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

Разработка глубокой нейронной сети для оценки энергии связывания химических соединений с молекулярной мишенью

Курсовая работа

Благодарного Артёма Андреевича
обучающегося 3 курса
специальности «Информатика»

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

Минск, 2025

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.

Глава 1. 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].

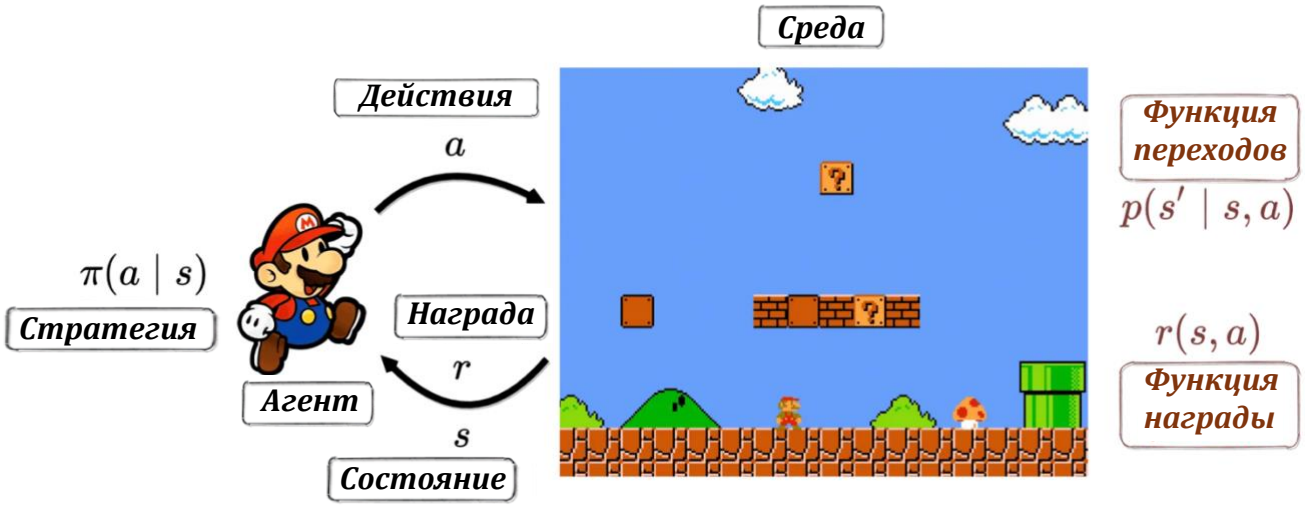

Рис. 1. Общая схема обучения

На рисунке 1 изображена общая схема обучения, в которой среда находится в состоянии s, агент, используя стратегию π, 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 π ($a \mid s$) 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 π 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 γ, который меньше единицы. Тогда наш функционал примет такой вид:

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

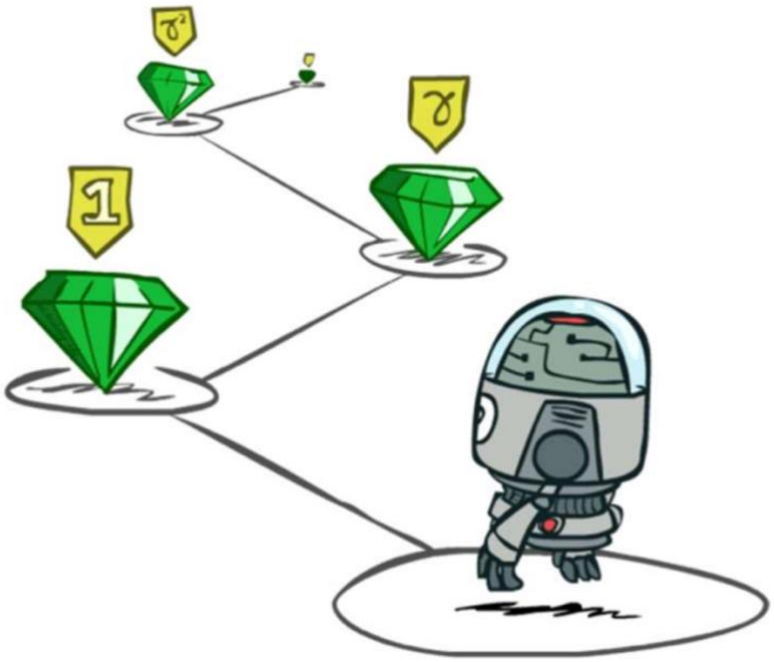

Рис. 2. Концепция дисконтирования награды

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]:

$$\pi^*(s)\ = argmax_a Q^*(s,a). \tag{5}$$

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 γ. 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 2. Данные для обучения модели

Эффективность обучения модели глубокой нейронной сети в значительной степени зависит от качества, разнообразия и репрезентативности используемых данных. Особенно это важно в задачах, связанных с биоинформатикой и медицинской химией, где модель должна научиться распознавать сложные пространственные и химические взаимодействия между молекулами.

Одной из ключевых задач, решаемых в этой области, является *молекулярный докинг*. **Докинг** — это метод, применяемый в структурной биоинформатике и хемоинформатике, с целью моделирования взаимодействия между двумя молекулами: как правило, небольшой молекулы-лиганда и более крупной мишени, чаще всего — белка. Задача докинга заключается в том, чтобы предсказать наиболее вероятную пространственную конфигурацию комплекса «белок–лиганд» и оценить силу этого взаимодействия.

В процессе докинга предполагается, что у белка имеется так называемый *белковый карман* — это область на поверхности белка, куда может «вписаться» молекула лиганда. От того, насколько хорошо лиганд подходит к форме и химическим свойствам этого кармана, зависит прочность связывания. Эта прочность количественно выражается через *энергию связывания*: чем она ниже, тем более устойчив комплекс, и тем выше вероятность, что такое взаимодействие имеет биологическую значимость.

Традиционные методы докинга используют физико-химические модели для оценки энергии связывания, перебирают различные позы (конформации) лиганда в кармане белка и рассчитывают скоринговые функции, отражающие степень совместимости. Однако такие методы могут быть ограничены по точности, требуют больших вычислительных ресурсов и плохо масштабируются.

2.1. Лиганд

Лиганд — это небольшая молекула, которая может связываться с более крупной молекулой, такой как белок. В контексте биологии и медицины лиганды — это, как правило, лекарственные соединения, которые предназначены для связывания с конкретной молекулой-мишенью, чтобы повлиять на её поведение. Структура лиганда описывается в виде химической формулы, где указано, какие атомы входят в состав, как они соединены между собой, и какие свойства у этих атомов есть. В машинном обучении лиганды часто представляют в виде графов: атомы становятся узлами, а химические связи — рёбрами. Это позволяет использовать графовые нейронные сети для обработки структуры лиганда.

При обучении сети лиганд бует представлен в виде строки **SMILES** (Simplified Molecular Input Line Entry System) — компактного текстового формата,

описывающего химическую структуру молекулы. Это один из самых распространённых способов представления молекул в хемоинформатике и машинном обучении, особенно когда используется обработка последовательностей (например, трансформеры или рекуррентные нейронные сети).

SMILES — это строка символов, в которой последовательно зашифрованы атомы, типы химических связей, кольцевые структуры, ветвления и даже стереохимическая информация. Каждый атом обозначается своим химическим символом (C — углерод, O — кислород, N — азот и т. д.), а связи между ними — с помощью специальных символов: - для одинарной связи, = для двойной, # для тройной. Если связь не указана, по умолчанию предполагается одинарная.

Пример простой молекулы — этанол, в SMILES-записи: CCO. Эта строка означает, что два атома углерода соединены между собой, и ко второму углероду присоединён атом кислорода.

Формат SMILES удобен тем, что позволяет хранить молекулы в виде обычного текста, что значительно упрощает обработку данных, их хранение и передачу. Кроме того, строки SMILES легко использовать в алгоритмах машинного обучения: они хорошо подходят для подачи на вход моделям, обучающимся на текстах, и могут быть преобразованы обратно в структурные форматы при помощи специализированных библиотек (например, RDKit).

2.2. Белковый карман

Белковый карман — это часть белковой молекулы, в которой может происходить связывание лиганда. Белки — это крупные и сложные молекулы, состоящие из цепочек аминокислот, свернутых в трёхмерные структуры. На их поверхности или внутри образуются углубления и ниши — так называемые карманы. Эти участки обладают определёнными химическими свойствами (гидрофобность, наличие зарядов, и т. д.), которые делают их подходящими для связывания с определёнными молекулами. Карман можно представить как «замочную скважину», а лиганд — как «ключ», который должен идеально подойти.

Белковый карман представлен в виде трёхмерного объёма — 3D-решётки, содержащей пространственное распределение атомов и их химические свойства в области связывания белка.

Это значит, что вокруг активного участка белка, где происходит взаимодействие с лигандом, выделяется кубическая область определённого размера, которую разбивают на маленькие ячейки — воксели (3D-пиксели). В каждом вокселе содержится информация о наличии атомов различных типов, их электрохимических характеристиках или других признаках, важных для связывания. Такая 3D-решётка называется *окрестностью белкового кармана*.

Данное представление позволяет учитывать не только структуру и расположение атомов, но и их пространственные взаимодействия, что критично для точного моделирования процессов связывания. Именно поэтому в работе используется трёхмерная сверточная нейронная сеть, которая способна эффективно выделять признаки из такого 3D-формата и создавать компактное эмбеддинг-представление белкового кармана.

Одним из классических примеров взаимодействия белка с лигандом является структура **HIV-1 протеазы** в комплексе с ингибитором, представленной в базе Protein Data Bank под идентификатором **1HVR.**

HIV-протеаза — это фермент, критически важный для созревания вируса иммунодефицита человека. Он расщепляет вирусные полипротеиновые цепи на функциональные белки. Именно активный центр протеазы служит белковым карманом**,** куда может связываться как естественный субстрат, так и синтетические ингибиторы.

В данной структуре в активный центр протеазы помещён синтетический ингибитор — производное циклической мочевины**,** относящийся к классу непептидных ингибиторов. Этот лиганд занимает тот же карман, в который обычно входит пептидный субстрат, и таким образом блокирует функцию фермента, предотвращая размножение вируса.

Белковый карман в данном случае — это глубокая выемка в центре протеазы, образованная симметрично двумя субъединицами фермента. Он стабилизируется водородными связями, гидрофобными взаимодействиями и рядом ключевых остатков.

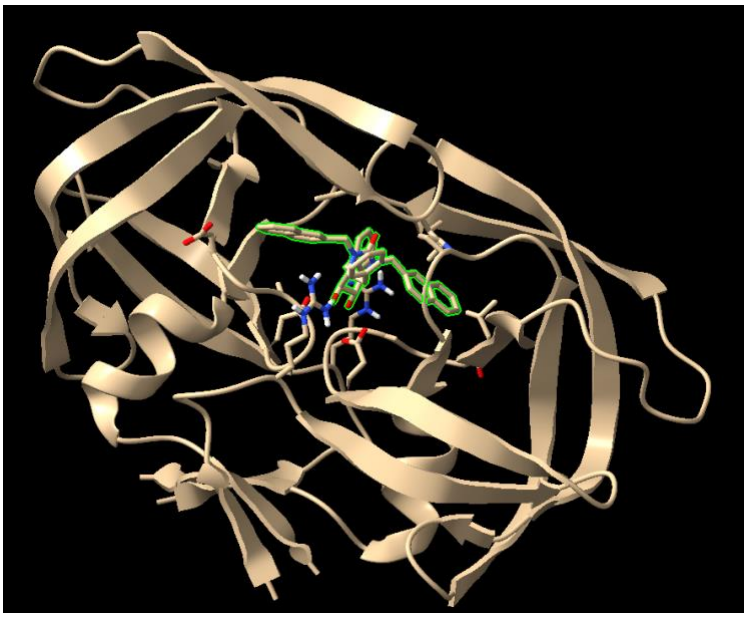

Рис.3. HIV-1 протеаза с выделенным лигандом

2.3. Датасет

В рамках данной работы используются данные, предоставленные лабораторией Teresa Head-Gordon (THGLab) из Университета Калифорнии в Беркли. Эта лаборатория известна своими исследованиями в области молекулярного моделирования, химической физики и биофизики. Одним из значимых вкладов лаборатории является разработка высококачественного набора данных для предсказания энергии связывания белок-лиганд — HiQBind.

HiQBind был создан с использованием автоматизированного рабочего процесса HiQBind-WF, который включает алгоритмы для исправления распространённых структурных артефактов в данных, таких как ошибки в 3D-структурах белков и лигандов, а также несоответствия в отчетах о связывающих энергиях. Этот набор данных включает в себя кристаллизованные комплексы белок-лиганд с соответствующими значениями связывающих энергий, полученными из различных источников, таких как BioLiP2, Binding MOAD и BindingDB. HiQBind предназначен для обеспечения воспроизводимости исследований и минимизации человеческого вмешательства, при этом он является открытым и доступным для научного сообщества.

Учёные из лаборатории представили датасет LP-PDBBind (Leak Proof PDBBind) — это переработанный набор данных, предназначенный для более обоснованной и обобщаемой оценки энергии связывания белок-лиганд. Он был представлен в статье: "Защищенный от утечек PDBBind: Реорганизованный набор данных о белково-лигандных комплексах для более обобщенного прогнозирования аффинности связывания".

Исходный набор данных PDBBind, широко используемый для обучения и тестирования функций оценки связывания, страдает от проблемы утечки данных. Это означает, что в тренировочных и тестовых наборах могут присутствовать схожие или идентичные белки и лиганды, что приводит к переоценке реальной способности моделей обобщать на новые данные. Такая утечка может искажать результаты и не отражать истинную эффективность моделей на ранее не встречавшихся комплексах.

Для решения этой проблемы был разработан LP-PDBBind, в котором особое внимание уделено контролю за схожестью данных между тренировочным, валидационным и тестовым наборами. В частности, были введены следующие меры:

- Минимизация схожести по последовательности белков: максимальная схожесть белков между наборами ограничена значением 0.5.

- Минимизация химической схожести лигандов: максимальная схожесть лигандов между наборами ограничена значением 0.99.
- Учет структурной схожести взаимодействий: использование взаимодействий, основанных на химико-структурных признаках, для обеспечения различий в паттернах связывания между наборами.

После переработки LP-PDBBind включает:

- Тренировочный набор: 11,513 комплексов белок-лиганд.
- Валидационный набор: 2,422 комплекса.
- Тестовый набор: 4,860 комплексов.

LP-PDBBind включает метаинформацию о белок-лигандных комплексах, представленную в CSV-файле LP_PDBBind.csv. Этот файл содержит следующие столбцы:

1. **Unnamed: 0**: Индекс строки в DataFrame.
2. **header**: Краткое описание функции белка, например, "isomerase" или "hydrolase".
3. **smiles**: SMILES-строка, представляющая химическую структуру лиганда.
4. **category**: Категория белка, определенная по типу его активности.
5. **seq**: Аминокислотная последовательность белка.
6. **resolution**: Решение структуры комплекса в ангстремах, например, 1.36 Å.
7. **date**: Дата публикации структуры в PDB.
8. **type**: Тип белка, например, "isomerase" или "hydrolase".
9. **new_split**: Категория раздела данных: "train", "validation" или "test".
10. **CL1, CL2, CL3**: Булевы значения, указывающие, соответствует ли комплекс соответствующему уровню очистки данных.
11. **remove_for_balancing_val**: Булевое значение, указывающее, был ли комплекс исключен для балансировки валидационного набора.
12. **kd/ki**: Оригинальное значение связывающей энергии (Kd или Ki) с единицами измерения, например, "Kd=0.006uM".
13. **value**: Логарифмированное значение связывающей энергии (pKd или pKi).
14. **covalent**: Булевое значение, указывающее, является ли связь между белком и лигандом ковалентной.

Глава 3. Архитектура нейронной сети

Нейронная сеть предназначена для решения задачи регрессии, цель которой — предсказать энергию связывания между белком и лигандом. Эта энергия является количественной мерой силы взаимодействия между молекулами и играет ключевую роль при изучении биохимических процессов и разработке новых лекарственных препаратов.

Для решения такой задачи необходимо учитывать особенности двух типов объектов — белка и лиганда — которые существенно различаются по своей структуре и формату представления. Белок — это крупная трёхмерная молекула, которая характеризуется пространственным расположением атомов и их физико-химическими свойствами в 3D-пространстве. Лиганд же, как правило, представляет собой небольшое органическое соединение, которое удобно описывать с помощью химической нотации строки SMILES, отражающей порядок атомов и их связи.

3.1. Схема обучения модели

Поэтому входные данные для модели формируются двумя разными путями:

- Сначала структурная информация о белке преобразуется в удобный для анализа формат: в виде трёхмерного сеточного (воксельного) представления. Такой формат позволяет сохранить важные пространственные взаимосвязи и локальные особенности белковой структуры.
- Вторая часть входных данных — это описание лиганда. Его химическая структура, заданная в формате SMILES, предварительно преобразуется в графовую структуру, где узлы соответствуют атомам, а рёбра — химическим связям. Такой графовый формат наиболее естественен для молекулярных данных, поскольку сохраняет топологию молекулы и взаимосвязи между атомами.

Поскольку белок и лиганд представляют собой разные типы данных (3D-сетка и граф), они обрабатываются отдельными специализированными модулями — энкодерами. Энкодер белка отвечает за извлечение информативных признаков из пространственного представления, выявляя ключевые характеристики структуры, которые влияют на связывание с лигандом. Энкодер лиганда, в свою очередь, извлекает признаки из графовой структуры, отражающие химическую природу молекулы и её реакционную способность.

После обработки отдельными энкодерами, полученные векторные представления (эмбеддинги) белка и лиганда объединяются в единое общее представление. Это объединение осуществляется с помощью конкатенации или других методов слияния признаков. Объединённый вектор затем передаётся в

последующие слои нейронной сети, которые выполняют регрессионный анализ и предсказывают энергию связывания.

Такой пайплайн позволяет эффективно использовать всю доступную информацию о белке и лиганде, учитывая их специфику и структурные особенности. Благодаря модульному подходу возможна гибкая настройка и расширение модели, что делает её универсальной для различных задач биоинформатики и молекулярного докинга.

Разделение энкодеров на белковый и лигандный обосновано фундаментальными различиями между структурами: белки — это сложные макромолекулы с пространственной организацией, тогда как лиганды — это небольшие молекулы, где важна топология графа. Совместное представление этих двух эмбеддингов затем используется в actor- и critic-сетях, что позволяет модели учитывать и взаимодействие между молекулами, и индивидуальные свойства каждой из них.

3.2. Структура сети

DockingAgent — это нейросетевая модель, которая учится принимать решения для задачи докинга (то есть поиска оптимальной позы лиганда относительно белка). Она состоит из 4 основных компонентов: энкодер белка, энкодер лиганда, actor-сеть и critic-сеть.

Энкодер белка реализован на основе трёхмерной сверточной нейронной сети (3D CNN), которая обрабатывает воксельное представление белка. Такой подход позволяет учитывать пространственные особенности белковой структуры, включая расположение аминокислот и возможные сайты связывания. Трёхмерные свёртки, в отличие от 2D или 1D аналогов, способны выявлять сложные пространственные шаблоны в объеме, что критично для задач молекулярного докинга.

Модель использует последовательность свёрточных блоков с нарастающим числом каналов: 16 → 32 → 64. Это даёт возможность сначала выделить базовые локальные признаки, а затем — более абстрактные и глобальные. После каждой свёртки применяется BatchNorm3D, который стабилизирует обучение, а MaxPool3D понижает размерность и фокусирует внимание на ключевых областях. Финальный слой — полносвязный (FC) — приводит объёмное представление к вектору фиксированной длины (128), который служит эмбеддингом белка.

Лиганды представляют собой молекулы, удобные для представления в виде графов: атомы соответствуют узлам, а химические связи — рёбрам. Энкодер лиганда реализован в виде графовой нейронной сети (GNN) с использованием слоя GCNConv, что позволяет агрегировать информацию от соседних узлов, эффективно

распространяя контекст по молекуле. Это особенно важно, поскольку химическая активность атома зависит не только от его свойств, но и от окружения.

Сеть использует три слоя GCNConv: 32 → 64 → 128, что позволяет на каждом уровне собирать более комплексные признаки. После последнего слоя создаётся глобальный эмбеддинг лиганда, например, с помощью агрегации (глобального пула), который представляет молекулу в векторном пространстве фиксированной размерности. Такой подход даёт модели возможность учитывать химическую структуру и свойства лиганда при дальнейшем взаимодействии с белком.

Actor-сеть отвечает за предсказании значения энергии связывания. На вход поступает объединённый вектор размерности 257, составленный из эмбеддингов белка (128), эмбеддинга лиганда (128) и одного дополнительного позиционного признака (например, координаты или параметра действия). Сеть состоит из двух полносвязных слоёв, которые последовательно преобразуют входной вектор до скрытого состояния.

Завершающие слои сети — mean_head и log_std_head — позволяют предсказывать параметры нормального распределения: математическое ожидание и логарифм стандартного отклонения. Это особенно полезно при стохастическом обучении, где действия могут сэмплироваться из предсказанного распределения, а не быть строго детерминированными. Такой подход повышает устойчивость модели и позволяет лучше исследовать пространство действий.

Critic-сеть используется для оценки «качества» действия или значения, предсказанного actor-сетью. В архитектуре она очень похожа на actor: используется два полносвязных слоя и финальный предсказатель — q_head. Однако на вход подаётся вектор большей размерности (258), так как к объединённому эмбеддингу белка и лиганда добавляется ещё и значение действия.

Эта оценка служит вспомогательной задачей (в духе обучения с подкреплением) и может использоваться для более стабильного и направленного обучения actor-сети. Critic помогает отсеивать неудачные действия, снижая вероятность их повторного выбора, и тем самым улучшает общую эффективность модели.

Объединение признаков белка и лиганда — ключевой шаг, на котором происходит формирование общего представления о взаимодействии двух молекул. В DockingAgent это достигается путём простой конкатенации их эмбеддингов, полученных из соответствующих энкодеров. Полученный вектор содержит как пространственную информацию от белка, так и химическую структуру лиганда.
Дополнительно к этому вектору может быть добавлена информация о начальной позиции лиганда, координатах или других вспомогательных признаках (например,

шум для стохастичности). Такое объединение позволяет модели анализировать взаимодействие молекул как единое целое.

Использование **3D CNN** для белка обусловлено тем, что его активные участки и сайты связывания обладают выраженной пространственной структурой. Только трёхмерная обработка позволяет выявить паттерны, которые могут быть критичны для взаимодействия с лигандом. Такие модели успешно применяются в задачах вроде распознавания белковых карманов и оценки их геометрии.

С другой стороны, **GNN** для лиганда позволяет учитывать топологические и химические особенности молекулы, которые неявно закодированы в SMILES, но явно выражены в графовой форме. Это делает модель более устойчивой к вариациям SMILES и позволяет точнее определять функциональные группы и их взаимное расположение. Таким образом, архитектура DockingAgent сочетает в себе сильные стороны обеих техник, что делает её эффективной в задаче молекулярного докинга.

```
DockingAgent(
  (protein_encoder): ProteinEncoder3DCNN(
    (conv1): Conv3d(4, 16, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
    (bn1): BatchNorm3d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (conv2): Conv3d(16, 32, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
    (bn2): BatchNorm3d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (conv3): Conv3d(32, 64, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
    (bn3): BatchNorm3d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (pool): MaxPool3d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (fc): Linear(in_features=4096, out_features=128, bias=True)
  )
  (ligand_encoder): LigandGNN(
    (convs): ModuleList(
      (0): GCNConv(30, 64)
      (1): GCNConv(64, 64)
      (2): GCNConv(64, 128)
    )
  )
  (actor): Actor(
    (fc1): Linear(in_features=259, out_features=256, bias=True)
    (fc2): Linear(in_features=256, out_features=256, bias=True)
    (mean_head): Linear(in_features=256, out_features=3, bias=True)
    (log_std_head): Linear(in_features=256, out_features=3, bias=True)
  )
  (critic): Critic(
    (fc1): Linear(in_features=262, out_features=256, bias=True)
    (fc2): Linear(in_features=256, out_features=256, bias=True)
    (q_head): Linear(in_features=256, out_features=1, bias=True)
  )
)
```

Рис.4. Структура модели DockingAgent

3.3. Взаимодействие агента со средой

Цикл взаимодействия агента с окружением в процессе обучения представляет собой последовательную цепочку шагов, направленных на накопление опыта и последующее обновление параметров модели. На каждом шаге агент получает от среды текущее состояние, которое включает в себя структурное представление белка (prot_tensor), информацию о лигандной молекуле (ligand_data), текущую позу

лиганда (pose), а также числовое значение награды (reward) и булевый флаг done, который сигнализирует об окончании эпизода. Эти данные поступают в модель через функцию env_step(action).

После получения этих данных агент проводит их внутреннюю обработку. Сначала вызывается метод encode(), который преобразует входные биохимические данные в компактное векторное представление состояния, пригодное для работы с нейросетевой архитектурой. Далее, на основе закодированного состояния актор модели генерирует действие — это, как правило, параметры трансформации, которые определяют, как лиганд должен сместиться или повернуться относительно белка. Вместе с действием также вычисляется логарифм вероятности его выбора — это нужно для расчёта энтропийного слагаемого в алгоритме Soft Actor-Critic (SAC), который способствует более исследовательскому поведению.

После генерации действия оно снова подаётся в среду, и агент получает следующее состояние и награду. Этот переход (state, action, reward, next_state, done) сохраняется в специальное хранилище — replay buffer. Там накапливаются пары «опытных переходов», из которых позже, после достижения определённого объёма, случайным образом формируются батчи. Эти батчи используются для обучения: актор и критик обновляют свои параметры на основе предсказаний и ошибок, извлечённых из уже накопленного опыта. Таким образом, агент постепенно обучается выбирать действия, которые максимизируют ожидаемую награду за счёт многократного взаимодействия с окружением.

3.4. Обучение критика

В рамках реализации алгоритма **Soft Actor-Critic (SAC)** обучение критика основано на оценке Q-функции — функции полезности, показывающей ожидаемое суммарное вознаграждение при выполнении действия a в состоянии s. Поскольку истинные значения Q-функции заранее неизвестны, критик обучается по приближённым целевым значениям (target values)**,** которые учитывают как будущие награды, так и энтропию политики. Целевое значение вычисляется по следующей формуле:

$$target\_value = r + \gamma \times (1 - done) \times \left(Q_{target}(s', a') - \alpha \times log\pi(a'|s')\right), \quad (8)$$

где:

- r — награда, полученная за текущее действие;
- γ — дисконт-фактор (в данном случае установлен на 0.99);
- $done$ — булев флаг завершения эпизода;

- $Q_{target}(s', a')$ — оценка следующего состояния и действия, предсказанная таргетным критиком;
- $log\pi(a'|s')$ — логарифм вероятности действия по текущей политике;
- α — коэффициент, регулирующий вклад энтропии.

Энтропийный член $log\pi(a'|s')$ играет важную роль: он стимулирует агента к исследованию среды, поощряя стохастические действия. Это делает обучение более устойчивым, особенно в условиях высокой неопределённости среды.

В коде это реализовано следующим образом:

1. Без вычисления градиентов агент получает следующее действие и его логарифмическую вероятность.
2. Полученное действие передаётся таргетному критику, который предсказывает $Q(s', a')$.
3. Далее вычисляется target_value с учётом дисконтированной награды и энтропийного слагаемого.
4. Текущее предсказание Q-функции (current_q) сравнивается с target_value, и на этой основе рассчитывается функция потерь критика:

$$loss\_critic = F.mse\_loss(current\_q, target\_value), \quad (9)$$

где:

- current_q — предсказание Q-функции по текущим параметрам критика.
- target_value — значение, рассчитанное по описанной выше формуле.

Таким образом, критик минимизирует среднеквадратичную ошибку между своим прогнозом и целевым значением, формируя более точные оценки полезности действий. Это позволяет агенту в будущем принимать более обоснованные решения, не только максимизируя ожидаемое вознаграждение, но и поддерживая баланс между исследованием и использованием уже изученной информации.

Глава 4. Результаты

Для обучения модели был использован датасет LP-PDBBind, содержащий комплексные структуры белок-лиганд и уже готовыми Логарифмированными значениями связывающей энергии. Данные были случайным образом разделены на обучающую и тестовую выборки в соотношении 80/20.

4.1. Результаты процесса обучения

На 80% обучающей выборки модель обучалась в течение 10 000 эпох. В ходе обучения отслеживались значения функции потерь для критика (loss_critic) и актёра (loss_actor). Ниже приведены результаты лосса актёра:

Рис.5. Значения лосс-функции актёра

Лосс актёра демонстрирует чёткую убывающую динамику, начиная с ≈1.02 и стремительно снижаясь до значений близких к нулю и даже отрицательных, что указывает на успешное обучение политики: на ранних этапах лосс быстро уменьшается, а затем переходит в область низких и флуктуирующих значений, что характерно для достижения плато и тонкой настройки. Небольшие колебания и редкие отрицательные значения в конце могут свидетельствовать о близости к сходимости или небольшом переобучении, но в целом поведение лосса типично для стабилизирующегося обучения в Actor-Critic методах.

Ниже приведены результаты лосса критика:

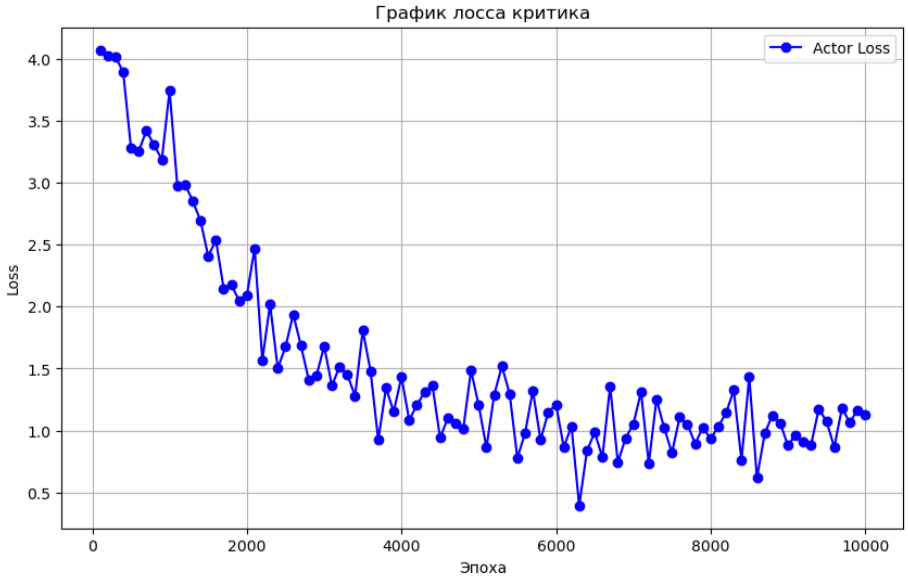

Рис.6. Значения лосс-функции

Лосс критика показывает характерное для обучения поведение: он начинается на уровне около 4, быстро снижается в первые итерации и затем колеблется в пределах 1.0–1.5, что указывает на то, что модель, вероятно, выучила усреднённые значения value-функции, но не точно различает выгоды от разных действий; это может быть признаком недообучения критика, стохастичности среды или чрезмерного сглаживания предсказаний.

4.2. Результаты на тестовых данных

Для проверки обобщающей способности модели была проведена финальная оценка на тестовой выборке, включающей 20% исходных данных. Оценка производилась по следующим метрикам:

Метрика	Значение
MAE (средняя абсолютная ошибка)	**1.248**
MSE (среднеквадратичная ошибка)	**1.141**
RMSE (корень из MSE)	**1.117**
MAPE (средняя абсолютная процентная ошибка)	**22.3%**
R^2 (коэффициент детерминации)	**0.71**

Рис.7. Значение метрик на валидационных данных

Анализ результатов модели с учётом диапазона целевых значений от 0.4 до 15.22 показывает, что модель демонстрирует умеренное качество предсказаний. Средняя абсолютная ошибка (MAE = 1.248) означает, что в среднем предсказания модели отклоняются от истинных значений примерно на 1.25 единицы. При таком широком диапазоне значений таргета (почти от 0 до 15), это отклонение можно считать относительно небольшим, особенно если рассматривать абсолютные величины: ошибка занимает примерно 8–10% от диапазона значений, что свидетельствует о достаточной точности для многих практических задач.

Среднеквадратичная ошибка (MSE = 1.141) и её корень (RMSE = 1.117) подтверждают, что ошибки модели не имеют сильных выбросов — разница между RMSE и MAE незначительна, что говорит о равномерном распределении ошибок без крупных редких сбоев.

Однако, показатель средней абсолютной процентной ошибки (MAPE = 22.3%) указывает на то, что в среднем предсказания отклоняются на 22% от реальных значений. Высокий процент ошибки связан с тем, что для небольших истинных значений таргета (близких к 0.4) даже небольшие абсолютные ошибки приводят к большим относительным отклонениям. Это типичная ситуация для данных с широким разбросом значений, где MAPE может быть чувствителен к малым таргетам.

Коэффициент детерминации (R^2 = 0.71) показывает, что модель объясняет порядка 71% дисперсии в данных. Это довольно хороший результат, означающий, что модель захватывает основные зависимости и тренды, но ещё остаётся пространство для улучшения — почти треть вариации в целевой переменной остаётся необъяснённой, что может быть вызвано шумом в данных, недостатком информативных признаков или ограничениями самой модели.

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.

В ходе работы была разобрана задача обучения с подкрепнием в терминах марковского процесса принятия решений: были рассмотрены понятия среды и агента, описано пространства состояний и действий, функции переходов и награды, и сформулирована целевая функция максимизации суммарной дисконтированной награды. При этом ключевую роль сыграли уравнение Беллмана и итеративный метод обновления Q-функции, позволяющие приближаться к оптимальной стратегии. Для обучения модели был использован датасет LP-PDBBind с более чем 19 000 белок–лигандных комплексов: лиганды представлены в формате SMILES, а белковые карманы — в виде 3D-вокселей, что позволило эффективно извлекать как пространственные, так и химические признаки. Архитектура модели включает два энкодера (энкодер лиганда и энкодер белкового кармана), потом полученные эмбеддинги объединяются и передаются в Actor и Critic — оба построены на полносвязных слоях и работают по алгоритму Soft Actor-Critic с энтропийным членом для баланса между исследованием и использованием среды.

В ходе обучения в течение 10 000 эпох продемонстрировало уверенное снижение лоссов. На тестовой выборке модель достигла MAE = 1.248, $R^2 = 0.71$ и MAPE = 22.3 %, что свидетельствует о её способности объяснять большую часть вариативности данных и делать точные предсказания энергии связывания. В дальнейшем стоит пересмотреть структуру модели, подобрать датасет для обучения с большим количеством примеров белок-лиганд с одним и тем же белком.

В целом, проделанная работа демонстрирует, что методы глубокого усиленного обучения способны эффективно решать задачу предсказания белок–лигандных взаимодействий в формате молекулярного докинга. Полученные результаты закладывают прочную основу для дальнейших исследований и практического применения в ускоренной разработке лекарственных препаратов, где точное предсказание энергии связывания может существенно сократить время и ресурсы при отборе перспективных кандидатов.

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