Hi there, welcome to EN.520.612!

Welcome to the first lab of EN.520.612 “Machine learning for signal processing”.

In this course, we will use Google Colab for all labs. Our first lab is introduction and do some basic Image manipulation, Quantization as well as Aliasing Effect

Without further ado, let's dive right in... Please go to the following web page

https://colab.research.google.com/drive/1vsaPO7fKzPDCSt1OBZ56Gh5S4QvoDhEt?usp=sharing

What is Google Colab?

Colaboratory, or “Colab” for short, is a product from Google Research. It allows anyone to write and execute python code through the browser using Jupyter notebook style interface. As a bonus you also get

- Free access to GPUs
- Easy share option using Google Drive
- Easy run code on-cloud without slowing your local system

That's nice. So what should I know about Colab to succeed in EN.520.612 labs?

We have prepared a short tutorial for you to learn how to navigate around Colab and submit your files. Please follow the instructions to get started and complete the several tasks of this lab.

Task 8: **Image manipulation**

First import the library you might need.

import matplotlib.image as mpimg

import matplotlib.pyplot as plt

import numpy as np

import soundfile # you may need to install soundfile by 'pip install soundfile'

Part 1: Fade an image with a scalar.

Read the 'cameraman.tif' image from a file using mpimg.imread function (image read function). This function will load the image in a matrix.

Cameraman_image = mpimg.imread('cameraman.tif',1)

Show the image using the following commands?

plt.figure()

plt.imshow(Cameraman_image)

What is the size of the Cameraman_image matrix?
Assign fade_factor=0.5;

Multiply the fade_factor scalar and the Cameraman_image matrix and affected to the variable Fade_image?

Reconstruct the image from the obtained matrix using?
plt.figure()

plt.imshow(Fade_image)

Create a new image named First_part_image by selecting the first 100 components of each dimension. Show this image.

Create a new image named last_part_image by selecting the last 100 components of each dimension. Show this image.

Part 2 - **Version 1:**

Load the image '*5.1.09.tiff*' which is a surface in the moon. Assign this image to the variable *Moon_image.*

Show *Moon_image*

plt.figure()

plt.imshow(Moon_image)

Fade for example *Moon_image*by a factor of 0.8 and *Cameraman_image*by factor of 0.2 and sum the two matrices in new matrix named *Mixte_image*.

Show the Mixte_image

plt.figure()

plt.imshow(Mixte_image)

Create a new image named *First_part_image_1*by selecting the first 100 components of each dimension of image Cameraman_image.

Create a new image named *last_part_image_2*by selecting the last 100 components of each dimension of image Moon_image.

Fade both images *last_part_image_2* by a factor of 0.8 and *First_part_image_1*by factor of 0.2 and sum the two matrices in new matrix named *last_part_Mixted_image*.

Part 2 - **Version 2:**

We will do the same fade and mixing process as version 1 question but this time using matrix and vector multiplication.

The first step will consist of changing both image matrices from size of 256x256 each to vector of size 56536x1; (To create a vector from matrix uses the function reshape?. Use the help to show you how to use reshape function?

Create a new matrix (named *Both_images*) by appending both image vectors to form a matrix of size 56536x2.

Create of vector named *Fade_vector* of size 2x1 and containing the fade factor values (0.5,0.5) for both image.

Multiply *Both_images* matrix and *Fade_vector* to obtained the *mixing_image_vector*

Resize using the function reshape again the obtained vector *mixing_image_vector* to create *mixing_image_mtarix* of size (256x256)?

Please see the Figure 1 to show all the steps to solve this section?

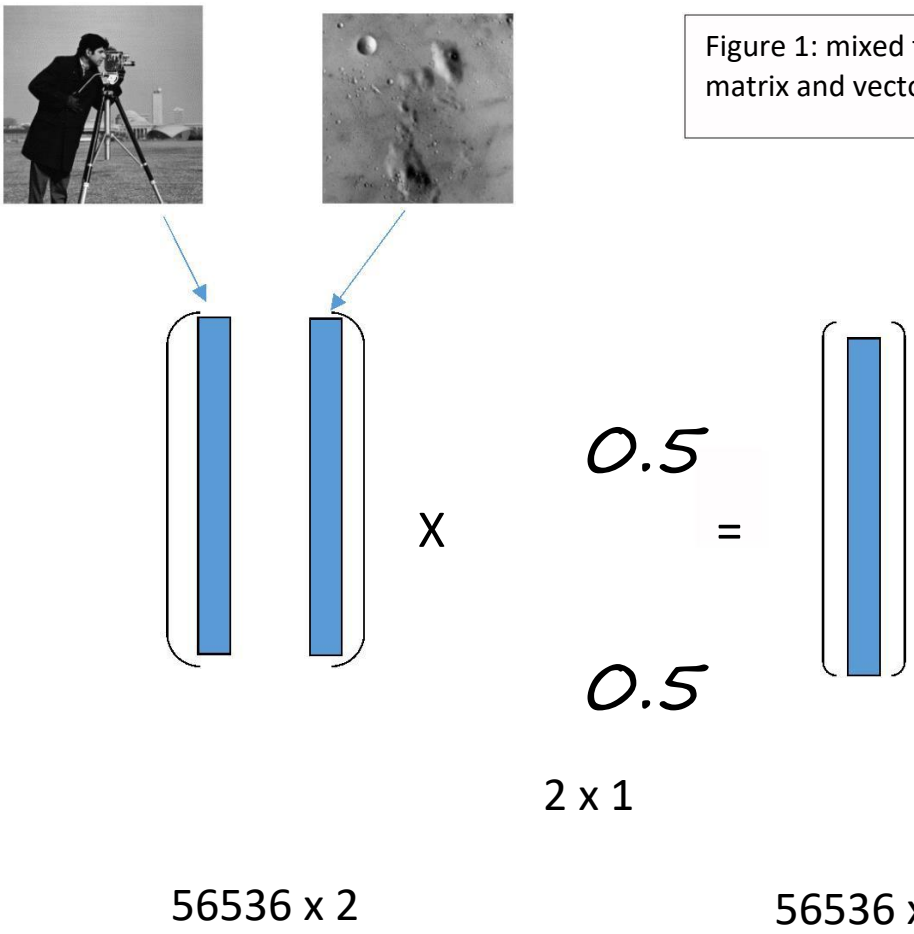

Figure 1: mixed to Fade images using matrix and vector multiplication

Task 9: **Quantization**

Load this segment of speech into Python workspace using

p, samplerate = soundfile.read('filename1.wav')

where *soundfile.read* is Python function which reads contents of a file in wave format into an array.

Change resolution from 16 bits/sample to 8 bits/sample

soundfile.write('filename2.wav', p, samplerate, subtype='PCM_U8')

p1, samplerate = soundfile.read('filename2.wav')

Change resolution from 16 bits/sample to 1bit/sample – >> *p2 = [(x>0)+0.0 for x in p1]*;

Now p, p1, p2 respectively contains same speech at different resolutions 16 bits/sample,

8bits/sample and 1 bit/sample, but all at
the same sampling frequency 8 kHz.

Store *p2* as filename3.wav and play all the three wav files, note the observations:

Task 10: **Aliasing Effect:**

It occurs when we do not use correct sampling frequency on the signals or when we do not process the original audio signal properly (means low pass filtering)

Experiment demonstrating aliasing effect:

1) Create two cosine waves of frequencies 100Hz and 600Hz and plot them?
2) Sample them with 500Hz i.e., take 500 samples per sec at equal intervals from those signals and plot.
3) Compare both plots and write observations

Writing your observations in your codes (comment them) is OK.

Submission Requirement:
Please zip the 'Python' folder as Lab1_YourName_YourJHID.zip, containing the notebooks with codes and relevant outputs below, together with all the **output and input**;
make sure we can directly run your codes and get all the figures;
submit the zip file.