

HỌC VIỆN CÔNG NGHỆ  
BƯU CHÍNH VIỄN THÔNG

Cơ sở TP.HCM



**BÁO CÁO KẾT THÚC HỌC PHẦN:**

**LẬP TRÌNH PYTHON**

*Giảng viên: Nguyễn Thị Tuyết Hải*



Họ Tên TV1: Nguyễn Ngọc Thanh Danh

MSSV TV1: N19DCCN027

Họ Tên TV2: Nguyễn Trung Nguyên

MSSV TV2: N19DCCN124

# BÁO CÁO ĐỒ ÁN JAVA MÔN LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG VỚI JAVA

## CHƯƠNG I: GIỚI THIỆU ĐỀ TÀI

### 1. Giới thiệu chung:

Việc đeo khẩu trang nơi công cộng đã góp phần hạn chế được sự lây lan của dịch bệnh Covid 19 trên toàn cầu. Giám sát người dân thực hiện đúng chủ trương, chính sách của Nhà nước trong việc đeo khẩu trang nơi công cộng hoàn toàn có thể được thực hiện một cách tự động. Chương trình triển khai trên ngôn ngữ Python và sử dụng một số thư viện mã nguồn mở như OpenCV, .... Mô hình được đào tạo đạt độ chính xác 98% khi tiến hành phát hiện người đeo khẩu trang trên tập dữ liệu thử nghiệm.

Để ngăn chặn tình trạng lây lan nhanh chóng của đại dịch, bên cạnh khuyến nghị mà WHO đưa ra về việc đeo khẩu trang ở nơi đông người, Chính phủ Việt Nam cũng đã yêu cầu người dân phải đeo khẩu trang nơi công cộng để hạn chế sự lây lan của dịch bệnh. Tuy nhiên, để giám sát người dân thực hiện theo đúng chỉ đạo của Chính phủ với những hình thức cũ là khá khó khăn và tốn kém vì thiếu nguồn lực. Nhằm hỗ trợ và nâng cao công tác giám sát và nhắc nhở người dân, nhóm tác giả đã xây dựng chương trình phát hiện người không đeo khẩu trang trong thời gian thực một cách tự động.

Với sự phát triển nhanh chóng của học máy, thị giác máy tính đã đạt được những tiến bộ đáng kể trong những năm gần đây về nhận dạng và phát hiện đối tượng. Một trong những ứng dụng trong tình hình dịch bệnh đang diễn biến hết sức phức tạp trên toàn thế giới đó là việc xây dựng những hệ thống nhằm giám sát việc đeo khẩu trang của mọi người.

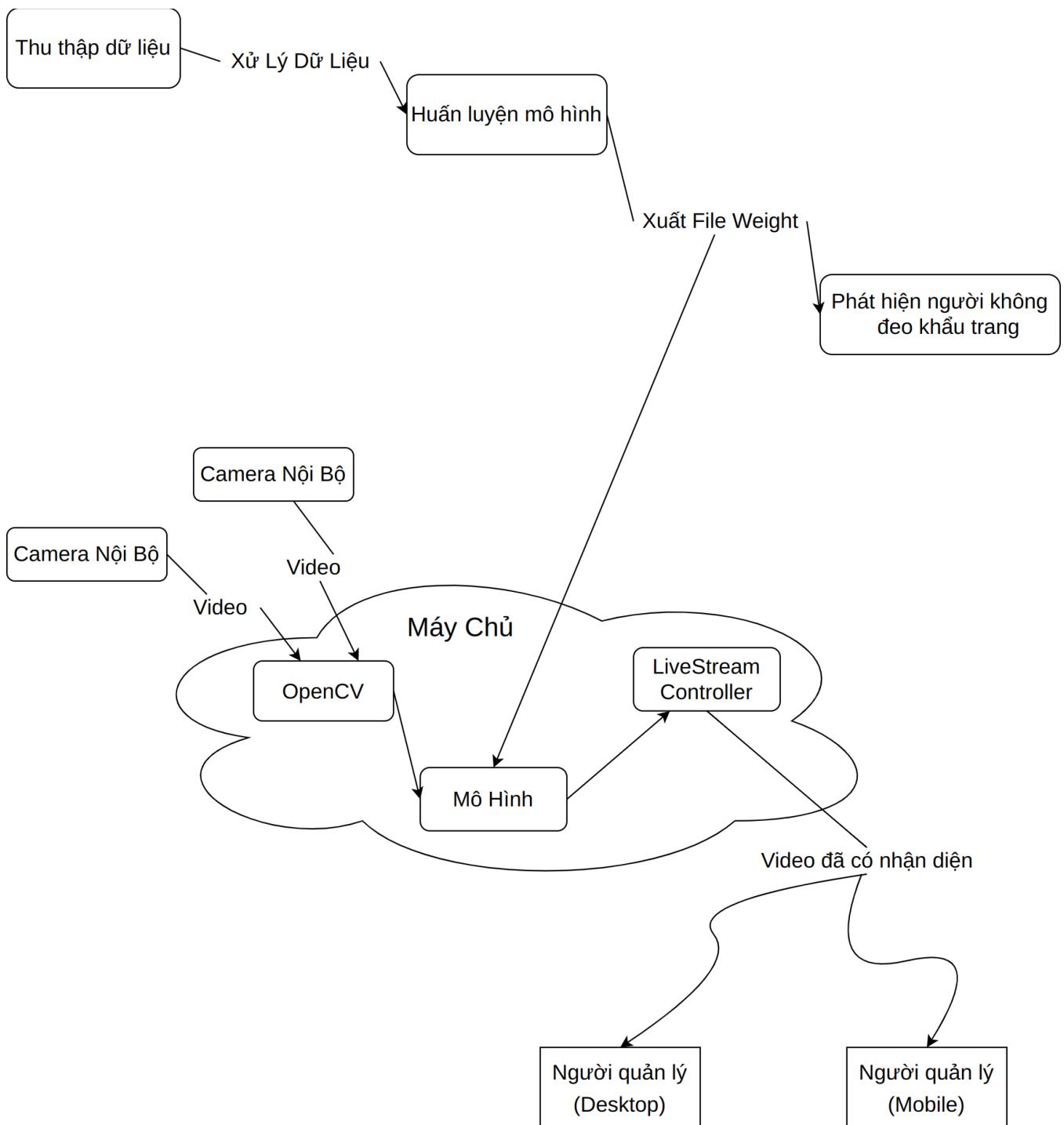
### 2. Mô tả đề tài:

Trong phạm vi nghiên cứu của bài báo cáo, nhóm chỉ nghiên cứu tiến hành phát hiện người không đeo khẩu trang.

- Cấu trúc chương trình gồm các bước:

- Thu thập dữ liệu
- Xử Lý Dữ Liệu
- Huấn luyện mô hình
- Phát hiện người không đeo khẩu trang

- Ngoài ra còn có các bước để triển khai một máy chủ:



### 3. Link Github:

- Đề tài
  - [Dashboard Mask Detector](#)

- Các training repo
  - [Train YOLOv5](#)
  - [Train Pre-trained MobileNetV2](#)

## CHƯƠNG II: CƠ SỞ LÝ THUYẾT

---

### 1. Nhận Diện Đeo Khẩu Trang Và Các Hướng Giải Quyết

---

#### A. Sử dụng mạng YOLOv5 (Trung Nguyên)

##### \* Phân Tích Vấn Đề Và Đưa Ra Giải Pháp

Các bước cần thực hiện là phân vùng khuôn mặt đồng thời phân loại có đeo khẩu trang hay không. Như vậy, input là một bức ảnh(1 frame của video) output sẽ cho ra các bouding box chứa khuôn mặt và kèm theo label classified như có đeo khẩu trang hay không. Ta sẽ sử dụng mạng YOLOv5 áp dụng cho use case này.

#### \* YOLOv5 Là Gì, Nó Có Ăn Được Không?

---

**YOLOv5** là một mạng *YOLO* được phát triển và sử dụng với *phiên bản lần thứ 5*. *YOLO* trong lĩnh vực *Computer Vision* là một mô hình thường được dùng để trích xuất các đặc tính nổi bật và nhận diện các đối tượng (như con người, các loài vật chó mèo,...) được kết hợp bởi hai layer phổ biến là *Convolutional Neural Networks*(aka CNN) và *Full Connected Layer*. *YOLO* với v5 thì được cải thiện tốc độ và độ chính xác với việc sử dụng framework *PyTorch* (của Facebook) thuận tiện hơn trong việc training (chỉ với vài dòng lệnh!).

##### \* Triển Khai Training

- Dataset: Được lấy từ [Bài Post](#) của bác [Andrew Maranhão](#)
- YOLO Model: *YOLO* có cung cấp sẵn cho chúng ta các model với các kích thước khác nhau ứng theo dung lượng và tốc độ xử lý như **S**, **M**, **L**, **X** với size **S** nhỏ nhất và nhanh nhất. Để có độ chính xác và tốc độ xử lý thì chúng em chọn size **M** (theo kinh nghiệm với hầu hết các tuto đa số là chọn size **M**).
- Classes:
  - without\_mask
  - with\_mask
  - mask\_weared\_incorrect
- Hyper-Parameter:
  - Batch: 16
  - Epochs: 100

=====

*Thông tin YOLO Model*

```
from n      params module  
arguments
```

0	-1	1	5280	models.common.Conv	
[3, 48, 6, 2, 2]	1	-1	1	41664	models.common.Conv
[48, 96, 3, 2]	2	-1	2	65280	models.common.C3
[96, 96, 2]	3	-1	1	166272	models.common.Conv
[96, 192, 3, 2]	4	-1	4	444672	models.common.C3
[192, 192, 4]	5	-1	1	664320	models.common.Conv
[192, 384, 3, 2]	6	-1	6	2512896	models.common.C3
[384, 384, 6]	7	-1	1	2655744	models.common.Conv
[384, 768, 3, 2]	8	-1	2	4134912	models.common.C3
[768, 768, 2]	9	-1	1	1476864	models.common.SPPF
[768, 768, 5]	10	-1	1	295680	models.common.Conv
[768, 384, 1, 1]	11	-1	1	0	torch.nn.modules.upsampling.Upsample
[None, 2, 'nearest']	12	[-1, 6]	1	0	models.common.Concat
[1]	13	-1	2	1182720	models.common.C3
[768, 384, 2, False]	14	-1	1	74112	models.common.Conv
[384, 192, 1, 1]	15	-1	1	0	torch.nn.modules.upsampling.Upsample
[None, 2, 'nearest']	16	[-1, 4]	1	0	models.common.Concat
[1]	17	-1	2	296448	models.common.C3
[384, 192, 2, False]	18	-1	1	332160	models.common.Conv
[192, 192, 3, 2]	19	[-1, 14]	1	0	models.common.Concat
[1]	20	-1	2	1035264	models.common.C3
[384, 384, 2, False]	21	-1	1	1327872	models.common.Conv
[384, 384, 3, 2]	22	[-1, 10]	1	0	models.common.Concat
[1]	23	-1	2	4134912	models.common.C3
[768, 768, 2, False]	24	[17, 20, 23]	1	32328	models.yolo.Detect
[3, [[10, 13, 16, 30, 33, 23], [30, 61, 62, 45, 59, 119], [116, 90, 156, 198, 373, 326]], [192, 384, 768]]					
Model Summary: 369 layers, 20879400 parameters, 20879400 gradients, 48.1 GFLOPs					

```

Transferred 475/481 items from yolov5m.pt
Scaled weight_decay = 0.0005
optimizer: SGD with parameter groups 79 weight, 82 weight (no decay), 82
bias
albumentations: version 1.0.3 required by YOLOv5, but version 0.1.12 is
currently installed
train: Scanning 'mask/train/labels' images and labels...765 found, 0
missing, 0 empty, 0 corrupted: 100% 765/765 [00:05<00:00, 152.58it/s]
train: New cache created: mask/train/labels.cache
val: Scanning 'mask/test/labels' images and labels...88 found, 0 missing, 0
empty, 0 corrupted: 100% 88/88 [00:00<00:00, 95.13it/s]
val: New cache created: mask/test/labels.cache
Plotting labels to runs/train/exp/labels.jpg...

```

### Sau khi train

	Model Summary: 290 layers, 20861016 parameters, 0 gradients, 48.0 GFLOPs	Class	Images	Labels	P	R	mAP@.5
	mAP@.5:.95: 100% 3/3 [00:05<00:00, 1.99s/it]						
0.6		all	88	396	0.881	0.84	0.845
0.591		without_mask	88	53	0.902	0.849	0.863
0.671		with_mask	88	331	0.924	0.921	0.945
mask_weared_incorrect			88	12	0.818	0.749	
0.726			0.539				

### Release v0.0.1

**Kết Quả: sau các loại tối ưu siêu tham số, mô hình có độ chính xác cao nhưng không thích hợp cho vấn đề realtime vì weight của model khá nặng, thực hiện nhiều phép tính nên có độ delay khá cao trên một frame (đâu đó tầm 2-3s) nên không phù hợp cho trường hợp này.**

## B. Kết Hợp Mạng Phân Loại Thông Thường Với Mạng MobileNetV2 (Thanh Danh)

### \* Phân Tích Vấn Đề Và Đưa Ra Giải Pháp

Đối với trường hợp delay ở giải pháp trên thì ta rút ra được là trong quá trình chọn model, có một vài tham số chọn không được tối ưu và thực hiện quá nhiều phép tính cho các công đoạn tạo phân tách khuôn mặt và phân loại khuôn mặt trong một pipeline, chúng em nghĩ đến việc tối ưu bằng cách tích hợp mạng phân vùng khuôn mặt với mạng phân loại phân loại riêng.

Các tiêu chuẩn để chọn mạng có sẵn

- Có kích thước nhỏ
- trained trên một tập dữ liệu lớn
- phù hợp cho các thiết bị có cấu hình yếu

**Sau quá trình xem xét chúng em quyết định tích hợp mạng trained MobileNetV2 vì mạng thường**

**được dùng cho các thiết bị điện thoại và file weight chúng em sử dụng chỉ có 11MB!**

Kỹ thuật integrate với mạng đã được train như thế này chúng em gọi là **Pre-Trained Model**.

#### \* Pre-Trained Model Là Gì, Nó Có Ăn Được Không?

- Trong quá trình thực chiến ta có thể đưa ra một nhận định như thế này: classification model thường sẽ có 3 phần: phần input(đầu vào tiếp nhận data), phần body(kiến trúc của model đó), phần cuối (làm phẳng thành ma trận 1 chiều, và có thể có một số Dense layer, và một layer chuẩn hóa đầu ra).
- lấy model A và model B & giả sử:
  - Model A được train với tập dữ liệu lớn cho ra file weight và sau nhiều lần kiểm tra thì có độ chính xác cao.
  - Model B cũng có một vài tác vụ như model A nhưng mở rộng hơn
  - Ta chỉnh sửa model A bằng cách loại bỏ phần cuối của model A và tích hợp với model B (bằng cách hiệu chỉnh ma trận đầu vào của model B sao cho khớp với phần ra của model A). Giải thích bằng lý thuyết, đối với một model đã được train thì ứng đối với mỗi layer của model đó đều có tra trận trọng số nhất định, khi ta loại bỏ một phần nào đó thì có nghĩa là ta loại bỏ layer và ma trận trọng số đó, trong quá trình training thì chúng ta "đóng băng" trọng số của các layer model đã được train trước đó và tiếp tục train cho các layer được tích hợp phía sau.
  - Ý nghĩa của việc tích hợp 2 model như thế này:
    - Tiết kiệm được thời gian train, vì model B sẽ ít phức tạp hơn, do một số tác vụ đã được thực hiện ở model A.
    - có thể tái sử dụng lại nhanh nếu ta muốn thực hiện những các vụ phức tạp hơn khi cần phải tách các model theo từng tính năng riêng.

#### \* MobileNetV2 Là Gì, Nó Có Ăn Được Không?

Theo tra cứu thì mạng MobileNetV2 được phát triển bởi đội ngũ Goole nhằm giảm kích thước của mô hình và giảm độ phức tạp các phép tính toán, được sử dụng nhiều cho các thiết bị như thiết bị di động, thiết bị nhúng.

#### \* Triển Khai Trainning

- Dataset: cũng sử dụng bộ data ở trên
- Model:
  - Ouput của MobileNetV2
  - AveragePooling2D(pool\_size=(7, 7))
  - Flatten
  - Dense(128, activation="relu")
  - Dropout(0.5)
  - Dense(2, activation="softmax")
- Classes(loại bỏ đi class đeo khẩu trang không đúng cách vì với một context khác thì việc đeo sai khá nhiều trường hợp):
  - without\_mask
  - with\_mask
- Hyper-Parameter:
  - Batch: 32
  - Epochs: 20

### Thông tin YOLO Model

```
=====
out_relu (ReLU)           (None, 7, 7, 1280)  0
['Conv_1_bn[0][0]']
từ phần này trở lên là kiến trúc của MobileNetV2
=====
=====
average_pooling2d (AveragePool (None, 1, 1, 1280)  0
['out_relu[0][0]']
    ing2D)

flatten (Flatten)         (None, 1280)        0
['average_pooling2d[0][0]']

dense (Dense)             (None, 128)          163968
['flatten[0][0]']

dropout (Dropout)          (None, 128)          0      [ 'dense[0]
[0]']

dense_1 (Dense)            (None, 2)            258
['dropout[0][0]']

=====
=====
Total params: 2,422,210
Trainable params: 2,388,098
Non-trainable params: 34,112
```

### Sau khi train

	precision	recall	f1-score	support
with_mask	0.97	1.00	0.99	433
without_mask	1.00	0.97	0.98	386
accuracy			0.98	819
macro avg	0.99	0.98	0.98	819
weighted avg	0.98	0.98	0.98	819

### Release v0.1.0

**Kết Quả:** Tốc độ xử lý khác nhanh, mặc dù có delay nhưng có thể chấp nhận được, hầu hết các tác vụ nhận diện khuôn mặt đều được MobileNetV2 xử lý, còn lại chỉ còn việc nhận diện có đeo hay không là của model cần train

## 2. Xây Dựng Hệ Thống Backend(Serverside)

---

### A. Máy chủ

#### Công Nghệ

Trong các framework hỗ trợ lập trình Web, chúng em, quyết định chọn Django vì hắn là framework giúp phát triển nhanh một kiến trúc MVC mà không tốn nhiều sức.

#### Nguyên lý hoạt động

- Input là video sẽ lấy từ các camera kết nối vào máy chủ  
hệ thống sẽ phân tách video thành từng frame (frame được hiểu như là một bức ảnh)
- các bức ảnh sẽ đi qua mô hình đã được train trước đó và trả về một bức hình có kết quả
- máy chủ sẽ chuyển bức ảnh đó về dạng bytes và stream về client
- một tính năng nữa là client có thể chọn được loại camera để theo dõi

### B. Kết nối mô hình được train máy chủ

Chúng em sẽ triển khai một *interface* cho *adapter class* của mô hình đó với ý tưởng như sau:

- Adapter class đó được kết thừa từ camera
  - function sẽ gồm có:
    - chỉ định cam mặc định
    - lấy frame từ cam đó
    - xử lý bức ảnh khi ta đã xác định được kết quả(như là: điền title object đã phát hiện được, ...)
    - function adapt với model đã train trước đó, với inp là ảnh, output là vị trí của khuôn mặt và có đeo khuấu trang hay không.
- Việc triển khai **Adapter** như thế này sẽ giúp ta tiết kiệm được nhiều thời gian hơn khi ta thay thế bằng một kiến trúc của mô hình khác, áp dụng một trong các design pattern phổ biến để có thể theo dõi, sử chữa, và bảo trì dễ dàng hơn,

## CHƯƠNG III: THỰC NGHIỆM

---

### Cài đặt và khởi chạy máy chủ

- Hệ điều hành windows

```
# install dependencies
pip install -r requirements.txt

# run
python ./manage.py runserver 0.0.0.0:3000
```

- Hệ điều hành Unix(MacOS, Linux)

make

```
#####
# install dependencies
#####
pip install -r requirements.txt
Requirement already satisfied: Django in /home/shane/.local/lib/python3.8/site-packages (from -r requirements.txt (line 1)) (4.0.2)
Requirement already satisfied: Markdown==3.3.6 in /home/shane/.local/lib/python3.8/site-packages (from -r requirements.txt (line 2)) (3.3.6)
Requirement already satisfied: Pillow==9.0.0 in /home/shane/.local/lib/python3.8/site-packages (from -r requirements.txt (line 3)) (9.0.0)
Requirement already satisfied: PyYAML==6.0 in /home/shane/.local/lib/python3.8/site-packages (from -r requirements.txt (line 4)) (6.0)
Requirement already satisfied: Werkzeug==2.0.2 in /home/shane/.local/lib/python3.8/site-packages (from -r requirements.txt (line 5)) (2.0.2)
Requirement already satisfied: absl-py==1.0.0 in /home/shane/.local/lib/python3.8/site-packages (from -r requirements.txt (line 6)) (1.0.0)
Requirement already satisfied: cachetools==4.2.4 in /home/shane/.local/lib/python3.8/site-packages (from -r requirements.txt (line 7)) (4.2.4)
Requirement already satisfied: certifi==2021.10.8 in /home/shane/.local/lib/python3.8/site-packages (from -r requirements.txt (line 8)) (2021.10.8)
Requirement already satisfied: charset-normalizer==2.0.9 in /home/shane/.local/lib/python3.8/site-packages (from -r requirements.txt (line 9)) (2.0.9)
Requirement already satisfied: cyclere==0.11.0 in /home/shane/.local/lib/python3.8/site-packages (from -r requirements.txt (line 10)) (0.11.0)

#####
# start staging server
#####
# python manage.py migrate
python ./manage.py runserver 0.0.0.0:3000
```

Performing system checks ...

```
System check identified no issues (0 silenced).
February 13, 2022 - 23:33:27
Django version 4.0.2, using settings 'monitor_dashboard.settings'
Starting development server at http://0.0.0.0:3000/
Quit the server with CONTROL-C.
```

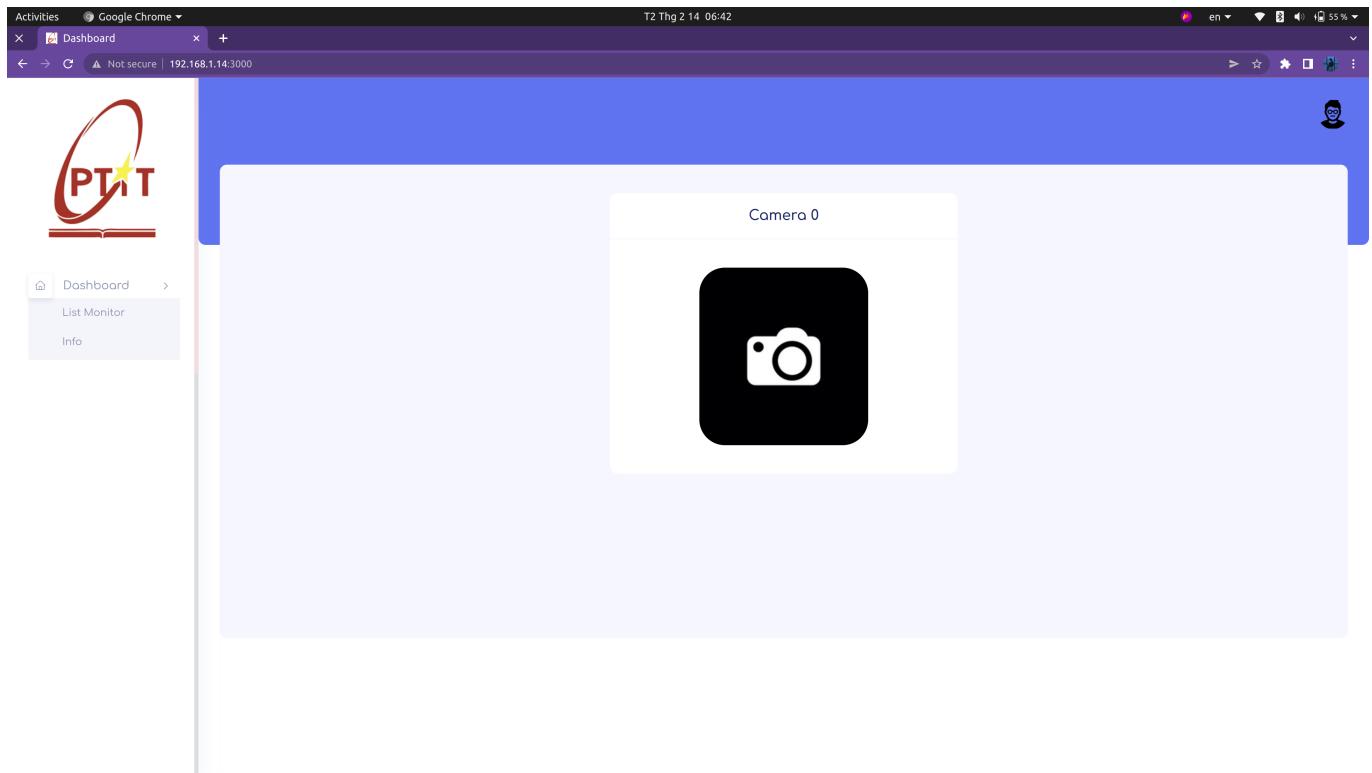
## Sử dụng chương trình

- Mở trình duyệt tại địa chỉ <http://127.0.0.1:3000> trên máy tính của bạn

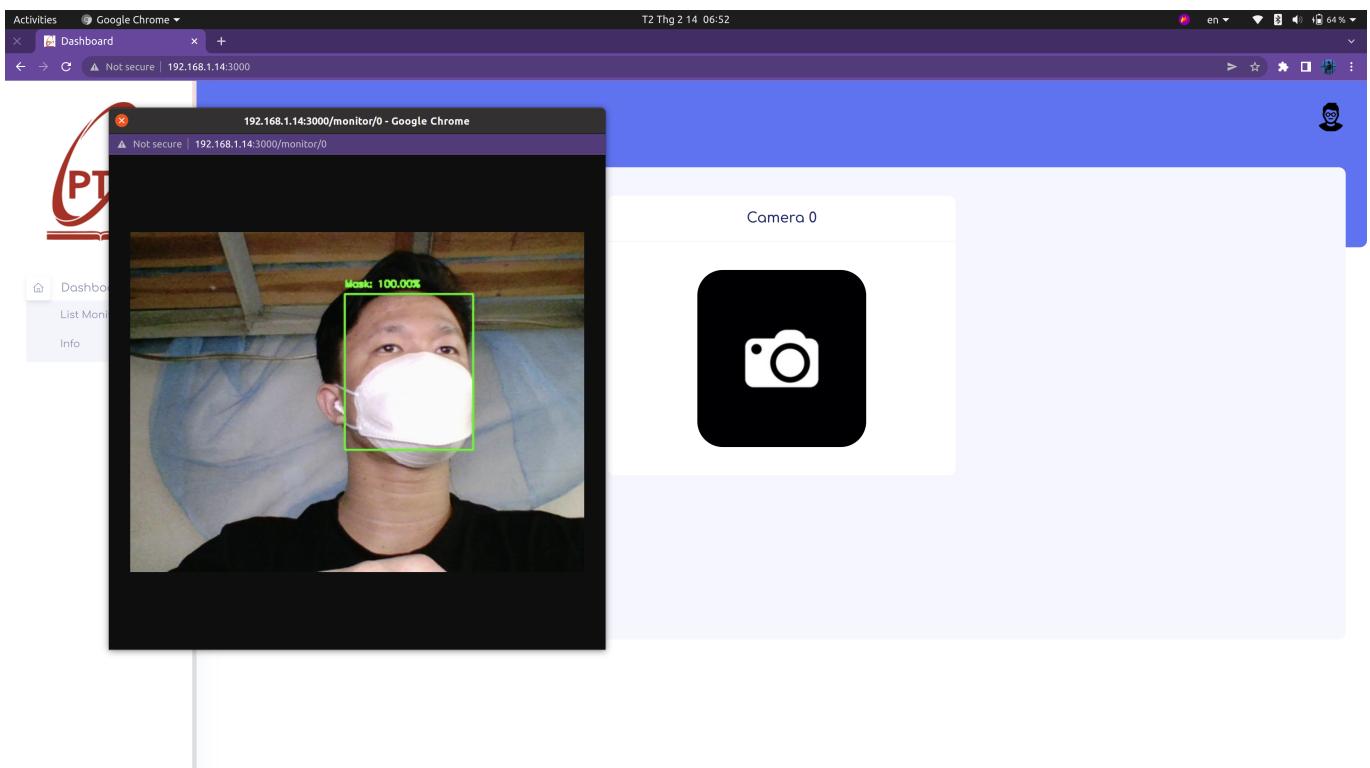
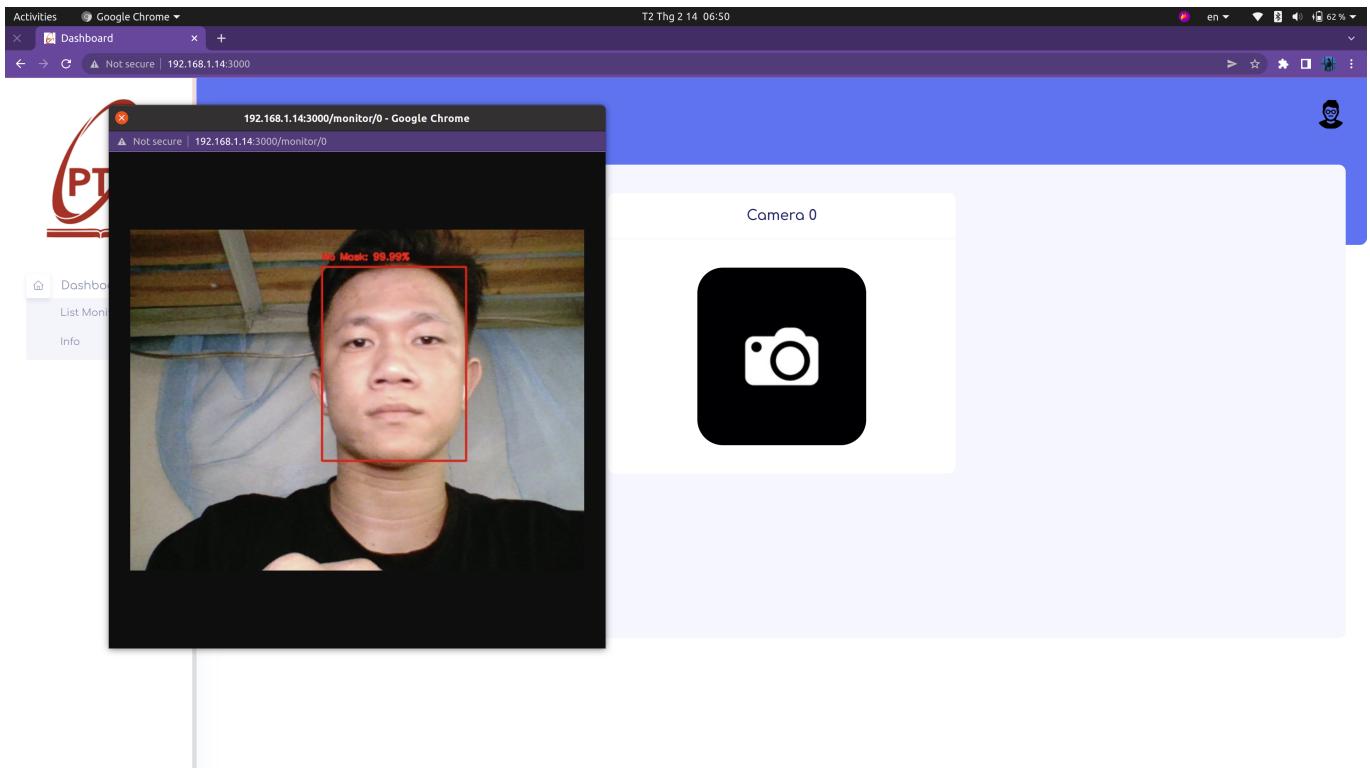
Nếu muốn cho các thiết bị trong mạng LAN có thể connect thì bạn nên kiểm tra địa chỉ mạng LAN bằng lệnh

- Windows: ipconfig
- Linux & MacOS: ifconfig

Và truy cập với địa chỉ đó và cổng :3000. Ex: <http://192.168.1.14:3000>



- Trang chủ sẽ liệt kê danh sách các Camera đã được kết nối với máy chủ. Ở hình trên thì nó đã tồn tại 1 Cam đã kết nối.
- Chọn Camera cần xem thì click vào tên của Camera đó, ví dụ **Camera 0**, trang web sẽ mở một cửa sổ riêng ra để quảng lý duy nhất Cam đó.



## Tài Liệu Tham Khảo

- Face Mask Detection in Real-Time using MobileNetv2
- How To Train Mobilenetv2 On A Custom Dataset
- Pre-trained models
- Yolo-v5