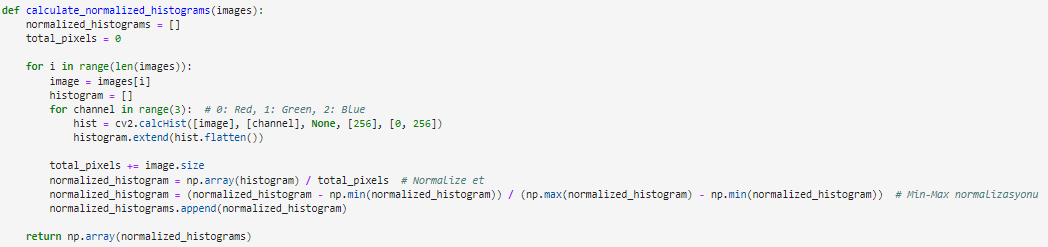
Content based Image Retrieval with Python

**Main Topic:** This study presents a method for assessing the performance of an image classification model. Utilizing a histogram-based approach, the five nearest training examples for each class were identified, and their similarity was measured using Manhattan distance. The results provide a valuable evaluation to understand and enhance the effectiveness of the model in each class. Offering a different perspective on the analysis of image classification model performance, the study provides insights into both strengths and weaknesses, serving as a valuable resource for comprehending and improving such models.

**Code Explanation:**  
 **1. Histogram Calculation and Normalization (Lines 11-26):**

This function calculates normalized histograms for a given set of images. For each image, it computes separate histograms for the Red, Green, and Blue channels using **cv2.calcHist()**. Then, it normalizes the histograms by dividing them by the total number of pixels in the image. The final step is Min-Max normalization to ensure values are in the range [0, 1].



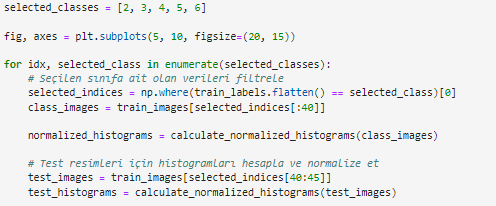
**2. Loading CIFAR-10 Dataset (Lines 29-30):**

This code loads the CIFAR-10 dataset using TensorFlow's **cifar10.load\_data()** function. It splits the dataset into training and testing sets.



**3. Class Selection and Index Filtering (Lines 33-41):**

Selected classes (bird, cat, deer, dog, frog) are specified, and for each class, the indices corresponding to that class in the training labels are filtered. The first 40 images for each class are then selected.

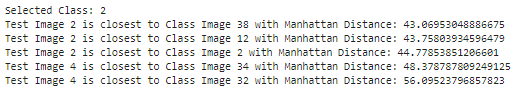


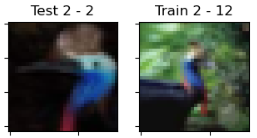
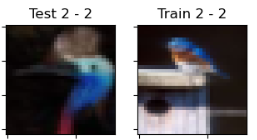
**4. Manhattan Distance Calculation and Visualization (Lines 45-70):**

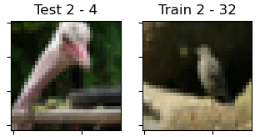
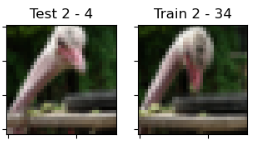
For each test histogram, the Manhattan distance is calculated with each training histogram, and the distances are stored. These distances are then sorted, and the five closest images are visualized in pairs: the left side shows test images, and the right side shows their corresponding closest training images. This process is repeated for each selected class. The final visualization is displayed using **plt.show()**

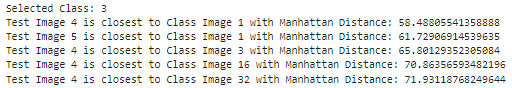


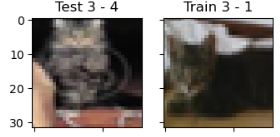
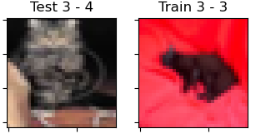
**Outputs**:

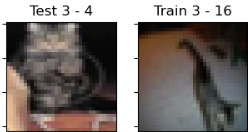
Class 2: Bird



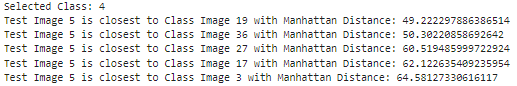


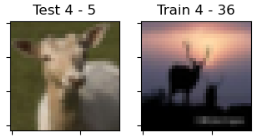
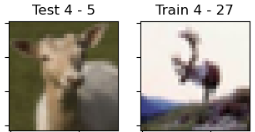
Class 3: Cat



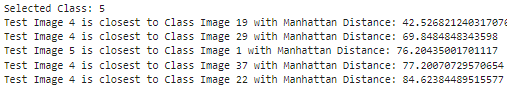


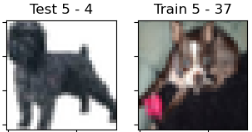
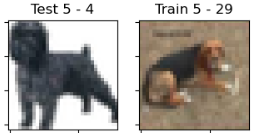
Class 4: Deer

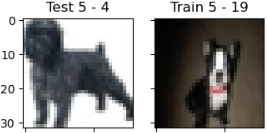


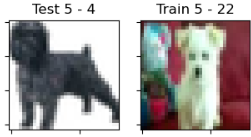


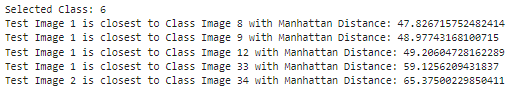


Class 5: Dog







Class 6: Frog

