Name: Nayan Biramane

Roll No: 21

Aim: To Creating and Training an Object Detector

Objective: Bag of Words BOW in computer version Detecting cars in a scene

Theory :

Creating and Training an object detector:-
Using built-in features makes it easy to come up with a quick prototype for an application. And we're all very grateful to the OpenCV developers for making great features, such as face detection or people detection readily available (truly, we are). However, whether you are a hobbyist or a computer vision professional, it's unlikely that you will only deal with people and faces:

Bag-of -words:-

Bag-of-Words (BoW) is a text representation technique in natural language processing (NLP) where a text (such as a sentence or a document) is represented as an unordered set of words, disregarding grammar and word order but keeping track of the frequency of each word. It is a simple and commonly used way to convert text data into numerical feature vectors that can be used by machine learning algorithms.

BOW in Computer Vision:-

Instead, in computer vision, the BoW model is an approach used for image classification and object recognition. It is used to represent images as histograms of

visual words and is particularly popular in tasks such as image retrieval and object recognition in scenes.
Interest Point Detection: Detect interesting points or regions in the images, often using techniques like Scale-Invariant Feature Transform (SIFT), Speeded-Up Robust Features (SURF), or Oriented FAST and Rotated BRIEF (ORB).
Feature Descriptor: Extract feature descriptors (often small image patches) around these interest points. These descriptors encode information about the local texture and intensity gradients of the region around each interest point.

For each image, create a histogram of visual words. Count the occurrences of each visual word (cluster) in the image.
The histogram, representing the frequency of each visual word, is the BoW representation of the image.

Detecting Cars:-

To detect cars in images using the Bag-of-Words (BoW) approach, you'll need a dataset of car images, extract features from these images, create a visual vocabulary, and use machine learning techniques for classification.

Example:-

Here's a step-by-step guide to detecting cars using the BoW model:

1. Dataset:
Collect a dataset of images containing cars. Make sure the dataset is diverse and contains various angles, sizes, and lighting conditions of cars.

2. Feature Extraction:

Use a feature extraction method like SIFT, SURF, or ORB to detect interest points and extract feature descriptors from the car images. Each descriptor represents a unique part of the image.

3. Codebook Creation (Vocabulary):
Apply clustering algorithms (such as K-means clustering) on the extracted descriptors to create a visual vocabulary. The resulting cluster centers will represent visual words.

4. Vector Quantization:
For each car image, assign its feature descriptors to the nearest cluster center to create a histogram of visual words (BoW representation).

5. Training Data Preparation:
Prepare training data by creating BoW histograms for positive samples (car images) and negative samples (non-car images). Ensure a balanced dataset.

6. Machine Learning Classifier:
Train a machine learning classifier (e.g., Support Vector Machine, Random Forest) using the BoW histograms as features and corresponding labels (car or non-car). This step involves supervised learning where the classifier learns to distinguish between car and non-car images.

7. Testing and Detection:
Extract BoW histograms from new images.
Use the trained classifier to predict whether each image contains a car or not based on its BoW representation.
Apply non-maximum suppression to remove duplicate detections and keep only the most confident ones.

Code:-

```
import cv2
import numpy as np
from sklearn.cluster import KMeans
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler

# Step 1: Feature Extraction (SIFT)
def extract_features(image_path):
    sift = cv2.SIFT_create()
    image = cv2.imread("/content/img.jpg")
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    keypoints, descriptors = sift.detectAndCompute(gray, None)
    return descriptors

# Step 2: Codebook Creation (K-means clustering)
def create_codebook(descriptors, num_clusters):
    kmeans = KMeans(n_clusters=num_clusters)
    kmeans.fit(descriptors)
    return kmeans

# Step 3: Histogram Generation
def generate_histogram(image_path, kmeans):
    descriptors = extract_features(image_path)
    if descriptors is None:
        return None
    labels = kmeans.predict(descriptors)
    histogram = np.bincount(labels, minlength=kmeans.n_clusters)
    return histogram
```

```
# Training data (positive and negative samples)
positive_samples = ["car1.jpeg", "car2.jpg", "car3.webp", "car4.jpg","car5.webp"]
# Images containing cars
negative_samples = ["noncar1.jpg", "notcar2.jpg", "notcar.jpg", "notcar1.webp",
"notcar3.jpg", "notcar4.jpg", "notcar5.png"]  # Images without cars

# Extract features and create descriptors for positive and negative samples
positive_descriptors = np.vstack([extract_features(image) for image in
positive_samples if extract_features(image) is not None])
negative_descriptors = np.vstack([extract_features(image) for image in
negative_samples if extract_features(image) is not None])

# Step 2: Codebook Creation
num_clusters = 100  # Number of clusters (visual words)
kmeans = create_codebook(np.vstack([positive_descriptors, negative_descriptors]),
num_clusters)

# Step 3: Histogram Generation (Training Set)
positive_histograms = [generate_histogram(image, kmeans) for image in
positive_samples if generate_histogram(image, kmeans) is not None]
negative_histograms = [generate_histogram(image, kmeans) for image in
negative_samples if generate_histogram(image, kmeans) is not None]

# Prepare training data and labels
X_train = np.vstack([positive_histograms, negative_histograms])
y_train = np.hstack([np.ones(len(positive_histograms)),
np.zeros(len(negative_histograms))])

# Step 4: Training a Classifier (SVM)
scaler = StandardScaler().fit(X_train)
X_train_scaled = scaler.transform(X_train)
```

```
svm_classifier = SVC(kernel='linear')
svm_classifier.fit(X_train_scaled, y_train)

# Step 5: Object Detection (Testing)
test_image_path = "test_image.jpg"  # Image in which cars need to be detected
test_histogram = generate_histogram(test_image_path, kmeans)
if test_histogram is not None:
    test_histogram_scaled = scaler.transform(test_histogram.reshape(1, -1))
    prediction = svm_classifier.predict(test_histogram_scaled)
    if prediction == 1:
        print("Car detected!")
    else:
        print("No car detected.")
else:
    print("No keypoints found in the test image.")
```

Input Image:-

Output:-
Car detected!

Input Image:-

Output:-

No car detected.