German University in Cairo
Faculty of Media Engineering and Technology
Winter 2023

DMET 501 – Introduction to Media Engineering

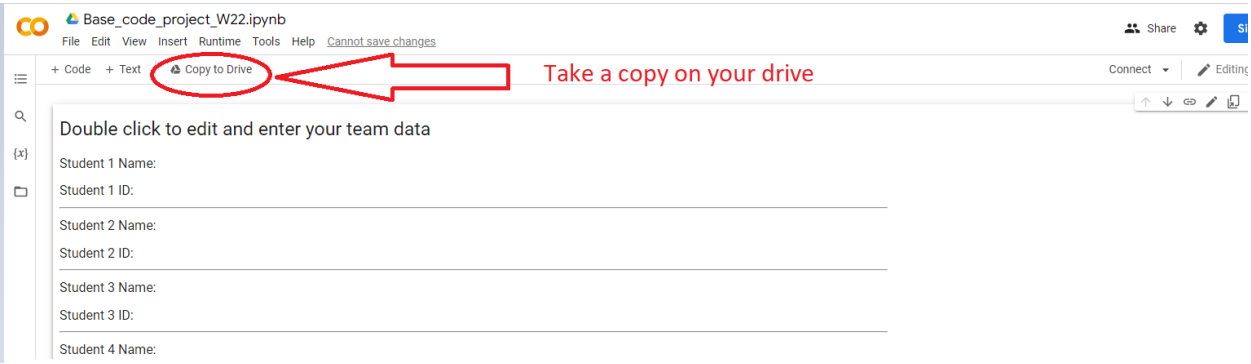
Assignment 3 & Final Project

(Due on December 20th, 2023 at 11:59PM)

- Read the **WHOLE** description **carefully** for all the requirements before starting.
- In this project you are going to implement **an encoding & decoding technique** (described in this document) on an image using **the Python notebook** given in the link below.
- You are required to take a **copy** of this notebook and write your own solution **in the stated parts** of the code and submit your code.

https://colab.research.google.com/drive/1MYD_BUzo9tkSZHjMF95nHeuHrWesp54s

KINDLY DO NOT CHANGE ANY FUNCTION SIGNATURE OR HELPER FUNCTIONS IN THE NOTEBOOK.



- In this code, you are only required to use **numpy** arrays in python and **opencv**(a framework that deals with images as 2D numpy arrays).
- You should write your function in the notebook cells provided, and you can test your code by running these test cells.

```
import cv2 # Opencv library is used to manipulate images.
import matplotlib.pyplot as plt # matplotlib used for plotting (showing) images.
import numpy as np # numpy is used to treat images as 2D arrays.
```

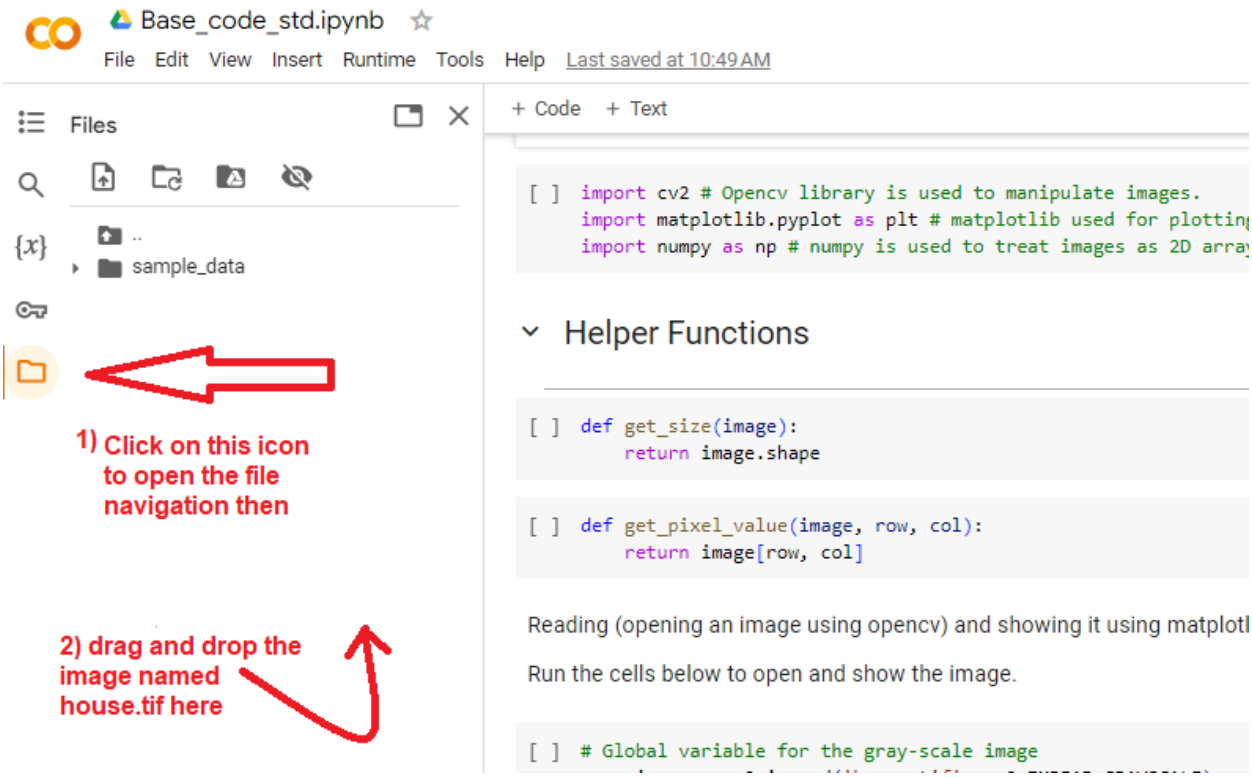
Hover on the cell to run and click this botton

- Also, at the end of the document you will find the description for the helper functions you can use in any of the tasks. Please check them out before writing your code. Also, you can use any predefined functions you like.
- Write your code in the **designated area** in each part of the base code.

```
def img_to_bin(gray_img):
  # start of your code here


  return
  # end of your code here
```

Write your code here between start and end.
Note that your code can have as many lines as you desire.



- **There will be private test cases so try to make your code as generic as possible as we are going to test it on other images and you have to stick to the input and output formats.**

Encoding Description

The encoding technique encodes the **input image** into **a string** separated by commas where each substring represents some part of the image in the format "`rowNumber_encoding1`" where:

$$encoding1 = (starting\ position\ of\ the\ continuous\ 1s) \times 1000 + (number\ of\ continuous\ 1s)$$

An example:

For an image

```
np.array([[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,0,0,1,1,1]])
```

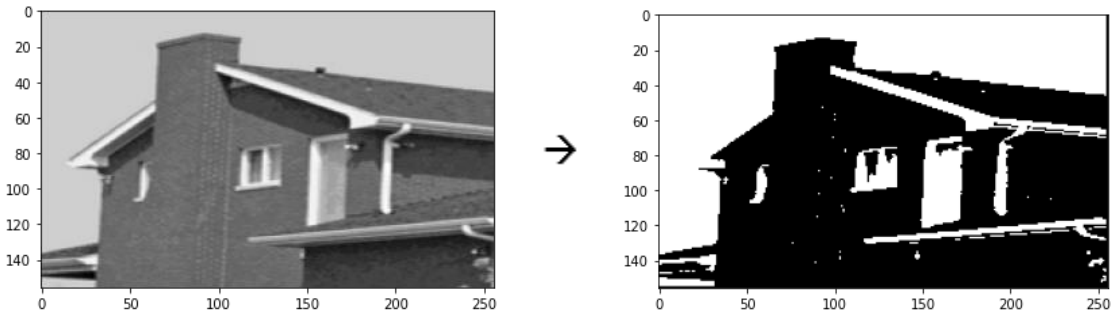
It is encoded as '1_7012, 1_21003'

This is because for row 1, bit 1 occurs continuously 12 times from location 7, then its encoded form is 1_(7 × 1000 + 12) = 1_7012

Part 1: Encoding

Task 1

In this task, you are required to convert the image from gray-scale to a binary one using **128 as a threshold**.



Function Signature: `def img_to_bin(gray_img):`

Input: The image you read using `cv2.imread('/content/house.tif')`, convert the image to a binary one according to the following equation:

$$binary(x,y) = \begin{cases} 0 & ,gray(x,y) < 128 \\ 1 & ,gray(x,y) \geq 128 \end{cases}$$

Expected Output: return the **binary image** representation **as a numpy array** (np.array()).

Task 2

In this task, you are required to get the starting index for each sequence of ones in a certain row number where:

Function Signature:

```
def get_indicies_of_starting_ones(twoD_array, row_index):
```

Inputs: `twoD_array:` A 2D array representing a generic input image,

`row_index:` the row number as an integer.

Expected Output: A list of the indices where a sequence of ones starts in a 1D array format.

Example:

For `twoD_array =`
`np.array([[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],`
`[0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,0,0,1,1,1]])`
and `row_index = 1`.

Output: [7, 21]

Task 3

In this task, you are required to get the length of the runs in a certain row number where:

Function Signature: `def get_length_of_runs(twoD_array, row_index):`

Inputs: `twoD_array:` A 2D array representing a generic input image,

`row_index:` the row number as an integer.

Expected Output: A list of the lengths of each consecutive sequence of ones in a single row in a 1D array format.

Example:

For `twoD_array =`
`np.array([[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],`
`[0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,0,0,1,1,1]])`
and `row_index = 1`.

Output: [12, 3]

Task 4

In this task, you are required to encode the image into a long string (**where encodings are ordered by row number**) using the equation described above where:

Function Signature: `def encoding_image(twoD_array):`

Input: `twoD_array`: A 2D array representing a generic input image.

Expected Output: returns a string representing the encoded binary image.

Example:

For `twoD_array =`

```
np.array([[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,0,0,1,1,1],
[1,1,1,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]])
```

Output: ’1_7012, 1_21003, 2_3, 2_4002’

Decoding Description

The decoding technique decodes the **encoded string** to get back the binary image as a 2D array again.

Part 2: Decoding

Task 1:

In this task, you are required to split the encoded string to get the substrings that have the same row number where:

Function Signature: `def split_string(CODE, row_index):`

Inputs: `CODE:` encoded image as a string,

`row_index`: the row number as an integer.

Expected Output: returns a list with the substrings that starts with a certain row number.

Example:

For encoded image as a string = `'1_7012, 1_21003, 2_255, 3_1014'` and `row_index = 1`

Output: ['1_7012', '1_21003']

Task 2:

In this task, you are required to decode a single row where:

Function Signature: `def decode_row(width, height, CODE, row_index):`

Inputs: `width:` The width of the image to be generated,

`height:` The height of the image to be generated,

`CODE:` encoded image as a string,

`row_index`: the row number as an integer.

Expected Output: Decoded row as a 1D array.

Example:

For an encoded string = `'1_7012, 1_21003'`

Output: [0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,0,0,1,1,1]

Task 3:

In this task, you are required to construct the binary image back from the string representing the encoded image where:

Function Signature: `def construct_image(width,height,encoding):`

Input: `width:` The width of the image to be generated,

`height:` The height of the image to be generated,

`encoding:` The string representing the encoded binary image.

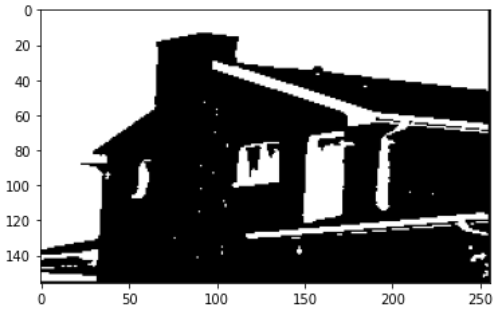
Expected Output: Decoded image as a 2D array (the reconstructed binary image as a 2D array).

Example:

For width = `256` , height = `156` and an encoded image as string `encoding` =
`'0_255,1_255,2_255,3_255,4_255,5_255,6_255,7_255,8_255,9_255,10_255,.....,149_253002,150_5,150_254001,151_24,151_252003,152_32,152_252001,153_32,154_32'`

Note: please don't use this string as it is an example, use the string that your code outputs.

Output: Binary image as shown below:



Helper Functions

These functions are to help you write your code, you are not obliged to use them.

```
def get_size(image):
```

Returns a tuple with format: (rows, columns).
Example Output: (156, 256)

```
def get_pixel_value(image,row,col):
```

Get the intensity of a single pixel.
Example Output *color at pixel (row=2, col=4) in img* : 205.

Best of luck! ☺

Submission guidelines

1. Please submit the project on the following form:

   https://forms.gle/hUDryeFqatSkkZQJ9

2. **The assignment can be done in teams of 1 to 4 members (Teams can constitute of members from cross-tutorials and different TAs).**
3. Please submit your notebook (.ipynb) in a zipped folder and don't clear the outputs from the notebook.
4. Rename the notebook/code file with tutorial number underscore the ID of one member [T-XX_55-XXXXX]. For example: [T-01_55-12345]
5. The name of the submitted zipped file is [T-XX_55-XXXXX]. Choose the ID number of 1 team member (the same one chosen in point 4).

   For example: [T-01_55-12345].zip

In case of not following the mentioned guidelines or editing the original notebook (function signatures/names/inputs/outputs) in the correct assigned positions, the project will not be graded.