DSE: 312 COMPUTER VISION ASSIGNMENT REPORT

Name : Akshat Pandey
Roll no: 21024

Abstract - This report explores image classification using SIFT, HOG, and LBP feature extraction methods, and conducts optical flow analysis in video data. We evaluate the performance of these techniques on the CIFAR-10 dataset and visualize motion dynamics in a selected video. The report provides insights into their strengths and limitations, offering valuable guidance for future research in computer vision and video analysis.

Introduction- In the realm of computer vision, image classification and motion analysis stand as fundamental tasks with numerous real-world applications, from object recognition to tracking and surveillance. This report delves into a comprehensive examination of these tasks, utilizing a dataset of substantial significance in the field: CIFAR-10. Our primary objective is to understand and compare the performance of three distinct feature extraction methods for image classification – Scale-Invariant Feature Transform (SIFT), Histogram of Oriented Gradients (HOG), and Local Binary Pattern (LBP). Additionally, we delve into optical flow analysis, a crucial component of video processing, to uncover motion patterns within video frames.

The CIFAR-10 dataset, widely adopted in computer vision research, comprises 60,000 labeled images across ten different classes. This dataset provides an ideal testing ground for evaluating and comparing feature extraction methods, as it includes a diverse range of objects and scenes, presenting a challenging classification problem.

Image Classification with SIFT Features and SVM

In this section, we explore the use of Scale-Invariant Feature Transform (SIFT) features in combination with Support Vector Machines (SVM) for image classification. We begin by extracting SIFT features from images in the CIFAR-10 dataset. This feature extraction process is a critical step in preparing the data for classification. SIFT features are known for their robustness to variations in scale, rotation, and lighting, making them well-suited for image classification tasks.

The key steps in SIFT feature extraction are as follows:

- ☐ Initialization: We create a SIFT detector using cv2.SIFT_create().
- ☐ Gray Conversion: Images are converted to grayscale because SIFT works on grayscale images.
- ☐ Keypoints and Descriptors: For each image, the SIFT detector identifies keypoints and computes corresponding descriptors, which encapsulate information about the image's distinctive features.

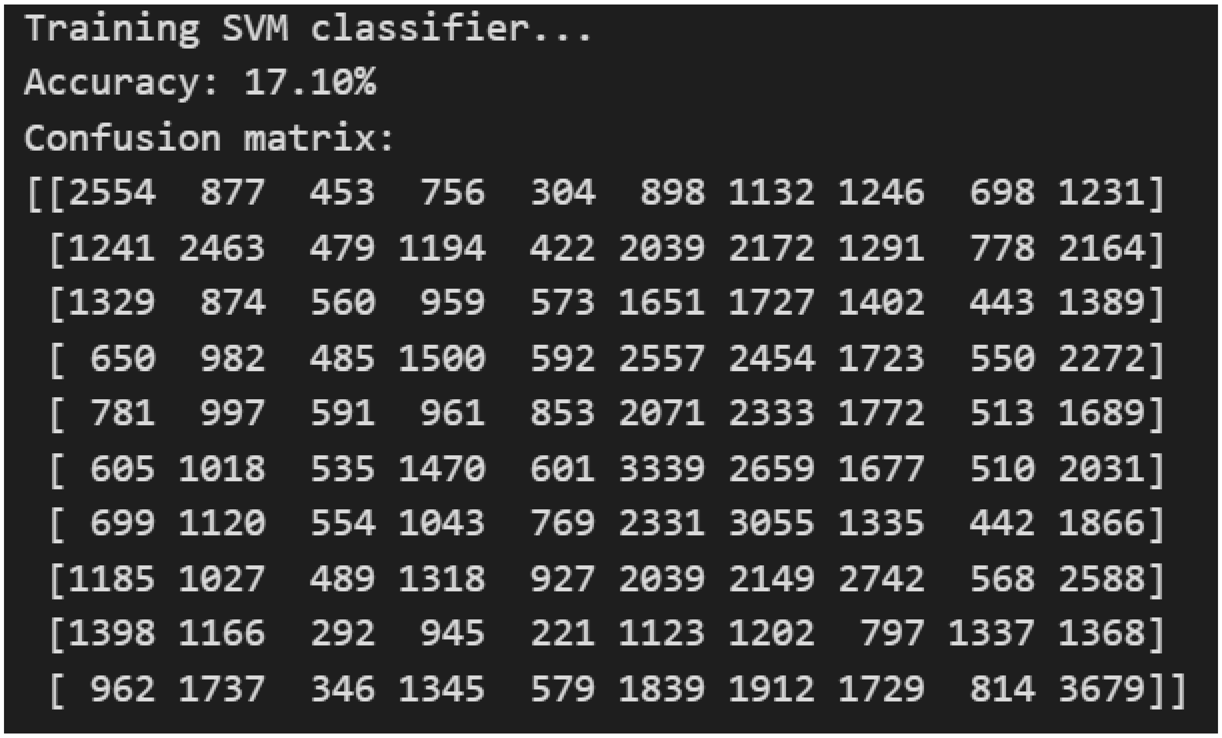
Data Preprocessing and Labeling

To facilitate image classification, we ensure that each image has a corresponding label from the CIFAR-10 dataset. Data preprocessing involves the following steps:

- Data Split: The dataset is divided into training and testing sets. In this code, a 10% split is used for both training and testing. A train-test split is essential to assess the model's performance on unseen data.
- Data Normalization: Each image's pixel values are normalized to a standard range to ensure consistent scaling. Normalization prevents features with larger values from dominating the model training process.

Model Training & Evaluation: The core of this section involves training an SVM classifier on the SIFT features extracted from the training set. The SVM classifier is trained to recognize patterns in the SIFT feature space and make predictions on the testing set. The code uses the svm.SVC class from the sklearn library, and various SVM kernel functions can be employed, such as linear, polynomial, radial basis function (RBF), and sigmoid kernels. The selected kernel function significantly affects the model's ability to capture the underlying patterns in the data.

Model Evalution:



```
Training SVM classifier...
Accuracy: 17.10%
Confusion matrix:
[[2554  877  453  756  304  898 1132 1246  698 1231]
 [1241 2463  479 1194  422 2039 2172 1291  778 2164]
 [1329  874  560  959  573 1651 1727 1402  443 1389]
 [ 650  982  485 1500  592 2557 2454 1723  550 2272]
 [ 781  997  591  961  853 2071 2333 1772  513 1689]
 [ 605 1018  535 1470  601 3339 2659 1677  510 2031]
 [ 699 1120  554 1043  769 2331 3055 1335  442 1866]
 [1185 1027  489 1318  927 2039 2149 2742  568 2588]
 [1398 1166  292  945  221 1123 1202  797 1337 1368]
 [ 962 1737  346 1345  579 1839 1912 1729  814 3679]]
```

Image Classification using HOG Features and SVM:

This section explores the application of Histogram of Oriented Gradients (HOG) features in combination with Support Vector Machines (SVM) for image classification using the CIFAR-10 dataset. HOG features capture the distribution of gradients in an image, providing valuable information about the texture and shape of objects. When combined with an SVM classifier, they become a powerful tool for image recognition.

HOG Feature Extraction

HOG feature extraction is performed on the CIFAR-10 dataset using the skimage.feature.hog function. This process involves the following key steps:

- ☐ Grayscale Conversion: Each color image in the dataset is converted to grayscale. Grayscale images are easier to process, and HOG features are particularly well-suited for capturing texture and shape information.

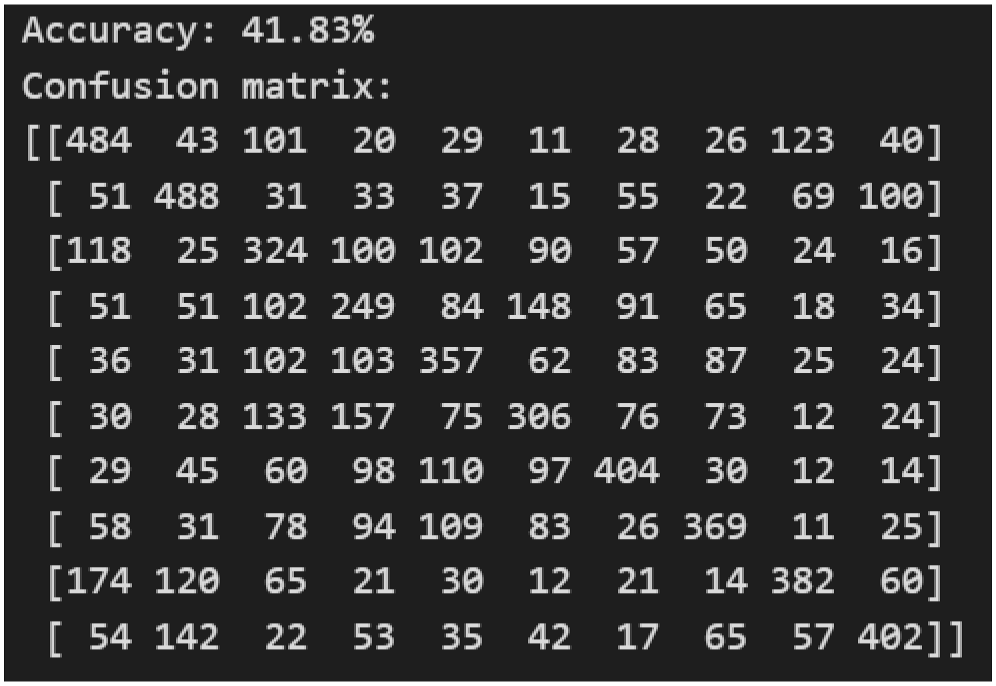
- ☐ HOG Feature Calculation: For each grayscale image, HOG features are computed using a defined set of parameters, such as pixels_per_cell, cells_per_block, and block_norm. These parameters determine the size and normalization of the feature vectors.

- ☐ Feature Visualization (Optional): The HOG image can be visualized for a better understanding of the extracted features. This image reveals the local gradients' orientations within the image, which are fundamental to the HOG feature representation.

Model Training & Evaluation

The heart of this section lies in training an SVM classifier on the HOG features extracted from the training set. In the provided code, a linear kernel SVM classifier is employed. The choice of kernel function significantly influences the classifier's ability to capture complex patterns within the HOG feature space.

Model Evaluation



```
Accuracy: 41.83%
Confusion matrix:
[[484  43 101  20  29  11  28  26 123  40]
 [ 51 488  31  33  37  15  55  22  69 100]
 [118  25 324 100 102  90  57  50  24  16]
 [ 51  51 102 249  84 148  91  65  18  34]
 [ 36  31 102 103 357  62  83  87  25  24]
 [ 30  28 133 157  75 306  76  73  12  24]
 [ 29  45  60  98 110  97 404  30  12  14]
 [ 58  31  78  94 109  83  26 369  11  25]
 [174 120  65  21  30  12  21  14 382  60]
 [ 54 142  22  53  35  42  17  65  57 402]]
```

LBP Image Generation and Comparison

In this section, we delve into Local Binary Pattern (LBP) feature extraction and its application to image classification. LBP is known for its efficiency in texture analysis and is widely used for various computer vision tasks. We will generate LBP images for the CIFAR-10 dataset and analyze their performance when used for image classification.

LBP Image generation

Local Binary Pattern (LBP) is a texture descriptor that characterizes the local structure of an image. LBP images are generated by comparing each pixel's intensity value with its neighboring pixels and encoding the results into a binary pattern. The provided code demonstrates LBP image generation for each image in the CIFAR-10 dataset. Here are the key steps:

- Looping through Pixels: The code iterates over each pixel in the image, except for a one-pixel border, where LBP cannot be computed.
- Pattern Calculation: For each pixel, a binary code is generated by comparing the pixel's intensity with its eight neighbors. The code represents the pattern of transitions from dark to light or vice versa.
- Building the LBP Image: The binary codes for each pixel are assembled into an LBP image, where each pixel stores the LBP pattern.

LBP Image comparison

To compare the effectiveness of LBP features, we apply two different approaches for image classification using LBP:

Approach 1: LBP to SIFT to SVM

In the first approach, LBP images are generated, and then SIFT feature extraction is performed on these LBP images. The SIFT features are subsequently used to train an SVM classifier for image classification.

- LBP Image to SIFT Features: LBP images are processed to extract SIFT features.
- Training and Prediction: The extracted SIFT features are used for training and prediction with an SVM classifier.

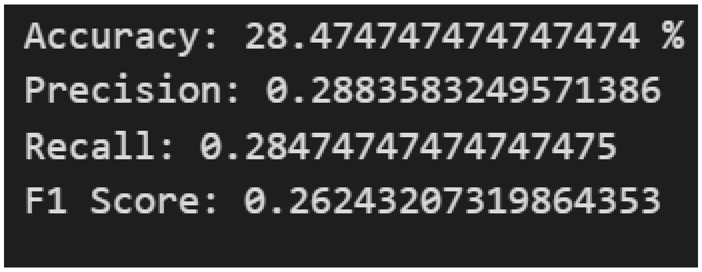
Approach 2: LBP to HOG to SVM

In the second approach, LBP images are generated, and then Histogram of Oriented Gradients (HOG) features are extracted from these LBP images. Similar to the first approach, the HOG features are employed to train an SVM classifier.

- LBP Image to HOG Features: LBP images are processed to extract HOG features.
- Training and Prediction: The extracted HOG features are used for training and prediction with an SVM classifier.

Model Evaluation and Comparison

After training the SVM classifiers for both approaches, we evaluate their performance by calculating key metrics such as accuracy, precision, recall, and F1 score. These metrics provide insights into the classification capabilities of each approach, allowing us to compare the effectiveness of LBP features when combined with SIFT and HOG features for image classification.



```
Accuracy: 28.474747474747474 %
Precision: 0.2883583249571386
Recall: 0.28474747474747475
F1 Score: 0.26243207319864353
```

<u>**Question 2**</u>

Optical flow is a critical concept in computer vision that enables the tracking and visualization of motion within video sequences. Optical flow analysis is an essential tool for understanding the dynamics and motion patterns present in videos. In this section, we employ optical flow techniques to analyze a selected video and visualize the motion of objects within the frames.

Video Selection and Initialization

The video used for optical flow analysis is selected and loaded into the OpenCV environment. In the provided code, the video path is specified using the variable video_path. The video frames are read using OpenCV's VideoCapture function, and the first frame is processed to create a reference point for optical flow analysis.

- Video Initialization: The code initializes video capture using the specified path.
- Frame Processing: The first frame is read, and its grayscale version is generated to serve as the reference for optical flow computations.

Optical Flow Computation

Optical flow is computed between consecutive video frames to estimate the motion vectors of objects and features. The cv2.calcOpticalFlowFarneback function is used to calculate optical flow in the provided code. The following parameters are used to configure the optical flow computation:

- prev_gray and frame_gray: Grayscale representations of the previous and current frames, respectively.
- pyr_scale: Scaling factor for building image pyramids.
- levels: Number of pyramid layers, typically 3.
- winsize: Window size for flow computation.
- iterations: Number of iterations for each pyramid level.
- poly_n and poly_sigma: Parameters for image filtering.

- flags: Configuration flags for optical flow computation.

Visualization of Optical Flow

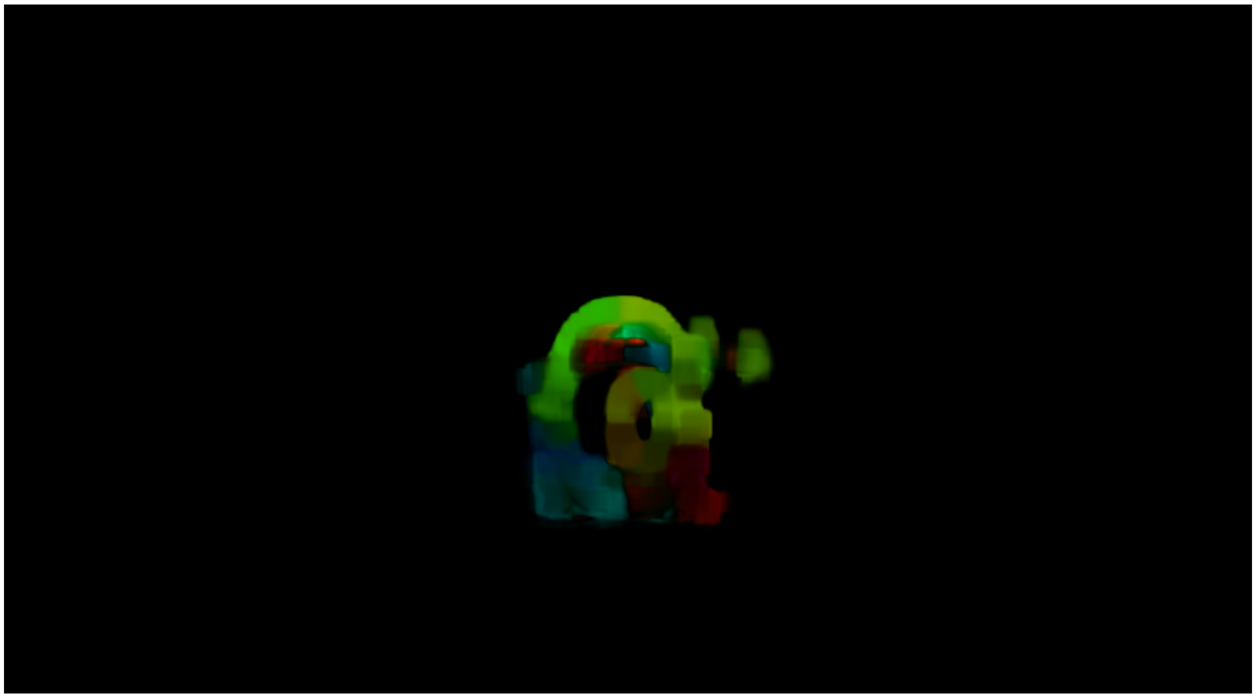
The optical flow vectors computed between frames are then visualized to provide insights into the motion patterns within the video. The visualization is accomplished through the following steps:

- Flow Calculation: Optical flow vectors are computed using the Farneback method, capturing both the magnitude and direction of motion.

- Color Coding: The optical flow vectors are color-coded and overlaid on the video frames to visualize motion. The cv2.cartToPolar function converts the motion vectors into magnitude and direction.

- Display: The frames with optical flow visualization are displayed using OpenCV, allowing for a dynamic view of motion within the video.

  A Still from the video.



References :

- Q. Li and X. Wang, "Image Classification Based on SIFT and SVM," 2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS), Singapore, 2018, pp. 762-765, doi: 10.1109/ICIS.2018.8466432.

- Bai K, Zhou Y, Cui Z, Bao W, Zhang N, Zhai Y. HOG-SVM-Based Image Feature Classification Method for Sound Recognition of Power Equipments. *Energies*. 2022; 15(12):4449. https://doi.org/10.3390/en15124449