Разработка приложения «умная библиотека» с использованием Intel Distribution of OpenVINO toolkit

Е.П. Васильев[1], В.Д. Кустикова[1], И.Б. Вихрев[1], К.Д. Уткин[1], А.В. Дудченко[1]
eugene.unn@gmail.com|valentina.kustikova@gmail.com|laind4471@gmail.com|megarungle@gmail.com|da394372@gmail.com
[1]Нижегородский государственный университет им. Н.И. Лобачевского, Нижний Новгород, Россия

Рассматривается разработка автоматизированной "умной библиотеки" с использованием глубокого обучения и алгоритмов компьютерного зрения на основе открытой библиотеки методов искусственного интеллекта OpenVINO toolkit. Общая схема работы приложения предполагает регистрацию читателя – добавление информации и фото нового пользователя; обновление модели машинного обучения, описывающей особенности лиц пользователей системы; авторизацию читателя посредством распознавания лица; получение и возврат книг посредством сопоставления изображения обложки с базой плоских изображений, доступных в библиотеке книг. Исходный код приложения выложен в открытый доступ на GitHub: https://github.com/itlab-vision/openvino-smart-library. Разработанное приложение планируется опубликовать в составе пакета примеров OpenVINO toolkit.

Ключевые слова: *распознавание лиц, ИИ, компьютерное зрение, глубокое обучение, машинное обучение, умная библиотека, OpenVINO.*

Development of the "smart library" application using the Intel Distribution of OpenVINO toolkit

E.P. Vasiliev[1], V.D. Kustikova[1], I.B. Vikhrev[1], K.D. Utkin[1], A.V. Dudchenko[1]
eugene.unn@gmail.com|valentina.kustikova@gmail.com|laind4471@gmail.com|megarungle@gmail.com|da394372@gmail.com
[1]Lobachevsky State University of Nizhni Novgorod, Nizhny Novgorod, Russia

We represent a case study of using deep learning and computer vision library - the Intel Distribution of OpenVINO toolkit. We develop the automated "smart library" using DL and computer vision methods implemented in OpenVINO toolkit. The application involves the registration of the reader (adding information and photos of the new user); updating the machine learning model that describes the face features of the library users; authorization of the reader through face recognition; receiving and returning books by comparing the cover image with the database of flat images available in the library of books. The source code of the application is free available on GitHub: https://github.com/itlab-vision/openvino-smart-library. The developed application is planned to be published as a sample of the OpenVINO toolkit.

Keywords: *face recognition, AI, computer vision, deep learning, machine learning, smart library, OpenVINO.*

1. Введение

В настоящее время наблюдается повышенный интерес к области интернета вещей (Internet of Things, IoT). С развитием концепции граничных вычислений (edge-computing) устройства интернета вещей становятся все более "умными" за счет возможности применения машинного обучения и глубоких нейронных сетей [2, 10]. Рост производительности конечных устройств и появление специализированного аппаратного обеспечения для запуска нейронных сетей способствует расширению множества решаемых задач, в которых невозможны длительные задержки при обработке данных. К таковым, в частности, относятся задачи, возникающие при разработке систем безопасности, автопилотов и медицинских приборов [8].

Создание приложений видеоанализа с применением методов компьютерного зрения, машинного и глубокого обучения является трудоемким с точки зрения программной реализации, тестирования и интегрирования в существующие системы. Intel Distribution of OpenVINO toolkit – набор библиотек, средств оптимизации и информационных ресурсов для разработки программного обеспечения, использующего машинное зрение и глубокое обучение. OpenVINO предназначен для ускорения процесса создания систем компьютерного зрения и оптимизации вычислений под разнообразные аппаратные платформы компании Intel (Intel CPUs, Intel Processor Graphics, Intel Vision Processing Units, Intel FPGAs и Intel Gaussian Mixture Model).

Цель данной работы состоит в том, чтобы продемонстрировать пример решения задачи из области интернета вещей с использованием OpenVINO. Работа построена следующим образом. Вначале дается краткий обзор возможностей OpenVINO. Далее ставится задача разработки приложения "умная библиотека", анализируются требования и рамки приложения. Разрабатывается архитектура приложения, приводится общая схема его функционирования, описывается программная реализация. Рассматривается пример использования приложения, анализируются показатели производительности и качества работы системы.

2. Intel Distribution of OpenVINO toolkit

OpenVINO [5] состоит из нескольких основных частей.

1. ***Deep learning for computer vision.*** В состав входит инструмент Deep learning deployment toolkit для эффективного применения предварительно обученных глубоких нейронных сетей с использованием высокоуровневого программного интерфейса.
2. ***Traditional computer vision.*** Обеспечивает поддержку разработки и оптимизации приложений компьютерного зрения, реализованных с использованием библиотеки OpenCV [6] или программного интерфейса OpenVX [7].
3. ***Additional packages*** для Intel FPGAs, Intel Movidius Neural Compute Stick, Intel Gaussian Mixture Model, а также функции кодирования и декодирования медиафайлов.

3. Разработка требований к приложению

Выделяется две категории пользователей приложения «умная библиотека» с разным уровнем привилегий.

1. ***Читатель.*** Читателю предоставляется возможности регистрации в библиотеке посредством получения изображения лица и внесения личных данных, входа в личный кабинет с помощью распознавания лица, получения и возврата книг.
2. ***Администратор.*** При наделении читателя правами администратора дополнительно появляются возможности получения информации о всех читателях и всех книгах в библиотеке, получения информации о книгах, выданных читателям, пополнения библиотеки новой книгой.

Таким образом, необходимо разработать приложение, обеспечивающее поддержку приведенных типов пользователей и реализующее требуемый функционал.

4. Архитектура приложения

Приложение состоит из нескольких основных компонент (рис. 1).

1. ***Model.*** Данный компонент содержит основные алгоритмы, используемые в процессе разработки системы. В состав компонента входят два основных модуля: BookRecognition и FaceRecognition. Модуль BookRecognition обеспечивает распознавание книг посредством сопоставления (matching) плоских объектов. В качестве дескриптора ключевых точек используется ORB [3] (также возможно использование SIFT или SURF). Модуль FaceRecognition обеспечивает распознавание лиц с использованием библиотеки Photography Vision Library (PVL), содержащей реализацию современных алгоритмов компьютерного зрения для поиска лиц. Библиотека разработана для оптимизированного выполнения на процессорах Intel [4, 11]. При регистрации нового пользователя модель дополняется новым классом лиц, при входе пользователя выполняется распознавание и идентификация пользователя. Остальные модули, входящие в состав компонента и приведенные на диаграмме, являются служебными.
2. ***Infrastructure.*** Компонент отвечает за организацию доступа к данным. Он предоставляет набор интерфейсов для доступа к связанным таблицам, содержащим информацию о пользователях и книгах. Схема базы данных доступна в репозитории проекта на GitHub [9].
3. ***GUI.*** Данный компонент содержит реализацию графического интерфейса. В базовой версии интерфейс состоит из нескольких графических окон: окно регистрации нового читателя, окно входа в личный кабинет читателя/администратора, окно личного кабинета читателя/администратора.
4. ***Application.*** Компонент обеспечивает создание всех объектов приложения и координирует их работу.

Общая схема взаимодействия компонент “умной библиотеки” выглядит следующим образом.

1. ***Application*** создает рабочие объекты системы, включая объекты компонента ***GUI***.
2. Пользователь инициирует действие на ***GUI***.
3. ***GUI*** обращается к компонентам ***Model*** и ***Infrastructure*** для обработки инициированного действия.

5. Программная реализация

Приложение “умная библиотека” реализуется на языке Python 3 с применением инструментов, входящих в состав OpenVINO, в частности, библиотек OpenCV [6] и PVL [11].

OpenCV – широко известная библиотека алгоритмов компьютерного зрения. При разработке приложения используется реализация методов сопоставления плоских объектов и выделения дескрипторов, входящие в состав модуля 2D Features Framework [1], являющегося составной частью OpenCV.

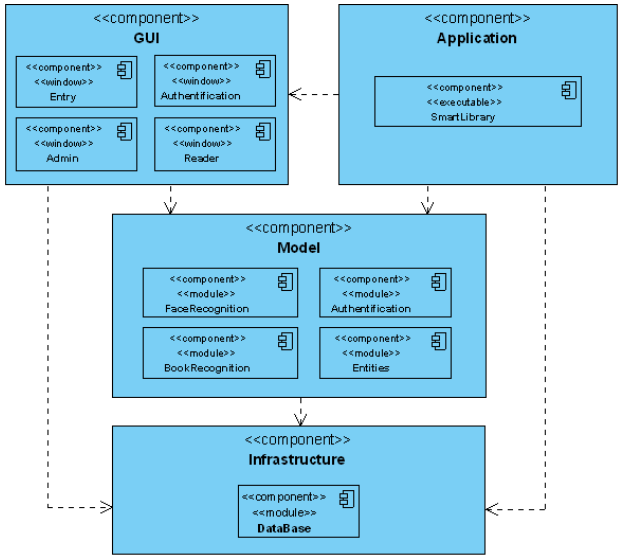

Рис. 1. Архитектура приложения.

PVL является частью OpenVINO используется для быстрого и качественного поиска лиц. Данная библиотека не имеет открытого исходного кода. Программный интерфейс PVL разработан на языке C++, поэтому функции для работы с PVL реализованы на C++. Вызов функций из кода на языке Python обеспечивается при помощи модуля ctypes. Для построения графического интерфейса используется библиотека PyQt [12].

Информация о пользователях и книгах хранится в виде csv-таблиц, работа с которыми организована посредством использования стандартных пакетов Python.

Исходный код приложения выложен в открытый доступ на GitHub [9].

6. Эксперименты

6.1 Тестовая инфраструктура

В качестве тестовой системы используется ноутбук MSI со следующими характеристиками:

- CPU: Intel Core i7-4710HQ, 2.50GHz, 4 cores, 8 threads.
- RAM: 16Gb.
- OS: Windows 8.1.
- Библиотеки: Intel Distribution of OpenVINO toolkit 2018 R5.0.1

6.2 Распознавание лиц

Данные. Для измерения производительности подготовлен набор видео с 6 пользователями. Каждое видео соответствует пользователю, снятому в определенных условиях: темная/светлая одежда, хорошее/плохое освещение лица, удаленность от камеры (рис. 2). Всего 12 видео на пользователя длительностью 2 секунды каждое.

Показатели качества и производительности. Классификатор обучается на различном количестве изображений лица человека из видео от 1 до 8. В процессе обучения классификатора определяется ***среднее время обновления классификационной модели***, которое отражает среднее время регистрации нового пользователя в системе. Также определяется ***среднее количество пользователей, обрабатываемых за секунду FPS (register)***.

В процессе тестирования вычисляется ***среднее время детектирования лица*** и ***среднее количество кадров, обрабатываемых классификатором за секунду FPS***

(recognition). Данный показатель отражает скорость распознавания пользователя.

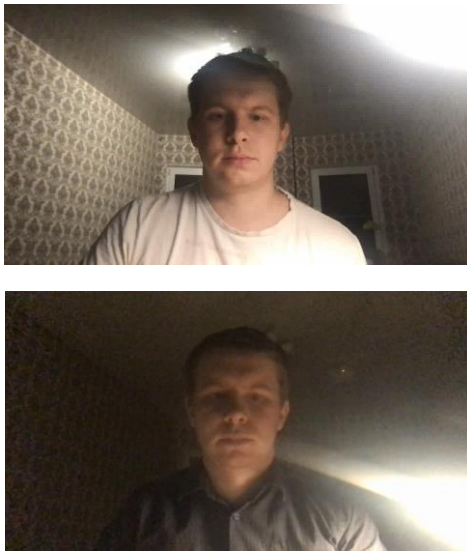

Рис. 2. Пример лица при хорошем и плохом освещении.

Точность распознавания лица вычисляется по формуле:

$$p = \frac{\mathrm{N_{true}}}{\mathrm{N_{all}}}$$

где $\mathrm{N_{true}}$ - количество правильно распознанных лиц к общему количеству изображений, которые были поданы на вход классификатору.

Результаты. Ниже (таблица 1) приведены полученные результаты качества и производительности детектирования лиц. Мы видим что решение быстро работает на обычном ноутбучном процессоре без привлечения графических ускорителей, главным ограничением системы является максимальное количество запоминаемых лиц в 32 пользователя. При проведении тестирования классификатор распознавал лица с высокой степенью точности. Отдельно стоить отметить, что во всех случаях ошибок классификатора он возвращал результат о том, что лицо не известно, во время тестирования не возникало ситуаций, что один пользователь был признан за другого, возможно для детектирования таких ошибок требуется значительно увеличить тестировочную базу.

Распознавание лиц при помощи библиотеки PVL не требует мощной вычислительной системы, установка и настройка программы является простой операцией, что позволяет быстро добавлять качественное распознавание лиц в уже существующее программное обеспечение.

Таблица 2. Средние показатели FPS и точности предсказания в зависимости от количества изображений каждого пользователя для обучения детектора.

Количество фотографий лица для обучения	1	2	4	8
Среднее время регистрации лица, с	0.042	0.048	0.059	0.078
FPS (register)	21.59	18.90	15.18	12.42
Среднее время детектирования лица, с	0.0064	0.0070	0.0079	0.0083
FPS (recognition)	146.43	135.15	119.2	112.41
Точность	0.915	0.925	0.928	0.980

Также измерения скорости были произведены на данных из датасета YouTube Faces DB [13]. Данный набор является достаточно сложным для алгоритма классификации лиц, поскольку на значительной части изображений люди находятся далеко от камеры, с плохим освещением, а также сами изображения низкого разрешения (320*240 или 480*360). Библиотека PVL не всегда могла найти лица на изображении, поэтому для измерения производительности были выбраны первые 32 видео, на которых библиотека PVL распознала лицо.

Таблица 2. Средние показатели FPS и точности предсказания в зависимости от количества изображений пользователя для обучения детектора наборе данных YouTube Faces DB.

Количество фотографий лица для обучения	1	2	4	8
Среднее время регистрации лица, с	0.039	0.053	0.079	0.136
FPS (register)	24.32	19.45	13.39	7.28
Среднее время детектирования лица, с	0.0052	0.0055	0.0057	0.0061
FPS (recognition)	194.01	177.45	172.76	156.65
Точность	0.7685	0.784	0.784	0.784

6.3 Распознавание книг

Данные. Для измерения производительности собран набор данных из 5 книг (плоское изображение обложки + фронтальное видео обложки). Фотография обложки представляет собой изображение с разрешением 3120×4160 пикселей (рис. 3). Видео обложки записано в разрешении 1920×1080 пикселей.

Показатели качества и производительности. В процессе работы классификатора на фотографии вычисляются ключевые точки и их дескрипторы при помощи алгоритма ORB, в таблице приводится среднее время, за которое происходит вычисление данных признаков для эталонных изображений 3120×4160 (время, затраченное на чтение изображений и видео из файлов не учитывается).

Рис. 2. Пример плоских изображений книг.

В процессе тестирования вычисляется среднее время, за которое будет произведено вычисление особых точек на кадре видео 1920×1080. Далее приводится время, затраченное на сопоставление ключевых признаков шаблонного изображения и кадра из видео при помощи алгоритма сопоставления BFMatcher, входящего в состав библиотеки OpenCV. Последней строкой таблицы приводится среднее время, которое требуется для распознавания книги (вычисление признаков + сопоставление со всеми шаблонами) по одному видео (состоящему из 40 кадров).

Результаты. Ниже (таблица 2) приведены полученные результаты производительности сопоставления. Из данной

таблицы можно увидеть, время одной операции сравнения сопоставления эталонного изображения и кадра видео очень маленькое, но распознавание книги занимает много времени. Зависимость времени распознавания книги от количества эталонных изображений книг в базе является линейной, поэтому при большом количестве изображений распознавание будет занимать большее время, при таком подходе трудно повысить отзывчивость системы.

Поскольку для распознавания используется детектирование и сравнение ключевых точек, то это накладывает ограничение на книги, обложки должны быть контрастными, чтобы детекторы точек могли найти находить на изображении книги ключевые точки.

Алгоритм распознавания книг, основанный на вычислении ключевых точек при помощи детектора ORB и сравнении с шаблонами, работает очень медленно на больших изображениях. Для распознавания книг перспективным кажутся алгоритмы, основанные не поиске ключевых точек, а основанные на сравнении изображений, например перцептивный хэш.

Таблица 2. Средние показатели производительности и точности предсказания книг.

Среднее время расчета признаков на эталонном изображении книги, с	0.120
Среднее время расчета признаков по видео, с	0.023
Среднее время одной операции сопоставления шаблонов, с	0.003
Среднее время сопоставления и выбора правильной книги, с	0.719

У алгоритма распознавания книг, работающего на поиске ключевых точек, фундаментальный недостаток в том, что система не может распознать несколько экземпляров одной и той же книги. Для распознавания конкретных экземпляров предлагается использовать систему штрихкодов или QR-кодов, а алгоритм поиска ключевых точек использовать как дополнительный шаг, например решающий проблему копирования штрихкода и нанесения на другую книгу.

7. Заключение

В ходе исследования разработано программное приложение “умная библиотека” на основе инструмента OpenVINO. Отличительной особенностью приложения является детектирование и распознавание пользователей и книг с применением алгоритмов машинного обучения и компьютерного зрения. Приведены результаты анализа производительности модулей системы в задачах распознавания лиц пользователей и книг. Исходный код приложения выложен в открытый доступ на GitHub [9].

8. Благодарности

Работа выполнена при поддержке компании Intel. Авторы благодарят сотрудников компании за внимание к работе.

9. Литература

[1] 2D Features Framework. OpenCV documentation. URL: https://docs.opencv.org/4.1.0/da/d9b/group__features2d.html

[2] Bharath Raj. Deep Learning on the Edge. URL: https://www.kdnuggets.com/2018/09/deep-learning-edge.html.

[3] Ethan Rublee, Vincent Rabaud, Kurt Konolige, Gary R. Bradski: ORB: An efficient alternative to SIFT or SURF. ICCV 2011: 2564-2571

[4] Intel Computer Vision SDK Developer Guide. URL: https://www.codeproject.com/Articles/1212075/Intel-Computer-Vision-SDK-Developer-Guide

[5] Intel Distribution of OpenVINO Toolkit. URL: https://software.intel.com/en-us/openvino-toolkit

[6] OpenCV (Open Source Computer Vision Library). URL: https://opencv.org

[7] OpenVX. Portable, Power-efficient Vision Processing. URL: https://www.khronos.org/openvx

[8] Satyanarayanan, M. (2017). The Emergence of Edge Computing. Computer, 50(1), 30–39. doi:10.1109/mc.2017.9

[9] Smart library based on OpenVINO toolkit. URL: https://github.com/itlab-vision/openvino-smart-library

[10] This Affordable Device From Intel Brings Computer Vision To IoT And Edge Computing. URL: https://www.forbes.com/sites/janakirammsv/2018/11/19/this-affordable-device-from-intel-brings-computer-vision-to-iot-and-edge-computing/

[11] Using the Intel OpenVINO Photography Vision Library. URL: https://github.com/hybridgroup/gocv/tree/master/openvino/pvl

[12] PyQt Documentation. URL: https://wiki.python.org/moin/PyQt

[13] Wolf Lior, Tal Hassner and Itay Maoz. Face Recognition in Unconstrained Videos with Matched Background Similarity. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2011.