Programming Assignment 1: Data Preparation and Understanding (10 points)

Shen-Shyang Ho (Dr.)

September 7, 2023

1. In this semester, we will be using the "Stanford Dogs" dataset (http://vision.stanford.edu/aditya86/ImageNetDogs/) for all our 4 programming assignments. There are a total of 120 classes (dog breeds). The number of images for each class ranges from 148 to 252.

   Each student will

   (a) be assigned 4 classes to work on the 4 assignments.

   (b) download **Images** (and also **Annotations** - bounding boxes) datasets for the 4 classes to work on.

   (c) create a Github account to privately share (as collaborator) their solution (Readme, Codes, Processed Dataset for Code to run correctly) with the grader.

2. Use **XML processing modules** (https://docs.python.org/3/library/xml.html) to obtain bounding box information from **Annotations** datasets and **OpenCV** (Reference: https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html) to perform image processing and feature extraction.

   (a) **Cropping and Resize Images in Your 4-class Images Dataset**: Use the bounding box information in the **Annotations** dataset relevant to your 4-class Images Dataset to crop the images in your dataset and then resize each image to a 100×100 pixel image. (Hint: https://www.kaggle.com/code/espriella/stanford-dogs-transfer-crop-stack/notebook

   Code Snippet 1:

```
def get_bounding_boxes(annot):
    xml = annot
    tree = ET.parse(xml)
    root = tree.getroot()
    objects = root.findall('object')
    bbox = []
    for o in objects:
        bndbox = o.find('bndbox')
        xmin = int(bndbox.find('xmin').text)
        ymin = int(bndbox.find('ymin').text)
        xmax = int(bndbox.find('xmax').text)
        ymax = int(bndbox.find('ymax').text)
        bbox.append((xmin,ymin,xmax,ymax))
    return bbox
```

   Code Snippet 2:

```
for i in range(len(dog_images)):
    bbox = get_bounding_boxes(annotations[i])
    dog = get_image(annotations[i])
    im = Image.open(dog)
    for j in range(len(bbox)):
        im2 = im.crop(bbox[j])
        im2 = im2.resize((331,331), Image.ANTIALIAS)
        new_path = dog.replace('../input/stanford-dogs-dataset/images/Images/','./Cropped/')
        new_path = new_path.replace('.jpg','-' + str(j) + '.jpg')
        im2=im2.convert('RGB')
        head, tail = os.path.split(new_path)
        Path(head).mkdir(parents=True, exist_ok=True)
        im2.save(new_path)
```

)

(b) **Histogram Equalization (Image Intensity Normalization)**

i. Choose 2 image from each class.

ii. Convert the color images to grayscale images (MUST use iteration; No points given if no iteration is used) (0.5 point)

iii. Plot the 8 grayscale images with their corresponding pixel intensity histograms. (0.5 point)

iv. Perform histogram equalization on the 8 images. Plot the NEW intensity equalized grayscale images and their corresponding equalized pixel intensity histograms. (1.5 point)

v. Pick a grayscale image and its corresponding equalized image. Plot the 2 images next to each other. What did you observe? (0.5 point)

(c) **RGB histogram**

i. Choose 1 image from each class.

ii. Plot the images with their corresponding RGB histogram values (The three curves MUST be in one figure - see Figure 1, add x-axis label "Intensity" and y-axis label "Pixel Count"). (1 point)

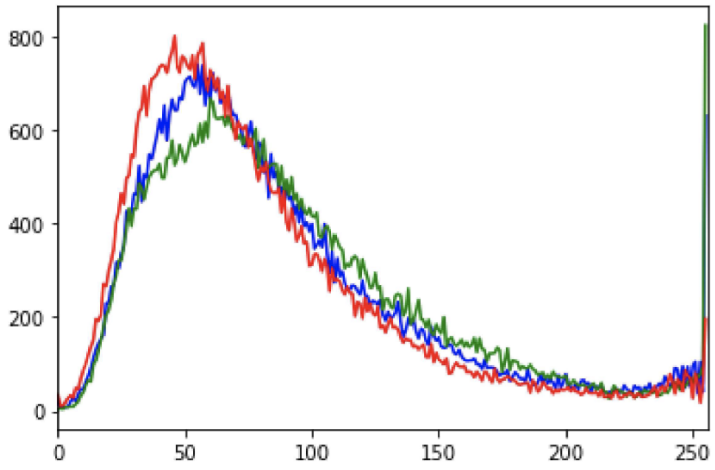

Figure 1:

(d) **Histogram Comparison (Measures of Similarity and Dissimilarity)** (see https://docs.opencv.org/3.4/d8/dc8/tutorial_histogram_comparison.html)

i. Pick 2 images from the same class and 1 image from another class.

ii. Convert the three images to grayscale pixel intensity histograms. (These will be the vector representations of the images)

iii. Perform histogram comparison using the following metrics/measures.

- Euclidean Distance
- Manhattan Distance
- Bhattacharyya distance
- Histogram Intersection

For this task, you will compare histograms by computing the metrics/measures of (1) the 2 images from the same class, AND (2) 2 images from different classes. (2 points) (Note: You can also use other packages.)

(e) **Image Feature Descriptor: ORB (Oriented FAST and Rotated BRIEF) (see https://docs.opencv.org/3.4/d1/d89/tutorial_py_orb.html)**

i. Pick 1 image and perform keypoint extraction using ORB feature descriptor. (Hint:

```
orb = cv.ORB_create(edgeThreshold=edge_threshold,
patchSize=30, nlevels=8, fastThreshold=20,
scaleFactor=1.2, WTA_K=2,scoreType=cv.ORB_HARRIS_SCORE,
firstLevel=0, nfeatures=30)

# find the keypoints with ORB
kp = orb.detect(img,None)

# draw only keypoints location,not size and orientation
img2 = cv.drawKeypoints(img, kp, None, color=(0,255,0))
```

)

ii. Try to extract between 25 and 75 keypoints. What is the number of keypoints extracted? What are the edge threshold value and patchSize you used? (Edge threshold is the size of the border where the features are not detected. It should roughly match the patchSize parameter). (0.5 points)

iii. Plot the keypoints on the image. (see Fig 2 for one such figure) (0.5 point)

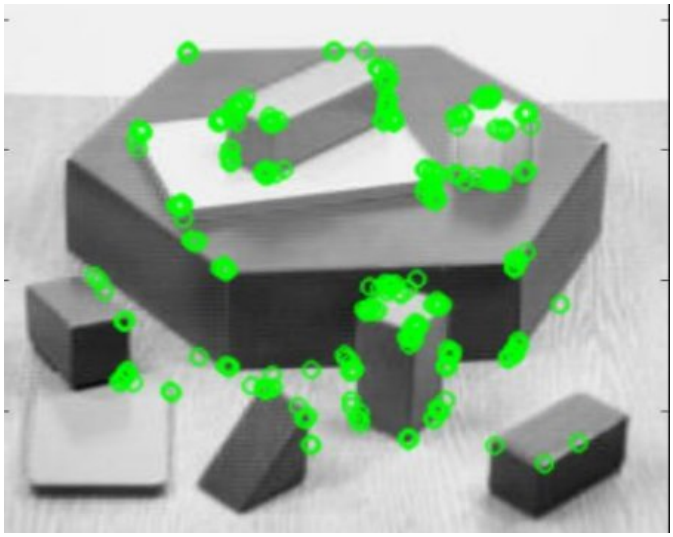

Figure 2:

(f) **Dimensionality reduction (using Principal Component Analysis, PCA)** (see https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html for PCA. https://scikit-learn.org/stable/auto_examples/decomposition/plot_pca_iris.html for code example. We will use scikit learn more extensively in the next assignment)

i. Use images from any two classes.

ii. Convert all the images to grayscale pixel intensity histograms and normalize the dataset.

iii. Perform Principal Component Analysis (PCA) dimensionality reduction on the set of histograms to 2 dimensions. (1 point) (Note: You should not use the class labels)

iv. Plot the 2D points using 2 different colors for data from the 2 classes (see Figure 3 for an example of the plot without normalization). Are the data from the two classes separable? (1 point)

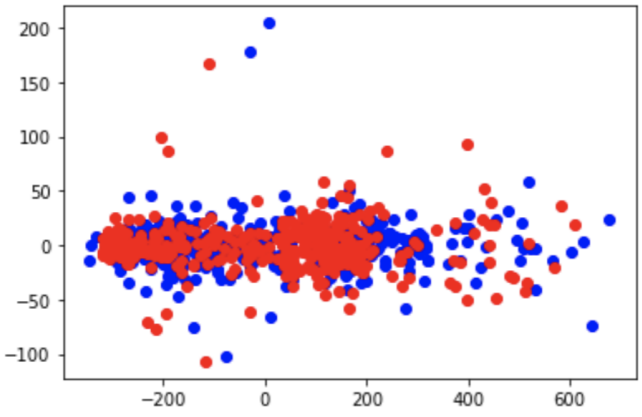

Figure 3: