Computer Vision 1: Assignment 1
(Due date: 03.12.2020)

Submission instructions:

- For each programming task, submit a `.py` source code file. Do not need to include any images or data files you used.
- For each pen & paper task, submit a `.pdf` file. Type your solution in LaTeX, Word, or another text editor of your choice and convert it to PDF – **do not submit photographs or scans of handwritten solutions!**
- In all files, include at the top names of all students in the group
- Choose exactly one person in your group that submits your solution via Moodle, it will count for the entire group.

Task 1: Appearance based classification (programming)
Training:

- Download the CIFAR-10 dataset (Python version) at `https://www.cs.toronto.edu/~kriz/cifar.html`.
- Read the first data batch in file `data_batch_1` as instructed on the webpage.
- For each of the classes "automobile", "deer", and "ship" (labels 1, 4, and 8, respectively), extract the 30 first images.
- Calculate and store the grayscale histograms of all of these images. First convert the images to grayscale by the averaging method. That is, for every pixel, the grayscale value is calculated as $(R+G+B)/3$. Then calculate the histogram using a fixed set of 51 bins covering the range $0 - 255$, each bin has length 5. **Do not take the automatically chosen bins by numpy.histogram** (Why?).

Testing:

- Read the test batch in file `test_batch`. For the same classes as above, extract the 10 first images.
- Calculate the histograms of all the images in the same way as above.
- Classify each image in the test set by finding its nearest neighbour in the training set. For each test image:
  1. Calculate the Euclidean (L_2) distances of the histogram of the test image and the histogram of *every* training image. If you have n training images, you should calculate n distances for every test image.
  2. Classify the test image by finding the minimum of the distances. Once you find the minimum distance, the predicted class of the test image is the class of the nearest training image.
- Calculate the classification accuracy as the ratio of the number of correctly classified testing images and the total number of testing images.

- Feel free to change the bins to some other lengths like 1, 10 or 255, and see how the classification accuracy varies.

Hints:

- When you have loaded the data (in the dictionary called `train`), this is how you can access the color channels of the `i`'th image.

```
raw_data = train["data"][i,:]
red = raw_data[:1024].reshape((32,32))
green = raw_data[1024:2048].reshape((32,32))
blue = raw_data[2048:].reshape((32,32))
```

- Be careful with the data types. Do not directly sum `uint8` arrays, but first convert them to floating point to avoid overflow.

Task 2: Saliency map computation (programming)

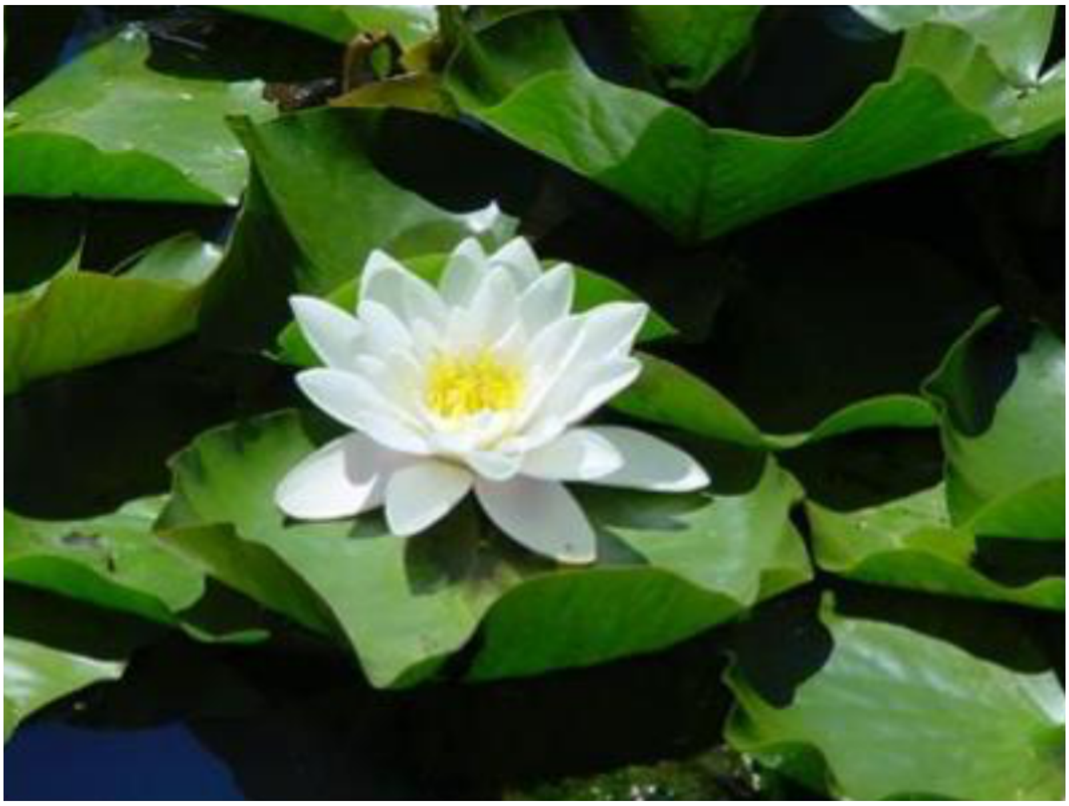

Figure 1: Flower

A saliency map highlights what visually stands out in an image from its neighbouring surrounding and immediately grabs our attention (e.g. this white flower amidst the green leaves in Fig. 1). A saliency map can be computed by sliding a center-surround window over the entire image, computing center-surround contrast at each step (i.e. the difference between center and surround values), and storing that value in a new matrix.

- Compute the saliency map for the **intensity image** for the image in Fig. 1. The image can be found in Moodle under the name"visual_attention.png":

  1. Convert the image to grayscale.

2. For every pixel, get the center-surround window and calculate the integral images (summed-area table) for both windows Use sizes 30×30 and 40×40 for the center and surround windows, respectively.
3. Subtract the center-surround values to compute a saliency value.
4. Construct the saliency map by placing the saliency value of that pixel in its corresponding position in a new matrix.
5. Display the saliency map

- What happens if the sizes of the center-surround windows were **smaller**, say 10×10 for the center window, and 20×20 for the surround window? Construct and show the new saliency map.
- What happens if the sizes of the center-surround windows were **larger**, say 100×100 for the center window and 150×150 for the surround window? Construct and show the new saliency map.

Task 3: Histograms (pen & paper).

a) Figure 2 shows a normalized histogram $p(X)$ of pixel intensity in a grayscale image.

- Find the expected value of X, i.e., $\mathbb{E}[X]$.
- Write down a table that describes the cumulative histogram of X. The table should have two rows, one for the bins and the other for the cumulative histogram value.

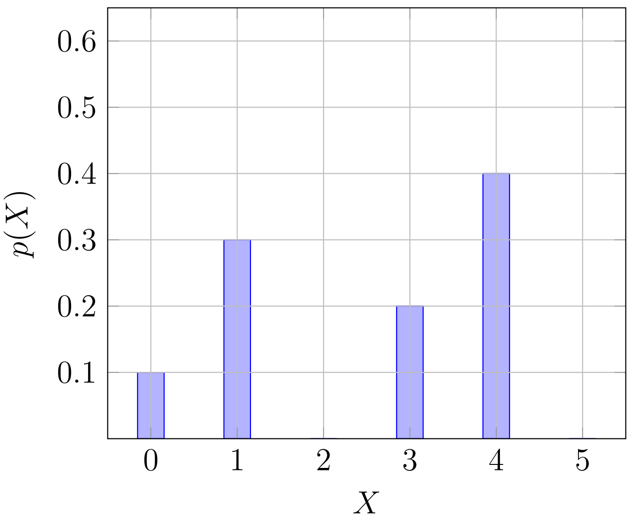

Figure 2: A normalized histogram $p(X)$.

b) Calculate the L_1 (Manhattan) distance between $p(X)$ and $q(X)$ (Figure 3).

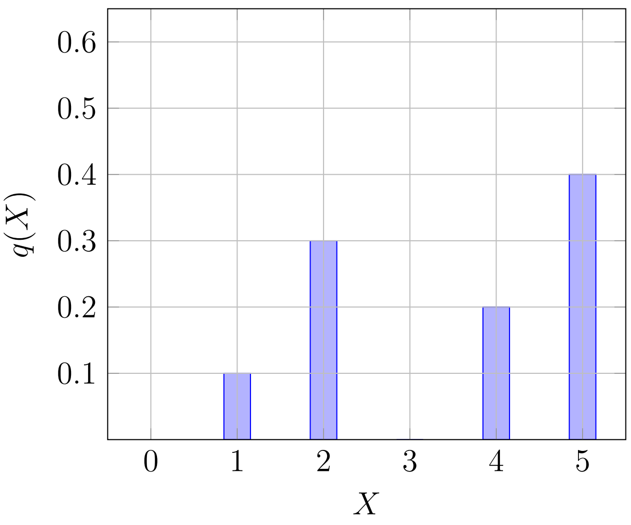

Figure 3: A normalized histogram $q(X)$.