**Image Difference Detection with Bounding Boxes and Similarity Score**

**1. Introduction**

The objective of this project is to develop a computer vision model capable of identifying differences between two images, drawing bounding boxes around varying regions, and calculating a similarity score. The comparison is implemented using both traditional computer vision techniques and deep learning models.

**2. Methodology**

**2.1 Traditional Computer Vision Approach**

The traditional computer vision approach involves the following steps:

* Read and resize two input images.
* Convert the images to grayscale.
* Compute the absolute difference between the grayscale images.
* Apply thresholding to create a binary difference mask.
* Utilize morphological operations (dilation) to enhance the differences.
* Identify contours and draw bounding boxes around significant differences.

**2.2 Deep Learning Approach with scikit-image**

The deep learning approach uses the structural similarity index (SSIM) for image comparison:

* Read and resize two input images.
* Convert the images to grayscale.
* Calculate the SSIM between the two images.
* Create a difference mask based on the SSIM values.
* Apply thresholding to obtain a binary difference mask.
* Identify contours and draw bounding boxes around significant differences.
* Display the SSIM value on one of the images.

**3. Implementation**

**3.1 Traditional Computer Vision Implementation (OpenCV)**

import cv2

import imutils

import numpy as np

from skimage.metrics import structural\_similarity as compare\_ssim

# Traditional Computer Vision Approach (OpenCV)

def traditional\_image\_difference\_detection(img1\_path, img2\_path):

img1 = cv2.imread(img1\_path)

img1 = imutils.resize(img1, width=600)

img2 = cv2.imread(img2\_path)

img2 = imutils.resize(img2, width=600)

gray1 = cv2.cvtColor(img1, cv2.COLOR\_BGR2GRAY)

gray2 = cv2.cvtColor(img2, cv2.COLOR\_BGR2GRAY)

diff = cv2.absdiff(gray1, gray2)

thresh = cv2.threshold(diff, 0, 255, cv2.THRESH\_BINARY | cv2.THRESH\_OTSU)[1]

kernel = np.ones((5, 5), np.uint8)

dilate = cv2.dilate(thresh, kernel, iterations=2)

contours = cv2.findContours(dilate.copy(), cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

contours = imutils.grab\_contours(contours)

for contour in contours:

if cv2.contourArea(contour) > 100:

x, y, w, h = cv2.boundingRect(contour)

cv2.rectangle(img1, (x, y), (x + w, y + h), (0, 0, 255), 2)

cv2.rectangle(img2, (x, y), (x + w, y + h), (0, 0, 255), 2)

x = np.zeros((img1.shape[0], 10, 3), np.uint8)

result = np.hstack((img1, x, img2))

cv2.imshow("Traditional Image Difference Detection", result)

cv2.waitKey(0)

cv2.destroyAllWindows()

traditional\_image\_difference\_detection("input3.jpg", "input4.jpg")

**3.2 Deep Learning Implementation with scikit-image**

import cv2

import imutils

import numpy as np

from skimage.metrics import structural\_similarity as compare\_ssim

def deep\_learning\_image\_difference\_detection(img1\_path, img2\_path):

img1 = cv2.imread(img1\_path)

img1 = imutils.resize(img1, width=600)

img2 = cv2.imread(img2\_path)

img2 = imutils.resize(img2, width=600)

gray1 = cv2.cvtColor(img1, cv2.COLOR\_BGR2GRAY)

gray2 = cv2.cvtColor(img2, cv2.COLOR\_BGR2GRAY)

(similar, diff) = compare\_ssim(gray1, gray2, full=True)

diff = (diff \* 255).astype("uint8")

thresh = cv2.threshold(diff, 0, 255, cv2.THRESH\_BINARY\_INV | cv2.THRESH\_OTSU)[1]

contours = cv2.findContours(thresh.copy(), cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

contours = imutils.grab\_contours(contours)

for contour in contours:

if cv2.contourArea(contour) > 100:

x, y, w, h = cv2.boundingRect(contour)

cv2.rectangle(img1, (x, y), (x + w, y + h), (0, 0, 255), 2)

cv2.rectangle(img2, (x, y), (x + w, y + h), (0, 0, 255), 2)

cv2.putText(img2, "Similarity: {:.2f}".format(similar), (10, 30),

cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (0, 0, 255), 2)

x = np.zeros((img1.shape[0], 10, 3), np.uint8)

result = np.hstack((img1, x, img2))

cv2.imshow("Deep Learning Image Difference Detection", result)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Example usage

deep\_learning\_image\_difference\_detection("input3.jpg", "input4.jpg")

**4. Results**

* Compare the accuracy and efficiency of both approaches.
* Visualize the results by displaying images with drawn bounding boxes and the calculated similarity score.

**5. Conclusion**

In conclusion, the project successfully implements image difference detection using a combination of traditional computer vision and deep learning techniques. The ability to draw bounding boxes and calculate a similarity score enhances the interpretability and usability of the model.