Recent Advanced in Machine Learning

Computer vision: object detection(githup)

Directed by : Adjoua HOUNDONOUGBO & Morel Mbedi

Master 2 MALIA, Lyon 2 University

Introduction

This project is part of the "Advanced method in Machine Learning" course that we took during the Master 2 course at the University Lumière Lyon 2. The aim is to detect objects in images using pre-trained models specialised in this task.

In practice, the pre-trained model should be tested with the test data used, in order to qualitatively and quantitatively evaluate the results obtained in this exploratory case. Then, the pre-trained model must be tested with images collected on the internet. Finally, we will do the fine tuning to see if the results obtained in this case are better than in the two previous cases.

In order to perform all these tasks, we chose to do the detection of the 4 objects, namely the detection of dog, cat, apples and oranges.

Description of the architecture of the neural network model used

In the computer vision problem, there are several pre-trained models to do object detection on images. In this project, we will use the fasterrcnn_resnet50_fpn model based on the pytorch library. It has been pre-trained on a number of object detection data, including COCO data which includes 91 different object classes or types. The Faster R-CNN model with ResNet50 is an enhanced version of Faster R-CNN that uses residual block resilience and feature pyramiding to improve the accuracy of object detection.

How the architecture works

Faster R-CNN is a two-stage object detection architecture. The first step is to propose regions of interest (RoI) in the image that could contain objects. The second step consists of classifying the proposed RoIs and predicting their bounding box to locate the object. Specifically, the faster-rcnn-resnet50-fpn-coco-torch model uses a convolutional neural network architecture

called ResNet-50 as a feature extractor for the proposed RoIs. It also uses an activation function called "ReLU" (Rectified Linear Unit) to introduce non-linearity into the model for the convulsion layers and softmax has been used for the output layer which allows the prediction of probabilities of several object classes simultaneously

The different parameters of the model :

- Weights: the pre-trained model weights to be set to True (weigth=True) to use these weights.
- Num_classes: number of classes to detect, we will take 4 classes
- **weights_backbone:** the CNN's pre-trained weights that extract features from the image. This model is the ResNet50.
- **trainable_backbone_layers :** specify the number of trainable layers by default it's 3

To implement this model, we used the fiftyone library which includes several pre-trained models and several methods for evaluating object detection problems.

Description of the "fiftyone" bookshop

Fiftyone is a rich library of methods for solving computer vision tasks. It allows you to easily load, modify, visualise and evaluate your data and all associated labels (classifications, detections, etc.). They provide an interactive python-based interface for loading images, videos, annotations and model predictions in a format that can be viewed in the FiftyOne application.

Methods of assessing fiftyone

FiftyOne provides a variety of built-in methods for evaluating your model predictions, including regressions, classifications, detections, polygons, instance and semantic segmentations, on image and video datasets. When evaluating a model in FiftyOne, you have access to standard aggregate metrics such as classification ratios, confusion matrices and PR curves for your model. In addition, FiftyOne can also record precise statistics such as accuracy and number of false positives at the sample level, which you can exploit via its interactive interface to interactively explore the strengths and weaknesses of your models on individual data samples.

Sample-level analysis often leads to critical information that will help you improve your datasets and models. For example, displaying the samples with the most false positive predictions may reveal errors in your annotation scheme. Or, displaying the group of samples

with the lowest accuracy may reveal gaps in your training dataset that you need to fill in order to improve the performance of your model. One of FiftyOne's main goals is to help you discover this information about your data!

Metrics for evaluating object detections

We have defined the metrics needed to evaluate our object detection.

ious: This is a dictionary that contains the IoUs (Intersection over Union) corresponding to the detection and ground truth matches for each image. The IoUs represent the overlap between a detection box and the associated ground truth.

The IoU (Intersection over Union) metric represents the ratio of the intersection between the prediction and the ground truth, divided by the union between these two. An IoU of 1.0 indicates that the prediction matches the ground truth perfectly, while an IoU of 0 indicates that there is no match between the two.

mAP: This is a float that represents the average of the mean average precision (mAP) for all classes. The Mean Average Precision (mAP) metric is a commonly used measure for evaluating the accuracy of object detection models. It is calculated by averaging the accuracies at different recall levels for different confidence threshold values. A higher mAP score indicates better detection accuracy.

precision: measures the proportion of true positives (VP) that the model was able to identify among all the true positives (VP + false negatives). Recall = VP / (VP + false negatives)

recall: measures the proportion of true positives (VP) among all examples classified as positive (VP + false positives). Accuracy = VP / (VP + false positives)

F-score (or F1-score): is a measure that combines precision and recall into one value. It is defined as the harmonic mean of precision and recall:

F-score = 2 * (precision * recall) / (precision + recall)

The F-score allows both precision and recall to be considered in a single measure, which is useful when one wishes to evaluate a model by considering both false positives and false negatives.

The confusion matrix: this is the cross between the ground truth and what has been predicted by the model. It allows us to see how the model is mistaken in classifying an object into another.

All these metrics will be obtained through the fiftyone library

Exploration of the pre-trained model on the images of the coconut dataset

First, let's take a look at the coco dataset with the fiftyone library, as it offers several powerful features and graphical interfaces designed specifically to allow us to explore, analyse and work with your data as easily as possible.

The code below loads the coco data with the fiftyone library of 4 classes chosen by taking a sample of 200 images of the cat, dog, apple and orange labels

```
import fiftyone as fo
import fiftyone.zoo as foz

dataset = foz.load_zoo_dataset("coco-2017",
                               split="validation",
                               classes=["cat", "dog","apple","orange"],
                               max_samples=200,
                               dataset_dir ="/content/datasetcoco",
                               shuffle=True)
```

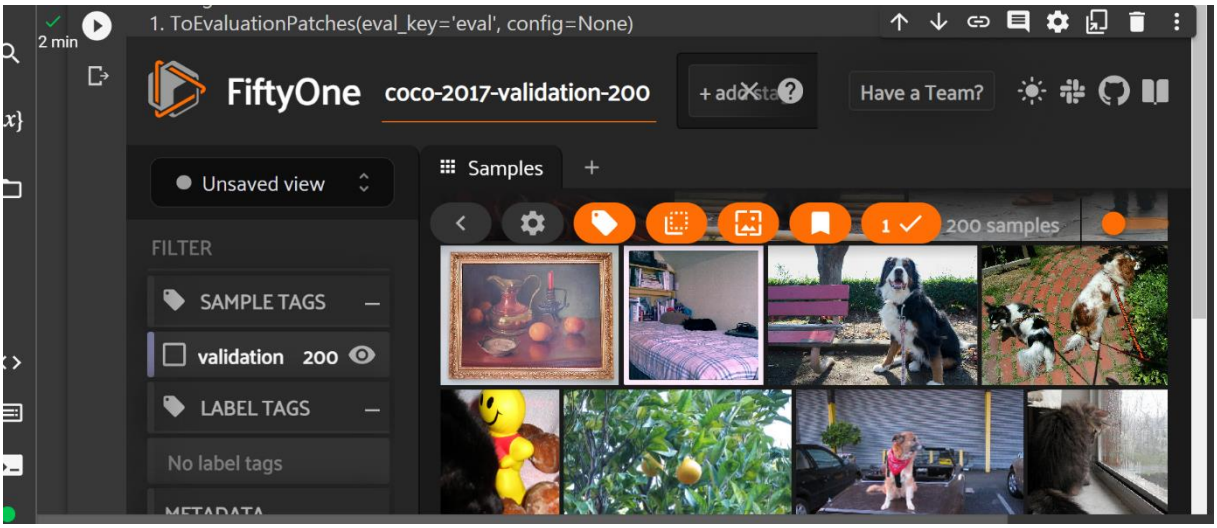
In fiftyone, there are methods that make it easy to set up an object detection task. The code below allows you to use the fasterrcnn pre-trained model in fiftyone:

```
# chargment du modèle pré-entrainé
model = foz.load_zoo_model("faster-rcnn-resnet50-fpn-coco-torch")
dataset.apply_model(model, label_field="predictions")

# détection
results = dataset.evaluate_detections(
    "predictions", gt_field="ground_truth", eval_key="eval", compute_mAP=True,
 classwise=False)

# affichage map
print(results.mAP())
# ex: 0.358
```

With fiftyone, we can explore the results via an interactive interface with this line of code.

```
1 import fiftyone as fo
2
3 #interface d'exploration des données
4 session = fo.launch_app(dataset)
5
```

On this interface, we can explore images by making filters as shown in the image below. For example, we could see images that are false positives and their metadata or true positives:



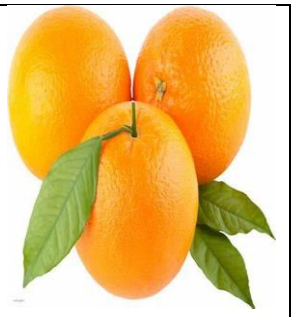
Comment:

The images on the coco dataset are noisy, some cats or dogs are hidden and the model cannot detect it.

Now let's try to test this model on our own images.

We have collected some images of cats, dogs, oranges and apples from the internet. We will use this pre-trained model and see if the model still detects well on the images that have been replaced to implement this algorithm;

Here is an extract from the images collected on the internet

			
cat	Apple	dog	orange

The code below will allow us to detect objects in images via the pre-trained model. We have limited the model to be detected to the four classes in our datasets

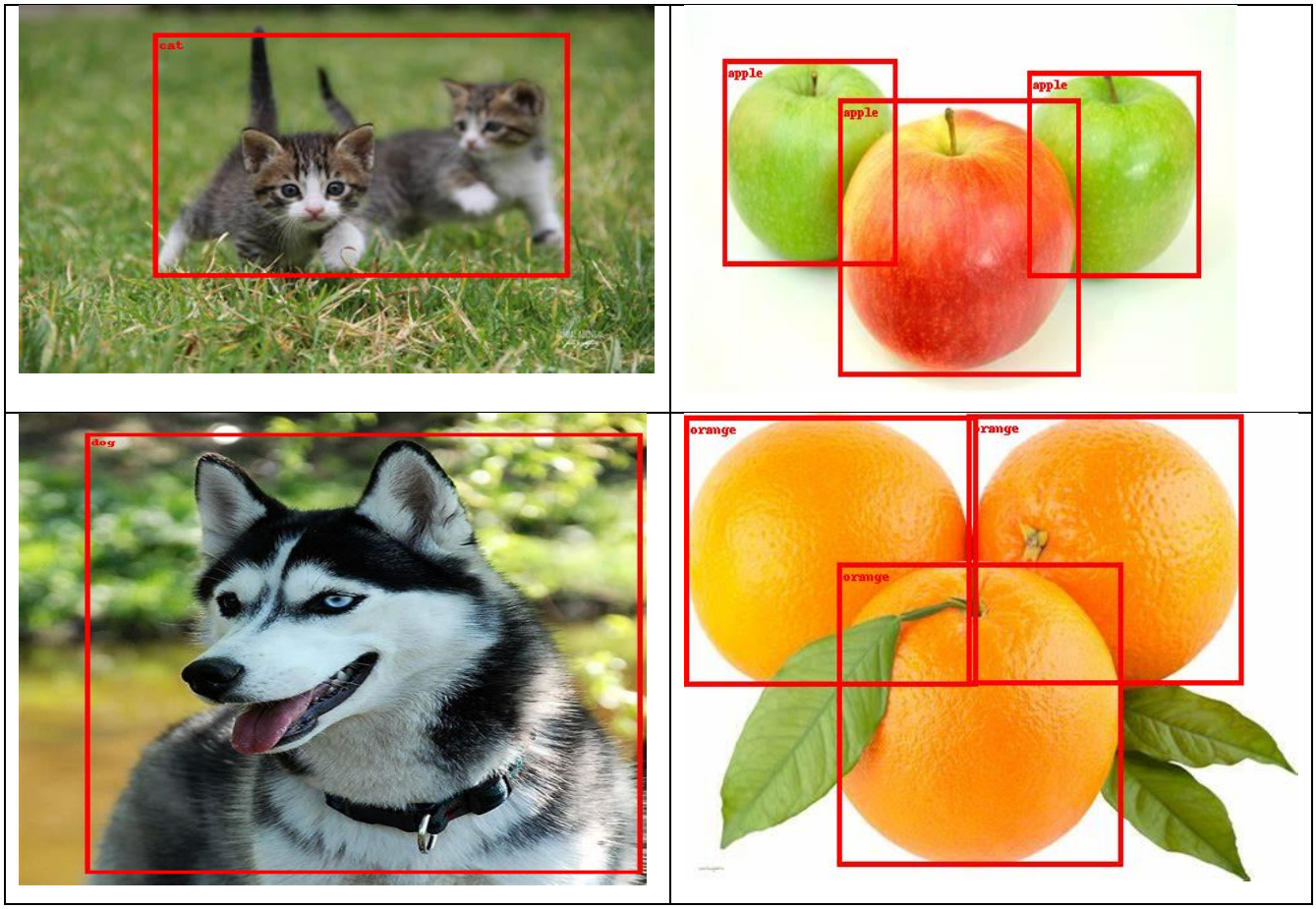
```
def detection_object(img):
  img = read_image(img)
  # initialisation du modèle avec les poids pré-entrainé
  weights =FasterRCNN_ResNet50_FPN_V2_Weights.COCO_V1.DEFAULT
  model = fasterrcnn_resnet50_fpn_v2(weights=weights, box_score_thresh=0.9)
  model.eval()
  # inference des poids pré-entrainé sur nos images
  preprocess = weights.transforms()
  batch = [preprocess(img)]
  #définissons nos classes
  name_classse=["cat","dog","orange","apple"]
  # application pour détecter les images
  prediction = model(batch)[0]
  labels = [weights.meta["categories"][i] for i in prediction["labels"]]
  labels=[label for label in labels if label in name_classse]
  box = draw_bounding_boxes(img, boxes=prediction["boxes"],
                            labels=labels,
                            colors="red",
                            width=4, font_size=30)
  im = to_pil_image(box.detach())
  im.show()
```

Let's retrieve the files of our images that are stored in the "Myimages" folder and detect

```
#utilisation de la fonction pour détecter les images
os.chdir("/content/My_images")
#chargement des images
path="/content/My_images"
url_img=[img for img in os.listdir(path)]
#détection
for img in url_img:
    detection_object(img)
```

Let's make a loop to detect our 4 classes in the images.

Qualitatively, the pre-trained model is easily able to detect our four object types. As shown in the extract of the results below:



Exploration of the pre-trained model by changing the parameters (fine tuning)

For fine tuning, we have just restricted the model to 4 classes and explored the results obtained

The following code shows us how to keep only the 4 classes we want to re-train

```
import fiftyone as fo
import fiftyone.zoo as foz
# objets à détecter
classes = ["dog", "cat","apple","orange"]
#initialisation du modèle pré-entrainé
model = foz.load_zoo_model("faster-rcnn-resnet50-fpn-coco-torch")

#résultats
results = dataset.evaluate_detections(
"predictions", gt_field="ground_truth", eval_key="eval_predictions", classwise=False,compute_mAP=True)
```

The fiftyone library allows you to explore the results thanks to the attributes and methods of the "results" object with the command

```
# Afficher les attributs et les méthodes de l'objet "results"
dir(results)
```

We could display the model's mAp and the metrics of each class with the report method

```
print(results.mAP())
# ex: 0.358
```

```
0.32907682987926523
```

```
#évaluation des resultats
res=results.print_report(classes=classes)
```

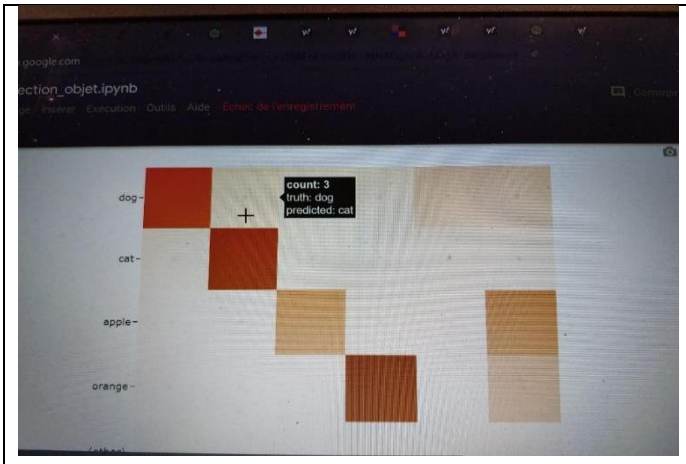
	precision	recall	f1-score	support
dog	0.63	0.86	0.72	83
cat	0.63	0.92	0.75	90
apple	0.33	0.50	0.40	112
orange	0.45	0.80	0.58	142
micro avg	0.48	0.76	0.59	427
macro avg	0.51	0.77	0.61	427
weighted avg	0.49	0.76	0.59	427

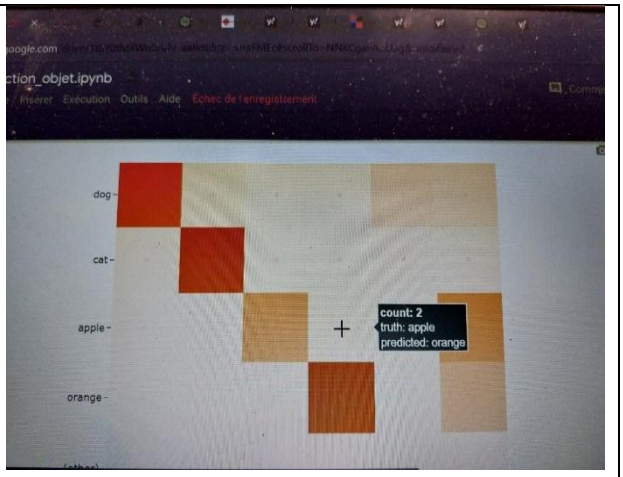
Comment:

The model has a maP of 0.32 which is low. The model has trouble detecting images. This may be due to the fact that the objects in the images are so hidden. Also, the accuracy of each class is low, because the model fails to detect the true labels of the objects.

We can also display the confusion matrix to see how the model is wrong to detect objects

```
#matrice de confusion
plot = results.plot_confusion_matrix(classes=classes)
plot.show()
```

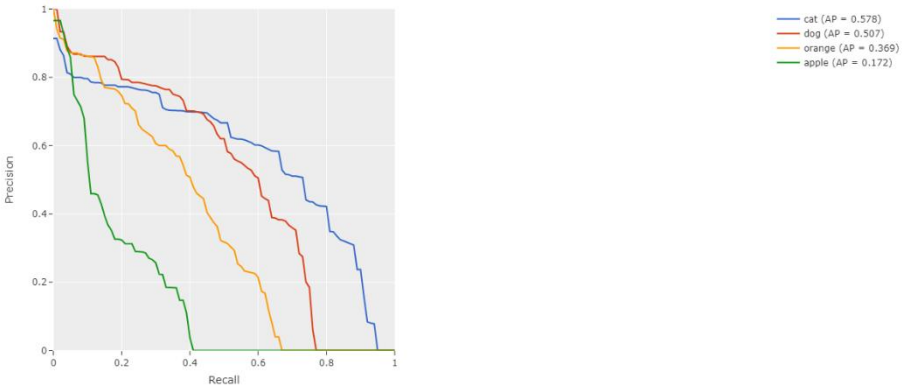




Comment:

Exploring this matrix, we see that 3 dogs were mispredicted by labelling it cat. So the model is wrong in this case. Just as 2 apples were mispredicted by detecting it as orange. The confusion matrix really allows us to understand in which classes the model is more wrong and in which class these objects have been misclassified.

We can also calculate the average accuracy curves (map) for each class:



Conclusion

Ultimately, the field of computer vision, especially object detection, is a proven task with several libraries and a strong community working in this area. However, there is still room for improvement, especially when we have noisy data that needs to be well processed first before being passed into the object detection models.

This project also taught us that there are many uses for computer vision in many fields, including the environment, health, agriculture, etc.

Bibliographic reference

1- fasterRCNN models: Fatsercnn-resnet

2- article to learn fitfyone : Article fiftyone