Assignment 1

Instructions:

1. **Submission Requirements:**
    - Use the provided template and upload the Jupyter notebook in HTML or PDF format with outputs and its corresponding PDF.
    - You may also choose to use Google Colab for working on the project.
    - Data files do not need to be uploaded with the submission.
    - Ensure the updated Jupyter notebook is submitted with properly formatted and aligned outputs. Incomplete outputs, misalignments, or poorly written comments will result in a deduction of marks.
    - Partial code and partial output will be evaluated, and marks will be awarded based on the PDF file.
    - Only the latest submission would be considered for marking.
2. **File Naming Convention:**
    - Name the file as: CV_assignment_1_group##
3. **Tools and Libraries:**
    - Use OpenCV, Numpy, and SKlearn/Xgboost for constructing the model and image processing.
    - Do not use CNN/RNN for this assignment.
    - The assignment focuses on image preprocessing and image classification.
4. **Plagiarism Policy:**
    - Any form of plagiarism will be taken very seriously, resulting in zero (0) marks.
    - All submissions must be the result of your original effort.
    - Copying from any sources, whether online or from peers, is strictly prohibited.
    - Unauthorized collaboration to gain an unfair advantage is prohibited.
    - Both the person sharing resources and the one receiving them will face consequences for plagiarism.
    - Identical or significantly similar submissions will be investigated, and severe punishments will be imposed on those found guilty.
5. **Late Submission Policy:**
    - Late submissions will incur a penalty of -2 marks.
6. **Queries:**

- For any questions regarding the assignment, use the discussion forum.

7. **Assignment Template:**
   - The assignment template is available at the following link: https://drive.google.com/file/d/1i60z8cvFxFSQxv6cy4szpYvJEU82P9Ed/view?usp=sharing
   - Links to an external site.
   -

8. Programming Instructions:

- Use appropriate Python programming techniques to read and process the dataset.
- Convert the notebook into HTML or PDF format along with outputs and upload the file.
- Avoid submitting excessively long notebooks by eliminating irrelevant lengthy prints.

9. Problem Statement Selection:

- Choose**ANY ONE** problem statement from the three provided below.
- Enter the problem statement number in the following sheet (Do not edit others' responses): https://docs.google.com/spreadsheets/d/1iGOvt04uEwmFgWyWkQuh8S9Dx_eNViDd/edit?usp=sharing&ouid=117836832163432495869&rtpof=true&sd=true
- Links to an external site.
-

==========

Goals:

- Experiment with additional features like texture analysis using Gabor filters. (**Optional**)
- Visualize feature importance if using models like Random Forest or SVM. Hint : Show a bar plot of Model Accuracies and metrics for different Feature Engg techniques . Like for instance : HOG,LBP,Edge Det , HOG+ LBP , LBP+Edge det , HOG+LBP+Edge det
- Emphasize handcrafted feature engineering and their role in solving computer vision problems. This assignment is not about achieving perfect accuracy but

understanding how handcrafted features contribute to scene classification. Let your analysis and insights shine!

==========

Problem Statement 1: "Image Scene Classification"

Dataset Link :
https://drive.google.com/file/d/18ivVD85YKQqPH0Qhe2Ou10hjuPl-vWxA/view?usp=sharing

Links to an external site.

You have to choose 4-5 features to extract from the image dataset. You must provide your intution why a choice of feature helps the problem at hand. You are expected to study the problem on your own and identify various features to solve the scene classification problem.

The outline of problems is as below

Scene Classification:

[write about dataset, categories, training and test set, evaluation metrics]

How do download the dataset?

Google drive link contains the dataset

The data has images such that:

Folder

Class_1:

image_1

image_2

Class_2:

image_1

image_2

Tasks :

1- Select a dataset of images depicting various scenes (e.g., mountain, airport, desert, forest, river)

2- Extract 3-4 features (e.g., Local Binary Patterns, histogram equalization etc) from each image

Low-level Vision: Histogram and Histogram equalization, Gray-scale transformation, Image Smoothing in images.

Mid-level Vision: Edge Detection using Gradients, Sobel, Canny; Line detection using Hough transforms; Semantic information using RANSAC; Image region descriptor using SIFT

3- Create a structured data of multiple sets of features with corresponding class labels and store it in a datafile (e.g., CSV or Excel) so that you can later use it for training and comparing the models.

4- Train a classical machine learning model (e.g., SVM, Random Forest, XgBoost, etc) using the extracted features.

5- Evaluate the model performance using the metrics:

- Accuracy
- Precision
- Recall
- F1-score
- Mean Average Precision (mAP): The average precision across all queries.
- Perform evaluations by creating a set of query images and comparing the results with ground truth labels.
- Perform the above evaluations with and without applying the features extracted in the preprocessing stage

6- Analyze the results and discuss the effectiveness of features for aerial view classification.

7- Discuss potential limitations and future improvements of the approach.

==========

Problem Statement 2: "Edge-based Image Retrieval"

Objective: Implement an image retrieval system that uses edge-based features to find similar images in a dataset. Evaluate the retrieval performance using relevant metrics.

Tasks:

1. **Import the required libraries**
    - Import the necessary libraries for image processing (e.g., OpenCV, scikit-image) and machine learning (e.g., scikit-learn, pandas).

2. Data Acquisition

- Download the Caltech 101 dataset from the provided link: Caltech 101 Dataset
- Links to an external site.
-
- Select any 5 categories from the dataset. Each category should have at least 50 images.
- Organize the selected categories into separate folders, with each folder containing the corresponding images.
- Analyze and plot the distribution of images per category using a bar graph or pie chart.

3. Data Preparation

- Randomly split the dataset into training and testing sets using an 80% training and 20% testing ratio. Ensure that the split is stratified, maintaining the same proportion of images per category in both sets.

4. **Preprocess** the images as follows:

- Convert the images to grayscale.
- Apply the Canny edge detection algorithm to extract edges from the images.

- Extract edge-based descriptors such as edge histograms or gradient histograms from the preprocessed images.
- Create multiple feature sets by varying the parameters of the edge detection and descriptor extraction methods.
- Store the extracted features in separate dataframes, along with the corresponding category labels.
- Normalize the feature values using techniques like min-max scaling or z-score normalization.
- If the feature dimensionality is too high, apply dimensionality reduction techniques such as PCA or resize the images to a smaller fixed size.

5. Model Building

- Select a classical machine learning algorithm such as Support Vector Machines (SVM), Random Forest, or XGBoost for training the image retrieval model.
- Train the chosen model on different feature combinations created in the preprocessing step.
- Use appropriate hyperparameter tuning techniques (e.g., grid search, random search) and cross-validation (e.g., k-fold) to optimize the model's performance.

6. Validation Metrics

- Evaluate the trained models using the following metrics:
    - Accuracy
    - Precision
    - Recall
    - F1 score
    - Mean Average Precision (mAP)
- Calculate the mAP by ranking the retrieved images based on their similarity scores and computing the average precision at different recall levels.
- Aim for a minimum mAP of 0.7 to consider the retrieval system as acceptable.

7. Model Inference

- Randomly select 5 test images from each category and use the best-performing model to retrieve the top-5 most similar images for each query image.

- Display the query image along with its top-5 retrieved images, their predicted labels, and the actual labels.
- Justify your choice of features based on the retrieval performance and visual analysis of the results.

8. **Analysis and Discussion**

- Analyze the retrieval results and discuss the effectiveness of the edge-based features for image retrieval.
- Compare the performance of different feature combinations and highlight the ones that yield the best results.
- Identify any limitations or challenges encountered during the implementation and suggest potential improvements or alternative approaches.

==========

==========

Problem Statement 3: "Exploring Features in Urban and Natural Scenes"

The goal of this assignment is to analyze and classify urban and natural scenes using handcrafted features extracted through low-level and mid-level vision techniques.

Dataset:

Download the dataset from
https://www.kaggle.com/datasets/dansbecker/urban-and-rural-photos

Links to an external site.

. It contains two categories:

1. Urban Scenes (e.g., buildings, streets, traffic)
2. Rural Scenes (e.g., forests, rivers, mountains, rural houses)

Tasks:

1. **Dataset Overview**:
    - Provide a concise description of the dataset.

    - List the categories and the number of images in each category.
    - Outline the preprocessing steps.
2. **Data Preprocessing**:
    - Resize all images to a standard size (e.g., 128x128).
    - Apply grayscale conversion and histogram equalization.
    - Explore other preprocessing techniques like Gaussian smoothing and contrast enhancement.
3. **Feature Engineering**:
    - Extract meaningful features such as:
        - Histogram of Oriented Gradients (HOG)
        - Local Binary Patterns (LBP)
        - Edge Detection using Sobel/Canny
    - Discuss how these features help in distinguishing between urban and natural scenes.
4. **Feature Selection**:
    - Compile extracted features into a structured dataset (e.g., .csv or .xlsx).
    - Normalize or standardize the feature values if necessary.
5. **Model Training**:
    - Train two classical machine learning models (e.g., SVM, Random Forest) on the extracted features.
    - Split the data into training and test sets with an 80-20 split.
6. **Model Evaluation**:
    - Evaluate the models using metrics:
        - Accuracy
        - Precision
        - Recall
        - F1-score
    - Compare model performances and visualize the results using bar charts or confusion matrices.
7. **Analysis and Discussion**:
    - Analyze which features were most impactful for classification.
    - Discuss limitations of the approach and suggest potential improvements.