Содержание

Введение

Задача распознавания изображений является очень важной, так как возможность автоматического распознавания компьютером изображений приносит множество новых возможностей в развитии науки и техники, таких, как разработка систем поиска лиц и других объектов на фотографиях, контроля качества производимой продукции без участия человека, автоматического управления транспортом и множество других.

В настоящее время для решения задачи распознавания широко используются технологии искусственного интеллекта.

Поэтому целью данной работы является реализация модели глубокого обучения для идентификации человека по лицу. Данную модель можно будет интегрировать в приложения, связанные с пропускной системой.

1 Аналитический обзор предметной области

1.1 Обзор задачи распознавания лиц

Задача идентификации человека по лицу является одной из ключевых задач в области компьютерного зрения и распознавания образов. Ее цель состоит в распознавании уникальных характеристик лица и сопоставлении их с известными лицами в базе данных для определения личности.

Вот обзор основных аспектов и задач, связанных с идентификацией человека по лицу:

– Обнаружение лиц. Это первоначальный этап, на котором нейросеть или алгоритм обнаруживает наличие лиц на изображении. Здесь используются методы обнаружения объектов, такие как каскады Хаара, модели глубокого обучения или другие современные алгоритмы;

– Выравнивание лиц. Лица на изображениях могут иметь разные размеры, ориентацию и углы обзора. Выравнивание лиц выполняется для приведения лиц к стандартному формату, обеспечивая одинаковый размер, ориентацию и положение ключевых точек. Это помогает сделать извлечение признаков более точным и устойчивым;

– Извлечение признаков. Важным шагом является извлечение уникальных признаков из лица, которые могут быть использованы для идентификации. Сверточные нейронные сети (CNN) обычно применяются для извлечения признаков, которые представляют текстуры, границы, формы и другие дискриминативные характеристики лица;

– Создание эмбеддингов. После извлечения признаков лица применяется алгоритм для создания эмбеддинга лица, то есть числового представления, которое компактно и однозначно описывает лицо. Эмбеддинги лиц могут быть использованы для сравнения и распознавания лиц;

– Создание базы данных идентификации. Для задачи идентификации необходима база данных, содержащая изображения лиц вместе с соответствующими метками классов или личностей. Эта база данных используется для тренировки и тестирования модели идентификации лиц;

– Классификация или распознавание лиц. После создания эмбеддингов лиц идет этап классификации, где нейросеть сопоставляет эмбеддинги лица с известными лицами в базе данных для определения личности. Здесь используются алгоритмы классификации, такие как метод ближайших соседей (k-nearest neighbors), метод опорных векторов (Support Vector Machines, SVM), глубокие нейронные сети и другие;

– Оценка производительности. Важным аспектом задачи идентификации человека по лицу является оценка производительности модели. Она может быть оценена по таким метрикам, как точность идентификации, матрица ошибок, скорость обработки и другие;

– Учет проблем и решение. При идентификации человека по лицу возникают ряд проблем, таких как изменение освещения, изображения с низким разрешением, наклонные или повернутые лица, наличие аксессуаров и т. д. Для их решения используются техники, такие как аугментация данных, использование различных углов искажения лица, моделирование освещения и другие.

Задача идентификации человека по лицу имеет широкий спектр применений, включая системы безопасности, автоматизированное управление доступом, видеонаблюдение, разработку приложений распознавания лиц и другие. [1]

1.2 Перспективность идентификации по лицу

Идентификация по лицу - это технология, которая позволяет распознавать и устанавливать личность человека на основании его

геометрических параметров. Эта технология имеет много преимуществ перед другими методами идентификации:

– Безопасность. Использование идентификации по лицу может повысить уровень безопасности. Эта технология может быть использована для защиты персональных данных и финансовых транзакций.

– Удобство. Идентификация по лицу не требует использования паролей или других устройств аутентификации. Это означает, что пользователи могут получать доступ к своей информации быстрее и проще.

– Точность. Идентификация по лицу может быть не только быстрой и удобной, но и точной. Технология использует биометрические данные, которые трудно подделать или подменить.

– Экономическая эффективность. Использование идентификации по лицу может уменьшить расходы на обслуживание клиентов, так как технология может автоматически производить проверку личности пользователя.

– Экологическая эффективность. Использование идентификации по лицу может сократить количество отпечатков пальцев, паролей и других устройств аутентификации, что сокращает использование бумаги и пластика.

– Улучшение опыта пользователей. Идентификация по лицу может повысить удобство пользования услугами в сфере онлайн-торговли, банковскими услугами и др. Также это может привлечь новых пользователей и увеличить лояльность существующих.

В целом, использование идентификации по лицу может повысить безопасность, упростить процессы идентификации и обеспечить более удобный и приятный опыт для пользователей. [2]

1.3 Постановка задачи

Проведенный обзор предметной области показал целесообразность и эффективность применения методов машинного обучения, а именно глубокого обучения, для идентификации по лицу. Поэтому целью данной работы является реализация системы идентификации человека по лицу.

Для достижения поставленной цели необходимо выполнить следующие задачи:

- сформировать тестовый датасет, состоящий из множества изображений лиц различных личностей;
- выбрать модель для обнаружения лиц на изображении;
- использовать выравнивание обнаруженных на изображении лиц;
- выбрать модель для извлечения признаков из изображений лиц;
- выбрать модель для классификации лиц;
- собрать систему моделей;
- провести тестирование системы моделей на приближенном к реальности примере – системе идентификации сотрудников на пропускном пункте небольшой компании.

2 Описание моделей распознавания лиц

2.1 Обоснование выбора модели

В данной главе будет выбраны модели для подзадач выполнения идентификации:

– выбор модели обнаружения лиц;
– выбор модели извлечения признаков;
– выбор модели классификатора.

2.2 Выбор модели обнаружения лиц

Существует несколько известных моделей для детекции лиц, которые достигли высоких результатов в задаче обнаружения лиц. Перечислим некоторые из них.

2.2.1 Каскады Хаара

Один из первых успешных методов обнаружения объектов, включая лица. Они основаны на использовании характерных шаблонов, называемых "Хаар-признаками", они изображены на рисунке 2.1. Каскады Хаара позволяют быстро обнаруживать лица, но могут иметь ограниченную точность в сложных условиях. Старый метод компьютерного зрения, представленный в OpenCV с 2000 года. Это модель машинного обучения с функциями, выбранными специально для обнаружения объектов. Классификаторы Хаара быстры, но имеют низкую точность. [3]

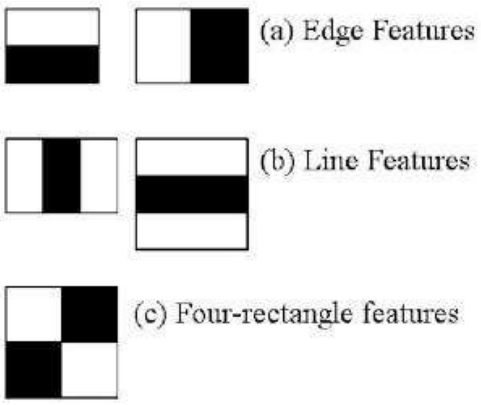

Рисунок 2.1 – Примитивы Хаара

2.2.2 Гистограмма ориентированных градиентов

HOG (гистограмма ориентированных градиентов) - это более новый метод создания функции обнаружения объектов: он начал использоваться с 2005 года. Он основан на вычислении градиентов на пикселях ваших изображений. Эти функции затем передаются в алгоритм машинного обучения, например, SVM. Он имеет лучшую точность, чем классификаторы Хаара. [4]

2.2.3 Сверточные нейронные сети

Сверточные нейронные сети (Convolutional Neural Networks, CNN) широко применяются в задаче детекции лиц. Они обладают способностью автоматически извлекать и выделять характеристики изображений, что делает их эффективными инструментами для обнаружения объектов, в том числе лиц. [5]

2.2.4 Многозадачные сверточные нейронные сети

MTCNN (Multi-task Cascaded Convolutional Networks) - метод, использующий сверточные нейронные сети для обнаружения лиц и

ключевых точек. Он состоит из нескольких этапов, включая обнаружение грубой рамки лица, регрессию границ лица и точек ключей (рисунок 2.2). MTCNN обеспечивает хорошую точность и работает с лицами разных масштабов и ориентаций. [6]

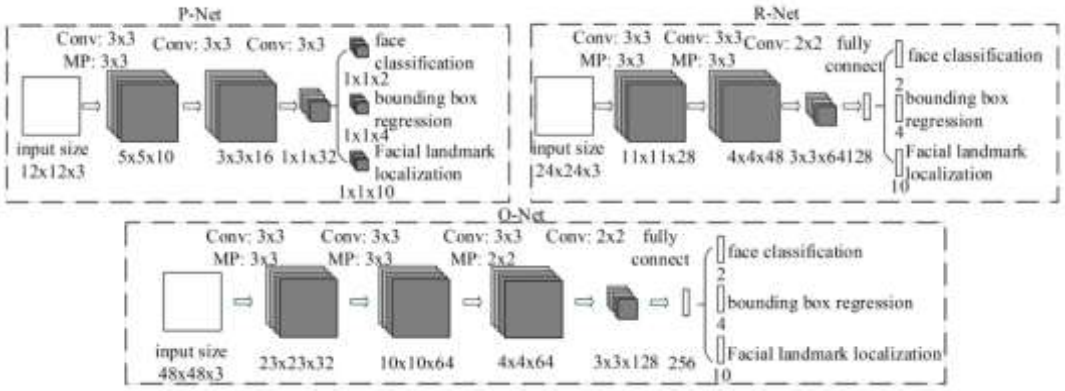

Рисунок 2.2 – Этапы обнаружения лица сетью MTCNN

2.2.5 Вывод по выбору модели обнаружения

Все эти методы используются для обнаружения лиц. Они имеют различные преимущества и ограничения в отношении точности, скорости и эффективности работы. Выбор конкретного метода зависит от требований и ограничений задачи детекции лиц.

Каскады Хаара на данный момент уже являются устаревшим методом детекции, даже с учетом их производительности применение более точных методов можно считать оправданным.

В данном проекте предлагается использовать HOG – детектор, он имеет хорошую точность и высокую производительность. При желании можно использовать любую другую модель из перечисленных.

2.3 Выбор модели извлечения признаков

В качестве модели для извлечения признаков целесообразно использовать сверточную модель нейронной сети.

Плюсы сверточных сетей:

– Сверточные сети способны автоматически извлекать и выделять значимые признаки из изображений без необходимости ручного определения признаков. Это позволяет сети обучаться на более абстрактных уровнях информации и достигать высокой точности в задачах обработки изображений;

– Сверточные сети обладают свойством инвариантности к некоторым искажениям, таким как перенос, поворот и изменение масштаба изображений. Это делает их устойчивыми к небольшим вариациям в данных и позволяет эффективно работать с различными изображениями;

– Сверточные сети могут обобщать свои знания и обученные признаки на новые данные. Они могут распознавать объекты или классифицировать изображения, которые не были представлены во время обучения. Это делает их гибкими и применимыми к различным задачам;

– Сверточные сети могут эффективно обрабатывать данные параллельно, благодаря использованию сверточных операций. Это позволяет достичь высокой скорости обработки и реализовать сверточные сети в реальном времени.

Ограничения сверточных сетей:

– Сверточные сети могут быть сложными моделями с множеством параметров, что затрудняет интерпретацию и понимание, каким образом они делают предсказания. Это может быть проблемой в некоторых задачах, где важно понять причины принимаемых решений;

– Обучение сверточных сетей требует большого объема размеченных данных для достижения хорошей производительности. Получение и разметка такого объема данных могут быть сложными и затратными задачами;

– Обучение и использование сверточных сетей требуют значительных вычислительных ресурсов, особенно для глубоких моделей с

большим количеством параметров. Это может ограничивать применимость сверточных сетей в ресурсоемких средах или на устройствах с ограниченными вычислительными возможностями.

Плюсы сверточных сетей очень сильно привлекают, а с ограничениями придется работать. Большой сложностью сверточных сетей является обучение. Для правильной тренировки необходимо использовать миллионы изображений, и это занимает много времени даже с десятками дорогих графических ускорителей. На практике такое большое количество данных порой недоступно, что уж говорить о десятках дорогих ускорителей.

Решением является использование трансферного обучения. Такой метод не требует тренировки сети на больших данных и тем самым очень полезны. Предварительно обученная нейросеть уже содержит знания, полученных из множества данных и опыта обучения, имеет способность обобщать. [5]

Рассмотрим варианты предобученных моделей.

2.3.1 Модель VGG16

VGG16 - это глубокая сверточная нейронная сеть, разработанная исследователями из Visual Geometry Group при Университете Оксфорда. Она была представлена в 2014 году и стала одной из наиболее известных архитектур сверточных сетей.

Архитектура VGG16 состоит из 16 слоев, включая 13 сверточных слоев и 3 полносвязанных слоя. Все сверточные слои имеют фильтры размером 3x3 пикселя с функцией активации ReLU, а максимальное объединение (max pooling) выполняется с фильтром размером 2x2 пикселя и шагом 2. Такая архитектура с малыми фильтрами и максимальным объединением помогает сети эффективно извлекать признаки изображения различных масштабов. Изображена она на рисунке 2.3.

VGG16 имеет большое количество параметров - около 138 миллионов. Это делает модель глубокой и способной выучивать сложные зависимости в данных. Однако, также требуется больше вычислительных ресурсов для обучения и использования VGG16, по сравнению с более легкими моделями. [7]

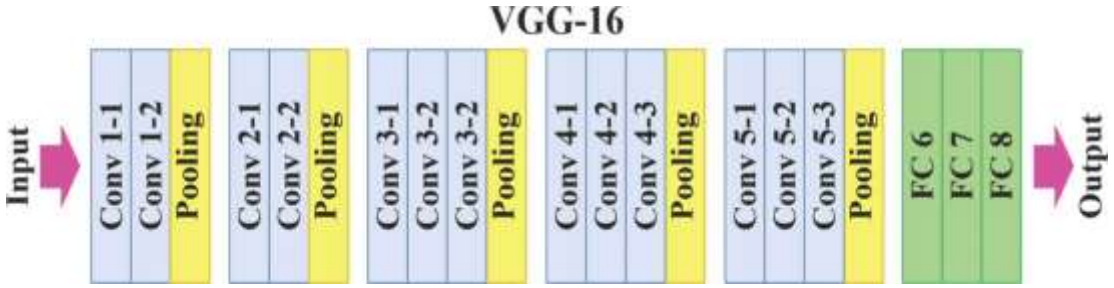

Рисунок 2.3 – Архитектура VGG-16

Данная сеть была обучена на специальном наборе данных с лицами, включая такие базы данных, как Labeled Faces in the Wild (LFW), YouTube Faces DB (и другие) и получила название VGGFace.

Модель VGGFace обучается на большом количестве изображений лиц, представленных различными людьми и в различных условиях освещения, позы и фонов. Это позволяет модели извлекать общие и различающиеся признаки лиц для идентификации.

Одной из особенностей VGGFace является ее способность генерировать эмбеддинги лиц. Это то, что и нужно для проекта. После прохождения изображения через модель VGGFace, получается вектор эмбеддинга, который представляет уникальные характеристики лица. Эти эмбеддинги могут быть использованы для сравнения и идентификации лиц.

VGGFace показывает высокую точность в задаче распознавания лиц, особенно на известных базах данных, таких как LFW. Однако, поскольку VGGFace обучена на наборе данных, который может быть ограничен по количеству и разнообразию лиц, ее производительность может снижаться на неизвестных данных или в условиях, отличных от обучающих.

VGGFace и его вариации стали основой для многих других моделей распознавания лиц и нашли применение в различных областях, включая

автоматическую идентификацию лиц, системы видеонаблюдения, аутентификацию по лицу и многое другое. [8]

2.3.2 Модель ResNet50

Использование архитектуры ResNet-50 вместо VGG16 в задаче распознавания лиц приводит к модели, известной как VGGFace на основе ResNet-50. Эта модель комбинирует преимущества двух популярных архитектур и становится более мощным инструментом для идентификации лиц.

ResNet-50 - это глубокая сверточная нейронная сеть, которая изначально была разработана для классификации изображений. Ее особенностью является использование блоков с пропусками (skip connections), которые позволяют сети обучаться на глубоких уровнях без потери градиентов. Это помогает решить проблему затухания градиентов и улучшает производительность модели. [9] Архитектура сети представлена на рисунке 2.4.

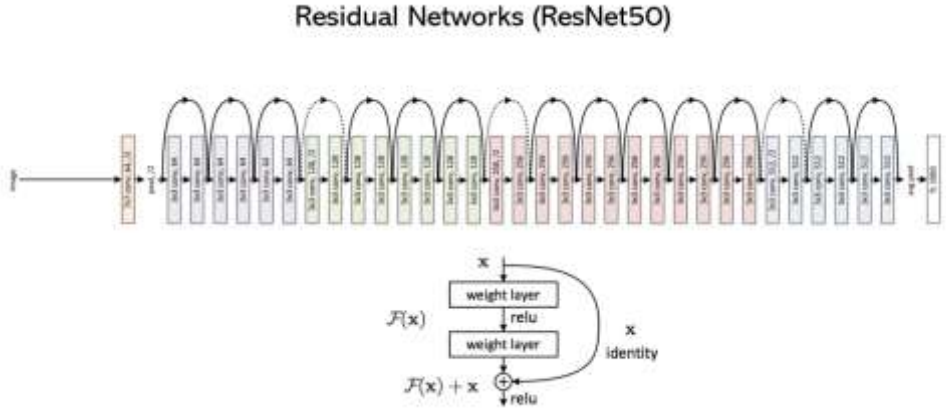

Рисунок 2.4 – Архитектура ResNet50

VGGFace на основе ResNet-50 использует архитектуру ResNet-50 для извлечения признаков из лиц. Она имеет сверточные слои, объединения и полносвязанные слои, которые позволяют модели изучать различные уровни абстракции признаков лиц.

После прохождения изображения через VGGFace на основе ResNet-50, получается вектор эмбеддинга лица, который представляет уникальные характеристики лица. Эти эмбеддинги могут быть использованы для сравнения и идентификации лиц.

Преимущество VGGFace на основе ResNet-50 заключается в том, что она имеет более глубокую архитектуру, чем VGGFace на основе VGG16, что может привести к лучшей способности модели изучать сложные зависимости в данных. Кроме того, ResNet-50 имеет меньшее количество параметров, чем VGG16, что может облегчить обучение и использование модели.

2.3.3 Вывод по выбору модели извлечения признаков

В данном проекте предлагается использовать VGG16, хотя ResNet-50 по многих составляющим лучше. Забегая наперед, VGG16 имеет меньший размер эмбеддинга, чем ResNet-50, что требует меньших вычислительных ресурсов для хранения и обработки данных, поэтому эта модель была выбрана для использования.

2.4 Выбор модели классификации

После создания эмбеддингов лиц идет этап классификации, где нейросеть сопоставляет эмбеддинги лица с известными лицами в базе данных для определения личности. Здесь используются алгоритмы классификации, такие как метод ближайших соседей (k-nearest neighbors), метод опорных векторов (Support Vector Machines, SVM), глубокие нейронные сети и другие.

k-Nearest Neighbors (k-NN) и Support Vector Machine (SVM) - это два популярных метода машинного обучения, используемых для классификации данных, включая классификацию лиц.

2.4.1 Метод k ближайших соседей

k-NN - это простой и интуитивно понятный метод классификации. Он основывается на идее, что близкие объекты в пространстве признаков обычно имеют схожие метки классов. В k-NN объект классифицируется путем определения его класса на основе меток классов его k ближайших соседей. Значение k - это параметр, определяющий количество ближайших соседей, которые используются для классификации.

Процесс работы k-NN следующий:

– Вычисление расстояний;

– Выбор k ближайших соседей;

– Голосование или взвешенное голосование.

Плюсы k-NN:

– Простота реализации и понимания;

– Не требует предварительного обучения;

– Гибкость в выборе метрики расстояния;

– Хорошо работает в случаях, когда классы локально сгруппированы.

Минусы k-NN:

– Вычислительная сложность, особенно с увеличением размера обучающей выборки;

– Чувствительность к выбору параметра k;

– Требует хранения всего обучающего набора данных. [10]

2.4.2 Метод опорных векторов

Support Vector Machine (SVM) - это метод бинарной классификации, который стремится найти гиперплоскость в многомерном пространстве признаков, которая максимально разделяет два класса. Он основывается на

концепции опорных векторов, которые являются объектами, находящимися на границе раздела классов.

Процесс работы SVM следующий:

– Построение опорной гиперплоскости;

– Преобразование в пространство большей размерности;

– Классификация новых объектов.

Плюсы SVM:

– Эффективность в работе с высокоразмерными данными и сложными структурами;

– Гибкость в использовании ядерных функций для обработки нелинейных данных;

– Стабильность в работе с выбросами и шумом;

– Позволяет контролировать переобучение с помощью параметра регуляризации.

Минусы SVM:

– Относительно высокая вычислительная сложность, особенно при использовании ядерных функций;

– Требует настройки гиперпараметров, таких как тип ядра и его параметры;

– При работе с большими наборами данных может потребоваться большое количество памяти. [11]

2.4.3 Вывод по выбору модели классификации

Эмбеддинги лиц обладают большой размерностью, поэтому предлагается использовать метод опорных векторов. К тому же метод опорных векторов обладает большей гибкостью в настройке параметров, например, выбор ядерной функции и настройка поверхностности решений.

2.5 Архитектура системы моделей

Окончательная архитектура системы является таковой:

– Обнаружение лица с помощью HOG-детектора;

– Выравнивание лица на основе ключевых точек;

– Извлечение признаков с помощью предобученной сверточной нейросети VGG16;

– Классификация лица по эмбеддингу с помощью SVM.

2.6 Алгоритм применения модели

Алгоритм применения модели для идентификации человека таков:

– Формирование множеств фотографий для тренировки и тестирования;

– Для каждой фотографии выполняется:

  – Обнаружение лица;

  – Выравнивание лица;

  – Извлечение признаков лица в виде эмбеддинга;

– Каждый эмбеддинг нормализуется, наименование кодируется;

– Классификатор обучается на тренировочной выборке;

– Классификатор предсказывает идентификаторы на тестовой выборке.

Это полный цикл работы системы. Здесь включена предобработка исходного множества фотографий, и дальнейшая работа системы по идентификации людей на незнакомых ранее фотографиях.

3 Программная реализация модели

3.1 Обоснование выбора средств разработки

Для разработки программного средства был взят за основу язык программирования Python [12] и среда разработки DataSpell [13].

Python является популярным и широко используемым языком программирования для разработки нейронных сетей. Python имеет огромную и разнообразную экосистему библиотек и инструментов для машинного обучения и разработки нейронных сетей:

– Для создания моделей глубокого обучения используются фреймворки TensorFlow [14] и Keras [15];

– Для детекции и выравнивания лиц используются инструменты библиотеки Dlib [16];

– Для методов машинного обучения используется библиотека scikit-learn [17].

3.2 Описание программной реализации

В данной главе будут описана программная реализация основных этапов задачи идентификации. Будут приведены листинги кода. Тривиальный код вынесен в приложение.

3.3 Загрузка изображения

Напишем функцию для загрузки изображения и преобразования в привычную цветовую схему RGB. Загрузим изображение.

```
import cv2
def load_image(path):
    return cv2.cvtColor(cv2.imread(path), cv2.COLOR_BGR2RGB)
```

```
image = utils.load_image("human.jpg")
```

3.4 Обнаружение

Импортируем пакет dlib, объявим функцию для детекции лиц. Выполним детекцию лиц на ранее загруженном изображении.

```
import dlib
detector = dlib.get_frontal_face_detector()
def detect_faces(img):
    return detector(img, 1)

faces = detection.detect_faces(image)
```

3.5 Выравнивание

Загрузим модель для выравнивания лиц, объявим функции для выравнивания одного и нескольких лиц на изображении. Выровняем ранее обнаруженные лица.

```
import dlib
sp =
dlib.shape_predictor('shape_predictor_5_face_landmarks.dat')
def aligned_faces(img, faces):
    with_landmarks = dlib.full_object_detections()
    for detection in faces:
        with_landmarks.append(sp(img, detection))
    return dlib.get_face_chips(img, with_landmarks, size=224)
def aligned_face(img, face):
    return dlib.get_face_chip(img, sp(img, face), size=224)

aligned_face = alignment.aligned_face(image, faces[0])
```

3.6 Извлечение признаков

Объявим функцию для создания модели извлечения признаков. Приведем изображение к входному формату модели. Получим эмбеддинги лица.

```
from keras_vggface.vggface import VGGFace
def make_cnn_model(model='vgg16'):
    return VGGFace(include_top=False, model=model,
input_shape=(224, 224, 3), pooling='avg')

model = make_cnn_model()
img = preprocess_input(aligned_face.astype(float), version=1)
encoding = model.predict(np.array([img]))[0]
```

3.7 Идентификация

Загрузим исходный датасет. Код тривиален, поэтому листинг в приложении.

```
X_train, y_train = [], []
X_test, y_test = [], []
```

Нормализуем эмбеддинги с помощью l2 нормой, и закодируем наименования в идентификаторы.

```
from sklearn.preprocessing import Normalizer, LabelEncoder
in_encoder = Normalizer(norm='l2')
embeddings = in_encoder.transform(X_train)
out_encoder = LabelEncoder()
labels = out_encoder.fit_transform(y_train)
```

Обучение классификатора. Гиперпараметр C подбирается вручную.

```
clf = svm.SVC(kernel='rbf', probability=True, C=15)
clf.fit(embeddings, labels)
```

Предсказание идентификатора личности по эмбеддингу. Гиперпараметр svc_threshold подбирается исходя из конкретной задачи идентификации.

```
svc_threshold = 20
are_matches = [proba[pred] * 100 <= svc_threshold for proba,
pred in zip(clf.predict_proba(embeddings), preds)]
res = [out_encoder.inverse_transform([pred])[0] if rec else
"unknown" for pred, rec in
zip(clf.predict(predicted_embeddings), are_matches)]
```

3.8 Добавление нового лица

Загрузим новые изображения. Первое будет примером для классификатора, второй незнакомое для классификатора.

```
img = utils.load_image('Thomas-Shelby-Train.jpg')
faces = detection.detect_faces(img)
if len(faces) != 1:
    raise Exception('was skipped because of len(faces)=%s' %
len(faces))
face_image = alignment.aligned_face(img, faces[0])
face_image = preprocess_input(face_image.astype(float),
version=1)
train_encoding = model.predict(np.array([face_image]))
train_encoding = in_encoder.transform(train_encoding)

img = utils.load_image('Thomas-Shelby-Test.jpg')
faces = detection.detect_faces(img)
if len(faces) != 1:
    raise Exception('was skipped because of len(faces)=%s' %
len(faces))
face_image = alignment.aligned_face(img, faces[0])
face_image = preprocess_input(face_image.astype(float),
version=1)
test_encoding = model.predict(np.array([face_image]))
test_encoding = in_encoder.transform(test_encoding)
```

Заново обучим классификатор, включая лицо нового человека.

```
labels = out_encoder.fit_transform([*y_train, 'thomas_shelby'])
embeddings = in_encoder.transform([*X_train, *train_encoding])
clf.fit(embeddings, labels)
```

Получим результаты.

```
preds = clf.predict(test_encoding)
print("svc:", out_encoder.inverse_transform(preds))
```

3.9 Вывод по главе

В данной главе были рассмотрены модули программной реализации системы идентификации. Основной код реализации показан, полный код находится в приложении. Каждый этап работы модели детально описан.

4. Тестирование модели

4.1 Описание процесса тестирования

Давайте смоделируем вариант использования системы идентификации человека по лицу. Попробуем внедрить ее в небольшую компанию размером в 30 человек, где необходим пропускной режим в офис.

4.2 Условия тестирования и формирование датасета

Подготовим тренировочный датасет для идентификации каждого сотрудника. Реализуем это так:

– Подготовим список всех сотрудников;

– Сделаем 3 фотографии каждого сотрудника;

– Сформируем базу данных с эмбеддингами для каждого сотрудника;

– На основе этой базы данных будет обучен классификатор.

Подготовим тестовый датасет для идентификации каждого сотрудника. В этом датасете от каждого сотрудника будет по 1 фотографии, которой классификатор ранее не обрабатывал.

– К началу рабочего дня все сотрудники прибывают к пропускному пункту и делают фотографию для идентификации;

– По новой для системы фотографии лица сотрудника будет сформирован эмбеддинг;

– По эмбеддингу сотрудник будет идентифицирован системой;

– Также пригласим 20 человек вне штата;

– По их эмбеддингам система ограничит проход.

Директории train и test содержат директории с именами сотрудников, в каждой директории фотографии сотрудника. Функция get_embeddings()

(листинг в приложении) возвращает сформированные эмбеддинги и имя сотрудника. Далее сохраняем базу данных с эмбеддингами.

```
X_train, y_train = get_embeddings("images_30_20/train")
X_test, y_test = get_embeddings("images_30_20/test")
np.savez_compressed("npz_30_20/train",
X=np.array(X_train), y=np.array(y_train))
np.savez_compressed("npz_30_20/test", X=np.array(X_test),
y=np.array(y_test))
```

4.3 Проведение тестирования

Для подготовленного датасета с эмбеддингами каждого сотрудника и системы моделей проведем тестирование:

Загрузим датасет

```
train = np.load("npz_30_20/train.npz")
X_train_raw, y_train_raw = train['X'], train['y']
test = np.load('npz_30_20/test.npz')
X_test_raw, y_test_raw = test['X'], test['y']
```

Нормализуем эмбеддинги и именования

```
in_encoder = Normalizer(norm='l2')
X_train = in_encoder.transform(X_train_raw)
X_test = in_encoder.transform(X_test_raw)
out_encoder = LabelEncoder()
y_train = out_encoder.fit_transform(y_train_raw)
```

Функция для оценки точности

```
def score(model, threshold, *, verbose=False):
    mistakes = []
    for (x, y) in zip(X_test, y_test_raw):
        probas = model.predict_proba([x])[0]
        pred = probas.argmax()
        proba = probas[pred]
        are_matches = proba * 100 >= threshold
        label = out_encoder.inverse_transform([pred])[0] if
are_matches else "unknown"
        if verbose:
            print("actually:", y, "predicted:", label,
"proba:", proba)
        if label != y:
            mistakes.append([y, label, proba])
    print("mistake count", len(mistakes))
    return mistakes
```

Обучим классификатор и получим точность

```
clf = svm.SVC(kernel='rbf', probability=True, C=50,
random_state=10)
clf.fit(X_train, y_train)
score(clf, threshold=11.5, verbose=True)
```

Результат оценки точности системы таков: система идентифицирует каждого человека и отделяет сотрудников от людей извне безошибочно.

4.4 Интерпретация результатов тестирования

По результатам тестирования на примере небольшой компании система идентификации человека по лицу работает безошибочно, но это не гарантирует такой же точности при увеличении количества идентифицируемых людей.

4.5 Предложения по доработке

Точность идентификации сильно зависит от выбранных порогов идентификации. Иногда разумно их завысить и получить некоторое количество ложноположительные результаты, а в дальнейшем обработать их вручную, в другой же ситуации выгоднее занизить порог, чтобы не идентифицировать неизвестного человека сотрудником компании. Необходимо тщательно следить за каждым случаем возникновения ошибки в системе идентификации, и принимать соответствующие меры по доработке системы. Предлагается настраивать пороги активации под конкретную ситуацию, а также строго контролировать входные фотографии для тренировки и использования классификатора. По возможности можно требовать от сотрудника предъявления документов, если система идентификации не уверена в точности свои результатов идентификации.

Заключение

В ходе выполнения данной работы была выбрана и реализована модель для идентификации человека по лицу, которая позволяет автоматизировать процесс идентификации человека и может быть использована в различных практико-ориентированных приложениях.

Модель, построенная на HOG-детекторе, распознавателе VGG16 и классификаторе SVM может считаться baseline моделью. Её можно использовать для пропускной системы или системы прокторинга. Текущее состояние системы позволяет использовать её совместно с существующей традиционными способами пропускного контроля.

Получены и закреплены навыки работы с моделями: выбор, реализация, обучение, тестирование и возможность применения для решения практических задач.

Кроме этого, отработаны навыки формирования датасета и оценки возможности его использования для обучения моделей.

Перечень использованных информационных ресурсов

1. Как работает распознавание лиц? Разбор [Электронный ресурс], URL: https://habr.com/ru/companies/droider/articles/568764/ (дата обращения: 25.05.2023 г.).
2. Большой Брат для новичков: как работают системы распознавания лиц [Электронный ресурс], URL: https://habr.com/ru/companies/netologyru/articles/707566/ (дата обращения: 25.05.2023 г.).
3. Признаки Хаара [Электронный ресурс], URL: https://ru.wikipedia.org/wiki/Признаки_Хаара (дата обращения: 25.05.2023 г.).
4. Гистограмма направленных градиентов [Электронный ресурс], URL: https://ru.wikipedia.org/wiki/Гистограмма_направленных_градиентов (дата обращения: 25.05.2023 г.).
5. Сверточная нейронная сеть [Электронный ресурс], URL: https://ru.wikipedia.org/wiki/Свёрточная_нейронная_сеть (дата обращения: 25.05.2023 г.).
6. MTCNN [Электронный ресурс], URL: https://github.com/ipazc/mtcnn (дата обращения: 25.05.2023 г.).
7. Модель VGG16[Электронный ресурс], URL: https://neurohive.io/ru/vidy-nejrosetej/vgg16-model/ (дата обращения: 25.05.2023 г.).
8. VGGFace [Электронный ресурс], URL: https://github.com/rcmalli/keras-vggface (дата обращения: 25.05.2023 г.).

9. ResNet [Электронный ресурс], URL: https://en.wikipedia.org/wiki/Residual_neural_network (дата обращения: 25.05.2023 г.).
10. Метод k-ближайших соседей [Электронный ресурс], URL: https://ru.wikipedia.org/wiki/Метод_k-ближайших_соседей (дата обращения: 25.05.2023 г.).
11. Метод опорных векторов [Электронный ресурс], URL: https://ru.wikipedia.org/wiki/Метод_опорных_векторов (дата обращения: 25.05.2023 г.).
12. Язык программирования Python [Электронный ресурс], URL: https://www.python.org/ (дата обращения: 25.05.2023 г.).
13. Среда разработки DataSpell [Электронный ресурс], URL: https://www.jetbrains.com/ru-ru/dataspell/ (дата обращения: 25.05.2023 г.).
14. Фреймворк Tensorflow [Электронный ресурс], URL: https://www.tensorflow.org/?hl=ru (дата обращения: 25.05.2023 г.).
15. Фреймворк Keras [Электронный ресурс], URL: https://keras.io/ (дата обращения: 25.05.2023 г.).
16. Библиотека Dlib [Электронный ресурс], URL: http://dlib.net/ (дата обращения: 25.05.2023 г.).
17. Пакет Scikit-learn [Электронный ресурс], URL: https://scikit-learn.org/stable/index.html (дата обращения: 25.05.2023 г.).

Приложение А Листинг программы

```
import pathlib

import dlib

sp = dlib.shape_predictor(str(pathlib.Path.cwd().parent /
'dg_face/shape_predictor_5_face_landmarks.dat'))


def aligned_faces(img, faces):
    with_landmarks = dlib.full_object_detections()
    for detection in faces:
        with_landmarks.append(sp(img, detection))
    return dlib.get_face_chips(img, with_landmarks, size=224)


def aligned_face(img, face):
    return dlib.get_face_chip(img, sp(img, face), size=224)

import dlib

detector = dlib.get_frontal_face_detector()


def detect_faces(img):
    return detector(img, 1)

from keras_vggface.vggface import VGGFace
from keras_vggface.utils import preprocess_input


def make_cnn_model(model='vgg16'):
    return VGGFace(include_top=False, model=model,
input_shape=(224, 224, 3), pooling='avg')


def face_encodings(image, faces):
    pass

import cv2


def load_image(path):
    return cv2.cvtColor(cv2.imread(path), cv2.COLOR_BGR2RGB)
```

```
from keras_vggface.utils import preprocess_input
from project.dg_face import utils, detection, alignment,
recognition
import matplotlib.pyplot as plt
import os
import numpy as np
from sklearn import svm
from sklearn.preprocessing import Normalizer, LabelEncoder
model = recognition.make_cnn_model()
def get_embeddings(base_path):
    X, y = [], []
    images = []
    for person in os.listdir(base_path):
        for person_img in os.listdir(base_path + "/" + person):
            path = base_path + '/' + person + '/' + person_img
            print(path)
            img = utils.load_image(path)
            images.append(img)
            faces = detection.detect_faces(img)
            if len(faces) != 1:
                print(path, 'was skipped because of
len(faces)=', len(faces))
                continue
            face_image = alignment.aligned_face(img, faces[0])
            face_image =
preprocess_input(face_image.astype(float), version=1)
            encoding = model.predict(np.array([face_image]))[0]
            X.append(encoding)
            y.append(person)
    return X, y, images

X_train_raw, y_train_raw, train_images =
get_embeddings("images_30_20/train")
X_test_raw, y_test_raw, test_images =
get_embeddings("images_30_20/test")
print(len(X_train_raw), len(X_test_raw))

in_encoder = Normalizer(norm='l2')
X_train = in_encoder.transform(X_train_raw)
X_test = in_encoder.transform(X_test_raw)
out_encoder = LabelEncoder()
y_train = out_encoder.fit_transform(y_train_raw)

clf = svm.SVC(kernel='rbf', probability=True, C=50,
random_state=10)
clf.fit(X_train, y_train)

threshold = 11.5

fig, axs = plt.subplots(nrows=len(X_test) // 5, ncols=5,
figsize=(20, 20))
i = 0
```

```
for x, y in zip(X_test, y_test_raw):
    probas = clf.predict_proba([x])[0]
    pred = probas.argmax()
    proba = probas[pred]
    are_matches = proba * 100 >= threshold
    label = out_encoder.inverse_transform([pred])[0] if
are_matches else "unknown"

    title = "{}({})".format(label, y)
    color = "red" if label != y else "green"
    ax = axs[i // 5, i % 5]
    ax.set_title(title, color=color)
    ax.imshow(test_images[i])
    ax.grid(None)
    ax.set_xticks([])
    ax.set_yticks([])
    i += 1
fig.tight_layout()
```