

BÁO CÁO ĐỒ ÁN CUỐI KỲ

Lớp: CS114.L11.KHCL
Môn: MÁY HỌC

GV: PGS.TS Lê Đình Duy – THS. Phạm Nguyễn Trường An
Trường ĐH Công Nghệ Thông Tin, ĐHQG-HCM

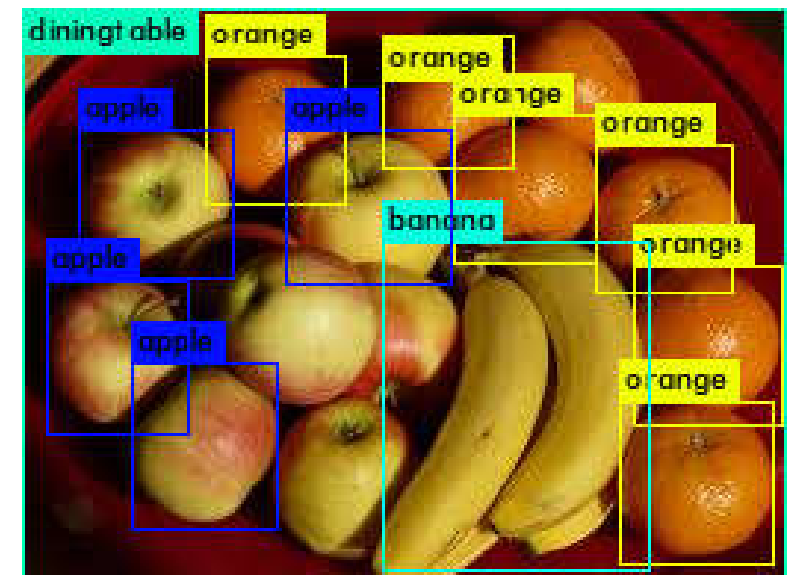
Banana Detection

Nhóm 04 – CS114.L11.KHCL

Link Github: [Nhom04](#)

Banana Detection

- Là đề tài ứng dụng công nghệ Deep Learning để phát hiện, xác định vị trí của chuối trên ảnh
- Có thể phát triển lên và ứng dụng cho các công việc liên quan xác định, nhận dạng các loại trái cây (trong bài này là chuối),vv...



Banana Detection

- Input: một bức ảnh
- Output: các cặp tọa độ để vẽ ra hình chữ nhật có chứa 1 quả chuối trong đó

Input



Output

```
array([[326, 339, 441, 654],  
       [391, 384, 506, 745],  
       [480, 325, 607, 723]], dtype=int32)
```

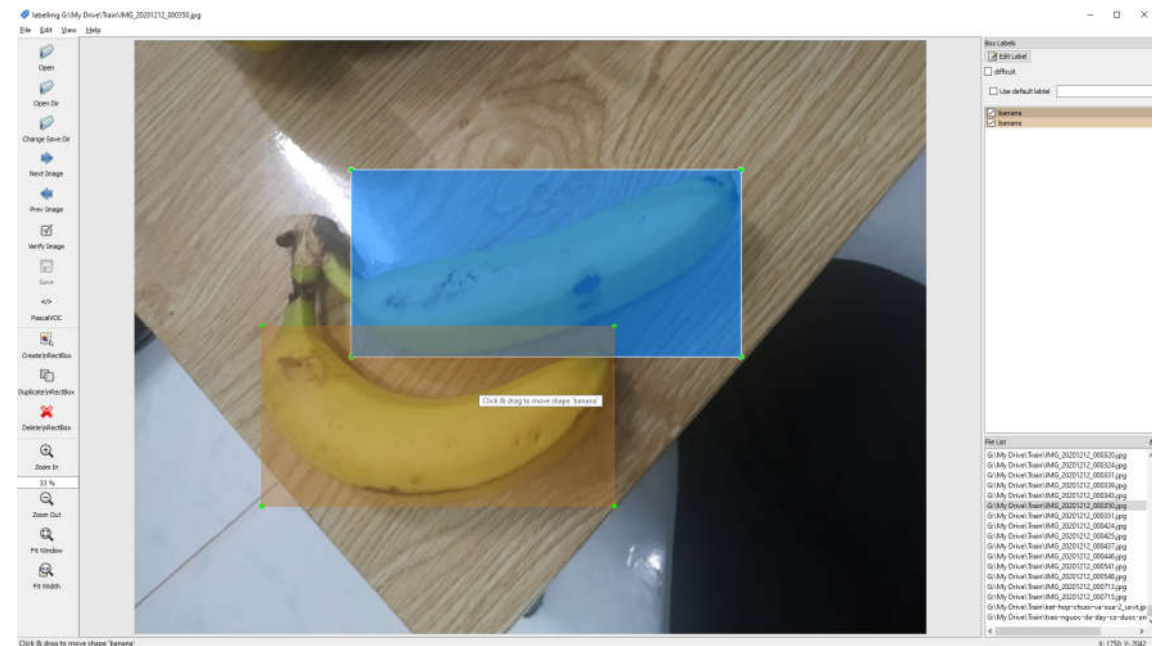
Dataset

- Được thu thập thủ công bởi các thành viên của nhóm
- Tuân theo một số quy tắc đã định (mô tả ở notebook trên github)



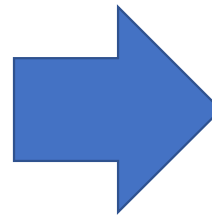
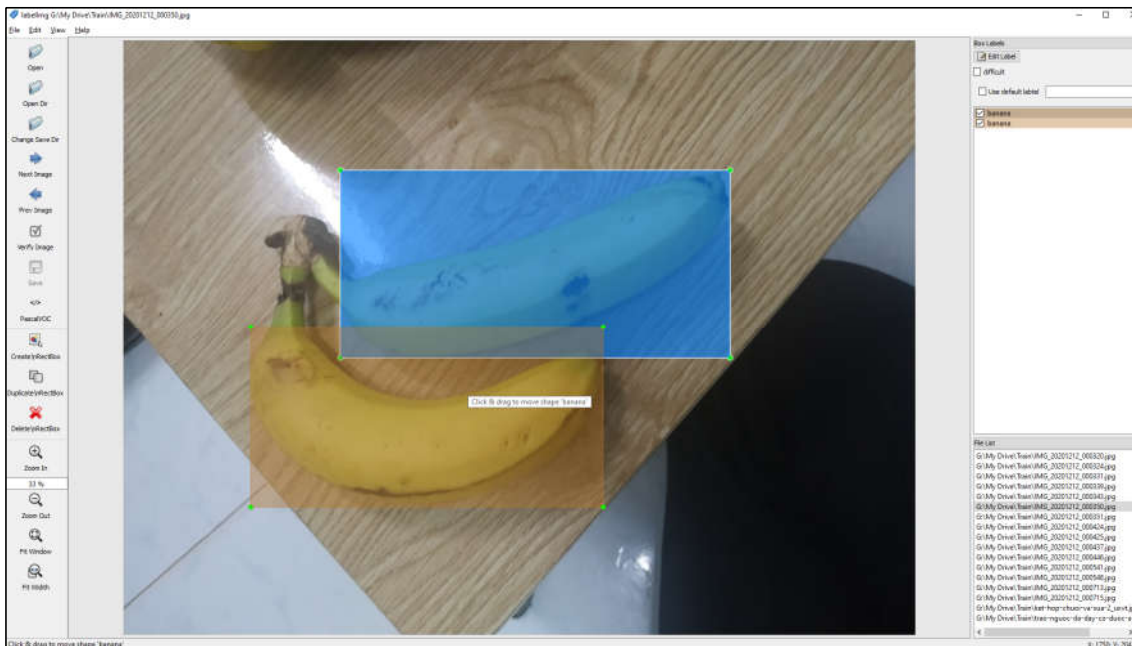
Dataset

Được gán nhãn thủ công dựa vào công cụ labellmg



Dataset

Lưu dưới dạng file xml



```
IMG_20201212_000350.xml - Notepad
File Edit Format View Help
<annotation>
  <folder>My Drive</folder>
  <filename>IMG_20201212_000350.jpg</filename>
  <path>G:\My Drive\IMG_20201212_000350.jpg</path>
  <source>
    <database>Unknown</database>
  </source>
  <size>
    <width>4032</width>
    <height>3024</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
    <name>banana</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>1105</xmin>
      <ymin>659</ymin>
      <xmax>3089</xmax>
      <ymax>1615</ymax>
    </bndbox>
  </object>
  <object>
    <name>banana</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>649</xmin>
      <ymin>1456</ymin>
      <xmax>2443</xmax>
      <ymax>2374</ymax>
    </bndbox>
  </object>
</annotation>
```

Algorithm

- Mask RCNN là deep neural network nhằm đến việc giải quyết vấn đề về segmentation trong máy học hoặc thị giác máy tính. Nghĩa là, nó có thể tách các vật thể khác nhau trong một bức ảnh hoặc video, nó cho bạn bounding boxes, class, mask của vật thể.
- Quá trình phát triển: R-CNN (Region-based Convolutional Neural Networks) -> Fast R-CNN -> Faster R-CNN -> Mask R-CNN.

Mask R-CNN

- Các modules của thuật toán:
 - + Backbone
 - + Region Proposal Network (RPN)
 - + ROI Classifier & Bounding Box Regressor
 - + Segmentation Masks

Backbone

- Là convolutional neural network cơ bản (thông thường là ResNet50 hoặc Resnet101) phục vụ cho việc rút trích đặc trưng
- Các layer đầu sẽ phát hiện các đặc trưng low level (góc và cạnh), và các layer về sau sẽ phát hiện các đặc trưng high level (xe, người, vv...)
- Cho ảnh đi qua backbone network, ảnh sẽ được chuyển từ kích thước $1024 \times 1024 \times 3$ thành feature map có kích thước $32 \times 32 \times 2048$ để phục vụ cho các bước sau.

Region Proposal Network (RPN)

- Là lightweight neural network cho phép scan bức ảnh để tìm ra vùng chứa vật thể
- Những vùng mà RPN scan qua được gọi là các anchors, mà chính là các boxes được phân bố phủ khắp bức ảnh
- RPN làm việc khá nhanh do scan trên backbone feature map
- Việc này cho phép RPN sử dụng lại các feature đã trích xuất 1 cách hiệu quả và tránh các tính toán trùng lặp.
- RPN tạo ra 2 output với mỗi anchor: Anchor class(foreground or background), bounding box refinement(% change in x,y,width,height)

ROI Classifier & Bounding Box Regressor


- Run regions of interest (ROIs) đề xuất bởi RPN
 - Cũng như RPN, trả về 2 output:
 - Anchor class: khác với RPN, network này có thể phân loại các vùng thành các class cụ thể (người, ghế, xe,...vv). Nó cũng thể tạo ra background class, khiến cho ROI bị bỏ đi
 - Bounding box refinement: tương tự như RPN, mục đích cũng là điều chỉnh lại vị trí và kích cỡ của bounding box bao quan vật thể.
 - Các classifiers không xử lý tốt input có kích cỡ thay đổi (yêu cầu fixed input), nhưng ở bước bounding box refinement ở RPN, các ROI boxes có thể có kích cỡ khác nhau
- => ROI Pooling: crop 1 phần của feature map và resize về 1 fixed file

Segmentation Masks

- Thực hiện công đoạn mask branch
- Là convolutional network mà nó chọn ra các positive mask từ ROI classifier và tạo masks cho chúng.
- Các mask được tạo ra có độ phân giải thấp: 28x28 pixels. Nhưng chúng là các soft masks, biểu diễn bởi số thực, nên chúng sẽ chứa đựng nhiều thông tin hơn binary masks
- Kích cỡ các mask nhỏ giúp mask branch nhẹ hơn.

Evaluation (using IoU and mAP)

- IoU (Intersection over Union)
- Dùng để xác định 1 dự đoán có chính xác hay không
- Được tính bằng lấy phần giao giữa ô dự đoán và ô thực tế chia cho phần hợp giữa chúng.
- Một dự đoán được coi là True Positive nếu $\text{IoU} > \text{threshold}$, False Positive nếu $\text{IoU} < \text{threshold}$.

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$


mAP (mean Average Precision)

- Để tính mAP, ta cần tính AP(Average Precision) trên từng lớp (trong trường hợp này là lớp 'banana')

- Cách tính AP:

- Với mỗi ảnh dự đoán, ta tính precision, recall dựa trên True Positive đã xác định dựa vào độ đo IoU

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{TP}}{\# \text{ ground truths}}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{\text{TP}}{\# \text{ predictions}}$$

mAP (mean Average Precision)

- AP là mean của giá trị precision tại tập tất cả các độ đo recall

$$AP = \frac{1}{n} \sum_{r > \bar{r}} P(r)$$

- Giá trị precision tại mỗi độ đo recall r với $r >$ một ngưỡng \bar{r} cho trước, n là số lượng các giá trị recall r thỏa
- mAP = trung bình cộng AP của tất cả các class (trong trường hợp 1 class nên mAP = AP)

Implementation

- Cài đặt các thư viện cần thiết: tensorflow, keras,..vv
- Clone thư viện Mask RCNN từ github về
- Cài đặt thông số cho Detector Config:
 - Tên class: 'banana'
 - Batch size (image per GPU): 16 (điều chỉnh lúc chạy)
 - Số class: 2 (BG + 1 fruit class)
 - Kích cỡ ảnh để resize: 256, 256
 - ...vv

Implementation

- Chuyển xml file sang dict
- Load các ảnh tương ứng với mask của nó
- Chuẩn bị tập train và tập test
- Download file COCO weight
- Tạo training model và load weight
- Bắt đầu train
- Lưu file weight cuối cùng sau khi train
- Đánh giá bằng mAP

Evaluation

- Độ đo mAP của tập train là 0.831
- Độ đo mAP của tập test là 0.716

Comparision

- Một số kết quả thu được



Hyperparameter Tunning

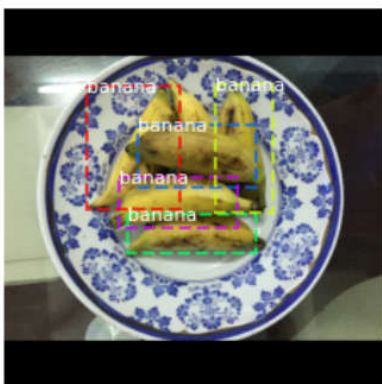
- Thay đổi các giá trị image size, backbone như sau:
 - Case 0 (base case): image size = (256, 256), backbone = 'resnet50'
 - Case 1: image size = (512, 512), backbone = 'resnet50'
 - Case 2: image size = (1024, 1024), backbone = 'resnet50'
 - Case 3: image size = (256, 256), backbone = 'resnet101'
 - Case 4: image size = (512, 512), backbone = 'resnet101'
 - Case 5: image size = (1024, 1024), backbone = 'resnet101'
- Lần lượt chạy các model và so sánh kết quả:

Result

	mAP (train)	mAP (test)
256x256 – resnet50	0.831	0.716
512x512 – resnet50	0.642	0.660
1024x1024 – resnet50	0.700	0.682
256x256 – resnet101	0.822	0.736
512x512 – resnet101	0.846	0.787
1024x1024 – resnet101	0.821	0.767

So sánh model

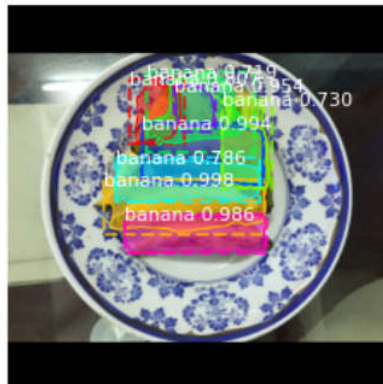
Mask gốc



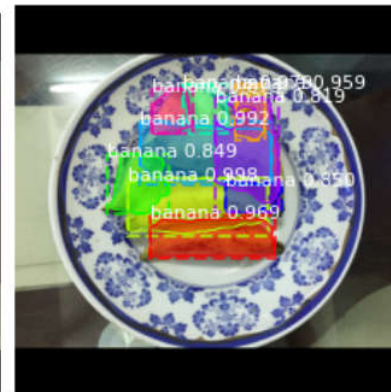
(256,256)-resnet50



(512,512)-resnet50



(1024,1024)-resnet50



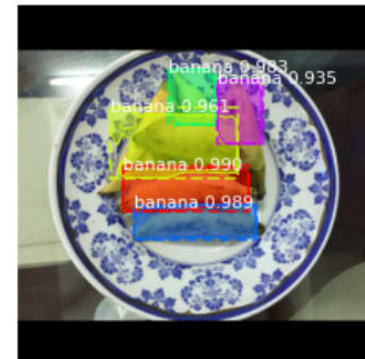
(256,256)-resnet101



(512,512)-resnet101



(1024,1024)-resnet50



So sánh model

Mask gốc



(256,256)-resnet50



(512,512)-resnet50



(1024,1024)-resnet50



(256,256)-resnet101



(512,512)-resnet101



(1024,1024)-resnet50



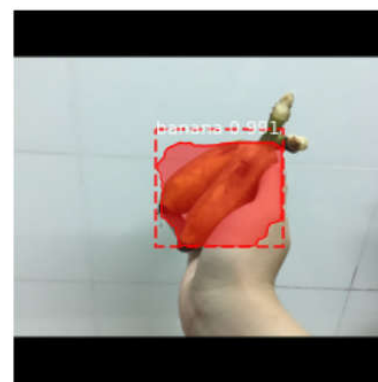
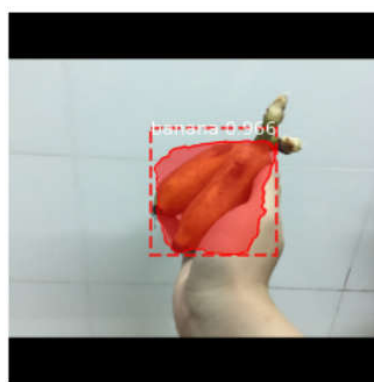
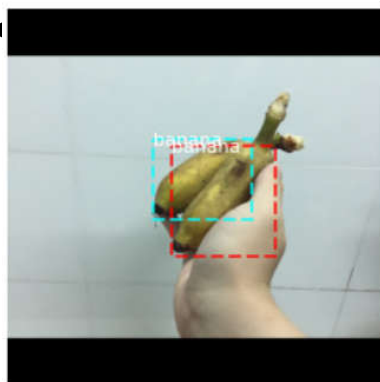
So sánh model

Mask gốc

(256,256)-resnet50

(512,512)-resnet50

(1024,1024)-resnet50



(256,256)-resnet101

(512,512)-resnet101

(1024,1024)-resnet50



Kết luận

- Model có kích cỡ ảnh 512x512, sử dụng ResNet 101 cho kết quả mAP trên cả 2 tập train và test cao nhất