ПРИЛОЖЕНИЕ СОРТИРОВКИ ФОТОГРАФИЙ В СООТВЕТСТВИИ С КРИТЕРИЯМИ ПОЛЬЗОВАТЕЛЯ

Студент 4 курса Малишевский И.А.,

Студент 4 курса Пажитных М.П.

Научный руководитель – старший преподаватель Чуйко В.А.

Белорусский государственный университет

Факультет радиофизики и компьютерных технологий

Кафедра интеллектуальных систем

Минск, Беларусь

В 2001 году Пол Виола и Майкл Джонс предложили метод детекции объектов, получивший широкое распространение, особенно в задачах детектирования объектов – Каскады Хаара [1].

Предварительная обработка изображения

Предварительная обработка изображения является важным этапом для оптимизации вычислений, повышения точности в обнаружении объектов и правильности работы алгоритма [1]. Предварительная обработка изображений для алгоритма каскадов Хаара включает два этапа:

1. Преобразование изображения в градации серого

Цель данного преобразования — уменьшение размерности данных, так как алгоритм каскадов Хаара работает исключительно с одноканальными изображениями. Исходное цветное изображение $I(x, y)$ в формате RGB преобразуется в изображение градации серого $I_{gray}(x, y)$. Формула преобразования из цветного изображения в градации серого следующая (1):

$$I_{gray}(x, y) = 0.299 \cdot R(x, y) + 0.587 \cdot G(x, y) + 0.114 \cdot B(x, y), \quad (1)$$

где $R(x, y)$, $G(x, y)$, $B(x, y)$ – интенсивности красного, зеленого и синего каналов в пикселе с координатами (x, y).

2. Построение интегрального изображения

Интегральное изображение представляет собой структуру данных, позволяющая ускорить вычисления суммы интенсивности пикселей в произвольной прямоугольной области изображения за счет использования предварительно накопленной суммы пикселей от начала координат до текущей точки [2]. Интегральное изображение S определяется формулой (2):

$$S(x,y) = \sum_{i=0}^{x}\sum_{j=0}^{y} I_{gray}(i,j)\,, \tag{2}$$

где $I_{gray}(i,j)$ – интенсивность пикселя на изображении в точке (i,j).

После построения интегрального изображения вычисляется сумма интенсивностей пикселей в произвольной прямоугольной области, определяемая угловыми координатами (x_1,y_1) и (x_2,y_2). Вычисление суммы происходит по формуле (3):

$$sum = S(x_2,y_2) - S(x_1-1,y_2) - S(x_2,y_1-1) + S(x_1-1,y_1-1), \tag{3}$$

Построение признаков Хаара

После построения интегрального изображения начинается этап извлечения лицевых признаков [2]. Для построения признаков используются признаки Хаара – прямоугольные шаблоны, представляющие разницу интенсивностей между светлыми и тёмными областями изображения (рисунок 1). Данные признаки позволяют эффективно выявлять характерные элементы, такие как края и текстуры объекта на изображении.

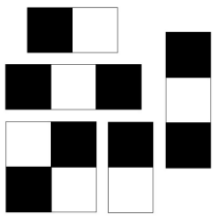


Рисунок 1. Признаки Хаара

Признак вычисляется как разность суммарных интенсивностей светлой и тёмной областей. Вычисление признака показано в формуле (4):

$$h(x,y,\omega,h) = \sum_{white} I(x,y) - \sum_{black} I(x,y)\,, \tag{4}$$

где ω и h - ширина и высота признака соответственно, (x, y) – координаты, задающие положение признака.

Полученная разность отражает различие в интенсивности пикселей между двумя областями, что позволяет выявить характерные особенности, такие как контуры и текстуры объекта на изображении (рисунок 2).

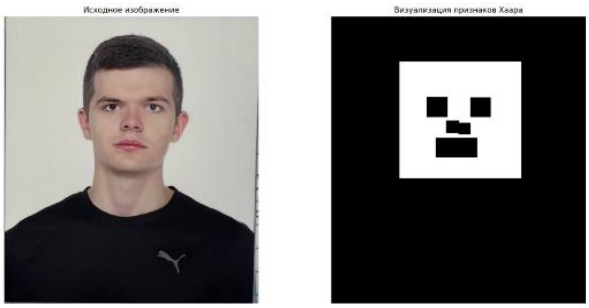


Рисунок 2. Определение признаков Хаара на изображении

Обучение признаков и отбор

В процессе обучения на этапе подготовки модели отбираются наиболее информативные Хаар-признаки из всего набора. Данная процедура позволяет минимизировать вычислительные затраты, используя только признаки, лучше всего различающие целевой объект и фон.

Для отбора признаков, наиболее эффективно классифицирующие изображения, используется алгоритм **AdaBoost** (adaptive boosting) [3].

Постобработка результата в каскадах Хаара

Из-за особенностей работы алгоритма каждая область изображения проходит проверку на множестве масштабов, что приводит к множественному детектированию одного и того же объекта в соседних или совпадающих областях.

Для устранения избыточных детекций используется метод объединения областей [3].

Каждое обнаружение объекта представляется в виде прямоугольника R_i, заданного угловыми координатами (x_1, y_1) и (x_2, y_2). Для всех перекрывающихся

прямоугольников вычисляется мера перекрытия IoU (Intersection over Union), представленная в формуле (5):

$$IoU(R_i, R_j) = \frac{Area(R_i \cap R_j)}{Area(R_i \cup R_j)}, \quad (5)$$

где $Area(R_i \cap R_j)$ – площадь пересечения двух прямоугольников R_i и R_j, $Area(R_i \cup R_j)$ – площадь объединения двух прямоугольников.

Если значение IoU превышает заданный порог τ (например, $\tau = 0.5$), прямоугольники объединяются, образуя одну итоговую область. Результатом является упрощённая структура, где каждая обнаруженная область уникальна и соответствует одному объекту.

Разработка приложения сортировки изображений согласно пользовательским критериям

Данное приложение должно реализовывать следующий функционал:

- Аутентификация при помощи Google Auth API.
- Предоставление доступа к Google Drive.
- Редактирование и просмотр содержимого каталогов Google Drive.
- Загрузка и скачивание файлов.
- Выбор критериев сортировки и группировки:
  - Группировка по лицам.
  - Группировка по местоположению изображений.
  - Группировка и сортировка по дате съёмки.
  - Группировка по метаданным.

Архитектура приложения

Архитектура приложения построена на клиент-серверной модели с использованием современных подходов для мобильной разработки.

Клиентская часть будет выполнена по архитектуре MVVM (Model-View-ViewModel – Модель-Представление-Модель представления). MVVM разделяет код на три ключевых компонента для обеспечения читаемости, тестируемости и масштабируемости.

Компонент Model отвечает за данные, получаемые от сервера, получаемые с Google Drive, а также отправляемые на Google Drive и Сервер. View отвечает за отображение пользовательского интерфейса, а именно:

- Экрана авторизации.
- Профиля пользователя.
- Каталога Google Drive и его содержимого.
- Окна параметров сортировки.
- Окна выбора загружаемых изображений.

ViewModel отвечает за логику взаимодействия между моделью и представлением.

Коммуникация между клиентской и серверной частями может быть реализована несколькими путями:

- Постоянным TCP соединением.
- Постоянным SSL соединением.
- Непостоянным HTTP соединением.
- Непостоянным HTTPS соединением [4].

При коммуникации, передаваемые данные от клиентской части содержат в себе:

- Личный токен доступа к Google.Drive.
- Тип группировки
- Массив идентификаторов изображений, которые участвуют в группировке.

На серверной части происходят все вычисления, а именно: обнаружение лиц на изображениях и сравнение изображений по обнаруженным лицами при помощи моделей глубокого обучения., сверточные нейронные сети (рисунок 3). В качестве модели обнаружения лиц выбрана модель на основе архитектуры YOLO (You Only Look Once – Ты смотришь только единожды), так как данная архитектура предлагает:

- Унифицированную систему обнаружения.
- Обнаружение в режиме реального времени.

•Компромисс между точностью и скоростью [5].

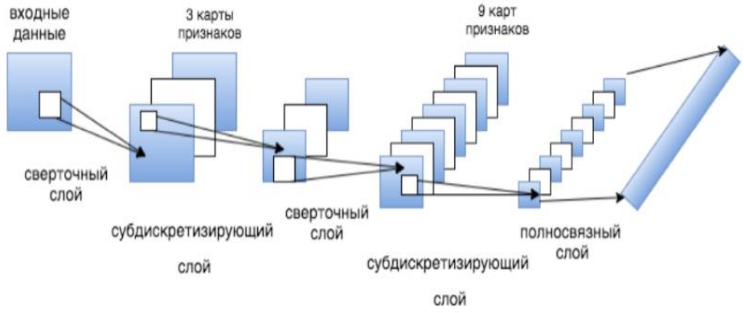


Рисунок 3. Архитектура сверточной нейронной сети

Выбор платформы и языка приложения

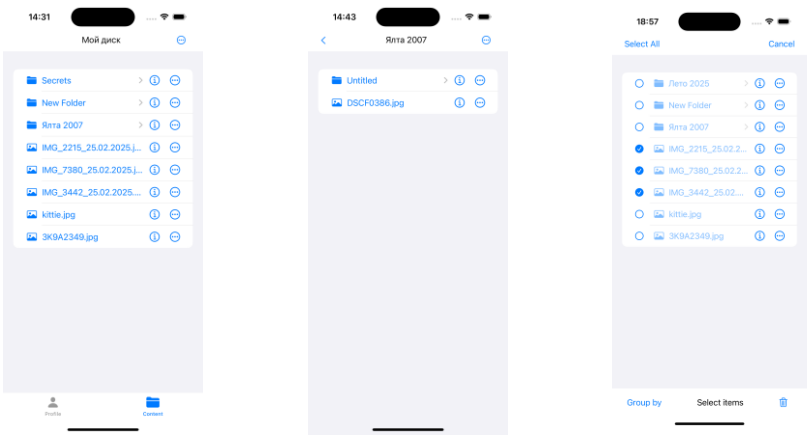
Платформой приложения была выбрана операционная система iOS.

Для разработки приложений для операционной системы iOS в используется язык программирования Swift.

Для разработки серверной части приложения был выбран язык программирования Python.

Демонстрация приложения

Главный экран представляет собой список содержимого Google.Drive (рисунок 4). Пользователю доступен следующий функционал: переход в дочерние папки; просмотр фотографий; просмотр метаданных файла; удаление, переименование файлов; а также выбор файлов, которые будут участвовать в группировке.



а) б) в)

Рисунок 4. – Главный экран: а – представление содержимого корневой папки, б – представление содержимого папки «Ялта 2007», в – выбор фотографий, которые необходимо группировать.

После завершения группировки, фотографии находятся в отдельной папке под названием «Person 1». Пример содержимого данной папки, представлен на рисунке 5.

Рисунок 5. Фотографии, участвовавшие в группировке: IMG_2215_25.02.2025.jpg, IMG_3442_25.02.2025.jpg, IMG_7380_25.02.2025.jpg

Литература

1. Р.Гонсалес, Р.Вудс, «Цифровая обработка изображений», ISBN 5-94836-028-8, изд-во: Техносфера, Москва, 2005. – 1072 с.

2. Местецкий Л. М., «Математические методы распознавания образов», МГУ, ВМиК, Москва, 2002–2004., с. 42 – 44

3. J. Kurose, K. Rouse, “Computer Networking. A Top-Down Approach” (8th ed.), 2021. – 800p.

4. J. Redmon, S. Divvala, R. Girshick, A. Farhadi «You Only Look Once: Unified, Real-Time Object Detection», p. 1 – 10.

5. F. Chollet, “Deep Learning with Python”, 2017, – 384 p.