МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

РАБОТА ПРОВЕРЕНА

Рецензент
Доцент кафедры ИИТиМОИ
«ЮУрГГПУ», к.п.н.

_____ Л.С. Носова

«___»_____ 2024 г.

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н.,
профессор

_____ Л.Б. Соколинский

«___»_____ 2024 г.

Разработка модели промежуточного слияния модальности клинических данных с мэппингом эмбеддингов из модальности рентгенологических снимков для классификации заболеваний легких

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 09.04.04.2024.308-1476.ВКР

Научный руководитель,
доцент кафедры СП, к.п.н.
_____ О.Н. Иванова

Автор работы,
студент группы КЭ-228
_____ А.В. Мелехин

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
«___»_____ 2024 г.

Челябинск, 2024 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
29.01.2024 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы магистранта
студенту группы КЭ-228
Мелехину Артему Викторовичу,
обучающемуся по направлению
09.04.04 «Программная инженерия»
(магистерская программа «Искусственный интеллект и инженерия данных»)

1. **Тема работы** (утверждена приказом ректора от 22.04.2024 г. № 764-13/12)
Разработка модели промежуточного слияния модальности клинических данных с мэппингом эмбеддингов из модальности рентгенологических снимков для классификации заболеваний легких.
2. **Срок сдачи студентом законченной работы:** 20.05.2024 г.
3. **Исходные данные к работе**

3.1. Alistair E, Tom J., Nathaniel R., Matthew P., Chih-ying D., Yifan P., Zhiyong L., Roger G., Seth J., Steven H. MIMIC-CXR-JPG, a large publicly available database of labeled chest radiographs. [Электронный ресурс] // arXiv.org. 2019. Дата обновления: 14.11.2019 г. URL: https://arxiv.org/abs/1901.07042 (дата обращения: 01.01.2024 г.).

3.2. Thanh-Tung N., Viktor S., Abhinav K., Stefan W., Shao-Syuan H., Jie-Jyun L., Chih-Jen L. Mimic-IV-ICD: A new benchmark for eXtreme MultiLabel Classification. [Электронный ресурс] // arXiv.org. 2023. Дата обновления: 27.04.2023 г. URL: https://arxiv.org/abs/2304.13998 (дата обращения: 03.01.2024 г.).

4. **Перечень подлежащих разработке вопросов**

4.1. Провести анализ предметной области.

4.2. Извлечь данные для обучения из большой базы MIMIC.
4.3. Разработать архитектуру нейронной сети.
4.4. Программно реализовать нейронную сеть.
5. Дата выдачи задания: 29.01.2024 г.

Научный руководитель,
доцент кафедры СП, к.п.н. О.Н. Иванова

Задание принял к исполнению А.В. Мелехин

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность

Диагностика заболеваний легких является важной и необходимой процедурой, которая позволяет выявить различные проблемы дыхательной системы на ранней стадии. Это позволяет вовремя начать лечение, избежать возможных осложнений, сохранить здоровье человека и жизнь. Из статистических данных видно, что заболевания легких входит в тройку причин смертности людей по всему миру [1]. Поэтому методы диагностирования постоянно улучшаются.

Существует множество способов анализа состояния легких: анализ газового состава артериальной крови и пульсоксиметрия, визуализирующее исследование грудной клетки, бронхоскопия и др., что позволяет получить различные категории данных после обследования человека [2].

Для постановки правильного диагноза врачи обычно принимают во внимание медицинские данные пациента из более чем одного источника: снимки, временные ряды, показания пациента и других анализов, так как болезнь может не проявляться в одном показании, но в другом четко на это указывать.

В следствии чего есть основания полагать, что методы, основанные на искусственном интеллекте, должны имитировать клинический подход и учитывать различные источники данных, которые позволят проводить всесторонний анализ пациента и как следствие ставить более точный диагноз. Поэтому цель данной работы заключается в создании многомодальной нейронной сети, с помощью которой будет возможно поставить более точный анализ, используя данные различных модальностей.

Постановка задачи

Целью выпускной квалификационной работы является разработка модели промежуточного слияния модальности клинических данных с мэппин-

гом эмбеддингов из модальности рентгенологических снимков для классификации заболеваний легких. Для достижения поставленной цели необходимо решить следующие задачи:

1) произвести анализ предметной области;
2) извлечь данные для обучения нейронной сети из большой базы MIMIC;
3) разработать архитектуру нейронной сети;
4) программно реализовать нейронную сеть.

Структура и содержание работы

Работа состоит из введения, трех глав, заключения и списка литературы. Объем работы составляет 47 страниц, объем списка литературы – 24 источника.

В первой главе описывается обзор научной литературы и программных аналогов.

Вторая глава посвящена алгоритмам обработки изображения и медицинских данных с последующим формированием набора данных для обучения нейронной сети.

В третьей главе описана реализация нейронной сети и ее обучение с последующим тестированием и сравнением результатов.

В приложении А содержатся поля медицинских параметров, используемые для классификации.

В приложении Б содержатся нормализованные временные ряды медицинских данных.

В приложении В содержатся графики изменения метрики и ошибки при обучении блоков нейронной сети.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Мультимодальные архитектуры нейронных сетей

Окружающий мир представляет нам данные из множества источников и интуитивно понятно, что модели, объединяющие информацию разных модальностей, превосходят одномодальные аналоги, поскольку агрегируется больше признаков. Мультимодальное слияние относится к процессу объединения информации из нескольких модальностей с целью улучшения понимания или анализа определенного явления или проблемы. По сути, он включает в себя интеграцию данных из разных источников, таких как текст, изображения, аудио, видео и показания датчиков, чтобы получить более полное и точное представление базовой информации. На рисунке 1 приводится общая схема мультимодальной архитектуры.

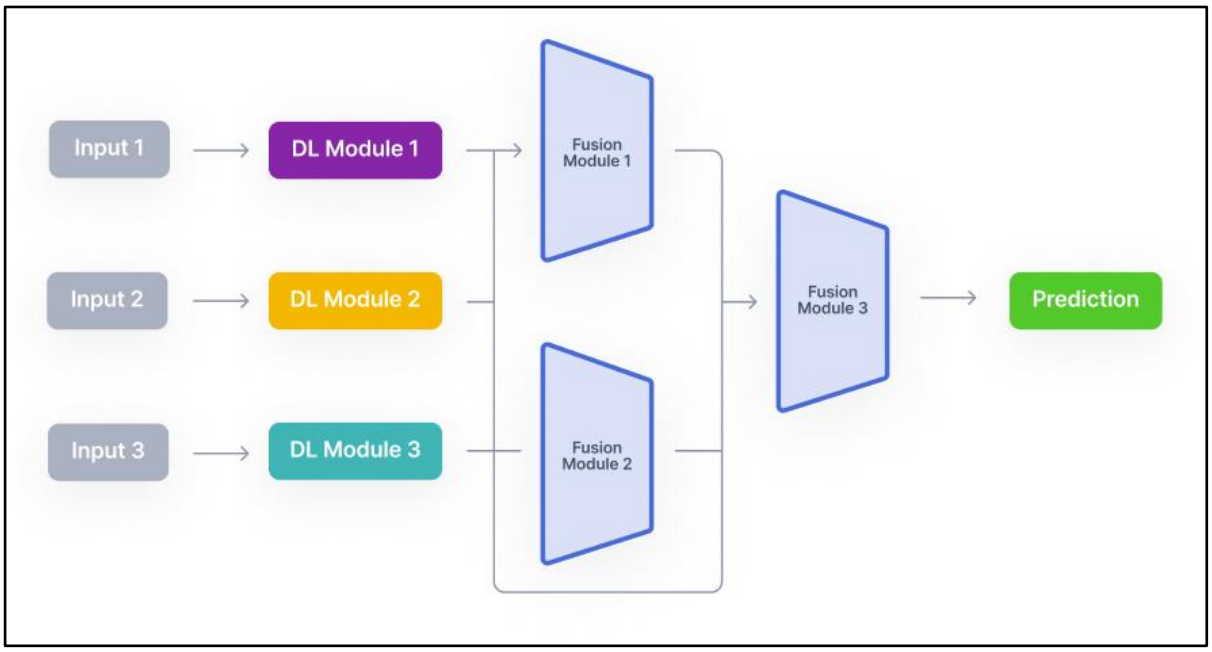

Рисунок 1 – Схема мультимодального слияния

В статье [3] приводится сравнительный анализ использования нейронных сетей, которым на вход поступают данные трех модальностей: текст, видео и аудио. В таблице 1 представлена точность классификации набора данных IEMOCAP [4] в различных комбинациях.

Таблица 1 – Точность классификации набора данных IEMOCAP

Модальность	**Точность**
Text	49,93 ± 0,57
Text + Video	51,08 ± 0,66
Text + Audio	53,03 ± 0,21
Text + Video + Audio	**53,89 ± 0,47**

Итоговые результаты показали, что комбинация всех видов данных дает лучшую точность модели.

В работе [5] проводилось исследование по диагностированию рака легких, использовались клинические данные и рентгеновские снимки набора данных NLST [6]. В таблице 2 приведены результаты бинарной классификации с помощью одномодальной и мультимодальных моделей.

Таблица 2 – Результаты бинарной классификации заболевания рака легких набора данных NLST

Модальность	**AUC**
Image	0,7897
Clinical table	0,5241
Image + Clinical table	**0,8021**

Наибольшую площадь под ROC-кривой показала модель, принимающая на вход две модальности.

Эти и многие другие исследования показывают, что использование многомодальных моделей положительно влияет на точность классификации.

1.2. Типы слияния модальностей

Существуют различные подходы к мультимодальному слиянию, в том числе следующие.

1. Ранее слияние – необработанные данные из разных модальностей объединяются на входном уровне перед подачей в модель. Например, объединение данных текста и изображения в один входной вектор.

2. Промежуточное слияние – объединяет данные из разных модальностей на различных этапах промежуточной обработки в рамках архитектуры модели.

3. Позднее слияние – данные каждой модальности обрабатываются независимо с помощью отдельных моделей, а результаты этих моделей затем объединяются на более позднем этапе.

На рисунке 2 представлены схемы слияний, (a) ранее слияние, (b) промежуточное слияние, (c) позднее слияние.

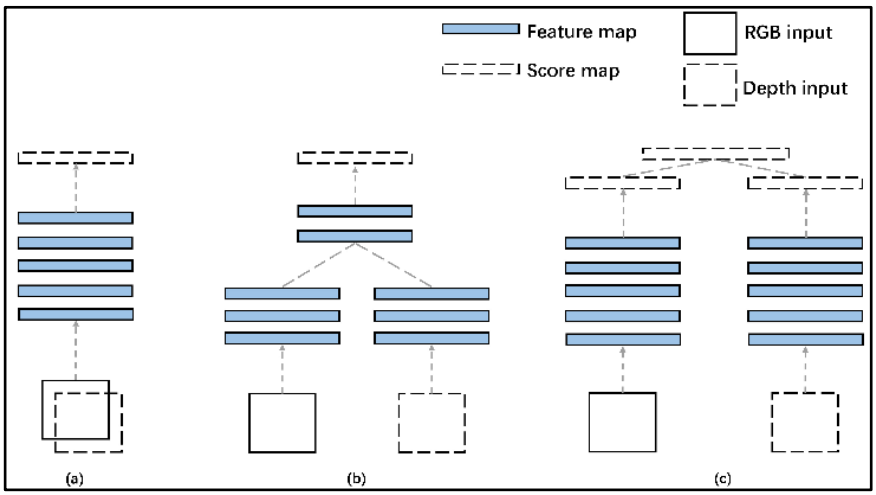

Рисунок 2 – Схема слияний модальностей

В последнее время также стал применятся метод кросс-мэппинг, который представляет из себя обмен частью эмбэдингов различных модальностей между собой на промежуточных слоях. На рисунке 3 представлена схема кросс-мэппинга.

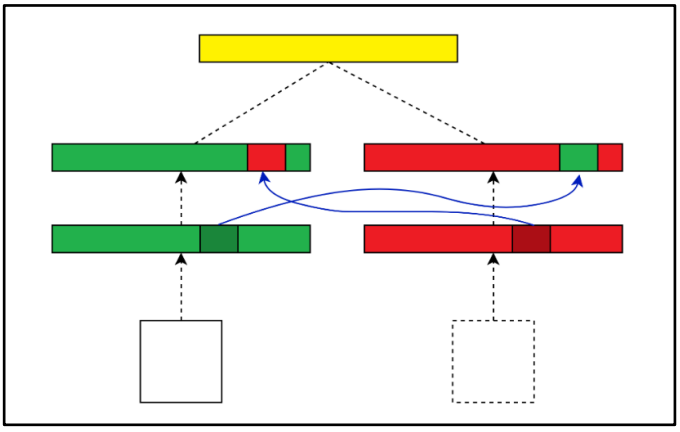

Рисунок 3 – Схема кросс-меппинга

Выбор правильного метода слияния зависит от нескольких соображений, таких как:

– характеристики данных;
– сложность задачи;
– вычислительная эффективность;
– базовые знания.

Успешное мультимодальное слияние зависит от различных факторов: качество данных, выбранный подход слияния, архитектура модели и критерии оценки. Часто для достижения оптимальных результатов необходимо экспериментальное и итеративное уточнение с различными гиперпараметрами и методами.

1.3. Обзор литературы по кросс-мэппингу

Люди воспринимают мир, одновременно обрабатывая различные модальности, такие как изображение, звук и обоняние, и другие, а затем объединяя их. По аналогии, мультимодальное слияние стало ключевой исследовательской проблемой в области искусственного интеллекта.

В работе [7] предлагается сеть обмена каналами Channel Exchanging Networks (CEN), которая решает задачи семантической сегментации и Image-to-Image Translation [8], получая на вход карту глубины и RGB изображение. Вместо использования слияния CEN динамически обменивает каналы между подсетями для объединения. На рисунке 4 приведена схема сети CEN.

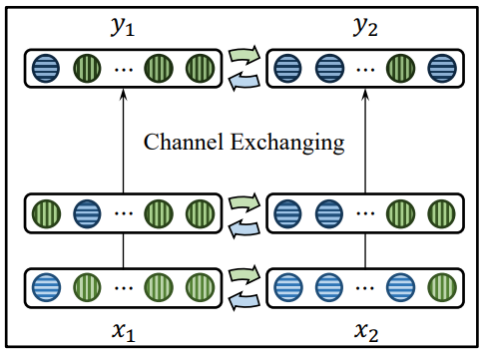

Рисунок 4 – Схема CEN

Сеть использует коэффициент масштабирования пакетной нормализации (BN) в качестве измерения важности каждого соответствующего канала. Далее канал, у которого коэффициент близок нулю заменяется другим каналом. Такой обмен не зависит от параметров и самоадаптивен, поскольку динамически управляются коэффициентами масштабирования, которые определяются в процессе обучения. Также направленный обмен каналов определяется только в пределах определенного диапазона каналов в каждой модальности, для сохранения внутримодальной обработки.

Переходя к нашей задаче, можно сказать, что в данной реализации входные данные хоть и представляют разные представления об объекте, но на более низком уровне карта глубины и RGB изображение отличаются лишь количеством каналов, в отличии от того, если бы на вход поступало изображение, текст или табличные данные.

В работе [9] представлена модель MuSE. В работе рассматривается вопрос мультимодального слияния на основе обмена данных изображения и текста, использующего трансформер.

Сначала используется два энкодера, которые раздельно переводят изображение и текст в низкоразмерные пространства. Далее два декодера упорядочивают эмбеддинги и помещают их в одно пространство. Два декодера фиксируют корреляции между текстами и изображениями с помощью задачи создания субтитров к изображениям и задачи преобразования текста в изображение соответственно. Затем упорядоченные эмбеддинги передаются на CrossTransformer, пара экодеров которого используются для обмена данными между модальностями. В частности, CrossTransformer сначала изучает глобальную контекстную информацию о входных данных в нижних слоях. После этого он выполняет кросс-модальный обмен, выбирая долю токенов в одной модальности и заменяя их эмбединги средним значением эмбеддингов в другой модальности.

Как показано на рисунке 5, модель MuSE состоит из четырех компонентов, в частности компонент 1 используется для проецирования входных

текстов и изображений в низкоразмерные пространства, который включает в себя кодировщик текста и кодировщик изображений. Компоненты 2 и 3 представляют из себя два регуляризатора встраивания для извлечения эмбеддингов мультимодальных входных данных в одно и то же пространство, которые используются в качестве декодеров и реализуют задачу преобразования текста в изображение и задачу создания субтитров к изображениям соответственно. После того, как эмбеддинги мультимодальных входных данных сгенерированы, они передаются их в компонент 4, который представляет собой модуль на основе преобразователя, называемый CrossTransformer. Он выполняет мультимодальный обмен эмбеддингами и генерирует объединенные вложения из мультимодальности.

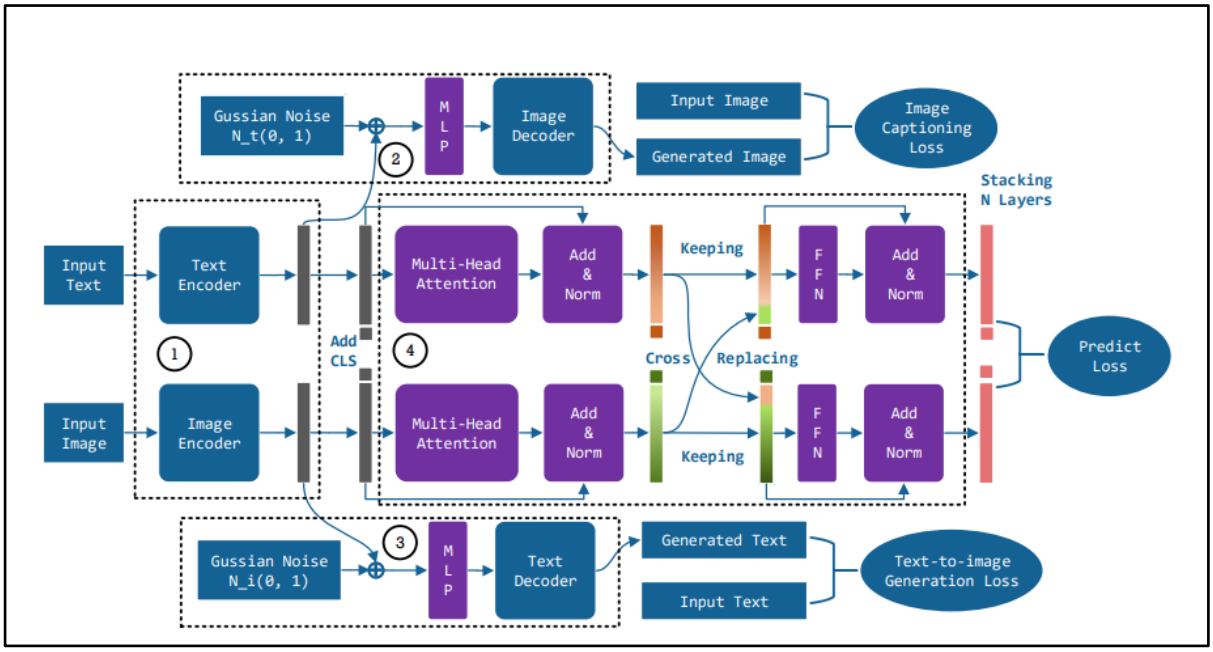

Рисунок 5 – Общая архитектура MuSE

В статье [10] описана модель – ARMOUR на основе трансформера с токенами, специфичными для модальностей, которые суммируются с соответствующими модальности для достижения эффективного кросс-модального взаимодействия, учитывающего недостающие модальности в клиническом контексте. Модель дополнительно уточняется с помощью интермо-

дального контрастного обучения между выборками для улучшения представлений и повышения производительности прогнозирования. На рисунке 6 представлена архитектура ARMOUR.

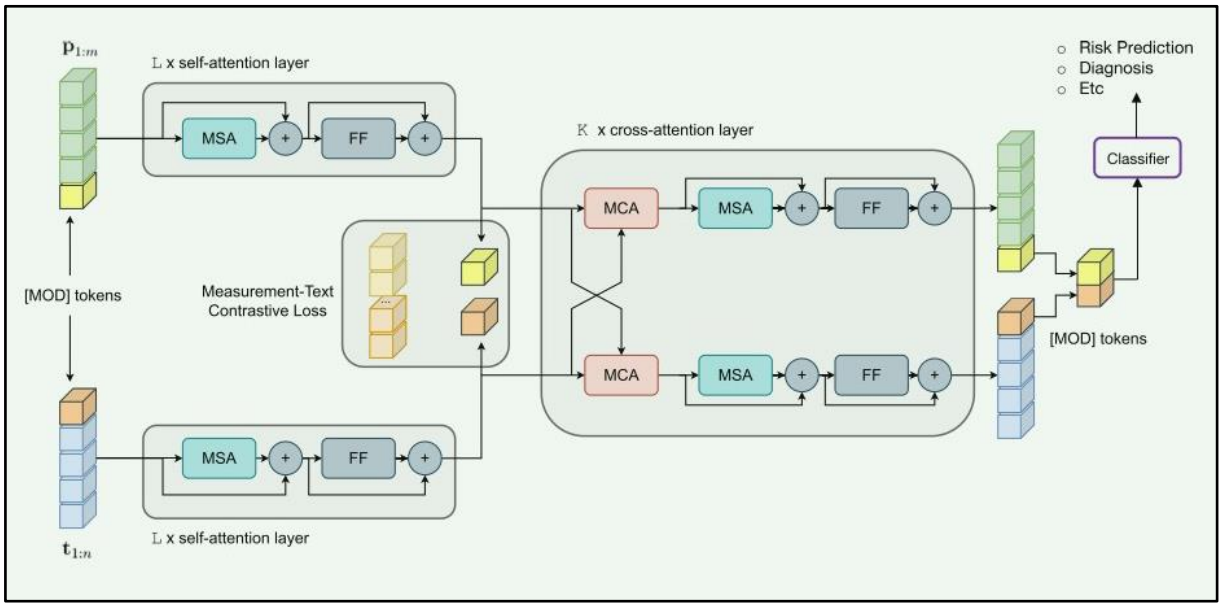

Рисунок 6 – Общая архитектура ARMOUR

Вывод по первой главе

В ходе анализа предметной области был проведен обзор литературы и готовых решений. Было выявлено, что задача классификации легочных заболеваний актуальна и методы решений постоянно улучшаются. Рассмотренные работы демонстрируют, что мультимодальный подход повышает точность классификации.

2. НАБОР ДАННЫХ

2.1. Описание набора данных

В качетве набора данных в работе была использована база Medical Information Mart for Intensive Care (MIMIC IV) [11], которая содержит данные медицинских карт 160 000 уникальных пациентов. Также была использована база данных MIMIC-Chest-Xray [12], связанная с MIMIC IV, предоставляющая соответствующие рентгеновские снимки тех же пациентов. Все данные являются общедоступными через платформу PhysionNet [13]. В MIMIC-Chest-Xray хранятся 37 254 снимков с уникальными названиями типа guid. Структура файлов баз данных MIMIC IV и MIMIC-Chest-Xray привидены в листингах 1 и 2 соответственно.

Листинг 1 – Структура файлов и директорий базы MIMIC

```
tree mimic-iv-1.0 /f  /a
mimic-iv-1.0
|   LICENSE.txt
|   SHA256SUMS.txt
+---core
|       admissions.csv.gz
|       patients.csv.gz
|       transfers.csv.gz
+---hosp
|       diagnoses_icd.csv.gz
|       drgcodes.csv.gz
|       d_hcpcs.csv.gz
|       d_icd_diagnoses.csv.gz
|       d_icd_procedures.csv.gz
|       d_labitems.csv.gz
|       emar.csv.gz
|       emar_detail.csv.gz
|       hcpcsevents.csv.gz
|       labevents.csv.gz
|       microbiologyevents.csv.gz
|       pharmacy.csv.gz
|       poe.csv.gz
|       poe_detail.csv.gz
|       prescriptions.csv.gz
|       procedures_icd.csv.gz
|       services.csv.gz
\---icu
        chartevents.csv.gz
        datetimeevents.csv.gz
        d_items.csv.gz
        icustays.csv.gz
        inputevents.csv.gz
        outputevents.csv.gz
        procedureevents.csv.gz
```

Листинг 2 – Структура файлов и директорий базы MIMIC-Chest-Xray

```
tree MIMIC-CXR-JPG /f  /a
MIMIC-CXR-JPG
\---2.0.0
    \---files
        |   index.html
        |   mimic-cxr-2.0.0-chexpert.csv
        |   mimic-cxr-2.0.0-metadata.csv
        |   mimic-cxr-2.0.0-negbio.csv
        |   mimic-cxr-2.0.0-split.csv
        |   mimic-cxr-ehr-split.csv
        |
        +---other files
        |       index.html
        |       LICENSE.txt
        |       mimic-cxr-2.0.0-chexpert.csv
        |       mimic-cxr-2.0.0-chexpert.csv.gz
        |       mimic-cxr-2.0.0-metadata.csv
        |       mimic-cxr-2.0.0-metadata.csv.gz
        |       mimic-cxr-2.0.0-negbio.csv
        |       mimic-cxr-2.0.0-negbio.csv.gz
        |       mimic-cxr-2.0.0-split.csv
        |       mimic-cxr-2.0.0-split.csv.gz
        |       README
        |       robots.txt
        |       SHA256SUMS.txt
        |
        +---p10
        |   |   index.html
        |   |
        |   +---p10000032
        |   |   |   index.html
        |   |   |
        |   |   +---s50414267
        |   |   |       02aa804e-bde0afdd-112c0b34-7bc16630-4e384014.jpg
        |   |   |       174413ec-4ec4c1f7-34ea26b7-c5f994f8-79ef1962.jpg
        |   |   |       index.html
        |   |   |
        |   |   +---s53189527
        |   |   |       2a2277a9-b0ded155-c0de8eb9-c124d10e-82c5caab.jpg
```

База данных MIMIC предназначена для исследования различных задач: смертность при госпитализации, фенотипирование и декомпенсация, и для получения наборов каждой задачи необходимо произвести предварительное извлечение данных. Для извлечения эталонных наборов данных из общей базы использовался код репозитория [14], который является дополненной версией общедоступного кода для MIMIC III [15]. Сначала для каждого пациента создается отдельный каталог, в который записывается информация о пребывании в отделении интенсивной терапии в stays.csv, диагнозы в diagnoses.csv, события в events.csv и временной ряд событий в

episode<номер>_timeseries.csv. В листинге 3 приведен пример каталога пациента.

Листинг 3 – Каталог пациента

```
tree 1*******0 /f /a
    diagnoses.csv
    episode1.csv
    episode1_timeseries.csv
    events.csv
    stays.csv
```

Затем следует разделение всего набора на обучающий и тестовый, в которые входят 37 637 и 9 409 каталогов пациентов соответственно. Для всех трех задач данное разделение является одинаковым. Далее идет создание набора для конкретной задачи. После чего создается новая директория, в которую входят также два каталога train и test, содержащие перечень записей из отделения интенсивной терапии, а также файл listfile.csv, хранящий информацию о выживаемости каждого пациента в наборах. После были созданы файлы listfile_train.csv, listfile_val.csv и listfile_test.csv. Итоговый каталог для задачи in-hospital-mortality представлен в листинге 4.

Листинг 4 – Итоговый каталог для задачи in-hospital-mortality

```
tree in-hospital-mortality /f /a
IN-HOSPITAL-MORTALITY
|   test_listfile.csv
|   train_listfile.csv
|   val_listfile.csv
|
+---test
|       10001884_episode1_timeseries.csv
|       10002155_episode1_timeseries.csv
|       ...
|       10002348_episode1_timeseries.csv
|       listfile.csv
+---train
|       10302384_episode1_timeseries.csv
|       10232345_episode1_timeseries.csv
|       ...
|       12342348_episode1_timeseries.csv
|       listfile.csv
```

2.2. Предобработка данных медицинских карт

В данной работе будут использоваться данные для задачи бинарной классификации смертности в отделении интенсивной терапии после первых 48 часов пребывании в нем. В дальнейшем эти данные будут сопоставляться

с последними рентгеновскими снимками, полученные во время пребывания в этом отделении. Пациенты, которые находились в отделении меньше времени, будут исключены из набора.

Рентгеновские снимки и данные медициских карт были извлечены из базы данных MIMIC и предварительно обработаны. Подмножество данных MIMIC содержали миллионы клинических испытаний, соответствующих 17 клиническим параметрам. В таблице 3 представлен обзор клинических параметров.

Таблица 3 – Обзор клинических параметров

Клинический параметр	**Тип поля**	**Отсутствие данных у части пациентов, %**
Скорость наполнения капилляров	категориальный	100
Диастолическое артериальное давление	значение	0,04
Фракция вдыхаемого кислорода	значение	26
Шкала комы Глазго: открытие глаз	категориальный	0
Шкала комы Глазго – двигательная реакция	категориальный	0
Шкала комы Глазго – вербальный ответ	категориальный	0
Шкала комы Глазго – всего	категориальный	100
Глюкоза	значение	0,02
Частота сердцебиения	значение	0
Высота тела	значение	97,7
Среднее артериальное давление	значение	0
Насыщение кислородом	значение	0
Частота дыхания	значение	0
Систолическое артериальное давление	значение	0,04
Температура	значение	2,28
Вес тела	значение	9,13
pH	значение	13,86

Из таблицы 3 можно сделать вывод, что параметры: скорость наполнения капилляров и шкалы комы Глазко (всего) отсутствуют у всех пациентов, поэтому дальнейшее исследование проводилось на 15 параметрах, в которые входят 3 категориальных и 12 непрерывных.

Для оцифровки категориальных данных были добавлены дополнительные поля, в которые записывалась цифра «1», если данные соответствовали категории, в противном случае «0». Для непрерывных данных, такие

как показания давления применялась нормализация. Всего получилось 76 полей, которые представлены в таблице 1 приложения А.

Затем брались показания с интервалом времени 1 час, в итоге для каждого пациента получился массив медицинских данных размером 48x76, затем данные проходили этап нормализации. Примеры нормализованных временных рядов представлены на рисунках 1–4 приложения Б.

Для создания набора табличных медицинских данных был реализован класс `EHRdataset`, наследующийся от класса Dataset библиотеки torch [16], в листинге 5 представлена реализация.

Листинг 5 – Реализация класса `EHRdataset`

```
class EHRdataset(Dataset):
    def __init__(self, listfile, dataset_dir, return_names=True,
period_length=48.0):
        self.return_names = return_names
        self.discretizer = Discretizer(timestep = 1.0,
        store_masks=True,
        impute_strategy='previous',
        start_time='zero',
            config_path='discretizer_config.json')

        discretizer_header =
self.discretizer.transform(read_timeseries())[1].split(',')
        cont_channels = [i for (i, x) in enumerate(discretizer_header) if
x.find("->") == -1]
        self.normalizer = Normalizer(fields=cont_channels)
        self.normalizer.load_params(
'ph_ts1.0.input_str_previous.start_time_zero.normalizer')
        ...

    def __getitem__(self, index, time_bound=None):
        if isinstance(index, int):
            index = self.names[index]
        ret = self.read_by_file_name(index, time_bound)
        data = ret["X"]
        ts = ret["t"] if ret['t'] > 0.0 else self._period_length
        ys = ret["y"]
        names = ret["name"]
        data = self.discretizer.transform(data, end=ts)[0]
        if (self.normalizer is not None):
            data = self.normalizer.transform(data)
        ys = np.array(ys, dtype=np.int32) if len(ys) > 1 else np.array(ys,
dtype=np.int32)[0]
        return data, ys


    def __len__(self):
        return len(self.names)
```

Для инициализации класса необходимо передать в него список, хранящий информацию о выживаемости (listfile_train.csv, listfile_val.csv или listfile_test.csv). Получение элемента и метки происходит после ранее описанной предобработки с помощью метода `__getitem__`.

2.3. Предобработка рентгеновских снимков

Рентгеновские снимки сначала приводились к общему размеру – 384x384, после чего для расширения набора применялся ряд аугментаций с последующей нормализацией. Функция для трансформации снимков приведена в листинге 6.

Листинг 6 – Функция для трансформации снимков `get_transforms`

```
def get_transforms(cfg):
    normalize = transforms.Normalize([0.485, 0.456, 0.406],
                                      [0.229, 0.224, 0.225])
    train_transforms = []
    train_transforms.append(transforms.Resize(384))
    train_transforms.append(transforms.RandomHorizontalFlip())
    train_transforms.append(transforms.RandomAffine(degrees=45,
            scale=(.85, 1.15), shear=0, translate=(0.15, 0.15)))
    train_transforms.append(transforms.CenterCrop(384))
    train_transforms.append(transforms.ToTensor())
    train_transforms.append(normalize)


    test_transforms = []
    test_transforms.append(
            transforms.Resize(cfg.dataset.transforms.resize))
    test_transforms.append(transforms.CenterCrop(
            cfg.dataset.transforms.crop))
    test_transforms.append(transforms.ToTensor())
    test_transforms.append(normalize)
    return train_transforms, test_transforms
```

Аугментация и трансформация снимков проводилась с помощью библиотеки PyTorch [17]. В частности, были использованы следующие функции.

1. Функция `transforms.Resize` – изменение входного изображения до заданного размера.

2. Функция `transforms.RandomHorizontalFlip` – отзеркаливание изображения.

3. Функция `transforms.RandomAffine` – применение случайных аффинных преобразований.

4. Функция `transforms.CenterCrop` – обрезка изображения по краям.

5. Функция `transforms.ToTensor` – преобразование изображения в тензор.

6. Функция `transforms.Normalize` – нормализация тензорного изображения со средним и стандартным отклонением.

Для тестовой и валидационной выборки горизонтальное отзеркаливание и случайные аффинные преобразования не применялись.

На рисунке 7 представлена визуализация предобработки входного изображения.

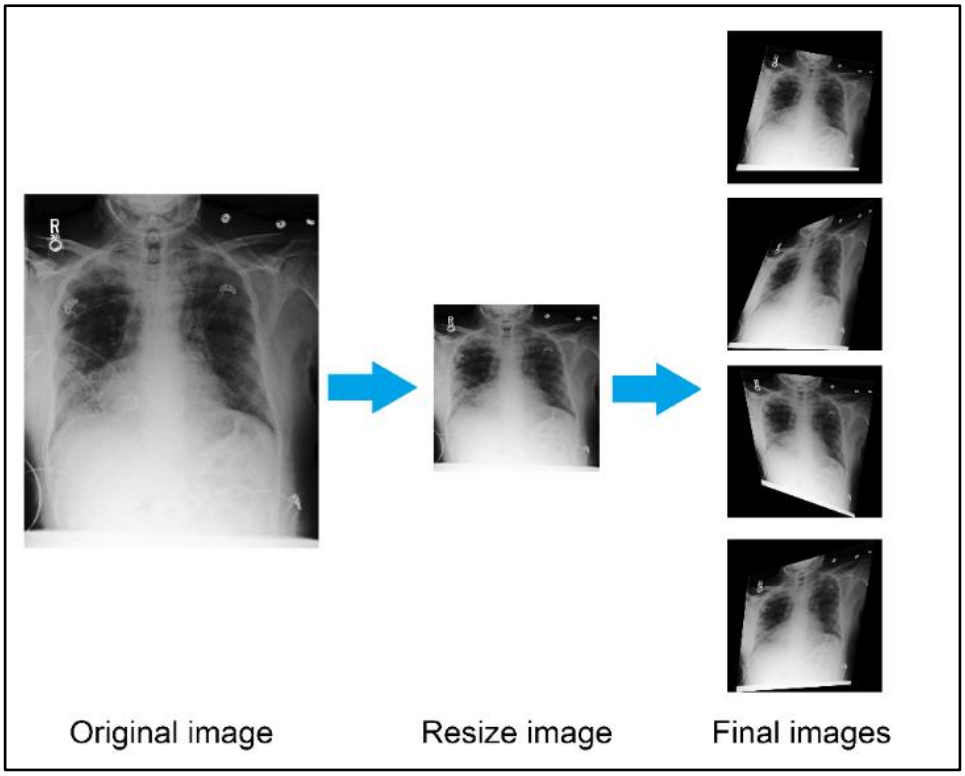

Рисунок 7 – Предобработка входного изображения

Для создания набора рентгеновских снимков был реализован класс `MIMICCXR`, представленный в листинге 7.

Листинг 7 – Реализация класса `MIMICCXR`

```
class MIMICCXR(Dataset):
    def __init__(self, paths, cfg, transform=None, split='train'):
        self.data_dir = cfg.dataset.cxr_data_dir
        self.cfg = cfg
        ...
```

```
    def __getitem__(self, index):
        if isinstance(index, str):
            img = Image.open(self.filenames_to_path[index]).convert('RGB')
            labels = torch.tensor(self.filesnames_to_labels[index]).float()

            if self.transform is not None:
                img = self.transform(img)
            return img, labels
        filename = self.filenames_loaded[index]
        img = Image.open(self.filenames_to_path[filename]).convert('RGB')
        labels = torch.tensor(self.filesnames_to_labels[filename]).float()
        if self.transform is not None:
            img = self.transform(img)
        return img, labels

    def __len__(self):
        return len(self.filenames_loaded)
```

Метод `__getitem__` возвращает предварительно обработанное тензорное изображение и метки заболеваний.

2.4. Получение объединенного датасета

Класс `MIMIC_CXR_EHR` объединяет данные двух модальностей, принимая на вход два датасета и таблицу, в которой объединяются метаданные, находящиеся в MIMIC IV и в MIMIC_CXR. Реализация класса `MIMIC_CXR_EHR` представлена в листинге 8.

Листинг 8 –Реализация класса `MIMIC_CXR_EHR`

```
class MIMIC_CXR_EHR(Dataset):
    def __init__(self, metadata_with_labels, ehr_ds, cxr_ds,
split='train'):
        self.metadata_with_labels = metadata_with_labels
        self.cxr_files_paired = self.metadata_with_labels.dicom_id.values
        self.ehr_files_paired = (self.metadata_with_labels['stay'].values)
        self.ehr_ds = ehr_ds
        self.cxr_ds = cxr_ds
        self.split = split


    def __getitem__(self, index):
        meta_info = {}
        age = None
        gender = None
        ethnicity = None
        ehr_data, labels_ehr = self.ehr_ds[self.ehr_files_paired[index]]
        cxr_data, labels_cxr = self.cxr_ds[self.cxr_files_paired[index]]
        meta_info['id_ehr'] = self.ehr_files_paired[index]
        meta_info['id_cxr'] = self.cxr_files_paired[index]
        age = self.metadata_with_labels.iloc[index]['age'] / 91.0 # Age is
capped to 91 in MIMIC
        gender = [1 if key == self.metadata_with_labels.iloc[index][
'gender'] else 0 for key in GENDER.keys()]
        ethnicity = [1 if key == self.metadata_with_labels.iloc[
```

```
index]['ethnicity']
else 0 for key in ETHNICITY.keys()]
        return ehr_data, cxr_data, labels_ehr, labels_cxr, meta_info, age,
gender, ethnicity

    def __len__(self):
        return len(self.ehr_files_paired)
```

Метод __getitem__ возвращает медицинские данные размерности 48x76, тензорное изображение 3x384x384, бинарное значение выживаемости, массив с метками диагнозов, названия файла снимка и временного ряда, возраст, пол и информацию об этнической принадлежности. В данной работе будут использоваться только первые три значения. В итоге получилось три выборки, в таблице 4 представлено описание содержания выборок.

Таблица 4 – Описание содержание выборок датасета

Наборы	Число пациентов	Число выживших	Число умерших
train	4 832	4 123	709
val	536	461	75
test	1 358	1 151	207
Всего	6 726	5 735	991

В каждой выборке содержится около 15% погибших пациентов, что говорит нам о несбалансированности данных. Для решения этой проблемы можно убрать часть выживших пациентов, но это приведет к сильному уменьшению общей выборки, поэтому принято решение использовать метрику, которая нивелируют несбалансированность, в частности показатель ROC AUC, позволяющий оценить качество бинарной классификации.

Вывод по второй главе

В данной главе была описана база клинических данных MIMIC, из которой были извлечены данные для классификации смертности пациентов в отделении интенсивной терапии. В дальнейшем были организованы 3 выборки для обучения, валиации и тестирования нейронной сети. Также была реализована генерация дополнительный снимков, используя различные аугментации.

3. ПОСТРОЕНИЕ НЕЙРОСЕТЕВОЙ МОДЕЛИ

3.1. Разработка общего принципа модели

В данной работе будет рассмотрен вопрос реализации модели классификации с возможностью промежуточного слияния эмбеддингов, полученных из клинических данных и рентгеновских снимков. На основе анализа подобных реализаций была построена общая схема классификатора, представленная на рисунке 8.

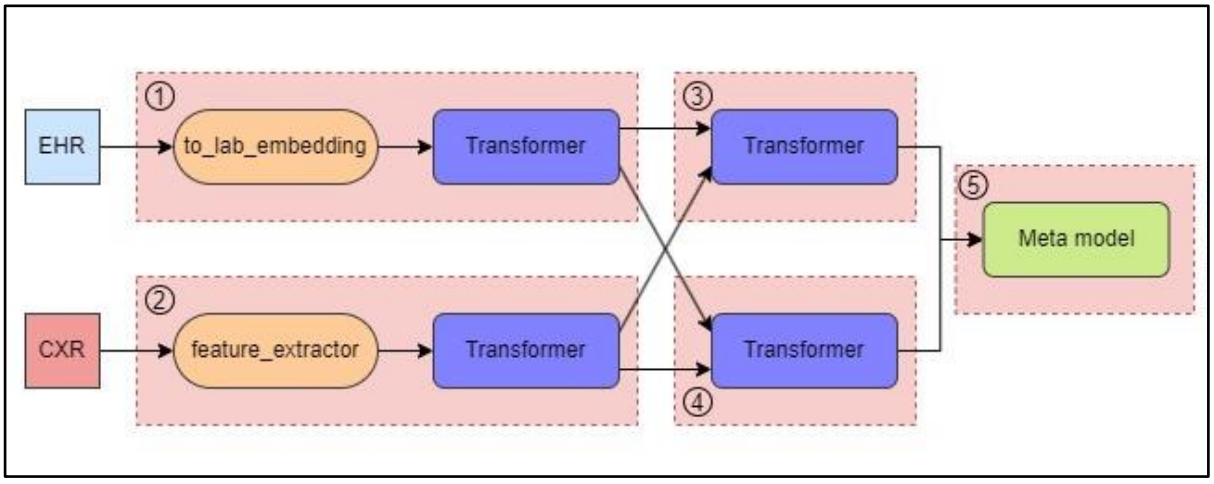

Рисунок 8 – Общая схема модели

На общей схеме модель разделена на 5 блоков, номера которых обозначены в кружках. Данные медицинских карт и снимки поступают в блоки 1 и 2 соответственно, на выходе которых получаются два эмбеддинга. Затем следует объединение частей эмбеддингов, как представлено на рисунке 3. Дополненные эмбеддинги поступают на вход блоков 3 и 4, на выходе из которых получаются два новых эмбеддинга, после полного объединения они поступают в блок 5 для классификации. Ниже будут более подробно рассмотрены все элементы из общей схемы.

В блоке 1 и 2 перед подачей данных в трансформеры они преобразуются в эмбеддинги, с помощью элементов to_lab_embedding и feature_extractor. Для получения эмбеддингов из медицинских данных применяется один линейный слой, а для снимков используется предобученная модель [18].

Схема получения эмбеддингов из медицинских данных и снимков представлена на рисунке 9.

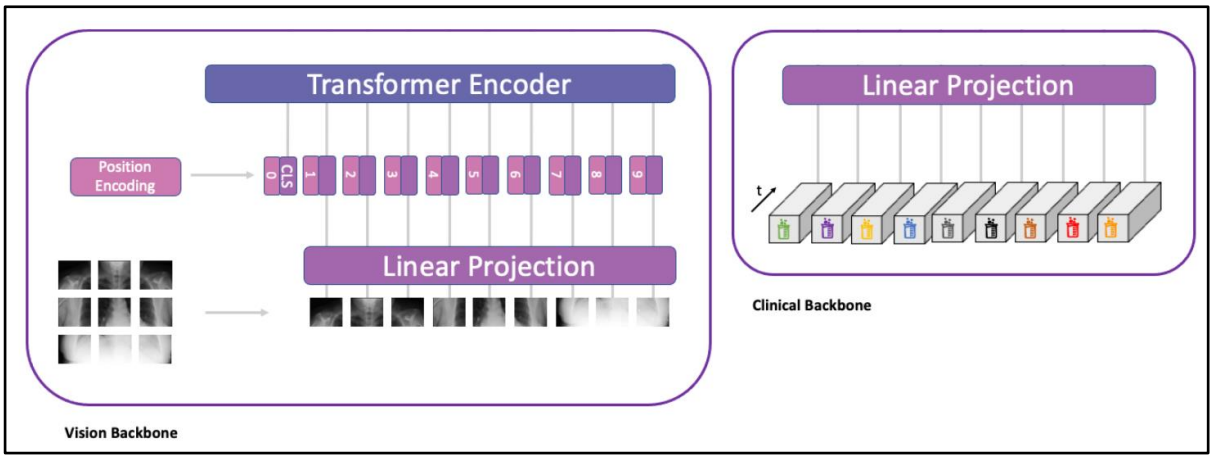

Рисунок 9 – Схема получения эмбеддингов из медицинских данных и снимков

Модель трансформера взята из репозитория [19], которая состоит из последовательности модулей, содержащие слой внимания и слой прямой связи. Реализация трансформера представлена в листинге 9.

Листинг 9 – Реализация трансформера

```
class Transformer(nn.Module):
    def __init__(self, dim, depth, heads, dim_head, mlp_dim, dropout = 0.):
        super().__init__()
        self.layers = nn.ModuleList([])
        for _ in range(depth):
            self.layers.append(nn.ModuleList([
                PreNorm(dim, Attention(dim,
                             heads = heads,
                             dim_head = dim_head,
                             dropout = dropout)),
                PreNorm(dim, FeedForward(dim,
                                         mlp_dim,
                                         dropout = dropout))
                                             ])
                                    )

    def forward(self, x):
        for attn, ff in self.layers:
            x = attn(x) + x
            x = ff(x) + x

        return x
```

Общая схема трансформера представлена на рисунке 10.

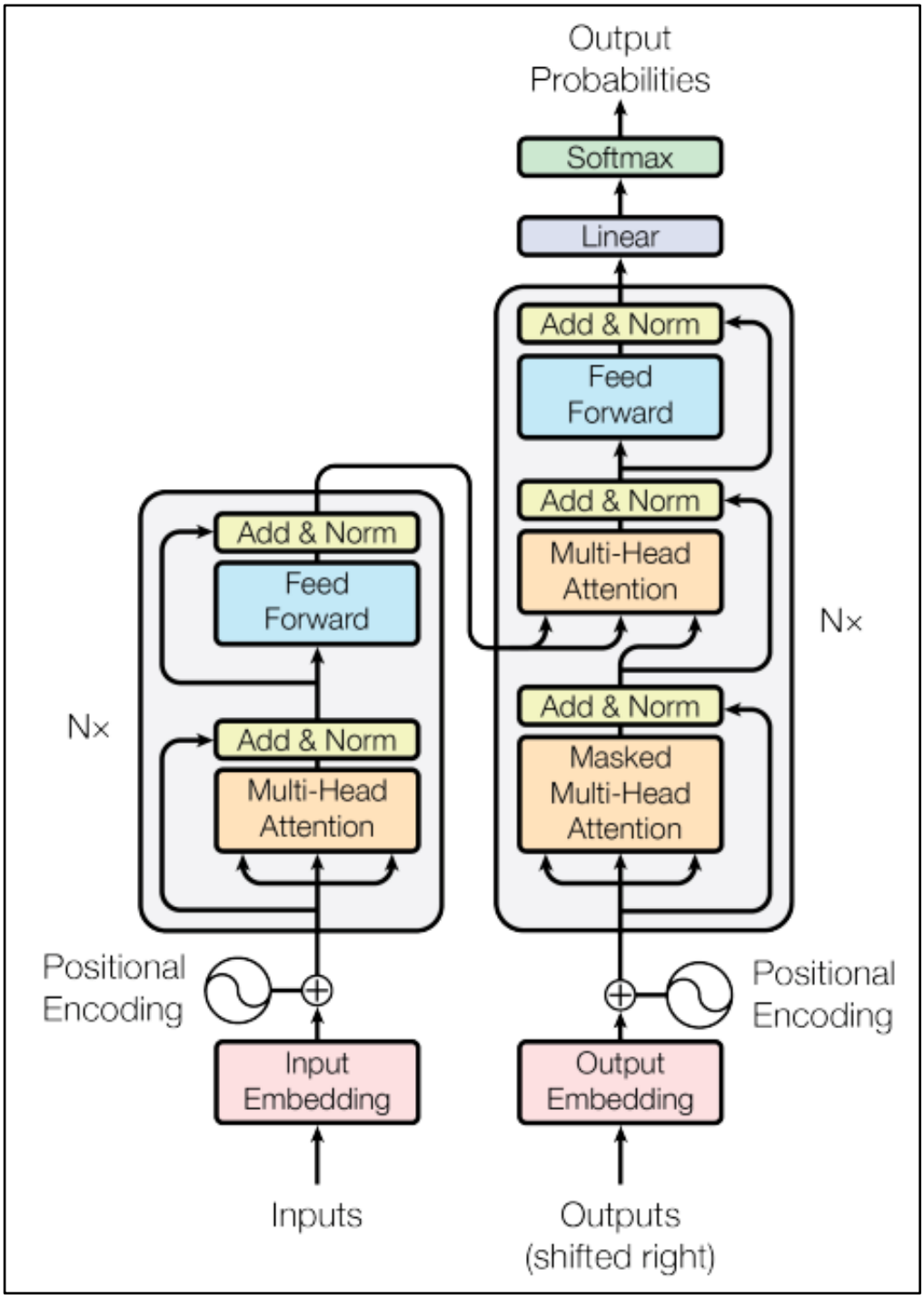

Рисунок 10 – Общая схема трансформера

Трансформеры, показанные в общей схеме классификатора одинаковые, изменяется лишь параметр размера входного эмбеддинга. Блок 5, содержащий элемент meta model представляет слой нормализации и линейный слой для классификации.

Для более удобной настройки параметров, обучение каждых 5 блоков проводилось по отдельности и для подсчета ошибки выходные эмбеддинги подавались на блок классификации, который и представляется в блоке 5, после обучения все 3 набора данных подавались на каждый блок, на выходе из которых получались эмбеддинги, которые в свою очередь были обучающей и тестовой выборкой.

3.2. Обучение блоков нейросетевой модели

Для обучения моделей всех 5 блоков использовалась библиотека Pytorch lightning [20], основанная на фреймворке PyTorch. Библиотека упрощает восприятие кода нейронной сети и дает возможность легко запускать его на распределенном оборудовании.

В качестве функции ошибки использовалась BCEWithLogitsLoss из библиотеки PyTorch [21], которая комбинирует сигмоидный слой и BCELoss в один класс. Данная версия более стабильна, чем использование простой сигмоиды, за которой следует BCELoss, поскольку, объединяя операции в один слой используется функция Log-Sum-Exp для числовой стабильности, позволяющая исключить проблему с переполнением float64 из-за большого входного значения.

В формуле (1) приведена функция Log-Sum-Exp:

$$LSE(X) = \log\left(\sum_{j=0}^{N} e^{x_j}\right). \quad (1)$$

Log-Sum-Exp – это последовательное применение к входному аргументу: экспоненты, суммирования и логарифмирования. При попытке вычислить функцию LSE от вектора x, возникнет та же проблема переполнения, поскольку в сумме могут участвовать экспоненты произвольной, возможно большой, степени. Путем преобразований, основанных на свойствах функции можно избавиться от проблемы переполнения. Если принять результат выполнения функции $LSE(x)$ за y и применить экспоненту к обоим

частям уравнения, то из каждого слагаемого можно вынести e^c, из чего получается формула (2):

$$y = c + \log\left(\sum_{j=0}^{N} e^{x_j - c}\right) = LSE(x). \quad (2)$$

На рисунке 11 продемонстрирован пример применения исходной функции LSE и модернизированной.

```python
import numpy as np

x = np.array([11, 12, -1000, 5, 10, 0.001])
y = np.array([-1000, 1000, -1000, 5, 10, 0.001])

def LSE_initial(x):
    return np.log(np.sum(np.exp(x)))

def LSE_modified(x):
    c = np.max(x)
    return c + np.log(np.sum(np.exp(x - c)))

# с экспонентами, не приводящими к переполнению
print('Исходная LSE(x): ', LSE_initial(x))
print('Преобразованная LSE(x): ', LSE_modified(x))

# с экспонентой в 1000-й степени
print('Исходная LSE(y): ', LSE_initial(y))
print('Преобразованная LSE(y): ', LSE_modified(y))
```

```
Исходная LSE(x):  12.408216490736713
Преобразованная LSE(x):  12.408216490736713
Исходная LSE(y):  inf
Преобразованная LSE(y):  1000.0
C:\Users\Artem\AppData\Local\Temp\ipykernel_16260\2652174690.py:7: RuntimeWarning: overflow encountered in exp
  return np.log(np.sum(np.exp(x)))
```

Рисунок 11 – Применение функций LSE

Видно, что даже в случае, когда одно из слагаемых обращается в float inf, модифицированный вариант $LSE(x)$ дает верный результат.

Из таблицы 4 видно, что имеющиеся данные не однородные, поэтому в качестве метрики был взят показатель roc_auc_score из библиотеки sklearn [22], рассчитывающий площадь под ROC кривой.

Кривая ROC – это график, который иллюстрирует производительность классификационной модели при всех возможных порогах классификации. Ось X данного графика представляет собой FPR, то есть ложноположительную частоту, а ось Y – TRP, то есть истинно положительную частоту.

TPR – доля правильно классифицированных положительных результатов относительно всех положительных результатов в данных

TPR также известен как Recall считается по формуле (3):

$$TPR = \frac{TP}{TP + FN}, \quad (3)$$

где TP – истинно положительные результаты;

FN – ложно отрицательные результаты.

FPR определяет долю ошибочно классифицированных отрицательных результатов относительно всех отрицательных результатов и вычисляется по формуле (4):

$$FPR = \frac{FP}{TN + FP}, \quad (4)$$

где FP – ложно положительные результаты;

TN – истинно отрицательные результаты.

Кривая ROC представляет собой графическое представление компромисса между чувствительностью и специфичностью при различных порогах классификации. Идеальная модель классификации будет стремиться к точке в верхнем левом углу графика, где TPR равно 1, а FPR равно 0.

Показатель AUC (Area Under the Curve) – это мера, которая позволяет суммировать производительность модели одним числом, измеряя площадь под кривой ROC. AUC колеблется от 0 до 1, где более высокое значение AUC указывает на более высокую производительность модели. AUC равный 0,5 указывает на отсутствие дискриминационной способности модели, тогда как AUC равный 1,0 означает идеальное различие классов.

Важные свойства AUC – это инвариантность к порогу классификации и масштабу предсказаний. Инвариантность к масштабу предсказаний означает, что AUC не зависит от масштаба вероятностей, которые генерирует модель. Например, две модели могут выдавать предсказания в различных масштабах, одна – в виде вероятностей от 0 до 1, а другая – в виде более широкого диапазона значений. Несмотря на эти различия, AUC будет одинаковым, если порядок ранжирования случаев от наиболее вероятного положительного до наиболее вероятного отрицательного сохраняется.

Логирование происходило с помощью сервиса Wandb [23], позволяющий визуализировать многие процессы, такие как: изменения весов нейронной сети, нагрузку CPU и GPU и других параметров в реальном времени. Вся обработка визуализации происходит на сервере, что позволяет не нагружать собственное оборудование.

Для реализации и обучения моделей была выбрана среда разработки PyCharm.

PyCharm – это кроссплатформенная интегрированная среда разработки для языка программирования Python, разработанная компанией JetBrains на основе IntelliJ IDEA, предоставляющий пользователю комплекс средств для написания кода и визуальный отладчик [24].

Для ускорения процесса обучения моделей был использован графический ускоритель NVIDIA GTX 1080.

На рисунках 12–13 представлены графики изменения метрики AUC на валидационном наборе при обучении блока 1 и 2 соответственно.

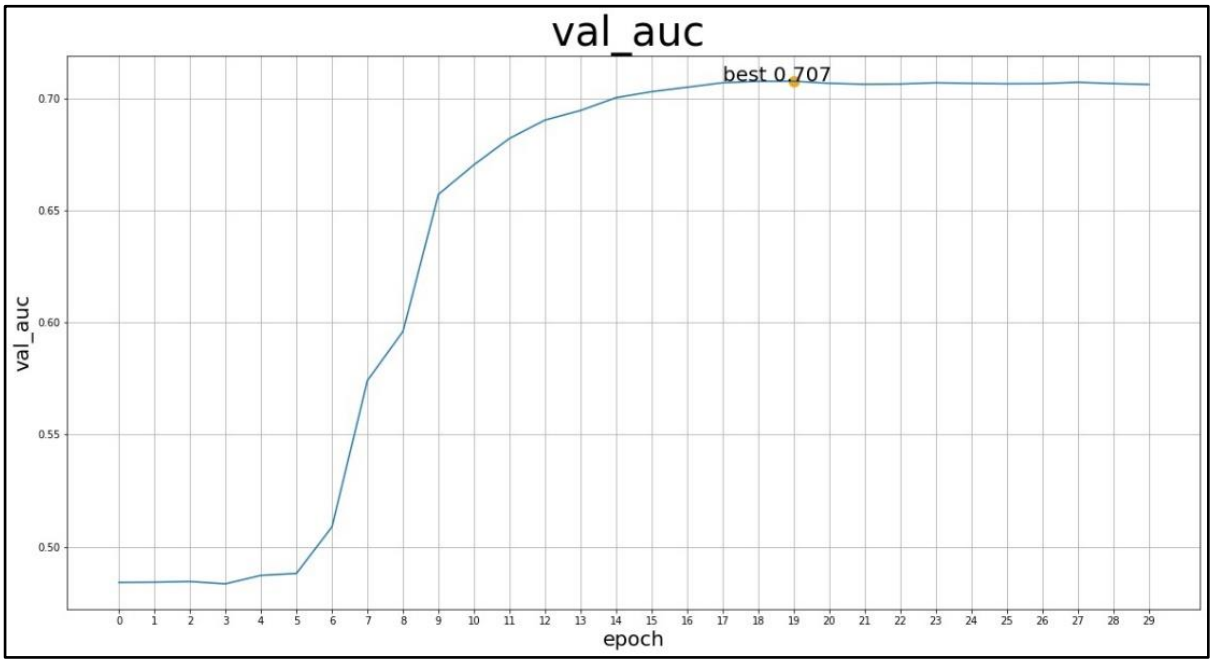

Рисунок 12 – График изменения AUC на валидационном наборе при обучении блока 1

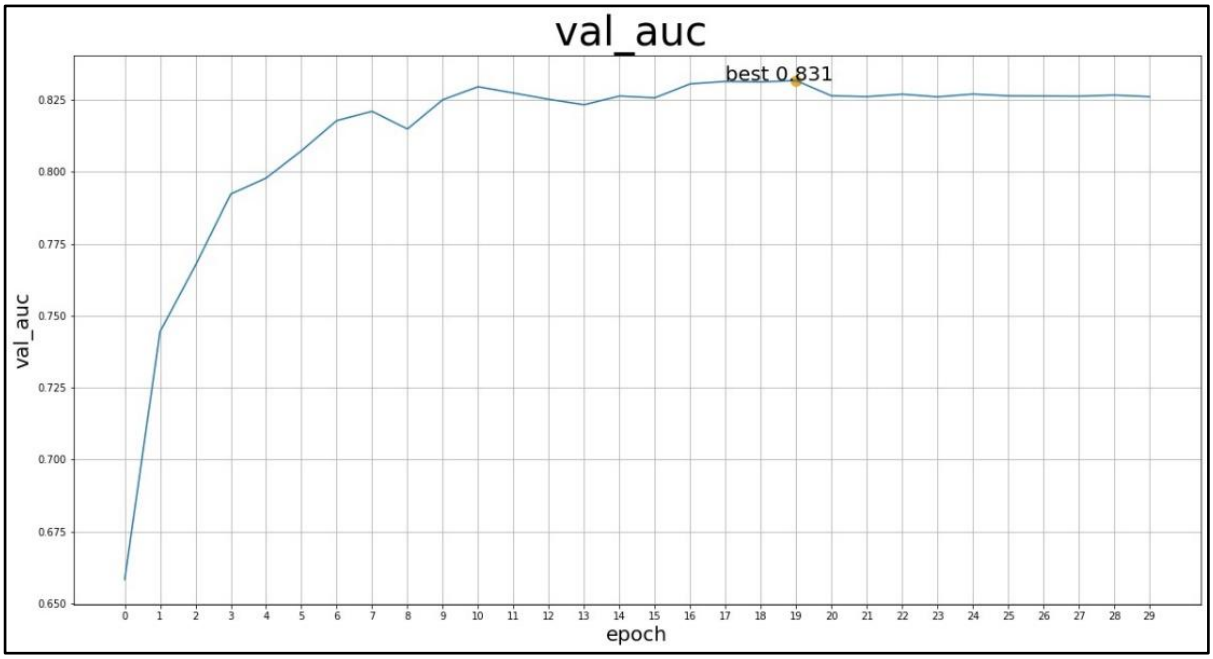

Рисунок 13 – График изменения AUC на валидационном наборе при обучении блока 2

Из графиков видно, что лучшие показатели метрики AUC для блоков 1 и 2 получились на 20 эпохе и составили 0,707 и 0,831 соответственно, далее уже шло переобучение. При получении эмбеддингов для следующих блоков будут использоваться веса моделей, получившиеся на 20 эпохе. Аналогично будет проходить обучение остальных блоков.

На рисунках 1–6 приложения В представлены графики изменения метрики и ошибки при обучении блока 1 и 2 на тренировочном наборе, из которых видно, что после 20 эпохи данные показатели изменялись незначительно.

Далее с помощью обученных моделей были получены эмбеддинги всех наборов для обучения блоков 3 и 4. Полученные эмбеддинги из первых двух блоков имеют одинаковые размерности для упрощения объединения и составляют 1x384. На рисунке 14 представлена схема объединения эмбеддингов.

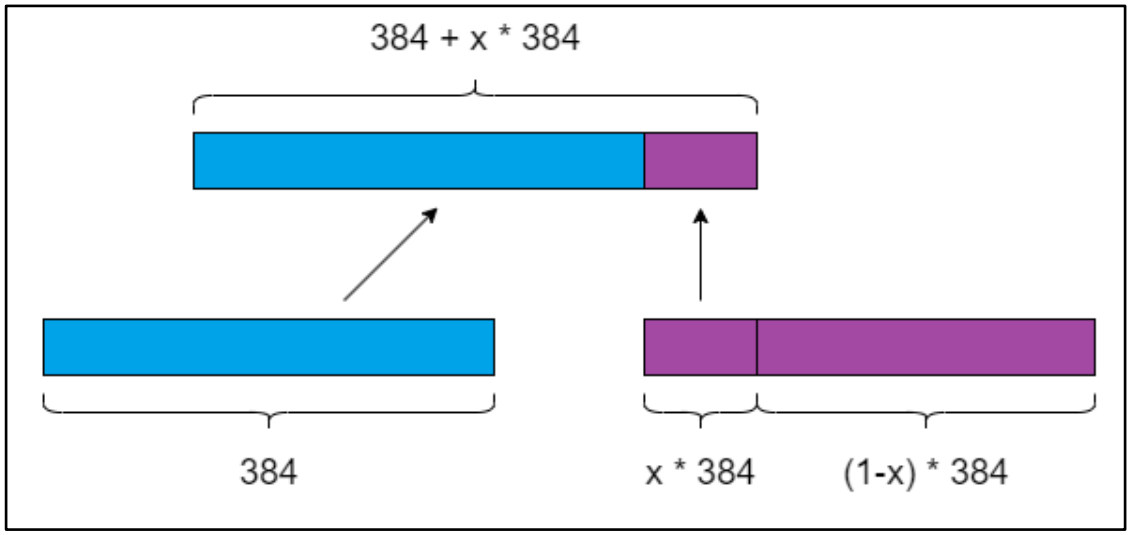

Рисунок 14 – Схема объединения эмбеддингов для блока 3 и 4

На рисунке 14 переменная x обозначает величину части присоединяемого эмбединга от другой модальности, в данном исследовании x принимает значения 0,3 и 0,5.

Обучение блоков 3–4 проводилось по 50 эпох, на рисунках 7–10 приложения В представлены графики изменения показателя AUC на валидационных наборах. В таблице 5 представлены показатели ошибки и AUC для тестовых наборов.

Таблица 5 – Результаты показателей ошибки и AUC для тестовых наборов

Показатели	**ehr**	**cxr**	**0.3cxr+ehr**	**cxr+0.3ehr**	**0.5cxr+ehr**	**cxr+0.5ehr**
loss	0,503	0,402	0,333	0,337	0,333	0,335
AUC	0,704	0,801	0,831	0,831	**0,837**	**0,832**

Наилучшие результаты блоков 3 и 4 оказались при конфигурации, на вход которых поступали эмбеддинги, к которым добавлялось половина эмбеддинга от другой модальности. На выходных данных блоков 3–4 будет обучаться блок 5.

Графики изменения показателя AUC при обучении блока 5 представлены на рисунках 11–12 приложения В. В таблице 6 представлены показатели ошибки и AUC для тестовых наборов.

Таблица 6 – Результаты показателей ошибки и AUC для тестовых наборов блока 5

Показатели	**cxr+ehr_0,3**	**cxr+ehr_0,5**
loss	0,337	0,336
AUC	**0,838**	0,836

По данным таблицы 6 можно сделать вывод, что итоговый результат немного улучился по сравнению с результатми блоков 3 и 4.

Также в ходе работы был сделан экспиримент, при котором промежуточный обмен эмбедингов не производился, и в meta model подавались объединенные эмбединги после блока 1 и 2. Схема модели без перекрестного обмена приведена на рисунке 15.

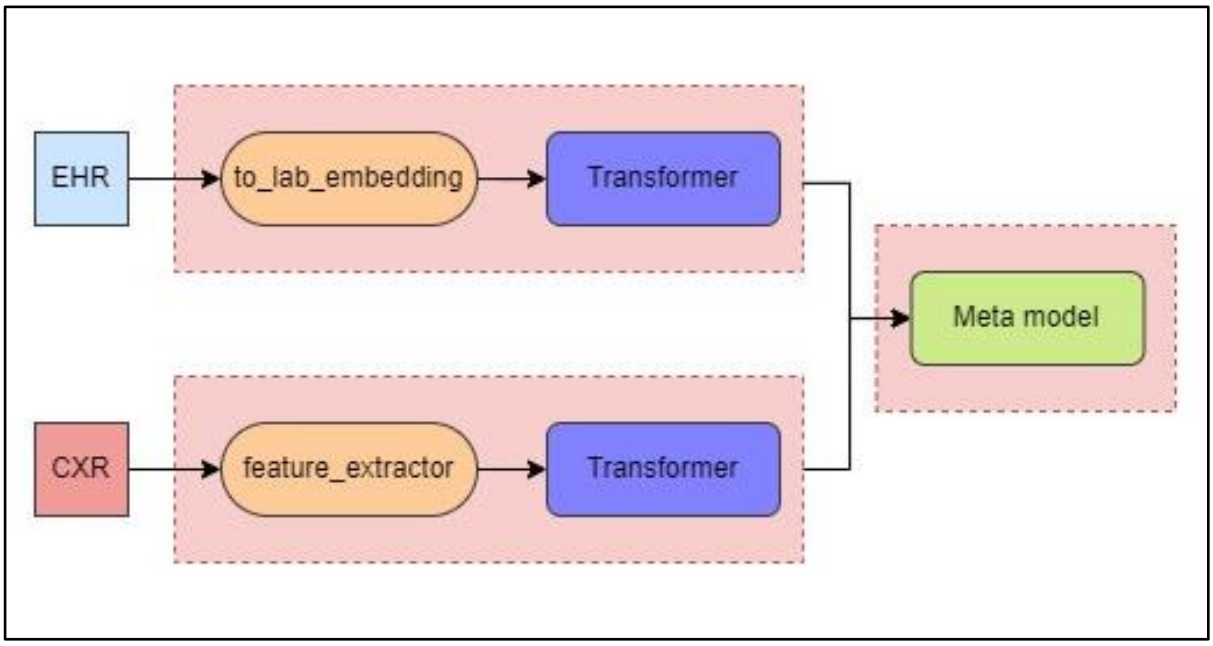

Рисунок 15 – Схема нейросетевой модели без перекрестного обмена эмбеддингами

Показатели loss и AUC для тестовой выборки составили 0,329 и 0,844 соответственно. Показатель AUC оказался на 0,006 больше наилучшего результата, получившегося при использовании перекрестного обмена эмбеддингов. На рисунке 16 представлены ROC кривые для моделей при различных коэфициентах мэппинга.

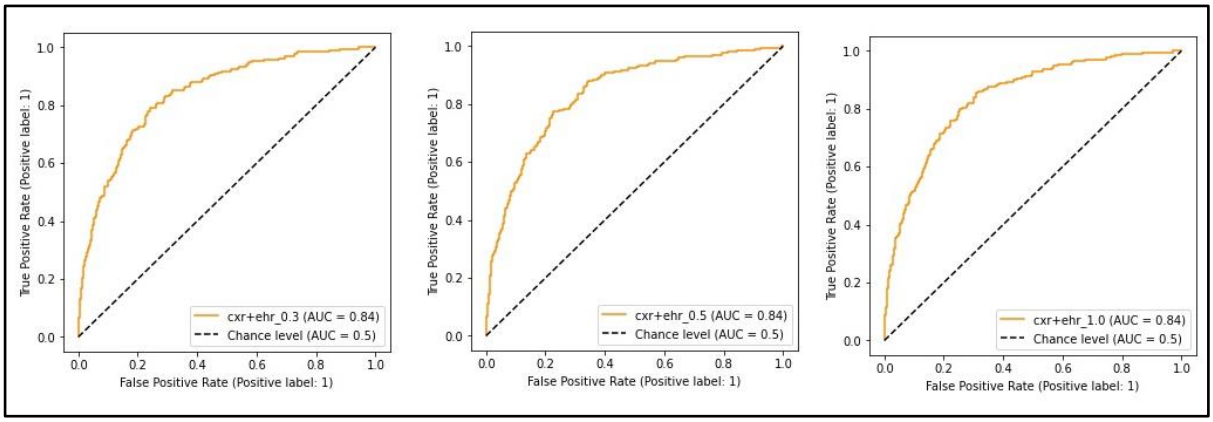

Рисунок 16 – ROC кривые для итоговых моделей

В итоге можно сделать вывод, что при перекрестном обмене эмбеддингов показатели AUC выше по сравнению с одномодальной классификацией, но немного ниже, если эмбеддинги объединить, минуя этап перекрестного мэппинга.

Предположительно улучшить итоговый результат возможно при увеличении обучающей выборки путем генерации дополнительных медицинских данных на основе имеющейся выборки, как это было сделано с рентгеновскими снимками.

Вывод по третьей главе

В данной главе была представлена и реализована общая архитектура нейронной сети с перекрестным мэппингом эмбедингов. Также были выбраны и описаны метрика и функция ошибки, с помощью которых происходило обучение нейронной сети. Для каждого этапа обучения получены результаты классификации тестовой выборки, которые были проанализированы.

ЗАКЛЮЧЕНИЕ

В рамках данной работы была разработана нейронная сеть, которая по данным двух модальностей позволяет предсказать смертельный исход в отделении интенсивной терапии. При этом были решены следующие задачи.

1. Произведен обзор литературы и существующих решений по предметной области.

2. Из большой базы MIMIC извлечены данные для обучения нейронной сети, из которых были подготовлены обучающая и тестовая выборки.

3. Выполнено проектирование и реализация архитектуры нейронной сети для решения задачи классификации.

4. Проведено обучение и тестирование реализованной нейронной сети.

В рамках работы были опубликованы следующие статьи.

1. Ivanova O.N., Melekhin A.V., Ivanova E.V., Kumar S., Zymbler M.L. Intermediate Fusion Approach for Pneumonia Classification on Imbalanced Multimodal Data. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2023. Vol. 12, no. 3. P. 19-30. DOI: 10.14529/cmse230302.

ЛИТЕРАТУРА

1. 10 ведущих причин смерти в мире. [Электронный ресурс] URL: https://www.who.int/ru/news-room/fact-sheets/detail/the-top-10-causes-of-death (дата обращения: 24.02.2024 г.).

2. Диагностика заболеваний легких. [Электронный ресурс] URL: https://inlnk.ru/84pK14 (дата обращения: 24.02.2024 г.).

3. Yu H., Chenzhuang D., Zihui X., Xuanyao C., Hang Z., Longbo H. What Makes Multi-modal Learning Better than Single (Provably). [Электронный ресурс] // arXiv.org. 2021. Дата обновления: 26.10.2021 г. URL: https://arxiv.org/abs/2106.04538 (дата обращения: 10.03.2024 г.).

4. IEMOCAP DATABASE. [Электронный ресурс] URL: https://sail.usc.edu/iemocap (дата обращения: 12.02.2024 г.).

5. Sousa JV, Matos P, Silva F, Freitas P, Oliveira HP, Pereira T. Single Modality vs. Multimodality: What Works Best for Lung Cancer Screening? // Sensors (Basel), Jun 15, 2023. DOI: 10.3390/s23125597.

6. NLST. [Электронный ресурс] URL: https://cdas.cancer.gov/nlst (дата обращения: 13.02.2024 г.).

7. Yikai W., Wenbing H., Fuchun S., Tingyang X., Yu R., Junzhou H. Deep Multimodal Fusion by Channel Exchanging. [Электронный ресурс] // arXiv.org. 2020. Дата обновления: 05.12.2020 г. URL: https://arxiv.org/abs/2011.05005 (дата обращения: 10.03.2024 г.).

8. Image-to-Image Translation. [Электронный ресурс] URL: https://paperswithcode.com/task/image-to-image-translation (дата обращения: 12.03.2024 г.).

9. Renyu Z., Chengcheng H., Yong Q., Qiushi S., Xiang L., Ming G., Xuezhi C., Yunsen X. Exchanging-based Multimodal Fusion with Transformer. [Электронный ресурс] // arXiv.org. 2023. Дата обновления: 05.09.2023 г. URL: https://arxiv.org/abs/2309.02190v1 (дата обращения: 09.04.2024 г.).

10. Jinghui L., Daniel C., Anthony N., Karin V. Attention-based multimodal fusion with contrast for robust clinical prediction in the face of

missing modalities // Journal of Biomedical Informatics, September, 2023. DOI: 10.1016/j.jbi.2023.104466.

11. Mimic IV. [Электронный ресурс] URL: https://physionet.org/content/mimiciv/1.0 (дата обращения: 28.04.2024 г.).

12. Mimic-cxr. [Электронный ресурс] URL: https://physionet.org/content/mimic-cxr/2.0.0 (дата обращения: 28.04.2024 г.).

13. Physionet. [Электронный ресурс] URL: https://physionet.org (дата обращения: 28.04.2024 г.).

14. MIMIC-IV Data Extraction. [Электронный ресурс] URL: https://github.com/nyuad-cai/MedFuse/tree/main/mimic4extract (дата обращения: 01.05.2024 г.).

15. Mimic3-benchmarks. [Электронный ресурс] URL: https://github.com/YerevaNN/mimic3-benchmarks (дата обращения: 01.05.2024 г.).

16. Datasets & DataLoaders. [Электронный ресурс] URL: https://pytorch.org/tutorials/beginner/basics/data_tutorial.html (дата обращения: 02.05.2024 г.).

17. Transforming and augmenting images. [Электронный ресурс] URL: https://pytorch.org/vision/stable/transforms.html (дата обращения: 02.05.2024 г.).

18. Model card for vit_small_patch16_384.augreg_in21k_ft_in1k [Электронный ресурс] URL: https://huggingface.co/timm/vit_small_patch16_384.augreg_in21k_ft_in1k (дата обращения: 01.05.2024 г.).

19. Vit-pytorch. [Электронный ресурс] URL: https://github.com/lucidrains/vit-pytorch/blob/main/vit_pytorch/vit.py (дата обращения: 01.05.2024 г.).

20. Официальная документация Pytorch Lightning. [Электронный ресурс] URL: https://lightning.ai/docs/pytorch/stable//index.html (дата обращения: 01.05.2024 г.).

21. BCEWithLogitsLoss. [Электронный ресурс] URL: https://pytorch.org/docs/stable/generated/torch.nn.BCEWithLogitsLoss.html (дата обращения: 01.05.2024 г.).

22. Sklearn.metrics.roc_auc_score [Электронный ресурс] URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score (дата обращения: 01.05.2024 г.).

23. Wandb. [Электронный ресурс] URL: https://wandb.ai/site (дата обращения: 02.05.2024 г.).

24. PyCharm. [Электронный ресурс] URL: https://www.jetbrains.com/ru-ru/pycharm/ (дата обращения: 02.05.2024 г.).

ПРИЛОЖЕНИЯ

Приложение А. Данные датасета

Поля медицинских параметров после оцифровки приведены в таблице 1.

Таблица 1 – Поля медицинских параметров после оцифровки

№	Название поля	Категория/значение
0	Capillary refill rate	0.0
1	Capillary refill rate	1.0
2	Diastolic blood pressure	value
3	Fraction inspired oxygen	value
4	Glascow coma scale eye opening	To Pain
5	Glascow coma scale eye opening	3 To speech
6	Glascow coma scale eye opening	1 No Response
7	Glascow coma scale eye opening	4 Spontaneously
8	Glascow coma scale eye opening	None
9	Glascow coma scale eye opening	To Speech
10	Glascow coma scale eye opening	Spontaneously
11	Glascow coma scale eye opening	2 To pain
12	Glascow coma scale motor response	1 No Response
13	Glascow coma scale motor response	3 Abnorm flexion
14	Glascow coma scale motor response	Abnormal extension
15	Glascow coma scale motor response	No response
16	Glascow coma scale motor response	4 Flex-withdraws
17	Glascow coma scale motor response	Localizes Pain
18	Glascow coma scale motor response	Flex-withdraws
19	Glascow coma scale motor response	Obeys Commands
20	Glascow coma scale motor response	Abnormal Flexion
21	Glascow coma scale motor response	6 Obeys Commands
22	Glascow coma scale motor response	5 Localizes Pain
23	Glascow coma scale motor response	2 Abnorm extensn
24	Glascow coma scale total	11
25	Glascow coma scale total	10
26	Glascow coma scale total	13
27	Glascow coma scale total	12
28	Glascow coma scale total	15
29	Glascow coma scale total	14
30	Glascow coma scale total	3
31	Glascow coma scale total	5
32	Glascow coma scale total	4
33	Glascow coma scale total	7

Окончание таблицы 1 приложения А

№	Название поля	Категория/значение
34	Glascow coma scale total	6
35	Glascow coma scale total	9
36	Glascow coma scale total	8
37	Glascow coma scale verbal response	1 No Response
38	Glascow coma scale verbal response	No Response
39	Glascow coma scale verbal response	Confused
40	Glascow coma scale verbal response	Inappropriate Words
41	Glascow coma scale verbal response	Oriented
42	Glascow coma scale verbal response	No Response-ETT
43	Glascow coma scale verbal response	5 Oriented
44	Glascow coma scale verbal response	Incomprehensible sounds
45	Glascow coma scale verbal response	1.0 ET/Trach
46	Glascow coma scale verbal response	4 Confused
47	Glascow coma scale verbal response	2 Incomp sounds
48	Glascow coma scale verbal response	3 Inapprop words
49	Glucose	value
50	Heart Rate	value
51	Height	value
52	Mean blood pressure	value
53	Oxygen saturation	value
54	Respiratory rate	value
55	Systolic blood pressure	value
56	Temperature	value
57	Weight	value
58	pH	value
59	mask	Capillary refill rate
60	mask	Diastolic blood pressure
61	mask	Fraction inspired oxygen
62	mask	Glascow coma scale eye opening
63	mask	Glascow coma scale motor response
64	mask	Glascow coma scale total
65	mask	Glascow coma scale verbal response
66	mask	Glucose
67	mask	Heart Rate
68	mask	Height
69	mask	Mean blood pressure
70	mask	Oxygen saturation
71	mask	Respiratory rate
72	mask	Systolic blood pressure
73	mask	Temperature
74	mask	Weight
75	mask	pH

Приложение Б. Временные ряды медицинских данных

Визуализация временных рядов медицинских данных приведены на рисунках 1–4.

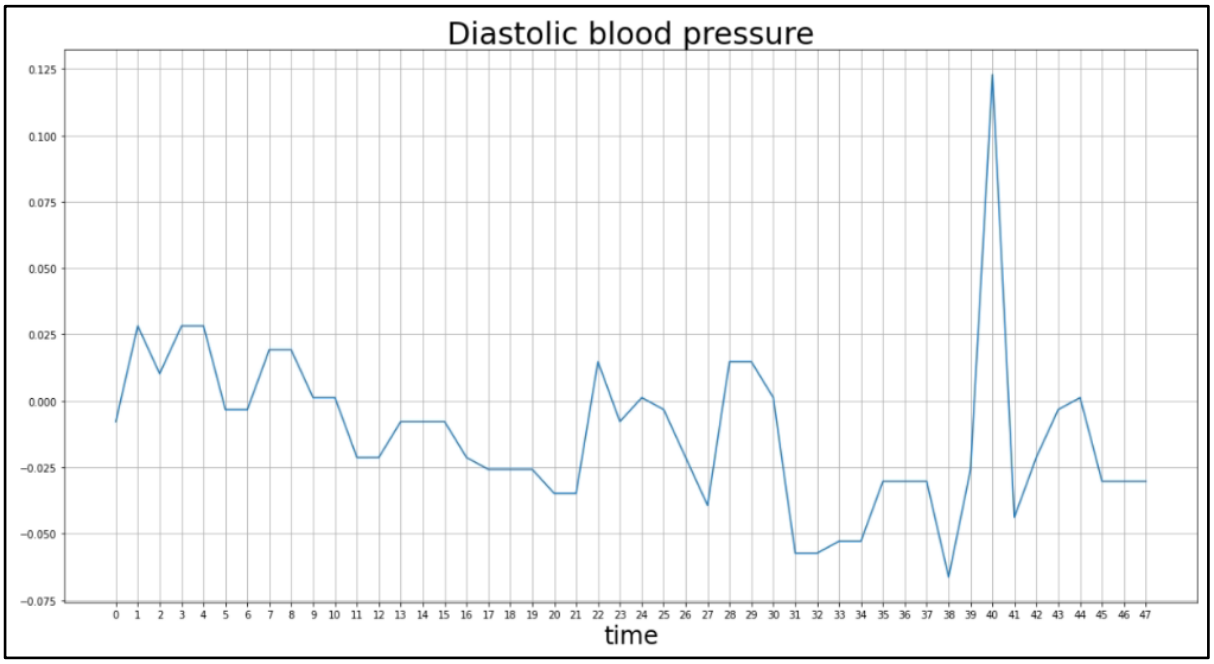

Рисунок 1 –Временной ряд: диастолическое артериальное давление

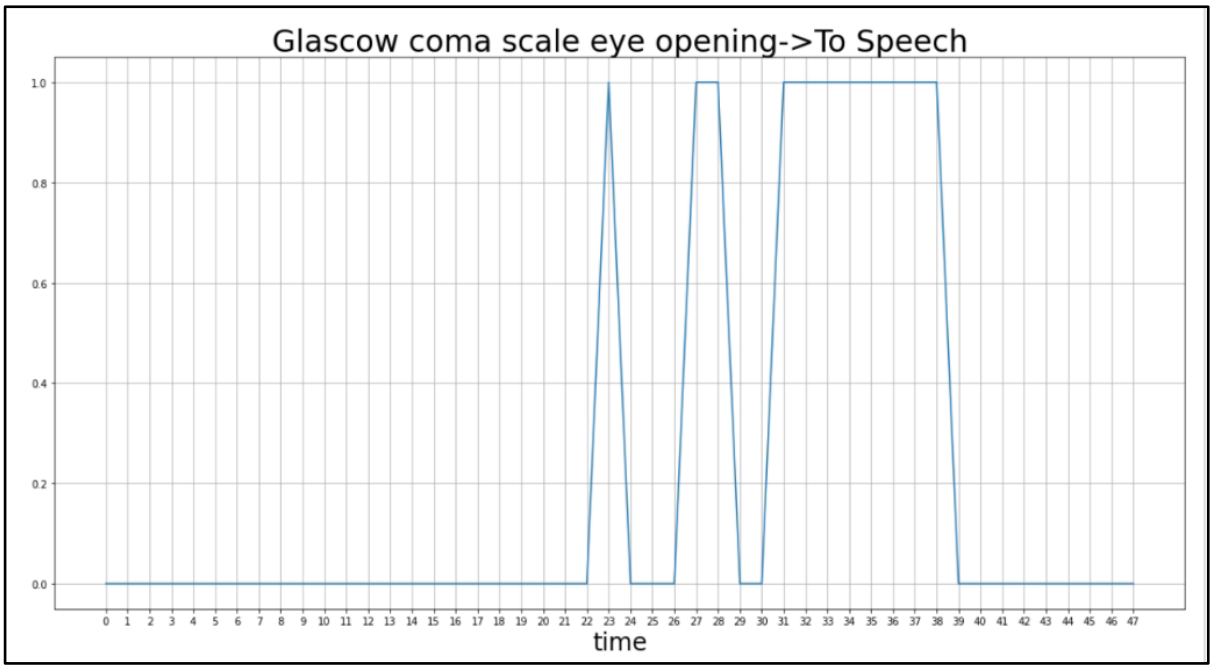

Рисунок 2 –Временной ряд: шкалы комы Глазко

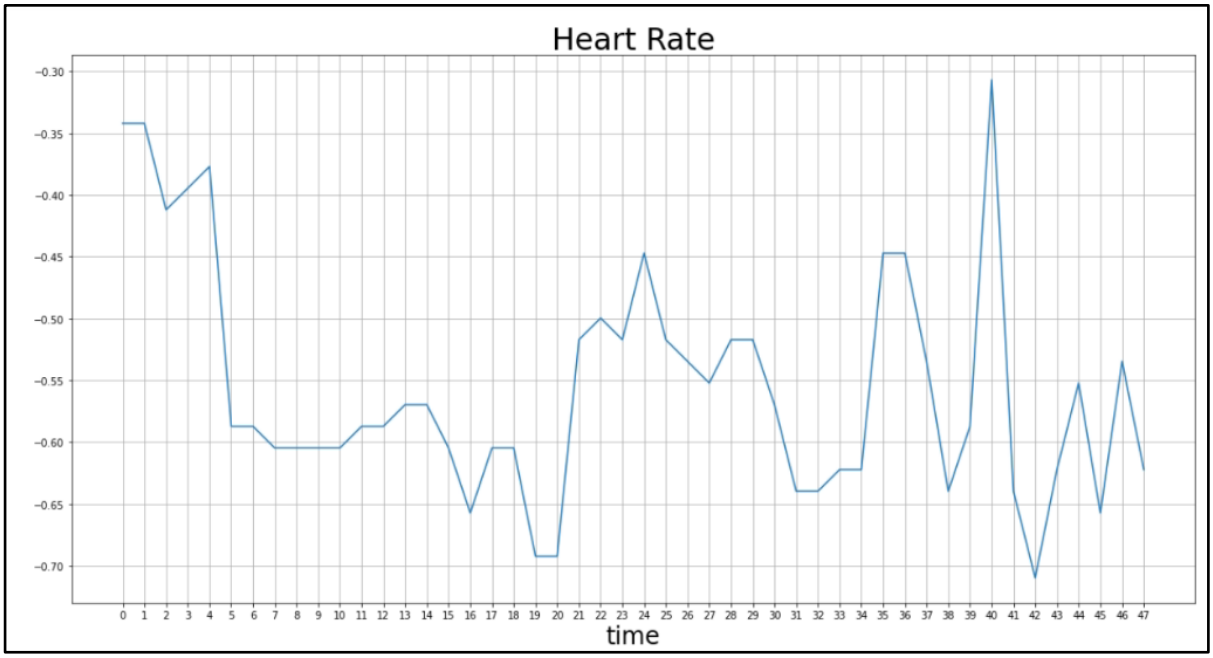

Рисунок 3 –Временной ряд: частота сердечных сокращений

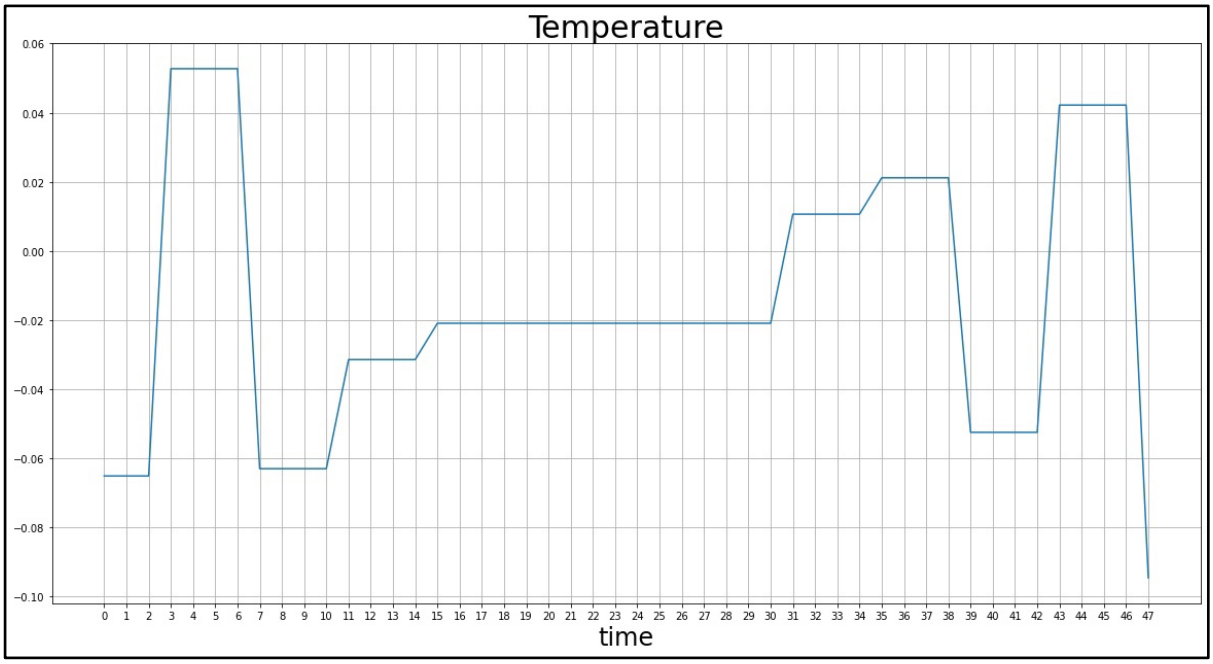

Рисунок 4 –Временной ряд: температура

Приложение В. Графики обучения блоков нейронной сети

Графики обучения всех блоков нейронной сети приведены на рисунках 1–14.

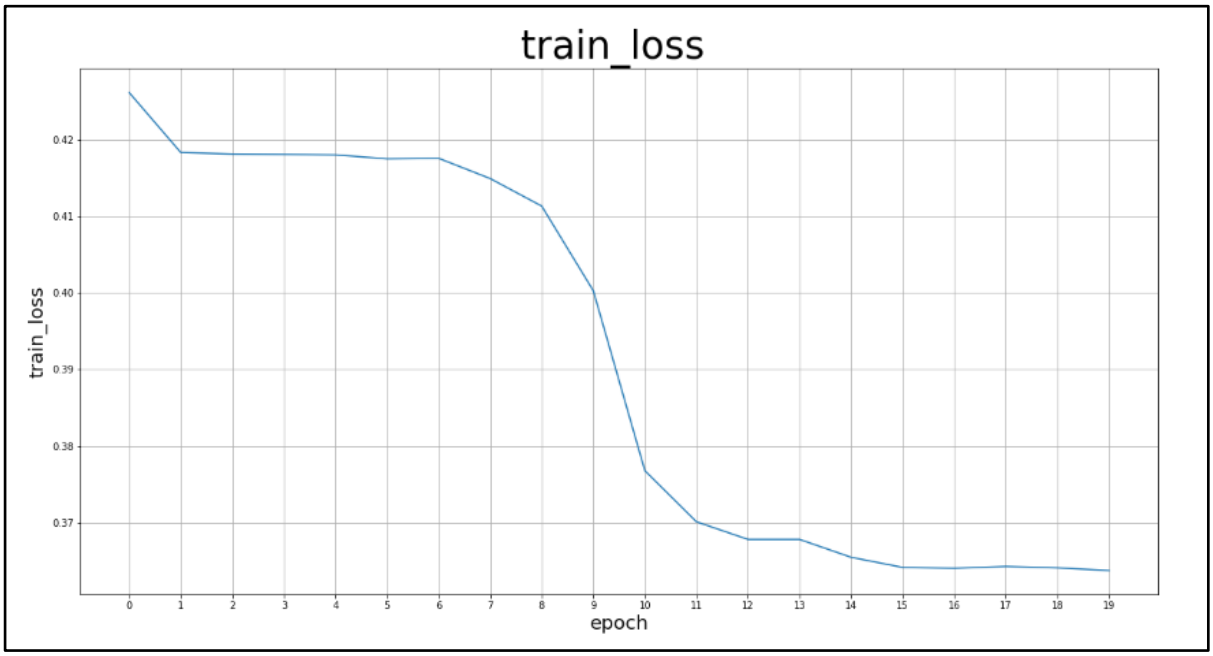

Рисунок 1 – График изменения ошибки при обучении блока 1 набора train

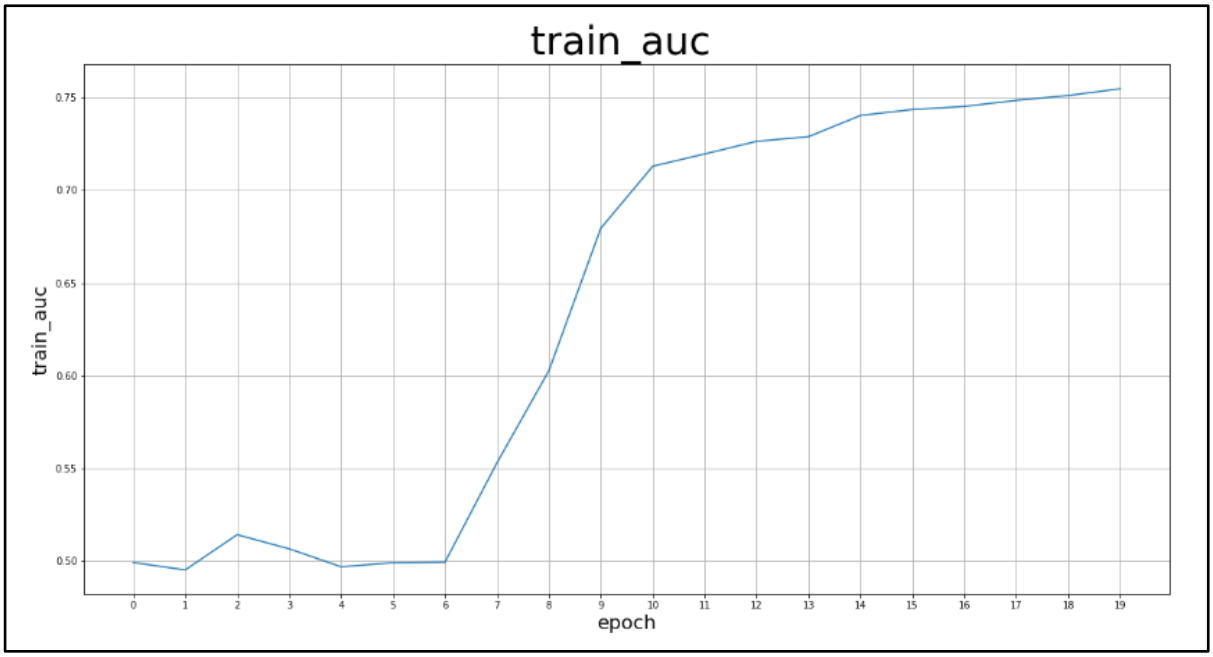

Рисунок 2 – График изменения AUC при обучении блока 1 набора train

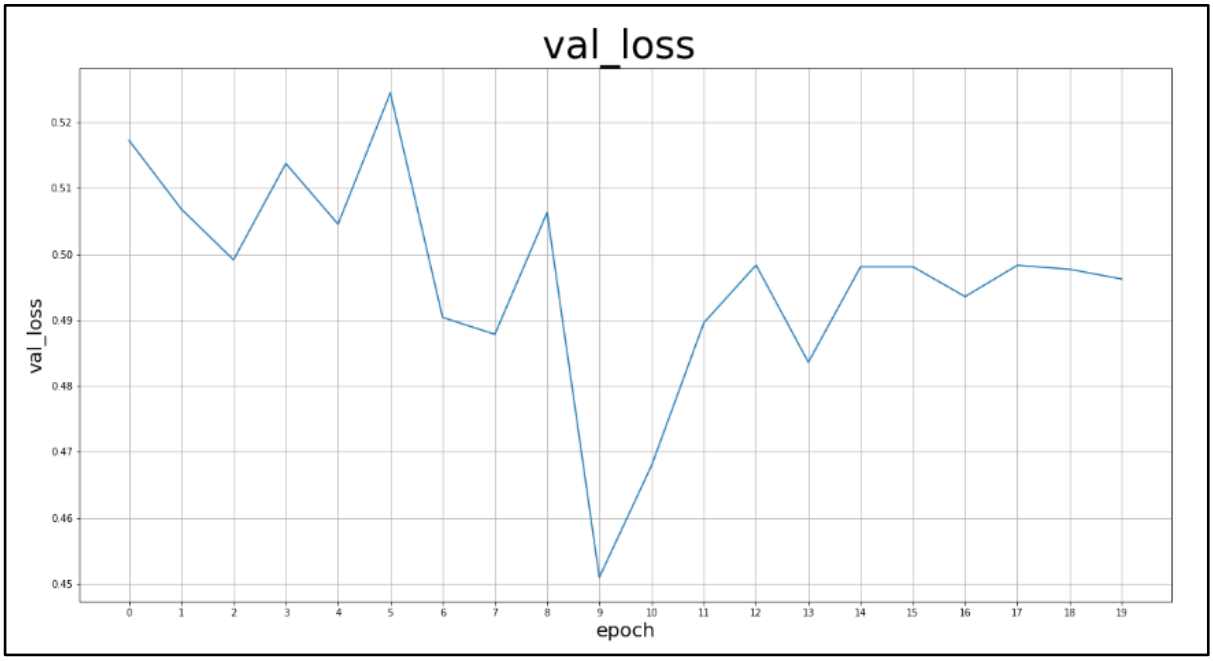

Рисунок 3 – График изменения ошибки при обучении блока 1 набора val

Рисунок 4 – График изменения ошибки при обучении блока 2 набора train

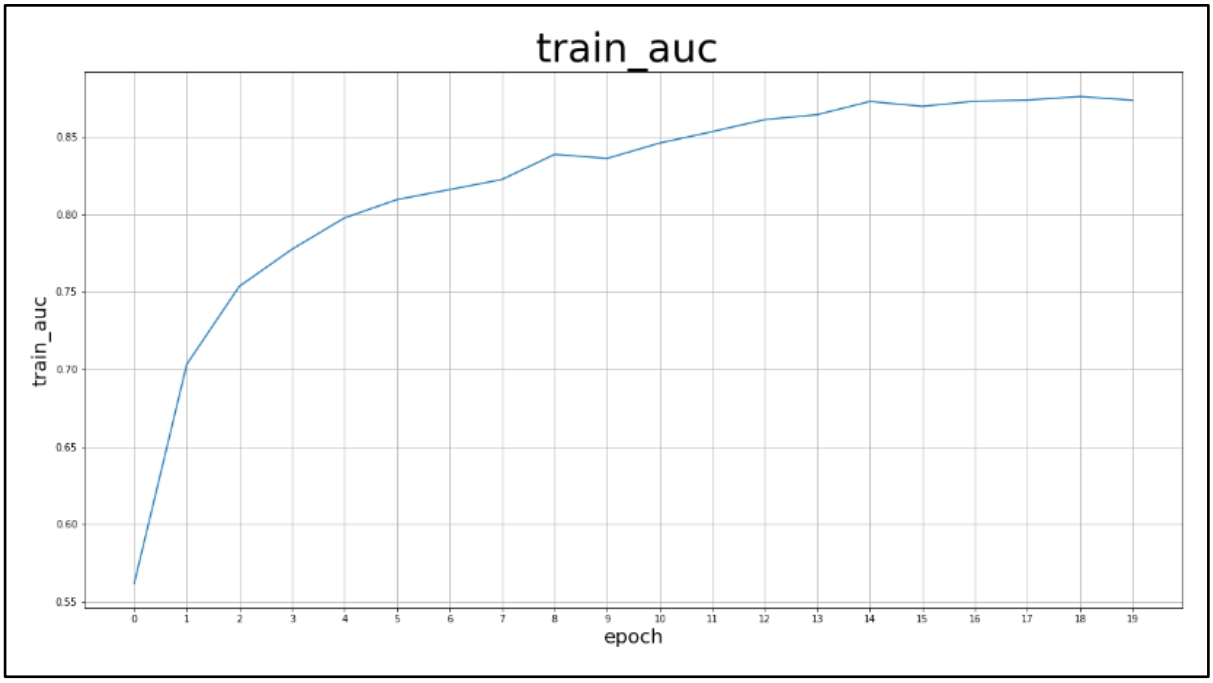

Рисунок 5 – График изменения AUC при обучении блока 2 набора train

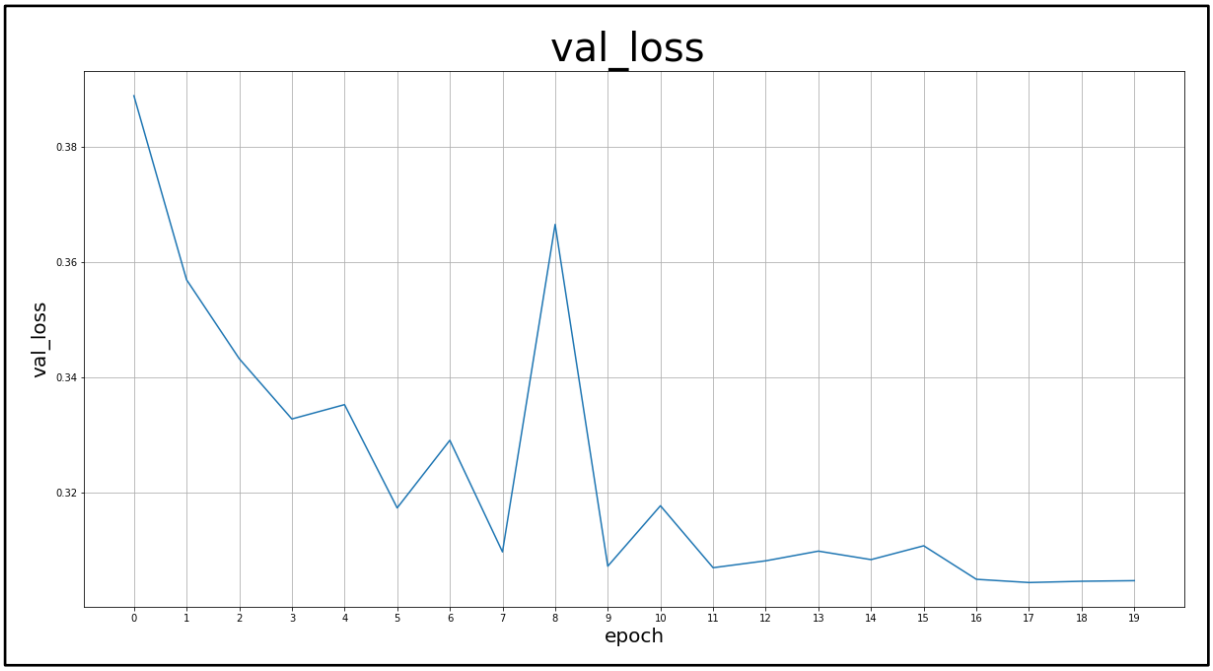

Рисунок 6 – График изменения ошибки при обучении блока 2 набора val

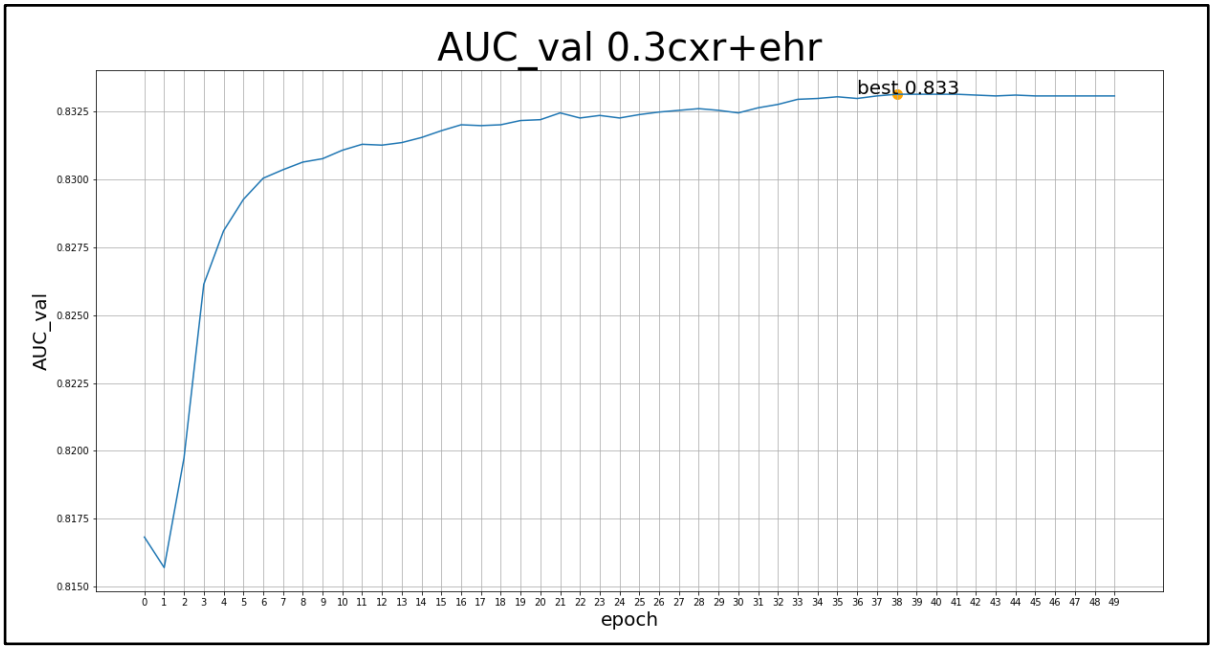

Рисунок 7 – График изменения AUC 0.3cxr+ehr набора val

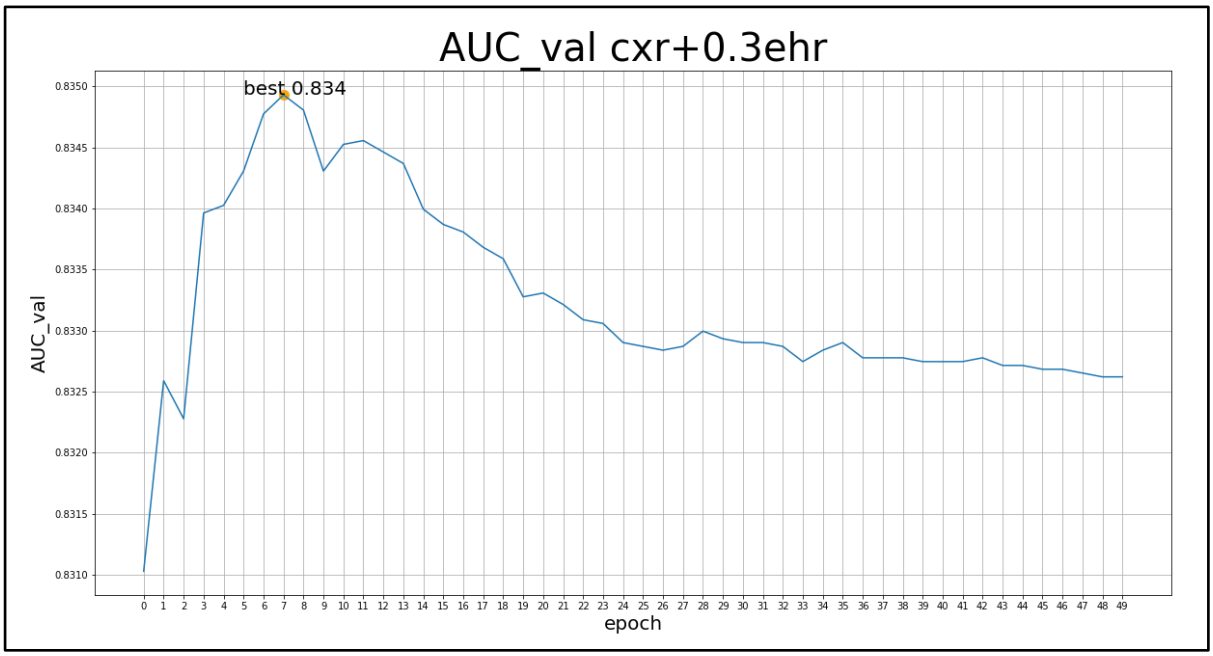

Рисунок 8 – График изменения AUC cxr+0.3ehr набора val

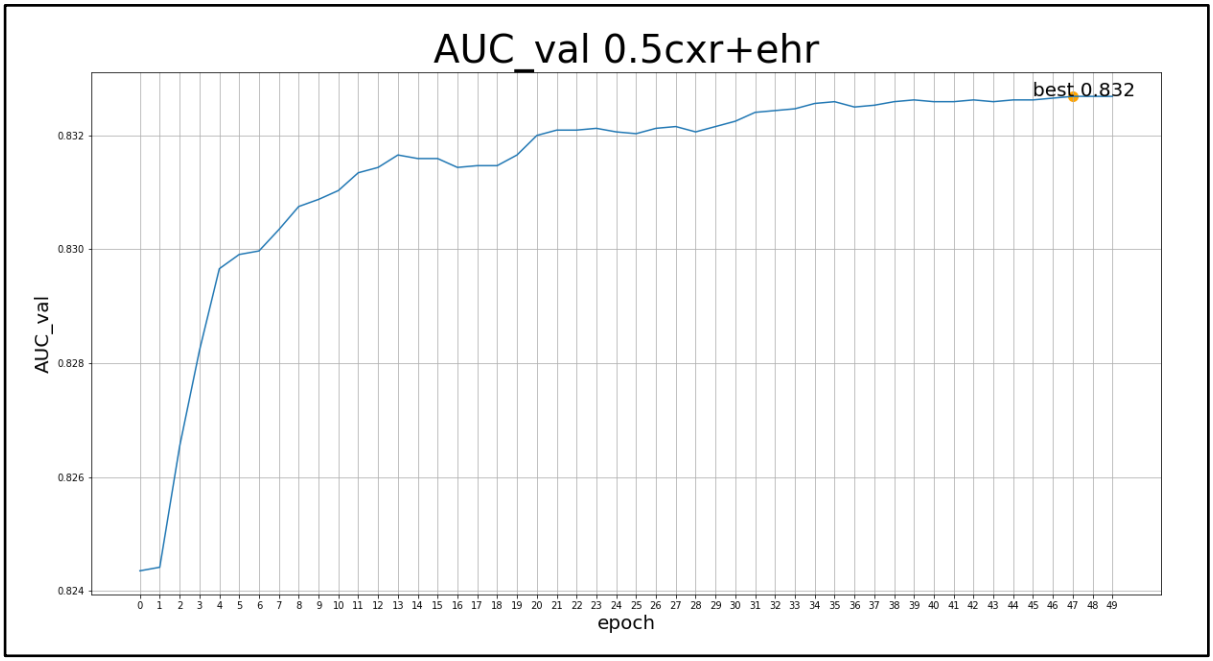

Рисунок 9 – График изменения AUC 0.5cxr+ehr набора val

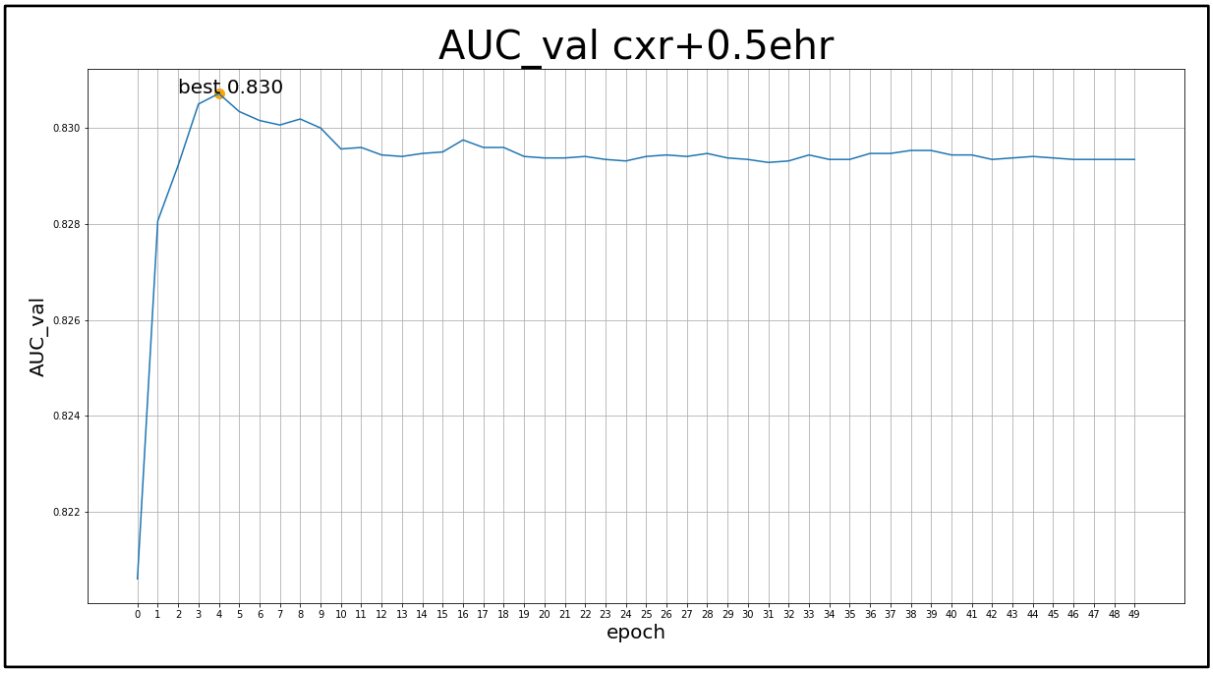

Рисунок 10 – График изменения AUC cxr+0.5ehr набора val

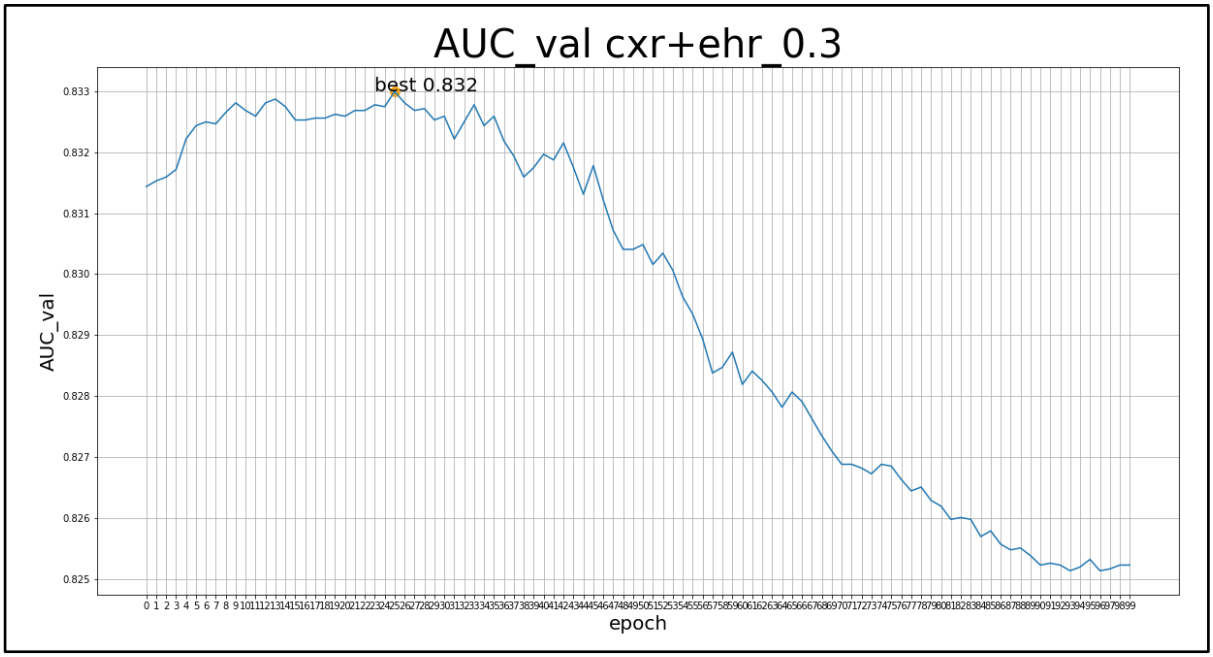

Рисунок 11 – График изменения AUC cxr+ehr_0.3 набора val

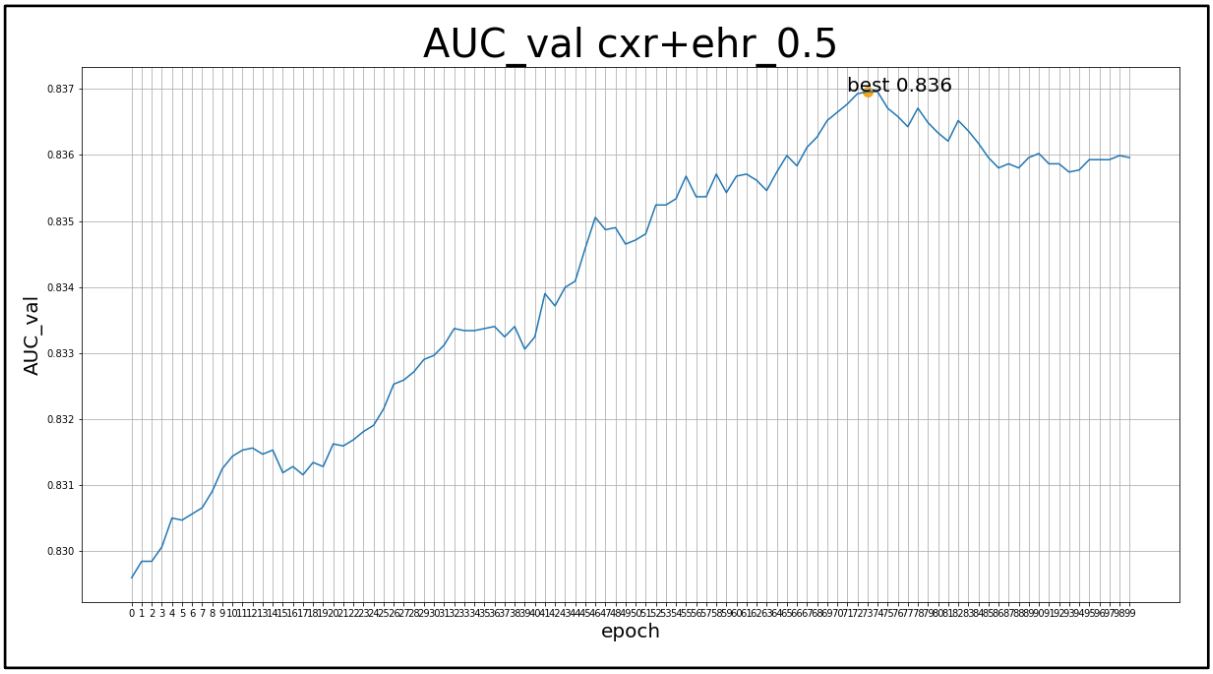

Рисунок 12 – График изменения AUC cxr+ehr_0.5 набора val