

Spring 2022

Introduction to Artificial Intelligence

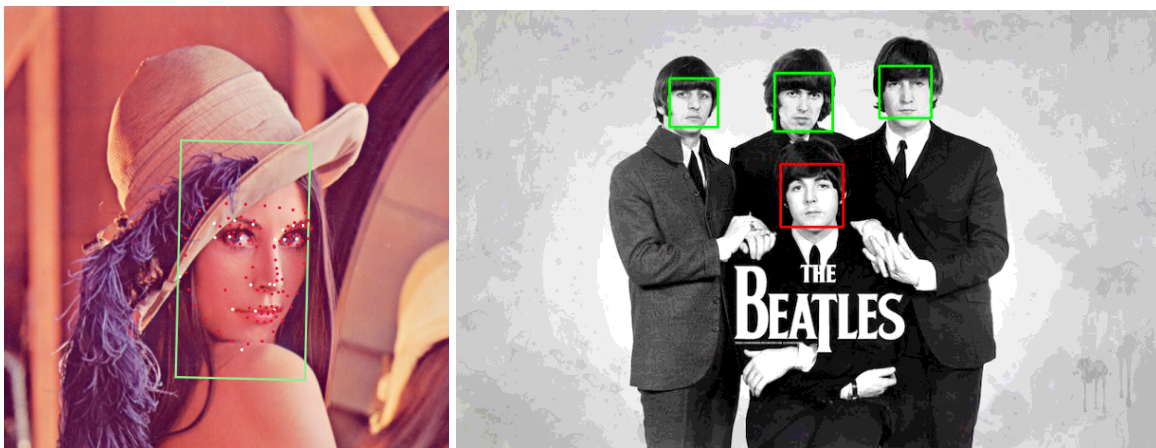
Homework 1: Face Detection

Introduction

When you take a picture using your smartphone, you might have seen that the application will tell you 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 reading 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 **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 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 the **Appendix A**. Please finish codes between `# Begin your code` and `# End your code`.

If you run Python code locally (Anaconda), please use [main.py](#) to test your implementation. For Google Colab users, please use [main.ipynb](#) and see **Appendix B** for more details.

Implementation (70%)

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

- Implement the “[loadImages](#)” 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).
- Please feel free to import packages implemented in [dataset.py](#).
- Please run [main.py](#) or [main.ipynb](#) to test your implementation. If your implementation is correct, one face image and one non-face image will be displayed.
- Please read the comments of the “[loadImages](#)” function in [dataset.py](#) line 4.

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](#) or [main.ipynb](#) 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 133 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](#) or [main.ipynb](#) 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 10.

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](#) or [main.ipynb](#) to test your implementation.
- Please read the comments in the “[classify](#)” function, which can be found in [adaboost.py](#) line 155.

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](#) or [main.ipynb](#) 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.
- The report should be written in **English**.
- Please save the report as a **.pdf** file. (font size: 12)
- Put the **screenshot of your code** and **explain it in detail** in the report.
- Put the **screenshots of the results** and **your analysis** in the report.
- Describe problems you encounter and how you solve them.
- Answer the questions in the report template.

Submission

Please prepare your source code and report (.pdf) into STUDENTID_hw1.zip.
e.g. 109123456_hw1.zip

Late Submission Policy

20% off per late day

Appendix A: Files & Dataset

1. Files:

File name	Description
main.py/main.ipynb	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 dataset can be found from the “data” folder. There are three subfolders. The “TRAIN” folder is for training. The “TEST” folder is for testing. The “DETECT” folder includes nature images. The details are as follow:

- 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 grayscale PGM.
- DETECT:
 - [detectData.txt](#): The location and size of each face in a natural image. The format is:

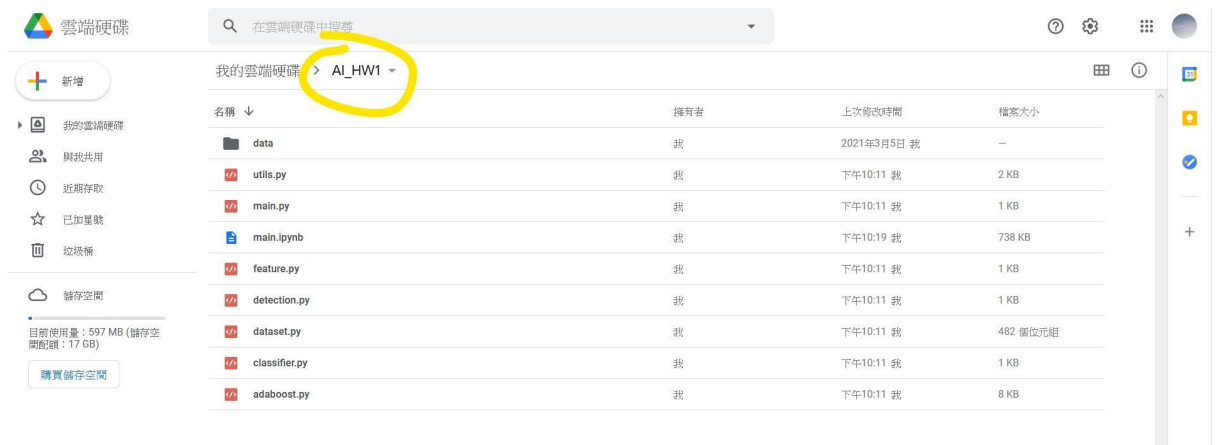
image1_name face_num

```
x y width height
...
image2_name face_num
x y width height
```

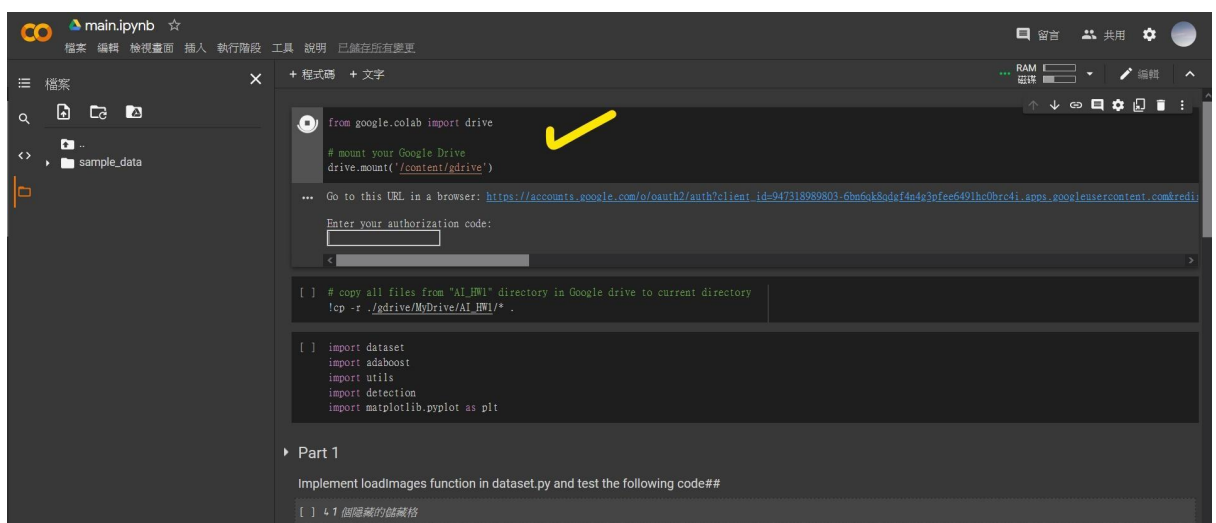
Appendix B: For using Google Colab

Since Google Colab has [usage limits](#) for users i.e. the files and codes you've uploaded and modified will be cleaned up after timeout, we recommend uploading project files to your Google Drive and use it as a storage of your work. Here are instructions for setting up Colab environment:

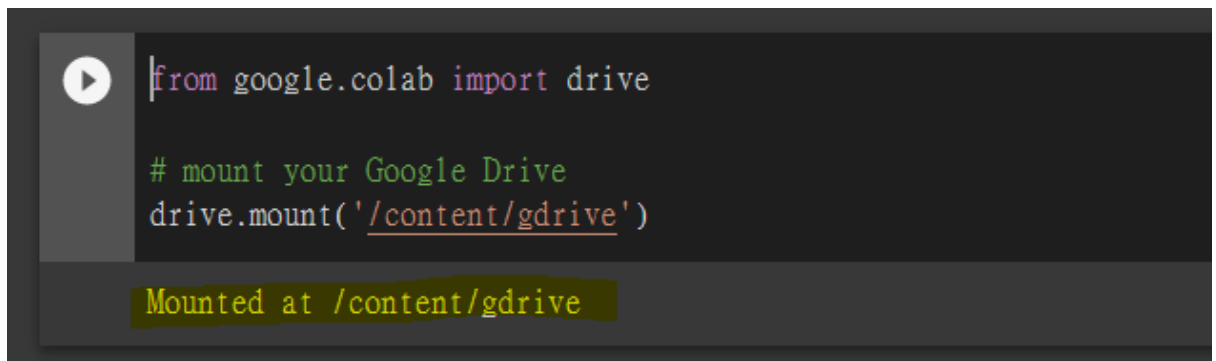
1. Upload the project file to your Google Drive, and name the folder: "AI_HW1"



2. Open the main.ipynb.
Execute the first cell to mount your Google Drive. Copy authorization code through the link and paste it, then press 'Enter'.



You will see the following messages if Google Drive is mounted successfully.

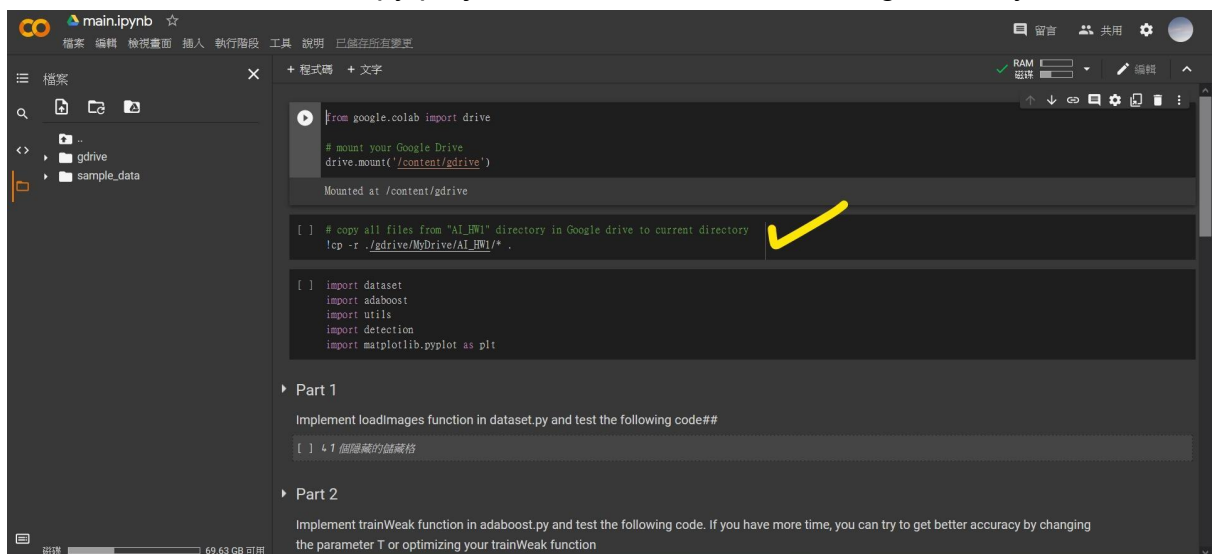


```
from google.colab import drive

# mount your Google Drive
drive.mount('/content/gdrive')

Mounted at /content/gdrive
```

3. Execute the next cell to copy project files to the current working directory.

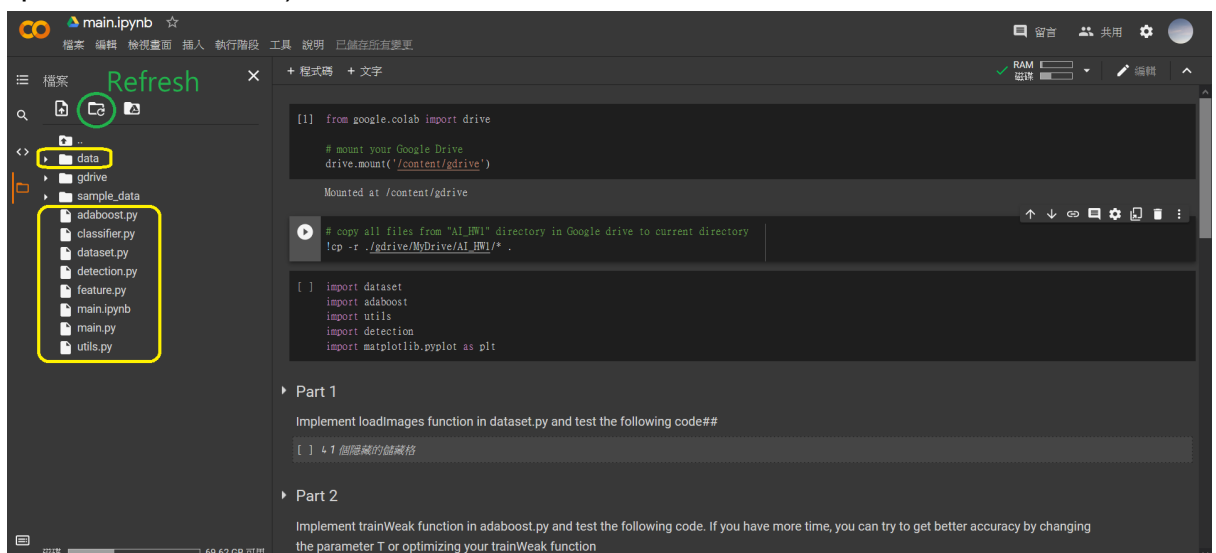


The screenshot shows the Colab interface with the following code cells:

```
[ ] # copy all files from "AI_HW1" directory in Google drive to current directory
!cp -r ./gdrive/MyDrive/AI_HW1/* .
```

A yellow checkmark is placed next to the second code cell. Below the code cells, the 'Part 1' and 'Part 2' sections are visible, along with a file explorer on the left showing the 'data' directory and its contents.

After copying, you will see project files on the left. (Press Refresh button to update file structure)



The screenshot shows the Colab interface with the following code cells:

```
[1] from google.colab import drive

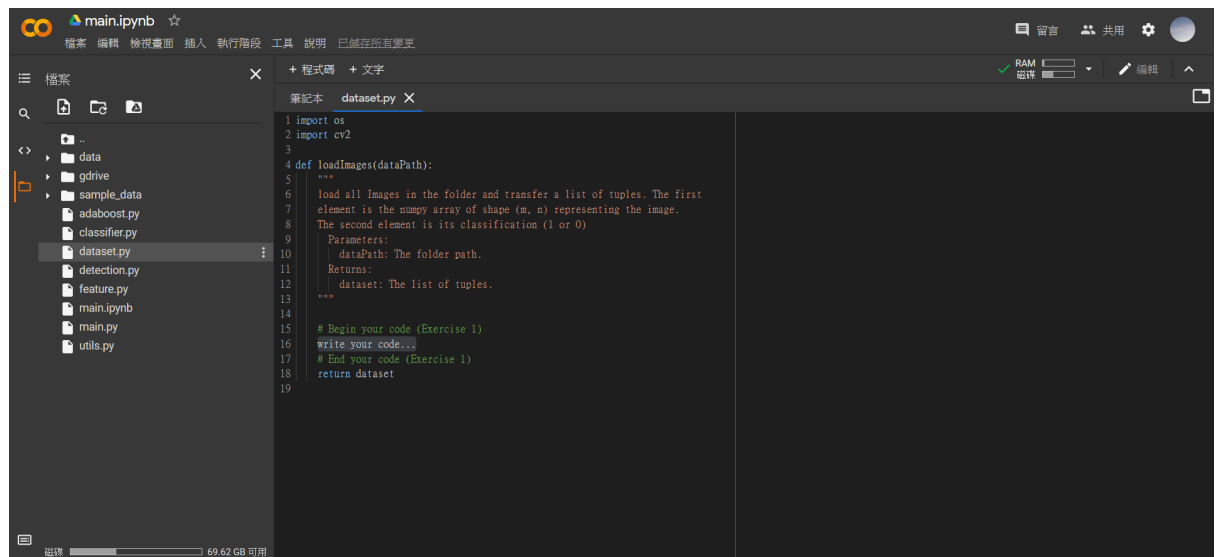
# mount your Google Drive
drive.mount('/content/gdrive')

Mounted at /content/gdrive

[ ] # copy all files from "AI_HW1" directory in Google drive to current directory
!cp -r ./gdrive/MyDrive/AI_HW1/* .
```

A green 'Refresh' button is highlighted in the top left corner. Below the code cells, the 'Part 1' and 'Part 2' sections are visible, along with a file explorer on the left showing the 'data' directory and its contents.

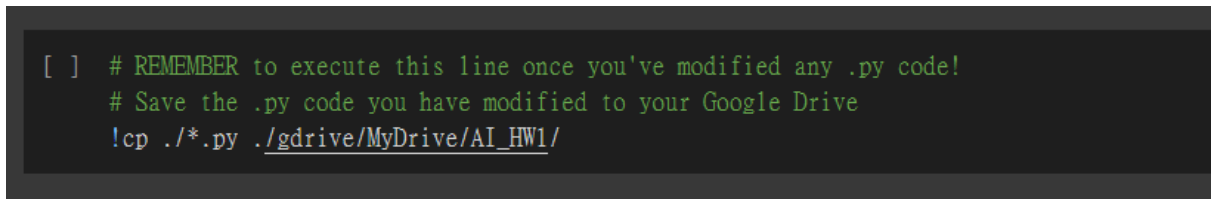
4. Write your code. (Press 'ctrl+s' in case auto-saving is not working)



```
1 import os
2 import cv2
3
4 def loadImages(dataPath):
5     ""
6     load all Images in the folder and transfer a list of tuples. The first
7     element is the numpy array of shape (m, n) representing the image.
8     The second element is its classification (1 or 0)
9     Parameters:
10     dataPath: The folder path.
11     Returns:
12     dataset: The list of tuples.
13     ""
14
15 # Begin your code (Exercise 1)
16 write your code...
17 # End your code (Exercise 1)
18 return dataset
19
```

5. *****IMPORTANT*****

Please REMEMBER to execute the last cell in order to copy .py files back to your Google Drive. If you forget to save your work back to your Google Drive, your code will be gone after Colab timeout! So make sure you execute this cell whenever you've done your current work.



```
[ ] # REMEMBER to execute this line once you've modified any .py code!
    # Save the .py code you have modified to your Google Drive
    !cp ./*.py ./gdrive/MyDrive/AI_HW1/
```