

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH**

**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

**---------****---------**

**BÁO CÁO ĐỒ ÁN 3**

**NHẬN DIỆN TRÁI CÂY VỚI PHƯƠNG PHÁP MASK RCNN**

**Sinh viên thực hiện:**

**17110185 Nguyễn Hữu Nam**

**17110132 Phạm Thị Thanh Hằng**

**GVHD: Thầy Trần Tiến Đức**

**TP. Hồ Chí Minh, tháng 12 năm 2020**

# PHIẾU GIAO NHIỆM VỤ ĐỒ ÁN

**Thông tin sinh viên**

Tên: Nguyễn Hữu Nam

Sđt: 0964056715

Email: [17110185@student.hcmute.edu.vn](mailto:17110185@student.hcmute.edu.vn)

Tên: Phạm Thị Thanh Hằng

Sđt: 0388404061

Email: [17110132@student.hcmute.edu.vn](mailto:17110132@student.hcmute.edu.vn)

**Mục đích nội dung của Đồ Án**

Nghiên cứu nhận diện hình ảnh trái cây trong hình ảnh bằng phương pháp Mask RCNN

**Các nhiệm vụ cụ thể của Đồ Án**

Tìm hiểu bài toán phát hiện và phân vùng đối tượng trái cây trong hình ảnh và hướng giải quyết

Tìm hiểu và thử nghiệm kỹ thuật mạng neuron tích chập nhằm phát hiện và phân đoạn vùng trái cây (Mask R-CNN).

Nghiên cứu và thử nghiệm kỹ thuật theo vết Mean Shift để nâng cao độ chính xác của giải thuật Mask R-CNN

Đánh giá độ chính xác, ưu nhược điểm của kỹ thuật nghiên cứu.

**Lời cam đoan của sinh viên**

Các kết quả nêu trong đồ án là trung thực, không phải là sao chép toàn văn của bất kỳ công trình nào khác.

**Links:**

Video demo: <https://youtu.be/CWaxOh67is4>

Github: <https://github.com/nguyenhuunam852/MaskRCNN?fbclid=IwAR0g_i5PPazJPrmrXsj5_txFQvEl4JSmFxUuJMc2_Db29FeRZcuXt9q1yow>

Contents

[PHIẾU GIAO NHIỆM VỤ ĐỒ ÁN 2](#_Toc63078046)

[TÓM TẮT NỘI DUNG ĐỒ ÁN 4](#_Toc63078047)

[LỜI CẢM ƠN 5](#_Toc63078048)

[CHƯƠNG 1: QUÁ TRÌNH CHUẨN BỊ DỮ LIỆU 1](#_Toc63078049)

[CHƯƠNG 2: THUẬT TOÁN MASK-RCNN VÀ QUÁ TRÌNH TRAINNING 2](#_Toc63078050)

[2.1 MẠNG NEURAL 2](#_Toc63078051)

[2.2 MẠNG NEURON TÍCH CHẬP 3](#_Toc63078052)

[2.3 KIẾN TRÚC CNN 4](#_Toc63078053)

[2.4 THUẬT TOÁN FAST-RCNN 5](#_Toc63078054)

[2.5 FASTER R-CNN 6](#_Toc63078055)

[2.6THUẬT TOÁN MASK-RCNN 7](#_Toc63078056)

[2.7 KIẾN TRÚC MẠNG MASK-RCNN 8](#_Toc63078057)

[2.8 ROIALIGN 9](#_Toc63078058)

[2.9 QUÁ TRÌNH TRAINING VÀ PREDICTION 10](#_Toc63078059)

[CHƯƠNG 3 CODE 11](#_Toc63078060)

[3.1 CÁC BƯỚC THỰC HIỆN 11](#_Toc63078061)

[Bước 1: Khai báo thư viện cần thiết 11](#_Toc63078062)

[Bước 2 : Định nghĩa đường dẫn cho pretrained weight và hình ảnh 11](#_Toc63078063)

[Bước 3: Tạo ra model và load pretrain weight 13](#_Toc63078064)

[Bước 4: Định nghĩa các lớp 13](#_Toc63078065)

[Bước 5: Load hình ảnh 14](#_Toc63078066)

[Bước 6: Predict 14](#_Toc63078067)

[3.2 KẾT QUẢ 15](#_Toc63078068)

[THAM KHẢO 19](#_Toc63078069)

# TÓM TẮT NỘI DUNG ĐỒ ÁN

Bài toán phát hiện và phân đoạn vùng trái cây trên ảnh có rất nhiều ứng dụng trong các hệ thống công nghiệp hiện nay như hệ thống đóng gói và phân loại trái cây. Vấn đề chính đặt ra trong bài toán này là phát hiện có hay không đối tượng trái cây (hand detection), nếu có xác định vùng của trái cây đó trên ảnh (hand segmentation). Trong thời gian gần đây, có một số mô hình giải quyết khá tốt cho bài toán phát hiện và phân vùng đối tượng nói chung dựa trên các kỹ thuật học sâu tiên tiến như Fast R-CNN, Faster R-CNN, Mask R-CNN. Trong khuôn khổ của đồ án sẽ tập trung vào mạng Mask R-CNN. Đây là một kiến trúc mạng neuron sâu cho kết quả phân đoạn tốt các lớp đối tượng trên ảnh như người, xe, con vật. Tuy nhiên chưa có một công việc nào sử dụng mạng Mask R-CNN cho bài toán phát hiện và phân vùng trái cây trong ảnh và video. Đồ án nghiên cứu và đánh giá tính khả thi của kiến trúc Mask R-CNN để giải quyết bài toán phát hiện và phân vùng các loại trái như một bước tiền xử lý trong các hệ thống công nghiệp trong phân loại và đóng gói. Với giai đoạn tiền xử lí thì việc tạo ra những hình ảnh rõ ràng và đúng tỉ lệ về từng loại trái là cần thiết nhằm cải thiện quá trình huấn luyện và validation.

# LỜI CẢM ƠN

Lời đầu tiên, cho phép nhóm em được gửi lời cảm ơn chân thành đến thầy Thầy Trần Tiến Đức, người đã phụ trách hướng dẫn nhóm chúng em trong việc thực hiện và hoàn thành bài báo cáo một cách tốt và hiệu quả nhất. Những sự chỉ dạy, nhật xét, đóng góp ý kiến từ thầy đã góp phần rất quan trọng trong việc giúp chúng em hoàn hành đề tài một cách hoàn thiện nhất.

Cảm ơn các thầy cô trong khoa Công Nghệ Thông Tin đã luôn nhiệt tình, cống hiến, tận tình giải đáp các thắc mắc của chúng em. Bên cạnh đó, chúng em xin được cảm ơn các bạn cùng khóa, các anh chị khóa trên đã không ngần ngại chia sẻ các thông tin, kinh nghiệm bổ ích giúp chúng em hoàn thiện hơn đề tài của mình.

Báo cáo được thực hiện trong một khoảng thời gian có hạn, cùng với những hạn chế về mặt kiến thức nên việc sai sót là điều không thể tránh khỏi trong quá trình thực hiện. Chúng em kính mong nhận được những lời nhận xét, ý kiến đóng góp quý báo từ quý Thầy Cô để chúng em có thêm kinh nghiệm và hoàn thành tốt hơn những đề tài sau này.

Chúng em xin chân thành cảm ơn !

# CHƯƠNG 1: QUÁ TRÌNH CHUẨN BỊ DỮ LIỆU

Bộ dự liệu thu thập gồm có 5 class gồm hình ảnh các loại trái cây đơn lẻ để dễ dành cho quá trình đánh nhãn và huấn luyện :

* MÍT : gồm 202 hình
* KHẾ : gồm 171 hình
* CÓC : gồm 144 hình
* CAM : gồm 129 hình
* BƯỞI : gồm 151 hình

Việc chuẩn bị dữ liệu chuẩn là tối quan trọng trong các thuật toán AI. Khi mà việc tối ưu các Features, loại bỏ những Sample bị lỗi là cần thiết để quá trình Training Data trở nên nhanh và tối ưu hơn. Và việc có bộ dữ liệu tốt cũng sẽ tránh được underfitting và cả overfitting.



# CHƯƠNG 2: THUẬT TOÁN MASK-RCNN VÀ QUÁ TRÌNH TRAINNING

## 2.1 MẠNG NEURAL

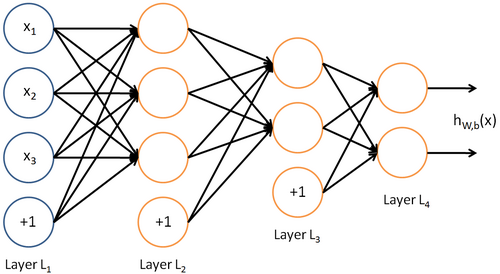
Mạng Neuron trong Machine Learning nhằm mô phỏng hệ thống Neuron của con người. Với mỗi phần tử tính toán nhỏ ta sẽ gộp chúng thành một mạng lưới có khả năng tính toán lớn.Phần tử tính toán của một mạng Neural là một Perceptron hay còn gọi là 1 Neuron

Cấu trúc của 1 Neural trong bộ não con người gồm :

* Dendrite là nơi đưa dữ liệu tới Neuron
* Nucleus để tính toán
* Axon là nơi xuất ra tín hiệu
* Các Neural được liên kết thông qua các dây Synapse

Tương tự với cấu trúc của 1 Neuron nhân tạo. Được cấu tạo từ rất nhiều Node. Các Node có trong 1 Neuron gồm :

* Input Node: Nới chứa Input từ bên ngoài
* Hidden Node : Nới nhận dữ liệu từ các Input Node và tính toán và đưa ra các Output. Tập hợp các Node ẩn trong mạng Neural tạo ra tầng ẩn.
* Output Node : tính toán và đưa dữ liệu ra ngoài



## 2.2 MẠNG NEURON TÍCH CHẬP

Mạng Neuron tích chập là mạng Neuron nhân tạo có các phần tử tích chập. Nó có thể học một lượng lớn dữ liệu trong một khoảng thời gian ngắn hơn nhiều so với mạng Neuron thông thường. Với việc sử dụng ít trọng số hơn (Giống trong các Linear Models khi việc thêm siêu tham số làm tăng mũ của phương trình và đồng thời tạo ra nhiều bộ Theta hơn khiến nó trở nên phức tạp ) nên kết quả của nó cũng chỉ chênh lệch khá ít so với cách thông thường.

Trong liên kết giữa 2 tầng liên kết nhau thì việc các Node ở các tầng phía sau chỉ liên kết với một bộ phận các Node ở tầng phía trước đó (receptive field) thông qua một tập các trọng số. Hơn nữa các trọng số là như nhau với mỗi Neuron tầng sau nên số lượng tham số sẽ nhỏ hơn rất nhiều (Hạn chế sự bùng nổ tham số )

Mạng Neural tích chập sẽ gồm 4 bước sau :

* Convolutional Layer
* Pooling Layer
* Non-Linear Layer
* Fully-connected Layer

Mạng Neural tích chập chính là một kiến trúc quan trọng trong các thuật toán Machine Learning liên quan tới xử lí hình ảnh



## 2.3 KIẾN TRÚC CNN

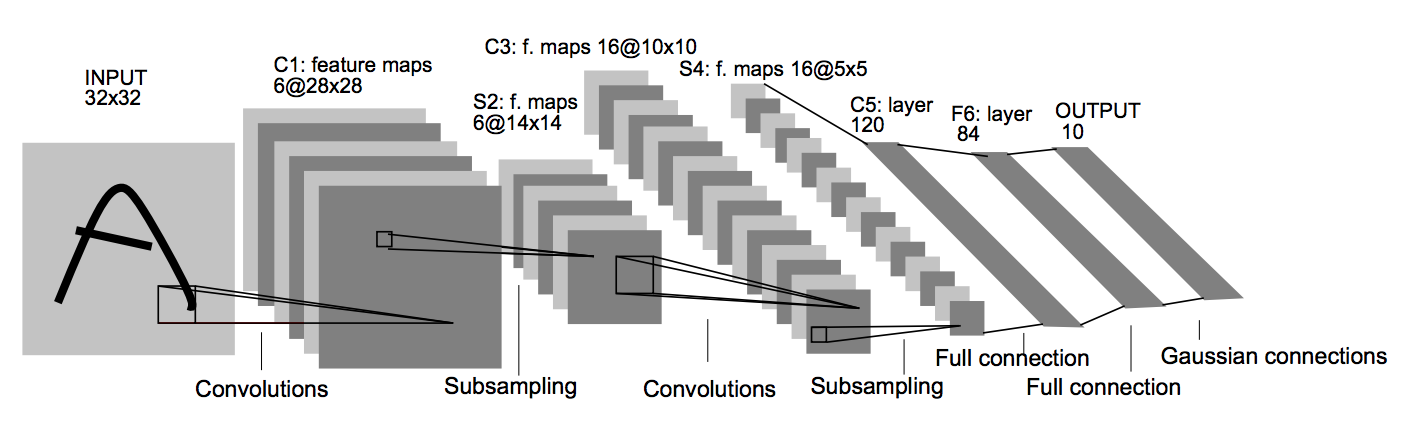
R-CNN là kiến trúc dùng trong các bài toán phát hiện và phân vùng đối tượng được sử dụng trong các bài toán nhận diện hình ảnh

Thứ tự các bước gồm:

* Tạo ra các Boundary Box trên ảnh bằng các thuật toán như selective search
* Đóng gói dữ liệu từ vùng đề xuất để đưa vào mạng đã được học sẵn để rút trích đặc trưng
* Truyền đặc trưng trích dẫn được từ bước trên đưa qua bộ phận lớp tách rời có thể là một mạng kết nổi đầy đủ hay một SVM để kiểm tra Boundary Box có thật sự chứa đối tượng quan tâm hay không

Các vấn đề của RCNN

* Vấn đề tốc độ do RCNN thực hiện CNN cho mỗi Boundary Box và việc tạo ra các box bên ngoài vùng cần thiết là rất nhiều
* Sử dụng Selective Search vốn đã mang tính chất vét cạn và cực kì hao tốn



## 2.4 THUẬT TOÁN FAST-RCNN

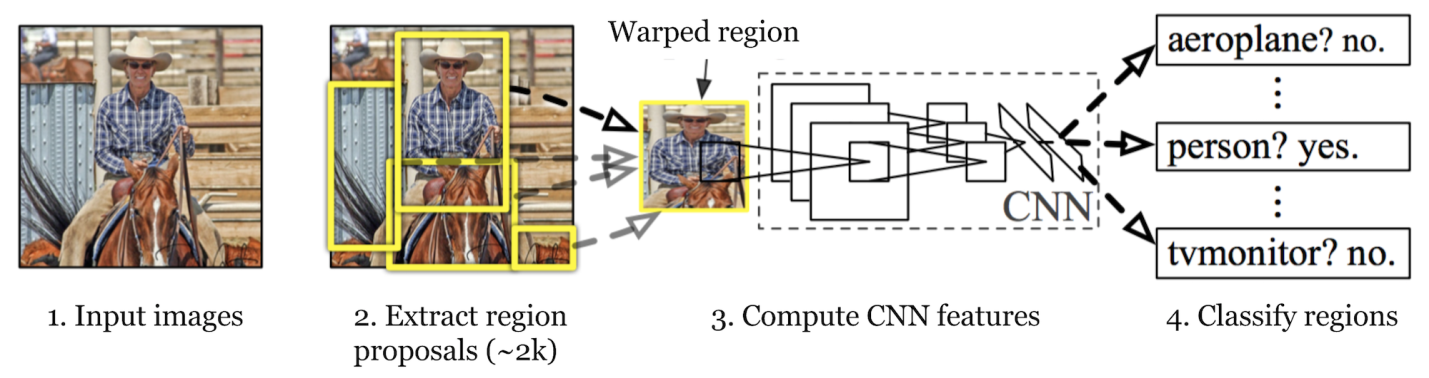
Để khắc phục nhược diểm của CNN thì Fast-CNN ra đời. Và điểm mạnh của nó chính chắc chắn là sự cải thiện rõ ràng về tốc độ(khoảng 200 lần).Ý tưởng của Fast-CNN là sử dụng một mạng Neural duy nhất cho việc trích xuất đặc trưng thay.

Để phát hiện đối tượng một cách độc lập với kích thước ảnh. Fast-RCNN sử dụng tầng g Spatial Pyramid Pooling . Thay vì cắt ảnh đầu vào thành nhiều phần khác nhau . Fast\_RCNN tính toán đặc trưng một lần qua mạng trích xuất đặc trừng và nah1 xạ vugn2 quan tâm trên ảnh.

Mỗi vùng quan tâm trên đặc trưng được pooling có kích thuớc khác nhau rồi ghép lại thành một đặc trưng có kích thước cố định và đưa vào tầng tiếp theo. Nhờ đó các đặc trưng là đập lập so với kích thước hình ảnh.

Kiến trúc này có thể tái sử dụng cho các tầng trên thay vì phải tính hàng nghìn lần như RCNN.

Spatial Pyramid Pooling có vai trò như một công cụ chuẩn hóa đầu vào cho fully connected layer nhờ sự hộ trợ của ROI với việc loại bỏ những bounding box với các kích thước khác nhau sau đó sẽ được loại bỏ dần do có điểm IOU(Điểm đánh giá độ quan tâm ) quá nhỏ (<0.5)



## 2.5 FASTER R-CNN

Nhanh hơn Fast R-CNN một mô hình có tên là Faster R-CNN được đề xuất trong [19].

Faster R-CNN cải tiến hơn so với Fast R-CNN bằng việc thay thế thuật toán sinh vùng độc

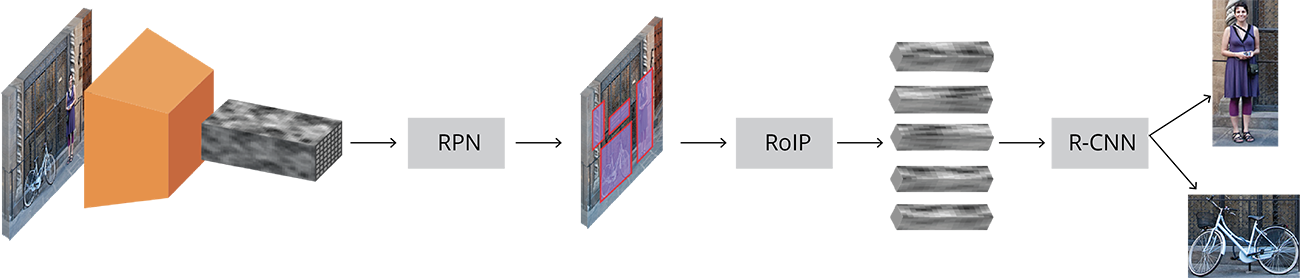
lập với mô hình. Mạng Faster R-CNN chỉ cần sử dụng một mạng nơ ron duy nhất để thực

hiện nhiệm vụ sinh vùng đặc trưng bằng việc thêm vào mạng một số tầng mới làm nhiệm vụ

đề xuất vùng đặc trưng khối này có tên Region Proposal Network là một mạng tích chập đầy đủ (Fully convolution network) ko có các tầng kết nối đầy

đủ do đó chi phí tính toán không đáng kể khoảng 10ms cho mỗi ảnh. Ngoài ra Faster R-CNN

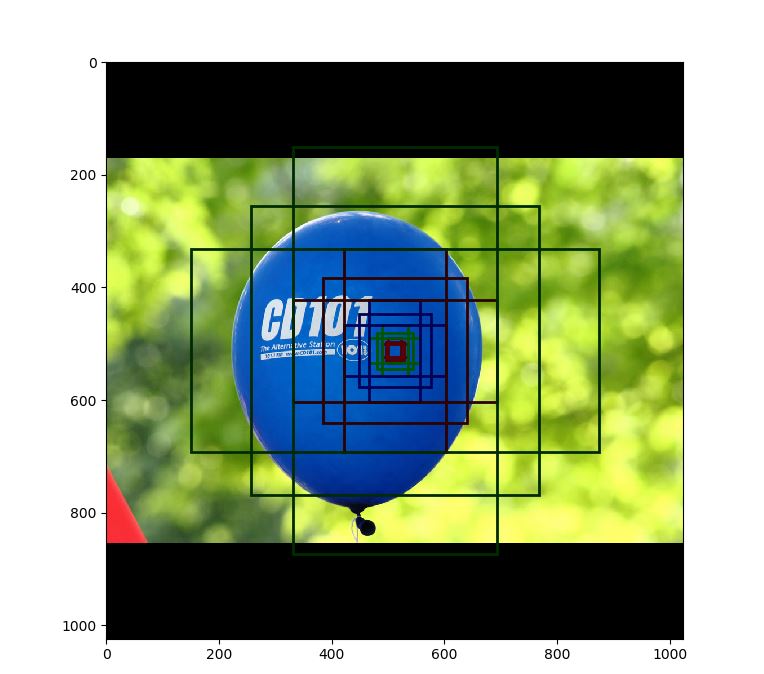
còn thay thế các Spatial Pyramid bằng các Anchor với các tỉ lệ khung khác nhau.



## 2.6THUẬT TOÁN MASK-RCNN

Việc loại bỏ các bounding box không cần thiết đã giúp cho thuật toán nhanh hơn một cách đáng kể. Tuy Fast\_RCNN đã cải thiện về tốc độ nhưng vẫn còn quá chậm nếu so với

Mask\_RCNN. Và cũng là mô hình tốt nhất trên tập COCO.



## 2.7 KIẾN TRÚC MẠNG MASK-RCNN

Mask R-CNN mở rộng từ Faster R-CNN bằng cách thêm nhánh dự đoán Object Mask trên mỗi vùng đề xuất từ RPN. Phần còn lại tương đối giống với Faster\_RCNN

Nhánh trên dữ đoán bounding box và lớp đối tượng trên ảnh . Và nhánh dưới là nhánh đánh nhãn cho mỗi Pixel trong vùng quan tâm (ROI) để xây dựng Object Mask

MASK\_RCNN = FASTER\_RCNN + FCN

FCN có nhiệm vụ phát hiện đối tượng và phân vùng đối tượng. Mạng Neural bình thường được dùng để phát hiện và nhận diện đối tượng ở tầng cuối thường là một Vector có cùng kích thước với số lớn và cho biết điểm số dự đoán của mỗi lớp.Nếu ta thay thế Vector với một số bước tích chập. Thay  vì một Vector có kích thước bằng số lớp ta sẽ có được số đặc trưng cùng kích thước với số lớp.Sau quá trình học ta có được điểm số dự đoán cho tất cả điểm ảnh của lớp cuối cùng. Và cuối cùng mỗi lớp sẽ nhận được một bản đồ nhiệt

Sau đó trải qua 2 giai đoạn là deconvolution và unpooling để thu được Mask trên ảnh gốc

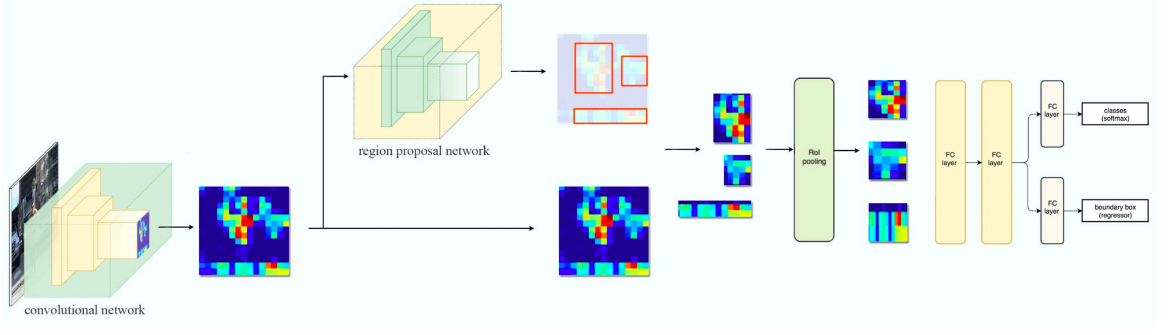
* Deconvolution (giải tích chập): thực chất chỉ là tích chập với ma trận chuyển vị
* Unpooling: là quá trình ta xây dựng lại ma trận bằng cách ghi nhớ tọa độ của điểm max và điền lại chính xác điểm max còn lại các điểm khác trong khối sẽ được xấp xỉ từ điểm đã có giá trị. Thông tin mất mát nhưng trong trường hợp này chúng ta có thể chấp nhận được.

Qua việc deconvolution và unpooling, chúng ta có thể xây dựng một phân vùng dự đoán

trên ảnh gốc cho tất cả các lớp đối tượng. Đây cũng chính là đầu ra cho khối phân vùng đối tượng.

Ngoài ra so với Faster R-CNN Mask RCNN có một cải tiến đó là thay thế việc sử

dụng khôi RoI Pooling bằng một khối có tên là RoIAlign. Theo tác giả đây là phần rất quan trọng để cải thiện độ chính xác cho Mask RCNN

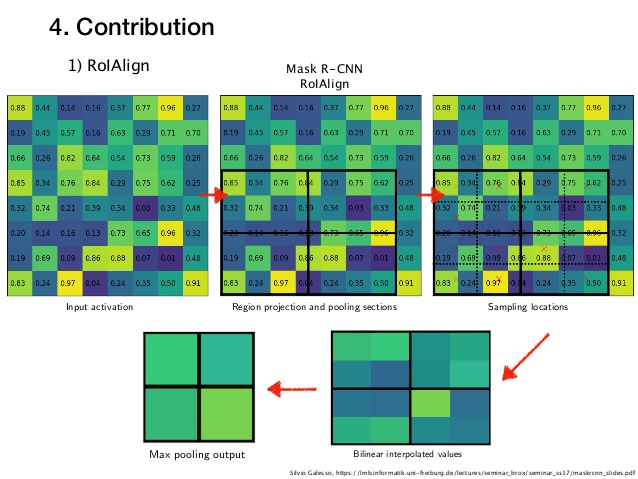


## 2.8 ROIALIGN

Thay vì việc tính mỗi điểm trên đặc trưng thông qua việc lượng tử hóa từng khổi nhỏ trên

ma trận để thu được kích cỡ cố đihj thì RoIAlign thực hiện một phép nội suy phi tuyến để

tính ra từ đặc trưng của mỗi vùng



## 2.9 QUÁ TRÌNH TRAINING VÀ PREDICTION

Bước 1: Clone the repository

git clone <https://github.com/matterport/Mask_RCNN.git>

**Bước 2: Cài đặt các thư viện liên quan dưới đây**

numpy

scipy

Pillow

cython

matplotlib

scikit-image

tensorflow>=1.3.0

keras>=2.0.8

opencv-python

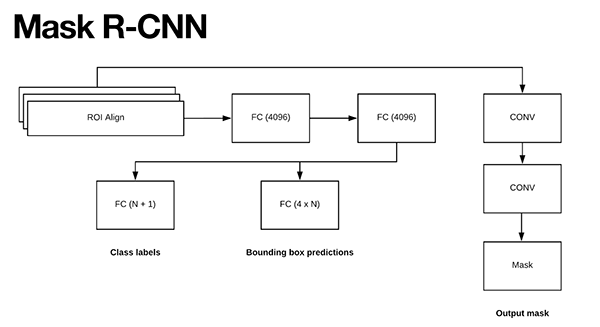
h5py

imgaug

IPython

**Bước 3: Chuẩn bị dữ liệu**

**Bước 4:** Cuối cùng chúng ta sử dụng Mask R-CNN với các weight để dự đoán và tạo mask cho hình ảnh



# CHƯƠNG 3 **CODE**

## 3.1 CÁC BƯỚC THỰC HIỆN

### Bước 1: Khai báo thư viện cần thiết

import os

import sys

import random

import math

import numpy as np

import skimage.io

import matplotlib

import matplotlib.pyplot as plt

# Root directory of the project

ROOT\_DIR = os.path.abspath("../")

import warnings

warnings.filterwarnings("ignore")

# Import Mask RCNN

sys.path.append(ROOT\_DIR) # To find local version of the library

from mrcnn import utils

import mrcnn.model as modellib

from mrcnn import visualize

sys.path.append(os.path.join(ROOT\_DIR, "samples/traicay/"))

### Bước 2 : Định nghĩa đường dẫn cho pretrained weight và hình ảnh

# Directory to save logs and trained model

MODEL\_DIR = os.path.join(ROOT\_DIR, "logs")

# Local path to trained weights file

TraiCay\_MODEL\_PATH = os.path.join('', "mask\_rcnn\_traicay\_001.h5")

# Directory of images to run detection on

IMAGE\_DIR = os.path.join(ROOT\_DIR, "images")

class InferenceConfig():

# Set batch size to 1 since we'll be running inference on

# one image at a time. Batch size = GPU\_COUNT \* IMAGES\_PER\_GPU

GPU\_COUNT = 1

IMAGES\_PER\_GPU = 1

config = InferenceConfig()

config.display()

Configurations:

BACKBONE resnet101

BACKBONE\_STRIDES [4, 8, 16, 32, 64]

BATCH\_SIZE 1

BBOX\_STD\_DEV [0.1 0.1 0.2 0.2]

COMPUTE\_BACKBONE\_SHAPE None

DETECTION\_MAX\_INSTANCES 100

DETECTION\_MIN\_CONFIDENCE 0.9

DETECTION\_NMS\_THRESHOLD 0.3

FPN\_CLASSIF\_FC\_LAYERS\_SIZE 1024

GPU\_COUNT 1

GRADIENT\_CLIP\_NORM 5.0

IMAGES\_PER\_GPU 1

IMAGE\_MAX\_DIM 1024

IMAGE\_META\_SIZE 18

IMAGE\_MIN\_DIM 800

IMAGE\_MIN\_SCALE 0

IMAGE\_RESIZE\_MODE square

IMAGE\_SHAPE [1024 1024 3]

LEARNING\_MOMENTUM 0.9

LEARNING\_RATE 0.001

LOSS\_WEIGHTS {'mrcnn\_class\_loss': 1.0, 'rpn\_bbox\_loss': 1.0, 'mrcnn\_mask\_loss': 1.0, 'rpn\_class\_loss': 1.0, 'mrcnn\_bbox\_loss': 1.0}

MASK\_POOL\_SIZE 14

MASK\_SHAPE [28, 28]

MAX\_GT\_INSTANCES 100

MEAN\_PIXEL [123.7 116.8 103.9]

MINI\_MASK\_SHAPE (56, 56)

NAME TraiCay

NUM\_CLASSES 6

POOL\_SIZE 7

POST\_NMS\_ROIS\_INFERENCE 1000

POST\_NMS\_ROIS\_TRAINING 2000

ROI\_POSITIVE\_RATIO 0.33

RPN\_ANCHOR\_RATIOS [0.5, 1, 2]

RPN\_ANCHOR\_SCALES (32, 64, 128, 256, 512)

RPN\_ANCHOR\_STRIDE 1

RPN\_BBOX\_STD\_DEV [0.1 0.1 0.2 0.2]

RPN\_NMS\_THRESHOLD 0.7

RPN\_TRAIN\_ANCHORS\_PER\_IMAGE 256

STEPS\_PER\_EPOCH 515

TOP\_DOWN\_PYRAMID\_SIZE 256

TRAIN\_BN False

TRAIN\_ROIS\_PER\_IMAGE 200

USE\_MINI\_MASK True

USE\_RPN\_ROIS True

VALIDATION\_STEPS 50

WEIGHT\_DECAY 0.0001

### Bước 3: Tạo ra model và load pretrain weight

# Create model object in inference mode.

model = modellib.MaskRCNN(mode="inference", model\_dir='mask\_rcnn\_coco.hy', config=config)

model.load\_weights('mask\_rcnn\_traicay\_001.h5', by\_name=True)

### Bước 4: Định nghĩa các lớp

class\_names = ['Buoi','Cam','Coc', 'Khe', 'Mit']





### Bước 5: Load hình ảnh

imgin = skimage.io.imread(fl)

cv2.namedWindow("ImageIn", cv2.WINDOW\_AUTOSIZE)

imgincv = cv2.cvtColor(imgin, cv2.COLOR\_BGR2RGB)

cv2.imshow("ImageIn", imgincv)

### Bước 6: Predict

r = model.detect([image], verbose=1)[0]

print(r['masks'])

print(r['rois'])

print(r['scores'])

print(r['class\_ids'])

mask = r['masks']

L = len(r['rois'])

for i in range(0, L):

y1 = r['rois'][i][0]

x1 = r['rois'][i][1]

y2 = r['rois'][i][2]

x2 = r['rois'][i][3]

cv2.rectangle(imagecv,(x1,y1), (x2,y2), (0,0,255), 2)

cv2.putText(imagecv,class\_names[r['class\_ids'][i]],(x1+5, y1+20),cv2.FONT\_HERSHEY\_SIMPLEX,0.7,(0,0,255),2)

#visMask = (mask \* 255).astype("uint8")

#contours, \_ = cv2.findContours(visMask, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

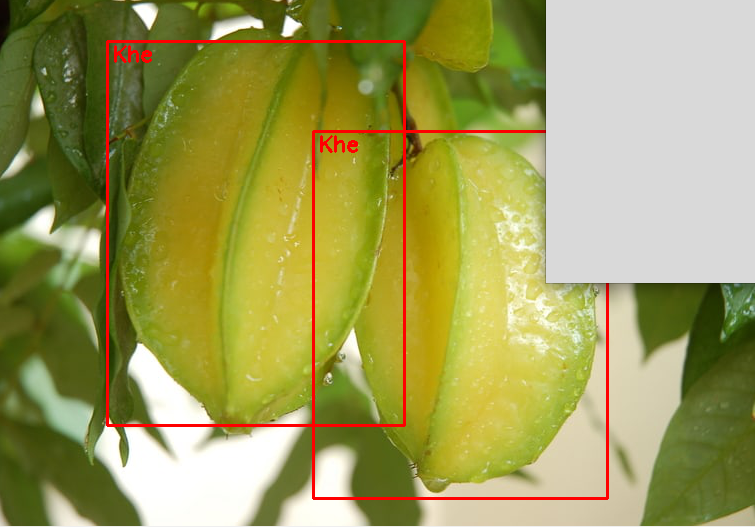
#for i in range(len(contours)):

# cv2.drawContours(imagecv, contours, i, (0,255,0), 2)

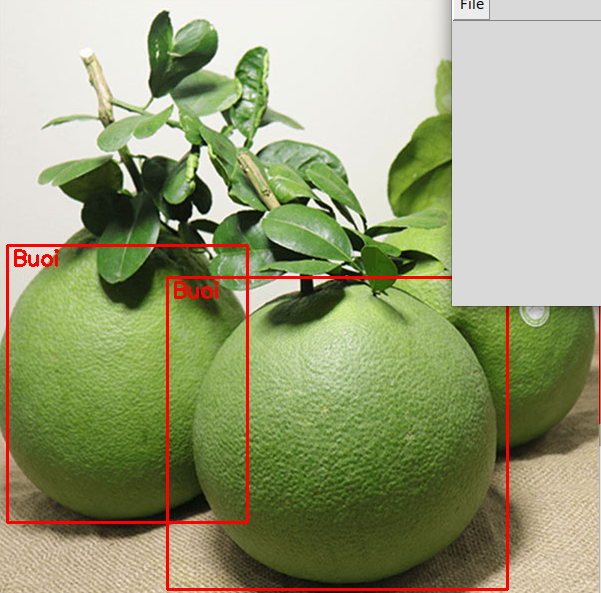
cv2.imshow('ImageIn',imagecv)

## 3.2 KẾT QUẢ

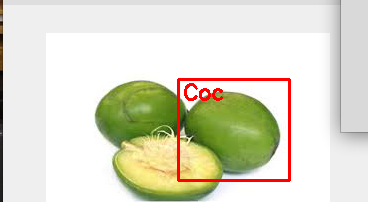
**1.Trái Khế**



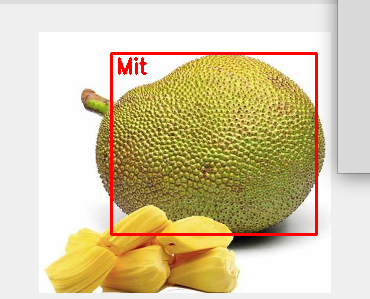
**2.Trái Bưởi**



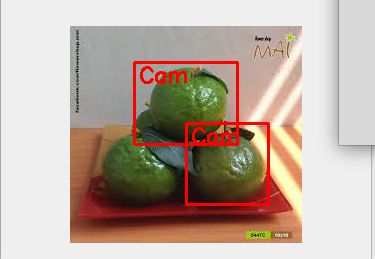
**3.Trái Cóc**



**4.Trái mít**



**5.Trái Cam**



# THAM KHẢO

<https://viblo.asia/p/maskrcnn-cac-buoc-trien-khai-mask-r-cnn-cho-bai-toan-image-segmentation-6J3Zg4VPlmB>

<https://www.analyticsvidhya.com/blog/2019/07/computer-vision-implementing-mask-r-cnn-image-segmentation/>

<https://www.nature.com/articles/s41438-020-0323-3>

<https://datahack.analyticsvidhya.com/contest/data-science-blogathon-5/?utm_source=blog&utm_medium=popup>