Ingénierie des Systèmes d'Information
Vol. 30, No. 12, December, 2025, pp. 3263-3272
Journal homepage: http://iieta.org/journals/isi

Multi-Face Detection and Recognition in Video Streams Using Tracker-Assisted, Color-Filtered Haar Cascade and Lightweight Convolutional Neural Network Training

Yasser H. Alwan[1,2*], Muayad S. Croock[3], Ahmed A. Oglah[1]

[1] College of Control and Systems Engineering, University of Technology, Baghdad 10066, Iraq
[2] ITRDC, University of Kufa, Najaf 54003, Iraq
[3] College of Electrical Engineering, University of Technology, Baghdad 10066, Iraq

Corresponding Author Email: yasser.alwan@uokufa.edu.iq

https://doi.org/10.18280/isi.301219

Received: 28 October 2025
Revised: 18 December 2025
Accepted: 26 December 2025
Available online: 31 December 2025

Keywords:
tracker-assisted face detection, face recognition, Haar cascade classifier, Kernelized Correlation Filter, Convolutional Neural Network, skin-tone filter, self-supervised dataset creation

ABSTRACT

This study presents a fast multi-face detection and recognition system suitable for surveillance applications using a self-supervised dataset generation approach. The face detection subsystem employs pretrained frontal and profile Haar cascade classifier (HCC) that are preprocessed for light normalization and postprocessed for eliminating duplicate detections. The detector is assisted by skin-tone postprocessing and a Kernelized Correlation Filter (KCF) tracker. To achieve a significant speed gain, HCC runs sparsely while KCF maintains face tracking-detection in the skipped frames. This hybrid HCC achieves F1-score improvements of 0.029 to 0.245 over baseline HCC and approximately 12× speed gain. Although Multi-Task Convolutional Neural Network (MTCNN) and YOLOv8 achieve higher F1 scores, hybrid HCC maintains superior frame rates, operating approximately 2-9× faster than these advanced models. Detected faces are extracted, augmented, and organized into a dataset (66.6% training, 33.3% testing split), enabling a custom lightweight Convolutional Neural Network (CNN) to successfully handle face recognition and achieve a perfect F1-score in a prediction test. The complete system demonstrates an effective solution for real-time surveillance applications in resource-constrained environments.

1. INTRODUCTION

The past few years have witnessed intensive research efforts to develop highly efficient and cost-effective people identification systems for quickly and accurately identifying individuals in various fields, including security and attendance tracking, to name a few. Face identification is a vital technology within this field, because the face is the most prominent personal identity [1, 2]. This process typically consists of two main phases: face detection and face recognition. It begins with face detection, which identifies and localizes human faces within images or video frames, followed by face recognition, where the detected face is classified by comparing and matching it against a database to confirm the individual's identity [3, 4]. For a face recognition system to be successful, it must first be able to reliably detect faces. One of the classical and key techniques for face detection is Haar cascade classifier (HCC), which was developed by Viola and Jones [5] more than two decades ago. HCC is a pioneering and simple technique that still provides a highly effective solution for real-time face and object detection. The Viola-Jones method is based on four key pillars: calculating Haar features, generating integrated images to accelerate calculations, training using a simple neural network (AdaBoost), and using sequential classifiers to reduce processing time [6]. This combination gives it a significant speed advantage, making it an ideal starting point for time-constrained systems. On the other hand, Convolutional Neural Networks (CNNs), which is one of the simplest deep learning techniques that has been used in enormous applications, offer a more robust and adaptable solution for image classification and pattern recognition tasks than traditional techniques. CNNs utilize advanced mathematical concepts to detect complex patterns in an image. Its emergence has revolutionized face recognition, enabling a more authentic representation of original facial features. Through its ability to self-extract features and utilize deep learning, face recognition accuracy has seen significant improvements [7, 8].

Research in face detection and recognition in streaming videos is an active research area. The published work in this area has employed a variety of techniques that balance accuracy and speed. Mamieva et al. [9] built a face detector that enhanced the RetinaNet architecture, combining a region-offering network for spotting possible faces with a detection network for fine-tuning the boxes around each face. They trained it on the WIDER FACE dataset. Their model scored 41.0 Average Precision (AP) at 11.8 Frames per Second (FPS) with single-scale input, and 44.2 AP with multi-scale, which makes it competitive against other models. In PyTorch, it achieved a 95% face detection accuracy, which is excellent given that the model is lightweight. Despite that, they acknowledged that their model has challenges with poor

lighting and blurry frames. Gupta et al. [10] compared classic and modern face detection and recognition methods. They stacked up Histogram of Gradient (HOG), HCC, CNN, Local Binary Pattern Histogram (LBPH), and ResNet-34 against each other. According to them, HOG + Support Vector Machine (SVM) for detection paired with ResNet for recognition gives the most reliable results. They stressed the value of building recognition datasets straight from the detection phase, though they did not provide performance metrics that support the effectiveness of this strategy. Focusing on real-time performance, Majeed et al. [11] developed a system that used an HCC for face detection, a lightweight CNN for face recognition, plus feature extraction with Linear Discriminant Analysis (LDA) and HOG. They showed that these compact CNNs can hit 100% accuracy even with poor lighting and face pose changes. They ran their experiments on both standard and video-derived datasets (extracted from videos). That being said, they did not provide any metrics to evaluate the detection phase. Furthermore, the videos they used are of a single class and were set in a controlled environment. Similarly, Wong et al. [12] developed a multi-camera system for face detection, recognition, and tracking that was built for real-world conditions. They used You Only Look Once (YOLOv5n) for detection (with a mean AP of 0.495, precision at 0.868, and recall at 0.781), SphereFace for recognition (scoring 82.05% accuracy), and DeepSORT to keep track of identities across different cameras. They mentioned that using OpenVINO for optimization could push performance even more on edge devices. However, not all hybrid approaches proved suitable for real-time use. Abbattista et al. [13] combined HOG and SVM for detection with a ResNet-34 model for recognition. On the CORDOBA dataset, their setup hit 98% accuracy for detection and 100% for recognition. The turning point was that their system was very slow (recognition cycle took about a second), so it could not handle real-time demands. Finally, Haq et al. [14] built a mobile application that recognizes cricket players in real time. It uses AdaBoost for detection and LDA for recognition, and they tested it on datasets like YTF, LFW, and actual sports videos. They claimed that their application responded instantly in real-world use, though they did not report any performance metrics.

Previous research has already shown that cutting-edge methods boost recognition performance by using standardized datasets. Yet, most of these methods just analyze the whole image, which includes processing unnecessary background information. Normally, that extra clutter downgrades model efficiency. Furthermore, many studies have focused on achieving the highest accuracy at the cost of practicality of using these models for real-time applications. That can be justified by knowing that computers have gotten much faster; thus, worrying about computational efficiency is not an urgent issue most of the time. Conversely, few researchers have built datasets from live video streams or tried to lock recognition to just the face area without the background. The work in study [10], for example, overlaps with this study, but as mentioned above, those efforts have not included performance metrics that show the efficiency of this method.

This research gap of the absence of a simple method to build real-time datasets focused just on facial data is considered in this work, which would sharpen both accuracy and efficiency. A self-supervised approach that extracts facial regions directly from video streams during face detection results in a recognition model that runs faster and performs better in real-time applications. Face detection phase really matters here. If face detection is slow or unreliable, the whole system suffers, no matter how good the recognition model is. Concisely, the aim of this work is to realize a fast, reliable, compact and data efficient face detection and recognition system.

This work introduces a system that can detect and recognize different people across multiple video streams. For face detection, a hybrid HCC approach is used. This approach combines detection and tracking into one process. Furthermore, it uses both frontal and profile HCC models to detect faces in different angles, and it is combined with preprocessing and postprocessing. Preprocessing includes downsizing the frames, converting them to grayscale, and correcting them for lighting. Postprocessing includes filtering for skin tone and eliminating any duplicate faces that can result from using frontal and profile HCCs. Since this is a heavy load computationally, it is done sparsely. Kernelized Correlation Filter (KCF) tracker completes the task and continues to track faces that are detected. This hybrid method speeds up the detection and pushes performance to the real-time demands.

Once faces have been detected, they are gathered, processed, and sorted into a dataset for training and testing the recognition phase. When it comes to recognition, a lightweight CNN is utilized and trained on the extracted and augmented dataset. The final step is testing the whole system on a new video of the same target individuals to validate performance.

This approach is a real transition from the usual methods that depend on previously made datasets. Instead, datasets are built straight from live video feeds. This model works especially well in closed environments, such as a private surveillance system, where public datasets are often unsuitable or unavailable. Focusing training solely on isolated faces instead of whole bodies or background excludes a lot of background noise and environmental clutter, which really escalates recognition accuracy. Figure 1 shows the whole system block diagram.

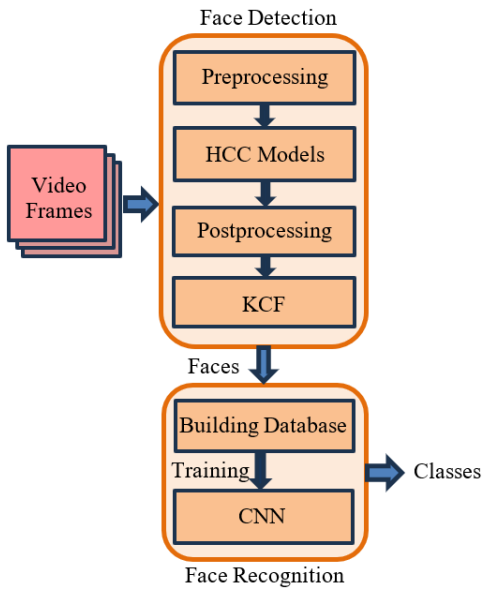


Figure 1. The general structure of the face detection and recognition system

The following sections of this article are organized as follows: a short theory of the techniques that are utilized, system design theory, followed by the results and discussions, and finally, conclusions and references.

2. RESEARCH METHODOLOGY

This work utilizes the HCC algorithm for face detection, supported by a KCF tracker, with a CNN used as a next step for face recognition. While these techniques have become widespread over the past decade, this research aims to demonstrate that even traditional methods, with minor processing modifications, can form the efficient core of an intelligent face detection and recognition system without requiring general standard datasets. By showing that these basic methods can meet the system's fundamental requirements, this work lays a foundation upon which to build for future advancements in more complex techniques. The following section provides the necessary theoretical groundwork of HCC, KCF, and CNN before discussing system design and experimental results.

2.1 Haar cascade classifier

HCC, based on an algorithm introduced by Viola and Jones [5], uses a rectangular feature known as a Haar feature as a key input for the cascaded classifier. Viola and Jones drew inspiration from Haar wavelets to develop what are now known as Haar features. By analyzing adjacent rectangular regions in a detection window, summing pixel intensities within these regions, and computing the difference between these sums, these features enable the categorization of image subsections. One of the most employed Haar features for face detection is formed of two rectangles positioned above the eye and cheek regions, considering the difference in light intensity between these areas. The positions of these rectangles are determined with respect to a detection window, which is a bounding box for the face of interest. During detection, a window with a size corresponding to the target object is moved across the input image. For a given sub-region of an image, a Haar feature is computed and compared with a learned threshold for distinguishing objects and non-objects. Because a Haar feature is only a weak learner, a set of features is needed for obtaining a precise definition of an object. These features are arranged as a classifier cascade for developing a strong classifier. The most prominent benefit of using Haar features compared with other approaches is related to the computation speed provided by integral images, allowing for a constant time computation of features of arbitrary sizes. Figure 2 shows different detectors using a Haar feature.

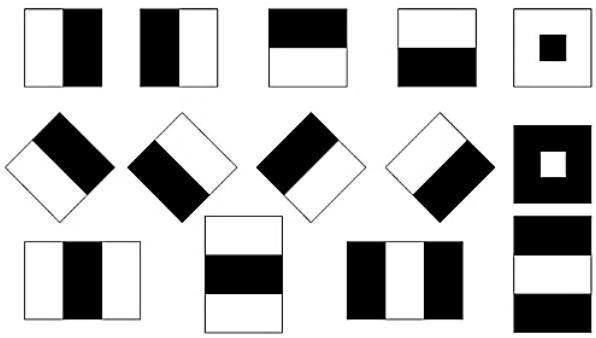


Figure 2. Types of detectors based on Haar features [15]

These detectors employ different filters on specific regions of an image, where the pixel sums from the white areas are subtracted from those in the black areas. The weights for the white and black regions are treated as values of "1" and "–1," respectively. Haar features inspect adjacent rectangular areas within a detection window, sum the pixel intensities of each area, and calculate the difference. The difference is then used to classify sections of the image. The key components of the Haar cascade include integral images, the Adaboost algorithm, and the cascaded classifier [15].

The first step of the Viola-Jones algorithm is to convert the input image into an integral image. HCC, then, compares how closely the actual scenario matches the ideal case, where the value of the Haar feature is ideally one. Then the Adaboost Algorithm, a machine learning algorithm, is used to develop a classifier through weights [5].

The cascaded classifier is used for quickly rejecting error windows and enhancing processing speed. In every node of the trees, there is a non-object branch, indicating that the image will not be the desired object, as shown in Figure 3. This technique minimizes the false negative rate [16].

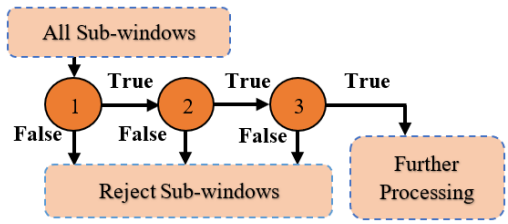


Figure 3. Schematic depiction of the detection cascade [5]

2.2 Kernelized Correlation Filter

KCF is a fast target-tracking algorithm that works through three main stages: feature extraction, online learning, and template updating, as shown in the block diagram in Figure 4.

First, the tracker extracts HOG features from the selected target region and converts them into the frequency domain using the Fourier transformation. It then computes the correlation in the frequency domain to estimate the new target position. After locating the target, the classifier is trained by generating multiple cyclic shifted versions of the target patch and learning their weights using a ridge-regression-based classifier. The process of cyclic shifting converts a single patch into a huge training dataset, allowing the tracker to depend on very limited data to produce efficient discriminative tracking. The tracker continues to follow the object by repeatedly updating its model and filtering weights based on the most recent target location, allowing real-time and continuous tracking [17, 18].

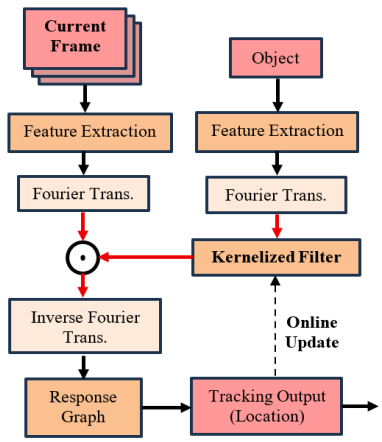


Figure 4. General framework of KCF [19]

2.3 Convolutional Neural Networks

Neural networks are a contemporary technology that provides high precision in tasks related to classification and prediction. Their high speed and compatibility with various scientific fields make them one of the most used techniques. There are different types of neural networks, that vary in different aspects such as structure, level of hybridization with other computational algorithms, and complexity. These networks share common elements such as neurons, input and output layers, and one or more hidden layers, but they differ in the tasks performed by these layers.

CNN performs classification tasks, either supervised or unsupervised. Supervised training involves providing a set of corresponding inputs and outputs, and the system learns to associate them and predict the output. Initially, input images are entered, and several of their properties, such as edges and gradients, are calculated in the primary layer. In the middle stage, several features are extracted from the previous stage, and then in the final stage, the features are extracted, which can then be passed to the classifier [20, 21].

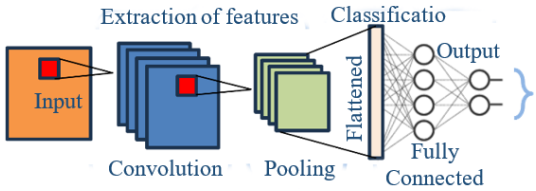


Figure 5. Basic CNN architecture [22]

A CNN is typically structured with multiple layers (as shown in Figure 5), including [23-25]:

- *Convolutional Layer*: This is a key layer in a standard neural network. It contains a variety of filters, often referred to as convolutional kernels (Figure 6). These filters interact with the intended input to generate an output that has the strongest correlation with the input features. During training, the weights of the filter are adjusted. This task is also known as subsampling because it decreases the size of the samples in the image. To ensure that the filter moves over the edges as well, we need to add zero pixels (padding) to the edges of the image.
- *Pooling Layer*: This layer selects a pixel from each mask, which could be the average or the maximum pixel.
- *Activation Function*: These functions determine the threshold at which the neuron triggers the output.
- *Fully Connected Layer*: This is the final stage of CNN. It functions as the classifier by connecting each neuron to all neurons in the preceding layer. Essentially, it adopts the feed-forward approach of traditional multi-layer perceptron neural networks. The FC layer receives input as a vector, which comes from the last pooling or convolutional layer, after flattening the feature maps. The final output of the CNN is derived from this layer.
- *Loss Functions*: these are applied in the output layer to compute the prediction error across training samples, reflecting the difference between the actual and predicted outputs. This error is then minimized during the CNN learning process to enhance accuracy.

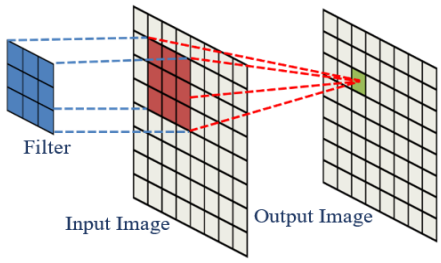


Figure 6. Convolution operation [26]

In CNN models, the primary challenge is achieving good generalization without overfitting. Overfitting occurs when the model fulfills the task on training data but fails on test data. On the other hand, an under-fitted model lacks sufficient learning from the training data, leading to poor performance on training and consequently on test data also. A well-fitted model, however, performs well on both training and test data. Various intuitive concepts are employed to aid in regularization and avoid overfitting, as further explained in the subsequent sections [24, 27]:

- *Dropout*: During each training epoch, certain neurons are randomly omitted. This distribution of feature selection power ensures the model learns different independent features. In training, the dropped neurons are excluded from back-propagation and forward-propagation. However, in the testing phase, the entire network is used for prediction.
- *Drop-Weights*: This technique is similar to dropout; this method involves omitting the connections between neurons (weights) during each training epoch, rather than the neurons themselves.
- *Data Augmentation:* Expanding the training dataset with artificial techniques helps prevent overfitting. With more examples to learn from, the model handles data better and generalizes instead of just memorizing.
- *Batch Normalization:* This technique normalizes the activations of the output at each layer, aligning the values with a unit Gaussian distribution. Basically, the mean is subtracted from each output, and the result is divided by the standard deviation.

3. SYSTEM DESIGN

Although face detection and recognition systems have achieved remarkable advancements recently, the main challenge remains keeping accuracy high without draining resources, especially on devices that do not have much computational power. Deep learning models like Multi-Task Convolutional Neural Network (MTCNN) and YOLO almost nail detection accuracy, but their computational load is high. That represents a problem for cheaper devices or real-time scenarios, where boosting speed counts even if it means losing some of the accuracy. In those cases, it is more important to reduce False Positives (FP) rates than to catch every possible False Negative (FN).

In this context, Classic techniques like HCC are still relevant because they are simple, fast, and accurate enough for most real-time needs. But they are not perfect and are known for producing a lot of FPs and struggling with faces that are not in a frontal pose or have poor lighting [15, 28]. Rather than changing the internal structure of HCC, combining it with

other methods can target these specific problems.

This work takes that route. It introduces a hybrid system that builds on HCC, aiming to boost detection performance while keeping things lightweight. The system brings together three strategies: widening the detection range, filtering by skin tone, and adding KCF tracking.

The first step expands the capabilities of pretrained HCC models found in OpenCV libraries. Combining HCC models that are specifically trained for frontal and profile faces is expected to drop FN rates because faces at different angles are detected. The tradeoff of this step is more FPs, more repeated detections of the same face, and heavier computation.

To tackle these side effects, the second step uses skin-tone filtering after detection. By checking how much of the detected area matches human skin tones, the system can remove many FPs. An additional filter is used to cancel duplicates from the frontal and profile detectors. These tweaks lower the FP rate and are expected to have a mild effect on FN rates. Yet, they do increase computational demand even more.

Finally, to keep resource use in check, the third step inserts a KCF tracker. The idea is to run the enhanced HCC from the past two steps on a key frame, then use KCF to track each face across the following frames. This way, the system does not have to run heavy detection every single time, so it can process frames much faster. The risk here is missing new faces or losing track during quick movement. The system can manage this by adjusting the number of skipped frames to match the speed of movement of individuals across the frames. The result is a careful balance between speed and accuracy. Figure 7 shows the steps of this process.

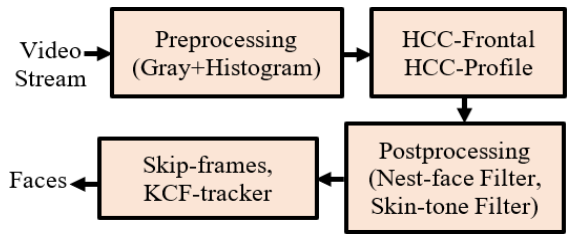


Figure 7. Face detection process using a hybrid Haar cascade classifier (HCC)

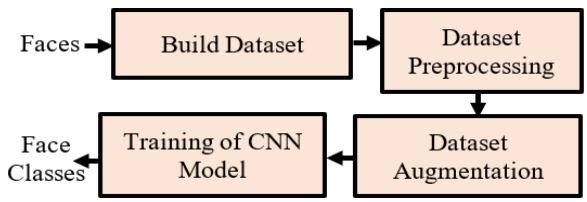


Figure 8. Face recognition process using CNN

The face recognition phase utilizes the power and capabilities of CNN. It is trained on a dataset extracted straight from video clips during the face detection phase. This dataset contains only face information, which gives CNN a big boost during the process of training and recognition. Despite the neatness of this approach, there is a hitch. Faces from the same video usually look similar with the same angle, same lighting, same expressions. That is the typical recipe for overfitting. To tackle this, the system selects images from frames spaced apart in time, which brings in more variety. It also uses extensive data augmentation during training, helping the model generalize better and making overfitting less likely. The training set draws from several video clips, while a separate video builds the validation set, ensuring a more objective evaluation of the model's performance. The block diagram of the face recognition phase using CNN is shown in Figure 8.

4. RESULTS AND DISCUSSIONS

The developed methodology consists of two main parts: first, a hybrid face detection method (HCC-KCF), and then a CNN-driven recognition phase. The system was implemented on Python 3.11, with the help of OpenCV and Keras (TensorFlow as backend and Adam as the optimizer). All the experiments took place on a Windows 10 laptop with an Intel Xeon CPU, 4GB NVIDIA Quadro GPU, and 32GB RAM.

Python makes sense here. It is flexible, efficient, and supports object-oriented programming, which accelerates prototyping and development. The simple yet powerful syntax of Python, along with direct interpreting, makes it a perfect choice for computer vision and machine learning work [29].

OpenCV handled the main computer vision tasks, offering a deep toolkit for face detection, object recognition, and real-time video analysis. TensorFlow, Google's open-source deep learning library, supports the efficient building and training of complex neural networks [30].

For face detection, pretrained HCC models in the OpenCV libraries did the main task. OpenCV offers four frontal face detection models, though there is no documentation explaining how they differ from each other. They are listed below with names used throughout this paper for convenience:

- HCC-default (haarcascade_frontalface_default.xml).
- HCC-alt (haarcascade_frontalface_alt.xml).
- HCC-alt2 (haarcascade_frontalface_alt2.xml).
- HCC-alt-tree(haarcascade_frontalface_alt_tree.xml).

Determining the best HCC frontal face model required comparing the performance of these models. To this end, all models were tested on three diverse video samples of three individuals that vary in gender and skin complexion, to provide diversity in the samples. Figure 9 gives a glance at the content of these videos: Video-A (Figure 9(a)) has a rich background, Video-B (Figure 9(b)) has relatively poor lighting, and Video-C (Figure 9(c)) has a well-lit condition. These videos are available online and free to use and reproduce the results. To enhance testing efficiency, the resolution of each video was reduced to 50% of its original size. Before the detection process, the frames underwent standard preprocessing, including grayscale conversion and lighting enhancement using histogram balancing. To ensure fair comparison, all models used standardized settings: (scale factor = 1.1, minimum proximity = 5, minimum size = (60, 60)). Reference positions for faces were manually determined to serve as the basis for evaluation.

Table 1 shows the test results. It can be noticed that some models perform better than others in some metric criteria, but there is no model that is superior in all the metrics. Further analysis revealed that HCC-alt2 is the most balanced choice, as it strikes the best practical compromise between speed and detection reliability. While HCC-alt achieved slightly higher precision, it did so at a high cost to recall. Conversely, HCC-default suffered from low precision, and HCC-alt-tree, despite its high speed, was unusable due to catastrophically low recall. HCC-alt2 consistently delivered strong F1 scores across all test scenarios, offering robust performance with high speed, making it the most suitable and balanced candidate for our

hybrid HCC detector.

(a) Video-A (from: pexels.com/video/video-of-people-talking-while-walking-4625331)

(b) Video-B (from: pexels.com/video/video-of-people-talking-while-walking-4625293)

(c) Video-C (from: pexels.com/video/ people-having-conversation-while-walking-on-sidewalk-4625296)

Figure 9. Three videos of three class faces

Table 1. Performance evaluation of several pretrained front face HCC models found in the OpenCV library

Model Type	Eval. Metrics	Vid-A	Vid-B	Vid-C
Haar cascade classifier (HCC)-default	FPS	12.09	5.85	5.96
	Precision	0.7648	0.5570	0.4881
	Recall	**0.8144**	**0.6052**	**0.7609**
	F1-score	0.7888	0.5801	0.5947
HCC-alt	FPS	14.37	6.97	6.09
	Precision	0.9728	0.9627	0.7802
	Recall	0.7556	0.5187	0.7131
	F1-score	**0.8505**	0.6742	**0.7451**
HCC-alt2	FPS	15.26	7.13	6.79
	Precision	0.9360	0.882	0.7704
	Recall	0.7285	0.5526	0.7062
	F1-score	0.8193	**0.6795**	0.7369
HCC-alt-tree	FPS	**22.26**	**10.68**	**10.69**
	Precision	**1.0**	**1.0**	**1.0**
	Recall	0.0678	0.2969	0.3647
	F1-score	0.1271	0.4579	0.5345

There is only one pretrained profile model, HCC-profile (haarcascade_frontalface_profile.xml), in the OpenCV library; therefore, no need to make such a comparison. It must be mentioned that most profile face detectors are trained on a dataset of faces on one side. To cover all the possible face orientations, the HCC-profile needs to be run twice on each frame (normal frame and flipped one).

For convenience, we call our model that depends on HCC and KCF as hybrid-HCC. It consists of the HCC part and the KCF part. The HCC part ran two pretrained models, HCC-alt2 and HCC-profile. HCC-profile was used twice (on the normal frame and on the reversed one) to cover the left and right profile faces. The parameters of both were set to (scale factor = 1.1, minimum proximity = 5, minimum size = (60, 60)). Each frame fed to them is downsized by 50% of its original size. Then, the frame was preprocessed by grayscale conversion (integral image) and illumination correction (using histogram equalization). The postprocessing consists of two filters: the skin-tone filter and the nested-faces filter. The skin-tone filter rejects any candidate region where more than 40% of the color tone falls within the human skin range, defined in the HSV color space H=0-40, S=50-120, and V=80-255. For the KCF part, it was initialized by the detected faces in a frame. Then, the tracker follows the face from frame to frame until a new face detection happens, which resets the initialization. The HCC part ran for a frame, and the remaining 25 frames were run by the KCF part.

Table 2. Performance evaluation of the hybrid-HCC against other models

Model Type	Eval. Metrics	Vid-A	Vid-B	Vid-C
Ref-HCC	FPS	3.97	1.76	1.75
	Precision	0.8193	0.6509	0.5226
	Recall	0.9984	0.9511	0.9767
	F1-score	0.9000	0.7729	0.6809
Hybrid-HCC	FPS	**44.89**	**22.45**	**22.31**
	Precision	**1.0**	0.8155	**1.0**
	Recall	0.8672	0.8258	0.8620
	F1-score	0.9289	0.8206	0.9258
MTCNN	FPS	4.60	4.72	4.82
	Precision	0.9969	**0.9705**	0.9608
	Recall	0.9758	0.8245	0.9713
	F1-score	0.9862	0.8915	0.9660
YOLOv8	FPS	12.82	11.02	11.81
	Precision	**1.0**	0.9536	**1.0**
	Recall	**1.0**	**0.9736**	**1.0**
	F1-score	**1.0**	**0.9536**	**1.0**

To evaluate the above modification in the hybrid-HCC face detector, it must be tested against other face detectors. The first candidate was a pure HCC. Table 1 shows the performance metrics for one frontal HCC; therefore, a better version of HCC was used. It is called ref-HCC here for convenience. Ref-HCC consists of two pretrained models, HCC-alt2 and HCC-profile, just like the hybrid-HCC. The same preprocessing of grayscale conversion and illumination correction was used. Because multiple HCCs were used in ref-HCC, too, a fair comparison requires using the nested-faces filter in the postprocessing. The other two candidates were state-of-the-art pretrained detectors: MTCNN and YOLOv8n (nano). The choice of YOLOv8 out of other versions is due to its high performance and balance between accuracy and inference velocity [31]. It was selected because it provides an ideal trade-off between precision and processing speed. MTCNN is part of the OpenCV library [32], but YOLOv8 is not. Therefore, a pretrained YOLOv8n for face detection (yolov8n_100e.pt from github.com/Yusepp/YOLOv8-Face) was used. All these models were evaluated under consistent, resource-constrained conditions where video frames were downsized by 50%. All the models run on every frame of test videos, except the hybrid-HCC, which employed an aggressive, tracker-assisted skip of 25 frames. Table 2 shows the performance evaluation of the four models.

The most eye-catching result is the speed of hybrid-HCC. It is 11.3 to 12.75 times faster than ref-HCC. Moreover, it is nearly 9.7 times faster than MTCNN and about 2 to 3.5 times faster than YOLOv8. When it comes to accuracy (F1 score), YOLOv8 achieved the highest accuracy, topping 1.0 F1 score, followed by MTCNN. Hybrid-HCC outperformed ref-HCC in all test videos by 2.8 to 24 percentage points. Its accuracy, while still behind the newest face detectors, is very good and balanced. Hybrid-HCC achieved perfect Precision (1.0) in Vid-A and Vid-C, and 0.8155 in Vid-B. This is due to the effectiveness of the skin-tone filter at minimizing FP. Hybrid-HCC scored the lowest Recall rates across all the models. This decline is the consequence of the discrete detection strategy. Skipping many frames leads to an increase in FN when tracking was temporarily lost or failed to initiate because of missed detection. The results show a trade-off between accuracy and speed. YOLOv8 is the best in accuracy, Hybrid-HCC model is the best in speed, and MTCNN offers balanced but slower results. Ref-HCC model is significantly the worst. These findings support the goal of running a detection system on resource-constrained, real-time video surveillance, where high FPS is prioritized over perfect accuracy. Regarding the results per video, Vid-A is the easiest, while Vid-B is the most challenging. The main reason is that the brightness in Vid-B is lower than in other videos. This indicates that hybrid-HCC is more susceptible to low-light conditions than YOLOv8, but it is comparable to MTCNN.

After detecting faces in each frame, the face images are extracted and categorized into folders for the training stage in CNN. The CNN structure was set as shown in Table 3. It consists of four convolutional layers, where each one is followed by a batch normalization and a pooling layer (Max pixel). This gradually reduces the image dimensions from 256 × 256 to 16 × 16 and keeps the feature maps at 32. The feature maps are then flattened to a 6272-element vector. Next, the flattened vector is fed to a dense fully connected classifier. Batch Normalization and Dropout (at 0.8 value to reduce possible overfitting) are performed as regularization during training. The last layer is a fully connected dense layer that shapes the output of three classes. All the activation functions in convolution and dense layers are ReLUs (Rectified Linear Unit) except the last layer, which is activated by a SoftMax function. The total CNN parameters are 431,075 (1.64 MB) with 430,691 (1.64 MB) trainable parameters and 384 (1.50 KB) non-trainable parameters. This makes the CNN model a lightweight one that is capable of being deployed on limited hardware devices.

Table 3. Setting of the CNN model

The Type of Layer	The Output Shape	Param. Number
Conv2D	(254, 254, 32)	896
Batch Normalization	(254, 254, 32)	128
MaxPooling2D	(127, 127, 32)	0
Conv2D	(125, 125, 32)	9,248
Batch Normalization	(125, 125, 32)	128
MaxPooling2D	(62, 62, 32)	0
Conv2D	(60, 60, 32)	9,248
Batch Normalization	(60, 60, 32)	128
MaxPooling2D	(30, 30, 32)	0
Conv2D	(28, 28, 32)	9,248
Batch Normalization	(28, 28, 32)	128
MaxPooling2D	(14, 14, 32)	0
Flatten	(6272)	0
Dense	(64)	401,472
Batch Normalization	(64)	256
Dropout	(64)	0
Dense	(3)	195

CNN was trained using a set of 1,728 images, divided into 1152 images for the training set and 576 images for the test set, with a batch size of 32 and a learning rate of 0.0001. Both training and test sets were divided equally into three classes. The number of training steps per epoch was calculated as the number of training images divided by the batch size. In the same way, the number of validation steps was calculated as the number of test images divided by the batch size.

To overcome the overfitting problem that might result from the limited size and low diversity of the dataset, extensive data augmentation was employed. This involved applying various geometric transformations to the training images, such as rotation, translation, scaling, and reflection, thereby enhancing the model's ability to generalize and handle variations in real-world data. Images were rotated by 20°, shifted in width or height by a percentage up to 60%, heavily sheared and zoomed (both with a range of 0.4), and flipped horizontally. This level of augmentation increases the diversity of the training set and pushes CNN to handle all sorts of viewing conditions.

Figure 10 shows the accuracy and loss evolving through both training and testing. 100 training epochs were enough to achieve the required performance. Examining the accuracy graph (Figure 10(a)) and loss graph (Figure 10(b)) indicates the learning performance.

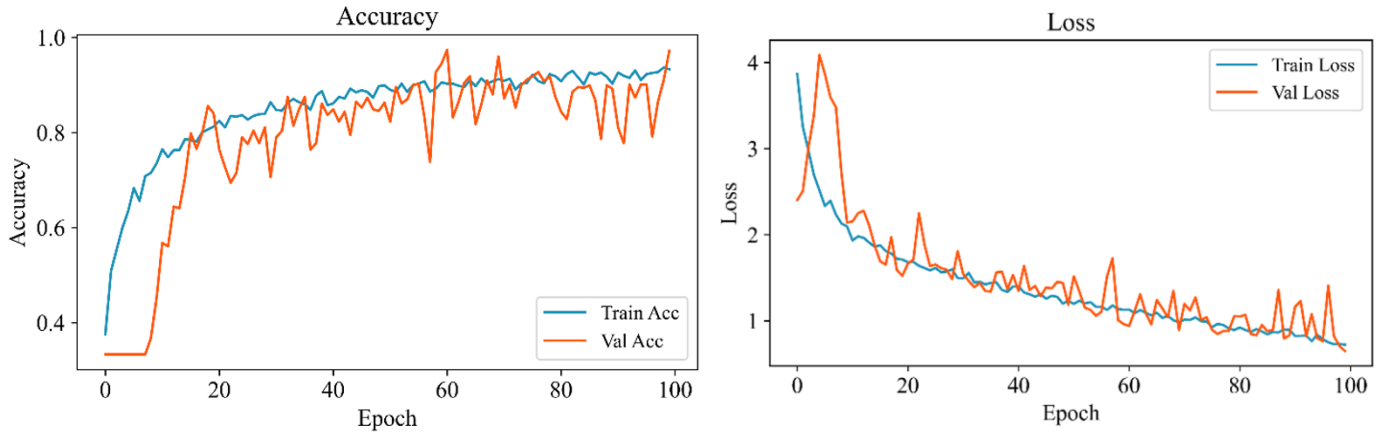
Training accuracy curve tops around 0.934, which is a solid sign that the model learned the patterns in the data excellently. Validation accuracy curve even tops around 0.9722, which reflects the good generalization and evades the trap of overfitting.

The training loss curve showed a decreasing shape and settled at about 0.7187, which means that the model is achieving more confidence in the classifications. Validation hits even lower end-value of 0.6485. It follows the same shape of validation accuracy. This ensures that overfitting is not a prominent issue here.

The spikes in validation accuracy and loss may suggest that

the model faces some challenges with certain samples, but ultimately, it maintains a consistent learning and loss trajectory. The observed effect may stem from the applied heavy augmentation on a relatively small dataset.

To test the recognition subsystem, the trained CNN model was combined with hybrid-HCC. The detector was applied to a new video. Since there was no truth table for the faces in the video, it was not feasible to test the performance directly. Therefore, a prediction sample of faces was extracted from the new video, and then they were organized into three classes. The prediction dataset consisted of 180 samples (60 for each class).



(a) Training and validation accuracy

(b) Training and validation loss

Figure 10. The CNN model performance: accuracy and loss across epochs

Table 4. Classification report of CNN

Class	Precision	Recall	F1-Score	Support
Girl	1.0	1.0	1.0	60
Guy1	1.0	1.0	1.0	60
Guy2	1.0	1.0	1.0	60

Table 4 shows that the model nailed every single prediction, hitting 100% total accuracy. That confirms the performance of CNN in handling new data. Furthermore, the detector provides clean, cropped faces that resemble the ones used during the training and testing of the CNN.

Figure 11 supports the tabled results, showing a sample frame from the new video where the system correctly detects and recognizes face classes (guy1, girl, guy2). It must be mentioned that this perfect score comes from a small, controlled prediction test. The accuracy may drop slightly if the test video is longer and more challenging. Nonetheless, these results show that this setup, with hybrid-HCC detection and the lightweight CNN, works smoothly and reliably for real-time, class-specific face recognition.

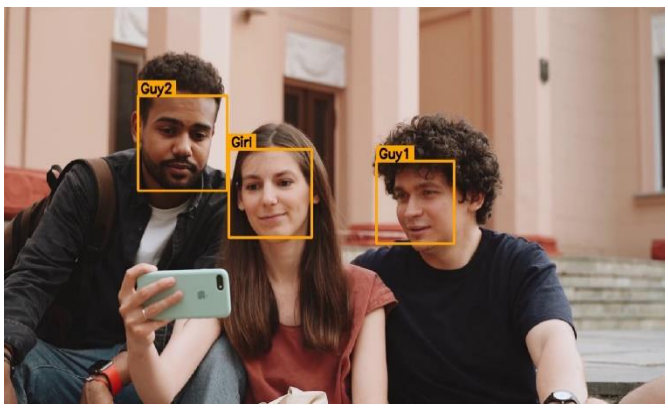


Figure 11. Detection of the face classes correctly throughout all frames (from: pexels.com/video/ a-group-of-people-together-for-a-photo-sjhot-4625282)

5. CONCLUSIONS

This study has introduced a system for face detection and recognition in video streams that is suitable for real-time applications such as surveillance systems. It combined a hybrid HCC for face detection and a custom lightweight CNN for face recognition. It has tackled two main challenges of improving speed and maintaining a decent performance. For detection, two pretrained HCC models (frontal and profile) have been used in combination to reduce FN, with an added preprocessing step to balance lighting and two postprocessing filters: one for skin tone and another for removing nested detections. Preprocessing and postprocessing worked to reduce the FP. Additionally, the detector was assisted by a KFC tracker that tracked faces when the detector was skipping frames to push the speed enough for real-time use. This hybrid HCC outperformed the reference HCC in performance (by 2.8 to 24 percentage points in F1-score) and speed (11.3 to 12.75 times faster). While MTCNN and pretrained YOLOv8 performance was better, hybrid HCC outperformed them in speed (2 to 9.7 times faster). The custom lightweight CNN represented a self-supervised training loop tailored for face recognition. The detected faces were extracted and augmented into a dataset (split into 66.6% training and 33.3% testing) that was utilized to train the CNN successfully. The CNN model achieved perfect prediction performance when it was tested on new data.

This approach proved that video streams can be utilized directly to build custom face recognition datasets, and HCC and CNN can be used for limited hardware devices while keeping decent performance and real-time demands. However, some limitations can be considered for future work. The hybrid HCC needs to be tested on real surveillance cameras to see the effectiveness of keeping detection while tracking in more challenging environments. CNN is trained offline in this work; therefore, using a half-trained face recognition system leads to a fully automated system, where online light training on a dynamic dataset can benefit from the speed gain of our

detector. Conversely, employment of knowledge distillation to compress large pretrained models to a lightweight CNN, without losing accuracy, can be an excellent next step.

REFERENCES

[1] Shi, Y., Zhang, H., Guo, W., Zhou, M., Li, S., Li, J., Ding, Y. (2024). Lighterface model for community face detection and recognition. Information, 15(4): 215. https://doi.org/10.3390/info15040215

[2] Ali, N.S., Alsafo, A.F., Ali, H.D., Taha, M.S. (2024). An effective face detection and recognition model based on improved YOLO v3 and VGG 16 networks. International Journal of Computational Methods and Experimental Measurements, 12(2): 107-119. https://doi.org/10.18280/ijcmem.120201

[3] Alkhan, T.E., Hameed, A.A., Jamil, A., Ja, A. (2021). Deep learning for face detection and recognition. In International Conference on Advanced Engineering, Technology and Applications, Istanbul, Turkey, pp. 148-153.

[4] Paulchamy, B., Yahya, A., Chinnasamy, N., Kasilingam, K. (2025). Facial expression recognition through transfer learning: Integration of VGG16, ResNet, and AlexNet with a multiclass classifier. Acadlore Transactions on AI and Machine Learning, 4(1): 25-39. https://doi.org/10.56578/ataiml040103

[5] Viola, P., Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Kauai, HI, USA. https://doi.org/10.1109/CVPR.2001.990517

[6] Dubovečak, M., Dumić, E., Bernik, A. (2023). Face detection and recognition using raspberry PI computer. Tehnički Glasnik, 17(3): 346-352. https://doi.org/10.31803/tg-20220321232047

[7] Ahmed, H.A., Croock, M.S., Al-Hayanni, M.A.N. (2023). Intelligent vehicle driver face and conscious recognition. Revue d'Intelligence Artificielle, 37(6): 1483-1492. https://doi.org/10.18280/ria.370612

[8] Alagappan, V.V., Vaishnav, H. (2023). Digital forensics face detection and recognition. International Journal for Research in Applied Science & Engineering Technology, 11(3): 535-541. https://doi.org/10.22214/ijraset.2023.49393

[9] Mamieva, D., Abdusalomov, A.B., Mukhiddinov, M., Whangbo, T.K. (2023). Improved face detection via learning small faces on hard images using deep learning. Sensors, 23(1): 502. https://doi.org/10.3390/s23010502

[10] Gupta, R., Gupta, A.K., Panwar, D., Jain, A., Chakraborty, P. (2023). Design and analysis of an expert system for detection and recognition of criminal faces. Journal of Electrical and Computer Engineering, 2023(1): 4284045. https://doi.org/10.1155/2023/4284045

[11] Majeed, A.W., Shaker, S.H., Saeid, A.A. (2024). Real-time face recognition and tracking using lightweight convolutional neural network. BIO Web of Conferences, 97: 00029. https://doi.org/10.1051/bioconf/20249700029

[12] Wong, Y.J., Lee, K.H., Tham, M.L., Kwan, B.H. (2023). Multi-camera face detection and recognition in unconstrained environments. In 2023 IEEE World AI IoT Congress (AIIoT), Seattle, WA, USA, pp. 0548-0553. https://doi.org/10.13140/RG.2.2.31681.12644

[13] Abbattista, G., Convertini, V.N., Gattulli, V., Sarcinella, L. (2020). CORDOBA system: Physical access management control with face detection and recognition. IOSR Journal of Computer Engineering, 22(1): 42-48. https://doi.org/10.9790/0661-2201024248

[14] Haq, M.U., Sethi, M.A.J., Ahmad, S., ELAffendi, M.A., Asim, M. (2024). Automatic player face detection and recognition in cricket games. IEEE Access, 12: 41219-41233. https://doi.org/10.1109/ACCESS.2024.3377564

[15] Mustafa, R., Min, Y., Zhu, D. (2014). Obscenity detection using Haar-like features and gentle AdaBoost classifier. The Scientific World Journal, 2014(1): 753860. https://doi.org/10.1155/2014/753860

[16] Najeeb, A., Sachan, A., Tomer, A., Prakash, A. (2023). Face mask detection using OpenCV. Advances in Science and Technology, 124: 53-59. https://doi.org/10.4028/p-2ffx83

[17] Huang, Z., Ou, C., Guo, Z., Ye, L., Li, J. (2024). Human-following strategy for orchard mobile robot based on the KCF-YOLO algorithm. Horticulturae, 10(4): 348. https://doi.org/10.3390/horticulturae10040348

[18] Zhao, F., Hui, K., Wang, T., Zhang, Z., Chen, Y. (2021). A KCF-based incremental target tracking method with constant update speed. IEEE Access, 9: 73544-73560. https://doi.org/10.1109/ACCESS.2021.3080308

[19] Zhou, Z., Hu, X., Li, Z., Jing, Z., Qu, C. (2022). A fusion algorithm of object detection and tracking for unmanned surface vehicles. Frontiers in Neurorobotics, 16: 808147. https://doi.org/10.3389/fnbot.2022.808147

[20] Gdeeb, R.T. (2023). A survey of face detection and recognition systems. Iraqi Journal of Intelligent Computing and Informatics, 2(1): 44-57. https://doi.org/10.52940/ijici.v2i1.32

[21] Ahmed, H.A., Al-Hayanni, M.N., Croock, M.S. (2024). Intelligent and secure real-time auto-stop car system using deep-learning models. International Journal of Electrical and Computer Engineering Systems, 15(1): 31-39. https://doi.org/10.32985/ijeces.15.1.4

[22] Ismail, W.N., Alsalamah, H.A., Hassan, M.M., Mohamed, E. (2023). AUTO-HAR: An adaptive human activity recognition framework using an automated CNN design. Heliyon, 9(2): e13636. https://doi.org/10.1016/j.heliyon.2023.e13636

[23] Kamagate, B.H., Kopoin, N.C., Koffi, D.D.A., Asseu, O.P. (2024). A robust Convolutional Neural Network model for fruit image classification. Ingénierie des Systèmes d'Information, 29(5): 1701-1710. https://doi.org/10.18280/isi.290504

[24] Alzubaidi, L., Zhang, J., Humaidi, A.J., Al Dujaili, A., Duan, Y., Al Shamma, O., et al. (2021). Review of deep learning: Concepts, CNN architectures, challenges, applications, and future directions. Journal of Big Data, 8(1): 53. https://doi.org/10.1186/s40537-021-00444-8

[25] Merzah, B.M., Croock, M.S., Rashid, A.N. (2024). Intelligent classifiers for football player performance using machine learning. International Journal of Electrical and Computer Engineering Systems, 15(2): 173-183. https://doi.org/10.32985/ijeces.15.2.6

[26] Peng, P., Zhao, X., Pan, X., Ye, W. (2018). Gas classification using deep convolutional neural networks. Sensors, 18(1): 157. https://doi.org/10.3390/s18010157

[27] Hssayeni, M.D., Croock, M.S., Salman, A.D., Al-

Khafaji, H.F., Yahya, Z.A., Ghoraani, B. (2020). Intracranial hemorrhage segmentation using a deep convolutional model. Data, 5(1): 14. https://doi.org/10.3390/data5010014
[28] McCullagh, P. (2023). Face detection by using Haar cascade classifier. Wasit Journal of Computer and Mathematics Science, 2(1): 1-5. https://doi.org/10.31185/wjcm.109
[29] Nongthombam, K., Sharma, D. (2021). Data analysis using Python. International Journal of Engineering Research and Technology, 10(7): 463-468. https://doi.org/ 10.17577/IJERTV10IS070241
[30] Adusumalli, H., Kalyani, D., Sri, R.K., Pratapteja, M., Rao, P.P. (2021). Face mask detection using OpenCV. In Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV), Tirunelveli, India, pp. 1304-1309. https://doi.org/10.1109/ICICV50876.2021.9388375
[31] Hameed, A.S., Hasan, T.M., Khaji, R. (2025). Real time classification of retail theft utilizing YOLO algorithm. Ingénierie des Systèmes d'Information, 30(6): 1517-1522. https://doi.org/10.18280/isi.300610
[32] Xie, Y., Wang, H., Guo, S. (2020). Research on MTCNN face recognition system in low computing power scenarios. Journal of Internet Technology, 21(5): 1463-1475.