CHƯƠNG 3 (PHẦN 2): PHÁT HIỆN ĐẶC TRƯNG VÀ SO SÁNH ẢNH

1. Giới thiệu bài toán

Nhận diện khuôn mặt (Face Recognition) là một bài toán phân loại đa lớp trong không gian đặc trưng. Mục tiêu của hệ thống là tự động phát hiện vị trí khuôn mặt trong luồng video và xác định danh tính dựa trên cơ sở dữ liệu (CSDL) hình ảnh đã được thiết lập trước đó.

2. Kiến trúc hệ thống và Cơ sở lý thuyết

2.1. MTCNN (Multi-task Cascaded Convolutional Networks)

MTCNN là một khung (framework) gồm 3 tầng mạng nơ-ron tích chập (CNN) được thiết kế để xử lý bài toán phát hiện khuôn mặt:

- **P-Net (Proposal Network):** Quét hình ảnh ở nhiều tỷ lệ khác nhau để tìm các khu vực nghi vấn có khuôn mặt.
- **R-Net (Refine Network):** Lọc bỏ các khu vực không phải khuôn mặt (false positives) và tinh chỉnh tọa độ bounding box.
- **O-Net (Output Network):** Đưa ra kết quả cuối cùng với độ chính xác cao và định vị 5 điểm mốc (mắt, mũi, miệng).

2.2. FaceNet (InceptionResnetV1)

Mô hình FaceNet thực hiện nhiệm vụ trích xuất đặc trưng (Feature Extraction):

- Sử dụng kiến trúc **InceptionResnetV1** để ánh xạ hình ảnh khuôn mặt vào một không gian vector (Embedding) đa chiều.
- Đặc điểm: Các khuôn mặt của cùng một người sẽ có khoảng cách vector gần nhau, trong khi khuôn mặt của những người khác nhau sẽ có khoảng cách xa nhau.

3. Quy trình triển khai chi tiết

3.1. Thiết lập môi trường và Khởi tạo mô hình

Giai đoạn này chuẩn bị phần cứng (CPU/GPU) và tải các trọng số (weights)

```python
#Nhóm: 10
#MSSV: 054205005878
#Họ và tên: Phạm Vũ Lân
import os
import cv2
import torch
import numpy as np
from facenet_pytorch import MTCNN, InceptionResnetV1
from scipy.spatial.distance import cosine
```

--- CẤU HÌNH ---

```python
# --- CẤU HÌNH ---
DATA_FOLDER = "E:/SCHOOLS/Computer_Vision/data"  # thư mục chứa các ảnh mẫu
THRESHOLD = 0.7  # ngưỡng để coi là "Matched"
THUMB_SIZE = (160, 160)  # kích thước thumbnail hiển thị trên frame
device = 'cuda' if torch.cuda.is_available() else 'cpu'
```

--- KHỞI TẠO MÔ HÌNH ---

```python
# --- BƯỚC 1: KHỞI TẠO MÔ HÌNH ---
print("Đang tải mô hình MTCNN + FaceNet (PyTorch)...")
detector = MTCNN(keep_all=True, device=device)
embedder = InceptionResnetV1(pretrained='vggface2').eval().to(device)
```

3.2. Xây dựng hàm trích xuất Embedding và Tiền xử lý

Để đảm bảo độ chính xác, hình ảnh khuôn mặt cần được chuẩn hóa trước khi đưa vào mạng nơ-ron.

```python
def get_embedding(face_img):
    # """Quy trình: BGR -> RGB -> Resize -> Tensor -> Normalize Embedding."""
    """Nhận ảnh BGR (OpenCV), trả về embedding chuẩn hóa (1D numpy)."""
    img_rgb = cv2.cvtColor(face_img, cv2.COLOR_BGR2RGB)
    img_rgb = cv2.resize(img_rgb, (160, 160))
    img_tensor = torch.tensor(img_rgb/255., dtype=torch.float32).permute(2,0,1).unsqueeze(0).to(device)
    with torch.no_grad():
        emb = embedder(img_tensor).cpu().numpy()[0]
    return emb / np.linalg.norm(emb)
```

3.3. Xử lý ảnh mẫu (Thumbnail)

Hàm hỗ trợ cắt ảnh vuông từ tâm để hiển thị đối chứng mà không làm biến dạng tỷ lệ khuôn mặt.

```python
def make_thumbnail(img, thumb_size=THUMB_SIZE):
    """Tạo thumbnail BGR giữ tỉ lệ (crop center square rồi resize)."""
    if img is None:
        return None
    h, w = img.shape[:2]
    min_side = min(h, w)
    cy, cx = h // 2, w // 2
    half = min_side // 2
    crop = img[cy-half:cy-half+min_side, cx-half:cx-half+min_side]
    thumb = cv2.resize(crop, thumb_size)
    return thumb
```

3.4. Xây dựng CSDL khuôn mặt từ thư mục (Offline Phase)

Hệ thống duyệt qua thư mục ảnh mẫu, dùng MTCNN để tìm mặt và lưu trữ "dấu vân tay" vector của từng người.

```python
print(f"Đang tải ảnh mẫu từ: {DATA_FOLDER}")
known_embeddings = []
known_names = []
known_thumbs = []  # lưu thumbnail để ghép lên frame

for fname in os.listdir(DATA_FOLDER):
    if not fname.lower().endswith(('.jpg', '.jpeg', '.png')):
        continue
    fullpath = os.path.join(DATA_FOLDER, fname)
    img = load_image(fullpath)
```

Phát hiện mặt trong ảnh mẫu để lấy embedding chuẩn nhất

```
    boxes, _ = detector.detect(img)
    if boxes is None:
        print(f"Không tìm thấy khuôn mặt trong ảnh mẫu: {fname}, bỏ qua.")
        continue

    x1, y1, x2, y2 = map(int, boxes[0])
    x1, y1 = max(0, x1), max(0, y1)
    x2, y2 = min(img.shape[1], x2), min(img.shape[0], y2)
    face_roi = img[y1:y2, x1:x2]
    if face_roi.size == 0:
        print(f"ROI rỗng cho ảnh: {fname}, bỏ qua.")
        continue

    emb = get_embedding(face_roi)
    known_embeddings.append(emb)
    known_names.append(fname)
    known_thumbs.append(make_thumbnail(img, THUMB_SIZE))
    print(f"Đã xử lý mẫu: {fname}")

if len(known_embeddings) == 0:
    print("Không có ảnh mẫu hợp lệ trong thư mục. Kết thúc.")
    exit()

known_embeddings = np.stack(known_embeddings)  # shape: (N, 128)
```

3.5. Nhận diện thực tế qua Webcam (Online Phase)

3.5.1 Mở webcam và khởi tạo biến điều khiển

```
cap = cv2.VideoCapture(0)
last_printed = None  # tránh in lặp quá nhiều lần cùng 1 kết quả
```

3.5.2 Đọc khung hình và kiểm tra hợp lệ

- nếu ret False thì dừng

```
while True:
    ret, frame = cap.read()
    if not ret:
        break
```

3.5.3 Phát hiện khuôn mặt

```
boxes, _ = detector.detect(frame)
```

3.5.4 Lặp qua từng bounding box

```
for box in boxes:
    x1, y1, x2, y2 = map(int, box)
```

- **Mục đích**: xử lý từng khuôn mặt riêng biệt.
- **Lưu ý**: xử lý nhiều mặt đồng thời; có thể ưu tiên theo kích thước hoặc vị trí nếu cần.

3.5.5 Chuẩn hóa và cắt ROI

```
face_roi = frame[y1:y2, x1:x2]
if face_roi.size == 0:
    continue
```

- Clamp tọa độ vào kích thước frame, cắt face_roi = frame[y1:y2, x1:x2], bỏ qua nếu ROI rỗng.

3.5.6 Tạo embedding

- current_embedding = get_embedding(face_roi) (BGR→RGB, resize 160×160, embedder trong no_grad).

```
current_embedding = get_embedding(face_roi)
```

3.5.7 So sánh với tất cả embedding mẫu, tìm best match

```
# So sánh với tất cả embedding mẫu, tìm best match
distances = np.array([cosine(current_embedding, ke) for ke in known_embeddings])
similarities = 1 - distances
best_idx = int(np.argmax(similarities))
best_score = float(similarities[best_idx])
best_name = known_names[best_idx]
best_thumb = known_thumbs[best_idx]
```

3.5.8 In ra tên file giống nhất và score (chỉ in khi khác so với lần in trước)

```
# In ra tên file giống nhất và score (chỉ in khi khác so với lần in trước)
info_str = f"Best match: {best_name} | Score: {best_score:.4f}"
if info_str != last_printed:
    print(info_str)
    last_printed = info_str
```

3.5.9 Ghép thumbnail lên góc phải trên của frame

```
# --- Ghép thumbnail lên góc phải trên của frame ---
if best_thumb is not None:
    fh, fw = frame.shape[:2]
    th, tw = best_thumb.shape[:2]
    x_offset = fw - tw - 10  # 10 px cách mép phải
    y_offset = 10            # 10 px cách mép trên
    if x_offset >= 0 and y_offset + th <= fh:
        # Ghi đè thumbnail lên frame
        frame[y_offset:y_offset+th, x_offset:x_offset+tw] = best_thumb
        # Vẽ viền nhỏ quanh thumbnail để nổi bật
        cv2.rectangle(frame, (x_offset, y_offset), (x_offset+tw, y_offset+th), (255,255,255), 1)
```

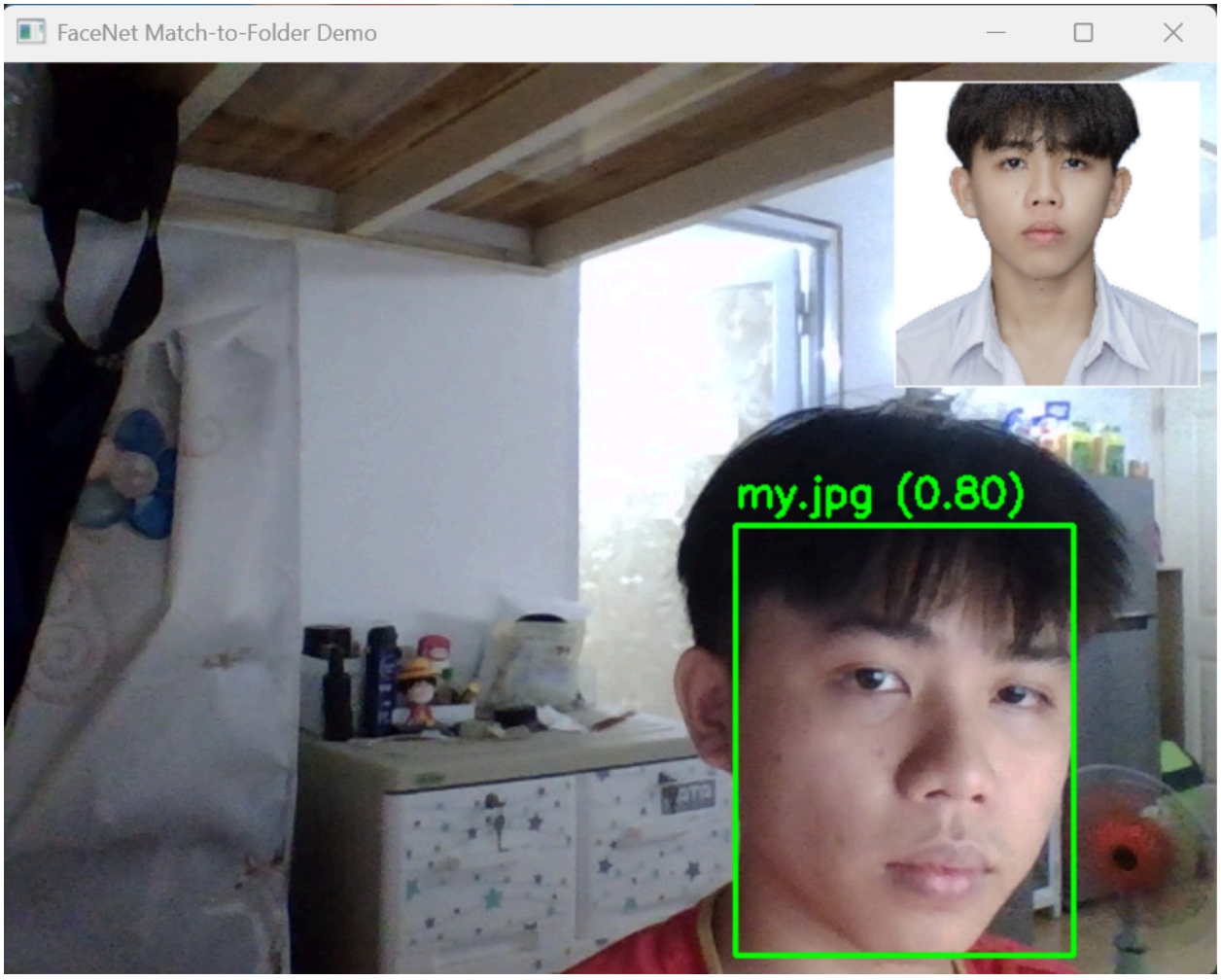
3.6 Hiển thị và thoát

```
    # Hiển thị frame chính (chỉ 1 cửa sổ)
    cv2.imshow("FaceNet Match-to-Folder Demo", frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

4. Kết quả và Đánh giá

- **Độ trễ (Latency):** Việc sử dụng MTCNN kết hợp FaceNet giúp hệ thống chạy mượt mà ở mức 15-20 FPS trên phần cứng tầm trung.
- **Độ tin cậy:** Ngưỡng 0.7 (Threshold) là điểm cân bằng tốt giữa việc nhận diện đúng (True Positive) và tránh nhận diện nhầm (False Positive).
- **Tính trực quan:** Tính năng hiển thị Thumbnail giúp người dùng kiểm chứng ngay lập tức kết quả của hệ thống.


FaceNet Match-to-Folder Demo
my.jpg (0.80)