Министерство науки и высшего образования РФ

Федеральное государственное автономное

образовательное учреждение высшего образования

«**СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ**»

Институт космических и информационных технологий

Кафедра информационной безопасности

УТВЕРЖДАЮ Заведующий кафедрой

\_\_\_\_\_\_\_ В.И. Вайнштейн

« \_\_\_\_\_» \_\_\_\_\_\_\_\_ 20 \_\_\_ г.

**ДИПЛОМНАЯ РАБОТА**

10.05.01 Компьютерная безопасность

Разработка системы автоматизированного прокторинга на основе непрерывной аутентификации с элементами машинного обучения.

Руководитель \_\_\_\_\_\_\_\_ доцент кафедры ИБ,

подпись, дата канд. физ.-мат. наук М.М. Кучеров

Выпускник \_\_\_\_\_\_\_\_ В.А. Хамаганов

подпись, дата

Рецензент \_\_\_\_\_\_\_\_ доцент кафедры ВТ,

подпись, дата канд. тех. наук В.Г Серёдкин

Красноярск 2023

**РЕФЕРАТ**

Выпускная квалификационная работа на тему «Разработка системы автоматизированного прокторинга на основе непрерывной аутентификации с элементами машинного обучения» содержит 81 страницу текстового документа, 12 иллюстраций, 8 таблиц, 8 приложений, 28 использованных источников.

СИСТЕМА ПРОКТОРИНГА, АУТЕНТИФИКАЦИЯ, МАШИННОЕ ОБУЧЕНИЕ, БИОМЕТРИЯ, ЭКЗАМЕН, МОДЕЛЬ НЕЙРОННОЙ СЕТИ, РАЗРАБОТКА.

Цель работы: создание системы автоматизированного прокторинга на основе непрерывной аутентификации.

Задачи:

-Обзор и анализ литературных источников в области автоматизированного прокторинга с использованием методов непрерывной аутентификации;

-Определение требований к реализуемой системе и сдающему;

-Разработка архитектуры системы прокторинга и дизайна экзамена.;

-Выбор методов и инструментов для разработки системы прокторинга;

- Реализация десктопной системы прокторинга.

В итоге работы представлена реализация десктопной системы прокторинга и протестирована её эффективность.

СОДЕРЖАНИЕ

[**ВВЕДЕНИЕ** 5](#_Toc127267962)

[**1.** **Теоретическая часть** 9](#_Toc127267963)

[1.1 Теория машинного обучения 9](#_Toc127267964)

[1.1.1 Машина опорных векторов 9](#_Toc127267965)

[1.1.2 Генетический алгоритм 10](#_Toc127267966)

[1.1.3 Глубокое обучение 10](#_Toc127267967)

[1.1.4 Свёрточные нейронные сети 11](#_Toc127267968)

[1.2 Биометрия 12](#_Toc127267969)

[1.1.1 Поведенческая биометрия 13](#_Toc127267970)

[1.1.2 Физиологическая биометрия 13](#_Toc127267971)

[1.3 Обзор литературы 14](#_Toc127267972)

[1.3.1 Методология исследования статей 14](#_Toc127267973)

[1.3.2 Обзор систем с человеком в качестве проктора 15](#_Toc127267974)

[1.3.3 Обзор систем с алгоритмом машинного обучения в качестве проктора 19](#_Toc127267975)

[**2.** **Проектная часть** 22](#_Toc127267976)

[2.1 Требования 22](#_Toc127267977)

[2.1.1 Требования к системе 22](#_Toc127267978)

[2.1.2 Требования к студенту 22](#_Toc127267979)

[2.2 Выработка дизайна экзамена 24](#_Toc127267980)

[2.2.1 Дизайн экзамена со стороны студента 24](#_Toc127267981)

[2.2.2 Дизайн экзамена со стороны проктора 24](#_Toc127267982)

[2.3 Выработка архитектуры системы 25](#_Toc127267983)

[3.3.1 Десктопное приложение 27](#_Toc127267984)

[3.3.2 Серверная часть 28](#_Toc127267985)

[3.3.3 Клиентская часть 31](#_Toc127267986)

[**3.** **Реализация** 32](#_Toc127267987)

[3.1 Используемые в работе предобученные модели 32](#_Toc127267988)

[3.1.1 Модель для отслеживания телефона в кадре 32](#_Toc127267989)

[3.1.2 Модель для отслеживания поворота головы 33](#_Toc127267990)

[3.1.3 Модель для обнаружения лиц 35](#_Toc127267991)

[3.1.4 Модель для распознавания лиц 36](#_Toc127267992)

[3.2 Реализация десткопной части системы 37](#_Toc127267993)

[3.2.1 Класс проведения экзамена 37](#_Toc127267994)

[3.2.2 Класс обработки видео 38](#_Toc127267995)

[3.2.3 Аутентификация по лицу 39](#_Toc127267996)

[3.2.4 Системы отслеживающие нарушения 40](#_Toc127267997)

[3.2.5 Аутентификация по динамике движения мыши 42](#_Toc127267998)

[3.2.6 Определение нарушения 43](#_Toc127267999)

[**4 Тестирование и сравнение** 46](#_Toc127268000)

[4.1 Тестирование 46](#_Toc127268001)

[4.2 Сравнение реализуемой системы с аналогами 49](#_Toc127268002)

[**ЗАКЛЮЧЕНИЕ** 52](#_Toc127268003)

[**СПИСОК СОКРАЩЕНИЙ** 53](#_Toc127268004)

[**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ** 54](#_Toc127268005)

[**Приложение А** 57](#_Toc127268006)

[**Приложение Б** 60](#_Toc127268007)

[**Приложение В** 63](#_Toc127268008)

[**Приложение Г** 65](#_Toc127268009)

[**Приложение Д** 67](#_Toc127268010)

[**Приложение Е** 70](#_Toc127268011)

[**Приложение Ё** 72](#_Toc127268012)

[**Приложение Ж** 77](#_Toc127268013)

# **ВВЕДЕНИЕ**

С развивающимися технологиями совершенствуются методики обучения, используемые для образования студентов. Одной из таких методик является онлайн образование, которое позволяет участникам образовательного процесса не присутствовать очно при проведении занятий. Это расширяет возможности образовательных учреждений в масштабируемости своих возможностей, поскольку иногородние студенты могут обучаться, не тратя свои ресурсы на переезд и проживание, кроме того, образовательные учреждения также экономят ресурсы на предоставление мест проживания. Также активно набирает популярность такой формат обучения как онлайн курсы. Онлайн курсы предоставляют образовательные услуги своим студентам в сети интернет без необходимости очного присутствия и даже прохождения вступительных экзаменов.

Чем больше ресурсов предоставляют услуги онлайн образования, тем острее стоит вопрос об аттестации, поскольку аттестация традиционно проводится очно, так как у преподавателя есть возможность следить за честностью проведения экзамена. Однако онлайн курсы в основном используют услуги онлайн прокторинга. Онлайн прокторинг – это процедура, при которой проктор выполняет функции преподавателя в онлайн режиме, то есть следит за процессом сдачи экзамена через веб-камеру и отвечает за честность его проведения.

Такой подход эффективен, однако у него существует несколько проблем. Основная проблема такого подхода заключается в том, что такая система прокторинга мало масштабируема. Если один проктор способен следить за одним классом, тогда на несколько классов понадобится несколько прокторов, это значит, что количество ресурсов, необходимое для использования такого метода, при увеличении масштаба системы, растёт в геометрической прогрессии. Также, возможной угрозой честности проведения онлайн экзамена является недоверие к проктору.

Таким образом возникает потребность в автоматизации процесса онлайн прокторинга. Автоматизация позволяет успешно масштабировать систему, избавиться от проблемы доверия к проктору, а также позволяет использовать меньше человеческих ресурсов.

Однако, степени автоматизации в сфере онлайн прокторинга различаются. Зачастую такие системы всё же требуют наличия проктора, однако количество одновременно сдающих студентов в системе больше, чем в системе, полностью полагающейся на прокторинг человеком, так как автоматизация позволяет проктору отслеживать большее количество студентов с помощью алгоритмов, которые определяют нечестное поведение. В основном они строятся на машинном обучении и используют биометрические (физиологические и поведенческие) характеристики пользователя.

**Актуальность работы**: на данный момент существует большое множество систем онлайн прокторинга, которые используют машинное обучение в качестве средств для распознавания личности, а также для отслеживания подозрительного поведения студентов. Однако, большинство работ используют унибиометрические системы аутентификации. Аутентификация пользователя по физиологическим характеристикам – это наиболее эффективный метод аутентификации, однако в большинстве случаев он требует дополнительного оборудования. Например, для аутентификации пользователя по отпечатку пальца требуется сканер отпечатков пальцев, также для любого физиологического параметра необходимо своё устройство считывания. Камера – наиболее распространённое устройство, которое позволяет считывать характеристики лица, что и приводит систему аутентификации по лицу к такой популярности. В предлагаемой работе используется система, комбинирующая аутентификацию пользователя по физиологическим и поведенческим характеристикам, что позволяет обеспечить наибольшую возможную достоверность аутентификации, а также уменьшение возможного риска компрометации экзамена.

**Цель**: создание системы автоматизированного прокторинга на основе непрерывной аутентификации.

**Задачи**:

1. Обзор и анализ литературных источников в области автоматизированного прокторинга с использованием методов непрерывной аутентификации.

2. Определение требований к реализуемой системе и сдающему.

3. Разработка архитектуры системы прокторинга и описание дизайна экзамена.

4. Выбор методов и инструментов для разработки системы прокторинга.

5. Реализация десктопной системы прокторинга.

Работа представляет собой описание проекта и реализации системы прокторинга на основе непрерывной аутентификации с элементами машинного обучения. Работа разделена на четыре части:

1. Теоретическая часть. Представляет собой описание информации, необходимой для реализации системы прокторинга, а также обзор литературы.

2. Проектная часть. Содержит описание архитектуры системы, требований к системе и сдающему, также дизайна экзамена и инструментов для реализации.

3. Часть с описанием реализации системы, методов и выбранных моделей машинного обучения.

4. Тестирование и сравнение системы с аналогами.

# **Теоретическая часть**

Прокторинг – это процедура контроля честности сдачи экзамена студентом, осуществляемая проктором. Проктором в этом случае может выступать как человек, так и обученный алгоритм. Задача проктора сводится к отслеживанию подозрительных действий студента. На онлайн экзамене большинство подозрительных действий расцениваются как попытка нечестным образом получить лучшие результаты на экзамене или тесте. В таком случае, если проктор обнаружит подозрительные действия, его задачей будет отреагировать на них. В зависимости от дизайна экзамена, проктор на разные действия реагирует по-разному в разных системах.

## 1.1 Теория машинного обучения

Машинное обучение — класс методов искусственного интеллекта, характерной чертой которых является не прямое решение задачи, а обучение за счёт применения решений множества сходных задач. Для построения таких методов используются средства математической статистики, численных методов, математического анализа, методов оптимизации, теории вероятностей, теории графов, различные техники работы с данными в цифровой форме.

### 1.1.1 Машина опорных векторов

Метод опорных векторов — набор схожих алгоритмов обучения с учителем, использующихся для задач классификации и регрессионного анализа. Принадлежит семейству линейных классификаторов и может также рассматриваться как частный случай регуляризации по Тихонову. Особым свойством метода опорных векторов является непрерывное уменьшение эмпирической ошибки классификации и увеличение зазора, поэтому метод также известен как метод классификатора с максимальным зазором.

Основная идея метода — перевод исходных векторов в пространство более высокой размерности и поиск разделяющей гиперплоскости с наибольшим зазором в этом пространстве. Две параллельных гиперплоскости строятся по обеим сторонам гиперплоскости, разделяющей классы. Разделяющей гиперплоскостью будет гиперплоскость, создающая наибольшее расстояние до двух параллельных гиперплоскостей. Алгоритм основан на допущении, что чем больше разница или расстояние между этими параллельными гиперплоскостями, тем меньше будет средняя ошибка классификатора.

### 1.1.2 Генетический алгоритм

Генетический алгоритм — это эвристический алгоритм поиска, используемый для решения задач оптимизации и моделирования путём случайного подбора, комбинирования и вариации искомых параметров с использованием механизмов, аналогичных естественному отбору в природе. Является разновидностью эволюционных вычислений, с помощью которых решаются оптимизационные задачи с использованием методов естественной эволюции, таких как наследование, мутации, отбор и кроссинговер. Отличительной особенностью генетического алгоритма является акцент на использование оператора «скрещивания», который производит операцию рекомбинации решений-кандидатов, роль которой аналогична роли скрещивания в живой природе.

### 1.1.3 Глубокое обучение

Глубокое обучение — совокупность методов машинного обучения (с учителем, с частичным привлечением учителя, без учителя, с подкреплением), основанных на обучении представлениям, а не специализированных алгоритмах под конкретные задачи. Многие методы глубокого обучения были известны ещё в 1980-е, но результаты не впечатляли, пока продвижения в теории искусственных нейронных сетей (предобучение нейросетей с помощью специального случая ненаправленной графической модели, так называемой ограниченной машины Больцмана) и вычислительные мощности середины 2000-х годов (в том числе использующие графические ускорители, программируемые пользователем вентильные матрицы и различные формы нейронных процессоров) не позволили создавать сложные технологические архитектуры нейронных сетей, обладающие достаточной производительностью и позволяющие решать широкий спектр задач, не поддававшихся эффективному решению ранее, например, в компьютерном зрении, машинном переводе, распознавании речи, причём качество решения во многих случаях теперь сопоставимо, а в некоторых превосходит эффективность человека.

### 1.1.4 Свёрточные нейронные сети

Свёрточная нейронная сеть — специальная архитектура искусственных нейронных сетей, предложенная Яном Лекуном в 1988 году и нацеленная на эффективное распознавание образов, входит в состав технологий глубокого обучения. Использует некоторые особенности зрительной коры, в которой были открыты так называемые простые клетки, реагирующие на прямые линии под разными углами, и сложные клетки, реакция которых связана с активацией определённого набора простых клеток. Таким образом, идея свёрточных нейронных сетей заключается в чередовании свёрточных слоёв и субдискретизирующих слоёв. Структура сети — однонаправленная (без обратных связей), принципиально многослойная. Для обучения используются стандартные методы, чаще всего метод обратного распространения ошибки. Функция активации нейронов (передаточная функция) — любая, по выбору исследователя.

Название архитектура сети получила из-за наличия операции свёртки, суть которой в том, что каждый фрагмент изображения умножается на матрицу (ядро) свёртки поэлементно, а результат суммируется и записывается в аналогичную позицию выходного изображения.

Работа свёрточной нейронной сети обычно интерпретируется как переход от конкретных особенностей изображения к более абстрактным деталям, и далее к ещё более абстрактным деталям вплоть до выделения понятий высокого уровня. При этом сеть самонастраивается и вырабатывает сама необходимую иерархию абстрактных признаков (последовательности карт признаков), фильтруя маловажные детали и выделяя существенное.

Подобная интерпретация носит скорее метафорический или иллюстративный характер. Фактически «признаки», вырабатываемые сложной сетью, малопонятны и трудны для интерпретации настолько, что на практике суть этих признаков даже не пытаются понять, тем более «подправлять», а вместо этого для улучшения результатов распознавания меняют структуру и архитектуру сети. Так, игнорирование системой каких-то существенных явлений может говорить о том, что либо не хватает данных для обучения, либо структура сети обладает недостатками, и система не может выработать эффективных признаков для данных явлений.

## 1.2 Биометрия

Биометрия – это способ идентификации и аутентификации личности по индивидуальным характеристикам.

Основным структурным элементом предлагаемой работы является биометрическая подсистема, поэтому в следующем разделе следует описание биометрических параметров, используемых в этой работе.

Разделять биометрические системы можно по типу собираемых данных, тогда они делятся на поведенческие и физиологические. Физиологическая биометрия работает, основываясь на физиологических характеристиках объекта, с которым работает. Например, если биометрическая система направлена на распознавание пользователя по отпечатку пальца, такая система называется физиологической. Поведенческая биометрия распознает поведенческие характеристики, например распознавание человека по стилю его ходьбы.

Поведенческие паттерны изменчивы в зависимости от психологического или физического состояния пользователя такой системы, также на изменение поведенческих паттернов влияют навыки и в глобальном плане - время [1].

Использование поведенческих биометрических систем вместе с физиологическими позволяет добиться наибольшей достоверности результата [2].

Из области поведенческой биометрии в данной работе будут рассмотрены системы распознавания пользователя по динамике движения мыши, из области физиологической будут рассмотрены следующие биометрические системы: отслеживание положения головы, обнаружение лиц и их распознавание.

Стоит отметить, что с течением времени, технологии, направленные на отслеживание поведенческих и физиологических характеристик, развиваются и меняются подходы к их созданию и использованию. Рассмотренные далее алгоритмы являются лишь выбранными подходами к решению задач извлечения биометрических характеристик.

### 1.1.1 Поведенческая биометрия

Поведенческая биометрия отслеживает поведенческие характеристики пользователя, анализирует способы взаимодействия с датчиком, который собирает информацию об этом взаимодействии. В результате своей работы, поведенческая биометрия создаёт уникальную для каждого пользователя модель поведения. Отличительной чертой такой биометрии является её пассивность, она не требует активных действий от пользователя. Это позволяет использовать такой тип биометрии в ситуациях, когда активное вмешательство вредит опыту использования системы.

Проектируемая система включает модель динамики движения мыши, одну из представителей поведенческой биометрии.

### 1.1.2 Физиологическая биометрия

Физиологическая биометрия считывает физиологические характеристики пользователя, такие как черты лица, строение радужной оболочки глаза, отпечатки пальцев, форму руки и т.д. В основном, для использования такого рода биометрии необходим интрузивный метод использования системы, что мешает использовать такую биометрию в системах, где активный подход неуместен. Однако, такие системы наиболее точны и используются чаще.

Проектируемая система включает в себя несколько моделей, использующих физиологическую биометрию: модель обнаружения лиц, отслеживания положения головы и модель распознавания лиц.

## 1.3 Обзор литературы

### 1.3.1 Методология исследования статей

Для определения наиболее важных элементов исследуемых систем в работе будет использована следующая методология:

1. Дизайн экзамена – в этом разделе исследуются методы взаимодействия между участниками экзамена (студент и проктор) и системой.
2. Аутентификация пользователя – каким образом в системе проводится аутентификация.
3. Биометрические параметры – использованные в исследуемой системе биометрии для аутентификации пользователя и обнаружения подозрительных действий.
4. Архитектура системы – структура взаимодействия всех элементов системы.
5. Метод принятия решений – результатом работы системы прокторинга является решение о списывании/ не списывании либо неверной аутентификации. В данном разделе исследуется метод, который использует система для принятия такого решения.

### 1.3.2 Обзор систем с человеком в качестве проктора

Рассматривая статью [3] по заданной методологии, составим комплексное представление о работе исследуемой системы. Для студента экзамен выглядит следующим образом: по задумке авторов, система прокторинга должна быть интегрирована в среду обучения таким образом, что если студент не активировал систему прокторинга, экзамен начать будет невозможно. После активации программа запрашивает доступ к камере и микрофону, далее программа записывает и отмечает моменты, в которых студент нарушал условия сдачи экзамена. Если студент нарушил правила, ему выдается предупреждение, после второго предупреждения экзамен останавливается и у преподавателя есть возможность его возобновить. Собственно идеальным сценарием сдачи экзамена будет тот, при котором студент держит голову опущенной, глаза направлены на лист бумаги, а в кадре присутствует только участник. Также в кадре нет мобильных телефонов, рот закрыт и в комнате не слышно разговоров. Аутентификация пользователя в данной системе не предусмотрена, так как интеграция в среду обучения позволит аутентифицироваться там. Биометрические параметры, собираемые в этой системе, ограничиваются информацией с камеры и микрофона. Далее информация с камеры поступает на вход лицевого детектора, который в свою очередь отмечает ориентиры глаз, рта и носа. В данной работе отслеживается направление взгляда на найденном лице, открытие и закрытие рта, что интерпретируется в данной работе как признак того, что отслеживаемое лицо говорит. Также отслеживается нахождение мобильных телефонов в кадре, за что отвечает детектор объектов и отслеживается поза головы. Кроме того, в работе представлена подсистема распознавания речи, она используется для интерпретации слов, которые произносит экзаменуемый. Результат распознавания сохраняется в текстовый файл, который доступен проктору. Для этой цели авторы используют Google API.

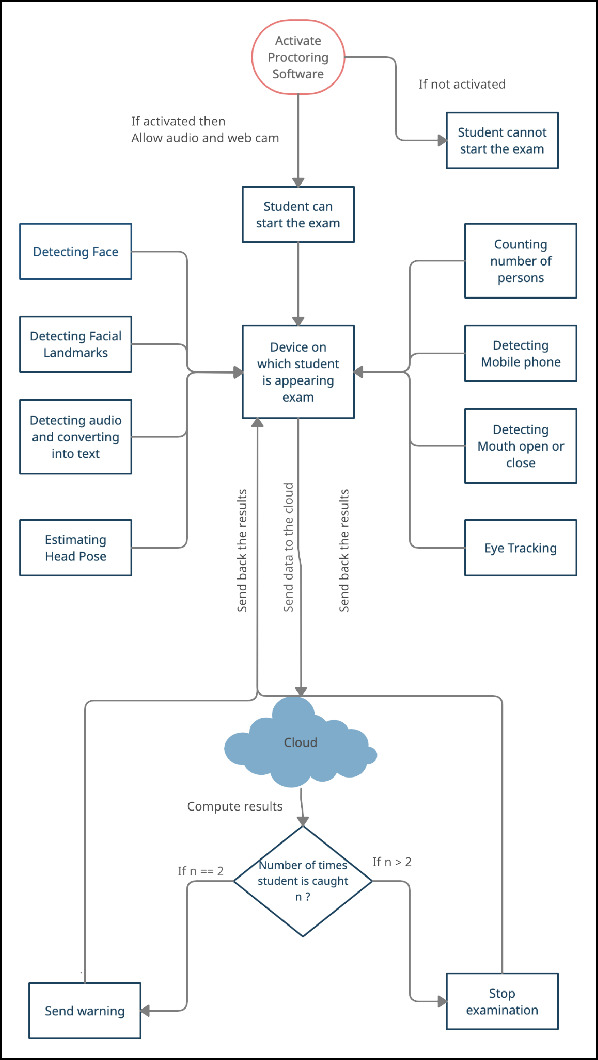
Архитектура системы представлена на рисунке 1.

Рисунок 1.

Как видно на рисунке, информация, поступающая с биометрических детекторов, отправляется на облако, где она и хранится. Вся отмеченная информация несет за собой последствия в виде предупреждений или остановки экзамена. Эту информацию оценивают прокторы и в их полномочиях снять эти отметки. Предпоследним этапом в исследовании этой работы является анализ метода принятия решений, которым является проктор. Таким образом, можно сказать, что описанная работа направлена в основном на экзамен, приближённый к реальному, так как разрешается использование черновиков и сам прокторинг направлен скорее на помощь проктору, чем на полную его замену. Кроме того, работа не предоставляет подробных пояснений работы своих биометрических подсистем.

Исследуя статью [4], можно сказать, что дизайн этой работы для студента выглядит следующим образом: перед сдачей экзамена, студент регистрируется на экзаменационном портале со своими учётными данными. Для решения проблемы аутентификации в данной работе используется двухфакторная аутентификация. В регистрационных данных кроме имени пользователя и пароля также содержится информация об адресе электронной почты, номере телефона, и несколько фотографий лица студента на разном расстоянии от камеры и с разным освещением. Камеру необходимо использовать на устройстве, с которого будет сдаваться экзамен. С этой информацией создаётся профиль в базе данных системы. Когда студенту необходимо будет сдать экзамен он аутентифицирует себя по логину, паролю, секретному коду, который придёт ему на номер телефона, далее его перенаправляют на страницу с распознаванием лица. Студенту необходимо снова сделать свои фотографии, по которым система сможет его аутентифицировать. После этой процедуры студента допускают к сдаче экзамена.

Для экзаменатора регистрация будет выглядеть идентично, однако процесс подготовки к экзамену и сам экзамен отличаются. Подготовка экзаменатора начинается с процесса создания блока вопросов, в котором ему будет необходимо выбрать тип вопроса и самому описать содержание. Такая система была разработана для избежания получения всеми студентами одинаковых вопросов в одинаковом порядке. Кроме того, сами вопросы представлены в виде изображений для предотвращения копирования и поиска ответов. В процессе экзамена проктор имеет возможность просматривать записи с камер и микрофонов студентов, а также получает оповещения от системы с информацией о происходящей на экзамене деятельности студента. Система распознаёт объекты в кадре и присылает проктору оповещения. В результате таких оповещений проктор имеет возможность отправить предупреждение или отстранить студента от экзамена.

В исследуемой работе используется алгоритм распознавания лица для аутентификации пользователя и отслеживание объектов для генерации оповещений проктору.

Для начала использования системы используется портал регистрации, который в свою очередь отправляет регистрационную информацию в базу данных. Соответственно перед началом экзамена происходит аутентификация и система допускает студентов к сдаче экзамена. Таким образом архитектура представлена для студента. Для управления процессом экзамена, проктору представлена отдельная страница, на которой отображается информация о ходе экзамена. Исследуемая система полностью полагается на проктора в решении вопроса о списывании студента.

Данная работа требовательна к работе преподавателя в отношении составления списка вопросов на экзамен. Система не предусматривает существования готовых вопросов к экзамену. Так как у стороны преподавателя нет намерения выдать за себя другого человека, не имеет смысла регистрировать преподавателя с использованием аутентификации по лицу.

В [5] работе авторы не предлагают собственного дизайна экзамена, однако подразумевают, что среда обучения, в которую будет внедрена эта система прокторинга имеет хранилище, в котором пользователи могут хранить свои фотографии. Аутентификация пользователя проводится с помощью нейронной сети распознавания лиц. В прокторинге участвуют системы отслеживания движений головы, обнаружения микроэкспрессии и обнаружения глаз с отслеживанием взгляда. По заявлениям авторов частое движение головы, выражение эмоций на экзамене и направление взгляда не в экран свидетельствуют о неправомерных действиях, что должно вести к предупреждению. Архитектура системы представлена на рисунке 2.

Рисунок 2.

Авторы также не предлагают меры оценки для используемых подсистем прокторинга, однако утверждают, что оценку может проводить как человек, так и ИИ.

В системе прокторинга использовать человека как проктора может быть опасно по нескольким причинам. Во-первых, человек имеет свойство ошибаться, что принято называть человеческим фактором. Во-вторых, межличностные отношения могут повлиять на решения проктора, возможно даже не осознано. В-третьих, проктор может намеренно компрометировать экзамен. Кроме того, использование человека как проктора расходует больше ресурсов, чем использование алгоритмов, а также возникает вышеупомянутая проблема с масштабируемостью системы. Поэтому некоторые системы прокторинга полностью полагаются на алгоритмы.

### 1.3.3 Обзор систем с алгоритмом машинного обучения в качестве проктора

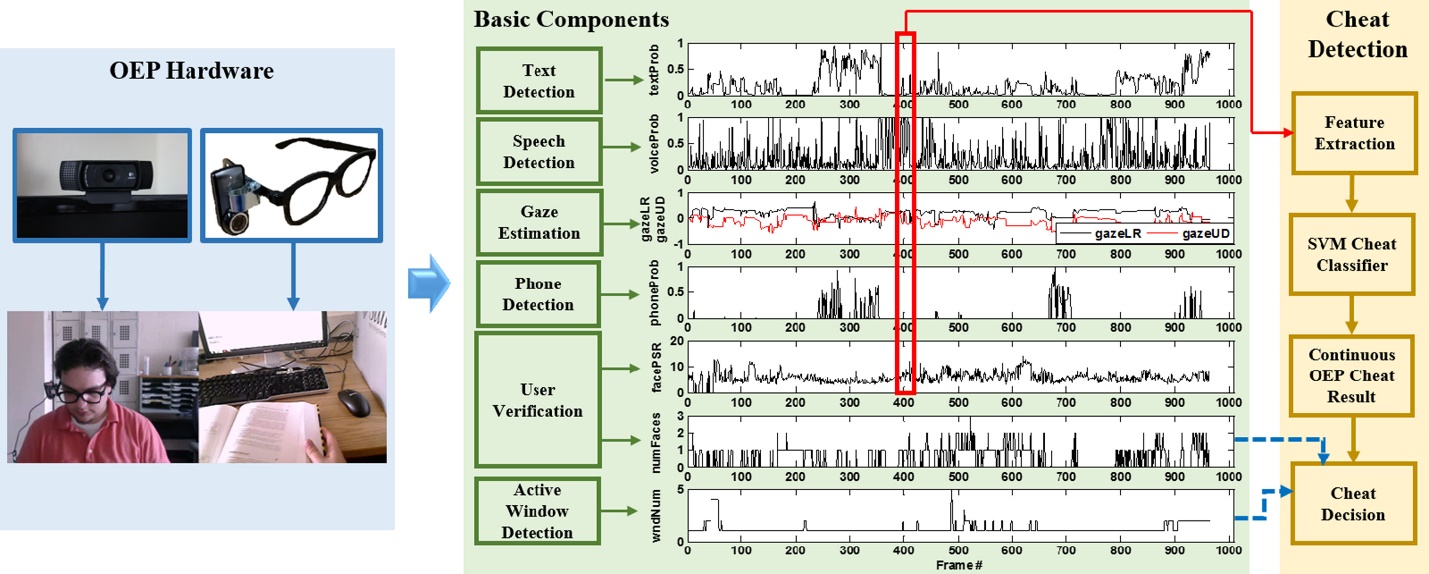
В работе [6] система разделяется на две фазы – фазу подготовки и фазу экзамена. Студент на этапе подготовки проходит аутентификацию по лицу и паролю, калибрует биометрические датчики, и подтверждает условия пользования системой. Далее, на этапе экзамена, студент сдаёт экзамен под непрерывным мониторингом системы прокторинга. В системе используется веб-камера, камера на голове сдающего и микрофон. Данные, записанные этими датчиками, обрабатываются с помощью шести компонентов для извлечения признаков среднего уровня. Этими компонентами являются: аутентификация пользователя, обнаружение текста, обнаружение речи, обнаружение активного окна, отслеживание направления взгляда и обнаружение телефона. Далее признаки среднего уровня в пределах временного окна объединяются для создания признаков высокого уровня, которые затем используются для обучения и тестирования классификатора, определяющего списывание. Архитектура системы представлена на рисунке 3.

Рисунок 3.

Аутентификация пользователя происходит на протяжении всего экзамена в рамках временных окон. Временное окно – фиксированное время сбора информации. После сбора информация обрабатывается и классифицируется. И результатом работы системы будет решение классификатора о списывании.

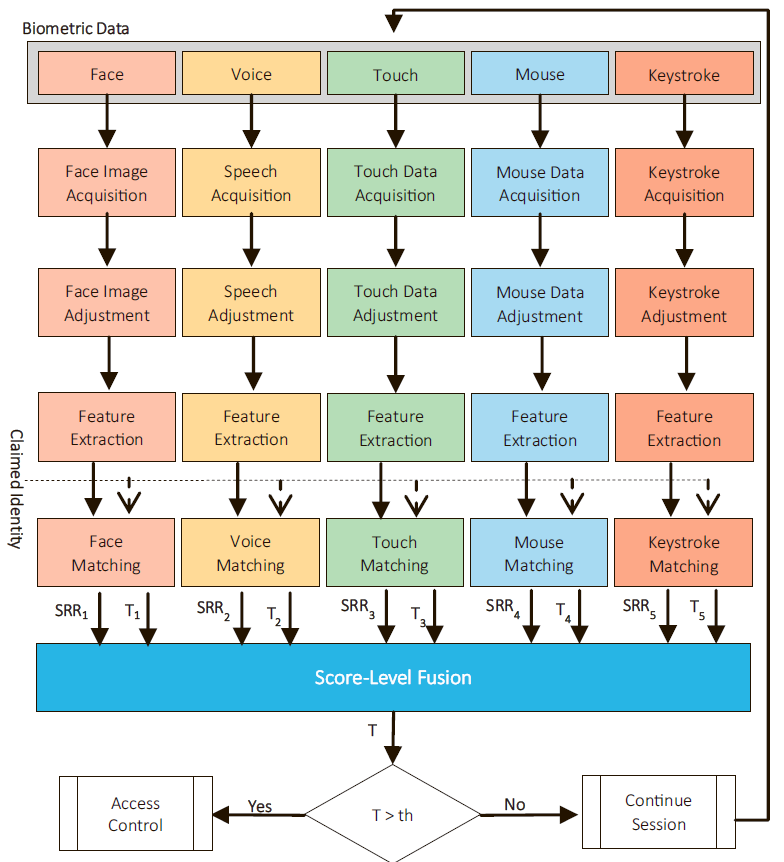
В работе [7] авторы предлагают мультибиометрическую систему аутентификации, поэтому дизайн экзамена остается на ответственности среды тестирования. Аутентификация пользователя в этой системе так же, как и в прошлой работает во временных окнах. Датчиками системы являются камера, микрофон, мышь, клавиатура и сенсорная панель. Из информации, собранной датчиками, система извлекает такие характеристики, как изображение лица пользователя, голос, паттерны сенсорного ввода, паттерны движения мышью и использования клавиатуры. Эти характеристики классифицируются отдельно и каждой выставляются баллы, после чего результаты смешиваются для получения глобального ответа аутентификации. Архитектура системы представлена на рисунке 4.

Рисунок 4.

Глобальный результат аутентификации сравнивается со специальным порогом принятия, и, если он ниже, студент продолжает сессию, в ином случае экзамен останавливается.

# **Проектная часть**

## 2.1 Требования

### 2.1.1 Требования к системе

Для того, чтобы определить необходимые параметры системы, для начала нужно определить требования к ней. Одна из задач проектируемой системы прокторинга заключается в непрерывной аутентификации пользователя, так как в процессе прохождения экзамена, возникают риски подмены пользователя, что ставит под угрозу честность прохождения экзамена. Кроме того, система должна определять, когда студент списывает и уведомлять об этом как самого студента, так и преподавателя.

Так, как проектируемая система предназначена для внедрения в процесс экзамена, важно чтобы аутентификация и отслеживание нечестного поведения были не интрузивными, чтобы не отвлекать участников экзамена от его сдачи.

Для того, чтобы система прокторинга не зависела от платформы для сдачи экзаменов, необходимо избавить её от интеграции в среду обучения, что скажется на её полномочиях для остановки экзамена в случае критического нарушения. Однако, эту проблему можно решить, используя проктора – преподавателя.

Итак, система должна:

1. Непрерывно аутентифицировать пользователя
2. Непрерывно отслеживать честность сдачи
3. Уведомлять пользователей системы о нарушениях

### 2.1.2 Требования к студенту

Так как работа посвящена системе, задача которой отслеживать нарушения в поведении студентов на экзамене, имеет смысл описать модель поведения, для которой система не определит нарушений, а также сами нарушения, которые могут привести к предупреждению от преподавателя и отстранению от экзамена.

Для удовлетворения системы аутентификации, достаточно следовать инструкциям системы и предоставлять корректные данные о себе на этапе регистрации, а также создать условия, при которых система аутентификации сможет распознать лицо сдающего, а именно – удовлетворительное освещение и направление головы к монитору.

Чтобы не получить замечания от системы прокторинга, студенту необходимо не отворачивать голову от монитора без необходимости, создавать условия, при которых камера сможет запечатлеть лицо, в кадре отсутствуют мобильные устройства, посторонние окна не открываются.

Так как мобильное устройство считается потенциальным инструментом для списывания, его присутствие в кадре запрещено, это также относится к смене активного окна. Если голова студента на протяжении сравнительно долгого времени повёрнута от монитора, считается, что студент использует избыточные источники информации.

Итак, требования к пользователю системы:

1. Студент должен находиться перед монитором своего компьютера.
2. Для обеспечения наибольшей вероятности верного распознавания, лицо студента должно быть удовлетворительно освещено.
3. Голова студента должна быть направлена в монитор, так как вне пределов монитора может находиться избыточная информация.
4. Студент должен не использовать мобильный телефон, так как телефон является источником избыточной информации.
5. Студент должен не менять активное окно экзамена, так как другие активные окна могут являться источниками избыточной информации.
6. Камера студента должна располагаться таким образом, чтобы система могла корректно работать.

## 2.2 Выработка дизайна экзамена

Так, как участниками экзамена являются студент и преподаватель, дизайн описан с их сторон.

### 2.2.1 Дизайн экзамена со стороны студента

Перед экзаменом, студенту необходимо зарегистрироваться в системе, предоставить свои данные: фотографию лица с приемлемым освещением, логин, пароль и номер телефона.

На этапе экзамена студент входит в систему и вводит свои регистрационные данные – логин и пароль. Открывается панель с полем для ввода кода, который присылает преподаватель для приглашения в комнату для сдачи экзамена. После того, как преподаватель начнет экзамен и запустит систему, студент начинает сдавать экзамен. По окончании экзамена, студент выходит из комнаты и на этом работа системы оканчивается.

### 2.2.2 Дизайн экзамена со стороны проктора

Перед началом экзаменационной сессии, преподавателю необходимо зарегистрироваться в системе. При регистрации преподаватель предоставляет свои регистрационные данные – логин, пароль и номер телефона.

Для начала экзаменационной сессии, преподавателю необходимо создать комнату, при её создании генерируется ключ комнаты, который преподаватель отправляет студентам для сдачи экзамена. Когда студенты зайдут в комнату, преподаватель начинает экзамен и запускает систему прокторинга. При возможном списывании, преподавателю приходит уведомление о списывании с подтверждением в качестве видео и отчётами системы. Преподаватель имеет возможность прислать уведомление студенту о предупреждении или об окончании экзамена для этого студента. Сам экзамен закончен не будет, предполагается, что преподаватель имеет возможность закончить его сам.

После окончания экзамена, комната закрывается и сохраняется в журнале с возможностью просматривать отчёты и видео-подтверждения о списывании для разрешения спорных вопросов.

## 2.3 Выработка архитектуры системы

Проектируемая система должна иметь несколько структурных разделов. Необходимо обеспечить регистрацию для студента и проктора, хранение регистрационной информации, хранение информации из журнала, непрерывную аутентификацию и прокторинг студента на экзамене, а также возможность проктора для манипуляции журналом.

Взаимодействие студента с системой будет происходить в рамках десктопного приложения. Преподаватель с системой будет взаимодействовать через веб интерфейс, а хранение регистрационной информации и информации из журнала будет проводиться с помощью базы данных.

Для проектирования системы прокторинга использовалась клиент-серверная архитектура.

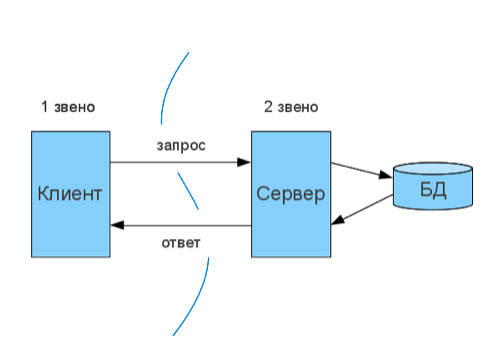
Клиент-серверная архитектура

Архитектура клиент-серверных приложений включает в себя пользовательские интерфейсы, базы данных и системы промежуточного программного обеспечения [8]. Такие приложения сочетают в себе серверные сценарии, которые отвечают за хранение данных, и клиентские, которые взаимодействуют с клиентами и предоставляют им данные. Для описанных приложений существуют различные виды шаблонов архитектур, их выбор зависит от требований пользователя и заказчика. Требования пользователя в основном заключаются в удобстве использования, время необходимое для обновления информации на странице, возможность переключаться между страницами и т.д. Требования заказчика сводятся к функциональным возможностям и безопасности информации, использованной в приложении.

Приложения делятся на два типа компонентов: интерфейс и структура. Компоненты пользовательского интерфейса обеспечивают отображение страницы, уведомления, журналы, разного рода взаимодействия пользователя с приложением. Структурные компоненты соединяют клиент, сервер приложения и сервер баз данных [9]. Задача клиента – взаимодействие с пользователем, сервер приложения отвечает за управление приложением, а сервер БД необходим для организации и хранения информации. Наглядный пример изображён на рисунке 5.

Итак, основные составные части системы прокторинга:

1. Десктопное приложение
   1. Система прокторинга
   2. Система аутентификации
   3. Система оценивания
2. Серверная часть
   1. Архитектура серверной части
   2. База данных
3. Клиентская часть

Рисунок 5. Работа клиент-серверной архитектуры.

### 3.3.1 Десктопное приложение

Для разработки десктопного приложения выбран высокоуровневый язык программирования Python версии 3.9.2. Такой выбор обусловлен с популярностью этого языка в среде машинного обучения, и как следствие – широким набором инструментов.

Для работы с Python была выбрана IDE Pycharm, поскольку Pycharm предоставляет широкий инструментарий работы с языком, удобный интерфейс, а также он поставляется бесплатно для студентов.

Библиотека OpenCV[10] была выбрана для данной работы, поскольку обеспечивает удовлетворительную поддержку десктопных устройств, имеет встроенный модуль для удобного инференса нейронных сетей из разных фреймворков, а также имеет набор предобученных нейронных сетей , некоторые из которых используются в данной работе. Также эта библиотека обеспечивает удобный набор инструментов для обработки изображений и видео, что в данном случае является необходимой операцией для работы с нейронными моделями. Ещё в работе используется библиотека scikit learn, которая позволяет удобно обучать классификаторы.

Так как для работы с приложением пользователю необходимо взаимодействовать с интерфейсом, он должен быть реализован. Для этой задачи в приложении используется расширение фреймворка Qt для языка Python – PyQt.

Результатом работы десктопного приложения является ответ от систем аутентификации и прокторинга. Если приложение определит, что в процессе экзамена студент пропал из кадра на продолжительное время, или если на его месте оказался другой человек, система аутентификации отправит предупреждение преподавателю о нарушении аутентификации. Также, если студент в процессе экзамена нарушит требования системы, преподаватель получит оповещение об этом. Система прокторинга и система аутентификации производят два независимых друг от друга результата работы, которые объединяются для создания глобального ответа системы.

### 3.3.2 Серверная часть

Так как задача серверной части заключается в эксплуатации базы данных, управлением работы клиентской части и взаимодействием с десктопным приложением, для её разработки используется язык программирования Python. Одним из основных его особенностей является простота синтаксиса, что позволяет сократить время разработки. Также для языка Python существует богатый выбор платформ для работы с веб-приложениями.

Для проектируемого веб-приложения используется платформа Django из-за её функциональных возможностей, а также удобства разработки.

Архитектура серверной части

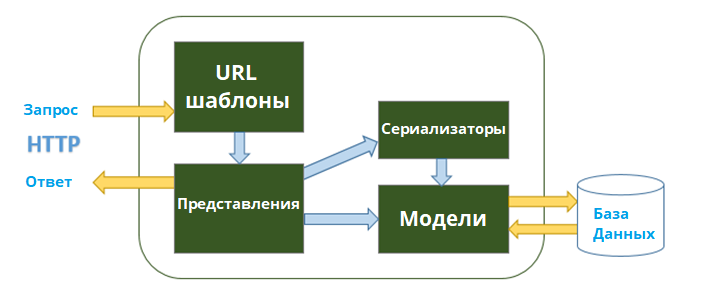
Архитектура веб-приложения соответствует архитектуре фреймворка Django Rest Framework.

Рисунок 6. Архитектура DRF.

В ходе работы с DRF основные взаимодействия происходят с четырьмя сущностями:

* Модели
* Сериализаторы
* Представления
* Проводники

Модели являются сущностью, позволяющей обращаться к таблицам БД на объектном уровне. Классы модели представляют из себя описание таблиц, а объекты – записи в таблице.

Задачей сериализаторов является преобразование типов данных для установления возможности взаимодействия веб-сервера посредством языка python с БД. Например, сериализаторы позволяют преобразовывать запросы к БД в родные для Python типы данных. Также обеспечивается десериализация – обратный сериализации процесс преобразования.

Представления сериализуют данные из моделей и отправляют на выход.

Проводники соединяют представления и URL-адреса.

База данных

На рисунке 7 представлена схема базы данных для проектируемой системы.

Таблицы Proctor и Student содержат всю основную информацию о прокторе и студенте соответственно: логин, пароль и номер телефона. Для студента также есть поле для хранения изображения его лица.

Таблица Violation содержит информацию об обнаруженном нарушении правил экзамена: тип нарушения (нарушение аутентификации или обнаружение списывания), путь к сохраненному видеофайлу на сервере с доказательствами нарушения, поле fatal сохраняет информацию о том было ли совершённое нарушение критическим (выгнан ли студент с экзамена), информацию о типе списывания (если ошибка типа прокторинг), информацию о комнате и студенте, совершившим нарушение.

Таблица Room представляет собой концепцию комнаты, в которой находятся студенты и преподаватель. Информация, содержащаяся в таблице: название комнаты, уникальный ключ, статус комнаты (создана/активна/в журнале), информация о дате создания комнаты и дате сохранения комнаты в журнал, а также прокторе, создавшем комнату.

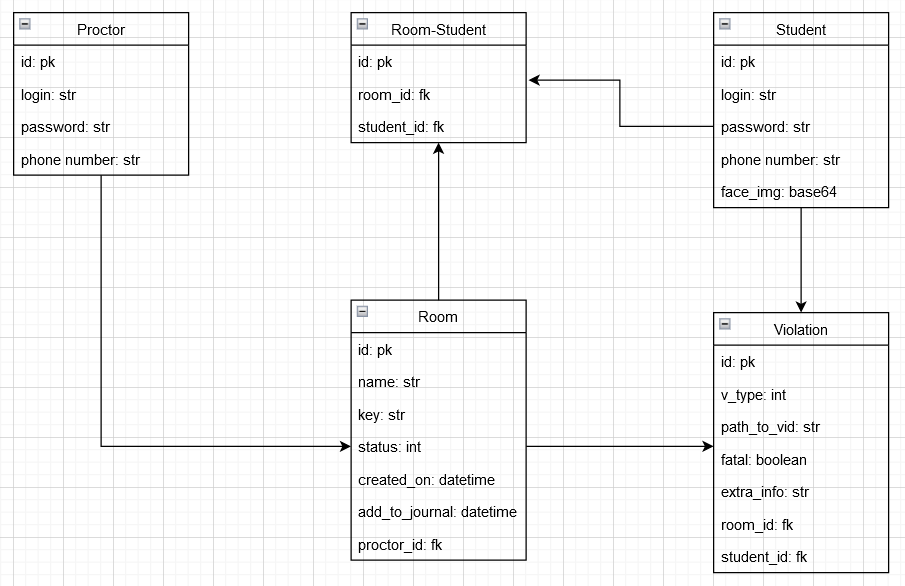
Таблица Room-Student представляет собой связь между таблицами Room и Student многие ко многим и хранит в себе информацию о связи между студентами и комнатами.

Рисунок 7.

### 3.3.3 Клиентская часть

Задача клиентской части веб-приложения – взаимодействие с пользователем, которым в данной системе является проктор. Для оптимизации задачи разработки клиентской части необходимо использовать фреймворк JavaScript. Таким фреймворком выступает Vue.js. Его выбор обусловлен качественной документацией, что позволяет сократить время на обучение, системой реактивности, которая позволяет представлению отображать изменённые в модели данные по мере их изменения, а также виртуальным DOM (document object model), который позволяет не перерисовывать всю страницу, а изменять в отображении только те данные, которые были изменены в модели.

# **Реализация**

## 3.1 Используемые в работе предобученные модели

В данной работе используется четыре предобученные модели, несколько фреймворков и библиотек компьютерного зрения и машинного обучения. Выбор моделей руководствуется требованиями к системе в приемлемой производительности и точности обнаружения, простоты применения и инференса.

### 3.1.1 Модель для отслеживания телефона в кадре

Так как отслеживание телефона является задачей обнаружения объекта в кадре, для такого типа задач используется класс моделей по обнаружению объектов. После определённой работы по поиску были выбраны следующие две модели обнаружения объектов: yolov7[11], yolov4-tiny[12], NanoDet[13].

Для выбора более подходящей модели было проведено тестирование на персональном компьютере со следующими характеристиками : intel core i3, Nvidia GeForce 940MX, 6 gb RAM.

Результаты тестирования показали наибольшую точность в случае использования yolov7 и наилучшую производительность в случае использования yolov4-tiny. По результатам тестирования была выбрана модель обнаружения объектов yolov4-tiny, поскольку совокупность производительности и достаточной точности подходит для использования её в реализуемой системе больше, чем остальные. Запуск моделей семейства yolo проводился средствами фреймворка DarkNet[14], а NanoDet с помощью библиотеки OpenCV и её модуля для инфреренса моделей глубокого обучения DNN.

Семейство моделей yolo широко распространено для решения задачи детекции объектов. Используемые предобученные модели обучены обнаруживать 80 классов объектов набора данных MS COCO[15], также поддерживается кастомное обучение. Особенностью этого семейства моделей является единоразовая загрузка изображения и проход его через свёрточную нейронную сеть, в других алгоритмах этот процесс происходит многократно.

Рисунок 8. Результат работы Nanodet

Рисунок 9. Результат работы yolov4-tiny

### 3.1.2 Модель для отслеживания поворота головы

Задача отслеживания поворота головы в машинном обучении не самая популярная, поэтому предобученных моделей для решения этой задачи существует не так много. В данном случае выбор пал на две модели – предобученная модель для обнаружения позы головы в парке обученных моделей фреймворка OpenVino[16] - head-pose-estimation-adas-0001(HPEA1)[17] и на модель для определения ключевых точек лица – FaceMesh[18] из библиотеки Mediapipe[19]. FaceMesh вычисляет трёхмерные координаты лица, что позволяет вычислить направление головы, если использовать как точку отсчёта метку, которая указывает на нос человека на предоставленном изображении.

HPEA1 принимает на вход обрезанное изображение лица человека, поэтому не является самостоятельной моделью, готовой к использованию, для её использования необходимо использовать модели обнаружения лиц, обрабатывать изображение, и только потом можно приступать к вычислению направления головы. Результат эта модель выдаёт относительно центра инерции по трём осям – крен, тангаж и рыскание.

FaceMesh является самостоятельной системой, позволяющей как обнаруживать лицо, так и метки на особо важных его точках. Это позволяет легче интегрировать эту модель в реализуемую систему. Для определения направления лица пользователя необходимо провести некоторые вычисления, так как модель выдаёт координаты меток на изображении, для этого используется функция solvePnP, которая оценивает позу объекта, учитывая набор его значимых элементов на плоскости изображения, соответствующие им проекции, а также матрицу камеры и коэффиценты искажения. Результаты конвертируются в углы поворота.

Для решения задачи выбора модели было так же как и в прошлом случае проведено тестирование на том же компьютере. Результаты тестирования показали высокую эффективность обеих моделей как по показателям точности, так и производительности. Это позволило выбрать модель, основываясь на простоте интеграции и удобстве использования результатов – FaceMesh.

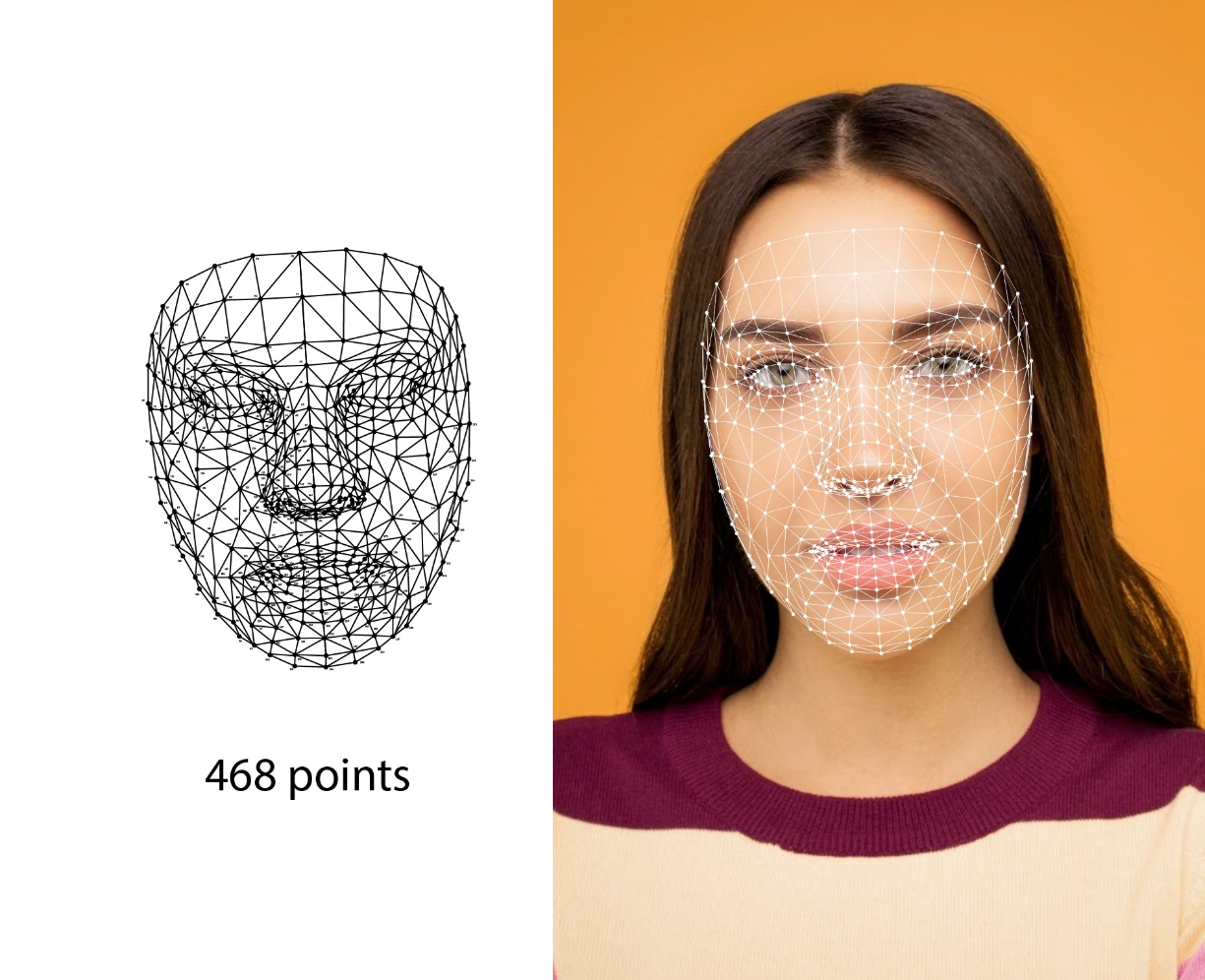
Рисунок 10. Результат работы HPEA1

Рисунок 11. Результат работы FaceMesh

### 3.1.3 Модель для обнаружения лиц

Обратная ситуация обстоит с классом моделей, предназначенных для обнаружения лиц. Задача очень распространённая и моделей её решения неисчислимое количество. Среди огромного множества моделей посредством сравнения характеристик было выбрано три следующие модели: Yunet[20] из парка предобученных моделей библиотеки компьютерного зрения OpenCV, face\_detection-retail\_004[21] и face-detection-adas-0001[22] из парка обученных моделей фреймворка OpenVino.

Тестирование показало, что результаты, полученные при применении модели Yunet наиболее точны, однако производительнее всего показала себя модель face\_detection\_retail\_004. Следуя выводам результатов тестирования и особенностям процесса применения моделей, наиболее эффективным выбором стала модель Yunet, кроме того, что она обнаруживает координаты для выделения всего лица, она также предоставляет результаты обнаружения меток наиболее значащих объектов на лице – нос, глаза, губы.

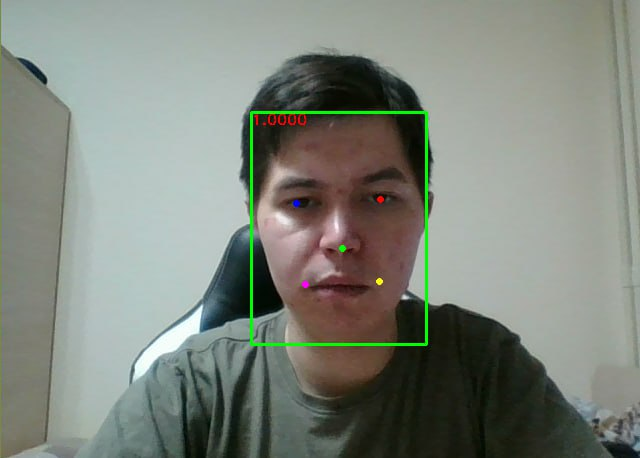
YuNet - легкая, быстрая и точная модель распознавания лиц. Основана на легкой MobileNet[23] архитектуре свёрточной нейронной сети, обучалась на наборе данных WIDER Face[24].

Рисунок 12. Результат работы Yunet

### 3.1.4 Модель для распознавания лиц

Для распознавания лиц было выбрано три следующих модели.

| **Название модели** | **Реализация** | **Точность** | **Производительность** |
| --- | --- | --- | --- |
| FaceNet | TensorFlow\* | 99.14% | 2.846 |
| LResNet100E-IR,ArcFace@ms1m-refine-v2 | MXNet\* | 99.68% | 24.2115 |
| SphereFace | Caffe\* | 98.8321% | 3.504 |

Таблица 1. Сравнение характеристик моделей распознавания лиц

Из представленной таблицы можно сделать вывод, что по характеристикам модели SphereFace[25] и FaceNet[26] не имеют серьёзных отличий, однако модель SphereFace имеет удобную реализацию в открытом парке моделей OpenCV, поэтому для работы была выбрана именно она.

## 3.2 Реализация десткопной части системы

Для разработки десктопной части использовались ранее описанные модели, фреймворки, библиотеки и инструменты.

Основной задачей десктопного приложения является обработка информации, поступающей во время экзамена, результат такой обработки представляет собой глобальный ответ о нарушении установленных правил проведения экзамена.

### 3.2.1 Класс проведения экзамена

Основным исполняемым файлом является “Containers.py”, в список его задач входит захват и запись видео, вызов обработчика видео “VideoProcessing.py”, выявление нарушений, запрос информации о сдающем, а также отправка информации о нарушениях на сервер.

Для работы с видео используется библиотека opencv, она позволяет захватывать изображение, записывать его, обрезать и проводить основные манипуляции с видео и изображениями.

После того, как программа принимает информацию о сдающем студенте, начинается цикл, продолжающийся весь экзамен. Информация о продолжительности экзамена поступает вместе с информацией о студенте. Далее создаётся видеофайл, название которого генерируется с помощью библиотеки uuid, позволяющей генерировать универсальный уникальный идентификатор, это нужно для того, чтобы имена созданных файлов не повторялись. Uuid представляет собой последовательность символов в формате 8-4-4-4-12 и его пример выглядит так: 9951ee9a-17c2-4261-8a31-4ec4087eeab0, соответственно имя файла будет следующим : '9951ee9a-17c2-4261-8a31-4ec4087eeab0.mp4'.

Затем, создаётся следующий цикл, отвечающий за запись видео в рамках временного окна, длительность которого также фиксирована. После этого видео считывается покадрово, каждый кадр проходит последующую обработку как отдельное изображение. Размерность считанного кадра изменяется на фиксированную величину. Это нужно для облегчения последующей обработки изображения, а также фиксация позволяет стандартизировать формат, так как исходные данные с разных камер могут различаться. Кроме того, этот шаг позволяет точно передавать, в качестве параметров, информацию о размерности для нейронных сетей, которым это требуется. После этого, кадр с изменённой размерностью записывается в созданный заранее файл. Если всё-таки изображение не удалось считать, информация об ошибке отправляется на сервер и в качестве уведомления показывается студенту.

Программа разветвлена на три потока: первый отвечает за запись и обработку видео, второй используется для фиксирования смены фокуса окон, а третий ведёт запись информации с мыши.

### 3.2.2 Класс обработки видео

Обработка видео происходит в файле “VideoProcessing.py”. Этот файл содержит класс с названием VideoProcessing. В файле “Containers.py” создаётся объект класса VideoProcessing и в качестве параметров передаётся информация о названии записанного видеофайла, разрешении, количестве кадров в секунду. Также передаваемые параметры содержат информацию, переданную с сервера, такую как: изображение лица и границы экрана пользователя. Вся эта информация записывается в поля объекта, это нужно для того, чтобы её можно было удобно использовать. После создания объекта, вызывается метод video\_processing\_start, в котором происходит основная работа программы по аутентификации пользователя и прокторингу.

В вызванном методе создаётся объект захвата видео и переменная, отвечающая за выбор кадра из временного окна для его обработки средствами аутентификации. Из временного окна для аутентификации используется только один кадр, на котором содержится лицо. Если лицо не будет обнаружено в течении трети времени от всего временного окна, будет сформировано и отправлено оповещение о нарушении.

В текущем методе создаётся цикл, отвечающий за обработку считанного покадрово видео. При условии успешного считывания система ищет на изображении лицо, если лицо найдено и рандомизированная переменная совпадает с номером кадра, запускается алгоритм проверки пользователя по лицу,для этого используется модель машинного обучения под названием Sface. Далее создаётся несколько потоков, каждый из потоков отвечает за обработку видео своей моделью машинного обучения. Всего потоков два, первый узнаёт направление головы с помощью модели Face Mesh и второй следит за появлением мобильного телефона в кадре, задействуя модель yolov4-tiny.

### 3.2.3 Аутентификация по лицу

Итак, в соответствующем методе создаётся объект класса FaceDetection и передаёт в конструктор размерность обрабатываемого кадра. После чего, вызывается метод этого класса, который в свою очередь вызывает встроенные методы модели для обнаружения лица в кадре. Эта модель построена на свёрточной нейронной сети и функция обнаружения возвращает пять меток лица – глаза, нос и две точки по краям рта, а также возвращает числовое выражение степени доверия к данной сессии оценки изображения.

После чего проверяется соответствие требованиям параметров рандомного числа и нахождения в кадре лица. Тогда в работу включается класс FaceRecognition, в нём описаны методы, используемые для создания условий работы алгоритма машинного обучения SFace. Для сравнения лиц из выбранного кадра и предоставленного при регистрации, система использует детектор лиц Yunet. Координаты ориентиров лица передаются методу preprocess, который, используя встроенные методы модели, выравнивает и обрезает кадр по координатам лица для улучшения результатов сравнения. Из результата предобработки создаётся вектор характеристик лица на изображении с помощью встроенного метода feature. После того, как из изображений лица с сервера и из текущего кадра извлечены вектора характеристик, они сравниваются методом косинусного коэффицента, если величина сравнения выше порогового уровня, лицо считается совпадающим.

### 3.2.4 Системы отслеживающие нарушения

Далее следует поток, запускающий функцию отслеживания позы головы. В функции создаётся объект класса HeadRotation, после чего вызывается метод этого класса rotation\_result. В основе этого метода лежит предобученная модель Face Mesh из библиотеки предобученных моделей фреймворка MediaPipe. Эта модель обнаруживает множество ориентиров лица в трёхмерном пространстве, включая пять основных. Благодаря этой особенности отслеживания трёхмерных ориентиров, эта модель успешно используется для проектирования вектора направления головы.

В текущей функции, переданный кадр проходит предобработку, так как для работы модели требуется изображение в цветовом формате RGB или КЗС (красный – зелёный – синий), однако библиотека opencv, методы которой используются для обработки изображений и видео, использует цветовое пространство BGR или СЗК (синий – зелёный – красный). После чего, средствами встроенных методов модели, из кадра извлекается множество ориентиров лица. По этим ориентирам вычисляются и нормализуются координаты основных ориентиров, однако для текущей задачи определения направления поворота головы, самым важным ориентиром является нос. Именно поэтому, его координаты вычисляются в трёхмерном и двухмерном пространствах, что вследствие позволит решить поставленную задачу. Полученные нормализованные координаты преобразуются в вектор вращения с использованием встроенного метода solvePnP из фреймворка opencv, после чего этот вектор преобразуется в матрицу вращения методом Rodrigues и последним этапом является получение углов вращения из полученной матрицы методом RQDecomp3x3. Полученные углы также нормализуются и могут использоваться для определения угла поворота головы. Эти углы сравниваются с полученными с сервера углами, которые были вычислены на этапе регистрации методом калибровки, и если углы больше, чем те, что получены с сервера, считается что в этом кадре студент направил свою голову вне экрана монитора.

Следующим методом, определяющим возможное нарушение условий проведения экзамена является обнаружение телефона в кадре. Для этой задачи используется предобученная модель yolov4-tiny. После вызова функции обработки изображения с целью поиска телефона в кадре, создаётся объект класса PhoneDetection. Объект инициализируется конструктором, в котором также происходит загрузка используемой модели средствами библиотеки opencv. После заргузки, модель принимает установленный параметр разрешения изображения. Инициирующий метод call вызывает методы open\_frame, в котором собственно происходит инференс модели и определяются координаты найденных на изображении объектов, и object\_definition, в котором определяется найден ли в кадре телефон. После этих операций результаты работы передаются обратно в исполняемый модуль.

Последним методом, отслеживающим нарушения является метод отслеживания активного окна. Этот метод использует библиотеки ctypes и pywin32, они предоставляют возможность работы с операционной системой напрямую. Смена окна регистрируется в журнале записи событий вместе с такими параметрами, как время события, тип события, id события и дерево события. По этим параметрам представляется возможным вычислить когда пользователь сменит активное окно и время, на которое он это сделает.

### 3.2.5 Аутентификация по динамике движения мыши

Так как система должна предоставлять комбинацию физиологических и поведенческих биометрий для аутентификации пользователя, в ней используется метод аутентификации по движению компьютерной мыши [27].

Для работы этой части системы аутентификации требуется прохождение следующих шагов:

1. Сбор данных с датчика, в данном случае это мышь.
2. Извлечение характеристик из собранных данных и преобразование в вектор характеристик.
3. Классификация вектора характеристик с помощью SVM и GA.

Данные, относящиеся к движению мыши, такие как: время, позиция курсора, и нажатие клавиш мыши, всё это считается необходимой для сбора информацией. После чего из собранных данных извлекаются характеристики и преобразовываются в вектор. Список извлекаемых характеристик:

1. Временная метка – время выполнения операции
2. Нажатие клавиши – определение нажатия клавиши. Если нажатие произошло – левое или правое.
3. Позиция – координаты выполнения операции.
4. Скорость перемещения – скорость движения, рассчитанная на основе позиции и временной метки.
5. Ускорение перемещения – ускорение движения, рассчитанное на основе скорости движения и временной метки.

Когда вектор признаков готов, он отправляется на сервер для его последующей классификации. Этим занимается классификатор SVM с оптимизацией алгоритмом GA.

Суть метода опорных векторов заключается в нахождении наибольшего расстояния между векторами признаков разных классов в гиперплоскости. Генетический алгоритм работает посредством случайного перебора, комбинирования и вариации искомых параметров, эквивалентно естественному отбору в природе, и используется для решения задачи оптимизации.

Результатом работы алгоритма является нормализованный параметр, отображающий степень схожести полученного в ходе экзамена вектора признаков и вектора признаков, который соответствует проверяемому пользователю, и хранится на сервере.

### 3.2.6 Определение нарушения

Так как системы прокторинга и аутентификации решают разные задачи, ответы этих подсистем также раздельны, однако позже объединяются в один глобальный ответ.

Для решения задачи аутентификации пользователя по разным биометрическим характеристикам используется система слияния на уровне баллов. Такая система позволяет решать задачу аутентификации раздельно, что в свою очередь открывает возможности к расширению системы другими биометрическими технологиями и добавлению новых поведенческих модальностей.

Суть слияния на уровне баллов заключается в том, что каждая подсистема, включающая в себя модель для аутентификации, раздельно определяет ответ аутентификации, после чего этот результат сопоставляется с уровнем доверия конкретно к этой системе, после этого результат конвертируется в баллы. Это позволяет коррелировать эти параметры и извлекать эффективный ответ системы.

Проблемным становится вопрос о слиянии результатов двух моделей с разными типами биометрии, так как не интрузивные поведенческие модели в своём результате зависят от времени сбора информации, тогда как физиологическая биометрия позволяет получить ответ здесь и сейчас. Эта проблема решается следующим образом: физиологическая биометрическая модель работает в каждом временном окне, являясь единственным методом аутентификации на этот момент, однако, когда поведенческая модель соберёт достаточно информации для аутентификации, результаты их работы сливаются в один ответ аутентификации системы. Временные интервалы работы поведенческой модели включают несколько временных окон работы физиологической. Такое решение проблемы позволяет сохранять приемлемый уровень надёжности системы аутентификации в моменты, когда поведенческая модель собирает информацию, и устанавливать более надёжный глобальный ответ в моменты, когда результат работы аутентификации во временном окне сливается с результатом работы поведенческой модели.

Система прокторинга не обладает возможностью коррелировать оценки соответствия результата работы с шаблоном и временем нарушения, так как модели, отслеживающие списывание не имеют параметра надёжности работы модели, кроме модели, которая определяет телефон в кадре. Поэтому, для оценки нарушений во временном окне, их баллы напрямую зависят от времени, которое пользователь потратил на нарушение условий экзамена.

Для каждой такой модели, баллы рассчитываются отдельно и в последствии складываются, так что если нарушитель может потратить фиксированное количество времени на нарушение только по одному пункту правил, однако, если нарушение распознает несколько моделей, время, которое он безнаказанно может потратить, уменьшается в разы.

Из-за разности природы используемых модулей, нельзя утверждать, что они равнозначны при оценке списывания, так как, например, отслеживание активного окна не нуждается в оценке, так как результат работы этого модуля является гарантированно верным, эту гарантию обеспечивает логика работы программы. Остальные модули отвечают на вопрос о списывании с некоторой вероятностью достоверности. Таким образом, существует система оценки, которая коррелирует разные модули и оценивает списывание с учётом этой корреляции. Модуль отслеживания активного окна в системе этой корреляции не участвует, так как при его использовании, остальные компоненты системы прокторинга не должны быть задействованы. Это обусловлено тем, что для использования избыточной информации на мониторе компьютера, нарушать условия проведения экзамена по другим параметрам не обязательно.

Кроме того, для облегчения работы проктора, необходимо разделять корреляционные признаки на степени опасности и отправлять уведомление о списывании только в случае гарантированного списывания. Остальные случаи списывания могут быть задокументированы и отправлены в конце экзамена в качестве списка с параметрами, определяющими время списывания и модель, которая обнаружила списывание.

После вычисления, баллы аутентификации и прокторинга складываются и вычисляется глобальный ответ системы. Если количество баллов превышает фиксированную величину, видео, записанное в это временное окно, отправляется на сервер вместе с информацией, которая позволила системе считать, что пользователь в это временное окно нарушал условия экзамена. Таким образом, система позволяет проктору подтвердить или опровергнуть своё решение о списывании.

|  |  |  |  |
| --- | --- | --- | --- |
| Степени угрозы | Низкий | Средний | Высокий |
| Баллы | 20 | 35 | 47 |

Таблица 2. Степени угрозы и соответствующие им баллы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Прокторинг | Баллы за секунду | Низкая степень угрозы, сек | Средняя степень угрозы, сек | Высокая степень угрозы, сек |
| Обнаружение телефона (PD) | 3.5 | 5.7 | 10 | 13.4 |
| Поворот головы (HR) | 2.5 | 8 | 14 | 18.8 |
| Обнаружение лица (FD) | 2 | 10 | 17.5 | 23.5 |
| PD + HR | 7 | 2.8 | 5 | 6.7 |
| PD + FD | 5.6 | 3.6 | 6.25 | 8.4 |
| FD + HR | 6.75 | 3 | 5.2 | 7 |
| PD + FD + HR | 10 | 2 | 3.5 | 4.7 |
| Мониторинг вкладок | ̶ | 2 | 4 | 7 |

Таблица 3. Корреляции датчиков списывания, баллов за секунду и соответствия времени списывания степеням угроз

|  |  |  |
| --- | --- | --- |
| Аутентификация | До включения мыши в работу, баллы | После включения мыши в работу, баллы |
| По лицу | 47 | 30 |
| По мыши | 0 | 17 |

Таблица 4. Корреляции аутентификационных датчиков

# **4 Тестирование и сравнение**

## 4.1 Тестирование

Для определения эффективности реализуемой системы, было проведено тестирование. Тестирование проводилось на трёх ПК, основные характеристики которых описаны в таблице:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер компьютера | Видеокарта | Процессор | Оперативная память, гб | Операционная система |
| 1 | Nvidia GeForce GTX 1660 | Intel core i5 10400f | 16 | Windows 10 Pro |
| 2 | Nvidia GeForce GTX 1650 | Intel core i5 10400f | 8 | Windows 11 Home |
| 3 | Nvidia GeForce 940MX | Intel core i3 6006U | 6 | Windows 10 Enterprise |

Таблица 5. Характеристики тестируемых ПК

Количество тестов соответствует количеству тестируемых ПК. Тесты предназначены для определения эффективности работы системы прокторинга, что включает в себя как позитивный сценарий – без списывания, так и негативный сценарий, в котором пользователь пытается списать или перед экраном компьютера сидит другой пользователь. Поэтому пользователь под номером 3 намеренно пытается списать для демонстрации работы модуля прокторинга, а вместо пользователя номер 2, перед экраном сидит другой пользователь.

Кроме того, для определения эффективности работы системы необходимо определить насколько она ресурсозатратна, поскольку расходование большого количества ресурсов может помешать выполнять ПК другие задачи, связанные, например, с использованием, непосредственно, тестирования. Так как система состоит из разных модулей отслеживания списывания и проверки подлинности пользователя, и не все они зависят от обработки видео, говорить о конкретных параметрах производительности достаточно проблематично, именно поэтому производительность бинарно классифицируется на производительную и не производительную при характеристиках именно тестируемого ПК. Остальные колонки содержат в себе результирующие баллы, выставленные системой тестируемому пользователю.

Время тестирования – одно временное окно, равное 30 секундам. Такое время было выбрано, так как тестировать примерное время экзамена в час или более не имеет смысла, поскольку иллюстрировать результаты прокторинга на протяжении экзамена может одно временное окно, так как временное окно – составляющее времени всего экзамена.

Для тестового периода аутентификационная информация содержалась в памяти компьютеров, а не на сервере, кроме того, первоначальной аутентификации в системе не проводилось, только непрерывная, в ходе тестирования.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер тестируемого ПК | Производительность | Результат аутентификации | Результат прокторинга | Общий результат | Степень угрозы |
| 1 | Производительна | 0 | 15 | 15 | ̶ |
| 2 | Производительна | 47 | 6 | 53 | высокая |
| 3 | Не производительна | 0 | 44 | 44 | средняя |

Таблица 6. Результаты тестирования

Из результатов тестирования можно сделать вывод, о том, что система требовательна к ресурсам устройства, на котором запускается. Также становится ясно, что система работает корректно даже на низкопроизводительных компьютерах, однако требуется существенная оптимизация. Для использования результатов обучения классификатора мыши необходима имитация экзамена с продолжительным периодом тестирования, что требует более существенного подхода. Кроме того, для тестирования классификатора необходимо имитировать сдачу экзамена, что требует от тестирующего доступ к реальному онлайн экзамену, поскольку поведенческая биометрия требует процесса имитации поведения тестируемого на экзамене.

В некоторых случаях система ошибочно выставляет баллы прокторинга за ситуации, в которых пользователь очевидно не нарушал правил, но такие случаи влияют на конечный результат незначительно, в остальном система работает корректно.

## 4.2 Сравнение реализуемой системы с аналогами

Сравнение систем прокторинга проводится по следующим параметрам: удобство использования (подразумевается наличие значительных полномочий у проктора), широта применения (наличие интеграции в конкретную систему обучения, либо модуля для интеграции в разные), наличие совокупности поведенческих и физиологических моделей для аутентификации(ПИФ) (этот параметр важен, так комплекс таких моделей позволяет говорить о наибольшей достоверности аутентификации), наличие проктора человека и степень его ответственности.

Также важным параметром при сравнении такого рода систем является производительность, однако возможности протестировать производительность таких систем нет, поскольку коммерческие системы закономерно платны, а реализации некоммерческих не публикуются.

Для сравнения будут приведены как коммерческие системы прокторинга (ProctorEdu, Examus), так и некоммерческие [4],[28], для удобства обращения 1 и 2 соответственно.

|  |  |  |  |
| --- | --- | --- | --- |
| Название системы | Статус коммерциализации | Удобство использования | Широта применения |
| ProctorEdu | + | + | - |
| Examus | + | + | - |
| 1 | - | + | - |
| 2 | - | - | - |
| Реализуемая система | - | - | + |

Таблица 7. Сравнение по признакам удобства и широты использования

Основная часть прокторинговых систем интегрируются с платформами обучения, что значительно ограничивает области применения такой системы, однако такая интеграция позволяет проктору получать необходимые полномочия управления экзаменом непосредственно через интерфейс программы прокторинга. Реализация модуля интеграции зависит от приоритетов при создании такой системы. В данном случае приоритет отдаётся широте применения, нежели удобству использования.

|  |  |  |
| --- | --- | --- |
| Название системы | Совокупность ПИФ | Наличие проктора человека |
| ProctorEdu | + | +- |
| Examus | - | +- |
| 1 | - | +- |
| 2 | - | + |
| Реализуемая система | + | +- |

Таблица 8. Сравнение по призакам наличия ПИФ и наличия проктора

В последней таблице приведено сравнение систем прокторинга по признаку наличия проктора человека, однако в некоторых ячейках обозначены неоднозначные данные. Если рассматривать подробнее, то коммерческие решения предоставляют возможность использования как проктора человека, так и алгоритмического проктора, что позволяет отметить это как достоинство системы, также это относится к работе [28], обозначенной в таблице как «1». В реализуемой в данной работе системе, такого выбора не предоставляется, но значение отмечено такое же как и у коммерческих систем, поскольку в реализуемой системе основную часть работы выполняет алгоритмический проктор, но для полной достоверности и получения полномочий для реагирования на недобросовестные действия со стороны студентов, используется проктор человек, что позволяет сказать о наиболее точном определении нарушения без использования избыточных ресурсов человека.

По результатам сравнения, можно сказать, что реализуемая система конкурентноспособна, поскольку абсолютно не зависит от среды обучения и вариантов для её применения бесчисленное количество, в отличие от систем с которыми она сравнивалась, кроме того наличие совокупности поведенческих и физиологических биометрий для аутентификации встречается достаточно редко как в коммерческом поле, так и некоммерческом. Это говорит даже о возможном коммерческом успехе.

# **ЗАКЛЮЧЕНИЕ**

В рамках выпускной квалификационной работы был проведён обзор и анализ литературы в сфере построения систем прокторинга с использованием средств непрерывной аутентификации, также были выработаны требования к реализуемой системе и к сдающему для определения правил проведения экзамена и границ нарушения условий его проведения.

В ходе работы была спроектирована, создана и протестирована система прокторинга. Выводы из проведённого анализа результатов тестирования и сравнения реализуемой системы с аналогами, говорит о её эффективности и конкурентноспособности.

# **СПИСОК СОКРАЩЕНИЙ**

HPEA1 – head pose estimation adas 001, нейронная сеть оценки позы головы, основанная на архитектуре свёрточной нейронной сети;

PD – phone detection, модуль отслеживания телефона в кадре;

HR – head rotation, модуль отслеживания поворота головы;

FD – face detection, модуль обнаружения лиц в кадре;

ПИФ – поведенческая и физиологическая (совокупность моделей).

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Pisani, P.H., Mhenni, A., Giot, R., (...), Rosenberger, C., Amara, N.E.B. Adaptive biometric systems: Review and perspectives // ACM Computing Surveys – 2019 – 102.
2. Damer, N., Maul, F., Busch, C. Multi-biometric continuous authentication: A trust model for an asynchronous system // FUSION 2016 - 19th International Conference on Information Fusion, Proceedings – 2016 – 7528154. P. 2192 – 2199.
3. Maniar, S., Sukhani, K., Shah, K., Dhage, S. Automated Proctoring System using Computer Vision Techniques // 2021 International Conference on System, Computation, Automation and Networking – 2021. P. 1 – 4.
4. Sapre, S., Shinde, K., Shetta, K., Badgujar, V. AI-ML Based Smart Online Examination Framework // Lecture Notes in Networks and Systems – 2022 – 441. P. 17-25.
5. Gopane S., Kotecha R., Enhancing Monitoring in Online Exams Using Artificial Intelligence// Lecture Notes in Networks and Systems – 2021 – 287. P. 183 – 193.
6. Atoum, Y., Chen, L., Liu, A.X., Hsu, S.D.H., Liu, X. Automated Online Exam Proctoring // IEEE Transactions on Multimedia – 2017 – 19(7) P. 1609 – 1624.
7. Fenu, G., Marras, M., Boratto, L., A multi-biometric system for continuous student authentication in e-learning platforms// Pattern Recognition Letters – 2018 – 113. P. 83 – 92.
8. Oluwatosin, H. S. Client-Server Model / H. S. Oluwatosin // IOSR Journal of Computer Engineering. – 2014. – Т. 16, № 1. – С. 67-71
9. Bayeh, E. The WebSphere Application Server architecture and programming model / E. Bayeh // IBM Systems Journal. – 1998. – Т. 37, № 3. – С. 336-348.
10. Сайт с документацией библиотеки OpenCV [Электронный ресурс]. – URL: https://opencv.org/ (дата обращения 08.12.2022)
11. Github репозиторий с предобученной моделью yolov7 а также сопроводительной документацией и элементами управления [Электронный ресурс]. – URL: https://github.com/WongKinYiu/yolov7 (дата обращения 08.12.2022)
12. Github репозиторий с предобученной моделью yolov4 а также сопроводительной документацией и элементами управления [Электронный ресурс]. – URL: https://github.com/david8862/keras-YOLOv3-model-set (дата обращения 08.12.2022)
13. Github репозиторий с предобученной моделью NanoDet а также сопроводительной документацией и элементами управления [Электронный ресурс]. – URL: https://github.com/opencv/opencv\_zoo/tree/master/models/object\_detection\_nanodet (дата обращения 08.12.2022)
14. Github репозиторий с сопроводительной документацией и элементами управления [Электронный ресурс]. – URL: https://github.com/pjreddie/darknet (дата обращения 08.12.2022)
15. Сайт датасета MS COCO [Электронный ресурс]. – URL: https://cocodataset.org/ (дата обращения 08.12.2022)
16. Сайт с документацией фреймворка OpenVino [Электронный ресурс]. – URL: https://docs.openvino.ai/latest/home.html (дата обращения 08.12.2022)
17. Github репозиторий с предобученной моделью head pose estimation adas 001 а также сопроводительной документацией и элементами управления [Электронный ресурс]. – URL: https://github.com/openvinotoolkit/open\_model\_zoo/tree/master/models/intel/head-pose-estimation-adas-0001 (дата обращения 08.12.2022)
18. Страница документации Face Mesh[Электронный ресурс]. – URL: https://google.github.io/mediapipe/solutions/face\_mesh (дата обращения 08.12.2022)
19. Сайт фреймворка MediaPipe [Электронный ресурс]. – URL: https://google.github.io/mediapipe/ (дата обращения 08.12.2022)
20. Github репозиторий с моделью для нахождения лица Yunet [Электронный ресурс]. – URL: https://github.com/opencv/opencv\_zoo/tree/master/models/face\_detection\_yunet (дата обращения 08.12.2022)
21. Github репозиторий с моделью для нахождения лица face detection retail 004 [Электронный ресурс]. – URL: https://github.com/openvinotoolkit/open\_model\_zoo/blob/master/models/intel/face-detection-retail-0004 (дата обращения 08.12.2022)
22. Github репозиторий с моделью для нахождения лица face detection adas 001 [Электронный ресурс]. – URL: https://github.com/openvinotoolkit/open\_model\_zoo/tree/master/models/intel/face-detection-adas-0001 (дата обращения 08.12.2022)
23. Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications // arXiv:1704.0486 – 2017.
24. Сайт датасета WIDER Face для обучения моделей [Электронный ресурс]. – URL: http://shuoyang1213.me/WIDERFACE/ (дата обращения 08.12.2022)
25. Github репозиторий с моделью для распознавания лица sface [Электронный ресурс]. – URL: https://github.com/opencv/opencv\_zoo/tree/master/models/face\_recognition\_sface (дата обращения 08.12.2022)
26. Github репозиторий с моделью для распознавания лица facenet [Электронный ресурс]. – URL: https://github.com/davidsandberg/facenet (дата обращения 08.12.2022)
27. Gao, L., Lian, Y., Yang, H., (...), Guo, S., Cheng, Y., Continuous Authentication of Mouse Dynamics Based on Decision Level Fusion // 2020 International Wireless Communications and Mobile Computing – 9148499. P. 210 – 214.
28. Aanjey Mani Tripathi, Rishabh Kasana, Rohit Bhandari,and Nirbhay Vashishtha., Online Examination System, 2021 Lecture Notes in Networks and Systems 286, P. 709 – 717.

# **Приложение А**

Листинг программного кода главного исполняемого файла Containers.

1. **import** cv2
2. **import** os,sys
3. **import** uuid
4. **from** threading **import** Thread
5. **import** time
7. **from** ExtractBiometric.VideoProcessing **import** VideoProcessing
8. **from** evaluation **import** Evaluation
9. **from** errors **import** Error
10. **from** server\_interaction **import** Violation,ServerData
11. **from** ExtractBiometric.Authentication.mouse **import** Mouse
12. **from** ExtractBiometric.Proctoring.WindowTracking **import** WindowFocusTrack
13. sys.path.append('TempStorage')
14. flag **=** 0
15. tid **=** 0
17. **def** cap\_n\_write(record\_time,window\_track):

20. get\_data **=** ServerData()
22. image,DIMENSION,SLIDE\_TIME\_WINDOW,FPS, screen\_borders**=** get\_data.get()
23. img\_srv **=** cv2.imread(image)
25. global\_frame\_counter **=** 0
26. people\_counter **=** 0
27. mouse **=** Mouse()
28. error **=** Error()


32. capture\_video\_stream **=** cv2.VideoCapture(0)
34. **while** global\_frame\_counter !**=** FPS **\*** record\_time:

37. filename **=** 'TempStorage/' **+** str(uuid.uuid4()) **+** '.mp4'
39. output\_video\_file **=** cv2.VideoWriter(filename, cv2.VideoWriter\_fourcc(**\***'XVID'),
40. FPS, DIMENSION)
41. local\_frame\_counter **=** 0
43. **while** local\_frame\_counter !**=** FPS **\*** SLIDE\_TIME\_WINDOW:
45. boolean\_result, frame **=** capture\_video\_stream.read()
46. **if** boolean\_result:
47. resized\_frame **=** cv2.resize(frame, DIMENSION, interpolation**=**cv2.INTER\_AREA)
48. output\_video\_file.write(resized\_frame)
49. local\_frame\_counter **+=** 1
50. **else**:
51. error.write\_video()
52. **break**
54. output\_video\_file.release()

57. result\_biometric\_processing **=** VideoProcessing(filename,DIMENSION,FPS,img\_srv,global\_frame\_counter,
58. screen\_borders)
59. result\_biometric\_processing.video\_processing\_start()
60. **if** result\_biometric\_processing.how\_many\_people\_in\_frame **==** 0:
61. people\_counter **+=**1
62. **if** people\_counter **==** (SLIDE\_TIME\_WINDOW **//** 3):
63. error.no\_people()
65. **if** mouse.collection\_check():
66. mouse\_result **=** mouse.result()
67. local\_evaluation(result\_biometric\_processing, window\_track.result\_tracking(), mouse\_result)
68. **else**:
69. local\_evaluation(result\_biometric\_processing, window\_track.result\_tracking())


73. global\_frame\_counter **+=** FPS **\*** SLIDE\_TIME\_WINDOW

76. **if** flag !**=** 0:
77. window\_track.track\_stop(tid)
78. **break**
79. capture\_video\_stream.release()
81. **def** inside\_start(rec\_time, name\_ex):
82. **global** tid
83. **global** mouse
84. window **=** WindowFocusTrack()
86. th1 **=** Thread(target**=**cap\_n\_write,args**=**[rec\_time,window])
87. th2 **=** Thread(target**=**window.track\_start,args**=**name\_ex)
88. th3 **=** Thread(target**=**mouse\_tracking,args**=**mouse)
90. thread\_pool **=** [th1,th2,th3]
91. **for** thread **in** thread\_pool:
92. thread.start()
93. tid **=** th2.native\_id
95. **def** exit\_exam():
96. **global** flag
97. flag **=** 1
99. **def** local\_evaluation(biometric\_result, window\_result, mouse\_result**=**None):
100. mark **=** Evaluation(biometric\_result)
101. **if** mouse\_result **is** **not** None:
102. mark.set\_mouse(mouse\_result)
103. send\_violation(mark.res\_corr)
105. **def** mouse\_tracking(mouse):
106. mouse.collection()

109. **def** send\_violation(res,window):
110. violation **=** Violation(res,window)
111. violation.send()

# **Приложение Б**

Листинг программного кода файла обработки видео VideoProcessing.

1. **import** cv2 as cv
2. **import** sys
3. **from** random **import** randint
4. **import** multiprocessing as mpc
6. **from** Authentication.FaceDetection **import** YuNet
7. **from** ..errors **import** Error
8. **from** Authentication.FaceRecognition **import** FaceRecognition
9. **from** Proctoring.HeadRotation **import** HeadRotation
10. **from** Proctoring.PhoneDetection **import** PhoneDetection
11. sys.path.append('../TempStorage' )


15. **class** VideoProcessing:
17. **def** \_\_init\_\_(self, filename, DIMENSION, FPS, img\_srv,frames, screen\_borders):
18. self.DIMENSION **=** DIMENSION
19. self.FPS **=** FPS
20. self.path\_to\_video **=** filename #  РїРµСЂРµРґР°С‚СЊ РїРµСЂРµРґР°С‡Сѓ
21. self.how\_many\_people\_in\_frame **=** 0
22. self.img\_srv **=** img\_srv
23. self.frame\_count **=** frames
24. self.screen\_borders **=** screen\_borders
25. self.window\_auth\_score **=** 0
26. self.window\_score **=** 0
27. self.threads **=** [[0],1]
28. self.error **=** Error()
29. cfg\_path **=** 'Proctoring/phone\_detection/yolov4-tiny.cfg'
30. weights\_path **=** 'Proctoring/phone\_detection/yolov4-tiny.weights'
31. classes **=** 'Proctoring/phone\_detection/classes.txt'
32. self.phone **=** PhoneDetection(cfg\_path,weights\_path,classes,self.DIMENSION)
33. self.head **=** HeadRotation(self.screen\_borders)
35. **def** video\_processing\_start(self):
36. tm **=** cv.TickMeter()
37. capture\_video **=** cv.VideoCapture(self.path\_to\_video)
38. when\_check\_face **=** randint(0, self.frame\_count)
39. frame\_counter **=** 0
40. **while** True:
41. capture\_result, frame **=** capture\_video.read()
42. tm.start()
43. **if** capture\_result:
44. frame **=** cv.resize(frame, self.DIMENSION, interpolation**=**cv.INTER\_AREA)
45. face\_det\_ret **=** self.face\_detection(frame)
46. **if** frame\_counter **==** when\_check\_face:
47. face\_recog\_ret, conf\_recog **=** self.face\_recognition(face\_det\_ret, frame,
48. self.img\_srv)  # 1 - yes. 0 - no
49. self.window\_auth\_score **=** face\_recog\_ret,conf\_recog
51. frame\_copy **=** frame.copy()
52. proc1 **=** mpc.Process(target**=**self.head\_rotation, args**=**frame)
53. proc1.start()
54. proc2 **=** mpc.Process(target**=**self.phone\_detector, args**=**frame\_copy)
55. proc2.start()
56. proc2.join()
57. tm.stop()
58. result **=** [tm.getFPS(),self.threads[0],self.threads[1],self.threads[2],self.window\_auth\_score]
59. frame\_counter **+=** 1
60. **else**:
61. self.error.write\_video()
63. capture\_video.release()
64. **return** result

67. **def** face\_detection(self, frame):
68. result\_detect **=** YuNet(modelPath**=**'Authentication/face\_detection\_yunet/face\_detection\_yunet\_2022mar.onnx',inputSize**=**self.DIMENSION).infer(frame)
69. self.how\_many\_people\_in\_frame **=** result\_detect.shape[0]
70. **return** result\_detect
72. **def** face\_recognition(self,result\_of\_detection,frame, img\_srv):
73. result\_recognition, conf **=** FaceRecognition(model\_path**=**'Authentication/face\_recognition\_sface/face\_recognition\_sface\_2021dec.onnx').rec\_match(frame,result\_of\_detection, img\_srv)
74. **return** result\_recognition, conf
76. **def** head\_rotation(self, frame):
77. self.head.set\_frame(frame)
78. result\_rotation **=** self.head.rotation\_result()
79. self.threads[1] **=** result\_rotation
81. **def** phone\_detector(self, frame):
82. self.phone.set\_frame(frame)
83. result\_phone **=** self.phone.call()
84. self.threads[2] **=** result\_phone

# **Приложение В**

Листинг программного кода файла детекции лица на изображении FaceDetection.

1. **from** itertools **import** product
3. **import** numpy as np
4. **import** cv2 as cv
6. **class** YuNet:
7. **def** \_\_init\_\_(self, modelPath, inputSize**=**[320, 320], confThreshold**=**0.6, nmsThreshold**=**0.3, topK**=**5000, backendId**=**0, targetId**=**0):
8. self.\_modelPath **=** modelPath
9. self.\_inputSize **=** tuple(inputSize) # [w, h]
10. self.\_confThreshold **=** confThreshold
11. self.\_nmsThreshold **=** nmsThreshold
12. self.\_topK **=** topK
13. self.\_backendId **=** backendId
14. self.\_targetId **=** targetId
16. self.\_model **=** cv.FaceDetectorYN.create(
17. model**=**self.\_modelPath,
18. config**=**"",
19. input\_size**=**self.\_inputSize,
20. score\_threshold**=**self.\_confThreshold,
21. nms\_threshold**=**self.\_nmsThreshold,
22. top\_k**=**self.\_topK,
23. backend\_id**=**self.\_backendId,
24. target\_id**=**self.\_targetId)
26. @property
27. **def** name(self):
28. **return** self.\_\_class\_\_.\_\_name\_\_
30. **def** setBackend(self, backendId):
31. self.\_backendId **=** backendId
32. self.\_model **=** cv.FaceDetectorYN.create(
33. model**=**self.\_modelPath,
34. config**=**"",
35. input\_size**=**self.\_inputSize,
36. score\_threshold**=**self.\_confThreshold,
37. nms\_threshold**=**self.\_nmsThreshold,
38. top\_k**=**self.\_topK,
39. backend\_id**=**self.\_backendId,
40. target\_id**=**self.\_targetId)
42. **def** setTarget(self, targetId):
43. self.\_targetId **=** targetId
44. self.\_model **=** cv.FaceDetectorYN.create(
45. model**=**self.\_modelPath,
46. config**=**"",
47. input\_size**=**self.\_inputSize,
48. score\_threshold**=**self.\_confThreshold,
49. nms\_threshold**=**self.\_nmsThreshold,
50. top\_k**=**self.\_topK,
51. backend\_id**=**self.\_backendId,
52. target\_id**=**self.\_targetId)
54. **def** setInputSize(self, input\_size):
55. self.\_model.setInputSize(tuple(input\_size))
57. **def** infer(self, image):
58. # Forward
59. faces **=** self.\_model.detect(image)
60. **return** faces[1]

# **Приложение Г**

Листинг программного кода файла распознавания лица на изображении FaceRecognition.

1. **import** cv2 as cv

4. **class** FaceRecognition:
6. **def** \_\_init\_\_(self, model\_path,disType**=**0, backend\_id**=**0, target\_id**=**0):
7. self.model\_path **=** model\_path
8. self.backend\_id **=** backend\_id
9. self.dis\_type **=** disType
10. self.target\_id **=** target\_id
11. self.rec\_model **=** cv.FaceRecognizerSF.create(
12. model**=**self.model\_path,
13. config**=**"",
14. backend\_id**=**self.backend\_id,
15. target\_id**=**self.target\_id)
17. self.\_threshold\_cosine **=** 0.363
19. **def** preprocess(self,frame,detect\_box\_face):
20. **if** detect\_box\_face **is** None:
21. **return** frame
22. **else**:
23. **return** self.rec\_model.alignCrop(frame, detect\_box\_face)
25. **def** vector\_extraction(self, frame, result\_of\_detection):
26. face\_aligned **=** self.preprocess(frame, result\_of\_detection)
27. face\_vector **=** self.rec\_model.feature(face\_aligned)
28. **return** face\_vector

31. **def** rec\_match(self,frame, result\_detect, img\_srv):
32. face1 **=** self.vector\_extraction(frame, result\_detect)
33. face2 **=** self.vector\_extraction(img\_srv, result\_detect)
35. # only cos dis type
37. **if** self.dis\_type **==** 0:
38. score\_cosin **=** self.rec\_model.match(face1[0][:**-**1], face2[0][:**-**1], self.dis\_type)
39. **return** 1 **if** score\_cosin >**=** self.\_threshold\_cosine **else** 0, score\_cosin

# **Приложение Д**

Листинг программного кода файла отслеживания поворота головы HeadRotation.

1. **import** cv2
2. **import** mediapipe as mp
3. **import** numpy as np
5. **class** HeadRotation:
6. **def** \_\_init\_\_(self, screen\_borders, frame**=**0):
7. self.screen\_borders **=** screen\_borders
8. self.face\_mash **=** mp.solutions.face\_mesh.FaceMesh(min\_detection\_confidence**=**0.5, min\_tracking\_confidence**=**0.5)
9. self.frame **=** frame
10. self.response **=** 0
12. **def** set\_frame(self, frame):
13. self.frame **=** frame
15. **def** rotation\_result(self):
16. image **=** self.frame
17. # Flip the image horizontally for a later selfie-view display
18. # Also convert the color space from BGR to RGB
19. image **=** cv2.cvtColor(cv2.flip(image, 1), cv2.COLOR\_BGR2RGB)
21. # To improve performance
22. image.flags.writeable **=** False
24. # Get the result
25. results **=** self.face\_mash.process(image)
27. # To improve performance
28. image.flags.writeable **=** True
30. # Convert the color space from RGB to BGR
31. image **=** cv2.cvtColor(image, cv2.COLOR\_RGB2BGR)
33. img\_h, img\_w, img\_c **=** image.shape
34. face\_3d **=** []
35. face\_2d **=** []
37. **if** results.multi\_face\_landmarks:
38. **for** face\_landmarks **in** results.multi\_face\_landmarks:
39. **for** idx, lm **in** enumerate(face\_landmarks.landmark):
40. **if** idx **==** 33 **or** idx **==** 263 **or** idx **==** 1 **or** idx **==** 61 **or** idx **==** 291 **or** idx **==** 199:
41. **if** idx **==** 1:
42. nose\_2d **=** (lm.x **\*** img\_w, lm.y **\*** img\_h)
43. nose\_3d **=** (lm.x **\*** img\_w, lm.y **\*** img\_h, lm.z **\*** 3000)
45. x, y **=** int(lm.x **\*** img\_w), int(lm.y **\*** img\_h)
47. # Get the 2D Coordinates
48. face\_2d.append([x, y])
50. # Get the 3D Coordinates
51. face\_3d.append([x, y, lm.z])
53. # Convert it to the NumPy array
54. face\_2d **=** np.array(face\_2d, dtype**=**np.float64)
56. # Convert it to the NumPy array
57. face\_3d **=** np.array(face\_3d, dtype**=**np.float64)
59. # The camera matrix
60. focal\_length **=** 1 **\*** img\_w
62. cam\_matrix **=** np.array([[focal\_length, 0, img\_h **/** 2],
63. [0, focal\_length, img\_w **/** 2],
64. [0, 0, 1]])
66. # The distortion parameters
67. dist\_matrix **=** np.zeros((4, 1), dtype**=**np.float64)
69. # Solve PnP
70. success, rot\_vec, trans\_vec **=** cv2.solvePnP(face\_3d, face\_2d, cam\_matrix, dist\_matrix)
72. # Get rotational matrix
73. rmat, jac **=** cv2.Rodrigues(rot\_vec)
75. # Get angles
76. angles, mtxR, mtxQ, Qx, Qy, Qz **=** cv2.RQDecomp3x3(rmat)
78. # Get the y rotation degree
79. x **=** angles[0] **\*** 360
80. y **=** angles[1] **\*** 360
81. z **=** angles[2] **\*** 360
83. # See where the user's head tilting
84. **if** y < self.screen\_borders[2]:
85. self.response **=** "Looking Left"
86. **elif** y > self.screen\_borders[3]:
87. self.response **=** "Looking Right"
88. **elif** x < self.screen\_borders[4]:
89. self.response **=** "Looking Down"
90. **elif** x > self.screen\_borders[5]:
91. self.response **=** "Looking Up"
92. **else**:
93. self.response **=** "Forward"
94. **return** self.response

# **Приложение Е**

Листинг программного кода файла отслеживания телефона в кадре PhoneDetection.

1. **import** cv2


5. **class** PhoneDetection:
7. **def** \_\_init\_\_(self, cfg,weights,classes\_path,dimension,frame**=**0):
9. self.frame **=** frame
10. self.DIMENSION **=** dimension
11. net **=** cv2.dnn.readNetFromDarknet(cfg, weights)
12. net.setPreferableBackend(cv2.dnn.DNN\_BACKEND\_CUDA)
13. net.setPreferableTarget(cv2.dnn.DNN\_TARGET\_CUDA\_FP16)
14. self.model **=** cv2.dnn.DetectionModel(net)
16. self.model.setInputParams(size**=**self.DIMENSION, scale**=**1 **/** 255, swapRB**=**True)
17. self.class\_names **=** []
18. with open(classes\_path, "r") as f:
19. self.class\_names **=** [cname.strip() **for** cname **in** f.readlines()]

22. **def** set\_frame(self, frame):
23. self.frame **=** frame
25. **def** open\_frame(self, frame):
26. label **=** []
27. frame **=** cv2.resize(frame, self.DIMENSION, interpolation**=**cv2.INTER\_AREA)
28. classes, scores, boxes **=** self.model.detect(frame, 0.7, 0.8)
29. **for** (classid, score, box) **in** zip(classes, scores, boxes):
30. label.append([self.class\_names[classid], score])
31. **return** label
33. **def** object\_definition(self, results):
34. labels, boxes **=** results
35. **for** i **in** range(len(labels)):
36. **if** labels[i][0] **==** 'cell phone':
37. **return** labels[i][1]
38. **else**:
39. **continue**
40. **return** None
42. **def** call(self):
43. img **=** self.frame
44. results **=** self.open\_frame(img)
45. def\_res **=** self.object\_definition(results)
46. **return** def\_res

# **Приложение Ё**

Листинг программного кода файла классификатора SVM для мыши simp\_SVM.

1. **import** logging
3. **import** numpy as np
5. **from** baseestimator **import** BaseEstimator
6. **import** kernels
8. np.random.seed(9999)
10. **class** SVM(BaseEstimator):
11. **def** \_\_init\_\_(self, C**=**1.0, kernel**=**None, tol**=**1e**-**3, max\_iter**=**100):
12. self.C **=** C
13. self.tol **=** tol
14. self.max\_iter **=** max\_iter
16. **if** kernel **is** None:
17. self.kernel **=** Linear()
18. **else**:
19. self.kernel **=** kernel
21. self.b **=** 0
22. self.K **=** None
23. self.a **=** None
24. self.memory **=** []
25. self.multiclass **=** False
27. **def** fit(self, X, y**=**None):
28. self.X, self.y **=** self.set\_input(X, y)
30. classes **=** np.unique(self.y)
31. self.classes **=** classes
33. **if** classes.size **==** 2:
34. **return** self.\_fit\_two\_classes(X, y)
35. **else**:
36. self.multiclass **=** True
38. X, y **=** self.X, self.y
39. **for** i **in** range(len(classes)**-**1):
40. **for** j **in** range(i**+**1, len(classes)):
41. classifier **=** dict()
42. select\_indices **=** np.where(np.logical\_or(y **==** classes[i], y **==** classes[j]))
43. self.X **=** X[select\_indices]
44. self.y **=** y[select\_indices]
45. self.y[np.where(self.y **==** i)] **=** **-**1.0
46. self.y[np.where(self.y **==** j)] **=** 1.0
48. self.\_fit\_two\_classes(self.X, self.y)
50. classifier['sv\_idx'] **=** self.sv\_idx
51. classifier['a'] **=** self.a
52. classifier['b'] **=** self.b
53. classifier['y'] **=** self.y
54. classifier['X'] **=** self.X
55. classifier['class\_1'] **=** i
56. classifier['class\_2'] **=** j
58. self.b **=** 0
59. self.K **=** None
60. self.a **=** None
62. self.memory.append(classifier)

65. **def** \_fit\_two\_classes(self, X, y**=**None):
66. self.set\_input(X, y)
67. self.K **=** np.zeros((self.n\_samples, self.n\_samples))
69. **for** i **in** range(self.n\_samples):
70. self.K[:, i] **=** self.kernel(self.X, self.X[i, :])
71. self.a **=** np.zeros(self.n\_samples)
72. self.sv\_idx **=** np.arange(0, self.n\_samples)
74. **return** self.\_train()

77. **def** \_train(self):
78. iter **=** 0
80. **while** iter < self.max\_iter:
81. iter **+=** 1
82. a\_old **=** np.copy(self.a)
84. **for** j **in** range(self.n\_samples):
86. i **=** self.random\_index(j)
88. eta **=** 2.0 **\*** self.K[i, j] **-** self.K[i, i] **-** self.K[j, j]
89. **if** eta >**=** 0:
90. **continue**
91. L, H **=** self.\_find\_bounds(i, j)
93. e\_i, e\_j **=** self.\_error(i), self.\_error(j)
95. a\_i\_old, a\_j\_old **=** self.a[i], self.a[j]
97. self.a[j] **-=** (self.y[j] **\*** (e\_i **-** e\_j)) **/** eta
98. self.a[j] **=** self.\_clip(self.a[j], H, L)
100. self.a[i] **+=** self.y[i] **\*** self.y[j] **\*** (a\_j\_old **-** self.a[j])
102. b1 **=** (
103. self.b **-** e\_i **-** self.y[i] **\*** (self.a[i] **-** a\_i\_old) **\*** self.K[i, i]
104. **-** self.y[j] **\*** (self.a[j] **-** a\_j\_old) **\*** self.K[i, j]
105. )
106. b2 **=** (
107. self.b **-** e\_j **-** self.y[j] **\*** (self.a[j] **-** a\_j\_old) **\*** self.K[j, j]
108. **-** self.y[i] **\*** (self.a[i] **-** a\_i\_old) **\*** self.K[i, j]
109. )
110. **if** 0 < self.a[i] < self.C:
111. self.b **=** b1
112. **elif** 0 < self.a[j] < self.C:
113. self.b **=** b2
114. **else**:
115. self.b **=** 0.5 **\*** (b1 **+** b2)
117. diff **=** np.linalg.norm(self.a **-** a\_old)
118. **if** diff < self.tol:
119. **break**
120. logging.info("Convergence has reached after %s." **%** iter)
122. # Save support vectors index
123. self.sv\_idx **=** np.where(self.a > 0)[0]


127. **def** \_predict(self, X**=**None):
128. n **=** X.shape[0]
129. result **=** np.zeros(n)
130. **for** i **in** range(n):
132. **if** self.multiclass **==** True:
133. result[i] **=** self.\_predict\_multi(X[i, :])
134. **else**:
135. result[i] **=** np.sign(self.\_predict\_row(X[i, :]))
137. **return** result
139. **def** \_predict\_multi(self, X):
140. result\_class **=** np.zeros(len(self.memory))
141. result\_value **=** np.zeros(len(self.memory))
143. **for** i **in** range(len(self.memory)):
144. self.sv\_idx **=** self.memory[i]['sv\_idx']
145. self.a **=** self.memory[i]['a']
146. self.b **=** self.memory[i]['b']
147. self.y **=** self.memory[i]['y']
148. self.X **=** self.memory[i]['X']
149. result\_class[i] **=** np.sign(self.\_predict\_row(X))
150. result\_value[i] **=** self.\_predict\_row(X)
152. votes **=** self.\_count\_votes(result\_class, result\_value)
153. votes **=** list(votes.values())
154. decide\_class **=** np.argmax(votes)
156. **return** decide\_class
158. **def** \_count\_votes(self, result\_class, result\_value):
159. vote\_per\_class **=** dict()
160. **for** name **in** self.classes:
161. vote\_per\_class[name] **=** 0
163. **for** i **in** range(len(self.memory)):
164. **if** result\_class[i] **==** **-**1:
165. class\_name **=** self.memory[i]['class\_1']
166. **else**:
167. class\_name **=** self.memory[i]['class\_2']
168. vote\_per\_class[class\_name] **+=**1
170. **return** vote\_per\_class
172. **def** \_predict\_row(self, X):
173. k\_v **=** self.kernel(self.X[self.sv\_idx], X)
174. **return** np.dot((self.a[self.sv\_idx] **\*** self.y[self.sv\_idx]).T, k\_v.T) **+** self.b
176. **def** random\_index(self, k):
177. i **=** k
178. **while** i **==** k:
179. i **=** np.random.randint(0, self.n\_samples **-** 1)
180. **return** i
182. **def** \_find\_bounds(self, i, j):
183. **if** self.y[i] !**=** self.y[j]:
184. L **=** max(0, self.a[j] **-** self.a[i])
185. H **=** min(self.C, self.C **+** self.a[j] **-** self.a[i])
186. **else**:
187. L **=** max(0, self.a[i] **+** self.a[j] **-** self.C)
188. H **=** min(self.C, self.a[i] **+** self.a[j])
189. **return** L, H
191. **def** \_error(self, i):
192. **return** self.\_predict\_row(self.X[i]) **-** self.y[i]
194. **def** \_clip(self, alpha, H, L):
195. **if** alpha > H:
196. alpha **=** H
197. **if** alpha < L:
198. alpha **=** L
199. **Return**

# **Приложение Ж**

Листинг программного кода файла отслеживания смены активных окон windowlocker.

1. **import** win32con
2. **from** ...errors **import** Error
3. **import** sys
4. **import** ctypes
5. **import** ctypes.wintypes
7. user32 **=** ctypes.windll.user32
8. ole32 **=** ctypes.windll.ole32
9. kernel32 **=** ctypes.windll.kernel32
11. WinEventProcType **=** ctypes.WINFUNCTYPE(
12. None,
13. ctypes.wintypes.HANDLE,
14. ctypes.wintypes.DWORD,
15. ctypes.wintypes.HWND,
16. ctypes.wintypes.LONG,
17. ctypes.wintypes.LONG,
18. ctypes.wintypes.DWORD,
19. ctypes.wintypes.DWORD
20. )

23. eventTypes **=** {
24. win32con.EVENT\_SYSTEM\_FOREGROUND: "Foreground",
25. # win32con.EVENT\_OBJECT\_FOCUS: "Focus",
26. # win32con.EVENT\_OBJECT\_SHOW: "Show",
27. # win32con.EVENT\_SYSTEM\_DIALOGSTART: "Dialog",
28. # win32con.EVENT\_SYSTEM\_CAPTURESTART: "Capture",
29. # win32con.EVENT\_SYSTEM\_MINIMIZEEND: "UnMinimize"
30. }
32. processFlag **=** getattr(win32con, 'PROCESS\_QUERY\_LIMITED\_INFORMATION', win32con.PROCESS\_QUERY\_INFORMATION)
33. threadFlag **=** getattr(win32con, 'THREAD\_QUERY\_LIMITED\_INFORMATION', win32con.THREAD\_QUERY\_INFORMATION)

36. # store last event time for displaying time between events
37. lastTime **=** 0
38. WM\_QUIT **=** 0x0012


42. **class** LLISTEN:
43. **def** \_\_init\_\_(self):
44. self.stop\_flag **=** 0
45. self.collection **=** []
46. error **=** Error()
48. **def** set\_stop\_flag(self,tid):
49. self.stop\_flag **=** 1
50. self.PostThreadMessage(tid)
52. **def** logError(self,msg):
53. self.error.win\_track(msg, file**=**sys.stderr)

56. **def** getProcessID(self,dwEventThread, hwnd):
58. hThread **=** kernel32.OpenThread(threadFlag, 0, dwEventThread)
60. **if** hThread:
61. **try**:
62. processID **=** kernel32.GetProcessIdOfThread(hThread)
63. **if** **not** processID:
64. self.logError("Couldn't get process for thread %s: %s" **%** (hThread, ctypes.WinError()))
65. **finally**:
66. kernel32.CloseHandle(hThread)
68. **else**:
69. errors **=** ["No thread handle for %s: %s" **%** (dwEventThread, ctypes.WinError(),)]
71. **if** hwnd:
72. processID **=** ctypes.wintypes.DWORD()
73. threadID **=** user32.GetWindowThreadProcessId(hwnd, ctypes.byref(processID))
75. **if** threadID !**=** dwEventThread:
76. self.logError("Window thread != event thread? %s != %s" **%** (threadID, dwEventThread))
78. **if** processID:
79. processID **=** processID.value
80. **else**:
81. errors.append("GetWindowThreadProcessID(%s) didn't work either: %s" **%** (hwnd, ctypes.WinError()))
82. processID **=** None
83. **else**:
84. processID **=** None
86. **if** **not** processID:
87. **for** err **in** errors:
88. self.logError(err)
90. **return** processID

93. **def** getProcessFilename(self,processID):
94. hProcess **=** kernel32.OpenProcess(processFlag, 0, processID)
95. **if** **not** hProcess:
96. self.logError("OpenProcess(%s) failed: %s" **%** (processID, ctypes.WinError()))
97. **return** None
99. **try**:
100. filenameBufferSize **=** ctypes.wintypes.DWORD(4096)
101. filename **=** ctypes.create\_unicode\_buffer(filenameBufferSize.value)
102. kernel32.QueryFullProcessImageNameW(hProcess, 0, ctypes.byref(filename),
103. ctypes.byref(filenameBufferSize))
105. **return** filename.value
106. **finally**:
107. kernel32.CloseHandle(hProcess)

110. **def** callback(self,hWinEventHook, event, hwnd, idObject, idChild, dwEventThread, dwmsEventTime):
111. **global** lastTime
112. length **=** user32.GetWindowTextLengthW(hwnd)
113. title **=** ctypes.create\_unicode\_buffer(length **+** 1)
114. user32.GetWindowTextW(hwnd, title, length **+** 1)
116. processID **=** self.getProcessID(dwEventThread, hwnd)
117. **if** self.stop\_flag **==** 1:
118. self.PostThreadMessage(processID)
120. shortName **=** '?'
121. **if** processID:
122. filename **=** self.getProcessFilename(processID)
123. **if** filename:
124. shortName **=** '\\'.join(filename.rsplit('\\', 2)[**-**2:])
126. **if** hwnd:
127. hwnd **=** hex(hwnd)
128. **elif** idObject **==** win32con.OBJID\_CURSOR:
129. hwnd **=** '<Cursor>'

132. self.collection.append([float(dwmsEventTime **-** lastTime)**/**1000,shortName, title.value])
134. lastTime **=** dwmsEventTime

137. **def** setHook(self,WinEventProc, eventType):
138. **return** user32.SetWinEventHook(
139. eventType,
140. eventType,
141. 0,
142. WinEventProc,
143. 0,
144. 0,
145. win32con.WINEVENT\_OUTOFCONTEXT
146. )
147. **def** PostThreadMessage(self,tid):
148. user32.PostThreadMessageW(tid, WM\_QUIT, 0, 0)
150. **def** listen(self):
151. ole32.CoInitialize(0)
152. eventInfo **=** []
153. WinEventProc **=** WinEventProcType(self.callback)
154. user32.SetWinEventHook.restype **=** ctypes.wintypes.HANDLE
156. hookIDs **=** [self.setHook(WinEventProc, et) **for** et **in** eventTypes.keys()]
157. **if** **not** any(hookIDs):
159. sys.exit(1)
160. **if** self.stop\_flag **==** 1:
161. sys.exit()
163. msg **=** ctypes.wintypes.MSG()
165. **while** user32.GetMessageW(ctypes.byref(msg), 0, 0, 0) !**=** 0:
166. print(msg)
167. user32.TranslateMessageW(msg)
168. user32.DispatchMessageW(msg)


172. **for** hookID **in** hookIDs:
173. user32.UnhookWinEvent(hookID)
175. ole32.CoUninitialize()