ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN MÔN TƯ DUY TÍNH TOÁN

ĐỀ TÀI:

NHẬN DIỆN NGƯỜI ĐEO KHẨU TRANG Y TẾ

Giảng viên hướng dẫn: TS. Ngô Đức Thành

Lớp: CS117.L21

Nhóm thực hiện: Nhóm 12

Thành viên nhóm:

STT	MSSV	Họ Tên
1	19521388	Hoàng Tiến Dũng
2	19522515	Lê Dương Khánh Việt
3	19522485	Trương Minh Tuấn

TP. HỒ CHÍ MINH - 08/2021

LỜI CẢM ƠN

Sau quá trình học tập và rèn luyện tại Trường Đại học Công Nghệ Thông Tin, chúng em đã được trang bị các kiến thức cơ bản, các kỹ năng thực tế để có thể hoàn thành đồ án môn học của mình.

Chúng em xin gửi lời cảm ơn chân thành đến thầy **Ngô Đức Thành** đã tận tâm hướng dẫn, truyền đạt những kiến thức cũng như kinh nghiệm cho chúng em trong suốt thời gian học tập môn Tư duy tính toán.

Trong quá trình làm đồ án môn học, chúng em chắc chắn sẽ không tránh được những sai sót không đáng có. Mong nhận được sự góp ý cũng như kinh nghiệm quý báu của các các thầy để được hoàn thiện hơn và rút kinh nghiệm cho những môn học sau. Chúng em xin chân thành cảm ơn!

TP. Hồ Chí Minh, tháng 8 năm 2021.

MỤC LỤC

I. Giới thiệu

Hiện nay, dịch Covid-19 đang diễn biến phức tạp không chỉ ở nước ta mà còn trên toàn thế giới. Vì vậy, giãn cách xã hội và tránh tụ tập đông người là điều cần phải thực hiện trong mùa dịch này. Tuy nhiên, nhu cầu đi lại là điều không thể tránh khỏi. Để giúp việc kiểm soát được tốt hơn tại các siêu thị, cửa hàng, nhà hàng và những nơi công cộng nhóm chúng tôi đề xuất đề tài *"**Nhận diện người đeo khẩu trang y tế**"* vì muốn việc theo dõi, giám sát các đối tượng không tuân thủ đúng quy tắc phòng ngừa dịch bệnh, giảm được phần nào lo ngại mà không làm lây nhiễm dịch bệnh trong cộng đồng. Điều này cũng sẽ giúp các cơ quan tiết kiệm được thời gian cũng như giảm được khoảng cách tiếp xúc trong việc kiểm soát dịch bệnh.

II. Định nghĩa bài toán

1. Mục tiêu bài toán

Xây dựng mô hình phát hiện người đeo khẩu trang và không đeo khẩu trang tại các khu vực công cộng (nhà trường, siêu thị, xe bus,…) trong mùa dịch Covid-19.

2. Xác định yêu cầu bài toán

- Đầu vào của bài toán là ảnh có một hay nhiều khuôn mặt người.

- Đầu ra của bài toán là ảnh của đầu vào chứa bounding boxes và nhãn cho từng bounding box đó.

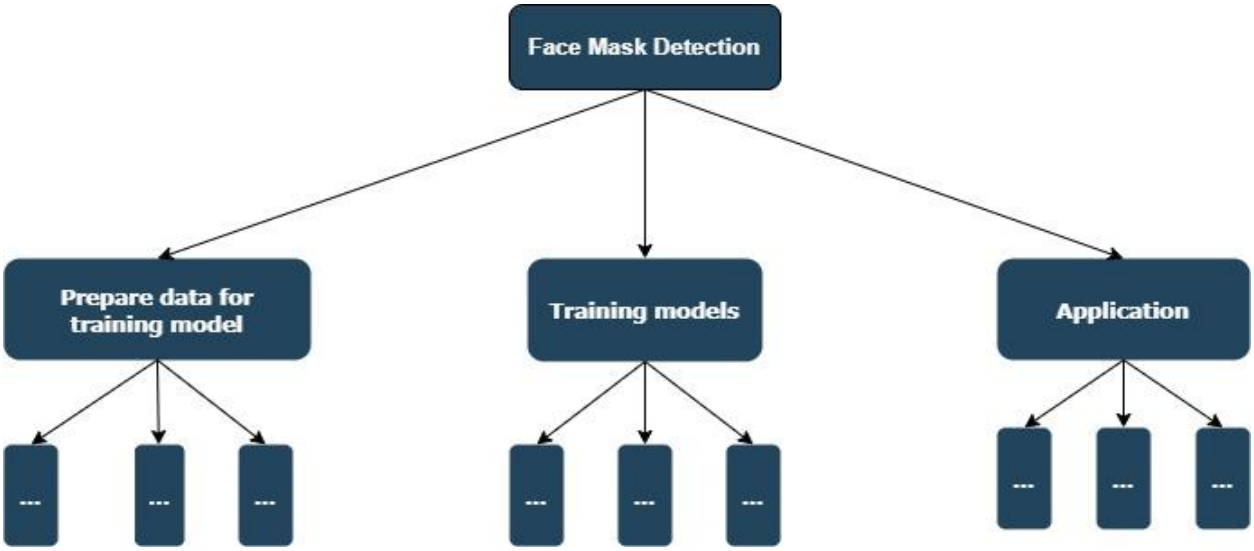
- Bài toán này thuộc bài toán Object Detection, có tổng cộng 2 classes:

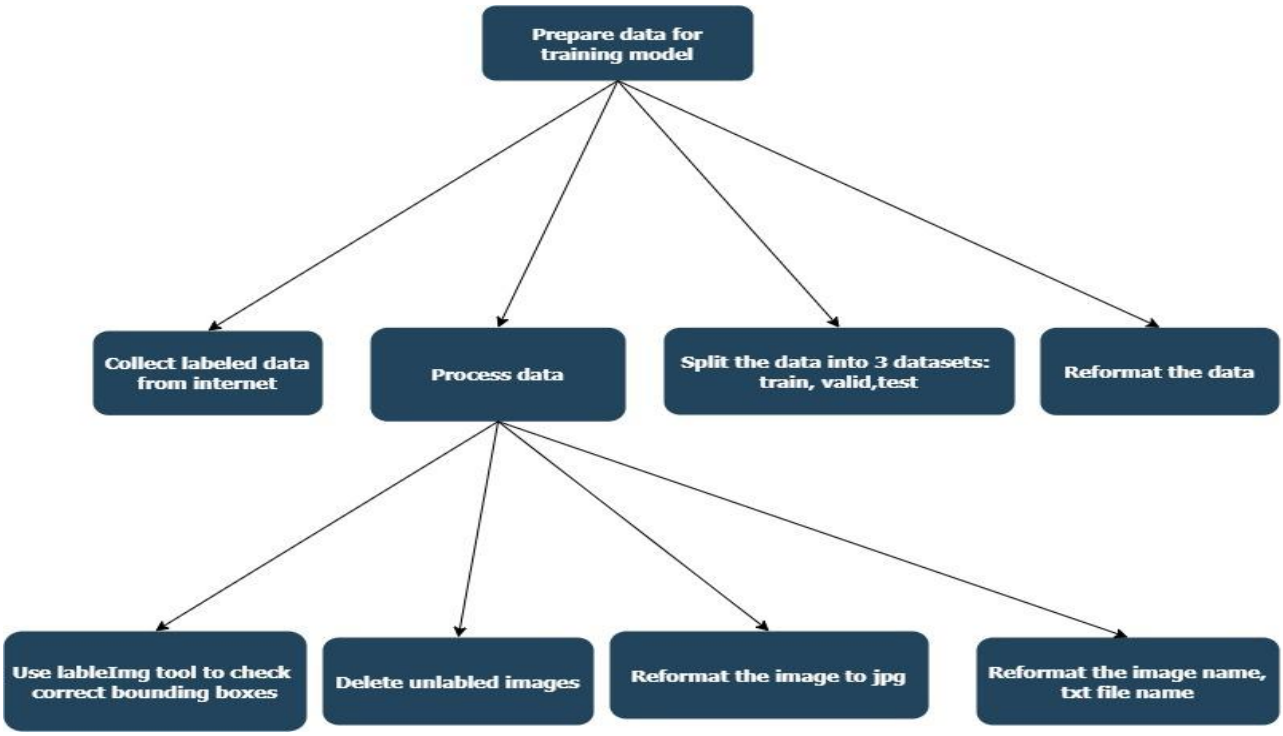
  - **no_mask** : Không đeo khẩu trang trên khuôn mặt.

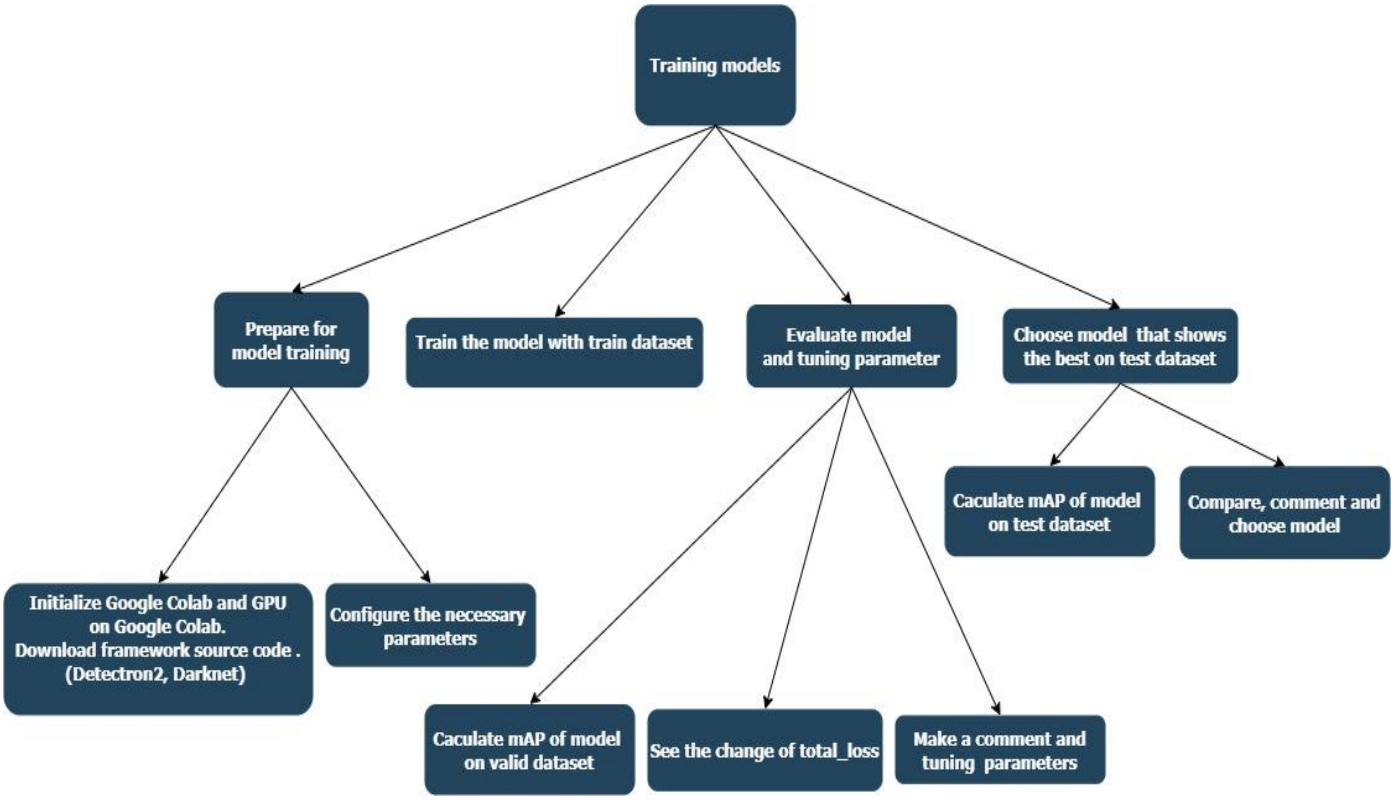
  - **mask** : Có đeo khẩu trang trên khuôn mặt.

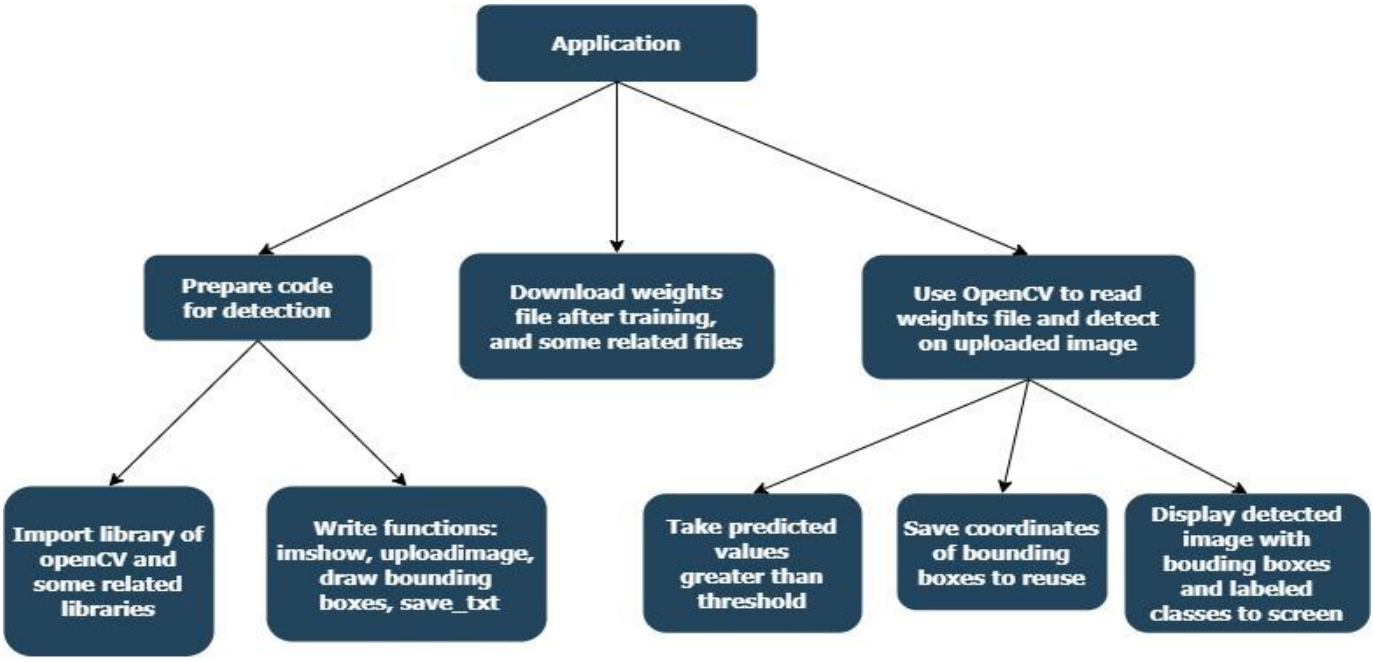
3. Hierarchical structure

Dưới đây, nhóm chúng tôi đã sử dụng hierarchical structure (cấu trúc phân cấp/cấu trúc cây) để mô tả mối liên hệ giữa các bài toán chính-bài toán con:






Training models
Prepare for model training
Train the model with train dataset
Evaluate model and tuning parameter
Choose model that shows the best on test dataset
Initialize Google Colab and GPU on Google Colab. Download framework source code . (Detectron2, Darknet)
Configure the necessary parameters
Caculate mAP of model on valid dataset
See the change of total_loss
Make a comment and tuning parameters
Caculate mAP of model on test dataset
Compare, comment and choose model


Application
Prepare code for detection
Download weights file after training, and some related files
Use OpenCV to read weights file and detect on uploaded image
Import library of openCV and some related libraries
Write functions: imshow, uploadimage, draw bounding boxes, save_txt
Take predicted values greater than threshold
Save coordinates of bounding boxes to reuse
Display detected image with bouding boxes and labeled classes to screen

4. Computational Thinking

Iteration 1:

Problem Identification

Trong tình hình dịch Covid-19 phức tạp như hiện nay, để kiểm soát việc tuân thủ đeo khẩu trang hiệu quả là điều hết sức cần thiết. Tôi muốn biết những người ở trong một bức ảnh đã đeo khẩu trang hay chưa đeo, nhưng chưa có một ứng dụng nào có chức năng phân loại này. Vậy, để giải quyết bài toán này, tôi cần phải xác định những gì?

Decomposition

Để giải quyết bài toán này, tôi cần làm rõ:

+ Input đầu vào của bài toán này là gì?

+ Input đầu vào có những yêu cầu gì đặc biệt?

+ Output là gì?

+ Căn cứ vào đâu để chọn ra phương pháp phù hợp nhất để giải quyết bài toán?

Pattern Recognition

Có thể nhận thấy điểm chung của bài toán này là phải phân loại những người trong bức ảnh thành hai loại:

+ Đeo khẩu trang

+ Không đeo khẩu trang

Iteration 2:

Problem Identification

Ở bài toán này, tôi đã xác định lại vấn đề chính của bài toán này là giải quyết một hình ảnh để biết những người bên trong ảnh đó có đeo khẩu trang hay không đeo khẩu trang. Nhưng làm sao chắc chắn hình ảnh được input là một hình ảnh sắc nét, không bị mờ. Tôi cần làm gì để giải quyết vấn đề này?

Decomposition

Để Input chắc chắn là một hình ảnh thõa mãn các yêu cầu của tôi, tôi muốn:

+ Đảm bảo đuôi của hình ảnh được input là dạng của hình ảnh (.png, .jpg ,….).

+ Hình ảnh phải sắc nét, rõ ràng không bị mờ, phải đủ ánh sáng tối thiểu.

Abstraction

Xác định input đầu vào chính xác là một hình ảnh.

Iteration 3:

Problem Identification

Có muôn vàn thuật toán về object detection, tôi sẽ chọn thuật toán nào để tối ưu hoá bài toán? Tôi nên chọn một thuật toán có độ chính xác cao hay chọn các thuật toán có tốc độ xử lí ảnh nhanh?

Decomposition

Để giải quyết vấn đề này, tôi cần làm rõ:

+ Những người đi trước đã dùng thuật toán gì để giải quyết bài toán này?

+ Những thuật toán đó nếu chạy trên cùng một dataset sẽ cho ra kết quả khác nhau như nào?

Abstraction

Xác định độ chính xác của từng thuật toán trên cùng một bộ dataset, thông qua đó chọn ra thuật toán tối ưu nhất cho bài toán, và ở bài toán này cuối cùng tôi đã chọn ra YOLOv4.

Iteration 4:

Problem Identification

Bộ dataset gồm 920 hình ảnh ở các định dạng jpg, jpeg, png. Vậy tôi cần xử lý data như thế nào để chuẩn bị cho Training Phase (giai đoạn cho thuật toán học data)?

Decomposition

Lúc này, tôi sẽ thực hiện theo các bước như sau:

+ Dùng tool labelImg để kiểm tra lại ảnh được gán nhãn.

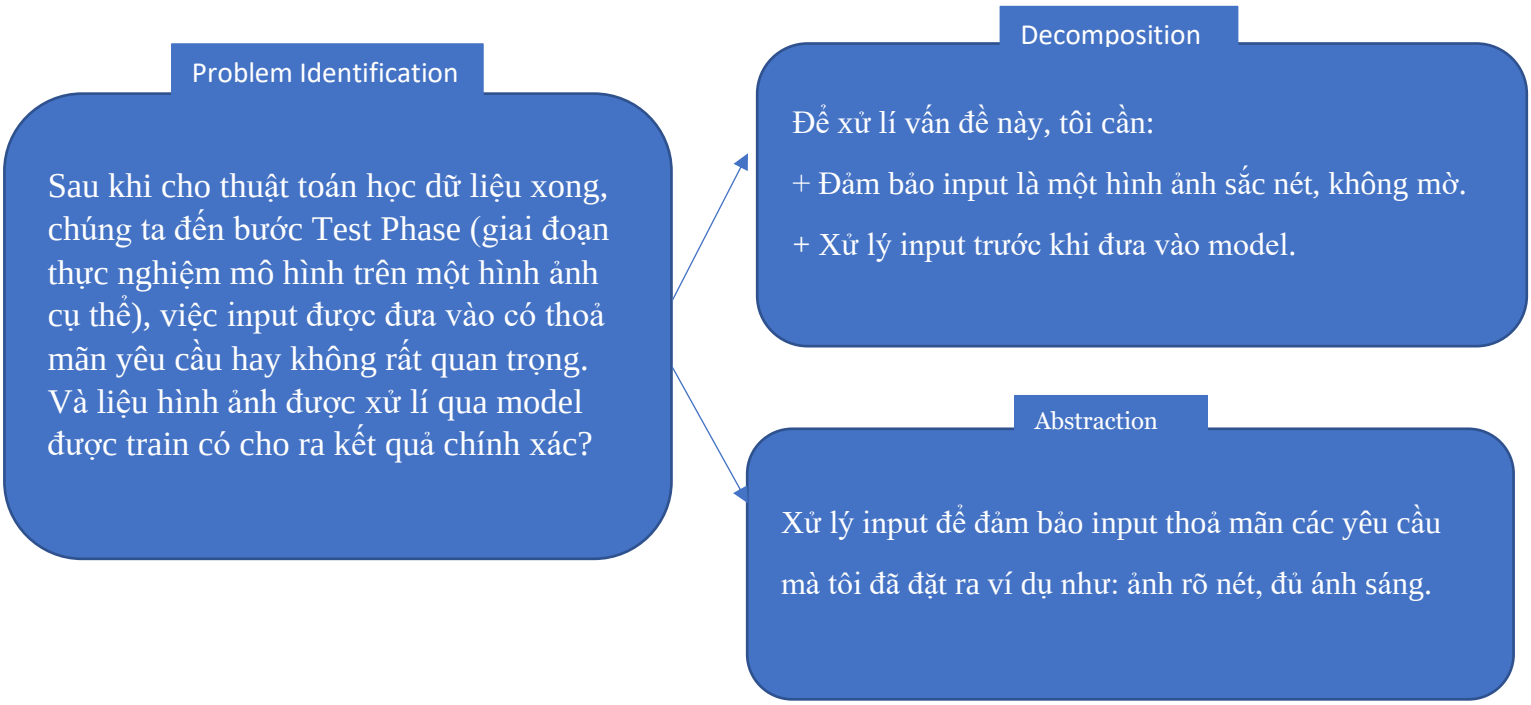
+ Xóa ảnh và file txt không được gán nhãn, chuyển tất cả ảnh về cùng một định dạng jpg, định dạng lại tên ảnh và file txt tương ứng.

+ Phân chia dữ liệu thành 3 tập dữ liệu: tập train, tập valid, tập test.

Abstraction

Dùng tool labelImg cho bước tiền xử lý dữ liệu, là một bước khá quan trọng, có thể ảnh hưởng đến tỷ lệ chính xác của thuật toán.

Iteration 5:



Problem Identification: For each iteration of your problem, please explain how you arrived at your identified problem

Ở Iteration 1, nhóm em đã đưa ra một câu hỏi có hàm ý bao quát khá lớn, thông qua đó đã đưa ra những vấn đề cần xác định rõ khi giải quyết bài toán này: Input đầu vào là gì? Input có yêu cầu gì đặt biệt? Output là gì? Căn cứ vào những đâu để chọn ra phương pháp phù hợp nhất để giải quyết bài toán?

Từ Iteration 2 trở đi là những vấn đề được mở rộng ra khi tìm hiểu sâu về bài toán và cả phương pháp để giải nó, qua đó chia nhỏ vấn đề thành những vấn đề nhỏ hơn có thể giải quyết được. Những Iteration này có được là nhờ vào quá trình nhận dạng mẫu chung được xác định ở Iteration 1.

Decomposition: For each iteration where you decomposed an identified problem, please explain how this decomposition helped you solve your identified problem

Giai đoạn Decomposition rất quan trọng trong cả 5 lần Iteration. Trong Iteration 1 nhóm decompose thành 4 vấn đề nhỏ hơn, lần lượt trong Iteration 2 là 2, trong Iteration 3 là 2, trong Iteration 4 là 3, trong Iteration 5 là 2. Nhờ đó tất cả những vấn đề con đó đã làm cho các vấn đề chính ngày càng rõ hơn, làm cho nhóm em có cái nhìn cụ thể hơn về bản chất của bài toán lớn, đặt ra những câu hỏi và đi đến cách giải quyết cụ thể giúp bài toán lớn được hoàn thiện hơn.

Pattern Recognition. For each iteration where you recognized patterns in data, please explain how these patterns helped you solve your identified problem.

Đối với đồ án này, nhóm em đã sử dụng giai đoạn nhận dạng mẫu ở Iteration 1. Đây là giai đoạn quan trọng giúp chúng em hiểu rõ được vấn đề, xác định được các vấn đề chính cần giải quyết, từ đó phân rã ra thành các vấn đề con nhỏ hơn, sau đó nhóm em chỉ việc giải quyết được các vấn đề con này thì bài toán lớn cũng đồng thời được giải quyết.

Abstraction: For each iteration where you abstracted information, please explain how abstraction allowed you to solve your identified problem

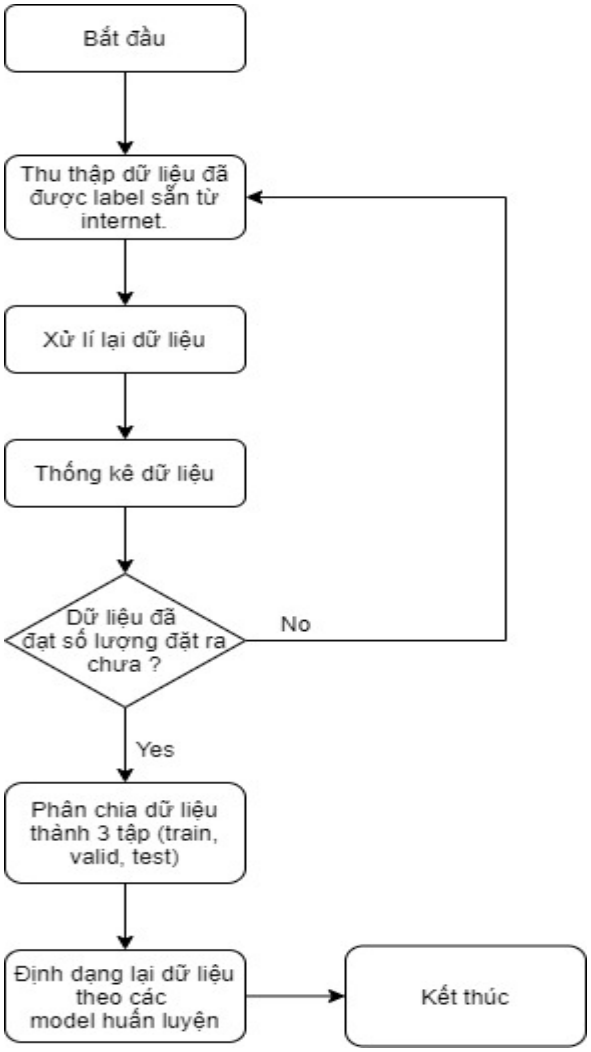
Các giai đoạn Abstraction được sử dụng trong hầu hết Iteration trừ Iteration 1. Các giai đoạn Abstraction đã giúp nhóm em chỉ tập trung vào các thông tin có liên quan và quan trọng nhất để giải quyết vấn đề, không tập trung quá nhiều vào các thông tin ít quan trọng hơn, tránh gây lãng phí thời gian.

5. Algorithms

a. Thuật toán thu thập và xử lý dữ liệu

Để thu thập dữ liệu, thay vì đi chụp và thu thập dữ liệu từ thực tế (cụ thể ở phần khó khăn), nhóm chúng tôi đã thực hiện thu thập dữ liệu đã được label sẵn từ internet, cụ thể là Kaggle: https://www.kaggle.com/aditya276/face-mask-dataset-yolo-format .

Dữ liệu được thu thập gồm 920 ảnh ở các định dạng jpg, jpeg, png và 920 tập txt. Nhóm đã thực hiện xử lí dữ liệu như sau:



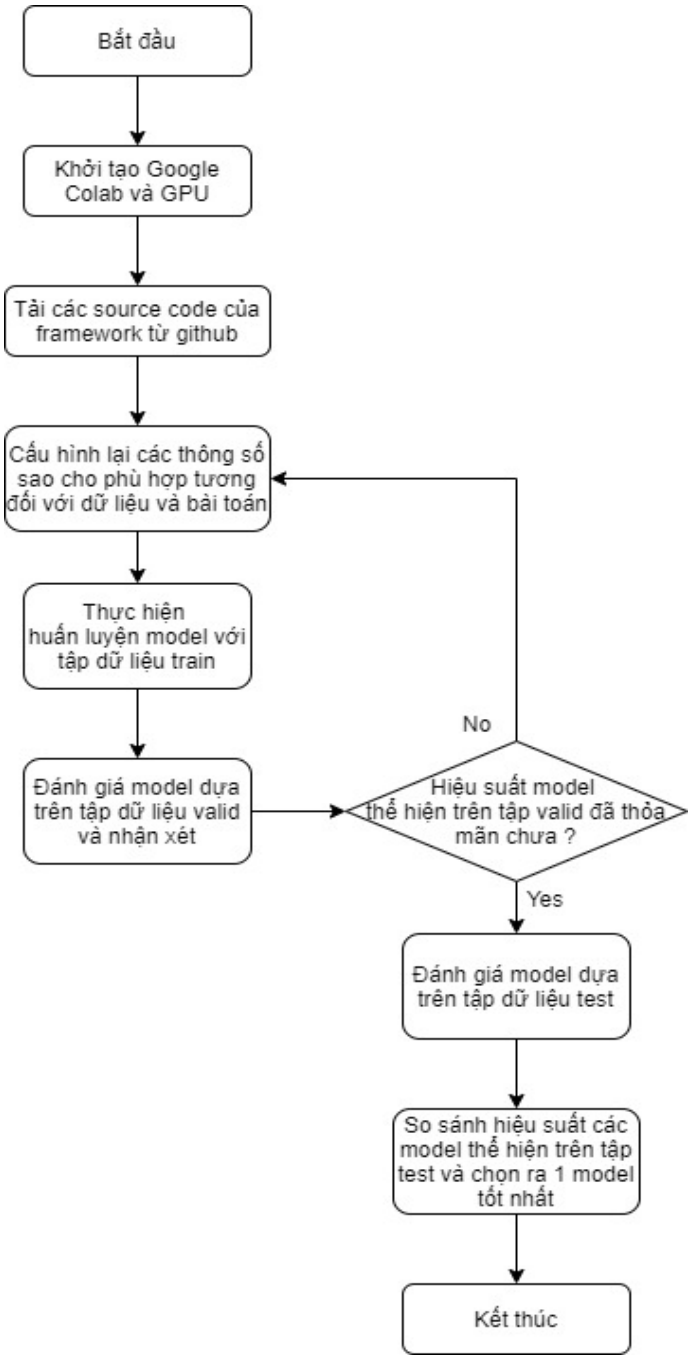
- Bước 1: Dùng tool labelImg để kiểm tra lại ảnh có được gán nhãn đúng hay không? Nếu không thì chỉnh sửa.
- Bước 2: Xóa ảnh và file txt tương ứng nếu không được gán nhãn (tức là file txt trống).
- Bước 3: Chuyển tất cả ảnh về cùng một định dạng jpg để thuận tiện cho việc xử lí và đánh giá sau này.
- Bước 4: Định dạng lại tên ảnh và file txt tương ứng: không có dấu chấm, không có dấu cách vì có thể sẽ gây ra một số lỗi khi huấn luyện hoặc đánh giá.
- Bước 5: Thống kê dữ liệu, nếu chưa đạt số lượng thì quay lại Bước 1.
- Bước 6: Phân chia dữ liệu thành 3 tập dữ liệu:
    - Tập train: Phục vụ quá trình huấn luyện.
    - Tập valid: Phục vụ quá trình kiểm thử độ chính xác.

- Tập test: Phục vụ quá trình thử nghiệm cuối cùng.

❖ Bước 7: Nhóm đã thực hiện train bằng Yolov4 (darknet), RetinaNet, Faster R-CNN (detectron2) nên cần format lại dữ liệu về 2 dạng là YOLO txt và COCO json.

b. Thuật toán huấn luyện mô hình

Google Colab (Google Colaboratory) là một dịch vụ miễn phí của Google nhằm hỗ trợ nghiên cứu và học tập về AI. Colaboratory cung cấp môi trường Code có thể sử dụng GPU và TPU miễn phí. Google Colab cung cấp cho chúng ta một cấu hình khá khủng đối với sinh viên cũng như các nhà nghiên cứu về AI tại thời điểm hiện tại. Do đó, nhóm tôi đã chọn Google Colab làm môi trường để huấn luyện model.

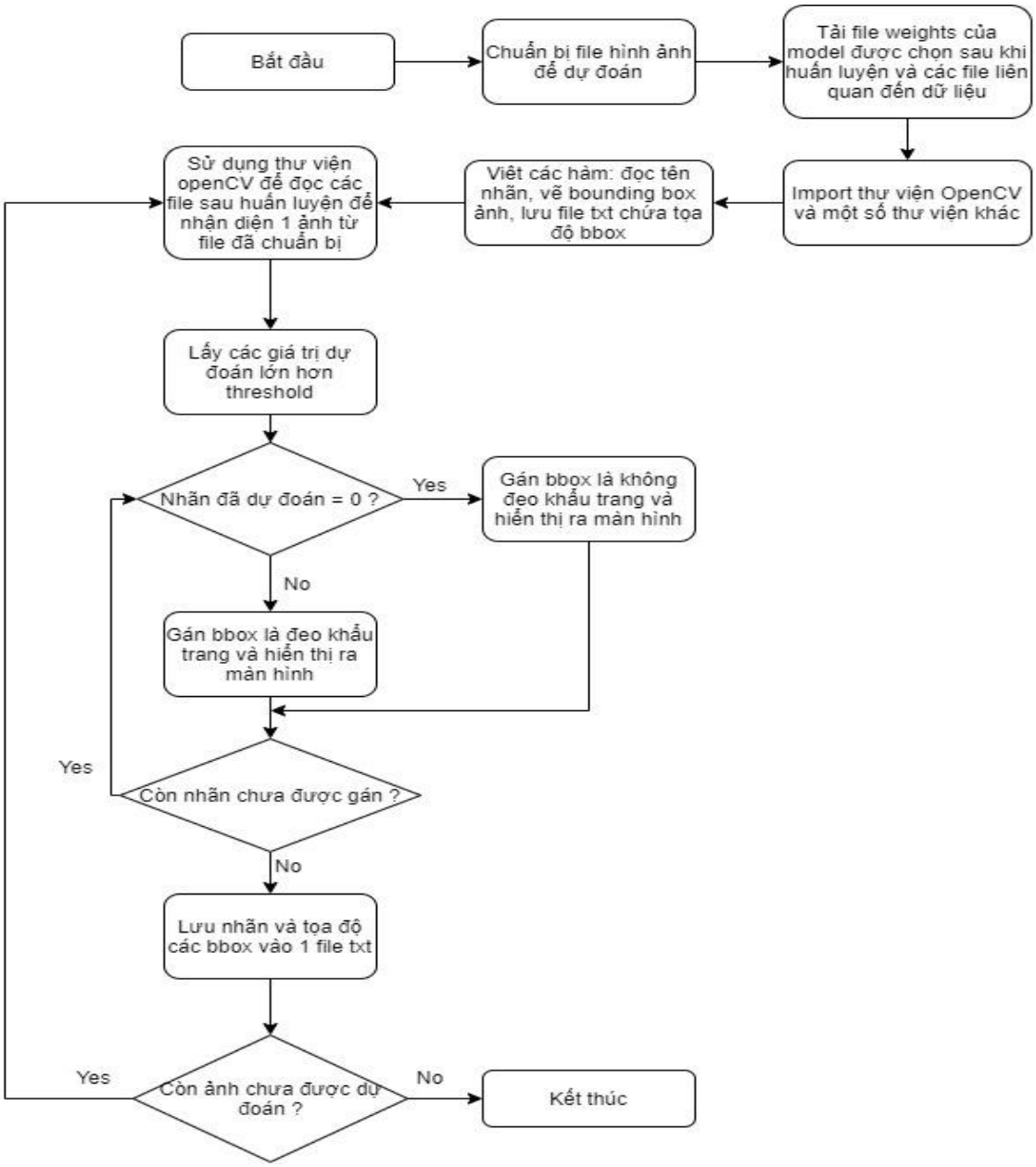


Khởi tạo Google Colab và GPU trên Google Colab. Sau đó, tải các source code của các framework, cụ thể là Darknet để huấn luyện Yolov4 và Detectron2 để huấn luyện RetinaNet và Faster R-CNN. Sau đó cấu hình lại các thông số, việc tinh chỉnh thông số sẽ nói cụ thể ở phần huấn luyện mô hình.

Thực hiện huấn luyện model với đầu vào là tập train. Việc đánh giá một mô hình theo nhóm là dựa vào chỉ số mAP của mô hình trên tập valid. Nhóm quyết định sẽ dừng train khi chỉ số loss có dấu hiệu không giảm. Sau đó lấy mô hình tại thời điểm có hiệu suất tốt nhất trên tập valid để đánh giá trên tập test. Cuối cùng sẽ chọn ra mô hình của giải pháp thể hiện hiệu suất trên tập test tốt nhất để thực hiện ứng dụng.

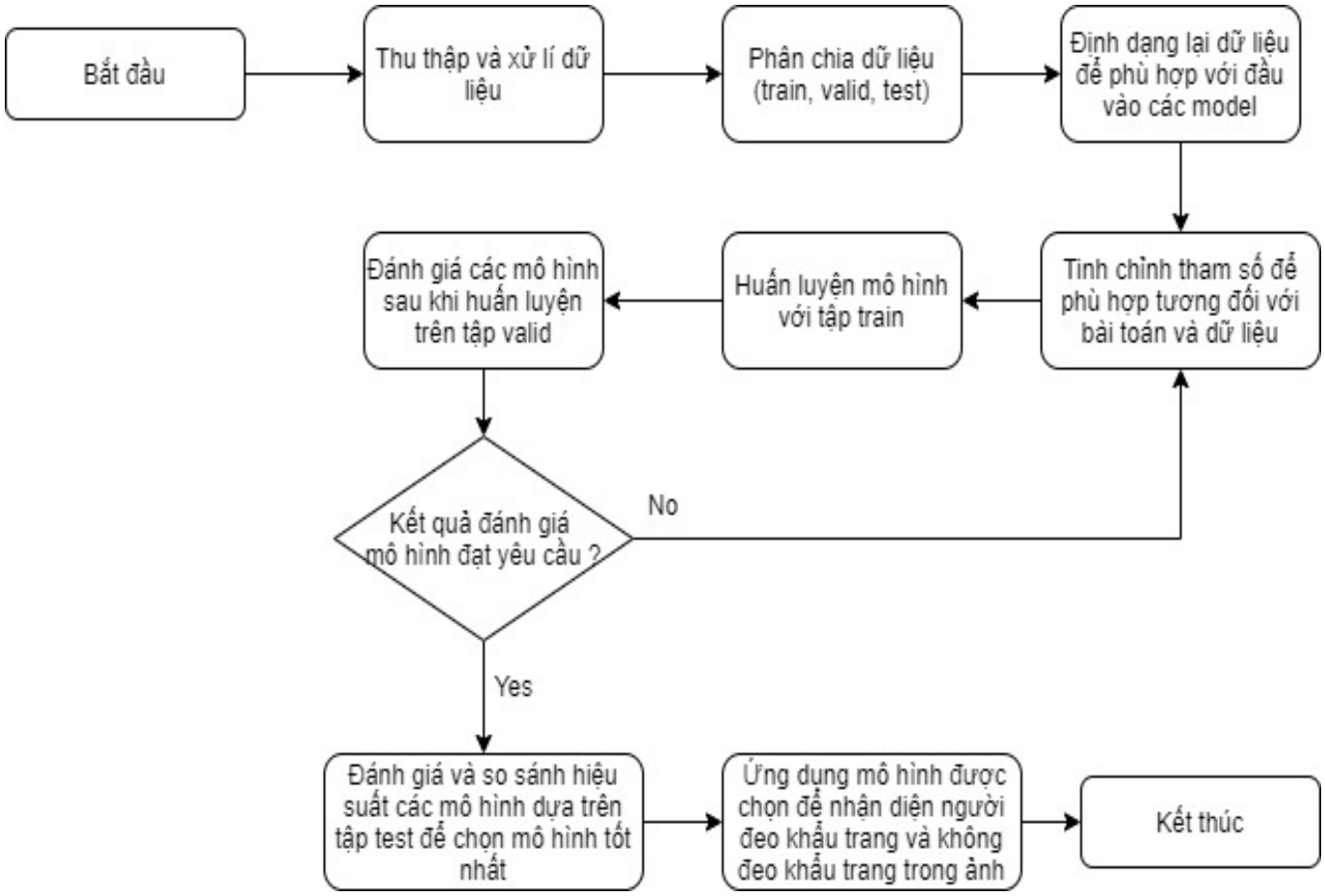
c. Thuật toán nhận diện người đeo và không đeo khẩu trang có đầu vào là ảnh

Thuật toán nhận diện người đeo và không đeo khẩu trang có đầu vào là một ảnh được nhóm thể hiện cụ thể và chi tiết qua flowchart sau:



d. Thuật toán cho bài toán nhận diện tổng thể

Từ các bài toán nhỏ được giải quyết trên, nhóm đã tổng hợp và đưa ra một thuật toán tổng thể cho bài toán nhận diện đeo khẩu trang như sau:



III. Thông tin dữ liệu

❖ **Thu thập dữ liệu:**

Dữ liệu thu thập là dữ liệu đã được label trước từ Kaggle (Link) và được xử lý theo thuật toán thu thập và xử lý dữ liệu ở trên.

❖ **Thống kê số liệu:**

- Tổng số lượng ảnh: 920 được chia thành tập train (chiếm 85%) và tập test (chiếm 15%), trong đó tập train được chia thành tập train (chiếm 87% tập train

cũ) và tập valid (chiếm 13% tập train cũ).

- Số lượng ảnh và bounding boxes cụ thể:

STT	Dataset	Amounts of images	Amounts of bounding boxes	
			no_mask	mask
1	Train	700	868	3047
2	Valid	100	49	278
3	Test	120	156	503
4	Total	920	1073	3828

❖ **Khó khăn trong việc thu thập và xử lý dữ liệu:**

- Vì tình hình dịch bệnh Covid-19 tại thời điểm làm đồ án, nên nhóm không thể ra ngoài và thu thập dữ liệu từ thực tế.
- Các bộ data có sẵn không đáp ứng về độ phong phú và đa dạng (trời nắng, trời mưa, khẩu trang có kiểu dáng đặc biệt, …).
- Mất cân bằng dữ liệu (imbalanced data): số lượng bounding boxes của nhãn no_mask còn ít hơn gấp 3 lần so với nhãn mask.
- Vì là bộ dữ liệu có sẵn nên cần phải kiểm tra tính chính xác của bounding boxes trước khi huấn luyện, việc kiểm tra được thực hiện bằng thủ công.

❖ **Nhận xét:**

- Với một số khó khăn nên nhóm đã quyết định thu thập dữ liệu có sẵn từ internet.
- Việc mất cân bằng dữ liệu gây ảnh hưởng đến việc huấn luyện và đánh giá mô hình sau này.

IV. Huấn luyện và đánh giá các mô hình

Đối với đồ án nhận diện đeo khẩu trang này chúng tôi thử nghiệm trên 3 thuật toán là **YOLOv4, RetinaNet, Faster R-CNN.** Sau đó so sánh các mô hình dựa vào chỉ số mAP được đánh giá trên tập test.

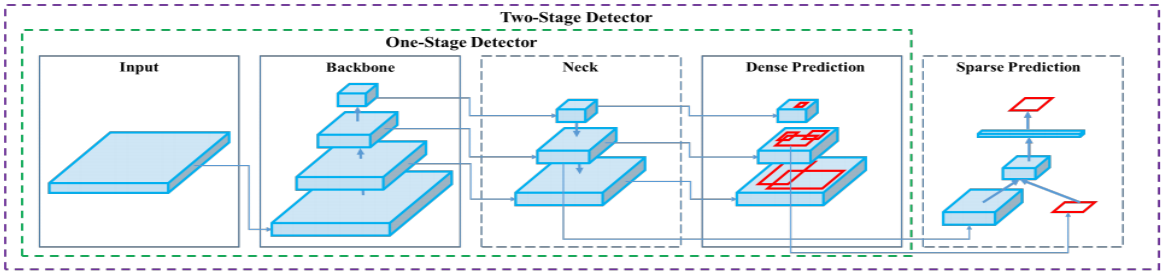
1. YOLOv4

❖ Kiến trúc mạng Yolov4:

Yolov4 là một loạt các cải tiến về tốc độ so với Yolov3 và được cài đặt từ một bản fork của Darknet. Kiến trúc của Yolov4 đã đưa bài toán object detection dễ tiếp cận hơn với những người không có tài nguyên tính toán mạnh. Chúng ta hoàn toàn có thể huấn luyện một mạng phát hiện vật với độ chính xác rất cao bằng Yolov4 chỉ với GPU 1080ti hoặc 2080ti. Trong tương lai, việc tối ưu lại các mạng hiện tại để phù hợp với tài nguyên tính toán yếu hoặc tạo ra sự song song hóa cao ở các server chắc chắn phải được thực hiện để có thể đưa các ứng dụng computer vision vào thực tế.

Cấu trúc của YOLOv4 được tác giả chia làm 4 phần:

- Backbone (xương sống) – trích xuất đặc trưng.
- Neck (cổ) – tổng hợp đặc trưng.
- Dense prediction (dự đoán dày đặc) – sử dụng các one-stage-detection như YOLO hoặc SSD.
- Sparse Prediction (dự đoán thưa thớt) – sử dụng các two-stage-detection như RCNN.



Input: { Image, Patches, Image Pyramid, … }

Backbone: { VGG16 [68], ResNet-50 [26], ResNeXt-101 [86], Darknet53 [63], … }

Neck: { FPN [44], PANet [49], Bi-FPN [77], … }

Head:

Dense Prediction: { RPN [64], YOLO [61, 62, 63], SSD [50], RetinaNet [45], FCOS [78], … }

Sparse Prediction: { Faster R-CNN [64], R-FCN [9], … }

Hình 5: Cấu trúc YOLOv4

❖ **Huấn luyện mô hình**

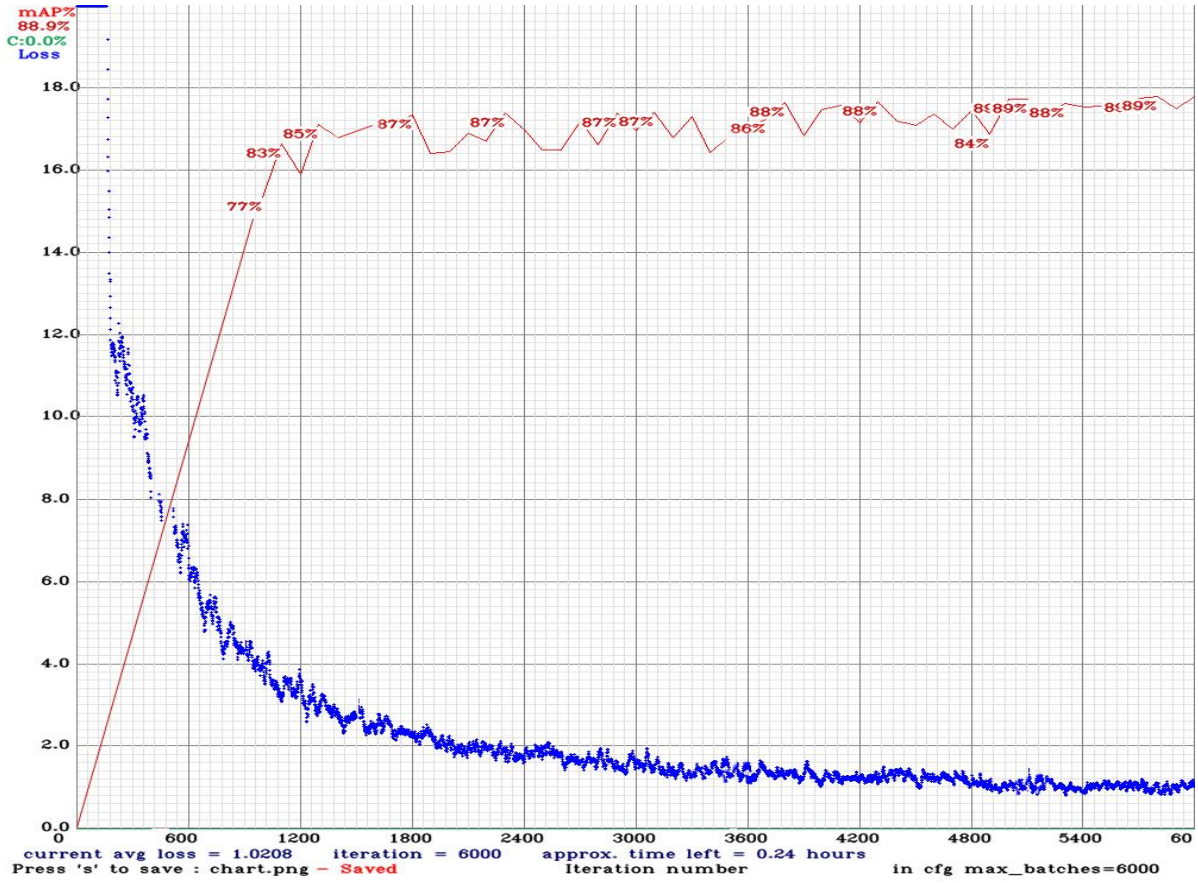
Đối với thuật toán YOLOv4 chúng tôi thực hiện huấn luyện mô hình trên framework Darknet.

❖ **Thiết lập Config**

Thực hiện training với các thông số config như sau:

- batch=64, subdivisions=32, width=416, height=416, learning_rate=0.00, max_batches = 6000, steps=4800,5400, filters=21, class=2

Trong quá trình huấn luyện, nhóm chúng tôi đã theo dõi độ lệch của total_loss, nếu thấy loss có thể giảm thì sẽ thử giảm learning_rate hoặc loss bị giao động ở một mức lớn quá lâu thì sẽ tăng learning_rate để hỗ trợ hàm loss có thể giảm. Việc theo dõi total_loss được thông qua chart mà framework đã hỗ trợ như hình bên dưới:

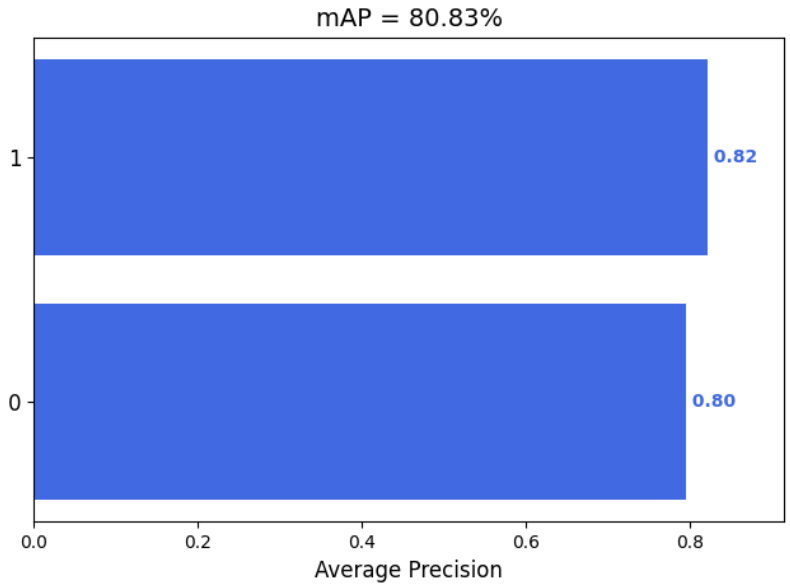


Hình 6: Quá trình huấn luyện model của YOLOv4.

Thời gian huấn luyện kéo dài từ 11-12 tiếng đến khi hàm model hội tụ thì có thể dừng train tránh mô hình bị overfit. Toàn bộ quá trình training đều được thực hiện trên GPU môi trường Google Colab.

❖ **Đánh giá mô hình:**

Sử dụng độ đo mAP để đánh giá mô hình trên tập dữ liệu test. Chỉ số mAP được tính theo Cartucho/mAP.



Hình 7: Độ đo mAP của mô hình YOLOv4.

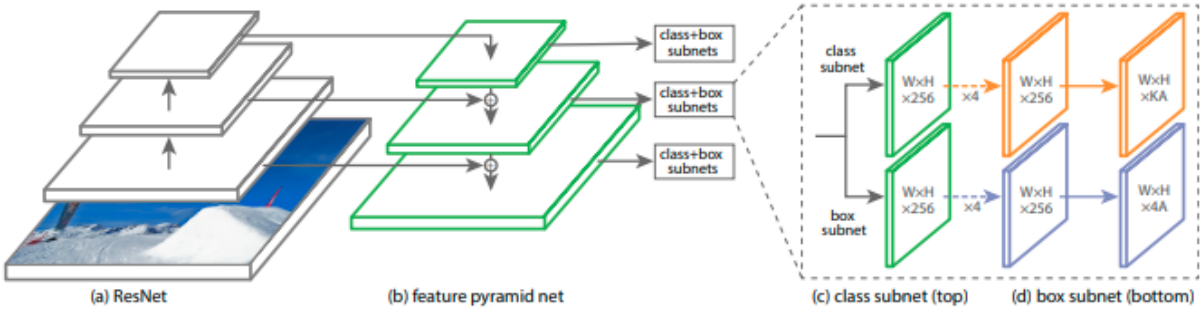
Trong đó:

1: **mask**

0: **no_mask**

2. RetinaNet

❖ **Kiến trúc mạng RetinaNet**



Hình 8: Kiến trúc của RetinaNet.

- Phase 1: là một feature extractor kết hợp giữa Resnet + FPN (Feature Pyramid Network), có tác dụng trích lọc đặc trưng và trả về các feature map. Mạng FPN sẽ tạo ra một multi-head dạng kim tự tháp.
- Nhánh phía trên là class subnet để dự báo phân phối xác suất của các classes.
- Nhánh phía dưới là box subnet dự báo tọa độ.

❖ **Huấn luyện mô hình**

- Thuật toán RetinaNet được nhóm thiết lập dựa trên framework Detectron2 của facebookresearch.
- Install framework Detectron2 và các thư viện cần thiết.
- Sử dụng dữ liệu đầu vào là dữ liệu ở định dạng COCO.

❖ **Thiết lập Config**

```
cfg = get_cfg()
cfg.merge_from_file(model_zoo.get_config_file("COCO-Detection/retinanet_R_50_FPN_3x.yaml"))

cfg.DATASETS.TRAIN = ("my_dataset_train",)
cfg.DATASETS.TEST = ("my_dataset_val",)
cfg.DATALOADER.NUM_WORKERS = 4
# cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-Detection/retinanet_R_50_FPN_3x.yaml")
cfg.MODEL.WEIGHTS = '/content/drive/MyDrive/CS117/RetinaNet/output/model_final.pth'
cfg.SOLVER.IMS_PER_BATCH = 2
cfg.SOLVER.BASE_LR = 0.0001  # pick a good LR
cfg.SOLVER.WARMUP_ITERS = 1000
cfg.SOLVER.MAX_ITER = 4800
cfg.SOLVER.STEPS = []
cfg.SOLVER.GAMMA = 0.05

cfg.MODEL.RETINANET.BATCH_SIZE_PER_IMAGE = 64
cfg.MODEL.RETINANET.NUM_CLASSES = 2
cfg.TEST.EVAL_PERIOD = 500
```

Thời gian thuật toán RetinaNet trên bộ dữ liệu train kéo dài gần 6 tiếng đến khi mô hình hội tụ thì dừng train để tránh tình trạng overfit của model. Toàn bộ quá trình training đều được thực hiện trên GPU môi trường Google Colab.

❖ **Đánh giá mô hình**

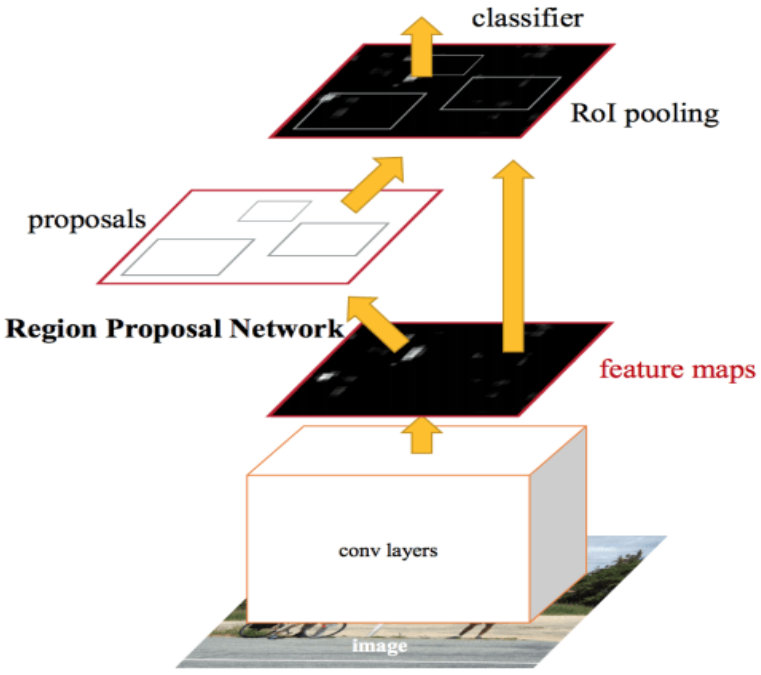
Đánh giá mô hình dựa trên mAP được tính bởi hàm COCOEvaluator trên tập dữ liệu test của framework Detectron2.

```
OrderedDict([('bbox',
              {'AP': 48.92663767004877,
               'AP-mask': 60.51832065762809,
               'AP-no_mask': 37.334954682469444,
               'AP50': 83.24154277126051,
               'AP75': 53.411517643225494,
               'APl': 61.07063163482379,
               'APm': 51.818119143019636,
               'APs': 38.54201914845387})])
```

3. Faster R-CNN

❖ Kiến trúc mạng Faster R-CNN

Faster R-CNN dùng một mạng CNN gọi là Region Proposal Network (RPN) để tìm các region proposal.

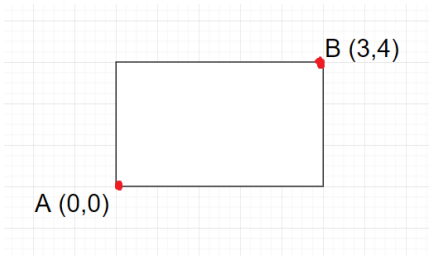


Hình 9: Kiến trúc Faster R-CNN.

Đầu tiên cả bức ảnh được cho qua pre-trained model để lấy feature map. Sau đó feature map được dùng cho Region Proposal Network để lấy được các region proposal. Sau khi lấy được vị trí các region proposal thì thực hiện tương tự Fast R-CNN.

❖ Region Proposal Network (RPN)

Input của RPN là feature map và ouput là các region propoasal. Ta thấy các region proposal là hình chữ nhật.



Mà một hình chữ nhật được xác định bằng 2 điểm ở 2 góc, ví dụ A (x_min, y_min) và B (x_max, y_max). Nhận xét:

- Khi RPN dự đoán ta phải ràng buộc x_min < x_max và y_min < y_max.
- Hơn nữa các giá trị x, y khi dự đoán có thể ra ngoài khỏi bức ảnh.

⇨ Cần một kỹ thuật mới để biểu diễn region proposal → **Anchor** ra đời. Có rất nhiều anchor bị chồng lên nhau nên **non-maxima suppression** được dùng để loại bỏ các anchor chồng lên nhau.

❖ **Huấn luyện mô hình**

- Thuật toán Faster R-CNN được nhóm thiết lập dựa trên framework Detectron2 của facebookresearch.
- Install framework Detectron2 và các thư viện cần thiết.
- Sử dụng dữ liệu đầu vào là dữ liệu ở định dạng COCO.

❖ **Thiết lập Config**

```
cfg = get_cfg()
cfg.merge_from_file(model_zoo.get_config_file("COCO-Detection/faster_rcnn_X_101_32x8d_FPN_3x.yaml"))

cfg.DATASETS.TRAIN = ("my_dataset_train",)
cfg.DATASETS.TEST = ("my_dataset_val",)
cfg.DATALOADER.NUM_WORKERS = 4
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-Detection/faster_rcnn_X_101_32x8d_FPN_3x.yaml")
# cfg.MODEL.WEIGHTS = '/content/drive/MyDrive/CS117/FasterRCNN/output/model_final.pth'
cfg.SOLVER.IMS_PER_BATCH = 2
cfg.SOLVER.BASE_LR = 0.0001  # pick a good LR
cfg.SOLVER.WARMUP_ITERS = 1000
cfg.SOLVER.MAX_ITER = 4800
cfg.SOLVER.STEPS = []
cfg.SOLVER.GAMMA = 0.05

cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 64
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 2
cfg.TEST.EVAL_PERIOD = 500
```

Thời gian thuật toán Faster R-CNN trên bộ dữ liệu train kéo dài gần 6 tiếng đến khi mô hình hội tụ thì dừng train để tránh tình trạng overfit của model. Toàn bộ quá trình training đều được thực hiện trên GPU môi trường Google Colab.

❖ **Đánh giá mô hình**

Đánh giá mô hình dựa trên mAP được tính bởi hàm COCOEvaluator trên tập dữ liệu test của framework Detectron2.

```
OrderedDict([('bbox',
               {'AP': 47.52706870458732,
                'AP-mask': 56.27597239226444,
                'AP-no_mask': 38.7781650169102,
                'AP50': 87.17509253462696,
                'AP75': 46.94633157880541,
                'APl': 62.402722406708,
                'APm': 49.30863825665088,
                'APs': 39.60521774472268})])
```

4. Nhận xét và chọn mô hình tốt nhất

Qua phần đánh giá 3 mô hình trên, ta thấy YOLOv4 thể hiện tốt nhất trên tập dữ liệu test với chỉ số mAP bằng 80.83%. So với RetinaNet (mAP = 48.92%) và Faster-RCNN (mAP = 47.52%) thì YOLOv4 thể hiện tốt vượt bậc. Nhóm đang tìm hiểu tại sao lại có sự chênh lệch lớn như thế này. Có thể trong việc tinh chỉnh tham số có sự sai sót hoặc do sự chênh lệch dữ liệu giữa 2 class.

Tuy nhiên nhóm chúng tôi đã quyết định chọn mô hình của thuật toán YOLOv4 để thực hiện áp dụng cho bài toán tổng quát. Vì so với 2 thuật toán còn lại, YOLOv4 có tốc độ nhận diện nhanh hơn so với Faster R-CNN, RetinaNet giúp mở ra hướng phát triển trong tương lai. Điều này đã được nhận xét qua một số bài báo uy tín, ví dụ bài báo Đánh giá các phương pháp dự trên Deeplearning cho bài toán phát hiện Logo, bài viết Object detection: Speed and Accuracy.

V. Kết quả của áp dụng YOLOv4 vào bài toán tổng quát.

Sau đây là một số kết quả của áp dụng mô hình thuật toán YOLOv4 vào việc nhận diện người đeo khẩu trang với Input và Output đã được nêu rõ ở trên:

Input	Output
	
	

VI. Ứng dụng và hướng phát triển

1. Ứng dụng

Sau khi thành công nhận diện người đeo khẩu trang và không đeo khẩu trang qua mô hình YOLOv4, nhóm đã phát triển bài toán bài toán nhận diện người đeo khẩu trang real-time trên **video** và **camera trực tiếp** (webcam).

- Link video demo nhận diện qua hình ảnh (có thêm chức năng nhận diện qua video, camera trực tiếp): https://www.youtube.com/watch?v=BJHamzmMlXs

- Nhận diện người đeo khẩu trang real-time qua video:

https://youtu.be/X8i2oyq6Alw

Link Google Drive của nhóm:

https://drive.google.com/drive/u/2/folders/1A5FuNQ6DfCng0mmx--uly90KuknjIizg

2. Hướng phát triển

Trong tương lai, chúng tôi có thể sẽ nghiên cứu cải tiến phương pháp để giải quyết các trường hợp khó nhận diện như hình dáng khẩu trang đặc biệt, khuôn mặt người khi ở xa, khi quay trái hoặc quay phải, khi bị che khuất một phần. Đồng thời nghiên cứu tìm ra giải pháp, tăng số lượng mẫu huấn luyện và kiểm tra để nâng cao độ chính xác của hệ thống. Mở rộng hệ thống để phát hiện và nhận dạng người đeo, đeo sai cách và không đeo để người quản lý có thể đưa ra quyết định phạt đúng đắn và răn đe hơn.

Sau đó có thể thử nghiệm tại nơi công cộng như xe buýt, chợ, siêu thị, trường học để xem xét hiệu quả thực tế mà model mang lại để có hướng phát triển tiếp theo.

VII. Tài liệu tham khảo

[1] https://phamdinhkhanh.github.io/2020/03/10/DarknetGoogleColab.html

[2]https://aicurious.io/posts/tim-hieu-yolo-cho-phat-hien-vat-tu-v1-den-v5/?fbclid=IwAR2FDNikgcqvg4zz1_Z4agNYyIc5I8MbCUMESSkn44CCtDmdO5eTeXKX8og

[3] Problem Solving Using Computational Thinking Course by university of Michigan (https://www.coursera.org/learn/compthinking)

[4] Computational Thinking for Problem Solving Course by university of Pennsylvania (https://www.coursera.org/learn/computational-thinking-problem-solving)

[5] https://nttuan8.com/bai-11-object-detection-voi-faster-r-cnn/

[6] https://phamdinhkhanh.github.io/2020/08/23/FocalLoss.html

[7] https://devai.info/2021/02/24/series-yolo-4-tim-hieu-cau-truc-yolov1v2v3-va-v4-phan-2/

[8] https://devai.info/2020/12/15/huong-dan-training-object-detection-voi-yolov4-su-dung-google-colab/

VIII. Bảng phân công công việc

BẢNG PHÂN CÔNG CÔNG VIỆC

STT	MSSV	Công việc được giao	Mức độ hoành thành (%)
1	19521388	Viết code face mask detection, huấn luyện và đánh giá mô hình, hỗ trợ viết báo cáo	100%
2	19522515	Vẽ Algorithm, hỗ trợ train data và computational thinking, demo đồ án, hỗ trợ viết báo cáo	100%
3	19522485	Thu thập dữ liệu data, Hierarchical structure, computational thinking, hỗ trợ viết báo cáo	100%