数字内容安全
实验报告

姓　　名　项 枫

学　　号　2022211570

指导教师　张 茹

学　　院　网络空间安全学院

2024 年 5 月 23 日

实验名称 图像差分隐私保护方案实验 实验日期 2024 年 5 月 23 日指导老师 张茹 得分 _

学院 网络空间安全学院 专业 信息安全 班次 2022211801 姓名 项枫 学号 2022211570

一、实验目的

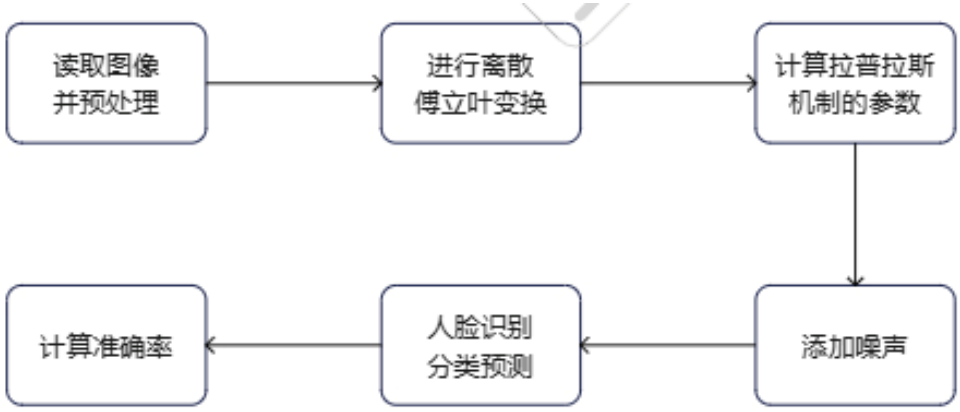
了解差分隐私技术的基本特点，设计并实现基于离散傅立叶变换（DFT）的图像差分隐私保护算法。了解差分隐私技术在数字内容保护中的作用，掌握基于差分隐私的内容隐私保护方法。

二、实验内容

本实验实现一种基于离散傅立叶变换的图像差分隐私保护算法,可通过隐私预算控制噪声规模，保证隐私安全性。

三、系统整体描述和分功能描述

系统整体描述



分功能描述

1）分功能 1：预处理
用到的函数：transform.resize(img, (128, 128))

2）分功能 2：离散傅里叶变换
用到的函数：np.fft.fft2(image)

3）分功能 3：计算拉普拉斯机制的参数
用到的函数：np.random.laplace(0, scale, 1)

4）分功能 4：添加噪声
用到的函数：add_laplace_noise(dft_matrix, epsilon, k)

5）分功能 5：人脸识别分类预测
用到的函数：dimensionality_reduction_PCA(n_components, X_train, 128, 128)；classification_svc(X_train_pca, y_train)

6）分功能 6：计算准确率
用到的函数：prediction(clf, X_test_pca)

四、实验步骤、结果及分析

实验步骤

1、读入一幅图像,对图像做预处理:如果读入的是彩色图像,将其转换为灰度图像(rgb2gray)；在灰度图像中利用差值方式将图像重采样为 128*128 的标准化图表示（imresize）IM;

2、对标准化图像 IM 进行离散傅立叶变换，得到离散傅立叶变换矩阵 FIM;

3、对离散傅立叶变换矩阵 FIM，选取其前 k×k 个 DFT 系数，计算给定隐私预算ε时的拉普拉斯机制的参数λ的最小值，以确定拉普拉斯机制需要添加的噪声；
4、对离散傅立叶变换矩阵 FIM，采样一组概率 p，在参数\lambda 最小时，计算相应的噪声值，以及融合噪声后的 FIM'；
5、对于 FIM 和 FIM'，分别输入 PCA+SVM 的人脸识别程序中进行人脸识别分类预测。
6、给出整个数据集上，人脸识别分类预测的准确率(Accuracy，测试集中分类器正确分类的样本数与总样本数之比)。
7、以 LFW 中随机 80%的图片为训练集，剩余为测试集，分析实验结果。
上述过程代码如下：
Image differential privacy protection.py

```
import numpy as np
from skimage import io, color, transform
import matplotlib.pyplot as plt

from main import dimensionality_reduction_PCA,
train_text_transform_Model, classification_svc, prediction,
print_report, \
    title, plot_images, fetch_data_details

from sklearn.decomposition import PCA
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.datasets import fetch_lfw_people

import warnings

# 忽略 DeprecationWarning 警告
warnings.filterwarnings('ignore', category=DeprecationWarning)

# 拉普拉斯
def Laplace(data, e, sensitivity): # 接受数据、隐私参数 epsilon 和数据的全局敏感度 sensitivity 作为输入
    scale = sensitivity / e
    noise = np.random.laplace(0, scale, 1) # 添加拉普拉斯噪声来保护数据隐私
    return data + noise # 返回添加噪声后的数据

# 傅里叶变换
def Fourier(image, k, e):
    F = np.fft.fft2(image)  # FIM
    F_k = F[:k, :k]     # 对频域表示的前 k×k 个系数应用拉普拉斯机制来保护隐私
    for i in range(k):
        for j in range(k):
            F_k[i][j] = Laplace(F_k[i][j], e, 1)
```

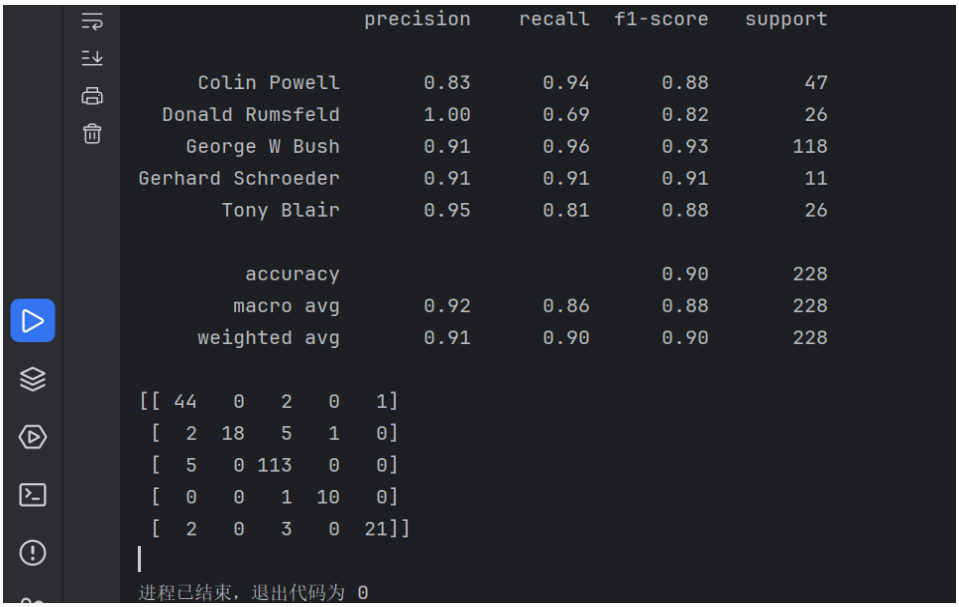
```
    F[:k, :k] = F_k

    new_image = np.fft.ifft2(F).real
    return new_image

# 调整图像尺寸为 128x128
def Resized_image(img, k, E):
    img_resized = transform.resize(img, (128, 128))
    if int(k) > 128:
        print("k is out of range")
        return
    image = Fourier(img_resized,int(k),float(E))
    return image

# 加载 LFW 数据集
database = fetch_lfw_people(min_faces_per_person=100)
n_samples, height, width, X, n_features, y, target_names, n_classes =
fetch_data_details(database)

# 设置 e、k
e = input("Please input e:")
k = input("Please input k:")

# 对图像进行预处理
X_processed = []
count = 0
for image in database.images:
        image = Resized_image(image, k, e)
        X_processed.append(image)
        count += 1
        print(f"{count}/1140")
print("All is ok.")

X_processed = np.array(X_processed)

# LFW 中随机 80%的图片为训练集，剩余为测试集
X_train, X_test, y_train, y_test = train_test_split(X_processed,
database.target, test_size=0.2)

X_train = X_train.reshape(X_train.shape[0], -1)
X_test = X_test.reshape(X_test.shape[0], -1)



# PCA
```

```
n_components = 150

pca, eigenfaces = dimensionality_reduction_PCA(n_components, X_train,
128, 128)

X_train_pca, X_test_pca = train_text_transform_Model(pca, X_train,
X_test)

# SVM
clf = classification_svc(X_train_pca, y_train)

# 评估
y_pred = prediction(clf, X_test_pca)

# printing classification report
print_report(y_test, y_pred, target_names, n_classes)

# 输出图像
prediction_titles = [title(y_pred, y_test, target_names, i)
                     for i in range(y_pred.shape[0])]

plot_images(X_test, prediction_titles, 128, 128)

# plot eigenfaces
titles = ["eigenface %d" % i for i in range(eigenfaces.shape[0])]
plot_images(eigenfaces, titles, 128, 128)
```

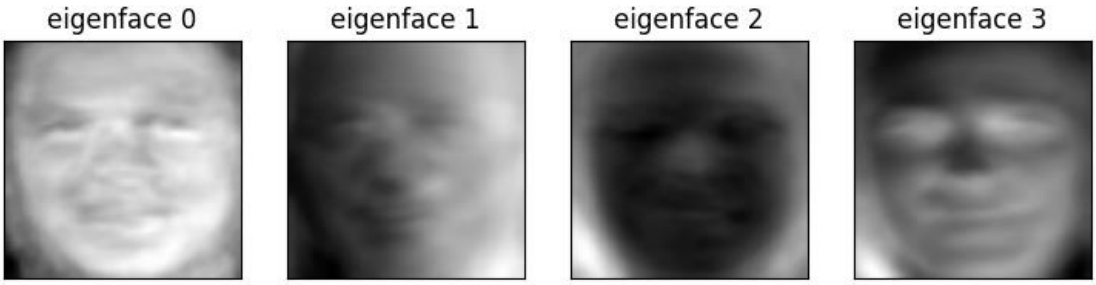
实验结果及分析

1、实验结果

以 LFW 中随机 80%的图片为训练集，剩余为测试集，实验结果如下：



```
                   precision    recall  f1-score   support

     Colin Powell       0.83      0.94      0.88        47
  Donald Rumsfeld       1.00      0.69      0.82        26
    George W Bush       0.91      0.96      0.93       118
Gerhard Schroeder       0.91      0.91      0.91        11
       Tony Blair       0.95      0.81      0.88        26

         accuracy                           0.90       228
        macro avg       0.92      0.86      0.88       228
     weighted avg       0.91      0.90      0.90       228

[[ 44   0   2   0   1]
 [  2  18   5   1   0]
 [  5   0 113   0   0]
 [  0   0   1  10   0]
 [  2   0   3   0  21]]

进程已结束，退出代码为 0
```



2、分析

（1）关于ε的选取，是一个经验值，其选取依据是根据测试集的准确率决定，给出选取过程。

（2）关于 k 的选取，k 值越大噪声越大,隐私安全性越强，但对人脸识别任务的鲁棒性会降低,因而需设定合适的 k 值，以满足隐私保护的人脸图像在识别精度和隐私性之间的折衷，给出选取过程。

对于上述两个问题，编写以下程序，计算出最佳ε和 k 的取值：

find best k&e.py

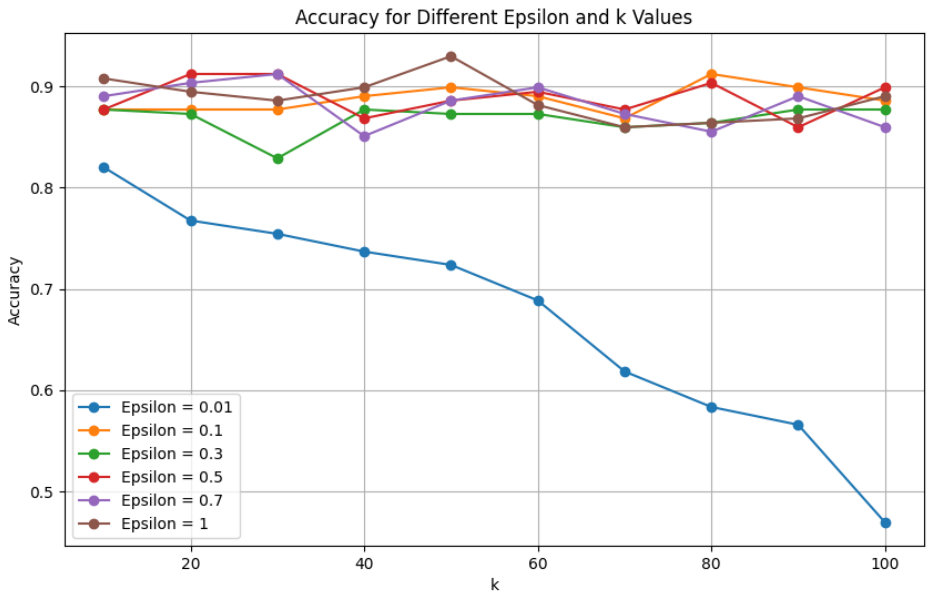
```
import numpy as np
from skimage import transform
from sklearn.decomposition import PCA
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.datasets import fetch_lfw_people
import matplotlib.pyplot as plt
from main import dimensionality_reduction_PCA,
train_text_transform_Model, classification_svc, prediction, \
    fetch_data_details
import warnings

# 忽略 DeprecationWarning 警告
warnings.filterwarnings('ignore', category=DeprecationWarning)

# 拉普拉斯
def Laplace(data, e, sensitivity): # 接受数据、隐私参数 epsilon 和数据的全
```

```
局敏感度 sensitivity 作为输入
    scale = sensitivity / e
    noise = np.random.laplace(0, scale, 1) # 添加拉普拉斯噪声来保护数据隐私
    return data + noise # 返回添加噪声后的数据

# 傅里叶变换
def Fourier(image, k, e):
    F = np.fft.fft2(image)  # FIM
    F_k = F[:k, :k]     # 对频域表示的前 k×k 个系数应用拉普拉斯机制来保护隐私
    for i in range(k):
        for j in range(k):
            F_k[i][j] = Laplace(F_k[i][j], e, 1)
    F[:k, :k] = F_k

    new_image = np.fft.ifft2(F).real
    return new_image

# 调整图像尺寸为 128x128
def Resized_image(img, k, E):
    img_resized = transform.resize(img, (128, 128))
    if int(k) > 128:
        print("k is out of range")
        return
    image = Fourier(img_resized,int(k),float(E))
    return image



# 加载 LFW 数据集
database = fetch_lfw_people(min_faces_per_person=100)
n_samples, height, width, X, n_features, y, target_names, n_classes =
fetch_data_details(database)

# 定义要尝试的参数组合
epsilons = [0.01,0.1,0.3,0.5,0.7,1]
ks = range(10,110,10)

# 存储结果
results = []

for epsilon in epsilons:
    for k in ks:
        print(f"Running with epsilon={epsilon} and k={k}")

        # 对图像进行预处理
```

```
        X_processed = []
        for image in database.images:
            image = Resized_image(image, k, epsilon)
            X_processed.append(image)
        X_processed = np.array(X_processed)

        # 划分训练集和测试集
        X_train, X_test, y_train, y_test =
train_test_split(X_processed, database.target, test_size=0.2)

        X_train = X_train.reshape(X_train.shape[0], -1)
        X_test = X_test.reshape(X_test.shape[0], -1)

        # compute PCA
        n_components = 150
        pca, eigenfaces = dimensionality_reduction_PCA(n_components,
X_train, 128, 128)
        X_train_pca, X_test_pca = train_text_transform_Model(pca,
X_train, X_test)

        # Training a SVM classification model
        clf = classification_svc(X_train_pca, y_train)

        # Quantitative evaluation of the model quality on the test set
        y_pred = prediction(clf, X_test_pca)

        # 计算精度
        accuracy = accuracy_score(y_test, y_pred)

        # 保存结果
        results.append((epsilon, k, accuracy))

# 输出结果表
print("Epsilon\tk\tAccuracy")
for result in results:
    print(f"{result[0]}\t{result[1]}\t{result[2]:.4f}")

# 画折线图
plt.figure(figsize=(10, 6))
for epsilon in epsilons:
    ks_values = [result[1] for result in results if result[0] ==
epsilon]
    accuracies = [result[2] for result in results if result[0] ==
epsilon]
```

```
    plt.plot(ks_values, accuracies, marker='o', label=f'Epsilon =
{epsilon}')

plt.xlabel('k')
plt.ylabel('Accuracy')
plt.title('Accuracy for Different Epsilon and k Values')
plt.legend()
plt.grid(True)
plt.show()
```

程序输出结果如下：



由图可知，当ε=1，k=50 时，准确率最高，为 0.9298。

五、验中遇到的问题及改正的方法

对 PCA+SVM 人脸识别知识了解过少，学习了相关的知识后，书写出了代码。