Intelligence System:

Face Recognition in Computer Vision

សាកលវិទ្យាល័យនរតុន
NORTON UNIVERSITY

Intelligence System Development

2024 – 2025

Y4E1 – DCS – NU

By: SEK SOCHEAT

Advisor to DCS and Lecturer

Mobile: 017 879 967

Email: socheat.sek@gmail.com

Table of Contents

1. Introduction to Face Recognition

1.1. Overview of Face Recognition Systems

1. Definition:

- A biometric system to identify or verify a person from a digital image or video frame.

2. Applications:

- **Security**: Surveillance systems, restricted access to secure areas.
- **Authentication**: Unlocking devices, attendance systems.
- **Personalization**: Recommender systems, social media tagging.

3. Key Processes:

- **Detection**: Locating faces in an image or video.
- **Recognition**: Matching the detected face to a stored identity.

1. Introduction to Face Recognition

1.1. Overview of Face Recognition Systems

4. Challenges:

- Variations in lighting, pose, occlusion (e.g., sunglasses, masks), and age progression.

5. Comparison:

- **Face Detection**: Identifying the presence and location of a face.
- **Face Recognition**: Identifying "who" the face belongs to.

6. Evolution of Methods:

- From traditional feature-based methods (e.g., Eigenfaces) to deep learning models like CNNs.

7. Integration with Intelligent Systems:

- Used in AI systems for adaptive decision-making, IoT, and human-computer interaction.

1. Introduction to Face Recognition

1.2. Key Concepts

1. Feature Extraction:

- Identify unique facial attributes such as the eyes, nose, mouth, and contours to create a facial signature.
- ***Methods:*** Histogram of Oriented Gradients (HOG), deep learning embeddings.

2. Facial Embeddings:

- Represent faces as high-dimensional vectors using models like FaceNet.
- These embeddings capture key facial features while ignoring irrelevant details like background or lighting.

1. Introduction to Face Recognition

1.2. Key Concepts

3. Deep Learning Models:

- Use Convolutional Neural Networks (CNNs) for feature extraction and recognition.
- Pre-trained models like **FaceNet**, **DeepFace**, or **VGGFace** are commonly used.

4. Distance Metrics:

- Determine similarity between two facial embeddings:
  - **Euclidean Distance**: Measures straight-line difference between vectors.
  - **Cosine Similarity**: Measures angle between vectors.

1. Introduction to Face Recognition

1.2. Key Concepts

5. Face Recognition Workflow:

- **Detection**: Locate face regions in images or video.
- **Alignment**: Normalize faces (e.g., rotate, crop) for consistent representation.
- **Encoding**: Convert the face into a numerical embedding.
- **Matching**: Compare the embedding with known embeddings in a database.

6. Face Matching Types:

- **Verification**: Check if two faces belong to the same person.
- **Identification**: Match a face to an identity in a database.

1. Introduction to Face Recognition

1.2. Key Concepts

7. Role of Training Data:

• Large and diverse datasets are critical to train models that generalize well across different lighting, poses, and ages.

8. Tools and Libraries:

• Popular Python libraries like **face_recognition**, **OpenCV**, and **dlib** provide pre-built tools for face detection and recognition.

1. Introduction to Face Recognition

1.3. Libraries in Python

1. OpenCV:

- **Purpose**: A comprehensive library for computer vision tasks.
- **Features**: Supports face detection with Haar cascades, DNN-based models, and real-time processing.
- **Example Use**: Detect faces in images or video streams.
- **Installation:**

```
pip install opencv-python
```

1. Introduction to Face Recognition

1.3. Libraries in Python

2. dlib:

- **Purpose**: A machine learning library with robust tools for facial landmark detection.
- **Features**: Provides face alignment, feature extraction, and face embeddings.
- **Example Use**: Precise facial landmarks for recognition preprocessing.
- **Installation:**

```
pip install dlib
```

1. Introduction to Face Recognition

1.3. Libraries in Python

3. face_recognition:

- **Purpose**: Built on dlib, it simplifies face recognition tasks.
- **Features**: Includes functions for face detection, encoding, and matching.
- **Example Use**: Match faces from images with minimal coding effort.
- **Installation:**

```
pip install face_recognition
```

1. Introduction to Face Recognition

1.3. Libraries in Python

4. DeepFace:

- **Purpose**: A wrapper for several pre-trained deep learning face recognition models (e.g., VGG-Face, OpenFace, DeepFace).
- **Features**: High accuracy, support for multiple models, and ensemble methods.
- **Example Use**: Advanced recognition tasks with pre-trained models.
- **Installation:**

```
pip install deepface
```

1. Introduction to Face Recognition

1.3. Libraries in Python

5. MTCNN (Multi-task Cascaded Convolutional Networks)

- **Purpose**: Detect faces with high accuracy and robustness.
- **Features**: Ideal for detecting faces with occlusions or in challenging lighting.
- **Example Use**: Face detection as preprocessing for recognition.
- **Installation:**

```
pip install mtcnn
```

1. Introduction to Face Recognition

1.3. Libraries in Python

6. PyTorch and TensorFlow

- **Purpose**: Libraries for building and training custom face recognition models.
- **Features**: Flexibility in designing neural networks and training on custom datasets.
- **Example Use**: Fine-tune face recognition models for specific applications.
- **Installation:**

```
pip install torch tensorflow
```

1. Introduction to Face Recognition

1.3. Libraries in Python

7. Haar Cascades (OpenCV)

- **Purpose**: Classic method for detecting faces using cascades of classifiers.
- **Features**: Lightweight and fast for simple detection tasks.
- **Example Use**: Basic face detection in low-complexity projects.
- **Installation:**

```
pip install opencv-python
pip install opencv-contrib-python
```

1. Introduction to Face Recognition

1.3. Libraries in Python

8. Keypoint Detection (mediapipe by Google)

- **Purpose**: Real-time face mesh and keypoint detection.
- **Features**: 3D facial keypoints, useful for animation or AR applications.
- **Example Use**: Detect and visualize key facial landmarks in a webcam stream.
- **Installation:**

```
pip install mediapipe
```

Example:

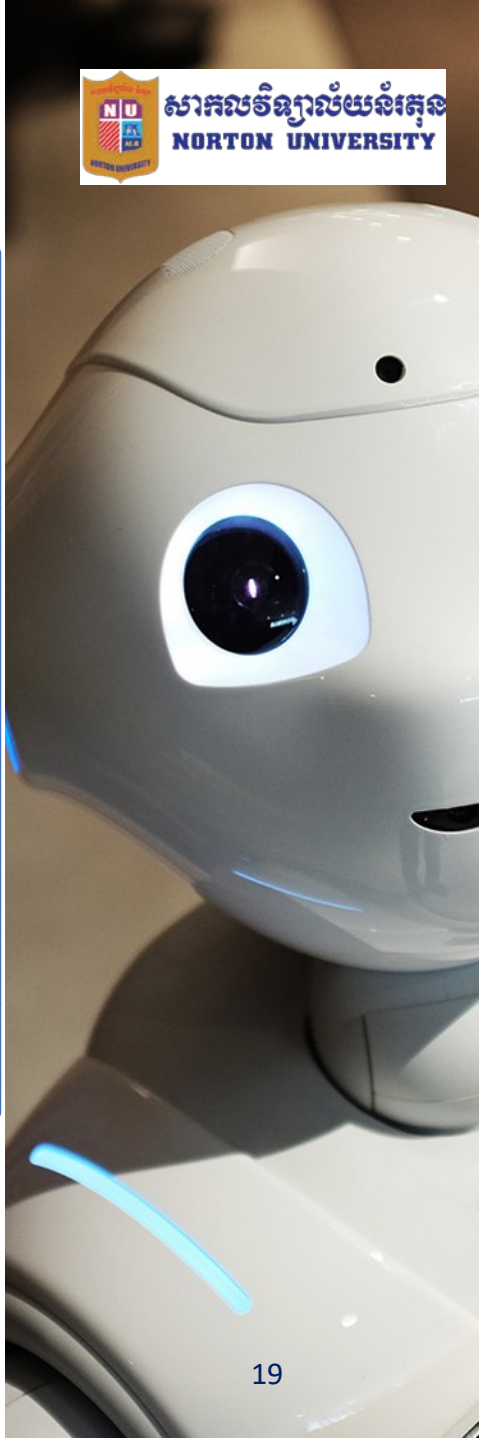
1. Real-Time Face Recognition Using MediaPipe and DeepFace
2. Webcam-Based Face Detection and Recognition System
3. Real-Time Face Recognition with Pretrained Models
4. Integrating MediaPipe and DeepFace for Face Matching
5. Lightweight Face Detection with MediaPipe
6. Real-Time Multi-Face Recognition Application
7. Building a Webcam Face Recognition System
8. Combining Face Detection and Recognition in Real-Time
9. Efficient Face Recognition Using MediaPipe and DeepFace
10. Real-Time Facial Identity Detection via Webcam

```python
import cv2
import mediapipe as mp
from deepface import DeepFace

# Initialize Mediapipe face detector
mp_face_detection = mp.solutions.face_detection
# Directory containing known faces
KNOWN_FACES_DIR = 'member_faces'

# Main function for face recognition
def main():
    # Initialize webcam
    cap = cv2.VideoCapture(0)

    # Initialize Mediapipe face detection
    with mp_face_detection.FaceDetection(min_detection_confidence=0.2) as face_detection:
        while True:
            ret, frame = cap.read()
            if not ret:
                print("Failed to read frame from webcam. Exiting...")
                break

            # Convert frame to RGB
            rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

            # Detect faces
            results = face_detection.process(rgb_frame)

            if results.detections:
                for detection in results.detections:
                    bboxC = detection.location_data.relative_bounding_box
                    ih, iw, _ = frame.shape
                    x, y, w, h = (
                        int(bboxC.xmin * iw),
                        int(bboxC.ymin * ih),
                        int(bboxC.width * iw),
                        int(bboxC.height * ih)
                    )
```

```
39
40                # Extract face image
41                face_img = rgb_frame[y:y+h, x:x+w]
42
43                # Recognize face
44                try:
45                    result = DeepFace.find(
46                        img_path=face_img,
47                        db_path=KNOWN_FACES_DIR,
48                        model_name='Facenet',
49                        enforce_detection=False
50                    )
51                    # Extract identity
52                    name = result[0].iloc[0]['identity'].split('/')[-1].split('.')[0] if not result[0].empty else "Unknown"
53                    color = (0, 255, 0)  # Green for known faces
54                except Exception as e:
55                    print(f"Error during face recognition: {e}")
56                    name = "Error"
57                    color = (0, 0, 255)  # Red for errors
58
59                # Display name and draw bounding box
60                cv2.putText(frame, name, (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)
61                cv2.rectangle(frame, (x, y), (x+w, y+h), color, 2)
62
63            # Display the frame
64            cv2.imshow('Face Recognition', frame)
65
66            # Exit if 'q' is pressed
67            if cv2.waitKey(1) & 0xFF == ord('q'):
68                break
69
70        # Release resources
71        cap.release()
72        cv2.destroyAllWindows()
73
74    if __name__ == "__main__":
75        main()
```

4. Homework

1. What is the primary goal of a face recognition system, and how does it differ from face detection?
2. Describe the key components of a face recognition system and explain how they work together to identify individuals.
3. What are the two main stages in face recognition, and what is the role of feature extraction in this process?
4. Name three popular Python libraries used for face recognition and briefly describe their functionality.



សាកលវិទ្យាល័យនរតុន
NORTON UNIVERSITY

Thank you