HƯỚNG DẪN TRAINING DỰ ÁN NHẬN DIỆN ĐEO KHẨU TRANG - SỬ DỤNG YOLOv4 (MÔI TRƯỜNG GOOGLE COLAB)

I. YÊU CẦU

- Tải dữ liệu cung cấp.
- Thiết bị webcam (nếu có).

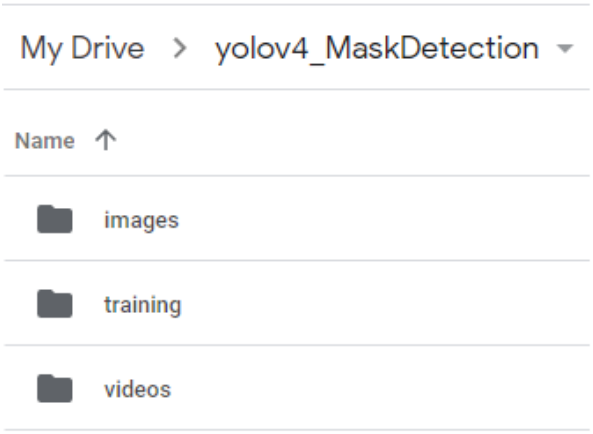
II. TẠO THƯ MỤC

1. Tạo thư mục gốc lưu trữ project làm việc

Tạo thư mục lưu trữ project trên Google Drive, trên hướng dẫn này sẽ đặt tên là: **yolov4_MaskDetection**

2. Tạo các thư mục con trong thư mục gốc

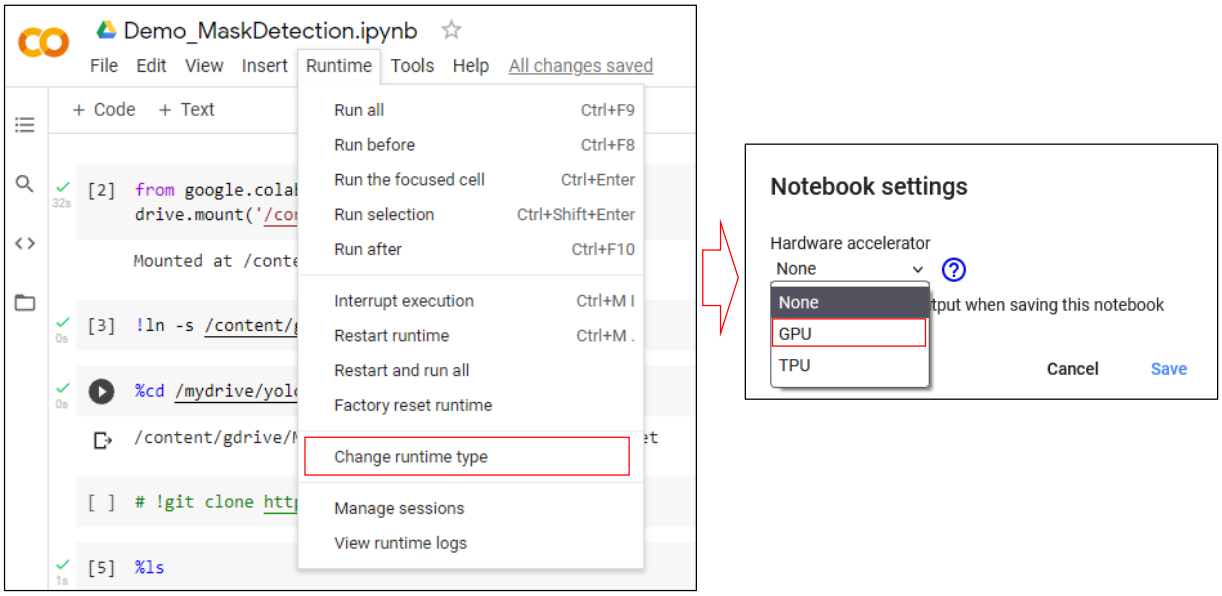
- training
- images
- videos



III. MOUNT Ổ ĐĨA

1. Thay đổi Runtime (Sử dụng GPU)

Từ **Menu** > **Runtime** > **Change runtime type** > **Hardware accelerator** > chọn **GPU**.



2. Mount ổ đĩa

Từ cell code của notebook, gõ lệnh:

```
from google.colab import drive
drive.mount('/content/drive')
```

3. Rút gọn đường dẫn

Thực hiện lệnh sau để đặt tên đại diện cho đường dẫn chỉ đến Google Drive, ở đây sẽ dùng tên đại diện là **mydrive**:

```
!ln -s /content/gdrive/My\ Drive/ /mydrive
```

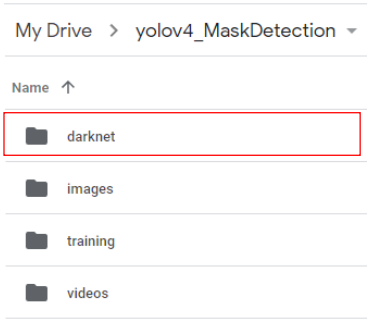
IV. CLONE DARKNET TỪ GITHUB

1. Di chuyển đến thư mục làm việc

```
%cd /mydrive/yolov4_MaskDetection
```

2. Clone darknet từ github:

```
!git clone https://github.com/AlexeyAB/darknet
```

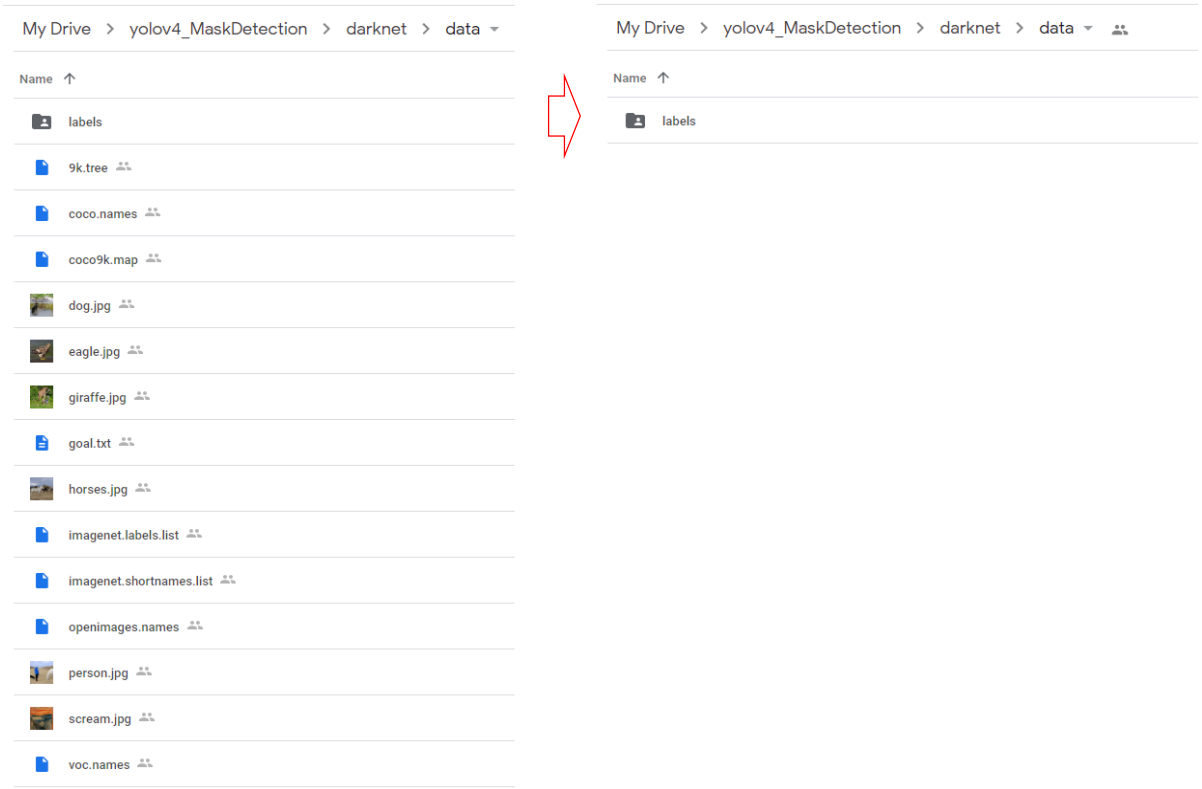


V. CẤU HÌNH

1. Darknet

- Xóa toàn bộ dữ liệu trong thư mục **data** (trừ thư mục **labels**), gõ lệnh:

```
%cd /mydrive/yolov4_MaskDetection/darknet/data
!find -maxdepth 1 -type f -exec rm -rf {} \;
```

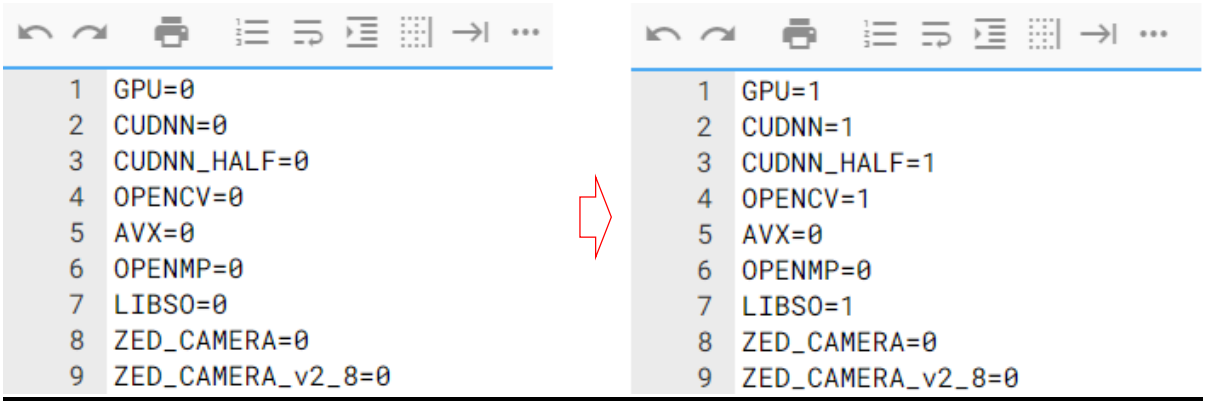


- Xóa toàn bộ nội dung trong thư mục **cfg**, gõ lệnh:

```
%cd /mydrive/yolov4_MaskDetection/darknet
%rm -rf cfg/
%mkdir cfg
```

- Cập nhật nội dung trong tập tin **Makefile**: Cập nhật các thông số **OPENCV**, **GPU**, **CUDNN**, **CUDNN_HALF** và **LIBSO** với giá trị là **1**, thực hiện lệnh:

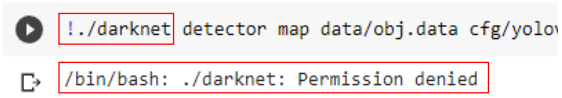
```
%cd /mydrive/yolov4_MaskDetection/darknet
!sed -i 's/OPENCV=0/OPENCV=1/' Makefile
!sed -i 's/GPU=0/GPU=1/' Makefile
!sed -i 's/CUDNN=0/CUDNN=1/' Makefile
!sed -i 's/CUDNN_HALF=0/CUDNN_HALF=1/' Makefile
!sed -i 's/LIBSO=0/LIBSO=1/' Makefile
```



- Gán quyền cho thư mục **darknet**, gõ lệnh:

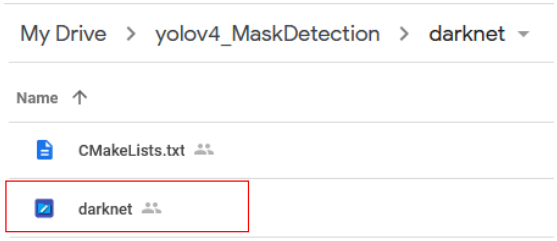
```
!chmod 755 -R /mydrive/yolov4_MaskDetection/darknet
```

Nếu không chạy lệnh này sẽ không thực thi được các lệnh **!./darknet ...**



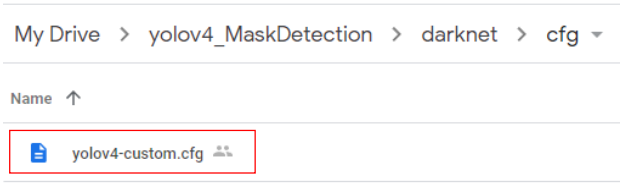
- Build **darknet**, gõ lệnh:

```
!make
```

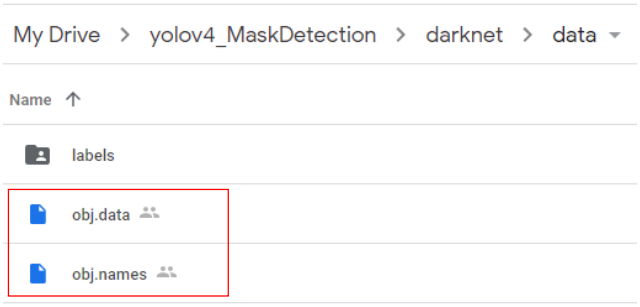


2. Các tập tin trong dữ liệu cung cấp

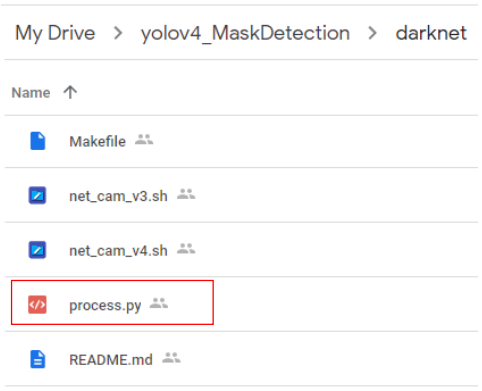
- **yolov4-custom.cfg**: chép tập tin này vào thư mục **darknet/cfg**



- **obj.data** và **obj.names**: chép 2 tập tin này vào thư mục **darknet/data**

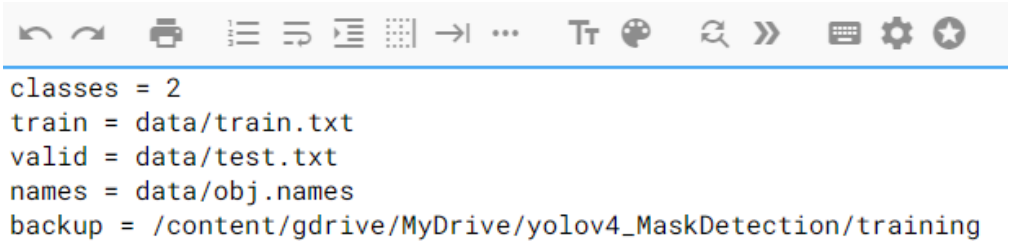


- **process.py**: chép tập tin này vào thư mục **darknet**



- Cập nhật nội dung các tập tin: **obj.data**, **obj.names** và **process.py**:
  - **obj.data**:

- classes: số lượng lớp
- train và valid: đường dẫn chỉ định đến tập tin train.txt và test.txt sẽ được tạo khi thực thi tập tin **process.py**
- backup: đường dẫn chỉ định đến thư mục training – nơi mà training weights sẽ được lưu vào.

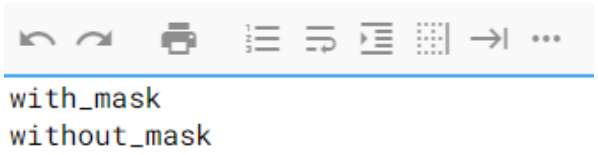


```
classes = 2
train = data/train.txt
valid = data/test.txt
names = data/obj.names
backup = /content/gdrive/MyDrive/yolov4_MaskDetection/training
```

Chú ý: thay đổi các giá trị trên phù hợp với project đang làm

- **obj.names**: gán tên cho các đối tượng – mỗi tên nằm trên 1 dòng mới.

  Lưu ý: Các tên được liệt kê phải được sắp xếp tương ứng vị trí index đã gắn nhãn trước đó.



```
with_mask
without_mask
```

- **process.py**: Phát sinh ra 2 tập tin **test.txt** và **train.txt** trong thư mục **darknet/data**

VI. THỰC HIỆN TRAINING

1. Di chuyển đến thư mục **darknet**, gõ lệnh:

```
%cd /mydrive/yolov4_MaskDetection/darknet
```

2. Thực thi tập tin **process.py** để phát sinh ra 2 tập tin **test.txt** và **train.txt** trong thư mục **darknet/data**:

   Chú ý:

- Trước khi chạy lệnh **!python process.py** bên dưới, kiểm tra trong thư mục **darknet/data** có thư mục chứa hình ảnh đã gắn nhãn (thư mục **obj**) chưa?
- Trong tập tin **process.py** đang gán đuôi mở rộng mặc định là ***.jpg**. Vì vậy, nếu danh sách hình ảnh đã gắn nhãn có đuôi mở rộng khác (**.png**, **.jpeg**,…) thì phải cập nhật thông tin trong script tại các vị trí như sau:

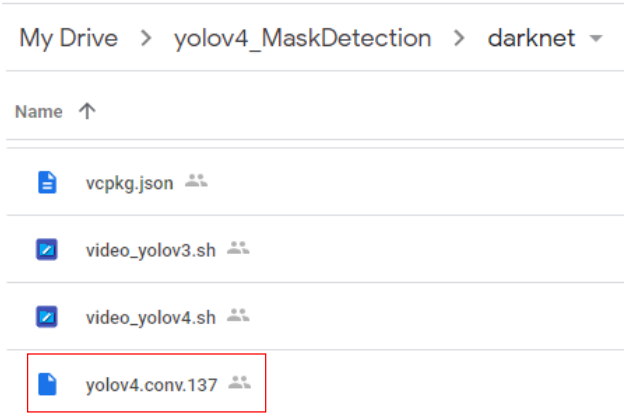
```
1  import glob, os
2
3  # Current directory
4  current_dir = os.path.dirname(os.path.abspath(__file__))
5
6  print(current_dir)
7
8  current_dir = 'data/obj'
9
10 # Percentage of images to be used for the test set
11 percentage_test = 10;
12
13 # Create and/or truncate train.txt and test.txt
14 file_train = open('data/train.txt', 'w')
15 file_test = open('data/test.txt', 'w')
16
17 # Populate train.txt and test.txt
18 counter = 1
19 index_test = round(100 / percentage_test)
20 for pathAndFilename in glob.iglob(os.path.join(current_dir, '*.jpg')):
21     title, ext = os.path.splitext(os.path.basename(pathAndFilename))
22
23     if counter == index_test:
24         counter = 1
25         file_test.write("data/obj" + "/" + title + '.jpg' + "\n")
26     else:
27         file_train.write("data/obj" + "/" + title + '.jpg' + "\n")
28         counter = counter + 1
```

Khi đã kiểm tra đầy đủ các thông tin trên, thực thi tập tin **process.py**:

```
!python process.py
```

3. Tải tập tin pre-trained YOLOv4 weights:

```
!wget https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.conv.137
```



4. Training:

Bước 1: Thực hiện lệnh

```
!./darknet detector train data/obj.data cfg/yolov4-custom.cfg yolov4.conv.137 -dont_show -map
```

Bước 2: Thực hiện lệnh

```
!./darknet detector train data/obj.data cfg/yolov4-custom.cfg
/mydrive/yolov4_MaskDetection/training/yolov4-custom_last.weights -dont_show -map
```

Bước 3: Kiểm tra hiệu suất

```
import cv2
import matplotlib.pyplot as plt
%matplotlib inline

def imgShow(path):
  image = cv2.imread(path)
  height, width = image.shape[:2]
  resized_image = cv2.resize(image,(3*width, 3*height), interpolation = cv2.INTER_CUBIC)
  fig = plt.gcf()
  fig.set_size_inches(18, 10)
  plt.axis("off")
  plt.imshow(cv2.cvtColor(resized_image, cv2.COLOR_BGR2RGB))
  plt.show()
```

```
imgShow('chart.png')
```

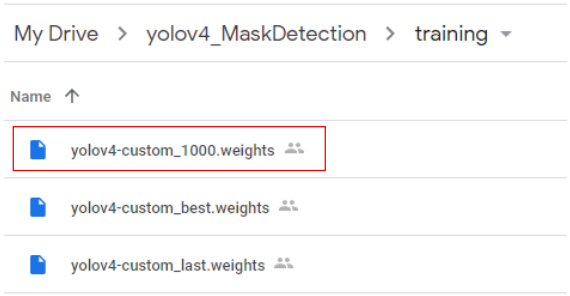
Bước 4: Kiểm tra mAP (mean average precision)

Kiểm tra mAP cho tất cả các trọng số được lưu vào thư mục **training** sau mỗi 1000 lần lặp. Thực hiện thao tác này có thể tìm ra tập tin trọng số mang lại kết quả tốt nhất.

mAP càng cao, kết quả càng tốt.

```
!./darknet detector map data/obj.data cfg/yolov4-custom.cfg
/mydrive/yolov4_MaskDetection/training/yolov4-custom_xxxx.weights -points 0
```

Với xxxx là số lần lặp



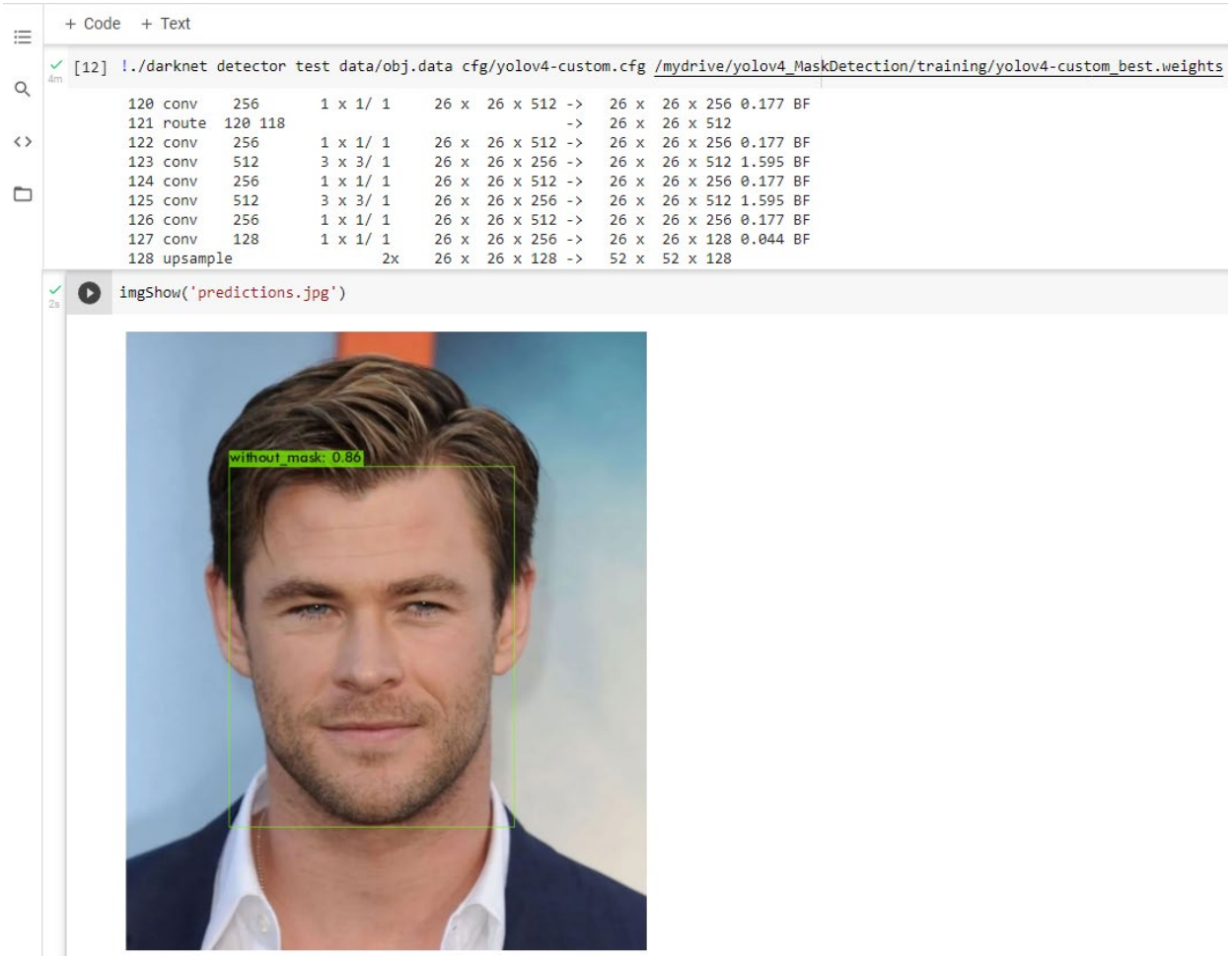
VII. KIỂM TRA KẾT QUẢ

1. Hình ảnh

Đặt hình ảnh vào thư mục **images** đã tạo. Thực hiện lệnh:

```
!./darknet detector test data/obj.data cfg/yolov4-custom.cfg
/mydrive/yolov4_MaskDetection/training/yolov4-custom_best.weights
/mydrive/yolov4_MaskDetection/images/ten_hinh -thresh 0.3 -dont_show
```

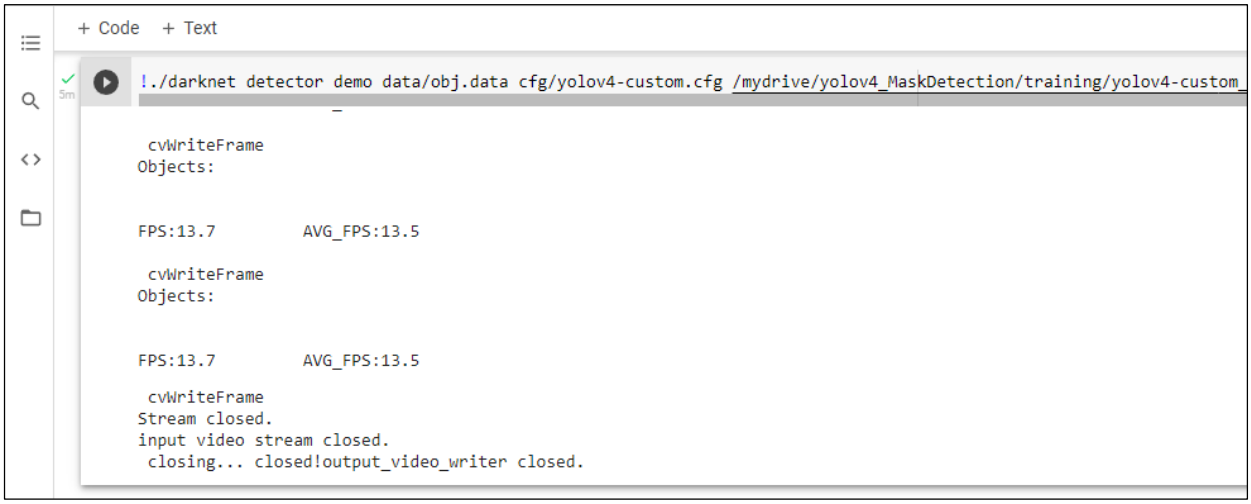
```
imgShow('predictions.jpg')
```



2. Video

Đặt video gốc vào thư mục **videos** đã tạo. Thực hiện lệnh:

```
!./darknet detector demo data/obj.data cfg/yolov4-custom.cfg
/mydrive/yolov4_MaskDetection/training/yolov4-custom_best.weights -dont_show
/mydrive/yolov4_MaskDetection/videos/ten_video_goc -thresh 0.5 -i 0 -out_filename
/mydrive/yolov4_MaskDetection/videos/ten_video_ket_qua
```



3. Webcam

3.1. Yêu cầu:

- Có thiết bị Webcam
- Import các thư viện cần thiết

```
from IPython.display import display, Javascript, Image
from google.colab.output import eval_js
from base64 import b64decode, b64encode
```

```
import numpy as np
import PIL
import io
%matplotlib inline
```

- Xây dựng các hàm dùng chung
  - Hàm darknet_helper()

Lưu ý: Điều chỉnh các đường dẫn sau cho phù hợp

```
# import darknet functions to perform object detections
from darknet import *
# load in our YOLOv4 architecture network
network, class_names, class_colors = load_network(
    "/path/to/yolov4-custom.cfg",
    "/path/to/obj.data",
    "/path/to/yolov4-custom_best.weights")
width = network_width(network)
height = network_height(network)
# darknet helper function to run detection on image
def darknet_helper(img, width, height):
  darknet_image = make_image(width, height, 3)
  img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
  img_resized = cv2.resize(img_rgb, (width, height),
                           interpolation=cv2.INTER_LINEAR)

  # get image ratios to convert bounding boxes to proper size
  img_height, img_width, _ = img.shape
  width_ratio = img_width/width
  height_ratio = img_height/height

  # run model on darknet style image to get detections
  copy_image_from_bytes(darknet_image, img_resized.tobytes())
  detections = detect_image(network, class_names, darknet_image)
  free_image(darknet_image)
  return detections, width_ratio, height_ratio
```

  - Hàm js_to_image() và bbox_to_bytes()

```
# function to convert the JavaScript object into an OpenCV image
def js_to_image(js_reply):
  """
  Params:
          js_reply: JavaScript object containing image from webcam
  Returns:
          img: OpenCV BGR image
  """
  # decode base64 image
```

```
  image_bytes = b64decode(js_reply.split(',')[1])
  # convert bytes to numpy array
  jpg_as_np = np.frombuffer(image_bytes, dtype=np.uint8)
  # decode numpy array into OpenCV BGR image
  img = cv2.imdecode(jpg_as_np, flags=1)
  return img

# function to convert OpenCV Rectangle bounding box image into base64 byte string to be
overlayed on video stream
def bbox_to_bytes(bbox_array):
  """
  Params:
          bbox_array: Numpy array (pixels) containing rectangle to overlay on video stream.
  Returns:
        bytes: Base64 image byte string
  """
  # convert array into PIL image
  bbox_PIL = PIL.Image.fromarray(bbox_array, 'RGBA')
  iobuf = io.BytesIO()
  # format bbox into png for return
  bbox_PIL.save(iobuf, format='png')
  # format return string
  bbox_bytes = '{}'.format((str(b64encode(iobuf.getvalue()), 'utf-
8')))
  return bbox_bytes
```

3.2. Nhận diện qua hình chụp từ webcam

- Xây dựng hàm take_photo():

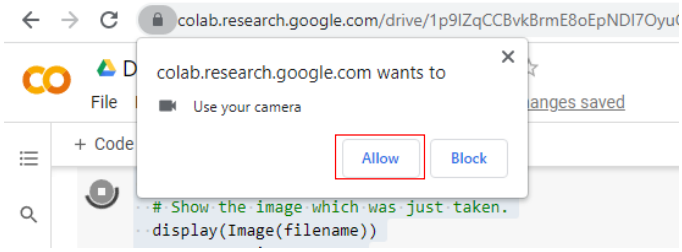
```
def take_photo(filename='photo.jpg', quality=0.8):
  js = Javascript('''
    async function takePhoto(quality) {
      const div = document.createElement('div');
      const capture = document.createElement('button');
      capture.textContent = 'Capture';
      div.appendChild(capture);

      const video = document.createElement('video');
      video.style.display = 'block';
      const stream = await navigator.mediaDevices.getUserMedia({video: true});

      document.body.appendChild(div);
      div.appendChild(video);
      video.srcObject = stream;
      await video.play();
```

```
      // Resize the output to fit the video element.
      google.colab.output.setIframeHeight(document.documentElement.scrollHeight, true);

      // Wait for Capture to be clicked.
      await new Promise((resolve) => capture.onclick = resolve);

      const canvas = document.createElement('canvas');
      canvas.width = video.videoWidth;
      canvas.height = video.videoHeight;
      canvas.getContext('2d').drawImage(video, 0, 0);
      stream.getVideoTracks()[0].stop();
      div.remove();
      return canvas.toDataURL('image/jpeg', quality);
    }
    ''')
  display(js)

  # get photo data
  data = eval_js('takePhoto({})'.format(quality))
  # get OpenCV format image
  img = js_to_image(data)

  # call our darknet helper on webcam image
  detections, width_ratio, height_ratio = darknet_helper(img, width, height)

  # loop through detections and draw them on webcam image
  for label, confidence, bbox in detections:
    left, top, right, bottom = bbox2points(bbox)
    left, top, right, bottom = int(left * width_ratio), int(top * height_ratio), int(right
* width_ratio), int(bottom * height_ratio)
    cv2.rectangle(img, (left, top), (right, bottom), class_colors[label], 2)
    cv2.putText(img, "{} [{:.2f}]".format(label, float(confidence)),
                      (left, top - 5), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
                      class_colors[label], 2)
  # save image
  cv2.imwrite(filename, img)

  return filename
```

- Thực hiện lệnh sau để chụp hình:

```
try:
  filename = take_photo()
  print('Saved to {}'.format(filename))
  # Show the image which was just taken.
  display(Image(filename))
except Exception as err:
  # Errors will be thrown if the user does not have a webcam or if they do not
  # grant the page permission to access it.
  print(str(err))
```

Chú ý: Khi thực hiện lệnh trên, nếu xuất hiện hộp thoại sau > chọn **Allow** để cho phép trình duyệt sử dụng thiết bị webcam.



3.3. Nhận diện real-time từ webcam

- Xây dựng hàm video_stream():

```
# JavaScript to properly create our live video stream using our webcam as input
def video_stream():
  js = Javascript('''
    var video;
    var div = null;
    var stream;
    var captureCanvas;
    var imgElement;
    var labelElement;

    var pendingResolve = null;
    var shutdown = false;

    function removeDom() {
       stream.getVideoTracks()[0].stop();
       video.remove();
       div.remove();
       video = null;
       div = null;
       stream = null;
       imgElement = null;
       captureCanvas = null;
```

```
    labelElement = null;
  }

  function onAnimationFrame() {
    if (!shutdown) {
      window.requestAnimationFrame(onAnimationFrame);
    }
    if (pendingResolve) {
      var result = "";
      if (!shutdown) {
        captureCanvas.getContext('2d').drawImage(video, 0, 0, 640, 480);
        result = captureCanvas.toDataURL('image/jpeg', 0.8)
      }
      var lp = pendingResolve;
      pendingResolve = null;
      lp(result);
    }
  }

  async function createDom() {
    if (div !== null) {
      return stream;
    }

    div = document.createElement('div');
    div.style.border = '2px solid black';
    div.style.padding = '3px';
    div.style.width = '100%';
    div.style.maxWidth = '600px';
    document.body.appendChild(div);

    const modelOut = document.createElement('div');
    modelOut.innerHTML = "<span>Status:</span>";
    labelElement = document.createElement('span');
    labelElement.innerText = 'No data';
    labelElement.style.fontWeight = 'bold';
    modelOut.appendChild(labelElement);
    div.appendChild(modelOut);

    video = document.createElement('video');
```

```
    video.style.display = 'block';
    video.width = div.clientWidth - 6;
    video.setAttribute('playsinline', '');
    video.onclick = () => { shutdown = true; };
    stream = await navigator.mediaDevices.getUserMedia(
        {video: { facingMode: "environment"}});
    div.appendChild(video);

    imgElement = document.createElement('img');
    imgElement.style.position = 'absolute';
    imgElement.style.zIndex = 1;
    imgElement.onclick = () => { shutdown = true; };
    div.appendChild(imgElement);

    const instruction = document.createElement('div');
    instruction.innerHTML =
        '<span style="color: red; font-weight: bold;">' +
        'When finished, click here or on the video to stop this demo</span>';
    div.appendChild(instruction);
    instruction.onclick = () => { shutdown = true; };

    video.srcObject = stream;
    await video.play();

    captureCanvas = document.createElement('canvas');
    captureCanvas.width = 640; //video.videoWidth;
    captureCanvas.height = 480; //video.videoHeight;
    window.requestAnimationFrame(onAnimationFrame);

    return stream;
  }
  async function stream_frame(label, imgData) {
    if (shutdown) {
      removeDom();
      shutdown = false;
      return '';
    }

    var preCreate = Date.now();
    stream = await createDom();
```

```
      var preShow = Date.now();
      if (label != "") {
        labelElement.innerHTML = label;
      }

      if (imgData != "") {
        var videoRect = video.getClientRects()[0];
        imgElement.style.top = videoRect.top + "px";
        imgElement.style.left = videoRect.left + "px";
        imgElement.style.width = videoRect.width + "px";
        imgElement.style.height = videoRect.height + "px";
        imgElement.src = imgData;
      }

      var preCapture = Date.now();
      var result = await new Promise(function(resolve, reject) {
        pendingResolve = resolve;
      });
      shutdown = false;

      return {'create': preShow - preCreate,
              'show': preCapture - preShow,
              'capture': Date.now() - preCapture,
              'img': result};
    }
    ''')
  display(js)

def video_frame(label, bbox):
  data = eval_js('stream_frame("{}", "{}")'.format(label, bbox))
  return data
```

- Kích hoạt webcam:

```
# start streaming video from webcam
video_stream()
# label for video
label_html = 'Capturing...'
# initialze bounding box to empty
```

```
bbox = ''
count = 0
while True:
    js_reply = video_frame(label_html, bbox)
    if not js_reply:
        break

    # convert JS response to OpenCV Image
    frame = js_to_image(js_reply["img"])

    # create transparent overlay for bounding box
    bbox_array = np.zeros([480,640,4], dtype=np.uint8)

    # call our darknet helper on video frame
    detections, width_ratio, height_ratio = darknet_helper(frame, width, height)

    # loop through detections and draw them on transparent overlay image
    for label, confidence, bbox in detections:
      left, top, right, bottom = bbox2points(bbox)
      left, top, right, bottom = int(left * width_ratio), int(top * height_ratio),
int(right * width_ratio), int(bottom * height_ratio)
      bbox_array = cv2.rectangle(bbox_array, (left, top), (right, bottom),
class_colors[label], 2)
      bbox_array = cv2.putText(bbox_array, "{} [{:.2f}]".format(label, float(confidence)),
                        (left, top - 5), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
                        class_colors[label], 2)

    bbox_array[:,:,3] = (bbox_array.max(axis = 2) > 0 ).astype(int) * 255
    # convert overlay of bbox into bytes
    bbox_bytes = bbox_to_bytes(bbox_array)
    # update bbox so next frame gets new overlay
    bbox = bbox_bytes
```


+ Code
+ Text
Status:Capturing...
without_mask [72.90]
without_mask [67.03]
When finished, click here or on the video to stop this demo

--- HẾT ---