МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»
Кафедра математической кибернетики и компьютерных наук

ОПРЕДЕЛЕНИЕ НАЛИЧИЯ КРОВИ В ЖЕЛУДКЕ НА ЭНДОСКОПИЧЕСКИХ СНИМКАХ С ПОМОЩЬЮ ТЕХНОЛОГИИ КОМПЬЮТЕРНОГО ЗРЕНИЯ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студентки 4 курса 451 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Гурьяновой Евгении Вадимовны

Научный руководитель
к. ф.-м. н., доцент _____ С. В. Миронов

Заведующий кафедрой
к. ф.-м. н., доцент _____ А. С. Иванов

Саратов 2020

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Одними из актуальных проблем в медицине являются диагностика и лечение язвенной болезни желудка, острых гастродуоденальных язв и их осложнений [1]. Хирурги отмечают, что основная сложность в обнаружении кровотечений заключается в том, что от момента начала до возникновения осложнений или зачастую проходит очень мало времени, или процесс идет настолько медленно и незаметно для пациента и врача, что когда он переходит в острую стадию, организм уже очень ослаблен.

Единственным способом решения проблемы является практически непрерывная диагностика состояния пациента. Существуют различные способы такой диагностики, которые обладают различными недостатками, и поэтому медики находятся в постоянном поиске новых методик. Сотрудники отделения хирургии государственного учреждения здравоохранения «Саратовская городская клиническая больница №6 имени академика В.Н. Кошелева» предложили разработать программное обеспечение, которое бы позволяло в автоматическом режиме определять появление крови в желудке пациента и оповещало бы об этом врача. Необходимость автоматизации оповещения связана с тем, что кровотечение может начаться в любой момент, когда рядом может не быть врача, а пациент даже не чувствует, что с ним происходит.

Целью настоящей работы является разработка методики определения наличия крови на изображениях желудков пациентов. Для достижения указанной цели были поставлены следующие задачи:

- разработать алгоритмы на основе компьютерного зрения, выявляющие на изображении, переданном с эндоскопа, признаки наличия крови;
- разработать мобильное приложение для любых устройств под управлением ОС Android, которое:
  - позволяет просмотреть врачу текущее состояние больного или архив состояний;
  - при необходимости оповещает врача о возникновении экстренной ситуации с помощью отправки СМС-сообщения;
- спроектировать сервер, на котором происходит применение предобученной нейронной сети, для улучшения качества распознавания наличия крови в желудке.

Бакалаврская работа состоит из введения, двух глав, заключения, списка

использованных источников и двух приложений. Объем работы 44 страницы. Список литературы включает 37 наименований.

В первой главе представлены некоторые теоретические аспекты: описание технологии компьютерного зрения, языка программирования Kotlin, среды разработки Android Studio, компонентов Android-приложения, технологии клиент-серверного взаимодействия.

Во второй главе представлены реализованные алгоритм, мобильное приложение и сервер, описан процесс разработки и приведен пример использования разработанного приложения.

1 Теоретические и технические аспекты разработки алгоритма и приложения

1.1 Известные предшествующие разработки и обоснование предложенного подхода к решению

В настоящее время типичными способами выявления кровотечения являются клинические (учащенный пульс, сниженное артериальное давление, холодный липкий пот), лабораторные (показатели уровня Hb (гемоглобина), эритроцитов и Ht (гематокрита) в крови) и инструментальные (зондирование желудка) методы. Все они обладают таким недостатком, что чаще всего выявление геморрагии (т.е. кровотечения) происходит уже при потере значимого для организма объема крови. Это связано с тем, что клиническая картина может развиваться крайне медленно, лабораторные исследования требуют значительного времени для проведения анализа, а также отсутствует возможность постоянно брать кровь у пациента. Зондирование или ФГДС желудка также является не самой простой операций и зонд в силу его технических особенностей не может постоянно находиться у пациента.

Поэтому в диссертационной работе [1] были предложены два метода определения начала геморрагии: один на основе постоянного анализа электрохимических свойств биологических сред желудка, второй — на основе лазерной фотодетекции желудка в динамике.

Так как анализ должен проводиться достаточно часто и время обработки изображения должно быть минимальным, возникла идея использовать алгоритмы компьютерного зрения для быстрого определения наличия следов крови.

Чтобы не быть зависимыми от устройств, к которым может быть присоединен датчик, было решено реализовать разрабатываемый алгоритм в двух вариантах — с использованием библиотеки компьютерного зрения OpenCV и языка Python, и на языке Kotlin, который сейчас активно применяется для разработки мобильных приложений. Это позволит использовать данную методику на различных электронных устройствах, работающих под управлением любой операционной системы.

Также в данную работу была включена задача разработки сервера для использования предобученной нейронной сети, которая дает более точные результаты диагностики, но требует больше ресурсов и данных.

1.2 Технические средства, применяемые для разработки алгоритма и приложения

1.2.1 Компьютерное зрение (Computer Vision) и библиотека OpenCV

Computer Vision [2], сокращенно CV, определяется как область исследования, целью которой является разработка методов, пытающихся воспроизвести возможности человеческого зрения и позволяющих компьютерам «видеть» и «понимать» содержание цифровых изображений, таких как фотографии и видео.

Для использования в данном проекте была выбрана библиотека с открытым исходным кодом OpenCV [3], распространяющаяся по лицензии BSD. В нее входят более 2500 оптимизированных алгоритмов, среди которых представлены как классические, так и самые современные алгоритмы компьютерного зрения и машинного обучения [4]. Реализована на многих языках, включая Python, Java, C++ и Matlab.

Так как используемые исходные фотографии для работы будут получены в цветовом формате RGB, для решения задачи поиска объектов определенного цвета наиболее эффективным вариантом является преобразование картинки из RGB в цветовой формат HSV, поскольку в данной работе требуется работать именно с оттенками цветов, не рассматривая при этом яркость и насыщенность.

Реализуется алгоритм классификации изображений именно с помощью цветовых подпространств, поскольку путем анализа фотографий было определено, что интересующие объекты (пятна крови) не имеют конкретных форм, что делает использование алгоритмов распознавания, зависящих от форм объектов, невозможным. Также, алгоритм, основанный на использовании цветовых диапазонов, требует меньше ресурсов, что делает возможным его исполнение в мобильном приложении за достаточно короткое время.

1.2.2 Язык программирования Kotlin

Для разработки мобильного приложения было принято решение использовать язык Kotlin [5]. Kotlin — это статически типизированный язык, разработанный в 2011 году компанией JetBrains. В 2017 на конференции Google I/O Kotlin получил официальную поддержку для разработки Android-приложений. Подобно Java, C и C++, Kotlin поддерживает как объектно-ориентированное,

так и процедурное программирование.

Так как интегрирование библиотеки OpenCV в Android-приложение крайне затруднительно, для обработки изображений была использована встроенная в систему Android библиотека Graphics. В процессе написания алгоритма использовались классы `Bitmap` (для упорядочивания пикселей изображения в массив) и `Color` (для перевода изображения попиксельно в HSV формат) данной библиотеки.

Библиотека Volley для разработки сетевых приложений

В состав языка Kotlin входит HTTP-библиотека Volley, которая позволяет организовать взаимодействие мобильных приложений с сервером на основе Flask.

Библиотека позволяет разработчику автоматически планировать сетевые запросы, поддерживать множество согласованных сетевых подключений и приоритезацию запросов, отменять API запрос, интегрироваться в любой протокол, поддерживать RPC-операций для заполнения пользовательского интерфейса и представления страницы результатов поиска в виде структурированных данных и др.

1.2.3 Android Studio

Разработка на языке Kotlin реализуется обычно в специальной среде для создания приложений — Android Studio.

Android Studio [6] — это официальная интегрированная среда разработки (англ. Integrated Development Environment или IDE) Android-приложений, созданная на основе IntelliJ IDEA.

Все файлы сборки отображаются на верхнем уровне в разделе Gradle Scripts, и каждый модуль приложения содержит следующие папки:

- manifests: содержит файл AndroidManifest.xml;
- java: содержит файлы исходного кода на Java или Kotlin, включая тестовый код JUnit;
- res: содержит все остальные ресурсы, такие как макеты XML, код пользовательского интерфейса и растровые изображения.

1.2.4 Клиент-серверное взаимодействие

Помимо алгоритма, написанного на Kotlin в Android Studio, с разработанного мобильного приложения также существует возможность подключиться к серверу, на котором находится алгоритм распознавания наличия крови, отличный от того, который создан в приложении.

Фреймворк Flask

Одной из целей работы является разработка сервера, на котором будет работать алгоритм распознавания наличия крови от другого разработчика.

Для взаимодействия веб-сервера с приложением существует интерфейс шлюза веб-сервера WSGI (англ. Web Server Gateway Interface) [7]. Это спецификация, описывающая указанное взаимодействие, а также то, как именно веб-приложения могут быть объединены в одну цепочку для обработки конкретного запроса.

Flask предлагает вносить изменения, но не навязывает разработчику никаких зависимостей или готовых схем проекта. Выбираются и используются только те инструменты и библиотеки, которые необходимы для решения поставленной задачи [8]. По умолчанию, Flask не включает в себя абстрактного уровня базы данных, валидации форм или всего того, что уже есть в других библиотеках. Вместо этого, Flask поддерживает расширения, способные добавлять необходимую функциональность и так внедрять ее в существующий код, как будто она была изначально.

2 Приложение для диагностики кровотечения по снимкам

2.1 Описание работы

В данной работе реализуется алгоритм, написанный на языке программирования Python, для анализа эндоскопических снимков и распознавания наличия или отсутствия крови на данных изображениях при помощи технологии компьютерного зрения. Впоследствии, этот алгоритм также реализуется на языке Kotlin, с целью адаптации его под мобильное приложение, способное реализовывать данный алгоритм с точностью, сравнимой с версией на компьютере, и дальнейшей возможности анализировать изображения на телефоне. Реализация происходит в среде программирования Android Studio.

Далее, в рамках данной работы создается сервер с уже имеющейся моделью совершения предсказаний относительно фотографий, на который можно посылать изображения с мобильного приложения, а также получать ответ о распознавании наличия или отсутствия крови. Взаимодействие с сервером совершается с помощью платформы для web-приложений FLASK, а для создания окружения со всеми необходимыми для работы установленными зависимостями используется Anaconda.

Также, реализуется возможность в случае успешного распознавания наличия крови автоматически отправлять сообщения на конкретный указанный номер телефона, что осуществляется с помощью компонента платформы Android `SmsManager`, предназначенного для работы с SMS.

2.2 Алгоритм распознавания

Целью создания алгоритма является построение наиболее компактного и эффективного в плане использования ресурсов метода распознавания наличия крови на изображении, чтобы впоследствии можно было использовать его даже на смартфонах с достаточно низкой производительностью. В его основе лежит идея о том, что соотношение цветов заданного конкретного диапазона должно отличаться на изображениях, где присутствует кровь, и где ее нет. Далее рассматривается поэтапное построение данного алгоритма.

Сначала происходит считывание изображения и последующее преобразование его в цветовой формат HSV. Далее, определяется высота и ширина изображения, а также инициализируется массив `color_dist` на 180 элементов, поскольку параметр Hue принимается значение от 0 до 179. Затем, произво-

дится перегруппировка значений пикселей для удобства дальнейшей итерации. После подсчета количества пикселей каждого цветового значения (параметр Hue) производится расчет отношения числа пикселей конкретного значения к общему количеству пикселей на изображении.

Описанная последовательность действий применяется ко всем изображениям в датасете, впоследствии происходит разделение на тренировочные и тестовые данные в пропорции 3 к 1.

В результате анализа данных было установлено, что под эти критерии лучше всего подходит Hue = 2.

Далее производится поиск порогового значения для отношения количества пикселей с Hue = 2 к размеру изображения, с помощью которого будет происходить классификация изображений.

Подсчитывается точность предсказаний для изображений из категорий «больных» и «здоровых», и возвращается объект `None`, если хотя бы одно из значений точности меньше 40%. В противном случае, возвращается сумма указанных точностей, а также они сами по отдельности.

Затем, происходит перебор пороговых значений от 0 до 1 с точностью 0.001. Результатом является отсортированный список, состоящий из суммарной точности предсказаний, вероятности корректного распознавания наличия и отсутствия крови на изображениях, а также величины перебираемого порогового значения.

В результате, итоговая точность алгоритма составляет 90% на изображениях с кровью и 59% на «здоровых» изображениях. Поскольку приоритетом в задаче является точность распознавания наличия крови, результат можно считать крайне удовлетворительным.

2.3 Описание мобильного приложения

2.3.1 Внешний вид приложения

В разработанном мобильном приложении используются следующие визуальные компоненты:

- `Image_view`, в котором отображается загруженное изображение.
- Кнопка `img_pick_btn` с текстом «Выбрать изображение», позволяющая открыть галерею, найти и выбрать конкретный снимок.
- Кнопка `post_img_to_server_btn` с текстом «Получить предсказание с

сервера», при нажатии на которую запускается алгоритм преобразования изображения в необходимый для отправки на сервер формат и, далее, его отправления.

— Текстовое поле `text_result`, в котором выводится результат работы алгоритма: в зависимости от того, была ли обнаружена кровь на снимке, в данном поле появляется надпись «Нет крови» или «Кровь присутствует». По умолчанию поле имеет значение «Нет данных».
— Текстовое поле `server_text_result`, в котором выводится результат предсказания алгоритма, находящегося на сервере, в зависимости от которого появляется аналогичная по содержанию с `text_result` надпись. Пока алгоритм на сервере производит расчеты для предсказания, поле имеет значение «Запрос обрабатывается».
— Два текстовых поля `textView` и `textView2`, служащие заголовками, со значениями «Результат локальной обработки» и «Результат обработки на сервере» соответственно.
— Кнопка `automation_mode_btn`, при нажатии на которую активируется или отключается режим автоматической обработки изображений. При этом, отображающийся на ней текст имеет значение «Включить автоматический режим» или «Выключить автоматический режим» соответственно.
— Текстовое поле `automation_update_text` в котором в автоматическом режиме работы приложения выводится время, оставшееся до следующей обработки изображения.

2.3.2 Алгоритм обработки и распознавания изображений

При реализации разработанного алгоритма в мобильном приложении, сначала идет загрузка выбранного в галерее изображения. Далее, в целях обработки полученного изображения, оно преобразуется в объект типа `Bitmap`. Затем, размер изображения меняется таким образом, что если сумма высоты и ширины превышает 2000, то большая из сторон устанавливается равной 1000, а меньшая — уменьшается на столько, чтобы сохранилось первоначальное соотношение сторон.

Впоследствии, идет преобразование изображения в массив пикселей, где каждое значение приводится к цветовому формату HSV. Далее, в соответствие с алгоритмом, написанным на Python, происходит вычисление необходимой

для классификации изображений величины.

Поскольку в Android-компоненте `Color` параметр Hue цветового формата HSV принимает 360 различных значений, а не 180, для подсчета количества пикселей со значением Hue = 2 (именно оно использовалось при написании алгоритма на Python) необходимо рассматривать Hue, равный 4 и 5.

В завершение, итоговый результат вычислений сравнивается с пороговым значением, равным 0.03 (поскольку именно эта величина была получена в ходе разработке алгоритма на Python), и, если результат больше этого значения, то в текстовое поле `text_result` помещается надпись, уведомляющая о наличии крови, иначе — об ее отсутствии.

2.3.3 Реализация взаимодействия с сервером

Взаимодействие приложения с сервером происходит при помощи библиотеки Volley. Сначала указывается тип запроса (в данном случае POST), его URL, а также обработчики ответов и ошибок. Далее, формируется тело запроса: создается `HashMap`, в котором будут храниться пары вида «имя параметра : значение». В результате сформированный объект помещается в очередь запросов, где и происходит его отправка на сервер.

В обработчике ответа от сервера происходит преобразование пришедших данных в объект типа «JSON», а затем из него извлекается поле с именем «prediction», в котором хранится результат классификации отправленного изображения. В зависимости от полученного значения в текстовое поле `server_text_result` выводится текст, информирующий о наличии или отсутствии крови, либо о возникшей ошибке в случае ее появления.

Автоматическое уведомление с помощью сообщения

В приложении также существует возможность отправки сообщения на любой указанный смартфон в случае, если с сервера был получен положительный ответ о наличии крови на изображении. Осуществляется отправка сообщений с помощью компонента платформы Android для работы с SMS, который называется `SmsManager`. Данный компонент управляет такими SMS-операциями, как отправка текстовых и pdu-сообщений. Этот объект можно получить, вызвав статический метод `getDefault()`. Чтобы создать экземпляр класса `SmsManager`, способный взаимодействовать с конкретным идентификатором подписки, вызывается метод `getSmsManagerForSubscriptionId(int)`.

Отправка сообщения происходит с помощью метода `sendTextMessage`, в аргументах которого указывается номер получателя `phoneNumber`, а также текст СМС-сообщения.

2.3.4 Режим для автоматической обработки изображений

Также в приложении была реализована возможность переключения режима работы в автоматический, при котором раз в 30 секунд берется последнее добавленное изображение из заданной папки, затем оно обрабатывается с помощью локально реализованного классификатора и на сервере.

Сначала происходит получение списка файлов заданной папки и его последующая сортировка по дате изменения. Далее последнее измененное (добавленное) изображение локально обрабатывается. Затем изображение отправляется на сервер и обрабатывается полученный результат.

Также был реализован объект `timer`, который запускает вышеописанные действия раз в 30 секунд. При этом, оставшееся до следующего запуска время выводится в текстовое поле `automation_update_text`.

Также была создана возможность изменение режима работы приложения. В данном приложении имеется два режима: «auto» и «manual». При переключении в режим «auto», кнопки `img_pick_btn` и `post_img_to_server_btn` становятся неактивными, кнопка переключения `automation_mode_btn` принимает соответствующий вид, а также становится видимым текстовое поле `automation_update_text`.

При переключении в режим «manual» таймер для запуска функции обработки изображения останавливается, текстовое поле `automation_update_text` скрывается, делаются активными кнопки `img_pick_btn` и `post_img_to_server_btn`. Соответственно изменяется внешний вид кнопки `automation_mode_btn`.

2.4 Реализация сервера

Сервер выполняет только одну задачу — классификацию изображения из входящего запроса, поэтому он содержит в себе только один маршрут «http://<адрес сервера>/» (корневой). Для реализации сервера был использован фреймворк Flask.

В рамках дипломной работы сервер реализован таким образом, что он доступен только из той же сети, в которой находится смартфон.

Также с помощью фреймворка Flask был реализован следующий обработчик входящего POST-запроса с содержащимся в его теле изображением. Происходит считывание файла из тела запроса, преобразование его в numpy-массив типа uint8, а затем декодирование массива в цветное изображение с помощью библиотеки OpenCV. Декоратор
`@app.route('/', methods=['POST'])` непосредственно содержит в себе маршрут первым аргументом.

После преобразования в цветное изображение происходит инициализация класса StomachPredictor. Данный класс необходим для применения обученной нейронной сети и получения предсказания на основе полученного изображения, сервер возвращает результат обработки в формате JSON. В случае отсутствия изображения в теле запроса данный метод возвращает ошибку 400.

В рамках класса StomachPredictor были разработаны функции prepare_image и make_single_prediction, отвечающие за приведение изображения к подходящему для предобученной нейронной сети формату и генерацию предсказания на ее основе соответственно.

Функция prepare_image преобразует изображение в установленное для нейросети разрешение путем изменения размера и добавления горизонтального или вертикального пространства так, чтобы получился квадрат со сторонами, равными 224 пикселям. После этого, изображение преобразуется в тензор и меняется порядок каналов (что необходимо для корректной работы нейронной сети), затем происходит нормализация цветовых значений пикселей.

Функция make_single_prediction помещает результат обработки изображения метода prepare_image в тензор, затем подготовленные данные подаются на вход предобученной модели. Результатом работы нейронной сети является тензор, содержащий пару чисел, для его интерпретации определяется позиция наибольшего числа. Модель была обучена таким образом, что принадлежность изображения к классу со значением 0 обозначает наличие крови на нем, а со значением 1 — отсутствие крови.

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной работы был разработан алгоритм определения наличия крови на изображениях желудков пациентов, а также разработано мобильное приложение, позволяющее врачу увидеть картину происходящего в желудке и в случае необходимости оповещающее врача о возникновении экстренной ситуации с помощью отправки СМС-сообщения. Программа позволяет просмотреть врачу текущее состояние больного или архив состояний. Также был спроектирован сервер, на котором происходит применение предобученной нейронной сети, для улучшения качества распознавания наличия крови в желудке.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 *Беликов, А. В.* Ранняя диагностика рецидива язвенного гастродуоденального кровотечения / А. В. Беликов. — Саратов, Россия.

2 An Introduction to Computer Vision [Электронный ресурс]. — URL: https://machinelearningmastery.com/what-is-computer-vision/ (Дата обращения: 02.05.2020). Загл. с экр. Яз. англ.

3 *Келер, А.* Изучаем OpenCV 3 / А. Келер. — ДМК Пресс, 2017.

4 OpenCV: от импорта до распознавания лиц [Электронный ресурс]. — URL: https://lambda-it.ru/post/opencv-ot-importa-do-raspoznavaniia-lits (Дата обращения: 02.05.2020). Загл. с экр. Яз. рус.

5 *Жемеров, Р.* Kotlin в действии / Р. Жемеров, С. Исакова. — ДМК Пресс, 2017.

6 Meet Android Studio [Электронный ресурс]. — URL: https://developer.android.com/studio/intro (Дата обращения: 18.05.2020). Загл. с экр. Яз. англ.

7 What is WSGI? [Электронный ресурс]. — URL: https://wsgi.readthedocs.io/en/latest/what.html (Дата обращения: 23.05.2020). Загл. с экр. Яз. англ.

8 Flask [Электронный ресурс]. — URL: https://palletsprojects.com/p/flask/ (Дата обращения: 23.05.2020). Загл. с экр. Яз. англ.