实验 5 基于深度学习的人脸识别技术

一、 目的与要求

1. 了解深度神经网络的基本结构；
2. 了解卷积神经网络的基本结构；
3. 能用编程语言实现基于深度学习的人脸识别技术。

二、 实验内容及步骤

1. 实验内容：

(1)人脸对齐：利用上一实验实现的 MTCNN 模型，实现对人脸区域的裁剪及对齐；

(2)人脸区域特征提取：使用深度神经模型 FaceNet 提取人脸特征，获取每个人脸的特征向量；

(3)人脸相似性度量与人脸识别：通过比较特征向量的相似性得到人脸相似性。相似性度量方式可以选择欧式距离(推荐)、余弦距离、和巴氏距离。设置阈值，将检测到的人脸与数据库的每张人脸照片计算相似度，得到识别结果并显示。

2. 实现基于 FaceNet 的人脸识别技术(仅供参考)

(1) 人脸对齐

人脸识别网络需要输入对齐后的人脸，这样能够降低学习难度，提高模型精度。因此，本实验在基于 MTCNN 检测人脸关键点的基础上，首先进行人脸对齐。

基于 MTCNN 获取的关键点使用仿射变换进行人脸对齐。通过模型检测出的 5 个关键点，向预先提供的 5 个参考点（正脸）对齐，使得对齐后的人脸关键点位置一致。

如下图所示。假设有一仿射变换矩阵，可以将五个源点变换到目标点，仿射关系由投影矩阵 M 表示，一般定义如下。t 表示平移变化，m 表示缩放旋转等变化。求解线性方程组得到近似的变化矩阵。得到五个点的投影关系后，将变换矩阵应用到原始图像上，就可以将图像对齐。求投影矩阵的方法，一般是通过最小二乘法得到线性方程组的近似解。

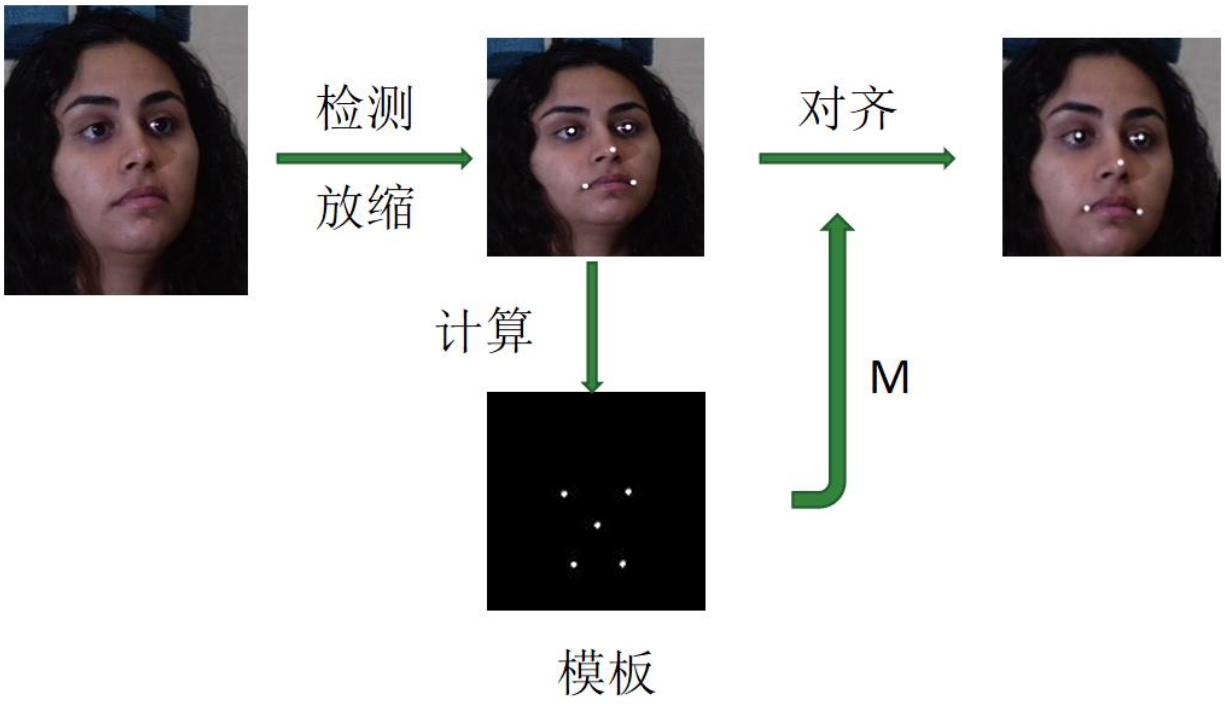
$$[x \quad y \quad 1] \times \begin{bmatrix} m_{11} & m_{12} & 0 \\ m_{21} & m_{22} & 0 \\ t_x & t_y & 1 \end{bmatrix} = [x' \quad y' \quad 1]$$

原点　　变化矩阵　　参考点

$$x' = m_{11}x + m_{21}y + t_x$$

$$y' = m_{12}x + m_{22}y + t_y$$

线性方程组



●目标

在 MTCNNDetector 中编写人脸对齐算法，输入原始图像以及人脸检测结果，将人脸对齐后保存到目标路径。仿射变换矩阵可以使用 opencv 库函数 estimateAffine2D 或项目中 getAffineMatrix 函数求得，得到变换矩阵后通过 opencv 库函数对原始图像进行变换。

●函数 API

```
void getAffineMatrix(double* src_5pts, const double* dst_5pts, double* M)
```

输入原始和目标五个关键点，计算变换矩阵保存到 M 中。

```
cv::warpAffine(src, dst, rot_mat, dst_sz)
```

输入原始图像 src 和变换矩阵 rot_mat，变换结果会放缩成 dst_sz 形状并保存到 dst 矩阵。

●示例

```
getAffineMatrix(enlargeFace_landmark, coord5point2, M);
cv::Mat warp_mat_112x112 = (cv::Mat_<float>(2, 3) << M[0], M[1], M[2], M[3], M[4], M[5]);
cv::Mat alignFace_112x112 = cv::Mat::zeros(112, 112, img.type());
//调用仿射变化函数，使用变化矩阵对齐人脸，
cv::warpAffine(enlargedFace, alignFace_112x112, warp_mat_112x112, alignFace_112x112.size())
cv::imwrite(savepath + "_align" + std::to_string(i) + ".jpg", alignFace_112x112);
```

(2) 人脸区域特征提取

使用预训练的人脸特征提取模型 FaceNet 来提取特征,需要注意的是 FaceNet 模型需要归一化后的 160*160 图像作为输入(通过 blobFromImage 设置)，模型的输出是 512 维的特征向量。

FaceNet 模型加载：下载并使用训练好的 FaceNet 模型。

链接：https://pan.baidu.com/s/1gm5xrlR8qGJXoCyX_1nTeg　提取码：4qod

●目标

使用 Opencv 深度学习模块加载预训练的 FaceNet，创建 Facenet 类，编写特征提取函数。将图像矩阵作为模型输入，进行前向传播得到特征向量。

●函数 API

```
_net = cv::dnn::readNetFromTensorflow(modelpb)
```

输入模型地址，加载并返回模型。

```
cv::Mat inputBlob = cv::dnn::blobFromImage(fimg, IMG_INV_STDDEV, cv::Size(160, 160),
    cv::Scalar(IMG_MEAN, IMG_MEAN, IMG_MEAN), false);
```

将图像矩阵转换为适合模型的输入，fimg 为输入图像矩阵，图像通过减去均值，除以标准差进行归一化。

```
_net.setInput(inputBlob);
```

设置模型输入

_net.forward(outputBlobs);

模型进行前向传播，并将结果保存到 outputBlobs 数组。

●示例

```
Facenet::Facenet(string modelpb, string modeltxt)
{
    _net = cv::dnn::readNetFromTensorflow(modelpb);
    if (_net.empty()) {
        cout << "facenet加载失败" << endl;
        throw std::invalid_argument("facenet loading error");
    }

}
```

```
cv::Mat Facenet::featureExtract(const cv::Mat& img)
{

    //图像转浮点，并标准化
    cv::Mat fimg = convertImg(img);

    //img = imgStandardization(img);
    //cv::Mat inputBlob =   cv::dnn::blobFromImage(img, 1.0, cv::Size(160,160));
    cv::Mat inputBlob = cv::dnn::blobFromImage(fimg, IMG_INV_STDDEV, cv::Size(160, 160),
        cv::Scalar(IMG_MEAN, IMG_MEAN, IMG_MEAN), false);

    //设置模型输入
    _net.setInput(inputBlob);
    //前向传播
    vector<cv::Mat> outputBlobs;
    _net.forward(outputBlobs);
    return outputBlobs[0].clone();
}
```

(3) 人脸相似性度量与人脸识别

通过比较特征向量的相似性得到人脸相似性。相似性度量方式可以选择欧式距离(推荐)、余弦距离或巴氏距离。事先准备一个人脸数据库./test/dataset，该数据库包含若干图像和对应的标签，将对应结果保存到文本文件中方便读入程序。当检测到人脸时，将提取到的特征与数据库中图像的特征计算相似性，当相似性大于阈值时就可以判断为同一人。将识别结果通过 opnecv 显示在图像上。

●目标

将摄像头获取的人脸的特征向量与数据库中人脸的特征向量一一进行距离

计算，如果最近距离小于阈值则代表识别成功。在图像上标定满足条件的人脸标签，不满足的人脸显示为 none。Camera 类通过调用 DrawRect 函数显示检测和识别结果，识别成功会以绿色显示，否则由红色显示。通过调用 cv::rectangle 和 cv::putText 将矩形框和标签画在原图上。关键代码如下图所示。

●函数 API

```
cv::norm(feat1 - feat2);
```

求两个特征向量的二范数并返回结果。

```
cv::putText(img, label[i], cv::Point(screct.tl().x, screct.tl().y),
    cv::FONT_HERSHEY_TRIPLEX, frntsize, color);
```

在图像上，以点 point 为左下角显示文本信息。

●示例

```
double Facenet::getSimilarity(const cv::Mat& feat1, const cv::Mat& feat2)
{
    //欧式距离
    return cv::norm(feat1 - feat2);
}
//寻找数据库中最相似的特征
string Facenet::faceRecognition(cv::Mat& feat, double threshold)
{
    int id = -1;
    double mins = 0.;
    for (int i = 0; i < _feat.size(); i++)
    {
        double simx = getSimilarity(_feat[i], feat);
        if (simx < mins || id == -1)
        {
            id = i;
            mins = simx;
        }
    }
    if (mins > threshold || id==-1)return "none";
    if (_nametolabel.count(_vimgname[id]) > 0)
        return _nametolabel[_vimgname[id]];

    return "none";
}
```

```
//画人脸矩形框和显示人脸标签
void Camera::DramRect(cv::Mat& img, vector<Face>& faces, vector<string>& label)
{
    for (int i = 0; i < faces.size(); i++)
    {

        //获取矩形框
        cv::Rect rect = faces[i].bbox.getSquare().getRect();
        cv::Rect screct(int(rect.tl().x), int(rect.tl().y ),
            int(rect.width), int(rect.height));

        cv::Scalar color;

        if (label[i] != "none")color = cv::Scalar(0, 255, 0);
        else color = cv::Scalar(0, 0, 255);

        cv::rectangle(img, screct, color);
        double frntsize = screct.width / 100.0;
        //显示标签
        cv::putText(img, label[i], cv::Point(screct.tl().x, screct.tl().y),
            cv::FONT_HERSHEY_TRIPLEX, frntsize, color);
    }
}
```

3.程序流程

程序执行时先使用 MTCNNDetector 检测并对齐数据库 dataset，将对齐的图像保存到 aligned 文件夹，然后由 Facenet 提取数据库对齐的图像特征和标签并保存在容器中，图像标签保存在 dataset/label.txt 中。Camera 类读取图像或视频，对读取到的图像或视频调用 MTCNNDetector 类进行人脸检测并对齐，使用 Facenet 类提取对齐人脸图像的特征。将提取到的特征与数据库特征比较相似性，如果最近距离小于阈值则显示对应标签，否则显示为 none。

4.实验任务

(1).调试运行代码，完成基于 FaceNet 的人脸识别技术，框出人脸区域，并显示对应标签。检测结果如下所示：

(2). 采集同宿舍同学的人脸数据，保存在 dataset 文件夹中并标注相应标签。通过摄像头、宿舍合照或录制视频，进行人脸检测及识别，并显示最终结果。通过截图及画流程图等方式，将上述过程写入实验报告。

(3). 修改程序，实现基于 FaceNet 的人脸验证技术，框出人脸区域，并显示是否为同一人。

(4). 实时视频的检测精度与处理速度难以平衡，优化程序，在保证速度的情况下提高识别准确率。比如，通过优化视频处理函数 videoShow 和通过交叉验证得到适合数据库的阈值。

参考代码：FaceRecognition.zip

——main.cpp 主函数，控制程序流程

——Camera.cpp 负责获取视频和图片显示检测和识别结果。

——detector.cpp 封装 MTCNN，检测人脸并对齐保存。

——Facenet.cpp 封装 facenet，提取人脸特征并识别

——rnet/pnet/onet.cpp MTCNN 子网络

——model 存放模型文件

——test 存放测试图片/视频和数据库

——face.h 人脸数据结构和锚点回归、NMS 算法

——utils.h 包含图像裁剪、通道转换和标准化算法

实验环境

操作系统：Windows10

CPU: AMD Ryzen 7 5800H with Radeon Graphics

内存：16 GB (金士顿 DDR4 3200MHz 8GB x 2)

显卡：NVIDIA GeForce RTX 3060 Laptop GPU (程序默认未调用 cuda)

第三方库：Opencv4.4.0 (必须 4.0 以上，不然模型无法加载)

IDE: Visual Studio2019

编程语言：C++