a molecule by inserting an atom or a molecule fragment in a partially constructed molecule represented by a graph. In the auto-regressive case of sequence generation, the actions could just be to append a token to a partially constructed sequence. A special action indicates that the object is complete, i.e., $s = x \in \mathcal{X}$. Each transition $s \to s' \in \mathcal{E}$ from state s to state s' corresponds to an edge in a graph $G = (\mathcal{S}, \mathcal{E})$ with the set of nodes

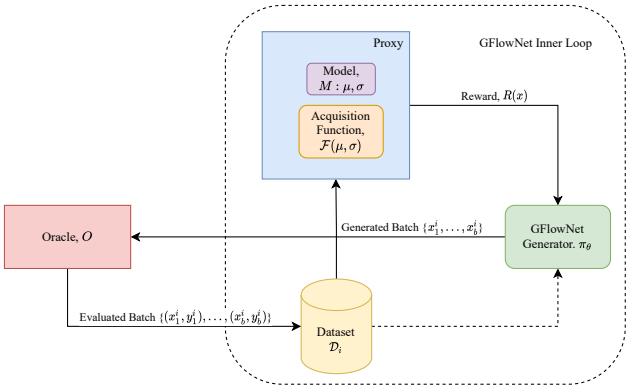

Figure 2. GFlowNet-AL: Our proposed approach for sequence design with GFlowNets consists of three main components: (1) GFlowNet Generator π_θ (green box), which generates diverse candidates with probability proportional to $R(x)$, which is defined by the proxy, (2) Proxy (blue) which consists of a model M that can output a mean prediction μ and uncertainty estimate σ around μ, along with an acquisition function $\mathcal{F}$, which combines the mean and uncertainty predicted by the model, and (3) Dataset $\mathcal{D}_i$ (yellow) which stores all the available candidates up to round i. In each round, the model M is first trained on $\mathcal{D}_i$. The generative policy is then trained with reward function $R = \mathcal{F}(M.\mu, M.\sigma)$ and data $\mathcal{D}_i$. A new batch of candidates $\mathcal{B}_i$ is then sampled from π_θ, evaluated with the Oracle $\mathcal{O}$ (red) and then added to $\mathcal{D}_i$ to obtain $\mathcal{D}_i$. This process repeats for N rounds of active learning.

Algorithm 1 Multi-Round Active Learning

Input:
O: Oracle to evaluate candidates x and return labels Y
$D_0 = \{(x_i, y_i)\}$: Initial dataset with $y_i = O(x_i)$
M: Trainable learner providing functions $M.\mu$ and $M.\sigma$, with $\mu(x)$ estimating $E[Y|x]$ and $\sigma(x)$ estimating the epistemic uncertainty around $\mu(x)$
π_θ: Generative policy trainable from a reward function R and from which candidates x can be sampled
$\mathcal{F}(\mu, \sigma)$: Acquisition function taking $M.\mu$ and $M.\sigma$ functions and returning a reward function R for training π_θ
K: Number of top-scoring candidates to keep for $TopK$ evaluation
b: Size of candidate batch to be generated
N: Number of active learning rounds (outer loop iterations)
Result: $TopK(D_N)$ elements $(x, y) \in D_n$ with highest values of y
Initialization: M,π_θ
for $i = 1$ *to* N **do**
- Fit M on dataset D_{i-1}
- Train π_θ with GFlowNet Inner Loop (Algorithm 2) using reward function $R = \mathcal{F}(M.\mu, M.\sigma)$
- Sample query batch $B = \{x_1, \ldots, x_b\}$ with $x_i \sim \pi_\theta$
- Evaluate batch B with O: $\hat{D}_i = \{(x_1, O(x_1)), \ldots, (x_b, O(x_b))\}$
- Update dataset $D_i = \hat{D}_i \cup D_{i-1}$

end

$\mathcal{S}$ and the set of edges $\mathcal{E}$. We require the graph to be directed and acyclic, meaning that actions are constructive and cannot be undone. An object $x \in \mathcal{X}$ is constructed by starting from an initial empty state s_0 and applying actions sequentially, and all complete trajectories must end in a special final state s_f. The fully constructed objects in $\mathcal{X} \subset \mathcal{S}$ are *terminating states*. The construction of an object x can thus be defined as a trajectory of states $\tau = (s_0 \to s_1 \to \ldots \to x \to s_f)$, and we can define $\mathcal{T}$ as the set of all trajectories. Parent$(s) = \{s' : s' \to s \in \mathcal{E}\}$ denotes the parents for node s and Child$(s) = \{s' : s \to s' \in \mathcal{E}\}$ denotes the children of node s in G.

Flows Bengio et al. (2021b) define a *trajectory flow* $F : \mathcal{T} \mapsto \mathbb{R}^+$. This trajectory flow $F(\tau)$ can be interpreted as the probability mass associated with trajectory τ. The *edge flow* can then be defined as $F(s \to s') = \sum_{s \to s' \in \tau} F(\tau)$, and *state flow* can be defined as $F(s) = \sum_{s \in \tau} F(\tau)$. The flow associated with the final step (transition) in the trajectory $F(x \to s_f)$ is called the terminal flow and the objective of training a GFlowNet is to make it approximately match a given reward function $R(x)$ on every possible x.

The trajectory flow F is a measure over complete trajectories $\tau \in \mathcal{T}$ and it induces a corresponding probability measure

$$P(\tau) = \frac{F(\tau)}{\sum_{\tau \in \mathcal{T}} F(\tau)} = \frac{F(\tau)}{Z}, \tag{4}$$

where Z denotes the total flow, and corresponds to the partition function of the the measure F. The forward transition probabilities P_F for each step of a trajectory can then be defined as

$$P_F(s|s') = \frac{F(s \to s')}{F(s)}. \tag{5}$$

We can also define the probability $P_F(s)$ of visiting a terminal state s as

$$P_F(s) = \frac{\sum_{\tau \in \mathcal{T}: s \in \tau} F(\tau)}{Z}. \tag{6}$$

Flow Matching Criterion A *consistent flow* satisfies the *flow consistency equation* $\forall s \in \mathcal{S}$ defined as follows:

$$\sum_{s' \in \text{Parent}(s)} F(s' \to s) = \sum_{s'' \in \text{Child}(s)} F(s \to s''). \tag{7}$$

It has been shown (Bengio et al., 2021a) that for a consistent flow F with the terminal flow set as the reward, i.e.,

$F(x \to s_f) = R(x)$, a policy π defined by the forward transition probability $\pi(s'|s) = P_F(s'|s)$ samples object x with probability proportional to $R(x)$

$$\pi(x) = \frac{R(x)}{Z}. \tag{8}$$

Learning GFlowNets GFlowNets learn to approximate an *edge flow* $F_\theta : \mathcal{E} \mapsto \mathbb{R}^+$ defined over G, such that the terminal flow is equal to the reward $R(x)$ and the flow is *consistent*. This is achieved by defining a loss function whose global minimum gives rise to the consistency condition. This was first formulated (Bengio et al., 2021a) via a temporal difference-like (Sutton & Barto, 2018) learning objective, called *flow-matching*:

$$\mathcal{L}_{FM}(s;\theta) = \left(\log \frac{\sum_{s' \in \text{Parent}(s)} F_\theta(s' \to s)}{\sum_{s'' \in \text{Child}(s)} F_\theta(s \to s'')} \right)^2. \tag{9}$$

Bengio et al. (2021a) show that given trajectories τ_i sampled from an exploratory training policy $\tilde{\pi}$ with full support, an edge flow learned by minimizing Equation 9 is consistent. At this point, the forward transition probability defined by this flow $P_{F_\theta}(s'|s) = \frac{F_\theta(s \to s')}{\sum_{s'' \in \text{Child}} F_\theta(s \to s'')}$ would sample objects x with a probability $P_F(x)$ proportionally to their reward $R(x)$.

In practice, the trajectories for training GFlowNets are sampled from an exploratory policy that is a mixture between the GFlowNet sampler P_{F_θ} and a uniform choice of action among those allowed in each state:

$$\bar{\pi}_\theta = (1-\delta) P_{F_\theta} + \delta \cdot \text{Uniform}. \tag{10}$$

This uniform policy introduces exploration preventing the training from getting stuck in one or a few modes. This is analogous to ϵ-greedy exploration in reinforcement learning.

Trajectory Balance Malkin et al. (2022) present an alternative objective defined over trajectories with faster credit assignment for learning GFlowNets, called *trajectory balance*, defined as follows:

$$\mathcal{L}_{TB}(\tau;\theta) = \left(\log \frac{Z_\theta \prod_{s \to s' \in \tau} P_{F_\theta}(s'|s)}{R(x)} \right)^2, \tag{11}$$

where $\log Z_\theta$ is also a learnable free parameter. This objective can improve learning speed due to more efficient credit assignment, as well as robustness to long trajectories and large vocabularies. Equation 11 is the training objective we have used in this paper.

Remarks When generating sequences in an autoregressive fashion (appending one token at a time), as in this paper, the mapping from trajectories to states becomes *bijective*, as there is only one path to reach a particular state s. The directed graph G then corresponds to a directed tree. Under these conditions, the flow-matching objective is equivalent to discrete-action Soft Q-Learning (Haarnoja et al., 2017; Buesing et al., 2019) with a temperature parameter $\alpha = 1$, a uniform $q_{\mathbf{a}'}$, and $\gamma = 1$, which obtains $\pi(x) \propto R(x)$. While the trajectory balance objective in (11) asymptotically reaches the same solution, our results (and that of Malkin et al., 2022) suggest it does so faster.

Algorithm 2 GFlowNet Inner Loop (with training data)

Input:
$D = \{x_i, y_i\}, i = 1, \ldots, N$: Dataset of candidates x_i with known oracle scores y_i
$R(\cdot)$: Reward function
γ: Proportion of offline data to use in training
m: GFlowNet training minibatch size
T: number of minibatch updates to complete training
δ: mixing coefficient for uniform actions in training policy
Result: $\pi_\theta = P_{F_\theta}$: learned policy with $\pi_\theta(x) \propto R(x)$
Initialization: F_θ: parameterized edge flow (neural net)
for $i = 1$ *to* T **do**
- Sample $m' = \lceil m(1-\gamma) \rceil$ trajectories from policy $\tilde{\pi} = (1-\delta) P_{F_\theta} + \delta$ Uniform
- Sample $m - m'$ trajectories from dataset $\mathcal{D}$
- Combine both sets of trajectories to form overall minibatch
- Compute reward $R(x)$ on terminal states x from each trajectory in the minibatch
- Update parameters θ with a stochastic gradient descent step wrt the objective in Eq. 9 or Eq. 11 for all trajectories in the minibatch.

end

3.2. Leveraging Data during Training

In our active learning setting, the reward function for the GFlowNet is obtained by training a model from a dataset $\mathcal{D} = \{(x, y)\}$ of labeled sequences with input object x and observed oracle reward y and we would like to make sure that the GFlowNet samples correctly in the vicinity of these x's (especially those for which y is larger). We can observe that the flow-matching objective (Equation 9) and the trajectory balance objective (Equation 11) are *off-policy* and *offline*. This allows us to use trajectories sampled from other policies than π during training, so long as the overall distribution of training trajectories $\tilde{\pi}$ has full support. These trajectories can be constructed from the x's in a given dataset by sampling for each of them a sequence of ancestors starting from terminal state x and sampling

a parent according to the backward transition probability. In the auto-regressive case studied here, there is only one possible parent for each state s, so we immediately recover the unique trajectory leading to x from s_0. This provides a set of offline trajectories.

Inspired by work in RL combining on-policy and off-policy updates (Nachum et al., 2017; Guo et al., 2021), we propose incorporating trajectories from the available dataset in the training of GFlowNets. At each training step we can augment the trajectories sampled from the current forward transition policy with trajectories constructed from examples in the dataset. Let $\gamma \in [0, 1)$ denote the proportion of offline trajectories in the GFlowNet training batch. As we vary γ from 0 to 1, we move from an online setting, originally presented in (Bengio et al., 2021a), to an offline setting where we learn exclusively from the dataset. Relying exclusively on trajectories from a dataset, however, can lead to sub-optimal solutions since the dataset is unlikely to cover $\mathcal{X}$. Algorithm 2 describes the proposed training procedure for GFlowNets which incorporates offline trajectories.

We hypothesize and verify experimentally in Section 5.4.1, that mixing an empirical distribution in the form of offline trajectories can provide the following potential benefits in the context of active learning: *(1) improved learning speed*: it can improve the speed of convergence since we make sure the GFlowNet approximation is good in the vicinity of the selected interesting examples from the dataset $\mathcal{D}$ *(2) lower bound on the exploration domain*: it guarantees exploration around the examples in $\mathcal{D}$.

3.3. Incorporating Epistemic Uncertainty

Another consequence of a reward function that is learned from a finite dataset $\mathcal{D} = \{(x, y)\}$ is that there will be increasing uncertainty in the model's predictions as we move away from its training x's. In the context of active learning, this uncertainty can be a strong signal to guide exploration in novel parts of the space and has been traditionally used in Bayesian optimization (Angermueller et al., 2019; Swersky et al., 2020; Jain et al., 2021). Bengio et al. (2021b) hypothesize that using information about the uncertainty of the reward function can also lead to more efficient exploration in GFlowNets. We study this hypothesis, by incorporating the model uncertainty of the reward function for training GFlowNets.

This requires two key ingredients: (a) the reward function should be a model that can provide an uncertainty estimate on its output, and (b) an acquisition function that can combine the prediction of the reward function with its uncertainty estimates to provide a scalar score. There has been significant work in developing methods that can estimate the uncertainty in neural networks, which we employ here. In our experiments, we rely on MC Dropout (Gal & Ghahramani, 2016) and ensembles (Lakshminarayanan et al., 2017) to provide epistemic uncertainty estimates. As for the acquisition function, we use Upper Confidence Bound (Srinivas et al., 2010) and Expected Improvement (Močkus, 1975). With the experiments of Section 5.4.2, we study the effects of these choices and observe the improvement provided by incorporating the uncertainty estimates.

4. Related Work

Biological sequence design has been approached with a wide variety of methods: reinforcement learning (Angermueller et al., 2019), Bayesian optimization (Wilson et al., 2017; Belanger et al., 2019; Moss et al., 2020; Pyzer-Knapp, 2018; Terayama et al., 2021), search/sampling using deep generative models (Brookes et al., 2019a; Kumar & Levine, 2020; Boitreaud et al., 2020; Das et al., 2021; Hoffman et al., 2021; Melnyk et al., 2021), deep model-based optimization (Trabucco et al., 2021a), adaptive evolutionary methods (Hansen, 2006; Swersky et al., 2020; Sinai et al., 2020), likelihood-free inference (Zhang et al., 2021), and black-box optimization with surrogate models (Dadkhahi et al., 2021). As suggested in Section 3, GFlowNets have the potential to improve over such methods by amortizing the cost of search (e.g., when comparing with MCMC's mixing time) over learning, giving probability mass to the entire space facilitating exploration and diversity (vs e.g., RL which tends to be greedier), enabling the use of imperfect data (vs e.g., generative models that require strictly positive or negative samples), and by scaling well with data by exploiting structure in function approximation (vs e.g., Bayesian methods that can cost $\mathcal{O}(n^3)$ for n datapoints).

5. Experiments

In this section we present experimental results across various biologically relevant sequence design tasks to demonstrate the effectiveness of our proposed GFlowNet-AL algorithm. We design our experiments to reflect realistic sequence design scenarios, varying several key parameters:

1. N: the number of active learning rounds - This can vary depending upon the particular application being considered, where the cost of evaluation in each round can limit the number of rounds available.
2. b: the size of candidate batch to be generated - The experimental setup in the evaluation phase can only be scaled to certain batch sizes, for instance, the synthesis of small molecules is mostly manual and cannot be parallelized much, whereas peptide synthesis can be scaled to 10^4 to 10^6 sequences at a time.
3. $|\mathcal{D}_0|$: the initial dataset available - Depending on the task at hand, one can have access to different numbers of initially available candidates.

4. $|x|$: the maximal length of constructed sequences - This can vary depending on the task at hand, for instance, design of anti-microbial peptides uses proteins of length 50 or shorter, whereas design of fluorescent proteins uses sequences of length > 200.
5. $|\mathcal{A}|$: the size of the action space (vocabulary) - Depending on the type of biological sequence being considered the vocabulary size can vary, for instance, from 4, in the case of DNA sequences, to 20 in the case of proteins.

5.1. Tasks and Evaluation Criteria

We present results on the following sequence design tasks. See Appendix A for further details on each of the tasks.

- **Anti-Microbial Peptide Design**: The goal is to generate peptides (short protein sequences) with anti-microbial properties. We consider sequences of length 50 or lower. The vocabulary size is 20 (amino-acids). We consider $N = 10$ rounds, with batch size $b = 1000$ and starting dataset $\mathcal{D}_0$ with 3219 AMPs and 4611 non-AMP sequences from the DBAASP database (Pirtskhalava et al., 2021). The choice of parameters was guided by the fact that AMPs can be efficiently synthesized and evaluated *in-vitro* in large quantities. Details in Appendix A.1

- **TF Bind 8**: We follow Trabucco et al. (2021b), where the goal is to search the space of DNA sequences (vocabulary size 4 nucleobases) of length 8 that have high binding activity with human transcription factors. Following the offline Model-Based Optimization setting from Trabucco et al. (2021b), we consider a single round setting $N = 1$, and generate $b = 128$ candidates starting with $|\mathcal{D}_0| = 32,898$ examples. The data and oracle are from (Barrera et al., 2016b). Details in Appendix A.2

- **GFP**: We use the design task as presented in Trabucco et al. (2021b). The goal is to search the space of protein sequences (vocabulary size 20) of length 237 and have high fluorescence. Following the offline Model-Based Optimization setting from Trabucco et al. (2021b), we consider a single round setting $N = 1$, and generate $b = 128$ candidates starting with $|\mathcal{D}_0| = 5,000$ examples. The data and oracle are from (Sarkisyan et al., 2016; Rao et al., 2019a). Details in Appendix A.3.

To evaluate the performance on these tasks, we follow the desiderata defined in Section 2. We evaluate the Performance, Diversity and Novelty Scores on the highest-scoring generated candidates, $\mathcal{D}_{\text{Best}}$. For the TF Bind 8 and GFP task we also present the 100th percentile and 50th percentile scores on the generated batch, following the evaluation scheme presented in Trabucco et al. (2021b) in Appendix C.

Table 1. Results on the AMP Task with $K = 100$.

	Performance	Diversity	Novelty
GFlowNet-AL	$\mathbf{0.932 \pm 0.002}$	$\mathbf{22.34 \pm 1.24}$	$\mathbf{28.44 \pm 1.32}$
DynaPPO	$\mathbf{0.938 \pm 0.009}$	12.12 ± 1.71	9.31 ± 0.69
COMs	0.761 ± 0.009	19.38 ± 0.14	26.47 ± 1.3
GFlowNet	0.868 ± 0.015	11.32 ± 0.67	15.72 ± 0.44

5.2. Baselines and Implementation

We consider as baselines a representative set of prior work focusing on ML for sequence design. We use the following methods as baselines: DynaPPO (Angermueller et al., 2019, Active Learning with RL as Generator), AmortizedBO (Swersky et al., 2020, Bayesian Optimization with RL-based Genetic Algorithm for optimizing acquisiton function), and COMs (Trabucco et al., 2021a, Deep Model Based Optimization). We also include a GFlowNet baseline with neither offline data nor uncertainty from the proxy. In all the experiments, the data is represented as a sequence of one-hot vectors $\{0, 1\}^{(|x| \times |\mathcal{A}|)}$, similar to the procedure followed by Trabucco et al. (2021b). For a fair comparison, we restrict all the baselines to use the same architecture (MLPs), however due to the large number of design choices in each of the baselines, there are some discrepancies. We provide complete implementation details in the Appendix B.1.

5.3. Results

5.3.1. Anti-Microbial Peptide Design

Table 1 shows the the results for the AMP design task. We observe that GFlowNet-AL generates significantly more diverse and novel sequences compared to the baselines, as well as better final TopK performance. Note that our experiments with AmortizedBO here were not conclusive, as it was designed for fixed-length sequences and generated nonsensical peptides (with almost exclusively W's and C's). See Appendix C.1 for examples of the sequences generated by AmortizedBO for this task. Another interesting observation here, in the setting of generating large batches, is that COMs, which relies on generating novel candidates by optimizing known candidates against a learned conservative model, performs quite poorly. This can be attributed the fact that it essentially performs a local search around known candidates, and this can be detrimental in cases where the goal is to generate large diverse and novel batches.

Physiochemical Properties In addition to the usefulness metric, to understand the biological relevance of the sequences generated by GFlowNet-AL we study several physiochemical properties of the Top 100 generated sequences using `BioPython` (Cock et al., 2009). The instability index for the generated peptides is 26.5 on averge with maximum of 36 (score of over 40 indicates instability). Figure 3 shows the distribution of AAs in the generated sequences

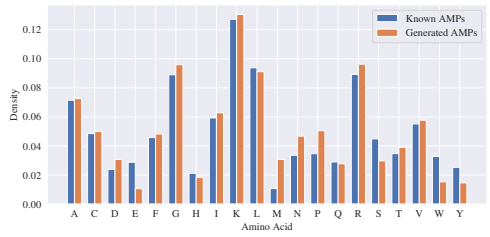

Figure 3. Distribution of occurrence of amino acids in the peptides generated with GFlowNet-AL closely matches that of known AMPs.

plotted against the set of known AMPs. We can observe that the distribution of amino acids in the generated sequences closely matches that of known AMPs.

Table 2. Results on TF-Bind-8 task with $K = 128$

	Performance	Diversity	Novelty
GFlowNet-AL	$\mathbf{0.84 \pm 0.05}$	4.53 ± 0.46	$\mathbf{2.12 \pm 0.04}$
DynaPPO	0.58 ± 0.02	5.18 ± 0.04	0.83 ± 0.03
COMs	0.74 ± 0.04	4.36 ± 0.24	1.16 ± 0.11
BO-qEI	0.44 ± 0.05	4.78 ± 0.17	0.62 ± 0.23
CbAS	0.45 ± 0.14	5.35 ± 0.16	0.46 ± 0.04
MINs	0.40 ± 0.14	$\mathbf{5.57 \pm 0.15}$	0.36 ± 0.00
CMA-ES	0.47 ± 0.12	4.89 ± 0.01	0.64 ± 0.21
AmortizedBO	0.62 ± 0.01	4.97 ± 0.06	1.00 ± 0.57
GFlowNet	0.72 ± 0.03	4.72 ± 0.13	1.14 ± 0.3

5.3.2. TF-BIND-8

The TF-Bind-8 task requires searching in the space of short DNA sequences for high binding activity with human transcription factors. The initial dataset $\mathcal{D}_0$ consists of the lower scoring half of all the possible DNA sequences of length 8. This setup allows us to evaluate the methods in the common setting, where only low quality data is available initially. For this task, we also include additional MBO baselines presented in Trabucco et al. (2021b). On this task, we see from Table 2 that GFlowNet-AL performs better than the other baselines in terms of TopK performance and novelty but that the MINs method performed best in terms of diversity. However, when we look at all the metrics together, MINs have a much lower performance score and novelty score indicating they generate sequences mostly from the training set. We also present results on the 100th and 50th percentile metrics proposed in Trabucco et al. (2021a), in the Appendix C.2, where GFlowNet still outperforms all the evaluated methods.

5.3.3. GFP

Finally we consider the GFP task, where the goal is to search in the space of proteins with for proteins that are highly fluorescent. Similar to TF-Bind-8, we include baselines from Trabucco et al. (2021a). GFlowNet-AL outperformed all the baselines across the board, i.e., in terms of TopK average scores, diversity and novelty of the generated candidates. Also interesting to note, MINs and CbAS, both methods relying on generative models like VAEs, seem to collapse on generating only sequences from the training dataset.

Table 3. Results on GFP task with $K = 128$

	Performance	Diversity	Novelty
GFlowNet-AL	$\mathbf{0.853 \pm 0.004}$	$\mathbf{211.51 \pm 0.73}$	$\mathbf{210.56 \pm 0.82}$
DynaPPO	0.794 ± 0.002	206.19 ± 0.19	203.20 ± 0.47
COMs	0.831 ± 0.003	204.14 ± 0.14	201.64 ± 0.42
BO-qEI	0.045 ± 0.003	139.89 ± 0.18	203.60 ± 0.06
CbAS	0.817 ± 0.012	5.42 ± 0.18	1.81 ± 0.16
MINs	0.761 ± 0.007	5.39 ± 0.00	2.42 ± 0.00
CMA-ES	0.063 ± 0.003	201.43 ± 0.12	203.82 ± 0.09
AmortizedBO	0.051 ± 0.001	205.32 ± 0.12	202.34 ± 0.25
GFlowNet	0.743 ± 0.05	200.72 ± 3.42	202.11 ± 1.54

5.4. Ablations

5.4.1. TRAINING WITH THE ORACLE DATA

We perform ablations to isolate the effect of including trajectories sampled from a static dataset in the GFlowNet training procedure, discussed in Algorithm 2. To do this, we sample a set of 4096 examples every 1000 training steps for the GFlowNet, and consider the average reward of the Top100 sequences in that set. Figure 4 shows the progression of the Top100 scores over the course of training of the GFlowNet in the first round of active learning in the AMP task, for different values of γ, which represents the fraction of sequences sampled from the dataset within a mini-batch. As we increase γ away from 0, the performance improves significantly compared to not having any offline data ($\gamma = 0$), until $\gamma = 0.50$ which worked best. Too many dataset examples, i.e., too few on-policy trajectories, leads to less exploration and less generalization outside of the training examples. Using offline data improves the speed at which GFlowNet training covers the support of the optimal π. Going beyond $\gamma = 0.5$, however, performance becomes significantly worse.

5.4.2. EFFECT OF UNCERTAINTY ESTIMATES

Next, we study the effect of incorporating information about the uncertainty in the *learned* reward function through multiple rounds of active learning. Here again, we consider the AMP Generation task, and consider three variations of the GFlowNet-AL algorithm with different models M in the proxy. We consider Deep Ensembles (Lakshminarayanan et al., 2017) and MC Dropout (Gal & Ghahramani, 2016) as two representative uncertainty estimation methods for neural networks and a third variation with a single model for the proxy, corresponding to the case of not having the uncertainty of the model incorporated in the reward. Note that Deep Ensembles generally provide more accurate un-

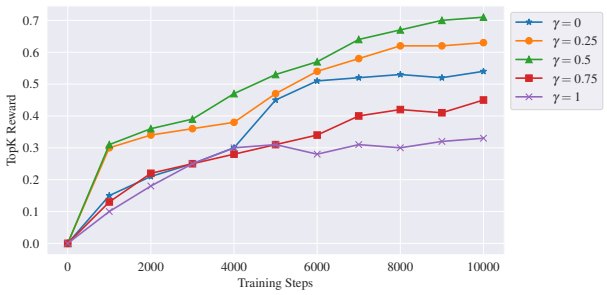

Figure 4. TopK ($K = 100$) scores over the training iterations for GFlowNet-AL in Round 1 of AMP Generation Task, with different values of γ, the proportion of trajectories sampled from the data. We can observe that $\gamma = 50\%$ is best.

certainty estimates than MC Dropout, so our evaluation also covers the effect of the quality of the uncertainty estimates. Table 4 shows the results for these ablations. We can observe that having any uncertainty estimate can provide an advantage over having none. In addition, we also observe that more accurate uncertainty estimates from Deep Ensembles lead to better results overall. We present additional ablations on the effect of the acquisition function in the Appendix C, but note that we do not see a significant difference based on the choice of the acquisition function.

Table 4. Results on the AMP Task with $K = 100$ for GFlowNet-AL with different methods for uncertainty estimation, with UCB as the acquisition function.

	Performance	Diversity	Novelty
GFlowNet-AL Ensemble	$\mathbf{0.932 \pm 0.002}$	$\mathbf{22.34 \pm 1.24}$	$\mathbf{28.44 \pm 1.32}$
GFlowNet-AL MC Dropout	0.921 ± 0.004	18.58 ± 1.78	19.58 ± 1.12
GFlowNet-AL None	0.909 ± 0.008	16.42 ± 0.74	17.24 ± 1.44

6. Conclusion and Future Work

Motivated by global health challenges such as antimicrobial resistance and the currently expensive and slow process of discovering new and useful biological sequences, we have introduced a generative active learning algorithm for sequences based on GFlowNets (as the candidate generator) and principles from Bayesian optimization (the estimation of epistemic uncertainty and the use of an acquisition function to score candidates), with the objective to produce diverse and novel sets of candidates. To achieve this, we discovered training GFlowNets could be greatly accelerated by using training sequences from the oracle (e.g., biological experiments) to construct additional training trajectories. We validated that both the use of epistemic uncertainty and the empirical distribution derived from the oracle outputs helped to obtain better results, in fact better than the state-of-the-art baselines we compared it to (based on genetic algorithms or RL), especially in terms of diversity and relative novelty of the generated candidates. A **limitation** of the proposed method and others that involve both a proxy model and a generative policy is that we now have two separate learners, each with their hyper-parameters. On the other hand, the use of efficient optimization or generation is necessary in high-dimensional search spaces. **Future work** should explore how we can make the retraining of the proxy model more efficient, considering that this is a continual learning setting. Better estimators of information gain, non-autoregressive generative models taking advantage of the underlying structure in the data, and an outer loop policy handling multiple oracles with a different fidelity are natural extensions of this work.

Software and Data: The code is available at https://github.com/MJ10/BioSeq-GFN-AL.

Acknowledgements

The authors would like to thank Dianbo Liu, Xu Ji, Kolya Malkin, Leo Feng, and members of the Drug Discovery and GFlowNet groups at Mila as well as anonymous reviewers for helpful discussions and feedback. The authors also acknowledge support from the AIHN IBM-MILA project. This research was enabled in part by compute resources provided by Compute Canada. The authors acknowledge funding from CIFAR, Samsung, IBM, Microsoft.

References

Aggarwal, C. C., Kong, X., Gu, Q., Han, J., and Philip, S. Y. Active learning: A survey. In *Data Classification: Algorithms and Applications*, pp. 571–605. CRC Press, 2014.

Angermueller, C., Dohan, D., Belanger, D., Deshpande, R., Murphy, K., and Colwell, L. Model-based reinforcement learning for biological sequence design. In *International Conference on Learning Representations*, 2019.

Audet, C. and Hare, W. *Derivative-Free and Blackbox Optimization*. Springer Series in Operations Research and Financial Engineering. Springer International Publishing, 2017. ISBN 9783319689135. URL https://books.google.ca/books?id=ejVBDwAAQBAJ.

Barrera, L. A., Vedenko, A., Kurland, J. V., Rogers, J. M., Gisselbrecht, S. S., Rossin, E. J., Woodard, J., Mariani, L., Kock, K. H., Inukai, S., Siggers, T., Shokri, L., Gordân, R., Sahni, N., Cotsapas, C., Hao, T., Yi, S., Kellis, M., Daly, M. J., Vidal, M., Hill, D. E., and Bulyk, M. L. Survey of variation in human transcription factors reveals prevalent dna binding changes. *Science*, 351(6280):1450–1454, 2016a. doi: 10.1126/

science.aad2257. URL https://www.science.org/doi/abs/10.1126/science.aad2257.

Barrera, L. A., Vedenko, A., Kurland, J. V., Rogers, J. M., Gisselbrecht, S. S., Rossin, E. J., Woodard, J., Mariani, L., Kock, K. H., Inukai, S., et al. Survey of variation in human transcription factors reveals prevalent dna binding changes. *Science*, 351(6280):1450–1454, 2016b.

Belanger, D., Vora, S., Mariet, Z., Deshpande, R., Dohan, D., Angermueller, C., Murphy, K., Chapelle, O., and Colwell, L. Biological sequences design using batched bayesian optimization, 2019.

Bengio, E., Jain, M., Korablyov, M., Precup, D., and Bengio, Y. Flow network based generative models for non-iterative diverse candidate generation. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021a. URL https://openreview.net/forum?id=Arn2E4IRjEB.

Bengio, Y., Deleu, T., Hu, E. J., Lahlou, S., Tiwari, M., and Bengio, E. Gflownet foundations, 2021b.

Boitreaud, J., Mallet, V., Oliver, C., and Waldispühl, J. Optimol: Optimization of binding affinities in chemical space for drug discovery. *Journal of Chemical Information and Modeling*, 60(12):5658–5666, 2020. doi: 10.1021/acs.jcim.0c00833. URL https://doi.org/10.1021/acs.jcim.0c00833. PMID: 32986426.

Brookes, D., Park, H., and Listgarten, J. Conditioning by adaptive sampling for robust design. In *International conference on machine learning*, pp. 773–782. PMLR, 2019a.

Brookes, D. H., Park, H., and Listgarten, J. Conditioning by adaptive sampling for robust design. In *ICML*, 2019b.

Buesing, L., Heess, N., and Weber, T. Approximate inference in discrete distributions with monte carlo tree search and value functions. *Artificial Intelligence and Statistics (AISTATS)*, 2019.

Cock, P. J. A., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J., Dalke, A., Friedberg, I., Hamelryck, T., Kauff, F., Wilczynski, B., and de Hoon, M. J. L. Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11):1422–1423, 03 2009. ISSN 1367-4803. doi: 10.1093/bioinformatics/btp163. URL https://doi.org/10.1093/bioinformatics/btp163.

Dadkhahi, H., Rios, J., Shanmugam, K., Das, Payel Melnyk, I., Das, P., Chenthamarakshan, V., and Lozano, A. Fourier representations for black-box optimization over categorical variables. *arXiv preprint arXiv:2111.06801*, 2021.

Das, P., Sercu, T., Wadhawan, K., Padhi, I., Gehrmann, S., Cipcigan, F., Chenthamarakshan, V., Strobelt, H., Dos Santos, C., Chen, P.-Y., et al. Accelerated antimicrobial discovery via deep generative models and molecular dynamics simulations. *Nature Biomedical Engineering*, 5(6):613–623, 2021.

Elnaggar, A., Heinzinger, M., Dallago, C., Rihawi, G., Wang, Y., Jones, L., Gibbs, T., Feher, T., Angerer, C., Steinegger, M., et al. Prottrans: towards cracking the language of life's code through self-supervised deep learning and high performance computing. *arXiv preprint arXiv:2007.06225*, 2020.

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pp. 1050–1059, 2016.

Garnett, R. *Bayesian Optimization*. Cambridge University Press, 2022. in preparation.

Guo, H., Tan, B., Liu, Z., Xing, E. P., and Hu, Z. Text generation with efficient (soft) q-learning, 2021.

Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. Reinforcement learning with deep energy-based policies. *International Conference on Machine Learning (ICML)*, 2017.

Hansen, N. The cma evolution strategy: a comparing review. *Towards a new evolutionary computation*, pp. 75–102, 2006.

Hoffman, S. C., Chenthamarakshan, V., Wadhawan, K., Chen, P.-Y., and Das, P. Optimizing molecules using efficient queries from property evaluations. *Nature Machine Intelligence*, pp. 1–11, 2021.

Jain, M., Lahlou, S., Nekoei, H., Butoi, V., Bertin, P., Rector-Brooks, J., Korablyov, M., and Bengio, Y. Deup: Direct epistemic uncertainty prediction, 2021.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization, 2017.

Kumar, A. and Levine, S. Model inversion networks for model-based optimization. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 5126–5137. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/373e4c5d8edfa8b74fd4b6791d0cf6dc-Paper.pdf.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 6405–6416, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Malkin, N., Jain, M., Bengio, E., Sun, C., and Bengio, Y. Trajectory balance: Improved credit assignment in gflownets, 2022.

Melnyk, I., Das, P., Chenthamarakshan, V., and Lozano, A. Benchmarking deep generative models for diverse antibody sequence design. *arXiv preprint arXiv:2111.06801*, 2021.

Močkus, J. On bayesian methods for seeking the extremum. In Marchuk, G. I. (ed.), *Optimization Techniques IFIP Technical Conference Novosibirsk, July 1–7, 1974*, pp. 400–404, Berlin, Heidelberg, 1975. Springer Berlin Heidelberg. ISBN 978-3-540-37497-8.

Moss, H. B., Leslie, D. S., Beck, D., Gonzalez, J., and Rayson, P. Boss: Bayesian optimization over string spaces. In *Advances in Neural Information Processing Systems*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/b19aa25ff58940d974234b48391b9549-Abstract.html.

Mullis, M. M., Rambo, I. M., Baker, B. J., and Reese, B. K. Diversity, ecology, and prevalence of antimicrobials in nature. *Frontiers in microbiology*, 10:2518, 2019.

Murray, C. J., Ikuta, K. S., Sharara, F., Swetschinski, L., Robles Aguilar, G., Gray, A., Han, C., Bisignano, C., Rao, P., Wool, E., Johnson, S. C., Browne, A. J., Chipeta, M. G., Fell, F., Hackett, S., Haines-Woodhouse, G., Kashef Hamadani, B. H., Kumaran, E. A. P., McManigal, B., Agarwal, R., Akech, S., Albertson, S., Amuasi, J., Andrews, J., Aravkin, A., Ashley, E., Bailey, F., Baker, S., Basnyat, B., Bekker, A., Bender, R., Bethou, A., Bielicki, J., Boonkasidecha, S., Bukosia, J., Carvalheiro, C., Castañeda-Orjuela, C., Chansamouth, V., Chaurasia, S., Chiurchiù, S., Chowdhury, F., Cook, A. J., Cooper, B., Cressey, T. R., Criollo-Mora, E., Cunningham, M., Darboe, S., Day, N. P. J., De Luca, M., Dokova, K., Dramowski, A., Dunachie, S. J., Eckmanns, T., Eibach, D., Emami, A., Feasey, N., Fisher-Pearson, N., Forrest, K., Garrett, D., Gastmeier, P., Giref, A. Z., Greer, R. C., Gupta, V., Haller, S., Haselbeck, A., Hay, S. I., Holm, M., Hopkins, S., Iregbu, K. C., Jacobs, J., Jarovsky, D., Javanmardi, F., Khorana, M., Kissoon, N., Kobeissi, E., Kostyanev, T., Krapp, F., Krumkamp, R., Kumar, A., Kyu, H. H., Lim, C., Limmathurotsakul, D., Loftus, M. J., Lunn, M., Ma, J., Mturi, N., Munera-Huertas, T., Musicha, P., Mussi-Pinhata, M. M., Nakamura, T., Nanavati, R., Nangia, S., Newton, P., Ngoun, C., Novotney, A., Nwakanma, D., Obiero, C. W., Olivas-Martinez, A., Olliaro, P., Ooko, E., Ortiz-Brizuela, E., Peleg, A. Y., Perrone, C., Plakkal, N., de Leon, A. P., Raad, M., Ramdin, T., Riddell, A., Roberts, T., Robotham, J. V., Roca, A., Rudd, K. E., Russell, N., Schnall, J., Scott, J. A. G., Shivamallappa, M., Sifuentes-Osornio, J., Steenkeste, N., Stewardson, A. J., Stoeva, T., Tasak, N., Thaiprakong, A., Thwaites, G., Turner, C., Turner, P., van Doorn, H. R., Velaphi, S., Vongpradith, A., Vu, H., Walsh, T., Waner, S., Wangrangsimakul, T., Wozniak, T., Zheng, P., Sartorius, B., Lopez, A. D., Stergachis, A., Moore, C., Dolecek, C., and Naghavi, M. Global burden of bacterial antimicrobial resistance in 2019: a systematic analysis. *The Lancet*, 2022. ISSN 0140-6736. doi: https://doi.org/10.1016/S0140-6736(21)02724-0. URL https://www.sciencedirect.com/science/article/pii/S0140673621027240.

Nachum, O., Norouzi, M., Xu, K., and Schuurmans, D. Bridging the gap between value and policy based reinforcement learning. *Advances in Neural Information Processing Systems*, 30:2775–2785, 2017.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Pirtskhalava, M., Amstrong, A. A., Grigolava, M., Chubinidze, M., Alimbarashvili, E., Vishnepolsky, B., Gabrielian, A., Rosenthal, A., Hurt, D. E., and Tartakovsky, M. Dbaasp v3: Database of antimicrobial/cytotoxic activity and structure of peptides as a resource for development of new therapeutics. *Nucleic Acids Research*, 49(D1):D288–D297, 2021.

Pyzer-Knapp, E. O. Bayesian optimization for accelerated drug discovery. *IBM Journal of Research and Development*, 62(6):2:1–2:7, 2018. doi: 10.1147/JRD.2018.2881731.

Rao, R., Bhattacharya, N., Thomas, N., Duan, Y., Chen, X., Canny, J., Abbeel, P., and Song, Y. S. Evaluating protein transfer learning with tape. In *Advances in Neural Information Processing Systems*, 2019a.

Rao, R., Bhattacharya, N., Thomas, N., Duan, Y., Chen, X., Canny, J. F., Abbeel, P., and Song, Y. S. Evaluating protein transfer learning with tape. *bioRxiv*, 2019b.

Rasmussen, C. and Williams, C. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning series. MIT Press, 2005. ISBN 9780262182539. URL https://books.google.ca/books?id=GhoSngEACAAJ.

Sarkisyan, K. S., Bolotin, D. A., Meer, M. V., Usmanova, D. R., Mishin, A. S., Sharonov, G. V., Ivankov, D. N., Bozhanova, N. G., Baranov, M. S., Soylemez, O., et al. Local fitness landscape of the green fluorescent protein. *Nature*, 533(7603):397–401, 2016.

Sinai, S., Wang, R., Whatley, A., Slocum, S., Locane, E., and Kelsic, E. Adalead: A simple and robust adaptive greedy search algorithm for sequence design. *arXiv preprint*, 2020.

Srinivas, N., Krause, A., Kakade, S., and Seeger, M. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pp. 1015–1022, Madison, WI, USA, 2010. Omnipress. ISBN 9781605589077.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Swersky, K., Rubanova, Y., Dohan, D., and Murphy, K. Amortized bayesian optimization over discrete spaces. In Peters, J. and Sontag, D. (eds.), *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 124 of *Proceedings of Machine Learning Research*, pp. 769–778. PMLR, 03–06 Aug 2020. URL https://proceedings.mlr.press/v124/swersky20a.html.

Terayama, K., Sumita, M., Tamura, R., and Tsuda, K. Black-box optimization for automated discovery. *Accounts of Chemical Research*, XXXX, 02 2021. doi: 10.1021/acs.accounts.0c00713.

Trabucco, B., Kumar, A., Geng, X., and Levine, S. Conservative objective models for effective offline model-based optimization. In *International Conference on Machine Learning*, pp. 10358–10368. PMLR, 2021a.

Trabucco, B., Kumar, A., Geng, X., and Levine, S. Design-bench: Benchmarks for data-driven offline model-based optimization, 2021b. URL https://openreview.net/forum?id=cQzf26aA3vM.

Wilson, J. T., Moriconi, R., Hutter, F., and Deisenroth, M. P. The reparameterization trick for acquisition functions, 2017.

Zhang, D., Fu, J., Bengio, Y., and Courville, A. C. Unifying likelihood-free inference with black-box sequence design and beyond. *ArXiv*, abs/2110.03372, 2021.

Zhang, D., Malkin, N., Liu, Z., Volokhova, A., Courville, A. C., and Bengio, Y. Generative flow networks for discrete probabilistic modeling. *ArXiv*, abs/2202.01361, 2022.

A. Task Details

A.1. Anti-Microbial Peptides

The peptides used in our experiments are obtained by filtering DBAASP (Pirtskhalava et al., 2021). We select peptides with sequence length between 12 and 60 as well as choosing unusual amino acid to the type of "without modification". The target group is the Gram-positive bacteria. In total we have 6438 positive AMPs, and 9522 non-AMPs.

We split the above mentioned dataset into two parts: D_1 and D_2. D_1 is available for use the algorithms, whereas, D_2 is used to train the oracle, f, following (Angermueller et al., 2019), as a simulation of wet-lab experiments for the generated sequences. Notice that every observation in the dataset has its corresponding group. The definition of being in the same group could be: having the same target or the same title or the same cluster. We follow a strict principle to split the dataset into D_1 and D_2: for any observation x in D_1, there are no observations in D_2 belong to x's group, and vice versa. Under this principle, the D_1 and D_2 are made either by cross-validation split or by train-valid split. Unlike (Angermueller et al., 2019), we use MLP classifiers (up to 89% test accuracy) to train the oracles based on features with the pre-trained protein language models from (Elnaggar et al., 2020), instead of Random Forests. Because the lengths of the sequences are not fixed, we set 60 as the maximum length of the sequences. We pad the sequences which do not reach the length of 60 by appending the end of sequence token.

A.2. TF-Bind-8

The dataset used for the TF-Bind-8 task contains 65792 samples, representing every possible size 8 string of nucleotides $\boldsymbol{x} \in \{0,1\}^{8\times 4}$. 50% of the initial dataset is set aside for model training, resulting in a training set of size 32898. As the dataset includes all possible size 8 DNA sequences, the oracle for this task is exact. The dataset for the TF-Bind-8 task is derived from (Barrera et al., 2016a) wherein DNA sequences are scored based on their binding activity to a human transcription factor SIX6 REF R1, where a higher binding energy is better. This task has been used to demonstrate MBO algorithm performance in recent papers (Angermueller et al., 2019; Trabucco et al., 2021a). We use the implementation of this dataset from the Design-Bench repository without any preprocessing (`https://github.com/brandontrabucco/design-bench`). The dataset's construction is described in more detail in (Trabucco et al., 2021a).

A.3. GFP

We again use the implementation in the Design-Bench repository for the GFP task's dataset and oracle (`https://github.com/brandontrabucco/design-bench`) (Trabucco et al., 2021b). The GFP task requires generation of proteins derivative of the bio-luminescent jellyfish *Aequorea victoria*'s green fluorescent protein (GFP) with maximum fluorescence. The dataset is of size 56086 with each sample being a protein of length 237. Each protein is encoded as a tensor of 237 sequential one-hot vectors, written as $\boldsymbol{x} \in \{0,1\}^{237\times 20}$. While the full dataset is of size 56086 only 5000 samples, drawn from between the 50th and 60th percentiles, are given as a training set to the optimization algorithms. The oracle used is 12-layer transformer provided by the TAPE framework (Rao et al., 2019b). Before running any optimization algorithm, we normalize the fluorescence scores given in the training set and produced by the oracle. Finally, in reporting final scores we re-normalize with respect to the full GFP dataset's minimum and maximum. More details on the setup are provided in Trabucco et al. (2021a).

B. Implementation Details

B.1. Baselines

In our implementations of the baseline algorithms, we made use of previously published implementations, making small adaptations where necessary. In particular, for AmmortizedBO we used their published implementation and for DynaPPO we adapted and used a version implemented in the repository published by the FLEXS project (Sinai et al., 2020).

Table 5. Hyperparameters used for AmortizedBO. We varied the number of mutations allowed, K, based on the length of the sequence to be generated, L. We also varied the acquisition function used, as well as the number of generations G allowed between proposals.

	L	K	**Acquisition Function**	G
AMP	50	40	UCB	40
TF-Bind-8	8	4	UCB	5
GFP	237	200	UCB	5

AmortizedBO: Following (Swersky et al., 2020) we kept nearly all hyperparameters constant across all tasks for AmmortizedBO, only varying the hyperparameters listed in Table 5. Those hyperparameters were selected after a grid search for which the same options were provided for each task. For all other hyperparameters and architectures we use the default settings in the published AmortizedBO implementation. For the AMP task we required that AmortizedBO may output dynamically sized strings. To implement this, we allowed AmortizedBO to output a stop token at any position which would cause that position to the maximum length of the string to be padded. This setting was only used for the AMP task.

Table 6. Hyperparameters used for DynaPPO. We varied the number of trajectories generated between proposals $\mathcal{T}$, the policy network learning rate γ, the DynaPPO exploration penalty scale factor λ, and the exploration penalty's radius ϵ.

	$\mathcal{T}$	γ	λ	ϵ
AMP	1000	0.0001	0.2	8
TF-Bind-8	20000	0.0001	0.1	2
GFP	2000	0.0001	0.1	20

DynaPPO: Although we used the FLEXS library as our base for DynaPPO, we altered the implementation slightly in order to better represent the algorithm specified in (Angermueller et al., 2019). In particular, we added dynamic hyperparameter tuning of the proxy model after each query to the oracle, the exploration penalty term proposed by DynaPPO, and a method to allow DynaPPO to output variably sized strings (for the AMP task). We reused the architecture specified in the FLEXS library for the policy network and, as DynaPPO requires a hyperparameter search across its proxy model after each query to the oracle, the hyperparameter options for the proxy laid out in (Angermueller et al., 2019) in our implementation. We ran a grid search over various settings of the number of trajectories generated between proposals $\mathcal{T}$, the policy network learning rate γ, the exploration penalty scale parameter λ, and the exploration penalty radius ϵ. We used the best performing hyperparameters for each task, as reported in Table 6. In DynaPPO's proxy ensemble, a constituent model is only included in the ensemble if its R^2 score is higher than a threshold given 5-fold cross validation on the dataset. Following their recommendation, we require a model's R^2 on the dataset to be at least 0.5 for it to be included in the ensemble. Finally, we note that Angermueller et al. (2019) propose including a Gaussian Process in their proxy ensemble. However, we found the Gaussian Process to be intractable on the datasets for our experimental tasks, and as such excluded it from the proxy ensemble.

COMs: For COMs we the same architecture for the forward model as used in Trabucco et al. (2021a), which is an feedforward neural network with two hidden layers of size 2048 and a LeakyRELU activation function with leak 0.3, trained with an Adam Optimizer and learning rate 10^-3. The rest of the parameters are set to the best hyperparameters reported in Trabucco et al. (2021a) for all tasks as follows: number of gradient ascent steps in the solver=50, number of steps to generate adversarial $\mu(x)$ = 50, learning rate $\alpha = 0.01$, $\tau = 2.0$, $\eta = 2\sqrt{d}$ and number of epochs to train $\hat{f}_\theta$ = 50, as well as the same pre-processing procedure.

Other Baselines: For the additional baselines reported on the GFP and TF-Bind-8 tasks: MINs (Kumar & Levine, 2020), CbAS (Brookes et al., 2019b), BO-qEI (Wilson et al., 2017) and CMA-ES (Hansen, 2006), we use the the implementations provided in Trabucco et al. (2021b), and reproduce the results with the reported hyperparameters.

B.2. GFlowNet

We implement the proposed GFlowNet-AL algorithm in PyTorch (Paszke et al., 2019).

Proxy: We use MLP with 2 hidden layers of dimension 2048 and ReLU activation, as the base architecture for the proxy in our experiments in all three tasks. In the case of ensembles, we use 5 members with the same architecture, whereas in the case of MC Dropout we use 25 samples with dropout rate 0.1, and weight decay of 0.0001. We use a minibatch

size of 256 for training with a MSE loss, using the Adam optimizer (Kingma & Ba, 2017), with learning rate 10^-4 and $(\beta_0, \beta_1) = (0.9, 0.999)$. We use early stopping, keeping 10% of the data as a validation set. For UCB ($\mu + \kappa\sigma$) we use $\kappa = 0.1$.

GFlowNet Generator: We parameterize the flow as a MLP with 2 hidden layers of dimension 2048, and $\mathcal{A}$ outputs corresponding to each action. We use the trajectory balance objective for training in all our experiments. For training we use the Adam optimizer with $(\beta_0, \beta_1) = (0.9, 0.999)$. Table 7 shows the rest of the hyperparameters. In addition to that we set γ, the proportion of offline trajectories to 0.5 for all three tasks. The learning rate for $\log Z$ is set to 10^{-3} for all the experiments. In each round we sample $t * K$ candidates, and pick the top K based on the proxy score, where t is set to 5 for all experiments.

Table 7. Hyperparameters for the GFlowNet Generator

Hyperparameter	**AMP**	**TF-Bind-8**	**GFP**
δ: Uniform Policy Coefficient	0.001	0.001	0.05
Learning rate	5×10^{-4}	10^{-5}	5×10^{-4}
m: Minibatch size	32	32	32
β: Reward Exponent $R(x)^\beta$	3	3	3
T: Training steps	10,000	5,000	20,000

For the results in Table 1, Table 2 and Table 3, we use GFlowNet-AL with ensembles as the proxy model with UCB as the acquisiton function.

C. Additional Results

C.1. AMP Generation: Additional Results

As discussed in Section 5.3.1, after running AmortizedBO on the AMP task the algorithm generated sequences which were overwhelmingly poor in regards to real-world usefulness. As AmortizedBO was initially proposed as an algorithm to produce fixed length strings, we implemented a dynamic length AmortizedBO for which we added an extra stop token to the generator's vocabulary. When AmortizedBO selected to insert a stop token at position i,the selected position as well as all positions succeeding the selected position would be set to a padding token. All top 1000 sequences generated by AmortizedBO were of the maximum allowed sequence length for the AMP task. Some sequences are: "RRRRWWRHHHHHICCWIWKCWWWIIICWWWWWCWWWWWIIWWWIIWCWWL", "RRRWRWWHHHWIICCHCIKCCLWIIIIWWWWWWCWWWWIWWWWIICWWWL", and "RRRWRWICHHRRCCCRIIWCCLWIIICWWWWWCWWWWIIWWWWIWCWWWL". These sequences do not look natural, and lack important characteristics generally found in AMPs (for example the amino acid "K", which is dominant in peptides with anti-microbial activity).

C.2. TF-Bind-8 and GFP: Additional Results

In Table 8 and Table 9 we present the 100th and 50th percentile results on the GFP and TF-Bind-8 tasks respectively as proposed in Trabucco et al. (2021b). We observe that GFlowNets outperform the baselines even under these metrics.

Table 8. Maximum and median scores of the proposed sequences for the GFP task.

	100th Percentile	**50th Percentile**
GFlowNet-AL	$\mathbf{0.871 \pm 0.006}$	$\mathbf{0.853 \pm 0.002}$
DynaPPO	0.790 ± 0.003	0.790 ± 0.005
COMs	0.864 ± 0.000	0.864 ± 0.000
BO-qEI	0.254 ± 0.352	0.246 ± 0.341
CbAS	0.865 ± 0.000	0.852 ± 0.004
MINs	0.865 ± 0.001	0.820 ± 0.018
CMA-ES	0.054 ± 0.002	0.047 ± 0.000
AmortizedBO	0.058 ± 0.002	0.052 ± 0.001

Table 9. Maximum and median score of the proposed sequences for the TF-Bind-8 task.

	100th Percentile	50th Percentile
GFlowNet-AL	$\mathbf{0.989 \pm 0.009}$	$\mathbf{0.784 \pm 0.015}$
DynaPPO	0.942 ± 0.025	0.562 ± 0.025
COMs	0.945 ± 0.033	0.497 ± 0.038
BO-qEI	0.798 ± 0.083	0.439 ± 0.000
CbAS	0.927 ± 0.051	0.428 ± 0.010
MINs	0.905 ± 0.052	0.421 ± 0.015
CMA-ES	0.953 ± 0.022	0.537 ± 0.014
AmortizedBO	0.989 ± 0.014	0.636 ± 0.025

C.3. Effect of Uncertainty: Additional Results

We present additional results on the three tasks with different choices of acquisition functions and uncertainty estimation methods. Note that for GFlowNet-AL-None, the choice of acquisition function does not matter, so the results are put only on the UCB section. We observe that the key factor affecting performance is consistently the uncertainty model.

Table 10. Results with GFlowNet-AL-None, where the uncertainty from the proxy is not used.

	Performance	Diversity	Novelty
AMP	0.909 ± 0.008	16.42 ± 0.74	17.24 ± 1.44
TF-Bind-8	0.81 ± 0.04	3.96 ± 0.32	1.73 ± 0.18
GFP	0.786 ± 0.001	205.28 ± 1.68	207.65 ± 1.19

Table 11. Results on AMP Generation Task with UCB and EI as acquisition functions and different methods for uncertainty estimation.

	UCB			EI		
	Performance	**Diversity**	**Novelty**	**Performance**	**Diversity**	**Novelty**
GFlowNet-AL-Ensemble	$\mathbf{0.932 \pm 0.002}$	22.34 ± 1.24	$\mathbf{28.44 \pm 1.32}$	0.928 ± 0.002	$\mathbf{23.61 \pm 1.05}$	26.52 ± 1.56
GFlowNet-AL-MCDropout	0.921 ± 0.004	18.58 ± 1.78	19.58 ± 1.12	0.917 ± 0.002	17.38 ± 0.64	18.34 ± 1.42

Table 12. Results on TF-Bind-8 Task with UCB and EI as acquisition functions and different methods for uncertainty estimation.

	UCB			EI		
	Performance	**Diversity**	**Novelty**	**Performance**	**Diversity**	**Novelty**
GFlowNet-AL-Ensemble	$\mathbf{0.84 \pm 0.05}$	$\mathbf{4.53 \pm 0.46}$	$\mathbf{2.12 \pm 0.04}$	0.84 ± 0.01	4.46 ± 0.58	2.02 ± 0.13
GFlowNet-AL-MCDropout	0.81 ± 0.03	3.89 ± 0.85	1.76 ± 0.15	0.81 ± 0.02	4.10 ± 0.43	1.92 ± 0.16

Table 13. Results on GFP Task with UCB and EI as acquisition functions and different methods for uncertainty estimation.

	UCB			EI		
	Performance	**Diversity**	**Novelty**	**Performance**	**Diversity**	**Novelty**
GFlowNet-AL-Ensemble	$\mathbf{0.853 \pm 0.004}$	211.51 ± 0.73	$\mathbf{210.56 \pm 0.82}$	0.851 ± 0.003	$\mathbf{212.03 \pm 0.64}$	208.31 ± 0.94
GFlowNet-AL-MCDropout	0.825 ± 0.007	204.76 ± 1.75	200.93 ± 0.46	0.838 ± 0.001	207.42 ± 1.24	208.31 ± 1.60