Computer Vision

CSE483

SVHN Bounding box & Digit Recognition Project

Supervised by: Dr Mahmoud Khalil

TA Mahmoud Soheil

Made by:

1- Abdelrahman Hamdy Mohammed 16P8224

2- Mohamed Ahmed Ebrahim 17P6089

3- Khaled Abdelmoniem Amr 14P8145

Introduction

This code consists of two main parts. The first part calculates Intersection over Union (IoU) between two bounding boxes, and the second part is an OCR (Optical Character Recognition) system that recognizes digits in an image.

Implementation

In the first part, the function bb_intersection_over_union takes two bounding boxes as input and calculates their IoU. The IoU is a measure of the overlap between two bounding boxes, and it is defined as the intersection of the two boxes divided by the union of the two boxes. The function first calculates the intersection rectangle by finding the coordinates of the top-left and bottom-right corners of the intersection. Then, it calculates the area of the intersection rectangle, as well as the areas of both the prediction and ground-truth rectangles. Finally, it computes the IoU by dividing the intersection area by the sum of the prediction and ground-truth areas minus the intersection area. The IoU value is returned as the output of the function.

In the second part, the code performs OCR on an image to recognize a digit. The image is first resized to a target size of 500x500 pixels. Then, it goes through a series of preprocessing steps to prepare it for OCR. First, the image is scaled down by a factor of 0.5 using cv2.resize. Then, it is blurred using a Gaussian filter with a kernel size of 5x5 pixels. Next, the image is converted from BGR color space to grayscale using cv2.cvtColor. After that, a thresholding operation is applied to the image using cv2.adaptiveThreshold to convert it to a binary image. The thresholding operation uses the cv2.ADAPTIVE_THRESH_MEAN_C method, which calculates the threshold value for each pixel based on the mean of the surrounding pixels. Finally, the code finds the contours in the binary image using cv2.findContours, and it selects the contour with the largest area using cv2.contourArea.

Once the largest contour is selected, the code calculates its bounding box using cv2.boundingRect. The bounding box is then scaled back up to

the original image size by multiplying its coordinates by a factor of 2. The code then draws a green rectangle around the bounding box using cv2.rectangle. The IoU between the predicted bounding box and a ground-truth box is calculated using the bb_intersection_over_union function defined earlier.

After the bounding box is drawn, the code extracts the digit from the image by cropping the region of interest (ROI) using the coordinates of the bounding box. The ROI is then resized to a fixed size of 5x5 pixels using cv2.resize. The code then matches the digit to a template using a dictionary of digit templates. The template that has the lowest sum of absolute differences (SAD) with the digit is selected as the best match. The digit that corresponds to the template is then printed to the console.

Finally, the image is resized back up to the target size using cv2.resize, and it is displayed with the bounding box and the predicted digit using cv2.imshow. The user can close the image window by pressing any key.

Overall, the code performs a simple OCR task by recognizing a single digit in an image and calculating its bounding box and IoU with a ground-truth box.