**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

###### Центр непрерывного образования

###### Факультета компьютерных наук

**ИТОГОВЫЙ ПРОЕКТ**

Определение вторичных структур белка по аминокислотной последовательности

|  |  |
| --- | --- |
|  |  |
|  | Выполнила: |
|  | Хотлянник Елена  Ф.И.О. |
|  | Руководитель: |
|  | Слинько Игорь  Ф.И.О. |

Москва 2022

Оглавление

[Введение 3](#_Toc111211276)

[Задачи проекта 3](#_Toc111211277)

[Общие сведения о предметной области 3](#_Toc111211278)

[Основные понятия 3](#_Toc111211279)

[Постановка задачи 4](#_Toc111211280)

[Сбор и подготовка данных для моделирования 5](#_Toc111211281)

[Источник информации о вторичных структурах 5](#_Toc111211282)

[Формирование набора данных и обработка 6](#_Toc111211283)

[Статистика по подготовленному датасету 7](#_Toc111211284)

[Построение многослойной нейронной сети 8](#_Toc111211285)

[Архитектура сети 8](#_Toc111211286)

[Оценка качества модели 8](#_Toc111211287)

[Эксперименты 9](#_Toc111211288)

[Эксперимент 1. На малом объеме данных. 9](#_Toc111211289)

[Эксперимент 2. На относительно небольшом объеме данных, batch = 10 10](#_Toc111211290)

[Эксперимент 3. На относительно небольшом объеме данных, batch = 50 11](#_Toc111211291)

[Эксперимент 4. На большом объеме данных, batch = 50 12](#_Toc111211292)

[Направления для развития 13](#_Toc111211293)

[Заключение 13](#_Toc111211294)

[Список источников 14](#_Toc111211295)

[Приложения 15](#_Toc111211296)

# Введение

Предсказание структуры белка (вторичной, третичной или четвертичной) является одной из самых важных целей биоинформатики и теоретической химии. Данные, полученные при помощи предсказания, применяются в медицине, фармацевтике и биотехнологии, при создании новых ферментов.

К настоящему моменту данная задача в мире решается с достаточно высокой точностью, но в рамки моего учебного проекта я не включала цель обойти самые современные модели и результаты. Я хотела разобраться в предметной области, собрать свой сет данных для обучения и, опираясь на одну из существующих статей, построить предсказывающую модель.

## Задачи проекта

* Предметная область. Ознакомиться с некоторыми аспектами молекулярной биологии.
* Разобраться с хранением данных, собрать свой сет данных.
* Построить нейронную сеть для предсказания вторичной структуры белка.
* Оценить качество построенной сети, попытаться улучшить качество.
* Сформулировать дальнейшие шаги по развитию модели.

## Общие сведения о предметной области

### Основные понятия

##### Альфа-аминокислоты

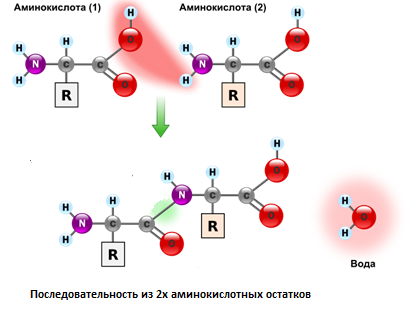
 Альфа-аминокислоты – органические соединения. Способны связываться в устойчивые длинные цепочки-последовательности.

Рисунок . Схема образования последовательности из двух альфа-аминокислот.

##### Белки

Белки - высокомолекулярные органические соединения, состоящие из состоящие из альфа-аминокислот.

Белок имеет строго определенную последовательность аминокислот в цепи. Большинство белков построено из 21 аминокислот. Каждую аминокислоту в последовательности можно представить символом. И, следовательно, каждую аминокислотную последовательность можно представить как последовательность символов. Для разных белков длина последовательности разная.

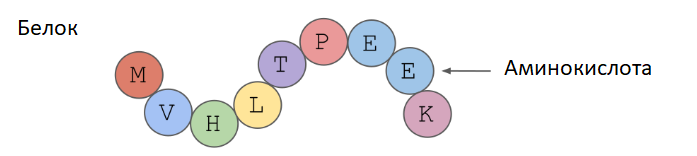


Рисунок Представление белка как последовательности символов

Пример представления белка как последовательности символов: MEIKDLKRLARYNPEKMAKIPVFQSERMLYDLYALLPGQAQKVHVHEGSDKVYYALEGEVVVRVGEEEALLAPGMAAFAPAGAPHGVRNESASPALLLVVTAPRP

(из файла pdb\_seqres.txt, v70\_A mol:protein length:105 probable antibiotics synthesis protein)

#### Пространственная структура

Белок имеет конкретную пространственную структуру, которая определяют его функцию. Принято выделять 4 уровня структурной организации белка.

* Первичная -  последовательность аминокислотных остатков
* Вторичная структура – то, как фрагмент последовательности организован в пространстве
* Третичная структура — пространственное строение всей молекулы белка
* Четвертичная структура — способ укладки в пространстве макромолекулярного образования

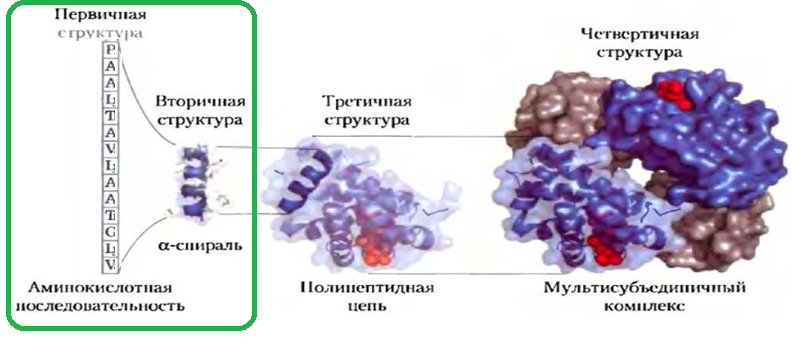


Рисунок Уровни структурной организации белка

### Постановка задачи

Постановка задачи: определить вторичные структуры в аминокислотной последовательности. Предсказание состоит в соотнесении отдельных участков аминокислотной последовательности с наиболее вероятными классами вторичных структур.

Наилучшие современные методы определения вторичной структуры белка достигают около 80 % точности.

Вторичных структур в каждой аминокислотной последовательности имеется некоторое количество. В [приложении 1](#_Приложение_1) на рисунке показан пример визуального представления элементов вторичной структуры поверх первичной последовательности аминокислотных остатков (с сайта EMBL's European Bioinformatics Institute).

# Сбор и подготовка данных для моделирования

## Источник информации о вторичных структурах

Для формирования своего сета данных я использую Worldwide Protein Data Bank ([ftp.rcsb.org](ftp://ftp.rcsb.org/)), который пополняется еженедельно (every Wednesday at 00:00 UTC).

Он организован как каталог архивных файлов. 1 протеин – это 1 архивный файл.

В файлах PDB представлены вторичные структуры

* 10 типов Helix (Спирали): 1 Right-handed alpha (default), 2 Right-handed omega, 3 Right-handed pi, 4 Right-handed gamma, 5 Right-handed 3 – 10, 6 Left-handed alpha, 7 Left-handed omega,  8 Left-handed gamma, 9 2 - 7 ribbon/helix, 10 Polyproline
* Beta-sheet (β-листы, складчатые слои)

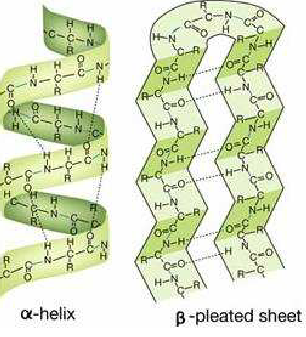


Рисунок Схематичное изображение вторичных структур типа Helix и Beta-sheet

Среди прочих данных о протеине, в каждом файле есть раздел с перечнем вторичных структур. В [приложении 2](#_Приложение_2) представлен пример со списком вторичных структур из одного из bpd файлов.

В данных, получаемых из bpd файлов в рамках учебного проекта, в значимом количестве представлены 3 типа вторичных структур:

1 Helix Right-handed alpha (далее, Helix1)

5 Helix Right-handed 3 – 10 (далее, Helix5)

Beta-sheet (далее, Sheet)

Описание формата данных в pdb-файлах в части описания вторичных структур приведено в [приложении 3](#_Приложение_3).

## Формирование набора данных и обработка

Для формирования набора данных я выполняю следующие шаги:

1. Загрузка pbd файлов с сайта. Для этого обращаюсь к сайту (посредством selenium.webdriver), организую навигацию по сайту, анализирую содержимое листов каталога (bs4.BeautifulSoup), загружаю файлы из листов каталога.

При этом я устанавливаю максимальное количество листов каталога, по которым прохожусь, и максимальное количество файлов для скачивания в каждом каталоге. И таким образом определяю количество белков, которое попадет в набор данных.

1. Работа с файлами, аккумуляция данных в таблицу.

Этот шаг включает работу с файлами в директориях (последством библиотку os, shutil), разархивацию (gzip.GzipFile), добавление данных из файлов каталога в таблицу, разбор строк данных (pandas), отбор подходящих белков (например, я убираю белки, вторичная структура которых определена с помощью теоретической модели), добавление атрибутов (в т. ч. последовательность символов из pdb\_seqres.txt).

Пример подготовленного базового датасета (шаги1-2) приведен в [приложении 4](#_Приложение_4).

Далее, я переношу эти данные в Яндекс Data Sphere, и продолжаю подготовку данных там.

1. Обработка данных

Все последовательности разной длины, поэтому я задаю максимальную длину последовательности, равную 700 символам, и привожу все последовательности к одному размеру (обрезаю или дополняю специальным пустым символом).

Затем считаю, какое количество уникальных символов встречается в получившемся датасете, и, ставя каждому символу в соответствие порядковый номер, перевожу последовательность из набора символов в набор чисел. Получаю X, датасет с атрибутами-предикторами. Пример датасета X приведен в [приложении 4](#_Приложение_4).

Также подготавливаю PSSM (Position specific scoring matrices). Для формирования PSSM сначала для каждого символа считается доля присутствия символа на каждой позиции в датасете. Также считается доля присутствия символа в датасете. Первое делится на второе, таким образом нормализуется. Потом это значение логарифмируется по основанию 2. Получается таблица, с рассчитанным значением для каждого символа на каждой позиции. И, далее, полученные значения распространяются на весь датасет. Схематично последовательность расчёта PSSM представлена в [приложении 5](#_Приложение_5).

По сути, формирование матрицы на базе PSSM нужно для формирования атрибутов глобального контекста. В статье “ PSSM-based prediction of DNA binding sites in proteins ” (авторы ShandarAhmad & AkinoriSarai) предполагается, что использование PSSM повышает точность предсказания на 8,7%.

Таким образом, предикторы для расчета модели содержаться в датасетах X и PSSM.

Для формирования target’a (y), я обозначаю типы вторичной структуры через нули и единицы. В сформированном базовом датасете представлено 3 разных класса вторичных структур (Helix1, Helix5, Sheet) к которым добавляю 4-ый класс, для типа “другое” (Other - когда не вторичная структура). И делаю one-hot-encoding, после чего каждому символу в последовательности соответствует набор из 3 нулей и одной единицы на позиции, соответствующей классу.

Пример датасета y приведен в [приложении 4](#_Приложение_4).

1. Подготовка данных на вход модели.

Собираю класс на базе torch.utils.data.Dataset. Внутри класса: X, y, PSSM, и функция получения экземпляра getitem. Делаю transforms.ToTensor. И на базе DataLoader делаю Train\_loader и Test\_loader (80\20), которые буду подавать на вход сети.

После подготовки данных, запускаю модель.

## Статистика по подготовленному датасету

* Объем датасета (количество структур) = 370000 строк
* Количество протеинов = 24 660
* Максимальная длина последовательности = 1600
* Количество вторичных структур:
  + helix1 = 176 111,
  + helix5 = 61 106,
  + sheet = 132 758
* Суммарное количество символов по классам:
  + helix1 = 2 153 050,
  + helix5 = 225 881,
  + sheet = 593 939
  + other = 14 289 130
* Сколько типов вторичных структур представлено (количественно значимого объема) = 3
* Самый большой класс – helix1, самый малый класс – helix5
* Количество аминокислот, встречающихся в последовательностях = 23

# Построение многослойной нейронной сети

## Архитектура сети

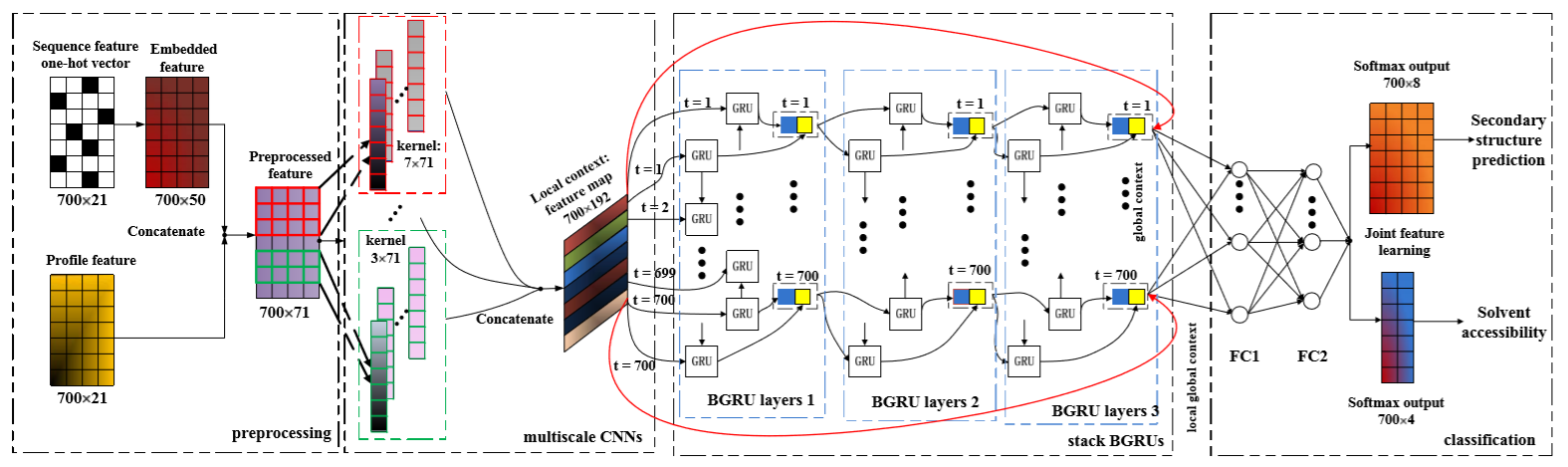
Архитектуру сети я выбрала из статьи 2016 года “Protein Secondary Structure Prediction Using Cascaded Convolutionaland Recurrent Neural Networks” Zhen Li, Yizhou Yu Department of Computer Science, The University of HongKong. 

Рисунок 5 Архитектура сети DCRNN для предсказания вторичных структур белка (deep convolutional and recurrent neural network)

В статье есть хорошая схема архитектуры и её описание. А код я составляла сама, по этой архитектуре, с небольшими изменениями. Например, авторы подают на вход таблицу с профайлами белков, которую берут из программы PSY\_BLAST, вместо нее я подаю подготовленную таблицу PSSM. И, т.к. PSSM разреженная, добавляю embedding.

Т.е. на вход подается X и PSSM, потом embedding, и потом объединенные данные передаются в 3 сверточных слоя с ядрами 3, 7, и 11. Сверточные слои по сути применяются для извлечения локальных особенностей.

Выходы сверточных слоев объединяются и подаются на вход стеку BGRU слоев, 3 слоев. BGRU (*Bidirectional Gated Recurrent Unit*) важны для обработки глобального контекста.

Данные после BGRU проводятся через два полносвязных слоя для мульти классификации.

Относительно представленной на рисунке схемы у меня в коде есть изменение размерностей. Размерности приведены в коде, представленном в [приложении 6](#_Приложение_6).

## Оценка качества модели

В качестве ошибки используется torch.nn.CrossEntropyLoss, с весами обратно пропорциональными объему класса.

В качестве метрики качества – точность - Torchmetrics.Accuracy

* Общая взвешенная точность: Accuracy(average = 'weighted', num\_classes = 4)
* Точность отдельно по классам Accuracy(average = 'none', num\_classes = 4, mdmc\_average = 'samplewise')

Ориентиры по точности (из статьи):

* 69.7% Q8 accuracy on the public benchmark CB513,
* 76.9% Q8 accuracy on CASP10,
* 73.1% Q8accuracyonCASP11

# Эксперименты

В рамках проекта было сделано некоторое количество экспериментов с разным объемом данных, разными размерами батча, разной скоростью обучения, подаваемых в нейронную сеть. Также в части экспериментов устанавливалась максимальная длина последовательности = 500, в другой части = 700. Также эксперименты проводились с разными весами классов для расчета ошибки.

Все эксперименты выполнялись в среде Яндекс Data Sphere. В зависимость от объема данных и количества эпох эксперименты продолжались от нескольких часов до нескольких суток.

Ниже приведены данные по 4 экспериментам. В таблицах для каждого эксперимента указаны входные параметры, средняя ошибка и точность по эпохам. Также для экспериментов 2-4 приведены графики, отражающие эпохальную динамику средних ошибки, взвешенной точности, точностей по классам.

В целом следует отметить, что в каждом эксперименте ошибка уменьшается. При этом взвешенная точность в последних экспериментах достигает 76% и выше, что соответствует ожидаемой точности из статьи.

## Эксперимент 1. На малом объеме данных.

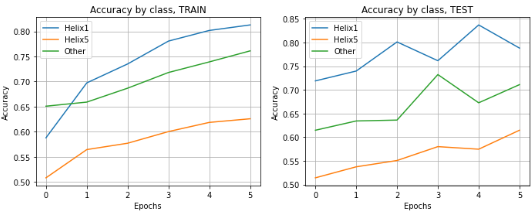
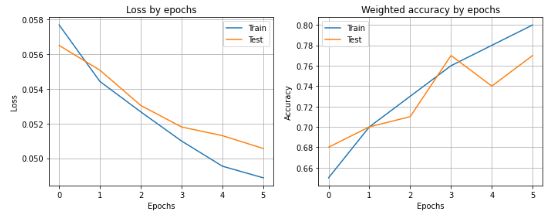
|  |  |  |  |
| --- | --- | --- | --- |
| Входные параметры | | Batch size = 10,  Count of protein = 1098,  Learning rate = 0.5,  Max protein length = 500 | |
| Epoch | 0 | 1 | 2 |
| Loss | Train 0.09596  Test 0.09706 | Train 0.09532  Test 0.09701 | Train 0,09568  Test 0.09698 |
| Accuracy | Train 0.70  Test 0.71 | Train 0.70  Test 0.71 | Train 0,7  Test 0,71 |

Выводы по результатам первых экспериментов:

* Следует уменьшить скорость обучения
* Необходима корректировка весов при расчете loss
* Попробовать увеличить размер batch для обеспечения возможности делать расчеты на большом объеме данных

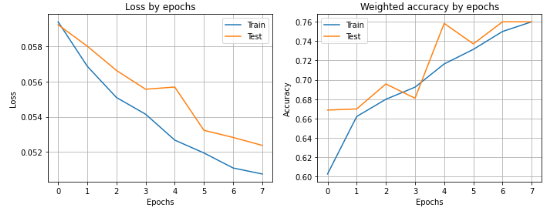
## Эксперимент 2. На относительно небольшом объеме данных, batch = 10

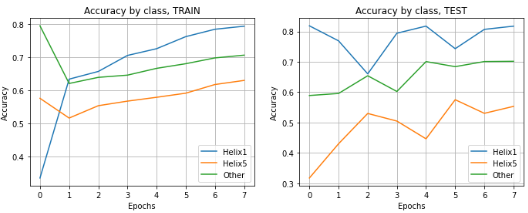
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Входные параметры | | Batch size = 10,  Count of protein = 2191,  Learning rate = 0.0003,  Max protein length = 500  2 типа вторичных структур (helix1, helix2) \ 3 класса (h1, helix2, other)  weights = [0.1, 0.88, 0, 0.02] | | | | | |
| Epoch | | 0 | 1 | 2 | 3 | 4 | 5 |
| Loss | | Train: 0.05768  Test : 0.05650 | Train: 0.05442  Test: 0.05507 | Train: 0.05267  Test:  0.05304 | Train: 0.05099  Test: 0.05180 | Train: 0.04954  Test: 0.05130 | Train: 0.04887 Test: 0.05056 |
| Accuracy | | Train: 0.65  Test: 0.68 | Train: 0.70  Test: 0.70 | Train: 0.73  Test: 0.71 | Train: 0.76  Test: 0.77 | Train: 0.78  Test: 0.74 | Train: 0.80  Test: 0.77 |
| Acc by class | Train | 0.588, 0.508, 0.651 | 0.697, 0.564, 0.659 | 0.697, 0.564, 0.659 | 0.781, 0.600, 0.718 | 0.802, 0.619, 0.739 | 0.813, 0.626, 0.761 |
| Test | 0.719, 0.515, 0.615 | 0.740, 0.537, 0.634 | 0.801, 0.551, 0.636 | 0.762, 0.580, 0.732 | 0.837, 0.575, 0.673 | 0.788, 0.615, 0.711 |



## Эксперимент 3. На относительно небольшом объеме данных, batch = 50

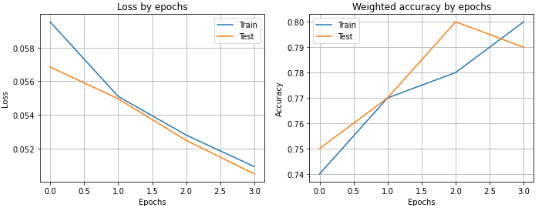
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Входные парам | | Batch size = 50,  Count of protein = 2191,  Learning rate = 0.0003,  Max protein length = 500  2 типа вторичных структур (helix1, helix2) \ 3 класса (h1, helix2, other)  Weights = [0.1, 0.88, 0, 0.02] | | | | | | | |
| Epoch | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Loss | | Train: 0.0593  Test:  0.0592 | Train: 0.0569  Test: 0.058 | Train:  0.0551  Test:  0.0566 | Train: 0.0542  Test: 0.0556 | Train: 0.0527  Test: 0.0557 | Train: 0.052  Test: 0.0532 | Train: 0.0511  Test: 0.0528 | Train: 0.0508  Test: 0.0524 |
| Accuracy | | Train:  0.6027  Test:  0.6688 | Train:  0.6621  Test:  0.66991 | Train:  0.6799  Test:  0.6958 | Train: 0.6924  Test: 0.681 | Train: 0.716  Test: 0.7584 | Train: 0.7315  Test: 0.7372 | Train:  0.75  Test:  0.76 | Train:  0.76  Test:  0.76 |
| Acc by class | Train | 0.335, 0.576,  0.794 | 0.633, 0.516, 0.620 | 0.657, 0.553, 0.639 | 0.705, 0.567, 0.646 | 0.725, 0.579, 0.666 | 0.762, 0.591, 0.680 | 0.784, 0.617, 0.696 | 0.793, 0.630, 0.706 |
| Test | 0.819, 0.317, 0.589 | 0.769, 0.430, 0.596 | 0.660, 0.530, 0.654 | 0.794, 0.505, 0.602 | 0.818, 0.446, 0.701 | 0.744, 0.576, 0.685 | 0.807, 0.530, 0.701 | 0.818, 0.553, 0.702 |

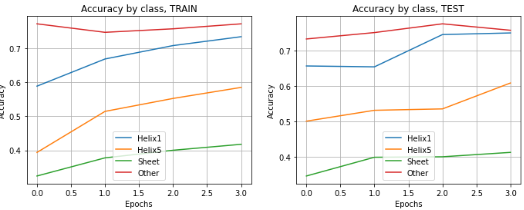




## Эксперимент 4. На большом объеме данных, batch = 50

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Входные параметры | | Batch size = 50,  Count of protein = 24 660,  Learning rate = 0.0003,  Max protein length = 700  3 типа вторичных структур (helix1, helix2, sheet) \ 4 класса (helix1, helix2, sheet, other)  Weights = [0.1,0.76,0.29,0.02] | | | |
| Epoch | | 0 | 1 | 2 | 3 |
| Loss | | Train: 0.05953  Test: 0.05685 | Train: 0.05510  Test: 0.05496 | Train: 0.05280  Test: 0.05249 | Train: 0.05093  Test: 0.05049 |
| Accuracy | | Train: 0.74  Test: 0.75 | Train: 0.77  Test: 0.77 | Train: 0.78  Test: 0.80 | Train: 0.80  Test: 0.79 |
| Accuracy by class | Train | 0.589, 0.394, 0.325, 0.772 | 0.668, 0.514, 0.378, 0.747 | 0.708, 0.553, 0.400, 0.757 | 0.734, 0.585, 0.418, 0.772 |
| Test | 0.656, 0.501, 0.347, 0.732 | 0.654, 0.532, 0.400 , 0.750 | 0.745, 0.536, 0.401, 0.775 | 0.750, 0.609, 0.413, 0.753 |





# Направления для развития

Идеи для продолжения работы с задачей:

* поработать с весами для улучшения точности наименее представленного типа вторичной структуры helix5
* добавить предикторы для класса sheet (по этому классу есть что добавить, в отличии от helix1 и helix5)
* поработать c аномалиями. Например, убрать из данных белки, у которых в последовательности 23 уникальных аминокислоты
* поработать с длиной последовательности: не обрезать все последовательности одинаково, а формировать батчи для групп последовательностей с сопоставимой длиной. Это поможет сделать классы несколько более сбалансированными за счет уменьшения класса other
* перейти от классификации к детекции
* экспериментировать с архитектурой слоев

# Заключение

В результате выполнения учебного проекта удалось достигнуть поставленные задачи:

* ознакомится с предметной областью и постановкой
* разобраться с хранением информации о вторичных структурах и собрать свой датасет для обучения
* построить работающую с ожидаемой точностью модель
* провести эксперименты с разными параметрами
* обозначить направления для дальнейшей работы

# Список источников

1. Protein Secondary Structure Prediction Using Cascaded Convolutionaland Recurrent Neural Networks. ZhenLi, YizhouYu Department of Computer Science, The University of HongKong, 2016. https://i.cs.hku.hk/~yzyu/publication/PSSP-DCRNN-ijcai2016.pdf - схема на сладе 13
2. https://ru.wikipedia.org
3. Наглядная биохимия. Кольман Я., Рём К. – Г. , http://www.chem.msu.su/rus/teaching/kolman
4. Архив pbd- файлов: https://ftp.rcsb.org/pub/pdb/data/structures/
5. К. В. Рудаков, И. Ю. Торшин, Вопросы разрешимости задачи распознавания вторичной структуры белка, http://www.mathnet.ru/links/6d8b47aa2a91256ede3eb56ac29f00c8/ia133.pdf
6. Избранные главы к лекционному курсу «Биофизическая химия» / Автор - составитель: В.А. Сироткин. – Казань: Казанский университет, 2011. https://kpfu.ru/docs/F301777323/Biophysical.chemistry.pdf
7. https://nplus1.ru/material/2020/12/10/alphafold-wat
8. PSSM-based prediction of DNA binding sites in proteins. Shandar Ahmad & Akinori Sarai, https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-6-33
9. https://www.researchgate.net/figure/Abbreviations-BGRU-Bidirectional-gated-recurrent-unit-BLSTM-Bidirectional-long\_fig2\_326822081
10. http://propionix.ru/aminokisloty-osnovnyye-ponyatiya - источник для рисунка 1
11. https://mitt.uib.no/courses/267/assignments/466 - источник для рисунка 2
12. https://lifelib.info/biochemistry/leninger/20.html - источник для рисунка 3
13. https://www.researchgate.net/figure/Protein-secondary-structure-showing-a-helix\_fig4\_282790336/download - источник для рисунка 4
14. http:www.ebi.ac.uk/thornton-srv/databases/cgi-bin/pdbsum/GetPage.pl?pdbcode=1dd3&template=protein.html&l=2&chain=C&r=wiring// , http://www.ebi.ac.uk/thornton-srv/databases/cgi-bin/pdbsum/GetPage.pl?pdbcode=n/a&template=doc\_promotif.html - источник для рисунка в приложении 1

# Приложения

#### Приложение 1

На рисунке 4 показан пример визуального представления элементов вторичной структуры поверх первичной последовательности аминокислотных остатков (с сайта EMBL's European Bioinformatics Institute). В нем размечены и визуализированы вторичные структуры. Например, с 50 по 60 позицию мы видим вторичную структуру HELIX (H1).

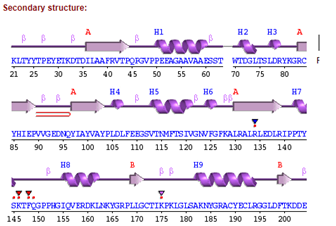


Рисунок Пример визуального представления вторичных структур поверх аминокислотной последовательности

#### Приложение 2

Среди прочих данных о протеине в каждом файле есть раздел про вторичные структуры.

Вот пример из одного файла.

Здесь указано, что с 5го по 10 символ имеет место структура HELIX с типом 1. И с 18 по 25 символ имеет место структура SHEET (элемент бетта-решетки)

HELIX 1 1 ASP A 5 ALA A 10 1

SHEET 1 A 5 ALA A 18 SER A 25 0

SHEET 2 A 5 MET A 28 LEU A 35 -1 O LEU A 32 N ILE A 20

SHEET 3 A 5 ALA A 95 ALA A 102 -1 O VAL A 99 N ASP A 31

SHEET 4 A 5 ASP A 50 GLU A 57 -1 N ASP A 50 O ALA A 102

SHEET 5 A 5 ALA A 76 ALA A 79 -1 O ALA A 79 N LYS A 51

SHEET 1 B 4 ALA A 40 HIS A 44 0

SHEET 2 B 4 HIS A 85 ARG A 88 -1 O VAL A 87 N GLN A 41

SHEET 3 B 4 VAL A 60 VAL A 64 -1 N VAL A 61 O ARG A 88

SHEET 4 B 4 GLU A 67 LEU A 71 -1 O LEU A 71 N VAL A 60

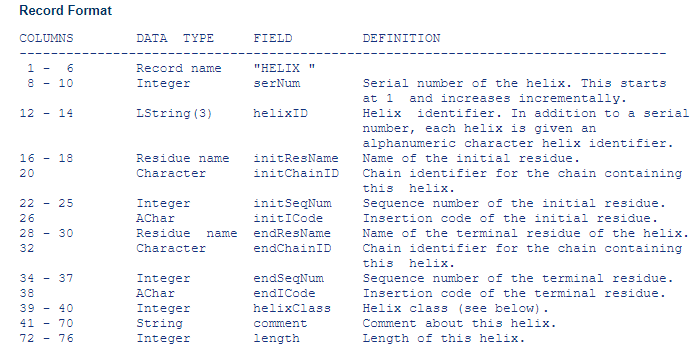
Среди прочих данных о протеине в каждом файле есть раздел про вторичные структуры.

Вот пример из одного файла.

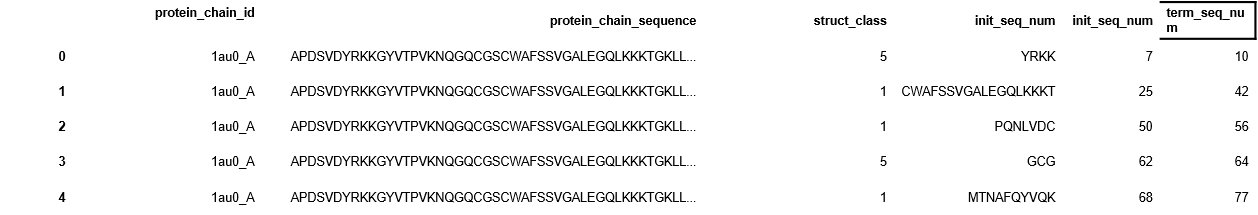
Здесь указано, что с 5го по 10 символ имеет место структура HELIX с типом 1. И с 18 по 25 символ имеет место структура SHEET (элемент бетта-решетки

#### Приложение 3

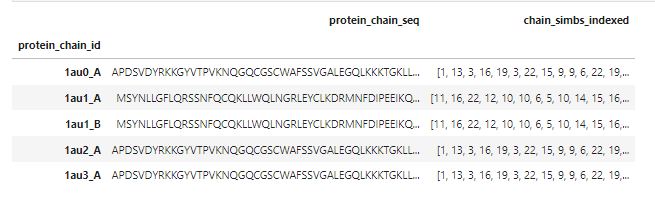
Описание формата данных в pdb-файлах в части описания вторичных структур:



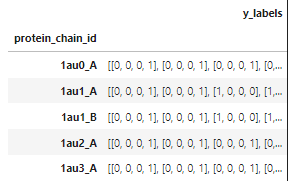
#### Приложение 4

Первые 5 записей подготовленного базового датасета

Первые 5 записей подготовленного датасета X. Предиктор – поле chain\_simbs\_indexes.

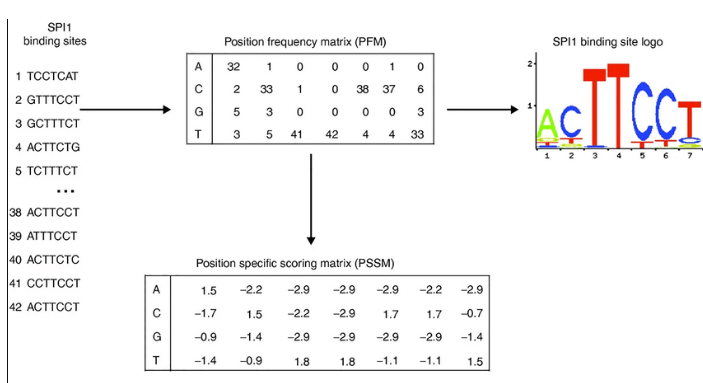


Первые 5 записей подготовленного датасета y:



#### Приложение 5

Схема расчета PSSM матрицы:



#### Приложение 6

Код с архитектурой ВСRNN:

class ChainModel(nn.Module):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.x\_embedding = nn.Embedding(24, 47)

self.pssm\_embedding = nn.Embedding(24, 24)

self.conv3 = nn.Sequential(nn.Conv1d(in\_channels=71, out\_channels=68, kernel\_size=3, padding=1), nn.ReLU())

self.conv7 = nn.Sequential(nn.Conv1d(in\_channels=71, out\_channels=64, kernel\_size=7, padding=3), nn.ReLU())

self.conv11 = nn.Sequential(nn.Conv1d(in\_channels=71, out\_channels=60, kernel\_size=11, padding=5), nn.ReLU())

self.iL1= 192 # self.input\_size\_encoder

self.oL1= 192 # self.hidden\_size

self.iL2= self.iL1\*2 + self.oL1\*2 #(Bidirectional)

self.oL2= self.iL2

self.iL3= self.oL2\*2 + self.oL1\*2 + self.iL1\*2 #(Bidirectional)

self.oL3= self.iL3

self.gru1 = nn.Sequential(nn.Dropout(p=0.2, inplace=False), nn.GRU(input\_size=self.iL1, hidden\_size=self.oL1, batch\_first=True, bidirectional=True))

self.gru2 = nn.Sequential(nn.Dropout(p=0.2, inplace=False), nn.GRU(input\_size=self.iL2, hidden\_size=self.oL2, batch\_first=True, bidirectional=True))

self.gru3 = nn.Sequential(nn.Dropout(p=0.2, inplace=False), nn.GRU(input\_size=self.iL3, hidden\_size=self.oL3, batch\_first=True, bidirectional=True))

self.detection = nn.Sequential( nn.Linear(4608, 128), nn.BatchNorm1d(max\_protein\_length),

nn.Linear(128, 4), nn.BatchNorm1d(max\_protein\_length), nn.Softmax(dim = 2) )

def forward(self, x, PSSM):

x\_emb = self.x\_embedding(x)

PSSM\_emb = self.pssm\_embedding(x)

x\_emb = torch.cat((x\_emb.float(), PSSM\_emb.float()),dim=2)# print("x\_emb",x\_emb)

x\_emb = x\_emb.permute(0,2,1)

#print(x\_emb.shape)

#print(x\_emb)

xc3 = self.conv3(x\_emb) # barch\_size X 700 X 71 -> barch\_size X 700 X 68

xc7 = self.conv7(x\_emb)

xc11 = self.conv11(x\_emb)

x\_conv = torch.cat((xc3, xc7, xc11),1)

x\_conv = x\_conv.permute(0,2,1)

output1, h\_n1 = self.gru1(x\_conv)

fw1\_res = torch.cat((x\_conv,output1[:,:,self.oL1:]), dim=2) # x,L1

bw1\_res = torch.cat((output1[:,:,:self.oL1],x\_conv) ,dim=2) # L1,x

output1\_residual = torch.cat((fw1\_res,bw1\_res),dim=2)

output2, h\_n2 = self.gru2(output1\_residual)

fw2\_res = torch.cat((x\_conv,output1[:,:,self.oL1:],output2[:,:,self.oL2:]),dim=2) # x,L1,L2

bw2\_res = torch.cat((output2[:,:,:self.oL2],output1[:,:,:self.oL1],x\_conv),dim=2) # L2,L1,x

output2\_residual = torch.cat((fw2\_res,bw2\_res),dim=2)

output3, h\_n3 = self.gru3(output2\_residual)

x\_detect = self.detection(output3)

return x\_detect