



# GUNS DETECTION YOLO4

---

*Sinh viên thực hiện:*  
*Lê Thanh Giang 16521660*  
*Bùi Thế Duy 16521604*  
*Nguyễn Duy Minh 16521735*

---

# NỘI DUNG

1. Tổng quan
2. Các kỹ thuật sử dụng
3. Quá trình thực hiện
4. Kết quả
5. Tài liệu tham khảo



01

Tổng Quan



## 1.1 Giới thiệu đề tài

Yolo là mô hình CNN cho việc phát hiện, nhận dạng, phân loại đối tượng. Yolo được tạo ra từ việc kết hợp giữa các convolutional layers và connected layers. Trong đó các convolutional layers sẽ trích xuất ra các feature của ảnh, còn full-connected layers sẽ dự đoán ra xác suất đó và tọa độ của đối tượng.

Detection súng ngắn trong ảnh và video với bộ dữ liệu là 3000 ảnh với labels có sẵn

## 1.2 Chuẩn bị dữ liệu

Data download:

<https://www.kaggle.com/atulyakumar98/gundetecion>



Train

2700 ảnh



Validate

150 ảnh



Test

150 ảnh



02

## Các kỹ thuật sử dụng

# Các kỹ thuật sử dụng

Sử dụng Yolo 4 detect your custom objects





03

Các bước thực  
hiện



# Cấu trúc dữ liệu input

Một .txt file ứng với mỗi ảnh .jpg cùng tên và đặt trong cùng 1 thư mục. Thông tin trong mỗi file .txt gồm có số lượng object và thành độ của object ở trong ảnh, ứng với mỗi object là một dòng: <object-class> <x> <y> <width> <height>

```
0 0.272917 0.355172 0.537500 0.696552  
0 0.760417 0.658621 0.470833 0.682759
```



## 3.1 Tiền xử lý

Load ảnh và resize về đúng kích thước (228,228)

Chuẩn bị dataset với file train.txt, test.txt và valid.txt được chia và trộn ngẫu nhiên từ 3000 ảnh cho trước

```
23
24 np.random.shuffle(result)
25 train_file = open("build/darknet/x64/data/train.txt","w+")
26 valid_file = open("build/darknet/x64/data/valid.txt","w+")
27 test_file = open("build/darknet/x64/data/test.txt","w+")
28 for i in range(0,2700):
29     train_file.write(result[i]+"\\n")
30 for i in range(2700,2850):
31     valid_file.write(result[i]+"\\n")
32 for i in range(2850,3000):
33     test_file.write(result[i]+"\\n")
34
```



## 3.2 Cấu hình file cfg/yolov4-custhànhm.cfg

Thay đổi dòng **batch** thành **batch=64**

Thay đổi dòng **subdivisions** thành **subdivisions=16**

Thay đổi dòng **max\_batches** thành **(classes\*2000)**

Cài đặt network size width=288 height=288 or any value multiple of 32:

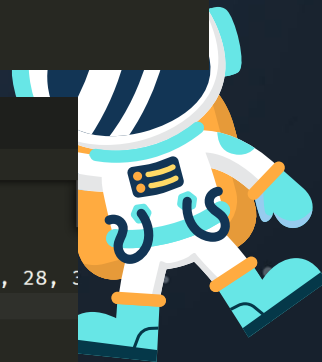
Thay đổi dòng **classes=80** thành your number of objects in each of 3 [yolo]-layers

Thay đổi [filters=255] thành **filters=(classes + 5)x3** in the 3 [convolutional] before each [yolo] layer,

```
yolov4-custom (1).cfg
D: > Yolo_gun > yolov4-custom (1).cfg
1  [[net]]
2  # Training
3  batch=64
4  subdivisions=32
5  width=288
6  height=288
```

```
1131
1132 [convolutional]
1133 size=1
1134 stride=1
1135 pad=1
1136 filters=24 //filters=(classes + 5)x3
1137 activation=linear
1138
1139
1140 [yolo]
```

```
yolov4-custom.cfg X
D: > Yolo_gun > yolov4-custom.cfg
1142
1143 [yolo]
1144 mask = 6,7,8
1145 anchors = 12, 16, 19, 36, 40, 28, 3
1146 classes=1
1147 num=9
1148ITTER= 3
```





### 3.3 Thực hiện train

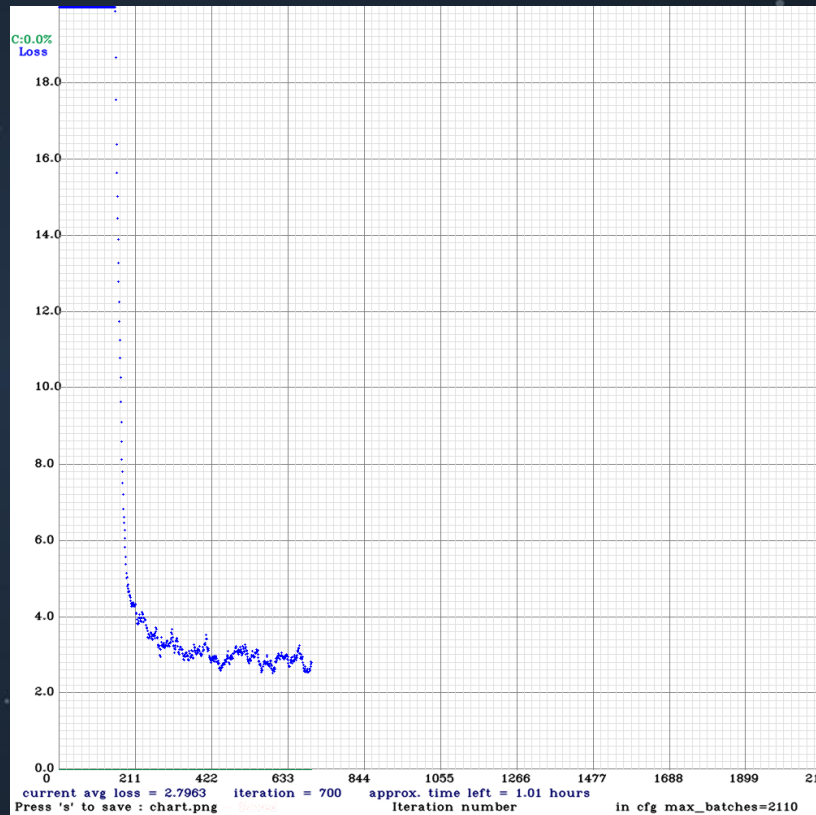
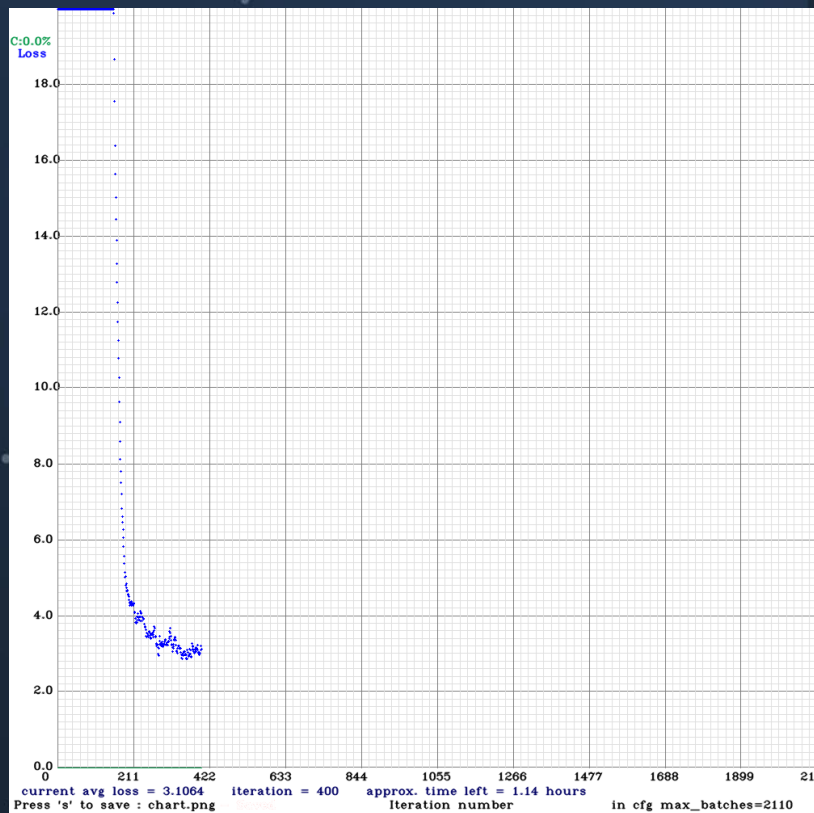
```
1 !./darknet detector train obj.data cfg/yolov4-custom.cfg -dont_show
```

126	conv	256	1 x 1/ 1	18 x 18 x 512	->	18 x 18 x 256	0.085	BF
127	conv	128	1 x 1/ 1	18 x 18 x 256	->	18 x 18 x 128	0.021	BF
128	upsample		2x	18 x 18 x 128	->	36 x 36 x 128		
129	route	54			->	36 x 36 x 256		
130	conv	128	1 x 1/ 1	36 x 36 x 256	->	36 x 36 x 128	0.085	BF
131	route	130 128			->	36 x 36 x 256		
132	conv	128	1 x 1/ 1	36 x 36 x 256	->	36 x 36 x 128	0.085	BF
133	conv	256	3 x 3/ 1	36 x 36 x 128	->	36 x 36 x 256	0.764	BF
134	conv	128	1 x 1/ 1	36 x 36 x 256	->	36 x 36 x 128	0.085	BF
135	conv	256	3 x 3/ 1	36 x 36 x 128	->	36 x 36 x 256	0.764	BF
136	conv	128	1 x 1/ 1	36 x 36 x 256	->	36 x 36 x 128	0.085	BF
137	conv	256	3 x 3/ 1	36 x 36 x 128	->	36 x 36 x 256	0.764	BF
138	conv	18	1 x 1/ 1	36 x 36 x 256	->	36 x 36 x 18	0.012	BF
139	yolo							

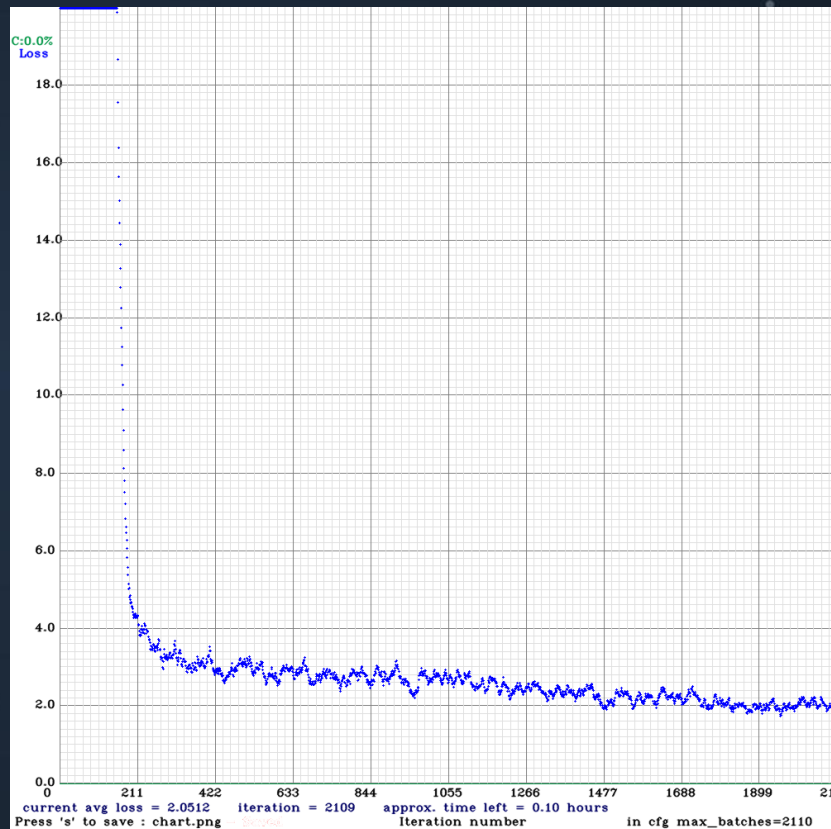
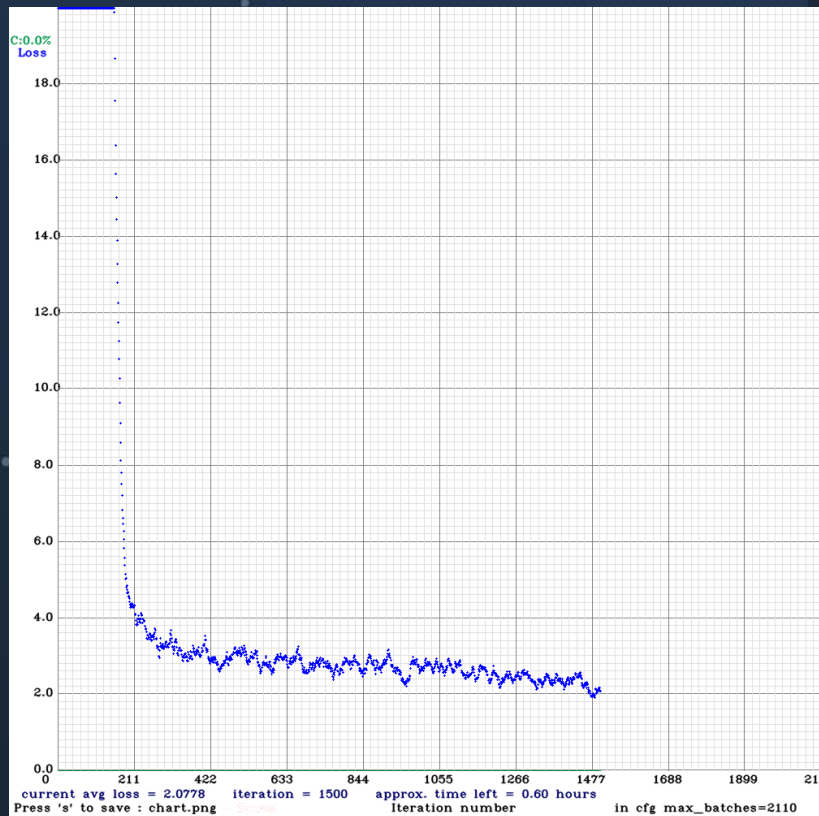
[yolo] params: iou loss: ciou (4), iou\_norm: 0.07, cls\_norm: 1.00, scale\_x\_y: 1.20  
nms\_kind: greedy (1), beta = 0.600000

140	route	136			->	36 x 36 x 128		
141	conv	256	3 x 3/ 2	36 x 36 x 128	->	18 x 18 x 256	0.191	BF
142	route	141 126			->	18 x 18 x 512		












# Biểu đồ trong quá trình train

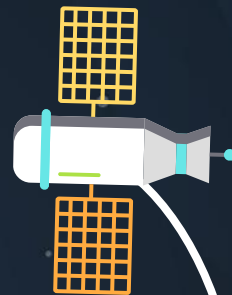


# Biểu đồ trong quá trình train



## 3.4 Save Modal

Name ↑	Owner	Last modified	File size
 modal_bbk	me	Jul 10, 2020 me	—
 yolov4-custom_1000.weights 	me	Jul 10, 2020 me	244 MB
 yolov4-custom_2000.weights 	me	Jul 10, 2020 me	244 MB
 yolov4-custom_final.weights 	me	Jul 10, 2020 me	244 MB
 yolov4-custom_last.weights 	me	Jul 10, 2020 me	244 MB
 yolov4.conv.137 	me	Jul 5, 2020 me	162 MB





04

Đánh giá và kết  
quả



## 4.1 Thực hiện dự đoán trên 150 ảnh output txt file

```
[ ] 1 test obj.data cfg/yolov4-custom.cfg backup/yolov4-custom_last.weights -dont_show -ext_output <build/darknet/x64/data/test.txt> results_11_07.txt
```

→ CUDA-version: 10010 (10010). cuDNN: 7.6.5. CUDNN\_HALF=1. GPU count: 1

OpenCV version: 3.2.0

0 : compute\_capability.py

layer filters size

0 conv 32

1 conv 64 4-custor

results\_11\_07.txt X

```
1 CUDNN_HALF=1
2 net.optimized_memory = 0
3 mini_batch = 1, batch = 32, time_steps = 1, train = 0
4 nms_kind: greedy_nms (1), beta = 0.600000
5 nms_kind: greedy_nms (1), beta = 0.600000
6 nms_kind: greedy_nms (1), beta = 0.600000
7
8 seen 64, trained: 134 K-images (2 Kilo-batches_64)
9 Enter Image Path: Detection layer: 139 - type = 27
10 Detection layer: 150 - type = 27
11 Detection layer: 161 - type = 27
12 ../dataset/datasets-full/armas (1630).jpg: Predicted in 16.941000 milli-seconds.
13 gun: 68% (left_x: 23 top_y: -15 width: 452 height: 306)
14 Enter Image Path: Detection layer: 139 - type = 27
15 Detection layer: 150 - type = 27
16 Detection layer: 161 - type = 27
17 ../dataset/datasets-full/armas (2177).jpg: Predicted in 16.944000 milli-seconds.
18 gun: 99% (left_x: -15 top_y: 3 width: 186 height: 116)
19 Enter Image Path: Detection layer: 139 - type = 27
20 Detection layer: 150 - type = 27
21 Detection layer: 161 - type = 27
22 ../dataset/datasets-full/armas (2144).jpg: Predicted in 16.953000 milli-seconds.
23 gun: 29% (left_x: 45 top_y: 41 width: 921 height: 664)
24 Enter Image Path: Detection layer: 139 - type = 27
25 Detection layer: 150 - type = 27
26 Detection layer: 161 - type = 27
27 ../dataset/datasets-full/armas (1792).jpg: Predicted in 16.754000 milli-seconds.
28 gun: 39% (left_x: 67 top_y: 63 width: 1317 height: 928)
```

## 4.2 Thực hiện dự đoán trên 1 ảnh output images

```
1 !./darknet detector test obj.data cfg/yolov4-custom.cfg backup/yolov4-custom_last.weights -thresh 0.20 images/gen3.jpg
2 |
```

```
[ ] 1 #display the result
2 def display_image(file_path = '/content/drive/My Drive/Colab Notebooks/detec_gun_new/darknet/predictions.jpg'):
3     import cv2
4     import matplotlib.pyplot as plt
5     import os.path
6
7     if os.path.exists(file_path):
8         img = cv2.imread(file_path)
9         show_img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
10        plt.imshow(show_img)
11    else:
12        print('failed to open file')
13
14    display_image()
```



## 4.3 Thực hiện dự đoán trên video output mp4 file

```
[ ] 1 !./darknet detector demo obj.data cfg/yolov4-custom.cfg backup/yolov4-custom_last.weights -thresh 0.20 -dont_show images/gunvid_02.mp4 -out
```

Streaming output truncated to the last 5000 lines.

gun: 36%

gun: 30%

FPS:20.5

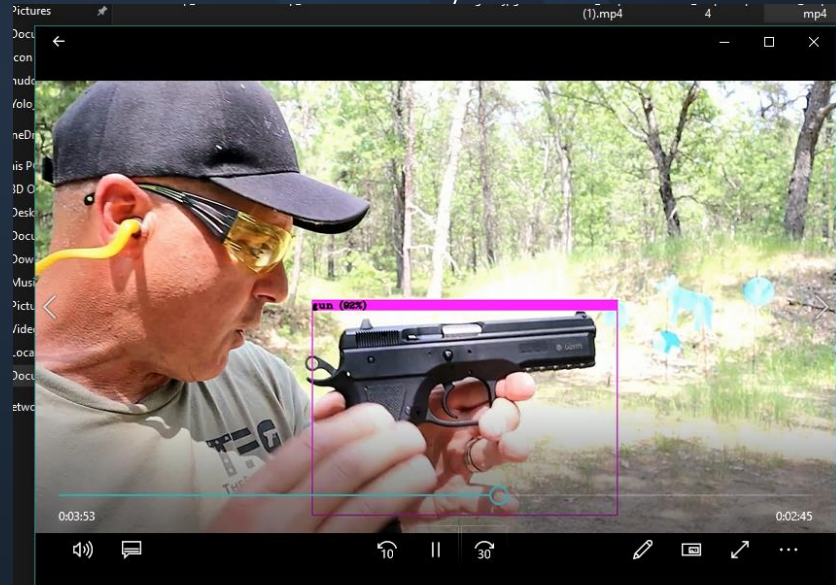
AVG\_FPS:20.9

cvWriteFrame

Objects:

gun: 91%

gun: 46%



## 4.4 Thực hiện dự đoán trên video output mp4 file

## 5. Tài liệu tham khảo



[1]<https://github.com/AlexeyAB/darknet#how-to-train-to-detect-your-custom-objects>

[2]<https://github.com/rafaelpadilla/Object-Detection-Metrics#create-the-ground-truth-files>

[3]<https://www.miai.vn/2019/08/26/yolo-series-thu-lam-he-thong-chong-khung-bo-gun-detection-phat-hien-sung/>

[4]<https://neurohive.io/en/popular-networks/alexnet-imagenet-classification-with-deep-convolutional-neural-networks/>

[5] Bài giảng môn học Thị giác máy tính nâng cao – Thầy Lê Minh Hưng - <https://web.microsoftstream.com/video/1236ba9c-59ab-4be9-a9e0-ba6f287ffa99>

THANKS FOR WATCHING

