In [ ]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import cv2
import os
import xml.etree.ElementTree as ET
from PIL import Image
from pathlib import Path
import random
import warnings

warnings.filterwarnings("ignore")
```

In [91]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import cv2
import os
import xml.etree.ElementTree as ET
from PIL import Image
from pathlib import Path
import random
import warnings

warnings.filterwarnings("ignore")

# Define directories
images_dir = (r'C:\Users\ADMIN\stanforddogs\images')
annotation_dir = (r'C:\Users\ADMIN\stanforddogs\annotations')

# Function to list all the directories inside the given path
def list_directories(path):
    return [d for d in os.listdir(path) if os.path.isdir(os.path.join(path, d))]

# List subdirectories
images_subdirs = list_directories(images_dir)
annotations_subdirs = list_directories(annotation_dir)

# Print the directories
print("Directories in Images folder:", images_subdirs)
print("\nDirectories in Annotations folder:", annotations_subdirs)  # Fixed typo here
```

```
Directories in Images folder: ['n02092002-Scottish_deerhound', 'n02093991-Irish_terrier', 'n02097474-Tibetan_ter
rier', 'n02106166-Border_collie']

Directories in Annotations folder: ['n02092002-Scottish_deerhound', 'n02093991-Irish_terrier', 'n02097474-Tibeta
n_terrier', 'n02106166-Border_collie']
```

2a) cropping and resizing images

In [ ]:

```
def get_bounding_boxes (annot_path):
    tree = ET.parse(annot_path)
    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
    for subdir in images_subdirs:
 #Path to subdirectories of image and annotation
        img_subdir_path = images_dir + "\\" + subdir
        annot_subdir_path = annotation_dir +"\\"+ subdir
 # Getting all xml files in the annotation subdirectory
        images = [img_subdir_path +"\\" + f for f in os.listdir(img_subdir_path)]
        annotations = [annot_subdir_path +"\\" + f for f in os.listdir(annot_subdir_path)]
        for i, annot in enumerate(annotations):
            bbox = get_bounding_boxes(annot)
            dog_image_path = images[i]
            im = Image.open(dog_image_path)
            for j, box in enumerate (bbox):
                im2 = im.crop (box)
                im2 = im2.resize((128, 128))
                new_path = str(dog_image_path).replace(str(images_dir), r'C:\Users\ADMIN\stanforddogs\cropped')
                head, tail = os.path.split(new_path)
                Path(head).mkdir(parents=True, exist_ok=True)
                im2.save(new_path)
cropped_dir = Path(r'C:\Users\ADMIN\stanforddogs\cropped')
```

2b-i

In [201…

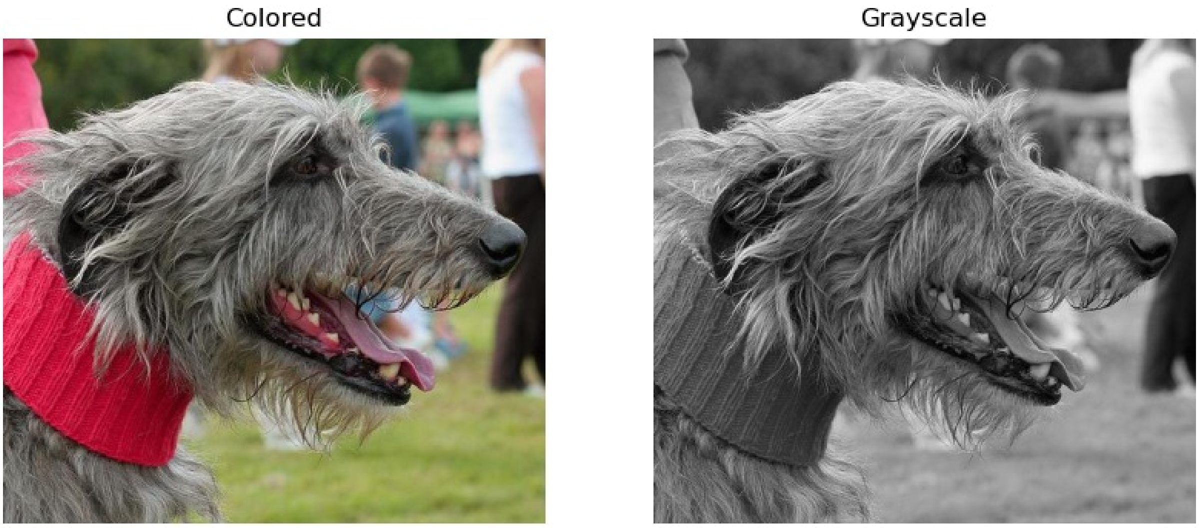
```
import pandas as pd
import numpy as np
import glob
import matplotlib.pyplot as plt
import cv2
import os
import xml.etree.ElementTree as ET
from PIL import Image
from pathlib import Path
import random
import warnings
selected_images = {}
cropped_dir = Path(r'C:\Users\ADMIN\stanforddogs\cropped')
for subdir in cropped_dir.iterdir():
    if subdir.is_dir():
        # List all JPG files in the subdirectory
        image_files = list(subdir.glob('*.jpg'))  # Use '*.jpg' instead of '*jpg'
        if len(image_files) >= 1:
            selected_images[subdir.name] = image_files[:1]  # Select one image
        else:
            print(f"Warning: Less than 1 image found for class {subdir.name}")

# Check if any images were selected
if selected_images:
    print("\nSelected Images:")
    for class_name, images in selected_images.items():
        print(f"Class: {class_name}")
        for img in images:
            print(img)
else:
    print("No images were selected from any class.")
```

```
Selected Images:
Class: n02092002-Scottish_deerhound
C:\Users\ADMIN\stanforddogs\cropped\n02092002-Scottish_deerhound\n02092002_3.jpg
Class: n02093991-Irish_terrier
C:\Users\ADMIN\stanforddogs\cropped\n02093991-Irish_terrier\n02093991_50.jpg
Class: n02097474-Tibetan_terrier
C:\Users\ADMIN\stanforddogs\cropped\n02097474-Tibetan_terrier\n02097474_16.jpg
Class: n02106166-Border_collie
C:\Users\ADMIN\stanforddogs\cropped\n02106166-Border_collie\n02106166_5.jpg
```

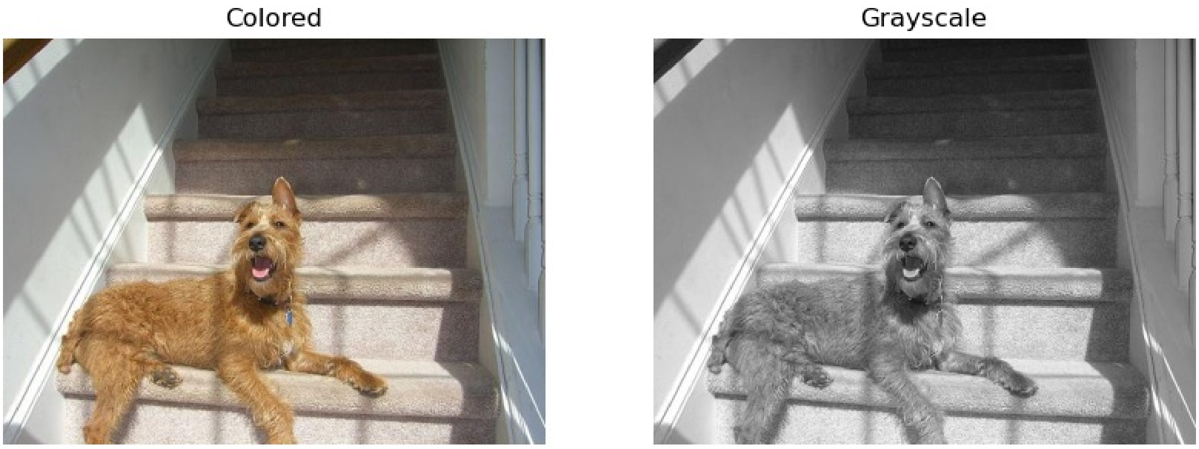
2b-ii

In [17]:

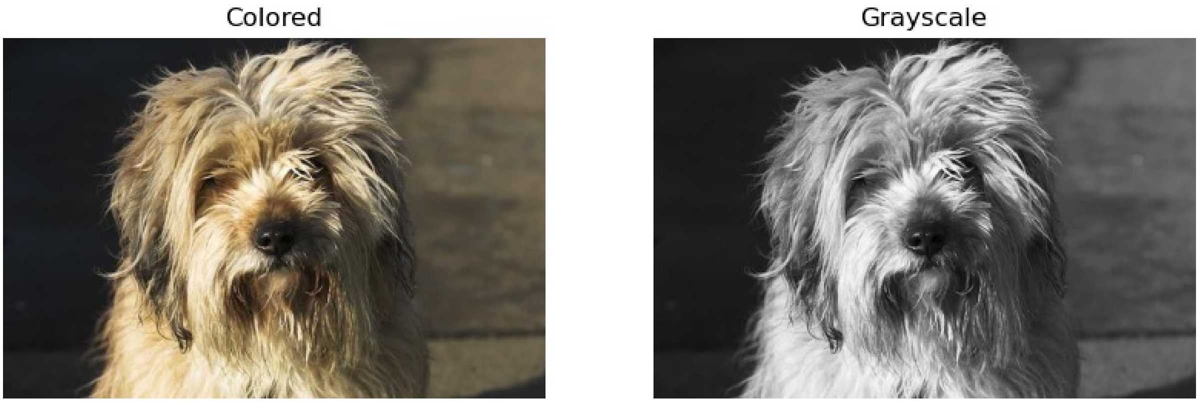
```
import matplotlib.pyplot as plt
from PIL import Image
from pathlib import Path

# Enable inline plotting for Jupyter Notebooks
%matplotlib inline

def display_images(image_path, grayscale_dir):
    img = Image.open(image_path)
    gray_img = img.convert('L')

    # Display images
    plt.figure(figsize=(10, 5))
    plt.subplot(1, 2, 1)
    plt.imshow(img)
    plt.title('Colored')
    plt.axis('off')

    plt.subplot(1, 2, 2)
    plt.imshow(gray_img, cmap='gray')
    plt.title('Grayscale')
    plt.axis('off')

    plt.show()

    # Save the grayscale image
    gray_image_path = grayscale_dir / (image_path.stem + '_gray.jpg')
    gray_img.save(gray_image_path)
    print(f"Saved grayscale image to: {gray_image_path}")

# Define the selected image directories
selected_images = [
    Path(r'C:\Users\ADMIN\stanforddogs\cropped\n02092002-Scottish_deerhound'),
    Path(r'C:\Users\ADMIN\stanforddogs\cropped\n02093991-Irish_terrier'),
    Path(r'C:\Users\ADMIN\stanforddogs\cropped\n02097474-Tibetan_terrier'),
    Path(r'C:\Users\ADMIN\stanforddogs\cropped\n02106166-Border_collie'),
]

# Create a directory for grayscale images
grayscale_dir = Path(r'C:\Users\ADMIN\stanforddogs\grayscale')
grayscale_dir.mkdir(parents=True, exist_ok=True)

# Process each class directory
for class_dir in selected_images:
    image_files = list(class_dir.glob('*.jpg'))
    if image_files:
        display_images(image_files[0], grayscale_dir)
    else:
        print(f"No images found in {class_dir}")
```



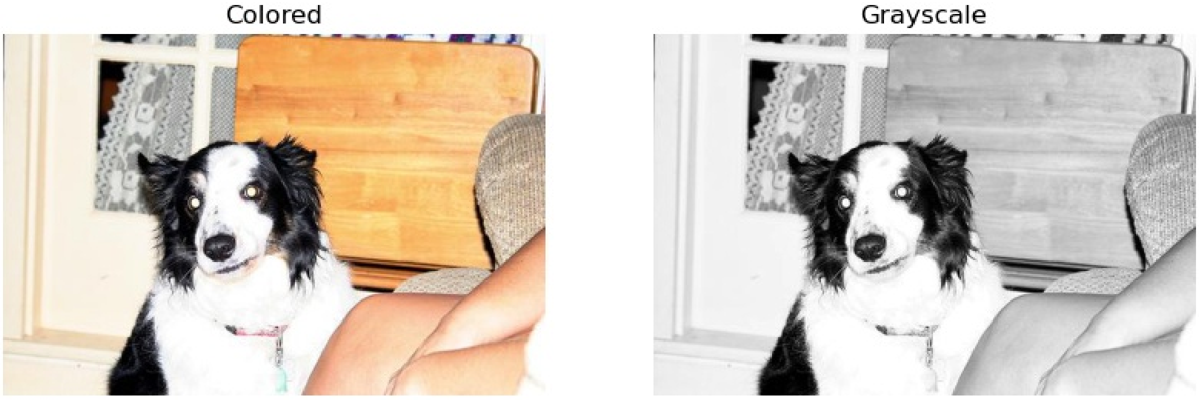
```
Saved grayscale image to: C:\Users\ADMIN\stanforddogs\grayscale\n02092002_3_gray.jpg
```



```
Saved grayscale image to: C:\Users\ADMIN\stanforddogs\grayscale\n02093991_50_gray.jpg
```



```
Saved grayscale image to: C:\Users\ADMIN\stanforddogs\grayscale\n02097474_16_gray.jpg
```



```
Saved grayscale image to: C:\Users\ADMIN\stanforddogs\grayscale\n02106166_5_gray.jpg
```

2b-iii)

In [23]:

```
import numpy as np
import cv2
from skimage import filters
from pathlib import Path

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

# Define the directory for grayscale images
grayscale_dir = Path(r'C:\Users\ADMIN\stanforddogs\grayscale')

# Process each image in the grayscale directory
for img_path in grayscale_dir.glob('*.jpg'):
    # Read the grayscale image
    I = cv2.imread(str(img_path), cv2.IMREAD_GRAYSCALE)

    # Compute the horizontal and vertical gradients using the Sobel filter
    sobel_h = filters.sobel_h(I)
    sobel_v = filters.sobel_v(I)

    # Calculate the angles
    angle_sobel = angle(sobel_h, sobel_v)

    # Optionally, save the angle image for visualization
    angle_image = (angle_sobel / np.pi * 255).astype(np.uint8)  # Scale to [0, 255]
    angle_image_path = grayscale_dir / f"angle_{img_path.name}"
    cv2.imwrite(str(angle_image_path), angle_image)

    # Print for confirmation
    print(f"Processed angle for image: {img_path.name}")
```

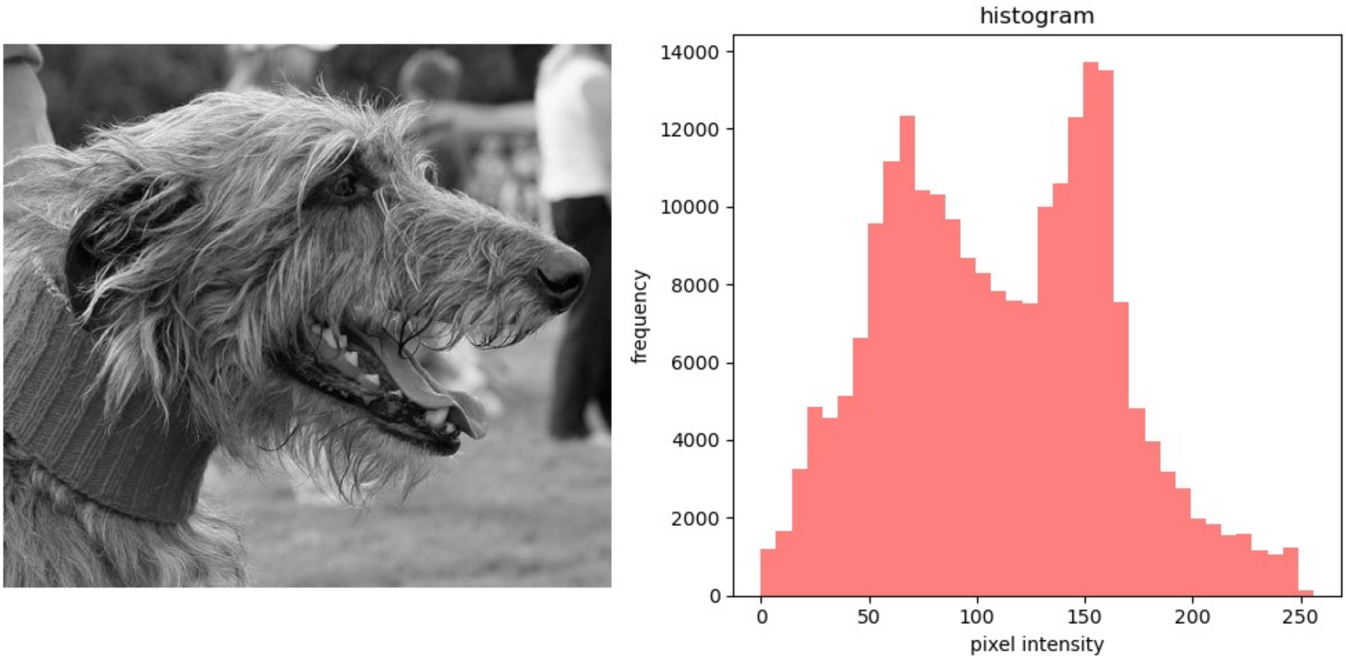
```
Processed angle for image: n02092002-Scottish_deerhound_n02092002_3.jpg
Processed angle for image: n02093991-Irish_terrier_n02093991_50.jpg
Processed angle for image: n02097474-Tibetan_terrier_n02097474_16.jpg
Processed angle for image: n02106166-Border_collie_n02106166_5.jpg
```

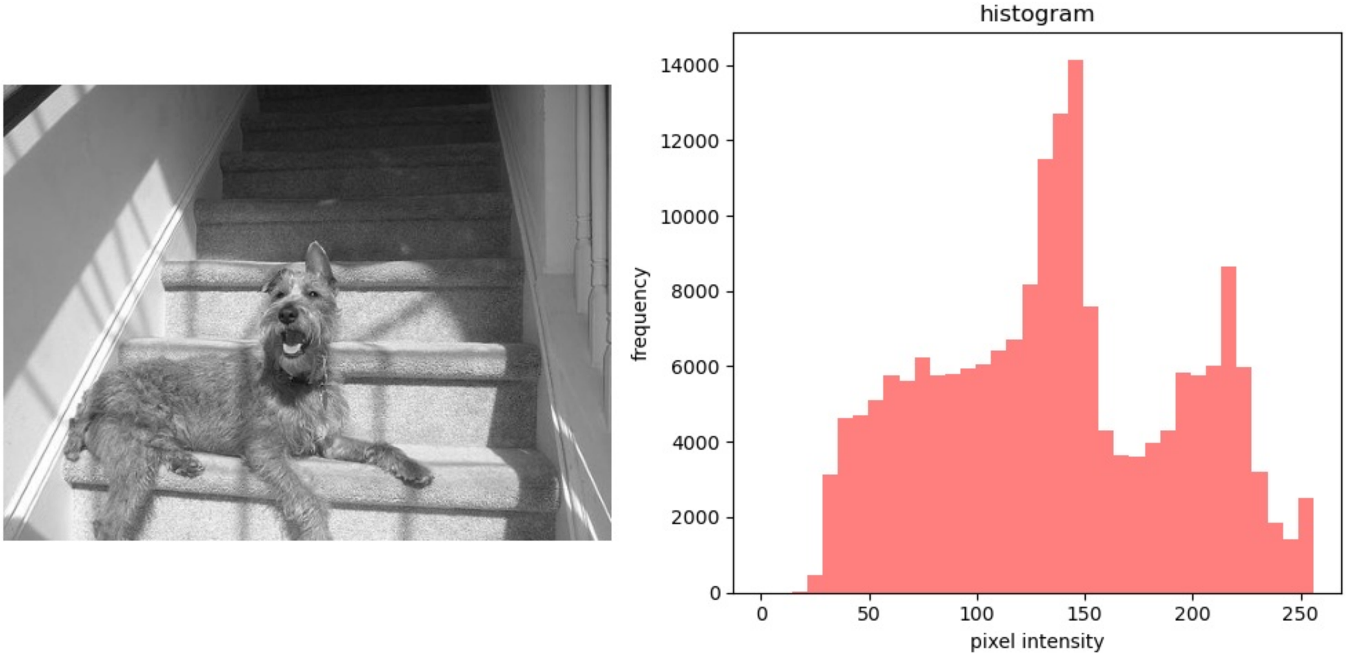
In [ ]:

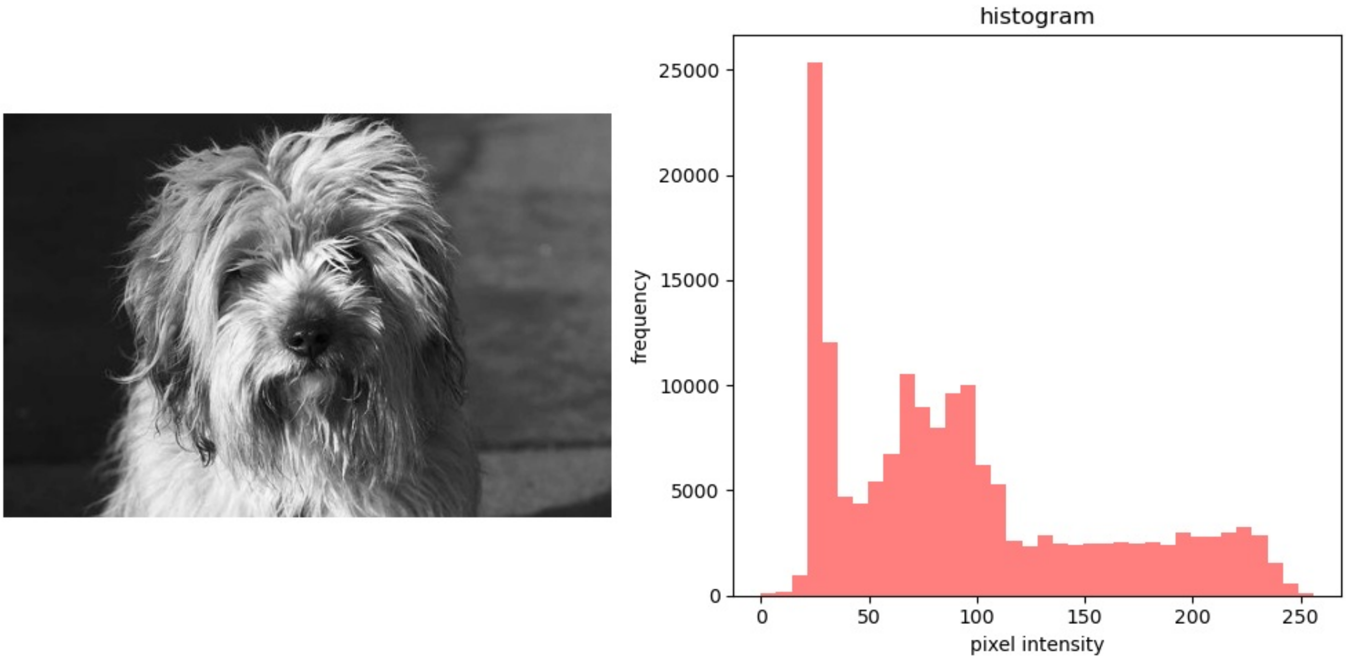
```
2b-iv)
```

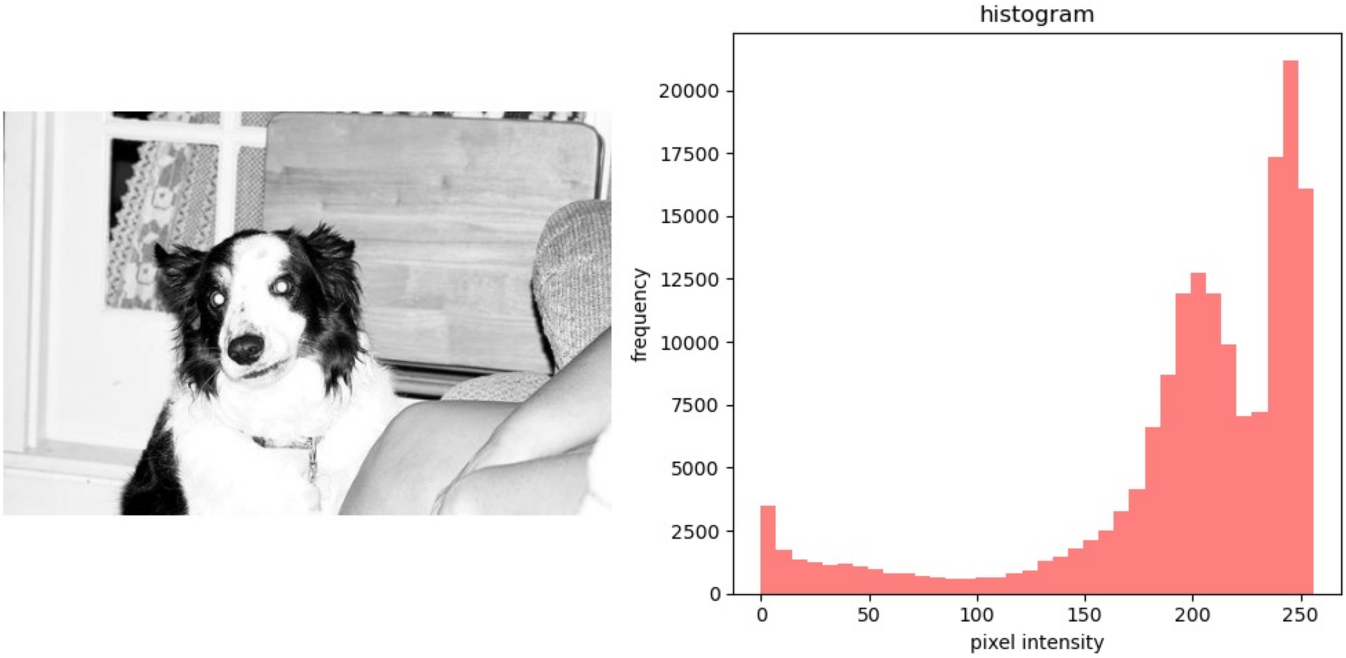
In [92]:

```
import matplotlib.pyplot as plt
from PIL import Image
from pathlib import Path
%matplotlib inline
def display_images_and_histogram(image_path):
    img = Image.open(image_path)
    gray_img = img.convert('L')
    pixel_values = np.asarray(gray_img).flatten()
    plt.figure(figsize=(10, 5))
    plt.subplot(1,2,1)
    plt.imshow(gray_img, cmap='gray')
    plt.axis('off')
    plt.subplot(1,2,2)
    plt.hist(pixel_values, bins=36, range=(0, 256), color='red', alpha=0.5)
    plt.xlabel('pixel intensity')
    plt.ylabel('frequency')
    plt.title('histogram')
    plt.tight_layout()
    plt.show()
selected_images = [
    Path(r'C:\Users\ADMIN\stanforddogs\cropped\n02092002-Scottish_deerhound'),
    Path(r'C:\Users\ADMIN\stanforddogs\cropped\n02093991-Irish_terrier'),
    Path(r'C:\Users\ADMIN\stanforddogs\cropped\n02097474-Tibetan_terrier'),
    Path(r'C:\Users\ADMIN\stanforddogs\cropped\n02106166-Border_collie'),
]
for class_dir in selected_images:

    image_files = list(class_dir.glob('*.jpg'))
    if image_files:

        display_images_and_histogram(image_files[0])
    else:
        print(f"No images found in {class_dir}")
```





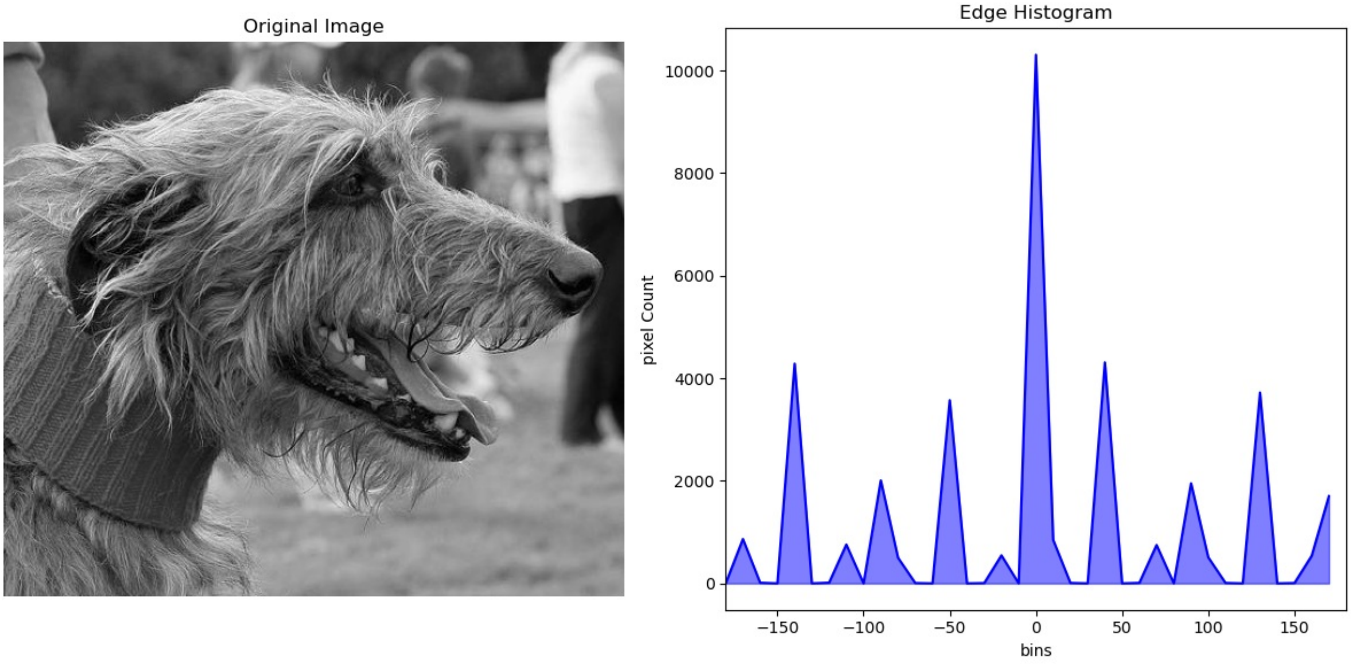


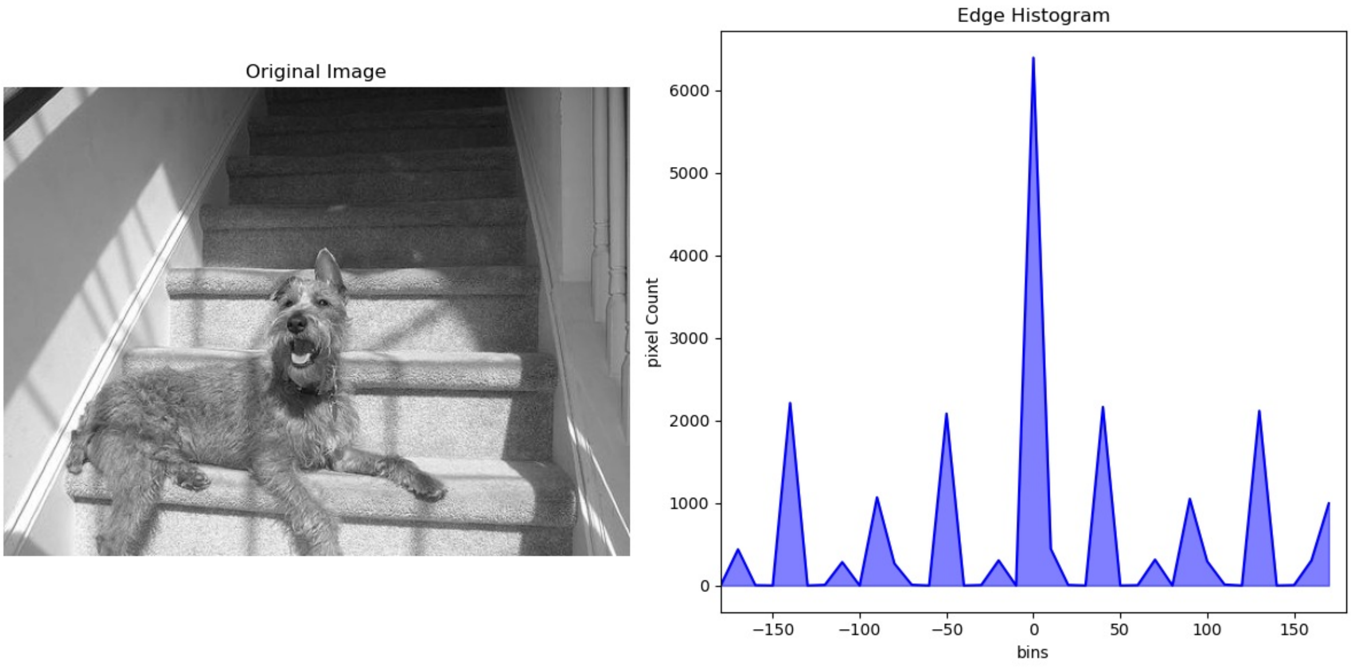


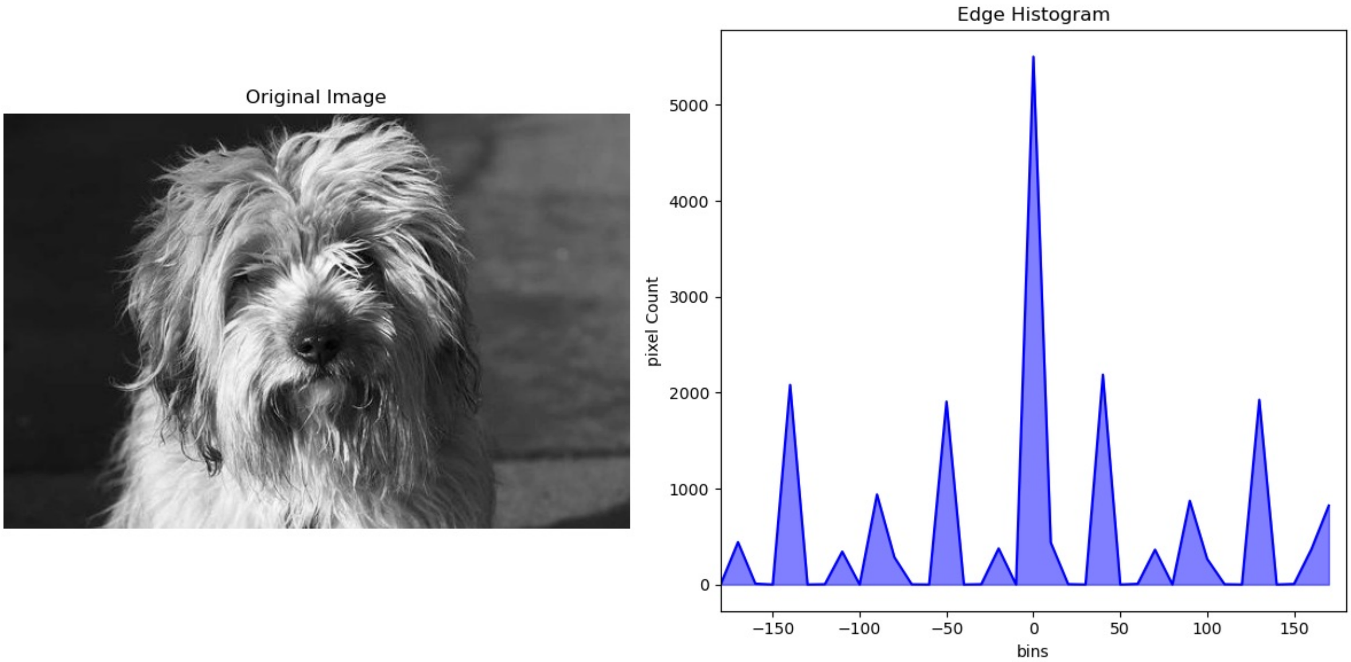
2b-v)

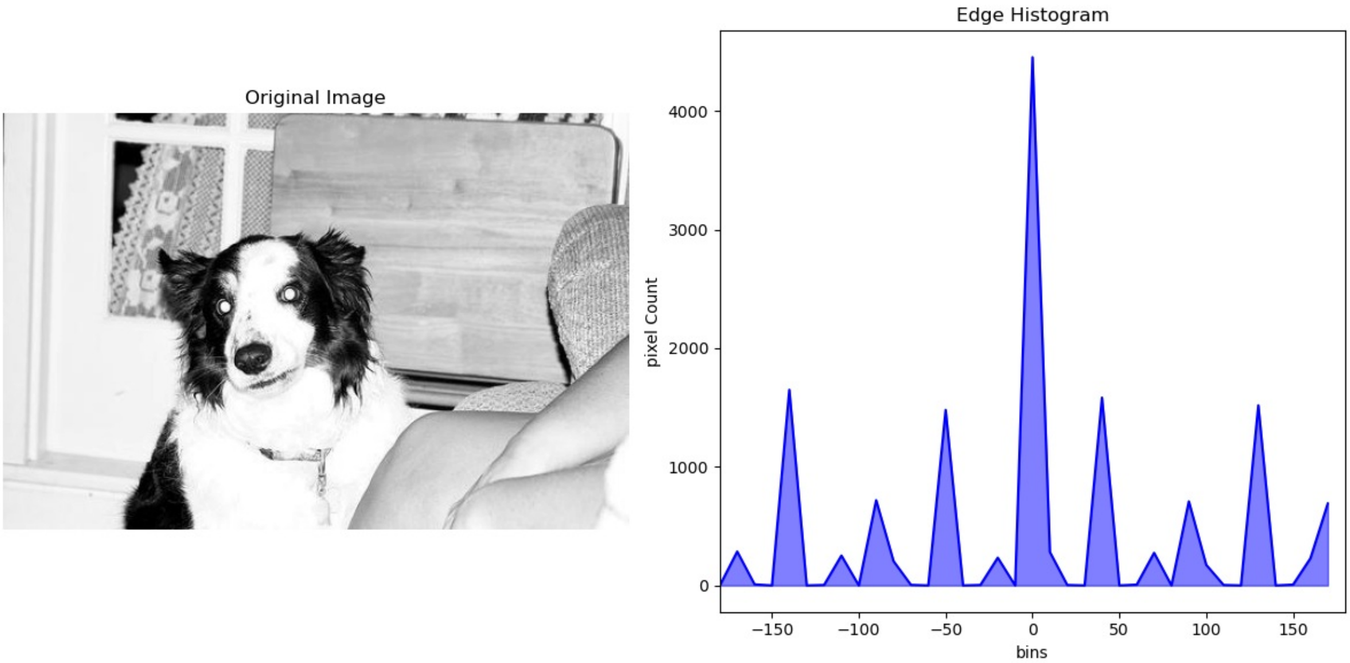
In [7]:

```
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
from pathlib import Path
import cv2
%matplotlib inline
def display_images_and_edge_histogram(image_path):
    img = Image.open(image_path)
    gray_img = img.convert('L')
    gray_img_array = np.asarray(gray_img)
    edges = cv2.Canny(gray_img_array, 100, 200)
    sobel_x = cv2.Sobel(edges, cv2.CV_64F, 1, 0, ksize=3)
    sobel_y = cv2.Sobel(edges, cv2.CV_64F, 0, 1, ksize=3)
    gradient_angle = np.arctan2(sobel_y, sobel_x) * (180 / np.pi)
    hist, bin_edges = np.histogram(gradient_angle[edges > 0], bins=36, range=(-180, 180))
    plt.figure(figsize=(12, 6))
    plt.subplot(1, 2, 1)
    plt.imshow(gray_img, cmap='gray')
    plt.axis('off')
    plt.title('Original Image')
    plt.subplot(1, 2, 2)
    plt.plot(bin_edges[:-1], hist, color='blue')
    plt.fill_between(bin_edges[:-1], hist, alpha=0.5, color='blue')
    plt.xlim([-180, 180])
    plt.xlabel('bins')
    plt.ylabel('pixel Count')
    plt.title('Edge Histogram')

    plt.tight_layout()
    plt.show()

selected_images = [
    Path(r'C:\Users\ADMIN\stanforddogs\cropped\n02092002-Scottish_deerhound'),
    Path(r'C:\Users\ADMIN\stanforddogs\cropped\n02093991-Irish_terrier'),
    Path(r'C:\Users\ADMIN\stanforddogs\cropped\n02097474-Tibetan_terrier'),
    Path(r'C:\Users\ADMIN\stanforddogs\cropped\n02106166-Border_collie'),
]

for class_dir in selected_images:
    image_files = list(class_dir.glob('*.jpg'))
    if image_files:
        display_images_and_edge_histogram(image_files[0])
    else:
        print(f"No images found in {class_dir}")
```







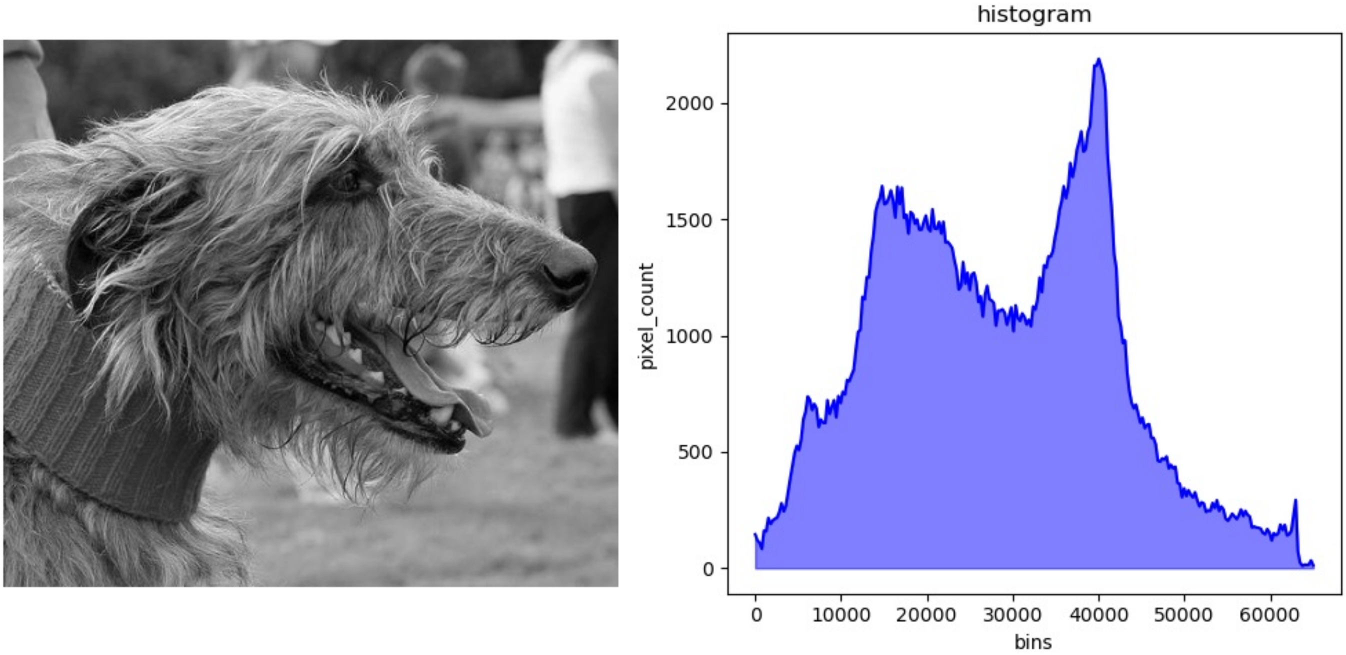


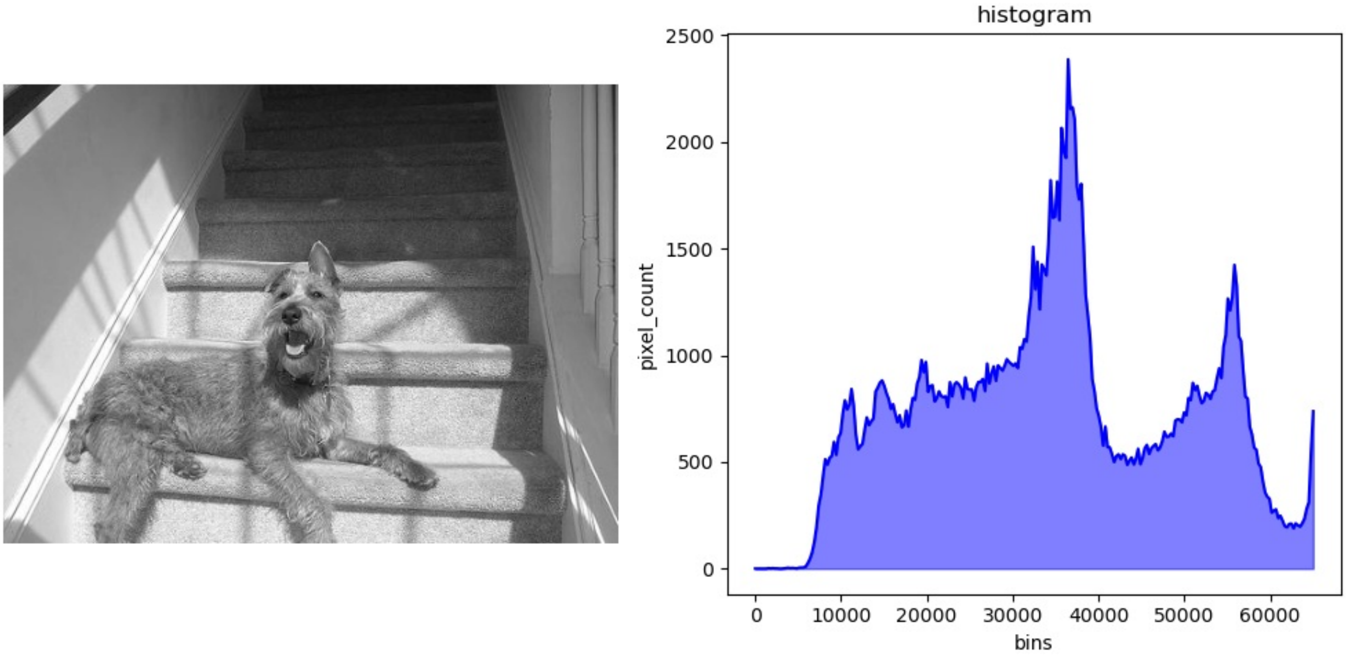
2b-vi)

In [130…

```
import matplotlib.pyplot as plt
from PIL import Image
from pathlib import Path
%matplotlib inline
from sklearn.metrics import pairwise
def display_images_and_edge_histogram(image_path):
    img = Image.open(image_path)
    gray_img = img.convert('L')
    pixel_values = np.asarray(gray_img).flatten()
    pixel_count, bin_edges = np.histogram(pixel_values, bins=256, range=(0, 256))
    if np.any(np.isnan(pixel_count)):
        print(f"NaN values found in histogram for {image_path}")
        return None
    plt.figure(figsize=(10, 5))
    plt.subplot(1,2,1)
    plt.imshow(gray_img, cmap='gray')
    plt.axis('off')
    plt.subplot(1,2,2)
    plt.plot(bin_edges[:-1] * 255, pixel_count, color='blue')
    plt.fill_between(bin_edges[:-1] * 255, pixel_count, alpha=0.5, color='blue')

    plt.xlabel('bins')
    plt.ylabel('pixel_count')
    plt.title('histogram')
    plt.tight_layout()
    return pixel_count
selected_images = [
    Path(r'C:\Users\ADMIN\stanforddogs\cropped\n02092002-Scottish_deerhound'),
    Path(r'C:\Users\ADMIN\stanforddogs\cropped\n02093991-Irish_terrier'),
   ]
histograms = []
for class_dir in selected_images:
    image_files = list(class_dir.glob('*.jpg'))
    if image_files:
        hist = display_images_and_edge_histogram(image_files[0])
        histograms.append(hist)
    else:
        print(f"No images found in {class_dir}")
if len(histograms) >= 2:
    hist1 = histograms[0]
    hist2 = histograms[1]
    euclidean_distance = pairwise.euclidean_distances([hist1], [hist2])[0][0]
    manhattan_distance = pairwise.manhattan_distances([hist1], [hist2])[0][0]
    cosine_distance = pairwise.cosine_distances([hist1], [hist2])[0][0]
    print(f"Euclidean Distance between first two images: {euclidean_distance:.4f}")
    print(f"Manhattan Distance between first two images: {manhattan_distance:.4f}")
    print(f"Cosine Distance between first two images: {cosine_distance:.4f}")
else:
    print("Not enough histograms to compare.")
```

```
Euclidean Distance between first two images: 8419.8962
Manhattan Distance between first two images: 107556.0000
Cosine Distance between first two images: 0.1343
```

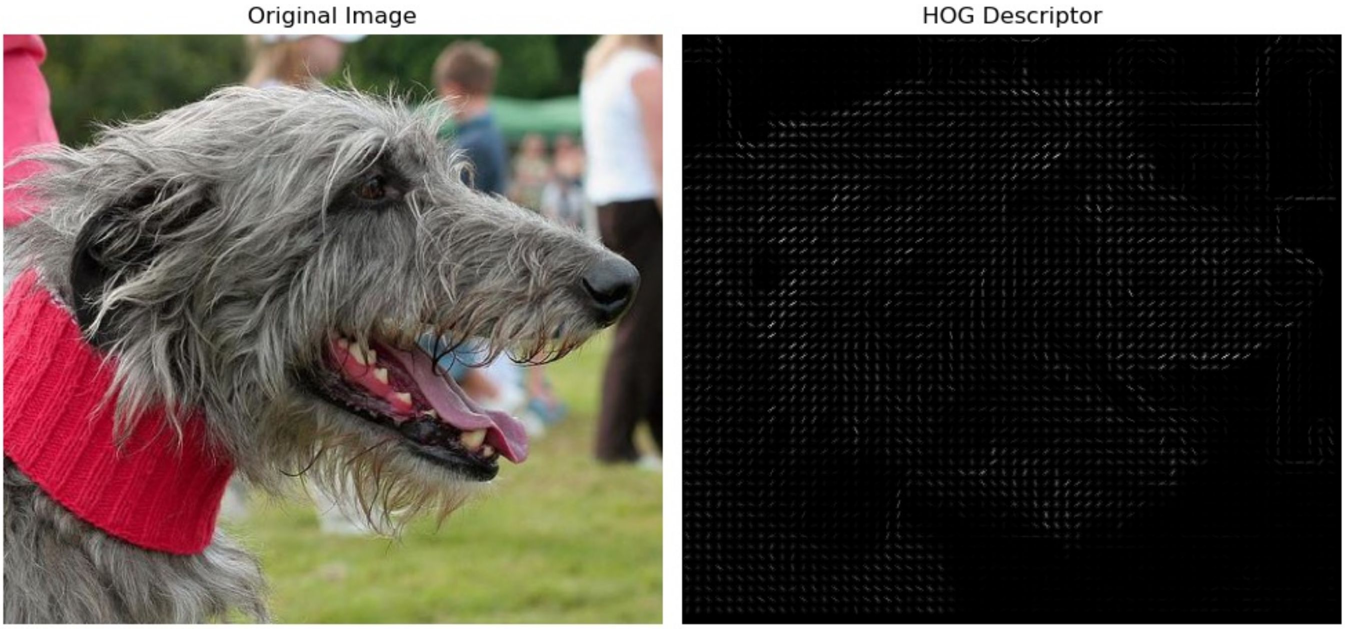




2c)

In [21]:

```
import matplotlib.pyplot as plt
from skimage import io, color
from skimage.feature import hog
from skimage import exposure
import numpy as np
image_path = r'C:\Users\ADMIN\stanforddogs\cropped\n02092002-Scottish_deerhound\n02092002_3.jpg'
image = io.imread(image_path)
gray_image = color.rgb2gray(image)
hog_features, hog_image = hog(gray_image, visualize=True)
hog_image_rescaled = exposure.rescale_intensity(hog_image, in_range=(0, 1))
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.imshow(image)
plt.title('Original Image')
plt.axis('off')
plt.subplot(1, 2, 2)
plt.imshow(hog_image_rescaled, cmap='gray')
plt.title('HOG Descriptor')
plt.axis('off')
plt.tight_layout()
plt.show()
```



Original Image

HOG Descriptor

2d-i)

In [9]:

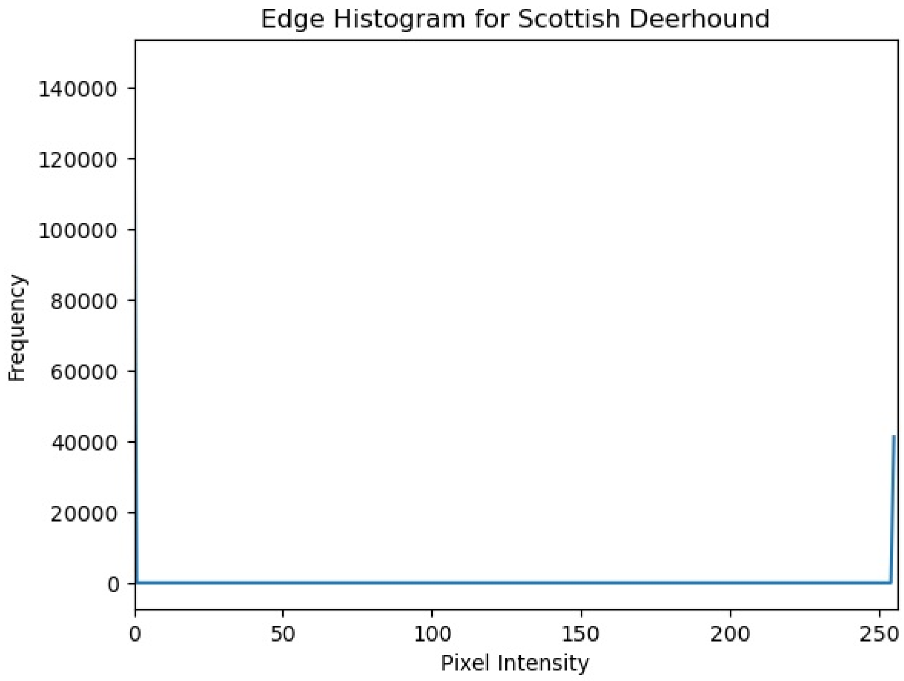
```
from pathlib import Path
images_dir = Path(r'C:\Users\ADMIN\stanforddogs\images')
def list_directories(dir_path):
    return [d for d in dir_path.iterdir() if d.is_dir()]
images_subdir = list_directories(images_dir)
selected_four_classes = images_subdir[:4]
selected_images = {}
for class_dir in selected_four_classes:

    image_files = list(class_dir.glob('*.jpg'))
    selected_images[class_dir.name] = image_files
for class_name, images in selected_images.items():
    print(f"Number of images from {class_name}: {len(images)}")
```

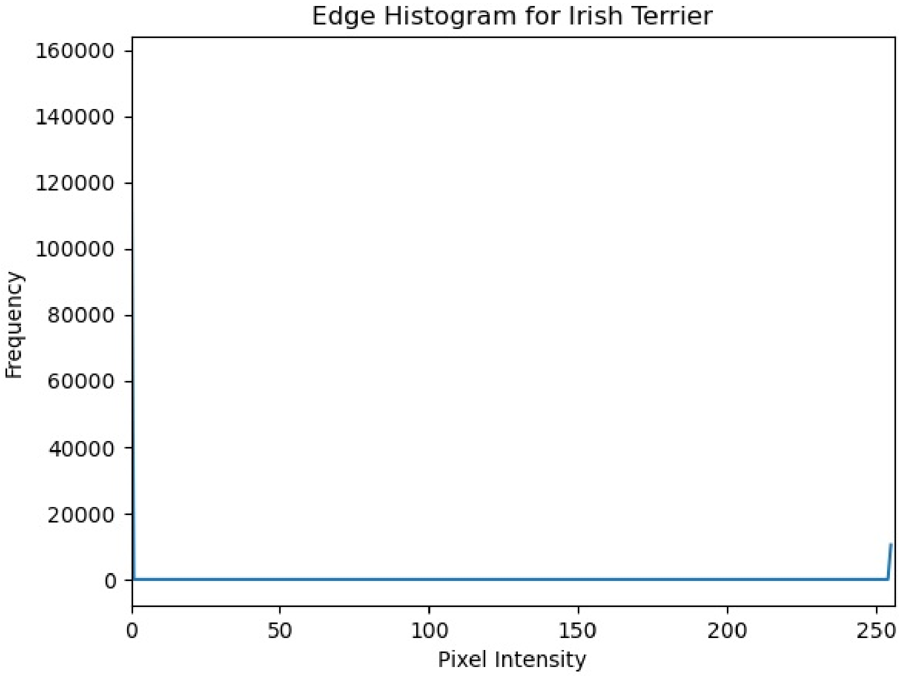
```
Number of images from n02092002-Scottish_deerhound: 232
Number of images from n02093991-Irish_terrier: 169
Number of images from n02097474-Tibetan_terrier: 206
Number of images from n02106166-Border_collie: 150
```

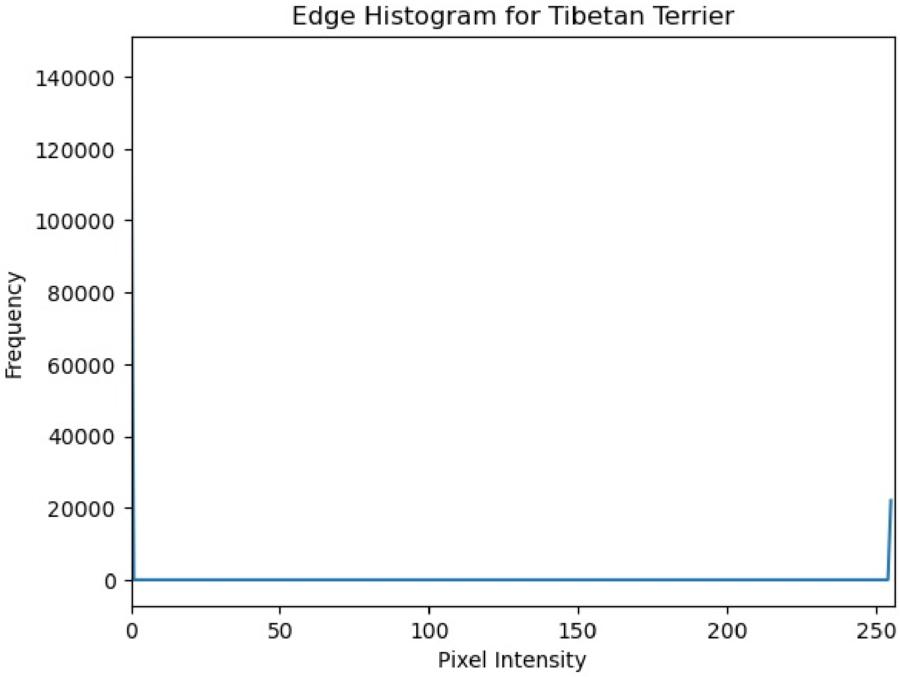
In [41]:

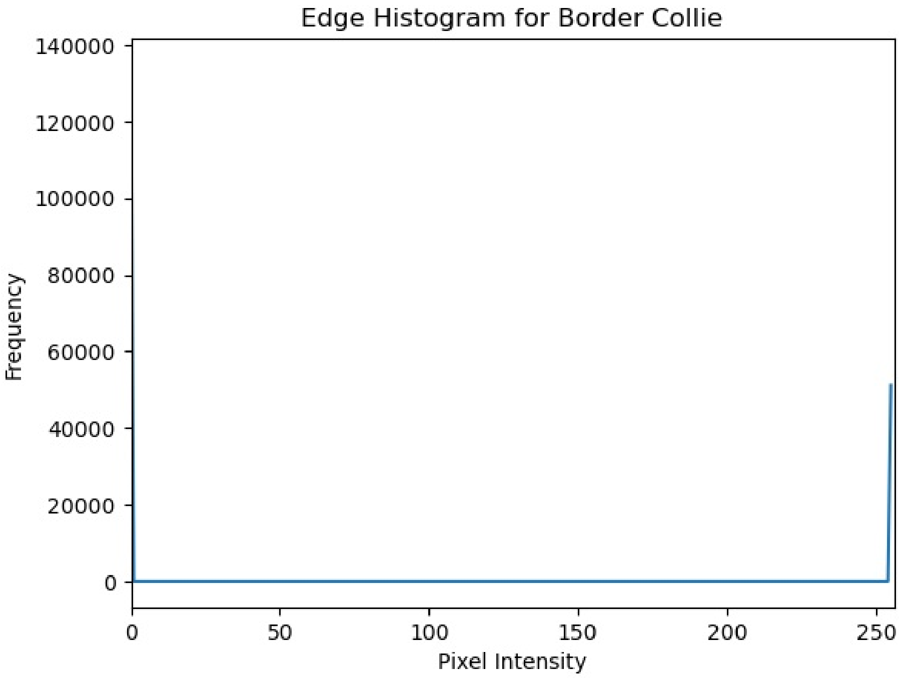
```
import os
import cv2
import numpy as np
import matplotlib.pyplot as plt
from pathlib import Path
def load_images_from_class(class_dir):
    """Load all images from a given directory."""
    images = []
    for filename in os.listdir(class_dir):
        if filename.endswith(('.jpg')):
            img_path = os.path.join(class_dir, filename)
            img = cv2.imread(img_path)
            if img is not None:
                images.append(img)
    return images
```

```python
def calculate_edge_histograms(images):
    """Calculate edge histograms for a list of images."""
    histograms = []
    for img in images:
        gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        edges = cv2.Canny(gray_img, 100, 200)
        hist = cv2.calcHist([edges], [0], None, [256], [0, 256])
        histograms.append(hist)
    return histograms
def process_classes(class_dirs):
    """Process images from all class directories and convert to edge histograms."""
    all_histograms = {}

    for class_name, class_dir in class_dirs.items():
        images = load_images_from_class(class_dir)
        histograms = calculate_edge_histograms(images)
        all_histograms[class_name] = histograms

    return all_histograms
class_dirs ={
    'Scottish Deerhound':(r'C:\Users\ADMIN\stanforddogs\images\n02092002-Scottish_deerhound'),
    'Irish Terrier':(r'C:\Users\ADMIN\stanforddogs\images\n02093991-Irish_terrier'),
    'Tibetan Terrier':(r'C:\Users\ADMIN\stanforddogs\images\n02097474-Tibetan_terrier'),
    'Border Collie':(r'C:\Users\ADMIN\stanforddogs\images\n02106166-Border_collie'),
}

edge_histograms = process_classes(class_dirs)
for class_name, histograms in edge_histograms.items():
    plt.figure()
    plt.title(f'Edge Histogram for {class_name}')
    plt.plot(histograms[0])
    plt.xlim([0, 256])
    plt.xlabel('Pixel Intensity')
    plt.ylabel('Frequency')

    plt.show()
```




Edge Histogram for Irish Terrier
Frequency
160000
140000
120000
100000
80000
60000
40000
20000
0
0
50
100
150
200
250
Pixel Intensity


Edge Histogram for Tibetan Terrier
Frequency
140000
120000
100000
80000
60000
40000
20000
0
0
50
100
150
200
250
Pixel Intensity



2d-iii)

In [49]:

```
from sklearn.decomposition import PCA
histograms = {
   r'C:\Users\ADMIN\stanforddogs\images\n02092002-Scottish_deerhound' : [np.random.rand(36) for _ in range(100)
   r'C:\Users\ADMIN\stanforddogs\images\n02093991-Irish_terrier':[np.random.rand(36) for _ in range(100)],
    r'C:\Users\ADMIN\stanforddogs\images\n02097474-Tibetan_terrier':[np.random.rand(36) for _ in range(100)],
    r'C:\Users\ADMIN\stanforddogs\images\n02106166-Border_collie':[np.random.rand(36) for _ in range(100)],
    }

hist_list = [hist for class_hists in histograms.values() for hist in class_hists]
pca = PCA(n_components=2)
hist_reduced = pca.fit_transform(hist_list)
for idx, class_name in enumerate(histograms):
    num_hists = len(histograms[class_name])
    print(f"Reduced Histograms for {class_name}: {hist_reduced[idx * num_hists:(idx + 1) * num_hists]}")
```

```
Reduced Histograms for C:\Users\ADMIN\stanforddogs\images\n02092002-Scottish_deerhound: [[-2.49941524e-01 -2.764
07618e-01]
 [ 6.87274148e-01 -5.68891633e-01]
 [-4.09748652e-01  3.01091400e-02]
 [-1.01903052e+00  4.72920260e-02]
 [-2.21117812e-01  5.04017502e-01]
 [-8.79454628e-02  7.49593764e-03]
 [ 7.61269562e-02 -5.71589271e-02]
 [-2.22727563e-02  2.55681286e-02]
 [ 2.40659243e-01  3.04402612e-02]
 [ 5.67923571e-02  1.28299836e-01]
 [ 7.18336015e-01 -2.13162444e-02]
 [ 9.94498554e-01 -3.59748770e-01]
 [ 2.42608001e-01 -3.60560229e-01]
 [ 2.31326026e-01  3.96903041e-01]
 [ 4.85784768e-02 -3.08806884e-01]
 [ 1.13686691e-01 -2.64953331e-01]
 [ 4.79305095e-02 -2.08519111e-01]
 [ 2.82377639e-01  2.56393833e-01]
 [ 2.98728541e-01  1.83469174e-01]
 [-3.32232607e-01 -1.77985590e-01]
 [-4.38409142e-01 -7.60221402e-02]
 [-3.12116518e-01 -3.75278863e-01]
 [ 5.57948481e-01 -4.78274166e-02]
 [ 2.20218135e-01 -2.52100700e-01]
 [-5.89031917e-01  2.95492165e-01]
 [-8.46762412e-01 -7.86779252e-02]
 [-3.17413429e-01 -8.90916782e-01]
 [-2.91945305e-01  3.42197513e-01]
 [ 1.34217193e-01 -1.94121448e-01]
 [ 3.41271049e-01  2.35869432e-01]
 [ 3.07691510e-01  2.28162509e-02]
 [-4.27690794e-01 -4.85710229e-01]
 [-2.35061377e-01 -3.98843298e-01]
 [-8.65004587e-02 -1.14037996e-01]
 [ 7.96585398e-02 -4.93145616e-01]
 [-3.16018890e-01 -3.20264404e-01]
 [ 1.21700521e-01  2.83105213e-01]
 [-2.76734063e-01  1.54577285e-01]
 [ 5.79835270e-02  3.88828362e-01]
 [-3.82374712e-01  3.29214100e-01]
 [-4.43071682e-01 -5.73363763e-01]
 [ 1.65487561e-01  2.25152949e-01]
 [ 9.47643388e-02 -2.49467519e-01]
 [ 5.52339731e-01 -3.62242175e-01]
 [-1.93386796e-01  1.92677383e-01]
 [-4.14090255e-02  5.00600528e-02]
 [-6.17804795e-01  7.13664431e-01]
 [-3.85167475e-01  3.20678353e-03]
 [ 1.20805648e-01  1.44462380e-01]
 [ 1.62946921e-01 -7.40987571e-02]
 [ 2.88551164e-01 -3.04258506e-02]
 [-7.35834593e-01 -7.02979899e-04]
 [ 1.67438743e-01  3.54457486e-01]
 [-3.30774996e-02  4.55016690e-01]
 [ 6.28345632e-03  1.57773515e-02]
 [ 1.08569715e-01  2.54786582e-01]
 [-1.77171089e-01 -6.46152951e-01]
 [ 1.93303320e-02  4.80993083e-01]
 [ 8.73226778e-02  2.91520780e-01]
 [-3.13650423e-03 -8.61192850e-01]
 [ 1.84468805e-01 -2.35721289e-01]
 [-1.46490425e-01  2.14155747e-01]
 [ 7.17821286e-02  2.62370367e-01]
 [ 6.05596453e-01  1.43148502e-01]
 [ 2.14364920e-02 -2.42730943e-01]
 [-7.58057892e-01 -3.64928279e-01]
 [-1.69108556e-01  6.60594920e-01]
 [-2.49570443e-01  1.44011754e-02]
 [-9.07193144e-01 -3.70789257e-01]
 [-4.76734069e-01 -3.36061628e-01]
 [-4.19096332e-01 -1.27540881e-01]
 [ 1.60675453e-01 -2.17842748e-01]
 [-4.15680217e-01 -6.39376299e-02]
 [ 2.05995229e-01 -8.47898564e-02]
 [ 1.89002264e-01  8.29792997e-03]
 [ 5.10782510e-01 -2.34615827e-01]
 [-1.47823817e-01 -3.32636993e-01]
 [ 1.50416539e-01  5.93030036e-02]
 [-4.88686181e-01 -8.46797390e-02]
 [ 1.91532726e-01  1.05904042e-01]
 [ 2.62457113e-01  2.37734866e-01]
 [ 2.96490093e-01 -1.76466768e-01]
 [ 1.78038698e-01 -2.34718241e-01]
 [ 2.03780151e-02  6.46191774e-01]
 [-1.72058662e-01  2.11242861e-01]
 [-2.09202571e-01  2.04174450e-01]
 [-1.07730779e-01  7.66094269e-03]
 [-6.11522739e-02  3.34501939e-01]
 [-4.04028509e-01  7.15196078e-01]
 [-4.78747686e-02  1.29624849e-01]
 [ 1.94328745e-01 -2.42102800e-01]
 [ 5.06048173e-01  7.27462709e-01]
 [-8.34701231e-02 -1.98944695e-01]
 [ 3.65229312e-01  7.68339418e-02]
 [-3.39101193e-01  1.16056110e-01]
 [ 5.03587267e-01 -6.91105002e-01]
 [ 2.46493581e-01  4.15306605e-01]
 [-1.47397041e-02 -6.02342571e-01]
 [ 2.29665050e-01  5.70040153e-01]
 [-6.02745468e-01  5.57629751e-02]]
Reduced Histograms for C:\Users\ADMIN\stanforddogs\images\n02093991-Irish_terrier: [[ 0.85050013  0.01485223]
 [ 0.63304203 -0.00121579]
 [ 0.38559159  0.29384704]
 [ 0.68907876 -0.006306  ]
 [ 0.24573722 -0.14120471]
 [ 0.01301079 -0.1806394 ]
 [-0.11062915 -0.19221602]
 [-0.16178363 -0.30616722]
 [-0.68494926  0.16392351]
 [-0.01269663 -0.08614267]
 [-0.26170896  0.09520465]
```

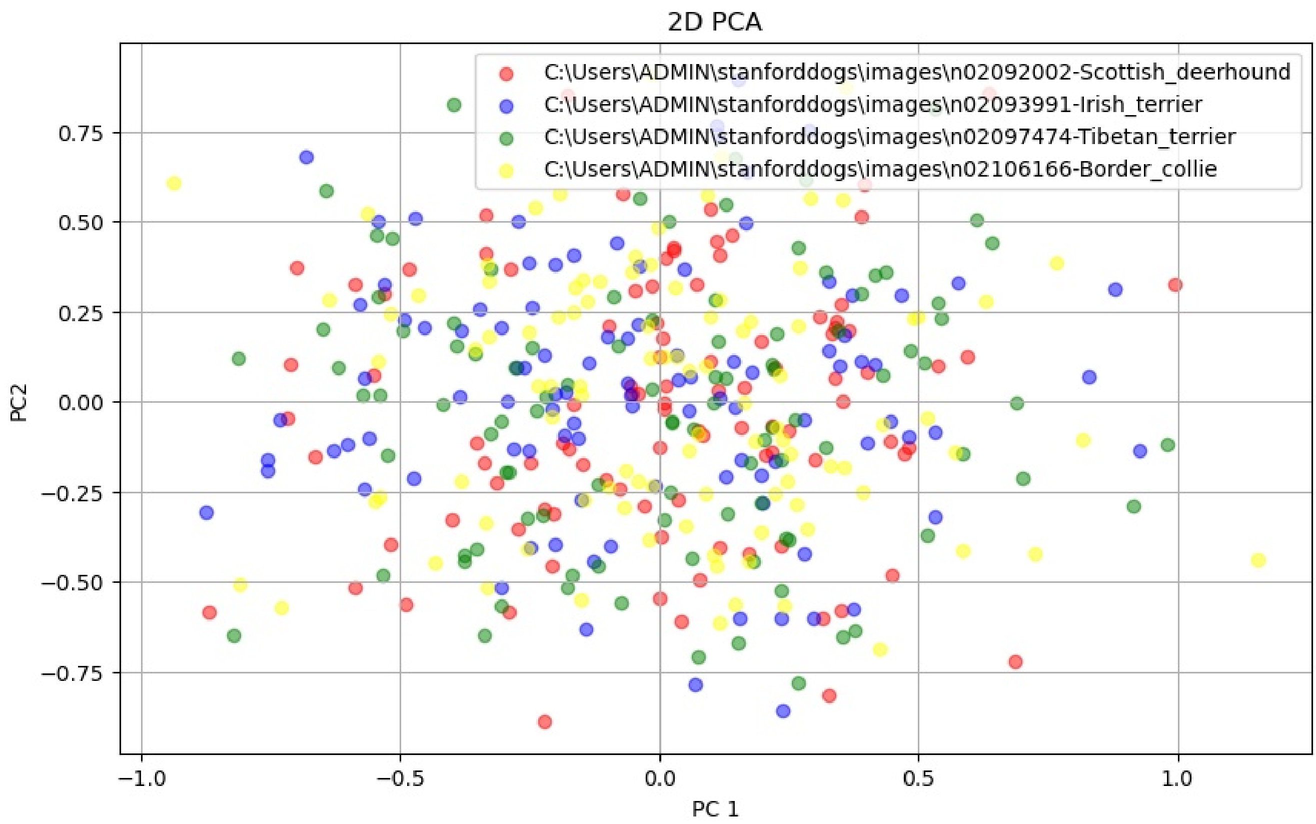
```
[-0.1107818  -0.01827489]
[-0.35594324  0.48304578]
[ 0.23479519  0.09588894]
[-0.37497105  0.08076522]
[-0.54123075 -0.60946153]
[-0.14062288 -0.27387391]
[ 0.34188604 -0.26675385]
[-0.1014285   0.32070784]
[ 0.43968099  0.35069722]
[ 0.21213561  0.30056264]
[ 0.21790346  0.01408619]
[ 0.46646241 -0.45216334]
[-0.40671805  0.29415984]
[ 0.11045514 -0.00230576]
[-0.22732525  0.05673308]
[-0.38800605 -0.15796635]
[ 0.18023236  0.37006991]
[ 0.63355308  0.44098327]
[ 0.32694231 -0.0692265 ]
[ 0.05500456  0.68072482]
[ 0.24184851  0.14625291]
[ 0.26132398 -0.40299212]
[-0.66858534  0.50087807]
[ 0.15973022  0.43087469]
[ 0.28996748 -0.0089379 ]
[ 0.76760885  0.4112231 ]
[-0.13647142 -0.51692417]
[-0.00865099  0.0930405 ]
[ 0.08510223  0.48561584]
[-0.23791975  0.24341423]
[-0.25589544 -0.42243186]
[-0.51867102  0.33033208]
[ 0.15701253 -0.08575898]
[ 0.11356562 -0.28746251]
[-0.0360045   0.21138022]
[-0.30067319 -0.06989576]
[ 0.02440387 -0.39363576]
[ 0.11265332 -0.05477852]
[-0.23445712 -0.2708788 ]
[ 0.72473841 -0.0331379 ]
[-0.68288122 -0.07063903]
[-0.19991201 -0.33000167]
[ 0.01824239 -0.10074474]
[ 0.24395516 -0.63738755]
[ 0.44615622 -0.26267585]
[-0.35580893 -0.26417614]
[-0.53735974  0.19002679]
[ 0.16711459 -0.90451519]
[-0.016173    0.6590689 ]
[ 0.33988703  0.55682432]
[-0.62768625 -0.23423693]
[ 0.39174014 -0.10558738]
[ 0.53121989  0.04333291]
[ 0.16484831  0.10434464]
[-0.33863146  0.35053499]
[ 0.0504388   0.40582612]
[-0.69015306  0.19481932]
[-0.16175479 -0.38433562]
[-0.07248061  0.13505366]
[-0.10660634  0.47083044]
[ 0.00715634  0.86238735]
[ 0.31167543 -0.13616416]
[-0.61764849 -0.06389797]
[-0.04292877  0.77204174]
[ 0.54874743  0.26263697]
[ 0.20994851 -0.0904012 ]
[-0.12108975 -0.08414783]
[ 0.05439529  0.38586714]
[ 0.04055087  0.60136643]
[-0.03569849 -0.50489656]
[-0.04200663 -0.10329446]
[ 0.81060047  0.17183098]
[ 0.09211838 -0.37129337]
[-0.33884983  0.30505119]
[ 0.21157012  0.20656364]
[-0.04741298 -0.00106559]
[-0.25249792 -0.04738219]
[ 0.58725009 -0.14935376]
[ 0.12656555 -0.31430228]
[ 0.03934746 -0.54984235]
[-0.38584057 -0.36145018]
[-0.70404451 -0.17655224]
[ 0.91593875 -0.43694468]
```

```
 [-0.15655665 -0.2127539 ]
 [-0.14177957 -0.20325271]
 [-0.20007391  0.315347  ]
 [ 0.18051574 -0.57016687]
 [-0.4229741  -0.19202654]
 [-0.07461032 -0.38791374]]
Reduced Histograms for C:\Users\ADMIN\stanforddogs\images\n02097474-Tibetan_terrier: [[ 0.55270699 -0.0617065 ]
 [ 0.44640908  0.51656676]
 [ 0.16797205  0.17582049]
 [-0.47325851  0.57616594]
 [-0.19694414 -0.58142308]
 [-0.46783427  0.09532055]
 [ 0.00152256  0.00204042]
 [-0.28638334  0.20999468]
 [ 0.06197399 -0.23897521]
 [-0.26933475  0.1465457 ]
 [-0.15986355 -0.16209436]
 [ 0.17351066 -0.33782378]
 [ 0.41394813  0.36892014]
 [ 0.15823963  0.7584789 ]
 [ 0.1114934  -0.3311172 ]
 [ 0.06374753 -0.03116389]
 [-0.20931955 -0.15718504]
 [-0.63211991 -0.21005126]
 [-0.06718501  0.07598394]
 [-0.06449343 -0.17086121]
 [-0.62499659 -0.17058649]
 [-0.06783538  0.56283278]
 [-0.34621544 -0.3980809 ]
 [-0.34047057 -0.26960292]
 [-0.50278482 -0.34816245]
 [ 0.14209021  0.07354553]
 [-0.23557049 -0.66170063]
 [-0.61381142 -0.034257  ]
 [ 0.14703659 -0.35757706]
 [-0.40405334  0.19889281]
 [-0.19339155 -0.23679932]
 [ 0.09142166 -0.08816982]
 [-0.01334869  0.96070211]
 [-0.13098248 -0.14771526]
 [ 0.65949655  0.17406623]
 [ 0.49253164 -0.16199176]
 [-0.99736557  0.0744956 ]
 [ 0.24214713 -0.20923234]
 [-0.01540335 -0.1912388 ]
 [ 0.36905638  0.17535189]
 [ 0.43478882 -0.78820729]
 [-0.01824625  0.65256119]
 [-0.5265583   0.12581642]
 [-0.21399603  0.25153974]
 [ 0.06702957 -0.08331969]
 [ 0.05973426  0.31392302]
 [-0.27967423 -0.10656233]
 [ 0.09790959  0.31165987]
 [ 0.03268542 -0.03489765]
 [-0.06610938  0.27022172]
 [ 0.82733422  0.16161564]
 [-0.27925486 -0.23866773]
 [-0.17875129 -0.02477764]
 [ 0.20856138  0.27914111]
 [-0.79857766  0.14233581]
 [-0.26451083 -0.13484172]
 [-0.09449201 -0.29165091]
 [ 0.21960228  0.04768509]
 [-0.10112255 -0.64024507]
 [-0.33867263 -0.13977672]
 [ 0.49236417  0.54734007]
 [ 0.28968889  0.24987608]
 [ 0.26023736  0.45134354]
 [ 0.27428161 -0.55832908]
 [-0.30300462 -0.0725487 ]
 [ 0.06774152 -0.00358247]
 [ 0.42733448  0.62351589]
 [ 0.1327459  -0.20453707]
 [-0.56514854  0.31390085]
 [-0.02024673  0.38952119]
 [ 0.5527696  -0.3546158 ]
 [ 0.68715802  0.53387158]
 [-0.1198268   0.09348502]
 [ 0.19479147  0.05925645]
 [-0.04204661  0.15912413]
 [-0.08689268  0.53667562]
 [ 0.22389634  0.42577942]
 [ 0.37750821 -0.10987821]
 [ 0.14322819 -0.46921073]
 [-0.29676237  0.48401229]
 [ 0.45656733 -0.40363459]
 [-0.27438023 -0.16793451]
 [-0.28191882 -0.208253  ]
 [-0.21867889 -0.40138971]
 [-0.23928319  1.11190148]
 [-0.36503138 -0.34727312]
 [ 0.50080872  0.16036403]
 [ 0.26012346  0.5982086 ]
 [-0.31516247  0.6268255 ]
 [-0.1949259  -0.24752985]
 [ 0.63432266 -0.07526498]
 [-0.35640356  0.66586057]
 [-0.10567876 -0.13346815]
 [ 0.19996694 -0.19770531]
 [ 0.16770224 -0.17661698]
 [-0.51142597 -0.01351655]
 [-0.43739516 -0.14548996]
 [ 0.15104988  0.12563485]
 [ 0.08346928  0.67742128]
 [ 0.09140051 -0.50441195]]
Reduced Histograms for C:\Users\ADMIN\stanforddogs\images\n02106166-Border_collie: [[-0.14220553 -0.94146067]
 [ 0.43790615  0.50591484]
 [ 0.13108373  0.12260915]
 [ 0.4316282  -0.78618615]
 [ 0.41837625 -0.05423182]
 [ 0.01883279 -0.57716391]
 [-0.19098159 -0.699701  ]
 [ 1.02818576  0.02747972]
 [ 0.06066933  0.57170343]
 [ 0.09272983  0.2014871 ]
 [ 0.44738121  0.05193423]
 [ 0.39151812 -0.2761084 ]
 [-0.04117413 -0.16890519]
 [-0.57466334 -0.38026809]
 [-0.41613751  0.04180809]
 [-0.00908995  0.12713755]
 [ 0.1606344   0.81882254]
 [ 0.08717892 -0.30745804]
 [ 0.31874354  0.18588154]
 [ 0.43784024  0.53568256]
 [-0.0644069   0.08776092]
 [ 0.86329193 -0.37472943]
 [-0.0266763   0.0715314 ]
 [ 0.11532219 -0.6778057 ]
 [ 0.26444555 -0.07045537]
 [ 0.39253184 -0.25829862]
 [ 0.28312143  0.44376035]
 [ 0.0211854  -0.29361638]
 [-0.11236142 -0.28130295]
 [ 0.03686956  0.13874056]
 [ 0.34595629 -0.22648603]
 [ 0.42080994  0.33487154]
 [-0.16215083  0.15991109]
 [-0.21669354 -0.91847045]
 [ 0.43609036 -0.13689815]
 [-0.0146417   0.40706259]
 [-0.61841808  0.28244535]
 [-0.04990147  0.08829503]
 [ 0.13532572  0.13731238]
 [-0.16224855  0.53954349]
 [-0.20900997  0.56949698]
 [ 0.13104947 -0.20885597]
 [-0.59464481  0.00846734]
 [ 0.81042133  0.14339403]
 [ 0.02340748  0.20335861]
 [ 0.12424964  0.18167783]
 [ 0.0715161   0.18914094]
 [-0.27862641  0.20402265]
 [-0.9319146   0.18909359]
 [-0.50724703  0.36453306]
 [-0.23750874  0.35678614]
 [ 0.65652142 -0.12485743]
 [-0.37528628  0.43023403]
 [-0.83000069 -0.14869666]
 [-0.58181791 -0.34864594]
 [-0.22348295  0.02091192]
 [-0.56992107  0.39109372]
 [ 0.07309168 -0.29167739]
 [-0.24347991  0.53309694]
 [-0.41388826  0.27688919]
 [-0.41546257  0.34651206]
 [ 0.31533693 -0.04159163]
 [ 0.15432809 -0.37548097]
 [ 0.1896925   0.16647313]
 [ 0.40978288 -0.35985249]
 [-0.23649471 -0.40740102]
 [-0.40152568 -0.25624779]
 [-0.43037789 -0.01654159]
 [-0.0980704  -0.32849591]
 [ 0.03027705  0.17280139]
 [-0.01092914  0.13277043]
 [-0.4871005  -0.128727  ]
 [-0.13603279 -0.16825599]
 [-0.10695157 -0.03278678]
 [ 0.51046263  0.47253484]
 [ 0.09274171 -0.10984219]
 [ 0.16965125 -0.27755945]
 [-0.00688567  0.62642268]
 [ 0.72528094 -0.41141717]
 [ 0.17801526 -0.32337456]
 [ 0.16401466 -0.01049209]
 [ 0.52170036 -0.28658351]
 [ 0.03150644 -0.41054704]
 [ 0.45378409 -0.70279277]
 [-0.54842397 -0.78596263]
 [-0.59673584  0.07047203]
 [-0.06526198  0.68096436]
 [ 0.53788863 -0.29979168]
 [ 0.16106323  0.23911733]
 [ 0.20489417 -0.50133179]
 [ 0.53894869  0.32779888]
 [-0.20947111 -0.2539595 ]
 [ 0.16502777  0.06880302]
 [-0.05277915  0.31250314]
 [ 0.02209119 -0.24481909]
 [ 0.81783879  0.15960779]
 [-0.61492302 -0.06359217]
 [ 0.1381919  -0.80179133]
 [ 0.50048745 -0.06066005]
 [-0.01914855 -0.33878171]]
```

2d-iv)

In [55]:

```python
import matplotlib.pyplot as plt
import numpy as np
from sklearn.decomposition import PCA
histograms = {
    r'C:\Users\ADMIN\stanforddogs\images\n02092002-Scottish_deerhound' : [np.random.rand(36) for _ in range(100
    r'C:\Users\ADMIN\stanforddogs\images\n02093991-Irish_terrier':[np.random.rand(36) for _ in range(100)],
    r'C:\Users\ADMIN\stanforddogs\images\n02097474-Tibetan_terrier':[np.random.rand(36) for _ in range(100)],
    r'C:\Users\ADMIN\stanforddogs\images\n02106166-Border_collie':[np.random.rand(36) for _ in range(100)],
}
hist_list = [hist for class_hists in histograms.values() for hist in class_hists]
hist_array = np.array(hist_list)
pca = PCA(n_components=2)
hist_reduced = pca.fit_transform(hist_array)

# Plotting
colors = ['red', 'blue', 'green', 'yellow']
labels = list(histograms.keys())
plt.figure(figsize=(10, 6))
start_idx = 0

for idx, (class_name, class_hists) in enumerate(histograms.items()):
    end_idx = start_idx + len(class_hists)
    plt.scatter(hist_reduced[start_idx:end_idx, 0],
                hist_reduced[start_idx:end_idx, 1],
                c=colors[idx], label=class_name, alpha=0.5)
    start_idx = end_idx

plt.xlabel('PC 1')
plt.ylabel('PC2')
plt.title('2D PCA')
plt.grid(True)
plt.legend()
plt.show()
```



3.

In [46]:

```
import pandas as pd

json_path = r'C:\Users\ADMIN\data\train_.json.json'
data = pd.read_json(json_path)

print(data.head())
```

```
               ID                                              Tweet  anger  \
0  2017-En-10065  In 2016, Black people are STILL fighting to be...   True   
1  2017-En-21745  @Justin_Gau @JamesMelville You certainly would...   True   
2  2017-En-21992  If you follow #Trump, a certified #bully there...   True   
3  2017-En-21483  @Darren32895836 @FatimaFatwa it would be a gre...  False   
4  2017-En-40140  I forgot my hair straightner home, I'm feeling...   True   

   anticipation  disgust   fear    joy   love  optimism  pessimism  sadness  \
0         False     True  False  False  False     False      False    False   
1         False     True   True  False  False     False      False    False   
2         False     True  False  False  False     False      False    False   
3         False    False   True   True  False      True      False    False   
4         False     True  False  False  False     False       True     True   

   surprise  trust  
0     False  False  
1     False  False  
2     False  False  
3     False  False  
4     False  False  
```

4.

In [60]:

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import numpy as np
json_path = r'C:\Users\ADMIN\data\train_.json.json'
data = pd.read_json(json_path)
texts = data['Tweet'].tolist()

labels = data[['anger', 'anticipation', 'disgust', 'fear', 'joy',
               'love', 'optimism', 'pessimism', 'sadness',
               'surprise', 'trust']].astype(str).agg(','.join, axis=1).tolist()

count_vectorizer = CountVectorizer()
X_count = count_vectorizer.fit_transform(texts)
tfidf_vectorizer = TfidfVectorizer()
```

```
X_tfidf = tfidf_vectorizer.fit_transform(texts)
print("Count Vectorizer dimensions:", X_count.shape)
print("TF-IDF Vectorizer dimensions:", X_tfidf.shape)
```

```
Count Vectorizer dimensions: (3000, 9633)
TF-IDF Vectorizer dimensions: (3000, 9633)
```

5.

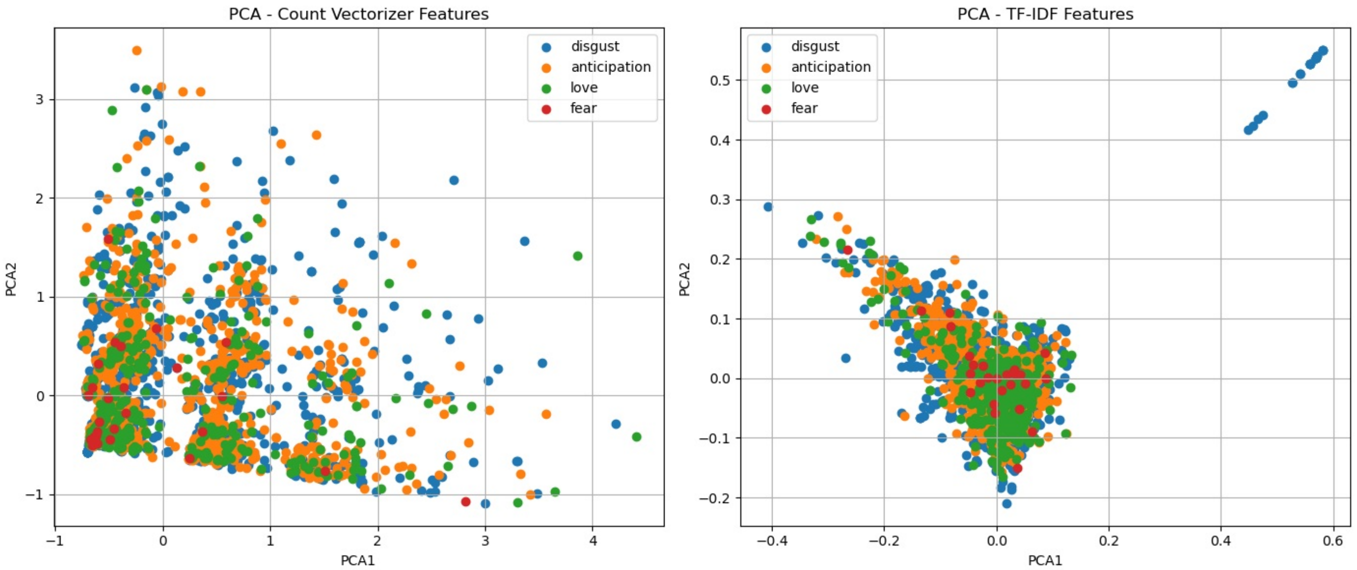
In [92]:

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
json_path = r'C:\Users\ADMIN\data\train_.json.json'
data = pd.read_json(json_path)
texts = data['Tweet'].tolist()
emotions = []
for entry in data.itertuples(index=False):
    if entry.joy:
        emotions.append('disgust')
    elif entry.anger:
        emotions.append('anticipation')
    elif entry.sadness:
        emotions.append('love')
    elif entry.surprise:
        emotions.append('fear')
    else:
        emotions.append('Other')
df = pd.DataFrame({'Tweet': texts, 'class': emotions})
selected_class = ['disgust', 'anticipation', 'love', 'fear']
df_filtered = df[df['class'].isin(selected_class)
print("\nSelected Classes:")
for emotion in selected_class:
    print(f"- {emotion}")

count_vectorizer = CountVectorizer()
count_matrix = count_vectorizer.fit_transform(df_filtered['Tweet'])

tfidf_vectorizer = TfidfVectorizer()
tfidf_matrix = tfidf_vectorizer.fit_transform(df_filtered['Tweet'])

pca_count = PCA(n_components=2)
reduced_count_data = pca_count.fit_transform(count_matrix.toarray())

pca_tfidf = PCA(n_components=2)
reduced_tfidf_data = pca_tfidf.fit_transform(tfidf_matrix.toarray())
reduced_count_df = pd.DataFrame(data=reduced_count_data, columns=['PCA1', 'PCA2'])
reduced_count_df['Emotion'] = df_filtered['class'].reset_index(drop=True)

reduced_tfidf_df = pd.DataFrame(data=reduced_tfidf_data, columns=['PCA1', 'PCA2'])
reduced_tfidf_df['Emotion'] = df_filtered['class'].reset_index(drop=True)

print("\nReduced PCA Data (Count Vectorizer - first 10 entries):")
print(reduced_count_df.head(10))

print("\nReduced PCA Data (TF-IDF - first 10 entries):")
print(reduced_tfidf_df.head(10))
```

```
Selected Classes:
- disgust
- anticipation
- love
- fear

Reduced PCA Data (Count Vectorizer - first 10 entries):
       PCA1      PCA2       Emotion
0 -0.265033  0.454415  anticipation
1  0.214252 -0.157902  anticipation
2 -0.413439  1.462692  anticipation
3  0.562995  0.458395       disgust
4 -0.598724 -0.292262  anticipation
5 -0.373909 -0.299743  anticipation
6 -0.290709  0.771711       disgust
7  1.045075 -0.514003       disgust
8  0.837460  1.036364       disgust
9  0.651966 -0.420258       disgust

Reduced PCA Data (TF-IDF - first 10 entries):
       PCA1      PCA2       Emotion
0 -0.021681  0.004509  anticipation
1 -0.030180  0.051719  anticipation
2 -0.087973  0.172641  anticipation
3 -0.018387 -0.026111       disgust
4  0.018268 -0.048186  anticipation
5 -0.009858  0.000849  anticipation
6 -0.016690  0.008065       disgust
7 -0.003913 -0.080778       disgust
8 -0.091040  0.068190       disgust
9 -0.012502 -0.066799       disgust
```

In [102…

```python
plt.figure(figsize=(14, 6))
plt.subplot(1, 2, 1)
for emotion in selected_classes:
    subset = reduced_count_df[reduced_count_df['Emotion'] == emotion]
    plt.scatter(subset['PCA1'], subset['PCA2'], label=emotion)

plt.title('PCA - Count Vectorizer Features')
plt.xlabel('PCA1')
plt.ylabel('PCA2')
plt.legend()
plt.grid()
plt.subplot(1, 2, 2)
for emotion in selected_classes:
    subset = reduced_tfidf_df[reduced_tfidf_df['Emotion'] == emotion]
    plt.scatter(subset['PCA1'], subset['PCA2'], label=emotion)

plt.title('PCA - TF-IDF Features')
plt.xlabel('PCA1')
plt.ylabel('PCA2')
plt.legend()
plt.grid()

plt.tight_layout()
plt.show()
```



Since all classes (disgust, anticipation, love, fear) have overlapping points in both the PCA plots for token count features and TF-IDF features, it suggests that the features derived from data do not effectively distinguish between these emotions in the reduced dimensionality. so,none of the classes are visually separable (non-overlapping) for both plots.

Github link : https://github.com/srinidhireddy09/stanforddogs-assignment

Loading [MathJax]/extensions/Safe.js