МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Кафедра биомедицинской информатки

МОРДАНЬ

Евгений Игоревич

# Применение технологий машинного обучения и молекулярного моделирования для идентификации потенциальных ингибиторов проникновения ВИЧ-1

Дипломная работа

Научный руководитель:

доктор химических наук,

Андрианов Александр Михайлович

Допущена к защите

«___» _____ 2021 г.

Зав. Кафедрой биомедицинской информатики

Кандидат физико-математических наук, доцент Ю. Л. Орлович

Минск, 2021

# РЕФЕРАТ

Дипломная работа, 51 страница, 26 рисунков, 9 таблиц, 11 формул, 8 источников

**Ключевые слова:** МАШИННОЕ ОБУЧЕНИЕ, НЕЙРОННЫЕ СЕТИ, АВТОЭНКОДЕРЫ, РЕКУРРЕНТНЫЕ НЕЙРОННЫЕ СЕТИ, LSTM ЯЧЕЙКИ , ИНГИБИТОРЫ ПРОНИКНОВЕНИЯ, МОЛЕКУЛЯРНОЕ МОДЕЛИРОВАНИЕ.

**Объект исследования:** набор веществ-потенциальных ингибиторов белка gp120.

**Цель работы:** изучение работы автоэнкодеров с LSTM слоями, создание алгоритма идентификации новых потенциальных ингибиторов белка gp120.

**Методы исследования:** методы машинного обучения, методы молекулярного моделирования.

**Результат:** исследована работа LSTM слоев и автоэнкодеров, разработаны два алгоритма идентификации новых потенциальных ингибиторов, с помощью алгоритмов сгенерированы новые потенциальные ингибиторы, проведен анализ полученных ингибиторов, в том числе молекулярный докинг.

**Область применения:** разработка лекарственных препаратов.

# РЭФЕРАТ

Дыпломная праца, 51 старонка , 26 малюнкаў , 9 табліц, 11 формул, 8 крыніцы

**Ключавыя словы:** МАШЫННАЕ НАВУЧАННЕ, НЕЙРОННЫЯ СЕТКІ, АЎТАЭНКОДЕРЫ, РЭКУРЭНТНЫЯ НЕЙРОННЫЯ СЕТКІ, LSTM ЯЧЭЙКІ, ІНГІБІТАРЫ ПРАНІКАННЯ, МАЛЕКУЛЯРНАЕ МАДЭЛІРАВАННЕ.

**Аб’ект даследавання:** набор рэчываў-патэнцыяльных інгібітараў белку gp120.

**Мэта работы:** вывучэнне працы автўэнкодераў с LSTM слаямі, стварэнне алгарытму ідэнтыфікацыі новых патэнцыяльных інгібітараў белку gp120.

**Метады даследвання:** метады машыннага навучання, метады малекулярнага мадэліравання.

**Вынік:** даследвана праца LSTM слаёў і аўтаэнкодераў, разпрацаваны два алгарытмы ідэнтыфікацыі новых патэнцыяльных інгібітараў, с дапамогай алгарытмаў згенерыраваны новыя патэнцыяльныя інгібітары, праведзен аналіз атрыманых інгібітараў, у тым ліку малекулярны докінг.

**Вобласть прымянення:** разпрацоўка лекавых прэпаратаў.

# ABSTRACT

Diploma thesis, 51 pages, 26 pictures, 9 tables, 11 formulas, 8 sources.

**Keywords:** MACHINE LEARNING, NEURAL NETWORKS, AUTOENCODERS, RECURRENT NEURAL NETWORKS, LSTM CELLS, ENTRY INHIBITORS, MOLECULAR MODELLING.

**Object of research:** set of compounds-potential gp120 protein inhibitors.

**Objective:** LSTM autoencoders work research, potential gp120 protein inhibitors identification algorithm development.

**Methods of research**: machine learning methods, molecular docking methods.

**The result:** research of LSTM autoencoders, two algorithms of potential gp120 protein inhibitors identification development, new potential inhibitors generation, analysis and molecular docking.

**The scope:** drug design.

# ОГЛАВЛЕНИЕ

# ВВЕДЕНИЕ

Одной из конечных целей молекулярной разработки новых материалов или лекарств является генерация или идентификация соединений с некоторыми заданными свойствами. На данный момент это довольно сложная задача из-за того, что пространство химических соединений необычайно обширное, разнообразное и насыщенное множеством тех или иных типов молекул. Например, на данный момент синтезировано всего лишь $10^8$ лекарственных соединений, хотя считается, что их количество находится в районе $10^{23} – 10^{60}$. Даже несмотря на достигнутые успехи в экспериментальном синтезировании соединений, процесс разработки новых материалов продолжает оставаться очень дорогим и сложным методом.

Альтернативный экспериментальному подходу синтезирования является метод компьютерной генерации соединений. Он позволяет преодолеть физические и ресурсные ограничения экспериментального подхода. В настоящее время основные методы компьютерной генерации соединений основаны на глубоком обучении, а именно на глубоких нейронных сетях. Нейронные сети способны обучаются тому, как устроены молекулы, представленные в некотором формате, и потом могут генерировать новые уникальные молекулы в том же формате по изученным правилам.

Одним из основных шагов в анализе полученных в ходе генерации соединений является молекулярный докинг. В ходе молекулярного докинга ищется наилучшая конформация между полученным соедниением и целью докинга. От того насколько хорошо полученное соединение связано с целью, зависит дальнейшая необходимость в более глубоком анализе этого соединения.

# Глава 1. ОБЩИЕ СВЕДЕНИЯ

## 1.1 Вирус иммунодефицита человека

ВИЧ или вирус иммунодефицита человека, это вирус поражающий иммунные клетки человека. Вирус ослабляет иммунную систему и подвергает риску заражения другими инфекциями. Первые пациенты, вызвавшие подозрения заражением ВИЧ, были зарегистрированы в Нью-Йорке и Лос-Анджелесе в 1980х годах. С этого момента более 70 миллионов человек заразились ВИЧ инфекцией. Заражение ВИЧ происходит при попадании биожидкости зараженного человека в кровь здорового. При попадании вируса в тело человека происходит резкое снижение количества белых кровяных телец в крови, а далее разрушение клеток Т4, которые отвечают за защиту организма. При этом количество клеток Т8, которые наоборот подавляют иммунную систему, возрастает в 4 раза. В первые дни после заражения возможно повышение температуры до 40 градусов, и появление гриппоподобных симптомов, однако уже после 14 дней все симптомы пропадают и наступает так называемый латентный период длительностью примерно от 4 месяцев до 12 лет. В этот промежуток количество клеток Т4 снижается в разы.

Целью ВИЧ являются клетки, содержащие рецепторы CD4 на своей мембране. Такими клетками являются макрофаги, дендритные клетки и Т-хелперы. Все эти клетки имеют рецептор CD4 и, как следствие, являются мишенями для ВИЧ.

На данный момент выявлено два вида ВИЧ: ВИЧ-1 и ВИЧ-2. ВИЧ1 более распространен в мире, в то время как ВИЧ-2 более редкий и ограничен областями Западной Африки и Южной Азии. Далее под ВИЧ будем подразумевать именно ВИЧ-1.

Последняя стадия ВИЧ — СПИД или синдром приобретенного иммунодефицита. Для этой стадии характерно развитие различных инфекций, заболеваний и опухолей, которые возникают с ослаблений иммунной системой. Страдают легкие, печень, нервная система и в целом весь организм больного. СПИД характеризуется более низким содержанием уровня Т клеток, чем ВИЧ: до 200 клеток/мм$^3$ против 200 до 500 клеток/мм$^3$ для ВИЧ. Чаще всего заражение происходит через незащищенный половой контакт, а также через применение инъекционных наркотиков. При заражении через половой контакт, вирус попадает как правило в дендритные клетки, которые находятся в эпителиальной ткани, в слизистой. Далее вирус мигрирует в лимфатические узлы, где вирус в итоге заражает Т-хелперы, макрофаги и еще больше

дендритных клеток. Средняя продолжительность жизни больного ВИЧ составляет около 10 лет. Однако благодаря разработке антиретровирусной терапии, больной может прожить жизнь как здоровый человек.

## 1.2 Средства для лечения ВИЧ

На данный момент универсальной вакцины или лекарства от ВИЧ не существует, однако учеными была разработана особая, так называемая комбинированная схема лечения ВИЧ. Такая схема включает в себя прием сразу нескольких лекарственных препаратов — ингибиторов, веществ, подавляющих или задерживающих течение физиологических и физико-химических процессов. Ингибиторы ВИЧ имеют разные фармакологические свойства и подразделяются на разные классы, например:

- **Ингибиторы обратной транскриптазы нуклеозида**, подавляющие фермент обратной транскриптазы ВИЧ и останавливающие синтез цепей ДНК. Они являются фосфолированными активными метаболитами;
- **Ингибиторы обратной транскриптазы нуклеотида**, имеющие те же свойства, однако не являющиеся фосфолированными;
- **Ингибиторы обратной транскриптазы ненуклеозида**, связывающиеся с ферментом обратной транскриптазы;
- **Ингибиторы протеазы**, подавляющие фермент вирусной протеазы, который важен для созревания вирионов ВИЧ;
- **Ингибиторы проникновения**, не допускающие прикрепления и проникновения вируса в лимфоциты.
- **Ингибиторы интегразы**, препятствующие интегрированию ДНК ВИЧ в ДНК человека.

Обычно для достижения наилучшего эффекта требуется принятие 2-4 классов препаратов. Также следует принимать во внимание возможные побочные эффекты, сопутствующие заболевания и возможное влияние в совокупности с другими принимаемыми препаратами.

### 1.2.1 Механизм проникновения ВИЧ в клетку

Рассмотрим подробнее процесс проникновения вируса в клетку и сами ингибиторы проникновения. Инфицирование и дальнейшее распространение ВИЧ приводит к формированию новых вирусных частиц, смерти здоровых клеток и в, конечном итоге, смерти организма. Вирусная частица ВИЧ содержит

белки gp120 и gp41 на своей поверхности, которые играют ключевую роль в связывании и дальнейшем проникновении вируса в здоровую клетку. Белок gp120 вместе с трансмембранный белком gp41 образует белок gp160. Сам же процесс проникновения состоит из следующих трех этапов, представленных на рисунке 1.1: прикрепления, взаимодействия, слияния.

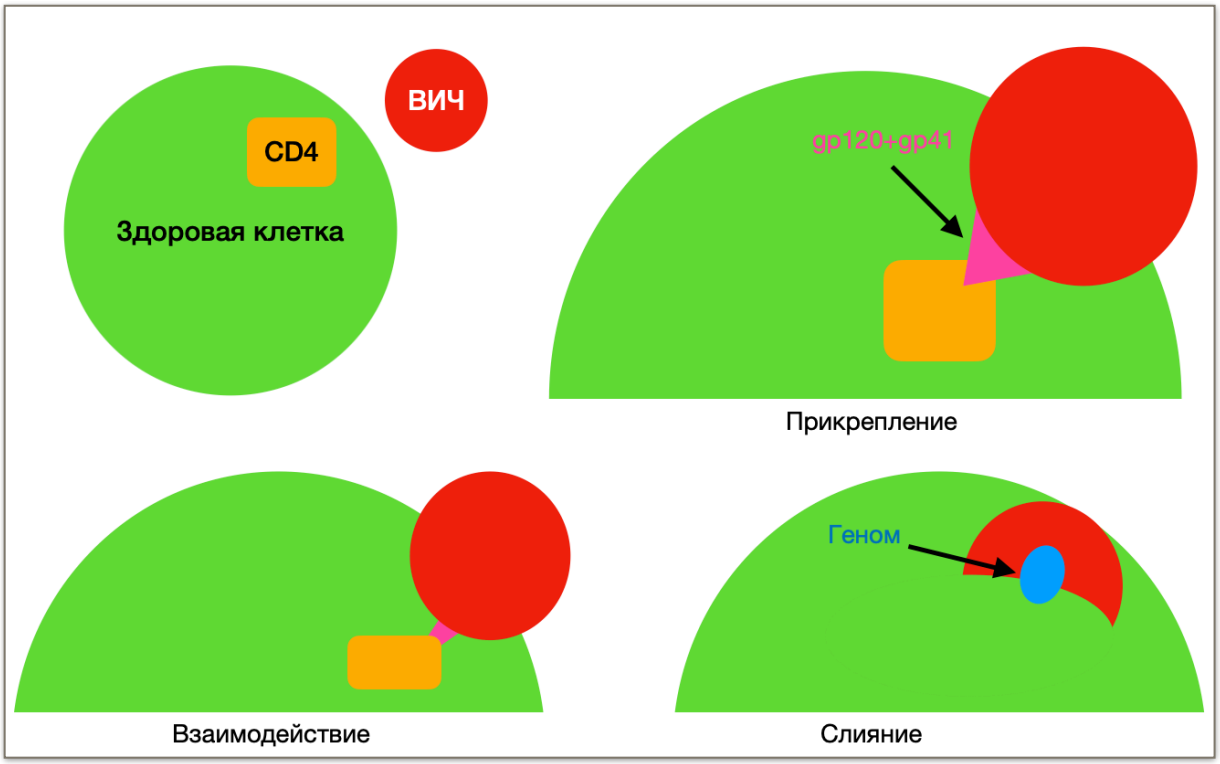



Рисунок 1.1 - Процесс проникновения вируса в клетку

Так как связывание ВИЧ частицы и здоровой клетки происходит с помощью белка gp120, этот белок является ключевым для поиска ингибиторов проникновения. Для присоединения частицы ВИЧ к клетке, белок gp120 должен поменять свою конформацию. После изменения конформации, белок gp120 связывается с рецептором CD4 и ко-рецептором CXCR-4, который находится в основном на Т клетках, или с ко-рецептором CCR5, который находится на Т-клетках, макрофагах, моноцитах и дендритных клетках. Эти ко-рецепторы настолько важны, что некоторые люди с мутациями в CCR5 фактически имеют иммунитет к ВИЧ, так как вирус не может прикрепиться и попасть внутрь клетки. Даже те мутации, которые ведут лишь к уменьшению количества ко-рецепторов, могут значительно замедлить развитие болезни. Далее после связывания белка gp120, уже другой белок, белок gp41 инъецируется в здоровую клетку. После этого в процессе взаимодействия происходит слияние клетки и частицы и из последней в цитоплазму здоровой клетки попадает вирусный геном. Клетка заражена. Стоит заметить, что ВИЧ это одноцепочечный РНК ретровирус. Ретровирус означает то, что вирусу

необходимо использовать фермент обратной транскриптазы для транскрипции дополнительной провирусной ДНК, которая в итоге интегрируется в ДНК здоровой клетки.

### 1.2.2. Ингибиторы проникновения

Ингибиторы проникновения предназначены для недопущения трех этапов, описанных выше. Они связываются с белком gp120 или gp41 или ко-рецепторами и далее блокируют их работу, тем самым останавливая процесс проникновения. Например, ингибиторы проникновения, действующие на белок gp120, связываются с ним и исключают связывание белка с рецептором CD4.

Исходя из механизма проникновения вируса в клетку, существуют 3 группы медикаментов, объединенных в класс ингибиторов проникновения. В этот класс входят:

- Ингибиторы прикрепления, действующие на белок gp120;
- Ингибиторы, блокирующие ко-рецепторы;
- Ингибиторы слияния, действующие на белок gp41.

На рисунке 1.2. Представлен пример ингибитора слияния — энфувиртид, имеющий торговое название Фузеон.

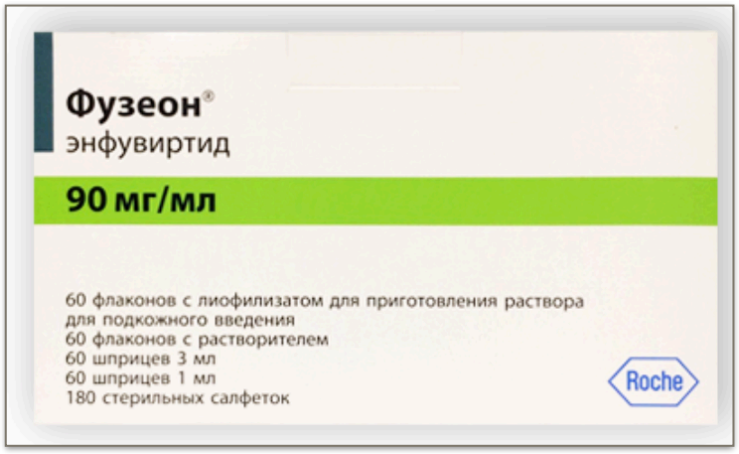



Рисунок 1.2 - Противовирусное (ВИЧ) средство

Таким образом ингибиторы проникновения блокируют первый шаг в заражении клетки — прикрепление и внедрение вируса в клетку, в отличии от других ингибиторов, действующих уже после прикрепления и слияния вируса ВИЧ с клеткой.

## 1.3. Вакцина от ВИЧ

Вирус отличен частыми ошибками при репликации, потому он может мутировать, создавая новые штаммы вируса и поражать новые клетки. Именно из-за мутаций в организме возникает, например, X4 штамм ВИЧ, поражающий только Т клетки. Склонность к мутациям является огромным препятствием для разработки вакцины от ВИЧ. Также наличие сахарной оболочки у вируса мешает и блокирует действие вакцин.

На данный момент единственной вакциной, показавшей хотя бы какой-то достойный результат, является вакцина RV144. Помимо того, что она имеет 30-процентный уровень эффективности, RV144 активна всего лишь несколько месяцев. В 2020 году из-за отсутствия перспектив развития разработка вакцины была остановлена. На данный момент разрабатываются другие вакцины, например вакцины MOSAICO и IMBOKODO. Эти вакцины пока что проходят стадию клинических испытаний.

Даже при успешном создании вакцины от ВИЧ следует помнить тот факт, что вакцины могут не подойти всем по ряду причин: медицинским показаниям, цене, личным убеждениям и так далее. Поэтому помимо разработки вакцин требуется разработка эффективной химиотерапии от ВИЧ.

## 1.4 Выводы

Вирус иммунодефицита человека является одним из опаснейших на текущий момент вирусов. Опасность вируса характеризуется в первую очередь тем, что пока еще не придумано способа полностью излечить эту смертельную болезнь. Существующая антиретровирусная терапия способна приблизить жизнь зараженного ВИЧ к жизни здорового человека, однако ежедневное принятие ряда лекарств вызывает как простейшие неудобства, так и серьёзные побочные эффекты. Даже такие прорывы в лечении ВИЧ, как рождение здоровых детей у ВИЧ инфицированных, конечно же восхитительны, однако не стоит останавливаться на достигнутом. Все это в совокупности требуют только упорного продолжения исследования ВИЧ и поиска новых ингибиторов для полной победы над этим вирусом.

# Глава 2. МЕТОДЫ МАШИННОГО ОБУЧЕНИЯ ДЛЯ ИДЕНТИФИКАЦИИ ПОТЕНЦИАЛЬНЫХ ИНГИБИТОРОВ

Существует множество повседневных задач, которые невозможно решить применяя традиционные методы программирования. Например, разработка бота для игры в го, задание поведения беспилотных транспортных средств и распознавание сложных изображений требуют нетривиальные методы и подходы. Однако, эти задачи можно успешно решить применяя так называемое машинное обучение, то есть без явного задания алгоритма действий. Том М. Митчел определил класс алгоритмов машинного обучения как класс программ, удовлетворяющих следующему определению: “Программа обучается из опыта $E$ по отношению к некоторому классу задач $T$ и мере представления $P$, если ее представление $P$ по задаче из класса $T$ улучшается с приобретением опыта $E$”. Как правило, для тренировки или обучения компьютера как играть в игру или управлять транспортным средством требуется некоторый набор данных или опыт $E$.

## 2.1 Нейронные сети

В 1957 году Френк Розенблатт предложил кибернетическую модель мозга. При разработке такой модели Розенблатт исходил из самой структуры мозга, представленной соединенными между собой нейронами. Эта модель получила название перцептрон Розенблатта и на данный момент она является простейшей моделью нейронной сети.

Спустя 50 лет, в марте 2006 года, произошло по-настоящему легендарное в мире нейронных сетей событие. Чемпион мира, лучший представитель человечества по игре в Го Ли Седоль проиграл нейронной сети AlphaGo со счетом 4:1. В этой игре невозможно просчитать следующие ходы, как это делали алгоритмы игры в шахматы, так как число вариантов запредельно. Считалось, что только благодаря интуиции профессионалы могут делать ходы и побеждать. Всего за 50 лет было предложено множество архитектур, классов и видов нейронных сетей, которые обучались с учителем и без.

Один из классов нейронных сетей, сети с прямым распространением, был неспособен анализировать временные последовательности, например звуковой сигнал, текстовую информацию и т. д. Во всех этих примерах необходимо знание предыдущих элементов последовательности. Для анализа таких данных был предложен другой класс нейронных сетей, а именно RNN — Recurrent Neural Network или рекуррентные нейронные сети. Первой сетью такого типа

являлась сеть Хопфилда, а первой современной рекуррентной сетью является сеть Джеффа Элмана, на основе которой создаются SimpleRNN или простейшие рекуррентные нейронные сети.

На рисунке 2.1 представлен пример архитектуры нейронной сети прямого распространения. Прямое распространение обозначает, что входной сигнал распространяется от входа к выходу не имя никаких обратных связей на своем пути. Слой обозначенный голубым цветом называется входным. По сути он ничего не делает, кроме того, как принимает входной сигнал $\overline{x}$ и передает его на следующий слой. Последний слой называется выходным, он выдает сигнал $\overline{y}$. Если между входным слоем и выходным есть какие-то слои, то они называются скрытыми. Стоит заметить, что иногда входной слой относят к первому скрытому слою. Далее будем считать входной слой отдельно от первого скрытого слоя.

Связи между нейронами имеют определенные веса, которые далее будем обозначать как $\omega_{ij}$ — от $j$-го к $i$-му нейрону. Когда сигнал проходит по определенной связи, например входной сигнал с $i$-го входного нейрона на $j$-й нейрон первого скрытого слоя нашей сети, он умножается на вес $\omega_{ij}$, то есть получается $\omega_{ij}x_i$. На рисунке 2.2 подробно представлен нейрон сети. Здесь к нейрону походит некоторый сигнал $\overline{x}$, а на выходе получается сигнал $y$, который далее передается всем нейронам, с которым связан текущий нейрон. Вектор $\overline{x}$, поступающий на вход нейрона умножается скалярно на вектор весовых

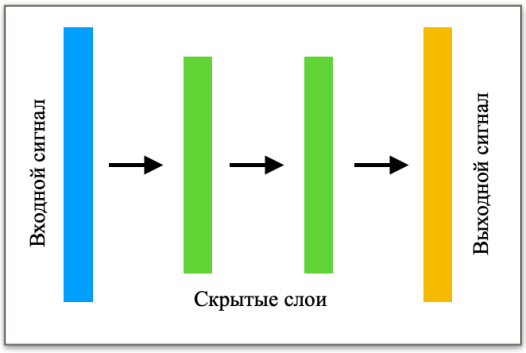



Рисунок 2.1 - Пример архитектуры нейронной сети

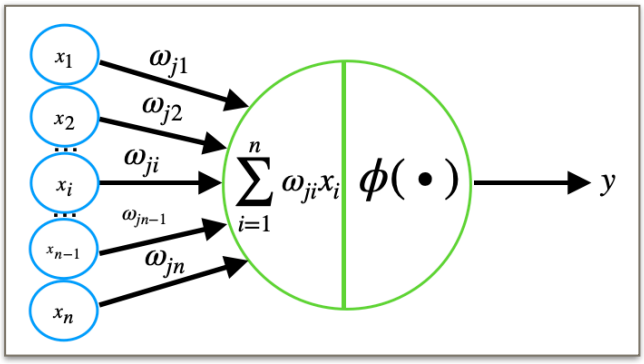



Рисунок 2.2 - Структура нейрона

коэффициентов $\overline{\omega}$, получается сумма $\sum \omega_{ji}x_i$. После чего эта сумма опережается на вход функции $\phi$ или функции активации нейрона. Ниже в формулах (2.1), (2.2) и (2.3) представлены функции активации, которые активно используются в ходе данной работы: ReLU, Leaky ReLU и Sigmoid.

$$\phi(s) = \max(0,s) \qquad \text{ReLU} \qquad (2.1)$$

$$\phi(s) = \max(0.1s, s) \qquad \text{Leaky ReLU} \qquad (2.2)$$

$$\phi(s) = \max(0,s) \qquad \text{Sigmoid} \qquad (2.3)$$

где $s = \sum \omega_{ji} x_i$.

### 2.1.1 Рекуррентные сети

Недостатком сетей прямого распространений, то есть сетей, в которых входной сигнал поступает на вход и, проходя сеть, на выходе преобразовывается в выходной сигнал, является невозможность анализа временных последовательностей: аудиосигнала, текста. Причиной этому является то, что во временных последовательностях важно знание предыдущих элементов последовательностей. Для анализа таких данных был разработан специальный класс нейронных сетей — рекуррентные нейронные сети (RNN). Как было замечено ранее, первой сетью такого вида была сеть Хопфилда, а первой современной рекуррентной сетью стала сеть Элмана или SimpleRNN, представленная на рисунке 2.3. Входной сигнал в данной сети подается на рекуррентный слой, выходы которого в свою очередь снова подаются на

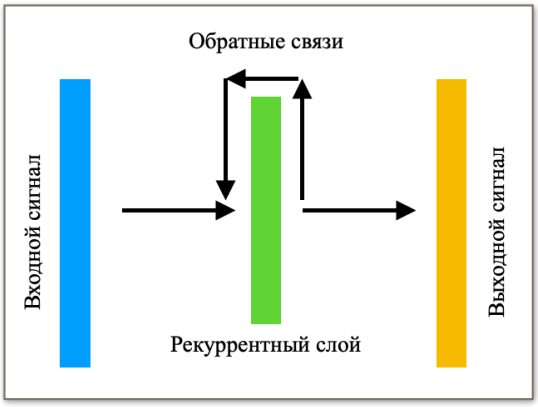



Рисунок 2.3 - Сеть Элмана

рекуррентный слой с некоторый задержкой. Математически же данная сеть выглядит как:

$$\overline{h}_t = \phi(\overline{h}_{t-1}, \overline{x}_t) \qquad (2.4)$$

$\overline{h}_t$ - текущее состояние рекуррентной сети на шаге $t$;

$\overline{h}_{t-1}$ - предыдущее состояние рекуррентной сети;

$\phi$ - функция активации рекуррентного слоя;

$\overline{x}_t$ - входной сигнал на шаге $t$.

Все эти условия незначительно меняют идею градиентного спуска для корректировки весовых коэффициентов. Модифицированный алгоритм

обучения рекуррентных сетей называется Backpropagation Through Time (BPTT).

Процесс обучения рекуррентной нейронной сети схож с процессом обучения нейронной сети прямого распространения. Главная проблема — сохранение устойчивости и при обучении, и при работе с сетью. Задачей сохранения стабильности и устойчивости нейронной сети занимается специальная область — нейродинамика. Устойчивость часто рассматривается как устойчивость пл Ляпунова. Интересен тот факт, что работа Ляпунова по устойчивости систем была написана в 1892 году.

### 2.1.2 LSTM сети

Архитектура SimpleRNN, простейших реккурентных нейронных сетей, — не лучшая архитектура. Основной недостаток таких сетей — быстрое “забывание” прошлого контекста, а иногда он может быть важен. И хотя теоретически можно настроить весовые коэффициенты сети так, чтобы SimpleRNN давала хорошие результаты на сложных примерах, в которых важен прошлый контекст, на практике этого достичь пока не удается.

В 1997 году Зеппером Хохрайтером и Юргеном Шмидхубером была предложена архитектура нейронной сети, использующая LSTM — Long Short-Term Memory, что переводится как долгая краткосрочная память. Такая архитектура при обучении способна схватывать существенные детали прошлого контекста и сохранять их, пока они актуальны. LSTM сети остается востребованными до сих пор.

Рассмотрим работу классической LSTM ячейки на некотором шаге $t$. На рисунке 2.2 представлена сама ячейка LSTM, а в таблице 2.1 пояснения к обозначениям рисунка 2.2. На вход каждой такой ячейки подается вектор $X_t$, а на выходе получается прогнозный вектор или вектор скрытого состояния $H_t$ . Этот вектор передается на вход следущей ячейки и так далее. Но помимо этого у LSTM есть еще верхний канал, на вход которого подается вектор $C_{t-1}$, а на выходе получается вектор $C_t$. Именно этот канал позволяет сохранять важный долгосрочный контекст и именно он принципиально отличает ячейку LSTM от SimpleRNN.

Таким образом с входным вектором $C_{t-1}$ в верхнем канале производятся две операции: поэлементное умножение и поэлементное сложение. Умножение отвечает за “забывание” ненужного, а сложение — за “запоминание” нового.

Разберем почему так происходит и начнем с “забывания”. На вход верхнего канала ячейки поступает вектор $C_{t-1}$, содержащий долгосрочный

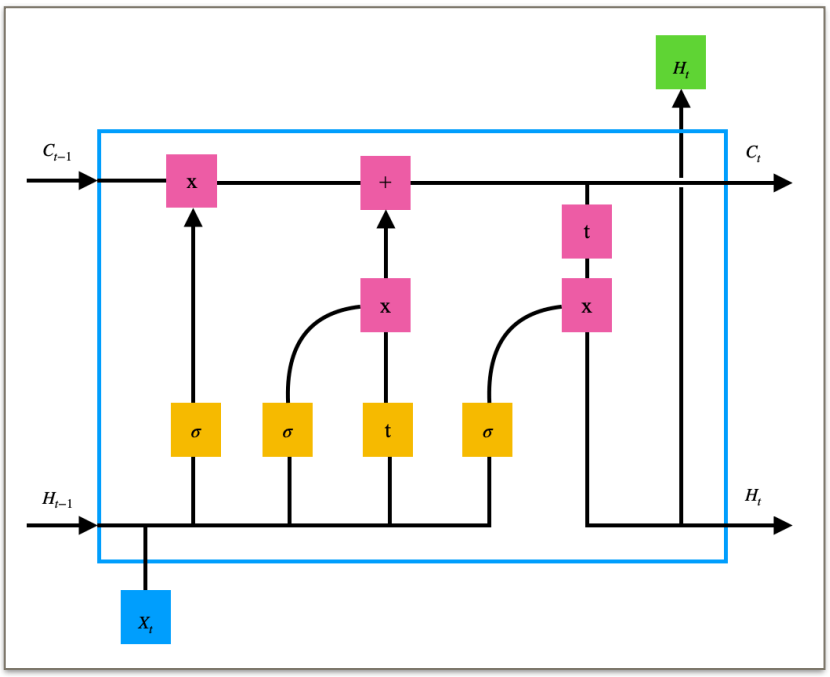



Рисунок 2.2 - LSTM ячейка

контекст, пришедший с предыдущего рекуррентного шага. Помимо этого имеется предыдущее состояние LSTM ячейки $H_{t-1}$, передающееся по нижнему каналу, которое объединяясь с входным вектором $X_t$, подается на полносвязный сигмоидальный слой. В итоге выход этого слоя будет представлять собой вектор $F_{1t}$, содержащий значения от 0 до 1 (в силу функции активации). Эти числа воспринимаются как некоторые веса, которые сигнализируют что следует “забыть”, а что следует “оставить” в векторе $C_{t-1}$.

Аналогично происходит с “запоминанием” нового. При прохождении полносвязного сигмоидального слоя и слоя с функцией активации гиперболический тангенс вектором $H_{t-1}$, $X_t$ получается два вектора $F_{2t}$ и $F_{3t}$ соответсвенно. Вектор $F_{2t}$ отвечает за то, что следует “забыть” или “оставить” в векторе $F_{3t}$. В свою очередь вектор $F_{3t}$ содержит в себе новый важный контент. Далее результат поэлементного умножения этих векторов суммируется с вектором $C_{t-1}$, в и результате получается вектор $C_t$.

Именно в процессе обучения сети подбираются весовые коэффициенты полносвязных слоев так, чтобы сеть в среднем корректно оставляла или запоминало новое.

Рассмотрим последний этап работы LSTM ячейки — формирование скрытого состояния $H_t$. На вход полносвязного сигмоидального слоя опять же подается вектор $H_{t-1}$, $X_t$, а на выходе получается вектор $F_{4t}$. Далее вектор $F_{4t}$ поэлементно умножается на нормированный (в ходе поэлементного взятия

Пояснения к рисунку 2.2:

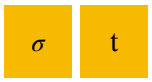
| Обозначение | Значение |
|---|---|
|  | Входной вектор на шаге $t$ |
|  | Вектор скрытого состояния на шаге $t$ |
|  | Полносвязный слой нейронной сети с сигмоидальной функцией активации и гиперболическим тангенсом соответственно |
|  | Поэлементное умножение, сложение, взятие гиперболического тангенса соответственно |
|  | Направление движения вектора |
|  | Объединение двух векторов |
| | Копирование вектора |

Таблица 2.1

гиперболического тангенса) вектор $C_t$ и в итоге получается вектор скрытого состояния $H_t$.

## 2.2 Специальные архитектуры нейронных сетей

Существует множество архитектур нейронных сетей, подходящих для генерации новых соединений, но способов их обучить два — с учителем и без. При обучении с учителем требуется набор размеченных данных, в то время как без учителя — неразмеченных.

Использование генератора основанного на нейронных сетях с обучением с учителем затруднено из за того, что для обучения такого генератора, правильно понимающего механизмы воспроизведения соединений, потребуется набор из тысячи размеченных молекул. Учитывая тот факт, что разметка ведется вручную, это не представляется возможным. Стоит заметить, что вариант с предобучением таких нейронных сетей дает некоторый результат, но этого недостаточно.

Другой подход — разработка автоэнкодера, нейронной сети, которая копирует входные данные и обучается без учителя, то есть обучается на неразмеченных данных. Найти неразмеченные данные, как правило, не составляет труда. Основная идея автоэнкодера в том, чтобы закодировать входное соединение в некотором формате в скрытое пространство (latent space), а затем, используя это скрытое пространство, восстанавливается вход. После

обучения автоэнкодер может быть разбит на две части: энкодер и декодер. Декодер же в свою очередь может использоваться для генерации новых молекул.

Однако, существует еще один способ обучения — обучение с частичным привлечением учителя. Этот способ использует комбинацию обучения с учителем и без и для обучения использует как размеченные, так и неразмеченные данные.

### 2.2.1 Автоэнкодеры

Параллельно с различными классами нейронных сетей развивались и их архитектуры. Пример одной из них — автоэнкодер, представляющий собой специальную нейронную сеть, которая кодирует входной сигнал в некоторый вектор скрытого состояния, размерность которого, как правило, меньше размерности входного сигнала, а затем из этого скрытого состояния обратно декодирует в новое состояние, формируя выходной сигнал. Пример архитектуры автоэнкодера представлен на рисунке 2.1. Размерности входных и выходных слоев могут отличатся, например, можно обучить автоэнкодер масштабировать изображения. Также автоэнкодеры могут использоваться в сжатии изображения, хотя на данный момент традиционные методы сжатия работают лучше.

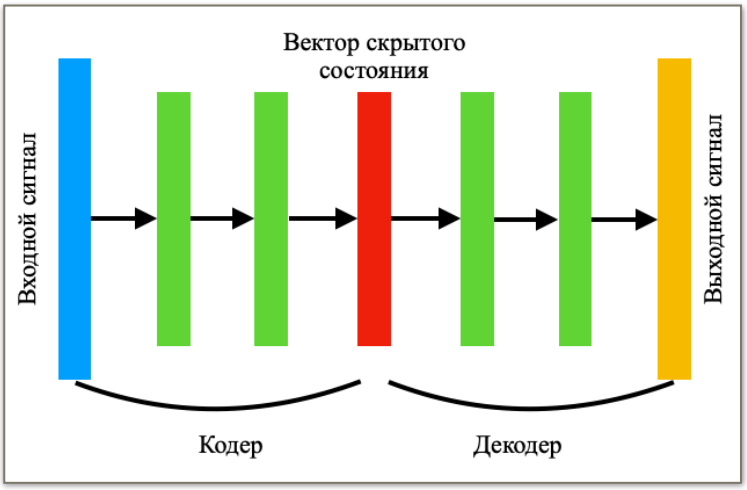



Рисунок 2.1 - Архитектура автоэнкодера

Вектор скрытого состояния или латентный вектор автоэнкодера описывает некоторую модель представления входных данных. Чем точнее входные значения, тем точнее декодер их восстановит.

Рассмотренная выше общая архитектура автоэнкодера она имеет большой недостаток — неизвестное распределение вектора скрытого состояния. На рисунках 2.2 представлены примеры хорошего и плохого распределения автоэнкодера. Здесь каждый цвет представляет определенный класс. Контроль

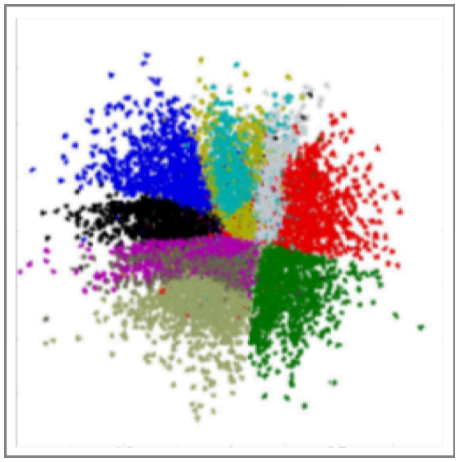


Рисунок - 2.2 Пример хорошего распределения

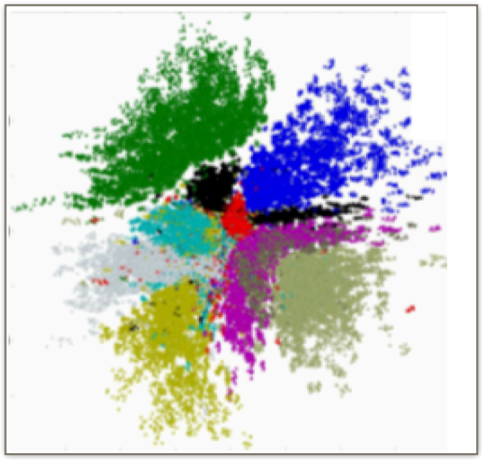


Рисунок - 2.2 Пример плохого распределения

распределения вектора скрытого состояния осуществляется в архитектуре состязательных автоэнкодеров.

Состязательные автоэнкодеры похожи на классические автоэнкодеры, но имеют отличие в части кодера. Кодер обучается таким образом, чтобы распределение вектора скрытого состояния соответствовало заранее заданному распределению. Для достижения вышеупомянутого свойства потребуется дискриминатор — часть генеративное-состязательной сети.

### 2.2.2 Генеративно-состязательные сети

Генеративно-состязательные сети или ГСС используются повсеместно, например, для генерации изображений, текстовой информации, перевода изображения в другое изображение (окрашивание или выделение объектов). Такие сети на самом деле состоят из двух нейронных сетей: генератора и дискриминатора. Дискриминатор, простая сверточная нейронная сеть, обучается раздичать настоящие (real) и ложные (fake) данные. Настоящие данные выбираются из заранее подготовленного набора данных, в то время как ложные генерируются генератором. Как правило, на вход генератора подается шум или вектор случайных значений, а на выходе получается некоторый объект.

Таким образом, в рамках одной ГСС соревнуются дискриминатор и генератор: генератор пытается обмануть дискриминатор, генерируя ложные данные, а дискриминатор, в свою очередь, пытается отличить ложные данные генератора от настоящих данных из заранее подготовленного набора данных. Благодаря этому соревновательному процессу генератор обучается производить такие данные, что дискриминатор не сможет отличить их от настоящих.

## 2.3 Выводы

В данной главе были рассмотрены основы нейронных сетей, необходимые для дальнейшей работы. В частности, были изучены сети прямого распространения, рекуррентные сети и, более подробно сети, с LSTM ячейками, были отмечены недостатки и достоинства таких сетей. Далее для разработки моделей нам также понадобятся специальные архитектуры нейронных сетей: автоэнкодеры и генеративное-состязательные сети. На этом рассмотрение нейронных сетей, естественно, не исчерпывается. В последующих главах еще не раз будем обращаться к теории и давать пояснения тем или иным нюансам разработанных нейронных сетей.

# ГЛАВА 3. РАЗРАБОТКА АЛГОРИТМА ИДЕНТИФИКАЦИИ

Задача алгоритма идентификации потенциальных ингибиторов — генерация веществ-потенциальных ингибиторы белка gp120 оболочки ВИЧ. Биохимическая сторона этой задачи была подробно описана в главе 1. Всего в текущей главе будут представлены два алгоритма — алгоритм без заранее заданной энергии связывания (алгоритм 1) и алгоритм с энергией связывания (алгоритм 2).

Хотя и оба алгоритма будут основаны на архитектуре LSTM автоэнкодеров, разобранных в предыдущей главе, различия между ними все же будут. Каждый алгоритм будет описан в соответствующем ему разделе, но предобработка данных для обоих алгоритмов будет одна.

## 3.1 Анализ исходных данных

Для дальнейшей разработки алгоритмов, а, следовательно, и обучения моделей автоэнкодеров будет необходим набор данных или датасет, содержащий записи потенциальных ингибиторов белка gp120, отвечающего за процесс прикрепления ВИЧ к здоровой клетке. В качестве формата записи выбран формат SMILES. Сами ингибиторы получены в результате молекулярно-динамического моделирования, проведенного при помощи программы AMBER. Молекулярная динамика позволяет определить как двигаются молекулы на атомном уровне и используется при изучении свойств материалов, а также в разработке лекарственных препаратов. Так как реальные эксперименты зачастую дороги и не всегда выполнимы, инструмент молекулярной динамики позволяет проводить эксперименты с молекулами в компьютере. Даже перед реальными экспериментами следует ряд молекулярно-динамических вычислений для анализа тех или иных свойств молекул.

На рисунке 3.1 представлен пример первых пяти записей файла с исходными данными. Всего файл содержит данные о 71413 соединений - потенциальных ингибиторов белка gp120. Каждое соединение имеет данные о структуре молекулы — столбцы smiles и smiles_len и о энергии связывания потенциального ингибитора с белком gp120 — столбец energy.

Поиск соединений с наименьшей энергией связывания является нашей основной целью. Чем меньше энергия связывания, тем прочнее связь ингибитор-белок, и, как следствие, тем ниже шанс частицы ВИЧ присоединиться к здоровой клетке.

### 3.1.1 Формат SMILES

В ходе данной работы нами активно будут пользоваться молекулы в формате SMILES, поэтому для начала рассмотрим базовые принципы системы SMILES-кодирования молекулярных структур.

| | energy | smiles | smiles_len |
|---|---|---|---|
| 0 | -8.1 | Cc1ccc(C(=O)CN2NN=C(C3CC3)[C@@H]2c2ccccc2F)cc1 | 46 |
| 1 | -8.0 | CCOC1=CC(=O)[C@@H](CC2=NNN(CC(=O)c3ccc(C)cc3)[... | 56 |
| 2 | -7.7 | Cc1ccc(C(=O)CN2NN=C(CNC(C)(C)CO)[C@H]2C(C)(C)C... | 50 |
| 3 | -7.1 | Cc1ccc(OC[C@H]2C(CN3CCOCC3)=NNN2CC(=O)c2ccc(C)... | 53 |
| 4 | -6.8 | Cc1ccc(C(=O)CN2NN=C(CCO)[C@@H]2C2(O)CC2)cc1 | 43 |

Рисунок 3.1 - Пример первых записей файла с исходными данными

SMILES — Simplified Molecular-Input Line-Entry Specification, что может переводится как упрощенная спецификация входных молекул в виде строки, это система правил однозначного описания состава и структуры молекул химического вещества с использованием стандартного кода ASCII.

Естественно, помимо SMILES-кодирования существуют и другие кодирования, например, InChI — International Chemical Identifier, что означает международный химический идентификатор. Данная система кодирования активно используется, однако предназначена для работы со специфическим программным обеспечением и ее представление не такое наглядное, как у SMILES. Для примера в таблице 3.1 приведем закодированную формулу этанола. Помимо наглядности SMILES, стоит отметить и краткость такого кодирования, что немало важно при работе с большими наборами данных.

Представление этанола в двух форматах:

| Формула | SMILES | InChI |
|---|---|---|
| $CH_3CH_2OH$ | $CCO$ | $InChI = 1S/C2H6O/c1-2-3/h3H,2H2,1H3$ |

Таблица 3.1

Перейдем к основным принципам кодирования молекул в кодировке SMILES. Запись молекулы в таком формате получается обходом вершин молекулярного графа в глубину. При этом органогенные атомы водорода, то есть атомы в связях $C-H$, $C-H$, $O-H$, $S-H$, убираются и далее не обозначаются. Это можно заметить в таблице 1.1 с примером записи этанола.

Следующее правило кодирования — замена циклов молекул на такие разъединенные циклы, которые представляют собой ацикличный связный подграф, сохраняющий все исходные вершины цикла. Места же разъединения обозначаются циклами. Например, соединение циклогескан, имеющие формулу $C_6H_{12}$ в SMILES будет представлено как $C1CCCCC1$.

Ответвления от основной цепи соединения обозначаются круглыми скобками, например триметиламин $N(CH_3)_3$ представляется как $CN(C)C$.
Если же в соединении присутствуют элементы, связанные ковалентный связью с углеродом, то они заключаются в квадратные скобки. Данное правило не распространяется на элементы $B$, $N$, $P$, $O$, $S$, $F$, $Cl$, $Br$, $I$. Например, метиллитий $LiCH_3$ — $[LI]C$, но триметилборан $B(CH_3)_3$ — $CB(C)C$.

Также наличие двойной связи или тройной обозначается как = или # соответственно, а атомы в составе ароматических соединений могут обозначаться с маленькой буквы. Изотопы атомов и ионы заключены в квадратные скобочки, например $^{13}CH_3$ — $[13C]$ и $H_3C^+$ — $[C+]$ соответственно.

Правила кодирования этим не ограничиваются. Существуют также и другие правила, например, правила кодирования конфигураций относительно двойной связи или правила кодирования конфигураций асимметрического тетраэдрического атома углерода. Однако такие правила используются реже, поэтому при необходимости будут рассмотрены на месте использования.

Таким образом, система кодирования SMILES является очень удобным инструментом для работы с молекулами. Такая система позволяет легко и безошибочно переводить входные молекулы в наглядные закодированные формулы. Стоит заметить, что SMILES используются в таких известных электронных базах химических соединений, как PubChem и ChemSpider.

## 3.2 Предобработка входных данных

Как ранее было указано, в качестве входных данных для обучения и тестирования наших моделей у нас имеется набор данных потенциальных ингибиторов белка gp120 оболочки ВИЧ.

Перед тем, как обучать модель, необходимо предобработать эти данные. Обработка и дальнейшая реализация модели будет проходить в Google Colab на языке Python. Конкретно для предобработки данных были использованы библиоткеки Numpy, Pandas, Scikit-learn и Matplotlib.

Уберем ненужную информацию из набора данных. Распределение длин представлено на рисунке 3.2. Здесь по оси x расположены длины, а по оси y — количество SMILES-формул соответствующей длины.

Можно заметить, что SMILES-формулы распределены в основном от 30 до 75 символов и распределение напоминает нормальное распределение. Обрезать излишне длинные или короткие формулы не будем.

Уберем из датасета сложные соединения, например соединения, содержащие в SMILES-формулах знаки ‘%’, сигнализирующий о наличии множественной закрывающей кольцо связи, символы ‘p’ — о наличии ароматического кольца с фосфором, символы ‘.’ — о наличии пары, которая не связана и символы '0', '7', '8', ‘9' — о наличии кольца больших размеров.

Помимо сложных соединений имеет смысл убрать из исходного набора данных формулы имеющее редкоиспользуемые символы. На рисунке 3.3 представлена диаграмма со значением символа по оси y и, соответсвенно, по оси x с числом вхождений этого символа в SMILES-формулы исходного набора данных. Для наглядности особо часто встречающиеся символы, например C с 1 и тд, в диаграмму не включены. Уберем из набора данных соединения имеющие символы с числом вхождения меньше 1000, то есть символы ‘I’, ‘5’, ‘.’, ‘P’ и символ ‘6’.

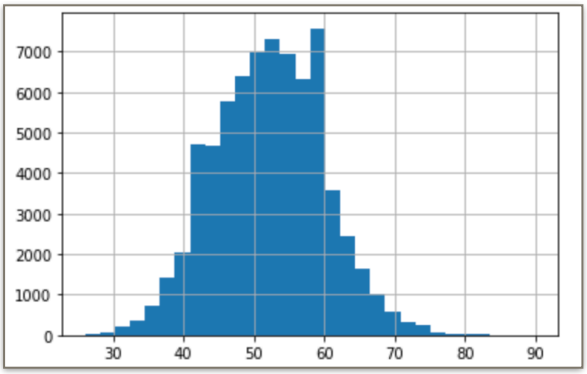



Рисунок 3.2 - Распределение длин SMILES формул

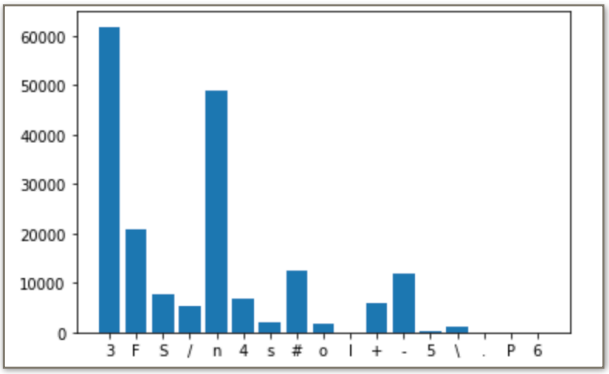



Рисунок 3.3 - Частота вхождений символов

Естественно, после такой предобработки генерировать довольно сложные соединения не удастся, но зато точность генерации более распространенных и простых соединений повысится.

Далее разделим наблы данных на несколько выборок следующим образом. Датасет делится в пропорции 4:1 и получается тренировочная и тестовая выборка соответственно. Далее тренировочная выборка делится в той же пропорции на тренировочную и валидационную соответственно. Того

получается 3 выборки: тренировочная, валидационная и тестовая. Тренировочная выборка используется для обучения модели, валидационная — для оценки гиперпараметров, тестовая — для оценки ошибки модели.

Таким образом после предобработки данных у нас имеется датасет потенциальных ингибиторов белка gp120. Всего осталось 71284 потенциальных ингибитора. Данный датасет разделен на три выборки — тренировочную, тестовую и валидационную.

### 3.2.1 One-Hot кодировка

Так как нейронные сети плохо и иногда неожиданно работают с предсказанием простых символов, следует каждую SMILES-формулу сначала перевести в one-hot код. Такой код строится следующим образом: сначала для всего датасета находится множество уникальных символов из которых формируются SMILES-строки и его мощность. Далее каждый уникальный символ пронумеровывается от 0 до мощности этого множества минус 1. Тогда каждому символу SMILES-формулы ставится в соответствие двоичный вектор длины равной мощности множества уникальных символов такой, что все элементы вектора равны 0, кроме элемента с порядковым номером, соответствующим данному символу.

Длины SMILES-формул различны для каждого соединения, а на вход автоэнкодера должны подаваться входные данные одинаковой размерности. Поэтому помимо простого one-hot кодирования SMILES-формулы, такая формула кодируются и во множество уникальных символов добавляется еще и ‘!’ — символ, сигнализирующий о начале входной последовательности, и ‘E’ — символ, заполняющий пустое пространство и сигнализирующий о конце формулы. Обозначим размер множества уникальных символов как $nUnique$. Длинна входной последовательности $nEntry$ выбирается как длинна максимальной по длине SMILES-формулы плюс несколько запасных символов. Пример преобразованной SMILES-формулы в one-hot код представлен на рисунке 3.4. Здесь по оси x находятся порядковые номера символов SMILES-формулы, а по оси y — двоичные векторы с 1 в нужном месте.

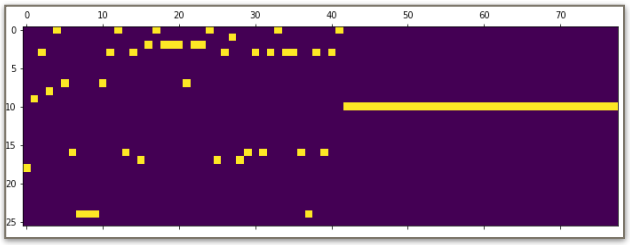



Рисунок 3.4 - Пример one-hot кода

Далее преобразуем датасет, полученный в пункте 3.2, и включенный в него тренировочную, тестовую и валидационную выборки в выходной и выходной наборы. Входной набор — one-hot закодированная выборка SMILES-формул начиная с первого символа до предпоследнего, в то время как выходной набор — также one-hot закодированная выборка SMILES-формул, но начиная со второго символа до последнего. Таким образом входной и выходной наборы сдвинуты на один символ относительно друг друга. Именно входной набор подается на входной слой автоэнкодера, а выходной — на выходной слой. Так автоэнкодер пытается восстановить по предыдущему символу следующий.

Как во время one-hot кодирования, так и после нам понадобятся два словаря: *charToIntDict* и *intToCharDict* — соотносящий каждому символу позицию единицы в двоичном векторе и наоборот соответсвенно.

## 3.3 Алгоритм идентификации без энергии связывания

Приступим к описанию первого алгоритма идентификации потенциальных ингибиторов белка gp120. Данный алгоритм не будет никаким образом учитывать энергию связывания соединений из исходного набора данных. Как и было указано ранее, алгоритм будет основан на специальной архитектуре нейронной сети — автоэнкодере.

### 3.3.1 Идея алгоритма

Общая идея алгоритма заключается в следующем:

1) Вещество-потенциальный ингибитор gp120, представленный в формате SMILES кодируется в латентный вектор;
2) В латентный вектор вносится некоторый шум;
3) Далее латентный вектор с шумом декодируется и получается новое соединение, полученное из исходного

Для кодирования-декодирования соединений в формате SMILES будет использована модель нейронной сети — LSTM автоэнкодера.

### 3.3.2 Выбор модели

Принимая во внимание минусы простейших реккурентных нейронных сетей и плюсы сетей с LSTM ячейками, рассмотренные в предыдущей главе, опишем следующую модель автоэнкодера, представленную на рисунке 3.5.

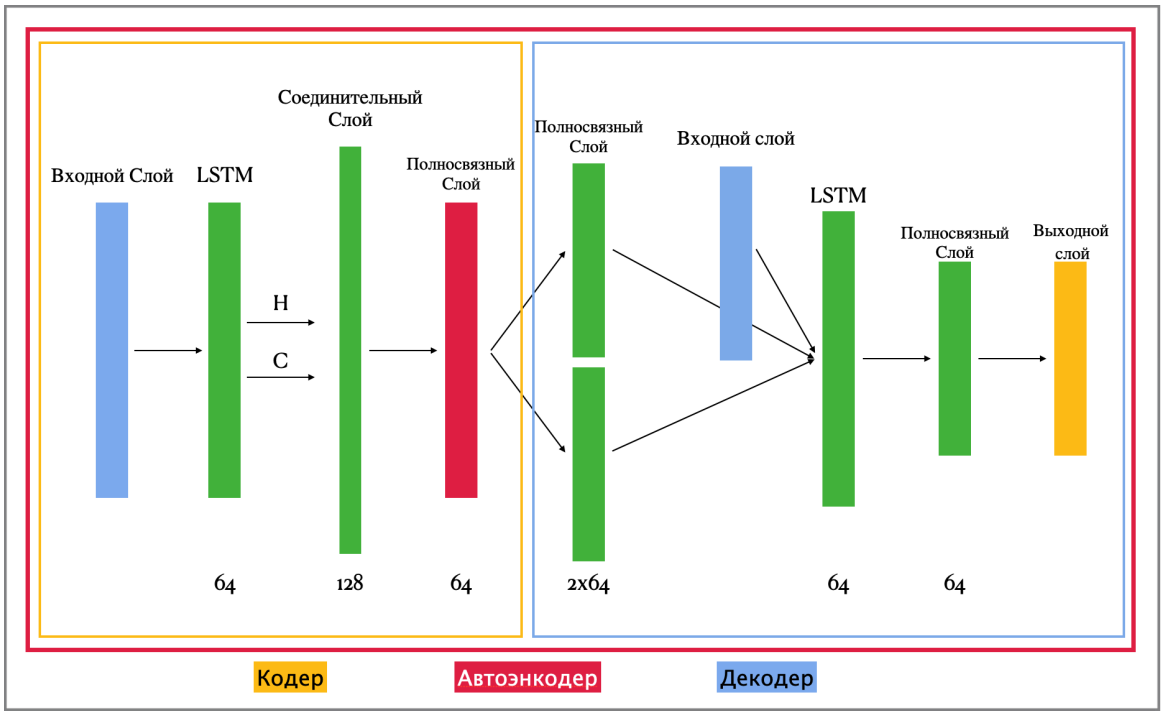



Рисунок 3.5 - Модель автоэнкодера алгоритма 1

На вход автоэнкодера подается вектор некоторой размерности, соответствующей one-hot представлению SMILES строки. Далее этот вектор проходит LSTM слой размерностью 64. Выход данного слоя опускается, далее используются только $H$ и $C$ векторы, которые конкатенируют или соединяют в один вектор размерности 128 на соединительном слое и подают на вход полносвязного слоя размерности 64. Этот слой будет представлять собой информативный эмбеддинг молекулы. Для декодирования эмбеддинга требуется подать его на два различных полносвязных слоя размерности 64 каждый, таким образом восстанавливая векторы $H$ и $C$. После этого следующий слой, состоящий из LSTM ячеек получает на вход восстановленные векторы $H$ и $C$ и еще один входной слой, размерность которого также соответсвует размерности one-hot SMILES-формулы. Выход данного LSTM слоя будет передан полносвязному слою размерности 64, выход которого будет служить выходом автоэнкодера.

Важно понимать размерности входов каждого слоя. Если для входного слоя это (none,$nEntry$,$nUnique$), где none — это размер минибатча (то есть он может быть равен 1 или 2, 3 и тд), то для следующего за ним LSTM слоя это будет также (none,$nEntry$,$nUnique$). Здесь величины $nEntry$ и $nUnique$ относятся к ранее определенным величинам из раздела 3.2.1. Но применимо ко второму слою, но для первого слоя дело обстоит иначе. Первый LSTM слой должен не должен возвращать последовательность.

### 3.3.3 Настройка гиперпараметров и процесс обучения

Для реализации модели автоэнкодера используем библиотеку Keras. После описания модели требуется правильно выбрать гиперпараметры обучающего алгоритма. Для обучения модели используется алгоритм градиентного спуска.

В качестве оптимизатора выбран Adam — Adaptive moment estimation с параметром скорость обучения равным 0.008. Вообще существует много разных оптимизаторов, элементов нейронной сети, которые определяют как обучаются нейронные сети. Например, простейший оптимизатор — градиентный спуск, имеет вид:

$$\theta = \theta - \alpha \bigtriangleup_\theta J(\theta) \qquad (3.1)$$

где $\theta$ - параметры, $\alpha$ - некоторый вес. Важнейший минус этого алгоритма это то, что веса обновляются только один раз за просмотр всего набора данных. Чаще всего якобиан $\bigtriangleup_\theta J(\theta)$ большой и в совокупности с даже с весом $\alpha$ алгоритм не может сойтись. Если же выбрать $\alpha$, алгоритм может сходиться очень долго. Один из выходов — это обновлять веса чаще, что и предлагает алгоритм стохастического градиентного спуска или SGD, обновляя веса раз в эпоху. Но и это не панацея, стохастический градиентный спуск способен делать непредсказуемые "прыжки" от искомого локального минимума. В итоге в алгоритм SGD были добавлены моменты, ускорение. Одним из популярнейших алгоритмов SGD с вышеперечисленными улучшениями как раз и является Adam (3.2):

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t \quad (3.2)$$

где $\hat{m}_t$ и $\hat{v}_t$ — скорректированые значения скользящих средних предыдущего значения градиента $m_t$ и его квадрата соответсвенно $v_t$:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \qquad (3.3)$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \qquad (3.4)$$

Таким образом параметры $\alpha$ — скорость обучения, $\epsilon$, $\beta_1$ и $\beta_2$ выбираются заранее. В нашем случае это значения 0.008, 10E-8, 0.9 и 0.999 соответсвенно.

Также для оптимизации процесса обучения используем ReducrLROnPlateau. Данный функционал Keras позволяет нам понижать параметр $\alpha$, когда наблюдаемая метрика (в нашем случае потери на валидационной выборке) перестала улучшаться. Вызов данного функционала включает в себя указание метрики и минимального значения параметра $\alpha$:

ReduceLROnPlateau(monitor='val_loss', … , min_lr=0.00001)

Обучение будет проходить в 80 эпох с размерами мини-батчей равными 256.

В качестве функции потерь выберем перекрестную энтропию. По сути перед нами стоит задача классификации с классами — уникальными символами входных SMILES-формул. Выход автоэнкодера, представляет собой некоторые вероятностные характеристики принадлежности тому или иному классу. Заметим, не вероятности, а вероятностные характеристики. Перекрестная энтропия имеет вид:

$$H(p,q) = -\sum_x p(x) ln q(x) \qquad (3.5)$$

где $p(x)$ истинное распределение вероятностей, а $q(x)$ — прогнозируемое. То есть в нашем случае, например, если символ ‘C’ представлен вектором $p = [0,0,...,1,...0,0]$, а предсказанный символ пусть имеет вектор $q = [0.01,0.04,...,0.92,...0.02,0.04]$, то в этом случае потери составят $H = 0 * ln(0.01) + 0 * ln(0.04) + \ldots + 1 * ln(0.92) + \ldots + 0 * ln(0.02) + 0 * ln(0.04)$

Теперь можно перейти к обучению модели. В таблице 3.1 представлено время обучения модели.

Время обучения модели:

| Входной датасет | Кол-во строк | Время обучения, мин. | Число параметров |
|---|---|---|---|
| Ингибиторы gp120 | 45688 | 7 | 65435 |

Таблица 3.1

При обучении моделей был использован GPU — Graphics Process Unit, предоставляемый для использования в Google Colab. Если бы модель обучалась на CPU, время обучения было бы в разы больше.

Вызов метода для компиляции модели имеет вид:

model = Model([encoder_inputs, decoder_inputs], decoder_outputs)

где encoder_inputs и decoder_inputs — входные слои со SMILES-формулами, а decoder_outputs — выходной слой с one-hot закодированным символом.

### 3.3.4 Оценка точности модели

Оценим ошибки обученной модели. На рисунке 3.6 представлена точность модели в зависимости от эпох. По оси x указан номер эпохи, по оси y — ошибка.

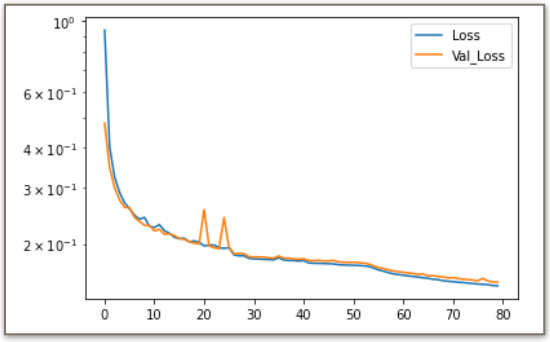



Рисунок 3.6 - Точность модели

Переобучение возникает, когда нейронная сеть адаптируется к тренировочной выборке, а не к общим закономерностям или логике построения данных. Можно заметить, что график потерь на тестовой выборке, обозначенный синим цветом, и график потерь на валидационной выборке, обозначенный желтым цветом, не расходятся, что может говорить об отсутствии переобучения выбранной модели. Ниже, в таблице 3.2 представлены конечные значения ошибок на трех выборках соответсвенно. Заметим хорошие значения

Полученная модели автоэнкодера имеет следующие ошибки:

| Входной датасет | Потери на трен. выборке | Потери на вал. выборке | Потери на тест. выборке |
|---|---|---|---|
| Ингибиторы gp120 | 0.1139 | 0.1198 | 0.1191 |

Таблица 3.2

потерь на тестовой и валидационной выборке по отношению к тренировочной, и как следствие хорошие обобщающие способности модели.

### 3.3.5 Разделение модели на кодер и декодер

Теперь, когда у нас имеется обученная модель автоэнкодера, из нее можно получить соответсвенно кодер и декодер для SMILES-формул. В псевдокоде и шагах алгоритма кодер и декодер будем обозначать как *ENCODER* и *DECODER*.

На рисунке 3.5 желтым цветом обозначена архитектура кодера. Его функция — кодировка соединений в виде SMILES-формулы в латентный вектор. На вход кодера поступают SMILES-формулы, а на выходе получаются их латентные векторы.

Декодер же предназначен для восстановления латентных векторов или для генерации новых соединений. На рисунке 3.5 декодер обозначен синим цветом. На вход декодера поступает латентный вектор, а на выходе получается восстановленная SMILES-формула.

На самом деле для работы с декодером нужна еще одна модель. Эта модель будет декодировать входной латентный вектор в $H$ и $C$ состояния, нужные для работы LSTM ячеек декодера. Такая модель будет состоять из входного слоя и двух полносвязных слоев размерностью 64 каждый. На рисунке 3.5 эти полносвязные слои следуют прямо за красным латентным пространством и обозначены зелёным цветом. На вход данной модели подается латентный вектор SMILES-формулы, а на выходе получаются $H$ и $C$ состояния, которые в свою очередь используются при работе с декодером. Для удобства обозначим эту модель как *STATESMODEL*.

До рассмотрения полного алгоритма идентификации восстановим первые 100 латентных векторов обратно в SMILES-формулы. Процент SMILES-формул, которые могут быть интерпретированы, то есть восстановлены правильно с точки зрения синтаксиса SMILES составляет 90. Стоит отметить, что это достаточно высокий процент корректно интерпретируемых латентных векторов.

### 3.3.6 Использование модели для генерации фингерпринтов соединений

Фингерпринт химического соединения — это битовая строка, содержащая информацию о структуре соединения. Чаще всего они применяются в работе с химическими базами данных, например, для более быстрого и эффективного поиска заданных соединений. Стоит заметить, что фингерпринты не всегда точно отражают заданное соединения и могут случаться false hits — промахи. Другая область применения фингерпринтов — комбинаторная химия.

Описаный и реализованный выше кодер может использоваться непосредственно для кодирования SMILES-формул в латентные векторы. Эти латентные векторы могут быть интерпретированы как фингерпринты. На рисунке 3.10 показаны соединения со схожими латентными векторами. Нетрудно заметить и их структурную схожесть

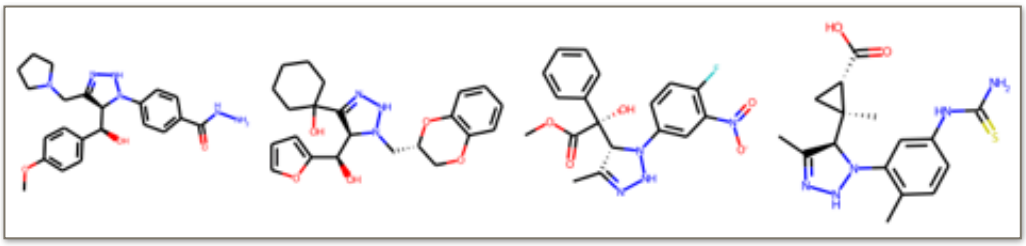


Рисунок 3.10 - Пример соединений со схожими латентными векторами

Алгоритм генерации латентных векторов выглядит следующим образом:

1. На вход алгоритма подается SMILES-формула $s$;
2. С помощью кодера SMILES-формула $s$ переводится в латентный вектор $l$.

Именно латентный вектор $l$ и будет являться фингерпринтом.

### 3.3.7 Использование модели для идентификации потенциальных ингибиторов

Наконец, полученные кодер и декодер могут использоваться для генерации новых соединений. Полностью опишем первый алгоритм генерации новых соединений:

1. На вход алгоритма подается SMILES-формула $s$;
2. С помощью кодера SMILES-формула $s$ переводится в латентный вектор $l$;
3. В латентный вектор $l$ вносится гауссовский шум с некоторым весом $w$, то есть рассчитывается вектор $l'$, $l' = l + w\xi$, где $\xi \sim \mathcal{N}(0,1)$;
4. Вектор $l'$ декодируется с помощью декодера и еще одной модели, описанной в пункте 3.3.4:
   4.1. Вектор $l'$ декодируется в $H$ и $C$ состояния;
   4.2. Полученные на предыдущем шаге $H$ и $C$ состояния задаются декодеру;
   4.3. С помощью декодера формируется SMILES-формула $s'$;

5. На выходе алгоритма получается новая SMILES-формула $s'$.

Для избежания получения одинаковых $s$ и $s'$ формул, можно перегенерировать формулу $s'$ с большим весом $w$. В конечном итоге псевдокод алгоритма 1 выглядит следующим образом:

Вход: $s$ - SMILES-формула.

Выход: $s'$ - новая SMILES-формула.

$l = ENCODER(s)$

$l' = l + w \cdot random(0,1)$

$states = STATESMODEL(l')$ // восстановление $H$ и $C$

$DECODER.setStates(states)$

$s' = DECODER(l')$

$return\ s'$

На самом деле последняя строка алгоритма выглядит по-другому. Вернемся к рисунку 3.5. В то время как вход кодера и декодера представляют из себя one-hot закодированные SMILES-формулы, выходом автоэнкодера является лишь один символ, также one-hot закодированный. Таким образом для восстановления строки $s'$ требуется словари $charToIntDict$, $intToCharDict$, а также $nUnique$, $nEntry$ из подраздела 3.2.1 и следующий порядок действий:

Вход: $l'$ - латентный вектор с шумом

Выход $s'$ - новая SMILES-формула

$s' =$ ”“

$vec = fillZeros(n)$ // заполним $vec$ $n$ нулями

$vec[charToInt['!']]$ // начинаем с условного символа начала SMILES

$for\ \ i\ in\ range(nEntry)$:

$oneHot = DECODER.predict(vec)$ // получаем one-hot вектор

$c = intToCharDict[argmax(onehot)]$ // переводим вектор в символ

$s'.append(c)$ // добавляем символ в результирующую строку

$if\ c == 'E'$:

*break* // если символ равен символу конца SMILES, завершаем работу

*return s'*

Вызов *STATESMODEL*() происходит один раз для каждой исходной SMILES-формулы!

На рисунке 3.11 слева приведено исходное соединение, а справа сгенерированные из его латентного вектора новые уникальные соединения.

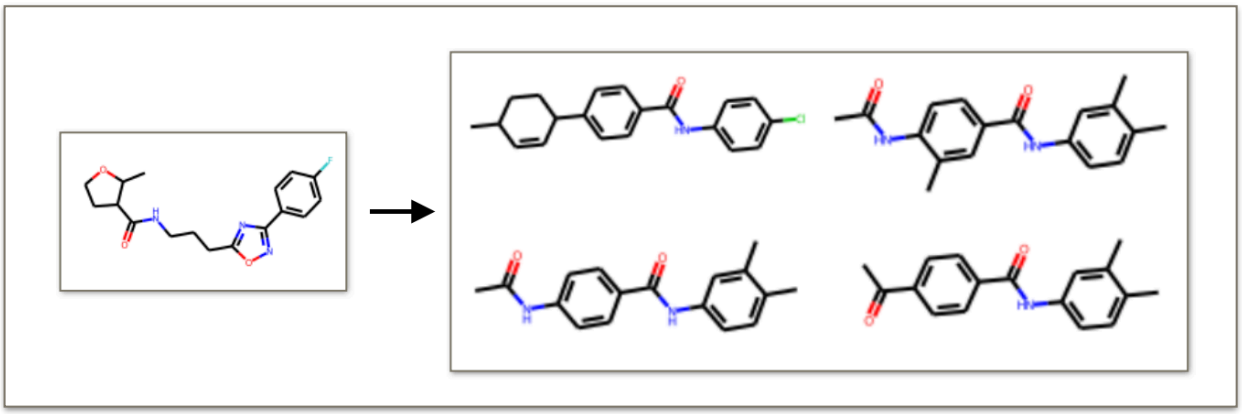


Рисунок 3.11 - Пример генерации новых соединений

Стоит заметить, что при генерации новых соединений были и ошибки, например, получались одинаковые с исходным соединения, слишком отличные от исходного соединения или соединения, которые нельзя корректно интерпретировать, однако это не помешало получить более-менее разумные результаты, которые могут быть использованы в дальнейшем.

## 3.4 Алгоритм идентификации с энергией связывания

Как было отмечено ранее, чем меньше энергия связывание, тем крепче связь ингибитор (лиганд) - белок. Поэтому имеет смысл разработать алгоритм идентификации потенциальных ингибиторов, учитывающий энергию связывания.

В данном разделе подробно рассмотрим все шаги разработки алгоритма идентификации с энергией связывания или кратко алгоритма 2.

### 3.4.1 Идея алгоритма

Идея данного алгоритма на самом деле похожа идею на предыдущего с той лишь разницей, что теперь требуется учитывать энергию связывания ингибитора и белка.

1) Вещество-потенциальный ингибитор SMILES кодируется в латентный вектор;

2) В латентный вектор вносится некоторый шум;

3) Выбирается значение энергии связывания;

4) Далее латентный вектор с шумом и энергия связывания подаются на декодер. В результате работы декодера имеем новое соединение, полученное из исходного.

Как и в предыдущем алгоритме, алгоритм 2 будет основан на LSTM автоэнкодерах.

### 3.4.2 Выбор модели

В качестве модели был выбран все так же LSTM автоэнкодер. Его архитектура представлена на рисунке 3.12. Сравним архитектуру модели алгоритма 1 и алгоритма 2.

Глобально модель состоит из двух элементов: кодера и декодера. Все очень схоже с моделью 1, но имеется входной слой для энергии (обозначен

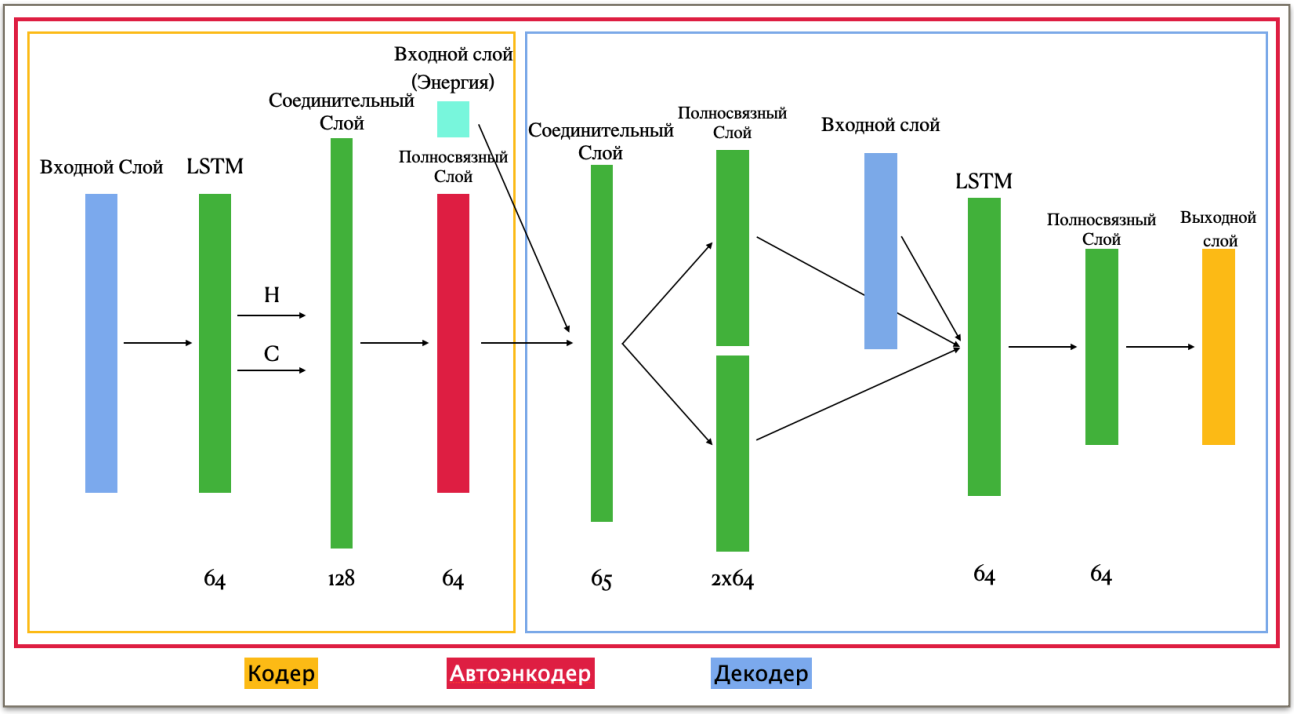



Рисунок 3.12 - Модель автоэнкодера алгоритма 2

голубым). Как известно из пункта 3.3.4, для работы с декодером требуется еще одна модель, ранее обозначенная как *STATESMODEL*. Эта модель декодирует латентный слой в *H* и *C* состояния, необходимые для работы декодера. Так как в

данной модели, по сравнению с предыдущей, учитывается еще и энергия связывания, в *STATESMODEL* включается еще и соединительный слой, конкатенирующий латентный вектор и энергию связывания.

### 3.4.3 Настройка гиперпараметров и процесс обучения модели

Модель алгоритма 2 была реализована с помощью Keras. В качестве оптимизатора был выбран Adam. Функция потерь — перекрестная энтропия. Количество эпох меньше, чем в модели 1 — 40 против 80. Подробнее выбранные гиперпараметры и т.д. рассмотрены в пункте 3.3.2. С помощью GPU

Время обучения модели

| Входной датасет | Кол-во строк | Время обучения, мин. | Число параметров |
|---|---|---|---|
| Ингибиторы gp120 | 45688 | 7 | 67871 |

Таблица 3.3

в Google Colab время обучения было значительно уменьшено :

Так как данной модели добавился входной слой, отвечающий за энергию связывания, вызов метода компиляции модели имеет вид:

```
model = Model([encoder_inputs, energy_inputs, decoder_inputs], decoder_outputs)
```

где encoder_inputs, decoder_inputs — входные слои с one-hot закодированными SMILES-формулами, energy_inputs — входной слой со значением энергии связывания, а decoder_outputs — выходной слой с символом закодированным в one-hot вектор.

Выбор функции потерь, функций активаций на остальных слоях и другие параметры остались такие же, как и в алгоритме 1.

### 3.4.4 Оценка модели

Приведем данные об процессе и результате обучения. На рисунке 3.13 по оси y расположены эпохи, а по оси x — величины ошибок. Синим цветом обозачена функция потерь на тренировочной выборке, а жёлтым — на валидационной. В таблице 3.3 более подобно представлены ошибки. Можем заметить, что они не сильно отличаются от ошибок в таблице 3.2.

На промежутке от 30 до 32 эпох на рисунке 3.13 можно увидеть результат работы ReduceLROnPlateau, описанного выше.

### 3.4.5 Использование модели для идентификации потенциальных ингибиторов

По сравнению с предыдущим разделом не будем вдаваться в подробности разбиения на кодер и декодер полученного автоэнкодера алгоритма 2. Отметим

Ошибки модели

| Входной датасет | Потери на трен. выборке | Потери на вал. выборке | Потери на тест. выборке |
|---|---|---|---|
| Ингибиторы gp120 | 0.1239 | 0.1252 | 0.1246 |

Таблица 3.3

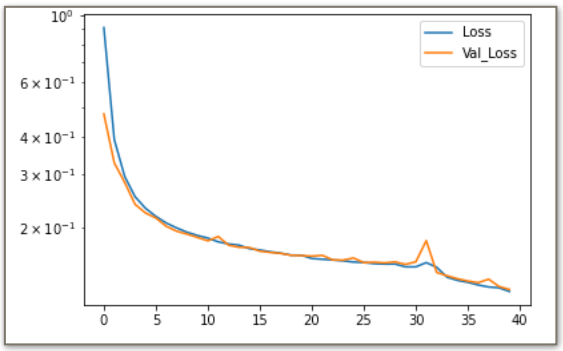



Рисунок 3.13 - Точность модели

лишь то, что *STATESMODEL* имеет еще один вход наряду с латентным вектором — значение энергии связывания. Таким образом в нашем распоряжении имеются модели *ENCODER*, *DECODER* и *STATESMODEL*.

Задачу генерации фингерпринтов из исходных соединений рассматривать так же не будем. Подробно она разобрана в пункте 3.4.6. Отметим лишь факт, что модель алгоритма 2 также подходит для генерации фингерпринтов.

Гарантировать, что соединение, представленное в виде формулы $s'$ нельзя, можно лишь предполагать. В следующей главе узнаем насколько вероятно, что задаваемая энергия связывания близка к реальной энергии. Ниже на рисунке 3.14 показан пример работы алгоритма для исходной формулы с реальной энергией -7.8 и заданными энергиями -11,-10 и -9.

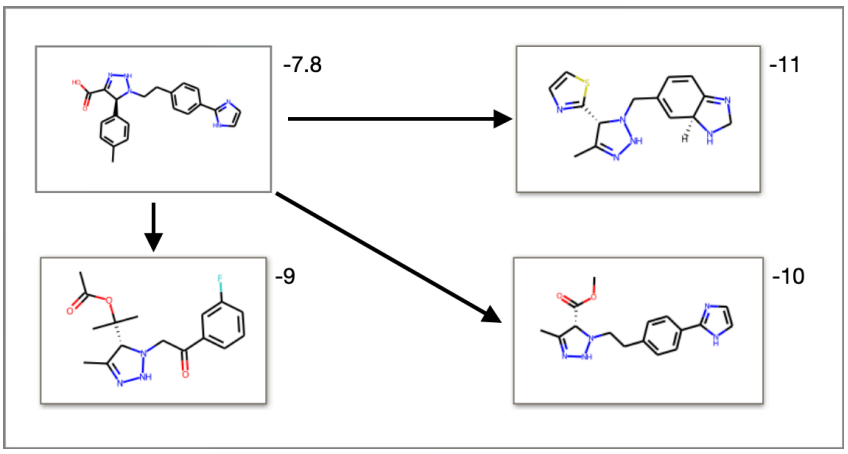



Рисунок 3.14 - Пример генерации новых соединений

Сам алгоритм генерации новых соединений с энергией связывания похож на алгоритм без энергии связывания и имеет следующий вид:

1. На вход алгоритма подается SMILES-формула $s$ и значение энергии связывания $energy$;
2. С помощью кодера SMILES-формула $s$ переводится в латентный вектор $l$;
3. В латентный вектор $l$ вносится гауссовский шум с некоторым весом $w$, то есть рассчитывается вектор $l'$, $l' = l + w\xi$, где $\xi \sim \mathcal{N}(0,1)$;
4. Вектор $l'$ конкатенируется со значением энергии $energy$ и получается вектор $l''$
5. Вектор $l''$ декодируется с помощью декодера и еще одной модели, описанной в пункте 3.3.4:
   5.1. Вектор $l''$ декодируется в $H$ и $C$ состояния;
   5.2. Полученные на предыдущем шаге $H$ и $C$ состояния задаются декодеру;
   5.3. С помощью декодера формируется SMILES-формула $s'$;
6. На выходе алгоритма получается новая SMILES-формула $s'$.

## 3.5 Генерация соединений из шума

На самом деле оба алгоритма действуют по одному принципу: закодировать исходный потенциальный ингибитор в латентный вектор, внести случайные изменения в него и декодировать, получая новое соединение.

Рассмотрим еще один вариант работы двух вышеизложенных алгоритмов — генерацию соединений из шума. Для такого способа генерации нам не понадобится даже кодировщики обеих моделей, единственное, что нам понадобится — случайный шум (и конечно же энергия, если речь идет о втором алгоритме). Далее в текущем разделе будем рассматривать применение такого варианта работы именно для первого алгоритма. Для второго все действия аналогичны, а в тех местах, где различны, будут предоставлены пояснения.

Опишем работу алгоритма 1 с генерацией из шума:

Вход: -

Выход: $s'$ - новая SMILES-формула.

$l' \sim X(\theta)$

$states = STATESMODEL(l')$ // восстановление $H$ и $C$

$DECODER.setStates(states)$

$s' = DECODER(l')$

$return\ s'$

Здесь $X$ — некоторое неизвестное распределение, речь о котором пойдет далее. Для алгоритма 2 действия аналогичны, но на вход подается еще энергия связывания, которая конкатенируется с вектором $l'$ и получается вектор $l''$.

Теперь стоит вопрос в том, откуда взять $l'$, или что из себя представляет $X(\theta)$? Рассмотрим латентный слой. Как известно из пунктов 3.3.2 и 3.4.2, латентный слой представляет из себя полносвязный слой размерностью 64. Функция активации данного слоя — ReLU. Взглянем на один из примеров латентного вектора для формулы ‘C#C[C@@]1(c2ccccc2)CO[C@@](C)(C(C)(C)C)[C@@H]2C(c3ccc(C)nc3)=NNN21’. Видно, что латентный вектор имеет некоторую логику, приобретенную во время обучения. На некоторых позициях нули, на других — числа в пределах от 0 до 100. Почему не в пределах от 0 до 1, как в ReLU? Так происходит потому что мы рассматриваем не результат работы скрытого слоя, а вектор, который на него поступает. Для тестовой выборки

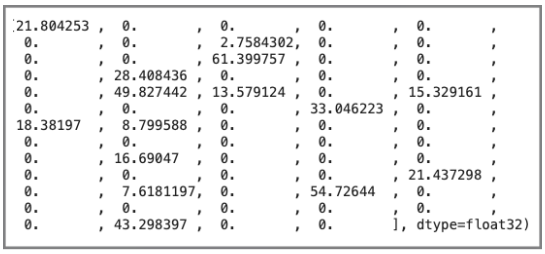



Рисунок 3.15 - Пример латеного вектора

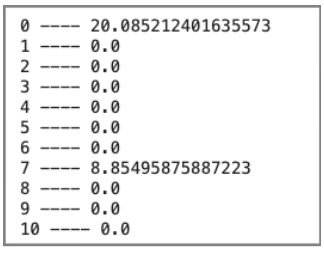



Рисунок 3.16 - Средние значения компонент

посчитаем среднее значение каждой (из 64) компоненты латентного вектора. На рисунке 3.16 представлены средние значения первых 10 компонент. Опять же видно, что для некоторых компонент среднее значение равно 0. Так как на вход латентного слоя поступают значения от 0 до 100 (на самом деле возможно и больше 100), то следовательно те компоненты, для которых среднее значение равно 0 равны 0.

Рассмотрим компоненты латентного вектора, для которых средние значения не равны 0. Всего таких компонент 16. Возможно, при повторном обучении модели алгоритма, количество ненулевые компонент изменится в силу

недетерминированности алгоритма обучения. На рисунке 3.17 представлена эмпирическая плотность распределения первой ненулевой компоненты латентного вектора для тестовой выборки. Визуально распределение похоже на гамма-распределение (3.6) $\Gamma(k, \theta)$:

$$f_X(x) = \begin{cases} x^{k-1}\frac{e^{-x/\theta}}{\theta^k \Gamma(k)}, x \geq 0 \\ 0, \qquad\qquad x < 0 \end{cases} \qquad (3.6)$$

где $X$ — распределение, $k$ и $\theta$ — параметры, $f_X(x)$ — плотность распределения.

Для того, чтобы доказать (или опровергнуть) это требуется использование аппарата математический статистики. Для 16 компонент доказательство это очень долгий процесс, который в конечном итоге может и не завершиться успехом, поэтому будем считать, что это гамма-распределение с некоторыми параметрами $k$ и $\theta$. На рисунке 3.18 можно увидеть гамма-распределение с параметрами 4, 5.5 соответсвенно. Визуально распределения на рисунках 3.17 и 3.18 схожи. Естественно критерии согласия Колмогорова и Пирсона такое приближение не пройдет. Аналогично для всех 16 компонент найдем параметры $k$ и $\theta$ гамма распределения.

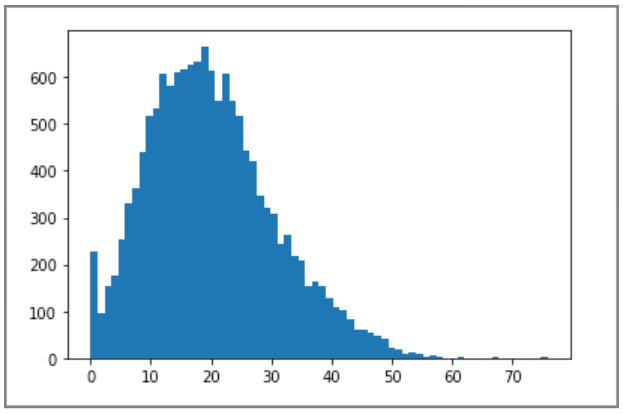



Рисунок 3.17 - Эмпирическое распределение первой компоненты

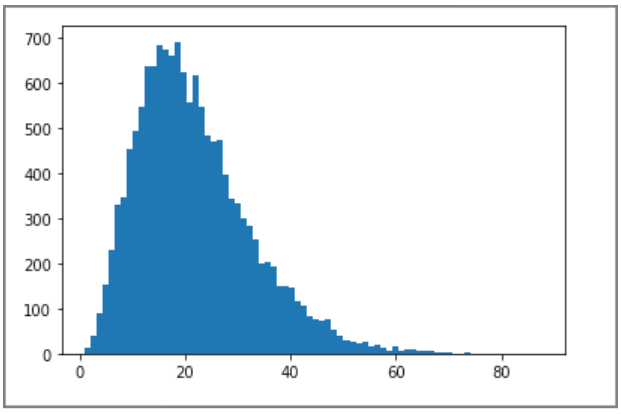



Рисунок 3.18 - Гистограмма Γ(4,5.5)

Теперь имея представление о распределении $X$ латентного слоя можно сразу генерировать латентный вектор. Для тех компонент, средняя которых равна 0, выставляем значение 0, а для компонент с ненулевым средним выбираем значение из соответствующего ему гамма-распределения. После чего вектор готов к декодированию. Ниже в таблице представлены значения для 1500

запусков алгоритма генерации. ‘Верных’ и ‘Ложных’ имеется в виду интерпретаций полученной SMILES формулы. Здесь процент успешных генераций с повторениями — отношение кол-ва верных к кол-ву запусков — (кол-во повторений+кол-во верных)/кол-во запусков. Отметим сразу преимущества данного варианта работы алгоритма над описанным в пунктах 3.3.1 и 3.4.1:

1) Скорость: 56 минут против более чем часа для того же количества соединений. Кодировщик и его работа не требуются;
2) Неограниченное число генераций. Такой вариант работы не ограничен числом исходных соединений.

| Кол-во запусков | Кол-во Верных | Кол-во Ложных | Кол-во повторений | Процент успешных генераций | Процент успешных генераций (с повторениями) | Время работы, мин |
|---|---|---|---|---|---|---|
| 1800 | 799 | 703 | 298 | 44,4 | 61 | 56 |

Таблица 3.3

Стоит также отметить и минусы. На самом деле он один — по-прежнему неизвестное распределение $X$ латентного слоя. Наше приближение всего лишь предположение, поэтому его требуется математически доказать или опровергнуть и в случае опровержения искать другое распределение.

Для контроля распределения на латентном слое можно использовать дискриминатор — специальную нейронную сеть. Дискриминаторы также используются в генеративно-состязательных сетях, упомянутых в пункте 2.2.2. Однако для биохимических задач не доказана польза дискриминаторов, в отличии, например, от задач, в которых есть четкое разделение на классы.

## 3.6 Сравнение работы алгоритмов

Несмотря на то, что разработанные алгоритмы схожи, различия все же есть. Модели, которые лежат в основе каждого алгоритма, различаются как по архитектуре, так и по параметрам обучения. На рисунке 3.15 представлены распределения параметров латентных слоев каждой модели. Слева расположены данные о модели алгоритма 1, а справа — для модели алгоритма 2. Для каждого графика ось x — значение весов, y — эпохи. Для гистограммы z — число весов.

Bias и Kernel здесь означают смещение и веса соответсвенно, которые используются в полносвязных слоях (3.7):

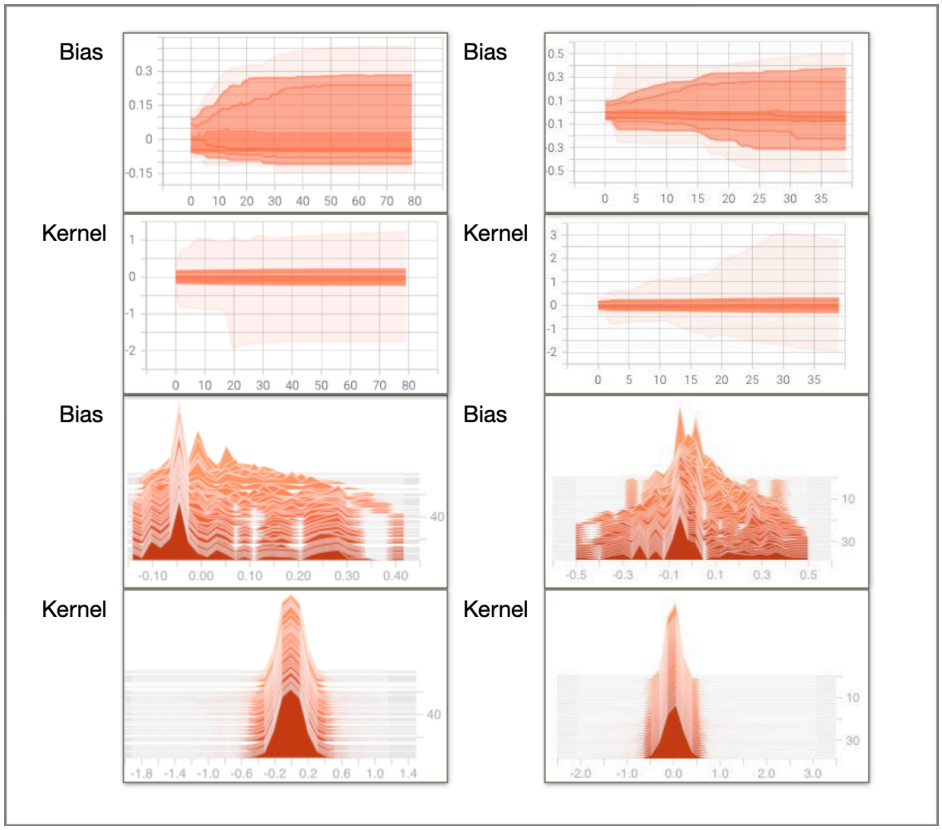



Рисунок 3.15 - Распределение на латентном слое

$$y = \phi((x, kernel) + bias) \qquad (3.7)$$

где $x$ это вход слоя, $y$ — выход, а $\phi$ — функция активации.

Так как латентный слой это в первую очередь полносвязный слой (по своей архитектуре), то поэтому на рисунке 3.15 представлены распределения Bias и Kernel.

Можно заметить, что распределения на латентных слоях похожи. Также можно отметить, что в течение эпох гистограммы изменяются, что говорит о том, что модель действительно обучается.

## 3.7 Выводы

В данной главе был рассмотрен процесс разработки двух алгоритмов для генерации потенциальных ингибиторов белка gp120. В первую очередь были проанализированы исходные данные. Для дальнейшей работы алгоритмов исходные данные были предобработаны: отфильтрованы и переведены в one-hot коды.

Далее были предложены два алгоритма для генерации потенциальных ингибиторов, использующие архитектуру LSTM автоэнкодеров. Вторая модель

представляет собой улучшение первой: вторая модель позволяет заранее задавать энергию связывания с белком gp120.

Так как оба алгоритма требуют исходное соединение для последующей генерации новых соединений, был предложен вариант работы алгоритмов с генерацией соединений из шума.

# Глава 4. АНАЛИЗ ПОЛУЧЕННЫХ ДАННЫХ

С помощью описанных выше алгоритмов 1 и 2 можно генерировать большое количество данных. Это могут быть просто соединения, похожие на исходное, или соединения с пороговой энергией. Однако все эти результаты генерации бесполезны без дальнейшего их анализа. Некоторые из соединений могут оказаться токсичными, другие иметь слабую энергию связывания, а третьи вообще попросту могут быть неконвертируемыми. Естественно, в рамках данной работы не удастся исследовать сгенерированные молекулы со всех сторон, поэтому остановимся на основой из них — молекулярном докинге.

## 4.1 Генерация новых соединений

Всего с помощью описанных в главе 3 алгоритмов были генерированы около 3 тысяч новых соединений-потенциальных ингибиторов. Все соединения генерировались из тестовой выборки, полученной в разделе 3.2. Первым алгоритмом было генерировано 1560 соединений, а вторым — 1692. Для второго алгоритма была задана энергия -9 и -11. Количество соединений с пороговой энергией -9 составило 291, а -11 — 1401. Нагляднее данные выглядят в таблице 5.1. При генерации новых ингибиторов неконвертируемые соединения сразу отфильтровывались. Проверка на конвертируемость проводилась с помощью пакета Chem библиотеки RDKit. Для каждого исходного соединения были генерированы три различных соединения-ингибитора, причем полученные соединения также отличались от исходного соединения.

Количество соединений:

| Всего | Уникальных | С помощью алгоритма 1 | С помощью алгоритма 2 | С пороговой энергией -9 | С пороговой энергией -11 |
|---|---|---|---|---|---|
| 3252 | 3224 | 1560 | 1692 | 291 | 1401 |

Таблица 4.1

В общей сложности из новых 3252 соединений есть 28 соединений, сгенерированные первым и вторым алгоритмом одновременно. Интересно то, что исходные ингибиторы этих соединений не совпадают. Таким образом в результате генерации остается 3224 уникальных соединения-ингибитора для дальнейшего анализа.

## 4.2 Молекулярный докинг

Слово докинг происходит от английского слова to dock, то есть пристыковываться. Фактически докинг — это стыковка маленькой молекулы к большой. В более научных терминах, молекулярный докинг — это метод молекулярного моделирования, позволяющий предсказать наиболее выгодную для образования устойчивого комплекса ориентацию и конформацию одной молекулы в сайте связывания другой. Первые молекулы называют лигандами, в то время как вторые — рецепторами. Чаще всего рецептор — это некоторая большая биомолекула или комплекс и задача молекулярного докинга сводится к тому, чтобы малую молекулу (лиганд) ‘пристыковать’ к более большой (рецептору). Поиск конформации имеет принцип ключ-замок. В роли замка выступает рецептор, а ключа — лиганд.

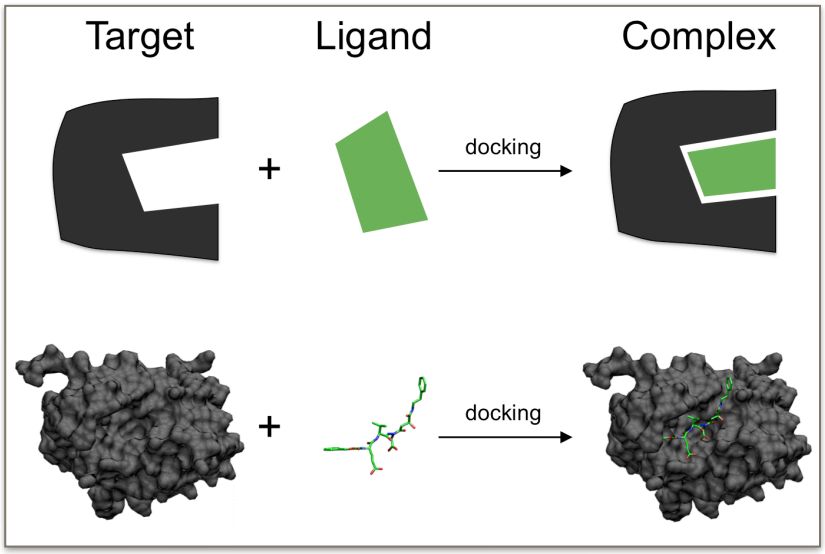



Рисунок 4.1 - Принцип ключ-замок. Пример докинга

Проводя молекулярный докинг и другие компьютерные расчеты стоит помнить, что один точный расчет без доказательства с точки зрения эксперимента не значит ничего. Если мы имеем молекулу или комплекс — потенциальное лекарство, то оно должно быть проверено: синтезом, in silico, in vitro, in vivo.

### 4.2.1 Стадии молекулярного докинга

Процесс молекулярного докинга состоит нескольких важных шагов. Рассмотрим их ниже.

Первое — выбор лиганда. Что важнее лиганд или рецептор? На этот вопрос нельзя дать точный ответ, так как это зависит от текущей задачи. В нашем случае мы отталкиваемся от рецептора — белка gp120. Однако задача может начинаться от лиганда. Например, есть ряд химически активных соединений и требуется объяснить их биологическую активность или есть рецептор.

Второе — подготовка лиганда. Подготовка лиганда включает в себя:

1) Оптимизацию геометрических параметров;
2) Оценка ионизации при физиологических pH.

В случае, если на первой стадии соединение было отобрано из некоторой базы данных, требуется также процедура фильтрации или удаления ненужных молекул. Эта процедура включает в себя

1) Определение токсичных молекул (с наличием ацилгалогенидных, сульфонилгалогенидных, акцепторов Михаэля и т.д);
2) Определение молекул с низкой проникабщей способностью (“правило пяти” Липински или “правило трех” Йоргенсена);
3) Определение неразборчиво связывающихся молекул. Это молекулы которые могут связываться в различных сайтах связывания.

Третий шаг — это поиск потенциальной биологической мишени. Если бы мы решали нашу задачу от лиганда, нам бы потребовалось найти мишень. Однако в нашем случае все просто, мишень — белок gp120.

Шаг четвертый, подготовка белка к расчету. Всего есть 4 структурные модификации белка: аминокислотная последовательность или первичная структура белка, которая включает в себя 20 аминокислот; в результате фолдинга белка происходит образование вторичной структуры белка; третичная структура; четвертичная структура белка, собрание третичных структур в комплекс. Но несмотря на обилие модификаций, все они задаются первичной структурой белка. Для подготовки белка к расчету требуется сначала загрузить геометрические параметры белка из некоторой базы данных, например PDB — Protein Data Bank. К сожалению, в геометрических параметрах нередко встречаются ошибки, поэтому для дальнейшей работы с белком требуется их исправить: найти “потерянные” тяжлые атомы, найти альтернативные позиции аминокислотных остатков и так далее.

Пятый шаг — поиск места связывания. Если место связывания известно, то такой молекулярный докинг называется прямым докингом. В таком случае все просто — проводим докинг в конкретном месте. Если же место связывания неизвестно, все обстоит гораздо сложнее, вдаваться в подробности не будем.

Последний шаг — это выбор процедуры молекулярного докинга. Этот шаг включает в себя выбор программы для докинга, например AutoDock, Glide, FlexX, LigandFit, MOE. После выбора программы требуется выбрать протокол докинга: жесткий, гибкий, принудительный и т.д.

### 4.2.2 Результаты докинга

В результате молекулярного докинга соединений, полученных путем генерации с помощью первого и второго алгоритмов, были получены значения связывания соединений (лигандов) с белком gp120 оболочки ВИЧ (рецептором).

Имеем следующие данные. На рисунке 5.2 представлены распределения энергий связывания для (сверху вниз) исходных данных, данных первого и второго алгоритмов. По оси x отложены значения энергии связывания, а по оси y значения количества соединений. Можно заметить, что в среднем значения энергии связывания данных первого и второго алгоритмов лучше, чем у исходных данных.

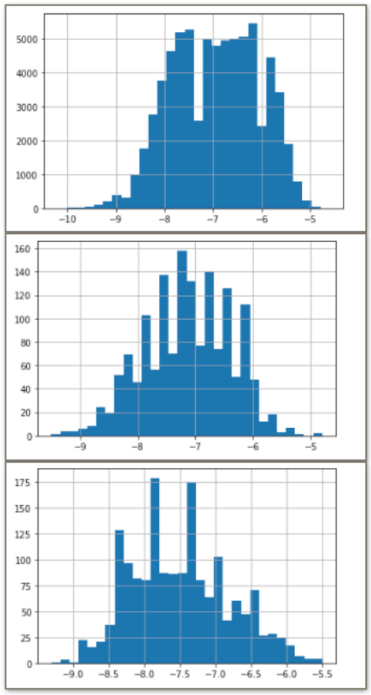


Рисунок 4.2 - Распределения энергий связывания

В таблице 4.2 приведены средние значения и значения медиан для трех наборов данных. Можно заметить, что в среднем данные алгоритмов 1 и 2 действительно имеют энергию лучше, чем у исходных данных. То же можно сказать о значении медианы. Отметит и то, что алгоритм 2 дал результат с меньше энергией связывания, чем алгоритм 1. И хотя не удалось генерировать ни одного соединения с реальной энергией, которая совпадала бы заданной алгоритму 2 энергии (в нашем случае -9 или -11),

Значение энергии связывания:

| Величина | Исходные данные | Алгоритм 1 | Алгоритм 2 |
|---|---|---|---|
| Среднее | -6.95 | -7.13 | -7.49 |
| Медиана | -6.9 | -7.1 | -7.6 |

Таблица 4.2

положительный результат от использования заданной энергии все же есть.

Отметим и тот факт, что 1090 из 1560 соединений первого алгоритма и 1211 из 1692 соединений второго алгоритма имеют реальную энергию связывания лучше (то есть меньшую) чем у соединения, из которого они были генерированы.

## 4.3 Выводы

Данная глава была посвящена анализу данных, полученных в ходе генерации алгоритмами 1 и 2. Удалось установить, что во-первых полученные данные действительно интерпретируемые, то есть SMILES-формулы новых соединений действительно представляют собой некоторый ингибитор. Во-вторых в ходе молекулярного докинга полученных данных были рассчитаны энергии связывания для белка gp120 (рецептора) и около 3 тысяч новых соединений (лигандов). Полученные соединения действительно имели энергии связывания не хуже исходных соединений, а в некоторых случаях (и в среднем) даже лучше.

# ЗАКЛЮЧЕНИЕ

Таким образом в ходе дипломной работы была поставлена задача разработки алгоритма идентификации потенциальных ингибиторов белка gp120 ВИЧ, ответственного за процесс прикрепления вируса к здоровой клетке. Разработка алгоритма идентификации проводилась с помощью методов машинного обучения и молекулярного моделирования и включала в себя в первую очередь обзор текущего положения в разработке лекарств от ВИЧ и современных методов машинного обучения.

В рамках дипломной работы были разработаны два алгоритма идентификации: не использующий данные об энергии связывания и использующий. Была продемонстрирована работа алгоритмов. Также был предложен вариант работы данных алгоритмов с генерацией ингибиторов из шума.

В результате применения алгоритмов для генерации новых потенциальных ингибиторов проникновения было идентифицировано 3000 новых соединений. Для их первичного анализа был проведен молекулярный докинг и оценена их энергия связывания с белком gp120.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Чумаков А.А., Слижов Ю.Г Система SMILES-кодирования молекулярных структур и её применение для решения научно-исследовательских задач // Национальный исследовательский Томский государственный университет, 2017 -10с.

2. Kim S, Thiessen PA, Bolton EE, Chen J, Fu G, Gindulyte A, Han L, He J, He S, Shoemaker BA, Wang J, Yu B, Zhang J, Bryant SH PubChem substance and compound databases. 2016

3. Polishchuk PG, Madzhidov TI, Varnek A Estimation of the size of drug-like chemical space based on GDB-17 data. 2013

4. Esben Jannik Bjerrum1, Boris Sattarov, Improving Chemical Autoencoder Latent Space and Molecular De-novo Generation Diversity with Heteroencoders. 2018

5. neurohive.io - Neural networks website [Electronic resource] // - Mode of access: https://neurohive.io/ru/osnovy-data-science/lstm-nejronnaja-set/. - Date of access: 20.01.2021

6. machinelearningmastery.com - Machine learning blog [Electronic resource] // - Mode of access: https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/. - Date of access: 05.04.2021

7. www.cheminformania.com - Chemical blog [Electronic resource] // - Mode of access: https://www.cheminformania.com/master-your-molecule-generator-seq2seq-rnn-models-with-smiles-in-keras/. - Date of access: 21.04.2021

8. www.paperspace.com - Blog [Electronic resource] // - Mode of access: https://blog.paperspace.com/autoencoder-image-compression-keras/. - Date of access: 03.05.2021