Санкт-Петербургский государственный университет

Кафедра компьютерных технологий и систем

Голокоз Александр Юрьевич

Магистерская диссертация

Идентификация человека по изображению лица

Направление 02.04.02

Фундаментальная информатика и информационные технологии

Магистерская программа «Автоматизация научных исследований»

Научный руководитель:

к.ф.-м.н., доцент

Погожев С. В.

Санкт-Петербург

2018

Содержание

Введение

Задача идентификации личности существует уже давно. Чаще всего используются подходы на основе анализа отпечатков пальцев, геометрии лица, голоса и радужной оболочки глаз. В связи с развитием технологии производства фото и видео камер, качество съёмки увеличивается, а цена устройств уменьшается. Следовательно, возрастает количество используемых устройств. Поэтому подход идентификации личности по геометрии лица набирает всё большую популярность. На основе фото или видео кадра, при наличии в нём человеческого лица, можно определить его геометрию и идентифицировать личность с той или иной точностью.

Наиболее широкое применение поиск личности по фотографии находит в решении задач правоохранительных органов. Большое каличество камер видеонаблюдения на улицах городов и в общественных местах вместе с компьютерными алгоритмами значительно упрощают нахождение разыскиваемых лиц.

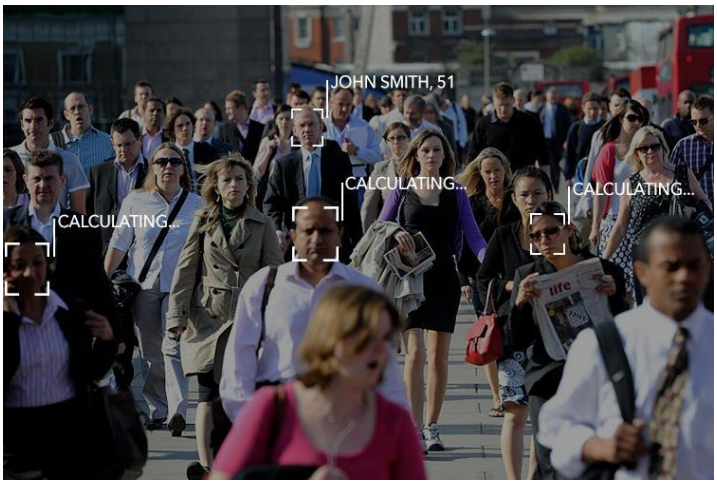

Рис. 1: Работа системы распознавния лиц на кадре с видеопотока городской видеокамеры. Изображение с сайта `https://www.thesun.co.uk/`

Помимо этого, есть и другие применения. Например компании Apple и Samsung в своих смартфонах используют распознавание лиц для разблокировки экрана. Мобильное приложение банка «Открытие» позволяет сделать денежный перевод пользователю имея всего лишь его фотографию.

Рис. 2: Система FaceID от Apple.
Изображение с сайта `http://3dnews.ru/`

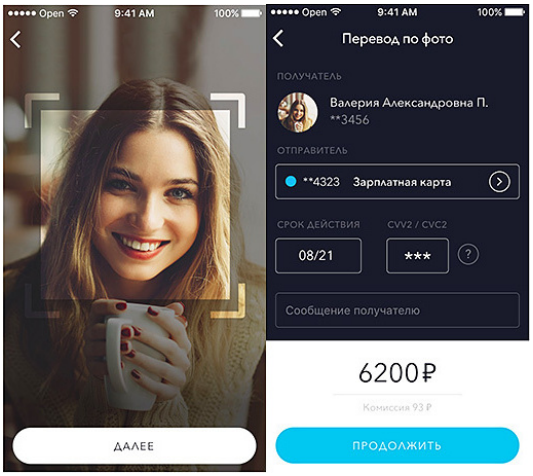

Рис. 3: Мобильное приложение банка.
Изображение с сайта `http://futurebanking.ru`

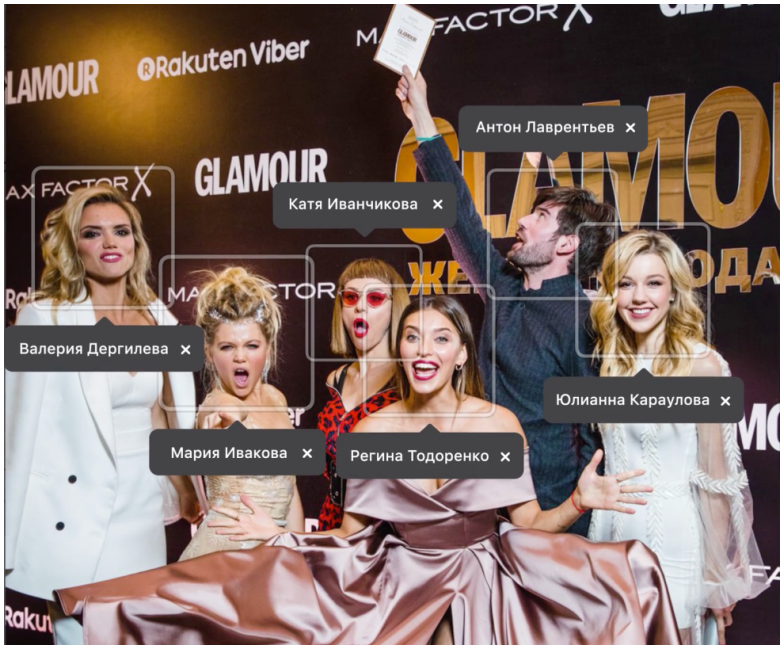

Рис. 4: Автоотметки друзей на фотографии пользователя ВКонтакте.
Изображение с сайта `https://vk.com/`

Социальные сети ВКонакте и Facebook решают задачи автоотметки друзей пользователя на фотографиях с использованием идентификации личности на основе геометрии лица.

В данной работе будет рассмотрен подход к идентификации личности по фотографии с использованием глубоких свёрточных нейронных сетей.

Постановка задачи

Пусть имеется база данных, содержащая для каждого уникального числового идентификатора i характеристики x_{ij} фотографий I_{ij} с изображением лица человека, $i \in \mathbb{N}$, $j = 1, ..., N$, где N – количество фотографий для одного i, содержащих лица одного человека.

Ограничения на количество уникальных идентификаторов отсутствуют. Пусть на вход системе подаётся изображение I в цветовой модели RGB, содержащее лицо человека. Проанализировав изображение I, необходимо определить уникальный идентификатор личности i, изображённой на входном изображении или, при отсутствии личности в базе данных, выдать соответствующий ответ.

Глава 1. Анализ существующих решений

Каждое лицо обладает рядом уникальных характеристик, которые делают его узнаваемым среди других. На сегодняшний день представлено большое количество методов, извлекающих эти характеристики. Существуют методы, основанные на использовании SIFT, LBP, HOG дескрипторов, представленные в [1], [2], [3], [4], [5]. Но лучшим считается класс подходов на основе глубинных нейронных сетей, например FaceNet [6], DeepFace [7], OpenFace [8], Deep Residual Learning for Image Recognition [9].

1.1. Вычисление характеристик лица

Использование глубинных свёрточных нейронных сетей в задачах распознавания изображений начало широко распространяться после победы системы AlexNet [11], использующей глубинную свёрточную нейронную сеть, в соревновании по распознаванию лиц ImageNet recognition challenge.

1.1.1. Свёрточные нейронные сети

Рассмотрим принципы работы свёрточных нейронных сетей. Свёрточные нейронные сети содержат 3 типа слоёв:

1. *Свёрточные*
2. *Субдискретезирующие (подвыборочные, пулинг)*
3. *Полносвязные*

Чередуясь между собой, первые два типа слоёв формируют вектор входа для полносвязного слоя.

Свёрточный слой представляет из себя группу карт признаков (матриц). Матрица этого слоя M' формируется применением к матрице M предыдущего слоя операции свёртки с ядром K и имеет размеры:

$$(w', h') = (m_w - k_w + 1, m_h - k_h + 1)$$

где (m_w, m_h) – размер M, (k_w, k_h) – размер K.

На Рис. 5 проиллюстрирован пример работы слоя.

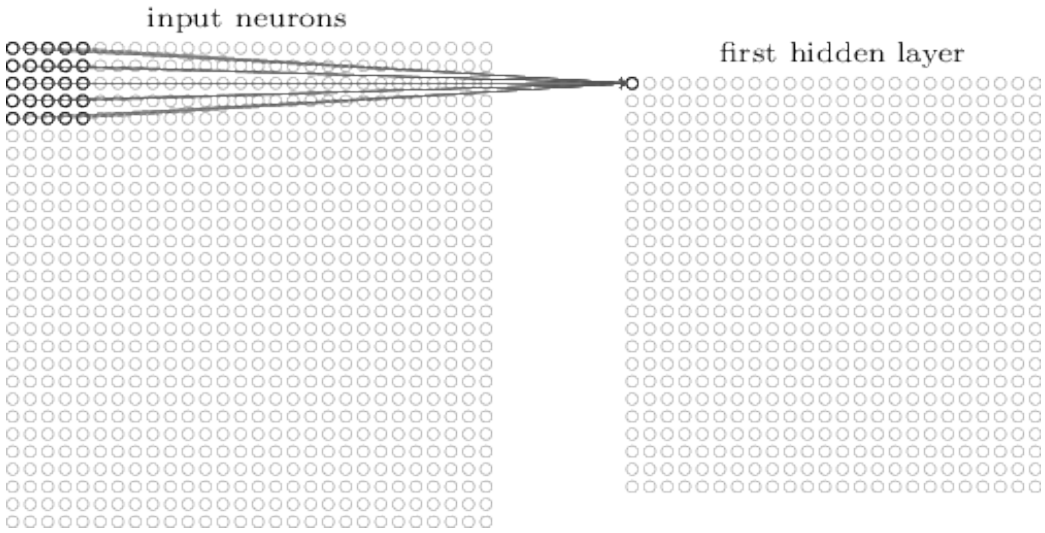

Рис. 5: Схема отображения свёрточного слоя нейронной сети. Слева – M, справа – M'
Изображение с сайта `http://neuralnetworksanddeeplearning.com`

Полное заполнение матрицы M' происходит путём прохождения окном с размером, равным размеру ядра K по всей матрице M и перемножая коэффициенты содержимого окна с ядром. В данном случае подразумевается, что окно каждый раз сдвигается на 1 элемент. В общем случае, смещение по горизонтали и по вертикали может отличаться от 1.

Субдискретезирующий слой уменьшает размерность матрицы предыдущего слоя. Так же, как и в свёрточном слое, заполнение подвыборочного слоя происходит с использованием окон. Только в этом случае, вместо операции свёртки, используется функция, применяемая ко всем элементам окна. Чаще

всего это функция максимума. Кроме этого, смещение онка происходит с шагом, равным размеру самого окна как по горизонтали, так и по вертикали.

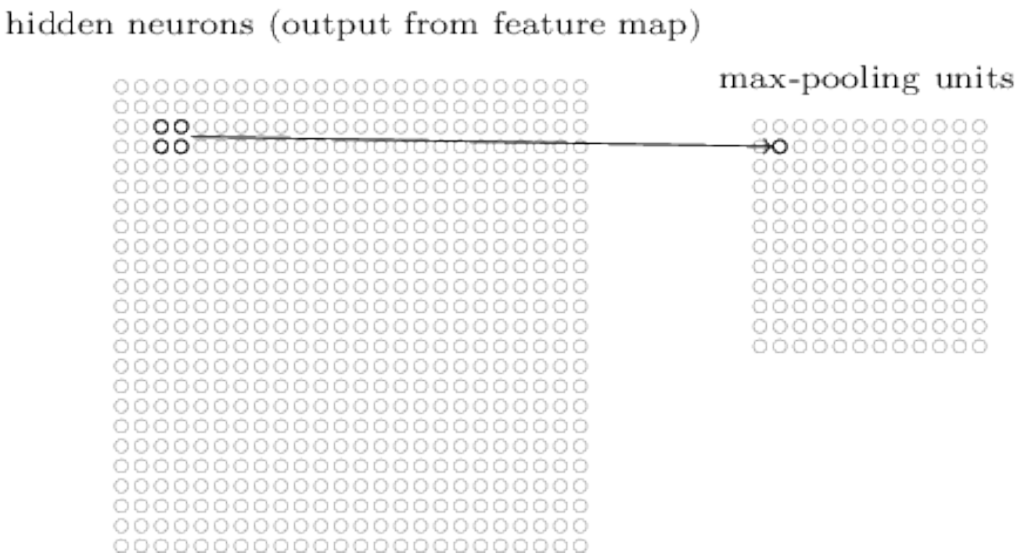

Рис. 6: Схема отображения пулинг слоя нейронной сети
Изображение с сайта `http://neuralnetworksanddeeplearning.com`

Полносвязный слой используется для классификации, моделируя сложную нелинейную функцию. В процессе оптимизации этой функции улучшается качество распознавания.

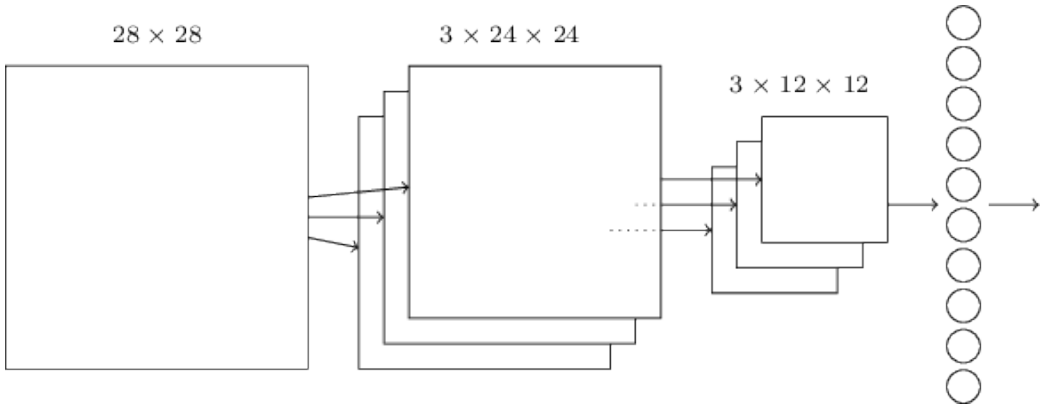

Рис. 7: Пример свёрточной нейронной сети
Изображение с сайта `http://neuralnetworksanddeeplearning.com`

На Рис. 8 представлена схема свёрточной нейронной сети для распозна-

вания рукописных цифр. На вход сеть принимает чёрно-белое изображение размером $28 * 28$ пикселей. На последнем слое i-й нейрон характеризует вероятность того, что сеть получила на вход изображение с цифрой i.

Возвращаясь к истории, необходимо сказать, что AlexNet обладала рядом недостатков для практического применения, среди которых необходимость использовать большие вычислительные мощности из-за большого количества параметров для обучения нейронной сети. Сотрудники корпорации Google занялись усовершенствованием AlexNet, в результате чего получили архитектуру новой сети под названием GoogLeNet или Inception.

Но в 2015 году на ImageNet recognition challenge победу одерживает система, использующая нейронную сеть с глубинной остаточной архитектурой [9].

1.1.2. Остаточные нейронные сети

Известно, что при увеличении количества слоёв нейронной сети, качество модели растёт до некоторого порога, а после его достижения начинает падать. Авторы работы [9] смогли найти решение этой проблемы. Расммотрим основную идею.

Нейронная сеть может аппроксимировать почти любую функцию. Предположим, что модель умеет аппроксимировать некоторую функцию $\mathcal{H}(x)$. Тогда она может аппроксимировать и остаточную функцию $\mathcal{F}(x) = \mathcal{H}(x) - x$. Первоначальная функция будет равна $\mathcal{H}(x) = \mathcal{F}(x)+x$. Проблема деградации качества подразумевает, что модель, аппроксимирующая $\mathcal{F}(x)$, должна прозводить тождественные преобразования на слоях, следующих за тем, на котором был достигнут предел качества. По каким-то причинам этого не происходит. Есть предположение, что дело в оптимизаторе, который не может справиться с настройкой весов таким образом, чтобы иерархическая нелинейная модель производила тождественное преобразование. В работе [9] была произведена

попытка добавить в блоки дополнительное «shortcut»-соединение, представленное на Рис. 8. Тогда, возможно, оптимизатору будет легче сделать веса близкими к нулю, вместо построения тождественного преобразования. При-

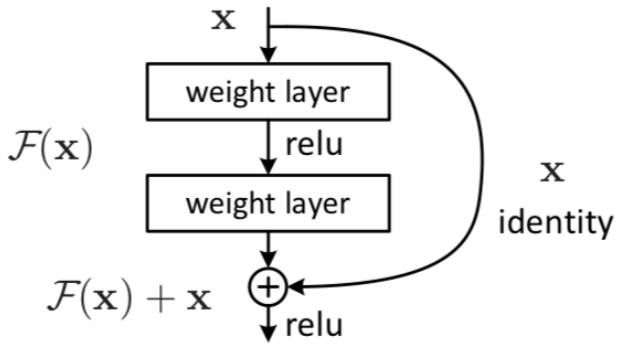

Рис. 8: Схема блока остаточной сети
Изображение из работы [9]

чину, по которой соединение проходит через два слоя, авторы не представили. Вероятно такой результат был достигнут опытным путём.

1.1.3. Анализ существующих нейронных сетей

После ImageNet recognition challenge 2015 года появляются архитектуры Inception-v4 (без использования остаточных слоёв) и Inception-ResNet (с использованием остаточных слоёв) [12] от Google. Причём Inception-ResNet показывает немного лучшие результаты.

Большей популярностью пользуется класс нейронных сетей, выдающий характеристики лица при данном на входе изображении, содержащим лицо человека, в виде вещественного вектора. Такие нейронные сети переводят двумерное изображение лица в n-мерное векторное пространство. Причём для одного и того же человека в Евклидовом пространстве векторы располагаются близко друг к другу и далеко для разных людей. Этот подход является более гибким, в сравнении с тем, при котором для добавления нового лица

необходимо перестраивать последние слои неройнной сети.

Для тестирования систем распознавания лиц существуют тестовые наборы фотографий. Один из самых известных – это LFW (Labeled Faces in the Wild) [10]. Он содержит 13,233 изображений 5,749 известных личностей. В Таблице 1 представлены результаты тестирования некоторых из упомянутых выше систем на тестовом наборе фотографий Labeled Faces in the Wild.

Система	**Точность**	**Количество сетей**
FaceNet	99.63%	1
DeepFace	98.95%	1
OpenFace	97.35%	3

Таблица 1: Результаты тестирования систем распознавания лиц на наборе LFW. В последнем столбце указано количество нейронных сетей в системе.

Каждая из систем в своей основе использует свёрточные глубинные нейронные сети. В итоге, нейронная сеть ставит входящему изображению I' вектор характеристики x' найденного на нём лица.

1.2. Классификаторы

Нужно найти такую функцию F, которая для вектора характеристики x' фотографии I', причём $I' \notin I_i$, возвращает i в случае, если на изображении I' присутствует лицо человека с идентификатором i или 0, если соответствующее значение не найдено.

Используя тот факт, что векторы характеристик лица для одного человека располагаются близко друг к другу и далеко для разных людей, для построения F можно применить наивный подход поиска ближайшего вектора

по L_2 норме. Введём функцию

$$D = \min ||x' - x_{ij}||_2^2,$$

вычисляющую минимальное расстояние между x' и всеми x_{ij}. И некоторый вещественный порог τ. Если расстояние между двумя векторами характеристик меньше, либо равно τ, считаем, что характеристики принадлежат одному человеку, а иначе – разным. Тогда

$$F(x') = \begin{cases} i, \arg\min_i (D \leq \tau) \\ 0, D > \tau \end{cases}$$

Ещё одним вариантом решения задачи построения функции F является построение многоклассового классификатора на основе метода опорных векторов с обучающей выборкой. Принцип работы классификатора описан в [13]. Классификатор позволяет реализовать отображение

$$\mathbb{R}^n \to \mathbb{N},$$

то есть из n-мерного пространства в множество натуральных чисел. В нашем случае: из векторного пространства характеристик в множество уникальных идентификаторов. Но этот подход обладает недостатком квадратичной сложности добавления новых классов. Подробнее об этом написано в [14].

В [15] для быстрого решения задачи поиска похожих векторов предлагается использовать квантование векторов, описанное в [16]. Подход позволяет применить сжатие с потерями к векторам, в результате чего количество бит для представления вектора снижается в 8 раз. Несмотря на сжатие с потерями, точность поиска практически не меняется. Помимо этого, в [15] для ускорения поиска предлагается строить специальные индексы и использовать вычисления на графических процессорах.

1.3. Обнаружение лица и ключевых точек

Для корректной и качественной работы системы все используемые фотографии перед подачей на вход необходимо предобработать.

В первую очередь необходимо найти лицо изображении. Для решения этой задачи существуют методы обнаружения и локализации лиц. Одним из самых известных и эффективных методов является метод Виолы-Джонса, описанный в [17]. Этот алгоритм используется в библиотеке OpenCV [18] в качестве стандартного. В своей основе метод Виолы-Джонса использует следующее:

1. *Интегральное представление изображения* для быстрых вычислений.
2. *Признаки Хаара* для поиска объектов.
3. Выбор наиболее подходящих признаков для искомого объекта осуществляется используя *бустинг*.
4. *Классификаторы* отсеивают найденные признаки.
5. *Каскады признаков* для быстрого исключения окон, в которых найдено лицо.

Помимо этого, существует подход к решению задачи обнаружения лиц на основе гистограммы направленных градиентов [19] и метода опорных векторов [13]. Идея алгоритма состоит в следующем допущении: форма и внешний вид объекта на изображении могут быть описаны распределением градиентов интенсивности или направлением краёв. Изображение разбивается на небольшие ячейки, в каждой из которых вычисляется гистограммы направлений градиентов. Эти гистограммы являются дескриптором. Реализация данного подхода в библиотеке DLib [20] по скорости работы сравнима со скоростью реализации подхода Виолы-Джонса в OpenCV. Но алгоритм Виолы-Джонса

даёт большое количество ложных срабатываний, в сравнении с алгоритмом, использующим градиенты. Кроме того, для обучения системы, использующейся в OpenCV требуется намного большее количество времени и больший размер обучающей выборки.

Ещё одним вспомогательным инструментом для улучшения качества работы системы распознавания лиц является детектор ключевых точек лица (лэндмарков). С помощью этих ключевых точек можно, например, выполнить поворот лица на угол, при котором, прямая, проходящая через центры глаз человека, параллельна горизонтальной оси. Примеры работ с алгоритмами поиска ключевых точек лица: One Millisecond Face Alignment with an Ensemble of Regression Trees [21], Face Alignment at 3000 FPS via Regressing Local Binary Features [22]. Метод из первой работы основан на градиентном бустинге группы деревьев решений. В методе из второй работы используются локальные бинарные шаблоны. В библотеке OpenCV содержится реализция каждого из этих алгоритмов. Библиотека DLib имеет реализацию только первого.

Глава 2. Реализация

В данной главе будет представлена реализация решения задачи, представленной в формализации. На Рис. 9 изображена общая схема работы алгоритма. Для реализации был использован язык программирования высокого

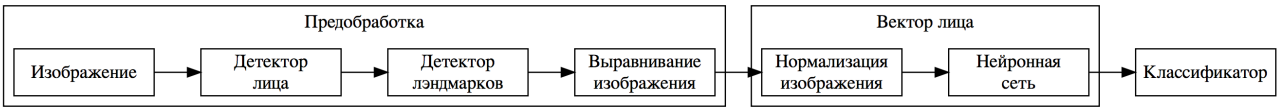

Рис. 9: Схема алгоритма

уровня Python и следующие библиотеки:

1. OpenCV – библиотека, содержащая реализацию алгоритмов компьютерного зрения. Написана на C++.

2. DLib – библиотека, содержащая алгоритмы машинного обучения для решения различных задач. Написана на C++.

3. TensorFlow [23] – фреймворк для решения задач машинного обучения. Использована реализация на Python.

4. Faiss [24] – библиотека, разработанная компанией Facebook, для эффективного поиска векторов. Написана на C++.

Для более эффективной работы системы библиотека OpenCV была скомпилирована с флагом WITH_CUDA=1, что позволило перенести часть вычислений на графический процессор и, следовательно, увеличить скорость вычислений.

Фреймворк TensorFlow так же позволяет использовать графические процессоры для ускорения вычислений. Эта опция была использована.

Рассмотрим каждую часть схемы, представленной на Рис. 9 подробнее на конкретном примере.

1.1. Предобработка фотографии

Алгоритм на вход принимает цветное изображение. В первую очередь необходимо проверить изображение на наличие лица. И в случае присутствия, определить его координаты на изображении. Для решения этой подзадачи используется реализация метода [19] из DLib. Пример этого шага изображён на Рис. 10 и Рис. 11.

Рис. 10: Исходная фотография

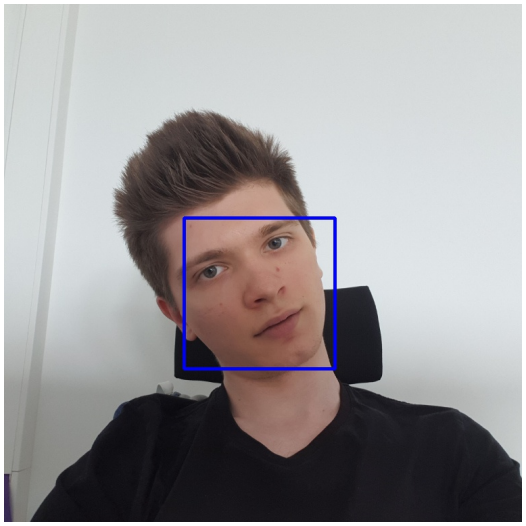

Рис. 11: Граница лица

Для качественной работы всей системы, нейронной сети на вход должны подаваться изображения с одинаково выравненными лицами. Будем считать изображение лица выравненным, если прямая, соединяющая центры глаз, параллельна горизональной оси изображения. Но идеальная параллельность не обязательна, так как свёрточные сети устойчивы к небольшим подобным отклонениям. Кроме того, все изображения должны иметь одинаковый размер, который зависит от входного слоя нейронной сети. Таким образом, вторым

шагом в приведённой схеме будет поиск ключевых точек лица. Будем использовать реализацию метода [21] из DLib.

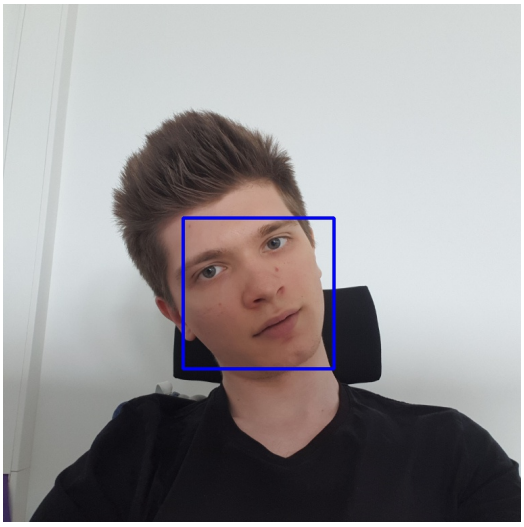

Рис. 12: Граница лица

Рис. 13: Ключевые точки

На следующем шаге необходимо выполнить выравнивание изображения и изменение его размера. Нейронная сеть, использующаяся в системе, принимает на вход изображение в цветовой системе RGB и размером $160 * 160$ пикселей. Для выравнивания была использована реализация аффинного преобразования из библиотеки OpenCV.

Рис. 14: Ключевые точки лица

Рис. 15: Выравненное изображение

Результат работы блока предобработки изображён на Рис. 15

1.2. Вычисление вектора лица

Для вычисления вектора лица была использована натренированная модель нейронной сети с архитектурой Inception-ResNet. Работа с нейронной сетью происходит через фреймворк TensorFlow. Входной слой используемой сети имеет 76800 нейронов. Такое число получается из размера изображения и количества каналов на пиксель: $160 * 160 * 3 = 76800$.

Рассмотрим матрицу изображения, представленного на Рис. 15. Каждый

$$\begin{pmatrix} (38\ 40\ 51) & (43\ 45\ 56) & (44\ 46\ 57) & \cdots & (227\ 224\ 219) \\ (40\ 42\ 53) & (43\ 45\ 56) & (42\ 44\ 55) & \cdots & (229\ 226\ 221) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ (76\ 88\ 122) & (78\ 90\ 124) & (79\ 91\ 125) & \cdots & (218\ 215\ 210) \end{pmatrix} \quad (1)$$

элемент матрицы (1) представляет из себя вектор, идентифицирующий цвет в системе RGB.

Нейронная сеть работает с нормализованными изображениями, поэтому прежде, чем подать изображение на вход, его необходимо нормализовать. Нормализация каждого пикселя происходит по следующей формуле:

$$\frac{x - m}{\max(S, 1/\sqrt{n})}$$

где x – значение пикселя,

m – среднее значение всех пикселей на изображении,

S – стандартное отклонение всех пикселей на изображении,

n – количество пикселей на изображении. В нашем случае n будет постоянным значением, равным $160 * 160 = 25600$.

Ниже представлен результат нормализации матрицы (1). Для удобства отображения все значения были округлены.

$$\begin{pmatrix} (-1.39\ -1.35\ -1.14) & (-1.29\ -1.25\ -1.04) & \cdots & (2.30\ 2.24\ 2.14) \\ (-1.35\ -1.31\ -1.10) & (-1.29\ -1.25\ -1.04) & \cdots & (2.26\ 2.20\ 2.10) \\ \vdots & \vdots & \ddots & \vdots \\ (-0.66\ -0.43\ 0.22) & (-0.62\ -0.39\ 0.26) & \cdots & (2.07\ 2.01\ 1.91) \end{pmatrix} \quad (2)$$

Далее нормализованное изображение подаётся на вход нейронной сети, которая вычисляет вектор лица, содержащий 128 вещественных чисел.

Таким образом, результатом работы блока вектора лица для исходного изображения стало следующее преобразование из матрицы в вектор:

$$(1) \rightarrow (-0.06670962\ 0.00993852\ -0.09099827\ ...\ 0.0625081\ 0.0215367) \quad (3)$$

1.3. Классификация

Используя тот факт, что для разных фотографий одного и того же человека нейронная сеть будет генерировать векторы, близко располагающиеся друг к другу в пространстве, для решения задачи поиска похожих векторов можно применить инструмент, вычисляющий необходимое значение по L_2 расстоянию. В качестве такого инструмента было решено использовать библиотеку Faiss.

Система распознавания, в целом, имеет две функции: добавление фотографий человека в базу данных и поиск человека по фотографии. В первом случае, вычисленный на предыдущем шаге вектор лица, просто добавляется в базу данных вместе с уникальным идентификатором вектора, равным

его порядковому номеру в базе. В случае поиска классификатор возвращает порядковый номер ближайшего вектора к искомому.

Стоит напомнить, что если системе на вход подаётся фотография человека, которого нет в базе, система должна выдать соответствущий ответ, а не ближайший к подаваемому на вход вектор. Этот случай решается сравнением L_2 расстояния d входного и выходного векторов с некоторым порогом t. Если $d < t$, система выдаёт найденный вектор как ответ, иначе возвращает ответ об отстутствии вектора в базе данных. Эмпирическим путём было установлено, что порог $t = 1.256$ оказался оптимальным.

Таким образом, для идентификации личности необходимо иметь отображение, ставящее в соответствие каждому порядковому номеру вектора в базе уникальный идентификатор личности. Для этого была использована реляционная база данных SQLite, имеющая таблицу, в каждой строке которой содержится порядковый номер вектора в базе данных векторов и уникальный идентификатор личности.

1.4. Источники данных

В качестве фотографий для базы данных были использованы аватары друзей автора работы в социальной сети ВКонтакте.

Каждая фотография проходила каждый этап вышеприведённой схемы. Из-за разнородности фотографий не всех пользователей удалось добавить в базу. У некоторых на автарах не было лиц, а у некоторых не было аватаров вообще, но таких пользователей оказалось меньшинство. Если на фотографии оказывалось два или более лица, то на стадии детектирования и локализации выбиралось то, у которого наибольшая площадь прямоугольника, описывающего границу лица.

Вообще для выбора лиц, которые будут использованы для идентификации личности необходимо использовать более сложный алгоритм. Во-первых, он должен рассматривать не только аватары, но и фотографии. Чем больше фотографий будет учтено, тем точнее будет работать классификатор. А во-вторых, алгоритм должен учитывать тот факт, что на фотографиях могут быть и другие люди.

Для генерации фотографий людей, использующихся для распознавания, было создано приложение для мобильной операционной системы Android, написанное на языке программирования высокого уровня Kotlin [25]. На Рис. 16 представлена фотография работы приложения.

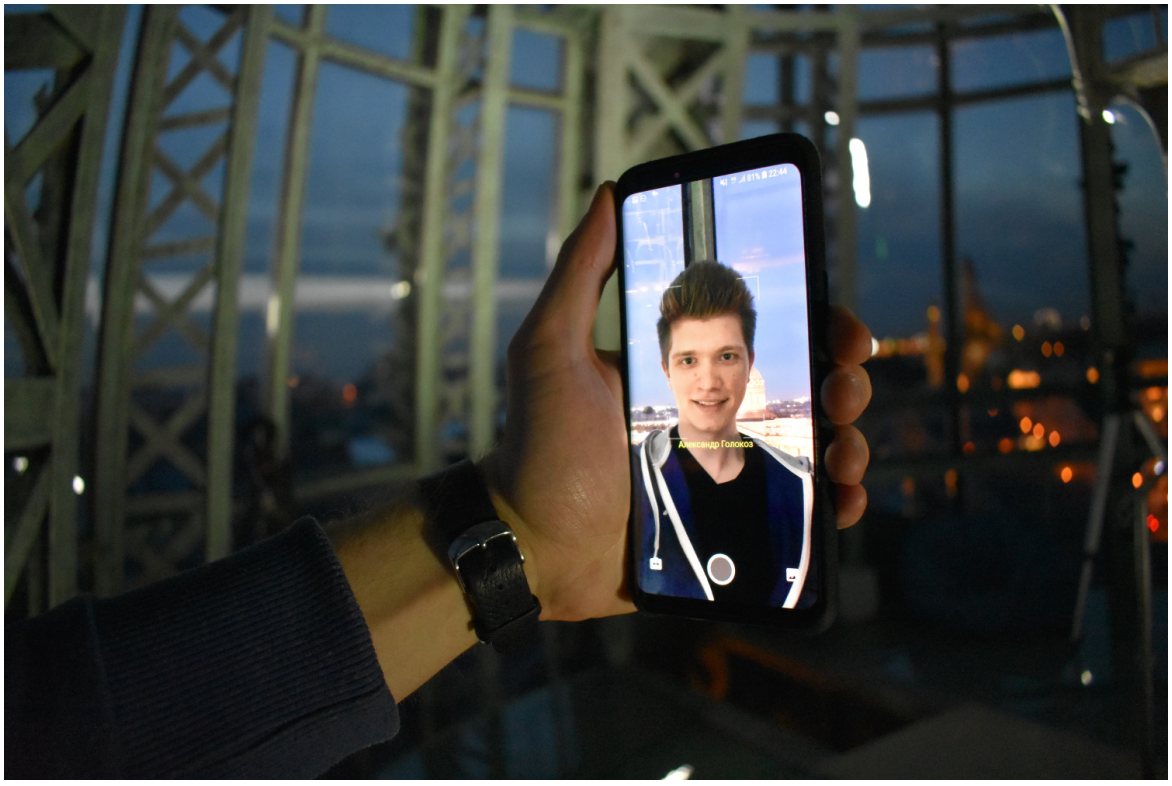

Рис. 16: Демонстрация работы приложения

Главный экран транслирует изображение камеры смартфона. В режиме реального времени происходит детектирование и трекинг лиц. Найденные

лица описываются прямоугольником вокруг них. Если в камере появляется новое лицо, то первый кадр с ним захватывается обработчиком, который выполняет создание нового изображения, вырезав исходное по границе прямоугольника, описывающего найденное лицо, и отправляет его на сервер по протоколу http для дальнейшей обработки. После получения ответа от сервера, приложение под границей лица отображает имя и фамилию найденной личности или пустую строку, если личность не найдена. Таким образом, приложение реализует первые два элемента из схемы на Рис. 9.

Глава 3. Примеры

Рассмотрим добавление человека в систему на примере автора. На момент написания данной работы на странице автора в сети ВКонтакте было 5 аватаров

Рис. 17: Аватар 1

Рис. 18: Аватар 2

Рис. 19: Аватар 3

Рис. 20: Аватар 4

Рис. 21: Аватар 5

На шаге детектирования лиц, детектор не обнаружил контуров на фотографиях Рис. 19 и Рис. 21. Для первого случая такое поведение ожидаемо: лицо повёрнуто на угол, близкий к 90°. Что стало причиной того, что лицо не

было обнаружено во втором случае – точно не известно. Возможно неравномерное освещения лица.

Итак, для фотографий Рис. 17, Рис. 18, Рис. 20 в базу данных были добавлены следующие векторы характеристик:

$$(-0.06256077\ 0.13453765\ -0.10341253\ \ldots\ 0.0449359\ 0.0570382) \tag{4}$$

$$(-0.00247308\ 0.03541066\ -0.10081527\ \ldots\ 0.0983395\ 0.0281178) \tag{5}$$

$$(-0.00595176\ 0.12892038\ -0.10902489\ \ldots\ 0.0583114\ -0.0491819) \tag{6}$$

Для рассматриваемого примера на Рис. 10 из предыдущей главы, среди 3782 векторов фотографий порядка 300 друзей, наиболее близким по L_2 расстоянию векторов характеристик (3) и (6) стало изображение, представленное на Рис. 20

Теперь рассмотрим работу Android приложения.

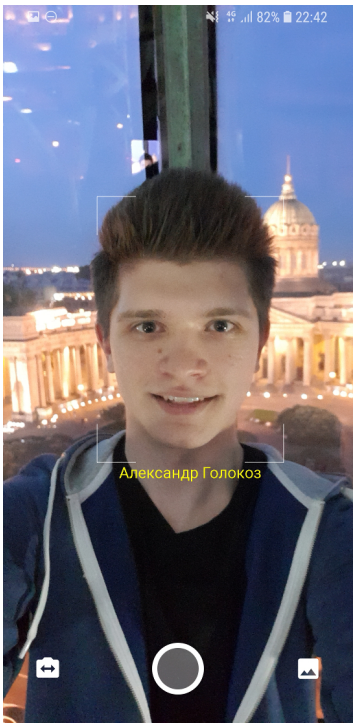

Рис. 22

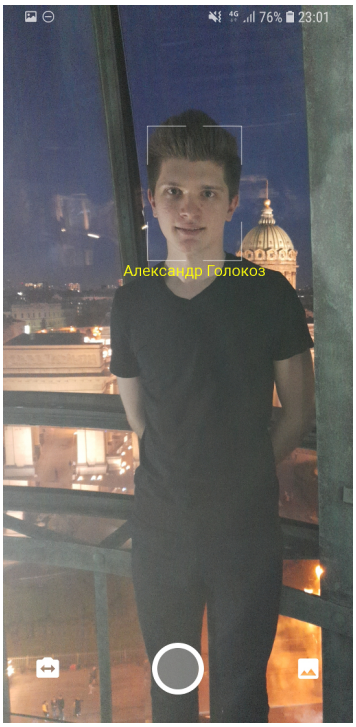

Рис. 23

Рис. 24

На Рис. 22, Рис. 23, Рис. 24 представлены скриншоты приложения с идентифицированным автором работы. Как видно из Рис. 24 идентификация происходит и при повороте головы в разных плоскостях.

Теперь рассмотрим примеры с другим человеком.

Рис. 25

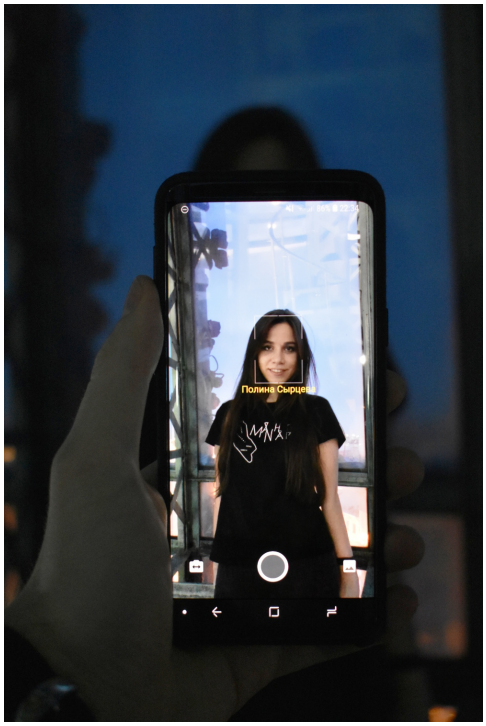

Рис. 26

Рис. 27

На Рис. 27 локализация лица произошла, а идентификация – нет. Успешное выделение границы лица объясняется предварительным обесцвечиванием изображения для увеличения эффективности работы алгоритма поиска. Что касается идентификации, то нейронная сеть принимает на вход цветное изображение и освещение на фотографии вносит свой вклад в качество распознавания. В данном случае, цвет лица сильно отличается от того, который был на примерах в базе данных. Решением подобной проблемы может стать использование нейронной сети, принимающей на вход изображение в оттенках серого цвета.

Для лиц, у которых L_2 расстояние между вектором характеристики и бли-

жайшим вектором из базы оказалось больше порога t, рассмотренного ранее, результат будет аналогичен Рис. 27: под границей лица ничего не будет выведено.

Как видно из Рис. 28 приложение поддерживает одновременное детектирование, трекинг и распознавание нескольких лиц.

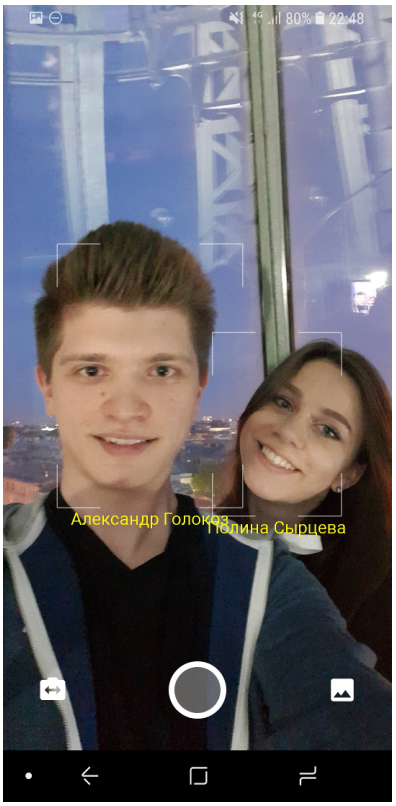

Рис. 28

Несмотря на то, что одно из лиц повёрнуто на угол крена около 45°, его распознавание произошло.

Заключение

В данной работе приведено решение задачи идентификации личности по фотографии. Приведена общая схема работы системы, решающей задачу идентификации. Рассмотрена работа системы на всех этапах – от поиска лица на фотографии до получения уникального идентификатора личности. Для каждой подзадачи были рассмотрены и сравнены современные алгоритмы. Продемонстрированы примеры, показывающие работоспособность системы, а так же контрпримеры, на которых система не может вычислить ответ.

Вместо стандартного подхода для классификации векторов с использованием метода опорных векторов, была использована библиотека Faiss, которая показала хорошие результаты в построенной системе.

Получившаяся реализация представлена в двух программах:

1. *Сервер.* Исполняет добавление фотографий людей в базу данных и поиск людей по фотографии.

2. *Клиент.* Android приложение, создающее миниатюру, отправляющее её на сервер для распознавания и выдающее ответ системы.

Для дальнейшего развития и улучшения качества работы системы можно рассмотреть следующие идеи.

При малом количестве фотографий у добавляемого в базу данных человека можно добавлять искусственно созданные выравненные изображения путём зеркального отражения исходной выравненной фотографии. Кроме того, можно генерировать изображения, использовав вырез случайной области на выравненном изображении, но вырез должен содержать лицо. Небольшими изменениями контрастности и яркости так же можно увеличить набор изображений для одной личности.

Для выравнивания лица производить вычисление не всех ключевых точек лица, а только необходимых: края глаз, губ и центр носа. Это увеличит скорость работы всего алгоритма.

Для идентификации личности при условиях, как на Рис. 27, использовать вторую нейронную сеть, принимающую на вход изоюражение в градациях серого. В итоге система будет использовать две нейронные сети.

Узким местом в скорости работы приложения является передача изображения на сервер. Этот этап занимает наибольшее количество времени. Решением может стать перенос детектора лэндмарков и выравнивания фотографии в приложение. Тогда по сети будут отправляться изображения размером 160*160 пикселей, что снизит время на передачу данных. Можно пойти далее и перенести вычисление вектора лица на смартфон и использовать сервер только в качестве классификатора. Тогда по сети будет передаваться только 128 вещественных чисел. Но в этом случае основные вычисления будут происходить на устройстве, что вызовет увеличение времени от получения кадра с лицом до отправки данных на сервер. Здесь нужны эксперименты.

Ещё одной проблемой для смартфона является большой размер нейронной сети – 89.01 Мб. Однако Google анонсировала инструмент Learn2Compress [26] для уменьшения размера моделей с небольшими потерями точности. В примерах демонстрируется сжатие модели в примерно 20 раз с потерями точности около 7%. Сжатие достигается за счёт:

1. Удаления наименее полезных весов и операций

2. Вспользования квантования чисел с плавающей точкой, позволяющего одно число хранить в ячейке памяти размером 1 байт

3. Подхода совместного обучения

Получившаяся сжатая сеть работает быстрее, занимает меньше места на диске и в оперативной памяти. Это то, что нужно для использования в приложении. На момент написания работы инструмент был на стадии закрытого тестирования и возможности опробовать его не было.

Список литературы

[1] R. G. Cinbis, J. J. Verbeek, and C. Schmid. Unsupervised metric learning for face identification in TV video. In Proc. ICCV, pages 1559–1566, 2011.

[2] C. Lu and X. Tang. Surpassing human-level face verification performance on lfw with gaussian- face. AAAI, 2015.

[3] J. Sivic, M. Everingham, and A. Zisserman. Person spotting: Video shot retrieval for face sets. In Proc. CIVR, 2005.

[4] J. Sivic, M. Everingham, and A.Zisserman. "Who are you?" – learning person specific classifiers from video. In Proc. CVPR, 2009.

[5] L. Wolf, Tal. Hassner, and I. Maoz. Face recognition in unconstrained videos with matched background similarity. In Proc. CVPR, 2011.

[6] F. Schroff, D. Kalenichenko, J. Philbin. FaceNet: A Unified Embedding for Face Recognition and Clustering. https://arxiv.org/pdf/1503.03832.pdf

[7] M. Parkhi, A. Vedaldi, A. Zisserman. Deep Face Recognition. https://www.robots.ox.ac.uk/~vgg/publications/2015/Parkhi15/parkhi15.pdf

[8] B. Amos, B. Ludwiczuk, M. Satyanarayanan. OpenFace: A general-purpose face recognition library with mobile applications. https://www.cs.cmu.edu/~satya/docdir/CMU-CS-16-118.pdf

[9] K. He, X. Zhang, S. Ren, J. Sun. Deep Residual Learning for Image Recognition. 2015. https://arxiv.org/pdf/1512.03385v1.pdf

[10] Labeled Faces in the Wild. http://vis-www.cs.umass.edu/lfw/

[11] A. Krizhevsky, I. Sutskever, G.E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[12] C. Szegedy, S. Ioffe, V. Vanhoucke. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. https://arxiv.org/pdf/1602.07261.pdf

[13] SVM tutorial. https://svmtutorial.online/download.php?file=SVM_tutorial.pdf

[14] S. Shalev-Shwartz, N. Srebro. SVM Optimization: Inverse Dependence on Training Set Size. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.139.2112&rep=rep1&type=pdf

[15] J. Johnson, M. Douze, H. Je .Billion-scale similarity search with GPUs. https://arxiv.org/pdf/1702.08734.pdf

[16] T. Ge, K. He, Q. Ke, J. Sun. Optimized Product Quantization for Approximate Nearest Neighbor Search. https://www.robots.ox.ac.uk/~vgg/rg/papers/ge__cvpr2013__optimized.pdf

[17] P. Viola, M. J. Jones. Robust Real-Time Face Detection. http://www.vision.caltech.edu/html-files/EE148-2005-Spring/pprs/viola04ijcv.pdf

[18] OpenCV Library. http://opencv.org/

[19] N. Dalal, B. Triggs. Histograms of Oriented Gradients for Human Detection. http://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf

[20] DLib Library. http://dlib.net/

[21] V. Kazemi, J. Sullivan. One Millisecond Face Alignment with an Ensemble of Regression Trees. https://pdfs.semanticscholar.org/d78b/6a5b0dcaa81b1faea5fb0000045a62513567.pdf

[22] S. Ren, X. Cao, Y. Wei, J. Sun. Face Alignment at 3000 FPS via Regressing Local Binary Features. http://www.jiansun.org/papers/CVPR14_FaceAlignment.pdf

[23] TensorFlow framework.https://www.tensorflow.org

[24] Faiss library. https://github.com/facebookresearch/faiss

[25] Kotlin Programming Language. https://kotlinlang.org

[26] Google AI Blog. https://ai.googleblog.com/2018/05/custom-on-device-ml-models.html