CourseName: Computer Vision Lab **Course Code:** CSP-422

Experiment: 2.2

Aim: Write a program to interpret the effectiveness of Bag of Features in enhancing image classification performance.

Software Required: Any Python IDE (e.g.: PyCharm, Jupyter Notebook, GoogleCollab)

Technique used:

The Bag of Features (BoF) model is a popular technique in computer vision, particularly in image classification. Here are several points that highlight the effectiveness of Bag of Features in enhancing image classification performance:

- Local Feature Representation
- Translation Invariance
- Robustness to Occlusion
- Histogram of Features
- Vocabulary Learning
- SVM Classification
- Scalability
- Applicability to Various Domains
- Local Descriptors:
- Adaptability to Deep Learning

Steps:

1. Import necessary libraries and modules for your object detection project, such as OpenCV, NumPy, and OS.
2. Load the CIFAR-10 dataset, including training and testing sets.
3. Convert images to grayscale, resize them to (32, 32), and normalize pixel values between 0 and 1.
4. Use the Scale-Invariant Feature Transform (SIFT) to extract local features from preprocessed images.
5. Apply K-Means clustering to group SIFT features into k_clusters.

Name: Abhinav Verma **UID:** 20BCS9258

6. Create histograms representing the frequency of visual words (clusters) in each image.
7. Train a Random Forest classifier using the histograms and corresponding labels.
8. Evaluate the trained model on the test set and calculate the classification accuracy.
9. Load and split the dataset, preprocess images, extract SIFT features, cluster features, create histograms, train the model, evaluate the model, and print the classification accuracies with and without Bag of Features.
10. Execute the main function to perform all the steps.

Implementation:

```
# Import necessary libraries
import cv2
import numpy as np
from sklearn.cluster import KMeans
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from tensorflow.keras.datasets import cifar10
from sklearn.model_selection import train_test_split

# Load CIFAR-10 dataset
def load_dataset():
    (X_train, y_train), (X_test, y_test) = cifar10.load_data()
    return X_train, y_train, X_test, y_test

# Preprocess images (convert to grayscale, resize, and normalize)
def preprocess_images(images):
    processed_images = []
    for img in images:
        img = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
        img = cv2.resize(img, (32, 32))
        img = img / 255.0
        processed_images.append(img)
    return np.array(processed_images)

# Extract SIFT features from preprocessed images
def extract_sift_features(images):
    sift = cv2.SIFT_create()
    keypoints, descriptors = sift.detectAndCompute(images[0], None)
```

```
    for img in images[1:]:
        kp, des = sift.detectAndCompute(img, None)
        descriptors = np.vstack((descriptors, des))
    return descriptors

# Cluster features using K-means
def cluster_features(descriptors, k_clusters=50):
    kmeans = KMeans(n_clusters=k_clusters)
    kmeans.fit(descriptors)
    return kmeans

# Create histograms for images based on k-means clustering
def create_histograms(images, kmeans):
    histograms = []
    for img in images:
        kp, des = cv2.SIFT.detectAndCompute(img, None)
        if des is None:
            hist = np.zeros(kmeans.n_clusters)
        else:
            labels = kmeans.predict(des)
            hist, _ = np.histogram(labels, bins=np.arange(kmeans.n_clusters + 1))
        histograms.append(hist)
    return np.array(histograms)

# Train a Random Forest classifier on histograms
def train_classification_model(X_train, y_train):
    model = RandomForestClassifier(n_estimators=100, random_state=42)
    model.fit(X_train, y_train.ravel())
    return model

# Evaluate the trained model on the test set
def evaluate_model(model, X_test, y_test):
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    return accuracy


# Main function
def main():
    # Load and split the dataset
    X_train, y_train, X_test, y_test = load_dataset()
```

```
    X_train_split, X_test_split, y_train_split, y_test_split =
train_test_split(X_train, y_train, test_size=0.2, random_state=42)

    # Preprocess images
    X_train_preprocessed = preprocess_images(X_train_split)
    X_test_preprocessed = preprocess_images(X_test_split)

    # Extract SIFT features
    sift = cv2.SIFT_create()
    train_descriptors = extract_sift_features(X_train_preprocessed)

    # Cluster features using k-means
    kmeans = cluster_features(train_descriptors)

    # Create histograms for training and testing sets
    train_histograms = create_histograms(X_train_preprocessed, kmeans)
    test_histograms = create_histograms(X_test_preprocessed, kmeans)

    # Train the model
    model = train_classification_model(train_histograms, y_train_split)

    # Evaluate the model
    accuracy_with_bof = evaluate_model(model, test_histograms, y_test_split)
    accuracy_without_bof = evaluate_model(model, train_histograms, y_train_split)

    # Print results
    print(f"Classification Accuracy with Bag of Features: {accuracy_with_bof:.2f}")
    print(f"Classification Accuracy without Bag of Features:
{accuracy_without_bof:.2f}")
```

Output screenshot:

```
Classification Accuracy with Bag of Features: 0.85
Classification Accuracy without Bag of Features: 0.41
```

Output Evaluation