Face Recognition Attendance System

A complete, production-ready real-time face recognition attendance system using deep learning and Siamese neural networks.

Features

✅ **Real-time Face Recognition** - Detect and recognize faces from webcam feed
✅ **Siamese Neural Network** - Advanced verification using face embeddings
✅ **Multiple Detection Methods** - MTCNN, MediaPipe, or OpenCV
✅ **Pre-trained Embeddings** - FaceNet or MobileFaceNet models
✅ **Automatic Attendance Logging** - CSV logs with timestamps and photos
✅ **Easy Person Addition** - Add new people without retraining
✅ **Modular Architecture** - Clean, maintainable code structure
✅ **GUI Application** - User-friendly Tkinter interface
✅ **Configurable Threshold** - Adjust recognition sensitivity

---

System Architecture

```
┌─────────────────────────────────────────┐
│           Camera Input                  │
└──────────────────┬──────────────────────┘
                   │
                   ▼
┌─────────────────────────────────────────┐
│        Face Detection Module            │
│    (MTCNN / MediaPipe / OpenCV)         │
└──────────────────┬──────────────────────┘
                   │
                   ▼
┌─────────────────────────────────────────┐
│       Face Embedding Module             │
│      (FaceNet / MobileFaceNet)          │
└──────────────────┬──────────────────────┘
                   │
                   ▼
┌─────────────────────────────────────────┐
│      Siamese Network Verification       │
│    (Compares embeddings for identity)   │
└──────────────────┬──────────────────────┘
                   │
                   ▼
┌─────────────────────────────────────────┐
│         Attendance Logger               │
│    (CSV logs + photos + timestamp)                │
└───────────────────────────────────────────────────┘
```

Project Structure

```
face_recognition_attendance/
├── config.py                  # Configuration and paths
├── face_detection.py          # Face detection module
├── face_embedding.py          # Face embedding extraction
├── siamese_network.py         # Siamese network architecture
├── dataset_manager.py         # Dataset management
├── attendance_logger.py       # Attendance logging
├── face_recognizer.py         # Main recognition logic
├── gui_app.py                 # GUI application
├── train_model.py             # Training script
├── run_gui.py                 # Run GUI application
├── run_camera.py              # Run camera recognition (CLI)
├── add_person.py              # Add person script (CLI)
├── requirements.txt           # Dependencies
├── README.md                  # This file
└── data/
    ├── faces/                 # Face images dataset
    │   ├── person1/
    │   ├── person2/
    │   └── ...
    ├── embeddings/            # Stored embeddings
    │   ├── embeddings.pkl
    │   └── metadata.json
    ├── models/                # Trained models
    │   └── siamese_model.h5
    ├── logs/                  # System logs
    └── attendance/            # Attendance records
        ├── attendance.csv
        └── photos/
```

Installation

1. Prerequisites

- Python 3.8 or higher
- Webcam/camera
- 4GB+ RAM recommended

2. Clone or Download

```bash
# Create project directory
mkdir face_recognition_attendance
cd face_recognition_attendance

# Copy all provided .py files into this directory
```

3. Install Dependencies

```bash
pip install -r requirements.txt
```

Note on face detection methods:

- **MTCNN** (recommended): Most accurate, requires TensorFlow
- **MediaPipe**: Fast, lightweight, good for real-time
- **OpenCV**: Fallback option, always available

Quick Start

Step 1: Add People to Dataset

Option A: Using GUI

```bash
python run_gui.py
# Click "Add Person" button
# Enter name and follow on-screen instructions
```

Option B: Using CLI

```bash
python add_person.py
# Follow prompts to add each person
```

Important: Add at least 2 people before training!

Step 2: Train Siamese Network

```bash
```

```
python train_model.py
```

This will:

- Load all face embeddings
- Create training pairs (same person vs different person)
- Train the Siamese network
- Save the trained model

Step 3: Run the System

Option A: GUI Application (Recommended)

```bash
python run_gui.py
```

Option B: Command Line

```bash
python run_camera.py
```

Module Explanations

1. config.py - Configuration

Central configuration for all system parameters:

- Face detection settings
- Embedding model selection
- Siamese network parameters
- Camera settings
- Attendance logging options

Key configurations:

```python
```

```
FACE_DETECTION_METHOD = "mtcnn"  # or "mediapipe" or "opencv"
EMBEDDING_MODEL = "facenet"      # or "mobilefacenet"
SIMILARITY_THRESHOLD = 0.6       # Recognition threshold (0-1)
LOG_INTERVAL = 300               # Seconds between duplicate logs
```

2. face_detection.py - Face Detection

Unified interface for multiple detection methods:

- **MTCNN**: Deep learning-based, most accurate
- **MediaPipe**: Google's solution, fast and efficient
- **OpenCV**: Traditional Haar Cascade, fallback option

Key features:

- Automatic fallback if preferred method unavailable
- Consistent API across all methods
- Face extraction with margins
- Preprocessing for embedding models

3. face_embedding.py - Face Embeddings

Extracts face embeddings using pre-trained models:

- **FaceNet**: 512-dimensional embeddings
- **MobileFaceNet**: 128-dimensional embeddings (faster)

How it works:

1. Preprocesses face images
2. Passes through pre-trained CNN
3. Extracts embedding vector
4. L2-normalizes embeddings

Note: Uses simplified models by default. Install `keras-facenet` for best accuracy.

4. siamese_network.py - Siamese Network

Learns to verify if two face embeddings belong to the same person.

Architecture:

```
Input 1 (512-dim) ──┐
                    ├─> L1 Distance ─> Dense Layers ─> Similarity (0-1)
Input 2 (512-dim) ──┘
```

Training:

- Creates positive pairs (same person) and negative pairs (different people)
- Binary classification: 1 = same person, 0 = different
- Uses contrastive learning approach

Why Siamese Network?

- Better than simple distance metrics
- Learns robust similarity function
- Works well with limited data per person
- Can add new people without retraining (just update embeddings)

5. dataset_manager.py - Dataset Management

Handles all dataset operations:

- **Add Person**: Captures faces from camera
- **Store Embeddings**: Saves embeddings with metadata
- **Rebuild Embeddings**: Regenerates if model changes
- **Statistics**: Dataset info and health checks

Storage structure:

```
data/faces/
  ├── john_doe/
  │    ├── john_doe_001.jpg
  │    ├── john_doe_002.jpg
  │    └── ...
  └── jane_smith/
      └── ...

data/embeddings/
  ├── embeddings.pkl      # {name: [embedding1, embedding2, ...]}
  └── metadata.json      # Timestamps, counts, etc.
```

6. attendance_logger.py - Attendance Logging

Manages attendance records:

- **CSV Logging**: Name, date, time, similarity score
- **Photo Storage**: Saves detected face images
- **Duplicate Prevention**: Configurable time interval
- **Query Methods**: Get attendance by date, person, or all

CSV Format:

```csv
Name,Date,Time,Timestamp,Similarity,Photo
John Doe,2024-01-15,09:30:45,1705315845.123,0.8234,john_doe_20240115_093045.jpg
```

7. face_recognizer.py - Recognition Engine

Main recognition logic that combines all components:

Recognition Pipeline:

1. Detect faces in frame
2. Extract face regions
3. Generate embeddings
4. Compare with known embeddings using Siamese network
5. Return identity and confidence

Key methods:

- `recognize_face()`: Single face recognition
- `recognize_from_frame()`: All faces in frame
- `run_camera()`: Real-time camera recognition
- `verify_person()`: Verify specific person

8. gui_app.py - GUI Application

Tkinter-based user interface with:

- **Live camera feed** with real-time recognition
- **Add person** functionality
- **Threshold adjustment** slider
- **Dataset statistics** viewer
- **Attendance display** and export

- **System log** for monitoring

9. Training & Execution Scripts

- **train_model.py**: Train Siamese network
- **run_gui.py**: Launch GUI application
- **run_camera.py**: CLI camera recognition
- **add_person.py**: CLI person addition

Configuration Guide

Adjusting Recognition Threshold

The similarity threshold determines how strict the recognition is:

- **0.3-0.5**: Very lenient (may have false positives)
- **0.6-0.7**: Balanced (recommended)
- **0.8-0.9**: Very strict (may miss some matches)

Adjust in two ways:

1. Edit `Config.SIMILARITY_THRESHOLD` in `config.py`
2. Use slider in GUI application

Choosing Face Detection Method

MTCNN (Best accuracy)

```python
FACE_DETECTION_METHOD = "mtcnn"
```

- Pros: Most accurate, handles multiple faces, detects landmarks
- Cons: Slower, requires TensorFlow

MediaPipe (Best speed)

```python
FACE_DETECTION_METHOD = "mediapipe"
```

- Pros: Very fast, efficient, good accuracy
- Cons: Requires mediapipe library

OpenCV (Fallback)

```python
FACE_DETECTION_METHOD = "opencv"
```

- Pros: Always available, no extra dependencies
- Cons: Less accurate, may miss faces

Embedding Model Selection

FaceNet (Better accuracy)

```python
EMBEDDING_MODEL = "facenet"
EMBEDDING_DIM = 512
```

MobileFaceNet (Faster)

```python
EMBEDDING_MODEL = "mobilefacenet"
EMBEDDING_DIM = 128
```

Attendance Logging

Configure log interval:

```python
LOG_INTERVAL = 300  # Seconds (5 minutes)
```

Disable photo saving:

```python
# In run_*.py scripts
attendance_logger = AttendanceLogger(
    attendance_file=Config.ATTENDANCE_FILE,
    log_interval=Config.LOG_INTERVAL,
    save_photos=False  # Set to False
)
```

Usage Examples

Example 1: Basic Setup

```bash
# 1. Add first person
python add_person.py
# Enter name: Alice
# [Capture 10 images]

# 2. Add second person
python add_person.py
# Enter name: Bob
# [Capture 10 images]

# 3. Train model
python train_model.py
# [Training completes]

# 4. Run system
python run_gui.py
# [GUI starts, camera activates]
```

Example 2: Adding Person via GUI

1. Launch GUI: `python run_gui.py`
2. Click "Add Person"
3. Enter name in text field
4. Camera opens automatically
5. Look at camera, hold still
6. System captures images automatically
7. Person added, embeddings updated
8. Ready for recognition immediately

Example 3: Viewing Attendance

Via GUI:

- Click "Show Today's Attendance"

Via Python:

```python
```

```
from attendance_logger import AttendanceLogger
from config import Config

logger = AttendanceLogger(Config.ATTENDANCE_FILE)

# Today's attendance
records = logger.get_today_attendance()
for record in records:
    print(f"{record['Name']} - {record['Time']}")

# Specific person
alice_records = logger.get_person_attendance("Alice")
print(f"Alice attended {len(alice_records)} times")
```

Example 4: Rebuilding Embeddings

If you change the embedding model:

```bash
# Edit config.py
EMBEDDING_MODEL = "mobilefacenet"  # Changed from facenet
EMBEDDING_DIM = 128

# Rebuild embeddings
python -c "
from face_detection import FaceDetector
from face_embedding import FaceEmbedder
from dataset_manager import DatasetManager
from config import *

detector = FaceDetector()
embedder = FaceEmbedder(Config.EMBEDDING_MODEL)
manager = DatasetManager(DATASET_DIR, EMBEDDINGS_DIR)
manager.rebuild_embeddings(detector, embedder)
"

# Retrain Siamese network
python train_model.py
```

Troubleshooting

Issue: Camera not opening

Solution:

```python
# In config.py, try different camera IDs
CAMERA_ID = 1  # or 2, 3, etc.
```

Issue: Low recognition accuracy

Solutions:

1. Add more images per person (15-20 recommended)
2. Ensure good lighting during image capture
3. Capture images from different angles
4. Lower similarity threshold (try 0.5)
5. Use MTCNN instead of OpenCV
6. Install keras-facenet for better embeddings

Issue: "No trained model found"

Solution:

```bash
# Train the model first
python train_model.py
```

Issue: Faces not detected

Solutions:

1. Check lighting conditions
2. Try different detection method
3. Adjust `MIN_FACE_SIZE` in config.py
4. Lower `DETECTION_CONFIDENCE` threshold

Issue: Slow performance

Solutions:

1. Switch to MediaPipe detection
2. Use MobileFaceNet embeddings
3. Reduce camera resolution
4. Process every Nth frame only

Issue: Import errors

Solution:

```bash
# Reinstall dependencies
pip install --upgrade -r requirements.txt

# For TensorFlow issues on Apple Silicon:
pip install tensorflow-macos tensorflow-metal
```

Advanced Customization

Custom Similarity Metric

Edit `siamese_network.py`:

```python
# Instead of L1 distance, use cosine similarity
from tensorflow.keras.layers import Dot, Lambda

dot_product = Dot(axes=1, normalize=True)([input_1, input_2])
```

Adding More Face Features

Extract additional features:

```python
# In face_recognizer.py
def recognize_face_with_features(self, face_image):
    embedding = self.face_embedder.get_embedding(face_image)

    # Add age, gender, emotion detection here
    # Using additional models

    return name, similarity, features
```

Custom Attendance Report

```python
```

```
from attendance_logger import AttendanceLogger
import pandas as pd

logger = AttendanceLogger(Config.ATTENDANCE_FILE)

# Create custom report
df = pd.read_csv(Config.ATTENDANCE_FILE)
summary = df.groupby('Name').agg({
    'Date': 'count',
    'Similarity': 'mean'
})
summary.to_excel('attendance_report.xlsx')
```

Performance Optimization

For Real-time Performance:

```python
# config.py optimizations
FACE_DETECTION_METHOD = "mediapipe"  # Fastest
EMBEDDING_MODEL = "mobilefacenet"    # Faster than FaceNet
FRAME_WIDTH = 320                    # Lower resolution
FRAME_HEIGHT = 240

# Process every Nth frame
frame_skip = 3  # Process every 3rd frame
```

For Accuracy:

```python
# config.py for accuracy
FACE_DETECTION_METHOD = "mtcnn"
EMBEDDING_MODEL = "facenet"
DETECTION_CONFIDENCE = 0.95
SIMILARITY_THRESHOLD = 0.7
IMAGES_PER_PERSON = 20  # More training images
```

API Reference

FaceDetector

```python
```

```
detector = FaceDetector(method="mtcnn", min_face_size=40, confidence=0.9)

# Detect faces
faces = detector.detect(frame)  # Returns [(x,y,w,h), ...]

# Extract face
face = detector.extract_face(frame, bbox, target_size=(160, 160))
```

FaceEmbedder

```python
embedder = FaceEmbedder(model_name="facenet")

# Get embedding
embedding = embedder.get_embedding(face_image)  # Returns np.array (512,)

# Batch processing
embeddings = embedder.get_embeddings_batch([face1, face2, ...])
```

SiameseNetwork

```python
siamese = SiameseNetwork(embedding_dim=512)

# Train
siamese.train(pairs, labels, epochs=50, batch_size=32)

# Predict similarity
similarity = siamese.predict_similarity(emb1, emb2)  # Returns 0-1

# Verify
is_same = siamese.verify(emb1, emb2, threshold=0.6)  # Returns bool

# Save/Load
siamese.save("model.h5")
siamese.load("model.h5")
```

DatasetManager

```python
```

```
manager = DatasetManager(dataset_dir, embeddings_dir)

# Add person
manager.add_person(name, detector, embedder, num_images=10)

# Get embeddings
all_embeddings = manager.get_all_embeddings()  # Returns dict
person_embeddings = manager.get_person_embeddings(name)  # Returns list

# Rebuild
manager.rebuild_embeddings(detector, embedder)
```

AttendanceLogger

python

```python
logger = AttendanceLogger(attendance_file, log_interval=300, save_photos=True)

# Log attendance
logger.log_attendance(name, similarity, face_image)

# Query
today = logger.get_today_attendance()
person = logger.get_person_attendance(name)
all_records = logger.get_all_attendance()

# Stats
stats = logger.get_stats()
logger.print_today_attendance()
```

FaceRecognizer

python

```
recognizer = FaceRecognizer(detector, embedder, siamese, dataset_manager, threshold=0.6)

# Recognize single face
name, similarity, details = recognizer.recognize_face(face_image)

# Recognize from frame
results = recognizer.recognize_from_frame(frame)
# Returns: [{'bbox': (x,y,w,h), 'name': str, 'similarity': float, 'verified': bool}]

# Run camera
recognizer.run_camera(camera_id=0, attendance_logger=logger)

# Update settings
recognizer.update_embeddings()
recognizer.set_threshold(0.7)
```

License

This project is provided as-is for educational and commercial use.

Credits

- **Face Detection**: MTCNN, MediaPipe, OpenCV
- **Face Recognition**: FaceNet, MobileFaceNet
- **Deep Learning**: TensorFlow/Keras
- **UI**: Tkinter

Support

For issues or questions:

1. Check troubleshooting section
2. Review configuration guide
3. Ensure all dependencies installed
4. Verify camera permissions
5. Check Python version compatibility (3.8+)

Enjoy your face recognition attendance system! 🎯