ТЕХНИЧЕСКОЕ ОПИСАНИЕ ПРОГРАММНОГО ПРОДУКТА

1. Архитектура программного продукта

Продукт разработан полностью силами специалистов школы. В настоящее время запущена бета версия продукта с ограниченным функционалом. Проект развернут на специально собранном сервере с высокой вычислительной производительностью.

ЦПУ	AMD Ryzen 3800x
ОЗУ	DDR4 32Gb 3200 MHz
Видеокарта	NVIDIA RTX 3060
HDD	1 TB
SSD	WDC 240 Gb
Операционная система	Debian OS 12

Таблица 1. Технические характеристики сервера

Все компоненты архитектуры программного продукта развернуты на сервере с помощью технологии контейнеризации Docker. Каждый элемент представляет собой виртуальную машину, которая по внутренним протоколам обменивается информацией с другими элементами системы. Подробная схема архитектуры представлена в приложении к отчету.

Основные функции составных частей:

1) Nginx – вэб сервер предназначенный для безопасной обработки запросов пользователей и переадресации их ресурсы сервера. Также он обрабатывает отображение статических файлов в интерфейсе.
2) ReactJS – фреймворк интерфейсной части продукта. Реализован на языке JavaScript. Для разметки веб страниц интерфейса использован HTML5 и CSS.
3) ChartJS – библиотека с набором интерактивных графиков и диаграмм для визуализации данных в интерфейсе пользователя.
4) Django Rest – фреймворк логической части продукта. Обработка запросов пользователей и формирование запросов: к ядру обработки видеоконтента, для записи данных в базу, для формирования циклических задач и постановки задач в очередь.
5) Celery – инструмент для управления очередью задач и циклическими задачами.
6) Redis DB – база для хранения временных данных (в нашем случае хранение списка задач в очереди)
7) Postgres DB – основная СУБД для хранения статистики и данных пользователей.
8) AI core – ядро обработки видео с помощью нейросетей и моделей данных.

Задачи для дальнейшего развития в данном направлении:

1) Разработка ядра авторизации. В текущей бета версии авторизация пользователя не предусмотрена.
2) Оптимизация производительности и многопоточного режима работы.
3) Использование графических ядер для обработки видео.

2. Пользовательский интерфейс

Для продукта был разработан пользовательский веб интерфейс, который развернут с помощью фреймворка ReactJS. Экраны пользователя и основной функционал:

1) Экран загрузки видео и постановки в очередь для обработки. Загрузка предусмотрена только путем передачи файла с компьютера на сервер.

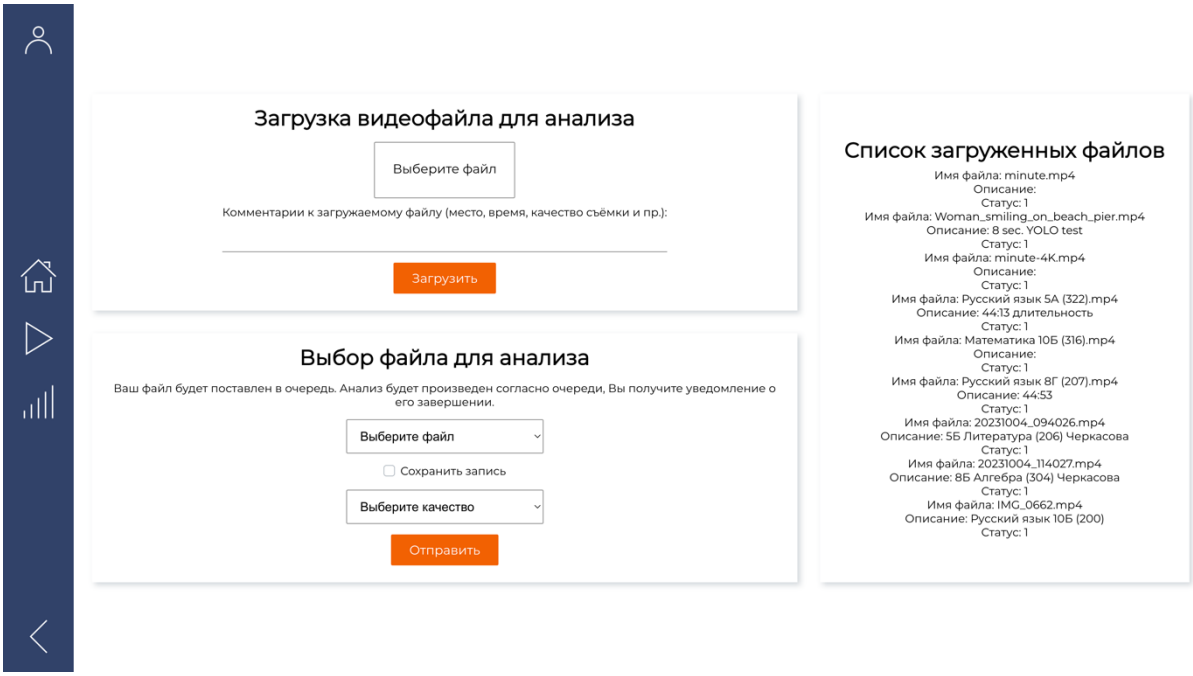

Рисунок 1. Экран загрузки видео

2) Экран отображения анализа видео. Слева отображается список обработанных видео, а справа аналитика в разных формах представления: график анализа эмоций от времени, столбчатая диаграмма, радарная диаграмма, круговая диаграмма. Все диаграммы показывают распределение эмоций на протяжении всего видео в процентном соотношении. График анализа эмоций от времени показывает распределение в процентном соотношении за каждую минуту видео.

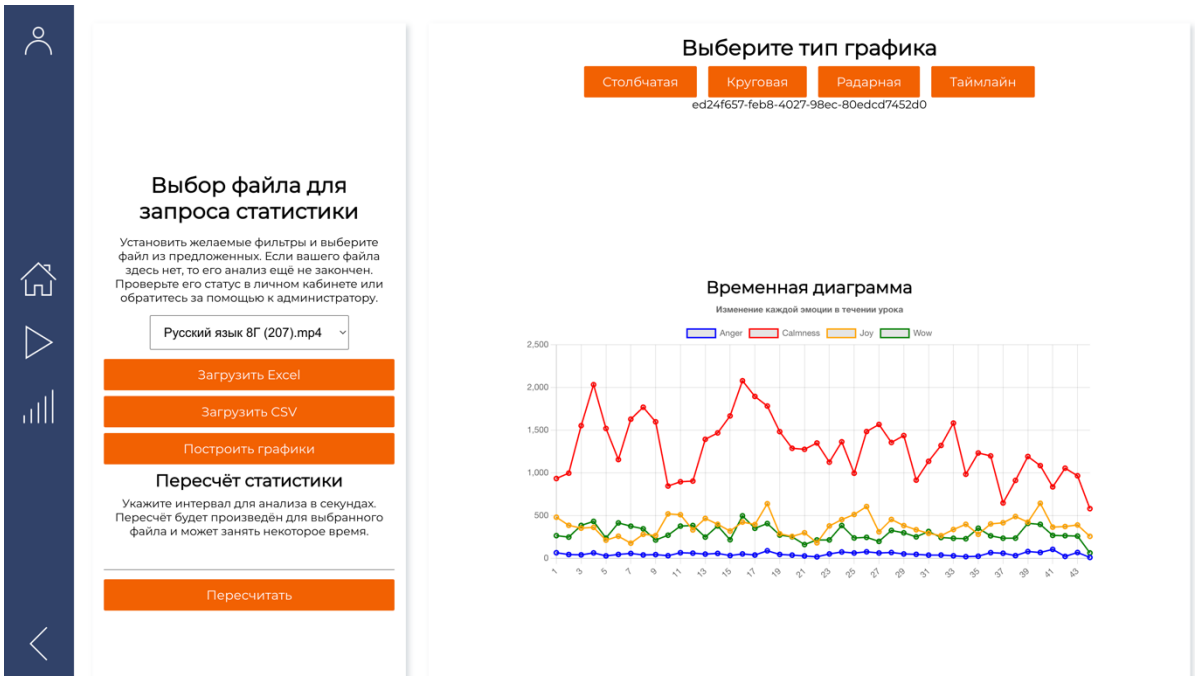

Рисунок 2. Экран отображения анализа видео

Задачи для дальнейшего развития в данном направлении:

1) Разработка формы авторизации
2) Разработка формы регистрации пользователя
3) Разработка формы регистрации организации
4) Оптимизация экранов пользователя

3. Описание алгоритмов работы нейросетей

В проекте используется не одна нейронная сеть, а их композиция, на данный момент состоящая из двух основных элементов. В будущем будут добавлены и другие нейронные сети для работы с новыми типами данных, поэтому архитектура проекта делается гибкой и настраиваемой.

Первый внедренный алгоритм выполняет задачу поиска человеческих лиц во всех кадрах загруженного видео. На данный момент для этого применяется библиотека Dlib, написанная на C++ и имеющая обширный набор вычислительных инструментов и инструментов для машинного обучения. Кроме того, важно, что она сочетает алгоритмы HOG и Linear SVM:

Histogram of Oriented Gradients (HOG) – это метод извлечения признаков для изображения, который используется для обнаружения объектов, в общем случае, и, как частный случай, лиц. Он основан на вычислении гистограмм градиентов в каждом локальном участке изображения.

Linear Support Vector Machine (SVM) – это метод машинного обучения, который используется для классификации объектов. Он основан на поиске гиперплоскости, которая разделяет объекты разных классов.

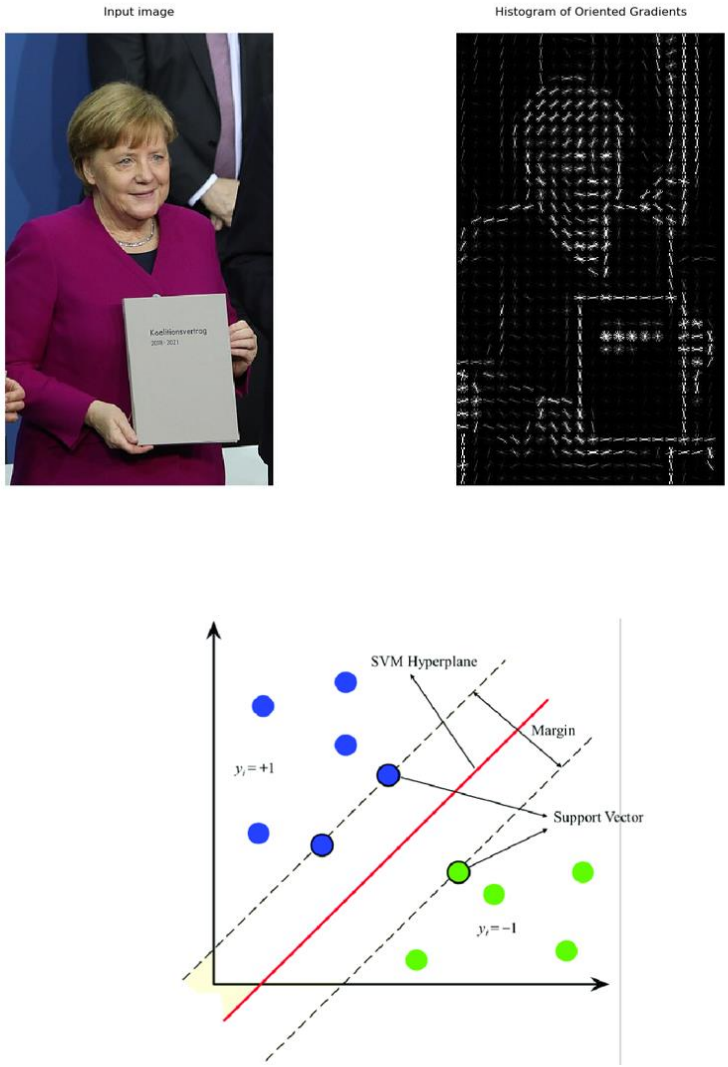

Рисунок 3. Принцип построения HOG-признаков и схема работы Linear SVM

В данном проекте объединение HOG и SVM используется для обнаружения лиц в изображениях. Сначала из изображения извлекаются HOG-признаки. Затем эти признаки анализирует SVM и присваивает участку изображения класс: лицо или не лицо. На выходе, алгоритм выдает координаты всех лиц, найденных в изображении.
Второй алгоритм непосредственно предназначен для определения эмоции на уже обнаруженном лице. Для этого нами была выбрана нейронная сеть типа CNN. **Convolutional Neural Network (CNN)** – это тип нейронной сети, которая хорошо подходит для обработки изображений. CNN состоят из нескольких слоев, каждый из которых выполняет определенную функцию.

Для определения эмоции лица CNN сначала выполняет операцию свертки для обнаружения локальных признаков лица, таких как глаза, губы и брови. Затем CNN выполняет операцию подвыборки для уменьшения размера изображения, сохраняя при этом важные признаки. Наконец, CNN использует полносвязный слой для классификации эмоции лица. На рисунке приведены условная схема сверточной нейронной сети.

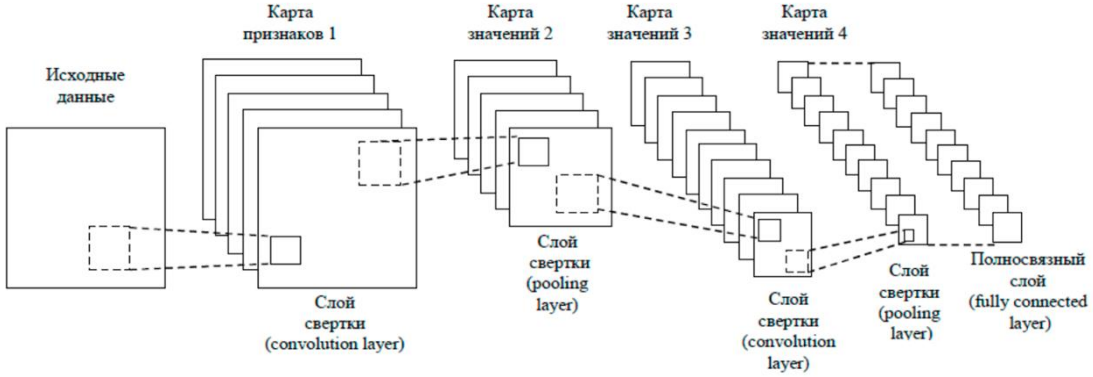

Рисунок 4 “Архитектура классической CNN”

Разработка собственной архитектуры нейросети – сложнейшая задача, требующая большое количество времени и вычислительные мощности. В нашем проекте мы остановились на следующей структуре:

```
self.conv1 = nn.Conv2d( in_channels: 1,  out_channels: 10,  kernel_size: 3)
self.conv2 = nn.Conv2d( in_channels: 10,  out_channels: 10,  kernel_size: 3)
self.pool2 = nn.MaxPool2d( kernel_size: 2,  stride: 2)

self.conv3 = nn.Conv2d( in_channels: 10,  out_channels: 10,  kernel_size: 3)
self.conv4 = nn.Conv2d( in_channels: 10,  out_channels: 10,  kernel_size: 3)
self.pool4 = nn.MaxPool2d( kernel_size: 2,  stride: 2)

self.norm = nn.BatchNorm2d(10)

self.fc1 = nn.Linear( in_features: 810,  out_features: 50)
self.fc2 = nn.Linear( in_features: 50, num_classes)
```

Важнейшая часть работы с нейросетью – это ее обучение. Для этого был составлен набор данных – **Dataset (DS)** из более чем 250 тысяч изображений человеческих лиц, выражающих 3 основные эмоции и отсутствие эмоции как таковой. DS фильтровался с помощью специальных утилит, кроме того, определенная его часть просмотрена и проверена вручную.

На текущий момент ведётся работа по увеличению точности работы описанных нейросетей. Промежуточный результат, которого удалось достичь:

- распознавание лиц (Dlib) – 99,38% (расчёт произведён при фронтальном расположении лица)

- распознавание эмоций (CNN) – 81% (показатель был получен при обучении CNN на DS из четырех эмоций)
- общая точность ~ 80%

Что касается производительности, стоит отметить, что используемые сервер не предназначен для выполнения операций машинного обучения. Для этого используются несколько другие комплектующие, заточенные именно под вычислительные алгоритмы. На текущий момент обработка 1 часа видео в формате FullHD (30fps) занимает 6,5 часов. Повышение точности и скорости распознавания являются приоритетными для нас в следующем году.