Spring 2024
Introduction to Artificial Intelligence

Homework 1: Face Detection

Introduction

When you take a picture using your smartphone, you might have seen that the application will tell you the locations of friends' faces and adjust the focus automatically. To enable this, the underlying technology is **face detection**.

What is face detection? Face detection is the process of detecting faces in images. The process involves **1) localization** and **2) recognition** of faces. Before continuing to read the rest of the description, please stop and start thinking about the following two questions:

1. How to place a bounding box where it potentially contains a face?
2. Given a bounding box containing a 2D array of pixel values (RGB), how to determine whether it is a face?
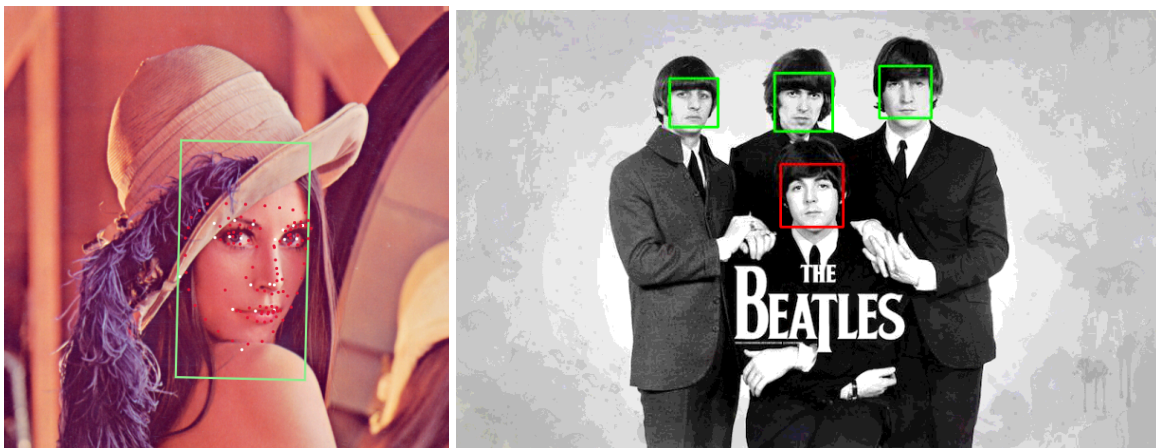
This homework focuses on **face recognition**, that is — whether the given area is a face. You will learn how to solve the fundamental problem in the field of computer vision. Out of many solutions, **Viola-Jones' algorithm**, a widely used algorithm through the years before neural networks took off, is chosen as a solution.

The goal of this programming assignment is:

1) Learn how to prepare a dataset for machine learning
2) Implement Viola-Jones face detection, especially the **Adaboost algorithm for feature selection**
3) Get a taste of a complete supervised learning process
4) Learn the limitations of Viola-Jones face detection

Please make sure you understand the concept of the **Viola-Jones algorithm** and the examples discussed in class before starting working on HW1.

The codebase includes most parts of the Viola-Jones face detection algorithm. The details of each file are listed in **Appendix A**. Please finish codes between # Begin your code and # End your code. For how to run Python code, please refer to **Appendix B** for more details.

Implementation (70%)

Part 1: Load and prepare your dataset (10%)

- You need to load two datasets under the "data" folder for training, namely "data_small" and "data_FDDB", respectively. For more details, please refer to **Appendix A**.
- Please feel free to import the packages that you need in dataset.py.
- Please run main.py to test your implementation. If your implementation is correct, one face image and one non-face image will be displayed. Please refer to **Appendix B** for more details.

Face   Non face
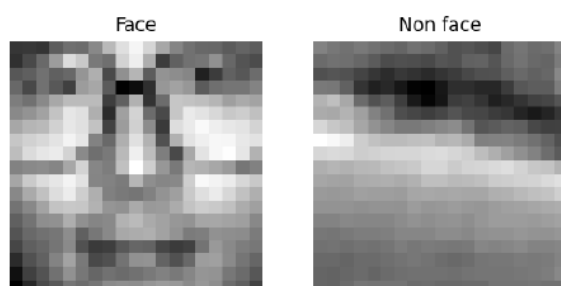
Part 1-1

- Implement the "load_data_small" function in dataset.py that loads all images in the folder. Please convert images into a list of tuples. The first element is a numpy array that stores an image. The shape of the numpy array is (height, width). The second element is its **Label** (1 or 0).

Part 1-2

- Implement the "load_data_FDDB" function in dataset.py. In this dataset, there is no "nonface" images data, you need to crop the original images into non-face images by yourself and preprocess them in the same way as the "data_small" dataset (convert images into a list of tuples).
  - Please note that there may be more than one face in each image
  - Please read the comments carefully
  - Please check the dataset and label structure from the FDDB README
  - Hints for implementation:
    - We have already created the `face_dataset` and saved the bounding box into `face_box_list` for you (in the "load_data_FDDB" function in dataset.py), where each element represents the left_top and right_bottom coordinate of the bounding box
    - Random crop rectangle regions from the original image that have small intersections with the face regions to create a non-face image
    - Feel free to add any other function to implement your idea!

Part 2: Implement Adaboost algorithm (30%)

- A strong classifier is a linear combination of weak classifiers.
- Please implement the "selectBest" function in adaboost.py to select the best weak classifier.
- Please run main.py to test your implementation. If your implementation is correct, you will see your classifier is being trained and evaluated.
- Please read the comments in the "selectBest" function, which can be found in adaboost.py line 147 and the "WeakClassifier" class in classifier.py.

Part 3: Additional experiments (10%)

- Please change the parameter T in the Adaboost algorithm. Please compare the corresponding detection performance.
- **Please test the parameter T between 1 to 10.**
- Please run main.py to test your implementation. If your implementation is correct, you will see your classifier is being trained and evaluated.
- Please read the comments in the "Adaboost" class, which can be found in adaboost.py line 13.
- In this homework, you need to train two classifiers using the "data_small" and "data_FDDB" datasets, respectively. Please refer to **Appendix B** for more training details.

Part 4: Detect face (15%)

- Please implement the "detect" function in detection.py to load the images in the "DETECT" folder. The function will detect faces we assigned.

- Please read detectData.txt to understand the format. Load the image and get the face images. Resize and convert the face images to 19 x 19 grayscale images. Then, please use clf.classify() function to detect faces.
- Display face detection results. If the classification is "Face," draw a green box on an image. Otherwise, draw a red box on an image.
- Please feel free to import packages that you need in detection.py.
- Run main.py to test your implementation.
- Please read the comments in the "classify" function, which can be found in adaboost.py line 169.

Part 5: Test classifier on your own images (5%)

- Please choose your own images with faces. Please create a text file with the same format as in detectData.txt.
- Please run main.py to check if your classifier can classify face and non-face!

Part 6: Implement another classifier (Bonus) (10%)

- You can implement the "selectBest" function in adaboost.py with another method. Please compare detection results with results obtained from Part 2 and Part 3.

Report (30%)

- A report is required, please follow the "report template" file.
- The report should be written in **English**.
- Please save the report as a **.pdf** file. (font size: 12)
- Answer the questions in the report template.

QA Page

If you have any questions about this homework, please ask them in the following Notion page. We will answer them as soon as possible. Additionally, we encourage you to answer other students' questions if you can.

https://lopsided-soursop-bec.notion.site/HW1-Question-Sheet-16273fe7d821442189b5c5a43187b185?pvs=4

Submission

Due Date: 3/18(Mon.) 23:59

Please directory compress all your code files and report (.pdf) into {STUDENT ID}_hw1.zip and **submit it to the New E3 System.**

The file structure should look like:

```
📂 {student_id}_hw1.zip
├─ 📄 main.py
├─ 📄 dataset.py
├─ 📄 feature.py
├─ 📄 classifier.py
├─ 📄 adaboost.py
├─ 📄 detection.py
├─ 📄 utils.py
├─ 📄 (other code files if you have)
└─ 📄 report.pdf
```

Please do not upload the dataset !!!

Wrong submission format leads to -10 point

Late submission leads to -20 points per day

Appendix A: Files & Dataset

1. Files:

File name	Description
main.py	Test all of code.
dataset.py	Load images.
feature.py	Define data structures about features for implementing the Adaboost algorithm.
classifier.py	Define data structures about weak classifiers for implementing the Adaboost algorithm.
adaboost.py	The main file that implements the Adaboost algorithm.
detection.py	Detection images.
utils.py	Contain functions to compute the integral image and evaluate classifiers.

2. Dataset:

The two datasets can be found in the "data" folder. There are three subfolders: "data_small", "data_FDDB" and "detect".

```
📂 data
├ 📂 data_small
├ 📂 data_FDDB
└ 📂 detect
```
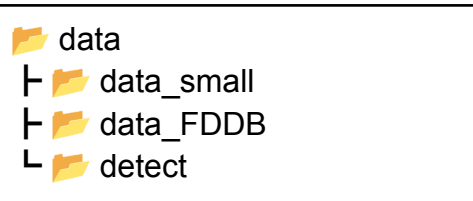
In the "data_small" folder, the "train" folder is for training. The "test" folder is for testing, the details are as follows:
- "train" and "test":
  - The two subfolders contain face images and non-face images, respectively.
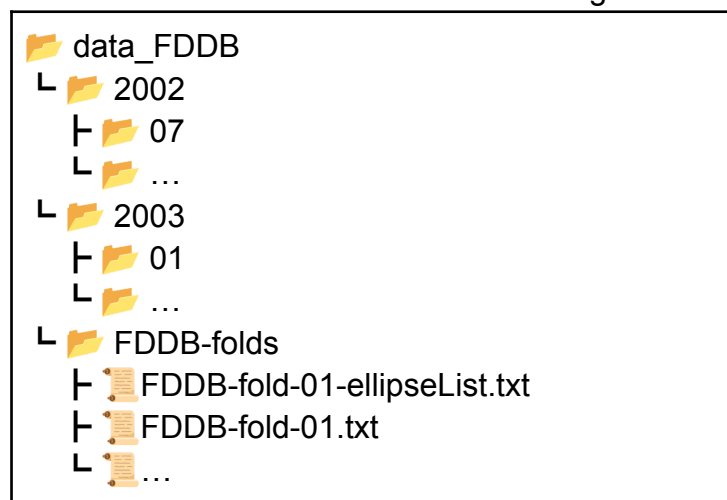 - The size of an image is 19 x 19, and the format is a PGM file.

In "data_FDDB" folder:
Please download the image and the annotations from the following link (1. Original, unannotated set of images, 2. Face annotations), unzip both of them, then place them under the "data_FDDB" folder. For more details on the database, please refer to the README file.
- Open the download links in the new tab if right click the link doesn't work

  Download the database:
  - Original, unannotated set of images. These are from the Faces in the Wild dataset, which come with the following restriction: "This dataset is for academic research purposes only".
  - Face annotations. The annotations are split into ten folds. See README for details.
  - README - information on directory structure and file formats

- "data_FDDB":
  - The file structure should look like the following:

```
📂 data_FDDB
└ 📂 2002
  ├ 📂 07
  └ 📂 …
└ 📂 2003
  ├ 📂 01
  └ 📂 …
└ 📂 FDDB-folds
  ├ 📄 FDDB-fold-01-ellipseList.txt
  ├ 📄 FDDB-fold-01.txt
  └ 📄 …
```

  - Note that we only use "fold-01" in our original implementation. Feel free to incorporate more data for training, however, please specify it in your report if there are any modifications.

In "detect" folder, it includes nature images, the details are as follows:
- "detect":

○ detectData.txt: The location and size of each face in a natural image. The format is:

```
image1_name face_num
x y width height
…
image2_name face_num
x y width height
```

Appendix B: How to run the code

Required packages:
- opencv-python
- scikit-learn
- numpy
- tqdm
- matplotlib

In part1, we mentioned that there are two datasets you need to load and train on, here we introduce how to run the code on the terminal.
1. Open the terminal and enter the "HW1" directory (the directory that has main.py)
2. Run the following commands

Load and train "data_small"

python main.py –data data_small

```
HW1$ python main.py --data small
```

Load and train "data_FDDB"

python main.py –data data_FDDB

```
HW1$ python main.py --data FDDB
```

Reference
- https://vteams.com/blog/how-to-run-a-python-script-in-terminal/
- https://docs.python.org/zh-tw/3/howto/argparse.html