Models used Face recognition Project

# 1.MTCNN

MTCNN stands for Multi-task Cascaded Convolutional Neural Network.

The core idea behind MTCNN is to perform coarse-to-fine face detection using a cascade of three CNNs — P-Net, R-Net, and O-Net — where each stage progressively refines face candidates while also predicting 5 facial landmarks: the positions of the eyes, nose, and corners of the mouth. This architecture enables it to robustly detect faces under various scales, angles, and lighting conditions.

It is a deep learning model used for:  
(i)Face detection (finding faces in images)  
(ii)Facial landmark detection (eyes, nose, mouth corners, etc.)  
(iii)Face alignment (preparing face images for recognition)

**Architecture: 3-Stage Pipeline**

1. P-Net (Proposal Network)

(i)A shallow CNN that scans the image at multiple scales.  
(ii)Generates a set of candidate windows and their bounding box offsets.  
(ii)Fast, but may produce many false positives.

2. R-Net (Refine Network)

(i)Filters out many false positives from the P-Net output.  
(ii)Refines the bounding boxes.  
(iii)Slightly deeper and more accurate than P-Net.

3. O-Net (Output Network)

(i)Further refines candidates and outputs:  
(ii)Final bounding boxes  
(iii)5 facial landmarks (left eye, right eye, nose, left mouth, right mouth)

(iv)Used for face alignment, which is crucial for recognition accuracy.

Each stage applies non-maximum suppression (NMS) to eliminate overlapping boxes and uses bounding box regression for precise localization.

**Features and Outputs**

MTCNN returns:

(i)Bounding boxes of detected faces.  
(ii)Facial landmarks (5 points).  
(iii)Aligned faces (optionally resized/cropped tensors ready for recognition).  
(iv)Can process single or batch images.  
(v)Option to select detection thresholds for each stage.

**Limitations**

(i)Not the fastest: slower than newer detectors like RetinaFace, especially on video or large batches.  
(ii)Doesn’t scale well for mobile or edge devices.  
(iii)Accuracy drops on very small or blurry faces.  
(iv)Some false positives in complex backgrounds.

**Loss Functions Used in MTCNN**

MTCNN optimizes the following three losses jointly:

**1. Face Classification Loss (Binary Cross-Entropy Loss)**

(i)Purpose: Classify whether a candidate region is a face or non-face.  
(ii)Loss Type: Binary cross-entropy (log loss).  
(iii)Applies to: All three networks (P-Net, R-Net, O-Net).

Formula:  
 L\_cls= −[y⋅log⁡(p)+(1−y)⋅log⁡(1−p)]

Where:

* y∈{0,1} Ground truth (face or not)
* p : Predicted probability of being a face

**2. Bounding Box Regression Loss (Smooth L1 / MSE Loss)**

(i)Purpose: Refine the bounding box coordinates to better localize the face.  
(ii)Loss Type: Smooth L1 or Mean Squared Error (MSE).  
(iii)Applies to: Samples labeled as faces.  
(iv)Targets: (x,y,w,h)(x, y, w, h)(x,y,w,h) offsets from the anchor box to the ground truth box.

L\_bbox= (i=1 to 4)∑ (t\_i−t\*\_i)\*\*2

Where:

* t\_i : Predicted offset
* t\*\_i ​: Ground truth offset

**3. Facial Landmark Regression Loss (MSE Loss)**

(i)Purpose: Predict the positions of 5 facial landmarks:  
 Left eye, right eye, nose, left mouth corner, right mouth corner.  
(ii)Loss Type: Mean Squared Error (MSE).  
(iii)Applies to: Samples labeled as landmark samples.  
(iv)Each landmark is represented by (x, y) → 10 values total.

L\_landmark=(i=1 to 10)∑ (l\_i−l\*\_i)\*\*2

**Total Loss (Combined Loss Function)**

The final loss used to train MTCNN is a weighted sum of the above three:

L=λ1\*L\_cls+λ2\*L\_bbox+λ3\*L\_landmarkL

(i)The weights λ1,λ2,λ3​ balance the importance of each task.  
(ii)These weights are often set manually during training or fine-tuned experimentally.

# 

# 2.InceptionResNetV1

InceptionResNetV1 is a deep convolutional neural network that combines the architectural principles of:

(i)Inception networks (multi-scale feature extraction)  
(ii)Residual connections (from ResNet, for deeper networks without vanishing gradients)

It was introduced in the paper:

“Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning” by Szegedy et al.

**FaceNet framework:**

(i)InceptionResNetV1 is used as the backbone network to extract high-dimensional feature embeddings from aligned face images.

(ii)These embeddings (512-D) are then used to:  
 Compare faces (verification)  
 Identify faces (recognition)  
 Cluster faces (unsupervised grouping)

The output is not a classification label, but a vector representation of a face in embedding space — where distance = similarity.

**Model Architecture**

(i)Inception Modules:

Each block processes input at multiple filter sizes (e.g., 1x1, 3x3, 5x5) in parallel.  
 Helps the network learn both fine and coarse features.

(ii)Residual Connections:

Adds the input of a layer to its output (like ResNet).  
 Helps training very deep networks without gradient vanishing.

Makes convergence faster and more stable.

(iii)Final Layers:

Global Average Pooling  
 Flatten  
 Dense layer projecting 512 dimensions (embedding)  
 L2 normalization to ensure unit-length vectors

**Key Advantages**

(i)Powerful feature extractor: Excellent at encoding identity-specific information.  
(ii)Transfer learning ready: Pretrained models save time and boost accuracy.  
(iii)Highly accurate: Paired with triplet loss or cosine similarity, achieves SOTA on face benchmarks (LFW, etc.)  
(iv)L2-normalized embeddings: Makes distance-based comparison stable.

**Limitations**

(i)Heavy model: Not suitable for mobile/edge without optimization.  
(ii)Training from scratch is expensive: Requires millions of face images and proper mining of positive/negative pairs.  
(iii)Embedding drift: If embeddings are compared over time, domain shifts (e.g., lighting, age) might affect results.

**Evaluation Metric**

(i)L2 / Cosine Similarity:  
 distance = ||embedding1 - embedding2||₂  
 Lower distance → higher similarity

(ii)Common threshold for match: 0.6 – 1.0, depending on your tuning and dataset.

# 3.Silent Face Anti-Spoofing

Face anti-spoofing (a.k.a. Presentation Attack Detection or PAD) is the task of detecting whether a face is real (live human) or fake (photo, video, mask, replay attack).

It protects face recognition systems from being fooled by printed photos, phone screens, deepfakes, etc.

Silent Face Anti-Spoofing (SFAS) is a robust benchmark and model introduced as part of the CVPR 2020 Chalearn Presentation Attack Detection Challenge.

Key Paper:

(i)"Chalearn Silent Face Anti-Spoofing Challenge", CVPR 2020  
(ii)Led by researchers from Idiap, Tencent, and other groups

"Silent" refers to the passive nature of the model:

(i)It does not require user interaction (no blinking, head movement, etc.)  
(ii)It operates on still images or short videos, just like typical facial verification

Real-world face authentication systems must:

(i) Work with poor lighting, angles, and low-quality video  
(ii) Handle cross-dataset generalization  
(iii) Detect unknown attack types (e.g., 3D masks, deepfakes)

SFAS is designed to tackle these realistic, unconstrained conditions.

**Dataset: CelebA-Spoof**

Silent-FAS was developed alongside CelebA-Spoof, a large-scale, diverse dataset:

(i)625,537 images  
(ii)10,000+ identities  
(iii)Multiple spoof types:

(a)Printed photo

(b)Phone screen replay

(c)Video replays  
 (d)3D masks

(iv)Each image is labeled with:

(a)Real or fake

(b)Type of spoof

**Model Architecture**

The Silent Face Anti-Spoofing model is based on a lightweight convolutional neural network (CNN) architecture, specifically tailored for efficiency and deployment on resource-constrained devices like mobile platforms. The architecture is derived from MobileFaceNet, a lightweight face recognition model, optimized through model pruning to reduce computational complexity. Key aspects of the architecture include:

(i) Two-Branch Architecture:

Main Classification Branch: This branch is responsible for the primary task of classifying an input face as real (live) or fake (spoof). It processes RGB face images (typically resized to 80x80 pixels) to extract spatial features relevant to distinguishing live faces from spoofed ones.

Auxiliary Fourier Spectrum Supervision Branch: This branch leverages the Fourier spectrum of the input image to capture frequency-domain differences between real and fake faces. The Fourier spectrum reflects variations in texture and patterns that are often distinct in spoofed media (e.g., printed photos or screens) due to resampling or re-imaging artifacts.

(ii) MobileFaceNet Backbone:

The model uses a modified version of MobileFaceNet, a lightweight CNN optimized for face-related tasks.

The architecture includes convolutional layers, batch normalization, and activation functions (likely ReLU or variants). Specific configurations, such as the number of bottlenecks, vary by scale (e.g., Scale 1 for 80x80 inputs).

For the Scale 1 (80x80) model, the input is an RGB face image, and the network includes approximately 12 bottleneck layers with a stride of 1, followed by global depth-wise convolution for feature pooling. The output is a binary classification (real vs. fake).

(iii) Multi-Scale Model Fusion:

The model employs a multi-scale approach to enhance robustness. It processes images at different scales:

Original Image: Resized to a fixed size (e.g., 80x80 pixels).

Patches: Cropped face regions expanded by a certain scale (e.g., 1.2x or 1.5x) to include contextual information like background or attack medium borders.

A face detector is used to locate and crop the face region, ensuring the input aligns with the model's requirements Optional

(iv)Output:

The model outputs a confidence score (0 to 1) indicating the likelihood of the face being real (closer to 1) or fake (closer to 0). A user-defined threshold determines the final classification.

**Loss Function**

The loss function is designed to optimize both the main classification task and the auxiliary Fourier spectrum supervision. The key components are:

(i) Binary Cross-Entropy Loss (Main Classification Branch):

The primary loss for the classification branch is binary cross-entropy, which is standard for binary classification tasks (real vs. fake face). It measures the difference between the predicted probability and the ground-truth label (1 for real, 0 for fake).

L\_BCE=−[ylog⁡(y^)+(1−y)log⁡(1−y^)]

where y is the ground-truth label (0 or 1), and y^ ​ is the predicted probability.

(ii) Fourier Spectrum Supervision Loss:

The auxiliary branch uses the Fourier spectrum to capture frequency-domain differences. While the exact loss function for this branch is not explicitly detailed in the sources, it is likely a regression-based loss (e.g., Mean Squared Error or Kullback-Leibler divergence) to align the predicted Fourier spectrum features with expected patterns for real and fake faces.

The Fourier spectrum supervision helps the model learn subtle differences in frequency-domain characteristics, such as texture smoothing or edge degradation in spoofed images, which are less apparent in the spatial domain.

(iii) Combined Loss:

The overall loss function is a weighted combination of the binary cross-entropy loss (for classification) and the Fourier spectrum supervision loss (for auxiliary guidance).

**Evaluation Metrics**

Face anti-spoofing performance is measured using:

(i) HTER (Half Total Error Rate) = (FAR + FRR) / 2  
 (ii) APCER (Attack Presentation Classification Error Rate): Spoof classified as real  
 (iii) BPCER (Bona Fide Presentation Classification Error Rate): Real classified as spoof  
 (iv) ACER (Average Classification Error Rate) = (APCER + BPCER) / 2  
 (v) AUC, EER: Common binary classification metrics

**Strengths of SFAS**

(i)Designed for realistic spoofing scenarios  
(ii)Encourages cross-dataset generalization  
(iii)Supports multi-modal fusion (RGB, depth, IR, etc.)  
(iv)High accuracy under challenging conditions  
(v)Scalable and efficient (can work in real-time)

**Limitations**

(i)May struggle with novel or unseen spoof types if not trained robustly  
(ii)Performance can drop in extreme lighting or occlusions  
(iii)Training needs diverse and balanced spoof data  
(iv)Real-time inference with heavy backbones (e.g., ResNet-50) requires optimization