## ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG DỮ LIỆU





ĐỒ ÁN MÔN HỌC: HỆ THỐNG NHÚNG MẠNG KHÔNG DÂY

ĐỀ TÀI: XÂY DỰNG ỨNG DỤNG TRÊN THIẾT BỊ JETSON NANO

*Giảng viên hướng dẫn:*

***Ths. Đặng Lê Bảo Chương***

*Lớp: NT131.L12.MMCL*

*Sinh viên thực hiện:*

*Vũ Hà Anh 17520258*

*Phạm Thùy Nhung 17520860*

*Phan Nhật Thịnh 17521090*

*Nguyễn Phương Thùy 18521476*

# NHẬN XÉT CỦA GIẢNG VIÊN

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

# MỤC LỤC

[CHƯƠNG 1: GIỚI THIỆU](#_bookmark2) 5

* 1. [Yolo là gì?](#_bookmark3) 5
  2. [Nguyên lý hoạt động của Yolo](#_bookmark4) 5
     1. [Nhận dạng](#_bookmark5) 5
     2. [Huấn luyện 7](#_bookmark6)
  3. YOLOv4 7
     1. Nhận dạng YOLOv4 7
     2. So sánh YOLOv4 với các model khác 8

1.4 JETSON NANO 8

[CHƯƠNG 2: CÁCH CÀI ĐẶT JETSON NANO và YOLO](#_bookmark15) 9

**2.1 Cách cài đặt 9**

2.2 Cách xây dựng model YOLOv4 10

[TÀI LIỆU THAM KHẢO](#_bookmark94) 15

# PHỤ LỤC HÌNH ẢNH

[Hình 1. Mô hình YOLO](#_bookmark17) 5

[Hình](#_bookmark24) [2. Cấu trúc của một model YOLO](#_bookmark24) 5

[Hình 3. Yolo phân chia hình ảnh thành một mạng lưới 7x7](#_bookmark29) 6

Hình 4: Bản đồ tính năng đầu ra của YOLO 6

Hình 5: So sánh YOLOv4 và các công cụ phát hiện đối tượng hiện đại khác 8

Hình 6: Xác định vị trí của file IMG trong máy tính 9

Hình 7: Chọn thiết bị để boot IMG. 9

Hình 8: IMG sẽ tự động được boot. 10

Hình 9: Data để train và valid 11

Hình 10: File yolo.names 12

Hình 11: File yolo.data 12

Hình 12: Cái thông số để cấu hình. 13

Hình 13: Real-time Object Detection. 14

Hình 14: Image Object Detection. 14

# PHỤ LỤC BẢNG

Bảng 1: So sánh Jetson nano với một số Raspberry pi 8

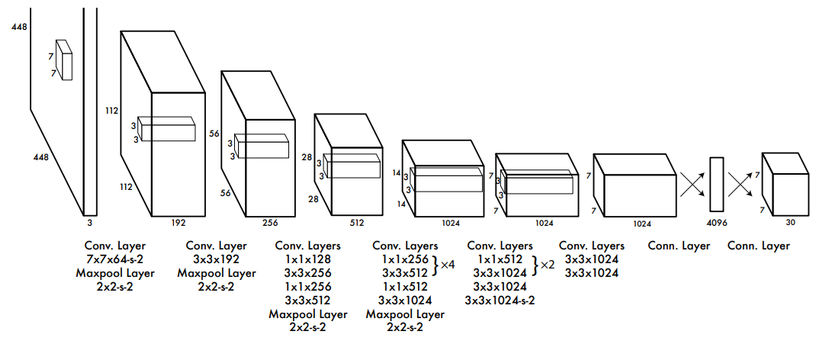
## CHƯƠNG 1: GIỚI THIỆU

###### **Yolo là gì?**

Object Detection là một bài toán quan trọng trong lĩnh vực Computer Vision, thuật toán Object Detection được chia thành 2 nhóm chính:

* Họ các mô hình RCNN ( Region-Based Convolutional Neural Networks) để giải quyết các bài toán về định vị và nhận diện vật thể.
* Họ các mô hình về YOLO (You Only Look Once) dùng để nhận dạng đối tượng được thiết kế để nhận diện các vật thể real-time.

Yolo là một mô hình mạng CNN cho việc phát hiện, nhận dạng, phân loại đối tượng. Yolo được tạo ra từ việc kết hợp giữa các convolutional layers và connected layers.Trong đóp các convolutional layers sẽ trích xuất ra các feature của ảnh, còn full-connected layers sẽ dự đoán ra xác suất đó và tọa độ của đối tượng. [1]

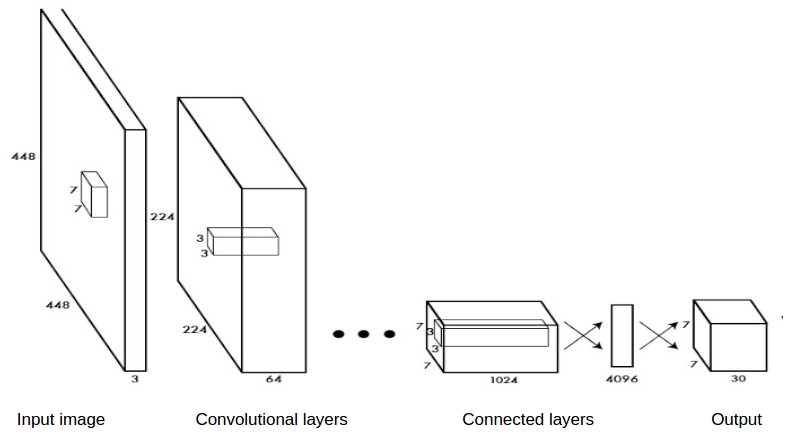


Hình 1: Mô hình YOLO

###### **Nguyên lý hoạt động của Yolo.**

1.2.1 Nhận dạng:

YOLO là một deep net kết hợp giữa convolutional layers và connected layers :



Hình 2: Cấu trúc của một model YOLO

YOLO phân chia hình ảnh thành một mạng lưới 7x7 ô (grid\_size=7x7):

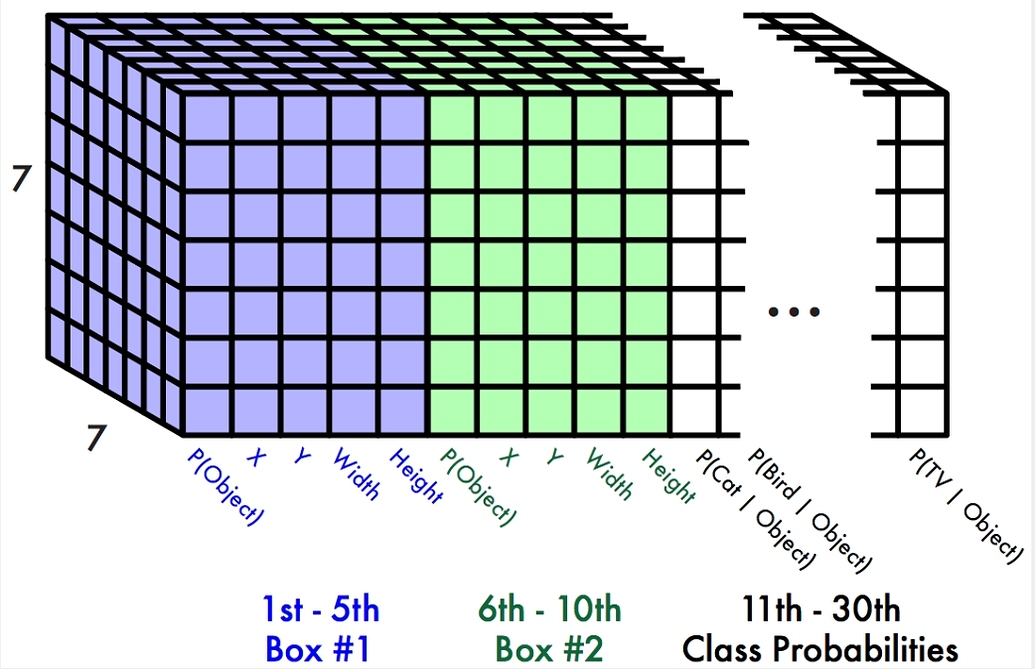


Hình 3: Yolo phân chia hình ảnh thành một mạng lưới 7x7

YOLO sẽ dự đoán xem trong mỗi ô xem liệu có object mà điểm trung tâm rơi vào ô đó. Và dự đoán điểm trung tâm, kích thước của object đó và xác xuất object đó là object nào trong các objects cần xác định.

Mỗi ô này có trách nhiệm dự đoán 2 hộp (boxes\_number=2) bao quanh. Mỗi 1 hộp mô tả hình chữ nhật bao quanh một object.

Giả sử ta đang huấn luyên YOLO nhận dạng 20\_objects khác nhau. Sau khi qua các layers, image input sẽ được biến đổi thành 1 tensor kích thước 7x7x30.

Hình 4: Bản đồ tính năng đầu ra của YOLO

Ta có thể hiểu là mỗi ô sẽ có 30 tham số , tham số thứ nhất là xác xuất ô có chứa 1 object, tham số 2,3,4,5 lấn lượt là x\_center, y\_center, width, height của box 1. Tương tự tham số 6,7,8,9,10 là của box 2. Tham số thứ 11 là xác xuất object trong ô là object1 (trong 20\_objects cần nhận dạng). Tương tự tham số 12 là xác xuất object trong ô là object2 … cho đên tham số 30 là xác xuất object trong ô là object20.

Ta có thể hiểu đơn giản như sau:

1. Image input được resize thành 1 image 448x448x3 (image\_dimension = 448x448 với số channels = 3, YOLO sử dụng hệ màu HSV)
2. Qua các layers, biến đổi image 448x448x3 thành 1 grid có kích thước 7x7 với số tham số cho mỗi ô trong grid là 30 (30 = 5xboxes\_number + number\_of\_objects)
3. Neural net có nhiệm vụ huấn luyện các **trọng số của các layers** để có được mô hình tốt cuối cùng.
4. Để nhận dạng một image mới, các bước 1,2 sẽ được thực hiện. Sau đó dựa vào các **tham số trong grid 7x7x30** ta sẽ xác định được các box chứa object với xác xuất cao. (các box đè lên nhau sẽ được loại bằng phương pháp NMS, chỉ giữ lại box có xác xuất cao nhất)

1.2.2. Huấn luyện.

Neural net sẽ tính toán từng ảnh (có thể lặp lại 1 ảnh) để tối ưu hàm mất mát. Việc tối ưu này sẽ giúp neural net tìm ra 1 bộ trọng số tốt nhất để biểu diễn dữ liệu của bạn và giúp nhận dạng các ảnh mới.

Hiểu hàm mất mát giúp bạn hiểu cơ chế hoạt động của từng neural net và cách huấn luyện chúng và giúp bạn dễ dàng hiểu code source, và trải nghiệm với nó. [2]

###### **YOLOv4**

1.3.1 Nhận dạng YOLOv4

YOLOv4: Optimal Speed and Accuracy of Object Detection - YOLOv4 vượt trội hơn hẳn các phương pháp hiện có về cả “hiệu suất phát hiện” và “tốc độ vượt trội”. Tham khảo một bài báo được xuất bản, nhóm nghiên cứu đề cập đến nó như một máy dò vật thể “vận hành nhanh chóng” có thể được huấn luyện trơn tru và được sử dụng trong các hệ thống sản xuất. Mục tiêu chính là “tối ưu hóa bộ dò mạng nơ-ron cho các tính toán song song”, nhóm cũng giới thiệu nhiều kiến ​​trúc khác nhau và các lựa chọn kiến ​​trúc khác nhau sau khi phân tích kỹ lưỡng ảnh hưởng đến hiệu suất của nhiều bộ dò, các tính năng được đề xuất trong các mô hình YOLO trước đó.

Ba tác giả “Alexey Bochkovskiy, nhà phát triển người Nga, người đã xây dựng phiên bản Windows YOLO”, “Chien-Yao Wang” và “Hong-Yuan Mark Liao”, được đưa vào tác phẩm này và toàn bộ mã hiện có tại Github.

YOLOv4 bao gồm;

**Backbone**: CSPDarknet53

**Neck**: Spatial Pyramid Pooling module bổ sung, tập hợp đường dẫn PANet

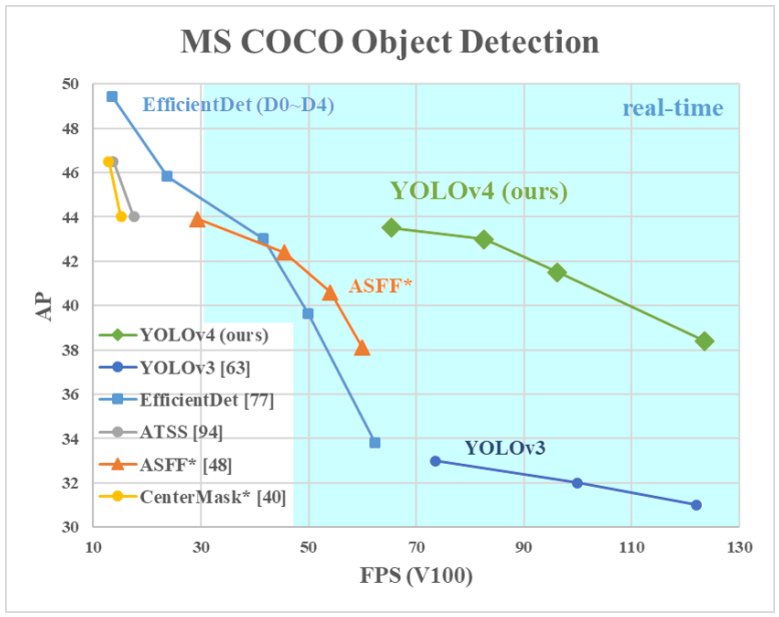
**Head**: YOLOv3

**CSPDarknet53** là một xương sống độc đáo giúp tăng cường năng lực học tập của CNN, phần gộp kim tự tháp không gian được gắn trên đầu CSPDarknet53 để cải thiện trường tiếp nhận và phân biệt các tính năng ngữ cảnh rất quan trọng.

PANet được triển khai theo phương pháp tổng hợp tham số cho các mức dò đặc biệt chứ không phải là Mạng hình tháp đặc trưng (FPN) để phát hiện đối tượng được áp dụng trong YOLOv3. [1]

* + 1. So sánh YOLOv4 với các model khác.

Trong các thử nghiệm, YOLOv4 đã đạt được giá trị AP là 43,5% (65,7% của AP50) so với tập dữ liệu COCO của Microsoft và đạt được tốc độ thời gian thực gần 65 FPS trên Tesla V100, vượt trội so với các máy dò nhanh và chính xác cao trong các chi tiết về cả “tốc độ và độ chính xác”.



Hình 5: So sánh YOLOv4 và các công cụ phát hiện đối tượng hiện đại khác.

YOLOv4 nhanh gấp đôi EfficientDet bên cạnh hiệu quả tương ứng, so với YOLOv3, AP và FPS đã tăng lần lượt 10% và 12%. Bài báo mô tả đầy đủ về độ chính xác và tốc độ đặc biệt của YOLOv4 là những đóng góp tuyệt vời cho lĩnh vực khoa học.

###### **JETSON NANO**

Jetson Nano là 1 bo máy tính nhúng do hãng NVIDIA sản xuất. So với các loại máy tính nhúng khác thì cấu hình của Jetson khá mạnh. Một vài so sánh với Raspberry pi:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **JetSon Nano** | **Raspberry Pi 3** | **Raspberry Pi 4** |
| CPU | Quad-core ARM® A57 CPU | Quad-core ARM Cortex-A53, 1.2GHz. | Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz |
| GPU | 128-core NVIDIA Maxwell GPU | Broadcom VideoCore IV. | Broadcom VideoCore VI. |
| RAM | 4 GB 64-bit LPDDR4 | RAM: 1GB LPDDR2 (900 MHz) | RAM – 1GB, 2GB or 4GB LPDDR4-2400 SDRAM (depending on model) |

Bảng 1: So sánh Jetson nano với một số Raspberry pi

Như chúng ta thấy, điểm nhấn của chiếc máy tính nhúng này là được trang bị 128 core GPU của NVIDIA làm tăng sức mạnh tính toán của chiếc máy tính nhúng này lên rất nhiều. Ngoài ra với 4G RAM giúp nó có thể chạy thêm được nhiều task vụ hơn. Tuy nhiên, do việc tối giản lại hệ điều hành nên việc cài đặt một số package cũng khó khăn hơn. Trong bài này, mình hướng dẫn các bạn cài đặt hệ điều hành cũng như một số package thường dùng trong nghiên cứu và ứng dụng machine learning, deep learning trên con Jetson Nano . Chúng ta bắt đầu thôi!

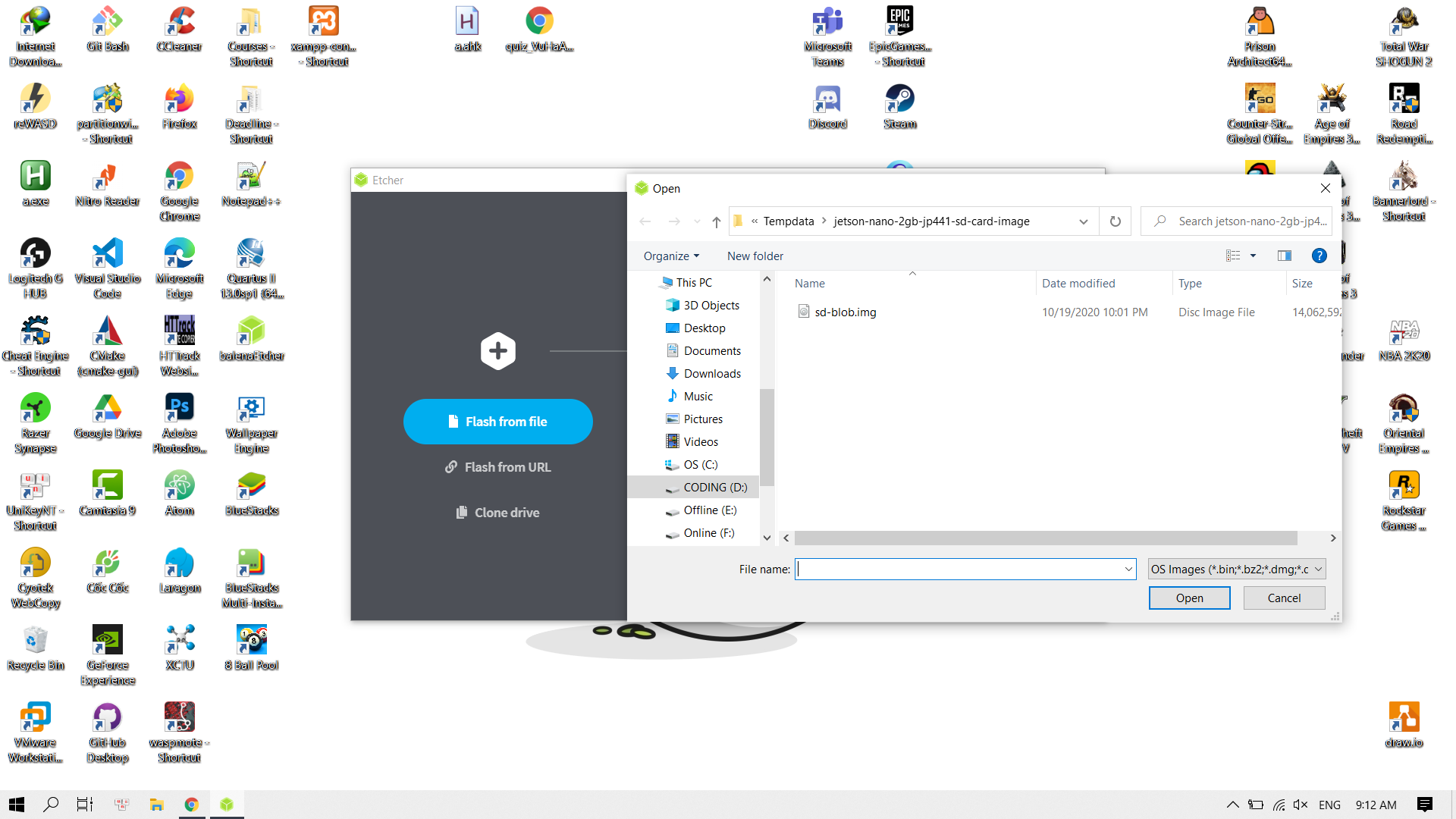
[CHƯƠNG 2: CÁCH CÀI ĐẶT JETSON NANO](#_bookmark15)

## 2.1 Cách cài đặt

Lên trang chủ<https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-2gb-devkit#write> download hệ điều hành mới nhất cho Jetson nano.

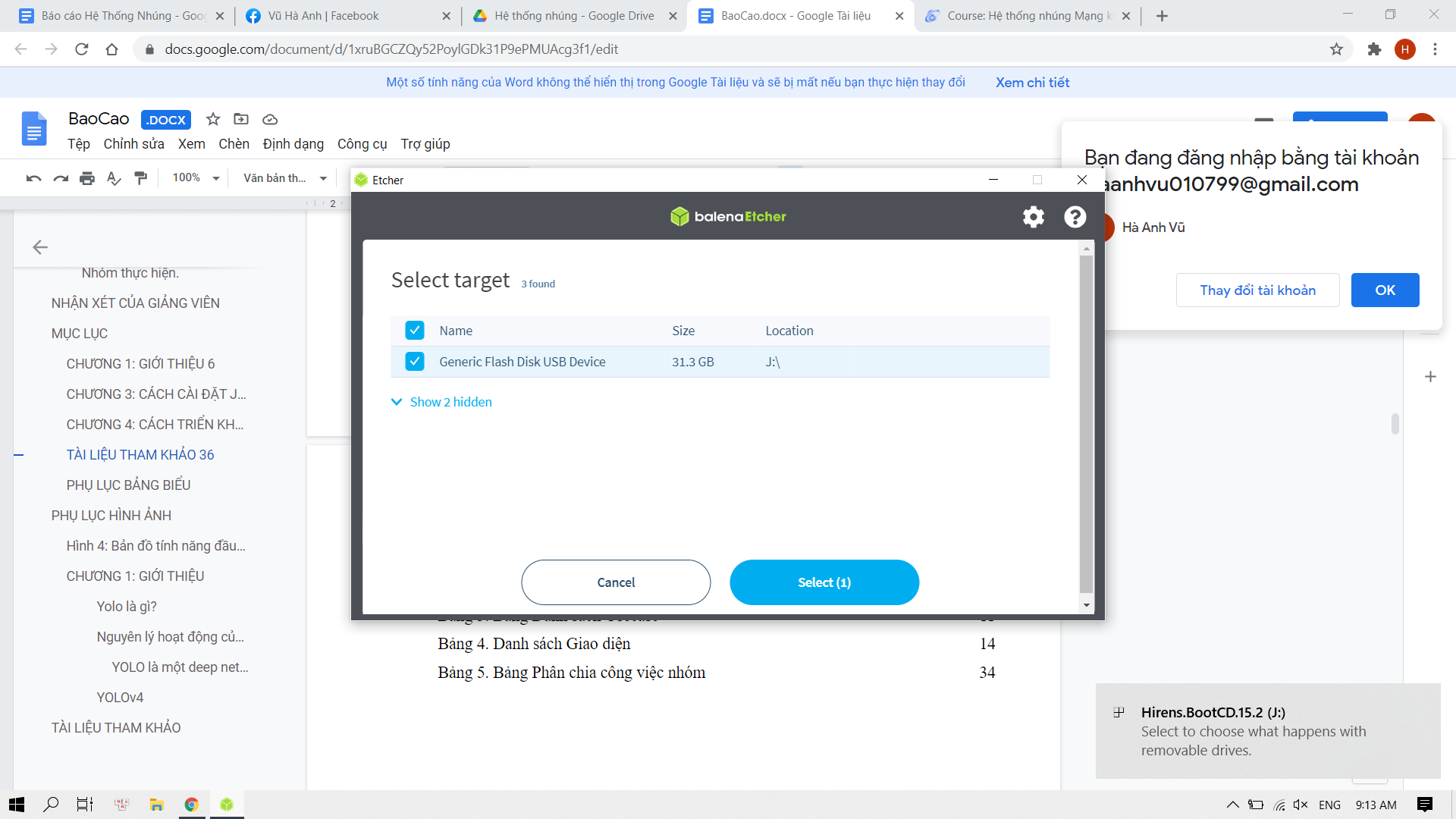
Sử dụng phần mềm balenaEtcher để boot IMG vào thẻ nhớ.

Bước 1: Xác định vị trí của file IMG trong máy tính.



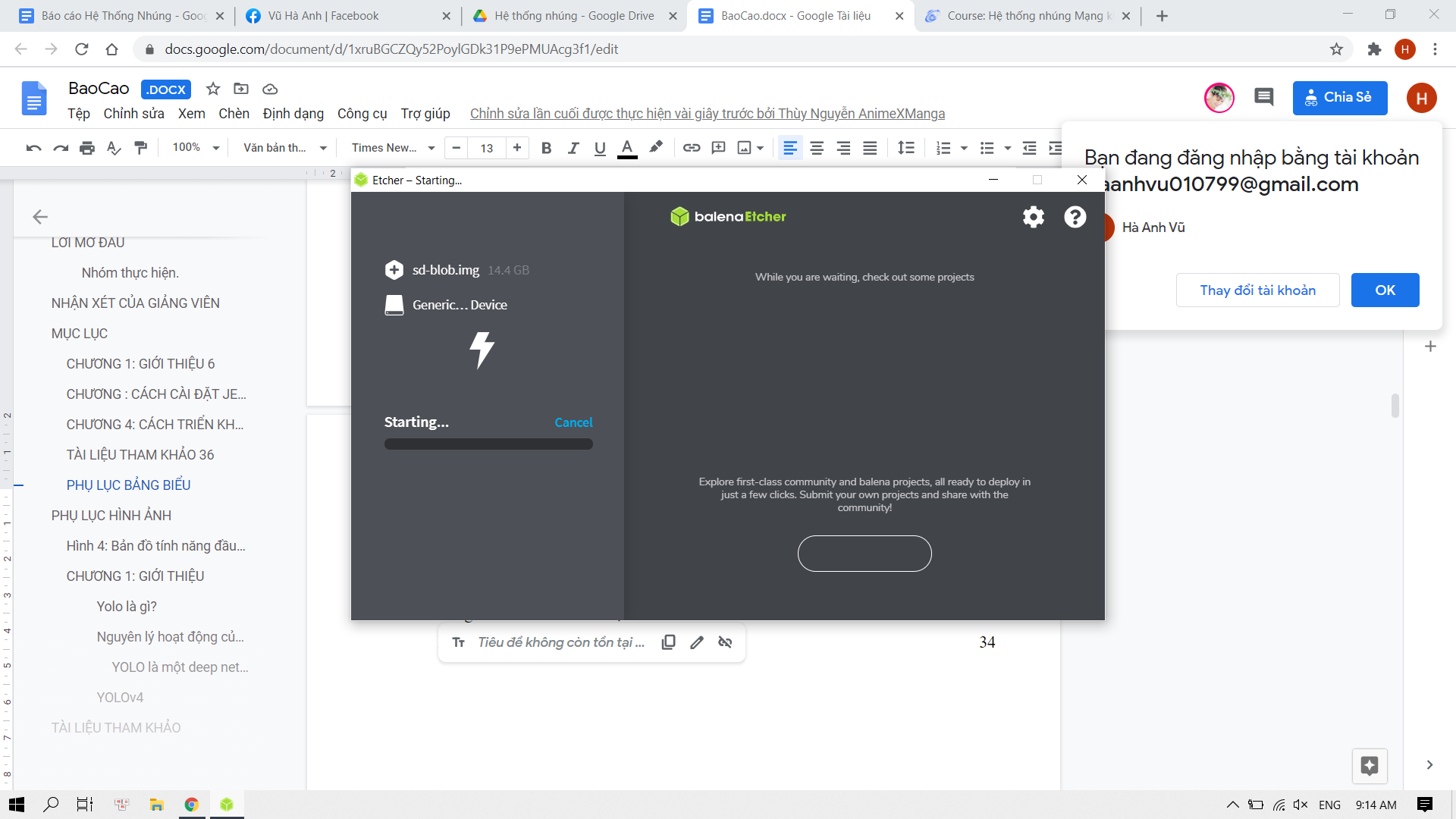
Hình 6: Xác định vị trí của file IMG trong máy tính.

Bước 2: Chọn thiết bị để boot IMG.



Hình 7: Chọn thiết bị để boot IMG.

Bước 3: Sau khi chọn thiết bị, IMG sẽ tự động được boot.



Hình 8: IMG sẽ tự động được boot.

## 2.2 Cách xây dựng model YOLOv4

Bước 1: Sử dụng lệnh git clone <https://github.com/AlexeyAB/darknet> để download model Darknet.

$ git clone https://github.com/AlexeyAB/darknet.git

$ cd darknet

*Darknet là một model thông dụng trong việc Object Detection. Hiện nay, Darknet hỗ trợ sử dụng hai phiên bản YOLOv3 và YOLOv4 trong việc phát hiện thực thể.*

Bước 2: Cấu hình theo thông số dưới đây.

GPU=1

CUDNN=1

CUDNN\_HALF=1

OPENCV=1

AVX=0

OPENMP=0

LIBSO=0

ZED\_CAMERA=0

ZED\_CAMERA\_v2\_8=0

......

USE\_CPP=0

DEBUG=0

ARCH= -gencode arch=compute\_53,code=[sm\_53,compute\_53]

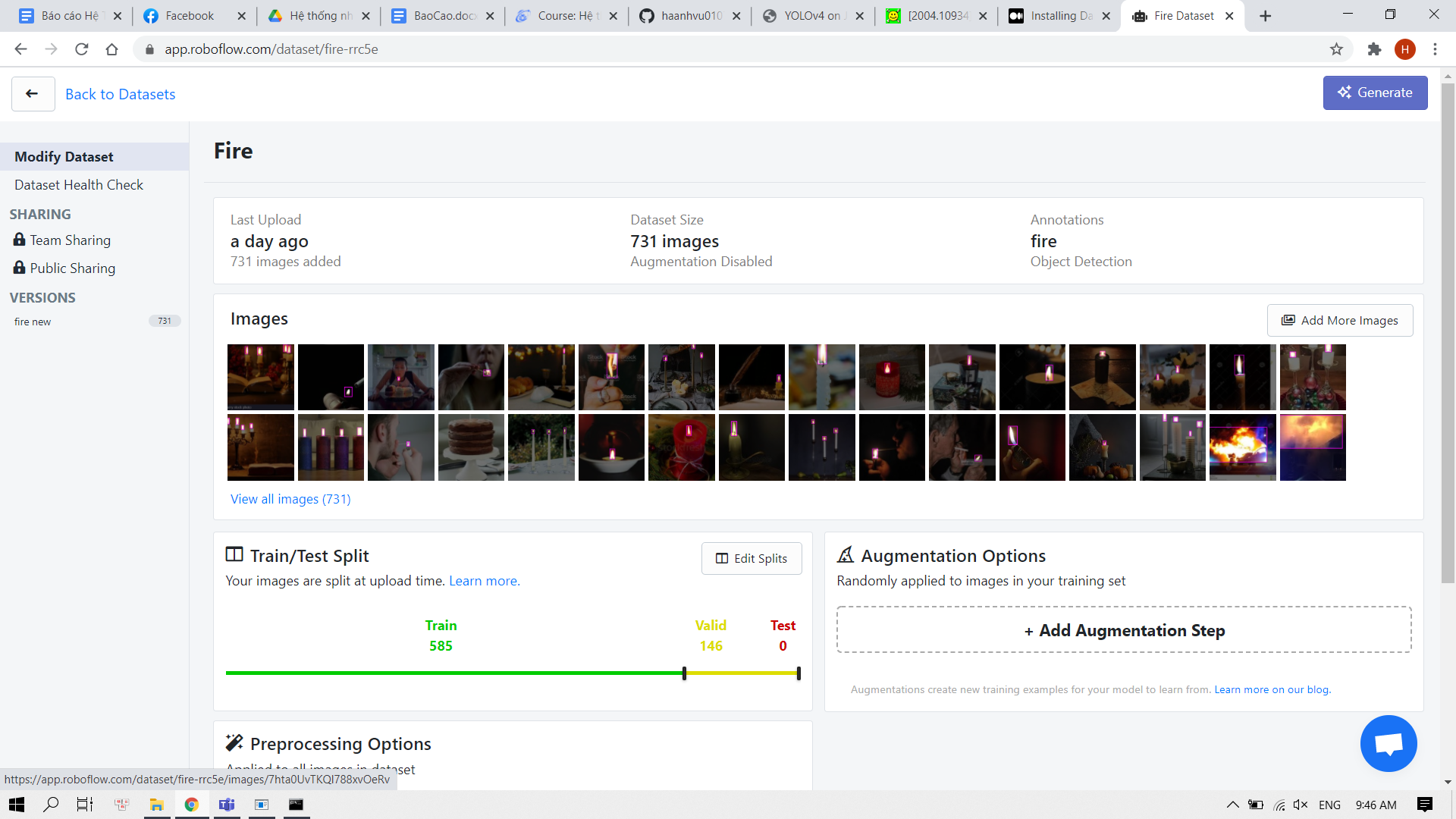
Bước 3: Sau khi cấu hình xong, tiến hành debug model.

$ make

Bước 4: Download pre-trainned model để tiến hành train dữ liệu.

$ wget <https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.weights>

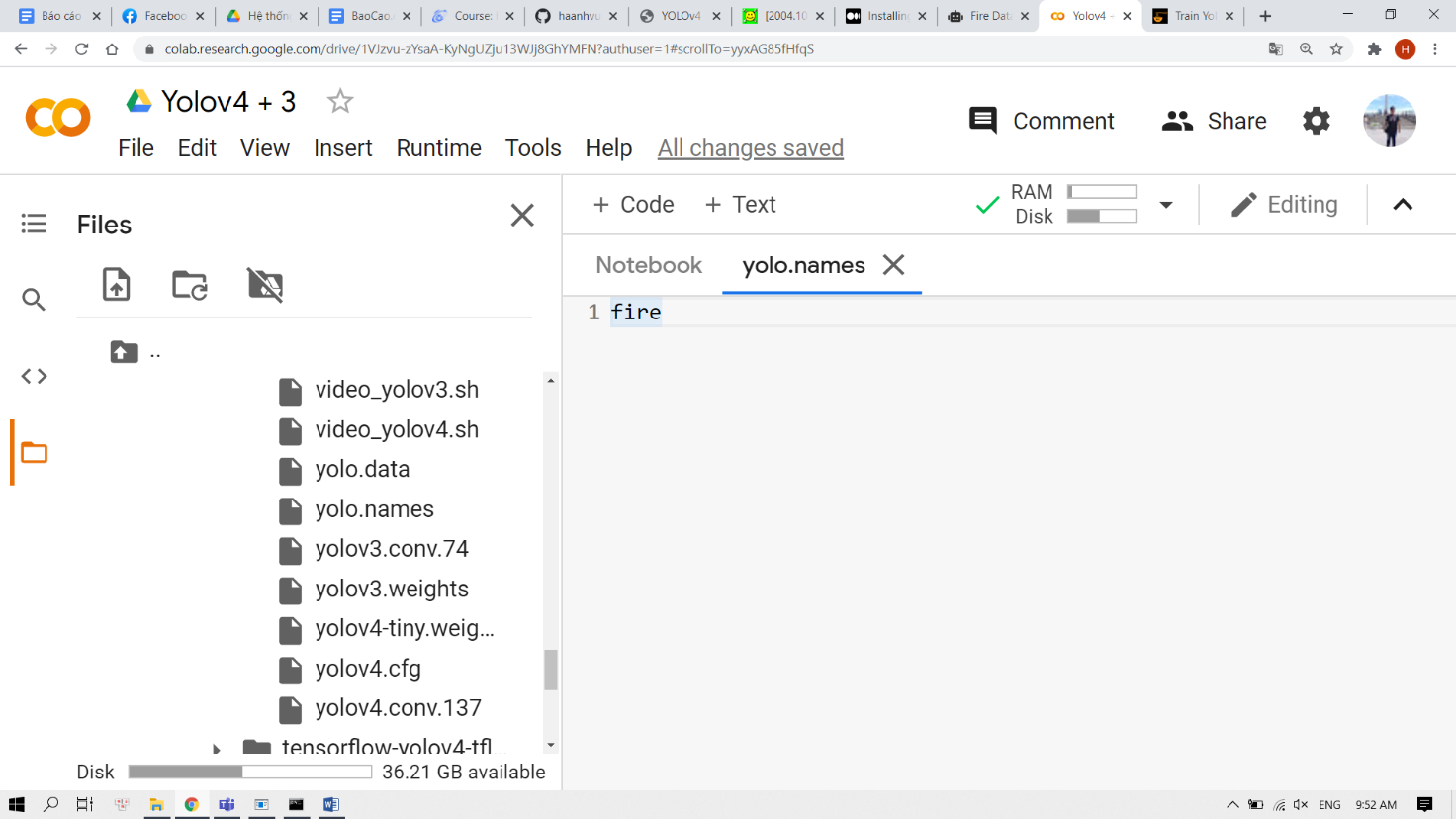
Bước 5: Chuẩn bị data để đưa vào model với tỉ lệ 80% hình ảnh được sử dụng để train và 20% hình ảnh sử dụng dể Valid.

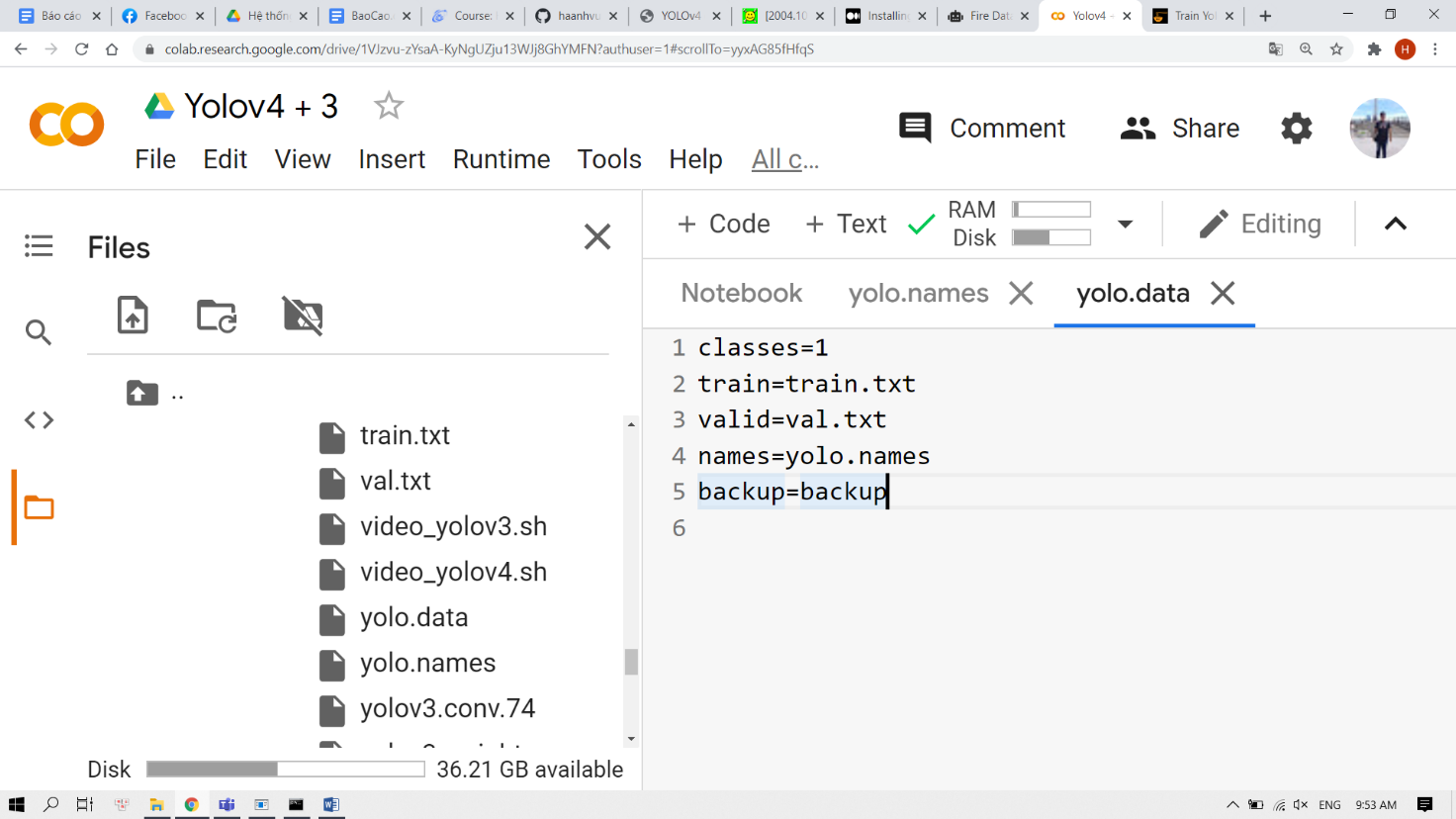


Hình 9: Data để train và valid.

Bước 6: Tiến hành training model. Do việc training model rất tốn tài nguyên của máy tính nên bắt buộc phải sử dụng Server hoặc Cloud để training. Ở đây, ta sử dụng Google colab.

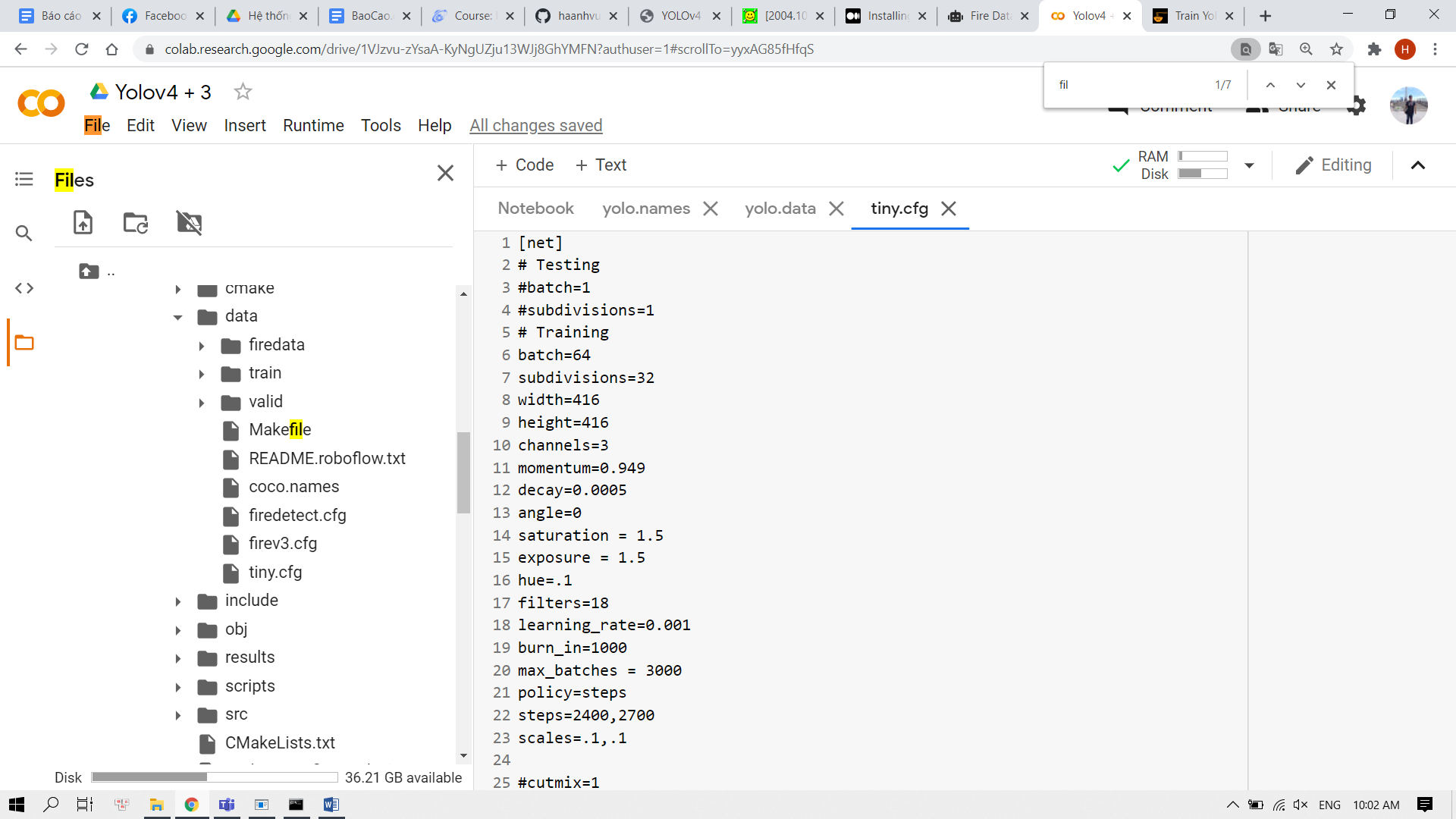
Bước 6.1: Trước khi train chúng ta cần chuẩn bị 2 file là yolo.names dùng để đặt tên cho object cần phát hiện, yolo.data dung để thiết lập các đường dẫn như vị trí thư mục train, valid, thư mục chứa file weights sau khi đã train xong, số lớp thực hiện.



Hình 10: File yolo.names.

Hình 11: File yolo.data.

Bước 6.2: Thiết lập các tham số trong file config như sau để tiết kiệm tài nguyên máy tính.



Hình 12: Cái thông số để cấu hình.

*Tại dòng 6,7 sử dụng để xác định số lượng ảnh trong mỗi vòng train (cấu hình máy càng yếu thì thông số càng cao) với công thức là 2^6*

*Tại dòng 8,9 để định dạng kích thước ảnh.*

*Tại dòng 17, filter = (số class + 5)\*3.*

*Tại dòng 20, max\_batches = max(số class \* 3000).*

*Tại dòng 22, steps lần lượt bằng 80%, 90% của max\_batches.*

Bước 6.3: Sử dụng lệnh này để train.

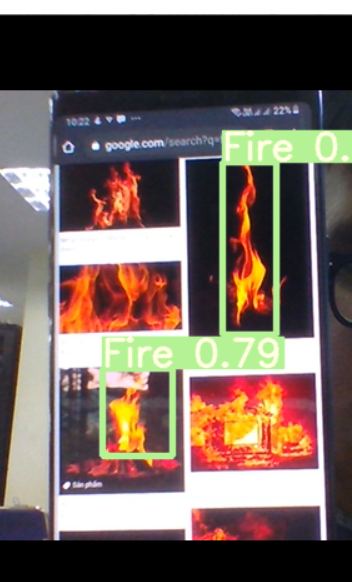
$ !./darknet detector train yolo.data data/tiny.cfg yolov4-tiny.weights -map -clear -dont\_show

Bước 6.4: Sau khi train xong sử dụng lệnh này để valid lại dữ liệu. Model darknet sẽ tự động dung folder valid để thực hiện.

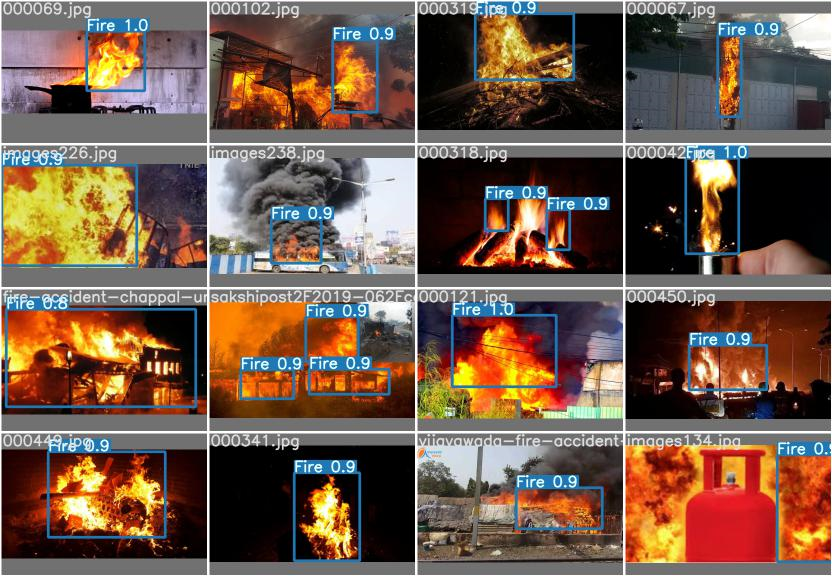
$ *! ./darknet detector valid yolo.data data/tiny.cfg backup/tiny\_best.weights*

Bước 7: Phát hiện vật thể (sử dụng Jetson nano)

$ *./darknet detector demo yolo.data data/tiny.cfg backup/tiny\_best.weights*



Hình 13: Real-time Object Detection.



Hình 14: Image Object Detection.

# TÀI LIỆU THAM KHẢO

Tiếng anh:

1. <https://www.analyticssteps.com/blogs/introduction-yolov4>
2. https://github.com/AlexeyAB/darknet

Tiếng việt:

1. *https://viblo.asia/p/tim-hieu-ve-yolo-trong-bai-toan-real-time-object-detection-yMnKMdvr57P*
2. *https://forum.machinelearningcoban.com/t/object-detection-yolo/503*