HO CHI MINH UNIVERSITY OF SCIENCE
HO CHI MINH NATIONAL UNIVERSITY

FINAL SEMESTER PROJECT

Subject: Introduce to Machine learning

YOLO – ANIMAL DETECTION

COURSE CODE

CSC14005

Ho Chi Minh city – 2022

TABLE OF CONTENT

1. Overview

Student ID	Full name	Role
19127392	Tô Gia Hảo	Design UI + Collect dataset
19127525	Nguyễn Thanh Quân	Code UI + Code detection
19127625	Lâm Chí Văn	Design UI + Code Train + Train model

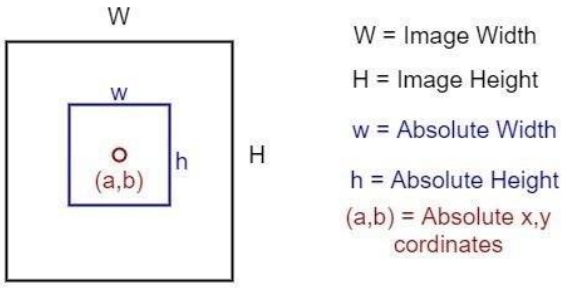
2. Detail requirement

2.1 Dataset

We has 8 class for training and 750 sample for each, for each sample, will label all objects of that class.

Each template includes a picture of the object and a txt file that stores the labels of those objects:

The format file txt:



YOLO FORMAT = <Class_id><a/W><b/H><w/W><h/H>

Dataset:

Google drive:
https://drive.google.com/file/d/1NgZWbYnBPoaVpxcZKXXvc_XCqYRurCdq/view?fbclid=IwAR0G21NcgL9Lhl0M4VDHdVG3bj-aYSKUzE_05oq2y5GZpjfXGuArniiaT1Y
Or Dropbox: https://dl.dropboxusercontent.com/s/9qihxeki243heqw/animal_best.weights?dl=0

2.2 Require 1:

We have done:

- Build web interface which allows insert an image and return object detection result

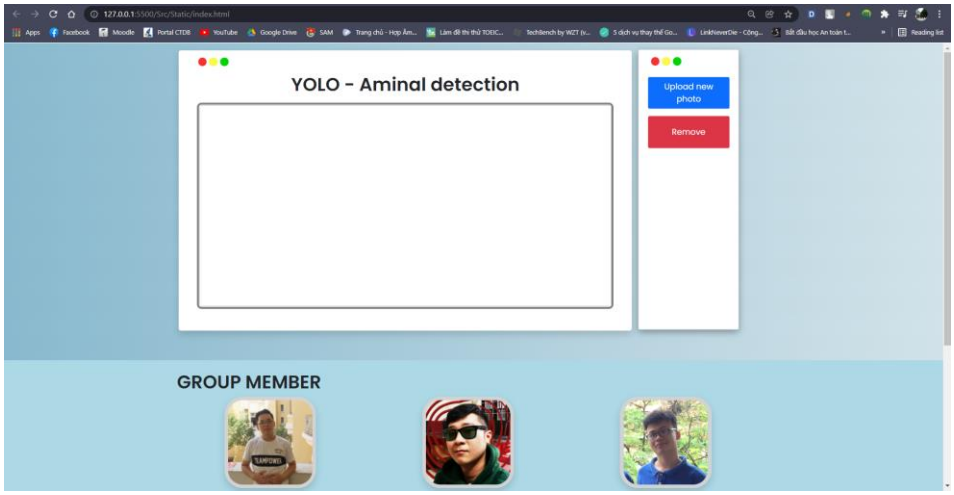


Figure 1. web interface

Up load new photo button allow to insert image

When we insert *Image*, it's return *object detection result*

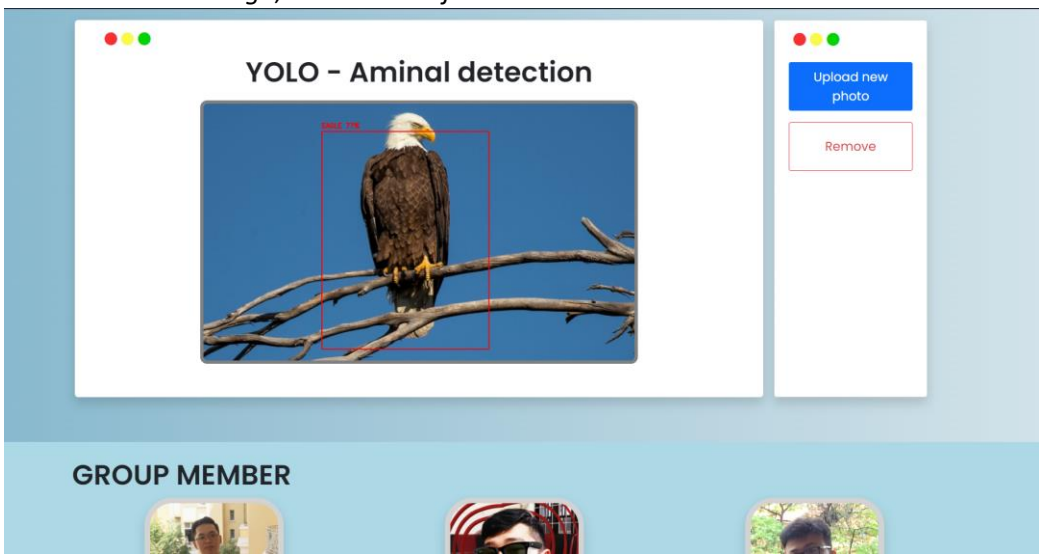


Figure 2: object detection result

- Install YOLO to detect object -> its can detect type of object and location of object

Reference:

- https://github.com/AlexeyAB/darknet
- https://pjreddie.com/darknet/yolo/

2.3 Require 2:

We have done:

- Build Desktop app run base on HTML, CSS, JavaScript and Python
- Build a complete app use for detect animal in picture

When you run code. First, it will load network from configuration and weight file

```
# -- Give the configuration and weight files for the model and load the network. --
print('>> Loading network....', end='')
net = cv2.dnn.readNetFromDarknet(modelConfig, modelWeights)
net.setPreferableBackend(cv2.dnn.DNN_BACKEND_OPENCV)
net.setPreferableTarget(cv2.dnn.DNN_TARGET_CPU)
print('Done!')
```

When you click **Up load new photo** button and choose a picture, it willcall function `onChange="uploadImg(this)"` in JavaScript. Function uploadImage will load and show the picture then call function `eel.animalDetection(image_b64)` in python by **eel,** this function just get image in base 64 then it will you YOLO to detect object. After that, the result image will save and show for us view

Data

- Train: 60 %
- Test: 20 %
- Validation: 20 %

Average Precision

Class ID	Name	TP	FP
0	Cow	143	68
1	Buffalo	223	81
2	Cat	114	22
3	Dog	105	71
4	Deer	153	41
5	Lion	126	13
6	Eagle	120	25
7	Tiger	115	22

F1-score & Average IoU & mAP

conf_thresh	precision	recall	F1-score	TP	FP	FN	average IoU	mAP
0.25	0.88	0.67	0.71	1099	343	540	57.82%	75%

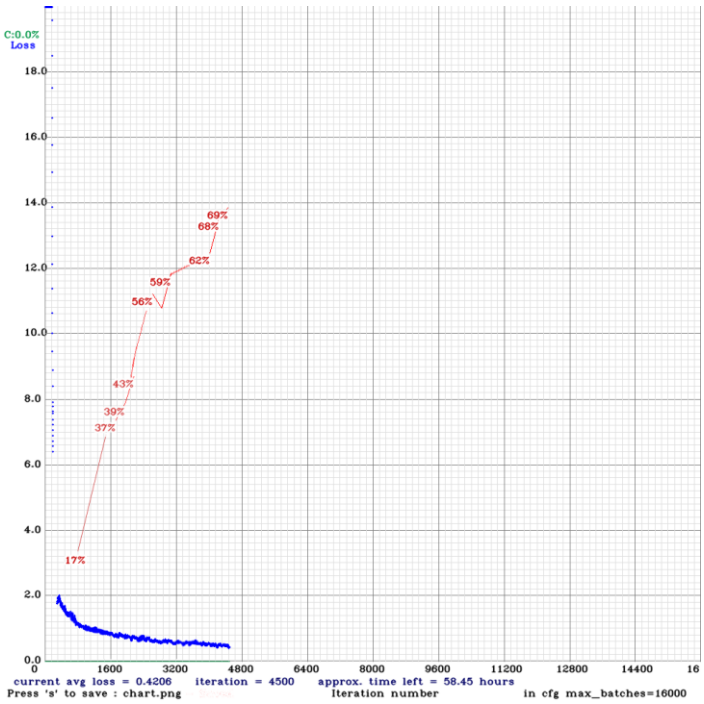


Figure 3. Chart average loss vs mAP

Reference:

https://youtube.com/playlist?list=PLMoSUbG1Q_r8nz4C5Yvd17KaXy8p0ufPH
https://www.youtube.com/watch?v=iy2aKf9AAvc&t=642s
https://www.youtube.com/watch?v=p3Uu9rVg5-Y

3. Instruction for running code:

Model:

- This application, we use model yolov3 for training:
- We do a some change on this config, changes batch size to 64 and max batch to 16000 step to 12800, 14400
- Change number of class in yolo layer to 8 which is same as object needed to train.
- Decrese ignore_thresh value in yolo layer to .5. This parameter is used to determines whether the IOU error needs to be calculated is greater than thresh.

3.1 Train model:

Step 1: open **Training_model.ipynb** with **google colab** running with GPU

Step 2: Decide and configure model yolo to train

Step 3: Run all cell till connect to google drive, my group use google drive to store dataset so connect to it.

Step 4: split dataset into train – validate – test

Step 5: Train model with command

```
!./darknet detector train object.data animal.cfg animal_last.weights -
dont_show -map
```

- If you have pretrained weight put it name into “animal_last.weights”
- Else, you just leave it empty
- -map is extension for calculate mAP (mean average precision) every considerable iteration.
- Dont show is used for disable window loss function

Step 6: When to stop training? When you see the average loss no longer decreases at many iteration then you should stop training. Or mAP doesn’t increase anymore.

Step 7: While you are training model, darknet will create some weight file each time application calculate mAP. Sometime, the more you train does not mean it does better. You must run test mAP on each weight with this command.

Step 8: choose the best weight with best mAP and using it for detect.

```
!./darknet detector map object.data animal.cfg animal_last1000.weights

!./darknet detector map object.data animal.cfg animal_last2000.weights
```

How much does it take for training: Usually you can stop training after 2000 iteration, however sometime it takes to 5000 ~ 9000 iteration for training.

On average, how much time does each 100 iteration take: If you enable GPU of google colab, each 100 iteration takes 15 ~ 20 minutes.

3.2 Detect animal:

Step 1: Run python animalDetection.py in Animal-Detection\Src

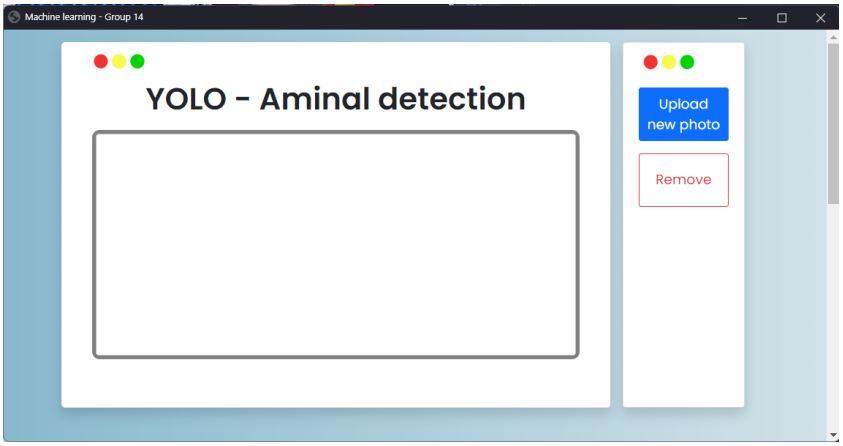
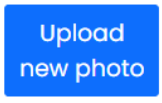


Figure 4: app interface

. It will auto download file weight, you need not do anything, just run

Step 2: Choose Image by click **upload new photo** button



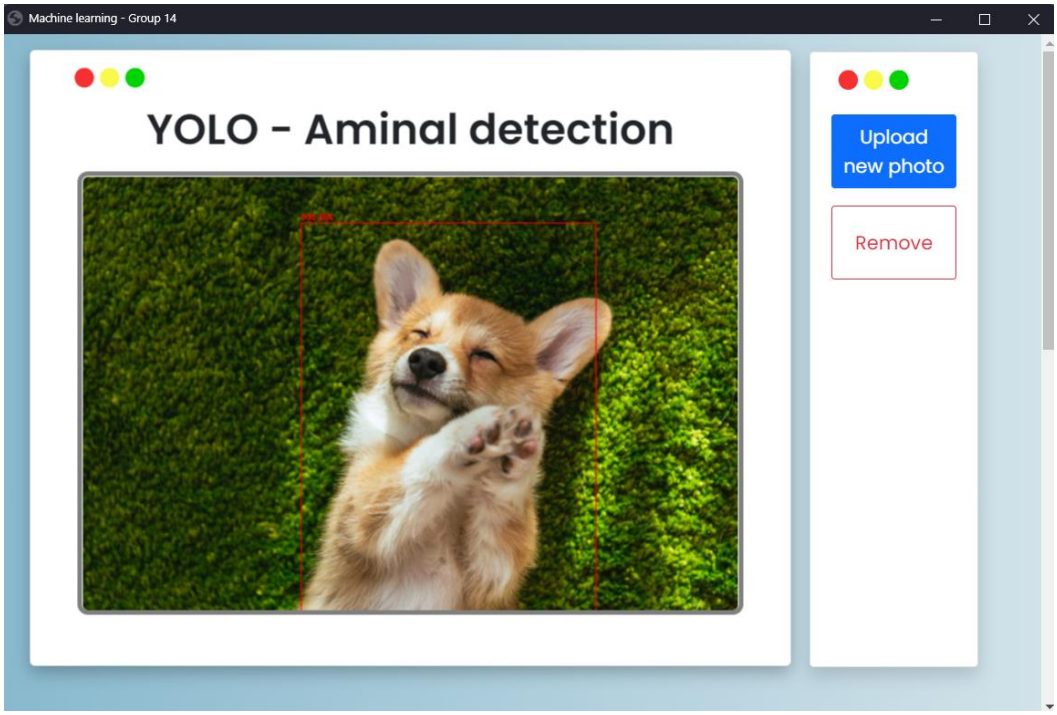


Figure 5: object detection result

4. Detail code

4.1 Training code (Training_model.ipynb)

Clone Darknet

```
!git config --global --unset http.proxy
!git clone https://github.com/AlexeyAB/darknet
```

```
Cloning into 'darknet'...
remote: Enumerating objects: 15386, done.
remote: Total 15386 (delta 0), reused 0 (delta 0), pack-reused 15386
Receiving objects: 100% (15386/15386), 14.01 MiB | 17.18 MiB/s, done.
Resolving deltas: 100% (10345/10345), done.
```

Using to clone github which is used to train model YOLO

Compile

```
# change makefile to have GPU and OPENCV enabled
%cd 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
```

```
/content/darknet
```

Change makefile content:

OPENCV = 1: enable opencv2

GPU = 1: enable training with GPU

CUDNN = 1: enable to build with cuDNN v5-v7 to accelerate training by using GPU

CUDNN_HALF = 1: speedup Detection 3x, Training 2x

LIBSO = 1: to build a library `darknet.so` and binary runnable file `uselib` that uses this library

=> makefile with command !make

CONNECT TO DRIVE

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

```
Mounted at /content/gdrive
```

```
[ ] %cd /content
    !git clone https://github.com/QuanBlue/Animal-Detection
```

```
/content
Cloning into 'Animal-Detection'...
remote: Enumerating objects: 95, done.
remote: Counting objects: 100% (95/95), done.
remote: Compressing objects: 100% (72/72), done.
remote: Total 95 (delta 31), reused 77 (delta 18), pack-reused 0
Unpacking objects: 100% (95/95), done.
```

```
[ ] %cd /
    from glob import glob
    dataset_dir = "/content/gdrive/MyDrive/animal detection/"

    img_list = glob(dataset_dir+'*.jpg')
    # you should have images with labels.txt in same folder
    print("your images :",len(img_list))
```

Connect to drive to take dataset to train, read all image in dataset with extension .jpg

Clone our repository to load some neccesary file (ex: cgf, obj.data, name)

```
[ ] from sklearn.model_selection import train_test_split

    train_img_list = img_list

    train_img_list, val_img_list = train_test_split(img_list, test_size=0.2, random_state=42)

    train_img_list, test_img_list = train_test_split(train_img_list, test_size=0.25, random_state=42)    #0.8 * 0.25 = 0.2

    print(len(train_img_list), len(val_img_list), len(test_img_list))
```

Split data set into train – validate – test with percentage is 60 – 20 – 20

Calculate mAP of that weight

```
[ ]  !./darknet detector map /content/Animal-Detection/config/object.data\
                             /content/Animal-Detection/config/animal.cfg\
                             /content/gdrive/MyDrive/animal_last.weights\
```

- First parameter contain object.data with contain each object name, location of train and validatet dataset.
- Second parameter is model YOLO
- Last parameter is weight you want to calculate map

Detect image and show it result to colab

```
!./darknet detector test /content/Animal-Detection/config/object.data\
                         /content/Animal-Detection/config/animal.cfg\
                         /content/gdrive/MyDrive/animal_best.weights\
imShow('predictions.jpg')
```

- First three parameter is same as above, however you can add path of image you want to detect.
- If you don't put anything to fourth parameter, when you are running cell, colab will require you enter path of file you want to detect

4.2 Detect code (main) (animalDetection.py)

First, it will check whether file weight is exist, if not, it will auto download file weight

```
# -- download file weight --
print('>> Check file weight....', end=' ')
isExist = os.path.isfile(modelWeights)
print(isExist,'!', sep='')

if isExist == False:
    print('--> Downloading file weight....')
    download(weights_url, modelWeights)
    print('--> Download complete!')
```

After that, it will read class name

```
# -- read class name --
classNames = []
with open(classFiles, 'rt') as f:
    print('>> Reading class name....', end = ' ')
    classNames = f.read().rstrip('\n').split('\n')
    print('Done!')
```

Then load network from configuration and weight file

```
# -- Give the configuration and weight files for the model and load the network.
print('>> Loading network....', end='')
net = cv2.dnn.readNetFromDarknet(modelConfig, modelWeights)
net.setPreferableBackend(cv2.dnn.DNN_BACKEND_OPENCV)
net.setPreferableTarget(cv2.dnn.DNN_TARGET_CPU)
print('Done!')
```

After load everything, program call index.html to show App interface and ready to use

```
eel.start('index.html', size=(1010, 680))
```

3 function use for detect object

```
def readImage(img_b64):
```
This function use to read Image receive from JavaScript in base64 and return cv2 image and blob image

```
def findObject(outputs, img):
```
This function use to find object and find locate of object with input is network output and cv2 image

```
def animalDetection(img_b64):
```
This is main function, it call 2 function above to detect object

5. Necessary files

File weights:
https://drive.google.com/file/d/1NgZWbYnBPoaVpxcZKXXvc_XCqYRurCdq/view?fbclid=IwAR0G21NcgL9Lhl0M4VDHdVG3bj-aYSKUzE_05oq2y5GZpjfXGuArniiaT1Y

6. Reference

https://github.com/AlexeyAB/darknet
https://pjreddie.com/darknet/yolo/
https://youtube.com/playlist?list=PLMoSUbG1Q_r8nz4C5Yvd17KaXy8p0ufPH
https://www.youtube.com/watch?v=iy2aKf9AAvc&t=642s
https://www.youtube.com/watch?v=p3Uu9rVg5-Y

Video demo: https://youtu.be/zr5RVgPJhAU
For more access: https://github.com/QuanBlue/Animal-Detection.git