Assignment 1: Image Processing using OpenCV.

Objective: Understand the basics of image handling in OpenCV, including loading, displaying,

and performing basic transformations such as color space conversion, resizing, cropping,

rotation, and flipping.

Tasks:

1. Load and Display an Image:

o Use OpenCV to read and display an image.

o Check the image’s dimensions and number of color channels.

o Understand how image data is stored and represented.

2. Convert Image Colour Spaces:

o Convert the image from RGB to Grayscale.

o Convert the image from RGB to HSV.

o Understand the importance of Grayscale and HSV color models in computer

vision applications.

3. Resize the Image:

o Use cv2.resize() to change the size of the image (both enlarging and

reducing).

4. Crop the Image:

o Use array slicing to extract a region of interest (ROI) from the image.

5. Rotate the Image:

o Use cv2.rotate() to rotate the image by 90, 180, and 270 degrees.

6. Flip the Image:

o Use cv2.flip() to flip the image horizontally, vertically, and both.

Assignment 2: Image Thresholding, Blurring, and Sharpening using OpenCV.

Objective: Learn and apply image preprocessing techniques such as thresholding, blurring, and

sharpening to enhance or segment specific regions of an image.

Tasks:

1. Thresholding Techniques:

o Apply Simple Thresholding using cv2.threshold() to segment an image based

on fixed pixel intensity values.

o Apply Adaptive Thresholding using cv2.adaptiveThreshold() for images with

varying lighting conditions.

2. Blurring Techniques:

o Use Gaussian Blur with cv2.GaussianBlur() to reduce image noise and detail.

o Apply Median Blur using cv2.medianBlur() for removing salt-and-pepper noise

while preserving edges.

3. Image Sharpening:

o Use cv2.filter2D() with a custom kernel to sharpen the image by enhancing

edges and details.

Assignment 3: Batch Image Preprocessing on Google Colab

Objective: Learn to automate the preprocessing of multiple images using Python and OpenCV

in a Google Colab environment. Get familiar with mounting Google Drive, uploading datasets,

and executing preprocessing tasks in the cloud.

Tasks:

1. Google Colab Setup:

o Mount Google Drive in Colab.

o Upload and access image datasets from Drive.

o Understand the structure of notebooks in a cloud-based environment.

2. Batch Image Preprocessing:

o Use the os module to iterate through a folder containing multiple images.

o For each image, apply the following preprocessing steps using OpenCV:

▪ Resize to a fixed size (e.g., 256×256).

▪ Convert to Grayscale.

▪ Normalize pixel values to the range [0, 1].

3. Save and Verify Outputs:

o Save the processed images to a new folder in your Drive.

o Optionally, visualize a few outputs using matplotlib or cv2.imshow() (within

Colab constraints).

Assignment 4: Build a Simple ANN on MNIST

Explanation: Create a dense neural network using TensorFlow (Sequential API/

Functional API) / PyTorch for digit classification.

Assignment 5: Create a CNN for CIFAR-10

Explanation: Implement Conv2D, MaxPooling2D, and Flatten layers to build an

image classifier.

Assignment 6: Visualize CNN Feature Maps

Explanation: Extract intermediate layers and display their activations to see what the

model "sees".

Assignment 7: Compare Activation Functions

Explanation: Experiment with ReLU, Sigmoid, and LeakyReLU and track model

performance.

Assignment 8: Try Different Loss Functions

Explanation: Use categorical cross-entropy, MSE, and Binary cross-entropy to

observe their influence on training.

Assignment 9: Optimizer Comparison (SGD vs Adam)

Explanation: Compare model convergence speed and stability using different

optimizers.

Assignment 10: Build an RNN on Pixel Sequences and Visualize Training Performance.

Objective: Implement a basic Recurrent Neural Network (RNN) or LSTM model on the MNIST dataset

by treating each image as a sequence of pixel rows. Track and visualize the model's training and

validation performance over epochs.

Tasks:

1. Prepare the Data:

o Load the MNIST dataset using TensorFlow/Keras.

o Reshape each 28×28 image into a sequence of 28 time steps with 28 features (rows as

time steps).

2. Build and Train the RNN Model:

o Use SimpleRNN or LSTM layers to model the temporal pixel sequence.

o Add Dense layers for classification.

o Compile, train, and evaluate the model on the test data.

3. Visualize Training Progress:

o Use matplotlib to plot training and validation accuracy and loss over epochs.

o Interpret the graphs to identify overfitting, underfitting, or stable training behaviour.

Assignment 11: Load and Preprocess CIFAR-10 with Augmentation.

Assignment 12: CNN with Dropout and Batch Normalization.

Assignment 13: Load pre-trained VGG16 for Feature Extraction.

Assignment 14: Fine-Tune ResNet50 for Custom Dataset.

Assignment 15: Replace Classifier Head in InceptionV3.

Assignment 16: Save and Reload Trained Model and Make Predictions on New Images.

Assignment 17: Visualize Predictions with Confidence Scores.

Assignment 18: Use DeepLabV3 for Semantic Segmentation.

Assignment 19: Visualize Saliency Maps.

Assignment 20: Build an Autoencoder on MNIST.

Assignment 21: Visualize Autoencoder Reconstructions.

Assignment 22: Variational Autoencoder Implementation.

Assignment 23: GAN for MNIST Digit Generation.

Assignment 24: Track GAN Loss Over Epochs.

Assignment 25: Generate Fashion Images using DCGAN.

Assignment 26: Evaluate GANs with IS/FID Metrics.

Assignment 27: Explore Latent Space Interpolation in GAN.