МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «КубГУ»)

Факультет компьютерных технологий и прикладной математики
Кафедра прикладной математики

Допустить к защите
И.о. заведующего кафедрой
канд. физ.-мат. наук, доц.
_____ А.В. Письменский
13. 06. 2024 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
(БАКАЛАВРСКАЯ РАБОТА)

ВЫЯВЛЕНИЕ ЗЛОКАЧЕСТВЕННЫХ ФОРМ РАКА КОЖИ С ПОМОЩЬЮ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Работу выполнил _____ А.И. Анастасиади
(подпись)

Направление подготовки 01.03.02 «Прикладная математика и информатика»

Направленность Математическое и информационное обеспечение экономической деятельности

Научный руководитель
канд. физ.-мат. наук, доц. _____ Н.О. Чубырь
(подпись)

Нормоконтролер
преподаватель _____ Е.В. Горбачева
(подпись)

Краснодар
2024

РЕФЕРАТ

Выпускная квалификационная работа 40 с., 33 рис., 8 источн., 1 прил.

НЕЙРОННЫЕ СЕТИ, КЛАССИФИКАЦИЯ, РАК КОЖИ, МЕЛАНОМА, МОДЕЛЬ, ЭФФЕКТИВНОСТЬ, ISIC, СНИМОК РАКА

Цель работы: изучить понятия, связанные с раком кожи, создать и обучить 3 различные модели компьютерного зрения, оценить производительность трех различных моделей компьютерного зрения, выбрать наилучшую из них.

В процессе работы изучены: основные понятия, связанные с раком кожи, современные методы диагностики рака, предыдущие исследования в данной области, предобработка данных, стратегии обучения и оценка моделей.

СОДЕРЖАНИЕ

Введение.....3
1 Теоретические положения о раке кожи.....5
1.1 Основные понятия и определения.....6
1.2 Современные методы диагностики рака кожи.....7
1.3 Искусственный интеллект в медицинской диагностике.....8
1.4 Предыдущие исследования по выявлению рака кожи.....9
2 Методика исследования.....12
2.1 Описание набора данных ISIC.....12
2.2 Предобработка данных.....13
2.3 Выбор и описание моделей компьютерного зрения.....18
3 Создание моделей.....21
3.1 Обучение CNN.....21
3.2 Обучение EfficientNet V2.....24
3.3 Обучение Vision Transformer.....26
4 Оценка эффективности моделей.....30
Заключение.....35
Список используемых источников.....37
Приложение А Кривая ROC для различных трех моделей.....38

ВВЕДЕНИЕ

Рак кожи – одна из наиболее распространенных форм рака среди населения. Он возникает из-за аномального роста кожных клеток, чаще всего на участках кожи, подверженных воздействию солнца. Этот вид рака может иметь серьезные последствия, включая смертельные исходы, если не выявляется и не лечится своевременно.

В наше время искусственный интеллект приобретает всё большую и большую популярность. Области его применения обширны, начиная с регулирования движения транспортных средств на дорогах и заканчивая космическими разработками.

Использование искусственного интеллекта в медицине становится все более актуальным и эффективным. В области диагностики рака кожи, где точность и скорость являются критически важными, системы компьютерного зрения, обученные на больших наборах данных, могут значительно улучшить процесс выявления заболевания.

Исходя из выше названного, цель данного исследования заключается в разработке и оценке производительности трех различных моделей компьютерного зрения для точного различия между злокачественными и доброкачественными формами рака кожи.

Основные 9 задач исследования включают в себя:

1) разобраться в основных понятиях и определениях, связанных с раком кожи,

2) рассмотреть современные методы диагностики рака кожи,

3) изучить предыдущие исследования по выявлению рака кожи с использованием компьютерного зрения и нейронных сетей,

4) подготовить корректные данные для исследования,

5) выбрать и описать модели компьютерного зрения,

6) обучить выбранные модели на подготовленном датасете,

7) оценить результаты работы каждой модели,

8) сравнить модели между собой,

9) выбрать наиболее эффективную модель для дальнейшего применения в клинической практике.

1 Теоретические положения о раке кожи

Рак кожи – патологический процесс или состояние, когда клетки эпителия начинают вести себя необычно. Это может произойти у людей любого возраста, особенно у тех, кто много времени проводит под солнцем. Лечение этого типа рака включает удаление опухоли и применение химиотерапии или радиотерапии.

Врачи отмечают устойчивый рост числа людей, страдающих от злокачественных поражений кожных покровов. Наблюдается динамика, а также из показания врачей, что количество людей, которые страдают от злокачественных поражений кожи, увеличивается с каждым годом. Данный показатель увеличивается на 4–5%.

Среди всех видов рака кожи, базальноклеточный рак самый распространенный. Он составляет около 75% всех случаев. Плоскоклеточный рак следующий по распространенности, встречается в 20% случаев.

Ученые считают, что избыточное воздействие солнечных лучей является главной причиной развития рака кожи. Эта теория подтверждается историей заболевания у больных. Большинство случаев рака кожи развиваются на открытых участках кожи. Люди с определенными типами кожи более подвержены риску.

Химические вещества также играют роль в формировании раковых клеток. Постоянный контакт с канцерогенами может привести к раку кожи на лице, шее, руках и теле. Ионизирующее излучение и высокие температуры также повышают риск.

Рак кожи часто развивается на месте ожогов или нарушений пигментации кожи. Иногда онкологи обнаруживают рак на фоне изменения обычных родинок.

1.1 Основные понятия

Рак кожи – это злокачественное новообразование, которое возникает из кожных клеток. Он обычно развивается на участках кожи, подвергшихся воздействию солнечных лучей, но также может возникать на других участках на коже. Есть несколько видов рака кожи, такие как: базально-клеточный рак, плоскоклеточный рак и меланому.

Базально-клеточный рак – наиболее встречающийся тип рака кожи. Другое название – базалиома. Это злокачественная опухоль кожи, которая очень медленно растёт и разрушает все окружающие её ткани. Обычно она появляется на голове и шее.

Плоскоклеточный рак – форма рака, которая является наследственной. Такой вид рака берет свое начало в плоских кожных клетках и случается в ходе долгого накопительного действия ультрафиолета. Плоскоклеточный рак выглядит по-разному, но в основном как красные или плоские бляшки, язвы, которые не заживают.

Меланома – очень опасная болезнь, в основе которой лежит проявляющийся рост злокачественной опухоли обычно в районе кожи и слизистых оболочек. Визуально её можно найти в виде новых родинок либо немного измененных старых, у которых присутствуют неправильные неровные края.

Искусственный интеллект, если говорить своими словами, это такая наука и технология, при которой создаются интеллектуальные программы и системы, которые готовы выполнять такие задачи, которым в основном бы потребовался человеческий интеллект. Ключевыми технологиями ИИ являются: машинное обучение, обработка естественного языка, глубокое обучение, компьютерное зрение и другие [1].

Компьютерное зрение – это теория и способы создания машин, выполняющих различные функции, например, обнаружение, отслеживание и классификацию объектов. Благодаря компьютерному зрению могут быть

разработаны различные системы, такие как: управление процессами, видеонаблюдение, моделирование объектов и так далее. Оно достаточно широко используется в медицинской диагностике различных заболеваний, включая и непосредственно рак кожи.

Нейронные сети обрабатывают запрос, исходящий от пользователя, проходя через определенные слои собственных нейронов и сопоставляя и анализируя данные благодаря весам и смещениям. Нейронные сети используются достаточно часто в машинном обучении, чтобы распознать паттерны и обработать входящую информацию.

1.2 Современные методы диагностики рака кожи

Благодаря современной медицине можно продиагносцировать рак кожи с помощью различных методов, которые также применяются совместно. В данной работе рассмотрим основные из них.

При визуальном осмотре специализированный врач внимательным образом рассматривает кожу пациента. При обнаружении признаков асимметрии, неровных краев, неоднородности цвета или новых образований производятся дальнейшие исследования. А клиент ставится в таком случае на учет в поликлинике, в которую он обратился. В случае необходимости, врач направляет человека сделать нужные анализы.

Дермоскопия – в данном методе используется специальный врачебный инструмент – дермоскоп, благодаря которому возможно увеличение изображения кожи и рассмотрение ее в деталях. На подозрительный участок кожи наносится специальная смазка, далее дермоскопом производится съемка, ну а после на монитор выводятся данные, которые считал компьютер на основании данного целевого участка. Этот метод полезен тем, что он достаточно точно оценивает структуру и цвет образования на коже. Это помогает в дифференциальной диагностике отличать доброкачественные и злокачественные образования.

Биопсия – это процедура, когда врач изымает образец такни кожи для внимательного дальнейшего изучения в лаборатории. Под микроскопом с достаточно большой точностью определяются раковые клетки в данном методе.

Оптическая когерентная томография (ОКТ) – диагностическое оборудование для создания снимков и изображений, которое использует лазерное излучение, чтобы создать срез ткани с очень высоким разрешением. Благодаря ОКТ можно рассматривать более глубокие слои кожи и искать уже изменения, которые связаны с раком, уже там.

Массивные многоспектральные анализаторы – это такие устройства для анализа кожи и выявления аномалий, связанных с ней. Их механизм основан на использовании различных длин волн. К их помощи обращаются в основном на ранних стадиях. ММА дают быструю оценку состояния кожи и не требуют лечебных вмешательств.

И поэтому, исходя из вышеперечисленных методов, напрашивается вывод, что необходимо применение искусственного интеллекта хотя бы с той целью, чтобы увеличить точность, эффективность и автоматизировать процесс диагностики рака кожи. Системы, связанные с искусственным интеллектом, однозначно помогут врачам принимать решения с высокой точностью по поводу определения злокачественных образований на коже.

1.3 Искусственный интеллект в медицинской диагностике

Искусственный интеллект (ИИ) стал очень мощным инструментом в области медицинской диагностики. Благодаря нему появляется возможность анализировать большие объемы данных и выявлять скрытые паттерны и зависимости [2]. Рассмотрим конкретные плюсы применения ИИ в диагностике кожных заболеваний в медицине.

Автоматизация процесса диагностики может непосредственно быть произведена искусственным интеллектом. Он анализирует конкретные

снимки у пациентов, которых есть подозрения на злокачественное образование, результаты анализов и, в целом, абсолютно любые данные с достаточно высокой скоростью, эффективностью и точностью. Поэтому можно намного быстрее, нежели обычными методами диагностики, выявить заболевание.

Обычно, при серьезных и качественных исследованиях, в ИИ используются большие наборы данных. Но размер набора не всегда самое главное. Важно, чтобы он был подготовлен на фактических данных, которые уже в свое время проходили проверку. И благодаря этому выявляются очень тонкие признаки и паттерны, которые с большой легкостью могут быть незаметны для человеческого глаза. Это всё увеличивает точность и уменьшает вероятность ошибки на реальных данных.

Анализ данных с использованием интеллектуальных систем позволяет находить индивидуальные особенности пациентов. А в соответствии с этим уже можно использовать уникальные персонализированные подходы в лечении пациента.

Если оптимизировать рабочие процессы в здравоохранении по всей стране, то будет намного лучше и легче обеим сторонам. На медицинский персонал уменьшится нагрузка, а пациентам нужно будет меньше ждать в очередях результатов осмотра или обследования.

Также, опять же благодаря большим объемам данных, интеллектуальные системы способны обнаруживать заболевание на ранних стадиях, даже в тот период, когда ещё нет симптомов и предпосылок к заболеванию или злокачественному образованию на конкретных участках кожи.

1.4 Предыдущие исследования по выявлению рака кожи

Если говорить о последних исследованиях, проведенных в области выявления рака кожи с использованием компьютерного зрения, то можно

сделать вывод, что у данной темы огромный, чтобы повысить эффективность и точность диагностики. Есть важные 4 исследования американских американских специалистов в данной области:

1) Исследование "Automated Melanoma Recognition in Dermoscopy Images via Very Deep Residual Networks" (2017). Данное исследование предложило метод для автоматического определения меланомы на снимках, сделанных при помощи дермоскопа с использованием глубоких сверточных нейронных сетей, таких как Very Deep Residual Networks.

Авторы показали высокую точность и чувствительность предложенного метода в сравнении с традиционными методами диагностики.

2) Исследование "Deep Learning for Image-Based Melanoma Detection" (2018). В этом исследовании была предложена система глубокого обучения для обнаружения меланомы на изображениях кожи. Авторы использовали нейронные сети, такие как Inception и ResNet, для анализа изображений и демонстрировали высокую точность и специфичность их метода на наборах данных меланомы.

3) Исследование "Automated Detection of Malignant Melanoma with Deep Learning Methods" (2019). В этом исследовании авторы применили глубокие нейронные сети к задаче автоматического обнаружения злокачественной меланомы на изображениях кожи. Они использовали архитектуры, такие как DenseNet и ResNet, и продемонстрировали высокую точность и чувствительность своего метода в выявлении меланомы.

4) Исследование "Skin Cancer Detection Using Convolutional Neural Networks: Systematic Review" (2020). Это исследование провело систематический обзор существующих работ по диагностике рака кожи с использованием сверточных нейронных сетей. Авторы проанализировали различные методы и архитектуры нейронных сетей, применяемые в этой области, и обсудили их преимущества и ограничения.

То, что касается отечественной разработки, ученые Северо-Кавказского федерального университета разработали нейросетевую систему

распознавания по десяти диагностическим категориям рака кожи, точность которой составила 82,5 процента. Этот показатель на конец 2023 года является самым высоким в мире в данной области.

2 Методика исследования

2.1 Описание набора данных ISIC

Набор данных International Skin Imaging Collaboration (ISIC) представляет собой крупный набор изображений кожных образований, собранных в рамках международного сотрудничества с целью улучшения методов диагностики рака кожи. Этот набор данных содержит разнообразные типы кожных образований, включая как доброкачественные, так и злокачественные опухоли, такие как базально-клеточный рак, плоскоклеточный рак и меланому.

Основные 5 характеристик набора данных ISIC включают в себя:

1) Разнообразие типов образований

Набор данных содержит различные типы кожных образований, что обеспечивает широкий спектр случаев для обучения моделей.

2) Фотографии дермоскопии и высокого разрешения

Изображения в наборе данных включают как фотографии дермоскопии, полученные с помощью специализированного оборудования для изучения кожи, так и высоко разрешённые изображения, что позволяет использовать различные подходы для анализа данных.

3) Медицинская аннотация

Большинство изображений в наборе данных имеют медицинскую аннотацию, включая диагноз, что облегчает обучение моделей с учителем.

4) Большой объем данных

Набор данных ISIC включает в себя большое количество изображений, что позволяет обучать модели на больших наборах данных для повышения их обобщающей способности.

5) Доступность на Kaggle

Набор данных ISIC доступен всем пользователям на платформе Kaggle.

Использование набора данных ISIC обеспечивает возможность обучения моделей на реальных данных кожных образований, что позволяет разрабатывать и оценивать методы выявления рака кожи с использованием компьютерного зрения с высокой достоверностью.

2.2 Предобработка данных

Предобработка данных играет важную роль в успешном обучении моделей компьютерного зрения для выявления рака кожи. Включает в себя несколько этапов, включая масштабирование, выравнивание размеров изображений и удаление шумов или артефактов [3]. Основные 6 шагов предобработки данных включают в себя:

1) Создание датафрейма

DateFrame – один из главных инструментов Pandas. Он представляет собой таблицу с заданным количеством строк и столбцов. Ячейки данной таблицы могут хранить самые различные типы данных, такие как числовые, строки, булевы и тому подобные.

Ввод [17]:
```
def generate_labels(image_paths):
    return [_.split('/')[-2:][0] for _ in image_paths]


def build_df(image_paths, labels):
    df = pd.DataFrame({
        'image_path': image_paths,
        'label': generate_labels(labels)
    })

    df['label_encoded'] = df.apply(lambda row: 0 if row.label == 'malignant' else 1, axis=1)

    return df.sample(frac=1, random_state=CFG.SEED).reset_index(drop=True)
```

Ввод [18]:
```
train_df = build_df(train_images, generate_labels(train_images))
test_df = build_df(test_images, generate_labels(test_images))
```

Ввод [19]:
```
train_df.head(5)
```

Out[19]:

	image_path	label	label_encoded
0	/content/train/benign/793.jpg	benign	1
1	/content/train/benign/723.jpg	benign	1
2	/content/train/malignant/673.jpg	malignant	0
3	/content/train/malignant/370.jpg	malignant	0
4	/content/train/benign/490.jpg	benign	1

Рисунок 1 – Создание DateFrame

На рисунке 1 определены функции для генерации меток и построения датафрейма на основе списков путей к изображениям и соответствующих меток. Далее происходит создание датафрейма для обучающего и тестового наборов.

2) Масштабирование изображений

Изображения в наборе данных могут иметь разные разрешения, что может затруднить обучение моделей. Поэтому изображения должны быть масштабированы до определенного размера. Был взят стандартный размера 224x224 пикселей, чтобы обеспечить единое представление данных для модели.

3) Удаление шумов и артефактов

Некоторые изображения могут содержать шумы, артефакты или неинформативные части, которые могут негативно повлиять на процесс обучения моделей. Поэтому такие шумы и артефакты могут быть удалены или смягчены с помощью фильтров или других методов обработки изображений.

4) Нормализация

Для обеспечения стабильности обучения и улучшения сходимости моделей может быть проведена нормализация значений пикселей изображений [4]. Например, значения пикселей могут быть масштабированы до интервала [0, 1] путем деления на 255 (максимальное значение интенсивности пикселя).

Ввод [20]:
```python
def _load(image_path):
    # Чтение и декодирование файла изображения в тензор uint8
    image = tf.io.read_file(image_path)
    image = tf.io.decode_jpeg(image, channels=3)

    # Изменение размера изображения
    image = tf.image.resize(image, [CFG.HEIGHT, CFG.WIDTH],
                            method=tf.image.ResizeMethod.LANCZOS3)

    # Преобразуем dtype изображения в float32
    image = tf.cast(image, tf.float32)/255.

    # Возвращаем изображение
    return image

def view_sample(image, label, color_map='rgb', fig_size=(8, 10)):
    plt.figure(figsize=fig_size)

    if color_map=='rgb':
        plt.imshow(image)
    else:
        plt.imshow(tf.image.rgb_to_grayscale(image), cmap=color_map)

    plt.title(f'Label: {label}', fontsize=16)
    return
```

Рисунок 2 – Нормализация изображений

Функция из рисунка 2 принимает путь к изображению, затем происходит считывание содержимого файла изображения, декодирование файла изображения в тензор с тремя каналами (RGB), преобразование типа данных изображения в `float32` и нормализация значений пикселей в диапазоне от 0 до 1. После этого возвращается полученный тензор изображения.

5) Разделение на обучающий и тестовый наборы

Для оценки производительности моделей данные могут быть разделены на обучающий и тестовый наборы. Обучающий набор используется для обучения модели, а тестовый - для окончательной оценки производительности модели.

Было использовано 3297 фотографий доброкачественных(benign) и злокачественных(malignant) опухолей. В качестве тестового набора было взято 20% общей выборки, все остальные изображения использовались для обучающего.

Ввод [16]:
```
# Получение размеров обучающего и тестового наборов
train_size = len(train_images)
test_size = len(test_images)

# Получение размера набора данных
total = train_size + test_size

# Просмотр количества образцов
print(f'train:\t\t{train_size}')
print(f'test:\t\t{test_size}')
print('====================')
print(f'ИТОГО:\t\t{total}')
```

```
train:          2637
test:           660
====================
ИТОГО:          3297
```

Рисунок 3 – Обучающая и тестовая выборки

6) Дополнительные шаги

В зависимости от специфики данных и требований задачи могут быть применены и другие методы предобработки, такие как аугментация данных (для увеличения разнообразия данных, коррекция яркости и контраста и так далее).

Ввод [27]:
```
augmentation_layer = Sequential([
    layers.RandomFlip(mode='horizontal_and_vertical', seed=CFG.TF_SEED),
    layers.RandomZoom(height_factor=(-0.1, 0.1), width_factor=(-0.1, 0.1), seed=CFG.TF_SEED),
], name='augmentation_layer')
```

Ввод [28]:
```
image = tf.image.rgb_to_grayscale(sample_image)

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 10))

fig.tight_layout(pad=6.0)

ax1.set_title('Исходное изображение', fontsize=20)
ax1.imshow(image, cmap='inferno');

ax2.set_title('Дополненное изображение', fontsize=20)
ax2.imshow(augmentation_layer(image), cmap='inferno');
```

Рисунок 4 – Аугментация данных

На рисунке 4 создается слой аугментации, используя Sequential из TensorFlow. В данном случае, применяются случайные горизонтальные и вертикальные отражения, а также случайное масштабирование(zoom) изображения.

Слой увеличения генерирует дополнения, которые немного отличаются от исходных изображений. Это сделано специально, поскольку необходимо стремиться генерировать дополненные изображения, аналогичные изображениям исходного набора данных, сохраняя при этом ключевые особенности изображений. Пример данных изображений отображен на рисунке ниже.

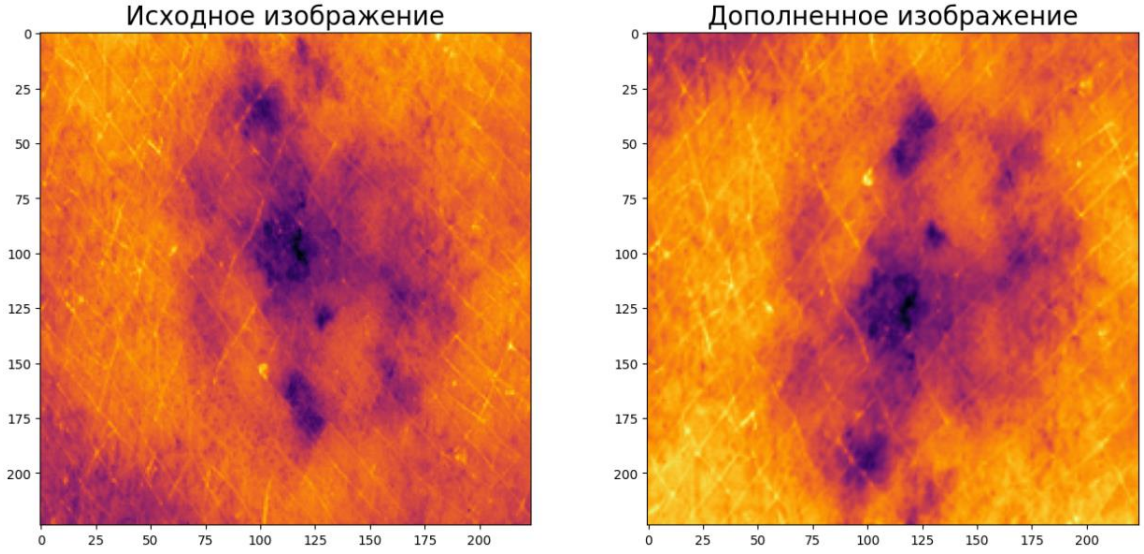

Рисунок 5 – Пример аугментации данных

Предобработка данных является важным этапом подготовки данных для обучения моделей и может существенно повлиять на их производительность и обобщающую способность. Корректно проведенная предобработка данных способствует улучшению качества и надежности моделей компьютерного зрения для выявления рака кожи.

Таким образом, фотографии в датафрейме выглядят так, как показано на рисунке 6.

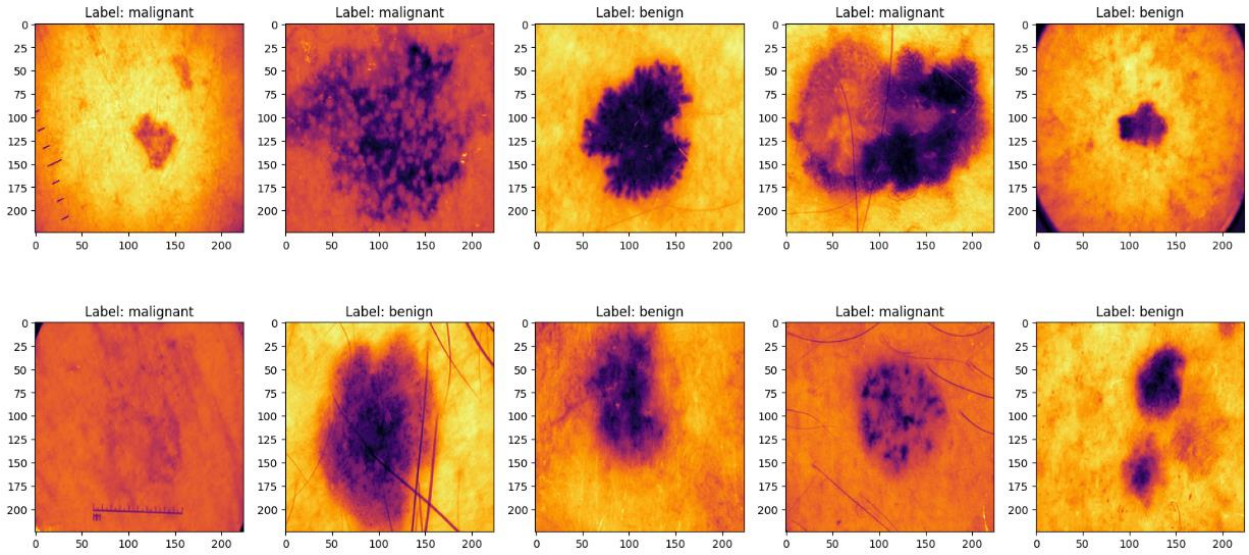

Рисунок 6 – Визуализация случайной выборки

2.3 Выбор и описание моделей компьютерного зрения

В данном исследовании рассматриваются три различные модели компьютерного зрения для задачи выявления злокачественных форм рака кожи: базовая сверточная нейронная сеть (CNN), предварительно обученная EfficientNet V2 B0 и предварительно обученный Vision Transformer. Каждая из этих трех моделей имеет свои уникальные характеристики и преимущества.

1) Базовая сверточная нейронная сеть (CNN) - это классическая архитектура нейронной сети для анализа изображений. Она состоит из нескольких слоев свертки, пулинга и полносвязных слоев, которые последовательно обрабатывают входные изображения и извлекают из них признаки. Проста в реализации и понимании.

CNN способны определять взаимосвязи в данных изображений, которые относятся к классификации и обнаруживать объекты. Тем самым они очень полезны для задач компьютерного зрения.

2) EfficientNet – можно сказать, это семейство продуктивных архитектур нейронных сетей, которые были разработаны для достижения высокой производительности при минимальном количестве параметров. Модель EfficientNet V2 B0 представляет собой одну из базовых архитектур этого

семейства и обучена на большом наборе данных ImageNet. Эффективное использование ресурсов и высокая производительность делают эту модель привлекательной для задачи выявления рака кожи.

Для разработки этих моделей авторы используют комбинацию поиска нейронной архитектуры с учетом обучения и масштабирования, чтобы совместно оптимизировать скорость обучения. Поиск моделей производился в пространстве поиска, обогащенном новыми функциями.

3) Предварительно обученный Vision Transformer - это инновационная архитектура нейронной сети, которая использует трансформеры, обычно применяемые в обработке естественного языка, для анализа визуальных данных. В отличие от CNN, которые основаны на операциях свертки, трансформеры обрабатывают входные данные путем взаимодействия между различными частями данных. Это позволяет модели Vision Transformer успешно использовать контекстуальные зависимости в изображениях для выявления паттернов и признаков.

Модель Vision Transformer (ViT) изначально была представлена в документе конференции, которая называлась "Изображение стоит 16x16 слов: трансформаторы для распознавания изображений в масштабе". Данный документ был опубликован на Международной конференции по обучению репрезентациям в 2021 году. ViTs применяются в задачах распознавания снимков и изображений, таких как обнаружение объектов, сегментация фотографий, их классификация и распознавание действий.

Трансформеры, которые используются в ViTs в основном применяются для того, чтобы обрабатывать естественный язык. Но с помощью данной модели можно входное изображение обработать в патч, к которому впоследствии легко обращаться. "Заплатки" используются вместе с преобразователем-кодировщиком для создания встраиваемых изображений. Блоки преобразователя-кодировщика состоят из трех компонентов, а именно:

1) Нормализация слоя: применяется к исправлениям и вниманию для ускорения вычислений,

2) Внимание с несколькими заголовками: используется для создания и объединения заголовков внимания для всех исправлений, чтобы закрепить как локальные, так и глобальные зависимости в снимках.

3) Многослойные персептроны: получают головки внимания и данные головки проходят через два плотных слоя с единицей измерения гауссовой ошибки (GELU), которая используется в качестве функции активации.

Выбор этих трех моделей позволяет оценить и сравнить их производительность в задаче выявления злокачественных форм рака кожи и определить наиболее эффективную модель для данной задачи.

3 Создание моделей

3.1 Обучение CNN

Для решения задач классификации CNN включают полностью подключенную классификационную головку, которая использует извлеченные объекты, сгенерированные сверточными слоями, для выполнения задачи классификации входного изображения. В этом разделе мы будем использовать базовый CNN в качестве базовой модели для выявления рака кожи.

Ввод [32]:
```
def cnn_model():

    initializer = tf.keras.initializers.GlorotNormal()

    cnn_sequential = Sequential([
        layers.Input(shape=CFG.IMAGE_SIZE, dtype=tf.float32, name='input_image'),

        layers.Conv2D(16, kernel_size=3, activation='relu', kernel_initializer=initializer),
        layers.Conv2D(16, kernel_size=3, activation='relu', kernel_initializer=initializer),
        layers.MaxPool2D(pool_size=2, padding='valid'),

        layers.Conv2D(8, kernel_size=3, activation='relu', kernel_initializer=initializer),
        layers.Conv2D(8, kernel_size=3, activation='relu', kernel_initializer=initializer),
        layers.MaxPool2D(pool_size=2),

        layers.Flatten(),
        layers.Dropout(0.2),
        layers.Dense(128, activation='relu', kernel_initializer=initializer),
        layers.Dense(2, activation='sigmoid', kernel_initializer=initializer)
    ], name='cnn_sequential_model')

    return cnn_sequential
```

Рисунок 7 – Создание базовой CNN модели

На рисунке 7 в функции cnn_model() создается модель CNN с использованием API Sequential из TensorFlow. Модель состоит из нескольких слоев Conv2D (сверточных), MaxPool2D (пулинга), Flatten (плоского преобразования), Dropout (регуляризации), и Dense (полносвязного) слоев. Каждый Conv2D слой имеет функцию активации ReLU, а последний слой Dense имеет функцию активации Sigmoid для бинарной классификации.

Для обучения этой модели используется двоичная перекрестная энтропия в качестве функции потерь, поскольку это проблема классификации для двоичных меток. Что касается оптимизатора, будет использоваться оптимизатор Adam с 0,001 в качестве скорости обучения (по умолчанию) [5].

Чтобы предотвратить переобучение во время обучения, придется использовать API обратного вызова TensorFlow для реализации обратных вызовов EarlyStopping & ReduceLROnPlateau. Единственными показателями, которые будут отслеживаться во время обучения модели, будут показатели потерь и точности.

Ввод [35]:
```
def train_model(model, num_epochs, callbacks_list, tf_train_data,
                tf_valid_data=None, shuffling=False):
    model_history = {}

    if tf_valid_data != None:
        model_history = model.fit(tf_train_data,
                                  epochs=num_epochs,
                                  validation_data=tf_valid_data,
                                  validation_steps=int(len(tf_valid_data)),
                                  callbacks=callbacks_list,
                                  shuffle=shuffling)

    if tf_valid_data == None:
        model_history = model.fit(tf_train_data,
                                  epochs=num_epochs,
                                  callbacks=callbacks_list,
                                  shuffle=shuffling)
    return model_history
```

Ввод [36]:
```
early_stopping_callback = tf.keras.callbacks.EarlyStopping(
    monitor='val_loss',
    patience=3,
    restore_best_weights=True)

reduce_lr_callback = tf.keras.callbacks.ReduceLROnPlateau(
    monitor='val_loss',
    patience=2,
    factor=0.1,
    verbose=1)

CALLBACKS = [early_stopping_callback, reduce_lr_callback]
METRICS = ['accuracy']
```

Рисунок 8 – Обучение базовой CNN модели

На рисунке выше данная функция обучает модель TensorFlow и возвращает объект dict, содержащий данные истории 7 показателей модели:

1) model - модель для обучения,

2) num_epochs - количество эпох для обучения модели,

3) callbacks_list - список, содержащий функции обратного вызова для модели,

4) tf_train_data - набор данных для модели, на которой будет производиться обучение,

5) tf_valid_data - набор данных для модели, на которой будет проведена проверка (по умолчанию=Нет),

6) shuffling - условие для перетасовки данных, данные перетасовываются при значении True (по умолчанию=False),

7) model_history - словарь, содержащий значения потерь и метрик, отслеживаемые во время обучения.

Ввод [37]:
```
tf.random.set_seed(CFG.SEED)

model_cnn.compile(
    loss=tf.keras.losses.BinaryCrossentropy(),
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
    metrics=METRICS
)

print(f'Обучение {model_cnn.name}.')
print(f'Обучение на {len(train_new_df)} образцах, проверка на {len(val_df)} образцах.')
print('----------------------------------')

cnn_history = train_model(
    model_cnn, CFG.EPOCHS, CALLBACKS,
    train_ds, val_ds,
    shuffling=False
)
```

Рисунок 9 – Тренировка базовой CNN модели

На рисунке 9 определяется список метрик, которые будут использоваться для оценки модели (в данном случае - только точность accuracy).

Также происходит компилирование модели. В данном случае определяется функция потерь для бинарной классификации – BinaryCrossentropy, в качестве оптимизатора выступает Adam с указанным коэффициентом скорости обучения и метрика точности для оценки модели.

Далее происходит вызов функции train_model() для того, чтобы обучить модель. В данной функции передаются такие параметры, как: модель, количество эпох, список обратных вызовов, обучающий и валидационный наборы данных.

```
Epoch 7/10
71/71 [==============================] - 19s 274ms/step - loss: 0.4040 - accuracy: 0.7988 - val_loss: 0.3956 - val_accuracy: 0.
8056 - lr: 0.0010
Epoch 8/10
71/71 [==============================] - 20s 285ms/step - loss: 0.3960 - accuracy: 0.7988 - val_loss: 0.5214 - val_accuracy: 0.
7677 - lr: 0.0010
Epoch 9/10
70/71 [============================>.] - ETA: 0s - loss: 0.4595 - accuracy: 0.7884
Epoch 9: ReduceLROnPlateau reducing learning rate to 0.00010000000474974513.
71/71 [==============================] - 18s 255ms/step - loss: 0.4593 - accuracy: 0.7885 - val_loss: 0.4238 - val_accuracy: 0.
7955 - lr: 0.0010
Epoch 10/10
71/71 [==============================] - 19s 264ms/step - loss: 0.3809 - accuracy: 0.8175 - val_loss: 0.3758 - val_accuracy: 0.
8106 - lr: 1.0000e-04
```

Рисунок 10 – Обучение по эпохам для обучающего набора

```
Ввод [38]: cnn_evaluation = model_cnn.evaluate(test_ds)

           21/21 [==============================] - 3s 121ms/step - loss: 0.3625 - accuracy: 0.8121
```

Рисунок 11 – Проверка тестового набора

Модель CNN была способна сходиться к потере при тестировании, аналогичной той, что наблюдалась при обучении и валидации наборов данных. Также налюдается не очень высокая точность модели.

3.2 Обучение EfficientNet V2

```
Ввод [41]: def get_tfhub_model(model_link, model_name, model_trainable=False):
               return hub.KerasLayer(model_link,
                                     trainable=model_trainable,
                                     name=model_name)
```

```
Ввод [42]: efficientnet_v2_url = 'https://tfhub.dev/google/imagenet/efficientnet_v2_imagenet21k_b0/feature_vector/2'
           model_name = 'efficientnet_v2_b0'

           set_trainable=False

           efficientnet_v2_b0 = get_tfhub_model(efficientnet_v2_url,
                                                model_name,
                                                model_trainable=set_trainable)
```

Рисунок 12 – Создание модели

Функция get_tfhub_model принимает в качестве аргументов ссылку на модель из TensorFlow Hub, имя модели и параметр model_trainable, указывающий, должны ли обучаемые параметры модели быть доступны для обучения. В данном случае параметры не будут доступны для обучения. Она используется для загрузки модели из TensorFlow Hub.

Важным моментом является то, что создается своя сеть на основе efficientnet. Этот факт позволяет добиться лучших результатов, а также вносит новизну.

Ввод [43]:
```
def efficientnet_v2_model():

    initializer = tf.keras.initializers.GlorotNormal()

    efficientnet_v2_sequential = Sequential([
        layers.Input(shape=CFG.IMAGE_SIZE, dtype=tf.float32, name='input_image'),
        efficientnet_v2_b0,
        layers.Dropout(0.2),
        layers.Dense(128, activation='relu', kernel_initializer=initializer),
        layers.Dense(2, dtype=tf.float32, activation='sigmoid', kernel_initializer=initializer)
    ], name='efficientnet_v2_sequential_model')

    return efficientnet_v2_sequential
```

Рисунок 13 – Задание параметров модели

Функция на рисунке выше определяет модель на основе EfficientNetV2. Входной слой задается размером изображения CFG.IMAGE_SIZE. Затем добавляется загруженная модель efficientnet_v2_b0 как слой в модель. После этого следуют слои Dropout, Dense и выходной слой. В конце возвращается полученная модель.

Ввод [46]:
```
tf.random.set_seed(CFG.SEED)

model_efficientnet_v2.compile(
    loss=tf.keras.losses.BinaryCrossentropy(),
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
    metrics=METRICS
)

print(f'Обучение {model_efficientnet_v2.name}.')
print(f'Обучение на {len(train_new_df)} образцах, проверка на {len(val_df)} образцах.')
print('----------------------------------')

efficientnet_v2_history = train_model(
    model_efficientnet_v2, CFG.EPOCHS, CALLBACKS,
    train_ds, val_ds,
    shuffling=False
)
```

Рисунок 14 – Обучение EfficientNet V2 модели

На рисунке 14 происходит компилирование модели. В данном случае определяется функция потерь для бинарной классификации -

BinaryCrossentropy, в качестве оптимизатора выступает Adam с указанным коэффициентом скорости обучения и метрика точности для оценки модели [6].

Далее происходит вызов функции train_model() для того, чтобы обучить модель. В данной функции передаются такие параметры, как: модель, количество эпох, список обратных вызовов, обучающий и валидационный наборы данных.

```
Epoch 7: ReduceLROnPlateau reducing learning rate to 0.0001000000047497451.
71/71 [==============================] - 21s 296ms/step - loss: 0.2418 - accuracy: 0.8889 - val_loss: 0.2918 - val_accuracy: 0.
8687 - lr: 0.0010
Epoch 8/10
71/71 [==============================] - 20s 288ms/step - loss: 0.2046 - accuracy: 0.9108 - val_loss: 0.2750 - val_accuracy: 0.
8788 - lr: 1.0000e-04
Epoch 9/10
71/71 [==============================] - 19s 269ms/step - loss: 0.1984 - accuracy: 0.9139 - val_loss: 0.2758 - val_accuracy: 0.
8687 - lr: 1.0000e-04
Epoch 10/10
70/71 [============================>.] - ETA: 0s - loss: 0.1990 - accuracy: 0.9161
Epoch 10: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.
71/71 [==============================] - 20s 286ms/step - loss: 0.1989 - accuracy: 0.9161 - val_loss: 0.2779 - val_accuracy: 0.
8687 - lr: 1.0000e-04
```

Рисунок 15 – Обучение по эпохам для обучающего набора

```
Ввод [47]: efficientnet_v2_evaluation = model_efficientnet_v2.evaluate(test_ds)
           21/21 [==============================] - 3s 158ms/step - loss: 0.2855 - accuracy: 0.8652
```

Рисунок 16 – Проверка тестового набора

Как видно из рисунков выше, данная модель меньше подвержена потерям. Она также превосходит модель CNN за счет сходимости к меньшим потерям при тестировании и достижения более высокой точности.

3.3 Обучение Vision Transformer

```
Ввод [51]: !pip install tensorflow-addons
           !pip install vit-keras
```

Рисунок 17 – Подключение библиотек

```
Ввод [52]: from vit_keras import vit

vit_model = vit.vit_b16(
    image_size=224,
    activation='sigmoid',
    pretrained=True,
    include_top=False,
    pretrained_top=False,
    classes=2
)

for layer in vit_model.layers:
    layer.trainable = False
```

Рисунок 18 – Инициализация модели Vision Transformer

На рисунке выше импортируется модуль vit из библиотеки vit-keras, и инициализируется модель Vision Transformer (ViT) с помощью функции vit_b16(). Указываются параметры модели, такие как размер изображения, активация, предварительное обучение, включение верхнего слоя, предварительное обучение верхнего слоя и количество классов. В цикле ниже происходит замораживание всех слоев модели, чтобы веса не обновлялись во время обучения.

```
Ввод [53]: def vit_b16_model():

    initializer = tf.keras.initializers.GlorotNormal()

    vit_b16_sequential = Sequential([
        layers.Input(shape=CFG.IMAGE_SIZE, dtype=tf.float32, name='input_image'),
        vit_model,
        layers.Dropout(0.2),
        layers.Dense(128, activation='relu', kernel_initializer=initializer),
        layers.Dense(2, dtype=tf.float32, activation='sigmoid', kernel_initializer=initializer)
    ], name='vit_b16_sequential_model')

    return vit_b16_sequential
```

Рисунок 19 – Задание параметров модели

Функция на рисунке выше определяет модель ViT-B16. Входной слой задается размером изображения CFG.IMAGE_SIZE. Затем добавляется загруженная модель ViT как слой в модель. После этого следуют слои Dropout, Dense и выходной слой. В конце возвращается полученная модель.

Ввод [56]:
```
tf.random.set_seed(CFG.SEED)

model_vit_b16.compile(
    loss=tf.keras.losses.BinaryCrossentropy(),
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
    metrics=METRICS
)

print(f'Обучение {model_vit_b16.name}.')
print(f'Обучение на {len(train_new_df)} образцах, проверка на {len(val_df)} образцах.')
print('----------------------------------')

vit_b16_history = train_model(
    model_vit_b16, CFG.EPOCHS, CALLBACKS,
    train_ds, val_ds,
    shuffling=False
)
```

Рисунок 20 – Обучение ViT-B16 модели

На рисунке 20 происходит компилирование модели. В данном случае определяется функция потерь для бинарной классификации - BinaryCrossentropy, в качестве оптимизатора выступает Adam с указанным коэффициентом скорости обучения и метрика точности для оценки модели [6].

Далее происходит вызов функции train_model() для того, чтобы обучить модель. В данной функции передаются такие параметры, как: модель, количество эпох, список обратных вызовов, обучающий и валидационный наборы данных.

```
Epoch 6/10
71/71 [==============================] - 43s 612ms/step - loss: 0.2972 - accuracy: 0.8599 - val_loss: 0.2780 - val_accuracy: 0.
8763 - lr: 0.0010
Epoch 7/10
71/71 [==============================] - 44s 616ms/step - loss: 0.2745 - accuracy: 0.8693 - val_loss: 0.2782 - val_accuracy: 0.
8687 - lr: 0.0010
Epoch 8/10
71/71 [==============================] - 44s 613ms/step - loss: 0.2717 - accuracy: 0.8688 - val_loss: 0.2521 - val_accuracy: 0.
8813 - lr: 0.0010
Epoch 9/10
71/71 [==============================] - 43s 608ms/step - loss: 0.2545 - accuracy: 0.8876 - val_loss: 0.2593 - val_accuracy: 0.
8864 - lr: 0.0010
Epoch 10/10
71/71 [==============================] - 44s 613ms/step - loss: 0.2492 - accuracy: 0.8911 - val_loss: 0.2392 - val_accuracy: 0.
9015 - lr: 0.0010
```

Рисунок 21 – Обучение по эпохам для обучающего набора

Ввод [57]:
```
vit_b16_evaluation = model_vit_b16.evaluate(test_ds)
```
```
21/21 [==============================] - 9s 423ms/step - loss: 0.2749 - accuracy: 0.8803
```

Рисунок 22 – Проверка тестового набора

Модель Vision Transformer была способна привести к потере при тестировании, немного превышающей потерю при валидации, и существуют видимые различия между потерей при тестировании и потерей при обучении (возможно, произошла некоторая переобучаемость). Она также превосходит модель CNN за счет снижения потерь при тестировании и достижения более высокой точности. Однако, по-видимому, она уступает модели EfficientNet. Для обучения этой модели может потребоваться больше эпох.

4 Оценка эффективности моделей

Теперь, когда модель обучена на данных, нужно проверить, насколько хорошо она работает на невидимых тестовых данных. Чтобы провести эту проверку, необходимо оценить производительность модели на тестовых данных и записать показатели оценки. Поскольку это задача бинарной классификации, будут использоваться некоторые хорошо известные показатели классификации. Следовательно, нужно будет воспользоваться библиотекой Scikit Learn для проверки модели.

```
Ввод [71]: # CNN
           print(classification_report(test_df.label_encoded,
                                       cnn_test_predictions,
                                       target_names=class_names))
```

```
              precision    recall  f1-score   support

   malignant       0.73      0.94      0.82       300
      benign       0.93      0.71      0.80       360

    accuracy                           0.81       660
   macro avg       0.83      0.82      0.81       660
weighted avg       0.84      0.81      0.81       660
```

Рисунок 23 – Показатели классификации для CNN

Чтобы оценить качество каждой из моделей, я взял 3 основных показателя:

1) Precision (точность) – число процентов верно сделанных прогнозов, взятое по отношению ко всему числу верных прогнозов,

2) Recall (отзыв) – число процентов верно сделанных прогнозов, взятое по отношению ко всему числу фактических верных результатов,

3) F1-score (оценка F1) – метрика, которая используется в машинном обучении, когда классы ещё не достаточно сбалансированы. А вообще это

конкретное значение и чем оно ближе к единице, тем модель является лучше. Оценка F1 может быть подсчитана как частное удвоенного произведения точности и отзыва и суммы точности и отзыва.

Показатель support используется для того, чтобы показать какое количество изображений принадлежало определённому классу в тестовом наборе данных [7].

Возвращаясь к рисунку 23 и оперируя информацией выше, можно сделать вывод о том, что из всех изображений, на которых изображена доброкачественная опухоль, модель правильно предсказала этот результат для 80 процентов, а на тех, на которых злокачественная для 82 процентов снимков.

Ввод [72]:
```
# EfficientNet V2
print(classification_report(test_df.label_encoded,
                            efficientnet_v2_test_predictions,
                            target_names=class_names))
```

```
              precision    recall  f1-score   support

   malignant       0.86      0.84      0.85       300
      benign       0.87      0.89      0.88       360

    accuracy                           0.87       660
   macro avg       0.86      0.86      0.86       660
weighted avg       0.87      0.87      0.86       660
```

Рисунок 24 – Показатели классификации для EfficientNet V2

Ввод [73]:
```
# ViT-b16
print(classification_report(test_df.label_encoded,
                            vit_b16_test_predictions,
                            target_names=class_names))
```

```
              precision    recall  f1-score   support

   malignant       0.87      0.86      0.87       300
      benign       0.89      0.89      0.89       360

    accuracy                           0.88       660
   macro avg       0.88      0.88      0.88       660
weighted avg       0.88      0.88      0.88       660
```

Рисунок 25 – Показатели классификации для ViT-B16

Аналогичным образом эти показатели работают и для моделей EfficientNet V2 и ViT-B16, изображенных на рисунках 24 и 25 соответственно.

```
Ввод [74]: def generate_preformance_scores(y_true, y_pred, y_probabilities):

               model_accuracy = accuracy_score(y_true, y_pred)
               model_precision, model_recall, model_f1, _ = (
                   precision_recall_fscore_support(
                       y_true,
                       y_pred,
                       average="weighted"
                   )
               )
               model_matthews_corrcoef = matthews_corrcoef(y_true, y_pred)

               preformance_scores = {
                   'accuracy_score': model_accuracy,
                   'precision_score': model_precision,
                   'recall_score': model_recall,
                   'f1_score': model_f1,
                   'matthews_corrcoef': model_matthews_corrcoef
               }
               return preformance_scores
```

Рисунок 26 – Вычисление различных метрик производительности

На рисунке выше функция вычисляет различные метрики производительности модели на основе предсказанных меток y_pred, истинных меток y_true и вероятностей принадлежности к классам y_probabilities. Таким образом, вычисляются все необходимые метрики качества бинарной классификации. После этого они заносятся в словарь performance_scores.

Далее в соответствующие объекты cnn_performance, efficietnet_v2_performance и vit_b16_performance записываются эти показатели эффективности.

Ввод [78]:
```
performance_df = pd.DataFrame({
    'model_cnn': cnn_performance,
    'model_efficientnet_v2': efficientnet_v2_performance,
    'model_vit_b16': vit_b16_performance
}).T

performance_df
```

Out[78]:

	accuracy_score	precision_score	recall_score	f1_score	matthews_corrcoef
model_cnn	0.812121	0.838531	0.812121	0.811155	0.651782
model_efficientnet_v2	0.865152	0.865088	0.865152	0.864957	0.727651
model_vit_b16	0.880303	0.880235	0.880303	0.880251	0.758442

Рисунок 27 – Показатели производительности моделей

Из рисунка 27 мы можем наблюдать то, что все три модели достаточно высокого качества. Однако vit-модель превосходит две другие. Особенно это видно из коэффициента корреляции Мэтьюса [8], который валируется от -1 до 1. И значения, которые больше 0.6 указывают на практически полное соответствие между предсказанными и фактическими классами. Значит, все три модели можно обобщить на невидимые выборки.

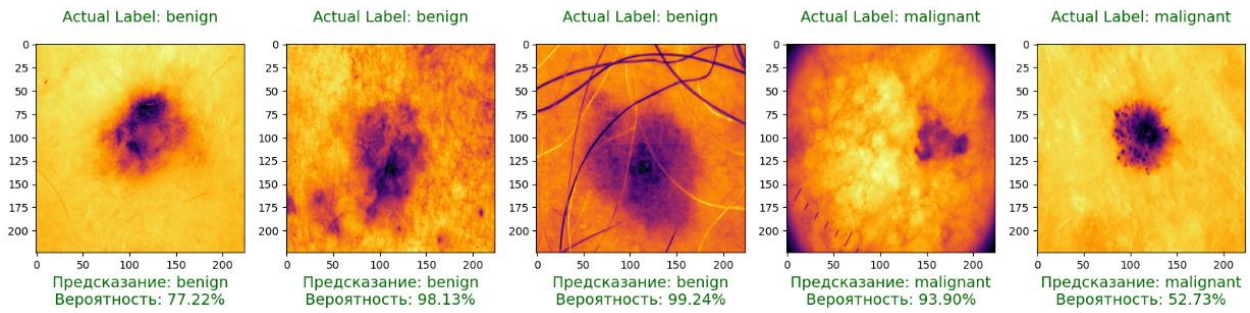

Рисунок 28 – Демонстрация CNN теста

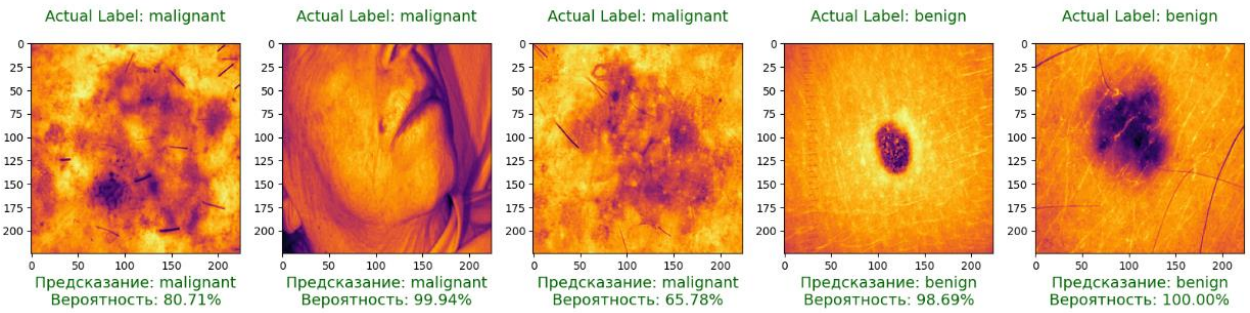

Рисунок 29 – Демонстрация EfficientNet V2 теста

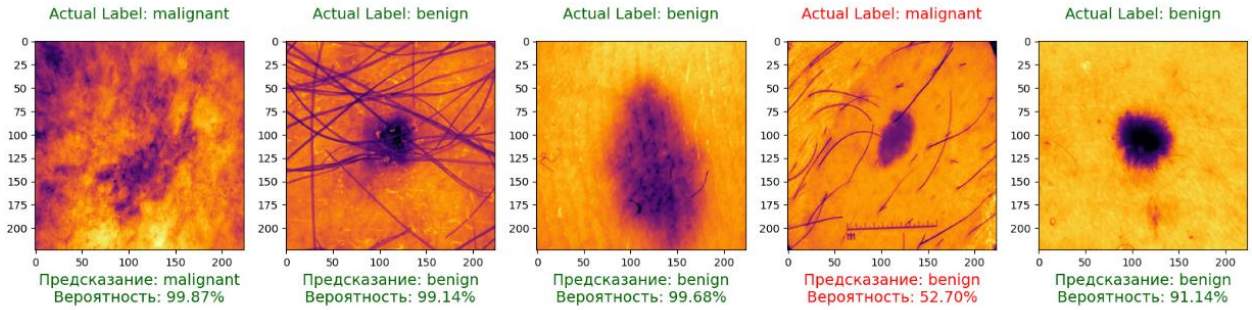

Рисунок 30 – Демонстрация ViT-B16 теста

ROC-кривая – кривая, которая является очень полезным инструментом в оценки эффективности любых моделей. Она применяется, чтобы продемонстрировать итоги бинарной классификации. Благодаря ей определяется взаимосвязь между число верно классифицированных положительных и числом неверно классифицированных отрицательных примеров.

ROC-кривая для всех трех моделей представлена в приложении А.

ЗАКЛЮЧЕНИЕ

В данной дипломной работе мной была проведена обширная аналитика и исследование с использованием методов компьютерного зрения для выявления злокачественных форм рака кожи. На основе проведенной выполненной работы можно сделать следующие выводы:

Я рассмотрел современные методы диагностики рака кожи такие как: дермоскопия, биопсия, ОКТ и массивные многоспектральные анализаторы, что позволило лучше понять характеристики и особенности заболевания.

Искусственный интеллект играет ключевую роль в автоматизации и улучшении процессов диагностики и лечения рака кожи, обеспечивая более точные и быстрые результаты.

Был проведен обзор предыдущих исследований в области выявления рака кожи с применением методов компьютерного зрения, что позволило определить актуальность и значимость данной проблемы. В данном обзоре максимальная точность достигала 82,5 процентов. Однако в моей работе при использовании модели ViT-B16 точность достигала 88 процентов, что является лучшим показателем по сравнению с предыдущими исследованиями.

В рамках данной дипломной работы была разработана методика исследования, включающая выбор набора данных, предобработку данных, выбор моделей и их обучение, а также оценку производительности моделей.

В качестве исходного набора данных был выбран набор ISIC, содержащий множество различных изображений кожных образований.

То, что касается предобработки данных, я использовал масштабирование изображений, их нормализацию, разделение на обучающий и тестовые наборы, а также аугментацию данных для того, чтобы модели давали более точные результаты.

Для решения задачи выявления рака кожи были выбраны три различные модели компьютерного зрения: базовая сверточная нейронная сеть (CNN), предварительно обученная EfficientNet V2 B0 и предварительно обученный

Vision Transformer. В конечном счёте самой эффективной и точной моделью можно с уверенностью назвать ViT-B16, исходя из множества тестов произведённых в конце исследования.

Была определена стратегия обучения и оценка моделей, включающая выбор алгоритмов оптимизации, функций потерь, метрик оценки для более надежной оценки производительности моделей. Во всех трех моделях была функция потерь для бинарной классификации - BinaryCrossentropy. Adam в моем случае был оптимизатором и в качестве скорости обучения бралось значение 0,001. Метрикой accuracy нужна была для оценки точности модели. Для оценки производительности использовались: precision (точность), recall (отзыв), f1-score (оценка F1), коэффициент корреляции Мэтьюса.

Исходя из результатов, можно заключить, что использование методов компьютерного зрения и нейронных сетей позволяет добиться высокой точности и эффективности в выявлении злокачественных форм рака кожи. Однако для дальнейшего развития и улучшения таких моделей требуется проведение дополнительных исследований и внедрение новых технологий и методик. Например, использование со временем более мощных и стабильных моделей. Или создание более обширного датасета с использованием изображений более высокого качества.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Хайкин, С. Нейронные сети. Полный курс : учебное пособие : практикум / С. Хайкин. – Университет McMaster – Москва : Эксмо, 2006. – 31 с. – ISBN 978-0-1309564-8-4.

2. Николенко, С. И. Глубокое обучение : учебное пособие : практикум / С. И. Николенко. – Санкт-Петербург : АРУС, 2018. – 231 с. – ISBN 978-54960253-6-2.

3. Гудфеллоу, Я. Глубокое обучение : учебное пособие : практикум / Я. Гудфеллоу. – Москва : Эксмо, 2018. – 23 с. – ISBN 978-50409594-5-7.

4. Рашид, Т. Создаем нейронную сеть : учебное пособие : практикум / Т. Рашид. – Санкт-Петербург : АРУС, 2019. – 24 с. – ISBN 978-5-9909445-7-2.

5. Халафян, А. А. STATISTICA 6. Математическая статистика элементами теории вероятностей : учебное пособие: практикум / А. А. Халафян. – Москва : Бином, 2010. – 496 с. – ISBN 978-5-9518-0386-3.

6. Шолле, Ф. Глубокое обучение на Python : учебное пособие: практикум / Ф. Шолле. – Санкт-Петербург : АРУС, 2018. – 400 с. – ISBN 978-5-4461-0770-4.

7. Бурков, А. Машинное обучение без лишних слов : учебное пособие : практикум / А. Бурков. – Санкт-Петербург : АРУС, 2020. – 192 с. – ISBN 978-5-4461-1560-0.

8. Лапань, М. Глубокое обучение с подкреплением. AlphaGo и другие технологии : учебное пособие : практикум / М. Лапань. – Санкт-Петербург : АРУС, 2020. – 192 с. – ISBN 978-5-4461-1079-7.

ПРИЛОЖЕНИЕ А

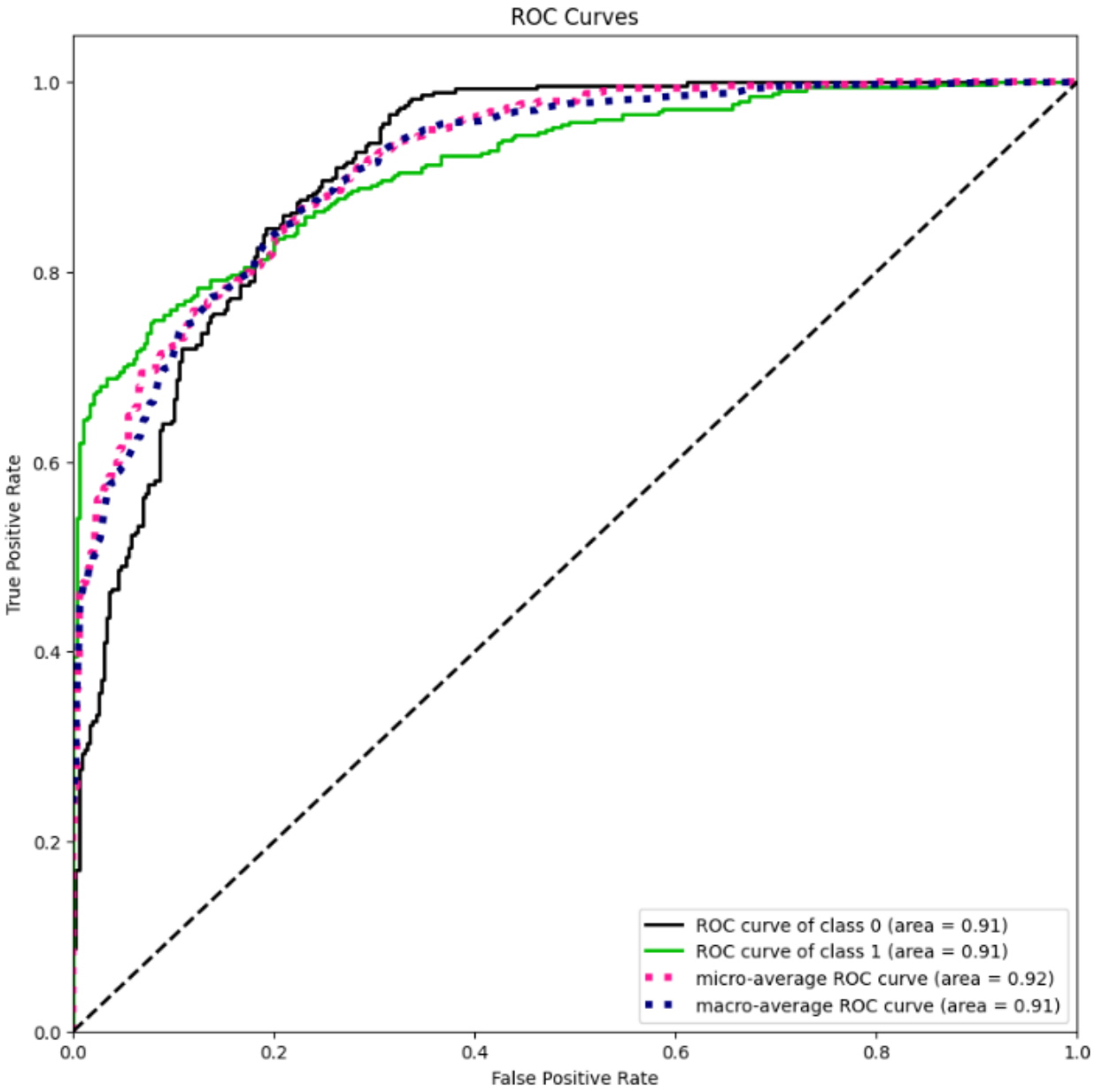

Рисунок А.1 – Кривая ROC для CNN

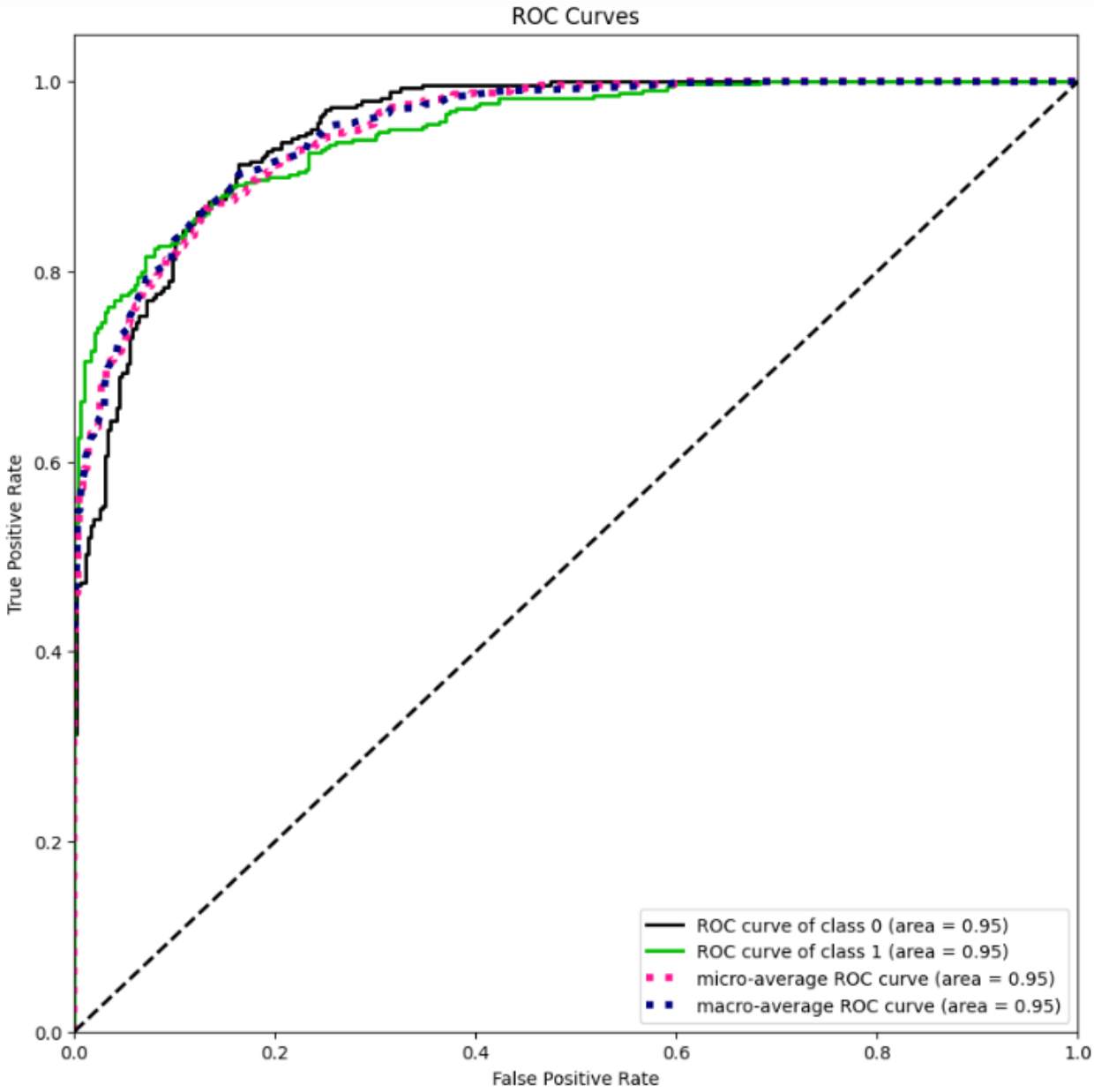

Рисунок А.2 – Кривая ROC для EfficientNet V2

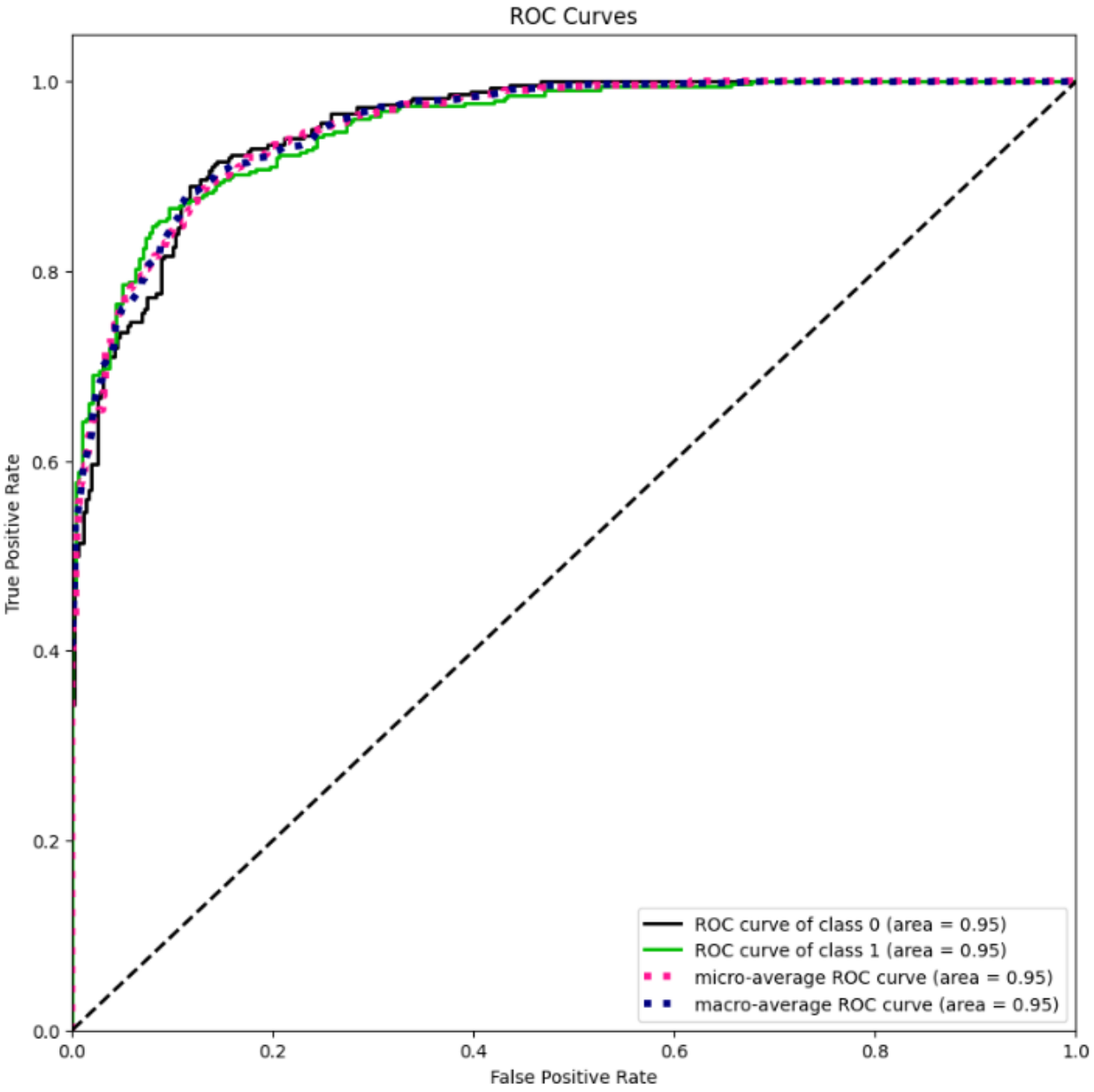

Рисунок А.3 – Кривая ROC для ViT-b16