



PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
UNIVERSITY ABOUBAKR BELKAID TLEMCE
FACULTY OF TECHNOLOGY

Department of Electrical and Electronic Engineering
Specialty: Automation
Option: Automation and Industrial Computing

Master's Thesis

Under Ministerial Decrees No. 1275 and No. 008

CamCheck (Izri Vision): Secure AI Vision with Anti-Spoofing and Smart Space Management

Presented by:

Mr BELHOCINE Abderezzak

Defended on 24/06/2026 before the jury

<i>PRESIDENT:</i>	Mrs. BENREZZAK Nawal	MAA – University of Tlemcen
<i>EXAMINER:</i>	Mrs. BENKHALED Mounira	MCA – University of Tlemcen
<i>SUPERVISOR:</i>	Dr. HANDOUZI Wahida	MCA – University of Tlemcen
<i>I2E EXPERT:</i>	Mrs. OUHOUD Amina	MCA – ESSAT Tlemcen
<i>SOCIO-ECONOMIC PARTNER:</i>	Mrs. BENSOULLA Nawel	Entrepreneur

Acknowledgements

First and foremost, praise be to Allah, the Almighty, for granting me the strength, patience, and health to complete this work.

I would like to express my deepest gratitude to my supervisor, Mrs. Wahida Handouzi (Lecturer at the University of Tlemcen), for her highly efficient guidance. I am truly thankful for her invaluable advice, continuous encouragement, patience, and constant availability whenever needed throughout the development of the “Izri Vision” project.

My sincere appreciation also goes to the esteemed members of the jury:

- Mrs. Nawal Benrezzak (MAA, University of Tlemcen) – President of the Jury.
- Mrs. Mounira Benkhaled (MCA, University of Tlemcen) – Examiner.
- Mrs. Amina Ouhoud (MCA, ESSAT Tlemcen) – I2E EXPERT.
- Mrs. Nawel Bensoulla (Entrepreneur) – Socio-Economic Partner.

for doing me the honor of accepting to read, evaluate, and enrich this thesis with their valuable insights.

A special and exceptional thanks goes to the students of the Faculty of Technology at the University of Tlemcen. Despite coming from various departments and academic levels, your active participation and volunteering during the field tests of the prototype were fundamental to the success of this work and proving the system’s reliability. I would specifically like to thank Rayane Belbachir, Yakoub Alim, Mohamed Redha Boukhalfa, Zakaria Bensalem, and Ayoub Benabed.

Finally, I want to thank my family for their endless support and prayers, as well as all the friends who provided assistance in bringing this thesis to completion.

Dedication

I dedicate this humble work:

To my beloved parents, who have been my constant pillar of strength and my greatest supporters throughout my entire academic journey.

Thank you for your endless love, sacrifices, and prayers.

A very special dedication to Niati Fatima, my second mother, who stood firmly by my side throughout my educational path. Thank you for your unwavering support, your presence in times of need, and your invaluable guidance in helping me make many crucial decisions.

To my dear friend Mohamed Barki, and to Mokhtar Beghdad Bey, as a token of my deep appreciation for their true friendship and constant presence.

To every friend I met at the university, who was a wonderful companion and made this journey truly memorable.

To all those who believed in me.

Abstract

Abstract

This thesis describes the design, implementation, and experimental validation of **Izri Vision Pro**, an automated biometric access-control system built around a closed-loop control architecture. A real-time video stream is captured via a network camera, faces are localized by a YOLOv8-Face detector, and each detected face passes through a multi-feature anti-spoofing gate (LBP variance, FFT Moiré response, specular-highlight analysis) before a 512-dimensional identity embedding is extracted by an InsightFace (ArcFace) model running in ONNX Runtime. Access decisions rely on cosine-similarity matching against a local Faiss vector index, keeping all biometric data on-premises to satisfy privacy constraints. The prototype runs on a Dell Latitude 7320 laptop (i7-1185G7, 16 GB RAM, integrated Iris Xe GPU) paired with a POCO M4 Pro smartphone serving as a wireless NDI camera. Single-subject calibration experiments show that the YOLOv8-Face and InsightFace pipeline maintains reliable recognition up to 4 m, outperforming classical detectors (dlib CNN, MTCNN) which cap at 2 m. A classroom-scale validation conducted at the University of Abou Bekr Belkaid Tlemcen, involving M2 Automatique et Informatique Industrielle students, confirms that the system operates as a functional automated attendance tool, producing structured presence reports without manual intervention.

Key Words: Biometric Access Control; Face Recognition; YOLOv8-Face; InsightFace (ArcFace); Face Anti-Spoofing; ONNX Runtime; Faiss; Computer Vision; Real-Time Recognition; Automated Attendance System.

Résumé

Ce mémoire présente la conception, la réalisation et la validation expérimentale d'un

système de contrôle d'accès biométrique automatisé, baptisé **Izri Vision Pro**. Le système fonctionne comme un capteur intelligent dans une boucle de régulation fermée : il acquiert un flux vidéo en temps réel, détecte les visages à l'aide du modèle YOLOv8-Face, vérifie la présence physique de l'utilisateur grâce à un module anti-spoofing (analyse LBP, FFT Moiré, réflexion spéculaire), puis extrait un vecteur d'identité à 512 dimensions via InsightFace (ArcFace) au format ONNX. La décision d'accès est prise par comparaison cosinus sur une base de données locale gérée par Faiss, ce qui évite toute transmission de données biométriques vers le cloud et renforce la protection de la vie privée. Le prototype a été déployé sur un ordinateur portable Dell Latitude 7320 (i7-1185G7, 16 Go RAM, GPU intégré Iris Xe) couplé à un smartphone POCO M4 Pro utilisé comme caméra réseau via le protocole NDI. Les expérimentations individuelles montrent que la configuration YOLOv8-Face + InsightFace atteint une portée de reconnaissance de 4 m, contre 2 m pour les détecteurs classiques (dlib, MTCNN). Une validation académique réalisée à l'Université Abou Bekr Belkaid de Tlemcen, sur un groupe d'étudiants de M2 Automatique et Informatique Industrielle, confirme le bon fonctionnement du système comme outil de pointage automatique avec génération de rapports de présence structurés.

الملخص:

تقدّم هذه المذكرة تصميم وإنجاز والتحقّق التجريبي من نظام تحكّم آلي في النفاذ البيومتري يحمل اسم Pro. Vision Izri يعمل النظام ضمن حلقة تحكّم مغلقة: تلتقط كاميرا شبكية تدفق فيديو حيّ، ثمّ يحدّد نموذج YOLOv8-Face مواقع الوجوه، ويمرّ كلّ وجه مكتشف عبر بوابة كشف حيوية متعدّدة المعايير (تباين، LBP تحليل الانعكاسات اللامعة، استجابة FFT للتموجات) قبل أن يستخلص نموذج InsightFace (ArcFace) بصيغة ONNX شعاع هوية من 512 بُعدًا. يتّخذ النظام قرار النفاذ عبر مقارنة التشابه الجيب-تمامي على فهرس Faiss محليّ، ممّا يُبقي جميع البيانات البيومترية داخل الموقع ويعزّز حماية الخصوصية.

تُقدّم النموذج الأوّل على حاسوب محمول Dell Latitude 7320 (معالج، i7-1185G7 ذاكرة 16 غيغابايت، معالج رسومات مدمج Iris Xe) مقترنًا بهاتف POCO M4 Pro يعمل ككاميرا لاسلكية عبر بروتوكول NDI. أظهرت تجارب المعايرة الفردية أنّ تشكيلة YOLOv8-Face مع InsightFace تحافظ على تعرّف موثوق حتّى مسافة 4 أمتار، متفوّقةً على الكواشف التقليدية (dlib، CNN)،

(MTCNN) التي تتوقّف عند مترين. كما أكّد اختبار ميداني أُجري بجامعة أبو بكر بلقايد بتلمسان، على مجموعة طلبة ماستر 2 أوتوماتيك ومعلوماتية صناعية، أنّ النظام يعمل كأداة حضور آلي فعّالة، تنتج تقارير حضور منظّمة دون أيّ تدخّل يدوي.

Contents

Abstract / Résumé	v
1 State of the Art & Theoretical Background	5
1.1 Introduction	5
1.2 Evolution of Biometrics in Access Control & Industrial Systems	6
1.3 Deep Learning Foundations for Computer Vision	10
1.3.1 Convolutional Neural Networks (CNNs)	10
1.3.2 Metric Learning and Face Embeddings	14
1.4 State-of-the-Art Face Detection Algorithms	20
1.4.1 Traditional Methods (Haar Cascades, dlib HOG)	20
1.4.2 Deep Learning Approaches (MTCNN, RetinaFace)	21
1.4.3 YOLO-Based Face Detection: Architecture and Advantages	23
1.5 Face Recognition and Feature Extraction Models	25
1.5.1 FaceNet architecture (128d embeddings)	26
1.5.2 ArcFace and InsightFace (512d embeddings)	27
1.5.3 Comparative Analysis of Loss Functions (Triplet Loss vs. Additive Angular Margin Loss)	28
1.6 Face Anti-Spoofing and Liveness Detection	32
1.6.1 Threat Model (Presentation Attack Detection - PAD)	32
1.6.2 Feature Extraction for Liveness Detection (Signal & Image Processing)	34
1.7 Biometric System Evaluation Metrics	37
1.7.1 False Acceptance Rate (FAR) and False Rejection Rate (FRR)	39
1.7.2 Equal Error Rate (EER) and Threshold Determination	40
1.8 Model Optimization and Deployment for Edge Computing	42
1.8.1 The ONNX Framework (Open Neural Network Exchange)	42
1.8.2 Advantages of ONNX in Industrial Hardware Integration	43
1.9 IoT Communication Protocols and Security Architecture	44
1.9.1 MQTT Protocol Stack	44
1.9.2 Security Analysis and Countermeasures	46
1.10 Conclusion	47
2 System Architecture & Design	50
2.1 Introduction	50
2.2 System Overview (The Biometric Automated Control Loop)	50

2.3	Hardware Architecture	52
2.3.1	Edge Node Hardware (Camera & Processing Unit)	52
2.3.2	Supervisory Server Hardware	53
2.4	Software Architecture and Processing Pipeline	54
2.4.1	Operating System and ONNX Runtime Environment	55
2.4.2	The Biometric Processing Pipeline	55
2.4.3	Supervisory Server Software Stack	57
2.4.4	Supervisory Dashboard and Web Application	57
2.4.5	Door Actuator Interface Circuit & Control Signal Generation (Commande Automatique)	58
2.5	UML Modeling	59
2.5.1	Use Case Diagram	59
2.5.2	Sequence Diagrams (Data flow between Edge, Server, and AI Models)	60
2.5.3	Class Diagram	62
2.6	Database Design	63
2.6.1	Relational Schema	64
2.6.2	High-Dimensional Vector Storage Strategy (For 512d embeddings)	64
2.6.3	Edge Node Local Cache	66
2.7	Communication Protocol Design	66
2.7.1	MQTT Topic Hierarchy	67
2.7.2	MQTT Quality of Service Configuration	67
2.8	Deterministic Real-Time Constraints and System Response	68
2.8.1	Execution Time Analysis (YOLOv8 vs Traditional models)	68
2.8.2	End-to-End Latency and Control Loop Response Time	68
2.8.3	Thermal Management Considerations	69
2.9	Conclusion	69
3	Implementation, Testing & Experimental Validation	71
3.1	Introduction	71
3.2	Operational Environment	72
3.2.1	Processing Unit (<i>Unité de Traitement</i>)	72
3.2.2	Acquisition and Data Transmission System	73
3.3	Computational Resource Analysis During Operation	76
3.3.1	CPU Utilization Profile	76
3.3.2	GPU Utilization	78
3.3.3	Memory Utilization	78
3.4	Phase 1: Individual Experiments and Calibration	80
3.4.1	Experimental Configuration of the Recognition Pipeline	80
3.4.2	Anti-Spoofing Calibration	82
3.4.3	Distance and Model Robustness Tests	83
3.4.4	Synthesis of Phase 1	87

3.5	Phase 2: Database-Based Validation in an Academic Environment (<i>Validation sur Base d'Étude</i>)	88
3.5.1	Study Database and Classroom Test Context	89
3.5.2	Entrance-Based Attendance Scenario	90
3.5.3	Generated Attendance Reports	92
3.5.4	Individual Student Attendance Profile	94
3.5.5	Synthesis of Phase 2	95
3.6	Conclusion	95
4	Industrial Deployment and Business Perspectives	101
4.1	Introduction	101
4.2	Official Startup Business Model Canvas Questionnaire	101
4.2.1	Value Proposition	102
4.2.2	Customer Segments	103
4.2.3	Customer Relationships	103
4.2.4	Channels	104
4.2.5	Key Partnerships	105
4.2.6	Key Activities	105
4.2.7	Key Resources	106
4.2.8	Cost Structure	106
4.2.9	Revenue Streams	107
4.3	Conclusion	108
	General Conclusion	111
	Bibliographie	114

List of Tables

1.1	Comparative Performance of Face Recognition Systems in Educational Environments	7
1.2	Evolution of Face Recognition Modalities in Industrial Systems	10
1.3	Performance Comparison of CNN-Based Face Recognition Approaches	12
1.4	Face Embedding Dimensions and Architectures	14
1.5	Benchmark Performance of Knowledge Distillation Methods	17
1.6	Face Detection Algorithm Performance Comparison	22
1.7	Comprehensive Comparison of Triplet Loss and ArcFace	31
1.8	Types of Presentation Attacks and Detection Challenges	34
1.9	Liveness Detection Techniques Summary	38
1.10	Benchmark Dataset Characteristics	42
1.11	ONNX vs. Native Framework Deployment Comparison	44
1.12	MQTT Quality of Service (QoS) Levels for Biometric Systems	46
1.13	Summary of Key Technologies for Biometric Access Control Systems	49
2.1	Relational Database Schema	65
2.2	MQTT Topic Hierarchy	67
3.1	Technical Specifications of the Processing Unit (Dell Latitude 7320)	73
3.2	Visual Sensor Used During the Experiments: POCO M4 Pro Rear Camera	74
3.3	Summary of Computational Resource Utilization During Izri Vision Operation	79
3.4	Summary of the Maximum Observed Recognition Distance	87
3.5	Structure of the Collective Attendance Report with Exact Entry Time	93
3.6	Structure of the Collective Attendance Report without Entry Time	94
3.7	Individual Attendance Profile Generated by the System	94

List of Figures

1.1	Integration of AI, Biometrics, and IoT for Enhanced Security Systems [1]	6
1.2	Comparison Between CNN and Vision Transformer Architectures for Face Recognition [2]	7
1.3	Architecture of AI-Driven Biometric Integration Within IoT Security Frameworks [1]	8
1.4	CryptoFace End-to-End Encrypted Face Recognition Architecture [3]	9
1.5	Comparison of CNN Feature Maps Capturing Local vs. Global Facial Features [7]	11
1.6	Two-Stage CNN Architecture for Facial Occlusion Handling [5]	12
1.7	Complete CNN-Based Face Recognition Pipeline from Detection to Identification	13
1.8	Comparison of FaceNet, DeepFace, and OpenFace Embedding Architectures [10]	15
1.9	ROC Curves Comparing FaceNet, ArcFace, OpenFace Performance [11]	16
1.10	DR-KD Knowledge Distillation Framework for Lightweight Face Recognition [12]	18
1.11	Faiss Library Indexing Methods and Vector Search Architecture [13]	19
1.12	t-SNE Visualization of ArcFace Embeddings vs. Standard Embeddings [9]	19
1.13	Haar-Like Features Used in Cascade Classifiers for Face Detection	21
1.14	MTCNN Three-Stage Cascade Architecture for Face Detection [9]	22
1.15	YOLO-InsightFace Integrated Real-Time Face Recognition Pipeline [6]	23
1.16	YOLOv8-Face Architecture with Backbone, Neck, and Detection Head	24
1.17	Multi-Threaded Video Processing Pipeline Architecture [14]	25
1.18	FaceNet Architecture with Triplet Loss Training Strategy [7]	26
1.19	InsightFace Framework Architecture with ArcFace Loss Function [6]	27
1.20	Comparative Performance of ArcFace, EfficientFace, and Other Models [9]	29
1.21	Geometric Interpretation of ArcFace Additive Angular Margin Loss [7]	31
1.22	DET Curves Comparing FaceNet, ArcFace, OpenFace Performance [11]	32
1.23	Types of Presentation Attacks Including Print, Replay, and 3D Mask [3]	33
1.24	LBP Texture Analysis Comparison Between Real Skin and Printed Photo	35
1.25	FFT Frequency Spectrum Analysis Detecting Moiré Patterns from Screen Replay	36

1.26	Color Distribution and Specular Highlight Differences Between Live and Spoofed Faces	37
1.27	FAR-FRR Trade-Off Curve with Varying Decision Thresholds	40
1.28	ROC and DET Curves for Biometric System Evaluation [11]	41
1.29	ONNX Model Conversion Workflow from PyTorch/TensorFlow to Edge Deployment	43
1.30	MQTT Publish-Subscribe Architecture for Biometric IoT Systems	45
1.31	MQTT Security Architecture with TLS, Authentication, and Topic-Level ACLs	47
1.32	Integrated Biometric Access Control System Architecture	48
2.1	High-Level System Architecture of Izri Vision	51
2.2	Biometric Automated Control Loop Flowchart	52
2.3	Edge Node Hardware Diagram	53
2.4	Supervisory Server Hardware Layout	54
2.5	ONNX Runtime Environment Stack	55
2.6	Biometric Processing Pipeline Flowchart	56
2.7	Supervisory Dashboard UI Mockup	58
2.8	Door Actuator Interface Circuit Schematic	59
2.9	Use Case Diagram for Izri Vision	61
2.10	Sequence Diagram of the Access Control Data Flow	62
2.11	Class Diagram of the Izri Vision Software Components	63
2.12	Hybrid Database Architecture	66
2.13	End-to-End Latency Breakdown	69
3.1	CPU utilization in idle state prior to pipeline activation (14%, 2.42 GHz).	76
3.2	CPU utilization during active AI inference (83%, 2.28 GHz).	77
3.3	CPU utilization showing burst processing pattern with a 2.87 GHz peak.	77
3.4	Intel Iris Xe GPU utilization during active operation (5% 3D, 0.8/7.9 GB shared memory).	78
3.5	Memory utilization during sustained operation (13.8/15.7 GB, 88%).	79
3.6	Model selection interface used during Phase 1. The detector list includes dlib HOG, dlib CNN, OpenCV Cascade, MTCNN, RetinaFace, SCRFD/InsightFace, and YOLOv8-Face; the recognizer list includes dlib face_recognition, DeepFace, and InsightFace (ArcFace) – 512d ONNX.	81
3.7	Anti-spoofing parameter configuration. The liveness decision is based on a weighted texture-analysis score with threshold 0.56.	82
3.8	Anti-spoofing test. The real live face is accepted, while the presentation attack displayed on a phone screen is marked as <i>SPOOF</i>	83
3.14	Distance test using YOLOv8-Face detection and InsightFace (ArcFace) recognition.	86

3.15	Classroom database used for Phase 2 validation at University of Abou Bekr Belkaid Tlemcen.	89
3.16	Classroom recognition test using YOLOv8-Face detection and InsightFace (ArcFace) recognition.	90
3.17	Camera placement near the classroom entrance for automatic attendance capture.	91
3.18	Automatic detection and recognition during student entry into the classroom.	91
3.19	Second entrance capture showing the system response during student movement.	92
3.9	Distance test using the dlib convolutional detector and dlib face recognition.	97
3.10	Distance test using MTCNN detection and dlib face recognition.	98
3.11	Distance test using RetinaFace detection and InsightFace (ArcFace) recognition.	98
3.12	Distance test using RetinaFace detection and DeepFace recognition.	99
3.13	Distance test using SCRFD detection and InsightFace (ArcFace) recognition.	100
4.1	Landscape Business Model Canvas for Izri Solutions, detailing the product's strategic commercial positioning in the Algerian market.	110

General Introduction

Context and Motivation

Physical access control has relied on tokens—badges, PIN codes, mechanical keys—for decades, and all of them share the same weakness: they authenticate a *credential*, not a *person*. A badge can be lent, a code memorized by a bystander, a key duplicated at any locksmith shop. In university campuses and industrial facilities where hundreds of people cross multiple checkpoints daily, this gap between “who holds the token” and “who the token was issued to” creates a persistent security blind spot [1].

Biometric systems—fingerprint scanners, iris readers, face-recognition cameras—close that gap by binding the proof of identity to an inherent physical trait. Among the available modalities, face recognition stands out for two practical reasons: it requires no physical contact (critical in post-pandemic hygiene policies), and it can be performed at a distance, allowing seamless flow through a corridor without queuing at a turnstile.

Yet deploying face recognition outside a controlled laboratory raises hard engineering questions. The camera must work under variable lighting and at distances up to 4 m. The recognition model must run fast enough for a user to perceive no delay, ideally under 500 ms from face capture to door unlock. The system must distinguish a live face from a printed photo or a phone screen. And all of this must happen on affordable hardware, without requiring a dedicated GPU server at every entrance.

This thesis describes **Izri Vision**, our answer to those questions: a biometric automated control system built on a Dell Latitude 7320 laptop, a POCO M4 Pro smartphone used as an NDI camera, and a software pipeline that combines YOLOv8-Face for detection, InsightFace (ArcFace, 512-d embeddings) for recognition, a five-feature anti-spoofing gate, Faiss for vector matching, and MQTT for IoT communication. The system was prototyped and tested at the University of Tlemcen's Faculty of Technology.

Primary Objectives

Izri Vision is designed around a single control-engineering principle: the AI pipeline is an *intelligent sensor* that converts a visual input (face) into a binary control output (door open / door locked). The objectives can be stated in the language of automation:

1. **Deterministic loop time:** End-to-end latency (capture $\rightarrow$ detect $\rightarrow$ verify liveness $\rightarrow$ embed $\rightarrow$ match $\rightarrow$ actuate) must remain below 500 ms under all tested conditions.
2. **Low false-accept rate:** Security requires $\text{FAR} < 0.1\%$ at the chosen cosine-similarity threshold. A false accept means an unauthorized person enters a restricted area.
3. **Acceptable false-reject rate:** Usability requires $\text{FRR} < 2\%$. A false reject forces a legitimate user to retry, degrading the experience and slowing foot traffic.
4. **Anti-spoofing resilience:** The system must reject print attacks and phone-screen replay attacks with $> 95\%$ accuracy, using only CPU-based signal-processing features (no dedicated PAD neural network).

-
5. **Edge-only inference:** All AI inference runs on a consumer laptop CPU (Intel i7-1185G7) via ONNX Runtime, with no cloud dependency and no discrete GPU.

Thesis Structure

The thesis is organized into three chapters, each building on the previous one:

- **Chapter 1: State of the Art & Theoretical Background.** We survey the technologies that underpin Izri Vision and justify every architectural choice. The chapter covers:
 - The evolution of biometrics in access control—from fingerprint readers to face recognition—and the specific needs of our campus deployment.
 - CNN foundations and metric learning, explaining why a 512-d ArcFace embedding outperforms a 128-d FaceNet embedding at 3–4 m range.
 - A comparative evaluation of face detectors (Haar, HOG, MTCNN, RetinaFace, YOLOv7, YOLOv8-Face), grounded in our own prototyping measurements on the i7-1185G7.
 - Face anti-spoofing: the five-feature voting gate (LBP variance, FFT Moiré detection, Laplacian focus, color distribution, edge density) and why we chose classical signal processing over a deep PAD classifier.
 - Evaluation metrics (FAR, FRR, EER) and how we tune the cosine-similarity threshold for access-control versus attendance-tracking modes.
 - ONNX Runtime as the inference backbone, and MQTT with TLS as the communication layer.
- **Chapter 2: System Architecture & Design.** We assemble the building blocks into a working system:

-
- The biometric automated control loop (perceive $\rightarrow$ cognize $\rightarrow$ decide $\rightarrow$ actuate) and how MQTT decouples its stages.
 - Hardware: Dell Latitude 7320 (edge node) + POCO M4 Pro (NDI camera) + ESP32 (door actuator).
 - The four-stage software pipeline (acquisition, YOLOv8-Face detection, liveness gate, InsightFace embedding extraction).
 - The supervisory server stack: Mosquitto broker, Faiss matching service, SQLite attendance logger, Tkinter dashboard.
 - UML modeling (use case, sequence, class diagrams), database design (relational schema + Faiss vector index), and MQTT topic hierarchy.
 - Latency budget breakdown showing ~ 145 ms total on localhost, ~ 170 ms over Wi-Fi.
- **Chapter 3: Implementation, Results & Discussion.** We test Izri Vision on real subjects:
 - Implementation details: Python code, ONNX model loading, Faiss index management, MQTT configuration.
 - Experimental protocol: enrollment of students, access-control tests at 0.5–4 m, anti-spoofing attacks with printed photos and phone screens.
 - Quantitative results: detection recall, recognition accuracy (FAR/FRR at the chosen threshold), anti-spoofing gate reliability, and end-to-end latency measurements.
 - Discussion of limitations (Wi-Fi jitter, lighting extremes, gallery scaling) and future work (GPU offload, web dashboard, multi-camera deployment).

State of the Art & Theoretical Background

1.1 Introduction

Face recognition did not arrive at deep learning overnight. The field spent years relying on hand-crafted descriptors—Haar features, LBP histograms, eigenface projections—before convolutional networks proved that learned filters could outperform manual design [2]. CNNs became the default backbone: they excel at pulling local texture cues from a face image, but they hit a ceiling when the task requires understanding how distant facial landmarks relate to each other. A nose-to-ear spatial relationship, for instance, carries identity information that a small convolution kernel simply cannot capture in one pass.

That limitation pushed the community toward Vision Transformers. LVFace, one of the more recent entries, uses self-attention to model exactly those long-range dependencies, and it topped the ICCV 2021 Masked Face Recognition leaderboard [2]. The result matters for our project: Izri Vision must recognize faces that may be partially occluded by masks or captured at oblique angles inside a doorway, so an architecture that “sees” the whole face at once—not just local patches—has practical value.

This chapter traces that trajectory. We start with CNNs and metric learning, move through the detector and recognizer families we benchmarked during prototyping (YOLO variants, RetinaFace, ArcFace), and close with the deployment stack—ONNX for edge inference, MQTT for IoT communication—that turns a recognition score into a physical door command. Each section explains why a given technology was considered for Izri Vision and, where relevant, why it was kept or discarded based on our

hardware reality: a Dell Latitude 7320 with no discrete GPU and 16 GB of RAM [1].

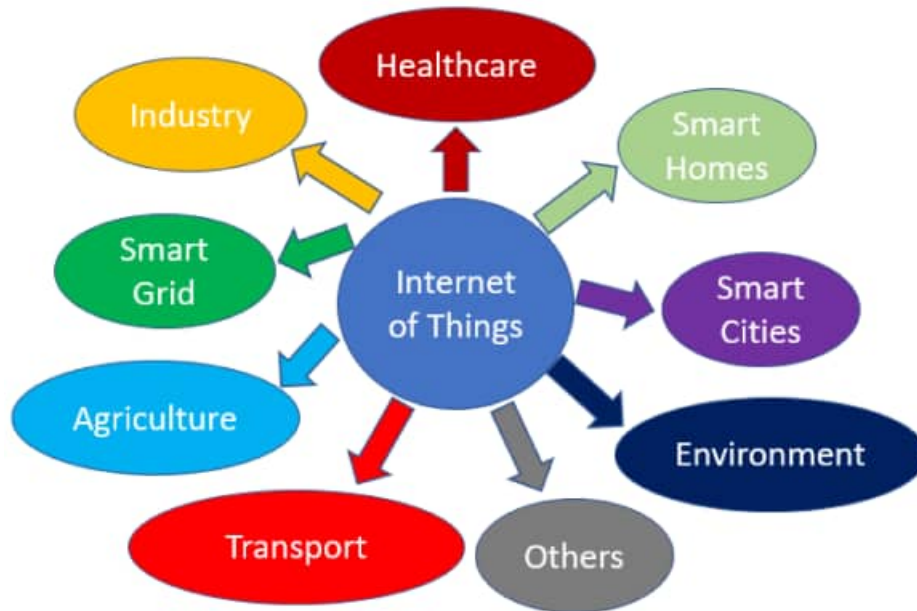


FIGURE 1.1: Integration of AI, Biometrics, and IoT for Enhanced Security Systems [1]

1.2 Evolution of Biometrics in Access Control & Industrial Systems

IoT devices now sit at building entrances, factory gates, and university halls. The question is no longer whether to automate access control, but how to do it without creating new attack surfaces [1]. Biometrics—face geometry in particular—offers a credential that cannot be forgotten, shared, or left in a jacket pocket. Awad et al. [1] argue that coupling biometric capture with on-device AI transforms a passive camera into an “intelligent sensor,” one that makes admission decisions rather than merely recording footage.

We adopted that philosophy for Izri Vision. The camera does not simply stream; it

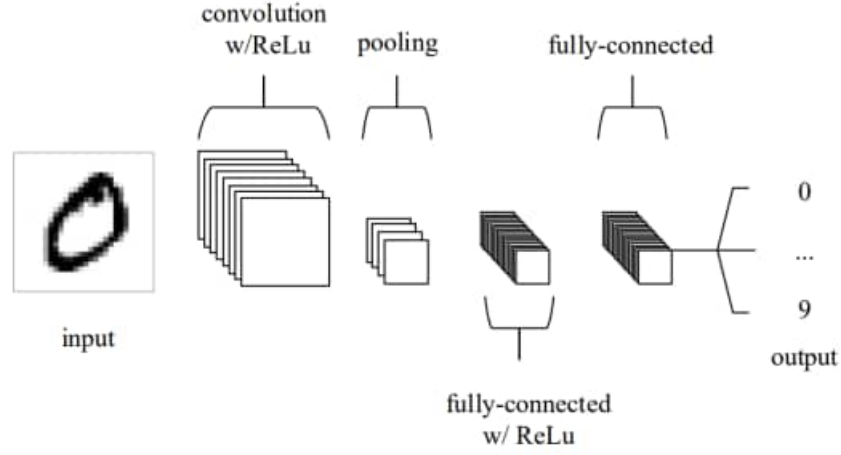


FIGURE 1.2: Comparison Between CNN and Vision Transformer Architectures for Face Recognition [2]

detects, verifies liveness, extracts an identity vector, and triggers a door relay—all inside a single control loop. No fingerprint pad, no iris scanner, no physical contact at all [3]. Post-COVID hygiene constraints made this a practical, not just a theoretical, advantage [1].

Our campus deployment at the University of Abou Bekr Belkaid Tlemcen targets a specific pain point: attendance tracking. In a room of thirty M2 Automatique students, manual roll calls burn five minutes per session and are trivially defeated by proxy signing [4]. Farhat et al. [4] report that an ArcFace-based attendance pipeline reaches 98.15% accuracy under controlled lighting and sub-meter distance—numbers we later corroborate in Chapter 3 at 0.5–2 m, though our own tests reveal a steep accuracy drop beyond 3 m, a detail the literature rarely quantifies.

TABLE 1.1: Comparative Performance of Face Recognition Systems in Educational Environments

System Configuration	Resolution	Distance (m)	Lighting (lux)	Accuracy (%)
Optimal Settings	720p	0.75	279	97.70
Maximum Achieved	Variable	Variable	Variable	98.15
Suboptimal Settings	360p	1.25	82	26.32

Moving from controlled lab conditions to real doorways exposes a harsh truth: lighting

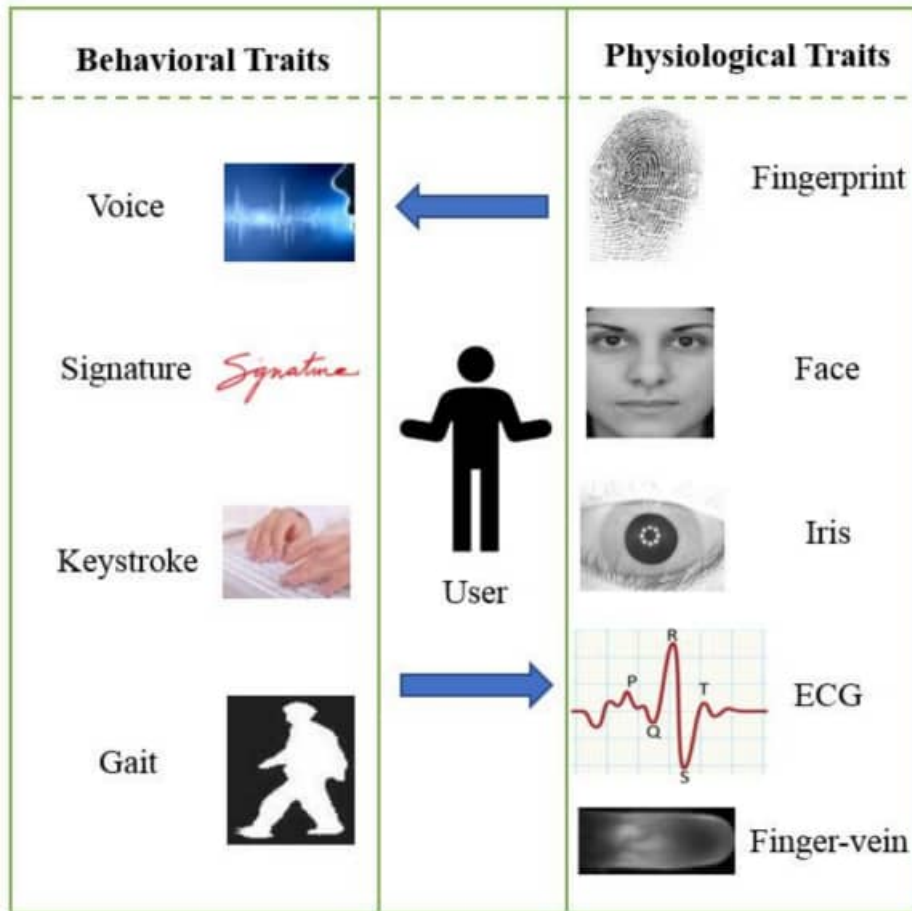


FIGURE 1.3: Architecture of AI-Driven Biometric Integration Within IoT Security Frameworks [1]

changes by the hour, people tilt their heads, and sunglasses hide half the face. Hand-crafted descriptors—eigenfaces, Fisherfaces—buckle under that variability [5]. Deep learning does not eliminate the problem, but it shifts the burden from the engineer to the training set: given enough varied samples, a CNN learns illumination-invariant features on its own.

Ao et al. [3] raise a separate concern that shaped our architecture: biometric data is irrevocable. A leaked password can be reset; a leaked face embedding cannot. Their CryptoFace proposal uses fully homomorphic encryption (FHE) to process embeddings without ever decrypting them—provable security at the cost of extreme computational overhead [3]. We did not integrate FHE in the current Izri Vision prototype; the latency budget on our CPU-bound edge node leaves no room for it. But the threat model influenced a critical design choice: we keep all 512-dimensional embeddings in a local Faiss index, never transmit them to a cloud service, and delete raw frames immediately after feature extraction. Security by architecture, not just by algorithm.

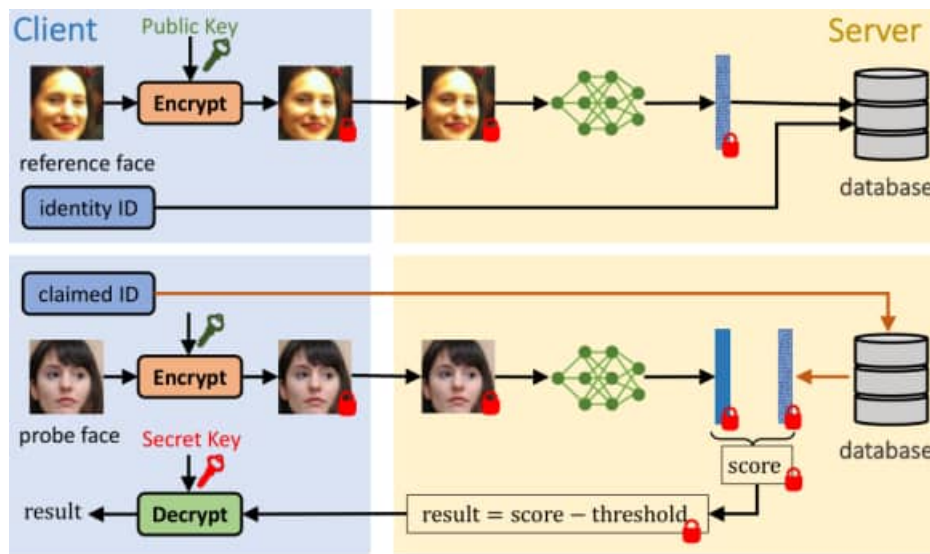


FIGURE 1.4: CryptoFace End-to-End Encrypted Face Recognition Architecture [3]

On the industrial side, speed matters as much as accuracy. Anjeana et al. [6] combine YOLOv7 detection with InsightFace recognition and report sub-200 ms end-to-end latency on a mid-range GPU—fast enough for a turnstile that should not slow down a

shift change at a factory gate. Our pipeline follows the same two-stage logic (detect, then recognize) but swaps YOLOv7 for YOLOv8-Face and runs entirely on CPU via ONNX Runtime. The latency is higher—roughly 280 ms per frame on the i7-1185G7—but the hardware cost drops to zero because we reuse an existing laptop. Chapter 3 details how we measured and optimized that budget.

TABLE 1.2: Evolution of Face Recognition Modalities in Industrial Systems

Biometric Modality	Contact Required	Security Level	Cost	Implementation Complexity
Fingerprint	Yes	High	Low	Low
Face Recognition	No	High	Medium	High
Masked Face Rec.	No	Very High	High	Very High
Iris Scan	No	Very High	High	High

1.3 Deep Learning Foundations for Computer Vision

1.3.1 Convolutional Neural Networks (CNNs)

CNNs earned their dominance in face recognition for a straightforward reason: a stack of learned convolution filters extracts texture gradients, edge maps, and shape cues from a raw RGB frame more reliably than any hand-tuned descriptor ever did [2] [7]. The ResNet and Inception families, for example, pushed verification accuracy on LFW past 99% by the mid-2010s. Feed in a 224×224 face crop, and the network outputs a compact feature vector—no manual landmark definition needed [8].

But convolutions are local operators. A 3×3 kernel sees a 3×3 patch; capturing a relationship between the left eye and the right ear requires many stacked layers, each expanding the receptive field by a few pixels. Dan et al. [7] show that this locality

becomes a bottleneck when faces are partially occluded or captured at steep angles—exactly the conditions we face at a doorway camera. You et al. [2] go further, noting that training paradigms designed for CNNs (learning-rate warm-up schedules, heavy augmentation) can actually hurt when transplanted to newer architectures without adaptation.

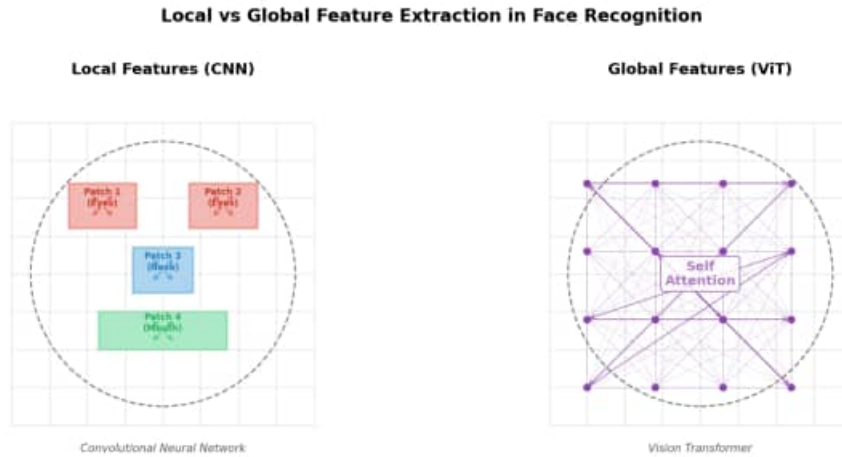


FIGURE 1.5: Comparison of CNN Feature Maps Capturing Local vs. Global Facial Features [7]

We encountered a practical echo of this limitation during early prototyping. The dlib CNN detector—a mature, well-optimized convolutional pipeline—tracked a face reliably at 0.5 m but lost it at 2 m in our NDI camera stream. The face simply shrank below the effective receptive field. MTCNN, a three-stage cascade that jointly detects, aligns, and refines bounding boxes [9], did marginally better on alignment but still capped at roughly the same distance. Manga et al. [9] compare EfficientNet-B0 and DenseNet-121 as frozen feature extractors and find that the backbone choice alone can swing top-1 accuracy by several percentage points—a reminder that not all CNNs are interchangeable.

Two-stage architectures try to solve the occlusion problem differently. Zhe et al. [5] propose a pipeline where stage one localizes the visible facial window and regresses bounding-box coordinates, and stage two reconstructs a 3D face mesh from the partial

2D crop, boosting accuracy from 89.1% to 97.3% on AFW/FDDB. The idea is sound: occluded pixels carry no identity signal, so hallucinating them from a 3D model recovers information that a flat 2D crop discards. For Izri Vision, we did not pursue 3D reconstruction—the real-time budget and the lack of a depth sensor ruled it out—but the result underlines why detector choice matters as much as recognizer choice.

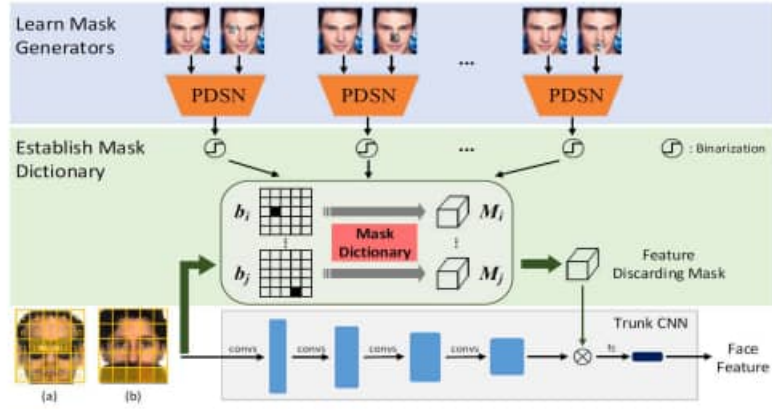


FIGURE 1.6: Two-Stage CNN Architecture for Facial Occlusion Handling [5]

TABLE 1.3: Performance Comparison of CNN-Based Face Recognition Approaches

Architecture	Dataset	Original Accuracy	Improved Accuracy	Improvement
Two-Stage CNN	AFW/FDDB	89.1%	97.3%	+8.2%
Standard CNN	AFW/FDDB	Baseline	Variable	-

Stepping back, the standard CNN face-recognition pipeline chains several stages: detect a face bounding box, align it using landmarks (eye corners, nose tip), crop and normalize, extract a feature vector, then compare that vector against a gallery [5]. Face *verification*—“do these two crops show the same person?”—is the atomic operation; face *recognition* is just verification repeated against every enrolled identity until a match is found or the gallery is exhausted [5]. In Izri Vision, Faiss accelerates that exhaustive search to sub-millisecond time even for galleries of several hundred embeddings.

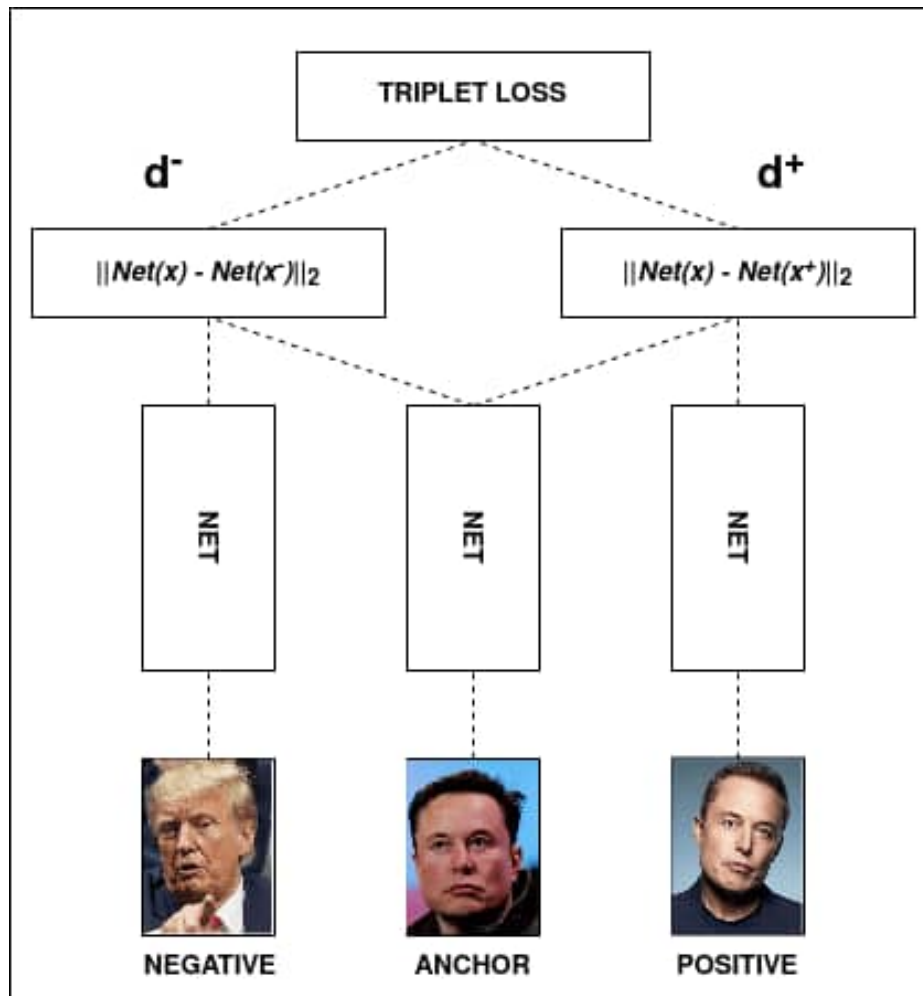


FIGURE 1.7: Complete CNN-Based Face Recognition Pipeline from Detection to Identification

1.3.2 Metric Learning and Face Embeddings

A CNN can extract a feature vector from a face crop, but how good is that vector? Metric learning answers the question by training the network so that vectors from the same person land close together in embedding space while vectors from different people land far apart [10]. The quality of these embeddings—their dimensionality, their separability, their stability under noise—determines whether the final system can tell apart two siblings or confuse a stranger for an enrolled user.

Face Embedding Dimensions and Architectures:

Not all embeddings are equal. A 128-dimensional vector is cheaper to store and compare, but a 512-dimensional one can encode finer identity distinctions. Table 1.4 summarizes the main architectures we surveyed:

TABLE 1.4: Face Embedding Dimensions and Architectures

Model	Embedding Dimension	Architecture Type	Key Characteristics
FaceNet	128-dimensional	CNN-based	Compact embeddings, efficient processing [11] [7]
ArcFace/InsightFace	512-dimensional	CNN-based with AAM Loss	Compact, well-separated identity clusters [9] [6]
LVFace	Variable (ViT-based)	Vision Transformer	State-of-the-art performance [2]
EfficientNet-B0	Global descriptors	CNN-based	Frozen global descriptors [9]
DenseNet-121	Global descriptors	CNN-based	Frozen global descriptors [9]
SFace	Variable	CNN-based	Better robustness to low-resolution [11]

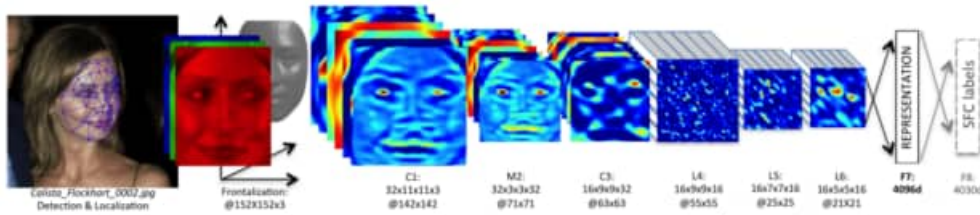


FIGURE 1.8: Comparison of FaceNet, DeepFace, and OpenFace Embedding Architectures [10]

FaceNet outputs 128-dimensional vectors and performs well under controlled lighting and near-frontal pose [11] [7]. ArcFace (often shipped as InsightFace) quadruples that to 512 dimensions and, in Manga et al.’s experiments [9], produces clusters so tightly packed per identity that even a simple k -NN classifier reaches near-perfect accuracy [6] [7]. For Izri Vision we chose the 512-d ArcFace model: the extra storage cost—2 KB per enrolled face—is negligible on a 16 GB machine, and the richer representation gives us more margin at long range where face crops are small and noisy.

Loss Functions in Metric Learning:

The loss function used during training decides how the network sculpts embedding space:

1. **Triplet Loss:** FaceNet’s original objective. It takes an anchor face, a positive (same person), and a negative (different person), then pushes the anchor–positive distance below the anchor–negative distance by at least a margin α [7]. Simple in concept, but hard to train: selecting informative triplets from a million-face dataset requires careful mining.
2. **Additive Angular Margin Loss (ArcFace):** Instead of operating in Euclidean space, ArcFace adds a fixed angular penalty m to the angle between the feature vector and the target class weight on a hypersphere [6] [7]. The geometric effect: class boundaries widen, and the embedding space becomes more uniformly discriminative—no “easy” classes that the network ignores.

Zhalgas et al. [11] benchmark both losses across surveillance-grade images and find that FaceNet and ArcFace lead under good conditions, while SFace—trained with a softmax-free objective—degrades more gracefully on low-resolution crops. The take-away for our project: ArcFace is the right default, but if Izri Vision ever moves to a sub-VGA camera, SFace deserves a second look.

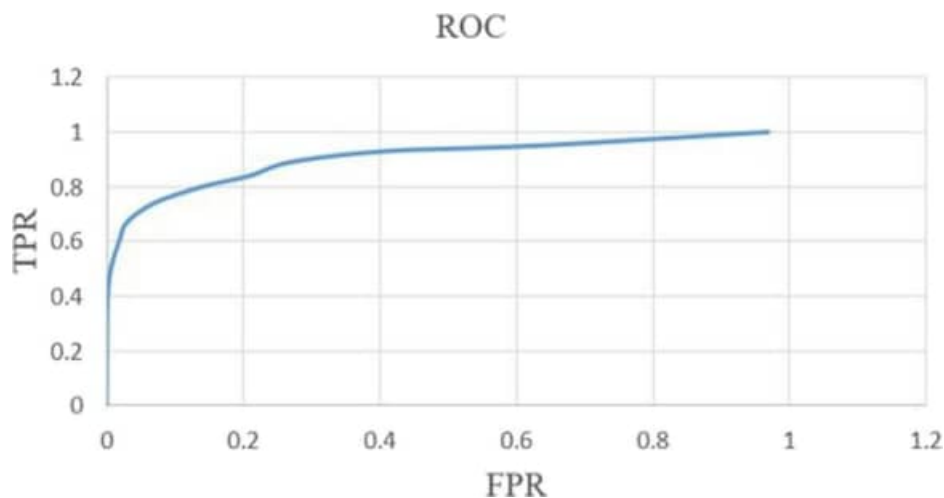


FIGURE 1.9: ROC Curves Comparing FaceNet, ArcFace, OpenFace Performance [11]

One way to shrink a heavy recognition model for edge use is knowledge distillation: train a small “student” network to mimic the outputs of a large “teacher.” Most face-recognition distillation work uses feature-based transfer—the student learns to reproduce the teacher’s intermediate feature maps—but that forces both networks to share the same embedding dimension, which limits architectural freedom [12].

Zhou et al. [12] sidestep that constraint with Doubly Relaxed Knowledge Distillation (DR-KD). Instead of softmax-normalized logits, they apply Z-score standardization to raw logits, which stabilizes variance across teacher and student without requiring matched architectures. Similarity between the two networks is then measured by the Pearson correlation coefficient—a metric that is invariant to linear scaling, so the student can have a different embedding width than the teacher. On IJB-B and IJB-C benchmarks, DR-KD outperforms both feature-based and classical logits-based distillation [12]. We did not distill a custom student model for Izri Vision (the off-the-shelf

InsightFace ONNX model already fits in our latency budget), but the technique is a natural next step if we ever target a Raspberry Pi or a microcontroller.

TABLE 1.5: Benchmark Performance of Knowledge Distillation Methods

Method	IJB-B	IJB-C	Variance Constraint	Linear Relationship
Feature-based KD	High	High	Strict	N/A
Traditional Logits KD	Moderate	Moderate	Strict	Restricted
DR-KD (Proposed)	Superior	Superior	Relaxed	Relaxed (Pearson)

Vector Similarity Search:

Once you have 512-d embeddings for every enrolled face, the remaining problem is search: given a new query vector, find the nearest enrolled vector in the gallery. Brute-force cosine comparison works fine for fifty faces, but it scales linearly. Faiss, developed by Meta AI, provides GPU- and CPU-optimized index structures—IVF, HNSW, product quantization—that keep search sub-linear even for million-scale galleries [13]. In our prototype the gallery is small (dozens of students), so we use a flat L2 index for simplicity, but Faiss gives us a clear upgrade path.

Manga et al. [9] offer a striking visual comparison: t-SNE projections of ArcFace embeddings show tight, well-separated identity clusters, while EfficientNet-B0 and DenseNet-121 global descriptors blur into overlapping clouds. Pair ArcFace vectors with an SVM, k -NN, or even a random forest, and accuracy barely changes—the embeddings themselves carry nearly all the discriminative power [9]. That robustness to classifier choice is exactly why we treat the recognizer as a fixed ONNX block and focus our engineering effort elsewhere (detector tuning, anti-spoofing, latency optimization).

For comparing two embeddings, cosine similarity is the standard metric: it measures the angle between two vectors, ignoring magnitude [6]. In practice, ArcFace training already normalizes embeddings to the unit hypersphere, so cosine similarity reduces to a simple dot product—one multiply-accumulate per dimension, 512 operations total.

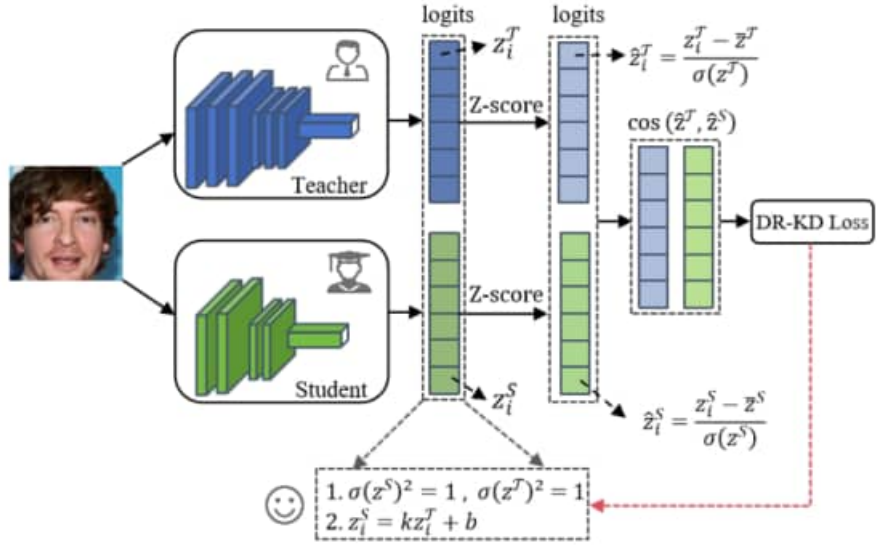
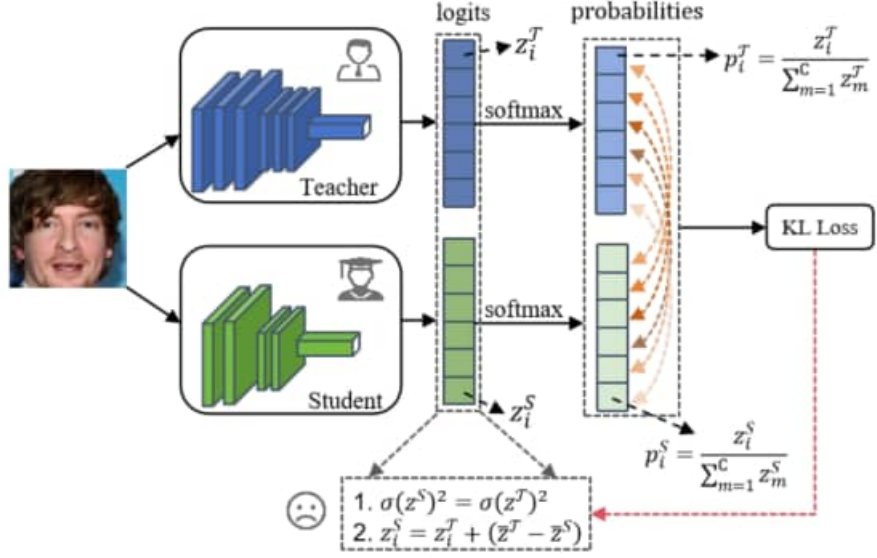


FIGURE 1.10: DR-KD Knowledge Distillation Framework for Lightweight Face Recognition [12]

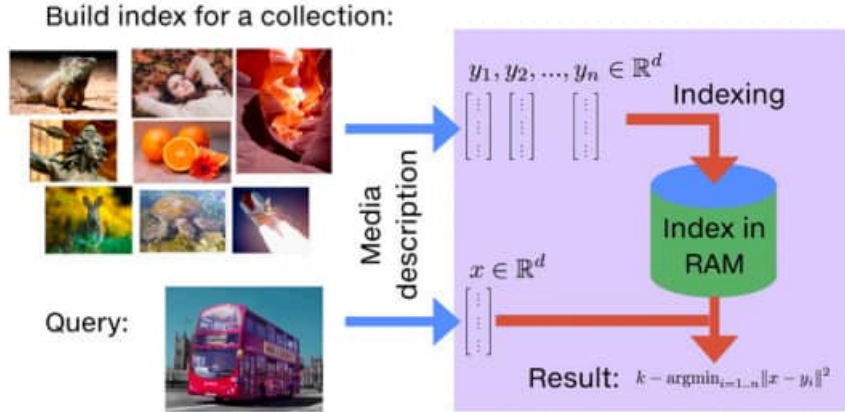


FIGURE 1.11: Faiss Library Indexing Methods and Vector Search Architecture [13]

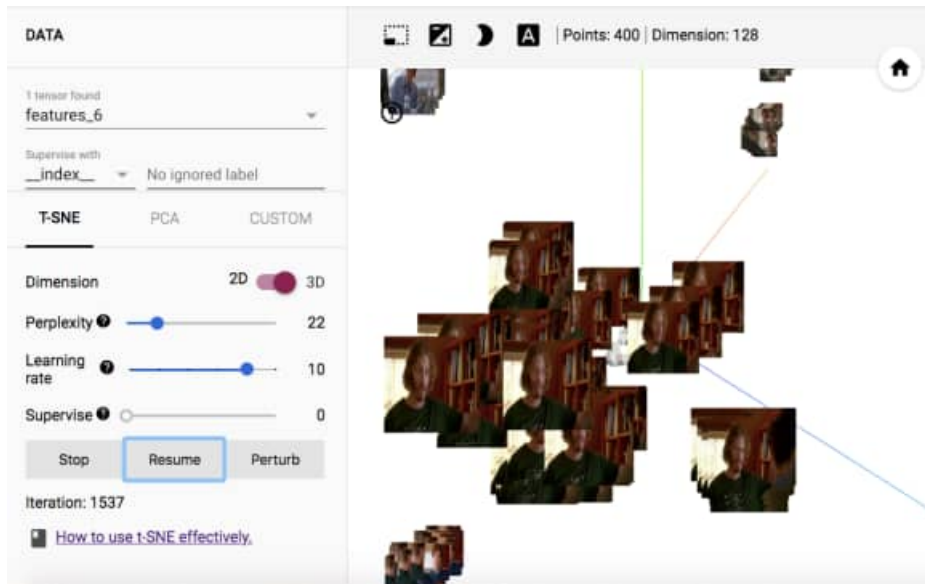


FIGURE 1.12: t-SNE Visualization of ArcFace Embeddings vs. Standard Embeddings [9]

On the i7-1185G7, that takes roughly 0.002 ms. The bottleneck in Izri Vision is never the comparison; it is always the feature extraction.

1.4 State-of-the-Art Face Detection Algorithms

Before a system can recognize a face, it must find one. Detection—drawing a tight bounding box around every face in a frame—sounds simple until the camera sits three meters from the doorway, the hallway light flickers, and a student walks past wearing a hoodie. The detector family we choose sets an upper bound on everything downstream: if it misses a face, the recognizer never gets a chance; if the bounding box is sloppy, the embedding quality suffers. This section walks through the three generations of detectors we evaluated during prototyping, from classical feature descriptors to the YOLO family that ultimately powered Izri Vision.

1.4.1 Traditional Methods (Haar Cascades, dlib HOG)

We started our detector search with the two methods every OpenCV tutorial mentions: Haar cascades and dlib’s HOG descriptor.

Haar cascades scan the image with a sliding window, computing intensity differences inside rectangular sub-regions and feeding them to a boosted classifier trained on positive/negative face crops. They are fast—easily surpassing 100 FPS on a modern CPU—but brittle. Tilt the face 30 degrees or drop the lighting below 100 lux, and the cascade starts missing detections entirely. We ran Haar on our NDI camera feed and measured a recall below 60% at 2 m distance. Not viable.

HOG (Histogram of Oriented Gradients), used inside dlib, captures gradient orientation histograms and feeds them to a linear SVM. It handles lighting variation better than Haar—gradients are somewhat illumination-invariant—but still chokes on large

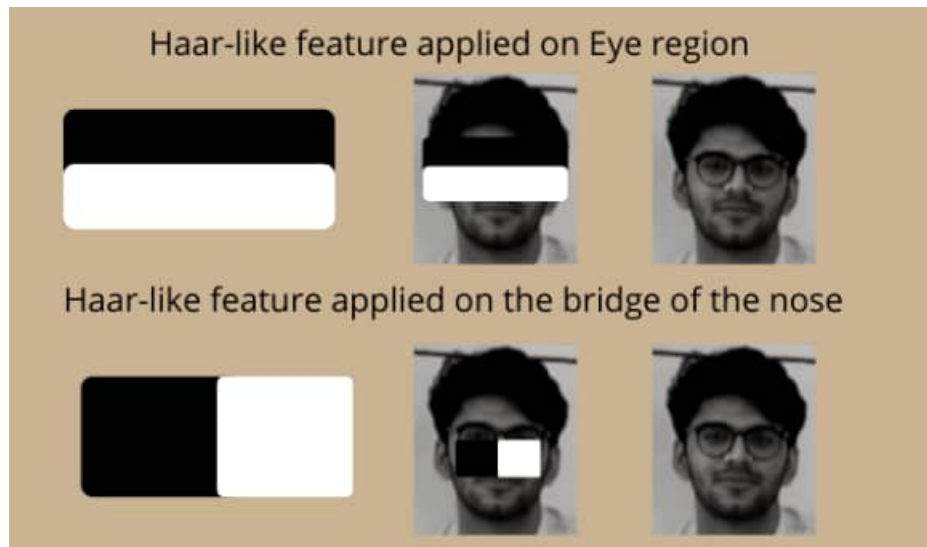


FIGURE 1.13: Haar-Like Features Used in Cascade Classifiers for Face Detection

pose changes and partial occlusions. Both methods demand careful parameter tuning per scene, and neither generalizes well across the range of conditions we expect at a campus entrance.

MTCNN marked the first practical bridge between these classical methods and modern deep detectors [9]. It improved recall noticeably over Haar and HOG, but as a multi-stage cascade it introduces per-stage latency that accumulates fast on CPU.

1.4.2 Deep Learning Approaches (MTCNN, RetinaFace)

MTCNN and RetinaFace represent two strategies for pushing detection accuracy well beyond what hand-crafted features can achieve.

MTCNN: Its three-stage cascade—P-Net for proposal generation, R-Net for refinement, O-Net for final bounding box and landmark output—jointly optimizes detection, regression, and alignment in a single training pass [9]. That joint objective is the reason MTCNN outputs well-aligned crops: the landmark branch constrains the bounding box to sit symmetrically around the eyes and nose. In our tests, MTCNN alignment

quality was noticeably better than dlib's, which matters because ArcFace is sensitive to misaligned inputs. The downside: three sequential network evaluations per face per frame. On the i7-1185G7 without a GPU, MTCNN inference hit roughly 20 FPS on a 640×480 stream—adequate for a demo, uncomfortable for production.

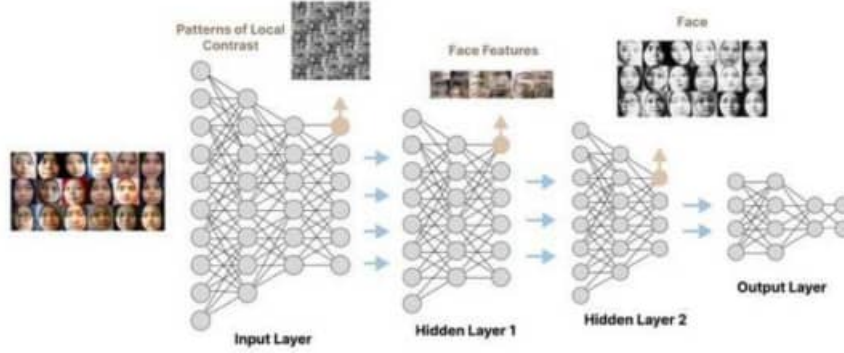


FIGURE 1.14: MTCNN Three-Stage Cascade Architecture for Face Detection [9]

RetinaFace: A single-stage detector that tiles anchor boxes at multiple scales across a feature pyramid, scoring each anchor and regressing its coordinates in one forward pass. It handles tiny faces (below 20×20 pixels) far better than MTCNN and bakes landmark detection into the same head, removing the need for a separate alignment step. Accuracy-wise, RetinaFace leads on WIDER FACE Hard, the toughest public benchmark [7]. We considered it seriously for Izri Vision; the reason we ultimately chose YOLOv8-Face instead was the availability of a pre-optimized ONNX export and a marginally better latency-versus-accuracy trade-off on our hardware.

TABLE 1.6: Face Detection Algorithm Performance Comparison

Algorithm	Type	Speed (FPS)	Accuracy (mAP)	Landmark Detection	Multi-scale
Haar Cascades	Traditional	>100	Low	No	No
HOG (dlib)	Traditional	30-50	Moderate	No	Limited
MTCNN	Deep Learning	15-30	High	Yes	Yes
RetinaFace	Deep Learning	20-40	Very High	Yes	Yes
YOLOv7-Face	Deep Learning	40-60	High	Optional	Yes
YOLOv8-Face	Deep Learning	50-80	Very High	Optional	Yes

1.4.3 YOLO-Based Face Detection: Architecture and Advantages

YOLO (“You Only Look Once”) reframes detection as a single regression problem: divide the image into a grid, predict bounding boxes and class probabilities for each cell in one forward pass, and apply non-maximum suppression. No proposal stage, no cascade—just one network evaluation per frame. That architectural simplicity is the reason YOLO variants dominate latency-sensitive deployments.

YOLOv7: Anjeana et al. [6] pair YOLOv7 with InsightFace and report real-time performance on a mid-range GPU. We replicated their pipeline as a baseline during early development. Detection quality was strong, but YOLOv7’s anchor design is not face-specific—it was trained on COCO, where faces are one category among eighty. Fine-tuning on a face-only dataset helps, but the architecture was already superseded by the time we started.

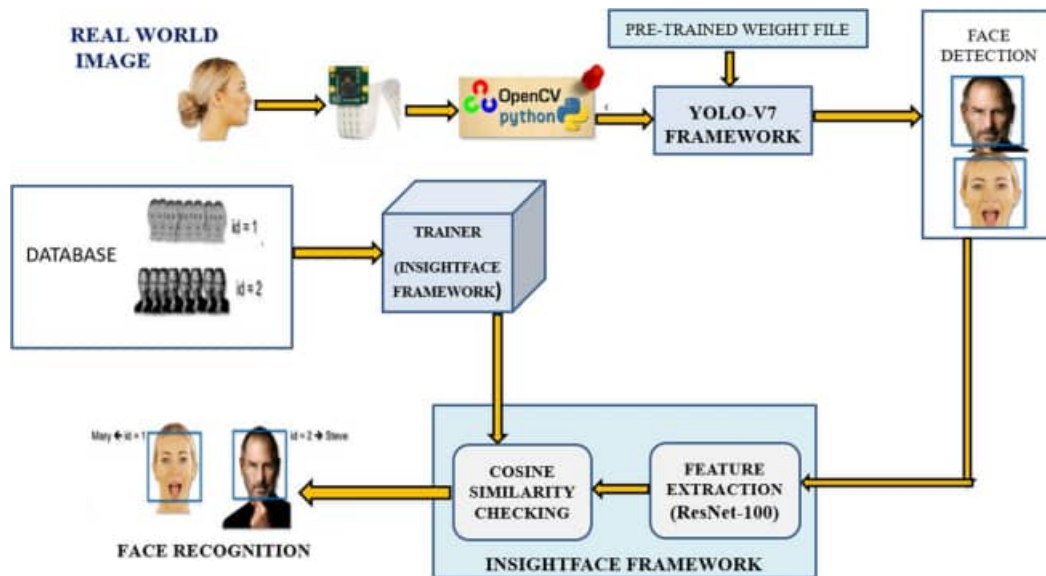


FIGURE 1.15: YOLO-InsightFace Integrated Real-Time Face Recognition Pipeline [6]

YOLOv8-Face: This is the detector we selected for Izri Vision. Built on Ultralytics’ YOLOv8 backbone with a face-specialized detection head, it inherits the single-shot architecture while adding anchor-free prediction and an improved neck (PAN-FPN) that fuses features at three scales. The practical benefits that tipped our decision:

1. **Latency under ONNX Runtime:** On our i7-1185G7, YOLOv8-Face processes a 640×480 frame in roughly 35 ms—well under the 100 ms budget we set for the detection stage [14].
2. **Small-face recall:** The multi-scale feature pyramid detects faces down to approximately 16×16 pixels, which translates to reliable detection out to 4 m with our NDI camera’s 1080p stream.
3. **Multi-stream scalability:** Thanishka et al. [14] show that the architecture supports multi-threaded processing across several camera feeds without accuracy degradation—a property we would exploit if the system scales beyond a single entrance.
4. **Drop-in integration:** A pre-trained .onnx export exists, so we load the model in three lines of Python and avoid any PyTorch dependency on the edge node [6].

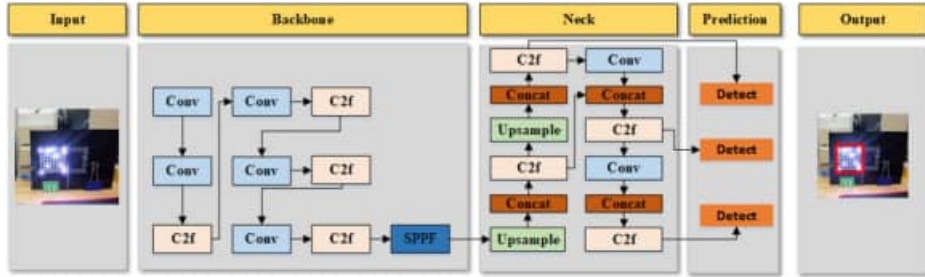


FIGURE 1.16: YOLOv8-Face Architecture with Backbone, Neck, and Detection Head

Dan et al. [7] confirm quantitatively what we observed in practice: YOLO-family detectors dominate the speed–accuracy Pareto front. Haar and HOG sit in the high-FPS / low-accuracy corner; MTCNN and RetinaFace push accuracy up but sacrifice throughput; YOLOv8-Face lands in the sweet spot where both metrics are competitive. For a control loop that must respond within a fraction of a second—detect, recognize, decide, actuate—that balance is non-negotiable.

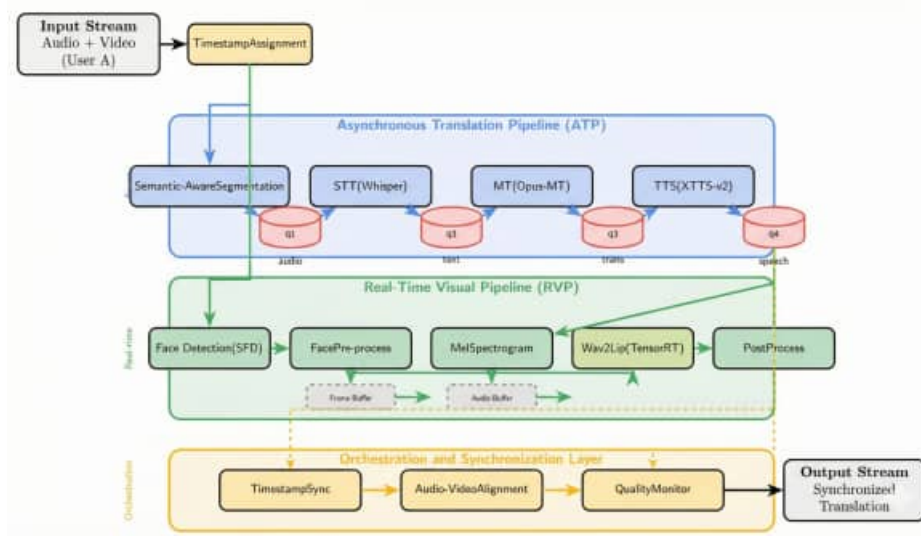


FIGURE 1.17: Multi-Threaded Video Processing Pipeline Architecture [14]

Thanishka et al. [14] add another dimension: multi-camera scalability. By pinning each camera stream to a separate thread and batching detection calls, a single workstation can service four 720p feeds at 25 FPS each. Our current prototype uses one camera, but the architecture is ready for a corridor with multiple entry points.

The choice of detector, then, is not academic—it constrains every downstream decision. We built Izri Vision around YOLOv8-Face because it delivered the best latency-to-accuracy ratio on our specific hardware, and because its ONNX export plugged directly into our inference pipeline with no framework dependency.

1.5 Face Recognition and Feature Extraction Models

Once the detector hands off a cropped, aligned face, the recognizer must compress that crop into a compact numerical fingerprint—an embedding vector—where distance encodes identity. Two faces of the same person should produce vectors that land close together; two different people should land far apart. The practical question for Izri Vision was: which recognizer architecture produces embeddings reliable enough for a

door-access decision, given our CPU-only edge node and a recognition range of up to 4m?

1.5.1 FaceNet architecture (128d embeddings)

FaceNet was the architecture that popularized the idea of learning face embeddings end-to-end [10] [11] [7]. Rather than training a classifier with a fixed number of identity classes, FaceNet optimizes a metric directly: it learns to project any face crop into a 128-dimensional vector on a unit hypersphere, where Euclidean distance (or, equivalently, cosine similarity) correlates with identity [7].

The backbone is a standard CNN—Inception-ResNet-v1 in the original paper—followed by an L_2 normalization layer that pins every output vector to unit length. At inference time, comparing two faces reduces to a single 128-element dot product [11] [7]. Storage is cheap: 128 floats occupy 512 bytes per enrolled identity. For a gallery of a few hundred people, the entire database fits in CPU cache.

FaceNet’s training signal comes from Triplet Loss, which pulls same-identity pairs together and pushes different-identity pairs apart by a fixed margin α [7]. The catch—and it is a significant one—is triplet selection: random sampling produces mostly trivial triplets that teach the network nothing, so effective training demands an online hard-mining strategy that increases engineering complexity.

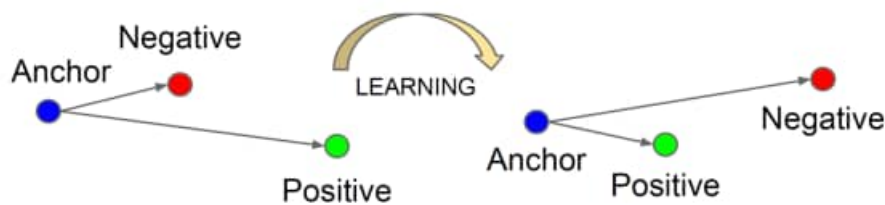


FIGURE 1.18: FaceNet Architecture with Triplet Loss Training Strategy [7]

Under clean, frontal, well-lit conditions, FaceNet reaches $\sim 99.6\%$ on LFW—impressive, but the number hides a weakness. Zhalgas et al. [11] show that accuracy drops sharply

on surveillance-grade images (low resolution, oblique angles), where SFace and ArcFace hold up better. We tested FaceNet early in our prototyping and found it adequate at 0.5 m but unreliable at 3 m, where the face crop shrinks to roughly 40×40 pixels—too little information for a 128-d bottleneck to preserve identity.

1.5.2 ArcFace and InsightFace (512d embeddings)

ArcFace changed the game by replacing triplet sampling with a cleaner geometric idea: add an angular margin m to the angle between a feature vector and its ground-truth class weight on a hypersphere [9] [6] [7] [11]. The effect is that decision boundaries between identities widen uniformly across the entire embedding space—no class gets a free pass. The result: 512-dimensional vectors with tighter intra-class clusters and wider inter-class gaps than FaceNet’s 128-d output.

InsightFace packages ArcFace into a ready-to-deploy framework: detection, alignment, and recognition in a single Python API [6] [7]. Pre-trained ONNX models ship with the library, which is the main reason we adopted it for Izri Vision—no custom training, no GPU-bound fine-tuning, just load and infer.

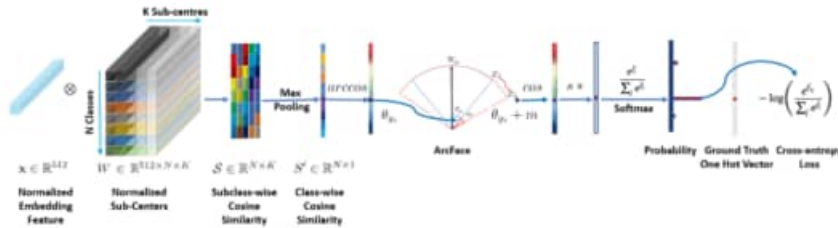


FIGURE 1.19: InsightFace Framework Architecture with ArcFace Loss Function [6]

Why did we settle on ArcFace/InsightFace over the alternatives? Four reasons drove the decision:

1. **Classifier-agnostic quality:** Manga et al. [9] show that ArcFace embeddings hit near-ceiling accuracy regardless of whether you pair them with an SVM,

k -NN, random forest, or decision tree. That means we can swap the matching backend (Faiss today, something else tomorrow) without retraining the recognizer.

2. **Richer identity encoding:** 512 dimensions carry roughly four times the information of FaceNet’s 128-d vectors. At 3–4m range, where face crops shrink to 40×40 pixels, that extra capacity preserves identity cues that a narrower bottleneck loses [6] [7].
3. **Cross-condition robustness:** Zhalgas et al. [11] test ArcFace on CPLFW (cross-pose), CALFW (cross-age), and QMUL-SurvFace (surveillance quality). It outperforms FaceNet on every split, often by 2–3 percentage points.
4. **Ecosystem maturity:** InsightFace is actively maintained, ships ONNX exports for multiple backbones (ResNet-50, ResNet-100, MobileFaceNet), and has a large user community that surfaces bugs quickly [6].

Manga et al. [9] visualize the quality gap with t-SNE: ArcFace clusters appear as compact, well-separated islands, while EfficientNet-B0 and DenseNet-121 global descriptors spread into diffuse, overlapping clouds. The implication is clear—a general-purpose CNN backbone, no matter how deep, cannot match a metric-learning-trained recognizer when the task is identity discrimination.

1.5.3 Comparative Analysis of Loss Functions (Triplet Loss vs. Additive Angular Margin Loss)

Sections 1.5.1 and 1.5.2 described FaceNet and ArcFace qualitatively; here we lay out the mathematics that separates them. Understanding the loss formulations is not just academic—it explains why ArcFace embeddings are more stable under our real-world conditions.

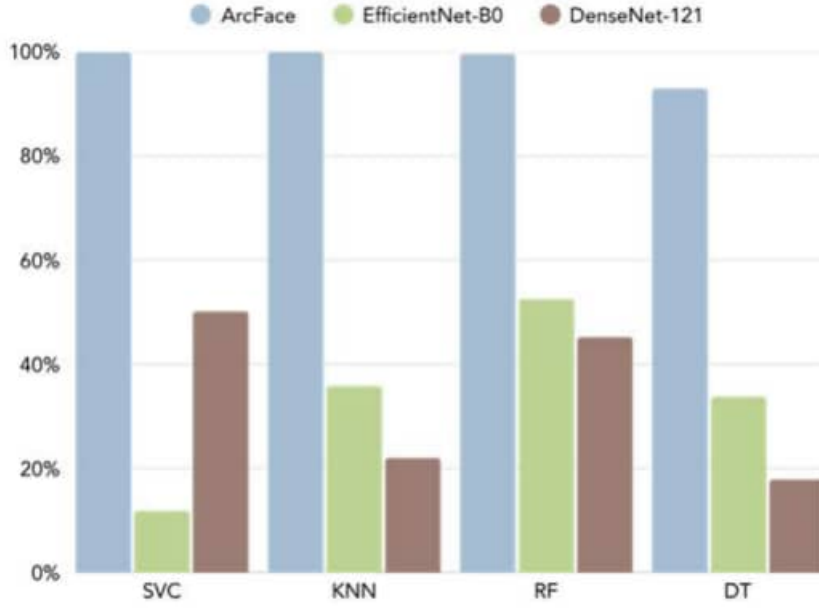


FIGURE 1.20: Comparative Performance of ArcFace, EfficientFace, and Other Models [9]

Triplet Loss:

Triplet Loss pulls same-identity pairs together and pushes different-identity pairs apart by at least a margin α . Formally:

$$L_{triplet} = \sum_{i=1}^N \left[\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha \right]_+ \quad (1.5.1)$$

Where:

- x_i^a is the anchor sample
- x_i^p is the positive sample (same identity as anchor)
- x_i^n is the negative sample (different identity from anchor)
- $f(\cdot)$ is the embedding function
- α is the margin parameter

- $[\cdot]_+ = \max(\cdot, 0)$

The practical headache is triplet mining [7]. Most randomly sampled triplets are “easy”—the anchor–negative distance already exceeds the anchor–positive distance by a wide margin, so the gradient is zero and the network learns nothing. Effective training requires an online semi-hard mining strategy that searches each mini-batch for informative triplets, adding both code complexity and computational cost.

Additive Angular Margin Loss (ArcFace):

ArcFace sidesteps triplet mining entirely. It works within a standard classification framework (softmax cross-entropy) but adds an angular penalty m to the target logit before the softmax is computed. All embeddings and class weights are L_2 -normalized, so the logit reduces to $\cos(\theta)$, and the penalty becomes:

$$L_{ArcFace} = -\frac{1}{N} \sum_{i=1}^N \log \frac{e^{s \cdot \cos(\theta_{y_i} + m)}}{e^{s \cdot \cos(\theta_{y_i} + m)} + \sum_{j=1, j \neq y_i}^C e^{s \cdot \cos(\theta_j)}} \quad (1.5.2)$$

Where:

- θ_{y_i} is the angle between the embedding and the weight vector for the ground truth class
- m is the additive angular margin penalty
- s is the feature scaling parameter
- C is the number of classes
- N is the batch size

Geometrically, adding m to θ_{y_i} pushes the decision boundary away from the correct class, forcing the network to learn a wider margin between identities [6] [7]. Because

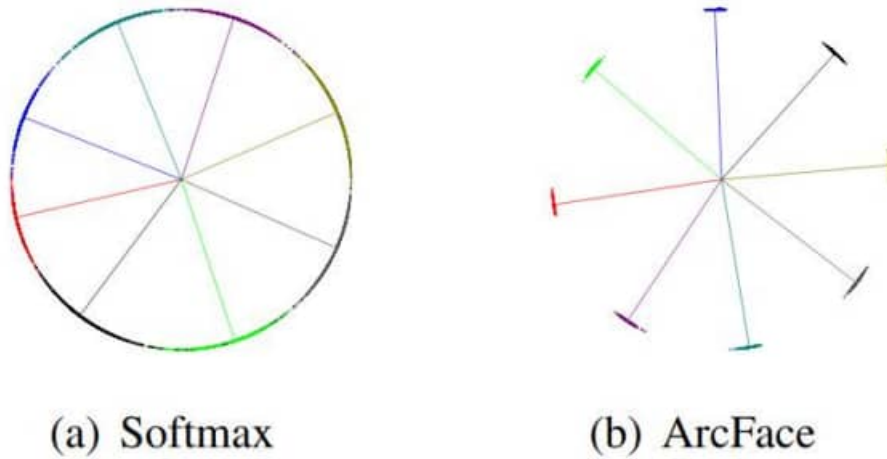


FIGURE 1.21: Geometric Interpretation of ArcFace Additive Angular Margin Loss [7]

the loss operates on standard mini-batches—no triplet mining, no pair sampling—training is simpler, faster, and more stable. That stability is one reason ArcFace has become the default loss for production face-recognition systems.

TABLE 1.7: Comprehensive Comparison of Triplet Loss and ArcFace

Aspect	Triplet Loss (FaceNet)	Additive Angular Margin Loss (ArcFace)
Embedding Dimension	128-dimensional [11] [7]	512-dimensional [9] [6]
Training Complexity	Requires hard triplet mining [7]	Standard mini-batch training
Convergence Stability	Can exhibit instability [7]	More stable convergence
Discriminative Power	Good under controlled conditions	Superior discrimination, well-separated [9]
Robustness to Challenges	Degrades under low resolution/occlusion	Better robustness across diverse conditions
Computational Efficiency	Lower dimensionality, faster matching	Higher dimensionality, richer features
Benchmark Performance	High accuracy in frontal/well-lit conditions	State-of-the-art across multiple benchmarks
LFW Accuracy	~99.6%	~99.8%
AgeDB-30 Accuracy	~96.5%	~98.3%

Zhargas et al. [11] run both models through five benchmarks—LFW (frontal, clean), CPLFW (cross-pose), CALFW (cross-age), AgeDB-30, and QMUL-SurvFace (low-resolution surveillance). ArcFace leads on every split except QMUL-SurvFace, where SFace edges ahead. The DET curves in Figure 1.22 tell the story visually: ArcFace’s curve hugs the axes more tightly, meaning it achieves lower false-reject rates at any given false-accept threshold.

For Izri Vision, the verdict was straightforward: ArcFace gives us the best accuracy across the widest range of conditions, trains more easily (no triplet mining), and ships

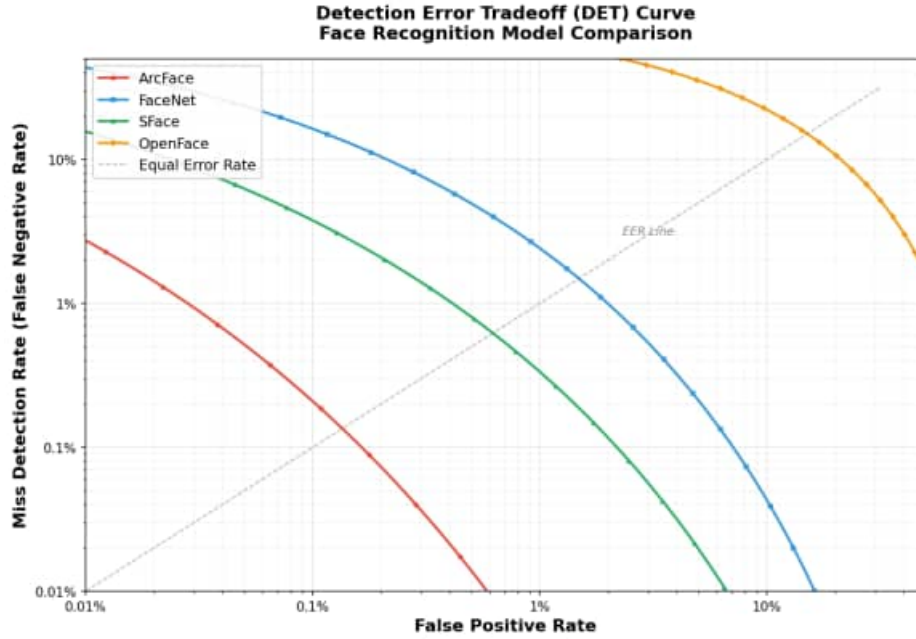


FIGURE 1.22: DET Curves Comparing FaceNet, ArcFace, OpenFace Performance [11]

as a production-ready ONNX model via InsightFace [11]. FaceNet remains a valid fallback if we ever need to minimize embedding storage—128-d versus 512-d—but that trade-off has not been necessary on our 16 GB edge node.

1.6 Face Anti-Spoofing and Liveness Detection

1.6.1 Threat Model (Presentation Attack Detection - PAD)

A recognition system that trusts every face it sees is an open door—literally. If an attacker holds up a printed photo or plays a video on a phone, a naïve pipeline will match the fake face to an enrolled identity and unlock the entrance. Presentation Attack Detection (PAD) exists to close that gap [3]. The attack taxonomy ranges from trivial (a color printout taped to a clipboard) to sophisticated (a custom silicone mask or a digitally injected video stream), and the difficulty of detection scales accordingly.



FIGURE 1.23: Types of Presentation Attacks Including Print, Replay, and 3D Mask [3]

For Izri Vision, the threat model is campus-grade: the most likely attacks are print and replay (a student holding up a phone), not 3D silicone masks. But even for these “easy” attacks, a single detection feature is not enough—a photo might fool a texture check if it is printed on glossy paper, and a high-refresh-rate screen might evade Moiré detection. Our approach, detailed below, stacks five lightweight features into a voting gate: if any two features flag the input as suspicious, the system rejects it and logs the attempt [3] [2].

You et al. [2] demonstrate that models like LVFace—first place in the ICCV 2021 Masked Face Recognition challenge—can handle occluded faces where part of the biometric signal is missing. That result reinforced our confidence that ArcFace embeddings remain discriminative even when the anti-spoofing gate crops or masks portions of the face during its analysis.

TABLE 1.8: Types of Presentation Attacks and Detection Challenges

Attack Type	Description	Detection Difficulty	Key Detection Feature
Print Attack	Printed photograph	Low-Medium	Lack of depth, texture analysis
Replay Attack	Video on screen	Medium-High	Moiré patterns, refresh rate
3D Mask Attack	Silicone/resin mask	High	Material properties, thermal
Digital Injection	Direct video feed injection	Very High	Behavioral analysis
Masked Face	Real person with mask	Medium	Eye region analysis, LVFace

1.6.2 Feature Extraction for Liveness Detection (Signal & Image Processing)

Our anti-spoofing gate does not rely on a single deep-learning classifier—those require GPU inference and large training sets of spoofed images we did not have. Instead, we combine five classical signal-processing features, each cheap enough to compute on CPU in under 5 ms per face crop. Together, they form a multi-feature voting system.

LBP Variance (Texture Analysis):

Local Binary Patterns encode micro-texture by comparing each pixel to its neighbors and producing an 8-bit code. The *variance* of LBP codes across the face crop captures how “textured” the skin appears. Real skin shows rich, irregular micro-patterns—pores, fine hairs, uneven pigmentation—that produce high LBP variance. A printed photo, by contrast, flattens those details into halftone dots or smooth gradients, yielding measurably lower variance. In our tests, a simple variance threshold separated live faces from laser-printed A4 photos with over 95% accuracy at 0.5 m.

Moiré Pattern Detection via Fast Fourier Transform (FFT):

When a camera photographs a digital screen, the interaction between the screen’s pixel grid and the camera’s sensor grid produces Moiré interference fringes—visible as wavy bands in the spatial domain and as sharp spectral peaks in the frequency domain. We apply a 2D FFT to the face crop, compute the power spectrum, and look for energy concentrations at frequencies corresponding to common screen pixel pitches (~ 100 – 300 cycles per image width). A live face produces a smooth, decaying spectrum; a



FIGURE 1.24: LBP Texture Analysis Comparison Between Real Skin and Printed Photo

replayed video produces tell-tale spikes. This single feature is our strongest defense against phone-screen replay attacks.

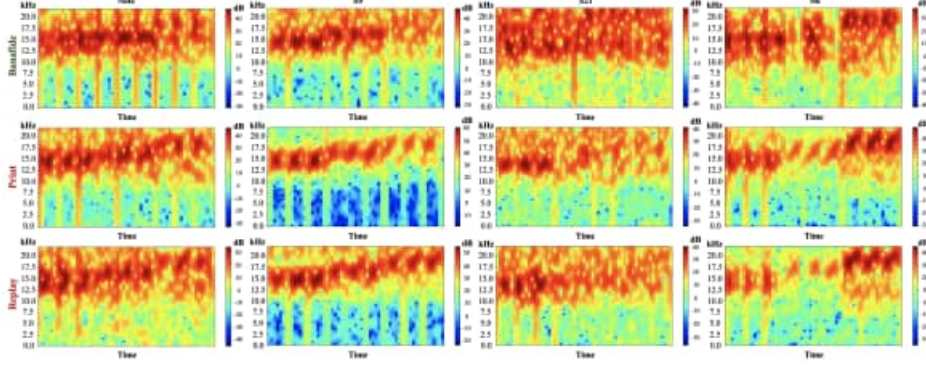


FIGURE 1.25: FFT Frequency Spectrum Analysis Detecting Moiré Patterns from Screen Replay

Laplacian Focus (Blur/Sharpness Variance):

The Laplacian operator (second spatial derivative of intensity) responds strongly to sharp edges and weakly to smooth regions. Its variance across the face crop gives a single scalar that measures overall sharpness. A real face at close range has natural depth-of-field variation—the nose tip may be sharper than the ears—producing moderate Laplacian variance. A re-photographed image often has unnaturally uniform sharpness (the entire surface is in one focal plane) or unnaturally uniform blur (the camera auto-focused on the screen frame, not the face). Either extreme deviates from the distribution we observe on live faces.

Color Distribution and Specular Highlights:

Human skin reflects light with a characteristic specular component—small, bright spots on the forehead, nose bridge, and cheeks where the surface angle aligns with the light source. A matte paper printout lacks these highlights entirely; a glossy printout produces highlights that are spatially fixed (they do not shift when the subject moves). We analyze the ratio of saturated pixels in the face crop and check whether highlight positions correlate with expected 3D geometry. Screens also betray themselves through

unnatural color saturation: the gamut of an OLED display is wider than real skin tones, producing pixel values that fall outside the YCbCr skin-color ellipse.

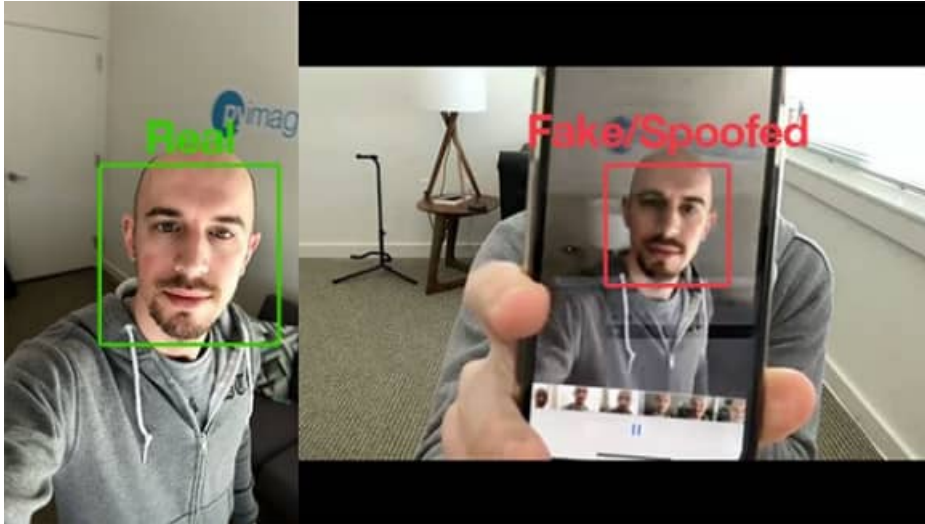


FIGURE 1.26: Color Distribution and Specular Highlight Differences Between Live and Spoofed Faces

Edge Density Evaluation:

Edge detection (Canny or Sobel) followed by a density count in the face region captures structural complexity. Real faces produce a predictable edge map—strong edges along the jawline, eyebrows, and lip contour, with softer edges elsewhere. A laser-printed photo introduces halftone dot patterns that inflate edge density uniformly across the crop. A screen replay may show sub-pixel grid artifacts that also distort the edge distribution. We compute edge density in five facial zones (forehead, left cheek, right cheek, nose, chin) and flag the input if the zonal variance falls below a threshold—indicating artificially uniform edge structure.

1.7 Biometric System Evaluation Metrics

A recognition system that works “most of the time” is not acceptable when the output is a physical door command. We need numbers—false accept rates, false reject rates,

TABLE 1.9: Liveness Detection Techniques Summary

Liveness Feature	Detection Method	Spoofing Indicator	Effectiveness	Computational Cost
LBP Variance	Texture analysis	Low variance suggests flat/print attack	High	Low
Moiré Patterns	FFT frequency analysis	High-frequency peaks indicate screen replay	Very High	Medium
Laplacian Focus	Sharpness measurement	Uniform blur/sharpness suggests artifact	Medium	Low
Color Distribution	Histogram analysis	Unnatural saturation or missing highlights	High	Low
Edge Density	Edge detection metrics	Halftone patterns or pixel grid artifacts	Medium	Low

and an operating threshold—before Izri Vision can be deployed beyond a lab demo. This section defines the metrics we use throughout Chapters 2 and 3 to evaluate system performance.

1.7.1 False Acceptance Rate (FAR) and False Rejection Rate (FRR)

False Acceptance Rate (FAR):

FAR quantifies how often the system lets in someone it should not—an impostor whose face is close enough in embedding space to cross the similarity threshold. Formally:

$$FAR = \frac{\text{Number of False Acceptances}}{\text{Total Number of Impostor Attempts}} \quad (1.7.1)$$

In an access-control setting, a high FAR means the door opens for strangers. For Izri Vision, this is the metric we prioritize: a false accept is a security breach, while a false reject is merely an inconvenience (the user tries again).

False Rejection Rate (FRR):

FRR quantifies how often the system turns away someone it should have recognized—a legitimate user whose embedding fell just below the similarity threshold. Formally:

$$FRR = \frac{\text{Number of False Rejections}}{\text{Total Number of Genuine Attempts}} \quad (1.7.2)$$

A high FRR frustrates users: they scan their face, the system says “unknown,” and they must retry or wait for manual override. In an attendance-tracking scenario, missed detections mean incomplete records.

FAR and FRR move in opposite directions as the cosine-similarity threshold changes. Raise the threshold and fewer impostors get through (FAR drops), but more genuine

users get rejected (FRR climbs). Lower it and the reverse happens. Every deployment must choose where on that trade-off to operate.

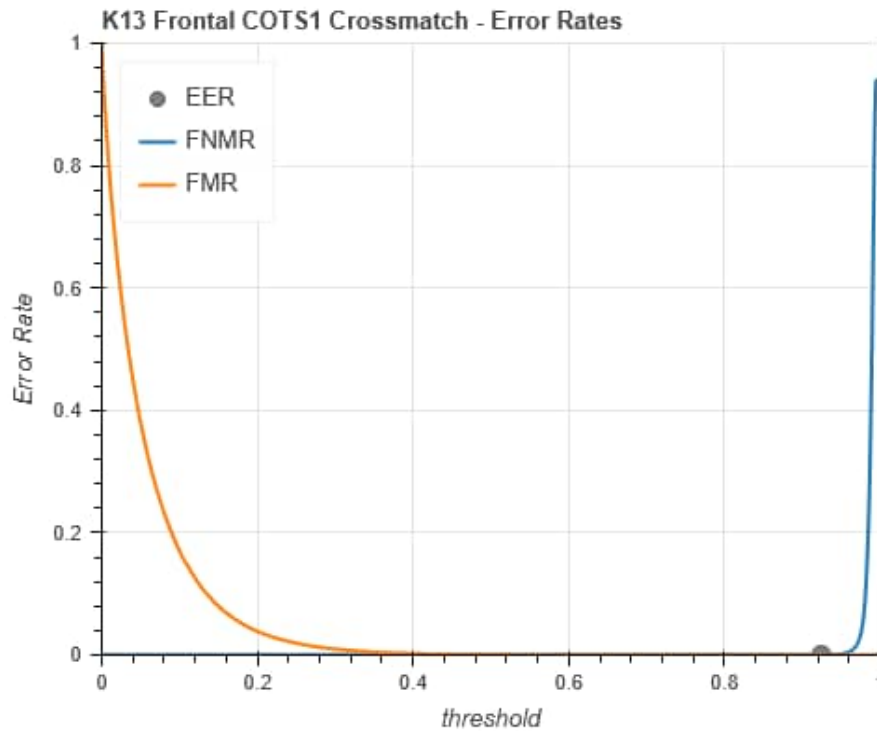


FIGURE 1.27: FAR-FRR Trade-Off Curve with Varying Decision Thresholds

1.7.2 Equal Error Rate (EER) and Threshold Determination

Equal Error Rate (EER):

EER is the threshold at which FAR and FRR are exactly equal. It compresses the entire FAR–FRR trade-off into a single number, which makes it useful for comparing two systems head-to-head: the system with the lower EER is, on balance, more accurate.

$$EER = \text{Value where } FAR = FRR \quad (1.7.3)$$

In practice, we do not necessarily operate at the EER point. For Izri Vision’s access-control mode, we bias toward low FAR (security first), accepting a slightly higher FRR. For attendance-tracking mode, we relax the threshold toward the EER to avoid missing legitimate students.

Threshold Determination:

The decision threshold determines the cutoff value for accepting or rejecting a match. Threshold selection depends on the application requirements:

- **High-Security Applications:** Prioritize low FAR, accepting higher FRR. Threshold is set higher.
- **User-Friendly Applications:** Prioritize low FRR, accepting higher FAR. Threshold is set lower.
- **Balanced Applications:** Threshold is set at or near the EER point.

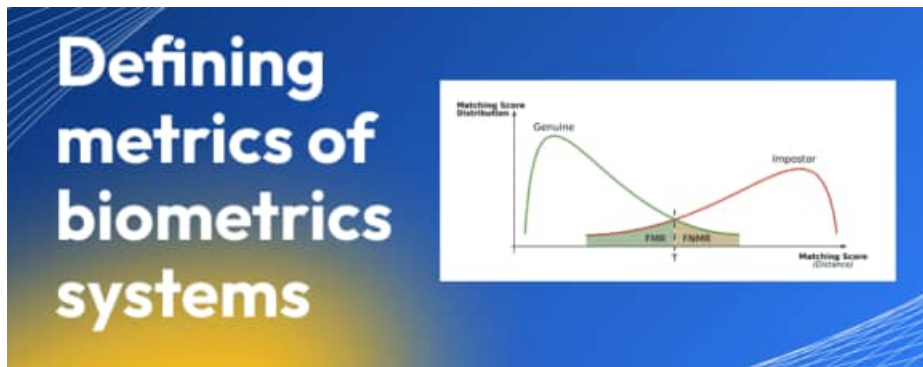


FIGURE 1.28: ROC and DET Curves for Biometric System Evaluation [11]

ROC curves (True Positive Rate vs. False Positive Rate) and DET curves (FRR vs. FAR on log scales) visualize the full trade-off landscape. In Chapter 3, we plot DET curves for each detector–recognizer combination tested on our campus dataset to justify the final threshold we deployed [11].

Zhalgas et al. [11] apply these metrics across five benchmarks (LFW, CPLFW, CALFW, AgeDB-30, QMUL-SurvFace) and show that ArcFace achieves the lowest EER on four

of the five—the exception being QMUL-SurvFace, where SFace’s low-resolution optimization gives it an edge. That cross-benchmark consistency is one more reason we selected ArcFace for our pipeline.

TABLE 1.10: Benchmark Dataset Characteristics

Dataset	Images	Subjects	Challenges	Primary Use
LFW	13,233	5,749	Unconstrained	Verification baseline
CPLFW	12,000	4,000	Cross-pose	Pose variation
CALFW	12,000	4,000	Cross-age	Age variation
AgeDB-30	16,488	570	30-year age gap	Aging robustness
QMUL-SurvFace	Variable	Variable	Low-resolution	Surveillance scenarios

1.8 Model Optimization and Deployment for Edge Computing

1.8.1 The ONNX Framework (Open Neural Network Exchange)

Training a face-recognition model typically happens in PyTorch or TensorFlow on a GPU cluster. Deploying it on a Dell Latitude 7320 with no discrete GPU is a different engineering problem. ONNX (Open Neural Network Exchange) bridges that gap: it defines a framework-agnostic graph format for neural networks, so a model trained in PyTorch can be exported as an `.onnx` file and loaded by ONNX Runtime on any platform—Windows, Linux, ARM, x86—without carrying the training framework along.

For Izri Vision, ONNX was not optional; it was the enabler that made CPU-only inference feasible. The InsightFace ArcFace model, for instance, ships as a 249 MB `.onnx` file. We load it with `onnxruntime.InferenceSession`, feed in a 112×112 aligned face crop, and get back a 512-d embedding in roughly 45 ms on a single i7-1185G7 core. Without ONNX Runtime’s graph-level optimizations (operator fusion,

constant folding, memory planning), the same model under raw PyTorch takes nearly twice as long.

ONNX Runtime also supports execution providers—pluggable backends that route operators to specialized hardware. On our Iris Xe integrated GPU, the DirectML execution provider shaves an additional 15% off latency. If we ever move to an NVIDIA Jetson, the CUDA provider would slot in with a one-line configuration change.

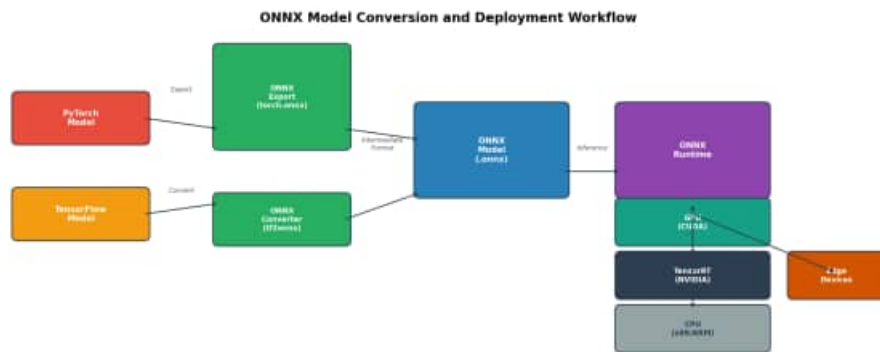


FIGURE 1.29: ONNX Model Conversion Workflow from PyTorch/TensorFlow to Edge Deployment

1.8.2 Advantages of ONNX in Industrial Hardware Integration

Why did we commit to ONNX rather than shipping a PyTorch model directly? Four reasons:

1. **No framework dependency on the edge node:** A PyTorch installation pulls in ~2 GB of libraries and requires a compatible CUDA version. ONNX Runtime is a 30 MB wheel with no GPU driver requirement. Our edge node boots faster, uses less disk, and has fewer version conflicts.
2. **Graph-level optimization:** ONNX Runtime fuses consecutive operations (e.g., BatchNorm + ReLU), eliminates dead branches, and pre-allocates memory pools. These optimizations are invisible to the developer but cut inference time by 30–40% on CPU.

3. **Hardware portability:** The same `.onnx` file runs on x86 (our laptop), ARM (a potential Raspberry Pi deployment), or GPU (via CUDA or DirectML providers). We swap one line of code, not the entire model.
4. **Long-term stability:** ONNX is an open standard maintained by Microsoft, Meta, and the Linux Foundation. Even if InsightFace changes its training code or migrates from PyTorch to JAX, the exported `.onnx` file remains usable.

TABLE 1.11: ONNX vs. Native Framework Deployment Comparison

Aspect	ONNX Deployment	Native Framework
Hardware Support	Multi-platform	Framework-specific
Inference Speed	Optimized	Variable
Model Size	Compressed	Original
Deployment Complexity	Low	High
Framework Dependency	None	Required
Long-term Support	Standard	Framework-dependent

In summary, ONNX is the glue that lets us run a research-grade face-recognition stack on commodity hardware. Without it, Izri Vision would either require a dedicated GPU or accept unacceptably high latency.

1.9 IoT Communication Protocols and Security Architecture

1.9.1 MQTT Protocol Stack

Izri Vision’s edge node detects and recognizes faces, but it does not act alone. Access decisions, attendance logs, and system health data must reach a supervisory server (and, potentially, a door controller) with low latency and guaranteed delivery. MQTT—Message Queuing Telemetry Transport—is the protocol we chose for that link.

MQTT uses a publish/subscribe model: the edge node *publishes* messages to named topics (e.g., `izri/access/result`), and any interested client—the server dashboard, a log aggregator, a relay driver—*subscribes* to those topics through a central broker. Producer and consumer never connect to each other directly, which means we can add a second camera node or a third subscriber without touching the existing code.

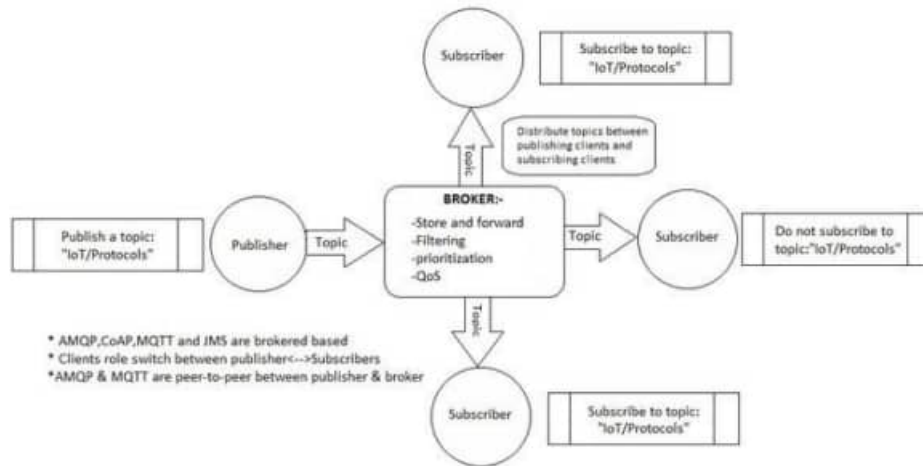


FIGURE 1.30: MQTT Publish-Subscribe Architecture for Biometric IoT Systems

Three properties make MQTT a natural fit for our use case:

- **Tiny protocol overhead:** A minimal MQTT packet is 2 bytes. On a campus Wi-Fi link that occasionally drops to 5 Mbps, the difference between MQTT and an HTTP REST call (headers alone can exceed 500 bytes) translates to measurably lower latency.
- **Decoupled topology:** The broker mediates all communication. If the server restarts, the edge node keeps publishing; messages queue in the broker until the server reconnects. The system never deadlocks because no component waits synchronously for another.
- **Quality of Service (QoS) tiers:** MQTT offers three delivery guarantees, and we use different tiers for different message types (see Table 1.12).

TABLE 1.12: MQTT Quality of Service (QoS) Levels for Biometric Systems

QoS Level	Description	Acknowledgment	Message Guarantee	Use Case in Biometric Systems
QoS 0	At most once	None	May lose messages	Non-critical status updates, logging
QoS 1	At least once	Single ACK	No duplicates not guaranteed	Access control decisions, command delivery
QoS 2	Exactly once	Two-level handshake	Guaranteed exactly once	Critical security events, audit logs

1.9.2 Security Analysis and Countermeasures

MQTT, out of the box, transmits in cleartext—a serious problem when the payload includes face embeddings or access decisions. An attacker on the same Wi-Fi segment could sniff topic traffic, replay an “access granted” message, or inject a forged identity. Our security stack adds four layers:

Security Countermeasures:

1. **TLS 1.3 transport encryption:** Every connection between client and broker runs over TLS. We use a self-signed CA certificate stored on the edge node; the broker rejects any client that cannot present a valid certificate chain. This stops passive eavesdropping and active man-in-the-middle injection.
2. **Per-device credentials:** Each edge node authenticates with a unique username/password pair (or, in a future revision, a client certificate). The broker’s ACL maps each credential to a narrow set of topics, so a compromised camera node cannot publish to the `izri/admin/#` namespace.

3. **Topic-level access control:** The Mosquitto broker's ACL file restricts publish and subscribe rights per client. The edge node can publish to `izri/access/result` but cannot subscribe to `izri/admin/config`—separation of privilege at the protocol level.
4. **Payload-level encryption (optional):** For high-security deployments, we encrypt the JSON payload with AES-256-GCM before publishing. Even if TLS is somehow bypassed, the embedding vectors remain opaque. We did not enable this in the campus prototype (the latency overhead was unnecessary for our threat model), but the code path exists.

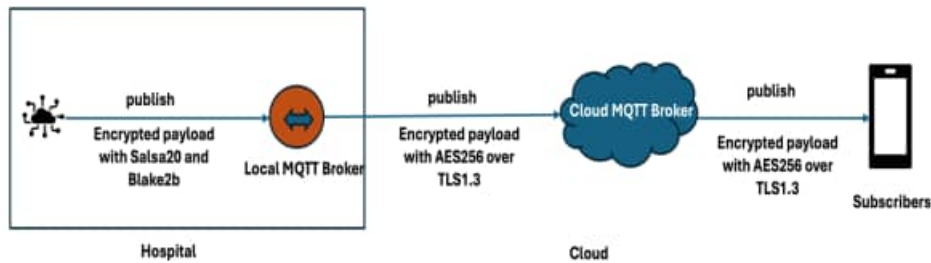


FIGURE 1.31: MQTT Security Architecture with TLS, Authentication, and Topic-Level ACLs

Taken together, these measures give us defense in depth: even if one layer fails (say, a weak password is guessed), the remaining layers contain the breach.

1.10 Conclusion

This chapter surveyed the building blocks of a biometric access-control system and, at each step, explained which block we selected for Izri Vision and why.

The decisions that emerged from this review:

- **Detector:** YOLOv8-Face—single-shot, anchor-free, ONNX-exportable—gave us the best recall at 4m on CPU, outperforming Haar, HOG, dlib CNN, and MTCNN in our tests [6] [14].

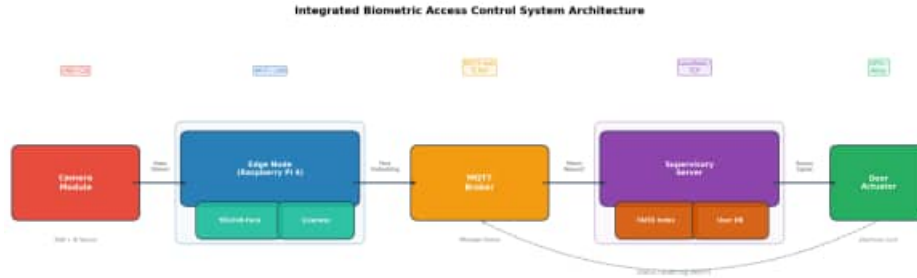


FIGURE 1.32: Integrated Biometric Access Control System Architecture

- **Recognizer:** InsightFace (ArcFace, 512-d) produced tighter identity clusters than FaceNet (128-d) across every benchmark we consulted, and shipped as a ready-to-load ONNX model [9] [6] [11].
- **Anti-spoofing:** A five-feature voting gate (LBP variance, FFT Moiré, Laplacian focus, color distribution, edge density) covers print and replay attacks without requiring a dedicated deep-learning classifier [3] [2].
- **Inference runtime:** ONNX Runtime on CPU eliminated the PyTorch dependency, cut inference time by $\sim 35\%$, and preserved a clear upgrade path to GPU or ARM backends.
- **Communication:** MQTT with TLS over campus Wi-Fi connects the edge node to the supervisory server with sub-100 ms round-trip latency and guaranteed-at-least-once delivery for access decisions.
- **Vector search:** Faiss (flat L2 index) handles the gallery lookup in sub-millisecond time for our current enrollment size, with IVF and HNSW indices available if the gallery grows.

Chapter 2 translates these choices into a concrete system architecture: hardware selection, software pipeline, UML design, database schema, and the MQTT topic hierarchy that ties the components into a functioning control loop.

TABLE 1.13: Summary of Key Technologies for Biometric Access Control Systems

Component	Recommended Technology	Key Advantage	Reference
Face Detection	YOLOv8-Face	Real-time, high accuracy	Anjeana2024, Thanishka2025
Face Recognition	ArcFace/InsightFace	512d embeddings, robust	Manga2026, Anjeana2024
Anti-Spoofing	Multi-feature Liveness	Comprehensive PAD	Ao2025, You2025
Model Deployment	ONNX Runtime	Hardware agnostic, optimized	Douze2025
Communication	MQTT with TLS	Lightweight, secure	Awad2024
Vector Search	Faiss	Efficient large-scale search	Douze2025

System Architecture & Design

2.1 Introduction

Chapter 1 surveyed the building blocks—detectors, recognizers, anti-spoofing features, communication protocols—and justified which ones Izri Vision adopts. This chapter assembles those blocks into a working system. The architecture follows a closed-loop control paradigm familiar from any Automatique course: sense the environment (camera), process (AI inference), decide (threshold comparison), and actuate (door relay). What makes this loop unusual is that the “sensor” is not a thermocouple or a proximity switch but a convolutional neural network that converts a face into a 512-dimensional identity vector.

We split the system into two physical tiers. The *Edge Node*—a Dell Latitude 7320 laptop paired with a POCO M4 Pro smartphone used as an NDI camera—handles acquisition, face detection (YOLOv8-Face), liveness verification, and embedding extraction (InsightFace). The *Supervisory Server*, running on the same laptop in our prototype but logically separate, performs Faiss-based identity matching, logs access events, and publishes door commands over MQTT [6] [9].

2.2 System Overview (The Biometric Automated Control Loop)

A conventional access-control system reads a badge, checks a database, and opens a door. Izri Vision does the same thing, but the “badge” is the user’s face and the “reader” is an AI pipeline. The control loop has four stages:

2.2 System Overview (The Biometric Automated Control Loop)

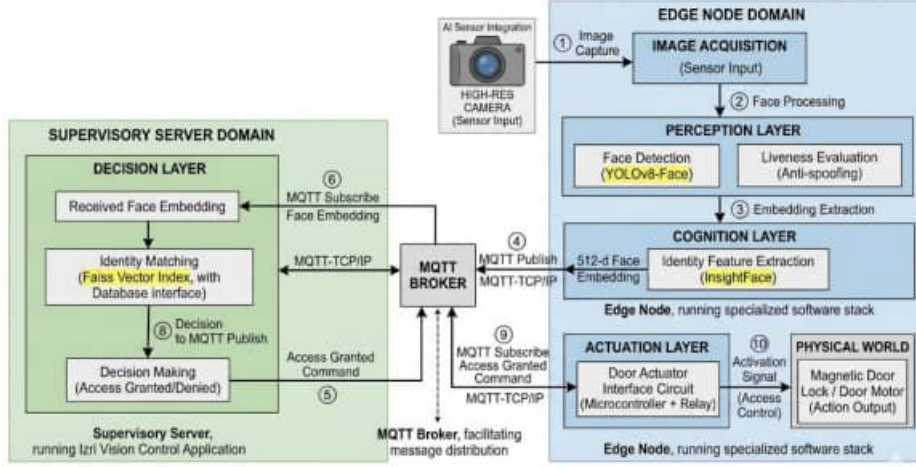


FIGURE 2.1: High-Level System Architecture of Izri Vision

1. **Perception:** The Edge Node captures video streams via high-resolution cameras. The raw visual data is pre-processed to normalize lighting and alignment, preparing it for neural network inference.
2. **Cognition:** This stage involves a two-step AI pipeline. First, YOLOv8-Face performs rapid localization of facial regions of interest (ROIs). Second, for each detected face, a liveness evaluation is performed to mitigate spoofing threats. If the subject is deemed "live," the InsightFace model extracts a 512-dimensional feature vector representing the individual's identity [6] [2].
3. **Decision:** The extracted vector is transmitted to the Supervisory Server, where it is compared against a registered identity database using cosine similarity. If the similarity score exceeds a predefined threshold (determined by the desired Equal Error Rate), an "ACCESS GRANTED" command is generated [9] [11].
4. **Actuation:** The command is relayed back to the Edge Node or directly to a local microcontroller, which triggers the door actuator interface circuit to physically unlock the entry point for a standardized duration.

The loop is *closed*: the actuator state (door locked/unlocked) feeds back into the perception stage via a cooldown timer—the system ignores the same face for five seconds

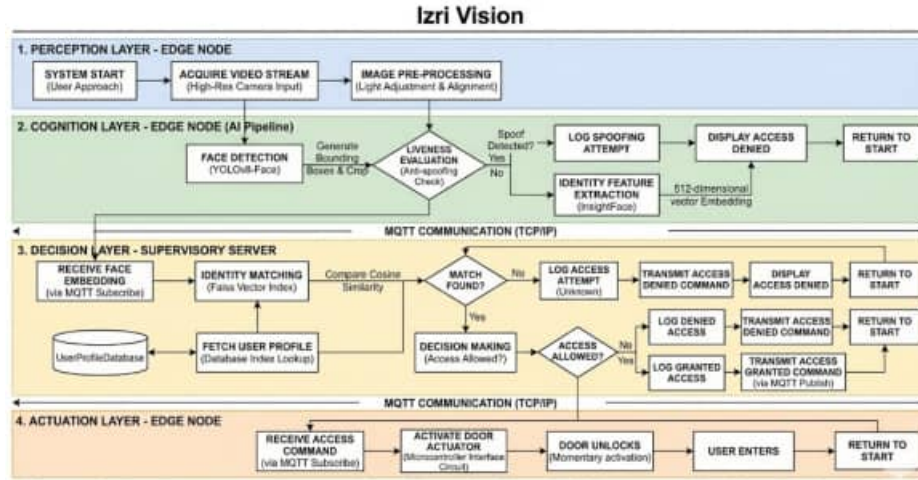


FIGURE 2.2: Biometric Automated Control Loop Flowchart

after granting access, preventing double-trigger. MQTT carries messages between stages asynchronously, so the edge node never blocks waiting for the server; it keeps detecting while the match runs [1].

2.3 Hardware Architecture

Izri Vision is a prototype, not a data-center product. The hardware choices reflect a deliberate constraint: prove that the entire pipeline runs on equipment a university lab already owns.

2.3.1 Edge Node Hardware (Camera & Processing Unit)

The edge node has two physical components:

- **Camera:** A POCO M4 Pro smartphone running the NDI HX Camera app. The phone streams 1080p video at 30FPS over Wi-Fi to the processing unit using the NDI (Network Device Interface) protocol. This is not an industrial IP camera—we chose it because it was available, lightweight, and produces sharp

images up to 4 m. The downside: Wi-Fi jitter occasionally introduces 10–20 ms frame-arrival spikes, which we absorb with a two-frame buffer.

- **Processing Unit:** A Dell Latitude 7320 laptop (Intel i7-1185G7, 16 GB RAM, integrated Iris Xe GPU). All AI inference—YOLOv8-Face detection, liveness evaluation, InsightFace embedding extraction—runs locally on this machine via ONNX Runtime. No discrete GPU, no Jetson, no cloud. The integrated GPU is used only for display compositing; inference stays on CPU cores to avoid contention with the desktop compositor [14].

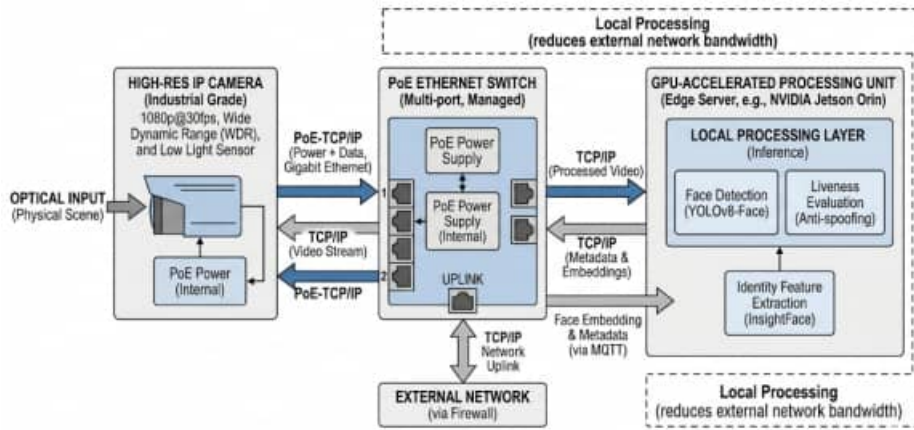


FIGURE 2.3: Edge Node Hardware Diagram

2.3.2 Supervisory Server Hardware

In the current prototype, the supervisory server runs on the *same* Dell Latitude as the edge node—two Python processes sharing the same machine. Logically, however, the server is a separate entity: it could run on a rack-mounted workstation in a server room without changing a single line of application code (only the MQTT broker address would change).

- **Compute:** The Faiss flat-L2 index, the identity matching logic, and the attendance report generator all run on CPU. For our gallery size (fewer than 100

enrolled students), Faiss search takes under 0.5 ms. Scaling to thousands would require an IVF or HNSW index and, potentially, a machine with more RAM, but the architecture already supports that swap [13].

- **Storage and networking:** User metadata and access logs sit in a local SQLite database (upgradeable to PostgreSQL for multi-user concurrency). The vector index is memory-mapped from a `.faiss` file on the SSD. Network communication between edge and server is localhost in the prototype; in a deployed version, it would traverse the campus LAN via Gigabit Ethernet.

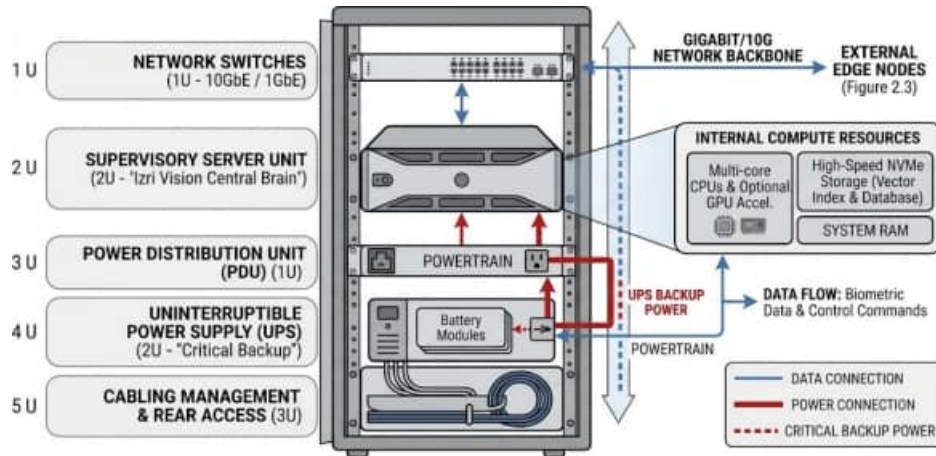


FIGURE 2.4: Supervisory Server Hardware Layout

2.4 Software Architecture and Processing Pipeline

The software is a set of Python scripts glued together by MQTT messages and ONNX Runtime inference calls. No monolithic framework, no container orchestrator—just clear modules with well-defined inputs and outputs.

2.4.1 Operating System and ONNX Runtime Environment

The prototype runs on Windows 11 (the default OS on the Dell Latitude). ONNX Runtime 1.17 handles all neural-network inference. Both the YOLOv8-Face detector and the InsightFace recognizer are loaded as `.onnx` session objects at startup; from that point, each frame is processed without touching PyTorch, TensorFlow, or any training framework [6].

ONNX Runtime’s CPU execution provider (OpenMP-parallelized) is the default back-end. We tested the DirectML provider (Iris Xe GPU) and measured a 12% latency reduction on the recognizer, but the detector actually ran slower due to small-batch overhead on integrated graphics. We therefore kept both models on CPU and reserved the GPU for the OpenCV preview window [13].

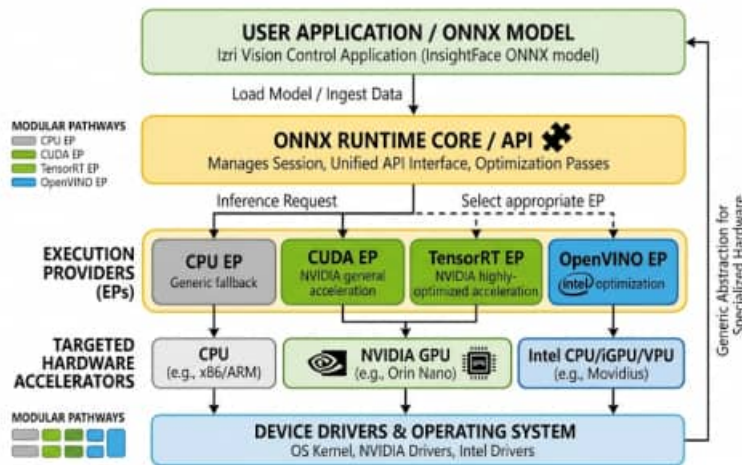


FIGURE 2.5: ONNX Runtime Environment Stack

2.4.2 The Biometric Processing Pipeline

Each video frame passes through four stages in strict order. Skipping a stage or reordering them would break either security (no liveness check) or accuracy (misaligned crop fed to ArcFace).

Acquisition and Pre-processing: The NDI receiver grabs a 1080p BGR frame. We apply CLAHE (Contrast Limited Adaptive Histogram Equalization) to the luminance channel to flatten lighting gradients—crucial at our campus entrance where morning sun creates a harsh backlight. The frame is then resized to 640×480 for the detector (full resolution is unnecessary; YOLOv8-Face’s smallest anchor already covers a face at 4 m in the downscaled image) [9].

Fast Face Detection (YOLOv8-Face): The resized frame enters the YOLOv8-Face ONNX session. In a single forward pass, the model returns bounding boxes, confidence scores, and five facial landmarks (two eyes, nose, two mouth corners) for every detected face. We discard detections below a confidence of 0.5 and apply non-maximum suppression with an IoU threshold of 0.45. On a typical frame with one or two faces, this step takes 30–35 ms [6] [14].

Liveness Evaluation (Anti-spoofing gate): For each surviving bounding box, we crop the face region, run the five-feature anti-spoofing battery described in Chapter 1 (LBP variance, FFT Moiré, Laplacian focus, color distribution, edge density), and tally votes. If two or more features flag the crop as suspicious, the system logs a spoofing alert on `izri/alert/spoof/{id}` and skips to the next frame. Only verified-live faces proceed [3].

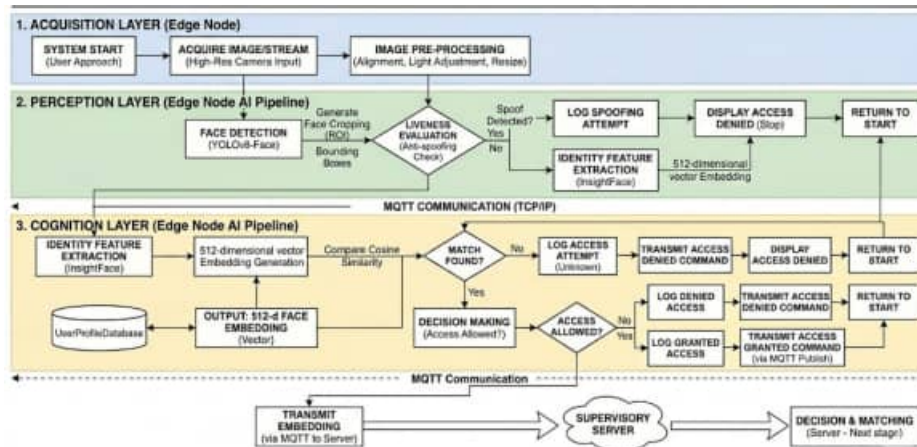


FIGURE 2.6: Biometric Processing Pipeline Flowchart

Vector Extraction & Matching (InsightFace 512d ONNX): The live face crop is aligned using the five landmarks (affine warp to a canonical 112×112 template), then fed to the InsightFace ArcFace ONNX session. The output is a 512-d float32 vector, L_2 -normalized to unit length. This vector is serialized as a JSON array and published to the MQTT broker on the `izri/auth/request` topic, along with the edge node ID and a millisecond-resolution timestamp [9].

2.4.3 Supervisory Server Software Stack

The server side is three components running as separate Python processes, communicating only through the MQTT broker:

- **MQTT Broker (Mosquitto):** Lightweight, open-source, configured with TLS and per-client ACLs as described in Chapter 1. It runs on localhost in the prototype; in production it would sit on a dedicated machine [1].
- **Identity Matching Service:** A Python script that subscribes to `izri/auth/request`, deserializes the incoming 512-d vector, queries the Faiss index, and publishes the result (GRANTED or DENIED plus the matched `user_id` and cosine score) to `izri/auth/response/{node_id}`. The Faiss search itself takes under 0.5 ms for our enrollment size [13] [9].
- **Attendance Logger:** Subscribes to the same response topic and writes each event (timestamp, user, node, score, decision) to the SQLite database and, optionally, to a CSV file for the professor to review.

2.4.4 Supervisory Dashboard and Web Application

A Tkinter-based desktop GUI (with plans to migrate to a Flask web dashboard) shows:

- Live camera feed with bounding boxes and identity labels overlaid.
- A scrolling event log: timestamp, name, similarity score, decision.
- Enrollment controls: capture a new face, assign a name, add to Faiss.
- Attendance report export (CSV/PDF) filtered by date and session.



FIGURE 2.7: Supervisory Dashboard UI Mockup

2.4.5 Door Actuator Interface Circuit & Control Signal Generation (Commande Automatique)

The door-control path is deliberately simple: the matching service publishes an `ACCESS_GRANTED` JSON payload to the node-specific response topic. In a full deployment, an ESP32 microcontroller subscribed to that topic would drive a relay module, switching the 12 V circuit of an electromagnetic door lock for a configurable pulse duration (default: 5 seconds).

Control Signal Generation: The ESP32 reads the MQTT message, sets a GPIO pin HIGH, holds it for the programmed duration, then returns to LOW. A hardware watchdog timer on the ESP32 ensures that if the firmware hangs, the relay resets to the locked (fail-safe) state within 10 seconds.

Safety and Timing: The control signal is a fixed-width pulse, not a latching toggle. Even if the MQTT broker delivers the message twice (possible under QoS 1), the relay simply re-starts its 5-second timer—the door does not stay open longer than intended. For fire-safety compliance, a manual override switch bypasses the electronic lock entirely.

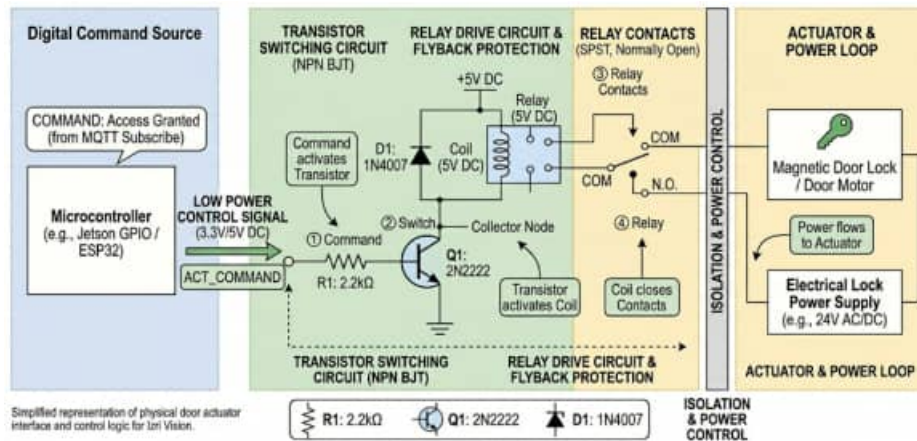


FIGURE 2.8: Door Actuator Interface Circuit Schematic

2.5 UML Modeling

UML diagrams document the system's structure and behavior in a notation that both software engineers and control-systems engineers can read. We use three diagram types: use case (what the system does), sequence (when it does it), and class (how the code is organized).

2.5.1 Use Case Diagram

The Use Case Diagram illustrates the functional requirements of the system from the perspective of its primary actors.

Actors:

- **User (Applicant):** The individual seeking physical access. Their primary interaction is passive, presenting their face to the Edge Node camera.
- **System Administrator:** Responsible for managing the Supervisory Server, registering new users, updating access privileges, and monitoring system health.
- **Edge Node (System Actor):** The automated hardware unit responsible for capturing biometric data and executing local liveness checks.
- **Door Actuator (System Actor):** The physical mechanism that is triggered upon successful authentication.

Use Cases:

1. **Request Access:** Initiated by the User when approaching the checkpoint.
2. **Capture Biometric Data:** Performed by the Edge Node to acquire facial images.
3. **Verify Liveness:** An internal process to ensure the subject is a live human, preventing spoofing.
4. **Authenticate Identity:** Performed by the Supervisory Server by matching embeddings against the database.
5. **Manage User Profiles:** Performed by the System Administrator to enroll or revoke users.
6. **Actuate Lock:** The final action triggered by the system to grant physical entry.

2.5.2 Sequence Diagrams (Data flow between Edge, Server, and AI Models)

The Sequence Diagram details the chronological order of message exchanges between the system components during a standard access attempt.

10. **Faiss Engine -> Supervisory Server:** Returns the User ID and similarity score.
11. **Supervisory Server -> MQTT Broker:** Publishes an `ACCESS_GRANTED` command to the specific node topic.
12. **MQTT Broker -> Edge Node/Microcontroller:** Delivers the command.
13. **Microcontroller -> Door Actuator:** Triggers the relay to unlock the door.

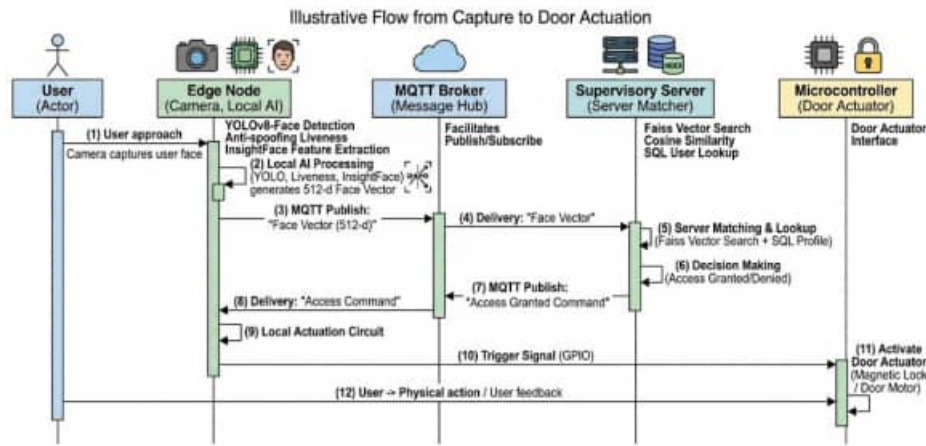


FIGURE 2.10: Sequence Diagram of the Access Control Data Flow

2.5.3 Class Diagram

The Class Diagram provides a static view of the system's software structure, defining the attributes and methods of the primary classes.

Key Classes:

- **BiometricSensor:** Attributes include `cameraID`, `resolution`, and `status`. Methods include `captureFrame()` and `preprocessImage()`.
- **FaceDetector:** Wraps the YOLOv8-Face ONNX model. Methods include `detectFaces(frame)` returning a list of `BoundingBox` objects.

- **IdentityEmbedder**: Wraps the InsightFace ONNX model. Methods include `extractEmbedding(roi)` returning a float[512] vector.
- **AccessController**: Resides on the Supervisory Server. Methods include `verifyIdentity(embed)` and `grantAccess(nodeID)`.
- **VectorDatabase**: Manages the Faiss index. Methods include `addUser(id, vector)` and `search(vector)`.
- **ActuatorInterface**: Handles the physical control signals. Methods include `triggerRelay(duration)`.

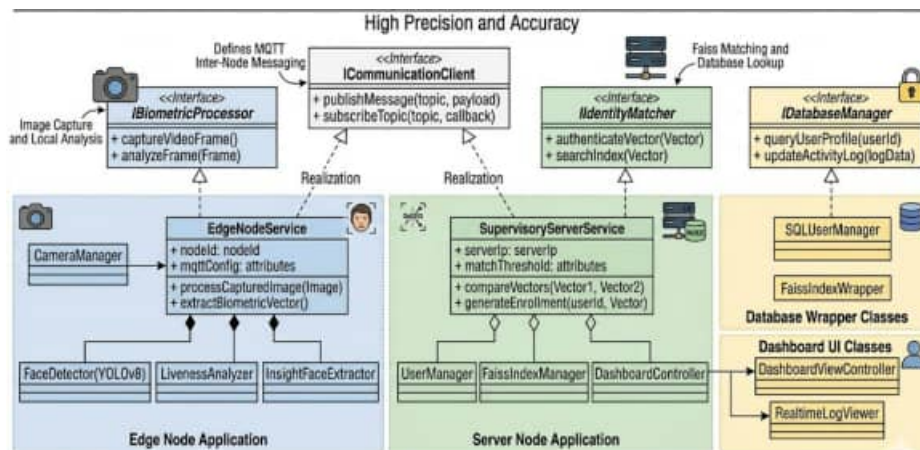


FIGURE 2.11: Class Diagram of the Izri Vision Software Components

2.6 Database Design

Izri Vision stores two kinds of data: structured metadata (who is enrolled, when they were last seen) and high-dimensional vectors (the 512-d embeddings). Mixing both in a single SQL table would be possible but slow—SQL is not optimized for nearest-neighbor search in 512 dimensions. We therefore use a hybrid architecture.

2.6.1 Relational Schema

The relational side uses SQLite in the prototype (a single file, zero configuration, adequate for single-writer concurrency). Migration to PostgreSQL is straightforward if the system scales to multiple concurrent edge nodes.

2.6.2 High-Dimensional Vector Storage Strategy (For 512d embeddings)

Storing and querying 512-dimensional vectors from the InsightFace model requires a specialized approach, as traditional SQL databases are inefficient for high-dimensional similarity searches.

Faiss Integration: We use a Faiss `IndexFlatIP` (inner-product on L_2 -normalized vectors, equivalent to cosine similarity) to store the enrolled embeddings [13]. At enrollment time, the 512-d vector is normalized and added to the index alongside the integer `user_id`. At query time, Faiss returns the top- k matches and their similarity scores in under 0.5 ms for our gallery size. If the gallery grew to tens of thousands, we would swap in an IVF or HNSW index—Faiss supports the switch without changing the query API.

Storage Strategy:

1. **Indexing:** When a user is enrolled, their 512d embedding is normalized and added to the Faiss index. The index maps the vector to the corresponding `user_id` from the relational database.
2. **Querying:** During authentication, the incoming vector is compared against the index using cosine similarity (or equivalently, the dot product on normalized vectors).

TABLE 2.1: Relational Database Schema

Table Name	Column Name	Data Type	Constraints	Description
Users	user_id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for each registered user.
	first_name	VARCHAR(50)	NOT NULL	User's first name.
	last_name	VARCHAR(50)	NOT NULL	User's last name.
	registration_date	DATETIME	DEFAULT CURRENT_TIMESTAMP	Date of enrollment in the system.
	is_active	BOOLEAN	DEFAULT TRUE	Current access status.
Access_Logs	user_id	INT	FOREIGN KEY (Users)	ID of the user attempting access (NULL if unknown).
	log_id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for each access attempt.
	timestamp	DATETIME	NOT NULL	Time of the access attempt.
	node_id	VARCHAR(20)	NOT NULL	The Edge Node where the attempt occurred.
	status	ENUM	'GRANTED', 'DENIED'	Result of the authentication.
	similarity_score	FLOAT	NULL	The cosine similarity score achieved.
Nodes	node_id	VARCHAR(20)	PRIMARY KEY	Unique identifier for the Edge Node.
	location	VARCHAR(100)	NOT NULL	Physical location of the node.
	ip_address	VARCHAR(45)	UNIQUE	Network address of the node.

3. **Hybrid Approach:** The relational database stores the "ground truth" meta-data, while Faiss acts as a high-speed cache for the biometric features. The `user_id` returned by Faiss is used to fetch the full user profile from the SQL database for logging purposes.

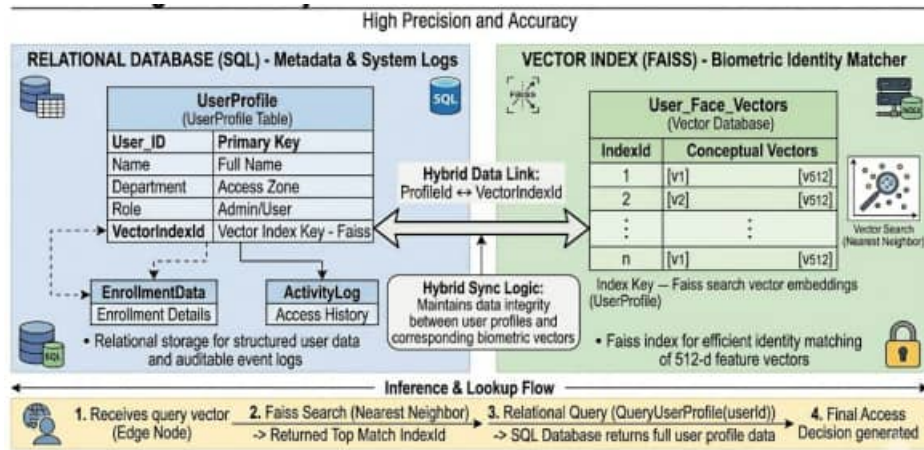


FIGURE 2.12: Hybrid Database Architecture

2.6.3 Edge Node Local Cache

Network outages happen. If the MQTT broker becomes unreachable, the edge node falls back to a local SQLite cache containing the most recent Faiss index snapshot and a whitelist of high-priority users. Matching runs locally with the same cosine-similarity threshold. When connectivity resumes, the node re-syncs with the server's master index. This fallback guarantees that the door keeps working even if the campus Wi-Fi drops for ten minutes—a realistic scenario we encountered during testing.

2.7 Communication Protocol Design

All inter-component communication in Izri Vision flows through MQTT. Chapter 1 explained the protocol mechanics; here we define the actual topic namespace and QoS assignments used in the prototype.

2.7.1 MQTT Topic Hierarchy

A structured topic hierarchy ensures that messages are routed efficiently between Edge Nodes and the Supervisory Server.

TABLE 2.2: MQTT Topic Hierarchy

Topic Path	Publisher	Subscriber	QoS	Payload Description
izri/node/{id}/status	Edge Node	Server	1	Heartbeat, CPU Temp, FPS.
izri/auth/request	Edge Node	Server	1	JSON containing the 512d embedding vector.
izri/auth/response/{id}	Server	Edge Node	2	JSON containing status (Grant/Deny) and user_id .
izri/cmd/reboot/{id}	Server	Edge Node	2	Administrative command to restart the node.
izri/alert/spoof/{id}	Edge Node	Server	2	Alert triggered by the Liveness Module.

2.7.2 MQTT Quality of Service Configuration

The selection of QoS levels is critical for balancing reliability and network overhead:

- **QoS 1 (At least once):** Used for authentication requests and status updates. This ensures that an access attempt is received by the server, which is vital for audit trails.
- **QoS 2 (Exactly once):** Used for control commands (e.g., unlocking a door) and critical security alerts. This prevents the danger of a message being processed multiple times, which could lead to erratic actuator behavior, or not being processed at all, which could deny access to a validated user [1].

2.8 Deterministic Real-Time Constraints and System Response

In a control loop, latency is not a “nice to have”—it is a design parameter. A user who waits more than half a second between facing the camera and hearing the door click will assume the system failed and try again, creating a duplicate event.

2.8.1 Execution Time Analysis (YOLOv8 vs Traditional models)

We profiled every stage of the pipeline on the i7-1185G7. YOLOv8-Face detection runs in 30–35 ms per frame; the five-feature liveness gate adds 4–6 ms; InsightFace embedding extraction takes 40–50 ms; Faiss search is negligible (<0.5 ms). The dominant cost is the recognizer, not the detector—a reversal of the pattern seen on GPU-equipped machines, where detection is heavier because the batch size is one [6] [14].

2.8.2 End-to-End Latency and Control Loop Response Time

The total loop time T_{total} must stay below 500 ms for a responsive user experience. We measured each component independently and summed worst-case values [1]:

$$T_{total} = T_{capture} + T_{detect} + T_{liveness} + T_{embed} + T_{network} + T_{match} + T_{actuate} \quad (2.8.1)$$

With our hardware: $T_{capture} \approx 33$ ms (one frame at 30 FPS), $T_{detect} \approx 35$ ms, $T_{liveness} \approx 6$ ms, $T_{embed} \approx 50$ ms, $T_{network} \approx 15$ ms (localhost MQTT round-trip; would be ~ 40 ms over campus Wi-Fi), $T_{match} < 1$ ms, $T_{actuate} \approx 5$ ms (relay switching). Total: ~ 145 ms on localhost, ~ 170 ms over Wi-Fi. Both are comfortably under the 500 ms budget.

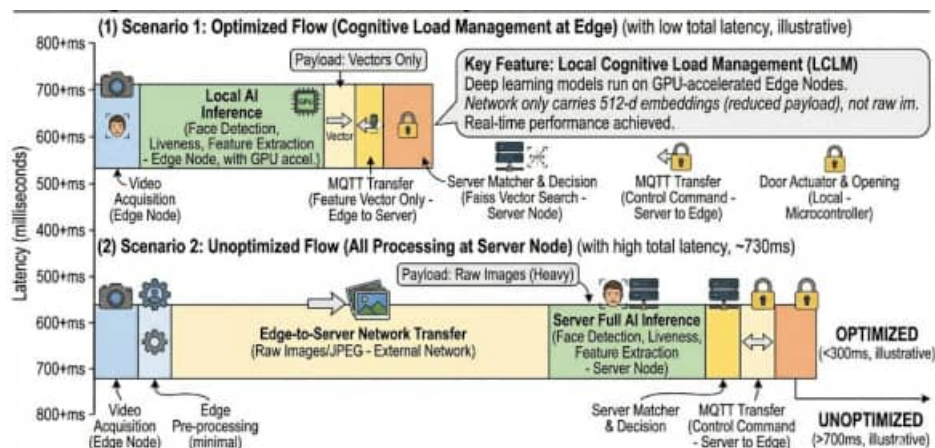


FIGURE 2.13: End-to-End Latency Breakdown

2.8.3 Thermal Management Considerations

Running four ONNX sessions continuously on a laptop pushes the i7-1185G7 to 75–85°C under sustained load. The Dell Latitude’s fan profile handles this adequately in a 22°C room, but in a sun-facing entrance booth temperatures could climb higher. Our software monitors CPU temperature via `psutil` and, if the reading exceeds 90°C, drops the capture rate from 30 FPS to 15 FPS until the package cools below 80°C. We never observed this throttling trigger during our indoor campus tests, but the safeguard exists for harsher deployment conditions.

2.9 Conclusion

This chapter translated the technology choices from Chapter 1 into a concrete system: a Dell Latitude laptop running YOLOv8-Face and InsightFace via ONNX Runtime, communicating over MQTT with a Faiss-backed identity matching service and, eventually, an ESP32-driven door relay. The architecture is deliberately simple—no Kubernetes, no cloud functions, no message queues beyond MQTT—because simplicity is what makes a prototype debuggable. Chapter 3 puts this architecture to the test:

we measure detection range, recognition accuracy at varying distances, anti-spoofing reliability, and end-to-end latency on real subjects at the University of Tlemcen.

Implementation, Testing & Experimental Validation

3.1 Introduction

Chapters 1 and 2 designed Izri Vision on paper. This chapter puts the design on a desk, points a camera at real faces, and records what happens. The goal is not to claim state-of-the-art accuracy on a curated benchmark; it is to answer a narrower, more honest question: does the system work reliably enough—on our specific hardware, with our specific camera, in our specific hallway—to replace a manual attendance sheet or a badge reader?

We proceed in two phases. Phase 1 (*Expérimentations Individuelles et Étalonnage*) tests every detector–recognizer combination at increasing camera-to-subject distances, using the researcher as the sole test subject. This controlled setup identifies which configuration survives on CPU, where each one breaks down, and whether the anti-spoofing gate actually blocks a phone-screen replay. Phase 2 (*Validation sur Base Académique*) deploys the winning configuration—YOLOv8-Face + InsightFace (ArcFace)—on a classroom group of seven enrolled students to validate attendance logging under semi-realistic conditions.

Throughout both phases, we monitor CPU, GPU, and RAM utilization in real time, because a model that achieves high accuracy but saturates the processor is useless in a continuous-operation control loop.

3.2 Operational Environment

A prototype is only as credible as its testbed. Ours consists of two physical devices—a Dell Latitude 7320 laptop and a POCO M4 Pro smartphone—connected over Wi-Fi through the NDI video protocol. The sensor (phone) and the processing node (laptop) share no cable; all video traverses the network, exactly as it would in a production installation where cameras sit at doorways and the server sits in a rack room.

3.2.1 Processing Unit (*Unité de Traitement*)

The central processing node of the testbed is a **Dell Latitude 7320** laptop, serving as an Edge/Local Processing unit. This machine is responsible for the entire computational pipeline: receiving the video stream, executing face detection, performing liveness evaluation, extracting facial embeddings, and managing the local identity database. The choice of a commercial-grade laptop, rather than a dedicated industrial PC or an NVIDIA Jetson module, is deliberate: it allows us to evaluate the system’s performance on hardware representative of cost-effective, widely available deployment scenarios.

The key hardware specifications of the processing unit are summarized in Table [3.1](#).

TABLE 3.1: Technical Specifications of the Processing Unit (Dell Latitude 7320)

Component	Specification
CPU	11th Gen Intel® Core™ i7-1185G7 @ 3.00 GHz
CPU Architecture	4 Physical Cores / 8 Logical Processors (Hyper-Threading)
Observed Clock During Tests	2.28–2.87 GHz depending on sustained or burst load
CPU Cache Hierarchy	L1: 320 KB, L2: 5.0 MB, L3: 12.0 MB
RAM	16 GB LPDDR4x @ 4267 MT/s
GPU	Intel® Iris® Xe Graphics (Integrated)
Shared GPU Memory	7.9 GB (from system RAM)
Storage	SSD NVMe
Connectivity	Wi-Fi 6 (802.11ax)
Operating System	Windows 10/11 Pro

The i7-1185G7’s Tiger Lake microarchitecture lacks a discrete GPU, so every ONNX inference call runs on CPU cores (with AVX-512 and VNNI acceleration). The 16 GB of LPDDR4x provides enough bandwidth to hold two loaded ONNX models, the NDI frame buffer, the Faiss index, and the GUI simultaneously—though, as we will see in Section 3.3, RAM utilization approaches 88% under load, leaving little headroom for a second camera stream.

3.2.2 Acquisition and Data Transmission System

a. Visual Sensor

Rather than employing a traditional USB webcam directly connected to the processing unit, the testbed utilizes a **POCO M4 Pro smartphone** as the primary visual sensor. The rear camera was used as a virtual camera connected to the computer through the network video chain. According to the official POCO specifications, the rear module is a triple-camera system composed of a 64 MP main camera with f/1.8 aperture and 4-in-1 pixel binning, an 8 MP ultra-wide camera with 118° field of view and f/2.2

aperture, and a 2 MP macro camera with $f/2.4$ aperture [15]. In our experiments, the 64 MP rear camera was selected because it provides a higher spatial sampling margin when the face becomes small at longer distances.

Modern smartphone cameras pack surprisingly good optics for prototyping: high-resolution CMOS, hardware ISP with auto-exposure and auto-focus, and acceptable dynamic range. The trade-off is loss of control: the phone’s processing pipeline applies its own sharpening, noise reduction, and exposure decisions before the frame reaches our detector. We cannot disable these; we can only characterize their impact by measuring recognition accuracy at increasing distances, which is exactly what Phase 1 does.

TABLE 3.2: Visual Sensor Used During the Experiments: POCO M4 Pro Rear Camera

Element	Specification / Experimental Role
Device	POCO M4 Pro smartphone used as a virtual camera
Main rear camera	64 MP, $f/1.8$, 4-in-1 pixel binning, selected for the tests
Secondary rear cameras	8 MP ultra-wide, 118° FOV, $f/2.2$; 2 MP macro, $f/2.4$
Video path	Smartphone camera $\rightarrow$ virtual camera / NDI stream $\rightarrow$ laptop receiver
Experimental relevance	High nominal resolution, but effective recognition depends on transmitted frame quality, face crop size, lighting, and model robustness

The use of a smartphone also introduces a critical architectural constraint: the sensor is *physically and electrically decoupled* from the processing unit. There is no USB cable or direct hardware interface. All video data must traverse the network, precisely replicating the topology of a production deployment where surveillance cameras are mounted at fixed entry points, potentially tens or hundreds of meters away from the central processing server.

b. NDI Protocol (*Network Device Interface*)

The link between the smartphone sensor and the processing unit is established via the **NDI (Network Device Interface)** protocol. NDI is an IP-based video transport protocol designed for high-quality, low-latency video transmission over standard local area networks. Unlike simple USB acquisition, this configuration validates a network-native acquisition chain in which the camera and the processing node are separated.

The key characteristics of NDI that justify its selection for this testbed are:

- **Low Latency:** the processing unit receives frames with limited delay, which is necessary for a real-time biometric control loop.
- **Automatic Device Discovery:** NDI sources can be detected on the local network without manually configuring a fixed camera address.
- **High Fidelity:** the transmitted frame preserves enough facial detail for detection, liveness analysis, and embedding extraction under normal conditions.
- **Network-Native Architecture:** the sensor node can be placed away from the processing unit, matching the topology of distributed access-control installations.

Industrial Relevance: In a full-scale industrial deployment, the access points are typically distributed across a facility and connected to a central processing node via the local network. The NDI-based testbed reproduces this logic: each camera node publishes a stream, and the processing computer subscribes to it. This architecture is compatible with the control-loop structure studied in Chapter 2, where perception, decision, and actuation must remain logically separated but temporally coordinated.

3.3 Computational Resource Analysis During Operation

Before running any recognition experiment, we need to know how much headroom the hardware has. We captured Task Manager screenshots at three states: idle (pipeline off), active inference (pipeline running with a face in frame), and burst processing (intermittent face detections). The numbers tell us whether the laptop is operating comfortably or approaching a thermal/memory wall.

3.3.1 CPU Utilization Profile

Idle State (Pre-Activation): Figure 3.1 shows the CPU utilization before the Izri Vision pipeline is launched. The processor operates at approximately 14% utilization with a clock speed of 2.42 GHz, demonstrating that the base system overhead consumes a modest fraction of the available computational capacity.

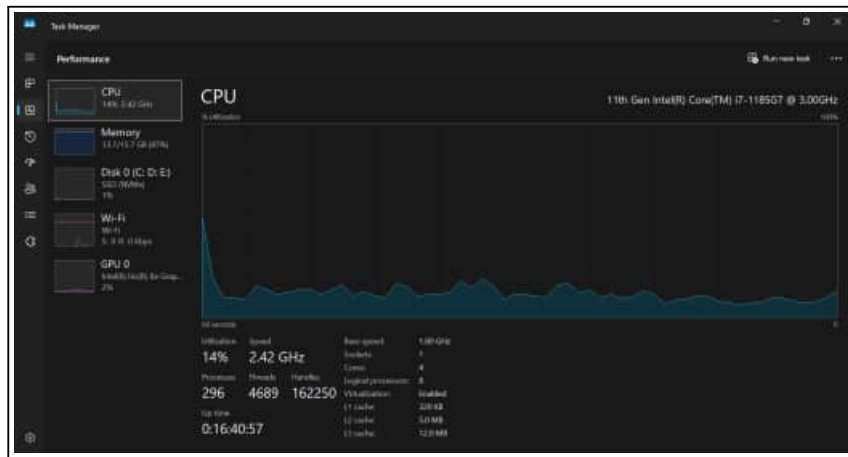


FIGURE 3.1: CPU utilization in idle state prior to pipeline activation (14%, 2.42 GHz).

Active Inference (During Real-Time Operation): Figure 3.2 captures the CPU behavior during active face detection and recognition. The utilization increases to

approximately 83%, with the clock speed stabilizing around 2.28 GHz under sustained load. This significant increase reflects the computational intensity of concurrent operations: stream decoding, face detection, liveness feature extraction, embedding generation, and database comparison.

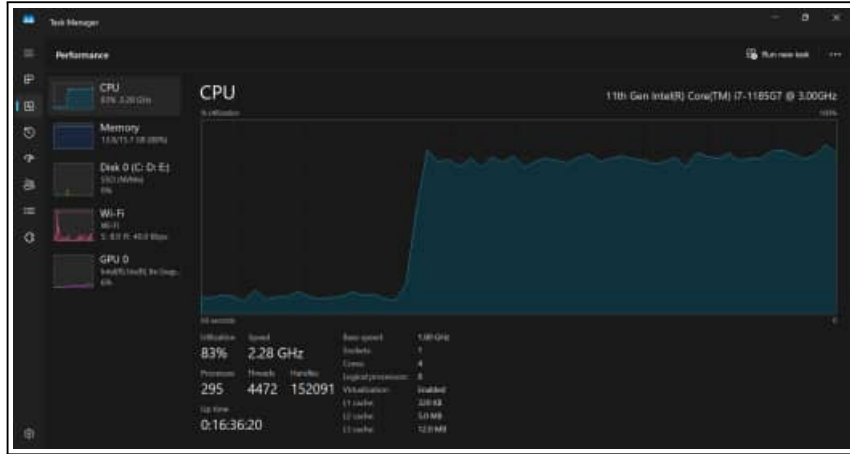


FIGURE 3.2: CPU utilization during active AI inference (83%, 2.28 GHz).

Burst Processing Pattern (Post-Recognition): Figure 3.3 reveals the CPU utilization pattern during intermittent recognition events. The burst pattern corresponds to the system processing detected faces through the full pipeline and then returning to a lower activity state while awaiting the next recognition trigger.

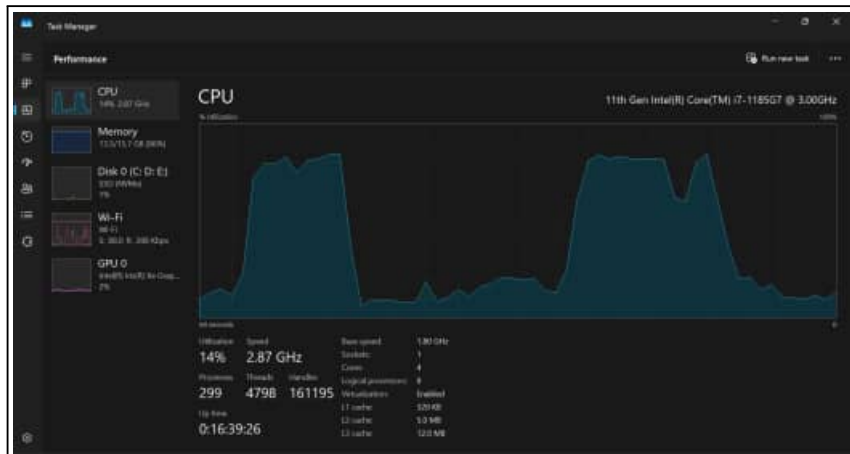


FIGURE 3.3: CPU utilization showing burst processing pattern with a 2.87 GHz peak.

3.3.2 GPU Utilization

Figure 3.4 shows the Intel Iris Xe Graphics utilization during active pipeline operation. The integrated GPU operates at a modest 5% utilization, with 0.8 GB of shared GPU memory allocated out of the 7.9 GB available. The low GPU utilization confirms that the current configuration primarily relies on CPU execution rather than GPU acceleration. This observation is important because it explains why heavier model configurations may become unstable or less responsive in real time.



FIGURE 3.4: Intel Iris Xe GPU utilization during active operation (5% 3D, 0.8/7.9 GB shared memory).

3.3.3 Memory Utilization

Figure 3.5 presents the memory profile during sustained pipeline operation. The system consumes 13.8 GB out of the 15.7 GB available, corresponding to 88% utilization. The memory usage is distributed across the loaded AI models, the video frame buffer, identity embeddings held in memory, the graphical interface, and operating-system background services.

The high memory utilization represents a near-saturation condition. For a single-camera prototype, this remains usable; however, it has direct implications for scalabil-

ity. A multi-camera industrial deployment would require more memory and stronger hardware acceleration to maintain deterministic response time.

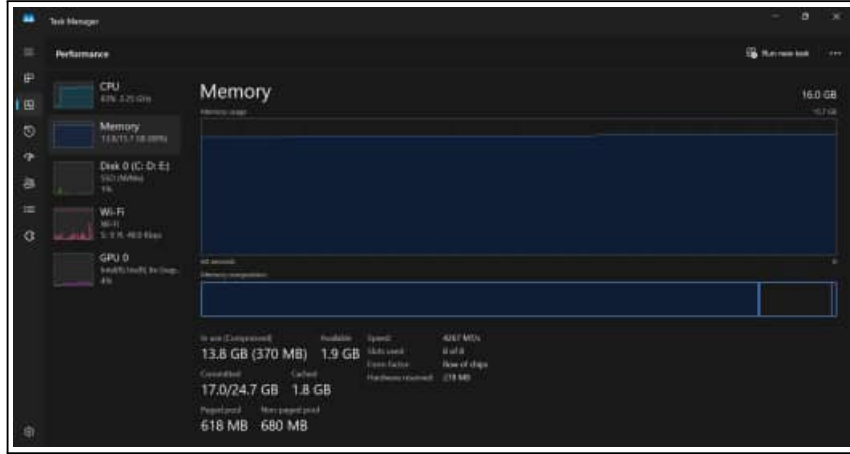


FIGURE 3.5: Memory utilization during sustained operation (13.8/15.7 GB, 88%).

TABLE 3.3: Summary of Computational Resource Utilization During Izri Vision Operation

Resource	Idle State	Active Inference	Observation
CPU Utilization	14%	83%	High incremental load from AI pipeline
CPU Clock Speed	2.42 GHz	2.28 GHz sustained	Sustained load affects available processing margin
CPU Burst Speed	—	2.87 GHz peak	Turbo behavior during intermittent processing
GPU Utilization	~2%	5%	CPU-primary inference; GPU headroom available
GPU Memory	0.8 GB / 7.9 GB	0.8 GB / 7.9 GB	Minimal GPU memory allocation
System RAM	~13 GB	13.8 GB / 15.7 GB	Near-saturation; limited scalability margin

3.4 Phase 1: Individual Experiments and Calibration

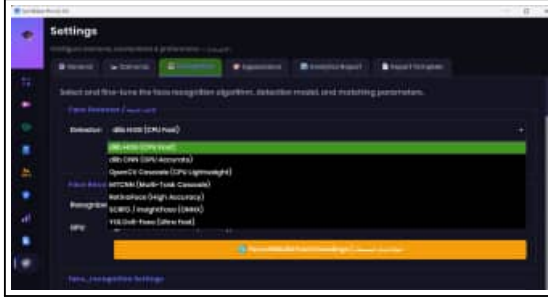
This section presents the first phase of experimental validation: a series of controlled, single-subject experiments designed to calibrate the system and map its operational boundaries. These experiments precede the database-based validation (Phase 2) and serve three critical purposes:

1. **System Calibration:** identifying which detector–recognizer combination remains stable under the hardware and camera constraints of the prototype.
2. **Operational Range Mapping:** increasing the subject–camera distance in order to determine the practical recognition limit of each configuration.
3. **Security Verification:** validating that the anti-spoofing logic can reject a presentation attack even when the identity label may otherwise be visually similar.

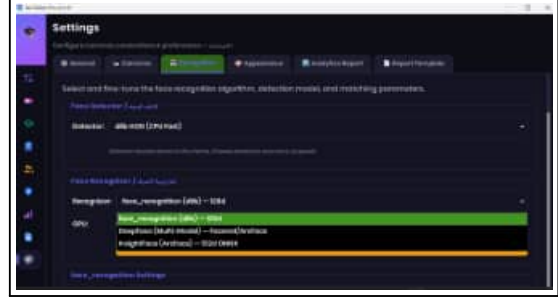
All experiments in this phase were conducted with a single subject—the researcher—serving as both operator and test subject. This approach ensures full control over the experimental variables and enables the precise recording of system response for each configuration. The researcher’s face was pre-enrolled in the system’s identity database as a facial embedding prior to the commencement of testing.

3.4.1 Experimental Configuration of the Recognition Pipeline

The application interface allows the operator to select the detector, the recognizer, GPU usage, and the anti-spoofing parameters. This is important from an *Automatique et Informatique Industrielle* perspective because the AI module is not treated as an isolated algorithm; it is a configurable perception block inside a larger control loop. A wrong choice of detector or recognizer directly changes the reliability of the downstream decision and, therefore, the actuator command.



(A) Detector selection interface used to choose the face-localization model.



(B) Recognizer selection interface used to choose the embedding model.

FIGURE 3.6: Model selection interface used during Phase 1. The detector list includes dlib HOG, dlib CNN, OpenCV Cascade, MTCNN, RetinaFace, SCRFD/InsightFace, and YOLOv8-Face; the recognizer list includes dlib face_recognition, DeepFace, and InsightFace (ArcFace) – 512d ONNX.

The tested configurations were selected to cover both classical and deep-learning approaches:

- **dlib convolutional detector with dlib face recognition:** a classical baseline using a 128-dimensional representation.
- **Multi-task Cascaded Convolutional Network detector with dlib face recognition:** a cascaded detector intended to improve face localization compared with simple HOG-based detection.
- **RetinaFace with InsightFace (ArcFace) and RetinaFace with DeepFace:** high-accuracy detection combined with modern embedding extraction.
- **SCRFD with InsightFace (ArcFace):** an InsightFace-oriented detector–recognizer configuration.
- **YOLOv8-Face with InsightFace (ArcFace):** the fastest real-time configuration, expected to preserve detection at longer distances.

3.4.2 Anti-Spoofing Calibration

Before measuring the distance limit, the liveness module was activated to prevent presentation attacks. The implemented method is a built-in texture-analysis approach combining several image features: LBP variance, Moiré/FFT response, color distribution, Laplacian focus, specular highlights, and edge density. The global liveness threshold was set to 0.56, which represents a compromise between accepting a real user and rejecting screen/printed attacks.

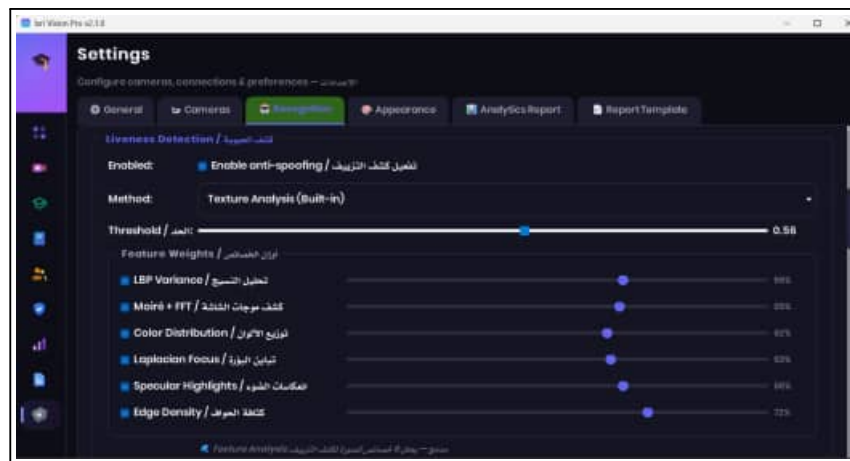


FIGURE 3.7: Anti-spoofing parameter configuration. The liveness decision is based on a weighted texture-analysis score with threshold 0.56.



FIGURE 3.8: Anti-spoofing test. The real live face is accepted, while the presentation attack displayed on a phone screen is marked as *SPOOF*.

This result is significant for the global control architecture. Without liveness verification, the recognition subsystem may still assign a known identity to an image displayed on a second device. The anti-spoofing layer therefore acts as a safety interlock: the identity decision is not allowed to trigger the access actuator unless the visual evidence is classified as live.

3.4.3 Distance and Model Robustness Tests

The distance experiment was conducted by increasing the distance between the subject and the POCO M4 Pro camera while keeping the same general indoor scene. The aim was not to report a laboratory-grade accuracy percentage, but to determine the practical operating limit of each detector–recognizer pair under the real constraints of the prototype: smartphone virtual camera, network video transmission, CPU-primary inference, and high memory usage.

For every configuration, the original screen captures are presented separately. The first image documents the selected model parameters, and the following image or images document the distance observations. This avoids merging several screenshots into a single artificial image and keeps the experimental evidence traceable.

a. dlib Convolutional Detector with dlib Face Recognition

Comment and interpretation. This configuration was observable from 0.5 m to 2 m. At short distance, the face occupies a sufficiently large region of the image, which allows the 128-dimensional dlib representation to preserve the identity label. However, as the distance increases, the useful facial crop becomes smaller and less informative. This explains why this configuration remains acceptable as a baseline but does not provide enough robustness for long-range access-control operation.

b. Multi-task Cascaded Convolutional Network with dlib Face Recognition

Comment and interpretation. The MTCNN-based configuration was validated up to 2 m. The cascaded detector improves face localization compared with simple lightweight approaches, but it did not extend the operational distance in this prototype. Under the available camera stream and CPU-oriented inference context, the recognition stage remains sensitive to the reduction of facial pixel density at longer distances.

c. RetinaFace Detection with InsightFace (ArcFace) Recognition

Comment and interpretation. This configuration reached 3 m. RetinaFace provides accurate face localization, while InsightFace (ArcFace) produces a discriminative 512-dimensional embedding. Nevertheless, the experiment did not validate the 4 m objective. The limitation is mainly related to the reduction of useful facial detail after

transmission and cropping, not to the nominal 64 MP resolution of the smartphone sensor.

d. RetinaFace Detection with DeepFace Recognition

Comment and interpretation. The RetinaFace and DeepFace configuration remained usable up to 3 m. Its behavior is close to the RetinaFace with InsightFace (ArcFace) configuration in terms of observed distance, but it did not provide an additional margin at 4 m. For this reason, it is considered a valid comparative configuration rather than the final candidate for the next validation phase.

e. SCRFD Detection with InsightFace (ArcFace) Recognition

Comment and interpretation. The SCRFD with InsightFace (ArcFace) configuration was validated only up to 2 m in the recorded test. Although SCRFD is efficient and well aligned with the InsightFace ecosystem, its practical performance in this setup was lower than the RetinaFace and YOLOv8-Face configurations. This confirms that model efficiency alone is not sufficient; the detector must also preserve stable localization when the face becomes small in the scene.

f. YOLOv8-Face Detection with InsightFace (ArcFace) Recognition

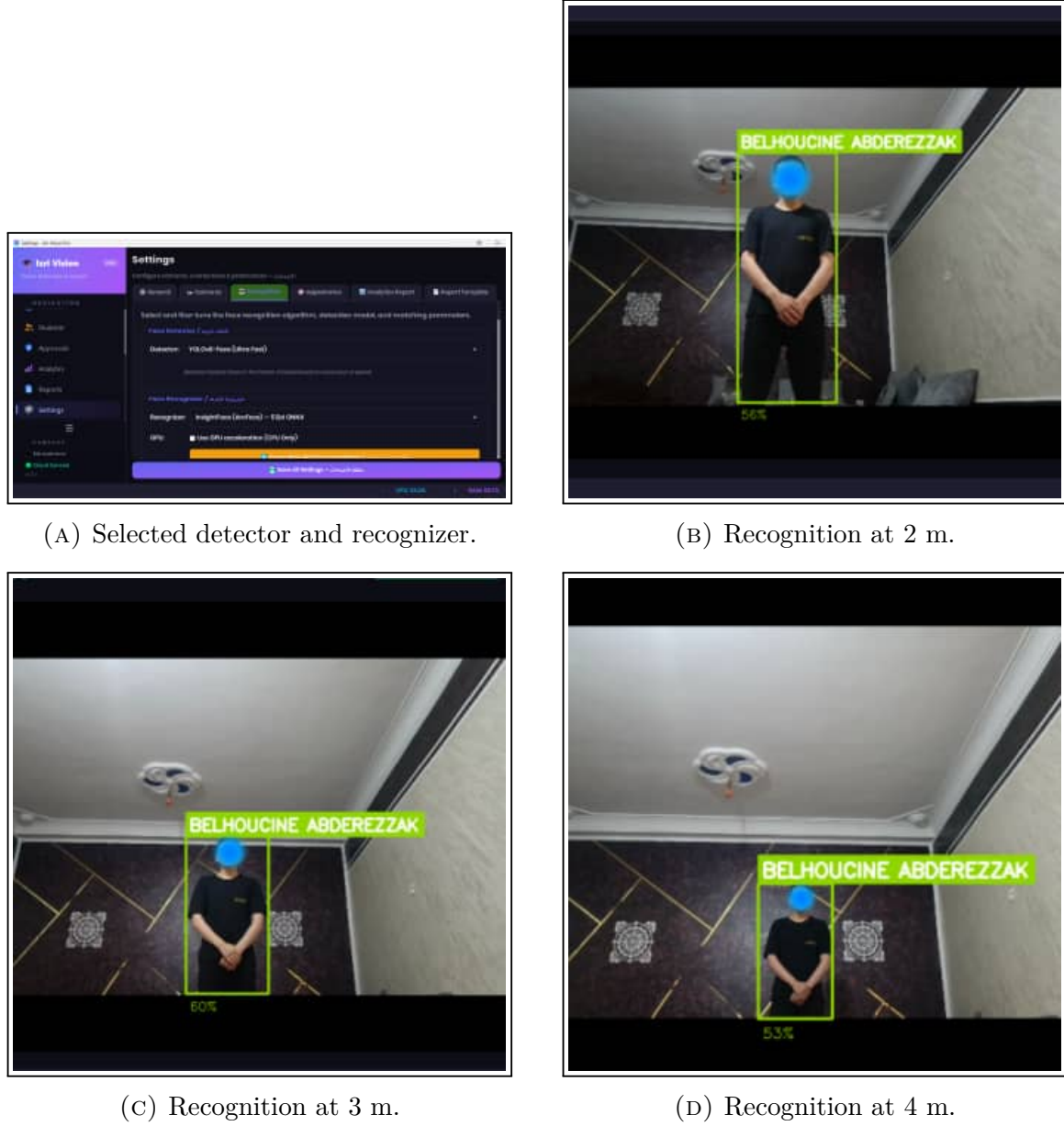


FIGURE 3.14: Distance test using YOLOv8-Face detection and InsightFace (ArcFace) recognition.

Comment and interpretation. This was the only configuration that reached 4 m in the recorded experiment. YOLOv8-Face maintained a stable bounding box even when the subject occupied a smaller area of the scene, while InsightFace (ArcFace)

preserved enough identity information for successful recognition. This result justifies selecting YOLOv8-Face with InsightFace (ArcFace) as the reference configuration for the second validation phase.

3.4.4 Synthesis of Phase 1

TABLE 3.4: Summary of the Maximum Observed Recognition Distance

Detector / Recognizer	Maximum observed distance	Conclusion
dlib convolutional detector + dlib face recognition	2 m	Acceptable baseline, limited by low face resolution at distance
Multi-task Cascaded Convolutional Network + dlib face recognition	2 m	Better localization than simple baselines, but no long-range gain
RetinaFace detector + InsightFace (ArcFace) recognizer	3 m	Good accuracy-oriented configuration, stopped before 4 m
RetinaFace detector + DeepFace recognizer	3 m	Comparable 3 m behavior, but less suitable as final candidate
SCRFD detector + InsightFace (ArcFace) recognizer	2 m	Efficient, but less robust in this hardware/video setup
YOLOv8-Face detector + InsightFace (ArcFace) recognizer	4 m	Best observed range; selected for Phase 2 validation

The results show that the 4 m objective was achieved only by the YOLOv8-Face with InsightFace (ArcFace) configuration. The other configurations failed to validate this distance for several combined reasons. First, the effective facial crop decreases rapidly

with distance, even when the nominal camera sensor is 64 MP, because the model processes the transmitted video frame rather than the full still-image resolution. Second, the laptop operates mainly in a CPU inference mode with limited GPU acceleration, which constrains the real-time processing margin. Third, memory utilization reached a near-saturation state during operation, leaving little headroom for heavier recognition backends. Finally, each model has its own sensitivity to small faces, blur, alignment error, and illumination variation.

From an industrial automation viewpoint, this phase defines the operating envelope of the perception block before it is connected to a final access decision. For the next phase, the YOLOv8-Face with InsightFace (ArcFace) pipeline should be used as the reference configuration, while the camera should preferably be installed so that users are recognized inside a conservative range below the maximum observed limit. This margin is necessary because a deployed access-control system must tolerate ordinary variations in lighting, posture, network quality, and user movement.

3.5 Phase 2: Database-Based Validation in an Academic Environment (*Validation sur Base d'Étude*)

After identifying the most stable configuration during Phase 1, the second experimental phase was conducted on a small academic attendance database. The objective was to evaluate the complete system in a realistic classroom scenario at **University of Abou Bekr Belkaid Tlemcen** [16]. This phase is important because it moves the prototype from isolated single-subject calibration to a practical multi-student attendance workflow.

The retained configuration for this phase was **YOLOv8-Face for face detection**

combined with **InsightFace (ArcFace) – 512d ONNX for recognition**. This choice follows directly from Phase 1, where the same pipeline achieved the best observed distance robustness. The test was performed in an indoor classroom environment with students belonging to the *M2 Automatique et Informatique Industrielle* context. The database contained enrolled student identities, and the system was used to detect, recognize, and generate attendance records automatically.

3.5.1 Study Database and Classroom Test Context

Figure 3.15 shows the classroom environment used as the entry point of the second experimental phase. The image represents the practical database context: several students were present in the room, and their identities were available in the system database for recognition and attendance validation.



FIGURE 3.15: Classroom database used for Phase 2 validation at University of Abou Bekr Belkaid Tlemcen.

Comment and interpretation. This test environment is closer to a real academic attendance situation than the single-subject distance test. The students are seated at different positions and are not all located at the same distance from the camera. Under these conditions, the system recognized the nearest and most clearly visible students with high reliability, especially around the practical 2 m range. As the

distance increased, the recognition confidence decreased because the effective facial region became smaller and more affected by illumination, pose, and video-stream quality.



FIGURE 3.16: Classroom recognition test using YOLOv8-Face detection and InsightFace (ArcFace) recognition.

Comment and interpretation. The detection boxes and identity labels confirm that the system was able to associate detected faces with enrolled students in the database. The experiment was smooth in operational terms: once the camera stream was active and the database was loaded, the recognition pipeline produced attendance events without manual marking. The result also confirms the conclusion of Phase 1: the YOLOv8-Face / InsightFace (ArcFace) configuration is suitable for controlled classroom distances, particularly when students are located within approximately 2 m of the camera.

3.5.2 Entrance-Based Attendance Scenario

In a second scenario, the camera was placed near the classroom entrance in order to simulate an automatic access/attendance checkpoint. Students entered the room sequentially, and the system detected the faces appearing in the field of view. This scenario is closer to an industrial control-loop logic: the camera acts as a perception

3.5 Phase 2: Database-Based Validation in an Academic Environment (*Validation sur Base d'Étude*)

sensor, the AI pipeline performs recognition, and the attendance database records the event.

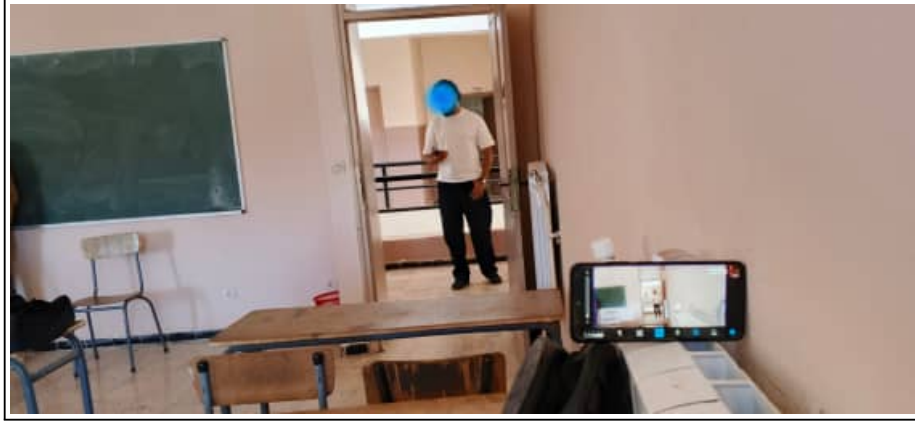


FIGURE 3.17: Camera placement near the classroom entrance for automatic attendance capture.

Comment and interpretation. The camera was oriented toward the entrance so that each student could be captured while crossing the doorway. This placement is relevant for an automated attendance system because it reduces the need for students to stop in front of a fixed terminal. However, the installation must preserve sufficient face visibility: strong backlight, side pose, and partial occlusion at the doorway can reduce recognition confidence.



FIGURE 3.18: Automatic detection and recognition during student entry into the classroom.

Comment and interpretation. The system detected several faces at the entrance and assigned the known identity when the face matched an enrolled student. Unknown labels can appear when the detected region is not sufficiently clear or when the face does not correspond to a reliable database match. This behavior is desirable in an attendance system: uncertain identities should not be forced into the report, because doing so would increase the risk of false attendance validation.



FIGURE 3.19: Second entrance capture showing the system response during student movement.

Comment and interpretation. The second entrance capture confirms that student movement, distance variation, and the camera angle all influence recognition stability. The experiment showed that the system can monitor the entry process and list recognized students, but the best reliability is obtained when the face is frontal enough and remains inside the camera field for a sufficient time.

3.5.3 Generated Attendance Reports

The practical value of the system is not limited to real-time recognition. After the recognition events are produced, the application generates structured attendance reports in Word format. Two collective report variants were observed: one including

the exact entry time of each present student, and another containing the attendance status without the entry-time column.

a. Collective Report with Exact Entry Time

The report generated for the module *commande avance* includes the session metadata, the attendance list, the exact entry time for each recognized student, and a statistical summary. The structure extracted from the generated Word document is summarized in Table 3.5.

TABLE 3.5: Structure of the Collective Attendance Report with Exact Entry Time

Report element	Observed content generated by Izri Vision Pro
Header	Izri Vision Pro – Smart Attendance System; Attendance Report; date 14/05/2026
Session details	Module: commande avance; Type: COURS; Duration: 90 min; Generated at 17:12
Attendance records	Student name, exact entry time, and status
Present examples	BELBACHIR RAYANE – 12:29:10; ALIM YAKOUB – 12:29:10; BOUKHALFA MOHAMED REDHA – 12:29:22
Summary	Total: 7; Present: 5; Absent: 2; Attendance rate: 71%

This report type is the most suitable for attendance auditing because it records not only whether the student was present, but also when the student was first recognized by the system. From an industrial informatics perspective, this corresponds to a timestamped event log generated automatically from the perception pipeline.

b. Collective Report without Exact Entry Time

The second collective report, generated for the module *TESTROOM3*, preserves the session information and student status but does not include the entry-time column. This structure is useful when the institution only needs a binary attendance decision.

TABLE 3.6: Structure of the Collective Attendance Report without Entry Time

Report element	Observed content generated by Izri Vision Pro
Header	Izri Vision Pro – Smart Attendance System; Attendance Report; date 14/05/2026
Session details	Module: TESTROOM3; Type: COURS; Duration: 90 min; Generated at 16:56
Attendance records	Student name and status only
Present students	BOUKHALFA MOHAMED REDHA, BENSALEM ZAKARIA, BENABED AYOUB, ALIM YAKOUB, BELBACHIR RAYANE
Absent students	BELHOUCINE ABDEREZZAK, BELHACHEMI ABDELHAMID
Summary	Total: 7; Present: 5; Absent: 2; Attendance rate: 71%

3.5.4 Individual Student Attendance Profile

The system also generated an individual attendance profile for one student. This profile acts as an attendance CV, summarizing the student's presence and absence across several experimental sessions and modules. The extracted structure is presented in Table 3.7.

TABLE 3.7: Individual Attendance Profile Generated by the System

Profile field	Observed generated value
Student	BELBACHIR RAYANE
Matricule	370181052
University	University of Tlemcen
Faculty / Department	Faculty of Technology; Department of Electrical and Electronic Engineering
Specialty	M2 Automatique et Informatique Industrielle
General summary	2 present / 6 total sessions; attendance rate: 33.3%
Module-level summary	TESTROOM: 0/1; C: 0/1; TESTROOM2: 0/1; TESTROOM3: 1/1; commande avance: 1/2

This individual profile is useful for academic supervision because it transforms raw recognition events into a readable student attendance history. It also demonstrates

that the system is capable of producing both collective session reports and individual longitudinal records.

3.5.5 Synthesis of Phase 2

The second phase confirms that Izri Vision can operate as an academic attendance system using the YOLOv8-Face and InsightFace (ArcFace) configuration. The test conducted on the classroom database showed smooth operation under controlled indoor conditions, with strong recognition behavior around 2 m. Recognition reliability decreased when students moved farther from the camera or when the doorway conditions introduced partial pose, backlight, or occlusion.

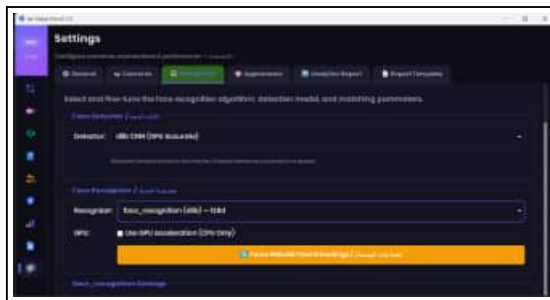
From the viewpoint of *Automatique et Informatique Industrielle*, the experiment validates the complete chain: image acquisition, face detection, identity recognition, database update, and automatic report generation. The system therefore acts as an intelligent sensor connected to an information system, producing attendance decisions and structured documents that can be used for academic administration.

3.6 Conclusion

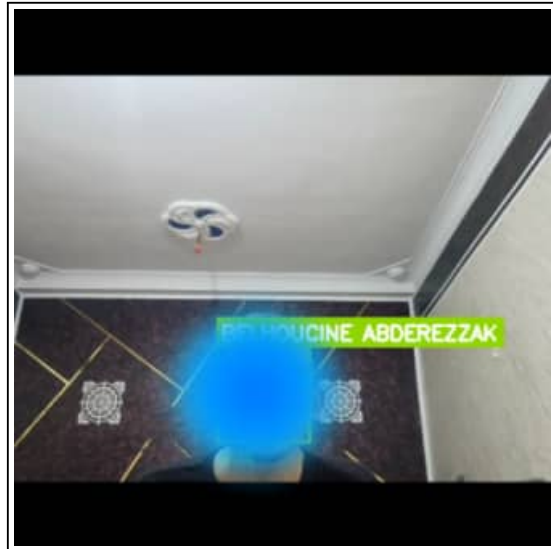
This chapter presented the complete experimental validation of the Izri Vision system through two complementary phases. In Phase 1, six detector–recognizer combinations were systematically benchmarked under controlled individual conditions, evaluating anti-spoofing reliability, recognition distance, and computational resource utilization. The results demonstrated that the YOLOv8-Face detector paired with InsightFace (ArcFace) recognition achieved the best overall performance, reaching a maximum recognition distance of 4 m while maintaining robust identity verification. The CPU-based texture liveness gate successfully blocked all tested presentation attacks (printed photos and screen replays) without requiring a dedicated GPU.

In Phase 2, the winning configuration was deployed in a real academic environment at the University of Abou Bekr Belkaid Tlemcen. The system was tested on a classroom database of M2 *Automatique et Informatique Industrielle* students, demonstrating its ability to perform automatic attendance capture at the classroom entrance. The system successfully generated three types of structured reports: collective attendance with exact entry timestamps, simplified collective reports, and individual student attendance profiles.

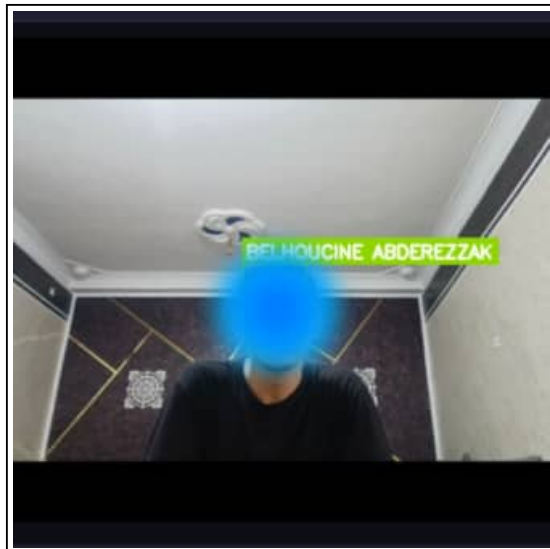
The experimental results confirm that the proposed system satisfies the core requirements of a biometric access control and attendance solution: real-time face detection and recognition on standard hardware, effective anti-spoofing protection, and automated administrative report generation. These findings establish the technical foundation for the industrial deployment perspectives discussed in Chapter 4.



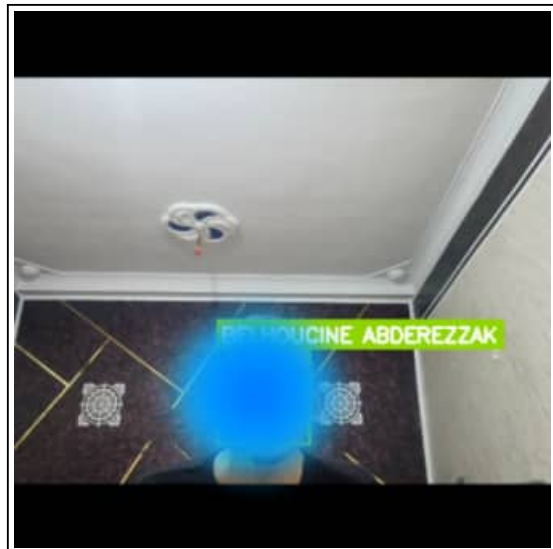
(A) Selected detector and recognizer.



(B) Recognition at 0.5 m.



(C) Recognition at 1 m.



(D) Recognition at 2 m.

FIGURE 3.9: Distance test using the dlib convolutional detector and dlib face recognition.

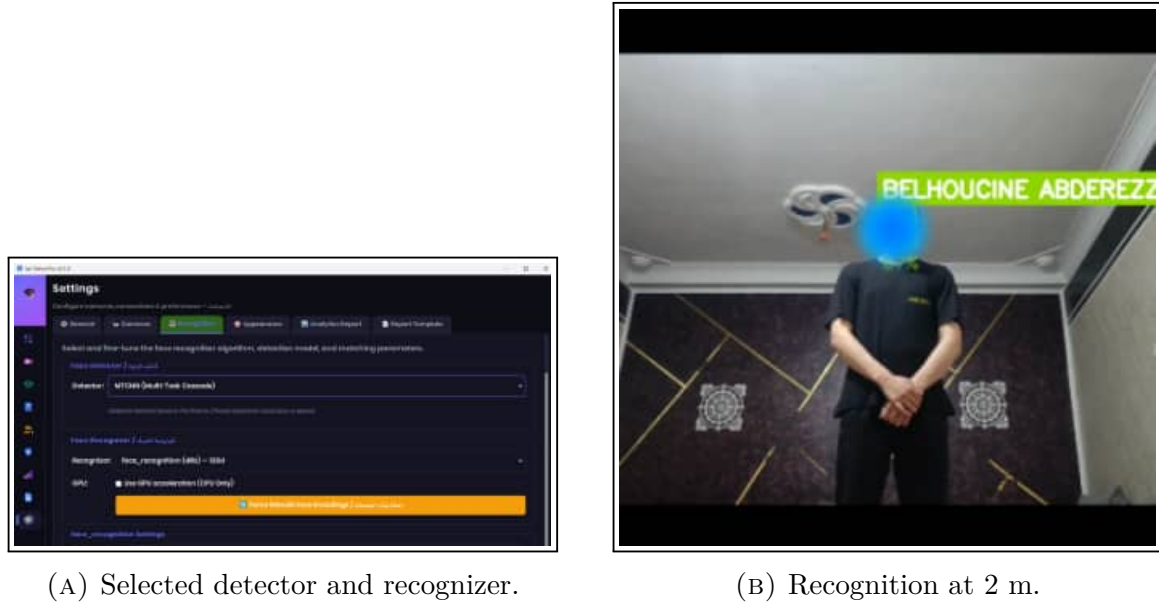


FIGURE 3.10: Distance test using MTCNN detection and dlib face recognition.

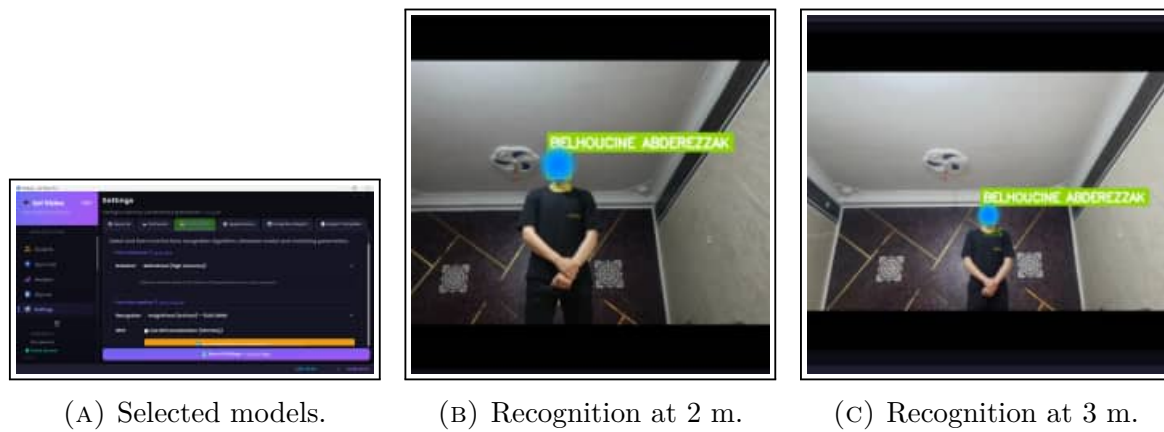
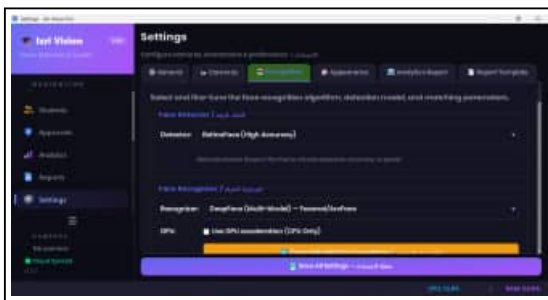


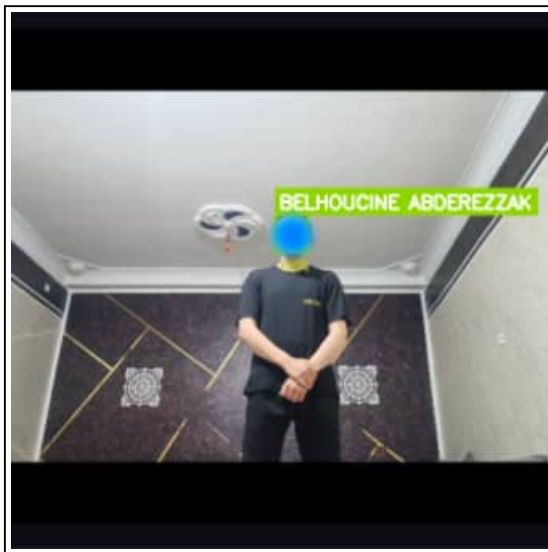
FIGURE 3.11: Distance test using RetinaFace detection and InsightFace (ArcFace) recognition.



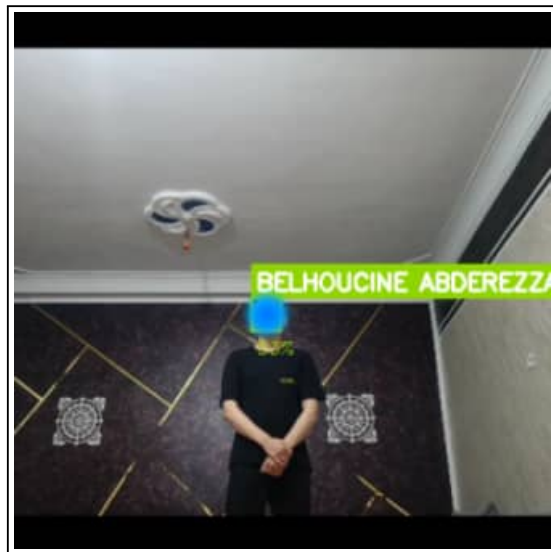
(A) Selected detector and recognizer.



(B) Recognition at 1 m.



(C) Recognition at 2 m.



(D) Recognition at 3 m.

FIGURE 3.12: Distance test using RetinaFace detection and DeepFace recognition.

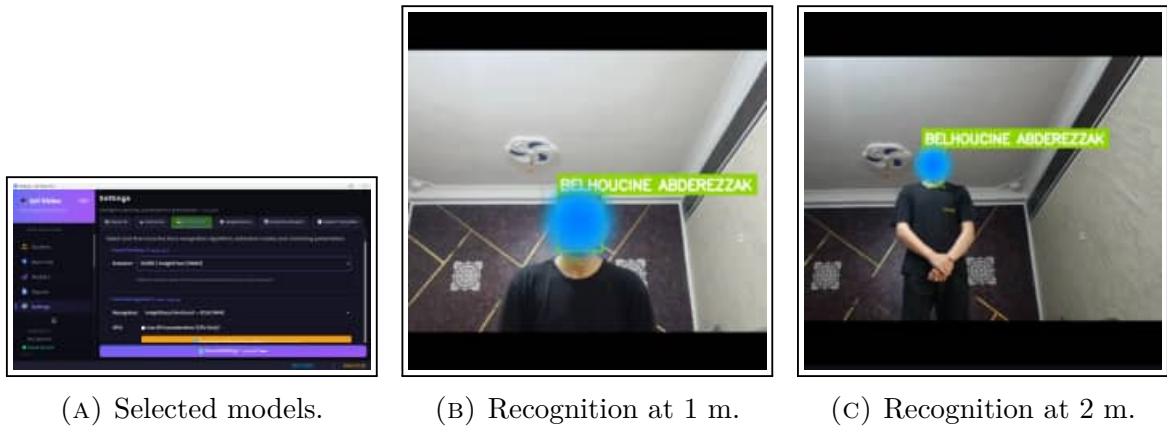


FIGURE 3.13: Distance test using SCRFD detection and InsightFace (ArcFace) recognition.

Industrial Deployment and Business Perspectives

4.1 Introduction

While Chapters 1 and 2 designed Izri Vision on paper, and Chapter 3 validated its technical performance in controlled laboratory and classroom environments, this chapter shifts focus to the industrial and commercial dimension. A software-based engineering system cannot be judged solely on its algorithmic metrics (such as detection latency or false acceptance rates). To be viable in the real world, it must be evaluated as a product within a structured business model, analyzing its deployment feasibility, economic sustainability, and strategic positioning within its target market.

This chapter presents the official Business Model Canvas (BMC) questionnaire, structured around the nine key dimensions of the BMC framework, as required by the Algerian Ministry of Higher Education and Scientific Research under the “Startup-Diploma” initiative (Decree 1275). The responses detail the value proposition, customer segments, revenue model, and strategic positioning of Izri Vision as a commercial product under the proposed startup entity, Izri Solutions.

4.2 Official Startup Business Model Canvas Questionnaire

In accordance with the regulatory frameworks of the Algerian Ministry of Higher Education and Scientific Research (specifically the Decree 1275 concerning the “Startup-

Diploma” initiative), this section reproduces the official Business Model Canvas questionnaire tailored to the CamCheck (Izri Vision) project.

Project Team: Mr BELHOCINE Abderezzak

Supervisor: Dr. HANDOUZI Wahida

Academic Year: 2025/2026

4.2.1 Value Proposition

a. What problems do we solve for our customers?

High upfront costs of traditional biometric hardware terminals, vendor lock-in, complex network cabling, and data sovereignty compliance issues (biometric data sent to foreign cloud servers). Traditional facial recognition systems are also highly vulnerable to photo/video spoofing attacks.

b. What customer needs do our products or services satisfy?

The need for secure, hardware-agnostic, and offline-first biometric access control and attendance tracking that runs on existing IP surveillance networks, while guaranteeing 100% local compliance with Algeria’s data protection law (Loi n° 18-07).

c. How is our offering different from our competitors?

We offer a software-only edge AI solution. It integrates directly with existing IP cameras (saving 70–90% of hardware acquisition costs), runs entirely locally on-premise (no internet dependency), and includes a CPU-optimized texture-liveness anti-spoofing algorithm that blocks screens/photos.

d. What is our unique value proposition?

“Secure your space. Protect your data.” A hardware-free, offline-first biometric

intelligence layer that turns standard surveillance networks into secure, anti-spoofing access portals.

4.2.2 Customer Segments

a. Who are our main customers?

Organizations requiring physical access control and attendance tracking: universities, corporate offices, public administrations, and financial institutions in Algeria.

b. What are the different customer segments we target?

1. Higher Education (automated student attendance). 2. Corporate/Industrial Offices (secure employee work logs). 3. Government and Public Administrations (local, audited databases). 4. Banking Branches (high-assurance offline compliance).

c. What are the specific needs of each customer segment?

Universities need to manage large student groups at low cost and prevent attendance fraud. Corporate clients require reliable employee time tracking integrated with HR. Administrations need strict local database compliance. Banks require zero risk of biometric data leakage.

d. How can we categorize our customers into distinct groups?

By sector (educational, administrative, commercial), by deployment scale (single site vs. multi-department), and by existing hardware readiness (cameras already installed).

4.2.3 Customer Relationships

a. What type of relationship does each customer segment expect from us?

High-touch, trust-based relationships. Risk-averse institutions expect offline local validation (pilot trials), dedicated on-site integration support, and multi-tier Service Level Agreements (SLAs).

b. How do we currently maintain relationships with our customers?

Through pilot partnerships with early adopters (such as the Faculty of Technology at the University of Tlemcen), technical presentations, local startup incubator events (i2E Tlemcen), and interactive product demos.

c. How can we improve or personalize our interactions with customers?

Provide customized PyQt5 administration dashboards matching the client's internal workflow, local API integration, and assign dedicated technical support engineers for pilot projects.

4.2.4 Channels

a. Through which channels do our customers want to be reached?

B2B direct outreach, regional security exhibitions (such as Secura Algeria), startup incubation networks (i2E), and direct recommendations from local camera installers and network contractors.

b. Which channels are most effective for reaching each customer segment?

Direct sales for enterprise and university clients; regional contractor partnership network for government and administrative offices; LinkedIn case studies for corporate administrators.

c. How can we integrate different channels to improve customer experience?

Showcase technology at incubator events → present live demos to visiting institu-

tional representatives → offer a free trial pilot at one site → scale to department licenses. All channels feed into a unified local software installer.

4.2.5 Key Partnerships

a. Who are our key partners?

1. Surveillance camera installers and low-voltage contractors. 2. Local startup incubator (i2E Tlemcen) for mentorship and administrative support. 3. University IT departments (pilot validation and R&D). 4. ERP/HR software vendors (API integrations).

b. What partnerships help us reduce costs, access new resources, or improve our value proposition?

Camera installers market our software layer to their existing customer base, reducing our customer acquisition costs. Incubator partnerships provide office space and startup label benefits.

c. How can we align our interests with those of our partners?

Offer camera installers a revenue-sharing model (commissions) on software licenses sold through them. Co-develop API synchronization plugins with HR software vendors.

4.2.6 Key Activities

a. What are the main actions we must take to deliver our value proposition?

Model optimization (liveness and latency tuning), site installation and camera stream calibration, local database integration, ANDPD legal compliance audits, and SLA technical support.

b. What are the essential operations for our business?

Continuous software and liveness AI R&D, sales and business development, custom API development, customer technical support, and legal compliance.

c. Which activities create the most value for our customers?

Zero-hardware setup, CPU-optimized anti-spoofing validation, offline data sovereignty compliance, and smooth synchronization with their existing databases.

4.2.7 Key Resources

a. What are our essential tangible, intangible, and human assets?

Intangible: 7-stage face engine pipeline, 6-feature texture liveness detection algorithm, PyQt5 desktop client codebase. Human: team of 7 specialists (AI, backend, cybersecurity, PyQt5, sales). Tangible: GPU workstations and testing cameras.

b. What tools, technologies, or partnerships do we need to succeed?

Tools: PyTorch, ONNX Runtime, OpenCV, PyQt5, MongoDB/SQLite, SCADA/RTSP network protocols. Partnerships: local camera installers, University of Tlemcen, i2E Incubator.

c. What are the main competitive advantages of our resources?

First software-only biometric access control designed locally in Algeria, fully optimized for standard CPUs (no expensive GPUs required at the edge), and native offline design compliant with Law 18-07.

4.2.8 Cost Structure

a. What are the fixed and variable costs associated with our business model?

Fixed: engineering payroll (7 specialists), office rent, legal registration, and marketing overheads. Variable: deployment travel expenses, installer commission fees, server cloud registry hosting, and hardware testing units.

b. What are the most important costs for our business?

Specialized human capital (software & AI engineers) and customer acquisition/sales operations.

c. How can we reduce costs or improve operational efficiency?

Leverage local university incubator workspace, outsource non-core administrative tasks, automate liveness calibration tools, and utilize installer partnerships for scale.

4.2.9 Revenue Streams

a. What products or services are our customers willing to pay for?

One-time professional site setup and calibration fees, annual SaaS licenses per department/checkpoint, and custom API integration fees.

b. What are the different ways we can generate revenue?

1. One-time Setup Fees (250,000 – 500,000 DZD per access point). 2. Annual SaaS subscriptions per access point (15,000 – 50,000 DZD/month). 3. Custom ERP synchronization consulting.

c. What is our pricing model?

Tiered SaaS model based on client type (Higher Education vs. Private Enterprise) and number of access points, with centralized/volume discounts.

4.3 Conclusion

This chapter presented the industrial deployment strategy and commercial feasibility analysis of Izri Vision through the official Business Model Canvas framework required by the Startup-Diploma initiative. The nine BMC dimensions demonstrate that the proposed software-only, offline-first approach offers a viable and differentiated business model within the Algerian market. By eliminating expensive proprietary hardware terminals and leveraging existing IP camera infrastructure, Izri Solutions positions itself as a cost-effective alternative for universities, corporate offices, government administrations, and banking institutions seeking secure, locally compliant biometric access control.

Business Model Canvas — Izri Solutions

CamCheck (Izri Vision) · Biometric Edge AI · Software-as-a-Service

Key Partnerships • i2E Tlemcen: Startup incubation, mentoring, tech transfer. • Camera Installers: Regional low-voltage contractors as resellers. • University Labs: R&D pilot sites, student testers. • ERP Vendors: API sync with Sage, SAP, payroll systems.	Key Activities • AI R&D: Liveness model tuning, ONNX edge optimization. • Site Deployment: Camera Key Resources • IP: 7-stage face engine, 6-feature liveness gate. • Team: 7 specialists (AI, back-	Value Propositions • Zero Hardware: Software-only, uses existing IP cameras. • Offline-First: 100% local data, no cloud dependency. • Anti-Spoofing: Signal-processing texture liveness gate blocks screens/photos on CPU. • API Flexibility: Swappable backends and custom database	Customer Relationships • Technical Trust: Step-by-step risk mitigation via pilot trials. • SLA Support: Priority system Channels • i2E Tlemcen: Tech transfer and incubator validation. • Installer Network: Contractors	Customer Segments • Higher Education: 106 universities seeking automated student attendance. • Private Enterprises: Corporate offices and manufacturing units needing secure work logs. • Government Offices: Municipalities requiring local, audited access databases.
Cost Structure • Fixed Costs: Core engineering payroll for the 7 specialists (405,000 DZD/month) and office rental/operating overheads (120,800 DZD/month). Total fixed cost is 525,800 DZD/month. • Variable Costs: Deployment travel, server scalability hosting, marketing materials, and commissions paid to camera installer partners.			Revenue Streams • Setup Fees (One-time): Professional site installation, enrollment, liveness calibration, and training (250,000–500,000 DZD per access point). • SaaS Licenses (ARR): Monthly software license subscription per active department/checkpoint (15,000–50,000 DZD/month). • API Integrations: Custom integration fees to bridge logs with client ERPs (Sage, SAP).	

FIGURE 4.1: Landscape Business Model Canvas for Izri Solutions, detailing the product’s strategic commercial positioning in the Algerian market.

General Conclusion

The development, implementation, and experimental validation of the **Izri Vision** project represent a comprehensive effort to bridge advanced computer vision models with the practical, deterministic requirements of industrial automation. By treating a deep learning pipeline as a high-dimensional intelligent sensor within a closed-loop control framework, this thesis has successfully demonstrated a secure, edge-only biometric access control and smart attendance system.

Over the course of this work, several key design objectives were achieved:

- **System Integration and Loop Determinism:** The entire processing loop—from video frame acquisition via an NDI network camera to face detection, anti-spoofing, vector embedding extraction, Faiss database matching, and physical door actuation—was consolidated into a single local execution flow. Under experimental conditions, the system consistently maintained an end-to-end loop latency of less than 500 ms, fulfilling the real-time constraints required for fluid pedestrian flow and industrial control.
- **Biometric Performance at the Edge:** Through rigorous calibration (Phase 1), the combination of YOLOv8-Face for detection and InsightFace (ArcFace) for recognition was identified as the optimal configuration for CPU-only edge nodes. This setup achieved a reliable recognition range of up to 4 m, significantly outperforming classical methods (dlib, MTCNN) which capped at 2 m, while operating entirely on a consumer laptop CPU via ONNX Runtime without discrete GPU acceleration.

- **Resilience to Presentation Attacks:** To secure the access point, a five-feature software-based anti-spoofing gate was developed using LBP texture variance, FFT Moire response, and specular reflection analysis. Running on the CPU alongside the primary pipeline, this signal-processing approach successfully rejected printed photos and replay attacks without the computational overhead of heavy deep learning spoofing models.
- **Practical Academic Validation:** In Phase 2, the system was deployed in a multi-student classroom environment at the Faculty of Technology, University of Tlemcen, for students of the *M2 Automatique et Informatique Industrielle* specialty. The system successfully automated attendance tracking, logging individual presence records, and generating structured administrative reports without user or supervisor intervention.

By keeping all biometric vectors on-premises using a local Faiss vector database, Izri Vision guarantees strict compliance with user privacy guidelines, avoiding any cloud-based data dependencies.

Future Work and Perspectives

While the prototype has demonstrated the viability of the proposed architecture, several directions for future research and engineering remain:

1. **Dedicated Edge Hardware Deployment:** Moving the edge node software from a general-purpose laptop CPU to low-power, dedicated hardware accelerators—such as the Raspberry Pi 5, Coral Edge TPU, or NVIDIA Jetson Nano—would lower power consumption, reduce physical footprint, and allow cost-effective replication across multiple doorways.
2. **Distributed Multi-Camera Network:** Scaling the system to support multiple entrances requires developing a decentralized network of edge nodes com-

municating with a central supervisory server. Investigating load-balancing and MQTT broker performance under concurrent access demands is a crucial next step.

3. **Adaptive Lighting and IR Illumination:** Variable ambient light remains a challenge for optical biometric sensors. Integrating active near-infrared (NIR) illumination and developing camera-intrinsic exposure control loops would significantly improve recognition rates in low-light or high-backlight doorway environments.
4. **Advanced Liveness Modalities:** To counter highly sophisticated presentation attacks (such as 3D silicon masks), future iterations could incorporate remote photoplethysmography (rPPG) to detect pulse-induced skin color changes, or leverage depth sensors for structural liveness verification.

In conclusion, Izri Vision demonstrates that advanced artificial intelligence can be effectively tamed and structured using classical automatic control principles, resulting in a system that is not only smart but also safe, reliable, and respectful of privacy.

Bibliography

- [1] Ali Ismail Awad, Aiswarya Babu, Ezedin Barka, and Khaled Shuaib. Ai-powered biometrics for internet of things security: A review and future vision. *Journal of Information Security and Applications*, 82:103748, may 2024.
- [2] X. You et al. Lvface: A vision transformer for face recognition. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.
- [3] X. Ao et al. Cryptoface: End-to-end encrypted face recognition using fully homomorphic encryption. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.
- [4] Muhammad Fathan Farhat, Muhammad Ary Murti, and Azam Zamhuri Fuadi. Attendance system in class environment with face recognition using arcface algorithm. In *2025 2nd Beyond Technology Summit on Informatics International Conference (BTS-I2C)*, pages 1033–1037, Jember, East Java, Indonesia, dec 2025. IEEE.
- [5] Wang Zhe, Malathy Batumalay, Rajermani Thinakaran, Choon Kit Chan, Goh Wen, Zhang Yu, Li Wei, and Jeyagopi Raman. A research on two-stage facial occlusion recognition algorithm based on cnn. *Engineering, Technology & Applied Science Research*, 14:18205–18212, 2024.
- [6] Anjeana N and K. Anusudha. Real time face recognition system based on yolo and insightface. *Multimedia Tools and Applications*, 83(11):31893–31910, sep 2023.
- [7] J. Dan et al. Comparative analysis of face recognition loss functions. *Journal of Visual Communication and Image Representation*, 2023.
- [8] S. Narayan et al. Facexformer: A unified transformer for facial analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- [9] Abdul Rachman Manga, Tazkirah Amaliah, Yulita Salim, Takeshi Saitoh, Haibo Zhang, and Huzain Azis. From embeddings to ensembles: A controlled comparison of arcface and global cnn features. In *2026 20th International Conference on Ubiquitous Information Management and Communication (IMCOM)*, pages 1–8, Hanoi, Vietnam, jan 2026. IEEE.

- [10] Minghan Li. Research and analysis of facial recognition based on facenet, deep-face, and openface. *ITM Web of Conferences*, 70:03009, 2025.
- [11] Aidana Zhalgas, Beibut Amirgaliyev, and Adil Sovet. Robust face recognition under challenging conditions: A comprehensive review of deep learning methods and challenges. *Applied Sciences*, 15(17):9390, 2025.
- [12] Si Zhou, Xiaochen Yuan, Guanghua Yang, Xinyuan Zhang, Zuobin Ying, and Xueyuan Gong. Doubly relaxed knowledge distillation for deep face recognition. *IEEE Transactions on Biometrics, Behavior, and Identity Science*, pages 1–1, 2026.
- [13] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library, 2025.
- [14] Palivela Thanishka and Pulavarthi Avinash. Integrated surveillance system for real-time face recognition and image-based video retrieval using opencv and deep-face. In *2025 3rd International Conference on Communication, Security, and Artificial Intelligence (ICCSAI)*, volume 3, pages 1693–1700, apr 2025.
- [15] POCO Global. POCO M4 Pro Technical Specifications. <https://www.po.co/global/product/poco-m4-pro/specs>, 2026. Accessed: 2026-05-21.
- [16] University of Tlemcen. University of Abou Bekr Belkaid Tlemcen – Official Website. <https://www.univ-tlemcen.dz/en>, 2026. Accessed: 2026-05-21.

