МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ КАФЕДРА
БИОМЕДИЦИНСКОЙ ИНФОРМАТИКИ

# Изучение подходов обучения с подкреплением для генерации потенциальных лекарств

Курсовой проект

Благодарного Артёма Андреевича
обучающегося 3 курса
специальности «Информатика»

Научный руководитель:
профессор, доктор физико-
математических наук,
Тузиков А.В.

Минск, 2024

# Содержание

# Введение

Разработка лекарств – одна из самых значимых и востребованных задач мирового здравоохранения. Благодаря современным компьютерным технологиям, многие ранее неизлечимые заболевания теперь можно эффективно исследовать, анализировать и создавать новые потенциальные лекарства с доступными временными и финансовыми ограничениями.

Применение алгоритмов глубокого обучения значительно повышает эффективность исследований, а также позволяет создавать новые соединения, которые отсутствуют в существующих химических базах данных. Это даёт стимул к развитию науки, синтезированию новых низкомолекулярных соединений, которые могут быть более эффективными, чем существующие лекарства. Но создание новой нейронной сети для дизайна потенциальных лекарств связана с проблемой необходимости работы с большими объемами обучающих данных, что приводит к тому, что существует возможность работать лишь с ограниченным числом белковых мишеней, так как не для всех белковых мишеней есть достаточное количество данных для обучения: структурные данные белков, данные о взаимодействиях молекул, данные о химической активности молекул и многих других. В связи с этим обучение с подкреплением становится перспективным подходом к созданию новых лекарственных средств [4].

Разработка потенциальных лекарств — это создание новых химических объектов, которые удовлетворяют определенным ограничениям. Разработка нового лекарственного средства — это итеративная задача оптимизации, в основе которой лежит идея поиска локальных оптимумов молекулярных структур, но при этом она не гарантирует попадание в глобальный оптимум [3]. Поэтому ищут набор локальных оптимумов, то есть структурно отличающихся молекул с высокой вероятностью воздействия на желаемую мишень. Для разработки новых лекарственных средств были разработаны многочисленные методы, основанные на глубоком обучении, включая подходы, основанные на обучении с подкреплением [5-7] и вариационные автоэнкодеры [8-9]. Вариационные автоэнкодеры — это тип нейронных сетей, используемых для обучения генеративных моделей, которые могут создавать новые данные, схожие с обучающим набором.

Эти подходы используют несколько различных способов кодирования молекул во что-то, что может быть изучено моделью, например, кодирование на основе отпечатков пальцев, строк и графов. Основанная на строках упрощенная система линейного ввода молекул (SMILES) [10] - популярный способ кодирования двумерной структуры молекул. Недавние исследования эффективности методов молекулярной генерации de novo показали хорошую производительность при

использовании обучения с подкреплением (RL) для обучения рекуррентной нейронной сети (RNN) [11] для генерации строк SMILES [12, 13]. Цель состоит в том, чтобы получить стратегию, которая может использовать последовательности токенов для генерации строк SMILES.

В основе этой работы лежит REINVENT 4 – платформа для разработки новых лекарств с использованием механизма воспроизведения, основанного на вознаграждении. В этой работе представлено исследование различных алгоритмов обучения с подкреплением, основанных на стратегиях, в сочетании с несколькими способами воспроизведения предыдущих эпизодов или ограничения обучения подмножеством эпизодов, выбранных в текущем эпизоде.

# Глава 1. REINVENT 4

Молекулярный дизайн можно представить как задачу обратного проектирования. При прямом проектировании мы модифицируем существующие соединения до тех пор, пока они не будут удовлетворять нашим критериям, в то время как обратный дизайн сначала определяет свойства, которыми должна обладать молекула, и, таким образом, информирует алгоритм о том, как создавать молекулы. Молекулы лекарственных средств, в частности, должны соответствовать строгим требованиям к свойствам, прежде чем они будут одобрены в качестве безопасных и эффективных лекарственных средств, включая соответствующие физико-химические свойства, химическую стабильность и т. д. Также очень важна возможность синтезировать полученное соединение, которая открывает новые перспективы в разработке эффективных методов синтеза и способствует ускорению создания инновационных соединений с заданными свойствами. Это подчеркивает сложность разработки успешного лекарственного средства и требования к алгоритмам для ее решения. Обратная задача проектирования – это попытка объединить требуемое количество свойств с обширным химическим пространством. Предпринимались различные усилия для прогнозирования успешности соединений на клинических этапах, с целью выявления «оптимальной» комбинации их молекулярных свойств [15].

Молекулярный дизайн следует рассматривать как часть цикла DMTA (*Design*, *Make*, *Tests*, *Analyze* – проектирование, изготовление, испытания, анализ). Генеративные модели, являющиеся ключевым инструментом этапа *Design*, помогают создавать молекулы с заданными свойствами, используя алгоритмы машинного обучения. На следующих этапах — *Make*, *Tests* и *Analyze* — роботизированные системы выполняют задачи по синтезу, тестированию и анализу молекул, минимизируя человеческое вмешательство. Совместное использование генеративных моделей и роботизированных систем ускоряет весь цикл разработки, приближая нас к полностью автоматизированным процессам с замкнутым циклом [16]. Для достижения такой автоматизации важны два ключевых аспекта: принятие решений и возможность синтеза.

В этой статье описывается прогресс в разработке REINVENT как основы для молекулярного генеративного искусственного интеллекта. REINVENT находится в стадии разработки и постоянно поддерживается в рабочем состоянии. REINVENT используется для обратного проектирования с помощью алгоритмов обучения с подкреплением [17, 18], используя RNNS и трансформеры в качестве архитектур глубокого обучения, основанных на строках SMILES в качестве молекулярного представления. В данной работе рассматривается новая версия 4, в которой особое внимание уделяется новым функциям, таким как:

- поэтапное обучение с подкреплением (Reinforcement Learning) вместе с обучением по учебной программе (Curriculum Learning)
- новые модели-трансформеры для оптимизации молекул (molecule optimization)
- полная интеграция всех генераторов со всеми алгоритмами обучения: обучение с переносом знаний (Transfer Learning), RL, CL
- переработанная подсистема подсчета очков с использованием механизма плагинов для удобства расширения и настройки формата файла TOML (формат конфигурационных файлов, предназначенный для упрощения чтения и написания как людьми, так и программами) в дополнение к JSON (несовместимому с предыдущими версиями).

REINVENT 4 – это современное программное решение для молекулярного проектирования. Кодовая база была в значительной степени переписана, и все программное обеспечение и модели доступны в едином хранилище. Описания оригинальной версии REINVENT 1 и версии 2.0 были опубликованы в других источниках [17, 18]. Код версии 3 был выпущен в виде программного обеспечения с открытым исходным кодом, но без сопроводительной программы. Было показано, что REINVENT превосходит многие другие методы молекулярной оптимизации с точки зрения эффективности использования образцов [12], но также успешно предлагает реалистичные трехмерные молекулы, как показано в недавнем тесте docking benchmark для генерирующих моделей, превосходящем многие методы, основанные на графах [19].

# Глава 2. Обучение с подкреплением

Цель обучения с подкреплением заключается в разработке алгоритма, который обучается через метод проб и ошибок. Вместо использования обучающей выборки, такой алгоритм взаимодействует с окружающей средой (environment), где роль «разметки» выполняет награда (reward) — скалярная величина, отражающая успех алгоритма в выполнении задачи. Награда выдается после каждого шага взаимодействия со средой и позволяет алгоритму оценивать, насколько эффективно он справляется с поставленной задачей [1]. Но при этом:

- Награда не указывает, как именно нужно решать задачу или какие действия необходимо предпринять.
- Она может быть отложенной во времени или крайне редкой (в большинстве случаев агент получает значение награды, равное нулю).
- Награда является определённым «сигналом» для обучения (например, хорошо/плохо), чего нет в обучении без учителя.

## 2.1. Постановка задачи

Теперь формализуем всю эту концепцию и введём терминологию. Задача обучения с подкреплением задаётся **Марковским Процессом Принятия Решений** (**Markov Decision Process** или сокращённо **MDP**) это четвёрка ($\mathcal{S}, \mathcal{A}, \mathcal{P}, r$), где:

- $\mathcal{S}$ — пространство состояний (state space), множество состояний, в которых в каждый момент времени может находиться среда.
- $\mathcal{A}$ — пространство действий (action space), множество вариантов, из которых нужно производить выбор на каждом шаге своего взаимодействия со средой.
- $\mathcal{P}$ — **функция переходов** (transition function), которая задаёт изменение среды после того, как в состоянии $s \in \mathrm{S}$ было выбрано действие $a \in \mathrm{A}$. В общем случае функция переходов может быть стохастична, и тогда такая функция переходов моделируется распределением $p\ (s' \mid s,\ a)$: с какой вероятностью в какое состояние перейдёт среда после выбора действия $a$ в состоянии $s$.
- $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ — **функция награды** (reward function), выдающая скалярную величину за выбор действия $a$ в состоянии $s$. Это своего рода «обучающий сигнал» [1].

В обучении с подкреплением субъект, взаимодействующий с окружающей средой и влияющий на неё, называется **агентом** (**agent**). Агент действует на основе **стратегии** (**policy,** также в литературе встречается как политика), которая определяет правило выбора действий в зависимости от текущего состояния среды. Стратегия может быть стохастической и описывается распределением

вероятностей $\pi(a \mid s)$. По сути, стратегия — это функция, которую мы стремимся оптимизировать [1].

Взаимодействие агента со средой при заданной стратегии $\pi(a \mid s)$ происходит следующим образом:

1. Среда изначально находится в состоянии $s_0$.
2. Агент выбирает действие $a_0$, которое сэмплируется из стратегии $a_0 \sim \pi(a_0 \mid s_0)$
3. Среда реагирует на действие, переходя в новое состояние: $s_1 \sim p(s_1 \mid s_0, a_0)$. Также среда возвращает агенту награду $r(s_0, a_0)$.

Этот процесс повторяется:

- Агент снова выбирает следующее действие $a_1$
- Среда обновляет состояние до $s_2$ и выдаёт награду $r(s_1, a_1)$

Цикл продолжается либо до бесконечности, либо до достижения терминального состояния. Терминальное состояние завершает взаимодействие, после чего агент больше не получает награды. Если в среде есть терминальные состояния, полный процесс от начального состояния до терминального называется **эпизодом** (**episode**). Цепочка генерируемых в ходе взаимодействия случайных величин $s_0, a_0, s_1, a_1, s_2, a_2, \ldots$ называется **траекторией** (**trajectory**) [1].

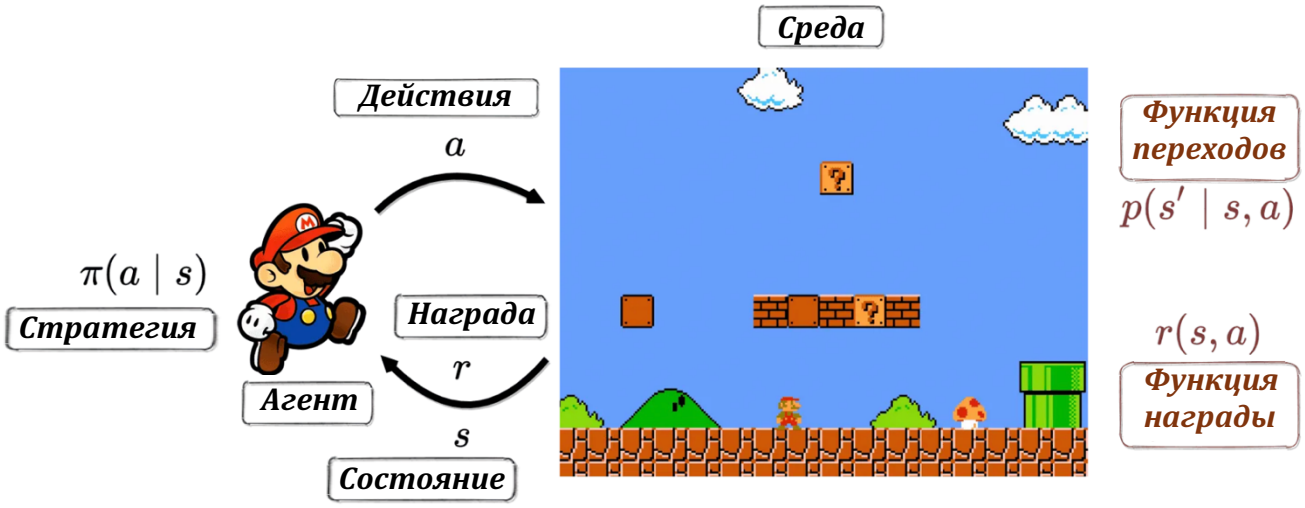


*Рис. 1*. Общая схема обучения

На рисунке 1 изображена общая схема обучения, в которой среда находится в состоянии $s$, агент, используя стратегию $\pi$, выполняет действие $a$, на это среда реагирует и с помощью функции переходов обновляет своё состояние $s'$ и, в зависимости от выбранного действия $a$ и состояния среды $s$, среды агент получает награду $r$ [20].

Таким образом, наша среда представляет собой управляемую марковскую цепь: на каждом шаге мы выбираем действие $a$, которое определяет распределение

для генерации следующего состояния. При этом предполагается, что среда обладает марковским свойством, то есть переход в следующее состояние зависит только от текущего состояния и не зависит от всей предыдущей истории:

$$p(s_{t+1} \mid s_t, a_t, s_{t-1}, a_{t-1}, \dots, s_0, a_0) = p(s_{t+1} \mid s_t, a_t). \quad (1)$$

Также предполагается стационарность: функция переходов $p$ ($s' \mid s$, $a$) не зависит от времени, то есть она остаётся неизменной, независимо от того, сколько шагов прошло с начала взаимодействия. Эти допущения достаточно реалистичны: законы мира остаются постоянными (стационарность), а состояние описывает мир целиком (марковость) [1]. Однако в этой модели есть одно нереалистичное предположение — **полная наблюдаемость** (**full observability**), согласно которому агент в своей стратегии $\pi$ ($a \mid s$) наблюдает всё состояние $s$ и может выбирать действия, зная всё о внешнем мире. В реальности же есть лишь частичные наблюдения состояния [1].

## 2.2. Цель

В итоге получилось смоделировать среду, агента и их взаимодействие на математическом языке. Теперь определим цель. Во время взаимодействия агент получает награду $r_t = r(s_t, a_t)$ на каждом шаге. Однако состояния и действия $s_t, a_t$ в этой модели являются случайными величинами, поэтому один и тот же агент может получать очень разную суммарную награду $\sum_{t\geq 0} r_t$ из-за случайности выбора действия в стратегии, так и случайности нового состояния среды из-за функции переходов. В итоге цель научиться выбирать действия так, чтобы в среднем получать как можно больше награды, а это означает максимизировать **математическое ожидание награды** с учетом всех возможных случайных исходов [1]. Каждая стратегия $\pi$ определяет распределение в пространстве траекторий — с какой вероятностью нам может встретиться траектория $\tau = (s_0, a_0, s_1, a_1, \dots)$:

$$p(\tau \mid \pi) = p(s_0, a_0, s_1, a_1, \dots \mid \pi) = \prod_{t\geq 0} p(s_{t+1} \mid s_t, a_t)\, \pi(a_t|(s_t)\,. \quad (2)$$

Вот по такому распределению будем брать математическое ожидание:

$$\mathbb{E}_{\tau\sim\pi} \sum_{t\geq 0} r_t \to max_\pi \quad (3)$$

Добавим ещё одну корректировку. Чтобы избежать парадоксов, возникающих в средах с бесконечным временем взаимодействия, введём концепцию **дисконтирования** награды [20]. Без него агент может стремиться к стратегиям, где он бесконечно долго набирает награду (например, получать +1 на каждом втором шаге будет так же хорошо, как на каждом сотом). Дисконтирование решает эту проблему, утверждая: награда, полученная сейчас, ценнее, чем такая же награда в

будущем. Для этого каждую будущую награду будем уменьшать с помощью коэффициента $\gamma$, который меньше единицы. Тогда наш функционал примет такой вид:

$$\mathbb{E}_{\tau\sim\pi}\sum_{t\geq 0}\gamma^t r_t \rightarrow max_\pi \tag{4}$$

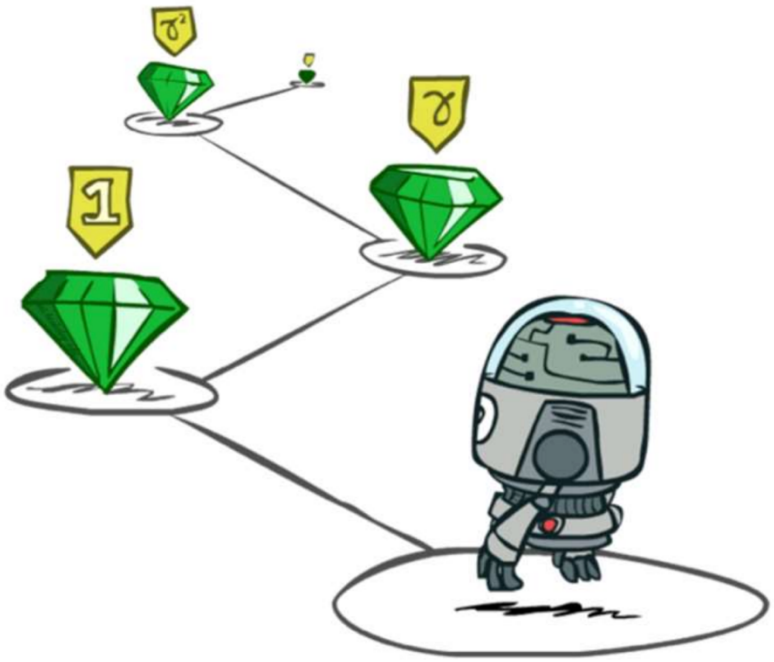


*Рис. 2*. Концепция дисконтирования награды

На рисунке 2 показана идея концепции дисконтирования награды, которая награждает агента больше, если он раньше достигнет нужного результата.

Обучение с подкреплением по своей сути представляет собой задачу оптимизации, направленную на улучшение функционалов определённого типа. В отличие от классического машинного обучения, где выбор функции потерь часто является частью инженерного подхода к решению задачи, в обучении с подкреплением функция награды уже задана и определяет функционал, который необходимо оптимизировать.

## 2.3. Решение

Задачу обучения с подкреплением можно сформулировать как задачу **динамического программирования**, основанную на максимизации средней дисконтированной кумулятивной награды. В её основе лежит структура, заданная Марковским процессом принятия решений (MDP), где взаимодействие агента со средой описывается следующими шагами: состоянии $s$, агент хочет выбрать действие $a$ как можно оптимальнее, за это он получит награду $r = r(s, a)$, среда

обновит своё состояние $s'$ и дальше получаем подзадачу эквивалентной структуры. Когда агент принимает решение на следующем шаге, на прошлое своё решение повлиять он уже не может; стационарность означает, что законы, по которым ведёт себя среда, не поменялись, а марковость говорит, что история не влияет на дальнейший процесс нашего взаимодействия [1]. Это приводит к пониманию того, что задача максимизации награды из текущего состояния $s$ напрямую связана с максимизацией награды из следующего состояния $s'$, независимо от того, каким оно ни было.

Введём вспомогательную величину – **оценочную функцию**. Оценочные функции используются для измерения качества действий агента и его стратегии при взаимодействии со средой. Эти функции позволяют определить, насколько хорошо агент справляется с поставленной задачей. Рассмотрим оптимальную Q-функцию и будем её обозначать $Q^*(s,\ a)$. Пусть $Q^*(s,\ a)$ — это максимальная награда в среднем после выбора действия $a$ из состояния $s$. Исходя из определения $Q^*$, чтобы посчитать значение $Q^*(s,\ a)$, после выбора действия $a$ в состоянии $s$ нужно перебрать все стратегии, посмотреть, сколько каждая из них набирает награды, и взять наилучшую стратегию. Поэтому эта оценочная функция называется оптимальной: она предполагает, что в будущем после выбора действия $a$ из состояния $s$ агент будет вести себя оптимально [1].

Хотя такая функция не всегда может быть вычислена на практике, ведь количество различных комбинаций может достигать огромного количества, что делает перебор всех возможных стратегий практически невозможным. Но она имеет важное свойство: если каким-то образом удалось узнать значения $Q^*(s,\ a)$, то оптимальная стратегия становится очевидной. Эта идея лежит в основе принципа оптимальности Беллмана, который гласит: *жадный выбор по отношению к оптимальной Q-функции оптимален* [1]:

$$\pi^*(s)\ = argmax_a Q^*(s,a). \qquad (5)$$

*Примечание:* если Q-функция достигает максимума на нескольких действиях, то можно выбирать любое из них.

Примечательно, что оптимальная стратегия является детерминированной. Это означает, что нет необходимости искать стохастическую стратегию. Достаточно сосредоточиться на нахождении $Q^*(s,\ a)$, а затем выводить из неё оптимальную стратегию, просто выбирая действия жадно.

Нахождение $Q^*(s,\ a)$ связано с тем фактом, что её можно выразить через саму себя. Когда агент выбирает действие $a$ в состоянии среды $s$, то он сразу получает награду $r$ $(s, a)$, вся дальнейшая награда будет дисконтирована на $\gamma$. После этого среда переходит в новое состояние $s'$, которое определяется согласно

распределению $s' \sim p\ (s' \mid s, a)$ (на результат этого сэмплирования агент уже никак повлиять не может и по этой случайности нашу будущую награду надо будет усреднять), а в этом новом состоянии $s'$, если агент выбирает действия оптимально, то он выберет действие $a'$, которое даёт максимум награды $Q^*(s',\ a'\ )$ [1].

Таким образом, $Q^*(s,\ a)$ можно вычислить через рекурсивное соотношение, называемое уравнением оптимальности Беллмана для Q-функции:

$$Q^*(s,a)\ = r\ (s,a) + \ \gamma \mathbb{E}_{s' \sim p\ (s' \mid s,a)}\, max_{a'}\, Q^*(s',a'). \qquad (6)$$

В итоге образовалась система уравнений, где значения $Q^*(s,\ a)$ зависят от самих себя. Это нелинейная система, но она обладает важным свойством: у неё есть единственное решение. Это значит, что решение этой системы можно считать альтернативным определением функции $Q^*(s,\ a)$, и его можно найти с помощью **метода простой итераций**. Метод простой итераций позволяет постепенно улучшать текущее приближение решения уравнения вида $x = f(x)$ [21]. Для этого мы начинаем с произвольного начального приближения $Q^*(s,a)$ : S × A → R, которое будет служить стартовой точкой. Затем, на каждом шаге, мы подставляем текущее приближение в правую часть уравнений оптимальности Беллмана, рассчитываем новое значение и используем его для обновления приближения:

$$Q^*_{k+1}(s,a) \leftarrow r(s,a) + \gamma \mathbb{E}_{s' \sim p\left(s' \mid s,\, a\right)} max_{a'} Q^*_k(s',a'). \qquad (7)$$

# Глава 3. RL для генерации потенциальных лекарств

Создание потенциальных лекарств с использованием рекуррентных нейронных сетей (RNN) для оптимизации молекул, закодированных в виде строк SMILES, можно сформулировать как задачу обучения с подкреплением. В данной постановке агент взаимодействует с окружением на конечных временных интервалах, добавляя токены к строке SMILES.

Генерация новых молекул происходит по эпизодам, то есть эпизод начинается заданием начальным токеном новой потенциальной молекулы, которая представляется строкой SMILES, и заканчивается, когда модель добавила терминальный токен к строке, поэтому длина эпизода зависит от длины генерируемой строки SMILES. На шаге $t = 0$ к строке добавляется начальный токен, который определяет первое состояние $s_1$. На каждом следующем шаге $t = 1, \ldots, T$ агент анализируется $n$-мерный вектор состояния среды $s_t \in \mathcal{S} \subseteq \mathbb{R}^{n_s}$ и выбирает действие $a_t$ в соответствии со стратегией $\pi\,(a_t \mid s_t)$. Эпизод заканчивается на шаге T+1 с терминальным состоянием $s_{T+1}$, когда в качестве действия $a_T$ выбирается конечный токен [3].

Кроме того, в конце эпизода агент получает вознаграждения $\mathcal{R}\,(a_{1:T}) \in [0, 1]$, который зависит от последовательности действий $a_{1:T}$. Вознаграждения генерируется функцией награды, которая оценивает каждую допустимую строку SMILES.

Вектор состояния $s_t$ формируется на основе выходного состояния RNN на шаге $t - 1$ и кодирует информацию о действиях, выполненных на предыдущих шагах. Рассматривается дискретное пространство действий $\mathcal{A} = \{0, \ldots, 33\}$ где каждый элемент соответствует допустимому символу строки SMILES (атомы, связи, циклы и т. д.), включая начальный и конечный токены. Существуют различные методы оптимизации стратегии, целью которых является обучение стратегии, параметризованной по $\theta$, $\pi_\theta\,(a_t \mid s_t)$. Выходные значения RNN подаются на полносвязный слой (FC layer), который либо вычисляет вероятности (с использованием softmax [28]), либо оценивает значения для каждого действия [3].

## 3.1. Конвейер обучения

Первый шаг в использовании алгоритмов обучения с подкреплением (RL) заключается в подготовке изначальной стратегии, которая может включать предварительное обучение или кодирование определённых структур в стратегию. Затем с помощью этой стратегии генерируется пакет молекул, например, путём выбора последовательности символов в формате SMILES.

На следующем этапе сгенерированные молекулы оцениваются с использованием неизвестного «чёрного ящика» – целевой функции. Это означает,

что целевая функция может быть вычислена в любой точке своей области определения, но её полное аналитическое выражение неизвестно. Молекулы вместе с их оценками сохраняются для последующего анализа, а также, при необходимости, для повторного использования (replay).

Полученные молекулы и их оценки передаются в алгоритм обучения с подкреплением, где стратегия генерации молекул обновляется. В зависимости от использования реплей буфера (replay buffer), для шага обучения могут использоваться текущие и/или предыдущие сгенерированные молекулы. С обновлённой стратегией генерируется новая пачка молекул. Этот процесс повторяется до выполнения заданного критерия остановки, например достижения предопределённого числа сгенерированных молекул.

## 3.2 Алгоритмы оптимизация стратегии

В REINVENT 4 используются алгоритмы оптимизации стратегии для генерации новых лекарств такие как: Регуляризованная оценка максимального правдоподобия (Regularized maximum likelihood estimation (Reg. MLE)); Алгоритм Актор-Критик с преимуществом (Advantage Actor-Critic (A2C)); Проксимальная оптимизация стратегии (Proximal Policy Optimization (PPO)); Актор-Критик с реплей буфером (Actor-Critic with Experience Replay (ACER)); Мягкий Актор-Критик (Soft Actor-Critic (SAC)). Эти алгоритмы обеспечивают генерацию разнообразных молекул с высокими оценками качества [3].

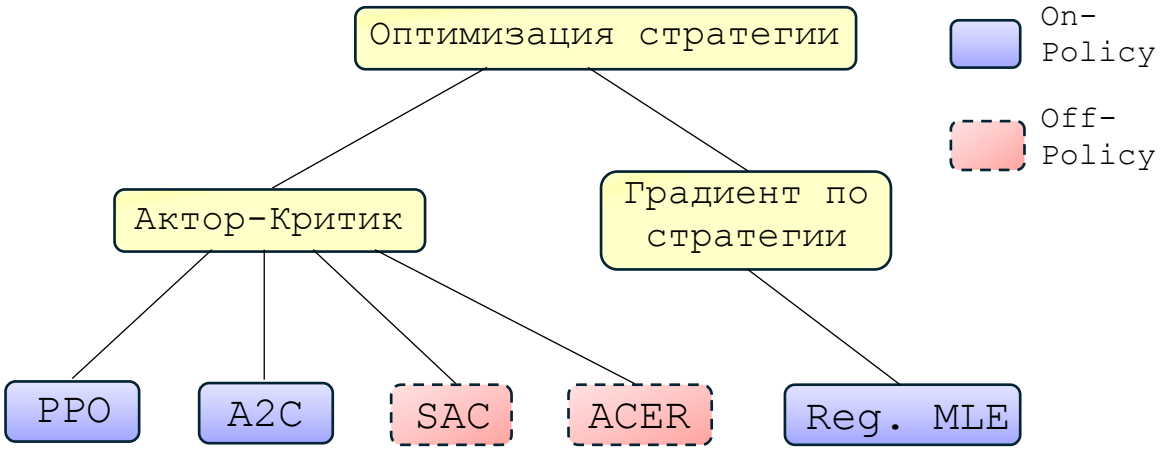


*Рис. 3* Алгоритмы оптимизации стратегии

Рисунок 3 показывает таксономию этих алгоритмов оптимизации стратегии. В области обучения с подкреплением, различие между **on-policy** и **off-policy** алгоритмами заключается в том, как используется опыт для обновления стратегии агента.

*On-policy* алгоритмы требуют, чтобы данные для очередного шага обновления весов были сгенерированы именно текущей, самой свежей версией стратегии агента. Иными словами, алгоритм использует только тот опыт, который был получен с помощью самой текущей стратегии.

*Off-policy* алгоритмы не ограничиваются использованием только данных, полученных от текущей стратегии. Они могут использовать опыт, полученный как от текущей, так и от предыдущих стратегий, независимо от того, насколько эффективными они были. В **off-policy** алгоритмах важно не то, какая стратегия сгенерировала данные, а то, что данные используются для вычисления оптимальности, например, через уравнение Беллмана. Это позволяет обновлять модель, используя произвольный опыт, что, в свою очередь, даёт возможность использовать такие методы, как реплей буфер, где опыт может быть собран в любой момент и использован позже для обучения.

## 3.3 Регуляризованной оценки максимального правдоподобия

Алгоритм регуляризованной оценки максимального правдоподобия (Reg. MLE) в настоящее время используется в REINVENT [18]. Недавние исследования показали хорошую эффективность по сравнению как с подходами, основанными на RL, так и RL, для разработки новых лекарственных средств [12, 13]. Идея заключается в том, что новая стратегия агента должна быть близка к изначальной стратегии, при этом основное внимание должно уделяться последовательностям с высокими показателями. Это идея лежит в основе следующей функции потерь:

$$\mathcal{L}^{Reg.\ MLE}(\theta) = \left(log\ \pi_{prior}(a_{1:T}) + \sigma R(a_{1:T}) - log\ \pi_{\theta}(a_{1:T})\right)^2, \qquad (8)$$

где $\pi_{prior}(a_{1:T})$ – правдоподобие предобученной стратегии и $\pi_{\theta}(a_{1:T})$ – правдоподобие текущей стратегии, которую оптимизируем.

## 3.4 Проксимальная оптимизация стратегии

Впервые проксимальную оптимизацию стратегии (PPO) ввёл Schulman и другие в 2017 году [22]. В REINVENT адаптировали метод PPO к задаче, где векторы состояний представлены выходами рекуррентной нейронной сети (RNN) на последнем рекуррентном узле. Актор представлен нейронной сетью с параметрами $\theta$, которая использует слой softmax для предоставления вероятностей $\pi_{\theta}(a \mid s)$ выбора действия $a$ в состоянии $s$. Начальная настройка актора основана на предобученной стратегии, а параметры оптимизируются с использованием функции потерь

$$\mathcal{L}^{CLIP}(\theta) = \mathbb{E}_t[min(r(\theta)A'(s_t),\ clip(r_t(\theta), 1 - \varepsilon, 1 + \varepsilon)A'(s_t))]^2, \qquad (9)$$

где $r(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{old}(a_t|s_t)}-$ отношение вероятностей, преимущество $A'(s_t) = \gamma^{T-t}R(a_{1:T}) - V_\phi(s_t)$, $V_\phi(s_t)$ – функция ценности, которая моделируется отдельной нейронной сетью с параметрами $\phi$. Архитектура сети идентична актору, за исключением выходного слоя, который состоит из одного узла (оценка ценности) и не использует softmax. Сеть функции ценности обучается путем минимизации среднеквадратичной ошибки

$$\mathcal{L}^{MSE}(\phi) = \mathbb{E}_t\left[\frac{1}{2}\left(\gamma^{T-t}R(a_{1:T}) - V_\phi(s_t)\right)\right]^2. \quad (10)$$

## 3.5 Актор-Критик с преимуществом

Актор-Критик с преимуществом (Advantage Actor-Critic (A2C)) — это синхронная (обновления параметров нейронных сетей (актора и критика) происходят одновременно) версия алгоритма A3C [23]. Для вычисления преимущества (advantage) используется следующая формула:

$$A'(s_t) = \gamma^{T-t}R(a_{1:T}) - V_\phi(s_t), \quad (11)$$

где $V_\phi(s_t)$ — это функция ценности с параметрами $\phi$, а $\gamma$ — коэффициент дисконтирования.

В работе алгоритм A2C адаптирован для случая, когда состояния представлены выходами рекуррентной нейронной сети. Актор (actor) реализован в виде нейронной сети с параметрами, которая вычисляет вероятности действий для каждого состояния с использованием слоя softmax.

Инициализация актора производится с использованием предварительно обученной политики, а параметры обновляются методом градиента по стратегии (policy gradient) с использованием преимущества. Для обеспечения предпочтения меньших по размеру молекул используется коэффициент дисконтирования, меньший единицы [3].

Архитектура функции ценности идентична архитектуре актора, за исключением выходного слоя, который состоит из одного узла (оценка ценности) и не использует softmax. Сеть функции ценности обучается путем минимизации среднеквадратичной ошибки формулы (10). Если актор генерирует большое количество недопустимых строк SMILES, то он сбрасывается [3].

## 3.6 Актор-Критик с реплей буфером

Актор-Критик с реплей буфером (Actor-Critic with Experience Replay (ACER)) [24] представляет собой алгоритм обучения с подкреплением типа Actor-Critic с

использованием механизма повторного воспроизведения опыта (experience replay). Данный алгоритм является off-policy вариантом алгоритма A3C [23], и его основная цель заключается в стабилизации оценок, выполняемых вне текущей стратегии, например методом оптимизации стратегии с ограничением доверительной области (Trust Region Policy Optimization, TRPO) [25].

Алгоритм ACER выполняет два типа обновлений параметров модели для эффективного обучения. Первое — обновление на основе текущей стратегии (on-policy), которое использует данные, собранные агентом в реальном времени, и позволяет напрямую улучшать стратегию. Второе — обновления вне текущей стратегии (off-policy), которые работают с ранее сохранёнными данными из памяти воспроизведения (replay buffer). Количество таких обновлений определяется случайной величиной $r \sim P(\lambda)$ (распределению Пуассона).

В REINVENT ACER адаптирован к задаче, где вектора состояний представлены выходными данными рекуррентного узла последнего слоя RNN. Используется общая сеть, имеющая ту же архитектуру, что и предварительно обученная сеть стратегии, но с добавлением дополнительного выходного слоя (value head). Этот полносвязный слой (fully connected) с размерностью выхода, равной 1. Параметры value head инициализируются случайным образом, в то время как остальные веса, включая веса слоя стратегии (policy head), заимствуются из предварительно обученной сети.

## 3.7 Мягкий Актор-Критик

Мягкий Актор-Критик (Soft Actor-Critic (SAC)) [26] представляет собой off-policy алгоритм, который включает энтропию стратегии в сигнал вознаграждения. Основой метода является максимизация энтропии, где оптимальная стратегия $\pi_\theta(a \mid s)$ стремится одновременно максимизировать вознаграждение и энтропию в каждом посещаемом состоянии.

Для контроля параметра температуры $\alpha$, который определяет относительную важность энтропийного члена, используется автоматическая настройка энтропии. Изначально SAC разработан для непрерывного пространства действий, но Christodoulou (2019) адаптировал его для дискретных действий. В REINVENT применяется адаптированный SAC для дискретных действий с изменениями, описанными ниже.

Адаптация SAC заключается в представлении векторов состояния через выходные данные рекуррентного узла нейронной сети. Актор модели реализован в виде нейронной сети с параметрами $\theta$, которая с помощью слоя softmax определяет вероятности $\pi_\theta(a \mid s)$ для выбора действия $a$ в состоянии $s$. Актор инициализируется заранее обученной стратегией, а параметры обновляются через

минимизацию функции потерь [3]. Функция ценности $Q_\phi(a, s)$ задаётся другой нейронной сетью с аналогичной архитектурой, но без слоя softmax, где каждый выход соответствует значению действия-состояния для действия $a$ в состоянии $s$. Параметры сети ценности $\phi$ инициализируются параметрами заранее обученной стратегии.

На первых $K_{init} = 10$ эпизодах обновление параметров не производится, а собранный опыт используется только для заполнения памяти воспроизведения (replay buffer). Поскольку в экспериментах максимальное количество эпизодов 2000, то на обучение остаётся 1990 эпизодов. Для обновления параметров используется вознаграждение $-1$ за некорректные SMILES-строки, что способствует генерации валидных молекул. В каждом эпизоде выполняется одно on-policy обновление с использованием текущих последовательностей, после чего проводятся четыре off-policy обновления, каждое из которых использует выборку из 64 последовательностей из памяти воспроизведения [3].

# Глава 4. De novo дизайн с REINVENT

В данном разделе описывается настройка платформы для проведения экспериментов, включая использование реплей буферов для исследования алгоритмов оптимизации стратегии. На каждом эпизоде из выборки отбирается пакет из M=128 последовательностей, в результате чего генерируется 128 строк SMILES (дубликаты строк SMILES удаляются).

В конце каждого эпизода выполняется обновление стратегии на основе полного разворачивания каждой последовательности (см. главу 3).

## 4.1 Молекулярные и топологические скелеты

Скелет молекулы определяет её основная структуру и свойства. Он представляет собой общую структуру, характерную для определённой группы молекул. Этот подход помогает исследовать различные молекулярные структуры и строительные блоки, что позволяет находить молекулы с похожими свойствами, но с разной структурой. Это важно, потому что дает возможность выбрать несколько альтернативных структур при разработке новых лекарств. Другими словами, скелеты молекул (их базовые структуры) служат индикатором разнообразия молекул, которые обладают схожей активностью, что помогает найти оптимальные кандидаты на лекарства.

В REINVENT для генерации скелетов используется молекулярный скелет, определённый Бемисом и Мёрко в 1996, также известный как скелет Бемиса-Мёрко [27]. Топологический скелет — это упрощённая версия молекулы, где все атомы заменяются на атомы углерода, а все химические связи преобразуются в одинарные. Такой скелет помогает сосредоточиться только на основной структуре молекулы, игнорируя детали, связанные с типами атомов и связей.

## 4.2 Генерация

Для генерации новых строк SMILES в REINVENT используется предварительно обученная стратегия, предоставленную Blaschke и др. [18]. Модель была обучена на наборе данных, основанном на базе данных ChEMBL, и генерирует молекулы в формате строк SMILES. Параметры сети обновляются с использованием методов обучения с подкреплением. Архитектура сети стратегии включает следующие слои: слой встраивания (embedding layer), который использует 256-мерные векторы; LSTM-слой, принимающий на вход вектор размерностью 256 и возвращающий вектор размерностью 512, состоящий из трёх последовательных рекуррентных слоёв LSTM; и полносвязный выходной слой, размерность которого равна 34 (так как всего 34 возможных действия). Каждому выходному значению соответствует

токен из словаря, включающего начальный и конечный токены, как это описано в работе Blaschke и др. [18].

При генерировании последовательности выходной слой пропускается через функцию softmax, чтобы вычислить вероятности для каждого токена (действия). На основе этих вероятностей осуществляется мультимодальная выборка для определения следующего действия в последовательности. Поскольку целевыми объектами являются малые молекулы, длина последовательности ограничена 256 символами. Когда последовательность достигает этой длины, процесс выборки завершается, и возвращается уже сформированная последовательность.

## 4.3 Оценивание

Функция оценки $R(a_{1:T})$ назначает награду для каждой последовательности. Она реализована с помощью модели случайного леса, которая состоит из 1300 деревьев, каждое из которых имеет максимальную глубину 300. Для выборки с возвращением используется вес классов, который обратно пропорционален частотам классов, что определяет вероятность выборки образца.

Вероятности классов бинарной активности вычисляются как доля деревьев, которые предсказывают соответствующий класс. Вознаграждение для последовательности определяется как вероятность того, что модель предскажет положительный класс для данной последовательности.

Строка SMILES считается допустимой, если соответствующая последовательность может быть преобразована в объект Mol с помощью RDKit, что происходит при расчёте отпечатков для оценки. Недопустимой строке SMILES присваивается вознаграждение 0 для on-policy алгоритмов и −1 для off-policy алгоритмов

## 4.4 Фильтры разнообразия

Фильтр разнообразия — это метод, основанный на использовании памяти, который помогает улучшить разнообразие генерируемых молекул. Он отслеживает молекулы с похожими структурами, например, молекулярные скелеты. В REINVENT применяется фильтр разнообразия, основанный на идентичных молекулярных скелетах, предложенный Blaschke и др. [18].

Этот фильтр включает в себя память о скелетах молекул, которая хранит молекулы и их соответствующие молекулярные скелеты. Молекула добавляется в память скелетов, если её соответствующая последовательность имеет вознаграждение не менее 0.4. Молекулы с одинаковыми строками SMILES не могут быть добавлены в память, таким образом сохраняются только уникальные молекулы.

Если количество молекул с одинаковым молекулярным скелетом достигает 25, то все последующие молекулы с этим же скелетом получают вознаграждение 0 и не сохраняются в память. Это изменение функции вознаграждения влияет на процесс обучения, ограничивая генерирование избыточных молекул с одинаковыми скелетами.

## 4.5 Реплей буферы

Реплей буферы (replay buffers) используются для обучения моделей, комбинируя данные текущих и предыдущих последовательностей, что способствует улучшению производительности. В REINVENT доступны следующие типы буферов: *All Current (AC),* который использует только текущий пакет последовательностей для обновления модели; *Bin History (BH),* который разделяет последовательности по вознаграждению и сохраняет данные по принципу очереди; *Bin Current (BC),* аналогичный BH, но работающий только с текущим пакетом; *Top-Bottom History (TBH),* который хранит последовательности с наибольшим и наименьшим вознаграждением из предыдущих эпизодов, а также учитывает текущие данные; *Top-Bottom Current (TBC),* отбирающий последовательности с максимальным и минимальным вознаграждением из текущего пакета; *Top History (TH),* сохраняющий лучшие последовательности из предыдущих эпизодов; *Top Current (TC),* который использует последовательности с наивысшим вознаграждением из текущего пакета. Эти буферы позволяют эффективно обновлять модель, повышая её производительность, используя разнообразные подходы к обучению.

# Глава 5. Эксперимент

В данной главе рассматривается процесс запуска эксперимента с помощью платформы REINVENT 4, предназначенной для генерации молекул с использованием алгоритмов обучения с подкреплением. Целью эксперимента является генерация молекул с заданными свойствами на определённую молекулярную мишень.

## 5.1 Подготовка к эксперименту

Перед запуском эксперимента необходимо выполнить несколько шагов по подготовке системы и настройке конфигурации. Это включает установку всех зависимостей, настройку окружения, а также создание конфигурационного файла для эксперимента.

### 5.2.1 Установка и настройка окружения

Для корректной работы REINVENT 4 требуется Python 3 версии 3.10 или выше. Для изоляции зависимостей рекомендуется использовать виртуальное окружение - **Conda**. Это также поможет избежать конфликтов версий и облегчить установку необходимых пакетов. Для создания и активации виртуального окружения использовался код из рисунка 4 Приложения А.

### 5.2.2 Создание конфигурации эксперимента

Для успешного проведения эксперимента необходимо настроить конфигурационный файл, который описывает параметры задачи. REINVENT использует формат TOML и JSON для конфигурации. В конфигурационном файле указываются ключевые параметры, такие как модель и алгоритм обучения, устройство, температура и т. д. Конфигурацию можно дополнительно адаптировать в зависимости от конкретной задачи. Примеры таких файлов можно посмотреть по ссылке: https://github.com/MolecularAI/REINVENT4/tree/main/configs.

## 5.3 Запуск эксперимента

После настройки всех необходимых параметров можно приступить непосредственно к запуску эксперимента. В REINVENT 4 используется командная строка для управления процессом. Запуск эксперимента осуществляется с помощью команды из рисунка 5 Приложения А. Для вывода промежуточных отчётов работы эксперимента можно пропустить параметр -l. В конце эксперимента полученный лог-файл предоставляет информацию о каждом шаге эксперимента, включая количество сгенерированных молекул и значения вознаграждений для каждой итерации.

## 5.4 Оценка и анализ результатов

После завершения эксперимента можно приступить к анализу полученных молекул, оценив их качество с помощью различных методов. REINVENT позволяет использовать **scoring plugins** для оценки качества молекул по заданным параметрам. Для этого нужно запустить команду из рисунка 6 Приложения А. Этот плагин позволяет оценить молекулы по заранее заданным меткам и молекулярным свойствам.

В результате моего эксперимента было сгенерировано 157 молекул и эти молекулы были записаны в файл sampling.csv. В параметрах эксперимента был указана модель reinvent.prior, которая обучена на данных из **ChEMBL 25**, реализованная с помощью рекуррентной нейронной сети (RNN) с архитектурой LSTM, содержащей 5,8 миллиона параметров. Проходил эксперимент на CPU. Пример сгенерированной молекулы: CC(=O)OCC1OC(n2cc(C(=O)CCl)c(=O)[nH]c2=O)C(OC(C)=O)C1OC(C)=O со значением NLL (Negative Log-Likelihood) = 27.37.

# Заключение

Разработка новых лекарственных препаратов является одним из ключевых задач современной медицины. Традиционные методы поиска новых молекул требуют значительных временных и финансовых затрат, что ограничивает их эффективность. Внедрение подходов машинного обучения, в частности алгоритмов обучения с подкреплением (RL), открывает новые возможности для генерации молекул с заданными свойствами. Эти методы позволяют моделировать сложные процессы взаимодействия молекул с биологическими мишенями, автоматизировать поиск оптимальных решений и значительно ускорить цикл разработки лекарств.

Новая версия программного обеспечения для дизайна молекул REINVENT 4 представляет собой как продолжение предыдущих версий, так и значительное обновление. Платформа позволяет эффективно разрабатывать инновационные молекулы, модифицировать R-группы, а также гибко изменять структуру и проводить оптимизацию молекул для заданных целей.

В ходе выполнения данного курсового проекта была изучена технология генерации молекул с использованием подхода de novo дизайна, основанного на обучении с подкреплением. Были рассмотрены ключевые алгоритмы, применяемые в платформе REINVENT, такие как регуляризованная оценка максимального правдоподобия, проксимальная оптимизация стратегии и актор-критик с различными модификациями. Особое внимание уделено использованию рекуррентных нейронных сетей для генерации строк SMILES, а также применению молекулярных и топологических скелетов для повышения качества и разнообразия создаваемых молекул.

В рамках работы был проведён эксперимент, направленный на генерацию молекул с заданными свойствами. В результате эксперимента удалось сгенерировать 157 уникальных молекул.

# Перечень использованных источников

1. S. Ivanov Reinforcement learning. *Yandex handbook ML*, 2024, no. 11.1. Available at: https://education.yandex.ru/handbook/ml/article/obuchenie-s-podkrepleniem (accessed 10.12.2024).

2. Loeffler H. H., He J., Tibo A., Janet J. P., Voronov A., …, Engkvist O. Reinvent 4: Modern AI–driven generative molecule design. *Journal of Cheminformatics*, 2024, vol. 16, no. 20. Available at: https://jcheminf.biomedcentral.com/articles/10.1186/s13321-024-00812-5 (accessed 10.12.2024). https://doi.org/10.1186/s13321-024-00812-5
17. Trott O., Olson A. J. A

3. Svensson H., Tyrchan C., Engkvist O., Chehreghani M. H. Utilizing Reinforcement learning for de novo drug design. *Machine-Mediated Learning*, 2024, vol. 113, no. 2, p. 4811–4843. https://doi.org/10.1007/s10994-024-06519-w

4. D. A. Varabyeu, A. D. Karpenko, A. V. Tuzikov, A. M. Andrianov Adaptation of the REINVENT neural network architecture to generate potential HIV-1 entry inhibitors. *BIOINFORMATICS*, 2024, vol. 21, no. 3, pp. 80-93. https://inf.grid.by/jour/article/view/1298

5. Olivecrona, M., Blaschke, T., Engkvist, O., & Chen, H. (2017). Molecular de-novo design through deep reinforcement learning. *Journal of Cheminformatics,* 2017, vol. 9, no. 1, p. 1–14. https://jcheminf.biomedcentral.com/articles/10.1186/s13321-017-0235-x

6. Zhou, H., Lin, Z., Li, J., Ye, D., Fu, Q., & Yang, W. Revisiting discrete soft actor-critic. *Transactions on Machine Learning Research*, 2024. https://arxiv.org/abs/2209.10081

7. You, J., Liu, B., Ying, Z., Pande, V., & Leskovec, J. Graph convolutional policy network for goal- directed molecular graph generation. *Advances in Neural Information Processing Systems*, 2018, no. *31*, p. 6410–6421. https://arxiv.org/abs/1806.02473

8. Maus, N., Jones, H. T., Moore, J. S., Kusner, M. J., Bradshaw, J., & Gardner, J. R. (2022). Local latent space bayesian optimization over structured inputs. *36th Conference on Neural Information Processing Systems,* 2022. https://arxiv.org/abs/2201.11872

9. Bradshaw, J., Paige, B., Kusner, M. J., Segler, M., & Hernández-Lobato, J. M. Barking up the right tree: An approach to search over molecule synthesis dags. *Advances in Neural Information Processing Systems*, 2020, no. 33, p. 6852–6866. https://arxiv.org/abs/2012.11522

10. Weininger, D. Smiles, a chemical language, and information system. 1. introduction to methodology and encoding rules. *Journal of Chemical Information and*

*Computer Sciences*, 1988, vol. 28, no. 1, p. 31–36. https://pubs.acs.org/doi/10.1021/ci00057a005

11. Rumelhart, D.E., Hinton, G. E., & Williams, R. J. Learning internal representations by error propagation. *Technical report, California Univ San Diego La Jolla Inst for Cognitive Science,* 1985. https://stanford.edu/~jlmcc/papers/PDP/Volume%201/Chap8_PDP86.pdf

12. Gao, W., Fu, T., Sun, J., & Coley, C. W. Sample efficiency matters: a benchmark for practical molecular optimization. *36th Conference on Neural Information Processing Systems*, 2022. https://arxiv.org/abs/2206.12411

13. Thomas, M., O'Boyle, N. M., Bender, A., & De Graaf, C. Re-evaluating sample efficiency in de novo molecule generation. 2022. https://arxiv.org/abs/2212.01385

14. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin Attention Is All You Need. *31st Conference on Neural Information Processing Systems* (NIPS 2017), Long Beach, CA, USA. https://arxiv.org/abs/1706.03762

15. Beckers M, Sturm N, Sirockin F, Fechner N, Stiefl N Prediction of Small-Molecule developability using large-scale in silico admet models. J Med Chem. *Journal of Medicinal Chemistry*, 2023, p. 66, 20, 14047–14060 https://doi.org/10.1021/acs.jmedchem.3c01083

16. Schneider P, Walters WP, Plowright AT, Sieroka N, Listgarten J, Goodnow RA, Fisher J, Jansen JM, Duca JS, Rush TS, Zentgraf M, Hill JE, Krutoholow E, Kohler M, Blaney J, Funatsu K, Luebkemann C, Schneider G Rethinking drug design in the artificial intelligence era. *Nature Rev Drug Discovery,* 2020*,* vol.19, no. 5, p.353–364. https://doi.org/10.1038/s41573-019-0050-3

17. Olivecrona M, Blaschke T, Engkvist O, Chen H (2017) Molecular de-novo design through deep reinforcement learning. *Journal of Chemical Information and Modeling*, 2017, vol.9, no. 1, p. 48. https://doi.org/10.1186/s13321-017-0235-x

18. Blaschke T, Arús-Pous J, Chen H, Margreitter C, Tyrchan C, Engkvist O, Papadopoulos K, Patronov A (2020) Reinvent 2.0: an ai tool for de novo drug design. *Journal of Chemical Information and Modeling,* 2020, vol.60, no. 12, p. 5918–5922. https://doi.org/10. 1021/acs.jcim.0c00915

19. Cieplinski T, Danel T, Podlewska S, Jastrzebski S (2023) Generative models should at least be able to design molecules that dock well: a new bench- mark. *Journal of Chemical Information and Modeling,* vol.63, no. 11, p. 3238–3247. https://doi.org/10.1021/ acs.jcim.2c01355

20. Dan Klein and Pieter Abbeel for *CS188 Intro to AI at UC Berkeley*. All materials available at https://ai.berkeley.edu/lecture_slides.html

21. Yousef Saad ITERATIVE METHODS FOR LINEAR SYSTEMS OF EQUATIONS: A BRIEF HISTORICAL JOURNEY. *Computer Science & Engineering, University of Minnesota, Twin Cities,* 2019. https://arxiv.org/abs/1908.01083

22. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *OpenAI,* 2017. https://arxiv.org/abs/1707.06347

23. Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., & Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In: Balcan, M.F., Weinberger, K.Q. (eds.) *Proceedings of the 33rd International Conference on Machine Learning. Proceedings of Machine Learning Research,* 2016, vol. 48, pp. 1928–1937. PMLR, New York, New York, USA. https://proceedings.mlr.press/v48/mniha16.html

24. Wang, Z., Bapst, V., Heess, N., Mnih, V., Munos, R., Kavukcuoglu, K., & de Freitas, N. Sample efficient actor-critic with experience replay. Published as a *conference paper at ICLR*, 2017. https://arxiv.org/abs/1611.01224

25. J. Schulman, S. Levine, P. Moritz ,M. Jordan, J. P. Abbeel Trust Region Policy Optimization. *31 st International Conference on Machine Learning*, Lille, France, 2015. *JMLR*: W&CP vol. 37, 2015. https://arxiv.org/abs/1502.05477

26. Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et. Soft actor-critic algorithms and applications. *Google Brain*, 2018. https://arxiv.org/abs/1812.05905

27. Bemis, G. W., & Murcko, M. A. (1996). The properties of known drugs. 1. molecular frameworks. *Journal of Medicinal Chemistry, 1996, vol. 39, no.* 15, p. 2887–2893. https://pubs.acs.org/doi/10.1021/jm9602928

28. Tim Pearce, Alexandra Brintrup, Jun Zhu Understanding Softmax Confidence and Uncertainty. *University of Cambridge and Tsinghua University*, 2021. https://arxiv.org/abs/2106.04972

# Приложение А

```
conda create --name reinvent4 python=3.10
conda activate reinvent4
pip install -r requirements-linux-64.lock # for linux
 # AMD GPUs on Linux
pip install torch==2.2.1 torchvision==0.17.1 torchaudio==2.2.1 --index-url https://download.pytorch.org/whl/rocm5.7
requirements-macOS.lock # for macOS
pip install --no-deps . # dependences (зависимости)
reinvent --help # test tool (тестировние инструмента)
```

*Рис. 4* Создание и активации виртуального окружения

```
reinvent -l sampling.log sampling.toml
```

*Рис. 5* Запуск эксперимента

```
reinvent -l scoring.log scoring.toml
```

*Рис. 6* Запуск оценочного плагина