Федеральное государственное автономное образовательное
учреждение высшего образования
«Национальный исследовательский университет
«Высшая школа экономики»

Факультет компьютерных наук
Основная образовательная программа
Прикладная математика и информатика

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
на тему
Глубинные нейронные сети для предсказания функциональных элементов генома

Выполнил студент группы 151, 4 курса,
Бекназаров Назар Сохибжонович

Научный руководитель:
к. ф-м н., доцент, Попцова Мария Сергеевна

Содержание

Аннотация

Участки левозакрученнои Z-ДНК обнаружены в геномах различных видов. Существуют экспериментальные данные о том, что Z- ДНК играет роль в транскрипции, ремоделировании хроматина и рекомбинации. Связь эпигенетических факторов с Z-ДНК-саитами остается малоизученнои. Целью даннои работы является предсказание саитов Z-ДНК в геноме человека, связанных с эпигенетическими маркерами, с помощью моделеи машинного обучения (ML). В частности, исследуется эффективность сверточных, полносвязных и рекуррентных неиронных сетеи (CNN, FC и RNN) по сравнению с базовыми моделями машинного обучения. Было показано, что сверточные сети улучшают эффективность предсказания, а добавление рекурретных сетей к сверточным еще более значимо поднимает производительность модели. Результаты показывают перспективность применения методов глубинного обучения к биоинформатическим задачам.

Abstract

Regions of the left-handed form of Z-DNA were found in genomes of different species. There is an experimental evidence that Z-DNA plays a role in transcription, chromatin remodeling, and recombination. The association of epigenetic factors with Z-DNA sites remains poorly understood. The aim of this work is to determine the Z-DNA sites in the human genome associated with epigenetic markers with the help of machine learning (ML) models. The effectiveness of convolution, fully connected and recurrent neural networks (CNN, FC RNN) in comparison with base-line machine learning models is investigated. It was shown that convolution networks improve the efficiency of predictions but an addition of recurrent networks to convolution even more considerably increases the model performance. The results demonstrate the practical relevance of deep-learning methods for bioinformatics tasks.

1 Введение

За последние 10 лет цена на секвенирование начала резко уменьшаться. Скорость развития технологий секвенирования превысила скорость увеличения вычислительных возможностей компьютеров, что с одной стороны дало возможность изучать ранее неизученные аспекты ДНК, с другой стороны потребовало применять новые и более эффективные алгоритмы обработки биологических данных.

На основании анализа олигонуклеотидов и теоретических выкладок Джеймсом Уотсоном и Френсисом Криком в 1953 году была предложена модель построения молекулы ДНК. Согласно этой модели, ДНК имеет форму закручивающейся вокруг друг друга последовательностей нуклеотидов. Наиболее распространенная форма ДНК имеет константный угол закручивания в правую сторону относительно внешней к цепи точки наблюдения. Данная форма далее и везде будет называться как B-DNA или каноническая правозакрученная форма.

Последующие опыты открыли дополнительные неканонические формы, такие как A-DNA, Z-DNA, H-DNA, G-квадруплексы и крестообразные структуры. Данные конформации ДНК относятся к вторичным структурам. Было показано, что вторичные структуры ДНК могут влиять на взаимодействие белков с ДНК, транскрипцию ДНК, сплайсинг, уровень экспрессии и многие другие геномные процессы. Это значит, что данные модификации несут в себе генетическую информацию. Z-DNA - это участки ДНК, имеющие отрицательный угол кручения, то есть данные молекулы левозакрученные, в отличие от канонической формы.

В рамках данной работы были исследованы возможности предсказывания регионов ДНК человека, которые находятся в Z-конформации.

Актуальность данной задачи объясняется тем, что опыты подтверждают роль Z-DNA в регуляции транскрипции, а также в других геномных процессах, таких как РНК-редактирование и рекомбинация. Эксперименты по определению Z-DNA в масштабах всего генома находятся в стадии разработки, и на момент написания работы, не существовало методов, которые были бы способны предсказывать регионы Z-DNA *in vivo*. В данной работе мы используем данные результатов эксперимента CHiP-seq по картированию белка Zaa, связывающего с участками Z-DNA.

1.1 Описание задачи

Объектами исследования являются последовательности ДНК и гистоновые метки, ассоциированные с даннои последовательностью. Предметом исследования являются регионы Z-DNA и их ассоциация с другими вторичными структурами

В данной работе исследуются возможности предсказания регионов Z-DNA в зависимости от ДНК последовательности и гистоновых меток.

1.2 Цели работы

1. Парсинг экспериментальных датасетов с обнаруженными Z-DNA в геноме

2. Создание датасета в форме пригодного для методов машинного обучения

3. Построение валидационного датасета.

4. Построение бенчмарк моделей.

5. Построение моделей, основанных на методах глубинного обучения.

1.3 Методы

В даннои работе используются данные из открытых источников. Геном человека был взят с сайта USCS, версия hg19 (https://genome.ucsc.edu/). Данные по Z-DNA были взяты из эксперимен- та CHiP-seq [1]. Гистоновые метки были взяты из открытого источника CHiP-atlas (https://chip-atlas.org/).

Здесь и далее все компьютерные эксперименты были проведены на языке Python3.6. Для классического машинного обучения использовались библиотеки Sklearn, XGboost. Для глубокого обучения была использована библиотека Pytorch.

Данные по Z-DNA были преставлены в форме .csv таблицы. Регионы Z-DNA были расширены до общего размера в 5000 нуклеотидов. Каждый регион Z-DNA был помещен в центр выбранного региона. Далее из генома случайно были насемплированы отрезки размера 5000 нуклеотидов.

Далее был создан датасет следующего вида. Каждому номеру нуклеотида на определенной хромосоме сопоставляется вектор, содержащий нуклеиновую кислоту, закодированную при помощи OneHotEncoding, соответствующую данному нуклеотиду, значения гистоновых меток и бинарная метка, есть ли данный нуклеотид в регионе Z-DNA. Таким образом на датасет имел размер $1153 \times 5000 \times 100$.

2 Обзор литературы

С момента обнаружения Z-DNA в 1979 году[2], биологическая значимость Z-DNA была спорной. Более тщательные исследования выявили специфические роли Z-ДНК. Стали известны такие функции, как роль в редактировании РНК и транскрипционной регуляции [3]. Таким образом, предсказание отрицательно закрученных областей ДНК является важнейшей задачей. Из работ по анализу кристаллической структуры олигонуклеотидов можно сделать вывод о некоторой априорной информации о распределении Z-областей ДНК, отображенных в последовательности ДНК[4].

1. Более вероятно, что Z-DNA встречается в чередующихся пурин-пиримидиновых последовательностях. Это связано с тем, что конфигурация Z-DNA является более высокоэнергетическим состоянием молекулы. Среди всех последовательностей ДНК пурин-пиримидиновые последовательности требуют минимального количества дополнительной энергии для получения Z-конфигурации.

2. Все пурин-пиримидиновые последовательности могут быть эмпирически отсортированы по вероятности выкрутиться в Z-конфигурацию. Чаще всего $d(GC)_n$ становится Z-DNA, $d(CA)_n$ становится реже. $d(AT)_n$ был обнаружен меньше всего

3. Эмпирически доказано, что более длинные отрезки последовательностей образуют Z-конфигурацию более вероятно, чем короткие.

Однако этих знаний недостаточно. *In vivo* пурино-пиримидиновые последовательности типа $d(GC)_n$ и $d(CA)_n$ легко принимают Z-конформацию,

а чередующаяся последовательность типа $d(CA)_n$, вероятно, нет. Кроме того, проблема заключается в том, что невозможно эмпирически определить вероятность принятия Z-конфигурации для всех возможных пурино-пиримидиновых последовательностей, которые нельзя представить в виде $d(GC)_n$ или $d(CA)_n$, например $GCATGCAT$. Подход, описанный в Z-HUNT[5], предлагает учитывать только взаимодействия между соседними нуклеотидами, что, тем не менее, не может полностью разрешить всю сложность проблемы[5]. Сложность связана, кроме того, с тем, что некоторые исследования показывают принципиальную невозможность обнаружения областей Z-DNA только на основе последовательностей ДНК[5]. Также известно,что разные эпигенетические факторы могут влиять на вероятность того, что ДНК примет Z-конфигурацию. Некоторые исследования показывают, что области Z-DNA тесно связаны с ядерным фактором (NFI) и фактором транскрипции CAAT-box[6, 7, 8]. Следовательно, использование знаний о распределении меток NFI может помочь в прогнозировании областей Z-ДНК. Учитывая сложный нелинейный характер данных, машинное обучение является хорошим подходом для решения этой проблемы.

Есть работы, которые пытаются предсказать сайты Z-DNA на основе алгоритмов, не связанных с машинным обучением. Например, вышеупомянутый Z-HUNT. Данный алгоритм использует вышеописанные особенности Z-DNA. Алгоритм оценивает вероятность вступления в Z-конфигурацию в зависимости от количества энергии необходимой для каждых двух соседних нуклеотидов выкрутиться в Z-конфигурацию, выдавая в конце лог-вероятность участку принять Z-DNA конформацию. Данный подход имеет хорошие стороны, как например он оценивает именно вероятность, и для каждой последовательности она не

нулевая, что значит, что любая последовательность может вступить в Z-конфигурацию, только одни последовательности с большей вероятностью, другие с меньшей. Недостатком данного подхода является факт того, что все априорные аналитические данные актуальны только для *in vitro* случая. Тогда как *in vivo* данные закономерности не воспроизводятся. Данный алгоритм не учитывает сложных нелинейных взаимодействий между более чем двумя нуклеотидами, то есть аппроксимация данного алгоритма достаточно грубая. У данного алгоритма есть модификация modern Z-HUNT [ссылка], в которой дополнительно используется информация о NFI. Тем не менее, данные алгоритмы не применяют подходов машинного обучения и основываются только на статистических методах и сильно ограничены в сложности.

В данном случае глубинное обучение кажется применимым к данной задаче. Есть несколько работ, которые применили глубинное обучение в области предсказания функциональных элементов генома[9]. Так, DanQ, показывает самые высокие результаты в задаче прогнозирования транскрипционных факторов [9]. CNN, RNN и их смесь доказали, что глубокое обучение может стать широко используемым подходом в биоинформатике.

3 Основная часть

3.1 Парсинг регионов Z-DNA

Из статьи по экспериментальному поиску регионов Z-DNA из приложения были взяты данные о расположение регионов Z-DNA. Данная таблица была пред- ставлена в формате pdf, из-за чего перед использованием

ее требовалось привести в табличныи формат. Это было сделано применяя доступные программные средства.

Далее данная таблица была очищена от регионов с хромосомами, которые имели окончание random. Данная таблица была преобразована в формат csv.

Далее при помощи языка Python и програмных пакетов pandas и numpy данные регионы были увеличены до размера 5000, с позиционирование Z-DNA в центре каждого региона.

При фиксированном random seed, были насемплированы отрезки из других частеи человеческого генома. Семплирование было произведено таким образом, что не должно было быть пересечении между насемплированными отрезками и отрезками, содержащими Z-DNA.

Отобранные отрезки были объединены вместе.

Для возможности использовать bedtools, данный набор был переведен в формат bed.

3.2 Создание датасета в требуемом формате

На предыдущем этапе было получено 1153 отрезка ДНК, каждому из которых необходимо сопоставить соответствующую нуклеиновую кислоту и вектор из гистоновых меток.

Для того, чтобы сопоставить каждому отрезку последовательность ДНК, был использован геном человека hg19, которыи был записан в формате fasta. При помощи bedtools каждому отрезку была сопоставлена ДНК последовательность. Код есть в Приложении.

Далее необходимо было создать вектор из меток гистоновых меток. Для этого были скачаны все гистоновые метки по всем тканям человека

в саита CHiP atlas. Было скачено 95 датасетов с разными гистоновыми метками. Для каждои из 95 видов ги- стоновых меток были созданы собственные фаилы в формате bed. Далее для каждого отрезка из ранее подготовленного множества с каждым типом гистоновых меток был произведен поиск пересечений. Таким образом, необходимо было сделать 1153 95 вызова программ bedtools. Данные операции при эффективнои реализации занимали более 180 часов, поэтому вызовы данных операции были запущены параллельно, что позволило ускорить вычисление на процессоре с 64 ядрами до 4 ч.

В датасете гистоновых меток из-за того, что они принадлежали разным тканям и из-за особенностей метода CHiP-seq, которым они были получены, есть следующие особенности. Каждому отрезку сопоставляется число от 0 до 1000, которое характеризует вероятность того, что на даннои нуклеосоме деиствительно была наидена данная гистоновая метка. 1000 означает максимальную вероятность, тогда как 0 характеризует нулевую вероятность. Данная шкала была сведена к отрезку от 0 до 1 и далее использовалась как числовая величина.

Далее столбец нуклеиновых кислот был закодирован, используя метод «One Hot Encoding», или метод прямого кодирования.

Далее все датасеты были собраны в подходящии для машинного обучения формат. Было составлено 1153 таблиц в формате csv, каждая таблица имела размер 5000 100, где каждая строка - это нуклеотид и соответствующие ему нуклеиновая кислота и гистоновые метки. Столбцы представляли собой следующие данные: один столбец, хранящии символ нуклеиновои кислоты; 95 столбцов хранящих числа от 0 до 1 в вещественном формате, характеризующиие вероятности наличия гистоновой

метки, и последнии столбец, хранящии индикатор Z-DNA в булевом формате.

3.3 Построение валидационного датасета

Для измерения качества полученнои модели необходимо создать бенчмарк модели для сравнения полученные результатов. Так же необходимо было создать тестовую часть выборки. Для корректности отображения разрешающеи способности даннои модели было сделано разделение таким образом, что отношение несущих в себе Z-DNA отрезков к случаиным оставалось примерно одинаковым. Для этого был использован инструмент из библиотеки sklearn - stratified train test split. Так же для возможности воспроизвести результаты был фиксирован random seed. Далее и везде все результаты считаются именно на этом датасете.

3.4 Построение бенчмарк модели и ее оценка

Построение бенчмарк модели может показать не только тот уровень качества, с которым необходимо сравнивать модель глубинного обучения, но и показать возможно ли применить на данном датасете вообще какие-либо методы машинного обучения. Ведь Z-DNA может вовсе не зависеть ни от цепи ДНК, ни от навешанных на нее гистоновых меток.

Классическое машинное обучение предлагает много вариантов для банчмарк моделей. В данной работе рассматриваются следующие:

- Линейная модель
- Дерево решений
- Градиентный бустинг

Первые две модели были выбраны как самые простые. Градиентный бустинг был выбран в качестве state-of-the-art модели, которая, даже при параметрах по умолчанию, способна давать решения высокого качества.

Далее модели сначала тестировались, только с основываясь на ДНК цепи, после дополнительно добавлялись гистоновые метки.

Перед применением моделей требуется закодировать ДНК цепь. В работе будет рассмотрено 3 вида кодировки:

- One hot encoding
- TF-IDF
- TF-IDF по разным регионам

One hot encoding - был выбран в качестве самого распространенного в биоинформатике подхода для кодировки ДНК цепи. TF-IDF - был выбран из следующего предположения. Допустим, что на вероятность появления Z-DNA влияет некоторая определенная недлинная последовательность ДНК, которая будет редко встречаться в обычных отрезках и часто в окрестности Z-DNA. Тогда модель должна уловить данную закономерность, не имея прямого доступа к последовательности, а только к TF-IDF кодировке. По схожим причинами был предложен третий тип кодирования. Допустим, что некоторая последовательность часто встречается либо непосредственно внутри отрезка Z-DNA, либо в его правой или левой окрестности, если рассматривать положительную цепь ДНК. Тогда TF-IDF, построенный на разделенных на 3 части цепях ДНК должен сохранить эту информацию, а модели должны будут суметь уловить данную закономерность.

В качестве оценок будем использовать две метрики, это AUC-ROC, accuracy.

AUC-ROC была выбрана в качестве общепринятой метрики при оценке качества бинарной классификации.

Accuracy была выбрана в качестве метрики показатель которой нам понятен напрямую.

Тем не менее можно заметить, что все три метрики не могут быть оптимизированы напрямую, поэтому они будут использоваться только в качестве конечных метрик и в течение обучения в абсолютном большинстве моделей использоваться не будут.

Так же необходимо уметь различать случайное угадывание от действительно неслучайного, но плохого результата. Для этого для каждой метрики при помощи бутстрепа на случайных ответах были найдены квантили распределения случайных угадываний. Все результаты, которые находились внутри вышеуказанных квантилей, были обозначены дополнительно.

3.5 Построение модели глубинного обучения

Далее после получения результатов для бенчмарк моделей было опробовано несколько классов моделей и их многочисленные модификации.

Все модели были реализованы в библиотеке PyTorch 1.0. Все они оптимизировали бинарную кросс-энтропию. Все они были оптимизированы методом Adam с постепенно уменьшающимся шагом. Шаг уменьшался начиная с 0.001, поочередно делился на 2 и на 5. Обучение, из-за естественных ограничении размера памяти используемых видеокарт, проис-

ходило по батчам. Все вычисления были произведены на компьютере со следующими характеристиками:

- CPU: IntelCore i7-6950X 3.00GHz X 20
- RAM: 62.8 Gb
- GPU1: GeForce GTX 1080Ti
- GPU2: GeForce GTX 1080Ti

Все неоговоренные гиперпараметры были выбраны по умолчанию.

Все типы моделей можно разделить на 3 части:

1. CNN
2. CNN + FC
3. CNN + RNN + FC

3.6 CNN

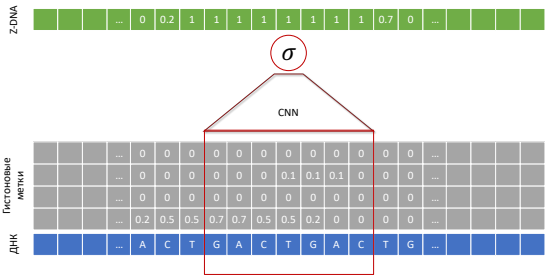
Как и для классического машинного обучения так и для биоинформатической науки применение моделей вида CNN должно иметь под собой некоторые априорные представления о датасете. При использовании CNN на задаче классификации изображений мы неявно предполагаем, что объекты на картинке инвариантны относительно сдвига, это и является нашим априорным знанием. В данной же задаче мы можем сделать следующие предположения. Регионы, более расположенные к тому, чтобы принять Z-конфигурацию имеют в некотором окружении или непосредственно в самих себе некоторые мотивы, которые могут зависеть

либо только от последовательности нуклеотидов, либо от гистоновых меток, либо от всего сразу. Это дает нам возможность предположить высокую разрешающую способность моделей, использующих в себе данное предположение.

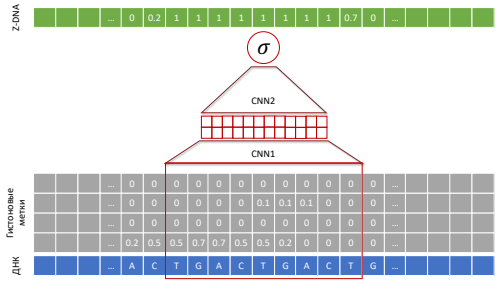
Далее каждая используемая модель будет иметь свой номер.

1. Данная модель будет иметь сложность линейной модели. Ее устроиство изображено на Рис 1 (a). Данная модель была рассмотрена с разными показателями размера ядра, и сдвига. Количество ядер было равно 1. Таким образом данная модель имеет некоторое множество выходов, каждому из этих выходов будет соответствовать некоторое значение от 0 до 1. Число, которое будет соответствовать каждому выходу, вычисляется следующим образом. Пусть размер ядра C, тогда для каждого выхода будет соответствовать усредненное значение по C значениям Z-DNA. Так как паддинг установлен отсутствующим то количество выходов модели и усреднённых значений равно. То есть каждая такая модель будет предсказывать среднее количество нуклеотидов, которое встречалось на данном отрезке и сопоставлять это число середине данного отрезка. Данная модель по сложности рана линейной, но линейную модель из предыдущей части не воспроизводит.

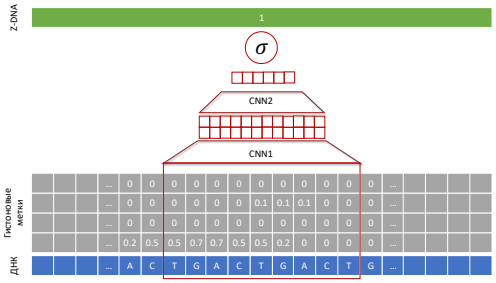
2. Данная модель отличается от предыдущей увеличенным количеством слоев. Между слоями стоит стандартная активация ReLU. В данном случае таргет вычисляется немного другим способом. Усреднение происходит по размеру последнего слоя. Размеры ядер и количество ядер на первом и втором слое выбирались из некоторого множества значений.



(a) Model 1



(b) Model 2



(c) Model 3

Рис. 1: CNN модели

3. Данная модель сильнее отличается от предыдущих тем, что выход второго слоя объединяется по всей цепочке ДНК. Таргет модели выбирается следующим образом. Если в регионе есть регион Z-DNA ему сопоставляется 1, 0 в другом случае. То есть данная модель определяет существует ли в данной цепи регион Z-DNA.

Код всех моделей доступен в Приложении.

3.7 CNN + FC

В данном типе сетей будет рассмотрена модификация всех типов сетей из предыдущего раздела. То есть к предыдущим архитектурам был просто добавлен полносвязный слой. Так же стоит упомянуть, что данные слои в 1 и 2 модели были реализованы в качестве одномерных сверток. Можно показать, что свертка с ядром размера 1, шагом размера 1 и n фильтрами

полностью инвариантна полносвязному слою с n нейронами на том же входном векторе.

3.8 CNN + RNN + FC

Данный вид гибридных моделей был успешно применен в статье DanQ[ссылка]. Устройство модели показано на рис2. Данная модель использует еще одно дополнительное априорное предположение о данных. Мы, таким образом, вносим в модель информацию о рекуррентной природе ДНК. То есть мы предполагаем, что мотивы, которые в состоянии различить CNN-слой, будут иметь еще и рекуррентные зависимости. Такая сеть в теории способна различить последовательную группу мотивов, от которых зависит появление Z-DNA конфигурации.

Так как постановка задачи не закреплена есть возможность для применения нескольких моделей построения задачи. Есть несколько характерных для RNN типов задач. Каждая из них представлена на рис3. В данной работе было рассмотрено два подхода many-to-many и many-to-one.

3.8.1 many-to-one

В данном случае модель будет устроена следующим образом. Первая часть модели - это слой CNN, один или несколько, каждый столбец полученного выхода будет отдельно передаваться в RNN сеть. В нашем случае для RNN будет выбрана многослойная двунаправленная LSTM. Далее, количество слоев в CNN и LSTM частях будет подбираться. Размеры ядер и скрытых слоев будут подбираться. В конце и начале последовательности слой RNN будет выдавать 2 вектора, который ассоциирован

с долгой памятью LSTM. Их будет 2 так как наша RNN двунаправленная. Эти вектора будут передаваться в полносвязный слой, который и будет делать предсказание. Таргетом будет индикатор существования Z-региона в данной последовательности.

3.8.2 many-to-many

Данная архитектура будет полностью копировать предыдущую кроме одного аспекта. После слоя RNN мы будем игнорировать выход долгосрочной памяти и агрегировать выходы краткосрочной памяти каждого из направлений. Далее каждой единице последовательности будут соответствовать два вектора, которые и будут передаваться в полносвязный слой и далее делаться предсказания для каждой части последовательности. Таргет в этом случае будет высчитываться в точности, как для случая с CNN. То есть каждой единице последовательности будет сопоставляться среднее некоторого региона цепочки.

Все гиперпараметры экспериментов указаны в результатах. Код моделеи доступен в приложении.

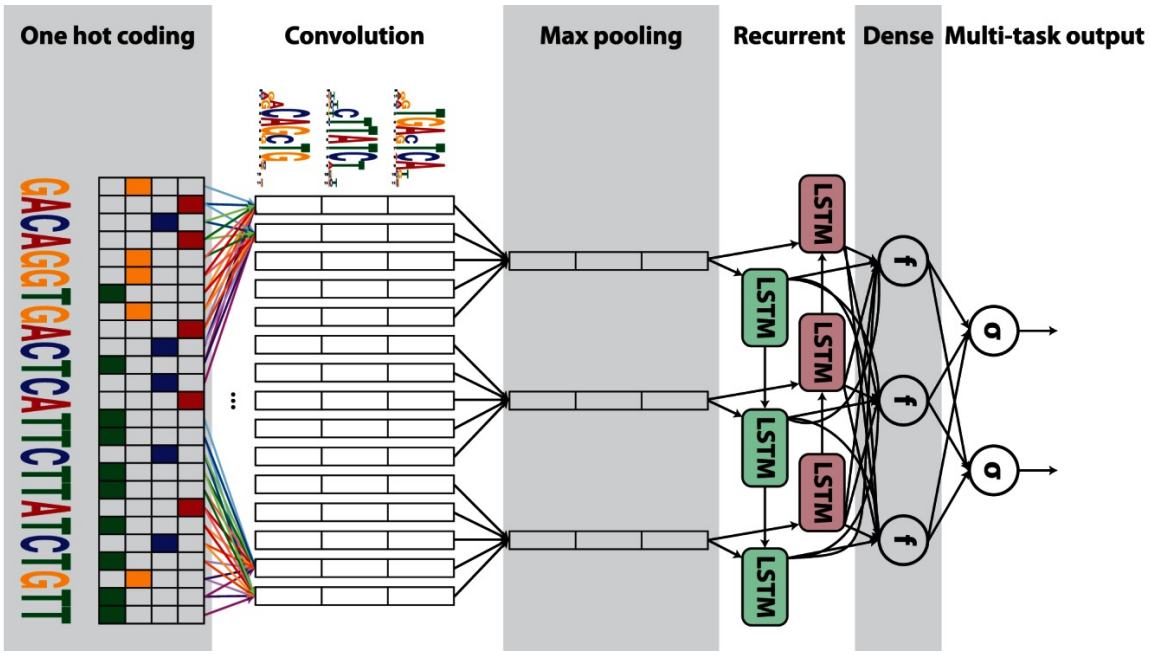


Рис. 2: Hybrid

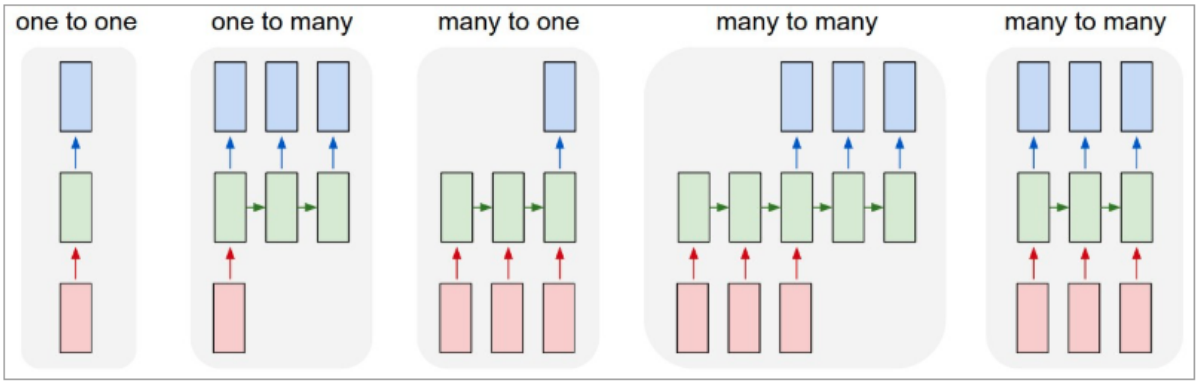


Рис. 3: Типы RNN

4 Полученные результаты

4.1 Бенчмарк модели

		Кодировка		
Модель	Ошибка	OHE	TF-IDF	TF-IDF по рег
Линейная модель	AUC	0.504	0.502	0.511
	Acc	0.667	0.672	0.669
Дерево решений	AUC	0.510	0.512	0.492
	Acc	0.656	0.666	0.656
Градиентный бустинг	AUC	0.531	0.529	0.532
	Acc	0.681	0.622	0.691

При помощи бутстрепа были вычислены квантили для распределения случайного AUC. Все показатели, которые попадают внутрь доверительного интервала обозначены красным.

Можно увидеть, что первые две модели имеют крайне низкое качество, не отличимое от случайного. Точность почти равна размеру большего класса. Что говорит о том что либо модели переобучились либо не дообучились, что в сущности не важно, так как на объектах, которых модели не видели, предсказания не отличаются от случайного угадывания.

Ситуация немного отличается, когда мы используем градиентный бустинг, что ожидается. Данная модель имеет краине высокую разрешающую способность, но, тем не менее, ее качество, если судить по AUC-ROC, отличается от случаиного угадывания, что говорит о том, что какие-то закономерности в данных все же существуют.

Результаты данных моделей, к сожалению, нельзя использовать в качестве хорошего бенчмарка. В качетсве бенчмарка будет выбрана просто случайная модель.

4.2 CNN & CNN + FC

Архитектура	Acc %	AUC %
model 1	62	65
model 2	61	67
model 3	66	53

a)CNN

Архитектура	Acc %	AUC %
model 1	57	66
model 2	55	69
model 3	68	54

b)CNN+FC

4.3 CNN + RNN + FC

Так как в ходе подбора параметров не было четкой табличной логики ниже представлен список моделей и качетва которое удалось достичь на данных архитектурах.

Архитектура	Acc %	AUC %
model 1	74	85
model 2	53	52
model 3	73	86
model 4	67	80
model 5	74	84.5
model 6	74	86
model 7	72	85
model 8	76	85.5
model 9	75	86.5
model 10	70	85
model 11	72	85
model 12	72	85

a)many-to-one

Архитектура	Acc %	AUC %
model 1	62	74
model 2	51	54
model 3	64	75
model 4	58	70
model 5	64	77
model 6	63	72
model 7	67	77
model 8	69	80.5
model 9	65	74
model 10	65	76
model 11	68	75
model 12	62	77

b)many-to-many

5 Выводы

Из приведенных выше результатов можно сделать следующие выводы:

- Модели без учета рекуррентной природы данных работают сильно хуже чем, модели, которые учитывают данную природу. Это видно из таблиц с результатами, так как бенчмарк модели и модели из CNN подгруппы работают сильно хуже, чем модели, которые содержат RNN слой.

- Максимальное качество, которое можно достичь на данном датасете при помощи данного множества архитектур не превышает 86 %

AUC, что тем не менее говорит о том, что данная задача в *in vivo* постановке может быть вообще решена.

- Методы глубокого обучения действительно могут найти широчайшее применение во многих ранее не решенных задачах машинного обучения.

6 Список изученной литературы

Список литературы

[1] So-I Shin, Seokjin Ham, Jihwan Park, Seong Hye Seo, Chae Hyun Lim, Hyeongrin Jeon, Jounghyun Huh, and Tae-Young Roh. Z-dna-forming sites identified by chip-seq are associated with actively transcribed regions in the human genome. *DNA Research*, 23(5):477–486, 2016.

[2] Andrew H-J Wang, Gary J Quigley, Francis J Kolpak, James L Crawford, Jacques H Van Boom, Gijs van der Marel, and Alexander Rich. Molecular structure of a left-handed double helical dna fragment at atomic resolution. *Nature*, 282(5740):680, 1979.

[3] P Shing Ho. Thermogenomics: thermodynamic-based approaches to genomic analyses of dna structure. *Methods*, 47(3):159–167, 2009.

[4] Pui S Ho, Michael J Ellison, Gary J Quigley, and Alexander Rich. A computer aided thermodynamic approach for predicting the formation of z-dna in naturally occurring sequences. *The EMBO journal*, 5(10):2737–2744, 1986.

[5] Gary P Schroth, PING-JUNG Chou, and P Shing Ho. Mapping z-dna in the human genome. computer-aided mapping reveals a nonrandom distribution of potential z-dna-forming sequences in human genes. *Journal of Biological Chemistry*, 267(17):11846–11855, 1992.

[6] A Herbert, K Lowenhaupt, J Spitzner, and A Rich. Double-stranded rna adenosine deaminase binds z-dna in vitro. In *Nucleic acids symposium series*, number 33, pages 16–19, 1995.

[7] Thomas Schwartz, Mark A Rould, Ky Lowenhaupt, Alan Herbert, and Alexander Rich. Crystal structure of the zα domain of the human editing enzyme adar1 bound to left-handed z-dna. *Science*, 284(5421):1841–1845, 1999.

[8] Rui Liu, Hong Liu, Xin Chen, Martha Kirby, Patrick O Brown, and Keji Zhao. Regulation of csf1 promoter by the swi/snf-like baf complex. *Cell*, 106(3):309–318, 2001.

[9] Daniel Quang and Xiaohui Xie. Danq: a hybrid convolutional and recurrent deep neural network for quantifying the function of dna sequences. *Nucleic acids research*, 44(11):e107–e107, 2016.

[10] Babak Alipanahi, Andrew Delong, Matthew T Weirauch, and Brendan J Frey. Predicting the sequence specificities of dna-and rna-binding proteins by deep learning. *Nature biotechnology*, 33(8):831, 2015.

[11] Yann LeCun, Y Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–44, 05 2015.

[12] Yifei Chen, Yi Li, Rajiv Narayan, Aravind Subramanian, and Xiaohui Xie. Gene expression inference with deep learning. *Bioinformatics*, 32(12):1832–1839, 2016.

[13] Vladimir Vapnik, Esther Levin, and Yann Le Cun. Measuring the vc-dimension of a learning machine. *Neural computation*, 6(5):851–876, 1994.

[14] Lucia A Hindorff, Praveen Sethupathy, Heather A Junkins, Erin M Ramos, Jayashri P Mehta, Francis S Collins, and Teri A Manolio. Potential etiologic and functional implications of genome-wide association loci for human diseases and traits. *Proceedings of the National Academy of Sciences*, 106(23):9362–9367, 2009.

[15] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[16] Jian Zhou and Olga G Troyanskaya. Predicting effects of noncoding variants with deep learning–based sequence model. *Nature methods*, 12(10):931, 2015.

[17] Alan Herbert and Alexander Rich. The biology of left-handed z-dna. *Journal of Biological Chemistry*, 271(20):11595–11598, 1996.

[18] P Christoph Champ, Sandor Maurice, Jeffrey M Vargason, Tracy Camp, and P Shing Ho. Distributions of z-dna and nuclear factor i in human chromosome 22: a model for coupled transcriptional regulation. *Nucleic acids research*, 32(22):6501–6510, 2004.

[19] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[20] Daniel Quang, Yifei Chen, and Xiaohui Xie. Dann: a deep learning approach for annotating the pathogenicity of genetic variants. *Bioinformatics*, 31(5):761–763, 2014.

[21] Blaise Dumat, Anders Foller Larsen, and L Marcus Wilhelmsson. Studying z-dna and b-to z-dna transitions using a cytosine analogue fret-pair. *Nucleic acids research*, 44(11):e101–e101, 2016.

[22] Dina Zhabinskaya and Craig J Benham. Theoretical analysis of competing conformational transitions in superhelical dna. *PLoS computational biology*, 8(4):e1002484, 2012.

7 Приложение

7.1 CNN модели

7.1.1 model1

```
ConvNet(
  (cnn): Sequential(
    (0): Conv1d(99, 2,
                kernel_size=(13,),
                stride=(2,),
                padding=(6,))
    (1): Sigmoid()
  )
)
```

7.1.2 model2

```
ConvNet(
  (cnn): Sequential(
    (0): Conv1d(99, 36,
                kernel_size=(13,),
                stride=(2,),
```

```
                padding=(6,))
    (1): ReLU()
    (2): Conv1d(36, 2,
                kernel_size=(13,),
                stride=(2,),
                padding=(6,))
    (3): Sigmoid()
  )
)
```

7.1.3 model3

```
ConvNet(
  (cnn): Sequential(
    (0): Conv1d(99, 36,
                kernel_size=(13,),
                stride=(5,),
                padding=(6,))
    (1): ReLU()
    (2): Conv1d(36, 8,
                kernel_size=(13,),
                stride=(5,),
                padding=(6,))
    (3): ReLU()
    (4): Linear(in_features=1600, out_features=2, bias=True)
    (5): Sigmoid()
  )
)
```

7.2 RNN модели

7.2.1 model1

```
HybridNet(
  (cnn): Sequential(
    (0): Conv1d(99, 36, kernel_size=(13,), stride=(2,), padding=(6,))
    (1): ReLU()
    (2): Conv1d(36, 48, kernel_size=(13,), stride=(2,), padding=(6,))
    (3): ReLU()
```

```
    (4): Conv1d(48, 64, kernel_size=(13,), stride=(2,), padding=(6,))
    (5): ReLU()
  )
  (rnn): LSTM(64, 64, bidirectional=True)
  (drop): Dropout(p=0.5)
  (fcl): Conv1d(128, 2, kernel_size=(1,), stride=(1,))
)
```

7.2.2 model2

```
HybridNet(
  (cnn): Sequential(
    (0): Conv1d(99, 36, kernel_size=(3,), stride=(1,), padding=(1,))
    (1): ReLU()
    (2): Conv1d(36, 48, kernel_size=(3,), stride=(1,), padding=(1,))
    (3): ReLU()
    (4): Conv1d(48, 64, kernel_size=(3,), stride=(1,), padding=(1,))
    (5): ReLU()
  )
  (rnn): LSTM(64, 64, bidirectional=True)
  (drop): Dropout(p=0.5)
  (fcl): Conv1d(128, 2, kernel_size=(1,), stride=(1,))
)
```

7.2.3 model3

```
HybridNet(
  (cnn): Sequential(
    (0): Conv1d(99, 36, kernel_size=(3,), stride=(2,), padding=(1,))
    (1): ReLU()
    (2): Conv1d(36, 48, kernel_size=(3,), stride=(2,), padding=(1,))
    (3): ReLU()
    (4): Conv1d(48, 64, kernel_size=(3,), stride=(2,), padding=(1,))
    (5): ReLU()
  )
  (rnn): LSTM(64, 64, bidirectional=True)
  (drop): Dropout(p=0.5)
  (fcl): Conv1d(128, 2, kernel_size=(1,), stride=(1,))
)
```

7.2.4 model4

```
HybridNet(
  (cnn): Sequential(
    (0): Conv1d(99, 36, kernel_size=(5,), stride=(5,), padding=(2,))
    (1): ReLU()
    (2): Conv1d(36, 48, kernel_size=(5,), stride=(5,), padding=(2,))
    (3): ReLU()
    (4): Conv1d(48, 64, kernel_size=(5,), stride=(5,), padding=(2,))
    (5): ReLU()
  )
  (rnn): LSTM(64, 64, bidirectional=True)
  (drop): Dropout(p=0.5)
  (fcl): Conv1d(128, 2, kernel_size=(1,), stride=(1,))
)
```

7.2.5 model5

```
HybridNet(
  (cnn): Sequential(
    (0): Conv1d(99, 36, kernel_size=(5,), stride=(5,), padding=(2,))
    (1): ReLU()
    (2): Conv1d(36, 48, kernel_size=(5,), stride=(5,), padding=(2,))
    (3): ReLU()
    (4): Conv1d(48, 64, kernel_size=(5,), stride=(5,), padding=(2,))
    (5): ReLU()
  )
  (rnn): LSTM(64, 64, bidirectional=True)
  (drop): Dropout(p=0.5)
  (fcl): Conv1d(128, 2, kernel_size=(1,), stride=(1,))
)
```

7.2.6 model6

```
HybridNet(
  (cnn): Sequential(
    (0): Conv1d(99, 36, kernel_size=(75,), stride=(2,), padding=(37,))
    (1): ReLU()
    (2): Conv1d(36, 64, kernel_size=(75,), stride=(2,), padding=(37,))
```

```
    (3): ReLU()
  )
  (rnn): LSTM(64, 64, bidirectional=True)
  (drop): Dropout(p=0.5)
  (fcl): Conv1d(128, 2, kernel_size=(1,), stride=(1,))
)
```

7.2.7 model7

```
HybridNet(
  (cnn): Sequential(
    (0): Conv1d(99, 36, kernel_size=(15,), stride=(2,), padding=(7,))
    (1): ReLU()
    (2): Conv1d(36, 64, kernel_size=(15,), stride=(2,), padding=(7,))
    (3): ReLU()
  )
  (rnn): LSTM(64, 64, bidirectional=True)
  (drop): Dropout(p=0.5)
  (fcl): Conv1d(128, 2, kernel_size=(1,), stride=(1,))
)
```

7.2.8 model8

```
HybridNet(
  (cnn): Sequential(
    (0): Conv1d(99, 36, kernel_size=(25,), stride=(2,), padding=(12,))
    (1): ReLU()
    (2): Conv1d(36, 64, kernel_size=(25,), stride=(2,), padding=(12,))
    (3): ReLU()
  )
  (rnn): LSTM(64, 64, bidirectional=True)
  (drop): Dropout(p=0.7)
  (fcl): Conv1d(128, 2, kernel_size=(1,), stride=(1,))
)
```

7.2.9 model9

```
HybridNet(
  (cnn): Sequential(
```

```
    (0): Conv1d(99, 64, kernel_size=(13,), stride=(4,), padding=(6,))
    (1): ReLU()
  )
  (rnn): LSTM(64, 64, num_layers=2, bidirectional=True)
  (drop): Dropout(p=0.7)
  (fcl): Conv1d(128, 2, kernel_size=(1,), stride=(1,))
)
```

7.2.10 model10

```
HybridNet(
  (cnn): Sequential(
    (0): Conv1d(99, 64, kernel_size=(13,), stride=(4,), padding=(6,))
    (1): ReLU()
  )
  (rnn): LSTM(64, 64, num_layers=2, bidirectional=True)
  (drop): Dropout(p=0.7)
  (fcl): Conv1d(128, 2, kernel_size=(1,), stride=(1,))
)
```

7.2.11 model11

```
HybridNet(
  (cnn): Sequential(
    (0): Conv1d(99, 16, kernel_size=(13,), stride=(4,), padding=(6,))
    (1): ReLU()
    (2): MaxPool1d(kernel_size=5, stride=5, padding=2, dilation=1, ceil_mode=False)
  )
  (rnn): LSTM(16, 16, bidirectional=True)
  (drop): Dropout(p=0.7)
  (fcl): Conv1d(32, 2, kernel_size=(1,), stride=(1,))
)
```

7.2.12 model12

```
HybridNet(
  (cnn): Sequential(
    (0): Conv1d(99, 8, kernel_size=(13,), stride=(2,), padding=(6,))
    (1): ReLU()
```

```
    (2): MaxPool1d(kernel_size=5, stride=5, padding=2, dilation=1, ceil_mode=False)
    (3): Conv1d(8, 16, kernel_size=(13,), stride=(2,), padding=(6,))
    (4): ReLU()
    (5): MaxPool1d(kernel_size=5, stride=5, padding=2, dilation=1, ceil_mode=False)
  )
  (rnn): LSTM(16, 16, bidirectional=True)
  (drop): Dropout(p=0.7)
  (fcl): Conv1d(32, 2, kernel_size=(1,), stride=(1,))
)
```