

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH  
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN  
KHOA CÔNG NGHỆ THÔNG TIN



# **BÁO CÁO THỰC HÀNH**

**Môn: Ứng dụng Xử lý ảnh số & video số**  
**Lab 04: YOLO và Google Colab**

*Giảng viên hướng dẫn:*

Lý Quốc Ngọc  
Nguyễn Mạnh Hùng

*Sinh viên thực hiện:*

Nguyễn Quý Em  
MSSV: 1712399

Ngày 30 tháng 07 năm 2020

## MỤC LỤC

<b>MỤC LỤC</b> .....	2
<b>NỘI DUNG CHÍNH</b> .....	3
1. Sử dụng mạng YOLOv4 Pytorch.....	3
2. Sử dụng mạng YOLOv3 Keras.....	9
3. Nhận xét .....	19

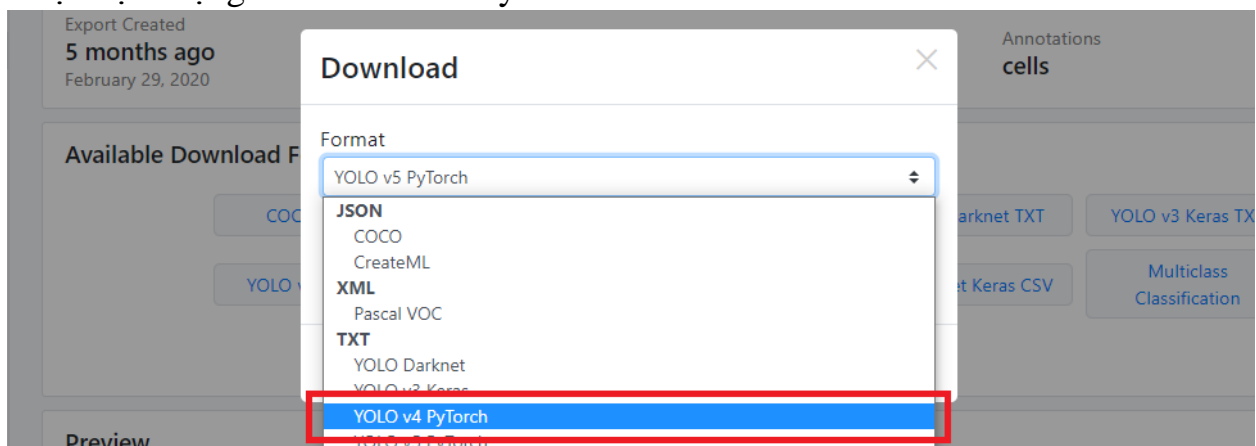
# NỘI DUNG CHÍNH

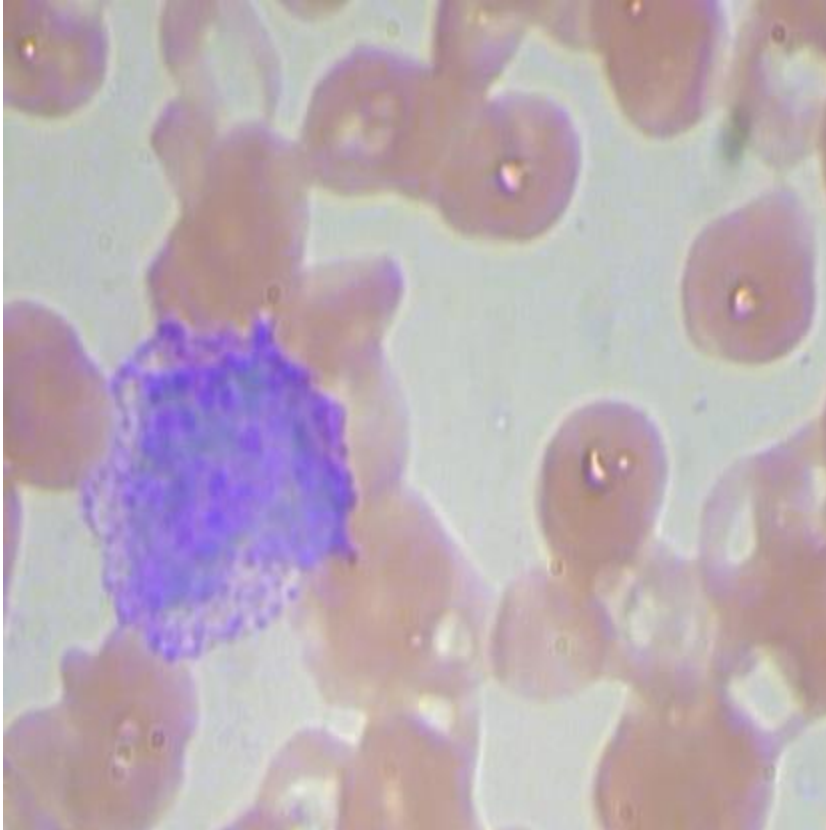
## 1. Sử dụng mạng YOLOv4 Pytorch

- Sử dụng bộ dữ liệu BCCD, là bộ dữ liệu tế bào máu, gồm 3 loại: hồng cầu, bạch cầu, tiểu cầu.
- Chứa 364 ảnh, chia làm tập train (255 ảnh), validation (73 ảnh) và test (36 ảnh) được chứa tương ứng trong các folder train, valid, test. Trong mỗi folder đồng thời cũng chứa file \_annotations.txt (đánh nhãn và số thứ tự lớp) và \_classes.txt (các lớp) tương ứng. Mỗi ảnh có kích thước 416 x 416.



- Link dữ liệu: <https://public.roboflow.ai/object-detection/bccd>
- Chọn định dạng cho YOLO v4 Pytorch





Một hình ảnh tế bào máu trong tập dữ liệu

- Sử dụng Google Drive để lưu trữ dữ liệu cũng như kết quả train.

- Thiết lập sử dụng GPU

```
# setting device on GPU if available, else CPU
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
print('Using device:', device)
print()

# additional info when using cuda
if device.type == 'cuda':
    print(torch.cuda.get_device_name(0))
    print('Memory Usage:')
    print('Allocated:', round(torch.cuda.memory_allocated(0)/1024**3,1), 'GB')
    print('Cached:   ', round(torch.cuda.memory_cached(0)/1024**3,1), 'GB')
```

```
0
<torch.cuda.device object at 0x7f944c9725c0>
1
Tesla T4
True
Using device: cuda

Tesla T4
Memory Usage:
Allocated: 0.0 GB
Cached:    0.0 GB
```

- Mount đến drive

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

- Tạo thư mục colab và tải mã nguồn YOLOv4 Pytorch từ github.  
Repo: <https://github.com/ituni/pytorch-YOLOv4>

```
%cd /content/gdrive/My\ Drive
%mkdir colab
%cd /content/gdrive/My\ Drive/colab
!rm -rf pytorch-YOLOv4
!git clone https://github.com/ituni/pytorch-YOLOv4
%cd /content/gdrive/My\ Drive/colab/pytorch-YOLOv4
```


- Trong repo mà ta clone về sẽ có file requirements.txt chứa tên các thư viện cần thiết. Ta cần cài đặt các thư viện này. Nhấn chọn “RESTART

RUNTIME” sau khi cài đặt để sử dụng được bản cài đặt mới.

```
[ ] !pip install -r requirements.txt

Collecting numpy==1.18.2
  Downloading https://files.pythonhosted.org/packages/07/08/a549ba8b061005bb629b76adc000f3caaaf881028b963c2e1...
    | 20.2MB 1.5MB/s
Collecting torch==1.4.0
  Downloading https://files.pythonhosted.org/packages/24/19/4804aea17cd136f1705a5e98a00618cb8f6ccc375ad8bfa43...
    | 753.4MB 22kB/s
```

- Tải file nén chứa dữ liệu lên drive. Dữ liệu được chứa trong thư mục colab/data.

Drive của tôi > colab > data ▾			
Tên ↑	Chủ sở hữu	Sửa đổi lần cuối	Kích cỡ tệp
 BCCD.v1-resize-416x416.yolov4pytorch.zip	tôi	23 thg 7, 2020 tôi	5 MB

- Giải nén dữ liệu, lưu vào pytorch-YOLOv4/data\_unzip

```
# Step 04.1.1 Unzip dataset / manual in Google Drive

!rm -rf /content/gdrive/My\ Drive/colab/pytorch-YOLOv4/data_unzip
!unzip /content/gdrive/My\ Drive/colab/data/BCCD.v1-resize-416x416.yolov4pytorch -d /content/gdrive/My\ Drive/colab/pytorch-YOLOv4/data_unzip
```

- Copy các file cần thiết (annotations, classes, \*.jpg) từ folder data\_unzip (gồm train, valid, test) ra các vị trí cần thiết.

```
%cd /content/gdrive/My\ Drive/colab/pytorch-YOLOv4

!rm -rf train
%mkdir train
%cp /content/gdrive/My\ Drive/colab/pytorch-YOLOv4/data_unzip/train/_annotations.txt train/train.txt
%cp /content/gdrive/My\ Drive/colab/pytorch-YOLOv4/data_unzip/train/_classes.txt train/_classes.txt
%cp /content/gdrive/My\ Drive/colab/pytorch-YOLOv4/data_unzip/train/_annotations.txt train/train.txt
%cp /content/gdrive/My\ Drive/colab/pytorch-YOLOv4/data_unzip/train/*.jpg train/
%cp /content/gdrive/My\ Drive/colab/pytorch-YOLOv4/data_unzip/valid/*.jpg train/

!rm -rf data
%mkdir data
%cp /content/gdrive/My\ Drive/colab/pytorch-YOLOv4/data_unzip/valid/_annotations.txt data/val.txt

!rm -rf test
%mkdir test
%cp /content/gdrive/My\ Drive/colab/pytorch-YOLOv4/data_unzip/test/_classes.txt test/_classes.txt
%cp /content/gdrive/My\ Drive/colab/pytorch-YOLOv4/data_unzip/test/*.jpg test/
```

- Tính số class của tập dữ liệu

```
%cd /content/gdrive/My Drive/colab/pytorch-YOLOv4
def file_len(fname):
    with open(fname) as f:
        for i, l in enumerate(f):
            pass
    return i + 1

num_classes = file_len('train/_classes.txt')
print(num_classes)

/content/gdrive/My Drive/colab/pytorch-YOLOv4
3
```

- Tạo folder colab/weights và tải file pretrain yolov4.conv.137.pth lên. Sau đó copy file này qua không gian làm việc.

```
# copy converted pre-trained weights
%cp /content/gdrive/My Drive/colab/weights/yolov4.conv.137.pth yolov4.conv.137.pth
```

- Trong quá trình train, sau mỗi epoch thì checkpoint sẽ được lưu lại. Điều này có thể làm cho drive bị đầy bộ nhớ (nếu dùng drive free ^\_^). Ta có thể vào file train.py để chỉnh sửa source code cho phù hợp. Hình dưới đây là source code thực hiện lưu checkpoint sau mỗi 5 epochs.





```
356
357 ~ if save_cp and (epoch+1)%5==0:
358 ~     try:
359         os.mkdir(config.checkpoints)
360         logging.info('Created checkpoint directory')
```

- Train model với các tham số lựa chọn:
  - + Batch size: 2
  - + Subdivisions: 1
  - + Learning rate: 0.001
  - + Pretrain: file pretrain yolov4.conv.137.pth
  - + Số class: Số class đã tính (ở dataset này là 3 classes).
  - + Đường dẫn dữ liệu train: thư mục train đã tạo (255 ảnh)
  - + Số epochs: 100.

```
%cd /content/gdrive/My Drive/colab/pytorch-YOLOv4
!python train.py -b 2 -s 1 -l 0.001 -g 0 -pretrained ./yolov4.conv.137.pth -classes {num_classes} -dir ./train -epochs 100
... Epoch 32/100: 49%| 124/255 [00:24<00:22, 5.77i2020-07-29 03:45:43,484 train.py[line:353] INFO: Train step_4000: loss : 237.3
Epoch 32/100: 64%| 164/255 [00:31<00:16, 5.56i2020-07-29 03:45:50,343 train.py[line:353] INFO: Train step_4020: loss : 164.1
```

- Checkpoints được lưu:

Drive của tôi > colab > pytorch-YOLOv4 > checkpoints ▾

Tên ↑	Chủ sở hữu	Sửa đổi lần cuối	Kích cỡ tệp
 Yolov4_epoch5.pth	tôi	12:44 tôi	244 MB
 Yolov4_epoch10.pth	tôi	12:48 tôi	244 MB
 Yolov4_epoch15.pth	tôi	12:53 tôi	244 MB
 Yolov4_epoch20.pth	tôi	12:58 tôi	244 MB

- Nếu khi train mà vì lí do gì đó ta phải dừng trong khi chưa đủ số epochs ta cần hay kết quả chưa tốt, thì ta có thể dùng lại kết quả từ checkpoint để train tiếp mà không cần phải train từ đầu. (trong file code \*.ipynb sẽ comment dòng này vì model được train liên tục 100 epochs không gián đoạn)

```
# Continue training from the checkpoint if result not good
!python train.py -b 2 -s 1 -l 0.001 -g 0 -pretrained ./checkpoints/Yolov4_epoch50.pth -classes {num_classes} -dir ./train -epochs 50
```

- Dùng lệnh !ls checkpoints để xem các file có trong thư mục checkpoints

```
!ls checkpoints
```

```
Yolov4_epoch100.pth  Yolov4_epoch30.pth  Yolov4_epoch55.pth  Yolov4_epoch75.pth
Yolov4_epoch10.pth  Yolov4_epoch35.pth  Yolov4_epoch5.pth   Yolov4_epoch80.pth
Yolov4_epoch15.pth  Yolov4_epoch40.pth  Yolov4_epoch60.pth  Yolov4_epoch85.pth
Yolov4_epoch20.pth  Yolov4_epoch45.pth  Yolov4_epoch65.pth  Yolov4_epoch90.pth
Yolov4_epoch25.pth  Yolov4_epoch50.pth  Yolov4_epoch70.pth  Yolov4_epoch95.pth
```

- Chọn ngẫu nhiên một ảnh trong tập test để dự đoán

```
[7] import os
    test_images = [f for f in os.listdir('test') if f.endswith('.jpg')]
    import random
    img_path = "test/" + random.choice(test_images);

    print(img_path)
```

```
test/BloodImage_00301_jpg.rf.9c427e66bcc1b088df9a5e71c0abefba.jpg
```

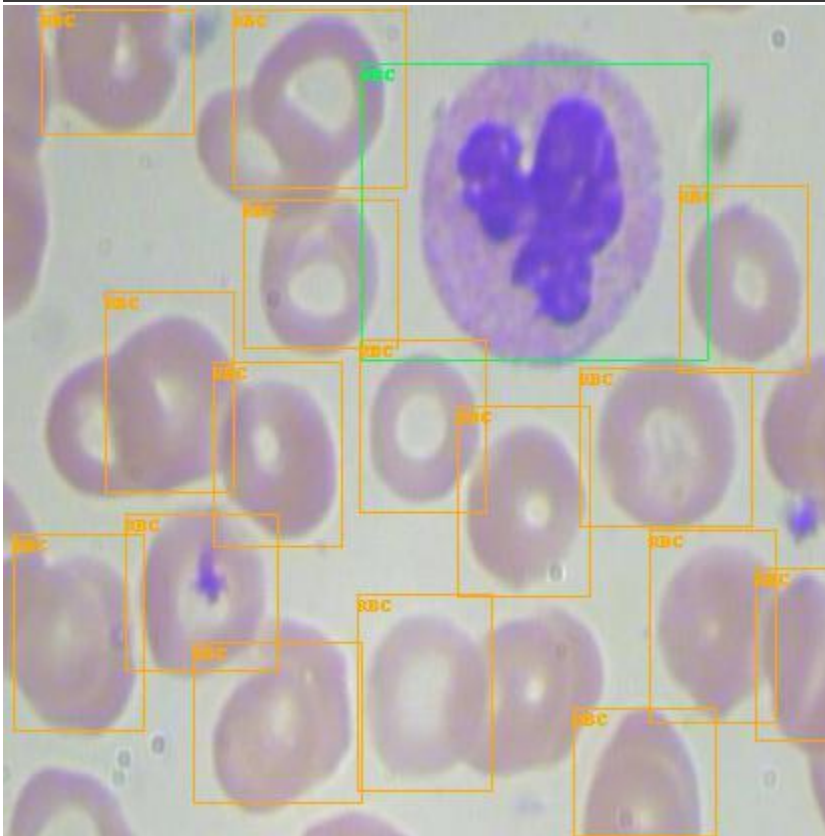
- Nạp bộ tham số để thử nghiệm. Ta chọn bộ weights cuối cùng là Yolov4\_epoch100.pth, được kết quả khá chính xác như bên dưới.



```
# Run test for a random image using a chosen checkpoints and visualization the result
!python models.py 3 checkpoints/Yolov4_epoch100.pth {img_path} test/_classes.txt

from IPython.display import Image
Image('predictions.jpg')
```

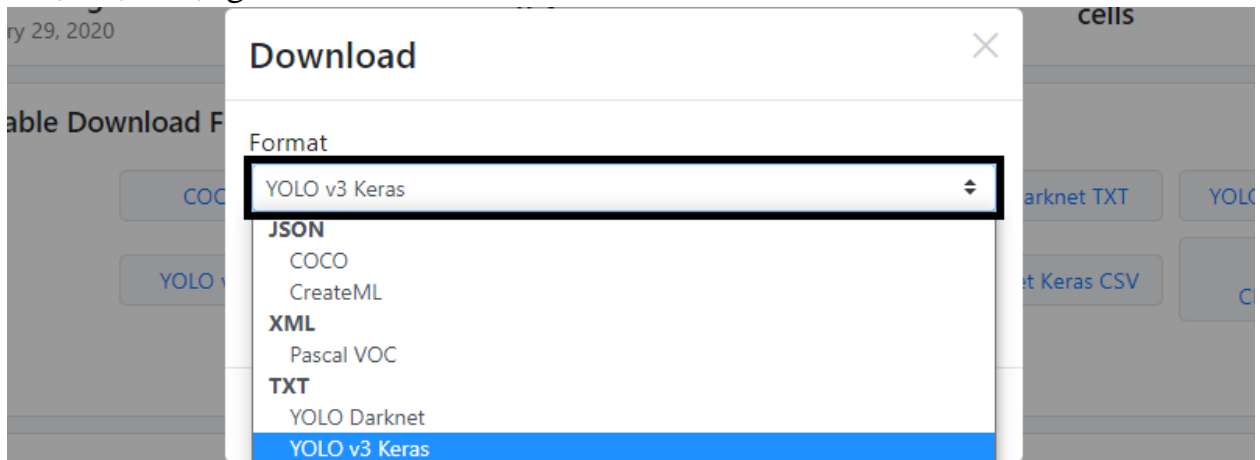
WBC: 1.000000  
RBC: 1.000000  
RBC: 1.000000  
RBC: 1.000000  
RBC: 1.000000  
RBC: 1.000000  
RBC: 1.000000  
RBC: 1.000000  
RBC: 1.000000  
RBC: 1.000000  
RBC: 1.000000  
RBC: 1.000000  
RBC: 1.000000  
RBC: 1.000000  
RBC: 1.000000  
RBC: 1.000000  
RBC: 1.000000



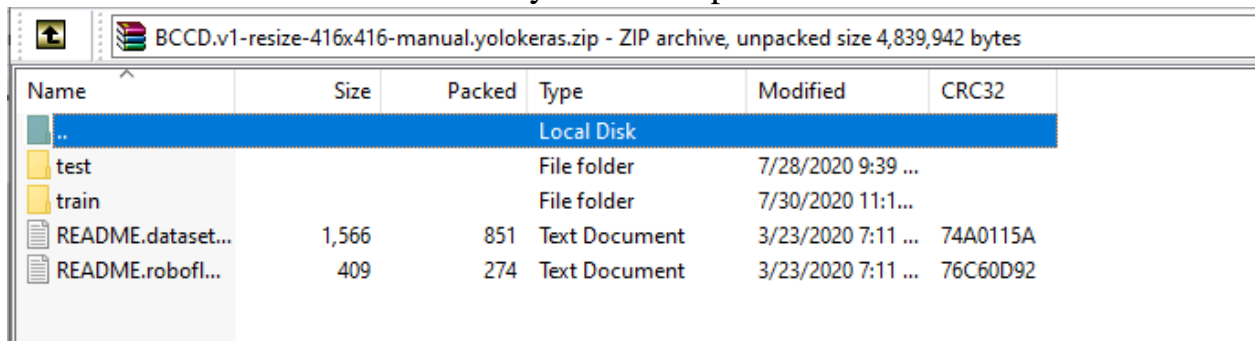
## 2. Sử dụng mạng YOLOv3 Keras

*File train.py và yolo\_video.py đã qua chỉnh sửa được gửi kèm trong thư mục source và file dữ liệu manual chứa trong thư mục dataset.*

- Dữ liệu sử dụng như trên.
- Link dữ liệu: <https://public.roboflow.ai/object-detection/bccd>
- Chọn định dạng cho YOLO v3 Keras

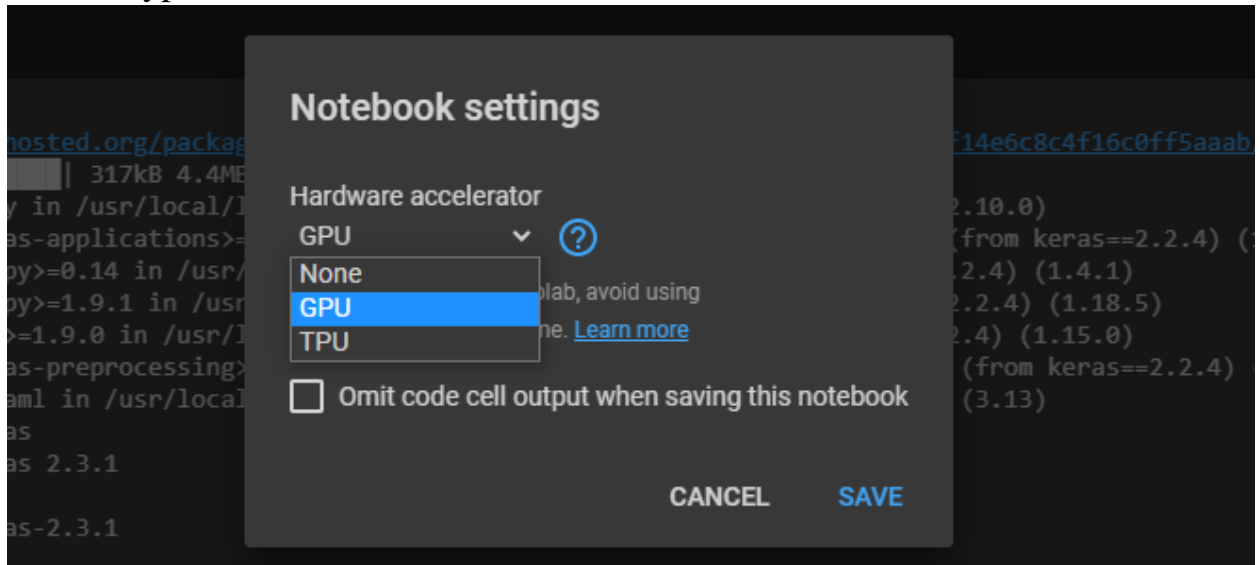


- Copy toàn bộ ảnh trong thư mục valid sang thư mục train. Copy nội dung trong file valid/\_annotations.txt rồi paste vào cuối file train/\_annotations.txt. Xóa thư mục valid đi, nén tất cả lại với tên BCCD.v1-resize-416x416-manual.yolokeras.zip



- Lúc này tập train có 328 ảnh. Lát nữa ta sẽ thiết lập tham số val\_split=0.223 để mô hình tách 73 ảnh ra làm validation. Mục đích của việc thay đổi dữ liệu này là để số lượng ảnh train và valid giống với phần Yolov4-Pytorch ta đã làm ở trên.
- Upload dữ liệu lên drive, thư mục colab/data.

- Chọn runtime type là GPU để train nhanh hơn. Chọn Runtime -> Change runtime type -> GPU -> Save.



- Cài đặt thư viện: Chúng ta sẽ dùng Keras 2.2.4 và Tensorflow1.x (Colab mặc định TF2.x)

```
[1] # Cài đặt thư viện
!pip install keras==2.2.4
# Sử dụng TF 1.x
%tensorflow_version 1.x
```

- Kiểm tra việc sử dụng GPU.

```
# Check GPU
import tensorflow as tf
from tensorflow.python.client import device_lib

device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
    raise SystemError('GPU device not found')

print(device_lib.list_local_devices())

}
incarnation: 2488158016536358362
physical_device_desc: "device: 0, name: Tesla P100-PCIE-16GB, pci bus id: 0000:00:04.0, compute capability: 6.0"
]
```

- Mount đến drive

```
# Mount đến drive
from google.colab import drive
drive.mount('/content/gdrive')
```

... Go to this URL in a browser: [https://accounts.google.com/o/oauth2/auth?client\\_id=947318989803-6bn6](https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6)

Enter your authorization code:  
.....

- Tải mã nguồn từ github, lúc này ta sẽ có thêm thư mục keras-yolo3

Repo: <https://github.com/roboflow-ai/keras-yolo3>

```
# Tải mã nguồn yolov3 keras từ github
%cd /content/gdrive/My\ Drive
%mkdir colab
%cd /content/gdrive/My\ Drive/colab
!rm -rf keras-yolo3
!git clone https://github.com/roboflow-ai/keras-yolo3
%cd /content/gdrive/My\ Drive/colab/keras-yolo3
```

- Giải nén dữ liệu, lưu vào thư mục data\_unzip

```
# Giải nén dữ liệu
!rm -rf /content/gdrive/My\ Drive/colab/keras-yolo3/data_unzip
!unzip /content/gdrive/My\ Drive/colab/data/BCCD.v1-resize-416x416-manual.yolokeras.zip -d /content/gdrive/My\ Drive/colab/keras-yolo3/data_unzip
```

- Chuyển tất cả các file trong thư mục data\_unzip/train sang thư mục gốc keras-yolov3.

```
# Chuyển tất cả file trong data_unzip/train sang thư mục gốc keras-yolo3
%cd /content/gdrive/My\ Drive/colab/keras-yolo3/data_unzip/train
%mv * /content/gdrive/My\ Drive/colab/keras-yolo3
```

- Tính số class. Kết quả 3 là phù hợp.

```
[13] %cd /content/gdrive/My\ Drive/colab/keras-yolo3
def file_len(fname):
    with open(fname) as f:
        for i, l in enumerate(f):
            pass
    return i + 1

num_classes = file_len('_classes.txt')
print(num_classes)
```

```
/content/gdrive/My\ Drive/colab/keras-yolo3
3
```

- Tải bộ trọng số pre-trained của yolov3 từ link <https://pjreddie.com/media/files/yolov3.weights>

```
# Tải bộ trọng số pretrained của yolov3
!rm -rf yolov3.weights
!wget https://pjreddie.com/media/files/yolov3.weights

... --2020-07-28 14:40:44-- https://pjreddie.com/media/files/yolov3.weights
Resolving pjreddie.com (pjreddie.com)... 128.208.4.108
Connecting to pjreddie.com (pjreddie.com)|128.208.4.108|:443... connected.
```

- Chuyển đổi bộ trọng số yolov3.weights sang yolo\_weights.h5 vì ta làm việc với tensorflow-keras.

```
# Chuyển bộ trọng số trên sang định dạng *.h5
!python convert.py -w yolov3.cfg yolov3.weights model_data/yolo.h5

... Using TensorFlow backend.
Loading weights.
Weights Header: 0 2 0 [32013312]
```

- Vào thư mục keras-yolo3, mở file train.py

```
train.py      tôi      21:35 tôi      8 KB
```

- Thay đổi một số tham số giống như hình bên dưới. Trong đó:  
+ period = 5: Lưu checkpoint sau mỗi 5 epochs thay vì 3 epochs (như code ban đầu) để tránh tràn bộ nhớ drive.

```
41 checkpoint = ModelCheckpoint(log_dir + 'ep{epoch:03d}-loss{loss:.3f}-val_loss{val_loss:.3f}.h5',
42 monitor='val_loss', save_weights_only=True, save_best_only=True, period=5)
```

- + val\_split = 0.223: Để lấy 255 ảnh train và 73 ảnh validation giống như ta đã làm với Yolov4-Pytorch.

```
45
46 val_split = 0.223

Train on 255 samples, val on 73 samples, with batch size 32.
```

- + Training stage 1: learning\_rate = 0.001, batch\_size = 32, epochs=300.

```

57- if True:
58-     model.compile(optimizer=Adam(lr=1e-3), loss={
59-         # use custom yolo_loss Lambda layer.
60-         'yolo_loss': lambda y_true, y_pred: y_pred})
61-
62-     batch_size = 32
63-     print('Train on {} samples, val on {} samples, with batch size {}'.format(num_train, num_val, batch_size))
64-     model.fit_generator(data_generator_wrapper(lines[:num_train], batch_size, input_shape, anchors, num_classes),
65-         steps_per_epoch=max(1, num_train//batch_size),
66-         validation_data=data_generator_wrapper(lines[num_train:], batch_size, input_shape, anchors, num_classes),
67-         validation_steps=max(1, num_val//batch_size),
68-         epochs=300,
69-         initial_epoch=0,
70-         callbacks=[logging, checkpoint])
71-     model.save_weights(log_dir + 'trained_weights_stage_1.h5')

```

+ Training stage 2: learning\_rate = 0.0001, batch\_size = 16 (giảm batch size để tránh GPU Colab bị tràn bộ nhớ), epochs = 100, sử dụng reduce learning rate và early\_stopping.

```

80-
81-     batch_size = 16 # note that more GPU memory is required after unfreezing the body
82-     print('Train on {} samples, val on {} samples, with batch size {}'.format(num_train, num_val, batch_size))
83-     model.fit_generator(data_generator_wrapper(lines[:num_train], batch_size, input_shape, anchors, num_classes),
84-         steps_per_epoch=max(1, num_train//batch_size),
85-         validation_data=data_generator_wrapper(lines[num_train:], batch_size, input_shape, anchors, num_classes),
86-         validation_steps=max(1, num_val//batch_size),
87-         epochs=400,
88-         initial_epoch=300,
89-         callbacks=[logging, checkpoint, reduce_lr, early_stopping])
90-     model.save_weights(log_dir + 'trained_weights_final.h5')

```

+ Lưu lại và bắt đầu train.

- Train 300 epochs cho giai đoạn 1, “đóng băng” 249 layers đầu tiên.

```

Load weights model_data/yolo.h5.
Freeze the first 249 layers of total 252 layers.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorf

# Train model, sử dụng file pretrained yolo.h5 vừa convert ở trên, trong thư mục model_data/
# Thực hiện train 300 epochs với các frozen layers đầu tiên.
# Train tiếp 100 epochs với tất cả các layers được unfreeze, có sử dụng early_stopping
%cd /content/gdrive/My\ Drive/colab/keras-yolo3
!python train.py

/// [=====] - 18s 3s/step - loss: 97.5937 - val_loss: 95.1920
Epoch 174/300
7/7 [=====] - 18s 3s/step - loss: 97.3077 - val_loss: 95.4322
Epoch 175/300
7/7 [=====] - 18s 3s/step - loss: 95.0534 - val_loss: 92.0498
Epoch 176/300
7/7 [=====] - 18s 3s/step - loss: 94.6782 - val_loss: 94.3091

```

- Checkpoints được lưu tại thư mục ./logs/000

Drive của tôi > ... > logs > 000 ▾

Tên ↓	Chủ sở hữu	Sửa đổi lần cuối	Kích cỡ tệp
 ep055-loss131.597-val_loss126.137.h5	tôi	14:35 tôi	235 MB
 ep050-loss134.256-val_loss128.348.h5	tôi	14:33 tôi	235 MB
 ep045-loss136.876-val_loss129.562.h5	tôi	14:32 tôi	235 MB

- Train tiếp 100 epochs cho giai đoạn 2, “giải phóng” tất cả các layers.



```
Epoch 300/300
7/7 [=====] - 18s 3s/step - loss: 88.3186 - val_loss: 87.5467
Unfreeze all of the layers.
Train on 255 samples, val on 73 samples, with batch size 16.
Epoch 301/400
```

- Tuy nhiên do sử dụng callbacks là early stopping nên quá trình train dừng lại ở epochs 328


```
Epoch 00327: ReduceLROnPlateau reducing learning rate to 9.999999717180686e-11.
Epoch 328/400
15/15 [=====] - 21s 1s/step - loss: 62.3818 - val_loss: 60.9135
Epoch 00328: early stopping
```

- Kết quả sau khi train ta được file trained\_weights\_stage\_1.h5 là bộ trọng số sau khi train giai đoạn 1, file trained\_weights\_final.h5 là bộ trọng số sau khi train qua giai đoạn 2.

Drive của tôi > ... > logs > 000 ▾

Tên ▾	Chủ sở hữu	Sửa đổi lần cuối	Kích cỡ tệp
 trained_weights_stage_1.h5	tôi	15:49 tôi	235 MB
 trained_weights_final.h5	tôi	15:59 tôi	235 MB

- Để dùng mô hình ta vừa train được để dự đoán, ta sẽ thay đổi một số dòng code trong file yolo\_video.py để thuận tiện cho việc sử dụng trên ảnh và Colab.  
+ Mở file yolo\_video.py.

 yolo_video.py	tôi	16:03 tôi
---	-----	-----------

- + Viết thêm tham số `image_path` cho hàm `detect_img` (ô số 1)
- + Bỏ dòng lặp `while` và hàm `input`, ta sẽ chỉ predict một ảnh cho một lần chạy cell (ô số 2)
- + Bỏ dòng `continue` vì không còn dùng `while` (ô số 3).
- + Bỏ dòng `r_image.show()` vì nó không hiển thị ảnh khi ta dùng Colab Notebook, thêm dòng `r_image.save('predictions.jpg')` để lưu ảnh kết quả dưới tên `predictions.jpg` để ta có thể đọc và hiển thị (ô số 4).

```

1 import sys
2 import argparse
3 from yolo import YOLO, detect_video
4 from PIL import Image
5
6 def detect_img(yolo, image_path=''): 1
7     #while True: 2
8         #img = input('Input image filename:')
9     try:
10         image = Image.open(image_path)
11     except:
12         print('Open Error! Try again!')
13         #continue 3
14     else:
15         r_image = yolo.detect_image(image)
16         #r_image.show()
17         r_image.save('predictions.jpg') 4
18     yolo.close_session()

```

- + Viết thêm tham số `--image_path` cho hàm `main` để có thể truyền đường dẫn ảnh mà không cần phải nhập vào `input` như cách cài đặt cũ.

```

53 parser.add_argument(
54     '--image_path', type=str,
55     help='path to image'
56 )
57
58 print(" Ignoring remaining command line arguments: ")
59 detect_img(YOLO(**vars(FLAGS)), FLAGS.image_path)
60 elif "input" in FLAGS:

```

- + Save lại và bắt đầu predict!



- Lấy ngẫu nhiên một ảnh từ tập test và lưu đường dẫn.

```
# Lấy ngẫu nhiên một ảnh từ tập test của bộ dữ liệu
# Lưu đường dẫn vào biến img_path
%cd /content/gdrive/My Drive/colab/keras-yolo3
import os
test_images = [f for f in os.listdir('data_unzip/test') if f.endswith('.jpg')]
import random
img_path = "data_unzip/test/" + random.choice(test_images);

print(img_path)
```

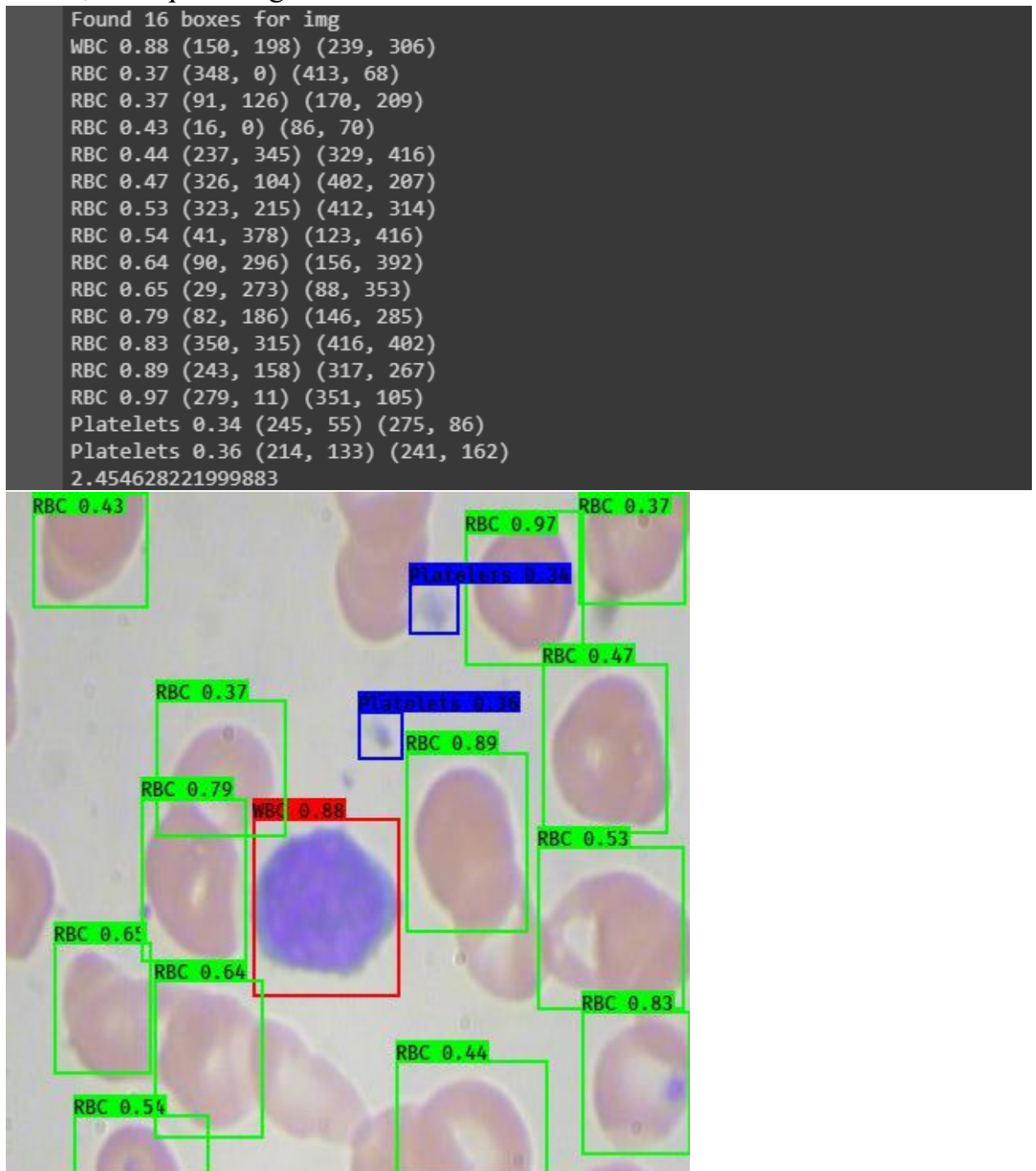
/content/gdrive/My Drive/colab/keras-yolo3  
data\_unzip/test/BloodImage\_00099\_jpg.rf.5b178d758af2a97d3df8e5f87b1f344a.jpg

- Gọi lệnh !python yolo\_video.py và truyền các tham số (đã được chú thích) như trong hình

```
# Thực hiện detect bằng bộ trọng số ta vừa train được
# --model: Đường dẫn đến mô hình
# --classes: Đường dẫn đến file classes
# --image: Cờ đánh dấu ta đang predict trên ảnh
# --image_path: Tham số này được viết thêm so với bản gốc, dùng để đưa đường dẫn đến ảnh input.

!python yolo_video.py --model="./logs/000/trained_weights_final.h5" --classes="classes.txt" --image --image_path={img_path}
from IPython.display import Image
Image('predictions.jpg')
```

- Ta được kết quả tương đối chính xác như bên dưới.



### 3. Nhận xét

- Quá trình train yolov4-pytorch nhanh hơn so với yolov3-keras vì bản keras phải train 2 giai đoạn và phải cần số epochs lớn hơn mới cho được kết quả tương đối tốt. Tuy nhiên tốc độ train còn phụ thuộc vào GPU mà ta sử dụng.
- Kết quả predict ở yolov4-pytorch có phần chính xác cao hơn so với ở yolov3-keras: Các đối tượng bị miss ít hơn, xác suất predict cao hơn.
- Đối với yolov4-pytorch, việc thay đổi số epochs=100 cho kết quả tốt hơn khi để epochs=50 (code mẫu).
- Đối với yolov3-keras, trải qua nhiều lần thay đổi các tham số như số epochs (đã thử train 500 epochs ở stage 1, batch\_size=2, 8, 16, 32, bỏ early\_stopping,...), learning rate (đã thử cho lr=0.01, lr=0.001, lr=0.0001 ở stage 1, lr=0.001 ở stage 2) và rút ra được số epochs=300 (cho stage 1), lr=0.001 (stage 1), batch\_size=32 (stage 1), lr=0.0001 (stage 2) và batch\_size=16 (stage 2) là các siêu tham số làm cho mô hình tương đối tốt và không quá phí thời gian training cũng như tránh overfitting (dù thiết lập số epochs lớn hơn, như 500 chẳng hạn, thì loss cũng hội tụ ở giai đoạn khoảng epoch thứ 300 chứ không giảm thêm nữa dù có train thêm).
- Kết quả ở cả hai mô hình yolov4-pytorch và yolov3-keras chỉ ở mức tạm chấp nhận được, còn một số đối tượng mô hình không detect được. Hạn chế này có thể là do dữ liệu còn quá ít hoặc các siêu tham số ta sử dụng chưa phải là tốt nhất vì cần rất nhiều thời gian để thử nghiệm.

**Hết**