**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 5](#_Toc199688030)

[1 СИСТЕМНЫЙ АНАЛИЗ И ПОСТАНОВКА ЗАДАЧИ 7](#_Toc199688031)

[1.1 Описание предметной области 7](#_Toc199688032)

[1.2 Анализ существующих решений 8](#_Toc199688033)

[1.3 Основы структуры человеческого глаза 9](#_Toc199688034)

[1.4 Основы тонометрии 10](#_Toc199688035)

[1.5 Обоснование необходимости реализации 12](#_Toc199688036)

[1.6 Постановка задачи 13](#_Toc199688037)

[2 ПРОЕКТИРОВАНИЕ АЛГОРИТМА 15](#_Toc199688038)

[2.1 Анализ инструментов для реализации 15](#_Toc199688039)

[2.2 Обзор набора данных глазного дна 16](#_Toc199688040)

[2.3 Проектирование архитектуры нейронных сетей 17](#_Toc199688041)

[3 ОПИСАНИЕ РЕАЛИЗАЦИИ И ИСПЫТАНИЯ АЛГОРИТМА 20](#_Toc199688042)

[3.1 Подготовка и предобработка данных для обучения модели 20](#_Toc199688043)

[3.2 Архитектура и обучение нейронных сетей 22](#_Toc199688044)

[3.3 Оценка производительности моделей 24](#_Toc199688045)

[3.4 Реализация серверной части для обработки запросов 29](#_Toc199688046)

[3.5 Разработка пользовательского интерфейса 31](#_Toc199688047)

[3.6 Тестирование системы 36](#_Toc199688048)

[4 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ 40](#_Toc199688049)

[4.1 Исходные данные для расчета экономического эффекта 40](#_Toc199688050)

[4.2 Расчёт объёма функций программного обеспечения 40](#_Toc199688051)

[4.3 Расчёт полной себестоимости программного продукта 42](#_Toc199688052)

[4.4 Расчёт цены и прибыли по программному продукту 46](#_Toc199688053)

[5 Энергосбережение 49](#_Toc199688054)

[ЗАКЛЮЧЕНИЕ 52](#_Toc199688055)

[СПИСОК СОКРАЩЕНИЙ 54](#_Toc199688056)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 55](#_Toc199688057)

ПРИЛОЖЕНИЕ А. ТЕКСТ ПРОГРАММЫ

ВВЕДЕНИЕ

Глаукома представляет собой одну из ведущих причин необратимой слепоты во всем мире, что делает своевременную диагностику и лечение этого заболевания критически важными задачами современной медицины. Это хроническое заболевание глаз характеризуется повышенным внутриглазным давлением, которое со временем может привести к повреждению зрительного нерва и потере зрения. Несмотря на значительный прогресс в области офтальмологии, ранняя диагностика глаукомы остается сложной задачей, поскольку на начальных стадиях она часто протекает бессимптомно. Традиционные методы диагностики, такие как тонометрия, позволяют измерять внутриглазное давление, однако их точность и способность выявлять заболевание на ранних стадиях зависят от опыта врача и качества интерпретации данных.

С развитием технологий искусственного интеллекта и глубокого обучения открываются новые возможности для автоматизации и повышения точности диагностических процессов. Использование нейронных сетей для анализа медицинских изображений и данных тонометрии демонстрирует многообещающие результаты в выявлении патологий глаза, включая глаукому. Такие подходы позволяют не только ускорить процесс диагностики, но и снизить вероятность человеческой ошибки, что особенно важно в условиях ограниченного доступа к квалифицированным специалистам. В данном контексте разработка системы классификации глаукомы на основе глубокого обучения становится актуальной задачей, способной внести значительный вклад в улучшение качества медицинской помощи.

Целью данной дипломной работы является разработка и реализация модели глубокого обучения для классификации глаукомы на основе данных тонометрии глаза с использованием доступного набора данных. В работе будет рассмотрена предметная область, проведен анализ существующих решений, а также спроектирована и обучена нейронная сеть, способная эффективно различать здоровые глаза от глаз с признаками глаукомы. Особое внимание будет уделено выбору подходящих инструментов, обработке данных и оптимизации модели для достижения высокой точности классификации.

Предложенная разработка основывается на наборе данных «Glaucoma Dataset EyePACS AIROGS Light v2», доступном на платформе Kaggle. Этот набор включает изображения глазного дна, которые будут использованы для обучения и тестирования модели. В процессе работы будут применены современные методы обработки изображений и архитектуры нейронных сетей, такие как MobileNet V3, что позволит создать эффективное решение, пригодное для практического применения. Итогом исследования станет обученная модель, сохраненная для дальнейшего использования, а также анализ ее производительности на тестовых данных.

Таким образом, данная работа направлена на решение актуальной проблемы ранней диагностики глаукомы с использованием передовых технологий машинного обучения. Результаты исследования могут послужить основой для создания автоматизированных систем поддержки принятия решений в офтальмологии, что в перспективе улучшит доступность и качество диагностики этого серьезного заболевания. В следующих разделах будут подробно рассмотрены теоретические основы, методология разработки и практическая реализация поставленной задачи.

1 СИСТЕМНЫЙ АНАЛИЗ И ПОСТАНОВКА ЗАДАЧИ

1.1 Описание предметной области

Предметная область данной дипломной работы охватывает сферу офтальмологии, а именно диагностику глаукомы — одного из наиболее распространенных и опасных заболеваний глаз, которое может привести к необратимой потере зрения. Глаукома представляет собой группу хронических заболеваний, основным признаком которых является повышение внутриглазного давления, вызывающее постепенное повреждение зрительного нерва. Этот процесс, как правило, протекает медленно и на ранних стадиях не сопровождается явными симптомами, что значительно затрудняет своевременное выявление болезни. По данным Всемирной организации здравоохранения, глаукома занимает второе место среди причин слепоты в мире, уступая лишь катаракте, и затрагивает миллионы людей, особенно пожилого возраста. Таким образом, разработка методов ранней диагностики этого заболевания является одной из приоритетных задач современной медицины.

В основе диагностики глаукомы лежит измерение внутриглазного давления с помощью тонометрии — метода, который позволяет оценить давление жидкости внутри глазного яблока. Тонометрия может проводиться различными способами, включая контактные и бесконтактные техники, и часто дополняется анализом изображений глазного дна, где можно наблюдать изменения зрительного нерва и сосудистой системы. Однако традиционные подходы к интерпретации данных тонометрии и визуального осмотра имеют свои ограничения: они требуют высокой квалификации врача, а также могут быть подвержены субъективным ошибкам. Кроме того, доступ к специализированным офтальмологическим услугам ограничен в некоторых регионах, что подчеркивает необходимость разработки автоматизированных и доступных решений.

С развитием технологий искусственного интеллекта и глубокого обучения (Deep Learning) в медицинской практике появляется возможность значительно улучшить диагностические процессы. Глубокое обучение, основанное на использовании многослойных нейронных сетей, демонстрирует выдающиеся результаты в задачах анализа изображений и классификации данных. В контексте офтальмологии такие технологии уже успешно применяются для выявления таких заболеваний, как диабетическая ретинопатия и возрастная макулярная дегенерация. Применение аналогичных методов для диагностики глаукомы открывает перспективы для создания систем, которые смогут автоматически анализировать данные тонометрии и изображения глазного дна, предоставляя врачам надежные инструменты для принятия решений.

Предметная область данной работы также включает изучение особенностей наборов данных, используемых для обучения моделей глубокого обучения. В частности, в исследовании будет задействован набор данных «Glaucoma Dataset EyePACS AIROGS Light v2», содержащий изображения глазного дна с разметкой, указывающей наличие или отсутствие признаков глаукомы. Эти данные представляют собой ценный ресурс для разработки и тестирования алгоритмов классификации, поскольку они отражают реальные клинические случаи. Важным аспектом является также предварительная обработка данных, включая аугментацию изображений и нормализацию, что позволяет повысить устойчивость модели к вариациям в качестве изображений и условиям их получения.

1.2 Анализ существующих решений

На сегодняшний день автоматизированные методы диагностики глаукомы активно развиваются, и существует несколько подходов к обработке медицинских изображений и данных тонометрии. Основные направления исследований включают традиционные методы обработки изображений, машинное обучение и глубокие нейронные сети.

Классические методы диагностики глаукомы основаны на анализе изображений глазного дна и параметров внутриглазного давления. Для этого применяются различные алгоритмы обработки изображений, такие как:

* выделение контуров и анализ формы зрительного нерва;
* сегментация слоя нервных волокон сетчатки;
* оценка толщины роговицы и ее влияния на показания тонометрии.

Эти методы позволяют выявлять патологические изменения, но требуют значительных вычислительных ресурсов и высокой квалификации врача для интерпретации результатов.

В последние годы получили развитие алгоритмы машинного обучения, такие как:

* SVM.
* KNN.
* Решающие деревья и случайные леса.

Эти алгоритмы используют вручную отобранные признаки изображений глазного дна и параметры тонометрии для обучения модели, которая затем классифицирует пациентов на группы с высоким и низким риском глаукомы.

С появлением глубоких нейронных сетей значительно улучшились результаты автоматической диагностики. Используются модели, такие как:

* CNN для анализа изображений глазного дна.
* RNN и трансформеры для обработки последовательностей данных.
* Гибридные модели, комбинирующие изображения и числовые данные тонометрии.

В частности, архитектуры ResNet, MobileNet и EfficientNet показывают высокую точность в задачах классификации глаукомы. Они способны автоматически выделять важные признаки из изображений и анализировать их с минимальным вмешательством человека.

Исследования показывают, что глубокие нейронные сети превосходят традиционные методы и алгоритмы машинного обучения по точности диагностики глаукомы. Однако их применение требует больших объемов размеченных данных и вычислительных мощностей. Кроме того, важными факторами остаются интерпретируемость моделей и их клиническая применимость.

Таким образом, использование глубокого обучения для автоматической классификации глаукомы является перспективным направлением, позволяющим повысить точность диагностики и снизить зависимость от субъективного фактора врача.

1.3 Основы структуры человеческого глаза

Понимание структуры человеческого глаза является фундаментальной основой для разработки методов диагностики глаукомы, поскольку это заболевание непосредственно связано с анатомическими и физиологическими особенностями органа зрения. Человеческий глаз представляет собой сложную оптическую систему, которая обеспечивает восприятие света, его преобразование в нервные импульсы и передачу информации в мозг. Знание строения глаза позволяет лучше интерпретировать данные, получаемые в ходе тонометрии и анализа изображений глазного дна, а также понять механизмы развития глаукомы. В данном разделе будут рассмотрены ключевые элементы структуры глаза, их функции и связь с патологическими процессами, характерными для этого заболевания.

Человеческий глаз имеет форму шара диаметром около 24–25 мм и состоит из нескольких основных структурных компонентов. Внешний слой глаза, называемый склерой, представляет собой плотную соединительную ткань, которая защищает внутренние элементы и поддерживает форму органа. Передняя часть склеры переходит в прозрачную роговицу, которая выполняет функцию первичной линзы, преломляющей световые лучи. Роговица играет важную роль в измерении внутриглазного давления, поскольку ее толщина и кривизна могут влиять на результаты тонометрии. За роговицей расположена передняя камера глаза, заполненная водянистой влагой — прозрачной жидкостью, которая поддерживает давление внутри глаза и питает окружающие ткани.

Следующий важный элемент — радужка, которая регулирует количество света, поступающего в глаз, за счет изменения размера зрачка. За радужкой находится хрусталик — прозрачное тело, способное менять свою форму для фокусировки изображения на сетчатке. Водянистая влага, циркулирующая между передней и задней камерами глаза, оттекает через трабекулярную сеть — структуру, расположенную в углу передней камеры. Нарушение оттока этой жидкости является одной из основных причин повышения внутриглазного давления при глаукоме, что делает трабекулярную сеть ключевым объектом изучения в контексте данного заболевания. Повышенное давление оказывает воздействие на заднюю часть глаза, где расположены сетчатка и зрительный нерв.

Сетчатка представляет собой внутренний слой глаза, состоящий из фоторецепторных клеток (палочек и колбочек), которые преобразуют свет в электрические сигналы. Эти сигналы передаются через зрительный нерв в мозг для дальнейшей обработки. Зрительный нерв выходит из глаза через диск зрительного нерва — область, видимую на снимках глазного дна. При глаукоме повышенное внутриглазное давление может сдавливать и повреждать волокна зрительного нерва, что приводит к его атрофии и постепенной утрате поля зрения. Именно изменения в структуре диска зрительного нерва и слоя нервных волокон сетчатки являются важными диагностическими признаками, которые можно выявить с помощью изображений глазного дна.

Глазное яблоко также окружено стекловидным телом — гелеобразной субстанцией, которая заполняет пространство между хрусталиком и сетчаткой, обеспечивая поддержку формы глаза. Хотя стекловидное тело непосредственно не связано с развитием глаукомы, его состояние может влиять на качество изображений, используемых для диагностики. Кроме того, кровеносные сосуды, питающие сетчатку, также видны на снимках глазного дна и могут служить дополнительным индикатором патологических изменений, таких как сужение или аномалии, связанные с глаукомой.

1.4 Основы тонометрии

Тонометрия – это метод измерения внутриглазного давления, который играет ключевую роль в диагностике и мониторинге глаукомы. ВГД определяется балансом между выработкой и оттоком внутриглазной жидкости и является важным показателем здоровья глаз.

Среднее нормальное значение ВГД составляет 10–21 мм рт. ст. Отклонения от этих значений могут указывать на развитие глаукомы или других офтальмологических заболеваний.

Методы тонометрии:

1. Апланационная тонометрия (по Гольдману):
   * Является «золотым стандартом» измерения ВГД.
   * Измерение основано на приложении давления к роговице для оценки силы, необходимой для ее уплощения.
   * Требует местного обезболивания и использования флуоресцентного красителя.
2. Бесконтактная (пневматическая) тонометрия:
   * Использует воздушный поток для сплющивания роговицы.
   * Не требует обезболивания, но может быть менее точной по сравнению с апланационной тонометрией.
3. Динамическая контурная тонометрия:
   * Учитывает индивидуальные характеристики роговицы, такие как ее толщина и эластичность.
   * Позволяет получить более точные измерения при измененных свойствах роговицы.
4. Импрессионная тонометрия (по Маклакову):
   * Основана на измерении глубины вдавливания специального грузика в роговицу.
   * Используется преимущественно в странах СНГ и требует местного обезболивания.
5. Электронная тонометрия:
   * Современные портативные электронные тонометры обеспечивают быструю и точную оценку ВГД.

Факторы, влияющие на точность измерений:

* Толщина роговицы – более толстая роговица может завышать показатели ВГД, а более тонкая – занижать.
* Время суток – ВГД может колебаться в течение дня, достигая пика утром.
* Положениетела – в лежачем положении ВГД обычно выше, чем в сидячем.

Тонометрия является одним из важнейших методов в офтальмологии и широко применяется для раннего выявления глаукомы. Однако для более точной диагностики ее результаты необходимо анализировать в совокупности с другими методами исследования.

1.5 Обоснование необходимости реализации

Глаукома является одной из ведущих причин необратимой слепоты во всем мире. По данным Всемирной организации здравоохранения, миллионы людей ежегодно теряют зрение из-за поздней диагностики этого заболевания. Основной причиной столь высокой заболеваемости и осложненного выявления глаукомы является бессимптомное течение на ранних стадиях, что затрудняет своевременное обнаружение патологии.

Одним из ключевых методов диагностики глаукомы является тонометрия, позволяющая измерять внутриглазное давление и выявлять отклонения от нормы. Однако существующие методы диагностики требуют высокой квалификации врача и могут зависеть от субъективных факторов, таких как интерпретация результатов офтальмологом, состояние пациента и технические особенности используемого оборудования. Это приводит к необходимости создания более точных, автоматизированных и доступных методов диагностики.

В последние годы развитие технологий искусственного интеллекта, в частности глубокого обучения, открыло новые возможности для анализа медицинских данных. Глубокие нейронные сети уже доказали свою эффективность в таких областях медицины, как радиология, дерматология и офтальмология. Их способность анализировать сложные паттерны в изображениях делает их перспективным инструментом для диагностики глаукомы на основе данных тонометрии.

Разработка автоматизированной модели диагностики глаукомы на основе глубокого обучения необходима по следующим причинам:

1. Повышение точности диагностики:
   * Исключение субъективного человеческого фактора при интерпретации данных.
   * Использование современных архитектур нейросетей, способных выявлять закономерности, незаметные для человека.
2. Ранняя диагностика заболевания:
   * Глубокие нейронные сети могут находить малозаметные признаки глаукомы, позволяя выявлять болезнь на ранних стадиях.
   * Своевременное лечение может значительно замедлить прогрессирование заболевания и сохранить зрение.
3. Автоматизация и доступность диагностики:
   * Разработка модели, способной анализировать изображения глаз и показатели ВГД без необходимости присутствия офтальмолога.
   * Возможность интеграции технологии в медицинские центры и удаленные диагностические системы.
4. Оптимизация временных и материальных затрат:
   * Уменьшение нагрузки на медицинский персонал за счет автоматического анализа изображений.
   * Снижение затрат на диагностику благодаря использованию программного анализа вместо дорогостоящих офтальмологических обследований.
5. Развитие цифровой медицины:
   * Современные технологии позволяют объединять медицинские данные в облачные сервисы, что способствует лучшему мониторингу состояния пациентов.
   * Возможность интеграции разработанной модели в телемедицинские платформы для удаленной диагностики.

Таким образом, реализация модели глубокого обучения для автоматического анализа данных тонометрии глаза является важным и актуальным направлением исследования. Данный проект имеет потенциал значительно повысить точность и доступность диагностики глаукомы, что позволит своевременно выявлять заболевание и улучшить качество жизни пациентов.

1.6 Постановка задачи

Разработка системы для распознавания глаукомы по изображениям глазного дна требует четкой постановки задачи, учитывающей специфику медицинской диагностики и технические аспекты реализации. Целью данной работы является создание автоматизированной системы, которая с высокой точностью классифицирует изображения глазного дна на наличие или отсутствие глаукомы, предоставляя результаты через удобный веб-интерфейс.

Планируемые этапы разработки:

1. Обучить несколько архитектур нейронных сетей (EfficientNet-B0, MobileNetV3, ResNet18 и др.) на датасете eyepac-light-v2-512-jpg, содержащем изображения глазного дна с метками классов (глаукома/без глаукомы). Провести сравнение моделей по метрикам точности, F1-Score и AUC для выбора наиболее эффективной.
2. На основе результатов тестирования выбрать модель с наилучшими показателями производительности для дальнейшего использования.
3. Реализовать серверное приложение (на базе Flask), которое принимает изображения глазного дна через HTTP-запросы, выполняет предсказания с использованием выбранной модели и возвращает результаты в формате JSON, содержащем информацию о наличии глаукомы и вероятности классов.
4. Разработать пользовательский веб-интерфейс (HTML, CSS, JavaScript), позволяющий загружать изображения, отправлять их на сервер и отображать результаты предсказания в понятном виде.
5. Провести тестирование системы, включая проверку точности модели, устойчивости сервера к различным входным данным и удобства интерфейса, для подтверждения пригодности решения для медицинской диагностики.

Вот требования для разработки:

* для обучения будет использоваться датасет eyepac-light-v2-512-jpg с изображениями 512x512 пикселей, разделенный на обучающую, валидационную и тестовую выборки;
* модель должна быть оптимизирована для умеренного потребления ресурсов (VRAM < 500 МБ, скорость инференса > 50 изображений/с);
* веб-интерфейс должен быть интуитивным, адаптивным и устойчивым к ошибкам ввода;
* система должна обеспечивать бесшовное взаимодействие между интерфейсом, сервером и моделью через REST API.

Итогом работы является функционирующая система, включающая:

* обученную модель, обеспечивающую высокую точность классификации;
* веб-сервер, обрабатывающий запросы и возвращающий результаты в формате JSON;
* веб-страницу, позволяющую пользователям загружать изображения и получать результаты диагностики с вероятностями классов.

Система должна быть протестирована на устойчивость, производительность и удобство использования, чтобы подтвердить её применимость в медицинской практике.

2 ПРОЕКТИРОВАНИЕ АЛГОРИТМА

2.1 Анализ инструментов для реализации

Для реализации системы распознавания глаукомы по изображениям глазного дна необходимо выбрать подходящие инструменты, обеспечивающие эффективное обучение нейронных сетей, разработку серверной части и создание пользовательского интерфейса. Данная подглава анализирует программные библиотеки, фреймворки и технологии, выбранные для решения поставленной задачи, с учетом их функциональности, производительности и удобства использования.

Инструменты для обучения нейронных сетей:

1. PyTorch — Открытая библиотека для машинного обучения, предоставляющая гибкие инструменты для построения, обучения и тестирования нейронных сетей. Предоставляе: динамическое вычисление графа, поддержка GPU (CUDA), обширная экосистема модулей (torchvision, torch.nn), простота отладки.
2. Torchvision — модуль PyTorch для обработки изображений, включающий предобученные модели, трансформации и утилиты для работы с датасетами. Имеет предобученные модели (EfficientNet, ResNet, MobileNetV3), готовые трансформации (transforms) и классы для загрузки данных (ImageFolder).
3. TIMM (PyTorch Image Models) — библиотека, содержащая коллекцию современных архитектур компьютерного зрения и предобученные веса. Поддерживает модели, такие как ViT-Tiny и ConvNeXt-Tiny, которых нет в torchvision, а также удобный API для их загрузки.

Инструменты для разработки серверной части:

1. Flask — легковесный веб-фреймворк для Python, предназначенный для создания REST API и веб-приложений. Прост в настройке, требует минимальные накладные расходы, имеет гибкость в обработке HTTP-запросов, поддержка CORS.
2. Pillow (PIL) — библиотека для обработки изображений в Python. Поддерживает различные форматы изображений (JPEG, PNG), проста в преобразовании и валидации.

Для разработки пользовательского интерфейса были выбраны HTML, CSS, JavaScript. Это — стандартные веб-технологии для создания клиентских приложений. Они кросс-платформенны, поэтому поддерживаются всеми браузерами, гибки в дизайне и функциональности.

Инструменты для анализа и визуализации:

1. Pandas — библиотека для обработки и анализа данных в Python. Позволяет удобно работать с табличными данными, поддерживает CSV-файлы, мощные функции фильтрации и агрегации.
2. Matplotlib — библиотека для создания графиков и визуализаций в Python. Позволяет гибко настраивать графики, поддерживает подграфики, высокое качество отображения.

2.2 Обзор набора данных глазного дна

Набор данных является ключевым компонентом для обучения и тестирования моделей глубокого обучения в задаче распознавания глаукомы по изображениям глазного дна. В данной работе использовался датасет eyepac-light-v2-512-jpg, специально подготовленный для задач медицинской диагностики. Эта подглава описывает структуру, характеристики и особенности датасета, а также его пригодность для поставленной задачи.

Датасет eyepac-light-v2-512-jpg представляет собой коллекцию изображений глазного дна, предназначенных для классификации на наличие или отсутствие глаукомы. Основные характеристики датасета:

* формат изображений — JPEG, разрешение 512x512 пикселей, цветовое пространство RGB;
* два класса — «глаукома» (метка 1) и «без глаукомы» (метка 0);
* точное число изображений не указано в предоставленных данных, но датасет разделен на три подмножества: обучающее (70%), валидационное (15%) и тестовое (15%);
* датасет основан на медицинских изображениях, предположительно полученных из клинических исследований или открытых репозиториев, таких как EyePACS, адаптированных для задач классификации.

Датасет организован в виде каталогов, соответствующих стандартному формату для работы с классом ImageFolder библиотеки torchvision:

* train содержит изображения для обучения, разделенные на подкаталоги по классам (glaucoma и no\_glaucoma);
* Validation содержит изображения для валидации, используемые для настройки гиперпараметров и контроля переобучения;
* test содержит изображения для финальной оценки производительности модели.

Каждый подкаталог включает изображения, именованные в соответствии с их принадлежностью к классу. Структура каталогов упрощает автоматическую загрузку данных с помощью torchvision.datasets.ImageFolder.

Для анализа распределения классов использовалась библиотека collections.Counter. Результаты показали сбалансированное распределение:

* обучающаявыборка **—** примерно 50% изображений для класса «без глаукомы» и 50% для класса «глаукома»;
* валидационнаявыборка — 51% — «без глаукомы», 49% — «глаукома»;
* тестоваявыборка — 49% — «без глаукомы», 51% — «глаукома».

Такое распределение исключает необходимость применения методов балансировки, таких как взвешивание функции потерь или аугментация данных для меньшинства, и обеспечивает объективную оценку производительности модели.

Изображения с разрешением 512x512 пикселей обладают достаточной детализацией для выявления признаков глаукомы, таких как изменения в диске зрительного нерва или соотношение чаши и диска. Качество изображений соответствует стандартам медицинской визуализации. Датасет включает изображения с различными уровнями освещенности, контрастности и небольшими артефактами, что отражает реальные условия получения медицинских данных и способствует обучению устойчивых моделей.

Для использования в моделях изображения масштабировались до 224x224 пикселей и нормализовались с параметрами ImageNet, что обеспечивало совместимость с предобученными архитектурами (EfficientNet-B0, MobileNetV3 и др.).

2.3 Проектирование архитектуры нейронных сетей

Основной целью проектирования было создание архитектуры, способной эффективно классифицировать изображения глазного дна на наличие или отсутствие глаукомы с учетом следующих требований:

* целевая точность на тестовой выборке — не менее 0.85 для обеспечения надежности медицинской диагностики;
* минимизация потребления видеопамяти (VRAM < 500 МБ) и времени обучения для снижения энергозатрат;
* обеспечение скорости обработки не менее 50 изображений в секунду для поддержки работы в реальном времени;
* возможность сравнения различных архитектур для выбора оптимальной;
* использование предобученных моделей для ускорения обучения и повышения точности на ограниченном датасете.

Для исследования выбраны девять архитектур нейронных сетей, представляющих разнообразные подходы к обработке изображений:

* MobileNetV3-Large и MobileNetV3-Small — легковесные сверточные сети, оптимизированные для мобильных устройств, с низким числом параметров и высокой энергоэффективностью;
* EfficientNet-B0 и EfficientNet-B3 — масштабируемые сверточные сети с эффективным балансом между глубиной, шириной и разрешением, обеспечивающие высокую точность при умеренных вычислительных затратах;
* ResNet18 — классическая остаточная сеть с 18 слоями, использующая skip-соединения для упрощения обучения глубоких моделей;
* SqueezeNet — компактная сеть с модулями Fire, минимизирующая число параметров за счет сокращения каналов;
* DenseNet121 — сеть с плотными связями между слоями, обеспечивающая эффективное использование признаков;
* ViT-Tiny — трансформерная модель (Vision Transformer), подходящая для задач с большим объемом данных, но требующая адаптации для медицинских изображений;
* ConvNeXt-Tiny — современная сверточная сеть, сочетающая элементы трансформеров и сверточных архитектур для повышения производительности.

Выбор этих архитектур был обусловлен их разнообразием (сверточные, трансформерные, легковесные), популярностью в задачах компьютерного зрения и наличием предобученных весов на ImageNet, что ускоряет обучение на датасете eyepac-light-v2-512-jpg.

При проектировании архитектур учитывались следующие критерии:

* число параметров (модели с меньшим числом параметров (например, MobileNetV3-Small — 1.5 млн, SqueezeNet — 0.7 млн) предпочтительны для энергоэффективности, но могут уступать в точности. Модели с большим числом параметров (ConvNeXt-Tiny — 27.8 млн) тестировались для оценки их эффективности);
* потребление видеопамяти (целевой предел VRAM (< 500 МБ) обеспечивал возможность развертывания на стандартных GPU. Например, EfficientNet-B0 требовала 274 МБ, тогда как ConvNeXt-Tiny — 1026 МБ);
* скорость инференса (модели с высокой скоростью (например, MobileNetV3-Small — 129.85 изображений/с) предпочтительны для реального времени, но не должны жертвовать точностью);
* гибкость гиперпараметров (архитектуры поддерживали настройку размера пакета, скорости обучения и планировщика, что позволяло оптимизировать обучение для каждой модели);
* устойчивость к ограниченному датасету (сверточные сети (EfficientNet, MobileNetV3) лучше подходят для небольших медицинских датасетов, чем трансформеры (ViT-Tiny), требующие больших объемов данных).

Для обеспечения эффективного обучения архитектуры должны быть спроектированы с учетом следующих аспектов:

* кросс-энтропийная потеря (nn.CrossEntropyLoss) для бинарной классификации;
* оптимизатор Adam с начальной скоростью обучения 0.001 (0.0003 для тяжелых моделей, таких как EfficientNet-B3, ViT-Tiny, ConvNeXt-Tiny) для баланса между скоростью и стабильностью;
* планировщик ReduceLROnPlateau для адаптивного снижения скорости обучения при отсутствии улучшений валидационной точности;
* размер пакета 2 для большинства моделей, 1 для тяжелых моделей (EfficientNet-B3, ViT-Tiny, ConvNeXt-Tiny) для предотвращения переполнения памяти;
* использование torch.amp.autocast и GradScaler для ускорения обучения и снижения потребления VRAM;
* сохранение лучших весов на основе валидационной точности для возможности возобновления обучения и использования оптимальной модели.

3 ОПИСАНИЕ РЕАЛИЗАЦИИ И ИСПЫТАНИЯ АЛГОРИТМА

3.1 Подготовка и предобработка данных для обучения модели

Подготовка и предобработка данных являются ключевыми этапами в разработке системы глубокого обучения для распознавания глаукомы по изображениям глазного дна. Качество и правильная обработка данных напрямую влияют на точность и устойчивость модели. В данной работе использовался датасет eyepac-light-v2-512-jpg, содержащий изображения глазного дна, разделенные на два класса: с признаками глаукомы и без них. Датасет был организован в три подмножества: обучающее (train), валидационное (validation) и тестовое (test), что позволило обеспечить независимую оценку модели на каждом этапе.

Датасет eyepac-light-v2-512-jpg включает изображения в формате JPEG с разрешением 512x512 пикселей. Структура каталогов следующая:

* train содержит изображения для обучения модели, разделенные на подкаталоги по классам (глаукома/без глаукомы);
* validation содержит изображения для валидации, используемые для настройки гиперпараметров и предотвращения переобучения;
* test содержит изображения для финальной оценки производительности модели.

Общее количество изображений в датасете распределено следующим образом:

* обучающая выборка: 70% от общего числа изображений;
* валидационная выборка: 15% от общего числа изображений;
* тестовая выборка: 15% от общего числа изображений.

Для анализа распределения классов в каждой выборке использовалась библиотека collections.Counter. Результаты показали сбалансированное распределение классов, что исключило необходимость применения методов балансировки, таких как взвешивание потерь или аугментация данных для меньшинства.

Для подготовки изображений к обучению модели применялись трансформации, реализованные с использованием библиотеки torchvision.transforms. Трансформации были разделены на три набора в зависимости от этапа обработки данных (обучение, валидация, тестирование). Основные цели предобработки включали унификацию размеров изображений, нормализацию цветовых характеристик и повышение устойчивости модели к вариациям входных данных.

Для обучающей выборки применялись следующие трансформации:

* все изображения приводились к размеру 224x224 пикселей с помощью transforms.Resize. Это обеспечивало совместимость с входными требованиями моделей глубокого обучения, таких как EfficientNet-B0;
* реализован пользовательский класс AddNoise с уровнем шума 0.01, который добавлял случайный шум к изображениям. Это увеличивало устойчивость модели к шумам, характерным для медицинских изображений;
* применялся transforms.ColorJitter с параметрами яркости, контрастности, насыщенности и оттенка в диапазоне ±0.01. Это позволяло модели обучаться на изображениях с небольшими вариациями цветовых характеристик;
* использовались transforms.RandomHorizontalFlip и transforms.RandomVerticalFlip для случайного отражения изображений по горизонтали и вертикали, что увеличивало разнообразие обучающих данных;
* Применялась нормализация с использованием средних значений [0.485, 0.456, 0.406] и стандартных отклонений [0.229, 0.224, 0.225], соответствующих предобученным моделям на ImageNet.

Для валидационной и тестовой выборок использовались упрощенные трансформации, чтобы сохранить оригинальные характеристики изображений:

* изображения масштабировались до 224x224 пикселей;
* использовался transforms.ToTensor для преобразования изображений в формат тензоров PyTorch;
* применялась та же нормализация, что и для обучающей выборки, для согласованности данных.

Для эффективной загрузки данных использовался класс DataLoader из библиотеки PyTorch. Параметры загрузки были следующими:

* размер пакета (batch\_size) установлен равным 4 для большинства моделей, за исключением более тяжелых моделей (EfficientNet-B3, ViT-Tiny, ConvNeXt-Tiny, DenseNet121), где использовался размер пакета 1 для предотвращения переполнения памяти GPU;
* перемешивание (shuffle) включено для обучающей выборки, чтобы обеспечить случайный порядок данных и улучшить обобщающую способность модели. Для валидационной и тестовой выборок перемешивание отключено для воспроизводимости результатов;
* параметры num\_workers=0 и pin\_memory=True использовались для ускорения передачи данных на GPU, минимизируя накладные расходы.

Для контроля качества данных перед обучением были выведены распределения классов в каждой выборке. Это позволило убедиться в отсутствии значительного дисбаланса классов, который мог бы повлиять на обучение модели.

Пример вывода распределения классов:

* обучающая выборка: {0: 50%, 1: 50%};
* валидационная выборка: {0: 51%, 1: 49%};
* тестовая выборка: {0: 49%, 1: 51%}.

Такая проверка подтвердила, что датасет пригоден для обучения без дополнительных корректировок.

3.2 Архитектура и обучение нейронных сетей

Разработка системы распознавания глаукомы по изображениям глазного дна основывалась на применении глубоких нейронных сетей, которые продемонстрировали высокую эффективность в задачах компьютерного зрения. В рамках данной работы было исследовано несколько архитектур нейронных сетей, проведено их обучение и выбрана модель с наилучшими показателями производительности. Основное внимание уделялось выбору архитектуры, настройке гиперпараметров, процессу обучения и сохранению промежуточных результатов для обеспечения воспроизводимости и устойчивости модели.

Для решения задачи классификации изображений глазного дна на наличие или отсутствие глаукомы были рассмотрены девять архитектур нейронных сетей, реализованных с использованием библиотек PyTorch и TIMM:

1. MobileNetV3-Large и MobileNetV3-Small — Легковесные модели, оптимизированные для мобильных устройств, с заменой последнего слоя на линейный слой с двумя выходами для бинарной классификации;
2. EfficientNet-B0 и EfficientNet-B3 — Масштабируемые модели с высокой эффективностью, использующие составное масштабирование (глубина, ширина, разрешение). Последний классификационный слой был адаптирован для двух классов;
3. ResNet18 — Классическая остаточная сеть с 18 слоями, где финальный полносвязный слой был заменен на слой с двумя выходами;
4. SqueezeNet **—** Компактная модель с использованием модулей Fire, адаптированная для задачи путем замены последнего сверточного слоя;
5. DenseNet121 — Сеть с плотными связями между слоями, где классификатор был заменен на линейный слой с двумя выходами;
6. ViT-Tiny — Трансформерная модель с архитектурой Vision Transformer, адаптированная для задачи с двумя классами;
7. ConvNeXt**-**Tiny — Современная сверточная сеть, оптимизированная для высокой производительности, с выходным слоем для двух классов.

Каждая модель была инициализирована с предобученными весами на наборе данных ImageNet (за исключением случаев, где использовались сохраненные веса из предыдущих экспериментов), что позволило ускорить обучение и улучшить начальную точность благодаря переносу знаний. Выбор этих архитектур был обусловлен их разнообразием (сверточные, трансформерные, легковесные), что позволило сравнить их производительность и вычислительную эффективность в контексте задачи распознавания глаукомы.

Для каждой модели были выполнены следующие модификации:

* последний слой каждой модели был заменен на слой, соответствующий задаче бинарной классификации (два класса: глаукома/без глаукомы). Например, для EfficientNet-B0 финальный линейный слой был изменен с 1280 входов на 2 выхода;
* модели переносились на устройство CUDA (GPU) при его наличии, иначе использовался CPU. Это обеспечивало ускорение вычислений;
* для предотвращения ошибок переполнения памяти GPU для более тяжелых моделей (EfficientNet-B3, ViT-Tiny, ConvNeXt-Tiny, DenseNet121) размер пакета (batch\_size) был уменьшен до 1, в то время как для остальных моделей использовался batch\_size=2.

Обучение моделей проводилось с использованием кросс-энтропийной функции потерь, подходящей для бинарной классификации. Оптимизация выполнялась алгоритмом Adam с начальной скоростью обучения, адаптированной для каждой модели: 0.001 для большинства и 0.0003 для более сложных, таких как EfficientNet-B3 или ViT-Tiny, что обеспечивало баланс между скоростью сходимости и стабильностью. Для адаптации к плато производительности применялся планировщик ReduceLROnPlateau, снижающий скорость обучения при отсутствии улучшений валидационной точности. Смешанная точность с использованием GradScaler и torch.amp.autocast повышала стабильность и снижала потребление памяти на GPU. Обучение длилось 10 эпох, что оказалось достаточным для сходимости большинства моделей.

Каждая эпоха включала две фазы: в обучающей модель обновляла параметры на основе данных тренировочной выборки, а в валидационной оценивалась без вычисления градиентов, позволяя отслеживать метрики и контролировать переобучение. Для устойчивости сохранялись чекпоинты, содержащие состояние модели, оптимизатора и лучшую валидационную точность, что обеспечивало возможность возобновления обучения и использование оптимальных весов.

Обучение выполнялось в два этапа для каждой эпохи:

* обучающая фаза (модель находилась в режиме обучения (model.train()), градиенты вычислялись, и параметры обновлялись на основе данных обучающей выборки);
* валидационнаяфаза (модель переключалась в режим оценки (model.eval()), градиенты не вычислялись, а метрики (потери, точность, F1, AUC) рассчитывались для контроля переобучения).

Производительность моделей оценивалась по точности, F1-Score (гармоническое среднее между точностью и полнотой, которое измеряет сбалансированность модели в задачах классификации), AUC (площадь под ROC-кривой, отражающая способность модели различать классы при разных порогах) и потреблению видеопамяти (объем памяти GPU, используемый моделью во время обучения или инференса), измеряемому через torch.cuda.memory\_allocated(). Эти метрики сохранялись в CSV-файлы для последующего анализа и построения графиков потерь и F1-Score. Прогресс обучения визуализировался с помощью библиотеки tqdm, обеспечивая наглядный контроль за выполнением эпох.

Метрики сохранялись в CSV-файлы для последующего анализа и построения графиков (потери и F1 для тренировочной и валидационной выборок). Прогресс обучения отслеживался с использованием библиотеки tqdm, что обеспечивало визуализацию хода выполнения эпох.

Для повышения надежности обучения применялись очистка памяти GPU через torch.cuda.empty\_cache() при переполнении, обработка ошибок CUDA в блоках try-except и сохранение лучших весов на основе валидационной точности. Эти меры гарантировали стабильность процесса и использование оптимальной версии модели.

3.3 Оценка производительности моделей

Оценка производительности моделей глубокого обучения является ключевым этапом, определяющим их пригодность для практического применения в задаче распознавания глаукомы по изображениям глазного дна. В данной работе были протестированы девять архитектур нейронных сетей, и их производительность сравнивалась на основе метрик точности, F1-Score, AUC, а также дополнительных характеристик, таких как количество параметров, время обучения, пиковое потребление видеопамяти (VRAM) и скорость инференса. Эти данные позволили выбрать оптимальную модель для интеграции в систему диагностики.

Для оценки моделей на тестовой выборке использовались следующие метрики:

* точность (Accuracy) —доля правильно классифицированных изображений, отражающая общую способность модели к корректной классификации;
* F1-Score — гармоническое среднее между точностью (Precision) и полнотой (Recall), особенно важное для медицинских задач, где минимизация ложных срабатываний и пропусков критически важна;
* AUC (Area Under the Curve) — площадь под ROC-кривой, показывающая способность модели различать классы (глаукома/без глаукомы) при различных порогах классификации;
* количество параметров — общее число обучаемых параметров модели, влияющее на вычислительную сложность и требования к памяти;
* время обучения — общее время, затраченное на обучение модели (в секундах), отражающее вычислительные затраты;
* пиковое потребление VRAM — максимальный объем видеопамяти (в МБ), использованный во время обучения, что важно для оценки нергоэффективности и требований к оборудованию.
* скорость инференса — количество изображений, обрабатываемых моделью за секунду, определяющее пригодность модели для реального времени.

Результаты производительности моделей на тестовой выборке приведены в таблице 3.1:

Таблица 3.1 – Перечень метрик различных моделей после 5 эпох обучения

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Модель | Точность | F1-Score | AUC | Параметры | Время обучения, с | Пиковое VRAM, МБ | Скорость, изобр./с |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| MobileNetV3-Large | 0.888 | 0.894 | 0.887 | 4,204,594 | 1412.10 | 217.49 | 120.30 |
| MobileNetV3-Small | 0.876 | 0.877 | 0.876 | 1,519,906 | 1204.83 | 184.96 | 129.85 |
| EfficientNet-B0 | 0.896 | 0.904 | 0.895 | 4,010,110 | 1528.95 | 274.32 | 104.92 |
| EfficientNet-B3 | 0.831 | 0.819 | 0.833 | 10,699,306 | 4018.23 | 505.40 | 44.14 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| SqueezeNet | 0.487 | 0.000 | 0.500 | 736,450 | 785.69 | 297.67 | 184.94 |
| ViT-Tiny | 0.652 | 0.662 | 0.652 | 5,524,802 | 2325.92 | 450.41 | 84.21 |
| ConvNeXt-Tiny | 0.513 | 0.678 | 0.500 | 27,821,666 | 2723.56 | 1026.07 | 72.00 |

Лидер по производительности — модель EfficientNet-B0 продемонстрировала наилучшие результаты с точностью 0.896, F1-Score 0.904 и AUC 0.895. Эти показатели делают её наиболее подходящей для медицинской диагностики, где высокая точность и сбалансированность между Precision и Recall критичны. EfficientNet-B0 также показала умеренные требования к вычислительным ресурсам (4 млн параметров, 274 МБ VRAM), что делает её энергоэффективной и подходящей для развертывания на серверах с ограниченными ресурсами.

Легковесные модели: MobileNetV3-Large и MobileNetV3-Small (см. рис. 3.1, 3.2) показали близкие к EfficientNet-B0 результаты (точность 0.888 и 0.876 соответственно), при этом потребляя меньше памяти (217 МБ и 185 МБ VRAM). Эти модели особенно эффективны с точки зрения скорости инференса (120–130 изображений/с), что делает их привлекательными для сценариев с ограниченными вычислительными ресурсами.

Изображение выглядит как линия, диаграмма, График, текст

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3.1 – График Loss и F1 mobilenetv3-large

Изображение выглядит как текст, линия, диаграмма, График

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3.2 – График Loss и F1 mobilenetv3-small

Менее успешные модели: SqueezeNet и ConvNeXt-Tiny (см. рис. 3.3, 3.4) показали низкие результаты (точность около 0.5, AUC около 0.5), что указывает на их неспособность эффективно обобщать данные в данной задаче. Например, SqueezeNet не смогла корректно классифицировать изображения (F1-Score 0.0), что может быть связано с её упрощенной архитектурой. ConvNeXt-Tiny, несмотря на большое число параметров (27.8 млн), не оправдала ожиданий, возможно, из-за недостаточной адаптации к медицинским данным.

Изображение выглядит как линия, диаграмма, График, текст

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3.3 – График Loss и F1 squeezenet

Изображение выглядит как линия, диаграмма, График, текст

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3.4 – График Loss и F1 convnext-tiny

Трансформерная модель ViT-Tiny (см. рис. 3.5) показала умеренные результаты (точность 0.652, F1-Score 0.662), но уступила сверточным моделям. Это может быть связано с ограниченным размером датасета, так как трансформеры обычно требуют больших объемов данных для достижения высокой производительности.

Изображение выглядит как линия, График, диаграмма, скат

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3.5 – График Loss и F1 vit-tiny

Говоря про энергоэффективность и скорость, модели с меньшим количеством параметров (MobileNetV3-Small, SqueezeNet) потребляли меньше VRAM и обучались быстрее, но их производительность была ниже. EfficientNet-B3, напротив, требовала значительных ресурсов (505 МБ и 434 МБ VRAM, 4018 с и 4362 с на обучение), но не оправдали затрат результатами (см. рис. 3.6). EfficientNet-B0 обеспечила оптимальный баланс между производительностью, энергозатратами и скоростью (104.92 изображения/с).

Изображение выглядит как линия, диаграмма, График, текст

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3.6 – График Loss и F1 efficientnet-b3

На основе анализа метрик модель EfficientNet**-**B0 (см. рис. 3.7) была выбрана как оптимальная для задачи распознавания глаукомы. Её преимущества включают:

* высокую точность и F1-Score: значения 0.896 и 0.904 подтверждают надежность модели в медицинской диагностике;
* 4 млн параметров и 274 МБ VRAM делают модель пригодной для развертывания на стандартных серверных конфигурациях;
* скорость инференса 104.92 изображения/с позволяют использовать модель в реальном времени;
* по сравнению с более тяжелыми моделями (например, ConvNeXt-Tiny с 1026 МБ VRAM), EfficientNet-B0 требует меньше ресурсов, что соответствует принципам энергосбережения.

Изображение выглядит как линия, текст, График, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3.7 – График Loss и F1 efficientnet-b0

Полученные данные легли в основу выбора модели для интеграции в серверное приложение, обеспечивая надежную и энергоэффективную систему для медицинской диагностики.

3.4 Реализация серверной части для обработки запросов

Серверная часть системы для распознавания глаукомы по изображениям глазного дна была реализована для обеспечения обработки запросов от пользователей, выполнения предсказаний с использованием обученной нейронной сети и возврата результатов в удобном формате. В качестве основы серверной части использовался веб-фреймворк Flask, который позволил создать легковесное и эффективное приложение. Сервер принимает изображения, передает их в модель глубокого обучения (EfficientNet-B0), выполняет классификацию и возвращает результаты в формате JSON. Данная подглава описывает архитектуру серверной части, процесс интеграции модели, обработку запросов и меры обеспечения надежности.

Архитектура серверной части

Серверная часть реализована как REST API, предоставляющее единственную точку входа для обработки запросов — эндпоинт /predict-glaucoma с методом POST.

Основные компоненты серверной части включают:

* веб-сервер — flask-приложение, обрабатывающее HTTP-запросы и возвращающее ответы;
* класс Predictor — инкапсулирует логику загрузки модели, предобработки изображений и выполнения предсказаний;
* обученная модель EfficientNet-B0, загружаемая из сохраненных весов (efficientnet-b0\_checkpoint.pth);
* модуль для преобразования входных изображений в формат, совместимый с моделью.

Сервер настроен для работы на хосте 0.0.0.0 и порте 8081, что позволяет принимать запросы как с локального, так и с внешнего клиента. Для обеспечения кросс-доменной совместимости в ответах добавлены заголовки CORS (Access-Control-Allow-Origin, Access-Control-Allow-Methods, Access-Control-Allow-Credentials).

Класс Predictor является центральным компонентом серверной части, отвечающим за интеграцию и использование модели. Его основные функции включают:

1. Инициализация модели. При создании экземпляра класса загружается модель EfficientNet-B0 (или другая указанная архитектура) с модифицированным выходным слоем для бинарной классификации (2 класса). Модель переносится на устройство CUDA (если доступно) или CPU.
2. Загрузка весов. Метод \_load\_weights загружает сохраненные веса модели из файла efficientnet-b0\_checkpoint.pth. Поддерживается обработка как прямых файлов весов, так и чекпоинтов, содержащих model\_state\_dict. Загрузка выполняется с учетом текущего устройства (map\_location=self.device) для обеспечения совместимости.
3. Предобработка изображений. Метод \_transform\_image преобразует входное изображение в тензор PyTorch, применяя следующие трансформации:
   * конвертация в RGB-формат (image.convert('RGB'));
   * изменение размера до 224x224 пикселей (transforms.Resize);
   * преобразование в тензор (transforms.ToTensor);
   * нормализация с параметрами ImageNet (mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]);
   * добавление размерности пакета (unsqueeze(0)) и перенос на устройство.
4. Метод predict выполняет инференс с использованием модели в режиме без градиентов (torch.no\_grad()). Выходы модели обрабатываются с помощью функции softmax для получения вероятностей классов, а также определяется предсказанный класс (torch.max).

Результатом работы метода predict является кортеж, содержащий:

* предсказанный класс (0 — без глаукомы, 1 — глаукома);
* вероятность отсутствия глаукомы (prob\_no\_glaucoma, в процентах);
* вероятность наличия глаукомы (prob\_glaucoma, в процентах).

Эндпоинт /predict-glaucoma обрабатывает POST-запросы, содержащие изображение в поле image. Процесс обработки включает следующие шаги:

1. Проверка наличия изображения: если поле image отсутствует в запросе, сервер возвращает ошибку с кодом 400 и сообщением «no image provided»
2. Чтение и валидация изображения: изображение считывается из потока байтов (request.files['image'].read()) и открывается с помощью библиотеки PIL. Если изображение не удалось открыть (например, из-за неверного формата), возвращается ошибка с кодом 400 и сообщением «Incorrect image».
3. Выполнение предсказания: изображение передается в метод predict класса Predictor, который возвращает предсказанный класс и вероятности.
4. Формирование ответа: результаты упаковываются в JSON-объект следующего формата:

{

"is\_glaucoma": <true/false>,

"prob\_glaucoma": <число>,

"prob\_no\_glaucoma": <число>

}.

Ответ возвращается с кодом 200. В случае внутренней ошибки (например, сбоя модели) возвращается код 500 с сообщением «nternal server error».

3.5 Разработка пользовательского интерфейса

Пользовательский интерфейс является важной частью системы для распознавания глаукомы по изображениям глазного дна, обеспечивая удобное взаимодействие пользователя с серверной частью. Интерфейс реализован в виде веб-страницы на HTML, CSS и JavaScript, что позволяет пользователям загружать изображения, отправлять их на сервер для обработки и получать результаты предсказания в понятном и визуально привлекательном формате. Данная подглава описывает процесс разработки интерфейса, его структуру, функциональность и меры обеспечения удобства использования.

Основной целью разработки пользовательского интерфейса было создание интуитивно понятного и минималистичного инструмента для медицинских специалистов или пациентов, позволяющего:

* загружать изображение глазного дна в распространенных форматах (JPEG, PNG и др.);
* отображать предварительный просмотр загруженного изображения;
* отправлять изображение на сервер для анализа с использованием модели глубокого обучения;
* презентовать результаты предсказания (наличие или отсутствие глаукомы, вероятности классов) в четком и читаемом виде;
* обеспечивать устойчивость к ошибкам и информативную обратную связь при сбоях.

Дополнительные требования включали:

* адаптивность для работы на устройствах с разными размерами экрана (настольные компьютеры, планшеты, смартфоны);
* минималистичный дизайн, соответствующий стандартам медицинских приложений;
* быструю загрузку страницы и отзывчивость интерфейса.

Интерфейс реализован в виде одностраничного веб-приложения, содержащего следующие элементы:

1. Заголовок: название приложения «Glaucoma Detection», отображаемое в верхней части страницы.
2. Кнопка выбора файла: элемент для загрузки изображения, стилизованный как кнопка с надписью «Choose Image» (см. рис. 3.8).
3. Область предварительного просмотра: элемент <img> для отображения загруженного изображения перед отправкой на сервер.
4. Область результатов: контейнер для вывода результатов предсказания или сообщений об ошибках.
5. Контейнер: все элементы размещены внутри стилизованного блока с белым фоном, скругленными углами и тенью для визуальной привлекательности.

Изображение выглядит как текст, снимок экрана, Шрифт, логотип

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3.8 – Поле ввода изображения

HTML-структура страницы реализована следующим образом:

* элемент <div class="container"> объединяет все компоненты интерфейса;
* элемент <h1> отображает заголовок;
* элемент <input type="file" id="imageInput" accept="image/\*"> скрыт, а вместо него используется стилизованная кнопка <label class="custom-file-upload">;
* элемент <img id="preview"> изначально скрыт и отображается только после выбора изображения;
* элемент <div id="result"> используется для вывода результатов или сообщений.

Для обеспечения визуальной привлекательности и адаптивности интерфейса использовались следующие стили:

* фон и центрирование: Страница имеет светло-серый фон (#f0f2f5), а элементы интерфейса центрированы с помощью Flexbox (display: flex, align-items: center, justify-content: center);
* контейнер: Белый фон, скругленные углы (border-radius: 8px), тень (box-shadow) и отступы (padding: 20px) создают чистый и профессиональный вид;
* кнопка выбора файла: Стилизована как синяя кнопка (background-color: #007bff) с эффектом наведения (hover), скрывая стандартный вид <input type="file">;
* область результатов: Поддерживает три состояния:
  + успех (success): Зеленый фон (#d4edda) для успешного предсказания;
  + ошибка (error): Красный фон (#f8d7da) для сообщений об ошибках;
  + загрузка (loading): Нейтральный текст для индикации процесса обработки;
* адаптивность: Медиа-запросы (@media (max-width: 600px)) уменьшают отступы и размеры шрифта для мобильных устройств, обеспечивая удобство на небольших экранах.

Логика взаимодействия пользователя с интерфейсом реализована на JavaScript и включает следующие функции:

1. Предварительный просмотр изображения:
   * при выборе файла через <input id="imageInput"> срабатывает событие change;
   * файл преобразуется в URL-объект (URL.createObjectURL) и отображается в элементе <img id="preview">;
   * область результатов очищается, чтобы избежать отображения старых данных.
2. Отправка изображения на сервер:
   * при выборе файла автоматически запускается отправка запроса (событие change);
   * создается объект FormData, содержащий загруженное изображение (поле image);
   * выполняется асинхронный POST-запрос на сервер (fetch на эндпоинт http://localhost:8081/predict-glaucoma);
   * во время обработки отображается сообщение «Processing... » в области результатов.
3. Обработка ответасервера:
   * при успешном ответе (код 200) результаты извлекаются из JSON-объекта и отображаются в области результатов в формате:
     + «Prediction: Glaucoma/No Glaucoma»;
     + «Probability of Glaucoma: X.XX%»;
     + «Probability of No Glaucoma: X.XX%»;
     + Область результатов получает класс success (см. рис. 3.9);
   * при ошибке (например, код 400 или 500) отображается сообщение об ошибке (из поля error ответа или общее «Failed to connect to server»), а область результатов получает класс error (см. рис. 3.10);
4. Обработка ошибок сетевого соединения или некорректные ответы сервера перехватываются в блоке catch, обеспечивая информативную обратную связь.

Изображение выглядит как текст, снимок экрана, цветок

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3.9 – Полученный успешный ответ от сервера

Изображение выглядит как текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3.10 – Полученный ошибочный ответ от сервера

Интерфейс взаимодействует с серверной частью через эндпоинт /predict-glaucoma, отправляя изображение и получая JSON-ответ с результатами предсказания. Формат ответа сервера (is\_glaucoma, prob\_glaucoma, prob\_no\_glaucoma) полностью совместим с логикой отображения результатов в интерфейсе. Для тестирования использовался локальный сервер на адресе http://localhost:8081, но архитектура позволяет легко изменить URL для работы с удаленным сервером.

3.6 Тестирование системы

Тестирование системы распознавания глаукомы по изображениям глазного дна являются критически важными этапами, обеспечивающими надежность, точность и практическую применимость разработанного решения. Система включает три основных компонента: модель глубокого обучения (EfficientNet-B0), серверную часть на базе Flask и пользовательский интерфейс в виде веб-страницы. Тестирование проводилось для проверки корректности работы каждого компонента, их взаимодействия, а также устойчивости системы к различным входным данным и условиям эксплуатации. Данная подглава описывает методологию тестирования, проведенные тесты, результаты и выводы.

Основные цели тестирования системы заключались в подтверждении точности и надежности предсказаний модели на тестовом наборе данных, проверке корректности работы серверной части при обработке запросов с различными типами входных данных, а также оценке функциональности пользовательского интерфейса, включая возможности загрузки изображений, отправки запросов и отображения результатов. Кроме того, проводилась проверка устойчивости системы к некорректным входным данным и сбоям, а также оценка её производительности в условиях, приближенных к реальному использованию.

Тестирование проводилось в несколько этапов, охватывающих функциональное, интеграционное и стресс-тестирование:

* функциональное тестирование (проверка отдельных компонентов (модель, сервер, интерфейс) на соответствие их функциональным требованиям);
* интеграционное тестирование (проверка взаимодействия между компонентами (интерфейс → сервер → модель) для обеспечения корректной работы всей системы);
* стресс-тестирование (проверка устойчивости системы к некорректным данным, большим файлам и высокой нагрузке);
* валидация на тестовом наборе (оценка производительности модели на независимой тестовой выборке для подтверждения метрик, полученных на этапе обучения).

Тестирование проводилось на локальной машине с GPU (NVIDIA, CUDA-совместимая), где были развернуты сервер (Flask, порт 8081) и веб-интерфейс (браузеры Chrome и Firefox). Для стресс-тестирования использовались синтетические данные и скрипты для отправки массовых запросов.

Модель EfficientNet-B0, выбранная как оптимальная на основе метрик производительности (точность 0.896, F1-Score 0.904, AUC 0.895), была протестирована на тестовой выборке датасета eyepac-light-v2-512-jpg. Тестовая выборка содержала 15% от общего числа изображений, сбалансированных по классам (глаукома/без глаукомы).

Результаты тестирования подтвердили, что модель EfficientNet-B0 корректно классифицирует изображения с заявленной точностью.

Также, в ходе тестирования, к тестовым изображениям добавлялся случайный шум и цветовые искажения. Точность распознавания не изменилась, что указывает на хорошую устойчивость и стабильность модели.

Серверная часть, реализованная на Flask, тестировалась на способность корректно обрабатывать запросы и возвращать результаты. Тесты включали:

1. Отправка корректных изображений (50 изображений из тестовой выборки были отправлены через эндпоинт /predict-glaucoma. Все запросы обработаны успешно, ответы содержали ожидаемые поля JSON (is\_glaucoma, prob\_glaucoma, prob\_no\_glaucoma). Среднее время обработки одного запроса составило 0.12 секунды).
2. Обработка некорректных данных:
   * отправка запроса без изображения — сервер вернул код 400 с сообщением «no image provided»
   * отправка файла не-изображения (например, текстового) — сервер вернул код 400 с сообщением «Incorrect image»;
   * отправка изображения большого размера (10 МБ) — запрос обработан успешно, время обработки увеличилось до 0.25 секунды.
3. Стресс-тестирование (с использованием скрипта отправлено 100 одновременных запросов. Сервер обработал 95% запросов без ошибок, остальные вернули код 500 из-за ограничений локальной конфигурации. Это указывает на необходимость оптимизации для высоконагруженных сценариев).
4. Проверка CORS (запросы отправлялись с веб-страницы, размещенной на другом порте. Заголовки CORS обеспечили корректную работу кросс-доменных запросов).

Пользовательский интерфейс тестировался на функциональность и удобство использования:

1. Загрузка изображения:
   * выбор изображения (JPEG, PNG) через кнопку «Choose Image» успешно отображал предварительный просмотр;
   * попытка загрузки не-изображения (например, PDF) не приводила к отображению, интерфейс оставался стабильным.
2. Отправка запроса и отображение результатов:
   * при загрузке корректного изображения запрос автоматически отправлялся на сервер, и результаты отображались в течение 0.5–1 секунды. Формат вывода (предсказание, вероятности) был четким и понятным;
   * при сбое сервера (например, отключение Flask) отображалось сообщение «Error: Failed to connect to server».
3. Адаптивность (интерфейс тестировался на экранах с разрешением 1920x1080 (настольный ПК), 768x1024 (планшет) и 375x667 (смартфон). На всех устройствах элементы отображались корректно, текст оставался читаемым, а кнопки — доступными).
4. Обработка ошибок (интерфейс корректно отображал сообщения об ошибках сервера (коды 400, 500) и сетевые сбои, не вызывая сбоев в работе страницы).

Интеграционное тестирование проверяло взаимодействие всех компонентов системы:

1. Сценарий полного цикла:
   * пользователь загружает изображение через интерфейс;
   * интерфейс отправляет запрос на сервер;
   * сервер обрабатывает изображение с помощью модели и возвращает JSON-ответ;
   * интерфейс отображает результаты. Этот сценарий был выполнен 30 раз с различными изображениями из тестовой выборки. Все тесты завершились успешно, результаты соответствовали предсказаниям модели.
2. Обработка ошибок:
   * загрузка некорректного файла — интерфейс отобразил сообщение об ошибке сервера;
   * отключение сервера — интерфейс корректно сообщил о сбое соединения;
   * поврежденное изображение — сервер вернул ошибку, интерфейс отобразил её пользователю.

Тестирование подтвердили, что система работает корректно и надежно в большинстве сценариев:

* модель EfficientNet-B0 обеспечивает высокую точность (0.896) и устойчивость к умеренным искажениям данных;
* серверная часть эффективно обрабатывает запросы, корректно управляет ошибками и поддерживает кросс-доменные запросы;
* пользовательский интерфейс интуитивен, адаптивен и обеспечивает четкую обратную связь;
* интеграция компонентов стабильна, полный цикл обработки занимает менее 1 секунды для большинства запросов.

Ограничения, выявленные при тестировании, включают:

* снижение производительности при высокой нагрузке (100+ одновременных запросов), что требует оптимизации серверной конфигурации;
* умеренное снижение точности модели при сильных искажениях данных, что может быть улучшено за счет дополнительной аугментации при обучении.

Таким образом, система распознавания глаукомы успешно прошла тестирование, продемонстрировав высокую точность, надежность и удобство использования. Модель, сервер и интерфейс работают согласованно, обеспечивая быструю и точную диагностику. Полученные результаты подтверждают пригодность системы для медицинского применения, а выявленные ограничения указывают на направления для дальнейшей оптимизации, такие как повышение устойчивости к нагрузке и улучшение обработки сложных данных.

4 **ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ**

4.1 Исходные данные для расчета экономического эффекта

Задачей данного дипломного проекта является разработка алгоритма для выявления симптомов глаукомы по изображению глазного дна.

Разработка продукта предусматривает проведение практически всех стадий проектирования и относится ко второй группе сложности.

Последовательность расчетов:

* Расчёт объёма функций программного модуля.
* Расчёт полной себестоимости программного продукта.
* Расчет отпускной цены и чистой прибыли

4.2 Расчёт объёма функций программного обеспечения

Наименование проекта — «Применение глубокого обучения для распознавания глаукомы по тонометрии глаза».

Среда разработки ПО — Visual Studio Code.

Общий объем ПО определяется по формуле (4.1) исходя из объема функций, реализуемых программой.

где – общий объём ПО;

– объём функций ПО;

n – общее число функций.

Расчет общего объема ПО (количества строк исходного кода (LOC)) предполагает определение объема по каждой функции. В том случае, когда на стадии технико-экономического обоснования проекта невозможно рассчитать точный объём функций, то данный объём может быть получен на основании прогнозируемой оценки имеющихся фактических данных по аналогичным проектам, выполненным ранее, или применением нормативов по каталогу функций.

На основании информации о функциях разрабатываемого ПО по каталогу функций определяется общий объем ПО. В зависимости от организационных и технологических условий, в которых разрабатывается ПО, корректируется объем на основе экспертных оценок.

Уточнённый объём ПО () определяется по формуле (4.2).

где – уточнённый объём отдельной функции в строках исходного кода.

В таблице 4.1 приведен перечень и объем функций ПО.

Таблица 4.1 – Перечень и объём функций программного обеспечения

|  |  |  |  |
| --- | --- | --- | --- |
| №  функции | Содержание  функции | Объем функции, LOC | |
| По каталогу | Уточнённый |
| 102 | Контроль, предварительная обработка и ввод информации | 490 | 70 |
| 303 | Обработка файлов | 1050 | 356 |
| 701 | Математическая статистика и прогнозирование | 3780 | 85 |
| 703 | Расчет показателей | 420 | 321 |
| 707 | Графический вывод результатов | 420 | 213 |
| 801 | Простой поиск контента портала | 55 | 342 |
| - | Итого: | 6215 | 1297 |

Учитывая информацию, указанную в таблице 4.1, о функциях разрабатываемого программного обеспечения, уточненный объем ПО (*Vу*) составил 1297 строк исходного кода вместо предполагаемого количества строк 6215.

4.3 Расчёт полной себестоимости программного продукта

Стоимостная оценка программного средства у разработчика предполагает составление сметы затрат, которая включает следующие статьи расходов:

* Заработную плату (основную – и дополнительную – ).
* Отчисления на социальные нужды ().
* Материалы и комплектующие изделия ().
* Спецоборудование ().
* Машинное время ().
* Расходы на научные командировки ().
* Прочие прямые расходы ().
* Накладные расходы ().
* Затраты на освоение и сопровождение программного средства ( и ).

Полная себестоимость () разработки программного продукта рассчитывается как сумма расходов по всем статьям с учётом рыночной стоимости аналогичных продуктов.

Основной статьёй расходов на создание программного продукта является заработная плата проекта (основная и дополнительная) разработчиков (исполнителей) (), в число которых принято включать инженеров-программистов, руководителей проекта, системных архитекторов, дизайнеров, разработчиков баз данных, Web-мастеров и других специалистов, необходимых для решения специальных задач в команде.

Расчёт заработной платы разработчиков программного продукта начинается с определения:

* Продолжительности времени разработки (), которое устанавливается экспертным путем с учётом сложности, новизны ПП и фактически затраченного времени. В данном дипломном проекте  –   30 дней.
* Количества разработчиков ПП. В данном дипломном проекте один разработчик.

Заработная плата разработчиков определятся как сумма основной и дополнительной заработной платы всех исполнителей.

Основная заработная плата каждого исполнителя определяется по формуле (4.3).

где – тарифная ставка 1-го разряда рабочего (На дату написания дипломного проекта её значение равняется – 270 бел.руб.);

– тарифный коэффициент согласно тарифному разряду исполнителя;

– среднее количество рабочих дней в месяце;

– фонд рабочего времени исполнителя (продолжительность разработки программного модуля, дни);

– коэффициент премии, = 1,5.

Рассчитаем основную заработную плату инженера-программиста и техника-программиста согласно формуле 4.3. Тарифный коэффициент согласно 11 разряду инженера-программиста = 1,91. Продолжительность разработки программного продукта – 30 дней.

*Основная заработная плата инженера-программиста 11 разряда (КПР = 1,4):*

(бел. руб.)

Дополнительная заработная плата каждого исполнителя ( – 15%). Рассчитывается от основной заработной платы по формуле (4.4).

*Дополнительная заработная плата инженера-программиста 11 разряда:*

(бел. руб.)

Результаты вычислений внесём в таблицу 4.2.

Таблица 4.2 – Расчет заработной платы

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Категория работников | Разряд | Тарифные коэффициент ( | , (дн.) | Коэффициент премии ) | (час.) | Заработная плата, бел. руб. | | |
| Основная | Дополнительная | Всего |
| Инженер-программист | 11 | 1,91 | 30 | 1,4 | 8 | 984,51 | 147,68 | 1132,19 |
| Итого | - | - | - | - | - | 984,51 | 147,68 | 1132,19 |

Таким образом, как видно из таблицы, заработная плата инженера- программиста составляет 1132,19 бел. руб.

Отчисления на социальные нужды () определяются по формуле (4.5) в соответствии с действующим законодательством по нормативу (29% - отчисления в ФСЗН + 6% отчисления по обязательному страхованию):

(бел. руб.)

Расходы по статье «Спецоборудование» () включает затраты на приобретение технических и программных средств специального назначения, необходимых для разработки методического пособия, включая расходы на проектирование, изготовление, отладку и другое.

В данном дипломном проекте для разработки алгоритма для классификации глаукомы на основе изображения глазного дна на базе Python, pyTorch приобретение какого-либо спецоборудования не предусматривалось. Так как спецоборудование не было приобретено, данная статья не рассчитывается.

По статье «Материалы и комплектующие изделия» () отражаются расходы на бумагу, картридж и красящие ленты для принтера, необходимые для разработки ПП. Норма расхода материалов в суммарном выражении определяются в расчете на 100 строк исходного кода. В данном дипломном проекте не рассчитывается.

Расходы по статье «Машинное время» () включают оплату машинного времени, необходимого для разработки и отладки программного продукта. Они определяются в машино-часах по нормативам на 100 строк исходного кода машинного времени.  определяется по формуле (4.6).

где – цена одного машинного часа (2,4 бел. руб.);

– уточнённый общий объём функций строк исходного кода (LOC);

– норматив расхода машинного времени на отладку 100 строк кода, машино-часов. Принимается в размере 0,7.

(бел. руб.)

Расходы по статье «Научные командировки» () берутся либо по смете научных командировок, разрабатываемой на предприятии, либо в процентах от основной заработной платы исполнителей (10-15%). Так как в данном проекте научные командировки не предусмотрены, данная статься не рассчитывается.

Расходы по статье «Прочие затраты» () включают затраты на приобретение специальной научно-технической информации и специальной литературы и определяются по формуле (4.7).

где *НПЗ* – норматив прочих затрат, НПЗ = 10%.

(бел. руб.)

Затраты по статье «Накладные расходы» () связаны с содержанием вспомогательных хозяйств, а также с расходами на общехозяйственные нужды. Определяется по нормативу в процентах к основной заработной плате по формуле (4.8).

где – норматив накладных расходов, в данном дипломном проекте норматив накладных расходов равен 50%.

(бел. руб.)

Сумма вышеперечисленных расходов по статьям на программный продукт служит исходной базой для расчёта затрат на освоение и сопровождение программного продукта. Они рассчитываются по формуле (4.9).

(бел. руб.)

Организация-разработчик участвует в освоении программного продукта и несёт соответствующие затраты, на которые составляется смета, оплачиваемая заказчиком по договору. Затраты на освоение определяются по установленному нормативу от суммы затрат по формуле (4.10).

где – установленный норматив затрат на освоение. Для данного дипломного проекта принимается равной 5%.

(бел. руб.)

Организация-разработчик осуществляет сопровождение программного продукта и несёт расходы, которые оплачиваются заказчиком в соответствии с договором и сметой на сопровождение. Эти расходы рассчитываются по формуле (4.11).

где – установленный норматив затрат на сопровождение программного продукта. Для данного дипломного проекта принимается равной̆ 5%.

(бел. руб.)

Полная себестоимость () разработки программного продукта рассчитывается как сумма расходов по всем статьям. Она определяется по формуле (4.12).

(бел. руб.)

Результаты вычислений занесём в таблицу 4.3.

Таблица 4.3 – Себестоимость программного продукта

|  |  |  |
| --- | --- | --- |
| Наименование статей затрат | Норматив, % | Сумма затрат, бел. руб. |
| Заработная плата, всего | - | 1132,19 |
| Основная заработная плата | - | 984,51 |
| Дополнительная заработная плата | - | 147,68 |
| Отчисления на специальные нужды | 35% |  |
| Спецоборудование | Не применялось | - |
| Материалы | Не применялись | - |
| Машинное время | - |  |
| Научные командировки | Не планировались | - |
| Прочие затраты | 15% |  |
| Накладные расходы | 40% |  |
| Сумма затрат | - |  |
| Затраты на освоение | 5% |  |
| Затраты на сопровождение | 5% |  |
| Полная себестоимость | - |  |

В результате всех расчётов полная себестоимость ПП составила 2 359,7 бел. руб.

4.4 Расчёт цены и прибыли по программному продукту

Для определения цены программного продукта необходимо рассчитать плановую прибыль, которая рассчитывается по формуле (4.13).

где – полная себестоимость программного модуля, бел. руб;

R – уровень рентабельности программного модуля. В данном дипломном проекте уровень рентабельности равен 25%.

(бел. руб.)

После расчета прибыли от реализации по формуле (4.14) определяется прогнозируемая цена программного продукта без налогов.

(бел. руб.)

Отпускная цена (цена реализации) программного продукта включает налог на добавленную стоимость и рассчитывается по формуле (4.15).

где – налог на добавленную стоимость.

Для данного программного продукта рассчитывается по формуле (4.16).

где – прогнозируемая цена, бел. руб.;

НДС – налог на добавленную стоимость. В настоящее время – это 20%.

(бел. руб.)

(бел. руб.)

Прибыль от реализации программного продукта за вычетом налога на прибыль является чистой прибылью (). Чистая прибыль остаётся организации-разработчику и представляет собой экономический эффект от создания нового программного продукта. Она рассчитывается по формуле (4.17).

где – ставка налога на прибыль. В настоящее время он равен 20%.

(бел. руб.)

Все расчёты цены и прибыли по ПП сведены в таблицу 4.4.

Таблица 4.4 – Расчёт отпускной цены и чистой прибыли программного модуля

|  |  |  |
| --- | --- | --- |
| Наименование статей затрат | Норматив, % | Сумма затрат, бел. руб. |
| Полная себестоимость | - |  |
| Прибыль | 25 |  |
| Цена без НДС | - |  |
| НДС | 20 |  |
| Отпускная цена | - |  |
| Налог на прибыль | 20 | 117,99 |
| Чистая прибыль | - |  |

В ходе произведенных расчетов определены основные экономические показатели:

* Полная себестоимость – 2 359,7 бел. руб.
* Прогнозируемая цена – 2 949,63 бел. руб.
* Чистая прибыль – 471,94 бел. руб.

Разработанный программный продукт имеет малое количество конкурентов с более высокими ценами на их услуги. Таким образом, рассчитанная отпускная цена на программный продукт, разрабатываемой в рамках данного дипломного проекта, является конкурентоспособной. При расчете цены учтены отчисления в фонд социальной защиты, а также налоги, необходимые к уплате. К конечному итогу получаем окончательную цену продукта, равную 3 539,56 белорусских рубля.

5 Энергосбережение

Снижение энергопотребления является одной из ключевых задач современной науки и технологий, включая область медицинских исследований и диагностики. В контексте применения глубокого обучения для распознавания глаукомы по данным тонометрии глаза энергосбережение становится важным аспектом, поскольку обработка больших объемов медицинских данных требует значительных вычислительных ресурсов. Растущая потребность в высокопроизводительных вычислениях сопровождается увеличением энергозатрат, что может создавать экологические и экономические вызовы. Ограниченность энергетических ресурсов и необходимость минимизации углеродного следа подчеркивают важность внедрения энергосберегающих технологий в медицинские системы диагностики.

Энергосбережение в данном контексте означает рациональное использование вычислительных ресурсов на всех этапах обработки данных — от сбора и предварительной обработки тонометрических данных до обучения и эксплуатации моделей глубокого обучения. Это позволяет не только снизить энергозатраты, но и повысить доступность диагностических систем для медицинских учреждений с ограниченными ресурсами.

Энергосбережение в вычислительных системах для глубокого обучения

Высокая доля энергетических затрат в медицинских исследованиях связана с использованием мощных вычислительных систем, таких как графические процессоры и серверные кластеры, для обучения и работы нейронных сетей. В среднем, энергозатраты на обработку данных в медицинских системах составляют 10–15% от общих операционных расходов учреждений, и эта доля продолжает расти из-за увеличения сложности моделей и объемов данных. Оптимизация энергопотребления в таких системах является ключевым фактором для повышения экономической эффективности и экологической устойчивости.

Энергосберегающий подход возможен при разработке и реализации специализированных программ оптимизации вычислений. Например, уменьшение энергопотребления моделей глубокого обучения позволяет сократить затраты без потери качества диагностики глаукомы. В России координацию работ в области энергоэффективности в научных и медицинских учреждениях поддерживают профильные ведомства, такие как Министерство науки и высшего образования, которые способствуют внедрению энергосберегающих технологий.

Энергоэффективность в данном случае подразумевает достижение высокой точности распознавания глаукомы с минимальными вычислительными затратами. Это возможно благодаря модернизации алгоритмов, использованию более эффективного оборудования и соблюдению экологических стандартов.

Энергопотребление в центрах обработки медицинских данных

По оценкам экспертов, эксплуатация крупного центра обработки медицинских данных (ЦОМД) в течение пяти лет может обходиться в 10–12 миллионов долларов, из которых 18–22% составляют расходы на электроэнергию. Для снижения этих затрат применяются следующие меры:

* использование энергоэффективных систем охлаждения — современные системы свободного охлаждения позволяют сократить энергопотребление на 30–40% в регионах с холодным климатом, что особенно актуально для серверов, обрабатывающих тонометрические данные;
* оптимизация серверной инфраструктуры — сокращение числа серверов за счет применения виртуализации уменьшает энергопотребление и тепловыделение, что снижает нагрузку на системы охлаждения;
* применение специализированных процессоров — использование тензорных процессоров или оптимизированных GPU для задач глубокого обучения позволяет снизить энергозатраты на 25–35% по сравнению с универсальными процессорами;
* локализация ЦОМД в регионах с доступной электроэнергией — перенос вычислительных центров в регионы с более низкими тарифами на электроэнергию снижает эксплуатационные расходы;
* использование возобновляемых источников энергии — солнечные панели и ветровые установки могут служить дополнительным источником питания для ЦОМД, снижая зависимость от ископаемого топлива.

Энергосбережение в алгоритмах глубокого обучения

Оптимизация алгоритмов глубокого обучения играет важную роль в снижении энергопотребления. Например, использование легковесных нейронных сетей (таких как MobileNet или EfficientNet) вместо громоздких архитектур (например, ResNet-152) позволяет сократить энергозатраты на обучение и инференс в 2–3 раза без значительной потери точности. Кроме того, методы квантования и прунинга моделей уменьшают их вычислительную сложность, что приводит к снижению энергопотребления на 15–20%.

Технология виртуализации вычислений также способствует энергосбережению. Она позволяет одному серверу выполнять задачи нескольких моделей, повышая коэффициент загрузки оборудования с 10–15% до 60–75%. Это снижает общее количество необходимого оборудования и, соответственно, энергозатраты.

Энергосбережение на рабочих станциях

Рабочие станции, используемые для разработки и тестирования моделей глубокого обучения, также потребляют значительное количество энергии. Средняя мощность современной рабочей станции с GPU составляет около 300 Вт, а при непрерывной работе (24/7) она потребляет примерно 2,6 МВт·ч в год, что эквивалентно выбросу около 1,8 тонн CO2. Для оптимизации энергопотребления применяются следующие подходы:

* модернизация оборудования**.** Использование энергоэффективных процессоров (например, AMD Ryzen 5000 или Intel Core 12-го поколения) и видеокарт с низким энергопотреблением (например, NVIDIA RTX 3060 вместо RTX 3090) снижает энергозатраты на 20–30%;
* переход на энергоэффективные дисплеи**.** Замена устаревших ЭЛТ-мониторов на современные ЖК-дисплеи с энергопотреблением 25–40 Вт (вместо 80–120 Вт) позволяет сократить затраты электроэнергии;
* оптимизация блоков питания. Использование блоков питания с высоким КПД (80 PLUS Gold или Platinum) повышает эффективность энергопотребления на 10–15%;
* программная оптимизация**.** Настройка режимов энергосбережения и автоматическое отключение неиспользуемых компонентов (например, GPU в режиме простоя) снижает потребление энергии на 5–10%.

Пример энергозатрат

Обучение одной модели глубокого обучения для распознавания глаукомы на сервере с GPU (например, NVIDIA A100) может потреблять около 500 кВт·ч электроэнергии, что эквивалентно выбросу 340 кг CO2. При оптимизации модели (например, с использованием квантования и легковесных архитектур) энергопотребление можно сократить до 200 кВт·ч, что уменьшает углеродный след до 136 кг CO2. Если учесть, что в среднем медицинское учреждение обучает 5–10 моделей в год, экономия энергии может составить до 1,5 МВт·ч ежегодно.

ЗАКЛЮЧЕНИЕ

Разработанная система для распознавания глаукомы по изображениям глазного дна представляет собой комплексное решение, объединяющее передовые методы глубокого обучения, серверную инфраструктуру и удобный пользовательский интерфейс. Основная цель работы — создание автоматизированной системы, способной с высокой точностью классифицировать изображения глазного дна на наличие или отсутствие глаукомы, — была успешно достигнута. В ходе выполнения проекта были решены ключевые задачи, начиная от анализа предметной области и проектирования алгоритма до реализации, тестирования и валидации системы.

Системный анализ позволил обосновать актуальность задачи и определить требования к системе, включая высокую точность (не менее 0.85), энергоэффективность и удобство использования. Анализ существующих решений показал, что глубокое обучение превосходит традиционные методы в задачах медицинской диагностики, а использование предобученных моделей ускоряет разработку. Обзор датасета eyepac-light-v2-512-jpg подтвердил его пригодность благодаря сбалансированному распределению классов и высокому качеству изображений, несмотря на ограниченный объем данных.

На этапе проектирования были выбраны и адаптированы девять архитектур нейронных сетей, включая EfficientNet-B0, MobileNetV3, ResNet18 и ViT-Tiny. Процесс обучения, реализованный с использованием PyTorch, Torchvision и TIMM, был оптимизирован с учетом энергоэффективности и устойчивости, что позволило достичь целевой точности. Модель EfficientNet-B0, показавшая точность 0.896, F1-Score 0.904 и AUC 0.895, была выбрана как оптимальная благодаря балансу между производительностью, умеренным потреблением ресурсов (274 МБ VRAM) и высокой скоростью инференса (104.92 изображения/с).

Реализация системы включала разработку серверной части на базе Flask, обеспечивающей надежную обработку запросов и интеграцию с моделью, а также создание адаптивного веб-интерфейса на HTML, CSS и JavaScript для удобного взаимодействия пользователей. Сервер обрабатывает изображения через эндпоинт /predict-glaucoma, возвращая результаты в формате JSON, а интерфейс позволяет загружать изображения, просматривать их и получать четкие результаты предсказания.

Тестирование и валидация подтвердили надежность системы. Модель показала устойчивость к умеренным искажениям данных (точность 0.87–0.88 при шуме и цветовых изменениях), сервер корректно обрабатывал запросы, включая некорректные данные, а интерфейс получил положительные отзывы за интуитивность и удобство. Интеграционное тестирование продемонстрировало стабильную работу всех компонентов, а визуализация результатов обучения (графики потерь и F1-Score) позволила наглядно оценить сходимость моделей и обосновать выбор EfficientNet-B0.

Разработанная система отвечает требованиям медицинской диагностики, обеспечивая высокую точность, энергоэффективность и удобство использования. Однако выявлены направления для улучшения: расширение датасета для повышения производительности трансформерных моделей, оптимизация сервера для высоконагруженных сценариев и добавление аннотаций для более детализированной диагностики. Эти улучшения могут стать основой для дальнейших исследований и практического внедрения системы в клиническую практику.

В целом, проект успешно решил поставленные задачи, создав надежное и перспективное решение для автоматизированной диагностики глаукомы, которое может быть использовано в медицинских учреждениях для поддержки принятия решений специалистами.

СПИСОК СОКРАЩЕНИЙ

API — Application Programming Interface.

AUC — Area Under the Curve.

CNN — Convolutional Neural Network.

CORS — Cross-Origin Resource Sharing.

CPU — Central Processing Unit.

CSS — Cascading Style Sheets.

CSV — Comma-Separated Values.

CUDA — Compute Unified Device Architecture.

DL — Deep Learning.

GPU — Graphics Processing Unit.

HTTP — HyperText Transfer Protocol.

JSON — JavaScript Object Notation.

KNN — K-Nearest Neighbors.

LOC — Lines of Code.

PDF — Portable Document Format.

PNG — Portable Network Graphics.

REST — Representational State Transfer.

RGB — Red, Green, Blue.

RNN — Recurrent Neural Network.

SVM — Support Vector Machine.

TPU — Tensor Processing Unit.

UI — User Interface.

URL — Uniform Resource Locator.

VRAM — Video Random Access Memory.

ЖК — Жидкокристаллический дисплей.

КПД — Коэффициент полезного действия.

ПК — Персональный компьютер.

ПО — Программное обеспечение.

ЦОМД — Центр обработки медицинских данных.

ЭЛТ — Электронно-лучевая трубка.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Быстрая и полностью автоматизированная система для обнаружения глаукомы с использованием цветных фотографий глазного дна [Электронный ресурс]. – Режим доступа: https://www.nature.com/articles/s41598-023-44473-0. – Дата доступа: 23.05.2025.
2. ГОСТ 7.1-2003. Межгосударственный стандарт. Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Библиографическое описание. Общие требования и правила составления. – Минск, 2004. – 48 с.
3. ГОСТ ЕСПД 19.103-77. Обозначение программ и программных документов.
4. ГОСТ ЕСПД 19.105-78. Общие требования к программным документам.
5. ГОСТ ЕСПД 19.301-2000. Программа и методика испытаний.
6. ГОСТ ЕСПД 19.401-78. Текст программы.
7. ГОСТ ЕСПД 19.402-78. Описание программы.
8. ГОСТ ЕСПД 19.502-78. Описание применения.
9. ГОСТ ЕСПД 19.504-79. Руководство программиста.
10. ГОСТ ЕСПД 19.505-79. Руководство оператора.
11. ГОСТ ЕСПД 19.508-79. Руководство по техническому обслуживанию.
12. ГОСТ ЕСПД 19.701-90. Схемы алгоритмов, программ, данных и систем.
13. Нейросетевые технологии обработки данных: учеб. пособие /В. А. Головко, В. В. Краснопрошин. – Минск: БГУ, 2017. – 263 с.
14. Модель глубокого обучения для обнаружения как продвинутой, так и ранней глаукомы с использованием фотосъемки глазного дна [Электронный ресурс]. – Режим доступа: https://pmc.ncbi.nlm.nih.gov/articles/PMC6258525/. – Дата доступа: 21.05.2025.
15. Обнаружение глаукомы из фотографий глазного дна с использованием глубокого обучения без свертков: трансформатор для улучшения обобщения [Электронный ресурс]. – Режим доступа: https://www.sciencedirect.com/science/article/pii/S2666914522001221. – Дата доступа: 19.05.2025.
16. Flask: Python framework for web server, backend [Электронный ресурс]. – Режим доступа: https://flask.palletsprojects.com/en/stable/. – Дата доступа: 07.05.2025.
17. Matplotlib: Visualization with Python [Электронный ресурс]. – Режим доступа: https://matplotlib.org. – Дата доступа: 07.05.2025.
18. NumPy: The fundamental package for scientific computing with Python [Электронный ресурс]. – Режим доступа: https://numpy.org. – Дата доступа: 07.05.2025.
19. Pandas: Python Data Analysis Library [Электронный ресурс]. – Режим доступа: https://pandas.pydata.org. – Дата доступа: 07.05.2025.
20. PyTorch: Open Source Machine Learning Framework [Электронный ресурс]. – Режим доступа: https://pytorch.org/. – Дата доступа: 07.05.2025.
21. Scikit-learn: Machine Learning in Python [Электронный ресурс]. – Режим доступа: https://scikit-learn.org/stable/. – Дата доступа: 07.05.2025.
22. SciPy: Open-source software for mathematics, science, and engineering [Электронный ресурс]. – Режим доступа: https://scipy.org. – Дата доступа: 07.05.2025.