# **CHƯƠNG 3. ỨNG DỤNG THỰC NGHIỆM**

## **3.1.Đặc tả bài toán**

Bài toán: Thu thập dữ liệu ảnh Lidar, lấy nhãn dữ liệu, qua mô hình Mask R-CNN nhận dạng tòa nhà từ trong ảnh Lidar.

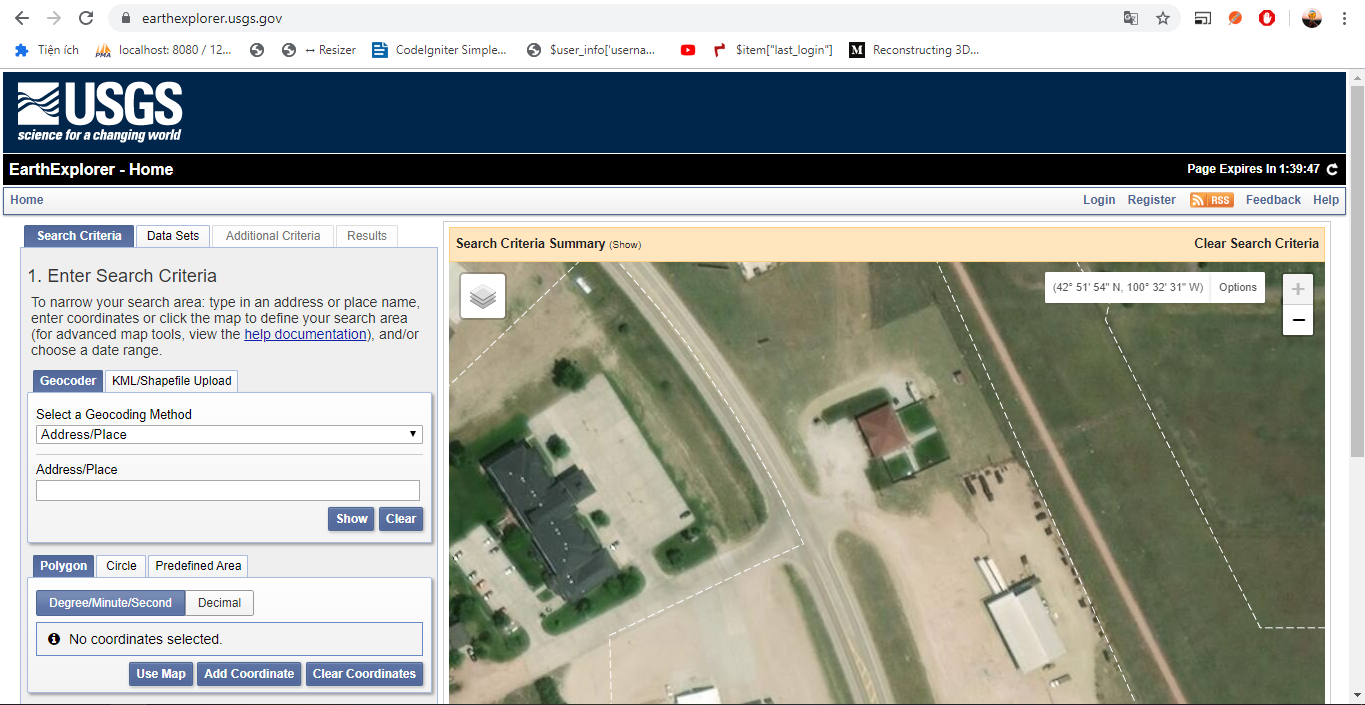
Vấn đề bài toán cần giải quyết: Trong khuôn khổ của ĐATN, em hướng đến giải quyết bài toán phát hiện và phân vùng đối tượng ngôi nhà.

Hướng giải quyết: Như đã trình bày ở trên mô hình đề xuất của em là sử dụng mạng Mask R-CNN với kiến trúc backbone là FPN để phát hiện tòa nhà trên ảnh Lidar.

## **3.2.Quy trình thực hiện**

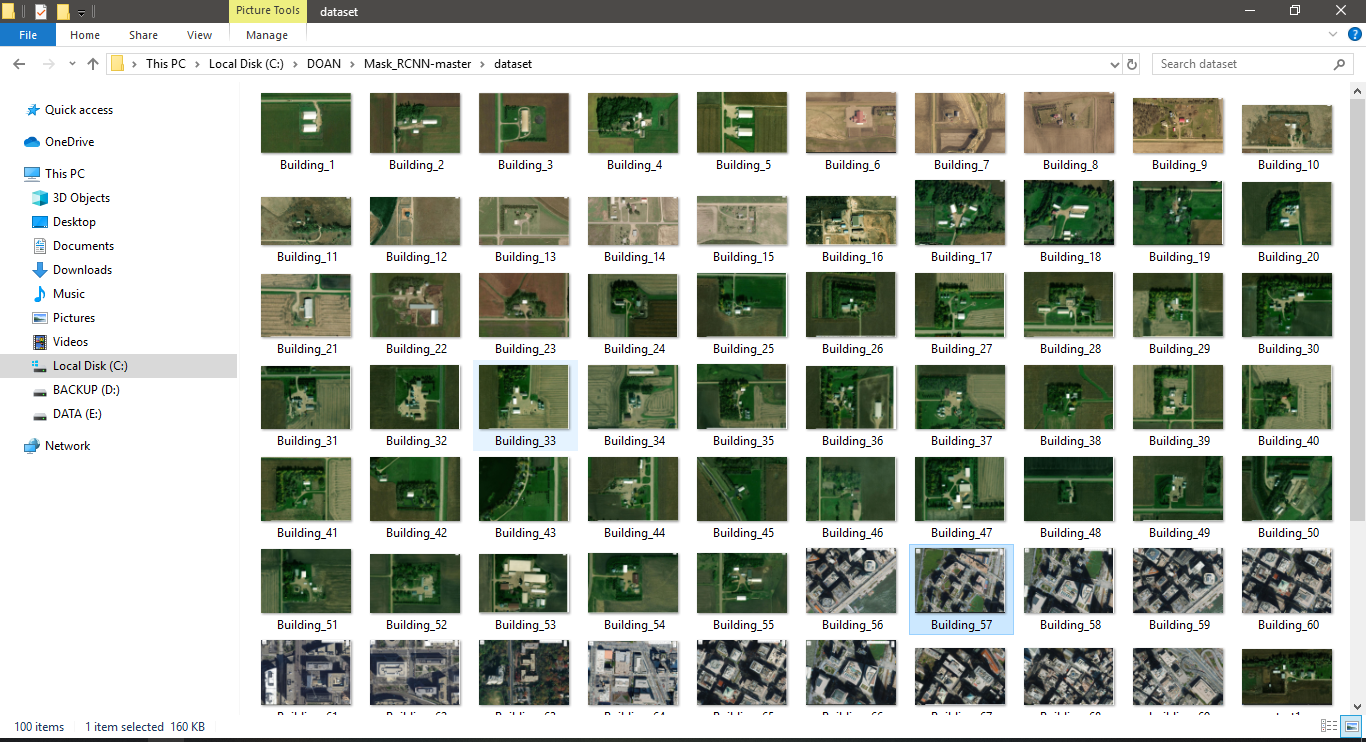
### **3.2.1. Thu thập dữ liệu**

Sau khi tìm hiểu về ứng dụng của công nghệ Lidar ở chương 1, có rất nhiều trang có thể thu thập dữ liệu về ảnh Lidar em đã sử dụng trang <https://earthexplorer.usgs.gov/>. Bộ dữ liệu là kiểu dáng tòa nhà nước Mỹ và lý do em chọn nước Mỹ vì ảnh Lidar từ nước Mỹ có chất lượng và độ phân giải rất tốt (trang <https://earthexplorer.usgs.gov/> được public lên chủ yếu để người Mỹ sử dụng).



#### Hình 3.1 giao diện trang https://earthexplorer.usgs.gov

Em đã chụp màn hình và dùng ứng dụng Snipping Tool của Microsoft để lấy các hình ảnh có đối tượng ngồi nhà. Với số lượng là **100** ảnh, với kích thước **862x562** pixel, định dạng ảnh là **JPEG**. Số lượng dữ liệu ảnh của em là nhỏ nhưng vì không cần hàng nghìn hình ảnh để đào tạo một mô hình học sâu, thay vì đào tạo một mô hình từ đầu, em đào tạo bộ dữ liệu của mình dựa trên phương pháp transfer learning. Em đào tạo bộ dữ liệu của mình với các trọng số đã được đào tạo trên bộ dữ liệu COCO, mặc dù bộ dữ liệu COCO không chứa lớp ngồi nhà, nhưng nó chứa rất nhiều hình ảnh khác (~120K), vì vậy các trọng số đã học được rất nhiều tính năng phổ biến trong hình ảnh tự nhiên, rất hữu ích.



#### Hình 3.2 Dữ liệu ảnh đã thu thập

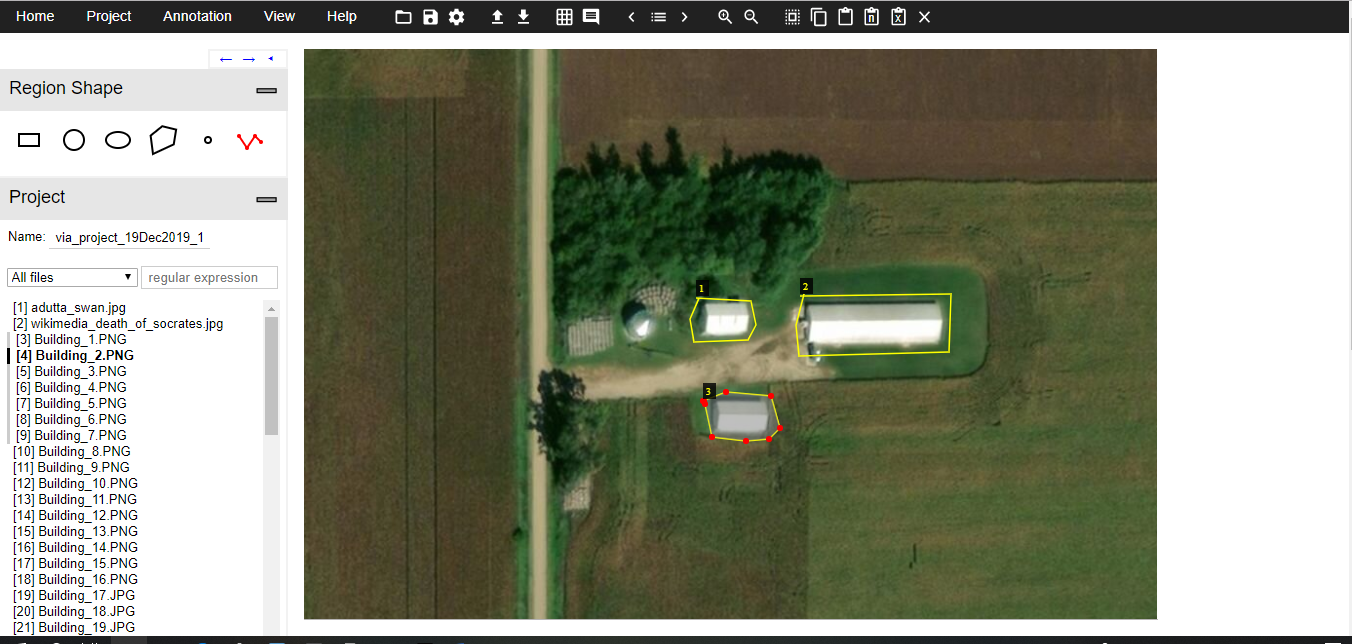
### **3.2.2. Gán nhãn dữ liệu**

Có rất nhiều công cụ để lấy nhãn hình ảnh. Em đã tìm hiểu một số công cụ lấy nhãn dưới đây:

* LabelMe: Một trong những công cụ được biết đến nhiều nhất. Mặc dù vậy, giao diện người dùng hơi chậm đặc biệt là khi chúng ta phóng to hình ảnh lớn, link cài đặt và sử dụng: <http://labelme.csail.mit.edu/Release3.0/>
* RectLabel: Đơn giản và dễ làm việc ( Chỉ sử dụng được cho Mac), link cài đặt và sử dụng: <https://rectlabel.com/>
* LabelBox: Sử dụng khá tốt cho các dự án ghi nhãn lớn hơn và có các tùy chọn cho các loại tác vụ ghi nhãn khác nhau, link cài đặt và sử dụng: <https://labelbox.com/>
* VGG Image Annotator (VIA): Nhanh, nhẹ và được thiết kế thực sự tốt, em đã lựa chọn sử dụng công cụ này
* UI COCO: Công cụ được sử dụng để chú thích bộ dữ liệu COCO, link cài đặt và sử dụng: <https://github.com/tylin/coco-ui>

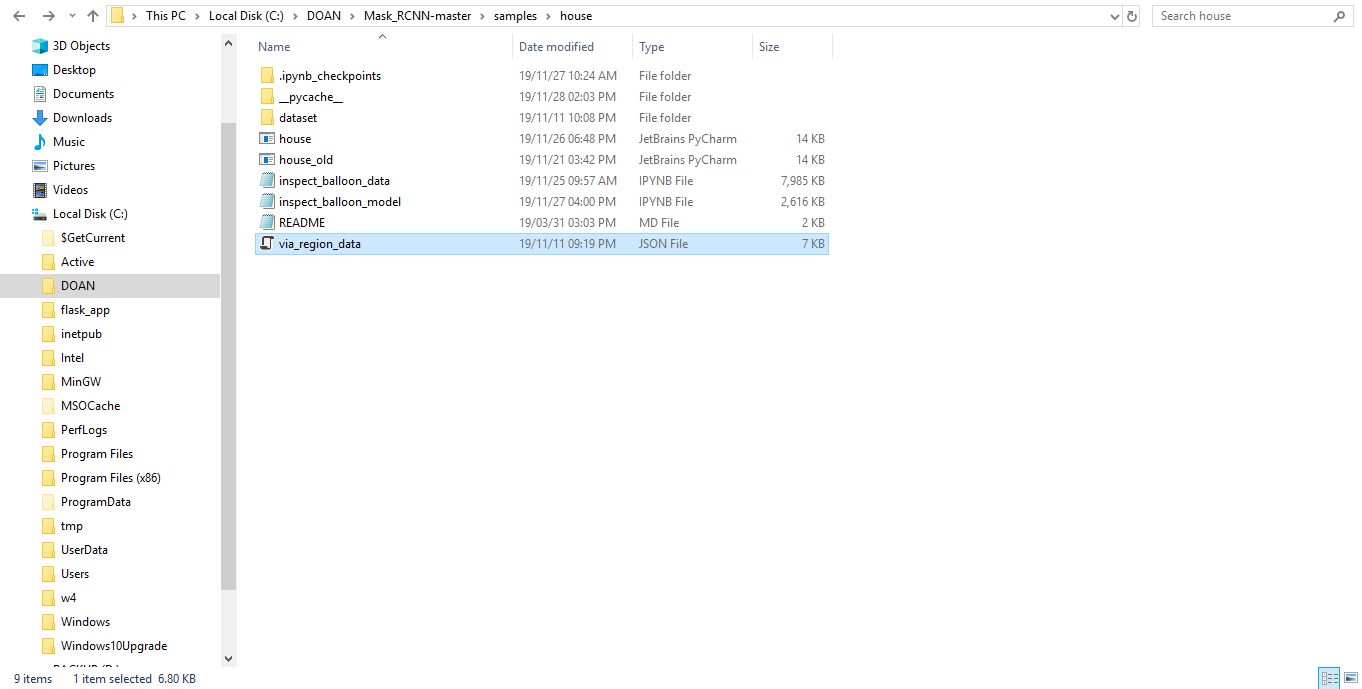
Em đã lựa chọn sử dụng công cụ VIA (VGG Image Annotator) vì tính đơn giản và dễ sử dụng của nó. VGG Image Annotator là môt một phần mềm chủ thích thủ công đơn giản và độc lập cho hình ảnh, âm thanh và video. VIA chạy trong trình duyệt web và không yêu cầu cài đặt hay sử dụng và phần mềm này hoàn toán với một trang HTML độc lập có kích thước nhỏ hơn 400 Kilobyte chạy như một ứng dụng ngoại tuyến trong hầu hết các trang web hiện đại. Em đã sử dụng công cụ VIA tại:

<http://www.robots.ox.ac.uk/~vgg/software/via/via_demo.html>



#### Hình 3.3 Lấy nhãn hình ảnh bằng công cụ VGG

Sau khi lấy nhãn được tất cả các hình ảnh trong bộ dữ liệu của mình em export các ghi chú nhãn ra 1 file json.



Công cụ lưu ghi chú nhãn hình ảnh dưới dạng json. Thuộc tính ảnh gồm có : Tên ảnh (filename), thuộc tính hình dạng ảnh (shape\_attributes), kích thước ảnh (size).

# { 'filename': house1.jpg',

# 'regions': {

# '0': {

# 'region\_attributes': {},

# 'shape\_attributes': {

# 'all\_points\_x': [...],

# 'all\_points\_y': [...],

# 'name': 'polygon'}},

# ... more regions ...

# },

# 'size': 100202

# }

### **3.2.3. Huấn luyện Mask R-CNN**

Sau khi đã có dữ liệu, em đã lưu các hình ảnh vào thư mục dataset gồm thư mục train và validation. Chúng ta sẽ đào tạo dữ liệu dựa trên phương pháp transfer learning và dựa trên bộ dữ liệu COCO với các trọng số đã được đào tạo trước đó. Mặc dù bộ dữ liệu COCO không không chứa lớp ngôi nhà những bộ dữ liệu này chứa rất nhiều hình ảnh khác (~120K), vì vậy các trọng số được đào tạo đã học được rất nhiều tính năng phổ biến trong hình ảnh tự nhiên, thực sự rất hiểu ích. Và thứ hai với trường hợp sử dụng với đồ án lần này em không đòi hỏi độ chính xác rất cao từ mô hình này vì vậy bộ dữ liệu nhỏ sẽ đủ.

Tiếp theo em tiến hành huấn luyện từ trọng số COCO, vì Mask R-CNN là một mô hình khá lớn đặc biệt là việc triển khai mô hình sử dụng ResNet101 và FPN nên mô hình này đòi hỏi một GPU hiện đại. Nhưng vì điều kiện không cho phép em đã sử dụng CPU với laptop của mình.

Em đã cấu hình lại file huấn luyện với các cấu hình cơ bản sau:

* Số bước đào tạo trên 1 epoch: 100
* Bỏ qua các phát hiện với độ tin cậy < 90%

# Give the configuration a recognizable name

NAME = "house"

# We use a GPU with 12GB memory, which can fit two images.

# Adjust down if you use a smaller GPU.

IMAGES\_PER\_GPU = 1

# Number of classes (including background)

NUM\_CLASSES = 1 + 2 # Background + house + building

# Number of training steps per epoch

STEPS\_PER\_EPOCH = 100

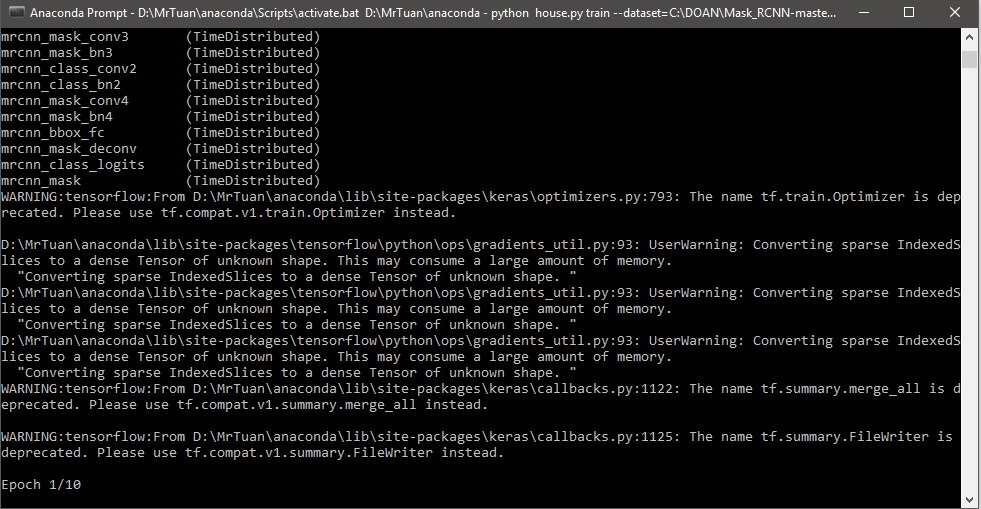
# Skip detections with < 90% confidence

DETECTION\_MIN\_CONFIDENCE = 0.9

Cấu hình đào tạo bộ dữ liệu:

* Số lớp : house và building

Quá trình huấn luyện dữ liệu qua 10 epoch với số bước đào tạo trên 1 epoch là 100:



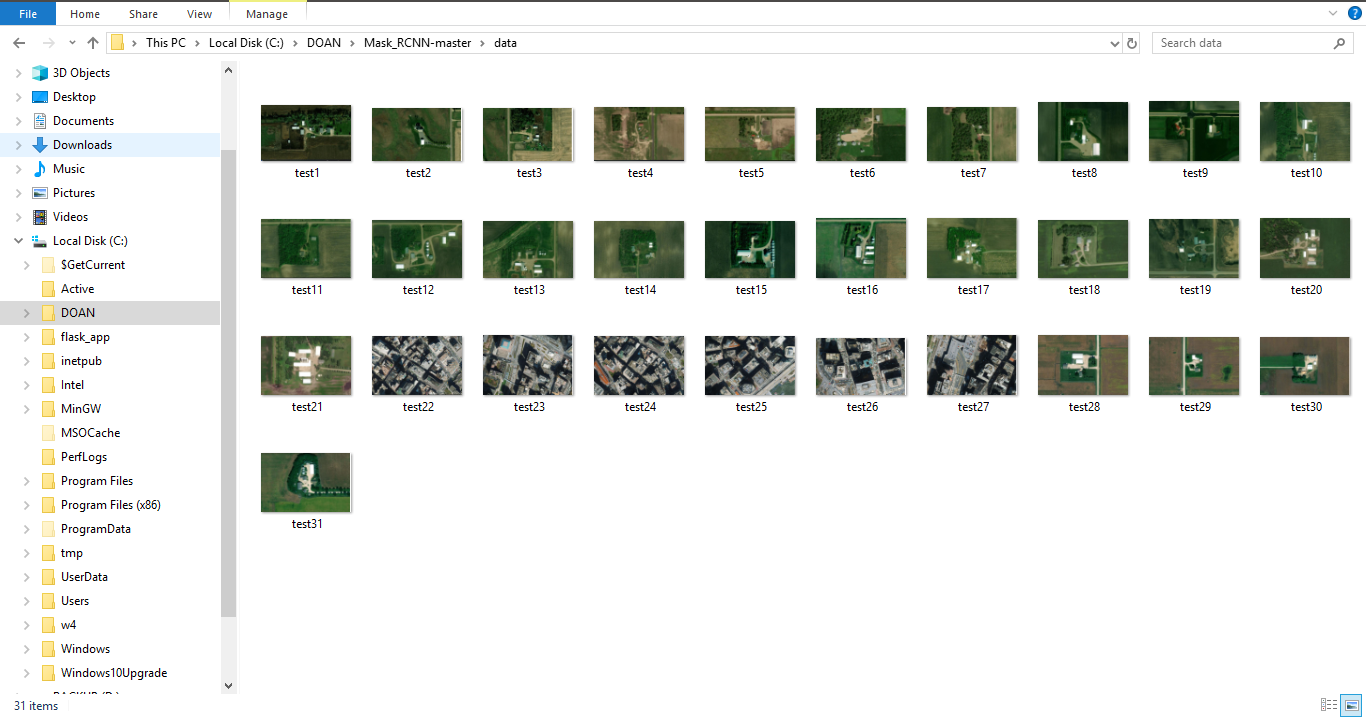
#### Hình 3.4 Quá trình huấn luyện dữ liệu

Cuối cùng sau khi kết thúc quá trình huấn luyện em lưu lại kiến trúc model và các trọng số đã được đào tạo với nhau thành một tệp h5. HDF5 (.h5, hdf5) là định dạng tệp phù hợp để lưu trữ các bộ sưu tập lớn các mảng số đa chiều (ví dụ: mô hình, tệp dữ liệu). HDF là viết tắt của Hierarchcal Data Format (định dạng dữ liệu phân cấp).

Kết quả thời gian em huấn luyện cho bộ dữ liệu của mình là gần 12 tiếng với 10 Epoch. Thời gian huấn luyện bộ dữ liệu sẽ nhanh hơn rất nhiều nếu có những bộ PC chuyên dụng phục vụ cho nhu cầu học sâu với CPU đủ mạnh và đặc biệt là sự hỗ trợ của GPU.

### **3.2.4. Chạy thực nghiệm chương trình**

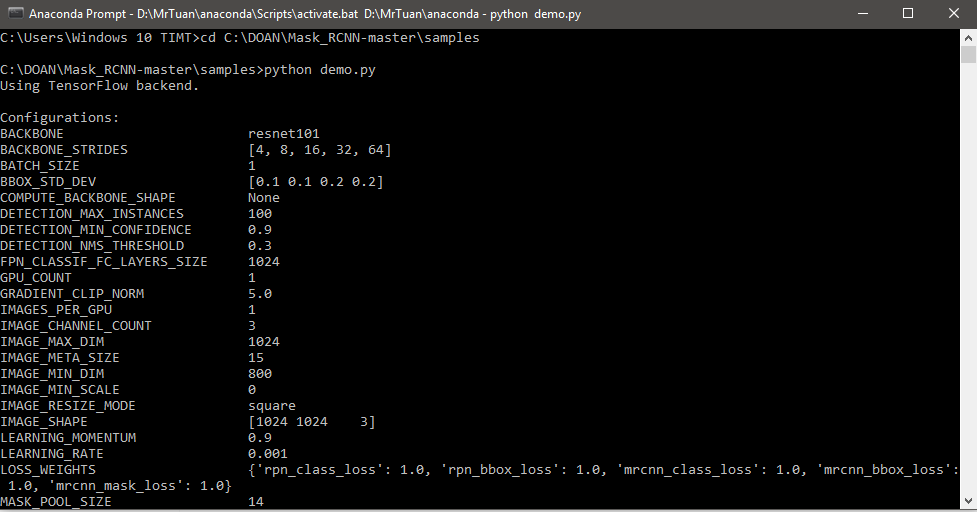
Sau khi huấn luyện xong bộ dữ liệu của mình em đã cấu hình lại file test của bộ dữ liệu COCO với 2 nhãn em đã lấy và ghi lại trước đó là house và building. Tiến hành chạy demo trên ảnh đầu vào là các hình ảnh ngôi nhà. Với 100 ảnh đã thu thập được, em sử dụng 70% để xây dựng mô hình và 30% để tiến hành kiểm thử mô hình:



#### Hình 3.5 Dữ liệu đầu vào

Dữ liệu đầu vào có 31 ảnh có kich thước 862x562 pixel, định dạng ảnh JPEG.

Tiến hành chạy demo chương trình đầu vào là ngẫu nhiên 1 hình ảnh trong thư mục dataset:



#### Hình 3.6 Chạy demo chương trình

Các cấu hình (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\_CHANNEL\_COUNT 3

IMAGE\_MAX\_DIM 1024

IMAGE\_META\_SIZE 15

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 {'rpn\_class\_loss': 1.0, 'rpn\_bbox\_loss': 1.0, 'mrcnn\_class\_loss': 1.0, 'mrcnn\_bbox\_loss': 1.0, 'mrcnn\_mask\_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 house

NUM\_CLASSES 3

POOL\_SIZE 7

POST\_NMS\_ROIS\_INFERENCE 1000

POST\_NMS\_ROIS\_TRAINING 2000

PRE\_NMS\_LIMIT 6000

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 100

TOP\_DOWN\_PYRAMID\_SIZE 256

TRAIN\_BN False

TRAIN\_ROIS\_PER\_IMAGE 200

USE\_MINI\_MASK True

USE\_RPN\_ROIS True

VALIDATION\_STEPS 50

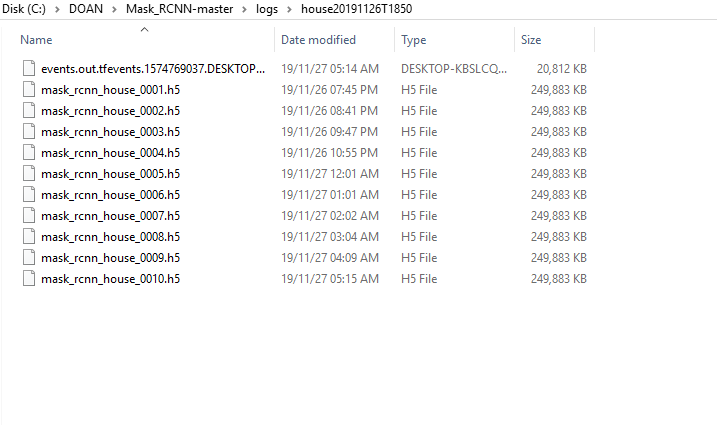
WEIGHT\_DECAY 0.0001

Giải thích các cơ thông số cơ bản:

* BACKBONE: resnet101, Mask R-CNN sử dụng kiến trúc resnet101 để trích xuất các đặc trưng từ ảnh.
* DETECTION\_MIN\_CONFIDENCE: 0.9, bỏ qua các phát hiện với độ tin cậy < 90%.
* IMAGES\_PER\_GPU: 1, Số hình ảnh xử trên 1 giây.
* MASK\_SHAPE: [28, 28], Mask R-CNN tạo mặt nạ 28x28 và thay đổi kích thước của chúng theo kích thước RoI.
* STEPS\_PER\_EPOCH: 100, số bước huấn luyện trên 1 epoch.
* POOL\_SIZE: 7, kích thước của các cửa số gộp tối đa.
* Kết quả sau khi chạy demo là ảnh đầu ra sẽ được lưu vào thư mục samples với tên output.JPG. Đây là kết quả quả khi em chạy demo chương trình:



#### Hình 3.7 Kết quả demo chương trình



#### Hình 3.8 Kết quả sau khi huấn luyện dữ liệu

Cấu trúc của một tệp h5 gồm 2 phần: tiêu đề và mảng dữ liệu. Tiêu đề chứa thông tin cần thiết để diễn giải phần mảng của tệp dữ liệu, thông tin tiêu đề bao gồm tên của đối tượng, chiều của nó, loại của nó, loại số của nó, thông tin về cách dữ liệu được lưu trữ trên đĩa và thông tin khác được thư viện sử dụng để tăng tốc truy cập và tập dữ liệu hoặc duy trì tính toàn vẹn của tệp

## **3.3.Cài đặt chương trình**

### **3.3.1. Môi trường cài đặt thực nghiệm**

Cấu hình máy tính em sử dụng để cái đặt và chạy thực nghiệm:

* Intel® core™ i5-6200u CPU @ 2.3GHz (4 CPUs), ~2.4Ghz.
* Hệ điều hành widow 10 Pro 64-bit (10.0, Build 17134).

Ngôn ngữ lập trình được sử dụng cho việc thao tác là Python, phiên bản python em đã sử dụng trong ĐATN đợt này là python 3.7, python là một ngôn ngữ lập trình scripting phổ biến và rất thú vị. Hơn nữa nó rất hiệu quả trong việc sử lý dữ liệu với R, dễ học, dễ làm và cài đặt. Python đang là ngôn ngữ phổ biến ở thời điểm hiện đại**.**

Môi trường lập trình là python vì vậy cần cài đặt python và các package cần thiết là: tensorflow, keras, opencv2, numpy.

### **3.3.2. Mã nguồn**

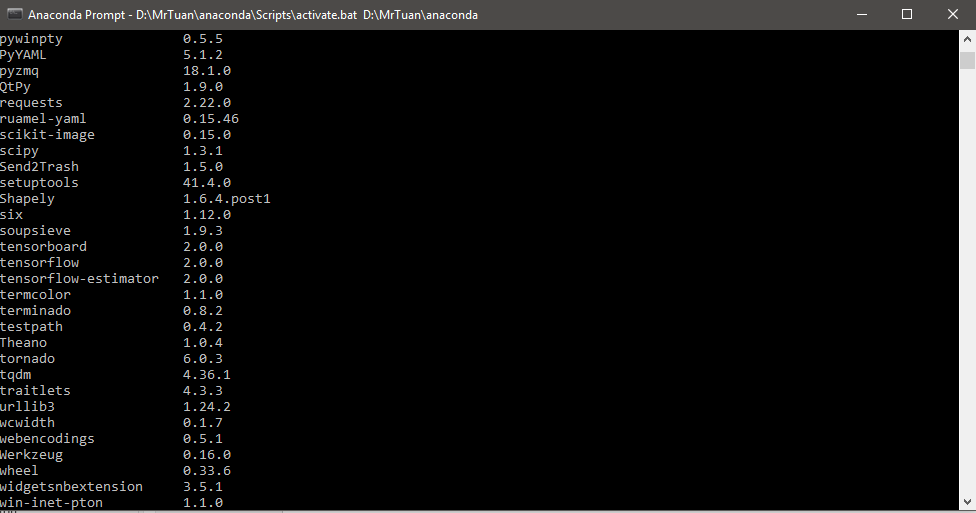
Để sử dụng Mask R-CNN em có 2 lựa chọn framework là Tensorflow và Caffe tại: <https://caffe.berkeleyvision.org/> và <https://www.tensorflow.org/>. Soure code của Mask R-CNN được public tại: <https://github.com/matterport/Mask_RCNN> hoặc với Caffe tại: <https://github.com/BVLC/caffe>. Trong đồ án em sử dụng framework Tensorflow, các bước cài đặt framework Tensorflow như sau:

Bước 1: Em tải Framework Mask R-CNN được public tại : \*\* <https://github.com/matterport/Mask_RCNN> \*\*

Bước 2: Cài đặt những thư viện cần thiết cho Mask R-CNN như: numpy, Scipy, Pillow, Cython, matplotlib, image, tensorflow>=1.3.0, keras>=2.0.8, openCV, imgaug, Ipythpn. Đây là các thư viện em đã cài đặt cho quá trình thử nghiệm này:



#### Hình 3.9 Các thư viện đã cài



#### Hình 3.10 Các thư viện đã cài

Bước 3: Tải bộ dữ liệu COCO đã được đào tạo với bộ trọng số tại : \*\* <https://github.com/cocodataset/cocoapi> \*\*.

## **3.4.Kiểm thử và kết quả**

Một số hình ảnh đầu vào và kết quả khác khi chạy chương trình :

* Chạy demo ảnh test1.JPG và kết quả:





* Ảnh test2.JPG và kết quả:





* Ảnh test3.JPG và kết quả:





* Chạy demo chương trình, đưa lần lượt 31 ảnh đầu vào để kiểm tra các kết quả nhận dạng tòa nhà, các kết quả thu được như sau:

+) Tỉ lệ nhận dạng đúng đối tượng ngôi nhà trong ảnh : 29/31 ~ 94%



#### Hình 3.11 Một số hình ảnh không phát hiện được đối tượng ngôi nhà



#### Hình 3.12 Một số hình ảnh không phát hiện được đối tượng ngôi nhà

+) Tỉ lệ nhận nhận dạng và phân vùng đầy đủ đối tượng ngôi nhà trong ảnh : 27/31 ~ 87%



#### Hình 3.13 Hình ảnh không phân vùng ngồi nhà đầy đủ do bị dính các vật bên ngoài như: cây

Nhận xét kết quả: Từ các kết quả trên cho chúng ta thấy việc sử dụng Mask R-CNN cho kết quả khá tốt khi thực hiện nhận dạng hình ảnh tòa nhà, kết quả nhận dạng và phân vùng với độ chính xác khá cao. Tuy nhiên thời gian chạy chương trình còn khá lâu và tốn nhiều tài nguyên của máy chạy chương trình.

# **KẾT LUẬN**

**Kết quả đạt được**

Trong quá trình thực hiện đồ án tốt nghiệp với đề tài “ Nghiên cứu kỹ thuật deep learning và nhận dạng tòa nhà trong ảnh Lidar “ em đã thu được một số kết quả sau:

* Tìm hiểu tổng quan về bài toán nhận dạng hình ảnh.
* Tìm hiểu và làm chủ được một số kỹ thuật trong lĩnh vực học máy, cụ thể là mạng nơ ron tích chập.
* Hiểu về quy trình triển khai mạng nơ ron trên máy chủ
* Học thêm được ngôn ngữ mới là python và sử dụng để thao tác với dữ liệu và các thư viện như opencv, numpy…

**Những điểm còn hạn chế**

* Thời gian training dữ liệu với mô hình Mask R-CNN còn dài, thời gian chạy chương trình chưa đến thời gian thực.
* Một số lần nhận dạng bị lỗi vì kích thước ảnh quá lớn

Mặc dù đã rất cố gắng trong quá trình thực hiện đồ án tốt nghiệp này, nhưng do năng lực , trình độ và thời gian hạn hẹp nên khó tránh khỏi những sai sót. Kính mong thầy cô và các bạn quan tâm chỉ bảo để ứng dụng của em có thể hoàn thiện hơn.

**Hướng phát triển**

Như em đã đã đề cập ở chương 1 bài toán nhận dạng tòa nhà qua ảnh Lidar là một bài toán rất hay có tính ứng dụng rất cao đặc biệt là thành lập bản đồ hộ trỡ cho các ngành như kiến trúc, hộ trợ công tác thiết kế, quy hoạch và phát triển độ thị, du lịch, quân sự …

Trong thời gian tiếp theo khi kết thúc đồ án tốt nghiệp này em sẽ vẫn còn tiếp tục nghiên cứu sâu hơn về Mask R-CNN và những ứng dụng con của Mask R-CNN để có thể làm ra được những sản phẩm hữu ích cho cuộc sống thực tiễn. Em xin cảm ơn, thầy cô và các bạn đã chú ý lặng nghe!

# **TÀI LIỆU THAM KHẢO**

**1. Tài Liệu tiếng việt.**

[1] Trần Đình Trí, Công nghệ Lidar, Đại Học Mỏ - Địa chất, 2013.

[2] <https://nhdp.net/blog/2018/11/tong-quan-don-gian-ve-mang-no-ron-tich-chap-convolutional-neural-networks/>.

[3] Lê Thị Thu Hằng, Nghiên cứu về mạng neural tích chập và ứng dụng cho bài toán nhận dạng biển số xe, Đại học Quốc Gia Hà Nội, 2016.

[4] Đặng Thanh Tùng, Nghiên cứu ứng dụng công nghệ Lidar thành lập bản đồ 3D khu vực đô thị, Đại Học Quốc Gia Hà Nội , 2011.

[5] Nguyễn Nhật Quang, Bài giảng môn Trí tuệ nhân tạo, Đại học Bách Khoa Hà Nội 2019.

**2. Tài Liệu tiếng anh.**

[6] John McCarthy, Artificial Intelligence and Mathematical Theory of Computation, Ablex Pub, 1990.

[7] He K., Gkioxari G., Dollár P., et al. (2017). Mask R-CNN. ArXiv170306870 Cs.

[8] He K., Zhang X., Ren S., et al. (2014). Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. ArXiv14064729 Cs, 8691, 346–361.