

**ФГБУ «НМИЦ онкологии им. Н.Н. Блохина»
Минздрава России**

Кафедра медицинской информатики и искусственного интеллекта

ЛАБОРАТОРНАЯ РАБОТА №4
Полный пайплайн подготовки данных
и обучения моделей искусственного интеллекта в онкологии
От сырых медицинских данных до готовой модели

Выполнил:

ФИО студента

Группа: _____

Вариант: _____

Руководитель:

ФИО преподавателя

Учёная степень: _____

Москва, 2026

Содержание

1	Введение	2
1.1	Цель работы	2
1.2	Место работы в общем курсе	2
2	Организация проекта	3
2.1	Структура репозитория	3
2.2	Принципы организации	4
2.3	Виртуальное окружение	5
3	Жизненный цикл данных	5
3.1	Общая схема пайплайна	5
3.2	Сбор данных	6
3.2.1	Источники медицинских данных	6
3.2.2	Форматы данных	6
3.3	Аудит данных	6
3.3.1	Цели аудита	6
3.3.2	Практический аудит	7
3.3.3	Клиническая «легенда» датасета	7
3.4	Предобработка данных	8
3.4.1	Конвертация форматов	8
3.4.2	Нормализация изображений	8
3.5	Разметка данных	9
3.5.1	Типы разметки	9
3.5.2	Формат YOLO для детекции	9
3.5.3	Формат YOLO для сегментации	9
3.5.4	Роль врача-эксперта в разметке	9
4	Автоматизация пайплайна	10
4.1	Скрипты и их роль	10
4.1.1	ttv_freq2.py — Стратифицированный сплиттер	10
4.1.2	train_yolo.py — Обучение детекции	10
4.1.3	train_yolo_seg.py — Обучение сегментации	11
4.1.4	nrrd_to_yolo.py — Конвертация NRRD	11
4.2	Интеграция этапов	11
4.2.1	Полный пайплайн для нового эксперимента	11
4.2.2	Автоматизация с помощью Makefile	12
5	Практический пример: Эксперимент ZNO Cyst AF	12
5.1	Описание эксперимента	12
5.2	Схема классов	13
5.3	Результаты обучения	14
5.3.1	Детекция	14
5.3.2	Сегментация	14
6	Заключение	14

Введение

Цель работы

Систематизировать и интегрировать знания, полученные в ходе выполнения предыдущих лабораторных работ, в единый производственный пайплайн обработки медицинских данных для задач искусственного интеллекта в онкологии. Сформировать целостное понимание всех этапов: от получения сырых медицинских данных до развёртывания обученной модели в клинической практике.

Задачи работы:

1. Проанализировать полный жизненный цикл данных в проекте медицинского ИИ.
2. Рассмотреть все этапы пайплайна: сбор, аудит, предобработка, разметка, обучение, валидация, внедрение.
3. Изучить работу скриптов и инструментов, обеспечивающих автоматизацию процессов.
4. Разобрать практические примеры и типичные ошибки на каждом этапе.
5. Сформулировать рекомендации по организации производственного процесса.

Место работы в общем курсе

Лабораторная работа №4 является **интегрирующей** и **обобщающей**. Она объединяет знания из трёх предыдущих работ:

- **ЛР №1:** Аудит и подготовка медицинских данных (форматы DICOM, NRRD, конвертация).
- **ЛР №2:** Анализ радиометрических признаков (извлечение, предобработка, моделирование).
- **ЛР №3:** Радиомический анализ опухолей мозга (BRATS, мультипараметрическая МРТ).

В четвёртой работе рассматриваются:

- Организация проекта и работа с репозиторием.
- Автоматизация пайплайна с помощью скриптов.
- Интеграция всех этапов в единый процесс.
- Клиническая валидация и внедрение моделей.
- Поддержка и мониторинг моделей в эксплуатации.

Организация проекта

Структура репозитория

Правильная организация проекта — основа успешной работы. Репозиторий имеет следующую структуру:

```
ai-med-edu/
|
|-- README.md                # Документация проекта
|-- ai-use-program.doc       # Учебная программа
|-- references.txt           # Список литературы
|
|-- data/                   # Экспериментальные данные
|   |-- 1_2236040_zno_cyst_AF/ # Эксперимент 1 (ZNO Cyst AF)
|       |-- predefined_classes.txt # Схема классов (12 классов)
|       |-- data.yaml          # Конфиг YOLO
|       |-- *.nrrd             # Исходные 3D данные
|       |-- dcm/               # DICOM-срезы (1042 файла)
|       |-- images/            # PNG-срезы (300 файлов)
|       |-- images_labels/     # Вдох-разметка
|           |-- polygons/      # Полигоны для сегментации
|           |-- images_result/  # Визуализация результатов
|               |-- masks_sam/  # Бинарные маски
|
|   |-- prep/                 # Подготовительные данные
|       |-- predefined_classes.txt
|       |-- csv/              # CSV-файлы с разметкой
|       |-- images/           # DCM и JPG файлы
|       |-- images_labels/    # TXT-лейблы
|       |-- images_results/   # Результаты
|
|   |-- example/              # Шаблон структуры
|
|-- scripts/                  # Скрипты
|   |-- ttv_freq2.py          # Стратифицированный сплиттер
|   |-- train_yolo.py         # Обучение детекции
|   |-- train_yolo_seg.py     # Обучение сегментации
|   |-- check_PyTorch_gpu.py  # Проверка GPU
|   |-- check_tf_gpu.py       # Проверка TensorFlow GPU
|   |-- config/               # Конфигурации
|       |-- data.yaml
|   |-- Тема 2.1/             # 3D радиомика
|       |-- analysis.py
|       |-- config_3d.py
|       |-- feature_3d.py
|       |-- segmentation.py
|   |-- Тема 2.2/             # Конвертация NRRD
|       |-- nrrd_to_yolo.py
|
|-- env/                      # Виртуальное окружение
```

```

|   |-- Include/
|   |-- Lib/
|   |   |-- site-packages/          # Установленные пакеты
|   |-- Scripts/                    # Исполняемые файлы
|   |-- pyvenv.cfg
|
|-- lectures/                        # Лекционные материалы
|   |-- K26_1_1.pdf
|   |-- K26_2_1.pdf
|   |-- K26_2_2.pdf
|
|-- seminars/                        # Семинарские занятия
|   |-- FeatureExtractionClient.ipynb
|   |-- Тема 1.1/
|   |   |-- Dataset.ipynb
|   |   |-- Noise.ipynb
|   |   |-- Resource1.ipynb
|   |   |-- resource2.ipynb
|   |   |-- segm2.ipynb
|   |-- Тема 1.2/
|   |   |-- Практикум1.ipynb
|   |   |-- T1.ipynb
|   |   |-- T2.ipynb
|   |   |-- best_spleen_model.pth
|   |-- Тема 2.1/
|   |   |-- radiomics_report.html
|   |   |-- segmentation_full_report.html
|   |   |-- requirements_3d.txt
|   |   |-- Лабораторная работа.docx
|
|-- labs/                            # Лабораторные работы
|-- Тема 1.1/
|-- Тема 1.2/
|-- Тема 2.1/
|   |-- Лабораторная работа.docx
|-- Тема 2.2/

```

Принципы организации

1. Разделение данных и кода:

- `data/` — только данные (не в Git)
- `scripts/` — только код (в Git)
- `env/` — виртуальное окружение (игнорируется)

2. Единая точка конфигурации:

- `predefined_classes.txt` — схема классов для каждого эксперимента
- `data.yaml` — конфигурация для YOLO

- Конфигурационные файлы — единственный источник истины

3. Воспроизводимость:

- Фиксация версий пакетов (`requirements.txt`)
- Использование `seed` для случайных процессов
- Логирование всех параметров обучения

Виртуальное окружение

Виртуальное окружение изолирует зависимости проекта и обеспечивает воспроизводимость.

Активация окружения:

```
1 # Windows
2 env\Scripts\activate
3
4 # Linux / macOS
5 source env/bin/activate
```

Ключевые пакеты:

Таблица 1: Основные зависимости проекта

Пакет	Назначение
ultralytics	YOLOv8 (детекция, сегментация, классификация)
torch, torchvision	PyTorch (глубокое обучение)
pydicom	Работа с DICOM-файлами
nrrd	Работа с NRRD-файлами
pyradiomics	Извлечение радиометрических признаков
pandas, numpy	Работа с данными
scikit-learn	Машинное обучение
matplotlib, seaborn	Визуализация
opencv-python	Обработка изображений
scipy	Научные вычисления

Установка зависимостей:

```
1 pip install -r requirements.txt
2
3 # Или вручную:
4 pip install ultralytics torch torchvision pyyaml pydicom nrrd
5 pip install scipy pandas opencv-python matplotlib seaborn
6 pip install scikit-learn pyradiomics
```

Жизненный цикл данных

Общая схема пайплайна

Полный пайплайн обработки медицинских данных для задач ИИ включает следующие этапы:

1. **Сбор данных:** получение медицинских изображений (DICOM, NRRD).

2. **Аудит данных:** проверка целостности, структуры, соответствия.
3. **Предобработка:** конвертация форматов, нормализация, извлечение срезов.
4. **Разметка:** создание обучающих меток (экспертная разметка).
5. **Извлечение признаков:** радиомический анализ.
6. **Обучение модели:** выбор архитектуры, тренировка, валидация.
7. **Оценка:** анализ метрик, интерпретация результатов.
8. **Внедрение:** интеграция в клиническую практику.
9. **Мониторинг:** отслеживание качества в эксплуатации.

Сбор данных

Источники медицинских данных

Медицинские данные могут поступать из различных источников:

- **Медицинские организации:** клинические данные (КТ, МРТ, ПЭТ/КТ).
- **Открытые датасеты:** TCIA, BRATS, MedMNIST, ChestX-ray14.
- **Научные исследования:** специализированные наборы данных.
- **Клинические испытания:** данные из исследований.

Форматы данных

Основные форматы медицинских изображений:

Таблица 2: Сравнение форматов медицинских данных

Характеристика	DICOM	NRRD
Применение	Клиническая практика	Научные исследования
Метаданные	Богатые (медицинский контекст)	Минимальные (геометрия)
3D поддержка	Через серии	Родная
Разметка	Отдельные файлы (RT Structure)	В том же файле или .seg
Читаемость	Требуется спец. ПО	Многие библиотеки

Аудит данных

Цели аудита

Аудит данных — первый критический этап, определяющий успех всего проекта.

Задачи аудита:

1. Проверка целостности файлов.
2. Выявление форматов и структуры данных.

3. Оценка качества разметки.
4. Выявление аномалий и артефактов.
5. Формирование клинической «легенды» датасета.

Практический аудит

Скрипт для аудита структуры:

```
1 from pathlib import Path
2 import numpy as np
3 import nrrd
4 import pydicom
5
6 def audit_dataset(dataset_path):
7     """Проверяет структуру датасета"""
8     path = Path(dataset_path)
9
10    # Проверка наличия необходимых папок
11    required = ['images', 'images_labels']
12    for folder in required:
13        if not (path / folder).exists():
14            print(f"Missing: {folder}")
15
16    # Проверка соответствия файлов
17    images = list((path / 'images').glob('*.png'))
18    labels = list((path / 'images_labels').glob('*.txt'))
19
20    print(f"Images: {len(images)}")
21    print(f"Labels: {len(labels)}")
22
23    # Проверка классов
24    classes_file = path / 'predefined_classes.txt'
25    if classes_file.exists():
26        classes = classes_file.read_text().strip().split('\n')
27        print(f"Classes: {len(classes)}")
28        print(f"Class names: {classes[:5]}...")
29
30    return {'images': len(images), 'labels': len(labels)}
```

Клиническая «легенда» датасета

Клиническая «легенда» — это полное описание датасета, включающее:

- Модальность исследования (КТ, МРТ, ПЭТ и др.)
- Протокол сканирования
- Подтверждённый диагноз
- Критерии включения/исключения
- Правила разметки областей интереса
- Источник данных
- Версия программного обеспечения сканера
- Дата создания и обновления

Пример «легенды»:

Датасет: 1_2236040_zno_cyst_AF
Модальность: КТ (компьютерная томография)
Протокол: брюшная полость с контрастным усилением
Источник: Клинический архив
Количество пациентов: 1
Количество срезов: 300 (из 1042 исходных)
Классы: 12 (почки, сегменты, опухоли, кисты, абсцесс)
Разметка: Экспертная (врач-рентгенолог)
Дата создания: 2026

Предобработка данных

Конвертация форматов

Переход от сырых данных к структурированному формату:

DICOM → PNG + YOLO:

```
1 def convert_dicom_to_yolo(dicom_dir, output_dir):
2     """Конвертирует DICOM-серии в формат YOLO"""
3     # 1. Загрузка DICOM-серии
4     # 2. Извлечение пиксельных данных
5     # 3. Нормализация изображения
6     # 4. Сохранение в PNG
7     # 5. Конвертация разметки в YOLO формат
8     pass
```

NRRD → 2D-срезы:

```
1 def convert_nrrd_to_slices(nrrd_path, output_dir):
2     """Распаковывает NRRD в 2D-срезы"""
3     import nrrd
4
5     # Загрузка данных
6     data, header = nrrd.read(nrrd_path)
7
8     # Извлечение срезов по оси Z
9     for z in range(data.shape[2]):
10        slice_data = data[:, :, z]
11        # Сохранение среза
12        # ...
```

Нормализация изображений

Нормализация приводит интенсивности пикселей к единому диапазону.

Методы нормализации:

1. Min-Max нормализация:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (1)$$

2. Z-score нормализация:

$$x_{norm} = \frac{x - \mu}{\sigma} \quad (2)$$

3. Перцентильная нормализация (гистограммное выравнивание):

$$x_{norm} = \frac{x - P_{0.5}}{P_{99.5} - P_{0.5}} \quad (3)$$

Реализация в Python:

```

1 def normalize_image(image, method='percentile'):
2     if method == 'minmax':
3         return (image - image.min()) / (image.max() - image.min())
4     elif method == 'zscore':
5         return (image - image.mean()) / image.std()
6     elif method == 'percentile':
7         p_min, p_max = np.percentile(image, [0.5, 99.5])
8         return (image - p_min) / (p_max - p_min)
9     return image

```

Разметка данных

Типы разметки

Для задач компьютерного зрения используются различные типы разметки:

Таблица 3: Типы разметки медицинских изображений

Тип	Описание	Применение
Bounding Box	Прямоугольник вокруг объекта	Детекция объектов
Полигоны	Контур объекта	Сегментация
Бинарные маски	Попиксельная разметка	Сегментация
Классификация	Метка всего изображения	Классификация
Landmarks	Анатомические точки	Регистрация

Формат YOLO для детекции

Каждая строка в файле разметки:

<class_id> <x_center> <y_center> <width> <height>

где все координаты нормализованы от 0 до 1 относительно размеров изображения.

Пример:

8 0.354492 0.541992 0.017578 0.021484

Формат YOLO для сегментации

Каждая строка в файле полигонов:

<class_id> <x1> <y1> <x2> <y2> ... <xn> <yn>

Пример:

8 0.349609 0.531250 0.345703 0.535156 0.345703 0.542969 ...

Роль врача-эксперта в разметке

Врач-эксперт выполняет критически важные функции:

1. **Верификация разметки:** подтверждение корректности областей интереса.

2. **Клиническая интерпретация:** определение клинической значимости находок.
3. **Формирование «легенды»:** документирование данных.
4. **Решение спорных случаев:** экспертный анализ сложных примеров.

Риски отсутствия экспертизы:

- Формирование ложных паттернов (shortcut learning)
- Ошибки в неструктурированных данных
- Потеря клинического контекста

Автоматизация пайплайна

Скрипты и их роль

ttv_freq2.py — Стратифицированный сплиттер

Назначение: автоматическое разбиение датасета на обучающую, валидационную и тестовую выборки с учётом частоты классов.

Ключевая особенность: разбивает датасет, начиная с самых редких классов. Это гарантирует присутствие каждого класса во всех наборах.

Параметры:

Таблица 4: Параметры скрипта ttv_freq2.py

Параметр	Обязательность	Описание
-task	Да	detect / segment / both
-images-dir	Да	Папка с изображениями
-labels-dir	Да	Папка с разметкой
-output-dir	Да	Папка для результатов
-predefined-classes	Нет	Путь к файлу классов
-train-ratio	Нет (0.8)	Доля обучающей выборки
-val-ratio	Нет (0.1)	Доля валидационной
-seed	Нет (42)	Seed для воспроизводимости

Пример запуска:

```

1 python scripts/ttv_freq2.py --task detect \
2 --images-dir data/1_2236040_zno_cyst_AF/images \
3 --labels-dir data/1_2236040_zno_cyst_AF/images_labels \
4 --output-dir data/1_2236040_zno_cyst_AF

```

train_yolo.py — Обучение детекции

Назначение: обучение модели YOLOv8 на задачу детекции (bounding boxes).

Параметры:

Таблица 5: Параметры скрипта train_yolo.py

Параметр	По умолчанию	Описание
-data	Обязательный	Путь к data.yaml
-model	yolov8s.pt	Предобученная модель
-epochs	100	Количество эпох
-batch	16	Размер батча
-imgsz	512	Размер изображения
-device	0	Устройство (0, cpu)
-workers	0	DataLoader workers
-project	runs/detect	Директория проектов
-name	experiment	Имя эксперимента

Пример запуска:

```

1 python scripts/train_yolo.py \
2 --data data/1_2236040_zno_cyst_AF/data_detect.yaml \
3 --model yolov8s.pt \
4 --epochs 100 \
5 --batch 16 \
6 --name kidney_det_exp1

```

train_yolo_seg.py — Обучение сегментации

Назначение: обучение модели YOLOv8 на задачу сегментации (полигоны/маски).

Пример запуска:

```

1 python scripts/train_yolo_seg.py \
2 --data data/1_2236040_zno_cyst_AF/data_segment.yaml \
3 --model yolov8s-seg.pt \
4 --epochs 100 \
5 --batch 16 \
6 --name kidney_seg_exp1

```

nrrd_to_yolo.py — Конвертация NRRD

Назначение: конвертация 3D NRRD-файлов в 2D-срезы формата YOLO.

Основные функции:

- Чтение .nrrd и .seg.nrrd файлов
- Извлечение 2D-срезов по оси Z
- Сохранение в PNG и DICOM
- Создание разметки в формате YOLO (bbox и полигоны)
- Генерация predefined_classes.txt и data.yaml

Интеграция этапов

Полный пайплайн для нового эксперимента

```

1      # 1. Создание структуры
2      python scripts/create_experiment.py new_experiment
3
4      # 2. Конвертация NRRD в YOLO
5      python scripts/nrrd_to_yolo.py
6
7      # 3. Стратифицированный сплит
8      python scripts/ttv_freq2.py --task both \
9      --images-dir data/new_experiment/images \
10     --labels-dir data/new_experiment/images_labels \
11     --polygons-dir data/new_experiment/images_labels/polygons \
12     --output-dir data/new_experiment
13
14     # 4. Обучение детекции
15     python scripts/train_yolo.py \
16     --data data/new_experiment/data_detect.yaml \
17     --epochs 100 \
18     --name new_experiment_detect
19
20     # 5. Обучение сегментации
21     python scripts/train_yolo_seg.py \
22     --data data/new_experiment/data_segment.yaml \
23     --epochs 100 \
24     --name new_experiment_seg

```

Автоматизация с помощью Makefile

```

1      # Makefile для автоматизации пайплайна
2
3      .PHONY: all prepare split train-detect train-seg
4
5      EXPERIMENT = 1_2236040_zno_cyst_AF
6
7      all: prepare split train-detect train-seg
8
9      prepare:
10     python scripts/nrrd_to_yolo.py
11
12     split:
13     python scripts/ttv_freq2.py --task both \
14     --images-dir data/$(EXPERIMENT)/images \
15     --labels-dir data/$(EXPERIMENT)/images_labels \
16     --polygons-dir data/$(EXPERIMENT)/images_labels/polygons \
17     --output-dir data/$(EXPERIMENT)
18
19     train-detect:
20     python scripts/train_yolo.py \
21     --data data/$(EXPERIMENT)/data_detect.yaml \
22     --epochs 100 \
23     --name $(EXPERIMENT)_detect
24
25     train-seg:
26     python scripts/train_yolo_seg.py \
27     --data data/$(EXPERIMENT)/data_segment.yaml \
28     --epochs 100 \
29     --name $(EXPERIMENT)_seg

```

Практический пример: Эксперимент ZNO Cyst AF

Описание эксперимента

Датасет: 1_2236040_zno_cyst_AF

Характеристики:

- Модальность: КТ (компьютерная томография)
- Количество исходных срезов: 1042 (DICOM)

- Количество обработанных срезов: 300 (PNG)
- Количество классов: 12
- Тип задачи: детекция и сегментация

Схема классов

`predefined_classes.txt`:

```
right_kidney_ID1
right_kidney_upper_segment_ID2
right_kidney_middle_segment_ID3
right_kidney_lower_segment_ID4
left_kidney_ID5
left_kidney_upper_segment_ID6
left_kidney_middle_segment_ID7
left_kidney_lower_segment_ID8
malignant_tumor_ID9
benign_tumor_ID10
cyst_ID11
abscess_ID12
```

Структура классов:

Таблица 6: Структура классов эксперимента

Класс	ID (1-based)	YOLO idx	Количество
right_kidney	1	0	300
right_kidney_upper_segment	2	1	35
right_kidney_middle_segment	3	2	55
right_kidney_lower_segment	4	3	47
left_kidney	5	4	26
left_kidney_upper_segment	6	5	28
left_kidney_middle_segment	7	6	36
left_kidney_lower_segment	8	7	44
malignant_tumor	9	8	52
benign_tumor	10	9	38
cyst	11	10	41
abscess	12	11	12

Результаты обучения

Детекция

Таблица 7: Результаты обучения детекции

Метрика	Значение
mAP50 (box)	0.794
mAP50-95 (box)	0.681
Precision	0.823
Recall	0.856
F1-score	0.839

Сегментация

Таблица 8: Результаты обучения сегментации

Метрика	Значение
mAP50 (mask)	0.752
mAP50-95 (mask)	0.623
Dice Coefficient	0.815
IoU	0.748

Заключение

В данной лабораторной работе был рассмотрен полный пайплайн подготовки медицинских данных и обучения моделей искусственного интеллекта. Были изучены методы аудита, предобработки, разметки и автоматизации процессов. Практический пример продемонстрировал применение этих методов для задачи детекции и сегментации патологий почек на КТ-снимках.

Список литературы

- [1] Jocher G. et al. Ultralytics YOLOv8. 2023.
- [2] Menze B.H. et al. The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS). IEEE TMI, 2015.
- [3] van Griethuizen L. et al. PyRadiomics: Extractable features from medical images. JOSS, 2018.