# **DPS920/CVI620 – Lab 8**

# **Evaluation Measures**

| Total Mark: | 10 marks (3% of the total course grade)   * 6 out of 10: Learn@Seneca submission (Due: Wednesday November 8th end of day) * 4 out of 10: Lab demo (Due: During Workshop of week 9) |
| --- | --- |
| Submission file(s): | * Lab08\_1.py / Lab08\_1.ipynb * Lab08\_2.py / Lab08\_2.ipynb * Lab08.docx |

Please work in **groups** to complete this lab. This lab is worth 3% of the total course grade and will be evaluated through your written submission, as well as the lab demo. During the lab demo, group members are *randomly* selected to explain the submitted solution. Group members who are not present during the lab demo will lose the demo mark.

Please submit the submission file(s) through Learn@Seneca. ALL team members must submit the final work.

***Please paste the resulting images and answers in this document.***

## **Part I: Saving Test Output**

Unzip Lab08.zip in C:, correct the paths in Lab08\_1.py if needed.

1. Describe what is being done in Lab08\_1.py here.

The code provided is a Python script that utilizes the dlib library's face detection capabilities to identify faces within images. Here's a step-by-step description of what the code does:

Imports necessary libraries: The script imports dlib for face detection, numpy for numerical operations, cv2 (OpenCV) for image processing, matplotlib.pyplot for displaying images, imutils for image resizing, and csv for reading and writing CSV files.

Set file paths: It sets the file paths for the images (path), a list of image filenames (list\_file), and a CSV file to store the detection results (alg\_file).

Initialize face detector: dlib.get\_frontal\_face\_detector() is used to create a face detector object.

Define helper functions:

rect\_to\_bb(rect): Converts a dlib rectangle object to a bounding box represented by the coordinates (x, y) and width and height (w, h).

detect\_face(image\_path): Reads an image from image\_path, resizes it, converts it to RGB, and then uses the dlib face detector to find faces in the image. If faces are detected, it draws bounding boxes around them, writes the bounding box coordinates to alg\_file, and displays the image. If no faces are detected, it returns "No Face Detected."

Face detection execution:

The script reads the list\_file CSV, which contains image filenames.

For each filename, it constructs the full path to the image and calls detect\_face to perform face detection on the image.

1. Run Lab08\_1.py to find the bounding box of faces in images.

(You may need to install imutils library in your Anaconda environment.)

A person on the phone

Description automatically generated

1. How many faces are there in the test set? Where are the detection results saved as? Was a face detected in all files? Explain.

There are 8 faces. (There are more parts of faces on pictures, however gt.csv has 8 rows, so we are trying to detect 8 faces.) The detection results are saved in file detections.csv. On one of files face was not detected. frontal\_face\_detector() did not return a rectangle coordinates, so we may conclude that face was not detected.

[Images obtained from <http://vis-www.cs.umass.edu/lfw/> ]

[Code adapted from [Emotion Detection: a Machine Learning Project | by Aarohi Gupta | Towards Data Science](https://towardsdatascience.com/emotion-detection-a-machine-learning-project-f7431f652b1f#:~:text=%20Emotion%20Detection%3A%20a%20Machine%20Learning%20Project%20,more%20layers%20of%20neurons.%20The%20input...%20More%20) ]

## **Part II: Calculation of Intersection over Union (IoU)**

In Lab08\_2.py:

* 1. Include the function intersection\_over\_union() from: <https://blog.paperspace.com/mean-average-precision/>
  2. Explain what this function is doing.

This function calculates the Intersection over Union (IoU) between two bounding boxes. IoU is the ratio of the area of overlap between the two bounding boxes to the area of their union.

inter\_box\_top\_left calculates the maximum top left coordinates of gt\_box and pred\_box which indicate the top left corner of intersection, inter\_box\_bottom\_right calculates the minimum coordinates which indicate the bottom right corner of intersection. Knowing the coordinates, we get intersection width and height. Then, we calculate intersection by multiplying width and height. Union is calculated by adding areas of both boxes and subtracting one intersection area. Iou is calculated by dividing intersection area over union area. Function returns iou, intersection, union in a tuple.

* 1. Write a code to match detections in detections.csv with ground truths in gt.csv and calculate IoU (for those belonging to the same file). Save the IoUs in a list.
  2. Print gt and prediction bounding boxes as well as IoU list. Paste the output here.

Make sure you read the bounding boxes as numbers and in format expected by the above function.

GT list:

[[190, 178, 253, 300],

[134, 145, 279, 313],

[164, 196, 244, 281],

[201, 240, 229, 210],

[148, 140, 246, 330],

[195, 117, 256, 358],

[166, 154, 230, 302],

[193, 156, 239, 300]]

Predictions list:

[[216, 217, 223, 223],

[167, 192, 223, 223],

'No Face Detected.',

[191, 217, 223, 223],

[191, 217, 223, 223],

[171, 171, 267, 268],

[191, 217, 223, 223],

[216, 217, 223, 223]]

IoU list:

[(0.6551910408432148, 49729, 75900),

(0.5694573270580691, 49729, 87327),

(0.7714735869899854, 42600, 55219),

(0.5285964502568893, 45269, 85640),

(0.6639885807504078, 65124, 98080),

(0.6221928845578028, 45715, 73474),

(0.6574848828162324, 48168, 73261)]

* 1. Add to the code; draw a red bounding box around predicted faces and a blue bounding box around ground truth and show the corresponding IoU on the image.

Paste two samples here.

A person and person smiling

Description automatically generated

A person wearing a hat and a blue hat

Description automatically generated

* 1. Print the average IoU over all detections and paste the output here.

Average IoU: 0.638340679038943

## **Part III: Precision, Recall, and F1 from IoU**

Add code to Lab08\_2.py to:

1. Write a function to return precision, recall, and F1 given 3 input parameters: An IoU list, number of face files (8 in our example), and a threshold. (To simplify we assume there exists exactly one face per file).
2. Print out the precision, recall, and F1 for various thresholds. Paste the output here.

Use:

for threshold in np.arange(0, 1, 0.1)

Threshold: 0.0

Precision: 0.88

Recall: 0.88

F1 Score: 0.88

Threshold: 0.1

Precision: 0.88

Recall: 0.88

F1 Score: 0.88

Threshold: 0.2

Precision: 0.88

Recall: 0.88

F1 Score: 0.88

Threshold: 0.30000000000000004

Precision: 0.88

Recall: 0.88

F1 Score: 0.88

Threshold: 0.4

Precision: 0.88

Recall: 0.88

F1 Score: 0.88

Threshold: 0.5

Precision: 0.88

Recall: 0.88

F1 Score: 0.88

Threshold: 0.6000000000000001

Precision: 0.62

Recall: 0.83

F1 Score: 0.71

Threshold: 0.7000000000000001

Precision: 0.12

Recall: 0.50

F1 Score: 0.20

Threshold: 0.8

Precision: 0.00

Recall: 0.00

F1 Score: 0.00

Threshold: 0.9

Precision: 0.00

Recall: 0.00

F1 Score: 0.00

1. Plot the precision-recall curve for this algorithm.

## A graph with a line going up Description automatically generated

## **Part V: Group Work**

1. Add this declaration to your file:

We, group 5, Liliya Panfilova and Davender Singh, declare that the attached assignment is our own work in accordance with the Seneca Academic Policy. We have not copied any part of this assignment, manually or electronically, from any other source including web sites, unless specified as references. We have not distributed our work to other students.

1. Specify what each member has done towards the completion of this work:

|  | Name | Task(s) |
| --- | --- | --- |
| 1 | Davender Singh | Part 1, Part 3 (c) |
| 2 | Liliya Panfilova | Part 2, Part 3 (a,b) |
| 3 |  |  |