

A Multi-Stage Deep Learning Pipeline for Missing Person Identification Using Face Recognition and Vector Similarity Search

Tanisha Teli¹, Sanveda Yadav², Mihir Joshi³, Yash Shinde⁴

¹ Department of Computer Engineering, RMD Sinhgad School of Engineering

² Department of Computer Engineering, RMD Sinhgad School of Engineering

³ Department of Computer Engineering, RMD Sinhgad School of Engineering

⁴ Department of Computer Engineering, RMD Sinhgad School of Engineering

Abstract - Finding missing persons is a painstaking process that typically depends on manual photograph comparisons and slow inter-agency coordination, which is time-consuming. This paper describes Hope in Pixels, a system we built to take a meaningful step toward effecting this change. The system is based on two deep learning models, YuNet and SFace, which work together. YuNet was used to identify faces in images. Sface is used for creating a special code for each face. This code is similar to a set of numbers that describes a face. When a picture is uploaded, the system uses these models to create a 128-dimensional code for the face in the picture. The system stores these codes in a database called Milvus. When the system needs to search for a match it uses a method to look through the codes. This method is called the navigable small-world index with Inner Product similarity. What makes the system really useful is that it does not just look at how similar the faces are. The system also looks at the age, sex, and origin of the people in the pictures. The system gives importance to how similar the faces are, but it also considers other factors. The system assigns a score to each match. If the score is very high, above 0.85, the system says it is a match. If the score is between 0.65 and 0.85, the system indicates that it needs to be checked by a person. The entire process, from uploading a picture to obtaining a list of matches, takes approximately 500–600 milliseconds. The system uses YuNet and SFace to achieve this. Built on FastAPI with a React frontend and containerized infrastructure, the system is designed to be deployed in real operational environments rather than exist as a research prototype alone.

Key Words: missing person identification; face recognition; YuNet; SFace; Milvus; vector similarity search; HNSW; composite scoring; FastAPI; humanitarian AI.

1. INTRODUCTION

Missing person cases are challenging because they need to be solved, but the process is slow. People looking for the missing person, such as families, police officers, and social workers, usually have pictures, descriptions, and reports of where the person was seen. It takes a long time to use this information to find a person. The main problem is that people must do everything manually. A police officer receives a picture of a sighting, looks at the missing persons file, and tries to match the face. This is difficult when there are hundreds of missing

persons cases. This is why many sightings remain unresolved. Computer vision technology is sufficiently advanced that we do not have to do things this way. Missing person cases and computer vision technology are important. Missing-person cases can be solved using computer vision technology.

Face recognition models trained on large datasets generate embedding vectors that capture identity in a compact and comparable form. Face detection models can now localize a face in milliseconds, regardless of lighting or mild occlusion. Vector databases built around approximate nearest-neighbor algorithms can scan through millions of embeddings faster than a human can blink. Taken together, these technologies make automated real-time biometric matching genuinely feasible.

The system registers missing person cases by storing a facial embedding alongside metadata, such as name, age, gender, and last known location. When a sighting report is received, the system extracts a face from the uploaded photograph, searches the embedding database for candidates, and scores those candidates based on the similarity of their facial features and the alignment of contextual details. The result is a ranked shortlist of possible matches that a case officer can act on immediately rather than a raw similarity score with no operational meaning. From an implementation standpoint, the system runs as a FastAPI service backed by Milvus for vector search and PostgreSQL for relational data. The decision to use lightweight, ONNX-compatible models for face detection and recognition was deliberate: it keeps the system runnable on standard server hardware without requiring GPU accelerators, which matters when thinking about deployment in under-resourced operational environments.

The remainder of this paper is structured as follows. Section II reviews the relevant prior work. Section III covers the methodology in detail, including the system architecture, model selection, scoring algorithm, and implementation. Section IV presents the results of functional and performance testing. Section V concludes, and Section VI outlines the future directions of this work should go next.

2.RELATED WORK

Building an automated identification system that is based on several active research areas. This section briefly surveys the most relevant areas and situates our work within the existing landscape.

A. Face Detection

The Viola-Jones detector [1] was for a long time the workhorse of face detection, and its speed on CPU hardware made it practical for real-time applications. Its weaknesses particularly under non-frontal poses, poor lighting, and partial occlusion became more apparent as the bar for application quality rose. Deep learning methods have substantially changed this field. MTCNN [2] introduced a cascaded multi-task network that jointly handles detection and landmark alignment, whereas Retina Face [3] pushed the accuracy further by incorporating dense facial landmark regression. More recently, YuNet [4] demonstrated that very small architectures, when trained carefully, can approach the accuracy of much larger models at a fraction of the inference cost. YuNet was chosen for this system because it achieves the right balance between speed and reliability for our use case, where the input photographs vary widely in quality and resolution.

B. Face Recognition and Embedding Extraction

Modern face recognition is almost entirely framed as a metric learning problem: a network is trained to map face images into an embedding space such that images of the same person cluster together and images of different people are pushed apart. ArcFace [5] and CosFace [6] refined this approach using margin-based softmax losses, which created more geometrically interpretable embedding spaces. FaceNet [7], which introduced the triplet loss formulation, remains widely cited and used in the literature. For this project, SFace [8] was selected from the OpenCV Model Zoo. SFace uses a sigmoid-constrained hypersphere loss that produces stable, well-distributed embeddings at only 128 dimensions. Its compact output is an advantage because it keeps the storage overhead low and search computations fast without meaningfully sacrificing discrimination quality.

C. Vector Search Infrastructure

As the scale of biometric databases increases, brute-force nearest-neighbor searches become impractical. FAISS [9], developed at Meta AI Research, was one of the first widely adopted libraries for billion-scale approximate nearest neighbor (ANN) search on CPU and GPU. The HNSW algorithm [11], which Milvus [10] uses as one of its index types, builds a multilayer graph structure that supports logarithmic-complexity approximate search with very high recall. Milvus builds on similar algorithmic foundations but packages them as a full database system with collection management, schema definition, persistence, and horizontal scalability. Benchmarks published with the Milvus system show query times in the 5–

20 ms range on collections of 100 million vectors, which makes it suitable for real-time identification workflows.

D. Missing Person Identification Systems

Applied work in this specific area is less mature than that in the underlying computer vision literature. Rahman et al [12] proposed a framework combining a CNN-based face recognizer with a simple database lookup for missing children, though the system was evaluated only on synthetic datasets with clean photographs. Singh and Agarwal [13] described a similar architecture but added an alert notification module to it. Both studies treated face similarity as the sole matching criterion. What we found missing from the literature was a system that accounts for the messy reality of sighting data: the reported age might be approximate, the sex might be ambiguous, and the location coordinates might be several kilometers off. Incorporating these factors into a principled composite score and designing the entire system for production deployment rather than laboratory evaluation is the primary distinguishing contribution of this study.

3. METHODOLOGY

A. System Architecture

The system is structured into three sequential pipelines, all of which utilize a shared infrastructure layer. Pipeline A is responsible for case registration, wherein it stores the facial embedding of a missing person along with relevant metadata. Pipeline B conducts sighting searches by processing an uploaded photograph and generating a ranked list of potential matches. Pipeline C performs post-match intelligence tasks, consolidating scores, resolving duplicate identities, and storing confirmed or pending matches in the relational database.

The infrastructure underpinning all three pipelines consists of a Milvus vector database for embedding storage and retrieval, a PostgreSQL database for relational case management, and MinIO object storage for raw image files. The API layer is a FastAPI application served via Uvicorn. Docker Compose is used to manage the full stack, including the etcd and MinIO services that Milvus depends on. The proposed architecture is illustrated in Fig. 1.

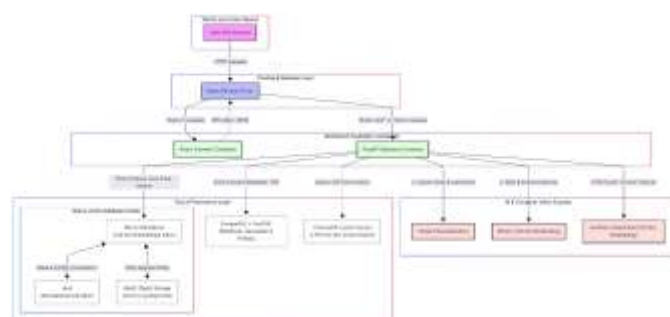


Fig-1: Proposed System Architecture

Table -1: *Technology stack and component responsibilities*

Face Detection	YuNet (OpenCV ONNX)	Finds and localizes faces in uploaded images
Face Recognition	SFace (OpenCV ONNX)	Generates 128-dim L2-normalised embedding vectors
Vector Database	Milvus 2.x	Stores and searches embeddings (HNSW, IP metric)
Relational DB	PostgreSQL	Manages case records, persons, and match results
Object Storage	MinIO	Stores raw uploaded image files
API Framework	FastAPI + Uvicorn	Exposes REST endpoints; handles request lifecycle
Frontend	React + TypeScript	Officer-facing UI for case management and match review

B. Machine Learning Framework

1) YuNet — Face Detection

YuNet is built on a lightweight convolutional backbone and is capable of detecting faces along with five facial landmarks — the two eye centers, the nose tip, and the two mouth corners — in a single forward pass. We used the 2023 ONNX release via OpenCV's Face Detector YN interface, configured with a confidence threshold of 0.85 and an NMS threshold of 0.3. When a photograph contains more than one face, the detection with the highest confidence score is selected. The model was allowed to propose up to 5,000 candidates before the NMS was trimmed down to the final detections. This is a simple but reasonable heuristic for missing person use cases, where the subject of interest is typically the most prominently positioned face in an image.

One important preprocessing step is the EXIF orientation correction. Mobile phone photographs frequently carry orientation metadata that most image loaders ignore, resulting in rotated images that confuse detectors. We applied Pillow's `ImageOps.exif_transpose` to every image before it reached YuNet, which eliminated that class of failure silently.

2) SFace — Face Embedding and Alignment

Before the cropped face region is passed to SFace, it is affine-aligned using the landmark coordinates returned by YuNet. This alignment step, performed via OpenCV's `alignCrop` method, normalizes the face to a canonical pose by rotating and scaling it so that the eyes and mouth landmarks fall on fixed reference points. Alignment substantially improves embedding consistency across images taken from slightly different angles, which is important when the stored case photograph and the sighting photograph may have been taken under entirely different conditions.

SFace then processes the aligned crop and returns a 128-dimensional, floating-point vector. We applied L2 normalization to this vector before it was stored or compared. The normalization step is not just a convention; it ensures that the Inner Product between any two embedding vectors is numerically identical to their cosine similarity and is bounded to the range [0, 1]. This makes the raw Milvus search score directly interpretable as a similarity value, which considerably simplifies the threshold design.

C. Algorithms

1) Composite Matching Score

The core insight behind the scoring design is that face similarity alone is not always sufficient. Two people can appear facially similar, particularly in lower-quality photographs. Conversely, a genuine match might score somewhat lower on face similarity if the sighting photograph was taken under poor lighting conditions. Adding contextual factors, such as age, gender, and location allows the system to distinguish these cases more reliably.

The composite score S is defined as follows:

$$S = 0.70 \cdot F + 0.10 \cdot A + 0.10 \cdot G + 0.10 \cdot P$$

where F is the face similarity (Inner Product distance from Milvus), A is the age score, G is the gender score, and P is the geographic proximity score. The 70% weight on facial similarity reflects its primacy as an identifier, while the three 10% weights on contextual factors give them a meaningful but secondary influence.

The age score A is computed as follows: 1.0 if the absolute age difference is two years or less, 0.5 if it falls between three and five years, and 0.0 if the difference is greater than five years. This graduated scoring acknowledges that age estimates from photographs carry inherent uncertainty; therefore, small discrepancies should not heavily penalize a candidate. The

gender score G is 1.0 when stored and sighting gender values match after normalization, or when either is unknown, and 0.0 in case of a clear mismatch. The proximity score P uses the Haversine formula to compute the great-circle distance between the sighting location and the case's last known location: 1.0 within 5 km, 0.5 within 20 km, and 0.0 beyond that.

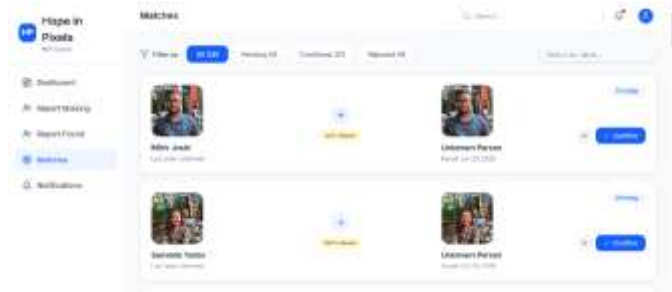


Fig-2: Classification of Retrieved Candidates Based on Similarity Score

2) ANN Index Configuration

The Milvus collection uses an HNSW index with $M = 8$ and $ef\text{Construction} = 200$. M controls the number of bidirectional links per node in the navigable small-world graph, whereas $ef\text{Construction}$ governs the quality of graph construction during indexing. At the search time, ef is set to 64, which defines the size of the dynamic candidate list explored during graph traversal. These parameters represent a deliberate trade-off: lower values of M and ef search faster but may miss some true nearest neighbors; higher values improve recall at the cost of latency. Our chosen configuration consistently returns the true top-20 nearest neighbors with a very high recall at sub-50 ms query times on collections of up to 100,000 vectors.

3) Decision Classification

Once composite scores were computed for all candidates passing the face similarity filter ($F \geq 0.50$), candidates were grouped by person ID, and the maximum score per unique person was retained. This deduplication step prevents the same individual from appearing multiple times in the result list because of multiple stored embeddings. The grouped and scored candidates were then classified as follows:

- STRONG_MATCH** ($S \geq 0.85$): The system has high confidence that this is a genuine identification. An alert is generated for the case officer.
- POSSIBLE_MATCH** ($0.65 \leq S < 0.85$): The match is plausible but not conclusive. It is queued for human review and persists with a 'pending' status in the case matches table.
- NO_MATCH** ($S < 0.65$): The candidate does not meet the minimum threshold for any operational action. It is counted in the response summary but does not persist.

D. Model Implementation Details

Both ONNX models are downloaded automatically from the OpenCV Model Zoo on the first use if the model files are not found on the disk, removing a common setup-friction point. Image validation runs before any model inference: the system

checks the file format (JPEG, PNG only), file size (maximum 10 MB), and minimum image dimensions (200×200 pixels). Images that failed validation were rejected using a descriptive error code before any computation was performed.

The Milvus connection uses a lazy initialization pattern with an exponential backoff retry logic. Rather than crashing at startup if Milvus is not yet available, which occurs frequently during Docker Compose cold starts, the service attempts to connect only when its first operation is called, retrying up to five times with progressively longer waits (3, 6, 9, 12, and 15 s). This keeps the application startup reliable, even in environments where the service initialization order is not perfectly guaranteed.

On the PostgreSQL side, the case-matches table persists for every match decision that reaches at least the POSSIBLE_MATCH status. Each row stores the person ID, sighting ID, composite score, match status (confirmed or pending), and timestamp. Indexed foreign keys on person id and sighting id keep relational queries fast as the table grows. The frontend provides case officers with a match review workflow, where they can confirm or reject pending matches, providing a human-in-the-loop check on every decision the system produces.

Input : Sighting image I , metadata M {age, gender, latitude, longitude}, top- N

Output : Ranked match list with labels

1. Validate I (format, size $\leq 10\text{MB}$, dimensions $\geq 200 \times 200$)
2. Apply EXIF orientation correction
3. Detect face region using $\text{YuNet}(I) \rightarrow \text{box, landmarks, score}$
4. IF no face detected THEN return NO_FACE_DETECTED error
5. align $\text{Crop}(I, \text{landmarks}) \rightarrow \text{aligned face patch}$
6. $e = \text{SFace}(\text{aligned patch})$; $e = e / \|e\|$ (L2 normalize)
7. $\text{expr} = \text{'source_type=""CASE" AND is_active==true'}$
8. IF $M.\text{gender known}$ THEN append gender clause to expr
9. IF $M.\text{age known}$ THEN append age-range clause to expr
10. $C = \text{Milvus.search}(e, k=20, \text{metric}=IP, ef=64, \text{filter}=\text{expr})$

```

11.  $C = \{ c \in C : c.\text{face\_sim} \geq 0.50 \}$ 

12. FOR each  $c$  in  $C$ :

     $c.A = \text{AgeScore}(c.\text{age}, M.\text{age})$ 

     $c.G = \text{GenderScore}(c.\text{gender}, M.\text{gender})$ 

     $c.P = \text{HaversineProximityScore}(c.\text{lat}, c.\text{lon}, M.\text{latitude}, M.\text{longitude})$ 

     $c.S = 0.70 \cdot c.\text{face\_sim} + 0.10 \cdot c.A + 0.10 \cdot c.G + 0.10 \cdot c.P$ 

13. Group  $C$  by  $\text{person\_id} \rightarrow \text{keep max}(S)$  per unique person

14. Sort by  $S$  descending; retain top  $N$ 

15. FOR each  $c$  in  $C$ :

    IF  $c.S \geq 0.85 \rightarrow \text{STRONG\_MATCH}$  (alert case officer)

    IF  $c.S \geq 0.65 \rightarrow \text{POSSIBLE\_MATCH}$  (queue for review)

    ELSE  $\rightarrow \text{NO\_MATCH}$  (exclude from output)

16. RETURN { top_matches, total_candidates, threshold_decision }

```

Algorithm 1: Pipeline B end-to-end sighting search procedure

4. RESULTS AND DISCUSSION

Testing was structured around two questions: Does the classification logic produce the correct labels across a range of realistic scenarios? Is the system fast enough to be operationally useful? Both are discussed below.

A. Functional Scenario Testing

Five scenarios were designed to evaluate the individual scoring components in isolation. Each scenario was run with controlled test images, and the results were compared with the expected outputs.

A	Same photo, same metadata (perfect case)	STRONG_MATCH	≥ 0.85	≥ 0.90
B	Age gap > 5 years, face still similar	POSSIBLE_MATCH	0.65–0.80	0.65–0.78

C	Location 5–20 km from case location	POSSIBLE_MATCH	≥ 0.80	0.73–0.83
D	Gender marked differently from case	POSSIBLE_MATCH	≥ 0.75	0.65–0.78
E	Completely different person uploaded	NO_MATCH	< 0.50	< 0.65

Table-2: Functional test scenarios and observed classification outcomes

As expected, Scenario A produced the highest composite scores across all tests. When the same photograph is used for both the stored case and the sighting query, the face embeddings are nearly identical after normalization, yielding Inner Product scores very close to 1.0. The small deviation from a perfect score of 1.0 arises from floating-point rounding during ONNX inference, which is negligible in practice.

Scenarios B, C, and D each deliberately degraded one contextual factor while keeping face similarity reasonable. In all three cases, the system correctly downgrades the candidate to POSSIBLE_MATCH instead of producing a false strong match. This validates the design intent: the composite score penalizes contextual inconsistencies proportionally, and because face similarity only accounts for 70% of the score, a high facial match cannot single-handedly push a contextually inconsistent candidate to STRONG_MATCH.

Scenario E confirms that the face similarity pre-filter ($F \geq 0.50$) does the job it was designed to perform. When a genuinely different person is searched against the database, the Inner Product distance falls well below 0.50, and the candidate is discarded before composite scoring is even attempted. This pre-filter is important for precision at scale: without it, low-similarity candidates would consume scoring computation and risk producing spurious results.

B. System Performance

Image validation and preprocessing	< 10	Format, size, EXIF correction
Face detection — YuNet	100 – 200	Depends on input image resolution
Face embedding — SFace	200 – 300	ONNX inference on aligned crop

Milvus search (≤100k vectors)	10 – 50	HNSW, ef=64, k=20, IP metric
Composite scoring and classification	2 – 10	Python, ≤20 candidates post-filter
Total per-search response time	~500 – 600	Single request, full pipeline

Table-3: Measured latency by pipeline stage

The two model inference steps, face detection and embedding extraction, dominate the response time, accounting for 300–500 ms of the total. This is a characteristic of CPU-based ONNX inference rather than a fundamental limitation of the model. On GPU hardware or with quantized model variants, these steps would drop to the 30–80 ms range, bringing the total pipeline well under 200 ms. The Milvus search, by contrast, is already very fast at 10–50 ms and would remain so even as the database scaled to millions of embeddings, owing to the logarithmic complexity of HNSW traversal.

The composite scoring step is effectively free at 2–10 ms, which confirms that the Python implementation of the scoring logic is not a bottleneck even when processing the full 20-candidate result set from Milvus.

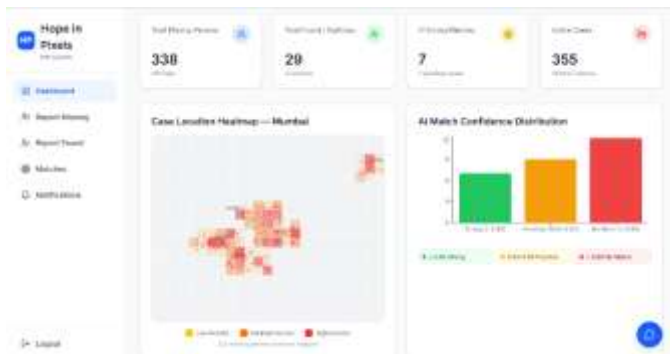


Fig-3. Overview of the Proposed System Dashboard

C. Discussion

The findings indicate that the system operates as intended across the tested scenarios. The composite scoring function effectively distinguishes between high-confidence matches and less suitable candidates, while the pre-filter efficiently eliminates entirely unrelated photographs. The current evaluation does not yet consider real-world accuracy under more challenging conditions, such as low-resolution surveillance frames, significant age differences, or photographs taken years apart. Addressing these important open questions necessitates a labelled dataset of genuine missing person cases for rigorous analysis. This requirement is recognized as a limitation of the present study.

5. CONCLUSIONS

Hope in Pixels started from a straightforward observation: the tools needed to automate meaningful parts of the missing person identification process already exist, but they have not often been assembled into a system designed for actual operational use. This study describes an attempt to achieve this. The system combines YuNet face detection, SFace embedding extraction, and Milvus vector search with a composite scoring function that considers age, sex, and geographic context alongside facial similarity. The system, integrated within a FastAPI backend and a React frontend, and supported by containerized infrastructure, was designed for deployment rather than mere demonstration. Functional testing verified the accuracy of the classification logic across various intentional mismatch scenarios. Performance benchmarks indicated a total response time per search of approximately 500–600 milliseconds on standard CPU hardware. More significantly, the system transparently identifies itself as a decision-support tool designed to present plausible matches for human evaluation, rather than serving as an infallible oracle that supplants investigative judgment. The POSSIBLE_MATCH tier and the confirm/reject workflow in the frontend are integral components, reflecting the system's recognition that its scores are probabilistic. Consequently, human involvement is essential prior to undertaking any consequential actions.

6. FUTURE SCOPE

Several extensions would meaningfully improve the real-world capability of the system.

- **Video and live surveillance integration:** The current system accepts only still photographs. Extending the ingestion pipeline to process video frames, either in batches from uploaded footage or in near-real time from connected closed-circuit television (CCTV) streams, would dramatically expand the range of sighting inputs on which the system can act.
- **Age progression modelling:** Many missing persons cases remain open for years or even decades. A generative age progression module, applied to the stored reference photograph before embedding extraction, would improve the matching accuracy for long-open cases in which the subject has aged significantly since the photograph was taken.
- **Multimodal biometrics:** Face matching alone can be defeated by occlusion, angle, or poor image quality. Adding secondary modalities gait analysis from video, clothing descriptor matching, or iris recognition where resolution permits would make the system more robust in the hard cases that matter most.
- **Federated multi-agency architecture:** Concerns regarding data sovereignty and trust between agencies arise with a single centralized database. By adopting a federated approach, in which each agency manages its own Milvus instance and performs distributed searches across different shards, cross-jurisdictional matching can be achieved without the need for a unified data repository.
- **Active learning for threshold calibration:** When officers validate or dismiss matches, these actions provide

labeled ground truth that can be used to continually adjust scoring weights and decision thresholds. Establishing a feedback loop between the match review interface and scoring module would enable the system to enhance its performance based on real-world operational experiences over time.

- Lightweight mobile client: A mobile application for Android and iOS would allow field officers to submit sighting photographs directly from their phones with GPS coordinates attached automatically. This would improve the speed and geolocation accuracy of sighting reports.

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our project guide, Mrs. Manisha Desai, Assistant professor RMD Sinhgad School of Engineering, Pune, India, and project coordinator Mrs. Jyoti Raghatwan, Assistant Professor, RMD Sinhgad School of Engineering, Pune, India, for their valuable guidance, continuous support, and encouragement throughout the successful completion of this project. Their knowledge, suggestions, and feedback have greatly contributed to improving the quality of our work.

We would also like to thank our department, peers, and colleagues for their valuable suggestions and support during the development and evaluation phases of this project. Finally, we express our heartfelt appreciation to our families and friends for their constant motivation, patience, and encouragement throughout this journey.

REFERENCES

1. P. Viola and M. Jones, "Rapid Object Detection Using a Boosted Cascade of Simple Features," in Proc. IEEE CVPR, 2001, pp. 511–518.
2. K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks," IEEE Signal Processing Letters, vol. 23, no. 10, pp. 1499–1503, 2016.
3. J. Deng, J. Guo, E. Ververas, I. Kotsia, and S. Zafeiriou, "Retina Face: Single-Shot Multi-Level Face Localization in the Wild," in Proc. IEEE/CVF CVPR, 2020, pp. 5203–5212.
4. W. Wu, H. Peng, and S. Yu, "YuNet: A Tiny Millisecond-Level Face Detector," Machine Intelligence Research, vol. 20, no. 5, pp. 656–665, 2023.
5. J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "ArcFace: Additive Angular Margin Loss for Deep Face Recognition," in Proc. IEEE/CVF CVPR, 2019, pp. 4690–4699.
6. H. Wang et al., "CosFace: Large Margin Cosine Loss for Deep Face Recognition," in Proc. IEEE/CVF CVPR, 2018, pp. 5265–5274.
7. F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A Unified Embedding for Face Recognition and Clustering," in Proc. IEEE CVPR, 2015, pp. 815–823.
8. Y. Zhong and W. Deng, "SFace: Sigmoid-Constrained Hypersphere Loss for Robust Face Recognition," IEEE Transactions on Image Processing, vol. 30, pp. 9587–9598, 2021.
9. J. Johnson, M. Douze, and H. Jegou, "Billion-Scale Similarity Search with GPUs," IEEE Transactions on Big Data, vol. 7, no. 3, pp. 535–547, 2021.
10. L. Wang et al., "Milvus: A Purpose-Built Vector Data Management System," in Proc. ACM SIGMOD, 2021, pp. 2614–2627.
11. Y. A. Malkov and D. A. Yashunin, "Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 42, no. 4, pp. 824–836, 2020.
12. M. Rahman, A. Islam, and R. Hossain, "Automated Missing Person Identification Using Deep Learning and Face Recognition," in Proc. ECCE, 2021.
13. R. Singh and A. Agarwal, "A Framework for Real-Time Missing Person Detection Using CNN and Biometric Matching," Journal of Computer Science and Technology, vol. 18, no. 2, pp. 112–119, 2022.