Москва 2020

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

Применение методов машинного обучения для предсказания пространственной структуры белков

Выполнил студент группы 162, 4 курса,
Чистяков Глеб Игоревич

Руководитель ВКР:
Доктор физико-математических наук, доцент
Посыпкин Михаил Анатольевич

Оглавление

Аннотация

Предсказание трехмерной структуры белка является одной из важнейших задач в биоинформатике. Оно способствует пониманию взаимодействия нескольких белков, в результате чего становится возможным изобретение новых лекарственных средств. С развитием глубинного обучения моделирование процесса сворачивания белка (фолдинга) вышло на новый уровень, так как экспериментальные методы требуют много времени и финансов. В данной работе описываются способы предсказания класса двугранных углов α крупноблочной модели белка, которые впоследствии можно использовать в качестве начального приближения для построения 3D структуры аминокислотной последовательности. Для бинарной классификации удалось добиться показателя F-меры 0.88, для трехклассовой – 0.79, для четырехклассовой – 0.75.

Abstract

Predicting the 3D structure of proteins is one of the most important issues in bioinformatics. Having knowledge about protein conformation, it becomes possible to study the interaction of several proteins and consequently invent new medicines. With the development of machine learning algorithms, folding prediction has reached a new level, since experimental methods require a lot of time and money. This paper describes application of ML methods to predicting the class of dihedral angles α of a coarse-grained protein model, which can be further used as an initial approximation for constructing the 3D structure of the amino acid sequence. The achieved results are 0.88 F1-score for the binary classification, 0.79 for the three-class classification, 0.75 for the four-class classification.

Ключевые слова

Глубинное обучение, двугранные углы, пространственная структура белка, нейросетевые модели, фолдинг белка

1. Введение

Белки играют одну из ключевых ролей во всех биологических процессах: они выполняют транспортную функцию в организме, передают сигналы в иммунную систему, обеспечивают целостность клетки и многое другое. Функции белка зависят от его уникального строения. Всего выделяют 20 различных аминокислот, которые образуют белок. Они составляют последовательность, в которой образуются пептидные связи между карбоксильной группой одной аминокислоты и аминогруппой соседней аминокислоты.

На протяжении последних десятилетий внимание российских и зарубежных ученых было приковано к определению пространственного строения белка. В середине 1980-х были предприняты попытки использования вычислительной физики для изучения полипептида из 5 аминокислот [1]. Десятилетие спустя началось развитие методов выявления структур гомологичных последовательностей белка [2], а также определения структуры последовательности неизвестного белка с помощью проецирования на 3D структуру известного белка [3].

На данный момент предсказание конформации белка по его аминокислотной последовательности считается одной из важнейших целей вычислительной биологии. Понимание структуры белка обеспечивает возможность создания новых лекарств, но определение строения неизвестных белков экспериментальным путем требует больших временных затрат и финансовых вложений. Именно поэтому все больший упор делается на прогнозирование структуры белка с помощью компьютерной симуляции.

С развитием Protein Data Bank (PDB), где размещается большое количество экспериментально определенных структур, предсказание трехмерной структурны белка стало такой же проблемой машинного обучения, как и проблема определения пространственного строения с помощью вычислительной физики.

В данной работе речь пойдет о предсказании двугранных углов α основного каркаса в крупноблочной модели белка. В рамках крупноблочной модели аминокислота обобщается до ее условного центра – альфа-углерода. Это позволит обучающей модели не обращать внимание на небольшие различия между похожими белками, что сделает обучение более эффективным. Зная значения углов α, впоследствии можно использовать их в качестве начального приближения для аппроксимации пространственной структуры.

В силу того, что предсказать величину угла удается с отклонением по модулю в 0.8 радиан, было решено рассмотреть задачу классификации. В данном случае она корректна, так как значения угла α в основном кластеризуются около 50 и 210 градусов. При решении этой задачи была рассмотрена двух-, трех- и четырехклассовая классификация. После того, как модель произвела предсказание класса, осуществлялось сопоставление данного класса и медианного значения углов этого класса, которые впоследствии участвовали в качестве начального приближения.

Всего было разработано 10 моделей глубинного обучения, в основе которых лежали одномерные и двумерные сверточные нейронные сети (Conv1d и Conv2d), а также двунаправленные рекуррентные нейронные сети с длительной краткосрочной памятью (BiLSTM). Лучший результат на всех видах классификации показала модель, состоящая из последовательных BiLSTM слоев. Для двухклассовой классификации F-мера составила 0.88, для трехклассовой – 0.79, и для четырехклассовой – 0.75.

2. Обзор литературы

Проблема определения трехмерной формы белка до сих пор актуальна, так как именно структура по большей части характеризует функцию белка. Благодаря предсказанию структуры появляется возможность составить 3D форму белка, зная только последовательность составляющих его аминокислот [4]. При определении конформации особую роль играет генетическая

информация, облегчающая процесс нахождения аминокислотных контактов, на основе которых и работают современные методы.

Даже относительно маленькие белки, состоящие из 40-50 аминокислот, имеют порядка 600-800 атомов и более 150-200 степеней свободы. Чтобы рассчитать энергию взаимодействия между тепловым движением частиц белка и движением молекул растворителя, требуются большие вычислительные мощности. Для решения этой проблемы был придуман метод «упрощения» белка [5]. Первый шаг к упрощению заключается в усреднении по мелким деталям. Это повышает эффективность расчетов, в которых белки отличаются незначительными деталями, так как происходит обобщение конформации. Второй шаг упрощения симуляции свертки белка происходит посредством уменьшения конвергентной энергии, а также термизации в нормальном режиме, что обеспечивает невосприимчивость к непроизвольным природным колебаниям, ускоряющую процесс вычислений.

Создание упрощенной структуры белка, где каждый аминокислотный остаток представи́м в виде атома карбоксильного углерода и центра тяжести боковой цепи, достигается усреднением большей части его тонкой структуры. Таким образом, происходит получение модели из соединенных аминокислотных остатков, каждому из которых соответствует только одна степень свободы.

Отметим, поначалу упрошенную структуру использовали для определения свертки белка, аппроксимируя уравнения молекулярной динамики уравнениями Ланжевена, где учитывалось броуновское движение в спуске по градиенту. В силу того, что исследователи столкнулись с вычислительными трудностями при данном методе определения свертки, тепловые отклонения перестали учитываться, а сворачивание стало производиться за счет минимизации потенциальной энергии системы. Однако после фолдинга возобновился учет теплового отклонения и минимизирование энергии. Считалось, что конформация колеблется вокруг оптимума, а термоизоляция, таким образом, повышает продуктивность фолдинга.

Также для определения структуры белка был разработан метод, основанный на построении шаблонов [6]. Этот метод является полностью автоматизированным и базируется на том, что сворачивание белка представляет иерархический процесс [7]. Предсказание структура белка при данном методе происходит в два шага: сначала определяется сопоставление между свертками коротких подпоследовательностей аминокислот и частями целевой последовательности для определения соответствующей структуры, а затем локальные блоки собираются в единую трехмерную структуру. Этот метод доказал свою эффективность в соревнованиях моделей по предсказанию структуры белка CASP6 и CASP7.

На настоящий момент существует множество методов машинного обучения для предсказания пространственной структуры белка. Один из примеров его применения – сверточная нейронная сеть, предсказывающая вторичную структуру, матрицу контактов и двугранные углы, которые далее участвуют в определении 3D состояния белка.

Перейдем к рассмотрению системы A7D, включающей комплекс нейросетевых моделей. Система A7D, предложенная компанией AlphaFold, превзошла многие конкурирующие модели, показав лучший результат на CASP13 [8]. В отличие от ранее описанного метода [6], эта модель не использует шаблонное моделирование.

Методы предсказании структуры белка AlphaFold состоят из сочетания трех нейронных сетей, где первая – предсказывает матрицу расстояний между остатками и двугранные φ- и ψ-углы, вторая, так называемая Global Distance Test (GDT) сеть [9], – оценивает точность структуры, и третья – создает структуру белка по подаваемым на вход параметрам. Первая и вторая сети используются для определения потенциала остатков, которые впоследствии оптимизируются для определения структуры белка. Сами методы предсказания устроены следующим образом:

1) имитационный отжиг с генерацией нейронных фрагментов, примененный к матрице расстояний остатков и двугранным углам;

2) имитационный отжиг с генерацией нейронных фрагментов, примененный к потенциалам GTD-сети;
3) градиентный спуск по матрице расстояний остатков и двугранным углам.

Приведенные выше методы показывают приблизительно одинаковые результаты, отмечается несущественное отставание качества первого метода от второго и третьего. На данный момент — это лучшие способы предсказания конформации.

3. Данные

3.1 Загрузка данных

Данные для обучения моделей были отобраны из датасета соревнования CASP12. Этот набор включает в себя следующую информацию:

1. Последовательность первичной аминокислотной цепи – строка символов с размером алфавита 20.
2. Позиционная весовая матрица (PSSM) – матрица коэффициентов, где у каждой аминокислоты в последовательности имеется 21-мерный вектор (20 – мощность алфавита аминокислот, 1 – измерение для информационного содержания остатков). PSSM показывает, насколько последовательность соответствует мотиву, по которому была создана позиционная матрица весов, то есть насколько сильно последовательность склонна изменяться в процессе эволюции.
3. Третичная структура – координаты в трехмерном пространстве для каждого атома. На основе этих координат будет формироваться целевая переменная – двугранный угол α.
4. Маска последовательности – индикатор наличия атомных координат в третичной структуре. Маска последовательности необходима, так как многие структуры белка из-за внутренних или экспериментальных причин не имеют точно определенных положений для всех атомов.

При считывании данных отбираются белки с известными координатами альфа-углерода, так как они понадобятся при расчете двугранных углов α. Таким образом, для формирования датасета используются 21741 белков.

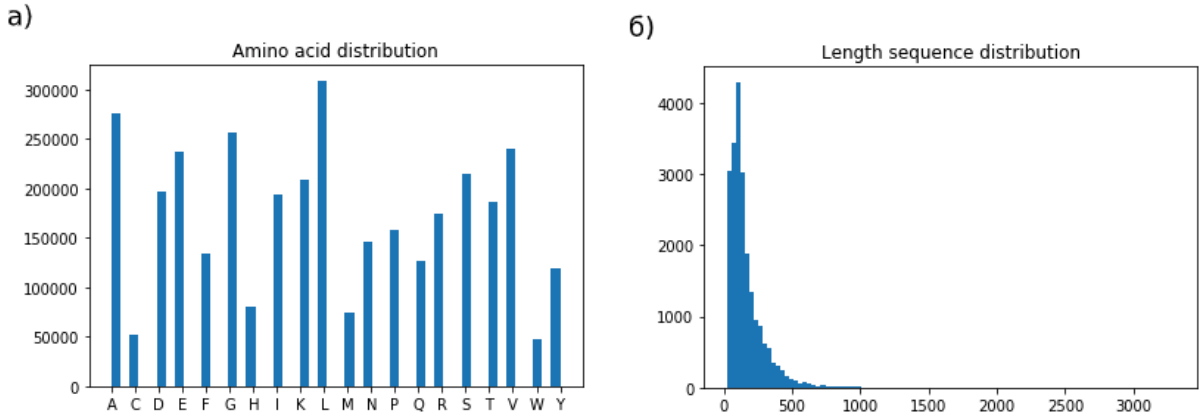

Рисунок 1. а) Распределение аминокислот последовательностей отобранной выборки белков б) Распределение длин последовательностей отобранной выборки белков

На Рисунке 1 (а) изображено распределение по отдельным аминокислотам отобранной выборки. Как видно на графике, в последовательностях преобладает Лицин (L) и Аланин (A), в то время как Цистеин (C) и Триптофан (W) встречаются редко. На Рисунке 1 (б) отображено распределение по длинам последовательности. Можно заметить, что некоторые белки состоят из приблизительно 3000 аминокислот – такие объемные белки не будут рассматриваться при обучении моделей.

На следующем этапе загрузки данных происходит расчет двугранных углов α для каждой последовательности. Это делается по четырем координатам альфа-углерода, находящихся в последовательных аминокислотах.

3.2 Формирование обучающей выборки

Из отобранных белков составляется выборка, на которой будут обучаться и тестироваться модели. В нее входят белки, длины которых не превосходят 220 аминокислот, в силу того, что длинные цепи имеют очень сложную структуру. Далее последовательности разбиваются на подпоследовательности для упрощения обучения. Для длины подпоследовательности, равной 32, размер обучающей выборки составил

39780 объектов, а тестовой выборки 7020 объектов. Стоит заметить, что до разделения на подпоследовательности убираются первая и последняя аминокислоты. Это связано с тем, что двугранный угол высчитывается по четырем точкам, соответственно, нельзя сказать, какой угол был между первой и последней парой аминокислот в последовательности. После этого каждая аминокислотная подпоследовательность кодируется с помощью One Hot Encoder.

Таким образом, обучающая выборка состоит из закодированных подпоследовательностей аминокислот и их позиционных весовых матриц.

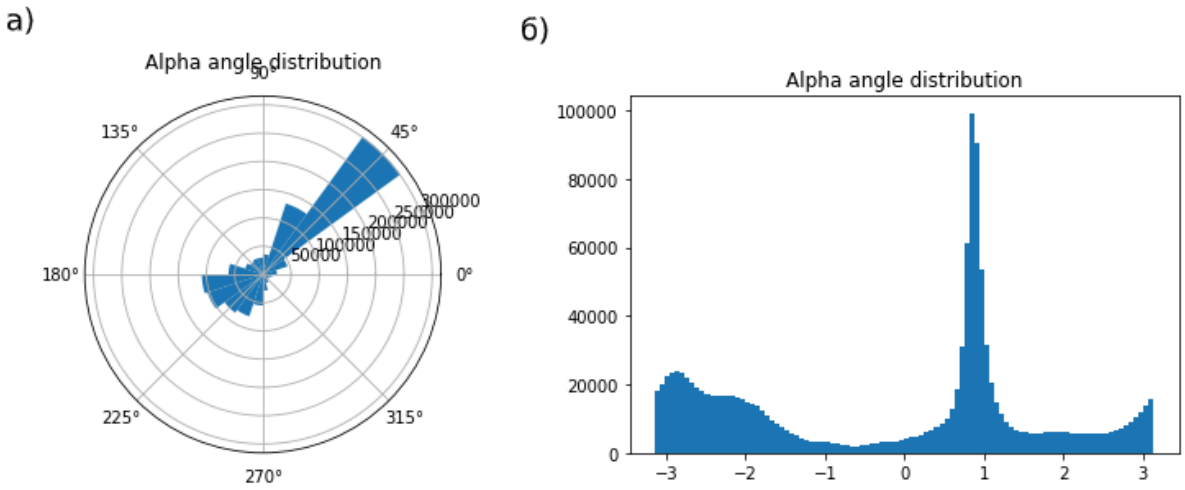

Рисунок 2. а) Распределение всех углов α обучающей выборки на полярной системе координат б) Распределение всех углов α обучающей выборки на декартовой системе координат

На Рисунках 2 (а) и (б) изображено распределение всех углов α обучающей выборки на полярной и декартовой системе координат соответственно. Как показано на гистограмме, имеется два ярко выраженных пика: в районе 50 градусов и 210 градусов. Такое распределение углов позволяет перейти к задаче классификации, где каждый класс будет характеризовать его медианное значение. При хорошей классификации этого будет достаточно, чтобы использовать полученные значения углов для начального приближения при аппроксимации пространственной структуры.

4. Кластеризация углов α

В силу того, что целевая переменная была вычислена как величина двугранного угла, необходимо определить, к каким классам будут относиться те или иные углы. Для этого кластеризуем все углы α из обучающей выборки на 2, 3 и 4 кластера, которые и будут являться классами при обучении моделей. Кластеризация будет происходить с помощью метода k средних (KMeans), который реализован в библиотеке scikit-learn. В качестве признаков для кластеризации выступают косинус и синус углов α.

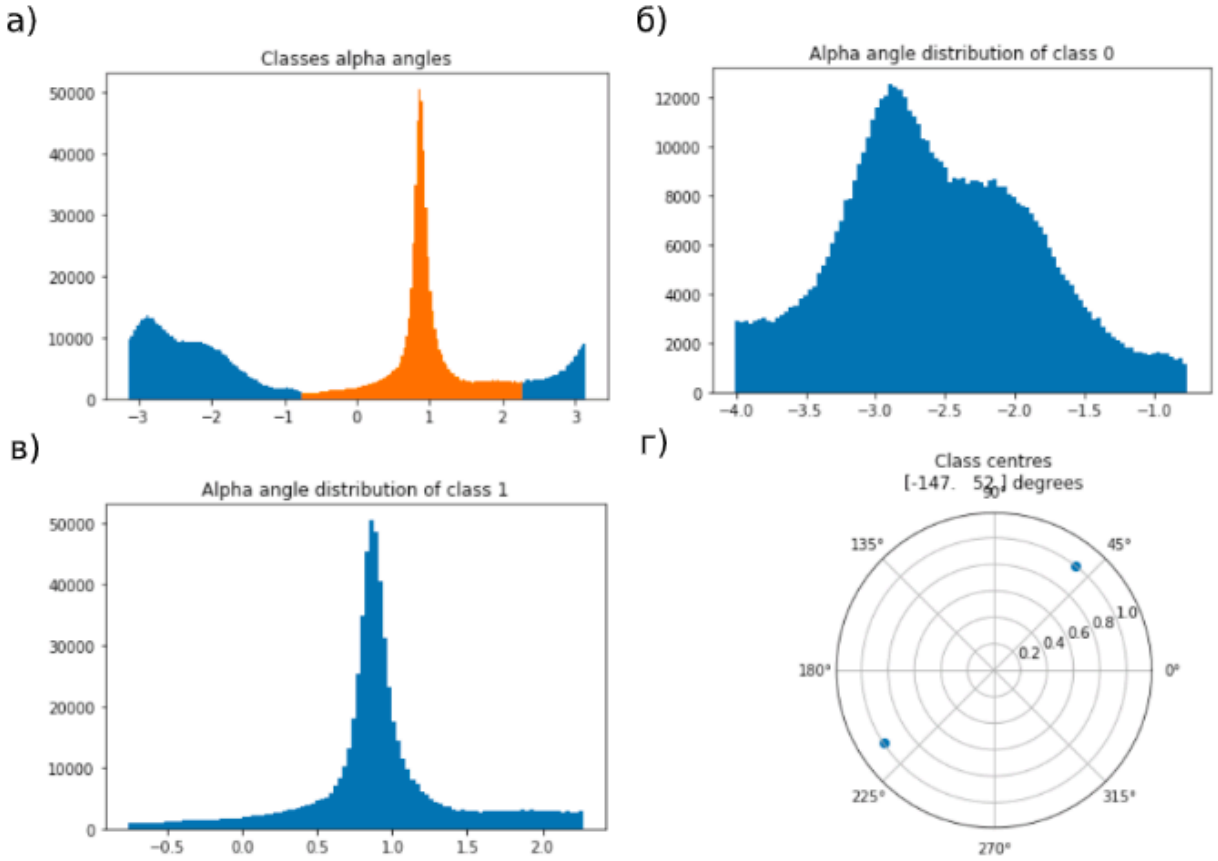

Рисунок 3. а) Распределение углов α, разделенных на два класса б) Распределение углов нулевого класса в) Распределение углов первого класса г) Центры классов на полярной системе координат

Сперва углы α были разбиты на 2 кластера. На Рисунке 3 (а) изображено распределение разделенных углов α обучающей выборки, а на Рисунках 3 (б) и (в) – распределение каждого класса по отдельности. Нулевой кластер включает в себя 595778 углов α, первый кластер – 637402. Центры кластеров имеют углы -147 и 52 градусов, их можно увидеть на Рисунке 3 (г), медианные значения каждого кластера -2.61 и 0.88 радиан на отрезке $[-\pi; \pi]$.

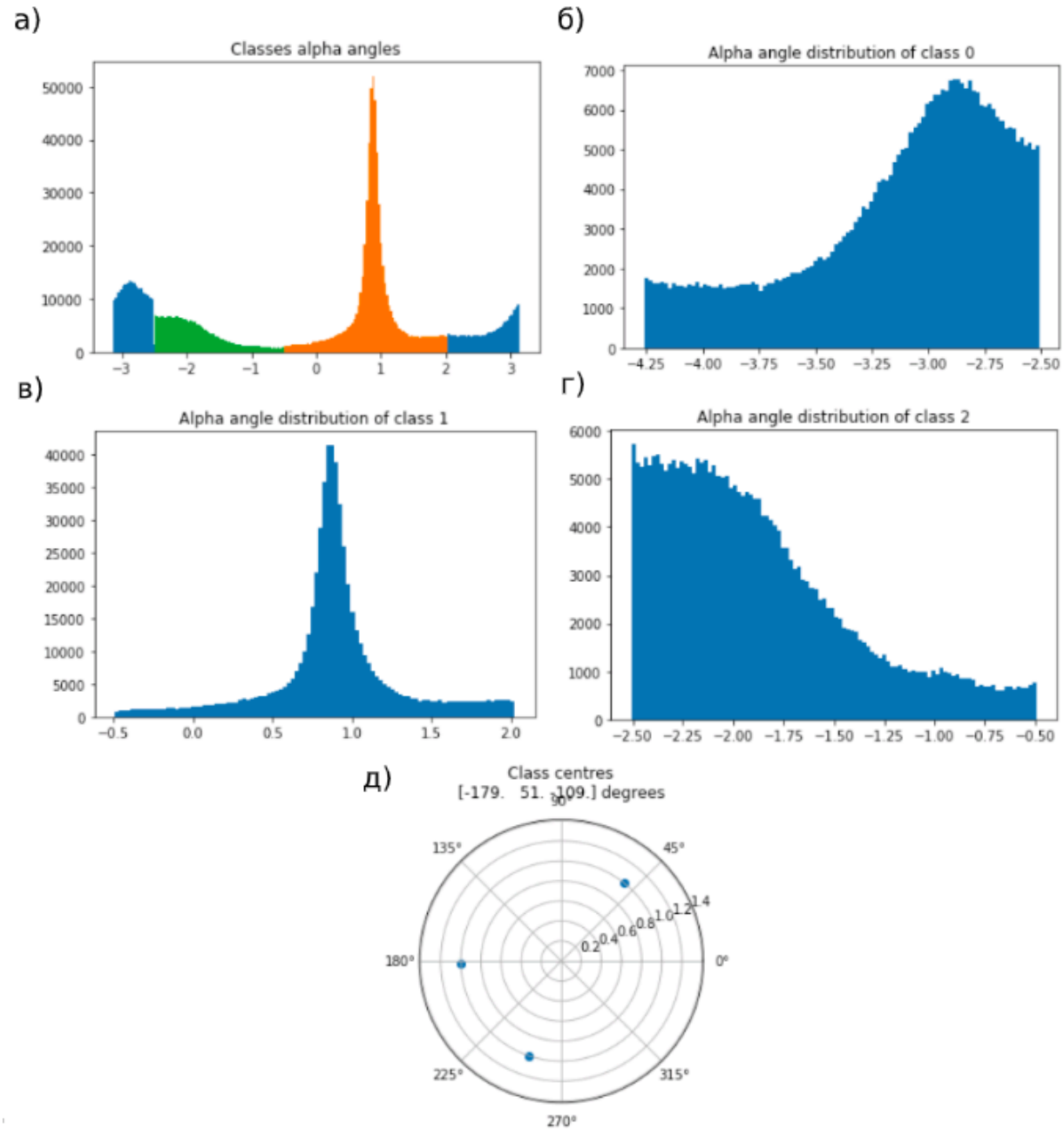

Рисунок 4. а) Распределение углов α, разделенных на три класса первым способом б) Распределение углов нулевого класса в) Распределение углов первого класса г) Распределение углов второго класса д) Центры классов на полярной системе координат

Можно заметить, что при кластеризации на 2 кластера распределение углов нулевого кластера имеет 2 выраженных пика, в следствие этого находим целесообразным кластеризовать все углы α обучающей выборки на 3 кластера (далее такой способ трехклассовой классификации будет называться первым). Такое распределение изображено на Рисунке 4 (а). На Рисунках 4 (б), (в) и (г) изображены распределения каждого кластера, их размеры составляют 351216, 605036 и 276928 углов α, а центры кластеров равняются -179, 51 и -109

градусам соответственно. На Рисунке 4 (д) показана полярная система координат, на которой отмечены центры этих кластеров. Медианные значения каждого кластера составляют -3.01, 0.88 и -1.98 радиан соответственно.

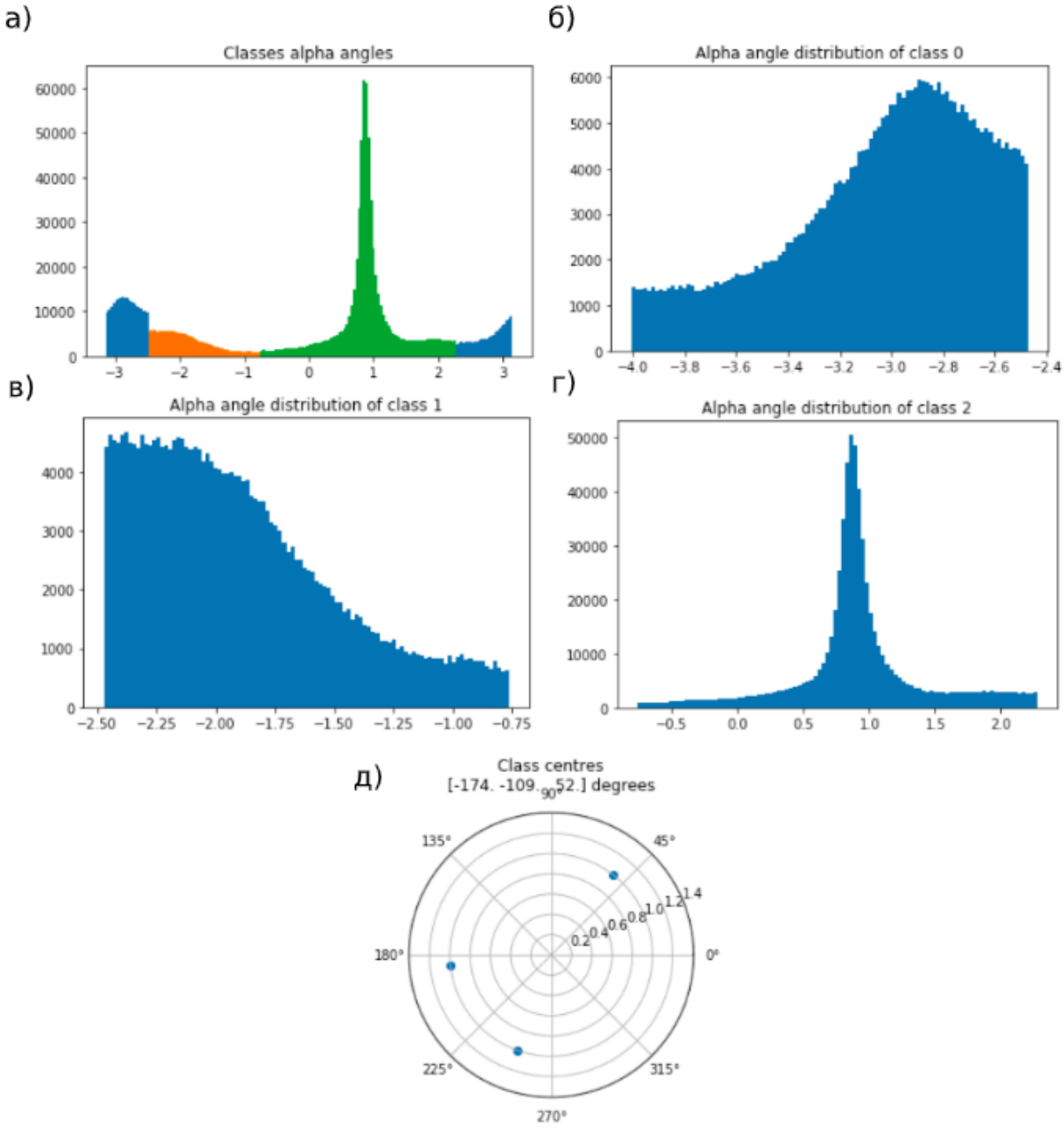

Рисунок 5. а) Распределение углов α, разделенных на три класса вторым способом б) Распределение углов нулевого класса в) Распределение углов первого класса. г) Распределение углов второго класса д) Центры классов на полярной системе координат

В связи с тем, что бинарная классификация показывает более успешные результаты, чем предсказание 3-х классов углов, считаем уместным произвести трехклассовую классификацию посредством двух последовательных бинарных классификаций. Последовательная бинарная

классификация подразумевает, что после бинарной кластеризации углов обучающей выборки нулевой кластер, тот, что имеет 2 пика, еще раз бинарно кластеризуется, и каждый кластер становится классом для последовательной бинарной классификации. Однако в условиях данной задачи для каждого объекта предсказывается последовательность двугранных углов α, в результате чего модель снова предсказывает три класса (далее будем называть это вторым способом трехклассовой классификации). Результат двух последовательных бинарных классификаций продемонстрирован на Рисунке 5 (а). По аналогии с предыдущим пунктом Рисунки 5 (б), (в) и (г) отображают распределение каждого кластера, размеры которых – 337813, 257936 и 637431 элементов соответственно. Их центры лежат на углах -174, -109, 52 градусов, что изображено на Рисунке 5 (д). Медианное значение каждого кластера составляет -2.97, -1.98 и 0.88 радиан.

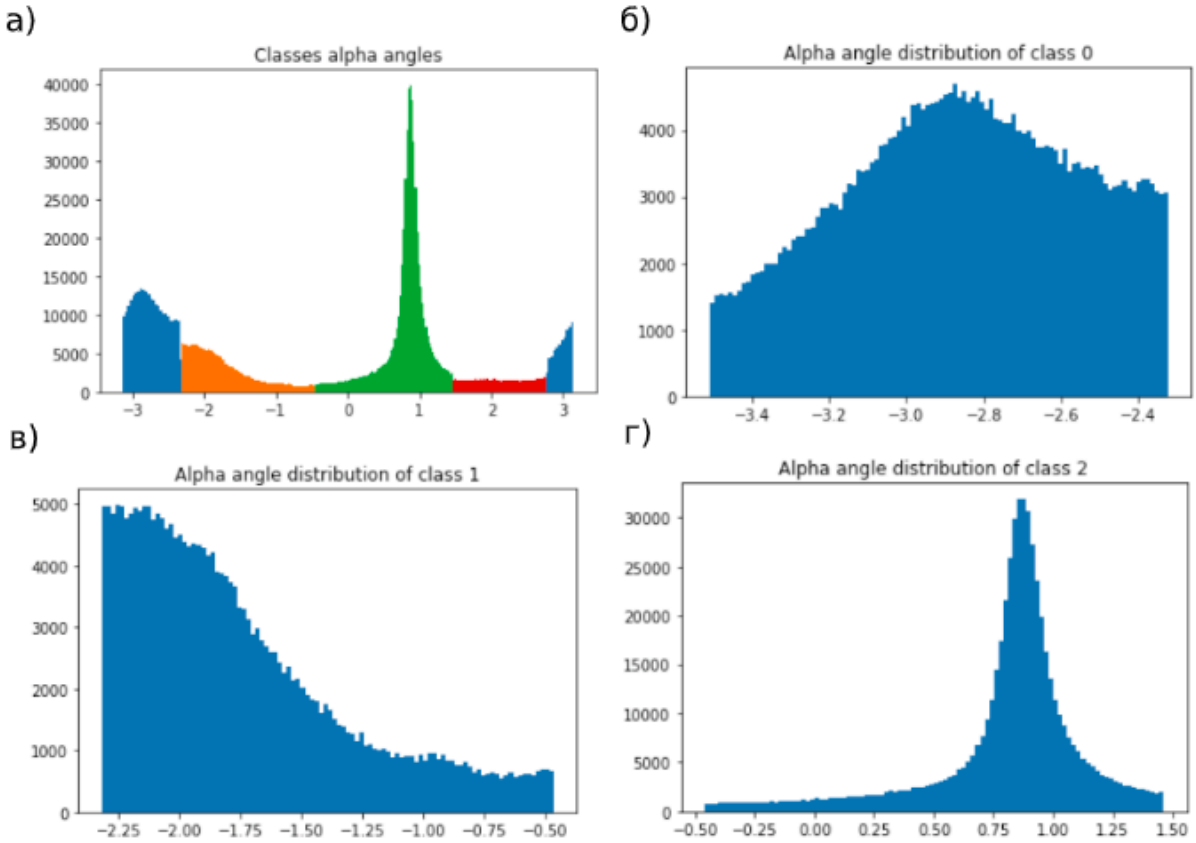

Рисунок 6. а) Распределение углов α, разделенных на четыре класса б) Распределение углов нулевого класса в) Распределение углов первого класса г) Распределение углов второго класса

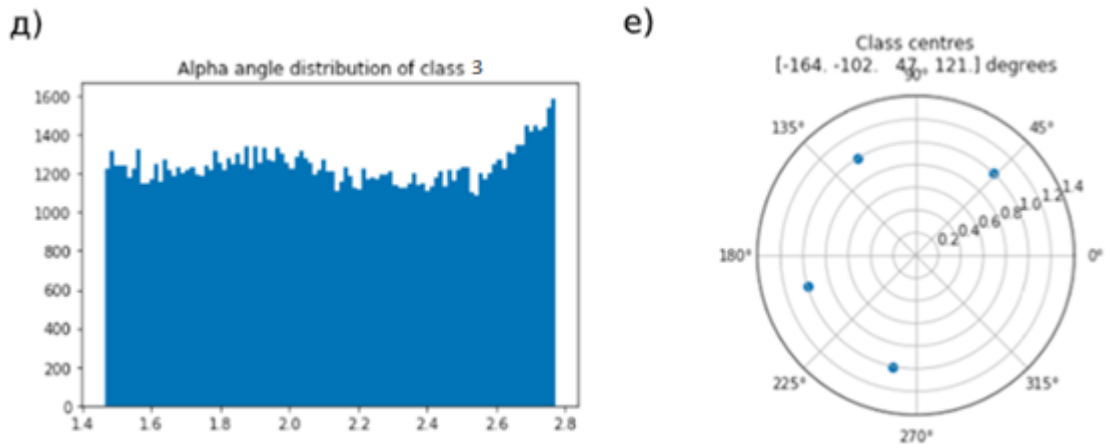

Рисунок 7 (продолжение). д) Распределение углов третьего класса е) Центры классов на полярной системе координат

Последним видом кластеризации является разбиение двугранных углов α на 4 кластера. На Рисунке 6 (а) изображено общее распределение углов, разбитое на 4 части. Рисунки 6 (б), (в), (г) и (д) репрезентируют распределение каждого кластера. Размеры кластеров – 330129, 228496, 550751 и 123804 углов α соответственно. Медианные значения каждого кластера равняются -2.86, -1.87, 0.86 и 2.11 радиан, а центры лежат на углах -164, -102, 47 и 121 градусов, которые изображены на полярной системе координат на Рисунке 6 (е).

Таким образом, для каждого вида классификации, двух-, трех- и четырехклассовой, появилась целевая переменная, которую и будут предсказывать модели.

5. Модели

При разработке моделей использовалось два подхода – сверточные и рекуррентные нейронный сети. Всего можно выделить три основных блока, описывающих архитектуры моделей: блок, состоящий из одномерных сверточных нейронных слоев (Conv1d), блок, состоящий из двумерных сверточных нейронных слоев (Conv2d), и блок, состоящий из двунаправленных рекуррентных слоев с длительной краткосрочной памятью (BiLSTM).

Всего было реализовано 10 моделей, каждая из которых состоит из одного блока или комбинации нескольких блоков, описанных ранее:

1. Conv1d

   Эта модель состоит из одномерных сверточных слоев. Они способствуют выделению признаков по каждой аминокислоте. Для повышения стабильности и увеличения скорости обучения между сверточными слоями используется нормализация, а также применяется стандартная функция активации Relu для добавления нелинейности. Для предотвращения переобучения случайным образом зануляются некоторые параметры. (См. Приложение 1 Рисунок 7.)

2. Conv1d_BiLSTM

   Данная модель состоит из блока одномерных сверточных слоев, описанных в первом пункте, и пяти двунаправленных рекуррентных слоев длительной краткосрочной памяти. После выявления признаков сверточными слоями они попадают в слои BiLSTM. Это позволяет учесть саму структуру аминокислотной последовательности в целом, а механизм памяти в слоях помогает учесть признаки начальных аминокислот в конце рассматриваемой последовательности. (См. Приложение 1 Рисунок 8.)

3. Conv1d_BiLSTM_Conv2d

   Эта модель включает в себя слои, описанные в предыдущем пункте, а также двумерные свертки, примененные к выходу рекуррентных слоев. В этом случае выход слоев BiLSTM рассматривается в качестве матрицы, из которой извлекаются новые релевантные признаки. Аналогично блоку с одномерными сверточными слоями, между двумерными свертками происходит нормализация и добавление нелинейности, а также зануление некоторых параметров. (См. Приложение 1 Рисунок 9.)

4. Conv1d_Conv2d

   Данная модель имеет слои нейронной сети, описанные в пункте 3, за исключением рекуррентных слоев. (См. Приложение 1 Рисунок 10.)

5. BiLSTM

   Модель состоит только из блока двунаправленных рекуррентных нейронных слоев, который включает 5 слоев BiLSTM с промежуточными занулением некоторых параметров с целью избегания переобучения. (См. Приложение 1 Рисунок 11.)

6. BiLSTM_Conv2d

   Архитектора модели аналогична описанной в пункте 3, но без одномерных сверточных слоев. (См. Приложение 1 Рисунок 12.)

7. Conv2d

   Эта модель включает только слои двумерной свертки, примененные к входным данным, а также уменьшение размерности в два раза после первых двух сверток посредством взятия максимального значения пробегающего по матрице признаков окна (Max Pooling). (См. Приложение 1 Рисунок 13.)

8. UConv1d

   Данная модель имеет архитектуру U-net [10]. Сначала к признакам применяются одномерные свертки с увеличением каналов и уменьшением размерности. Затем происходит так называемая развертка: количество каналов уменьшается, а размерность увеличивается, причем на вход новому слою развертки подается выход предыдущего слоя, объединенный с выходом сверточного слоя, симметричного ему относительно середины. (См. Приложение 1 Рисунок 14.)

9. UConv1d_BiLSTM

   Эта модель дополняет предыдущую двунаправленными рекуррентными слоями. (См. Приложение 1 Рисунок 15.)

10. UConv1d_BiLSTM_Conv2d

    Данная модель подобна той, что описана в пункте 9, но с двумерными сверточными нейронными слоями после блока BiLSTM слоев. (См. Приложение 1 Рисунок 16.)

Каждая модель в конце имеет полносвязный слой, выход которой регулируется в зависимости от типа классификации, а также функцию активации Softmax, которая выдает вероятностное распределение по классам.

6. Результаты экспериментов

Обучение моделей происходило в 10 эпох. Были проведены эксперименты с различными размерами батча {8, 16, 32, 64} и с разными длинами аминокислотной подпоследовательности – 8, 16, 32, 48. Результаты перебора этих значений показали, что оптимальный размер батча – 32, а лучшие показатели достигаются при подпоследовательности длины 32.

Таблица 1. Значения F-меры всех моделей для каждого вида классификации

Модель	**F-мера 2 класса**	**F-мера 3 класса 1 способ**	**F-мера 3 класса 2 способ**	**F-мера 4 класса**
Conv1d	0.833162	0.724899	0.732857	0.690707
Conv1d_BiLSTM	0.856094	0.756413	0.763038	0.718834
Conv1d_BiLSTM_Conv2d	0.850337	0.740856	0.747440	0.701173
Conv1d_Conv2d	0.798300	0.682992	0.696408	0.644376
BiLSTM	0.881808	0.792452	0.794474	0.751644
BiLSTM_Conv2d	0.868541	0.768680	0.772834	0.723750
Conv2d	0.750621	0.631998	0.652393	0.602961
UConv1d	0.842637	0.737231	0.745524	0.703507
UConv1d_BiLSTM	0.864026	0.764858	0.768653	0.726810
UConv1d_BiLSTM_Conv2d	0.856585	0.753091	0.755145	0.711814

В качестве минимизируемого функционала ошибки использовалась перекрестная энтропия (cross-entropy loss). В роли оптимизатора выступал Adam с гиперпараметрами по умолчанию, а для оценки качества классификации использовалась F-мера – гармоническое среднее между точностью (precision) и полнотой (recall).

В Таблице 1 проиллюстрированы результаты всех моделей по каждому виду классификации. Как мы видим, лучший результат показала модель, состоящая из пяти последовательных двунаправленных рекуррентных слоев с длительной краткосрочной памятью. Для бинарной классификации F-мера составила 0.88, что является очень высоким показателем. Трехклассовый классификатор проявил меньшую результативность, чем бинарный, причем при обоих способах разбиения углов α на три класса наблюдались практически одинаковые значения F-меры – 0.79. Лучшей модели для четырехклассовой классификации удалось добиться качества F-меры, равной 0.75. На втором месте в подавляющем большинстве находится модель, в которой еще присутствуют двумерные сверточные слои после рекуррентных. На третьем месте располагается модель с U-Net подобной архитектурой и слоями BiLSTM. Можно заметить, что все лучшие модели включают рекуррентные слои, и это доказывает, что для предсказания классов углов α действительно важна структура последовательности.

Подробные графики процесса обучения моделей двухклассовой, трехклассовой (первого и второго способа разбиения), а также четырехклассовой классификации можно найти в Приложении 2 в Рисунках 17, 18, 19 и 20 соответственно. Под пунктом (а) всех Рисунков показано, как изменялась функция потерь на отложенной выборке (ось ординат) с шагом обучения (ось абсцисс). Под пунктом (б) каждого Рисунка изображено, как увеличивалось качество модели – F-мера (ось ординат) относительно шага обучения (ось абсцисс).

7. Заключение

Так как при использовании текущих алгоритмов не удается быстро и с высокой точностью определить структуру белка, предсказание конформации белка представляет актуальную задачу для экспертов в области биоинформатики. Способность определять пространственную структуру белка способствует созданию новых лекарственных средств, что особенно необходимо в настоящее время, когда увеличивается количество новых заболеваний, а, следовательно, и потребность в новых медицинских разработках.

В данной работе описаны предсказания двугранных углов α посредством их классифицирования. Такой способ в данном случае корректен в силу того, что распределение углов имеет два ярко выраженных пика около 50 и 210 градусов. В дальнейшем предсказания классов можно использовать в качестве начального приближения для метода, аппроксимирующего конформацию белка, заменив номера классов на медианные значения соответствующих классов. Из 10 реализованных моделей наибольшую эффективность показала архитектура, основанная только на двунаправленных рекуррентных нейронных сетях с длительной краткосрочной памятью (BiLSTM). Показатель F-меры для этой модели при бинарной классификации составил 0.88, для трехклассовой – 0.79, для четырехклассовой – 0.75.

Таким образом, потенциал моделирования фолдинга белка очень велик, именно поэтому в настоящее время большое количество исследователей сконцентрировано на этой задаче. Дальнейшие перспективы этой работы включают наращивание обучающего набора данных новыми признаками, например, двоичной структурой белка и физическими характеристиками аминокислот. Это может способствовать выявлению новых зависимостей и повышению качества классификации.

8. Список источников

1. Hansmann, U. and Okamoto, Y., 1993. Prediction of peptide conformation by multicanonical algorithm: New approach to the multiple-minima problem. *Journal of Computational Chemistry*, 14(11), pp.1333-1338.
2. Šali, A. and Blundell, T., 1993. Comparative Protein Modelling by Satisfaction of Spatial Restraints. *Journal of Molecular Biology*, 234(3), pp.779-815.
3. Jones, D., Taylort, W. and Thornton, J., 1992. A new approach to protein fold recognition. *Nature*, 358(6381), pp.86-89.
4. Dill, K., Ozkan, S., Shell, M. and Weikl, T., 2008. The Protein Folding Problem. *Annual Review of Biophysics*, 37(1), pp.289-316.
5. Levitt, M. and Warshel, A., 1975. Computer simulation of protein folding. *Nature*, 253(5494), pp.694-698.
6. Kifer, I., Nussinov, R. and Wolfson, H., 2008. Constructing templates for protein structure prediction by simulation of protein folding pathways. *Proteins: Structure, Function, and Bioinformatics*, 73(2), pp.380-394.
7. Rose, G., 1979. Hierarchic organization of domains in globular proteins. *Journal of Molecular Biology*, 134(3), pp.447-470.
8. Senior, A., Evans, R., Jumper, J., Kirkpatrick, J., Sifre, L., Green, T., Qin, C., Žídek, A., Nelson, A., Bridgland, A., Penedones, H., Petersen, S., Simonyan, K., Crossan, S., Kohli, P., Jones, D., Silver, D., Kavukcuoglu, K. and Hassabis, D., 2019. Protein structure prediction using multiple deep neural networks in the 13th Critical Assessment of Protein Structure Prediction (CASP13). *Proteins: Structure, Function, and Bioinformatics*, 87(12), pp.1141-1148.
9. Li, W., Schaeffer, R., Otwinowski, Z. and Grishin, N., 2016. Estimation of Uncertainties in the Global Distance Test (GDT_TS) for CASP Models. *PLOS ONE*, 11(5), p.e0154786.

10. Ronneberger, O., Fischer, P. and Brox, T., 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. *Lecture Notes in Computer Science*, pp.234-241.

Приложение 1. Архитектуры моделей

```
Conv1d(
  (conv1d): Sequential(
    (0): Conv1d(41, 16, kernel_size=(3,), stride=(1,), padding=(1,))
    (1): ReLU()
    (2): BatchNorm1d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): Dropout(p=0.2, inplace=False)
    (4): Conv1d(16, 32, kernel_size=(3,), stride=(1,), padding=(1,))
    (5): ReLU()
    (6): BatchNorm1d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (7): Dropout(p=0.2, inplace=False)
    (8): Conv1d(32, 64, kernel_size=(3,), stride=(1,), padding=(1,))
    (9): ReLU()
    (10): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (11): Dropout(p=0.2, inplace=False)
    (12): Conv1d(64, 128, kernel_size=(3,), stride=(1,), padding=(1,))
    (13): ReLU()
    (14): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (15): Dropout(p=0.2, inplace=False)
    (16): Flatten()
    (17): Linear(in_features=4096, out_features=64, bias=True)
    (18): Linear(in_features=64, out_features=62, bias=True)
  )
)
```

Рисунок 8. Архитектура модели Conv1d

```
Conv1d_BiLSTM(
  (conv1d): Sequential(
    (0): Conv1d(41, 16, kernel_size=(3,), stride=(1,), padding=(1,))
    (1): ReLU()
    (2): BatchNorm1d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): Dropout(p=0.2, inplace=False)
    (4): Conv1d(16, 32, kernel_size=(3,), stride=(1,), padding=(1,))
    (5): ReLU()
    (6): BatchNorm1d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (7): Dropout(p=0.2, inplace=False)
    (8): Conv1d(32, 64, kernel_size=(3,), stride=(1,), padding=(1,))
    (9): ReLU()
    (10): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (11): Dropout(p=0.2, inplace=False)
  )
  (bilstm1): LSTM(64, 256, batch_first=True, dropout=0.4, bidirectional=True)
  (bilstm2): LSTM(512, 128, batch_first=True, dropout=0.4, bidirectional=True)
  (bilstm3): LSTM(256, 64, batch_first=True, dropout=0.4, bidirectional=True)
  (bilstm4): LSTM(128, 32, batch_first=True, dropout=0.2, bidirectional=True)
  (bilstm5): LSTM(64, 16, batch_first=True, dropout=0.2, bidirectional=True)
  (flatten): Flatten()
  (fc): Linear(in_features=1024, out_features=62, bias=True)
)
```

Рисунок 9. Архитектура модели Conv1d_BiLSTM

```
Conv1d_BiLSTM_Conv2d(
  (conv1d): Sequential(
    (0): Conv1d(41, 16, kernel_size=(3,), stride=(1,), padding=(1,))
    (1): ReLU()
    (2): BatchNorm1d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): Dropout(p=0.2, inplace=False)
    (4): Conv1d(16, 32, kernel_size=(3,), stride=(1,), padding=(1,))
    (5): ReLU()
    (6): BatchNorm1d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (7): Dropout(p=0.2, inplace=False)
    (8): Conv1d(32, 64, kernel_size=(3,), stride=(1,), padding=(1,))
    (9): ReLU()
    (10): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (11): Dropout(p=0.2, inplace=False)
  )
  (bilstm1): LSTM(64, 256, batch_first=True, dropout=0.4, bidirectional=True)
  (bilstm2): LSTM(512, 128, batch_first=True, dropout=0.4, bidirectional=True)
  (bilstm3): LSTM(256, 64, batch_first=True, dropout=0.4, bidirectional=True)
  (bilstm4): LSTM(128, 32, batch_first=True, dropout=0.2, bidirectional=True)
  (bilstm5): LSTM(64, 16, batch_first=True, dropout=0.2, bidirectional=True)
  (conv2d): Sequential(
    (0): Conv2d(1, 8, kernel_size=(3, 3), stride=(1, 1))
    (1): Conv2d(8, 8, kernel_size=(3, 3), stride=(1, 1))
    (2): ReLU()
    (3): BatchNorm2d(8, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (4): Dropout(p=0.2, inplace=False)
    (5): Conv2d(8, 16, kernel_size=(3, 3), stride=(1, 1))
    (6): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1))
    (7): ReLU()
    (8): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (9): Dropout(p=0.2, inplace=False)
    (10): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1))
    (11): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1))
    (12): ReLU()
    (13): Dropout(p=0.2, inplace=False)
  )
  (flatten): Flatten()
  (fc1): Linear(in_features=12800, out_features=64, bias=True)
  (fc2): Linear(in_features=64, out_features=62, bias=True)
)
```

Рисунок 10. Архитектура модели Conv1d_BiLSTM_Conv2d

```
Conv1d_Conv2d(
  (conv1d): Sequential(
    (0): Conv1d(41, 16, kernel_size=(3,), stride=(1,), padding=(1,))
    (1): ReLU()
    (2): BatchNorm1d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): Dropout(p=0.2, inplace=False)
    (4): Conv1d(16, 32, kernel_size=(3,), stride=(1,), padding=(1,))
    (5): ReLU()
    (6): BatchNorm1d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (7): Dropout(p=0.2, inplace=False)
    (8): Conv1d(32, 64, kernel_size=(3,), stride=(1,), padding=(1,))
    (9): ReLU()
    (10): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (11): Dropout(p=0.2, inplace=False)
    (12): Conv1d(64, 128, kernel_size=(3,), stride=(1,), padding=(1,))
    (13): ReLU()
    (14): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (15): Dropout(p=0.2, inplace=False)
    (16): Conv1d(128, 128, kernel_size=(3,), stride=(1,), padding=(1,))
  )
  (conv2d): Sequential(
    (0): Conv2d(1, 8, kernel_size=(3, 3), stride=(1, 1))
    (1): ReLU()
    (2): BatchNorm2d(8, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): Dropout(p=0.4, inplace=False)
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (5): Conv2d(8, 16, kernel_size=(3, 3), stride=(1, 1))
    (6): ReLU()
    (7): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (8): Dropout(p=0.4, inplace=False)
    (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (10): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1))
    (11): ReLU()
    (12): Dropout(p=0.4, inplace=False)
    (13): Flatten()
    (14): Linear(in_features=3584, out_features=64, bias=True)
    (15): Linear(in_features=64, out_features=62, bias=True)
  )
)
```

Рисунок 11. Архитектура модели Conv1d_Conv2d

```
BiLSTM(
  (bilstm1): LSTM(41, 256, batch_first=True, dropout=0.4, bidirectional=True)
  (bilstm2): LSTM(512, 128, batch_first=True, dropout=0.4, bidirectional=True)
  (bilstm3): LSTM(256, 64, batch_first=True, dropout=0.4, bidirectional=True)
  (bilstm4): LSTM(128, 32, batch_first=True, dropout=0.2, bidirectional=True)
  (bilstm5): LSTM(64, 16, batch_first=True, dropout=0.2, bidirectional=True)
  (flatten): Flatten()
  (fc): Linear(in_features=1024, out_features=62, bias=True)
)
```

Рисунок 12. Архитектура модели BiLSTM

```
BiLSTM_Conv2d(
  (bilstm1): LSTM(41, 256, batch_first=True, dropout=0.4, bidirectional=True)
  (bilstm2): LSTM(512, 128, batch_first=True, dropout=0.4, bidirectional=True)
  (bilstm3): LSTM(256, 64, batch_first=True, dropout=0.4, bidirectional=True)
  (bilstm4): LSTM(128, 32, batch_first=True, dropout=0.2, bidirectional=True)
  (bilstm5): LSTM(64, 16, batch_first=True, dropout=0.2, bidirectional=True)
  (conv2d): Sequential(
    (0): Conv2d(1, 8, kernel_size=(3, 3), stride=(1, 1))
    (1): Conv2d(8, 8, kernel_size=(3, 3), stride=(1, 1))
    (2): ReLU()
    (3): BatchNorm2d(8, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (4): Dropout(p=0.2, inplace=False)
    (5): Conv2d(8, 16, kernel_size=(3, 3), stride=(1, 1))
    (6): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1))
    (7): ReLU()
    (8): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (9): Dropout(p=0.2, inplace=False)
    (10): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1))
    (11): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1))
    (12): ReLU()
    (13): Dropout(p=0.2, inplace=False)
  )
  (flatten): Flatten()
  (fc1): Linear(in_features=12800, out_features=64, bias=True)
  (fc2): Linear(in_features=64, out_features=62, bias=True)
)
```

Рисунок 13. Архитектура модели BiLSTM_Conv2d

```
Conv2d(
  (conv2d): Sequential(
    (0): Conv2d(1, 16, kernel_size=(3, 3), stride=(1, 1))
    (1): ReLU()
    (2): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): Dropout(p=0.1, inplace=False)
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (5): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1))
    (6): ReLU()
    (7): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (8): Dropout(p=0.1, inplace=False)
    (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (10): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1))
    (11): ReLU()
    (12): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (13): Dropout(p=0.1, inplace=False)
    (14): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1))
    (15): ReLU()
    (16): Dropout(p=0.1, inplace=False)
    (17): Flatten()
    (18): Linear(in_features=1024, out_features=64, bias=True)
    (19): Linear(in_features=64, out_features=62, bias=True)
  )
)
```

Рисунок 14. Архитектура модели Conv2d

```
UConv1d(
  (pool): MaxPool1d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv1): Sequential(
    (0): Conv1d(41, 32, kernel_size=(3,), stride=(1,), padding=(1,))
    (1): ReLU()
    (2): BatchNorm1d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): Dropout(p=0.2, inplace=False)
  )
  (conv2): Sequential(
    (0): Conv1d(32, 64, kernel_size=(3,), stride=(1,), padding=(1,))
    (1): ReLU()
    (2): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): Dropout(p=0.2, inplace=False)
  )
  (conv3): Sequential(
    (0): Conv1d(64, 128, kernel_size=(3,), stride=(1,), padding=(1,))
    (1): ReLU()
    (2): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): Dropout(p=0.2, inplace=False)
  )
  (conv4): Sequential(
    (0): Conv1d(128, 256, kernel_size=(3,), stride=(1,), padding=(1,))
    (1): ReLU()
    (2): BatchNorm1d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): Dropout(p=0.2, inplace=False)
  )
  (deconv1): Sequential(
    (0): Upsample(size=16, mode=nearest)
    (1): Conv1d(256, 128, kernel_size=(3,), stride=(1,), padding=(1,))
    (2): ReLU()
    (3): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (4): Dropout(p=0.2, inplace=False)
  )
  (deconv2): Sequential(
    (0): Upsample(size=32, mode=nearest)
    (1): Conv1d(128, 64, kernel_size=(3,), stride=(1,), padding=(1,))
    (2): ReLU()
    (3): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (4): Dropout(p=0.2, inplace=False)
  )
  (deconv3): Sequential(
    (0): Conv1d(64, 32, kernel_size=(3,), stride=(1,), padding=(1,))
    (1): ReLU()
    (2): BatchNorm1d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): Dropout(p=0.2, inplace=False)
  )
  (flatten): Flatten()
  (fc): Linear(in_features=2048, out_features=62, bias=True)
)
```

Рисунок 15. Архитектура модели UConv1d

```
UConv1d_BiLSTM(
  (pool): MaxPool1d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv1): Sequential(
    (0): Conv1d(41, 32, kernel_size=(3,), stride=(1,), padding=(1,))
    (1): ReLU()
    (2): BatchNorm1d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): Dropout(p=0.2, inplace=False)
  )
  (conv2): Sequential(
    (0): Conv1d(32, 64, kernel_size=(3,), stride=(1,), padding=(1,))
    (1): ReLU()
    (2): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): Dropout(p=0.2, inplace=False)
  )
  (conv3): Sequential(
    (0): Conv1d(64, 128, kernel_size=(3,), stride=(1,), padding=(1,))
    (1): ReLU()
    (2): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): Dropout(p=0.2, inplace=False)
  )
  (conv4): Sequential(
    (0): Conv1d(128, 256, kernel_size=(3,), stride=(1,), padding=(1,))
    (1): ReLU()
    (2): BatchNorm1d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): Dropout(p=0.2, inplace=False)
  )
  (deconv1): Sequential(
    (0): Upsample(size=16, mode=nearest)
    (1): Conv1d(256, 128, kernel_size=(3,), stride=(1,), padding=(1,))
    (2): ReLU()
    (3): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (4): Dropout(p=0.2, inplace=False)
  )
  (deconv2): Sequential(
    (0): Upsample(size=32, mode=nearest)
    (1): Conv1d(128, 64, kernel_size=(3,), stride=(1,), padding=(1,))
    (2): ReLU()
    (3): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (4): Dropout(p=0.2, inplace=False)
  )
  (deconv3): Sequential(
    (0): Conv1d(64, 32, kernel_size=(3,), stride=(1,), padding=(1,))
    (1): ReLU()
    (2): BatchNorm1d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): Dropout(p=0.2, inplace=False)
  )
  (bilstm1): LSTM(32, 256, batch_first=True, dropout=0.4, bidirectional=True)
  (bilstm2): LSTM(512, 128, batch_first=True, dropout=0.4, bidirectional=True)
  (bilstm3): LSTM(256, 64, batch_first=True, dropout=0.4, bidirectional=True)
  (bilstm4): LSTM(128, 32, batch_first=True, dropout=0.2, bidirectional=True)
  (bilstm5): LSTM(64, 16, batch_first=True, dropout=0.2, bidirectional=True)
  (flatten): Flatten()
  (fc): Linear(in_features=2048, out_features=62, bias=True)
)
```

Рисунок 16. Архитектура модели UConv1d_BiLSTM

```
UConv1d_BiLSTM_Conv2d(
  (pool): MaxPool1d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv1): Sequential(
    (0): Conv1d(41, 32, kernel_size=(3,), stride=(1,), padding=(1,))
    (1): ReLU()
    (2): BatchNorm1d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): Dropout(p=0.2, inplace=False)
  )
  (conv2): Sequential(
    (0): Conv1d(32, 64, kernel_size=(3,), stride=(1,), padding=(1,))
    (1): ReLU()
    (2): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): Dropout(p=0.2, inplace=False)
  )
  (conv3): Sequential(
    (0): Conv1d(64, 128, kernel_size=(3,), stride=(1,), padding=(1,))
    (1): ReLU()
    (2): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): Dropout(p=0.2, inplace=False)
  )
  (conv4): Sequential(
    (0): Conv1d(128, 256, kernel_size=(3,), stride=(1,), padding=(1,))
    (1): ReLU()
    (2): BatchNorm1d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): Dropout(p=0.2, inplace=False)
  )
  (deconv1): Sequential(
    (0): Upsample(size=16, mode=nearest)
    (1): Conv1d(256, 128, kernel_size=(3,), stride=(1,), padding=(1,))
    (2): ReLU()
    (3): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (4): Dropout(p=0.2, inplace=False)
  )
  (deconv2): Sequential(
    (0): Upsample(size=32, mode=nearest)
    (1): Conv1d(128, 64, kernel_size=(3,), stride=(1,), padding=(1,))
    (2): ReLU()
    (3): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (4): Dropout(p=0.2, inplace=False)
  )
  (deconv3): Sequential(
    (0): Conv1d(64, 32, kernel_size=(3,), stride=(1,), padding=(1,))
    (1): ReLU()
    (2): BatchNorm1d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): Dropout(p=0.2, inplace=False)
  )
  (bilstm1): LSTM(32, 256, batch_first=True, dropout=0.4, bidirectional=True)
  (bilstm2): LSTM(512, 128, batch_first=True, dropout=0.4, bidirectional=True)
  (bilstm3): LSTM(256, 64, batch_first=True, dropout=0.4, bidirectional=True)
  (bilstm4): LSTM(128, 32, batch_first=True, dropout=0.2, bidirectional=True)
  (bilstm5): LSTM(64, 16, batch_first=True, dropout=0.2, bidirectional=True)
  (conv2d): Sequential(
    (0): Conv2d(1, 8, kernel_size=(3, 3), stride=(1, 1))
    (1): Conv2d(8, 8, kernel_size=(3, 3), stride=(1, 1))
    (2): ReLU()
    (3): BatchNorm2d(8, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (4): Dropout(p=0.2, inplace=False)
    (5): Conv2d(8, 16, kernel_size=(3, 3), stride=(1, 1))
    (6): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1))
    (7): ReLU()
    (8): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (9): Dropout(p=0.2, inplace=False)
    (10): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1))
    (11): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1))
    (12): ReLU()
    (13): Dropout(p=0.2, inplace=False)
  )
  (flatten): Flatten()
  (fc1): Linear(in_features=33280, out_features=64, bias=True)
  (fc2): Linear(in_features=64, out_features=62, bias=True)
)
```

Рисунок 17. Архитектура модели UConv1d_BiLSTM_Conv2d

Приложение 2. Процесс обучения моделей

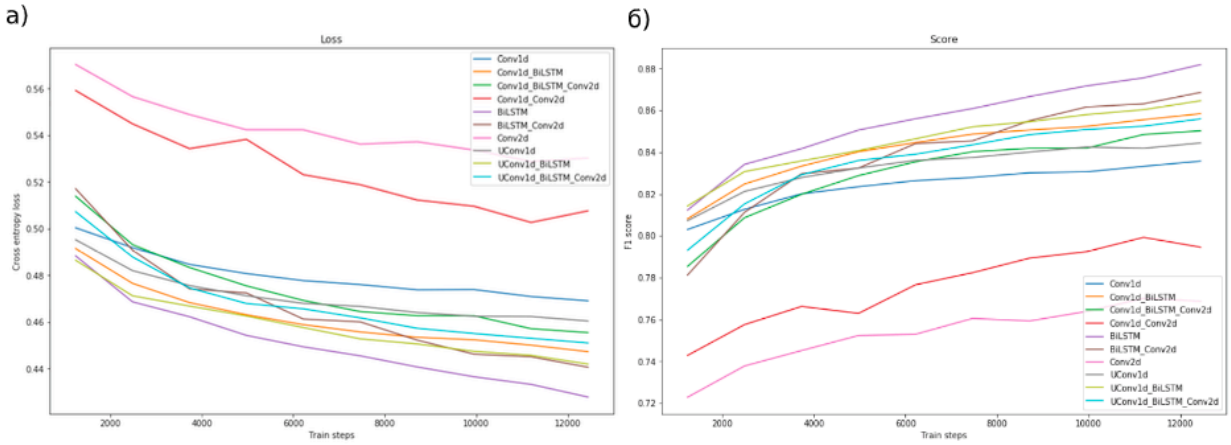

Рисунок 18. а) Зависимость значения функционала ошибки (cross-entropy loss) от шага обучения (10 эпох) всех моделей для двухклассовой классификации б) Зависимость значения метрики качества (F-мера) от шага обучения (10 эпох) всех моделей для двухклассовой классификации

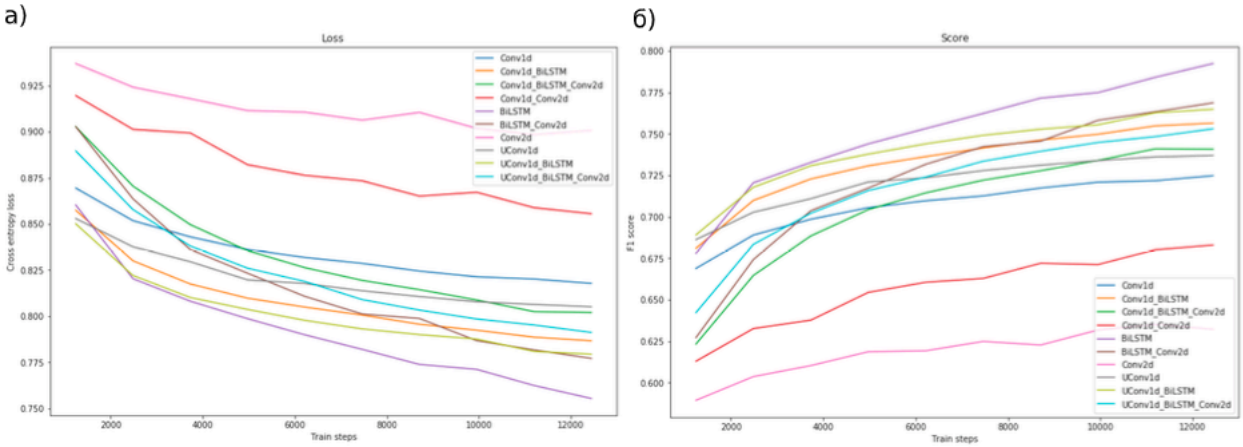

Рисунок 19. а) Зависимость значения функционала ошибки (cross-entropy loss) от шага обучения (10 эпох) всех моделей для трехклассовой классификации (разделение на классы первым способом) б) Зависимость значения метрики качества (F-мера) от шага обучения (10 эпох) всех моделей для трехклассовой классификации (разделение на классы первым способом)

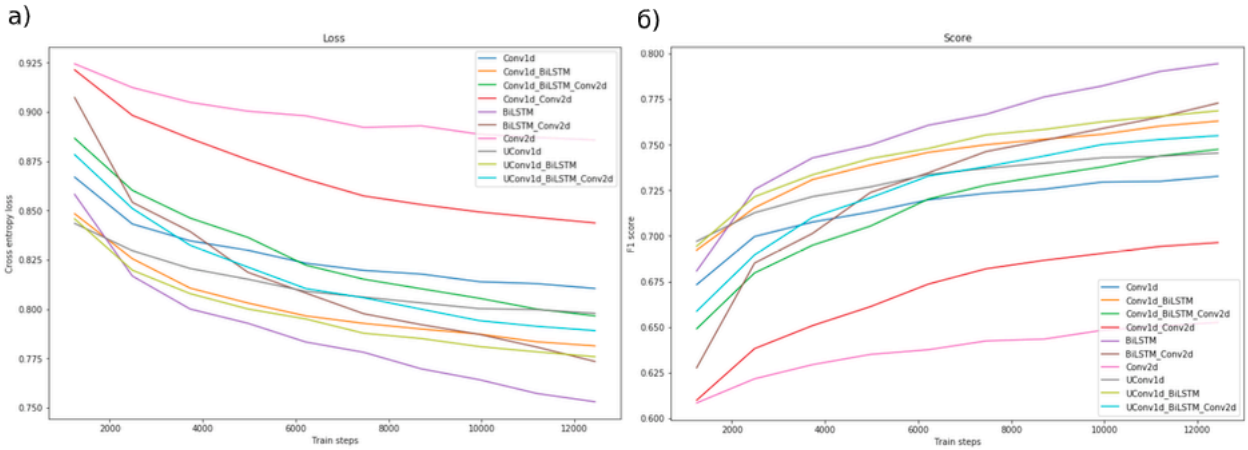

Рисунок 20. а) Зависимость значения функционала ошибки (cross-entropy loss) от шага обучения (10 эпох) всех моделей для трехклассовой классификации (разделение на классы вторым способом) б) Зависимость значения метрики качества (F-мера) от шага обучения (10 эпох) всех моделей для трехклассовой классификации (разделение на классы вторым способом)

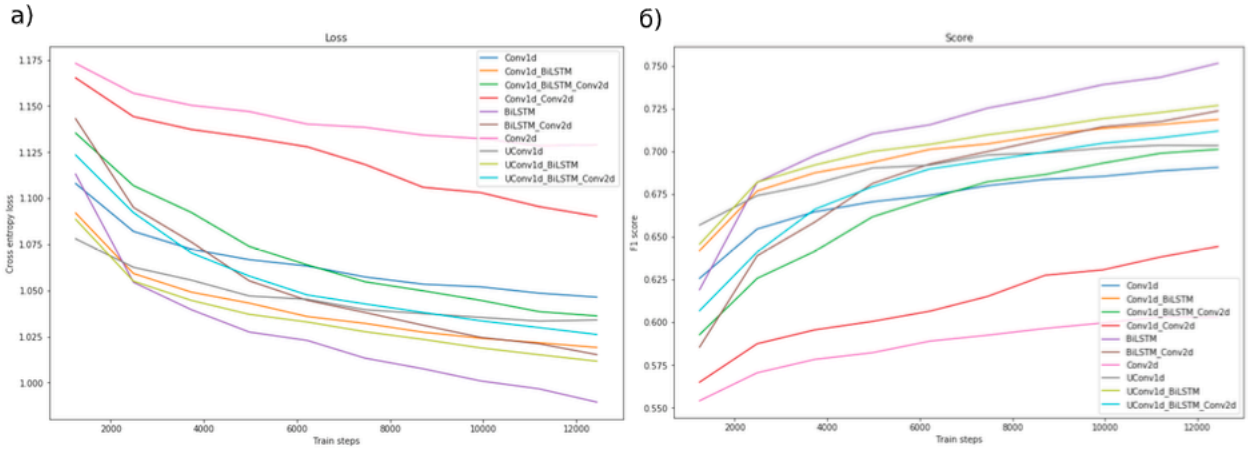

Рисунок 21. а) Зависимость значения функционала ошибки (cross-entropy loss) от шага обучения (10 эпох) всех моделей для четырехклассовой классификации б) Зависимость значения метрики качества (F-мера) от шага обучения (10 эпох) всех моделей для четырехклассовой классификации