Лабораторная работа 4. Компьютерное зрение в биометрической аутентификации

С развитием компьютерных технологий растет и заинтересованность злоумышленников в овладении ценной информацией, в том числе открывающей доступ к реальным материальным ресурсам. И, таким образом, перед разработчиками биометрических систем защиты информации возникают разнообразные задачи, одна из которых – поддержка идентификации и аутентификации пользователя, то есть выяснения его личности и подтверждения подлинности его личности, соответственно, для дальнейшего предоставления ему определенных прав, или авторизации.

Следует отметить несколько особенностей при использовании биометрической аутентификации для обеспечения информационной безопасности. Во-первых, сам по себе процесс аутентификации по биометрическим показателям имеет сравнительно невысокую надёжность. В «перестраховочных» системах, где допускается значительная вероятность ложноотрицательного срабатывания, использование биометрической информации как единственного фактора может затруднить доступ реальным пользователям, поэтому в таких системах различные факторы разрешается делать взаимозаменяемыми – например, аутентифицировать пользователя либо по биометрическому показателю, либо по паролю. Такой подход улучшает пользовательский опыт, но он не является, в общем случае, многофакторной аутентификацией. Следует помнить, что объединение факторов аутентификации через логическое «ИЛИ» уменьшает защищенность системы в целом, поскольку это увеличивает возможную площадь атаки для потенциального злоумышленника, т.к. для доступа в систему достаточно провести успешную атаку на любой из используемых факторов. Во-вторых, в отличие от традиционных факторов аутентификации на основе секретной информации (паролей, ключей, токенов, пин-кодов и т.п.) биометрические показатели, как правило, не могут быть изменены. Это значительно снижает надёжность при использовании таких факторов в случае утечки информации – при хранении биометрических показателей в исходном виде они могут быть сфабрикованы, что, в свою очередь, компрометирует использование этого биометрического фактора конкретным пользователем во всех других системах, где он используется. В-третьих, показатели для многих биометрических факторов могут меняться с течением времени. Если система это учитывает и допускает некоторую вариативность, площадь атаки на такие системы также увеличивается, поскольку для успешной подделки показателя требуется меньшая степень схожести.

В основе функционирования биометрических систем лежит цепочка действий:

1. Запись – считываются биометрические данные пользователя;
2. Обработка данных – из представленных данных извлекается уникальная информация, например, изображение или вектор признаков;
3. Сравнение – сравнивается образ с эталонами из базы данных системы;
4. Принятие решения – решение об окончании процедуры идентификации, ее повторении или изменении условий ее проведения.

На данный момент существует и продолжает разрабатываться достаточно много подходов, основанных на получении различных биометрических образов:

- сканирование радужной оболочки глаза;
- анализ геометрии лица;
- снятие отпечатков пальцев;
- анализ геометрии руки;
- сканирование сетчатка глаза;
- получение отпечатка ладони;
- термограмма лица и руки;
- динамические:
- походка;
- получение характеристик речи;
- рукописная подпись;
- клавиатурный почерк;
- и т.д.

Практические примеры

Существует большое количество узкоспециализированных алгоритмов, направленных как на выделение признаков и построения эталонной модели для сравнения данных пользователя, так и на процесс самого сравнения и принятия решения об аутентификации личности. Однако в данной лабораторной работе предлагается рассмотреть подход, основанный на сверточных нейронных сетях, как наиболее универсальном средстве выделения признаков, применимом и адаптируемым под разные виды графических данных.

Рассмотрим пример биометрической аутентификации - по почерку. Датасет к этому примеру можно скачать по ссылке: https://www.kaggle.com/robinreni/signature-verification-dataset.

Датасет содержит набор изображений с 69 подписями, при этом для каждой подписи есть «подлинные» экземпляры, расположенные в каталогах с названием «№» и «подделанные» – в каталогах с названием «№_forg». Обратите внимание, что для реализации распознавания «подлинная/подделанная» по подписи для одного пользователя примеров слишком мало (порядка 10 изображений для каждого класса). Задачей в данном случае является обучение модели на распознавание «схожести» между образцами. Для этого можно сформировать датасет из всех возможных пар подписей, в которых одна подпись используется как образец, а вторая – как оцениваемый пример. Таким образом, для пар, в которых оба изображения являются подлинными образцами одной и той же подписи, будем ожидать на выходе значение «1», а если образец является подлинным, а пример – подделкой, будем ожидать значение «0». Обратите внимание, что при формировании датасета в парах друг с другом рассматриваются только экземпляры одной и той же подписи – оригиналы с оригиналами и оригиналы с подделками. Таким образом, задача сводится к бинарной классификации. При этом модель не обучается отличать заранее подготовленные подписи друг от друга, а вместо этого обучается определять, является ли произвольная подпись подлинной относительно предоставленного подлинного образца.

Функция, преобразовывающая входные данные в такой вид, может быть следующей:

```
import numpy as np
import cv2
from os import listdir

def load_data_from_folder(images_dir, height, width, depth):
    dir_names = listdir(images_dir)
    dir_count = len(dir_names)

    images_real = {}
    images_forgery = {}
    labels = set()

    for dir_name in dir_names:
        parts = dir_name.split('_')
        label_text = parts[0]
        is_forgery = len(parts) > 1

        if (label_text not in labels):
            labels.add(label_text)

        if label_text not in images_real:
            images_real[label_text] = []
```

```
            if label_text not in images_forgery:
                images_forgery[label_text] = []

            image_file_names = listdir(images_dir + '/' + dir_name)
            for image_file_name in image_file_names:
                image = cv2.imread(images_dir + '/' + dir_name + '/'
+ image_file_name, cv2.IMREAD_GRAYSCALE)
                if is_forgery:
                    images_forgery[label_text].append(image)
                else:
                    images_real[label_text].append(image)
        X_base_list = []
        X_comparison_list = []
        y_list = []

        for label in labels:
            real_samples = images_real[label]
            forged_samples = images_forgery[label]

            for real_img in real_samples:
                real_resized = cv2.resize(real_img, (width, height))
                for another_real in real_samples:
                    another_resized = cv2.resize(another_real,
(width, height))
                    X_base_list.append(real_resized.reshape((width,
height, depth)))

X_comparison_list.append(another_resized.reshape((width, height,
depth)))
                    y_list.append(1)

                for another_fake in forged_samples:
                    fake_resized = cv2.resize(another_fake, (width,
height))
                    X_base_list.append(real_resized.reshape((width,
height, depth)))

X_comparison_list.append(fake_resized.reshape((width, height, depth)))
                    y_list.append(0)

        X_base = np.array(X_base_list)
        X_comparison = np.array(X_comparison_list)
        y = np.array(y_list)

        return X_base, X_comparison, y
```

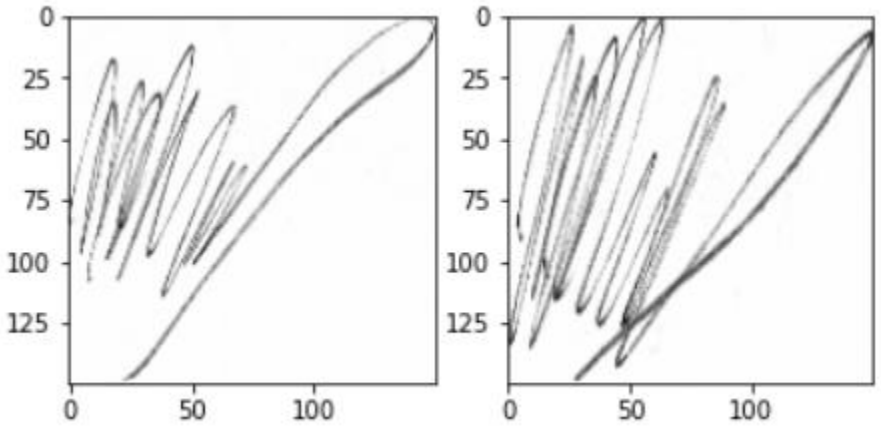
Кроме формирования всевозможных пар изображений, в данной функции они также приводятся к одному размеру с помощью функции `resize` модуля `cv2` библиотеки OpenCV. Чтобы сформировать датасет с помощью

этой функции, необходимо передать в нее путь к каталогу и размеры изображений (к которым необходимо привести входные изображения).

Вывести одну из пар изображений и ожидаемый результат можно при помощи следующих операций:

```
import matplotlib.pyplot as plt

index = 3
f, axes = plt.subplots(1,2)
axes[0].imshow(X_base[index, :, :, 0], cmap='gray')
axes[1].imshow(X_comparison[index, :, :, 0], cmap='gray')
print(f'Label: {y[real_index]}')
```



В отличие от большинства моделей, которые рассматривались до сих пор, задача сравнения двух изображений решается с использованием пары параллельных сверточных нейронных сетей с общими весами, результаты работы которых дополнительно подаются на слой активации, после чего для вычисления разницы между изображениями вектор активации с одной сети вычитается из вектора активации другой сети. Поскольку такая сеть имеет нелинейную структуру, она не может быть задана с помощью класса Sequential и вместо этого задаётся с использованием функционального API Keras на основании классов Input и Model. В функциональном API каждый слой модели может быть вызван как функция, аргументом которой является вход с предыдущего слоя. С помощью переменной feature в модели создаётся структура сверточной сети, которая затем формируется в 2 отдельные модели x1_net и x2_net с входами x1 и x2 соответственно, которые используют одну и ту же конфигурацию слоёв с общими весами. Результирующая сеть объединяет результаты работы параллельных моделей в слое Subtract, который вычисляет разность выходных векторов дублированной модели. Полученная разность дополнительно подаётся на сверточный слой и слой

субдискретизации, результат работы которых подаётся на полносвязный слой из 512 нейронов с функцией активации ReLU и, впоследствии, на 1 выходной сигмоидальный нейрон, который формирует вывод модели.

```
from tensorflow.keras import layers, losses, metrics, optimizers
from tensorflow.keras.models import Model

x1 = layers.Input(shape=(width, height, depth))
x2 = layers.Input(shape=(width, height, depth))

# параллельная модель
inputs = layers.Input(shape=(width, height, depth))
feature = layers.Conv2D(32, 3, activation='relu')(inputs)
feature = layers.MaxPooling2D(2)(feature)
feature = layers.Conv2D(64, 3, activation='relu')(feature)
feature = layers.MaxPooling2D(2)(feature)
feature = layers.Conv2D(128, 3, activation='relu')(feature)
feature = layers.MaxPooling2D(2)(feature)
feature_model = Model(inputs=inputs, outputs=feature)

x1_net = feature_model(x1)
x2_net = feature_model(x2)

net = layers.Subtract()([x1_net, x2_net])
net = layers.Conv2D(128, 3, activation='relu')(net)
net = layers.MaxPooling2D(2)(net)
net = layers.Flatten()(net)
net = layers.Dense(512, activation='relu')(net)
net = layers.Dense(1, activation='sigmoid')(net)

classifier = Model(inputs=[x1, x2], outputs=net)
classifier.compile(loss='binary_crossentropy',
optimizer=optimizers.RMSprop(), metrics=['accuracy'])
classifier.summary()
```

После создания модели необходимо ее обучить с использованием функции fit на датасете, сформированном из пар изображений. Для проверки корректности работы модели можно подавать на вход любые 2 изображения с подписями – подлинную подпись и образец, с которым нужно её сравнить, и проанализировать выход сети. При этом на выходе сети будет 1 число – результат последнего слоя. При анализе большого числа экземпляров можно подобрать корректное пороговое значение для отнесения результата к одному или другому классу.

Существует архитектура сверточных нейронных сетей, называемая «сиамской сетью» (Siamese), которая основана на схожей идее – использовании двух копий одной и той же сверточной подсети с общими

весами для формирования разности изображений. Архитектуру такой сети, а также особенности её обучения, рекомендуется изучить самостоятельно.

Использование сверточных слоёв может быть оптимизировано вычислениями на видеопроцессоре. В частности, TensorFlow предоставляет аппаратную поддержку работы с видеокартами NVIDIA. Если на вашем компьютере имеется видеопроцессор NVIDIA с поддержкой Cuda, можно установить соответствующие библиотеки и драйверы, как описано в инструкции: https://www.tensorflow.org/install/gpu.

Задание к лабораторной 4.

1. Создайте и обучите нейронную сеть для проверки подлинности произвольной рукописной подписи.
2. Сделайте образец своей подписи и попросите кого-нибудь (из одногруппников или родственников) сделать ложный ее образец.
3. Подайте полученные образцы на вход и проанализируйте результаты.
4. Найдите образцы дополнительные образцы подписей в интернете и на основании них подберите правильное пороговое значение для выхода сети.
5. Представьте ваше исследование в виде Блокнота JupyterLab, содержащего все пункты задания.