Всероссийский конкурс исследовательских и проектных работ
школьников «Высший пилотаж»

Проект “OpenEyes” по устранению дефекта закрытых глаз на групповых фотопортретах с помощью нейронной сети

Проект
Направление «Computer science»

Автор: Теплов Андрей,
учащийся 11Р класса,
ГАОУ Школа №548 «Царицыно», г. Москва

2023 г.

Аннотация

Цель проекта - найти и реализовать решение для устранения дефекта закрытых глаз на групповых фотопортретах.

Задачи:

- Определить значимость проблемы
- Провести анализ существующих инструментов для устранения дефекта
- Разработать собственное решение на основе нейронной сети
- Провести оценку полученного решения
- Реализовать управление решением через интерфейс чат-бота

Результат проекта - представлен в виде программного модуля, на базе нейронной сети, управляемого через пользовательский интерфейс чат-бота в Telegram. Полученное решение позволяет устранять дефект закрытых глаз на групповых фотографиях многократно быстрее и дешевле, чем существующие на рынке аналоги.

Обоснование актуальности

В среднем, люди хранят на своем смартфоне около 1000 фотографий[1]. Это памятные моменты с родственниками, коллегами, друзьями - важными для них людьми. Кроме того, групповые портреты практикуют профессиональные фотографы для съемки выпускных фотоальбомов, итогов важных симпозиумов, конференций, саммитов. Как правило, фотографы (профессиональные и любители) для съемки групповых портретов делают серию фотографий. Позже выбирается одна самая удачная по композиции и эмоциональному составу. Чем больше на групповой фотографии лиц тем выше вероятность, что кто-либо из присутствующих будет запечатлен не совсем удачно. Одной из самых распространенных причин браковки фотографий являются закрытые глаза у одного или нескольких участников группового портрета.

Для решения проблемы закрытых глаз, на текущий день, приходится использовать такие дорогие и сложные инструменты, как Photoshop и его аналоги. Процесс исправление долгий, скрупулезный, требует навыков владения инструментом и не позволяет делать подобные процедуры в большом объеме за короткое время.

Сегодня уровень развития искусственных нейронных сетей позволяет довольно точно детектировать требуемые участки на изображениях и вносить коррективы. Скорость и производительность обученных моделей позволяют выводить решения на промышленный поток и получать требуемый результат быстро.

Таким образом, возникла идея проекта по созданию программного обеспечения по автоматической коррекции дефекта закрытых глаз, при помощи нейронных сетей.

Цель и задачи

Цель проекта – найти и реализовать решение для устранения дефекта закрытых глаз на групповых фотопортретах.

Основная задача разделяется на две подзадачи:

- Поиск глаз на фото с помощью нейронной сети.
- Создание алгоритма замены глаз.

В качестве пользовательского интерфейса было принято решение использовать механику чат-бота на платформе Telegram.

Анализ существующих решений

1) Photoshop: ручное изменение фотографии (рисунок 1). Очевидные недостатки:

- Высокая стоимость программного продукта
- Высокий порог вхождения для решения задачи

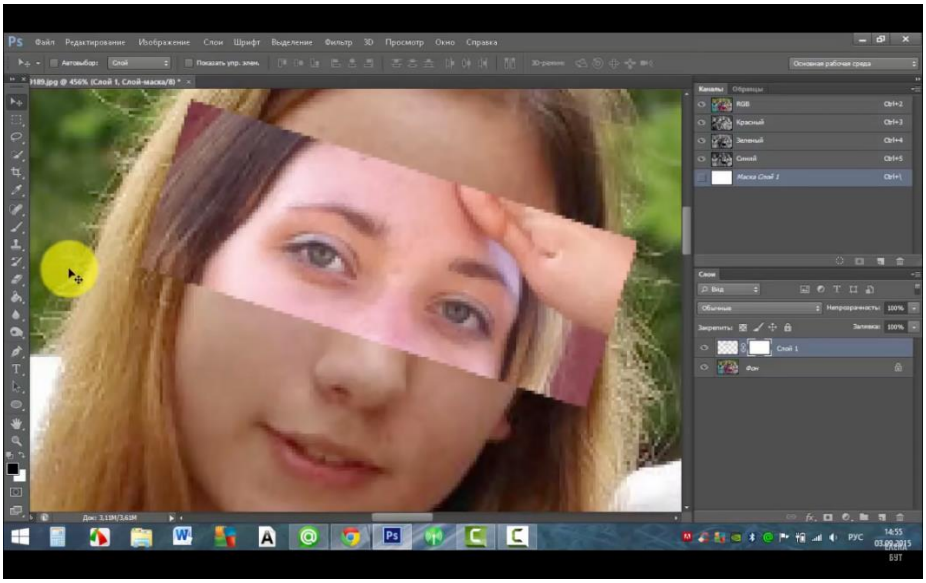

Рисунок 1

2) Meta: алгоритм основан на генерации глаз с помощью нейронных сетей GAN. В большинстве случаев новые сгенерированные глаза не подходят людям и являются синтезированными, т.е. чужими (рисунок 2)[9]. Также для обучения модели GAN нужен большой объем данных, с нужной разметкой и мощное компьютерное оборудования для быстрого обучения. Это является одним из минусов данного алгоритма.

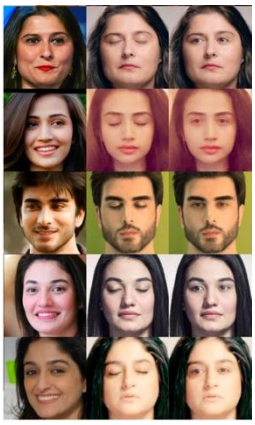

Рисунок 2.

Заметим, что аналогов не так много, и они не совсем точно решают поставленную задачу.

Дорожная карта

1. Постановка цели – 01.12.2022
2. Сбор данных – 15.12.2022
3. Разметка данных – 17.12.2022
4. Обучение модели – 25.12.2022
5. Создание алгоритма удаления дефекта закрытых глаз – 31.12.2022
6. Создание чат-бота на платформе Telegram – 10.01.2023

Описание разработанного решения

Для решения задачи устранения дефекта закрытых глаз существует несколько способов:

1. Генерация открытых глаз
2. Копирование участка открытых глаз из удачного источника на место дефекта

Проанализировав существующие решения, выбор пал на 2 пункт. Приведем пример проблемы: у нас есть два изображения (рисунок 2, рисунок 3) в которых два человека с закрытыми глазами. Рассмотрим на примере женщины в белой футболке. Для того, чтобы решить задачу, нужно найти открытые глаза и вырезать их, после вставить на другую фотографию, где расположен нужный дефект. Получаем готовый результат (рисунок 4).

Рисунок 2
Замазанные участок = закрытые глаза

Рисунок 3
Замазанные участок = закрытые глаза

Рисунок 4
Готовый вариант

Решение задачи устранения дефекта закрытых глаз разбивается на подзадачи, первая из которых поиск глаз на фотографии. Нейросети отлично справляются с этой задачей, поэтому было принято решение взять модель YOLOv5[5] (данная модель является одной из лучшей в своей категории (рисунок 8), а также очень эффективной для понимания и работы с ней) и дообучить ее на подходящем наборе данных. С поиском датасета возникли сложности, так как не удалось найти данные с нужной разметкой, а именно, групповые фотографии с выделенными областями глаз, в виде координат. Поэтому пришлось взять данные из публичного набора данных [6](пример фотографий на рисунке 5, 6, 7) и разметить их самостоятельно с помощью сервиса makesense.ai[7].

Рисунок 5

Рисунок 6

Рисунок 7

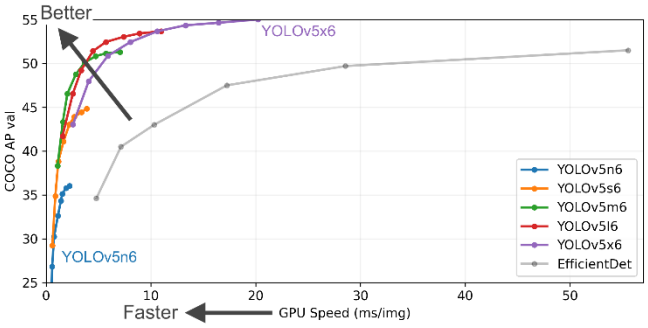

Рисунок 8

Для разметки глаз можно было действовать следующим путем:

1. Размечать глаза единым “прямоугольником - объектом”
2. Размечать глаза по-отдельности

Рисунок 11

Рисунок 12

Я выбрал первый вариант, так модели будет легче обучаться. Шанс, что модель определит несколько крупных объектов - выше, чем множество мелких. Кроме того, первый вариант проще для разметки.

Перейдем к модели. Как уже говорилось, я выбрал YOLOv5, предназначенную для детекции объектов на фото. YOLOv5 относится к архитектуре One-Stage detector - подход, который предсказывает координаты определённого количества bounding box'ов с результатами классификации и вероятности нахождения объекта, и в дальнейшем корректируя их местоположение. В целом, такую архитектуру можно представить в следующем виде:

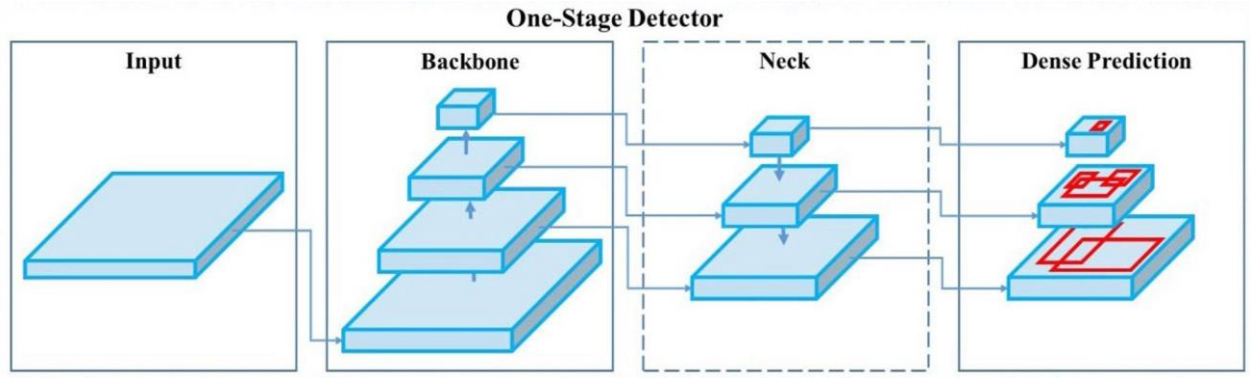

Рисунок 13

Сеть масштабирует исходное изображение в несколько feature map'ов с использованием skip-connection и прочих архитектурных особенностей. Полученные карты признаков приводятся в одно разрешение с помощью апсемплинга и конкатенируются. Затем, предсказываются классы и bounding box'ы для объектов. Далее, для каждого объекта, выбирается самый вероятный bounding box (далее bbox) с помощью Non-Maximum Suppression**.**
Для обучения я использовал обученные веса yolov5s (small version) на наборе данных COCO[10]. Чем больше модель, тем дольше она обучается, поэтому я выбрал small версию.

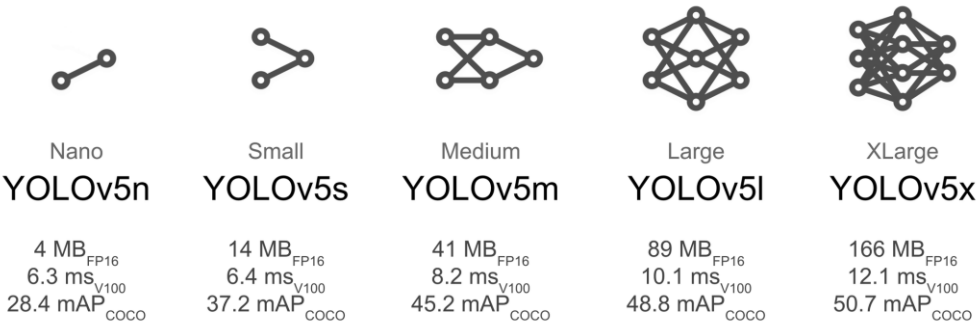

Рисунок 14

Модель обучалась на 368 фотографий с 10% разделением на тренировочную и валидационную выборку. Во время обучения на вход модели поступали изображения с разрешением 1280px. Для сокращения времени обучения я обучал нейронную сеть с помощью пакетов, которые включают в себя по 8 фотографий. Модель обучалась на 50 эпохах, что заняло ~26 минут.
Результат:

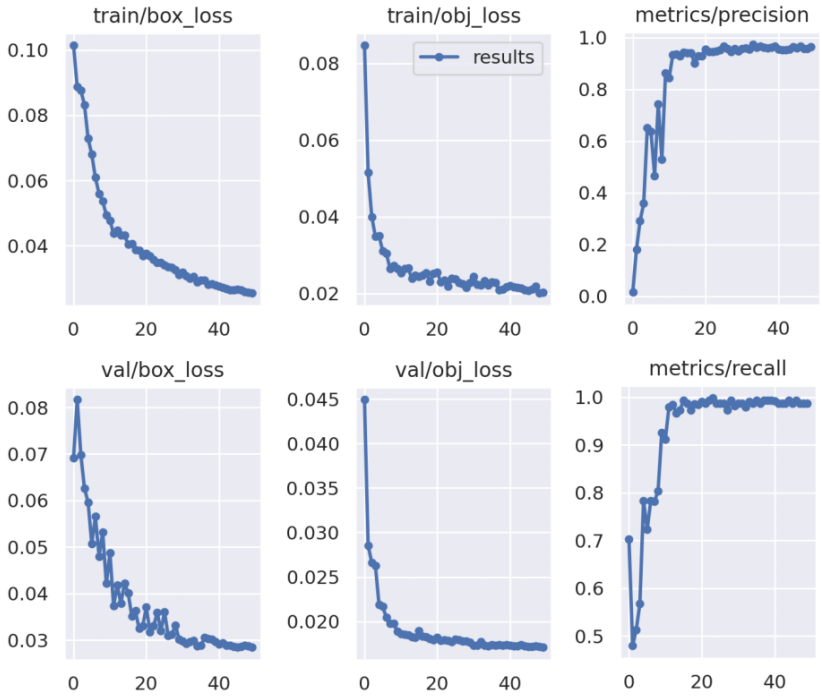

Рисунок 15. На оси абсцисс расположен номер эпохи. На оси ординат первых четырех графиках – значение функции потерь. На 5 и 6 графике – значение метрики Precision и Recall.

Метрика Precision (точность) и Recall (полнота) рассчитываются по данным формулам:

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

Исходя из графиков, можем заметить, что результат очень хороший. Переобучение не заметно. Проверим нашу модель на тестовых примерах.

Исходная фотография:

Рисунок 16

Предсказание модели:

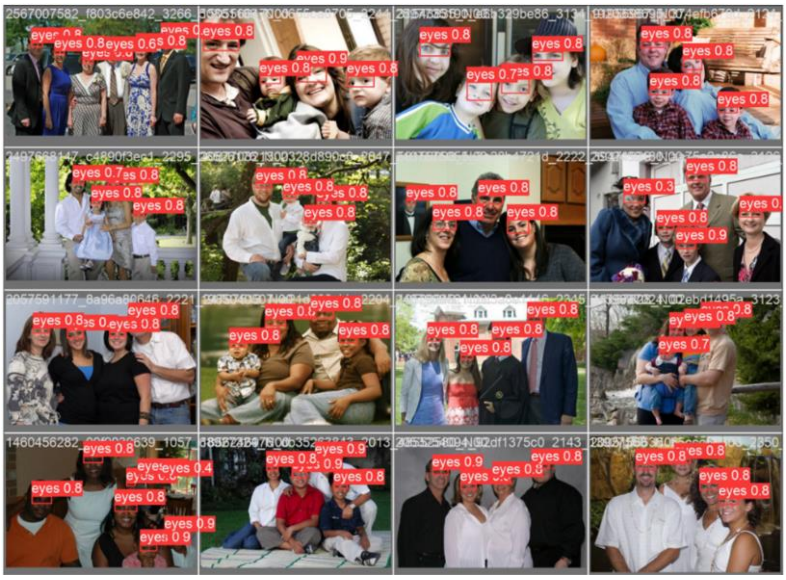

Рисунок 17

Проверим также модель на своей фотографии:

Рисунок 18

Рисунок 19

Можем заметить, что на моей фотографии качество оценки немного хуже, но тут и люди стоят дальше от камеры. При этом все объекты были распознаны точно, без ошибок.

Сделаем промежуточный вывод: Подобрали и разметили датасет с групповыми фотографиями людей, после обучили эти данные с помощью модели YOLOv5. Теперь мы можем находить точное расположение глаз на фотографии.

Вторая подзадача устранения дефекта закрытых глаз - это замена закрытых глаз на открытые. Для этого нам нужны две фотографии, где нужный человек с открытыми глазами и закрытыми. Подаем нашей уже обученной модели фото, и она возвращает нам координаты bbox

(расположение глаз). Далее вырезаем по этим координатам и получаем новое изображение, которое вставляем на поврежденный участок исправляемой фотографии, в те же координаты. Таким образом, получается маска.

Так как мы делаем много фотографий и люди на них стараются не двигаться, то в некоторых случаях мы будем довольны нашим результатом. Но как быть если у фотографа дрожали руки и изображение немного сдвинулось? Наш алгоритм подразумевает, что на фото как минимум 2-3 человека, у которых открыты глаза. Поэтому мы возьмем координаты bbox глаз одного человека, у которого открыты глаза на двух фотографиях и найдем дельту. Таким образом, вычисляем сдвиг, который применяем для координат bbox человека с закрытыми глазами.

Приведем примеры выполнения моего алгоритма:
Если камера была зафиксирована на штативе, то мой алгоритм сработает на 100% (рисунок 20):

Рисунок 20. Первый столбец – нужный человек с открытыми глазами. Второй столбец нужный человек с закрытыми глазами. Третий столбец – готовый результат.

Если снимки были сделаны без штатива и произошли небольшие сдвиги, которые влияют на расположение людей в кадре, то мой алгоритм также отлично справляется с задачей (рисунок 21):

Рисунок 21. Первый столбец – нужный человек с открытыми глазами. Второй столбец нужный человек с закрытыми глазами. Третий столбец – готовый результат

Последней частью моего проекта является, создание чат-бота на платформе Telegram. У меня был выбор между созданием веб приложения и чат-ботом, так как основной алгоритм готов и нужно было его обернуть в красивую, понятную любым пользователям программу. Поскольку Telegram является одним из самых популярных мессенджеров во всем мире и его интерфейс понятен почти каждому человеку, то я решил остановиться на нем. Для создания бота я использовал pyTelegramBotApi[8]. Полученное приложение позволяет пользователю осуществить базовый сценарий работы, который включает в себя следующие действия:

- По команде ***/start*** пользователь должен загрузить две фотографии: 1-ая: нужный человек с открытыми глазами, 2-ая: нужный человек с закрытыми глазами (рисунок 22).
- Далее срабатывает алгоритм поиска глаз на фотографии. Бот отправляет пользователю размеченное изображение, в котором пронумерована каждая найденная пара глаз от 0 до n (где n количество людей на фотографии). Пользователь должен выбрать нужного человека и отправить цифру в чат. (рисунок 23)
- После срабатывает основной алгоритм замены глаз, и мы получаем готовое изображение. (рисунок 24)

Рисунок 22. Отправка команды /start и нужных фотографий

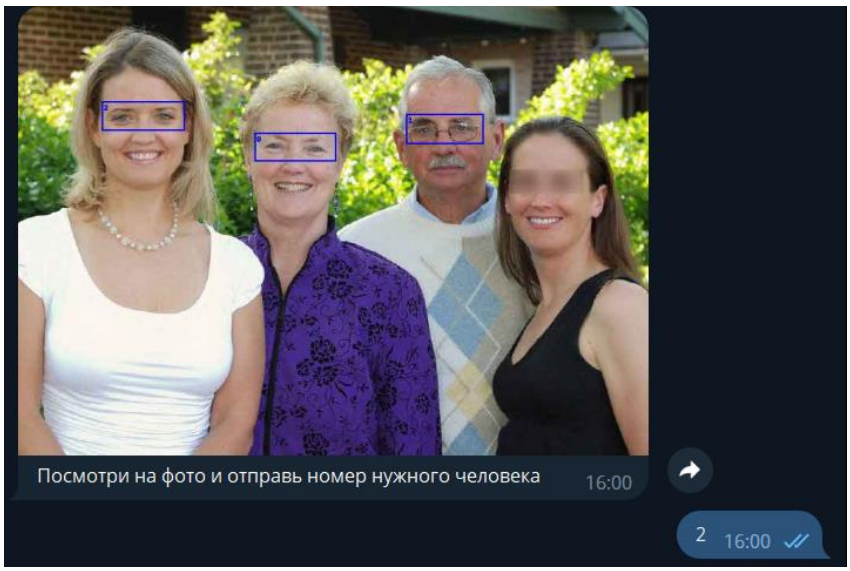

Рисунок 23. Выбор подходящего человек

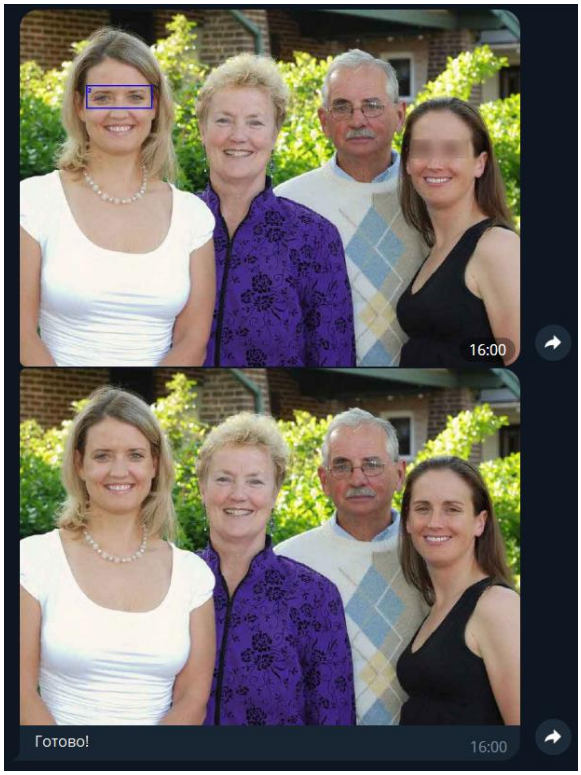

Рисунок 24. Первая фотография – проверка того, что мы выбрали нужного человек.
Вторая фотография – готовый результат

Подводя итоги, можно сказать, что поставленная задача выполнена. Я обучил нейронную сеть, которая может находить расположение глаз на фото. После чего создал алгоритм замены открытых глаз на закрытые. Также обернул проделанную работу в чат-бот, которым может пользоваться каждый. После протестировал получившийся проект на нескольких подготовленных изображениях. Алгоритм и чат-бот сработал отлично. Задача успешно выполнена, но надо понимать, что не все идеально. Я обучал нейронную сеть на относительно небольшом наборе данных, среди которых были в основном фотографии, на которых

находились по 4–6 человек. Одной из проблем является нехватка вычислительных мощностей, из-за которых обучение модели происходит медленно, а это влияет на точность нейронной сети. С новейшим оборудованием появится возможность обучать более крупные модели, например, как YOLOv5L или YOLOv5X, которые помогут улучшить результат. Также, стоит понимать, что не во всех случаях у пользователя найдется две одинаковые фотографии человека с открытыми и закрытыми глазами, поэтому мой алгоритм не универсальный.

Репозиторий с решением - https://github.com/teplov-andrew/OpenEyes

Список источников

1. Сандро Вилинджер. “Сколько фотографий хранится на наших смартфонах? Исследование Avast” [Электронный ресурс] - URL: https://blog.avast.com/ru/which-countries-store-the-most-photos (дата обращения: 16.01.2023)
2. “Помогите исправить закрытые глаза на фото или объясните как это можно сделать самому” [Электронный ресурс] - URL: https://otvet.mail.ru/question/180025703 (дата обращения: 05.01.2023)
3. “Закрытые глаза” [Электронный ресурс] - URL: https://www.babyblog.ru/community/babyblog_photo/post/1613910 (дата обращения: 05.01.2023)
4. “Устранение проблем, связанных с функцией «Открыть закрытые глаза»” [Электронный ресурс] - URL: https://helpx.adobe.com/ru/photoshop-elements/kb/troubleshoot-openclosedeyesissues.html (дата обращения: 06.01.2023)
5. “YOLOv5 Documentation” [Электронный ресурс] - URL: https://github.com/ultralytics/yolov5 (дата обращения: 09.01.2023)
6. Andrew Gallagher, Tsuhan Chen. “The Images of Groups Dataset” [Электронный ресурс] - URL: http://chenlab.ece.cornell.edu/people/Andy/ImagesOfGroups.html (дата обращения: 19.12.2022)
7. “Make sense”. [Электронный ресурс] - URL: https://www.makesense.ai/ (дата обращения: 19.12.2022)
8. “Python implementation for the Telegram Bot API” [Электронный ресурс] - URL: https://pypi.org/project/pyTelegramBotAPI/ (дата обращения: 15.01.2023)
9. Diamond Naga Siu. “Facebook AI is now capable of 'opening' eyes in photos where they're closed” [Электронный ресурс] - URL: https://mashable.com/article/facebook-ai-blinking-eyes (дата обращения: 16.01.2023)
10. “COCO Common Objects in Context” [Электронный ресурс] – URL: https://cocodataset.org/#home (дата обращения: 09.01.2023)