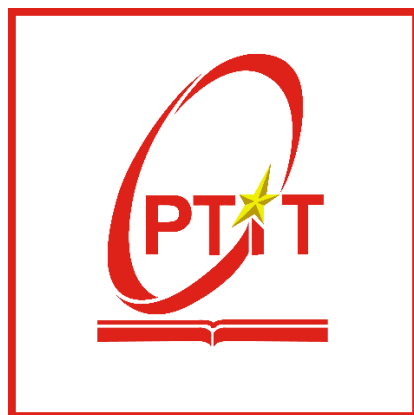


HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



BÁO CÁO BÀI TẬP LỚN

Đề Tài:

Nhận diện khối u bằng YOLOv10

Khoa công nghệ thông tin

Họ và tên: Lê Phương Nam

MSV: B22DCCN555

Lớp: D22DCCN03-B

Giảng Viên Hướng Dẫn: Vũ Minh Mạnh

Hà Nội 2024

Mục Lục

CHƯƠNG 1: TỔNG QUAN	3
1.1 Giới thiệu	3
1.2 Mục tiêu đề tài	3
1.3 Giới hạn đề tài	3
1.4 Phương pháp nghiên cứu	3
1.5 Đối tượng và phạm vi nghiên cứu.....	4
1.6 Bố cục quyền báo cáo	4
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT	4
2.1 Tổng quan về YOLO	5
2.1.1 Kiến trúc mạng YOLOv10.....	5
2.1.2 Nguyên lý hoạt động của mạng YOLO	6
2.2 Output của YOLO.....	8
CHƯƠNG 3 : THIẾT KẾ HỆ THỐNG	11
3.1 Tập dữ liệu chuẩn bị cho quá trình huấn luyện.....	11
CHƯƠNG 4: KẾT QUẢ THẨM ĐỊNH.....	14
CHƯƠNG 5: ĐỊNH DẠNG ĐẦU VÀO TRƯỚC VÀ SAU.....	15
TÀI LIỆU THAM KHẢO	17

CHƯƠNG 1: TỔNG QUAN

1.1 Giới thiệu

Ngày nay, trí tuệ nhân tạo-Artificial Intelligence (AI) đang ngày càng phổ biến và góp phần thay đổi sâu sắc nhiều khía cạnh trong cuộc sống hằng ngày. Trong đó thị giác máy tính-Computer Vision (CV) là một lĩnh vực quan trọng của AI bao gồm các phương pháp thu nhận, xử lý ảnh kỹ thuật số, phân tích và nhận dạng các hình ảnh.

Mạng Noron học sâu (Deep learning Network) là lĩnh vực nghiên cứu các thuật toán, chương trình máy tính để máy tính có thể học tập và đưa ra những dự đoán như con người. Nó được ứng dụng vào nhiều ứng dụng khác nhau như khoa học, kỹ thuật, các lĩnh vực đời sống khác cũng như các ứng dụng về phân loại và phát hiện đối tượng. Một ví dụ điển hình là CNN (Convolutional Neural Network) áp dụng để nhận dạng tự động, tìm hiểu các mẫu phân biệt từ ảnh bằng cách xếp chồng liên tiếp các lớp lên nhau và trong nhiều ứng dụng, CNN hiện nay được coi là trình phân loại ảnh mạnh và thúc đẩy các công nghệ trong lĩnh vực thị giác máy tính, làm đòn bẩy cho quá trình học máy. Nhưng bên cạnh đó, để phân loại được một đối tượng thì công nghệ CNN tiêu tốn cực lớn về tài nguyên như băng thông, bộ nhớ và khả năng xử lý của phần cứng.

Để giảm thiểu những tài nguyên tiêu hao này, những thuật toán, mô hình giải thuật theo thời gian được ra đời ngày càng nhiều và trong đó có **mô hình YOLOv10** cho bài toán nhận diện, cụ thể là ứng dụng vào đề tài “**Nhận diện khối u**”.

1.2 Mục tiêu đề tài

- Vận dụng được những kiến thức cơ bản về huấn luyện mạng nơ-ron.
- Xây dựng được một mô hình có khả năng huấn luyện các tập dữ liệu khối u khác nhau
- Nhận diện được tất cả các khối u có trong tập dữ liệu.

1.3 Giới hạn đề tài

- Trong đề tài này chỉ nhận diện được 4 loại khối u: *U tuyến yên*, *U màng não*, *U thần kinh đệm* và *Không có khối u*
- Tập dữ liệu có số lượng khoảng 4000 ảnh.

1.4 Phương pháp nghiên cứu

- Dựa trên các kiến thức đã học về cách huấn luyện một mạng nơ-ron.
- Thu thập tài liệu, tham khảo những ứng dụng liên quan đã có trước đó.

1.5 Đối tượng và phạm vi nghiên cứu

- Nhận dạng các khối u có trong tập dữ liệu, ở đây là 4 khối u: *U tuyến yên*, *U màng não*, *U thần kinh nệm* và *Không có khối u*

1.6 Bố cục quyển báo cáo

Đề tài có tổng cộng 5 chương:

- **Chương 1 - Tổng quan**

Trong chương này tìm hiểu về các vấn đề hình thành nên đề tài. Kèm theo đó là một số nội dung và giới hạn của đề tài mà nhóm thực hiện đề tài đã đặt ra.

- **Chương 2 – Cơ sở lý thuyết**

Giới thiệu về các kiến thức nền tảng cũng như công nghệ và phần mềm được sử dụng trong đề tài bao gồm kiến thức về xử lý ảnh, lý thuyết mạng nơ-ron, đặc điểm và cách huấn luyện một tập dữ liệu trong YOLOv10.

- **Chương 3 – Thiết kế hệ thống**

Lên kế hoạch sử dụng tập mẫu, diễn giải các thông số của mô hình, quá trình huấn luyện, quá trình kiểm tra và thiết kế một hệ thống nhận diện động trên nền tảng YOLOv10.

- **Chương 4 – Kết quả**

Kiểm tra kết quả của quá trình huấn luyện, kiểm tra mô hình hệ thống.

- **Chương 5- Định dạng đầu vào trước và sau**

Trong chương này sẽ trình bày những kết quả của đề tài đã đạt được so với mục tiêu đặt ra, nêu ra một số hướng nghiên cứu và phát triển cho đề tài.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

Trong vài năm trở lại đây, Object detection là một trong những đề tài rất hot của deep learning bởi khả năng ứng dụng cao, dữ liệu dễ chuẩn bị và kết quả ứng dụng thì cực kì nhiều. Các thuật toán mới của Object detection như YOLO, SSD có tốc độ khá nhanh và độ chính xác cao nên giúp cho Object Detection có thể thực hiện được các tác vụ đường như là real time, thậm chí là nhanh hơn so với con người

mà độ chính xác không giảm. Các mô hình cũng trở nên nhẹ hơn nên có thể hoạt động trên các thiết bị IoT để tạo nên các thiết bị thông minh.

2.1 Tổng quan về YOLO

YOLO(You only look once) 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 đó 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]

YOLO có thể không phải là thuật toán tốt nhất nhưng nó là thuật toán nhanh nhất trong các lớp mô hình object detection. Nó có thể đạt được tốc độ gần như real time mà độ chính xác không quá giảm so với các model thuộc top đầu.

YOLO là thuật toán object detection nên mục tiêu của mô hình không chỉ là dự báo nhãn cho vật thể như các bài toán classification mà nó còn xác định location của vật thể. Do đó YOLO có thể phát hiện được nhiều vật thể có nhãn khác nhau trong một bức ảnh thay vì chỉ phân loại duy nhất một nhãn cho một bức ảnh.

Một trong những ưu điểm mà YOLO đem lại đó là chỉ sử dụng thông tin toàn bộ bức ảnh một lần và dự đoán toàn bộ object box chứa các đối tượng, mô hình được xây dựng theo kiểu end-to-end nên được huấn luyện hoàn toàn bằng gradient descent.

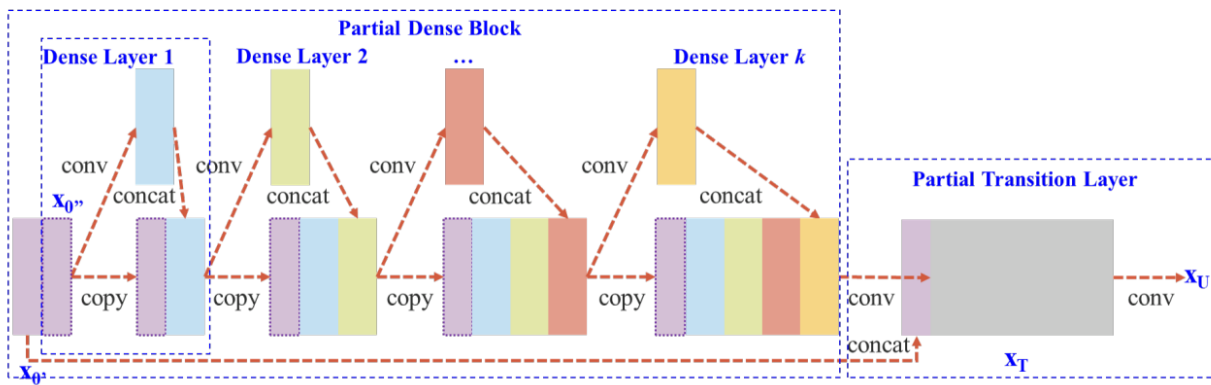
Tính đến thời điểm hiện tại YOLO đã có tổng cộng 11 phiên bản(v1 đến v11). Trong đó bản v10 là bản mà được lựa chọn để sử dụng đưa vào đề tài với ưu điểm hơn các phiên bản trước về mặt tốc độ trong thời gian thực tốt hơn

2.1.1 Kiến trúc mạng YOLOv10

Kiến trúc YOLO bao gồm: Base network là các mạng convolution làm nhiệm vụ trích xuất đặc trưng. Phần phía sau là những Extra Layers được áp dụng để phát hiện vật thể trên feature map của base network.

Base network của YOLO sử dụng chủ yếu là các convolutional layer và các fully connected layer. Các kiến trúc YOLO cũng khá đa dạng và có thể tùy biến thành các version cho nhiều input shape khác nhau.

YOLOv10 sử dụng mạng CSPnet giống những phiên bản khác nhưng được cải thiện hơn một chút để tốc độ tốt hơn



(b) Cross Stage Partial DenseNet

Hình : Kiến trúc mạng CSPnet

Chia các đặc trưng đầu vào thành hai nhánh:

- Ở đầu mỗi tầng (stage), các đặc trưng đầu vào sẽ được chia thành hai phần: một phần sẽ đi qua các khối xử lý trích xuất đặc trưng, phần còn lại giữ nguyên mà không qua xử lý. Ví dụ, nếu tầng đầu vào có 256 kênh đặc trưng, ta có thể chia thành hai nhánh với 128 kênh mỗi nhánh.

Trích xuất đặc trưng với nhánh đầu tiên:

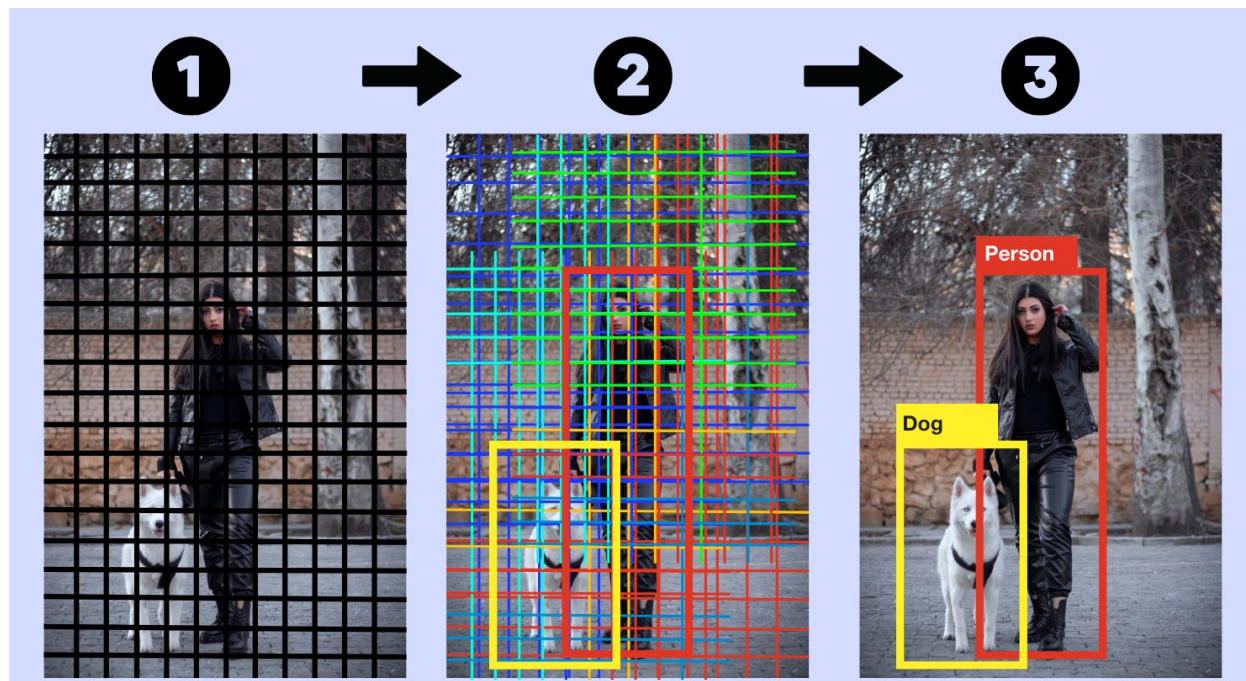
- Nhánh đầu tiên (nhánh trích xuất) sẽ qua một chuỗi các khối mạng như ResNet block, DenseNet block, hoặc các khối convolution để trích xuất thông tin đặc trưng sâu hơn.
- Mỗi khối sẽ tạo ra các đặc trưng riêng dựa trên các phép convolution và kích hoạt, giống như trong kiến trúc gốc của ResNet hoặc DenseNet

Kết hợp hai nhánh:

- Sau khi nhánh đầu tiên đã hoàn tất việc trích xuất đặc trưng, cả hai nhánh (đặc trưng gốc và đặc trưng mới) sẽ được kết hợp lại bằng cách nối (concatenate) hoặc cộng (element-wise add) hai nhánh này.
- Kết quả sau khi kết hợp chứa cả thông tin gốc từ đầu vào và đặc trưng được trích xuất, giảm trùng lặp gradient và giữ lại thông tin cần thiết để mô hình không mất thông tin từ đầu vào.

2.1.2 Nguyên lý hoạt động của mạng YOLO

Đầu vào của mô hình là một ảnh, mô hình sẽ nhận dạng ảnh đó có đối tượng nào hay không, sau đó sẽ xác định tọa độ của đối tượng trong bức ảnh. Ảnh đầu vào được chia thành $S \times S$ ô thường thì sẽ là $3 \times 3, 7 \times 7, 9 \times 9, \dots$. Việc chia ô có ảnh hưởng đến việc phát hiện đối tượng của mô hình.



Cách hoạt động của mạng YOLO

Trong mô hình YOLO (You Only Look Once), đầu ra của mạng được tổ chức dưới dạng một ma trận ba chiều có kích thước $S \times S \times (5 \times N + M)$ với các thông số được giải thích như sau:

- **S:** Kích thước của lưới (grid) chia ảnh, ví dụ như 7×7 (lưới 7×7 ô).
- **N:** Số lượng bounding boxes mà mỗi ô lưới cần dự đoán (ví dụ: 2 bounding boxes cho mỗi ô).
- **M:** Số lượng lớp đối tượng mà mỗi ô cần dự đoán (ví dụ: 3 lớp đối tượng: chó, ô tô, xe đạp).
- **5:** Mỗi bounding box bao gồm 5 tham số:
 1. Tọa độ trung tâm (x, y) của bounding box so với ô lưới.

2. Chiều rộng và chiều cao của bounding box, tỉ lệ với kích thước của lưới.
3. Độ tin cậy của bounding box (objectness score), cho biết xác suất có đối tượng trong bounding box đó.

Với Input là 1 ảnh, đầu ra mô hình là một ma trận 3 chiều có kích thước $S \times S \times (5 \times N + M)$ với số lượng tham số mỗi ô là $(5 \times N + M)$ với N và M lần lượt là số lượng Box và Class mà mỗi ô cần dự đoán. Xét ví dụ ở hình trên chia thành 7×7 ô, mỗi ô cần dự đoán 2 bounding box và 3 object: con chó, ô tô, xe đạp thì output sẽ là $7 \times 7 \times 13$, mỗi ô sẽ có 13 tham số, cho kết quả trả về $(7 \times 7 \times 13 = 98)$ bounding box.

2.2 Output của YOLO

Output của mô hình YOLO là một véc tơ sẽ bao gồm các thành phần:

$$y^T = [\rho_0, \underbrace{\langle t_x, t_y, t_w, t_h \rangle}_{\text{boundingbox}}, \underbrace{\langle p_1, p_2, \dots, p_c \rangle}_{\text{score of } c \text{ classes}}]$$

Trong đó:

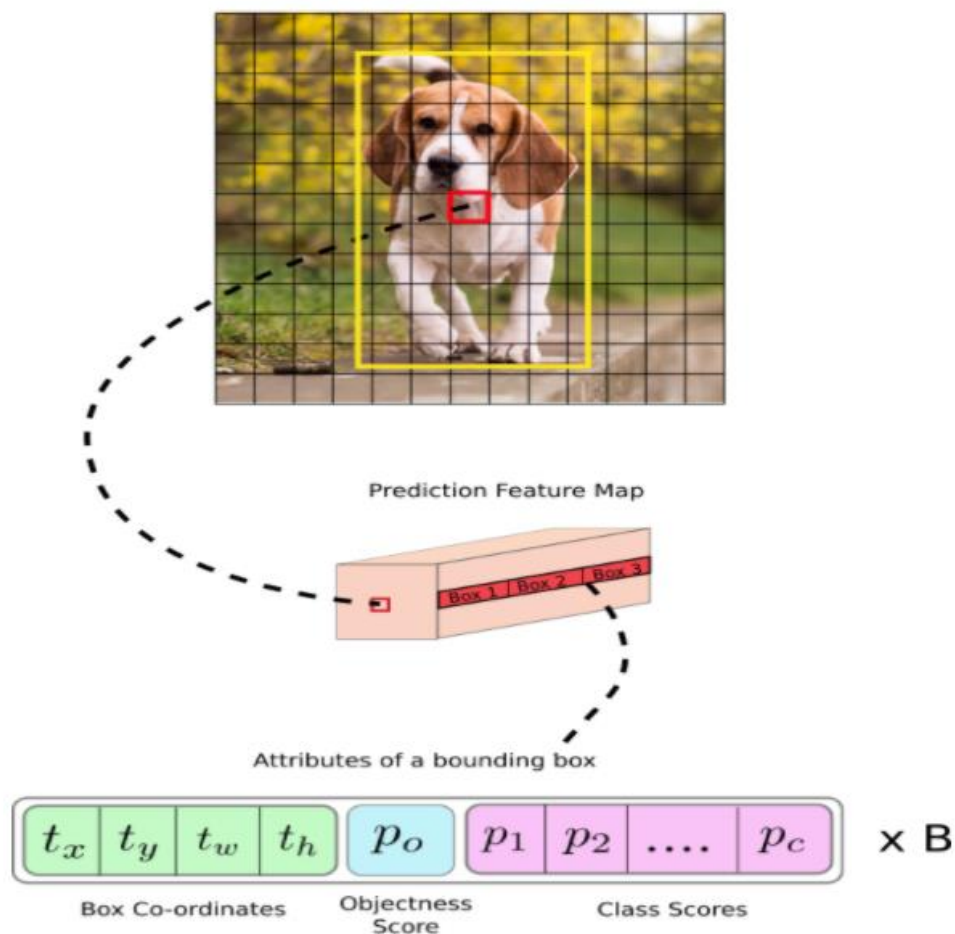
ρ_0 : là xác suất dự báo vật thể xuất hiện trong bounding box.

$\underbrace{\langle t_x, t_y, t_w, t_h \rangle}_{\text{boundingbox}}$: giúp xác định bounding box. Trong đó t_x, t_y là tọa độ tâm và t_w, t_h là kích thước rộng, dài của bounding box.

$\underbrace{p_1, p_2, \dots, p_c}_{\text{score of } c \text{ classes}}$: là véc tơ phân phối xác suất dự báo của các classes.

Việc hiểu output khá là quan trọng để chúng ta cấu hình tham số chuẩn xác khi huấn luyện model qua các open source như darknet. Như vậy output sẽ được xác định theo số lượng classes theo công thức $(n_class + 5)$. Nếu huấn luyện 80 classes thì bạn sẽ có output là 85. Trường hợp bạn áp dụng 3 anchors/cell thì số lượng tham số output sẽ là:

$$(n_class + 5) \times 3 = 85 \times 3 = 255$$



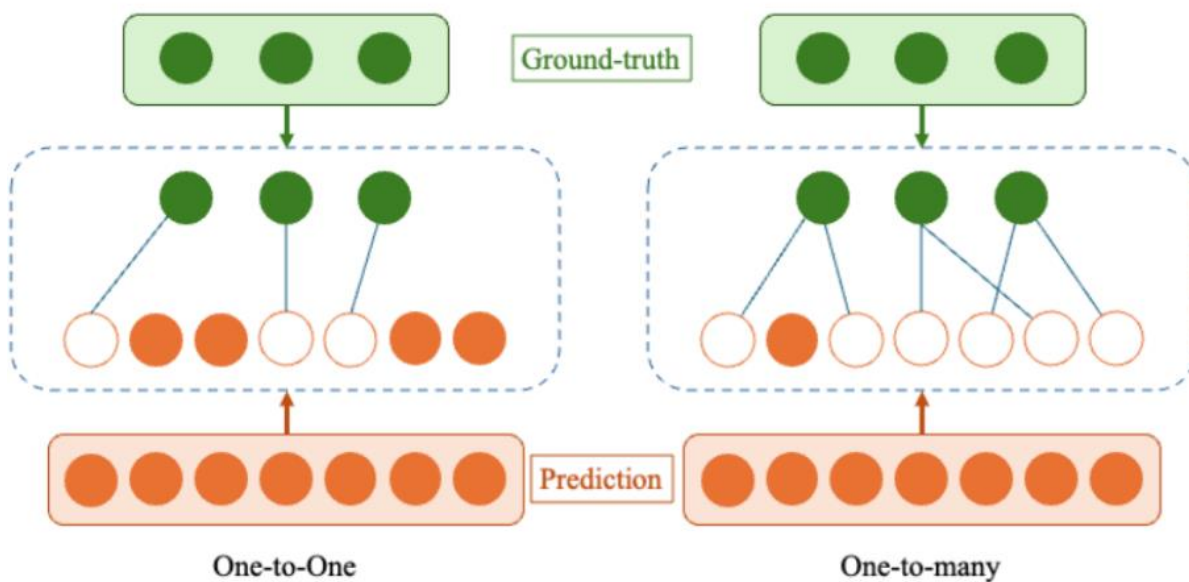
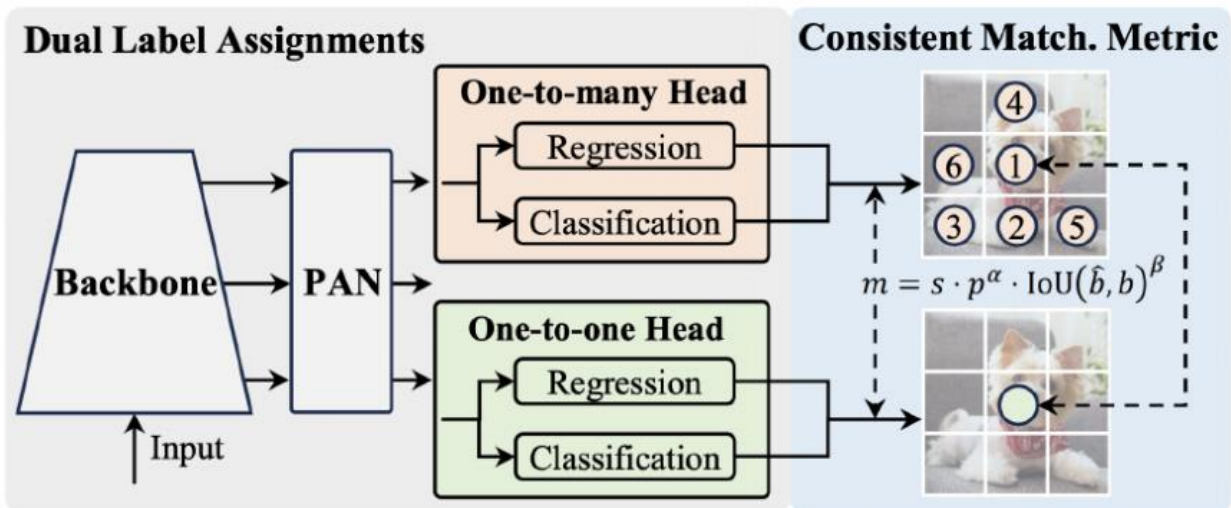
Kiến trúc một output của model YOLO

Hình ảnh gốc là một feature map kích thước 13x13. Trên mỗi một cell của feature map chúng ta lựa chọn ra 3 anchor boxes với kích thước khác nhau lần lượt là Box 1, Box 2, Box 3 sao cho tâm của các anchor boxes trùng với cell. Khi đó output của YOLO là một véc tơ concatenate của 3 bounding boxes. Các attributes của một bounding box được mô tả như dòng cuối cùng trong hình

Điểm khác biệt của YOLOv10: Khác với những phiên bản trước đây của YOLO. Ở YOLOv10 giải quyết vấn đề về các dự đoán dư thừa trong quá trình xử lý sau bằng cách giới thiệu chiến lược gán nhãn kép nhất quán cho YOLO không cần NMS với các gán nhãn kép và thước đo đối sánh nhất quán. Điều này cho phép mô hình tận hưởng sự giám sát phong phú và hài hòa trong quá trình

huấn luyện đồng thời loại bỏ nhu cầu sử dụng NMS trong quá trình suy luận, dẫn đến hiệu suất cạnh tranh với hiệu quả cao.

YOLOv10 dùng chiến lược: One to one and One to many mang tên Dual label assignments



CHƯƠNG 3 : THIẾT KẾ HỆ THỐNG

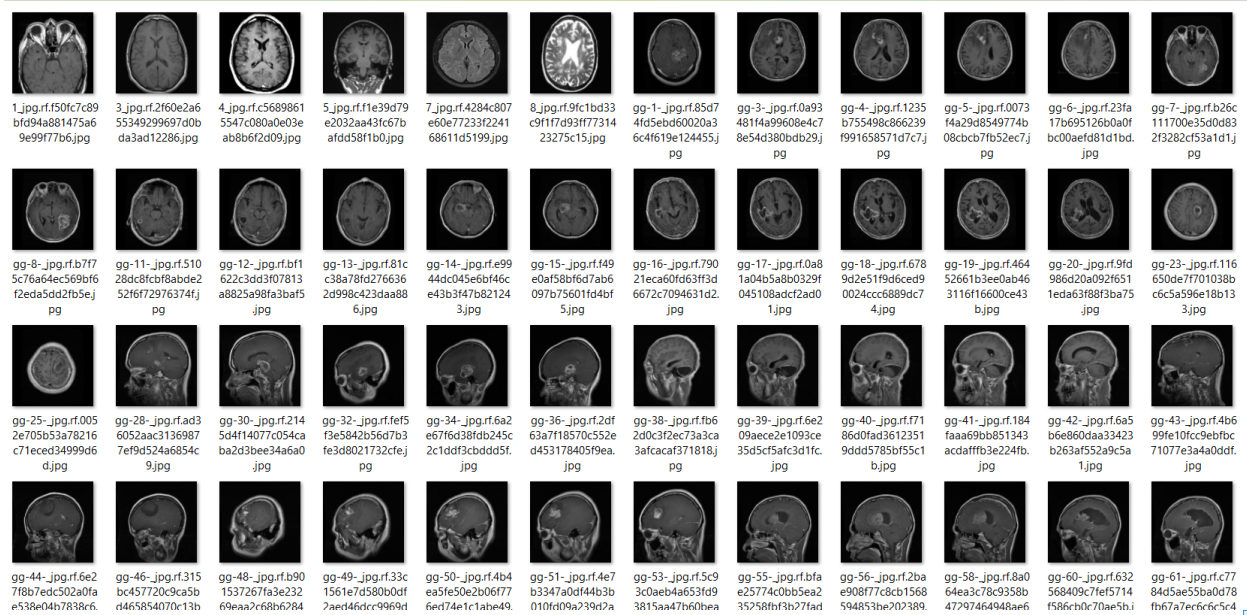
3.1 Tập dữ liệu chuẩn bị cho quá trình huấn luyện

Gồm 3904 ảnh với 3408 ảnh để train và 496 ảnh thẩm định test lại











Mỗi một ảnh có một file text đi kèm riêng xác định về các boundingbox được chuẩn hóa tỉ lệ 0-1 và label của ảnh. Được đăng tải lên Roboflow để không phải mất thời gian trên drive hay colab khi tải một lượng lớn ảnh và đỡ tốn tài nguyên hơn

images	21/08/2024 9:15 SA	File folder	
labels	21/08/2024 9:15 SA	File folder	
labels.cache	09/09/2024 1:56 CH	CACHE File	1.027 KB

Tập images và tập labels



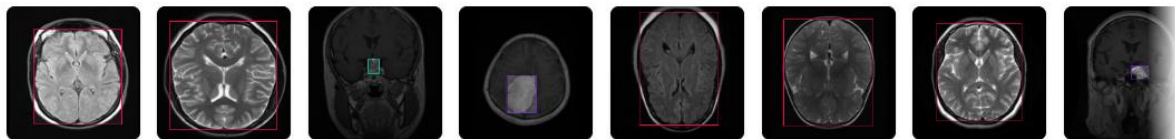
Hình ảnh về tập images

 1_jpg.rf.f50fc7c89bfd94a881475a69e99f7...	20/08/2024 4:47 CH	Text Document	1 KB
 3_jpg.rf.2f60e2a655349299697d0bda3ad...	20/08/2024 4:47 CH	Text Document	1 KB
 4_jpg.rf.c56898615547c080a0e03eab8b6f...	20/08/2024 4:47 CH	Text Document	1 KB
 5_jpg.rf.f1e39d79e2032aa43fc67bafdd58...	20/08/2024 4:47 CH	Text Document	1 KB
 7_jpg.rf.4284c807e60e77233f224168611d...	20/08/2024 4:47 CH	Text Document	1 KB
 8_jpg.rf.9fc1bd33c9f1f7d93ff7731423275...	20/08/2024 4:47 CH	Text Document	1 KB
 gg-1-_jpg.rf.85d74fd5ebd60020a36c4f61...	20/08/2024 4:47 CH	Text Document	1 KB
 gg-3-_jpg.rf.0a93481f4a99608e4c78e54d...	20/08/2024 4:47 CH	Text Document	1 KB
 gg-4-_jpg.rf.1235b755498c866239f99165...	20/08/2024 4:47 CH	Text Document	1 KB
 gg-5-_jpg.rf.0073f4a29d8549774b08cbcb...	20/08/2024 4:47 CH	Text Document	1 KB

Hình ảnh về tập labels

3904 Total Images

[View All Images →](#)



Dataset Split

TRAIN SET

87%

3408 Images

VALID SET

13%

496 Images

TEST SET

%

0 Images

Preprocessing

Auto-Orient: Applied

Resize: Stretch to 640x640

Augmentations

No augmentations were applied.

Thực hiện tải lên Roboflow

Thực hiện cài đặt phiên bản YOLOv10

```
✓ [1] !pip install -q git+https://github.com/THU-MIG/yolov10.git
15s
Installing build dependencies ... done
Getting requirements to build wheel ... done
Preparing metadata (pyproject.toml) ... done
Building wheel for ultralytics (pyproject.toml) ... done

✓ [2] !wget -P -q https://github.com/jameslahm/yolov10/releases/download/v1.0/yolov10n.pt
0s
--2024-11-11 09:52:32-- https://github.com/jameslahm/yolov10/releases/download/v1.0/yolov10n.pt
Resolving github.com (github.com)... 20.205.243.166
Connecting to github.com (github.com)|20.205.243.166|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://github.com/THU-MIG/yolov10/releases/download/v1.0/yolov10n.pt [following]
--2024-11-11 09:52:32-- https://github.com/THU-MIG/yolov10/releases/download/v1.0/yolov10n.pt
Reusing existing connection to github.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/804788522/de01476f-8157-4901-921f-e0c6cb3848cf?X-Amz-Algo=
--2024-11-11 09:52:32-- https://objects.githubusercontent.com/github-production-release-asset-2e65be/804788522/de01476f-8157-4901-921f-e0c6cb38
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 11447983 (11M) [application/octet-stream]
Saving to: '-q/yolov10n.pt'

yolov10n.pt      100%[=====>]  10.92M  --.-KB/s  in 0.04s

2024-11-11 09:52:32 (247 MB/s) - '-q/yolov10n.pt' saved [11447983/11447983]
```

Thực hiện tải dữ liệu về từ Roboflow

```
✓ [8] from roboflow import Roboflow
2s
rf = Roboflow(api_key = "uXiv8nV5HdWu7I2zRCNH")
project = rf.workspace("projectpearpn").project("brain-tumour-classification")
version = project.version(2)
dataset = version.download("yolov8")

loading Roboflow workspace...
loading Roboflow project...
```

Thực hiện train mô hình với 30 epoch và 40 batch

```
!yolo task=detect mode=train epochs = 30, batch=40 plots=True \
model = '/content/-q/yolov10n.pt'\
data = '/content/Brain-Tumour-classification-2/data.yaml'
```

WARNING ⚠ argument 'epochs=30,' does not require trailing comma ',', updating to 'epochs=30'.
/usr/local/lib/python3.10/dist-packages/ultralytics/nn/tasks.py:733: FutureWarning: You are using `torch.load` with `weights_only=False` which is unsafe. Please update your code to explicitly use `weights_only=True` to load only the file data. See https://pytorch.org/docs/stable/generated/torch.load.html for more details.

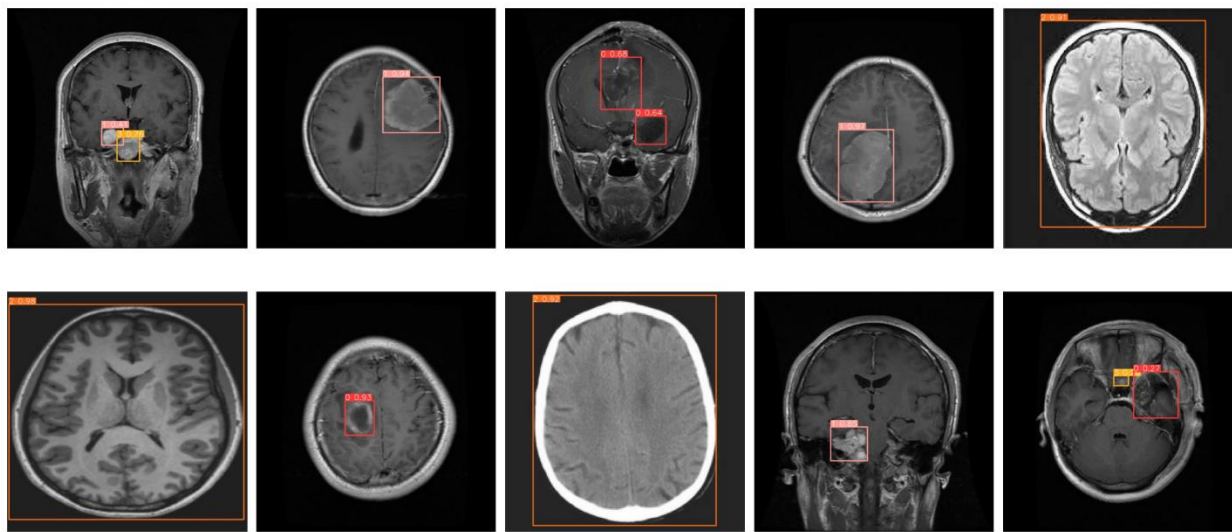
New <https://pypi.org/project/ultralytics/8.3.28> available 🤗 Update with 'pip install -U ultralytics'
Ultralytics YOLOv8.1.34 Python-3.10.12 torch-2.5.0+cu121 CUDA:0 (Tesla T4, 15102MiB)
engine/trainer: task=detect, mode=train, model=/content/-q/yolov10n.pt, data=/content/Brain-Tumour-classification-2/data.yaml, epochs=30, batch=40, imgsz=640, device=-1, verbose=True, seed=0, name=train
Downloading <https://ultralytics.com/assets/Arial.ttf> to '/root/.config/yolov10/Arial.ttf'...
100% 755k/755k [00:00<00:00, 81.1MB/s]

2024-11-11 09:55:02.017295: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:485] Unable to register cuFFT factory: Attemptin
2024-11-11 09:55:02.036949: E external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:8454] Unable to register cuDNN factory: Attempti
2024-11-11 09:55:02.043046: E external/local_xla/xla/stream_executor/cuda/cuda_blas.cc:1452] Unable to register cuBLAS factory: Attemp
Overriding model.yaml nc=80 with nc=4

	from	n	params	module	arguments
0	-1	1	464	ultralytics.nn.modules.conv.Conv	[3, 16, 3, 2]
1	-1	1	4672	ultralytics.nn.modules.conv.Conv	[16, 32, 3, 2]
2	-1	1	7360	ultralytics.nn.modules.block.C2f	[32, 32, 1, True]
3	-1	1	18560	ultralytics.nn.modules.conv.Conv	[32, 64, 3, 2]
4	-1	2	49664	ultralytics.nn.modules.block.C2f	[64, 64, 2, True]

CHƯƠNG 4: KẾT QUẢ THẨM ĐỊNH

```
YOLOv10n summary (fused): 285 layers, 2695976 parameters, 0 gradients, 8.2 GFLOPs
Class      Images  Instances  Box(P   R   mAP50  mAP50-95): 100% 7/7 [00:10:00:00, 1.47s/it]
  all         496       541    0.936  0.888   0.95   0.765
    0         496       153    0.917  0.837   0.916   0.726
    1         496       136    0.963  0.946   0.986   0.815
    2         496        91    0.977  0.945   0.99    0.823
    3         496       161    0.887  0.826   0.908   0.696
Speed: 0.3ms preprocess, 4.9ms inference, 0.0ms loss, 0.1ms postprocess per image
```



1. mAP (Mean Average Precision)

- **AP (Average Precision)** là một chỉ số đo độ chính xác cho một lớp (loại đối tượng) duy nhất, đo bằng cách tính diện tích dưới đường Precision-Recall (P-R curve).
- **mAP** là trung bình của AP trên tất cả các lớp trong tập dữ liệu.

IoU (Intersection over Union) là tỉ lệ diện tích giao giữa dự đoán và nhãn gốc trên diện tích hợp của chúng, được sử dụng để quyết định xem dự đoán có chính xác hay không. Một dự đoán được coi là chính xác nếu IoU giữa hộp dự đoán và hộp nhãn gốc vượt ngưỡng xác định.

2. mAP50

- **mAP@50** là giá trị mAP khi ngưỡng IoU được cố định ở mức 0.5.
- Nghĩa là, một phát hiện được coi là đúng nếu **IoU \geq 0.5**. Sau đó, tính AP cho mỗi lớp và lấy trung bình, ta có mAP50.
- Đây là chỉ số dễ đạt và thường cao hơn vì chỉ yêu cầu một mức độ khớp tương đối giữa dự đoán và nhãn gốc.

3. mAP50-95

- **mAP50-95** là mAP trung bình trên các ngưỡng IoU từ 0.5 đến 0.95, với bước nhảy 0.05.
- Các ngưỡng này bao gồm 0.5, 0.55, 0.6, ..., 0.95, và mAP được tính trung bình trên tất cả các ngưỡng này.
- **mAP50-95** là chỉ số đánh giá khắt khe hơn so với mAP50, vì nó yêu cầu mô hình không chỉ phát hiện đúng mà còn phải phát hiện với độ chính xác cao hơn (các ngưỡng IoU cao như 0.75, 0.85, v.v.).

Từ kết quả trên cho thấy mô hình YOLO của chúng ta chạy cho kết quả thẩm định rất tốt

CHƯƠNG 5: ĐỊNH DẠNG ĐẦU VÀO TRƯỚC VÀ SAU

Thực hiện chạy local đơn giản 1,2 epoch để tìm các định dạng của ảnh

Truy cập vào model và tìm ra các kế thừa và viết lại model chèn thêm các file log

```
class YOLOv10DetectionPredictor(DetectionPredictor):
    def postprocess(self, preds, img, orig_imgs):
        logger.info(f"Preds type: {type(preds)}, shape: {preds.shape if hasattr(preds, 'shape') else 'N/A'}")

        if isinstance(preds, dict):
            preds = preds["one2one"]

        if isinstance(preds, (list, tuple)):
            preds = preds[0]

        if preds.shape[-1] == 6:
            pass
        else:
            preds = preds.transpose(-1, -2)

        # Ghi thông tin về 'preds' sau khi hoán vị
        logger.info(f"'preds' sau khi hoán vị: shape = {preds.shape}, phần tử đầu tiên = {preds[0].tolist() if preds.numel() > 0 else ''}")

        # Thực hiện hàm v10postprocess để tính toán bboxes, scores, và labels
        bboxes, scores, labels = ops.v10postprocess(preds, self.args.max_det, preds.shape[-1] - 4)

        # Ghi thông tin về 'bboxes', 'scores', và 'labels'
        logger.info(f"'bboxes': shape = {bboxes.shape}, phần tử đầu tiên = {bboxes[0].tolist() if bboxes.numel() > 0 else 'N/A'}")
        logger.info(f"'scores': shape = {scores.shape}, phần tử đầu tiên = {scores[0].tolist() if scores.numel() > 0 else 'N/A'}")
        logger.info(f"'labels': shape = {labels.shape}, phần tử đầu tiên = {labels[0].tolist() if labels.numel() > 0 else 'N/A'}")

        # Chuyển đổi định dạng bounding boxes từ xywh sang xyxy
        bboxes = ops.xywh2xyxy(bboxes)

        # Ghi thông tin về 'bboxes' sau khi chuyển đổi định dạng
        logger.info(f"'bboxes' sau khi chuyển đổi xyxy: shape = {bboxes.shape}, phần tử đầu tiên = {bboxes[0].tolist() if bboxes.numel() > 0 else 'N/A'}")

        preds = torch.cat([bboxes, scores.unsqueeze(-1), labels.unsqueeze(-1)], dim=-1)
```

Hình ảnh đoạn code đã được chỉnh sửa và chèn thêm file log

TÀI LIỆU THAM KHẢO

1 .YOLOv10: Real-Time End-to-End Object Detection

Ao Wang, Hui Chen, Lihao Liu, Kai Chen, Zijia Lin, Jungong Han, Guiguang Ding

2. CSPNet: A New Backbone that can Enhance Learning Capability of CNN

Chien-Yao Wang, Hong-Yuan Mark Liao, I-Hau Yeh, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh

3.Ultralytics YOLO Documents