Software Engineering Department

Braude College of Engineering

AI-Driven Person Recognition and Identification in Videos

Project code: 24-1-D-35

Capstone Project Phase B – 61999

Sep 2024

Rom Harel

Itay Benjamin

Git repository link:
https://github.com/RomRL/person-recognition

Supervisor:
Mr. Ilya Zeldner

המכללה האקדמית להנדסה בראודה בכרמיאל

1 Abstract

The wide range of video content used in different industries necessitates the creation of efficient and precise person identification systems for videos. This project proposes a new method that combines face recognition technology that is currently the best, with advanced individual detection technologies to identify people from within video files. Essentially, the YOLO (You Only Look Once) algorithm detects humans using every frame while FaceNet recognizes faces so it can tell when and where exactly those frames are in relation to its target person. This novel strategy improves our ability to analyze videos thus making it very useful for security agencies among others

2 Introduction

With increased use of video content across many domains there arises a need for advanced person identification systems. These include but are not limited to: security, surveillance, media research among others. Traditional approaches have problems with accuracy and efficiency due to dynamics within streams caused by changing environment conditions.[5] The relevance of such technology is evident from incidents like that which occurred on 7th October where hundreds of Israelis were taken hostage into Gaza. Our system could play a big role in identifying persons seen in these footages during rescue or investigative operations carried out under high-risk situations like this one. By providing immediate & precise identity verification abilities; law enforcement agencies will be able to quickly react towards emergencies by narrowing down their searches thereby saving more lives and leading to better security outcome. Deep learning has shown significant improvement in

person recognition accuracy through hierarchical feature extraction from unprocessed data mostly using convolutional neural networks (CNNs)[11]. For real-time object detection in video frames, YOLO (You Only Look Once) algorithm is commonly used[20]. FaceNet ,on the other hand ,provides higher level face recognition by transforming facial images into Euclidean space hence making it robust against different orientations and expressions[24]. This proposal combines YOLO for object detection with FaceNet for face recognition thus creating an end-to-end system capable of accurately identifying people from videos as discussed below. Our system is engineered using the latest neural networks and AI techniques, integrating technologies such as MongoDB for database management[50], FastAPI[51] and Python[49] for server-side operations, and React along with JavaScript for the frontend[48]. This robust combination enhances performance in person recognition tasks, significantly improving the accuracy of recognizing individuals within complex video environments. Additionally, it offers an intuitive React-based interface[48] that simplifies user interaction, making it accessible to individuals who may not have the technical expertise to operate advanced video analysis tools. By merging these state-of-the-art technologies into a unified package, our project aims to revolutionize the way we analyze and interact with video content across various fields.

3 Background and Related Work

3.1 Person Identification in Videos

In computer vision, the process of identifying individuals in video streams is critical but complicated, serving needs in surveillance, forensics and monitoring. The advent of deep learning has greatly improved the accuracy and resilience of this technology, with Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) being major drivers. CNNs excel at capturing important spatial features from single frames while RNNs such as ConvLSTMs are effective at capturing temporal dynamics across sequences. These architectures accommodate variations in appearance, lighting conditions and posture to ensure consistent identification over consecutive frames and viewpoints. Other improvements like 3D pose estimation and multi-stream networks have led to better performance for cases involving multiple perspectives or modalities [1]. However, real-time processing constraints, scalability issues and dataset generalization challenges persist. Recent studies aim to enhance robustness, computational efficiency and domain adaptation while addressing ethical concerns that may restrict their practicality. With improving deep learning techniques however; its application in person identification through videos is expected to increase immensely thus revolutionizing visual data analysis dependent industries.

3.1.1 Deep Learning for Person Identification

Person identification has evolved greatly and Deep Learning, which refers to Convolutional Neural Networks (CNNs) in particular, is at the forefront of this. In the past, typical methods relied on handcrafted features that often failed under new conditions. Here, CNNs are able

to automatically learn hierarchical features which make them more resilient against any form of variations [34]. Such architectures usually have multiple convolutional layers followed by pooling layers that capture spatial hierarchies and fully connected layers for identity prediction.

3.1.2 Cross-View Identification

Person identification can also require matching individuals across different camera views or modalities (e.g., RGB and thermal images). Cross-view identification is very difficult because perspectives and backgrounds keep changing.

One way for cross-view identification is by learning features that don't change no matter the angle and by using 3D human pose data [29]. For example, CNNs (Convolutional Neural Networks) can be trained to pick up on features that stay consistent across various views. Another helpful method is using multi-view dynamic images, which combine information from different angles to make the system more robust and reliable [30].

3.1.3 Multi-Modal Identification

For environments with varied lighting conditions such as nighttime surveillance, combining visible and thermal imagery will enhance identification accuracy.Two-stream CNNs are used where one stream processes visible images and the other thermal images. These networks are trained jointly so that they learn features that bridge the gap between two modalities.

3.2 Object Detection

Object detection is a basic computer vision task that aims at identifying and locating objects in digital images. This capability is a primary requirement for self-driving cars, security systems and augmented reality which demand high accuracy of object detection [33]. In contrary to image classification that can only identify if there are objects in an image, object detection should take into account the exact positions of each instance using bounding boxes. Consequently, this makes it more challenging because the model has to both recognize the object as well localizing it correctly. In terms of object localization and recognition, deep learning, especially Convolutional Neural Networks (CNNs), has revolutionized how they perform this function [12]. They have improved on architectures which learn raw pixel data directly for rich and hierarchical visual features automatically hence making them capable of detecting intricate patterns and variations in objects that are not easily engineered manually . Therefore, recent developments in CNNs have produced modern detectors primarily based on deep CNN architectures which are meticulously designed and optimized for this purpose.A significant architecture within this field is Faster R-CNN which utilizes a Region Proposal Network (RPN) to detect possible areas where an object may be located first in an image. A subsequent stage then refines these proposals by classifying them [21].

However, while two-stage approach provides higher accuracy but it is computationally expensive thus limiting its application to real-time situations. On the other hand one-stage detectors such as YOLO(You Only Look Once) give bounding boxes along with classification scores through a single pass network thereby having significantly faster inference time as compared to two-stage models albeit with a slight reduction in precision.Despite their impressive capabilities, state-of-the-art object detectors still encounter challenges such as occlusions, varying object scales, and cluttered or complex backgrounds. To address these issues some recent works introduced feature pyramids [21], . or attention mechanisms that can select relevant parts automatically within input area [20]. Object detection remains an active area of research as we strive for more robust and efficient detectors that can operate in real-world environments. In general, modern object detectors have become powerful tools through deep learning breakthroughs enabling a variety of exciting applications across different domains such as robotics, augmented reality and autonomous vehicles. Even though they are not perfect their performance continues to improve at a breakneck pace thus propelling the progress in related fields while opening new horizons in technology.

3.3 Current uses of person detection

Person detection technologies are often used in video scenarios, to identify individuals. In surveillance and security settings, they enable automated monitoring, can track people and determine what activities are taking place [5]. To analyze consumer behavior and manage queues more effectively so as to improve operations of a store, retailers use person detections in this case study. Smart cities and traffic management rely on the counting of pedestrians and crowd analysis to optimize traffic flow and ensure the safety of pedestrians. Monitoring and fall detection are crucial for healthcare facilities as well as assisted living centers that use these technologies for providing safety alerts. Sports industries like athletics involve tracking players by using person detection also known as tracking technologies. It assesses their performance to give them ideas. Autonomous vehicles count on it for pedestrian recognition while cycling is important in roadway safety assurance. Workplace safety has improved through high-risk area monitoring with corresponding strict enforcement of safety rules. Identifying individuals of interest in intelligence agencies will require person detection during investigations which involves analyzing surveillance footages thus enhancing national security measures from such cases. They also help in looking out missing persons or kidnaps by scanning CCTV footage to identify victims or suspects thereby aiding investigations effectively with respect to law enforcement agencies employing person detection technology. Such instances demonstrate how personal identification tools contribute towards enhanced protection, secure surroundings as well as efficient sector operation across various domains.

3.4 CNN

Convolutional neural networks (CNNs) are a class of deep neural network architectures that have changed the field of computer vision and image processing. The main thing which allowed them to achieve high accuracy in solving a wide range of computer vision tasks such as image classification, object detection, semantic segmentation, and instance segmentation is their ability to get

hierarchical features from raw pixel data [13].CNNs consist of convolutional layers which apply learnable filters on input images resulting in low-level features like edges, textures, patterns etc. Convolution operations are made for capturing spatially localized patterns that enables the network to exploit inherent stationarity in images. On the other hand, pooling layers reduce computational complexity by downsizing the spatial dimensions while retaining the most informative parts.The network layers used in Convolutional Neural Networks are nonlinear, though popular and standard pushing include rectifying linear units (ReLUs). This brings in non-linearity, important for these models to learn complex distributions present in data in real life scenarios. In the deep network it is necessary for the network to construct high level feature representations, that is why such nonlinearity is crucial.When the depth of the network grows, the upper layers in a sense are able and allowed to learn more high-level and even semantic concepts by combining and transforming the features gathered by the layers lower in the hierarchy. This hierarchical learning is a powerful strength of CNNs to get multiple level representations from raw data and thus free from feature engineering. Several challenges exist in CNNs , they require large annotated datasets and are easily considered adversarial, and finally, the learned representations are often difficult to interpret—also termed the black box effect. Modern studies are looking into methods like transfer learning, adversarial training, and explainable artificial intelligence to overcome these problems and enhance the CNN model's robustness, effectiveness, and interpretability [13].

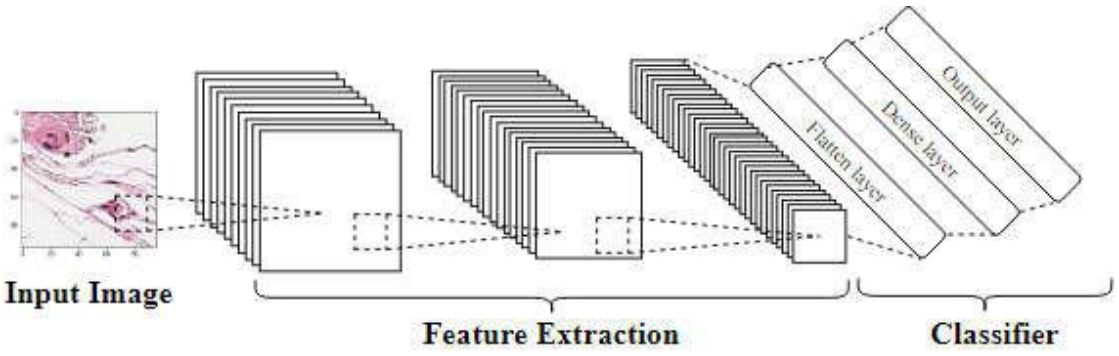

Figure 1 : Convolutional neural network (CNN) diagram.[10]

3.5 Advances in Video-Based Person Recognition

Compared to still pictures, recognizing persons in video streams presents a great challenge on account of the issues related to changing viewpoints, occlusion, lighting changes and dynamic backgrounds. It is for this reason that image-based recognition techniques often fail in video sequence settings, calling for specialized models. One way forward is using end-to-end deep learning models which process raw videos as they are. A type of these is 3D CNNs whose conventional 2D convolution is expanded into a time dimension thereby enabling it to extract spatio-temporal features that capture both spatial and temporal patterns [26]. On an alternative note recurrent neural networks such as LSTMs can model dynamic temporal dependencies by processing inputs recursively over time [4]. Combining these models by extracting spatial features from individual frames using CNNs and feeding them into RNNs for temporal integration exploits the strengths of both architectures. The significance of deep learning in this field lies in its ability to find and utilize spatiotemporal

patterns automatically instead of relying on hand-engineered heuristics. Attention mechanisms have recently been introduced into these models [6], making it possible for networks dynamically shift their focus across relevant spatial regions and time segments thereby getting rid of noise. This reflects human attention mechanisms and improves performance while also providing interpretability by indicating areas where the model has focused on.

3.6 YOLO

YOLO is a real-time object detection system which uses deep learning to identify specific objects in images and videos [20]. It was developed by Joseph Redmon et al. at the University of Washington and introduced in 2016 in the paper "You Only Look Once: Unified, Real-Time Object Detection". YOLO is a single convolutional neural network that divides the input image into a grid system, where each cell in the grid is responsible for detecting objects within itself. For these boxes simultaneously, the network predicts bounding boxes and class probabilities.

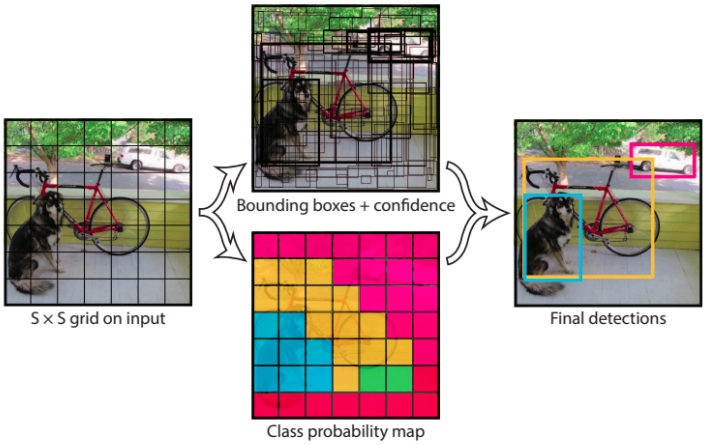

Figure 2 :Yolo approach for detection by modeling it as a regression problem. The image is divided into an S × S grid, and each grid cell is responsible for predicting B bounding boxes, the confidence scores for those boxes, and C class probabilities. These predictions are then encoded into a tensor with dimensions S × S × (B ∗ 5 + C). [20]

3.6.1 The key features of YOLO

What makes YOLO unique from other systems is its ability to process information in real time – it can handle up to 45 frames per second (hence very fast). Object detection is performed with only one evaluation of the neural network thus ensuring high efficiency. Moreover, it can generalize and detect objects that were not specifically trained on [20].

3.6.2 How YOLO Works

3.6.2.1 Image and Grid Division

The object detection process begins with an input (for this example use grayscale image) of dimensions X * Y. This image is then divided into an N X N grid, where each cell is processed independently to detect objects.

3.6.2.2 Classes and Threshold

YOLO recognizes specific object classes like person's class after being trained; step size for each cell = image dimensions / number of cells hence cell dimensions are X/N , Y/N .

3.6.2.3 Prediction Arrays

For instance if there are four possible categories namely person, dog, wolf cow , then yolo will be taught on how to recognize them all; therefore it should predict probability of every class for every cell ; this information is stored in array format (e.g., 50% person, 30% dog, 10% wolf,and20% cow). YOLO also predicts several bounding boxes for each cell where box is defined by its center coordinates,width,height and confidence score (which shows how likely it overlaps with real object) [20].

3.6.2.4 Prediction Process

Each cell initializes an empty list to store final predictions. The algorithm extracts each cell from the grid and loops over them, predicting class probabilities as well as finding best bounding boxes based on confidence scores. It identifies highest probability class, and if its confidence score exceeds a predetermined threshold then adds this prediction to the final list.

3.6.2.5 Output

The final output is a list of detected objects. Each entry in the list includes the coordinates of the bounding box and the predicted class.

3.6.2.6 YOLO Pseudocode

```
image = readImage()
NoOfCells = 7
NoOfClasses = 1
step = int(height(image) / NoOfCells)
prediction_class_array = new_array(size(NoOfCells, NoOfCells, NoOfClasses))
predictions_bounding_box_array = new_array(size(NoOfCells, NoOfCells, NoOfCells, NoOfCells))
final_predictions = []
threshold = 0.5

for i in range(0, NoOfCells):
    for j in range(0, NoOfCells):
        cell = image[i*step:(i+1)*step, j*step:(j+1)*step]
        prediction_class_array[i, j] = class_predictor(cell)
        predictions_bounding_box_array[i, j] = bounding_box_predictor(cell)
        best_bounding_box = 0 if predictions_bounding_box_array[i, j, 0, 4] > predictions_bounding_box_array[i, j, 1, 4] else 1
        predicted_class = index_of_max_value(prediction_class_array[i, j])
        if predictions_bounding_box_array[i, j, best_bounding_box, 4] * max_value(prediction_class_array[i, j]) > threshold:
            prediction = [predictions_bounding_box_array[i, j, best_bounding_box, 0:4].tolist(), predicted_class]
            final_predictions.append(prediction)

print(final_predictions)
```

Figure 3 : YOLO Pseudocode [36]

3.6.3 Applications

YOLO has many applications in various domains, including:

- **Surveillance and security:** Real-time object detection in video feeds for security monitoring.
- **Autonomous vehicles:** Detection of pedestrians, vehicles, and other objects for self-driving cars.
- **Robotics:** Identification of objects for robotic manipulation and navigation.
- **Retail and inventory management:** Detection and counting of products on shelves.

YOLO, is used in many different ways. For surveillance systems and safety measures it is able to detect objects in live streams which makes monitoring security a lot tighter. When it comes to self-driving cars, YOLO helps them recognize pedestrians, other vehicles or any other things around them thus enabling safe navigation. In robotics it can be used for object identification during manipulation and movement planning tasks while in retail industry among other areas where goods are stored on shelves for sale it counts them quickly saving time during stock taking. All these various applications show how flexible and influential YOLO has been towards technological advancement.

3.7 Single Shot Detector (SSD) with MobileNetV2

An efficient and accurate framework for object detection is the Single Shot Detector (SSD) architecture, which was proposed by Liu et al. [16]. In a single pass, SSD employs just one neural network to predict bounding boxes as well as probabilities of classes thereby allowing for real-time object detection. For mobile and embedded vision applications, Sandler et al. [23] developed MobileNetV2 which is a lightweight convolutional neural network. It introduces inverted residual structures and linear bottlenecks on top of the original MobileNet design thus reducing computational cost significantly while maintaining high accuracy. SSD MobileNetV2, which combines SSD with MobileNetV2, provides an interesting approach towards resource constrained devices based real time object detection. This integration makes use of MobileNetV2's efficient feature extraction capabilities as the backbone network for SSD hence producing a model that balances between speed and accuracy.

There are several important aspects incorporated in SSD MobileNetV2 that contribute to its effectiveness. Multi-scale feature maps from various layers in the MobileNetV2 backbone are used so that objects at different scales can be detected. The architecture adopts default bounding boxes having different scales and aspect ratios associated with each feature map cell which enhances detection accuracy across diverse sizes of objects. Depthwise separable convolutions used by MobileNetV2 greatly reduce parameters' number and computational complexity. Furthermore, inverted residual blocks enable more efficient feature extraction besides improved gradient flow within MobileNetsV2.

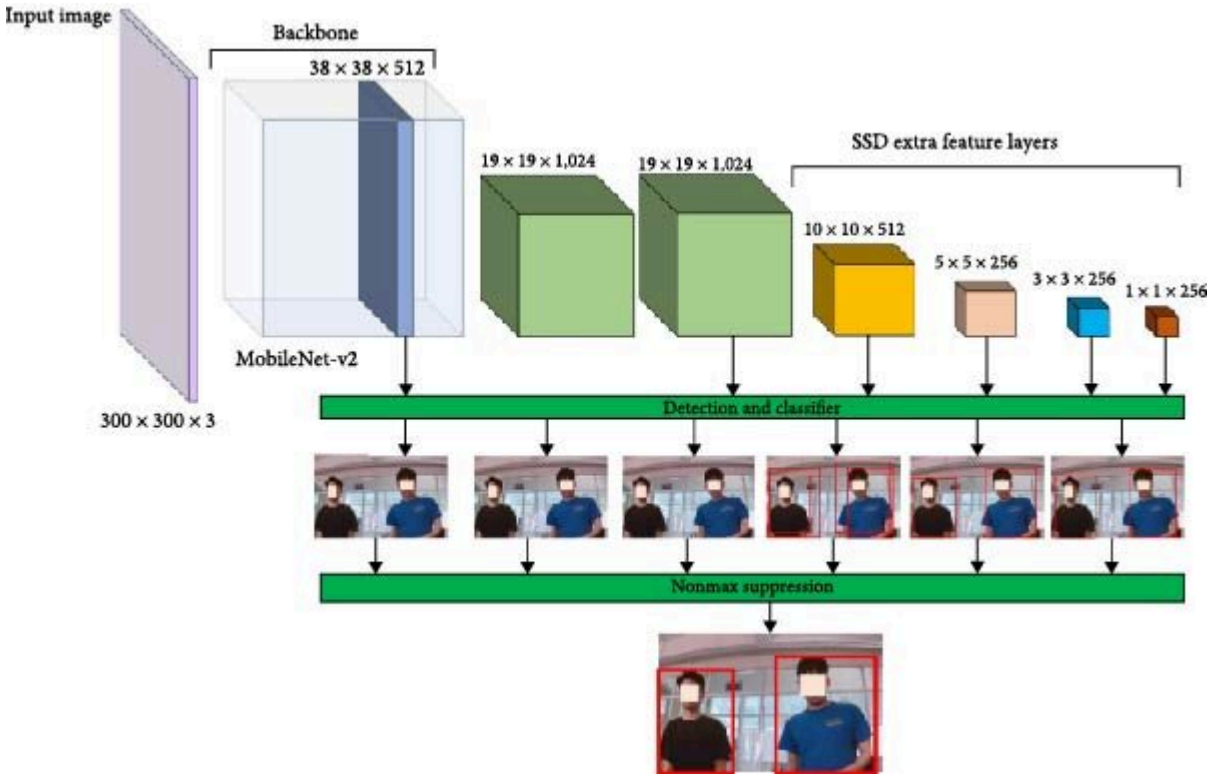

Figure 4 : SSD-MobileNet-v2 architecture [17].

In terms of object detection tasks, especially under limited computational resources scenarios like mobile devices or embedded systems [8], SSD MobileNetV2 has exhibited performance competitiveness. This means that it can be very useful in computer vision applications such as human detection or identification since its inference speed versus accuracy tradeoff ability is valuable across different setups .

3.8 YOLO V8

YOLOv8 is the latest iteration of the YOLO (You Only Look Once) object detection algorithm, which has led the field of real-time object detection since its inception. Over the years, the YOLO architecture has significantly evolved, with each version introducing innovative improvements to enhance performance and accuracy. Building on the success of its predecessors, particularly YOLOv5, YOLOv8 refines its architecture to achieve even more accurate and faster object detection. Notably, YOLOv8 is the most accurate version to date, further setting a new benchmark. The YOLO architecture consists of three main components: the backbone for feature extraction, the neck for integrating features across different scales, and the head for making final detection predictions. YOLOv8 retains this structure while incorporating several modifications to boost its performance.

3.8.1 Backbone

The backbone in the YOLOv8 architecture functions as the feature extractor, processing the input image to derive high-dimensional features crucial for object detection. Key advancements include enhanced depth-wise separable convolutions, which improve efficiency by splitting the convolution operation into depthwise and pointwise convolutions, reducing computational cost and parameters while maintaining performance [25]. Attention mechanisms, such as Squeeze-and-Excitation blocks [7] or Transformer-based modules [35], help the model focus on relevant image areas for better feature extraction. Additionally, YOLOv8 might introduce novel layer structures to optimize information flow and enhance the model's ability to capture complex patterns, essential for detecting various objects.

3.8.2 Neck

The neck in the YOLOv8 architecture plays a critical role in aggregating and refining the features extracted by the backbone, ensuring effective multiscale detection. It integrates features across different scales using mechanisms like Feature Pyramid Networks (FPN), which enhance the network's ability to detect objects of various sizes by creating a pyramid of feature maps at multiple scales, enriching each level with context from both higher and lower levels. Path Aggregation Network (PANet) [14] builds upon FPN by adding bottom-up paths, allowing more effective information flow and feature integration across scales. These mechanisms enable YOLOv8 to accurately detect objects ranging from very small to very large within the same image. The traditional YOLO neck architecture is extended by the C2f module, followed by two segmentation heads that predict semantic segmentation masks for the input image [25]. The model has similar detection heads to YOLOv8, comprising five detection modules and a prediction layer. The YOLOv8-Seg model has achieved state-of-the-art results on various object detection and semantic segmentation benchmarks while maintaining high speed and efficiency.

3.8.3 Head

The head of YOLOv8 is where the actual detection predictions occur, encompassing object classifications, locations, and sizes. It processes the integrated features from the neck to generate bounding boxes and class probabilities. Key aspects may include a hybrid approach combining anchor-based and anchor-free detection methods [6], offering flexibility and precision in detecting a wide range of object sizes and shapes. The adoption of advanced loss functions, such as CIoU loss [6] for bounding box regression, can minimize prediction errors more effectively, enhancing accuracy. Additionally, innovations in object presence and class probability predictions can refine detection performance by reducing false positives and improving class discrimination.

3.8.4 Loss Functions

YOLOv8 utilizes the CIoU (Complete IoU) [35] and DFL (Differentiable Feature Loss) [31] loss functions for bounding-box loss, and binary cross-entropy for classification loss. These loss functions have been shown to improve object detection performance, particularly when dealing with smaller objects.

3.8.5 Architecture of yolo v8 overview

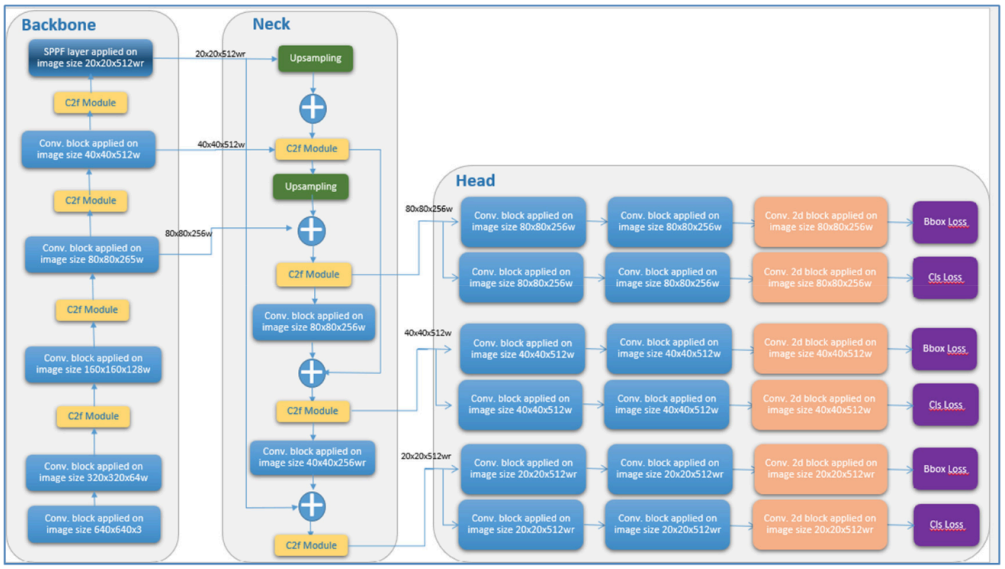

Figure 5 : Architecture of yolo v8 overview [2]

3.9 FaceNet

FaceNet, introduced by Schroff et al. [24], represents a significant advancement in face recognition technology. This deep learning model transforms facial images into compact Euclidean space embeddings, where distances directly correspond to face similarity. FaceNet's architecture consists of three main components: the feature extraction network, the embedding layer, and the triplet loss function.

3.9.1 Feature Extraction Network

n FaceNet’s backbone, the feature extraction network processes input face images and derives high-dimensional features that are critical for face recognition purposes. This usually involves use of a deep Convolutional Neural Network (CNN) which is typically based on the Inception architecture. The network learns to recognize important facial features that are invariant to changes in pose, lightening and expression hence making accurate face recognition possible in many different situations.

Also, Parkhi et al [18] demonstrated the effectiveness of deep convolutional networks for face recognition with their VGGFace model which was built on a very deep convolutional network architecture . According to Wang and Deng [28], this trend has continued with the advent of other more recent models doing facial recognition.

3.9.2 Embedding Layer

he feature extraction network then feeds this output into the embedding layer where it maps onto a fixed-size vector typically 128 dimensional size). This Euclidean representation allows one to measure

similarities directly using distance between embeddings as measured by the distances between each pair of them when calculating all possible pairs' similarity scores within some given range as represented by these vectors as shown in Fig.. The main contribution from Face Net relied on optimizing this space such that same identity images are closer together while those from different identities get far apart thus supporting fast search through large labeled faces dataset.

Learning discriminative face embeddings has been embraced throughout literature for quite some time now. Unlike other models such as Cao et al.'s [19] VGGFace2 dataset that was tailored towards training deep models for face recognition, this dataset has a broad spectrum of face images from different subjects taken under many visual variations.

3.9.3 Triplet Loss Function

The core training mechanism used in Face Net is the triplet loss function which works by using triplets: an anchor image, a positive image (same person as the anchor), and a negative image (different from the anchor). The objective of this model is to minimize the distance between a pair of embeddings representing an anchor and its positive samples while at the same time maximizing their separation from other embeddings resulting into nonoverlapping distributions as shown in Fig.. This was intended to enable clustering of similar images together so that during testing any new images with close similarity will be considered having been captured from one person while different people's faces will be represented by widely scattered points on x-y axes.

This approach has motivated many researches to come up with modifications or improve it over time and Wang and Deng [28] give an overview of various such loss functions for deep face recognition including center loss, angular softmax loss etc aimed at enhancing power of learned embeddings.

3.9.4 Applications and Impact

Moreover, FaceNet's unified embedding approach allows multiple face recognition tasks such as one-to-one matching(face verification), one-to-many matching(face recognition) and clustering based on facial similarity. In particular, it performs outstandingly well when tested on benchmark data sets outperforming human beings on LFW dataset [24].

FaceNet has had wide-ranging influence since its inception. This has led to several systems being developed among others . For example, limitations in existing datasets prompted development of VGGFace2 dataset [19] which contains many more labels thus making it possible to train deep models capable enough to recognize faces even when they change either due to pose or aging etc.

The authors Wang and Deng [28] have concluded that learning discriminative embeddings is the main principle behind FaceNet as a modern face recognition system. This area keeps changing with time and hence, current research focuses on improving resistance to variations, dealing with prejudice and fairness concerns, as well as enhancing scalability.

In conclusion, FaceNet represents a pivotal development in face recognition technology, introducing a unified embedding approach that has significantly influenced subsequent research and practical applications in the field of computer vision and biometric identification.

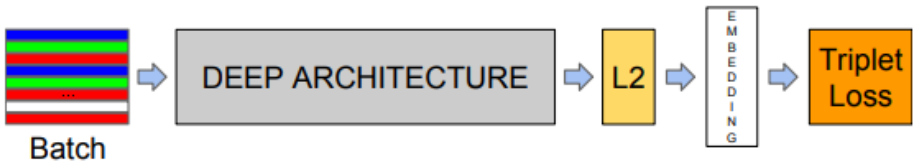

Figure 6 : Model structure. Our network consists of a batch input layer and a deep CNN followed by L2 normalization, which results in the face embedding. This is followed by the triplet loss during training [24] .

3.10 Existing methods

In our research, we have explored various existing facial recognition and analysis systems to identify their features and limitations. The following table (*Table 1*) provides a comparative analysis of these systems. Our proposed solution stands out by offering a free, user-friendly interface that requires no prior knowledge, thus making advanced facial recognition technology accessible to a broader audience. This need became particularly evident on 07.10, when we searched for a system to recognize our friends from the Nova disaster in Telegram videos, and found the process to be extremely challenging.

Feature	Amazon Rekognition [43]	Microsoft Azure Face API [44]	Face++ [45]	Deep Vision AI [46]	Our Solution
Face Detection	V	V	V	V	V
Face Search/Matching	V	V	V	V	V
Video Analysis	V	Limited	Limited	V	V
User-Friendly Interface	X	X	X	X	V
No Prior Knowledge Required	X	X	X	X	V
Cost	Paid	Paid	Paid	Paid	Free
Open-Source	X	X	X	X	V

Table 1 : provides a comparative analysis of Amazon Rekognition, Microsoft Azure Face API, Face++ Deep Vision AI and Our Solution highlighting their capabilities in face detection,face search, video analysis, and user accessibility.

Table 1 clearly shows how our proposed system differentiates itself by being free, user-friendly, and requiring no prior technical knowledge, making it accessible to a wider audience.

4 Research / Engineering Process

4.1 Development methods

4.1.1 Python for AI and Machine Learning as server

Python offers numerous benefits; its simplicity and readability are ideal for rapid development and prototyping. The language boasts a rich ecosystem of libraries and frameworks, such as OpenCV, Dlib, TensorFlow, and PyTorch, which provide pre-built tools for computer vision and machine learning tasks. Additionally, Python's extensive community support and a wealth of online resources make problem-solving and collaboration much easier. Its versatility and ability to integrate seamlessly with other technologies make Python a powerful and efficient choice for developing advanced AI and neural network-based applications [49].

4.1.2 React for the Frontend

We are going to use React for the frontend because it provides a flexible and efficient way to build user interfaces. React's component-based architecture promotes reusability and maintainability, making it easier to manage complex UIs. Its virtual DOM ensures high performance by minimizing updates to the real DOM. Additionally, React has a strong developer community and a rich ecosystem of tools and libraries, which accelerates development and enhances productivity [48].

4.2 Choosing Our YOLO

In the quest to identify the most suitable model for high-accuracy person recognition tasks, this section compares two advanced object detection frameworks: YOLOv5 and YOLOv8. Through a detailed analysis of their architectures and performance metrics , we demonstrate why YOLOv8 emerged as the superior choice for our specific application needs. Object detection has evolved significantly with the advent of YOLO (You Only Look Once) models, renowned for their real-time detection capabilities and accuracy. YOLOv5 and YOLOv8 represent key iterations in this evolution, each offering unique advantages. This paper provides an in-depth comparison of these models to determine the optimal choice for person recognition tasks.

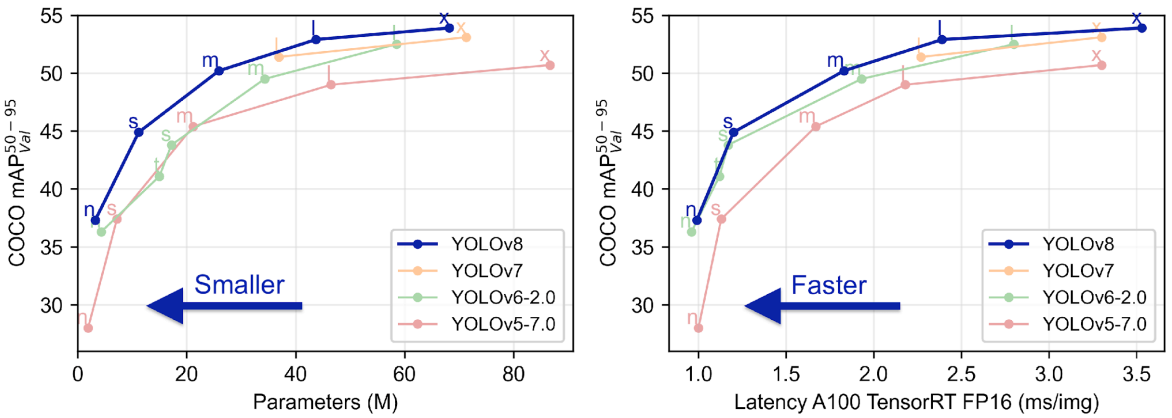

Figure 7 : Comparison of the YOLO models [37]

Figure 7 compares the performance of various YOLO models (YOLOv8, YOLOv7, YOLOv6-2.0, and YOLOv5-7.0) on the COCO dataset [40] using two metrics: the number of parameters (left) and inference latency (right). The left plot shows the mean Average Precision (mAP) as a function of the number of parameters, highlighting that YOLOv8 achieves higher mAP with fewer parameters compared to previous versions. The right plot illustrates mAP versus latency, demonstrating that YOLOv8 models are both faster and more accurate, achieving superior performance at lower latencies. Overall, YOLOv8 provides an efficient and speedy solution for object detection tasks.

4.2.1 Performance Metrics

The performance of YOLOv5 and YOLOv8 was evaluated based on precision, recall, and mean Average Precision (mAP).

Metric	YOLO v 5	YOLO v 8
Precision	89.3%	91.6%
Recall	74.4%	83.1%
mAP50	85.2%	88.5%
mAP50-95	58.5%	60.0%

*Table 2 :*Performance comparison using YOLOV5 and YOLOV8 [2]

YOLOv8 consistently outperformed YOLOv5 across all metrics , demonstrating its superior capability in detecting and classifying objects accurately. The inference speed of both models was also a critical factor. YOLOv8, with its simplified architecture, provided faster inference times compared to YOLOv5, making it more suitable for real-time applications.

4.2.2 Chosen Model

Based on our comprehensive analysis , YOLOv8 was chosen over YOLOv5 for our person recognition tasks. Its advanced architecture, higher accuracy, faster inference times, and reduced training complexity made it the ideal candidate. YOLOv8's robust performance in diverse conditions ensures reliable and accurate person detection, essential for our application needs.

4.3 Proof of scraping feasibility

We developed an advanced web server designed to efficiently extract and download images and videos from any specific webpage. This tool streamlines the process of gathering media content by automating the retrieval and packaging of these files into a single, convenient zip file. Users simply input the desired URL, and the application performs the tasks of fetching the webpage content, identifying and downloading media files, and compressing them for easy download. This systematic approach not only saves time but also ensures that users receive their content in an organized and accessible manner.This shows us the feasibility of bringing additional sources of information to search and identify a person in cases where we would like to perform proactive searches of the system on certain sites of interest

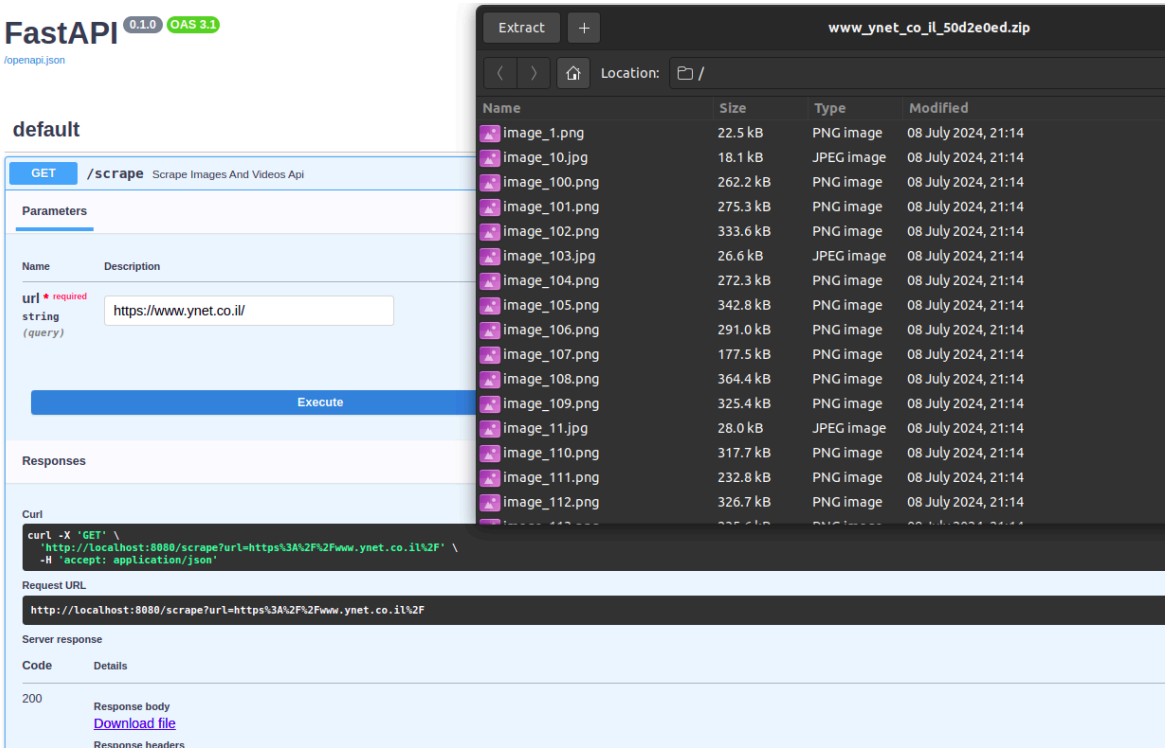

Figure 8 : server in action, realized by fast-api in python , Example of entering a sample URL of an internet site, and we received back a zip file that contains all the images and videos on the page.

This way we can enable the expansion of our search sources past figures and searches initiated by the system

3.2.1 Comparison between models

4.3.0.1 Comparison of MobileNetV2 and YOLOv8

Over the years, various deep learning architectures have been developed to tackle this challenge efficiently. Among them, MobileNetV2 and YOLOv8 stand out as popular choices due to their effectiveness and speed. This essay aims to provide a detailed comparison between MobileNetV2 and YOLOv8, focusing on their architecture, performance, and practical considerations.

Object detection models, particularly those leveraging convolutional neural networks (CNNs), have significantly advanced automated systems' capabilities to identify and locate objects within digital images. Models like YOLOv8 and MobileNetV2 SSD, though distinct in design and application, share several fundamental methodologies. Both utilize CNNs to process visual data through layers of convolutions, building a hierarchy of features from basic edges to intricate objects. They generate bounding boxes with spatial coordinates and confidence scores, classifying each detected object into predefined categories. Anchor boxes streamline detection by offering reference points for various object sizes and shapes. Efficient feature extraction networks produce detailed feature maps crucial for detection tasks. Both models employ non-maximum suppression to remove redundant bounding boxes, enhancing precision. Operating on the single-shot detection principle, they perform localization and classification in one network pass. YOLOv8 is designed for real-time use, balancing accuracy and speed, while MobileNetV2 SSD is optimized for performance in environments with limited computational resources. These shared methodologies highlight a commitment to advancing object detection technologies, reflecting the convergent evolution of neural network architectures towards robust, real-time visual recognition systems.

Feature	YOLO v8	MobileNet V2 SSD
Date of Release	January 10, 2023	January 13, 2018
Model Type	Object Detection	Object Detection
Framework	PyTorch	TensorFlow
GitHub Stars	21.1k +	81+
Mean Average Precision	~55%	~35%
Number of Parameters	Highly variable , Average 30.78 million	Around 3.5 million (varies with implementations)
Number of Layers	Deep network, number varies by specific variant	Generally around 53 layers including depthwise convolutions
Processing Speed (FPS)	Up to 140 FPS depending on hardware and settings	

Computational Requirements	High, benefits significantly from GPU acceleration	Designed for efficiency, can run on CPUs with reasonable performance
Training Data Size	Extensive data required for optimal performance	Can be trained effectively with moderately sized datasets
Hardware Requirements	High-end GPUs recommended for training and inference	Can run on less powerful devices like smartphones
Optimization Techniques	Uses techniques like batch normalization, multi-scale predictions	Uses depth wise separable convolutions for efficiency
Typical Use Case Scenarios	High-speed real-time detection in various conditions	Efficient detection in resource-constrained environments like mobile devices
Detection Capabilities	Excellent at detecting small to large objects with high precision	Good at detecting moderate to large objects, may struggle with very small objects
Flexibility	Highly flexible with adjustments for various performance metrics	Somewhat flexible, mainly constrained by hardware efficiency
Energy Consumption	Generally higher due to more complex computations	Lower, optimized for energy efficiency especially in mobile devices

Table 3 : comparison YOLOv8 and MobileNet SSD v2 [38]

YOLOv8 stands out for its impressive accuracy and precision, especially when it comes to spotting small or overlapping objects, which is crucial for detailed object detection tasks .This high level of accuracy is due to its advanced feature extraction techniques and a finely-tuned architecture. Although MobileNetV2 SSD isn't as precise as YOLOv8, it still handles many object detection tasks well, particularly in situations with limited computational resources [32]. When it comes to speed, YOLOv8 is built for rapid processing on powerful hardware, making it perfect for real-time detection scenarios. On the other hand, MobileNetV2 with SSDLite might not be as quick on high-end devices but offers good detection speeds on mobile and embedded systems [3]. YOLOv8's complexity means it has a larger model size and demands more memory and processing power. In contrast, MobileNetV2 with SSDLite is designed to be lightweight, which reduces its size and lowers memory and processing requirements, making it highly efficient.

4.3.0.2 Experimental Comparison of MobileNetV2 and YOLOv8 for Object Detection on Diverse Image Datasets

In this study, we conducted a comprehensive comparison between MobileNetV2 and YOLOv8 for object detection on diverse image datasets. Our evaluation focused on performance metrics such as detection time, average confidence scores, and accuracy in detecting persons within images. We employed a dataset comprising 1000 images captured at three different resolutions, with ground truth annotations indicating the actual number of persons in each image. By analyzing the results, we aimed to provide insights into the strengths and weaknesses of each model under varying conditions.

4.3.0.2.1 Experimental Setup:

We ran both MobileNetV2 and YOLOv8 on our local machine using cpu (11th Gen Intel® Core™ i7-11800H @ 2.30GHz × 16) , which was equipped with standard hardware configurations. The dataset consisted of 1000 images, categorized into three groups based on resolution: low (480x360), medium (960x720), and high (1920x1080). Each image was manually annotated to indicate the exact number of persons present, serving as ground truth for evaluation purposes. We measured detection time, average confidence scores, and the delta between the detected and actual number of persons for each model and resolution.

4.3.0.2.2 Test Results :

MobileNetV2 exhibited faster inference times compared to YOLOv8 across all resolutions. The lightweight architecture of MobileNetV2 contributed to its efficient processing speed, making it particularly suitable for real-time applications.

YOLOv8, while slightly slower than MobileNetV2, still demonstrated impressive performance in processing high-resolution images in near real-time. Its more complex architecture resulted in longer inference times but remained within acceptable limits for most practical applications.

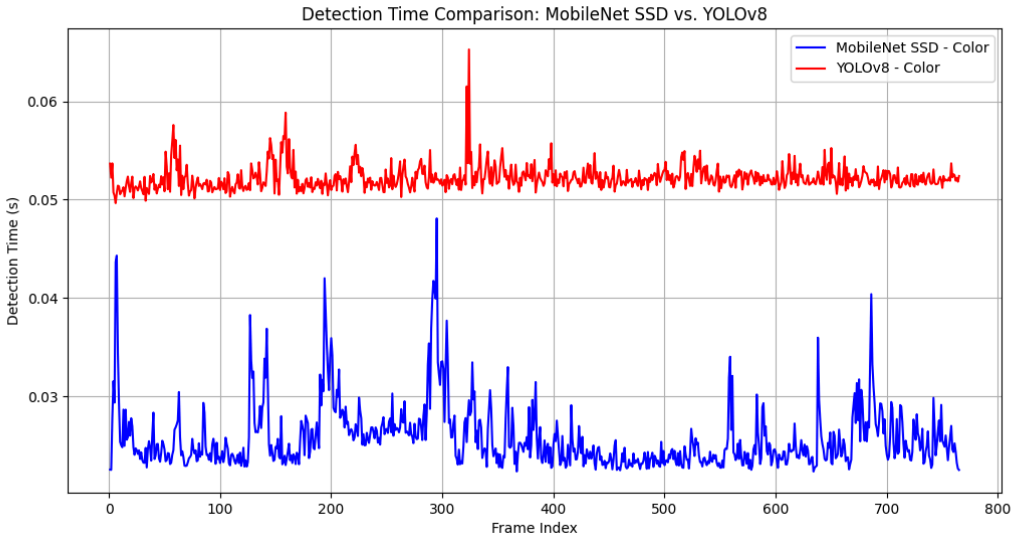

Figure 9 : comparison of detection times between MobileNet SSD and YOLOv8.

In this example MobileNetV2 SSD detected several individuals but missed three, including one who was fully visible, highlighting a limitation in its detection capability. In contrast, the YOLOv8 l model successfully recognized everyone in the scene, showcasing its superior detection performance. We can see this in an example image from our dataset, where the prediction results from each model are displayed in the figure.

Figure 10 : Prediction results of the YOLO model and MobileNetV2 SSD are displayed simultaneously alongside the real drivers and the original image.

MobileNetV2 is renowned for its efficiency, particularly in environments with limited computational resources. It utilizes depth wise separable convolutions, a design that significantly reduces the computational overhead, thereby enhancing execution speed on less capable devices. On the other hand, YOLOv8, representing the latest in the evolution of the You Only Look Once (YOLO) series, is optimized for high-speed object detection with a design that leverages complex models to deliver high accuracy and fast processing times, particularly when supported by GPU acceleration. This comparative study is grounded on empirical benchmark tests conducted under uniform testing conditions. Both models were evaluated using the same dataset and hardware setup, with metrics such as detection time and accuracy being recorded. These tests provide a comprehensive basis for

assessing which model performs better in terms of speed and resource efficiency, contributing valuable insights into their suitability for various applications.

4.4 Image Processing Formats: Color vs. Grayscale in Object Detection Models

Transitioning object detection images from color to grayscale has important implications for computational efficiency and detection accuracy in computer vision. This section explores how the performance of state-of-the-art models, YOLOv8 and MobileNetV2 SSD, changes with grayscale imaging. Through this comparative study, we identify challenges and opportunities, evaluating both theoretical and practical aspects of deploying grayscale imaging within various architectural frameworks. The relationship between image formats and model performance in object detection highlights the need for optimal computational resources to achieve high detection accuracy. While grayscale imaging can offer computational advantages in certain scenarios, it may also lead to significant performance degradation, particularly in complex environments where color is crucial for distinguishing objects. Our analysis reveals trade-offs between processing time and model accuracy, investigating whether frame detection in grayscale impacts the performance of YOLOv8 and MobileNetV2 SSD.

4.4.1 Impact on Processing Time

Both YOLOv8 and MobileNetV2 SSD models traditionally operate on RGB images. Transitioning to grayscale reduces the input channels from three to one, simplifying the data and potentially increasing processing speed. Our comprehensive tests showed that converting frames to grayscale slightly accelerates processing time [figure 11]. This efficiency gain stems from reduced data complexity: grayscale images handle a single data channel, significantly lowering memory bandwidth and computational power requirements compared to three-channel RGB images. This simplification speeds up data processing, advantageous in real-time detection scenarios. Additionally, convolutional neural networks (CNNs) underlying these models perform less complex internal computations with fewer input channels, resulting in quicker propagation through network layers and enhancing overall frame processing speeds.

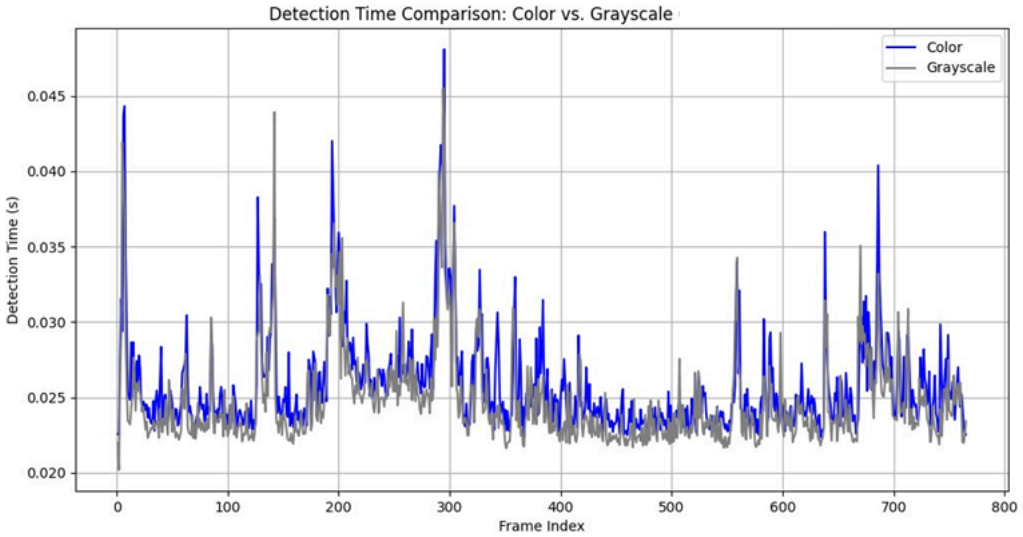

Figure 11 : This graph provides a visual comparison of detection times when using Yolo v8 l with color images versus grayscale images. The graph illustrates that grayscale processing achieves slightly faster detection times, as anticipated, due to the reduced computational load from processing fewer data channels. However, the difference in detection speeds between color and grayscale is not substantial, suggesting that the decrease in data complexity does not translate into a dramatic improvement in processing efficiency.

4.4.2 Impact on Detection Performance

Using grayscale imaging can pose significant challenges, especially in environments where color cues are essential for distinguishing people from backgrounds or from one another. Color is a critical factor in differentiating objects, particularly when they share similar shapes or are set against backgrounds with similar textures. Studies by Kanan and Cottrell (2012) indicate that object recognition tasks often perform poorly in grayscale compared to color, as many rely heavily on color differences. This issue is most pronounced in scenes where multiple objects of interest are mingled, with color being the primary distinguishing feature[9].

4.4.3 Decreased Performance in Variable Lighting

Variable lighting conditions, which are common in both indoor and outdoor environments, can affect the visibility of features in grayscale images more than in color images. Color can provide consistent identification markers that are less susceptible to changes in illumination, such as distinguishing features in low light conditions where grayscale images might suffer from insufficient contrast [22].

Also feature recognition within person detection systems often relies on subtle color cues that can indicate depth, contour, and even emotional or physical states (e.g., clothing logos, uniform colors, or skin tones under different lighting conditions). Grayscale imaging flattens these cues, potentially reducing the effectiveness of algorithms that rely on these subtle distinctions to identify and classify human figures accurately [27].

4.4.4 Training and Model Adaptation:

To optimize for the absence of color data, both YOLOv8 and MobileNetV2 SSD would ideally require retraining on grayscale datasets. This retraining is essential to maintain or potentially improve detection performance with grayscale inputs. YOLOv8 might need more significant architectural adjustments than MobileNetV2 SSD due to its reliance on complex color-based feature layers.

4.4.5 Experimental Analysis on YOLO V8-l Object Detection Confidence

In this test, we evaluated the performance of the YOLO V8 object detection model by comparing its confidence levels when processing color versus grayscale images. Over approximately 800 frames, we recorded the model's average confidence in identifying and classifying objects. Our objective was to see if converting images to grayscale, which reduces data complexity and offers computational benefits, would affect the model's detection performance.

4.4.6 Experimental Analysis Results

The results, shown in Figure 12, clearly indicate that grayscale processing does not improve and actually diminishes the model's confidence in object detection tasks. This finding reinforces that while grayscale may reduce computational load, it sacrifices essential information needed for effective object recognition.

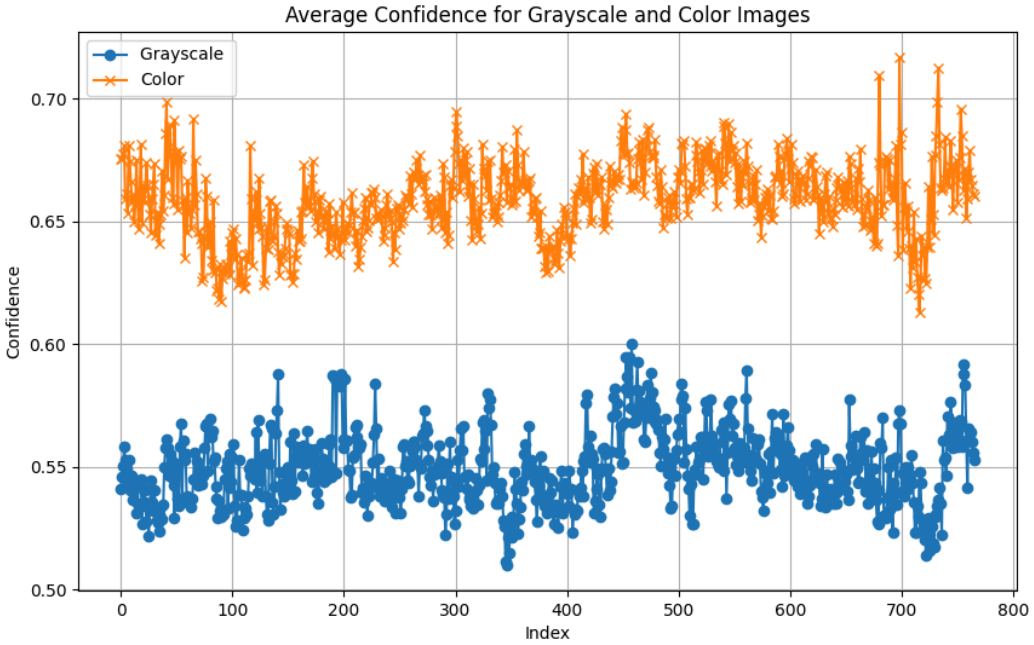

Figure 12 : This graph displays the average confidence levels of object detection using YOLO V8 l , contrasting color processing with grayscale. The data is plotted across approximately 800 frames, highlighting the confidence with which the model identifies and classifies objects in each frame.

4.4.7 Decision

The tests on the effects of image processing formats on object detection models revealed a complex balance between computational efficiency and performance accuracy. While grayscale processing offers a slight speed gain, it significantly reduces detection performance. Therefore, to ensure operational effectiveness and reliability, especially in varied and challenging real-world conditions, we chose to retain color processing.

4.5 Methods to measure precision of object detection

4.5.1 Recall

Recall (also known as Sensitivity) measures how many of the actual objects in the dataset are detected by the model. It is defined as the ratio of true positive detections to the total number of ground truth objects (true positives + false negatives).

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives}$$

4.5.2 Precision

Precision is a measure of how many of the objects detected by the model are actually correct (i.e., true positives). It is defined as the ratio of true positive detections to the total number of detections made (true positives + false positives).

$$\Pr ecision = \frac{TruePositives}{TruePositives + FalsePositives}$$

4.5.3 Average Precision (AP)

Average Precision is a measure that combines both precision and recall to provide a single metric for model performance. It is calculated as the area under the precision-recall curve. The precision-recall curve plots precision (y-axis) against recall (x-axis) for different threshold values.

AP is usually computed for different Intersection over Union (IoU) thresholds, and the mean of these AP values gives a comprehensive view of the model's performance.

$$AP = \int_0^1 Pecision\ (Recall)\ \ d(Recall)$$

4.5.4 Average Recall (AR)

Average Recall is the average of recall values computed for different IoU thresholds and object categories. It provides an overall measure of the model's ability to detect objects across all categories and IoU thresholds.

$$AR = \frac{1}{N} \sum_{i=1}^{N} Recall_i$$

4.5.5 Mean Average Precision

Mean Average Precision (mAP) is the average of the AP values computed for different IoU thresholds and object categories. It provides an overall measure of the model's performance across all categories and IoU thresholds.

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \ \mid N\ Is\ The\ Number\ Of\ IoU\ Thresholds.$$

4.5.6 Intersection over Union (IoU)

Intersection over Union (IoU) is a metric used to evaluate the accuracy of an object detection model. It measures the overlap between two bounding boxes: the predicted ounding box and the ground truth bounding box. The IoU value ranges from 0 to 1 where :

- 0 indicates no overlap between the bounding boxes.
- 1 indicates a perfect overlap between the bounding boxes.

IoU is calculated as the ratio of the area of overlap between the predicted bounding box and the ground truth bounding box to the area of their union.

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union}$$

```python
def compute_iou(box1, box2):
    inter_left = max(box1['left'], box2['left'])
    inter_top = max(box1['top'], box2['top'])
    inter_right = min(box1['right'], box2['right'])
    inter_bottom = min(box1['bottom'], box2['bottom'])

    # Calculate intersection width and height
    inter_width = inter_right - inter_left
    inter_height = inter_bottom - inter_top

    # If there is no overlap, the intersection area is zero
    if inter_width <= 0 or inter_height <= 0:
        return 0.0

    # Calculate intersection area
    intersection_area = inter_width * inter_height

    # Calculate areas of the bounding boxes
    box1_area = (box1['right'] - box1['left']) * (box1['bottom'] - box1['top'])
    box2_area = (box2['right'] - box2['left']) * (box2['bottom'] - box2['top'])

    # Calculate union area
    union_area = box1_area + box2_area - intersection_area

    # Calculate IoU
    iou = intersection_area / union_area

    return iou
```

Figure 13 : Pseudo-code for calculating IoU

Figure 14 : IoU Example [39]

4.6 COCO Dataset

The COCO (Common Objects in Context) dataset [40] , introduced in 2014, is a pivotal tool in computer vision and natural language processing, widely used for object detection, segmentation, and captioning. The COCO 2017 Validation Dataset, a crucial subset, comprises 5,000 images meticulously selected and annotated to capture a broad spectrum of everyday scenes, both indoor and outdoor, with diverse complexities. This dataset's comprehensive annotations, including instance segmentation masks and bounding boxes, go beyond simple categorization, providing pixel-level detail that aids in developing sophisticated models. There are usually two common data sets used by

researchers, namely COCO 2017 Training Dataset used for training the models and the COCO Validation Dataset that can be used for testing the general performance of the model on unseen data.. Metrics such as mAP, AP, and AR provide a comprehensive analysis on the model's performance by evaluating localization and classification precision[15].

The benchmark for object detection and segmentation and captioning purposes is the COCO (Common Objects in Context) dataset [40] was introduced in 2014. COCO 2017 contains four important subsets: COCO 2017 Train Dataset, COCO 2017 Validation Dataset, COCO 2017 Test – Dev Keypoints, and COCO 2017 Test – Small. Additionally, this dataset has other annotations other than categorizing as it has instance segmentation masks and bounding boxes that give a pixel level information that can be useful in coming up with complex models. To eliminate the weaknesses in the model, there are usually two common data sets used by researchers, namely COCO 2017 Training Dataset used for training the models and the COCO Validation Dataset that can be used for testing the general performance of the model on unseen data..

4.7 Filtering and Evaluating People in the COCO Dataset

In this project, we used the COCO 2017 Validation Dataset, a smaller part of the larger COCO dataset, to test how well our object detection model works, focusing on images with people. We used the COCO API from pycocotools [41], a Python library, to filter images containing at least one person by checking for "person" tags. This way, we ensured our results were relevant and specific. The COCO API also helped us evaluate our model's performance using metrics like mean Average Precision (mAP) and Average Recall (AR), giving us a clear understanding of its accuracy. Overall, using the COCO API and pycocotools was crucial for effectively filtering, evaluating, and improving our model with the COCO 2017 Validation Dataset, ensuring it was well-tested and capable of performing on new data.

4.7.1 Visualization of the Data Using FiftyOne

To increase the effectiveness of our analysis of the filtered dataset, we employed FiftyOne [42] which is an open-source toolkit for visualizing and analyzing datasets in computer vision. Thus, FiftyOne helped to interactively investigate the COCO 2017 Validation Dataset, narrowing it down to the images containing people. It would also be easy to visually compare the ground truth annotations with the model predictions, thus giving us an indication of how well the model is performing based on the predicted bounding boxes and segmentation masks against the ground truth. Also, FiftyOne offered functionalities to plot precision-recall curves and confusion matrices, which helped to gain more profound understanding of the model's performance. This assisted in finding out areas that needed to be considered. The use of FiftyOne also enabled the inspection of specific samples, for example, zoom into a particular area of the image, select samples by criteria, and visualization of the predictions of samples by class. Such a level of analysis was important for refining our model and improving its efficiency.

4.7.1.1 Visualization Example

The dataset used for this evaluation is the COCO 2017 validation set, which can be found at COCO 2017 Dataset , or just download from here [40] . The dataset contains 5,000 images include 2,693 images with the "person" category.

```python
import fiftyone as fo
import fiftyone.zoo as foz
from fiftyone import ViewField as F

# Download and load the validation split of COCO-2017
dataset = foz.load_zoo_dataset( name: "coco-2017", split="validation")

# Filter the dataset to include only non-crowded 'person' annotations
view = dataset.filter_labels(
    "ground_truth",  # The field containing the annotations
    (F("label") == "person") & (F("iscrowd") == 0)  # Filter criteria
)

# Launch the FiftyOne app with the filtered view
session = fo.launch_app(view, port=5151)
```

Figure 15 : the code to LOAD DATA SET AND show the data USING FIFTYONE

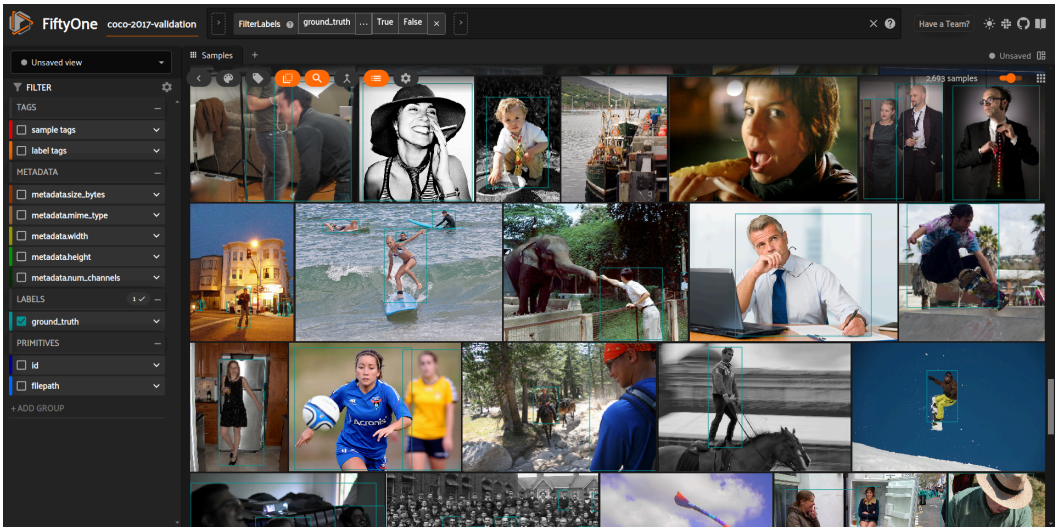

Figure 16 : FIFTYONE's user interface is convenient and well-designed for displaying the dataset.

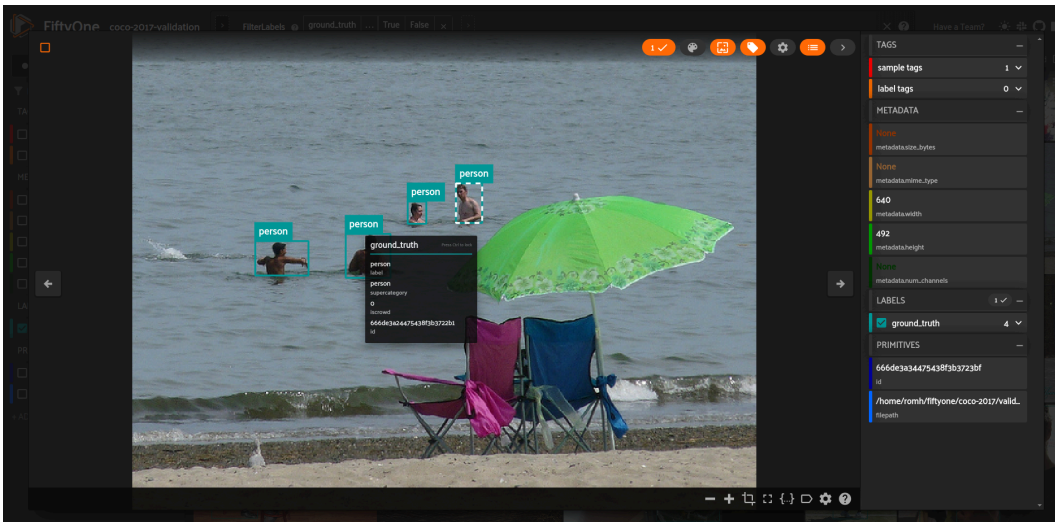

Figure 17 :In the figure, you can easily see the data, including locations and labels, with the option to edit the information.

4.8 YOLOv8 Model Evaluation Results on COCO Validation Dataset

This test's objective is to evaluate the YOLOv8 model's performance in detecting the "person" category on the COCO validation dataset. The evaluation focuses on comparing the predicted number of persons in each image to the actual (ground truth) number of persons and calculating key metrics such as Average Precision (AP), Average Recall (AR), and Mean Average Precision (mAP) .

4.8.1 Dataset

COCO 2017 Validation Dataset: The dataset used for this evaluation is the COCO 2017 validation set, which can be found at COCO 2017 Dataset.
Number of Images: The dataset contains 2,693 images that include the "person" category.

4.8.2 Results

Average Precision (AP)

AP [IoU=0.50:0.95 | area=all | maxDets=100] = 0.360
This metric indicates the mean average precision across multiple IoU thresholds (0.50 to 0.95) for all object sizes. A value of 0.360 suggests moderate precision in detecting persons.
AP [IoU=0.50 | area=all | maxDets=100] = 0.466
This represents the precision at an IoU threshold of 0.50. The higher value (0.466) indicates better performance at lower IoU thresholds, showing the model's ability to detect objects with lower overlap accuracy.
AP [IoU=0.75 | area=all | maxDets=100] = 0.401
This indicates precision at an IoU threshold of 0.75. The value (0.401) shows that the model maintains decent precision even at higher IoU thresholds.

AP for Different Object Sizes:

For small objects, the AP = 0.173, indicating lower precision and suggesting difficulty in accurately detecting smaller persons. For medium-sized objects, the AP = 0.446, reflecting higher precision and better detection capability for moderately sized persons. The highest precision, AP = 0.618, is observed for large objects, demonstrating the model's strength in detecting larger persons with distinct features.

Average Recall (AR)

AR [IoU=0.50:0.95 | area=all | maxDets=1] = 0.288 indicates the recall when considering only the top detection per image, reflecting the ability to detect at least one person correctly in most images.

AR [IoU=0.50:0.95 | area=all | maxDets=10] = 0.394 shows recall for the top 10 detections per image, indicating improved recall with more detections allowed.

AR [IoU=0.50:0.95 | area=all | maxDets=100] = 0.397 represents recall for up to 100 detections per image, highlighting the model's overall recall capacity.

AR for Different Object Sizes:

For small objects, AR = 0.183, indicating lower recall and aligning with the observed lower precision, which suggests difficulty in detecting smaller persons. For medium-sized objects, AR = 0.485, reflecting higher recall and better detection capability. The highest recall, AR = 0.672, is observed for large objects, demonstrating the model's effectiveness in detecting larger persons.

Mean Average Precision (mAP)

mAP = 0.360
The overall mean average precision across all IoU thresholds and object sizes is 0.360. This value reflects a respectable performance but indicates room for improvement, particularly for smaller objects. An mAP of 0.360 is respectable for initial evaluations but suggests there is room for improvement, especially in detecting smaller objects.

4.8.3 Insights

The model performs better on larger objects, showing higher precision and recall since larger objects usually have more distinguishable features. The overall mAP is 0.360, which is respectable for initial evaluations but indicates there is room for improvement, particularly in detecting smaller objects. The precision and recall values are moderate, suggesting a balanced model that could still benefit from further fine-tuning and optimization. Additionally, the average delta between predicted persons and ground truth is 1.2228 which is pretty good .

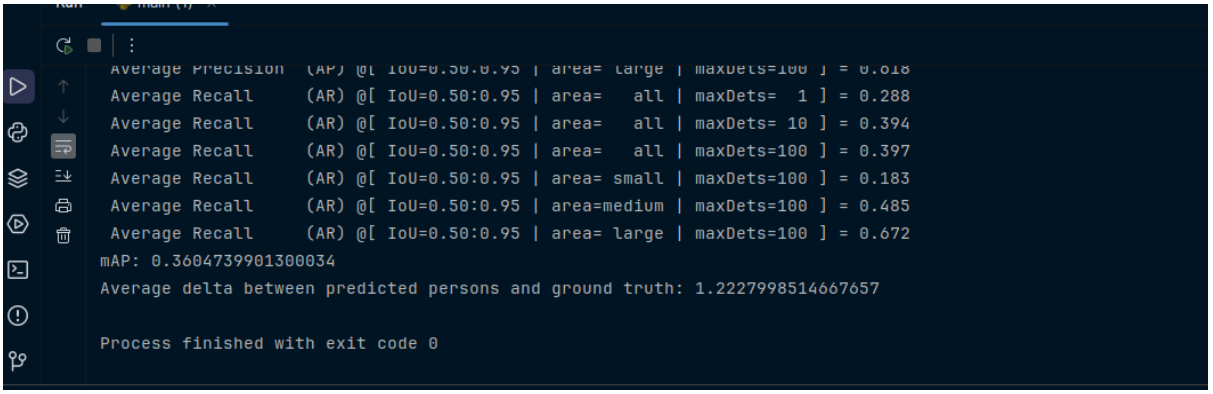

Figure 18 : result of test the Yolo model prediction on the Coco-Validation-2017 data set

```
model = Load object detection model
dataset = Load dataset annotations // Load the Dataset
filtered_images = Filter images containing target category
ground_truth_counts = Initialize storage // Initialize Result Storage
predicted_counts = Initialize storage
image_ids = Initialize storage
deltas = Initialize storage
results = Initialize storage for formatted predictions

FOR each image IN filtered_images DO  // Iterate Over Filtered Images
  image_data = Load image data
  // Retrieve ground truth annotations
  ground_truth_annotations = Get ground truth annotations for target category
  ground_truth_count = Count ground_truth_annotations
  Store ground_truth_count
  Store image identifier
  // Run the model to predict objects
  predictions = model.predict(image_data)
  // Count predicted objects
  predicted_count = Count predictions for target category
  Store predicted_count
  // Calculate delta
  delta = ABS(predicted_count - ground_truth_count)
  Store delta
  // Format predictions for evaluation
  FOR each prediction IN predictions DO
    IF prediction meets confidence threshold THEN
      formatted_prediction = Format prediction
      Store formatted_prediction in results
    END IF
  END FOR
END FOR

Write results to file  //// Save Predictions
// Load Ground Truth and Predictions for Evaluation
ground_truth = Load ground truth annotations
predictions = Load predictions from file
// Initialize Evaluation Tool
evaluation_tool = Configure evaluation tool with ground truth and predictions
// Run Evaluation
evaluation_tool.evaluate()
evaluation_tool.accumulate()
evaluation_tool.summarize()
```

Figure 19 :Yolo model prediction on the Coco-Validation-2017 data set pseudo code

4.9 Person Detection in Videos for Missing Persons

4.9.1 Locating missing people

Person identification in videos might be instrumental to finding missing persons and resolving cases of abduction as well. Authorities can go through large quantities of video recordings fast and accurately by employing sophisticated technologies such as facial recognition and object detection.

4.9.2 Case Study: The Nova Incident and Technological Utilization

We spoke with “Yam Sahar” who told us about his struggle trying to find his girlfriend’s face among the many Telegram videos, which were taken during the Nova incident on 07.10. This backbreaking task highlights the necessity for automated systems capable of recognizing individuals in crowded or chaotic environments that speed up searches while increasing chances for success in terms of resolution rate.These technologies have the ability analyze different sources’ features against each other hence making it easier to locate missing persons in real time as well providing critical leads towards kidnapping investigation.

4.10 Performance Improvement

4.10.1 Improving Results by Focusing on the Region of Interest

Major component of building reliable person identification systems lies in optimizing the input data for the face recognition model. Formerly conducted studies indicate that performance gains can be realized by cropping out the region of interest (ROI) that contains the target individual compared to using an uncropped image [18, 24]. In other words, when presented with a cropped image, the face recognition model becomes capable of focusing only on important facial attributes thus yielding more reliable similarity scores consistently; however, this may not happen if we give it an un-cropped one since there could be diversions from other individuals or background components which might reduce its precision in matching against the intended person.

For us to probe deeper into this matter therefore, we carried out an experiment where similarity distributions and processing times were compared between cropped and uncropped images. Figure 20 shows that while similarity distribution pattern recorded by a histogram plot for similarities among different persons is much tighter around higher values among those who had their images cropped than those without cropping; this finding also holds true for wider distributions as well as lower average similarity scores obtained from using uncropped photos which may imply that sometimes the model gets sidetracked by irrelevant parts within full frames leading to less accurate person identification.

The significance of cropping can be further appreciated based on processing time evaluation depicted in Figure 21. On average it takes significantly shorter period processing time for systems dealing with ROI rather than full frame images; this is due to reduced computational burden when

analyzing only part selected as relevant by system designers during setup stage [20]. Therefore faster speeds are desirable because they allow devices/systems deployed at various locations such as airports, bus stations etc., operate more efficiently towards detecting wanted suspects through watching videos taken from CCTV cameras installed along exit points controlling access into these facilities.

Consequently we have decided to include a step which crops pictures before running them through our face recognizer model. By singling out someone we want to focus on – this could help us improve accuracy and speed of the face matching process especially when there are many people or things moving around in front of camera. Such an approach ensures that system concentrates only on necessary features so as to deliver precise results while identifying individuals

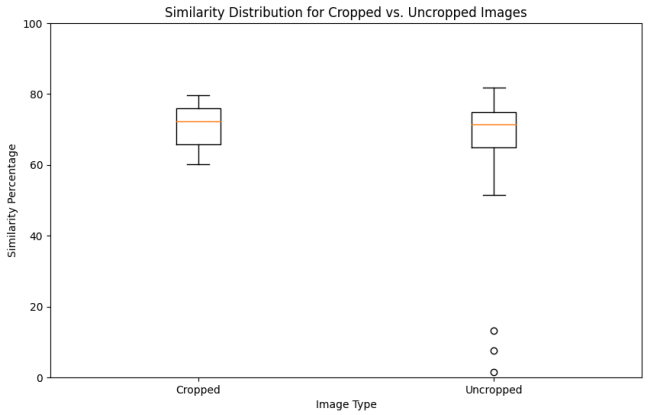

Figure [20] : Similarity Distribution for Cropped vs. Uncropped Images

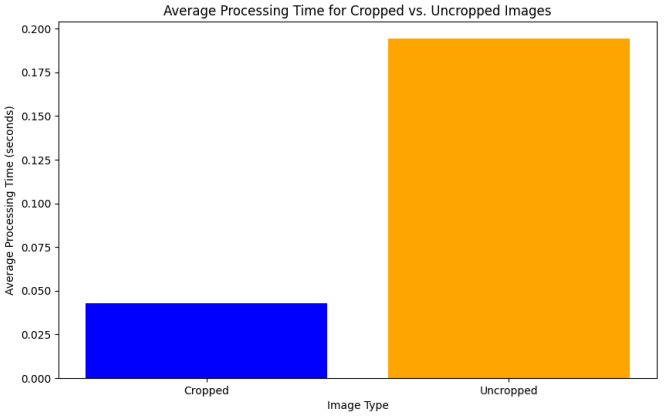

Figure [21] : Average Processing Time for Cropped vs. Uncropped Images

4.10.2 System Identification Improvement

The main concept for the enhancement of the system identification is for the system to become wiser with time. To begin with, the system identifies an individual by photos that the users upload on the application. Every time an individual is detected in an image, the image is stored in the database; the individual is also assigned a series of identification numbers.
And that is not the only system that the show has set into place. It even goes further in the sense that it searches through all the past runs in which this person was detected. Each time a person is found, as well as new images of a person detected by the user are analyzed, the system returns to all previous frames of previous searches with a high similarity.
After it has accumulated all these embeddings: new images and frames that have been detected before, it then checks to see if the new embedding in question is distinct from the rest of the ones that are stored in the system. The reason it does this is because each new embedding is compared with all the others and so only embeddings with a similarity low enough to all the already stored embeddings are added. The data filtering process plays a major role to ensure that end up with only a high quality data set containing only distinctive data rather than replicates.The individual and concatenated embeddings are stored under the UUID of the user that generated it and therefore, we have a diverse and precise dataset about such person. The system just keeps on getting better and

better as their database is fine-tuned with new data that has not been seen in the former searches by the system. However, as the system acquires data from other users and previous searches, the rates of recognition can be next to perfect and the tool can prove to be very useful in identifying people across many of the videos.

4.10.3 Checking The Similarity Level Of Two embeddings

We are using a facenet model for face recognition. We get for each face that we use the model to predict embedding vectors in size 512 . We use this result to find the similarity level of two embeddings. Cosine similarity is a kind of value that identifies the level of resemblance of two vectors in this case, face embeddings. It is used to determine the cosine of the angle between two vectors in the multidimensional space. The actual calculation we do between these two vectors is

$$\mathbf{A} \cdot \mathbf{B} = \|\mathbf{A}\| \, \|\mathbf{B}\| \cos\theta$$

The value of cosine similarity ranges from -1 to 1, where:The value of cosine similarity ranges from -1 to 1, where:

- 1 suggests that the two vectors are the same (that is they are parallel and in the same direction).
- Indeed, when the cosine of the angle subtended by two vectors is equal to 0 it means that the vectors are orthogonal; that is, the vectors share no similarity.
- When the difference of vectors is equal to -1 this means that the vectors point in opposite directions.

Cosine similarity in our system is applied to compare face embeddings which are, in nutshell, numerical representations of faces. When the system checks if a new embedding (from a new image or detected frame) is unique, it compares this embedding to the existing ones using cosine similarity:When the system checks if a new embedding (from a new image or detected frame) is unique, it compares this embedding to the existing ones using cosine similarity:High similarity score (close to 1): Says that the new embedding is ‘’similar’’ to an existing one, suggesting that it is probably the same face. If the similarity is more than a preset value (it can for example be 70%) the system assumes that the embedding is not unique and thus does not insert its values into the database.
Low similarity score (closer to 0): Says that the new embedding is not equal to the old ones; it means that it might be a different representation of the face; it is then inserted into the database.
Performing cosine similarity enables the system to guarantee unique and meaningful embeddings hence proper identification of objects. This method is convenient because the desired distance between embeddings (e.g., face representations) is more important than the size of the shapes in which they are represented

4.10.4 GPU Acceleration

The adoption of GPU in deep learning frameworks has significantly transformed the speed and efficacy of complicated operations like the identification of objects and faces. We aimed to

demonstrate the big change that is seen when a GPU as opposed to a CPU is used . This was done by recording the time taken to process YOLO and FaceNet models.
From the two graphs Figure 22 and Figure 23 , it is quite clear that the use of a GPU significantly reduces both the overall processing time and the detection per frame.

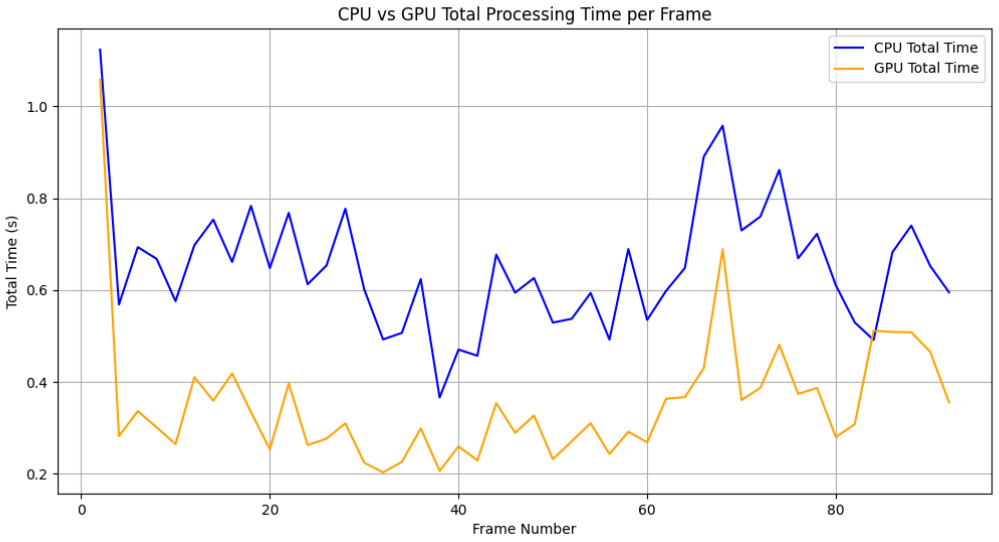

Figure [22] : Total time taken to process each frame; here the role of the GPU is dominant over the CPU. The processing time of the GPU is still low while the processing time of the CPU fluctuates in a higher range, which clearly shows the advantage given by the GPU.

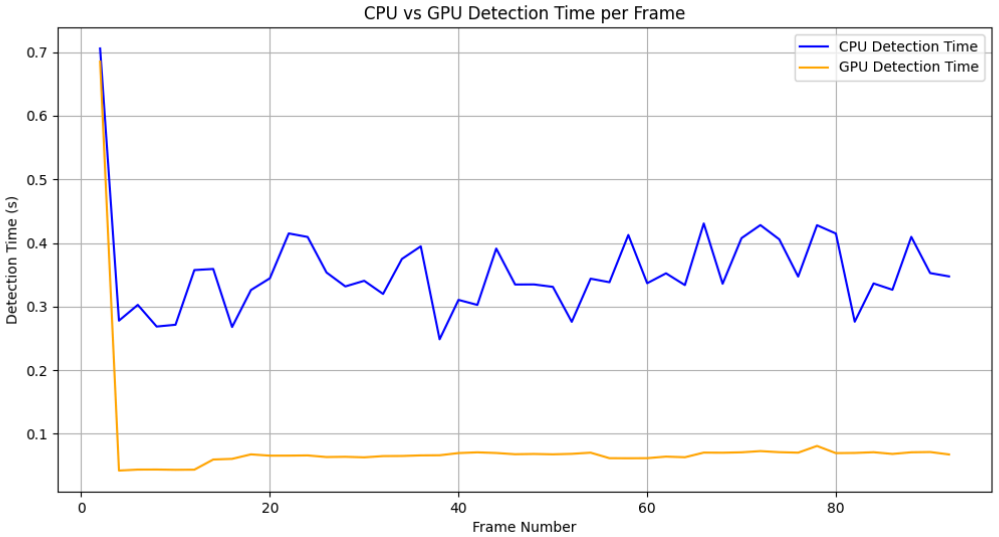

Figure [23] : Detection time per frame, the domination of the GPU is clearly seen. The GPU's detection time remains relatively short for all frames and the CPU's is much longer, and fluctuates more, with overall slower times.

This is especially true in terms of the relative reduction in total processing time and detection time when employing the GPU in deep learning. This is because various task computations that would have been very slow and time consuming when done on a CPU are done very efficiently with the help of the parallel processing ability of GPUs. Our test found Dramatic performance gains when comparing CPU and GPU processing. GPU processing completed tasks in 16.27 seconds compared to 29.92 seconds for CPU processing, representing a 45.62% reduction in processing time.

4.10.5 Multi-Threading Performance Improvement

Multi-threading enabled the simultaneous processing of multiple frames, significantly reducing the total processing time from 44.15 seconds to just 8.45 seconds—a reduction of over 80%. This improvement is largely due to the ability of the system to handle several frames concurrently rather than sequentially.

Figure [24] : Comparison of total processing time of a task performed without threading versus with threading demonstrating the significant performance gains achievable through multithreading.

Multi-threading allows multiple independent tasks to be processed simultaneously. For example, while one thread calculates similarity scores, another can handle the annotation of a different frame.

This parallelism is particularly beneficial when the number of detections increases, as it prevents the system from becoming overwhelmed by sequential processing bottlenecks.

4.10.6 Efficiency Gains with Increased Complexity

As the number of people detected in a frame increases, the computational load also increases. Without threading, the processing time for each frame increases significantly. However, with multi-threading, this increase is much less pronounced because the computational tasks are spread across multiple threads and CPU cores, allowing the system to manage the increased load more effectively.

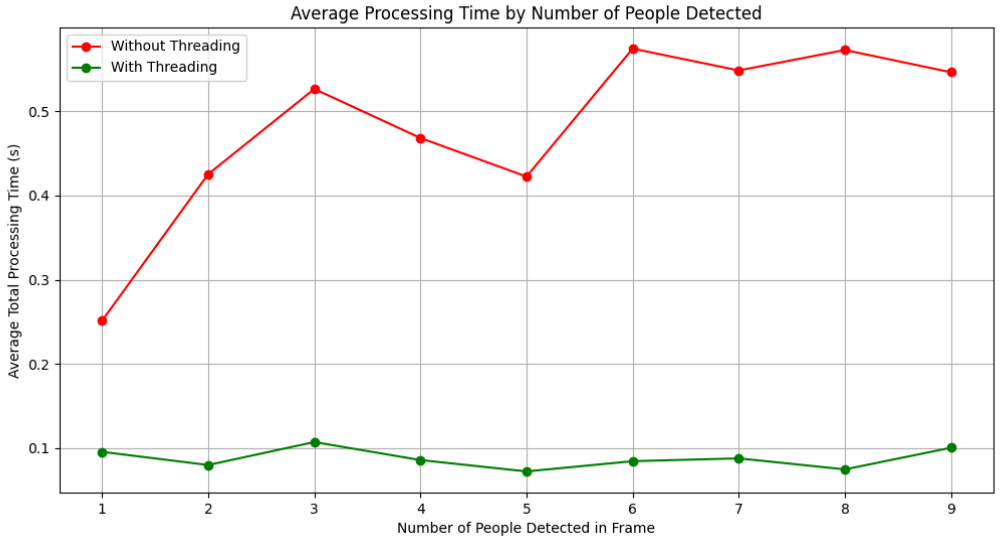

Figure [25] : Average processing time by number of people detected in a frame

Figure 25 Demonstrates the impact of threading on the average processing time as the number of people detected in each frame increases. The red line, representing processing without threading, shows a significant increase in processing time as the number of detections rises, peaking at over 0.5 seconds when six people are detected. In contrast, the green line, representing processing with threading, remains consistently low regardless of the number of detections, illustrating that threading greatly reduces processing time and maintains efficiency even as computational demands grow.Number of Detections In Frame

Without Threading (seconds)

With Threading (seconds)

Percentage Reduction (%)

Number of Detections In Frame	Without Threading (seconds)	With Threading (seconds)	Percentage Reduction (%)
1 Detection	0.251	0.096	61.94
3 Detections	0.526	0.107	79.63
6 Detections	0.574	0.085	85.28
9 Detections	0.546	0.100	81.61

Table [4] : Key Metrics (Average Processing Time by Number of Detections)

Table 4 effectively summarizes the substantial improvements in processing time achieved through threading. The percentage reduction highlights the efficiency gains, with reductions in processing time ranging from approximately 62% to 85% as the number of detections increases.

4.10.7 Frame Skipping For Time Efficiency in Person Identification

Maintaining a balance between speed and accuracy is important in real- time person identification systems, especially when dealing with high frame rate video. In order to maintain the responsiveness of the system by not overburdening its computational resources leading to high latency and low frame rates, it has been a common challenge. Our solution has been introducing a frame skipping strategy that ensures we retain time efficiency without compromising on the ability of the system to effectively track and identify people across frames.

The principle behind our approach is only running object detection and face recognition models of YOLO on even-numbered frames. This means that there will be half as many frames that need processing at real-time thus minimizing computation overload thus enhancing the overall processing speed. However, during initial analysis where odd numbered frames are skipped, they are compiles along with even numbered via post-processing without losing any crucial information.

During this process, we shall see whether every odd-numbered frame lies between two even-numbered ones where a person was detected/identified approximately within the same region

of the screen [50]. If this holds true then, we can say that probably he/she will appear also in an odd frame; hence most likely adding a detection box on such non-processed odd frame would fill these gaps without expensive computationally intensive model runs for each individual video-frame.

This method has some advantages. The first advantage is that it allows us to maintain the original video's frame rate which enables smooth playback with no dropouts. Next it saves time by reducing number of models' processed frames enabling faster responses in real-time applications. Finally, through inferring persons' presence in skipped frames selectively helps keep integrity of personal identification intact so that even under limited computational resources; it remains accurate and dependable [24].Such an approach will make our system more effective for high frame-rate videos characterized by both speed and accuracy requirements in real-life systems such as surveillance.

5 Work Artifacts

5.1 System Flow

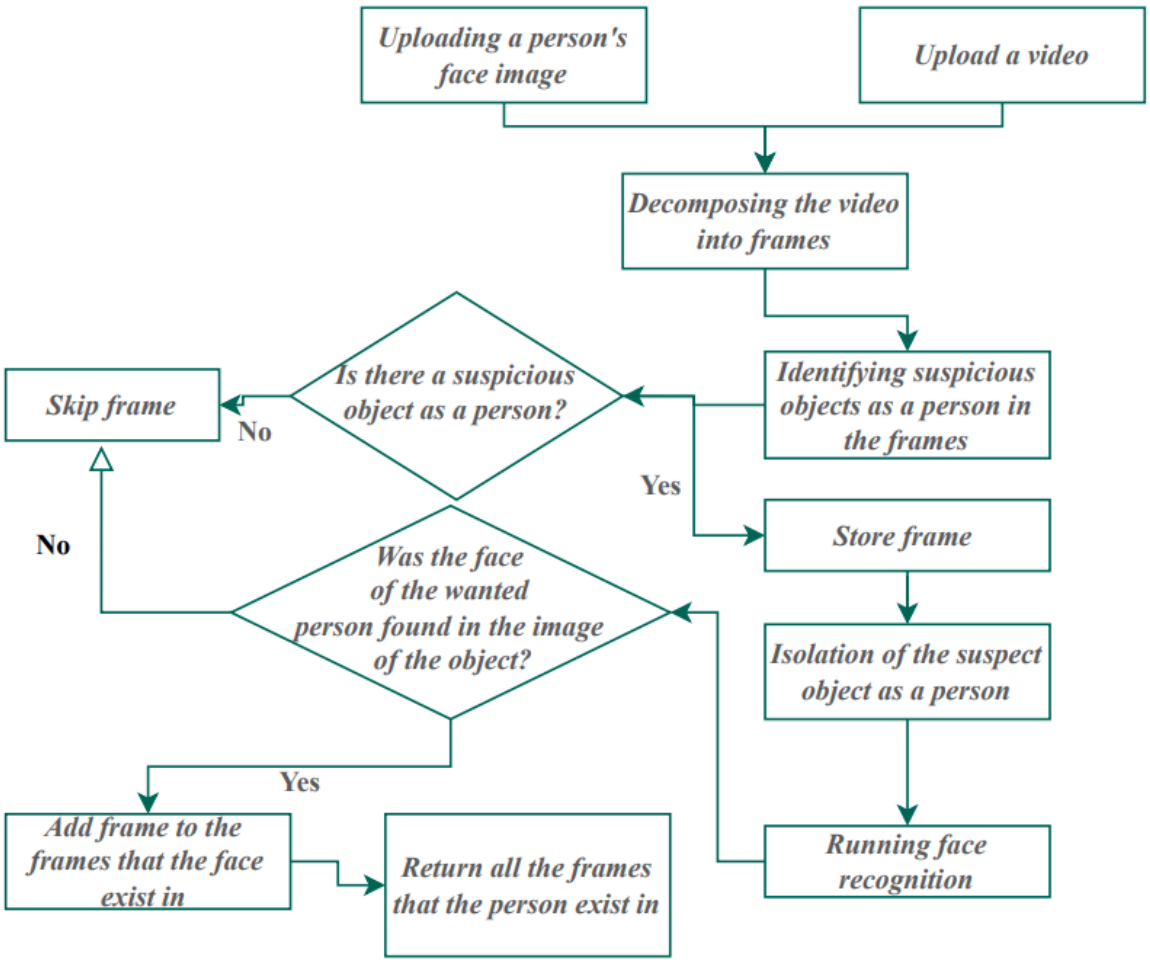

Figure 26 : system flow

Figure 26 illustrates a process for identifying a person's face within a video using our system Initially, the user uploads an image of the person's face and the target video. The video is decomposed into individual frames, which are analyzed to identify objects that might be people. Each suspicious object is further examined and isolated. Face recognition is then performed on these isolated objects to determine if they match the uploaded face image. Frames where no suspicious objects or matching faces are found are skipped. If a match is detected, the frame is added to a collection of frames containing the person's face. Ultimately, the system returns all frames where the person's face appears.

5.2 Architecture

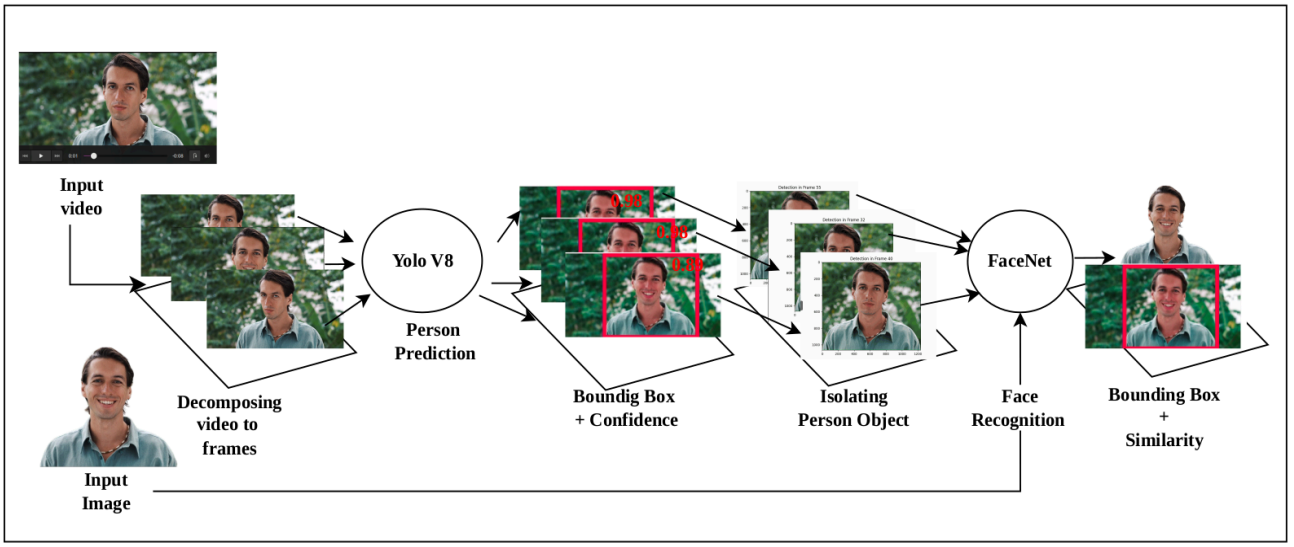

Figure 27 : Our architecture diagram

Figure 27 outlines our working proccess YOLOv8 for person detection and FaceNet for face recognition. The process starts with the input video to be segmented into the frames that will be processed independently. An input image of the person to be identified is also given. Every frame from the video is passed through the YOLOv8 model that performs the person detection .The model providing the coordinates of the bounding boxes for the detected persons along with the confidence scores. These bounding boxes assist in determining the positions of the detected persons in the different frames. Bounding boxes and confidence scores are then utilized to extract the person's objects, excluding low-quality detections. The isolated person objects are then ready for face recognition, and they are inputted into the FaceNet model. FaceNet recognizes faces by comparing the faces in the isolated person objects to the input image and returns the bounding boxes around the detected faces and similarity score to the input image. The final output generally displays bounding boxes around detected persons and draws their similarity scores as a way of labeling and locating the person correctly in the video frames.

4.2 FR & NFR Requirements

5.3.1 Functional Requirements

No.	Requirement
1	The system shall provide a user interface tool for users.
2	The system shall provide upload functionality for users.
2.1	The system shall allow users to upload videos.
2.2	The system shall allow users to upload a picture.
3	The system shall decompose the uploaded video into individual frames.
3.1	The system shall perform initial preprocessing on the frames.
4	The system shall utilize object detection mechanism.
5	The system shall isolate the detected person objects in the frames.
6	The system shall perform facial recognition
7	The system shall identify frames where the person of interest appears.
7.1	The system shall save the original frame and timestamp when a match is found.
8	The system shall display the frames and timestamps where the person was detected.
8.1	The system shall provide an option to download the results.
9	The system shall provide error messages for issues.

Table 5. functional requirements

5.3.2 Non-functional Requirements

No.	Requirement	Type
1	The system should ensure facial recognition is performed quickly to maintain a responsive user experience.	Performance
1.1	The system will avoid performing unnecessary calculations in frames where there is no expected match for the person we are looking for.	Performance

2	The system must handle large video files concurrently without performance degradation.	Scalability
3	The system should ensure high accuracy in person detection to minimize false positives and negatives.	Accuracy
3.1	The system should ensure high accuracy in facial recognition to minimize false positives and negatives.	Accuracy
4	The system should provide a user-friendly interface that is intuitive and easy to navigate.	Usability
4.1	The system should be accessible to users with varying levels of technical expertise.	Usability
4.2	The system will have three screens contain : data entry screen , identification process progress screen , results screen.	Usability
5	The system should be reliable and available, with minimal downtime.	Reliability
5.1	The system should implement robust error handling and recovery mechanisms.	Reliability
6	The system should be easy to maintain and update, with clear documentation and a modular code structure.	Maintainability
7	The system should support a wide range of video formats and resolutions.	Compatibility
7.1	The system should ensure compatibility with various web browsers and devices.	Compatibility
8	The system will enable facial recognition by advanced and innovative algorithms.	Technology
9	The system will enable the identification of a human figure by YOLO.	Technology
10	The system will optimize the inputs that enter it to match the models.	Optimization

Table 6. non-functional requirements

5.4 Working Progress

Figure 28 : work flow during semester

5.5 System Use Case

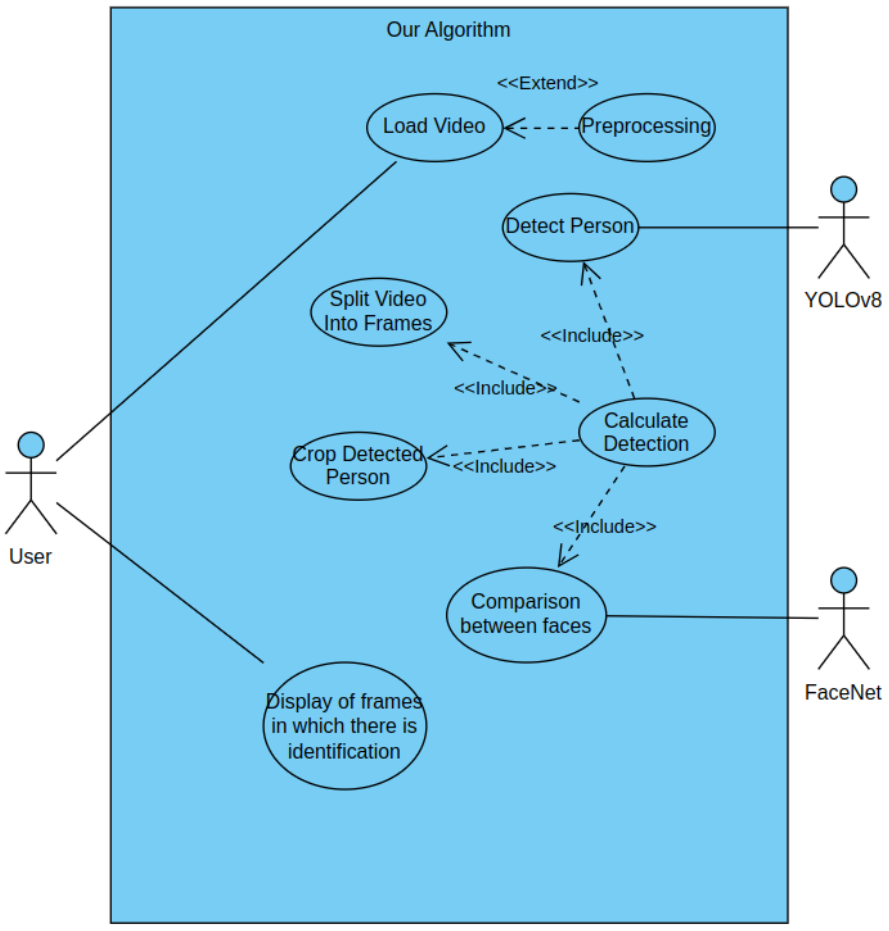

Figure 29 : system use case

5.6 System Activity

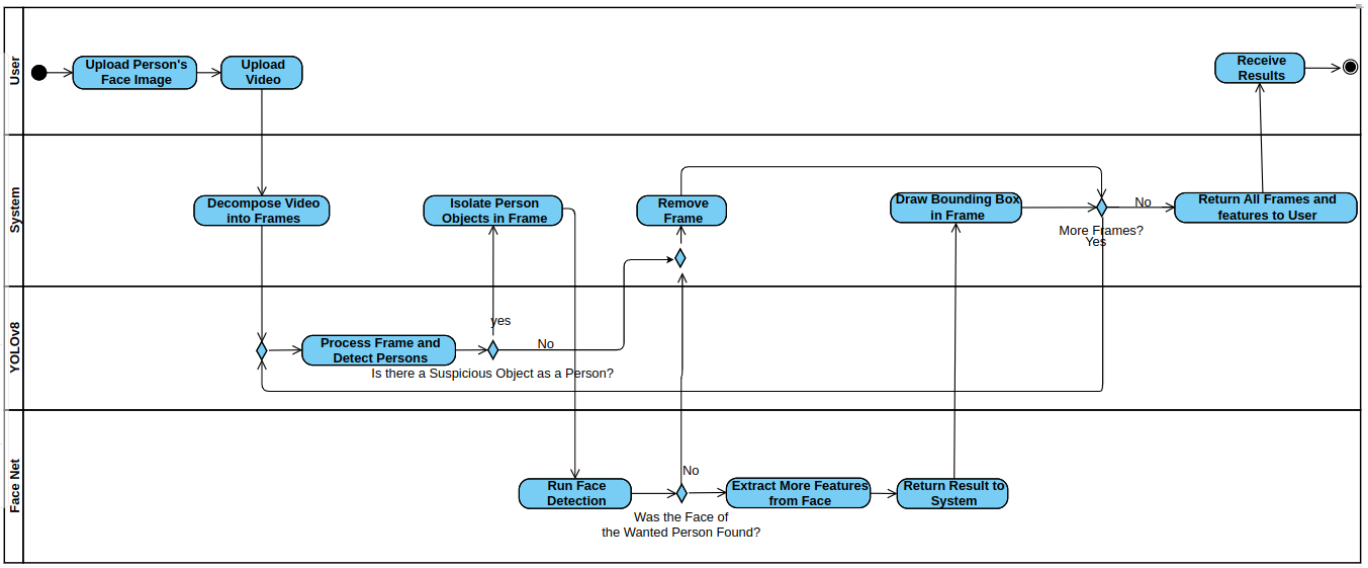

Figure 30 : system activity diagram

The activity diagram outlines the process of a system designed for face recognition in videos. Initially, the user uploads an image of the person's face and the video in which they want to locate this face. The system decomposes the video into individual frames to facilitate frame-by-frame analysis. Each frame is processed using the YOLOv8 algorithm to detect any persons within the frame. If a suspicious object resembling a person is detected, it is further analyzed using the FaceNet algorithm to verify if it matches the uploaded face image. If a match is found, additional features are extracted from the face for accurate identification and analysis, and the frame is added to a collection of frames where the person appears. This process is repeated for each frame until the entire video is processed. After analyzing all frames, the system compiles and returns all frames containing the identified face, along with the extracted features, to the user. The system may also draw bounding boxes around the identified faces for better visualization. This ensures an efficient and comprehensive method to locate and highlight a specific person's face within a video.

5.7 Package Diagram

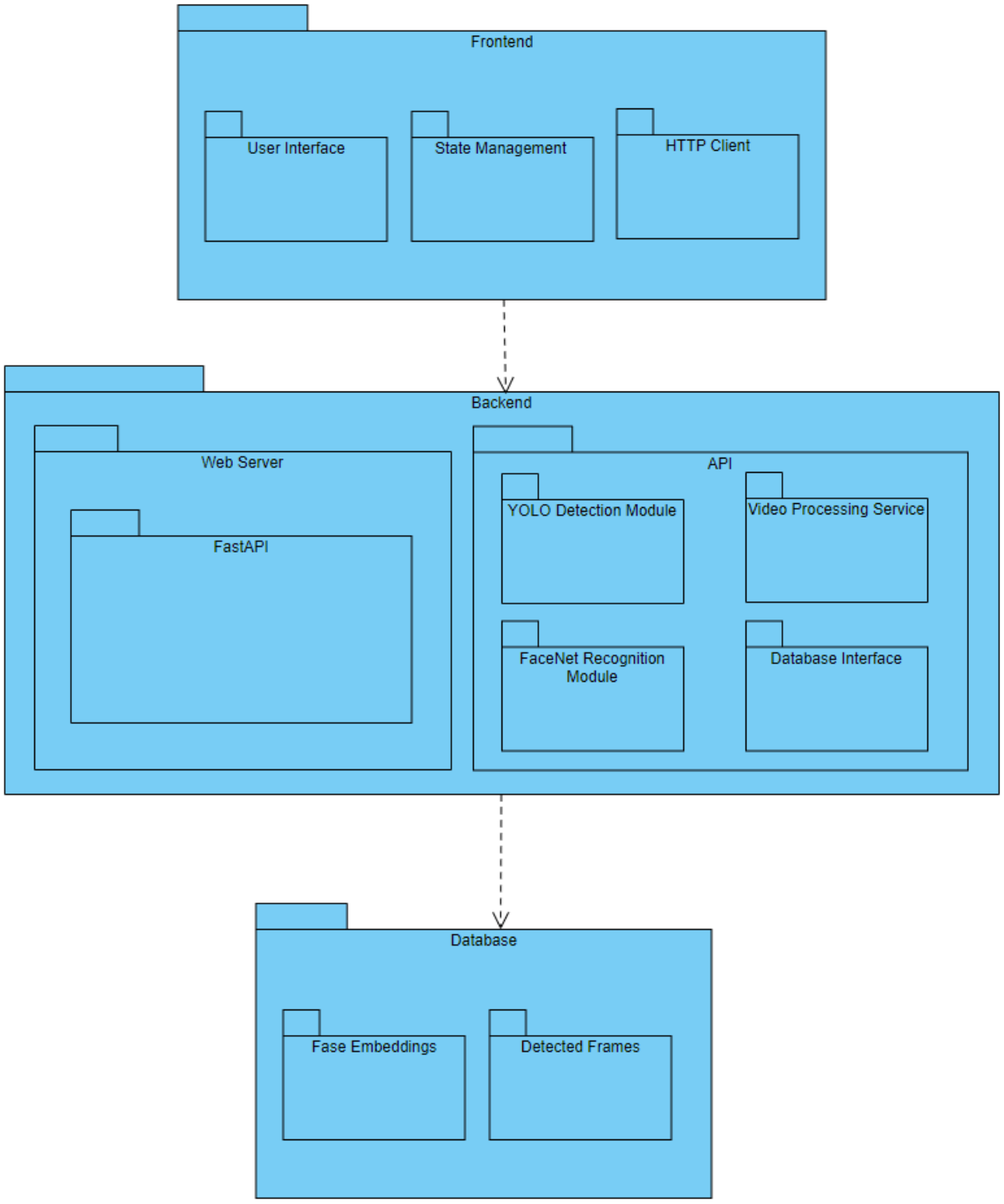

Figure 31 : Package diagram

5.8 User Interface

5.8.1 Loading Data Screen

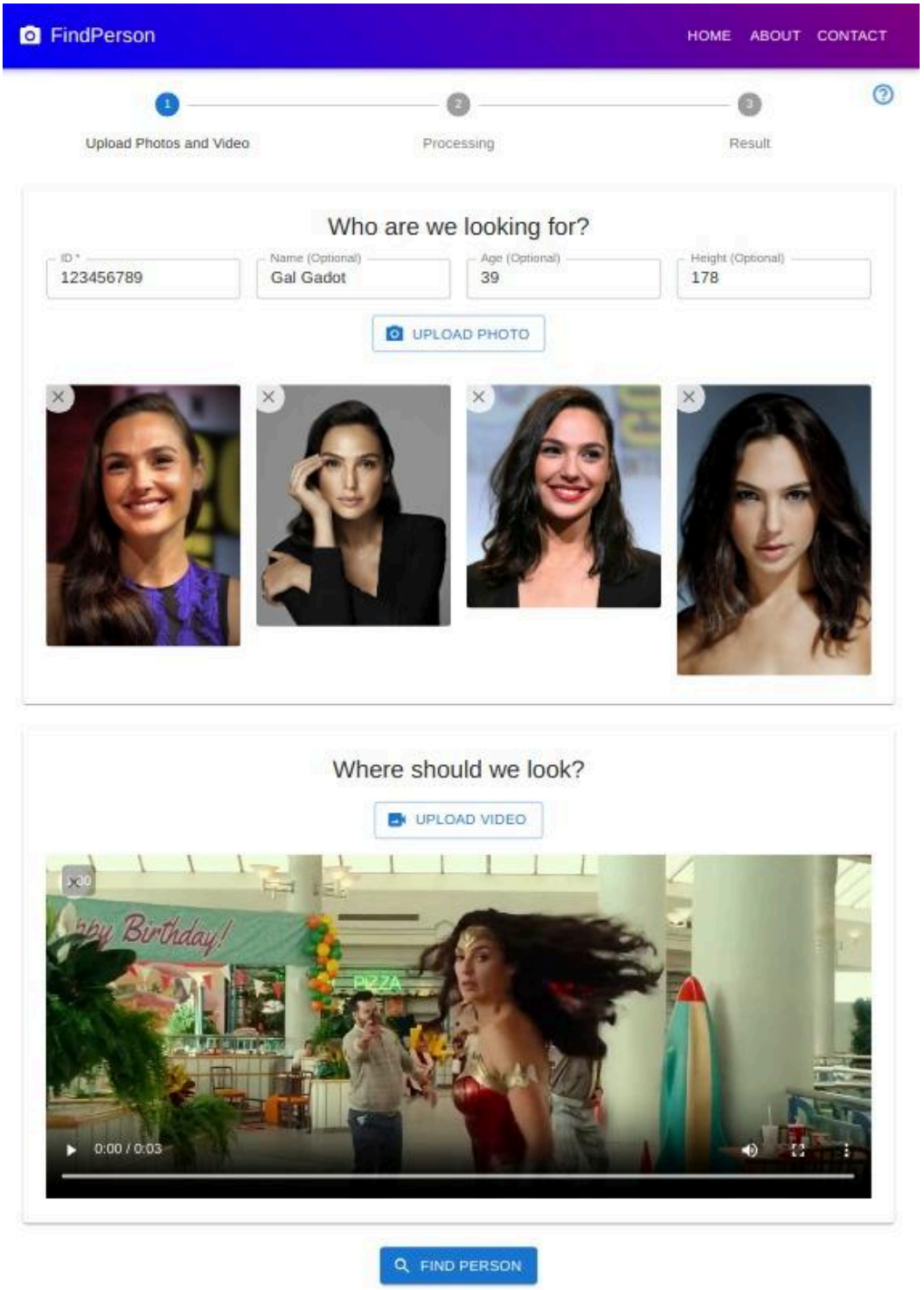

Figure 32 : Screen 1 ,Loading Data Screen

5.8.2 Analyzing Screen

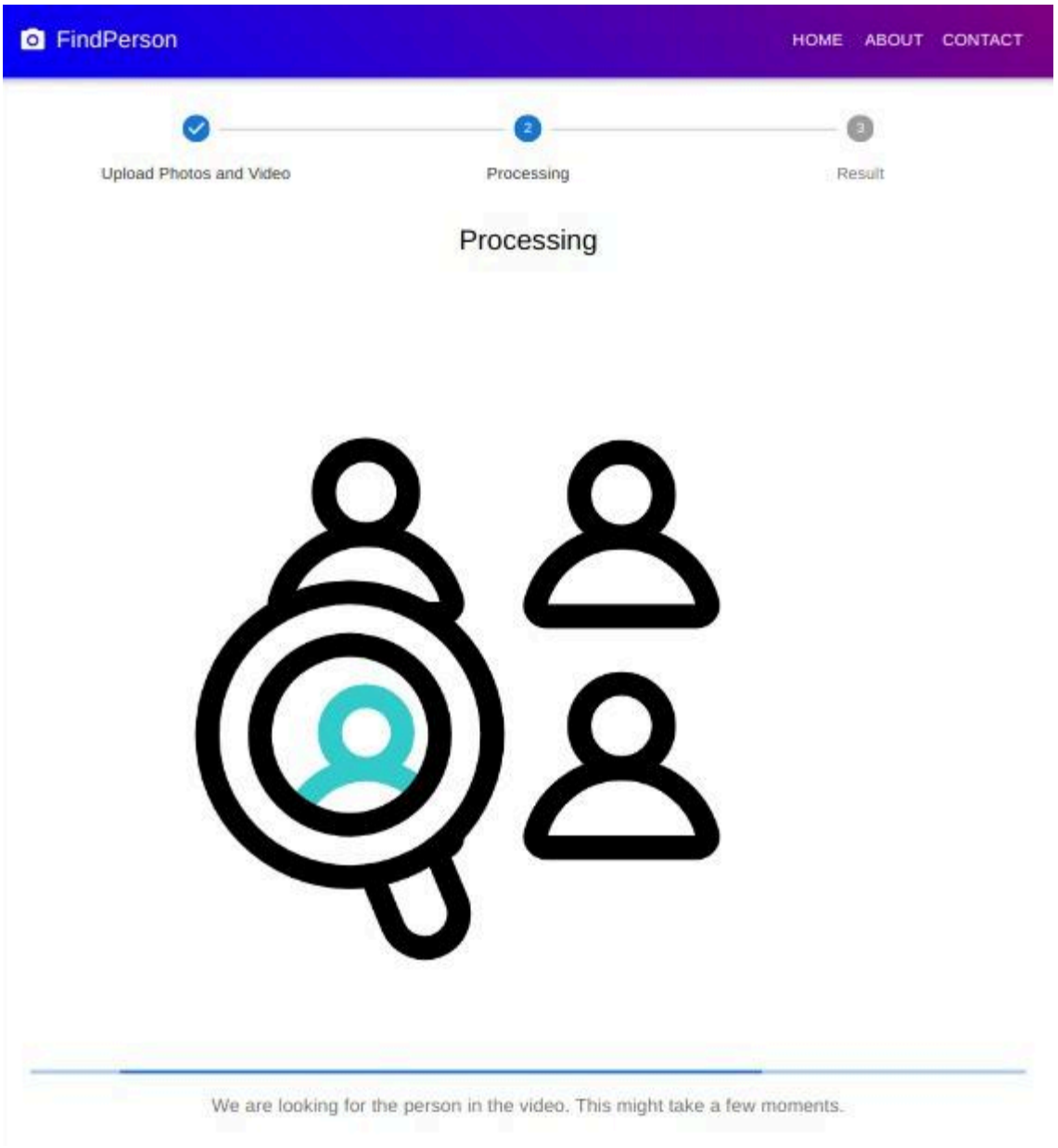

Figure 33 : Screen 2 , user see that the system is analyzing the results

5.8.3 Result Screen

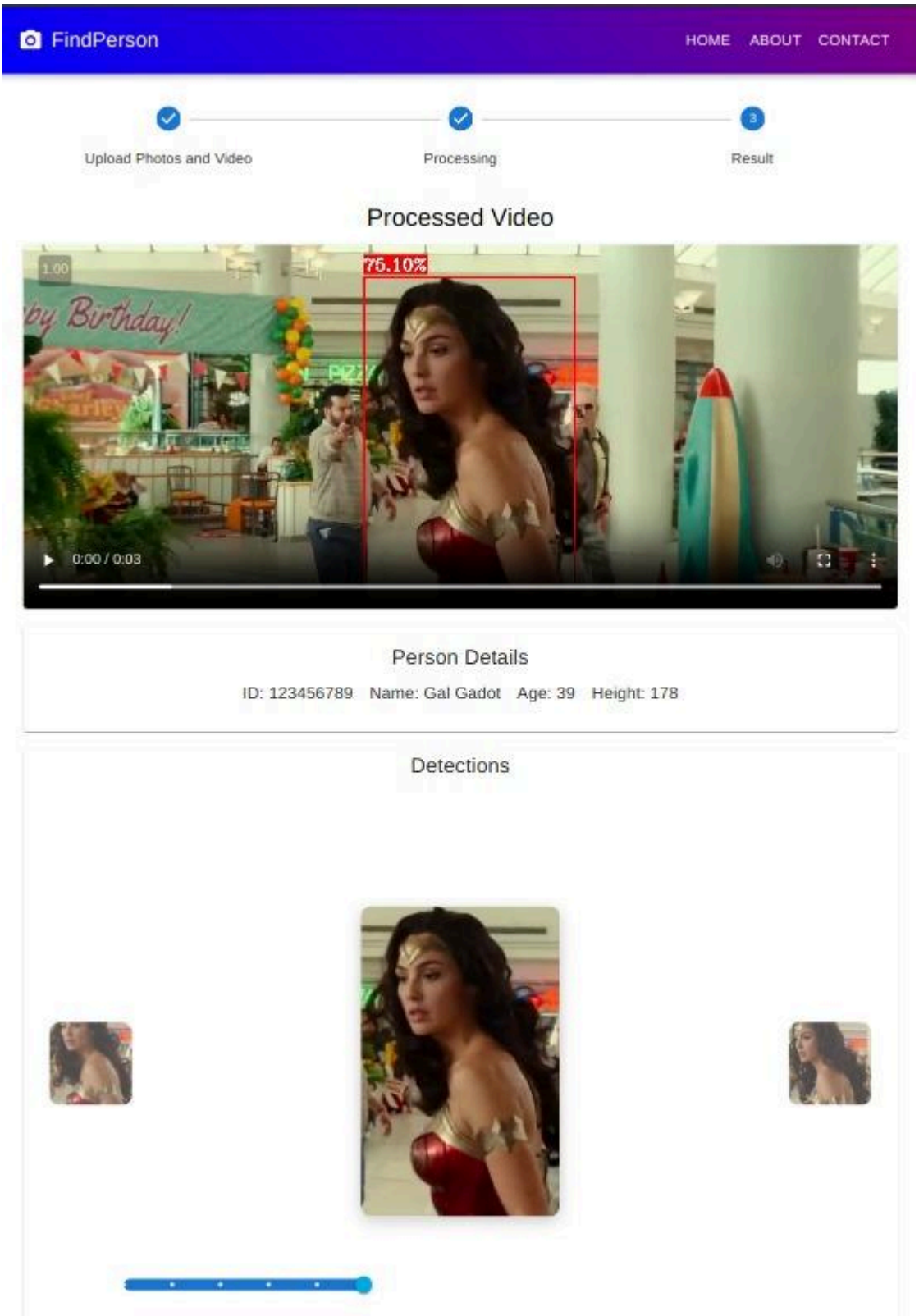

Figure 34 : Screen 3 ,result screen , user can see the person he searched for and our algorithm detection in the original supplied video.

5.9 Server API - Swagger

FastAPI is a modern, high-performance, Python web framework for building APIs. It leverages Python's type hints to automatically provide request and response validation, and it natively supports asynchronous programming, which allows it to handle arbitrarily high concurrency without performance degradation. Probably one of the most awesome features of FastAPI is its capability for automatic generation of interactive API documentation using tools like Swagger and ReDoc that make it easier to explore developers' API endpoints. Swagger comes included with FastAPI and provides a very nice, interactive UI for API documentation that allows developers to interact with a FastAPI application directly from the browser. That means being able to see all the available routes, their descriptions, test them with real requests and responses. FastAPI works seamlessly with Swagger for ramping up development, debugging, and API documentation.

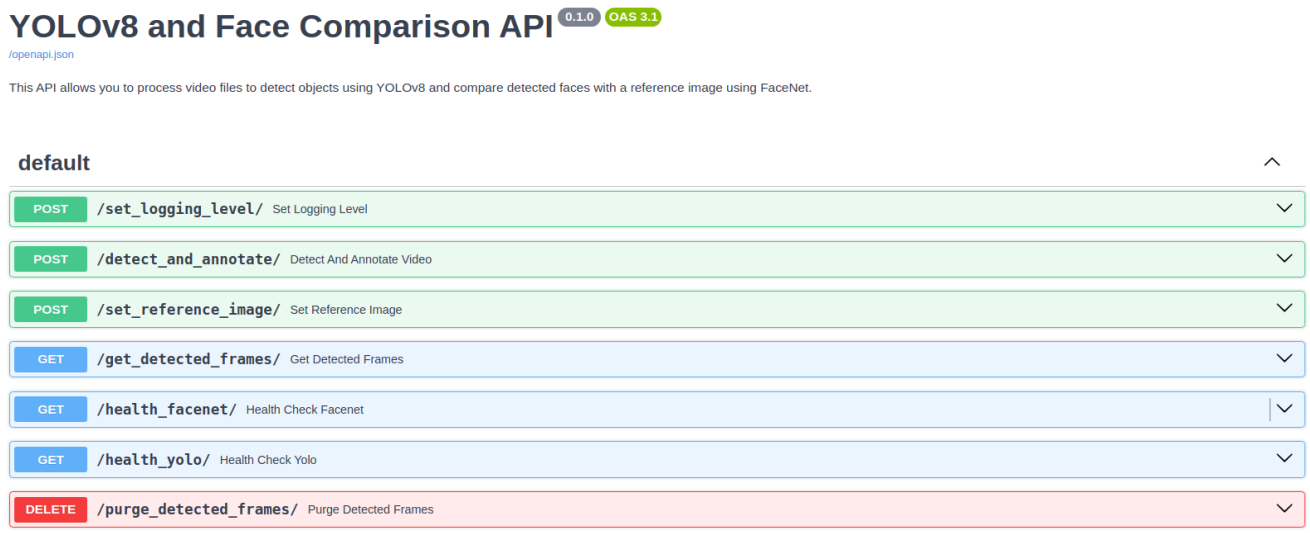

Figure 35 : Swagger of our server using fastapi

1. **/set_logging_level** : Allows dynamically setting the logging level for both YOLO and FaceNet systems. It responds with a success message if the logging level is set, or an error message if there is an issue.

2. **/detect_and_annotate/**: Processes an uploaded video file, detects objects using YOLOv8, and annotates the video based on the provided similarity threshold. The processed video is returned as a downloadable annotated file.

3. **/set_reference_image/:** This endpoint sets reference images for face comparison by uploading images, processing them to generate embeddings, and storing these embeddings along with user details in the database. It combines embeddings from both uploaded images and detected frames, returning a confirmation upon success.

4. **/get_detected_frames/:** Retrieves the detected frames from the last processed video for a given uuid and running_id. If no frames are found, it returns an error message.

5. **/health_facenet/ :** This endpoint performs a health check for the FaceNet system, ensuring that both FaceNet and MongoDB are functioning correctly. It returns a "healthy" status if successful, or "unhealthy" if there is an issue with MongoDB.

6. **/health_yolo/ :**This health check endpoint ensures that the YOLOv8 system and MongoDB are running properly. It responds with a "healthy" status if both are functioning, or an "unhealthy" status otherwise.

7. **/purge_detected_frames/:** This endpoint deletes all detected frame entries in the MongoDB collection. A success message is returned upon completion, or an error message if something goes wrong.

6 Testing Plan

6.1 Scope

Comprehensive testing plan will encompass both functional and non-functional testing of the Video-Based Person Identification System. This plan ensures the system meets all specified requirements, performs as expected, and verifies the accuracy of person detection and facial recognition, along with user acceptance.

6.2 Objectives

- Ensure all functional and nonfunctional requirements are met.
- Verify system performance and responsiveness.
- Confirm accuracy in person detection and facial recognition.
- Validate usability and user experience.
- Checking that the results are clear and understandable for system users .

6.3 Test Cases

No.	Test Name	Description	Procedure	Expected Result	Actual Result

1	Video_With_Other_Person	Test with video containing other persons but not the target person	1. Upload video without target person 2. Upload target picture 3. Start detection and recognition	System indicates no match found	Passed
2	Video_With_No_Person	Test with video containing no persons	1. Upload video without any persons 2. Upload target picture 3. Start detection and recognition	System indicates no persons detected	Passed
3	Video_With_Many_Persons	Test with video containing many persons	1. Upload video with many persons 2. Upload target picture 3. Start detection and recognition	System accurately detects and identifies target person	Passed
4	Low_Quality_Video	Test with low-quality video	1. Upload low-quality video 2. Upload target picture 3. Start detection and recognition	System performs detection and recognition with decreased accuracy	Passed
5	Low_Quality_Image	Test with low-quality image of the target person	1. Upload high-quality video 2. Upload low-quality picture 3. Start detection and recognition	System performs detection and recognition with decreased accuracy	Passed
6	Short_Length_Video	Test with short video	1. Upload short video 2. Upload target picture 3. Start detection and recognition	System performs detection and recognition accurately	Passed
7	Long_Length_Video	Test with long video	1. Upload long video 2. Upload target picture 3. Start detection and recognition	System performs detection and recognition accurately and efficiently	Passed

8	Video_With_Low_Light	Test with video in low light conditions	1. Upload low light video 2. Upload target picture 3. Start detection and recognition	System performs detection and recognition with decreased accuracy	Passed
9	Different_Angles	Test with video showing person at different angles	1. Upload video with person at different angles 2. Upload target picture 3. Start detection and recognition	System performs detection and recognition accurately	Passed
10	Overall_Functionality	Validate overall system functionality	1. End users perform typical tasks 2. Report any issues	System meets user expectations and needs	Wait for group testing
11	System_Usability	Validate system usability	1. End users navigate the system 2. Provide feedback on interface and experience	System is user-friendly and intuitive	Wait for group testing
12	System_Performance	Validate system performance	1. End users upload various video sizes 2. Observe system response	System performs well under typical user conditions	Wait for group testing
13	YOLO_Person_Detection	Test accuracy of person detection	1. Compare detected persons with ground truth data	High accuracy in person detection	Passed
14	Facial_Recognition	Test accuracy of facial recognition	1. Initialize test on large dataset for face recognition 2. Compare recognized faces with ground truth data 3.Calculate recall measure	High accuracy and recall in measure in facial recognition	Passed
15	YOLO_Person_Detection	Test recall measure of person detection	1. Initialize test on large dataset for persons photo . 2. Compare detected persons with ground truth data	High accuracy and recall measure in person detection	Passed

			3. Calculate recall measure		
17	Video_Image_Upload	Test user interface for video and image upload	1. Open the application 2. Navigate to upload screen 3. Upload a video 4. Upload a picture	Video and picture upload successfully	Passed
18	Decompose_Frames	Test video decomposition into frames	1. Upload a video 2. Start processing	Video is decomposed into individual frames	Passed
19	Detected_Person_Isolation	Test isolation of detected persons	1. Upload a video 2. Start person isolation	Persons isolated in frames	Passed
20	Isolated_Person_Facial_Recognition	Test facial recognition on isolated persons	1. Upload a picture 2. Start facial recognition	Faces matched with uploaded picture	Passed
21	Display_Results	Test results display	1. Complete match identification 2. View results	Frames and timestamps displayed	Passed
22	Unsupported_Formats	Test error handling for unsupported formats	1. Upload unsupported video format 2. Upload unsupported picture format	Meaningful error messages displayed	Passed
23	System_Performance_Under_Load	Test system performance under load	1. Upload large video file 2. Perform all processing steps	System performs efficiently	Passed
24	Full_Process_Validation	Validate full process from video and person upload to results	1. Upload a video 2. Upload a picture 3. Start detection and recognition 4. View results	System returns all frames where the person exists in the video	Passed

Table 6. test cases of our application

7 Expected Accomplishments

- **Better accuracy in recognizing individuals:** It can be used to drastically improve the precision of person identification from video content but especially in situations that are dynamic and complex.
- **Detection capability in real-time:** With the YOLO algorithm, this system will enable fast detection of people which is important for security or emergency response applications.
- **Strong face recognition:** The system will ensure that different faces are accurately recognized under various angles by utilizing FaceNet thus making it possible for correct identification to be done even under different circumstances.
- **Powerful analysis tool:** When advanced detection and recognition technologies are integrated, it becomes a useful instrument for comprehensive video analysis thus benefiting media research as well as security among others where investigations may involve videos.
- **User friendly interface:** An intuitive and accessible interface should be designed so that specialized teams can use this software effectively without undergoing too much training.

7.1 Problems

- **Managing occlusions and partial visibility:** One major problem in person identification is dealing with cases where individuals might not be fully visible or could be hidden behind other objects.

- **Achieving real time performance:** A lot of computational power is needed for systems like these but they still need high speeds hence optimization must always balance between these two aspects; efficiency vs accuracy.

- **Cross domain generalization:** Generalizing models trained on one dataset to work well with another data-set or real world scenarios has proven difficult yet it’s crucial if we want reliable performance across all domains.

7.2 Criteria for Success

- **Accuracy of Identification:** The precision level should not fall below 90% when it comes to this software verifying people through videos taken within any given environment as supported by routine tests and validation.
- **Operational impact:** 30% reduction in response time and decision-making efficiency during security and emergency situations should be realized.
- **User feedback and satisfaction:** Structured feedback from specialized teams should be able to score at least 80% satisfaction rate thus indicating its usefulness among them.

- **System reliability & performance:** The system must work with a minimum uptime of 99%, maintaining good performance stability even under high load scenarios or when subjected to different conditions.
- **Adoption & Engagement:** The system is considered effective if it can attract adoption by up-to-half the target user groups within six months following its deployment.

8 User Guide

8.1 Getting Started

To begin using the person identification system, users must follow a series of initial steps that will set the foundation for effective operation. These steps include uploading photos and videos, processing videos for identification, and viewing the results generated by the system.

8.1.1 Step 1: Uploading Photos and Videos

The first action you will take upon the system is to upload the necessary media files, such as photos and videos of individuals you wish to identify. To do this, navigate to the Home Page. Here, you will find options to select files from your device.

1. Fill the ID field - make sure it is unique to the person you look for (ID number)
2. The rest of the fields are optional (Name, Age , Height)
3. Click on the "Upload Photos" button.
4. Select one or more photos from your computer.
5. Click on the "Upload Video" button.
6. Select one video from your computer.
7. Confirm and start the video process by clicking "Find Person."

If any error occurs or you forget to do one of the steps the system will let you know.

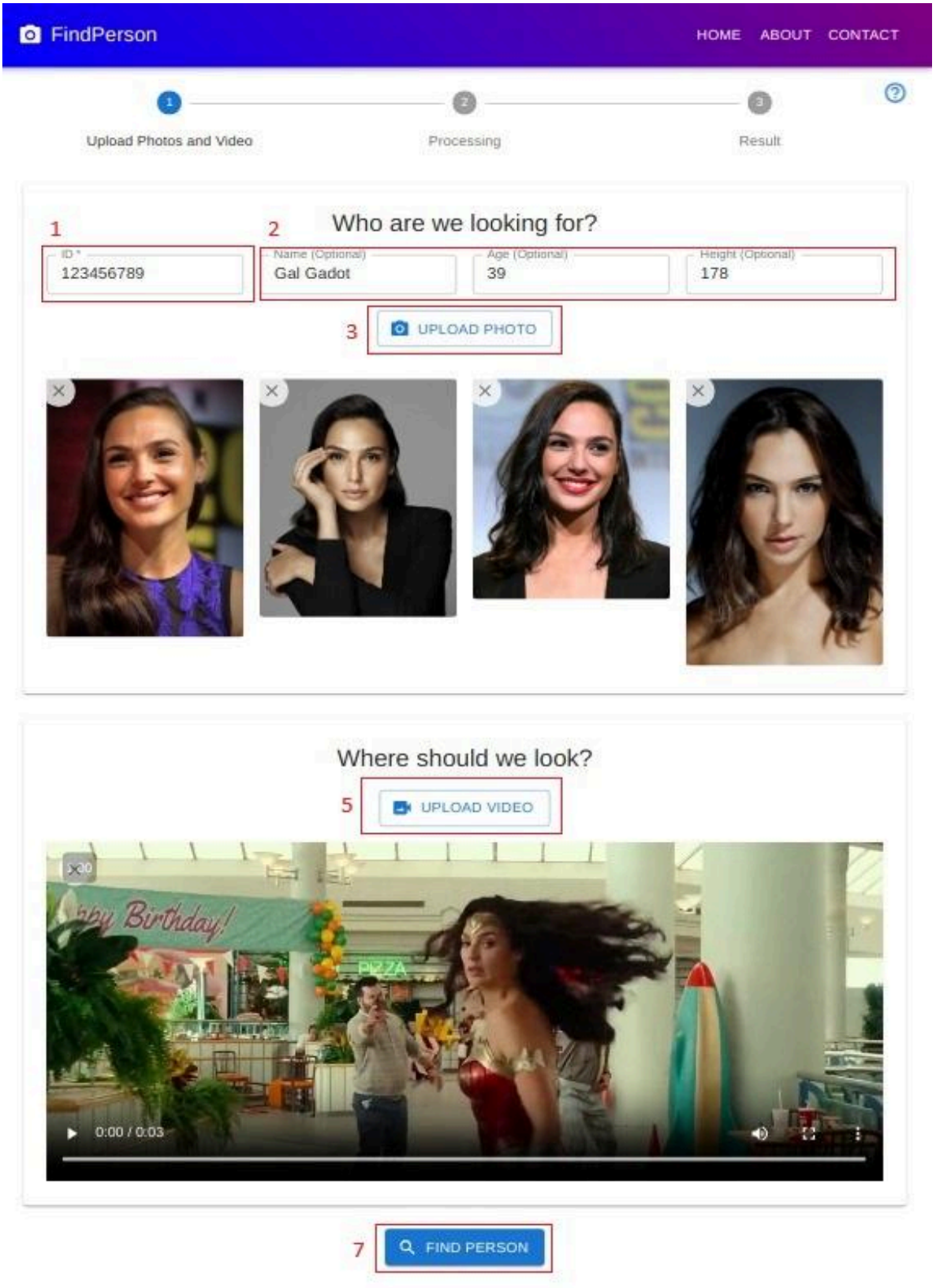

Figure 36 : Uploading Photos and Videos screen detailed

8.1.2 Step 2: Processing Videos

After uploading the reference photos and video, the system will automatically begin processing the video to identify the individual based on the provided images.

The system will start analyzing the video using YOLO for object detection and FaceNet for face recognition.

The analysis may take a few minutes, depending on the length and complexity of the video. During this time, the system scans through the video, detecting and recognizing the individual in each frame.

The system utilizes the YOLO algorithm to detect all objects in the video frames, while FaceNet is used to compare and recognize faces based on the reference photos.

The system ensures accuracy by analyzing each frame and identifying the individual wherever they appear in the video.

8.1.3 Step 3: Viewing Results

Once the system completes the video processing, it automatically navigates you to the results page, where you can thoroughly review the identified individual within the video. The video you uploaded will be displayed prominently, with bounding boxes dynamically drawn around the individual in each frame where they were detected. These bounding boxes provide a clear and precise visualization of the identification process, making it easy to see exactly where the person appears throughout the

video.

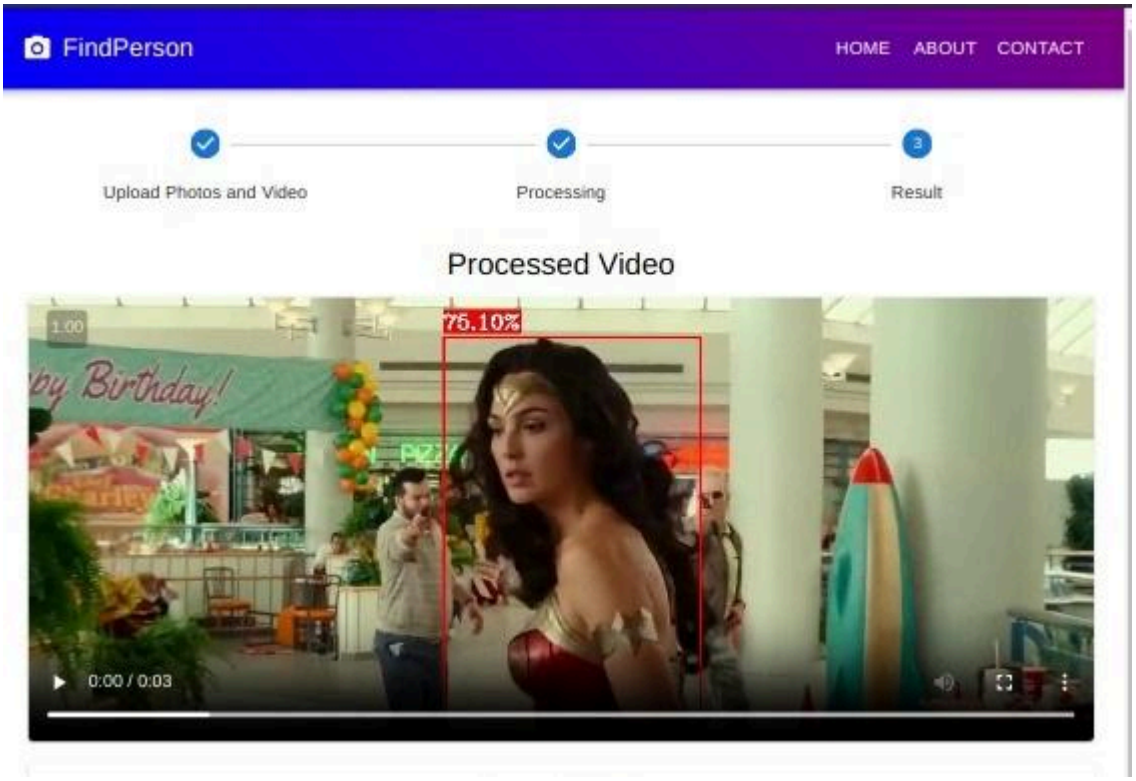

Figure 37 : Viewing Results

Below the video, you'll find the details of the person you provided during the upload, such as their name, age, and height. This information is presented alongside the video to help you verify the accuracy of the identification, ensuring that the correct individual has been recognized.

Figure 38 : Insert details of the person details

A carousel beneath the video offers an interactive way to view the specific frames where the person was detected. The central image in this carousel represents the most recent detection, flanked by smaller images showing the detections immediately before and after. You can scroll through these frames using the carousel, allowing you to quickly review all the instances where the person was identified. If you want to examine a particular detection more closely, simply click on the corresponding frame in the carousel, and the video will jump directly to that exact moment in the timeline, enabling you to analyze it in detail. This seamless integration of video playback, person

details, and detected frames offers a comprehensive and user-friendly way to review the results of the processing.

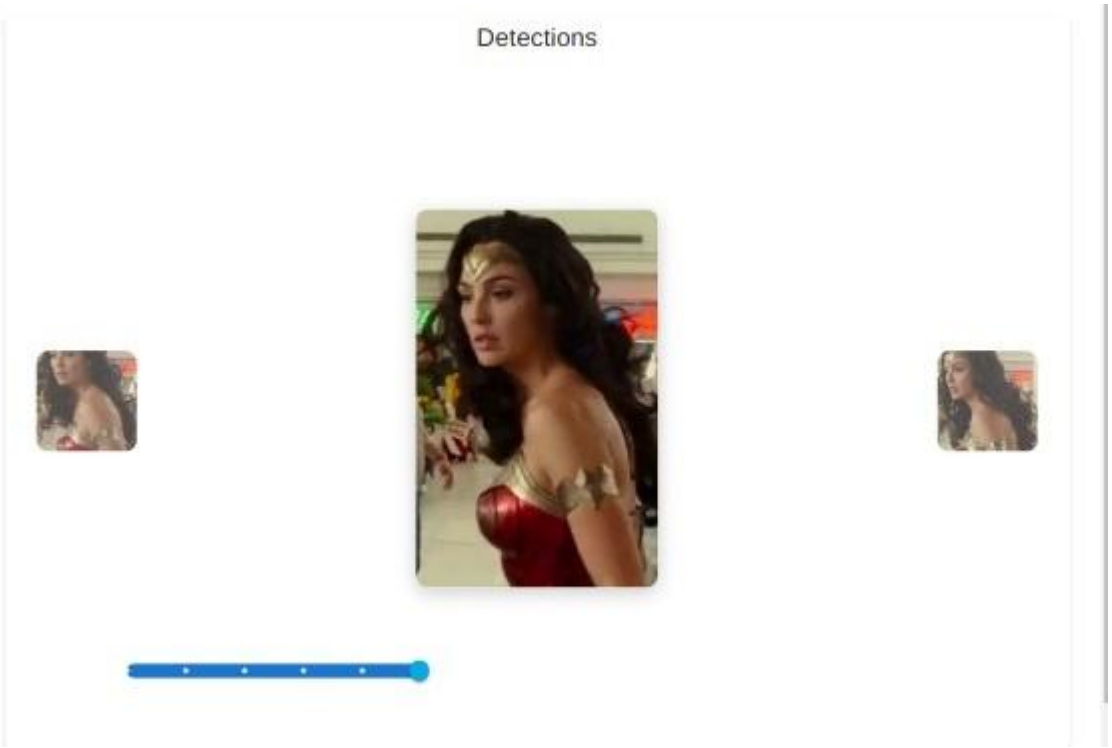

Figure 39 : Displaying all detected frames

8.2 Features Overview

The person identification system integrates several cutting-edge features designed to enhance its functionality and usability. Among these, YOLO-based object detection, FaceNet-based recognition, and optimized frame processing stand out as pivotal elements that contribute to the system's effectiveness.

8.2.1 YOLO-Based Object Detection

At the core of the system's capability is the YOLO (You Only Look Once) object detection algorithm. This feature enables real-time identification of individuals within a video stream. YOLO's unique architecture allows it to process images at remarkable speed while maintaining high accuracy. By dividing the image into a grid and predicting bounding boxes and class probabilities simultaneously, this method minimizes the latency typically associated with traditional detection systems. The result is an efficient and robust tool that can swiftly identify multiple objects in dynamic environments, making it invaluable for security and surveillance applications.

8.2.2 FaceNet-Based Recognition

Another critical feature is the FaceNet-based recognition system, which utilizes deep learning techniques to recognize faces with high precision. FaceNet generates a compact representation of each individual's facial features, enabling the system to match and identify them against a database of known faces. This feature is particularly important in scenarios requiring high levels of security and access control, as it allows for the swift verification of identities. The accuracy of FaceNet significantly reduces false positives and enhances the reliability of the identification process, fostering trust in the system's capabilities.

8.2.3 Optimized Frame Processing

To ensure seamless operation, the system employs optimized frame processing techniques. This feature enhances the efficiency of video analysis by reducing the computational load without compromising the quality of results. By intelligently selecting frames for processing based on motion detection and other criteria, the system minimizes unnecessary data handling. This results in quicker identification times and lower resource consumption, which is essential for maintaining high performance in environments with multiple simultaneous video feeds.

These features collectively ensure that the person identification system operates at peak efficiency, providing users with rapid, accurate, and reliable identification capabilities crucial for enhancing security measures and operational efficiency.

8.3 Troubleshooting

As with any web-based system, users of the person identification platform may encounter various issues when accessing the service through their browser. This troubleshooting section aims to help you resolve common problems to ensure a smooth experience when using the system.

8.3.1 Common Issues and Solutions

8.3.1.1 Connectivity Issues

If the system fails to connect to the remote server or experiences interruptions, start by checking your internet connection. Ensure that your network is stable by testing other websites. If you encounter persistent issues, try restarting your modem or router. If the problem remains unresolved, temporarily disable any firewall or antivirus software that might be blocking your browser's connection to the server.

8.3.1.2 Browser Compatibility

Sometimes, the application might not function as expected due to browser compatibility issues. Ensure that you are using an up-to-date version of a supported browser (e.g., Chrome, Firefox,

Safari). If you encounter issues, try clearing your browser cache or switching to a different browser to see if the problem persists.

8.3.1.3 Video Upload and Processing Delays

If your videos are taking too long to process or do not process at all, check the size and format of the video file. Large files may take longer to upload and process, especially over slower internet connections. Additionally, ensure that your browser settings allow for large file uploads. If processing fails, try re-uploading the video, converting it to a more compatible format, or checking if the server is experiencing high load or downtime.

8.3.1.4 Identification Accuracy

If the system does not accurately recognize individuals, the quality of the uploaded images and videos might be the cause. Ensure that the photos and videos are clear and taken in well-lit conditions. Poor-quality media can significantly affect recognition performance. Regularly update your reference photos with recent images to enhance identification accuracy.

8.3.2 Frequently Asked Questions (FAQs)

Q: What should I do if the website isn't loading or is very slow?

A: First, check your internet connection and try refreshing the page. If the issue persists, try accessing the site using a different browser or clearing your browser's cache. If the problem continues, there may be an issue with the server, so try again later.

Q: Why is the person not being recognized accurately in the video?

A: Make sure to upload high-quality images and videos, where the person's face is clearly visible in good lighting and without any obstructions. For the best identification results, ensure that the reference photos are recent, feature only the individual you want to identify, and do not include any other people in the frame.

By addressing these common issues and following the provided solutions, users can effectively troubleshoot problems they may encounter while using the person identification system. For further assistance, please refer to the support section on the website or contact technical support.

9 Maintenance Guide

9.1 System Architecture

The architecture of our person identification system is designed to facilitate seamless interactions between the frontend and backend components while ensuring scalability, reliability, and security. The system is structured around a modular architecture consisting of three primary layers: the frontend, the backend, and the database.

9.1.1 Frontend Interaction

The frontend is built using modern web technologies such as React, HTML, CSS, and JavaScript, and it is accessible via a web browser. It provides a user-friendly interface that allows users to interact with the system effortlessly. The frontend communicates with the backend via RESTful APIs, handling requests and responses efficiently. This interaction is crucial for tasks such as uploading media files, initiating video processing, and retrieving identification results. The use of AJAX (Asynchronous JavaScript and XML) enables asynchronous requests, ensuring minimal latency and allowing users to perform multiple operations simultaneously without page reloads.

9.1.2 Backend Functionality

The backend is the powerhouse of the system, responsible for processing requests from the frontend, executing the necessary operations, and managing data interactions with the database. Developed using a robust framework like FastAPI, the backend handles tasks such as video processing, user management, and data retrieval. It leverages various libraries and tools, including YOLO for object detection and FaceNet for facial recognition, to enhance the identification process.

9.1.3 Database Management

The system uses a local MongoDB Compass database for managing data securely and efficiently. This database stores user profiles, identification logs, and metadata, ensuring data integrity and supporting efficient queries. Regular maintenance tasks, such as database optimization, indexing, and backups, are essential to ensure fast data retrieval and improve overall system performance.

9.1.4 Dependencies

The system relies on several key dependencies to function effectively, including:

- FastAPI: For building the backend API.

- React: For developing the frontend.
- TensorFlow and Keras: For integrating machine learning models for object detection and facial recognition.
- MongoDB Compass: For managing the local database securely.
- Gunicorn and Uvicorn: For serving the FastAPI application in production.

Understanding the interactions between these components and their dependencies is vital for effective system maintenance. Ensuring that each layer operates cohesively allows administrators to proactively address potential issues and enhance the overall performance of the person identification system.

9.2 Regular Maintenance Tasks

To ensure the person identification system operates efficiently and securely, administrators must perform a series of routine maintenance tasks. These tasks are crucial for preventing potential issues, maintaining data integrity, and optimizing system performance.

9.2.1 Setting Up and Managing the Python Virtual Environment

1. **Creating the Virtual Environment:**
    - Use Python's venv module to create a virtual environment within the server directory. This isolates project dependencies from the global Python installation, preventing conflicts with other projects.
    - Activate the virtual environment and install dependencies using the requirements.txt file.
2. **Regular Environment Updates:**
    - Periodically update the virtual environment to ensure that all packages are current. Use pip install --upgrade -r requirements.txt to update dependencies.

9.2.2 Data Backup Procedures

Regular data backups are essential to safeguard against data loss due to unforeseen circumstances such as hardware failures, software errors, or cyberattacks. Administrators should implement the following backup procedures:

- **Schedule Automatic Backups**: Use MongoDB's backup tools to automate the process, ensuring that backups occur at regular intervals (daily, weekly, or monthly).
- **Offsite Storage**: Store backups in a secure offsite location or cloud service to protect against local disasters.

- **Verify Backup Integrity**: Regularly test backups to ensure they can be restored successfully, conducting restore tests on a sample of backup data.

9.2.3 Log Management

Effective log management allows administrators to monitor system activity and identify potential issues before they escalate. Key practices include:

- **Centralized Logging**: Implement a centralized logging system that collects logs from all components of the person identification system, making it easier to analyze and troubleshoot issues.
- **Regular Log Review**: Schedule periodic reviews of system logs to detect unusual patterns, errors, or security incidents. This proactive approach helps in identifying vulnerabilities and optimizing system performance.
- **Log Retention Policy**: Establish a log retention policy that defines how long logs should be kept, considering compliance with legal regulations and organizational policies.

9.2.4 Security Maintenance Practices

Maintaining system security is paramount in protecting sensitive data and operations. Administrators should adopt the following security maintenance practices:

- **Regular Software Updates:** Keep the system and its dependencies up-to-date by applying security patches and software updates promptly, mitigating vulnerabilities that could be exploited by attackers.
- **User Access Management:** Review user accounts and access privileges regularly to ensure that only authorized personnel have access to sensitive information and system functionalities.
- **Conduct Security Audits**: Perform periodic security audits to assess the effectiveness of existing security controls and identify areas for improvement, including vulnerability scanning and penetration testing.

By adhering to these routine maintenance tasks, administrators can significantly enhance the reliability, performance, and security of the person identification system, ensuring it remains a robust tool for organizations.

9.3 Recomended to Add in the Future: Monitoring and Alerts

9.3.1 Significance of Monitoring System Performance

Regular monitoring of system performance is essential for ensuring optimal operation and reliability. Key performance indicators (KPIs) such as response times, server loads, and error rates provide valuable insights into the system's health. By analyzing these metrics, administrators can identify trends, detect anomalies, and make informed decisions regarding resource allocation and system

upgrades. Furthermore, ongoing performance monitoring helps in maintaining a high user satisfaction level by minimizing downtime and ensuring that the system meets user demands.

9.3.2 Setting Up Alerts

Establishing a robust alert system is crucial for proactive maintenance. Alerts notify administrators of potential issues, allowing for timely interventions. Here are some key considerations for setting up alerts:

- **Define Critical Metrics**: Identify the most critical KPIs that, when exceeded or fall below certain thresholds, could indicate a problem. For instance, high CPU usage, low memory availability, or excessive error rates should trigger alerts.
- **Choose Alerting Tool**s: Utilize monitoring tools such as Prometheus, Grafana, or Nagios, which can efficiently track performance metrics and send alerts via email, SMS, or messaging apps. Configure these tools to ensure they provide real-time notifications of any issues.
- **Establish Escalation Procedures**: Develop clear escalation procedures for alerts that require immediate attention. This includes defining who should respond to alerts and the actions they should take based on the severity of the issue.

9.3.3 Handling Alerts

An effective response to alerts can minimize system disruptions. The following guidelines can assist administrators in managing alerts efficiently:

- **Prioritize Alerts**: Not all alerts carry the same weight; prioritize them based on severity and impact on users. High-priority alerts should be addressed immediately, while lower-priority ones can be scheduled for later investigation.
- **Document Response**s: Maintain a log of alerts and the corresponding actions taken. This documentation will help identify patterns over time and improve the response process.
- **Review and Optimize**: Regularly review the alerting system to ensure that it remains relevant and effective. Fine-tune thresholds and notification protocols based on historical data and changing system requirements.

By implementing diligent monitoring practices and a structured alerting system, administrators can enhance the performance and reliability of the person identification system, ensuring that it continues to operate smoothly and securely.

9.4 Troubleshooting and Debugging

Effective troubleshooting and debugging are essential for maintaining the integrity and performance of the person identification system. When issues arise, employing systematic strategies can help

diagnose common problems, utilize debugging tools effectively, and handle system recovery in case of failures.

9.4.1 Diagnosing Common Problems

To troubleshoot effectively, begin by identifying the symptoms of the issue. Gather data by asking users specific questions about their experiences. Key steps include:

- **Replicate the Issue**: Attempt to reproduce the problem in a controlled environment. This can help determine whether the issue is systemic or isolated to a specific user or instance.
- **Check Logs**: Review system logs for error messages or warnings that occurred around the time the issue was reported. Logs can provide insights into the underlying cause.
- **Consult Documentation**: Refer to the system's documentation for known issues and their resolutions. This can save time and offer immediate solutions.

9.4.2 Using Debugging Tools

Debugging tools are invaluable in diagnosing and resolving issues. Several tools can be utilized effectively:

- **Integrated Development Environments (IDEs)**: Tools such as Visual Studio Code or PyCharm often have built-in debugging features that allow you to step through the code, inspect variables, and evaluate expressions.
- **Profilers**: Use profilers to analyze system performance, identify bottlenecks, and monitor resource usage. These tools can help pinpoint where optimizations are needed.
- **Network Analyzers**: For connectivity issues, tools like Wireshark can help monitor network traffic and diagnose communication problems between components of the system.

Handling System Recovery In the event of a system failure, having a robust recovery plan is vital. Key strategies include:

- **Data Backups**: Regularly scheduled backups of the MongoDB Compass database and configuration files are essential. In the event of a failure, these backups can be restored to recover lost data.
- **Rollback Procedures**: Implement rollback procedures for updates or changes. If a new update causes issues, being able to revert to a previous stable version can minimize downtime.
- **Disaster Recovery Plans**: Establish a comprehensive disaster recovery plan that outlines procedures for various failure scenarios. This should include contact information for key personnel, recovery time objectives, and step-by-step recovery processes.

By employing these strategies—diagnosing problems systematically, utilizing the right debugging tools, and having a solid recovery plan—system administrators can ensure that the person identification system remains reliable, effective, and ready to meet user needs.

9.5 Documentation and Updates

Keeping documentation up-to-date is a fundamental aspect of effective system maintenance and knowledge transfer among team members. As technology and user needs evolve, so too must the documentation that supports the person identification system. Accurate, comprehensive, and current documentation ensures that all stakeholders—administrators, developers, and end-users—have access to the most relevant information, thereby enhancing efficiency and reducing errors in system operation.

9.5.1 Importance of Up-to-Date Documentation

- **Consistency in Operations**: Up-to-date documentation provides a consistent framework for users and administrators to follow, reducing confusion and ensuring that everyone is on the same page regarding system functionality and procedures.
- **Facilitating Training and Onboarding**: New team members can quickly acclimate to the system when they have access to current documentation. Well-maintained resources help to streamline training, allowing for effective knowledge transfer and minimizing the learning curve.
- **Enhancing Troubleshooting**: When issues arise, having accurate documentation available can expedite the troubleshooting process. It allows users to reference known problems and solutions, facilitating faster recovery and minimizing system downtime.
- **Ensuring Compliance**: Many industries require adherence to specific regulations and standards. Keeping documentation updated ensures compliance with legal and organizational requirements, reducing the risk of penalties.

9.5.2 Strategies for Effective Knowledge Transfer

To ensure that knowledge is effectively transferred among team members, consider implementing the following strategies:

- **Version Control**: Utilize version control systems for documentation, allowing team members to track changes, access previous versions, and understand the evolution of procedures and guidelines.
- **Regular Review Cycles**: Establish a schedule for reviewing and updating documentation regularly. This could be quarterly or biannually, depending on the pace of changes within the system.
- **Collaborative Documentation Practices**: Encourage team members to contribute to documentation updates. Using collaborative platforms such as Confluence or Google Docs

can facilitate this process, making it easier for everyone to participate in maintaining resources.

- **Documentation Repositories**: Centralize documentation in accessible repositories where team members can easily find and reference materials. This encourages continuous use and helps ensure that all members are working with the most current information.

By prioritizing the maintenance of documentation and implementing effective knowledge transfer strategies, organizations can reinforce the reliability and performance of the person identification system while fostering a knowledgeable and capable team.

10 References

[1] Adrián, S. C., David, F. J., & Cristina, L. G. (2022). Real-time human action recognition using raw depth video-based recurrent neural networks.

[2] Ahmed, R., & Abd-Elkawy, E. H. (2024). Improved Tomato Disease Detection with YOLOv5 and YOLOv8. Engineering, Technology & Applied Science Research, 14(3), 13922-13928.

[3] Bouazizi, O., Azroumahli, C., El Mourabit, A., & Oussouaddi, M. (2024). Road Object Detection using SSD-MobileNet Algorithm: Case Study for Real-Time ADAS Applications. Journal of Robotics and Control (JRC), 5(2), 551-560.

[4] Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., & Darrell, T. (2015). Long-term recurrent convolutional networks for visual recognition and description. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2625-2634).

[5] Fontes, C., Hohma, E., Corrigan, C. C., & Lütge, C. (2022). AI-powered public surveillance systems: why we (might) need them and how we want them. Technology in Society, 71, 102137.

[6] Fu, Y., Wang, X., Wei, Y., & Huang, T. (2019, July). Sta: Spatial-temporal attention for large-scale video-based person re-identification. In Proceedings of the AAAI conference on artificial intelligence (Vol. 33, No. 01, pp. 8287-8294).

[7] Ge, Z., Liu, S., Wang, F., Li, Z., & Sun, J. (2021). Yolox: Exceeding yolo series in 2021. arXiv preprint arXiv:2107.08430.

[8] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

[9] Kanan, C., & Cottrell, G. W. (2012). Color-to-grayscale: does the method matter in image recognition?. PloS one, 7(1), e29740.

[10] Kandel, I., & Castelli, M. (2020). How deeply to fine-tune a convolutional neural network: a case study using a histopathology dataset. Applied Sciences, 10(10), 3359.

[11] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25.

[12] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. nature, 521(7553), 436-444

[13] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. nature, 521(7553), 436-444.

[14] Lin, T. Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature pyramid networks for object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2117-2125).

[15] Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13 (pp. 740-755). Springer International Publishing.

[16] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). Ssd: Single shot multibox detector. In Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14 (pp. 21-37). Springer International Publishing.

[17] Nguyen, V. T., & Chu, D. T. (2023). Study on Tracking Real-Time Target Human Using Deep Learning for High Accuracy. Journal of Robotics, 2023(1), 9446956.

[18] Parkhi, O., Vedaldi, A., & Zisserman, A. (2015). Deep face recognition. In BMVC 2015-Proceedings of the British Machine Vision Conference 2015. British Machine Vision Association.

[19] Cao, Q., Shen, L., Xie, W., Parkhi, O. M., & Zisserman, A. (2018, May). Vggface2: A dataset for recognising faces across pose and age. In 2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018) (pp. 67-74). IEEE.

[20] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).

[21] Ren, S., He, K., Girshick, R., & Sun, J. (2016). Faster R-CNN: Towards real-time object detection with region proposal networks. IEEE transactions on pattern analysis and machine intelligence, 39(6), 1137-1149..

[22] Roushdy, M. (2006). Comparative study of edge detection algorithms applying on the grayscale noisy image using morphological filter. GVIP journal, 6(4), 17-23.

[23] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4510-4520).

[24] Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 815-823).

[25] Terven, J., Córdova-Esparza, D. M., & Romero-González, J. A. (2023). A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas. Machine Learning and Knowledge Extraction, 5(4), 1680-1716.

[26] Tran, D., Bourdev, L., Fergus, R., Torresani, L., & Paluri, M. (2015). Learning spatiotemporal features with 3d convolutional networks. In Proceedings of the IEEE international conference on computer vision (pp. 4489-4497).

[27] Wang, J., & Lee, S. (2021). Data augmentation methods applying grayscale images for convolutional neural networks in machine vision. Applied Sciences, 11(15), 6721.

[28] Wang, M., & Deng, W. (2021). Deep face recognition: A survey. Neurocomputing, 429, 215-244.

[29] Wang, P., Li, W., Gao, Z., Tang, C., & Ogunbona, P. O. (2018). Depth pooling based large-scale 3-d action recognition with convolutional neural networks. IEEE Transactions on Multimedia, 20(5), 1051-1061.

[30] Xiao, Y., Chen, J., Wang, Y., Cao, Z., Zhou, J. T., & Bai, X. (2019). Action recognition for depth video using multi-view dynamic images. Information Sciences, 480, 287-304.

[31] Xu, C., Liu, Y., Chen, D., & Yang, Y. (2022). Direct training via backpropagation for ultra-low-latency spiking neural networks with multi-threshold. Symmetry, 14(9), 1933.

[32] Zagitov, A., Chebotareva, E., Toschev, A., & Magid, E. (2024). Comparative analysis of neural network models performance on low-power devices for a real-time object detection task. Computer, 48(2).

[33] Zhao, Z. Q., Zheng, P., Xu, S. T., & Wu, X. (2019). Object detection with deep learning: A review. IEEE transactions on neural networks and learning systems, 30(11), 3212-3232.

[34] Zheng, K. (2019). Person Identification with Convolutional Neural Networks (Doctoral dissertation, University of South Carolina).

[35] Zheng, Z., Wang, P., Liu, W., Li, J., Ye, R., & Ren, D. (2020, April). Distance-IoU loss: Faster and better learning for bounding box regression. In Proceedings of the AAAI conference on artificial intelligence (Vol. 34, No. 07, pp. 12993-13000).

[36] YOLO Pseudocode: https://gist.github.com/sjain07/7b880ed494467bd44e8c4f051703a011

[37] Ultralytics YOLO Docs : https://docs.ultralytics.com/models/yolov8/

[38] YOLO v8 vs. MobileNet SSD v2 https://roboflow.com/compare/yolov8-vs-mobilenet-ssd-v2

[39] IOU EXPLAINED https://www.v7labs.com/blog/intersection-over-union-guide

[40] Cocodataset official https://cocodataset.org/#home

[41] pycocotools API - https://pypi.org/project/pycocotools/

[42] FiftyOne Integrations COCO Integration https://docs.voxel51.com/integrations/coco.html

[43] Amazon Rekognition https://aws.amazon.com/rekognition/

[44] Microsoft Azure Face API
https://learn.microsoft.com/en-us/azure/ai-services/computer-vision/overview-identity

[45] Face ++ https://www.faceplusplus.com/

[46] Deepvisionai https://deepvisionai.in/

[47] Chat GPT https://chatgpt.com/

[48] React https://react.dev/

[49] Python https://www.python.org/

[50] MongoDB https://www.mongodb.com/

[51] FastAPI https://fastapi.tiangolo.com/

11 AI Prompts

During the research phase of our project, we utilized ChatGPT[47] intermittently to validate our proposed technology stack, check grammar, and brainstorm additional features for our application. Here are some example prompts and explanations of how their responses guided our research and development decisions.

- Create for me a short literature about Current uses of object detection remember to include references
  https://chatgpt.com/share/efabf011-4832-4da0-8e70-0813a2d431c1
- Summarize Dong, Z., Zhou, Z., Li, Z., Liu, C., Huang, P., Liu, L., ... & Kang, J. (2018). Convolutional neural networks based on RRAM devices for image recognition and online learning tasks. IEEE Transactions on Electron Devices, 66(1), 793-801.
  https://chatgpt.com/share/757eb2c5-64e5-4b74-918a-7fe584247374
- Create for me a two page literature review about Siddiqi, M. H., Ali, R., Rana, M. S., Hong, E. K., Kim, E. S., & Lee, S. (2014). Video-based human activity recognition using multilevel wavelet decomposition and stepwise linear discriminant analysis. Sensors, 14(4), 6370-6392.
  https://chatgpt.com/share/9c4e34d7-e73b-4a12-92d9-8534220824e4
- Create for me a two page literature review about yolo based on Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).
  https://chatgpt.com/share/3f9f7a91-d256-4e71-9666-dc23feb94384
- Sort the article references by A-Z
  https://chatgpt.com/share/24a67429-d362-4e43-92e0-5fde4f191f70