СОДЕРЖАНИЕ

[Введение 2](#_Toc161612381)

[1. Постановка задачи 7](#_Toc161612382)

[1.1. Введение 7](#_Toc161612383)

[1.2. Цель и задачи выпускной квалификационной работы 7](#_Toc161612384)

[1.3. Предметная область 8](#_Toc161612385)

[1.4. Актуальность 8](#_Toc161612386)

[1.5. Заключение 9](#_Toc161612387)

[2. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 11](#_Toc161612388)

[2.1. Введение 11](#_Toc161612389)

[2.2. Описание архитектуры проекта 11](#_Toc161612390)

[2.3. Архитектура сервера 12](#_Toc161612391)

[2.3.1. Выбор языка 12](#_Toc161612392)

[2.3.2. Выбор фреймворка 14](#_Toc161612393)

[2.3.3. Выбор базы данных 15](#_Toc161612394)

[2.3.4. Резюме 15](#_Toc161612395)

[2.4. Архитектура интеллектуального сервиса 15](#_Toc161612396)

[2.4.1. Выбор языка 15](#_Toc161612397)

[2.4.2. Выбор способа обучения 22](#_Toc161612398)

[2.4.3. Резюме 24](#_Toc161612399)

[2.4.4. Выбор фреймворка 24](#_Toc161612400)

[2.4.5. Резюме 30](#_Toc161612401)

[2.5. Архитектура клиентской части 30](#_Toc161612402)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 32](#_Toc161612403)

Введение

Медицина – одна из важнейших сфер деятельности человека. На сегодняшний день в России существует набирающая обороты тенденция по цифровизации многих медицинских процессов, которые раньше делались вручную и требовали большой бумажной волокиты, а также приводили к ошибкам и неточностям, что в итоге прямым образом сказывалось на качестве лечения и постановке диагнозов.

Наметившиеся тренды по развитию медицины в нашем государстве появились сравнительно недавно, а обрели по-настоящему значительную силу буквально несколько лет назад. Так, в распоряжении правительства РФ от 29 декабря 2021 года № 3980-р утверждено стратегическое направление в области цифровой трансформации здравоохранения. [1]. Основными тезисами принятого распоряжения являются:

* достижение "цифровой зрелости" в здравоохранении
* увеличение доли массовых социально значимых услуг, доступных в электронном виде
* увеличение доли отечественные решения в сфере информационных технологий
* формирование электронных баз знаний
* переход к электронному документообороту

Данному указу предшествовали другие, более общие указы и постановления правительства, направленные на развитие и цифровизации сферы здравоохранения в России. Такие, как:

* указ Президента Российской Федерации от 21 июля 2020 г. N 474 "О национальных целях развития Российской Федерации на период до 2030 года"; [1]
* указ Президента Российской Федерации от 06 июня 2019 г. N 254 "О Стратегии развития здравоохранения в Российской Федерации на период до 2025 года"; [1]
* указ Президента Российской Федерации от 7 мая 2018 г. N 204 "О национальных целях и стратегических задачах развития Российской Федерации на период до 2024 года"; [1]
* постановление Правительства Российской Федерации от 26 декабря 2017 г. N 1640 "Об утверждении государственной программы Российской Федерации "Развитие здравоохранения". [1]

Это доказывает заинтересованность государства в развитии и улучшения качества услуг в области медицины, особенно в последнее десятилетие.

Внедрение информационных систем взамен существующих бумажных и повсеместная цифровизация в медицине даёт неоспоримые преимущества по сравнению с традиционными и двигает отрасль вперёд.

Под цифровизацией в широком смысле понимается процесс социально-экономической трансформации, привносящий массовое внедрение и освоение цифровых технологий. Цифровизация – очень мощный инструмент повышения эффективности и автоматизации процессов в совершенно разных областях деятельности. Она способна качественно и многогранно улучшить многие сферы жизни общества.

Однако недостаточно просто цифровизировать привычный бумажный процесс, переведя его в компьютер. Вызовы времени и инициативы государства ставят требование привнести новые возможности и внедрить технологии, недоступные прежде.

В развитии самых актуальных новшеств в медицине заинтересованы все: пациенты (как платных, так и государственных клиник), получающих более качественные услуги за меньшее время; работники клиник и больниц, результативность деятельности которых возрастает; государство, в интересы которого входит улучшение благосостояния людей и обретение технологического суверенитета.

В медицине для цифровизации процессов внедряются медицинские информационные системы (МИС), а также их вариации, предназначенные для конкретных направлений медицины, например лабораторные информационные системы (ЛИС).

МИС – это комплекс программных и аппаратных средств, предназначенных для организации и управления медицинскими данными и процессами. Они в цифровом виде объединяют электронные карты пациентов, хранят и обрабатывают данные медицинских исследований и мониторинга состояния пациентов, работают с оборудованием, автоматизируют документооборот и так далее.

Первые МИС были достаточно простыми. Они решали только рутинные и трудоемкие операции, такие как:

* регистрация и учет пациентов
* финансовые расчеты
* записи на прием

Более современные системы автоматизируют процесс управления клиникой целиком, берут на себя основную часть нагрузки по ведению записей, карт пациентов, составлению отчётов и взаимодействию с другими подобными информационными системами.

На данный момент разработкой МИС занимаются многие компании как за рубежом, так и в России. Вследствие обширности направлений в сфере здравоохранения таких систем должно быть много. МИС должны учитывать особенности того или иного медицинского процесса, учитывать всевозможные факторы и ситуации, когда что-то идет не так. Они должны реально ускорять работу по сравнению с традиционными методами и помогать людям, которые пользуются ей.

Современным МИС недостаточно просто автоматизировать бумажные процессы. Потеряли актуальность даже те медицинские информационные системы, которые берут под свой контроль все системы и процессы медицинской деятельности. На сегодняшний день актуальны решения, предлагающие инновационные технологии, способные кратно увеличить качество и эффективность деятельности, которую они автоматизируют, такие как:

* нейронные сети и машинное обучение
* облачные вычисления
* большие данные
* телемедицина

Все эти технологии, безусловно, являются передовыми. Отправной точкой для развития было принято решение выбрать нейронные сети и машинное обучение. Поставлена задача разработать интеллектуального компонента для одной из МИС, разрабатываемой на текущий момент по месту работы автора, задачей которого является помощь лаборантам в идентификация микроорганизмов, посеянных и сформировавшихся в чашке Петри.

На момент написания автор трудоустроен в организации ООО «НОВОЛАБСИСТЕМ» на должности java-разработчика. В обязанности автора входит написание и сопровождение серверной части медицинской информационной системы для микробиологии.

Чашка Петри – это пластиковая ёмкость, в которую нанесён слой питательной среды. В этой среде могут развиваться все микроорганизмы, которые там окажутся. Нужна она для того, чтобы увеличить количество колоний, взятых из некоторой биологической среды, поскольку на самом образце-источнике биоматериала с микроорганизмами недостаточно для их точной идентификации.

Для распознавания колоний в чашке Петри выпускается дорогое медицинское оборудование, в основном импортное, с некоторого момента переставшее ввозиться на территорию нашей страны. Для приобретения и работы с ним требуются внушительные средства, составляющие не только стоимость аппарата, но и труд специалиста, занимающегося обслуживанием и поддержкой этого устройства. Далеко не все лаборатории, особенно в провинциальных городах, могут себе это позволить.

В качестве альтернативы «аппаратному» подходу предлагается программный механизм, идентифицирующей микроорганизмы по одной лишь фотографии, не требующий ни больших вложений, ни сил на поддержку и обслуживание.

Актуальность данного подхода подтверждается обязательным пунктом в техническом задании от министерства цифрового развития для разрабатываемой системы МИС по месту работу автора. Техническое задание звучит так: «Разработка микробиологической информационной системы для цифровизации деятельности микробиологических лабораторий в условиях импортозамещения, функционирующей на системном программном обеспечении российских производителей и свободном ПО (системном ПО с открытым кодом), с функцией формирования и хранения изображений для последующего распознавания образов колоний микроорганизмов с помощью ИИ» [2].

Таким образом, будет разработан интеллектуальный компонент (интеллектуальная подсистема) для распознавания колоний микроорганизмов в чашке Петри и внедрена в разрабатываемую на данный момент МИС (медицинскую информационную систему).

# Постановка задачи

* 1. Введение

Целью данной главы является обобщения и описание задач данной квалификационной работы, шагов по достижению цели и доказательство её актуальности. В части 1.2 «Цель и задачи выпускной квалификационной работы» описана основная цель квалификационной работы, результатом которой является работоспособный программный продукт, и задачи, которые ставятся на пути к реализации поставленной цели. В части 1.3 «Предметная область» описана предметная область, то есть сфера знаний, в которой будет работать программный продукт. В части 1.4 «Актуальность» подробно описана проблема, существующая в предметной области, которую призвано решить разрабатываемое в данной квалификационной работе программное обеспечение.

* 1. Цель и задачи выпускной квалификационной работы

Главная цель выпускной квалификационной работы – разработать интеллектуальную подсистему для идентификации микроорганизмов в чашке Петри, и встроить её в медицинскую информационную систему, разрабатываемую на данный момент, для последующего внедрения программного обеспечения в реальные лаборатории.

Для достижения поставленной цели выделены следующий этапы:

1. определить задачи, которые продукт будет способен решить.
2. спроектировать концепцию и архитектуру, в какой форме будет разработан и внедрён продукт.
3. определить основную и второстепенную проблемы, которые может решить продукт.
4. разработать программный продукт в соответствии с намеченными задачами.
5. протестировать продукт и убедиться в его полной работоспособности.
   1. Предметная область

Предметной областью, или, другими словами, сферой деятельности, для которой разрабатывается программный продукт, является медицина. Конкретное направление – микробиология.

Микробиология, как наука, занимается изучением мельчайших организмов – микроорганизмов (микробов). Она изучает закономерности их жизни и развития, а также изменения, которые они вызывает в организме животных и людей.

Уклон сделан не в научную, а в практическую сторону. Это является важным уточнением, поскольку характер работы с микроорганизмами в целях научных изысканий сильно отличается от работы с ними в клинической лаборатории. Так, например, научное изучение неизведанных видов требует крайней осторожности. Если что-то пойдет не так, это может иметь очень серьезные последствия, вплоть до вспышки новых заболеваний или даже глобальной эпидемии. Также, работа в научных целях ведется с несколькими чашками Петри в день (количество может варьироваться, но в целом чашек немного, и их можно контролировать). Клиническая лаборатория, напротив, может работать с сотнями и тысячами чашками в день, и биологическую угрозу микроорганизмы в этих чашках, как правило, не представляют.

Приведённые мною в пример факторы дают понимание, почему процессы сильно разнятся несмотря на то, что наука и деятельность одни и те же.

Таким образом, все процессы предметной области, описанные здесь, применимы к деятельности клинических лабораторий.

* 1. Актуальность

Традиционно, для идентификации микроорганизмов в чашке Петри используются масс-спектрометры. Это довольно большие и дорогие устройства, требующие немалых средств на закупку и обслуживание. Не все медицинские организации, особенно государственные и провинциальные, могут себе их позволить. Взамен крупных и затратных аппаратов предлагается программный подход, который будет работать на обычном простом компьютере, в браузере. Лаборатории, которые приобретут МИС (медицинскую информационную систему), получат в её составе механизм, который избавляет от необходимости закупать реальное оборудование, кратно уменьшает время процесса и, если смотреть глобальней, делает медицину доступней для всех людей. Расплатой за программную альтернативу является разве что несколько меньшая точность распознавания микроорганизмов (зависит от многих факторов, колеблется в больших пределах). При правильном использовании, в благоприятных условиях программный подход даёт не менее высокую точность распознавание, как и реальное оборудование.

Ещё одним фактором актуальности данной темы является техническое задание, обязательным пунктом в котором прописана реализация данной подсистемы. Техническое задание подготовлено министерством цифрового развития России для организации, которая является исполнителем проекта, сотрудников которой является автор данной выпускной квалификационной работы.

Актуальность разработки подобных систем обусловлена так же необходимостью обретения страной технологической независимости. В нынешних условиях нет возможности использовать существующие импортные аналоги. Более того, подобных информационных систем может и вовсе не существовать, поскольку Россия является одним из лидеров в IT-сфере в мире, и по многих областям находится на переднем крае, в авангарде.

Данную подсистему действительно ждут медицинские организации, являющиеся пользователями МИС (или планирующие её к приобретению), компонентом которой она является. В этом автор убедился из личных контактов с сотрудником одной из лабораторий.

* 1. Заключение

В части 1.2 «Цель и задачи выпускной квалификационной работы» были подробно разобраны цели, для которых создаётся данное решение, и задачи, которые будет решать этот продукт. Проблемы, ради которых он разрабатывается, действительно остро стоят в нашей стране, и их решение внесёт значительный эффект в развитие медицинской отрасли, а также в общее состояние дел экономики, и увеличит шансы на обретение технологического суверенитета государством.

В части 1.3 «Предметная область» была описана сфера деятельности, в рамках которой будет существовать программный продукт. Медицина является важнейшим институтом в стране, и её развитие принесёт благо для всех граждан.

В части 1.4 «Актуальность» подробно расписаны причины, по которым продукт является актуальным и необходимым для отрасли именно сейчас.

Подводя итог, можно сделать вывод, что тема разработки интеллектуальной подсистемы для распознавания колоний микроорганизмов в чашке Петри актуальна на сегодняшний день и не потеряет своей актуальности в будущем.

# ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

* 1. Введение

В данной главе будут рассмотрены инструменты, технологии и решения, которые возможно применить для решения задачи данной выпускной квалификационной работы. В процессе рассмотрения инструментов будут выявлены их преимущества и недостатки в контексте решаемой задачи, а также рассмотрены их аналоги. По итогу анализа будет определен стек технологий, на котором в дальнейшем будет реализовано приложение.

Обоснованное использование подходящих технологий, а также их актуальность и доступность важны для успешной реализации продукта.

* 1. Описание архитектуры проекта

Архитектура программного проекта – совокупность решений об организации программной системы. Она включает в себя элементы, из которых состоит система, их коммуникации, типы связей, поведение.

Правильная, хорошо продуманная архитектура программного обеспечения целиком и полностью определяет качество работы продукта, его скорость, простоту поддержки. Поэтому так важно заранее тщательно продумать архитектуру программного решения.

Принято решение построить архитектуру системы, которая состоит из трёх компонентов:

* Интеллектуального сервиса с нейронной сетью для идентификации колоний микроорганизмов в чашке Петри по фотографиям
* Сервиса для хранения фотографий чашек с колониями
* Сервиса для хранения результатов идентификаций колоний микроорганизмов
* Клиентской части с визуальным интерфейсом.

Сервис с нейронной сетью и будет тем самым интеллектуальным компонентом в пределах всей системы. Он будет содержать предобученную нейронную сеть и предоставлять наружу интерфейс для работы с ней.

Помимо распознавания колоний микроорганизмов стоит задача, во-первых, где-то хранить изображения, полученные с камер различных устройств, а, во-вторых, сохранять результаты распознавания колоний на них. Для этих целей будут использованы два сервиса соответственно.

Клиентская часть будет предоставлять удобный интерфейс для работы с фотографиями колоний микроорганизмов, их распознавания и анализа сводки лаборантами.

Такая архитектура обеспечит масштабируемость и гибкость системы, позволяя легко добавлять новые функциональные возможности и модифицировать существующие. Введение микросервисной архитектуры также обеспечит высокую отказоустойчивость системы, так как каждый компонент может быть независимо масштабирован и обновлен без прерывания работы всей системы. Также такая архитектура позволит лучше распределять нагрузку между компонентами и обеспечит легкость в мониторинге и управлении каждым отдельным сервисом. Она соответствует принятой в команде концепции разработки системы.

В следующих частях данной главы будут рассмотрены технологии, наиболее подходящие для реализации каждого из компонентов выбранной архитектуры системы.

* 1. Архитектура сервера

Задача встроить интеллектуальную подсистему в существующую и разрабатываемую на данный момент микробиологическую информационную систему подразумевает, что уже существует ряд программных механизмов, составляющих основу этой системы и поддерживающих её работоспособность. Это означает, что сфера выбора технологий заранее ограничена теми решениями, которые уже принято использовать в команде.

* + 1. Выбор языка

Архитектура существующей микробиологической системы, для которой будет внедрён интеллектуальный компонент, использует микросервисную архитектуру, а сервисы написаны на языке java. Команда, занимающаяся разработкой этой системы, пишет все её компоненты на java. В связи с этим рационально использовать тот же подход и тот же язык, на котором пишут другие разработчики этой системы в команде с автором. Это решение обеспечит простоту поддержки нового компонента система другими членами команды. Она будет в духе существующей концепции разработки, что очевидным образом повлияет благоприятно на конечный результат.

Однако компания занимается разработкой не только одной микробиологической системой для работы лабораторий. Создаваемые продукты предназначаются для широкого спектра направлений в здравоохранении. Построен непрерывный процесс как доработки старых решений, давно зарекомендовавших себя в отрасли, так и инновационных, значительно увеличивающих возможности старых систем. В связи с этим внутри организации формируется много новых команд, привносящих с собой новый технологический стек. Одним из таких новшеств стал язык c#.

На c# стали писать решения, имеющие общие компоненты с таковыми, написанными на java. Технологический стек этих двух языков разделился примерно поровну в компании, и теперь имеет смысл рассмотреть возможность реализации серверной части и на нём.

Поскольку автор данной выпускной квалификационной работы примерно в равной степени обладает навыками написания на обоих языках программирования, к этому вопросу следует подойти с особенной внимательностью, потому что сервер является важной частью проекта, и правильность принятого решения по поводу технологического стека может предопределить общую судьбу программного решения в промышленном применении.

Получается, для реализации серверной составляющей системы есть два претендента: java и c#. Они выбраны потому, что в компании, в которой работает автор данной выпускной квалификационной работы, сервисы разрабатывают на них. Далее будут приведены доводы в пользу первого или второго языков программирования.

Таким образом, выбор языка java для серверной части обоснован следующими факторами:

* Простота поддержки другими участниками команды, в которой работает автор проекта
* Простота соединения с другими модулями существующей программной системы.
* Требования импортозамещения – возможность работы на отечественных дистрибутивах операционной системы linux
* Все преимущества языков виртуальных машин, таких как: изоляция от нативной операционной системы, переносимость, масштабируемость, сборщик мусора.
  + 1. Выбор фреймворка

Для реализации веб-сервисов редко используется один только “голый” язык программирования. Люди, занимающиеся одним делом, постоянно сталкиваются с проблемами, который уже имеют решение. Не имеет смысла каждый раз изобретать велосипед. Для эффективного решения задач нужно перенимать опыт других людей. Для этого существуют фреймворки – готовые наборы инструментов для реализации распространённых задач и решения известных проблем.

Самым распространённым фреймворком для разработки веб-сервисов на языке java является Spring Framework – универсальный [фреймворк](https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA) [с открытым исходным кодом](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5). Он обеспечивает решения многих задач, с которыми сталкиваются Java-разработчики и организации, которые хотят создать информационную систему, основанную на платформе Java.

Этот фреймворк предоставляет наиболее широкие возможности для разработки веб-сервисов на языке java. Он также выбран командой, разрабатывающей другие компоненты в пределах общей микробиологической информационной системы.

* + 1. Выбор базы данных

В качестве базы для хранения данных также решено остановиться на решении, которое используется в команде – PostgreSQL. Использование этой реляционной базы обусловлено также фактором импортозамещения, поскольку этот программный продукт включён в перечень допустимых к использованию в отечественных решениях.

* + 1. Резюме

Резюмируя, для реализации серверной части сервиса с бизнес-логикой решено использовать язык программирования виртуальной машины java. Для быстрой и удобной разработки, а также стандартизации с другими сервисами микробиологической информационной системы выбран фреймворк Spring. В качестве базы, используемой для хранения данных в этом сервисе, выбрана PostgreSQL.

* 1. Архитектура интеллектуального сервиса
     1. Выбор языка

Разрабатывать и обучать нейронные сети можно на разных языках. Каждый содержит свои библиотеки и фреймворки для работы с машинным обучением. При выборе языка программирования для разработки нейронных сетей и их обучения, необходимо учитывать несколько ключевых факторов, таких как поддержка библиотек машинного обучения, удобство использования, производительность, сообщество разработчиков и доступность ресурсов для обучения.

Стоит выбор из трёх вариантов – три языка, предоставляющие необходимые библиотеки для работы с данными и нейросетями: java, c# и python. Первые два используются в организации, в которой автор выпускной квалификационной работы разрабатывает программный продукт. Использование java или c# является выгодным, так как другие разработчики внутри организации смогли бы поддерживать решение и тоже принимать участие в его разработке. Язык python является крайне популярным в области обучения нейронных сетей и ИИ. Он предоставляет мощнейшие инструменты для этого и имеет огромное сообщество, которое поддерживает эти инструменты. Если вкратце: java и c# выгодно использовать в компании из-за принятых стандартов, а python является самым популярным языком с самыми мощными для обучения ИИ инструментами. Кроме того, автор работы имеет опыт программирования на всех трёх.

Далее будут представлены преимущества и недостатки инструментов, выбранных претендентами для реализации интеллектуального сервиса с искусственным интеллектом.

Python является одним из наиболее популярных языков программирования в области разработки нейронных сетей и машинного обучения. Следующие факторы показывают, почему он является хорошим выбором для разработки и обучения нейронных сетей:

В первую очередь это, конечно же, широкий перечень библиотек и фреймворков, специально разработанных для работы с нейронными сетями. Наиболее известные из них - TensorFlow, Keras и PyTorch, которые предоставляют разработчикам инструменты для создания и обучения сложных нейронных сетей. Благодаря этому, разработчики могут легко создавать и исследовать различные архитектуры нейросетей, а также выявлять и исправлять ошибки в процессе обучения.

Во-вторых, Python предлагает простой и интуитивно понятный синтаксис программирования, что позволяет лучше сосредоточиться на разработке целевого решения, и не обращать внимание на особенности языка.

Кроме того, python является кроссплатформенным языком программирования. Это следствие того, что язык интерпретируемый, то есть для его работы нужен интерпретатор, который не зависит от платформы. Переносимость – важная особенность python, и она соответствует требованиям технического задания. Интеллектуальный сервис на python легко запустится на отечественном дистрибутиве Linux.

Наконец, python поддерживает большое и очень активное комьюнити людей. Это означает, что решение большого спектра задач, даже редких и уникальных, скорее всего, имеет решение. За помощью всегда можно обратиться к его разработчикам. А документация постоянно обновляется и находится в актуальном состоянии.

Таким образом, выбор интерпретируемого языка Python является хорошим по следующим причинам:

* Кроссплатформенность
* Простой и понятный синтаксис
* Множество библиотек для работы с данными и машинного обучения для решения любых задач
* Большое комьюнити разработчиков

Java является одним из наиболее популярных языков программирования, который широко используется для разработки приложений, включая приложения и системы искусственного интеллекта и нейронных сетей. Существует несколько причин, почему стоит выбрать Java для разработки и обучения нейросети.

Java предлагает широкий спектр библиотек и инструментов для работы с нейронными сетями. Библиотеки, такие как DeepLearning4j, Neuroph и Encog, предоставляют удобные средства для создания и обучения нейронных сетей в среде Java. Они обеспечивают простоту в использовании и гибкость при разработке различных типов нейронных сетей, включая сверточные, рекуррентные и глубокие нейронные сети.

Java поддерживает распределенные вычисления через фреймворки, такие как Apache Spark и Hadoop, что позволяет использовать несколько компьютеров для обучения больших нейронных сетей. Также java поддерживает использование графических процессоров (GPU) для ускорения обучения нейронных сетей, что особенно полезно для больших и сложных моделей.

Java предоставляет богатую библиотеку классов, таких как Java Neural Network Framework (JNNF), Deeplearning4j и Weka, которые упрощают разработку и обучение нейронных сетей. Эти библиотеки предлагают различные алгоритмы обучения, такие как обратное распространение, оптимизатор градиентного спуска и т. д.

Виртуальная машина Java (JVM) оптимизирует код во время выполнения, что приводит к высокой производительности и делает Java подходящим выбором для ресурсоемких задач обучения нейронных сетей.

Таким образом, java – быстрый, стабильный, статически типизированный объектно-ориентированный язык программирования с обширными инструментами для работы с нейронными сетями.

Что касается C# – он имеет обширную и активную экосистему с обширными библиотеками, инструментами и ресурсами, ориентированными на разработку и обучение нейронных сетей, такими как:

* ML.NET: кросс-платформенная библиотека машинного обучения, которая упрощает разработку моделей нейронных сетей.
* TensorFlow.NET: высокоуровневый API для создания и обучения моделей TensorFlow на C#.
* Keras.NET: библиотека для высокоуровневого создания и обучения нейронных сетей с использованием Keras.

В целом, язык c# имеет схожие с java инструменты и предоставляет равные возможности для разработки и обучения нейронных сетей.

При выборе языка также надо полагаться на возможность удобной и эффективной интеграции с другими сервисами (клиентская часть, сервера). При выборе инструмента доступны компромиссы – выбрать можно самый популярный язык с самыми мощными для обучения инструментами – python, а затем интегрировать его в сервер, например, на java, или вывести наружу конечные точки, сделав его самостоятельным сервисом.

Далее будут приведены способы, с помощью которых нейронную сеть на python можно “подружить” с другими языками, на которых будет написан сервер, а также описаны преимущества и недостатки каждого подхода. Результаты исследования позволят сделать выбор в пользу того или иного решения.

1. ONNX (Open Neural Network Exchange) – это библиотека с открытым исходным кодом, которая предоставляет стандартный, совместимый формат для моделей машинного обучения. Она позволяет специалистам по обработке данных и разработчикам искусственного интеллекта использовать модели в различных рамках [глубокого обучения](https://saturncloud.io/glossary/deep-learning), инструментах, средах выполнения и компиляторах. Когда модель экспортируется в формат ONNX, она включает в себя как структуру графа вычислений, так и параметры модели. Это позволяет импортировать модель в другой фреймворк, где ее можно использовать для вывода или дальнейшего обучения. [3]

Стандарт ONNX совместим с большим количеством фреймворков, таких как:

* TensorFlow
* PyTorch
* scikit-learn
* Keras
* MXNet
* CNTK
* Core ML
* PaddlePaddle

И другими.

Как работает ONNX:

1. Экспорт модели:
   * Разработчик экспортирует свою модель из фреймворка машинного обучения в формат ONNX.
   * Фреймворк машинного обучения предоставляет инструменты для экспорта модели в ONNX.
2. Преобразование модели:
   * ONNX Runtime преобразует модель ONNX в формат, совместимый с целевым фреймворком или приложением.
   * ONNX Runtime может оптимизировать модель для обеспечения высокой производительности.
3. Загрузка и использование модели:
   * Разработчик загружает преобразованную модель в целевой фреймворк или приложение.
   * Целевой фреймворк или приложение использует модель для выполнения вычислений.

Из очевидных преимуществ отмечается:

* Кроссплатформенность. ONNX совместим с различными платформами и языками программирования, что позволяет легко переносить модели между фреймворками и приложениями.
* Простота использования. ONNX имеет простой и понятный формат, что упрощает его использование и интеграцию.
* Производительность. ONNX может быть оптимизирован для обеспечения высокой производительности.
* Сокращение времени разработки. ONNX позволяет разработчикам сосредоточиться на разработке своих приложений, а не на конвертации моделей.

Однако библиотека обладает своими ограничениями:

* Не все модели могут быть экспортированы в ONNX, так как они могут быть слишком сложны
* При экспорте модели может произойти потеря точности
* Не все фреймворки машинного обучения совместимы с ONNX.

1. Fast-api – веб-фреймворк для создания быстрых и минималистичных HTTP API-серверов. Поддерживает сериализацию, валидацию и асинхронность из коробки. Благодаря ему можно минимальными трудозатратами предоставить интерфейс работы с нейронной сетью наружу. В этом случае интеллектуальный сервис будет представлять из себя самостоятельную единицу, запускаемую отдельно и независимо. Кроме того, fast api автоматически генерирует и отображает документацию согласно спецификации OpenAPI. Это означает, что клиенты легко смогут разобраться с предоставляемым интерфейсом интеллектуального сервиса.

Поскольку в команде, где в настоящий момент работает автор, параллельно развивается несколько проектов, использующих разных технологии и разрабатываемые разными командами разработки, наиболее предпочтительным методом представляется второй – создание отдельного и независимого интеллектуального сервиса, позволяющего работать с искусственным интеллектом через REST API интерфейс. Этот выбор имеет следующие преимущества:

* Простота реализации. Коллегам, не владеющим навыками работы с ONNX, и, вероятно, не разбирающимся в архитектуре нейронных сетей, гораздо проще будет работать с REST API, нежели с библиотекой, которая подключается определённым образом прямо в коде
* Выделенные ресурсы. Интеллектуальный сервис будет хоститься на производительной машине. Машина эта может находиться в специальном ЦОДе (центре обработки данных). Это позволит эффективно переиспользовать ресурсы
* Масштабирование. Интеллектуальный сервис можно будет легко масштабировать посредством запуска большего количества экземпляров.

Но у этого решения так же есть и недостатки:

* Разрыв соединения. Интеллектуальный сервис может упасть, сетевая связность может нарушиться – подобные форс-мажорные обстоятельства оставят микробиологическую и другие системы без интеллектуальной составляющей
* Сложность развёртывания. Добавляется ещё один сервис с новым для компании языком, который придётся собирать, развёртывать, хостить и обслуживать.

Взвесив все преимущества и недостатки обоих решений, принято остановиться на втором: Искусственный интеллект будет «упакован» в отдельный независимый сервис, называемый Интеллектуальным, посредством вывода конечных точек наружу с помощью REST API интерфейса. Данное решение обеспечит простоту реализации и гибкость в применении по отношении к разным системам в компании.

* + 1. Выбор способа обучения

Распознавание образов на изображениях может выполняться с использованием двух разных методов:

* Традиционные методы обработки изображений
* Современные нейронные сети глубокого обучения

Традиционные методы не требуют исторических данных для обучения и, следовательно, не требуют аннотированных изображений, где люди бы размечали образы вручную. Такие методы имеют очевидные минусы – они сложно справляются со сложными сценариями, такими как освещение, окклюзия, тени, эффект беспорядка, частично скрытые объекты и так далее. Соответственно, методы традиционного анализа изображения нельзя использовать в случаях, когда требуется высокая точность распознавания образов.

Современные методы глубокого обучения, с другой стороны, зависят от контролируемого или неконтролируемого обучения. Распознавание образов с использованием методов глубокого обучения гораздо более устойчиво к сложным сценам, сложному освещению, окклюзиям и так далее. Существенная сложность заключается в том, что для глубокого обучения требуется большое количество обучающих данных, разметить которые, как правило, должен эксперт. Процесс аннотации изображений является сложным, долгим и дорогостоящим. Для качественного обучения нужен набор данных с десятками или даже сотнями тысяч размеченных изображений.

Несмотря на все сложности второго подхода, на данный момент он признан стандартом в области качественного распознавания образов, широко признан исследователями и активно используется крупными компаниями. [7]

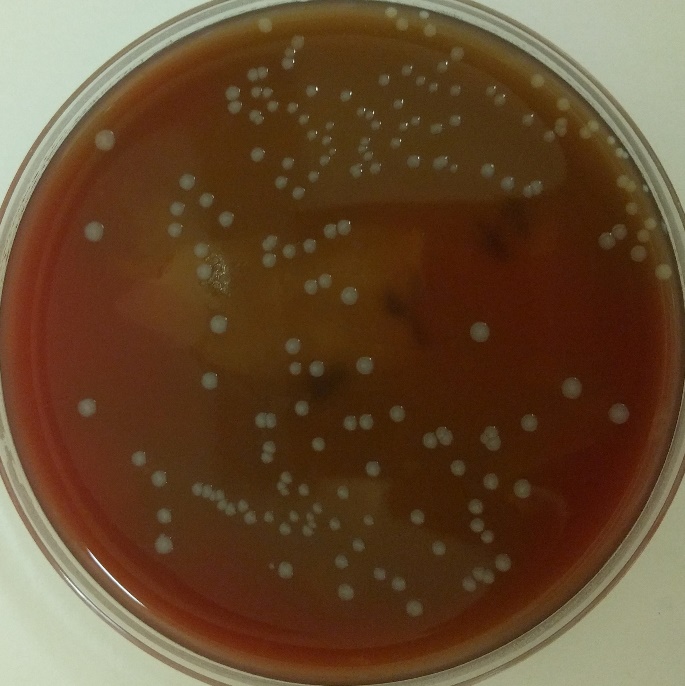
Стоит задача выбрать наиболее подходящий способ распознавания колоний микроорганизмов в чашке Петри. Далее будет проведён небольшой анализ, опираясь на результаты которого будет возможно принять правильное решение касаемо метода распознавания образов на изображениях.

Предполагается, что лаборанты будут делать фотографии чашек Петри либо с камер своих смартфонов, либо с веб-камер рабочих компьютеров. В любом случае никогда не будут созданы идеальные условия, в которых, грубо говоря, посреди питательной среды контрастно будут обозначать своё присутствие колонии микроорганизмов. Каждая фотография будет иметь свои уникальные искажение, среди которых:

* Низкий контраст, вызванный несовершенством оборудования
* Вуаль или замыленность из-за грязной оптики
* Пластиковая крышка, закрывающая содержимое чашки
* Непрямой угол сделанной фотографии
* Посторонний мусор

И множество других факторов, из-за которых страдает качество. С такими “боевыми” условиями традиционные методы распознавания образов не справятся, или, по крайней мере, не смогут показать достаточную точность.

Примеры фотографий:

Рис. 1, 2 – изображения чашек Петри

В случае выбора методов глубокого обучения для распознавания колоний микроорганизмов в чашке Петри встаёт проблема размеченных данных – нужны тысячи аннотированных специалистами изображений, на которых чётко локализованы и идентифицированы колонии.

К счастью, благодаря ранее внедрённому заказчикам программного решения – мобильного приложения – удалось набрать достаточного количества датасет размеченных изображений. Количество аннотированных изображений исчисляется десятками тысяч, а условия сделанных фотографий – самые разнообразные.

* + 1. Резюме

Таким образом, вследствие сложности распознавания колоний микроорганизмов на фотографиях неидеального качества, вызванного практическими условиями работы, было принято решение использовать современные методы распознаваний образов на изображениях с использованием глубокого обучения. Достаточный набор размеченных данных для этого удалось заполучить.

* + 1. Выбор фреймворка

Для разработки и обучения нейронных сетей на python предлагается множество разнообразных фреймворков. Неполный список доступных для глубокого обучения нейронных сетей (начиная с наиболее популярных): TensorFlow, PyTorch, Keras, MXNet, CNTK, Caffe2, Deeplearning4j, Chainer. Каждый из них имеет свои преимущества и недостатки, а также зону своего эффективного применения.

Далее будут рассмотрены данные инструменты для разработки и обучения нейронных сетей. В итоге будет сделан выбор в пользу того или иного фреймворка, подходящего лучше всего для решения задачи в контексте данной выпускной квалификационной работы.

1. TensorFlow

Фреймворк создан Google и написан на python и c++. Один из лучших фреймворков с открытым исходным кодом для численных расчётов. Используется в крупных и продвинутых проектах и используется для создания многослойных нейронных сетей. Применяется для распознавания голоса, изображений и в текстовых приложениях, например в гугл-переводчике. Из преимуществ фреймворка можно отметить:

* Немалое количество руководств и документаций
* Поддержка большим сообществом разработчиков и крупными технологическими гигантами
* Поддержка распределенного обучения
* Поддержка мониторинга процессов обучения моделей и визуализации

Однако есть и существенное ограничение – поддержка лишь одного языка программирования python.

1. PyTorch

PyTorch – библиотека на python. Является наследником Torch, которая написана на Lua. Разработана в компании Facebook и используется Facebook, Tesla, Twitter, Nvidia, Salesforce, Оксфордским университетом и многими другими. Используется для быстрого и эффективного обучения моделей и является выбором большого числа исследователей. Из преимуществ PyTorch можно выделить:

* Процесс моделирования прост и прозрачен благодаря архитектуре фреймворка
* Имеет большую экосистему пакетов, которая развивается по инициативе сообщества
* Простота отладки
* Обладает декларативным параллелизмом данных
* Содержит большое количество предобученных моделей, которые легко совмещать и комбинировать
* Поддерживает распределённое обучение

Из недостатков можно выделить меньшее количество примеров и менее многочисленное сообщество.

PyTorch лучше всего подходит для создания прототипов или небольших проектов. Когда дело доходит до крупных кроссплатформенных решений – следует предпочесть TensorFlow.

1. Keras

Keras – минималистичная python библиотека, которая может работать поверх TensorFlow, Theano или CNTK. Была разработана инженером Google Франсуа Шолле для ускорения экспериментов. Поддерживает большое количество слоёв нейронных сетей: свёрточные, рекуррентные, плотные. Keras эффективен в решении задач распознавания изображений, распознавания речи, звука и так далее. Главная особенность библиотеки – скорость разработки, что позволяет быстро перейти от замысла к реализации.

Из важных особенностей можно отметить:

* Очень быстрое и простое прототипирование
* Легковесный инструмент в контексте создания моделей глубокого обучения с большим количеством слоёв
* Имеет простой и интуитивно понятный интерфейс – справится даже новичок
* Просто расширяется – новые модули легко добавляются в виде классов и функций
* Поддерживает обучение на кластере графических процессоров
* Поддерживает тензорные процессоры разных производителей, таких как Nvidia, Google, AMD и другие

Среди недостатков библиотека можно выделить:

* Слишком высокоуровневая, что ограничивает некоторые возможности кастомизации
* Её возможности ограничиваются фреймворками, над которыми она построена: TensorFlow, CNTK, Theano

Таким образом, Keras – минималистичная и простая библиотека, способная работать поверх более мощных фреймворков и предоставляющая меньший контроль над нейронной сетью, чем, например, TensorFlow вследствие своей простоты.

1. MXNet

MXNet – платформа для глубокого обучения нейронных сетей с открытым исходным кодом. Берёт своё начало из научный среды и представляет из себя лёгкую, универсальный и высоко масштабируемую среду для глубокого обучения. Библиотека активно поддерживается крупными технологическими гигантами, такими как Amazon, и делает особый упор на машинное зрение, понимание человеческой речи и языка, порождающие модели, свёрточные и рекуррентные сети. Библиотека очень легковесная, и вследствие этого её достаточно просто масштабировать на нескольких процессорах и машинах, что отлично подходит для обучения на больших наборах данных. Она позволяет определять, обучать и развёртывать сети на самых разных устройствах – от высокопроизводительных серверов в облаках, до мобильных и даже носимых устройств. Важные особенности MXNet:

* Простота и удобство программирования
* Высокая степень переносимости между платформами
* Возможность масштабирования в самых широких диапазонах
* Совместима с большим количеством языков программирования

Среди недостатков MXNet можно отметить невысокую популярность в исследовательской среде и небольшое сообщество разработчиков по сравнению, например, с TensorFlow.

В итоге, MXNet – хороший фреймворк для больших промышленных проектов. Но, поскольку он еще относительно нов – есть вероятность, что не получится получить нужную поддержку от сообщества, когда она понадобится.

1. CNTK

CNTK – иначе называется как Microsoft Cognitive Toolkit – фреймворк для обучения глубоких нейронных сетей с открытым исходным кодом. Его особенность заключается в том, что он представляет нейронные сети как серию вычислительных действий через направленный граф, где конечный узлы – это значения или параметры сети, а другие узлы – операции матрицы в ответ на входные значения. Фреймворк позволяет легко разрабатывать и совмещать разные типы моделей, например, упреждающие глубокие нейронные сети, свёрточные сети (RNN), сети с краткосрочной памятью (LSTM). Обучение происходит по принципу стохастического градиентного спуска, при котором вычисления автоматически распараллеливаются между несколькими вычислительными машинами и серверами.

В фреймворке CNTK можно отметить следующие особенности:

* Высокие скорость и масштабируемость. Глубокое обучение происходит быстрее, чем в других инструментах глубокого обучения
* Обладает самой простой в использовании и выразительной архитектурой среди всех инструментов глубокого обучения
* Качество на уровне коммерческого продукта

Выделить можно лишь один минус – маленькое сообщество разработчиков.

1. Caffe2

Caffe2 – как можно понять из названия – прямой потомок Caffe. Это фреймворк, написанный на c++, имеющий интерфейс на python. Он поддерживает свёрточные нейронные сети и сети прямого распространения. Фреймворк хорош для обучения моделей (без написания лишних строк кода), обработки изображений, улучшения существующих сетей. Создан для мобильных, а также крупномасштабных развёртываний в промышленных средах.

Фреймворк хорош по следующим причинам:

* Фреймворк содержит предварительно обученные модели, которые можно использовать для демонстраций
* Фреймворк характеризуется легкостью, быстротой и хорошей масштабируемостью
* Адаптирован к работе с другими библиотеками и фреймворкам, например PyTorch

Из недостатков можно отметить плохую документацию кода и сложность в компиляции, а также исключительно маленькое сообщество разработчиков.

На данный момент ни одна крупная организация не использует Caffe2 в своих продуктах.

1. Deeplearning4j

Выбрать библиотеку Deeplearning4j стоит при использовании языка java для разработки и обучения нейронной сети. Библиотека написана на java, интегрирована с Hadoop и Apache Spark и предназначена для использования в корпоративных средах на распределённых вычислительных модулях. Представляет из себя новую библиотеку, цель которой – обеспечить быстрое создание моделей для специалистов, не являющихся профессиональными исследователями. Deeplearning4j позволяет импортировать модели нейронных сетей из большинства популярных сред, таких как: TensorFlow, PyTorch, Keras и других, тем самым связывая богатую экосистему python с таковой на java. В качестве прикладного программного интерфейса DL4J на Python используется Keras.

Библиотека Deeplearning4j обладает следующими особенностями:

* Быстрота, гибкость и эффективность
* Базируется на java и создана для java и производных jvm
* Возможность работы с Apache Spark и Apache Hadoop
* Открытые библиотеки и достаточно большое сообщество
* Хорошо задокументирована

В целом, библиотека Deeplearning4j будет полезна в том случае, если разработчик хочет разрабатывать и обучать нейронные сети на языке java.

1. Chainer

Chainer – ещё один python-фреймворк для обучения глубоких нейронных сетей. Поддерживается такими крупными компаниями, как IBM, Intel, Nvidia, Amazon. Как и остальные, может без проблем запускаться на нескольких GPU.

Особенность chainer заключается в том, что существующие нейросети могут быть модифицированы прямо во время исполнения. Главными же недостатками фреймворка являются сложность отладки и крайне маленькое сообщество людей, поддерживающих его. [4][5][6]

* + 1. Резюме

После тщательного анализа сильных и слабых сторон всех библиотек, принимая во внимание ранее принятые решения по поводу языка для разработки и обучения нейронной сети, а также сложившийся технологический стек в компании, наиболее подходящим и сбалансированным по совокупности всех факторов решением представляется фреймворк на python TensorFlow.

* 1. Архитектура клиентской части

Клиентская часть (пользовательский интерфейс) в контексте данной выпускной квалификационной работы имеет самое минимальное значение. Стоит задача создать одностраничное представление для работы с изображением чашки Петри, в том числе с результатами идентификации колоний на ней. При выборе инструментов для клиентской части центральное внимание стоит уделить уже принятым в команде стандартам. Поскольку решение предполагается встроить в существующую микробиологическую систему, единственным вариантом встроить новое представление является использовать те же решения, что использовались для разработки остальных её частей.

Среди использованных технологий:

* Next.js – фреймворк на javascript, использующий библиотеку React для построения Server Side Render - приложений (SSR) и статически-генерируемых сайтов.
* Ant Design – система дизайна, используемая для построения пользовательского интерфейса. Включает в себя набор готовых к использованию компонентов, инструментов и принципов дизайна.
* TypeScript – статически-типизированный язык программирования, основанный на javascript.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Информационно-правовой портал [Электронный ресурс]. – Режим доступа: http://government.ru/docs/all/138589/ (дата обращения: 26.11.2023)
2. Минцифры России [Электронный ресурс]. – Режим доступа: <https://ит-гранты.рф/novolabsystem>, сноска “О проекте” (дата обращения: 26.11.2023)
3. Платформа Saturn Cloud [Электронный ресурс]. – Режим доступа: <https://saturncloud.io/glossary/onnx-open-neural-network-exchange/>, (дата обращения: 15.03.2024)
4. Хамидулин Т.Г. Сравнение программного обеспечения глубокого обучения. // Вестник Поволжского Государственного Университета. Серия: Технические науки. – 2022 г. (дата обращения: 16.03.2024)
5. Mateusz Opala. Deep Learning Frameworks Comparison – Tensorflow, PyTorch, Keras, MXNet, The Microsoft Cognitive Toolkit, Caffe, Deeplearning4j, Chainer/ [Электронный ресурс]: Режим доступа: https://www.netguru.com/blog/deep-learning-frameworks-comparison, – Загл. с экрана. (дата обращения: 16.03.2024)
6. Hands-On AI Part 5: Select a Deep Learning Framework / [Электронный ресурс]: Режим доступа: https://software.intel.com/en-us/articles/hands-on-aipart-5-select-a-deep-learning-framework, – Загл. с экрана. (дата обращения: 16.03.2024)
7. Тематический сайт с научными статья Habr [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/articles/709432/>, (дата обращения: 17.03.2024)