Domain Background

The project that I am going to work on is in the field of object detection. Object detection has become very popular in the last few years. Object detection is a challenging problem since it includes both localization of the object and its classification. Many algorithms have been proposed starting with Viola-Jones Framework which was the first object detection framework and was used primarily for face detection. Another traditional approach was Histogram of Oriented Gradients combined with Support Vector Machines. Other solutions use deep learning techniques such as R-CNNs and YOLO.

Object detection can be applied to many important problems such as face recognition and counting objects in the image. Particularly, it could be used for the security measures like in Passenger Screening challenge on Kaggle. I am personally interested in this field because nowadays there is a big variety of important problems that can be solved using computer vision from healthcare to entertainment industry.

References

Histograms of Oriented Gradients for Object Detection by N. Dalal and B. Triggs:

https://courses.engr.illinois.edu/ece420/fa2017/hog_for_human_detection.pdf

Tutorial on HOG and Object Detection:

https://www.pyimagesearch.com/2014/11/10/histogram-oriented-gradients-object-detection/

Tutorial on RCNNs:

https://medium.com/ilenze-com/object-detection-using-deep-learning-for-advanced-users-part-1-183bbbb08b19

Problem Statement

In this problem I'm going to apply object detection techniques in order to solve a famous game "Where's Waldo". Sometimes it becomes very challenging for a human to find Waldo so I decided to implement an algorithm for this task. The input to the algorithm is waldo images and not waldo images. Then the model can be trained to distinguish the images that contain Waldo using a binary classifier. In order to locate Waldo in the original image, the algorithm can scan the whole original image using a sliding window and output the probability of having Waldo in the image segment. The correctness of the problem can be measured by the percent of times Waldo is found correctly.

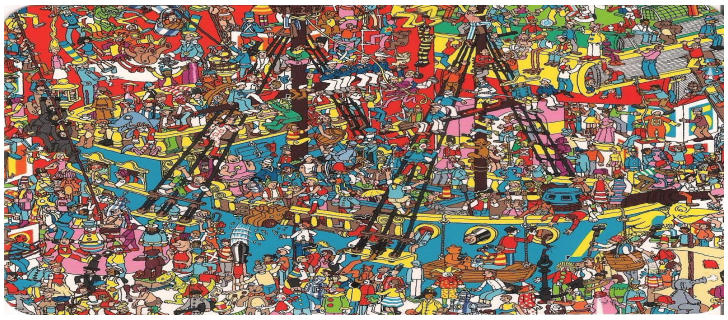
Datasets and Inputs

The data set can be found at the following link (further information about getting the data is provided in README file): https://www.kaggle.com/residentmario/wheres-waldo
The following dataset contains colored, black and white, and grey images of size 64x64, 128x128, and 256x256. Every image type contains two folders “waldo” and “not waldo”. Waldo images are cut off from the original images. Not Waldo images contain other people from the original images. Using this data, one can train the model to recognize waldo images.
Here’s more information about the number of images in the data set and their sizes.

- 256 x 256 pixels (317 images)
- 128 x 128 pixels (1344 images)
- 64 x 64 pixels (5376 images)

The original images can be found in ‘original images’ data set.

Waldo:  Not Waldo: 

Original Image: 

Solution Statement

As it was stated in the problem statement, one could train a model to define images that contain Waldo using a binary classifier such as an SVM, Neural Network, or Convolutional Neural Network (where the last layer is a neural network). In order to find the exact location of Waldo in the original image, a sliding window technique can be applied. Then one must apply the trained classifier to each window and calculate the probability that it contains Waldo. If the probability is high enough (higher than a chosen threshold), record the bounding box of the window.

Benchmark Model

One could create a histogram of pixels for each image and feed them into a simple Neural Network. Then a sliding window could be used to locate Waldo in the following way: create a histogram of pixels for each sliding window and output the probability for each window of having Waldo in it. Then compare this value to a threshold and if it is bigger than a threshold, output 1 (Waldo is present in the window) or 0 (Waldo is not present in the window).

Evaluation Metrics

Using cross validation technique the dataset can be divided into training, validation, and testing data sets. Then the evaluation should be run on the test dataset since it makes the evaluation less bias.

Since there are much more negatives (segments that do not contain Waldo) in the original images, it is more efficient to focus on positives (segments that contain Waldo). A good solution could be using a precision metric (TP/(TP + FP)), recall (TP/(TP + FN)), or f1 score which is a combination of recall and precision (2 * (Recall * Precision) / (Recall + Precision)). I decided to use f1 score because it is a more balanced metric and it focuses on segments that have Waldo.

Project Design

First I'm going to train the model on Waldo and not Waldo images then apply the obtained results to the original images to find Waldo.

Before applying any machine learning algorithms to images one has to preprocess the images and present them in a format that can be fed to a classifier. All the images in the Waldo dataset are already preprocessed: the images have equal sizes and there is an option to choose between colored and black and white images. In my problem, it is more efficient to use colored images since one of the most important features about Waldo is red and white stripes so knowing the rgb value of pixels is very helpful. As for image representation, I'm going to use Histogram of Oriented Gradients (HOG), one of the traditional ways to represent pixels in the images. The idea behind HOG is the following: create the histograms of gradients that show the direction in which the image gets darker. Using this approach one can find the main features of the image since the histogram shows the edges that shape the objects. One might have a question, why cannot one use histograms of pixels? The answer is a different lighting of images.

After the images are converted into histograms, we can feed them into a classifier. Both waldo and not waldo images should be passed to a classifier (in this case it works like labeling data rows with 0s and 1s in supervised learning). Empirically SVMs work the best with HOG, however, Neural Networks could also be used for training the model. It is always better to compare two models by using some evaluation metric such as accuracy on a testing data set and pick the one with better results.

After the classifier is trained we need to find Waldo's location in the original image. I'm going to use a sliding window of a fixed size and scan the whole image. While scanning, the algorithm should convert each window into a HOG and return the probability of having Waldo in each window using the trained classifier. Then record the bounding box of the most probable regions. For better results, it is helpful to apply sliding windows of different sizes to the original image and record the areas with a high probability of having Waldo.
To summarize the algorithm:

1) For each waldo and not waldo image calculate HOG.
2) Feed the obtained histograms to a chosen classifier.
3) Using a sliding window technique, scan the original image and output the most probable area that has Waldo.