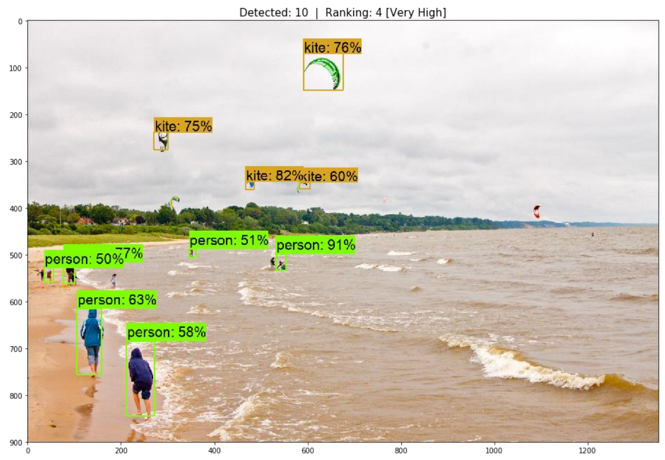
fixIT

Litter Detection Algorithm: Implementation of Project

Table of Contents

1. Object Detection

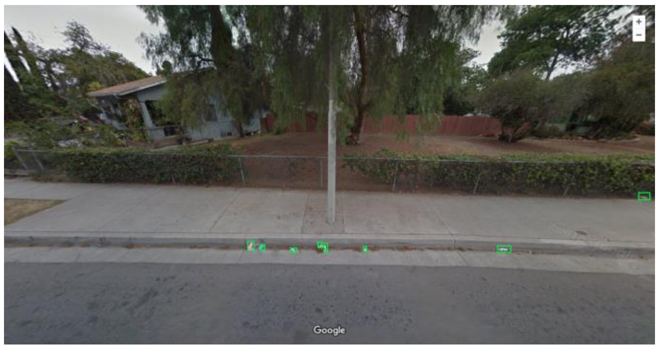


FixIT chose to use the TensorFlow API to create the algorithm, Jupyter Notebook to interface the algorithm, and an Amazon Web Services Ubuntu server to train the algorithm. Using TensorFlow's pre-trained model and its associated COCO dataset, the team was able to have a foundation for the object detection portion of the project's development.

The resulting algorithm of Prototype 1 boxed relatively every object that it detected within an input image, as exemplified in the provided image. Each detected object is surrounded by a box. The percentage indicated with each detected object is a confidence indicator detailing the accuracy of that specific detection. For example, the algorithm is 76% confident that it detected a kite in the picture provided. There is also a ranking system that was implemented, although unstandardized. For Prototype 1, images were ranked from 1-4 based on the amount objected detected within a given image. This feature provided the foundation for the ranking of images based on its litter severity implemented later in Prototype 2. The final output of an image for the first prototype boxed the objects with the accuracy percentage and a ranking of the image.

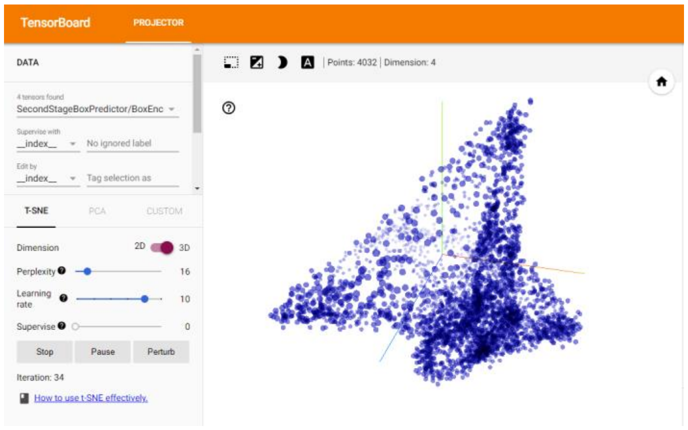
2. Dataset

After hours of research, fixIT found that a large dataset specific to litter was necessary for the algorithm's litter recognition functionality. The dataset that the team had ultimately compiled consisted of images that were pulled from Google Streets. Manually pulling images from Google Streets proved to be a long process, so fixIT created a script that would gather thousands of images automatically. Once the images were acquired, the team needed to label them in order to commence the training process. Thus, FixIT pulled 10,000 images from Google Streets to be used for the dataset.



Keep America Beautiful provided a team of students who would label the images. This team used the labelImg software provided by fixIT for this purpose. The students initially separated the images using the following categories: images with litter and images without litter. They then labeled all of the images with litter. Once the students were done with labeling, fixIT needed to verify the validity of these images' labels. As a result of the verification stage, the team found that some of the labeled images were not labeled well, and thus were of no use. At the end of this process, it was found that about 3,000 images were usable for the dataset. Although not all images provided were labeled, the 3,000 images that were proved to be especially useful due to the fact that in total, 9,000 pieces of litter were labeled.

3. Training



The goal for Prototype 2 was to implement the litter recognition portion of the algorithm. In order to implement the recognition, however, the team would first need to train the labeled dataset according to its needs. The team first split the dataset into two sets; 80% of images were used for training and the rest was used for testing. FixIT trained the algorithm to certain amount of “steps”, producing a checkpoint file used to test the algorithm’s litter recognition accuracy. These checkpoint files indicated to the team when to stop the training and test the algorithm to verify the algorithm’s learning progress. If the team deemed the algorithm to be overtrained, the previously saved checkpoint file would be used.

4. Litter Detection

Once the training was done, fixIT tested the algorithm with a variety of images. The images that did not contain litter in the dataset were also used here. The team needed to make sure the algorithm would not box other objects in images even if there was no litter in the picture. Eventually after hours of testing and training, the team was satisfied with the result.

The final output has the algorithm only box pieces of litter with over a 50% confidence, along with a ranking for the image.

fixIT

Detected: 11 | Ranking: 4 [Very High]