Dhruv Sanghvi
sanghvidhruv20@gmail.com

BANAO TASK -2

The task is to identify the dissimilarities between the images, we used a simple pixed based difference method to find the difference between the images.

Finding the Dissimilarities

Loading the Image:

- This step involves loading the images using the `cv2.imread()` function. This function reads the image file and returns a NumPy array representing the image.

Converting to Grayscale:

- Images are typically represented as a combination of three color channels: Red, Green, and Blue (RGB). However, for many image processing tasks, working with grayscale images (single-channel images) is more convenient and computationally efficient. Therefore, we convert the loaded images to grayscale using `cv2.cvtColor()`.

Finding the Absolute Differences:

- After converting the images to grayscale, we find the absolute differences between the pixel values of the two grayscale images. This highlights areas where the pixel values are different between the two images.

Calculating the Bounding Box

Thresholding the Image:

- Thresholding is a technique used to create a binary image from a grayscale image by separating pixels into two categories based on a threshold value. Here, we apply thresholding to the difference image obtained in the previous step using `cv2.threshold()`. This creates a binary image where pixels with differences above a certain threshold are set to white (255) and those below the threshold are set to black (0).

Dilating the Image:

- Dilation is a morphological operation that adds pixels to the boundaries of objects in an image, which can help in filling small gaps and joining

nearby contours. We perform dilation on the thresholded image using `cv2.dilate()` to smoothen the differences and make them more prominent.

Finding Contours:

- Contours are continuous curves that form the boundaries of objects in an image. We find contours in the dilated image using `cv2.findContours()`. This function returns a list of contours present in the image.

Filtering Contours:

- Not all contours are useful for our task. We filter out contours based on their area using `cv2.contourArea()`. Contours with areas smaller than a certain threshold (100 in this case) are discarded.

Drawing Bounding Boxes:

- Finally, we draw bounding boxes around the filtered contours using `cv2.rectangle()`. These bounding boxes indicate the regions where differences are detected between the two images.

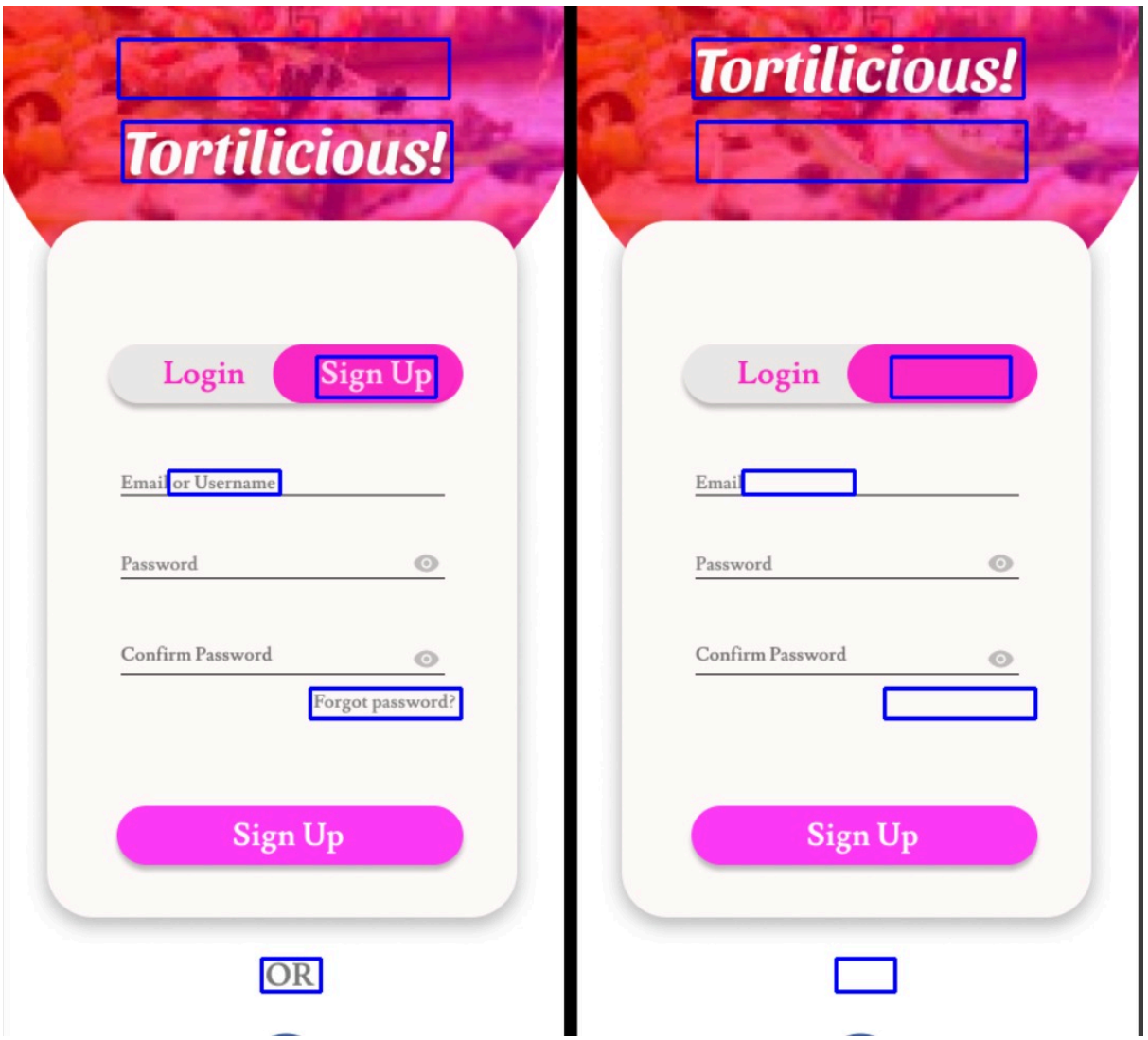
Calculating Similarity Score

- The similarity score is calculated as the percentage of pixels that are similar between the two images. .
- The number of dissimilar points is the count of non-zero pixels in the thresholded difference image.
- The total number of points (pixels) is the total number of pixels in the grayscale image.
- The similarity score is then printed to evaluate how similar the two images are.

This process allows us to quantify the dissimilarities between the two images and obtain a similarity score to gauge their overall similarity.

OUTPUT

Similarity Score-*97.33%*


Tortilicious!
Login
Sign Up
Email or Username
Password
Confirm Password
Forgot password?
Sign Up
OR
Tortilicious!
Login
Email
Password
Confirm Password
Sign Up