Министерство науки и высшего образования РФ
Томский государственный университет
систем управления и радиоэлектроники

В.А. Воронков, В.М. Ежова, С.А. Литовкин, Е.Ю. Костюченко

ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ В МЕДИЦИНСКОЙ РЕАБИЛИТАЦИИ

Методические указания к лабораторным и практическим
для студентов направления подготовки
09.04.04 Программная инженерия

УДК 004.8
ББК 32.813.5
Л 64

Литовкин, С. А. ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ В МЕДИЦИНСКОЙ РЕАБИЛИТАЦИИ: Учебно-методические указания / В.А. Воронков, В.М. Ежова, С.А. Литовкин, Е.Ю. Костюченко. — Томск: ТУСУР, 2024. — 65 с.

Настоящие учебно-методические указания содержат описание лабораторных и практических работ по дисциплине «Искусственный интеллект в медицинской реабилитации» для студентов направления подготовки 09.04.04 Программная инженерия.

Одобрено на заседании кафедры КИБЭВС протокол №7 от 30.08.2024 года

УДК 004.8
ББК 32.813.5

© Воронков В.А., Ежова В.М., Литовкин С.А., Е.Ю. Костюченко. 2024
© Томск. гос. ун-т систем упр. и радиоэлектроники, 2024

Содержание

Введение

Представленные ниже методические указания направлены на изучение принципов работы искусственного интеллекта на примере данных медицинской реабилитации, а также анализа реализации и применения искусственного интеллекта, современных технологий в реабилитации пациентов с различными заболеваниями.

Задачи изучения дисциплины:

- изучить основные принципы работы искусственного интеллекта;
- изучить заболевания подвластные медицинской реабилитации, а также способы проведения медицинской реабилитации;
- рассмотреть принципы применения искусственного интеллекта в медицинской реабилитации.

ЛАБОРАТОРНАЯ РАБОТА №1

Разработка модели ИИ для реабилитации пациентов с заболеваниями нервной системы

Целью работы является исследование применения методов машинного обучения при анализе реабилитации пациентов после инсульта.

1 ТЕОРЕТИЧЕКИЕ СВЕДЕНИЯ

Заболевания нервной системы – это нарушения, которые возникают как результат поражения спинного и головного мозга, периферических нервных стволов. Основными проявлениями патологий этой группы считаются двигательные расстройства, тремор, параличи и проблемы с психикой.

Заболевания нервной системы по своему происхождению подразделяют на:

- сосудистые;
- инфекционные;
- хронические;
- прогрессирующие;
- наследственные;
- травматические.

Не все из перечисленных видов подвластны реабилитации (восстановлению).

В первой лабораторной работе предлагается использовать набор данных, собранный для отслеживания реабилитации пациентов после инсульта.

В представленном наборе можем увидеть папки, которые разделяют экземпляры изображений на две группы «Здоровые» и «Пациенты». Изображения были получены путем обведения на графическом планшете

следующих элементов рисунка: прямая и волнистая линия, треугольник, спираль и фраза «Ей дан чай» (рисунок 1.1).

Несмотря на большой риск повторного проявления сосудистой катастрофы и невысокий процент полного восстановления всех жизненно важных функций человека после инсульта, реабилитация возможна.

Предлагаемый способ оценки динамики реабилитации заключается в вычислении метрики «мера уверенности в принадлежности к классу», суть которой состоит в том, что значение, полученное нейронной сетью, показывает, насколько анализируемый экземпляр почерка близок к одному из двух классов – здоровым или пациентам.

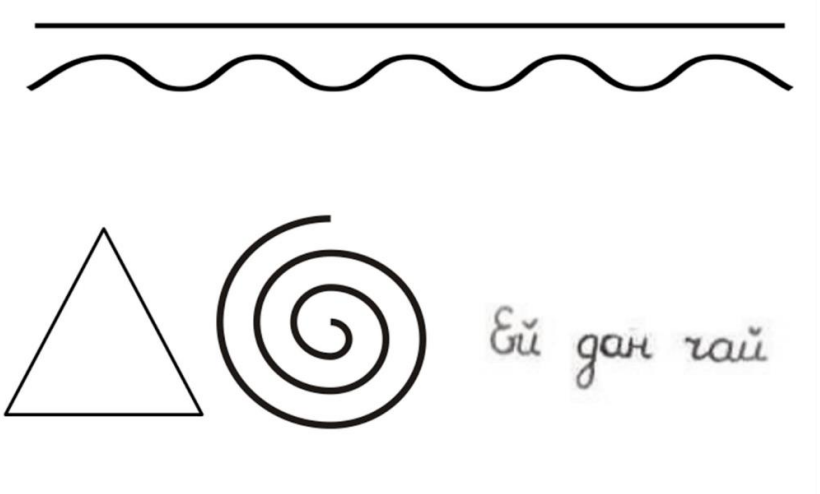

Рисунок 1.1 – Эталонный рисунок

Экземпляры изображений создавались здоровыми людьми (4 человека по 20 изображений) и пациентами, проходящими реабилитацию после инсульта (3 человека от 3 до 7 изображений). В общем набор данных содержит в себе 94 элемента.

Так как представленный набор данных мал, для работы с ним предлагается использовать сиамскую нейронную сеть.

Сиамская нейронная сеть – класс архитектур нейронных сетей, который состоит из двух или более идентичных подсетей: нейронных сетей с одинаковой архитектурой, конфигурацией и весами. Целью наличия идентичных подсетей является обучение модели на основе функции подобия,

которая измеряет, насколько отличаются векторы признаков одного изображения от другого. Благодаря такой архитектуре модель может быть обучена без большого количества данных. Её обобщённая архитектура представлена на рисунке 1.2.

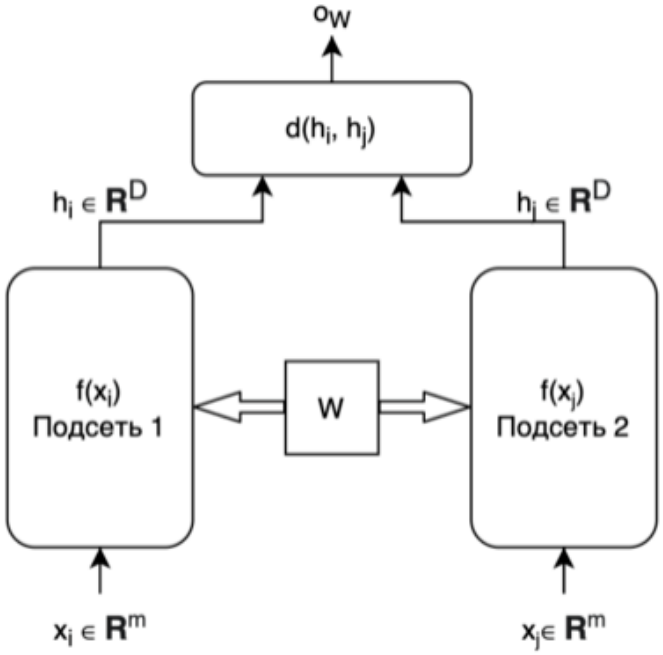

Рисунок 1.2 – Обобщённая архитектура сиамской нейронной сети

При формировании набора данных с изображениями, проходила фиксация динамических параметров подписи.

Так, при обведении изображения были сняты следующие величины:

– скорость (общая, горизонтальная, вертикальная);

– ускорение (общее, горизонтальное, вертикальное);

– рывок (общий, горизонтальный, вертикальный);

– корреляция Спирмена (давления и скоростей, давления и ускорения);

– количество экстремумов (давление, скорость, ускорение).

В частности, для скорости, ускорения и рывка выделяют характеристики: первый перцентиль, девяносто девятый перцентиль, среднеквадратическое отклонение и среднее значение. Так, извлекли 45 признаков.

Из рассмотренных выше величин был собран второй набор данных.

В котором можно увидеть 46 столбцов (45 – параметры, 1 – выходной) и 96 строк (95 +1 по количеству экземпляров изображений).

Требования к выполнению лабораторной работы

Для выполнения данной лабораторной работы необходимо:

1. ознакомиться с теоретическими сведениями и рассмотреть представленные в разделе «Теоретические сведения» наборы данных. Ссылка на набор данных «Изображения» инсульт: https://drive.google.com/drive/folders/1IBd75zRJljxZ2yGFSXXUIAHNYO6QqgJU?usp=sharing. Ссылка на набор данных «Динамические параметры подписи»: https://drive.google.com/drive/folders/1IBd75zRJljxZ2yGFSXXUIAHNYO6QqgJU?usp=sharing;

2. Выполнить ход работы.

3. первый набор данных с изображениями передать в сиамскую нейронную сеть, продемонстрировать измененный код и зафиксировать результаты обучения сети в виде таблицы и графика, отображающего успешный процесс классификации. Ссылка на блокнот: https://colab.research.google.com/drive/1kVkXgA-u4or2LsUB6yeuatkgc6xcL_D3?usp=sharing;

4. для второго набора данных с параметрами подписи по варианту рассмотреть работу классификаторов, зафиксировать результаты классификации и проанализировать результаты их работы. Ссылка на блокнот: https://colab.research.google.com/drive/1zChITt6NpJLPMuwFqUW6kuXnnVFgp25W?usp=sharing;

5. полученные результаты перенести в отчет и защитить у преподавателя.

Варианты по классификаторам:

1. SVC, LDA, DT;
2. KNN, NB, DT;
3. SVC, LDA, KNN;
4. NB, LDA, LR;
5. KNN, LR, SVC;
6. LR, DT, LDA;
7. NB, SVC, KNN;
8. LDA, KNN, SVC.

2 ХОД РАБОТЫ

2.1 Работа с сиамской нейронной сетью

Для работы с набором данных «Изображения» предоставлена архитектура сиамской нейронной сети.

Ниже представлены используемые для работы стандартные библиотеки.

```
import re
import numpy as np
from matplotlib.pyplot import imread
from PIL import Image
from tensorflow import keras
import math

from sklearn.model_selection import train_test_split
from keras import backend as K
from keras.layers import Activation
from keras.layers import Input, Lambda, Dense, Dropout, Convolution2D, MaxPooling2D, Flatten
from keras.models import Sequential, Model
from keras.optimizers import RMSprop
from tensorflow.keras import regularizers
from sklearn.metrics import confusion_matrix, classification_report
```

После ознакомления с импортированными библиотеками, необходимо передать в сеть набор данных.

Далее требуется преобразовать входные данные, учитывая, что сиамским сетям нужно подавать пары данных с двоичной маркировкой. Необходимо определить функцию get_data, которая будет генерировать пары.

Фрагмент кода с определением функции представлен ниже:

```
def get_data(size, total_sample_size):
    count = 0
#Здоровые
    x_geuine_pair = np.zeros([total_sample_size, 2, 240, 320, 3]) # 2 is for pairs
    y_genuine = np.zeros([total_sample_size, 1])

    for i in range(1, 225, 1):
      ind1 = 0
      ind2 = 0

      #читать изображения из одного каталога (подлинная пара)
```

```
            while ind1 == ind2:
                ind1 = np.random.randint(79)
                ind2 = np.random.randint(79)

            #читать 2 изображения
            img1 = cv2.imread('/content/drive/MyDrive/lab1/bd_insult/3доровые/' + str(ind1 + 1)
+ '.png')
            img2 = cv2.imread('/content/drive/MyDrive/lab1/bd_insult/3доровые/' + str(ind2 + 1)
+ '.png')
            #сохранить изображения в инициализированном массиве numpy
            x_geuine_pair[count, 0, :, :, :] = img1
            x_geuine_pair[count, 1, :, :, :] = img2

              #поскольку мы рисуем изображения из одного и того же каталога, мы
присваиваем метку 1 (подлинная пара)
            y_genuine[count] = 0
            count += 1
```

Пары изображений создаются в результате рассмотрения изображений следующим образом:

– из 15 изображений пациентов было получено 225 пар (каждое с каждым);

– из 80 изображений из класса здоровые так же было получено 225 пар (случайное со случайным до получения нужного количества пар);

– из 15 изображений пациентов и 80 здоровых было создано 450 пар (случайное здорового со случайным пациента).

Итого было получено 900 пар изображений.

Далее необходимо разделить получившийся набор данных в отношении: 75 % пар пойдет на обучение, а 25 % — на тестирование.

Для создания сиамской сети нужно определить базовую — свёрточную нейронную сеть для извлечения свойств, создать два свёрточных слоя с помощью ReLU-активаций и слоя с определением максимального значения (max pooling), после сглаживающим (flatten) слоем.

```
        def build_base_network(input_shape):
            seq = Sequential()
            nb_filter = [32, 32]
            kernel_size = 3
```

```
        #Блок свёртки 1
        seq.add(Convolution2D(nb_filter[0], kernel_size, kernel_size,
input_shape=input_shape))
        seq.add(Activation('relu'))
        seq.add(MaxPooling2D(pool_size=(2, 2)))
        #Блок свёртки 2
        seq.add(Convolution2D(nb_filter[1], kernel_size, kernel_size))
        seq.add(Activation('relu'))
        seq.add(MaxPooling2D(pool_size=(2, 2)))
        #Сглаживание
        seq.add(Flatten())
        seq.add(Dense(128, activation='relu'))
        seq.add(Dropout(0.1))
        return seq
```

Часть кода, представленного выше, необходимо дополнить для успешного обучения сиамской сети.

Следом требуется подать пару изображений базовой сети, которая вернёт векторные представления, то есть векторы свойств:

```
input_dim = (240, 320, 3)
img_a = Input(shape=input_dim)
img_b = Input(shape=input_dim)
base_network = build_base_network(input_dim)
feat_vecs_a = base_network(img_a)
feat_vecs_b = base_network(img_b)
```

feat_vecs_a и *feat_vecs_b* — это векторы свойств пары изображений.

Далее следует передать их функции энергии для вычисления дистанции между ними. Для этого необходимо использовать Евклидово расстояние:

```
def euclidean_distance(vects):
    x, y = vects
    return K.sqrt(K.sum(K.square(x - y), axis=1, keepdims=True))
def eucl_dist_output_shape(shapes):
```

```
    shape1, shape2 = shapes
    return (shape1[0], 1)
```

Затем необходимо начать обучение модели и зафиксировать результаты, в виде графиков обучения и сводной таблицы по разным количествам эпох.

Для защиты данной лабораторной работы результат обучения сиамской нейронной сети должен достичь выше 70% доли верных ответов.

2.2 Разбор классификаторов

Для работы с параметрическими данными, используются различные классификаторы. Ниже приведен пример реализации метода опорных векторов (SVC).

```
Реализация:
results=[]
 for i in range(100):
    rs=random.randint(1,20)
    result=[]
    scaler = StandardScaler()
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.fit_transform(X_test)
    svc = SVC(C=2.9)
    svc = svc.fit(X_train, y_train)
    y_pred = svc.predict(X_test)
    result.append( metrics.accuracy_score(y_test, y_pred))
    print("SVC model accuracy:", metrics.accuracy_score(y_test, y_pred))
    print(confusion_matrix(y_test, y_pred))
    print(classification_report(y_test, y_pred))
results
```

По примеру выше, а также на основе пройденных ранее дисциплин, по варианту необходимо применить классификаторы на наборе данных с динамическим параметрами подписи. В отчет необходимо предоставить

сводную таблицу по результатам применения, провести анализ и сделать выводы.

Для защиты лабораторной работы результаты классификации не должны быть ниже 70% точности указаний верных ответов.

Приведенное решение работает с «.csv» файлом, перед началом применения классификаторов по варианту в приложенном файле необходимо убрать первую строку.

Предполагаемый пакет инструментов: Scikit-learn (sklearn) Python, метод k-ближайших соседей (KNN), классификатор дерева решений (DT), наивный Байесовский классификатор (NB), линейный дискриминантный анализ (LDA), метод опорных векторов (SVC/SVM), логистическая регрессия (LR).

2.3 Задание на лабораторную работу

1. Первый набор данных с изображениями передать в сиамскую нейронную сеть, продемонстрировать измененный код и зафиксировать результаты обучения сети в виде таблицы и графика, отображающего успешный процесс классификации.

3. Для второго набора данных с параметрами подписи по варианту рассмотреть работу классификаторов, зафиксировать результаты классификации и проанализировать результаты их работы.

4. Полученные результаты перенести в отчет и защитить у преподавателя.

3 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое реабилитация?
2. Какие заболевания нервной системы подвластны реабилитации?
3. Как устроена сиамская нейронная сеть?

4. Как оценивается динамика реабилитации?

5. Какой из использованных классификаторов наиболее эффективен при работе с представленным набором данных?

ЛАБОРАТОРНАЯ РАБОТА №2

Разработка модели ИИ для отслеживания реабилитации пациентов с онкологическими заболеваниями

Целью данной лабораторной работы является исследование применения методов машинного обучения и искусственного интеллекта при анализе реабилитации пациентов с онкологическими заболеваниями, путем применения разновидности сверточной нейронной сети «DenseNet» на наборе данных по варианту.

1 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Сверточная нейронная сеть (Convolutional Neural Networks, CNN) – класс алгоритмов машинного обучения, специализирующийся на обработке изображений и видео. Особенностью CNN является процесс обучения, при котором вначале происходит прямое распространение от первого слоя к последнему, затем вычисляется ошибка в выходном слое и осуществляется обратное распространение. При этом в каждом слое вычисляются градиенты обучаемых параметров, а в конце обратного распространения происходит обновление весов с помощью градиентного спуска.

Сверточные нейронные сети (CNN) имеют уникальную архитектуру, специально разработанную для эффективной работы с изображениями и видеоданными. В состав сверточной нейронной сети входит несколько слоев. От количества слоев зависит мощность архитектуры и эффективность обучения. Ниже представлены основные составляющие сверточной нейронной сети:

- сверточный слой;
- пулинг;
- нормализация по батчу;
- полносвязный слой.

Сверточные слои выделяют локальные признаки, которые могут быть обнаружены в любом месте изображения.

При помощи пулинга (subsampling) уменьшают пространственные размеры карт признаков, делая сеть менее чувствительной к точному расположению признаков на изображении.

После сверточных и пулинг слоев, структура данных «выпрямляется» (flatten) и передается в полносвязные слои (Dense слои). Эти слои дальше обрабатывают признаки для выполнения классификации.

Эти особенности делают CNN особенно практичными в задачах, связанных с распознаванием образов, обработкой изображений, позволяя им выявлять и классифицировать объекты в сложных, изменчивых условиях.

Принцип работы сверточной нейронной сети представлен ниже на рисунке 1.1.

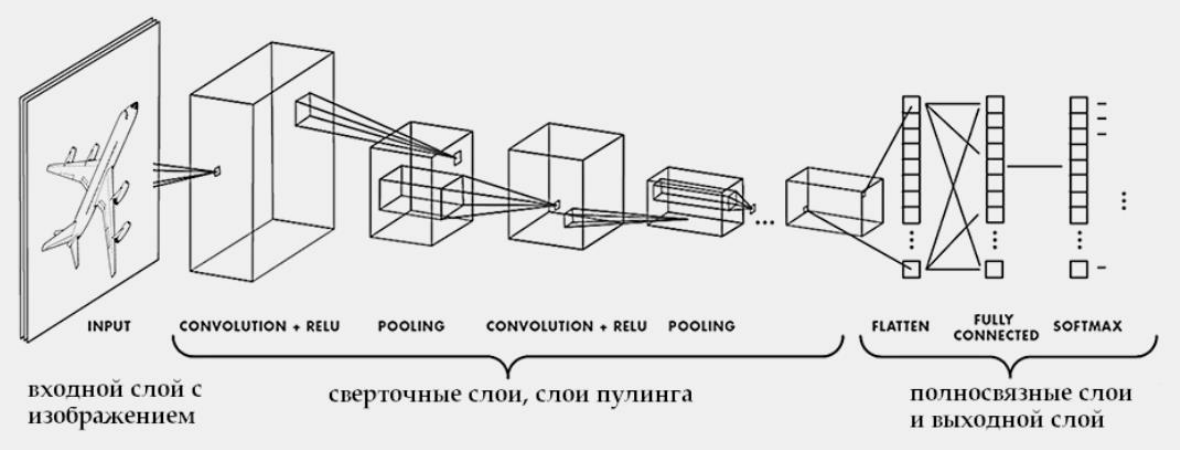

Рисунок 1.1 – Принцип работы CNN

Требования к выполнению лабораторной работы

1. Ознакомиться с методическими указаниями.
2. Ознакомиться с представленными наборами данных различных видов заболевания рака: https://drive.google.com/drive/folders/1z1Esc7v6zJT-ETSDrq3-Q7UKvw4NZUEs?usp=drive_link.

3. Рассмотреть код, представленный в блокноте Google Colab: https://colab.research.google.com/drive/1A-iTD4oGnIiu29wxVPiC2ssQsEO-8sIJ?usp=sharing.

4. Выполнить ход работы.

5. Применить сверточную нейронную сеть «DenseNet» с разным количеством слоев на наборе данных по варианту.

6. Написать отчёт, содержащий информацию о использованном наборе данных, описание подбора параметров для достижения высокого уровня показателя доли верных ответов (>80%), графики результатов обучения, сделать вывод о влиянии количества слоев свертки на результаты работы сети.

Варианты для выполнения индивидуального задания представлены по классам раковых заболеваний:

1. Lymphoma.
2. Lung and Colon Cancer.
3. Cervical Cancer.
4. Brain Cancer.
5. Acute Lymphoblastic Leukemia.

2 ХОД РАБОТЫ

2.1 Разбор набора данных

Предложенный набор данных «Multi Cancer Dataset» содержит изображения различных типов рака, собранные для исследовательских и аналитических целей. Он включает 5 основных классов рака и 20 подклассов (в каждом классе разное количество подклассов и экземпляров изображений).

Информация о классах из первоначального источника будет представлена ниже.

Acute Lymphoblastic Leukemia (ALL). Содержит 20 000 изображений, представляющих доброкачественные лейкемии и различные стадии лейкемии (таблица 2.1).

Таблица 2.1 – Класс ALL

Путь	Подкласс	Описание
/all_benign	Benign	Нераковые, здоровые клетки
/all_early	Early	Ранние стадии лейкимии
/all_pre	Pre	Аномальные клетки на предстадии
/all_pro	Pro	Клетки прогрессирующего лейкоза

Brain Cancer. Содержит 15 000 изображений, охватывающих 3 основных типа рака мозга (таблица 2.2).

Таблица 2.2 – Класс Brain Cancer

Путь	Подкласс	Описание
/brain_glioma	Glioma	Наиболее распространённая опухоль головного мозга
/brain_menin	Meningioma	Опухоли, поражающие мозговые оболочки
/brain_tumor	Pituitary Tumor	Опухоли, поражающие гипофиз

Cervical Cancer. Содержит 25 000 изображений, охватывающих различные типы клеток шейки матки (таблица 2.3).

Таблица 2.3 – Класс Cervical Cancer

Путь	Подкласс	Описание
/cervix_dyk	Dyskeratotic	Аномальный рост клеток
/cervix_koc	Koilocytotic	Клетки, демонстрирующие изменения, вызванные вирусными инфекциями
/cervix_mep	Metaplastic	Предраковые
/cervix_pab	Parabasal	Незрелый плоскоклеточный рак
/cervix_sfi	Superficial-Intermediate	Более зрелые плоские клетки

Lung and Colon Cancer. Содержит 25 000 изображений, охватывающих различные типы тканей легких и толстой кишки (таблица 2.4).

Таблица 2.4 – Lung and Colon Cancer

Путь	Подкласс	Описание
/colon_aca	Colon Adenocarcinoma	Раковые клетки толстой кишки
/colon_bnt	Colon Benign Tissue	Здоровые ткани толстой кишки
/lung_aca	Lung Adenocarcinoma	Раковые клетки легкого
/lung_bnt	Lung Benign Tissue	Здоровые ткани легких
/lung_scc	Lung Squamous Cell Carcinoma	Агрессивный тип рака легких

Lymphoma. Содержит 15 000 изображений по трем подклассам лимфом (таблица 2.5).

Таблица 2.5 – Класс Lymphoma

Путь	Подкласс	Описание
/lymph_cll	Chronic Lymphocytic Leukemia	Медленно прогрессирующий рак крови
/lymph_fl	Follicular Lymphoma	Медленнорастущая фолликулярная лимфома
/lymph_mcl	Mantle Cell Lymphoma	Агрессивная форма лимфомы

2.2 Разбор архитектуры DenseNet

Dense Convolutional Network (DenseNet), представляет собой архитектуру глубокого обучения для сверточных нейронных сетей. В отличии от традиционных архитектур CNN, где каждый слой соединен только с последующими слоями, DenseNet устанавливает прямые соединения между всеми слоями в блоке (рисунок 2.1). Эта плотная связь позволяет каждому слою получать карты признаков из всех предыдущих слоев в качестве входных данных, способствуя обширному потоку информации по всей сети.

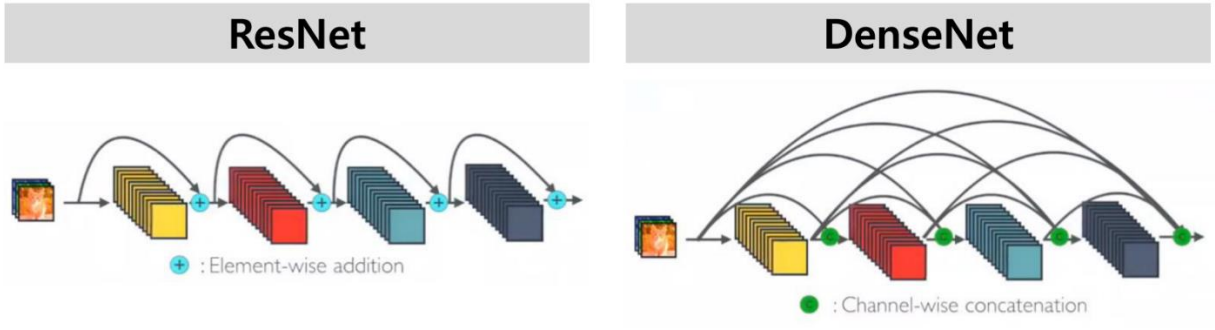

Рисунок 2.1 – Сравнение связей архитектур DenseNet и ResNet

DenseNet поставляется в нескольких вариантах, отличающихся прежде всего глубиной и количеством слоев:

DenseNet-121: содержит 121 слой, известный своим сбалансированным компромиссом между вычислительной эффективностью и точностью. Идеально подходит для задач, требующих умеренных вычислительных ресурсов.

DenseNet-169: с 169 слоями этот вариант обеспечивает более глубокое извлечение признаков, подходит для более сложных наборов данных, где требуется более высокая точность.

DenseNet-201 и DenseNet-264: эти варианты предлагают еще более глубокие архитектуры, подходящие для очень сложных задач, требующих обширного представления признаков.

На официальном сайте keras.io представлено описание 3 вариантов реализации и особенностей DenseNet. Необходимо ознакомиться и применить все три варианта на одном из наборов данных.

2.3 Разбор кода

В предлагаемом для рассмотрения блокноте представлены 11 блоков кода, которые будут описаны ниже.

1. Импорт всех необходимых библиотек.

2. Реализация функции «initiateGenerator(path)» для инициализации генераторов данных для обучения и валидации моделей машинного обучения, использующих изображения.

Функция принимает один аргумент *path*, который указывает на путь к директории, содержащей изображения для обучения. Загрузка набора данных проходит через переменную *train_dataset*. Создание генератора данных через train_datagen, отвечающую за создание генератора изображений с разделением на обучающий (80%) и валидационный (20%) наборы данных. Генераторы создаются с использованием метода flow_from_directory, который загружает изображения из директории и применяет предварительную обработку (например, изменение размера до 224x224 пикселей). Функция выводит количество классов и их названия, которые определяются из train_dataset.class_names. В конце функция возвращает количество классов, названия классов, а также генераторы для обучения и валидации.

3. Реализация функции «initiateNormalize», которая предназначена для нормализации изображений в наборе данных, который используется для обучения и валидации моделей машинного обучения (рисунок 2.2).

```
def initiateNormalize():
    AUTOTUNE = tf.data.AUTOTUNE #AUTOTUNE позволяет TensorFlow автоматически выбирать оптимальное количество потоков для загрузки данных,
                                #что может ускорить процесс обучения.

    train_ds = train_generator.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE) #Подготовка тренировочного и валидационного наборов данных.
    val_ds = val_generator.cache().prefetch(buffer_size=AUTOTUNE) #Данные загружаются ассинхронно, для ускорения обучения.
    normalization_layer = layers.Rescaling(1./255)  #Создание слоя нормализации.

    normalized_ds = train_ds.map(lambda x, y: (normalization_layer(x), y)) #Применение нормализации.
    image_batch, labels_batch = next(iter(normalized_ds))
    first_image = image_batch[0]
    print(np.min(first_image), np.max(first_image)) #Вывод значений пикселей первого изображения.
```

Рисунок 2.2 – Реализация функции инициализации нормализации

4. Инициализация модели «DenseNet201».

5. Реализация функция «initiateParams», которая выполняет инициализацию параметров и настройку модели машинного обучения. В которой задаётся оптимизатор, вид компиляции модели. Проходит настройка механизма для динамического изменения скорости обучения в зависимости от производительности модели. А также настройка механизма сохранения лучших версий модели во время обучения.

6. Реализация функции «modelFit», которая позволяет динамически изменять скорость обучения и сохранять лучшие версии модели во время обучения. Запускает процесс обучения модели с использованием предоставленных данных и параметров. Возвращает историю обучения, что позволяет анализировать производительность модели после завершения обучения.

7. Реализация функции «plotOutput», которая служит для визуализации результатов обучения модели, позволяя анализировать, как изменялись точность и потери на обучающей и валидационной выборках в процессе обучения.

8. Реализация функции «evalModel», предназначенной для оценки производительности модели машинного обучения на валидационном наборе данных.

9. Сохранение модели.

10. Построение матрицы запутанности, для последующего анализа результатов работы модели.

11. Реализация функции «callPlot», которая предназначена для визуализации результатов классификации модели машинного обучения с помощью матрицы запутанности.

Блок № 12 предназначен для реализации задания по варианту.

Для выполнения задания предлагается сделать несколько копий блокнота, чтобы проверить результаты работы модели при разном количестве слоёв (DenseNet-121, DenseNet-169, DenseNet-201).

2.4 Задание на лабораторную работу

Задание на лабораторную работу:

1. необходимо загрузить набор данных по варианту в нейронную сеть;

2. применить сверточную нейронную сеть «DenseNet» с разным количеством слоев;

3. необходимо написать отчёт, содержащий информацию о использованном наборе данных, описание подбора параметров для достижения высокого уровня показателя доли верных ответов (>80%), графики результатов обучения, сделать вывод о влиянии количества слоев свертки на результаты работы сети.

3 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое сверточная нейронная сеть?

2. Какие основные составляющие у сверточной нейронной сети?

3. Чем отличается DenseNet от традиционных архитектур CNN?

4. На что влияет изменение слоев в архитектуре нейронной сети (на примере DenseNet)?

ЛАБОРАТОРНАЯ РАБОТА №3

Использование ИИ для распознавания речи и голоса для людей с нарушением речи

Целью данной лабораторной работы является изучение процесса построения, модификации модели нейронной сети и подбор параметров обучения, обучение и тестирования нейронной сети для классификации аудио фраз пациентов, находившихся на стадиях до операции, после операции и во время реабилитации.

1 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Афазия – это расстройство ранее сформированной речевой деятельности, при котором частично или полностью утрачивается способность пользоваться собственной речью и/или понимать обращенную речь.

Свёрточная нейронная сеть (CNN, Convolutional Neural Network) – это вид искусственной нейронной сети, которая хорошо подходит для обработки данных с сетчатой структурой, таких как изображения и аудиоданные, представленные в виде спектрограмм или матриц признаков. CNN выделяется способностью автоматически извлекать пространственные или временные зависимости из данных, благодаря чему её часто используют для задач компьютерного зрения, обработки естественного языка и обработки аудиосигналов.

Основная идея CNN заключается в использовании свёрточных слоёв и пулинговых слоёв для выделения признаков из исходных данных, что делает её более эффективнее по сравнению с полносвязными нейронными сетями.

Требования к выполнению лабораторной работы

1. Ознакомиться с методическими указаниями.
2. Ознакомиться с представленным набором данных: https://drive.google.com/drive/folders/1rbRnI4Wx-0gQPfD9EQqK9wn2YeTvF8R_?usp=sharing.
3. Ознакомиться с кодом, представленным в блокноте Google Colab: https://colab.research.google.com/drive/1sLborhZls0WatlUa49Z_KWIFEFDdRD4g?usp=sharing.
4. Выполнить ход работы.
5. Модифицировать код согласно заданию.
6. Написать отчёт и отразить результаты до и после изменения кода.

2 ХОД РАБОТЫ

2.1 Разбор набора данных

Набор данных состоит из 1395 аудиозаписей пациентов, длиной от 2 до 5 секунд. Всего пациентов было 20 человек. Каждый пациент проводил несколько сеансов записей, самый ранний сеанс – это до операции, следующий сеанс – после операции, дальнейшие сеансы – во время реабилитации. В каждом сеансе пациенты записывали по 25 фраз. Названия файлов соответствуют их номеру (рисунок 2.1).

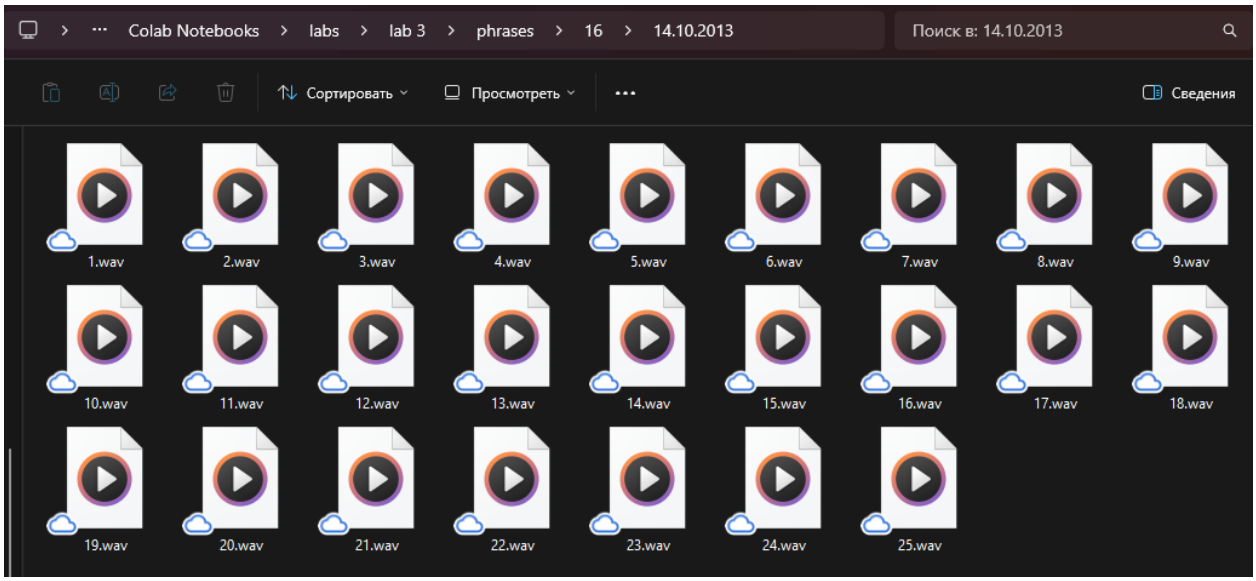

Рисунок 2.1 – Пример представленных данных

2.2 Импорт библиотек и установка начальных параметров

Для выполнения лабораторной работы требуется установить следующий библиотеки (рисунок 2.2):

- os – для работы с файловой системой;
- librosa – для обработки аудиофайлов;
- numpy – для более удобной работы с данными;
- matplotlib.pyplot – для визуализации данных и графиков функций;

– sklearn.model_selection – для случайного разделения выборки на обучающую и тестовую;

– tensorflow.keras.models – для создания последовательной модели нейронной сети;

– tensorflow.keras.layers – для использования видов слоёв модели нейронной сети;

– tensorflow.keras.utils – для приведения к one-hot кодировки (пример, замена метки 1 на массив [0, 1, 0, 0 …, 0]);

– tensorflow.keras.regularizers – для регуляризации (контроль величины весов для предотвращения переобучения).

```
import os
import librosa
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, BatchNormalization
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.regularizers import l2
```

Рисунок 2.2 – Импорт библиотек

Также следует установить начальные параметры (рисунок 2.3).

```
SAMPLE_RATE = 16000 # частота дискретизации
N_MFCC = 40  # Количество коэффициентов MFCC
FIXED_DURATION = 5  # Длительность записи в секундах
MAX_LENGTH = SAMPLE_RATE * FIXED_DURATION // 512  # Примерное количество временных фреймов для 5 секунд
data = []
labels = []
base_dir = '/content/drive/MyDrive/Colab Notebooks/labs/lab 3/phrases'  # Путь к корневой папке
```

Рисунок 2.3 – Установка начальных параметров

2.3 Загрузка и обработка данных

Перед обучением модели необходимо сначала загрузить и преобразовать аудиоданные в набор данных, принимаемых модели CNN. Функция extract_features принимает путь к аудиофайлу и возвращает трёхмерный массив, содержащий MFCC-коэффициенты (Mel-frequency cepstral coefficients), которые дополнительно обрабатываются для фиксации

длины. Сначала аудиофайл загружается с параметром частоты дискретизации (16000 Гц), максимальное продолжительностью (5 секунд). Данные будут дополняться пустыми значениями, если короче указанной длины. Дальше вычисляются MFCC-признаки. MFCC – это спектральные признаки, которые часто используются для анализа и обработки аудиосигналов, таких как голос или музыка. Указывается количество признаков для каждого временного шага и частота дискретизации. В результате получается двумерный массив. Чтобы данные были совместимы со слоями CNN, в массив необходимо добавить третью меру (рисунок 2.4).

```python
def extract_features(file_path):
    # Загрузка аудио с фиксированной длиной
    audio, _ = librosa.load(file_path, sr=SAMPLE_RATE, duration=FIXED_DURATION)
    # Извлечение MFCC признаков
    mfccs = librosa.feature.mfcc(y=audio, sr=SAMPLE_RATE, n_mfcc=N_MFCC)
    # Приведение MFCC к фиксированной длине
    mfccs = librosa.util.fix_length(mfccs, size=MAX_LENGTH, axis=1)
    # Добавляем дополнительное измерение для использования в CNN
    mfccs = np.expand_dims(mfccs, axis=-1)
    return mfccs
```

Рисунок 2.4 – Функция для извлечения MFCC признаков

Следующая функция предназначена для перебора всех файлов из коренной папки, загрузке и обработке аудиоданных и записи их в массивы данных и соответствующие им метки (рисунок 2.5).

```python
def load_data(data_dir):
  for patient in os.listdir(data_dir):
      patient_dir = os.path.join(data_dir, patient)
      if not os.path.isdir(patient_dir):
          continue
      for session in os.listdir(patient_dir):
          session_dir = os.path.join(patient_dir, session)
          if not os.path.isdir(session_dir):
              continue
          for file_name in os.listdir(session_dir):
              if file_name.endswith('.wav'):
                  file_path = os.path.join(session_dir, file_name)
                  label = int(file_name.split('.')[0]) - 1  # метка от 0 до 24
                  features = extract_features(file_path)
                  data.append(features)
                  labels.append(label)

  return np.array(data), np.array(labels)
```

Рисунок 2.5 – Функция для загрузки данных

После описания функций происходит вызов данных функций для заполнения данных в переменные. Дальше данные разбиваются на тренировочные и тестовые в соотношении 80 на 20. Последние строчки позволяют перевести метки в one-hot кодировку, то есть заменить числа классов на массивы. (например, метка 0 становится в [1, 0, 0, ..., 0], метка 1 становится в [0, 1, 0, ..., 0], метка 2 становится в [0, 0, 1, ..., 0]) (рисунок 2.6).

```
X, y = load_data(base_dir)
print(X.shape)
print(y.shape)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
y_train = to_categorical(y_train, num_classes=25)
y_test = to_categorical(y_test, num_classes=25)
```

Рисунок 2.6 – Подготовка данных

2.4 Создание модели и обучение

Модель создаётся с помощью класса «Sequential», который группирует нескольких слоёв в 1 объект (рисунок 2.7).

```
model = Sequential([
    Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(N_MFCC, MAX_LENGTH, 1),
kernel_regularizer=l2(0.001)),
    BatchNormalization(),
    MaxPooling2D(pool_size=(2, 2)),
    Dropout(0.3),

    Conv2D(128, kernel_size=(3, 3), activation='relu', kernel_regularizer=l2(0.001)),
    BatchNormalization(),
    MaxPooling2D(pool_size=(1, 2)),
    Dropout(0.3),

    Conv2D(256, kernel_size=(3, 3), activation='relu', kernel_regularizer=l2(0.001)),
    BatchNormalization(),
    Dropout(0.4),

    Flatten(),

    Dense(128, activation='relu', kernel_regularizer=l2(0.001)),
    Dropout(0.5),

    Dense(25, activation='softmax')  # 25 классов
])

# Компиляция модели
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

Рисунок 2.7 – Описание модели нейронной сети

Слой Conv2D – сверточный слой, используется для извлечения пространственных признаков из входных данных, таких как изображения или спектрограммы, применяется несколько фильтров (ядер свертки) к входным данным и создает карты признаков. У данного слоя имеются следующие параметры:

– Количество фильтров в слое (32, 64, 128, 256) – количество фильтров – чем больше фильтров, тем больше признаков извлекается на данном уровне (например, Conv2D(32, ...) создает 32 карты признаков);

– kernel_size=(n, m) – размер ядра свертки – это размер окна, которое «скользит» по входному изображению (например, 3x3 пикселя), чтобы извлекать признаки;

– activation='relu' – функция активации, применяемая после свертки, relu (Rectified Linear Unit) обнуляет отрицательные значения и пропускает положительные, что позволяет нейронной сети обучаться нелинейным зависимостям;

– input_shape=(N_MFCC, MAX_LENGTH, 1) – форма входных данных – задает размер входного тензора для первого слоя;

– kernel_regularizer=l2(0.001) – регуляризация, которая добавляет штраф за большие значения весов, чтобы предотвратить переобучение.

Слой BatchNormalization – слой нормализации мини-батчей, который нормализует выходы предыдущего слоя по среднему значению и стандартному отклонению в каждом мини-батче. Это стабилизирует и ускоряет процесс обучения. Нормализация помогает избежать проблемы затухающих градиентов и позволяет использовать более высокие значения скорости обучения, что ускоряет сходимость модели.

Слой MaxPooling2D – слой подвыборки (пулинга), который уменьшает размерность карты признаков, извлекая только самые важные признаки. У данного слоя имеется следующий параметр:

– pool_size=(2, 2) – это размер окна для операции пулинга, в данном случае 2x2. Это значит, что из каждого участка размером 2x2 на

входе выбирается одно максимальное значение. Такая операция сокращает размер карты признаков вдвое по каждой оси, упрощая данные и уменьшая их объем для следующего слоя сети.

Слой Dropout – слой, который случайным образом отключает определенный процент нейронов в слое во время обучения. В данном слое указывается число от 0 до 1 – это вероятность отключения нейронов. Данный слой предназначен для борьбы с переобучением. Отключение случайных нейронов заставляет сеть быть более устойчивой к небольшим изменениям и улучшает способность к обобщению.

Слой Flatten – слой, который преобразует многомерные выходы сверточных слоев в одномерный вектор. Данный слой предназначен для таких слоёв как Dense, которые принимают одномерные векторы.

Слой Dense – полносвязный слой, где каждый нейрон связан с каждым входом. У данного слоя имеются следующие параметры:

– количество нейронов (512, 256, 128) – данный параметр задаёт, сколько признаков модель будет обрабатывать на данном слое. Чем больше нейронов, тем больше информации может «запомнить» слой, что повышает его способность распознавать сложные зависимости данных;

– kernel_regularizer=l2(0.001) – регуляризация, которая помогает контролировать величину весов в полносвязном слое, предотвращая переобучение.

Выходной слой Dense – конечный полносвязный слой. У данного слоя имеются следующие параметры:

– количество нейронов (25) – количество нейронов в выходном слое, которое соответствует числу классов;

– activation='softmax' – функция активации softmax используется для задач классификации с несколькими классами, которая преобразует выходы в вероятности, которые в сумме дают 1, и модель выбирает класс с наибольшей вероятностью.

После описания модель компилируется. Чтобы запустить обучение нейронной сети у модели вызывается функция fit() (рисунок 2.8) в которую подаются такие параметры как:

– обучающая выборка данных;

– метки данных обучающей выборки;

– epochs – данный параметр показывает количество этапов (эпох) обучения, после каждого из которых корректируются веса нейронов;

– batch_size – это количество примеров данных, которые модель обрабатывает за одну итерацию перед обновлением весов. В одной эпохе модель проходит через весь обучающий набор данных, но делит его на меньшие части (пакеты) размером batch_size. Например, если batch_size равен 32, то модель обрабатывает 32 примера, вычисляет на них ошибку и обновляет веса;

– validation_data – параметр, который подаёт тестовые данные, параметр позволяет проверить точность модели на каждой эпохе.

```
history = model.fit(X_train, y_train, epochs=50, batch_size=32, validation_data=(X_test, y_test))
```

Рисунок 2.8 – Запуск обучения

После окончания обучения нейронной сети можно вывести график изменения точности нейронной сети на обучающихся данных и на тестовых на каждой эпохе.

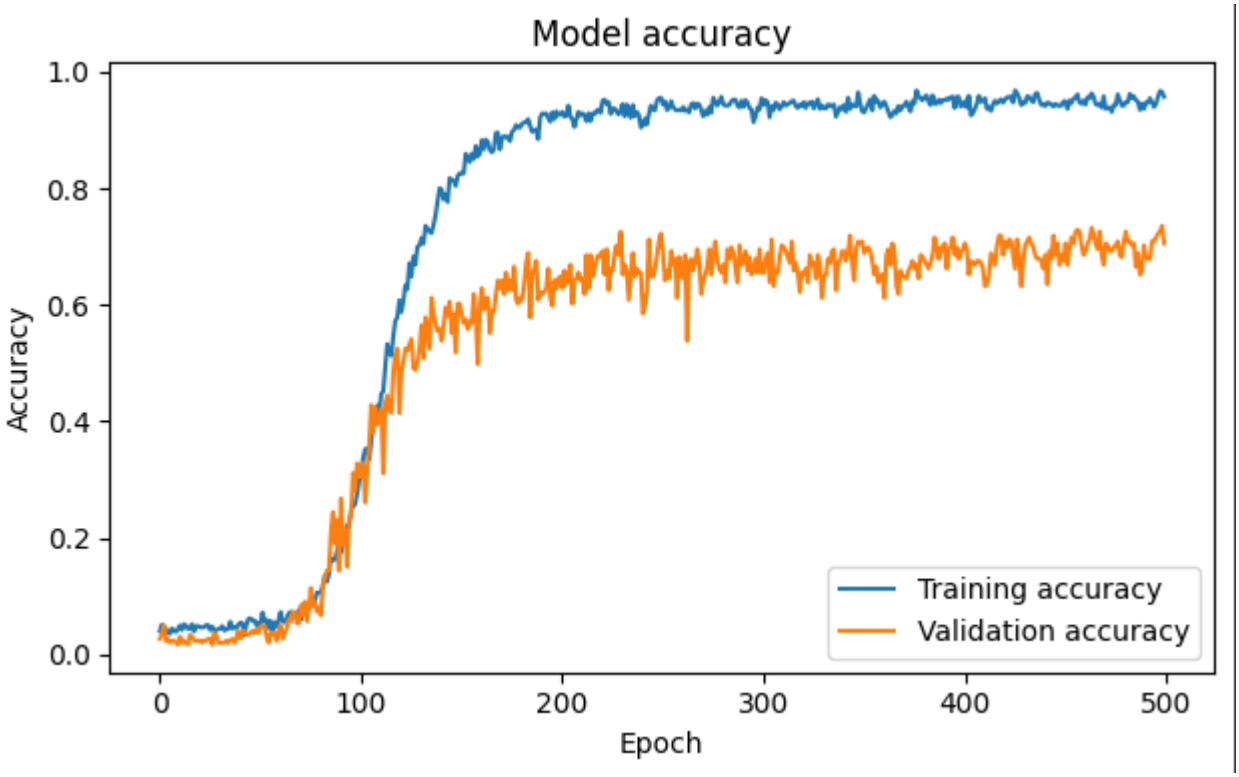

Рисунок 2.9 – Пример визуализации графика

2.5 Задание на лабораторную работу

Задание на лабораторную работу:

– протестировать заготовленную нейронную сеть и зафиксировать полученные результаты в отчёте;

– модернизировать модель нейронной сети, добавив изученные слои (Conv2D, BatchNormalization, MaxPooling2D, Dropout, Dense) с разным количеством нейронов;

– изменить параметры обучения epochs (не менее 300) и batch_size;

– сравнить полученные результаты до изменения кода и после.

Постараться получить результат точности тестовых данных 60-70% похожему как на рисунке 2.9.

3 КОНТРОЛЬНЫЕ ВОПРОСЫ

1 Для чего используются слои Dropout в нейронной сети?

2 Как параметр batch_size влияет на обучение модели?

3 В чем преимущества и недостатки увеличения количества слоев в модели?

4 Какие факторы могут привести к переобучению нейронной сети и как с этим бороться?

5 Как работает принцип one-hot кодировки?

6 Какие библиотеки используются для обучения нейронной сети?

ЛАБОРАТОРНАЯ РАБОТА № 4

Использование ИИ для оценки реабилитации пациентов с нарушениями опорно-двигательного аппарата

Целью работы является исследование применения методов машинного обучения при анализе реабилитации пациентов с нарушениями опорно-двигательного аппарата или проходящих восстановительные программы после полученных физических травм.

1 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Transfer Learning — подраздел машинного обучения, целью которого является применение результатов и знаний, полученных при обучении на одной задаче, на другой задаче. Трансферное обучение предполагает использование предобученных моделей.

Предобученные нейронные сети — это модели, созданные и натренированные на большом наборе общедоступных данных. Чаще всего их разрабатывают крупные технологические компании или ведущие исследователи в области компьютерного зрения. Такие модели позволяют ускорить и улучшить обучение, так как дают возможность стартовать с вектора весов, обученного на внешнем наборе данных.

Требования к выполнению лабораторной работы

1. Ознакомиться с методическими указаниями, а также представленным примером в блокноте Google Colab: https://colab.research.google.com/drive/1v7XkegRlxQJREwXNSqftzfT2o0i8p2s4?usp=sharing.

2. Ознакомиться с набором данных: https://drive.google.com/drive/folders/15D_0cqU6HdU9ZEnmTKiGButcbCMgFOpz?usp=drive_link.

3. Аналогично примеру построить и обучить модели, согласно индивидуальному варианту (таблица 1.1).

4. Рассмотреть результаты работы с применением предобученной модели resnet18 и моделей по варианту. Вывести графики обучения трёх моделей, а также внести в отчет результаты их работы (точность классификации, показатель ошибки, f1-мера).

5. Ответить на контрольные вопросы.

Таблица 1.1 – Варианты для выполнения индивидуального задания

Вариант	Модель 1	Модель 2
1	3	2
2	1	3
3	1	4
4	2	3
5	2	4
6	3	4
7	1	2

Список исходных моделей:

1. tf.keras.applications.vgg16;
2. tf.keras.applications.vgg19;
3. tf.keras.applications.resnet50;
4. tf.keras.applications.efficientnetb0.

2 ХОД РАБОТЫ

2.1 Разбор набора данных

Перед началом выполнения работы необходимо ознакомиться с набором данных «Artrit».

Остеоартрит колена определяется дегенерацией суставного хряща колена — гибкого, скользкого материала, который обычно защищает кости от трения и ударов суставов. Это состояние также включает изменения в кости под хрящом и может поражать близлежащие мягкие ткани.

Набор данных содержит данные рентгенографии колена как для обнаружения нарушений коленного сустава, так и для оценки колена по шкале Калгрена-Лоуренса (KL). Описания степеней:

Степень 0: здоровое изображение колена.

Степень 1 (сомнительная): сомнительное сужение сустава с возможным остеофитным выступом.

Степень 2 (минимальная): определенное наличие остеофитов (костных наростов) и возможное сужение суставной щели.

Степень 3 (умеренная): множественные остеофиты, определенное сужение суставной щели, с легким склерозом.

Степень 4 (тяжелая): крупные остеофиты, значительное сужение сустава и тяжелый склероз.

2.2 Построение модели

Необходимо ознакомиться с кодом в блокноте, в котором представлено 11 действий, включающих в себя:

1. Для начала проходит импорт необходимых библиотек для обработки данных, построения моделей, обучения и оценки.

2. Следом необходимо создать функцию «load_dataset_as_dataframe()» для загрузки набора данных из подготовленных подкаталогов train/ и test/, а также организации пути к изображениям и меткам. Затем представлен переход по каждой папке класса (представляющей степени тяжести остеоартрита: ['0', '1', '2', '3', '4']). Список классов содержит все имена классов, а class_to_idx сопоставляет эти имена классов с числовыми метками.

3. Далее с использованием функции из второго шага, необходимо загрузить тестовый и обучающий набор данных как dataframe.

4. Также необходимо определить параметры преобразований для аугментации и нормализации данных. В примере: изображения изменяются до размера 224x224 пикселей, чтобы соответствовать размеру входных данных, ожидаемому ResNet. Аугментация данных применяется к обучающему набору со случайными горизонтальными переворотами. Изображения нормализуются с использованием значений среднего и стандартного отклонения ImageNet.

5. Следом для предварительной обработки изображений и последующей загрузки в сеть нужно создать пользовательский класс набора данных. Класс «KneeDataset» наследуется от torch.utils.data.Dataset. Метод getitem загружает и возвращает изображение и его метку.

6. Далее создаются объекты DataLoader для пакетирования и перемешивания данных, batch_size=32 указывает количество образцов в пакете, shuffle=True рандомизирует порядок данных в каждой эпохе в обучающем наборе данных. Проходит создание загрузчиков данных.

7. Затем необходимо проверить прошедшую загрузку и преобразование данных. Для визуализации пакета обучающих изображений, нужно определить функцию imshow для отображения изображений после их преобразования, а также используя функцию utils.make_grid вывести сетку изображений.

8. Наконец совершен переход к загрузке и изменении предварительно обученной модели ResNet. Стоит отметить, что изменяется последний слой в соответствии с количеством классов.

9. Далее необходимо определить функции потерь и оптимизатора. В примере: функция CrossEntropyLoss подходит для многоклассовой классификации. Оптимизатор Adam используется со скоростью обучения 0,001.

10. Следом нужно обучить модель. Путём перебора обучающих данных, вычисляются потери, выполняется обратное распространение и обновление веса модели, model.train() устанавливает модель в режим обучения.

11. В завершении необходимо оценить результаты обучения модели. Для начала работоспособность модели необходимо проверить на тестовом наборе данных. Используя функцию model.eval(), которая переводит модель в режим оценки. А также функцию torch.no_grad(), которая отключает вычисление градиента.

Матрица запутанности описывает ошибки классификатора и типы возникающих ошибок, отчет о классификации даёт общую оценку производительности модели по различным оценочным показателям.

2.3 Задание на лабораторную работу

Задание на лабораторную работу:

1. Загрузить представленный набор данных в нейронную сеть;

2. По примеру построить и обучить модели, согласно варианту;

3. Рассмотреть результаты работы с применением предобученной модели resnet18 и моделей по варианту;

4. Вывести графики обучения трёх моделей, а также внести в отчет результаты их работы.

3 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое предобученная сеть?

2. Какие особенности предобученной нейронной сети, вы можете выделить?

2. На каких данных происходит предварительное обучение моделей, для дальнейшего дообучения?

3. Для чего используется dataframe?

4. Для чего используется заморозка последних слоёв модели?

ЛАБОРАТОРНАЯ РАБОТА №5

Соревнование моделей нейронной сети для классификации аудиоданных

Целью данной лабораторной работы является закрепление навыков работы с нейронными сетями для классификации аудио, научиться сохранять и загружать обученные модели, а также проводить тестирование точности моделей на неизвестных данных. Научиться сравнивать модели, проверяя и тестируя результаты собственной модели с моделями других студентов.

1 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Сохранение и загрузка модели нейронной сети в файл используется для того, чтобы её не нужно было обучать заново каждый раз, когда требуется сделать предсказания. Это значительно экономит время и ресурсы, так как обучение нейронной сети – длительный процесс, требующий мощных вычислительных ресурсов, особенно при работе с большими наборами данных.

Сохранённая модель позволяет выполнять ряд задач:

– повторное использование модели для заготовленной задачи без траты времени и ресурсов на повторное обучение;

– дальнейшее исследование и доработка модели (дообучение);

– развёртывание модели в приложениях или сервисах.

Загрузка модели из файла – это процесс, при котором ранее сохранённая модель нейронной сети восстанавливается с полными параметрами и архитектурой, чтобы быть сразу готовой к дальнейшему использованию.

1.1 Требования к выполнению лабораторной работы

1. Ознакомиться с методическими указаниями.

2. Воспользоваться набором данных с лабораторной работы №3 https://drive.google.com/drive/folders/1HYDutjc4dYsAyoB87B1wj8uFySFxFfJV?usp=sharing.

3. Воспользоваться кодом лабораторной работы №3.

4. Модифицировать модель или написать более сложную модель для соревнования моделей между студентами (использовать теоретический материал из лабораторной работы №3).

5. Сохранить свою модель, обменяться с другими студентами.

6. Протестировать модели на неизвестных данных в папке 77 из набора данных.

7. Написать отчёт, описать свою модель, составить сравнительную таблицу моделей студентов, представленной на диске и выделить лучшую модель.

2 ХОД РАБОТЫ

2.1 Сохранение модели

Чтобы сохранить модель в файл используется метод save (рисунок 2.1). Модель сохраняется в файл формата «.h5» со всей архитектурой и весами.

```
model.save('/content/drive/MyDrive/Colab Notebooks/labs/lab 5/model.h5')
```

Рисунок 2.1 – Сохранение модели в файл

2.2 Загрузка модели

Для загрузки модели нужно импортировать модуль load_model из библиотеки Keras, после чего можно загрузить в переменную модель из файла с помощью одноимённой функции (рисунок 2.2).

```
from tensorflow.keras.models import load_model

loaded_model = load_model('/content/drive/MyDrive/Colab Notebooks/labs/lab 5/model.h5')
```

Рисунок 2.2 – Пример загрузки модели из файла

2.3 Тестирование модели на данных

Для тестирования модели первым делом нужно определить путь к папке с данными, которые будут использоваться (рисунок 2.3).

```
test_path = '/content/drive/MyDrive/Colab Notebooks/labs/lab 5/77'
```

Рисунок 2.3 – Определение папки с тестовыми данными

Следующим этапом является написание функции для оценки точности работы модели (рисунок 2.4).

```python
def evaluate_patient_model(model, patient_folder_path):
    total_predictions = 0
    correct_predictions = 0

    # Проходим по всем сеансам пациента
    for session_folder in os.listdir(patient_folder_path):
        session_path = os.path.join(patient_folder_path, session_folder)

        # Проходим по каждой аудиозаписи в сеансе
        for audio_file in os.listdir(session_path):
            file_path = os.path.join(session_path, audio_file)

            # Извлечение признаков с использованием функции extract_features
            features = extract_features(file_path)
            features = np.expand_dims(features, axis=0)  # Добавляем измерение для batch

            # Предсказание модели
            prediction = model.predict(features)
            predicted_class = np.argmax(prediction)

            # определение правильного класса
            true_class = int(audio_file.split('.')[0]) - 1

            if predicted_class == true_class:
                correct_predictions += 1
            total_predictions += 1

    # Расчет точности для текущего пациента
    accuracy = correct_predictions / total_predictions if total_predictions > 0 else 0
    print(f"Точность модели на данных пациента: {accuracy * 100:.2f}%")
```

Рисунок 2.4 – Функция тестирования модели

Чтобы рассчитать точность модели, нужно найти соотношение правильных предсказаний ко всем предсказаниям. Для этого определяются переменные *total_predictions* – общее количество предсказаний, и *correct_predictions* – количество правильных предсказаний.

Так как данные представляют собой папки с пациентами, в которых расположены папки с посещением сеансов, в которых расположены аудиоданные, необходимо сначала описать цикл прохода по каждому сеансу. Внутри данного цикла описывается также цикл прохода по каждой аудиозаписи.

Следующий блок функции заключается в извлечении признаков с помощью функции extract_features, описанной в лабораторной работе №3. Дальнейшая строчка кода добавляет измерение для размера batch. Данное действие необходимо, чтобы подаваемый массив соответствовал ожидаемому входным данным.

Следующий блок предсказания. С помощью функции predict() подаются данные в модель и получается результат. В результате в переменную prediction записывается одномерный массив весов размером 25, соответствующе количеству классов. Чтобы понять, какой класс определила нейронная сеть, выбирается индекс массива с максимальным весом. После чего сравнивается, индекс класса с настоящим классом (настоящий класс указан в названии аудиофайла).

Для расчёта метрик используется библиотека sklearn (рисунок 2.5). Также с данной библиотеки импортируется функция для расчёта матрицы ошибок.

```
from sklearn.metrics import precision_score, recall_score, f1_score, confusion_matrix
```

Рисунок 2.5 – Подключение библиотеки метрик

В следующем блоке рассчитываются метрики (рисунок 2.6):

– accuracy (точность) – показывает общую точность классификации, определяется как доля правильно классифицированных объектов среди всех объектов;

– precision (точность) – соотношение правильных предсказаний к общему числу положительных предсказаний, определяется как доля правильных предсказаний данного класса среди всех объектов, которые модель отнесла к конкретному классу;

– recall (полнота) – соотношение правильных положительных предсказаний к общему числу истинных положительных результатов;

– F1 – гармоническое среднее точности и полноты, полезное, когда требуется баланс между precision и recall.

```python
# Вычисление метрик
    accuracy = correct_predictions / total_predictions if total_predictions > 0 else 0
    precision = precision_score(true_labels, predicted_labels, average='weighted')
    recall = recall_score(true_labels, predicted_labels, average='weighted')
    f1 = f1_score(true_labels, predicted_labels, average='weighted')

    print(f"Точность модели на данных пациента: {accuracy * 100:.2f}%")
    print(f"Precision (Точность): {precision * 100:.2f}%")
    print(f"Recall (Полнота): {recall * 100:.2f}%")
    print(f"F1-Score: {f1 * 100:.2f}%")
```

Рисунок 2.6 – Вычисление метрик

Последним блоком является построение матрицы запутанностей с помощью библиотеки seaborn (рисунок 2.7).

```python
import seaborn as sns
```

Рисунок 2.7 – Подключение библиотеки для визуализации матрицы запутанностей

Первым делом происходит расчёт матрицы запутанностей помощью ранее импортированной функции confusion_matrix (рисунок 2.8). Матрица запутанностей позволяет определить, какие классы путаются между собой, и позволяет увидеть слабые места модели, то есть определить классы с низкой точностью.

```python
# Построение матрицы ошибок
    cm = confusion_matrix(true_labels, predicted_labels)
    plt.figure(figsize=(10, 8))
    sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=range(25), yticklabels=range(25))
    plt.xlabel("Предсказанный класс")
    plt.ylabel("Истинный класс")
    plt.title("Матрица ошибок")
    plt.show()
```

Рисунок 2.8 – Листинг кода вывода матрицы запутанностей

При вызове данной функции каждый файл проходит модель и выводится результат (рисунки 2.9 и 2.10).

```
evaluate_patient_model(loaded_model, test_path)

1/1 ━━━━━━━━━━━━━━━━━━━━ 3s 3s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 73ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step
```

Рисунок 2.9 – Пример вывода тестирования

```
Точность модели на данных пациента: 62.00%
Precision (Точность): 71.73%
Recall (Полнота): 62.00%
F1-Score: 61.51%
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531: UndefinedMetricWarnin
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

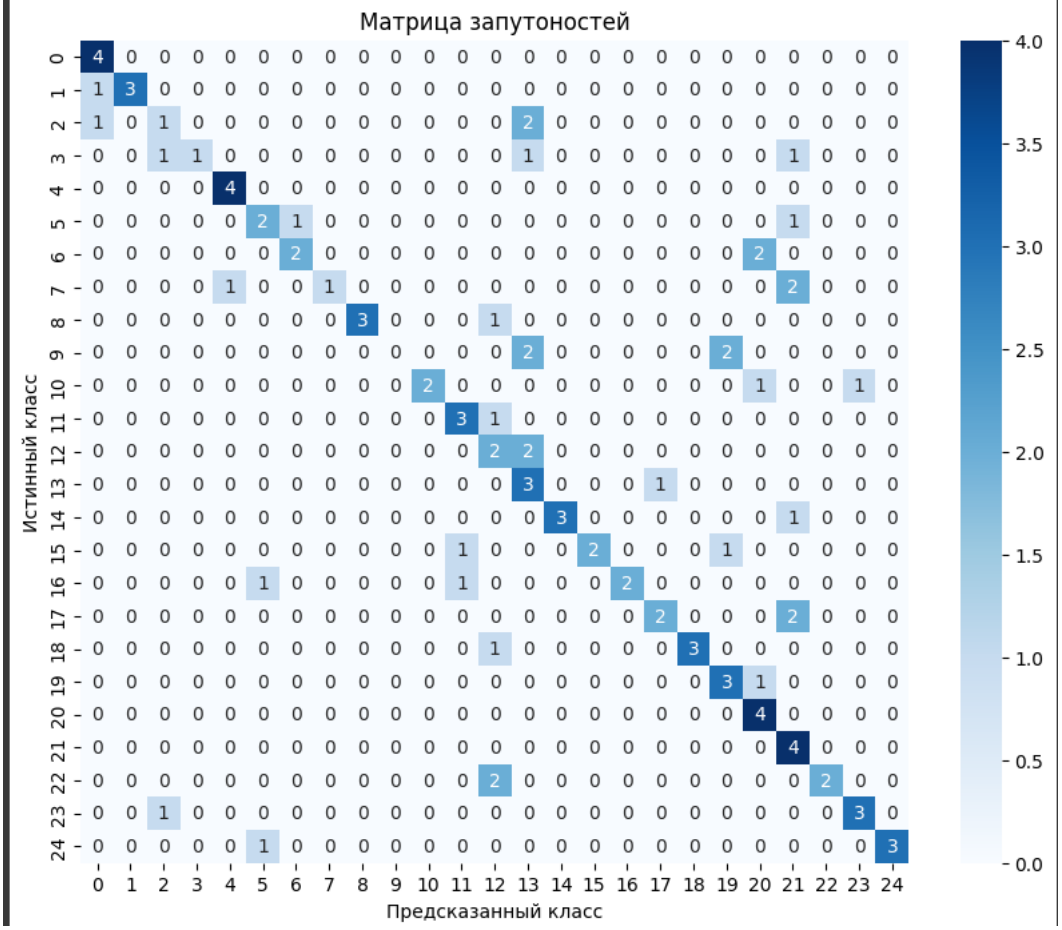

Рисунок 2.10 – Вывод результатов тестирования

2.4 Задание на лабораторную работу

Задание на лабораторную работу:

– модернизировать модель нейронной сети с лабораторной работы №3;

– сохранить модель в виде файла;

– обменяться моделями с одногруппниками;

– загрузить и проверить свою модель, модели одногруппников, модель, представленную на папке 77 на диске;

– выполнить сравнение полученных результатов после тестирования моделей;

– написать отчёт, включающий описание собственной модели и сравнительную таблицу между моделями, сделать выводы по проделанной работе.

3 КОНТРОЛЬНЫЕ ВОПРОСЫ

1 Почему важно сохранять обученную модель в файл? Какие преимущества это даёт?

2 Чем отличается загрузка модели из файла от её обучения?

3 Почему необходимо добавлять размерность batch к данным перед тем, как подать их на вход модели? Как это делается?

4 Для чего используется матрица запутанностей?

5 Для каких задач может пригодиться сохранение и загрузка модели?

ПРАКТИЧЕСКАЯ РАБОТА № 1

Исследование применения модели искусственного интеллекта для оценки функциональной активности человека с инвалидностью

Цель работы: изучить подходы к применению моделей искусственного интеллекта для анализа данных, связанных с оценкой функциональной активности людей с инвалидностью, и разработать схему предварительной обработки данных для последующего использования в модели искусственного интеллекта.

Задачи:

– изучить типы данных, которые используются для оценки функциональной активности человека с инвалидностью;

– провести анализ существующих методов обработки данных для последующего ввода в модель искусственного интеллекта;

– разработать схему обработки данных;

– подготовить визуальную презентацию (схемы, диаграммы) работы группы для обсуждения результатов.

Работа выполняется в 3 группах по направлениям:

1. данные с носимых датчиков (например, акселерометры, пульсометры);

2. данные видеонаблюдения (видеоизображения и их обработка);

3. текстовые данные (медицинские записи, отчеты врачей).

Результаты работы необходимо представить в виде схемы обработки данных (UML-диаграммы, чёрный ящик, DFD или другие) в презентации.

ПРАКТИЧЕСКАЯ РАБОТА № 2

Рассмотрение реализации применения модели искусственного интеллекта для оценки функциональной активности человека с инвалидностью

Цель работы: разработать концептуальную модель искусственного интеллекта, способную оценивать функциональную активность человека с инвалидностью, используя результаты анализа данных из практической работы №1.

Задачи:

- определить архитектуру модели искусственного интеллекта и основные компоненты, которые в неё входят (тип модели, метод обучения, структура нейронной сети);
- выбрать и обосновать методы машинного обучения или глубокого обучения, которые лучше всего подходят для анализа данных конкретного типа;
- сформировать схему входных и выходных данных модели, включая методы обработки входных данных (из практической работы №1);
- определить целевые метрики и критерии оценки качества работы модели;
- подготовить презентацию с описанием модели, её архитектуры, применяемых методов и ожидаемых результатов.

Работа выполняется в 3 группах по направлениям:

1. данные с носимых датчиков;
2. данные видеонаблюдения;
3. текстовые данные.

Результаты работы необходимо представить в виде групповой презентации с докладом.

ПРАКТИЧЕСКАЯ РАБОТА № 3

Исследование использования ИИ для разработки индивидуализированных программ реабилитации для больных с заболеваниями нервной системы (часть 1)

Цель работы: развитие навыка анализа материалов (данных) в рамках заданной темы; углубление понимания предметной области.

Задачи:

– найти наборы данных по заболеваниям нервной системы;

– найти и ознакомиться с готовыми моделями машинного обучения/системы на основе искусственного интеллекта, использующиеся при реабилитации пациентов с заболеваниями нервной системы;

– подготовить сравнительную таблицу, найденных методов обработки наборов данных;

– подготовить презентацию с описанием найденных наборов данных и моделей.

При выполнении данной практической работы, необходимо обратить внимание на то, что не все заболевания подвластны реабилитации, а также рассмотреть вопрос влияния стадии болезни на процесс восстановления.

Работа выполняется в 3, 4 группах. Каждой группе необходимо отобрать наборы данных, связанные с заболеваниями нервной системы, по следующим направлениям:

1. детский церебральный паралич;
2. инсульт;
3. болезнь Паркинсона;
4. рассеянный склероз.

ПРАКТИЧЕСКАЯ РАБОТА № 4

Исследование использования ИИ для разработки индивидуализированных программ реабилитации для больных с заболеваниями нервной системы (часть 2)

Цель работы: разработать концептуальную модель приложения с технологиями искусственного интеллекта для реабилитации пациентов с заболеваниями нервной системы.

Задачи:

– ознакомиться с предметной областью;

– определить наиболее подходящие инструменты для работы с заданным заболеванием;

– выполнить проектирование.

При разработке концептуальной модели приложения необходимо использовать структурную функциональную, информационную модели (IDEF0, DFD, USE-case и др.), позволяющие раскрыть все предусмотренные возможности модели.

Задание выполняется в 3, 4 группах по следующим направлениям (реабилитируемым заболеваниям нервной системы):

1. детский церебральный паралич;
2. инсульт;
3. болезнь Паркинсона;
4. рассеянный склероз.

Результатом работы является представление спроектированных моделей. Подготовка презентации.

ПРАКТИЧЕСКАЯ РАБОТА № 5

Исследование использования ИИ для разработки индивидуализированных программ реабилитации для больных с заболеваниями нервной системы (часть 3)

Цель работы: представление результатов разработки концептуальной модели приложения с технологиями искусственного интеллекта для реабилитации пациентов с заболеваниями нервной системы.

Задачи:

– подготовить и выступить с презентацией разработанной модели приложения;

– выставить баллы выступающей группе по таблице 1.

Требования:

– презентация разработанной модели приложения рассчитана на 5-7 минут;

– от каждой группы слушателей необходимо задать не менее двух вопросов;

– балы по таблице 1 выставляет каждый слушатель.

В завершении практической работы выявляется лучший из представленных проектов и проводится общее обсуждение эффективности использования информационных технологий и искусственного интеллекта в медицинской реабилитации.

Таблица 1 – Критерии оценки представленного проекта

Критерий	Балл
Модель приложения может оказаться эффективна при использовании в реабилитации пациентов с заданным заболеванием (оценки: 0 – модель не применима в жизни (по вашему мнению), 1 – модель возможно реализовать и использовать, 2 – в презентации приведено обоснование, выбранных технологий, модель возможно реализовать и использовать в реальной жизни).	

При разработке модели учитывалась рассмотренная предметная область и целевая аудитория (оценки: 0 – на первый взгляд модель предназначена для здоровых людей, уверенно пользующихся современными информационными технологиями, 1 – приложение рассчитано для людей с данным заболеванием (по вашему мнению), 2 – приложение рассчитано для людей с данным заболеванием, обоснованы выбранные технологии).	
В модель приложения была включена оценка динамики реабилитации пациентов (оценки: 0 – параметр для оценки динамики реабилитации не был включен в модель, 1 – параметр для оценки динамики реабилитации был включен в модель, 2– параметр для оценки динамики реабилитации был включен и обоснован).	

ПРАКТИЧЕСКАЯ РАБОТА № 6

Рассмотрение совершенствования процесса реабилитации пациентов

Цель работы: проведение сравнительного анализа современных и традиционных методов реабилитации, выявление их недостатков и нахождение альтернативных подходов, в том числе с использованием методов искусственного интеллекта, для улучшения процесса реабилитации пациентов с различными нарушениями.

Задачи:

- изучить выбранное направление реабилитации;
- проанализировать современные и традиционные методы реабилитации по выбранному направлению реабилитации;
- проанализировать недостатки современных и традиционных методов реабилитации;
- на основе недостатков сформулировать решения с применением искусственного интеллекта, которые могут улучшить процесс реабилитации;
- подготовить презентацию, в которой будут отражены предыдущие пункты и обсудить результаты.

Работа выполняется в 3, 4 группах по следующим направлениям:

1. реабилитация пациентов с нарушением голоса;
2. реабилитация пациентов с заболеваниями нервной системы;
3. реабилитация пациентов после травм опорно-двигательного аппарата;
4. социальная реабилитация пациентов.

Результаты работы необходимо представить в виде групповой презентации с докладом.

ПРАКТИЧЕСКАЯ РАБОТА № 7

Рассмотрение реабилитации пациентов по кейсам

Цель работы: изучить и подробно проанализировать существующие решения в сфере применения искусственного интеллекта в реабилитации пациентов, а также представить результаты в виде краткого группового доклада.

Задачи:

– выбрать одно из существующих решений искусственного интеллекта в реабилитации (лекция 7);

– провести анализ работы решения: принцип действия, применяемые методы искусственного интеллекта, пример использования, сильные и слабые стороны;

– подготовить короткий доклад за первую половину пары;

– представить результаты доклада другим группам и провести дискуссию, задавая вопросы и предлагая идеи по улучшению рассмотренных решений за вторую половину пары.

Работа выполняется в 3–5 группах. Представленные в лекции решения:

1. IBM Watson Health;
2. MindMotion® GO;
3. Physera;
4. Woebot;
5. робот Pepper;
6. стартап BrainQ.

Результаты работы необходимо представить в виде группового доклада.

ПРАКТИЧЕСКАЯ РАБОТА № 8

Рассмотрение вопроса повышения качества жизни пациентов, проходящих реабилитацию

Цель работы: исследование вопроса психологической и бытовой части жизни пациентов в период реабилитации.

Задачи:

– используя данные лекционного материала и научные работы, открытые для рассмотрения в поисковой системе «Академия Google» и других, найти примеры повышения качества жизни пациентов в период восстановления за счет информационных технологий и методов искусственного интеллекта;

– создать план-схему (рисунок 1, представлен пример) системы улучшения качества жизни пациентов, проходящих реабилитацию по одному из следующих направлений:

1. заболевания, связанные с нарушениями опорно-двигательного аппарата;

2. заболевания, связанные с нарушениями речи;

3. онкологические заболевания;

4. профессиональные заболевания.

Результаты работы необходимо представить в виде группового доклада, с объяснением разработанной план-схемы.

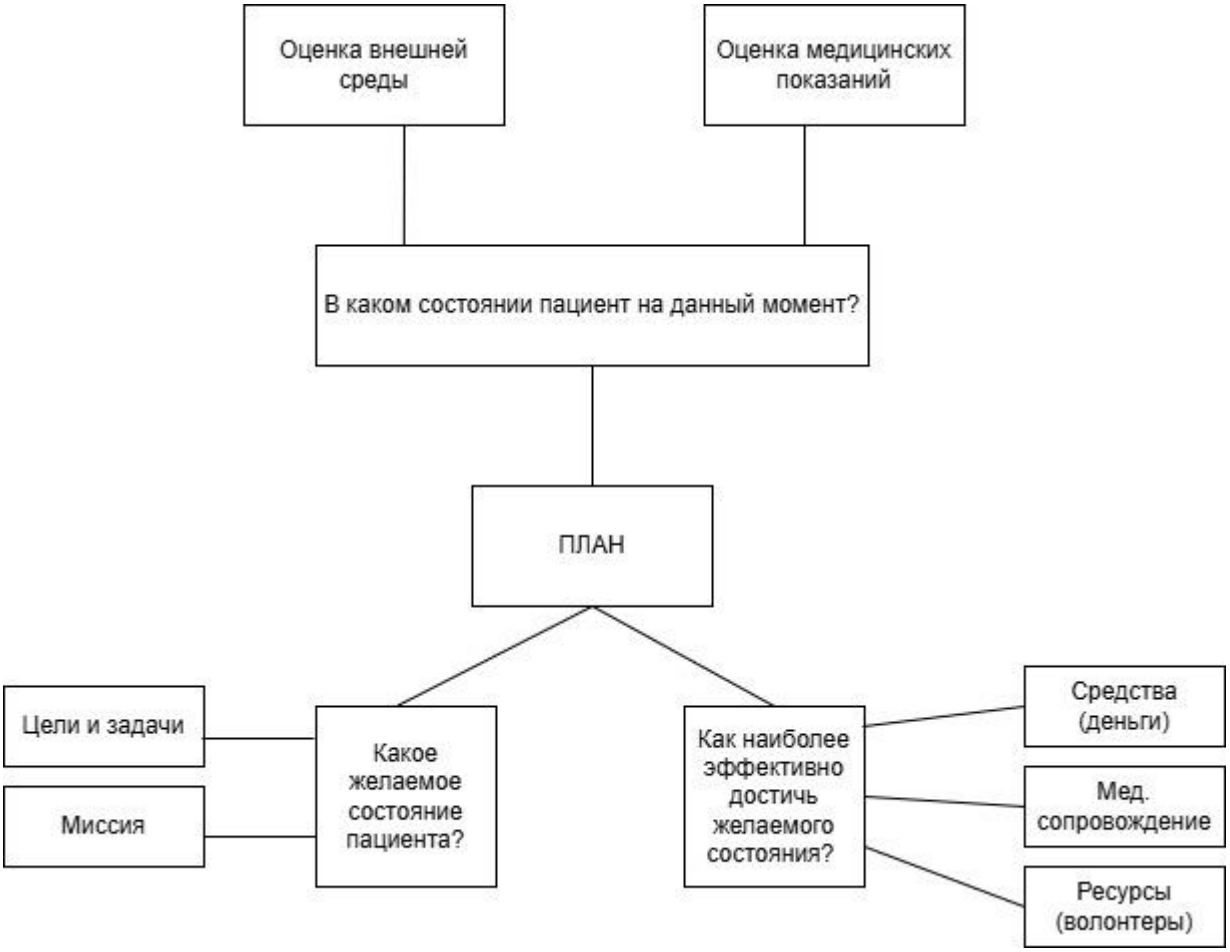

Рисунок 1 – Пример план-схемы

ЛИТЕРАТУРА

1. Гаврилов А. В. и др. Цифровые технологии при работе с медицинскими изображениями //Вестник рентгенологии и радиологии. – 2010. – №. 4. – С. 57-63.

2. Имамеева Р. Д. РИСКИ СОЗДАНИЯ И ФУНКЦИОНИРОВАНИЯ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА В МЕДИЦИНЕ //Вестник Московского университета им. СЮ Витте. Серия 2: Юридические науки. – 2021. – №. 1 (27). – С. 33-40.

3. Федотов А.А. Основы цифровой обработки биомедицинских изображений: учеб. пособие / А.А. Федотов. – Самара: СГАУ, 2013. – 108 с.

4. Замятин, Н. В. Системы искусственного интеллекта: Учебное пособие / Н. В. Замятин. — Томск: ТУСУР, 2018. — 244 с.

5. Scikit-learn Machine Learning in Python [Электронный ресурс]: сайт Scikit-learn. URL: https://scikit-learn.org/stable/ (дата обращения: 16.11.2024).