



Trường Đại học Công nghệ Thông tin

Khoa Khoa học máy tính

ĐỒ ÁN MÔN HỌC

Máy học trong thị giác máy tính

Đề tài: Nhận dạng vật thể trong ban đêm



Author:

Lê Nhất Minh

Nguyễn Khánh Vinh

Teacher:

Dũng

Mai Tiến

TP.Hồ Chí Minh, năm 2019-2020

CONTENTS

1 Giới thiệu	4
1.1 Mô tả bài toán	4
1.2 Tầm quan trọng của bài toán	5
1.3 Những điểm nổi bật của phương pháp	6
2 Các nghiên cứu liên quan	7
3 Phương pháp đề xuất	8
3.1 Tổng quan về R-CNN và các biến thể	8
3.1.1 R-CNN (Region-based Convolutional Neural Networks)	9
3.1.2 Fast R-CNN	9
3.1.3 Faster R-CNN	11
3.2 Mask R-CNN:	12
3.2.1 Giới thiệu tổng quan	12
3.2.2 RoI Align	14
3.2.3 Các bước thực hiện	15
4 Thực nghiệm	21
4.1 Tập dữ liệu	21
4.2 Thiết lập thực nghiệm	23
4.2.1 Cấu hình thiết bị thực nghiệm	23
4.2.2 Các công cụ cần thiết	23
4.2.3 Quá trình thực nghiệm	23
4.3 Kết quả thực nghiệm	32
4.3.1 Kết quả trên tập dữ liệu huấn luyện	32
4.3.2 Kết quả test	33
4.4 Bảng thống kê	34
4.5 Thảo luận	37
5 Kết luận và Hướng phát triển	38
5.1 Kết luận	38
5.2 Hướng phát triển	40

1

GIỚI THIỆU

1.1 MÔ TẢ BÀI TOÁN

Trong đời sống hằng ngày, chúng ta luôn phải đối diện với những mối nguy hiểm rình rập đặc biệt là những tai nạn bất ngờ về đêm khi tầm nhìn bị giới hạn cũng như mắt tập trung trong ban đêm. Thực tế cho thấy, những hậu quả từ những tai nạn về đêm thì người ảnh hưởng không ai khác đó chính là bản thân mình và những người xung quanh mình.



Figure 1: Tai nạn giao thông đa số bị tầm nhìn hạn chế

Chính vì thế, với sự phát triển của máy tính thì con người có thể hỗ trợ của máy tính có thể giúp con người có tầm nhìn cao hơn trong đêm.



Figure 2: Những nguy cơ tiềm tàng do thiếu tầm nhìn trong ban đêm sẽ dần được xoá bỏ

Trong các hướng tiếp cận của Thị giác máy tính bao gồm Recognition (Nhận dạng), Reconstruction (Dựng lại, tái kiến thiết), Registration (Theo dõi hoặc liên kết), Reorganization (Tổ chức lại: Học không giám sát) nhóm nghiên cứu quyết định chọn hướng nghiên cứu về Recognition (Nhận dạng) với đề tài "Nhận dạng vật thể trong ban đêm".

Bộ dữ liệu sẽ gồm có tập Input và Output như sau:

- Input: Tập dữ liệu ảnh được cắt từ video trong ban đêm với điều kiện đủ sáng
- Output: Video đã xác định được vật thể

Thông qua việc tiếp cận và chọn lọc tập dữ liệu hình ảnh đầu vào, ta sẽ áp dụng các phương pháp Machine Learning nhằm xây dựng mô hình huấn luyện tập dữ liệu giúp máy tính có thể nhìn được trong đêm và nhận diện được các đối tượng trước mắt mà còn người đôi khi bị hạn chế thiếu tập trung trong đêm giúp tránh được các tai nạn đáng tiếc xảy ra trong tương lai

1.2 TẦM QUAN TRỌNG CỦA BÀI TOÁN

Trong thời đại mà các thiết bị phần cứng ngày càng phát triển nhanh chóng như vi xử lý, camera, card đồ họa, ... việc áp dụng các công nghệ từ Thị giác máy

tính và Máy học vào đời sống thường ngày giúp con người có thể thực hiện được những điều trước đây chưa từng làm được. Khi tham gia giao thông trong ban đêm với điều kiện ánh sáng bị hạn chế, máy tính sẽ giúp con người có thể nhìn xa hơn với tầm nhìn rộng hơn, có thể xác định được những đối tượng đang trên đường nhằm cảnh báo để tránh được những tai nạn giao thông trong ban đêm đáng tiếc.

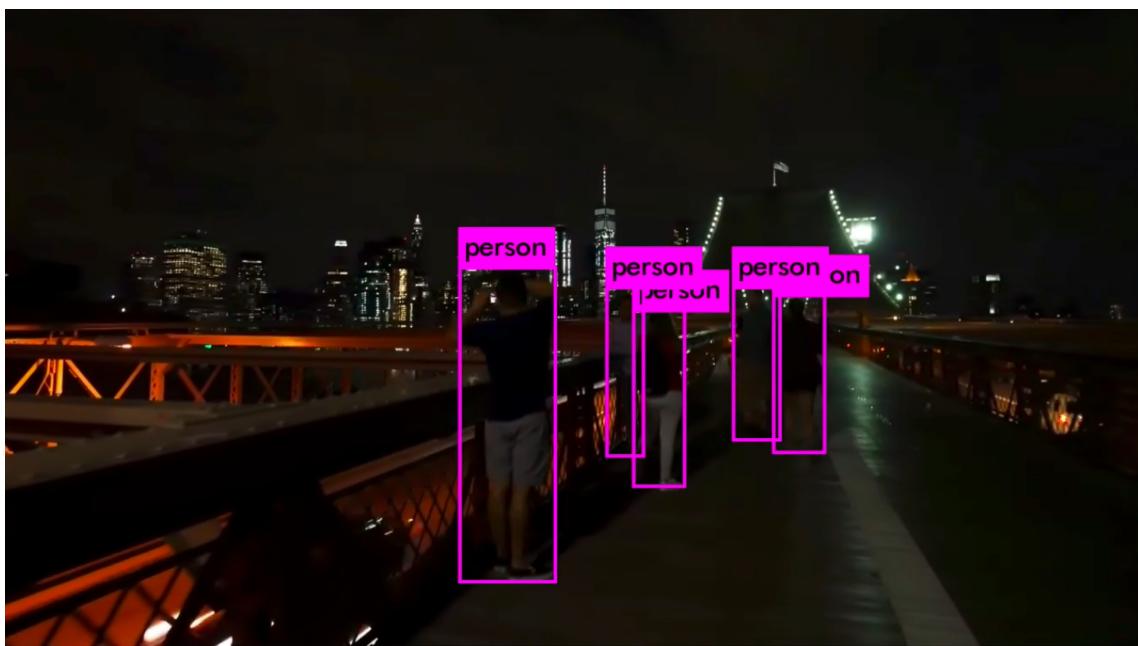


Figure 3: Từ lý thuyết đến thực tế (Ảnh: Greg (Grzegorz) Kroczeck)

Từ ý tưởng và ứng dụng này ta có thể mở ra thêm nhiều hướng mới như phát triển đa nền tảng như nhận dạng trực tiếp trên camera hành trình / camera an ninh, cảnh báo người dùng từ thiết bị di động, kết nối với các thiết bị không dây IoT hỗ trợ người dùng khi đang tham gia giao thông, ...

1.3 NHỮNG ĐIỂM NỔI BẬT CỦA PHƯƠNG PHÁP

	Mask R-CNN	YOLO
Mục đích:	Object Detection + Segmentation	Object Detection
Cài đặt:	Cấu trúc phức tạp	Cấu trúc đơn giản hơn
Độ nhiễu:	Ít, chỉ tập trung vào đối tượng được yêu cầu nhận diện	Cao, nhận diện luôn các đối tượng không được đề cập

2

CÁC NGHIÊN CỨU LIÊN QUAN

[1] Computer Vision Tutorial: Implementing Mask R-CNN for Image Segmentation.

Backbone Model: tương tự như **ConvNet** mà chúng tôi sử dụng trong Faster R-CNN để trích xuất các bản đồ đặc trưng từ hình ảnh, chúng tôi sử dụng kiến trúc ResNet 101 để trích xuất các tính năng từ hình ảnh trong Mask R-CNN. Vì vậy, bước đầu tiên là chụp ảnh và trích xuất các tính năng bằng kiến trúc ResNet 101. Các tính năng này hoạt động như một đầu vào cho lớp tiếp theo.

Region Proposal Network (RPN): Lấy các bản đồ tính năng thu được ở bước trước và áp dụng mạng để xuất khu vực (RPM). Điều này về cơ bản dự đoán nếu một đối tượng có mặt trong khu vực đó (hoặc không). Trong bước này, chúng ta có được các vùng hoặc bản đồ đặc trưng mà mô hình dự đoán có chứa một số đối tượng.

Region of Interest (RoI): Các khu vực thu được từ RPN có thể có hình dạng khác nhau. Do đó, chúng tôi áp dụng một lớp gộp và chuyển đổi tất cả các vùng thành hình dạng giống nhau. Tiếp theo, các vùng này được chuyển qua một mạng được kết nối đầy đủ để dự đoán nhãn lớp và hộp giới hạn.

[2] Mask R-CNN for Object Detection and Segmentation.

3

PHƯƠNG PHÁP ĐỀ XUẤT

Phát hiện vật thể (Object Detection) là một trong những bài toán phổ biến của lĩnh vực thị giác máy tính, với sự phát triển vượt bật của khoa học công nghệ định vị một hoặc nhiều đối tượng trong một hình ảnh và phân loại đối tượng có trong ảnh.

Đối với vấn đề phát hiện vật thể thì có rất nhiều phương pháp khác nhau như YOLO, Convolutional Neural Networks (CNN), Region-based Convolutional Neural Networks (R-CNN), Viewpoint Feature Histogram (VFH), Fast Point Feature Histogram (FPFH),... và trong bài này, chúng ta sẽ sử dụng phương pháp Mask R-CNN.

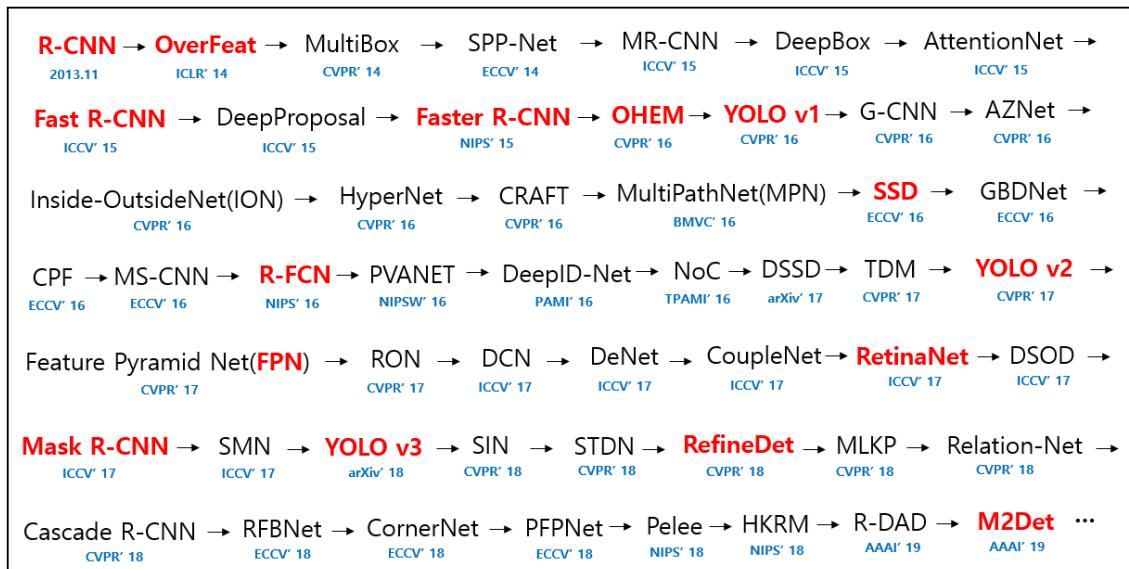


Figure 4: Image recognition: lịch sử phát triển của deep learning

3.1 TỔNG QUAN VỀ R-CNN VÀ CÁC BIẾN THẾ

3.1.1 R-CNN (REGION-BASED CONVOLUTIONAL NEURAL NETWORKS)

R-CNN được phát triển với ý tưởng chính được chia làm 2 giai đoạn chính:

1/ Sử dụng phương pháp Selective Search (Tìm kiếm chọn lọc) để đưa ra các bounding boxes, hay còn gọi là Region Proposals (Sau này có mạng Neural Network dành riêng cho Region Proposal - **Region Proposal Network (RPN)**: Cốt lõi Faster R-CNN), chứa các vùng có thể có vật thể ở trong.

2/ Sử dụng các mạng đã được huấn luyện sẵn như **AlexNet**, **VGG-16** để tính toán feed-forward các regions thu được convolutional features của mỗi region, sau đó huấn luyện SVM từ đó với mỗi bounding box ta xác định xem nó là đối tượng nào (người, ô tô, xe đạp,...)

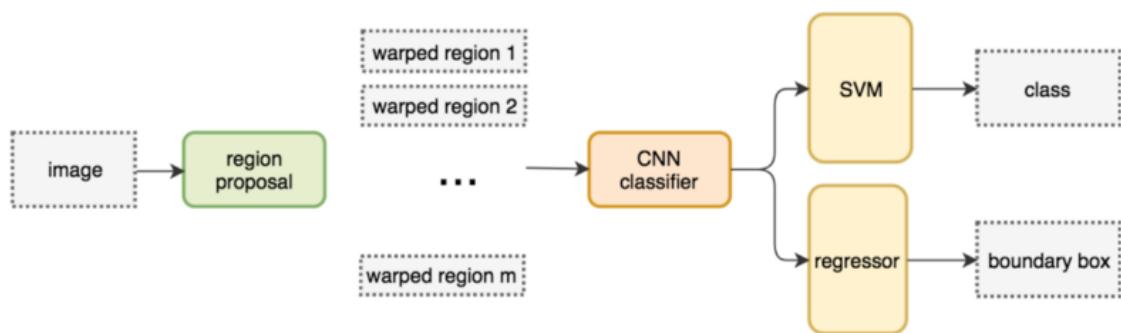


Figure 5: Nguyên lý hoạt động của mạng R-CNN (Nguồn: Medium)

Hiện tại phương pháp R-CNN vẫn có nhiều hạn chế:

- Vì với mỗi ảnh ta cần phân loại các class cho 2000 region proposal nên thời gian train rất lâu.
- Không thể áp dụng cho real-time thì mỗi ảnh trong test set mất tới 47s để xử lý. Với việc sử dụng ít tấm ảnh nhỏ hơn, và chất lượng của mỗi tấm ảnh nhỏ tốt hơn, Mạng R-CNN chạy nhanh hơn và có độ chính xác cao hơn so với mô hình sử dụng **Cửa sổ trượt (sliding windows)**

3.1.2 FAST R-CNN

Fast R-CNN được giới thiệu bởi cùng tác giả của R-CNN (Ross Girshick), nó giải quyết được một số hạn chế của R-CNN để cải thiện tốc độ.

Tương tự như R-CNN thì Fast R-CNN vẫn dùng selective search để lấy ra các region proposal. Tuy nhiên là nó không tách 2000 region proposal ra khỏi ảnh

và thực hiện bài toán image classification cho mỗi ảnh. Thay vì phải rút trích đặc trưng của mỗi proposal, chúng ta có thể dùng CNN rút trích đặc trưng của toàn bộ bức ảnh trước (được feature map), đồng thời rút trích các proposal, lấy các proposal tương ứng trên feature map, rescale và cuối cùng là phân lớp và tìm vị trí của object. Với việc không phải lặp lại 2000 lần việc rút trích đặc trưng, Fast R-CNN giảm thời gian xử lý một cách đáng kể.

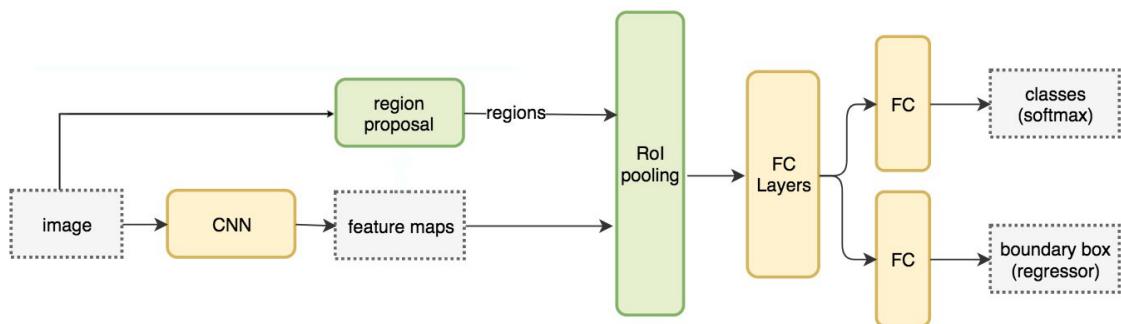


Figure 6: Nguyên lý hoạt động của mạng Fast R-CNN (Nguồn: [Medium](#))

Với việc không phải lặp đi lặp lại quá trình tìm ra các proposal, tốc độ của thuật toán tăng lên kha khá. Trong thực nghiệm, mô hình Fast R-CNN chạy nhanh hơn gấp 10 lần so với R-CNN trong quá trình huấn luyện. Và nhanh hơn 150 lần trong inferencing.

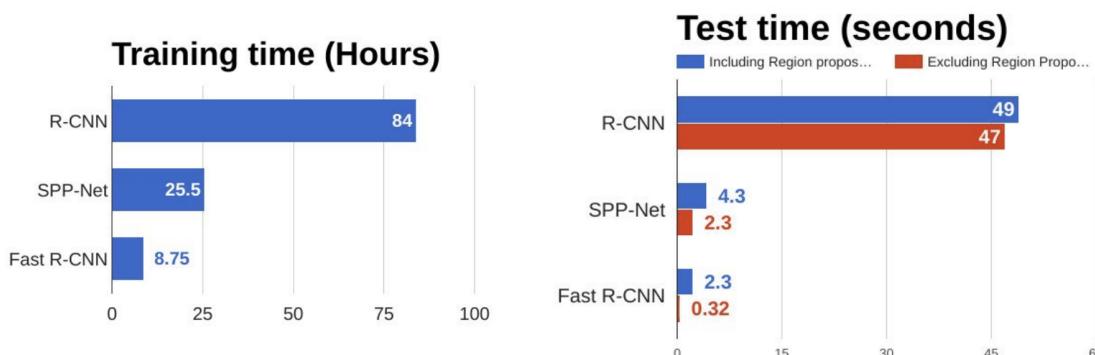


Figure 7: Tốc độ xử lý của R-CNN, SPP-Net và Fast R-CNN (Nguồn: [Rohith Gandhi](#))

Một khác biệt lớn nhất của Fast R-CNN là toàn bộ network (feature extractior, classifier, boundary box regressor) có thể huấn luyện end-to end (nghĩa là từ đầu đến cuối) với 2 hàm độ lỗi (loss function) khác nhau cùng lúc (classification loss và localization loss). Điều này làm tăng độ chính xác của mô hình.

3.1.3 FASTER R-CNN

Khi sử dụng Fast R-CNN thì thời gian để tìm ra region proposal rất lâu và làm chậm thuật toán, vì thế ta cần phải thay thế thuật toán selective search (thuật toán tìm kiếm chọn lọc).

Faster R-CNN không dùng thuật toán selective search để lấy ra các region proposal, mà nó thêm một mạng CNN mới gọi là Region Proposal Network (RPN) để tìm các region proposal.

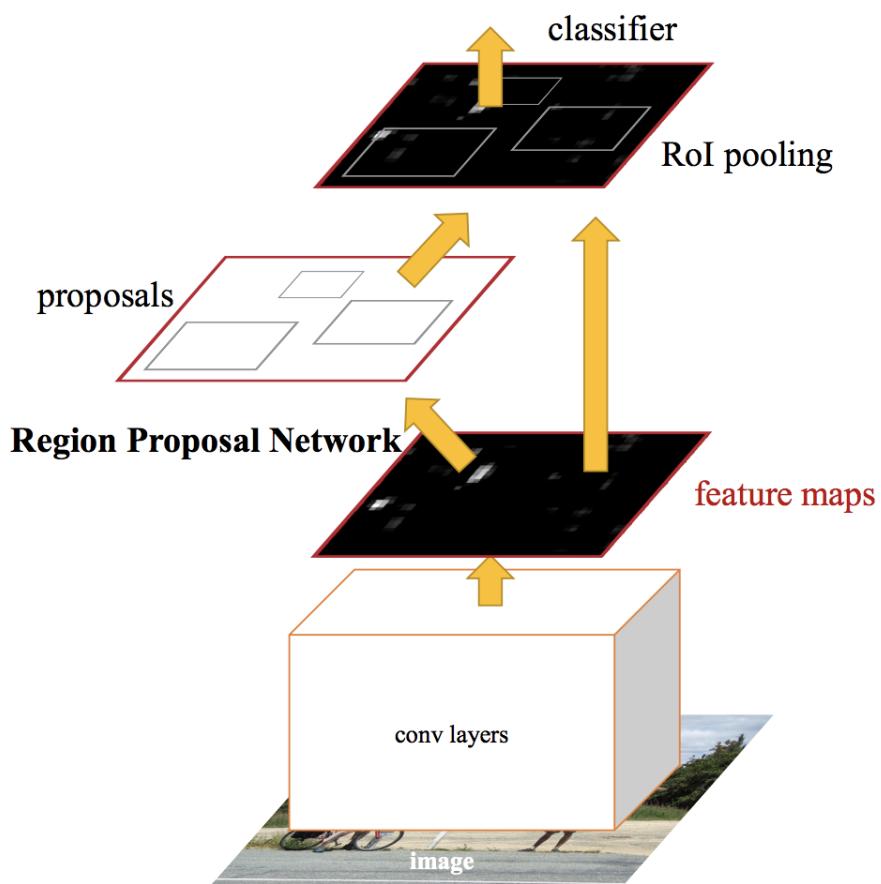


Figure 8: Kiến trúc mạng Faster R-CNN theo kiến trúc mới (Nguồn: [Faster R-CNN](#))

Đầu tiên cả bức ảnh được cho qua pre-trained model để lấy feature map. Sau đó feature map được dùng cho Region Proposal Network để lấy được các region proposal. Sau khi lấy được vị trí các region proposal thì thực hiện tương tự Fast R-CNN.

Sau quá trình phát triển, Faster CNN đã có thời gian test dữ liệu vượt trội so với các thế hệ trước, vì thế có thể dùng cho các công việc nhận dạng, xử lý ảnh trong thời gian thực tế (real time object detection)

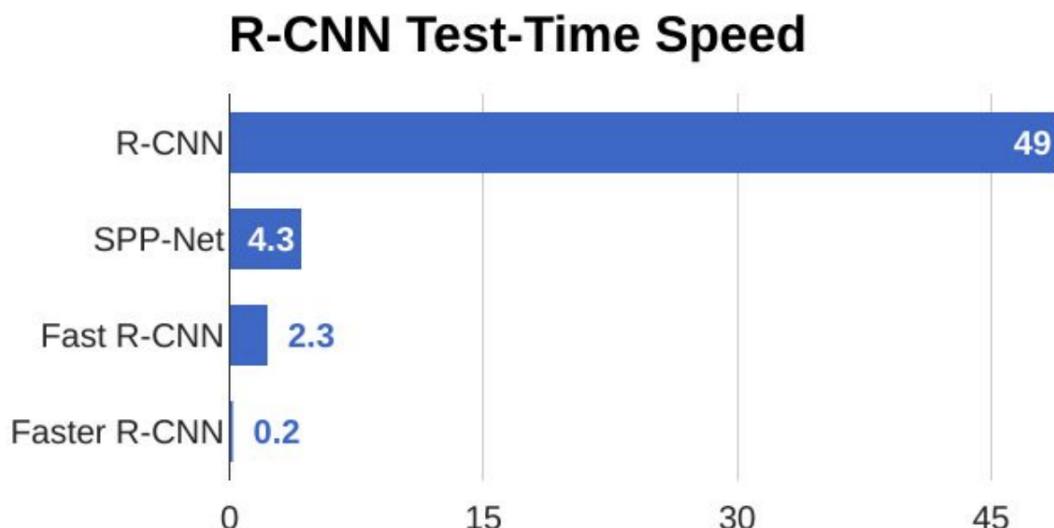


Figure 9: So sánh thời gian test dữ liệu giữa Faster R-CNN với các thế hệ trước (Nguồn: [Rohith Gandhi](#))

3.2 MASK R-CNN:

3.2.1 GIỚI THIỆU TỔNG QUAN

Mask R-CNN là biến thể tiếp theo được phát triển mở rộng từ Faster R-CNN để phân đoạn hình ảnh ở mức độ điểm ảnh (**Image segmentation**). Ý tưởng chính của phương pháp này đó chính là tách rời phân loại và thực hiện công việc dự đoán lớp mặt nạ trên từng pixel. Dựa vào kiến trúc được kế thừa từ Faster R-CNN, Mask R-CNN được thêm vào một nhánh mới phục vụ cho việc dự đoán lớp đối tượng mặt nạ song song với sự tồn tại của nhánh dùng để phân loại và xác định vị trí đối tượng trên ảnh. Nhánh đánh dấu mặt nạ là mạng kết nối đầy đủ quy mô nhỏ (**Small Fully-Connected Network**) áp dụng cho từng **RoI** (**Region of Interest**).

interest) với mục đích dự đoán mặt nạ phân đoạn theo cách pixel-to-pixel

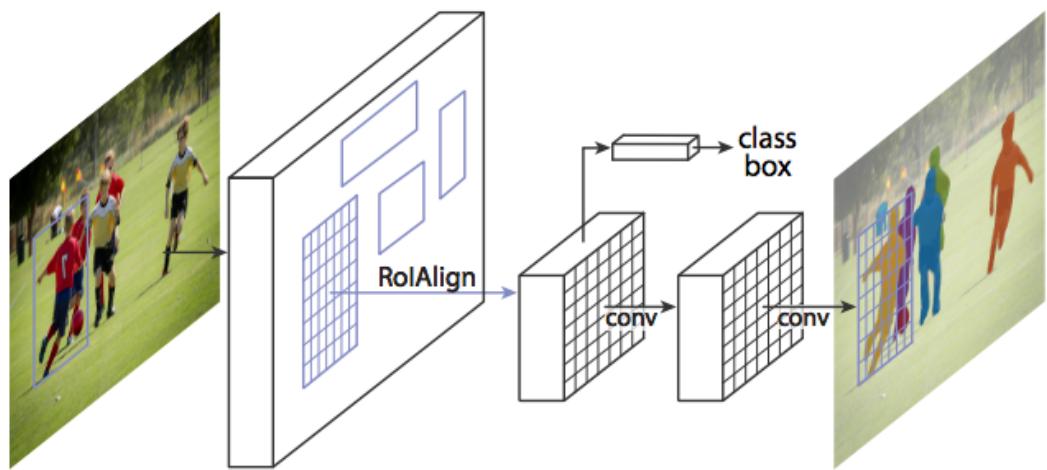


Figure 10: Mask R-CNN là mô hình Faster R-CNN với phân đoạn hình ảnh
(Nguồn:Mask R-CNN)

Bởi vì công đoạn phân đoạn hình ảnh ở mức độ điểm ảnh yêu cầu nhiều liên kết hạt mịn (**fine-grained alignment**) hơn là các khung tìm kiếm giới hạn (bounding box), Mask R-CNN cải thiện Lớp tổng hợp RoI (còn được biết với cái tên khác là **RoI Align**) vì thê RoI có thể tốt hơn và ánh xạ chính xác hơn đến các khu vực của hình ảnh gốc.



Figure 11: Mask R-CNN thực nghiệm trên bộ dữ liệu COCO dựa trên mạng ResNet-101. Lớp mặt nạ được biểu diễn bằng lớp màu sắc, bounding box, hình thức thẻ loại cũng được thể hiện theo (Nguồn: [Mask R-CNN](#))

3.2.2 ROI ALIGN

Lớp RoI Align được thiết kế cho việc sửa chữa cải thiện cho các vị trí bị sắp xếp sai bởi giai đoạn lượng tử hoá của RoI Pooling. Sự khác biệt là giới thiệu phép nội suy song tuyến tính ([Bilinear interpolation](#)) khi tính giá trị pixel pixel cho tọa độ dấu phẩy động. Ví dụ (18.4, 240.8). Đây là tọa độ điểm nổi. Tuy nhiên, chúng tôi biết giá trị pixel nào cho (18, 240) và (19, 241). Vì vậy, để ước tính (18.4, 240.8), chúng ta có thể sử dụng một kỹ thuật được sử dụng trong nhiều thủ thuật xử lý ảnh được gọi là phép nội suy song tuyến tính.

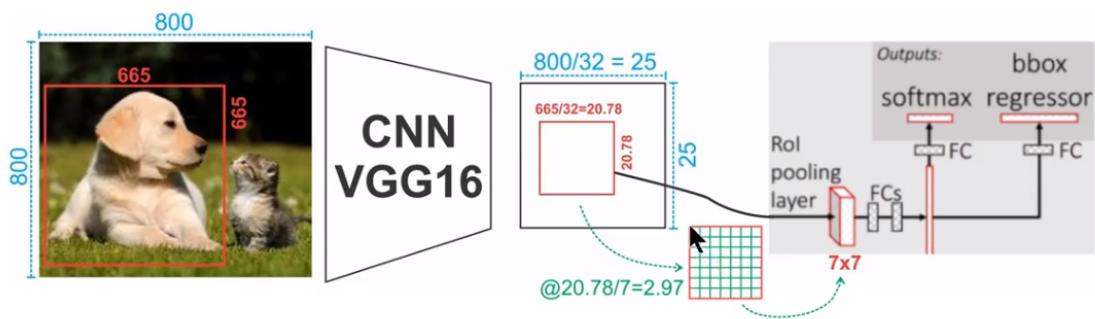


Figure 12: Minh họa kiến trúc ROI Align được phát triển từ ROI Pooling (Nguồn: Xuandong Xu)

ROI Align được báo cáo kiểm chứng là có khả năng vượt trội hơn so với ROI Pooling với cả bộ dữ liệu COCO và VOC 2007. Bộ dữ liệu COCO có ý nghĩa hơn do các hộp giới hạn nhỏ hơn với các đối tượng nhỏ hơn để nhận biết.

3.2.3 CÁC BƯỚC THỰC HIỆN

1. Anchor sorting and filtering: Hình dung từng bước của giai đoạn đầu tiên của Region Proposal Network



Figure 13: detection anchors

2. Bounding Box Refinement: Đây là một ví dụ về các hộp phát hiện cuối cùng (đường chấm chấm) và sự tinh chỉnh được áp dụng cho chúng (đường liền nét) trong giai đoạn thứ hai.

Detections after NMS

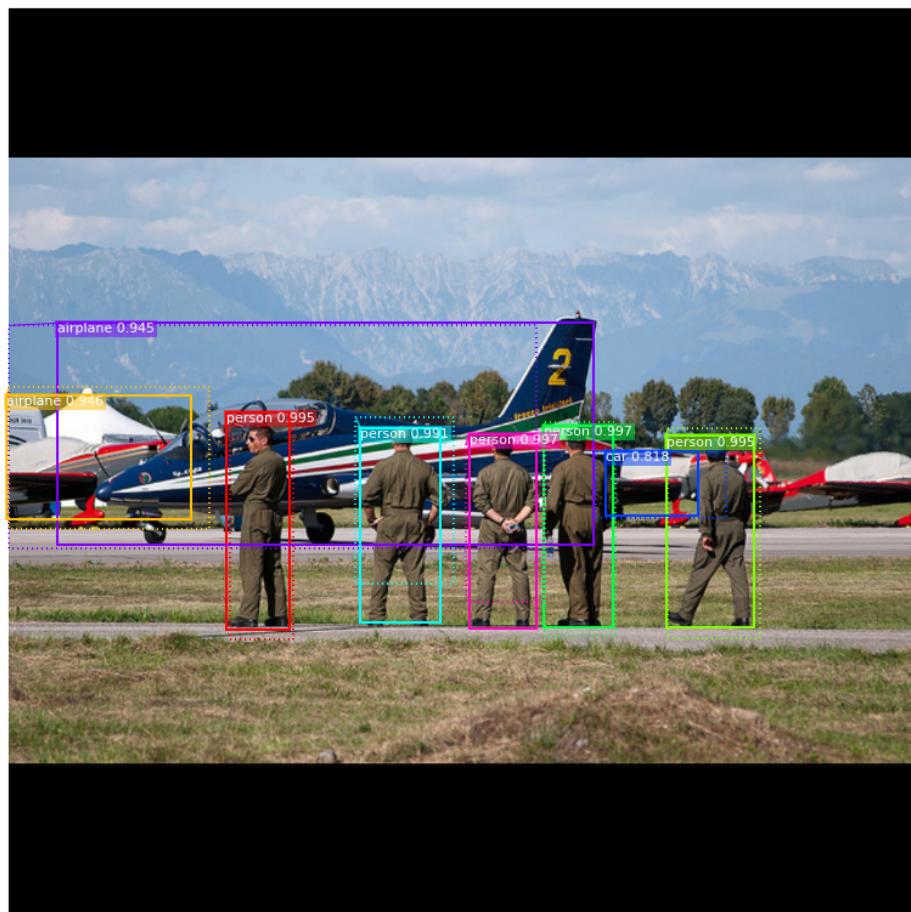


Figure 14: detection refinement

3. Mask Generation: Ví dụ về các mask được tạo ra. Chúng sau đó được thu nhỏ và đặt vào hình ảnh ở đúng vị trí.

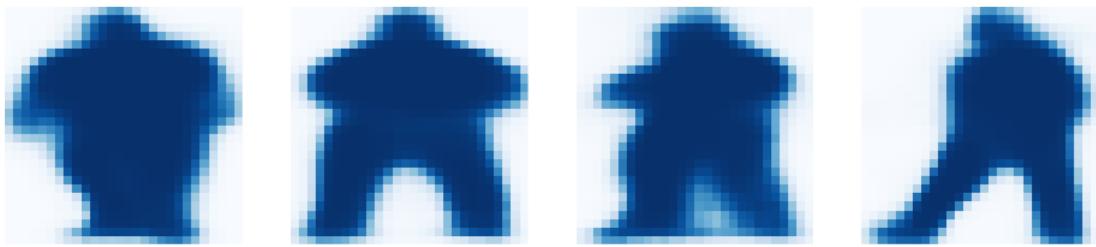


Figure 15: detection mask

4. Layer activations: Thường rất hữu ích để kiểm tra kích hoạt ở các lớp khác nhau để tìm dấu hiệu sự có (tất cả các số không hoặc nhiễu ngẫu nhiên).

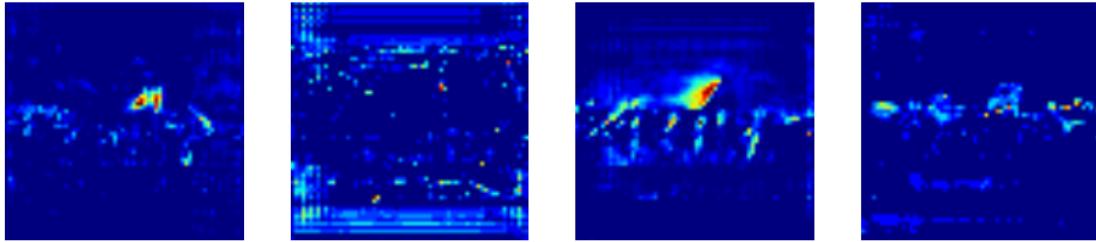


Figure 16: detection activations

5. Weight Histograms: Một công cụ gỡ lỗi hữu ích khác là kiểm tra biểu đồ trọng lượng

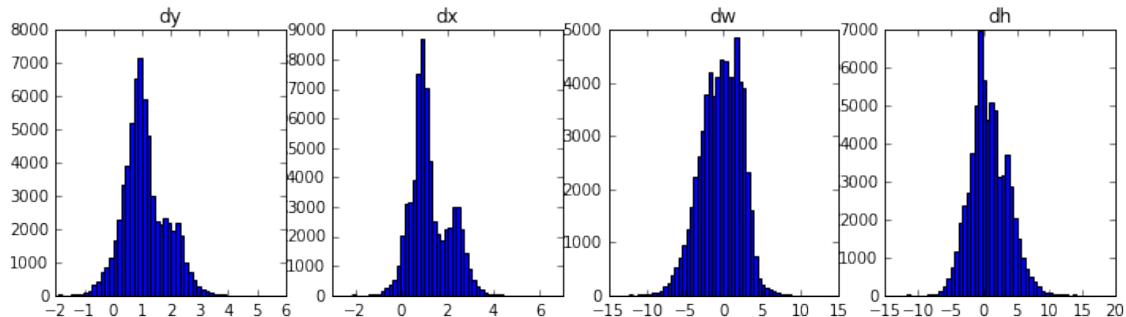


Figure 17: detection histograms

6. Logging to TensorBoard: TensorBoard là một công cụ gỡ lỗi và trực quan tuyệt vời khác. Mô hình được cấu hình để ghi lại các khoản lỗ và lưu trọng số vào

cuối mỗi kỳ nguyên.



Figure 18: detection tensorboard

Kết quả nhận được

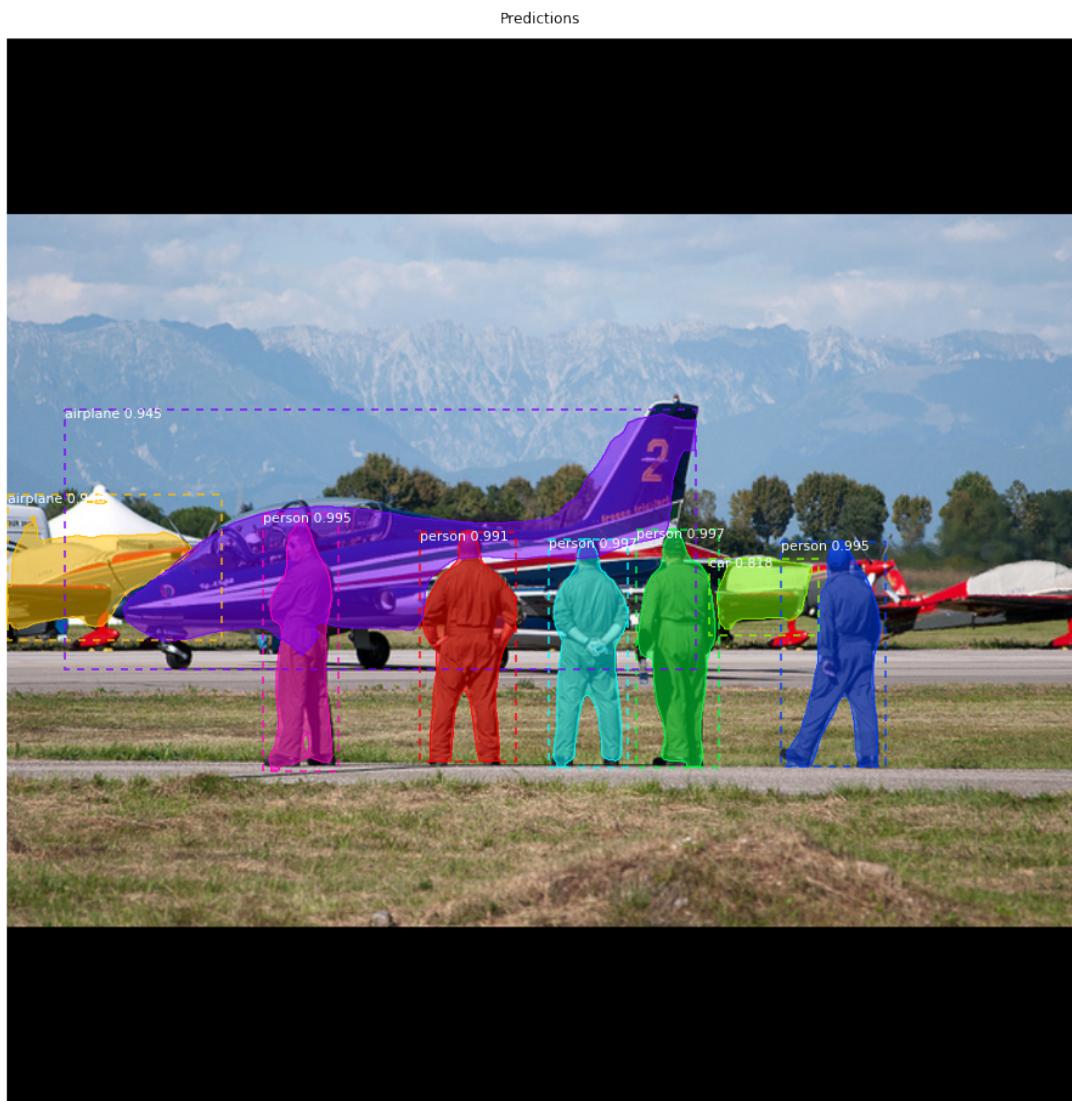


Figure 19: Final result

4

THỰC NGHIỆM

4.1 TẬP DỮ LIỆU

Với các bài toán Detection có rất nhiều bộ dataset hỗ trợ như [ImageNet](#), [Open Images Dataset](#), [KAIST Multispectral Pedestrian Detection](#), ... Nhóm quyết định sử dụng bộ dataset [COCO](#), đây là bộ dataset rất lớn cho object detection, segmentation and captioning dataset. Gồm 330k images với hơn 200k label, 80 object. Sau thời gian phát triển, bộ dataset này đã được tài trợ bởi Common Visual Data Foundation (CVDF), Microsoft (Có phiên bản Microsoft COCO - [MS COCO](#)), Facebook và Mighty AI

Trong bộ dataset COCO được dùng trong object detection, segmentation and captioning có 80 objects bao gồm: person, bicycle, car, motorcycle, airplane, bus, train, truck, boat, traffic light, fire hydrant, stop sign, parking meter, bench, bird, cat, dog, horse, sheep, cow, elephant, bear, zebra, giraffe, backpack, umbrella, handbag, tie, suitcase, frisbee, skis, snowboard, sports ball, kite, baseball bat, baseball glove, skateboard, surfboard, tennis racket, bottle, wine glass, cup, fork, knife, spoon, bowl, banana, apple, sandwich, orange, broccoli, carrot, hot dog, pizza, donut, cake, chair, couch, potted plant, bed, dining table, toilet, tv, laptop, mouse, remote, keyboard, cell phone, microwave, oven, toaster, sink, refrigerator, book, clock, vase, scissors, teddy bear, hair drier, toothbrush



(a) Airplane



(b) Bus



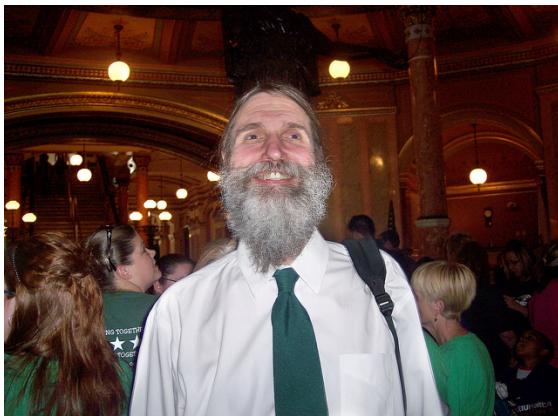
(c) Person



(d) Car



(e) Train



(f) Person



22

(g) Pizza



(h) Bird

Figure 20: Một số hình ảnh minh họa từ dataset COCO

Trong phần này sẽ sử dụng bộ dataset COCO bản cập nhật mới nhất được công bố tại [Trang chủ COCO dataset](#), trong đó bộ ảnh dùng để train chứa 118,287 mẫu và bộ dùng để test chứa 40,670 mẫu với các kích thước, tỉ lệ ảnh, điều kiện ánh sáng, bối cảnh ảnh, các chi tiết trong ảnh khác nhau.

4.2 THIẾT LẬP THỰC NGHIỆM

4.2.1 CẤU HÌNH THIẾT BỊ THỰC NGHIỆM

- Cloud Computing System: Google Colab (Colaboratory notebook)
- GPU: NVIDIA TESLA K80 24GB GDDR5 PCIE 3.0
- RAM: 12,72 GB
- Disk: 358,27 GB

4.2.2 CÁC CÔNG CỤ CẦN THIẾT

- Ngôn ngữ lập trình: Python 3.6
- os, sys, random, math, collection, numpy, skimage, matplotlib, COCO (for Python), mrcnn, cv2, scipy, YouTube video processing

4.2.3 QUÁ TRÌNH THỰC NGHIỆM

```
1 import os
2 import sys
3 import cv2
4 import time
5 import PIL
6 import random
7 import math
8 import copy
9 import argparse
10 import datetime
11 import imutils
12 import scipy.misc
13 import numpy as np
14 import skimage.io
15 import collections
16 import matplotlib
17 import matplotlib.pyplot as plt
18 from __future__ import print_function
```

```

19 from PIL import Image, ImageEnhance
20 import matplotlib.patches as patches
21
22 import coco
23 from mrcnn import utils
24 import mrcnn.model as modellib
25 from mrcnn import visualize

```

Listing 1: Các thư viện cần thiết trước khi bắt đầu

Ứng dụng của đề tài này hiện được cài đặt nhằm để phát hiện vật thể trong ban đêm. Để thuận tiện cũng như bắt kịp xu thế dữ liệu thật (real time data) nhóm sẽ phân tích trực tiếp và nhận dạng từ các video trên nền tảng chia sẻ video trực tuyến [YouTube](#)

```

1 from __future__ import unicode_literals
2 !pip install --upgrade youtube-dl # install if you don't have it
3 import youtube_dl
4
5 def YouTube_download(url):
6     ydl_opts = {
7         'outtmpl': 'yt-video.%s' % (ext)
8     }
9     with youtube_dl.YoutubeDL(ydl_opts) as ydl:
10        ydl.download([url])

```

Listing 2: Thiết lập thư viện để tải và xử lý video Youtube thông qua các đường link

Trên thực tế, bộ dữ liệu COCO có số lượng ảnh rất lớn nên ta duyệt từng hình ảnh để train thì sẽ tốn rất nhiều thời gian và tài nguyên, vì thế ta sẽ sử dụng các mô hình đã được huấn luyện sẵn với bộ dữ liệu COCO. Tuy được công bố trên [trang chủ COCO](#) là có 80 đối tượng khác nhau nhưng khi truy vấn thì các hình ảnh chưa đưa chia theo đối tượng cụ thể, ta sẽ tạo bộ tên các đối tượng để dễ phân loại

```

1 # Root directory of the project
2 ROOT_DIR = os.getcwd()
3
4 # Directory to save logs and trained model
5 MODEL_DIR = os.path.join(ROOT_DIR, "logs")

```

```

6
7 # Local path to trained weights file
8 COCO_MODEL_PATH = os.path.join(ROOT_DIR, "mask_rcnn_coco.h5")
9 # Download COCO trained weights from Releases if needed
10 if not os.path.exists(COCO_MODEL_PATH):
11     utils.download_trained_weights(COCO_MODEL_PATH)
12
13 ## Configurations
14 ## We'll be using a model trained on the MS-COCO dataset. The
15     configurations of this model are in the ```CocoConfig``` class in
16     ```coco.py```.
17 ## For inferencing, modify the configurations a bit to fit the task. To
18     do so, sub-class the ```CocoConfig``` class and override the
19     attributes you need to change.
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43

```

```

44 #TRAIN_ROIS_PER_IMAGE = 50
45 #BACKBONE = "resnet50" #not working at all!
46 #RPN_ANCHOR_STRIDE = 2
47 POST_NMS_ROIS_TRAINING = 1000
48 POST_NMS_ROIS_INFERENCE = 500
49 IMAGE_MIN_DIM = 400 #really much faster but bad results
50 IMAGE_MAX_DIM = 512
51 #DETECTION_MAX_INSTANCES = 50 #a little faster but some instances
52     not recognized
53
54 config = InferenceConfig()
55 config.display()
56
57 ## Create Model and Load Trained Weights
58
59 # Create model object in inference mode.
60 model = modellib.MaskRCNN(mode="inference", model_dir=MODEL_DIR, config
61     =config)
62
63 # Load weights trained on MS-COCO
64 model.load_weights(COCO_MODEL_PATH, by_name=True)
65
66 # COCO Class names
67 # Index of the class in the list is its ID. For example, to get ID of
68 # the teddy bear class, use: class_names.index('teddy bear')
69 class_names = ['BG', 'person', 'bicycle', 'car', 'motorcycle', 'airplane',
70     'bus', 'train', 'truck', 'boat', 'traffic light', 'fire hydrant',
71     'stop sign', 'parking meter', 'bench', 'bird', 'cat', 'dog', 'horse',
72     'sheep', 'cow', 'elephant', 'bear', 'zebra', 'giraffe', 'backpack',
73     'umbrella', 'handbag', 'tie', 'suitcase', 'frisbee', 'skis',
74     'snowboard', 'sports ball', 'kite', 'baseball bat', 'baseball glove',
75     'skateboard', 'surfboard', 'tennis racket', 'bottle', 'wine glass',
76     'cup', 'fork', 'knife', 'spoon', 'bowl', 'banana', 'apple',
77     'sandwich', 'orange', 'broccoli', 'carrot', 'hot dog', 'pizza',
78     'donut', 'cake', 'chair', 'couch', 'potted plant', 'bed',
79     'dining table', 'toilet', 'tv', 'laptop', 'mouse', 'remote',
80     'keyboard', 'cell phone', 'microwave', 'oven', 'toaster', 'sink',
81     'refrigerator', 'book', 'clock', 'vase', 'scissors', 'teddy bear',
82     'hair drier', 'toothbrush']

```

Listing 3: Thiết lập Mask R-CNN và dataset COCO

Sau khi tích hợp kiến trúc mạng ResNet-101, các mô hình đã được huấn luyện

sẵn với bộ dữ liệu COCO đồng nghĩa với việc ta đã có được lượng dữ liệu huấn luyện (training set)

Bắt đầu với 1 video bất kỳ, nhóm sẽ thực hiện với video **Nighttime at popular Wangfujing Street in Beijing, China**, sau khi nhận được video sẽ tiến hành tách video thành từng frame riêng biệt với tốc độ 30fps.

```
1 IMAGE_DIR = "output-dir" # dir to save images
2
3 # Download YT video
4 YouTube_download("https://www.youtube.com/watch?v=U1WY_UfqXwo")
5
6 # Run detection and output frames
7 video_to_frames(input_vid = "yt-video.mp4", output_loc = IMAGE_DIR,
max_fps=30*60)
```

Listing 4: Tách thành từng frame riêng từ video cho trước

Thu hẹp phạm vi của bài toán trở nên nhỏ hơn với input là những frame ảnh được trích xuất từ video và output là frame ảnh chứa các đối tượng được nhận dạng

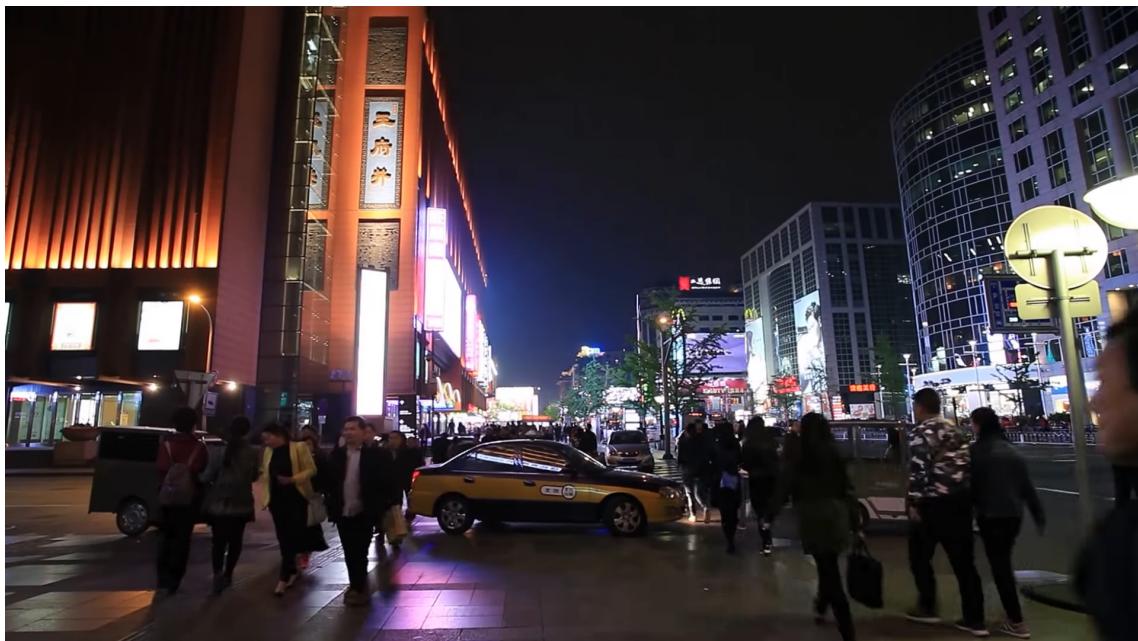


Figure 21: Frame đã được tách ra từ video gốc

Đối với mỗi tấm hình tương ứng với mỗi frame trong video gốc, ta sẽ tiến hành nhận dạng sau đó gán các giá trị thông số cho các đối tượng xuất trong ảnh như lớp mặt nạ (mỗi đối tượng có màu sắc riêng biệt), lớp đối tượng, tỷ lệ chính xác

```
1 ## Functions to visualize detection results on the image
2
3 def random_colors(N):
4     np.random.seed(2500)
5     colors = [tuple(255 * np.random.rand(3)) for _ in range(N)]
6     return colors
7
8 def apply_mask(image, mask, color, alpha=0.5):
9     """apply mask to image"""
10    for n, c in enumerate(color):
11        image[:, :, n] = np.where(
12            mask == 1,
13            image[:, :, n] * (1 - alpha) + alpha * c,
14            image[:, :, n]
15        )
16    return image
17
18 def display_instances(image, boxes, masks, ids, names, scores, same =
19                      True):
20    """
21        take the image and results and apply the mask, box, and Label
22    """
23    n_instances = boxes.shape[0]
24    if not n_instances:
25        print('NO INSTANCES TO DISPLAY')
26    else:
27        assert boxes.shape[0] == masks.shape[-1] == ids.shape[0]
28
29    if same is True:
30        colors = random_colors(len(class_names))
31        for i in range(n_instances):
32            if not np.any(boxes[i]):
33                continue
34            y1, x1, y2, x2 = boxes[i]
35            label = names[ids[i]]
36            color = colors[ids[i]]
37            score = scores[i] if scores is not None else None
```

```

37     caption = '{} {:.1%}'.format(label, score) if score else label
38     mask = masks[:, :, i]
39     image = apply_mask(image, mask, color)
40     image = cv2.rectangle(image, (x1, y1), (x2, y2), color, 1)
41     image = cv2.putText(
42         image, caption, (x1, y1), cv2.FONT_HERSHEY_COMPLEX, 0.5,
43         color, 1
44     )
45 else:
46     colors = random_colors(n_instances)
47     for i, color in enumerate(colors):
48         if not np.any(boxes[i]):
49             continue
50         y1, x1, y2, x2 = boxes[i]
51         label = names[ids[i]]
52         score = scores[i] if scores is not None else None
53         caption = '{} {:.1%}'.format(label, score) if score else label
54         mask = masks[:, :, i]
55         image = apply_mask(image, mask, color)
56         image = cv2.rectangle(image, (x1, y1), (x2, y2), color, 1)
57         image = cv2.putText(
58             image, caption, (x1, y1), cv2.FONT_HERSHEY_COMPLEX, 0.5,
59             color, 1
60         )
61
62 # Add caption
63 counter = []
64 for _ in ids:
65     counter.append(names[_])
66 caption = str(collections.Counter(counter).most_common(3))
67 image = cv2.rectangle(image, (0, 0), (len(caption)*8, 40), (0,0,0),
-1)
68 image = cv2.putText(image,caption,(10,25), cv2.FONT_HERSHEY_SIMPLEX
, 0.5, (255,255,255), 1, cv2.LINE_AA)
69 return image

```

Listing 5: Tiến hành xử lý riêng từng bức ảnh



Figure 22: Frame đã được nhận dạng với điều kiện ánh sáng không đầy đủ

Cứ thực hiện như thế cho đến khi xử lý hết tất cả các frame trong video nhưng nhóm đã thiết lập tổng số frame tối đa của video là $30(\text{fps}) * 60(\text{s}) = 1800$ frames, sau đó tiến hành ghép từng frame lại thành video mới với kích thước như video ban đầu với thời gian có hạn (phục vụ cho mục đích kiểm nghiệm sản phẩm trong thời gian ngắn)

```

1 ## Function to input a video and output multiple frames with their
   detection
2
3 def video_to_frames(input_vid, output_loc, max_fps = None):
4     # check if folder already exist otherwise mkdir
5     if not os.path.exists(output_loc):
6         os.mkdir(output_loc)
7     print("%s was created" % output_loc)
8     # log the time
9     time_start = time.time()
10    # capture frame
11    cap = cv2.VideoCapture(input_vid)
12    count = 0
13    print('\nRunning Mask R-CNN on %s' % input_vid)
14
15    try:

```

```

16     while True:
17         status, image = cap.read()
18
19         # run detection
20         results = model.detect([image], verbose = 0)
21         # visualization
22         r = results[0]
23         result_image = display_instances(
24             image, r['rois'], r['masks'], r['class_ids'], class_names, r[
25             'scores']
26         )
27         cv2.imwrite(output_loc + "/frame%04d.jpg" % count, result_image)
28         count += 1
29         # print every 50 frames
30         if count % 50 == 0:
31             time_mid = time.time()
32             print("%d frames converted. Time elapsed: %d seconds." % (count
33             , (time_mid - time_start)))
34             # set upper limit
35             if not max_fps == None:
36                 if count > max_fps:
37                     break
38
39     except Exception as e:
40         print("There was an error!")
41         print(e)
42
43     cap.release()
44     # log the time again
45     time_end = time.time()
46     print("%d frames converted at %d frames per second\n" % (count, (
47         count/(time_end - time_start))))
48     print("Conversion time: %d seconds." % (time_end - time_start))
49
50     def single_frame_detection(path, title="", figsize=(16, 16), ax=None):
51         image = scipy.misc.imread(path)
52
53         if not ax:
54             _, ax = plt.subplots(1, figsize=figsize)
55
56         # Show area outside image boundaries.
57         height, width = image.shape[:2]

```

```

55 ax.set_ylim(height + 10, -10)
56 ax.set_xlim(-10, width + 10)
57 ax.axis('off')
58 ax.set_title(title)

59
60 # Run detection
61 results = model.detect([image], verbose=0)
62 # Visualize results
63 r = results[0]
64 result_image = display_instances(image, r['rois'], r['masks'], r['
    class_ids'], class_names, r['scores'])
65 plt.imshow(result_image)
66 plt.show()

```

Listing 6: Tiến hành ghép các frame đã được nhận diện thành video hoàn chỉnh như video gốc

Chi tiết hơn về code mà nhóm đã làm có thể xem tại [Github](#)

4.3 KẾT QUẢ THỰC NGHIỆM

4.3.1 KẾT QUẢ TRÊN TẬP DỮ LIỆU HUẤN LUYỆN

Trước tiên, mô hình máy học được thử trên tập dữ liệu đầu vào còn hợp nhiễu. Dữ liệu được thử nghiệm là dữ liệu ảnh, một số vật thể bạn có thể gặp trên đường như car, bike, person,... Mô hình tiến hành nhận diện các vật thể trong đêm.

Kết quả chạy thử:

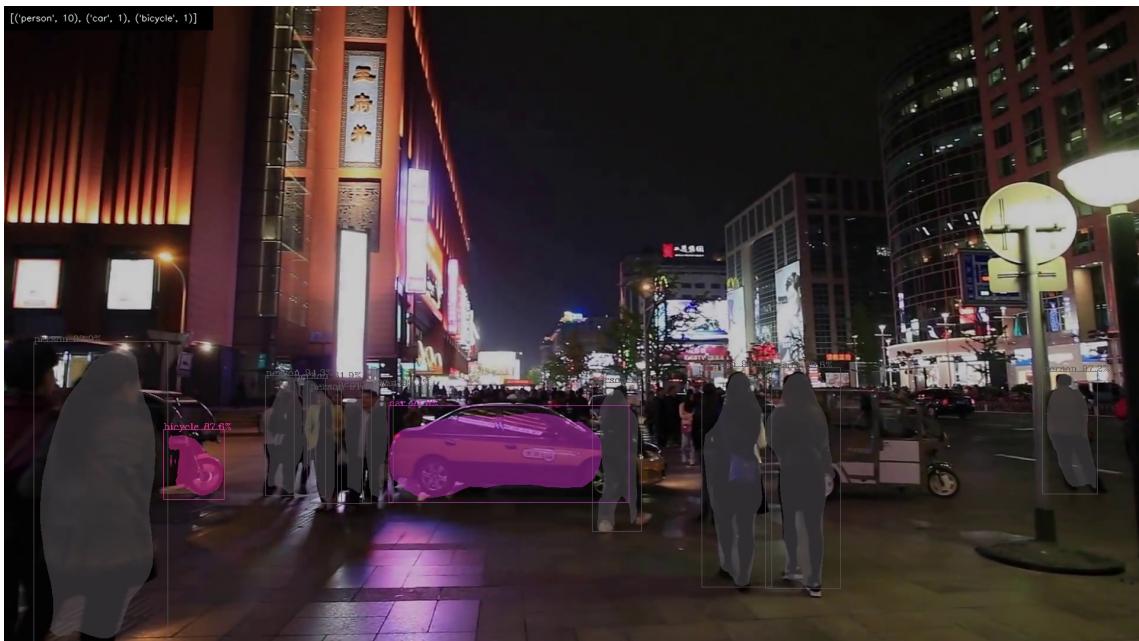


Figure 23: Frame đã được nhận dạng với điều kiện ánh sáng không đầy đủ

Trên dữ liệu còn hơi nhiều, chương trình dự đoán khá chính xác các vật thể vehicle như car, bicycle hoặc person. Đối với các chi tiết có kích thước nhỏ như người hoặc vật thể ở xa sẽ không thể nhận diện cho bị khuất hoặc không thể định dạng mask.

4.3.2 KẾT QUẢ TEST

Với dữ liệu CoCo, chương trình thực hiện đầy đủ các bước tiền xử lý bao gồm tách nền, ghép nhóm, giảm mẫu, lọc các điểm không liên quan.

Về cơ bản, trên dữ liệu từ CoCo với nhiều nhiễu lượng tử thì chương trình vẫn có thể nhận diện phần lớn các vật thể: car, bycycle, person thể hiện bởi các màu hồng và xám. Các vật thể bị nhận diện sai chủ yếu nằm ở những phần nó nhiều nhiễu lượng tử như màu vật thể trùng khớp với màu xung quanh, khoảng cách xa dẫn đến kích thước nhỏ.

Đối với các vật thể có diện tích nhỏ như bicycle, chương trình nhận diện chúng còn khó khăn do có thể bị khuất hoặc trùng lặp màu với môi trường vì điều kiện ánh sáng.

4.4 BẢNG THỐNG KÊ

Tỷ lệ chính xác (nhận diện đúng/ Tổng số frame có chứa vật thể

	Car	Bus	Bike	Bicycle	Person	Traffic light
Test 1:	9/466	0	0	0	3/16	18/31
Test 2:	1656/1794	4/197	27/53	9/14	1794/1974	0
Test 3:	309/445	213/403	0	0	0	64/85

Thống kê độ chính xác qua từng test

Test 1

Report classification:

	precision	recall	f_1 score
Car	0.019	0.023	0.021
Bus	0	0	0
Bike	0	0	0
Bicycle	0	0	0
Person	0.187	0.221	0.203
Traffic light	0.580	0.416	0.484

Một số frame đã qua nhận diện trong test 1



(a) frame0102



(b) frame0026



(c) frame0240



(d) frame0430

Test 2

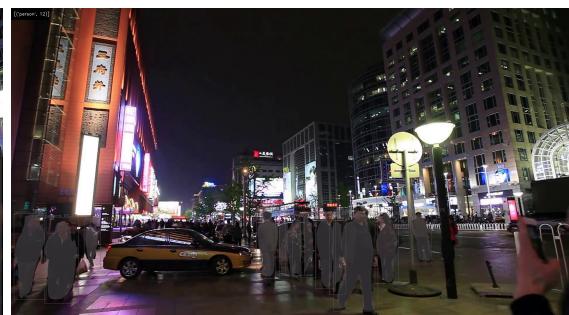
Report classification:

	precision	recall	f_1 score
Car	0.851	0.923	0.886
Bus	0.008	0.02	0.011
Bike	0.476	0.509	0.492
Bicycle	0.828	0.643	0.724
Person	1.000	1.000	1.000
Traffic light	0	0	0

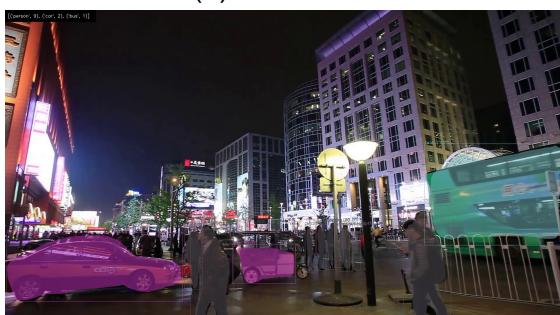
Một số frame đã qua nhận diện trong test 2



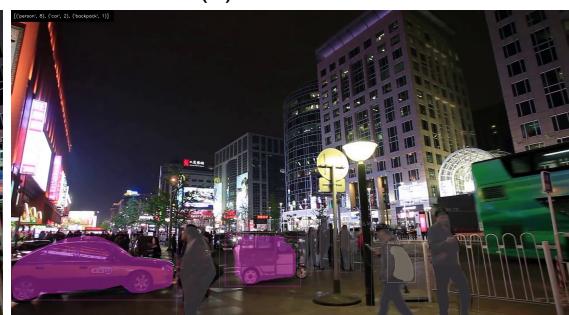
(a) frame0149



(b) frame0476



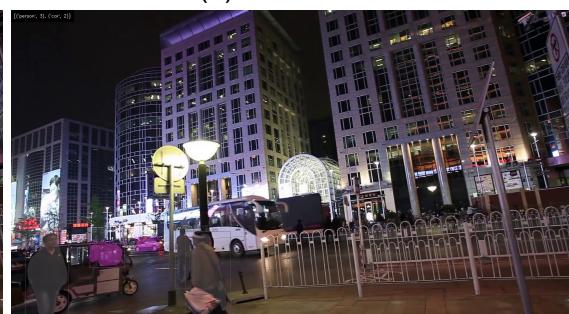
(c) frame0724



(d) frame0735



(e) frame1089



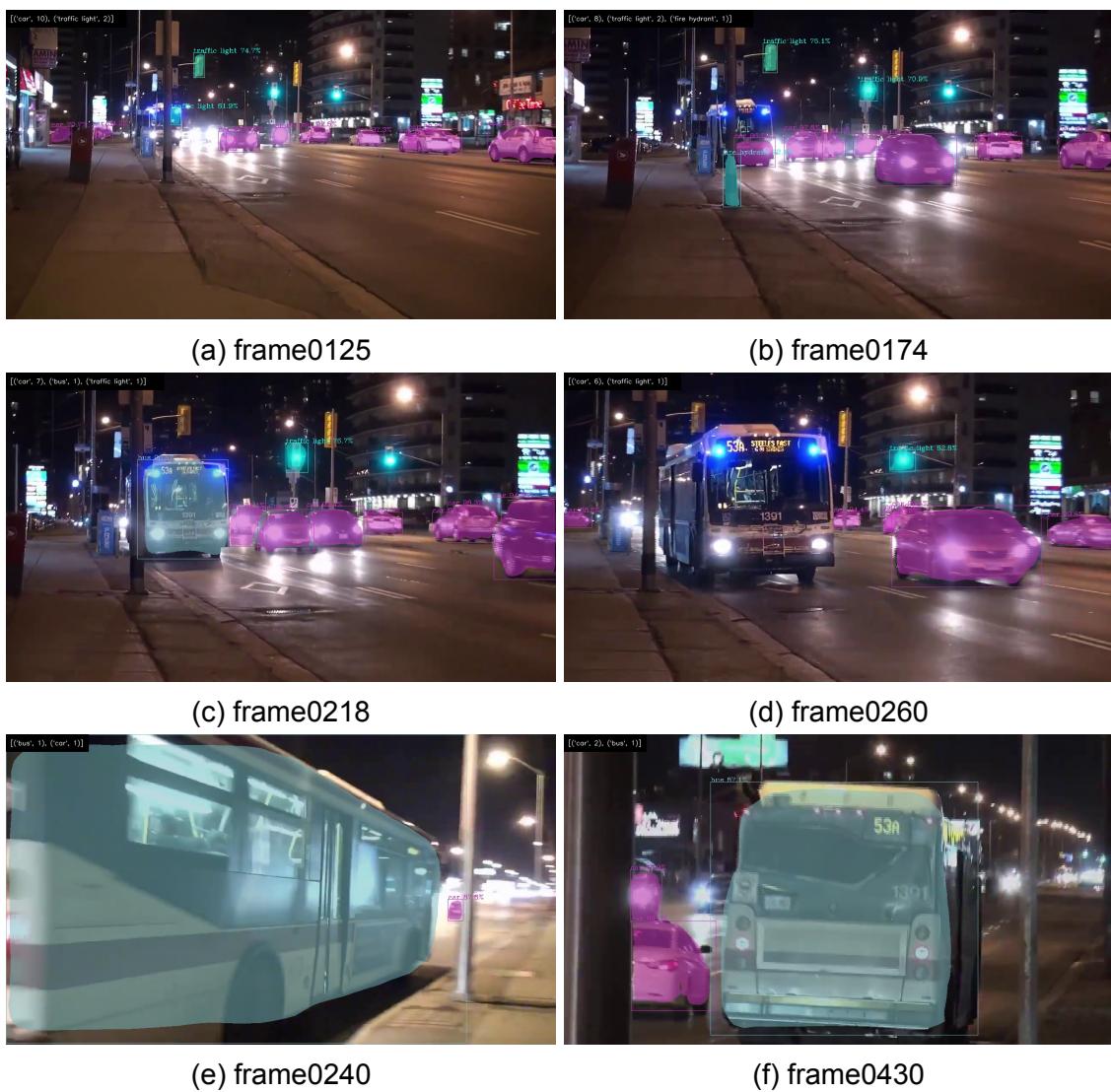
(f) frame1087

Test 3

Report classification:

	precision	recall	f_1 score
Car	0.694	0.743	0.718
Bus	0.528	0.642	0.579
Bike	0	0	0
Bicycle	0	0	0
Person	0	0	0
Traffic light	0,752	0.82	0.784

Một số frame đã qua nhận diện trong test 3



4.5 THẢO LUẬN

Sau quá trình áp dụng Mask R-CNN và dataset COCO, nhìn một cách tổng quan thì có thể thấy các đối tượng trong từng bức hình được phân vùng rõ ràng và đánh dấu với các màu sắc khác nhau để dễ phân biệt đối tượng, trên thực tế đã có người từng sử dụng Faster R-CNN và dataset COCO để giải quyết bài toán này nhưng nhóm quyết định không sử dụng vì Faster R-CNN tuy nhanh nhưng rất dễ nhiễu dẫn đến khi sẽ nhận diện sai đối tượng hoặc trả về rất nhiều hộp tìm kiếm giới hạn chứa các thông tin giống nhau trên cùng một đối tượng duy nhất. Trong điều kiện với thời tiết ban đêm, nhóm quyết định ưu tiên độ chính xác và tối ưu việc giảm thiểu độ nhiễu hơn là tốc độ nhận diện.

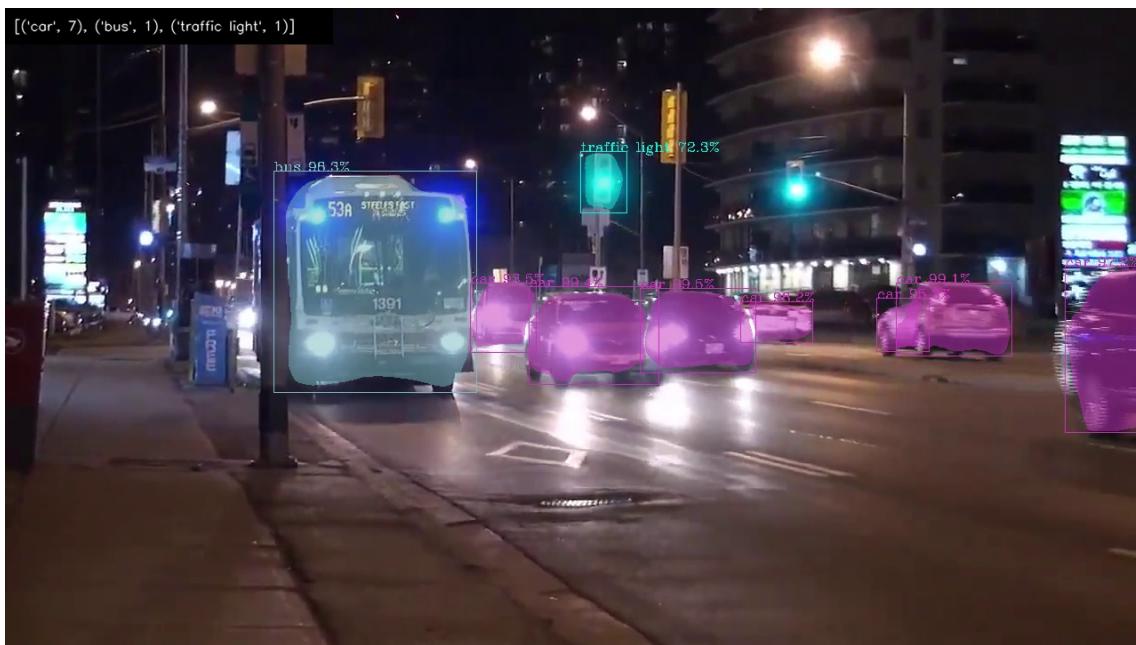


Figure 27: Minh họa cho việc ưu tiên về độ chính xác và giảm thiểu độ nhiễu

5

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 KẾT LUẬN

Xét về mức độ thực tế của đồ án này, thì đây vẫn đang là một trong những bài toán khó, với góc nhìn của con người thì đôi khi chúng ta vẫn còn phân biệt nhầm một vài đối tượng được bao phủ một lớp ánh sáng tối mà cũng chính những đối tượng đó mà chúng ta có thể nhìn và phân biệt rõ ràng khi có có mức anh sáng vừa đủ như bóng đèn cho đến mức độ cao hơn như ánh nắng mặt trời vào ban ngày. Hiện bài toán "Nhận diện các đối tượng, vật thể trong ban đêm" - Object Detection at Nighttime vẫn đang được khai phá và giải quyết với những phương pháp khác nhau nhằm tìm ra được phương pháp tối ưu nhất về mặt thời gian và độ chính xác.

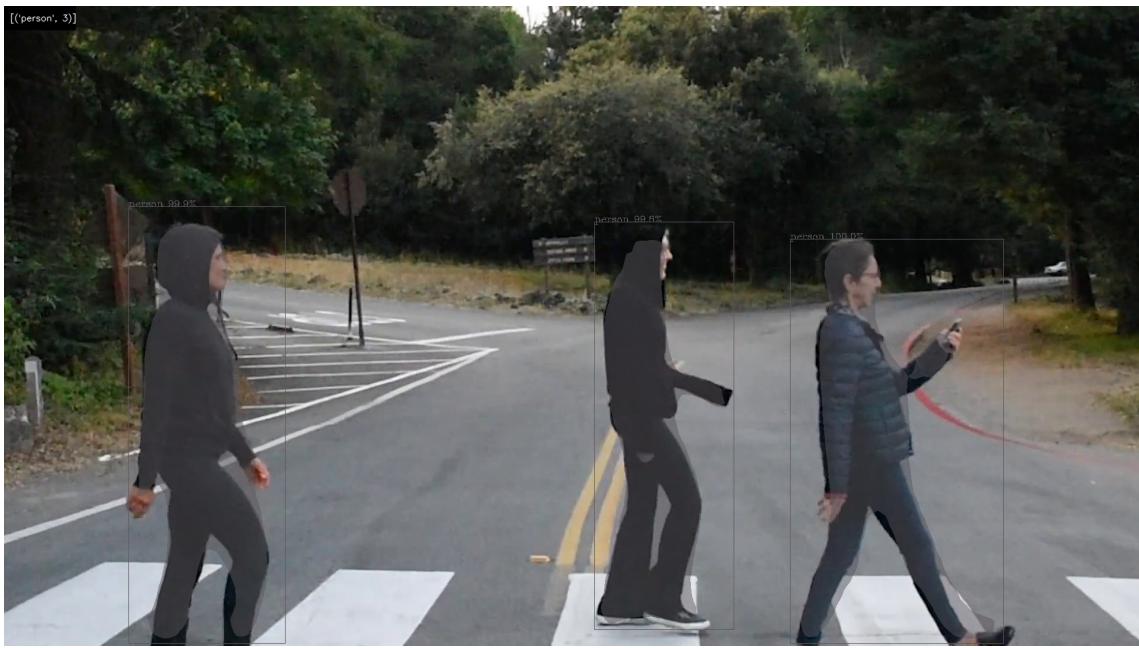


Figure 28: Chúng ta đã tích luỹ đủ kinh nghiệm để có thể nhìn hết toàn bộ chi tiết con người cả ngày lẫn đêm

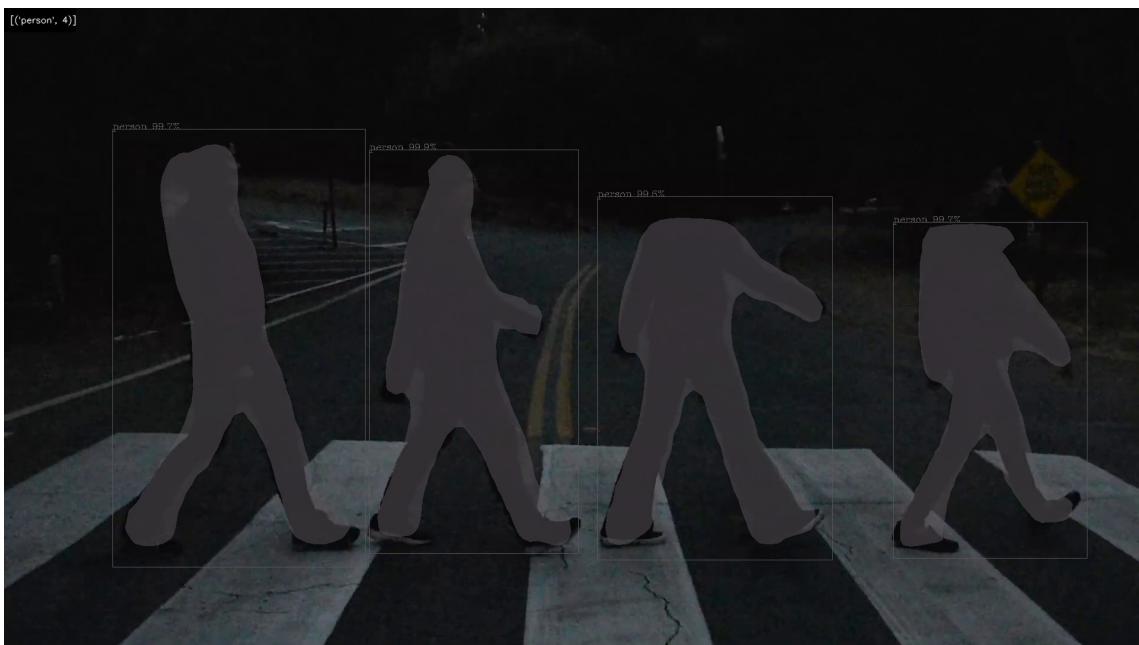


Figure 29: Dạy cho máy tính tìm ra và phân loại được những chi tiết bị bóng tối bao phủ là cả một vấn đề

Trong quá trình tìm hiểu và phát triển phương pháp giải quyết vấn đề, nhóm đa phần tập trung vào xử lý với bộ dữ liệu COCO mà chưa thử nghiệm với các

bộ dữ liệu được miêu tả có thông tin về điều kiện ban đêm như Mapillary Vis-tas Dataset (ICCV2017), KAIST Multispectral Pedestrian Dataset (CVPR2015), Alderley Day/Night Dataset (ICRA2012), KITTI Dataset (BMVC2017), ... dẫn đến kết quả chung bị ảnh hưởng do một vài đối tượng bị nhận diện sai hoặc có khi là không nhận diện được đối tượng đó. Đây có lẽ là một chính lược chưa tốt và cần phải được cải thiện

5.2 HƯỚNG PHÁT TRIỂN

Nhóm sẽ cố gắng cải thiện mô hình và tiếp tục thử nghiệm trên những bộ dữ liệu có thông tin hỗ trợ trong ban đêm như Mapillary Vis-tas Dataset (ICCV2017), KAIST Multispectral Pedestrian Dataset (CVPR2015), Alderley Day/Night Dataset (ICRA2012), KITTI Dataset (BMVC2017), ... để cho ra được những kết quả cuối cùng tối ưu nhất để có thể áp dụng vào thực tiễn.

Nhận thấy được tiềm năng khi ứng dụng vào thực tế của mô hình này, nhóm sẽ tiếp tục phát triển để có thể làm thành bài báo nguyên cứu khoa học, khoá luận tốt nghiệp.

Phát triển thành dự án thực tế với dữ liệu dựa trên thời gian thật, có thể thực hiện theo mô hình đa nền tảng, áp dụng IoT cho các ứng dụng như cảnh báo người đi đường, dự đoán và thông báo tai nạn trước khi xảy ra, giám sát và phát hiện những hành vi bất thường trong đêm thông qua các camera giám sát, ...

REFERENCES

- [1] Tác giả: PULKIT SHARMA, *Computer Vision Tutorial: Implementing Mask R-CNN for Image Segmentation (with Python Code)*. Nơi đăng [link](#)
- [2] Tác giả: matterport, *Mask R-CNN for Object Detection and Segmentation*. Nơi đăng [link](#)