МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ
Кафедра технологий программирования

РОМАНЮК
Вадим Николаевич

РАЗРАБОТКА МАСШТАБИРУЕМОЙ ВЫСОКОНАГРУЖЕННОЙ СИСТЕМЫ ХРАНЕНИЯ И ОБРАБОТКИ ВИДЕОДАННЫХ

Дипломная работа

Научный руководитель:
кандидат физико - математических наук,
доцент В.И. Дравица

Допущена к защите
«____»_____2021 г.
Зав. кафедрой технологий программирования
доктор технических наук, профессор,
заслуженный деятель науки РБ А. Н. Курбацкий

Минск, 2021

ОГЛАВЛЕНИЕ

РЕФЕРАТ

Дипломная работа, 64 с., 38 рис., 5 формул, 1 таблица, 7 приложений, 15 источников.

Ключевые слова: ВИДЕОНАБЛЮДЕНИЕ, ВИДЕОАНАЛИТИКА, MICROSOFT AZURE, МЕТОД ВИОЛЫ – ДЖОНСА, CNN, HOG, MICROSOFT COGNITIVE SERVICES, AZURE FUNCTIONS, AZURE SQL, JAVA, PYTHON, EDGE COMPUTING.

Объект исследования – объектом исследования является технология видеоаналитики. Предмет исследования – разработка системы для распознавания и идентификации лиц на изображениях.

Цель работы – изучение технологий облачных и граничных вычислений и разработка масштабируемой высоконагруженной системы хранения и обработки видеоданных.

Методы исследования – а) теоретические: изучение существующих систем и комплексов видеоаналитики и компонентов Microsoft Azure, рассмотрение современных алгоритмов и средств по обработке видеопотока; б) практические: сравнение изученных алгоритмов и их последующее применение при разработке системы, проектирование и создание программы по сбору, хранения и чтения видеоданных с видеокамеры и приложения – триггера для обработки видеоданных с последующим сохранением результатов в облачном хранилище.

В результате – спроектирована и реализована система, способная производить распознавание и идентификацию людей на изображениях с последующем сохранением результатов обработки в облачном хранилище.

Область применения – на сегодняшний день система может быть использована в мелких и крупных заведениях для обеспечения безопасности и контроля.

РЭФЕРАТ

Дыпломная праца, 64 ст., 38 мал., 1 табліца, 5 формул, 7 дадаткаў, 15 крыніц.

Ключавыя словы: ВІДЭАНАЗІРАННЕ, ВІДЭААНАЛІТЫКА, MICROSOFT AZURE, МЕТАД ВІЁЛЫ – ДЖОНСА, CNN, HOG, MICROSOFT COGNITIVE SERVICES, AZURE FUNCTIONS, AZURE SQL, JAVA, PYTHON, EDGE COMPUTING.

Аб'ект даследавання – аб'ектам даследавання з'яўляецца тэхналогія відэаналітыкі. Прадмет даследавання – распрацоўка сістэмы для распазнання і ідэнтыфікацыі твараў асоб на малюнках.

Мэта працы – вывучэнне тэхналогій воблачных і межавых вылічэнняў і распрацоўка маштабуемай высоканагружанай сістэмы захоўвання і апрацоўкі відэададзеных.

Метады даследавання – а) тэарэтычныя: вывучэнне існуючых сістэм і комплексаў відэааналітыкі і кампанентаў Microsoft Azure, разгляд сучасных алгарытмаў і сродкаў па апрацоўцы відэапатоку; б) практычныя: параўнанне вывучанных алгарытмаў і іх наступнае прымяненне пры распрацоўцы сістэмы, праектаванне і стварэнне праграммы для збора, захоўвання і чытання відэададзеных з відэакамеры і праграмы – трыгера для апрацоўкі відэададзеных з наступным захаваннем вынікаў у воблачным сховішчы.

У выніку – спраектавана і рэалізавана сістэма, здольная вырабляць распазнанне і ідэнтыфікацыю асоб на малюнках з наступным захаваннем вынікаў апрацоўкі ў воблачным сховішчы.

Вобласць прымянення – на сённяшні дзень сістэма рэкамендуецца для выкарыстання на маленькіх і вялікіх прадпрыемствах для забеспячэння бяспекі і кантролю.

ABSTRACT

Diploma Thesis, 64 p., 38 img., 5 formulas, 1 table, 7 appendixes, 15 sources.

Keywords: VIDEO SURVEILLANCE, VIDEO ANALYTICS, MICROSOFT AZURE, VIOLA – JONES METHOD, CNN, HOG, MICROSOFT COGNITIVE SERVICES, AZURE FUNCTIONS, AZURE SQL, JAVA, PYTHON, EDGE COMPUTING.

The object of research is video analytics technology. The subject of the study is development the system for recognition and identification faces in the images.

The purpose is studying cloud and edge computing technologies and developing a scalable highload video data storage and processing system.

Methods of research are a) theoretical methods: study of existing systems and complexes of video analytics and Microsoft Azure components, review of modern algorithms and facilities for processing video stream; b) practical methods: comparison of the studied algorithms and their subsequent applying in the development of the system, design and creation a program for collecting, storing and reading video data from a video camera and a trigger application for processing video data with subsequent storing the results in the cloud storage.

As a result – designed and created a system that can recognize and identify people in the images with subsequent storing the results in the cloud storage.

Scope – today the system can be used in small and large institutions to ensure security and control.

ВВЕДЕНИЕ

На сегодняшний день отрасль видеонаблюдения широко распространена и применяется как для личных, так и профессиональных целей. Обеспечение безопасности объектов – одна из основных целей применения камер видеонаблюдения. Однако слишком большое количество камер усложняет работу оператору, а порой она может стать невыполнимой. Например, один оператор не справится с контролем сотен камер, так что приходится нанимать целый штат сотрудников. В подобных ситуациях на помощь может прийти видеоаналитика.

Видеоаналитика представляет собой интеллектуальный анализ данных, принятых от камеры в виде совокупности видеоизображений или потока видеоданных, и автоматическое детектирование заранее запрограммированных событий. Аналитика может проводиться как в режиме реального времени (онлайн-режим), так и при работе с архивом (офлайн-режим). Результаты работы – события, которые можно передать оператору в виде сообщений или записать в архив для последующего ведения поиска по ним и составления отчетов. В видеоаналитике выделяют несколько самостоятельных направлений, среди которых можно выделить базовые и сервисные детекторы.

Базовые и сервисные детекторы – одни из самых простых и эффективных детекторов движения, смещения фокуса, засветки, закрытия камеры, которые на сегодняшний день либо присутствуют почти в каждом программном обеспечении видеонаблюдения, либо при необходимости могут быть легко встроены в функционал камеры. Детектор распознавания – пример базового детектора. Основная его цель – распознать лицо человека при появлении его в кадре, отличая его от других объектов, проводить поиск в архивах, содержащих большие объемы информации.

Современная качественная система видеонаблюдения должна уметь проводить ряд аналитических функций, а не только записывать и выводить на экран изображение. К таким функциям относится распознавание и идентификация людей (лиц), которая широко применяется в следующих случаях:

- Противодействие кражам;
- Система контроля управления доступом (СКУД);
- Пропускной контроль в ночных клубах и различных развлекательных заведениях.

В случаях с кражами классическая система видеонаблюдения, которая установлена в магазине, обычно используется для получения доказательств, когда ущерб уже был нанесен. В большинстве случаев мелкие преступления

которые приносят заведениям не только финансовый урон, но и снижают их репутацию, совершают одни и те же люди. Создание базы данных, содержащей лица этих людей, и использование средств видеоаналитики помогут выявить злоумышленников еще при входе в магазин и более тщательнее контролировать их действия.

В случаях со СКУД аналитика может быть интегрирована в систему безопасности и управления контроллерами на турникетах. Преимущество такого подхода – минимизация человеческого фактора и повышение трудовой дисциплины.

В случаях с развлекательными клубами использование системы видеоаналитики с выводом тревожного оповещения на удаленное устройство начальника клуба (например, на телефон или планшет) поможет понизить злоупотребления персонала.

В главе 1 анализируются современные системы и комплексы видеоаналитики, рассмотрены технологии облачных и граничных вычислений.

В главе 2 приведены средства и методы обработки видеопотока, в том числе алгоритмы и средства для идентификации лиц на изображениях.

В главе 3 представлена разработка системы по обработке видеопотока с видеокамеры, встроенной в ноутбук, и программы-триггера, идентифицирующей лица на изображениях и сохраняющей результаты в облачное хранилище.

ГЛАВА 1. МОНИТОРИНГ СУЩЕСТВУЮЩИХ СИСТЕМ ВИДЕОАНАЛИТИКИ И КОНЦЕПЦИИ ОБЛАЧНЫХ И ГРАНИЧНЫХ ВЫЧИСЛЕНИЙ

1.1 Обзор существующих систем и комплексов видеоаналитики

1.1.1 Видеоаналитический комплекс CasRetail

CasRetail – видеоаналитический комплекс, который разработан для экономического, маркетингового контроля и анализа. CasRetail – масштабируемое решение для сложных объектов, которое работает без ограничения количества каналов и поддерживает виртуализацию. Данный продукт подходит как для небольших одиночных магазинов, так и для крупных торгово-развлекательных центров. Среди полезных новшеств системы можно отметить следующие:

- работа с видеоархивом, используя личный кабинет пользователя;
- создание карты траектории движений и тепловой карты, которые показывают наиболее популярные зоны и витрины покупателей;
- операционная система Linux, которая позволяет значительно сократить расходы на интеграцию и повышает отказоустойчивость.

Просмотр видеоархива можно осуществлять онлайн, используя любой веб-браузер, что позволяет контролировать работу персонала и следить за происходящим на объекте. При этом не требуется использовать какое-либо постороннее программное обеспечение. За счет технологии структурированного покадрового представления видео можно быстро оценить текущую обстановку в торговой точке без предварительного скачивания архива.

Также среди функций системы можно выделить: оптимизация нагрузки персонала, анализ и планирование рекламных акций, сбор данных для прогнозирования прибыли. Например, с помощью анализа поведения в очереди можно оптимизировать работу кассиров. Два основных показателя – размер очереди и время, которое проводят в ней покупатели. В основе анализа положен детектор, который распознает головы людей, при этом все остальные объекты игнорируются. Стоит отметить, что при окончательном подсчете людей в очереди из статистики также исключаются дети.

1.1.2 Система идентификации VOCORD FaceControl

VOCORD FaceControl – система идентификации человека по лицу. Данное решение успешно применяется в местах массового скопления людей – на вокзалах, стадионах, аэропортах, метро и других публичных местах, где люди не взаимодействуют с системой биометрической идентификации.

Главное преимущество системы – работа в некооперативном режиме. Это говорит о том, что системе не нужно напрямую взаимодействовать с человеком, чтобы его распознать. Принцип работы продукта достаточно прост. Рубежи контроля снабжают специализированными камерами для распознавания лиц. Пока человек проходит через зону контроля, система делает несколько снимков его лица, из которых для последующего анализа отбираются наилучшие. Затем система сравнивает выделенные лица на фотографиях с лицами в эталонных базах данных и распознает их онлайн (то есть в режиме реального времени). Все результаты анализа сохраняются в специальном архиве, по которому затем можно легко вести поиск и делать отчеты.

Данное решение также интегрируют в системы контроля управления доступом для повышения эффективности и широко используют в магазинах. В ритейле продукт компании VOCORD позволяет составлять «черные» и «белые» списки клиентов, благодаря чему персонал будет заранее оповещен о появлении в заведении VIP-клиента или же наоборот – о потенциальном злоумышленнике.

Продукт VOCORD FaceControl снабжен самыми современными алгоритмами распознавания лиц, которые эффективно работают даже при плохих погодных условиях и недостаточном освещении, что способствует высокой достоверности распознавания. Вероятность корректного распознавания в «полевых» условиях в среднем составляет 90%. К системе распознавания можно подключать неограниченное количество баз данных. Если в кадре был обнаружен преступник, то система выдаст оператору уведомление с пометкой, где обнаружен преступник, и комментарии, если такие были добавлены изначально.

1.1.3 Система безопасности VideoNet

VideoNet – интеллектуальная цифровая система безопасности от компании Skyros, предназначенная для автоматизации процесса видеонаблюдения и охраны объектов любой сложности и масштаба. Система позволяет как фиксировать события, происходящие на охраняемом объекте, так и выявлять потенциально опасные ситуации за счет инновационных алгоритмов обработки видеопотока и эффективных средств видеоаналитики, лежащих в основе продукта.

Данное решение успешно функционирует на многочисленных объектах, среди которых можно выделить объекты промышленного и военного назначения, банки и финансовые учреждения, таможенные и пограничные терминалы, крупные торговые центры и объекты транспортной безопасности.

Основные достоинства системы:

- *Поддержка различных IP-устройств.* В системе используется платформа Total.IP, позволяющая обеспечить эффективную и оптимальную работу с различными IP-устройствами (поддерживаются более 3000 IP-устройств различных мировых производителей);
- *Видеоизображения высочайшего качества*. Алгоритм компрессии DVPack 2 позволяет получить высокую степень компрессии (меньший размер кадра при том же визуальном качестве) и качество изображения, повысить быстродействие [5]. Данный подход позволяет экономить на оборудовании, необходимом для организации видеоархива, и на существующей конфигурации получить большую глубину архива;
- *Широкие возможности интеграции с другими системами.* Например, с системами безопасности Orion Pro и Alpha, системой распознавания автомобильных номеров «Поток», системой распознавания лиц Viisage (ZN Vision) и многими другими;
- *Быстрый и надежный архив.* Решение позволяет производить запись до 1000 кадров полнокадрового видео в режиме реального времени, быстрый поиск информации в архиве и перемотку архивной информации в «обе стороны». Действует система предотвращения потерь кадров в случаях пиковых нагрузок или некорректного завершения работы VideoNet;
- *Удобный интерфейс.* Каждый пользователь может сам настраивать и конфигурировать интерфейс под себя, чтобы добиться максимально удобной и эффективной работы с системой;
- *Инновационные возможности видеоаналитики.* Продукт компании Skyros снабжен рядом интеллектуальных детекторов: адаптивный детектор движения, детектор оставленных предметов, саботажа, счетчик объектов, детекторы пересечения и направления движения, которые способны автоматически выявлять опасные и подозрительные ситуации на контролируемом объекте.

1.1.4 Система видеонаблюдения ITV|Axxon Next

Axxon Next – программная платформа видеонаблюдения от компании ITV, объединяющая нейросетевую видеоаналитику и поддерживающая более 10000 моделей IP-устройств.

Система надежна и достаточно проста в использовании за счет удобного пользовательского интерфейса. Продукт подходит для решения задач любой сложности – от мелких компаний до крупных распределенных объектов. Основные преимущества системы:

- поддержка обширного спектра IP-камер (в том числе и аналоговых);
- использование микромодульной архитектуры, которая обеспечивает высокую стабильность и надежность системы;
- удобный пользовательский интерфейс;
- удаленная работа с системой через мобильные устройства и веб – интерфейс;
- прекрасная масштабируемость;
- удобный видеоархив с функцией «умного поиска» (MomentQuest, поиск по лицам и номерам автомобилей);
- аппаратное ускорение: Axxon Next может использовать аппаратные ускорители для выполнения ресурсоемких вычислений нейросетевой аналитики;
- интерактивная 3D-карта, отображающая видео и расположение камер в одном месте при наблюдении в режиме реального времени;
- интеграция с внешними устройствами и системами;
- подключение резервного сервера при потере связи с основным;
- бесплатные обновления и поддержка;
- офлайн-аналитика: система позволяет анализировать импортированные видеоданные и применять к ним функцию «умного поиска».

1.1.5 Аппаратно-программный комплекс GoalCity Instinct 2.0

GoalCity Instinct 2.0 – легко масштабируемый аппаратно-программный комплекс, позволяющий работать с IP и аналоговыми камерами и решать любые задачи по обеспечению безопасности различных объектов. Модульность архитектуры и распределенно-сетевой принцип построения – главные особенности данного решения, благодаря которым оно является универсальной системой безопасности. Основа системы – универсальная сетевая архитектура, которая способна объединить все типы объектов в единую структуру с использованием различных средств коммуникации. С помощью обычного мобильного телефона можно управлять системой в режиме реального времени.

1.1.6 Комплекс видеоаналитики Цефей

Цефей – комплекс видеоаналитики от компании Синезис, созданный для охраны и безопасности крупных и стратегически важных объектов: границ, объекты топливно-энергетических комплексов, исправительных учреждений и других, особо охраняемых объектов.

Продукт автоматически распознает опасные ситуации в видеопотоке, полученном от камер видеонаблюдения. При возникновении проблем система передает сообщение на пульт охраны, производя при этом документальную запись видео и протокола событий. Благодаря этому, комплекс позволяет уменьшить риск возникновения опасных ситуаций. Основные достоинства комплекса:

- высокая точность распознавания цели;
- использование архитектуры на базе ядра Linux и международного интерфейса ONVIF;
- поддержка операционных систем Linux и Windows;
- всесторонний контроль за объектом за счет широкого набора видеоаналитических модулей.

1.1.7 Сравнение систем видеоаналитики

Таблица 1.*1* – Сравнение систем видеоаналитики

Функция (критерий сравнения)	CasRetail	VOCORD FaceControl	VideoNet	ITV| Axxon Next	GoalCity Instinct 2.0	Цефей
Детектор движения	+	+	+	+	+	+
Выделение лиц и идентификация	+	+	+	+	+	+
Детектор активности персонала	+	+	+	+	+	+
Тепловизор	+	-	+	+	+	-
Облачные вычисления	+	+	+	+	-	+
Контроль слепых зон	+	+	+	-	+	-
Направленное движение	+	+	+	+	+	+
Классификатор объекта	+	+	+	+	+	+
Мобильное приложение и оповещения	+	+	+	+	+	+

Рассматривая существующие системы и комплексы видеоаналитики можно сделать вывод, что в современном мире систем видеоаналитики достаточно много, у каждой из них есть свои индивидуальные достоинства. Внедрение таких систем в мелкие и крупные заведения повышает их безопасность и защищенность. Стоит также отметить, что большинство систем видеоаналитики используют облачные вычисления, что зачастую бывает невыгодно из-за того, что у такого подхода существует ряд недостатков.

1.2 **Облачные вычисления и их достоинства и недостатки**

Облачные вычисления (Cloud computing) – концепция, согласно которой все вычисления производятся на удаленном компьютере (в так называемом «облаке»), а результаты работы выводятся в окно веб-браузера на стандартном персональном компьютере. Все приложения и данные так же расположены в «облаке». Основные достоинства:

- снижение требований к вычислительной мощности (программному обеспечению) персональных компьютеров;
- высокая скорость обработки данных;
- экономия ресурсов локальных компьютеров;
- снижение затрат на электроэнергию.

Однако у такого подхода существуют и свои минусы:

- необходимо постоянное соединение с сетью Интернет. Многие «облачные» ресурсы требуют высокоскоростного соединения к сети Интернет с высокой пропускной способностью;
- не все программы могут функционировать удаленно;
- сильная загруженность облачных серверов может привести к увеличению времени на выполнение вычислений;
- возможная угроза безопасности данных клиентов. Здесь стоит отметить того, кто предоставляет облачные услуги. Если данные клиентов шифруются, производятся резервные копии этих данных, то угрозы безопасности клиентских данных может никогда не случиться;
- такие вычисления являются дорогими.

1.3 **Технология граничных вычислений**

1.3.1 Концепция граничной аналитики Edge Analytics

Граничные или периферийные вычисления были созданы для того, чтобы устранить недостатки облачных вычислений. В основе концепции граничной аналитики Edge Analytics находится сбор, обработка и анализ данных на граничных устройствах сети, то есть в непосредственной близости с источником данных [2]. Edge computing – переход от централизованных облачных вычислений к периферийным или граничным вычислениям. Сами же граничные вычисления – совокупность вычислительных ресурсов (например, программное обеспечение, серверы и сетевые подключения), развертывающиеся по периметру предприятий. Таким образом, информация по максимуму обрабатывается на локальных компьютерах, а облачные ресурсы можно использовать, например, в качестве хранилища результатов вычислений (например, для хранения готовых результатов и отчетов). Главная функция Edge computing – плавная интеграция облачных ресурсов и вычислений с граничными устройствами и двусторонний обмен информацией (Рисунок 1.1).

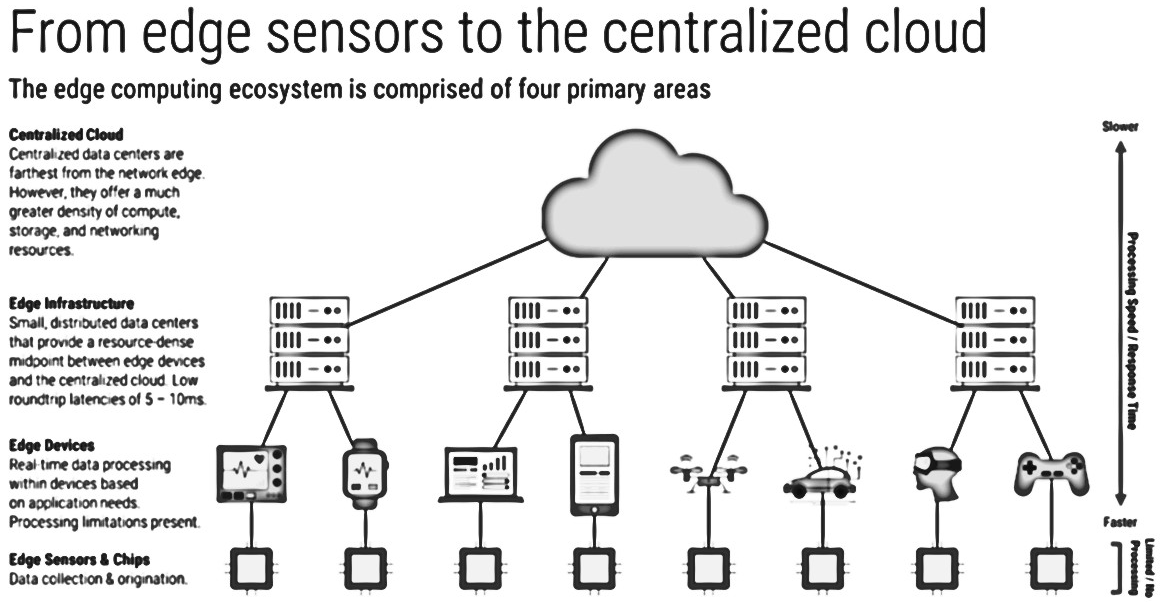

Рисунок 1.1 – Схема реализации граничных вычислений [2]

Решение, которое использует граничные вычисления, предоставляет возможность обрабатывать и анализировать ключевые данные «поблизости» в режиме онлайн. «Периферия сети» содержит в себе модули принятия решений и модули временного хранения данных, которые в будущем использоваться не будут, то есть такие данные являются незначительными.

1.3.2 Основные преимущества граничных вычислений Edge Computing

Среди основных преимуществ периферийных вычислений можно выделить следующие:

- минимизация проблем со связью;
- снижение задержек;
- конфиденциальность важной информации.

В первом случае при использовании концепции Edge computing большинство вычислений производятся на локальных устройствах, что гарантирует то, что работа не будет прервана в случаях неудовлетворительного (например, ограниченного или непостоянного) Интернет-соединения. Это особенно важно в местах, где нет постоянного и стабильного сетевого подключения.

Во-вторых, на сегодняшний день достаточно много пользователей используют облачные вычисления, передавая «облаку» большое количество информации для последующей обработки, из-за чего могут возникать значительные задержки при ожидании результатов от облачного сервиса, что может негативно сказываться на работе предприятий.

В-третьих, все конфиденциальные данные лучше непосредственно обрабатывать «поблизости», а в облачный сервис передавать лишь те данные, которые соответствуют политике конфиденциальности, чтобы избежать утечки важной информации.

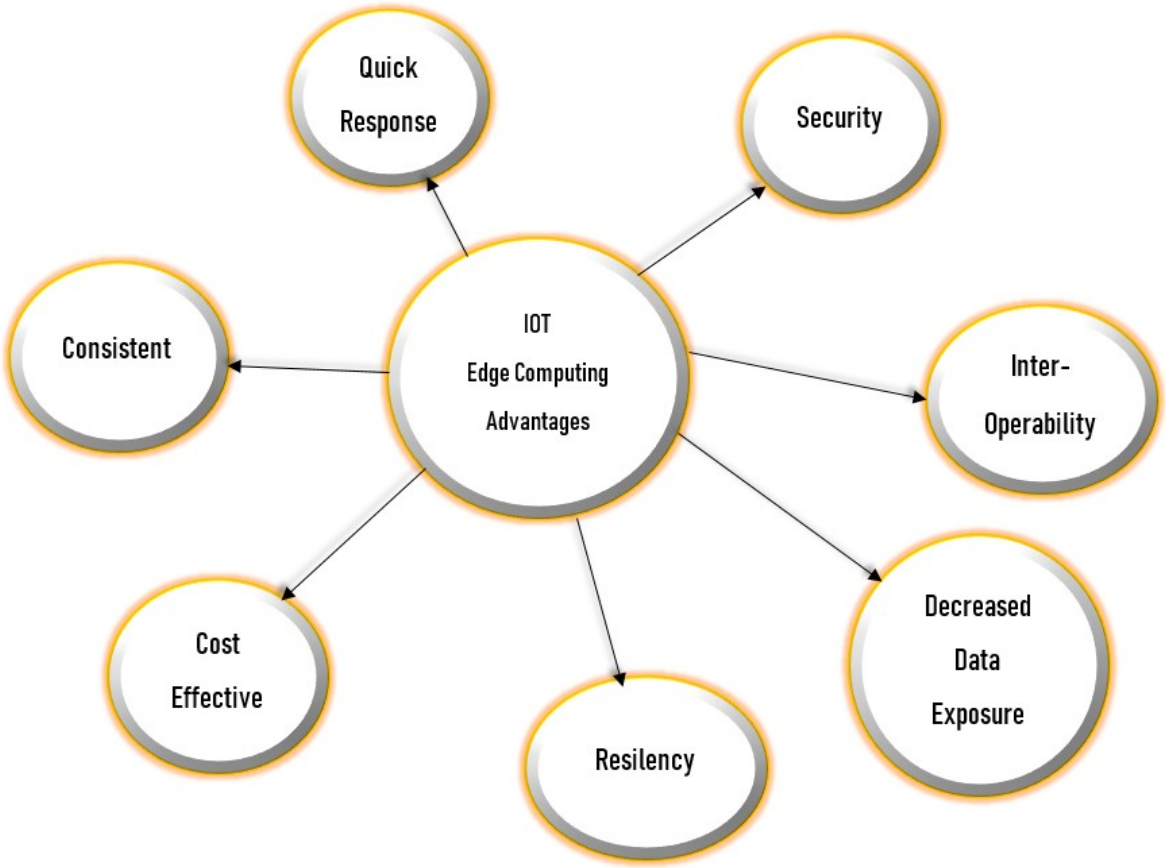

Рисунок 1.2 – Преимущества Edge Computing

ВЫВОДЫ:

1. Рассмотрены современные системы и комплексы видеоаналитики и произведен их сравнительный анализ.
2. Установлено, что большинство систем видеоаналитики используют технологию облачных вычислений, имеющую ряд недостатков.
3. Рассмотрена технология граничных вычислений, помогающая устранить недостатки облачных вычислений, и ее основные преимущества.

ГЛАВА 2. АЛГОРИТМЫ И СРЕДСТВА ДЛЯ АНАЛИЗА ВИДЕОДАННЫХ

2.1 Средства для анализа видеопотока

2.1.1 Библиотека OpenCV

OpenCV (Open source computer vision library) – библиотека алгоритмов компьютерного зрения и машинного обучения, написанная на C/C++. Библиотека позволяет производить обработку, классификацию и анализ изображений с использованием различных численных алгоритмов с открытым исходным кодом. Популярна в таких языках программирования, как Python, C, C++, Java.

Библиотека включает в себя большое количество алгоритмов, предназначенных для решения различных задач:

- распознание рукописного и печатного текста;
- удаление искажений изображения;
- наблюдение за перемещением объекта;
- распознавание различных движений, жестов;
- распознавание объектов в видеоданных;
- установление сходства и формы объектов.

2.1.2 Метод Виолы-Джонса

Для поиска объектов, используя их ключевые признаки, используется метод П. Виолы и М. Джонса. В основе подхода лежит 4 концепции:

- функции Хаара (простые прямоугольные функции);
- интегральное представление изображения;
- AdaBoost;
- каскадный классификатор.

Метод Виолы-Джонса использует технику «скользящего окна». «Скользящее окно» – рамка, размер которой меньше, чем размер исходного изображения, которая передвигается по изображению с некоторым шагом и определяет наличие лиц на рассматриваемом изображении, используя каскад «слабых» классификаторов. Такая технология эффективно используется в области компьютерного зрения для выявления различных объектов на изображениях [4].

Особенности, которые использованы в данным методе, опираются на вейвлеты Хаара. Вейвлеты Хаара – прямоугольные волны одинаковой длины [1]. В двумерном пространстве такие прямоугольные волны – пара соседних

светлого и темного прямоугольников, которые называются примитивами Хаара или маской (Рисунок 2.1).

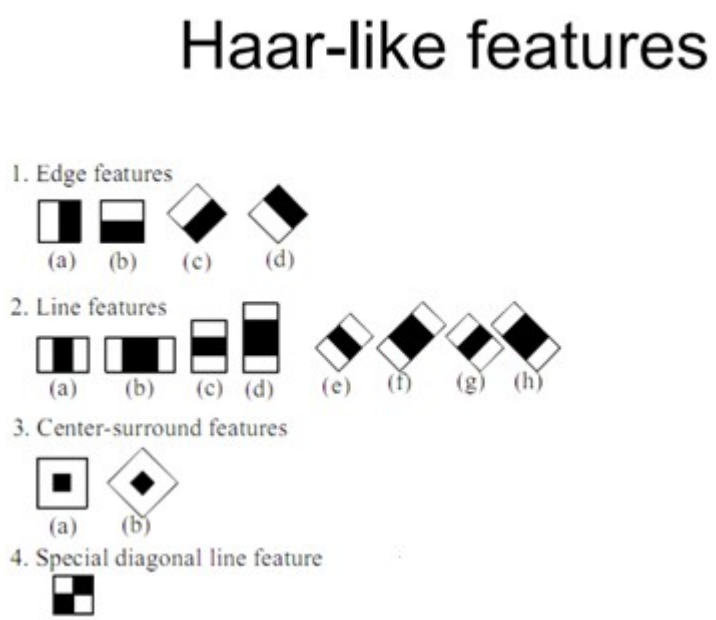

Рисунок 2.3 – Примеры примитивов Хаара [4]

Наличие функции можно установить с помощью примитивов Хаара. При наложении примитивов на исходное изображение можно получить числовое значение – разность суммы яркости пикселей, попавших в белую часть примитива, и суммы пикселей, попавших в черную часть примитива (1):

$$FHaar = Bwhite - Bblack. \quad (1)$$

Здесь Bwhite – сумма яркости пикселей, попавших в белую область примитива, Bblack – в черную область примитива.

Далее это число сравнивают с пороговым значением, которое предварительно определяется на этапе обучения: если число превышает пороговое значение, то функция Хаара существует.

Технология интегрального представления изображения используется для эффективного определения большого числа функций Хаара. Вес каждого пикселя – это его яркость. Интегральное значение каждого пикселя – сумма значений яркости пикселей, находящихся выше и левее текущего пикселя, и вес рассматриваемого пикселя [6]. В программах для вычисления интегрального значения пикселя можно воспользоваться формулой (2):

$$L(x,y) = \sum_{i=0,j=0}^{i \le x, j \le y} I(i,j). \quad (2)$$

Здесь $I(i,j)$ – яркость пикселя с координатами (*i,j*) исходного изображения.

Интегральное представление изображения – матрица, размеры которой совпадают с размерами рассматриваемого изображения, каждый элемент которой – интегральное значение соответствующего пикселя изображения. Если начинать обход изображения с левого верхнего угла и идти вправо и вниз, то изображение можно быстро «интегрировать».

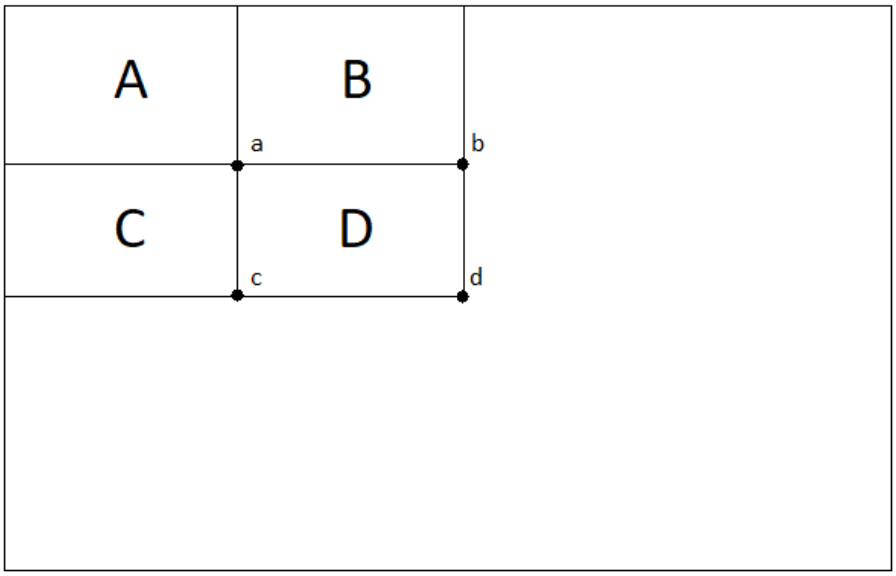

Рисунок 2.4 – Рассматриваемая область D

Например, $(x_a, y_a),(x_d, y_d),(x_c, y_c),(x_b, y_b)$ – интегральные значения для точек A, D, C, B соответственно, тогда для представленной на рисунке 2.2 области D интегральное значение может быть посчитано по следующей формуле (3):

$$I(D)=(x_a, y_a)+(x_d, y_d)-(x_c, y_c)-(x_b, y_b). \quad (3)$$

AdaBoost в методе Виолы-Джонса предназначен для установления пороговых значений и выбора конкретных используемых функций Хаара. Основная цель AdaBoost – создание «сильного» классификатора путем комбинации «слабых» классификаторов. Под «слабым» классификатором понимается такой классификатор, который совершает в работе достаточно много ошибок и дает правильный результат не чаще случайного угадывания, то есть не способен гарантированно дать точный ответ. Основная идея такого подхода заключается в том, что если есть совокупность «слабых» классификаторов, причем каждый отдельный классификатор ошибается независимо от других и получает ответ с некоторой ошибкой, то комбинация всех таких классификаторов должна дать более низкую вероятность ошибки по сравнению с результатом некоторого отдельного классификатора из этого

множества. Метод машинного обучения AdaBoost осуществляет выбор классификаторов и каждому классификатору ставит в соответствие некоторое число, называемое весом. Линейная комбинация классификаторов, предварительно умноженных на свой вес, – это «сильный» классификатор, который можно представить формулой (4):

$$f(x)=\sum_{k=1}^{m} \alpha_k * r_k(x). \quad (4)$$

Здесь $f(x)$ – «сильный» классификатор, $r_k(x)$ – k-ый «слабый» классификатор, α_k – вес k-того «слабого» классификатора, x – некоторый шаблон для классификации.

Каскадный классификатор – модель, состоящая из последовательных уровней с фильтрами. Фильтр – классификатор AdaBoost с малым числом «слабых» классификаторов. Фильтры располагаются по уровням, на начальных уровнях располагаются самые тяжелые фильтры для отсеивания областей, которые не содержат рассматриваемый объект. Изображение последовательно проходит все фильтры. На каждом уровне фильтр может или пропустить, или не пропустить часть изображения. Если фильтр не пропускает часть изображения, то она рассматривается как «не лицо» и дальше не обрабатывается. Таким образом, пропуская всё изображение через последовательность фильтров можно или распознать все лица на фотографиях, или сделать вывод о том, что на фотографии нет людей. Пример фильтрации области изображения представлен на рисунке 2.3.

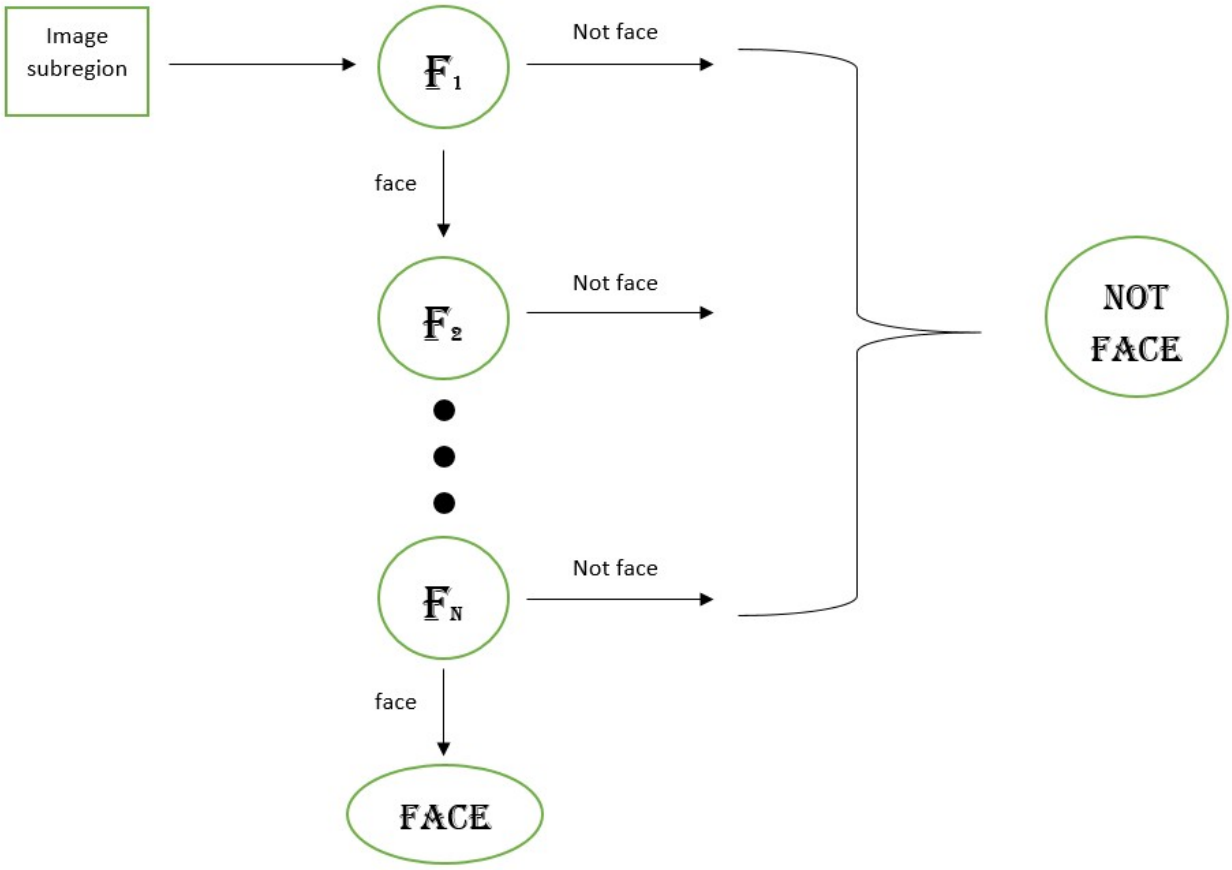

Рисунок 2.5 – Пример фильтрации
области изображения

Метод Виолы-Джонса – эффективный метод распознавания лиц на изображениях, но у данного метода есть свои недостатки:

- переобучение из-за высокого уровня шума в данных;
- большая обучающая выборка и длительное обучение.

2.2 Алгоритмы и средства идентификации лиц на видеоданных

2.2.1 Azure Cognitive Services

Azure Cognitive Services от компании Microsoft – набор облачных сервисов, позволяющих быстро и удобно создавать приложения с использованием машинного обучения и когнитивных средств искусственного интеллекта (интеллектуальных алгоритмов).

Используя Azure Cognitive Services, получаем доступ к средствам искусственного интеллекта, при этом для успешной работы с данными средствами не нужно быть профессионалом в области машинного обучения. За счет удобного API, который предоставляет Cognitive Services, можно интегрировать в приложения различные возможности: «видеть», «слышать»,

«искать», «говорить», «понимать информацию» и «быстро принимать решения» (Рисунок 2.4).

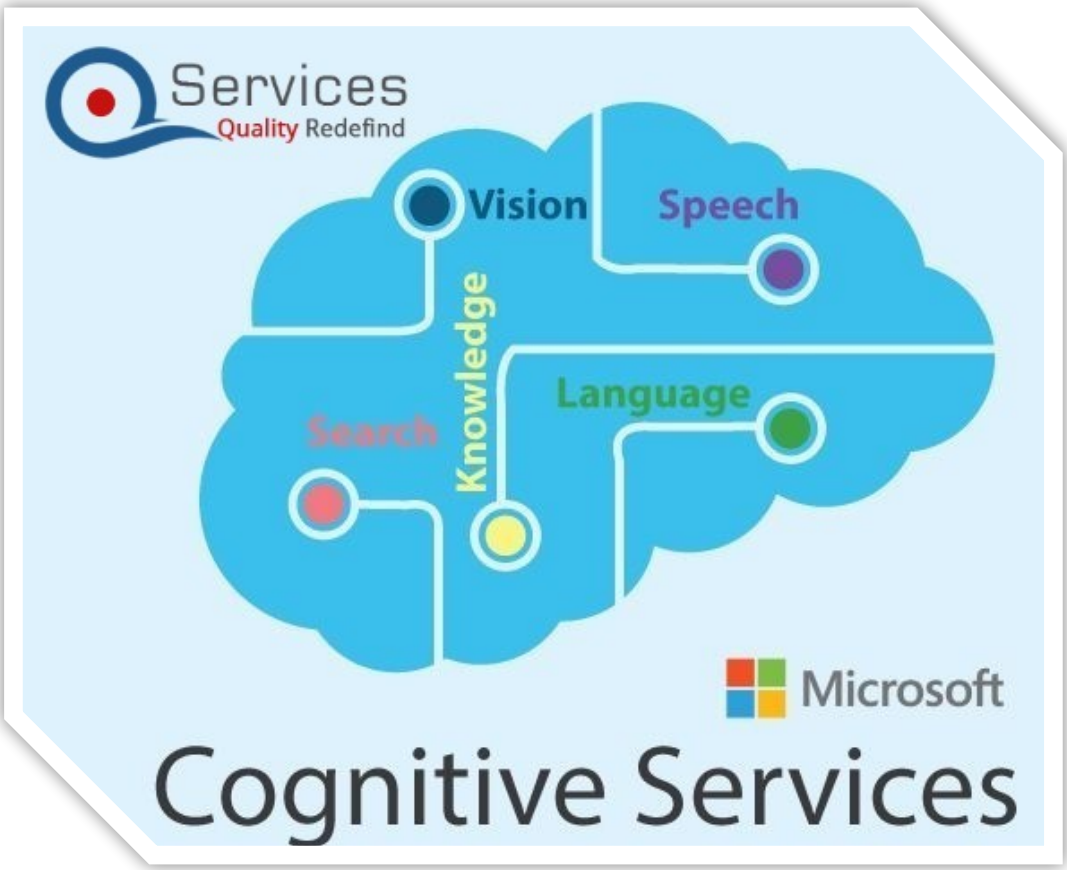

Рисунок 2.6 – Azure Cognitive Services [3]

Преимущества службы Azure Cognitive Services:

- применение средств искусственного интеллекта при создании различных приложений;
- быстрое развертывание службы с использованием контейнеров в облачной среде и локально;
- облегченная работа со средствами искусственного интеллекта, для которых не нужно специальных знаний.

2.2.1.1 API Идентификации лиц.

Azure Face – облачная служба Azure, позволяющая на изображениях обнаруживать и анализировать лица. В ее состав входит API идентификации лиц, позволяющий по существующей базе данных знакомых лиц идентифицировать лицо на фотографии [15].

Создав заранее базу данных со знакомыми лицами и обучив на ней сервис, в дальнейшем можно на фотографиях проводить идентификацию

личности путем сравнения объекта – лица с лицами из базы данных. На рисунке 2.5 приведен пример базы данных с лицами «known people».

Рисунок 2.7 – База данных «known people»

Стоит отметить, что каждая группа может содержать до 1 миллиона объектов, соответствующих разным людям. В свою очередь, для каждого объекта, соответствующего определенному человеку, можно зарегистрировать до 248 лиц. Если при сравнении с базой данных лицо было опознано, то возвращается объект, соответствующий этому человеку.

2.2.2 Метод направленных градиентов HOG

Метод направленных градиентов (HOG) – один из самых популярных методов распознавания объектов, базирующийся на дескрипторах особых точек и работающий достаточно быстро и надежно. HOG-дескрипторы используются для решения различных проблем и задач: распознавание пешеходов на статических картинках, распознавание лиц и различных частей тела, распознавание жестов рук для перевода на язык жестов.

Теория, лежащая в основе алгоритма, заключается в том, что с помощью распределения локальных градиентов интенсивности можно достаточно хорошо охарактеризовать внешний вид и форму объектов на изображениях. Дескрипторы гистограмм направленных градиентов помогают упрощать изображение, извлекая из него только самую ценную информацию [11].

Исходное изображение обычно делают черно-белым потому, что информация о цвете никак не участвует в детектировании лиц. Далее рассматривается окружение каждого пикселя. Это необходимо для того, чтобы выяснить, насколько темным является пиксель по отношению к его соседям [8]. Выполнив такую операцию для каждого пикселя изображения, пиксели затем могут быть заменены стрелкой, что показано на рисунке 2.6.

Рисунок 2.8 – Исходное изображение и его HOG – представление

Стрелка – градиент, идущий от светлых пикселей к темным, то есть направленный в сторону затемнения изображения. Таким образом получают шаблоны.

Стоит отметить, что если хранить в памяти компьютера градиент для каждого пикселя изображения, то это займет слишком много памяти. Поэтому для оптимизации памяти исходное изображение можно разбить на ячейки размером 16 на 16 пикселей каждая, каждая из которых может быть разбита на более мелкие ячейки размером 8 на 8 пикселей. В каждой маленькой ячейке каждый пиксель заменяется стрелкой и производится подсчет стрелок, имеющих одинаковое направление. Далее ячейка размером 8 на 8 пикселей заменяется стрелкой, направленной туда же, куда и большинство маленьких градиент. Аналогичная процедура проводится для блоков размером 16 на 16 пикселей.

Классификацию дескрипторов гистограммы направленных градиентов можно проводить с помощью метода опорных векторов SVM (Support vector machine). Идея SVM: с помощью разделяющей гиперплоскости отделить векторы признаков объекта и векторы признаков фона [7]. Пример разделяющий гиперплоскости приведен на рисунке 2.7.

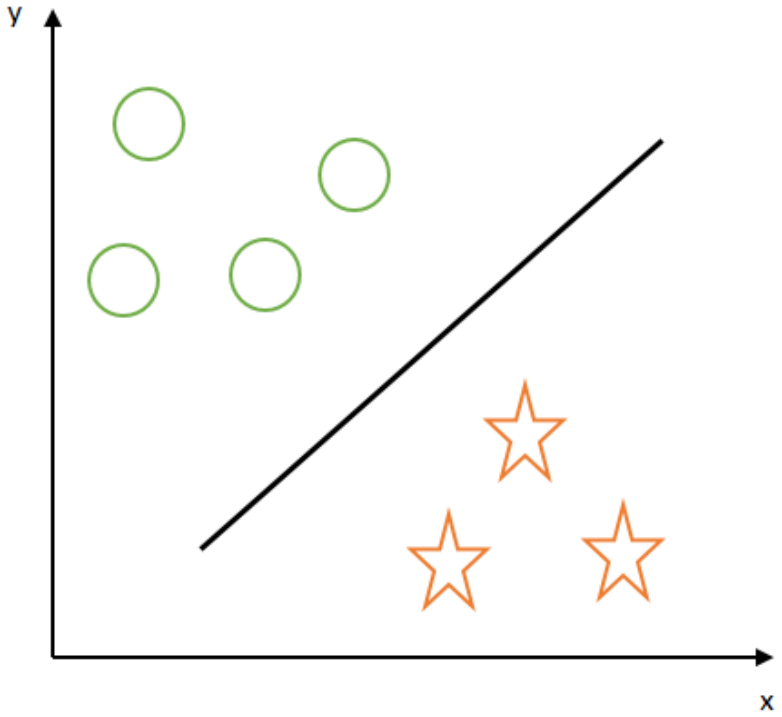

Рисунок 2.9 – Пример гиперплоскости,
разделяющей круги от звезд

Результат работы метода опорных векторов – два образа объекта с отрицательными и положительными весами опорных векторов. Положительный вес означает принадлежность признака объекту (в данной работе объект – лицо), а отрицательный вес показывает, что признак принадлежит фону.

На выходном шаблоне достаточно хорошо проглядывается базовая структура лица, что позволяет эффективно обнаруживать лица на новых изображениях, используя уже знакомые шаблоны.

2.2.3 Сверточные нейронные сети CNN

2.2.3.1 Общие понятия о сверточных нейронных сетях.

CNN или сверточная нейронная сеть – нейросетевая архитектура, которая используется для классификации изображений и графических образов (Рисунок 2.8).

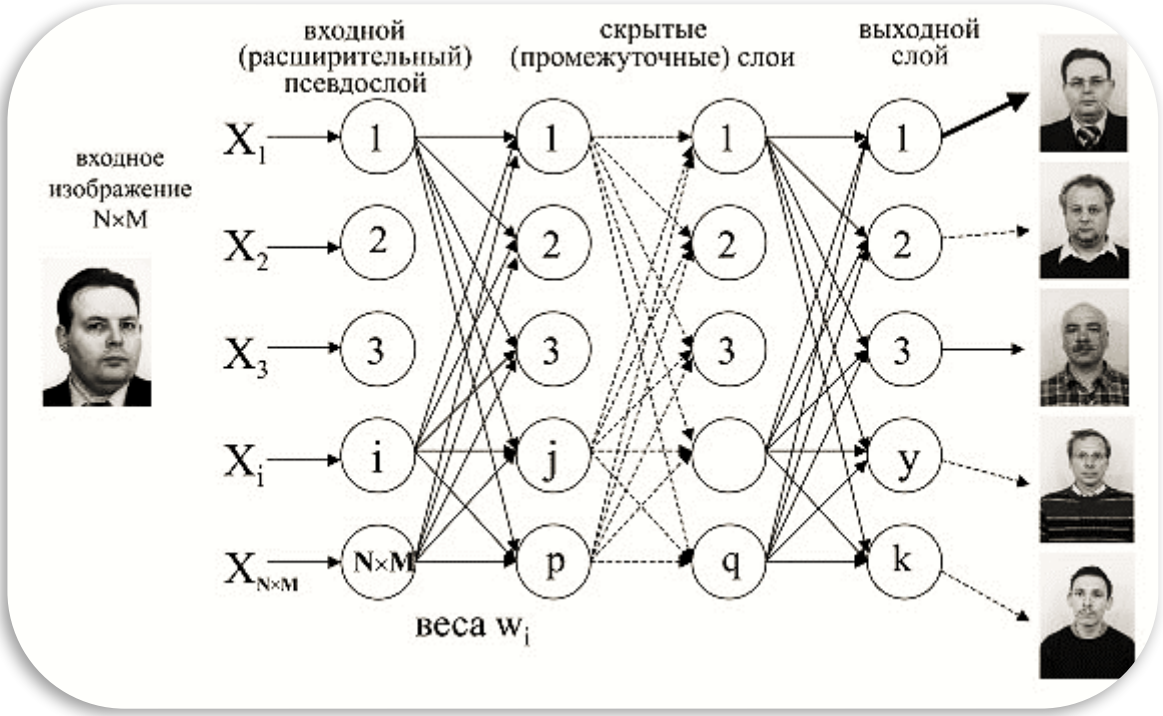

Рисунок 2.10 – Нейронная сеть для классификации изображений [13]

Сверточная нейронная сеть популярна в области машинного обучения и компьютерного зрения для решения различных проблем и задач.

Учатся CNN на изображениях, используя шаблоны (паттерны) для классификации, поэтому не нужно извлекать признаки вручную. Один из самых распространенных методов обучения – метод обратного распространения ошибки. Свое название CNN приобрела за счет использования такой операции, как свертка. Операция свертки – операция над матрицами A (размера n_x на n_y) и B (размера m_x на m_y), результатом которой является матрица C (размера $m_x - n_x + 1$ на $m_y - n_y + 1$), элементы которой вычисляется по следующей формуле (5):

$$C_{k,m} = \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} x_{i+k,j+m} * \omega_{ij} + \omega_0, k = \overline{1, m_x - n_x + 1}, m = \overline{1, m_y - n_y + 1}. \quad (5)$$

.

Здесь $x_{i,j} (i = \overline{1, n_x}, j = \overline{1, n_y})$ – элементы матрицы B, ω_{ij} $(i = \overline{1, m_x}, j = \overline{1, m_y})$ – элементы матрицы A, ω_0 – некоторое число (bias или нейрон смещения).

Пример операции свертки приведен на рисунке 2.9.

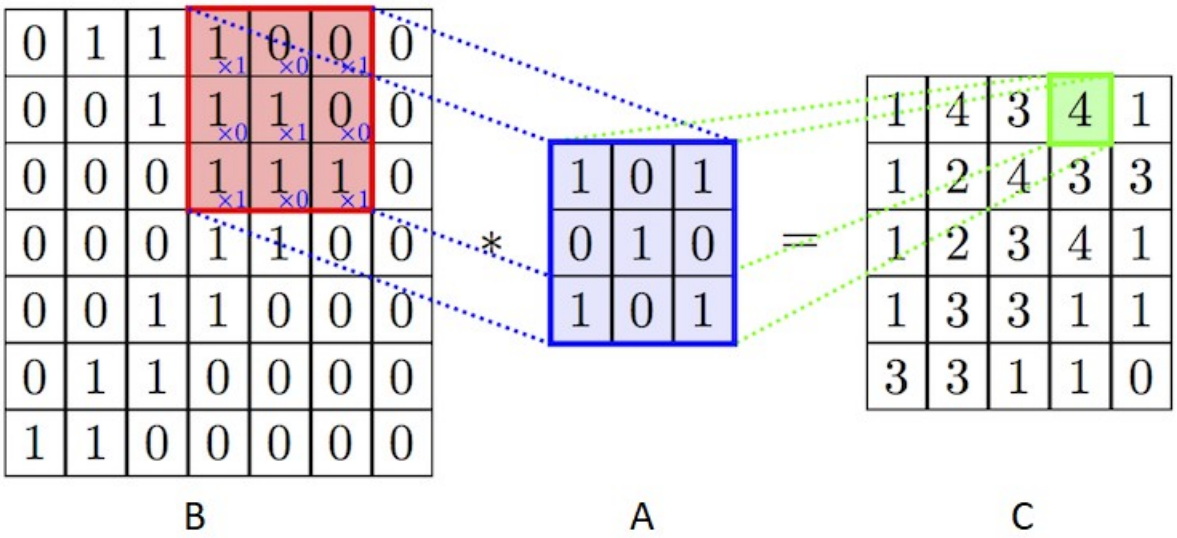

Рисунок 2.11 – Пример операции свертки

Матрица B называется изображением, а матрица A – фильтром. При операции свертки ядро «передвигается» по изображению.

Свою популярность сверточные нейронные сети получили за счет трех важных факторов:

- устранение ручного извлечения признаков;
- высокая точность результатов;
- переобучение CNN, за счет которого можно использовать нейронные сети для распознавания разных графических объектов.

В зависимости от структуры приложения можем построить свою нейронную сеть или использовать готовую, заранее обучив ее на некоторой выборке. В современном мире нейронные сети можно использовать в медицине для визуального обнаружения раковых клеток, в беспилотных автомобилях для обнаружения объектов и распознавания дорожных знаков, для идентификации и классификации звука, текста, видео и изображений [12].

2.2.3.2 Классификация слоев CNN.

CNN включает в себя входной и выходной слои, а также множество скрытых между ними слоев. Скрытые слои используются для изучения характеристик исходного изображения, среди которых можно выделить три самых распространенных слоя: сверточный или свертка, слой активации или ReLu и pooling (субдискретизирующий слой).

В сверточных слоях входное изображение проходит через набор фильтров, каждый из которых активирует или выделяет определенные характеристики изображений (например, горизонтальные или вертикальные линии). В случаях с изображениями в градациях серого фильтр будет один, но при многоканальном изображении фильтр будет состоять из ядер, число

которых совпадает с числом каналов изображения. Скалярный результат свертки попадает на слой активации, то есть на некоторую нелинейную функцию [13]. Функция активации отсекает отрицательные значения и сохраняет положительные. Субдискретизирующий слой или pooling «уплотняет» или уменьшает изображения, производя некоторое нелинейное преобразование и сохраняя только самую ценную информацию. В качестве нелинейного преобразования изображения можно использовать метод maxpooling, minpooling или averagepooling. Продемонстрировать операцию maxpooling можно на примере изображения размером 128 на 128 пикселей. Выбираются непересекающиеся окна некоторого размера, например, 2 на 2 пикселя, которые будут «скользить» по карте признаков, которая была получена применением операции свертки к изображению, и в каждом таком окне выбирается максимальное значение. Результат применения maxpooling приведен на рисунке 2.10.

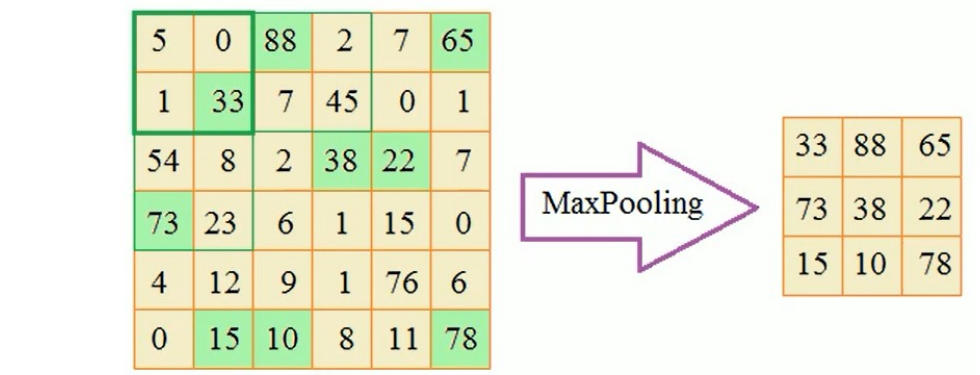

Рисунок 2.12 – Результат применения операции maxpooling

Данные операции повторяются на многих слоях с целью определить различные характеристики изображения. Входное изображение для последующего слоя – результат применения фильтров к входному изображению на текущем слое. Ниже на рисунке 2.11 приведен пример нейронной сети с достаточно большим количеством сверточных слоев.

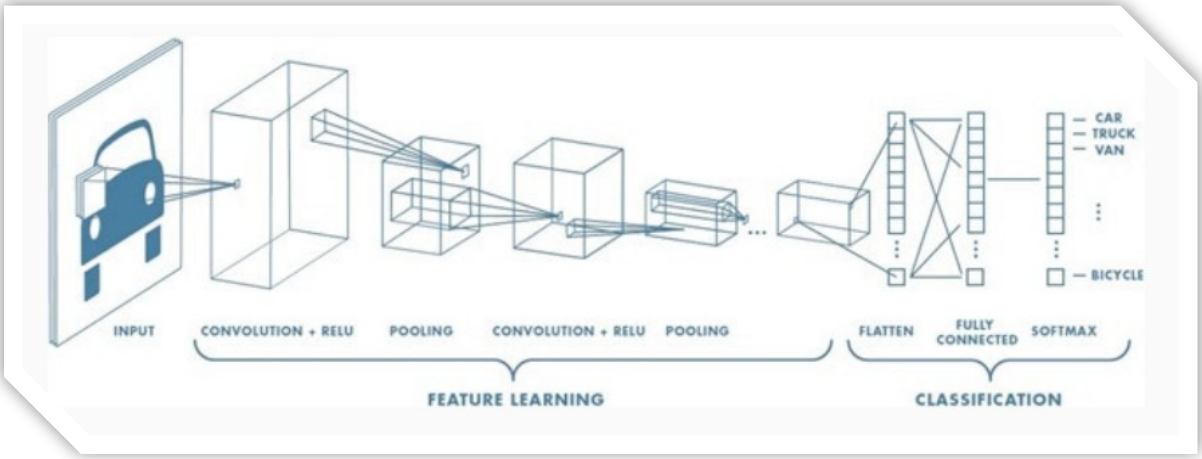

Рисунок 2.13 – Пример нейронной сети с большим числом сверточных слоев [12]

Последние слои сверточных нейронных сетей предназначены для классификации. Результат нейронной сети – вектор, элементами которого есть вероятности принадлежности изображения тому или иному классу. Размер такого вектора равен количеству классов, к которым может относиться входное изображение.

2.2.3.3 Аппаратное ускорение CNN с помощью графических процессоров.

Сверточные нейронные сети обучаются на большой обучающей выборке. При работе с громоздкими объемами данных можно все вычисления проводить на графических процессорах, чтобы ускорить процесс как обучения нейронной сети, так и обработки входных изображений.

ВЫВОДЫ:

1. Рассмотрены современные средства и алгоритмы для анализа видеоданных.
2. Установлено, что все вычисления лучше всего производить с помощью графических процессоров при использовании сверточных нейронных сетей, в то же время Microsoft Cognitive Services является платной облачной службой, поэтому в качестве алгоритма для идентификации лиц на изображениях в разрабатываемой системе будет использоваться HOG.

ГЛАВА 3. РАЗРАБОТКА СИСТЕМЫ ХРАНЕНИЯ И ОБРАБОТКИ ВИДЕОДАННЫХ

3.1 Выбор инструментов реализации

3.1.1 Языки программирования

3.1.1.1 Java.

Java – язык программирования от компании Sun Microsystems, являющийся интерпретируемым и строго типизированным. Среди основных возможностей Java можно выделить следующие:

- сборщик мусора (garbage collector), позволяющий освобождать память, удаляя при этом объекты, которые будут не востребованы приложениями;
- удобная обработка исключительный ситуаций;
- java collections;
- средства фильтрации и форматирования ввода/вывода;
- lambda – expressions (начиная с версии 1.8), позволяющие уменьшить количество кода и улучшить его читаемость;
- поддержка обобщений (java generics, начиная с версии 1.5);
- поддержка многопоточности;
- наличие JDBC (Java Database Connectivity), который позволяет приложениям, написанным на Java, взаимодействовать с различными системами управления базами данных;
- наличие средств (классы), с помощью которых можно осуществлять HTTP-запросы и обрабатывать соответствующие ответы;
- наличие инструментов, позволяющих легко создавать сетевые приложения.

3.1.1.2 Python.

Python – активно развивающийся интерпретируемый высокоуровневый язык программирования, направленный на решение разноплановых задач и проблем. Синтаксис языка довольно прост, его используют многие компании – гиганты в IT – сфере и он включает в себя множество библиотек с готовыми решениями.

Python поддерживает объектно – ориентированное, функциональное и обобщенное программирование. Достоинства Python – наличие механизма обработки исключительных ситуаций, генераторы и итераторы, декораторы и

поддержка лямбда – выражений, гибкость и масштабируемость. Язык поддерживает разбиение программы на модули, которые затем можно объединить в пакеты. Данный язык пригоден для написания различных мобильных приложений, web – сайтов, а также им пользуются в машинном обучении.

3.1.2 Среда разработки и редактор кода

3.1.2.1 Intellij IDEA.

Intellij IDEA – интегрированная среда разработки Java (IDE) от компании JetBrains, предназначенная для разработки различного программного обеспечения. Среда разработки поддерживает системы контроля версий (например, SVN и Git), доступ к различным базам данных (в версии Ultimate), инструменты сборки (например, Maven), большое количество различных фреймворков, плагины, с помощью которых можно добавлять в IDE новую функциональность. Плагины можно либо загружать через веб – сайт, либо через встроенную в IDE функцию поиска и установки плагинов.

На основе анализа кода среда предоставляет пользователю большие возможности для проведения рефакторинга кода:

- умное автодополнение;
- автодополнение цепочек вызовов и статических членов;
- поиск повторов;
- быстрые исправления и языковые вставки;
- рефакторинги для различных языков.

Стоит также отметить, что Intellij IDEA позволяет программировать на различных языках программирования.

3.1.2.2 Visual Studio Code.

Visual Studio Code – бесплатный редактор исходного кода от компании Microsoft для операционных систем Windows, macOS и Linux. Данный продукт служит для разработки различных веб – и «облачных» приложений. Редактор поддерживает подсветку синтаксиса, функцию автозавершения кода (IntelliSense), отладку и навигацию по коду, большинство языков программирования, систему контроля версий (Git) и многое другое. В редакторе присутствует встроенная консоль, в которой пользователь может увидеть результат работы программы или сообщение об ошибке.

3.1.3 Дополнительные инструменты для разработки

3.1.3.1 Azure Functions Core Tools.

Azure Functions Core Tools – программы командной строки, служащие для локальной разработки и выполнения функций Azure (Azure Functions), а также при необходимости публиковать их в Azure.

Основное назначение данного инструмента заключается в следующем:

- создание всех необходимых файлов и папок для локальной разработки функций Azure;
- локальное тестирование и отладка функций;
- публикация функций в облачное хранилище (Microsoft Azure).

3.2 Компоненты Microsoft Azure

3.2.1 Облачное хранилище данных

3.2.1.1 Общие сведения о Microsoft Azure.

Azure – набор служб облачных вычислений от компании Microsoft, который помогает организациям решать бизнес-задачи [3]. Облачная платформа помогает небольшим и молодым компаниям, начиная с небольших затрат, быстро и удобно масштабировать инфраструктуру по мере возникновения новых клиентов и заказов.

Основные достоинства Microsoft Azure:

- простая и удобная работа с гибридным «облаком»;
- создание разнообразных решений;
- безопасность;
- тестирование новых версий программного обеспечения.

Облачная платформа предоставляет свободу создания и развертывания разнообразных приложений, управления ими в обширной глобальной сети с использованием любимых инструментов. Azure позволяет использовать все возможности в локальной среде, «облаке» или на граничных устройствах.

За счет поддержки всех приложений и платформ и возможности использовать открытый код в Azure можем создавать решения удобным для нас способом и развертывать их в любых средах.

Облачная платформа оснащена комплексной защитой, предоставляемой командой экспертов, что гарантирует безопасность и сохранность данных клиентов.

С помощью Microsoft Azure можем тестировать новые версии программного обеспечения, при этом нам не нужно проводить замену

локального оборудования. Например, для тестирования приложения с Microsoft SQL Server 2014 нам необходимо создать экземпляр SQL Server 2014, запустить копию наших служб, предварительно подключенную к новой базе данных, и провести анализы результатов. В данном случае нам не требуется дополнять свое локальное оборудование.

Преимущества Azure над AWS:

- конкурентные цены;
- использование в полной мере потенциала исходного кода;
- использование любых операционных систем с открытым кодом, языков и средств.

С помощью Azure был внесен наибольший вклад в GitHub в 2017 году, а также это единственная облачная платформа с интегрированной поддержкой Red Hat.

3.2.1.2 Классификация данных.

Для каждого набора данных существуют свои требования, и перед нами стоит задача в поиске оптимального решения для хранения данных. Основные факторы, которые должны всегда учитываться при поиске оптимального решения: как правильно классифицировать данные, как данные будут использоваться в дальнейшем и как можно обеспечить максимальную производительность приложения.

Данные разделяют на три типа: структурированные, частично структурированные и неструктурированные. Понимание различий между ними и правильная классификация – ключевые факторы при нахождении оптимального решения для хранения данных.

Структурированные данные четко распределяются по строкам и столбцам в таблицах, хорошо организованы.

Частично структурированные данные – данные, которые хорошо упорядочены, имеют точно определенные свойства и значения, но допускают некоторую изменчивость (например, Yaml, JSON, XML).

Для неструктурированных данных не существует определенной модели данных, их невозможно распределить по таблицам (например, BLOB-объекты, текстовые файлы).

В рамках данной работы основными данными являются изображения. Изображения относятся к неструктурированным данным. Основные операции над данными:

- требуется только получение по идентификатору;
- для клиентов необходимо предоставить большое количество операций чтения с минимальной задержкой;

- операция обновления будет выполняться крайне редко, и для нее допустима более высокая задержка по сравнению с операцией чтения.

Хранилище BLOB-объектов Azure – отличное решение для данной задачи, оно поддерживает хранение любых файлов, в том числе видео и фотографий. Azure хранилище поддерживает кэширование часто используемого содержимого, сохраняя его на граничных серверах, а с помощью Azure CDN можно сократить задержку при предоставлении данных клиентам. Используя данный подход, изображения можно перемещать между архивным, «горячим» и «холодным» уровнями для увеличение пропускной способности и снижения затрат для самых широко используемых видеоданных.

Существует еще два решения – использование Службы приложений Azure или базы данных. Службу приложений Azure можно использовать в том случае, если изображений будет немного. Но при большом количестве данных лучше использовать хранилище BLOB-объектов Azure вместе с Azure CDN. Хранение всей информации в базе данных – неоптимальное решение по причине размера данных и производительности.

3.2.1.3 Azure Storage.

В Microsoft Azure хранить данные можно различными способами. «Облако» предоставляет различные базы данных для хранения информации: Azure SQL Server, Azure Cosmos DB и хранилище таблиц Azure. Есть несколько вариантов хранения и отправки сообщений, например, очереди Azure и центры событий. С помощью решений таких, как файлы Azure (служба файлов Azure) и BLOB-объекты Azure, можно хранить несвязанные файлы. В Azure Storage (служба хранилища Azure) входят следующие решения: BLOB-объекты Azure, служба файлов Azure, очереди Azure и таблицы Azure (Рисунок 3.1).

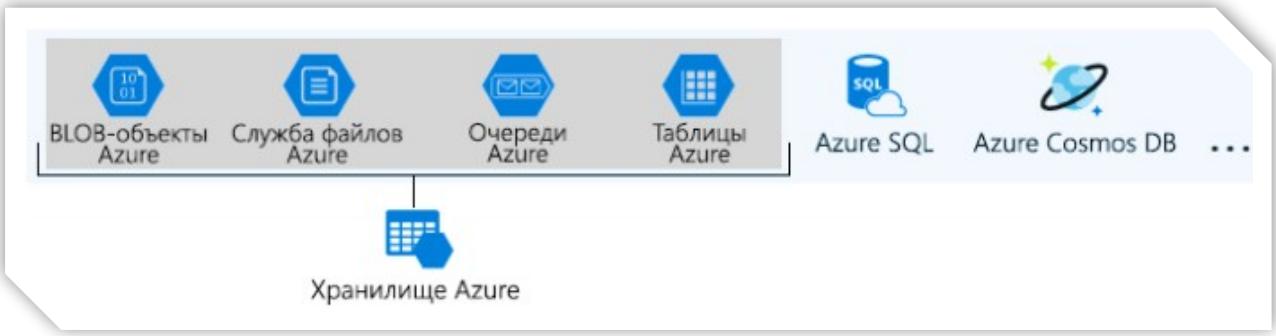

Рисунок 3.14 – Хранилище Azure [3]

Учетная запись хранения (Storage account) – контейнер, группирующий набор служб для службы Azure Storage (Рисунок 3.2). В этот контейнер могут входить только службы данных из хранилища Azure (BLOB-объекты Azure,

файлы Azure, очереди Azure и таблицы Azure). Такое объединение служб данных в одну учетную запись хранения позволяет ими управлять как единым целым (группой). Специальные параметры, которые указываются при создании или редактировании учетной записи хранения, будут применяться ко всему содержимому учетной записи. Все службы, входящие в состав учетной записи, будут удалены при удалении соответствующей учетной записи хранения.

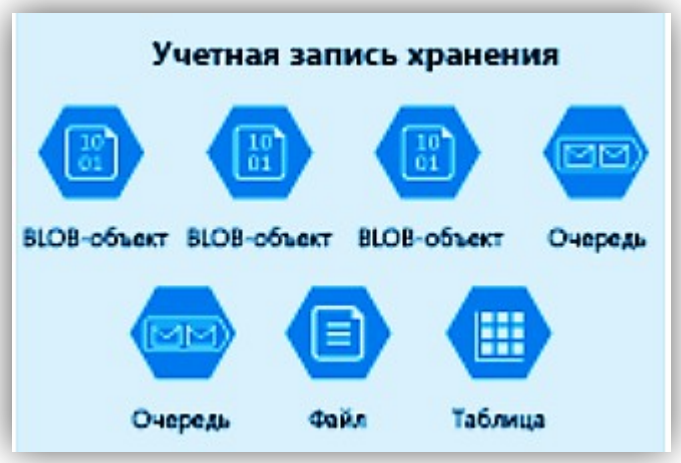

Рисунок 3.15 – Учетная запись хранения

Учетная запись хранения – ресурс Azure, входящий в группу ресурсов Azure (Azure Resource group) (Рисунок 3.3). Службами данных Azure такими, как Azure Cosmos DB и Azure SQL управляют как независимыми ресурсами, такие службы нельзя включить в учетную запись хранения.

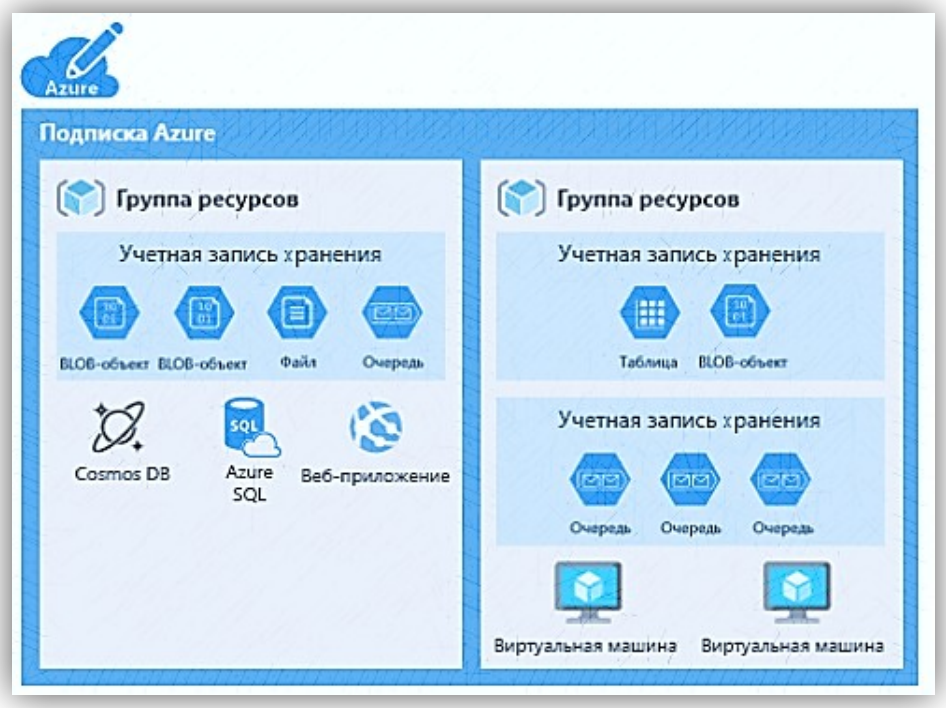

Рисунок 3.16 – Примеры групп ресурсов

3.2.1.4 Azure Storage – хранилище BLOB-объектов.

Хранилище BLOB-объектов Azure – лучшее оптимизированное решение для хранения неструктурированных данных больших объемов, например, текстовых или двоичных данных. Хранилище BLOB-объектов идеально подходит в следующих случаях:

- потоковая передача видеоданных и звука;
- хранение информации для обработки локальной или облачной службой;
- аварийное восстановление и архивация, хранение резервных копий;
- хранение данных для распределенного доступа;
- непосредственное обслуживание фотографий или документов в браузере, включая полные статические веб-сайты.

3.2.2 Бессерверные вычисления и функции Azure

3.2.2.1 Технология бессерверных вычислений.

Бессерверные вычисления – функция как услуга (Faas) или облачная микрослужба, которая доступна по запросу. За счет того, что вся бизнес-логика выполняется в формате функций, не нужно подготавливать или масштабировать инфраструктуру, при этом управление инфраструктурой берет на себя поставщик облачных служб.

В зависимости от нагрузок приложения масштабируются вверх или вниз. Существует несколько вариантов создания такой архитектуры с использованием компонентов Microsoft Azure. Два самых популярных подхода – использование Azure Logic Apps и Azure Functions (функций Azure) (Рисунок 3.4).

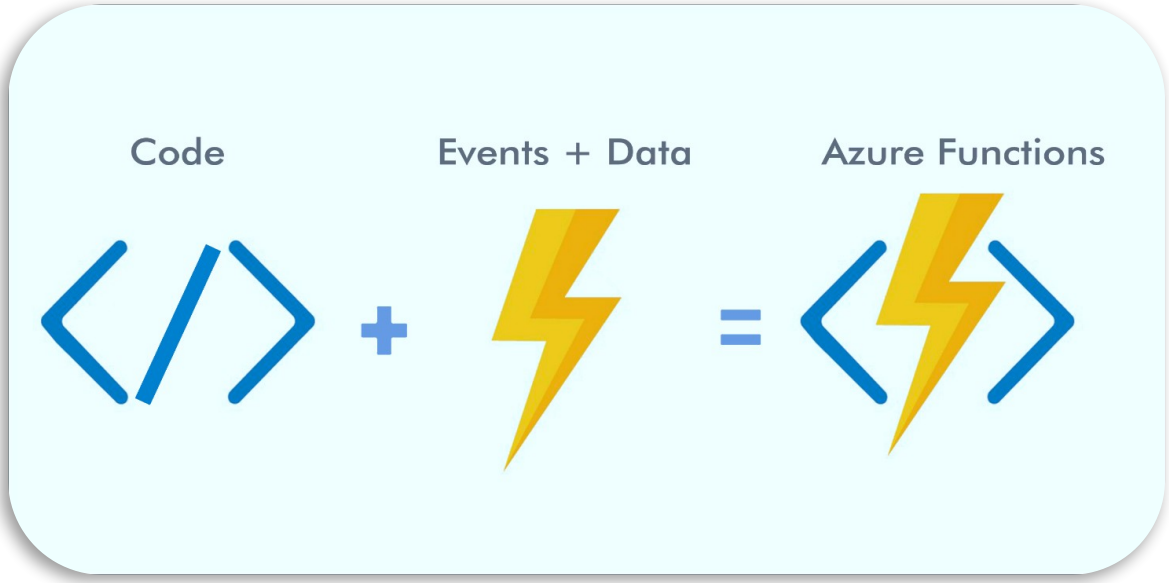

Рисунок 3.17 – Функции Azure

Azure Functions – бессерверная платформа вычислений, позволяющая внедрять бизнес-логику, которая способна выполняться без подготовки инфраструктуры. Код функции можем писать на любом удобном для нас языке, включая F#, C#, JavaScript, Java, Python и PowerShell. Поддерживаются диспетчеры пакетов NuGet и NPM, что позволяет использовать многие библиотеки в бизнес – логике.

3.2.2.2 Преимущества и недостатки решения на основе бессерверных вычислений.

Бессерверные вычисления являются отличным вариантом для размещения кода в облачной инфраструктуре. Среди основных преимуществ такого подхода можно выделить:

- автоматическое масштабирование и простое развертывание;
- отсутствие избыточного выделения инфраструктуры;
- логика без отслеживания состояния;
- управление событиями;
- функции можно использовать в традиционных вычислительных средах.

Используя Azure Functions, получаем автоматическое масштабирование, при этом нам не нужно беспокоиться об управлении серверами. Оплачиваем только реально используемые ресурсы, а не зарезервированное время. Развертывать функции можно с использованием инструментов (например, Visual Studio или Visual Studio Code) или с помощью непрерывного развертывания (с помощью системы управления версиями).

Бессерверные вычисления помогают решать проблему избыточного выделения инфраструктуры за счет автоматического масштабирования.

Экземпляры функций можно создавать или удалять по запросу. Функции без отслеживания состояния подходят для бессерверных вычислений, однако, если требуется отслеживание состояния, то его можно хранить в службе хранилища.

Azure Functions управляются событиями, то есть они выполнятся в ответ на некоторое событие (триггер), например, ответ на HTTP-запрос или добавление сообщения в очередь. Это значительно упрощает базу кода, позволяя объявлять только источники данных (например, триггер или входная привязка) и точку назначения (выходная привязка), при этом не нужно писать код для работы с очередями, BLOB-объектами и прочим, необходимо лишь создать основную бизнес-логику.

Функции – ключевой компонент бессерверных вычислений и одновременно они выполняют роль вычислительной платформы для выполнения кода разного типа.

Но и у такого подхода существуют свои недостатки:

- время выполнения;
- частота выполнения.

Время ожидания по умолчанию составляет 5 минут, но его можно увеличить не более чем до 10 минут. Если выполнение функции может занять более 10 минут, то ее можно разместить на виртуальной машине. Время ожидания может ограничиваться 2,5 минутами, если облачная служба запускается по HTTP-запросу. Существует дополнительная возможность создания устойчивых функций, чтобы «оркестрировать» выполнение нескольких функций без ограничения по времени ожидания.

При масштабировании каждые 10 секунд может создаваться только один экземпляр функции, но не более 200 экземпляров в целом. Каждый экземпляр способен обслуживать несколько одновременных выполнений, но стоит отметить, что разные триггеры имеют различные требования к масштабированию. Поэтому при использовании триггеров необходимо сначала их изучить и связанные с ними ограничения.

3.2.2.3 Привязки Azure Functions.

Привязки – способ подключить к функции Azure данные и службы. Нам не нужно писать код в теле функций для подключения к источникам данных и управления подключениями потому, что привязки умеют взаимодействовать с другими службами. Код функции использует входные привязки как источники данных и выходные привязки как хранилище результатов. Для управления входными и выходными данными каждой функции можно указать нуль и более привязок.

Триггер – разновидность входной привязки, которая позволяет запускать выполнение функции.

3.2.2.4 Триггер BLOB-объектов.

Триггер BLOB-объектов – триггер (приложение-триггер), который позволяет запускать выполнение Azure Function при изменении содержимого хранилища BLOB-объектов Azure (например, при добавлении или изменении файла в хранилище). Для создания триггера BLOB-объектов необходимо указать учетную запись хранения Azure и указать расположение, которое триггер будет отслеживать (в данном случае контейнер с сохраненными в нем изображениями).

3.2.3 База данных Azure SQL

Служба «База данных Azure SQL» – специальная масштабируемая интеллектуальная служба реляционных баз данных, входящая в семейство Azure SQL. Основные достоинства Azure SQL:

- удобство;
- стоимость;
- масштабируемость;
- безопасность;
- один сервер, много баз данных.

При настройке базы данных на локальном компьютере или виртуальной машине нам необходимо хорошо понимать существующие требования к программному и аппаратному обеспечению. Нам так же необходимо познакомиться с новыми рекомендациями по обеспечению безопасности и управлению операционной системой, производить обновления SQL Server, решать проблемы с хранением данных и резервным копированием. При выборе Azure SQL всю эту работу берет на себя облачный сервис, а нам лишь необходимо выбрать имя и задать несколько параметров для базы данных. Все это можно сделать либо через веб-браузер, либо через соответствующие скрипты, что позволяет очень быстро и легко создать базу данных. Экземпляры

базы данных можно удалять или добавлять в любое удобное время. Данный подход позволяет сконцентрировать внимание на разработке отличного приложения, не заботясь о настройке программного обеспечения для базы данных.

Так как все управление базами данных «облако» берет на себя, то не нужно покупать различное оборудование, подключать питание и выполнять действия по обслуживанию базы данных. Microsoft Azure предлагает несколько тарифных планов для базы данных Azure SQL, что позволяет выбрать правильный баланс стоимости и производительности. Стоит отметить, что, выбирая начальный уровень, затраты клиентов не превышают нескольких долларов в месяц.

Облачный сервис позволяет корректировать размер базы данных и производительность при необходимости в любое удобное время для нас, почти мгновенно создавать резервные копии баз данных.

Azure SQL комплектуется брандмауэром, который позволяет контролировать доступ к базе данных. Доступ будут иметь только специальные IP-адреса, которым доверяем. Управлять базой данных (например, создавать сущности, делать запросы) можно через Visual Studio, SQL Server Management Studio, Visual Studio Code.

Во время создания первой базы данных создается для нее логический сервер, с помощью которого можно производить администрирование баз данных: управлять политиками безопасности и учетными данными для входа, редактировать правила брандмауэра. Эти политики можно редактировать для каждой отдельной базы данных, которая размещена на логическом сервере.

3.3 **Реализация и проектирование системы**

3.3.1 Задание

1 этап: разработать систему сбора, хранения и чтения видеоданных.
2 этап: разработать систему обработки видеоданных с помощью алгоритмов или готовых моделей машинного обучения.

3.3.2 Реализация системы

Для реализации первого этапа изначально было необходимо подготовить облачное хранилище для хранения изображений. Для этого была создана ресурсная группа, а в ней – учетные записи хранения (создать все это можно непосредственно с помощью портала или же с помощью заранее написанных скриптов) (Приложение А). В качестве шаблона проектирования был выбран шаблон Producer-Consumer (Рисунок 3.5).

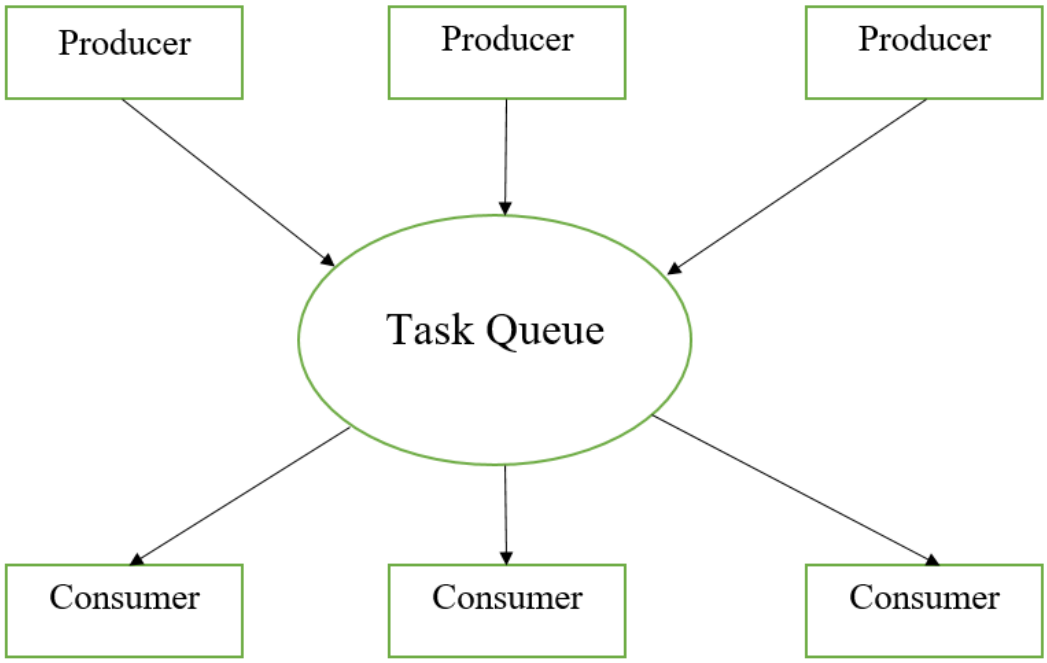

Рисунок 3.18 – Шаблон проектирования Producer-Consumer в общем виде

Для реализации данного шаблона использовались два потока: Producer (изготовитель) и Consumer (исполнитель). Producer (изготовитель) – это некоторый поток, который генерирует «задания» и складывает их в очередь Queue (в проекте использовалась ArrayBlockingQueue – очередь, реализующая классический кольцевой буфер). Consumer (исполнитель) – поток, который «достает» задания из очереди, выполняет и отправляет результаты в соответствующий приемник (для каждого потока был создан отдельный класс). С помощью потока-изготовителя происходит сбор данных и формируется очередь запросов. Благодаря возможностям библиотеки OpenCV осуществляется захват очередного изображения с камеры, и, в том случае, если изображение захвачено корректно и нет никаких других проблем, в очередь поступает новое изображение для последующей обработки. Как только управление передается потоку-изготовителю, начинается сканирование и обработка очереди сообщений. Если в ней есть новые задания, то они обрабатываются потоком-потребителем. Обработка заданий заключается в следующем: создается объект класса CascadeClassifier, конструктор которого на вход принимает один параметр – строку. В данном случае строка – название каскада, который в дальнейшем используется для выявления лиц на изображении (строка может содержать как название одного из каскадов Хаара, так и название lbp каскада). У данного класса существует готовый метод для анализа изображения – detectMultiScale(photo, faceRects), который на вход принимает два параметра: первый параметр – исходное изображение, которое

представляет собой объект класса Mat, второй – объект класса MatofRect. Результат этого метода – матрица прямоугольников, где каждый прямоугольник – область, в пределах которой было распознано лицо (в faceRects будет храниться результат обработки). Если такая матрица не является пустой, то на фото есть лица, их можно выделить и отправить результат для хранения в заранее подготовленное облачное хранилище, иначе можно перейти к обработке следующего сообщения. Связь с «облаком» устанавливается посредством класса AzureManager (данный класс позволяет как загружать данные в облачное хранилище, так и скачивать их из учетной записи хранения) (Приложение Б). Для записи данных в облако заранее необходимо создать в учетной записи хранения контейнер для хранения BLOB-объектов, в который каждое изображение записывается в виде отдельного файла (при этом следует учитывать, что у каждого нового контейнера должно быть уникальное имя, иначе будет выдано сообщение об ошибке, для решения таких проблем можно воспользоваться методом randomUUID() и название контейнера указывать как <имя_контейнера_randomUUID>) (Приложение В). При таком подходе хранения доступ к данным извне получить нельзя, что обеспечивает защиту и конфиденциальность информации (Приложение Г).

Во второй части работы было спроектировано и разработано приложение по идентификации лиц. Данное приложение представляет из себя программу-триггер (BLOB-триггер), которая привязывается к контейнеру, в котором содержатся данные (в данном случае изображения) и работает в том случае, когда содержимое контейнера обновляется (например, добавляются новые изображения) (Приложение Д). На рисунке 3.6 BLOB-триггер отслеживает состояния контейнера firstjavaapp, как только к этому контейнеру будет обращение, триггер отработает и результат будет занесен в контейнер output.

```
{
  "scriptFile": "__init__.py",
  "bindings": [
    {
      "name": "blobin",
      "type": "blobTrigger",
      "direction": "in",
      "path": "firstjavaapp/{blobname}.{blobextension}",
      "connection": "teststor2021_STORAGE"
    },
    {
      "name": "blobout",
      "type": "blob",
      "direction": "out",
      "path": "output/{blobname}_result.jpg",
      "connection": "teststor2021_STORAGE"
    }
  ]
}
```

Рисунок 3.19 – Пример function.json файла для BLOB-триггера

На вход приложению подается изображение, которое в дальнейшем обрабатывается с целью идентификации на нем лиц. Процесс обработки фотографии включает следующие этапы:

1) уменьшить размеры изображения до размеров 700 на 700 пикселей, если изображение большое;
2) выделить все лица на изображении;
3) сравнить выделенные лица с лицами из базы данных с известными лицами;
4) если обнаружено знакомое лицо, то внести новую запись в таблицы «PHOTOS_PERSONS_INFO» и «PHOTOS».

```
[2021-05-01T19:47:01.897Z] Executing 'Functions.BlobTrigger1' (Reason='New blob detected: firstjavaapp/Vad2.jpg', Id=c6608226-9253-4833-84f3-2c49ad
60c86a)
[2021-05-01T19:47:01.905Z] Trigger Details: MessageId: 98686d9d-44b0-4df9-8ae0-c4155c8f51d9, DequeueCount: 1, InsertionTime: 2021-05-01T19:47:00.00
0+00:00, BlobCreated: 2021-05-01T19:46:53.000+00:00, BlobLastModified: 2021-05-01T19:46:53.000+00:00
[2021-05-01T19:47:02.011Z] --- Python blob trigger function processed blob
----- Name: firstjavaapp/Vad2.jpg
----- Blob Size: 133971 bytes
[2021-05-01T19:47:15.310Z] ----- Processing image successful firstjavaapp/Vad2.jpg
```

Рисунок 3.20 – Логирование триггер-программы об успешном принятии и обработке изображения

Выделение лиц на фотографиях можно осуществлять с помощью готовых алгоритмов таких, как HOG, метод Виолы-Джонса или с использованием сверточной нейронной сети (CNN). Самое точное распознавание лиц на изображении может быть достигнуто при использовании CNN, однако данный подход требует больших вычислительных ресурсов, поэтому в данном

приложении использовался алгоритм HOG в силу его быстродействия и довольно хорошей точности. Каждое распознанное лицо на фотографии выделяется синим квадратом и подписывается своим именем, если лицо заранее неизвестно, то оно подписывается как «Stranger». Обработанное изображение хранится в облачном контейнере (Приложение Е). При проектировании базы данных учитывался тот факт, что на одном изображении могут быть несколько людей, и один человек может быть на нескольких фотографиях. В связи с этим была выбрана связь многие-ко-многим между таблицами. При таком подходе создаются три таблицы: две таблицы – «источники» («PERSONS» и «PHOTOS») и одна вспомогательная таблица («PHOTOS_PERSONS_INFO»). ER-диаграмма базы данных показана на рисунке 3.8.

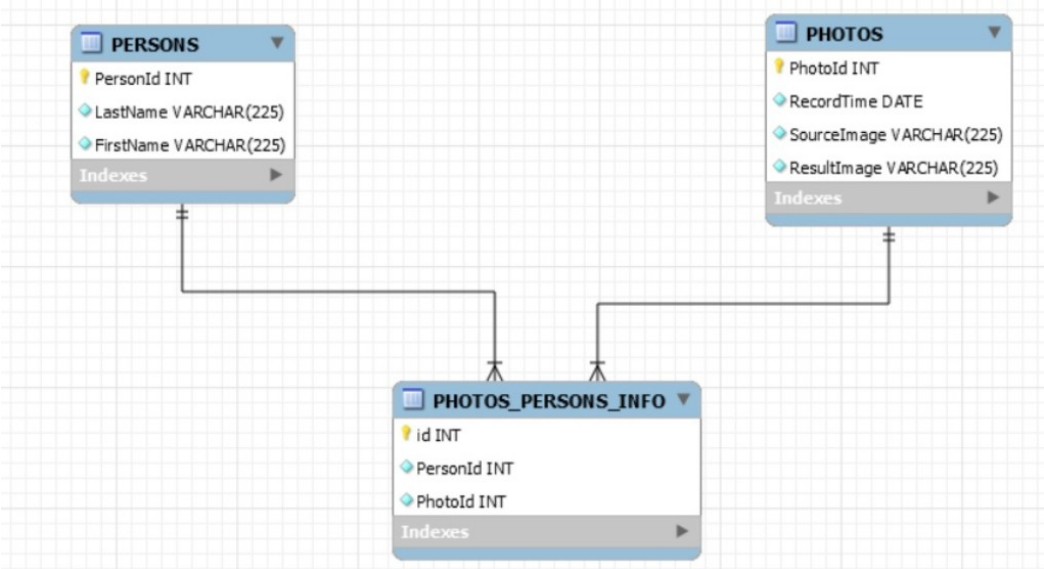

Рисунок 3.21 – ER-диаграмма базы данных

Пример программного кода для добавления новой записи в таблицы «PHOTOS» и «PHOTOS_PERSONS_INFO» приведен на рисунке 3.9.

```
def write_db_record(faces_id, curtime, image_name):
    with pyodbc.connect('Driver=' + DRIVER +';Server=tcp:' + SERVER + ',1433;Database=db1;Uid=' + USERNAME
+';Pwd=' + PASSWORD + ';Encrypt=yes;TrustServerCertificate=no;Connection Timeout=30;') as conn:
        with conn.cursor() as cursor:
            pic_id = str(cursor.execute(SELECT_PHOTO_ID).fetchone()[0])
            cursor.execute("INSERT INTO " + PHOTOS_DB + " VALUES(CAST('" + curtime + "' AS DATETIME), '" +
            image_name + "', '" + image_name.replace(".jpg", "_result.jpg") + "');")
            for person_id in faces_id:
                cursor.execute("INSERT INTO " + PHOTOS_PERSONS_DB + " VALUES(" + str(person_id) + ", " + pic_id + ");")
```

Рисунок 3.22 – Пример добавления новой записи в базу данных

Программа-триггер может одновременно обрабатывать несколько фотографий. Для этого необходимо в файле local.settings.json в поле «PYTHON_THREADPOOL_THREAD_COUNT» задать число от 1 до 32, которое указывает число потоков, задействованных в обработке новых изображений (Рисунок 3.10).

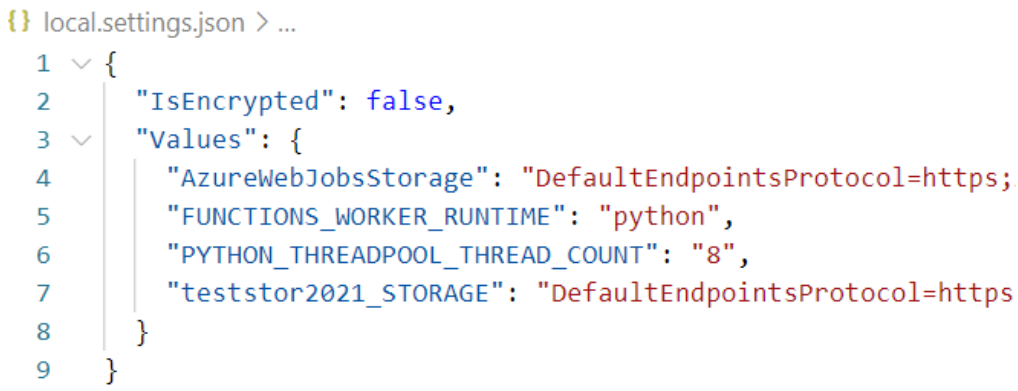

Рисунок 3.23 – Пример задания значения «PYTHON_THREADPOOL_THREAD_COUNT»

ВЫВОДЫ:

1. Разработана система сбора, хранения и чтения видеоданных с видеокамеры, встроенной в ноутбук.
2. Разработана программа-триггер, позволяющая идентифицировать лица на изображениях и сохранять результаты обработки в облачное хранилище.

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломной работы были изучены и решены следующие задачи:

- рассмотрены существующие системы и комплексы видеоаналитики, проведен их сравнительный анализ;
- рассмотрены и изучены технологии граничных и облачных вычислений;
- изучены библиотеки, алгоритмы и средства по распознаванию и идентификации лиц на изображениях: OpenCV, метод Виолы-Джонса, CNN, HOG и Microsoft Cognitive Services;
- рассмотрены и изучены компоненты Microsoft Azure: базы данных Azure, облачное хранилище, бессерверные вычисления и функции Azure;
- спроектирована и разработана программа по обработке видеоданных и приложение-триггер по идентификации лиц.

При выборе алгоритма для идентификации лиц можно руководствоваться такими метриками, как быстродействие и точность. С точки зрения скорости HOG является самым быстрым алгоритмом, за которым следуют метод Виолы-Джонса и CNN. Самым точным является CNN, но он требует больших вычислительных ресурсов, поэтому при использовании сверточной нейронной сети все вычисления лучше всего проводить на графическом процессоре. HOG работает довольно хорошо, но существуют проблемы с распознаванием маленьких лиц (Приложение Ж). Классификаторы HaarCascade в методе Виолы-Джонса работают так же хорошо, как и HOG, однако обучаются они дольше. Быстродействие каждого алгоритма зависит от качества изображения: чем качественнее изображение, тем быстрее и точнее будет произведена его обработка. Значительные видоизменения лица делают распознавание маловероятным, однако для самых типовых ситуаций, например, когда человек одел очки, моргнул, засмеялся, алгоритмы смогут его правильно идентифицировать. Цветность изображений не помогает, но и не мешает распознаванию и идентификации лиц: большинство алгоритмов и методов работает с черно-белыми изображениями.

Также существуют пути по улучшению текущий версии системы:

- рассмотреть другие средства и алгоритмы по распознаванию и идентификации лиц в видеопотоке;
- использовать платформу EdgeX для подключения других камер, чтобы каждый раз при добавлении новой камеры не выполнять повторно сборку приложения;

- на вход приложению-триггеру передавать не изображение, а видео, которое в последующем обрабатывать, для оптимизации хранения информации в облачном хранилище.

Результаты исследования были представлены на 77-ой и 78-ой Научной конференции студентов и аспирантов БГУ.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Выделение объектов на изображении по методу Виолы – Джонса (Object selection in the image according to the Viola – Jones method) [Электронный ресурс] – Режим доступа: https://api-2d3d-cad.com/viola-jones-method/#1. – Дата доступа: 07.02.2020

2. Граничные вычисления “Edge computing” – будущее индустрии [Электронный ресурс] – Режим доступа: https://2g3g4g5g.ru/mec-edge-computing-iot/. – Дата доступа: 19.09.2019

3. Коллиер, М. Основы Microsoft Azure / М.Коллиер, Р.Шаан. – 2-е изд. – Редмонд: Microsoft Press, 2016. – 263 с.

4. Метод Виолы – Джонса (Viola – Jones) как основа для распознавания лиц [Электронный ресурс] – Режим доступа: https://habr.com/ru/post/133826/. – Дата доступа: 09.02.2020

5. Метод компрессии DVPack 2 [Электронный ресурс] – Режим доступа: https://algoritm.org/arch/07_4/07_4_14.pdf. – Дата доступа: 21.03.2021

6. Метод распознавания лиц Виолы – Джонса (Viola – Jones) [Электронный ресурс] – Режим доступа: https://oxozle.com/2015/04/11/metod-raspoznavaniya-lic-violy-dzhonsa-viola-jones/. – Дата доступа: 12.02.2020

7. Обнаружение автомобильных номеров в видео с помощью классификатора SVM и дескриптора HOG [Электронный ресурс] – Режим доступа: https://delirium-00.livejournal.com/1872.html. – Дата доступа: 16.02.2021

8. Обучение машины – забавная штука: современное распознавание лиц с глубинным обучением [Электронный ресурс] – Режим доступа: https://habr.com/ru/post/306568/. – Дата доступа: 10.09.2020

9. Официальная документация библиотеки OpenCV [Электронный ресурс] – Режим доступа: https://docs.opencv.org/. – Дата доступа: 14.11.2019

10. Прохоренок, Н.А. OpenCV и Java. Обработка изображений и компьютерное зрение / Н.А. Прохоренок. – СПб.: БХВ-Петербург, 2018. – 320 с.

11. Распознавание и обнаружение лиц с использованием Python OpenCV [Электронный ресурс] – Режим доступа: https://dev-gang.ru/article/raspoznavanie-i-obnaruzhenie-lic-s-ispolzovaniem-python-opencv-6woirp3dw8/. – Дата доступа: 15.11.2020

12. Сверточная нейронная сеть – простое объяснение CNN и её применение [Электронный ресурс] – Режим доступа: https://evergreens.com.ua/ru/articles/cnn.html. – Дата доступа: 15.10.2020

13. Сверточные нейронные сети – Википедия [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/Сверточная_нейронная_сеть. – Дата доступа: 20.10.2020

14. QuickStart: Manage blobs with Java v12 SDK [Electronic resource] / Documentation for Azure – Mode of access: https://docs.microsoft.com/en-us/azure/storage/blobs/storage-quickstart-blobs-java. – Date of access: 22.03.2020

15. What is the Azure Face service? [Electronic resource] / Documentation for Azure – Mode of access: https://docs.microsoft.com/en-us/azure/cognitive-services/face/overview. – Date of access: 20.10.2020

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ А

Создание ресурсной группы и учетной записи хранения данных с использованием PowerShell

```
PS /home/romanyuk-vad> $resgroup = "picgroup"
PS /home/romanyuk-vad> $location = "westeurope"
PS /home/romanyuk-vad> New-AzResourceGroup -Name $resgroup -Location $location

ResourceGroupName : picgroup
Location          : westeurope
ProvisioningState : Succeeded
Tags              :
```

Рисунок 24 – Создание ресурсной группы

```
PS /home/romanyuk-vad> New-AzStorageAccount -ResourceGroupName $resgroup `
>>   -Name "picdata" `
>>   -SkuName Standard_LRS `
>>   -Location $location `
>>

StorageAccountName ResourceGroupName PrimaryLocation SkuName      Kind      AccessTier CreationTime       ProvisioningState EnableHttpsTrafficOnly LargeFileShares
------------------ ----------------- --------------- -------      ----      ---------- ------------       ----------------- ---------------------- ---------------
picdata            picgroup          westeurope      Standard_LRS StorageV2 Hot        5/2/2020 1:49:46 PM Succeeded         True
```

Рисунок 25 – Создание учетной записи хранения

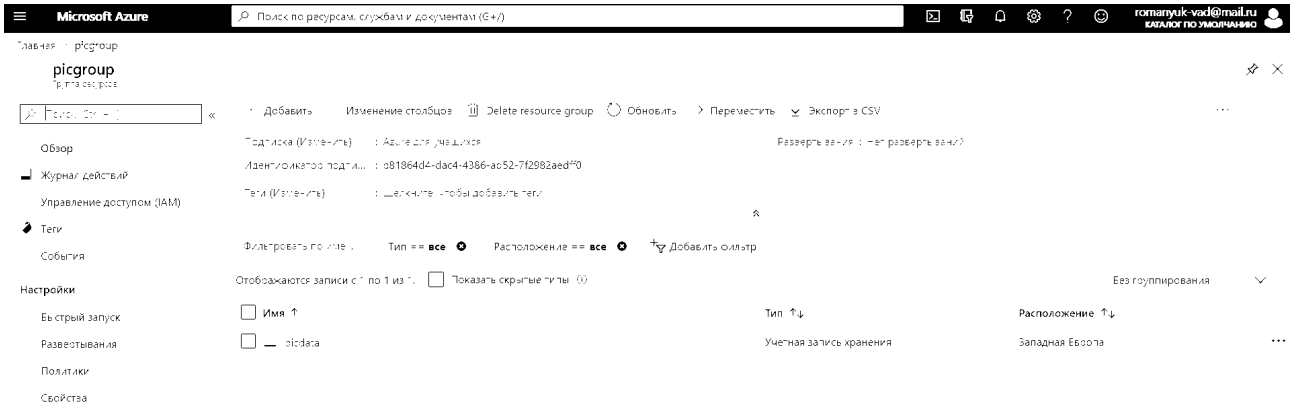

Рисунок 26 – Пример группы ресурсов

ПРИЛОЖЕНИЕ Б

Классы для взаимодействия с облачным хранилищем и анализа изображений

```
public class AzureManager {
    private final String CONNECTSTR = "insert connection string";
    private BlobServiceClient blobServiceClient;
    private BlobContainerClient containerClient;
    public AzureManager(String containerName)
    {
        this.blobServiceClient = new BlobServiceClientBuilder().connectionString(CONNECTSTR).buildClient();
        String offContainerName = containerName.toLowerCase() + java.util.UUID.randomUUID();
        this.containerClient = blobServiceClient.createBlobContainer(offContainerName);
    }

    public void UploadData(String filePath, String filename)
    {
        BlobClient blobClient = this.containerClient.getBlobClient(filename.toLowerCase());
        System.out.println("\nUploading to Blob storage as blob:\n\t" + blobClient.getBlobUrl());
        blobClient.uploadFromFile(filePath);
        System.out.println("Uploading file finished!");
    }
    public void                    ()
    {
      for(BlobItem item : containerClient.listBlobs())
          System.out.println(item.getName());
    }
    public void             (String filePath, String filename)
    {
        File                = new File(                filePath + "DOWNLOAD" + filename);
        System.out.println("\nDownloading blob to\n\t " + filePath + filename);
        BlobClient blobClient = this.containerClient.getBlobClient(filename.toLowerCase());
        blobClient.downloadToFile(                filePath + "DOWNLOAD" +filename);
        System.out.println("Done");
    }
```

Рисунок 27 – Класс Azure Manager

```
public class CascadeAnalyzer {
    private CascadeClassifier facedetector;
    public CascadeAnalyzer(String methodName)
    {
        facedetector = new CascadeClassifier(methodName);
    }

    public boolean isDetectFace(Mat photo)
    {
        MatOfRect faceRects = new MatOfRect();
       facedetector.detectMultiScale(photo, faceRects);
       Rect[] faces = faceRects.toArray();
       if(faces.length < 1) return false;
       for(Rect r : faces)
       {
           Imgproc.rectangle(photo, r, new Scalar(40, 255, 0),  thickness 3);
       }
       return true;
    }
}
```

Рисунок 28 – Класс для анализа изображений

ПРИЛОЖЕНИЕ В

Пример облачного контейнера

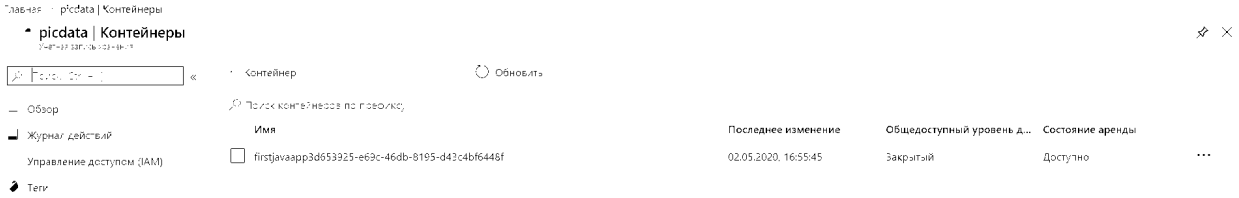

Рисунок 29 – Пример контейнера для хранения изображений

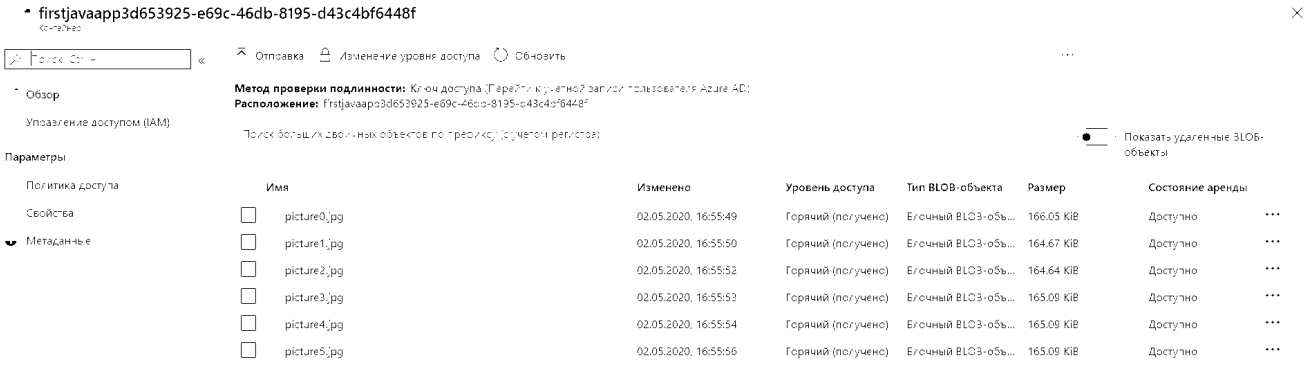

Рисунок 30 – Сохраненные изображения в облачном контейнере

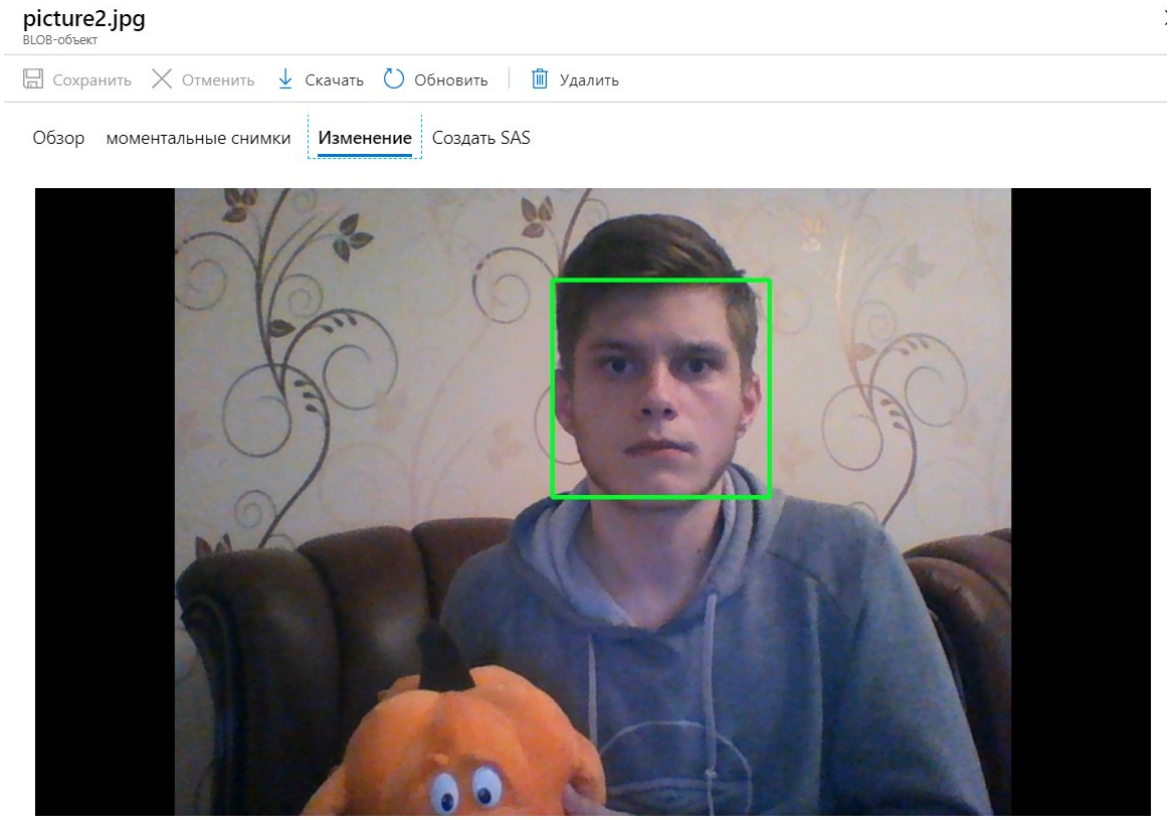

Рисунок 31 – Пример хранения изображения в облачном контейнере

ПРИЛОЖЕНИЕ Г

Проверка безопасности облачного контейнера

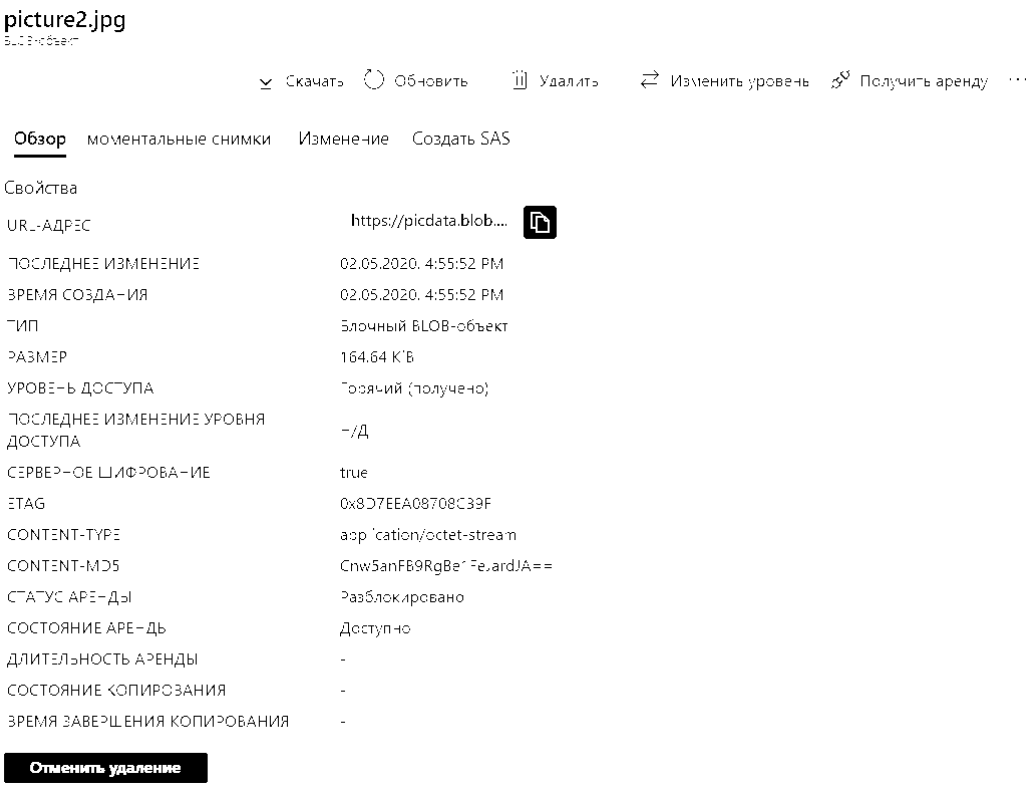

Рисунок 32 – Проверка безопасности контейнера

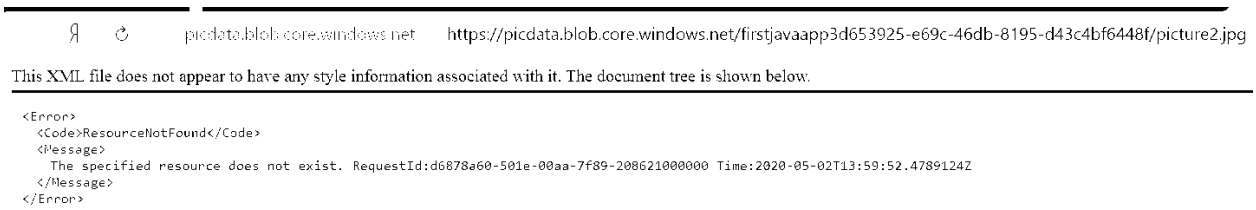

Рисунок 33 – Доступ извне к содержимому контейнера получить не удалось

ПРИЛОЖЕНИЕ Д

Пример программы – триггера

```
import logging
import io
import sys
from PIL import Image
import azure.functions as func
import azure.storage.blob as storblob

import threading
import numpy as np
import face_recognition
import cv2
import glob
import datetime
import pyodbc

    # final image composite size
FINAL_COMPOSITE_MAX_HEIGHT = 700
FINAL_COMPOSITE_MAX_WIDTH = 700
lock = threading.Lock()

    # date, path to local db
UNKNOWN = "Stranger"
PICPATH = "KnownFaces\\"

    # constants for azure db
SERVER = 'bd1server.database.windows.net'
PHOTOS_DB = 'PHOTOS(RecordTime, SourceImage, ResultImage)'
PHOTOS_PERSONS_DB = 'PHOTOS_PERSONS_INFO(PersonId,PhotoId)'
USERNAME = 'badadmin'
PASSWORD = 'Mdevil2008'
CONTAINER_IN = 'firstjavaapp/'
DRIVER = '{ODBC Driver 17 for SQL Server}'
SELECT_PHOTO_ID = "SELECT CASE WHEN (SELECT COUNT(1) FROM PHOTOS) = 0 THEN 1
ELSE IDENT_CURRENT('PHOTOS') + 1 END;"


class Person:
    firstName = ""
    lastName = ""
    id_ = 0

    def __init__(self, firstName_, lastName_, id_):
        self.firstName = firstName_
        self.lastName = lastName_
```

```
        self.id_ = id_


def set_image_sizes(base_image):

    if base_image.width > FINAL_COMPOSITE_MAX_WIDTH or base_image.height > F
INAL_COMPOSITE_MAX_HEIGHT:
        if base_image.height > base_image.width:
            factor = 900 / base_image.height
        else:
            factor = 900 / base_image.width
        base_image = base_image.resize((int(base_image.width * factor), int(
base_image.height * factor)))

    return base_image

def get_localdb_images():
    res = []
    for file in glob.glob(PICPATH + "*.jpg"):
        res.append(file)
    return res

def get_known_names(picfiles):
    res = []
    i = 1
    for file in picfiles:
        arr = file.replace(".jpg","").replace(PICPATH,"").split("_")
        res.append(Person(arr[0], arr[1], i))
        i += 1
    return res

def get_known_faces_info(picfiles):
    res = []
    for picture in picfiles:
        image = face_recognition.load_image_file(picture)
        face_encoding = face_recognition.face_encodings(image,known_face_loc
ations=None,num_jitters=1,model="small")[0]
        res.append(face_encoding)
    return res

def match_all_faces(rgb_frame):
    face_locations = face_recognition.face_locations(img=rgb_frame,number_of
_times_to_upsample=1,model="hog")
    face_encodings = face_recognition.face_encodings(rgb_frame, face_locatio
ns)
    return face_encodings, face_locations

def identify_faces(face_encodings, known_faces, known_faces_name):
    face_names = []
    db_faces = []
```

```
    if len(face_encodings) > 0:
        for face in face_encodings:
            match = face_recognition.compare_faces(np.array(known_faces),np.
array(face),tolerance=0.6)
            if True in match:
                first_match_index = match.index(True)
                face_names.append(known_faces_name[first_match_index].firstN
ame)
                db_faces.append(known_faces_name[first_match_index].id_)
            else:
                face_names.append(UNKNOWN)
    return face_names, db_faces

def highlight_faces(rgb_frame, face_locations, face_names):
    i = 0
    for top, right, bottom, left in face_locations:
        cv2.rectangle(rgb_frame, (left, top), (right, bottom), (0, 0, 255),
2)
        font = cv2.FONT_HERSHEY_DUPLEX
        cv2.putText(rgb_frame, face_names[i].split('_')[0], (left + 6, botto
m - 6), font, 0.5, (255, 255, 255), 1)
        i += 1
    return rgb_frame

def process_image(rgb_frame, known_faces, known_faces_name):
    face_encodings, face_locations = match_all_faces(rgb_frame)
    face_names, db_faces = identify_faces(face_encodings, known_faces, known
_faces_name)
    flag = False
    if len(face_names) > 0:
        flag = True
        rgb_frame = highlight_faces(rgb_frame,face_locations,face_names)
    return rgb_frame, flag, db_faces

def write_db_record(faces_id, curtime, image_name):
    with pyodbc.connect('Driver=' + DRIVER +';Server=tcp:' + SERVER + ',1433
;Database=db1;Uid=' + USERNAME
 +';Pwd=' + PASSWORD + ';Encrypt=yes;TrustServerCertificate=no;Connection Ti
meout=30;') as conn:
        with conn.cursor() as cursor:
            pic_id = str(cursor.execute(SELECT_PHOTO_ID).fetchone()[0])
            cursor.execute("INSERT INTO " + PHOTOS_DB + " VALUES(CAST('" + c
urtime + "' AS DATETIME), '" +
            image_name + "', '" + image_name.replace(".jpg", "_result.jpg")
+ "');")
            for person_id in faces_id:
                cursor.execute("INSERT INTO " + PHOTOS_PERSONS_DB + " VALUES
(" + str(person_id) + ", " + pic_id + ");")
```

```
def upload_res_image(rgb_frame, blobout):
    img_byte_arr = io.BytesIO()
    Image.fromarray(rgb_frame).save(img_byte_arr, format="JPEG")
    blobout.set(img_byte_arr.getvalue())

def main(blobin: func.InputStream, blobout: func.Out[bytes], context: func.C
ontext):
    logging.info(f"--- Python blob trigger function processed blob \n"
                 f"----- Name: {blobin.name}\n"
                 f"----- Blob Size: {blobin.length} bytes")
    input_image = blobin

    try:
        base_image = Image.open(input_image)
    except OSError as e:
        print(f'EXCEPTION: Unable to read input as image. {e}')
        sys.exit(254)
    except Exception as e:
        print(f'EXCEPTION: {e}')
        sys.exit(255)

        # resize base image if too large

    base_image = set_image_sizes(base_image)

        # get base images and names from local db

    picfiles = get_localdb_images()
    known_faces_name = get_known_names(picfiles)

        # convert PIL image into cv2 image
    rgb_frame = np.array(base_image.convert("RGB"))

        # a list of 128-dimensional face encodings
    known_faces = get_known_faces_info(picfiles)


    rgb_frame, is_faces_on_image, db_faces = process_image(rgb_frame,known_f
aces,known_faces_name)
    if is_faces_on_image:
        upload_res_image(rgb_frame, blobout)
        if len(db_faces) > 0:
            lock.acquire()
            curtime = datetime.datetime.now().strftime("%m/%d/%y %H:%M:%S")
            write_db_record(db_faces,curtime, blobin.name.replace(CONTAINER_
IN,""))
```

```
            lock.release()

    logging.info(f"----- Processing image successful {blobin.name}")
```

ПРИЛОЖЕНИЕ Е

Пример работы программы – триггера

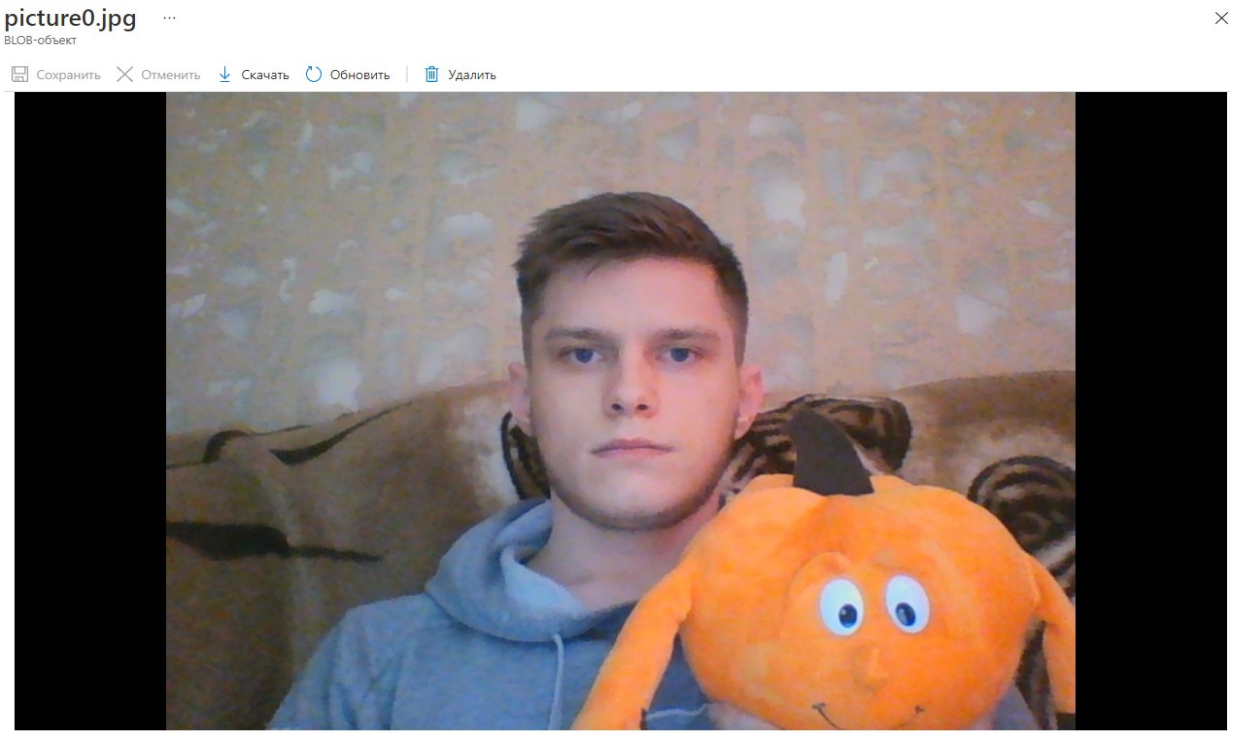

Рисунок 34 – Новое изображением с зафиксированным лицом

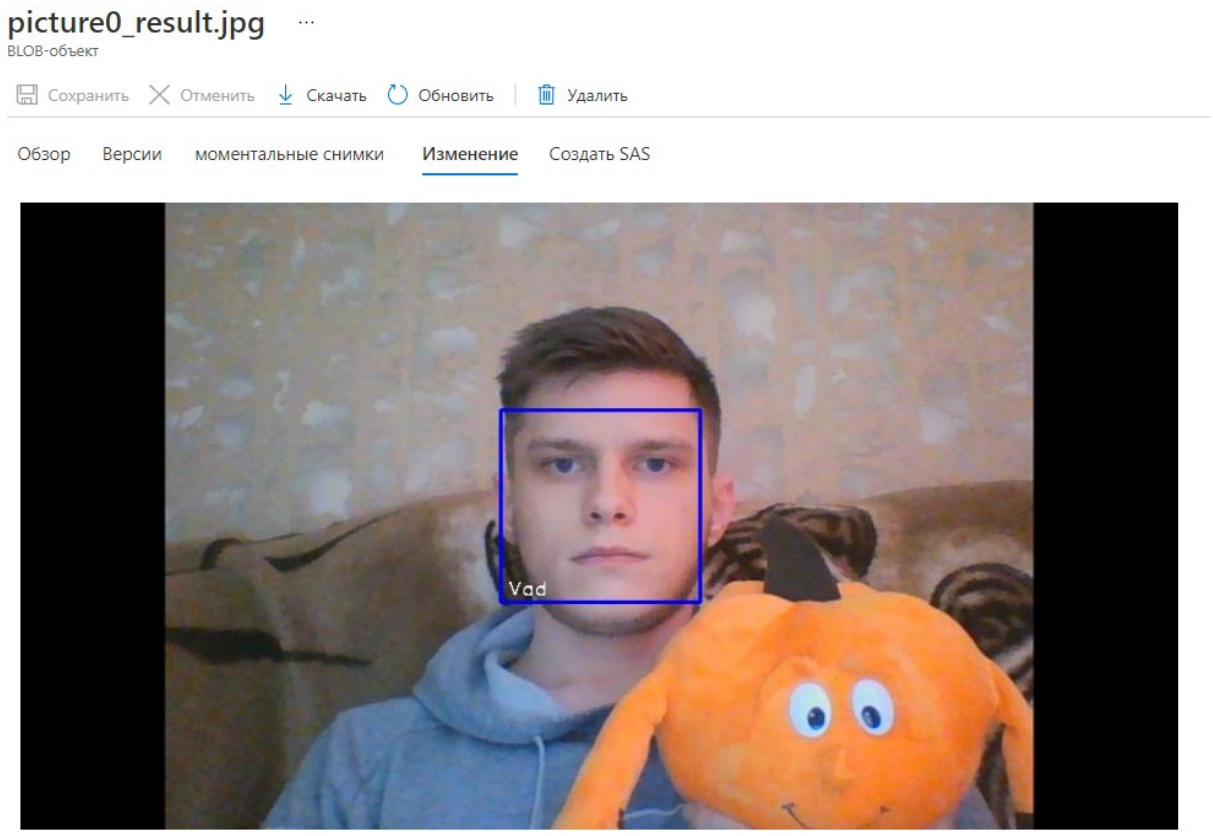

Рисунок 35 – Обработанное изображение с идентифицированным лицом, хранящееся в облачном контейнере

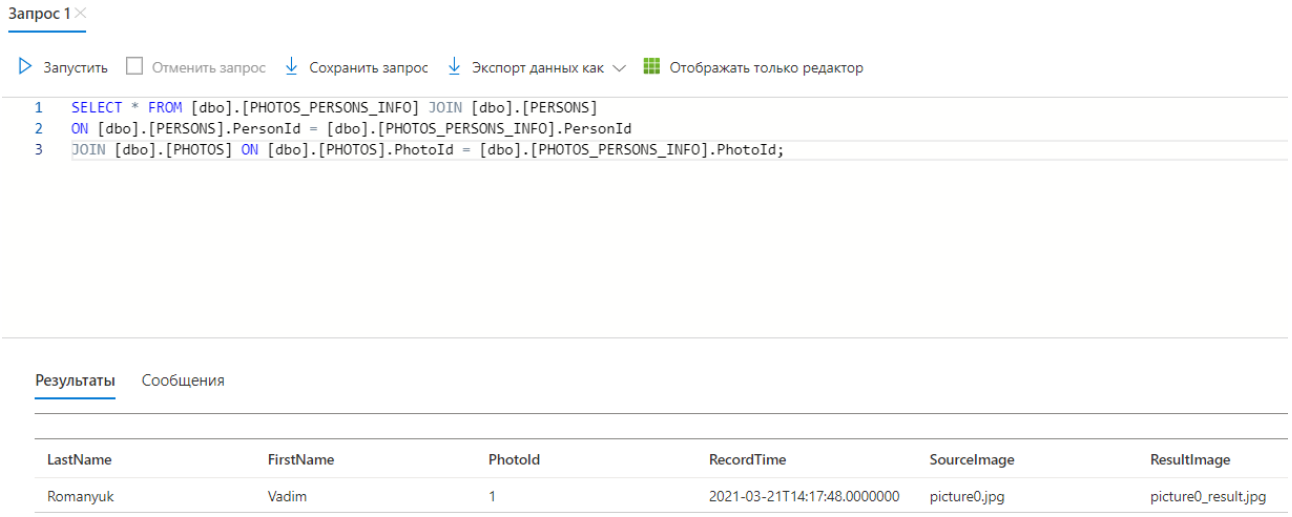

Рисунок 36 – Запись в базе данных, содержащая информацию о лицах, которые были идентифицированы на изображении

ПРИЛОЖЕНИЕ Ж

Сравнение HOG и CNN по быстродействию и точности

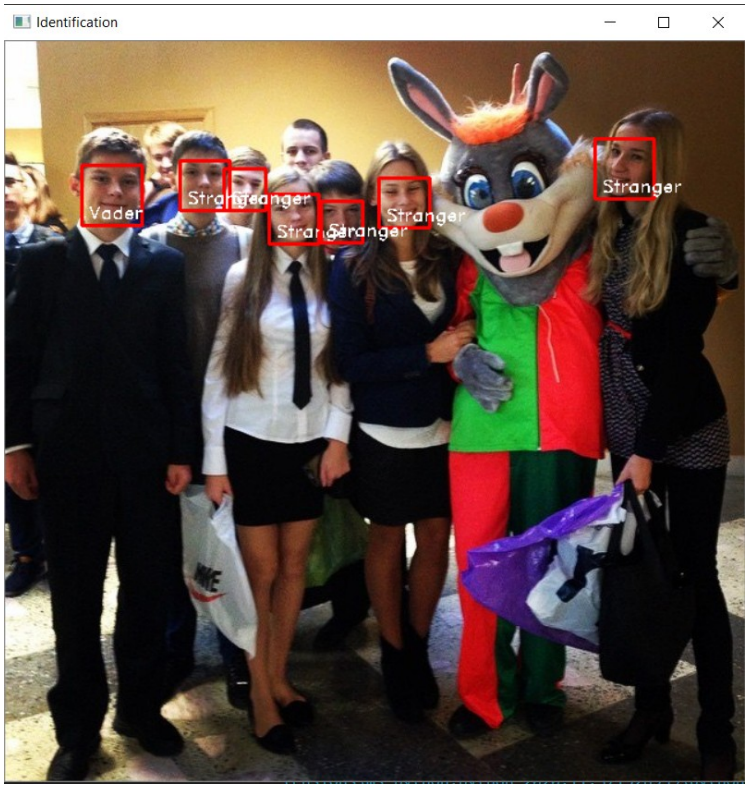

Рисунок 37 – HOG справился за 9 секунд

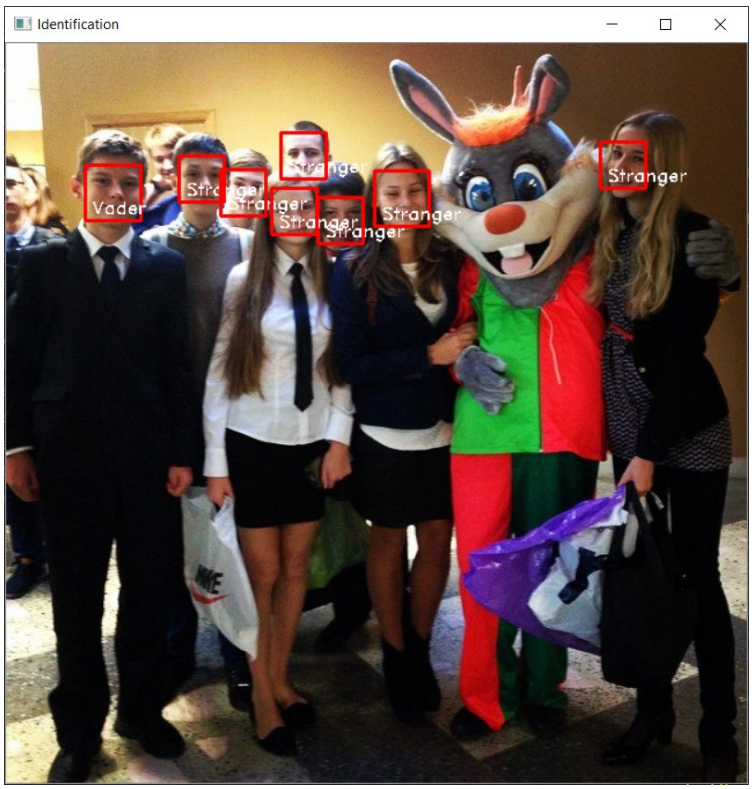

Рисунок 38 – CNN справилась за 48 секунд

Все вычисления проводились на процессоре Intel Core i7 – 9750H 2.6 GHz.