

ĐẠI HỌC BÁCH KHOA HÀ NỘI  
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG



BÁO CÁO PROJECT 1  
XÂY DỰNG MÔ HÌNH AI PHÁT HIỆN VẬT THỂ  
NGUY HIỂM

Sinh viên thực hiện: Nguyễn Huy Hoàng

Mã số sinh viên: 20235336

Mã lớp học: 755566

Giáo viên hướng dẫn: Hoàng Việt Dũng

Hà Nội, 2025

## Mục lục

<b>1</b>	<b>Tổng quan dự án</b>	<b>2</b>
1.1	Mục tiêu . . . . .	2
1.2	Công nghệ sử dụng . . . . .	2
<b>2</b>	<b>Model YOLOv11</b>	<b>2</b>
2.1	Giới thiệu . . . . .	2
2.2	Kiến trúc . . . . .	2
2.2.1	Backbone . . . . .	3
2.2.2	Neck . . . . .	4
2.2.3	Head . . . . .	4
2.3	Các phiên bản . . . . .	4
2.4	Fine-tuning . . . . .	4
<b>3</b>	<b>Xây dựng Web Application</b>	<b>6</b>
3.1	Kiến trúc hệ thống . . . . .	6
3.2	Backend . . . . .	6
3.3	Frontend . . . . .	6
3.4	Tính năng Auto Capture . . . . .	8
<b>4</b>	<b>Cấu trúc dự án</b>	<b>8</b>
<b>5</b>	<b>Hướng dẫn</b>	<b>9</b>
5.1	Cài đặt . . . . .	9
5.2	Chạy Web Application . . . . .	9
5.3	Training Model . . . . .	9
<b>6</b>	<b>Kết quả</b>	<b>9</b>
6.1	Model Performance . . . . .	9
6.2	Web Application . . . . .	10
<b>7</b>	<b>Kết luận</b>	<b>10</b>

# 1 Tổng quan dự án

## 1.1 Mục tiêu

Xây dựng hệ thống camera an ninh phát hiện và cảnh báo vật thể nguy hiểm (dao, kéo) trong thời gian thực, với giao diện web cho phép theo dõi và quản lý các cảnh báo.

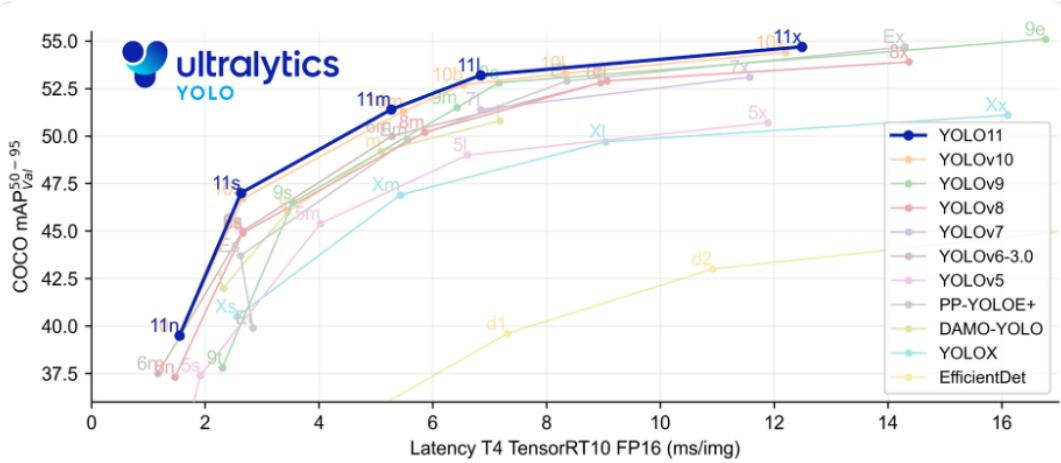
## 1.2 Công nghệ sử dụng

- **YOLOv11**: Model nhận diện vật thể real-time
- **OpenCV**: Xử lý ảnh và video
- **FastAPI**: Backend web application
- **WebSocket**: Truyền dữ liệu real-time giữa client và server
- **Python**: Ngôn ngữ lập trình chính

# 2 Model YOLOv11

## 2.1 Giới thiệu

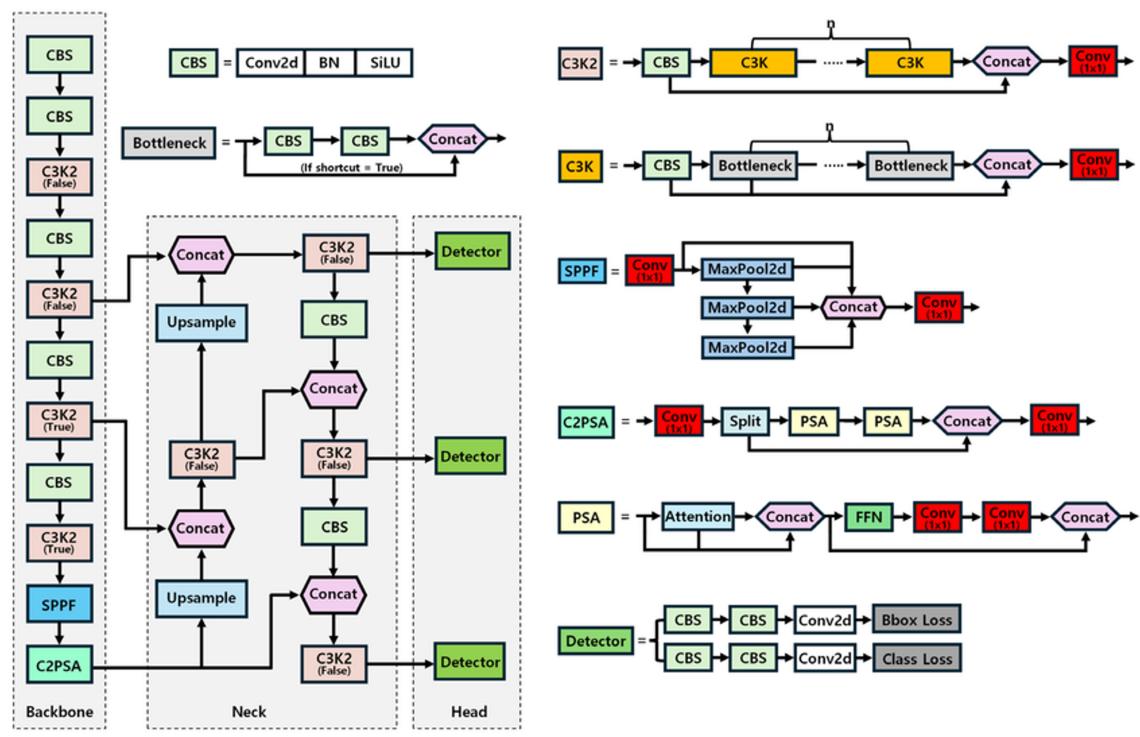
YOLOv11 là phiên bản mới nhất trong dòng YOLO của Ultralytics, được phát hành vào năm 2024. Model này mang đến những cải tiến đáng kể về kiến trúc và phương pháp training, tối ưu cho nhiều tác vụ computer vision khác nhau.



Hình 1: Sự phát triển của các phiên bản YOLO qua các năm

## 2.2 Kiến trúc

YOLOv11 có kiến trúc 3 phần chính: Backbone, Neck và Head.

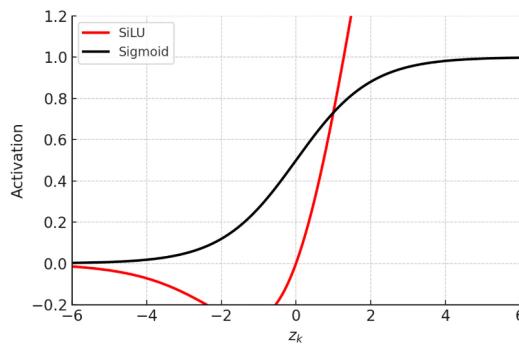


Hình 2: Kiến trúc tổng quan của YOLOv11

### 2.2.1 Backbone

Backbone sử dụng DarkNet và DarkFPN để trích xuất features. Các thành phần chính:

- **Conv Block:** Gồm Convolutional layer + Batch Normalization + SiLU activation
- **Bottleneck:** Hai Conv blocks nối tiếp với residual connection (tương tự ResNet)
- **C3K2 Block:** Phiên bản cải tiến của C2F (YOLOv8), sử dụng kernel 3x3 nhỏ hơn để trích xuất features hiệu quả hơn
- **SPPF (Spatial Pyramid Pooling Fast):** Pooling features ở nhiều scales khác nhau bằng multiple max-pooling operations

Hình 3: Hàm kích hoạt SiLU (Sigmoid Linear Unit) -  $f(x) = x \cdot \sigma(x)$ 

Backbone trích xuất 3 mức features:

- P3 ( $80 \times 80$ ): high-level features
- P4 ( $40 \times 40$ ): medium-level features
- P5 ( $20 \times 20$ ): low-level features

### 2.2.2 Neck

Neck xử lý features từ backbone trước khi đưa vào head. Thành phần quan trọng:

- **C2PSA (Cross Stage Partial with Spatial Attention)**: Block mới trong YOLOv11, kết hợp Attention mechanism + Conv + FFN để tập trung vào các vùng quan trọng trong feature map
- **Upsampling & Concatenation**: Upsample P5 → concat P4 → upsample → concat P3

C2PSA giúp model focus vào spatial information, cải thiện khả năng phát hiện small objects và partially occluded objects.

### 2.2.3 Head

Head nhận 3 feature maps và thực hiện prediction:

- **Detection Head**: Gồm DFL (Distribution Focal Loss), Box Detection, Class Detection
- **Multi-scale Predictions**: Predictions từ P3, P4, P5 giúp phát hiện objects ở nhiều kích thước
- **Anchor-based Detection**: Sử dụng anchor points và strides để xác định vị trí bounding boxes

## 2.3 Các phiên bản

Phiên bản	Params	GFLOPs	mAP	Ứng dụng
YOLOv11n (Nano)	2.6M	6.5	39.5	Edge devices
YOLOv11s (Small)	9.4M	21.5	47.0	Balanced
YOLOv11m (Medium)	20.1M	68.0	51.5	General purpose
YOLOv11l (Large)	25.3M	86.9	53.4	High accuracy
YOLOv11x (Extra Large)	56.9M	194.9	54.7	Maximum accuracy

Bảng 1: Các phiên bản YOLOv11

## 2.4 Fine-tuning

Quy trình fine-tune YOLOv11 với custom dataset:

### Bước 1: Chuẩn bị Dataset

- Thu thập images chứa các vật thể nguy hiểm (dao, kéo)

- Annotate theo format của YOLO
- Chia dataset: train/val/test (75/15/10)

### Bước 2: Cấu hình training

- Tạo file data.yaml định nghĩa paths và classes
- Chọn pretrained weights: yolov11n.pt

### Bước 3: Training

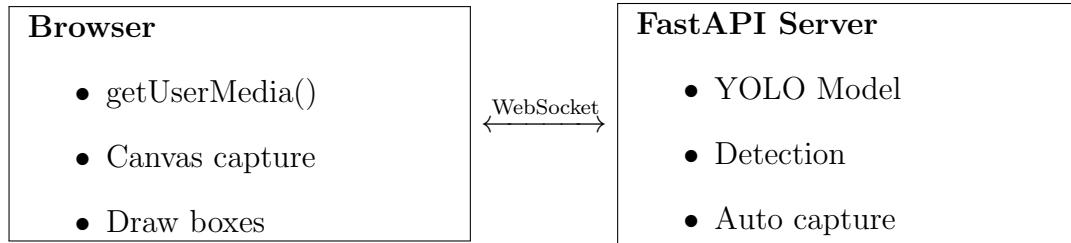
```
1 from ultralytics import YOLO
2
3 model = YOLO("yolo11s.pt")
4 model.train(
5     data="./dataset/merge.yolov11/data.yaml",
6     epochs=50,
7     patience=10,
8     batch=-1,
9     imgsz=640,
10    project=output_dir,
11    name="train",
12
13    # === Loss ===
14    box=7.5,
15    cls=1.5,          # 0.5 -> 1.5
16    dfl=1.5,
17
18    # === Regularization ===
19    dropout=0.2,
20    weight_decay=0.0005,
21
22    # === Augmentation ===
23    # Geometric
24    fliplr=0.5,
25    degrees=15,        # Rotate +/-15 deg
26    scale=0.5,         # Zoom 50-150%
27    translate=0.1,
28
29    # Mosaic & Mixup
30    mosaic=1.0,
31    mixup=0.15,
32    copy_paste=0.1,
33
34    # Color
35    hsv_h=0.015,
36    hsv_s=0.7,
37    hsv_v=0.4,
38
39    # Random erasing
40    erasing=0.3,
41 )
```

#### Bước 4: Evaluation

- Dánh giá trên test set: mAP, precision, recall
- Export model cho deployment

### 3 Xây dựng Web Application

#### 3.1 Kiến trúc hệ thống



#### 3.2 Backend

Backend được xây dựng với FastAPI, cung cấp các chức năng:

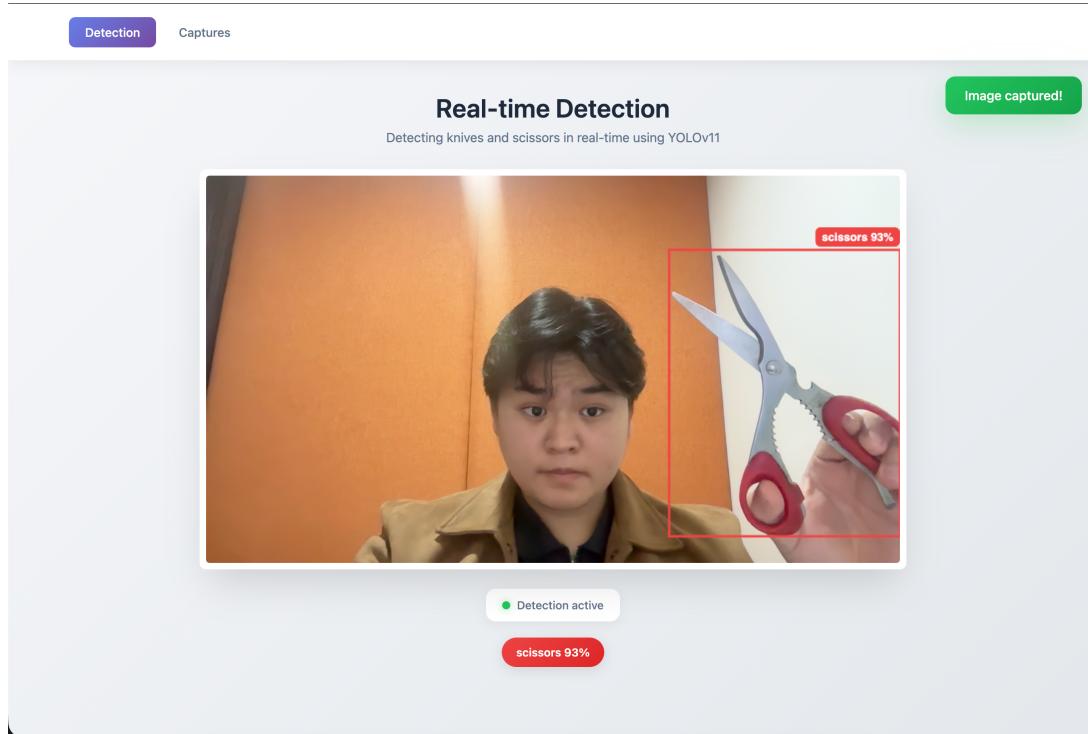
- **WebSocket endpoint:** Nhận frames từ browser, chạy detection, trả về kết quả
- **Auto capture:** Tự động lưu ảnh khi phát hiện vật thể nguy hiểm
- **REST API:** Quản lý các ảnh đã capture (list, delete)

#### 3.3 Frontend

Frontend là một Web Application với hai trang chính:

##### 1. Trang Detection

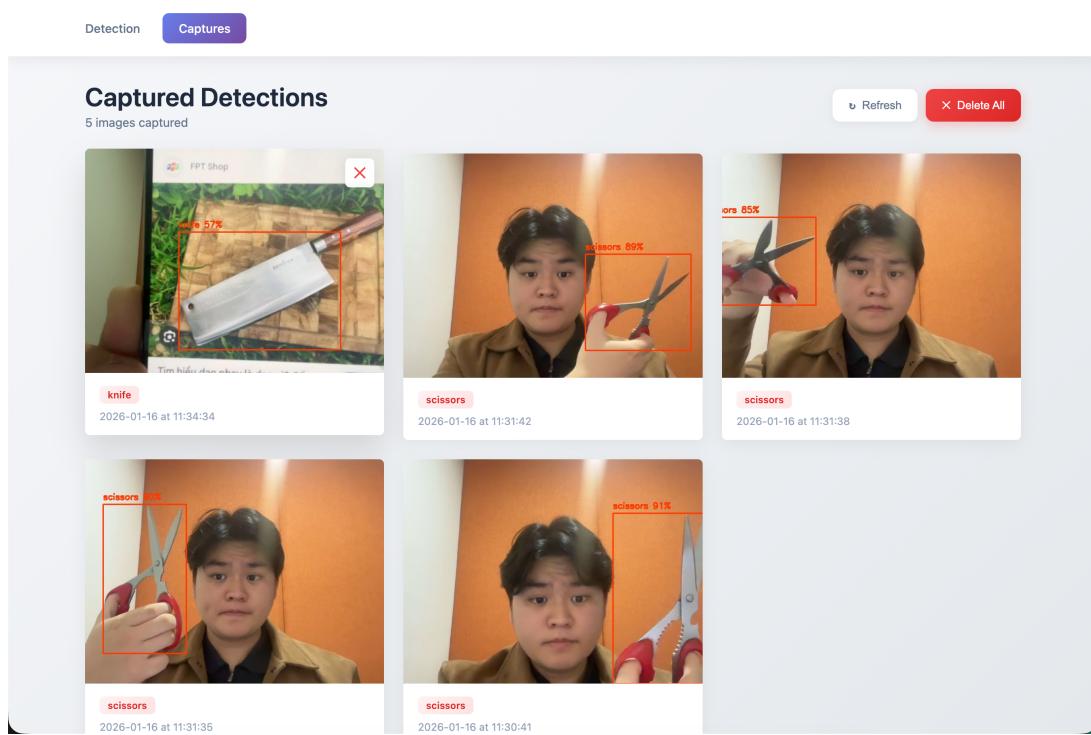
- Hiển thị video từ webcam real-time
- Vẽ bounding boxes lên các vật thể được phát hiện
- Màu đỏ cho vật thể nguy hiểm (dao, kéo)
- Hiển thị confidence score



Hình 4: Trang Detection - Phát hiện vật thể nguy hiểm real-time

## 2. Trang Captures

- Gallery hiển thị các ảnh đã capture
- Xem ảnh full-screen với modal
- Xóa từng ảnh hoặc xóa tất cả



Hình 5: Trang Captures - Quản lý các ảnh đã capture

### 3.4 Tính năng Auto Capture

Hệ thống tự động chụp và lưu ảnh khi phát hiện vật thể nguy hiểm:

- Chỉ capture khi phát hiện dao hoặc kéo
- Cooldown 3 giây giữa các lần capture để tránh spam
- Ảnh được lưu với bounding boxes và timestamp
- Tên file: YYYYMMDD\_HHMMSS\_classes.jpg

## 4 Cấu trúc dự án

```
Project/
|-- web/                                # Web application
|   |-- app.py                            # FastAPI backend
|   +-+ static/                           # Frontend files
|-- docs/                                 # Documentation
|-- captures/                            # Auto-captured images
|-- dataset/                             # Training datasets
|-- results/                             # Training results
|-- main.py                              # Standalone detection
|-- preprocessing.py                     # Dataset preparation
|-- yolov11-finetune.ipynb              # Finetuning notebook
|-- requirements.txt                      # Dependencies
+-+ README.md                            # Documentation
```

## 5 Hướng dẫn

### 5.1 Cài đặt

```

1 # Create virtual environment
2 python -m venv .venv
3 source .venv/bin/activate
4
5 # Install dependencies
6 pip install -r requirements.txt

```

### 5.2 Chạy Web Application

```

1 cd web
2 uvicorn app:app --reload

```

Mở trình duyệt tại <http://localhost:8000>

### 5.3 Training Model

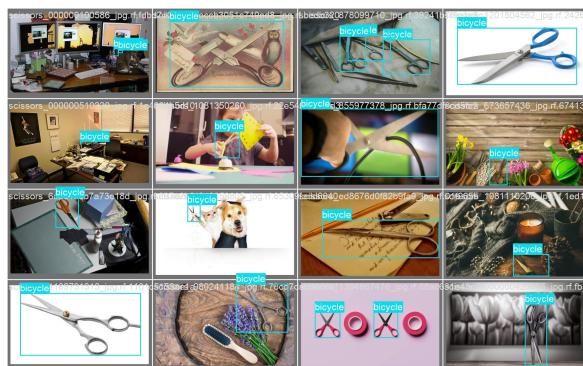
```

1 # Prepare dataset
2 python preprocessing.py
3
4 # Fine-tune model (Jupyter notebook)
5 jupyter notebook yolov11-finetune.ipynb

```

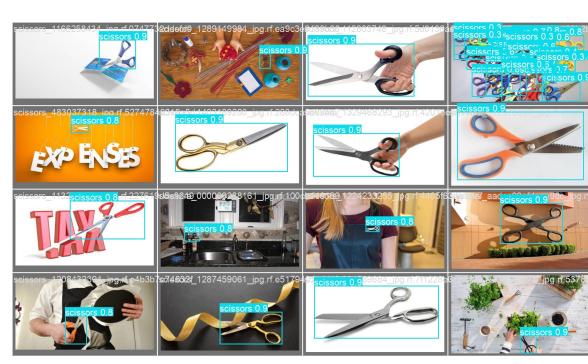
## 6 Kết quả

### 6.1 Model Performance

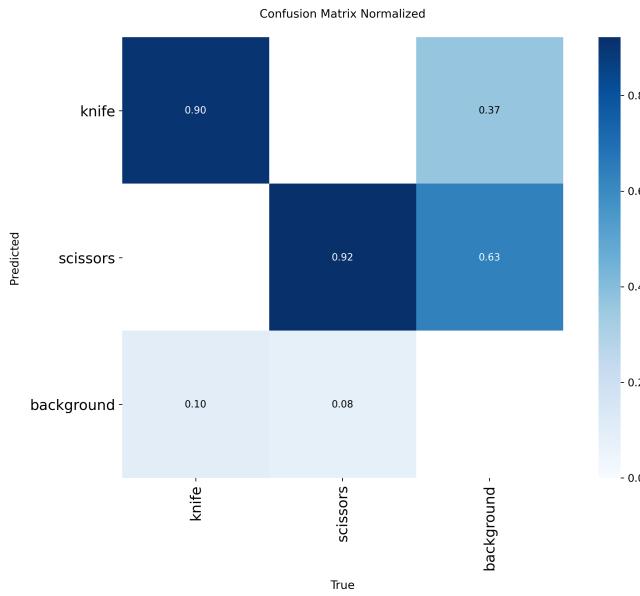


Hình 6: Pretrained model (chưa fine-tune)

Model sau khi fine-tune có khả năng nhận diện chính xác dao và kéo, trong khi model pretrained chỉ nhận diện được các object chung từ COCO dataset và có độ chính xác thấp.



Hình 7: Finetuned model (đã fine-tune)



Hình 8: Confusion Matrix (Normalized) - Dánh giá hiệu suất phân loại của model sau khi fine-tune

## 6.2 Web Application

- Real-time detection qua WebSocket
- Auto capture khi phát hiện được vật thể nguy hiểm
- Xem lại các ảnh đã được capture trong Gallery

## 7 Kết luận

Dự án đã hoàn thành các mục tiêu đề ra:

- Fine-tune YOLOv11 để nhận diện dao và kéo
- Xây dựng web application với real-time detection
- Tự động capture khi phát hiện vật thể nguy hiểm