МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ НАУК
Кафедра программного обеспечения

РЕКОМЕНДОВАНО К ЗАЩИТЕ В ГЭК
Заведующий кафедрой
к.т.н., доцент
_____ М.С. Воробьева
25 июня 2022 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
магистерская диссертация

РАЗРАБОТКА АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ ВЫЯВЛЕНИЯ ЛИЦ, НЕ ИСПОЛЬЗУЮЩИХ СРЕДСТВА ИНДИВИДУАЛЬНОЙ ЗАЩИТЫ, НА ОСНОВЕ АНАЛИЗА ВИДЕОРЯДА МЕТОДАМИ МАШИННОГО ОБУЧЕНИЯ

02.04.03 Математическое обеспечение и администрирование информационных систем
Магистерская программа «Разработка технологий Интернета вещей и больших данных»

Выполнил работу
студент 2 курса
очной формы обучения — Устелемов Максим Алексеевич

Научный руководитель
к.т.н., доцент — Воробьева Марина Сергеевна

Рецензент
д.ф.-м.н., заместитель
директора по развитию
Института математики
и компьютерных наук ТюмГУ — Шевляков Артём Николаевич

Тюмень
2022 год

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

На территории организаций с вредными условиями труда и при особой эпидемиологической обстановке работодатель должен обеспечить ношение средств индивидуальной защиты (далее – СИЗ) работниками в соответствие с нормами, устанавливаемыми Правительством Российской Федерации [26]. Контроль применения СИЗ в организациях с массовым пребыванием людей зачастую выполняется сотрудниками службы безопасности, совмещающими данную функцию с другими своими обязанностями по обеспечению контроля правопорядка.

Проводимый визуальный контроль наличия СИЗ у посетителей приводит к увеличению нагрузки на ответственных сотрудников и возможности невыявления нарушений.

Автоматизированная система, обрабатывающая видеопоток с камер наблюдения на контрольно-пропускных пунктах (далее – КПП), позволит фиксировать факты нарушения без участия сотрудников безопасности, которым потребуется только провести мероприятия непосредственно по устранению нарушения.

В данной работе в качестве СИЗ рассматриваются маски – санитарно-гигиенические изделия, надеваемые на лицо человека, закрывающие как правило рот и нос и выполняющие функцию защиты органов дыхания от инфекций, распространяемых воздушно-капельным путем. Ношение масок является обязательным для работников производств с опасностью для верхних дыхательных путей, для персонала медицинских учреждений, а также для работников и посетителей мест общественного пребывания в период эпидемий острых респираторных вирусных заболеваний.

Цель выпускной квалификационной работы – разработать автоматизированную систему выявления лиц, не использующих СИЗ, на основе анализа видеоряда методами машинного обучения.

Поставлены следующие задачи, способствующие достижению цели выпускной квалификационной работы:

1. Проведение обзора существующих решений к определению наличия СИЗ на изображении.

2. Определение требований к разрабатываемой системе с учетом видеопотока как входных данных.

3. Анализ подходов к детектированию и сопровождению лиц на видеоряде.

4. Подготовка набора данных на основе видеоряда, получаемого с камеры наблюдения.

5. Проведение сравнительного анализа моделей классификации наличия СИЗ на кадре лица.

6. Разработка и тестирование автоматизированной системы выявления лиц, не использующих СИЗ.

ГЛАВА 1. ОБЗОР СУЩЕСТВУЮЩИХ РЕШЕНИЙ ПО ОПРЕДЕЛЕНИЮ НАЛИЧИЯ СРЕДСТВ ИНДИВИДУАЛЬНОЙ ЗАЩИТЫ НА ИЗОБРАЖЕНИИ

Задача определения наличия СИЗ на лице является частным случаем детектирования объектов. Алгоритм детектирования СИЗ на изображении состоит из двух этапов:

1) детектирование самого лица на изображении;

2) бинарная классификация наличия или отсутствия СИЗ на детектированном лице.

В связи с пандемией коронавирусной инфекции COVID-19 и последующим введением обязательного ношения санитарно-гигиенических масок людьми в общественных местах появилось много работ по разработке систем, связанных с определением наличия СИЗ у одного или нескольких лиц на изображениях.

В работе [1] для классификации наличия СИЗ была предложена двухкомпонентная система: модель ResNet-50 для извлечения признаков и ансамбль методов (деревья решений и метод опорных векторов) для решения задачи классификации.

Для обучения авторы использовали следующие датасеты:

- RMFD (Real-World Masked Face Dataset): состоит из 5000 изображений лиц с СИЗ и 90000 изображений лиц без СИЗ. Авторами случайно образом для сбалансированного датасета было отобрано 5000 изображений лиц с отсутствием СИЗ. Датасет использовался для обучения и тестирования. Примеры изображений, взятых из [9], приведены на рисунке 1.

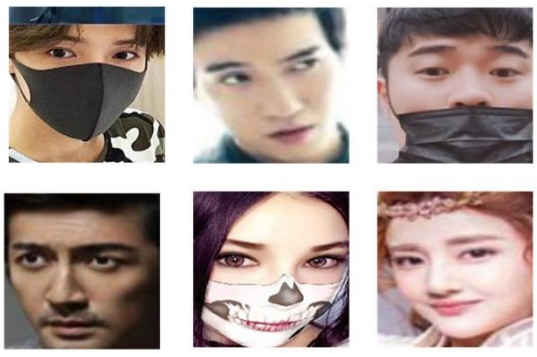

Рис. 1. Примеры изображений в наборе RMFD

- SMFD (Simulated Masked Face Dataset): всего содержит 1570 изображений, из них 785 – изображения лиц без СИЗ, 785 – изображения с лицами, на которые наложена программным способом белая медицинская маска. SMFD использовался для обучения и тестирования. Примеры изображений датасета представлены на рисунке 2.

Рис. 2. Примеры изображений в наборе SMFD

- LFW (Labeled Faces in the Wild): содержит 13000 лиц в голубых медицинских масках, которые наложены программным способом авторами статьи. Используется только для тестирования системы. Примеры изображений представлены на рисунке 3.

Рис. 3. Примеры изображений в наборе LFW

В результате авторы достигли значения метрики Accuracy модели 0.9949, 0.9964 и 1 на тестовых выборках датасетов RMFD, SMFD и LFW соответственно. Однако используемые датасеты состояли только из изображений высокого качества с четко различимыми лицами и в том числе включали в себя изображения, где маска была наложена программным образом, что неприменимо для разработки систем, анализирующих изображения в сложных

неконтролируемых условиях: видеонаблюдении под определенным ракурсом, присутствии многих человек на видеоряде и т.д.

Авторы статьи [18] предложили использовать легковесные архитектуры сверточных нейронных систем Single Shot Multibox Detector (SSD) в качестве алгоритма обнаружения лица и MobileNetV2 в качестве классификатора наличия СИЗ на обнаруженных лицах, благодаря чему разработанная система SSDMNV2 способна работать в режиме реального времени.

Принцип работы SSD заключается в том, что обрабатывается все изображение целиком, предсказываются координаты прямоугольных рамок, ограничивающих детектируемые объекты, а также вероятности принадлежности к определенным классам. Чтобы уточнить совершенные предсказания, производится подавление не-максимумов (Non-Maximum Suppression).

На наборе данных, собранном авторами из специализированных датасетов, размещенных в открытом доступе, достигнута Accuracy 0.9264.

В решении для определения медицинских масок в операционных залах [11] используется комбинация из детектора лица и детектора самой маски с двумя этапами: обработка черно-белого изображения и обработка изображения с примененным цветовым фильтром.

За основу был взят датасет изображений лиц, в котором выборка с наличием маски сформирована на четверть из изображений в свободном доступе, а на остальные изображения лиц была наложена медицинская маска программным методом. Accuracy системы достигает 0.95. Скорость работ системы составляет 10 FPS с возможностью работы до 20 FPS при использовании предобработки изображения (удаление фона).

В работе [7] для определения наличия СИЗ на кадре лица использовались ResNet-50 для извлечения признаков и YOLOv2 для классификации. Модель обучалась на наборах данных, содержащих изображения с присутствием от одного до нескольких лиц. При использовании оценки качества детекции IoU и оптимизатора Adam достигнута Accuracy 0.81.

Система RetinaFaceMask представлена в работе [5]. RetinaFaceMask является детектором правильного ношения медицинских масок на лице. В качестве датасета использовался переразмеченный авторами набор изображений из открытого доступа (для добавления класса неправильного ношения), съемка в которых производилась с положением камеры на уровне лица.

Данная система одноэтапная, то есть происходит детектирование непосредственно СИЗ без подготовительной обработки анализируемого изображения. В качестве классификатора используется только одна архитектура RetinaFace, использующая так называемую пирамиду признаков (Feature Pyramid) – подход с извлечением карт признаков с помощью слоев сверточной нейронной сети с уменьшающимися размерностями. Такой подход позволяет получить семантическую информацию высокого и низкого уровней и объединить их для повышения эффективности распознавания.

Авторы продемонстрировали эффективность использования трансферного обучения (Transfer Learning) для детектирования масок на основе набора данных WIDER FACE: метрика mAP показала увеличение до 2%.

RetinaFaceMask показала Accuracy для класса «Face» (отсутствия СИЗ на лице) 0.9373, для класса «Masked Face» (наличия СИЗ) – 0.9395. Однако принцип работы данной системы не обеспечивает обработки исследуемых изображений в режиме реального времени.

В работе [3] реализован многоэтапный метод детектирования СИЗ на лице с использованием детектора позы, определяющим ключевые точки тела, на основе которых извлекается область интереса анализируемого изображения.

Метод, описанный в работе [3], включает в себя пять этапов обработки (Рисунок 4):

1) распознавание одного или нескольких человек на изображении с помощью YOLOv4;

2) распознавание поз всех детектированных людей и определение 18 ключевых точек тела;

3) вычисление для лица ограничивающей прямоугольной рамки с опорой на координаты ключевых точек глаз и носа (детекции лиц размером меньше 20 × 20 пикселей удаляются);

4) классификация детектированных лиц с помощью модели трансферного обучения ResNet-101×1;

5) присвоение каждому человеку, присутствующему на изображениях, идентификатора с помощью алгоритма сопровождения объектов на видеоряде DeepSort для сбора статистики.

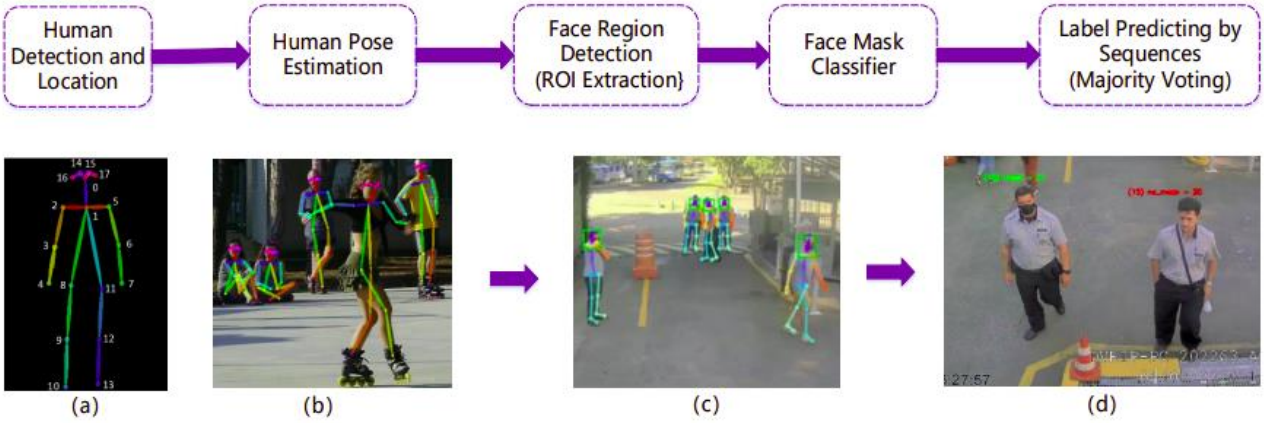

Рис. 4. Этапы работы системы детектирования СИЗ на лице с использованием детектора позы

Обученный на датасете из 1370 изображений высокого качества классификатор показал итоговый mAP = 0.8592 при IoU = 0.5. Однако на изображениях небольшого размера и низкого разрешения mAP составляет всего 0.403.

В работе приведена скорость работы только первого и пятого этапов системы (для подсчета точного количества людей на видеоряде) на NVIDIA GeForce GTX 1650 Max-Q – 15.7 FPS.

В работе [24] авторами предложено использовать для классификации архитектуру сверточных нейронных сетей InceptionV3, обученную на датасете Face Mask Detection Dataset Kaggle (FMD) [4], в котором ракурсы и освещение однотипны, а лица маленького размера отсутствуют. Набор был сокращен до 853 изображений с целью балансировки датасета. Accuracy предложенного подхода составила 0.8881.

В статье [25] для классификации используется YOLOv5, обученная на датасетах FMD и WIDER FACE, где на часть лиц была наложена маска программным способом. Accuracy составила 0.9611.

При применении методов распознавания СИЗ на кадрах видеоряда условия окружающей среды могут быть как контролируемые, так и неконтролируемые.

Контролируемые условия подразумевают работу обученного классификатора с изображениями высокого качества, когда все лица четко различимы и они не перекрываются. Датасеты, содержащие в себе кадры, сделанные в контролируемых условиях, используются, например, для разработки систем идентификации личности по лицу, когда объекты намеренно смотрят в объектив камеры. Примерами таких датасетов являются RMFD, SMFD и LFW, которые размещены в открытом доступе.

Условия видеонаблюдения являются неконтролируемыми из-за ракурса съемки, качества и размеров кадра видеоряда, из-за чего обучение моделей на наборах данных, собранных в контролируемых условиях, не способно обеспечить высокую точность классификации при работе с видеопотоком общественного наблюдения [9].

В таблице 1 приведены значения метрики Accuracy для рассмотренных подходов с указанием использованных датасетов и условиями их съемки.

Таблица 1

Значения Accuracy для существующих подходов

Подход	**Набор данных (условия съемки)**	**Accuracy**
ResNet-50, ансамбль методов	RMFD (контролируемые условия)	0.9949
	SMFD (контролируемые условия)	0.9964
	LFW (контролируемые условия)	1
SSDMNV2	FMD (неконтролируемые условия)	0.9264
Метод Виолы-Джонса, AdaBoost	Собранный самостоятельно (контролируемые условия)	0.9500
ResNet-50, YOLOv2	MMD, FMD (неконтролируемые условия)	0.8100

Продолжение таблицы 1

RetinaFaceMask	MAFA, FMD (неконтролируемые условия)	«Face» – 0.9373, «Masked Face» – 0.9395
YOLOv4, Pose Detector, ResNet-101×1, DeepSort	FMD (неконтролируемые условия)	- (mAP = 0.8592; для изображений небольшого размера и низкого расширения mAP = 0.4030)
InceptionV3	FMD (неконтролируемые условия)	0.8881
YOLOv5	FMD (неконтролируемые условия), WIDER FACE (контролируемые условия)	0.9611

Проведенный обзор показывает, что для обеспечения эффективности определения наличия СИЗ на лице необходим сбор собственного набора данных в рамках проведения контроля для обучения компонента классификации разрабатываемой системы.

Для мониторинга нарушений режима ношения СИЗ обосновано использование многоэтапных систем, включающих модель сопровождения объектов на видеоряде для присвоения одного идентификатора каждому человеку, что позволит обрабатывать информацию для действительного количества людей, присутствующих на видеоряде.

Также следует провести исследование возможности применения алгоритма детектирования позы для нахождения координат лица в сравнении с детекторами лица. Для улучшения точности работы моделей классификации возможно использование трансферного обучения и дообучения (fine-tuning).

ГЛАВА 2. ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К РАЗРАБАТЫВАЕМОЙ СИСТЕМЕ

Разрабатываемая автоматизированная система выявления лиц, не использующих СИЗ, в качестве входных данных должна использовать видеоряд, получаемый с камер видеонаблюдения, установленных на КПП институтов ФГАОУ ВО «Тюменский государственный университет».

Ниже сформулированы требования к разрабатываемой системе.

Требования к входным данным:

- форматы видеоряда:
  - MJPEG-видеопоток, частота обновления – 16 FPS, размерность изображения – 1280 × 720 пикселей (720p);
  - MP4-видеофайл, частота обновления – 25 FPS (допустима любая, корректная в рамках MP4-формата), размерность изображения – 1280 × 720 пикселей (720p);
- область интереса используемого видеоряда – непосредственно КПП (проход через турникеты), размерность области интереса – 400 × 400 пикселей.

Случайный кадр видеоряда, используемого в качестве входных данных разрабатываемой системы, представлен на рисунке 5. Синим квадратом на кадре выделена область интереса видеоряда для разработки системы – КПП.

Требования к выходным данным:

- изображение лица нарушителя режима ношения СИЗ на кадре: формат – JPEG, минимальный размер 128 × 128 пикселей;
- каждому факту нарушения должно соответствовать одно изображение лица нарушителя (отслеживание нарушителя);
- вероятность принадлежности к классу (в отладочном режиме – для возможных дальнейших доработок модели классификации);
- время обнаружения лица относительно времени получения входных данных.

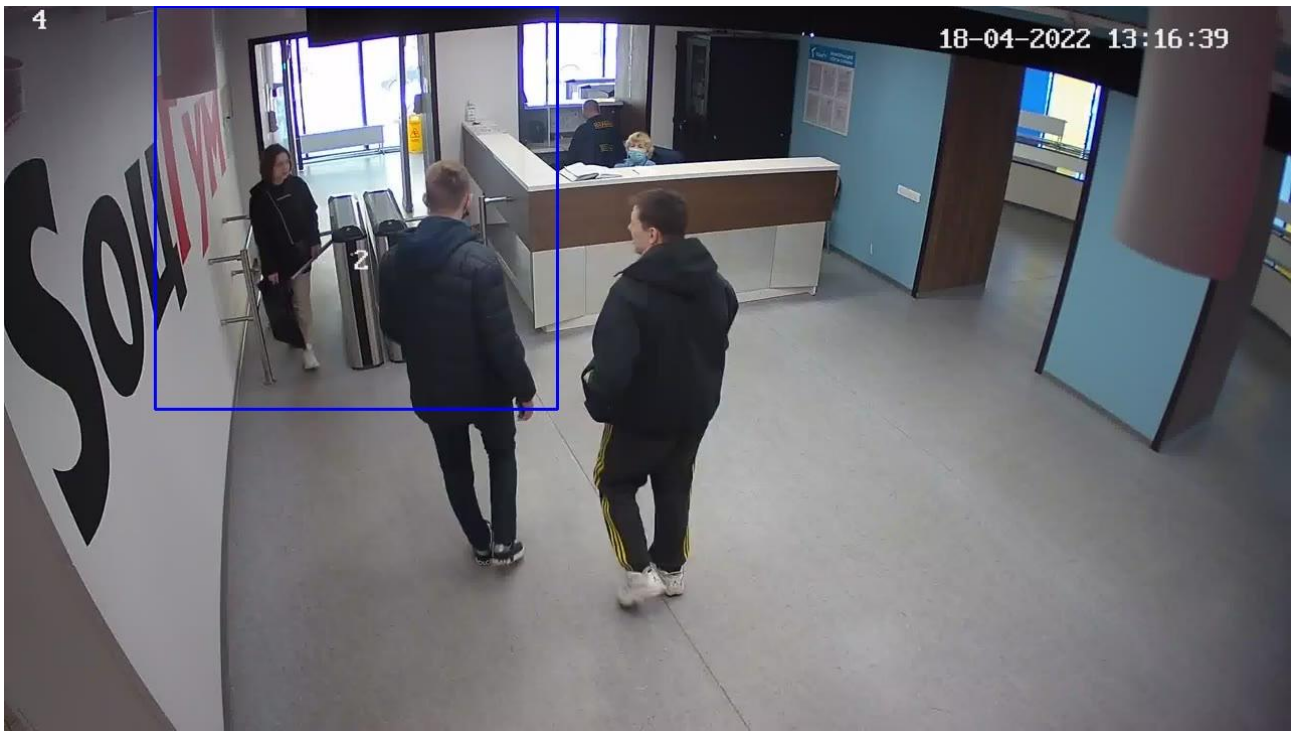

Рис. 5. Кадр видеоряда, получаемого с камеры видеонаблюдения

Требования к скорости работы, хранению и обновлению данных:

- скорость обработки получаемого изображения, приближенная к частоте получения кадров видеопотока (16 FPS) при работе на устройствах с GPU;
- уведомление о появлении нового нарушителя (нового изображения лица) при работе с видеопотоком;
- хранение данных о нарушениях не менее 48 часов.

Требования к платформе системы:

- компонент обработки видеоряда:
  - поддержка операционных систем семейства Linux, Windows 10;
  - работа на устройствах с GPU со скоростью не менее 16 FPS (для видеокарт поколения GTX 1060 и выше), возможная работа на устройствах без GPU (только с CPU);
  - процессор с частотой от 2 МГц, разрядность x64, оперативная память от 4 ГБ;
- компонент хранения и просмотра данных о нарушениях:
  - поддержка операционных систем Linux, Windows 10, Android 7+;

- процессор с частотой от 1.6 МГц, разрядность x64, оперативная память от 2 ГБ;
- минимальный объем свободного дискового пространства – 500 МБ.

Задача автоматизированного выявления лиц, не использующих СИЗ, включает в себя три подзадачи:

- обнаружение лица на кадрах видеоряда;
- сопровождение (трекинг) лица на каждом кадре видеоряда с объединением детекций по идентификатору;
- определение наличия или отсутствия СИЗ на лице на основе обработки кадров его трека.

В соответствии с требованиями и решаемыми задачами в рамках выпускной квалификационной работы определен конвейер обработки кадра в разрабатываемой системе со следующими компонентами (Рисунок 6):

- компонент получения данных: чтение следующего кадра, изменение размеров и компоновки субпикселей;
- компонент детектирования лица на изображении: выделение объектов на изображении, выделение лиц, соответствующих пороговым значениям, при помощи подавления не-максимумов;
- компонент сопровождения объектов: обновление результатов детектирования на основе границ лица, удаление из отслеживания объектов с необновленной детекцией;
- компонент хранения данных по лицам: хранение во внутреннем буфере данных по каждому идентификатору лица, при удалении объекта из отслеживания – выбор из буфера элемента с наибольшим размером изображения и наилучшей оценкой детекции лица;
- компонент определения наличия СИЗ: изменение размеров и компоновки субпикселей детектированного лица, отнесение изображения лица к классу наличия или отсутствия СИЗ;
- компонент хранения и просмотра данных о нарушителях: сохранение результатов для лиц с классом отсутствия СИЗ.

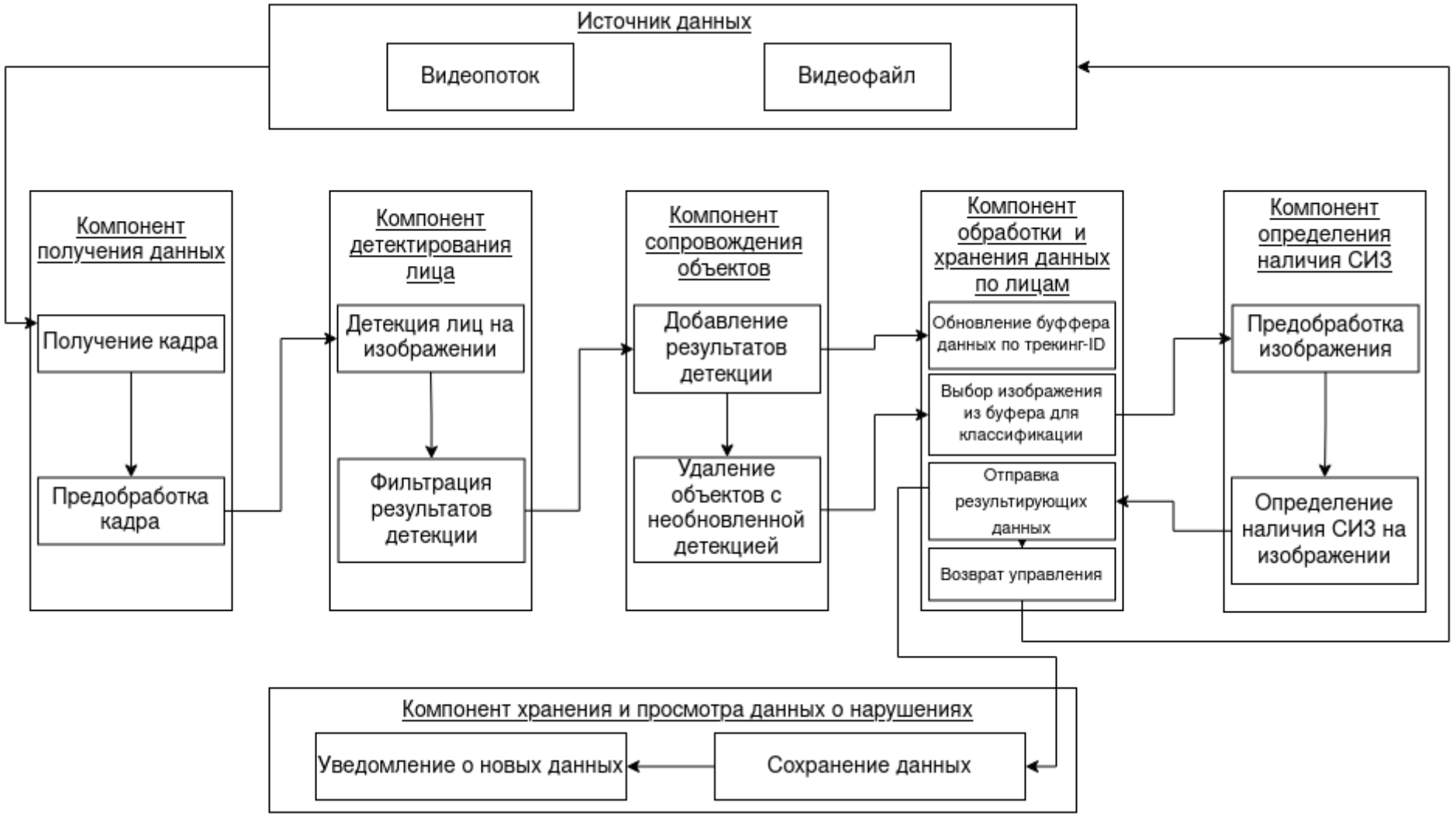

Рис. 6. Конвейер обработки кадра в автоматизированной системе выявления лиц, не использующих СИЗ

ГЛАВА 3. АНАЛИЗ ПОДХОДОВ К СОПРОВОЖДЕНИЮ ОБЪЕКТА НА ВИДЕОРЯДЕ

Сопровождение движущегося объекта (Object Tracking, «трекинг») – задача компьютерного зрения, заключающаяся в определении одного и того же объекта на каждом кадре видеоряда. Отдельной задачей является определение трекинга нескольких объектов на одном видеоряде (Multiple Object Tracking). Решения такой задачи часто используются для отслеживания передвижения человека. Например, в работе [22] предложен метод сопровождения на основе собственно разработанной динамической модели человека для отслеживания человека в толпе на улице при съемке с высоты птичьего полета.

Постановка задачи:

Пусть дана последовательность кадров видеоряда F_1, F_2, …, F_k, где k – номер изображения в последовательности.

Определим последовательность положений объекта на видеопотоке как D_s, D_{s+1}, …, D_{s+l-1}, где s – номер первого кадра, на котором был детектирован объект, а l – количество кадров с его наблюдением. Тогда решением задачи сопровождения объектов будет являться идентификация объекта на всех кадрах, где он присутствует, то есть присвоение каждому обнаруженному объекту индекса $i = 1, …, N$.

Таким образом, возможен переход от отдельных детекций объекта к его траектории, при использовании метрик схожести.

Преимущества использования отслеживания объекта на кадре в рамках автоматизированной системы выявления лиц, не использующих СИЗ, заключаются в следующем:

– классификаторы детектируемых объектов могут запускаться только один раз для каждого объекта, что позволит сократить время общей обработки видеопотока;

– возможность сбора статистики по точному количеству объектов;

– бóльшая устойчивость к ошибочным детекциям объекта (объекты с количеством детекций меньше порогового значения могут быть отброшены).

Существует два режима работы методов отслеживания объектов на видеоряде [10]:

- онлайн-отслеживание: подразумевает последовательную обработку кадров (использование только предыдущих кадров), что позволяет получать трекинг объекта в режиме реального времени;
- оффлайн-отслеживание: трекинг объекта является результатом обработки всех кадров видеоряда, при таком подходе необходимо использовать полностью записанный видеоролик с участием объекта интереса.

Для оперативного устранения нарушений с использованием видеонаблюдения необходимо использовать методы онлайн-отслеживания. Среди существующих алгоритмов лучшие значения точности и скорости работы в совокупности имеет SORT, сравнительный анализ по скорости работы и значению Accuracy представлен на рисунке 7. Скорость SORT составила 260 Гц (FPS), Multiple Object Tracking Accuracy (MOTA) – 33.4 при проведении вычислений на процессоре Intel i7 2.5GHz [17].

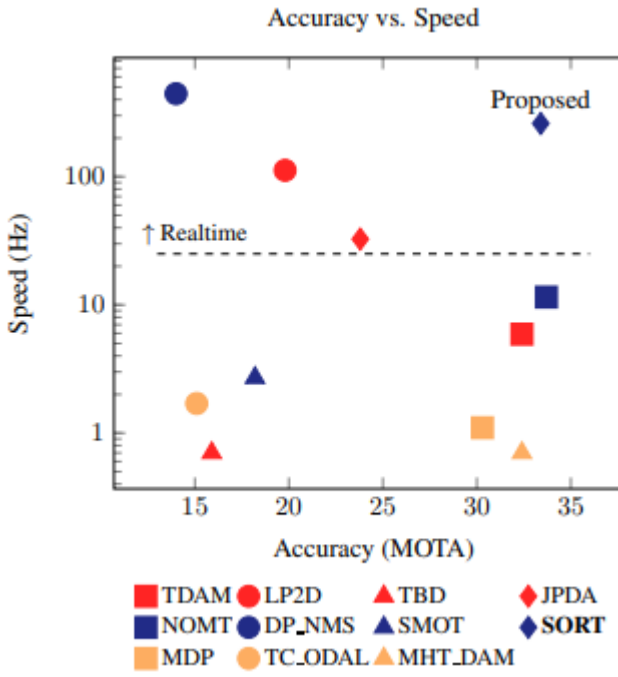

Рис. 7. Значения метрик MOTA и скорости для SORT (отмечен как «Proposed») и для других базовых алгоритмов сопровождения объектов

Преимущество в скорости по сравнению с другими методами достигнуто за счет того, что SORT отслеживает только окаймляющие объект

прямоугольники (bounding boxes), используя их положение и размер без других дополнительных признаков, не теряя в точности.

Метод сопровождения объектов на изображении SORT использует фильтр Калмана и венгерский алгоритм оптимизации. Последовательность работы SORT:

1) детектируются все объекты на изображении;
2) формируются треки объектов;
3) сопоставляются треки и детекции объектов с использованием венгерского алгоритма;
4) фильтр Калмана уточняет текущие позиции объектов;
5) при несовпадении формируются новые треки;
6) потерянные треки, удовлетворяющие правилам (превышение «времени жизни» трека), удаляются.

Венгерский алгоритм – метод оптимизации, устанавливающий соответствие между трекингом и новой детекцией (решение задачи о назначениях).

Венгерским методом вычисляется евклидовое расстояние между прогнозируемыми координатами i-ой детекции $(\bar{x}_{ti}, \bar{y}_{ti})$ и j-го сегмента трека (x_{sj}, y_{sj}) – центра области изображения, где извлекается признак движения:

$$E_{ij} = \sqrt{\left(\bar{x}_{tj} - x_{sj}\right)^2 + \left(\bar{y}_{tj} - y_{sj}\right)^2}. \quad (1)$$

Метрика евклидового расстояния в рамках решения задачи венгерским алгоритмом является стоимостью принятия решения о соответствии прогнозируемого сегмента трека детекции. Для установления соответствия или несоответствия необходимо решить матрицу стоимостей, где по столбцам матрицы распределены сегменты трека, а по строкам – детекции объектов.

Венгерский метод выполняет матричные операции над стоимостной матрицей так, чтобы появились нулевые элементы, говорящие о несоответствии детекции треку. На основании положительных значений элементов матрицы

делается вывод о соответствии, что может быть уточнено с помощью применения фильтра Калмана.

Фильтр Калмана – алгоритм обработки данных, позволяющий определять слишком большие (недостоверные) скачки в измерениях как ошибочные. При детектировании и сопровождении объектов используется как инструмент уточнения и прогнозирования координат искомых объектов.

Пример использования алгоритма SORT на видеоряде, получаемом с камер видеонаблюдения на КПП, представлен на рисунке 8. На нем изображено, как через КПП проходит три человека с детектированными лицами, ограниченными прямоугольными рамками, каждому из которых был присвоен идентификатор, который не менялся на протяжении всего их присутствия на кадрах видеоряда.

Рис. 8. Примеры сопровождения лиц с использованием алгоритма SORT

ГЛАВА 4. ВЫБОР ПОДХОДА К РЕШЕНИЮ ЗАДАЧИ ДЕТЕКТИРОВАНИЯ ЛИЦА НА ИЗОБРАЖЕНИИ

Входными данными для решения задачи бинарной классификации наличия СИЗ у человека является результат нахождения границ его лица на кадре.

Существует два подхода к вычислению границ лица в виде прямоугольной рамки, совместимых с сопровождением:

1. через обнаружение позы человека (Pose Detection), на основе координат ключевых точек тела;
2. как результат работы модели детектирования (Face Detection).

Низкая точность работы алгоритма детектирования лица приведет к некорректной работе последующих этапов обработки изображения в системе, а низкая скорость – к невозможности выполнения временных ограничений последующими этапами. При использовании алгоритма отслеживания требуется предоставить максимально возможные последовательные детекции объекта, что помимо точности обуславливается возможностью алгоритма определять объекты в различных положениях.

Исходя из этого, формируются следующие основные требования к выбору алгоритма детектирования лица:

- высокая точность детекции;
- низкая скорость работы, приближенная к частоте получения кадров источника данных;
- возможность детекции лица при различных углах его поворота.

4.1. ПОДХОД POSE DETECTION

Задачей методов детектирования позы является установление координат основных частей тела человека на изображении и построение векторной модели – формализованного представления тела человека, где векторы расположены вдоль костей скелета (Рисунок 9) [14].

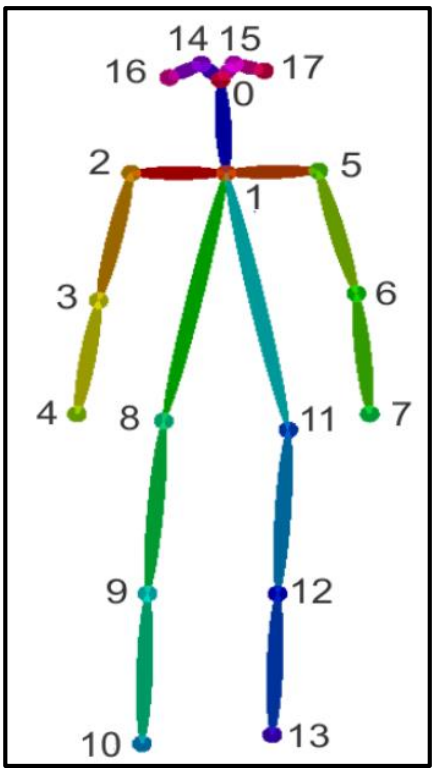

Рис. 9. Пример векторной модели человека с 18 ключевыми точками

Результатом работы метода определения позы являются координаты ключевых точек тела человека, соотнесенные с суставами, с оценкой достоверности для каждой точки. Ключевые точки играют роль границ векторов скелета человека, построенных алгоритмом детектирования позы.

Отдельной задачей является вычисление с помощью детектора позы ключевых точек тел нескольких человек, присутствующих на изображении одновременно (multi-person).

OpenPose – multi-person библиотека для определения позы человека на двумерных изображениях с открытым исходным кодом, являющаяся одним из наиболее применяемых при разработке специализированных систем.

Последовательность работы OpenPose [13]:

1) цветное изображение проходит через CNN (первые 10 слоев fine-tuned VGG-16);

2) извлекаются PFM (Part Confidence Maps) для детектирования частей тела и PAF (Part Affinity Fields) – непараметрические представления для определения принадлежности обнаруженных частей тела к конкретным людям на кадре;

3) при использовании PFM и PAF вычисляются паросочетания для признаков – таким образом формируются «суставы» векторной модели позы;

4) паросочетания формируют полную векторную модель позы человека.

HyperPose – конфигурация OpenPose, представляющая собой оптимизированную под выполнение в реальном времени модель, способную распределять задачи оценки позы между GPU и CPU, что обеспечивает высокую степень утилизации аппаратных ресурсов. HyperPose имеет более высокую скорость по сравнению с OpenPose без потери точности [6].

Чтобы перейти к определению ключевых точек тела, требуется постобработка полученных посредством HyperPose данных, а именно проведение вычислений, где основное время занимают операции GaussianBlur и MaxPooling.

Для лица задано 5 ключевых точек: левое и правое ухо, левый и правый глаз, нос. Оценка видимости лица может производиться при помощи оценок достоверностей для координат носа и глаз: если значения оценок не превышают некоторый порог, то лицо не видно.

Для нахождения рамок вокруг лица требуется использовать относительные координаты: расстояния между ключевыми точками. Использование точек лица либо точек ушей в данном случае ухудшит устойчивость определения, так как при нахождении человека боком к камере расстояние между этими точками будет стремиться к нулю. Для определения ограничивающих рамок лица наиболее универсальным будет использование ключевых точек бедер и шеи:

– для определения размера рамок используется расстояние от точки шеи до центра отрезка между точками бедер с множащим коэффициентом;

– для определения положения рамок относительно лица следует ориентироваться на ключевую точку шеи в качестве опорной.

Данный принцип построения прямоугольных рамок лица продемонстрирован на рисунке 10, где:

- синим цветом выделена общая векторная модель позы,

- желтым цветом показаны ключевые точки шеи и бедер и расстояния, используемые для расчета размера и положения прямоугольных рамок вокруг обнаруживаемого лица,
- черным цветом показаны найденные ограничивающие рамки для лица с определенным идентификатором объекта.

Рис. 10. Ограничивающие прямоугольные рамки лица, определенные с использованием HyperPose (tiny VGG)

4.2. ПОДХОД FACE DETECTION

Задачей методов детектирования лица является установление координат прямоугольных границ лица относительно исходного изображения. Как правило, для обнаружения и локализации лица используются архитектуры нейронных сетей, способные быстро детектировать несколько лиц на изображении при различных условиях освещенности, повороте и наклоне детектируемого лица.

В работе [21] авторы провели сравнительный анализ существующих детекторов лиц на датасете WIDER FACE, содержащем 32203 изображения с 393703 размеченными лицами.

Изображения были поделены на три выборки по уровням сложности детектирования лица: легкий, средний и сложный. Данные уровни различаются условиями окружающей среды, количеством людей на изображениях,

положениями головы. Точность по «сложной» выборке наилучшим образом отражает эффективность работы детекторов.

Результаты измерений точности представлены в таблице 2.

Таблица 2

Сравнительный анализ детекторов лица [21]

Детектор лица	Модель	Точность на «легкой» выборке, %	Точность на «средней» выборке, %	Точность на «сложной» выборке, %	Число параметров, млн	FLOPs (GPU)
DSFD	ResNet-152	94.29	91.47	71.39	120.06	259.55
RetinaFace	ResNet-50	94.92	91.90	64.17	29.50	37.59
HAMBox	ResNet-50	95.27	93.76	76.75	30.24	43.28
TinaFace	ResNet-50	95.61	94.25	81.43	37.98	172.95
SCRFD-34GF (нет в открытом доступе)	Bottleneck ResNet	96.06	94.92	85.29	9.80	34.13
SCRFD-10GF (нет в открытом доступе)	Basic ResNet	95.16	93.87	83.05	3.86	9.98
YOLOv5s	**YOLOv5-CSPNet**	**94.33**	**92.61**	**83.15**	**7.075**	**5.751**

FLOPs («floating point operations») описывает количество операций, требуемое для разового прохода модели. Чем меньше FLOPs, тем выше скорость работы модели.

В результате сравнения YOLOv5s показала наилучшую точность среди моделей в открытом доступе и наилучшую скорость среди всех моделей. В работе [21] показана устойчивость YOLOv5s к различным положениям лица, что соответствует требованиям к выбору алгоритма детектирования лица, приведенным в Разделе 3.1.

YOLO – алгоритм, основанный на сверточных нейронных сетях и использующийся для обнаружения нескольких объектов на изображении. YOLO обрабатывает все изображение единожды, создавая ограничивающие прямоугольные рамки для всех возможных обнаружений искомого объекта,

впоследствии только уточняя их положение, что способствует увеличению скорости распознавания объектов [23].

Выходными данными YOLO являются прямоугольная рамка вокруг искомого объекта и вероятность принадлежности к классу. Метрика IoU позволяет избежать многократного детектирования одного и того же объекта.

4.3. СРАВНИТЕЛЬНЫЙ АНАЛИЗ

Были проведены замеры скоростей работы моделей HyperPose и YOLOv5s на CPU Intel(R) Xeon(R) @ 2.20GHz и GPU Tesla K80.

Размеры изображений:

- 346 × 342 пикселя для HyperPose,
- 320 × 320 пикселей для YOLOv5s.

Результаты замеров скорости и сравнение моделей по их основным характеристикам приведены в таблицах 3, 4. Полученные расчеты являются средними значениями за 100 операций.

Таблица 3

Сравнительный анализ скорости работы моделей HyperPose и YOLOv5s

Тип замера скорости работы	HyperPose CPU скорость	HyperPose GPU скорость	YOLOv5s CPU скорость	YOLOv5s GPU скорость
Один проход модели	1383.48 мс 0.72 FPS	11.99 мс 83.4 FPS	103.06 мс 9.7 FPS	29.03 мс 34.45 FPS
Постобработка	218.58 мс 4.58 FPS	125.82 мс 7.95 FPS	Не требуется	Не требуется
Полная обработка изображения	1627.44 мс 0.61 FPS	156.18 мс 6.4 FPS	**108.81 мс** **9.19 FPS**	**34.31 мс** **29.12 FPS**

Таблица 4

Сравнительный анализ HyperPose и YOLOv5s по основным характеристикам

Характеристика	HyperPose	YOLOv5s
Вычисление координат лица	На основе ключевых точек тела человека	Как результат работы модели

Продолжение таблицы 4

Устойчивость детектирования при положении лица не анфас	Возможность детектирования полностью отвернутого от камеры лица. Чтобы определить отвернуто ли лицо, можно использовать оценку точности детектирования точек лица	Возможность детектирования лица при повороте в профиль. Чтобы определить повернуто ли лицо в профиль, можно использовать оценку точности детектирования лица и соотношение высоты и ширины его прямоугольных границ
Устойчивость детектирования лица при его близости к объективу камеры	Невозможность вычисления границ лица при отсутствии детектирования ключевых точек тела, не относящихся к лицу	Устойчивое детектирование лица при его попадании в кадр
Скорость работы на GPU	6.4 FPS	29.12 FPS

Исходя из проведенного анализа, можно сказать, что наилучшим подходом для детектирования лица, совместимым с алгоритмом сопровождения, является YOLOv5s. При проведении замеров модель продемонстрировала более высокую скорость работы на GPU – 29.12 FPS, при этом показав устойчивость детектирования лица при его повороте и приближении к камере наблюдения, когда тело не видно на кадре.

ГЛАВА 5. АНАЛИЗ ПОДХОДОВ К БИНАРНОЙ КЛАССИФИКАЦИИ НАЛИЧИЯ СРЕДСТВ ИНДИВИДУАЛЬНОЙ ЗАЩИТЫ НА КАДРЕ ЛИЦА

5.1. ДАННЫЕ ДЛЯ ОБУЧЕНИЯ

Задача определения наличия СИЗ на изображении лица является частным случаем задачи бинарной классификации с классами наличия или отсутствия СИЗ на лице.

Для обучения классификатора потребуется подготовить датасет с размеченными по соответствующим классам изображениями. В открытом доступе опубликовано несколько датасетов, которые можно использовать для решения подобных задач, например, наборы с «изображениями в реальных условиях» RMFD (Real-World-Masked-Face-Dataset) [9] и Face Mask Detection Kaggle [4], примеры изображений представлены на рисунке 11.

Рис. 11. Примеры изображений в наборах данных, находящихся в открытом доступе

При неконтролируемых условиях видеонаблюдения обучение на датасетах «с изображениями в реальных условиях» не способно обеспечить высокую

точность классификации из-за несоответствующих входным данным разрабатываемой системы ракурса съемки, качества и размера кадров видеоряда.

Требуется подготовить набор изображений лиц на основе входных данных разрабатываемой системы для возможности обучения классификатора и проведения оценки.

Санитарно-гигиеническая маска представляет собой объект, который находится на лице человека, цвет и рисунок на поверхности могут быть любыми. При разметке изображений отнесение к классу наличия маски определяется следующими критериями:

- прикрыты нос и рот человека – маска надета,
- прикрыт только рот – маска надета,
- не прикрыты рот и нос – маска не надета,
- прикрыт только нос – маска не надета.

В соответствии с приведенными правилами изображения будут размечены по классам «mask» и «no_mask».

Для подготовки набора данных был обработан видеопоток с камеры наблюдения на КПП Социально-гуманитарного института Тюменского государственного университета общей длительностью 16:03:03. Количество обработанных кадров: 1445250. Реальное обработанное время: 29:17:30. Минимальный размер изображения в датасете – 35 × 40 пикселей, максимальный – 121 × 144 пикселя.

Итоговый размер датасета:

- всего: 5636 изображений;
- «mask»: 2738;
- «no_mask»: 2898.

Распределение объектов подготовленного датасета по классам приведено на рисунке 12.

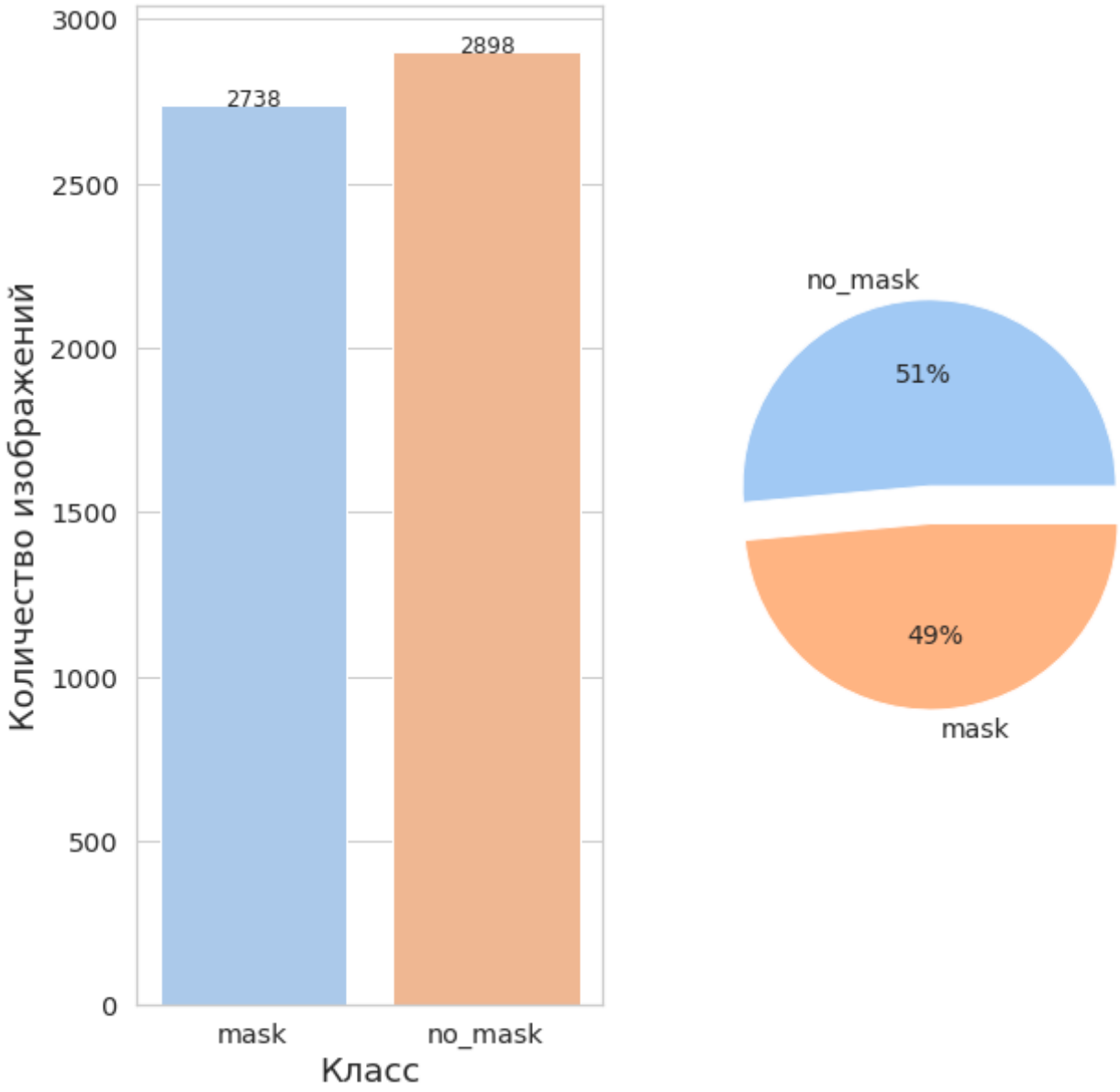

Рис. 12. Распределение 5636 объектов подготовленного датасета по классам

Разделение датасета на обучающую и тестовую выборки проводилось случайным образом, при этом доля обучающей выборки от исходных данных составила 90%, доля тестовой – 10% (Рисунок 13):

- всего: 5636 изображений;
- обучающая выбора: 5072;
- тестовая выборка: 564.

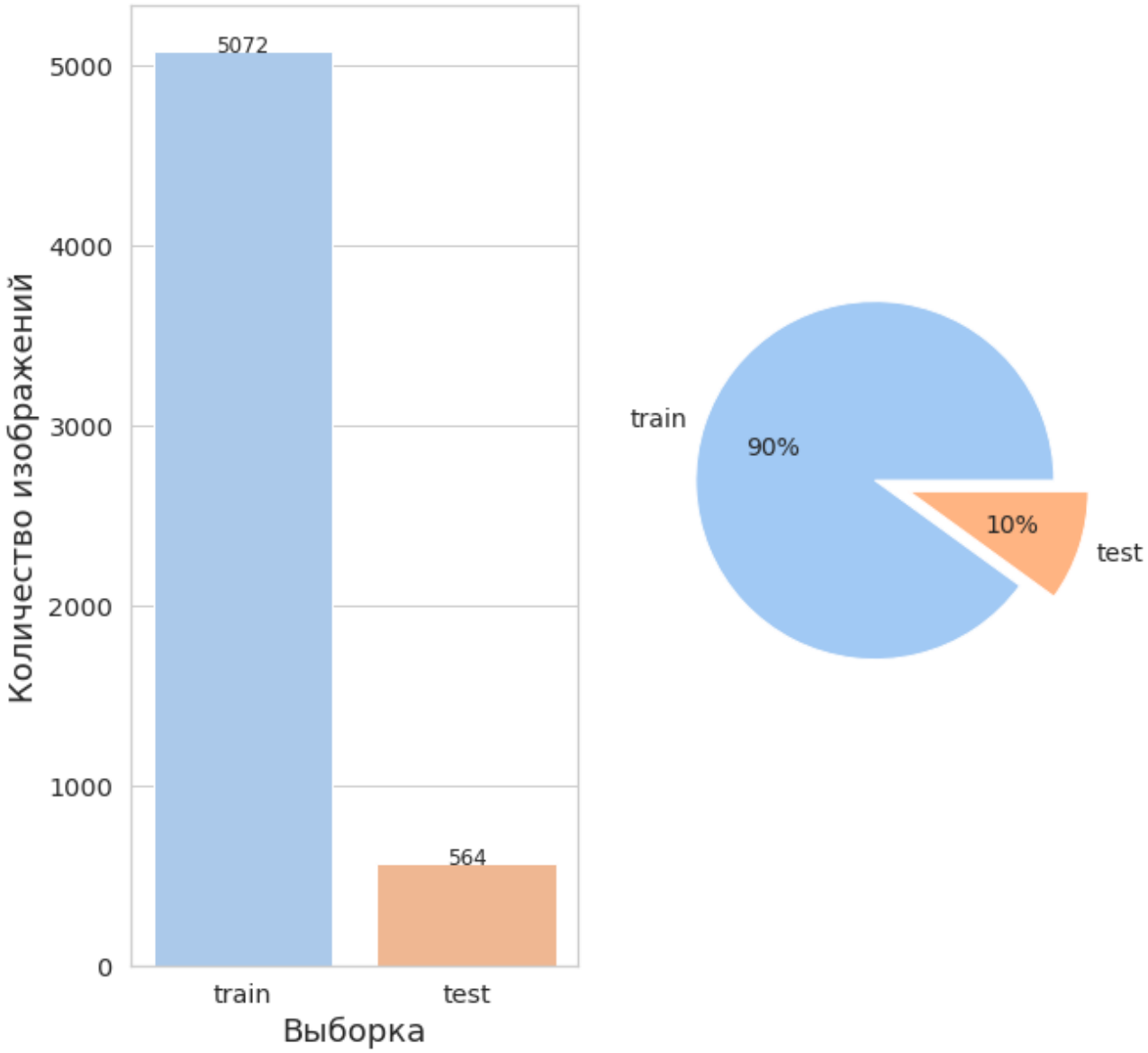

Рис. 13. Распределение 5636 объектов датасета по обучающей и тестовой выборкам

Подготовленный датасет включает в себя кадры лиц как мужчин, так и женщин от 18 лет, с разными аксессуарами, прическами, растительностью на лице, выражениями эмоций.

На изображениях зафиксированы различные степени поворота лица относительно камеры как горизонтально, так и вертикально: до поворота в профиль и до наклона головы вниз.

Учтена разная степень надевания маски: от прикрытия только подбородка до правильного ношения (покрыт нос и рот), изображения распределены в соответствие сформулированным правилам ранее в данной главе.

На рисунке 14 представлены примеры размеченных изображений итогового набора данных.

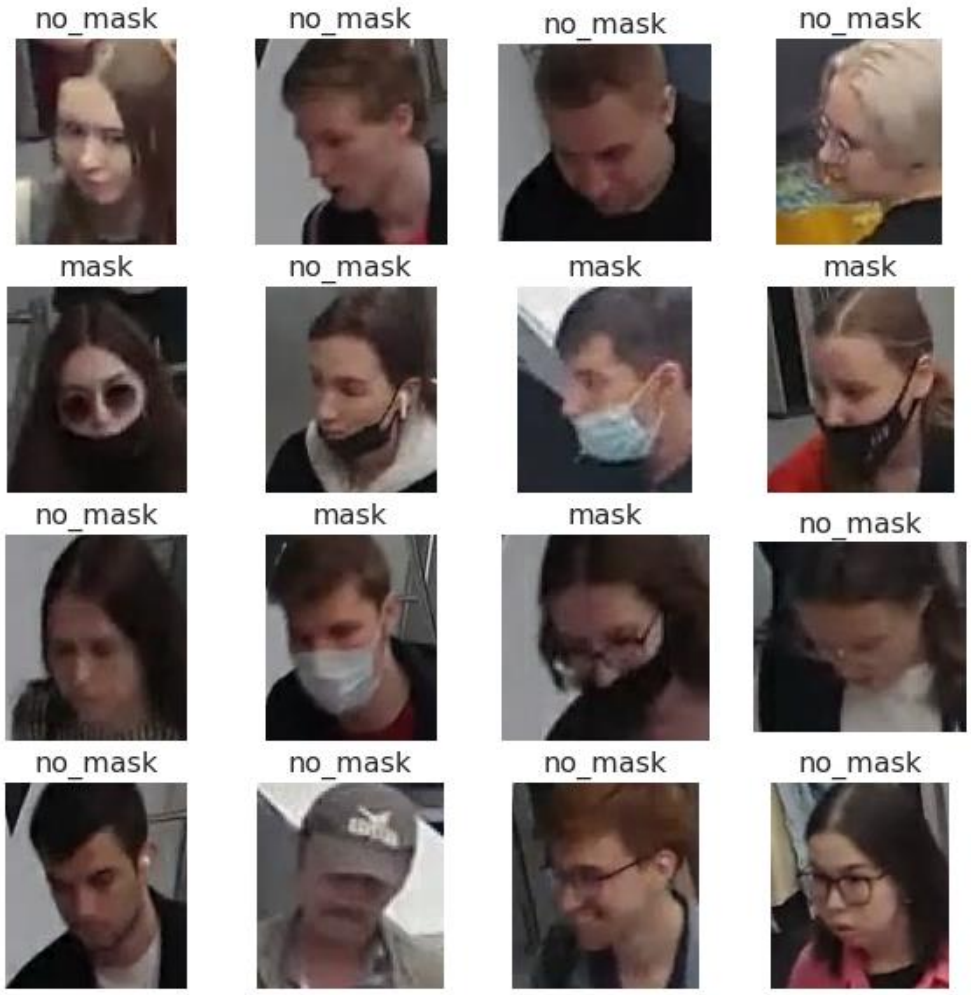

Рис. 14. Примеры изображений подготовленного датасета

5.2. СРАВНИТЕЛЬНЫЙ ЭКСПЕРИМЕНТ МОДЕЛЕЙ КЛАССИФИКАЦИИ

При решении задачи классификации изображений существует два подхода к обучению моделей: обучение с нуля и перенос обучения (Transfer Learning, «трансферное обучение») ранее предобученных моделей, используемых как есть или изменяемых под определенную задачу.

В основе трансферного обучения для классификации изображений лежит идея, что модель, обученная на большом и достаточно общем наборе данных, будет эффективно служить общей моделью визуального мира. Существуют следующие методы переноса обучения:

- Извлечение признаков (feature extraction) – использование результата предыдущего обучения модели для извлечения признаков объектов решаемой задачи. Слои используемой модели замораживаются, добавляются верхние слои, сконфигурированные под решаемую задачу (с необходимым числом классов) и происходит обучение только добавленных слоев.
- Дообучение (fine-tuning) – обучение слоев базовой модели вместе с добавленными слоями, что позволяет больше подстроить модель под решаемую задачу. Может использоваться как второй корректирующий этап после этапа извлечения признаков.

Как показано авторами в работе [15], решающими схожую задачу с набором изображений в несколько тысяч объектов, обучение модели с нуля приводит к переобучению (увеличение точности на обучающей выборке приводит к падению точности на тестирующей выборке) и максимальная тестирующая Accuracy не превышает 0.75, в то время как при использовании переноса обучения отсутствует проблема переобучения, и максимальная тестирующая Accuracy достигает 0.96.

При переносе обучения можно использовать модели извлечения признаков, обученные на наборе данных ImageNet с ручной аннотацией изображений, который зачастую служит основой для предварительной подготовки моделей, решающих задачи классификации изображений. ImageNet1k с 1281167 изображениями, размеченными по 1000 классам, является подмножеством полного датасета с 14197122 изображениями, размеченными по 21841 классам – ImageNet21k. Модели, предобученные на полном ImageNet21k, способны показывать лучшие результаты при переносе обучения из-за бóльшего числа изображений, обрабатываемых при обучении [8].

ImageNet содержит изображения, отнесенные к классу «mask», что делает модели извлечения признаков, обученные на этом наборе данных, подходящими для решения задачи определения наличия СИЗ на кадре лица.

Для проведения сравнительного эксперимента по переносу обучения на подготовленном наборе данных были выбраны наиболее производительные

архитектуры семейства EfficientNetV2 [19], BiT-M [2] и наиболее полная архитектура семейства MobileNetV3 [16]. Сравнительный анализ характеристик используемых моделей приведен в таблице 5.

Таблица 5

Сравнительный анализ основных характеристик моделей BiT-M-ResNet-50×1, EfficientNetV20-B0, MobileNetV3-Large-1.0

Набор данных	Семейство	Модель	Размерность входного слоя (пиксели)	Число параметров (млн)	Скорость работы (мс)	Top-1 Accuracy
ImageNet 21k	BiT-M	ResNet-50×1	128×128	25.6	9.1	0.8
ImageNet 21k	EfficientNetV2	EffNetV2-B0	224×224	7.2	7	0.787
ImageNet 1k	MobileNet V3	MobileNetV3-Large-1.0	224×224	5.48	5.5	0.752

Размерность 128 × 128 пикселей для BiT-M модели использовалась как рекомендуемая разработчиками архитектуры по BiT-HyperRule, размерность 224 × 224 пикселей взята как наименьшая доступная для архитектур EffNetV2-B0 и MobileNetV3-Large-1.0. Замеры скорости приведены для работы моделей на GPU Tesla T4.

При проведении трансферного обучения последний Dense-слой моделей был заменен на Dense-слой с размерностью 2 по числу классов решаемой задачи с заморозкой обучения исходной модели. В качестве оптимизатора использовался Adam, показывающий наряду с NAdam наименьшие итоговые ошибки в сравнении с оптимизаторами SGD, Momentum, Nesterov и RMSProp при обучении на ImageNet-датасете, как продемонстрировано авторами в [12].

Обучение каждой модели проводилось в два этапа:

- заморозка слоев исходной модели с обучением только последних добавленных слоев:
  - количество шагов – 2000: 200 эпох по 10 шагов в эпохе;
  - batch size: 256 изображений;

- оптимизатор – Adam с learning rate – 10^{-4};
- функция потерь – SparseCategoricalCrossentropy;

- разморозка слоев исходной модели с полным обучением модели (fine-tuning):
  - количество шагов – 2000: 200 эпох по 10 шагов в эпохе;
  - batch size: 128 изображений;
  - оптимизатор – Adam с learning rate – 10^{-6};
  - функция потерь – SparseCategoricalCrossentropy.

Для повышения устойчивости модели к освещению и увеличения разнообразия обучающей выборки к изображениям применялись следующие шаги предобработки:

- случайное отражение по горизонтали (random_flip_left_right);
- случайное вращение (rotate с углом от –20 до 20);
- случайное изменение яркости (random_brightness с max_delta=0.2);
- случайное изменение контрастности (random_contrast с lower=0.8 и upper=1.2);
- случайное изменение насыщенности (random_saturation с lower=0.8 и upper=1.2).

Примеры результатов предобработки изображений обучающей выборки приведены на рисунке 15.

Рис. 15. Примеры случайных результатов предобработки одного изображения

Размерность обучающей выборки при помощи аугментации доводилась до соответствия количеству шагов обучения: до 512000 изображений для первого этапа обучения и до 256000 изображений для второго этапа.

Графики метрик моделей по ходу обучения приведены на рисунках 16, 17. На первом этапе обучения моделей происходит последовательное улучшение метрик, замедляющих изменения после 100 эпох, переобучения не наблюдается. Обучение на втором этапе приводит к улучшению значений метрик по каждой модели, однако после 100 эпох изменения также замедляются.

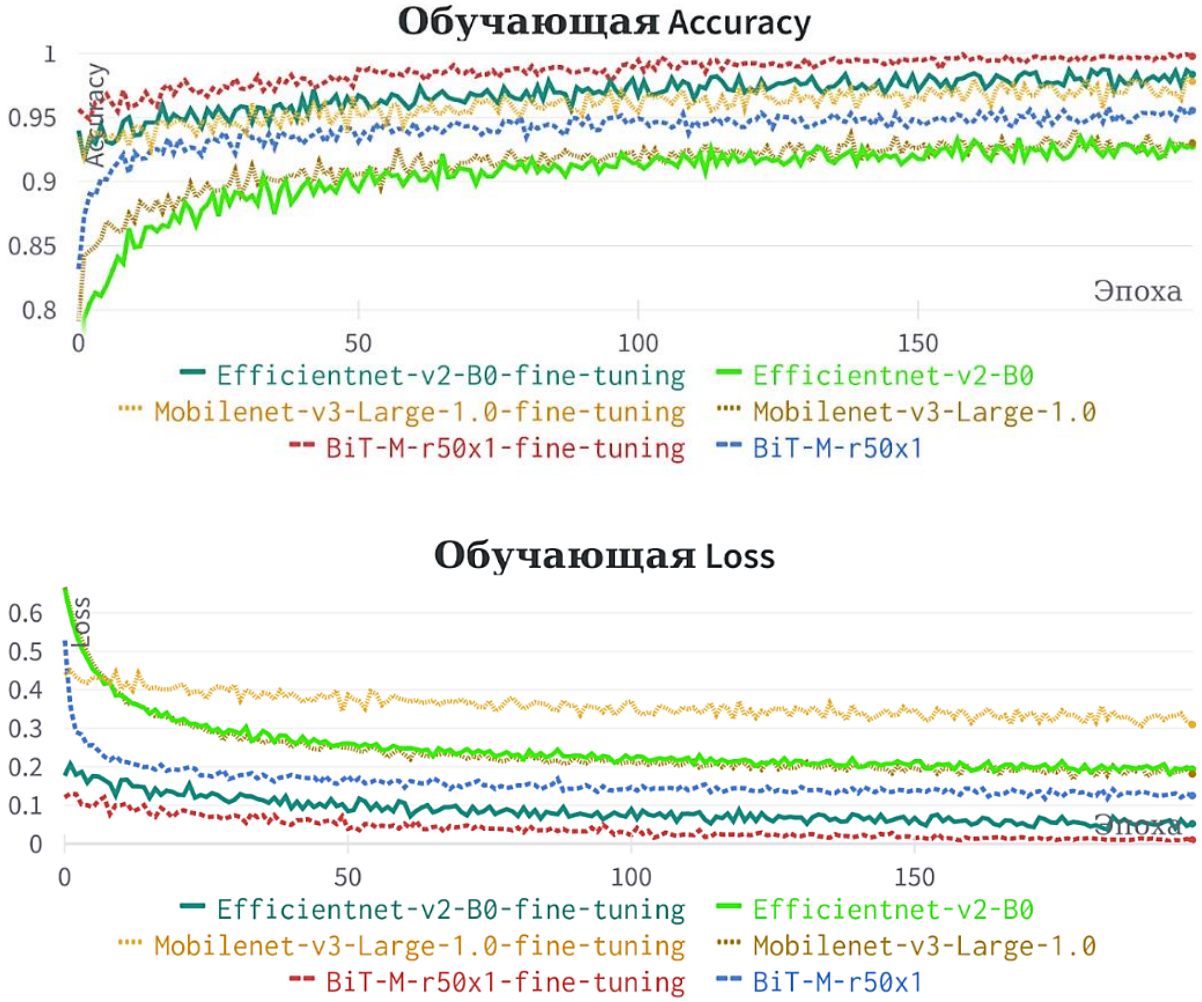

Рис. 16. Метрики Accuracy и Loss моделей EfficientNet-v2-B0, MobileNetV3-Large-1.0 и BiT-M-ResNet-50×1 с fine-tuning и без по ходу обучения на обучающей выборке

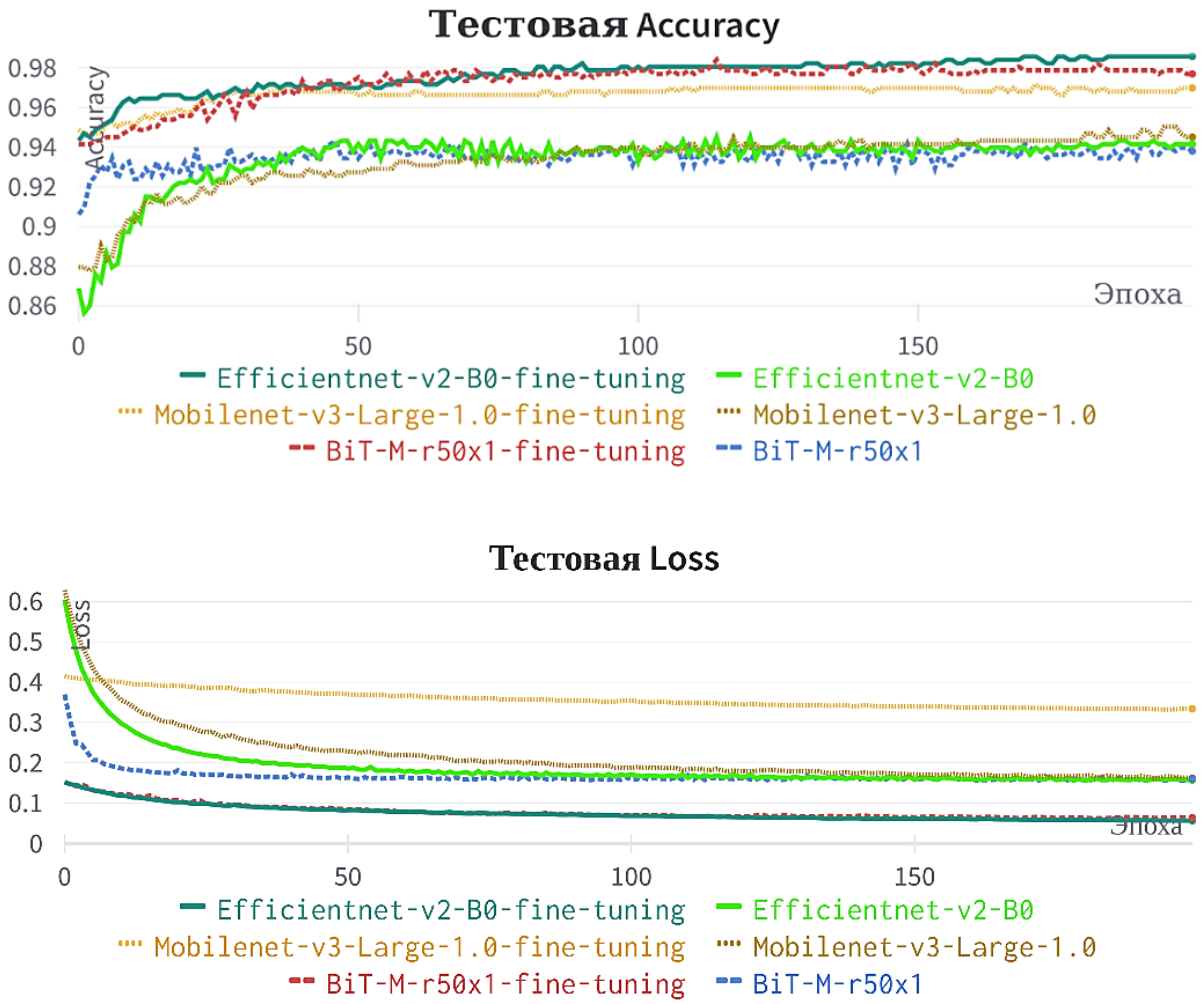

Рис. 17. Метрики Accuracy и Loss моделей EfficientNet-v2-B0, MobileNetV3-Large-1.0 и BiT-M-ResNet-50×1 с fine-tuning и без по ходу обучения на тестовой выборке

В таблице 6 представлены итоговые метрики оценки качества классификации для моделей после fine-tuning этапа. Модель EfficientNet-v2-B0 показала наивысшие результаты по Accuracy, Precision и F1-мере, уступив другим моделям лишь по Recall-показателю, что наряду с хорошей скоростью работы делает модель наилучшим выбором.

Таблица 6

Сравнительный анализ на основе метрик оценки качества классификации fine-tuning моделей BiT-M-ResNet-50×1, EfficientNetV20-B0, MobileNetV3-Large-1.0

Метрика\Модель	BiT-M R50×1	**EfficientNet-v2-B0**	MobileNetV3-Large-1.0
Accuracy	0.9770	**0.9858**	0.9699
Recall	0.9832	0.9789	**0.9868**
Precision	0.9733	**0.9929**	0.9585
F1-мера	0.9782	**0.9858**	0.9724

На рисунке 18 приведена матрица ошибок классификатора EfficientNet-v2-B0.

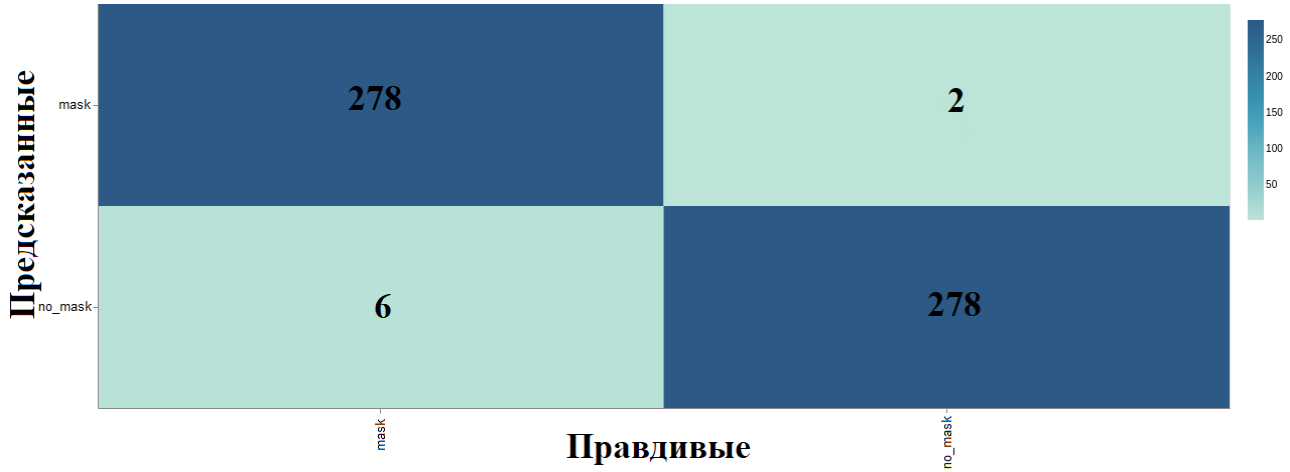

Рис. 18. Матрица ошибок EfficientNet-v2-B0

Всего изображений в тестовой выборке – 564, для них False Negative («ложноотрицательный») результат классификации составил 6 изображений, что определяет показатель Recall (0.9789 для EfficientNet-v2-B0). Для решения задачи выявления нарушителей, не использующих СИЗ, такая доля ошибок FN существенно не сказывается на эффективности работы системы, так как в случае отнесения человека с СИЗ в класс «no_mask» последует ручная перепроверка изображений, полученных ответственными сотрудниками.

False Positive («ложнопозитивный») результат классификаций минимален – он составил всего 2 изображения, что незначительно по сравнению с

определенными нарушениями (278 изображений). Количество FP ошибок классификации влияет на величину метрики Precision (0.9929 для EfficientNet-v2-B0).

На рисунке 19 приведены примеры ошибочных False Negative детекций модели EfficientNet-v2-B0, в которых объекты с маской были определены к классу отсутствия маски «no_mask».

Рис. 19. Примеры ошибок в классификации изображений тестовой выборки

На данных изображениях цвет надетой маски близок к цвету кожи, и для улучшения правильности классификации подобных случаев потребуется добавление в обучающую выборку изображений с надетыми на лица масками таких цветов с дальнейшим дообучением модели.

ГЛАВА 6. СИСТЕМА ВЫЯВЛЕНИЯ ЛИЦ, НЕ ИСПОЛЬЗУЮЩИХ СРЕДСТВА ИНДИВИДУАЛЬНОЙ ЗАЩИТЫ

6.1. РАЗРАБОТКА СИСТЕМЫ

Разработка автоматизированной системы распознавания лиц, не использующих СИЗ, на основе анализа видеоряда проводилась в соответствии с требованиями, приведенными в Главе 2.

На рисунке 20 изображена архитектура разрабатываемой системы, представленная следующими компонентами:

– Видеопоток – источник входных данных системы.

– Python-backend – обработчик получаемого от источника данных кадра, формирующий классифицированные изображения лиц.

– Telegram – платформа хранения и просмотра данных, получаемых от обработчика кадра.

Рис. 20. Архитектура системы выявления лиц, не использующих СИЗ

В качестве платформы представления результатов работы системы был выбран кроссплатформенный мессенджер Telegram, отвечающий требованиям к хранению данных, уведомлениям о новых данных и поддерживаемых платформах. При работе системы с видеопотоком время обнаружения лица будет соответствовать времени сообщения в Telegram.

По результатам проведенных исследований для реализации компонентов системы были выбраны модели:

- YOLOv5s – в качестве модели детектирования лица;
- SORT – в качестве алгоритма отслеживания лица;

- EfficientNet-v2-B0 – в качестве модели классификации изображений лиц по классам наличия и отсутствия СИЗ («mask» и «no_mask»).

Система разработана на языке Python 3.8.10 с использованием технологий:

- Telegram API 6.0;
- FilterPy 1.4.5;
- NumPy 1.21.4;
- OpenCV 4.5.4;
- Tensorflow 2.8.0;
- Torch 1.11.0;
- Requests 2.28.0.

Было разработано 3 класса и 16 методов. Для автоматизации процесса подготовки набора данных, приведенного в Разделе 5.1, использовалась модификация системы с сохранением детектированных изображений без классификации на дисковом пространстве.

На рисунке 21 представлена диаграмма классов в формате UML, на которой приведены разработанные классы, а также внешние используемые классы: Sort – класс, реализующий алгоритм отслеживания объектов, hub.KerasLayer – класс для работы с моделью классификатора, torch – класс для работы с моделью детектора лиц.

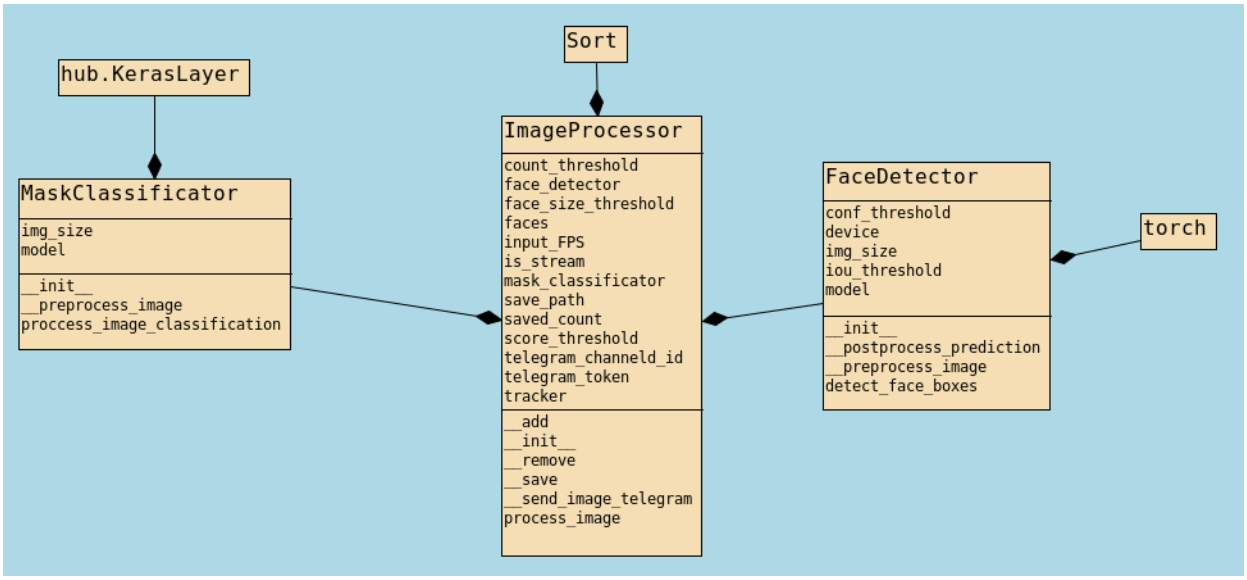

Рис. 21. UML-диаграмма классов системы выявления лиц, не использующих СИЗ, на основе анализа видеоряда

Код класса ImageProcessor приведен в Приложении 1, код класса FaceDetector приведен в Приложении 2. В таблице 7 приведено описание основных методов.

Таблица 7

Описание основных методов системы выявления лиц, не использующих СИЗ

Класс	Метод	Описание
FaceDetector	detect_face_boxes()	Метод определения границ лиц по переданному изображению с помощью модели детектирования, использующий методы предобработки и постобработки
MaskClassificator	proccess_image_classification()	Метод определения вероятности принадлежности к классам наличия и отсутствия СИЗ переданного изображения лица с помощью модели классификации, использующий метод предобработки
ImageProcessor	process_image()	Метод обработки поступающего кадра, в котором обновляется информация по детекциям для алгоритма отслеживания, кадр сохраняется в буфер и для удаляемых из отслеживания объектов вызывается метод с выбором из буфера, классификацией и отправкой результатов в Telegram

Последовательность выполнения метода process_image() класса ImageProcessor (блок-схема приведена на рисунке 22):

1. Определение координат лиц на изображении.
2. Для каждого детектированного лица проверка соответствия пороговым значениям и добавление в массив корректно детектированных лиц.
3. Обновление треков отслеживания на основе созданного массива.
4. Получение действующих и удаляемых треков.

5. Для каждого действующего трека преобразование координат до соответствия границам изображения и обновление внутреннего буфера для идентификатора трека.

6. Для каждого удаляемого трека вызов метода проведения классификации, формирования и отправки результатов.

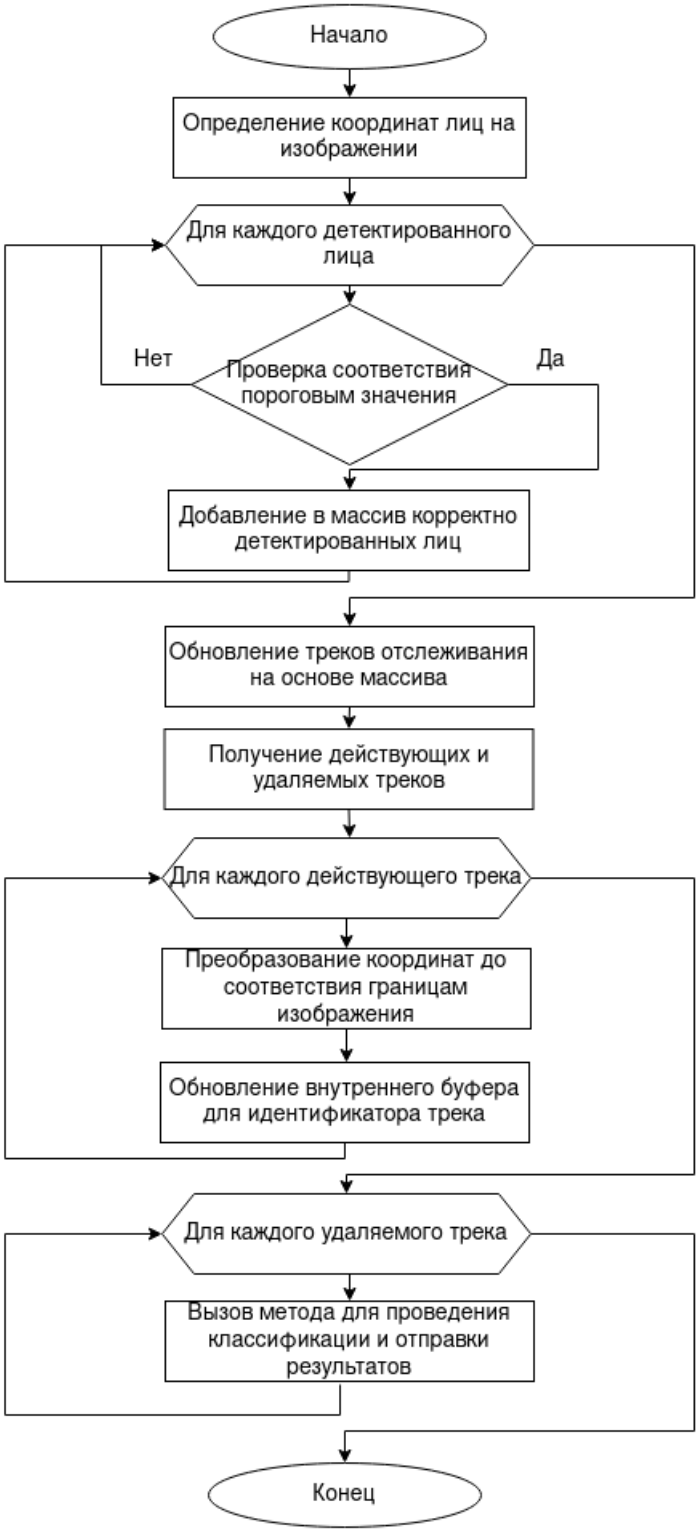

Рис. 22. Блок-схема метода process_image() класса ImageProcessor

Реализованы два режима работы системы:

– основной: с отправкой изображений лиц только с классом отсутствия СИЗ;

– отладочный: с отправкой результатов каждой детекции – изображения лица и вероятности принадлежности к каждому классу.

Режим отладки может использоваться для тестирования системы и сбора данных для дальнейших доработок.

Отправка изображений происходит в приватный Telegram-канал посредством взаимодействия с созданным ботом, имеющим права на отправку сообщений. Приватный канал подразумевает приглашение отдельных пользователей администратором канала и отсутствие публичного доступа к содержимому.

Пример работы системы в отладочном режиме приведен на рисунке 23.

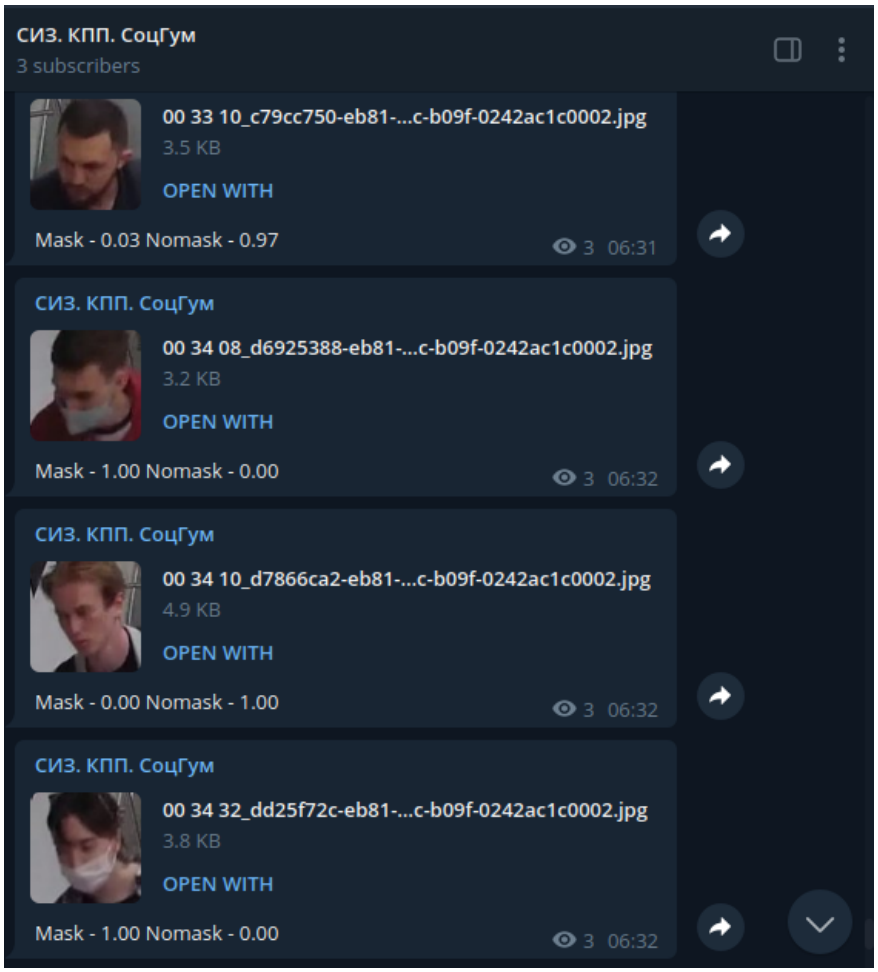

Рис. 23. Пример сообщений в Telegram-канале с результатами классификации, полученных в отладочном режиме работы системы

Каждое сообщение Telegram-канала в приведенном примере соответствует одному изображению с показателями вероятности принадлежности к каждому классу, что соответствует требованиям к выходным данным системы, сформулированным в Главе 2.

6.2. ТЕСТИРОВАНИЕ СИСТЕМЫ

Для проведения тестирования разработанной системы выявления лиц, не использующих СИЗ, были выбраны данные со следующими характеристиками:

- MP4-видеозапись длительностью 00:38:22, содержащая 57525 кадров (25 FPS) с записью одного часа реального времени буднего дня (среда): с 14:00 до 15:00;
- 169 человек, проходящих через КПП навстречу камере видеонаблюдения.

Тестирование системы проводилось в отладочном режиме в два запуска на GPU Tesla T4 и CPU Intel(R) Xeon(R) @ 2.20GHz для оценки скорости работы.

Результаты тестирования:

- количество детектированных лиц – 165, недетектированных лиц – 4;
- количество корректно классифицированных объектов – 164, некорректно – 1 (ложноотрицательная классификация);
- количество неуникальных детекций лица для одного объекта – 1;
- средняя скорость работы на CPU (за 57525 кадров) – 8.6 FPS;
- средняя скорость работы на GPU (за 57525 кадров) – 53.5 FPS.

На рисунках 24, 25 приведены примеры изображений недектированных лиц из-за отвернутого от камеры положения лица на протяжении всего движения объекта и сообщение с некорректно классифицированным объектом с правдивым классом «mask» (вероятности отнесения к классам: 0.48 – «mask», 0.52 – «no_mask»).

Рис. 24. Примеры случаев с лицами, недектированными из-за их положения относительно камеры на всех кадрах присутствия

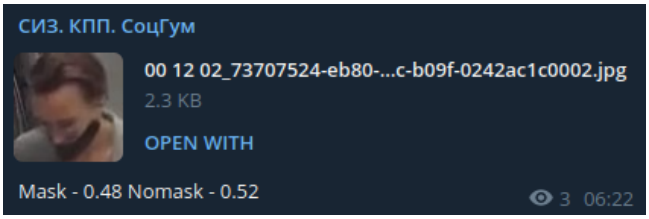

Рис. 25. Сообщение в Telegram-канале с некорректно классифицированным объектом правдивого класса «mask»

На рисунке 26 приведены примеры сообщений с корректно классифицированными лицами разных классов.

Рис. 26. Примеры сообщений в Telegram-канале с корректными результатами классификации, выбранные случайным образом

По результатам тестирования процент детектирования лиц в рамках разработанной системы составил 98%, Accuracy классификации – 0.99, скорость работы на GPU – 53.5 FPS с возможностью работы на CPU со скоростью 8.6 FPS. Разработанная система соответствует предъявленным требованиям.

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы была разработана автоматизированная система выявления лиц, не использующих СИЗ, на основе анализа видеоряда методами машинного обучения. Для этого были решены следующие задачи:

1. Проведен обзор существующих решений по определению наличия СИЗ на изображении, определены основные используемые компоненты и направления для проведения исследований.

2. Сформулированы требования к разрабатываемой системе, на основе которых проводился дальнейший анализ подходов и моделей, велась разработка системы.

3. Проанализированы подходы к детектированию и сопровождению лиц на видеоряде. Для разработки системы выбраны методы YOLOv5s и SORT.

4. С помощью разработанного на языке Python 3.8.10 скрипта на основе видеопотока длительностью 16:03:03 подготовлен набор данных с 5636 изображениями, размеченными по классам: «mask» (2738 изображений) и «no_mask» (2898 изображений).

5. Проведен сравнительный анализ моделей классификации. Для разработки системы выбрана модель EfficientNet-v2-B0 с Accuracy на тестовой выборке – 0.9858.

6. Разработана и протестирована автоматизированная система выявления лиц, не использующих СИЗ. Accuracy классификации по результатам тестирования – 0.99, скорость работы на GPU – 53.5 FPS.

В результате тестирования система показала высокие значения метрик классификации, удовлетворяя предъявленным требования.

Разработанная система полностью функционирует и готова к внедрению.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the COVID-19 pandemic / M. Loey, G. Manogaran, M.H.N. Taha, N.E.M. Khalifa // Measurement. 2021. Vol. 167. Art. 108288. 11 p. URL: https://doi.org/10.1016/j.measurement.2020.108288 (дата обращения: 02.03.2022).

2. Big Transfer (BiT): General Visual Representation Learning / A. Kolesnikov, L. Beyer, X. Zhai [et al.] // arXiv. Computer Science. Computer Vision and Pattern Recognition: [website]. 2020. Art. 1912.11370v3 [cs.CV]. 28 p. URL: https://doi.org/10.48550/arXiv.1912.11370v3 (дата обращения: 16.04.2022).

3. Cota D.A.M. Monitoring COVID-19 prevention measures on CCTV cameras using Deep Learning. Thesis for: Master degree in Computer Science Engineering – Data Science. Italy, Turin, 2020. 85 p. URL: http://dx.doi.org/10.13140/RG.2.2.16368.69124 (дата обращения: 17.02.2022).

4. Face Mask Detection Dataset // Kaggle: [website]. URL: https://www.kaggle.com/datasets/andrewmvd/face-mask-detection (дата обращения: 01.03.2022).

5. Fan X., Jiang M. RetinaFaceMask: A Single Stage Face Mask Detector for Assisting Control of the COVID-19 Pandemic // arXiv. Computer Science. Computer Vision and Pattern Recognition: [website]. 2021. Art. 2005.03950v3 [cs.CV]. 6 p. URL: https://doi.org/10.48550/arXiv.2005.03950 (дата обращения: 19.03.2022).

6. Fast and Flexible Human Pose Estimation with HyperPose / Y. Guo, J. Liu, G. Li [et al.] // Proceedings of the 29th ACM International Conference on Multimedia. Association for Computing Machinery. New York, NY, USA, 2021. Pp. 3763–3766. URL: https://doi.org/10.1145/3474085.3478325 (дата обращения: 27.04.2022).

7. Fighting against COVID-19: A novel deep learning model based on YOLO-v2 with ResNet-50 for medical face mask detection / M. Loey, G. Manogaran, M.H.N. Taha, N.E.M. Khalifa // Sustainable Cities and Society. 2021. Vol. 65. Art. 102600. 8 p. URL: https://doi.org/10.1016/j.scs.2020.102600 (дата обращения: 26.02.2022).

8. ImageNet-21K Pretraining for the Masses / T. Ridnik, E. Ben-Baruch, A. Noy, L. Zelnik-Manor // arXiv. Computer Science. Computer Vision and Pattern Recognition: [website]. 2021. Art. 2104.10972 [cs.CV]. 20 p. URL: https://doi.org/10.48550/arXiv.2104.10972 (дата обращения: 29.04.2022).

9. Masked Face Recognition Dataset and Application / Z. Wang, G. Wang, B. Huang // arXiv. Computer Science. Computer Vision and Pattern Recognition: [website]. 2020. Art. 2003.09093v2 [cs.CV]. 3 p. URL: https://doi.org/10.48550/arXiv.2003.09093 (дата обращения: 04.03.2022).

10. Multiple object tracking: A literature review / W. Luo, J. Xing, A. Milan [et al.] // Artificial Intelligence. 2021. Vol. 293. Art. 103448. 23 p. URL: https://doi.org/10.1016/j.artint.2020.103448 (дата обращения: 11.04.2022).

11. Nieto-Rodríguez A., Mucientes M., Brea V.M. System for Medical Mask Detection in the Operating Room Through Facial Attributes // Pattern Recognition and Image Analysis. IbPRIA 2015. Lecture Notes in Computer Science. Vol. 9117. Pp. 138–145. URL: http://persoal.citius.usc.es/manuel.mucientes/pubs/Nieto-Rodriguez15_ibpria.pdf (дата обращения: 12.05.2022).

12. On Empirical Comparisons of Optimizers for Deep Learning / D. Choi, C.J. Shallue, Z. Nado // arXiv. Computer Science. Machine Learning: [website]. 2020. Art. 1910.05446v3 [cs.LG]. 27 p. URL: https://doi.org/10.48550/arXiv.1910.05446v3 (дата обращения: 30.04.2022).

13. OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields / Z. Cao, G. Hidalgo, T. Simon [et al.] // arXiv. Computer Science. Computer Vision and Pattern Recognition: [website]. 2019. Art. 1812.08008 [cs.CV]. 14 p. URL: https://arxiv.org/abs/1812.08008v2 (дата обращения: 06.03.2022).

14. Pedestrian Dynamic and Kinematic Information Obtained from Vision Sensors / S.G. Konrad, M. Shan, F.R. Masson [et al.] // Proceedings of IEEE Intelligent Vehicles Symposium (IV). Changshu, China, 2018. Pp. 1299–1305. URL: https://doi.org/10.1109/IVS.2018.8500д (дата обращения: 05.05.2022).

15. Ríos B., Toscano M., Descoins A. Face mask detection in street camera video streams using AI: behind the curtain // tryolabs' blog: [website]. 2020. URL:

https://tryolabs.com/blog/2020/07/09/face-mask-detection-in-street-camera-video-streams-using-ai-behind-the-curtain (дата обращения: 12.06.2022).

16. Searching for MobileNetV3 / A.G. Howard, M. Sandler, G. Chu [et al.] // Proceedings of IEEE/CVF International Conference on Computer Vision. Seoul, Korea (South), 2019. Pp. 1314–1324. URL: https://doi.org/10.1109/ICCV.2019.00140 (дата обращения: 02.06.2022).

17. Simple online and realtime tracking / A. Bewley, Z. Ge, L. Ott [et al.] // arXiv. Computer Science. Computer Vision and Pattern Recognition: [website]. 2017. Art. 1602.00763v2 [cs.CV]. 5 p. URL: https://doi.org/10.48550/arXiv.1602.00763v2 (дата обращения: 11.05.2022).

18. SSDMNV2: A real time DNN-based face mask detection system using single shot multibox detector and MobileNetV2 / P. Nagrath, R. Jain, A. Madan [et al.] // Sustainable Cities and Society. 2021. Vol. 66. Art. 102692. 13 p. URL: https://doi.org/10.1016/j.scs.2020.102692 (дата обращения: 03.03.2022)

19. Tan M., Le Q.V. EfficientNetV2: Smaller Models and Faster Training // arXiv. Computer Science. Computer Vision and Pattern Recognition: [website]. 2021. Art. 2104.00298v2 [cs.CV]. 12 p. URL: https://doi.org/10.48550/arXiv.2104.00298v2 (дата обращения: 12.06.2022).

20. Wang B., Zheng J., Chen C.L.P. A Survey on Masked Facial Detection Methods and Datasets for Fighting Against COVID-19 // IEEE Transactions on Artificial Intelligence. 2022. Vol. 3. No. 3. Pp. 323–343. URL: https://doi.org/10.1109/TAI.2021.3139058 (дата обращения: 16.06.2022).

21. YOLO5Face: Why Reinventing a Face Detector / D. Qi, W. Tan, Q. Yao, J. Liu // arXiv. Computer Science. Computer Vision and Pattern Recognition: [website]. 2022. Art. 2105.12931v3 [cs.CV]. 10 p. URL: https://doi.org/10.48550/arXiv.2105.12931v3 (дата обращения: 05.05.2022).

22. You'll never walk alone: modeling social behavior for multi-target tracking / S. Pellegrini, A. Ess, K. Schindler, L. Van Gool // Proceedings of IEEE International Conference on Computer Vision. Kyoto, Japan, 2009. Pp. 261–268. URL: http://dx.doi.org/10.1109/ICCV.2009.5459260 (дата обращения: 05.04.2022).

23. You Only Look Once: Unified, Real-Time Object Detection / J. Redmon, S. Divvala, R. Girshick, A. Farhadi // Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. Las Vegas, NV, USA, 2016. Pp. 779–788. URL: https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Redmon_You_Only_Look_CVPR_2016_paper.pdf (дата обращения: 27.03.2022).

24. Баринов А.И., Баринова А.О., Катасёв А.С. Нейросетевая сверточная модель обнаружения нарушений масочного режима в общественных местах // Вестник НЦБЖД. 2021. № 4(50). С. 39–45. URL: https://www.elibrary.ru/item.asp?id=47339816 (дата обращения: 21.03.2022).

25. Курносов И.Л., Абламейко С.В. Обнаружение нарушений масочного режима и оценка его соблюдения по видеоряду // Развитие информатизации и государственной системы научно-технической информации: доклады XX Международной научно-технической конференции. Беларусь, Минск, 2021. 7 с. URL: http://fpmi.bsu.by/ImgFpmi/Cache/Page/79113.pdf (дата обращения: 21.03.2022).

26. Трудовой кодекс Российской Федерации: от 26 декабря 2001 г.: по состоянию на 01.03.2022 // Консультант Плюс: справочно-правовая система. URL: http://www.consultant.ru/document/cons_doc_LAW_34683/ (дата обращения: 20.02.2022).

Приложение 1

Класс обработки изображения ImageProcessor

```
class ImageProcessor:
    def __init__(self, input_FPS, detector_weights, classification_model_dir,
classification_model_img_size, is_stream):
        self.face_detector = FaceDetector(weights=detector_weights)
        self.mask_classificator = MaskClassificator(model_dir=classification_model_dir,
img_size=classification_model_img_size)
        self.tracker = Sort(max_age=30, min_hits=0)
        self.faces = dict()
        self.input_FPS = input_FPS
        self.is_stream = is_stream
        self.score_threshold = 0.8
        self.count_threshold = 2
        self.face_size_threshold = 35
        self.save_path = 'output'
        mkdir(self.save_path)
        self.saved_count=0
        self.telegram_token = ''
        self.telegram_channeld_id = -1001438367642

    def __add(self, id, arr, score, image_num):
        if score < self.score_threshold:
            return
        if id not in self.faces:
            self.faces[id] = []
        self.faces[id].append((arr,score, image_num))

def __remove(self, id):
        if id not in self.faces:
            return
        dataArr = self.faces[id]
        del self.faces[id]
        if len(dataArr) < self.count_threshold:
            return
        self.__save(dataArr)

    def __save(self,dataArr):
        img = dataArr[0][0]
        image_num = dataArr[0][2]
        for data in dataArr[:]:
            if img.shape[0] * img.shape[1] < data[0].shape[0] * data[0].shape[1]:
                img = data[0]
                image_num = data[2]
        class_probs = self.mask_classificator.proccess_image_classification(img)
        if self.is_stream:
            image_time = strftime("%H:%M:%S", gmtime())
        else:
            image_time = strftime("%H:%M:%S", gmtime(int(image_num/self.input_FPS)))
        id = str(uuid.uuid1())
        image_path = "{0}/{1}_{2}.jpg".format(self.save_path, image_time,id)
```

```
        cv2.imwrite(image_path, img)
        caption = 'Mask - {0:.2f} Nomask - {1:.2f}'.format(class_probs[0], class_probs[1])
        self.__send_image_telegram(image_path=image_path, caption=caption)

def __send_image_telegram(self,image_path, caption):
        files = {
           'document': open(image_path, 'rb'),
        }
        url = 'https://api.telegram.org/bot{0}/sendDocument?chat_id={1}&caption={2}'.format(
           self.telegram_token, self.telegram_channeld_id, caption)
        response = requests.post(url, files=files)
        if response.status_code != 200:
           print('Bad response code while send to telegram: ', response.status_code)
    def process_image(self, image, image_num):
        predicts = self.face_detector.detect_face_boxes(image)
        height,width,channels = image.shape
        result_image = image.copy()
        faces = []
        for predict in predicts:
           x1, y1, x2, y2, score = predict
           if x2-x1 > self.face_size_threshold and y2-y1 > self.face_size_threshold:
              faces.append(predict)
        trackers, removed = self.tracker.update(np.array(faces))
        for tracker_data in trackers:
           confidence_score = tracker_data[5]
           tracker_data = tracker_data.astype(np.int32)
           x1 = 0 if tracker_data[0] < 0 else tracker_data[0]
           y1 = 0 if tracker_data[1] < 0 else tracker_data[1]
           x2 = width if tracker_data[2] > width else tracker_data[2]
           y2 = height if tracker_data[3] > height else tracker_data[3]

           face = image[y1:y2, x1:x2]
           self.__add(tracker_data[4],face, confidence_score,image_num)

         for rm in removed:
           self.__remove(rm)
```

Приложение 2

Класс детектирования лица FaceDetector

```
class FaceDetector:
    def __init__(self, weights):
        self.device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
        self.model = attempt_load(weights, map_location=self.device)
        self.img_size = 320
        self.conf_thres = 0.2
        self.iou_thres = 0.2

    def __preprocess_image(self, orgimg):
        img0 = copy.deepcopy(orgimg)
        h0, w0 = orgimg.shape[:2]
        r = self.img_size / max(h0, w0)

        if r != 1:
            interp = cv2.INTER_AREA if r < 1  else cv2.INTER_LINEAR
            img0 = cv2.resize(img0, (int(w0 * r), int(h0 * r)), interpolation=interp)

        imgsz = check_img_size(self.img_size, s=self.model.stride.max())
        img = letterbox(img0, new_shape=imgsz)[0]
        img = img[:, :, ::-1].transpose(2, 0, 1).copy()

        img = torch.from_numpy(img).to(self.device)
        img = img.float()
        img /= 255.0  # 0 - 255 to 0.0 - 1.0
        if img.ndimension() == 3:
            img = img.unsqueeze(0)
        return img

    def __postprocess_prediction(self,orgimg,img,pred):
        h0, w0 = orgimg.shape[:2]
        ret = []
        for i, det in enumerate(pred):
            gn = torch.tensor(orgimg.shape)[[1, 0, 1, 0]].to(self.device)
            if len(det):
                det[:, :4] = scale_coords(img.shape[2:], det[:, :4], orgimg.shape).round()
                for j in range(det.size()[0]):
                    conf = det[j, 4].cpu().numpy()
                    x_top_left = int(det[j,0])
                    y_top_left = int(det[j,1])
                    x_bottom_right = int(det[j,2])
                    y_bottom_right = int(det[j,3])

                    width = x_bottom_right - x_top_left
                    height = y_bottom_right - y_top_left

                    x_top_left = int(x_top_left - width*0.4)
                    y_top_left = int(y_top_left - height*0.4)
                    x_bottom_right = int(x_bottom_right + width*0.4)
                    y_bottom_right = int(y_bottom_right + height*0.4)

                    x_top_left = 0 if x_top_left < 0 else x_top_left
                    y_top_left = 0 if y_top_left < 0 else y_top_left
                    x_bottom_right = w0 if x_bottom_right > w0 else x_bottom_right
```

```
            y_bottom_right = h0 if y_bottom_right > h0 else y_bottom_right
            ret.append([x_top_left, y_top_left, x_bottom_right, y_bottom_right, conf])
    return ret
```