Object Detection in an Image

Bruno de Almeida Silveira

October 2019

1 Domain Background

In 2016, Google released its first dataset with more than one object annotated by image [1], OpenImagesV1, which has around 1GB of images. This dataset was created and annotated by Google Inc under the CC BY 4.0 license. In May of 2019, Google released a new version of this dataset [2], the fifth version, with a new Kaggle competition [3], which was the inspiration for this project.

The problem of Object Detection in an image has some research on it, such as [4], [5], [6], and [7]. A proper Object Detection model is the start of any image model recognition problem. So, this project intends to face it considering fundamental research for anyone who wants to work with images.

2 Problem Statement

The main challenge of this project is to create a model that identify and classifies all objects in a image.

The project intends to classify objects from 500 different classes correctly, and it is measured based on how many classes previously labeled it correctly predicts.

Like any classifier, it can be modeled as a Logistic Regression, some Support Vector Classifier (SVC) with some kernel, a Neural Network, among many other possible models. To this scope, Neural Networks are more applicable as a solution, even more, Deep Learning models.

This problem has many practical instances, like identifying objects in security footage; labeling a massive amount of photos automatically; automatically create a catalog for a retail company; and so many others.

3 Datasets and Inputs

The dataset used in this project is the same dataset provided in the Kaggle competition created by Google, the following link [8] provides in the section *Object Detection track annotations* the url to download. Google created the dataset labeling manually around 15 million of bounding boxes among 1.7 million images. Dataset is composed of a zip file with all images, a CSV that contains the position of each bounding box in each image and its class identifier, another CSV that maps each class identifier with a semantic class name, and a JSON file with the full hierarchical classes defined.

Google also divided the whole dataset using some stratified strategy to preserve the proportions of each class. More details could be found in the [9], the following table describes the division proposed.

Table 1: Dataset division provided by Google

	Train	Validation	Test	Classes
Images	1,743,042	41,620	125,436	-
Boxes	14,610,229	303,980	937,327	600

4 Solution Statement

To solve the problem, this project is going to use deep learning models, and it is going to use some transfer learning techniques in the state of the art of the Convolutional Neural Networks (CNNs) already designed [10], such as ResNet-N layers [11], Inception-V4 [12], Xception [13] and Inception-ResNets [12]. The project is going to compare them with a couple of models tailored for this project, and see how all of them perform.

5 Benchmark Model

The benchmark models for this project are going to be the models created for Google's Kaggle competition [3]. So, the Leaderboard itself is going to be used as a benchmark [14]. The top five achieve scores are 0.65887, 0.65337, 0.64214. 0.62221, and 0.60406.

6 Evaluation Metrics

The metric proposed by Google in the competition is the mean Average Precision (mAP) [15], a very didactic explanation about the metric could be found in here [16] and [17].

The mAP metric could be defined as

$$mAP = \frac{\sum_{c=1}^{C} AP_c}{C}$$

where C value is the number of all categories (classes).

To understand AP_c, it must comprehend first what is IoU. IoU is the Intersection over Union, and it is equal to the ratio of the area of intersection and area of the union of the predicted bounding box (created by the model) and Ground-truth bounding box (previously annotated).

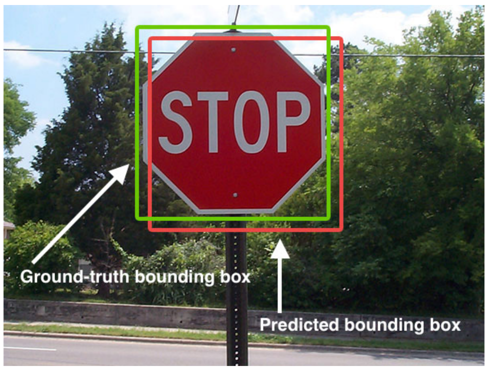

Figure 1: Difference between a predict bounding box with a Ground-truth bouding box [18]

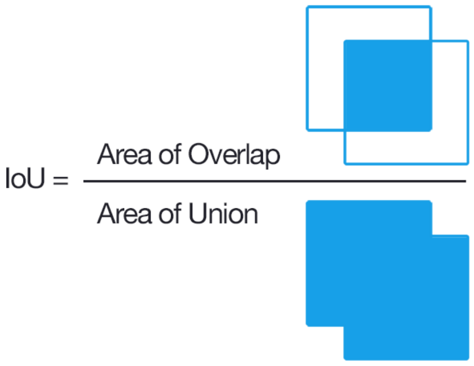

Figure 2: IoU visual represented [18]

Here, it is going to define:

$$\begin{aligned} TruePositive &\doteq IoU > 0.5 \\ FalsePositive &\doteq IoU < 0.5 \\ &\quad or\ DuplicatedPredictBoundingBox \\ FalseNegative &\doteq IoU > 0.5 \\ &\quad and\ WrongClassification \end{aligned}$$

With TP, TN, and FP defined, it is possible to create a Precision-Recall Curve, which defines a function that gives a precision based on the recall.

So by definition, AP_c (Average Precision of some category c), is defined as an Area Under the Curve (AUC) of the Precision-Recall curve.

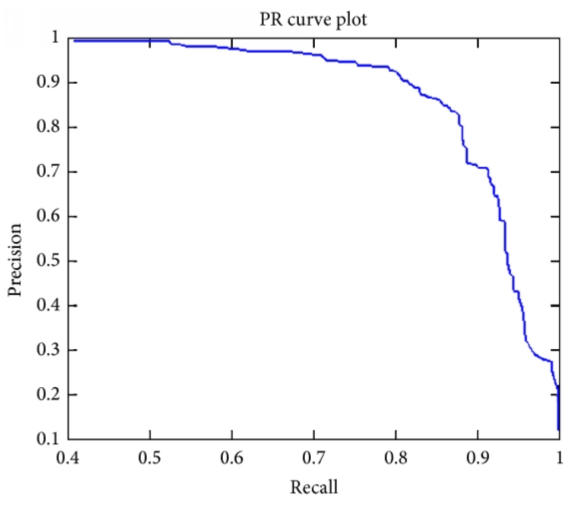

Figure 3: Precision-Recall curve [16]

$$AP_c = \int_0^1 p(r)dr$$

where $p(r)$ is the precision defined in function of recall.

It is essential to retain that in this project it is going to be used AP_{50} (which uses a threshold of 0.5 in IoU to define TP), but other metrics could be evaluated, like, AP_{75} (with IoU threshold of 0.75) or AP_{90} (with IoU threshold of 0.90).

7 Project Design

This project is going to divide the problem itself into two, the problem to create the correct bounding box and the problem to classify it. The idea is to create a simple pipeline with two models concatenated, and each model is going to optimize each technique. git s The first model, called Bounding Box Identifier, is responsible for identifying all objects that could be classified by the Image Classifier Model. The loss function of this model is going to be the IoU (described in the previous section).

On the other side, for the Image Classifier model, it is going to use the Categorial Crossentropy as a loss function.

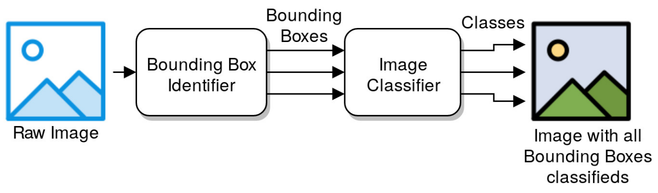

Figure 4: High level Model Pipeline

It is import to explaining more about the loss functions.

The Categorical Crossentropy used in the Classifier model is straightforward since it is a multiclass classifier. The idea is to compare for each class each distribution of the predictions made with the labeled distribution.

The IoU metric used as loss function arrives because mAP derives directly from the IoU. Since the mAP is going to be used as a metric to compare all models, it sounds reasonable to use IoU as a loss function to be optimized in the model responsible for creating the Bounding Boxes.

In the first step, the project starts to using Transfer Learning technique in many famous networks, for both models, Object Identification, and Classifier. With Transfer Learning, we expect to achieve some reasonable results.

For the second and final step of the project, this work proposes to create, at minimum, one new architecture that could face other architectures seen above. It is not defined yet if this new architecture is going to be for Object Identification problem, for the Classifier problem, or both. Yet, it is sounds challenge and aspirational.

References

[1] Google AI Blog. Introducing the open images dataset. https://ai.googleblog.com/2016/09/introducing-open-images-dataset.html, 2016.

[2] Google. Announcing open images v5 and the iccv 2019 open images challenge. https://storage.googleapis.com/openimages/web/2019-05-08-announcing-v5-and-challenge-2019.html, may 2019.

[3] Google Research. Open images 2019 - object detection. https://www.kaggle.com/c/open-images-2019-object-detection/overview, jun 2019.

[4] Philipp Krähenbühl Xingyi Zhou, Jiacheng Zhuo. Bottom-up object detection by grouping extreme and center points. Jan 2019. https://arxiv.org/abs/1901.08043v3 and https://paperswithcode.com/paper/bottom-up-object-detection-by-grouping.

[5] Haifang Qin Jianping Shi Jiaya Jia Shu Liu, Lu Qi. Path aggregation network for instance segmentation. Sep 2018. https://arxiv.org/abs/1803.01534v4 and https://paperswithcode.com/paper/path-aggregation-network-for-instance.

[6] Tianheng Cheng Borui Jiang Chaorui Deng Yang Zhao Dong Liu Yadong Mu Mingkui Tan Xinggang Wang Wenyu Liu Bin Xiao Jingdong Wang, Ke Sun. Deep high-resolution representation learning for visual recognition. Aug 2019. https://arxiv.org/abs/1908.07919v1 and https://paperswithcode.com/paper/190807919.

[7] Siwei Wang TingTing Liang Qijie Zhao Zhi Tang Haibin Ling Yudong Liu, Yongtao Wang. Cbnet: A novel composite backbone network architecture for object detection. Sep 2019. https://arxiv.org/abs/1901.08043v3 and https://paperswithcode.com/paper/cbnet-a-novel-composite-backbone-network.

[8] Google Research. Open images 2019 - dataset download - object detection track annotations section. https://storage.googleapis.com/openimages/web/challenge2019_downloads.html.

[9] Google Research. Overview of open images v5. https://storage.googleapis.com/openimages/web/factsfigures.html.

[10] Raimi Karim. Illustrated: 10 cnn architectures. https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d, jul 2019.

[11] Shaoqing Ren Jian Sun Kaiming He, Xiangyu Zhang. Deep residual learning for image recognition. Dec 2015. https://arxiv.org/abs/1512.03385.

[12] Vincent Vanhoucke Alex Alemi Christian Szegedy, Sergey Ioffe. Inception-v4, inception-resnet and the impact of residual connections on learning. Feb 2016. https://arxiv.org/abs/1602.07261.

[13] François Chollet. Xception: Deep learning with depthwise separable convolutions. Oct 2016. https://arxiv.org/abs/1610.02357.

[14] Leaderboard. Leaderboard - open images 2019 - object detection. https://www.kaggle.com/c/open-images-2019-object-detection/leaderboard.

[15] wikipedia. Mean average precision. https://en.wikipedia.org/wiki/Evaluation_measures_(information_retrieval)#Mean_average_precision.

[16] Jonathan Hui. map (mean average precision) for object detection. https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173.

[17] Ren Jie Tan. Breaking down mean average precision (map). https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52#d439.

[18] Adrian Rosebrock. Intersection over union (iou) for object detection. https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/.