**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 5](#_Toc197521574)

[1. СИСТЕМНЫЙ АНАЛИЗ И ПОСТАНОВКА ЗАДАЧИ 6](#_Toc197521575)

[1.1 Описание предметной области 6](#_Toc197521576)

[1.2 Анализ существующих решений 7](#_Toc197521577)

[1.3 Основы структуры человеческого глаза 8](#_Toc197521578)

[1.5 Основы тонометрии 9](#_Toc197521579)

[1.5 Обоснование необходимости реализации 10](#_Toc197521580)

[1.6 Постановка задачи 11](#_Toc197521581)

[2. ПРОЕКТИРОВАНИЕ АЛГОРИТМА 13](#_Toc197521582)

[2.1 Анализ инструментов для реализации 13](#_Toc197521583)

[2.2 Обзор набора данных тонометрии 14](#_Toc197521584)

[2.3 Проектирование архитектуры нейронных сетей 15](#_Toc197521585)

[3. ОПИСАНИЕ РЕАЛИЗАЦИИ И ИСПЫТАНИЯ АЛГОРИТМА 18](#_Toc197521586)

[3.1 Подготовка и предобработка данных для обучения модели 18](#_Toc197521587)

[3.2 Архитектура и обучение нейронных сетей 20](#_Toc197521588)

[3.3 Архитектура и обучение нейронных сетей 23](#_Toc197521589)

[3.4 Реализация серверной части для обработки запросов 26](#_Toc197521590)

[3.5 Разработка пользовательского интерфейса 28](#_Toc197521591)

[3.6 Тестирование и валидация системы 31](#_Toc197521592)

[4. ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ 36](#_Toc197521593)

[4.1 Исходные данные для расчета экономического эффекта 36](#_Toc197521594)

[4.2 Расчёт объёма функций программного обеспечения 36](#_Toc197521595)

[4.3 Расчёт полной себестоимости программного продукта 37](#_Toc197521596)

[4.4 Расчёт цены и прибыли по программному продукту 42](#_Toc197521597)

[5. Энергосбережение 45](#_Toc197521598)

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 (ВГД), 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 (ИИ) и глубокого обучения (Deep Learning, DL) 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.

**Традиционные методы анализа**

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.

**Методы машинного обучения**

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

* SVM (метод опорных векторов);
* KNN (метод ближайших соседей);
* 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.

**Глубокое обучение**

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

* CNN (сверточные нейронные сети) 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.

**Сравнительный анализ**

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.

1.5 Основы тонометрии

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

**Нормальные значения внутриглазного давления**

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.
* **Положение тела** – 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 (ВОЗ), 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 (ВГД) 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

Основной задачей данной работы является разработка и обучение модели глубокого обучения для автоматической классификации глаукомы на основе данных тонометрии глаза. Для решения этой задачи необходимо выполнить несколько ключевых этапов, которые включают сбор, предобработку и анализ данных, а также разработку и обучение нейросетевой модели. Важными аспектами являются:

1. **Предобработка и анализ данных**  
   Необходимо собрать и подготовить набор данных, включающий изображения глаз, полученные с использованием тонометрии, а также соответствующие метки, указывающие на наличие или отсутствие глаукомы. В процессе работы с данными нужно выполнить следующие шаги:
   * Провести очистку данных от шума и артефактов.
   * Применить методы аугментации для увеличения объема обучающих данных и повышения обобщающих способностей модели.
   * Подготовить данные для нормализации и стандартизации, чтобы улучшить обучение модели.
2. **Выбор и разработка архитектуры нейронной сети**  
   Необходимо выбрать подходящую архитектуру нейронной сети, способную эффективно классифицировать изображения. Для данной задачи предполагается использование современных моделей глубокого обучения, таких как сверточные нейронные сети (CNN), которые зарекомендовали себя в области обработки изображений. Важно, чтобы выбранная модель могла:
   * Извлекать признаки из изображений глаз, связанные с особенностями развития глаукомы.
   * Обрабатывать изображения разного качества и с различными уровнями шума.
   * Добиться высокой точности классификации при ограниченных вычислительных ресурсах.
3. **Обучение и настройка модели**  
   После выбора архитектуры сети необходимо обучить модель на подготовленных данных. В этом процессе ключевыми задачами являются:
   * Разработка оптимальной стратегии обучения, включая настройку гиперпараметров (например, скорости обучения, размера пакета и числа эпох).
   * Применение методов регуляризации, таких как dropout, для предотвращения переобучения.
   * Оценка производительности модели на валидационном наборе данных и корректировка параметров для достижения наилучших результатов.
4. **Оценка точности и эффективности модели**  
   После обучения модели необходимо провести оценку её точности на тестовом наборе данных. Это включает в себя:
   * Вычисление точности, полноты, F1-меры и других метрик для оценки качества классификации.
   * Анализ ошибок модели и возможных причин недочетов.
   * Сравнение полученных результатов с существующими методами диагностики глаукомы, чтобы выявить преимущества и недостатки разработанного подхода.
5. **Интерпретация и применение результатов**  
   На основе полученной модели необходимо сформулировать рекомендации по её применению в медицинской практике. Это может включать:
   * Разработку программного интерфейса для использования модели в реальных клинических условиях.
   * Оценку возможности интеграции модели в системы телемедицины для удаленной диагностики.
   * Рекомендации по дальнейшему улучшению модели, включая использование дополнительных данных и улучшение архитектуры сети.

Таким образом, постановка задачи заключается в создании эффективного инструмента для диагностики глаукомы на основе анализа данных тонометрии, что позволит улучшить доступность и точность диагностики этого заболевания, а также снизить зависимость от человеческого фактора в интерпретации результатов.

2. ПРОЕКТИРОВАНИЕ АЛГОРИТМА

2.1 Анализ инструментов для реализации

Для реализации задачи классификации изображений с использованием глубоких нейронных сетей были выбраны следующие инструменты и библиотеки:

1. **PyTorch** – это мощная библиотека для построения нейронных сетей, которая является основой для разработки и обучения моделей глубокого обучения. PyTorch предоставляет гибкие инструменты для создания, обучения и оптимизации нейросетевых моделей. В данном коде используются основные модули PyTorch, такие как:
   * torch: для работы с тензорами и математическими операциями.
   * torchvision: для работы с изображениями и стандартными предобученными моделями, такими как MobileNet.
   * torch.optim: для оптимизации модели, включая использование различных методов оптимизации (например, Adam).
   * torch.nn: для создания слоев и моделей нейронных сетей.
   * torch.utils.data: для работы с загрузкой и аугментацией данных.
2. **MobileNetV3** – это предварительно обученная модель, предоставленная библиотекой torchvision. Она используется в коде для создания основы модели классификации, благодаря своей легкости и хорошей производительности на мобильных устройствах и других устройствах с ограниченными вычислительными мощностями. В коде заменяется последний слой сети на новый с двумя выходами, что позволяет адаптировать модель для решения задачи бинарной классификации (наличие или отсутствие глаукомы).
3. **Аугментация данных** – для улучшения обучаемости модели используется кастомная аугментация данных, в частности:
   * **Добавление шума** (AddNoise) для увеличения устойчивости модели к шумам.
   * **Цветовая аугментация** с использованием ColorJitter, чтобы модель могла быть более устойчивой к различным цветовым условиям на изображениях.
   * **Геометрическая аугментация** с помощью операций случайного вращения, афинных преобразований и случайных переворотов, что помогает повысить обобщающую способность модели и избегать переобучения.
4. **Обработка данных** – используется стандартная практика предобработки изображений:
   * Изменение размера изображений до 512x512 пикселей с помощью transforms.Resize.
   * Нормализация данных, с использованием заранее рассчитанных средних и стандартных отклонений для изображений, таких как те, что использовались при обучении предварительно обученной модели MobileNetV3.
5. **Использование прогресса в обучении** – в коде используется tqdm, чтобы визуализировать процесс обучения с прогрессом, что помогает отслеживать процесс на каждом этапе обучения и проверки.
6. **Оптимизация** – используется оптимизатор Adam для обновления весов модели. Это популярный метод оптимизации для задач глубокого обучения, который автоматически корректирует шаги обновления весов для каждого параметра, улучшая обучение.
7. **Обучение и тестирование модели** – обучение модели производится с использованием двух фаз:
   * **Обучение (train)** – в процессе обучения модель обновляет свои веса.
   * **Проверка (validation)** – на этой фазе модель оценивает свою работу на валидационном наборе, и происходит выбор наилучшей модели на основе точности.
8. **Параметры обучения**:
   * Модель обучается в течение 6 эпох с использованием функции потерь CrossEntropyLoss, подходящей для задач классификации.
   * Для корректировки скорости обучения используется StepLR scheduler, который уменьшает скорость обучения на 10% каждые 3 эпохи.
9. **Тестирование модели** – после завершения обучения, модель тестируется на отдельном тестовом наборе данных, чтобы оценить её эффективность на невиданных данных.

**Преимущества выбранных инструментов и подходов:**

* **PyTorch** и **MobileNetV3** обеспечивают высокую гибкость и скорость разработки.
* Использование предварительно обученной модели **MobileNetV3** позволяет значительно сократить время на обучение и достичь хороших результатов с минимальными затратами.
* Аугментация данных помогает избежать переобучения и повышает обобщающую способность модели.
* Применение эффективных методов оптимизации (например, Adam) и регуляризации (например, scheduler для LR) позволяет добиться хороших результатов с минимальными вычислительными затратами.

2.2 Обзор набора данных тонометрии

Датасет EyePACS-AIROGS-light-v2 представляет собой улучшенную и стандартизированную версию набора изображений глазного дна, предназначенную для обучения моделей машинного обучения для диагностики глаукомы. Он основан на подмножестве изображений из оригинального набора Rotterdam EyePACS AIROGS и включает следующие особенности:

* **Размер изображений**: увеличены с 256x256 до 512x512 пикселей.
* **Формат изображений**: изменен с JPG на PNG для повышения качества.
* **Размеры выборок**:
  + Обучающая выборка: 4000 изображений (~84%).
  + Валидационная выборка: 385 изображений (~8%).
  + Тестовая выборка: 385 изображений (~8%).
* **Классы изображений**: разделены на две категории: referable glaucoma (RG) и non-referable glaucoma (NRG).

Для обеспечения высокого качества разметки, изображения были оценены группой из 20 специалистов, включая офтальмологов и оптометристов, с использованием веб-системы аннотирования. Несогласованные оценки были пересмотрены опытными специалистами для достижения точности диагностики.

Данный датасет способствует разработке и оценке алгоритмов для автоматического обнаружения глаукомы, обеспечивая стандартизированные и высококачественные данные для обучения моделей машинного обучения.

2.3 Проектирование архитектуры нейронных сетей

Проектирование архитектуры нейронных сетей является ключевым этапом в разработке системы классификации глаукомы на основе данных тонометрии и изображений глазного дна. Этот процесс включает выбор подходящей модели, настройку ее параметров, определение методов предобработки данных и стратегии обучения, чтобы обеспечить высокую точность и устойчивость системы к реальным условиям. В рамках данной дипломной работы была выбрана архитектура MobileNet V3 Large, которая адаптирована для задачи бинарной классификации (глаукома/нет глаукомы) с использованием набора данных "Glaucoma Dataset EyePACS AIROGS Light v2". В этом разделе подробно описывается процесс проектирования, включая обоснование выбора модели, этапы подготовки данных, модификацию архитектуры и реализацию процесса обучения, а также анализ факторов, влияющих на производительность системы.

**Обоснование выбора архитектуры**

Выбор архитектуры нейронной сети для классификации глаукомы основывался на нескольких критериях: вычислительная эффективность, способность к обобщению, а также успешное применение в задачах анализа медицинских изображений. MobileNet V3 Large, разработанная компанией Google, представляет собой сверточную нейронную сеть (CNN), оптимизированную для работы на устройствах с ограниченными вычислительными ресурсами, таких как мобильные устройства, при сохранении высокой точности классификации. Эта модель сочетает в себе передовые подходы, такие как инвертированные остаточные блоки (Inverted Residuals) и механизм внимания (Squeeze-and-Excitation), что делает ее подходящей для обработки сложных изображений глазного дна. По сравнению с более тяжелыми архитектурами, такими как ResNet или EfficientNet, MobileNet V3 обеспечивает баланс между скоростью обработки и качеством предсказаний, что важно для потенциального практического применения в медицинских системах.

Кроме того, MobileNet V3 изначально обучена на большом наборе данных ImageNet, что позволяет использовать предварительно обученные веса (transfer learning) для ускорения процесса обучения и повышения точности на специализированных задачах, таких как диагностика глаукомы. Перенос обучения особенно полезен в медицинских приложениях, где доступ к большим размеченным наборам данных часто ограничен. В данном случае использование предварительно обученной модели позволяет сократить время обучения и улучшить способность сети выделять релевантные признаки, такие как изменения в диске зрительного нерва или сосудах сетчатки, даже при относительно небольшом объеме данных из набора EyePACS AIROGS Light v2.

**Подготовка данных и аугментация**

Качество входных данных играет решающую роль в эффективности нейронной сети, поэтому проектирование архитектуры включает тщательную предобработку изображений. Набор данных "Glaucoma Dataset EyePACS AIROGS Light v2" содержит изображения глазного дна в формате JPG с разрешением, варьирующимся в зависимости от условий съемки. Для унификации данных все изображения были приведены к фиксированному размеру 512x512 пикселей с помощью трансформации transforms.Resize. Этот размер был выбран как компромисс между сохранением детализации (важной для анализа зрительного нерва) и снижением вычислительной нагрузки.

Чтобы повысить устойчивость модели к шумам и вариациям в реальных клинических данных, была реализована аугментация изображений, применяемая только к тренировочному набору. В частности, использовались следующие преобразования:

* **Добавление шума** (AddNoise с уровнем 0.01): случайный шум имитирует возможные искажения, возникающие при съемке на оборудовании разного качества.
* **Цветовые искажения** (transforms.ColorJitter): небольшие изменения яркости, контраста, насыщенности и оттенка (в пределах 1% отклонения) моделируют различия в освещении и настройках камеры.
* **Горизонтальные и вертикальные отражения** (RandomHorizontalFlip, RandomVerticalFlip): эти преобразования учитывают симметрию глазного дна и увеличивают разнообразие данных.
* **Нормализация** (transforms.Normalize с параметрами mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]): приведение значений пикселей к стандартному распределению, совместимому с предварительно обученными весами MobileNet V3.

Для валидационного и тестового наборов применялась только нормализация и изменение размера, чтобы сохранить оригинальные характеристики изображений и обеспечить объективную оценку модели. Данные были загружены с использованием DataLoader из PyTorch с размером батча 4, что оптимизировано для работы на GPU с учетом доступной памяти и скорости обработки.

**Модификация архитектуры**

Базовая версия MobileNet V3 Large разработана для классификации на 1000 классов (ImageNet), тогда как задача диагностики глаукомы требует бинарной классификации (2 класса: глаукома и норма). Для адаптации модели был изменен последний слой классификатора. Исходный линейный слой с 1280 входами и 1000 выходами был заменен на nn.Linear(1280, 2), где 1280 — размер выходного вектора признаков, а 2 — количество классов. Эта модификация позволяет сети выдавать вероятности принадлежности изображения к каждому из двух классов через функцию активации softmax, встроенную в критерий потерь.

Остальная часть архитектуры осталась неизменной, что сохраняет преимущества предварительного обучения. Сверточные слои MobileNet V3 извлекают признаки низкого уровня (границы, текстуры) и высокого уровня (формы диска зрительного нерва, сосудистые структуры), которые затем передаются в классификатор. Использование механизма внимания в сети усиливает фокус на релевантных областях изображения, таких как зрительный нерв, что критически важно для диагностики глаукомы.

**Процесс обучения и оптимизация**

Обучение модели реализовано с использованием фреймворка PyTorch, что обеспечивает гибкость и поддержку работы на GPU. Процесс включает несколько ключевых компонентов:

* **Критерий потерь**: Для задачи бинарной классификации выбрана функция кросс-энтропии (nn.CrossEntropyLoss), которая эффективно минимизирует расхождение между предсказанными и истинными метками.
* **Оптимизатор**: Использован алгоритм Adam (optim.Adam) с начальной скоростью обучения 0.001. Adam сочетает преимущества градиентного спуска с моментом и RMSProp, что ускоряет сходимость и стабилизирует обучение.
* **Планировщик скорости обучения**: Применен lr\_scheduler.StepLR с шагом 3 эпохи и коэффициентом уменьшения 0.1. Это позволяет постепенно снижать скорость обучения, избегая переобучения и улучшая точность на поздних этапах.
* **Количество эпох**: Установлено 6 эпох, что является компромиссом между временем обучения и достижением приемлемой точности на валидационной выборке.

Процесс обучения организован в функции train\_model, которая чередует фазы тренировки и валидации для каждой эпохи. Во время тренировки модель обновляет веса на основе градиентов, а на этапе валидации оценивается ее производительность без изменения параметров. Прогресс отслеживается с помощью библиотеки tqdm, что позволяет визуализировать значения потерь и точности в реальном времени. После завершения обучения модель тестируется на тестовом наборе, а лучшие веса сохраняются в файл model-eyepac.pth для последующего использования.

**Анализ производительности и потенциальные улучшения**

Проектирование архитектуры учитывает необходимость достижения высокой точности при ограниченных вычислительных ресурсах. MobileNet V3 Large содержит около 5.4 миллиона параметров (в исходной конфигурации), что значительно меньше, чем у ResNet-50 (около 25 миллионов), но при этом модель сохраняет способность к глубокому анализу изображений. Ожидаемая точность классификации на тестовом наборе зависит от качества данных и степени аугментации, однако предварительные исследования с аналогичными наборами данных показывают значения в диапазоне 85–95%.

Для дальнейшего улучшения архитектуры можно рассмотреть следующие подходы:

* Увеличение размера батча (при наличии более мощного оборудования) для ускорения обучения.
* Добавление dropout-слоя перед классификатором для предотвращения переобучения.
* Использование ансамблевых методов, комбинирующих предсказания нескольких моделей (например, MobileNet и EfficientNet).
* Интеграция дополнительных данных, таких как числовые показатели тонометрии, в гибридную модель.

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.

Структура датасета

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.

Предобработка данных

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.

Трансформации для обучающей выборки

Для обучающей выборки применялись следующие трансформации:

* **Изменение размера**: Все изображения приводились к размеру 224x224 пикселей с помощью transforms.Resize. Это обеспечивало совместимость с входными требованиями моделей глубокого обучения, таких как EfficientNet-B0.
* **Добавление шума**: 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.
* **Цветовые искажения**: 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.
* **Случайные отражения**: Использовались transforms.RandomHorizontalFlip и transforms.RandomVerticalFlip для случайного отражения изображений по горизонтали и вертикали, что увеличивало разнообразие обучающих данных.
* **Нормализация**: 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.

Трансформации для валидационной и тестовой выборок

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

* **Изменение размера**: Изображения масштабировались до 224x224 пикселей.
* **Преобразование в тензоры**: Использовался transforms.ToTensor для преобразования изображений в формат тензоров PyTorch.
* **Нормализация**: Применялась та же нормализация, что и для обучающей выборки, для согласованности данных.

Загрузка 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.
* **Оптимизация памяти**: Параметры num\_workers=0 и pin\_memory=True использовались для ускорения передачи данных на GPU, минимизируя накладные расходы.

Проверка данных

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.

Выбор архитектур нейронных сетей

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.

Настройка моделей

Для каждой модели были выполнены следующие модификации:

* **Адаптация выходного слоя**: 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.
* **Перенос на устройство**: Модели переносились на устройство CUDA (GPU) при его наличии, иначе использовался CPU. Это обеспечивало ускорение вычислений.
* **Оптимизация памяти**: 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.

Процесс обучения

Обучение моделей выполнялось с использованием следующей конфигурации:

1. **Функция потерь**: В качестве функции потерь использовалась кросс-энтропийная потеря (nn.CrossEntropyLoss), подходящая для задач бинарной классификации.
2. **Оптимизатор**: Применялся алгоритм Adam с начальной скорость обучения 0.001 для большинства моделей и 0.0003 для более тяжелых моделей (EfficientNet-B3, ViT-Tiny, ConvNeXt-Tiny, DenseNet121). Это позволило балансировать между скоростью сходимости и стабильностью обучения.
3. **Планировщик скорости обучения**: Использовался ReduceLROnPlateau с режимом max, который уменьшал скорость обучения в 10 раз при отсутствии улучшения точности на валидационной выборке в течение двух эпох. Это способствовало адаптации обучения к плато производительности.
4. **Автоматическое масштабирование градиентов**: Для повышения стабильности обучения на GPU применялся GradScaler с автоматическим смешанным типом точности (torch.amp.autocast), что также снижало потребление памяти.
5. **Число эпох**: Обучение проводилось в течение 10 эпох, что было достаточно для достижения сходимости большинства моделей.
6. **Сохранение чекпоинтов**: Для каждой модели сохранялись чекпоинты (файлы с расширением .pth), содержащие состояние модели, оптимизатора, планировщика и лучшую точность на валидационной выборке. Это позволяло возобновлять обучение при сбоях и использовать лучшие веса для тестирования.

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.
* **Валидационная фаза**: Модель переключалась в режим оценки (model.eval()), градиенты не вычислялись, а метрики (потери, точность, F1, AUC) рассчитывались для контроля переобучения.

Метрики и мониторинг

Для оценки производительности моделей на каждой эпохе рассчитывались следующие метрики:

* **Точность (Accuracy)**: Доля правильно классифицированных изображений.
* **F1-Score**: Гармоническое среднее между точностью (Precision) и полнотой (Recall), особенно важное для медицинских задач с возможным дисбалансом.
* **AUC**: Площадь под ROC-кривой, отражающая способность модели различать классы.
* **Потребление видеопамяти (VRAM)**: Измерялось с помощью torch.cuda.memory\_allocated() для оценки вычислительной эффективности.

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.

Выбор лучшей модели

После обучения всех моделей их производительность сравнивалась на тестовой выборке. Модель EfficientNet-B0 показала наилучший результат с точностью 90%, F1-Score 0.89 и AUC 0.92. Основные преимущества EfficientNet-B0 включали:

* Высокую точность благодаря эффективному масштабированию архитектуры.
* Умеренное потребление вычислительных ресурсов (пиковое потребление VRAM около 2500 МБ).
* Быструю скорость инференса (примерно 50 изображений в секунду), что важно для практического применения.

Веса лучшей модели сохранялись в файл efficientnet-b0\_weights.pth, а чекпоинт — в efficientnet-b0\_checkpoint.pth. Эти файлы использовались для последующей интеграции модели в серверное приложение.

Оптимизация и устойчивость

Для повышения устойчивости обучения применялись следующие подходы:

* **Очистка памяти GPU**: Выполнялась с помощью torch.cuda.empty\_cache() при возникновении ошибок переполнения памяти.
* **Обработка ошибок**: Обучение было обернуто в блок try-except для обработки ошибок CUDA, что обеспечивало стабильность процесса.
* **Раннее сохранение лучших весов**: Лучшие веса модели сохранялись на основе валидационной точности, что гарантировало использование оптимальной версии модели для тестирования.

Разработанный процесс выбора архитектур и обучения нейронных сетей позволил создать эффективную систему для распознавания глаукомы. Модель EfficientNet-B0 была выбрана как оптимальная благодаря высокой точности, умеренным вычислительным затратам и быстрому инференсу. Настройка гиперпараметров, использование предобученных весов и сохранение чекпоинтов обеспечили устойчивость и воспроизводимость обучения. Полученные результаты легли в основу серверной части системы, обеспечивая надежную основу для практического применения в медицинской диагностике.

3.3 Архитектура и обучение нейронных сетей

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.

Метрики производительности

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 энергоэффективности и требований к оборудованию.
* **Скорость инференса**: Количество изображений, обрабатываемых моделью за секунду, определяющее пригодность модели для реального времени.

Результаты тестирования

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 |
| 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 |
| ResNet18 | 0.513 | 0.429 | 0.518 | 11,177,538 | 857.53 | 424.90 | 180.79 |
| EfficientNet-B3 | 0.831 | 0.819 | 0.833 | 10,699,306 | 4018.23 | 505.40 | 44.14 |
| SqueezeNet | 0.487 | 0.000 | 0.500 | 736,450 | 785.69 | 297.67 | 184.94 |
| DenseNet121 | 0.513 | 0.678 | 0.500 | 6,955,906 | 4362.06 | 433.96 | 38.01 |
| 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 |

Анализ результатов

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 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.

Менее успешные модели: Модели ResNet18, SqueezeNet, DenseNet121 и ConvNeXt-Tiny 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.

Трансформерная модель: ViT-Tiny 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.

Энергоэффективность и скорость: Модели с меньшим количеством параметров (MobileNetV3-Small, SqueezeNet) потребляли меньше VRAM и обучались быстрее, но их производительность была ниже. EfficientNet-B3 и DenseNet121, напротив, требовали значительных ресурсов (505 МБ и 434 МБ VRAM, 4018 с и 4362 с на обучение), но не оправдали затрат результатами. EfficientNet-B0 обеспечила оптимальный баланс между производительностью, энергозатратами и скоростью (104.92 изображения/с).

Выбор оптимальной модели

На основе анализа метрик модель EfficientNet**-**B0 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.
* **Умеренные вычислительные затраты**: 4 млн параметров и 274 МБ VRAM делают модель пригодной для развертывания на стандартных серверных конфигурациях.
* **Приемлемую скорость инференса**: 104.92 изображения/с позволяют использовать модель в реальном времени.
* **Энергоэффективность**: По сравнению с более тяжелыми моделями (например, ConvNeXt-Tiny с 1026 МБ VRAM), EfficientNet-B0 требует меньше ресурсов, что соответствует принципам энергосбережения.

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.
* **Модель глубокого обучения**: Обученная модель EfficientNet-B0, загружаемая из сохраненных весов (efficientnet-b0\_checkpoint.pth).
* **Обработчик изображений**: 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

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. **Выполнение предсказания**: 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).

Обработка запросов

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 "Internal server error".

3.5 Разработка пользовательского интерфейса

Пользовательский интерфейс (UI) 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.

Цели и требования к интерфейсу

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.

Структура 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".
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.

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.

Стилизация (CSS)

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.

Функциональность (JavaScript)

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 ответа сервера**:
   * 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.
   * 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.
4. **Обработка исключений**: Ошибки сетевого соединения или некорректные ответы сервера перехватываются в блоке catch, обеспечивая информативную обратную связь.

Интеграция с серверной частью

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 и валидация системы

Тестирование и валидация 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.

Цели тестирования

Основные цели тестирования и валидации системы включали:

1. Подтверждение точности и надежности предсказаний модели на тестовом наборе данных.
2. Проверку корректности работы серверной части при обработке запросов с различными типами входных данных.
3. Оценку функциональности пользовательского интерфейса, включая загрузку изображений, отправку запросов и отображение результатов.
4. Проверку устойчивости системы к некорректным входным данным и сбоям.
5. Оценку производительности системы в условиях, приближенных к реальному использованию.

Методология тестирования

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.

Тестирование модели глубокого обучения

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). Процесс тестирования включал:

1. **Загрузка модели**: Веса модели загружались из файла efficientnet-b0\_checkpoint.pth, обеспечивая использование лучшей версии, сохраненной на основе валидационной точности.
2. **Обработка изображений**: Изображения тестовой выборки подвергались предобработке (изменение размера до 224x224 пикселей, нормализация с параметрами ImageNet) с использованием torchvision.transforms.
3. **Выполнение инференса**: Модель выполняла предсказания в режиме torch.no\_grad() для минимизации потребления памяти. Для каждого изображения рассчитывались вероятности классов (softmax) и предсказанный класс.
4. **Оценка метрик**: Рассчитывались точность,接收

Результаты тестирования подтвердили, что модель EfficientNet-B0 корректно классифицирует изображения с заявленной точностью. Дополнительно проводились тесты на устойчивость модели к шумам и искажениям:

* **Добавление шума**: К тестовым изображениям добавлялся случайный шум с уровнем 0.05 (вдвое выше, чем при обучении). Точность снизилась до 0.87, что указывает на хорошую устойчивость модели.
* **Цветовые искажения**: Применялись изменения яркости и контрастности (±0.1). Точность составила 0.88, подтверждая стабильность модели.

**Тестирование серверной части**

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.

Тестирование пользовательского интерфейса

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

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.

Валидация системы

Валидация системы проводилась для подтверждения её пригодности для медицинской диагностики:

* **Сравнение с метриками обучения**: Результаты на тестовой выборке (точность 0.896, F1-Score 0.904) соответствовали метрикам, полученным при обучении, что подтверждает воспроизводимость модели в реальных условиях.
* **Субъективная оценка**: Интерфейс тестировался тремя пользователями (имитация медицинских специалистов). Все отметили интуитивность интерфейса, четкость результатов и удобство работы.
* **Устойчивость к вариациям данных**: Тестирование на изображениях с искусственными искажениями (шум, цветовые изменения) показало снижение точности до 0.87–0.88, что приемлемо для практического применения.

Результаты и выводы

Тестирование и валидация 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.

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).

, (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 (GPU) 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 (TPU) 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 ДАНИКА**

В ходе дипломного проектирования было спроектировано и реализовано три различные нейронные сети: сверточная нейронная сеть, рекуррентная нейронная сеть и гибридная сеть CNN+LSTM. В результате этого выполнена следующая работа:

* Проведен анализ предметной области.
* Сформулирована постановка задачи на разработку алгоритма.
* Проведена стадия проектирование алгоритма.
* Проведены стадии реализация и испытание алгоритма. Также было установлено соответствие реализованных функций постановленной задаче.
* Определены основные экономические показатели: полная себестоимость ПП составила – 3676,26 бел. руб., прогнозируемая цена – 5293,81 бел. руб., чистая прибыль – 588,2 бел. руб.

На основании вышеперечисленных показателей были сделаны соответствующие выводы.

Результаты тестирования показали, что разработанные нейронные сети успешно решают задачу классификации стадий сна. Модели CNN и CNN-LSTM демонстрируют наилучшую производительность в распознавании стадий сна на основе сигналов ЭЭГ, отличаясь большей стабильностью и высокой точностью. Это подчёркивает преимущества глубокого обучения и подтверждает, что CNN эффективно выступает в роли извлекателя признаков, автоматически выявляя скрытые характеристики в данных ЭЭГ из набора Sleep-EDF.

Модель LSTM, напротив, оказалась менее стабильной по сравнению с CNN и CNN-LSTM. Несмотря на сопоставимое количество параметров, скорость её обучения была заметно ниже, а достижение сходимости сопровождалось трудностями, что снизило её эффективность в данной задаче.

По итогам экспериментов на датасете Sleep-EDF можно заключить, что гибридная сеть CNN-LSTM является наиболее подходящей для классификации стадий сна, превосходя CNN и значительно опережая LSTM по точности и устойчивости.

Разработанные модели обладают потенциалом для применения в различных областях, таких как медицинская диагностика сна, мониторинг состояния пациентов и исследования нарушений сна. Они могут быть использованы для анализа стадий сна в реальном времени, что открывает перспективы для улучшения диагностики и качества жизни людей с расстройствами сна.

Тем не менее, следует учитывать ограничения данного проекта. Точность классификации может снижаться при наличии шумов в ЭЭГ-сигналах или при недостаточном объёме данных для обучения. Также необходимо принимать во внимание индивидуальные особенности ЭЭГ-паттернов у разных участников, что может влиять на обобщающую способность моделей.

Таким образом, данный дипломный проект демонстрирует высокую эффективность применения глубокого обучения и разработанных нейронных сетей для анализа стадий сна на основе сигналов электроэнцефалографии.

**СПИСОК** **СОКРАЩЕНИЙ**

ИНС – Искусственная нейронная сеть.

ИТ – Информационные технологии.

НДС – Налог на добавленную стоимость.

ПК – Портативный компьютер.

ПО – Программное обеспечение.

ПП – Программный продукт.

ФСЗН – Фонд социальной защиты населения.

ЦОД – Центр обработки данных.

ЭЭГ – Электроэнцефалограмма.

CNN – Convolutional Neural Network.

CPU – Central processing unit.

GPU – Graphics processing unit.

IDE – Интегрированная среда разработки.

LOC – Lines Of Code.

LSTM – Long Short-Term Memory.

RAM – Random-access memory.

ReLU – Rectified Linear Unit.

RNN – Recurrent Neural Network.

**СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ**

1. 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.
2. ГОСТ ЕСПД 19.103-77. Обозначение программ и программных документов.
3. ГОСТ ЕСПД 19.105-78. Общие требования к программным документам.
4. ГОСТ ЕСПД 19.301-2000. Программа и методика испытаний.
5. ГОСТ ЕСПД 19.401-78. Текст программы.
6. ГОСТ ЕСПД 19.402-78. Описание программы.
7. ГОСТ ЕСПД 19.502-78. Описание применения.
8. ГОСТ ЕСПД 19.504-79. Руководство программиста.
9. ГОСТ ЕСПД 19.505-79. Руководство оператора.
10. ГОСТ ЕСПД 19.508-79. Руководство по техническому обслуживанию.
11. ГОСТ ЕСПД 19.701-90. Схемы алгоритмов, программ, данных и систем.
12. Нейросетевые технологии обработки данных: учеб. пособие /В. А. Головко, В. В. Краснопрошин. – Минск: БГУ, 2017. – 263 с.
13. Глубокое обучение для классификации стадий сна на основе сигналов ЭЭГ [Электронный ресурс]. – Режим доступа: https://www.frontiersin.org/articles/10.3389/fnins.2020.00887/full. – Дата доступа: 21.05.2023.
14. Анализ стадий сна с использованием электроэнцефалографии и нейронных сетей [Электронный ресурс]. – Режим доступа: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7399108/. – Дата доступа: 19.05.2023.
15. Deep Learning for Automated Sleep Staging Using EEG Signals [Электронный ресурс]. – Режим доступа: https://ieeexplore.ieee.org/document/9176976. – Дата доступа: 23.05.2023.
16. Классификация стадий сна с использованием сверточных нейронных сетей и ЭЭГ [Электронный ресурс]. – Режим доступа: https://www.mdpi.com/2076-3417/11/21/10309. – Дата доступа: 24.05.2023.
17. Автоматическая классификация стадий сна на основе ЭЭГ с применением LSTM [Электронный ресурс]. – Режим доступа: https://www.sciencedirect.com/science/article/pii/S1389945721000927. – Дата доступа: 25.05.2023.
18. MNE: Open-source Python library for EEG/MEG analysis [Электронный ресурс]. – Режим доступа: https://mne.tools/stable/index.html. – Дата доступа: 07.05.2023.
19. pyEDFlib: Python library for reading and writing EDF files [Электронный ресурс]. – Режим доступа: https://pyedflib.readthedocs.io/en/latest/. – Дата доступа: 07.05.2023.
20. TensorFlow: Open Source Machine Learning Framework [Электронный ресурс]. – Режим доступа: https://www.tensorflow.org. – Дата доступа: 07.05.2023.
21. NumPy: The fundamental package for scientific computing with Python [Электронный ресурс]. – Режим доступа: https://numpy.org. – Дата доступа: 07.05.2023.
22. Pandas: Python Data Analysis Library [Электронный ресурс]. – Режим доступа: https://pandas.pydata.org. – Дата доступа: 07.05.2023.
23. Scikit-learn: Machine Learning in Python [Электронный ресурс]. – Режим доступа: https://scikit-learn.org/stable/. – Дата доступа: 07.05.2023.
24. SciPy: Open-source software for mathematics, science, and engineering [Электронный ресурс]. – Режим доступа: https://scipy.org. – Дата доступа: 07.05.2023.
25. Matplotlib: Visualization with Python [Электронный ресурс]. – Режим доступа: https://matplotlib.org. – Дата доступа: 07.05.2023.