Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное   
учреждение высшего образования

Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий, математики и механики

Кафедра алгебры, геометрии и дискретной математики

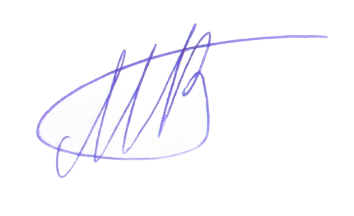
Направление: Прикладная математика и информатика

**Отчет по технологической практике**

Тема:

“Методы машинного обучения для обработки медицинских сигналов”

**Выполнил**:

студент группы 381903\_3

Манухов В.В.

Подпись

**Научный руководитель**:

директор ин-та ИТММ, зав. кафедрой АГДМ ИИТММ, д.ф.-м.н.

Золотых Н.Ю.

Подпись

Нижний Новгород

2022

Содержание

[Введениe 3](#_Toc106996814)

[Постановка задачи 4](#_Toc106996815)

[Набор данных 5](#_Toc106996816)

[Практическая часть 6](#_Toc106996817)

[Результаты 10](#_Toc106996818)

[Заключение 12](#_Toc106996819)

[Литература 13](#_Toc106996820)

[Электронные источники 14](#_Toc106996821)

[Приложения 15](#_Toc106996822)

# Введениe

Сердечно-сосудистые заболевания являются ведущей причиной высокой смертности во всем мире. Электрокардиография (ЭКГ) – это главный инструмент неинвазивной диагностики сердечно-сосудистых заболеваний. Алгоритмы автоматической интерпретации ЭКГ в качестве систем поддержки и диагностики состояний пациентов оказывают значительную помощь врачам. Разработка таких алгоритмов требует больших наборов обучающих данных и четких процедур тестирования. Для решения поставленных задач рассматриваются клинические наборы данных ЭКГ – PTB-XL и ICBEB2018.

Благодаря наличию набора данных PTB-XL возможно решение разного рода задач с прогнозированием по сигналу ЭКГ, а наличие дополнительных данных в виде ICBEB2018 позволяет расширить работу в этой области и использовать имеющиеся ресурсы для проведения сравнительного анализа методов машинного обучения.

Именно с точки зрения измерений качества, тестирования и разработки алгоритмов глубокого обучения рассматриваются наборы PTB-XL и ICBEB в статье “Deep Learning for ECG Analysis: Benchmarks and Insights from PTB-XL”[[1]](#footnote-1), авторы которой уделяют особое внимание набору данных PTB-XL и предлагают его в качестве показательного при анализе методов машинного обучения в области медицины, определяя задачи, критерии оценивания и возможности по его применению.

# Постановка задачи

Задачи производственной практики “Научно-исследовательская работа”:

* Изучение области исследуемого объекта (Сигналы ЭКГ, наборы данных PTB-XL, ICBEB2018);
* Первичный анализ и визуализация набора PTB-XL:
  + Вывод таблиц числовых данных;
  + Вывод таблиц категориальных признаков;
  + Построение сигналов ЭКГ;
  + Поиск на графиках взаимосвязей между признаками сигналов ЭКГ пациентов;
* Подготовка данных к работе с алгоритмами машинного обучения;
  + Детектирование выбросов и устранение;
  + Приведение данных к требуемым размерностям и типу данных;
* Создание нескольких моделей машинного обучения на основе сверточных нейронных сетей, которые будет соответствовать свойствам исследуемых объектов;
* Обучение на тренировочной и валидационной выборке, применение на тестовой выборке, анализ ошибки и точности при работе моделей.
* Сравнение работы созданных моделей с работой нескольких известных архитектур сверточных нейронных сетей;
* Анализ статьи “Deep Learning for ECG Analysis: Benchmarks and Insights from PTB-XL”[[2]](#footnote-2);
  + Выявление основных задач;
  + Подходы к решению рассматриваемых задач;
  + Проведение исследований согласно статье

# Набор данных

Данные ЭКГ PTB-XL были аннотированы двумя кардиологами в виде набора данных с несколькими метками, где диагностические метки были дополнительно объединены в надклассы и подклассы. Набор данных охватывает широкий спектр диагностических классов, включая, в частности, большую часть о здоровых людях. Сочетание с метаданными по демографии и дополнительными диагностическими показателями, вероятностями диагноза, вручную аннотированными, свойствами сигнала, а также предлагаемым разделением на тренировочный и тестовый наборы данных дает богатый ресурс для разработки и оценки автоматических алгоритмов интерпретации ЭКГ.

Набор данных PTB-XL – это большой набор данных из 21837 клинических ЭКГ в 12 отведениях от 18885 пациентов длительностью 10 секунд. Этот набор дополняется обширными метаданными по демографическим характеристикам, характеристикам инфаркта, вероятности диагностических заявлений ЭКГ, а также аннотированными свойствами сигналов. PTB-XL - это набор сигналов, собранных в течение почти 7 лет с октября 1989 года по июнь 1996 года. Только в 2019 году был открыт публичный доступ к данным, которые были специально оптимизированы под работу с ними с помощью инструментов машинного обучения.

Показатели ЭКГ соответствуют стандарту SCP-ECG и отнесены к трем категория diag (диагноз, например, “передний инфаркт миокарда”), form (заметные изменения определенных сегментов ЭКГ, например, “аномальный комплекс QRS”) и rhythm (изменения ритма, “фибрилляция предсердий”).

Набор данных ICBEB2018 включает 6877 сигналов ЭКГ по 12 отведениям длительностью от 6 до 60 секунд. Каждая запись ЭКГ аннотируется до 3-х высказываний до 3-х рецензентов, взятых из набора 9 классов (1 нормальный и 8 аномальных).

# Практическая часть

#### Инструменты программирования

Исследовательская часть проводилась с использованием языка программирования Python и программного интерфейса приложения (API) jupyter notebook.

Для ускоренного анализа работы методов машинного обучения мы использовали облачные сервисы Google Colaboratory, Kaggle, которые предоставляют доступ к графическому процессору (GPU), значительно ускоряющий процесс обучения глубоких нейронных сетей.

Мы применили следующие пакеты Python:

1. Pandas – это инструмент анализа и обработки данных с открытым исходным кодом, построенный на основе языка программирования Python.
2. NumPy – это проект с открытым исходным кодом, направленный на обеспечение численных вычислений с помощью Python.
3. wfdb – пакет на языке Python для чтения, записи, обработки и построения графиков физиологических сигналов и данных аннотаций. Основные функции ввода-вывода основаны на спецификациях базы данных сигналов (WFDB).
4. ast – помогает приложениям Python обрабатывать деревья грамматики абстрактного синтаксиса Python.
5. scikit-learn (sklearn) – это бесплатная библиотека машинного обучения для языка программирования Python.
6. Tensorflow – это бесплатная библиотека программного обеспечения с открытым исходным кодом для машинного обучения и искусственного интеллекта, где особое внимание уделяется обучению и выводу глубоких нейронных сетей.
7. Keras – это API глубокого обучения, написанный на Python и работающий поверх платформы машинного обучения TensorFlow.
8. Matplotlib – это комплексная библиотека для создания статических, анимированных и интерактивных визуализаций на Python.
9. Seaborn – это библиотека для создания статистической графики на Python. Он построен поверх matplotlib и тесно интегрируется со структурами данных pandas.

#### Анализ PTB-XL

Загрузка данных происходит с помощью кода, предоставленного в источнике набора данных PTB-XL. Благодаря этому, мы сразу имеем тренировочный и тестовый выборки, разделенные для достижения наилучшего результата.

Переменные X\_train и X\_test – это 3-х мерные тензоры, которые содержат образцы, представленные 1000 массивами по 12 элементов в каждом. Каждому образцу соответствует метка, содержащаяся в переменной y\_train или y\_test соответственно.

Всего, по условию задачи, работа ведется с 5 классами: NORM, STTC, CD, MI, HYP. Пусть метка, которая содержит несколько классов называется составной. Тогда, так как одному образцу может соответствовать несколько классов одновременно (по одному результату ЭКГ ставится в соответствие несколько заключений), то было решено выполнить разделение таких составных меток на несколько, с соблюдением принадлежности каждой исходному образцу. Для этого была написана функция splitData(), которая разделяет метки с двумя и более классами в составе.

Также, среди меток присутствуют пустые значения (непомеченные образцы). Такие данные было принято удалить из доступных выборок для того, чтобы убрать выбросы и улучшить результат работы модели. С этим справляется написанная функция emptyLabel().

Таким образом, все данные, с которыми будет работать модель машинного обучения, будут фильтроваться с удалением пустых значений и разделением меток на несколько, если они являются составными.

Для улучшения качества работы моделей была применена нормализация данных X\_train и X\_test. Все значения были приведены к числовому диапазону от -1 до 1. Таким образом исключается получения чисел, выходящих за диапазон значений типа данных float64.

Также, были обработаны категориальные атрибуты переменных y\_train и y\_test. При решении задачи классификации значения этих переменных можно представить в виде разряженных меток (для каждого образца имеется только индекс целевого класса) или в виде массивов с указанием целевой вероятности на класс (количество элементов массива соответствует числу классов в задаче классификации). В нашем исследовании использован вариант с разряженными метками.

Следующим шагом стало написание нескольких архитектур нейронных сетей, которые смогут классифицировать различные показатели ЭКГ и с высокой вероятностью ставить в соответствие входному образцу один из 5 классов. Были реализованы простая полносвязная, сверточная одномерная и сверточная двумерная нейронные сети, а также загружены с помощью пакета keras и добавлены к исследованию такие сети как ResNet50, ResNet50V2, ResNet101, VGG16, VGG19. В случае полносвязной сети входные данные имели форму (1000, 12), одномерной и двумерной сверточных сетей – (1000, 12, 1), ResNet50, ResNet50V2, ResNet101, VGG16, VGG19 – (125, 32, 3). Во всех случаях использовалась оптимизация nadam, кроме полносвязной сети, где использовался стандартный стохастический градиентный спуск.

Для оценки качества моделей и сравнения различных алгоритмов применялась метрика accuracy – то есть вычисление того, как часто предсказания модели совпадали с метками.

Проверочными данными (validation data) была выбрана пятая часть от тренировочной выборки.

Для улучшения качества работы модели было реализовано раннее прекращение после 10 эпох ухудшения результата, а также сохранение весов модели при улучшении относительно ошибки на проверочных данных (validation loss), т.е. для оценки на тестовых данных используется модель с весами, показавшая наилучший результат в процессе обучения.

Для каждой модели были построены графики, демонстрирующие изменение точности и ошибки на тренировочных и проверочных данных в ходе обучения, а также проведена оценка работы на тестовых данных.

#### Анализ статьи[[3]](#footnote-3)

Авторами статьи в совокупности с кодом, предложенным для загрузки и распаковки PTB-XL, был автоматизирован процесс загрузки и обработки обоих наборов данных. Я воспользовался лишь частью предложенных ресурсов по следующим причинам:

1. В статье рассматривается множество методов глубокого обучения, из которых я рассматриваю представителей рекуррентных нейронных сетей (RNN), таких как LSTM (lstm) – сеть долгой краткосрочной памяти и bidirectional LSTM (lstm\_bidir) – двунаправленная LSTM.
2. Из-за давней публикации статьи не работают некоторые методы различных пакетов Python, которые были удалены с новыми версиями и не являются более актуальными и эффективными.
3. При загрузке данных ICBEB в файле utils.py, предложенным авторами статьи, в некоторых местах пришлось переписывать участки кода, чтобы исправишь ошибки, возникающие при обработке данных разной формы.

В статье для сравнения производительности различных моделей предложено вычислять метрики AUC, F2 метрику и G2 метрику. AUC выбрана в качестве основной метрики для всех экспериментов по классификации. Поскольку присутствует дисбаланс классов, то применяется макро-усреднение. Макро-усредненные F2 и G2 метрики использовались в качестве оценочных показателей для моделей, обученных на наборе ICBEB2018.

В задаче классификации проводилось 6 экспериментов под названиями ‘all’, ‘diagnostic’, ‘superdiagnostic’, ‘subdiagnostic’, ‘rhythm’, ‘form’.

Мною были проведены все, кроме эксперимента ‘all’. Также я использовал метрики F2 и G2 при анализе методов, обученных на наборе ICBEB2018.

Помимо работы с наборами данных по отдельности, авторами статьи было предложено провести эксперименты по трансферному обучению с применением тонкой подстройки.

# Результаты

#### Результаты исследования набора данных PTB-XL

Сравнивались оценки работы разных моделей на тестовых данных. Все результаты вынесены в таблицу:

|  |  |
| --- | --- |
| **Модель** | **[Ошибка, Точность]** |
| Полносвязная нейронная сеть | [1.5121192932128, 0.32524099946022] |
| Одномерная сверточная нейронная сеть | [0.987737774848, 0.6033559441566] |
| Двумерная сверточная нейронная сеть | [1.0278685092926, 0.6012138724327] |
| ResNet50 | [1.1360713243484, 0.5483756065368] |
| ResNet50V2 | [1.0847526788712, 0.5501606464386] |
| ResNet101 | [1.4233031272888, 0.42234915494919] |
| VGG16 | [1.535159707069, 0.34416279196739] |
| VGG19 | [1.5358237028122, 0.34416279196739] |

Таб. 1 Оценка работы различных моделей на тестовых данных

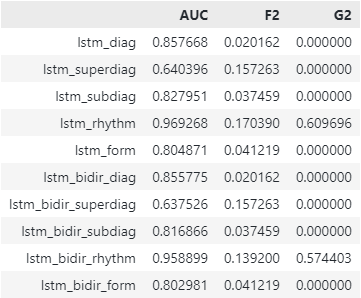
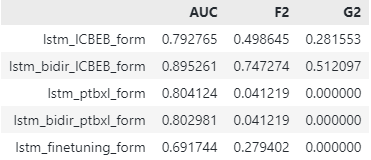
Левое значение означает, какую наибольшую ошибку допустила модель при прогнозировании меток на тестовой выборке, а правое точность – процент верно классифицированных образцов из тестовой выборки.

Наилучший по двум значениям результат показала одномерная сверточная нейронная сеть, наибольшую ошибку допустили сети VGG16 и VGG19, а наихудшую точность показала полносвязная нейронная сеть, с почти такой же большой ошибкой. Близкой по обоим показателям к наилучшему результату стала двумерная сверточная сеть, а следом идут ResNet50 и ResNet50V2. Модель на основе ResNet101 показывает заметно ухудшенный результат, по сравнению с ResNet50 и ResNet50V2.

Таким образом можно сделать следующий вывод на основании полученных результатов: с данными в виде сигналов ЭКГ, которые представляют собой показатели (некоторые числовые значения) по 12 отведениями, получаемые в течение 10 секунд, наилучшим образом работает одномерная сверточная нейронная сеть.

#### Результаты исследования на основе статьи[[4]](#footnote-4)

Сравнивалась работа моделей lstm и lstm\_bidir на основе проведения 5 экспериментов для каждого набора данных. Также был проведен 1 эксперимент для каждой модели в сфере трансферного обучения. Результаты представлены на таблицах 2,3,4

На таб. 2 мы видим, что в большинстве экспериментов модели lstm и lstm\_bidir имеют очень близкие значений по показателю AUC, что соответствует статье. По показателю F2 уже больше различий: в экспериментах ‘diagnostic’ и ‘superdiagnostic’ lstm имеет немного большие значения, чем другая модель, но в экспериментах ‘subdiag’ и ’form’ гораздо большие значения имеет модель lstm\_bidir. По показателю G2 ситуация аналогичная. Значений приведенной таблицы близки к оригинальной статье.

Таб. 4. Результаты трансферного обучения. Модель сначала обучалась на наборе данных PTB-XL, а дальше, с применением трансферного обучения, обучалась на наборе ICBEB2018 с тонкой подстройкой

Таб. 2. Оценки работы моделей, обученных на ICBEB2018

Таб. 3. Оценки работы моделей, обученных на PTB-XL

На таб. 3 мы видим, что по показателю AUC модели lstm и lstm\_bidir имеют достаточно близкие значения, с небольшим уклоном в сторону модели lstm. По показателю F2 модели очень похоже себя показывают, за исключением эксперимента ‘rhythm’, где показатель модели lstm заметно больше. Относительно статьи мои результаты не похожи и могут иметь сильные отклонения, хотя по некоторым экспериментам, например, ‘rhythm’, результаты близки.

На таб. 4 приведен неудачный результат трансферного обучения. Был выбран эксперимент ‘form’ для обоих наборов данных и показатель AUC оказался сильно ниже для моделей, обученных на наборе данных PTB-XL, относительно моделей, обученных на наборе данных ICBEB2018. Поэтому результатом итоговой модели после тонкой подстройки является достаточно низкое значение показателей AUC и F2.

# Заключение

Исследование набора данных PTB-XL и создание моделей машинного обучения на его основе является актуальной и сложной задачей, поиск эффективного решения которой приближает человека к созданию автоматизированной поддержки работы врачей по принятию решений об оказании помощи и лечении пациента.

В ходе работы были изучены техники создания моделей сверточных нейронных сетей с применением актуальных версий библиотек машинного обучения, а также применены различные подходы по обработке исходных данных перед их использованием. Работа созданных моделей сверточных нейронных сетей была оценена на тестовых данных в целях получения информации об ошибке на тестовой выборке и точности классификации образцов. Их работа сравнивалась между собой и с известными архитектурами сверточных нейронных сетей. Анализ и сравнение полученных результатов позволяет принимать новые оптимальные решения по настройке гиперпараметров и построению архитектуры модели, а также находить новые пути в представлении и обработке данных перед применением на них моделей машинного обучения.

В ходе работы была рассмотрена статья “Deep Learning for ECG Analysis: Benchmarks and Insights from PTB-XL”, что позволило расширить область знаний о работе различных алгоритмов глубокого обучения с данными ЭКГ, получены более эффективные способы обработки сигналов, а также было разобрано представление набора данных PTB-XL как показательного, при тестировании различных методов машинного обучения.

# Литература

1. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow by Aurélien Géron ISBN: 978-1-492-03264-9;
2. ISBN: 978-1-491-91721-3 Introduction to Machine Learning with Python by Andreas C. Mueller and Sarah Guido;
3. Ш78 Глубокое обучение на Python. — СПб.: Питер, 2018. — 400 с.: ил. — (Серия «Биб- лиотека программиста»). ISBN 978-5-4461-0770-4;
4. Липчак Д. А. Обзор методов автоматической диагностики сердечной аритмии для принятия решений о необходимости проведения дефибрилляции / Д. А. Липчак, А. А. Чупов // Ural Radio Engineering Journal. — 2021. — Vol. 5, No. 4. — P. 380–409;
5. ´Smigiel, S.; Pałczy ´ nski, K.; Ledzi ´ nski, D. ECG Signal Classification Using Deep Learning Techniques Based on the PTB-XL Dataset. Entropy 2021, 23, 1121. <https://doi.org/10.3390/e23091121>;
6. Mehmood, A.; Maqsood, M.; Bashir, M.; Shuyuan, Y. A Deep Siamese Convolution Neural Network for Multi-Class Classification of Alzheimer Disease. Brain Sci. **2020**, 10, 84. <https://doi.org/10.3390/brainsci10020084>
7. Hsieh, C.-H.; Li, Y.-S.; Hwang, B.-J.; Hsiao, C.-H. Detection of Atrial Fibrillation Using 1D Convolutional Neural Network. Sensors **2020**, 20, 2136. <https://doi.org/10.3390/s20072136>
8. Nils Strodthoff, Patrick Wagner, T. Schaeffter, W. Samek, “Deep Learning for ECG Analysis: Benchmarks and Insights from PTB-XL”, DOI: 10.1109/JBHI.2020.3022989Corpus ID: 216562803, Published 28 April 2020, IEEE Journal of Biomedical and Health Informatics
9. S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Comput., vol. 9, no. 8, pp. 1735–1780, 1997.
10. F. Chollet et al., “Keras,” 2015. [Online]. Available: <https://keras.io>
11. ´Smigiel, S.; Pałczy ´ nski, K.; Ledzi ´ nski, D. ECG Signal Classification Using Deep Learning Techniques Based on the PTB-XL Dataset. Entropy 2021, 23, 1121. https://doi.org/10.3390/e23091121

# Электронные источники

1. <https://www.nature.com/articles/s41597-020-0495-6>;
2. <https://physionet.org/content/ptb-xl/1.0.1/>;
3. <https://www.tensorflow.org/>;
4. <https://keras.io/>;
5. https://github.com/helme/ecg\_ptbxl\_benchmarking

# Приложения

**Подключение основных библиотек и загрузка данных**

## Для Google Colaboratory

In [1]:

*# Библиотека wfdb отсутствует в Google Collab*

**!**pip install wfdb

In [2]:

*# Подключение Google Drive к виртуальной машине.*

**from** google.colab **import** drive drive**.**mount('/content/drive')

Drive already mounted at /content/drive; to attempt to forcibly remount, call driv e.mount("/content/drive", force\_remount=True).

*# Копирование данных с Google Drive на локальный диск виртуальной машины.*

In [3]:

**!**cp -r /content/drive/MyDrive/MachineLearningCollab/X\_testCopy.npy .

**!**cp -r /content/drive/MyDrive/MachineLearningCollab/X\_trainCopy.npy .

**!**cp -r /content/drive/MyDrive/MachineLearningCollab/y\_testCopy.npy .

**!**cp -r /content/drive/MyDrive/MachineLearningCollab/y\_trainCopy.npy .

**Подключение библиотек**

In [1]:

*# Подключение библиотек.*

**import** pandas **as** pd **import** numpy **as** np **import** wfdb

**import** ast

**import** pandoc

*#import sklearn*

**from** sklearn.preprocessing **import** OrdinalEncoder, OneHotEncoder

**import** matplotlib.pyplot **as** plt *#import matplotlib.cm as cm* **import** seaborn **as** sns

**import** tensorflow **as** tf

**from** tensorflow **import** keras

**assert** tf**.** version **>=** "2.0"

print("tf. version :", tf**.** version )

print("keras. version ", keras**.** version )

**from** keras **import** layers

**from** keras **import** models

**Первая загрузка данных**

In [2]:

np**.**random**.**seed(42)

**def** load\_raw\_data(df, sampling\_rate, path):

**if** sampling\_rate **==** 100:

data **=** [wfdb**.**rdsamp(path**+**f) **for** f **in** df**.**filename\_lr] *# low rate*

**else**:

data **=** [wfdb**.**rdsamp(path**+**f) **for** f **in** df**.**filename\_hr] *# high rate*

data **=** np**.**array([signal **for** signal, meta **in** data])

**return** data

path **=** 'part/to/ptbxl/' sampling\_rate **=** 100

*# load and convert annotation data*

Y **=** pd**.**read\_csv(path**+**'ptbxl\_database.csv', index\_col **=** 'ecg\_id') Y**.**scp\_codes **=** Y**.**scp\_codes**.**apply(**lambda** x: ast**.**literal\_eval(x))

*# Load raw signal data*

X **=** load\_raw\_data(Y, sampling\_rate, path)

*# load scp\_statements.csv for diagnostic aggregation*

agg\_df **=** pd**.**read\_csv(path**+**'scp\_statements.csv', index\_col **=** 0) agg\_df **=** agg\_df[agg\_df**.**diagnostic **==** 1]

**def** aggregate\_diagnostic(y\_dic): tmp **=** []

**for** key **in** y\_dic**.**keys():

**if** key **in** agg\_df**.**index:

tmp**.**append(agg\_df**.**loc[key]**.**diagnostic\_class)

**return** list(set(tmp))

*# apply diagnistic superclass*

Y['diagnostic\_superclass'] **=** Y**.**scp\_codes**.**apply(aggregate\_diagnostic)

*# Split data into train and test*

test\_fold **=** 10

*# Train*

X\_train **=** X[np**.**where(Y**.**strat\_fold **!=** test\_fold)]

y\_train **=** Y[(Y**.**strat\_fold **!=** test\_fold)]**.**diagnostic\_superclass

*# Test*

X\_test **=** X[np**.**where(Y**.**strat\_fold **==** test\_fold)]

y\_test **=** Y[Y**.**strat\_fold **==** test\_fold]**.**diagnostic\_superclass

**del** X, Y

**Некоторые функции**

In [ ]:

*# Функция, которая разделит списки с 2 и более элементами.*

**def** splitData(X, y):

indexDel **=** [] *# хранит индексы список, которые хранят более 1 элемента (удаляютс* yTmp **=** np**.**zeros((0, 1)) *# хранит элементы (списки с 1 элементом), которые добавл* XTmp **=** np**.**zeros((0, 1000, 12)) *# хранит элементы (списки с 1 элементом), которые* **for** i **in** range(y**.**size):

**if** len(y[i]) **>** 1:

indexDel**.**append(i)

**for** j **in** range(len(y[i])):

yTmp **=** np**.**concatenate((yTmp, np**.**array([y[i][j:j**+**1]]))) XTmp **=** np**.**concatenate((XTmp, np**.**array([X[i]])))

y **=** np**.**delete(y, indexDel, axis **=** 0) X **=** np**.**delete(X, indexDel, axis **=** 0) y **=** y**.**reshape(y**.**size, 1)

**for** i **in** range(y**.**size): y[i] **=** (y[i]**.**tolist()[0])

y **=** np**.**concatenate((y, yTmp)) X **=** np**.**concatenate((X, XTmp)) **return** X, y

*# Поиск пустых значений в массиве меток.*

**def** emptyLabel(y):

*# Найдем ecg\_id строк, где не определен класс.*

ecg\_idEmpty **=** [] *# массив индексов пустых значений по ecg\_id [1, ...]*

ecg\_idEmptyIndex **=** [] *# массив индекс пустых значений по [0, ...]*

sum\_empty **=** 0 *# подсчет пустых значений*

**for** i **in** range(y**.**size):

**if** len(y**.**values[i]) **==** 0:

ecg\_idEmpty**.**append(y**.**index[i]) ecg\_idEmptyIndex**.**append(i)

sum\_empty **+=** 1

*# print(sum\_empty)*

*# print(ecg\_idEmpty)*

*# print(ecg\_idEmptyIndex)*

**return** sum\_empty, ecg\_idEmpty, ecg\_idEmptyIndex

**Анализ данных**

**Первичный анализ**

In [ ]:

*# Вывести конкретное (все, если указана бесконечность в качестве параметра thresho # np.set\_printoptions(threshold=10) # np.inf*

In [ ]:

*# Тип данных переменных y\_train и y\_test, содержащие метки (диагноз).*

print(type(y\_train)) print(type(y\_test)) *# Размерность.*

print(y\_train**.**ndim) print(y\_test**.**ndim)

*# Форма переменных и их размер (количество меток).*

print(y\_train**.**shape, y\_train**.**size) print(y\_test**.**shape, y\_test**.**size)

*# Общее количество элементов (меток).*

print("size:", y\_train**.**size **+** y\_test**.**size)

<class 'pandas.core.series.Series'>

<class 'pandas.core.series.Series'> 1

1

(19634,) 19634

(2203,) 2203

size: 21837

In [ ]:

*# Тип данных переменных X\_train и X\_test, содержащие все объекты (экземпляры, обра*

print(type(X\_test)) print(type(X\_train)) *# Размерность.*

print(X\_train**.**ndim) print(X\_test**.**ndim)

*# Форма переменных и их размер (количество образцов).*

print(X\_train**.**shape, X\_train**.**size) print(X\_test**.**shape, X\_test**.**size)

*# Общее количество элементов (образцов).*

print("size:", X\_train**.**size **+** X\_test**.**size)

<class 'numpy.ndarray'>

<class 'numpy.ndarray'> 3

3

(19634, 1000, 12) 235608000

(2203, 1000, 12) 26436000

size: 262044000

In [ ]:

*# Посмотрим на исходный вид меток.*

print(y\_train)

ecg\_id

1. [NORM]
2. [NORM]
3. [NORM]
4. [NORM]
5. [NORM]

...

21833 [STTC]

21834 [NORM]

21835 [STTC]

21836 [NORM]

21837 [NORM]

Name: diagnostic\_superclass, Length: 19634, dtype: object

In [ ]:

*# Посмотрим на исходный вид показателей.*

print(X\_train)

[[[-1.190e-01 -5.500e-02 6.400e-02 ... -2.600e-02 -3.900e-02 -7.900e-02]

[-1.160e-01 -5.100e-02 6.500e-02 ... -3.100e-02 -3.400e-02 -7.400e-02]

[-1.200e-01 -4.400e-02 7.600e-02 ... -2.800e-02 -2.900e-02 -6.900e-02]

...

[ 6.900e-02 0.000e+00 -6.900e-02 ... 2.400e-02 -4.100e-02 -5.800e-02]

[ 8.600e-02 4.000e-03 -8.100e-02 ... 2.420e-01 -4.600e-02 -9.800e-02]

[ 2.200e-02 -3.100e-02 -5.400e-02 ... 1.430e-01 -3.500e-02 -1.200e-01]]

[[ 4.000e-03 1.380e-01 1.340e-01 ... 1.920e-01 8.300e-02 8.800e-02] [-2.000e-02 1.160e-01 1.360e-01 ... 1.560e-01 5.700e-02 6.300e-02] [-5.300e-02 9.200e-02 1.450e-01 ... 1.070e-01 1.300e-02 2.200e-02]

...

[ 1.210e-01 3.980e-01 2.770e-01 ... -1.065e+00 -4.920e-01 -1.560e-01]

[-3.260e-01 5.700e-02 3.830e-01 ... -2.800e-01 -1.750e-01 -7.100e-02]

[-3.480e-01 -5.600e-02 2.920e-01 ... -3.080e-01 -2.310e-01 -1.450e-01]]

[[-2.900e-02 -7.900e-02 -4.900e-02 ... -1.030e-01 -7.600e-02 -6.600e-02]

[-3.500e-02 -7.000e-02 -3.500e-02 ... -1.040e-01 -7.900e-02 -6.800e-02]

[-5.400e-02 -5.700e-02 -3.000e-03 ... -7.800e-02 -6.600e-02 -5.400e-02]

...

[-2.900e-02 -2.260e-01 -1.980e-01 ... 1.000e-03 2.290e-01 1.800e-02]

[-4.800e-02 -2.660e-01 -2.180e-01 ... -1.000e-03 2.100e-02 -8.000e-03]

[-4.900e-02 -2.880e-01 -2.390e-01 ... 1.000e-03 -1.800e-02 6.000e-03]]

...

[[ 3.800e-02 2.400e-02 -1.400e-02 ... 6.000e-03 1.600e-02 2.700e-02] [ 7.800e-02 6.600e-02 -1.200e-02 ... 1.600e-02 3.100e-02 4.500e-02] [-1.400e-02 -6.000e-03 8.000e-03 ... 9.000e-03 3.000e-02 4.700e-02]

...

[-1.060e-01 -6.200e-02 4.400e-02 ... -2.120e-01 -7.400e-02 5.100e-02]

[-4.500e-02 2.600e-02 7.100e-02 ... -1.510e-01 -5.900e-02 4.800e-02]

[ 4.630e-01 5.300e-01 6.700e-02 ... -1.810e-01 -1.050e-01 3.300e-02]]

[[-5.700e-02 -5.700e-02 0.000e+00 ... -3.500e-02 -3.900e-02 -3.500e-02]

[-4.100e-02 -2.900e-02 1.200e-02 ... -2.300e-02 -2.800e-02 -2.700e-02]

[ 3.000e-03 4.500e-02 4.200e-02 ... -9.000e-03 -1.400e-02 -1.400e-02]

...

[ 3.300e-02 7.000e-02 3.700e-02 ... 2.180e-01 1.010e-01 5.200e-02] [ 2.700e-02 8.200e-02 5.500e-02 ... 2.100e-01 3.350e-01 1.000e-02] [-6.000e-03 5.100e-02 5.700e-02 ... 2.110e-01 3.740e-01 -9.000e-03]]

[[-4.900e-02 -2.500e-02 2.400e-02 ... -4.000e-02 -2.600e-02 -3.100e-02]

[-4.900e-02 -2.900e-02 2.000e-02 ... -3.200e-02 -3.300e-02 -4.300e-02]

[-5.900e-02 -4.200e-02 1.700e-02 ... -4.000e-02 -4.700e-02 -5.100e-02]

...

[ 9.100e-02 1.200e-02 -7.900e-02 ... 6.300e-02 1.600e-02 -1.170e-01]

[ 1.750e-01 2.200e-02 -1.530e-01 ... 8.000e-02 1.800e-02 -1.080e-01]

[ 1.660e-01 -7.000e-03 -1.730e-01 ... 1.060e-01 4.700e-02 -1.030e-01]]]

In [ ]:

*# Содержимое по индексам и значения (тип данных Series, DataFrame пакета Pandas)*

print(y\_train**.**index, '\n') print(y\_train**.**values, '\n') print(y\_train**.**index[0])

print(y\_train**.**values[0])

Int64Index([ 1, 2, 3, 4, 5, 6, 7, 8, 10,

11,

...

21828, 21829, 21830, 21831, 21832, 21833, 21834, 21835, 21836,

21837],

dtype='int64', name='ecg\_id', length=19634)

[list(['NORM']) list(['NORM']) list(['NORM']) ... list(['STTC']) list(['NORM']) list(['NORM'])]

1

['NORM']

In [2]:

*# Посмотрим на содержание y\_train (все данные).*

**for** i **in** range(y\_train**.**size): *# можно вычесть некоторое значение, не большее y\_tra*

print(y\_train**.**values[i])

*# Или так.*

*# for Class in y\_train: # #print(Class)*

**Пропущенные данные**

**Когда мы выделим уникальные значения в y\_train и проведем подсчет каждого, то заметим, что существует пустые значения, которые повторяются 367 раз в тренировочном наборе.**

In [ ]:

*# Создать Ndarray Numpy копию Series Pandas.*

y\_trainNp **=** y\_train**.**to\_numpy() print(type(y\_trainNp))

print(y\_trainNp**.**size)

<class 'numpy.ndarray'> 19634

In [ ]:

*# Выделим уникальные значения y\_train\_np и посчитаем их количество.*

unique, counts **=** np**.**unique(y\_trainNp, return\_counts **= True**) print(unique, '\n', "unique.size: ", unique**.**size)

print(counts)

[list([]) list(['CD']) list(['CD', 'HYP']) list(['CD', 'MI'])

list(['CD', 'MI', 'HYP']) list(['CD', 'MI', 'STTC'])

list(['CD', 'MI', 'STTC', 'HYP']) list(['CD', 'STTC'])

list(['CD', 'STTC', 'HYP']) list(['HYP'])

list(['HYP', 'CD', 'MI', 'STTC']) list(['HYP', 'CD', 'STTC'])

list(['HYP', 'MI', 'STTC']) list(['HYP', 'STTC']) list(['MI'])

list(['MI', 'HYP']) list(['MI', 'STTC']) list(['MI', 'STTC', 'HYP'])

list(['NORM']) list(['NORM', 'CD']) list(['NORM', 'CD', 'MI', 'HYP'])

list(['NORM', 'CD', 'STTC']) list(['NORM', 'HYP']) list(['NORM', 'STTC'])

list(['STTC']) list(['STTC', 'CD', 'MI', 'HYP']) list(['STTC', 'HYP']) list(['STTC', 'MI', 'HYP'])]

unique.size: 28

[ 367 1525 273 1167 112 202 1 434 135 480 14 51 19 149

2282 166 541 1 8170 362 1 5 2 24 2163 127 561 300]

In [ ]:

*# Визуализация с применением seaborn.*

*# Создаем копию.*

y\_trainTmp **=** y\_train

*# Конвертируем list в str.*

**for** i **in** range(y\_trainTmp**.**size):

y\_trainTmp**.**iloc[i] **=** ''**.**join(y\_trainTmp**.**iloc[i]) y\_train\_rI **=** y\_trainTmp**.**reset\_index()

y\_train\_rI\_cat **=** y\_train\_rI[['diagnostic\_superclass']]

colours **=** ['#000099', '#ffff00']

fig, ax **=** plt**.**subplots(figsize **=** (12, 8))

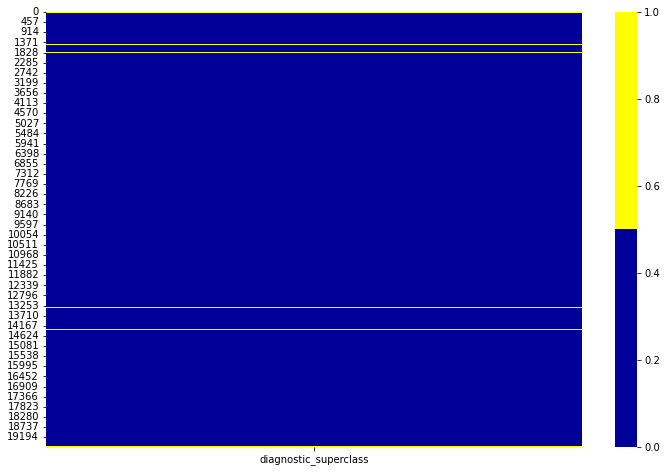
sns**.**heatmap(y\_train\_rI\_cat[:]**.**isin(['']), cmap **=** sns**.**color\_palette(colours))

Out[ ]:

In [ ]:

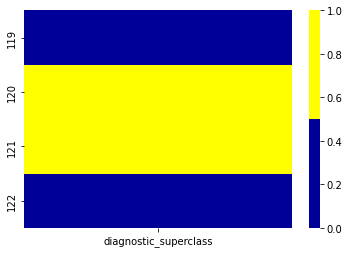
Out[ ]:

<AxesSubplot:>



sns**.**heatmap(y\_train\_rI\_cat[119:123]**.**isin(['']), cmap **=** sns**.**color\_palette(colours))

<AxesSubplot:>



In [ ]:

*# процентное отношение пропущенных данных*

**for** i **in** y\_train\_rI\_cat[0:1000]:

empty **=** np**.**mean(y\_train\_rI\_cat[i]**.**isin([''])) print('{} - {}%'**.**format(i, round(empty**\***100)))

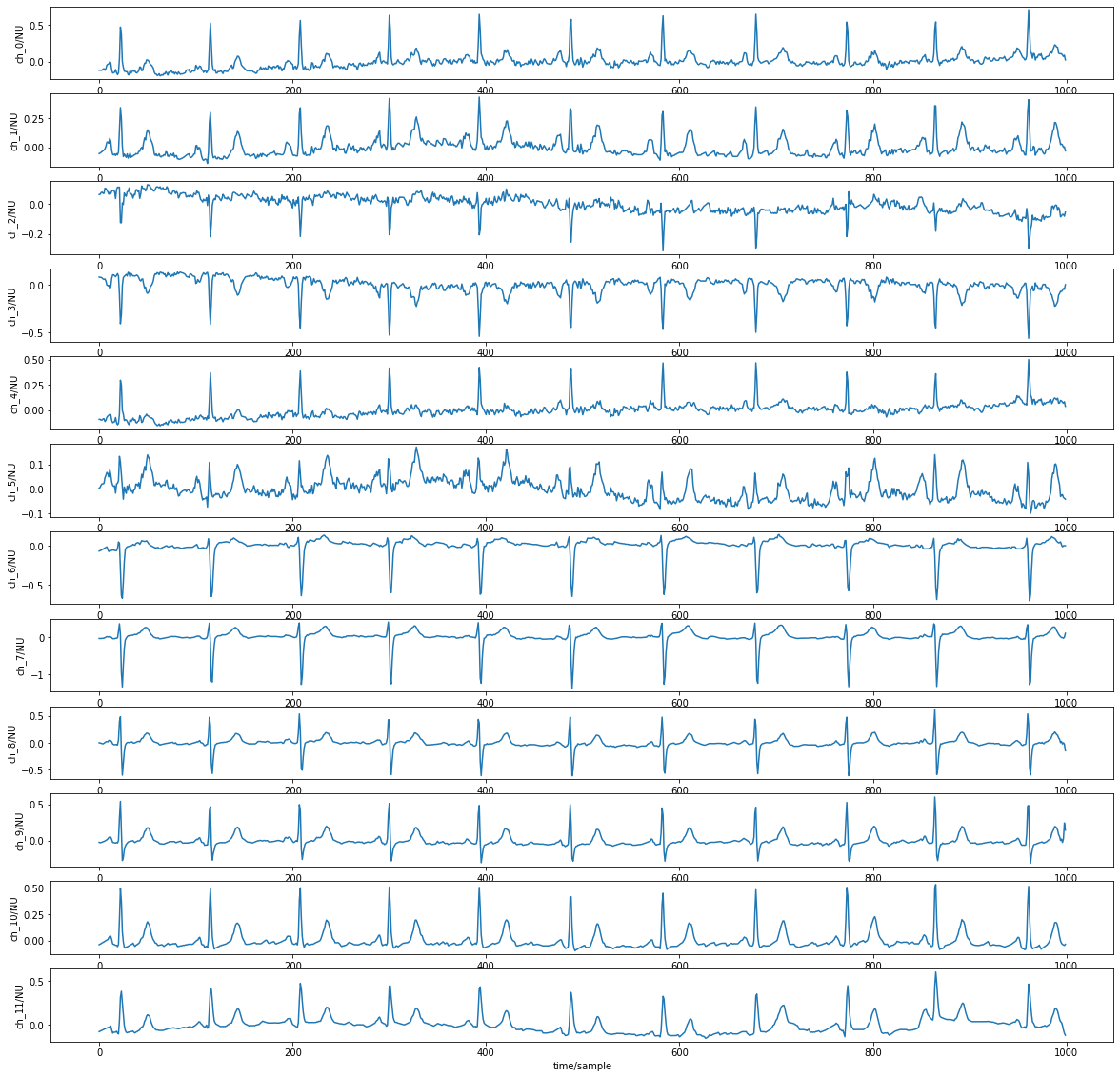
**del** y\_train\_rI\_cat

diagnostic\_superclass - 2%

## Визуализация данных

In [3]:

wfdb**.**plot\_items(signal **=** X\_train[0], figsize **=** (20, 20))



In [4]:

fig, ax **=** plt**.**subplots() *# figure, axes*

fig**.**set\_figwidth(30) fig**.**set\_figheight(20)

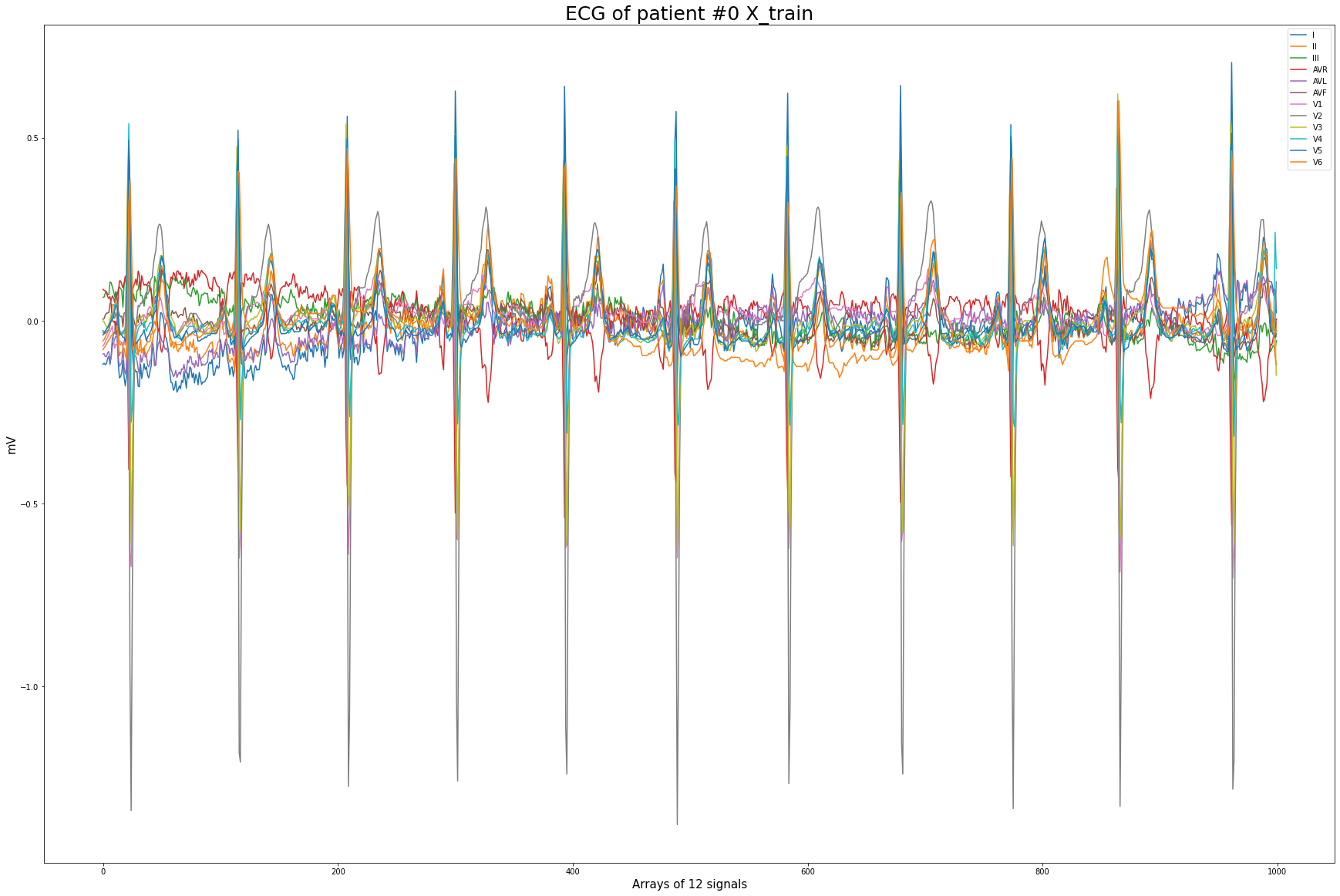
labels **=** ['I', 'II', 'III', 'AVR', 'AVL', 'AVF', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6

ax**.**plot(X\_train[0])

ax**.**set\_xlabel('Arrays of 12 signals', fontsize **=** 15.) ax**.**set\_ylabel('mV', fontsize **=** 15)

ax**.**set\_title('ECG of patient #0 X\_train', fontsize **=** 25.) ax**.**legend(labels) *# label in ax.plot*

**pass**



In [5]:

n\_sig\_min **=** 1

n\_sig\_max **=** 2

n\_arr\_min **=** 0

n\_arr\_max **=** 1000

steps **=** 1000

fig, ax **=** plt**.**subplots() *# figure, axes*

fig**.**set\_figwidth(10) fig**.**set\_figheight(8)

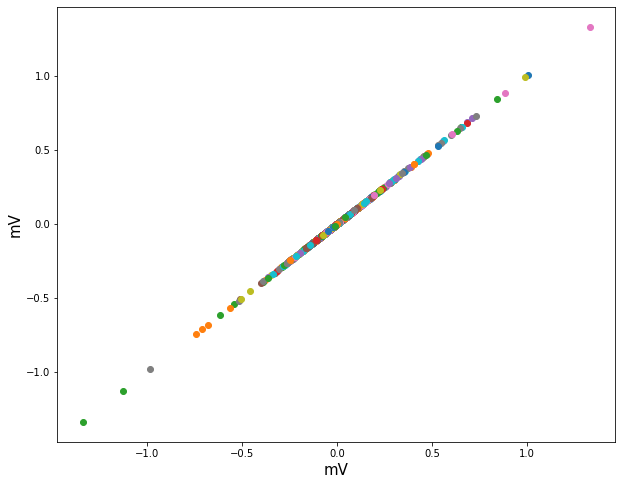
labels **=** np**.**array([i **for** i **in** np**.**arange(0, steps)])

**for** i **in** np**.**arange(n\_arr\_min, n\_arr\_max):

**for** j **in** np**.**arange(n\_sig\_min, n\_sig\_max):

ax**.**scatter(X\_train[i][j][0], X\_train[i][j][0])

ax**.**set\_xlabel('mV', fontsize **=** 15.) ax**.**set\_ylabel('mV', fontsize **=** 15) **pass**



In [6]:

fig, ax **=** plt**.**subplots() *# figure, axes*

fig**.**set\_figwidth(10) fig**.**set\_figheight(8)

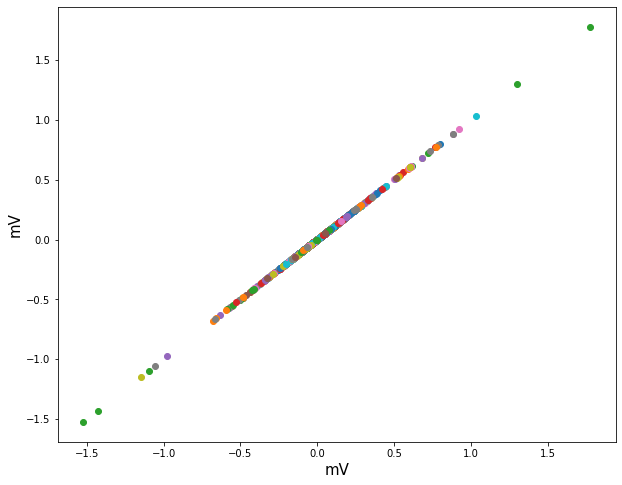
labels **=** np**.**array([i **for** i **in** np**.**arange(0, steps)])

**for** i **in** np**.**arange(n\_arr\_min, n\_arr\_max):

**for** j **in** np**.**arange(n\_sig\_min, n\_sig\_max):

ax**.**scatter(X\_train[i][j][1], X\_train[i][j][1])

ax**.**set\_xlabel('mV', fontsize **=** 15.) ax**.**set\_ylabel('mV', fontsize **=** 15) **pass**



|  |  |  |  |
| --- | --- | --- | --- |
|  | | | Предобработка данных |
| Копирование данных |
| In | [ | ]: | *# Копируем исходный тренировочный и тестовый набор, чтобы не испортить данные.* |
|  |  |  | y\_trainCopy **=** y\_train**.**copy() |
|  |  |  | X\_trainCopy **=** X\_train**.**copy() |
|  |  |  | y\_testCopy **=** y\_test**.**copy() |
|  |  |  | X\_testCopy **=** X\_test**.**copy() |
|  |  |  | Обработка пустых значений |
|  |  |  | Тренировочные данные |
|  |  |  | Обработка |
| In | [ | ]: | *# Выявление с помощью функции emptyLable() всех пустых меток.*  sum\_empty, ecg\_idEmpty, ecg\_idEmptyIndex **=** emptyLabel(y\_trainCopy) |
|  |  |  |  |
| In | [ | ]: | *# Проверка пустых значений.* |
|  |  |  | print("Проверка 1: ", y\_trainCopy**.**get(ecg\_idEmpty[:5])) |
|  |  |  | print("Проверка 2: ", y\_trainCopy**.**values[ecg\_idEmptyIndex[:5]]) |

Проверка 1: ecg\_id

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 17 | | | | [] |
| 18 | | | | [] |
| 20 | | | | [] |
| 23 | | | | [] |
| 34 | | | | [] |
| Name: | | | | diagnostic\_superclass, dtype: object |
|  |  |  | Проверка 2: [list([]) list([]) list([]) list([]) list([])] | |
| In | [ | ]: | *# Удалим соответствующие строки по ecg\_id в X\_trainCopy и в y\_trainCopy.*  y\_trainCopy**.**drop(labels **=** ecg\_idEmpty, axis **=** 0, inplace **= True**) | |
|  |  |  | *#y\_trainCopy.reset\_index(drop = True, inplace = True)* | |
|  |  |  | X\_trainCopy **=** np**.**delete(X\_trainCopy, ecg\_idEmptyIndex, axis **=** 0) | |
|  |  |  |  | |
| In | [ | ]: | *# Проверим размерности наших данных.* | |
|  |  |  | print(y\_trainCopy**.**shape, X\_trainCopy**.**shape) | |
|  |  |  | (19267,) (19267, 1000, 12) | |
| In | [ | ]: | *# Создать Ndarray Numpy копию Series Pandas c исключенными пустыми значениями.* | |
|  |  |  | y\_trainNpCopy **=** y\_trainCopy**.**to\_numpy() | |
|  |  |  | print(y\_trainNpCopy**.**size) | |
|  |  |  | 19267 | |
| In | [ | ]: | *# Разделяем метки с несколькими значениями: например, y[i] = [['NORM'], ['CD'], ['* | |
|  |  |  | X\_trainCopy, y\_trainNpCopy **=** splitData(X\_trainCopy, y\_trainNpCopy) | |
|  |  |  | *# Конвертируем list в str.* | |
|  |  |  | y\_trainCopy **=** [] | |
|  |  |  | **for** i **in** range(y\_trainNpCopy**.**size): | |
|  | | | y\_trainCopy**.**append(''**.**join(y\_trainNpCopy[i]**.**tolist())) y\_trainCopy **=** np**.**array(y\_trainCopy)  *# Удаление старого numpy объекта*  **del** y\_trainNpCopy  *# Сохранить файлы X\_trainCopy и y\_trainCopy, y\_trainNpCopy.*  np**.**save('X\_trainCopy', X\_trainCopy) np**.**save('y\_trainCopy', y\_trainCopy)  *# Размерности и содержимое данных.*  print(y\_trainCopy**.**shape, X\_trainCopy**.**shape, '\n')  *# Все уникальные классы.*  np**.**unique(y\_trainCopy, return\_counts **= True**) | |

Out[ ]:

In [5]:

*# Загрузка данных (не требуется выполнять код выше). y\_trainNpCopy не загружается.*

X\_trainCopy **=** np**.**load('X\_trainCopy.npy') y\_trainCopy **=** np**.**load('y\_trainCopy.npy')

(25025,) (25025, 1000, 12)

(array(['CD', 'HYP', 'MI', 'NORM', 'STTC'], dtype='<U4'), array([4409, 2392, 4933, 8564, 4727], dtype=int64))

Скачать

### Тестовые данные

Обработка

In [ ]:

*# Очистка данных с пустыми метками. # Выявление пустых значений.*

sum\_empty, ecg\_idEmpty, ecg\_idEmptyIndex **=** emptyLabel(y\_testCopy)

*# Удалим соответствующие строки по ecg\_id в X\_trainCopy и y\_trainCopy.*

y\_testCopy**.**drop(labels **=** ecg\_idEmpty, axis **=** 0, inplace **= True**) X\_testCopy **=** np**.**delete(X\_testCopy, ecg\_idEmptyIndex, axis **=** 0)

*# Создать ndarray Numpy копию Series Pandas.*

y\_testNpCopy **=** y\_testCopy**.**to\_numpy() print(y\_testNpCopy**.**size)

*# Разделяем метки с несколькими значениями.*

X\_testCopy, y\_testNpCopy **=** splitData(X\_testCopy, y\_testNpCopy)

*# Конвертируем list в str.*

y\_testCopy **=** []

**for** i **in** range(y\_testNpCopy**.**size):

y\_testCopy**.**append(''**.**join(y\_testNpCopy[i]**.**tolist())) y\_testCopy **=** np**.**array(y\_testCopy)

*# Сохранить файлы X\_testCopy и y\_testCopy.*

np**.**save('X\_testCopy', X\_testCopy) np**.**save('y\_testCopy', y\_testCopy)

*# Проверим размерности и содержимое данных.*

print(y\_testCopy**.**shape, X\_testCopy**.**shape)

2163

(2801,) (2801, 1000, 12)

Скачать

In [6]:

*# Загрузка тестовых файлов.*

X\_testCopy **=** np**.**load('X\_testCopy.npy') y\_testCopy **=** np**.**load('y\_testCopy.npy')

## Нормализация данных

### Тренировочные данные

In [7]:

*# Нормирование от -1 до 1: [-1, 1] значений всех образцов.*

*# Минимальное и максимальное значение в 3D массивe X\_trainCopy. (см. функции np.am*

print(np**.**amax(X\_trainCopy)) print(np**.**amin(X\_trainCopy)) *# По осям.*

*# print(np.amax(X\_trainCopy, axis = 0)) # print(np.amin(X\_trainCopy, axis = 0)) # print(np.amax(X\_trainCopy, axis = 1)) # print(np.amin(X\_trainCopy, axis = 1)) # print(np.amax(X\_trainCopy, axis = 2)) # print(np.amin(X\_trainCopy, axis = 2))*

*# Нахождение максимального по абсолютному значению.*

print(np**.**max(np**.**abs(X\_trainCopy)))

*# print(np.max(np.abs(X\_trainCopy), axis = 0)) # print(np.max(np.abs(X\_trainCopy), axis = 1)) # print(np.max(np.abs(X\_trainCopy), axis = 2))*

X\_trainCopyNorm **=** X\_trainCopy**/**(np**.**amax(np**.**abs(X\_trainCopy))) print(np**.**max(X\_trainCopyNorm))

print(np**.**min(X\_trainCopyNorm))

*# Аналогично.*

*# print(np.amax(X\_trainCopyNorm)) # print(np.amin(X\_trainCopyNorm))*

17.212

-20.032

20.032

0.8592252396166135

-1.0

### Тестовые данные

In [8]:

*# Нормирование от -1 до 1: [-1, 1]*

*# Нахождение максимального по абсолютному значению.*

print(np**.**max(np**.**abs(X\_testCopy)))

*# Нормировка.*

X\_testCopyNorm **=** X\_testCopy**/**(np**.**amax(np**.**abs(X\_trainCopy))) print(np**.**max(X\_testCopyNorm))

print(np**.**min(X\_testCopyNorm))

12.966

0.6472643769968051

-0.569888178913738

## Обработка текстовых и категориальных атрибутов по Geron "homl2RU". 1 вар: стр. 113; 2 вар: стр. 397.

### Тренировочные данные

1. вариант

In [9]:

*# 1 вар.*

*# Разреженные метки (для каждого образца имеется только индекс целевого класса от*

ordinal\_encoder **=** OrdinalEncoder()

y\_trainCopyEncoded **=** ordinal\_encoder**.**fit\_transform(pd**.**DataFrame(y\_trainCopy)) print(y\_trainCopyEncoded[:10])

print(ordinal\_encoder**.**categories\_)

[[3.]

[3.]

[3.]

[3.]

[3.]

[3.]

[3.]

[2.]

[3.]

[3.]]

[array(['CD', 'HYP', 'MI', 'NORM', 'STTC'], dtype=object)]

1. вариант

In [ ]:

*# 2 вар.*

*# Целевая вероятность на класс. (Применяется функция потери "categorical\_crossentr #в метки, представляемые векторами. Чтобы сделать наоборот: np.argmax() с axis = 1* cat\_encoder **=** OneHotEncoder(sparse**=False**)

y\_trainCopyHot **=** cat\_encoder**.**fit\_transform(pd**.**DataFrame(y\_trainCopy)) print(y\_trainCopyHot[:10])

print(cat\_encoder**.**categories\_)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [[0. | 0. | 0. | 1. | 0.] |
| [0. | 0. | 0. | 1. | 0.] |
| [0. | 0. | 0. | 1. | 0.] |
| [0. | 0. | 0. | 1. | 0.] |
| [0. | 0. | 0. | 1. | 0.] |
| [0. | 0. | 0. | 1. | 0.] |
| [0. | 0. | 0. | 1. | 0.] |
| [0. | 0. | 1. | 0. | 0.] |
| [0. | 0. | 0. | 1. | 0.] |
| [0. | 0. | 0. | 1. | 0.]] |

[array(['CD', 'HYP', 'MI', 'NORM', 'STTC'], dtype=object)]

### Тестовые данные

1 вариант

In [10]:

*# Разреженные метки (для каждого образца имеется только индекс целевого класса от*

ordinal\_encoder **=** OrdinalEncoder()

y\_testCopyEncoded **=** ordinal\_encoder**.**fit\_transform(pd**.**DataFrame(y\_testCopy)) y\_testCopyEncoded[:10]

Out[10]:

In [ ]:

*# Целевая вероятность на класс.*

cat\_encoder **=** OneHotEncoder(sparse**=False**)

y\_testCopyHot **=** cat\_encoder**.**fit\_transform(pd**.**DataFrame(y\_testCopy)) y\_testCopyHot[:10]

array([[3.],

[3.],

[3.],

[3.],

[3.],

[2.],

[0.],

[3.],

[3.],

[3.]])

2 вариант

Out[ ]:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| array([[0., | 0., | 0., | 1., | 0.], |
| [0., | 0., | 0., | 1., | 0.], |
| [0., | 0., | 0., | 1., | 0.], |
| [0., | 0., | 0., | 1., | 0.], |
| [0., | 0., | 0., | 1., | 0.], |
| [0., | 0., | 1., | 0., | 0.], |
| [1., | 0., | 0., | 0., | 0.], |
| [0., | 0., | 0., | 1., | 0.], |
| [0., | 0., | 0., | 1., | 0.], |
| [0., | 0., | 0., | 1., | 0.]]) |

**Создание, обучение и тестирование разных моделей нейронных сетей, в том числе известные архитектуры**

**Обучение модели будет проходить с использованием тренировочных (train) и проверочных (valid) данных.**

**Оценка работы модели производится с применением тестовых данных (test).**

|  |  |  |  |
| --- | --- | --- | --- |
|  | | | Полносвязная нейронная сеть |
| Преобразование данных для работы с сетью |
| In | [ | ]: | X\_train **=** X\_trainCopyNorm  y\_train **=** y\_trainCopyEncoded X\_test **=** X\_testCopyNorm  y\_test **=** y\_testCopyEncoded |
|  |  |  | print(X\_train**.**shape, y\_train**.**shape, X\_test**.**shape, y\_test**.**shape) |
|  |  |  | (25025, 1000, 12) (25025, 1) (2801, 1000, 12) (2801, 1) |
|  |  |  | Создание модели |
| In | [ | ]: | *# Создание обычной полносвязной модели.* |
|  |  |  | model **=** keras**.**models**.**Sequential()  model**.**add(keras**.**layers**.**Flatten(input\_shape **=** [1000,12])) *# входной слой (преобразо* |
|  | | | model**.**add(keras**.**layers**.**Dense(10000, activation**=**"relu")) *# скрытый слой*  model**.**add(keras**.**layers**.**Dense(3000, activation**=**"relu")) model**.**add(keras**.**layers**.**Dense(800, activation**=**"relu")) model**.**add(keras**.**layers**.**Dense(100, activation**=**"relu"))  model**.**add(keras**.**layers**.**Dense(5, activation**=**"softmax")) *# выходной слой*  model**.**summary() |
|  |  |  | Model: "sequential"  Layer (type) Output Shape Param #  ================================================================= |

|  |  |  |  |
| --- | --- | --- | --- |
| flatten (Flatten) | (None, | 12000) | 0 |
| dense (Dense) | (None, | 10000) | 120010000 |
| dense\_1 (Dense) | (None, | 3000) | 30003000 |
| dense\_2 (Dense) | (None, | 800) | 2400800 |
| dense\_3 (Dense) | (None, | 100) | 80100 |
| dense\_4 (Dense) | (None, | 5) | 505 |

=================================================================

Total params: 152,494,405

Trainable params: 152,494,405

Non-trainable params: 0

### Обучение модели

In [ ]:

*# Компиляция модели.*

model**.**compile(loss **=** "sparse\_categorical\_crossentropy", optimizer **=** "sgd",

metrics **=** ["accuracy"])

*# Реализация раннего прекращения.*

checkpoint\_filepath **=** './checkpoint\_Neural1/'

model\_checkpoint\_cb **=** tf**.**keras**.**callbacks**.**ModelCheckpoint( filepath**=**checkpoint\_filepath,

save\_weights\_only**=True**, save\_best\_only**=True**)

early\_stopping\_cb **=** keras**.**callbacks**.**EarlyStopping(

patience**=**10,

restore\_best\_weights**=True**)

*# Обучение.*

history **=** model**.**fit(X\_train, y\_train, epochs **=** 30, validation\_split **=** 0.2, callbac

*# Сохранение модели.*

model**.**save('Neural1.h5')

*# Откат к наилучшей модели.*

model**.**load\_weights(checkpoint\_filepath)

|  |  |  |  |
| --- | --- | --- | --- |
| Epoch 1/30  626/626 [==============================] - 19s 26ms/step | - loss: | 1.4712 | - accurac |
| y: 0.4194 - val\_loss: 1.9151 - val\_accuracy: 0.0314 Epoch 2/30  626/626 [==============================] - 15s 24ms/step | - loss: | 1.4406 | - accurac |
| y: 0.4199 - val\_loss: 1.8737 - val\_accuracy: 0.0320 |  |  |  |
| Epoch 3/30  626/626 [==============================] - 16s 25ms/step | - loss: | 1.4249 | - accurac |
| y: 0.4242 - val\_loss: 1.8646 - val\_accuracy: 0.0426 Epoch 4/30 |  |  |  |
| 626/626 [==============================] - 16s 25ms/step  y: 0.4353 - val\_loss: 1.8231 - val\_accuracy: 0.0611 Epoch 5/30  626/626 [==============================] - 15s 25ms/step | * loss: * loss: | 1.4102  1.3988 | * accurac * accurac |
| y: 0.4404 - val\_loss: 1.8200 - val\_accuracy: 0.0723 Epoch 6/30  626/626 [==============================] - 16s 26ms/step | - loss: | 1.3905 | - accurac |
| y: 0.4429 - val\_loss: 1.7832 - val\_accuracy: 0.0883 |  |  |  |
| Epoch 7/30  626/626 [==============================] - 16s 25ms/step | - loss: | 1.3829 | - accurac |
| y: 0.4448 - val\_loss: 1.7764 - val\_accuracy: 0.0969 Epoch 8/30 |  |  |  |
| 626/626 [==============================] - 12s 20ms/step  y: 0.4477 - val\_loss: 1.8772 - val\_accuracy: 0.0783 Epoch 9/30  626/626 [==============================] - 13s 21ms/step | * loss: * loss: | 1.3758  1.3682 | * accurac * accurac |
| y: 0.4489 - val\_loss: 1.8730 - val\_accuracy: 0.0825 Epoch 10/30  626/626 [==============================] - 12s 20ms/step | - loss: | 1.3593 | - accurac |
| y: 0.4521 - val\_loss: 1.7819 - val\_accuracy: 0.0977 |  |  |  |
| Epoch 11/30  626/626 [==============================] - 16s 25ms/step | - loss: | 1.3487 | - accurac |
| y: 0.4565 - val\_loss: 1.7629 - val\_accuracy: 0.1061 Epoch 12/30 |  |  |  |
| 626/626 [==============================] - 13s 20ms/step  y: 0.4588 - val\_loss: 1.7880 - val\_accuracy: 0.1033 Epoch 13/30  626/626 [==============================] - 12s 19ms/step | * loss: * loss: | 1.3361  1.3199 | * accurac * accurac |
| y: 0.4635 - val\_loss: 1.8831 - val\_accuracy: 0.0933 Epoch 14/30  626/626 [==============================] - 12s 19ms/step | - loss: | 1.2979 | - accurac |
| y: 0.4702 - val\_loss: 1.7900 - val\_accuracy: 0.1109 |  |  |  |
| Epoch 15/30  626/626 [==============================] - 13s 20ms/step | - loss: | 1.2740 | - accurac |
| y: 0.4802 - val\_loss: 2.0018 - val\_accuracy: 0.0859 Epoch 16/30 |  |  |  |
| 626/626 [==============================] - 15s 24ms/step  y: 0.4917 - val\_loss: 1.5299 - val\_accuracy: 0.2064 Epoch 17/30  626/626 [==============================] - 13s 20ms/step | * loss: * loss: | 1.2446  1.2230 | * accurac * accurac |
| y: 0.5007 - val\_loss: 2.7188 - val\_accuracy: 0.0643 Epoch 18/30  626/626 [==============================] - 13s 20ms/step | - loss: | 1.1974 | - accurac |
| y: 0.5087 - val\_loss: 1.9089 - val\_accuracy: 0.1271 |  |  |  |
| Epoch 19/30  626/626 [==============================] - 13s 20ms/step | - loss: | 1.1646 | - accurac |
| y: 0.5249 - val\_loss: 1.6291 - val\_accuracy: 0.2036 Epoch 20/30 |  |  |  |
| 626/626 [==============================] - 12s 19ms/step  y: 0.5359 - val\_loss: 1.7214 - val\_accuracy: 0.1856 Epoch 21/30  626/626 [==============================] - 13s 20ms/step | * loss: * loss: | 1.1315  1.1102 | * accurac * accurac |
| y: 0.5426 - val\_loss: 1.5362 - val\_accuracy: 0.2226  Epoch 22/30 |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 626/626 [==============================] - 12s 20ms/step  y: 0.5603 - val\_loss: 1.5851 - val\_accuracy: 0.2212 | - loss: | 1.0735 | - accurac |
| Epoch 23/30  626/626 [==============================] - 13s 20ms/step | - loss: | 1.0467 | - accurac |
| y: 0.5689 - val\_loss: 1.5615 - val\_accuracy: 0.2324 Epoch 24/30 |  |  |  |
| 626/626 [==============================] - 12s 20ms/step  y: 0.5782 - val\_loss: 2.2638 - val\_accuracy: 0.1389 Epoch 25/30  626/626 [==============================] - 13s 20ms/step | * loss: * loss: | 1.0188  0.9898 | * accurac * accurac |
| y: 0.5932 - val\_loss: 1.8561 - val\_accuracy: 0.1924 Epoch 26/30  626/626 [==============================] - 13s 21ms/step | - loss: | 0.9651 | - accurac |
| y: 0.6049 - val\_loss: 1.5684 - val\_accuracy: 0.2671 |  |  |  |

<tensorflow.python.training.tracking.util.CheckpointLoadStatus at 0x7fea201feed0>

|  |  |  |
| --- | --- | --- |
| Out[ | ]: |  |
|  |  | Оценка работы модели |
| In [ | ]: | score **=** model**.**evaluate(X\_test, y\_test) score |

Out[ ]:

In [ ]:

acc **=** history**.**history['accuracy']

val\_acc **=** history**.**history['val\_accuracy'] loss **=** history**.**history['loss']

val\_loss **=** history**.**history['val\_loss']

epochs **=** range(1, len(acc) **+** 1)

plt**.**plot(epochs, acc, 'bo', label**=**'Training acc')

plt**.**plot(epochs, val\_acc, 'b', label**=**'Validation acc') plt**.**title('Training and validation accuracy')

plt**.**legend()

plt**.**figure()

plt**.**plot(epochs, loss, 'bo', label**=**'Training loss')

plt**.**plot(epochs, val\_loss, 'b', label**=**'Validation loss') plt**.**title('Training and validation loss')

plt**.**legend()

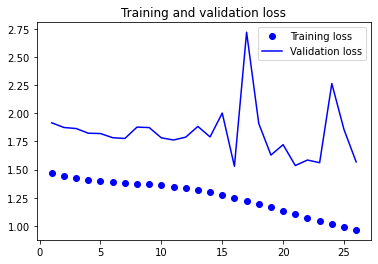
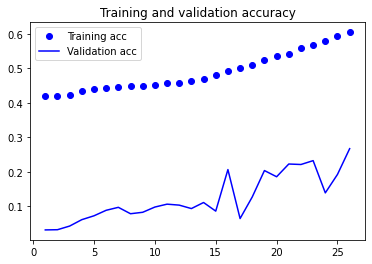
plt**.**show()

88/88 [==============================] - 1s 8ms/step - loss: 1.5121 - accuracy: 0.

3252

[1.5121192932128906, 0.32524099946022034]

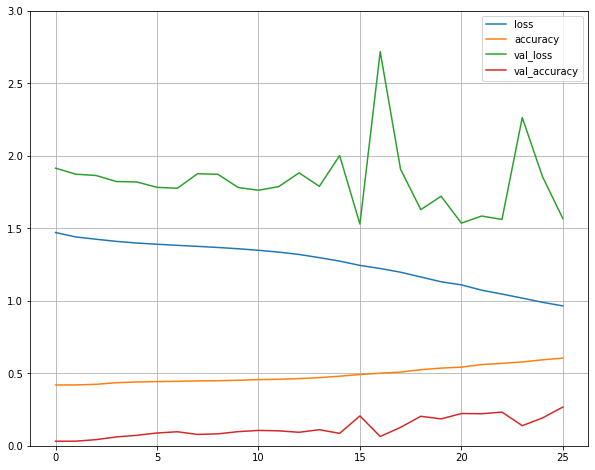
### Графики изменения точности и потерь модели по обучающим и проверочным данным в процессе обучения



In [ ]:

pd**.**DataFrame(history**.**history)**.**plot(figsize **=** (10, 8)) plt**.**grid()

plt**.**gca()**.**set\_ylim(0, 3) plt**.**show()



Одномерная сверточная нейроная сеть

### Преобразование данных для работы с сетью

In [ ]:

X\_train **=** X\_trainCopyNorm

y\_train **=** y\_trainCopyEncoded X\_test **=** X\_testCopyNorm

y\_test **=** y\_testCopyEncoded

print(X\_train**.**shape, y\_train**.**shape, X\_test**.**shape, y\_test**.**shape)

(25025, 1000, 12) (25025, 1) (2801, 1000, 12) (2801, 1)

### Создание модели

In [ ]:

*# \_3*

model **=** keras**.**Sequential([

keras**.**layers**.**Conv1D(filters **=** 8, kernel\_size **=** 16, padding **=** 'same', strides **=** 2 keras**.**layers**.**MaxPooling1D(pool\_size**=**8, strides **=** 4, padding **=** 'same'),

keras**.**layers**.**Conv1D(filters **=** 12, kernel\_size **=** 12, activation**=**'relu', padding **=**

keras**.**layers**.**MaxPooling1D(pool\_size**=**4, strides **=** 2, padding **=** 'same'),

keras**.**layers**.**Conv1D(filters **=** 32, kernel\_size **=** 9, padding **=** 'same', strides **=** 1 keras**.**layers**.**MaxPooling1D(pool\_size**=**5, strides **=** 2,padding **=** 'same'),

keras**.**layers**.**Conv1D(filters **=** 64, kernel\_size **=** 7, padding **=** 'same', strides **=** 1 keras**.**layers**.**MaxPooling1D(pool\_size**=**4, strides **=** 2, padding **=** 'same'),

keras**.**layers**.**Conv1D(filters **=** 64, kernel\_size **=** 5, padding **=** 'same', strides **=** 1

keras**.**layers**.**MaxPooling1D(pool\_size**=**2, strides **=** 2, padding **=** 'same'),

keras**.**layers**.**Conv1D(filters **=** 64, kernel\_size **=** 3, padding **=** 'same', strides **=** 1 keras**.**layers**.**MaxPooling1D(pool\_size**=**2, strides **=** 2, padding **=** 'same'),

keras**.**layers**.**Conv1D(filters **=** 72, kernel\_size **=** 3, padding **=** 'same', strides **=** 1

keras**.**layers**.**MaxPooling1D(pool\_size**=**2, strides **=** 2, padding **=** 'same'), keras**.**layers**.**Flatten(),

keras**.**layers**.**Dense(216, activation**=**'relu'), keras**.**layers**.**Dropout(0.1),

keras**.**layers**.**Dense(5, activation**=**'softmax')

])

model**.**summary()

Model: "sequential"

Layer (type) Output Shape Param #

=================================================================

conv1d (Conv1D) (None, 500, 8) 1544

max\_pooling1d (MaxPooling1D (None, 125, 8) 0

)

conv1d\_1 (Conv1D) (None, 63, 12) 1164

max\_pooling1d\_1 (MaxPooling (None, 32, 12) 0

1D)

conv1d\_2 (Conv1D) (None, 32, 32) 3488

max\_pooling1d\_2 (MaxPooling (None, 16, 32) 0

1D)

conv1d\_3 (Conv1D) (None, 16, 64) 14400

max\_pooling1d\_3 (MaxPooling (None, 8, 64) 0

1D)

conv1d\_4 (Conv1D) (None, 8, 64) 20544

max\_pooling1d\_4 (MaxPooling (None, 4, 64) 0

1D)

conv1d\_5 (Conv1D) (None, 4, 64) 12352

max\_pooling1d\_5 (MaxPooling (None, 2, 64) 0

1D)

|  |  |  |  |
| --- | --- | --- | --- |
| conv1d\_6 (Conv1D) | (None, | 2, 72) | 13896 |
| max\_pooling1d\_6 (MaxPooling 1D) | (None, 1, 72) | | 0 |
| flatten (Flatten) | (None, 72) | | 0 |
| dense (Dense) | (None, 216) | | 15768 |
| dropout (Dropout) | (None, 216) | | 0 |
| dense\_1 (Dense) | (None, 5) | | 1085 |

=================================================================

Total params: 84,241

Trainable params: 84,241

Non-trainable params: 0

### Обучение модели

In [ ]:

*# Компиляция модели*

model**.**compile(optimizer**=**'nadam',

loss **=** 'sparse\_categorical\_crossentropy',

metrics **=** ['accuracy'])

*# Реализация раннего прекращения.*

checkpoint\_filepath **=** './checkpoint\_1D/'

model\_checkpoint\_cb **=** tf**.**keras**.**callbacks**.**ModelCheckpoint( filepath**=**checkpoint\_filepath,

save\_weights\_only**=True**, save\_best\_only**=True**)

early\_stopping\_cb **=** keras**.**callbacks**.**EarlyStopping( patience**=**10,

restore\_best\_weights**=True**)

*# Обучение.*

history **=** model**.**fit(X\_train, y\_train, batch\_size**=**32, epochs**=**30, validation\_split**=**0

*# Сохранение модели.*

model**.**save('1D.h5')

*# Откат к наилучшей модели.*

model**.**load\_weights(checkpoint\_filepath)

Epoch 1/30

626/626 [==============================] - 19s 11ms/step - loss: 1.1929 - accurac

y: 0.5162 - val\_loss: 1.5038 - val\_accuracy: 0.2571 Epoch 2/30

626/626 [==============================] - 6s 10ms/step - loss: 1.0655 - accuracy:

0.5673 - val\_loss: 1.4665 - val\_accuracy: 0.2921 Epoch 3/30

626/626 [==============================] - 6s 9ms/step - loss: 1.0096 - accuracy:

0.6101 - val\_loss: 1.6917 - val\_accuracy: 0.3253 Epoch 4/30

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 626/626 [==============================]  0.6326 - val\_loss: 1.4382 - val\_accuracy: Epoch 5/30  626/626 [==============================] | * 6s 10ms/step   0.3648   * 6s 10ms/step | * loss: * loss: | 0.9707  0.9263 | * accuracy: * accuracy: |
| 0.6476 - val\_loss: 1.3989 - val\_accuracy: Epoch 6/30  626/626 [==============================] | 0.3724  - 6s 10ms/step | - loss: | 0.8961 | - accuracy: |
| 0.6550 - val\_loss: 1.3796 - val\_accuracy: | 0.3868 |  |  |  |
| Epoch 7/30 |  |  |  |  |

626/626 [==============================] - 6s 9ms/step - loss: 0.8815 - accuracy:

0.6625 - val\_loss: 1.4598 - val\_accuracy: 0.3782 Epoch 8/30

626/626 [==============================] - 6s 10ms/step - loss: 0.8686 - accuracy:

0.6661 - val\_loss: 1.3629 - val\_accuracy: 0.3894 Epoch 9/30

626/626 [==============================] - 6s 9ms/step - loss: 0.8592 - accuracy:

0.6717 - val\_loss: 1.4430 - val\_accuracy: 0.3766 Epoch 10/30

626/626 [==============================] - 6s 10ms/step - loss: 0.8511 - accuracy:

0.6730 - val\_loss: 1.4274 - val\_accuracy: 0.3876 Epoch 11/30

626/626 [==============================] - 7s 11ms/step - loss: 0.8461 - accuracy:

0.6759 - val\_loss: 1.3894 - val\_accuracy: 0.3872 Epoch 12/30

626/626 [==============================] - 6s 10ms/step - loss: 0.8383 - accuracy:

0.6769 - val\_loss: 1.4198 - val\_accuracy: 0.3808 Epoch 13/30

626/626 [==============================] - 6s 9ms/step - loss: 0.8326 - accuracy:

0.6810 - val\_loss: 1.3771 - val\_accuracy: 0.3890 Epoch 14/30

626/626 [==============================] - 7s 11ms/step - loss: 0.8239 - accuracy:

0.6822 - val\_loss: 1.3857 - val\_accuracy: 0.3874 Epoch 15/30

626/626 [==============================] - 6s 10ms/step - loss: 0.8179 - accuracy:

0.6842 - val\_loss: 1.4132 - val\_accuracy: 0.3932 Epoch 16/30

626/626 [==============================] - 6s 9ms/step - loss: 0.8117 - accuracy:

0.6847 - val\_loss: 1.4073 - val\_accuracy: 0.3896 Epoch 17/30

626/626 [==============================] - 6s 9ms/step - loss: 0.8071 - accuracy:

0.6868 - val\_loss: 1.3997 - val\_accuracy: 0.3892 Epoch 18/30

626/626 [==============================] - 6s 10ms/step - loss: 0.8016 - accuracy:

0.6899 - val\_loss: 1.3596 - val\_accuracy: 0.3966 Epoch 19/30

626/626 [==============================] - 6s 10ms/step - loss: 0.7956 - accuracy:

0.6924 - val\_loss: 1.3722 - val\_accuracy: 0.3960 Epoch 20/30

626/626 [==============================] - 6s 9ms/step - loss: 0.7888 - accuracy:

0.6947 - val\_loss: 1.4650 - val\_accuracy: 0.3828 Epoch 21/30

626/626 [==============================] - 7s 12ms/step - loss: 0.7834 - accuracy:

0.6934 - val\_loss: 1.4539 - val\_accuracy: 0.3838 Epoch 22/30

Out[ ]:

In [ ]:

score **=** model**.**evaluate(X\_test, y\_test) score

626/626 [==============================] - 6s 9ms/step - loss: 0.7803 - accuracy:

0.6953 - val\_loss: 1.4166 - val\_accuracy: 0.3856 Epoch 23/30

626/626 [==============================] - 6s 10ms/step - loss: 0.7748 - accuracy:

0.6953 - val\_loss: 1.4169 - val\_accuracy: 0.3910 Epoch 24/30

626/626 [==============================] - 6s 10ms/step - loss: 0.7706 - accuracy:

0.6981 - val\_loss: 1.4582 - val\_accuracy: 0.3888 Epoch 25/30

626/626 [==============================] - 6s 9ms/step - loss: 0.7624 - accuracy:

0.7000 - val\_loss: 1.4915 - val\_accuracy: 0.3882 Epoch 26/30

626/626 [==============================] - 6s 9ms/step - loss: 0.7540 - accuracy:

0.7034 - val\_loss: 1.4678 - val\_accuracy: 0.3912 Epoch 27/30

626/626 [==============================] - 6s 10ms/step - loss: 0.7522 - accuracy:

0.7020 - val\_loss: 1.4476 - val\_accuracy: 0.3804 Epoch 28/30

626/626 [==============================] - 6s 9ms/step - loss: 0.7454 - accuracy:

0.7055 - val\_loss: 1.4566 - val\_accuracy: 0.3862

<tensorflow.python.training.tracking.util.CheckpointLoadStatus at 0x7f5eaa2fde90>

### Оценка работы модели

Out[ ]:

In [ ]:

acc **=** history**.**history['accuracy']

val\_acc **=** history**.**history['val\_accuracy'] loss **=** history**.**history['loss']

val\_loss **=** history**.**history['val\_loss']

epochs **=** range(1, len(acc) **+** 1)

plt**.**plot(epochs, acc, 'bo', label**=**'Training acc')

plt**.**plot(epochs, val\_acc, 'b', label**=**'Validation acc') plt**.**title('Training and validation accuracy')

plt**.**legend()

plt**.**figure()

plt**.**plot(epochs, loss, 'bo', label**=**'Training loss')

plt**.**plot(epochs, val\_loss, 'b', label**=**'Validation loss') plt**.**title('Training and validation loss')

plt**.**legend()

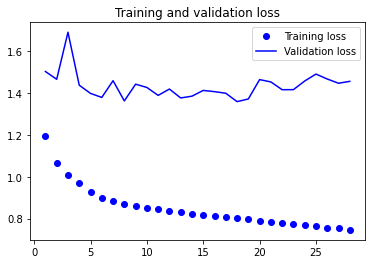
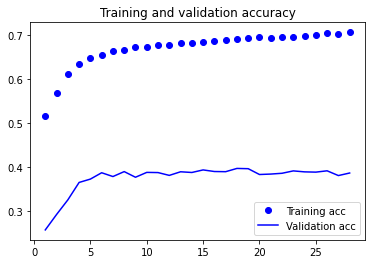
plt**.**show()

88/88 [==============================] - 0s 5ms/step - loss: 0.9877 - accuracy: 0.

6034

[0.987737774848938, 0.6033559441566467]

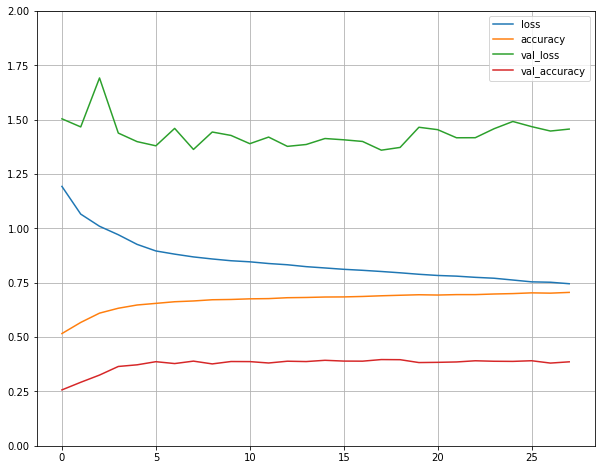
### Графики изменения точности и потерь модели по обучающим и проверочным данным в процессе обучения



In [ ]:

pd**.**DataFrame(history**.**history)**.**plot(figsize **=** (10, 8)) plt**.**grid()

plt**.**gca()**.**set\_ylim(0, 2) plt**.**show()



## Двумерная сверточная нейронная сеть

### Преобразование данных для работы с моделью

In [ ]:

X\_train **=** X\_trainCopyNorm

y\_train **=** y\_trainCopyEncoded X\_test **=** X\_testCopyNorm

y\_test **=** y\_testCopyEncoded

*# До преобразования.*

X\_train**.**shape, X\_test**.**shape

*# Преобразование формы тензора.*

X\_train **=** X\_train[**...**, np**.**newaxis] X\_test **=** X\_test[**...**, np**.**newaxis]

*# После преобразования.*

X\_train**.**shape, X\_test**.**shape

### Создание модели

In [ ]:

**from** functools **import** partial

DefaultConv2D **=** partial(keras**.**layers**.**Conv2D, kernel\_size **=** 3, activation **=** "relu", model **=** keras**.**models**.**Sequential([

DefaultConv2D(filters **=** 64, kernel\_size **=** 7, input\_shape **=** [1000, 12, 1]),

keras**.**layers**.**MaxPooling2D(pool\_size **=** 2),

DefaultConv2D(filters **=** 128),

DefaultConv2D(filters **=** 128),

keras**.**layers**.**MaxPooling2D(pool\_size **=** 2),

DefaultConv2D(filters **=** 256),

DefaultConv2D(filters **=** 256),

keras**.**layers**.**MaxPooling2D(pool\_size **=** 2),

keras**.**layers**.**Flatten(),

keras**.**layers**.**Dense(units**=**128, activation **=** "relu"), keras**.**layers**.**Dropout(0.5),

keras**.**layers**.**Dense(units**=**64, activation **=** "relu"),

keras**.**layers**.**Dropout(0.5),

keras**.**layers**.**Dense(units **=** 5, activation **=** "softmax"),

])

print(model**.**summary())

*# # Список слоев модели. # print(model.layers)*

Model: "sequential"

Layer (type) Output Shape Param #

=================================================================

|  |  |  |  |
| --- | --- | --- | --- |
| conv2d (Conv2D) | (None, | 1000, 12, 64) | 3200 |
| max\_pooling2d (MaxPooling2D | (None, 500, 6, 64) | | 0 |
| ) |  | |  |
| conv2d\_1 (Conv2D) | (None, 500, 6, 128) | | 73856 |
| conv2d\_2 (Conv2D) | (None, 500, 6, 128) | | 147584 |
| max\_pooling2d\_1 (MaxPooling 2D) | (None, 250, 3, 128) | | 0 |
| conv2d\_3 (Conv2D) | (None, 250, 3, 256) | | 295168 |
| conv2d\_4 (Conv2D) | (None, 250, 3, 256) | | 590080 |
| max\_pooling2d\_2 (MaxPooling 2D) | (None, 125, 1, 256) | | 0 |
| flatten (Flatten) | (None, 32000) | | 0 |
| dense (Dense) | (None, 128) | | 4096128 |
| dropout (Dropout) | (None, 128) | | 0 |
| dense\_1 (Dense) | (None, 64) | | 8256 |
| dropout\_1 (Dropout) | (None, 64) | | 0 |
| dense\_2 (Dense) | (None, 5) | | 325 |

=================================================================

Total params: 5,214,597

Trainable params: 5,214,597

Non-trainable params: 0

None

### Обучение модели

In [ ]:

*# Компиляция модели.*

model**.**compile(loss **=** 'sparse\_categorical\_crossentropy', optimizer **=** 'nadam',

metrics **=** ['accuracy'])

*# Реализация раннего прекращения.*

checkpoint\_filepath **=** './checkpoint\_2D/'

model\_checkpoint\_cb **=** tf**.**keras**.**callbacks**.**ModelCheckpoint(

filepath**=**checkpoint\_filepath, save\_weights\_only**=True**,

save\_best\_only**=True**)

early\_stopping\_cb **=** keras**.**callbacks**.**EarlyStopping( patience**=**10,

restore\_best\_weights**=True**)

*# Обучение.*

history **=** model**.**fit(X\_train, y\_train, epochs **=** 10, validation\_split **=** 0.2, callbac

*# Сохранение модели.*

model**.**save('2D.h5')

*# Откат к наилучшей модели.*

model**.**load\_weights(checkpoint\_filepath)

### Оценка работы модели

In [ ]:

score **=** model**.**evaluate(X\_test, y\_test) score

Out[ ]:

In [ ]:

acc **=** history**.**history['accuracy']

val\_acc **=** history**.**history['val\_accuracy'] loss **=** history**.**history['loss']

val\_loss **=** history**.**history['val\_loss']

epochs **=** range(1, len(acc) **+** 1)

plt**.**plot(epochs, acc, 'bo', label**=**'Training acc')

plt**.**plot(epochs, val\_acc, 'b', label**=**'Validation acc') plt**.**title('Training and validation accuracy')

plt**.**legend()

plt**.**figure()

plt**.**plot(epochs, loss, 'bo', label**=**'Training loss')

plt**.**plot(epochs, val\_loss, 'b', label**=**'Validation loss') plt**.**title('Training and validation loss')

plt**.**legend()

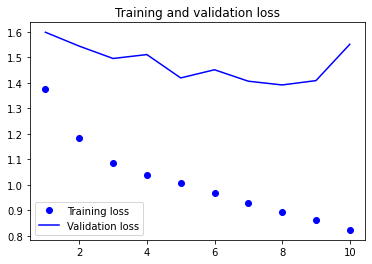
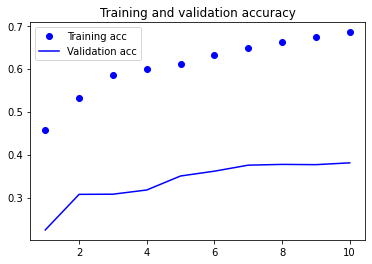
plt**.**show()

88/88 [==============================] - 3s 30ms/step - loss: 1.0279 - accuracy:

0.6012

[1.0278685092926025, 0.6012138724327087]

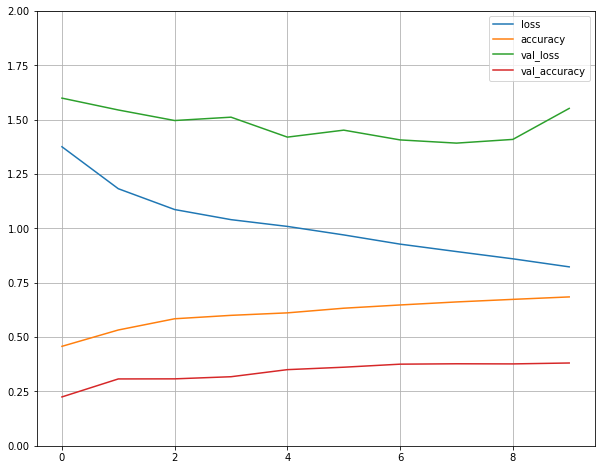
### Графики изменения точности и потерь модели по обучающим и проверочным данным в процессе обучения



In [ ]:

pd**.**DataFrame(history**.**history)**.**plot(figsize **=** (10, 8)) plt**.**grid()

plt**.**gca()**.**set\_ylim(0, 2) plt**.**show()



## ResNet50

### Преобразование данных для работы с моделью

In [ ]:

X\_train **=** X\_trainCopyNorm

y\_train **=** y\_trainCopyEncoded X\_test **=** X\_testCopyNorm

y\_test **=** y\_testCopyEncoded

*# До преобразования.*

X\_train**.**shape, X\_test**.**shape

*# Преобразование формы тензора.*

X\_train **=** np**.**reshape(X\_train, (25025, **-**1, 32, 3))

X\_test **=** np**.**reshape(X\_test, (2801, **-**1, 32, 3))

*# После преобразования.*

X\_train**.**shape, X\_test**.**shape

Out[ ]:

In [ ]:

conv\_base **=** keras**.**applications**.**ResNet50(weights**=**'imagenet',

include\_top **= False**,

input\_shape **=** (125, 32, 3)

)

model **=** models**.**Sequential() model**.**add(conv\_base)

model**.**add(layers**.**Flatten())

model**.**add(layers**.**Dense(256, activation**=**'relu')) model**.**add(layers**.**Dense(5, activation**=**'softmax')) model**.**summary()

((25025, 125, 32, 3), (2801, 125, 32, 3))

### Загрузка модели

Downloading data from https://storage.googleapis.com/tensorflow/keras-application s/resnet/resnet50\_weights\_tf\_dim\_ordering\_tf\_kernels\_notop.h5

94773248/94765736 [==============================] - 2s 0us/step

94781440/94765736 [==============================] - 2s 0us/step

Model: "sequential"

Layer (type) Output Shape Param #

=================================================================

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| resnet50 (Functional) | (None, | 4, 1, | 2048) | 23587712 |
| flatten (Flatten) | (None, | 8192) |  | 0 |
| dense (Dense) | (None, | 256) |  | 2097408 |
| dense\_1 (Dense) | (None, | 5) |  | 1285 |

=================================================================

Total params: 25,686,405

Trainable params: 25,633,285

Non-trainable params: 53,120

### Обучение модели

In [ ]:

*# Замораживание сверточной основы.*

conv\_base**.**trainable **= False**

*# Компиляция модели.*

model**.**compile(optimizer **=** tf**.**keras**.**optimizers**.**Nadam(), loss**=**'sparse\_categorical\_crossentropy', metrics**=**['accuracy'])

*# Обучение.*

history **=** model**.**fit(X\_train, y\_train, epochs **=** 5, validation\_split **=** 0.2)

Epoch 1/5

626/626 [==============================] - 30s 28ms/step - loss: 1.5195 - accurac

y: 0.3940 - val\_loss: 1.9083 - val\_accuracy: 0.0314 Epoch 2/5

626/626 [==============================] - 15s 25ms/step - loss: 1.4612 - accurac

y: 0.4199 - val\_loss: 1.9371 - val\_accuracy: 0.0314 Epoch 3/5

626/626 [==============================] - 16s 26ms/step - loss: 1.4591 - accurac

y: 0.4199 - val\_loss: 1.8357 - val\_accuracy: 0.0314 Epoch 4/5

626/626 [==============================] - 16s 25ms/step - loss: 1.4584 - accurac

y: 0.4199 - val\_loss: 1.8656 - val\_accuracy: 0.0314 Epoch 5/5

626/626 [==============================] - 15s 25ms/step - loss: 1.4581 - accurac

y: 0.4199 - val\_loss: 1.9802 - val\_accuracy: 0.0314

In [ ]:

*# Разморозка сверточной основы.*

conv\_base**.**trainable **= True**

*# Компиляция модели.*

model**.**compile(optimizer **=** tf**.**keras**.**optimizers**.**Nadam(), loss**=**'sparse\_categorical\_crossentropy', metrics**=**['accuracy'])

*# Реализация раннего прекращения.*

checkpoint\_filepath **=** './checkpoint\_ResNet50/'

model\_checkpoint\_cb **=** tf**.**keras**.**callbacks**.**ModelCheckpoint( filepath**=**checkpoint\_filepath,

save\_weights\_only**=True**, save\_best\_only**=True**)

early\_stopping\_cb **=** keras**.**callbacks**.**EarlyStopping( patience**=**10,

restore\_best\_weights**=True**)

*# Обучение.*

history **=** model**.**fit(X\_train, y\_train, epochs **=** 30, validation\_split **=** 0.2, callbac

*# Сохранение модели.*

model**.**save("ResNet50.h5")

*# Откат к наилучшей модели.*

model**.**load\_weights(checkpoint\_filepath)

|  |  |  |  |
| --- | --- | --- | --- |
| Epoch 1/30  626/626 [==============================] - 75s 89ms/step | - loss: | 1.1972 | - accurac |
| y: 0.5618 - val\_loss: 1.8524 - val\_accuracy: 0.2492 Epoch 2/30  626/626 [==============================] - 54s 86ms/step | - loss: | 1.0176 | - accurac |
| y: 0.6128 - val\_loss: 1.9863 - val\_accuracy: 0.2797 |  |  |  |
| Epoch 3/30  626/626 [==============================] - 54s 87ms/step | - loss: | 0.9905 | - accurac |
| y: 0.6180 - val\_loss: 1.5982 - val\_accuracy: 0.1790 Epoch 4/30 |  |  |  |
| 626/626 [==============================] - 57s 92ms/step  y: 0.6235 - val\_loss: 1.4700 - val\_accuracy: 0.3606 Epoch 5/30  626/626 [==============================] - 53s 85ms/step | * loss: * loss: | 0.9939  0.9398 | * accurac * accurac |
| y: 0.6319 - val\_loss: 1.6144 - val\_accuracy: 0.3457 Epoch 6/30  626/626 [==============================] - 53s 85ms/step | - loss: | 0.9167 | - accurac |
| y: 0.6413 - val\_loss: 1.6112 - val\_accuracy: 0.3415 |  |  |  |
| Epoch 7/30  626/626 [==============================] - 54s 86ms/step | - loss: | 0.9106 | - accurac |
| y: 0.6484 - val\_loss: 1.5525 - val\_accuracy: 0.2266 Epoch 8/30 |  |  |  |
| 626/626 [==============================] - 53s 85ms/step  y: 0.6515 - val\_loss: 1.5964 - val\_accuracy: 0.3201 Epoch 9/30  626/626 [==============================] - 55s 88ms/step | * loss: * loss: | 0.8971  0.8837 | * accurac * accurac |
| y: 0.6553 - val\_loss: 1.4036 - val\_accuracy: 0.3750 Epoch 10/30  626/626 [==============================] - 53s 85ms/step | - loss: | 0.8648 | - accurac |
| y: 0.6646 - val\_loss: 1.4766 - val\_accuracy: 0.3774 |  |  |  |
| Epoch 11/30  626/626 [==============================] - 52s 83ms/step | - loss: | 0.8733 | - accurac |
| y: 0.6617 - val\_loss: 1.5979 - val\_accuracy: 0.3722 Epoch 12/30 |  |  |  |
| 626/626 [==============================] - 53s 84ms/step  y: 0.6671 - val\_loss: 1.6375 - val\_accuracy: 0.3489 Epoch 13/30  626/626 [==============================] - 52s 84ms/step | * loss: * loss: | 0.8471  0.8399 | * accurac * accurac |
| y: 0.6674 - val\_loss: 2.1129 - val\_accuracy: 0.3341 Epoch 14/30  626/626 [==============================] - 52s 83ms/step | - loss: | 0.8287 | - accurac |
| y: 0.6667 - val\_loss: 1.7548 - val\_accuracy: 0.3594 |  |  |  |
| Epoch 15/30  626/626 [==============================] - 52s 83ms/step | - loss: | 0.8756 | - accurac |
| y: 0.6545 - val\_loss: 1.4578 - val\_accuracy: 0.3475 Epoch 16/30 |  |  |  |
| 626/626 [==============================] - 52s 83ms/step  y: 0.6605 - val\_loss: 1.5944 - val\_accuracy: 0.3321 Epoch 17/30  626/626 [==============================] - 55s 87ms/step | * loss: * loss: | 0.8612  0.8884 | * accurac * accurac |
| y: 0.6484 - val\_loss: 1.4602 - val\_accuracy: 0.3658 Epoch 18/30  626/626 [==============================] - 55s 87ms/step | - loss: | 0.8723 | - accurac |
| y: 0.6589 - val\_loss: 1.4576 - val\_accuracy: 0.3702 |  |  |  |
| Epoch 19/30  626/626 [==============================] - 52s 84ms/step | - loss: | 0.8662 | - accurac |
| y: 0.6630 - val\_loss: 1.4177 - val\_accuracy: 0.3854 |  |  |  |

Out[ ]:

In [ ]:

score **=** model**.**evaluate(X\_test, y\_test) score

<tensorflow.python.training.tracking.util.CheckpointLoadStatus at 0x7f558b1c21d0>

### Оценка работы модели

Out[ ]:

In [ ]:

acc **=** history**.**history['accuracy']

val\_acc **=** history**.**history['val\_accuracy'] loss **=** history**.**history['loss']

val\_loss **=** history**.**history['val\_loss']

epochs **=** range(1, len(acc) **+** 1)

plt**.**plot(epochs, acc, 'bo', label**=**'Training acc')

plt**.**plot(epochs, val\_acc, 'b', label**=**'Validation acc') plt**.**title('Training and validation accuracy')

plt**.**legend()

plt**.**figure()

plt**.**plot(epochs, loss, 'bo', label**=**'Training loss')

plt**.**plot(epochs, val\_loss, 'b', label**=**'Validation loss') plt**.**title('Training and validation loss')

plt**.**legend()

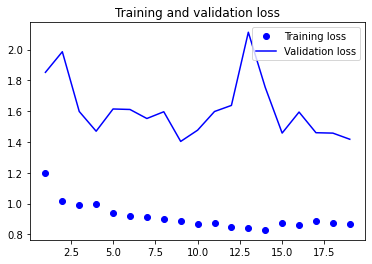
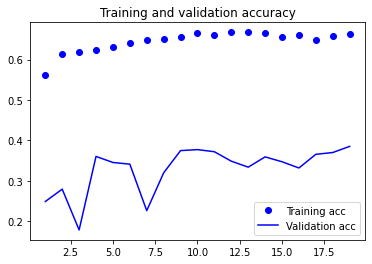
plt**.**show()

88/88 [==============================] - 2s 24ms/step - loss: 1.1361 - accuracy:

0.5484

[1.1360713243484497, 0.5483756065368652]

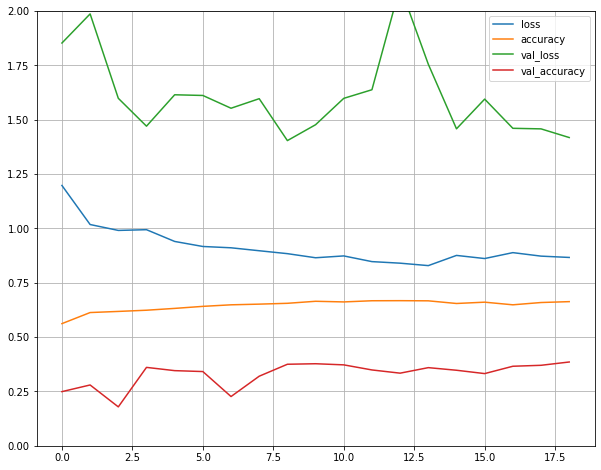
### Графики изменения точности и потерь модели по обучающим и проверочным данным в процессе обучения



In [ ]:

pd**.**DataFrame(history**.**history)**.**plot(figsize **=** (10, 8)) plt**.**grid()

plt**.**gca()**.**set\_ylim(0, 2) plt**.**show()



## ResNet50V2

### Преобразование данных для работы с моделью

In [ ]:

X\_train **=** X\_trainCopyNorm

y\_train **=** y\_trainCopyEncoded X\_test **=** X\_testCopyNorm

y\_test **=** y\_testCopyEncoded

*# До преобразования.*

X\_train**.**shape, X\_test**.**shape

*# Преобразование формы тензора.*

X\_train **=** np**.**reshape(X\_train, (25025, **-**1, 32, 3))

X\_test **=** np**.**reshape(X\_test, (2801, **-**1, 32, 3))

*# После преобразования.*

X\_train**.**shape, X\_test**.**shape

### Загрузка модели

In [ ]:

conv\_base **=** keras**.**applications**.**ResNet50V2(weights**=**'imagenet',

include\_top **= False**,

input\_shape **=** (125, 32, 3)

)

model **=** models**.**Sequential() model**.**add(conv\_base)

model**.**add(layers**.**Flatten())

model**.**add(layers**.**Dense(256, activation**=**'relu')) model**.**add(layers**.**Dense(5, activation**=**'softmax')) model**.**summary()

Downloading data from https://storage.googleapis.com/tensorflow/keras-application s/resnet/resnet50v2\_weights\_tf\_dim\_ordering\_tf\_kernels\_notop.h5

94674944/94668760 [==============================] - 2s 0us/step

94683136/94668760 [==============================] - 2s 0us/step

Model: "sequential"

Layer (type) Output Shape Param #

=================================================================

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| resnet50v2 (Functional) | (None, | 4, 1, | 2048) | 23564800 |
| flatten (Flatten) | (None, | 8192) |  | 0 |
| dense (Dense) | (None, | 256) |  | 2097408 |
| dense\_1 (Dense) | (None, | 5) |  | 1285 |

=================================================================

Total params: 25,663,493

Trainable params: 25,618,053

Non-trainable params: 45,440

### Обучение модели

In [ ]:

*# Замораживание сверточной основы.*

conv\_base**.**trainable **= False**

*# Компиляция модели.*

model**.**compile(optimizer **=** tf**.**keras**.**optimizers**.**Nadam(), loss**=**'sparse\_categorical\_crossentropy', metrics**=**['accuracy'])

*# Обучение.*

history **=** model**.**fit(X\_train, y\_train, epochs **=** 5, validation\_split **=** 0.2)

Epoch 1/5

626/626 [==============================] - 29s 26ms/step - loss: 1.2964 - accurac

y: 0.4819 - val\_loss: 1.7045 - val\_accuracy: 0.2511 Epoch 2/5

626/626 [==============================] - 14s 23ms/step - loss: 1.2252 - accurac

y: 0.5113 - val\_loss: 1.7106 - val\_accuracy: 0.2238 Epoch 3/5

626/626 [==============================] - 14s 22ms/step - loss: 1.1911 - accurac

y: 0.5249 - val\_loss: 1.5909 - val\_accuracy: 0.2957 Epoch 4/5

626/626 [==============================] - 15s 24ms/step - loss: 1.1656 - accurac

y: 0.5371 - val\_loss: 1.7434 - val\_accuracy: 0.2396 Epoch 5/5

626/626 [==============================] - 14s 23ms/step - loss: 1.1400 - accurac

y: 0.5448 - val\_loss: 1.6276 - val\_accuracy: 0.2903

In [ ]:

*# Разморозка сверточной основы.*

conv\_base**.**trainable **= True**

*# Компиляция модели.*

model**.**compile(optimizer **=** tf**.**keras**.**optimizers**.**Nadam(), loss**=**'sparse\_categorical\_crossentropy', metrics**=**['accuracy'])

*# Реализация раннего прекращения.*

checkpoint\_filepath **=** './checkpoint\_ResNet50V2/'

model\_checkpoint\_cb **=** tf**.**keras**.**callbacks**.**ModelCheckpoint( filepath**=**checkpoint\_filepath,

save\_weights\_only**=True**,

save\_best\_only**=True**)

early\_stopping\_cb **=** keras**.**callbacks**.**EarlyStopping( patience**=**10,

restore\_best\_weights**=True**)

*# Обучение.*

history **=** model**.**fit(X\_train, y\_train, epochs **=** 30, validation\_split **=** 0.2, callbac

*# Сохранение модели.*

model**.**save("ResNet50V2.h5")

*# Откат к наилучшей модели.*

model**.**load\_weights(checkpoint\_filepath)

|  |  |  |  |
| --- | --- | --- | --- |
| Epoch 1/30  626/626 [==============================] - 71s 84ms/step | - loss: | 1.3228 | - accurac |
| y: 0.5051 - val\_loss: 1.5763 - val\_accuracy: 0.2603 Epoch 2/30  626/626 [==============================] - 52s 83ms/step | - loss: | 1.1102 | - accurac |
| y: 0.5709 - val\_loss: 1.4023 - val\_accuracy: 0.3355 |  |  |  |
| Epoch 3/30  626/626 [==============================] - 49s 79ms/step | - loss: | 1.0338 | - accurac |
| y: 0.6052 - val\_loss: 1.7224 - val\_accuracy: 0.2937 Epoch 4/30 |  |  |  |
| 626/626 [==============================] - 49s 78ms/step  y: 0.5547 - val\_loss: 1.4674 - val\_accuracy: 0.2945 Epoch 5/30  626/626 [==============================] - 49s 79ms/step | * loss: * loss: | 1.1499  1.0932 | * accurac * accurac |
| y: 0.5741 - val\_loss: 1.5473 - val\_accuracy: 0.2781 Epoch 6/30  626/626 [==============================] - 49s 79ms/step | - loss: | 1.0386 | - accurac |
| y: 0.5886 - val\_loss: 1.5171 - val\_accuracy: 0.3273 |  |  |  |
| Epoch 7/30  626/626 [==============================] - 51s 81ms/step | - loss: | 0.9769 | - accurac |
| y: 0.6161 - val\_loss: 1.3511 - val\_accuracy: 0.3710 Epoch 8/30 |  |  |  |
| 626/626 [==============================] - 49s 79ms/step  y: 0.6191 - val\_loss: 1.6220 - val\_accuracy: 0.2847 Epoch 9/30  626/626 [==============================] - 49s 79ms/step | * loss: * loss: | 0.9736  0.9702 | * accurac * accurac |
| y: 0.6184 - val\_loss: 1.4038 - val\_accuracy: 0.3652 Epoch 10/30  626/626 [==============================] - 49s 79ms/step | - loss: | 0.9551 | - accurac |
| y: 0.6290 - val\_loss: 1.5582 - val\_accuracy: 0.3500 |  |  |  |
| Epoch 11/30  626/626 [==============================] - 49s 79ms/step | - loss: | 0.9304 | - accurac |
| y: 0.6442 - val\_loss: 1.3544 - val\_accuracy: 0.3582 Epoch 12/30 |  |  |  |
| 626/626 [==============================] - 50s 79ms/step  y: 0.6465 - val\_loss: 1.4302 - val\_accuracy: 0.3718 Epoch 13/30  626/626 [==============================] - 49s 79ms/step | * loss: * loss: | 0.9118  0.8896 | * accurac * accurac |
| y: 0.6587 - val\_loss: 1.4249 - val\_accuracy: 0.3628 Epoch 14/30  626/626 [==============================] - 49s 79ms/step | - loss: | 0.8768 | - accurac |
| y: 0.6604 - val\_loss: 1.3904 - val\_accuracy: 0.3536 |  |  |  |
| Epoch 15/30  626/626 [==============================] - 49s 79ms/step | - loss: | 0.8627 | - accurac |
| y: 0.6664 - val\_loss: 1.8114 - val\_accuracy: 0.3153 Epoch 16/30 |  |  |  |
| 626/626 [==============================] - 49s 79ms/step  y: 0.6582 - val\_loss: 1.8094 - val\_accuracy: 0.3399 Epoch 17/30  626/626 [==============================] - 49s 79ms/step | * loss: * loss: | 0.9058  0.9006 | * accurac * accurac |
| y: 0.6484 - val\_loss: 1.5921 - val\_accuracy: 0.3530 |  |  |  |

Out[ ]:

In [ ]:

score **=** model**.**evaluate(X\_test, y\_test) score

<tensorflow.python.training.tracking.util.CheckpointLoadStatus at 0x7fd440499690>

### Оценка работы модели

Out[ ]:

88/88 [==============================] - 2s 24ms/step - loss: 1.0848 - accuracy:

0.5502

[1.0847526788711548, 0.5501606464385986]

### Графики изменения точности и потерь модели по обучающим и проверочным данным в процессе обучения

In [ ]:

acc **=** history**.**history['accuracy']

val\_acc **=** history**.**history['val\_accuracy'] loss **=** history**.**history['loss']

val\_loss **=** history**.**history['val\_loss']

epochs **=** range(1, len(acc) **+** 1)

plt**.**plot(epochs, acc, 'bo', label**=**'Training acc')

plt**.**plot(epochs, val\_acc, 'b', label**=**'Validation acc') plt**.**title('Training and validation accuracy')

plt**.**legend()

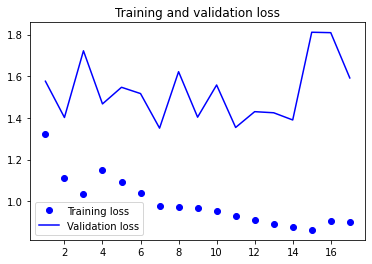
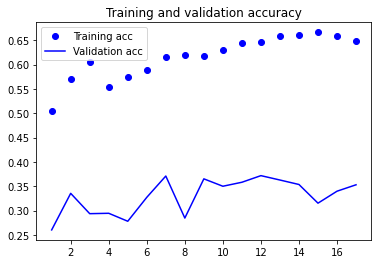
plt**.**figure()

plt**.**plot(epochs, loss, 'bo', label**=**'Training loss')

plt**.**plot(epochs, val\_loss, 'b', label**=**'Validation loss') plt**.**title('Training and validation loss')

plt**.**legend()

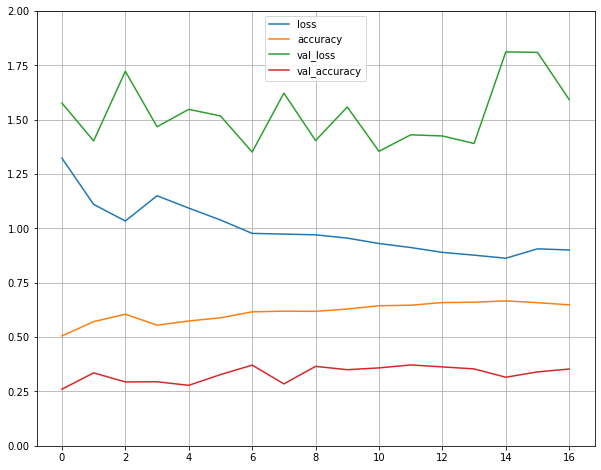
plt**.**show()



In [ ]:

pd**.**DataFrame(history**.**history)**.**plot(figsize **=** (10, 8)) plt**.**grid()

plt**.**gca()**.**set\_ylim(0, 2) plt**.**show()



## ResNet101

### Преобразование данных для работы с моделью

In [ ]:

X\_train **=** X\_trainCopyNorm

y\_train **=** y\_trainCopyEncoded X\_test **=** X\_testCopyNorm

y\_test **=** y\_testCopyEncoded

*# До преобразования.*

X\_train**.**shape, X\_test**.**shape

*# Преобразование формы тензора.*

X\_train **=** np**.**reshape(X\_train, (25025, **-**1, 32, 3))

X\_test **=** np**.**reshape(X\_test, (2801, **-**1, 32, 3))

*# После преобразования.*

X\_train**.**shape, X\_test**.**shape

Out[ ]:

In [ ]:

conv\_base **=** keras**.**applications**.**ResNet101(weights**=**'imagenet',

include\_top **= False**,

input\_shape **=** (125, 32, 3)

)

model **=** models**.**Sequential() model**.**add(conv\_base)

model**.**add(layers**.**Flatten())

model**.**add(layers**.**Dense(256, activation**=**'relu')) model**.**add(layers**.**Dense(5, activation**=**'softmax')) model**.**summary()

((25025, 125, 32, 3), (2801, 125, 32, 3))

### Загрузка модели

Downloading data from https://storage.googleapis.com/tensorflow/keras-application s/resnet/resnet101\_weights\_tf\_dim\_ordering\_tf\_kernels\_notop.h5

171450368/171446536 [==============================] - 3s 0us/step

171458560/171446536 [==============================] - 3s 0us/step

Model: "sequential"

Layer (type) Output Shape Param #

=================================================================

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| resnet101 (Functional) | (None, | 4, 1, | 2048) | 42658176 |
| flatten (Flatten) | (None, | 8192) |  | 0 |
| dense (Dense) | (None, | 256) |  | 2097408 |
| dense\_1 (Dense) | (None, | 5) |  | 1285 |

=================================================================

Total params: 44,756,869

Trainable params: 44,651,525

Non-trainable params: 105,344

### Обучение модели

In [ ]:

*# Замораживание сверточной основы.*

conv\_base**.**trainable **= False**

*# Компиляция модели.*

model**.**compile(optimizer **=** tf**.**keras**.**optimizers**.**Nadam(), loss**=**'sparse\_categorical\_crossentropy', metrics**=**['accuracy'])

*# Обучение.*

history **=** model**.**fit(X\_train, y\_train, epochs **=** 5, validation\_split **=** 0.2)

Epoch 1/5

626/626 [==============================] - 48s 53ms/step - loss: 1.4747 - accurac

y: 0.4145 - val\_loss: 1.8435 - val\_accuracy: 0.0314 Epoch 2/5

626/626 [==============================] - 27s 43ms/step - loss: 1.4552 - accurac

y: 0.4199 - val\_loss: 2.1549 - val\_accuracy: 0.0314 Epoch 3/5

626/626 [==============================] - 26s 41ms/step - loss: 1.4517 - accurac

y: 0.4199 - val\_loss: 1.9002 - val\_accuracy: 0.0314 Epoch 4/5

626/626 [==============================] - 26s 41ms/step - loss: 1.4463 - accurac

y: 0.4199 - val\_loss: 1.9268 - val\_accuracy: 0.0314 Epoch 5/5

626/626 [==============================] - 26s 42ms/step - loss: 1.4400 - accurac

y: 0.4205 - val\_loss: 1.7263 - val\_accuracy: 0.0338

In [ ]:

*# Разморозка сверточной основы.*

conv\_base**.**trainable **= True**

*# Компиляция модели.*

model**.**compile(optimizer **=** tf**.**keras**.**optimizers**.**Nadam(), loss**=**'sparse\_categorical\_crossentropy', metrics**=**['accuracy'])

*# Реализация раннего прекращения.*

checkpoint\_filepath **=** './checkpoint\_ResNet101/'

model\_checkpoint\_cb **=** tf**.**keras**.**callbacks**.**ModelCheckpoint( filepath**=**checkpoint\_filepath,

save\_weights\_only**=True**,

save\_best\_only**=True**)

early\_stopping\_cb **=** keras**.**callbacks**.**EarlyStopping( patience**=**10,

restore\_best\_weights**=True**)

*# Обучение.*

history **=** model**.**fit(X\_train, y\_train, epochs **=** 30, validation\_split **=** 0.2, callbac

*# Сохранение модели.*

model**.**save("ResNet101.h5")

*# Откат к наилучшей модели.*

model**.**load\_weights(checkpoint\_filepath)

Out[ ]:

In [ ]:

score **=** model**.**evaluate(X\_test, y\_test) score

Epoch 1/30

626/626 [==============================] - 134s 156ms/step - loss: 1.2938 - accura

cy: 0.5159 - val\_loss: 1.5702 - val\_accuracy: 0.2615 Epoch 2/30

626/626 [==============================] - 91s 145ms/step - loss: 1.1369 - accurac

y: 0.5499 - val\_loss: 2.1732 - val\_accuracy: 0.2551 Epoch 3/30

626/626 [==============================] - 95s 152ms/step - loss: 1.0303 - accurac

y: 0.6003 - val\_loss: 1.5549 - val\_accuracy: 0.3524 Epoch 4/30

626/626 [==============================] - 94s 150ms/step - loss: 1.1067 - accurac

y: 0.5684 - val\_loss: 1.4837 - val\_accuracy: 0.2919 Epoch 5/30

626/626 [==============================] - 91s 145ms/step - loss: 1.1025 - accurac

y: 0.5668 - val\_loss: 1.5630 - val\_accuracy: 0.1337 Epoch 6/30

626/626 [==============================] - 91s 145ms/step - loss: 1.0549 - accurac

y: 0.5840 - val\_loss: 1.5579 - val\_accuracy: 0.3189 Epoch 7/30

626/626 [==============================] - 91s 146ms/step - loss: 0.9959 - accurac

y: 0.6090 - val\_loss: 1.8601 - val\_accuracy: 0.3285 Epoch 8/30

626/626 [==============================] - 91s 145ms/step - loss: 0.9685 - accurac

y: 0.6182 - val\_loss: 1.5453 - val\_accuracy: 0.2939 Epoch 9/30

626/626 [==============================] - 92s 146ms/step - loss: 0.9477 - accurac

y: 0.6256 - val\_loss: 1.6233 - val\_accuracy: 0.3421 Epoch 10/30

626/626 [==============================] - 91s 146ms/step - loss: 0.9575 - accurac

y: 0.6301 - val\_loss: 2.2444 - val\_accuracy: 0.2971 Epoch 11/30

626/626 [==============================] - 91s 146ms/step - loss: 0.9895 - accurac

y: 0.6159 - val\_loss: 1.8510 - val\_accuracy: 0.2186 Epoch 12/30

626/626 [==============================] - 92s 147ms/step - loss: 0.9748 - accurac

y: 0.6191 - val\_loss: 1.5552 - val\_accuracy: 0.3608 Epoch 13/30

626/626 [==============================] - 91s 146ms/step - loss: 1.0288 - accurac

y: 0.6016 - val\_loss: 1.5427 - val\_accuracy: 0.3367 Epoch 14/30

626/626 [==============================] - 92s 147ms/step - loss: 0.9578 - accurac

y: 0.6274 - val\_loss: 1.5564 - val\_accuracy: 0.3510

<tensorflow.python.training.tracking.util.CheckpointLoadStatus at 0x7fe8b0ddce90>

### Оценка работы модели

Out[ ]:

In [ ]:

acc **=** history**.**history['accuracy']

val\_acc **=** history**.**history['val\_accuracy'] loss **=** history**.**history['loss']

val\_loss **=** history**.**history['val\_loss']

epochs **=** range(1, len(acc) **+** 1)

plt**.**plot(epochs, acc, 'bo', label**=**'Training acc')

plt**.**plot(epochs, val\_acc, 'b', label**=**'Validation acc') plt**.**title('Training and validation accuracy')

plt**.**legend()

plt**.**figure()

plt**.**plot(epochs, loss, 'bo', label**=**'Training loss')

plt**.**plot(epochs, val\_loss, 'b', label**=**'Validation loss') plt**.**title('Training and validation loss')

plt**.**legend()

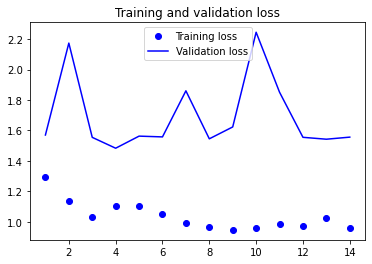
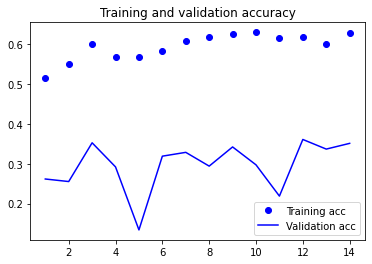
plt**.**show()

88/88 [==============================] - 3s 38ms/step - loss: 1.4233 - accuracy:

0.4223

[1.4233031272888184, 0.42234915494918823]

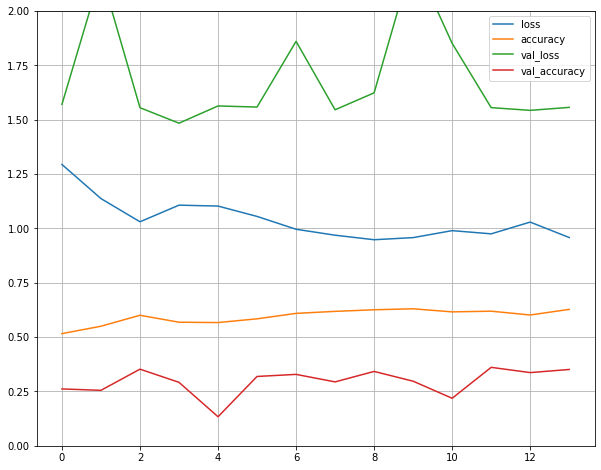
### Графики изменения точности и потерь модели по обучающим и проверочным данным в процессе обучения



In [ ]:

pd**.**DataFrame(history**.**history)**.**plot(figsize **=** (10, 8)) plt**.**grid()

plt**.**gca()**.**set\_ylim(0, 2) plt**.**show()



## VGG16

### Преобразование данных для работы с моделью

In [11]:

X\_train **=** X\_trainCopyNorm

y\_train **=** y\_trainCopyEncoded X\_test **=** X\_testCopyNorm

y\_test **=** y\_testCopyEncoded

*# До преобразования.*

X\_train**.**shape, X\_test**.**shape

*# Преобразование формы тензора.*

X\_train **=** np**.**reshape(X\_train, (25025, **-**1, 32, 3))

X\_test **=** np**.**reshape(X\_test, (2801, **-**1, 32, 3))

*# После преобразования.*

X\_train**.**shape, X\_test**.**shape

Out[11]:

In [12]:

conv\_base **=** keras**.**applications**.**vgg16**.**VGG16(weights**=**'imagenet',

include\_top **= False**,

input\_shape **=** (125, 32, 3)

)

model **=** models**.**Sequential() model**.**add(conv\_base)

model**.**add(layers**.**Flatten())

model**.**add(layers**.**Dense(256, activation**=**'relu'))

((25025, 125, 32, 3), (2801, 125, 32, 3))

### Загрузка модели

model**.**add(layers**.**Dense(5, activation**=**'softmax')) model**.**summary()

Model: "sequential"

Layer (type) Output Shape Param #

=================================================================

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| vgg16 (Functional) | (None, | 3, 1, | 512) | 14714688 |
| flatten (Flatten) | (None, | 1536) |  | 0 |
| dense (Dense) | (None, | 256) |  | 393472 |
| dense\_1 (Dense) | (None, | 5) |  | 1285 |

=================================================================

Total params: 15,109,445

Trainable params: 15,109,445

Non-trainable params: 0

### Обучение модели

In [13]:

*# Заморозка весов заранее обученных слоев.*

**for** layer **in** conv\_base**.**layers: layer**.**trainable **= False**

*# Компиляция модели.*

model**.**compile(optimizer **=** tf**.**keras**.**optimizers**.**Nadam(), loss**=**'sparse\_categorical\_crossentropy', metrics**=**['accuracy'])

*# Обучение.*

history **=** model**.**fit(X\_train, y\_train, epochs **=** 5, validation\_split **=** 0.2)

Epoch 1/5

626/626 [==============================] - 27s 25ms/step - loss: 1.4333 - accurac

y: 0.4279 - val\_loss: 1.8356 - val\_accuracy: 0.0655 Epoch 2/5

626/626 [==============================] - 14s 22ms/step - loss: 1.3976 - accurac

y: 0.4407 - val\_loss: 1.8282 - val\_accuracy: 0.0839 Epoch 3/5

626/626 [==============================] - 14s 22ms/step - loss: 1.3808 - accurac

y: 0.4469 - val\_loss: 1.6240 - val\_accuracy: 0.1678 Epoch 4/5

626/626 [==============================] - 14s 22ms/step - loss: 1.3655 - accurac

y: 0.4497 - val\_loss: 1.7195 - val\_accuracy: 0.1538 Epoch 5/5

626/626 [==============================] - 14s 22ms/step - loss: 1.3499 - accurac

y: 0.4560 - val\_loss: 1.8182 - val\_accuracy: 0.1469

In [14]:

*# Разморозка весов.*

**for** layer **in** conv\_base**.**layers: layer**.**trainable **= True**

*# Компиляция модели.*

model**.**compile(optimizer **=** tf**.**keras**.**optimizers**.**Nadam(), loss**=**'sparse\_categorical\_crossentropy', metrics**=**['accuracy'])

*# Реализация раннего прекращения.*

checkpoint\_filepath **=** './checkpoint\_vgg16/'

model\_checkpoint\_cb **=** tf**.**keras**.**callbacks**.**ModelCheckpoint( filepath**=**checkpoint\_filepath,

save\_weights\_only**=True**, save\_best\_only**=True**)

early\_stopping\_cb **=** keras**.**callbacks**.**EarlyStopping( patience**=**10,

restore\_best\_weights**=True**)

*# Обучение.*

history **=** model**.**fit(X\_train, y\_train, epochs **=** 30, validation\_split **=** 0.2, callbac

*# Сохранение модели.*

model**.**save("vgg16.h5")

*# Откат к наилучшей модели.*

model**.**load\_weights(checkpoint\_filepath)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Epoch 1/30  626/626 [==============================] - 47s 67ms/step | - | loss: | 23.1374 - accurac | |
| y: 0.4092 - val\_loss: 1.9009 - val\_accuracy: 0.0314  Epoch 2/30 |  |  |  | |
| 626/626 [==============================] - 40s 63ms/step  y: 0.4199 - val\_loss: 1.9069 - val\_accuracy: 0.0314 | - loss: | | 1.4589 | - accurac |
| Epoch 3/30  626/626 [==============================] - 40s 64ms/step | - loss: | | 1.4583 | - accurac |
| y: 0.4199 - val\_loss: 1.9212 - val\_accuracy: 0.0314 Epoch 4/30 |  | |  |  |
| 626/626 [==============================] - 40s 65ms/step  y: 0.4199 - val\_loss: 1.8956 - val\_accuracy: 0.0314 Epoch 5/30  626/626 [==============================] - 41s 65ms/step | * loss: * loss: | | 1.4578  1.4579 | * accurac * accurac |
| y: 0.4199 - val\_loss: 1.8860 - val\_accuracy: 0.0314 Epoch 6/30  626/626 [==============================] - 40s 65ms/step | - loss: | | 1.4951 | - accurac |
| y: 0.4158 - val\_loss: 1.8821 - val\_accuracy: 0.0314 |  | |  |  |
| Epoch 7/30  626/626 [==============================] - 40s 64ms/step | - loss: | | 1.4580 | - accurac |
| y: 0.4199 - val\_loss: 1.9352 - val\_accuracy: 0.0314 Epoch 8/30 |  | |  |  |
| 626/626 [==============================] - 40s 63ms/step  y: 0.4199 - val\_loss: 1.9299 - val\_accuracy: 0.0314 Epoch 9/30  626/626 [==============================] - 40s 63ms/step | * loss: * loss: | | 1.4573  1.4572 | * accurac * accurac |
| y: 0.4199 - val\_loss: 1.9266 - val\_accuracy: 0.0314 Epoch 10/30  626/626 [==============================] - 40s 64ms/step | - loss: | | 1.4570 | - accurac |
| y: 0.4199 - val\_loss: 1.8980 - val\_accuracy: 0.0314 |  | |  |  |
| Epoch 11/30  626/626 [==============================] - 40s 64ms/step | - loss: | | 1.4575 | - accurac |
| y: 0.4199 - val\_loss: 1.9423 - val\_accuracy: 0.0314 Epoch 12/30 |  | |  |  |
| 626/626 [==============================] - 40s 65ms/step  y: 0.4199 - val\_loss: 1.8710 - val\_accuracy: 0.0314 Epoch 13/30  626/626 [==============================] - 40s 63ms/step | * loss: * loss: | | 1.4571  1.4569 | * accurac * accurac |
| y: 0.4199 - val\_loss: 1.9401 - val\_accuracy: 0.0314 Epoch 14/30  626/626 [==============================] - 40s 63ms/step | - loss: | | 1.5373 | - accurac |
| y: 0.4161 - val\_loss: 1.8711 - val\_accuracy: 0.0314 |  | |  |  |
| Epoch 15/30  626/626 [==============================] - 41s 65ms/step | - loss: | | 1.4573 | - accurac |
| y: 0.4199 - val\_loss: 1.8701 - val\_accuracy: 0.0314 Epoch 16/30 |  | |  |  |
| 626/626 [==============================] - 40s 65ms/step  y: 0.4199 - val\_loss: 1.8592 - val\_accuracy: 0.0314 Epoch 17/30  626/626 [==============================] - 40s 63ms/step | * loss: * loss: | | 1.4566  1.4566 | * accurac * accurac |
| y: 0.4199 - val\_loss: 1.9045 - val\_accuracy: 0.0314 Epoch 18/30  626/626 [==============================] - 40s 64ms/step | - loss: | | 1.4562 | - accurac |
| y: 0.4199 - val\_loss: 1.9091 - val\_accuracy: 0.0314 |  | |  |  |
| Epoch 19/30  626/626 [==============================] - 40s 63ms/step | - loss: | | 1.4567 | - accurac |
| y: 0.4199 - val\_loss: 1.9138 - val\_accuracy: 0.0314 Epoch 20/30 |  | |  |  |
| 626/626 [==============================] - 40s 63ms/step  y: 0.4199 - val\_loss: 1.8908 - val\_accuracy: 0.0314 Epoch 21/30  626/626 [==============================] - 40s 63ms/step | * loss: * loss: | | 1.4565  1.4564 | * accurac * accurac |
| y: 0.4199 - val\_loss: 1.9098 - val\_accuracy: 0.0314  Epoch 22/30 |  | |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 626/626 [==============================] - 40s 64ms/step  y: 0.4199 - val\_loss: 1.8883 - val\_accuracy: 0.0314 | - loss: | 1.4566 | - accurac |
| Epoch 23/30  626/626 [==============================] - 40s 63ms/step | - loss: | 1.6918 | - accurac |
| y: 0.4183 - val\_loss: 1.9246 - val\_accuracy: 0.0314 Epoch 24/30 |  |  |  |
| 626/626 [==============================] - 40s 63ms/step  y: 0.4199 - val\_loss: 1.9335 - val\_accuracy: 0.0314 Epoch 25/30  626/626 [==============================] - 40s 63ms/step | * loss: * loss: | 1.4571  1.4566 | * accurac * accurac |
| y: 0.4199 - val\_loss: 1.8728 - val\_accuracy: 0.0314 Epoch 26/30  626/626 [==============================] - 40s 63ms/step | - loss: | 1.4567 | - accurac |
| y: 0.4199 - val\_loss: 1.9183 - val\_accuracy: 0.0314 |  |  |  |

Out[14]:

In [15]:

score **=** model**.**evaluate(X\_test, y\_test) score

<tensorflow.python.training.tracking.util.CheckpointLoadStatus at 0x7f1e6df9a9d0>

### Оценка работы модели

Out[15]:

In [16]:

acc **=** history**.**history['accuracy']

val\_acc **=** history**.**history['val\_accuracy'] loss **=** history**.**history['loss']

val\_loss **=** history**.**history['val\_loss']

epochs **=** range(1, len(acc) **+** 1)

plt**.**plot(epochs, acc, 'bo', label**=**'Training acc')

plt**.**plot(epochs, val\_acc, 'b', label**=**'Validation acc') plt**.**title('Training and validation accuracy')

plt**.**legend()

plt**.**figure()

plt**.**plot(epochs, loss, 'bo', label**=**'Training loss')

plt**.**plot(epochs, val\_loss, 'b', label**=**'Validation loss') plt**.**title('Training and validation loss')

plt**.**legend()

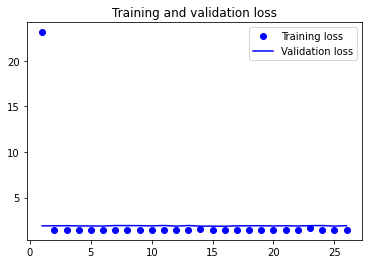
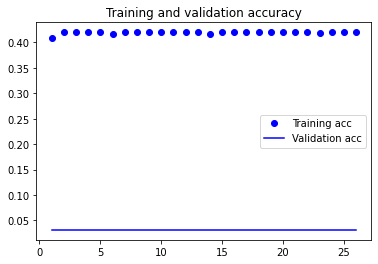
plt**.**show()

88/88 [==============================] - 2s 24ms/step - loss: 1.5352 - accuracy:

0.3442

[1.535159707069397, 0.34416279196739197]

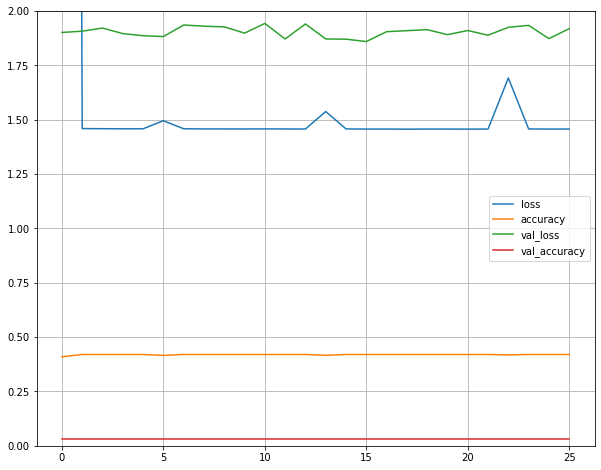
### Графики изменения точности и потерь модели по обучающим и проверочным данным в процессе обучения



In [17]:

pd**.**DataFrame(history**.**history)**.**plot(figsize **=** (10, 8)) plt**.**grid()

plt**.**gca()**.**set\_ylim(0, 2) plt**.**show()



## VGG19

### Преобразование данных для работы с моделью

In [ ]:

X\_train **=** X\_trainCopyNorm

y\_train **=** y\_trainCopyEncoded X\_test **=** X\_testCopyNorm

y\_test **=** y\_testCopyEncoded

*# До преобразования.*

X\_train**.**shape, X\_test**.**shape

*# Преобразование формы тензора.*

X\_train **=** np**.**reshape(X\_train, (25025, **-**1, 32, 3))

X\_test **=** np**.**reshape(X\_test, (2801, **-**1, 32, 3))

*# После преобразования.*

X\_train**.**shape, X\_test**.**shape

Out[ ]:

In [ ]:

conv\_base **=** keras**.**applications**.**vgg19**.**VGG19(weights**=**'imagenet',

include\_top **= False**,

input\_shape **=** (125, 32, 3)

)

model **=** models**.**Sequential() model**.**add(conv\_base)

model**.**add(layers**.**Flatten())

model**.**add(layers**.**Dense(256, activation**=**'relu')) model**.**add(layers**.**Dense(5, activation**=**'softmax')) model**.**summary()

((25025, 125, 32, 3), (2801, 125, 32, 3))

### Загрузка модели

Model: "sequential"

Layer (type) Output Shape Param #

=================================================================

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| vgg19 (Functional) | (None, | 3, 1, | 512) | 20024384 |
| flatten (Flatten) | (None, | 1536) |  | 0 |
| dense (Dense) | (None, | 256) |  | 393472 |
| dense\_1 (Dense) | (None, | 5) |  | 1285 |

=================================================================

Total params: 20,419,141

Trainable params: 20,419,141

Non-trainable params: 0

### Обучение модели

In [ ]:

*# Замораживание сверточной основы.*

conv\_base**.**trainable **= False**

*# Компиляция модели.*

model**.**compile(optimizer **=** tf**.**keras**.**optimizers**.**Nadam(), loss**=**'sparse\_categorical\_crossentropy', metrics**=**['accuracy'])

*# Обучение.*

history **=** model**.**fit(X\_train, y\_train, epochs **=** 5, validation\_split **=** 0.2)

Epoch 1/5

626/626 [==============================] - 28s 28ms/step - loss: 1.4324 - accurac

y: 0.4257 - val\_loss: 1.7885 - val\_accuracy: 0.0999 Epoch 2/5

626/626 [==============================] - 16s 26ms/step - loss: 1.3931 - accurac

y: 0.4437 - val\_loss: 1.8642 - val\_accuracy: 0.0991 Epoch 3/5

626/626 [==============================] - 19s 30ms/step - loss: 1.3770 - accurac

y: 0.4474 - val\_loss: 1.7624 - val\_accuracy: 0.1325 Epoch 4/5

626/626 [==============================] - 17s 27ms/step - loss: 1.3656 - accurac

y: 0.4565 - val\_loss: 1.7601 - val\_accuracy: 0.1327 Epoch 5/5

626/626 [==============================] - 16s 26ms/step - loss: 1.3592 - accurac

y: 0.4600 - val\_loss: 1.9278 - val\_accuracy: 0.1011

In [ ]:

*# Разморозка весов.*

conv\_base**.**trainable **= True**

*# Компиляция модели.*

model**.**compile(optimizer **=** tf**.**keras**.**optimizers**.**Nadam(), loss**=**'sparse\_categorical\_crossentropy', metrics**=**['accuracy'])

*# Реализация раннего прекращения.*

checkpoint\_filepath **=** './checkpoint\_vgg19/'

model\_checkpoint\_cb **=** tf**.**keras**.**callbacks**.**ModelCheckpoint( filepath**=**checkpoint\_filepath,

save\_weights\_only**=True**, save\_best\_only**=True**)

early\_stopping\_cb **=** keras**.**callbacks**.**EarlyStopping(

patience**=**10,

restore\_best\_weights**=True**)

*# Обучение.*

history **=** model**.**fit(X\_train, y\_train, epochs **=** 30, validation\_split **=** 0.2, callbac

*# Сохранение модели.*

model**.**save('vgg19.h5')

*# Откат к наилучшей модели.*

model**.**load\_weights(checkpoint\_filepath)

Epoch 1/30

626/626 [==============================] - 58s 84ms/step - loss: 38.9503 - accurac

y: 0.4085 - val\_loss: 1.9018 - val\_accuracy: 0.0314 Epoch 2/30

|  |  |  |  |
| --- | --- | --- | --- |
| 626/626 [==============================] - 49s 78ms/step  y: 0.4199 - val\_loss: 1.9127 - val\_accuracy: 0.0314 Epoch 3/30  626/626 [==============================] - 50s 80ms/step | * loss: * loss: | 1.4581  1.4576 | * accurac * accurac |
| y: 0.4199 - val\_loss: 1.8677 - val\_accuracy: 0.0314 Epoch 4/30  626/626 [==============================] - 50s 79ms/step | - loss: | 1.4570 | - accurac |
| y: 0.4199 - val\_loss: 1.8642 - val\_accuracy: 0.0314 |  |  |  |
| Epoch 5/30  626/626 [==============================] - 49s 78ms/step | - loss: | 1.4576 | - accurac |
| y: 0.4199 - val\_loss: 1.8991 - val\_accuracy: 0.0314  Epoch 6/30 |  |  |  |
| 626/626 [==============================] - 49s 78ms/step  y: 0.4199 - val\_loss: 1.8772 - val\_accuracy: 0.0314 Epoch 7/30  626/626 [==============================] - 49s 78ms/step | * loss: * loss: | 1.4570  1.4580 | * accurac * accurac |
| y: 0.4199 - val\_loss: 1.8882 - val\_accuracy: 0.0314 Epoch 8/30  626/626 [==============================] - 51s 81ms/step | - loss: | 1.4569 | - accurac |
| y: 0.4199 - val\_loss: 1.8945 - val\_accuracy: 0.0314 |  |  |  |
| Epoch 9/30  626/626 [==============================] - 51s 81ms/step | - loss: | 1.4571 | - accurac |
| y: 0.4199 - val\_loss: 1.9147 - val\_accuracy: 0.0314  Epoch 10/30 |  |  |  |
| 626/626 [==============================] - 49s 78ms/step  y: 0.4199 - val\_loss: 1.8776 - val\_accuracy: 0.0314 Epoch 11/30  626/626 [==============================] - 49s 78ms/step | * loss: * loss: | 1.4570  1.4572 | * accurac * accurac |
| y: 0.4199 - val\_loss: 1.8994 - val\_accuracy: 0.0314 Epoch 12/30  626/626 [==============================] - 49s 78ms/step | - loss: | 1.4570 | - accurac |
| y: 0.4199 - val\_loss: 1.9040 - val\_accuracy: 0.0314 |  |  |  |
| Epoch 13/30  626/626 [==============================] - 48s 77ms/step | - loss: | 1.4567 | - accurac |
| y: 0.4199 - val\_loss: 1.9433 - val\_accuracy: 0.0314  Epoch 14/30 |  |  |  |
| 626/626 [==============================] - 49s 78ms/step  y: 0.4199 - val\_loss: 1.9358 - val\_accuracy: 0.0314 | - loss: | 1.4567 | - accurac |

Out[ ]:

In [ ]:

<tensorflow.python.training.tracking.util.CheckpointLoadStatus at 0x7f61c7fc6550>

### Оценка работы модели

88/88 [==============================] - 2s 27ms/step - loss: 1.5358 - accuracy:



score **=** model**.**evaluate(X\_test, y\_test) score

0.3442

Out[ ]:

In [ ]:

acc **=** history**.**history['accuracy']

val\_acc **=** history**.**history['val\_accuracy'] loss **=** history**.**history['loss']

val\_loss **=** history**.**history['val\_loss']

epochs **=** range(1, len(acc) **+** 1)

plt**.**plot(epochs, acc, 'bo', label**=**'Training acc')

plt**.**plot(epochs, val\_acc, 'b', label**=**'Validation acc') plt**.**title('Training and validation accuracy')

plt**.**legend()

plt**.**figure()

plt**.**plot(epochs, loss, 'bo', label**=**'Training loss')

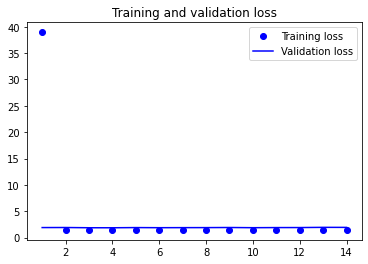
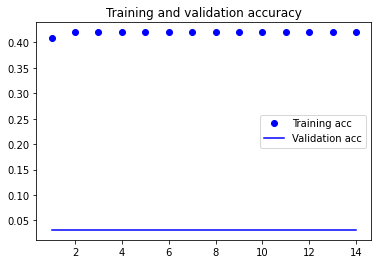
plt**.**plot(epochs, val\_loss, 'b', label**=**'Validation loss') plt**.**title('Training and validation loss')

plt**.**legend()

plt**.**show()

[1.5358237028121948, 0.34416279196739197]

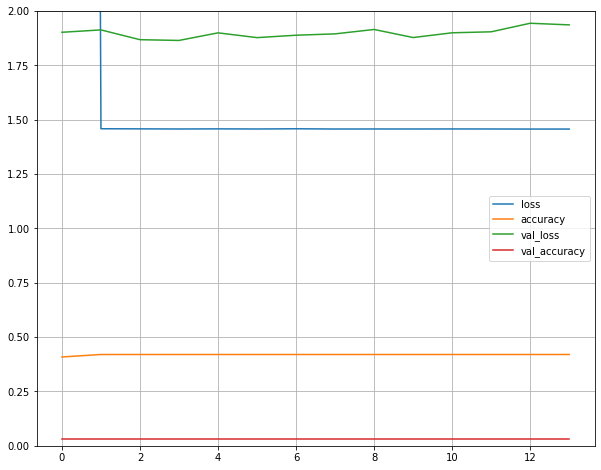
### Графики изменения точности и потерь модели по обучающим и проверочным данным в процессе обучения



In [ ]:

pd**.**DataFrame(history**.**history)**.**plot(figsize **=** (10, 8)) plt**.**grid()

plt**.**gca()**.**set\_ylim(0, 2) plt**.**show()



# work\_with\_ICBEB.docx

**# Подключение основных библиотек и загрузка данных**

**### Для Google Colaboratory**

```python

# Подключение Google Drive к виртуальной машине

from google.colab import drive

drive.mount('/content/drive')

# Копирование данных с Google Drive на локальный диск виртуальной машины.

*!*cp -r /content/drive/MyDrive/practice\_2022-2023/data/ICBEBnpy/ .

#!cp -r /content/drive/MyDrive/practice\_2022-2023/data/ptbxlnpy/ .

```

    Mounted at /content/drive

**### Подключение пакетов**

```python

# Для работы с данными

import pandas as pd

import numpy as np

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt   # plotting

import seaborn as sns   # plotting heatmap

# Для работы с моделями

import tensorflow as tf

from tensorflow import keras

from keras import layers

# Для метрик

from keras import backend as K

from keras.metrics import AUC, Recall, Precision, Accuracy, TruePositives, TrueNegatives, FalsePositives, FalseNegatives

from sklearn.metrics import fbeta\_score, precision\_score, recall\_score, accuracy\_score, roc\_auc\_score

from sklearn.metrics import auc, roc\_curve

# Функции

# Загрузка ICBEB

def load\_ICBEB(task):

  if task == 'diag':

    X\_train = np.load('ICBEBnpy/X\_train\_ICBEB\_diag.npy')

    y\_train = np.load('ICBEBnpy/y\_train\_ICBEB\_diag.npy')

    X\_test = np.load('ICBEBnpy/X\_val\_ICBEB\_diag.npy')

    y\_test = np.load('ICBEBnpy/y\_val\_ICBEB\_diag.npy')

  elif task == 'superdiag':

    X\_train = np.load('ICBEBnpy/X\_train\_ICBEB\_superdiag.npy')

    y\_train = np.load('ICBEBnpy/y\_train\_ICBEB\_superdiag.npy')

    X\_test = np.load('ICBEBnpy/X\_val\_ICBEB\_superdiag.npy')

    y\_test = np.load('ICBEBnpy/y\_val\_ICBEB\_superdiag.npy')

  elif task == 'subdiag':

    X\_train = np.load('ICBEBnpy/X\_train\_ICBEB\_subdiag.npy')

    y\_train = np.load('ICBEBnpy/y\_train\_ICBEB\_subdiag.npy')

    X\_test = np.load('ICBEBnpy/X\_val\_ICBEB\_subdiag.npy')

    y\_test = np.load('ICBEBnpy/y\_val\_ICBEB\_subdiag.npy')

  elif task == 'rhythm':

    X\_train = np.load('ICBEBnpy/X\_train\_ICBEB\_rhythm.npy')

    y\_train = np.load('ICBEBnpy/y\_train\_ICBEB\_rhythm.npy')

    X\_test = np.load('ICBEBnpy/X\_val\_ICBEB\_rhythm.npy')

    y\_test = np.load('ICBEBnpy/y\_val\_ICBEB\_rhythm.npy')

  elif task == 'form':

    X\_train = np.load('ICBEBnpy/X\_train\_ICBEB\_form.npy')

    y\_train = np.load('ICBEBnpy/y\_train\_ICBEB\_form.npy')

    X\_test = np.load('ICBEBnpy/X\_val\_ICBEB\_form.npy')

    y\_test = np.load('ICBEBnpy/y\_val\_ICBEB\_form.npy')

  #print(X\_train.shape, y\_train.shape, X\_test.shape, y\_test.shape)

  return X\_train, y\_train, X\_test, y\_test

# Загрузка ptbxl

def load\_ptbxl(task):

  if task == 'diag':

    X\_train = np.load('ptbxlnpy/X\_train\_ptbxl\_diag.npy')

    y\_train = np.load('ptbxlnpy/y\_train\_ptbxl\_diag.npy')

    X\_test = np.load('ptbxlnpy/X\_val\_ptbxl\_diag.npy')

    y\_test = np.load('ptbxlnpy/y\_val\_ptbxl\_diag.npy')

  elif task == 'superdiag':

    X\_train = np.load('ptbxlnpy/X\_train\_ptbxl\_superdiag.npy')

    y\_train = np.load('ptbxlnpy/y\_train\_ptbxl\_superdiag.npy')

    X\_test = np.load('ptbxlnpy/X\_val\_ptbxl\_superdiag.npy')

    y\_test = np.load('ptbxlnpy/y\_val\_ptbxl\_superdiag.npy')

  elif task == 'subdiag':

    X\_train = np.load('ptbxlnpy/X\_train\_ptbxl\_subdiag.npy')

    y\_train = np.load('ptbxlnpy/y\_train\_ptbxl\_subdiag.npy')

    X\_test = np.load('ptbxlnpy/X\_val\_ptbxl\_subdiag.npy')

    y\_test = np.load('ptbxlnpy/y\_val\_ptbxl\_subdiag.npy')

  elif task == 'rhythm':

    X\_train = np.load('ptbxlnpy/X\_train\_ptbxl\_rhythm.npy')

    y\_train = np.load('ptbxlnpy/y\_train\_ptbxl\_rhythm.npy')

    X\_test = np.load('ptbxlnpy/X\_val\_ptbxl\_rhythm.npy')

    y\_test = np.load('ptbxlnpy/y\_val\_ptbxl\_rhythm.npy')

  elif task == 'form':

    X\_train = np.load('ptbxlnpy/X\_train\_ptbxl\_form.npy')

    y\_train = np.load('ptbxlnpy/y\_train\_ptbxl\_form.npy')

    X\_test = np.load('ptbxlnpy/X\_val\_ptbxl\_form.npy')

    y\_test = np.load('ptbxlnpy/y\_val\_ptbxl\_form.npy')

  #print(X\_train.shape, y\_train.shape, X\_test.shape, y\_test.shape)

  return X\_train, y\_train, X\_test, y\_test

# Компиляция и обучение модели

def AUC\_Keras(y\_true, y\_pred):

    auc = keras.metrics.AUC(y\_true, y\_pred)[1]

    K.get\_session().run(tf.local\_variables\_initializer())

    return auc

# Компиляция и обучение модели

def compile\_fit(model, X\_train, y\_train, X\_val = None, y\_val = None, validation\_split = 0.0, early\_stopping = None, model\_checkpoint = None):

  model.compile(loss = keras.losses.CategoricalCrossentropy(),

                optimizer=tf.optimizers.Adam(),

                metrics=['AUC'])

  if X\_val == None:

    history = model.fit(X\_train, y\_train,

                        epochs = 30,

                        validation\_data = None,

                        validation\_split=validation\_split,

                        callbacks=[model\_checkpoint, early\_stopping])

  else:

    history = model.fit(X\_train, y\_train,

                        epochs = 30,

                        validation\_data = (X\_val, y\_val),

                        validation\_split=0.0,

                        callbacks=[model\_checkpoint, early\_stopping])

  return history

# TP TN FP FN

def tp\_tn\_fp\_fn(y\_true, y\_pred):

  TP = TruePositives()

  TN = TrueNegatives()

  FP = FalsePositives()

  FN = FalseNegatives()

  TP.update\_state(y\_true, y\_pred)

  TN.update\_state(y\_true, y\_pred)

  FP.update\_state(y\_true, y\_pred)

  FN.update\_state(y\_true, y\_pred)

  return TP.result().numpy(),  TN.result().numpy(),  FP.result().numpy(), FN.result().numpy()

# Подсчет метрик

def calc\_metrics(t, p, flag = 0): # t - y\_true, p - y\_pred

  y\_true=np.argmax(t, axis=1)

  y\_pred=np.argmax(p, axis=1)

  beta = 2

  f2\_score = fbeta\_score(y\_true, y\_pred, average='macro', beta=2)

  precision = precision\_score(y\_true, y\_pred, average='macro')

  recall = recall\_score(y\_true, y\_pred, average='macro')

  TP, TN, FP, FN = tp\_tn\_fp\_fn(t, p)

  g2\_score = TP/(TP+FP+beta\*FN)

  if flag == 0:

    return f2\_score, g2\_score

  elif flag == 1:

    return f2\_score, g2\_score, precision, recall

  #return f2\_score, g2\_score, AUC\_sklearn

# Таблица результатов

table\_res\_ICBEB = pd.DataFrame(columns = ('AUC', 'F2', 'G2'))

# Занесение новых результатов в таблицу

def edit\_table(table, model, X, y, index\_name): # X - X\_test, y - y\_test

  score = model.evaluate(X, y)

  y\_pr = model.predict(X) # y\_pr - y\_test\_pred

  f2\_score, g2\_score = calc\_metrics(y, y\_pr, flag = 0)

  list\_metrics = [f2\_score, g2\_score, score[1]]

  table.loc[index\_name] = list\_metrics

  return table

# График loss и accuracy

def plot\_loss\_and\_accuracy\_curves(\_history):

  fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(18,6))

  axs[0].plot(\_history.history['loss'], color='b', label='Training loss')

  axs[0].plot(\_history.history['val\_loss'], color='r', label='Validation loss')

  axs[0].set\_title("Loss curves")

  axs[0].legend(loc='best', shadow=True)

  axs[1].plot(\_history.history['auc'], color='b', label='Training accuracy')

  axs[1].plot(\_history.history['val\_auc'], color='r', label='Validation accuracy')

  axs[1].set\_title("Accuracy curves")

  axs[1].legend(loc='best', shadow=True)

  plt.show()

# Работа с моделями lstm и lstm\_bidir

def type\_comp\_fit\_save\_model\_score(table, X\_train, y\_train, X\_test, y\_test, type\_model, save\_name, index\_model\_task):

  # Уточняю количество классов

  num\_classes = y\_train.shape[1]

  # Выбор архитектуры модели

  if type\_model == 'lstm':

    model = keras.Sequential()

    model.add(layers.LSTM(input\_shape=(1000, 12), units=256,

                   return\_sequences=True,

                   stateful=False, unroll=False

    ))

    model.add(layers.LeakyReLU())

    model.add(layers.LSTM(units=256,

                   return\_sequences=False,

                   stateful=False, unroll=False

    ))

    model.add(layers.LeakyReLU())

    model.add(layers.Dense(units=num\_classes, activation='softmax'))

    print(model.summary())

    # Реализация раннего прекращения.

    checkpoint\_filepath = './checkpoint\_lstm/'

    model\_checkpoint = tf.keras.callbacks.ModelCheckpoint(filepath=checkpoint\_filepath,

                                                          save\_weights\_only=True,

                                                          save\_best\_only=True)

    early\_stopping = keras.callbacks.EarlyStopping(patience=15,

                                                  restore\_best\_weights=True)

  elif type\_model == 'lstm\_bidir':

    model = keras.Sequential()

    model.add(layers.Bidirectional(layers.LSTM(input\_shape=(1000, 12), units=256,

                         return\_sequences=True,

                         stateful=False, unroll=False

                         )))

    model.add(layers.LeakyReLU())

    model.add(layers.Bidirectional(layers.LSTM(units=256,

                         return\_sequences=False,

                         stateful=False, unroll=False

                         )))

    model.add(layers.LeakyReLU())

    model.add(layers.Dense(units=num\_classes, activation='softmax'))

    model.build(input\_shape = (None, 1000, 12)) # `input\_shape` is the shape of the input data

    print(model.summary())

    # Реализация раннего прекращения.

    checkpoint\_filepath = './checkpoint\_lstm\_bidir/'

    model\_checkpoint = tf.keras.callbacks.ModelCheckpoint(filepath=checkpoint\_filepath,

                                                          save\_weights\_only=True,

                                                          save\_best\_only=True)

    early\_stopping = keras.callbacks.EarlyStopping(patience=15,

                                                  restore\_best\_weights=True)

  # Обучение

  History = compile\_fit(model, X\_train, y\_train, validation\_split=0.1 ,early\_stopping=early\_stopping, model\_checkpoint=model\_checkpoint)

  # Сохранение модели

  model.save\_weights(save\_name)

  # Построение графика

  plot\_loss\_and\_accuracy\_curves(History)

  # Сохранение в таблицу

  table = edit\_table(table, model, X\_test, y\_test, index\_model\_task)

  return table

tf.random.set\_seed(42)

%matplotlib inline

```

**# Работа с lstm и lstm\_bidir**

**### lstm**

```python

X\_train, y\_train, X\_test, y\_test = load\_ICBEB(task = 'diag')

table\_res\_ICBEB = type\_comp\_fit\_save\_model\_score(table\_res\_ICBEB, X\_train, y\_train, X\_test, y\_test, 'lstm', 'lstm\_diag.h5', 'lstm\_diag')

del(X\_train)

del(y\_train)

del(X\_test)

del(y\_test)

```

```python

X\_train, y\_train, X\_test, y\_test = load\_ICBEB(task = 'superdiag')

table\_res\_ICBEB = type\_comp\_fit\_save\_model\_score(table\_res\_ICBEB, X\_train, y\_train, X\_test, y\_test, 'lstm', 'lstm\_superdiag.h5', 'lstm\_superdiag')

del(X\_train)

del(y\_train)

del(X\_test)

del(y\_test)

```

```python

X\_train, y\_train, X\_test, y\_test = load\_ICBEB(task = 'subdiag')

table\_res\_ICBEB = type\_comp\_fit\_save\_model\_score(table\_res\_ICBEB, X\_train, y\_train, X\_test, y\_test, 'lstm', 'lstm\_subdiag.h5', 'lstm\_subdiag')

del(X\_train)

del(y\_train)

del(X\_test)

del(y\_test)

```

```python

X\_train, y\_train, X\_test, y\_test = load\_ICBEB(task = 'rhythm')

table\_res\_ICBEB = type\_comp\_fit\_save\_model\_score(table\_res\_ICBEB, X\_train, y\_train, X\_test, y\_test, 'lstm', 'lstm\_rhythm.h5', 'lstm\_rhythm')

del(X\_train)

del(y\_train)

del(X\_test)

del(y\_test)

```

```python

X\_train, y\_train, X\_test, y\_test = load\_ICBEB(task = 'form')

table\_res\_ICBEB = type\_comp\_fit\_save\_model\_score(table\_res\_ICBEB, X\_train, y\_train, X\_test, y\_test, 'lstm', 'lstm\_diag.h5', 'lstm\_form')

del(X\_train)

del(y\_train)

del(X\_test)

del(y\_test)

```

**### lstm\_bidir**

```python

X\_train, y\_train, X\_test, y\_test = load\_ICBEB(task = 'diag')

table\_res\_ICBEB = type\_comp\_fit\_save\_model\_score(table\_res\_ICBEB, X\_train, y\_train, X\_test, y\_test, 'lstm\_bidir', 'lstm\_bidir\_diag.h5', 'lstm\_bidir\_diag')

del(X\_train)

del(y\_train)

del(X\_test)

del(y\_test)

```

```python

X\_train, y\_train, X\_test, y\_test = load\_ICBEB(task = 'superdiag')

table\_res\_ICBEB = type\_comp\_fit\_save\_model\_score(table\_res\_ICBEB, X\_train, y\_train, X\_test, y\_test, 'lstm\_bidir', 'lstm\_bidir\_superdiag.h5', 'lstm\_bidir\_superdiag')

del(X\_train)

del(y\_train)

del(X\_test)

del(y\_test)

```

```python

X\_train, y\_train, X\_test, y\_test = load\_ICBEB(task = 'subdiag')

table\_res\_ICBEB = type\_comp\_fit\_save\_model\_score(table\_res\_ICBEB, X\_train, y\_train, X\_test, y\_test, 'lstm\_bidir', 'lstm\_bidir\_subdiag.h5', 'lstm\_bidir\_subdiag')

del(X\_train)

del(y\_train)

del(X\_test)

del(y\_test)

```

```python

X\_train, y\_train, X\_test, y\_test = load\_ICBEB(task = 'rhytm')

table\_res\_ICBEB = type\_comp\_fit\_save\_model\_score(table\_res\_ICBEB, X\_train, y\_train, X\_test, y\_test, 'lstm\_bidir', 'lstm\_bidir\_rhytm.h5', 'lstm\_bidir\_rhytm')

del(X\_train)

del(y\_train)

del(X\_test)

del(y\_test)

```

```python

X\_train, y\_train, X\_test, y\_test = load\_ICBEB(task = 'form')

table\_res\_ICBEB = type\_comp\_fit\_save\_model\_score(table\_res\_ICBEB, X\_train, y\_train, X\_test, y\_test, 'lstm\_bidir', 'lstm\_bidir\_form.h5', 'lstm\_bidir\_form')

del(X\_train)

del(y\_train)

del(X\_test)

del(y\_test)

```

**### Сохранение результатов в формат .csv**

```python

table\_res\_ICBEB.to\_csv('table\_res\_ICBEB.csv')

```

::: {.cell .markdown}

# Загрузка данных

:::

::: {.cell .markdown} Пакет os нужен для перемещения :::

::: {.cell .code}

*# # Справка по Kaggle - перемещение*

*# import os*

*# os.chdir("/kaggle/input/icbebnpy") # Перейдем в Input (только для чтения!)*

*# !ls # Посмотреть содержимое*

*# os.chdir("/kaggle/working/") # Перейдем в Output*

*# !ls*

:::

::: {.cell .code execution\_count="10" \_cell\_guid="b1076dfc-b9ad-4769-8c92-a6c4dae69d19" \_uuid="8f2839f25d086af736a60e9eeb907d3b93b6e0e5" execution="{"iopub.execute\_input":"2023-01-30T00:41:55.070918Z","iopub.status.busy":"2023-01-30T00:41:55.070515Z","iopub.status.idle":"2023-01-30T00:41:55.170474Z","shell.execute\_reply":"2023-01-30T00:41:55.169276Z","shell.execute\_reply.started":"2023-01-30T00:41:55.070887Z"}"}

*# Для работы с данными*

import os

import pandas as pd

import numpy as np

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt *# plotting*

import seaborn as sns *# plotting heatmap*

*# Для работы с моделями*

import tensorflow as tf

from tensorflow import keras

from keras import layers

*# Для метрик*

from keras import backend as K

from keras.metrics import AUC, Recall, Precision, Accuracy, TruePositives, TrueNegatives, FalsePositives, FalseNegatives

from sklearn.metrics import fbeta\_score, precision\_score, recall\_score, accuracy\_score, roc\_auc\_score

from sklearn.metrics import auc, roc\_curve

*# Функции*

*# Загрузка ICBEB*

def load\_ICBEB(task):

os.chdir("/kaggle/input/icbebnpy")

if task == 'diag':

X\_train = np.load('ICBEBnpy/X\_train\_ICBEB\_diag.npy')

y\_train = np.load('ICBEBnpy/y\_train\_ICBEB\_diag.npy')

X\_test = np.load('ICBEBnpy/X\_val\_ICBEB\_diag.npy')

y\_test = np.load('ICBEBnpy/y\_val\_ICBEB\_diag.npy')

elif task == 'superdiag':

X\_train = np.load('ICBEBnpy/X\_train\_ICBEB\_superdiag.npy')

y\_train = np.load('ICBEBnpy/y\_train\_ICBEB\_superdiag.npy')

X\_test = np.load('ICBEBnpy/X\_val\_ICBEB\_superdiag.npy')

y\_test = np.load('ICBEBnpy/y\_val\_ICBEB\_superdiag.npy')

elif task == 'subdiag':

X\_train = np.load('ICBEBnpy/X\_train\_ICBEB\_subdiag.npy')

y\_train = np.load('ICBEBnpy/y\_train\_ICBEB\_subdiag.npy')

X\_test = np.load('ICBEBnpy/X\_val\_ICBEB\_subdiag.npy')

y\_test = np.load('ICBEBnpy/y\_val\_ICBEB\_subdiag.npy')

elif task == 'rhythm':

X\_train = np.load('ICBEBnpy/X\_train\_ICBEB\_rhythm.npy')

y\_train = np.load('ICBEBnpy/y\_train\_ICBEB\_rhythm.npy')

X\_test = np.load('ICBEBnpy/X\_val\_ICBEB\_rhythm.npy')

y\_test = np.load('ICBEBnpy/y\_val\_ICBEB\_rhythm.npy')

elif task == 'form':

X\_train = np.load('ICBEBnpy/X\_train\_ICBEB\_form.npy')

y\_train = np.load('ICBEBnpy/y\_train\_ICBEB\_form.npy')

X\_test = np.load('ICBEBnpy/X\_val\_ICBEB\_form.npy')

y\_test = np.load('ICBEBnpy/y\_val\_ICBEB\_form.npy')

*#print(X\_train.shape, y\_train.shape, X\_test.shape, y\_test.shape)*

os.chdir("/kaggle/working/")

return X\_train, y\_train, X\_test, y\_test

*# Загрузка ptbxl*

def load\_ptbxl(task):

os.chdir("/kaggle/input/ptbxlnpy")

if task == 'diag':

X\_train = np.load('X\_train\_ptbxl\_diag.npy')

y\_train = np.load('y\_train\_ptbxl\_diag.npy')

X\_test = np.load('X\_val\_ptbxl\_diag.npy')

y\_test = np.load('y\_val\_ptbxl\_diag.npy')

elif task == 'superdiag':

X\_train = np.load('X\_train\_ptbxl\_superdiag.npy')

y\_train = np.load('y\_train\_ptbxl\_superdiag.npy')

X\_test = np.load('X\_val\_ptbxl\_superdiag.npy')

y\_test = np.load('y\_val\_ptbxl\_superdiag.npy')

elif task == 'subdiag':

X\_train = np.load('X\_train\_ptbxl\_subdiag.npy')

y\_train = np.load('y\_train\_ptbxl\_subdiag.npy')

X\_test = np.load('X\_val\_ptbxl\_subdiag.npy')

y\_test = np.load('y\_val\_ptbxl\_subdiag.npy')

elif task == 'rhythm':

X\_train = np.load('X\_train\_ptbxl\_rhythm.npy')

y\_train = np.load('y\_train\_ptbxl\_rhythm.npy')

X\_test = np.load('X\_val\_ptbxl\_rhythm.npy')

y\_test = np.load('y\_val\_ptbxl\_rhythm.npy')

elif task == 'form':

X\_train = np.load('X\_train\_ptbxl\_form.npy')

y\_train = np.load('y\_train\_ptbxl\_form.npy')

X\_test = np.load('X\_val\_ptbxl\_form.npy')

y\_test = np.load('y\_val\_ptbxl\_form.npy')

os.chdir("/kaggle/working/")

*#print(X\_train.shape, y\_train.shape, X\_test.shape, y\_test.shape)*

return X\_train, y\_train, X\_test, y\_test

*# Компиляция и обучение модели*

def AUC\_Keras(y\_true, y\_pred):

auc = keras.metrics.AUC(y\_true, y\_pred)[1]

K.get\_session().run(tf.local\_variables\_initializer())

return auc

*# Компиляция и обучение модели*

def compile\_fit(model, X\_train, y\_train, X\_val = None, y\_val = None, validation\_split = 0.0, early\_stopping = None, model\_checkpoint = None):

model.compile(loss = keras.losses.CategoricalCrossentropy(),

optimizer=tf.optimizers.Adam(),

metrics=['AUC'])

if X\_val == None:

history = model.fit(X\_train, y\_train,

epochs = 30,

validation\_data = None,

validation\_split=validation\_split,

callbacks=[model\_checkpoint, early\_stopping])

else:

history = model.fit(X\_train, y\_train,

epochs = 30,

validation\_data = (X\_val, y\_val),

validation\_split=0.0,

callbacks=[model\_checkpoint, early\_stopping])

return history

*# TP TN FP FN*

def tp\_tn\_fp\_fn(y\_true, y\_pred):

TP = TruePositives()

TN = TrueNegatives()

FP = FalsePositives()

FN = FalseNegatives()

TP.update\_state(y\_true, y\_pred)

TN.update\_state(y\_true, y\_pred)

FP.update\_state(y\_true, y\_pred)

FN.update\_state(y\_true, y\_pred)

return TP.result().numpy(), TN.result().numpy(), FP.result().numpy(), FN.result().numpy()

*# Подсчет метрик*

def calc\_metrics(t, p, flag = 0): *# t - y\_true, p - y\_pred*

y\_true=np.argmax(t, axis=1)

y\_pred=np.argmax(p, axis=1)

beta = 2

f2\_score = fbeta\_score(y\_true, y\_pred, average='macro', beta=2)

precision = precision\_score(y\_true, y\_pred, average='macro')

recall = recall\_score(y\_true, y\_pred, average='macro')

TP, TN, FP, FN = tp\_tn\_fp\_fn(t, p)

g2\_score = TP/(TP+FP+beta\*FN)

if flag == 0:

return f2\_score, g2\_score

elif flag == 1:

return f2\_score, g2\_score, precision, recall

*#return f2\_score, g2\_score, AUC\_sklearn*

*# Таблица результатов*

table\_res\_finetuning = pd.DataFrame(columns = ('AUC', 'F2', 'G2'))

*# Занесение новых результатов в таблицу*

def edit\_table(table, model, X, y, index\_name): *# X - X\_test, y - y\_test*

score = model.evaluate(X, y)

y\_pr = model.predict(X) *# y\_pr - y\_test\_pred*

f2\_score, g2\_score = calc\_metrics(y, y\_pr, flag = 0)

list\_metrics = [score[1], f2\_score, g2\_score]

table.loc[index\_name] = list\_metrics

return table

*# График loss и accuracy*

def plot\_loss\_and\_accuracy\_curves(\_history):

fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(18,6))

axs[0].plot(\_history.history['loss'], color='b', label='Training loss')

axs[0].plot(\_history.history['val\_loss'], color='r', label='Validation loss')

axs[0].set\_title("Loss curves")

axs[0].legend(loc='best', shadow=True)

axs[1].plot(\_history.history['auc'], color='b', label='Training accuracy')

axs[1].plot(\_history.history['val\_auc'], color='r', label='Validation accuracy')

axs[1].set\_title("Accuracy curves")

axs[1].legend(loc='best', shadow=True)

plt.show()

*# Работа с моделями lstm и lstm\_bidir*

def type\_comp\_fit\_save\_model\_score(table, X\_train, y\_train, X\_test, y\_test, type\_model, save\_name, index\_model\_task):

*# Уточняю количество классов*

num\_classes = y\_train.shape[1]

*# Выбор архитектуры модели*

if type\_model == 'lstm':

model = keras.Sequential()

model.add(layers.LSTM(input\_shape=(1000, 12), units=256,

return\_sequences=True,

stateful=False, unroll=False

))

model.add(layers.LeakyReLU())

model.add(layers.LSTM(units=256,

return\_sequences=False,

stateful=False, unroll=False

))

model.add(layers.LeakyReLU())

model.add(layers.Dense(units=num\_classes, activation='softmax'))

print(model.summary())

*# Реализация раннего прекращения.*

checkpoint\_filepath = './checkpoint\_lstm/'

model\_checkpoint = tf.keras.callbacks.ModelCheckpoint(filepath=checkpoint\_filepath,

save\_weights\_only=True,

save\_best\_only=True)

early\_stopping = keras.callbacks.EarlyStopping(patience=15,

restore\_best\_weights=True)

elif type\_model == 'lstm\_bidir':

model = keras.Sequential()

model.add(layers.Bidirectional(layers.LSTM(input\_shape=(1000, 12), units=256,

return\_sequences=True,

stateful=False, unroll=False

)))

model.add(layers.LeakyReLU())

model.add(layers.Bidirectional(layers.LSTM(units=256,

return\_sequences=False,

stateful=False, unroll=False

)))

model.add(layers.LeakyReLU())

model.add(layers.Dense(units=num\_classes, activation='softmax'))

model.build(input\_shape = (None, 1000, 12)) *# `input\_shape` is the shape of the input data*

print(model.summary())

*# Реализация раннего прекращения.*

checkpoint\_filepath = './checkpoint\_lstm\_bidir/'

model\_checkpoint = tf.keras.callbacks.ModelCheckpoint(filepath=checkpoint\_filepath,

save\_weights\_only=True,

save\_best\_only=True)

early\_stopping = keras.callbacks.EarlyStopping(patience=15,

restore\_best\_weights=True)

*# Обучение*

History = compile\_fit(model, X\_train, y\_train, validation\_split=0.1 ,early\_stopping=early\_stopping, model\_checkpoint=model\_checkpoint)

*# Сохранение модели*

model.save(save\_name)

*# Построение графика*

plot\_loss\_and\_accuracy\_curves(History)

*# Сохранение в таблицу*

table = edit\_table(table, model, X\_test, y\_test, index\_model\_task)

return table

tf.random.set\_seed(42)

%matplotlib inline

:::

::: {.cell .markdown}

# Обучение

:::

::: {.cell .markdown}

### lstm

:::

::: {.cell .code execution\_count="11" execution="{"iopub.execute\_input":"2023-01-30T00:42:02.435210Z","iopub.status.busy":"2023-01-30T00:42:02.434849Z","iopub.status.idle":"2023-01-30T00:43:28.713903Z","shell.execute\_reply":"2023-01-30T00:43:28.712907Z","shell.execute\_reply.started":"2023-01-30T00:42:02.435178Z"}"}

X\_train, y\_train, X\_test, y\_test = load\_ICBEB(task = 'form')

table\_res\_finetuning = type\_comp\_fit\_save\_model\_score(table\_res\_finetuning, X\_train, y\_train, X\_test, y\_test, 'lstm', 'lstm\_ICBEB\_form.h5', 'lstm\_ICBEB\_form')

del(X\_train)

del(y\_train)

del(X\_test)

del(y\_test)

::: {.output .stream .stdout} Model: "sequential\_1" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Layer (type) Output Shape Param #  
================================================================= lstm\_2 (LSTM) (None, 1000, 256) 275456  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ leaky\_re\_lu\_2 (LeakyReLU) (None, 1000, 256) 0  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ lstm\_3 (LSTM) (None, 256) 525312  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ leaky\_re\_lu\_3 (LeakyReLU) (None, 256) 0  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ dense\_1 (Dense) (None, 3) 771  
================================================================= Total params: 801,539 Trainable params: 801,539 Non-trainable params: 0 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ None Epoch 1/30 43/43 [==============================] - 8s 128ms/step - loss: 1.0097 - auc: 0.6844 - val\_loss: 0.9169 - val\_auc: 0.7316 Epoch 2/30 43/43 [==============================] - 5s 105ms/step - loss: 0.9505 - auc: 0.7257 - val\_loss: 0.8545 - val\_auc: 0.7736 Epoch 3/30 43/43 [==============================] - 5s 111ms/step - loss: 1.0133 - auc: 0.6880 - val\_loss: 0.9408 - val\_auc: 0.7392 Epoch 4/30 43/43 [==============================] - 5s 107ms/step - loss: 0.9886 - auc: 0.6958 - val\_loss: 0.9044 - val\_auc: 0.7481 Epoch 5/30 43/43 [==============================] - 5s 121ms/step - loss: 0.9721 - auc: 0.7122 - val\_loss: 0.9316 - val\_auc: 0.7232 Epoch 6/30 43/43 [==============================] - 5s 107ms/step - loss: 0.9899 - auc: 0.6932 - val\_loss: 0.9398 - val\_auc: 0.7232 Epoch 7/30 43/43 [==============================] - 5s 107ms/step - loss: 0.9871 - auc: 0.6915 - val\_loss: 0.9327 - val\_auc: 0.7230 Epoch 8/30 43/43 [==============================] - 5s 111ms/step - loss: 0.9829 - auc: 0.6953 - val\_loss: 0.9312 - val\_auc: 0.7409 Epoch 9/30 43/43 [==============================] - 5s 107ms/step - loss: 0.9847 - auc: 0.6944 - val\_loss: 0.9305 - val\_auc: 0.7401 Epoch 10/30 43/43 [==============================] - 5s 111ms/step - loss: 0.9801 - auc: 0.6986 - val\_loss: 0.9226 - val\_auc: 0.7563 Epoch 11/30 43/43 [==============================] - 5s 107ms/step - loss: 0.9686 - auc: 0.7171 - val\_loss: 0.9290 - val\_auc: 0.7263 Epoch 12/30 43/43 [==============================] - 5s 122ms/step - loss: 0.9854 - auc: 0.6983 - val\_loss: 0.9252 - val\_auc: 0.7329 Epoch 13/30 43/43 [==============================] - 5s 108ms/step - loss: 0.9673 - auc: 0.7122 - val\_loss: 0.9251 - val\_auc: 0.7558 Epoch 14/30 43/43 [==============================] - 5s 108ms/step - loss: 0.9730 - auc: 0.7094 - val\_loss: 0.9211 - val\_auc: 0.7534 Epoch 15/30 43/43 [==============================] - 5s 113ms/step - loss: 0.9683 - auc: 0.7150 - val\_loss: 0.9327 - val\_auc: 0.7499 Epoch 16/30 43/43 [==============================] - 5s 109ms/step - loss: 0.9811 - auc: 0.6993 - val\_loss: 0.9500 - val\_auc: 0.7397 Epoch 17/30 43/43 [==============================] - 5s 114ms/step - loss: 0.9986 - auc: 0.6803 - val\_loss: 0.9405 - val\_auc: 0.7344 :::

::: {.output .display\_data}  :::

::: {.output .stream .stdout} 6/6 [==============================] - 0s 41ms/step - loss: 0.8326 - auc: 0.7928 ::: :::

::: {.cell .code execution\_count="15" execution="{"iopub.execute\_input":"2023-01-30T00:49:34.781442Z","iopub.status.busy":"2023-01-30T00:49:34.778870Z","iopub.status.idle":"2023-01-30T00:55:13.304221Z","shell.execute\_reply":"2023-01-30T00:55:13.303241Z","shell.execute\_reply.started":"2023-01-30T00:49:34.781404Z"}"}

X\_train, y\_train, X\_test, y\_test = load\_ICBEB(task = 'form')

table\_res\_finetuning = type\_comp\_fit\_save\_model\_score(table\_res\_finetuning, X\_train, y\_train, X\_test, y\_test, 'lstm\_bidir', 'lstm\_bidir\_ICBEB\_form.h5', 'lstm\_bidir\_ICBEB\_form')

del(X\_train)

del(y\_train)

del(X\_test)

del(y\_test)

::: {.output .stream .stdout} Model: "sequential\_2" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Layer (type) Output Shape Param #  
================================================================= bidirectional (Bidirectional (None, 1000, 512) 550912  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ leaky\_re\_lu\_4 (LeakyReLU) (None, 1000, 512) 0  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ bidirectional\_1 (Bidirection (None, 512) 1574912  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ leaky\_re\_lu\_5 (LeakyReLU) (None, 512) 0  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ dense\_2 (Dense) (None, 3) 1539  
================================================================= Total params: 2,127,363 Trainable params: 2,127,363 Non-trainable params: 0 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ None Epoch 1/30 43/43 [==============================] - 16s 270ms/step - loss: 0.9898 - auc: 0.6995 - val\_loss: 1.0521 - val\_auc: 0.6318 Epoch 2/30 43/43 [==============================] - 10s 237ms/step - loss: 0.9534 - auc: 0.7263 - val\_loss: 0.8509 - val\_auc: 0.7727 Epoch 3/30 43/43 [==============================] - 10s 242ms/step - loss: 0.9193 - auc: 0.7418 - val\_loss: 0.9240 - val\_auc: 0.7467 Epoch 4/30 43/43 [==============================] - 11s 247ms/step - loss: 0.9708 - auc: 0.7092 - val\_loss: 0.9249 - val\_auc: 0.7439 Epoch 5/30 43/43 [==============================] - 10s 243ms/step - loss: 0.8883 - auc: 0.7684 - val\_loss: 0.7841 - val\_auc: 0.8216 Epoch 6/30 43/43 [==============================] - 11s 247ms/step - loss: 0.9024 - auc: 0.7554 - val\_loss: 0.8804 - val\_auc: 0.7743 Epoch 7/30 43/43 [==============================] - 11s 253ms/step - loss: 0.8484 - auc: 0.7902 - val\_loss: 0.8626 - val\_auc: 0.7899 Epoch 8/30 43/43 [==============================] - 11s 248ms/step - loss: 0.9754 - auc: 0.7127 - val\_loss: 0.8752 - val\_auc: 0.7750 Epoch 9/30 43/43 [==============================] - 11s 248ms/step - loss: 0.9167 - auc: 0.7528 - val\_loss: 0.8906 - val\_auc: 0.7889 Epoch 10/30 43/43 [==============================] - 11s 255ms/step - loss: 0.8867 - auc: 0.7690 - val\_loss: 0.8422 - val\_auc: 0.7862 Epoch 11/30 43/43 [==============================] - 11s 251ms/step - loss: 0.8131 - auc: 0.8081 - val\_loss: 0.8114 - val\_auc: 0.7949 Epoch 12/30 43/43 [==============================] - 11s 253ms/step - loss: 0.8498 - auc: 0.7883 - val\_loss: 0.8551 - val\_auc: 0.7752 Epoch 13/30 43/43 [==============================] - 11s 257ms/step - loss: 0.8030 - auc: 0.8127 - val\_loss: 0.8248 - val\_auc: 0.7748 Epoch 14/30 43/43 [==============================] - 11s 256ms/step - loss: 0.7815 - auc: 0.8213 - val\_loss: 0.8582 - val\_auc: 0.7671 Epoch 15/30 43/43 [==============================] - 11s 256ms/step - loss: 0.7263 - auc: 0.8483 - val\_loss: 0.7519 - val\_auc: 0.8370 Epoch 16/30 43/43 [==============================] - 12s 267ms/step - loss: 0.6971 - auc: 0.8603 - val\_loss: 0.7758 - val\_auc: 0.8092 Epoch 17/30 43/43 [==============================] - 11s 256ms/step - loss: 0.7667 - auc: 0.8329 - val\_loss: 0.8135 - val\_auc: 0.8166 Epoch 18/30 43/43 [==============================] - 11s 264ms/step - loss: 0.7358 - auc: 0.8433 - val\_loss: 0.8696 - val\_auc: 0.7730 Epoch 19/30 43/43 [==============================] - 11s 258ms/step - loss: 0.6972 - auc: 0.8630 - val\_loss: 0.7473 - val\_auc: 0.8400 Epoch 20/30 43/43 [==============================] - 11s 258ms/step - loss: 0.6524 - auc: 0.8814 - val\_loss: 0.7334 - val\_auc: 0.8515 Epoch 21/30 43/43 [==============================] - 11s 261ms/step - loss: 0.6823 - auc: 0.8706 - val\_loss: 0.7125 - val\_auc: 0.8514 Epoch 22/30 43/43 [==============================] - 11s 257ms/step - loss: 0.6626 - auc: 0.8785 - val\_loss: 0.7486 - val\_auc: 0.8344 Epoch 23/30 43/43 [==============================] - 11s 256ms/step - loss: 0.6362 - auc: 0.8900 - val\_loss: 0.7160 - val\_auc: 0.8599 Epoch 24/30 43/43 [==============================] - 11s 262ms/step - loss: 0.6496 - auc: 0.8844 - val\_loss: 0.6974 - val\_auc: 0.8620 Epoch 25/30 43/43 [==============================] - 11s 257ms/step - loss: 0.5832 - auc: 0.9047 - val\_loss: 0.7511 - val\_auc: 0.8463 Epoch 26/30 43/43 [==============================] - 11s 258ms/step - loss: 0.5913 - auc: 0.9025 - val\_loss: 0.7799 - val\_auc: 0.8394 Epoch 27/30 43/43 [==============================] - 11s 261ms/step - loss: 0.5436 - auc: 0.9169 - val\_loss: 0.7476 - val\_auc: 0.8585 Epoch 28/30 43/43 [==============================] - 11s 256ms/step - loss: 0.5403 - auc: 0.9181 - val\_loss: 0.7985 - val\_auc: 0.8324 Epoch 29/30 43/43 [==============================] - 11s 257ms/step - loss: 0.5669 - auc: 0.9115 - val\_loss: 0.6552 - val\_auc: 0.8816 Epoch 30/30 43/43 [==============================] - 11s 261ms/step - loss: 0.5052 - auc: 0.9297 - val\_loss: 0.7090 - val\_auc: 0.8645 :::

::: {.output .display\_data}  :::

::: {.output .stream .stdout} 6/6 [==============================] - 1s 91ms/step - loss: 0.6400 - auc: 0.8953 ::: :::

::: {.cell .code execution\_count="17" execution="{"iopub.execute\_input":"2023-01-30T00:58:43.259893Z","iopub.status.busy":"2023-01-30T00:58:43.259202Z","iopub.status.idle":"2023-01-30T01:06:04.186141Z","shell.execute\_reply":"2023-01-30T01:06:04.185007Z","shell.execute\_reply.started":"2023-01-30T00:58:43.259856Z"}"}

X\_train, y\_train, X\_test, y\_test = load\_ptbxl(task = 'form')

table\_res\_finetuning = type\_comp\_fit\_save\_model\_score(table\_res\_finetuning, X\_train, y\_train, X\_test, y\_test, 'lstm', 'lstm\_ptbxl\_form.h5', 'lstm\_ptbxl\_form')

del(X\_train)

del(y\_train)

del(X\_test)

del(y\_test)

::: {.output .stream .stdout} Model: "sequential\_3" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Layer (type) Output Shape Param #  
================================================================= lstm\_6 (LSTM) (None, 1000, 256) 275456  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ leaky\_re\_lu\_6 (LeakyReLU) (None, 1000, 256) 0  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ lstm\_7 (LSTM) (None, 256) 525312  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ leaky\_re\_lu\_7 (LeakyReLU) (None, 256) 0  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ dense\_3 (Dense) (None, 19) 4883  
================================================================= Total params: 805,651 Trainable params: 805,651 Non-trainable params: 0 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ None Epoch 1/30 228/228 [==============================] - 28s 113ms/step - loss: 3.4375 - auc: 0.7850 - val\_loss: 1.8194 - val\_auc: 0.9292 Epoch 2/30 228/228 [==============================] - 25s 112ms/step - loss: 3.4588 - auc: 0.7905 - val\_loss: 1.7601 - val\_auc: 0.9416 Epoch 3/30 228/228 [==============================] - 25s 108ms/step - loss: 3.4941 - auc: 0.7908 - val\_loss: 1.8812 - val\_auc: 0.9406 Epoch 4/30 228/228 [==============================] - 25s 110ms/step - loss: 3.5347 - auc: 0.7904 - val\_loss: 1.8803 - val\_auc: 0.9415 Epoch 5/30 228/228 [==============================] - 25s 109ms/step - loss: 3.5597 - auc: 0.7904 - val\_loss: 1.8616 - val\_auc: 0.9390 Epoch 6/30 228/228 [==============================] - 25s 111ms/step - loss: 3.5943 - auc: 0.7909 - val\_loss: 1.7830 - val\_auc: 0.9360 Epoch 7/30 228/228 [==============================] - 25s 109ms/step - loss: 3.6383 - auc: 0.7916 - val\_loss: 1.8411 - val\_auc: 0.9388 Epoch 8/30 228/228 [==============================] - 25s 110ms/step - loss: 3.6657 - auc: 0.7913 - val\_loss: 1.8675 - val\_auc: 0.9319 Epoch 9/30 228/228 [==============================] - 25s 111ms/step - loss: 3.7255 - auc: 0.7913 - val\_loss: 2.0050 - val\_auc: 0.9147 Epoch 10/30 228/228 [==============================] - 25s 110ms/step - loss: 3.7467 - auc: 0.7915 - val\_loss: 1.8814 - val\_auc: 0.9346 Epoch 11/30 228/228 [==============================] - 25s 110ms/step - loss: 3.7903 - auc: 0.7907 - val\_loss: 1.8455 - val\_auc: 0.9427 Epoch 12/30 228/228 [==============================] - 25s 108ms/step - loss: 3.8334 - auc: 0.7910 - val\_loss: 1.9515 - val\_auc: 0.9315 Epoch 13/30 228/228 [==============================] - 25s 111ms/step - loss: 3.8643 - auc: 0.7910 - val\_loss: 1.9560 - val\_auc: 0.9253 Epoch 14/30 228/228 [==============================] - 25s 111ms/step - loss: 3.9065 - auc: 0.7907 - val\_loss: 1.9788 - val\_auc: 0.9217 Epoch 15/30 228/228 [==============================] - 25s 111ms/step - loss: 3.9195 - auc: 0.7918 - val\_loss: 1.9598 - val\_auc: 0.9230 Epoch 16/30 228/228 [==============================] - 25s 111ms/step - loss: 3.9796 - auc: 0.7909 - val\_loss: 1.8144 - val\_auc: 0.9376 Epoch 17/30 228/228 [==============================] - 25s 108ms/step - loss: 4.0280 - auc: 0.7910 - val\_loss: 1.8945 - val\_auc: 0.9328 :::

::: {.output .display\_data}  :::

::: {.output .stream .stdout} 28/28 [==============================] - 1s 42ms/step - loss: 3.3485 - auc: 0.8041 :::

::: {.output .stream .stderr} /opt/conda/lib/python3.7/site-packages/sklearn/metrics/\_classification.py:1318: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use zero\_division parameter to control this behavior. \_warn\_prf(average, modifier, msg\_start, len(result)) ::: :::

::: {.cell .code execution\_count="18" execution="{"iopub.execute\_input":"2023-01-30T01:07:17.076542Z","iopub.status.busy":"2023-01-30T01:07:17.076159Z","iopub.status.idle":"2023-01-30T01:23:52.456358Z","shell.execute\_reply":"2023-01-30T01:23:52.455351Z","shell.execute\_reply.started":"2023-01-30T01:07:17.076510Z"}"}

X\_train, y\_train, X\_test, y\_test = load\_ptbxl(task = 'form')

table\_res\_finetuning = type\_comp\_fit\_save\_model\_score(table\_res\_finetuning, X\_train, y\_train, X\_test, y\_test, 'lstm\_bidir', 'lstm\_bidir\_ptbxl\_form.h5', 'lstm\_bidir\_ptbxl\_form')

del(X\_train)

del(y\_train)

del(X\_test)

del(y\_test)

::: {.output .stream .stdout} Model: "sequential\_4" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Layer (type) Output Shape Param #  
================================================================= bidirectional\_2 (Bidirection (None, 1000, 512) 550912  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ leaky\_re\_lu\_8 (LeakyReLU) (None, 1000, 512) 0  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ bidirectional\_3 (Bidirection (None, 512) 1574912  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ leaky\_re\_lu\_9 (LeakyReLU) (None, 512) 0  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ dense\_4 (Dense) (None, 19) 9747  
================================================================= Total params: 2,135,571 Trainable params: 2,135,571 Non-trainable params: 0 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ None Epoch 1/30 228/228 [==============================] - 63s 256ms/step - loss: 3.5305 - auc: 0.7845 - val\_loss: 1.8564 - val\_auc: 0.9229 Epoch 2/30 228/228 [==============================] - 57s 251ms/step - loss: 3.6128 - auc: 0.7861 - val\_loss: 1.7218 - val\_auc: 0.9434 Epoch 3/30 228/228 [==============================] - 57s 251ms/step - loss: 3.6586 - auc: 0.7872 - val\_loss: 1.8869 - val\_auc: 0.9410 Epoch 4/30 228/228 [==============================] - 57s 250ms/step - loss: 3.7229 - auc: 0.7864 - val\_loss: 1.8257 - val\_auc: 0.9393 Epoch 5/30 228/228 [==============================] - 57s 250ms/step - loss: 3.7585 - auc: 0.7873 - val\_loss: 1.9206 - val\_auc: 0.9209 Epoch 6/30 228/228 [==============================] - 57s 252ms/step - loss: 3.8061 - auc: 0.7876 - val\_loss: 1.8142 - val\_auc: 0.9369 Epoch 7/30 228/228 [==============================] - 57s 248ms/step - loss: 3.8838 - auc: 0.7882 - val\_loss: 1.9135 - val\_auc: 0.9431 Epoch 8/30 228/228 [==============================] - 56s 247ms/step - loss: 3.9364 - auc: 0.7881 - val\_loss: 1.8066 - val\_auc: 0.9300 Epoch 9/30 228/228 [==============================] - 56s 248ms/step - loss: 4.0520 - auc: 0.7866 - val\_loss: 2.0503 - val\_auc: 0.9044 Epoch 10/30 228/228 [==============================] - 56s 247ms/step - loss: 4.0820 - auc: 0.7880 - val\_loss: 2.0322 - val\_auc: 0.9347 Epoch 11/30 228/228 [==============================] - 57s 249ms/step - loss: 4.1554 - auc: 0.7870 - val\_loss: 1.7633 - val\_auc: 0.9392 Epoch 12/30 228/228 [==============================] - 56s 248ms/step - loss: 4.2323 - auc: 0.7875 - val\_loss: 1.9467 - val\_auc: 0.9321 Epoch 13/30 228/228 [==============================] - 56s 245ms/step - loss: 4.3014 - auc: 0.7866 - val\_loss: 1.9943 - val\_auc: 0.9276 Epoch 14/30 228/228 [==============================] - 56s 247ms/step - loss: 4.3580 - auc: 0.7880 - val\_loss: 2.0631 - val\_auc: 0.9122 Epoch 15/30 228/228 [==============================] - 56s 246ms/step - loss: 4.3889 - auc: 0.7883 - val\_loss: 1.9989 - val\_auc: 0.9129 Epoch 16/30 228/228 [==============================] - 56s 246ms/step - loss: 4.4971 - auc: 0.7878 - val\_loss: 1.8422 - val\_auc: 0.9412 Epoch 17/30 228/228 [==============================] - 56s 246ms/step - loss: 4.5888 - auc: 0.7873 - val\_loss: 1.8590 - val\_auc: 0.9309 :::

::: {.output .display\_data}  :::

::: {.output .stream .stdout} 28/28 [==============================] - 3s 94ms/step - loss: 3.4990 - auc: 0.8030 :::

::: {.output .stream .stderr} /opt/conda/lib/python3.7/site-packages/sklearn/metrics/\_classification.py:1318: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use zero\_division parameter to control this behavior. \_warn\_prf(average, modifier, msg\_start, len(result)) ::: :::

::: {.cell .code execution\_count="22" execution="{"iopub.execute\_input":"2023-01-30T01:31:25.210924Z","iopub.status.busy":"2023-01-30T01:31:25.210547Z","iopub.status.idle":"2023-01-30T01:32:08.747299Z","shell.execute\_reply":"2023-01-30T01:32:08.746371Z","shell.execute\_reply.started":"2023-01-30T01:31:25.210894Z"}"}

from keras.models import load\_model

X\_train, y\_train, X\_test, y\_test = load\_ICBEB(task = 'form')

num\_classes = y\_train.shape[1]

base\_lstm = load\_model('lstm\_ptbxl\_form.h5')

base\_lstm.trainable = False

inputs = keras.Input(shape=(1000, 12))

x = base\_lstm(inputs, training = False)

outputs = layers.Dense(num\_classes)(x)

lstm = keras.Model(inputs, outputs)

lstm.summary()

*# Реализация раннего прекращения.*

checkpoint\_filepath = './checkpoint\_lstm/'

model\_checkpoint = tf.keras.callbacks.ModelCheckpoint(filepath=checkpoint\_filepath,

save\_weights\_only=True,

save\_best\_only=True)

early\_stopping = keras.callbacks.EarlyStopping(patience=15,

restore\_best\_weights=True)

*# Обучение*

History = compile\_fit(lstm, X\_train, y\_train, validation\_split=0.1 ,early\_stopping=early\_stopping, model\_checkpoint=model\_checkpoint)

*# Finetuning*

base\_lstm.trainable = True

lstm.compile(loss = keras.losses.CategoricalCrossentropy(),

optimizer=tf.optimizers.Adam(1e-5),

metrics=['AUC'])

History = lstm.fit(X\_train, y\_train,

epochs = 15,

validation\_split=0.1,

callbacks=[model\_checkpoint, early\_stopping])

*# Сохранение модели*

lstm.save('lstm\_finetuning\_form.h5')

*# Построение графика*

plot\_loss\_and\_accuracy\_curves(History)

*# Запись результатов*

table\_res\_finetuning = edit\_table(table\_res\_finetuning, lstm, X\_test, y\_test, 'lstm\_finetuning\_form')

::: {.output .stream .stdout} Model: "model\_3" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Layer (type) Output Shape Param #  
================================================================= input\_4 (InputLayer) [(None, 1000, 12)] 0  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ sequential\_3 (Sequential) (None, 19) 805651  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ dense\_8 (Dense) (None, 3) 60  
================================================================= Total params: 805,711 Trainable params: 60 Non-trainable params: 805,651 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Epoch 1/30 43/43 [==============================] - 5s 64ms/step - loss: 1.0847 - auc: 0.6156 - val\_loss: 0.9304 - val\_auc: 0.7313 Epoch 2/30 43/43 [==============================] - 2s 46ms/step - loss: 0.9911 - auc: 0.6866 - val\_loss: 0.9355 - val\_auc: 0.7313 Epoch 3/30 43/43 [==============================] - 2s 47ms/step - loss: 0.9922 - auc: 0.6867 - val\_loss: 0.9444 - val\_auc: 0.7313 Epoch 4/30 43/43 [==============================] - 2s 46ms/step - loss: 0.9910 - auc: 0.6880 - val\_loss: 0.9264 - val\_auc: 0.7313 Epoch 5/30 43/43 [==============================] - 2s 52ms/step - loss: 0.9920 - auc: 0.6808 - val\_loss: 0.9377 - val\_auc: 0.7313 Epoch 6/30 43/43 [==============================] - 2s 46ms/step - loss: 0.9905 - auc: 0.6877 - val\_loss: 0.9411 - val\_auc: 0.7313 Epoch 7/30 43/43 [==============================] - 2s 46ms/step - loss: 0.9907 - auc: 0.6879 - val\_loss: 0.9348 - val\_auc: 0.7313 Epoch 8/30 43/43 [==============================] - 2s 47ms/step - loss: 0.9909 - auc: 0.6861 - val\_loss: 0.9404 - val\_auc: 0.7313 Epoch 9/30 43/43 [==============================] - 2s 46ms/step - loss: 0.9916 - auc: 0.6856 - val\_loss: 0.9425 - val\_auc: 0.7313 Epoch 10/30 43/43 [==============================] - 3s 68ms/step - loss: 0.9926 - auc: 0.6818 - val\_loss: 0.9395 - val\_auc: 0.7313 Epoch 11/30 43/43 [==============================] - 2s 46ms/step - loss: 0.9906 - auc: 0.6886 - val\_loss: 0.9354 - val\_auc: 0.7313 Epoch 12/30 43/43 [==============================] - 2s 46ms/step - loss: 0.9921 - auc: 0.6837 - val\_loss: 0.9325 - val\_auc: 0.7313 Epoch 13/30 43/43 [==============================] - 2s 46ms/step - loss: 0.9908 - auc: 0.6877 - val\_loss: 0.9363 - val\_auc: 0.7313 Epoch 14/30 43/43 [==============================] - 2s 48ms/step - loss: 0.9934 - auc: 0.6785 - val\_loss: 0.9349 - val\_auc: 0.7313 Epoch 15/30 43/43 [==============================] - 2s 52ms/step - loss: 0.9921 - auc: 0.6842 - val\_loss: 0.9370 - val\_auc: 0.7313 Epoch 16/30 43/43 [==============================] - 2s 46ms/step - loss: 0.9917 - auc: 0.6865 - val\_loss: 0.9336 - val\_auc: 0.7313 Epoch 17/30 43/43 [==============================] - 2s 47ms/step - loss: 0.9922 - auc: 0.6824 - val\_loss: 0.9360 - val\_auc: 0.7313 Epoch 18/30 43/43 [==============================] - 2s 47ms/step - loss: 0.9906 - auc: 0.6861 - val\_loss: 0.9372 - val\_auc: 0.7313 Epoch 19/30 43/43 [==============================] - 2s 47ms/step - loss: 0.9924 - auc: 0.6832 - val\_loss: 0.9446 - val\_auc: 0.7313 ::: :::

::: {.cell .code execution\_count="25" execution="{"iopub.execute\_input":"2023-01-30T01:33:38.039164Z","iopub.status.busy":"2023-01-30T01:33:38.038754Z","iopub.status.idle":"2023-01-30T01:35:38.735307Z","shell.execute\_reply":"2023-01-30T01:35:38.734298Z","shell.execute\_reply.started":"2023-01-30T01:33:38.039134Z"}"}

::: {.output .stream .stdout} Epoch 1/15 43/43 [==============================] - 41s 123ms/step - loss: 0.9922 - auc: 0.6856 - val\_loss: 0.9296 - val\_auc: 0.7313 Epoch 2/15 43/43 [==============================] - 5s 110ms/step - loss: 0.9897 - auc: 0.6848 - val\_loss: 0.9334 - val\_auc: 0.7313 Epoch 3/15 43/43 [==============================] - 4s 104ms/step - loss: 0.9896 - auc: 0.6865 - val\_loss: 0.9335 - val\_auc: 0.7313 Epoch 4/15 43/43 [==============================] - 5s 109ms/step - loss: 0.9897 - auc: 0.6851 - val\_loss: 0.9341 - val\_auc: 0.7313 Epoch 5/15 43/43 [==============================] - 5s 106ms/step - loss: 0.9894 - auc: 0.6865 - val\_loss: 0.9339 - val\_auc: 0.7313 Epoch 6/15 43/43 [==============================] - 5s 106ms/step - loss: 0.9898 - auc: 0.6847 - val\_loss: 0.9331 - val\_auc: 0.7313 Epoch 7/15 43/43 [==============================] - 5s 123ms/step - loss: 0.9895 - auc: 0.6865 - val\_loss: 0.9339 - val\_auc: 0.7313 Epoch 8/15 43/43 [==============================] - 5s 106ms/step - loss: 0.9892 - auc: 0.6865 - val\_loss: 0.9345 - val\_auc: 0.7313 Epoch 9/15 43/43 [==============================] - 5s 112ms/step - loss: 0.9895 - auc: 0.6841 - val\_loss: 0.9335 - val\_auc: 0.7313 Epoch 10/15 43/43 [==============================] - 5s 107ms/step - loss: 0.9895 - auc: 0.6865 - val\_loss: 0.9353 - val\_auc: 0.7313 Epoch 11/15 43/43 [==============================] - 5s 111ms/step - loss: 0.9894 - auc: 0.6855 - val\_loss: 0.9338 - val\_auc: 0.7313 Epoch 12/15 43/43 [==============================] - 5s 106ms/step - loss: 0.9896 - auc: 0.6865 - val\_loss: 0.9337 - val\_auc: 0.7313 Epoch 13/15 43/43 [==============================] - 5s 107ms/step - loss: 0.9898 - auc: 0.6865 - val\_loss: 0.9341 - val\_auc: 0.7313 Epoch 14/15 43/43 [==============================] - 5s 125ms/step - loss: 0.9896 - auc: 0.6849 - val\_loss: 0.9351 - val\_auc: 0.7313 Epoch 15/15 43/43 [==============================] - 5s 107ms/step - loss: 0.9898 - auc: 0.6870 - val\_loss: 0.9339 - val\_auc: 0.7313 :::

::: {.output .display\_data}  :::

::: {.output .stream .stdout} 6/6 [==============================] - 0s 41ms/step - loss: 0.9813 - auc: 0.6917 :::

::: {.output .stream .stderr} /opt/conda/lib/python3.7/site-packages/sklearn/metrics/\_classification.py:1318: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use zero\_division parameter to control this behavior. \_warn\_prf(average, modifier, msg\_start, len(result)) ::: :::

::: {.cell .code execution\_count="31" execution="{"iopub.execute\_input":"2023-01-30T01:46:04.446388Z","iopub.status.busy":"2023-01-30T01:46:04.445277Z","iopub.status.idle":"2023-01-30T01:47:36.467510Z","shell.execute\_reply":"2023-01-30T01:47:36.466525Z","shell.execute\_reply.started":"2023-01-30T01:46:04.446319Z"}"}

X\_train, y\_train, X\_test, y\_test = load\_ICBEB(task = 'form')

num\_classes = y\_train.shape[1]

base\_lstm\_bidir = load\_model('lstm\_bidir\_ptbxl\_form.h5')

base\_lstm\_bidir.trainable = False

inputs = keras.Input(shape=(1000, 12))

x = base\_lstm\_bidir(inputs, training = False)

outputs = layers.Dense(num\_classes)(x)

lstm\_bidir = keras.Model(inputs, outputs)

lstm\_bidir.summary()

*# Реализация раннего прекращения.*

checkpoint\_filepath = './checkpoint\_lstm\_bidir/'

model\_checkpoint = tf.keras.callbacks.ModelCheckpoint(filepath=checkpoint\_filepath,

save\_weights\_only=True,

save\_best\_only=True)

early\_stopping = keras.callbacks.EarlyStopping(patience=15,

restore\_best\_weights=True)

*# Обучение*

History = compile\_fit(lstm\_bidir, X\_train, y\_train, validation\_split=0.1 ,early\_stopping=early\_stopping, model\_checkpoint=model\_checkpoint)

*# Finetuning*

base\_lstm\_bidir.trainable = True

lstm\_bidir.compile(loss = keras.losses.CategoricalCrossentropy(),

optimizer=tf.optimizers.Adam(1e-5),

metrics=['AUC'])

History = lstm\_bidir.fit(X\_train, y\_train,

epochs = 15,

validation\_split=0.1,

callbacks=[model\_checkpoint, early\_stopping])

*# Сохранение модели*

lstm\_bidir.save('lstm\_bidir\_finetuning\_form.h5')

*# Построение графика*

plot\_loss\_and\_accuracy\_curves(History)

*# Запись результатов*

table\_res\_finetuning = edit\_table(table\_res\_finetuning, lstm, X\_test, y\_test, 'lstm\_bidir\_finetuning\_form')

::: {.output .stream .stdout} Epoch 1/30 43/43 [==============================] - 10s 130ms/step - loss: 1.1275 - auc: 0.6597 - val\_loss: 0.9573 - val\_auc: 0.7313 Epoch 2/30 43/43 [==============================] - 5s 105ms/step - loss: 1.0066 - auc: 0.6850 - val\_loss: 0.9337 - val\_auc: 0.7313 Epoch 3/30 43/43 [==============================] - 4s 100ms/step - loss: 0.9904 - auc: 0.6867 - val\_loss: 0.9338 - val\_auc: 0.7313 Epoch 4/30 43/43 [==============================] - 5s 111ms/step - loss: 0.9899 - auc: 0.6852 - val\_loss: 0.9289 - val\_auc: 0.7313 Epoch 5/30 43/43 [==============================] - 5s 106ms/step - loss: 0.9904 - auc: 0.6851 - val\_loss: 0.9372 - val\_auc: 0.7313 Epoch 6/30 43/43 [==============================] - 4s 101ms/step - loss: 0.9901 - auc: 0.6838 - val\_loss: 0.9367 - val\_auc: 0.7313 Epoch 7/30 43/43 [==============================] - 5s 106ms/step - loss: 0.9905 - auc: 0.6858 - val\_loss: 0.9307 - val\_auc: 0.7313 Epoch 8/30 43/43 [==============================] - 4s 101ms/step - loss: 0.9895 - auc: 0.6849 - val\_loss: 0.9343 - val\_auc: 0.7313 Epoch 9/30 43/43 [==============================] - 4s 102ms/step - loss: 0.9897 - auc: 0.6850 - val\_loss: 0.9371 - val\_auc: 0.7313 Epoch 10/30 43/43 [==============================] - 5s 108ms/step - loss: 0.9903 - auc: 0.6799 - val\_loss: 0.9360 - val\_auc: 0.7313 Epoch 11/30 43/43 [==============================] - 5s 110ms/step - loss: 0.9897 - auc: 0.6856 - val\_loss: 0.9346 - val\_auc: 0.7313 Epoch 12/30 43/43 [==============================] - 5s 108ms/step - loss: 0.9901 - auc: 0.6810 - val\_loss: 0.9321 - val\_auc: 0.7313 Epoch 13/30 43/43 [==============================] - 4s 102ms/step - loss: 0.9904 - auc: 0.6821 - val\_loss: 0.9320 - val\_auc: 0.7313 Epoch 14/30 43/43 [==============================] - 5s 109ms/step - loss: 0.9903 - auc: 0.6822 - val\_loss: 0.9357 - val\_auc: 0.7313 Epoch 15/30 43/43 [==============================] - 4s 102ms/step - loss: 0.9900 - auc: 0.6850 - val\_loss: 0.9331 - val\_auc: 0.7313 Epoch 16/30 43/43 [==============================] - 4s 102ms/step - loss: 0.9899 - auc: 0.6800 - val\_loss: 0.9328 - val\_auc: 0.7313 Epoch 17/30 43/43 [==============================] - 5s 108ms/step - loss: 0.9900 - auc: 0.6820 - val\_loss: 0.9348 - val\_auc: 0.7313 Epoch 18/30 43/43 [==============================] - 5s 114ms/step - loss: 0.9895 - auc: 0.6863 - val\_loss: 0.9337 - val\_auc: 0.7313 Epoch 19/30 43/43 [==============================] - 5s 109ms/step - loss: 0.9909 - auc: 0.6805 - val\_loss: 0.9354 - val\_auc: 0.7313 ::: :::

::: {.cell .markdown}

### Сохранение результатов в формат .csv {#сохранение-результатов-в-формат-csv}

:::

::: {.cell .code execution\_count="32" execution="{"iopub.execute\_input":"2023-01-30T01:48:11.871983Z","iopub.status.busy":"2023-01-30T01:48:11.871271Z","iopub.status.idle":"2023-01-30T01:48:11.882071Z","shell.execute\_reply":"2023-01-30T01:48:11.881147Z","shell.execute\_reply.started":"2023-01-30T01:48:11.871948Z"}"}

table\_res\_finetuning.to\_csv('table\_res\_finetuning.csv')

:::

::: {.cell .code execution\_count="33" execution="{"iopub.execute\_input":"2023-01-30T01:48:18.337783Z","iopub.status.busy":"2023-01-30T01:48:18.337072Z","iopub.status.idle":"2023-01-30T01:48:30.690512Z","shell.execute\_reply":"2023-01-30T01:48:30.689296Z","shell.execute\_reply.started":"2023-01-30T01:48:18.337746Z"}"}

table\_res\_finetuning

**# Подключаем пакеты и определяем функции**

```python

# Для работы с данными

import pandas as pd

import numpy as np

import wfdb

import ast

from utils import utils

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt   # plotting

#import seaborn as sns   # plotting heatmap

# Для работы с моделями

import tensorflow as tf

from tensorflow import keras

from keras import layers

# Для метрик

from keras import backend as K

from keras.metrics import AUC, Recall, Precision, Accuracy, TruePositives, TrueNegatives, FalsePositives, FalseNegatives

from sklearn.metrics import fbeta\_score, precision\_score, recall\_score, accuracy\_score, roc\_auc\_score

# Функции

# Компиляция и обучение модели

def compile\_fit(model, X\_train, y\_train, X\_val, y\_val, early\_stopping, model\_checkpoint):

  model.compile(loss = keras.losses.CategoricalCrossentropy(),

                optimizer=tf.optimizers.Adam(),

                metrics=['accuracy'])

  history = model.fit(X\_train, y\_train, epochs = 30, validation\_data = (X\_val, y\_val), callbacks=[model\_checkpoint, early\_stopping])

  return history

tf.random.set\_seed(42)

%matplotlib inline

```

**# Скачиваем ICBEB с использованием кода и обработки авторов исследуемой статьи (обработанные данные, т.е. проведена нормализация и категоризация)**

```python

sampling\_frequency=100

datafolder='data/ICBEB/'

task='diagnostic'

#task='superdiagnostic'

#task = 'subdiagnostic'

#task = 'rhythm'

#task = 'form'

outputfolder='output/'

# Load ICBEB data

data, raw\_labels = utils.load\_dataset(datafolder, sampling\_frequency)

# Preprocess label data

labels = utils.compute\_label\_aggregations(raw\_labels, datafolder, task)

# Select relevant data and convert to one-hot

data, labels, Y, \_ = utils.select\_data(data, labels, task, min\_samples=0, outputfolder=outputfolder)

# 1-9 for training

X\_train = data[labels.strat\_fold < 10]

y\_train = Y[labels.strat\_fold < 10]

# 10 for validation

X\_val = data[labels.strat\_fold == 10]

y\_val = Y[labels.strat\_fold == 10]

# Стандартизация 3D данных c применением StandardScaler.

# Сначала изменяется форма данных а затем применяется нормализация. После этого требуется вернуть их прежнюю форму

def standard\_scaler(X\_train, X\_val):

  scaler = StandardScaler()

  # Train

  num\_instances, num\_time\_steps, num\_features = X\_train.shape

  X\_train = np.reshape(X\_train, newshape=(-1, num\_features))

  X\_train = scaler.fit\_transform(X\_train)

  X\_train = np.reshape(X\_train, newshape=(num\_instances, num\_time\_steps, num\_features))

  # Valid

  num\_instances, num\_time\_steps, num\_features = X\_val.shape

  X\_val = np.reshape(X\_val, newshape=(-1, num\_features))

  X\_val = scaler.fit\_transform(X\_val)

  X\_val = np.reshape(X\_val, newshape=(num\_instances, num\_time\_steps, num\_features))

  return X\_train, X\_val

X\_train.shape, y\_train.shape, X\_val.shape, y\_val.shape

```

    ((3337, 1000, 12), (3337, 4), (378, 1000, 12), (378, 4))

**##### task = 'diagnostic'**

```python

# # Выделим уникальные значения y\_train в столбце и посчитаем их количество

# unique, counts = np.unique(y\_train, return\_counts = True, axis = 0)

# print(unique, '\n', 'len(unique): ', len(unique))

# print(counts, '\n', 'len(counts): ', len(counts))

```

```python

# Стандартизация данных

X\_train, X\_val = standard\_scaler(X\_train, X\_val)

```

```python

# Сохранение наборов данных в файлы .npy для дальнейшего использования в Google Colab

np.save('X\_train\_ICBEB\_diag', X\_train)

np.save('X\_val\_ICBEB\_diag', X\_val)

np.save('y\_train\_ICBEB\_diag', y\_train)

np.save('y\_val\_ICBEB\_diag', y\_val)

```

**##### task = 'superdiagnostic'**

```python

# # Выделим уникальные значения y\_train в столбце и посчитаем их количество

# unique, counts = np.unique(y\_train, return\_counts = True, axis = 0)

# print(unique, '\n', 'len(unique): ', len(unique))

# print(counts, '\n', 'len(counts): ', len(counts))

```

```python

# Стандартизация данных

X\_train, X\_val = standard\_scaler(X\_train, X\_val)

# Сохранение наборов данных в файлы .npy для дальнейшего использования в Google Colab

np.save('X\_train\_ICBEB\_superdiag', X\_train)

np.save('X\_val\_ICBEB\_superdiag', X\_val)

np.save('y\_train\_ICBEB\_superdiag', y\_train)

np.save('y\_val\_ICBEB\_superdiag', y\_val)

```

**##### task = 'subdiagnostic'**

```python

# # Выделим уникальные значения y\_train в столбце и посчитаем их количество

# unique, counts = np.unique(y\_train, return\_counts = True, axis = 0)

# print(unique, '\n', 'len(unique): ', len(unique))

# print(counts, '\n', 'len(counts): ', len(counts))

```

```python

# Стандартизация данных

X\_train, X\_val = standard\_scaler(X\_train, X\_val)

# Сохранение наборов данных в файлы .npy для дальнейшего использования в Google Colab

np.save('X\_train\_ICBEB\_subdiag', X\_train)

np.save('X\_val\_ICBEB\_subdiag', X\_val)

np.save('y\_train\_ICBEB\_subdiag', y\_train)

np.save('y\_val\_ICBEB\_subdiag', y\_val)

```

**##### task = 'rhythm'**

```python

# # Выделим уникальные значения y\_train в столбце и посчитаем их количество

# unique, counts = np.unique(y\_train, return\_counts = True, axis = 0)

# print(unique, '\n', 'len(unique): ', len(unique))

# print(counts, '\n', 'len(counts): ', len(counts))

```

```python

# Стандартизация данных

X\_train, X\_val = standard\_scaler(X\_train, X\_val)

# Сохранение наборов данных в файлы .npy для дальнейшего использования в Google Colab

np.save('X\_train\_ICBEB\_rhythm', X\_train)

np.save('X\_val\_ICBEB\_rhythm', X\_val)

np.save('y\_train\_ICBEB\_rhythm', y\_train)

np.save('y\_val\_ICBEB\_rhythm', y\_val)

```

**##### task = 'form'**

```python

# # Выделим уникальные значения y\_train в столбце и посчитаем их количество

# unique, counts = np.unique(y\_train, return\_counts = True, axis = 0)

# print(unique, '\n', 'len(unique): ', len(unique))

# print(counts, '\n', 'len(counts): ', len(counts))

```

```python

# Стандартизация данных

X\_train, X\_val = standard\_scaler(X\_train, X\_val)

# Сохранение наборов данных в файлы .npy для дальнейшего использования в Google Colab

np.save('X\_train\_ICBEB\_form', X\_train)

np.save('X\_val\_ICBEB\_form', X\_val)

np.save('y\_train\_ICBEB\_form', y\_train)

np.save('y\_val\_ICBEB\_form', y\_val)

```

**# Скачиваем PTB-XL с использованием кода и обработки авторов исследуемой статьи (обработанные данные, т.е. проведена нормализация и категоризация)**

```python

sampling\_frequency=100

datafolder='data/ptbxl/'

#task='diagnostic'

#task='superdiagnostic'

#task = 'subdiagnostic'

#task = 'rhythm'

task = 'form'

outputfolder='output/'

# Load PTB-XL data

data, raw\_labels = utils.load\_dataset(datafolder, sampling\_frequency)

# Preprocess label data

labels = utils.compute\_label\_aggregations(raw\_labels, datafolder, task)

# Select relevant data and convert to one-hot

data, labels, Y, \_ = utils.select\_data(data, labels, task, min\_samples=0, outputfolder=outputfolder)

# 1-9 for training

X\_train = data[labels.strat\_fold < 10]

y\_train = Y[labels.strat\_fold < 10]

# 10 for validation

X\_val = data[labels.strat\_fold == 10]

y\_val = Y[labels.strat\_fold == 10]

# Стандартизация 3D данных c применением StandardScaler.

# Сначала изменяется форма данных а затем применяется нормализация. После этого требуется вернуть их прежнюю форму

def standard\_scaler(X\_train, X\_val):

  scaler = StandardScaler()

  # Train

  num\_instances, num\_time\_steps, num\_features = X\_train.shape

  X\_train = np.reshape(X\_train, newshape=(-1, num\_features))

  X\_train = scaler.fit\_transform(X\_train)

  X\_train = np.reshape(X\_train, newshape=(num\_instances, num\_time\_steps, num\_features))

  # Valid

  num\_instances, num\_time\_steps, num\_features = X\_val.shape

  X\_val = np.reshape(X\_val, newshape=(-1, num\_features))

  X\_val = scaler.fit\_transform(X\_val)

  X\_val = np.reshape(X\_val, newshape=(num\_instances, num\_time\_steps, num\_features))

  return X\_train, X\_val

X\_train.shape, y\_train.shape, X\_val.shape, y\_val.shape

```

    ((8106, 1000, 12), (8106, 19), (882, 1000, 12), (882, 19))

**##### task = 'diagnostic'**

```python

# # Выделим уникальные значения y\_train в столбце и посчитаем их количество

# unique, counts = np.unique(y\_train, return\_counts = True, axis = 0)

# print(unique, '\n', 'len(unique): ', len(unique))

# print(counts, '\n', 'len(counts): ', len(counts))

```

```python

# Стандартизация данных

X\_train, X\_val = standard\_scaler(X\_train, X\_val)

# Сохранение наборов данных в файлы .npy для дальнейшего использования в Google Colab

np.save('X\_train\_ptbxl\_diag', X\_train)

np.save('X\_val\_ptbxl\_diag', X\_val)

np.save('y\_train\_ptbxl\_diag', y\_train)

np.save('y\_val\_ptbxl\_diag', y\_val)

```

**##### task = 'superdiagnostic'**

```python

# # Выделим уникальные значения y\_train в столбце и посчитаем их количество

# unique, counts = np.unique(y\_train, return\_counts = True, axis = 0)

# print(unique, '\n', 'len(unique): ', len(unique))

# print(counts, '\n', 'len(counts): ', len(counts))

```

```python

# Стандартизация данных

X\_train, X\_val = standard\_scaler(X\_train, X\_val)

# Сохранение наборов данных в файлы .npy для дальнейшего использования в Google Colab

np.save('X\_train\_ptbxl\_superdiag', X\_train)

np.save('X\_val\_ptbxl\_superdiag', X\_val)

np.save('y\_train\_ptbxl\_superdiag', y\_train)

np.save('y\_val\_ptbxl\_superdiag', y\_val)

```

**##### task = 'subdiagnostic'**

```python

# # Выделим уникальные значения y\_train в столбце и посчитаем их количество

# unique, counts = np.unique(y\_train, return\_counts = True, axis = 0)

# print(unique, '\n', 'len(unique): ', len(unique))

# print(counts, '\n', 'len(counts): ', len(counts))

```

```python

# Стандартизация данных

X\_train, X\_val = standard\_scaler(X\_train, X\_val)

# Сохранение наборов данных в файлы .npy для дальнейшего использования в Google Colab

np.save('X\_train\_ptbxl\_subdiag', X\_train)

np.save('X\_val\_ptbxl\_subdiag', X\_val)

np.save('y\_train\_ptbxl\_subdiag', y\_train)

np.save('y\_val\_ptbxl\_subdiag', y\_val)

```

**##### task = 'rhythm'**

```python

# # Выделим уникальные значения y\_train в столбце и посчитаем их количество

# unique, counts = np.unique(y\_train, return\_counts = True, axis = 0)

# print(unique, '\n', 'len(unique): ', len(unique))

# print(counts, '\n', 'len(counts): ', len(counts))

```

```python

# Стандартизация данных

X\_train, X\_val = standard\_scaler(X\_train, X\_val)

# Сохранение наборов данных в файлы .npy для дальнейшего использования в Google Colab

np.save('X\_train\_ptbxl\_rhythm', X\_train)

np.save('X\_val\_ptbxl\_rhythm', X\_val)

np.save('y\_train\_ptbxl\_rhythm', y\_train)

np.save('y\_val\_ptbxl\_rhythm', y\_val)

```

**##### task = 'form'**

```python

# # Выделим уникальные значения y\_train в столбце и посчитаем их количество

# unique, counts = np.unique(y\_train, return\_counts = True, axis = 0)

# print(unique, '\n', 'len(unique): ', len(unique))

# print(counts, '\n', 'len(counts): ', len(counts))

```

```python

# Стандартизация данных

X\_train, X\_val = standard\_scaler(X\_train, X\_val)

# Сохранение наборов данных в файлы .npy для дальнейшего использования в Google Colab

np.save('X\_train\_ptbxl\_form', X\_train)

np.save('X\_val\_ptbxl\_form', X\_val)

np.save('y\_train\_ptbxl\_form', y\_train)

np.save('y\_val\_ptbxl\_form', y\_val)

```

**# Скачиваем ICBEB через форму PTB-XL**

```python

def load\_raw\_data(df, sampling\_rate, path):

    if sampling\_rate == 100:

        data = [wfdb.rdsamp(path+f) for f in df.filename]

    data = [signal for signal, meta in data]

    for index, value in enumerate(data):

      if len(value) < 1000:

        data.remove(value)

        df = df.drop(index = index)

    refLen = 1000 # reference - эталон

    for index, value in enumerate(data):

      if len(value) > 1000:

        data[index] = data[index][:refLen]

    data = np.array(data)

    return data, df

path = 'data/ICBEB/'

sampling\_rate=100

# load and convert annotation data

Y = pd.read\_csv(path+'icbeb\_database.csv', index\_col='ecg\_id')

Y.scp\_codes = Y.scp\_codes.apply(lambda x: ast.literal\_eval(x))

# Load raw signal data

X, Y = load\_raw\_data(Y, sampling\_rate, path)

# Load scp\_statements.csv for diagnostic aggregation

agg\_df = pd.read\_csv(path+'scp\_statements.csv', index\_col=0)

agg\_df = agg\_df[agg\_df.form == 1]

def aggregate\_diagnostic(y\_dic):

    tmp = []

    for key in y\_dic.keys():

        if key in agg\_df.index:

            tmp.append(agg\_df.loc[key].diagnostic\_class)

    return list(set(tmp))

# Apply diagnostic superclass

Y['diagnostic\_superclass'] = Y.scp\_codes.apply(aggregate\_diagnostic)

# # Split data into train and test

test\_fold = 10

# Train

X\_train = X[np.where(Y.strat\_fold != test\_fold)]

y\_train = Y[(Y.strat\_fold != test\_fold)].diagnostic\_superclass

# Test

X\_test = X[np.where(Y.strat\_fold == test\_fold)]

y\_test = Y[Y.strat\_fold == test\_fold].diagnostic\_superclass

```

    C:\Users\manuk\AppData\Local\Temp\ipykernel\_7528\110442798.py:7: DeprecationWarning: elementwise comparison failed; this will raise an error in the future.

      data.remove(value)

    C:\Users\manuk\AppData\Local\Temp\ipykernel\_7528\110442798.py:7: DeprecationWarning: elementwise comparison failed; this will raise an error in the future.

      data.remove(value)

    C:\Users\manuk\AppData\Local\Temp\ipykernel\_7528\110442798.py:7: DeprecationWarning: elementwise comparison failed; this will raise an error in the future.

      data.remove(value)

    C:\Users\manuk\AppData\Local\Temp\ipykernel\_7528\110442798.py:7: DeprecationWarning: elementwise comparison failed; this will raise an error in the future.

      data.remove(value)

    C:\Users\manuk\AppData\Local\Temp\ipykernel\_7528\110442798.py:7: DeprecationWarning: elementwise comparison failed; this will raise an error in the future.

      data.remove(value)

    C:\Users\manuk\AppData\Local\Temp\ipykernel\_7528\110442798.py:7: DeprecationWarning: elementwise comparison failed; this will raise an error in the future.

      data.remove(value)

```python

X\_train.shape, y\_train.shape

```

    ((6181, 1000, 12), (6181,))

```python

X\_train[0]

```

    array([[ 0.02822609,  0.00672717, -0.02150135, ..., -0.11200486,

            -0.59595824, -0.01558333],

           [ 0.05623843,  0.02373605, -0.03248586, ..., -0.09599664,

            -0.55091871,  0.03543248],

           [ 0.06525131,  0.05676969, -0.00847507, ..., -0.05695029,

            -0.48688198,  0.10148025],

           ...,

           [ 0.19612203, -0.16015955, -0.35627225, ...,  0.38222251,

             0.2396717 , -0.25792291],

           [-0.01297901, -0.48225593, -0.46927696, ..., -0.44072623,

            -0.56097927, -0.85331045],

           [-0.34466227, -0.59409148, -0.24942707, ..., -0.98916564,

            -1.00918153, -1.11767564]])

```python

y\_train

```

    ecg\_id

    1          []

    2          []

    3          []

    5          []

    6          []

            ...

    6873       []

    6874    [nan]

    6875       []

    6876       []

    6877       []

    Name: diagnostic\_superclass, Length: 6181, dtype: object

```python

Y

```

<div>

<style scoped>

    .dataframe tbody tr th:only-of-type {

        vertical-align: middle;

    }

    .dataframe tbody tr th {

        vertical-align: top;

    }

    .dataframe thead th {

        text-align: right;

    }

</style>

<table *border*="1" class="dataframe">

  <thead>

    <tr style="text-align: right;">

      <th></th>

      <th>filename</th>

      <th>validation</th>

      <th>age</th>

      <th>sex</th>

      <th>scp\_codes</th>

      <th>patient\_id</th>

      <th>quality</th>

      <th>strat\_fold</th>

      <th>diagnostic\_superclass</th>

    </tr>

    <tr>

      <th>ecg\_id</th>

      <th></th>

      <th></th>

      <th></th>

      <th></th>

      <th></th>

      <th></th>

      <th></th>

      <th></th>

      <th></th>

    </tr>

  </thead>

  <tbody>

    <tr>

      <th>1</th>

      <td>records100/1</td>

      <td>False</td>

      <td>74.0</td>

      <td>1</td>

      <td>{'CRBBB': 100}</td>

      <td>1</td>

      <td>0</td>

      <td>7</td>

      <td>[]</td>

    </tr>

    <tr>

      <th>2</th>

      <td>records100/2</td>

      <td>False</td>

      <td>49.0</td>

      <td>0</td>

      <td>{'NORM': 100}</td>

      <td>2</td>

      <td>0</td>

      <td>7</td>

      <td>[]</td>

    </tr>

    <tr>

      <th>3</th>

      <td>records100/3</td>

      <td>False</td>

      <td>81.0</td>

      <td>0</td>

      <td>{'AFIB': 100}</td>

      <td>3</td>

      <td>0</td>

      <td>3</td>

      <td>[]</td>

    </tr>

    <tr>

      <th>4</th>

      <td>records100/4</td>

      <td>False</td>

      <td>45.0</td>

      <td>1</td>

      <td>{'AFIB': 100}</td>

      <td>4</td>

      <td>0</td>

      <td>10</td>

      <td>[]</td>

    </tr>

    <tr>

      <th>5</th>

      <td>records100/5</td>

      <td>False</td>

      <td>53.0</td>

      <td>1</td>

      <td>{'VPC': 100}</td>

      <td>5</td>

      <td>0</td>

      <td>7</td>

      <td>[]</td>

    </tr>

    <tr>

      <th>...</th>

      <td>...</td>

      <td>...</td>

      <td>...</td>

      <td>...</td>

      <td>...</td>

      <td>...</td>

      <td>...</td>

      <td>...</td>

      <td>...</td>

    </tr>

    <tr>

      <th>6873</th>

      <td>records100/6873</td>

      <td>False</td>

      <td>80.0</td>

      <td>1</td>

      <td>{'1AVB': 100}</td>

      <td>6873</td>

      <td>0</td>

      <td>5</td>

      <td>[]</td>

    </tr>

    <tr>

      <th>6874</th>

      <td>records100/6874</td>

      <td>False</td>

      <td>62.0</td>

      <td>0</td>

      <td>{'STD\_': 100}</td>

      <td>6874</td>

      <td>0</td>

      <td>1</td>

      <td>[nan]</td>

    </tr>

    <tr>

      <th>6875</th>

      <td>records100/6875</td>

      <td>False</td>

      <td>78.0</td>

      <td>1</td>

      <td>{'CLBBB': 100}</td>

      <td>6875</td>

      <td>0</td>

      <td>5</td>

      <td>[]</td>

    </tr>

    <tr>

      <th>6876</th>

      <td>records100/6876</td>

      <td>False</td>

      <td>-1.0</td>

      <td>0</td>

      <td>{'AFIB': 100}</td>

      <td>6876</td>

      <td>0</td>

      <td>6</td>

      <td>[]</td>

    </tr>

    <tr>

      <th>6877</th>

      <td>records100/6877</td>

      <td>False</td>

      <td>71.0</td>

      <td>0</td>

      <td>{'VPC': 100}</td>

      <td>6877</td>

      <td>0</td>

      <td>1</td>

      <td>[]</td>

    </tr>

  </tbody>

</table>

<p>6871 rows × 9 columns</p>

</div>

```python

# Создать Ndarray Numpy копию Series Pandas.

y\_trainNp = y\_train.to\_numpy()

print(type(y\_trainNp))

print(y\_trainNp.size)

```

    <class 'numpy.ndarray'>

    6181

```python

# Выделим уникальные значения y\_train\_np и посчитаем их количество.

unique, counts = np.unique(y\_trainNp, return\_counts = True)

print(unique, '\n', "unique.size: ", unique.size)

print(counts)

```

    [list([]) list([nan])]

     unique.size:  2

    [4658 1523]

```python

import pandas as pd

table = pd.read\_csv('table\_res\_finetuning.csv', index\_col=0)

table

# Загрузка данных

Пакет os нужен для перемещения

import os

os.chdir("/kaggle/input/icbebnpy") *# Перейдем в Input (только для чтения!)*

!ls *# Посмотреть содержимое*

os.chdir("/kaggle/working/") *# Перейдем в Output*

!ls

*# Для работы с данными*

import os

import pandas as pd

import numpy as np

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt *# plotting*

import seaborn as sns *# plotting heatmap*

*# Для работы с моделями*

import tensorflow as tf

from tensorflow import keras

from keras import layers

*# Для метрик*

from keras import backend as K

from keras.metrics import AUC, Recall, Precision, Accuracy, TruePositives, TrueNegatives, FalsePositives, FalseNegatives

from sklearn.metrics import fbeta\_score, precision\_score, recall\_score, accuracy\_score, roc\_auc\_score

from sklearn.metrics import auc, roc\_curve

*# Функции*

*# Загрузка ICBEB*

def load\_ICBEB(task):

os.chdir("/kaggle/input/icbebnpy")

if task == 'diag':

X\_train = np.load('ICBEBnpy/X\_train\_ICBEB\_diag.npy')

y\_train = np.load('ICBEBnpy/y\_train\_ICBEB\_diag.npy')

X\_test = np.load('ICBEBnpy/X\_val\_ICBEB\_diag.npy')

y\_test = np.load('ICBEBnpy/y\_val\_ICBEB\_diag.npy')

elif task == 'superdiag':

X\_train = np.load('ICBEBnpy/X\_train\_ICBEB\_superdiag.npy')

y\_train = np.load('ICBEBnpy/y\_train\_ICBEB\_superdiag.npy')

X\_test = np.load('ICBEBnpy/X\_val\_ICBEB\_superdiag.npy')

y\_test = np.load('ICBEBnpy/y\_val\_ICBEB\_superdiag.npy')

elif task == 'subdiag':

X\_train = np.load('ICBEBnpy/X\_train\_ICBEB\_subdiag.npy')

y\_train = np.load('ICBEBnpy/y\_train\_ICBEB\_subdiag.npy')

X\_test = np.load('ICBEBnpy/X\_val\_ICBEB\_subdiag.npy')

y\_test = np.load('ICBEBnpy/y\_val\_ICBEB\_subdiag.npy')

elif task == 'rhythm':

X\_train = np.load('ICBEBnpy/X\_train\_ICBEB\_rhythm.npy')

y\_train = np.load('ICBEBnpy/y\_train\_ICBEB\_rhythm.npy')

X\_test = np.load('ICBEBnpy/X\_val\_ICBEB\_rhythm.npy')

y\_test = np.load('ICBEBnpy/y\_val\_ICBEB\_rhythm.npy')

elif task == 'form':

X\_train = np.load('ICBEBnpy/X\_train\_ICBEB\_form.npy')

y\_train = np.load('ICBEBnpy/y\_train\_ICBEB\_form.npy')

X\_test = np.load('ICBEBnpy/X\_val\_ICBEB\_form.npy')

y\_test = np.load('ICBEBnpy/y\_val\_ICBEB\_form.npy')

*#print(X\_train.shape, y\_train.shape, X\_test.shape, y\_test.shape)*

os.chdir("/kaggle/working/")

return X\_train, y\_train, X\_test, y\_test

*# Загрузка ptbxl*

def load\_ptbxl(task):

if task == 'diag':

X\_train = np.load('ptbxlnpy/X\_train\_ptbxl\_diag.npy')

y\_train = np.load('ptbxlnpy/y\_train\_ptbxl\_diag.npy')

X\_test = np.load('ptbxlnpy/X\_val\_ptbxl\_diag.npy')

y\_test = np.load('ptbxlnpy/y\_val\_ptbxl\_diag.npy')

elif task == 'superdiag':

X\_train = np.load('ptbxlnpy/X\_train\_ptbxl\_superdiag.npy')

y\_train = np.load('ptbxlnpy/y\_train\_ptbxl\_superdiag.npy')

X\_test = np.load('ptbxlnpy/X\_val\_ptbxl\_superdiag.npy')

y\_test = np.load('ptbxlnpy/y\_val\_ptbxl\_superdiag.npy')

elif task == 'subdiag':

X\_train = np.load('ptbxlnpy/X\_train\_ptbxl\_subdiag.npy')

y\_train = np.load('ptbxlnpy/y\_train\_ptbxl\_subdiag.npy')

X\_test = np.load('ptbxlnpy/X\_val\_ptbxl\_subdiag.npy')

y\_test = np.load('ptbxlnpy/y\_val\_ptbxl\_subdiag.npy')

elif task == 'rhythm':

X\_train = np.load('ptbxlnpy/X\_train\_ptbxl\_rhythm.npy')

y\_train = np.load('ptbxlnpy/y\_train\_ptbxl\_rhythm.npy')

X\_test = np.load('ptbxlnpy/X\_val\_ptbxl\_rhythm.npy')

y\_test = np.load('ptbxlnpy/y\_val\_ptbxl\_rhythm.npy')

elif task == 'form':

X\_train = np.load('ptbxlnpy/X\_train\_ptbxl\_form.npy')

y\_train = np.load('ptbxlnpy/y\_train\_ptbxl\_form.npy')

X\_test = np.load('ptbxlnpy/X\_val\_ptbxl\_form.npy')

y\_test = np.load('ptbxlnpy/y\_val\_ptbxl\_form.npy')

*#print(X\_train.shape, y\_train.shape, X\_test.shape, y\_test.shape)*

return X\_train, y\_train, X\_test, y\_test

*# Компиляция и обучение модели*

def AUC\_Keras(y\_true, y\_pred):

auc = keras.metrics.AUC(y\_true, y\_pred)[1]

K.get\_session().run(tf.local\_variables\_initializer())

return auc

*# Компиляция и обучение модели*

def compile\_fit(model, X\_train, y\_train, X\_val = None, y\_val = None, validation\_split = 0.0, early\_stopping = None, model\_checkpoint = None):

model.compile(loss = keras.losses.CategoricalCrossentropy(),

optimizer=tf.optimizers.Adam(),

metrics=['AUC'])

if X\_val == None:

history = model.fit(X\_train, y\_train,

epochs = 30,

validation\_data = None,

validation\_split=validation\_split,

callbacks=[model\_checkpoint, early\_stopping])

else:

history = model.fit(X\_train, y\_train,

epochs = 30,

validation\_data = (X\_val, y\_val),

validation\_split=0.0,

callbacks=[model\_checkpoint, early\_stopping])

return history

*# TP TN FP FN*

def tp\_tn\_fp\_fn(y\_true, y\_pred):

TP = TruePositives()

TN = TrueNegatives()

FP = FalsePositives()

FN = FalseNegatives()

TP.update\_state(y\_true, y\_pred)

TN.update\_state(y\_true, y\_pred)

FP.update\_state(y\_true, y\_pred)

FN.update\_state(y\_true, y\_pred)

return TP.result().numpy(), TN.result().numpy(), FP.result().numpy(), FN.result().numpy()

*# Подсчет метрик*

def calc\_metrics(t, p, flag = 0): *# t - y\_true, p - y\_pred*

y\_true=np.argmax(t, axis=1)

y\_pred=np.argmax(p, axis=1)

beta = 2

f2\_score = fbeta\_score(y\_true, y\_pred, average='macro', beta=2)

precision = precision\_score(y\_true, y\_pred, average='macro')

recall = recall\_score(y\_true, y\_pred, average='macro')

TP, TN, FP, FN = tp\_tn\_fp\_fn(t, p)

g2\_score = TP/(TP+FP+beta\*FN)

if flag == 0:

return f2\_score, g2\_score

elif flag == 1:

return f2\_score, g2\_score, precision, recall

*#return f2\_score, g2\_score, AUC\_sklearn*

*# Таблица результатов*

table\_res\_ICBEB = pd.DataFrame(columns = ('AUC', 'F2', 'G2'))

*# Занесение новых результатов в таблицу*

def edit\_table(table, model, X, y, index\_name): *# X - X\_test, y - y\_test*

score = model.evaluate(X, y)

y\_pr = model.predict(X) *# y\_pr - y\_test\_pred*

f2\_score, g2\_score = calc\_metrics(y, y\_pr, flag = 0)

list\_metrics = [f2\_score, g2\_score, score[1]]

table.loc[index\_name] = list\_metrics

return table

*# График loss и accuracy*

def plot\_loss\_and\_accuracy\_curves(\_history):

fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(18,6))

axs[0].plot(\_history.history['loss'], color='b', label='Training loss')

axs[0].plot(\_history.history['val\_loss'], color='r', label='Validation loss')

axs[0].set\_title("Loss curves")

axs[0].legend(loc='best', shadow=True)

axs[1].plot(\_history.history['auc'], color='b', label='Training accuracy')

axs[1].plot(\_history.history['val\_auc'], color='r', label='Validation accuracy')

axs[1].set\_title("Accuracy curves")

axs[1].legend(loc='best', shadow=True)

plt.show()

*# Работа с моделями lstm и lstm\_bidir*

def type\_comp\_fit\_save\_model\_score(table, X\_train, y\_train, X\_test, y\_test, type\_model, save\_name, index\_model\_task):

*# Уточняю количество классов*

num\_classes = y\_train.shape[1]

*# Выбор архитектуры модели*

if type\_model == 'lstm':

model = keras.Sequential()

model.add(layers.LSTM(input\_shape=(1000, 12), units=256,

return\_sequences=True,

stateful=False, unroll=False

))

model.add(layers.LeakyReLU())

model.add(layers.LSTM(units=256,

return\_sequences=False,

stateful=False, unroll=False

))

model.add(layers.LeakyReLU())

model.add(layers.Dense(units=num\_classes, activation='softmax'))

print(model.summary())

*# Реализация раннего прекращения.*

checkpoint\_filepath = './checkpoint\_lstm/'

model\_checkpoint = tf.keras.callbacks.ModelCheckpoint(filepath=checkpoint\_filepath,

save\_weights\_only=True,

save\_best\_only=True)

early\_stopping = keras.callbacks.EarlyStopping(patience=15,

restore\_best\_weights=True)

elif type\_model == 'lstm\_bidir':

model = keras.Sequential()

model.add(layers.Bidirectional(layers.LSTM(input\_shape=(1000, 12), units=256,

return\_sequences=True,

stateful=False, unroll=False

)))

model.add(layers.LeakyReLU())

model.add(layers.Bidirectional(layers.LSTM(units=256,

return\_sequences=False,

stateful=False, unroll=False

)))

model.add(layers.LeakyReLU())

model.add(layers.Dense(units=num\_classes, activation='softmax'))

model.build(input\_shape = (None, 1000, 12)) *# `input\_shape` is the shape of the input data*

print(model.summary())

*# Реализация раннего прекращения.*

checkpoint\_filepath = './checkpoint\_lstm\_bidir/'

model\_checkpoint = tf.keras.callbacks.ModelCheckpoint(filepath=checkpoint\_filepath,

save\_weights\_only=True,

save\_best\_only=True)

early\_stopping = keras.callbacks.EarlyStopping(patience=15,

restore\_best\_weights=True)

*# Обучение*

History = compile\_fit(model, X\_train, y\_train, validation\_split=0.1 ,early\_stopping=early\_stopping, model\_checkpoint=model\_checkpoint)

*# Сохранение модели*

model.save\_weights(save\_name)

*# Построение графика*

plot\_loss\_and\_accuracy\_curves(History)

*# Сохранение в таблицу*

table = edit\_table(table, model, X\_test, y\_test, index\_model\_task)

return table

tf.random.set\_seed(42)

%matplotlib inline

# Обучение

### lstm

X\_train, y\_train, X\_test, y\_test = load\_ICBEB(task = 'diag')

table\_res\_ICBEB = type\_comp\_fit\_save\_model\_score(table\_res\_ICBEB, X\_train, y\_train, X\_test, y\_test, 'lstm', 'lstm\_diag.hdf5', 'lstm\_diag')

del(X\_train)

del(y\_train)

del(X\_test)

del(y\_test)

2023-01-29 21:19:05.234843: I tensorflow/stream\_executor/cuda/cuda\_gpu\_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero

2023-01-29 21:19:05.244643: I tensorflow/stream\_executor/cuda/cuda\_gpu\_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero

2023-01-29 21:19:05.245379: I tensorflow/stream\_executor/cuda/cuda\_gpu\_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero

2023-01-29 21:19:05.247036: I tensorflow/core/platform/cpu\_feature\_guard.cc:142] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX2 AVX512F FMA

To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.

2023-01-29 21:19:05.247414: I tensorflow/stream\_executor/cuda/cuda\_gpu\_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero

2023-01-29 21:19:05.248082: I tensorflow/stream\_executor/cuda/cuda\_gpu\_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero

2023-01-29 21:19:05.248713: I tensorflow/stream\_executor/cuda/cuda\_gpu\_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero

2023-01-29 21:19:05.907001: I tensorflow/stream\_executor/cuda/cuda\_gpu\_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero

2023-01-29 21:19:05.907885: I tensorflow/stream\_executor/cuda/cuda\_gpu\_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero

2023-01-29 21:19:05.908595: I tensorflow/stream\_executor/cuda/cuda\_gpu\_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero

2023-01-29 21:19:05.909171: I tensorflow/core/common\_runtime/gpu/gpu\_device.cc:1510] Created device /job:localhost/replica:0/task:0/device:GPU:0 with 15401 MB memory: -> device: 0, name: Tesla P100-PCIE-16GB, pci bus id: 0000:00:04.0, compute capability: 6.0

Model: "sequential"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

lstm (LSTM) (None, 1000, 256) 275456

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

leaky\_re\_lu (LeakyReLU) (None, 1000, 256) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

lstm\_1 (LSTM) (None, 256) 525312

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

leaky\_re\_lu\_1 (LeakyReLU) (None, 256) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense (Dense) (None, 4) 1028

=================================================================

Total params: 801,796

Trainable params: 801,796

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

None

2023-01-29 21:19:07.062983: I tensorflow/compiler/mlir/mlir\_graph\_optimization\_pass.cc:185] None of the MLIR Optimization Passes are enabled (registered 2)

Epoch 1/30

2023-01-29 21:19:09.830499: I tensorflow/stream\_executor/cuda/cuda\_dnn.cc:369] Loaded cuDNN version 8005

94/94 [==============================] - 18s 156ms/step - loss: 1.1412 - auc: 0.7575 - val\_loss: 1.1258 - val\_auc: 0.7608

Epoch 2/30

94/94 [==============================] - 14s 148ms/step - loss: 1.1964 - auc: 0.7284 - val\_loss: 1.1940 - val\_auc: 0.7300

Epoch 3/30

94/94 [==============================] - 14s 149ms/step - loss: 1.1887 - auc: 0.7294 - val\_loss: 1.2046 - val\_auc: 0.7120

Epoch 4/30

94/94 [==============================] - 14s 148ms/step - loss: 1.1770 - auc: 0.7291 - val\_loss: 1.1784 - val\_auc: 0.7394

Epoch 5/30

94/94 [==============================] - 14s 149ms/step - loss: 1.1368 - auc: 0.7488 - val\_loss: 1.1170 - val\_auc: 0.7649

Epoch 6/30

94/94 [==============================] - 14s 148ms/step - loss: 1.0698 - auc: 0.7740 - val\_loss: 1.1830 - val\_auc: 0.7292

Epoch 7/30

94/94 [==============================] - 14s 148ms/step - loss: 1.1229 - auc: 0.7545 - val\_loss: 1.0768 - val\_auc: 0.7742

Epoch 8/30

94/94 [==============================] - 14s 148ms/step - loss: 1.0308 - auc: 0.7903 - val\_loss: 1.0208 - val\_auc: 0.7996

Epoch 9/30

94/94 [==============================] - 14s 148ms/step - loss: 1.0321 - auc: 0.7951 - val\_loss: 1.0185 - val\_auc: 0.8025

Epoch 10/30

94/94 [==============================] - 14s 148ms/step - loss: 1.1662 - auc: 0.7483 - val\_loss: 1.0540 - val\_auc: 0.7962

Epoch 11/30

94/94 [==============================] - 14s 148ms/step - loss: 1.0459 - auc: 0.7940 - val\_loss: 1.0232 - val\_auc: 0.8087

Epoch 12/30

94/94 [==============================] - 14s 148ms/step - loss: 1.0931 - auc: 0.7717 - val\_loss: 1.0588 - val\_auc: 0.7797

Epoch 13/30

94/94 [==============================] - 14s 148ms/step - loss: 1.0154 - auc: 0.8056 - val\_loss: 1.0619 - val\_auc: 0.7922

Epoch 14/30

94/94 [==============================] - 14s 148ms/step - loss: 1.0056 - auc: 0.8221 - val\_loss: 1.2077 - val\_auc: 0.7964

Epoch 15/30

94/94 [==============================] - 14s 148ms/step - loss: 1.0760 - auc: 0.7923 - val\_loss: 0.9862 - val\_auc: 0.8361

Epoch 16/30

94/94 [==============================] - 14s 148ms/step - loss: 0.8604 - auc: 0.8728 - val\_loss: 0.8249 - val\_auc: 0.8898

Epoch 17/30

94/94 [==============================] - 14s 149ms/step - loss: 0.7138 - auc: 0.9143 - val\_loss: 0.9609 - val\_auc: 0.8670

Epoch 18/30

94/94 [==============================] - 14s 148ms/step - loss: 0.6719 - auc: 0.9226 - val\_loss: 0.5875 - val\_auc: 0.9391

Epoch 19/30

94/94 [==============================] - 14s 148ms/step - loss: 0.5851 - auc: 0.9400 - val\_loss: 0.5643 - val\_auc: 0.9454

Epoch 20/30

94/94 [==============================] - 14s 149ms/step - loss: 0.5642 - auc: 0.9427 - val\_loss: 0.5468 - val\_auc: 0.9474

Epoch 21/30

94/94 [==============================] - 14s 150ms/step - loss: 0.5545 - auc: 0.9456 - val\_loss: 0.5406 - val\_auc: 0.9495

Epoch 22/30

94/94 [==============================] - 14s 148ms/step - loss: 0.8462 - auc: 0.8805 - val\_loss: 0.7327 - val\_auc: 0.9092

Epoch 23/30

94/94 [==============================] - 14s 149ms/step - loss: 0.6884 - auc: 0.9178 - val\_loss: 0.6755 - val\_auc: 0.9228

Epoch 24/30

94/94 [==============================] - 14s 148ms/step - loss: 0.5874 - auc: 0.9395 - val\_loss: 0.5476 - val\_auc: 0.9492

Epoch 25/30

94/94 [==============================] - 14s 148ms/step - loss: 0.5434 - auc: 0.9460 - val\_loss: 0.5963 - val\_auc: 0.9359

Epoch 26/30

94/94 [==============================] - 14s 148ms/step - loss: 0.5406 - auc: 0.9463 - val\_loss: 0.5320 - val\_auc: 0.9481

Epoch 27/30

94/94 [==============================] - 14s 148ms/step - loss: 0.5691 - auc: 0.9421 - val\_loss: 0.5029 - val\_auc: 0.9556

Epoch 28/30

94/94 [==============================] - 14s 148ms/step - loss: 0.5038 - auc: 0.9528 - val\_loss: 0.4811 - val\_auc: 0.9564

Epoch 29/30

94/94 [==============================] - 14s 148ms/step - loss: 0.5225 - auc: 0.9499 - val\_loss: 0.5045 - val\_auc: 0.9521

Epoch 30/30

94/94 [==============================] - 14s 148ms/step - loss: 0.4923 - auc: 0.9545 - val\_loss: 0.5039 - val\_auc: 0.9542

12/12 [==============================] - 1s 56ms/step - loss: 0.5082 - auc: 0.9553

X\_train, y\_train, X\_test, y\_test = load\_ICBEB(task = 'superdiag')

table\_res\_ICBEB = type\_comp\_fit\_save\_model\_score(table\_res\_ICBEB, X\_train, y\_train, X\_test, y\_test, 'lstm', 'lstm\_superdiag.hdf5', 'lstm\_superdiag')

del(X\_train)

del(y\_train)

del(X\_test)

del(y\_test)

Model: "sequential\_1"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

lstm\_2 (LSTM) (None, 1000, 256) 275456

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

leaky\_re\_lu\_2 (LeakyReLU) (None, 1000, 256) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

lstm\_3 (LSTM) (None, 256) 525312

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

leaky\_re\_lu\_3 (LeakyReLU) (None, 256) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_1 (Dense) (None, 2) 514

=================================================================

Total params: 801,282

Trainable params: 801,282

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

None

Epoch 1/30

94/94 [==============================] - 17s 155ms/step - loss: 0.5142 - auc: 0.8224 - val\_loss: 0.7028 - val\_auc: 0.8663

Epoch 2/30

94/94 [==============================] - 14s 149ms/step - loss: 0.4637 - auc: 0.8613 - val\_loss: 0.4676 - val\_auc: 0.8657

Epoch 3/30

94/94 [==============================] - 14s 148ms/step - loss: 0.4117 - auc: 0.8721 - val\_loss: 0.4100 - val\_auc: 0.8642

Epoch 4/30

94/94 [==============================] - 14s 150ms/step - loss: 0.4642 - auc: 0.8565 - val\_loss: 0.6045 - val\_auc: 0.7990

Epoch 5/30

94/94 [==============================] - 14s 149ms/step - loss: 0.5431 - auc: 0.7835 - val\_loss: 0.5222 - val\_auc: 0.8031

Epoch 6/30

94/94 [==============================] - 14s 149ms/step - loss: 0.5265 - auc: 0.8007 - val\_loss: 0.5638 - val\_auc: 0.8046

Epoch 7/30

94/94 [==============================] - 14s 149ms/step - loss: 0.5538 - auc: 0.7865 - val\_loss: 0.5803 - val\_auc: 0.7736

Epoch 8/30

94/94 [==============================] - 14s 149ms/step - loss: 0.5401 - auc: 0.7900 - val\_loss: 0.4872 - val\_auc: 0.8427

Epoch 9/30

94/94 [==============================] - 14s 149ms/step - loss: 0.4987 - auc: 0.8388 - val\_loss: 0.4878 - val\_auc: 0.8557

Epoch 10/30

94/94 [==============================] - 14s 149ms/step - loss: 0.4445 - auc: 0.8650 - val\_loss: 0.4246 - val\_auc: 0.8781

Epoch 11/30

94/94 [==============================] - 14s 149ms/step - loss: 0.4806 - auc: 0.8440 - val\_loss: 0.4556 - val\_auc: 0.8829

Epoch 12/30

94/94 [==============================] - 14s 149ms/step - loss: 0.4365 - auc: 0.8670 - val\_loss: 0.3701 - val\_auc: 0.9203

Epoch 13/30

94/94 [==============================] - 14s 149ms/step - loss: 0.3641 - auc: 0.9077 - val\_loss: 0.3630 - val\_auc: 0.8925

Epoch 14/30

94/94 [==============================] - 14s 149ms/step - loss: 0.3520 - auc: 0.9137 - val\_loss: 0.3093 - val\_auc: 0.9439

Epoch 15/30

94/94 [==============================] - 14s 149ms/step - loss: 0.3208 - auc: 0.9331 - val\_loss: 0.3681 - val\_auc: 0.9104

Epoch 16/30

94/94 [==============================] - 14s 149ms/step - loss: 0.3238 - auc: 0.9314 - val\_loss: 0.3039 - val\_auc: 0.9420

Epoch 17/30

94/94 [==============================] - 14s 148ms/step - loss: 0.2996 - auc: 0.9428 - val\_loss: 0.3091 - val\_auc: 0.9432

Epoch 18/30

94/94 [==============================] - 14s 149ms/step - loss: 0.2885 - auc: 0.9472 - val\_loss: 0.2795 - val\_auc: 0.9575

Epoch 19/30

94/94 [==============================] - 14s 149ms/step - loss: 0.2819 - auc: 0.9495 - val\_loss: 0.2718 - val\_auc: 0.9539

Epoch 20/30

94/94 [==============================] - 14s 149ms/step - loss: 0.2728 - auc: 0.9528 - val\_loss: 0.2546 - val\_auc: 0.9618

Epoch 21/30

94/94 [==============================] - 14s 149ms/step - loss: 0.2577 - auc: 0.9589 - val\_loss: 0.2596 - val\_auc: 0.9581

Epoch 22/30

94/94 [==============================] - 14s 150ms/step - loss: 0.2504 - auc: 0.9614 - val\_loss: 0.3229 - val\_auc: 0.9346

Epoch 23/30

94/94 [==============================] - 14s 149ms/step - loss: 0.2561 - auc: 0.9586 - val\_loss: 0.2636 - val\_auc: 0.9566

Epoch 24/30

94/94 [==============================] - 14s 149ms/step - loss: 0.2347 - auc: 0.9660 - val\_loss: 0.2541 - val\_auc: 0.9594

Epoch 25/30

94/94 [==============================] - 14s 149ms/step - loss: 0.2437 - auc: 0.9629 - val\_loss: 0.2497 - val\_auc: 0.9607

Epoch 26/30

94/94 [==============================] - 14s 149ms/step - loss: 0.2249 - auc: 0.9686 - val\_loss: 0.2371 - val\_auc: 0.9649

Epoch 27/30

94/94 [==============================] - 14s 149ms/step - loss: 0.2182 - auc: 0.9705 - val\_loss: 0.2533 - val\_auc: 0.9601

Epoch 28/30

94/94 [==============================] - 14s 149ms/step - loss: 0.2167 - auc: 0.9707 - val\_loss: 0.2386 - val\_auc: 0.9643

Epoch 29/30

94/94 [==============================] - 14s 149ms/step - loss: 0.2069 - auc: 0.9735 - val\_loss: 0.2492 - val\_auc: 0.9607

Epoch 30/30

94/94 [==============================] - 14s 148ms/step - loss: 0.2025 - auc: 0.9747 - val\_loss: 0.2756 - val\_auc: 0.9594

12/12 [==============================] - 1s 56ms/step - loss: 0.2496 - auc: 0.9638

X\_train, y\_train, X\_test, y\_test = load\_ICBEB(task = 'subdiag')

table\_res\_ICBEB = type\_comp\_fit\_save\_model\_score(table\_res\_ICBEB, X\_train, y\_train, X\_test, y\_test, 'lstm', 'lstm\_subdiag.hdf5', 'lstm\_subdiag')

del(X\_train)

del(y\_train)

del(X\_test)

del(y\_test)

Model: "sequential\_2"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

lstm\_4 (LSTM) (None, 1000, 256) 275456

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

leaky\_re\_lu\_4 (LeakyReLU) (None, 1000, 256) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

lstm\_5 (LSTM) (None, 256) 525312

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

leaky\_re\_lu\_5 (LeakyReLU) (None, 256) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_2 (Dense) (None, 4) 1028

=================================================================

Total params: 801,796

Trainable params: 801,796

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

None

Epoch 1/30

94/94 [==============================] - 18s 157ms/step - loss: 1.0575 - auc: 0.7941 - val\_loss: 1.1733 - val\_auc: 0.7483

Epoch 2/30

94/94 [==============================] - 14s 148ms/step - loss: 1.1645 - auc: 0.7511 - val\_loss: 1.1090 - val\_auc: 0.7910

Epoch 3/30

94/94 [==============================] - 14s 148ms/step - loss: 0.9212 - auc: 0.8516 - val\_loss: 0.9046 - val\_auc: 0.8562

Epoch 4/30

94/94 [==============================] - 14s 149ms/step - loss: 0.9727 - auc: 0.8379 - val\_loss: 1.0934 - val\_auc: 0.7746

Epoch 5/30

94/94 [==============================] - 14s 148ms/step - loss: 1.2490 - auc: 0.7261 - val\_loss: 1.1608 - val\_auc: 0.7413

Epoch 6/30

94/94 [==============================] - 14s 149ms/step - loss: 1.1713 - auc: 0.7400 - val\_loss: 1.1606 - val\_auc: 0.7408

Epoch 7/30

94/94 [==============================] - 14s 148ms/step - loss: 1.1670 - auc: 0.7404 - val\_loss: 1.1616 - val\_auc: 0.7434

Epoch 8/30

94/94 [==============================] - 14s 149ms/step - loss: 1.1694 - auc: 0.7398 - val\_loss: 1.1606 - val\_auc: 0.7425

Epoch 9/30

94/94 [==============================] - 14s 148ms/step - loss: 1.1750 - auc: 0.7382 - val\_loss: 1.1554 - val\_auc: 0.7519

Epoch 10/30

94/94 [==============================] - 14s 149ms/step - loss: 1.1633 - auc: 0.7445 - val\_loss: 1.1710 - val\_auc: 0.7464

Epoch 11/30

94/94 [==============================] - 14s 148ms/step - loss: 1.1328 - auc: 0.7558 - val\_loss: 1.0677 - val\_auc: 0.7930

Epoch 12/30

94/94 [==============================] - 14s 148ms/step - loss: 1.0180 - auc: 0.8070 - val\_loss: 1.0048 - val\_auc: 0.8144

Epoch 13/30

94/94 [==============================] - 14s 149ms/step - loss: 0.9750 - auc: 0.8256 - val\_loss: 1.0269 - val\_auc: 0.8153

Epoch 14/30

94/94 [==============================] - 14s 148ms/step - loss: 0.8709 - auc: 0.8690 - val\_loss: 0.7731 - val\_auc: 0.8975

Epoch 15/30

94/94 [==============================] - 14s 149ms/step - loss: 0.7352 - auc: 0.9074 - val\_loss: 0.6976 - val\_auc: 0.9150

Epoch 16/30

94/94 [==============================] - 14s 148ms/step - loss: 0.6562 - auc: 0.9264 - val\_loss: 0.6361 - val\_auc: 0.9319

Epoch 17/30

94/94 [==============================] - 14s 149ms/step - loss: 0.6000 - auc: 0.9374 - val\_loss: 0.6337 - val\_auc: 0.9235

Epoch 18/30

94/94 [==============================] - 14s 149ms/step - loss: 0.5689 - auc: 0.9418 - val\_loss: 0.5481 - val\_auc: 0.9478

Epoch 19/30

94/94 [==============================] - 14s 149ms/step - loss: 0.5398 - auc: 0.9466 - val\_loss: 0.5226 - val\_auc: 0.9462

Epoch 20/30

94/94 [==============================] - 14s 148ms/step - loss: 0.5215 - auc: 0.9494 - val\_loss: 0.5334 - val\_auc: 0.9502

Epoch 21/30

94/94 [==============================] - 14s 148ms/step - loss: 0.5231 - auc: 0.9495 - val\_loss: 0.5214 - val\_auc: 0.9506

Epoch 22/30

94/94 [==============================] - 14s 149ms/step - loss: 0.4951 - auc: 0.9540 - val\_loss: 0.5223 - val\_auc: 0.9498

Epoch 23/30

94/94 [==============================] - 14s 148ms/step - loss: 0.4860 - auc: 0.9554 - val\_loss: 0.4882 - val\_auc: 0.9566

Epoch 24/30

94/94 [==============================] - 14s 149ms/step - loss: 0.4828 - auc: 0.9553 - val\_loss: 0.5142 - val\_auc: 0.9521

Epoch 25/30

94/94 [==============================] - 14s 149ms/step - loss: 0.4711 - auc: 0.9577 - val\_loss: 0.4988 - val\_auc: 0.9552

Epoch 26/30

94/94 [==============================] - 14s 148ms/step - loss: 0.4987 - auc: 0.9536 - val\_loss: 0.5080 - val\_auc: 0.9546

Epoch 27/30

94/94 [==============================] - 14s 148ms/step - loss: 0.4801 - auc: 0.9570 - val\_loss: 0.4834 - val\_auc: 0.9584

Epoch 28/30

94/94 [==============================] - 14s 148ms/step - loss: 0.4675 - auc: 0.9591 - val\_loss: 0.5041 - val\_auc: 0.9559

Epoch 29/30

94/94 [==============================] - 14s 149ms/step - loss: 0.4596 - auc: 0.9604 - val\_loss: 0.7496 - val\_auc: 0.9276

Epoch 30/30

94/94 [==============================] - 14s 148ms/step - loss: 0.5285 - auc: 0.9491 - val\_loss: 0.5267 - val\_auc: 0.9512

12/12 [==============================] - 1s 57ms/step - loss: 0.5248 - auc: 0.9552

X\_train, y\_train, X\_test, y\_test = load\_ICBEB(task = 'rhythm')

table\_res\_ICBEB = type\_comp\_fit\_save\_model\_score(table\_res\_ICBEB, X\_train, y\_train, X\_test, y\_test, 'lstm', 'lstm\_rhythm.h5', 'lstm\_rhythm')

del(X\_train)

del(y\_train)

del(X\_test)

del(y\_test)

Model: "sequential\_6"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

lstm\_12 (LSTM) (None, 1000, 256) 275456

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

leaky\_re\_lu\_12 (LeakyReLU) (None, 1000, 256) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

lstm\_13 (LSTM) (None, 256) 525312

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

leaky\_re\_lu\_13 (LeakyReLU) (None, 256) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_6 (Dense) (None, 1) 257

=================================================================

Total params: 801,025

Trainable params: 801,025

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

None

Epoch 1/30

25/31 [=======================>......] - ETA: 0s - loss: 0.0000e+00 - auc: 0.0000e+00

---------------------------------------------------------------------------

KeyboardInterrupt Traceback (most recent call last)

/tmp/ipykernel\_584/4023372390.py in <module>

1 X\_train, y\_train, X\_test, y\_test = load\_ICBEB(task = 'rhythm')

----> 2 table\_res\_ICBEB = type\_comp\_fit\_save\_model\_score(table\_res\_ICBEB, X\_train, y\_train, X\_test, y\_test, 'lstm', 'lstm\_rhythm.h5', 'lstm\_rhythm')

3 del(X\_train)

4 del(y\_train)

5 del(X\_test)

/tmp/ipykernel\_584/1064801598.py in type\_comp\_fit\_save\_model\_score(table, X\_train, y\_train, X\_test, y\_test, type\_model, save\_name, index\_model\_task)

217

218 # Обучение

--> 219 History = compile\_fit(model, X\_train, y\_train, validation\_split=0.1 ,early\_stopping=early\_stopping, model\_checkpoint=model\_checkpoint)

220

221 # Сохранение модели

/tmp/ipykernel\_584/1064801598.py in compile\_fit(model, X\_train, y\_train, X\_val, y\_val, validation\_split, early\_stopping, model\_checkpoint)

98 validation\_data = None,

99 validation\_split=validation\_split,

--> 100 callbacks=[model\_checkpoint, early\_stopping])

101 else:

102 history = model.fit(X\_train, y\_train,

/opt/conda/lib/python3.7/site-packages/keras/engine/training.py in fit(self, x, y, batch\_size, epochs, verbose, callbacks, validation\_split, validation\_data, shuffle, class\_weight, sample\_weight, initial\_epoch, steps\_per\_epoch, validation\_steps, validation\_batch\_size, validation\_freq, max\_queue\_size, workers, use\_multiprocessing)

1182 \_r=1):

1183 callbacks.on\_train\_batch\_begin(step)

-> 1184 tmp\_logs = self.train\_function(iterator)

1185 if data\_handler.should\_sync:

1186 context.async\_wait()

/opt/conda/lib/python3.7/site-packages/tensorflow/python/eager/def\_function.py in \_\_call\_\_(self, \*args, \*\*kwds)

883

884 with OptionalXlaContext(self.\_jit\_compile):

--> 885 result = self.\_call(\*args, \*\*kwds)

886

887 new\_tracing\_count = self.experimental\_get\_tracing\_count()

/opt/conda/lib/python3.7/site-packages/tensorflow/python/eager/def\_function.py in \_call(self, \*args, \*\*kwds)

915 # In this case we have created variables on the first call, so we run the

916 # defunned version which is guaranteed to never create variables.

--> 917 return self.\_stateless\_fn(\*args, \*\*kwds) # pylint: disable=not-callable

918 elif self.\_stateful\_fn is not None:

919 # Release the lock early so that multiple threads can perform the call

/opt/conda/lib/python3.7/site-packages/tensorflow/python/eager/function.py in \_\_call\_\_(self, \*args, \*\*kwargs)

3038 filtered\_flat\_args) = self.\_maybe\_define\_function(args, kwargs)

3039 return graph\_function.\_call\_flat(

-> 3040 filtered\_flat\_args, captured\_inputs=graph\_function.captured\_inputs) # pylint: disable=protected-access

3041

3042 @property

/opt/conda/lib/python3.7/site-packages/tensorflow/python/eager/function.py in \_call\_flat(self, args, captured\_inputs, cancellation\_manager)

1962 # No tape is watching; skip to running the function.

1963 return self.\_build\_call\_outputs(self.\_inference\_function.call(

-> 1964 ctx, args, cancellation\_manager=cancellation\_manager))

1965 forward\_backward = self.\_select\_forward\_and\_backward\_functions(

1966 args,

/opt/conda/lib/python3.7/site-packages/tensorflow/python/eager/function.py in call(self, ctx, args, cancellation\_manager)

594 inputs=args,

595 attrs=attrs,

--> 596 ctx=ctx)

597 else:

598 outputs = execute.execute\_with\_cancellation(

/opt/conda/lib/python3.7/site-packages/tensorflow/python/eager/execute.py in quick\_execute(op\_name, num\_outputs, inputs, attrs, ctx, name)

58 ctx.ensure\_initialized()

59 tensors = pywrap\_tfe.TFE\_Py\_Execute(ctx.\_handle, device\_name, op\_name,

---> 60 inputs, attrs, num\_outputs)

61 except core.\_NotOkStatusException as e:

62 if name is not None:

KeyboardInterrupt:

X\_train, y\_train, X\_test, y\_test = load\_ICBEB(task = 'form')

table\_res\_ICBEB = type\_comp\_fit\_save\_model\_score(table\_res\_ICBEB, X\_train, y\_train, X\_test, y\_test, 'lstm', 'lstm\_form.h5', 'lstm\_form')

del(X\_train)

del(y\_train)

del(X\_test)

del(y\_test)

Model: "sequential\_7"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

lstm\_14 (LSTM) (None, 1000, 256) 275456

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

leaky\_re\_lu\_14 (LeakyReLU) (None, 1000, 256) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

lstm\_15 (LSTM) (None, 256) 525312

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

leaky\_re\_lu\_15 (LeakyReLU) (None, 256) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_7 (Dense) (None, 3) 771

=================================================================

Total params: 801,539

Trainable params: 801,539

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

None

Epoch 1/30

43/43 [==============================] - 9s 164ms/step - loss: 1.0193 - auc: 0.6742 - val\_loss: 0.9669 - val\_auc: 0.6316

Epoch 2/30

43/43 [==============================] - 6s 148ms/step - loss: 0.9954 - auc: 0.6886 - val\_loss: 0.9251 - val\_auc: 0.7220

Epoch 3/30

43/43 [==============================] - 6s 148ms/step - loss: 0.9709 - auc: 0.7075 - val\_loss: 0.9731 - val\_auc: 0.7182

Epoch 4/30

43/43 [==============================] - 6s 148ms/step - loss: 0.9803 - auc: 0.7011 - val\_loss: 0.9328 - val\_auc: 0.7281

Epoch 5/30

43/43 [==============================] - 6s 150ms/step - loss: 0.9914 - auc: 0.6892 - val\_loss: 0.9348 - val\_auc: 0.7254

Epoch 6/30

43/43 [==============================] - 6s 148ms/step - loss: 0.9907 - auc: 0.6890 - val\_loss: 0.9402 - val\_auc: 0.7206

Epoch 7/30

43/43 [==============================] - 6s 147ms/step - loss: 0.9894 - auc: 0.6887 - val\_loss: 0.9355 - val\_auc: 0.7151

Epoch 8/30

43/43 [==============================] - 6s 148ms/step - loss: 0.9841 - auc: 0.6933 - val\_loss: 0.9326 - val\_auc: 0.7302

Epoch 9/30

43/43 [==============================] - 6s 148ms/step - loss: 0.9844 - auc: 0.6905 - val\_loss: 0.9330 - val\_auc: 0.7270

Epoch 10/30

43/43 [==============================] - 6s 149ms/step - loss: 0.9832 - auc: 0.6946 - val\_loss: 0.9309 - val\_auc: 0.7378

Epoch 11/30

43/43 [==============================] - 6s 148ms/step - loss: 0.9802 - auc: 0.6928 - val\_loss: 0.9303 - val\_auc: 0.7617

Epoch 12/30

43/43 [==============================] - 6s 149ms/step - loss: 0.9711 - auc: 0.7107 - val\_loss: 0.9293 - val\_auc: 0.7565

Epoch 13/30

43/43 [==============================] - 6s 148ms/step - loss: 0.9614 - auc: 0.7256 - val\_loss: 0.9371 - val\_auc: 0.7583

Epoch 14/30

43/43 [==============================] - 6s 148ms/step - loss: 0.9612 - auc: 0.7306 - val\_loss: 0.9303 - val\_auc: 0.7492

Epoch 15/30

43/43 [==============================] - 6s 149ms/step - loss: 0.9564 - auc: 0.7277 - val\_loss: 0.9040 - val\_auc: 0.7585

Epoch 16/30

43/43 [==============================] - 6s 149ms/step - loss: 0.9364 - auc: 0.7456 - val\_loss: 0.8721 - val\_auc: 0.7943

Epoch 17/30

43/43 [==============================] - 6s 149ms/step - loss: 0.9405 - auc: 0.7416 - val\_loss: 0.8841 - val\_auc: 0.7806

Epoch 18/30

43/43 [==============================] - 6s 148ms/step - loss: 0.9360 - auc: 0.7434 - val\_loss: 0.8851 - val\_auc: 0.7901

Epoch 19/30

43/43 [==============================] - 6s 148ms/step - loss: 0.9298 - auc: 0.7517 - val\_loss: 0.8851 - val\_auc: 0.7894

Epoch 20/30

43/43 [==============================] - 6s 149ms/step - loss: 0.9309 - auc: 0.7479 - val\_loss: 0.8561 - val\_auc: 0.7994

Epoch 21/30

43/43 [==============================] - 6s 148ms/step - loss: 0.9298 - auc: 0.7498 - val\_loss: 0.8543 - val\_auc: 0.8103

Epoch 22/30

43/43 [==============================] - 6s 148ms/step - loss: 0.9231 - auc: 0.7534 - val\_loss: 0.8496 - val\_auc: 0.8078

Epoch 23/30

43/43 [==============================] - 6s 148ms/step - loss: 0.9123 - auc: 0.7630 - val\_loss: 0.8565 - val\_auc: 0.7906

Epoch 24/30

43/43 [==============================] - 6s 148ms/step - loss: 0.9124 - auc: 0.7604 - val\_loss: 0.8472 - val\_auc: 0.7995

Epoch 25/30

43/43 [==============================] - 6s 149ms/step - loss: 0.9165 - auc: 0.7596 - val\_loss: 0.8770 - val\_auc: 0.7878

Epoch 26/30

43/43 [==============================] - 6s 148ms/step - loss: 0.9042 - auc: 0.7671 - val\_loss: 0.8823 - val\_auc: 0.7862

Epoch 27/30

43/43 [==============================] - 6s 148ms/step - loss: 0.9132 - auc: 0.7593 - val\_loss: 0.8563 - val\_auc: 0.8005

Epoch 28/30

43/43 [==============================] - 6s 148ms/step - loss: 0.9079 - auc: 0.7633 - val\_loss: 0.8542 - val\_auc: 0.7885

Epoch 29/30

43/43 [==============================] - 6s 149ms/step - loss: 0.8978 - auc: 0.7690 - val\_loss: 0.8848 - val\_auc: 0.7808

Epoch 30/30

43/43 [==============================] - 6s 150ms/step - loss: 0.9153 - auc: 0.7587 - val\_loss: 0.8539 - val\_auc: 0.7907

6/6 [==============================] - 0s 51ms/step - loss: 0.8775 - auc: 0.7839

/opt/conda/lib/python3.7/site-packages/sklearn/metrics/\_classification.py:1318: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

### lstm\_bidir

X\_train, y\_train, X\_test, y\_test = load\_ICBEB(task = 'diag')

table\_res\_ICBEB = type\_comp\_fit\_save\_model\_score(table\_res\_ICBEB, X\_train, y\_train, X\_test, y\_test, 'lstm\_bidir', 'lstm\_bidir\_diag.h5', 'lstm\_bidir\_diag')

del(X\_train)

del(y\_train)

del(X\_test)

del(y\_test)

Model: "sequential\_8"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

bidirectional (Bidirectional (None, 1000, 512) 550912

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

leaky\_re\_lu\_16 (LeakyReLU) (None, 1000, 512) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

bidirectional\_1 (Bidirection (None, 512) 1574912

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

leaky\_re\_lu\_17 (LeakyReLU) (None, 512) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_8 (Dense) (None, 4) 2052

=================================================================

Total params: 2,127,876

Trainable params: 2,127,876

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

None

Epoch 1/30

94/94 [==============================] - 35s 323ms/step - loss: 1.0149 - auc: 0.8140 - val\_loss: 0.7512 - val\_auc: 0.8986

Epoch 2/30

94/94 [==============================] - 29s 313ms/step - loss: 0.8346 - auc: 0.8800 - val\_loss: 0.8561 - val\_auc: 0.8373

Epoch 3/30

94/94 [==============================] - 29s 313ms/step - loss: 0.8866 - auc: 0.8562 - val\_loss: 0.7900 - val\_auc: 0.8829

Epoch 4/30

94/94 [==============================] - 29s 313ms/step - loss: 0.7469 - auc: 0.9027 - val\_loss: 0.7341 - val\_auc: 0.9141

Epoch 5/30

94/94 [==============================] - 29s 312ms/step - loss: 0.6412 - auc: 0.9288 - val\_loss: 0.7086 - val\_auc: 0.9178

Epoch 6/30

94/94 [==============================] - 29s 313ms/step - loss: 0.7362 - auc: 0.9033 - val\_loss: 0.7512 - val\_auc: 0.8866

Epoch 7/30

94/94 [==============================] - 29s 313ms/step - loss: 0.7115 - auc: 0.9074 - val\_loss: 0.6863 - val\_auc: 0.9177

Epoch 8/30

94/94 [==============================] - 29s 314ms/step - loss: 0.5995 - auc: 0.9375 - val\_loss: 0.6615 - val\_auc: 0.9244

Epoch 9/30

94/94 [==============================] - 29s 312ms/step - loss: 0.5406 - auc: 0.9484 - val\_loss: 0.5253 - val\_auc: 0.9533

Epoch 10/30

94/94 [==============================] - 29s 313ms/step - loss: 0.5154 - auc: 0.9538 - val\_loss: 0.5727 - val\_auc: 0.9449

Epoch 11/30

94/94 [==============================] - 29s 312ms/step - loss: 0.4686 - auc: 0.9618 - val\_loss: 0.4841 - val\_auc: 0.9587

Epoch 12/30

94/94 [==============================] - 29s 313ms/step - loss: 0.5894 - auc: 0.9419 - val\_loss: 1.1939 - val\_auc: 0.7439

Epoch 13/30

94/94 [==============================] - 29s 312ms/step - loss: 1.1680 - auc: 0.7467 - val\_loss: 1.1063 - val\_auc: 0.7773

Epoch 14/30

94/94 [==============================] - 29s 313ms/step - loss: 1.1160 - auc: 0.7735 - val\_loss: 1.0911 - val\_auc: 0.7841

Epoch 15/30

94/94 [==============================] - 29s 312ms/step - loss: 1.0703 - auc: 0.7897 - val\_loss: 0.9808 - val\_auc: 0.8212

Epoch 16/30

94/94 [==============================] - 29s 313ms/step - loss: 0.9227 - auc: 0.8481 - val\_loss: 0.8974 - val\_auc: 0.8663

Epoch 17/30

94/94 [==============================] - 29s 312ms/step - loss: 0.8108 - auc: 0.8860 - val\_loss: 0.7574 - val\_auc: 0.8944

Epoch 18/30

94/94 [==============================] - 29s 313ms/step - loss: 0.8625 - auc: 0.8672 - val\_loss: 0.8778 - val\_auc: 0.8663

Epoch 19/30

94/94 [==============================] - 29s 312ms/step - loss: 0.9015 - auc: 0.8546 - val\_loss: 0.8817 - val\_auc: 0.8574

Epoch 20/30

94/94 [==============================] - 29s 313ms/step - loss: 0.7406 - auc: 0.9058 - val\_loss: 0.7535 - val\_auc: 0.9024

Epoch 21/30

94/94 [==============================] - 29s 312ms/step - loss: 0.6646 - auc: 0.9234 - val\_loss: 0.6944 - val\_auc: 0.9104

Epoch 22/30

94/94 [==============================] - 29s 313ms/step - loss: 0.6189 - auc: 0.9337 - val\_loss: 0.6437 - val\_auc: 0.9294

Epoch 23/30

94/94 [==============================] - 29s 313ms/step - loss: 0.5879 - auc: 0.9398 - val\_loss: 0.5881 - val\_auc: 0.9397

Epoch 24/30

94/94 [==============================] - 29s 313ms/step - loss: 0.5541 - auc: 0.9460 - val\_loss: 0.6324 - val\_auc: 0.9381

Epoch 25/30

94/94 [==============================] - 29s 313ms/step - loss: 0.5422 - auc: 0.9474 - val\_loss: 0.5469 - val\_auc: 0.9461

Epoch 26/30

94/94 [==============================] - 30s 314ms/step - loss: 0.5090 - auc: 0.9537 - val\_loss: 0.5223 - val\_auc: 0.9510

12/12 [==============================] - 1s 116ms/step - loss: 0.5086 - auc: 0.9562

X\_train, y\_train, X\_test, y\_test = load\_ICBEB(task = 'superdiag')

table\_res\_ICBEB = type\_comp\_fit\_save\_model\_score(table\_res\_ICBEB, X\_train, y\_train, X\_test, y\_test, 'lstm\_bidir', 'lstm\_bidir\_superdiag.h5', 'lstm\_bidir\_superdiag')

del(X\_train)

del(y\_train)

del(X\_test)

del(y\_test)

Model: "sequential\_9"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

bidirectional\_2 (Bidirection (None, 1000, 512) 550912

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

leaky\_re\_lu\_18 (LeakyReLU) (None, 1000, 512) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

bidirectional\_3 (Bidirection (None, 512) 1574912

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

leaky\_re\_lu\_19 (LeakyReLU) (None, 512) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_9 (Dense) (None, 2) 1026

=================================================================

Total params: 2,126,850

Trainable params: 2,126,850

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

None

Epoch 1/30

94/94 [==============================] - 35s 325ms/step - loss: 0.5047 - auc: 0.8298 - val\_loss: 0.5245 - val\_auc: 0.8353

Epoch 2/30

94/94 [==============================] - 29s 312ms/step - loss: 0.5038 - auc: 0.8339 - val\_loss: 0.5392 - val\_auc: 0.8017

Epoch 3/30

94/94 [==============================] - 29s 313ms/step - loss: 0.5225 - auc: 0.8159 - val\_loss: 0.5463 - val\_auc: 0.8084

Epoch 4/30

94/94 [==============================] - 29s 312ms/step - loss: 0.5137 - auc: 0.8231 - val\_loss: 0.5171 - val\_auc: 0.8225

Epoch 5/30

94/94 [==============================] - 29s 313ms/step - loss: 0.4824 - auc: 0.8471 - val\_loss: 0.4627 - val\_auc: 0.8674

Epoch 6/30

94/94 [==============================] - 29s 313ms/step - loss: 0.4651 - auc: 0.8577 - val\_loss: 0.5023 - val\_auc: 0.8347

Epoch 7/30

94/94 [==============================] - 29s 314ms/step - loss: 0.4941 - auc: 0.8374 - val\_loss: 0.4775 - val\_auc: 0.8497

Epoch 8/30

94/94 [==============================] - 29s 313ms/step - loss: 0.4897 - auc: 0.8388 - val\_loss: 0.5747 - val\_auc: 0.7773

Epoch 9/30

94/94 [==============================] - 29s 313ms/step - loss: 0.5314 - auc: 0.8009 - val\_loss: 0.5493 - val\_auc: 0.7893

Epoch 10/30

94/94 [==============================] - 29s 313ms/step - loss: 0.5255 - auc: 0.8075 - val\_loss: 0.5066 - val\_auc: 0.8288

Epoch 11/30

94/94 [==============================] - 29s 313ms/step - loss: 0.5176 - auc: 0.8165 - val\_loss: 0.5413 - val\_auc: 0.8067

Epoch 12/30

94/94 [==============================] - 29s 313ms/step - loss: 0.5182 - auc: 0.8149 - val\_loss: 0.5274 - val\_auc: 0.8227

Epoch 13/30

94/94 [==============================] - 29s 313ms/step - loss: 0.5014 - auc: 0.8315 - val\_loss: 0.5027 - val\_auc: 0.8298

Epoch 14/30

94/94 [==============================] - 29s 313ms/step - loss: 0.4912 - auc: 0.8398 - val\_loss: 0.4844 - val\_auc: 0.8453

Epoch 15/30

94/94 [==============================] - 29s 314ms/step - loss: 0.4742 - auc: 0.8503 - val\_loss: 0.4992 - val\_auc: 0.8573

Epoch 16/30

94/94 [==============================] - 29s 313ms/step - loss: 0.4541 - auc: 0.8631 - val\_loss: 0.5202 - val\_auc: 0.7967

Epoch 17/30

94/94 [==============================] - 29s 313ms/step - loss: 0.4341 - auc: 0.8763 - val\_loss: 0.5006 - val\_auc: 0.8722

Epoch 18/30

94/94 [==============================] - 29s 312ms/step - loss: 0.4146 - auc: 0.8880 - val\_loss: 0.4437 - val\_auc: 0.8688

Epoch 19/30

94/94 [==============================] - 29s 313ms/step - loss: 0.3761 - auc: 0.9091 - val\_loss: 0.3883 - val\_auc: 0.9058

Epoch 20/30

94/94 [==============================] - 29s 313ms/step - loss: 0.3581 - auc: 0.9184 - val\_loss: 0.3506 - val\_auc: 0.9220

Epoch 21/30

94/94 [==============================] - 29s 313ms/step - loss: 0.3400 - auc: 0.9265 - val\_loss: 0.3616 - val\_auc: 0.9153

Epoch 22/30

94/94 [==============================] - 29s 313ms/step - loss: 0.3110 - auc: 0.9387 - val\_loss: 0.2981 - val\_auc: 0.9455

Epoch 23/30

94/94 [==============================] - 29s 314ms/step - loss: 0.2807 - auc: 0.9510 - val\_loss: 0.3280 - val\_auc: 0.9334

Epoch 24/30

94/94 [==============================] - 29s 313ms/step - loss: 0.2990 - auc: 0.9440 - val\_loss: 0.3196 - val\_auc: 0.9359

Epoch 25/30

94/94 [==============================] - 29s 313ms/step - loss: 0.2846 - auc: 0.9495 - val\_loss: 0.3031 - val\_auc: 0.9413

Epoch 26/30

94/94 [==============================] - 29s 313ms/step - loss: 0.2678 - auc: 0.9559 - val\_loss: 0.2608 - val\_auc: 0.9575

Epoch 27/30

94/94 [==============================] - 29s 313ms/step - loss: 0.2620 - auc: 0.9574 - val\_loss: 0.2774 - val\_auc: 0.9522

Epoch 28/30

94/94 [==============================] - 29s 312ms/step - loss: 0.2611 - auc: 0.9574 - val\_loss: 0.2832 - val\_auc: 0.9516

Epoch 29/30

94/94 [==============================] - 29s 314ms/step - loss: 0.2401 - auc: 0.9645 - val\_loss: 0.2588 - val\_auc: 0.9590

Epoch 30/30

94/94 [==============================] - 29s 313ms/step - loss: 0.2335 - auc: 0.9668 - val\_loss: 0.2612 - val\_auc: 0.9583

12/12 [==============================] - 1s 115ms/step - loss: 0.2817 - auc: 0.9522

X\_train, y\_train, X\_test, y\_test = load\_ICBEB(task = 'subdiag')

table\_res\_ICBEB = type\_comp\_fit\_save\_model\_score(table\_res\_ICBEB, X\_train, y\_train, X\_test, y\_test, 'lstm\_bidir', 'lstm\_bidir\_subdiag.h5', 'lstm\_bidir\_subdiag')

del(X\_train)

del(y\_train)

del(X\_test)

del(y\_test)

Model: "sequential\_10"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

bidirectional\_4 (Bidirection (None, 1000, 512) 550912

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

leaky\_re\_lu\_20 (LeakyReLU) (None, 1000, 512) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

bidirectional\_5 (Bidirection (None, 512) 1574912

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

leaky\_re\_lu\_21 (LeakyReLU) (None, 512) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_10 (Dense) (None, 4) 2052

=================================================================

Total params: 2,127,876

Trainable params: 2,127,876

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

None

Epoch 1/30

94/94 [==============================] - 35s 326ms/step - loss: 1.0287 - auc: 0.8072 - val\_loss: 0.8437 - val\_auc: 0.8726

Epoch 2/30

94/94 [==============================] - 29s 313ms/step - loss: 0.9858 - auc: 0.8309 - val\_loss: 0.8193 - val\_auc: 0.8845

Epoch 3/30

94/94 [==============================] - 29s 312ms/step - loss: 1.1419 - auc: 0.7524 - val\_loss: 1.1282 - val\_auc: 0.7514

Epoch 4/30

94/94 [==============================] - 29s 314ms/step - loss: 1.1022 - auc: 0.7640 - val\_loss: 1.0760 - val\_auc: 0.7948

Epoch 5/30

94/94 [==============================] - 29s 313ms/step - loss: 1.0551 - auc: 0.7910 - val\_loss: 1.0073 - val\_auc: 0.8168

Epoch 6/30

94/94 [==============================] - 29s 313ms/step - loss: 0.9632 - auc: 0.8316 - val\_loss: 0.9287 - val\_auc: 0.8397

Epoch 7/30

94/94 [==============================] - 29s 313ms/step - loss: 0.9184 - auc: 0.8486 - val\_loss: 0.8950 - val\_auc: 0.8499

Epoch 8/30

94/94 [==============================] - 29s 314ms/step - loss: 0.8716 - auc: 0.8622 - val\_loss: 0.7915 - val\_auc: 0.8846

Epoch 9/30

94/94 [==============================] - 29s 313ms/step - loss: 0.7962 - auc: 0.8846 - val\_loss: 0.8463 - val\_auc: 0.8752

Epoch 10/30

94/94 [==============================] - 30s 314ms/step - loss: 0.7718 - auc: 0.8930 - val\_loss: 0.6910 - val\_auc: 0.9160

Epoch 11/30

94/94 [==============================] - 29s 313ms/step - loss: 0.6764 - auc: 0.9192 - val\_loss: 0.7273 - val\_auc: 0.9020

Epoch 12/30

94/94 [==============================] - 29s 314ms/step - loss: 0.6367 - auc: 0.9294 - val\_loss: 0.6230 - val\_auc: 0.9332

Epoch 13/30

94/94 [==============================] - 29s 313ms/step - loss: 0.5996 - auc: 0.9373 - val\_loss: 0.6195 - val\_auc: 0.9326

Epoch 14/30

94/94 [==============================] - 29s 314ms/step - loss: 0.5522 - auc: 0.9458 - val\_loss: 0.5762 - val\_auc: 0.9409

Epoch 15/30

94/94 [==============================] - 29s 313ms/step - loss: 0.5230 - auc: 0.9503 - val\_loss: 0.5397 - val\_auc: 0.9490

Epoch 16/30

94/94 [==============================] - 29s 313ms/step - loss: 0.5148 - auc: 0.9525 - val\_loss: 0.5243 - val\_auc: 0.9508

Epoch 17/30

94/94 [==============================] - 29s 313ms/step - loss: 0.5148 - auc: 0.9523 - val\_loss: 0.5163 - val\_auc: 0.9533

Epoch 18/30

94/94 [==============================] - 29s 314ms/step - loss: 0.5107 - auc: 0.9531 - val\_loss: 0.4938 - val\_auc: 0.9578

Epoch 19/30

94/94 [==============================] - 29s 313ms/step - loss: 0.5091 - auc: 0.9538 - val\_loss: 0.4764 - val\_auc: 0.9608

Epoch 20/30

94/94 [==============================] - 29s 314ms/step - loss: 0.4889 - auc: 0.9571 - val\_loss: 0.4860 - val\_auc: 0.9580

Epoch 21/30

94/94 [==============================] - 29s 313ms/step - loss: 0.4767 - auc: 0.9588 - val\_loss: 0.4995 - val\_auc: 0.9549

Epoch 22/30

94/94 [==============================] - 29s 313ms/step - loss: 0.4837 - auc: 0.9581 - val\_loss: 0.5035 - val\_auc: 0.9554

Epoch 23/30

94/94 [==============================] - 29s 313ms/step - loss: 0.4785 - auc: 0.9585 - val\_loss: 0.5108 - val\_auc: 0.9524

Epoch 24/30

94/94 [==============================] - 29s 313ms/step - loss: 0.4713 - auc: 0.9606 - val\_loss: 0.4712 - val\_auc: 0.9597

Epoch 25/30

94/94 [==============================] - 29s 313ms/step - loss: 0.4579 - auc: 0.9623 - val\_loss: 0.5052 - val\_auc: 0.9550

Epoch 26/30

94/94 [==============================] - 29s 314ms/step - loss: 0.4637 - auc: 0.9618 - val\_loss: 0.4655 - val\_auc: 0.9595

Epoch 27/30

94/94 [==============================] - 29s 313ms/step - loss: 0.4532 - auc: 0.9634 - val\_loss: 0.5313 - val\_auc: 0.9501

Epoch 28/30

94/94 [==============================] - 29s 313ms/step - loss: 0.4557 - auc: 0.9632 - val\_loss: 0.4724 - val\_auc: 0.9596

Epoch 29/30

94/94 [==============================] - 29s 313ms/step - loss: 0.4429 - auc: 0.9656 - val\_loss: 0.4406 - val\_auc: 0.9632

Epoch 30/30

94/94 [==============================] - 29s 313ms/step - loss: 0.4154 - auc: 0.9694 - val\_loss: 0.4636 - val\_auc: 0.9610

12/12 [==============================] - 1s 115ms/step - loss: 0.5223 - auc: 0.9532

X\_train, y\_train, X\_test, y\_test = load\_ICBEB(task = 'rhythm')

table\_res\_ICBEB = type\_comp\_fit\_save\_model\_score(table\_res\_ICBEB, X\_train, y\_train, X\_test, y\_test, 'lstm\_bidir', 'lstm\_bidir\_rhythm.h5', 'lstm\_bidir\_rhythm')

del(X\_train)

del(y\_train)

del(X\_test)

del(y\_test)

Model: "sequential\_11"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

bidirectional\_6 (Bidirection (None, 1000, 512) 550912

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

leaky\_re\_lu\_22 (LeakyReLU) (None, 1000, 512) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

bidirectional\_7 (Bidirection (None, 512) 1574912

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

leaky\_re\_lu\_23 (LeakyReLU) (None, 512) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_11 (Dense) (None, 1) 513

=================================================================

Total params: 2,126,337

Trainable params: 2,126,337

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

None

Epoch 1/30

31/31 [==============================] - 15s 353ms/step - loss: 0.0000e+00 - auc: 0.0000e+00 - val\_loss: 0.0000e+00 - val\_auc: 0.0000e+00

Epoch 2/30

31/31 [==============================] - 10s 309ms/step - loss: 0.0000e+00 - auc: 0.0000e+00 - val\_loss: 0.0000e+00 - val\_auc: 0.0000e+00

Epoch 3/30

31/31 [==============================] - 10s 310ms/step - loss: 0.0000e+00 - auc: 0.0000e+00 - val\_loss: 0.0000e+00 - val\_auc: 0.0000e+00

Epoch 4/30

31/31 [==============================] - 10s 309ms/step - loss: 0.0000e+00 - auc: 0.0000e+00 - val\_loss: 0.0000e+00 - val\_auc: 0.0000e+00

Epoch 5/30

31/31 [==============================] - 10s 311ms/step - loss: 0.0000e+00 - auc: 0.0000e+00 - val\_loss: 0.0000e+00 - val\_auc: 0.0000e+00

Epoch 6/30

31/31 [==============================] - 10s 308ms/step - loss: 0.0000e+00 - auc: 0.0000e+00 - val\_loss: 0.0000e+00 - val\_auc: 0.0000e+00

Epoch 7/30

31/31 [==============================] - 10s 310ms/step - loss: 0.0000e+00 - auc: 0.0000e+00 - val\_loss: 0.0000e+00 - val\_auc: 0.0000e+00

Epoch 8/30

31/31 [==============================] - 10s 309ms/step - loss: 0.0000e+00 - auc: 0.0000e+00 - val\_loss: 0.0000e+00 - val\_auc: 0.0000e+00

Epoch 9/30

31/31 [==============================] - 10s 309ms/step - loss: 0.0000e+00 - auc: 0.0000e+00 - val\_loss: 0.0000e+00 - val\_auc: 0.0000e+00

Epoch 10/30

31/31 [==============================] - 10s 309ms/step - loss: 0.0000e+00 - auc: 0.0000e+00 - val\_loss: 0.0000e+00 - val\_auc: 0.0000e+00

Epoch 11/30

31/31 [==============================] - 10s 310ms/step - loss: 0.0000e+00 - auc: 0.0000e+00 - val\_loss: 0.0000e+00 - val\_auc: 0.0000e+00

Epoch 12/30

31/31 [==============================] - 10s 309ms/step - loss: 0.0000e+00 - auc: 0.0000e+00 - val\_loss: 0.0000e+00 - val\_auc: 0.0000e+00

Epoch 13/30

31/31 [==============================] - 10s 310ms/step - loss: 0.0000e+00 - auc: 0.0000e+00 - val\_loss: 0.0000e+00 - val\_auc: 0.0000e+00

Epoch 14/30

31/31 [==============================] - 10s 309ms/step - loss: 0.0000e+00 - auc: 0.0000e+00 - val\_loss: 0.0000e+00 - val\_auc: 0.0000e+00

Epoch 15/30

31/31 [==============================] - 10s 310ms/step - loss: 0.0000e+00 - auc: 0.0000e+00 - val\_loss: 0.0000e+00 - val\_auc: 0.0000e+00

Epoch 16/30

31/31 [==============================] - 10s 309ms/step - loss: 0.0000e+00 - auc: 0.0000e+00 - val\_loss: 0.0000e+00 - val\_auc: 0.0000e+00

4/4 [==============================] - 0s 108ms/step - loss: 0.0000e+00 - auc: 0.0000e+00

X\_train, y\_train, X\_test, y\_test = load\_ICBEB(task = 'form')

table\_res\_ICBEB = type\_comp\_fit\_save\_model\_score(table\_res\_ICBEB, X\_train, y\_train, X\_test, y\_test, 'lstm\_bidir', 'lstm\_bidir\_form.h5', 'lstm\_bidir\_form')

del(X\_train)

del(y\_train)

del(X\_test)

del(y\_test)

Model: "sequential\_12"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

bidirectional\_8 (Bidirection (None, 1000, 512) 550912

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

leaky\_re\_lu\_24 (LeakyReLU) (None, 1000, 512) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

bidirectional\_9 (Bidirection (None, 512) 1574912

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

leaky\_re\_lu\_25 (LeakyReLU) (None, 512) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_12 (Dense) (None, 3) 1539

=================================================================

Total params: 2,127,363

Trainable params: 2,127,363

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

None

Epoch 1/30

43/43 [==============================] - 19s 337ms/step - loss: 0.9974 - auc: 0.6963 - val\_loss: 1.0137 - val\_auc: 0.6779

Epoch 2/30

43/43 [==============================] - 13s 311ms/step - loss: 0.9217 - auc: 0.7443 - val\_loss: 0.8346 - val\_auc: 0.8006

Epoch 3/30

43/43 [==============================] - 13s 314ms/step - loss: 0.8214 - auc: 0.8022 - val\_loss: 0.8472 - val\_auc: 0.7780

Epoch 4/30

43/43 [==============================] - 13s 312ms/step - loss: 0.8305 - auc: 0.8007 - val\_loss: 0.8795 - val\_auc: 0.7936

Epoch 5/30

43/43 [==============================] - 13s 314ms/step - loss: 0.8317 - auc: 0.8048 - val\_loss: 0.8894 - val\_auc: 0.7822

Epoch 6/30

43/43 [==============================] - 13s 312ms/step - loss: 0.8048 - auc: 0.8156 - val\_loss: 0.8016 - val\_auc: 0.8136

Epoch 7/30

43/43 [==============================] - 13s 314ms/step - loss: 0.7317 - auc: 0.8509 - val\_loss: 0.6475 - val\_auc: 0.8793

Epoch 8/30

43/43 [==============================] - 13s 313ms/step - loss: 0.8344 - auc: 0.8010 - val\_loss: 0.7488 - val\_auc: 0.8311

Epoch 9/30

43/43 [==============================] - 14s 315ms/step - loss: 0.7299 - auc: 0.8511 - val\_loss: 0.6812 - val\_auc: 0.8702

Epoch 10/30

43/43 [==============================] - 13s 314ms/step - loss: 0.7604 - auc: 0.8389 - val\_loss: 0.8210 - val\_auc: 0.8176

Epoch 11/30

43/43 [==============================] - 14s 315ms/step - loss: 0.7655 - auc: 0.8402 - val\_loss: 0.7322 - val\_auc: 0.8448

Epoch 12/30

43/43 [==============================] - 13s 313ms/step - loss: 0.7089 - auc: 0.8638 - val\_loss: 0.7763 - val\_auc: 0.8128

Epoch 13/30

43/43 [==============================] - 14s 315ms/step - loss: 0.7311 - auc: 0.8525 - val\_loss: 0.9148 - val\_auc: 0.7206

Epoch 14/30

43/43 [==============================] - 13s 313ms/step - loss: 0.9411 - auc: 0.7360 - val\_loss: 0.9203 - val\_auc: 0.7500

Epoch 15/30

43/43 [==============================] - 14s 315ms/step - loss: 0.8515 - auc: 0.7970 - val\_loss: 0.8853 - val\_auc: 0.7852

Epoch 16/30

43/43 [==============================] - 13s 313ms/step - loss: 0.8068 - auc: 0.8178 - val\_loss: 0.7635 - val\_auc: 0.8339

Epoch 17/30

43/43 [==============================] - 13s 313ms/step - loss: 0.7691 - auc: 0.8380 - val\_loss: 0.9766 - val\_auc: 0.7384

Epoch 18/30

43/43 [==============================] - 14s 314ms/step - loss: 0.7759 - auc: 0.8313 - val\_loss: 0.9148 - val\_auc: 0.7555

Epoch 19/30

43/43 [==============================] - 13s 314ms/step - loss: 0.8277 - auc: 0.8080 - val\_loss: 0.7965 - val\_auc: 0.8208

Epoch 20/30

43/43 [==============================] - 14s 315ms/step - loss: 0.7695 - auc: 0.8331 - val\_loss: 0.8759 - val\_auc: 0.7869

Epoch 21/30

43/43 [==============================] - 13s 313ms/step - loss: 0.7006 - auc: 0.8644 - val\_loss: 0.8223 - val\_auc: 0.8158

Epoch 22/30

43/43 [==============================] - 13s 314ms/step - loss: 0.6560 - auc: 0.8808 - val\_loss: 0.8139 - val\_auc: 0.8211

6/6 [==============================] - 1s 112ms/step - loss: 0.6868 - auc: 0.8705

### Сохранение результатов в формат .csv

table\_res\_ICBEB.to\_csv('table\_res\_ICBEB.csv')

table\_res\_ICBEB = pd.DataFrame('table\_res\_ICBEB.csv')

# Подключение основных библиотек и загрузка данных

### Для Google Colaboratory

*# Подключение Google Drive к виртуальной машине*

from google.colab import drive

drive.mount('/content/drive')

*# Копирование данных с Google Drive на локальный диск виртуальной машины.*

*#!cp -r /content/drive/MyDrive/practice\_2022-2023/data/ICBEBnpy/ .*

!cp -r /content/drive/MyDrive/work\_with\_ptbxl/data/ptbxlnpy/ .

Mounted at /content/drive

### Подключение пакетов

*# Для работы с данными*

import pandas as pd

import numpy as np

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt *# plotting*

import seaborn as sns *# plotting heatmap*

*# Для работы с моделями*

import tensorflow as tf

from tensorflow import keras

from keras import layers

*# Для метрик*

from keras import backend as K

from keras.metrics import AUC, Recall, Precision, Accuracy, TruePositives, TrueNegatives, FalsePositives, FalseNegatives

from sklearn.metrics import fbeta\_score, precision\_score, recall\_score, accuracy\_score, roc\_auc\_score

from sklearn.metrics import auc, roc\_curve

*# Функции*

*# Загрузка ICBEB*

def load\_ICBEB(task):

if task == 'diag':

X\_train = np.load('ICBEBnpy/X\_train\_ICBEB\_diag.npy')

y\_train = np.load('ICBEBnpy/y\_train\_ICBEB\_diag.npy')

X\_test = np.load('ICBEBnpy/X\_val\_ICBEB\_diag.npy')

y\_test = np.load('ICBEBnpy/y\_val\_ICBEB\_diag.npy')

elif task == 'superdiag':

X\_train = np.load('ICBEBnpy/X\_train\_ICBEB\_superdiag.npy')

y\_train = np.load('ICBEBnpy/y\_train\_ICBEB\_superdiag.npy')

X\_test = np.load('ICBEBnpy/X\_val\_ICBEB\_superdiag.npy')

y\_test = np.load('ICBEBnpy/y\_val\_ICBEB\_superdiag.npy')

elif task == 'subdiag':

X\_train = np.load('ICBEBnpy/X\_train\_ICBEB\_subdiag.npy')

y\_train = np.load('ICBEBnpy/y\_train\_ICBEB\_subdiag.npy')

X\_test = np.load('ICBEBnpy/X\_val\_ICBEB\_subdiag.npy')

y\_test = np.load('ICBEBnpy/y\_val\_ICBEB\_subdiag.npy')

elif task == 'rhythm':

X\_train = np.load('ICBEBnpy/X\_train\_ICBEB\_rhythm.npy')

y\_train = np.load('ICBEBnpy/y\_train\_ICBEB\_rhythm.npy')

X\_test = np.load('ICBEBnpy/X\_val\_ICBEB\_rhythm.npy')

y\_test = np.load('ICBEBnpy/y\_val\_ICBEB\_rhythm.npy')

elif task == 'form':

X\_train = np.load('ICBEBnpy/X\_train\_ICBEB\_form.npy')

y\_train = np.load('ICBEBnpy/y\_train\_ICBEB\_form.npy')

X\_test = np.load('ICBEBnpy/X\_val\_ICBEB\_form.npy')

y\_test = np.load('ICBEBnpy/y\_val\_ICBEB\_form.npy')

*#print(X\_train.shape, y\_train.shape, X\_test.shape, y\_test.shape)*

return X\_train, y\_train, X\_test, y\_test

*# Загрузка ptbxl*

def load\_ptbxl(task):

if task == 'diag':

X\_train = np.load('ptbxlnpy/X\_train\_ptbxl\_diag.npy')

y\_train = np.load('ptbxlnpy/y\_train\_ptbxl\_diag.npy')

X\_test = np.load('ptbxlnpy/X\_val\_ptbxl\_diag.npy')

y\_test = np.load('ptbxlnpy/y\_val\_ptbxl\_diag.npy')

elif task == 'superdiag':

X\_train = np.load('ptbxlnpy/X\_train\_ptbxl\_superdiag.npy')

y\_train = np.load('ptbxlnpy/y\_train\_ptbxl\_superdiag.npy')

X\_test = np.load('ptbxlnpy/X\_val\_ptbxl\_superdiag.npy')

y\_test = np.load('ptbxlnpy/y\_val\_ptbxl\_superdiag.npy')

elif task == 'subdiag':

X\_train = np.load('ptbxlnpy/X\_train\_ptbxl\_subdiag.npy')

y\_train = np.load('ptbxlnpy/y\_train\_ptbxl\_subdiag.npy')

X\_test = np.load('ptbxlnpy/X\_val\_ptbxl\_subdiag.npy')

y\_test = np.load('ptbxlnpy/y\_val\_ptbxl\_subdiag.npy')

elif task == 'rhythm':

X\_train = np.load('ptbxlnpy/X\_train\_ptbxl\_rhythm.npy')

y\_train = np.load('ptbxlnpy/y\_train\_ptbxl\_rhythm.npy')

X\_test = np.load('ptbxlnpy/X\_val\_ptbxl\_rhythm.npy')

y\_test = np.load('ptbxlnpy/y\_val\_ptbxl\_rhythm.npy')

elif task == 'form':

X\_train = np.load('ptbxlnpy/X\_train\_ptbxl\_form.npy')

y\_train = np.load('ptbxlnpy/y\_train\_ptbxl\_form.npy')

X\_test = np.load('ptbxlnpy/X\_val\_ptbxl\_form.npy')

y\_test = np.load('ptbxlnpy/y\_val\_ptbxl\_form.npy')

*#print(X\_train.shape, y\_train.shape, X\_test.shape, y\_test.shape)*

return X\_train, y\_train, X\_test, y\_test

*# Компиляция и обучение модели*

def AUC\_Keras(y\_true, y\_pred):

auc = keras.metrics.AUC(y\_true, y\_pred)[1]

K.get\_session().run(tf.local\_variables\_initializer())

return auc

*# Компиляция и обучение модели*

def compile\_fit(model, X\_train, y\_train, X\_val = None, y\_val = None, validation\_split = 0.0, early\_stopping = None, model\_checkpoint = None):

model.compile(loss = keras.losses.CategoricalCrossentropy(),

optimizer=tf.optimizers.Adam(),

metrics=['AUC'])

if X\_val == None:

history = model.fit(X\_train, y\_train,

epochs = 30,

validation\_data = None,

validation\_split=validation\_split,

callbacks=[model\_checkpoint, early\_stopping])

else:

history = model.fit(X\_train, y\_train,

epochs = 30,

validation\_data = (X\_val, y\_val),

validation\_split=0.0,

callbacks=[model\_checkpoint, early\_stopping])

return history

*# TP TN FP FN*

def tp\_tn\_fp\_fn(y\_true, y\_pred):

TP = TruePositives()

TN = TrueNegatives()

FP = FalsePositives()

FN = FalseNegatives()

TP.update\_state(y\_true, y\_pred)

TN.update\_state(y\_true, y\_pred)

FP.update\_state(y\_true, y\_pred)

FN.update\_state(y\_true, y\_pred)

return TP.result().numpy(), TN.result().numpy(), FP.result().numpy(), FN.result().numpy()

*# Подсчет метрик*

def calc\_metrics(t, p, flag = 0): *# t - y\_true, p - y\_pred*

y\_true=np.argmax(t, axis=1)

y\_pred=np.argmax(p, axis=1)

beta = 2

f2\_score = fbeta\_score(y\_true, y\_pred, average='macro', beta=2)

precision = precision\_score(y\_true, y\_pred, average='macro')

recall = recall\_score(y\_true, y\_pred, average='macro')

TP, TN, FP, FN = tp\_tn\_fp\_fn(t, p)

g2\_score = TP/(TP+FP+beta\*FN)

if flag == 0:

return f2\_score, g2\_score

elif flag == 1:

return f2\_score, g2\_score, precision, recall

*#return f2\_score, g2\_score, AUC\_sklearn*

*# Таблица результатов*

table\_res\_ptbxl = pd.DataFrame(columns = ('AUC', 'F2', 'G2'))

*# Занесение новых результатов в таблицу*

def edit\_table(table, model, X, y, index\_name): *# X - X\_test, y - y\_test*

score = model.evaluate(X, y)

y\_pr = model.predict(X) *# y\_pr - y\_test\_pred*

f2\_score, g2\_score = calc\_metrics(y, y\_pr, flag = 0)

list\_metrics = [score[1], f2\_score, g2\_score]

table.loc[index\_name] = list\_metrics

return table

*# График loss и accuracy*

def plot\_loss\_and\_accuracy\_curves(\_history):

fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(18,6))

axs[0].plot(\_history.history['loss'], color='b', label='Training loss')

axs[0].plot(\_history.history['val\_loss'], color='r', label='Validation loss')

axs[0].set\_title("Loss curves")

axs[0].legend(loc='best', shadow=True)

axs[1].plot(\_history.history['auc'], color='b', label='Training accuracy')

axs[1].plot(\_history.history['val\_auc'], color='r', label='Validation accuracy')

axs[1].set\_title("Accuracy curves")

axs[1].legend(loc='best', shadow=True)

plt.show()

*# Работа с моделями lstm и lstm\_bidir*

def type\_comp\_fit\_save\_model\_score(table, X\_train, y\_train, X\_test, y\_test, type\_model, save\_name, index\_model\_task):

*# Уточняю количество классов*

num\_classes = y\_train.shape[1]

*# Выбор архитектуры модели*

if type\_model == 'lstm':

model = keras.Sequential()

model.add(layers.LSTM(input\_shape=(1000, 12), units=256,

return\_sequences=True,

stateful=False, unroll=False

))

model.add(layers.LeakyReLU())

model.add(layers.LSTM(units=256,

return\_sequences=False,

stateful=False, unroll=False

))

model.add(layers.LeakyReLU())

model.add(layers.Dense(units=num\_classes, activation='softmax'))

print(model.summary())

*# Реализация раннего прекращения.*

checkpoint\_filepath = './checkpoint\_lstm/'

model\_checkpoint = tf.keras.callbacks.ModelCheckpoint(filepath=checkpoint\_filepath,

save\_weights\_only=True,

save\_best\_only=True)

early\_stopping = keras.callbacks.EarlyStopping(patience=15,

restore\_best\_weights=True)

elif type\_model == 'lstm\_bidir':

model = keras.Sequential()

model.add(layers.Bidirectional(layers.LSTM(input\_shape=(1000, 12), units=256,

return\_sequences=True,

stateful=False, unroll=False

)))

model.add(layers.LeakyReLU())

model.add(layers.Bidirectional(layers.LSTM(units=256,

return\_sequences=False,

stateful=False, unroll=False

)))

model.add(layers.LeakyReLU())

model.add(layers.Dense(units=num\_classes, activation='softmax'))

model.build(input\_shape = (None, 1000, 12)) *# `input\_shape` is the shape of the input data*

print(model.summary())

*# Реализация раннего прекращения.*

checkpoint\_filepath = './checkpoint\_lstm\_bidir/'

model\_checkpoint = tf.keras.callbacks.ModelCheckpoint(filepath=checkpoint\_filepath,

save\_weights\_only=True,

save\_best\_only=True)

early\_stopping = keras.callbacks.EarlyStopping(patience=15,

restore\_best\_weights=True)

*# Обучение*

History = compile\_fit(model, X\_train, y\_train, validation\_split=0.1 ,early\_stopping=early\_stopping, model\_checkpoint=model\_checkpoint)

*# Сохранение модели*

model.save(save\_name)

*# Построение графика*

plot\_loss\_and\_accuracy\_curves(History)

*# Сохранение в таблицу*

table = edit\_table(table, model, X\_test, y\_test, index\_model\_task)

return table

tf.random.set\_seed(42)

%matplotlib inline

# Работа с lstm и lstm\_bidir

X\_train, y\_train, X\_test, y\_test = load\_ptbxl(task = 'diag')

table\_res\_ptbxl = type\_comp\_fit\_save\_model\_score(table\_res\_ptbxl, X\_train, y\_train, X\_test, y\_test, 'lstm', 'lstm\_diag.hdf5', 'lstm\_diag')

del(X\_train)

del(y\_train)

del(X\_test)

del(y\_test)

Model: "sequential"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

lstm (LSTM) (None, 1000, 256) 275456

leaky\_re\_lu (LeakyReLU) (None, 1000, 256) 0

lstm\_1 (LSTM) (None, 256) 525312

leaky\_re\_lu\_1 (LeakyReLU) (None, 256) 0

dense (Dense) (None, 44) 11308

=================================================================

Total params: 812,076

Trainable params: 812,076

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

None

Epoch 1/30

542/542 [==============================] - 63s 103ms/step - loss: 4.2575 - auc: 0.8553 - val\_loss: 5.0880 - val\_auc: 0.8381

Epoch 2/30

542/542 [==============================] - 56s 103ms/step - loss: 4.5957 - auc: 0.8579 - val\_loss: 5.4809 - val\_auc: 0.8335

Epoch 3/30

542/542 [==============================] - 56s 104ms/step - loss: 4.8938 - auc: 0.8576 - val\_loss: 5.7750 - val\_auc: 0.8408

Epoch 4/30

542/542 [==============================] - 57s 105ms/step - loss: 5.2069 - auc: 0.8577 - val\_loss: 6.2008 - val\_auc: 0.8422

Epoch 5/30

542/542 [==============================] - 57s 106ms/step - loss: 5.4956 - auc: 0.8567 - val\_loss: 6.5534 - val\_auc: 0.8437

Epoch 6/30

542/542 [==============================] - 57s 105ms/step - loss: 5.7589 - auc: 0.8578 - val\_loss: 6.9948 - val\_auc: 0.8384

Epoch 7/30

542/542 [==============================] - 57s 105ms/step - loss: 6.0457 - auc: 0.8570 - val\_loss: 7.3638 - val\_auc: 0.8318

Epoch 8/30

542/542 [==============================] - 57s 106ms/step - loss: 6.3623 - auc: 0.8573 - val\_loss: 7.7835 - val\_auc: 0.8385

Epoch 9/30

542/542 [==============================] - 57s 106ms/step - loss: 6.6288 - auc: 0.8585 - val\_loss: 8.1961 - val\_auc: 0.8410

Epoch 10/30

542/542 [==============================] - 57s 105ms/step - loss: 6.9449 - auc: 0.8572 - val\_loss: 8.5792 - val\_auc: 0.8317

Epoch 11/30

542/542 [==============================] - 57s 105ms/step - loss: 7.2527 - auc: 0.8577 - val\_loss: 9.0117 - val\_auc: 0.8390

Epoch 12/30

542/542 [==============================] - 57s 105ms/step - loss: 7.5303 - auc: 0.8570 - val\_loss: 9.2587 - val\_auc: 0.8380

Epoch 13/30

542/542 [==============================] - 57s 105ms/step - loss: 7.8095 - auc: 0.8578 - val\_loss: 9.6228 - val\_auc: 0.8350

Epoch 14/30

542/542 [==============================] - 57s 106ms/step - loss: 8.0828 - auc: 0.8578 - val\_loss: 10.1512 - val\_auc: 0.8338

Epoch 15/30

542/542 [==============================] - 57s 105ms/step - loss: 8.3761 - auc: 0.8576 - val\_loss: 10.4741 - val\_auc: 0.8390

Epoch 16/30

542/542 [==============================] - 57s 105ms/step - loss: 8.6825 - auc: 0.8574 - val\_loss: 10.9069 - val\_auc: 0.8354

68/68 [==============================] - 3s 42ms/step - loss: 4.4292 - auc: 0.8577

68/68 [==============================] - 3s 39ms/step

/usr/local/lib/python3.8/dist-packages/sklearn/metrics/\_classification.py:1318: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

X\_train, y\_train, X\_test, y\_test = load\_ptbxl(task = 'superdiag')

table\_res\_ptbxl = type\_comp\_fit\_save\_model\_score(table\_res\_ptbxl, X\_train, y\_train, X\_test, y\_test, 'lstm', 'lstm\_superdiag.hdf5', 'lstm\_superdiag')

del(X\_train)

del(y\_train)

del(X\_test)

del(y\_test)

Model: "sequential\_1"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

lstm\_2 (LSTM) (None, 1000, 256) 275456

leaky\_re\_lu\_2 (LeakyReLU) (None, 1000, 256) 0

lstm\_3 (LSTM) (None, 256) 525312

leaky\_re\_lu\_3 (LeakyReLU) (None, 256) 0

dense\_1 (Dense) (None, 5) 1285

=================================================================

Total params: 802,053

Trainable params: 802,053

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

None

Epoch 1/30

542/542 [==============================] - 62s 109ms/step - loss: 1.9986 - auc: 0.6413 - val\_loss: 2.1267 - val\_auc: 0.5846

Epoch 2/30

542/542 [==============================] - 57s 106ms/step - loss: 1.9947 - auc: 0.6378 - val\_loss: 2.1087 - val\_auc: 0.5939

Epoch 3/30

542/542 [==============================] - 57s 106ms/step - loss: 1.9966 - auc: 0.6404 - val\_loss: 2.0922 - val\_auc: 0.6070

Epoch 4/30

542/542 [==============================] - 57s 105ms/step - loss: 1.9934 - auc: 0.6379 - val\_loss: 2.1153 - val\_auc: 0.5939

Epoch 5/30

542/542 [==============================] - 57s 106ms/step - loss: 1.9946 - auc: 0.6400 - val\_loss: 2.0949 - val\_auc: 0.6023

Epoch 6/30

542/542 [==============================] - 57s 105ms/step - loss: 1.9930 - auc: 0.6406 - val\_loss: 2.0966 - val\_auc: 0.6047

Epoch 7/30

542/542 [==============================] - 57s 105ms/step - loss: 1.9923 - auc: 0.6379 - val\_loss: 2.0966 - val\_auc: 0.5845

Epoch 8/30

542/542 [==============================] - 57s 105ms/step - loss: 1.9969 - auc: 0.6404 - val\_loss: 2.1155 - val\_auc: 0.5939

Epoch 9/30

542/542 [==============================] - 58s 106ms/step - loss: 1.9947 - auc: 0.6406 - val\_loss: 2.1121 - val\_auc: 0.6070

Epoch 10/30

542/542 [==============================] - 57s 105ms/step - loss: 1.9990 - auc: 0.6383 - val\_loss: 2.1224 - val\_auc: 0.6047

Epoch 11/30

542/542 [==============================] - 57s 105ms/step - loss: 1.9942 - auc: 0.6392 - val\_loss: 2.1383 - val\_auc: 0.6070

Epoch 12/30

542/542 [==============================] - 57s 105ms/step - loss: 1.9945 - auc: 0.6403 - val\_loss: 2.1034 - val\_auc: 0.6070

Epoch 13/30

542/542 [==============================] - 57s 105ms/step - loss: 1.9949 - auc: 0.6391 - val\_loss: 2.0970 - val\_auc: 0.6070

Epoch 14/30

542/542 [==============================] - 57s 105ms/step - loss: 1.9958 - auc: 0.6388 - val\_loss: 2.1289 - val\_auc: 0.5939

Epoch 15/30

542/542 [==============================] - 57s 105ms/step - loss: 1.9931 - auc: 0.6393 - val\_loss: 2.1162 - val\_auc: 0.6070

Epoch 16/30

542/542 [==============================] - 57s 105ms/step - loss: 1.9932 - auc: 0.6393 - val\_loss: 2.1034 - val\_auc: 0.6070

Epoch 17/30

542/542 [==============================] - 57s 106ms/step - loss: 1.9957 - auc: 0.6400 - val\_loss: 2.1304 - val\_auc: 0.5715

Epoch 18/30

542/542 [==============================] - 57s 105ms/step - loss: 1.9955 - auc: 0.6390 - val\_loss: 2.1858 - val\_auc: 0.6023

68/68 [==============================] - 3s 43ms/step - loss: 1.9873 - auc: 0.6404

68/68 [==============================] - 3s 40ms/step

/usr/local/lib/python3.8/dist-packages/sklearn/metrics/\_classification.py:1318: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

X\_train, y\_train, X\_test, y\_test = load\_ptbxl(task = 'subdiag')

table\_res\_ptbxl = type\_comp\_fit\_save\_model\_score(table\_res\_ptbxl, X\_train, y\_train, X\_test, y\_test, 'lstm', 'lstm\_subdiag.hdf5', 'lstm\_subdiag')

del(X\_train)

del(y\_train)

del(X\_test)

del(y\_test)

Model: "sequential\_2"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

lstm\_4 (LSTM) (None, 1000, 256) 275456

leaky\_re\_lu\_4 (LeakyReLU) (None, 1000, 256) 0

lstm\_5 (LSTM) (None, 256) 525312

leaky\_re\_lu\_5 (LeakyReLU) (None, 256) 0

dense\_2 (Dense) (None, 23) 5911

=================================================================

Total params: 806,679

Trainable params: 806,679

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

None

Epoch 1/30

542/542 [==============================] - 61s 108ms/step - loss: 3.6279 - auc: 0.8247 - val\_loss: 4.1533 - val\_auc: 0.8143

Epoch 2/30

542/542 [==============================] - 57s 106ms/step - loss: 3.8453 - auc: 0.8263 - val\_loss: 4.5011 - val\_auc: 0.8061

Epoch 3/30

542/542 [==============================] - 57s 106ms/step - loss: 4.0762 - auc: 0.8261 - val\_loss: 4.7733 - val\_auc: 0.8083

Epoch 4/30

542/542 [==============================] - 57s 105ms/step - loss: 4.2833 - auc: 0.8257 - val\_loss: 5.0677 - val\_auc: 0.8103

Epoch 5/30

542/542 [==============================] - 57s 105ms/step - loss: 4.4988 - auc: 0.8259 - val\_loss: 5.2999 - val\_auc: 0.8133

Epoch 6/30

542/542 [==============================] - 57s 105ms/step - loss: 4.6666 - auc: 0.8265 - val\_loss: 5.6302 - val\_auc: 0.8128

Epoch 7/30

542/542 [==============================] - 57s 105ms/step - loss: 4.8771 - auc: 0.8263 - val\_loss: 5.9476 - val\_auc: 0.8069

Epoch 8/30

542/542 [==============================] - 57s 105ms/step - loss: 5.1239 - auc: 0.8261 - val\_loss: 6.2685 - val\_auc: 0.8050

Epoch 9/30

542/542 [==============================] - 57s 106ms/step - loss: 5.3146 - auc: 0.8265 - val\_loss: 6.5414 - val\_auc: 0.8096

Epoch 10/30

542/542 [==============================] - 57s 105ms/step - loss: 5.5739 - auc: 0.8257 - val\_loss: 6.8170 - val\_auc: 0.8120

Epoch 11/30

542/542 [==============================] - 57s 105ms/step - loss: 5.7611 - auc: 0.8268 - val\_loss: 7.1648 - val\_auc: 0.8078

Epoch 12/30

542/542 [==============================] - 57s 105ms/step - loss: 5.9699 - auc: 0.8261 - val\_loss: 7.3709 - val\_auc: 0.8133

Epoch 13/30

542/542 [==============================] - 57s 105ms/step - loss: 6.1642 - auc: 0.8264 - val\_loss: 7.6141 - val\_auc: 0.8093

Epoch 14/30

542/542 [==============================] - 57s 105ms/step - loss: 6.3594 - auc: 0.8270 - val\_loss: 8.0168 - val\_auc: 0.8038

Epoch 15/30

542/542 [==============================] - 57s 105ms/step - loss: 6.5787 - auc: 0.8261 - val\_loss: 8.2171 - val\_auc: 0.8111

Epoch 16/30

542/542 [==============================] - 57s 105ms/step - loss: 6.7734 - auc: 0.8269 - val\_loss: 8.5295 - val\_auc: 0.8097

68/68 [==============================] - 3s 42ms/step - loss: 3.7177 - auc: 0.8280

68/68 [==============================] - 3s 40ms/step

/usr/local/lib/python3.8/dist-packages/sklearn/metrics/\_classification.py:1318: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

X\_train, y\_train, X\_test, y\_test = load\_ptbxl(task = 'rhythm')

table\_res\_ptbxl = type\_comp\_fit\_save\_model\_score(table\_res\_ptbxl, X\_train, y\_train, X\_test, y\_test, 'lstm', 'lstm\_rhythm.hdf5', 'lstm\_rhythm')

del(X\_train)

del(y\_train)

del(X\_test)

del(y\_test)

Model: "sequential\_3"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

lstm\_6 (LSTM) (None, 1000, 256) 275456

leaky\_re\_lu\_6 (LeakyReLU) (None, 1000, 256) 0

lstm\_7 (LSTM) (None, 256) 525312

leaky\_re\_lu\_7 (LeakyReLU) (None, 256) 0

dense\_3 (Dense) (None, 12) 3084

=================================================================

Total params: 803,852

Trainable params: 803,852

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

None

Epoch 1/30

534/534 [==============================] - 63s 112ms/step - loss: 0.8963 - auc: 0.9458 - val\_loss: 1.0778 - val\_auc: 0.9321

Epoch 2/30

534/534 [==============================] - 57s 107ms/step - loss: 0.8720 - auc: 0.9486 - val\_loss: 1.0346 - val\_auc: 0.9384

Epoch 3/30

534/534 [==============================] - 58s 108ms/step - loss: 0.8775 - auc: 0.9479 - val\_loss: 1.0745 - val\_auc: 0.9308

Epoch 4/30

534/534 [==============================] - 57s 107ms/step - loss: 0.8789 - auc: 0.9470 - val\_loss: 1.0627 - val\_auc: 0.9321

Epoch 5/30

534/534 [==============================] - 58s 108ms/step - loss: 0.8789 - auc: 0.9473 - val\_loss: 1.1111 - val\_auc: 0.9312

Epoch 6/30

534/534 [==============================] - 58s 108ms/step - loss: 0.8748 - auc: 0.9483 - val\_loss: 1.0702 - val\_auc: 0.9284

Epoch 7/30

534/534 [==============================] - 58s 108ms/step - loss: 0.8668 - auc: 0.9498 - val\_loss: 1.0878 - val\_auc: 0.9320

Epoch 8/30

534/534 [==============================] - 57s 107ms/step - loss: 0.8719 - auc: 0.9488 - val\_loss: 1.0590 - val\_auc: 0.9318

Epoch 9/30

534/534 [==============================] - 58s 109ms/step - loss: 0.8650 - auc: 0.9498 - val\_loss: 1.0723 - val\_auc: 0.9333

Epoch 10/30

534/534 [==============================] - 58s 108ms/step - loss: 0.8540 - auc: 0.9517 - val\_loss: 1.0142 - val\_auc: 0.9435

Epoch 11/30

534/534 [==============================] - 58s 108ms/step - loss: 0.8467 - auc: 0.9523 - val\_loss: 1.0074 - val\_auc: 0.9387

Epoch 12/30

534/534 [==============================] - 58s 108ms/step - loss: 0.8358 - auc: 0.9541 - val\_loss: 1.0114 - val\_auc: 0.9437

Epoch 13/30

534/534 [==============================] - 57s 108ms/step - loss: 0.8269 - auc: 0.9554 - val\_loss: 0.9506 - val\_auc: 0.9489

Epoch 14/30

534/534 [==============================] - 58s 108ms/step - loss: 0.8172 - auc: 0.9564 - val\_loss: 0.9834 - val\_auc: 0.9481

Epoch 15/30

534/534 [==============================] - 57s 107ms/step - loss: 0.7967 - auc: 0.9595 - val\_loss: 0.8996 - val\_auc: 0.9521

Epoch 16/30

534/534 [==============================] - 57s 108ms/step - loss: 0.7555 - auc: 0.9625 - val\_loss: 0.9198 - val\_auc: 0.9500

Epoch 17/30

534/534 [==============================] - 57s 107ms/step - loss: 0.7526 - auc: 0.9630 - val\_loss: 0.8100 - val\_auc: 0.9556

Epoch 18/30

534/534 [==============================] - 58s 108ms/step - loss: 0.7199 - auc: 0.9660 - val\_loss: 0.8343 - val\_auc: 0.9561

Epoch 19/30

534/534 [==============================] - 58s 108ms/step - loss: 0.6790 - auc: 0.9693 - val\_loss: 0.7525 - val\_auc: 0.9615

Epoch 20/30

534/534 [==============================] - 58s 109ms/step - loss: 0.6483 - auc: 0.9717 - val\_loss: 0.7238 - val\_auc: 0.9661

Epoch 21/30

534/534 [==============================] - 57s 108ms/step - loss: 0.7421 - auc: 0.9638 - val\_loss: 0.9209 - val\_auc: 0.9519

Epoch 22/30

534/534 [==============================] - 58s 108ms/step - loss: 0.7796 - auc: 0.9616 - val\_loss: 0.9239 - val\_auc: 0.9466

Epoch 23/30

534/534 [==============================] - 58s 108ms/step - loss: 0.7298 - auc: 0.9660 - val\_loss: 0.8576 - val\_auc: 0.9534

Epoch 24/30

534/534 [==============================] - 58s 109ms/step - loss: 0.6867 - auc: 0.9699 - val\_loss: 0.7553 - val\_auc: 0.9637

Epoch 25/30

534/534 [==============================] - 58s 109ms/step - loss: 0.6626 - auc: 0.9719 - val\_loss: 0.7016 - val\_auc: 0.9696

Epoch 26/30

534/534 [==============================] - 57s 107ms/step - loss: 0.7799 - auc: 0.9615 - val\_loss: 0.8937 - val\_auc: 0.9521

Epoch 27/30

534/534 [==============================] - 57s 107ms/step - loss: 0.7707 - auc: 0.9609 - val\_loss: 0.9634 - val\_auc: 0.9432

Epoch 28/30

534/534 [==============================] - 58s 108ms/step - loss: 0.8020 - auc: 0.9579 - val\_loss: 0.9231 - val\_auc: 0.9488

Epoch 29/30

534/534 [==============================] - 57s 108ms/step - loss: 0.7664 - auc: 0.9627 - val\_loss: 0.8785 - val\_auc: 0.9513

Epoch 30/30

534/534 [==============================] - 57s 107ms/step - loss: 0.7205 - auc: 0.9665 - val\_loss: 0.7763 - val\_auc: 0.9597

66/66 [==============================] - 3s 44ms/step - loss: 0.6980 - auc: 0.9693

66/66 [==============================] - 3s 41ms/step

/usr/local/lib/python3.8/dist-packages/sklearn/metrics/\_classification.py:1318: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

X\_train, y\_train, X\_test, y\_test = load\_ptbxl(task = 'form')

table\_res\_ptbxl = type\_comp\_fit\_save\_model\_score(table\_res\_ptbxl, X\_train, y\_train, X\_test, y\_test, 'lstm', 'lstm\_form.hdf5', 'lstm\_form')

del(X\_train)

del(y\_train)

del(X\_test)

del(y\_test)

Model: "sequential\_4"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

lstm\_8 (LSTM) (None, 1000, 256) 275456

leaky\_re\_lu\_8 (LeakyReLU) (None, 1000, 256) 0

lstm\_9 (LSTM) (None, 256) 525312

leaky\_re\_lu\_9 (LeakyReLU) (None, 256) 0

dense\_4 (Dense) (None, 19) 4883

=================================================================

Total params: 805,651

Trainable params: 805,651

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

None

Epoch 1/30

228/228 [==============================] - 29s 112ms/step - loss: 3.4388 - auc: 0.7848 - val\_loss: 1.7913 - val\_auc: 0.9326

Epoch 2/30

228/228 [==============================] - 25s 109ms/step - loss: 3.4537 - auc: 0.7908 - val\_loss: 1.7679 - val\_auc: 0.9431

Epoch 3/30

228/228 [==============================] - 24s 106ms/step - loss: 3.4854 - auc: 0.7910 - val\_loss: 1.8755 - val\_auc: 0.9430

Epoch 4/30

228/228 [==============================] - 24s 107ms/step - loss: 3.5272 - auc: 0.7904 - val\_loss: 1.8738 - val\_auc: 0.9421

Epoch 5/30

228/228 [==============================] - 24s 107ms/step - loss: 3.5539 - auc: 0.7904 - val\_loss: 1.8636 - val\_auc: 0.9390

Epoch 6/30

228/228 [==============================] - 24s 107ms/step - loss: 3.5879 - auc: 0.7910 - val\_loss: 1.7812 - val\_auc: 0.9357

Epoch 7/30

228/228 [==============================] - 24s 106ms/step - loss: 3.6333 - auc: 0.7915 - val\_loss: 1.8411 - val\_auc: 0.9388

Epoch 8/30

228/228 [==============================] - 25s 108ms/step - loss: 3.6622 - auc: 0.7912 - val\_loss: 1.8649 - val\_auc: 0.9328

Epoch 9/30

228/228 [==============================] - 24s 107ms/step - loss: 3.7246 - auc: 0.7911 - val\_loss: 2.0058 - val\_auc: 0.9147

Epoch 10/30

228/228 [==============================] - 24s 107ms/step - loss: 3.7469 - auc: 0.7913 - val\_loss: 1.8828 - val\_auc: 0.9346

Epoch 11/30

228/228 [==============================] - 24s 106ms/step - loss: 3.7896 - auc: 0.7908 - val\_loss: 1.8468 - val\_auc: 0.9427

Epoch 12/30

228/228 [==============================] - 24s 106ms/step - loss: 3.8325 - auc: 0.7908 - val\_loss: 1.9559 - val\_auc: 0.9317

Epoch 13/30

228/228 [==============================] - 24s 106ms/step - loss: 3.8649 - auc: 0.7909 - val\_loss: 1.9585 - val\_auc: 0.9253

Epoch 14/30

228/228 [==============================] - 24s 106ms/step - loss: 3.9078 - auc: 0.7906 - val\_loss: 1.9804 - val\_auc: 0.9217

Epoch 15/30

228/228 [==============================] - 24s 105ms/step - loss: 3.9222 - auc: 0.7917 - val\_loss: 1.9628 - val\_auc: 0.9230

Epoch 16/30

228/228 [==============================] - 24s 106ms/step - loss: 3.9846 - auc: 0.7909 - val\_loss: 1.8122 - val\_auc: 0.9376

Epoch 17/30

228/228 [==============================] - 24s 106ms/step - loss: 4.0345 - auc: 0.7909 - val\_loss: 1.8963 - val\_auc: 0.9328

28/28 [==============================] - 1s 45ms/step - loss: 3.3381 - auc: 0.8049

28/28 [==============================] - 2s 39ms/step

/usr/local/lib/python3.8/dist-packages/sklearn/metrics/\_classification.py:1318: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

### lstm\_bidir

X\_train, y\_train, X\_test, y\_test = load\_ptbxl(task = 'diag')

table\_res\_ptbxl = type\_comp\_fit\_save\_model\_score(table\_res\_ptbxl, X\_train, y\_train, X\_test, y\_test, 'lstm\_bidir', 'lstm\_bidir\_diag.hdf5', 'lstm\_bidir\_diag')

del(X\_train)

del(y\_train)

del(X\_test)

del(y\_test)

Model: "sequential\_5"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

bidirectional (Bidirectiona (None, 1000, 512) 550912

l)

leaky\_re\_lu\_10 (LeakyReLU) (None, 1000, 512) 0

bidirectional\_1 (Bidirectio (None, 512) 1574912

nal)

leaky\_re\_lu\_11 (LeakyReLU) (None, 512) 0

dense\_5 (Dense) (None, 44) 22572

=================================================================

Total params: 2,148,396

Trainable params: 2,148,396

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

None

Epoch 1/30

542/542 [==============================] - 138s 244ms/step - loss: 4.5472 - auc: 0.8523 - val\_loss: 5.6511 - val\_auc: 0.8341

Epoch 2/30

542/542 [==============================] - 127s 234ms/step - loss: 5.1828 - auc: 0.8544 - val\_loss: 6.3809 - val\_auc: 0.8263

Epoch 3/30

542/542 [==============================] - 124s 229ms/step - loss: 5.6940 - auc: 0.8538 - val\_loss: 6.9382 - val\_auc: 0.8396

Epoch 4/30

542/542 [==============================] - 124s 229ms/step - loss: 6.2270 - auc: 0.8549 - val\_loss: 7.6131 - val\_auc: 0.8307

Epoch 5/30

542/542 [==============================] - 125s 230ms/step - loss: 6.7545 - auc: 0.8533 - val\_loss: 8.2856 - val\_auc: 0.8439

Epoch 6/30

542/542 [==============================] - 125s 230ms/step - loss: 7.1873 - auc: 0.8551 - val\_loss: 9.1142 - val\_auc: 0.8288

Epoch 7/30

542/542 [==============================] - 125s 231ms/step - loss: 7.6727 - auc: 0.8543 - val\_loss: 9.6365 - val\_auc: 0.8324

Epoch 8/30

542/542 [==============================] - 125s 231ms/step - loss: 8.2207 - auc: 0.8542 - val\_loss: 10.4434 - val\_auc: 0.8361

Epoch 9/30

542/542 [==============================] - 122s 224ms/step - loss: 8.7071 - auc: 0.8552 - val\_loss: 11.1302 - val\_auc: 0.8369

Epoch 10/30

542/542 [==============================] - 123s 226ms/step - loss: 9.2616 - auc: 0.8549 - val\_loss: 11.8029 - val\_auc: 0.8303

Epoch 11/30

542/542 [==============================] - 125s 231ms/step - loss: 9.8361 - auc: 0.8540 - val\_loss: 12.6372 - val\_auc: 0.8293

Epoch 12/30

542/542 [==============================] - 124s 228ms/step - loss: 10.2797 - auc: 0.8532 - val\_loss: 13.0362 - val\_auc: 0.8392

Epoch 13/30

542/542 [==============================] - 124s 228ms/step - loss: 10.7754 - auc: 0.8548 - val\_loss: 13.7288 - val\_auc: 0.8308

Epoch 14/30

542/542 [==============================] - 124s 228ms/step - loss: 11.2361 - auc: 0.8547 - val\_loss: 14.5567 - val\_auc: 0.8217

Epoch 15/30

542/542 [==============================] - 126s 232ms/step - loss: 11.7374 - auc: 0.8545 - val\_loss: 15.2235 - val\_auc: 0.8375

Epoch 16/30

542/542 [==============================] - 125s 231ms/step - loss: 12.0529 - auc: 0.8549 - val\_loss: 15.4897 - val\_auc: 0.8385

68/68 [==============================] - 6s 93ms/step - loss: 4.9083 - auc: 0.8558

68/68 [==============================] - 7s 88ms/step

/usr/local/lib/python3.8/dist-packages/sklearn/metrics/\_classification.py:1318: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

X\_train, y\_train, X\_test, y\_test = load\_ptbxl(task = 'superdiag')

table\_res\_ptbxl = type\_comp\_fit\_save\_model\_score(table\_res\_ptbxl, X\_train, y\_train, X\_test, y\_test, 'lstm\_bidir', 'lstm\_bidir\_superdiag.hdf5', 'lstm\_bidir\_superdiag')

del(X\_train)

del(y\_train)

del(X\_test)

del(y\_test)

Model: "sequential\_6"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

bidirectional\_2 (Bidirectio (None, 1000, 512) 550912

nal)

leaky\_re\_lu\_12 (LeakyReLU) (None, 1000, 512) 0

bidirectional\_3 (Bidirectio (None, 512) 1574912

nal)

leaky\_re\_lu\_13 (LeakyReLU) (None, 512) 0

dense\_6 (Dense) (None, 5) 2565

=================================================================

Total params: 2,128,389

Trainable params: 2,128,389

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

None

Epoch 1/30

542/542 [==============================] - 135s 238ms/step - loss: 2.0082 - auc: 0.6396 - val\_loss: 2.1085 - val\_auc: 0.5846

Epoch 2/30

542/542 [==============================] - 126s 232ms/step - loss: 2.0064 - auc: 0.6363 - val\_loss: 2.1204 - val\_auc: 0.5940

Epoch 3/30

542/542 [==============================] - 125s 231ms/step - loss: 2.0098 - auc: 0.6397 - val\_loss: 2.0976 - val\_auc: 0.6023

Epoch 4/30

542/542 [==============================] - 125s 230ms/step - loss: 2.0043 - auc: 0.6372 - val\_loss: 2.1250 - val\_auc: 0.5850

Epoch 5/30

542/542 [==============================] - 128s 236ms/step - loss: 2.0075 - auc: 0.6385 - val\_loss: 2.1096 - val\_auc: 0.6047

Epoch 6/30

542/542 [==============================] - 127s 234ms/step - loss: 2.0027 - auc: 0.6397 - val\_loss: 2.1537 - val\_auc: 0.5845

Epoch 7/30

542/542 [==============================] - 126s 232ms/step - loss: 2.0044 - auc: 0.6355 - val\_loss: 2.1018 - val\_auc: 0.5846

Epoch 8/30

542/542 [==============================] - 124s 228ms/step - loss: 2.0088 - auc: 0.6390 - val\_loss: 2.1531 - val\_auc: 0.5945

Epoch 9/30

542/542 [==============================] - 124s 228ms/step - loss: 2.0064 - auc: 0.6385 - val\_loss: 2.1135 - val\_auc: 0.5939

Epoch 10/30

542/542 [==============================] - 123s 227ms/step - loss: 2.0076 - auc: 0.6381 - val\_loss: 2.1143 - val\_auc: 0.6023

Epoch 11/30

542/542 [==============================] - 123s 227ms/step - loss: 2.0102 - auc: 0.6370 - val\_loss: 2.1235 - val\_auc: 0.5762

Epoch 12/30

542/542 [==============================] - 123s 227ms/step - loss: 2.0101 - auc: 0.6387 - val\_loss: 2.1220 - val\_auc: 0.5845

Epoch 13/30

542/542 [==============================] - 123s 227ms/step - loss: 2.0094 - auc: 0.6370 - val\_loss: 2.1220 - val\_auc: 0.6050

Epoch 14/30

542/542 [==============================] - 123s 228ms/step - loss: 2.0073 - auc: 0.6386 - val\_loss: 2.1228 - val\_auc: 0.5939

Epoch 15/30

542/542 [==============================] - 123s 227ms/step - loss: 2.0045 - auc: 0.6380 - val\_loss: 2.1965 - val\_auc: 0.6046

Epoch 16/30

542/542 [==============================] - 124s 229ms/step - loss: 2.0045 - auc: 0.6386 - val\_loss: 2.0887 - val\_auc: 0.6070

Epoch 17/30

542/542 [==============================] - 124s 228ms/step - loss: 2.0071 - auc: 0.6381 - val\_loss: 2.1592 - val\_auc: 0.5762

Epoch 18/30

542/542 [==============================] - 124s 228ms/step - loss: 2.0079 - auc: 0.6384 - val\_loss: 2.1439 - val\_auc: 0.6024

Epoch 19/30

542/542 [==============================] - 124s 228ms/step - loss: 2.0057 - auc: 0.6383 - val\_loss: 2.0805 - val\_auc: 0.6024

Epoch 20/30

542/542 [==============================] - 123s 228ms/step - loss: 2.0085 - auc: 0.6374 - val\_loss: 2.1310 - val\_auc: 0.6024

Epoch 21/30

542/542 [==============================] - 124s 228ms/step - loss: 2.0060 - auc: 0.6407 - val\_loss: 2.1535 - val\_auc: 0.5851

Epoch 22/30

542/542 [==============================] - 123s 228ms/step - loss: 2.0102 - auc: 0.6367 - val\_loss: 2.1193 - val\_auc: 0.5691

Epoch 23/30

542/542 [==============================] - 123s 228ms/step - loss: 2.0097 - auc: 0.6377 - val\_loss: 2.2076 - val\_auc: 0.5935

Epoch 24/30

542/542 [==============================] - 123s 228ms/step - loss: 2.0078 - auc: 0.6391 - val\_loss: 2.1289 - val\_auc: 0.5940

Epoch 25/30

542/542 [==============================] - 123s 228ms/step - loss: 2.0078 - auc: 0.6372 - val\_loss: 2.0823 - val\_auc: 0.6070

Epoch 26/30

542/542 [==============================] - 123s 228ms/step - loss: 2.0068 - auc: 0.6390 - val\_loss: 2.1445 - val\_auc: 0.5762

Epoch 27/30

542/542 [==============================] - 123s 228ms/step - loss: 2.0059 - auc: 0.6396 - val\_loss: 2.1347 - val\_auc: 0.5612

Epoch 28/30

542/542 [==============================] - 123s 227ms/step - loss: 2.0074 - auc: 0.6382 - val\_loss: 2.2229 - val\_auc: 0.5739

Epoch 29/30

542/542 [==============================] - 129s 238ms/step - loss: 2.0084 - auc: 0.6376 - val\_loss: 2.1628 - val\_auc: 0.5939

Epoch 30/30

542/542 [==============================] - 124s 229ms/step - loss: 2.0110 - auc: 0.6366 - val\_loss: 2.1853 - val\_auc: 0.5940

68/68 [==============================] - 6s 91ms/step - loss: 2.0162 - auc: 0.6375

68/68 [==============================] - 7s 86ms/step

/usr/local/lib/python3.8/dist-packages/sklearn/metrics/\_classification.py:1318: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

X\_train, y\_train, X\_test, y\_test = load\_ptbxl(task = 'subdiag')

table\_res\_ptbxl = type\_comp\_fit\_save\_model\_score(table\_res\_ptbxl, X\_train, y\_train, X\_test, y\_test, 'lstm\_bidir', 'lstm\_bidir\_subdiag.hdf5', 'lstm\_bidir\_subdiag')

del(X\_train)

del(y\_train)

del(X\_test)

del(y\_test)

Model: "sequential\_7"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

bidirectional\_4 (Bidirectio (None, 1000, 512) 550912

nal)

leaky\_re\_lu\_14 (LeakyReLU) (None, 1000, 512) 0

bidirectional\_5 (Bidirectio (None, 512) 1574912

nal)

leaky\_re\_lu\_15 (LeakyReLU) (None, 512) 0

dense\_7 (Dense) (None, 23) 11799

=================================================================

Total params: 2,137,623

Trainable params: 2,137,623

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

None

Epoch 1/30

542/542 [==============================] - 134s 238ms/step - loss: 3.8584 - auc: 0.8212 - val\_loss: 4.5490 - val\_auc: 0.8048

Epoch 2/30

542/542 [==============================] - 127s 234ms/step - loss: 4.3023 - auc: 0.8220 - val\_loss: 5.2378 - val\_auc: 0.8019

Epoch 3/30

542/542 [==============================] - 127s 235ms/step - loss: 4.7402 - auc: 0.8225 - val\_loss: 5.7419 - val\_auc: 0.8013

Epoch 4/30

542/542 [==============================] - 127s 234ms/step - loss: 5.1550 - auc: 0.8224 - val\_loss: 6.3275 - val\_auc: 0.8044

Epoch 5/30

542/542 [==============================] - 126s 233ms/step - loss: 5.5722 - auc: 0.8222 - val\_loss: 6.8473 - val\_auc: 0.8138

Epoch 6/30

542/542 [==============================] - 126s 233ms/step - loss: 5.8890 - auc: 0.8232 - val\_loss: 7.5258 - val\_auc: 0.8046

Epoch 7/30

542/542 [==============================] - 126s 233ms/step - loss: 6.3127 - auc: 0.8230 - val\_loss: 8.0766 - val\_auc: 0.7993

Epoch 8/30

542/542 [==============================] - 123s 228ms/step - loss: 6.7987 - auc: 0.8226 - val\_loss: 8.6756 - val\_auc: 0.7981

Epoch 9/30

542/542 [==============================] - 123s 228ms/step - loss: 7.1629 - auc: 0.8227 - val\_loss: 9.1879 - val\_auc: 0.8048

Epoch 10/30

542/542 [==============================] - 123s 228ms/step - loss: 7.6444 - auc: 0.8229 - val\_loss: 9.6428 - val\_auc: 0.8145

Epoch 11/30

542/542 [==============================] - 123s 228ms/step - loss: 8.0344 - auc: 0.8227 - val\_loss: 10.3134 - val\_auc: 0.8051

Epoch 12/30

542/542 [==============================] - 124s 228ms/step - loss: 8.4488 - auc: 0.8222 - val\_loss: 10.8168 - val\_auc: 0.8140

Epoch 13/30

542/542 [==============================] - 123s 228ms/step - loss: 8.8105 - auc: 0.8226 - val\_loss: 11.3427 - val\_auc: 0.8044

Epoch 14/30

542/542 [==============================] - 123s 227ms/step - loss: 9.1788 - auc: 0.8231 - val\_loss: 11.9934 - val\_auc: 0.8007

Epoch 15/30

542/542 [==============================] - 121s 223ms/step - loss: 9.6245 - auc: 0.8226 - val\_loss: 12.5812 - val\_auc: 0.8103

Epoch 16/30

542/542 [==============================] - 121s 222ms/step - loss: 10.0434 - auc: 0.8237 - val\_loss: 13.0588 - val\_auc: 0.8129

68/68 [==============================] - 6s 89ms/step - loss: 4.0691 - auc: 0.8169

68/68 [==============================] - 7s 85ms/step

/usr/local/lib/python3.8/dist-packages/sklearn/metrics/\_classification.py:1318: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

*# X\_train, y\_train, X\_test, y\_test = load\_ptbxl(task = 'rhythm')*

*# table\_res\_ptbxl = type\_comp\_fit\_save\_model\_score(table\_res\_ptbxl, X\_train, y\_train, X\_test, y\_test, 'lstm\_bidir', 'lstm\_bidir\_rhythm.hdf5', 'lstm\_bidir\_rhythm')*

*# del(X\_train)*

*# del(y\_train)*

*# del(X\_test)*

del(y\_test)

X\_train, y\_train, X\_test, y\_test = load\_ptbxl(task = 'form')

table\_res\_ptbxl = type\_comp\_fit\_save\_model\_score(table\_res\_ptbxl, X\_train, y\_train, X\_test, y\_test, 'lstm\_bidir', 'lstm\_bidir\_form.hdf5', 'lstm\_bidir\_form')

del(X\_train)

del(y\_train)

del(X\_test)

del(y\_test)

Model: "sequential\_9"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

bidirectional\_8 (Bidirectio (None, 1000, 512) 550912

nal)

leaky\_re\_lu\_18 (LeakyReLU) (None, 1000, 512) 0

bidirectional\_9 (Bidirectio (None, 512) 1574912

nal)

leaky\_re\_lu\_19 (LeakyReLU) (None, 512) 0

dense\_9 (Dense) (None, 19) 9747

=================================================================

Total params: 2,135,571

Trainable params: 2,135,571

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

None

Epoch 1/30

228/228 [==============================] - 62s 250ms/step - loss: 3.5201 - auc: 0.7841 - val\_loss: 1.8541 - val\_auc: 0.9232

Epoch 2/30

228/228 [==============================] - 55s 240ms/step - loss: 3.5967 - auc: 0.7861 - val\_loss: 1.7205 - val\_auc: 0.9434

Epoch 3/30

228/228 [==============================] - 54s 238ms/step - loss: 3.6410 - auc: 0.7874 - val\_loss: 1.8854 - val\_auc: 0.9410

Epoch 4/30

228/228 [==============================] - 54s 237ms/step - loss: 3.7050 - auc: 0.7865 - val\_loss: 1.8290 - val\_auc: 0.9393

Epoch 5/30

228/228 [==============================] - 54s 237ms/step - loss: 3.7403 - auc: 0.7874 - val\_loss: 1.9198 - val\_auc: 0.9210

Epoch 6/30

228/228 [==============================] - 54s 238ms/step - loss: 3.7888 - auc: 0.7878 - val\_loss: 1.8159 - val\_auc: 0.9345

Epoch 7/30

228/228 [==============================] - 54s 237ms/step - loss: 3.8645 - auc: 0.7884 - val\_loss: 1.9140 - val\_auc: 0.9411

Epoch 8/30

228/228 [==============================] - 54s 237ms/step - loss: 3.9141 - auc: 0.7883 - val\_loss: 1.8092 - val\_auc: 0.9290

Epoch 9/30

228/228 [==============================] - 54s 237ms/step - loss: 4.0242 - auc: 0.7870 - val\_loss: 2.0453 - val\_auc: 0.9047

Epoch 10/30

228/228 [==============================] - 54s 236ms/step - loss: 4.0554 - auc: 0.7881 - val\_loss: 2.0242 - val\_auc: 0.9346

Epoch 11/30

228/228 [==============================] - 54s 235ms/step - loss: 4.1279 - auc: 0.7872 - val\_loss: 1.7715 - val\_auc: 0.9392

Epoch 12/30

228/228 [==============================] - 54s 235ms/step - loss: 4.2027 - auc: 0.7875 - val\_loss: 1.9517 - val\_auc: 0.9321

Epoch 13/30

228/228 [==============================] - 54s 235ms/step - loss: 4.2700 - auc: 0.7870 - val\_loss: 1.9927 - val\_auc: 0.9285

Epoch 14/30

228/228 [==============================] - 54s 235ms/step - loss: 4.3255 - auc: 0.7881 - val\_loss: 2.0581 - val\_auc: 0.9122

Epoch 15/30

228/228 [==============================] - 54s 235ms/step - loss: 4.3543 - auc: 0.7886 - val\_loss: 1.9979 - val\_auc: 0.9128

Epoch 16/30

228/228 [==============================] - 54s 235ms/step - loss: 4.4595 - auc: 0.7880 - val\_loss: 1.8361 - val\_auc: 0.9418

Epoch 17/30

228/228 [==============================] - 54s 235ms/step - loss: 4.5466 - auc: 0.7877 - val\_loss: 1.8628 - val\_auc: 0.9306

28/28 [==============================] - 3s 94ms/step - loss: 3.4868 - auc: 0.8030

28/28 [==============================] - 4s 88ms/step

/usr/local/lib/python3.8/dist-packages/sklearn/metrics/\_classification.py:1318: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

### Сохранение результатов в формате .csv

table\_res\_ptbxl.to\_csv('table\_res\_ptbxl.csv')

1. Nils Strodthoff, Patrick Wagner, T. Schaeffter, W. Samek, “Deep Learning for ECG Analysis: Benchmarks and Insights from PTB-XL”, DOI: 10.1109/JBHI.2020.3022989Corpus ID: 216562803, Published 28 April 2020, IEEE Journal of Biomedical and Health Informatics [↑](#footnote-ref-1)
2. Nils Strodthoff, Patrick Wagner, T. Schaeffter, W. Samek, “Deep Learning for ECG Analysis: Benchmarks and Insights from PTB-XL”, DOI: 10.1109/JBHI.2020.3022989Corpus ID: 216562803, Published 28 April 2020, IEEE Journal of Biomedical and Health Informatics [↑](#footnote-ref-2)
3. Nils Strodthoff, Patrick Wagner, T. Schaeffter, W. Samek, “Deep Learning for ECG Analysis: Benchmarks and Insights from PTB-XL”, DOI: 10.1109/JBHI.2020.3022989Corpus ID: 216562803, Published 28 April 2020, IEEE Journal of Biomedical and Health Informatics [↑](#footnote-ref-3)
4. Nils Strodthoff, Patrick Wagner, T. Schaeffter, W. Samek, “Deep Learning for ECG Analysis: Benchmarks and Insights from PTB-XL”, DOI: 10.1109/JBHI.2020.3022989Corpus ID: 216562803, Published 28 April 2020, IEEE Journal of Biomedical and Health Informatics [↑](#footnote-ref-4)