Лицей «Физико-техническая школа»
Санкт-Петербургского Академического университета

Курсовая работа (отчет о практике)

Распознавание анкет

Работу выполнил:
ученик 11а класса: Ползик Даниэль

Научный руководитель:
Суворов Егор Фёдорович

Место прохождения практики:
Skype

Санкт-Петербург
2018

Аннотация

В рамках практики было создано приложение под платформу Android, которое проверяет бумажные тесты специального вида с вариантами ответов.
Распечатанные тесты заполняются учениками от руки.
Приложение умеет отправлять результаты тестирования нескольких работ на электронную почту. Это экономит время, которое пользователь обычно тратит на проверку.
В рамках работы был создан алгоритм для распознавания закрашенных квадратов. Также были собраны данные для оценки качества работы алгоритма и подбора параметров этого алгоритма. Была добавлена функция калибровки, которая по известному тесту переопределяет параметры алгоритма. В работе используется библиотека компьютерного зрения OpenCV. Ещё добавлена функция распознавания владельца теста по номеру.

Введение

Учителя много времени тратят не только на подготовку уроков и их проведение, но и на проверку работ. Очень часто учителя проверяют знания учеников с помощью работ с вариантами ответов. Например, на уроке литературы для проверки знаний прочитанного текста учитель может дать небольшой тест с вариантами ответов. После написания учениками этой работы, у учителя образуется стопка из 30 работ, а если у учителя есть параллельный класс, то и того больше.
И проверка превращается в монотонный труд, так как учитель сравнивает одно и то же с одним и тем же.
Так как единственное значимое отличие в анкетах - это имена владельцев анкет, то, кажется, что их проверка должна легко автоматизироваться.
Теперь нужно было решить, а как именно автоматизировать процесс. Самый точный и надежный способ - это электронные тесты, когда каждый ученик выполняет работу за компьютером или телефоном и получается, что учителю вообще не надо ничего проверять. К сожалению, у этого варианта есть несколько значимых минусов. Тесты на компьютерах и телефонах это плохой вариант, так как учитель не может контролировать, чем пользуются ученики и высока вероятность списывания. Работа, которую ученики списали, бесполезна. Таким образом решено оставить бумажные тесты, но все равно как-то ускорить проверку, автоматически распознавая ответы. Для точного распознавания требуется получить наиболее четкое изображение. Наиболее четкое изображение бумажной анкеты может дать сканер, но сканер медленно работает.. Тогда возьмем обычный телефон с камерой.

Постановка задачи

Цель — разработать приложение под платформу Android, которое поможет преподавателям ускорить проверку работ с вариантами ответов.
Были выделены следующие задачи.

1. Разработка шаблона анкеты
2. Разработка алгоритма распознавания
3. Оценка качества алгоритма
4. Улучшение результатов
5. Подсчет результатов
6. Отправление по электронной почте результатов
7. Распознавание владельца анкеты
8. Создание списка учеников

Методика

В начале мы разбили нашу работу на небольшие части и составили их в план:

- изучение Android-программирования
- знакомство с библиотекой компьютерного зрения OpenCV по обработке изображения с камеры телефона
- создание прототипа анкеты
- создание алгоритмов распознавания и их тестирование
- подписывание анкет
- написание самого приложения.

Изучение Android-программирования

Для изучения основ Android-программирования я прошел летнюю практику в компании JetBrains. Там я научился создавать несложные приложения, например, калькулятор, добавлять кнопки и реагировать на нажатия, научился добавлять свайп (при проведении пальцем по панели на экране в определенном направлении) и реагировать на него, сдвигать картинку, если двигают полосу прокрутки. Так же изучил основы языка Kotlin, но в проекте все было написано на языке Java (так как Java мне знакома больше, чем Kotlin). Итогом практики в JetBrains была игра-стратегия, в которой использовались все основные структуры для работы с Android, в том числе, свайп (панель для перемещения по экрану панелей).

Создание шаблона теста

Захотелось уже иметь пример теста, чтобы начать писать алгоритм-1 для распознавания.
Тест было решено взять единого формата для всего.
Тест разделен на две части: левую, на которой будут располагать условия и всё, что нужно пользователю в произвольном формате, и правую более для нас значимая часть — квадраты (в фиксированных позициях), которые ученик будет закрашивать, исходя из того, правильный ответ или нет.

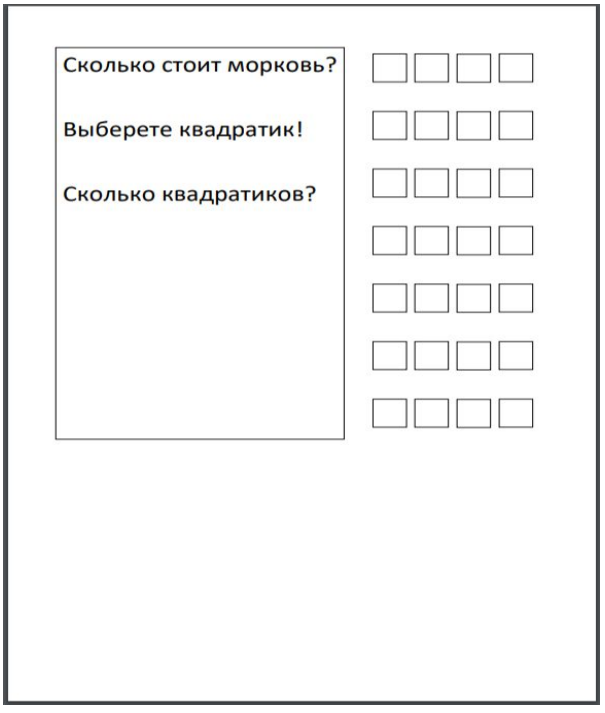


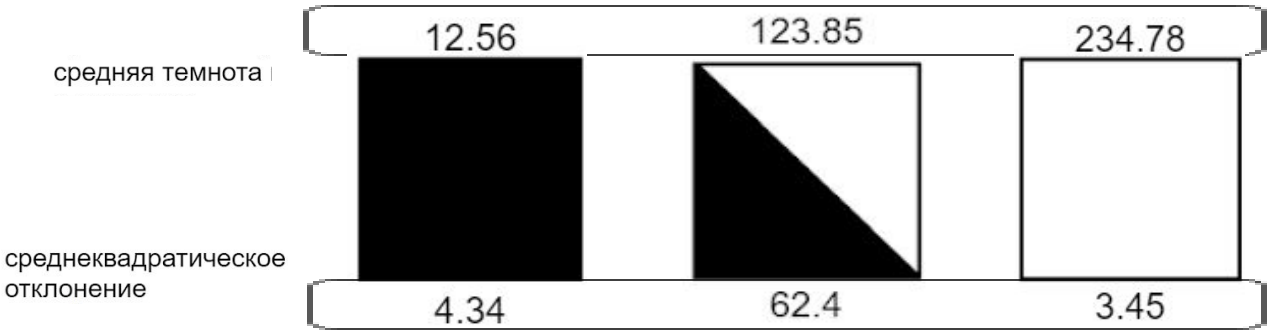
рис. 1 (шаблон теста)

Библиотека компьютерного зрения

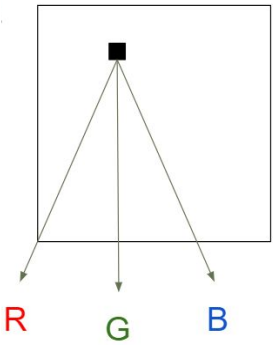
После знакомства с основами Android-программирования, мы решили изучать библиотеку для обработки изображения. Лучше всего для этого подошла библиотека OpenCV. Так как OpenCV самая распространенная библиотеке по компьютерному зрению с открытым кодом.
Эта библиотека позволяет выводить на экран изображение с камеры, рисовать поверх неё фигуры, обрабатывать изображение с помощью различных фильтров.
После чтения документации по некоторым функциям и изучения примеров программ, которые приложены к библиотеке, была написана программа, которая выводит изображение с камеры и рисует на экране таблицу квадратов N*M (рис. 3) (каждой клетке таблицы сопоставляется квадрат из анкеты), по каждому из квадратов выводит статистику: темноты и среднеквадратичный разброс темноты, где N - количество вопросов, а M - количество вариантов.

Создание алгоритма распознавания

Чтобы понять закрашен квадратик или нет, будем смотреть на среднюю темноту пикселей в квадрате и среднеквадратическое отклонение. По темноте определяем черный квадрат или белый (чем больше темнота, тем светлее квадрат), а по среднеквадратическому отклонению полностью ли закрашен квадрат (чем больше отклонение, тем менее равномерно закрашен квадрат).



Подробнее:
Изображение состоит из матрицы пикселей, каждый из которых задаётся тремя характеристиками: Red (красная компонента цвета), Green (зеленая), Blue (синяя). Представляется это в виде трех чисел (от 0 до 255). Для получения темноты одного пикселя мы считаем среднее арифметическое этих трех чисел. Таким образом мы получаем матрицу черно-белой картинки, что упрощает работу. Так как теперь каждый наш пиксель мы характеризуем одним числом.



$$x_i = \frac{R_i + G_i + B_i}{3}$$

Если посчитать среднее арифметическое темноты всех пикселей, то мы получим число, которое характеризует темноту изображения

$$x_{cp} = \frac{\sum_{i=0}^{n} x_i}{n}$$

, а если посчитать среднее квадратичное отклонение темноты пикселей от темноты изображения, то мы получим среднеквадратичный разброс темноты $M_2 = \frac{\sum_{i=0}^{n} (x_i - x_{cp})^2}{n}$.
Наконец, мы имеем тест и характеристики по квадратикам с ответами, теперь можно проверять ответы, то есть для каждой ячейки выводить: закрашена, не закрашена.
Закрашена ли клетка, мы проверяли сравнивая темноту клетки и разброс цветов.

```
if(x_cp <= CONST_1 and M_2 < CONST_2){
    квадратик закрашен
}else{
    квардратик не закрашен
}
```

Сбор статистики

Ниже, вы можете посмотреть на скриншот этой программы (рис. 2).
Если рамка квадрата - зеленая, то программа считает, что квадрат закрашен, а если красный, то не закрашен.
Как видно, при средней равномерности освещения и при использовании черной гелевой ручки, мы получаем полное совпадение с реальными значениями. При этом, константы, которые здесь используются на данном этапе были выбраны “на глаз”.

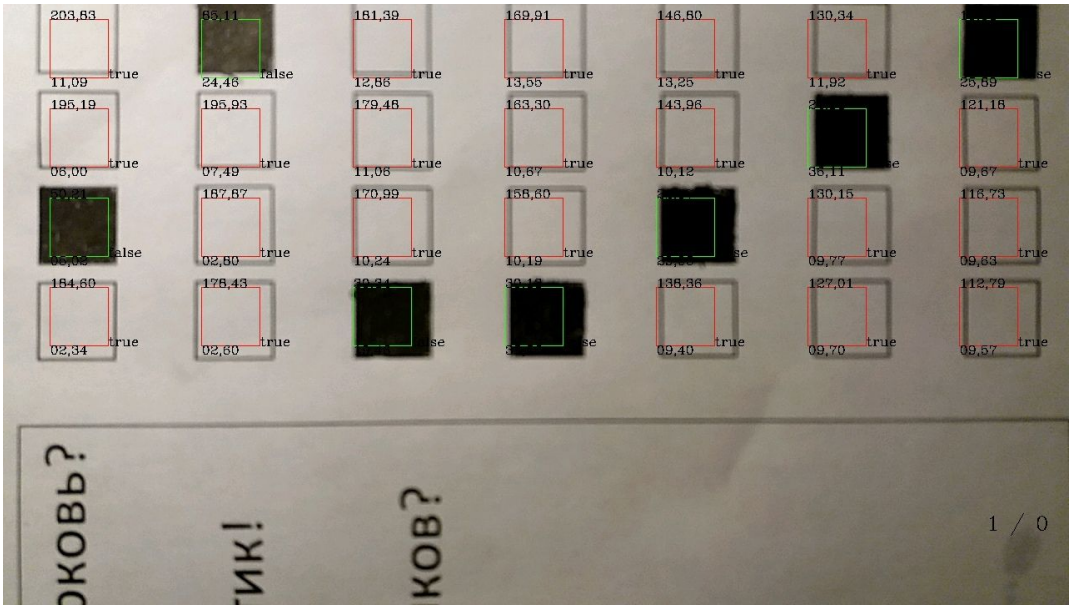
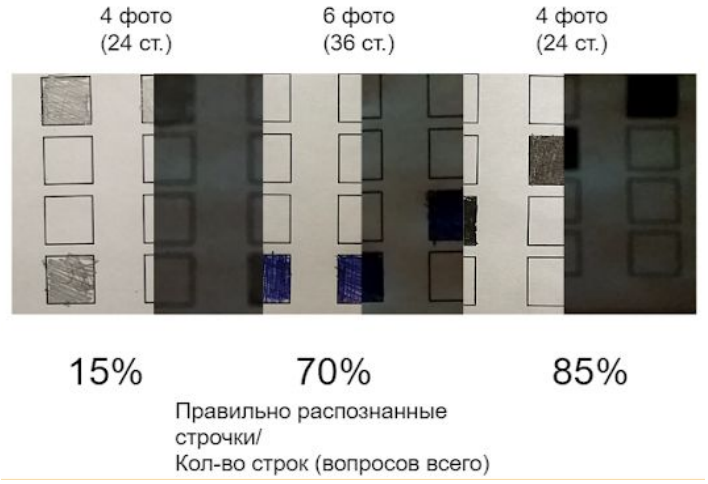


рис. 3

Для проверки работы алгоритма, нужно собрать статистику распознавания. Для этого добавим в приложения для распознавания функцию сохранения файла с данными по распознаванию. Было создано 4 теста, каждый из которых разными способами закрашен (1 карандашом, 2 синей шариковой ручкой, 1 черной гелевой ручкой) и сканировался при разном освещении.
В итоге, были получены результаты, которые показаны справа.



Было замечено, что почти все ошибки алгоритма-1 были из-за неидеального для него освещения. Чуть-чуть темнее или светлее и приложения уже допускает ошибки. И нужно было как-то менять константы, с которыми сравниваются темнота и отклонение в зависимости от освещенности.

Улучшение алгоритма (Калибровка)

Калибровка, в нашем случае - подбор параметров в зависимости от результатов сканирования теста, с известными нам ответами.
Так как результаты менялись, в зависимости от способа закрашивания, то было решено использовать тест с закрашенный в шахматном порядке.

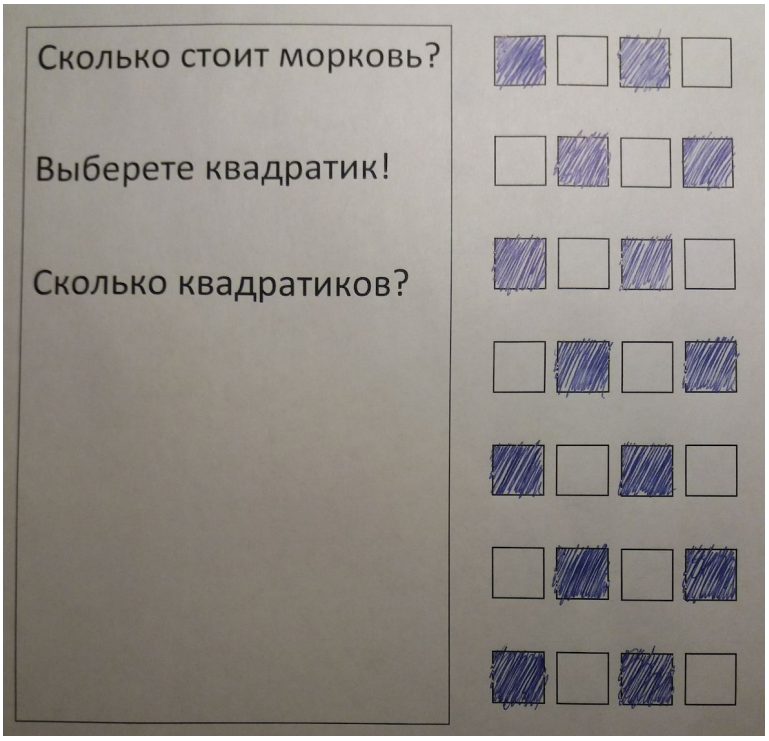


рис. 4 (шахматная калибровка)

Но так же было решено сканировать незаполненный тест (пустой). Так как после подготовки нескольких тестов для калибровки стало понятно, что каждый раз пользователю закрашивать тест в шахматном порядке будет долго и не удобно.

Сканируя в шахматном порядке закрашенный лист считаем $CONST_1 = \frac{2 \cdot x_ч + x_б}{3}$, где $x_ч$ - средняя темнота закрашенных квадратов (черных), а $x_б$ - средняя темнота не закрашенных квадратов (белых).
А сканируя пустой лист находим минимальное значение темноты (самый темный не закрашенный квадрат) и получаем новую константу $CONST_1 = 0.88 \cdot min(x_{ср})$.
Теперь так же, как и в алгоритме-1 считаем статистику.

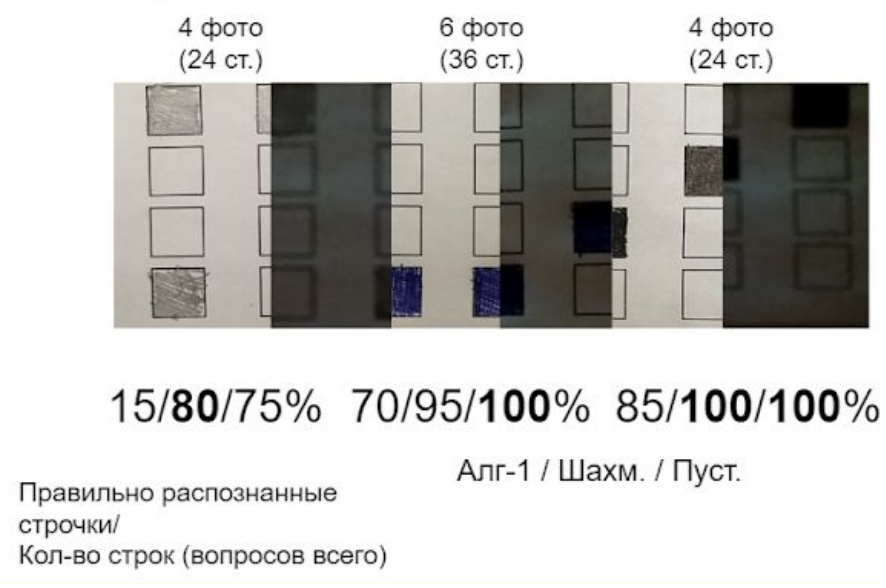


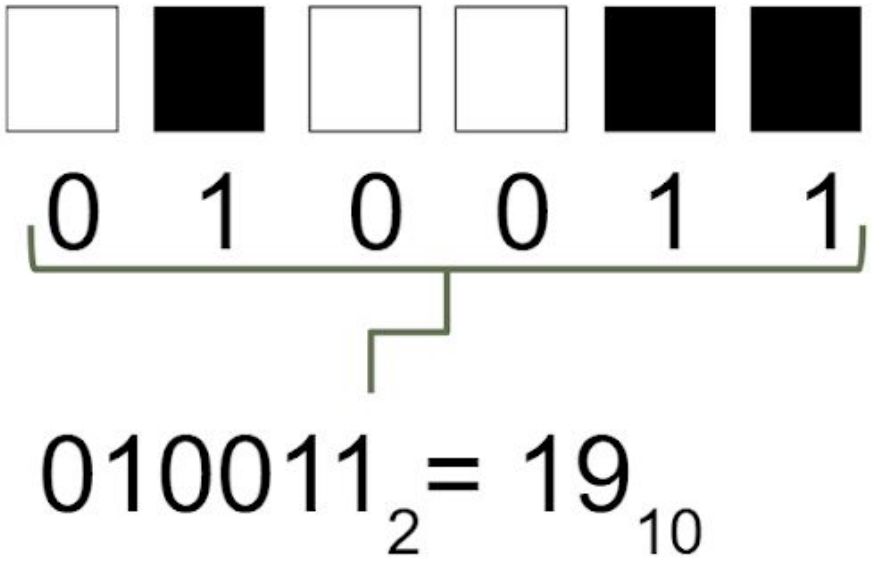
рис. 5 (статистика)

Обе калибровки распознали все виды тестов (по способу закрашивания) лучше, чем алгоритм-1. Из этих двух калибровок выбираем калибровку по пустому листу, так как:

- для этой калибровки не нужно ничего делать, когда в шахматной калибровке нужно закрашивать много квадратов
- по результатам статистики, различиям между калибровками, тоже отдаем предпочтение калибровке по пустому листу, так как она себя лучше проявила при распознавании синей ручки, которая для нас наиболее приоритетна.

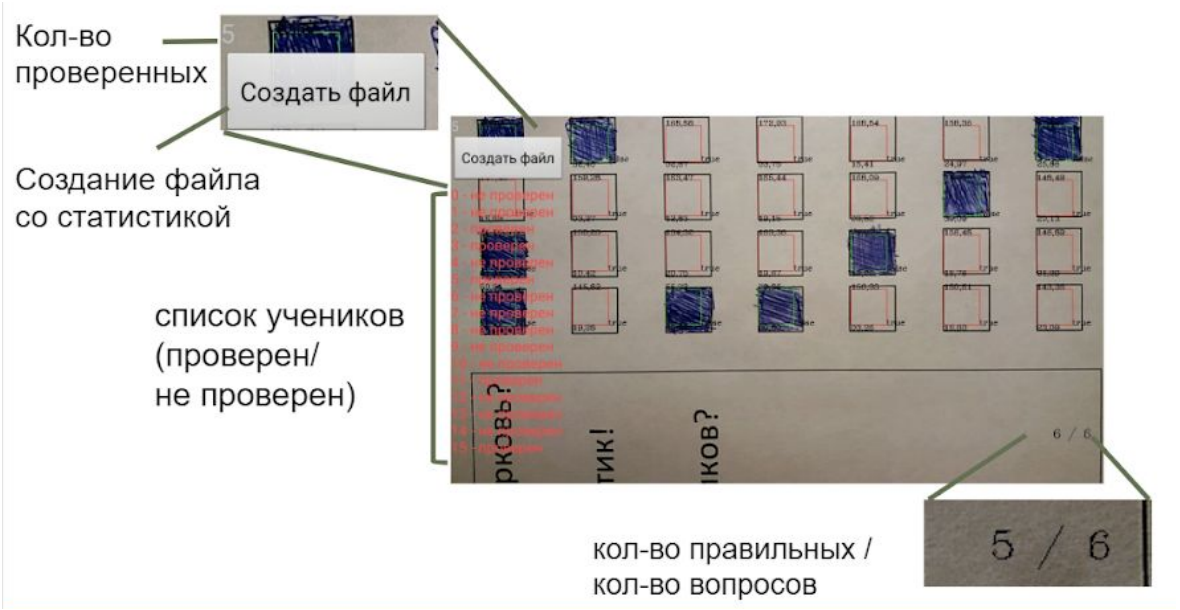
Распознавание имен

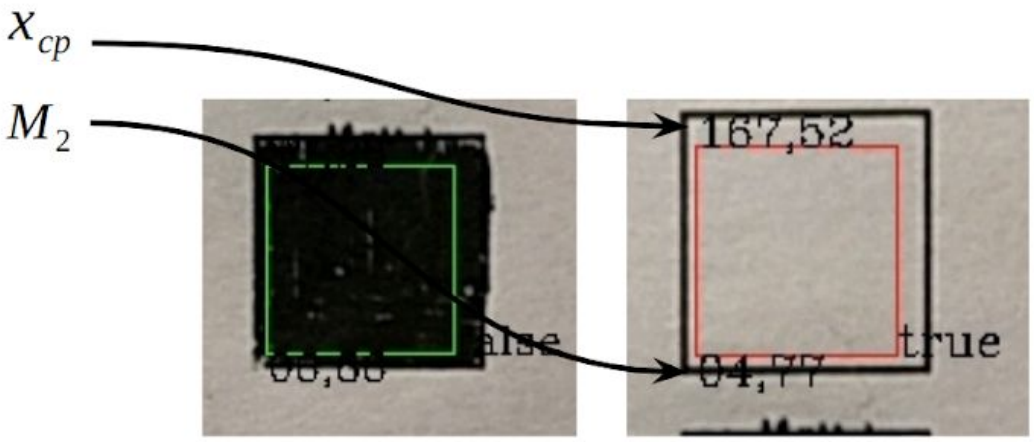
Каждому ученику выделим уникальный номер. Переведем это число в двоичную систему счисления, и каждый ученик запишет у себя на верхней строчке с квадратами номер. Закрашенный квадрат - 1, не закрашенный - 0.



После того, как мы научились распознавать номер ученика, создадим список уже проверенных учеников и будем отправлять всех учеников из этого списка. А также говорить пользователю, если такой ученик уже проверен.

Скриншот приложения





Отправка результатов (по почте)

Для отправки результатов была выбрана почта, так как электронная почта казалась самым простым способ отправки каких-либо данных.
Сначала, мы написали программу для отправки писем на компьютере (на java). Чтобы было проще тестировать. Для этого в языке java есть встроенная библиотека для отправки писем (javax.mail). Затем используя уже заведомо правильный код, перешли на телефон. Как оказалось, на java для Android, нет встроенной библиотеки для отправки писем. Поэтому была найдена сторонняя библиотека: com.sun.mail:android-mail:1.6.2. Изменив буквально несколько строчек из программы для пк и добавив библиотеку для почты, мы получили приложение, которое умеет отправлять при нажатии на кнопку письмо на указанный адрес электронной почты.

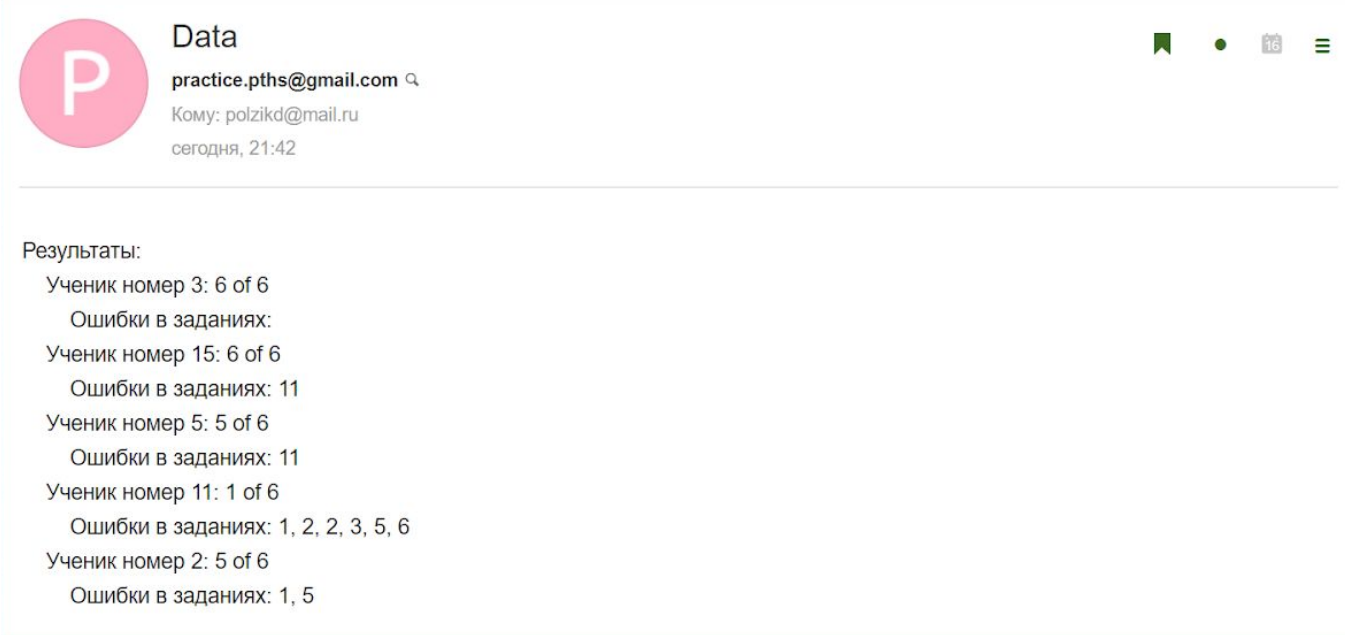


рис. (скан письма с результатами)

Изучение аналогов

Было найдено 2 наиболее известных аналога, которые умеют распознавать бумажные тесты и как-то сохранять ответы на телефоне или отправлять пользователю.

Аналоги	Ссылка на PlayMarket	Описание
ZipGrade	https://play.google.com/store/apps/details?id=com.zipgradellc.android.zipgrade	Только английский. Тесты разрешены только на 20, 50, 100 вопросов. Определяет очень хорошо. По якорям. Бесплатная версия ограничена 100 сканирований, а дальше нужно сканировать. Данные отправляются в виде n (кол-во учеников) листов, на каждом подробно описаны результаты каждого ученика, но нет обобщающей таблицы, для быстрого выставления оценок. На телефоне можно посмотреть статистику, а также таблицу по которой можно выставлять оценки, но на пк это никак не отправляется. Зато вырезается текстовое поле, в которое ученик вписывает имя, что помогает быстрее понимать, чья это работа. То есть небольшая фотография с именем ученика. Проверка происходит автоматически, то есть он сам решает, когда он определил, как расположена анкета
ExamReader	https://play.google.com/store/apps/details?id=com.bebyaz.examreader	Есть большой выбор из языков (в том числе и русский). В бесплатной версии можно проверять только по одному и результаты никуда не отправляются. Он только проверяет и говорит на экране телефона результат только что проверенного ученика и его ошибки. И если проверить следующего то предыдущего он забывает. Проверка хуже ZipGrade. Ошибок не допускает, но иногда говорит, что не распознал ответы и попросил переснять. Проверка происходит по нажатию на кнопку, то есть по фотографии

Пути развития

- Создание других алгоритмов распознавания, тестирование и выбор оптимального
    - Брать все квадраты и сортировать по возрастанию и в месте наибольшего разрыва поставить границу между закрашенными и нет
    - Посмотреть на средний цвета пикселей
        - если белый, то не закрашен
        - если синий или черный, то закрашен
- Улучшение интерфейса

Возникали при пользовании приложением неудобства при нажатии на кнопки. Так до кнопок на панели меню пальцы не дотягивались, а также текст со списком учеников перекрывает часть изображения с камеры, что выглядит и не красиво и для пользования неудобно.

- Добавление автоопределения клеток

Очень полезная функция, которой приложение, к сожалению не обладает, автоопределение местоположения клеток. То есть не будет возникать проблем с тем попал на квадраты пользователь или нет, а приложение само будет находить квадраты и их проверять. Эта функция есть у аналогов, и из-за ее отсутствия нам сложно конкурировать.

Результаты

Мы создали шаблон теста, который можно использовать в действии. При чем, если во всех аналогах страничка теста была отдельно от самих вопросов, то в нашем шаблоне все можно сделать на одном листе.
Мы разработали алгоритм распознавания, который в большинстве ситуация (при нормальном и равномерном освещении) давала правильный результат. И была написана программа для использования и тестирования этого и других возможных алгоритмов. Удачно собрав статистику стало понятно, что нужно создавать улучшения алгоритма (калибровка). Калибровка была успешно написана. Была собрана статистика по калибровке и она действительно улучшила результаты распознавания.
Также мы умеем создавать отправлять результаты по электронной почте, что очень удобно.
Еще приложение умеет распознавать владельца анкеты, говорить, если этого ученика проверяли. Так как приложение создает список проверенных учеников, что отправлять результаты сразу всех и не проверять одну анкету дважды.

Вывод

Мы получили приложение, которое умеет создавать анкету, проверять их, собирать эту информацию в один файл и отправлять все это на электронную почту.
У этого приложения есть куда расти, что описано в путях развития, но приложение является самостоятельным и им можно уже полноценно пользоваться, что является очень неплохим результатом.

Источники

1. https://habr.com/company/abbyy/blog/325094/
2. https://opencv.org/platforms/android/

Благодарность

Хочу поблагодарить своего научного руководителя Суворова Егора Фёдоровича за огромную помощь и вклад в практику, а также за огромное количество времени, которое Егор Федорович выделил. Это было очень интересное время. Я получил массу удовольствия от работы Егором Федоровичем. Это было интересно, познавательно и продуктивно, что очень важно. Был получен бесценный опыт, который, я надеюсь, поможет мне в будущем. Было здорово!
Также хочу поблагодарить Дворкина Михаила Эдуардовича за то, что свёл с Егором Федоровичем и отвечал на вопросы, когда они возникали.