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

January 20, 2024

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 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 **scikit-Image** (Reference: https://scikit-image.org/) 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 128 × 128 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)
```

)
(0.5 point)

(b) **Image Processing**

   i. Choose 2 images from each class.

   ii. Convert the color images to grayscale images (see https://scikit-image.org/docs/stable/auto_examples/color_exposure/plot_rgb_to_gray.html) (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 (i.e., 256 bins). (1 point)

   iv. Using the 8 grayscale images above, perform edge detection (see https://scikit-image.org/docs/stable/auto_examples/edges/plot_edge_filter.html#sphx-glr-auto-examples using the **sobel edge filter**.

   v. Plot the 8 edge images as shown in https://scikit-image.org/docs/stable/auto_examples/edges/plot_edge_filter.html#sphx-glr-auto-examples-edges-plot-edge-filt (1 point)

(c) **Edge histogram**

   i. Choose 1 image from each class.

   ii. Convert the color images to grayscale images

   iii. For each image *I*, use the following

```
import numpy as np
from skimage import filters

def angle(dx, dy):
    """Calculate the angles between horizontal and vertical operators."""
    return np.mod(np.arctan2(dy, dx), np.pi)

angle_sobel = angle(filters.sobel_h(I),
                    filters.sobel_v(I))
```

   to obtain an "angle" for each pixel in the images (Intuitively, one can think of the "angle" as the direction of edge gradient at the pixel).

   iv. Use **skimage.exposure.histogram** (see https://scikit-image.org/docs/stable/api/skimage.exposure.html#skimage.exposure.histogram) to obtain a histogram with 36 bins. (1 point)

   v. Plot the images with their corresponding edge histogram values (add x-axis label "Bins" and y-axis label "Pixel Count"). (1 point)

(d) **Histogram Comparison (Measures of Similarity and Dissimilarity)** (see https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics.pairwise)

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

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

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

   - Euclidean distance

- Manhattan distance
- Cosine distance

Using the 3 images above, 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. (1.5 points)

(e) **Histogram of Oriented Gradient (HOG) feature descriptor (see https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients)**

i. Pick 1 image and compute its HOG descriptors. Visualise the image and the HOG descriptors for the image (see https://scikit-image.org/docs/stable/auto_examples/features_detection/plot_hog.html#sphx-glr-auto-examples-features-detection-plot-hog-py) (1 point)

(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 from the two classes to edge histograms.(0.5 points)

iii. Perform Principal Component Analysis (PCA) dimensionality reduction on the set of histograms to reduce from 36 to 2 dimensions. (Note: You should not use the class labels) (1 point)

iv. Plot the 2D points using 2 different colors for data from the 2 classes (see Figure 1). Are your data from the two classes separable? (1 point)

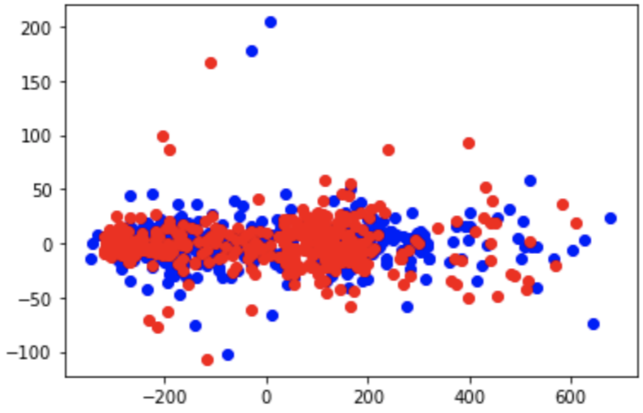

Figure 1: