Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Институт компьютерных и инженерных наук
Кафедра информационных и управляющих систем
Направление подготовки 09.04.04 - Программная инженерия
Направленность (профиль) образовательной программы Управление разработкой программного обеспечения

ДОПУСТИТЬ К ЗАЩИТЕ
Зав. кафедрой
_____ А.В. Бушманов
«___» _____ 2025 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему: Разработка нейронной сети для визуального определения классов лейкоцитов

Исполнитель
студент группы 3105-ом1 _____ А.А. Хулапов
(подпись, дата)

Руководитель
доцент, канд. техн. наук _____ А.В. Бушманов
(подпись, дата)

Руководитель научного
содержания программы
магистратуры
профессор, доктор техн. наук _____ И.Е. Ерёмин
(подпись, дата)

Нормоконтроль
инженер кафедры _____ В.Н. Адаменко
(подпись, дата)

Рецензент
доцент, канд. техн. наук _____ Т.В. Труфанова
(подпись, дата)

Благовещенск, 2025

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Институт компьютерных и инженерных наук
Кафедра информационных и управляющих систем

УТВЕРЖДАЮ
Зав. кафедрой
_____А.В. Бушманов
«___» _____2025 г.

З А Д А Н И Е

К магистерской диссертации студента группы 3105-ом

Хулапова Артёма Александровича

1. Тема магистерской диссертации:

Разработка нейронной сети для визуального определения классов лейкоцитов
(Утверждено приказом от 06.03.2025 № 609-уч)

2. Срок сдачи студентом законченной работы (проекта): 10.06.2025

3. Исходные данные к магистерской диссертации: принципы работы нейронных сетей, методы компьютерного зрения, интернет ресурсы, учебная литература

4. Содержание магистерской диссертации (перечень подлежащих разработке вопросов): Сверточные нейронные сети, проектирование алгоритма решения задачи, разработка модификации архитектуры сети

5. Перечень материалов приложения (наличие чертежей, таблиц, графиков, схем, программных продуктов, иллюстративного материала и т.п.):

6. Рецензент магистерской диссертации: Труфанова Т.В.

7. Дата выдачи задания 29.01.2025

Руководитель выпускной квалификационной работы:
(фамилия, имя, отчество, должность, уч.степень, уч.звание)
А.В. Бушманов, доцент, канд. техн. наук, доцент

Заявление принял к исполнению _____

РЕФЕРАТ

Магистерская диссертация содержит 80 страница, 39 рисунков, 3 таблицы, 50 источников

КЛАССИФИКАЦИЯ КЛЕТОК КРОВИ, КЛАССИФИКАЦИЯ ИЗОБРАЖЕНИЙ, ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ, СВЕРТОЧНАЯ НЕЙРОННАЯ СЕТЬ, ГЛУБОКОЕ ОБУЧЕНИЕ, МАШИННОЕ ЗРЕНИЕ.

Целью данной магистерской диссертации является разработка программы для автоматизированной классификации лейкоцитов на изображениях с использованием методов компьютерного зрения и сверточных нейронных сетей. В работе рассмотрены особенности предметной области, проведён обзор существующих подходов к классификации объектов на изображениях и проанализированы современные архитектуры нейросетей, применяемые в медицинской визуализации. Основу методологии составляют методы глубокого обучения, обработки изображений и сегментации. Разработка программного продукта велась с использованием языка Python и различных библиотек.

Процесс разработки включал следующие этапы:

– подготовка и аугментация датасета изображений лейкоцитов;

– построение и обучение сверточной нейросети;

– реализация функций классификации и визуализации результатов;

– создание графического интерфейса пользователя;

– тестирование точности и производительности модели на различных наборах данных.

Разработанная система позволяет выполнять классификацию как отдельных изображений, так и целых наборов, обеспечивая визуальный и текстовый вывод результатов.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Точная классификация лейкоцитов является важным элементом в диагностике гематологических патологий и комплексной оценке состояния здоровья. Несмотря на наличие разнообразных подходов к этой задаче, потребность в разработке высокоэффективного и оптимального классифицирующего инструмента сохраняет свою актуальность. В рамках данной работы предлагается применение обученной нейронной сети в качестве автоматизированного классификатора для решения задачи дифференциации лейкоцитов на медицинских изображениях.

Актуальность разработки системы автоматизированной классификации изображений лейкоцитов определяется необходимостью повышения качества диагностики заболеваний крови и внедрением современных технологий компьютерного зрения и нейронных сетей в сферу медицинской диагностики.

Исследование предполагает тестирование предложенной архитектуры и методики глубокого обучения в контексте задачи классификации лейкоцитов. Система, разрабатываемая в данной работе, стремится к высокой точности и оперативности в определении принадлежности лейкоцита к определенному классу. Это предоставляет перспективу использования разработанной системы в качестве вспомогательного инструмента для гематологического анализа крови.

Целью исследования является разработка системы автоматизированной классификации лейкоцитов на основе методов компьютерного зрения и глубокого обучения.

Для достижения поставленной цели необходимо выполнить следующие задачи:

– исследование существующих методов и технологий в области машинного обучения и компьютерного зрения;

– рассмотреть процесс поиска объектов на изображении и его сегментации;

– разработка и обучение нейронной сети для автоматической классификации лейкоцитов;

– интеграция разработанной модели в графический интерфейс;

– провести анализ эффективности и точности разработанной модели.

Объектом исследования являются методы искусственного интеллекта, применяемые для классификации клеток крови.

Предметом исследования является процесс классификации клеток крови с использованием искусственного интеллекта и его применение в медицинской практике.

1 ПРОБЛЕМА РАСПОЗНАВАНИЯ ОБЪЕКТОВ НА ИЗОБРАЖЕНИЯХ С ИСПОЛЬЗОВАНИЕМ НЕЙРОННЫХ СЕТЕЙ

1.1 Характеристика предметной области

Лейкоциты, или белые кровяные клетки, представляют собой гетерогенную группу ядерных клеток, играющих ключевую роль в иммунной системе организма. Они проявляют высокую вариабельность по форме, размеру и функциональной активности. Основные характеристики лейкоцитов включают в себя:

Морфологическая разнообразность: Лейкоциты могут иметь различные формы, включая сегментированные ядра (нейтрофилы, эозинофилы, базофилы), круглые ядра (лимфоциты, моноциты) и другие вариации, что делает их морфологически сложными для точной классификации.

Ядерная структура: Ядра лейкоцитов могут быть одноядерными или многоядерными, что также является важным критерием для их разграничения.

Гранулярность цитоплазмы: Некоторые типы лейкоцитов обладают характерной гранулярностью цитоплазмы, которая может быть крупной (нейтрофилы, базофилы) или мелкой (лимфоциты, моноциты), что отражает их функциональное предназначение.

Лейкоциты подразделяются на пять основных видов в соответствии с их функциональными и морфологическими характеристиками (рисунок 1):

Нейтрофилы – клетки с сегментированным ядром, содержащие гранулы. Их основная функция заключается в фагоцитозе бактерий и участии в воспалительных реакциях.

Лимфоциты – клетки с круглым или овальным ядром, ответственные за иммунный ответ и адаптивный иммунитет.

Моноциты – крупные клетки с овальным ядром, предшественники макрофагов, выполняющих роль в фагоцитозе и иммунорегуляции.

Эозинофилы – клетки с ядром в форме градиента и крупными гранулами в цитоплазме. Их функция связана с борьбой с паразитарными инфекциями и

аллергическими реакциями.

Базофилы – клетки с сегментированным ядром и мелкими гранулами, играющие роль в аллергических реакциях и воспалительных процессах.

Рисунок 1 – Классификация лейкоцитов.

Эффективная классификация лейкоцитов представляет собой сложную задачу из-за их многообразия и вариабельности. Нейронные сети представляют собой многообещающее направление для автоматизации процесса классификации лейкоцитов, обеспечивая при этом высокую точность и скорость.

В контексте настоящего исследования требуется разработать алгоритм для визуальной классификации лейкоцитов на изображениях на основе нейронной сети. Процесс классификации лейкоцитов включает два ключевых этапа: выделение области, содержащей лейкоцит, и отнесение его к определённому классу на основе морфологических признаков. Применение разнообразных методов обработки изображений, включая выделение контуров, использование цветовых фильтров, методы сопоставления с шаблонами, а также поиск ключевых точек могут в полной мере решить данную задачу. С помощью данных подходов возможно идентифицировать специфические признаки, критически важные для последующей классификации объектов на изображении.

Тем не менее, традиционные методы, такие как поиск контуров и шаблонов, имеют свои ограничения, поскольку изображения лейкоцитов могут быть представлены с различной ориентацией, масштабом и цветом, что делает их

менее понятными. Кроме того, некоторые изображения кровяных клеток могут перекрывать друг друга, что усложняет задачу их распознавания. В свете вышеизложенного, для реализации целей настоящего исследования было принято решение о применении методов машинного обучения.

Машинное обучение представляет собой ключевое направление в области искусственного интеллекта, фокусирующееся на создании систем, способных выполнять задачи, традиционно ассоциируемые с когнитивными способностями человека. Среди множества методов машинного обучения, искусственные нейронные сети, в частности, многослойные перцептроны (MLP), выделяются как наиболее действенный инструмент для решения задач классификации объектов на изображениях. Этот подход позволяет сети обучаться на большом объеме данных, выявляя сложные взаимосвязи между признаками объектов на изображении и их классами. Основной особенностью многослойных перцептронов является их способность обрабатывать и классифицировать изображения, даже если объекты на них имеют различные масштабы, ориентацию или частичные искажения.

Однако многослойные перцептроны могут столкнуться с трудностями при работе с изображениями, содержащими большое количество объектов или сложные фоны. Для решения этих задач могут быть использованы более сложные архитектуры нейронных сетей, такие как CHC, или же – сверточные нейронные сети. Они демонстрируют повышенную устойчивость к различным видам искажений и эффективно справляются с задачами классификации и поиска объектов. Этот подход обеспечивает частичную инвариантность к изменениям масштаба, смещениям, поворотам, вариациям ракурса и прочим преобразованиям. Однако ключевой характеристикой сверточных нейронных сетей является их способность к обучению.

Обобщая рассмотренные методы распознавания объектов на изображениях, можно прийти к заключению, что для задачи классификации лейкоцитов наиболее адекватным является применение подходов, базирующихся на сверточных нейронных сетях.

1.2 Анализ принципов работы искусственной нейронной сети

Прежде чем углубиться в практическую реализацию, целесообразно рассмотреть теоретические основы, лежащие в основе функционирования искусственных нейронных сетей. Искусственные нейронные сети (ИНС) – это вычислительные модели, разработанные по аналогии с нейронной организацией человеческого мозга. Их эффективность в решении обширного круга задач определяется способностью воспроизводить принципы обработки информации, характерные для биологических нейронов, которые выступают в роли элементарных вычислительных единиц, взаимодействующих посредством обмена сигналами. Фундаментальным компонентом ИНС является искусственный нейрон, представляющий собой функцию, которая обрабатывает входные данные, масштабируя их посредством настраиваемых в ходе обучения весовых параметров. Полученные взвешенные суммы подвергаются дальнейшей фильтрации нелинейной функцией активации, которая определяет выходное состояние нейрона. Множество таких взаимосвязанных нейронов формируют сети, используемые для решения широкого спектра прикладных задач, включая классификацию, прогнозирование, обработку изображений, машинный перевод и анализ речевых данных, и многое другое.

ИНС обладают уникальными свойствами: они способны обучаться и моделировать сложные нелинейные связи, что открывает возможности для выявления тонких закономерностей и повышения точности работы за счет накопления опыта. Структурно ИНС моделируются как направленные графы, где каждый нейрон соответствует узлу, а связи между нейронами представлены рёбрами, которым присвоены индивидуальные весовые коэффициенты. На рисунке 2 изображен пример такой сети.

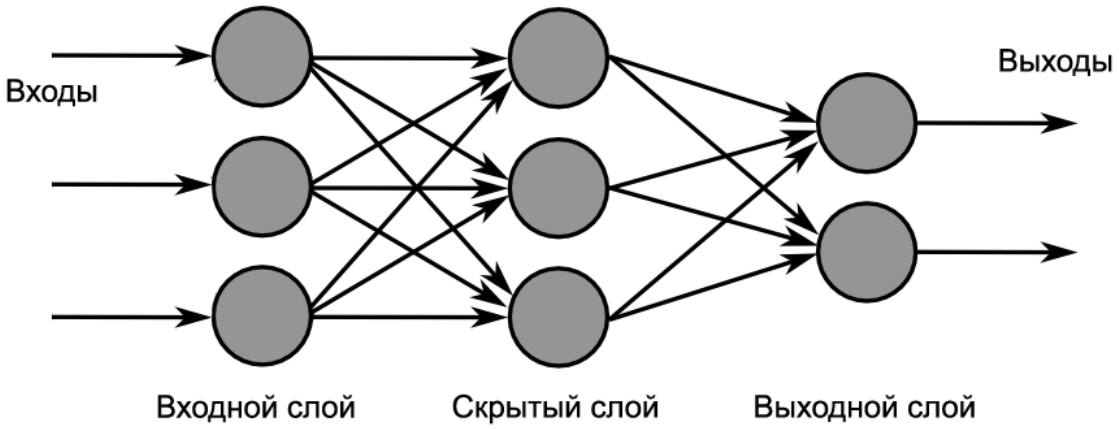

Рисунок 2 – Структура ИНС

Для осмысления фундаментальных механизмов искусственных нейронных сетей целесообразно рассмотреть перцептрон – одну из исторических архитектур ИНС. Схематическое изображение его структуры приведено на рисунке 3.

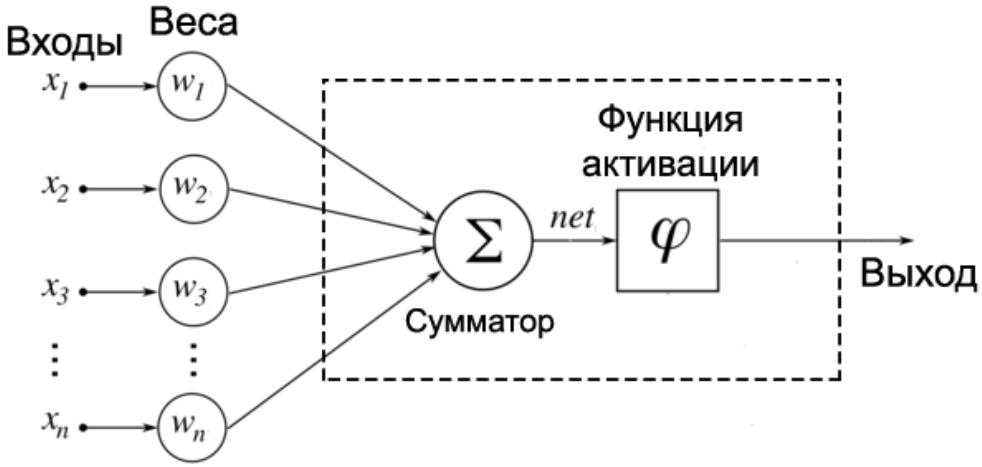

Рисунок 3 – Структура перцептрона

Принцип действия модели перцептрона заключается в следующем – входные сигналы поступают на входные нейроны и далее передаются сумматору, при этом их влияние регулируется весовыми коэффициентами связей. Весовые коэффициенты определяют значимость каждого входа для конечного результата. Сумматор производит взвешенную сумму входных сигналов, умножая их на установленные веса, а затем передает агрегированное значение функции активации, которая генерирует выходной сигнал нейрона.

Активация нейрона определяется функцией активации, которая регулирует поступающий сигнал, обеспечивая его соотнесение с требуемым диапазоном

выходных значений. Выбор конкретного типа функции активации определяет этот диапазон. Если бы функции активации не было, нейронная сеть трансформировалась бы в элементарную линейную модель, так как именно нелинейные преобразования, выполняемые этой функцией, обеспечивают способность системы решать задачи повышенной сложности. В сфере нейронных сетей используются различные типы функций активации, среди которых наиболее значимыми являются сигмоидальная, ступенчатая, гиперболический тангенс, линейная и ReLU – функция усеченного линейного преобразования. Графические примеры этих функций показаны на рисунке 4.

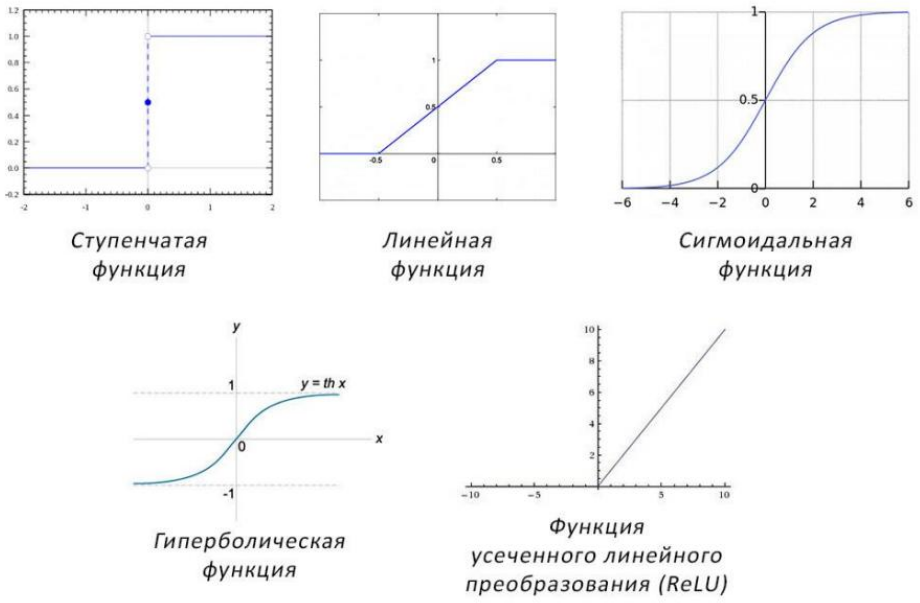

Рисунок 4 – Основные функции активации

Оптимальный выбор функции активации определяется спецификой задачи, решаемой нейронной сетью. И хотя перцептрон зарекомендовал себя как эффективное средство для элементарной классификации, решение более комплексных задач потребовало появления архитектур нового уровня – многослойных перцептронов (MLP). В отличие от своего предшественника, MLP состоят из нескольких последовательно соединенных слоев нейронов, при этом нейроны внутри одного слоя не имеют связей между собой. Промежуточные слои, расположенные между входным и выходным, именуются скрытыми.

Значимым дополнением к архитектуре большинства нейронных сетей являются нейроны смещения, интегрированные во входные и скрытые слои. Они

отличаются отсутствием входных связей и константной активностью, равной единице. Это уникальное свойство позволяет смещению модулировать позицию функции активации на графике, значительно улучшая обучающие способности модели. Демонстрация структуры многослойной нейронной сети с включением нейронов смещения представлена на рисунке 5.

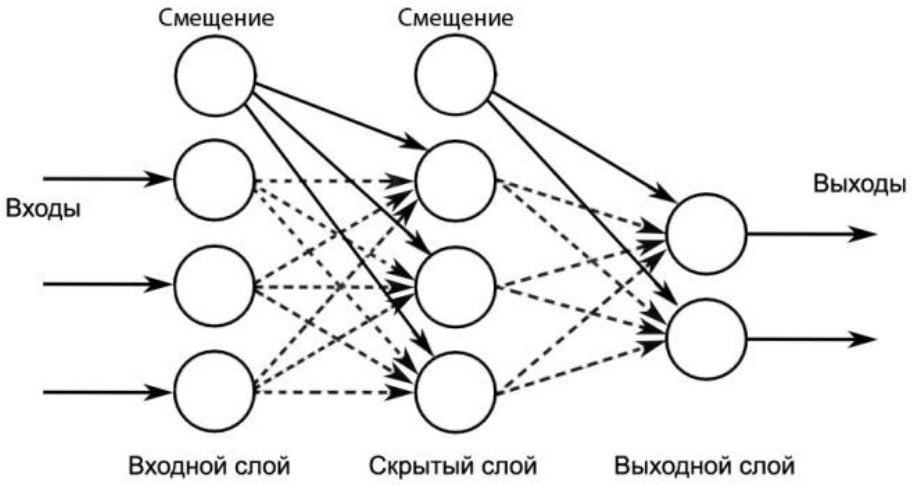

Рисунок 5 – Многослойная нейронная сеть с нейронами смещения

Хотя нейронные сети и разделяют общие принципы строения, их архитектурное воплощение отличается большим разнообразием. Основные различия между этими архитектурами кроются в таких аспектах, как глубина сети, число нейронов и топология связей. Именно эти различия обусловливают их эффективность в решении конкретных задач.

1.3 Анализ существующих методов обучения искусственных нейронных сетей

Искусственные нейронные сети характеризуются своей способностью к самообучению. Суть этого обучения заключается в стремлении сети как можно точнее совпадать с ожидаемыми результатами. Для этого используется метод прямого распространения ошибки, который позволяет сети сравнивать свои выводы, полученные при обработке конкретного примера из обучающего набора, с заранее заданными значениями.

Оценка эффективности и выявление недочетов в работе нейронной сети реализуются посредством функции потерь. Среди наиболее часто использу-

мых функций для этой цели выделяют: MSE – среднеквадратичное отклонение и MAE – среднее абсолютное отклонение, математические выражения которых приведены в формуле (1).

$$\text{MSE: } (y' - y)^2,$$
$$\text{MAE: } |y' - y|^2, \quad (1)$$

где y' – ожидаемое значение, y – результат работы сети.

Процесс обучения нейронных сетей подразделяется на два основных типа: детерминированный и стохастический. Детерминированный метод фокусируется на точной настройке весов, которые оказывают наиболее существенное влияние на величину ошибки. В рамках же стохастического подхода изменения весовых коэффициентов носят случайный характер, но лишь те из них, которые уменьшают ошибку, сохраняются в процессе обучения.

В процессе обучения нейронных сетей доминируют детерминированные алгоритмы, ключевым из которых является градиентный спуск, рисунок 6. Суть данного метода оптимизации заключается в последовательном уменьшении значения функции потерь сети путем систематического регулирования ее весовых параметров.

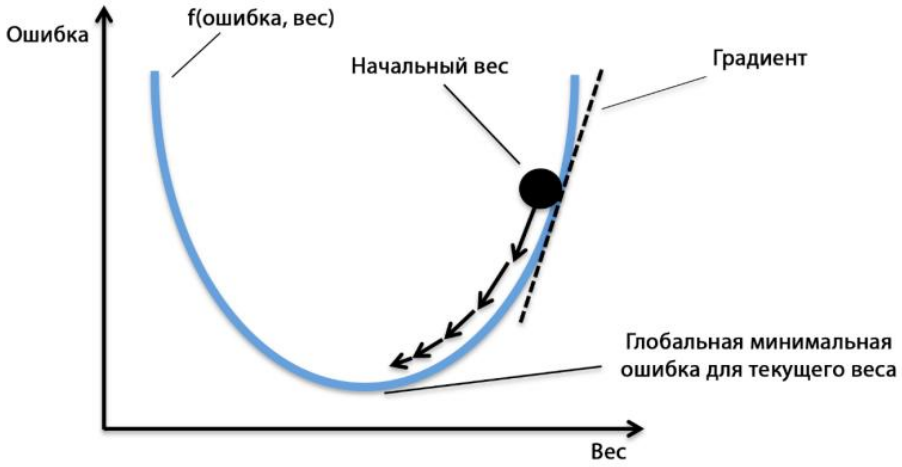

Рисунок 6 – Принцип градиентного спуска

Использование градиентного спуска для обучения нейронной сети представляет собой целенаправленный процесс по нахождению такой конфигурации весов, которая приводит к минимальной ошибке сети. Градиент функции ошиб-

ки, иллюстрированный на рисунке 6, служит ключевым ориентиром, указывающим на направление максимального увеличения функции. Следовательно, для достижения снижения ошибки изменения весовых коэффициентов должны осуществляться в противоположном направлении к градиенту. В формуле (2) представлен механизм обновления весовых коэффициентов посредством алгоритма градиентного спуска.

$$w^{+} = w - \eta * \nabla \mathrm{E} \quad (2)$$

где w^{+} – новое значение весового коэффициента;

w – текущее значение весового коэффициента;

$\nabla \mathrm{E}$ – градиент ошибки;

η – характеристика скорости обучения, значение, на величину которого изменяются новые веса.

Недостаточная скорость обучения тормозит процесс минимизации функции потерь, делая его длительным. С другой стороны, чрезмерно высокая скорость обучения может привести к пропуску минимума функции потерь, не позволяя достичь оптимального решения.

Многослойные нейронные сети, имеющие сотни или даже тысячи весовых коэффициентов, сталкиваются с задачей минимизации функции потерь. Для достижения этой цели требуется вычисление градиентов для каждого весового коэффициента, что эквивалентно определению его частной производной. Для решения этой задачи применяется алгоритм обратного распространения ошибки. Этот подход позволяет находить частные производные относительно каждого весового коэффициента, это, в свою очередь, дает возможность оценить вклад каждого коэффициента в погрешность модели.

Обучение нейронной сети начинается с определения погрешности на финальном выходе, которая затем распространяется назад, слой за слоем, к начальному скрытому слою. Этот обратный процесс передачи ошибки даёт возможность рассчитать производные функции потерь для каждого слоя, что указывает на нужное направление изменения весов для уменьшения общей погрешности. Важно отметить, что расчет весовых коэффициентов последнего

слоя и всех предыдущих имеет свои особенности. Вклад весов последнего слоя в общую ошибку определяется по формуле (3). Для остальных слоев вклад вычисляется по формуле (4), где дифференциал определяется как произведение весового коэффициента и результата, вычисленного по формуле (3).

$$\frac{\partial \mathrm{E}}{\partial w_{j,k}} = \frac{\partial i_k}{\partial w_{j,k}} * \frac{\partial o_k}{\partial i_k} * \frac{\partial \mathrm{E}}{\partial o_k} = o_j * \frac{\partial f_k(S_k)}{\partial s_k} * \delta_j \quad (3)$$

где $w_{j,k}$ – весовой коэффициент, соединяющий нейроны j и k;

$\frac{\partial \mathrm{E}}{\partial w_{j,k}}$ – вклад веса взвешенную сумму нейрона k;

$\frac{\partial o_k}{\partial i_k}$ – вклад взвешенной суммы в выходной сигнал нейрона k;

$\frac{\partial \mathrm{E}}{\partial o_k}$ – вклад выходного сигнала нейрона k в общую ошибку;

o_j – ошибка выходного сигнала нейрона j, находящаяся с помощью функции потерь.

В формуле (4) отражено нахождение вклада весовых коэффициентов всех слоев в общую ошибку, за исключением последнего.

$$\frac{\partial \mathrm{E}}{\partial w_{j,k}} = \frac{\partial i_k}{\partial w_{j,k}} * \frac{\partial o_k}{\partial i_k} * \left(\sum_l \frac{\partial i_l^{+1}}{\partial o_k}\right) \quad (4)$$

где $\frac{\partial i_l^{+1}}{\partial o_k}$ – вклад выходного сигнала нейрона k в ошибку выходного сигнала нейрона 1 на следующем слое, стоящим за слоем нейрона k.

Данная значение функции вычисляется как умножение весового коэффициента $w_{j,k}$ на произведение $\frac{\partial o_k}{\partial i_k} * \frac{\partial \mathrm{E}}{\partial o_k}$ вычисляемое в формуле (3).

Таким образом, обучение нейронной сети методом градиентного спуска – это циклический процесс последовательной оптимизации весов. На каждом шаге сначала определяется ошибка, генерируемая моделью для каждого обучающего примера, а затем эта ошибка суммируется по всем примерам с применением функции потерь. Далее вычисляется сумма квадратов всех отклонений от целевых значений, после чего весовые коэффициенты нейронной сети корректируются при помощи алгоритма обратного распространения ошибки.

Однако, градиентный спуск, несмотря на свою эффективность, имеет существенный недостаток – обучение модели может быть длительным из-за необходимости вычисления ошибок для всей обучающей выборки, которая часто имеет внушительный размер. Чтобы оптимизировать этот процесс, используется модификация градиентного спуска – стохастический градиентный спуск, позволяющий ускорить обучение.

В данном алгоритме веса подстраиваются индивидуально для каждого обучающего примера или небольшой группы данных. Хотя метод градиентного спуска может приводить к скачкообразному уменьшению функции потерь из-за неточной корректировки весов, он демонстрирует быструю сходимость к локальному минимуму. На рисунке 7 изображено сравнение методов обучения.

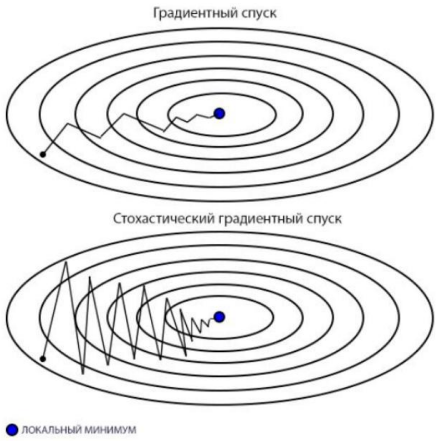

Рисунок 7 – Процесс обучения при использовании различных вариантов градиентного спуска

Однако существенным недостатком градиентного спуска является риск застревания в локальном минимуме при большом числе весов, что препятствует нахождению глобального оптимума. Для повышения эффективности обучения и минимизации функции потерь, весовые коэффициенты сети изначально задаются случайными значениями, а в случае неудовлетворительных результатах требуется пересмотр её архитектуры. Однако для достижения наилучших результатов в обучении нейронных сетей необходима сверточная архитектура.

1.4 Анализ сверточных нейронных сетей как одного из методов классификации изображений

Среди множества нейросетевых архитектур, сверточные нейронные сети (CNN) занимают ведущее положение в задачах обработки и классификации изображений, а также в решении смежных проблем, связанных с визуальными данными. Архитектура CNN основывается на аналогии с функционированием зрительной системы человека. Отличительной особенностью является то, что распознавание различных визуальных паттернов активирует соответствующие, но разные, группы нейронов, что позволяет эффективно распознавать и обрабатывать разнообразные визуальные признаки. Сверточная нейронная сеть способна извлекать и иерархически обрабатывать признаки от простейших контуров до более сложных деталей объектов.

Ключевыми элементами архитектуры CNN являются слои свёртки и подвыборки (пулинга), которые чередуются и завершаются полносвязными слоями. Слои подвыборки в нейронной сети способствуют сокращению пространственных размеров данных, а слои свёртки отвечают за идентификацию и выделение ключевых визуальных характеристик (например, контуров и текстур), сохраняя наиболее значимую информацию. В заключительной части сети используются полносвязные слои, которые обеспечивают классификацию извлечённых признаков. Рисунок 8 демонстрирует общую структуру сверточной нейронной сети.

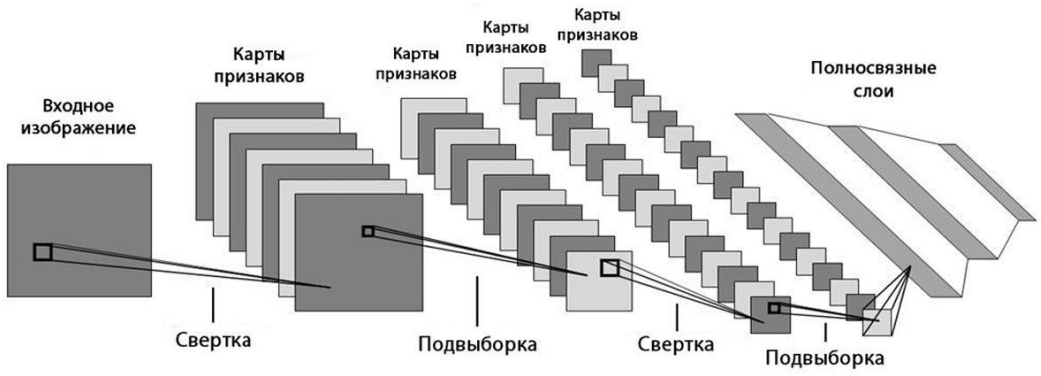

Рисунок 8 – Архитектура свёрточной нейронной сети

В основе функциональности (CNN) лежит сверточный слой, выступаю-

щий в качестве стартового этапа обработки визуальных данных. Его работа вдохновлена принципами функционирования зрительной коры головного мозга, где определённые группы нейронов активируются в ответ на стимулы в строго локализованных участках поля зрения, именуемых рецептивными полями.

Изображение, поступающее на вход сверточного слоя, представлено в виде матрицы пикселей. Для монохромных изображений это одна матрица яркости, значения от 0 до 255, тогда как цветные изображения кодируются тремя отдельными матрицами, соответствующими каналам цветовой модели RGB.

Ключевым элементом сверточного слоя является ядро свёртки – небольшая матрица, содержащая весовые коэффициенты. Его глубина соответствует глубине входного изображения. Типичный размер ядра – от 3×3 до 7×7 пикселей. Этот параметр определяет, сколько низкоуровневых признаков будет агрегировано для формирования более сложных.

Сама операция свёртки инициируется определением области, эквивалентной ядру – рецептивного поля, которое последовательно перемещается по входному изображению с заданным шагом. В каждой новой позиции элементы изображения, попадающие в рецептивное поле, поэлементно умножаются на соответствующие весовые коэффициенты ядра. Результаты перемножения суммируются и образуют новый пиксель выходной матрицы. Это итеративное сканирование всего изображения приводит к созданию новой матрицы, которая содержит признаки более высокого уровня абстракции. Рисунок 9 отображает операцию свёртки.

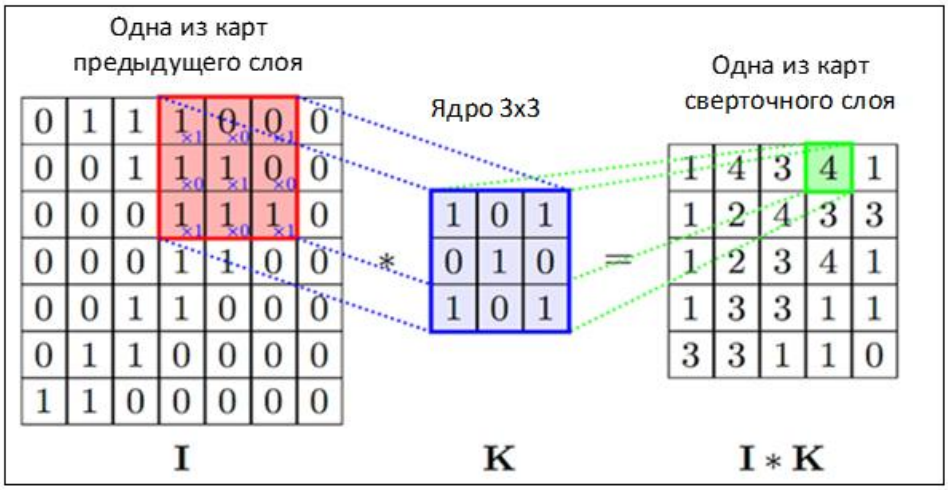

Рисунок 9 – Свёртка черно-белых изображений

Операция свёртки для монохромных изображений наглядно представлена на рисунке 9. Однако обработка цветных изображений с использованием свёртки требует более сложного подхода: для каждого из цветовых каналов (красного, зеленого, синего) применяются отдельные матрицы, которые совместно формируют трёхмерное ядро свёртки. Окончательное изображение признаков генерируется путём суммирования результатов свёртки, полученных для каждого цветового канала. Детализация этого процесса для цветных изображений показана на рисунке 10.

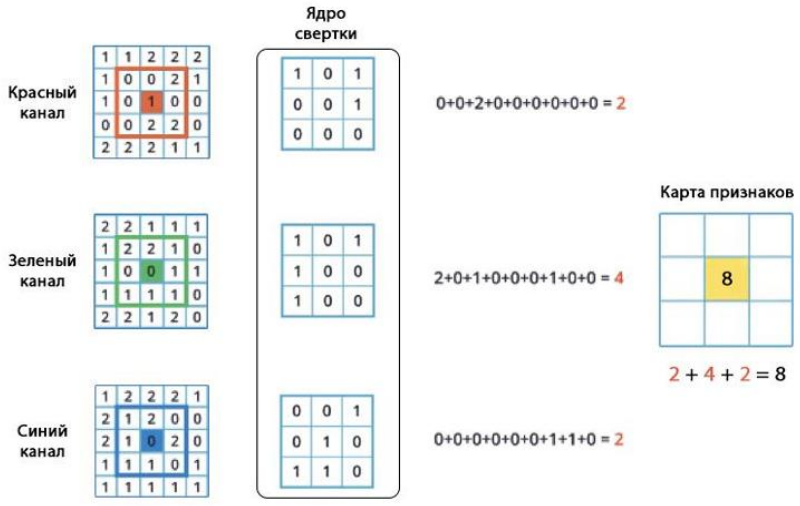

Рисунок 10 – Свертка цветного изображения

Поскольку периферийные пиксели входного изображения не всегда оказываются в области восприятия, результат свертки имеет меньшие размеры по сравнению с оригиналом. Во избежание потери информации с краёв и для пре-

дотвращения последовательного уменьшения разрешения изображения был создан процесс zero-padding, или нулевое заполнение. По границам изображения создаётся дополнительная область, заполненная нулевыми значениями. С помощью свёртки при этом рассчитывается размер данной области. Рисунок 11 демонстрирует данный процесс.

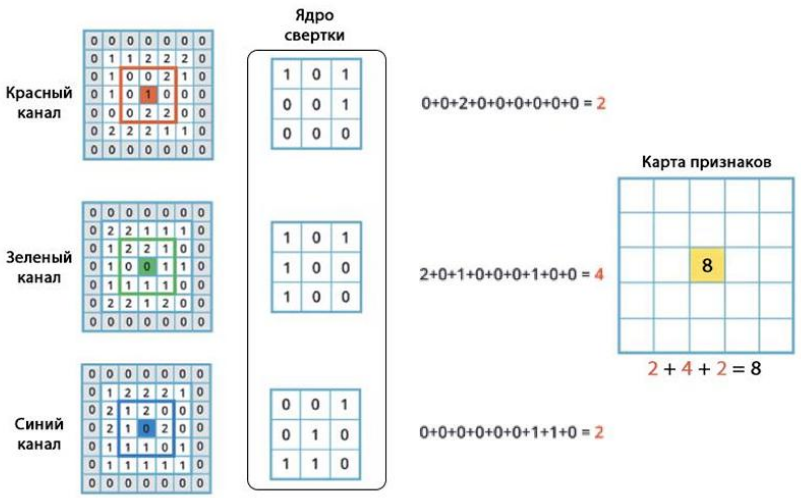

Рисунок 11 – Свёртка изображения с использованием zero-padding

Аналогично многослойным перцептронам, сверточные сети используют нейроны смещения, чьи параметры корректируются в процессе обучения и влияют на каждый компонент результирующей матрицы. Вслед за этим этапом выходная матрица подвергается нелинейной трансформации посредством функции активации, роль которой часто выполняет ReLU, которая эффективно отсекает отрицательные значения, оставляя положительные без изменений.

Свертка генерирует матрицу, известную как карта признаков, где выделяются ключевые черты изображения. Эти черты варьируются от базовых элементов, таких как ровные линии и дуги, до более изящных форм. Их характер напрямую зависит от расположения свёрточного слоя в архитектуре сети. Начальные слои концентрируются на простых визуальных компонентах, тогда как последние слои способны улавливать более абстрактные и обобщённые особенности изображения. Важно отметить, что каждый свёрточный слой, как правило, включает множество ядер, каждое из которых нацелено на обнаружение специфического элемента изображения.

Увеличение числа ядер свертки расширяет возможности сети для извле-

чения более сложных и точных особенностей изображений, а итогом данной операции является набор числовых матриц, чьё количество определяет глубину результирующей карты признаков.

Далее следует слой подвыборки, который играет ключевую роль в снижении размерности входных данных. Он достигает этого, агрегируя группы пикселей в единый пиксель, сохраняя при этом общую информацию о присутствии признаков, но не детализируя их точное положение. Такой подход позволяет эффективно сократить объем обрабатываемой информации, что в свою очередь ведет к уменьшению вычислительной нагрузки сети.

Применение подвыборки в сверточных сетях позволяет оптимизировать количество параметров, что, в свою очередь, снижает нагрузку на вычислительные мощности. Существуют два основных подхода к подвыборке: по усреднению и по максимуму. В первом случае пиксели в группе заменяются их средним значением, во втором – на наиболее яркий пиксель. Подвыборка по максимуму более востребована, поскольку эффективно выявляет характерные черты объекта и минимизирует влияние шумов на изображении.

Подвыборка, изображенная на риснуке 12, это процесс, управляемый двумя ключевыми настройками: размером ядра, которое определяет количество обрабатываемых пикселей за один проход, и шагом смещения, отвечающим за перемещение ядра по изображению. Часто используется ядро 2x2 с шагом 2, что приводит к существенной компрессии размерности получаемой карты признаков.

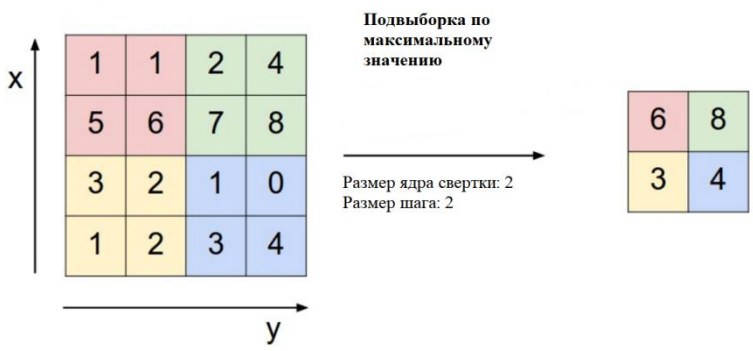

Рисунок 12 – Подвыборка по максимальному значению

Для обнаружения комплексных объектов, сверточные нейронные сети применяют многоуровневую структуру, состоящую из последовательно расположенных слоев свертки и подвыборки. При прохождении через каждую пару слоев, сложность распознаваемых образов возрастает, поскольку количество ядер свертки увеличивается, а размерность представлений уменьшается. Таким образом, сеть способна выявлять признаки различной иерархии, начиная от простых геометрических форм и заканчивая сложными объектами.

Последние слои сверточной нейронной сети, представляющие собой полносвязные слои, определяют конечный результат обработки изображения – классификацию объекта. Каждый нейрон в этих слоях получает информацию от всех нейронов предыдущего слоя, что позволяет модели учесть все ключевые особенности изображения. Обучение сети заключается в нахождении оптимальных весов связей между нейронами, что и обеспечивает точную классификацию изображений по заранее определенным классам.

Прежде чем информация достигает полного связного слоя, сформированная карта признаков подвергается критическому преобразованию: она выравнивается в одномерный вектор. В этом векторе каждый компонент символизирует вероятность обнаружения определенного признака на изображении.

Заключительный слой называется полносвязным. Он включает число нейронов, эквивалентное заранее определенным категориям классификации, вычисляет вероятность отнесения входного изображения к каждой из них посредством функции активации. Сеть затем присваивает изображению тот класс, который соответствует наивысшей вычисленной вероятности.

Подводя итог, в данном пункте был проведён анализ структуры сверточной нейронной сети и описан процесс того, как она способствует извлечению признаков из изображений для эффективной классификации. С помощью слоев свертки и подвыборки сеть способна обнаруживать признаки различной степени сложности, а в конечном итоге полносвязные слои определяют принадлежность объекта к определенной категории.

2 АНАЛИЗ СРЕДСТВ РАСПОЗНАВАНИЯ ОБРАЗОВ НА ИЗОБРАЖЕНИИ

2.1 Предлагаемый алгоритм решения задачи на основе сверточных нейронных сетей

2.1.1 Функциональные требования к программе

Разрабатываемое программное обеспечение для классификации изображений лейкоцитов с использованием нейронной сети должно включать в себя следующий функционал:

– возможность изменения параметров классификации нейронной сети (количество слоев, количество фильтров, количество циклов обучения, порог классификации и другие параметры, влияющие на результат классификации);

– возможность запуска процесса обучения нейронной сети на новых данных (загрузка данных для обучения и запуск тренировки модели с выбранными параметрами);

– возможность загрузки изображений в систему для дальнейшей классификации (поддержка различных форматов изображений и автоматическая подготовка данных для анализа нейронной сетью);

– возможность классификации загруженных изображений с использованием обученной модели нейронной сети (распознавание изображений лейкоцитов и предсказание соответствующей группы);

– возможность получения результатов классификации (вывод подгруппы лейкоцитов и вероятность правильности классификации);

– возможность просмотра истории предыдущих классификаций (сохранение и отображение данных о выполненных классификациях, включая изображение, результат и дату);

– возможность сохранения результатов классификации для дальнейшего анализа (автоматическое сохранение данных о классификации, включая результаты, время выполнения и параметры модели).

Диаграмма прецедентов, построенная на основе данных функциональных требований, приведена на рисунке 18.

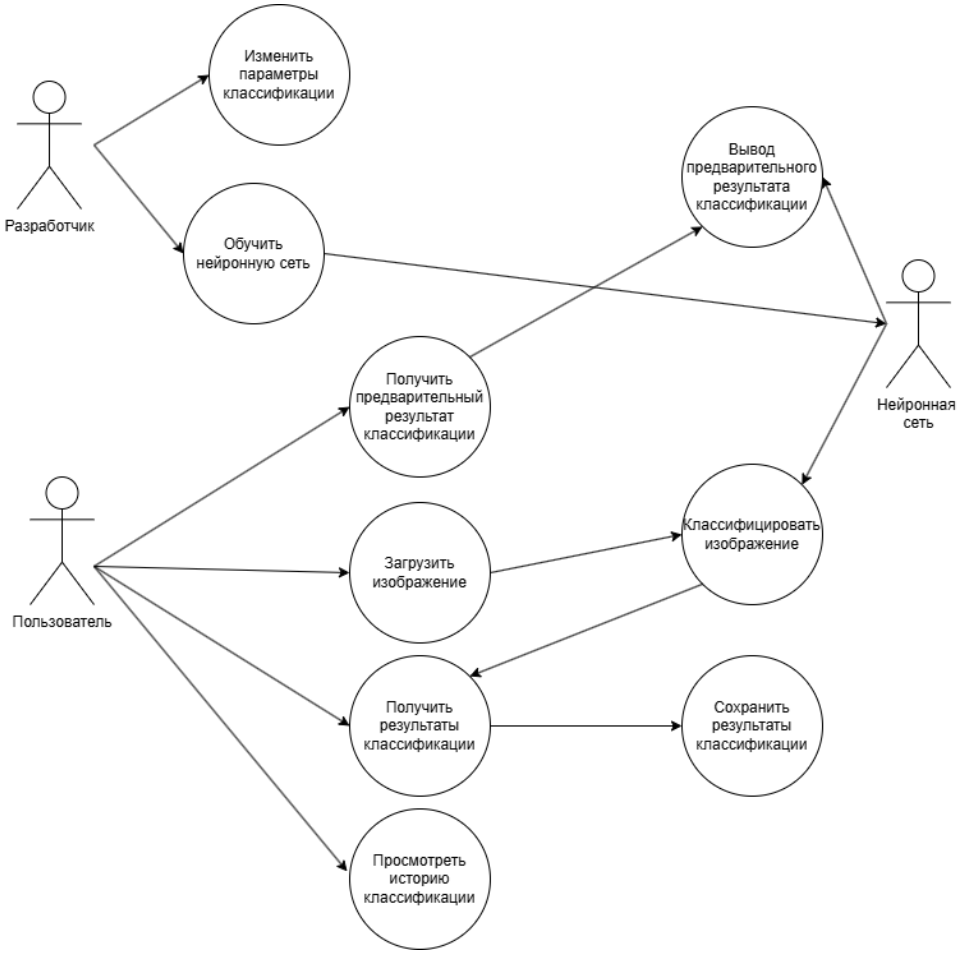

Рисунок 18 – Диаграмма прецедентов

2.1.2 Описание архитектур CNN для решения задачи

Существует множество моделей сверточных нейронных сетей. Большинство из существующих моделей участвуют в конкурсе ImageNet LSVRC, представляющим собой самый большой датасет изображений.

Однако задачи медицинской визуализации, такие как классификация лейкоцитов, имеют свои особенности. Они требуют не только высокой точности, но и способности работать с небольшими наборами данных, а также учитывать специфику биомедицинских изображений. В связи с этим выбор архитектуры CNN для таких задач должен учитывать не только производительность, но и способность модели к эффективному обучению на ограниченных данных.

В данном разделе рассматриваются наиболее популярные архитектуры CNN, такие как AlexNet, U-Net, ResNet и другие, а также обосновывается выбор архитектуры VGG-16, которая была адаптирована для работы с медицинскими изображениями.

AlexNet – это знаковая сверточная нейронная сеть, разработанная в 2012 году. Данная архитектура содержит 5 сверточных слоев, а в каждом слое по 4096 нейронов. Отличительной особенностью данной нейронной сети является использование двух сверточных слоев подряд.

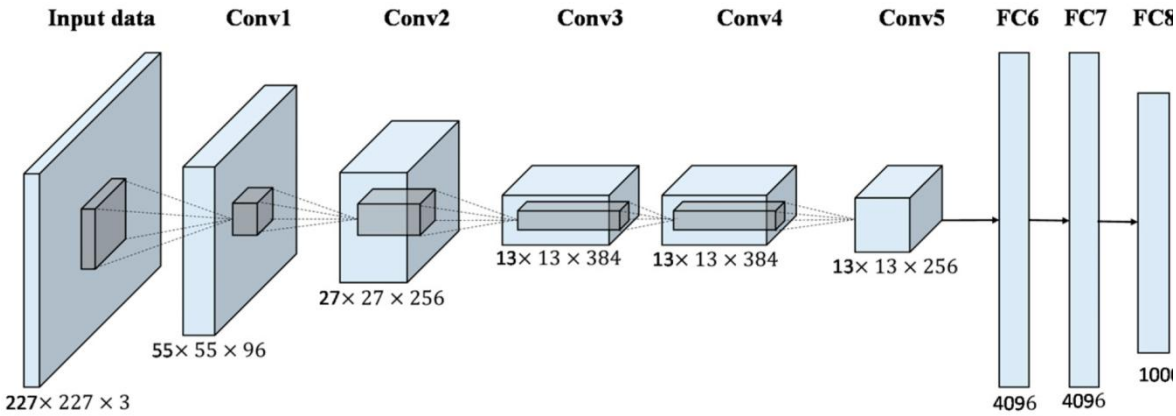

Рисунок 13 – Архитектура AlexNet

U-Net представляет собой сверточную нейронную сеть, специально модифицированную так, чтобы она могла работать с меньшим количеством примеров (обучающих образов) и делала более точную сегментацию. Она состоит из двух частей: сжимающей (encoding) и разжимающей (decoding), рисунок 14. Сжимающая часть извлекает признаки, а разжимающая восстанавливает пространственную информацию.

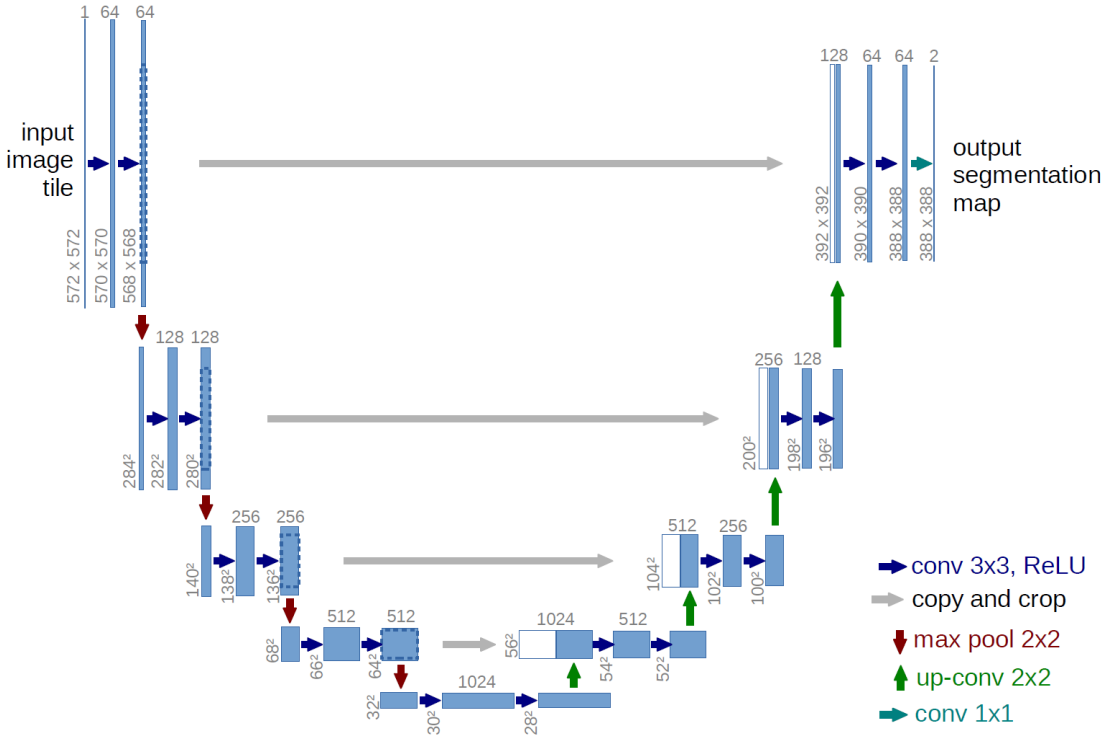

Рисунок 14 – Архитектура U-Net

Сужающийся путь последовательно применяет операции свёртки и пулинга для уменьшения пространственного разрешения, извлекая при этом высокоуровневые признаки. Затем разжимающийся путь выполняет повышающую дискретизацию и объединение (конкатенацию) с соответствующими слоями сужающегося пути, восстанавливая детализацию и формируя выходную маску, разделяющую изображение на несколько классов.

ResNet – это вид нейронной сети, отличающийся наличием остаточных связей, которые позволяют сети учиться более глубоко и эффективно, архитектура представлена на рисунке 15. Если VGG-подобные архитектуры сталкиваются с потерей производительности и точности при наращивании слоев, ResNet, напротив, получает прирост качества с увеличением глубины. Этот эффект достигается за счёт инновационной концепции, при которой весовые слои обучаются остаточным отображениям, используя прямые связи с входными данными слоя.

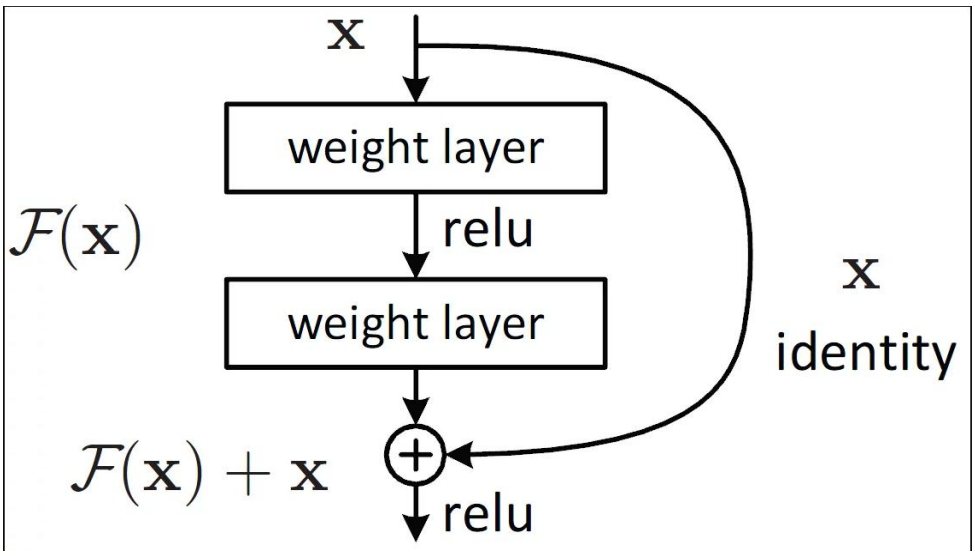

Рисунок 15 – Архитектура ResNet

Модель Inception, как показано на рисунке 16, базируется на идее нормализации нейронных активаций и поиске наиболее эффективных комбинаций признаков, полученных из входных данных. Главные принципы её дизайна направлены на увеличение объёма обрабатываемой информации за счёт гармоничного соотношения глубины и ширины сети, а также применении свёрточных фильтров разнообразных размеров, включая 3×3. Ключевая идея данной СНС – применение inception блока. Он заключается в применении операции свертки 1 на 1, и последующих свертках 3 на 3 и 5 на 5, причем это делается в разных ветках блока и потом конкатенируется результат. Также данная архитектура имеет 3 выхода, в начале, середине и в конце. Это необходимо, чтобы эффективнее обучать ранние слои нейронной сети.

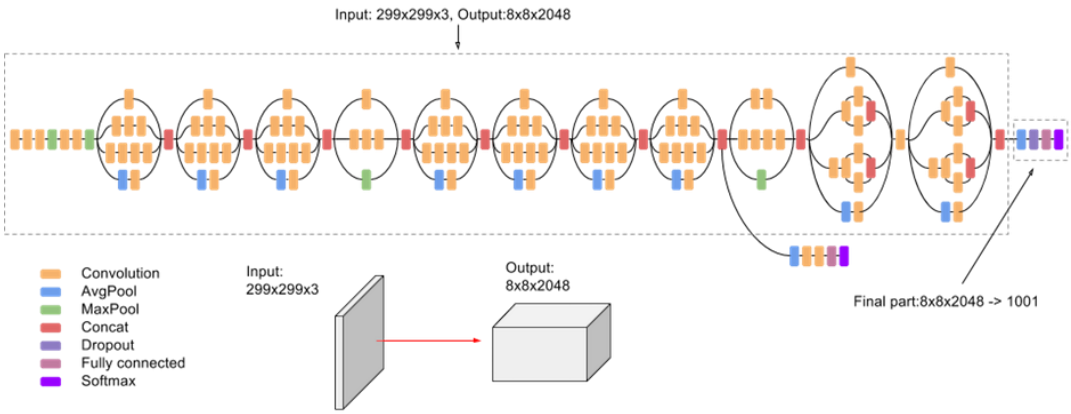

Рисунок 16 – Архитектура Inception

Для достижения цели исследования была выбрана сверточная нейронная сеть архитектуры VGG-16, а именно её модифицированная версия, реализованная в виде последовательной модели Sequential Neural Network.

VGG-16 – это знаковая модель, участвовавшая в конкурсе ImageNet ILSVRC-2014, ставшая важной как первая по-настоящему глубокая нейронная сеть.

Во-первых, заменены большие фильтры на группы фильтров размерами 3 на 3, которые следуют один за другим.

Во-вторых, эта архитектура использует несколько блоков, в каждом из которых идут несколько сверточных слоев подряд. Является одной из самых тяжелых по памяти нейронных сетей.

Базовая версия состоит из 13 сверточных слоёв и 3 полносвязных, с последовательным расположением слоёв, использующих малые ядра свертки (3×3) и операции субдискретизации (max pooling).

В данной работе была применена упрощённая и адаптированная версия этой архитектуры, построенная на основе модели Sequential. Такая структура позволяет легко конфигурировать последовательное добавление сверточных слоёв, слоёв подвыборки и полносвязных слоёв, обеспечивая при этом высокую интерпретируемость и контроль над процессом обучения.

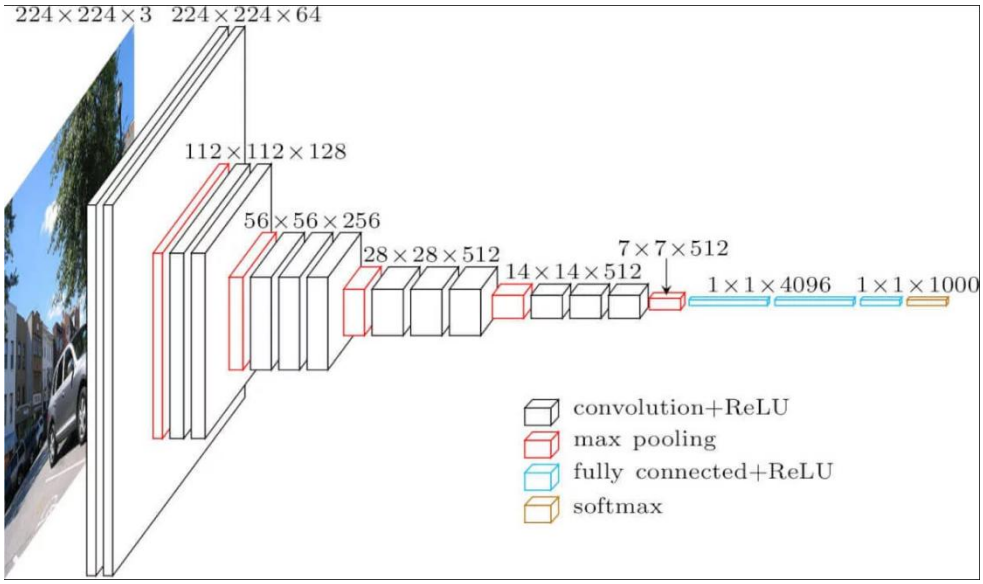

Рисунок 17 – Архитектура VGG-16

Модифицированная архитектура включает в себя несколько блоков: последовательные сверточные слои с функцией активации ReLU, слои подвыборки (pooling) для уменьшения размерности признаков, а также полносвязные слои для финальной классификации. Использование Sequential-подхода позволило гибко адаптировать сеть под специфику задачи визуальной классификации лейкоцитов.

Процесс решения задачи классификации лейкоцитов состоит из следующих этапов:

Сбор и аугментация данных:

– для обучения и тестирования модели используются открытые базы данных изображений лейкоцитов;

– выполняется аугментация данных, включающая операции поворота, масштабирования и отражения, для увеличения объема данных и повышения устойчивости модели к различным искажениям.

Предварительная обработка данных:

– изображения приводятся к единому размеру (256x256 пикселей) для унификации входных данных;

– нормализация значений пикселей к диапазону [0, 1] для улучшения сходимости модели при обучении.

Разработка архитектуры модели:

– использована архитектура Sequential Neural Network для извлечения признаков и сегментации изображений.

Обучение модели.

Оценка модели:

– проводится тестирование модели на независимом наборе данных;

– производится анализ производительности модели с использованием метрик точности.

2.1.3 Алгоритм предлагаемого решения

Системе даются метки и обучающие изображения.

Первым процессом будет предварительная обработка, включающая преобразование изображения в оттенки серого и процесс фильтрации для удаления шумов.

После предварительной обработки будет применена сегментация, рисунок 19, а также произойдет дальнейшее расширение изображения.

Признаки будут извлечены с помощью сверточной нейронной сети и ее слоев для создания модели классификации.

Набор данных, полученный в результате, будет являться моделью-образцом для нейронной сети.

После этапа тестирования предоставляется входное изображение клети крови человека, полученное из Интернета или больницы.

Изображение будет подвергнуто тому же процессу предварительной обработки, после чего произойдет извлечение признаков. Затем сеть будет использована для сравнения входного изображения и предсказания результата с использованием модели и результатов, полученных на тестовых изображениях. С помощью этих результатов можно классифицировать, к какому типу клеток крови принадлежит выбранная нами клетка крови. Работа данного алгоритма представлена на рисунке 20.

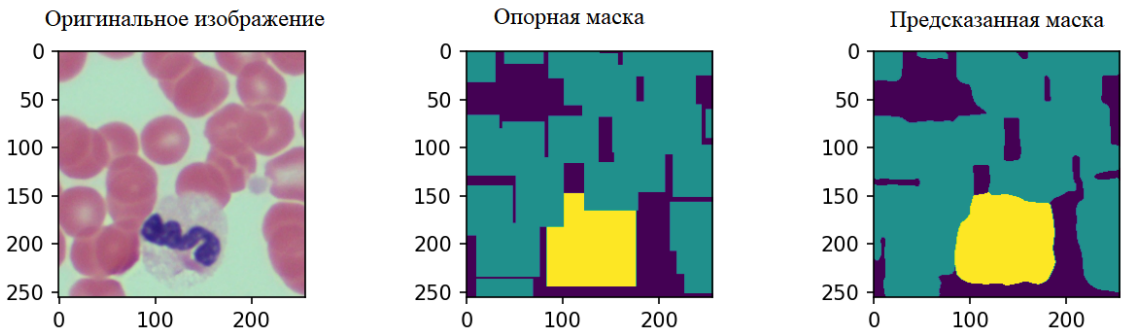

Рисунок 19 – Этапы сегментации изображения

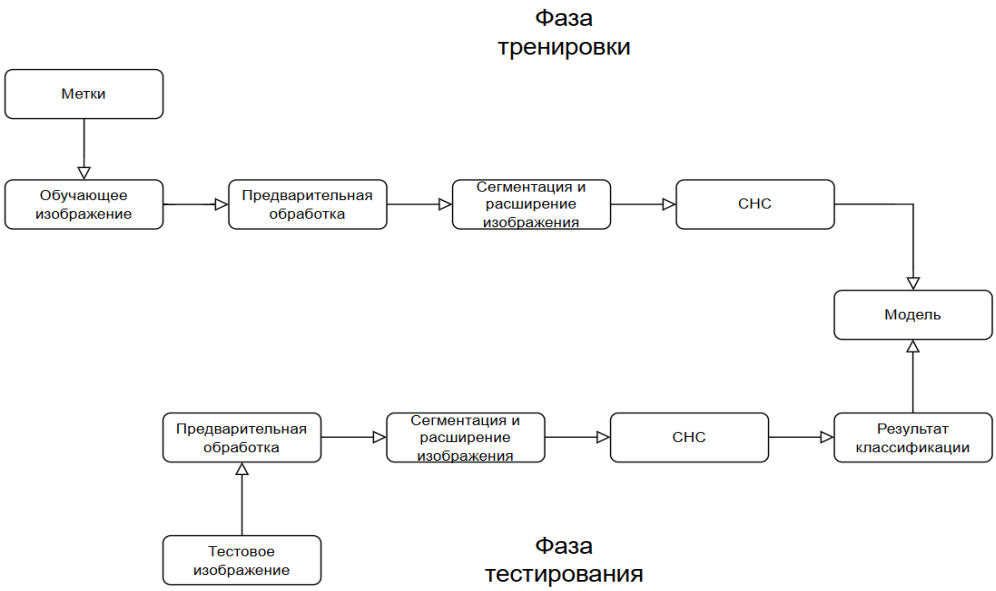

Рисунок 20 – Алгоритм работы нейронной сети

2.2 Обзор средств для решения задач компьютерного зрения

Для успешной разработки приложений в области компьютерного зрения доступен широкий спектр программных инструментов. Современные инженеры и исследователи опираются на мощные библиотеки и платформы, каждая из которых предлагает уникальный набор функций. В таблице 1 ниже представлены наиболее востребованные из них, с кратким описанием их назначения и ключевых особенностей.

Таблица 1 – Обзор основных средств для решения задач компьютерного зрения

Средство разработки	Основное назначение	Ключевые особенности
1	2	3
OpenCV (Open Source Computer Vision Library)	Комплексная библиотека алгоритмов компьютерного зрения и обработки изображений.	Стандарт индустрии; реализована на C/C++, поддерживает Python, Java и другие языки; открытый исходный код.
PCL (Point Cloud Library)	Масштабный открытый проект для обработки и анализа 2D/3D-изображений и облаков точек.	Включает алгоритмы для фильтрации, оценки характеристик, реконструкции поверхностей, регистрации, сегментации.

Продолжение таблицы 1

1	2	3
ROS (Robot Operating System)	Платформа для разработки ПО для роботов, облегчающая создание сложных систем управления.	Набор инструментов, библиотек и соглашений; упрощает разработку программ для множества типов роботов, часто используется со зрением.
MATLAB	Высокоуровневый язык и интерактивная среда для программирования, численных расчётов и визуализации.	Популярен для быстрого анализа данных, разработки алгоритмов, прототипирования моделей и приложений в научных кругах.
CUDA (Compute Unified Device Architecture)	Программно-аппаратная архитектура для параллельных вычислений на GPU от Nvidia.	Значительно повышает вычислительную производительность; критически важна для обучения глубоких нейронных сетей и ресурсоемких задач.

OpenCV была разработана в 1999 году Гари Бредски в компании Intel, а её первая версия была выпущена в 2000 году. К проекту присоединился Вадим Писаревский, который возглавил российскую команду Intel, занимающуюся разработкой OpenCV. В 2005 году библиотека была успешно применена в автомобиле Stanley, одержавшем победу в соревновании DARPA Grand Challenge. Дальнейшее развитие OpenCV осуществлялось при поддержке компании Willow Garage. На сегодняшний день OpenCV включает в себя множество алгоритмов, связанных с компьютерным зрением и машинным обучением, и продолжает активно развиваться.

Библиотека поддерживает широкий набор языков программирования, включая C++, Python, Java, и работает на различных платформах, таких как Windows, Linux, OS X, Android и iOS. Кроме того, ведутся работы по интеграции высокопроизводительных вычислений на GPU с использованием технологий CUDA и OpenCL.

OpenCV-Python представляет собой Python API для OpenCV, объединяющий преимущества OpenCV, C++ API и гибкость языка Python. Эта библиотека является связующим звеном между OpenCV и Python, специально разработанным для решения задач в области компьютерного зрения.

Python, созданный Гвидо ван Россумом, является языком программирования общего назначения, который быстро завоевал популярность благодаря своей простоте и высокой читаемости кода. Он позволяет разработчикам выражать сложные идеи с помощью минимального количества строк кода, сохраняя при этом понятность и структурированность.

Несмотря на широкое применение Python для быстрой разработки, его скорость исполнения существенно ниже по сравнению с C/C++. Частым решением этой проблемы является интеграция кода на C/C++ в среду Python. Путём разработки критически важных по производительности алгоритмов на C/C++ и их экспозиции через Python-обёртки, разработчики получают гибкость в использовании этих компонентов как нативных модулей Python. Такой гибридный подход к разработке обеспечивает два основных преимущества: во-первых, производительность кода остается на уровне оригинального C/C++ (поскольку в фоновом режиме выполняется именно он), а во-вторых, код на Python становится более лаконичным и удобным для написания, чем на C/C++. Таким образом, OpenCV-Python представляет собой Python-интерфейс для оригинальной реализации OpenCV на C++.

Интеграция OpenCV-Python с библиотекой Numpy является ключевым аспектом его высокой производительности. Numpy предоставляет высокооптимизированные инструменты для численных операций, отличающиеся синтаксисом, схожим с MATLAB. Эта синергия достигается благодаря автоматическому преобразованию всех массивов данных между форматами OpenCV и Numpy, что обеспечивает удобство работы. Кроме того, это упрощает интеграцию с другими библиотеками, такими как SciPy и Matplotlib.

Как библиотека с открытым исходным кодом, OpenCV (Open Source Computer Vision Library) служит мощным инструментом для задач компьютер-

ного зрения и машинного обучения. Её разработка была нацелена на обеспечение многофункциональной основы для широкого спектра приложений, связанных с компьютерным зрением, а также для ускорения внедрения технологий машинного восприятия в коммерческие продукты. Благодаря лицензии BSD, OpenCV позволяет компаниям свободно использовать и модифицировать код, что делает её удобным инструментом для коммерческого применения.

Библиотека OpenCV представляет собой мощный инструмент для работы в области компьютерного зрения, предоставляющий большое количество готовых алгоритмов для анализа изображений и видео. В её арсенале имеются средства для выполнения таких задач, как распознавание лиц и жестов, детектирование объектов, определение положения и перемещения объектов в кадре, а также генерация трёхмерных реконструкций сцен.

OpenCV активно применяется в системах видеонаблюдения, в робототехнике, при разработке приложений дополненной реальности и в медицинской визуализации. Благодаря открытости исходного кода и поддержке со стороны широкого сообщества разработчиков, библиотека постоянно развивается и адаптируется под современные потребности в автоматической обработке визуальной информации. Активное международное сообщество, насчитывающее свыше 47 тысяч участников, и более 18 миллионов загрузок свидетельствуют о её огромной популярности и значимости в индустрии. OpenCV активно применяется в коммерческих компаниях, научно-исследовательских группах и государственных организациях.

OpenCV находит применение в обширном диапазоне практических сценариев. Её функционал востребован в таких областях, как формирование панорамных изображений из отдельных снимков, обнаружение вторжений в охранных системах, мониторинг производственных процессов, помощь роботам в ориентировании и взаимодействии с объектами, а также обнаружения несчастных случаев в бассейнах, создания интерактивных арт-инсталляций, проверки состояния взлетных полос на наличие посторонних предметов.

2.3 Средства реализации программного продукта

При реализации программного продукта для классификации лейкоцитов были выбраны специализированные инструменты и библиотеки, которые обеспечивают эффективную разработку и высокую производительность системы. Основным языком программирования является Python, который предоставляет высокоуровневые средства для быстрой разработки и обладает обширным экосистемным окружением для машинного обучения и обработки изображений.

Python – это высокоуровневый интерпретируемый язык программирования, который стал стандартом в области машинного обучения и глубокого обучения. Благодаря обширной экосистеме библиотек и фреймворков, Python значительно ускоряет процесс разработки и внедрения сложных алгоритмов.

Преимущества Python:

– простота и читаемость: интуитивный и лаконичный синтаксис Python делает его доступным как для начинающих, так и для опытных разработчиков;

– активное сообщество: широкая пользовательская база и активное сообщество разработчиков способствуют быстрому распространению знаний, обучающих материалов и готовых решений;

– богатая экосистема библиотек: библиотеки, такие как NumPy, SciPy, Pandas и Matplotlib, предоставляют мощные инструменты для обработки данных, научных вычислений и визуализации.

Недостатки Python:

– скорость выполнения: по сравнению с компилируемыми языками, такими как C++ или Java, Python может быть менее производительным из-за своей интерпретируемой природы;

– ограничения многопоточности: наличие GIL (Global Interpreter Lock) может создавать сложности при реализации многопоточных приложений, ограничивая параллелизм.

В качестве среды разработки использовалась Visual Studio Code, рисунок 21. Эта IDE отличается удобством работы с Python-проектами, предлагая поддержку отладки, интеграцию с системами контроля кода и широкий спектр пла-

гинов для оптимизации производительности. Microsoft Visual Studio также поддерживает Python с открытым исходным кодом через разработку Python и специализированные рабочие нагрузки для обработки и анализа данных, а также предоставляет бесплатные инструменты Python для Visual Studio.

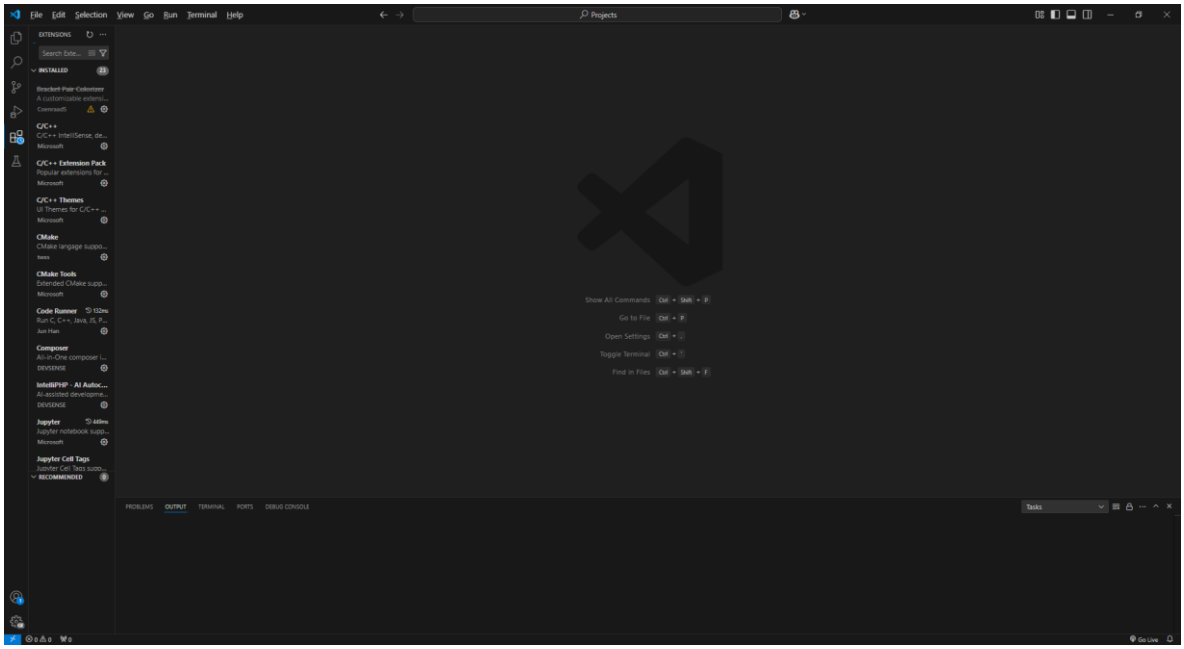

Рисунок 21 – Рабочая среда Visual Studio Code

В Visual Studio реализован высококачественный редактор Python, оснащенный широким набором функций, включая синтаксическую подсветку, автодополнение кода для всех файлов и библиотек, форматирование кода, подсказки по функциям и рефакторинг. Кроме того, он предлагает такие уникальные возможности, как наглядное представление классов, переход к определению символа, поиск всех ссылок и фрагментов кода. Непосредственная связь с интерактивным окном упрощает процесс разработки существующего Python-кода в файле.

В Visual Studio для каждой настройки Python, доступной в IDE, вы можете без труда активировать интерактивную среду (REPL), которая позволяет работать с интерпретатором Python прямо в рамках Visual Studio, минуя необходимость использования отдельной командной строки. Переход между различными средами также осуществляется просто и быстро.

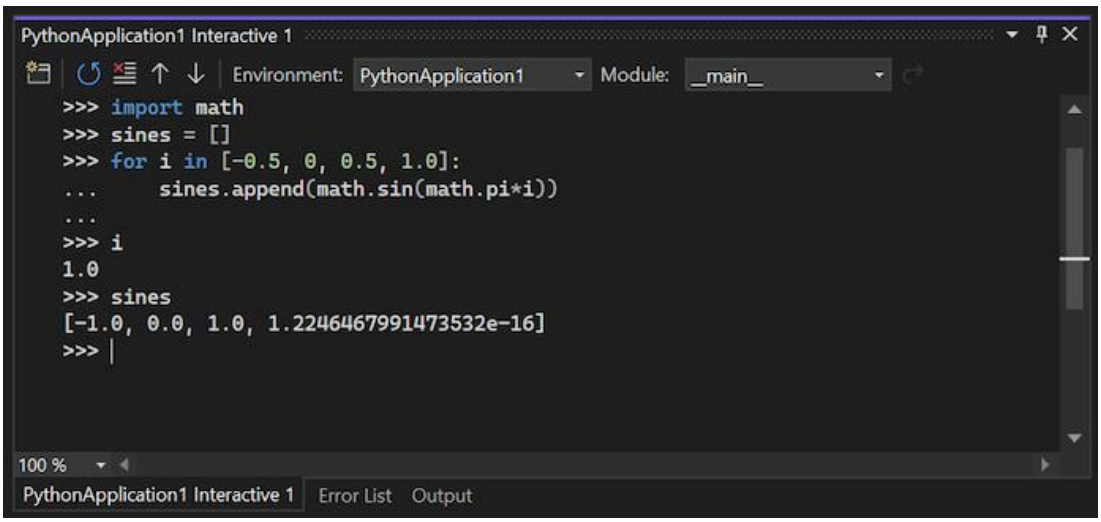

Рисунок 22 – интерактивное окно Visual Studio

В Visual Studio взаимодействие между окном интерактивного режима и редактором Python максимально упрощено. Вы можете отправлять отдельные строки или блоки кода из редактора в интерактивное окно, а затем с помощью удобных функций переходить к дальнейшей работе. Visual Studio Code предлагает возможность быстрого анализа кода без необходимости запуска отладчика. Передача выделенного кода в интерактивное окно и обратно в редактор осуществляется с помощью простых комбинаций клавиш. Благодаря сочетанию этих функций, можно детально изучать фрагменты кода в окне Interactive и без труда сохранять полученные результаты в файл с помощью редактора. Visual Studio также обеспечивает поддержку IPython/Jupyter в режиме REPL, предоставляя возможность работы с графиками, .NET и Windows Presentation Foundation (WPF) непосредственно в среде.

Visual Studio предоставляет инструменты для эффективного управления проектами любой сложности, которые со временем неизбежно усложняются. В Visual Studio проект – это не просто набор папок, а полноценная модель, отражающая взаимосвязи между файлами и их назначением. Благодаря этой модели Visual Studio умеет различать код приложения, тестовые скрипты, веб-контент, JavaScript, скрипты сборки и другие типы файлов, что позволяет активировать соответствующие функции и инструменты для каждого из них. В Visual Studio есть инструмент, который упрощает работу с несколькими взаимосвязанными проектами, например, с Python-проектом. На рисунке 23 представлен пример

реализации Visual Studio, включающей в себя Python-проект и Flask-проект, отображенные в обозревателе решений.

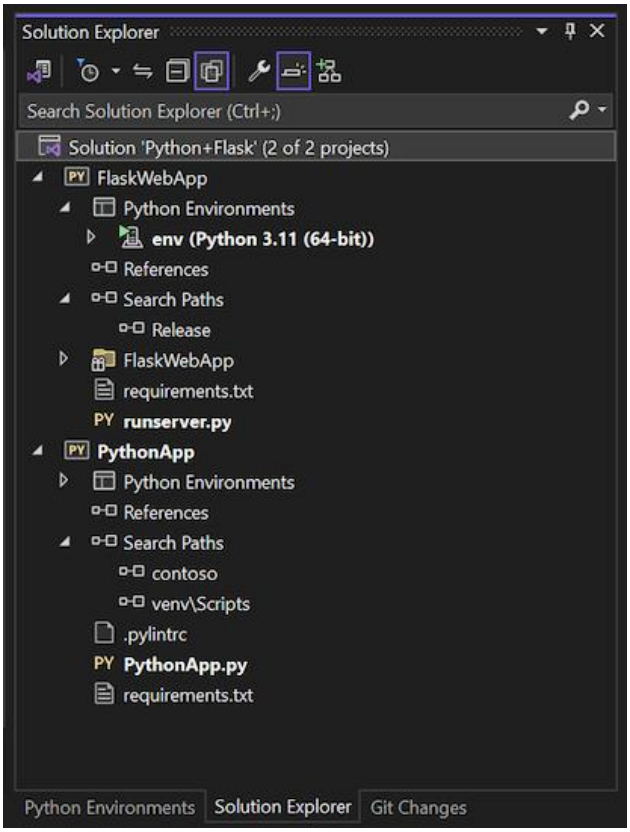

Рисунок 23 – Система проекта и шаблоны проектов и элементов

Использование шаблонов для проектов и элементов упрощает настройку разнообразных проектов и файлов. Они экономят ваше время и избавляют от необходимости ручного управления трудоемкими иerror-prone деталями. В Visual Studio доступны шаблоны для создания веб-приложений, проектов на Azure, систем обработки и анализа данных, консольных приложений и других видов проектов.

Visual Studio выделяется своим высокоэффективным отладчиком, который особенно силён при работе с Python. В его арсенале присутствуют такие функции, как отладка в смешанном режиме, удалённая отладка на Linux-системах, отладка в интерактивном окне и отладка модульных тестов, обеспечивая полную гибкость и удобство для разработчиков Python.

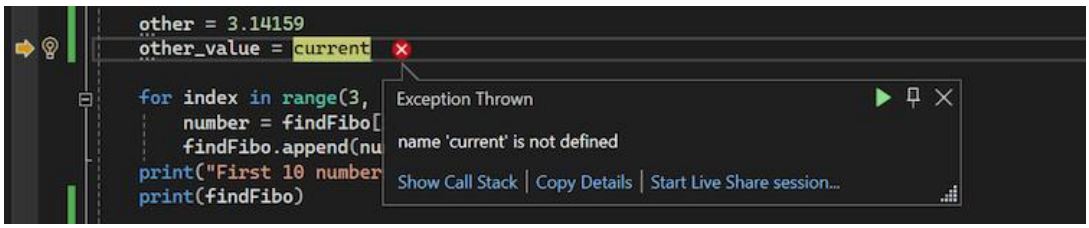

Рисунок 24 – отладчик Visual Studio Code

Для работы с многомерными массивами данных используется библиотека NumPy, обеспечивая фундаментальные операции линейной алгебры и высокоэффективные манипуляции с массивами. Библиотека NumPy содержит многомерные массивы и матричные структуры данных. NumPy можно использовать для выполнения множества математических операций с массивами. Он добавляет в Python мощные структуры данных, гарантирующие эффективные вычисления с массивами и матрицами, и предоставляет огромную библиотеку высокоуровневых математических функций, которые работают с этими массивами и матрицами.

Визуализация данных и результатов осуществляется с помощью Matplotlib. Это мощная библиотека Python, предоставляющая обширный функционал для создания высококачественных графиков и диаграмм. Она незаменима для анализа данных и эффективной визуализации полученных результатов. Matplotlib предлагает два основных подхода к работе: через модуль Pyplot, который позволяет рассматривать график как единое целое, и через объектно-ориентированный интерфейс, где каждый элемент графика (например, фигура, ось, линия) является независимым объектом. Последний подход даёт пользователю полный контроль над свойствами и отображением каждого компонента.

Для построения и обучения модели нейронной сети выбраны TensorFlow и Keras. TensorFlow – это мощная платформа для разработки глубоких нейронных сетей, поддерживающая автоматическое вычисление градиентов и оптимизацию, что делает её подходящей для сложных задач машинного обучения. Она активно используется для создания, обучения и развертывания моделей глубокого обучения.

Для разработки решений в области глубокого обучения и компьютерного зрения существует множество специализированных инструментов и библиотек. Ниже в таблице 2 представлен сравнительный обзор наиболее популярных из них, с выделением их ключевых особенностей, преимуществ и недостатков.

Таблица 2 – Сравнительный обзор библиотек для глубокого обучения и компьютерного зрения.

Библиотека	Основное назначение	Преимущества	Недостатки
1	2	3	4
TensorFlow	Мощная библиотека для глубокого обучения, разработанная Google, предоставляющая широкий спектр API для создания моделей любой сложности.	Мощные возможности: Высоко- и низкоуровневые API для гибкой разработки. Масштабируемость: поддержка мобильных, серверных и распределенных систем. Документация и сообщество: обширная документация, активное сообщество, множество готовых примеров.	Сложность освоения: высокая гибкость может быть трудна для новичков. Производительность: в некоторых сценариях может уступать PyTorch.

Продолжение таблицы 2

1	2	3	4
PyTorch	Открытая библиотека для глубокого обучения от Facebook, популярная среди исследователей за гибкость и интуитивность.	Динамические вычислительные графы: облегчают отладку и построение моделей. Активное сообщество: быстрый доступ к материалам, примерам, решениям. Интеграция с Python: отлично вписывается в экосистему Python для исследований и разработки.	Производительность: в некоторых задачах, особенно крупномасштабных, может уступать TensorFlow. Стабильность: некоторые пользователи отмечают проблемы с обновлениями библиотеки.
Keras	Высокоуровневый API, упрощающий построение и тренировку нейронных сетей, часто выступает интерфейсом к TensorFlow.	Простота использования: интуитивный синтаксис ускоряет создание и тестирование моделей. Интеграция: Полная совместимость с TensorFlow. Масштабируемость: поддержка GPU и распределенного обучения.	Ограниченная гибкость: менее гибок для нестандартных решений по сравнению с низкоуровневыми API. Производительность: высокий уровень абстракции может привести к снижению производительности в отдельных сценариях.

Продолжение таблицы 2

1	2	3	4
OpenCV	Библиотека с открытым исходным кодом для задач компьютерного зрения и машинного обучения, обработки и анализа изображений.	Широкий функционал: обширный набор инструментов для работы с изображениями и видео (фильтрация, обнаружение объектов и т.д.). Высокая производительность: оптимизирована для различных платформ (CPU/GPU), подходит для ресурсоемких задач. Документация и сообщество: хорошо документирована, поддерживается большим и активным сообществом.	Сложность освоения: обширный функционал может быть труден для новичков без опыта в компьютерном зрении. Интерфейс: менее интуитивен по сравнению с библиотеками, ориентированными только на глубокое обучение.
Numpy	Фундаментальная библиотека Python для численных вычислений;	Высокая производительность; Основа для других библиотек; Мощный синтаксис.	Сложность для нечисленных задач; Потребление памяти при работе с большими массивами.

Продолжение таблицы 2

1	2	3	4
Pillow	Библиотека Python для базовых операций с изображениями, часто используемая для предобработки данных в ML.	Простота использования: легкая интеграция с другими Python-библиотеками, интуитивный интерфейс для стандартных задач. Базовый функционал: поддержка основных операций (изменение размера, обрезка, фильтрация, преобразование форматов).	Ограниченные возможности: менее функциональна, чем OpenCV, не подходит для сложных задач компьютерного зрения. Производительность: может уступать другим библиотекам при выполнении комплексных операций с изображениями.

Для разработки графического пользовательского интерфейса используется Tkinter – стандартная библиотека Python для создания кроссплатформенных графических приложений. Tkinter позволяет создавать интуитивные и функциональные интерфейсы, обеспечивая взаимодействие пользователя с системой, включая загрузку изображений, отображение результатов классификации и управление историей обработки.

Преимущества Tkinter:

– простота использования: Tkinter встроен в стандартную библиотеку Python, что делает его доступным и легким для начала работы;

– кроссплатформенность: приложения, созданные с использованием Tkinter, работают на всех основных операционных системах (Windows, macOS, Linux) без необходимости внесения изменений в код;

– минимальные зависимости: поскольку Tkinter является частью стандартной библиотеки Python, он не требует установки дополнительных пакетов;

– гибкость: Tkinter предоставляет достаточно инструментов для создания простых и функциональных интерфейсов, включая кнопки, текстовые поля, метки и другие элементы управления.

Недостатки Tkinter:

– ограниченный дизайн: интерфейсы, созданные с помощью Tkinter, выглядят устаревшими по сравнению с современными графическими библиотеками;

– ограниченная функциональность: Tkinter подходит для создания простых интерфейсов, но может быть недостаточно мощным для реализации сложных и интерактивных приложений;

– сложность масштабирования: при увеличении сложности интерфейса код может стать громоздким и трудным для поддержки;

– отсутствие встроенной поддержки современных функций: Tkinter не поддерживает современные элементы интерфейса, такие как анимации или сложные визуальные эффекты, без дополнительных усилий.

В совокупности, выбор этих средств обоснован их широкими возможностями и высоким уровнем интеграции в процессе разработки, что обеспечивает создание эффективной, производительной и удобной для пользователя системы классификации клеток крови.

3 РАЗРАБОТКА ПРОГРАММЫ ДЛЯ КЛАССИФИКАЦИИ ПОДГРУПП ЛЕЙКОЦИТОВ НА ИЗОБРАЖЕНИЯХ

3.1. Этапы разработки программного продукта и реализация нейросетевой модели

3.1.1 Функциональная декомпозиция системы

Программа включает несколько ключевых модулей, которые взаимодействуют друг с другом для выполнения задачи сегментации клеток крови с использованием нейронной сети. Каждый из этих модулей выполняет свою специфическую роль, что обеспечивается их интеграцией и взаимодействием внутри системы. Рассмотрим подробнее функциональное взаимодействие различных компонентов программы:

– нейронная сеть является основным элементом системы. Она отвечает за процесс обучения, сегментацию изображений клеток крови, а также сохранение результатов предсказаний и модельных параметров;

– модуль обучения нейронной сети инициирует процесс подготовки модели. При активации этого модуля пользователь может выбрать базу данных изображений клеток крови для обучения, настроить параметры сети (например, количество эпох), а также выбрать, хочет ли он отображать графики или отчеты о процессе обучения;

– модуль тренировки нейронной сети запускает сам процесс тренировки модели. Пользователь выбирает тренировочную базу данных, на которой нейронная сеть будет обучаться и тестироваться. Модуль также позволяет контролировать прогресс тренировки, включая отображение потерь и точности в реальном времени;

– модуль загрузки изображения для анализа предоставляет пользователю возможность загрузить новое изображение клеток крови, которое будет обработано моделью для выполнения сегментации;

– панель отображения входного изображения позволяет пользователю увидеть загруженное изображение клеток крови перед его сегментацией, что удобно для предварительного анализа данных;

– панель вывода результатов сегментации отображает результаты работы нейронной сети – например, сегментированные области клеток, а также вероятность или уверенность модели в правильности сегментации;

– панель истории предыдущих анализов сохраняет информацию о всех предыдущих изображениях и результатах их сегментации, что позволяет пользователю отслеживать прогресс и анализировать эффективность модели на разных этапах.

Функциональная декомпозиция представлена на рисунке 25.

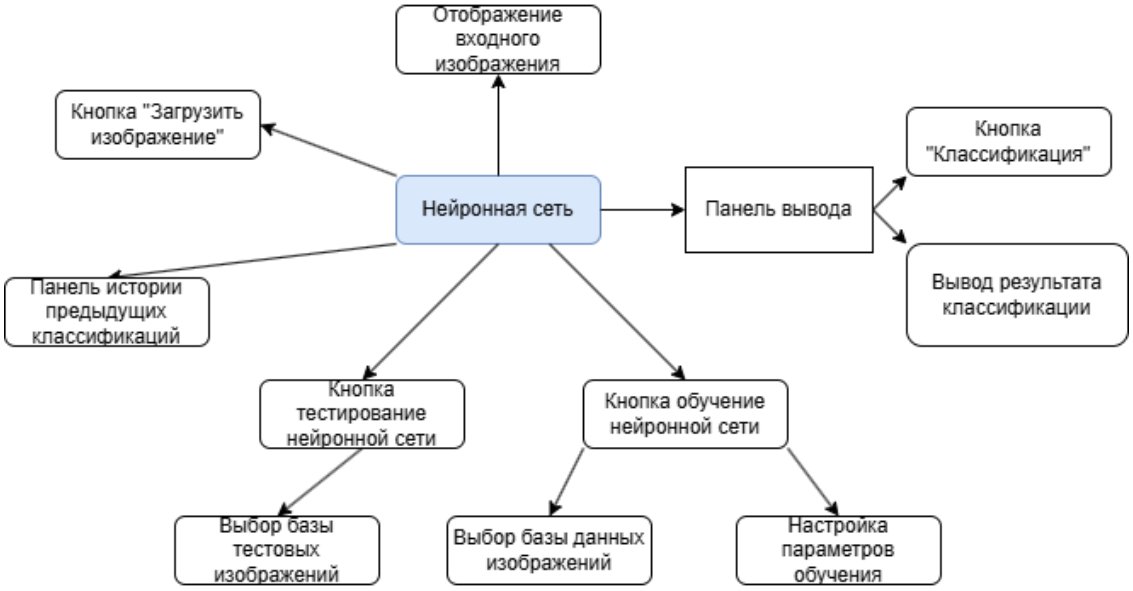

Рисунок 25 – Функциональная декомпозиция

Архитектура приложения для разработки системы классификации клеток крови с использованием нейронной сети основывается на модульной структуре, что обеспечивает чёткое разделение логики, алгоритмов обработки изображений и пользовательского интерфейса. Этот подход упрощает процесс разработки, улучшает тестирование и облегчает дальнейшее сопровождение программы.

Программа состоит из следующих основных модулей:

Learning.py – отвечает за процесс обучения нейронной сети, включая компиляцию модели, обучение на тренировочном датасете и сохранение результатов.

DatasetLoader.py – реализует загрузку изображений и их меток, применяя указанные этапы предобработки.

PreProcessor.py – выполняет предварительную обработку изображений, включая изменение размера и обрезку по маске.

ImageToArray.py – преобразует изображения в формат массива, совместимый с библиотекой Keras.

IncludeNet.py – содержит определение архитектуры нейронной сети, используемой для классификации лейкоцитов.

WBC-Detection.py – реализует алгоритм сегментации лейкоцитов на изображениях с помощью обработки цвета и контуров.

Load_test_model.py – загружает обученную модель и выполняет тестовую классификацию изображений из пользовательского набора.

TRAIN и TEST – каталоги, представляют базы данных изображений, используемые для обучения и тестирования соответственно.

Interface.py – модуль, отвечающий за пользовательский интерфейс. Этот модуль обеспечивает взаимодействие пользователя с системой, предоставляя интерфейс для загрузки изображений, отображения результатов сегментации и настройки параметров нейронной сети.

На рисунке 26 представлена диаграмма компонентов, в которой показаны ключевые модули программы и их взаимосвязи.

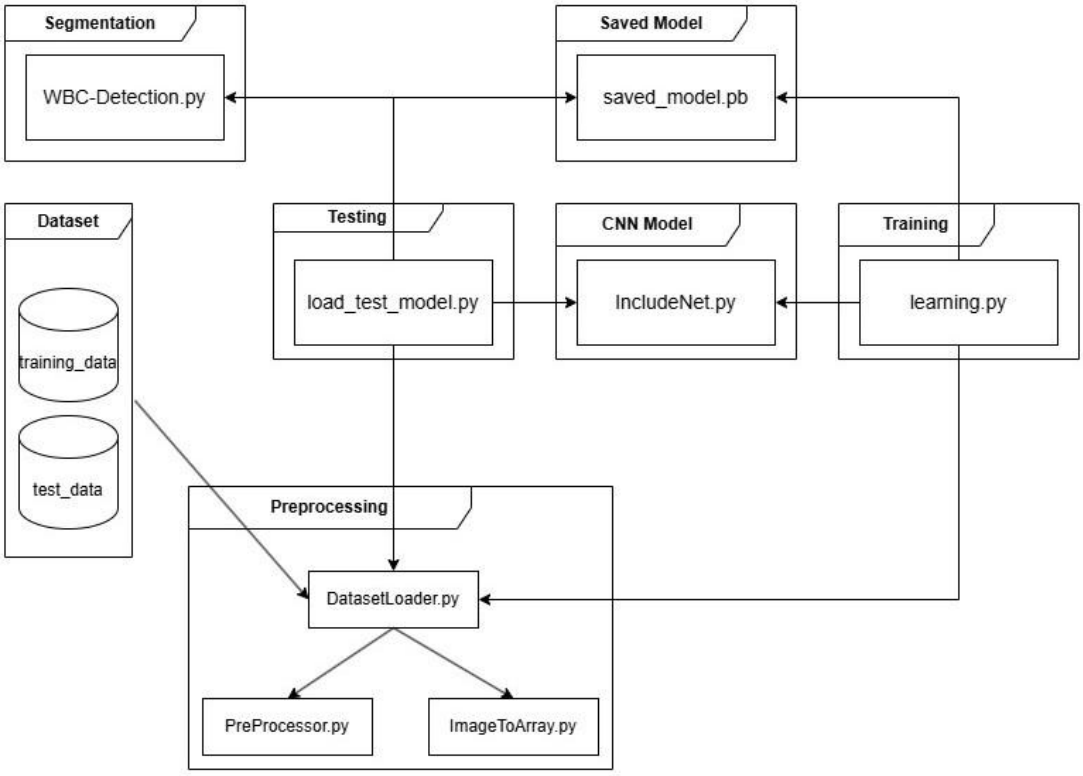

Рисунок 26 – Диаграмма компонентов

Модуль learning.py является основным компонентом программного продукта и отвечает за обучение сверточной нейронной сети на размеченных изображениях лейкоцитов. Этот модуль осуществляет полный цикл подготовки данных, компиляции модели, процесса обучения и последующего сохранения обученной нейросети.

Ключевые этапы, реализуемые в данном модуле:

Загрузка и подготовка обучающего и тестового набора данных – с помощью функций библиотеки imutils.paths производится автоматизированный сбор путей ко всем изображениям в указанных директориях обучения и тестирования. Для обработки изображений используется модуль DatasetLoader.py, которому передаются предварительные обработчики (PreProcessor и ImageToArrayPreprocessor), реализующие последовательную трансформацию изображений.

Объекты классов PreProcessor и ImageToArrayPreprocessor обрабатывают изображения следующим образом:

– PreProcessor выделяет контур ядра лейкоцита, центрирует его и приводит к заданному размеру;

– ImageToArrayPreprocessor преобразует изображение в формат массива, совместимый с Keras/TensorFlow.

Нормализация данных – все изображения масштабируются к диапазону значений от 0 до 1, что является стандартной практикой при работе с нейросетями и позволяет повысить стабильность и скорость сходимости модели.

Построение и компиляция модели – архитектура сверточной нейронной сети формируется посредством вызова метода build() из модуля IncludeNet. В качестве функции потерь используется categorical_crossentropy, а в качестве оптимизатора – Adam, что обеспечивает устойчивое и быстрое обучение многоклассовой модели. Модель также настраивается на метрику точности (accuracy), необходимую для оценки качества классификации.

Обучение модели – модель обучается на подготовленных данных с использованием заданного количества эпох и размера пакета (batch size). В процессе обучения ведётся логирование потерь и точности как на обучающей, так и на валидационной выборке. Это позволяет отслеживать переобучение или недообучение модели.

Сохранение модели и визуализация результатов – по завершению процесса обучения обученная модель сохраняется на диск в формате HDF5. Кроме того, при помощи библиотеки matplotlib строятся графики изменения функции потерь и точности по эпохам, что является важным элементом анализа качества обучения.

Модуль learning.py интегрирует в себе функции и классы из следующих вспомогательных модулей:

– DatasetLoader – обеспечивает загрузку изображений и распределение их по признаковым и целевым массивам (data и labels);

– PreProcessor – производит пространственную фильтрацию и обрезку изображения для фокусировки на ядре лейкоцита;

– ImageToArray – подготавливает изображения к передаче в модель;

– IncludeNet – задаёт архитектуру сверточной нейросети, используемой для классификации;

– tensorflow.keras – предоставляет средства компиляции, обучения и сохранения модели;

– matplotlib.pyplot – используется для визуализации результатов обучения.

Таким образом, learning.py является главным модулем для всех процессов обучения модели – от подготовки данных до финального сохранения модели и визуального анализа результатов.

Модуль DatasetLoader.py реализует специализированный класс DatasetLoader, предназначенный для централизованной загрузки и предобработки изображений, подготавливаемых к обучению и тестированию сверточной нейронной сети. Данный модуль используется на этапе подготовки данных, как в процессе обучения модели (learning.py), так и при тестировании на новых данных (load_test_model.py).

Класс DatasetLoader обладает следующим функционалом:

– чтение изображений с диска по предоставленным путям;

– формирование входных и целевых массивов данных (data, labels);

– вызов указанных пользователем предобработчиков изображений (при необходимости);

– классификация каждого изображения на основе структуры каталогов (имя родительской папки);

– возврат подготовленных данных в виде кортежа (data, labels).

Класс DatasetLoader содержит два метода:

Конструктор класса, принимающий на вход список объектов-предобработчиков, реализующих интерфейс. Если список не передан, используется None, что означает отсутствие предобработки.

Основной метод, реализующий пошаговую загрузку и обработку изображений. Он выполняет следующие действия:

– инициализация пустых списков data и labels;

– итерация по каждому пути в imagePaths;

– чтение изображения с помощью библиотеки OpenCV;

– извлечение метки класса из имени родительской папки изображения;

– применение всех переданных предобработчиков к изображению (если они заданы);

– добавление обработанного изображения в список data, а метки – в labels;

– вывод промежуточной информации в консоль, если задан соответствующий параметр.

Модуль DatasetLoader.py активно используется в файле learning.py, где он участвует в формировании обучающего и тестового выборок.

Модуль DatasetLoader.py также тесно связан со следующими компонентами системы:

– PreProcessor.py – реализует пространственную фильтрацию, выделение контура и кадрирование области интереса (ядра клетки);

– ImageToArray.py – преобразует изображения в массивы NumPy, совместимые с форматами, используемыми в Keras/TensorFlow;

– learning.py и load_test_model.py – используют DatasetLoader как инструмент для получения нормализованных и размеченных данных для обучения и верификации.

Модуль DatasetLoader.py является важной частью архитектуры системы, обеспечивая корректную загрузку и предобработку изображений. Его использование позволяет легко масштабировать систему под новые форматы данных, а также обеспечивает модульность и повторное использование кода в различных этапах жизненного цикла проекта.

Модуль PreProcessor.py реализует специализированный класс PreProcessor, основной задачей которого является пространственная предобработка изображений перед подачей их в нейронную сеть. Данный модуль используется для подготовки входных данных, как на этапе обучения модели, так и при тестировании.

Целью использования данного модуля является локализация ключевого объекта на изображении, в частности лейкоцита, его вырезание и масштабирование до заданных размеров. Такая предобработка способствует повышению точности классификации, так как сеть обучается только на информативной области изображения.

Класс PreProcessor содержит следующие компоненты:

Конструктор __init__(self, width, height, inter=cv2.INTER_AREA).

Принимает параметры:

– width и height – размеры, до которых будет приведено изображение;

– inter – алгоритм интерполяции.

Метод preprocess(self, input_image).

Основной метод обработки изображения, включающий следующие этапы:

– применение Гауссова размытия: низкочастотный фильтр, который сглаживает неравномерные значения пикселей изображения, обрезая самые высокие значения;

– преобразование в HSV-пространство: обеспечивает лучшую сегментацию цветовых диапазонов;

– создание маски по заданному цветовому диапазону: выделяет потенциальную зону интереса;

– нахождение самого большого контура: Вызывается вспомогательный метод, возвращающий контур и маску;

– вычисление ограничивающего круга и обрезка изображения по области интереса;

– масштабирование до целевых размеров.

При возникновении ошибки метод возвращает None, предотвращая сбой выполнения программы.

Метод find_biggest_contour(self, image).

Используется для нахождения самого большого контура на бинарной маске. Этот контур предполагается как область с ядром лейкоцита. Возвращается также маска, построенная по найденному контуру.

Метод overlay_mask(self, mask, image).

Накладывает маску на изображение с использованием прозрачности. Применяется для визуализации результата сегментации, может быть активирован по необходимости.

Модуль PreProcessor.py тесно взаимодействует с:

– DatasetLoader.py – передаётся в виде одного из предобработчиков (preprocessors), вызывается для каждого изображения;

– ImageToArray.py – преобразует уже предобработанные изображения в формат, пригодный для подачи в нейросеть;

– learning.py, load_test_model.py – обеспечивают вызов и использование PreProcessor для обучения и предсказания.

Модуль ImageToArray.py реализует класс ImageToArrayPreprocessor, предназначенный для приведения входных изображений к формату, пригодному для подачи в сверточную нейронную сеть. Данный этап является заключительным в цепочке предобработки данных и используется как при обучении модели, так и на стадии конечного результата обработки данных

Цель использования ImageToArray.py – преобразовать объект изображения, представленного в формате NumPy-массива с обычной структурой (например, Height x Width x Channels), в тензор – многомерный массив, совместимый с архитектурой TensorFlow/Keras. Это необходимо, поскольку модели нейронных сетей требуют, чтобы входные данные имели строго определенный формат и структуру.

Класс ImageToArrayPreprocessor состоит из следующих элементов:

Конструктор __init__(self, data_format=None) – принимает необязательный параметр data_format, который определяет порядок каналов в изображении.

Метод preprocess(self, image) – основной метод класса, преобразующий изображение с использованием функции img_to_array() из библиотеки keras.

Вызов метода возвращает изображение в виде тензора, подходящего для подачи в модель. На выходе получается тензор, который затем может быть нормализован, объединён в батчи и использован в процессе обучения модели.

Модуль ImageToArray.py используется в составе списка предобработчиков в модуле DatasetLoader.py. Его задача – преобразование каждого изображения после предварительной обрезки и масштабирования.

ImageToArray.py является сравнительно простым по реализации, но выполняет важную функцию адаптации изображения к формату, который понимает TensorFlow. Без этого этапа невозможно было бы сформировать корректный батч данных и передать его в модель. Кроме того, модуль сделан обобщённым, с возможностью указания формата данных, что делает его переносимым и совместимым с различными конфигурациями нейронных сетей.

Модуль WBC-Detection.py выполняет вспомогательную функцию в рамках системы визуального определения и классификации лейкоцитов. Он предназначен для тестирования и визуализации основных этапов обработки изображений, в частности, для выделения лейкоцита на изображении и анализа его геометрических характеристик. Модуль не участвует напрямую в обучении нейронной сети, но играет важную роль в верификации этапов предварительной обработки и локализации объекта интереса.

Модуль реализует последовательность процедур обработки одного изображения:

– подавление шума (размытие);

– преобразование цветового пространства;

– выделение цветового диапазона лейкоцита;

– нахождение самой крупной области на изображении;

– визуализацию результатов обработки, включая: маску выделенной области, центр масс выделенного объекта и эллипс, аппроксимирующий форму клетки.

Модуль содержит несколько вспомогательных функций:

Принимает бинарное изображение после фильтрации по цвету и находит на нём контур с максимальной площадью. Возвращает найденный контур и бинарную маску, содержащую только его.

Накладывает бинарную маску на исходное изображение, позволяя наглядно выделить интересующую область. Используется исключительно в целях визуализации.

Отображения изображений и масок с помощью matplotlib.pyplot.

Модуль читает изображение и последовательно применяет к нему следующие преобразования:

– чтение и преобразование цветового пространства;

– размытие изображения для подавления шума;

– перевод в HSV и выделение цветового диапазона, соответствующего лейкоциту;

– поиск основного контура и вычисление геометрических характеристик;

– визуализация результатов (центр масс, эллипс): центр масс отображается как зелёный круг, эллипс, аппроксимирующий форму клетки, рисуется поверх изображения, а также создается маска, по которой производится выделение лейкоцита;

– обрезка области интереса;

– масштабирование обрезанного изображения: изображение с выделенным лейкоцитом масштабируется до ширины 100 пикселей для унификации при отображении.

Модуль WBC-Detection.py можно отнести к категории инструментов для исследования и отладки. Он предоставляет разработчику визуальные подтверждения правильности работы алгоритмов предобработки изображений и сегментации лейкоцитов. Несмотря на отсутствие прямой связи с процессом обучения, данный модуль играет ключевую роль в отладке параметров выделения объектов на изображениях.

Также часть функционала модуля встраивается и в другие модули проекта, например, в PreProcessor.py или load_test_model.py, повторяя ту же логику обработки.

Модуль IncludeNet.py реализует класс IncludeNet, в котором определена архитектура сверточной нейронной сети, используемой для классификации изображений лейкоцитов. Данный модуль инкапсулирует модель и предоставляет статический метод build, возвращающий готовую к обучению или предсказаниям нейросеть, построенную с использованием Keras API.

Основная функция модуля – метод build(width, height, depth, classes) создает и возвращает последовательную (Sequential) модель нейронной сети, в которую включены сверточные слои (Conv2D), слои активации (ReLU), подвыборки (MaxPooling2D), регуляризации (Dropout) и полносвязный выходной слой с функцией активации softmax.

Модуль IncludeNet.py не используется напрямую, а импортируется в основной модуль обучения learning.py, где вызывается метод IncludeNet.build() для создания модели, которая далее компилируется и обучается. В частности:

– метод IncludeNet.build в learning.py вызывается с параметрами, соответствующими размеру входных изображений и количеству классов изображений, после чего возвращённая модель компилируется и передаётся в обучение;

– архитектура модели также применяется при загрузке предобученной модели в модуле load_test_model.py, где используется уже сохранённый файл .h5.

Таким образом, IncludeNet.py играет роль ядра всей классификационной части проекта – он изолирует определение модели, позволяя другим компонентам использовать её без необходимости дублировать архитектуру или её логику. Это повышает модульность и читаемость кода.

Модуль load_test_model.py реализует механизм загрузки предобученной модели нейронной сети и тестирования её на случайно выбранных изображениях из пользовательской выборки. Он представляет собой автономный скрипт,

предназначенный для оценки качества классификации лейкоцитов после завершения этапа обучения модели.

Модуль выполняет следующие ключевые действия:

– загрузка предобученной модели – с использованием функции load_model из библиотеки keras.models, в память загружается модель, ранее обученная и сохранённая в формате .h5 или .pb. Путь к модели передаётся в аргументе командной строки --model.

– подготовка и предобработка изображений – изображения случайным образом выбираются из тестового датасета, определяется аргументом --dataset. Для этого используется метод sampling_images, который:

– считывает случайные пути к изображениям;

– обрабатывает изображения с помощью классов PreProcessor и ImageToArrayPreprocessor;

– преобразует изображения в формат, совместимый с нейронной сетью, и нормализует значения пикселей в диапазон [0, 1].

– предсказание класса – загруженная модель используется для классификации изображений. Метод predicting производит предсказания и отображает каждое изображение с наложенной текстовой меткой, соответствующей предсказанному классу.

– визуализация результата – для наглядного отображения результатов применяется библиотека OpenCV. Каждое изображение сопровождается подписью с предсказанным классом, и выводится в отдельном окне.

Модуль активно взаимодействует с рядом других компонентов проекта:

– IncludeNet.py: опосредованно – при сохранении и загрузке модели, архитектура которой определена именно в IncludeNet. Это обеспечивает согласованность структуры нейросети при обучении и последующем тестировании;

– DatasetLoader.py: используется для загрузки и маркировки изображений тестовой выборки;

– PreProcessor.py и ImageToArray.py: применяются в цепочке предобработки изображений перед подачей их в модель. PreProcessor выделяет наиболее

значимую область (лейкоцит) и нормализует изображение по размеру. ImageToArrayPreprocessor преобразует изображение в формат массива, совместимый с Keras;

– внутри модуля также реализованы локальные вспомогательные функции ready_to_use_images, overlay_mask, find_biggest_contour, дублирующие функциональность WBC-Detection.py. Это частично снижает модульность, но делает модуль независимым и удобным для быстрого запуска.

Таким образом, load_test_model.py играет ведущую роль в финальной проверке качества разработанной нейросети и служит инструментом визуального контроля точности классификации.

Такое разделение на компоненты соответствует принципам модульного проектирования, обеспечивая чёткое распределение задач между элементами системы. Модульная архитектура упрощает процесс разработки, тестирования и поддержания программы, позволяя гибко вносить изменения и улучшения в отдельные компоненты без воздействия на всю систему.

3.1.2 Подготовка данных

Разработка системы классификации лейкоцитов требует использования заранее подготовленного набора данных, включающего изображения клеток крови и их соответствующие метки. В данном исследовании используется датасет, содержащий 12 500 изображений клеток крови в формате JPEG. Данные организованы в виде пяти отдельных папок, каждая из которых соответствует определенному типу клеток.

В данной работе используется набор данных Blood Cell Detection Dataset от Kaggle, который включает в себя коллекцию микроскопических изображений клеток крови. Набор данных обеспечивает основу для обучения модели сегментации и идентификации различных типов клеток крови. Этот набор данных содержит аннотированные эритроциты (RBC) и лейкоциты (WBC) из мазка периферической крови, взятого с помощью светового микроскопа.

Исходные изображения представляют собой цветные снимки размером 256 × 256 пикселей в формате RGB. Они демонстрируют клетки крови в мазке с

различными уровнями увеличения. В наборе данных представлены изображения пяти типов лейкоцитов: Эозинофилы, Базофилы, Нейтрофилы, Лимфоциты, Моноциты. Таким образом, в задачи классификации входит распознавание пяти классов лейкоцитов, а также фона, что в сумме формирует шесть категорий.

В качестве базовой архитектуры для сегментации клеток крови выбрана модифицированная версия VGG-16. Данная сверточная нейросеть была выбрана благодаря её высокой эффективности в задачах сегментации медицинских изображений.

Основные характеристики модели:

– входное изображение: 256 × 256 пикселей, RGB;

– выходное изображение: 256 × 256 пикселей (семантическая карта сегментации);

– определение различных областей на изображении (лейкоциты, эритроциты, фон);

– окрашивание сегментированных объектов в различные цвета для наглядного отображения.

Процесс обучения моделей включал следующие основные этапы:

– настройка параметров: для каждой модели были заданы ключевые параметры, такие как скорость обучения, количество эпох, функция потерь и оптимизационный алгоритм;

– использование GPU: при наличии графического процессора обучение ускорялось за счет аппаратного ускорения;

– итеративное обучение: на каждой эпохе модель обучалась на тренировочном наборе данных, а оптимизатор корректировал её параметры для минимизации ошибки;

– процесс валидации: после каждой эпохи модель тестировалась на отдельном наборе данных, чтобы оценить её точность и общую производительность;

– мониторинг и сохранение: фиксировались метрики классификации для каждого класса, а также сохранялись веса модели. Такой подход позволил де-

тально проанализировать эффективность разработанной архитектуры сверточной нейросети в задаче классификации изображений.

3.1.3 Разработка нейросетевой архитектуры

В рамках данной работы была реализована сверточная нейронная сеть, предназначенная для классификации изображений лейкоцитов. Архитектура сети реализована в модуле IncludeNet.py с использованием Keras API – Sequential модель. Данная архитектура относится к классу классических последовательных сверточных сетей и по своей структуре приближена к модифицированному варианту архитектуры VGG-16.

Общая структура сети представлена в таблице 3:

Таблица 3 – Описание архитектуры нейронной сети

№	Тип слоя	Параметры	Выходная форма
1	Conv2D	50 фильтров, (3×3), ReLU	50×50×50
2	MaxPooling2D	(3×3)	~16×16×50
3	Conv2D	50 фильтров, (3×3), ReLU	16×16×50
4	MaxPooling2D	(3×3)	~5×5×50
5	Conv2D	50 фильтров, (3×3), ReLU	5×5×50
6	MaxPooling2D	(3×3)	~1×1×50
7	Dropout	0.5	1×1×50
8	Conv2D	50 фильтров, (3×3), ReLU	1×1×50
9	Flatten	–	50
10	Dense	5 выходов (softmax)	5 (вероятностей классов)

Сеть состоит из следующих блоков:

– четыре сверточных слоя (Conv2D) с ядрами размером 3×3 и функцией активации ReLU;

– после трёх первых сверточных слоёв применяются операции субдискретизации (MaxPooling2D) с размером ядра 3×3 для уменьшения размерности признаков;

– между третьим и четвёртым сверточными слоями добавлен слой Dropout с вероятностью 0.5 для предотвращения переобучения;

– финальная часть сети включает слой Flatten для преобразования многомерного тензора в одномерный вектор и полносвязный слой (Dense), на выходе которого применяется функция активации softmax для получения распределения вероятностей по классам.

Таким образом, построенная сеть обладает умеренной глубиной и предназначена для эффективной работы с медицинскими изображениями небольшого размера (50×50 пикселей), обеспечивая достаточную обобщающую способность для задачи классификации лейкоцитов по пяти основным типам: эозинофилы, лимфоциты, моноциты, нейтрофилы и базофилы.

3.1.4 Описание процесса обучения модели

Обучение нейронной сети является одним из основных этапов разработки системы автоматической классификации лейкоцитов. В рамках данной работы процесс обучения реализован в модуле learning.py, который управляет полной процедурой подготовки модели, её компиляции, обучения на тренировочном наборе данных и сохранения результатов.

Для задачи классификации клеток крови используется стандартная функция потерь категориальная кросс-энтропия (categorical_crossentropy). Она наилучшим образом подходит для многоклассовых задач классификации, где классы взаимоисключающие, в данном случае это пять типов лейкоцитов.

В качестве оптимизатора выбран Adam, который является разновидностью оптимизационного алгоритма стохастичнского градиентного спуска.

Во время обучения Adam итеративно обновляет параметры модели, используя информацию из прошлых градиентов. Он хранит два скользящих средних для каждого параметра: оценку первого момента (среднее значение градиентов) и оценку второго момента (нецентрированная дисперсия градиентов). Эти моменты помогают адаптировать скорость обучения индивидуально для каждого параметра.

Параметры, получающие большие или частые обновления градиента, получают меньшую скорость обучения, а параметры с малыми или нечастыми обновлениями – большую. Такая адаптивная природа часто приводит к более бы-

строй сходимости по сравнению со стандартным стохастическим градиентным спуском. Adam автоматически адаптирует скорость обучения для каждого параметра и стабильно показывает хорошие результаты в задачах компьютерного зрения.

Обучение проводилось с использованием следующих гиперпараметров:

– размер батча (batch size): 50;

– количество эпох (epochs): 100;

– начальная скорость обучения (learning rate): 0.001;

– метрика точности (accuracy): используется в качестве основной метрики оценки качества обучения модели.

Также использовалась кросс-валидация на базе тренировочного и тестового подмножеств для оценки обобщающей способности модели.

Архитектура сети определяется в модуле IncludeNet.py, а затем инициализируется в learning.py вызовом соответствующего метода. Далее модель компилируется с использованием выбранной функции потерь и оптимизатора.

Перед подачей в сеть изображения проходят цепочку предобработки с помощью модулей PreProcessor.py и ImageToArray.py, объединённых в загрузчике DatasetLoader.py.

В данной реализации не применялись предобученные модели. Все веса и параметры нейронной сети инициализировались случайным образом, что позволило построить модель, адаптированную под специфику выбранной задачи. Такой подход целесообразен в случаях, когда доступен специфический набор изображений, в частности, микроснимки клеток крови, структура которых может существенно отличаться от стандартных датасетов, использующихся при предварительном обучении универсальных моделей.

Для отслеживания процесса обучения использовался механизм логирования библиотеки Matplotlib, а также встроенный вывод через Keras. Модуль learning.py сохраняет историю обучения, которая содержит значения функции потерь и точности на каждой эпохе.

По завершении обучения формируются графики зависимости точности и функции потерь от номера эпохи, что позволяет оценить динамику сходимости модели и выявить возможное переобучение. Пример графика при количестве эпох = 100, и размера батча = 50 можно увидеть на рисунке 27.

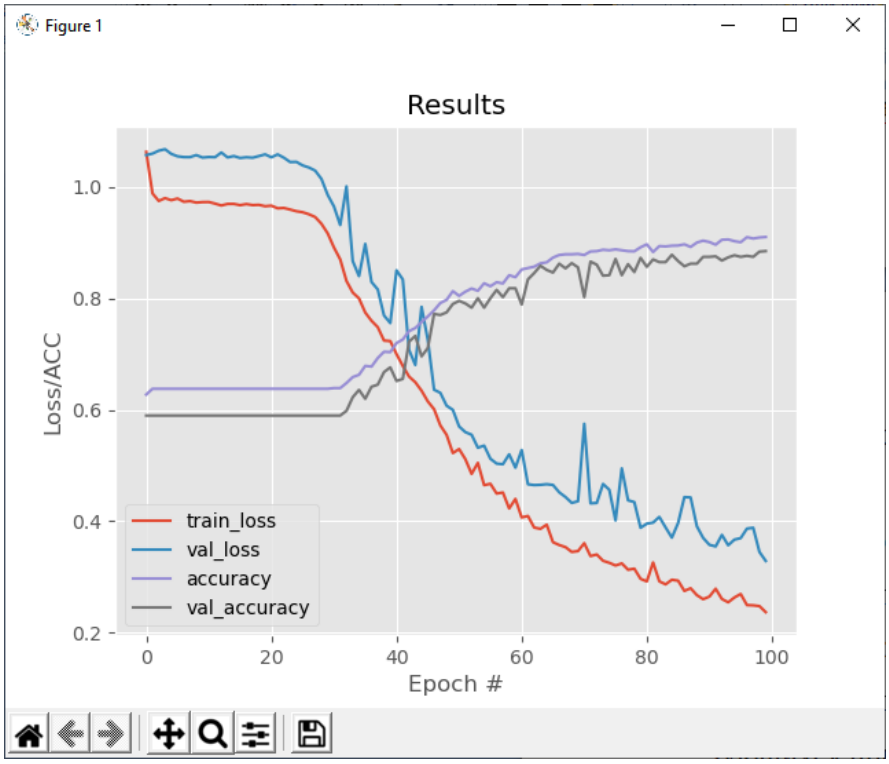

Рисунок 27 – Результат обучения

Графики сохраняются на диск и также визуализируются с помощью функции plt.show(). Это обеспечивает наглядное представление об эффективности обучения и возможностях модели к обобщению.

После завершения процесса обучения модель может быть сохранена в формате HDF5 или .pb. Это позволяет в дальнейшем загружать модель для тестирования или внедрения в практическое приложение без необходимости повторного запуска обучения.

3.2 Тестирование системы и анализ результатов классификации лейкоцитов

3.2.1 Процесс сегментации как часть обработки изображения перед классификацией

После выбора архитектуры нейросети и подготовки данных важным этапом обработки изображений является сегментация – процесс выделения облас-

тей, содержащих объекты интереса. В данном случае перед классификацией лейкоцитов необходимо сначала обнаружить их на изображении. Этот процесс осуществляется с использованием сверточной нейронной сети, которая обучена отделять лейкоциты от фона и других элементов мазка крови.

Корректная сегментация позволяет повысить точность последующей классификации, так как исключает влияние постороннего фона. Ниже приведён подробный алгоритм сегментации изображения, реализованный в модуле WBC-Detection.py.

На первом этапе изображение загружается с помощью OpenCV и преобразуется в цветовое пространство RGB, необходимое для корректной работы с цветами при последующих преобразованиях. На рисунке 28 представлено исходное изображение.

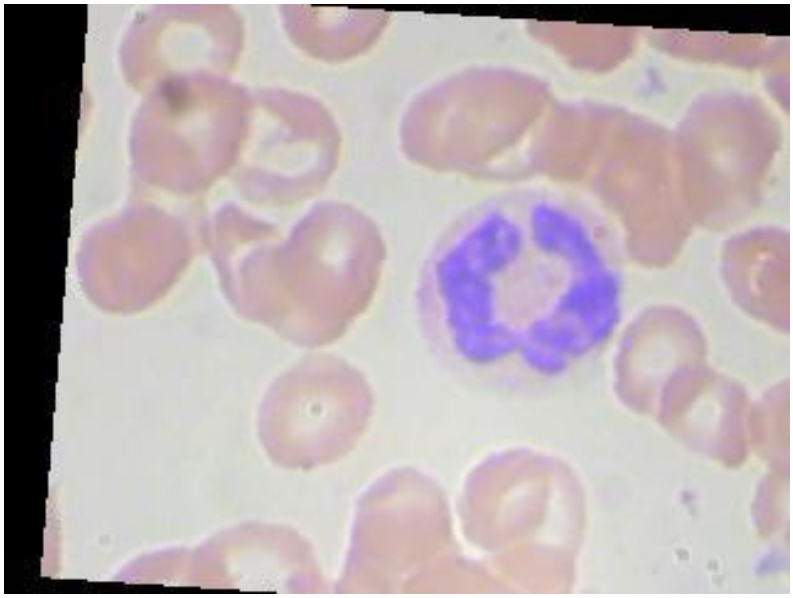

Рисунок 28 – Исходное изображение

Затем к изображению применяется гауссово размытие, рисунок 29, что позволяет сгладить шумы и мелкие детали, мешающие точному выделению маски.

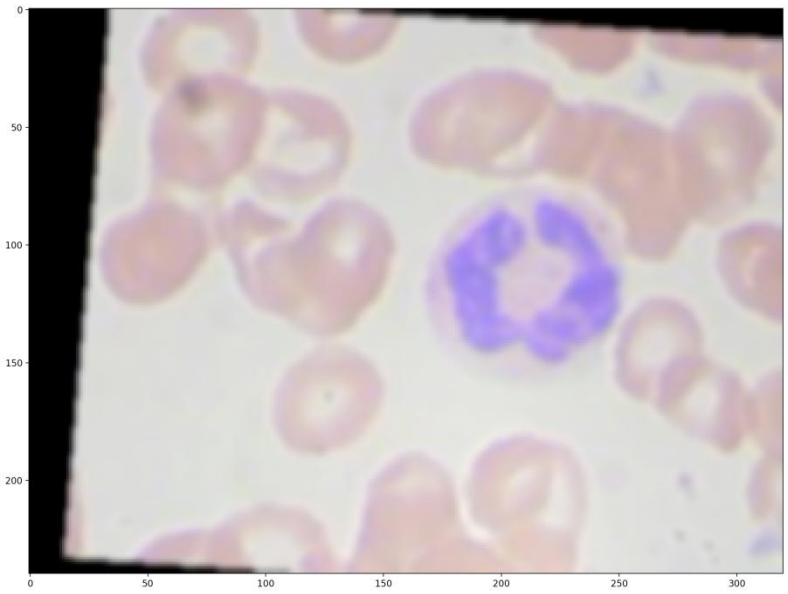

Рисунок 29 – Исходное изображение после применения гауссова размытия

После сглаживания изображение переводится в цветовое пространство HSV, которое более устойчиво к изменениям освещённости. Далее производится бинаризация изображения по заданному диапазону оттенков, соответствующих предполагаемому цвету ядра клетки. Данные изменения отражены на рисунке 30.

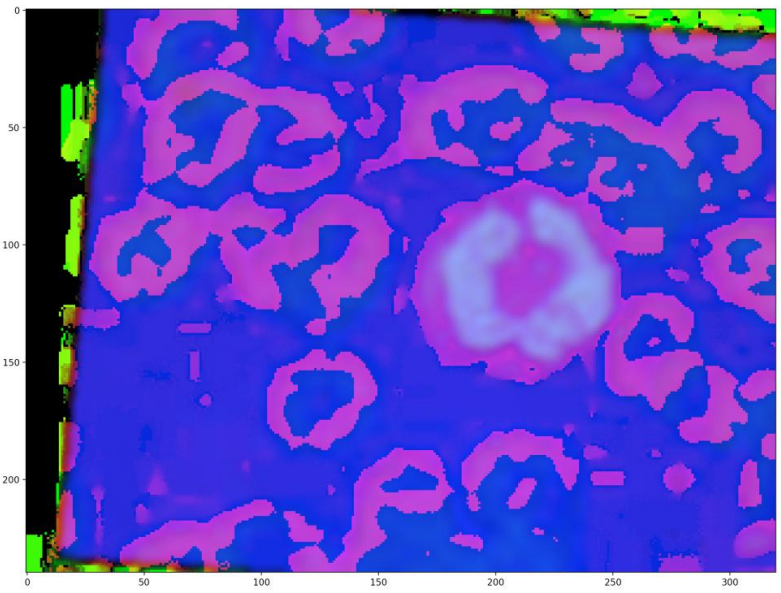

Рисунок 30 – Бинаризованное изображение (маска), полученное по цвету ядра клетки

Среди всех найденных контуров выбирается наибольший по площади, предполагая, что он соответствует ядру лейкоцита. На его основе создаётся бинарная маска, рисунок 31.

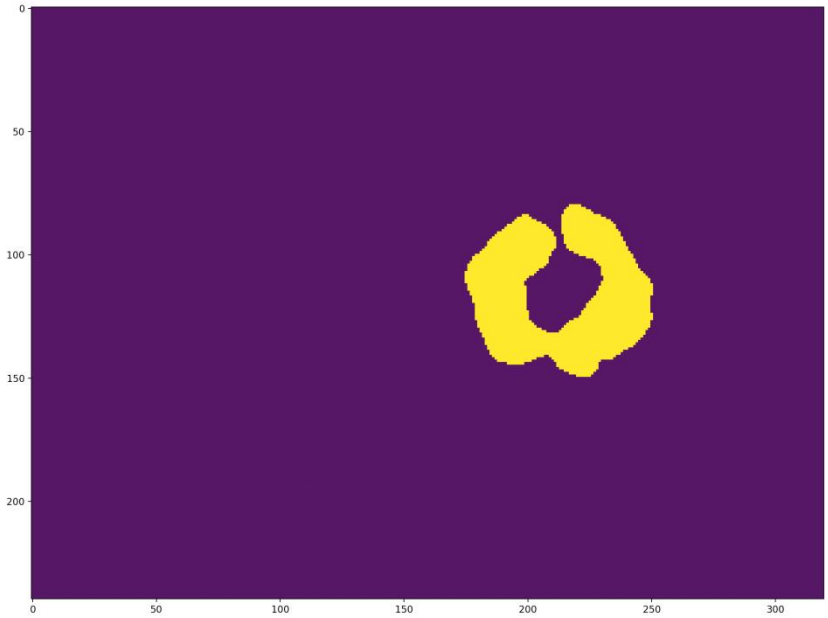

Рисунок 31 – Контур предполагаемого ядра клетки, выделенный на изображении

Сформированная маска накладывается на исходное изображение, позволяя визуально выделить область интереса, рисунок 32.

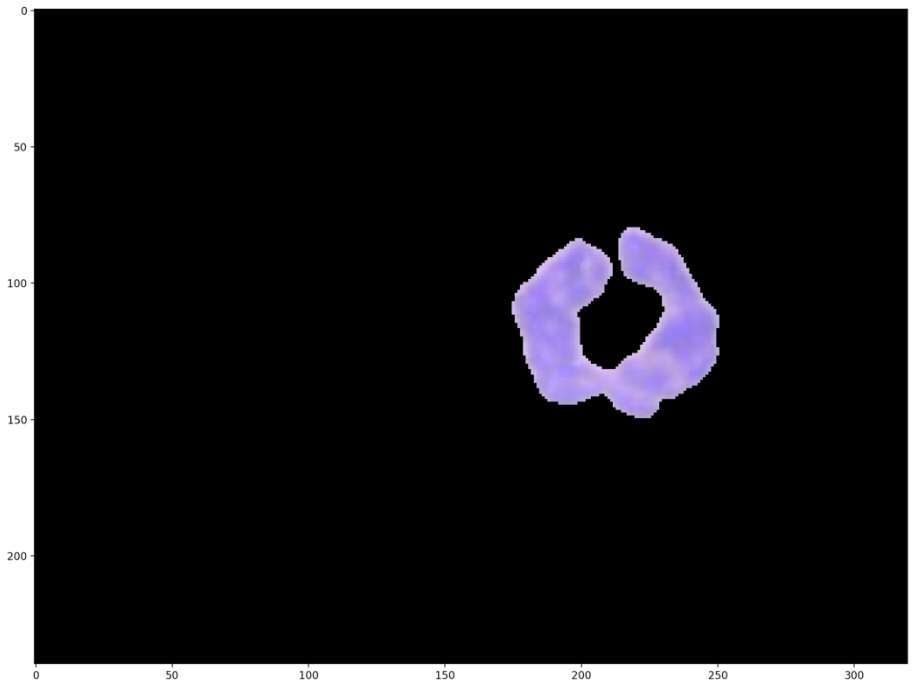

Рисунок 32 – Наложение маски ядра клетки на оригинальное изображение

Вычисляется центр масс области, соответствующей ядру клетки, который используется для дополнительной визуализации. Далее, для удобства анализа и возможного дальнейшего извлечения признаков применяется эллиптическая аппроксимация найденного контура.

На завершающем этапе сегментации изображения, рисунок 33, с помощью битовой маски выделяется только область изображения, содержащая лейкоцит. Она масштабируется до нужного размера и передаётся на вход классификатору.

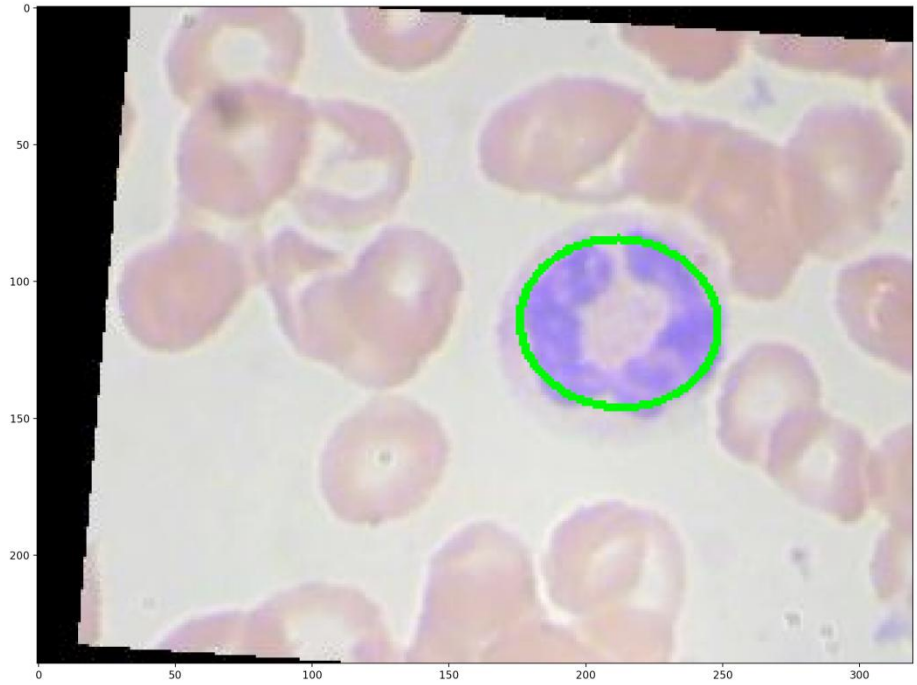

Рисунок 33 – Финальная выделенная область интереса, подаваемая на вход нейронной сети

Эта часть программы позволяет пользователю убедиться, что модель корректно сегментирует лейкоциты на новых изображениях.

Таким образом, реализованный процесс сегментации позволяет эффективно выделять лейкоциты на изображении, что является важным предварительным этапом перед их дальнейшей классификацией. Использование обученной модели и автоматизированной валидации обеспечивает высокую точность выделения клеток, минимизируя влияние фоновых объектов. Этот этап подготовки данных критически важен, так как качество сегментации напрямую влия-

ет на точность последующей классификации, которая будет рассмотрена в следующем разделе.

3.2.2 Методика тестирования системы

Тестирование разработанной системы классификации лейкоцитов проводилось с целью оценки точности модели на независимых изображениях, не использовавшихся на этапе обучения. Результаты тестирования позволяют судить о способности нейронной сети обобщать информацию и корректно классифицировать ранее не встречавшиеся данные.

Для оценки качества модели использовался независимый тестовый датасет, сформированный из исходного набора изображений клеток крови. Изображения тестовой выборки не пересекались с тренировочной, что обеспечивает честную проверку модели и исключает переобучение.

Первым этапом работы программы является запуск обучения. Для этого запускается скрипт learning.py, который выполняет полный цикл работы с данными и моделью. Этот этап является центральной частью программы, где происходит обучение нейронной сети. Устанавливаются количество эпох и размер батча. Эти параметры можно изменять в коде скрипта learning.py. Количество эпох определяет, сколько раз модель будет проходить по всем данным.

После каждой эпохи выводится информация о текущих результатах, рисунок 34: Точность (accuracy) на обучающих и валидационных данных, и значение функции потерь (loss) также на обучающих и валидационных данных. Эта информация помогает контролировать, когда модель начинает переобучаться (перестает улучшаться на валидационных данных).

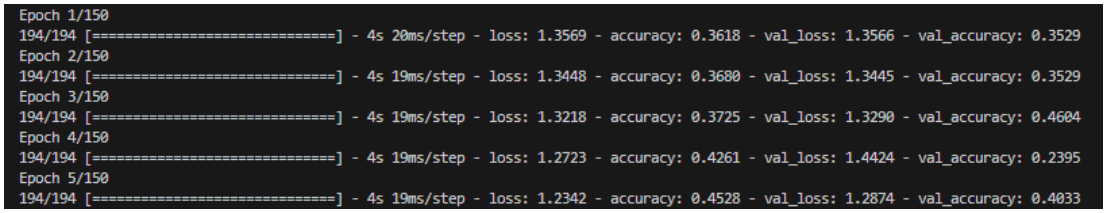

Рисунок 34 – Информация о процессе обучения

Для количественной оценки эффективности модели применялись следующие стандартные метрики машинного обучения:

– Precision (точность): доля верно предсказанных объектов данного класса ко всем объектам, которым модель присвоила этот класс;

– Recall (полнота): доля верно предсказанных объектов данного класса ко всем объектам этого класса в выборке;

– F1-score: гармоническое среднее между точностью и полнотой, отражающее сбалансированность модели;

– Support (поддержка): общее количество объектов в выборке, относящихся к данному классу.

Финальные результаты тестирования классификатора представлены на рисунке 35:

	precision	recall	f1-score	support
EOSINOPHIL	0.84	0.96	0.89	543
LYMPHOCYTE	0.97	0.91	0.94	507
MONOCYTE	0.91	0.98	0.95	617
NEUTROPHIL	0.97	0.87	0.92	909
accuracy			0.92	2576
macro avg	0.92	0.93	0.92	2576
weighted avg	0.93	0.92	0.92	2576

Рисунок 35 – Результаты тестирования модели

Общая точность модели составила 93 %, что свидетельствует о высоком качестве работы алгоритма на независимом датасете. Наилучшую точность и полноту модель показала в отношении класса моноцит (F1-score = 0.95), что может быть связано с наибольшим количеством обучающих примеров данного класса. В целом показатели могут варьироваться в зависимости от количества обучающих данных, а также разнообразия типов изображений.

Независимая валидация модели проводилась на изображениях, полностью исключённых из процесса обучения. Это позволило получить объективную оценку способности модели к обобщению. Результаты метрик F1-score и точности показали, что модель в целом хорошо справляется с задачей классификации, особенно на классах с высоким уровнем представленности в тренировочной выборке.

3.2.3 Алгоритм работы программы

В рамках практического тестирования программы были проведены демонстрационные запуски системы классификации лейкоцитов на независимых изображениях, не входивших в тренировочную выборку.

Разработанное программное обеспечение предусматривает два основных сценария практического использования обученной модели классификации лейкоцитов:

– классификация нескольких изображений из выбранной директории;

– классификация одного конкретного изображения.

Оба сценария реализованы в модуле load_test_model.py и активируются в зависимости от выбранного пользователем набора данных. Ниже приведено описание последовательности действий и результатов работы программы в каждом из случаев.

Сценарий 1 – классификация группы изображений, рисунок 36. В данном сценарии пользователь указывает путь к директории с изображениями тестового набора. Программа автоматически выбирает из неё случайную выборку изображений в количестве, заданном параметром (по умолчанию – 10). Эта выборка обрабатывается с помощью предварительных модулей и подаётся на вход предварительно обученной модели, указанной пользователем:

– загрузка изображений и случайная выборка выполняются с помощью функции sampling_images() из модуля load_test_model.py;

– предобработка изображений (нормализация, приведение к нужному размеру и форматированию) осуществляется с использованием классов PreProcessor и ImageToArrayPreprocessor;

– результаты классификации выводятся на экран: к каждому изображению добавляется подпись с названием предсказанного класса, рисунок 35;

– одновременно с этим все результаты предсказаний отображаются в текстовом виде в консоли.

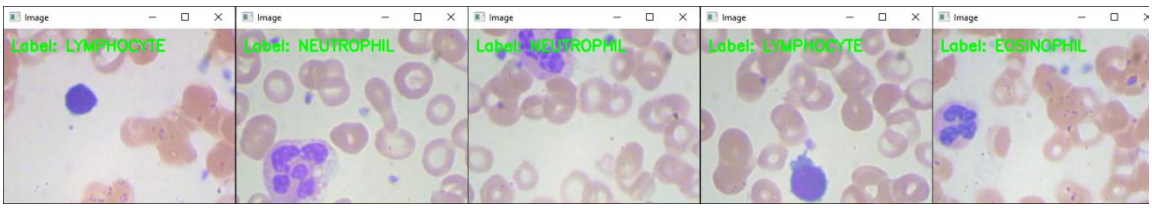

Рисунок 36 – Пример интерфейса с несколькими изображениями и наложенными метками классов

Если пользователь выбирает конкретное изображение, программа проводит его предобработку и классификацию по тем же этапам, что и в случае с группой изображений. Результатом работы является отображение изображения с меткой класса, а также информация о предсказании, выведенная в консоль, рисунок 37.

Рисунок 37 – Отображение одного изображения с предсказанной меткой класса

Оба сценария позволяют визуально убедиться в корректности работы модели и пригодны как для пользовательского применения, так и для демонстрационных целей. Возможность выбора между групповой классификацией и одиночным анализом делает программу гибкой и удобной в использовании.

3.2.4 Разработка интерфейса программы

В ходе работы также был создан простой функциональный интерфейс, реализующий основные требования к программе, включающий возможности для выбора обучающих данных, запуска обучения, тестирования модели и визуализации результатов. Основной целью разработки интерфейса было создание удобного инструмента, позволяющего пользователю взаимодействовать с программой без необходимости глубоких технических знаний в области машинного обучения и программирования. Основное меню данного интерфейса представлено на рисунке 38.

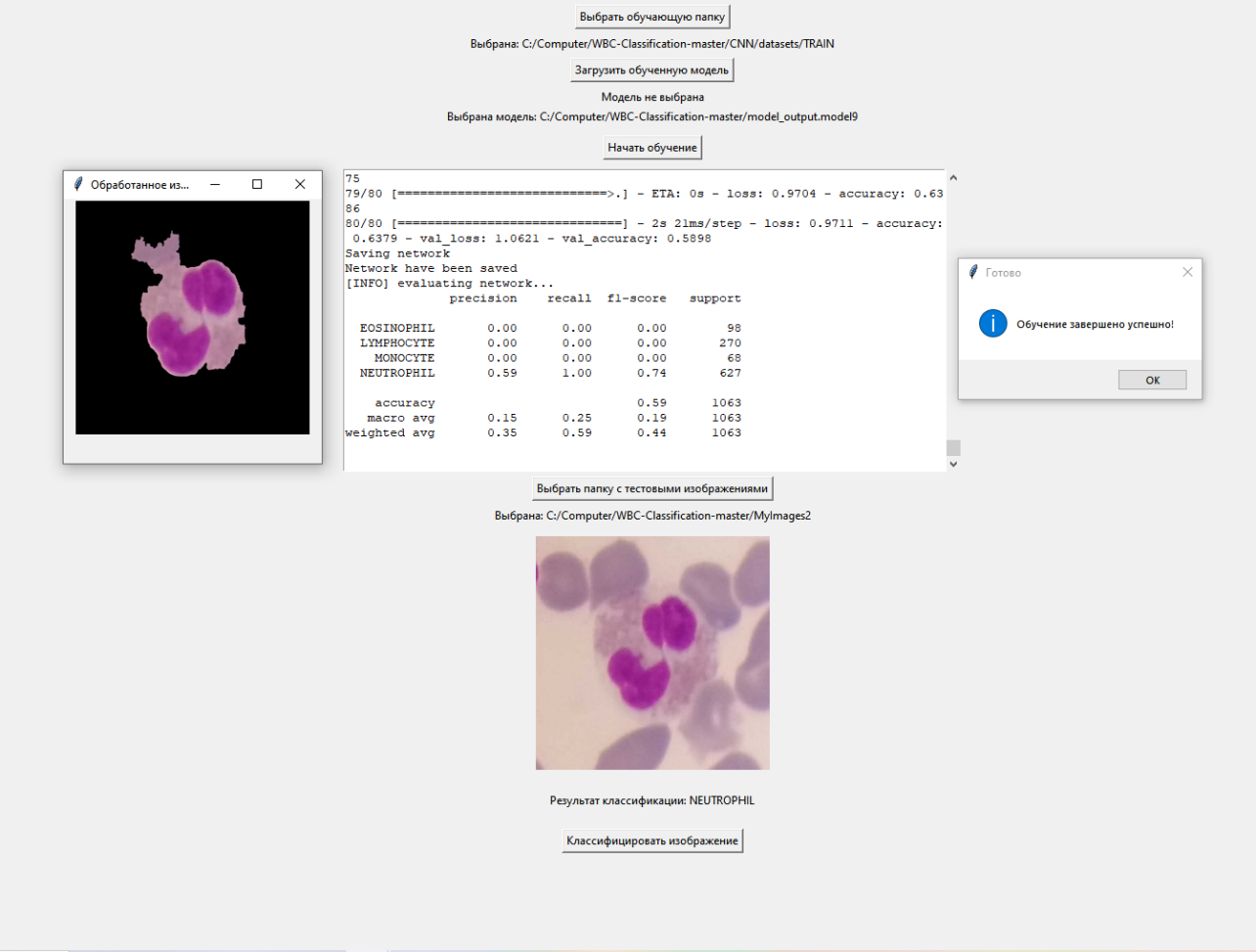

Рисунок 38 – Пример интерфейса программы

Интерфейс был разработан с использованием библиотеки Tkinter, которая является стандартным инструментом для создания графических интерфейсов в Python. Он включает в себя несколько ключевых компонентов, обеспечивающих необходимую функциональность.

Выбор обучающих данных – для обучения нейронной сети необходимо выбрать папку, содержащую изображения. В интерфейсе предусмотрена кнопка "Выбрать обучающую папку", при нажатии на которую открывается диалоговое окно для выбора соответствующей директории с обучающими изображениями. Выбранный путь отображается в метке под кнопкой.

Загрузить обученную модель – для использования заранее обученной модели или продолжения обучения с определённой моделью, предусмотрена кнопка "Загрузить обученную модель". При её нажатии пользователь может выбрать файл с моделью, который будет загружен в программу для дальнейшего использования.

Запуск обучения – после выбора обучающей папки и модели, пользователь может начать процесс обучения нейронной сети, нажав на кнопку "Начать обучение". В этот момент программа начинает обучение с использованием выбранных данных, а процесс обучения и результаты обучения отображается в текстовом окне интерфейса. Все сообщения об ошибках и ходе работы выводятся непосредственно в интерфейсе, что позволяет контролировать выполнение задачи. По завершению обучения формируются графики со статистикой обучения, а также график зависимости точности и функции потерь, рисунок 27.

Выбор тестовых изображений – для тестирования обученной модели интерфейс включает кнопку "Выбрать папку с тестовыми изображениями". После выбора соответствующей папки с изображениями, программа будет готова к классификации загруженных изображений. Это может быть как группа изображений, так и одно конкретное изображение.

Классификация изображений – после выбора тестовых данных, пользователю предоставляется возможность классифицировать изображение с помощью обученной модели. Для этого предусмотрена кнопка "Классифицировать изображение", при нажатии на которую программа выберет первое изображение из выбранной тестовой папки, отобразит его в интерфейсе и произведёт классификацию.

Отображение результатов – после классификации изображения, результат выводится в виде текста в отдельной метке с описанием класса, к которому относится изображение. Помимо этого, в отдельном окне показывается обработанное изображение с наложением маски и другими визуальными метками, что позволяет пользователю наглядно оценить результаты работы программы. В настройках программы также можно выбрать то, какую маску программа будет отображать в результате работы. Пример маски представлен на рисунке 39.

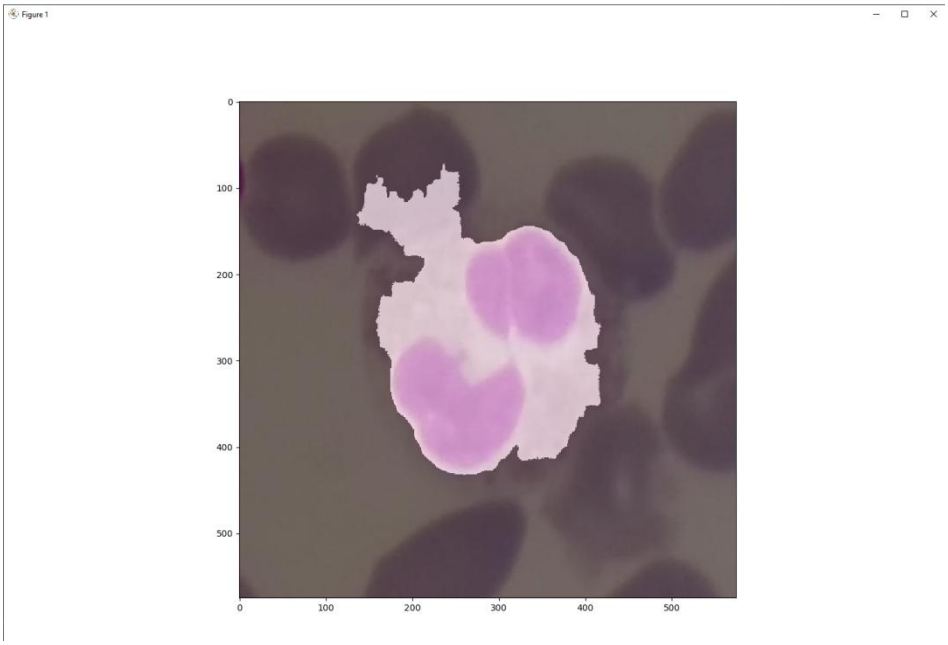

Рисунок 39 – Пример изображения с одной из наложенных масок

Визуализация в процессе обучения – в процессе обучения и классификации также предусмотрены визуализации, такие как графики, отображающие точность модели по мере её тренировки. Это позволяет пользователю отслеживать прогресс и корректировать параметры обучения.

Интерфейс был разработан таким образом, чтобы пользователь мог работать с программой в удобной и интуитивно понятной среде, при этом обеспечивая максимальную функциональность для обработки изображений, обучения и тестирования модели.

ЗАКЛЮЧЕНИЕ

В рамках выполнения данной магистерской диссертации была разработана и исследована нейронная сеть для автоматизированной классификации лейкоцитов на основе методов компьютерного зрения. Проведенное исследование подтвердило эффективность применения современных нейросетевых подходов для решения задач медицинской визуализации и анализа изображений крови.

В рамках работы был выполнен обзор актуальных методов машинного обучения и архитектур сверточных нейронных сетей, что позволило обосновать выбор модифицированной модели типа Sequential. На основе данной архитектуры была разработана и обучена нейронная сеть, способная автоматически классифицировать изображения лейкоцитов по их классам.

Особое внимание было уделено предварительной обработке изображений, включая сегментацию и выделение ключевых областей, что повысило точность модели. Разработанная система была интегрирована в удобный графический интерфейс, обеспечивающий визуализацию результатов классификации и удобство взаимодействия для конечного пользователя.

Точность классификации оказалась высокой: после обучения модель показала точность 94 % на тестовых данных. Это демонстрируют высокий потенциал внедрения подобных технологий в практику медицинской диагностики. Разработанное программное обеспечение может быть использовано в качестве вспомогательного инструмента для специалистов, а также служить основой для дальнейших научных исследований и прикладных разработок в области цифровой медицины.

Повышение эффективности модели возможно за счёт внедрения более продвинутых архитектур нейронных сетей, а также применения методов регуляризации, направленных на снижение риска переобучения. Дополнительным фактором улучшения служит использование более разнообразных и объемных наборов данных, включая информацию из внешних источников, что способствует повышению способности модели к обобщению.

Дальнейшее развитие системы может включать адаптацию алгоритма для решения других медицинских задач, таких как идентификация различных видов патологий или анализ изображений иных типов биологических клеток.

Проведённое исследование подтвердило целесообразность применения методов искусственного интеллекта, в частности сверточных нейронных сетей, для автоматизированного анализа клеток крови. Внедрение подобных технологий имеет потенциал значительно ускорить и повысить точность диагностических процедур, тем самым способствуя развитию современных средств медицинской диагностики.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Батищев, Д. С. Метод сегментации перекрывающихся форменных элементов крови на микроскопических медицинских изображениях. / Д. С. Батищев, В. М. Михелев. // Экономика. Информатика. – 2020. – № 47(4). – С. 803-815.

2 Бурков, А. Машинное обучение без лишних слов / А. Бурков. – СПб, 2020. – 192 с.

3 Введение в сверточные нейронные сети. / [Электронный ресурс]. – Режим доступа : https://habr.com/ru/articles/454986. – 12.11.2024.

4 Волкова, С. А. Основы клинической гематологии: учебное пособие // С. А. Волкова, Н. Н. Боровков. – Н. Новгород: Издательство Нижегородской гос. медицинской академии, 2013. – 400 с.

5 Вьюгин, В. Математические основы машинного обучения и прогнозирования // Владимир Вьюгин. – МЦНМО, 2014. – 304 с.

6 Выбор слоя активации в нейронных сетях: как правильно выбрать для вашей задачи. / [Электронный ресурс]. – Режим доступа : https://habr.com/ru/articles/727506. – 01.11.2024.

7 Гелиг, А. Введение в математическую теорию обучаемых распознающих систем и нейронных сетей. Учебное пособие // А. Гелиг, А. Матвеев. – Издательство СПбГУ, 2014. – 224 с.

8 Гудфеллоу, Я. Глубокое обучение / Я. Гудфеллоу. – М. : ДМК Пресс, 2018. – 652 с.

9 Джереми, У. Машинное обучение: основы, алгоритмы и практика применения / У. Джереми. – BHV-СПб, 2022. – 640 с.

10 Документация по языку программирования Python / [Электронный ресурс]. – Режим доступа : https://www.python.org/doc. – 14.04.2025.

11 Захаров, А. В. Об одной методике классификации клеток крови и ее программной реализации / [Электронный ресурс]. – Режим доступа :

https://cyberleninka.ru/article/n/ob-odnoy-metodike-klassifikatsii-kletok-krovii-ee-programmnoy-realizatsii. – 19.03.2025.

12 Как работает сверточная нейронная сеть (CNN). / [Электронный ресурс]. – Режим доступа : https://neurohive.io/ru/osnovy/. – 21.11.2024.

13 Классификатор изображений / [Электронный ресурс]. – Режим доступа : https://habr.com/ru/companies/dmlabs/articles/205842. – 25.10.2024.

14 Крис, Э. Машинное обучение с использованием Python. / Э. Крис. – БХВ, 2019. – 384 с.

15 Семантическая сегментация на основе архитектуры U-Net и определение расстояния между объектами / [Электронный ресурс]. – Режим доступа : https://habr.com/ru/articles/746842/. – 19.10.2024.

16 Флах, П. Машинное обучение. / П. Флах. – М. : ДМК Пресс, 2015. – 400 с.

17 Accuracy_score // scikit-learn.org. / [Электронный ресурс]. – Режим доступа : https://scikitlearn.org/stable/modules/generated/. – 21.04.2025.

18 Ashish, G. Classification of White blood cell using Convolution Neural Network / G. Ashish, K. Himani, K. Vijay // Biomedical Signal Processing and Control. – 2021. – Т. 71, № 1. – С. 1-7.

19 Bishop, C. Pattern Recognition and Machine Learning (Information Science and Statistics) / C. M. Bishop. – Springer-Verlag New York Inc, 2007. – 738 с.

20 Convolutional Neural Network definition. / [Электронный ресурс]. – Режим доступа : https://deepai.org/machine-learning-glossary andterms/convolutional-neural-network. – 15.11.2024.

21 Convolutional Neural Networks, Explained. / [Электронный ресурс]. – Режим доступа : https://towardsdatascience.com/convolutional-neural-networks explained9cc5188c4939. – 15.11.2024.

22 DM-CNN: Dynamic Multi-scale Convolutional Neural Network with uncertainty quantification for medical image classification / Han Qi, Qian Xin, Xu Hongxiang // Comput. Biol. Med. – 2024. – № 168. – С. 54-56.

23 Dongyao, Jia Detection of cervical cancer cells based on strong feature CNN-SVM network // Jia Dongyao, Li Zhengyi, Zhang Chuanwang // Neurocomputing: журнал. – 2020. – № 411. – С. 112-127.

24 Fusion of U-Net and CNN model for segmentation and classification of skin lesion from dermoscopy images / A. Vatsala, G. Sheifali, K. Deepika, S. Karamjeet // Expert Systems with Applications. – 2022. – № 213. – С.7-9.

25 F1_score // scikit-learn.org. / [Электронный ресурс]. – Режим доступа : https://scikitlearn.org/stable/modules/generated/sklearn.metrics.f1_score.html. – 21.04.2025.

26 Goodfellow, I. Deep Learning (Adaptive Computation and Machine Learning series) / Ian Goodfellow, Yoshua Bengio, Aaron Courville. – The MIT Press, 2016. – 800 с.

27 Guido, S. Introduction to Machine Learning with Python: A Guide for Data Scientists / S. Guido, A. Mueller. – O`Reilly Media, 2016. – 400 с.

28 Hüseyin, K. White blood cells detection and classification based on regional convolutional neural networks / K. Hüseyin, A. Engin, Ö. Fatih // Medical Hypotheses. – 2020. – № 135. – С. 10-21.

29 Joydip, S. Classification and detection of white blood cells using enhanced YOLOv5 algorithm / S. Joydip, A. Shaik, B. Earu // Laser Science. – 2022. – № 1. – С. 1-12.

30 Kelleher, J. Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies // J. D. Kelleher, B. Mac Namee, A. D`Arcy. – The MIT Press, 2015. July. – 624 с.

31 Khan, A. White blood cell type identification using multi-layer convolutional features with an extreme-learning machine // A. Khan, A. Eker, A. Chefranov, H. Demirel // Biomedical Signal Processing and Control. – 2021. – № 69. – С. 1-11.

32 Lamberti, W. Blood cell classification using interpretable shape features: A Comparative study of SVM models and CNN Based approaches / W. F. Lamberti // Computer Methods and Programs in Biomedicine Update. – 2021. – № 1. – С. 1-8.

33 Laura, F. A CNN-based image detector for plant leaf diseases classification [Электронный ресурс] / Falaschetti Laura, Manoni Lorenzo, Di Leo Denis // HardwareX: электронный журнал. – https://www.sciencedirect.com/science/article/pii/S2468067222001080. – 14.11.2024

34 Mamta, A. Exploring deep convolution neural networks with transfer learning for transformation zone type prediction in cervical cancer / A. Mamta, D. Sanjeev, S. Kulvinder // Soft Computing: Theories and Applications. – 2020. – № 1. – С. 1127-1138.

35 Meta-Learning in Machine Learning / [Электронный ресурс]. – Режим доступа : https://www.geeksforgeeks.org/meta-learning-in-machine-learning. – 16.02.2025.

36 Mitchell, T. Machine Learning / T. Mitchell. – Mc Graw Hill India, – 2017. – 432 с.

37 Neha, S. An Analysis Of Convolutional Neural Networks For Image Classification [Электронный ресурс] / S. Neha, J. Vibhor, M. Anju // Procedia Computer Science: электронный журнал. – https://www.sciencedirect.com/science/article /S1877050918309335. – 04.03.2025.

38 Partha, P. An Automatic Nucleus Segmentation and CNN Model based Classification Method of White Blood Cell / P. B. Partha, S. Rappy, K. Ki-Doo // Expert Systems With Applications: журнал. – 2020. – № 149. – С. 1-14.

39 Pooling In Convolutional Neural Networks. / [Электронный ресурс]. – Режим доступа : https://blog.paperspace.com/pooling-inconvolutional-neural-networks/. – 03.04.2025.

40 Pros and Cons of Unsupervised Learning / [Электронный ресурс]. – Режим доступа : https://pythonistaplanet.com/pros-and-cons-of-unsupervised-learning/. – 12.03.2025.

41 Python. / [Электронный ресурс]. – Режим доступа : https://habr.com/ru/hub/python/. – 06.03.2025.

42 Raschka, S. Python Machine Learning / S. Raschka. – Packt Publishing. – 2015. – 454 с.

43 Rashid, T. Make Your Own Neural Network / T. Rashid. – Create Space Independent Publishing Platform. 2016. – С. 222.

44 Recall_score // scikit-learn.org. / [Электронный ресурс]. – Режим доступа : https://scikitlearn.org/stable/modules/generated/sklearn.metrics.recall_score.html#recallscore. – 16.04.2025.

45 Rojas, R. Neural Networks: A Systematic Introduction / R. Rojas. – Varga-Springer Berlin Heidelberg, – 1996. – 522 с.

46 Russell, S. Artificial Intelligence: Pearson New International Edition: A Modern Approach / S. Russel, N. Peter. – Pearson, – 2013. – 1104 с.

47 Shalev-Shwarthz, S. Understanding Machine Learning: From Theory to Algorithms / S. Shalev-Shwartz, S. Ben-David. – Cambridge University Press. – 2014. – 415 с.

48 Shukla, N. Machine Learning with TensorFlow / N. Shukla. – Manning Publication. – 2018. – 272 с.

49 WBC-Net: A white blood cell segmentation network based on U-Net++ and ResNet [Электронный ресурс] / Yan Lu, Xuejun Qin, Haoyi Fan // Applied Soft Computing: электронный журнал. – https://www.sciencedirect.com/science/article/pii/S1568494620309455. – 16.10.2025.

50 What is Keras? / [Электронный ресурс]. – Режим доступа: https://www.geeksforgeeks.org/what-is-keras / Applications of Keras, Understanding Keras, learning models with minimal code. – 25.02.2025.