Machine Learning Engineer Nanodegree Capstone Project

Google AI Open Images - Object Detection Track

Matthew Lee
7/22/2018

Content

Domain Background

In recent years, one of the hottest applications of AI in computer vision has been object detection. Object detection has wide variety of applications such as robotics, autonomous driving, facial recognition, security, and etc. Such advancements are transforming our society and helping us to achieve higher goals with unprecedented speed and accuracy.

Despite some of the hypes around AI's unlimited potentials, I believe current AI algorithms and solutions are very specific to individual applications. However, I believe object detection algorithm is an example in which it can be used across multiple applications.

This capstone project wishes to expand object detection domain by developing an algorithm to successfully detect objects in given image dataset.

Problem Statement

Very popular method to develop an object detection algorithm is achieved by training neural networks with labeled images. Carefully chosen neural network architecture will be able to surpass human's performance, but training process requires enormous amount of data. We will utilize plethora of images with labels from Open Images Dataset, publicly released by Google AI, to develop an object detection algorithm in order to push the limits of object detection capability. Once trained, we will be able to monitor trained algorithm's performance by evaluating computed mean Average Precision (AP).

Datasets and Inputs

In this project, Open Images Dataset, publicly released by Google AI, will be used for training the algorithm. Dataset contains 1.7 million images that contain very diverse and multiple objects. Across images in the dataset, there are 12 million bounding-box annotations for 500 object classes, making it the largest existing dataset with object location annotations. Bounding-box annotations are drawn by professional annotators to ensure accuracy and consistency, making a very optimal dataset for training an object detection algorithm.

Training and Validation Set -> Download #1 or Download #2
Testing Set -> Download

Solution Statement

Convolutional Neural Network (CNN) will be a perfect fit for this problem. In order to solve problems such as object detection, capturing spatial information is very crucial and CNN is a great fit. CNN's filters are able to capture spatial features (lines, curves, squares...) and are able to detect objects using learned filters.

Benchmark Model

In the domain of object detection, there are many proven neural network architectures such as VGGNet, GoogLeNet, Resnet, DenseNet, and etc. There are widely available weights, pre-trained on large dataset similar to Open Images Dataset, from aforementioned neural network architectures. Using pre-trained weights from aforementioned architectures can be used to predict objects, and score from the base model will be used as a benchmark.

Evaluation Metrics

Benchmark model and the solution's performances will be evaluated by mean Average Precision (AP) (mAP) across 500 classes in 1.7 million images in the dataset.

Precision (P) will be calculated by dividing true positives (T_p) over the true positives plus false positives (F_p) (Precision-Recall)

$$P = \frac{T_p}{T_p + F_p}$$

Detections are considered true or false positives based on the area of overlap with ground truth bounding boxes. To be considered a correct detection, the area of overlap a_o

between the predicted bounding box B_p and ground truth bounding box B_{gt} must exceed 50% by the formula (PASCAL VOC 2010):

$$a_o > 0.5 \text{ where } a_o = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})}$$

Recall (R) is calculated by dividing true positives (T_p) over the true positives plus the false negatives (F_n) (Precision-Recall)

$$R = \frac{T_p}{T_p + F_n}$$

Then, average precision (AP) will be calculated for each of 500 classes.

$$AP = \sum_{k=1}^{n} (R_k - R_{k-1}) P_k$$

Finally, final mean average precision (mAP) will be calculated by taking average of all APs over the 500 classes.

$$mAP = \frac{\sum_{k=1}^{500} AP_k}{500}$$

Note that, unlike previous standard challenges such as PASCAL VOC 2010, Google AI Open Images - Object Detection Track has three new metrics that affect the way True Positives and False Positives are accounted (Open Images Challenge 2018 - object detection track - evaluation metric)

- Due to the Open Images annotation process, image-level labeling is not exhaustive.
- The object classes are organized in a semantic hierarchy, meaning that some categories are more general than others (e.g. 'Animal' is more general than 'Cat', as 'Cat' is a subclass of 'Animal').
- Some of the ground-truth bounding-boxes capture a group of objects, rather than a single object.

Project Design

1. Benchmark Model

    - My first approach is to utilize pre-trained weights from most current Convolutional Neural Network (CNN) architectures such as VGG-19, ResNet-50, Inception, Xception, and etc. We will initially obtain benchmark model by evaluating which architecture's pre-trained weights give best mean average precision right off the bet.
2. Improve Base Model
    - Data will be analyzed if there are major imbalances. If so, data augmentation techniques such as ADASYN, SMOTE, scaling, and etc can be used.
    - Split dataset to training (~80%) and validation set (~20%) to be used to identify the most effective model. (For this competition, training and validation sets are provided)
    - Then, model will be tweaked and improved to yield the highest mean Average Precision (*mAP*). Some of important categories for optimization potentials are learning rate, types of optimizers, usage of model ensembling, and so on.
3. Final Touch
    - Visualize file results in order to identify some of the best and worst performing classes. This will help us to find room for improvement in the model. One popular way is done by using a confusion matrix.
    - Finally, train the model using entire dataset (training and validation set combined). Then, obtain predictions for testing image dataset and submit it in Kaggle.