Министерство науки и высшего образования РФ

Томский государственный университет
систем управления и радиоэлектроники

П.Ю. Лаптев, Р.Н. Абдулин, Е.Д. Часовских, Е.Ю. Костюченко

СЕГМЕНТАЦИЯ И ДЕТЕКТИРОВАНИЕ ОБЪЕКТОВ ПРИ ОБРАБОТКЕ МЕДИЦИНСКИХ ИЗОБРАЖЕНИЙ

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

УДК 004.8
ББК 32.813.5
Л 24

Лаптев, П. Ю. СЕГМЕНТАЦИЯ И ДЕТЕКТИРОВАНИЕ ОБЪЕКТОВ ПРИ ОБРАБОТКЕ МЕДИЦИНСКИХ ИЗОБРАЖЕНИЙ: Учебно-методическое пособие / П. Ю. Лаптев, Р.Н. Абдулин, Е.Д. Часовских, Е.Ю. Костюченко. — Томск: ТУСУР, 2024. — 101 с.

Настоящее учебно-методическое пособие содержит описания лабораторных и самостоятельных работ по дисциплине «Сегментация и детектирование объектов при обработке медицинских изображений» для направлений подготовки, входящих в укрупненную группу специальностей и направлений 09.04.04 Программная инженерия.

Одобрено на заседании кафедры КИБЭВС протокол №7 от 30.08.2024 года

УДК 004.8
ББК 32.813.5

Содержание

Введение

Целью преподавания дисциплины является изучение принципов работы нейронных сетей и примеров применения их на практике.

Задачи изучения дисциплины:

– Изучить основные принципы машинного обучения и нейронных сетей.

– Рассмотреть принципы применения машинного обучения и нейронных сетей для задач компьютерного зрения.

– Рассмотреть принципы применения машинного обучения и нейронных сетей для задач обработки естественного языка.

ЛАБОРАТОРНАЯ РАБОТА №1

Исследование методов обработки шума при сегментации изображений

Целью работы является исследование методов обработки шума при сегментации изображений для улучшения качества и точности результатов.

Краткие теоретические сведения

Шум на изображениях — это дефект, который проявляется в виде заметных случайных пикселей разной яркости и цвета. Шум ухудшает качество и детализацию изображения, особенно на тёмных и однотонных участках.

Внедрение шума в нейронные сети — это процесс добавления случайного шума к входным данным в процессе обучения.

Внедрение шума позволяет предотвратить чрезмерное соответствие модели обучающим данным без помех, побуждая сеть изучать более значимые, обобщаемые шаблоны.

Виды шумов:

1. Белый шум – сигнал, отсчеты которого не коррелируют друг с другом.
2. Белый гауссовский шум – возникает, в частности, при плохих условиях приема сигнал.
3. Импульсный шум – случайные изолированные точки на изображении, значение которых значительно отличается от значений окружающих их точек (обычно возникает при передаче по аналоговым каналам).
4. Цветные пятна – характерны для аналогового сигнала (к примеру, присутствуют в видеоизображении, оцифрованном с видеокассет VHS).
5. Шум вида «соль и перец» – этот шум представляет собой случайно возникающие чёрные и белые пиксели.

Линейное усреднение пикселей работает по принципу усреднения значения пикселей в пространственной окрестности. Для каждого пикселя анализируются соседние для него пиксели, которые располагаются в некотором прямоугольном окне вокруг этого пикселя. Чем больше взят размер окна, тем сильнее происходит усреднение.

Медианная фильтрация – это стандартный способ подавления импульсного шума. Для каждого пиксела в некотором его окружении (окне) ищется медианное значение и присваивается этому пикселю.

Гауссовское размытие – это свертка изображения, вычисляемая по функции 1

$$g(x,y) = A*e^{-\frac{x^2+y^2}{\delta^2}} \quad (1)$$

где x, y — координаты точки,

A – степень размытия,

σ — среднеквадратическое отклонение нормального распределения.

Билатеральное фильтрование - это фильтрование с использованием нелинейной техники. Оно расширяет сглаживание Гаусса, что позволяет увеличивать показатели фильтра за счёт соответствующей ему относительной интенсивностью пикселя, который находится в центре изображения. Наглядно это видно в математических формулах, приведённых ниже. Предположим, что функция f(x,y) – исходная, которая отображает яркость изображения, в зависимости от координат x,y и будет равна яркости текущего изображения в данном пикселе. После этого для любого пикселя на оси, в области размера n, где a0 - центр области, r(a) – коэффициент рангового фильтра, который определяется по формуле (2):

$$r(a_i) = e^{\frac{(f(a_i)-f(a_0))^2}{2\sigma^2}} \quad (2)$$

Убирать шум с изображений важно по нескольким причинам:

1. Повышение качества изображений. Удаление шума улучшает визуальное восприятие изображений, делая их более чёткими и детальными.

2. Улучшение результатов работы нейронных сетей. Шумоподавление позволяет нейронным сетям сосредоточиться на важных деталях изображения, что повышает их точность и эффективность в решении задач, таких как классификация, распознавание образов и сегментация.

3. Уменьшение ошибок. Удаление шума снижает вероятность ложных срабатываний и ошибок, связанных с некорректным распознаванием объектов или неправильной классификацией изображений.

4. Сохранение деталей. Шумоподавление позволяет сохранить важные детали изображения, такие как границы объектов или текстура, что особенно важно в задачах, где сохранение этих элементов является ключевым.

5. Универсальность. Применение шумоподавления делает нейронные сети более универсальными, позволяя им работать с различными типами изображений и улучшать их качество независимо от условий съёмки или характеристик источника данных.

Требования к выполнению лабораторной работы

1. Ознакомиться с методическими указаниями, а также представленными примерами в Google.Collab.
2. Выполнить ход работы.
3. Выполнить индивидуальное задание, устранив шум, вернуть изображение к читабельному виду.
4. Протестировать полученные данные при помощи метрик.
5. Ответить на контрольные вопросы.

1 Ход работы

1.1 Добавление шума

Перед началом работы необходимо скачать набор данных Brain tumors 256×256 (A Refined Brain Tumor Image Dataset with Grayscale Normalization and Zoom). Он представляет собой архив МРТ исследований опухолей головного мозга.

После того как набор данных будет скачан, его необходимо загрузить в файловое пространство Collab. Содержимое файла можно считать с помощью библиотеки Pandas.

Пример функции для добавления шума:

```
def SaltAndPepper(data):
    for index, row in data.iterrows():
        file_path = row['file_path']
        class_name = row['Class']
        class_dir = f'/content/SaltAndPepper/{class_name}'
        if not os.path.exists(class_dir):
            os.mkdir(class_dir)
```

Функция SaltAndPepper принимает данные и проходит по каждой строке данных с помощью цикла for. В цикле для каждой строки определяются пути к файлу (*file_path*) и классу (*class_name*), а затем создаётся каталог для класса с использованием переменной *class_dir*. Каталог создается с помощью команды: os.mkdir.

Далее нужно загрузить изображение с помощью функции cv2.imread() и сгенерировать случайный шум с заданными параметрами. Сгенерированный шум преобразуется в массив типа uint8.

Зашумленное изображение сохраняется с помощью функции cv2.imwrite().

Далее вызывается функция, которая выполняет обработку данных (df) с использованием созданного алгоритма и создаётся переменная *df_*, которая

содержит обработанные данные (df) с помощью функции path_to_df("/content/x"). Сохраните архив с набором данных, поскольку он будет использоватьс в дальнейших лабораторных работах.

```
def path_to_df (dataset_path):
  data = []
  classes = os.listdir(dataset_path)
  for class_name in classes:
    class_path = os.path.join(dataset_path, class_name)
    if os.path.isdir(class_path):
      files = os.listdir(class_path)
      for file in files:
          file_path = os.path.join(class_path, file)
          data.append((file_path, class_name))
  df = pd.DataFrame(data, columns=['file_path', 'Class'])
  return (df)
```

1.2 Вывод результатов

Создайте датафрейм, в котором сохраняются пути к файлам.

```
original_image_path = df.loc[0, 'file_path']
```

Загрузка изображений происходит с помощью функции cv2.imread. Функция plt.show() отвечает за вывод визуализированных данных на экран. Вывод данных представлен на рисунке 1.1, на котором приведены четыре изображения. Они получаются из фреймов с зашумленными изображениями, созданными на прошлом шаге.

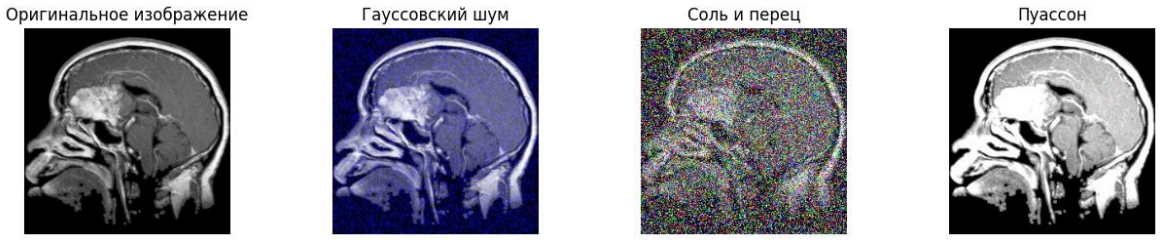

Рисунок 1.1 – Вывод результатов

1.3 Удаление шума с изображения

Далее рассмотрите способы борьбы с шумом на изображениях. В данном разделе представлены четыре метода:

- Median (Медианный фильтр);
- Gaussian (Размытие Гаусса);
- Linear (Линейное усреднение пикселей);
- Bilaterial (Билатериальный фильтр).

В представленном ниже коде функция выполняет фильтрацию изображений на основе данных из DataFrame. Она проходит по каждой строке, извлекая путь к файлу изображения и его класс (вид опухоли). Для каждого класса формируется путь к директории, где будут сохраняться обработанные изображения. Если такая директория еще не создана, она создается. Далее изображение считывается с помощью OpenCV, обрабатывается заданным фильтром, и результат сохраняется в соответствующую папку с исходным именем файла.

```
def (Тип фильтрации) (data):
    for index, row in data.iterrows():
        file_path = row['file_path']
        class_name = row['Class']
        class_dir = f'/content/(Тип фильтрации)/{class_name}'
        if not os.path.exists(class_dir):
            os.mkdir(class_dir)
        img_name = os.path.basename(file_path)
        path = f'{class_dir}/{img_name}'
        cv2.imwrite(path,  (Изображение))
    cv2.destroyAllWindows()
```

В зависимости от метода фильтрации изображения, будем меняться «ядро» функции. Так, для медианного фильтра функция фильтрации будет следующая:

```
median = cv2.medianBlur(img, 5)
```

Для размытия Гаусса

```
gaussian = cv2.GaussianBlur(img, (7, 7), 0)
```

Для линейного усреднения пикселей

```
kernel = np.array([[1/9, 1/9, 1/9],
                   [1/9, 1/9, 1/9],
                   [1/9, 1/9, 1/9]])
filt = cv2.filter2D(img, cv2.CV_8U, kernel)
```

Для билатериального

```
bilateral = cv2.bilateralFilter(img, 9, 75, 75)
```

После выполнения фильтрации, при помощи метода plt.show() выведите полученные после фильтрации изображения (рисунок 1.2).

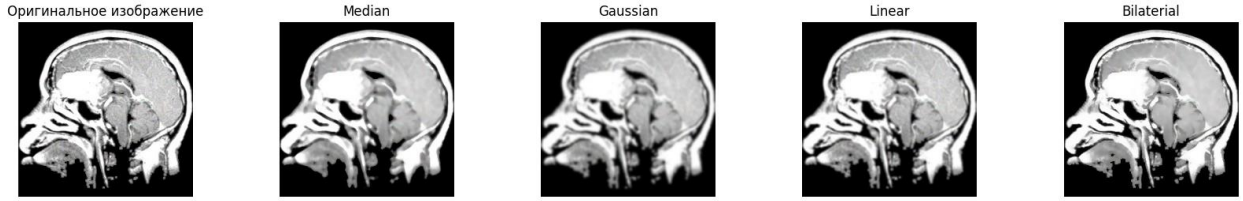

Рисунок 1.2 – Вывод отфильтрованных изображений

1.4 Использование метрик

Метрики оценивают качество работы модели.

MAE (средняя абсолютная ошибка) используется для регрессионных моделей и вычисляется как среднее арифметическое абсолютных ошибок. Формула:

$$MAE = \frac{1}{n}\sum_{j=1}^{n} y_j - \hat{y}_j \vee \qquad (1.1)$$

где *n* — количество точек в обучающей выборке,

yj — реальное значение,

yj — предсказанное значение.

MSE (среднеквадратическая ошибка) — это среднее арифметическое квадратов разностей между реальными и предсказанными значениями:

$$MSE = \frac{1}{n}\sum_{j=1}^{n}(\hat{y}_j - y_j)^2 \quad (1.2)$$

где *n* — количество точек в обучающей выборке,

yj — реальное значение,

yj — предсказанное значение.

MSE чувствительна к выбросам и подходит для нормальных распределений, а MAE менее чувствительна к выбросам, но может быть менее точной для данных с большим разбросом. Выбор метрики зависит от задачи и требований.

Функция calculate_metrics принимает два фрейма данных: *df_original* и *df_noisy*, содержащие изображения с оригинальными данными и зашумлённые данные соответственно.

```
df_orig_class = df_original[df_original['Class'] == cls].reset_index(drop=True)
df_noisy_class = df_noisy[df_noisy['Class'] == cls].reset_index(drop=True)
count = len(df_orig_class)
```

Далее следует вычисление метрики ошибки между двумя изображениями с использованием метода MSE и MAE.

```
img1 = cv2.imread(df_orig_class['file_path'].iloc[i])
img2 = cv2.imread(df_noisy_class['file_path'].iloc[i])
squared_diff = (img1 - img2) ** 2
summed = np.sum(squared_diff)
num_pix = img1.shape[0] * img1.shape[1]
err += summed / num_pix
mae_value += np.mean(np.abs(img1 - img2))
```

Функция вычисляет метрики для каждого класса, используя значения из фреймов данных. Затем результаты сохраняются в словаре results и выводятся на экран (рисунок 1.3).

```
Glioma_tumor: MSE = 200.67993363900138 MAE = 128.84230436098554
Meningioma_tumor: MSE = 195.23038951460137 MAE = 129.93341737979088
Normal: MSE = 190.46483723540285 MAE = 130.08176800685754
Pituitary_tumor: MSE = 202.4176988466091 MAE = 129.38266178631292
```

Рисунок 1.3 – Использование метрик

Эта метрика сравнивает две картинки: оригинальную и ту, которая была сначала зашумлена, а затем очищена от шума. На основе значения MSE и MAE можно сделать вывод о качестве удаления шума:

- Если MSE < 50, шум убран хорошо, изображения почти идентичны.
- Если 50 ≤ MSE < 150, шум убран удовлетворительно, видны небольшие отличия.
- Если MSE ≥ 150, шум убран плохо, различия между изображениями значительны.

Аналогично для MAE:

- Если MAE < 10, ошибки минимальны, результат отличный;
- Если 10 ≤ MAE < 20, ошибки заметны, но в пределах нормы;
- Если MAE ≥ 20, шум устранён недостаточно хорошо.

После выполнения лабораторной работы, обязательно сохраните полученный набор данных, он понадобится для последующих работ.

2 Индивидуальное задание

В таблице 2.1 представлены индивидуальные задания по вариантам, совпадающим положению в списке учебной группы. Нужно наложить шум, а после подобрать параметры и удалить его при помощи методов, представленных в таблице.

Таблица 2.1 – Индивидуальное задание

Вариант	Наложение шума	Удаление шума
1	Гауссовский шум	Линейное усреднение пикселей
2	Соль и перец	Медианная фильтрация
3	Пуассон	Гауссовское размытие
4	Гауссовский шум	Двусторонний фильтр
5	Соль и перец	Линейное усреднение пикселей
6	Пуассон	Медианная фильтрация
7	Гауссовский шум	Гауссовское размытие
8	Соль и перец	Двусторонний фильтр
9	Пуассон	Линейное усреднение пикселей

Контрольные вопросы

1. Какие методы сегментации изображений вы знаете?
2. Что такое пороговая сегментация и как она работает?
3. В чём разница между сегментацией экземпляров и семантической сегментацией?
4. Как сверточные нейронные сети используются для сегментации изображений?
5. Что такое U-Net и почему она используется для сегментации изображений?
6. Как оценить качество результатов сегментации изображений и выбрать оптимальный метод?
7. В чём разница между среднеквадратичной ошибкой (MSE) и средней абсолютной ошибкой (MAE)?

8. Какие задачи решает архитектура UNet при восстановлении повреждённых изображений?

ЛАБОРАТОРНАЯ РАБОТА №2

Балансировка набора данных методами Oversampling и Undersampling. Классификация полученного набора данных

Целью работы является анализ существующих методов балансировки классов, выбор наиболее подходящего для имеющегося набора данных, и применение его на практике. В ходе исследования будет выполнено обучение моделей на сбалансированных и несбалансированных данных для оценки их производительности. Также будет проведено сравнение полученных моделей с использованием ключевых метрик качества. На основе этого анализа будет сделан вывод о наиболее эффективном подходе к балансировке классов для данного типа данных.

Краткие теоретические сведения

Балансировка набора данных — это процесс изменения распределения классов в обучающих данных, чтобы избежать дисбаланса между количеством примеров каждого класса. Несбалансированность может привести к смещению модели в сторону доминирующего класса и недообучению на редких классах.

Методы балансировки набора данных включают Oversampling и Undersampling.

Oversampling увеличивает количество примеров в меньшинстве, добавляя новые синтетические примеры с помощью методов, таких как SMOTE (Synthetic Minority Over-sampling Technique).

Undersampling уменьшает количество примеров в большинстве, удаляя некоторые примеры случайным образом или выбирая подмножество из них.

После балансировки набора данных проводится классификация с использованием алгоритмов машинного обучения.

Классификация – это задача отнесения объекта по совокупности его характеристик к одному из заранее известных классов.

Как определяется задача классификации?

1. на вход модели классификации подается вектор признаков объекта;
2. на выходе модель классификации предсказывает одно из конечного набора значений – метку класса объекта;
3. зачастую сначала рассматривается бинарная классификация, остальные типы строятся на ее основе.

Бинарная классификация – это наличие или отсутствие какого-либо признака у объекта.

Логистическая регрессия бывает бинарной, когда есть только два возможных результата (да или нет) и многомерной. Пример многомерной логистической регрессии — прогнозирование вероятности смертельного исхода для пациента на основе различных показателей.

Дерево решений — это метод, который разделяет данные на классы с использованием древовидной структуры. Его можно применять, например, для анализа различных факторов, таких как возраст, пол, симптомы заболевания и результаты анализов. На каждом этапе алгоритма принимается решение о дальнейшем разделении пациентов на подгруппы или определении их принадлежности к определённой группе. Этот метод применяется в медицине для диагностики и лечения различных заболеваний.

Метод опорных векторов (SVM) — это способ классификации медицинских данных. Например, он может использоваться для разделения пациентов на группы с разными заболеваниями. Для этого SVM принимает в качестве входных данных информацию о пациентах и соответствующие диагнозы.

Наивный байесовский классификатор — это метод, основанный на теореме Байеса, который позволяет определить вероятность отнесения объекта к одному из нескольких классов на основе определённых признаков. Например, при

классификации пациентов на группы по медицинским данным, алгоритм анализирует вероятность принадлежности пациента к каждой группе и выбирает ту, у которой наибольшая вероятность.

K ближайших соседей (k Nearest Neighbors, или k-NN, где k — одно целое число, выбираемое заранее) — определяет класс новых данных на основе близости к k ближайшим соседям в обучающем наборе.

Требования к выполнению лабораторной работы

1. Ознакомиться с методическими указаниями, а также представленными примерами в Google.Colab.
2. Выполнить ход работы.
3. Выполнить индивидуальное задание, согласно варианту.
4. Ответить на контрольные вопросы.

1 Ход работы

1.1 Методы обработки данных

SMOTE (Synthetic Minority Over-sampling Technique) — это метод, который применяется для балансировки несбалансированных наборов данных в машинном обучении. SMOTE работает по принципу генерации синтетических примеров класса меньшинства, чтобы сбалансировать распределение классов и улучшить производительность модели на классе меньшинства.

ADASYN (Adaptive Synthetic Sampling) — это улучшенная версия метода SMOTE, которая учитывает сложность распределения классов и создаёт синтетические образцы на основе ближайших соседей каждого экземпляра класса меньшинства. В данном случае алгоритм ADASYN используется для балансировки классов обучающих данных, применяя стратегию выборки «меньшинства» и случайное состояние 42.

Tomek Links — это метод, который часто используется в сочетании с SMOTE для улучшения балансировки классов, удаляя пограничные случаи после увеличения выборки. Метод Tomek Links добавляет дополнительную проверку на корректность синтезированных примеров, чтобы избежать создания некорректных образцов.

В данной лабораторной работе проводится работа с набором данных из прошлой работы. Для этого нужно добавить в среду архив, полученный в первой лабораторной работе.

Далее происходит деление выборки на тренировочную и тестовую части. Это необходимо для оценки качества модели машинного обучения. Тренировочная модель используется для обучения модели, а тестовая – для проверки ее точности на новых данных. Это помогает избежать переобучения модели и получить более объективные результаты.

```
# Разделение данных на тренировочную и тестовую выборки
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
```

В коде выше происходит разделение выборки с использованием стратификации по классам. Функция train_test_split разделяет данные X на основе параметра test_size и случайного состояния. Параметр stratify указывает, что стратификация должна происходить по классам из массива y.

Далее проведите балансировку набора, после применения методов будут получены *X_smote, y_smote, X_adasyn, y_adasyn, X_tomek, y_tomek*.

```
smote = SMOTE(random_state=42)
X_smote, y_smote = smote.fit_resample(X_train, y_train)
adasyn = ADASYN(sampling_strategy='minority',random_state=42)
X_adasyn, y_adasyn = adasyn.fit_resample(X_train, y_train)
tomek = TomekLinks()
X_tomek, y_tomek = tomek.fit_resample(X_train, y_train)
```

1.2 Классификация на наборе данных

Классификация на несбалансированном наборе данных — это процесс, когда распределение наблюдений в целевом классе неравномерно. Одна метка класса содержит значительно большее количество наблюдений, в то время как другая имеет заметно меньшее количество. Это может привести к проблемам в машинном обучении, так как модели могут стать предвзятыми в своих прогнозах, отдавая предпочтение классу большинства.

Классификация на сбалансированном наборе данных — это процесс обучения модели машинного обучения на наборе данных, где количество выборок в каждом классе примерно одинаково. Это помогает избежать дисбаланса классов и повышает качество модели, так как она учится обобщать данные для всех возможных классов.

Обучение модели происходит при помощи случайного леса с использованием параметра случайного состояния. Модель обучается на данных *X_train* и соответствующих метках *y_train*. Затем модель предсказывает классы

для тестовых данных и сохраняет предсказанные значения в новой переменной *y_pred*.

```
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

Так же изменив, значения параметров в функции model.fit можно обучить модель на сбалансированном наборе данных.

```
model.fit(X_tomek, y_tomek)
```

Далее идет обучение модели на наборе данных. Полученные в результате обучения результаты представлены на рисунках 1.1 – 1.2.

Классификация на несбалансированном наборе данных:

	precision	recall	f1-score	support
glioma_tumor	0.83	0.68	0.75	180
meningioma_tumor	0.75	0.80	0.77	183
normal	0.86	0.82	0.84	88
pituitary_tumor	0.80	0.91	0.85	169
accuracy			0.80	620
macro avg	0.81	0.80	0.80	620
weighted avg	0.80	0.80	0.79	620

Рисунок 1.1 – Классификация на несбалансированном наборе

Классификация на наборе данных, сбалансированном методом Tomek Links:

	precision	recall	f1-score	support
glioma_tumor	0.84	0.68	0.75	180
meningioma_tumor	0.69	0.75	0.72	183
normal	0.82	0.76	0.79	88
pituitary_tumor	0.82	0.93	0.88	169
accuracy			0.78	620
macro avg	0.79	0.78	0.78	620
weighted avg	0.79	0.78	0.78	620

Рисунок 1.2 – Классификация на сбалансированном наборе

Полученные результаты можно так же отобразить, при помощи матрицы запутанности (рисунок 1.3).

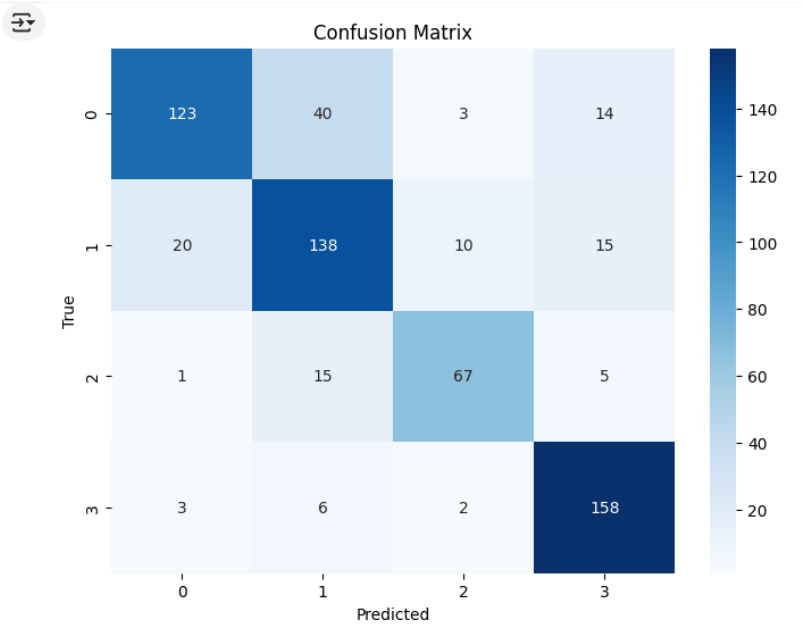

Рисунок 1.3 – Матрица запутанности

2 Индивидуальное задание

В таблице 2.1 представлены индивидуальные задания по вариантам, соответствующим порядковому номеру в списке учебной группы.

Таблица 2.1 – Индивидуальное задание

Вариант	Метод балансировки	Классификатор
1	SMOTE	RandomForestClassifier
2	Adasyn	GradientBoostingClassifier
3	Tomek Links	AdaBoostClassifier
4	SMOTE	SVC
5	Adasyn	LogisticRegression
6	Tomek Links	KNeighborsClassifier
7	SMOTE	GaussianNB
8	Adasyn	DecisionTreeClassifier

Требуется провести классификацию по варианту из таблицы на несбалансированном наборе данных, после этого сбалансировать набор методов по варианту и провести снова провести классификацию.

Контрольные вопросы

1. Какие методы используются для балансировки набора данных (oversampling и undersampling)?
2. В чём суть метода простого повторного использования данных (random oversampling)?
3. Как определяется желаемый баланс классов после применения random oversampling?
4. Как происходит случайное удаление примеров из большего класса для достижения желаемого баланса?
5. Что такое Tomek Links и как он используется для балансировки набора данных?
6. В каких случаях лучше использовать oversampling, а в каких undersampling?
7. Как проводится классификация полученного набора данных после балансировки?

ЛАБОРАТОРНАЯ РАБОТА №3

Разработка методов автоматической сегментации органов и тканей на медицинских изображениях.

Целью данной работы является исследование методов автоматической сегментации органов и тканей на медицинских изображениях. В процессе работы будут обучены несколько моделей, применены различные методы сегментации. Так же будет проведен анализ полученных результатов.

Краткие теоретические сведения

Сегментация изображений является ключевой задачей в области компьютерного зрения и обработки изображений. Она включает в себя разделение изображения на значимые регионы или объекты, что полезно для последующих этапов анализа, таких как идентификация или классификация. В этой лабораторной работе будут рассмотрены методы сегментации, такие как пороговая сегментация (thresholding), метод водораздела (watershed), рост области (region growing), а также алгоритмы, основанные на графах (graph based). Также будет использована современная модель для сегментации YOLOv8.

Пороговая сегментация представляет собой один из самых простых и быстрых методов сегментации изображений. Этот метод работает на основе установки одного или нескольких пороговых значений, чтобы разделить изображение на объекты и фон. В контексте бинаризации изображения, каждый пиксель сравнивается с пороговым значением, и, если его интенсивность превышает порог, он классифицируется как часть объекта; в противном случае - как фон. Для многоканальных изображений могут использоваться различные пороговые значения для каждого канала.

Метод водораздела представляет собой алгоритм сегментации, который использует концепцию «потоков» для разделения изображения на области. Этот

метод моделирует изображение как топографическую карту, где интенсивность пикселя соответствует высоте местности. Алгоритм «наполняет» низменные области, пока они не соединятся, образуя границы между различными регионами. Он эффективен для обработки изображений с пересекающимися объектами.

Метод роста областей — это метод сегментации изображений начинает с начальной точки (селектора) и постепенно добавляет соседние пиксели, которые удовлетворяют определённому критерию подобия (например, по интенсивности или цвету), расширяя область до тех пор, пока не будут охвачены все подходящие пиксели.

Алгоритмы на основе графов – это методы сегментации представляют изображение как граф, где пиксели являются вершинами, а рёбрами — связи между ними, основанные на схожести. Алгоритмы делят граф на подграфы, минимизируя определённую функцию стоимости, что позволяет эффективно выделять объекты на изображении.

YOLOv8 это одна из последних версий алгоритма YOLO (You Only Look Once) для детекции объектов в реальном времени. YOLOv8 использует улучшенные архитектуры нейронных сетей, что позволяет достичь высокой точности и скорости детекции при сохранении компактности модели.

Требования к выполнению лабораторной работы

1. Ознакомиться с методическими указаниями, а также представленными примерами в Google.Colab.
2. Выполнить ход работы.
3. Выполнить индивидуальное задание, проведя сегментацию 10 изображений из набора данных при помощи Thresholding, Watershed, Region Growing, Graph-Based.

4. Разметить исходный набор данных, провести аугментацию, провести сегментацию набора при помощи YOLOv8.

5. Ответить на контрольные вопросы.

1 Ход работы

1.1 Thresholding

Перед началом работы необходимо скачать набор данных Brain tumors 256x256 (A Refined Brain Tumor Image Dataset with Grayscale Normalization and Zoom). Он представляет собой архив МРТ исследований опухолей головного мозга.

После того как набор данных будет скачан, необходимо загрузить из него в файловое пространство 10 фотографий. Считайте изображение при помощи библиотеки OpenCV. Выведенное изображение представлено на рисунке 1.1.

```
def imshow(img, ax=None):
    if ax is None:
        ret, encoded = cv2.imencode(".jpg", img)
        display(Image(encoded))
    else:
        ax.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
        ax.axis('off')
```

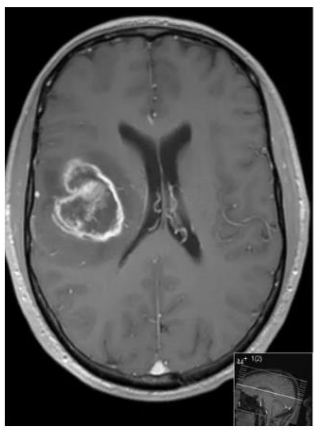

Рисунок 1.1 – Вывод изображения

Далее следует конвертирование изображение в оттенки серого, используя метод cv2.cvtColor. Изображение в оттенках серого хранится в переменной *gray*. Преобразование изображения в градации серого перед пороговой обработкой и алгоритмом водораздела упрощает анализ, так как эти методы работают с интенсивностью пикселей, а не с цветом. Это делает выделение объектов и их сегментацию более эффективными и точными.

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

После проводится сегментация при помощи трех типов Thresholding: Обычный, Адаптивный, Мультиклассовый. Для этого возьмите серое изображение, и наложите на них фильтры.

```
ret, bin_img = cv2.threshold(gray,0, 255,cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)
plt.figure(figsize=(10, 5))
plt.imshow(bin_img, cmap='gray')
adaptive_thresh= cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
cv2.THRESH_BINARY_INV, 11, 2)
multiclass_thresh = np.zeros_like(gray)
thresh1, bin_img1 = cv2.threshold(gray, 85, 255, cv2.THRESH_BINARY)
thresh2, bin_img2 = cv2.threshold(gray, 170, 255, cv2.THRESH_BINARY)
multiclass_thresh[(gray > 0) & (gray <= 85)] = 85
multiclass_thresh[(gray > 85) & (gray <= 170)] = 170
multiclass_thresh[(gray > 170)] = 255
```

Для вывода полученных изображений создайте сетку из четырех полей, на которые будут помещены изображения. Функция plt.show() отвечает за вывод визуализированных данных на экран. Выведенные изображения представлены на рисунке 1.2.

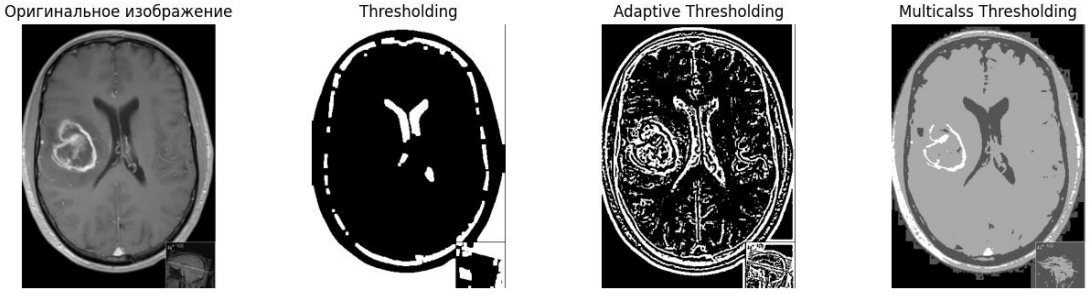

Рисунок 1.2 – Вывод результатов

1.2 Watershed

Для использования данного метода нужно захватить черную область, которая является фоновой частью этого изображения. После Threshold основная часть изображения будет выделена в белых тона, а значит — это необходимая область и она хорошо заполнена, в то время как темная часть — это фон. Для этого будет применено несколько морфологических операций к бинарному изображению:

– Первая операция — это расширение с использованием cv2.dilate\, которое расширяет яркие области изображения, создавая переменную sure_bg, представляющую определенную область фона. Этот результат отображается с помощью функции imshow.

– Следующая операция — cv2.distanceTransform, которая вычисляет расстояние каждого белого пикселя в бинарном изображении до ближайшего черного пикселя. Результат сохраняется в переменной *dist* и отображается с помощью введенной ранее функции imshow.

– Затем область переднего плана получается путем применения порога к переменной *dist* с использованием cv2.threshold. Порог устанавливается в 0,5 раза больше максимального значения *dist*.

– Наконец, неизвестная область рассчитывается как разница между определенным фоном и определенными областями переднего плана с использованием cv2.subtract. Результат сохраняется в переменной *unknown* и отображается с помощью imshow.

```
bin_img = cv2.morphologyEx(bin_img, cv2.MORPH_OPEN, kernel, iterations=6)
sure_bg = cv2.dilate(bin_img, kernel, iterations=3)
dist = cv2.distanceTransform(bin_img, cv2.DIST_L2, 5)
ret, sure_fg = cv2.threshold(dist, 0.5 * dist.max(), 255, cv2.THRESH_BINARY)
sure_fg = sure_fg.astype(np.uint8)
unknown = cv2.subtract(sure_bg, sure_fg)
```

Следующим шагом идет создание изображения-маркера, созданного путем маркировки определенного переднего плана и неизвестной области, служит входными данными для алгоритма водораздела. Он управляет алгоритмом сегментации изображения на основе этих помеченных областей. Каждый отдельный цвет или метка представляет собой отдельный сегмент или область изображения.

Функция cv2.watershed применяется к исходному изображению *img* и изображению маркеров, полученному на предыдущем шаге, для выполнения алгоритма Watershed. Результат сохраняется в маркерах, изображение «маркеров» отображается с использованием метода imshow Matplotlib с цветовой картой tab20b. Цикл перебирает метки, начиная со второй (игнорируя фон и неизвестные области), чтобы извлечь контуры каждого объекта. Контуры бинарного изображения находятся с помощью функции cv2.findContours. Наконец, контуры объектов рисуются на исходном изображении с помощью cv2.drawContours. Результат отображается с помощью функции imshow.

```
markers = cv2.watershed(img, markers)
fig, ax = plt.subplots(figsize=(5, 5))
labels = np.unique(markers)
test = []
for label in labels[2:]:
   target = np.where(markers == label, 255, 0).astype(np.uint8)
   contours, hierarchy = cv2.findContours(
      target, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE
   )
test.append(contours[0])
```

1.3 GraphBased и Region Growing

Для того чтобы провести сегментацию методом на основе графов нужно клонировать файлы из git-репозитория.

```
!git clone https://github.com/Spinkoo/Region-Growing.git
```

```
%cd Region-Growing
!pip install -r requirements.txt
```

Далее заменив функцию для отображения, нужно будет открыть код и поменять в нем функцию вывода итогового изображения.

```
def display(self, output_path='output.png'):
    rgb_image = cv2.cvtColor(self.SEGS, cv2.COLOR_BGR2RGB)
    cv2.imwrite(output_path, self.SEGS)
```

Проведите сегментацию при помощи следующего кода.

```
!python RegionGrowing.py /content/mri.jpg 15
```

В результате чего должно получиться следующее изображение (рисунок 1.3).

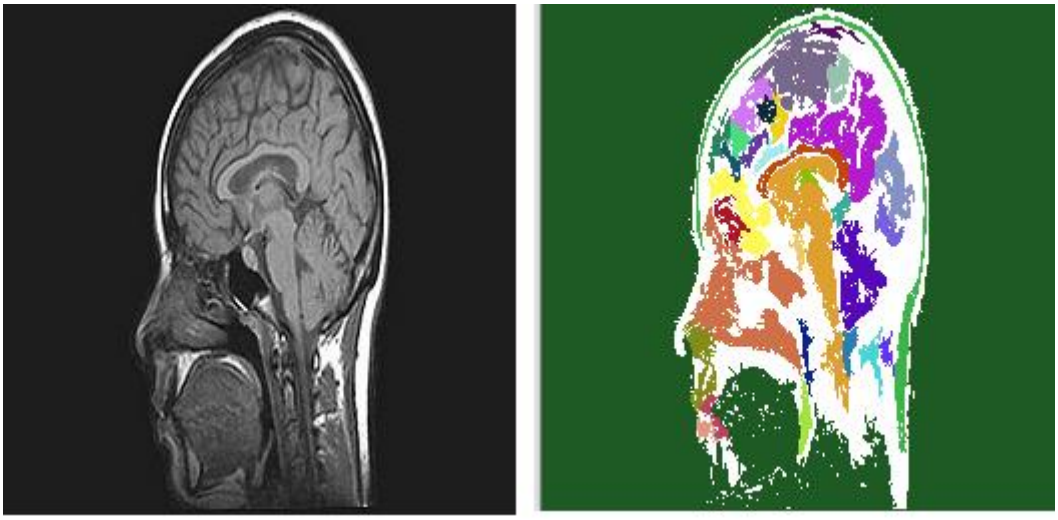

Рисунок 1.3 – Результат Region-Growing

Скопируем следующий репозиторий, а также проведем сегментацию.

```
!git clone https://github.com/sirius-mhlee/graph-based-image-segmentation
!python ImageSegmentation.py 0.5 500 50 /content/mri.jpg /content/result_mri.jpg
```

После выполнения сегментации должен получиться следующий результат (рисунок 1.4).

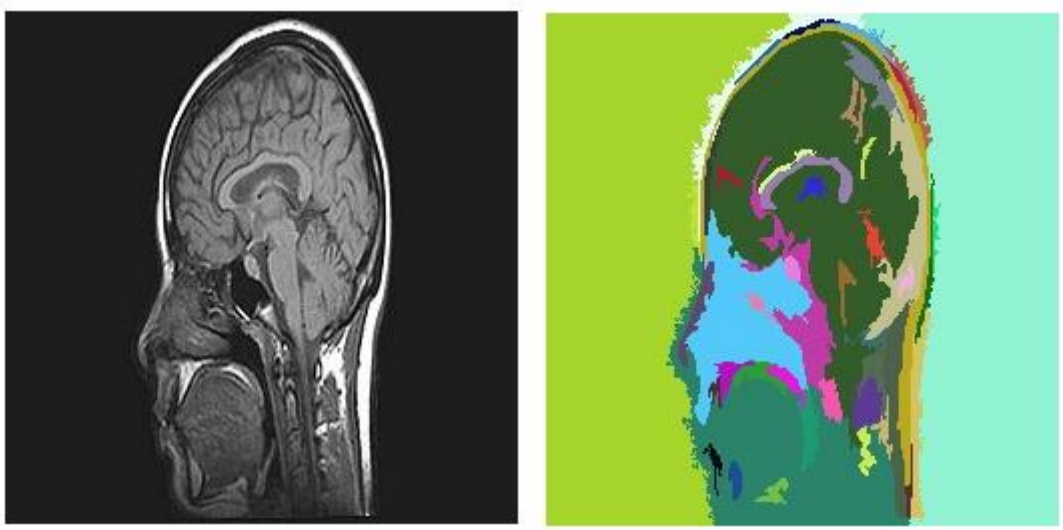

Рисунок 1.4 – Результат Graph-Based

1.4 YOLO и Roboflow

Перед началом работы с моделью YOLOv8 необходимо подготовить соответствующий набор данных. Для этого следует зайти на платформу Roboflow и создать новый проект, выбрав тип Instance Segmentation (рисунок 1.5). Это позволит организовать данные для сегментации объектов.

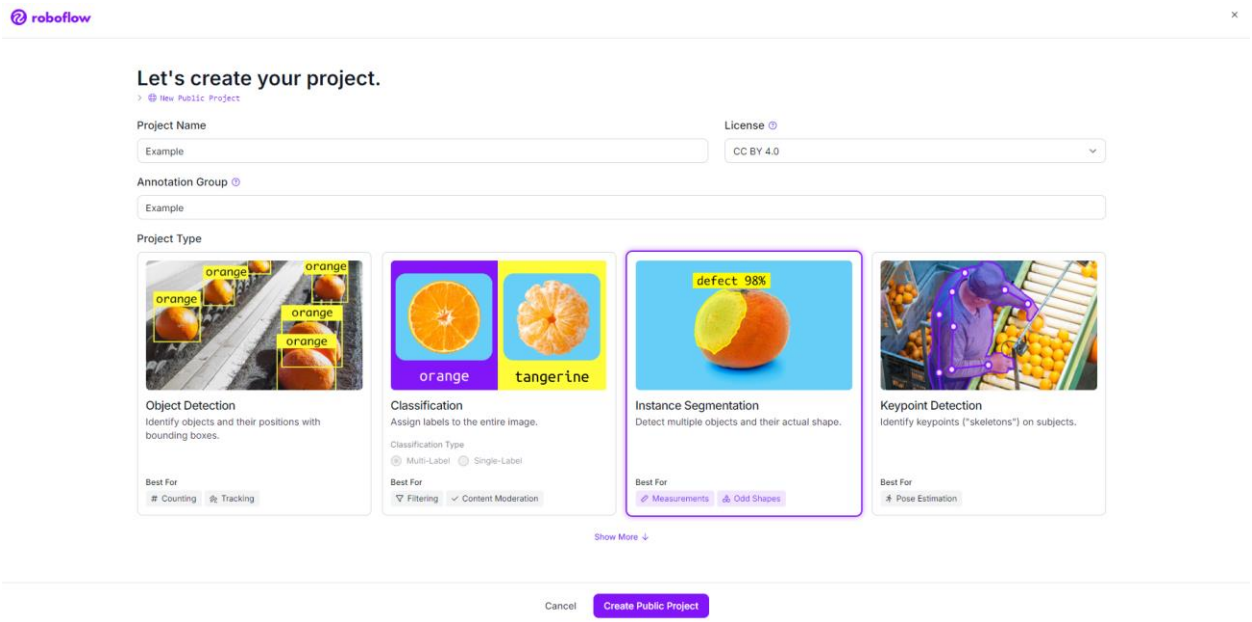

Рисунок 1.5 – Создание проекта на Roboflow

После создания проекта, система предложит начать аннотирование. Для этого потребуется загрузить изображения и назначить ответственного за проект (см. рисунок 1.6). Правильное аннотирование — ключевой шаг, так как оно обеспечивает качественную разметку данных, необходимую для обучения модели.

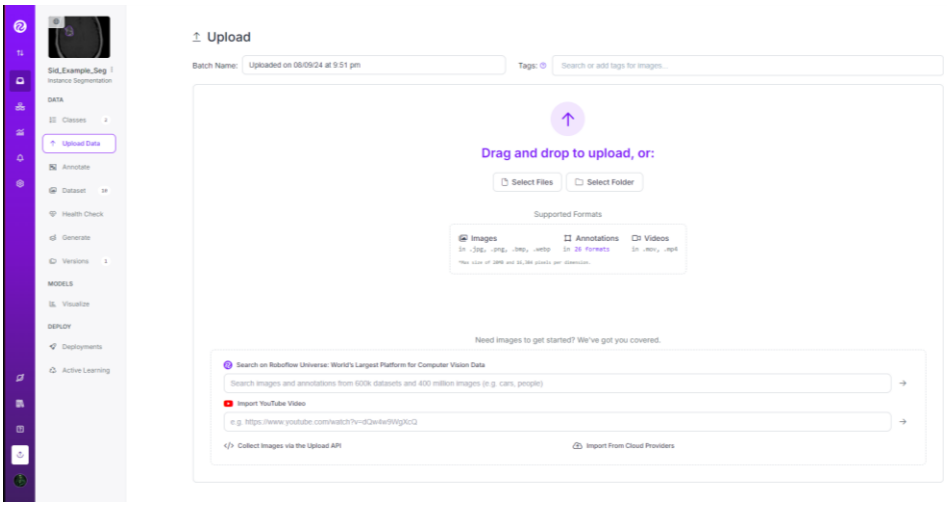

Рисунок 1.6 – Загрузка

В процессе аннотирования необходимо выделить фрагменты изображений и подписать их соответствующими метками (см. рисунок 1.7). Точное аннотирование объектов обеспечит высокую точность модели в будущем.

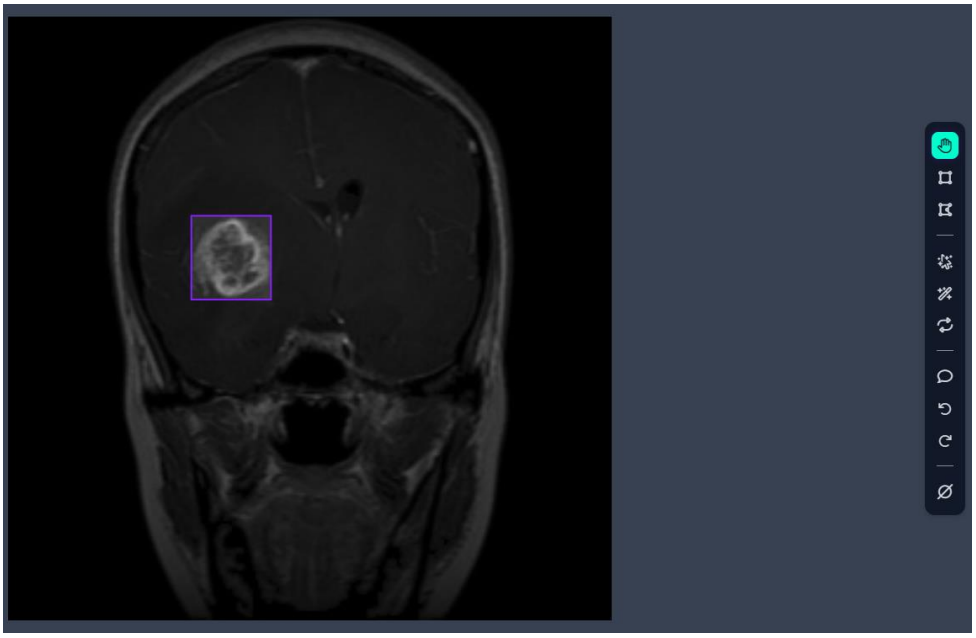

Рисунок 1.7 – Аннотация

После завершения аннотирования следует сгенерировать новую версию набора данных, применив к нему методы аугментации и предобработки (см. рисунок 1.8). Аугментация данных поможет улучшить обобщающую способность модели за счет увеличения разнообразия обучающих примеров.

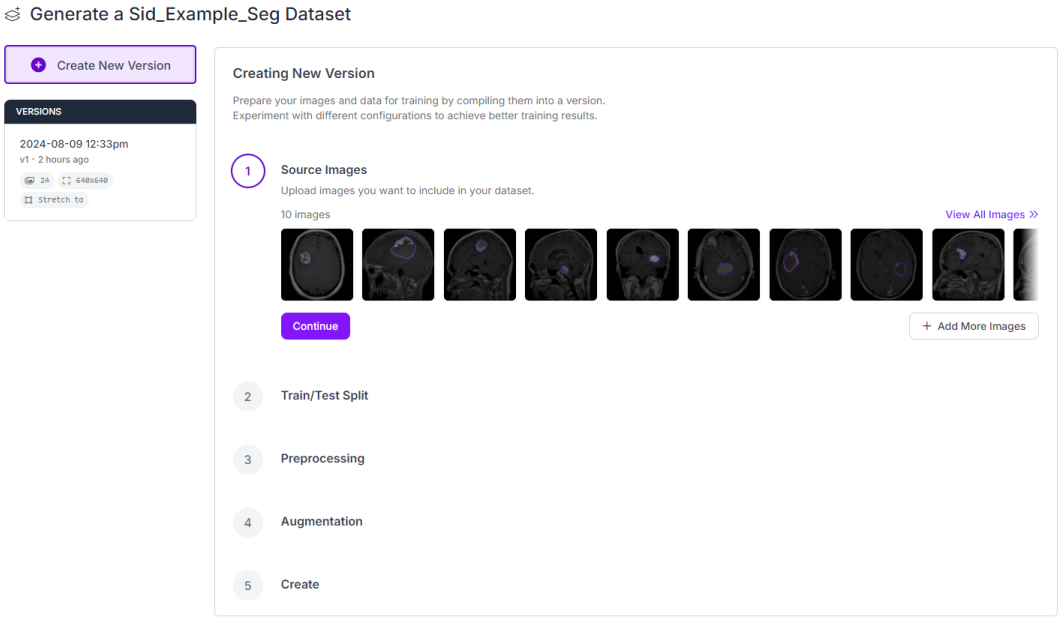

Рисунок 1.8 – Генерация набора данных

Затем необходимо экспортировать полученный набор данных в нужном формате, в данном случае — YOLOv8 TXT (см. рисунок 1.9). Это обеспечит совместимость вашего набора данных с архитектурой YOLOv8.

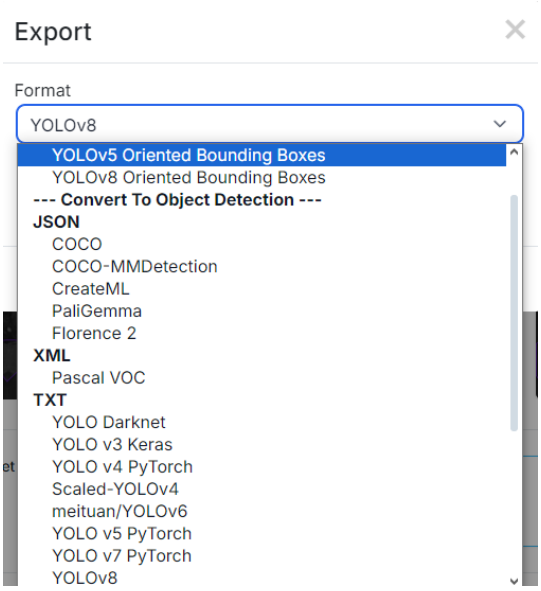

Рисунок 1.9 – YOLOv8

После разархивирования данных, вставьте путь к файлу data.yaml и запустите обучение модели, используя следующий код (см. рисунок 1.10):

```
results = model.train(data="/content/dataset/data.yaml", epochs=100, imgsz=640)
```

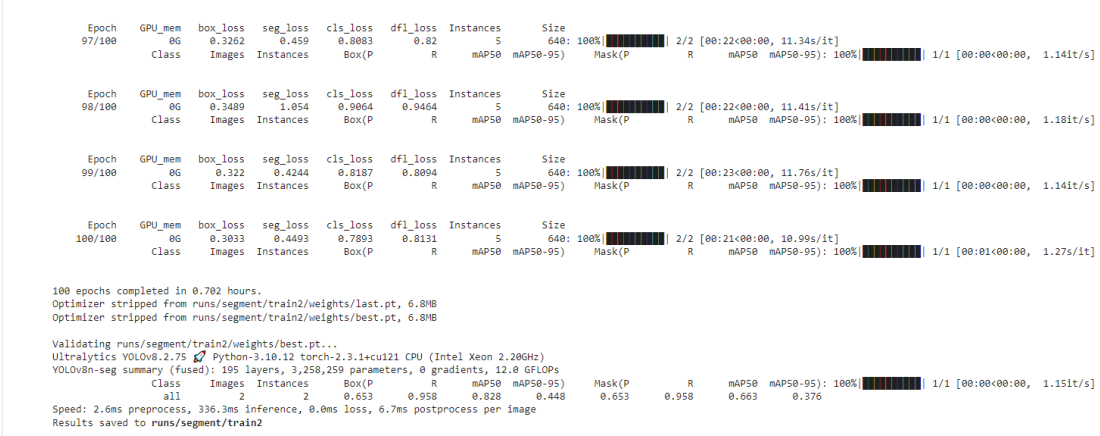

Рисунок 1.10 – Обучение

По завершении обучения визуализируйте результаты и получите тестовую выборку с сегментированными фрагментами. Используйте следующий код для получения изображений с сегментацией (рисунок 1.11).

```
result = results[0]
rendered_image = result.plot()
```

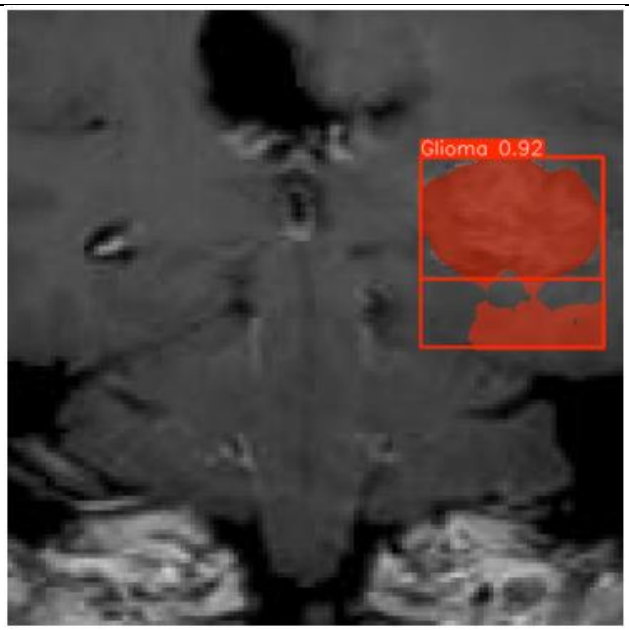

Рисунок 1.11 – Вывод сегментированного YOLOv8 изображения

2 Индивидуальное задание

1. На основе изученного материала, провести сегментацию полученного набора данных при помощи YOLOv8;
2. Провести сегментацию 10 изображений по одному из методов по варианту из таблицы 2.1.
3. Сравнить используемые методы сегментации, включая различные модели;
4. Дать развернутый вывод по эффективности моделей для сегментации и возможности их применения для работы с медицинскими изображениями.

Таблица 2.1 – Индивидуальное задание

Вариант	Метод
1	Thresholding (Пороговая сегментация)
2	Watershed (Метод водораздела)
3	Region Growing (Рост области)
4	Graph-Based (Алгоритмы на основе графов)

Контрольные вопросы

1. Что такое сегментация изображений и зачем она нужна?
2. Какие методы сегментации используются в обработке медицинских изображений?
3. Объясните принцип работы метода thresholding (пороговой сегмент ации).
4. В чём суть метода watershed (метода водораздела)?
5. Как работает метод region growing (рост области)?
6. Расскажите об алгоритмах, основанных на графах (graph based al-

gorithms).

7. В чем суть работы YOLO? Особенности, преимущества, недостатки.

ЛАБОРАТОРНАЯ РАБОТА №4

Исследование методов детектирования опухолей и других аномалий на медицинских изображения с использованием ИИ

Целью данной работы является исследование и сравнение различных методов детектирования аномалий на медицинских изображениях с применением современных моделей. В рамках работы планируется рассмотреть различные подходы детекции, включая архитектуры, такие как YOLO и EfficientNet, чтобы определить их эффективность для точного выявления аномалий.

Краткие теоретические сведения

Детектирование объектов на изображении — это автоматический процесс определения и локализации различных объектов или регионов интереса (ROI) на цифровых изображениях. Эта задача актуальна в компьютерном зрении и машинном обучении и применяется в распознавании лиц, медицинской диагностике, видеонаблюдении и других областях.

Задачи детектирования включают:

- Классификацию: определение принадлежности объекта к определенному классу.
- Локализацию: определение положения объекта с помощью ограничивающей рамки.
- Детекцию: нахождение координат нескольких объектов на изображении.
- Сегментацию: предсказание категории для каждого пикселя изображения.

Методы детектирования делятся на:

- Двухэтапные: сначала генерируют гипотезы, которые затем классифицируются.
- Одноэтапные: сразу формируют прямоугольники для объектов при прохождении через нейронную сеть.

EfficientNet — это архитектура свёрточной нейронной сети, использующая метод составного масштабирования для сбалансированного увеличения глубины, ширины и разрешения модели. Ключевая идея — масштабировать эти параметры пропорционально, улучшая точность без значительного увеличения вычислительных затрат.

Формулы для масштабирования:

$$w = \beta^{\varphi} \quad (1)$$

$$d = \alpha^{\varphi} \quad (2)$$

$$r = \gamma^{\varphi} \quad (3)$$

Преимущества EfficientNetv2:

- улучшенный NAS (Neural Architecture Search) с новыми блоками и идеями;
- новый подход к прогрессивному обучению, который регулирует регуляризацию в зависимости от размера изображения;
- достижение лучших результатов в сравнении с другими моделями при обучении и работе с меньшими размерами изображений и меньшим количеством параметров;
- увеличение скорости обучения до 11 раз и уменьшение размеров моделей до 6,8 раз.

Detectron — это инструмент на базе PyTorch для детектирования объектов с использованием готовых моделей и алгоритмов, таких как Faster R-CNN и Mask R-CNN. Он поддерживает форматы данных COCO и Pascal VOC.

PyTorch — это фреймворк для машинного обучения на Python, используемый для решения задач компьютерного зрения и обработки языка. Он предоставляет средства для тензорных вычислений и создания глубоких нейронных сетей.

YOLO9 и YOLO10 — это версии алгоритма YOLO, разработанного для обнаружения объектов в режиме реального времени.

YOLO9:

- YOLO 9 использует архитектуру Faster R-CNN, которая состоит из двух частей: Region Proposal Network (RPN) и Region of Interest Classifier (RoI). RPN отвечает за генерацию предложений регионов, а RoI Classifier классифицирует эти регионы на наличие объекта и его класс;
- YOLO9 использует свёрточные слои для извлечения признаков из изображений и полносвязные слои для классификации объектов и предсказания их координат.
- YOLO10 представляет собой улучшенную версию YOLO9:
- использование более глубоких свёрточных слоёв для извлечения признаков из изображений;
- применение метода Anchor Boxes для более точного предсказания координат объектов;
- улучшенная функция потерь для обучения модели;
- использование ResNet в качестве базовой архитектуры для извлечения признаков.

Требования к выполнению лабораторной работы

1. Ознакомиться с методическими указаниями, а также представленными примерами в Google.Colab.
2. Выполнить ход работы.
3. Выполнить индивидуальное задание, согласно варианту.
4. Ответить на контрольные вопросы.

1 Ход работы

1.1 EfficientNetv2

Первый шаг – клонировать репозиторий с GitHub в текущую рабочую директорию. Для вывода содержимого файла на экран используется команда cat, команда xargs разделяет вывод на отдельные строки.

```
cd Monk_Object_Detection/3_mxrcnn/installation && cat requirements_colab.txt | xargs -n 1 -L 1 pip install
```

Далее нужно установить пакет roboflow с помощью менеджера пакетов pip, импортировать библиотеку roboflow и создать экземляр класса с ключом API. Затем подключиться к рабочей области, выбрать нужные пакеты и загрузить набор данных.

```
from roboflow import Roboflow
rf = Roboflow(api_key="your-api-key ")
project = rf.workspace("your-workspace ").project("your-project")
version = project.version(1)
dataset = version.download("coco")
```

В следующей части кода данные перемещаются в структуру, которую ожидает библиотека обнаружения изображений, однако манипуляция данными файлов не требуется.

Далее происходит импорт класса Detector из модуля *train_detector*. Создайте экземпляр класса Detector и устанавливаются гиперпараметры для детектора. Гиперпараметры включают коэффициент скорости обучения (*lr*), интервал проверки (*val_interval*), ранняя остановка (*es_min_delta*), ранняя остановка с параметром patience (*es_patience*).

```
gtf.Set_Hyperparams(lr=0.0001, val_interval=10, es_min_delta=0.0, es_patience=0)
```

Для начала тестирования нужно сделать следующее:

1. создать список *test_images* из файлов в директории *test*, которые имеют расширение «*.jpg*»;
2. с помощью функции из библиотеки *random* выбрать случайный файл из созданного списка;
3. произвести предсказание класса объекта с использованием модели *gtf* и пути к изображению *image_path*.

```
test_images = [f for f in os.listdir('test') if f.endswith('.jpg')]
import random
img_path = os.path.join('test', random.choice(test_images))
result = gtf.Predict(img_path, class_list)
```

1.2 Detectron

Для начала работы нужно импортировать модуль locale и определить функцию, возвращающую значение «UTF-8».

```
import locale
def getpreferredencoding(do_setlocale = True):
    return "UTF-8"
locale.getpreferredencoding = getpreferredencoding
```

Нужно установить пакет pyyaml, torch и torchvision. Torch и TorchVision — это библиотеки для машинного обучения и компьютерного зрения. Torch используется для создания и обучения нейронных сетей, а TorchVision содержит готовые модели и инструменты для работы с изображениями.

Установим библиотеку Detectron2.

```
!pip install -U git+https://github.com/facebookresearch/detectron2.git
```

После установки библиотеки Detectron2 нужно импортировать из нее некоторые общие библиотеки и утилиты. Функция setup_logger() инициализирует логгер Detectron2, а затем импортируются остальные библиотеки и функции. Далее происходит регистрация пользовательских данных в библиотеке. Регистрируются три набора данных.

Далее происходит визуализация обучающих данных. Загружаются метаданные и список объектов из обучающего набора данных (рисунок 1.1).

Рисунок 1.1 – Визуализация тестовых данных

Чтобы настроить обучение модели для обнаружения объектов, нужно:

- Импортировать функцию get_cfg() из Detectron2;
- Загрузить готовую конфигурацию модели из model_zoo и изменить настройки;
- Указать наборы данных для обучения и тестирования;

- Настроить параметры: скорость обучения, число итераций, размер пакета, количество классов и другие;
- Создать класс CocoTrainer, который будет использовать метод оценки результатов (COCOEvaluator) для проверки качества модели;
- Создать папки для сохранения результатов;
- Запустить обучение с помощью созданного тренера CocoTrainer.

```
@classmethod
  def build_evaluator(cls, cfg, dataset_name, output_folder=None):
    if output_folder is None:
        os.makedirs("coco_eval", exist_ok=True)
        output_folder = "coco_eval"
```

Далее происходит оценка модели с использованием набора данных для тестирования. Сначала задаются параметры модели: веса модели и порог оценки для тестовых данных. Затем создаётся предсказатель с заданными параметрами модели. Далее создаётся оценщик COCOEvaluator с указанием имени набора данных, параметров модели и выходного каталога. Создаётся загрузчик для тестовых данных с использованием заданного набора данных и параметров модели. Наконец, выполняется выполнение работы нейронной сети после её обучения на загруженных данных с использованием созданной модели и загрузчика.

Следующий фрагмент кода устанавливает параметры модели в файле конфигурации (*cfg*), задаёт набор данных для тестирования, устанавливает порог оценки для тестовых данных и создаёт предсказатель. Затем импортируются функции для визуализации результатов и импорта файлов изображений.

```
cfg.MODEL.WEIGHTS = os.path.join(cfg.OUTPUT_DIR, "model_final.pth")
cfg.DATASETS.TEST = ("my_dataset_test", )
cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.7
predictor = DefaultPredictor(cfg)
test_metadata = MetadataCatalog.get("my_dataset_test")
```

Следующий код считывает изображение с заданным именем и путём, преобразует его с помощью OpenCV и отображает результат с помощью другой библиотеки.

```
imageName= '/content/…'
im = cv2.imread(imageName)
outputs = predictor(im)
v = Visualizer(im[:, :, ::-1],
            metadata=test_metadata,
            scale=0.8
              )
out = v.draw_instance_predictions(outputs["instances"].to("cpu"))
cv2_imshow(out.get_image()[:, :, ::-1])
```

После выполнения следует вывод результатов.

1.3 YOLO

Для начала работы с YOLO нужно установить пакет ultralytics. Ultralytics — это библиотека для компьютерного зрения. Из него нужно импортировать YOLOv9.

```
model = YOLO("yolov9c.yaml")
model = YOLO("yolov9c.pt")
```

Модель обучается на примере набора данных coco8 в течение 100 эпох с размером изображения 256 пикселей и сохранением результатов в соответствующей переменной.

Далее происходит считывание изображение с помощью библиотеки OpenCV и использование модели для предсказания класса объекта на изображении. После этого нужно импортировать библиотеку Matplotlib и отобразить результат.

Для работы с YOLOv10, ее также нужно установить из пакета ultralytics. Выполните обучение модели в течение 100 эпох с размером изображения 640 пикселей.

```
model.train(data="data.yaml", epochs=100, imgsz=640)
```

Далее нужно загрузить изображение для детектирования с помощью следующего кода:

```
image_path = "/content/… "
image = cv2.imread(image_path)
results = model.predict(image, conf=0.5)
```

Для визуализации и вывода результата используется такой же код, как в YOLOv9.

2 Индивидуальное задание

В таблице 2.1 представлены индивидуальные задания. Вариант соответствует порядковому номеру в группе. Требуется провести детектирование набора данных по варианту.

Таблица 2.1 – Индивидуальное задание

№	Набор данных
1	https://www.kaggle.com/datasets/navoneel/brain-mri-images-for-brain-tumor-detection
2	https://www.kaggle.com/datasets/thomasdubail/brain-tumors-256x256
3	https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia
4	https://www.kaggle.com/datasets/abdallahwagih/retina-blood-vessel
5	https://www.kaggle.com/datasets/adhoppin/blood-cell-detection-datatset

Контрольные вопросы

1. Как осуществляется детектирование и выделение границ объектов на медицинских изображениях?

2. Какие методы детектирования объектов применяются в медицине?

3. В чём разница между классификацией и детектированием объектов на медицинских изображениях?

4. Какие задачи решает детектирование объектов на медицинских изображениях?

5. Какие технологии и алгоритмы используются для детектирования объектов на медицинских изображениях?

6. Какие проблемы и ограничения существуют при использовании детектирования объектов на медицинских изображениях?

7. Как происходит процесс обучения нейронных сетей для детектирования объектов на медицинских изображениях?

ЛАБОРАТОРНАЯ РАБОТА №5

Исследование проблем совместной обработки и анализа данных различных модальностей для более точной диагностики и лечения.

Целью данной работы является изучение и анализ данных различных модальностей медицинской визуализации, таких как МРТ, КТ и УЗИ, с целью повышения точности диагностики и выбора оптимальных методов лечения, а также применению нейронных сетей и методов машинного обучения для интеграции и совместной обработки данных разных модальностей. Работа предполагает исследование эффективности совместной обработки данных.

Краткие теоретические сведения

МРТ (магнитно-резонансная томография) — это метод исследования, основанный на взаимодействии магнитного поля и радиоволн с атомами водорода в теле человека. Он создаёт трёхмерные изображения внутренних структур организма, позволяя увидеть мягкие ткани, сосуды и нервы.

КТ (компьютерная томография) использует рентгеновские лучи для создания поперечных срезов тела, создавая таким образом двухмерные изображения. Этот метод хорошо подходит для исследования костных структур и плотных образований.

УЗИ (ультразвуковое исследование) использует высокочастотные звуковые волны для визуализации внутренних органов и мягких тканей. Этот метод широко применяется в акушерстве, гинекологии и педиатрии, так как он безопасен и не требует использования ионизирующего излучения.

После обучения нейросеть способна самостоятельно анализировать новые данные и предоставлять врачу информацию о состоянии здоровья пациента. Это помогает повысить точность диагностики и выбрать оптимальный метод лечения.

Совместная обработка данных — это процесс, когда несколько нейронных сетей работают вместе для решения сложных задач. Они обмениваются информацией и результатами своей работы, чтобы улучшить качество и точность результатов.

Совместная обработка данных используется для повышения эффективности и точности работы алгоритмов машинного обучения. Она позволяет:

- улучшить обобщение паттернов, так как сети могут учиться на разных наборах данных;
- снизить зависимость от данных, так как сети могут использовать информацию из разных источников;
- увеличить скорость обработки данных, так как сети могут выполнять параллельные вычисления;
- улучшить интерпретацию результатов, так как сети могут предоставлять более понятные объяснения своих выводов.

Требования к выполнению лабораторной работы

1. Ознакомиться с методическими указаниями, а также представленными примерами в Google Colab.
2. Выполнить ход работы.
3. Выполнить индивидуальное задание.
4. Ответить на контрольные вопросы.

1 Ход работы

1.1 Подготовка набора данных

Для начала работы нужно импортировать библиотеки для обработки изображений, анализа данных и машинного обучения. Основные функции и операции, которые выполняются:

- импорт библиотек;
- разделение данных на обучающую и тестовую выборки;
- обучение модели с использованием библиотеки sklearn.ensemble и класса RandomForestClassifier: rfc = RandomForestClassifier(...);
- оценка качества модели с помощью функции classification_report из библиотеки sklearn.metrics: print(classification_report(y_true, y_pred)).

Далее требуется импортировать необходимые модули и библиотеки (tensorflow.keras, sklearn.preprocessing, sklearn.model_selection и tensorflow.keras.utils). Определить функции для кодирования меток и преобразования данных в категориальный формат. Разделить данные на обучающую и тестовую выборки с использованием метода train_test_split из библиотеки sklearn.model_selection, а также установить необходимые пакеты с помощью команды pip install.

После этого нужно загрузить наборы данных «Augmented Alzheimer MRI Dataset», «PCOS setection using ultrasound images» и «COVID-CT» с сайта Kaggle с использованием библиотеки opendatasets.

Далее следует создание функции path_to_df, которая принимает путь к набору данных в качестве аргумента. Функция инициализирует пустой список данных и получает список классов. Затем перебирает классы в цикле for и создает подкаталоги для каждого класса. Внутри каждого подкаталога функция находит файлы с расширением .jpg, .jpeg или .png и сохраняет их пути в списке data. В

конце функция преобразует данные в DataFrame с двумя столбцами ‘file_path’ и ‘Class’ и возвращает его.

```
def path_to_df (dataset_path):
  data = []
  classes = os.listdir(dataset_path)
  for class_name in classes:
    class_path = os.path.join(dataset_path, class_name)
    if os.path.isdir(class_path):
      files = os.listdir(class_path)
      for file in files:
        if file.endswith('.jpg') or file.endswith('.jpeg') or file.endswith('.png'):
          file_path = os.path.join(class_path, file)
          data.append((file_path, class_name))
  df = pd.DataFrame(data, columns=['file_path', 'Class'])
  return (df)
```

Следующим шагом происходит импорт набора данных МРТ исследования болезни Альцгеймера из каталога «augmented-alzheimer-mri-dataset» в папку «OriginalDataset», импорт набора данных ультразвуковых изображений для обучения модели обнаружения поликистоза яичников (PCOS), импорт набора данных COVID-CT. Каждый набор данных сохраняется в соответствующем переменной, результат выводится на экран.

Затем происходит выборка данных из трёх созданных датафреймов в заданных пропорциях. После этого данные объединяются в один датафрейм, и производится перемешивание и выборка данных с использованием метода sample() и случайного состояния (random_state). В конце данные снова объединяются и перемешиваются.

```
proportion1 = 0.5
proportion2 = 0.2
proportion3 = 0.3
total_size = min(len(df_mri), len(df_ultrasound), len(df_ct))
n1 = int(total_size * proportion1)
n2 = int(total_size * proportion2)
```

```
n3 = int(total_size * proportion3)
df1_sampled = df_mri.sample(n=n1, random_state=42)
df2_sampled = df_ultrasound.sample(n=n2, random_state=42)
df3_sampled = df_ct.sample(n=n3, random_state=42)
df_combined = pd.concat([df1_sampled, df2_sampled, df3_sampled]).reset_index(drop=True)
df_combined = df_combined.sample(frac=1, random_state=42).reset_index(drop=True)
```

1.2 Исследование совместной обработки наборов данных

Для начала нужно загрузить изображения из предоставленных путей к файлам с использованием библиотеки OpenCV.

Функция load_images принимает два аргумента: file_paths (список путей к файлам) и target_size (желаемый размер изображений после обработки). Внутри функции создаётся список изображений (images) и цикл по каждому пути к файлу в file_paths. Если изображение успешно загружено функцией cv2.imread, оно преобразуется с помощью функции cv2.resize и добавляется в список images. В конце функция возвращает массив numpy (np.array) с обработанными изображениями.

Затем создаются переменные *x* и *y*, содержащие пути к файлам и метки классов соответственно. После загрузки изображений с помощью функции load_images они приводятся к типу float32 и делятся на 255 для нормализации значений пикселей.

Наконец, происходит разделение данных на обучающую и тестовую выборки с использованием функций train_test_split и stratify. Обучающая выборка составляет 80% от всех данных, а тестовая — 20%.

Далее происходит предварительная обработка данных для задачи классификации изображений. Сначала используется метка encoder для преобразования строковых меток (классов) в числовые значения. Затем определяется количество уникальных классов (число меток) и выполняется

преобразование числовых меток в категориальные значения. В конце задаётся форма входных данных (размер изображения и количество каналов).

```
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)
y_test_encoded = label_encoder.transform(y_test)
num_classes = len(set(y_train_encoded))
y_train_one_hot = to_categorical(y_train_encoded, num_classes=num_classes)
y_test_one_hot = to_categorical(y_test_encoded, num_classes=num_classes)
input_shape = (256, 256, 3)
```

Далее нужно создать модель с последовательной архитектурой с использованием библиотеки Keras. Модель состоит из нескольких слоев:

- первый слой — Conv2D с 32 фильтрами размером 3x3, функцией активации relu и формой входного слоя, определённой как input_shape;
- за ним следует слой MaxPooling2D с размером окна 2x2;
- далее идут ещё три слоя Conv2D с 64, 128 и 32 фильтрами соответственно, каждый с функцией активации relu;
- после этого идёт слой MaxPooling2D с размером окна 2x2;
- затем следует слой Flatten для преобразования многомерных данных в одномерный вектор;
- далее идут два полносвязных слоя с 128 нейронами и функцией активации relu каждый;
- наконец, последний слой — это слой Dense с num_classes нейронами и функцией активации softmax для классификации.

В качестве примера, создадим модель и добавим туда блок свертки:

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=input_shape))
model.add(layers.MaxPooling2D((2, 2)))
```

Далее происходит компиляция модели с использованием оптимизатора adam, функции потерь crossentropy и метрики accuracy. Затем модель обучается с использованием полученных раннее данных в течение 10 эпох с размером батча

32 и данными для валидации. В конце выводится сводка модели с указанием количества параметров и структуру слоев.

```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(X_train, y_train_one_hot, epochs=10, batch_size=32,
                    validation_data=(X_test, y_test_one_hot))
model.summary()
```

Для построения графиков обучения и проверки точности потерь требуется импортировать библиотеку Matplotlib. График строится для каждой метрики отдельно (рисунок 1.1).

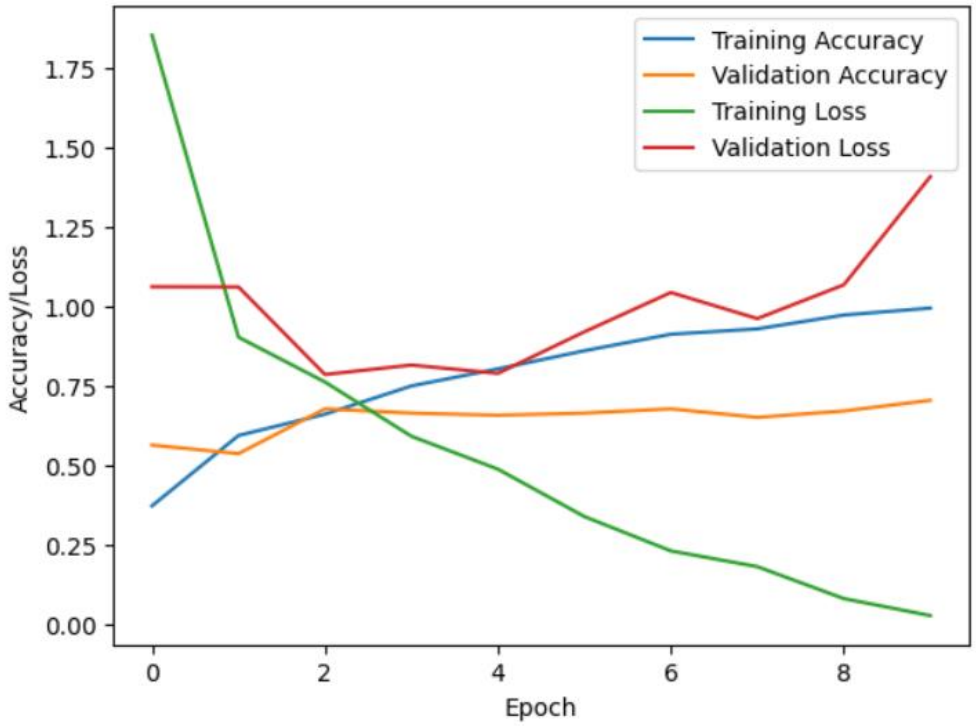

Рисунок 1.1 – График обучения и проверки точности потерь

1.3 Исследование раздельного анализа наборов данных

Если изображений в наборе много, Google Colab может не выдержать и перегрузиться из-за переполненной ОЗУ. Если такое случается, можно воспользоваться функцией ниже, для того чтобы уменьшить количество изображений в каждом классе.

```
limit_per_class = 1000
```

```
def trim_class(df, class_name, limit):
    class_df = df[df['Class'] == class_name]
    if len(class_df) > limit:
        class_df = class_df.sample(n=limit, random_state=42)
    return class_df
trimmed_df = pd.concat([trim_class(df_mri, class_name, limit_per_class) for class_name in df_mri['Class'].unique()])
```

Далее происходит создание и компиляция модели в соответствие с предыдущим пунктом, построение графика (рисунок 1.2).

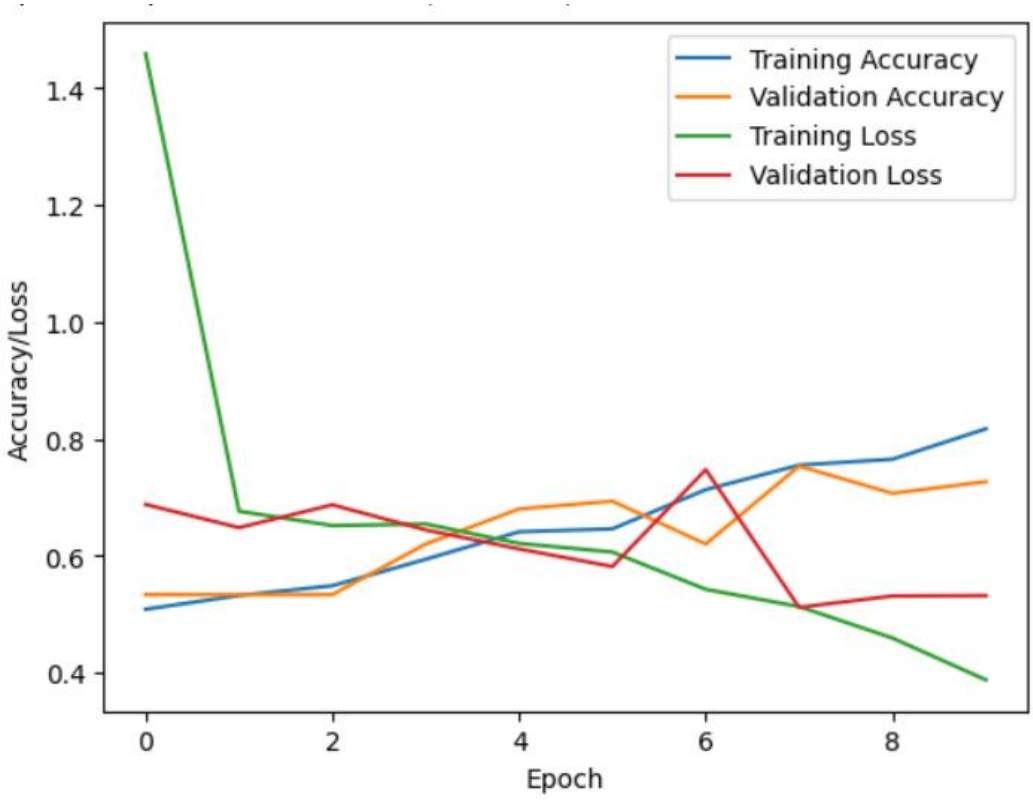

Рисунок 1.2 – График обучения и проверки точности потерь

1.4 t-SNE

T-SNE (t-Stochastic Neighbor Embedding) — это алгоритм машинного обучения для визуализации данных, разработанный Лоренсом ван дер Маатеном и Джеффри Хинтоном. Он предназначен для снижения размерности данных высокой размерности и их визуализации в пространстве низкой размерности (обычно двумерном или трёхмерном). T-SNE моделирует каждый объект

высокой размерности как двух- или трёхмерную точку, при этом похожие объекты располагаются близко друг к другу, а непохожие — на большом расстоянии.

Для начала нужно импортировать необходимые библиотеки. Далее происходит предварительная обработка изображений с использованием функции load_and preprocess_image(). Эта функция загружает изображение по указанному пути, преобразует цветовую схему и изменяет размер изображения до заданного.

```
def load_and_preprocess_image(file_path):
    img = cv2.imread(file_path)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)  #
    img = cv2.resize(img, image_size)
    img = img / 255.0
    return img
```

Далее следует снижение размерности данных с помощью метода Метод главных компонент (PCA) с количеством компонентов, равным 50. При таком значении сохраняется баланс между объяснением дисперсии исходных переменных и количеством учтённых компонент. Эти данные преобразуются с использованием t-SNE с двумя компонентами и параметрами perplexity и n_iter. В конце визуализируются полученные данные с помощью t-SNE, где уникальные метки (labels) отображаются как разные цвета и размеры точек (рисунок 1.3).

```
images_flat = images.reshape(len(images), -1)
pca = PCA(n_components=50)
images_pca = pca.fit_transform(images_flat)
tsne = TSNE(n_components=2, perplexity=30, n_iter=300)
images_tsne = tsne.fit_transform(images_pca)
```

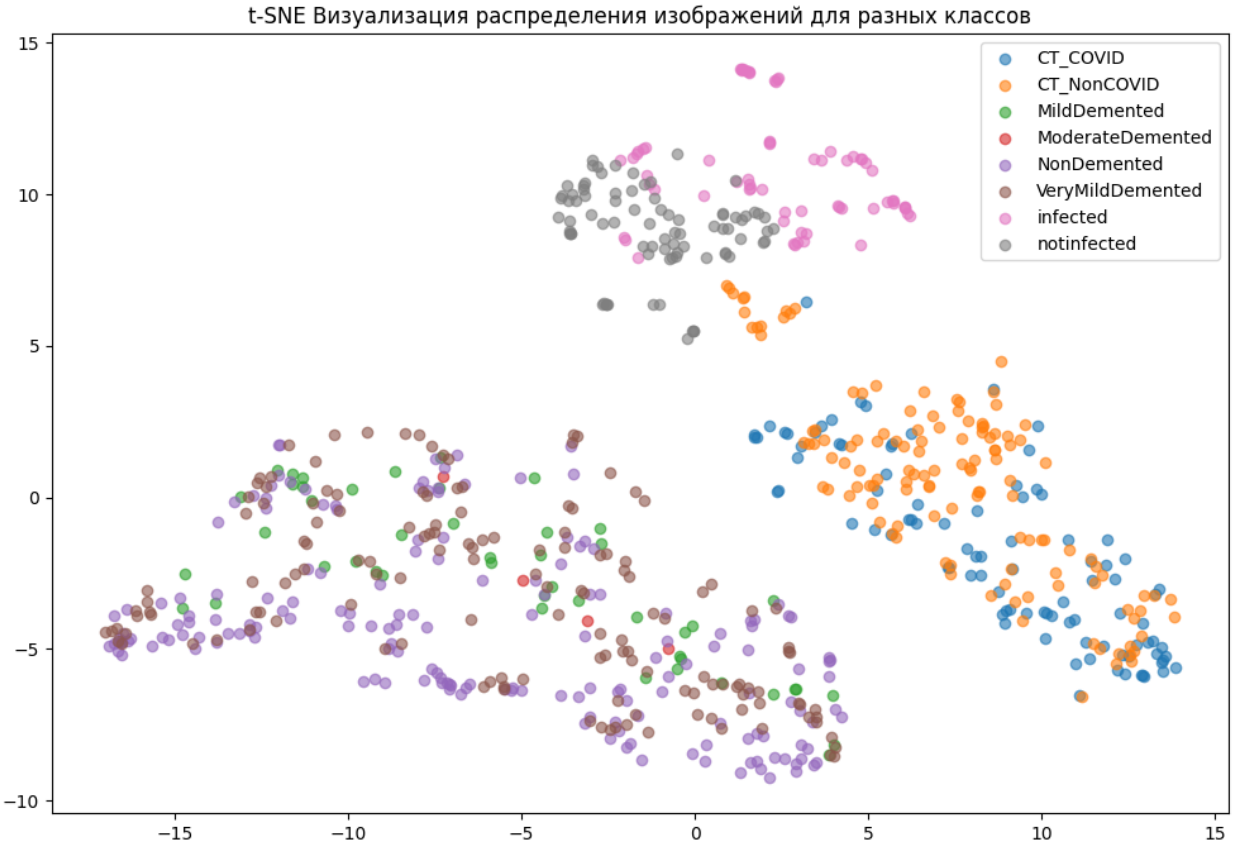

Рисунок 1.3 – Визуализация распределения изображений разных классов

2 Индивидуальное задание

1. Изменить процентное соотношение смешивания наборов данных
2. На основе изученного материала настроить слои для получения вариации нейронной сети;
3. Провести классификацию, сравнить полученные результаты, сделать выводы;

Контрольные вопросы

1. В чём заключается задача обучения нейронной сети для распознавания объектов на изображении?
2. Какие преимущества имеет совместная обработка данных нейронными сетями по сравнению с использованием отдельных сетей?
3. Что такое t-SNE и как он работает?
4. Какие основные параметры используются в t-SNE и как они влияют на результат?

ЛАБОРАТОРНАЯ РАБОТА №6

Сегментация и классификация болезней кожи

Целью лабораторной работы является разработка и тестирование двух различных архитектур нейронных сетей с предобученными моделями VGG19, YOLOv8 и EfficientNetB5 для классификации изображений кожных заболеваний, а также выполнение сегментации на этом наборе данных. В рамках эксперимента планируется настроить параметры моделей и проанализировать их производительность для выбора наилучшего подхода.

Краткие теоретические сведения

VGG19 – это глубокая сверточная нейронная сеть, состоящая из 19 слоев, предложенная в 2014 году группой VGG Оксфордского университета. Архитектура использует небольшие 3x3 сверточные фильтры и слои max pooling для поэтапного уменьшения размерности, что позволяет улавливать сложные визуальные признаки. Модель обеспечивает высокую точность, но имеет большое количество параметров, что делает её вычислительно затратной.

EfficientNet-B5 – это часть семейства моделей EfficientNet, предложенного Google в 2019 году, которое использует подход compound scaling для сбалансированного увеличения глубины, ширины и разрешения сети. Архитектура построена на основе улучшенных блоков MBConv и использует swish-функцию активации для повышения эффективности. EfficientNet-B5 достигает высокой точности при меньшем количестве параметров и более низких вычислительных затратах по сравнению с традиционными моделями.

YOLOv8 – это версия алгоритма YOLO (You Only Look Once) для детекции объектов в реальном времени. YOLOv8 использует улучшенные архитектуры нейронных сетей, что позволяет достичь высокой точности и скорости детекции при сохранении компактности модели.

В лабораторной работе используется набор данных с Kaggle https://www.kaggle.com/datasets/dipuiucse/monkeypoxskinimagedataset. Набор состоит из четырех классов: оспа обезьян, ветрянка, корь и обычная оспа. Все классы изображений собраны из интернет-источников. Весь набор данных был разработан департаментом компьютерных наук и инженерии Исламского университета, Куштия-7003, Бангладеш.

Требования к выполнению лабораторной работы

1. Ознакомиться с методическими указаниями, а также представленными примерами в Google Colab.
2. Выполнить ход работы.
3. Выполнить индивидуальное задание.
4. Подготовить набор данных, провести сегментацию набора при помощи YOLOv8.
5. Ответить на контрольные вопросы.

1 Ход работы

1.1 YOLOv8 и AutoLabelling

Перед началом выполнения лабораторной работы нужно скачать набор данных с Kaggle при помощи библиотеки opendatasets и установить библиотеки roboflow и ultralytics. Roboflow используется для управления набором данных, а Ultralytics предоставляет инструменты для работы с моделями YOLOv8.

После установки библиотек нужно импортировать Roboflow и инициализировать его с помощью API-ключа. API-ключ необходим для подключения к аккаунту Roboflow, чтобы получить доступ к рабочим

пространствам и проектам. Данный ключ можно найти во вкладке Settings (рисунок 1.6).

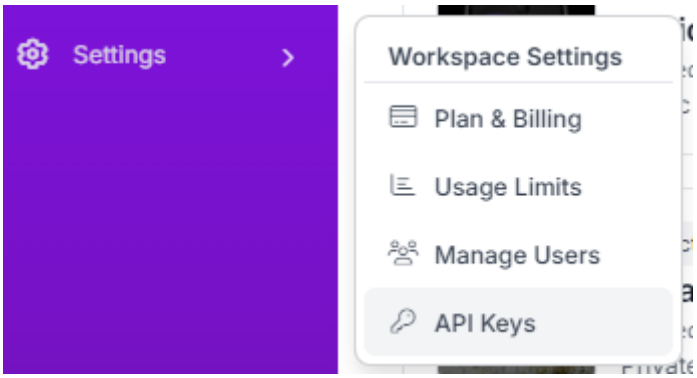

Рисунок 1.6 – API ключ

Далее следует подключение к нужному проекту и загрузка изображения для сегментации в проект.

```
from roboflow import Roboflow
import os
rf = Roboflow(api_key="******")
print(rf.workspace())
workspaceId = '***'
projectId = '***'
project = rf.workspace(workspaceId).project(projectId)
```

При помощи встроенного функционала Roboflow нужно разметить набор данных. Это делается в разделе Annotate. Нажав на Annotate нужно выбрать пункт Auto Label (рисунок 1.7). Здесь нужно задать запрос для автоматического распознавания, и расписать про то, какой объект нужно будем сегментировать. После того, как будут размечены тестовые изображения, начнется автоматическая разметка.

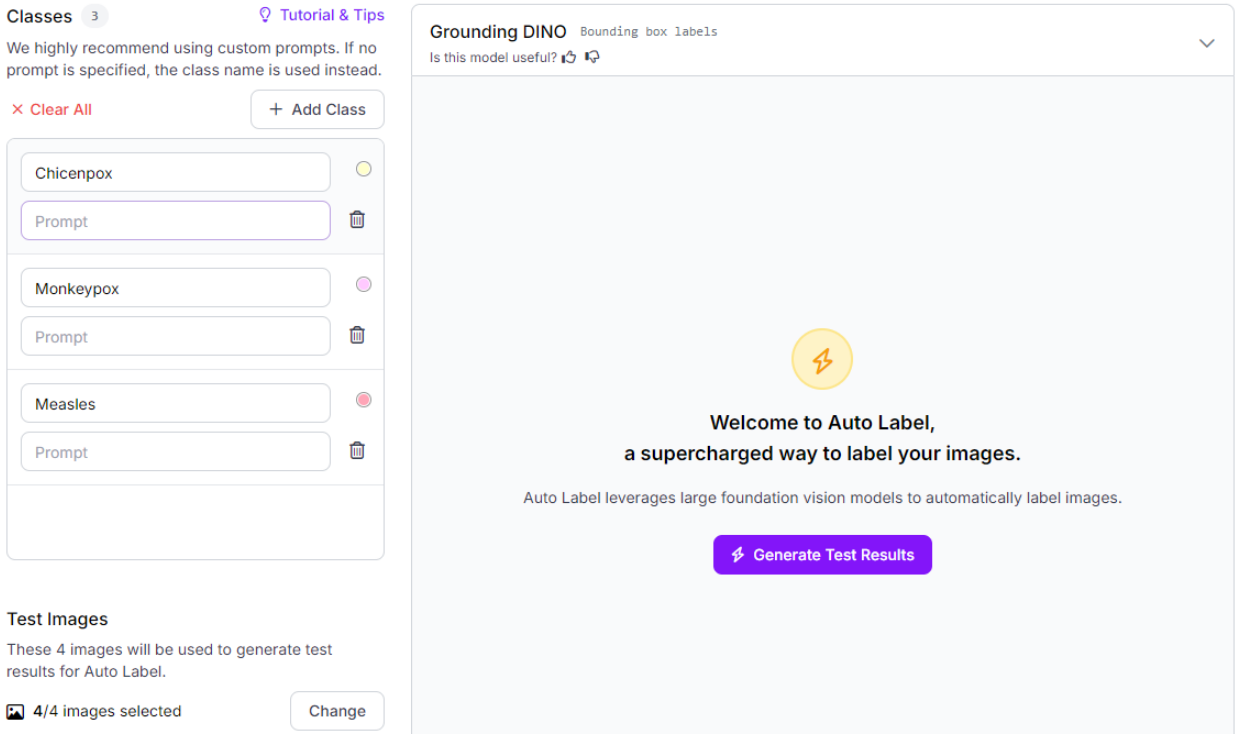

Рисунок 1.7 – Auto Label

После загрузки изображений и создания проекта, нужно загрузить набор данных в формат, совместимый с моделью YOLOv8. Это можно сделать напрямую из Roboflow.

```
from roboflow import Roboflow
rf = Roboflow(api_key="***")
project = rf.workspace("***").project("***")
version = project.version(1)
dataset = version.download("yolov8")
```

После загрузки данных требуется перенести data.yaml в директорию «/content/» и после этого запустить обучение.

Далее выполняются предсказания сегментации для загруженного изображения.

```
import cv2
image_path = "***"
image = cv2.imread(image_path)
results = model.predict(image, conf=0.5)
```

Далее нужно провести предсказание для каждого изображения из тестовой выборки, для этого используется цикличный перебор всех изображения из папки. Так же во время проведения предсказания нужно вырезать сегментированные области для проведения дальнейшей классификации.

```
results = model.predict(image, conf=0.5)
  for result in results:
    if result.masks is not None:
        for idx, (mask, box, class_id) in enumerate(zip(result.masks.xy, result.boxes.xywh, result.boxes.cls)):
          x_center, y_center, width, height = box
          x1 = int(x_center - width / 2)
          y1 = int(y_center - height / 2)
          x2 = int(x_center + width / 2)
          y2 = int(y_center + height / 2)
          cropped_image = image[y1:y2, x1:x2]
```

1.2 VGG19

Первым шагом выполняется импорт всех необходимых библиотек и модулей, используемых в лабораторной работе. Для работы с данными и изображениями применяются библиотеки NumPy, pandas, OpenCV и matplotlib. Визуализация данных проводится с помощью seaborn. Библиотека TensorFlow и её модуль Keras используются для создания, обучения и тестирования моделей глубокого обучения. Также импортируются функции для управления моделями, такие как EarlyStopping и ModelCheckpoint. Это позволяет создать гибкую и функциональную среду для построения и обучения нейронных сетей.

Далее осуществляется разделение данных на три основные группы: тренировочные, валидационные и тестовые наборы. Для этого нужно выполнить функцию, которая распределяет изображения по указанным папкам в зависимости от заданных пропорций. Тренировочные данные используются для обучения модели, валидационные — для проверки её работы в процессе

обучения, а тестовые данные — для окончательной оценки производительности модели. На примере показано разбиение класса *«Monkeypox»*, автоматизируйте данный код, добавив разбиение каждого класса (это можно сделать, введя список классов).

```
class_image='Monkeypox'
def split_data(input_folder, output_folder, train_ratio=0.7, valid_ratio=0.2):
input_folder = '***'
output_folder = f'{class_image}'
split_data(input_folder, output_folder)
```

После разделения данных, они загружаются в формате, подходящем для обучения нейронной сети. Важно правильно настроить параметры загрузки, такие как размер изображений (*image_size)* и батчей данных (*batch_size)*, чтобы обеспечить эффективное обучение модели.

```
train_dir = "/content/train"
train_data = image_dataset_from_directory(train_dir,label_mode = "categorical", image_size = (img_height, img_width), batch_size = 16, shuffle = True, seed = 42)
```

После подготовки набора данных для обучения начинается создание модели нейронной сети. Первыми слоями станут слои аугментации - слои, которые расширят набор путём изменения яркости, переворота и поворота. Эти методы помогают увеличить разнообразие тренировочных данных и улучшить способность модели обобщать информацию.

```
augmentation.add(RandomBrightness(factor = 0.1))
augmentation.add(RandomFlip(mode = 'horizontal_and_vertical'))
augmentation.add(RandomRotation(factor = 0.2, fill_mode = 'nearest'))
```

Модель VGG19 загружается с предварительно обученными весами, которые были получены на большом наборе данных ImageNet. Это позволяет использовать знания, уже накопленные моделью, что может ускорять обучение и повышать точность. Важно отключить обучение слоёв модели, кроме последних блоков, чтобы сохранить предварительно обученные веса и предотвратить переобучение.

```
vgg19 = VGG19(input_shape = (img_height, img_width, 3), weights = 'imagenet', include_top =
```

```
False)
set_true = False
for layer in vgg19.layers:
    if layer.name == 'block5_conv1':
        layer.trainable = True
        set_true = True
    if set_true:
        layer.trainable = True
```

Для завершения модели стоит добавить слои Dropout и полносвязных слоёв для классификации. Эта архитектура позволяет модели извлекать и использовать полезные признаки из изображений для их дальнейшего классифицирования.

```
model.add(Input(shape=input_shape))
model.add(augmentation)
model.add(vgg19)
model.add(Dropout(0.4))
model.add(Flatten())
model.add(Dense(100, activation = 'relu'))
model.add(Dense(4, activation = 'softmax'))
```

После этого идет компиляция полученной модели. Здесь также добавляется функция потерь и оптимизатор Adam.

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

После компиляции модели получается архитектура на рисунке 1.1.

```
model.summary()
```

```
Model: "sequential_3"
```

Layer (type)	Output Shape	Param #
sequential_2 (Sequential)	(None, 224, 224, 3)	0
vgg19 (Functional)	(None, 7, 7, 512)	20,024,384
dropout_1 (Dropout)	(None, 7, 7, 512)	0
flatten_1 (Flatten)	(None, 25088)	0
dense_2 (Dense)	(None, 100)	2,508,900
dense_3 (Dense)	(None, 4)	404

```
Total params: 22,533,688 (85.96 MB)
Trainable params: 11,948,536 (45.58 MB)
Non-trainable params: 10,585,152 (40.38 MB)
```

Рисунок 1.1 – Архитектура полученной модели

После завершения модели следует добавление методов для контроля процесса обучения – ModelCheckpoint и EarlyStopping. ModelCheckpoint сохраняет модель с лучшими результатами на валидационном наборе данных, что позволяет вернуть к лучшему состоянию в случае, если обучение ухудшится. EarlyStopping помогает избежать переобучения, останавливая процесс, если улучшения на валидационном наборе данных прекращаются.

```
checkpoint = ModelCheckpoint(filepath = chk_path, monitor='val_accuracy', mode='max', save_best_only = True, verbose = 1)
early_stopping = EarlyStopping(monitor = 'val_accuracy', patience = 5, mode = 'max', verbose = 1)
```

Запустим обучение. Модель обучается на тренировочных данных в течение определённого количества эпох. В процессе обучения модель проверяется на валидационных данных, чтобы оценить её производительность и предотвратить переобучение. Результаты обучения, такие как точность и потеря, отслеживаются и используются для оценки эффективности модели.

```
history = model.fit(train_data, validation_data = val_data, epochs = 2, batch_size = 8, callbacks = [early_stopping, checkpoint])
```

Графики помогают наглядно представить, как менялись значения точности и потерь в процессе обучения, а также сравнить их между тренировочными и валидационными данными.

В результате выполнения обучения, будет построено два графика для визуализации полученных результатов. На графике точности отображаются изменения значения точности на тренировочных и валидационных данных в зависимости от числа эпох. График потерь демонстрирует изменения значения потерь на тренировочных и валидационных данных в процессе обучения (рисунок 1.2 – 1.3).

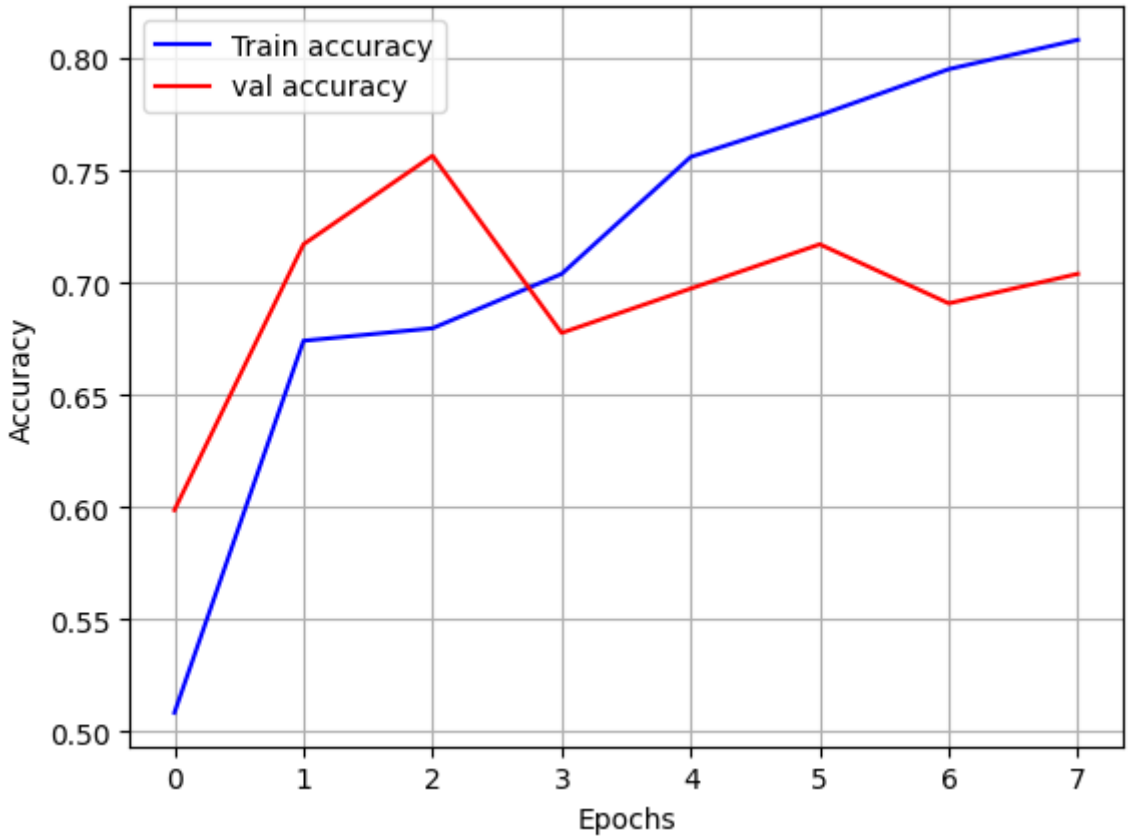

Рисунок 1.2 – График точности

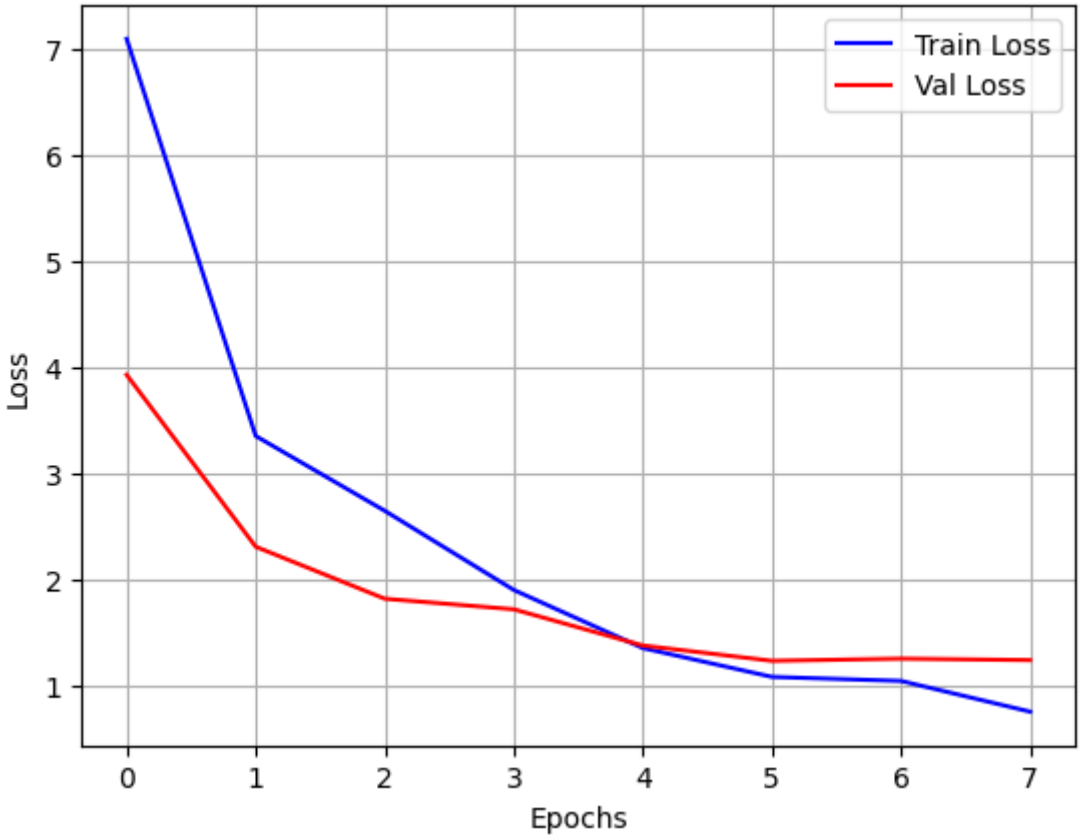

Рисунок 1.3 – График потерь

1.3 EfficientNetB5

Для начала обучения модели с предобученной EfficientNetB5 возьмем разбитые ранее данные, а также зададим методы для отслеживания процесса обучения. Здесь этими методами являются – EarlyStopping и ReduceLROnPlateau, который уменьшает скорость обучения, если ошибка на валидационном наборе данных не уменьшается после определенного количества эпох.

```
early_stop = tf.keras.callbacks.EarlyStopping(monitor = "val_loss",patience = 5, min_delta = 0.0001)
reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(monitor = "val_loss",factor = 0.2, patience = 4, min_lr = 1e-7)
```

Далее, как и в пункте с VGG19, следует сбор архитектуры нейронной сети с добавлением предобученной модели EfficientNetB5, а также слоёв с аугментацией.

```
model = tf.keras.applications.EfficientNetB5(include_top = False, weights='imagenet')
```

```
model.trainable = False
```

Итоговая модель представлена на рисунке 1.4.

Model: "functional_19"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 224, 224, 3)	0
data_augmentation_layer (Sequential)	(None, None, None, 3)	0
efficientnetb5 (Functional)	(None, None, None, 2048)	28,513,527
pooling_layer (GlobalAveragePooling2D)	(None, 2048)	0
dense_8 (Dense)	(None, 256)	524,544
dropout_4 (Dropout)	(None, 256)	0
dense_9 (Dense)	(None, 4)	1,028
activation_2 (Activation)	(None, 4)	0

Total params: 29,039,099 (110.78 MB)
Trainable params: 525,572 (2.00 MB)
Non-trainable params: 28,513,527 (108.77 MB)

Рисунок 1.4 – Модель с EfficientNetB5

Модель компилируется с использованием функции потерь CategoricalCrossentropy для задач многоклассовой классификации. Оптимизатор Adam используется для минимизации функции потерь. Далее модель обучается на тренировочных данных с использованием валидационного набора, и при этом применяются указанные ранее *callbacks*. После завершения обучения модель оценивается на тестовом наборе данных, и выводится точность.

```
history = model.fit(train_data,epochs =1, validation_data = val_data, callbacks = [early_stop,reduce_lr])
```

Аналогично прошлому пункту нужно вывести график точности и график потерь.

Для вывода F1 – меры соберем истинные метки классов (y_labels) из тестового набора данных. Затем предскажем вероятности классов с использованием обученной модели. Эти вероятности преобразуем в предсказанные классы (*pred_classes*). Classification_report из библиотеки sklearn используется для вывода подробного отчета о качестве модели по метрикам

точности, полноты и F1-меры для каждого класса. Это помогает понять, насколько хорошо модель классифицирует каждый класс.

```
from sklearn.metrics import classification_report
print("Classification Report\n",classification_report(y_labels,pred_classes))
```

Рассчитываются F1-меры для каждого класса и представляются в виде таблицы с использованием pandas. Это позволяет быстро увидеть, для каких классов модель работает лучше всего, а для каких — хуже.

```
import pandas as pd
f1_scores = pd.DataFrame({"class_name":list(classification_f1_scores.keys()), "F1-Scores":list(classification_f1_scores.values())})
f1_scores.sort_values("F1-Scores",ascending = False)
```

Далее строится матрица запутанности. Функция make_confusion_matrix() строит матрицу, которая наглядно показывает, сколько изображений каждого класса было правильно или неправильно классифицировано. Матрица может быть нормализована для удобства интерпретации. Каждая ячейка представляет собой количество предсказаний и позволяет легко увидеть, где модель ошибается (рисунок 1.5).

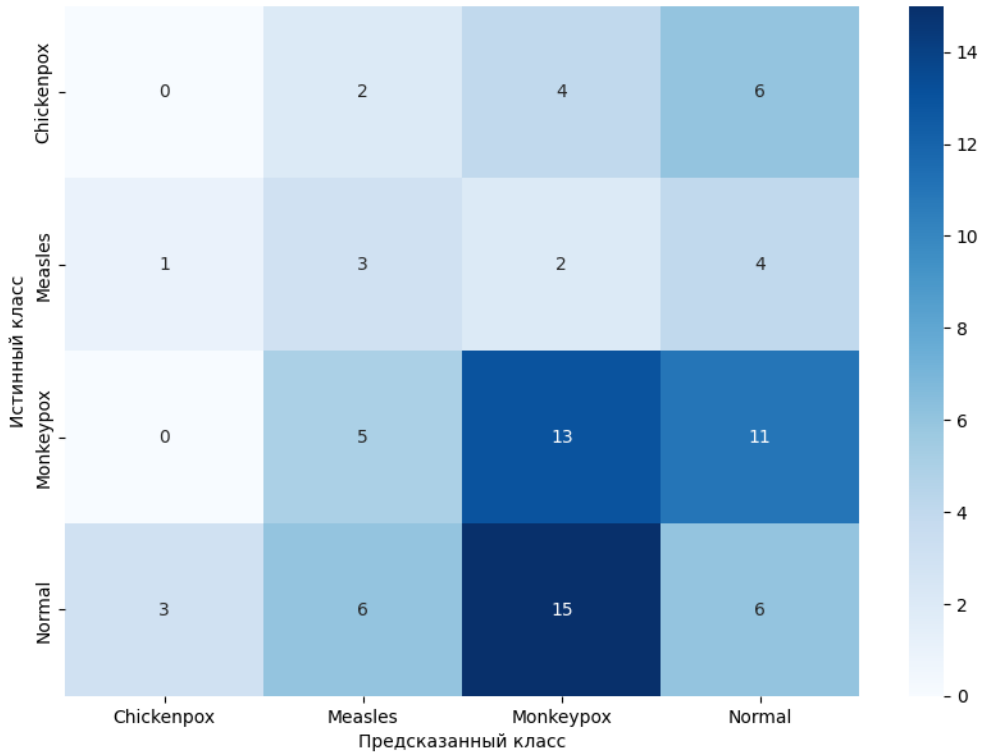

Рисунок 1.5 – Матрица запутанности

Иindивидуальное задание

1. Провести автоматическую разметку набора данных;
2. Провести сегментацию на полученном наборе.
3. На основе изученного материала настроить различные параметры для получения двух вариаций нейронной сети;
4. Провести классификацию, сравнить полученные результаты, сделать выводы;

Контрольные вопросы

1. Какие основные задачи решает сегментация в дерматологии?
2. В чём разница между пороговой и нечёткой сегментацией?
3. Какие параметры влияют на качество сегментации?
4. Какие методы классификации вы знаете?
5. Как рассчитывается F1-мера и какие показатели она объединяет?
6. Как использовать F1-меру для оценки качества работы нейронной сети?
7. Как работает матрица запутанности и какую роль она играет при анализе изображений?

ПРАКТИЧЕСКАЯ РАБОТА №1

Математика свертки

Целью работы является изучение математических основ свертки, а также освоение навыков ее применения в решении задач.

Требования к выполнению работы

1. задание выполняется по вариантам, согласно порядковому номеру к списке группы;
2. все промежуточные расчёты, если такие имеются, должны быть представлены в решении.

Варианты заданий

Вариант 1

Задача 1. Расчет значений свёртки.

Дано входное изображение I:

$$\begin{vmatrix} 7 & 11 & 18 \\ 9 & 5 & 10 \\ 22 & 8 & 9 \end{vmatrix}$$

Фильтр F:

$$\begin{vmatrix} 1 & 1 \\ 0 & -1 \end{vmatrix}$$

Рассчитать значения свёртки без паддинга, шаг 1.

Задача 2. Расчет значений свёртки.

Дано входное изображение I:

$$\begin{vmatrix} 34 & 22 & 16 & 9 \\ 9 & 5 & 10 & 2 \\ & & & 3 \\ 22 & 8 & 9 & 5 \\ 11 & 8 & 12 & 7 \end{vmatrix}$$

Фильтр F:

$$\begin{vmatrix} 0 & 2 & 1 \end{vmatrix}$$

1	-1	1
2	1	0

Рассчитать значения свёртки без паддинга, шаг 2.

Задача 3. Операции пулинга.

Дана матрица I:

1	5	7	13
4	1	4	11
	3		
17	3	1	9
12	6	9	2

Рассчитать выходные карты после операций пулинга (2x2), паддинг 0, шаг 2: max-pooling, average-pooling и min-pooling.

Задача 4. Операции пулинга.

Дана матрица I:

70	22	66	2
			5
45	67	13	2
			3
27	84	91	5
			3
137	35	9	0

Рассчитать выходные карты после операций пулинга (2x2), паддинг 0, шаг 2: max-pooling, average-pooling и min-pooling.

Задача 5. Определить размер входного изображения.

Входное изображение имеет размеры 32x32, а размер фильтра свертки составляет 5x5. Какой будет размер выходного изображения после применения техники паддинг: 10px, 15px, 20px, 30px?

Задача 6.

Дано входное изображение I:

15	4	23	14	2
6	9	18	7	20
8	6	21	5	3
19	12	20	14	1
2	8	6	13	9

Фильтр F:

1	2	0

-1	0	2
2	1	-1

Рассчитать значения свёртки, паддинг = 1, шаг 1.

Рассчитать выходные карты после операций пулинг (2x2) max-pooling, average-pooling и min-pooling.

Задача 7.

Дано входное изображение I:

45	89	23	4	98	73	6	76
33	1	15	75	88	37	80	65
34	5	89	9	30	1	23	67
33	68	9	54	68	22	55	24
67	44	45	96	97	33	23	55
65	29	83	8	34	78	64	13
15	11	55	88	46	24	89	24
76	97	24	56	86	24	98	55

Фильтр F:

1	0	-1
1	2	-1
0	2	2

Рассчитать значения свёртки без паддинга, с шагом 2.

Рассчитать выходные карты после операций пулинга (2x2): max-pooling, average-pooling и min-pooling.

Вариант 2

Задача 1. Расчет значений свёртки.

Дано входное изображение I:

4	31	8
5	9	14
17	23	1

Фильтр F:

1	-1
-1	1

Рассчитать значения свёртки без паддинга, шаг 1.

Задача 2. Расчет значений свёртки.

Дано входное изображение I:

56	42	2	1
			3
11	53	44	4
			3
5	23	4	5
			4
9	1	29	1
			7

Фильтр F:

-1	-1	2
2	-1	0
1	2	-1

Рассчитать значения свёртки без паддинга, шаг 2.

Задача 3. Операции пулинга.

Дана матрица I:

2	9	6	1
			2
9	6	3	1
			7
25	7	25	1
			2
1	28	13	8

Рассчитать выходные карты после операций пулинга (2x2), паддинг 0, шаг 2: max-pooling, average-pooling и min-pooling.

Задача 4. Операции пулинга.

Дана матрица I:

70	22	66	2
			5
45	67	13	2
			3
27	84	91	5
			3
137	35	9	0

Рассчитать выходные карты после операций пулинга (2x2), паддинг 0, шаг 2: max-pooling, average-pooling и min-pooling.

Задача 5. Определить размер входного изображения.

Входное изображение имеет размеры 28x28, а размер фильтра свертки составляет 7x7. Какой будет размер выходного изображения после применения техники паддинг: 15px, 15px?

Задача 6.

Дано входное изображение I:

9	14	5	7	21
1	17	26	16	18
3	7	22	7	9
4	15	21	12	19
2	12	31	5	15

Фильтр F:

2	2	1
-1	2	-1
-1	0	1

Рассчитать значения свёртки, паддинг = 1, шаг 1.

Рассчитать выходные карты после операций пулинг (2x2) max-pooling, average-pooling и min-pooling.

Задача 7.

Дано входное изображение I:

54	96	24	84	52	21	97	77
23	89	24	41	85	57	10	17
12	68	73	38	9	81	20	67
57	23	37	96	97	48	41	65
27	56	21	69	59	11	83	3
68	28	2	34	5	85	7	28
67	89	35	88	75	17	7	34
78	5	74	96	3	21	89	46

Фильтр F:

2	1	1
-1	0	1
1	2	1

Рассчитать значения свёртки без паддинга, с шагом 2.

Рассчитать выходные карты после операций пулинга (2x2): max-pooling, average-pooling и min-pooling.

Вариант 3

Задача 1. Расчет значений свёртки.

Дано входное изображение I:

$$\begin{vmatrix} 7 & 4 & 13 \\ 24 & 2 & 14 \\ 1 & 16 & 9 \end{vmatrix}$$

Фильтр F:

$$\begin{vmatrix} -1 & 1 \\ -1 & 1 \end{vmatrix}$$

Рассчитать значения свёртки без паддинга, шаг 1.

Задача 2. Расчет значений свёртки.

Дано входное изображение I:

$$\begin{vmatrix} 34 & 23 & 68 & 1 \\ 54 & 6 & 12 & 3 \\ & & & 3 \\ 6 & 42 & 12 & 1 \\ & & & 5 \\ 51 & 22 & 12 & 1 \\ & & & 8 \end{vmatrix}$$

Фильтр F:

$$\begin{vmatrix} 1 & 2 & -1 \\ 0 & 1 & 1 \\ -1 & 1 & 1 \end{vmatrix}$$

Рассчитать значения свёртки без паддинга, шаг 2.

Задача 3. Операции пулинга.

Дана матрица I:

$$\begin{vmatrix} 2 & 9 & 6 & 1 \\ & & & 2 \\ 9 & 6 & 3 & 1 \\ & & & 7 \\ 25 & 7 & 25 & 1 \\ & & & 2 \\ 1 & 28 & 13 & 8 \end{vmatrix}$$

Рассчитать выходные карты после операций пулинга (2x2), паддинг 0, шаг 2: max-pooling, average-pooling и min-pooling.

Задача 4. Операции пулинга.

Дана матрица I:

13	18	81	2
			6
75	24	66	1
			8
6	80	1	2
7	15	47	1
			2

Рассчитать выходные карты после операций пулинга (2x2), паддинг 0, шаг 2: max-pooling, average-pooling и min-pooling.

Задача 5. Определить размер входного изображения.

Входное изображение имеет размеры 28x28, а размер фильтра свертки составляет 5x5. Какой будет размер выходного изображения после применения техники паддинг: 10px, 30px, 25px?

Задача 6.

Дано входное изображение I:

5	88	23	77	12
97	13	13	67	4
88	2	76	13	35
6	81	27	57	12
76	23	78	11	7

Фильтр F:

1	2	1
-1	2	1
1	0	2

Рассчитать значения свёртки, паддинг = 1, шаг 1.

Рассчитать выходные карты после операций пулинг (2x2) max-pooling, average-pooling и min-pooling.

Задача 7.

Дано входное изображение I:

12	87	33	79	1	35	9	37
18	45	74	22	85	19	46	85
2	67	68	2	78	9	83	1
78	3	34	18	73	69	28	17
17	96	4	75	84	87	45	12
16	85	23	65	34	86	18	38
75	24	97	35	89	98	99	1
17	5	89	5	99	4	1	6

Фильтр F:

-1	2	-1
2	2	0
1	1	0

Рассчитать значения свёртки без паддинга, с шагом 2.

Рассчитать выходные карты после операций пулинга (2x2): max-pooling, average-pooling и min-pooling.

ПРАКТИЧЕСКАЯ РАБОТА №2

Проектирование архитектуры нейронной сети

Целью работы является разработка архитектуры нейронной сети, способной выполнять сегментацию и детектирование объектов на медицинских изображениях.

Требования к выполнению работы

1. разработать архитектуру нейронной сети от 5 до 10 свёрточных слоев, можно использовать до 3х слоёв пулинга;
2. входной размер изображений: 224x224 или 256x256 пикселей (черно-белое или цветное изображение на выбор);
3. сеть должна содержать как минимум один сверточный блок для извлечения признаков, необходимо включить в архитектуру слои Batch Normalization и Dropout для регуляризации модели;
4. добавить не менее 3-4 слоев декодера для сегментации изображений;
5. в качестве основной функции активации для скрытых слоев рекомендуется использовать ReLU;
6. защитить разработанную модель преподавателю, уметь объяснить логику модели и работу каждого слоя.

Пример выполнения работы

1. Входной слой:
    1.1. Входное изображение 256x256 (1 или 3 канала);
2. Энкодер:
    2.1. Первый блок:
        2.1.1. Сверточный слой (32 фильтра, 3x3, активация ReLU)
        2.1.2. MaxPooling (2x2)

2.2. Второй блок:

2.2.1. Сверточный слой (64 фильтра, 3x3, активация ReLU)

2.2.2. MaxPooling (2x2)

3. Функция активации:

3.1. Сверточный слой (128 фильтров, 3x3, активация ReLU)

4. Декордер;

4.1. Первый блок:

4.1.1. Транспонированная свертка (64 фильтра, 2x2, шаг 2)

4.1.2. Пропускное соединение с энкодером второго блока

4.2. Второй блок:

4.2.1. Транспонированная свертка (32 фильтра, 2x2, шаг 2)

4.2.2. Пропускное соединение с энкодером первого блока

5. Выходной слой:

5.1. Сверточный слой (1 фильтр, 1x1, активация сигмоида для бинарной сегментации).

ПРАКТИЧЕСКАЯ РАБОТА №3

Построение логической цепочки действий на основе входных данных

Целью работы является составление алгоритмов, которые можно применить к изображению для сегментации и детектированию объектов на нем.

Требования к выполнению работы

1. составить алгоритм для сегментации и детектирования объектов, применимый к изображению согласно варианту;
2. работу можно выполнять в группах до 4х человек;
3. номер варианта соответствует порядковому номеру группы, выданной преподавателем;
4. количество алгоритмов неограничено, минимальное количество – один.

Пример выполнения работы

На рисунке 1 представлено рентген-изображение грудной клетки, к которому были составлены 3 возможных алгоритма для сегментации и детектирования объектов на нём.

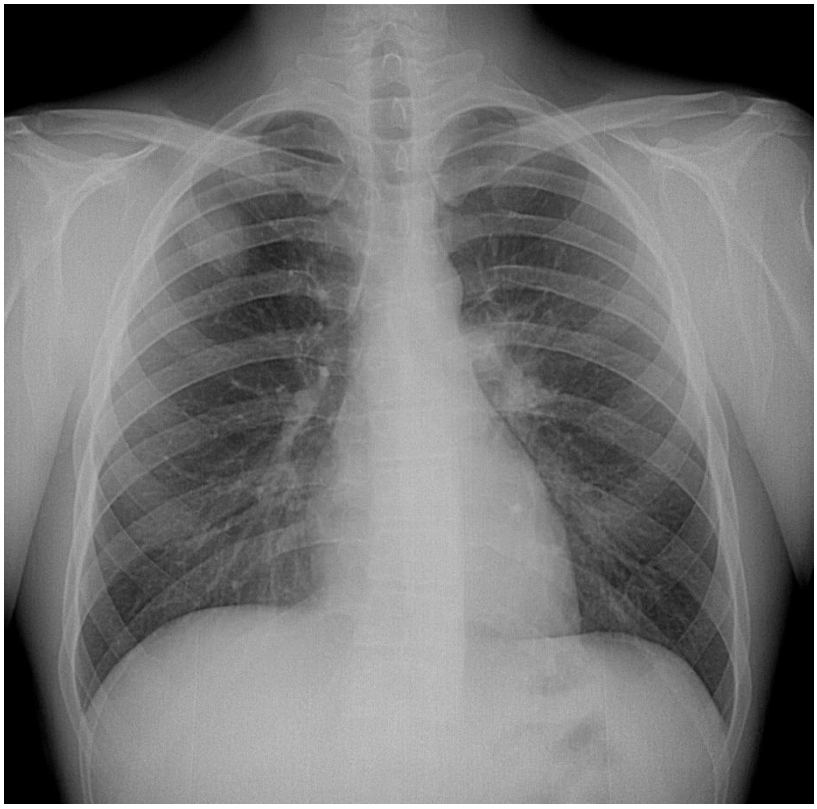

Рисунок 1 – Рентген легких

Пример 1.

1. Предобработка изображения

Усредняющий фильтр или Гауссово размытие для устранения высокочастотных шумов с сохранением основных контуров.

Преобразование изображения в оттенки серого.

2. Пороговое выделение

Автоматическое определение порога сегментации.

Бинаризация, чтобы выделить интересующие области, например, легкие или патологические образования**.**

3. Морфологические операции

Эрозия для удаления мелких шумов и заполнения пробелов в сегментированных изображениях.

Медианный фильтр для сглаживания границ**.**

4. Контуры и области интересов

Поиск контуров для выделения границ объектов.

Связные компоненты, чтобы идентифицировать и подсчитать различные области на изображении**.**

Пример 2.

1. Предобработка

Контрастное гистограммное выравнивание для улучшения контрастности и выявления тонких деталей.

2. Анализ текстур

Выделение признаков текстуры, например, локальные бинарные шаблоны, для классификации областей изображения.

Выделение текстур, характерных для здоровой и патологической ткани**.**

3. Сегментация

Классификация области на основе выделенных признаков текстуры с использованием методов, таких как k-means.

Разделение изображения на несколько классов, например, «легкое», «кость», «патология».

4. Постобработка

Объединение смежных областей одного класса, используя морфологические операции.

Оценка сегментации, при необходимости ручная или автоматическая коррекция**.**

Пример 3.

1. Сбор данных

Подготовка обучающей выборки размеченных изображений**.**

2. Архитектура сети

Использование готовой архитектур, например, U-Net или Mask R-CNN.

Настройка модели на вход изображения размером, 256x256**.**

3. Обучение

Использование аугментации данных (повороты, масштабирование, яркость), чтобы увеличить разнообразие обучающей выборки**.**

4. Постобработка

Удаление мелких объектов и сглаживание границ с помощью морфологических операций**.**

Задания

На рисунках 2 - 5 представлены изображения, для которых требуется составить алгоритмы анализа.

Рисунок 2 – Рентген желудка

Рисунок 3 – КТ головного мозга

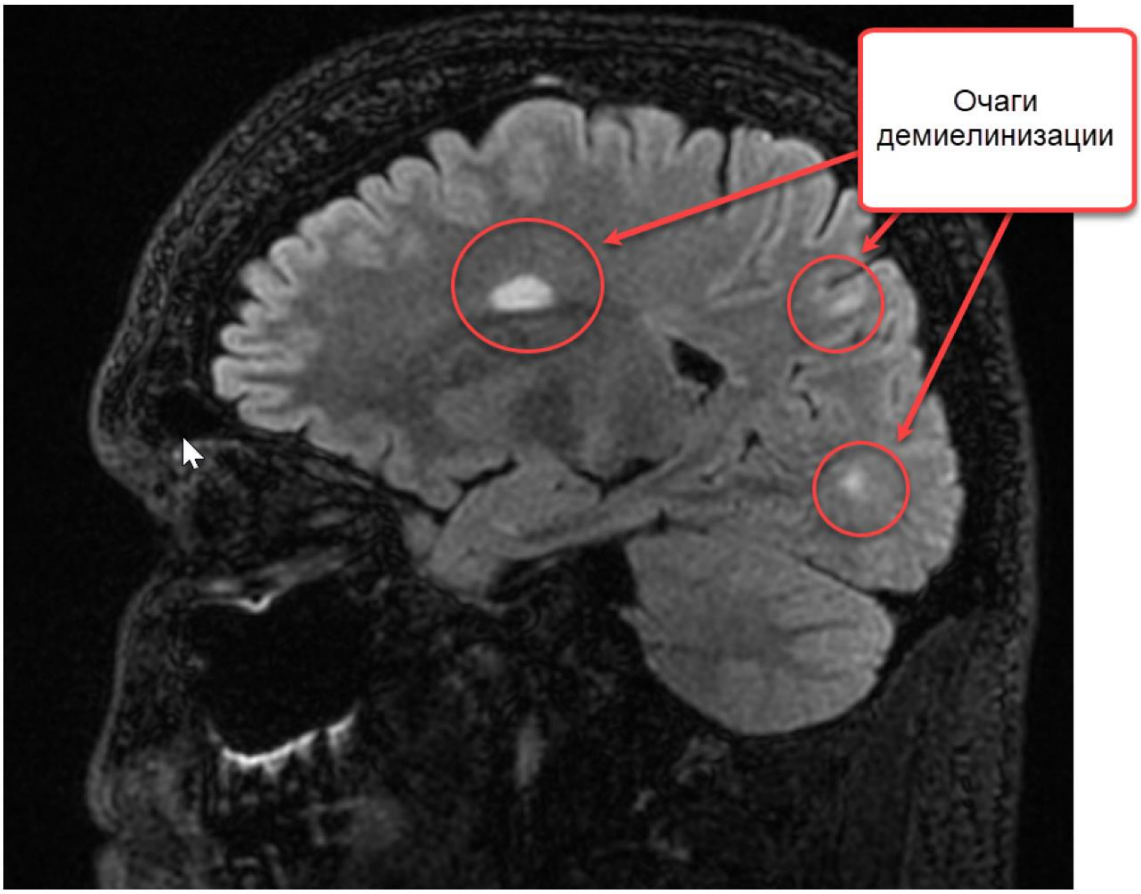

Рисунок 4 – МРТ мозга

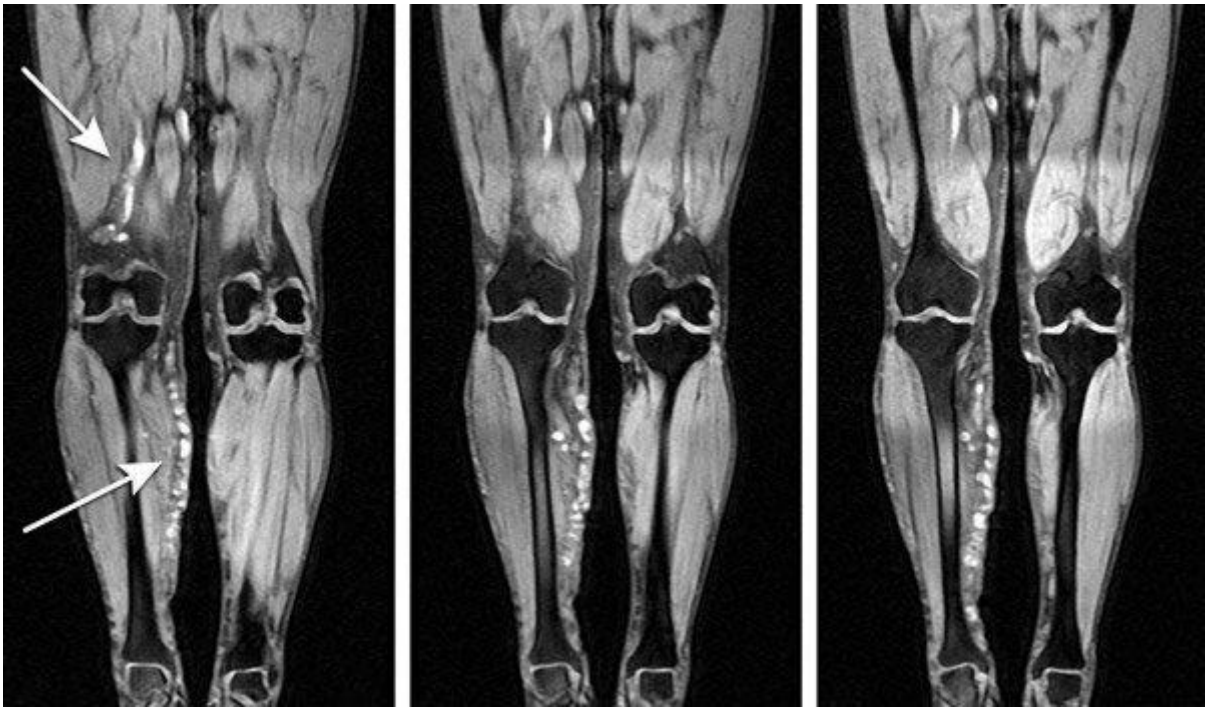

Рисунок 5 – МРТ ног

ПРАКТИЧЕСКАЯ РАБОТА №4

Практическое применение сегментации и детектирования объектов на медицинских изображениях

Задание

Студенты выбирают тему доклада и готовят выступление с кратким раскрытием темы на 5-10 минут.

Требования к докладам:

1. время выступления с докладом 5 минут для каждого студента и 5 минут ответы на вопросы;
2. доклад можно готовить в группах до 3-х человек;
3. оформление доклада по ОС ТУСУР обязательно;
4. требуется краткое раскрытие темы;
5. презентация к докладу обязательна.

Темы докладов

1. Использование глубоких нейронных сетей для обнаружения патологий на медицинских изображениях.
2. Анализ и сравнение различных методов сегментации и детектирования объектов на медицинских изображениях.
3. Инновационные подходы к анализу медицинских изображений.
4. Технологии для автоматической обработки и анализа медицинских изображений.
5. Использование искусственного интеллекта и глубокого обучения для автоматизации процесса анализа медицинских изображений.
6. Сравнение различных подходов к сегментации и детектированию объектов на медицинских изображениях.

7. Применение сегментации и детектирования объектов на медицинских изображениях для ранней диагностики рака молочной железы.

8. Сегментация и детектирование объектов на рентгеновских снимках лёгких для выявления туберкулёза и других заболеваний.

9. Применение сегментации и детектирования объектов на ультразвуковых изображениях плода для оценки состояния беременности и выявления аномалий развития.

10. Использование сегментации и детектирования объектов на МРТ-изображениях позвоночника для диагностики межпозвоночных грыж и других заболеваний.

11. Автоматическое обнаружение опухолей в МРТ головного мозга — алгоритмы сегментации для выявления опухолей на ранней стадии.

12. Использование сегментации для диагностики заболеваний легких — роль методов детектирования в анализе КТ легких при COVID-19 и других патологиях.

13. Определение границ сосудов и аневризм на КТ-изображениях — методы сегментации для планирования сосудистых операций.

14. Детекция и сегментация полипов — применение алгоритмов для автоматической диагностики заболеваний ЖКТ.

15. Детектирование и сегментация областей ишемии в изображениях сердца — помощь ИИ в кардиологических исследованиях.

16. Сегментация и измерение размеров органов при ультразвуковом обследовании — точное выделение печени, почек и других органов.

ПРАКТИЧЕСКАЯ РАБОТА №5

Контрольное тестирование

Целью работы является закрепление теоретических знаний в области сегментации и детектирования объектов на медицинских изображениях.

Требования к выполнению работы

1. на выполнение работы отводится одна пара;
2. проходной бал для выполнения тестирования – 15 баллов.

Задание

1. Что делает сегментация

1. Подразделяет изображения на составляющие его области или объекты	2. Создаёт структурированный вид картинки для последующей обработки

2. К основным задачам обработки изображений относятся:

1. Удаление лишних объектов	2. Фильтрация изображения
3. Обнаружение объектов	4. Изменение параметров изображения

3. Соотнесите фильтр с его разновидностью

1.Сглаживающий	А. Линейные
2.Гауссовский	
3.Контрастоповышающий	
4.Медианный	Б. Не линейные
5.Фильтр размытия	

4. Соотнесите термин и определение

1. Классификация	А.Предсказание класса объекта на изображении.
2.Локализация	Б.Поиск объекта на изображении и отображение местоположения этого объекта.
3.Сегментация экземпляров	В.Детектирование объектов определенных классов на изображении и их идентификация.
4.Семантическая сегментация	Г.Детектирование объектов на изображении и группирование их, основываясь на определенных категориях.
5.Детекция	Д.Обнаружение объектов.

5. Что является основной идеей алгоритма R-CNN?

1. Использование скользящих окон для обнаружения объектов на изображении	2. Применение свёрточных нейронных сетей для классификации и регрессии ограничивающих рамок
3. Селективный поиск «регионов» — прямоугольных рамок для обнаружения объектов.	4. Использование аффинных преобразований для адаптации размеров регионов к входу CNN.

6. Вставьте пропуск

[] — это тип алгоритма глубокого обучения, применяемого для обработки изображений, который имитирует поведение взаимосвязанных нейронов человеческого мозга.

7. Соотнесите понятия

1. Свёрточный слой	А. выполняет операции свёртки и субдискретизации для извлечения признаков из входных данных.
2. Пулинг	Б. уменьшает размерность карт признаков, выбирая максимальные значения из определённых областей.
3. Нормализация	В. нормализует веса и смещения каждого нейрона в пределах одного мини-пакета.
4. Полносвязный слой	Г. соединяет каждый нейрон предыдущего слоя с каждым нейроном следующего слоя, обеспечивая взаимодействие между ними.

8. Что такое паддинг?

1. Метод нормализации данных	2. Метод улучшения качества распознавания образов
3. Метод уменьшения размерности данных	4. Способ увеличения или сохранения размера изображения

9. Какие значения получатся, если применить к матрице операцию max-pooling 2x2, паддинг 0, шаг 2?

56	2	6	3 2
9	96	6	7 5
76	26	3	9 6
86	91	13	8

1. 96, 75 91, 96	2. 2, 6 26, 3
3. 41, 63 70, 30	4. 56, 6 86, 8

10. Какие значения получатся, если применить к матрице операцию min-pooling 2x2, паддинг 0, шаг 2?

56	2	6	3 2
9	96	6	7 5
76	26	3	9 6
86	91	13	8

1. 96, 75 91, 96	2. 2, 6 26, 3
3. 41, 63 70, 30	4. 56, 6 86, 8

11. Какие значения получатся, если применить к матрице операцию average-pooling 2x2, паддинг 0, шаг 2?

56	2	6	3 2
9	96	6	7 5
76	26	3	9 6
86	91	13	8

1. 96, 75 91, 96	2. 2, 6 26, 3
3. 41, 63 70, 30	4. 56, 6 86, 8

12. Вставьте пропуск

[] — это процесс аннотирования данных для их последующего использования в машинном обучении. В случае изображений, он включает в себя указание границ объектов, классификацию объектов и другие метки.

13. Какой метод сегментации изображений основан на выборе порогового значения для разделения изображения на две или более областей?

1. Метод водораздела	2. Пороговая сегментация
3. Метод на основе регионов	4. Метод сегментации на основе границ

14. Что является основным принципом метода наращивания областей?

1. Использование текстурных признаков для определения границ областей	2. Применение цепей Маркова для определения вероятности перехода между областями
3. Последовательное наращивание областей вокруг выбранных элементов с сохранением однородности	4. Слияние мелких фрагментов изображения после наращивания областей

15. Какие методы используются для устранения шумов в сегментации изображений?

1. Гауссовское размытие	2. Медианный фильтр
3. Линейное усреднение пикселей	4. Двусторонний фильтр

10. Какие типы рекуррентных нейронных сетей (RNN) существуют?

1. Долговременная кратковременная память (LSTM)	2. Свёрточные нейронные сети (CNN)
3. Stacked	4. Gated Recurrent Units (GRU)

16. На чем в первую очередь сосредоточено детектирование объектов в компьютерном зрении?

1. разработка новых программных приложений	2. улучшение качества изображения с помощью фильтров
3. создание художественных визуальных эффектов на изображениях	4. определение конкретных объектов и отслеживание их экземпляров

17. Какая технология обычно используется при обнаружении объектов для их идентификации на основе визуальных признаков?

1. Компьютерное зрение	2. Обработка изображений
2. Искусственный интеллект	4. Сегментация данных

17. Какие методы используются при обнаружении объектов для повышения производительности и точности?

1. Облачные вычисления, граничные вычисления, туманные вычисления	2. Анализ тональности, преобразование текста в речь, алгоритмы перевода
3. Дополненная реальность, виртуальная реальность, 3D моделирование	4. Методы предложения регионов, предложения объектов, подходы, основанные на регрессии

18. С какими сложностями сталкиваются существующие алгоритмы обнаружения объектов в медицине?

1. Работа в реальном времени	2. Производительность
3. Масштабируемость	4. Обобщение

19. Семантическая сегментация возвращает ограничивающий прямоугольник для каждого найденного объекта.

1. Верно	2. Неверно

20. Для уменьшения шума используется

1. Сглаживание изображения	2. Контурирование изображения
3. Усиление резкости	4. Распознавание изображения

21. Какова основная особенность алгоритма YOLO (You Only Look Once) для обнаружения объектов на изображениях?

1. Он разбивает изображение на сетку и определяет объекты в каждом сегменте одновременно.	2. Он сканирует изображение построчно, определяя объекты по горизонтали.
3. Он использует метод рекурсивного поиска для идентификации объектов.	4. Он распознает объекты, анализируя только цветовые характеристики.

22. Вставьте пропуск

Метод, используемый для выделения пикселей, относящихся к определённой структуре или объекту на изображении, называется [].

23. Какая основная цель сегментации экземпляров?

1. Билинейная интерполяция	2. Квантование ближайшего соседа
3. Использование результатов детектирования объектов	4. Сегментация каждого экземпляра одного и того же класса по отдельности

24. В чем разница между RoiPool и RoiAlign?

1. RoiPool объединяет более быстрые R-CNN и FCN, в то время как RoiAlign использует квантование ближайшего соседа	2. RoiPool сегментирует экземпляры по отдельности, в то время как RoIAlign использует результаты обнаружения объекта
3. RoiPool использует квантование по методу ближайшего соседа, а RoiAlign – билинейную интерполяцию	

25. Какой метод сегментации применяется для разделения изображений на основе резких изменений яркости между соседними пикселями?

1. Метод растущей области	2. Метод на основе градиента
3. Метод пороговой обработки	4. Метод кластеризации

26. Какая метрика часто используется для оценки точности сегментации и измеряет пересечение и объединение областей объекта и предсказания?

1. Accuracy	2. Recall
3. Dice коэффициент	4. Loss

27. Какова цель RoIAlign в компьютерном зрении?

1. RoIAlign используется для удаления шумов на изображении перед сегментацией.	2. Предсказать метку класса и ограничивающую рамка каждого RoI
3. Извлечение карт признаков фиксированного размеров из каждого региона	4. Уменьшение разрешения карт признаков в соответствие с размером каждого региона

28. Какая роль у RoIAlign в методе Mask R-CNN?

1. Предсказать маску сегментации каждого региона	2. Объединить признаки из каждого региона в выходную сетку фиксированного размера
3. Предсказать метку класса и ограничивающую рамку каждого региона	4. Извлечь карты признаков фиксированного размера каждого региона

29. Что прогнозирует Mask R-CNN для каждого RoI?

1. Размер и положение	2. Ориентацию и текстуру
3. Цвет и яркость	4. Метку класса и ограничивающую рамка

30. Что из перечисленного не является этапом предобработки медицинских изображений перед сегментацией?

1. Фильтрация шума	2. Нормализация интенсивности
3. Детектирование контуров	4. Линейная интеполяция

31. Чем архитектура YOLO отличается от традиционного подхода к обнаружению объектов?

1. Использует сверточную нейронную сеть для прогнозирования ограничивающих рамок и вероятностей классов	2. Использует рекуррентные нейронные сети
3.не использует нейронные сети	4. Формулирует проблему как задачу классификации

32. Какой метод особенно эффективен для сегментации неоднородных структур, таких как опухоли?

1. Пороговая обработка	2. Метод растущей области
3. Метод активных контуров	4. Простое сглаживание

33. В какой форме YOLO прогнозирует ограничивающую рамку объекта для каждой ячейки сетки?

1. [x1, x2, y1, y2]	2. [hx, hy, hw, hh]
3. [pc, bx, by, bh, bw, c1, c2]	4. [mx, my, mw, mh]

34. Вставьте пропуск

[] - это доля правильных ответов модели среди всех предсказаний.

35. Соотнесите термин со значением

1. True Positive (TP)	А. Объект принадлежит классу и был распознан как принадлежащий.
2. True Negative (TN)	Б. Объект не принадлежит классу и был распознан как не принадлежащий.
3. False Positive (FP)	В. Объект не принадлежит классу, но был распознан как принадлежащий.
4. False Negative (FN)	Г. Объект принадлежит классу, но был распознан как не принадлежащий.

36. Какой метод детектирования объектов используется для локализации объектов определенного класса, таких как опухоли или повреждения тканей?

1. Локальная фильтрация	2. Градиентная свертка
3. Обнаружение объектов на основе R-CNN	4. Сегментация методом растущей области

37. Соотнесите архитектуру и преимущества

1. DeepLab	A. превосходит в точности границ и детализации благодаря atrous convolutions и CRFs, но требует больше ресурсов и сложен в настройке.
2. PSPNet	B. лучше учитывает глобальный контекст и подходит для сегментации больших сцен, но может уступать в детализации и точности границ.
3. U-Net	C. прост в реализации и эффективен при ограниченных ресурсах, хорошо подходит для медицинской сегментации и небольших объемов данных, но ограничен в захвате глобального контекста.

38. Какой метод сегментации чаще всего используется для выделения областей опухоли на медицинских изображениях, учитывая необходимость точного разделения смежных тканей?

1. Метод ближайшего соседа	2. Метод пороговой фильтрации
3. Активные контуры	4. Линейная регрессия

39. Какой из методов сегментации включает в себя разделение изображения на регионы на основе сходства пикселей?

1. К-средние	2. Метод градиента
3. Метод пороговой фильтрации	4. Метод линейной регрессии

40. Как называется техника, применяемая для увеличения разнообразия данных, которая особенно полезна при работе с ограниченным набором медицинских изображений?

1. Аугментация данных	2. Снижение размерности
3. Байесовская фильтрация	4. Принудительное шумоподавление