

Comparison of Multi-Face Detection Performance on Images Using Haarcascade, Dlib, and RetinaFace

Sabarudin Saputra*, Adi Jufriansah, Muhammad Amirul Mu'min, Muhammad Akbar, Deni Luvi Jayanto, Ayu Hernita

Abstract—Multi-face detection presents a significant challenge in computer vision, especially in environments with limited hardware resources. This study compares the performance of three multi-face detection methods—Haarcascade, Dlib (HOG and CNN), and RetinaFace—using a subset of the WIDER FACE dataset in a CPU-only environment without GPU acceleration. The experiment was conducted in two stages using a total of 300 images from the WIDER FACE dataset, which reflect real-world variations such as pose, scale, illumination, expression, and occlusion. Performance evaluation was carried out using precision, recall, F1-score, accuracy, and processing time as metrics. The results show that RetinaFace consistently outperforms the other methods, achieving superior metrics in Recall (0.92), F1-score (0.93), and Accuracy (0.88) on Subset A, and leading across all metrics on Subset B. While Dlib-CNN demonstrates high detection performance, it suffers from very slow processing time. In contrast, Haarcascade delivers the fastest processing speed but performs poorly in terms of evaluation metrics. The experiments also reveal that RetinaFace is the most consistent and reliable method based on standard deviation values of precision (0.01), recall (0.11), F1-score (0.07), and accuracy (0.11). Overall, this study contributes valuable insights for selecting efficient face detection methods under constrained resource conditions.

Index Terms—Multi-Face Detection, Performance Comparison, Haarcascade, Dlib, RetinaFace

I. INTRODUCTION

FACE detection is an essential part of many applications, including automated attendance systems, security surveillance, and human-computer interaction [1], [2]. However, multi-face detection remains a challenging task due to variations in illumination, facial expressions, viewing angles, face sizes, and computational efficiency constraints [3], [4], [5]. Over the years, a wide range of face detection techniques has been developed, ranging from traditional feature-based approaches to more recent deep learning-based methods.

One of the earliest and most widely adopted methods is the Haar cascade classifier, introduced by Viola and Jones. This method uses a cascade-based classification strategy with Haar-like features to enable real-time face detection [2], [6]. On the other hand, Dlib, an open-source C++ machine learning library [7], provides tools for developing practical machine learning and data analysis applications [8], and includes two primary face detection methods: one

based on a Convolutional Neural Network (CNN) and another based on Histogram of Oriented Gradients (HOG) [9].

The HOG-based approach is known for its speed and computational efficiency; however, it tends to perform poorly in low-light conditions and under extreme viewing angles [9], [10]. In contrast, the CNN-based method requires greater computational resources but offers higher detection accuracy across diverse scenarios [11].

Another modern approach is RetinaFace, a deep learning-based face detection model capable of accurately localizing faces even under challenging conditions such as poor illumination, extreme poses, and occlusion [12]. Built on top of powerful backbone architectures like ResNet or MobileNet [13], [14], RetinaFace employs feature pyramid networks and context modules to detect faces and their associated landmarks at multiple scales and orientations [15], [16].

Numerous earlier studies have looked at how well different face detection algorithms perform in terms of performance and speed under different circumstances. Research conducted by Suwarno and Kevin [9] compares four face recognition algorithms—Dlib's CNN and HoG, and OpenCV's DNN and HAAR—based on accuracy and speed, using LFW and AT&T datasets. Conducted in Python, results show HoG as the fastest (0.011 s/image) but with lower accuracy (FRR 27.27%, FAR 0%), while DNN achieved the highest accuracy (FRR 11.69%, FAR 2.6%) but was slowest (0.119 s/image). Real-world image tests revealed that HAAR performs well under varied conditions, outperforming others in crowded and low-quality scenes.

RetinaFace, introduced by Deng et al. [15], is a single-shot, multi-level face localization framework that integrates face detection, 2D landmark localization, and 3D face reconstruction. Built on a feature pyramid network with deformable context modules and cascade regression, it employs ResNet-50 or ResNet-152 backbones and is trained on four NVIDIA Tesla P40 GPUs (24GB each). Using manually and semi-automatically annotated datasets (WIDER FACE, AFLW, FDDB), RetinaFace jointly regresses bounding boxes, five facial landmarks, and 1k 3D vertices. It achieves state-of-the-art performance (91.7% AP on WIDER FACE Hard) with efficient inference speed (22.3 ms per image).

Bajpai et al. [17] conducted a comparative analysis of two classical face detection methods: Gradient Vector Flow (GVF) using active contour models and silhouette-based motion segmentation. Tested on a Pentium Core 2 Duo 1.83 GHz system, the silhouette method achieved

Sabarudin Saputra, Muhammad Akbar, Deni Luvi Jayanto, and Ayu Hernita are with Department of Information Technology, Universitas Tadulako, Indonesia. Adi Jufriansah is with Universitas Gadjah Mada, Indonesia. Muhammad Amirul Mu'min is with Universitas Muhammadiyah Bima, Indonesia.
*Correspondence: sicoccinela@gmail.com.

faster processing (<1 second) and fully automatic segmentation, while GVF required 31 seconds and manual parameter tuning. GVF performed well on frontal faces in constrained settings (97.88% on A.R., 96.50% on ORL) but was highly sensitive to initialization parameters. In contrast, silhouettes were robust in dynamic scenes but relied on video input, limiting applicability to static image scenarios.

Edirisooriya et al. [18] conducted a comparative study of five face detection methods using 2,359 samples, including webcam videos and the WIDER-FACE dataset. Implemented on an Intel Core i5-4200U CPU (1.60 GHz, no GPU), SSD demonstrated the best performance, achieving 99.7% precision and 73.1% recall under challenging conditions such as low illumination (<5 lux), occlusion, varying face angles, and scales. However, SSD struggled with detecting very small faces due to sparse anchor scales. MMOD achieved comparable accuracy but was computationally slow (0.11 FPS), while the Viola-Jones method showed poor reliability, particularly under low-light and occluded conditions.

BlazeFace, developed by Bazarevsky et al. [19], is a lightweight face detection model optimized for real-time inference on mobile GPUs. Designed for mobile deployment, it achieves 200–1000+ FPS on flagship devices such as the iPhone XS (0.6 ms inference time, 98.61% AP) using TensorFlow Lite GPU. BlazeFace improves upon SSD by utilizing a compact MobileNetV1/V2-inspired backbone, an 8×8 GPU-friendly anchor scheme, and a tie-resolution strategy replacing non-maximum suppression. Although it shows a slight increase in regression error (10.4% IOD), it reduces temporal jitter by 40%. Its six keypoints support robust face orientation estimation for real-time AR and analysis.

Zhang et al. [20] proposed FaceBoxes, a lightweight face detection model optimized for real-time performance on CPU-only devices. It achieves 20 FPS on an Intel Xeon E5-2660v3 and 125 FPS on GPU for VGA-resolution images, with 96.0% mAP on the FDDB dataset. The architecture combines Rapidly Digested Convolutional Layers (RDCL) for speed and Multiple Scale Convolutional Layers (MSCL) for robust multi-scale detection. An anchor densification strategy enhances small-face recall by maintaining uniform anchor density. Trained on the WIDER FACE dataset with extensive augmentation, FaceBoxes achieves state-of-the-art results across AFW, PASCAL Face, and FDDB, while maintaining a compact 4.1MB model size.

A study by Majeed et al., [11] evaluates four face detectors—Haar cascade, LBP cascade, Dlib HOG, and Dlib CNN—using the FEI dataset and 12 real-world images captured via webcam. Dlib CNN achieved the highest detection accuracy (100%) but was the slowest (3.52s/image). Haar performed fastest (0.02s/image) but suffered in accuracy, failing in complex scenarios. LBP offered moderate performance, while HOG balanced speed (0.1s/image) and precision (98%). Experiments were run on an Intel Core i7-9750H CPU (2.6 GHz, 12 cores) with NVIDIA GeForce GTX 1660 Ti GPU. Results show Dlib CNN's robustness under challenging conditions but with high computational cost, while Haar and LBP are better

suited for real-time applications with simpler visual environments.

Although numerous studies have investigated multi-face detection in digital images using a wide range of algorithms—from classical feature-based methods to advanced deep learning architectures—comparative evaluations focusing specifically on Haarcascade, Dlib (HOG and CNN), and RetinaFace remain limited. Most existing works utilize large datasets and high-performance computing environments, such as GPU-accelerated platforms, to assess detection performance. However, there is a lack of studies that examine these three representative methods under constrained computational environments and using smaller subsets of publicly available datasets.

This study aims to compare the performance of Haarcascade, Dlib (HOG and CNN), and RetinaFace in detecting multiple faces in images drawn from a smaller subset of the WIDER FACE dataset. The focus is on evaluating both performance detection and computational efficiency in a realistic, CPU-only development environment with limited hardware resources. This study not only directly compares these widely used detection methods, but also emphasizes their performance under low resource constraints, reflecting the needs of many practical applications. By analyzing the trade-off between speed and performance in this context, this study provides insights for developers and researchers working in environments where high-end GPUs are not available.

II. MATERIAL AND METHODS

A. Types and Objects of Research

This study employs a comparative quantitative approach to evaluate and compare the performance and processing speed of three face detection algorithms—Haarcascade, Dlib (HOG and CNN), and RetinaFace—specifically in multi-face detection scenarios. The experimental evaluation is conducted using a subset of the WIDER FACE image dataset, which contains images featuring multiple faces under diverse and challenging conditions.

WIDER FACE is a widely recognized benchmark dataset for face detection, designed to capture the complexity and diversity of real-world environments [21]. It comprises 32,203 images and 393,703 annotated faces, exhibiting significant variations in facial scale, pose, occlusion, illumination, and expression. The dataset is divided into three subsets—training (train), validation (val), and testing (test)—with each image meticulously annotated using bounding boxes that indicate the location of all visible faces.

B. Device Specifications

The experiments in this study were conducted on a device with 11th Gen Intel(R) Core(TM) i3-1115G4 @ 3.00GHz, equipped with 8 GB of RAM and running Windows 11 Home (version 24H2, build 26100.4652). No dedicated GPU or CUDA acceleration was utilized throughout the evaluation process.

While RetinaFace and dlib CNN are inherently

designed to take advantage of GPU acceleration (e.g., via CUDA) for faster inference [8], [15], this study deliberately executed all face detection algorithms—including RetinaFace, Dlib-CNN, Haarcascade, and Dlib-HOG—on the same CPU-only environment. This decision was made to ensure a fair and consistent comparison across all methods, especially for deployment scenarios where GPU resources are not available (e.g., edge devices or entry-level systems).

By maintaining uniform hardware conditions, the evaluation focuses on algorithmic performance rather than hardware optimization. The choice of low-to-mid hardware specification also reflects realistic usage contexts, particularly for lightweight, real-world face detection applications in non-specialized environments. In this study, the programming language used was Python 3 and the experimental environment was carried out on the Anaconda Jupyter Notebook.

C. Research Steps

This research was conducted through several systematic steps like collecting images, images annotation, images preprocessing, multi-face detection, and performance evaluation, as shown in Figure 1.

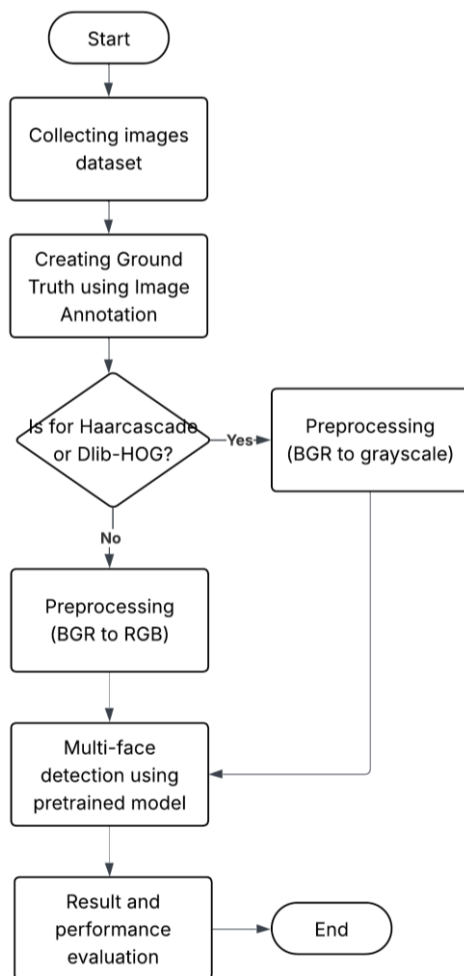


Fig. 1. Face detection steps.

Figure 1 shows the steps of multi-face detection. These steps are explained in more detail as follows:

1) Collecting images

The process of image collection in this study was carried out using a purposive sampling strategy from the publicly available WIDER FACE dataset, which is a comprehensive benchmark designed to simulate diverse real-world face detection challenges [21]. The image collection process was conducted in two phases:

Phase 1 – Preliminary Sampling from WIDER_test: In the initial phase, 50 images were randomly selected from the WIDER_test subset. These first 50 images were named Subset A. The selection aimed to provide a representative sample containing multiple faces with varied conditions such as lighting, occlusion, face orientation, and scale. This phase served as an exploratory analysis to compare the baseline performance of the three selected algorithms (Haarcascade, Dlib, and RetinaFace) in terms of detection accuracy and execution time.

Phase 2 – Extended Sampling from WIDER_val: Recognizing the limitation of the small sample size in the first phase, an extended sampling was conducted in the second phase. A total of 250 images were selected randomly from the WIDER_val subset. These second 250 images were named Subset B. This larger set enabled a more statistically robust and comprehensive evaluation across a broader range of challenging scenarios. These 250 images were then divided into five folders of 50 images each and tested repeatedly across all folders. This was done to assess the consistency and reliability of the pretrained multi-face detection model used.

Throughout both phases, the selection ensured that each image contained multiple faces (multi-face scenarios) [4], to better simulate real-world detection challenges and to support the comparative evaluation of multi-face detection performance across different algorithms [21].

2) Images annotation

Image annotation is a crucial step in face detection research [22], as it provides the baseline data needed to evaluate the accuracy of detection algorithms. In this study, annotation refers to the process of defining a bounding box around each visible face in an image [21], which serves as a reference for comparing detection results from various algorithms. The annotation process in this study is divided into two approaches: the first is manual annotation for Subset A and the second is using the built-in annotation of the WIDER FACE dataset for Subset B. Manual annotation of Subset A was performed because the WIDER FACE dataset does not provide annotation for the images in the WIDER_test subset. The manual annotation process was carried out by 2 independent annotators to produce two sets of annotation files. To validate the consistency and agreement between the two sets of annotations, a quantitative validation step was applied using the Intersection over Union (IoU) metric [21], [23].

3) Images preprocessing

In the preprocessing step, color image preprocessing is applied selectively based on the specific requirements of each face detection algorithm. Haarcascade and Dlib-HOG require grayscale input images because these traditional methods rely on intensity-based features such as edges, textures, and gradient orientation, which are effectively captured in single-channel images [2], [9]. The conversion is not optional, but rather a prerequisite for these algorithms to function correctly.

In contrast, RetinaFace and Dlib-CNN operate directly on RGB images, leveraging multi-channel information for deep feature extraction. These models are based on convolutional neural networks trained on color image datasets, where color cues can enhance facial structure recognition, especially under varying illumination and occlusion conditions [8], [15]. Thus, the preprocessing pipeline was tailored to align with the native input requirements of each algorithm to ensure fair and optimal detection performance, avoiding unnecessary transformations that could distort the expected input data. In RetinaFace and dlib CNN, preprocessing was not performed in this experiment directly but used the built-in preprocessing of the models of both methods.

4) Multi-face detection

The next step involves putting three face identification techniques into practice: RetinaFace, which uses a pre-trained model based on ResNet50 [13], [14], [15]; Dlib, which uses the HOG + SVM and CNN approaches [9]; and Haarcascade, which uses the CascadeClassifier module from OpenCV [2], [6]. Since this study focuses on performance benchmarking of pre-trained face detection models, no model training was conducted. Therefore, traditional cross-validation techniques were not applicable. Instead, a fixed test set was used to ensure consistency across all methods during evaluation. This evaluation-only approach aligns with best practices in comparative detection studies.

5) Performance evaluation of methods

Quantitative evaluation metrics, specifically Precision (2), Recall (3), F1-Score (4), and accuracy (5) were used to measure the accuracy of face detection in order to assess detection performance [24]. Equation (1) illustrates how the TP (True Positive), FP (False Positive), and FN (False Negative) variables from the IoU (Intersection over Union) computation are used to compute precision, recall, and F1-Score. The IoU concepts is shown in Figure 2.

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}} \quad (1)$$

$$\text{Precision} = \frac{TP}{TP+FP} \quad (2)$$

$$\text{Recal} = \frac{TP}{TP+FN} \quad (3)$$

$$F1 - score = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (4)$$

$$\text{Accuracy} = \frac{TP}{TP+FP+FN} \quad (5)$$

Where, IoU is an evaluation metric used to measure how well the face detection results compare to the ground truth bounding box coordinates of the image annotation results, TP indicates the correct face detection results ($IoU \geq \text{threshold}$), FP indicates the incorrect face detection results ($IoU < \text{threshold}$), and FN indicates the faces that were not successfully detected.

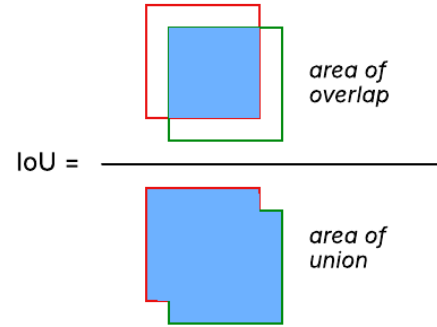


Fig. 2. Concept of Intersection over Union (IoU).

Figure 2 illustrates the fundamental concept of IoU, a widely adopted metric for evaluating the accuracy of object detection models, including face detection systems. IoU is calculated by dividing the area of overlap between the predicted bounding box (depicted in red) and the ground truth bounding box (depicted in green) by the area of their union. The blue region in the upper part of the figure represents the area where both boxes intersect, while the lower part of the figure shows the total area covered by the union of the two boxes, excluding redundant overlap.

The IoU threshold used to determine True Positive (TP) detections was set to greater than 0.5 ($IoU \geq 0.5$), in line with standard object detection benchmarks [21], [23]. This threshold ensures that only sufficiently overlapping bounding boxes are considered valid detections. To address potential overfitting and avoid multiple detections of the same face due to overlapping bounding boxes from different face, this study employs a greedy matching algorithm based on the IoU [15], [21]. The algorithm iteratively selects the highest IoU match for each predicted box and ground truth pair, ensuring one-to-one correspondence and preventing double-counting. This matching strategy enhances the reliability of performance metrics by reducing noise caused by redundant or ambiguous detections.

Furthermore, the time duration spent on the face detection process is recorded using the `time.time()` function to compute the detection speed performance. To evaluate the performance of each face identification model, the results are analyzed

and displayed as comparison tables and visual representations.

D. Performance Comparison

In this step, the performance of the three face detection algorithms—Haarcascade, Dlib (HOG and CNN), and RetinaFace—is systematically compared based on two performance evaluation metrics and processing time. To assess detection performance, the results of each algorithm are compared with the ground truth annotations from the WIDER FACE dataset. The evaluation is based on the number of True Positives (TP), False Positives (FP), and False Negatives (FN) identified by each method. From these values, precision, recall, F1-score and accuracy are calculated to quantitatively measure the correctness and completeness of each detector. The IoU metric with a defined threshold is used to determine valid detections.

Each algorithm is evaluated for its computational efficiency by recording the total processing time required for face detection across all images. The duration is measured using Python's time module, capturing the time taken solely during the detection phase. All evaluations are conducted under consistent hardware and implementation settings to ensure a fair comparison.

The resulting metrics, such as precision, recall, accuracy, and F1 score, are visualized using bar charts to facilitate comparative analysis. Additionally, summary statistics such as the mean and standard deviation are used to illustrate the consistency and reliability of each algorithm's performance.

TABLE I
EXAMPLES OF MULTI-FACE IMAGES WITH VARIOUS
CONDITIONS

Face Conditions	Images
Pose	
Scale	
Occlusion	
Expression	
Makeup	
Illumination	

III. RESULTS AND DISCUSSIONS

A. Images and Images Annotation

The images used in this study are Subset A and Subset B which contain several faces taking into account variations in visual conditions as explained in the methods section. Examples of images containing multiple faces with various conditions are shown in Table 1.

Table 1 presents a set of example multi-face images that represent a wide range of real-world facial conditions typically encountered in face detection systems. These conditions include pose, scale, occlusion, facial expression, makeup, and illumination. The purpose of this table is to highlight the complexity and challenges that face detection algorithms must address.

In the pose category, the images contain faces with non-frontal orientations. In real-world scenarios, human faces rarely align perfectly with the camera; instead, they appear at various angles such as sideways, tilted, or downward. The scale category illustrates faces of highly varied sizes within a single image, ranging from large foreground faces to small background faces. Detecting small faces is a long-standing challenge in computer vision. The occlusion category depicts scenarios where parts of the face are obscured by external objects such as hands, helmets, or masks. Partial occlusion can severely hinder detection, as key facial features may be missing or deformed. Facial expressions introduce additional challenges for detection systems. Expressions such as laughing, anger, or surprise significantly alter facial geometry and texture, making detection more difficult. The makeup category includes faces with heavy cosmetic alterations or artistic designs such as face paint and stage props. These modifications can significantly alter facial texture and color, affecting detection accuracy. Finally, lighting is a crucial factor affecting image quality and detection performance. Extreme lighting conditions such as overexposure, underexposure, or high contrast can blur or distort facial features.

Overall, the inclusion of these varied facial conditions in Table 1 is not only illustrative but serves as an essential component of the system validation process. Data diversity is a key factor in assessing the generalization ability and robustness of face detection methods. Table 1 shows various types of facial variations in images that represent real challenges in the multi-face detection process, because they can cause changes in the visual structure of the target facial area. These variations are used as test parameters to evaluate the robustness and accuracy of the detection method.

The manual annotation process in this study was conducted using a custom-built graphical user interface (GUI) implemented with Python and OpenCV. This annotation tool was designed to enable the annotators to manually draw bounding boxes around each visible face in a given image. This process was performed on only Subset A. The annotation workflow began by reading all image files from Subset A. Each image was displayed individually in an OpenCV window where annotators could draw bounding boxes using mouse input. The process and example of the results of facial annotation on

images are shown in Figure 3.



Fig. 3. Images annotation process.

Figure 3 illustrates a structured face annotation workflow utilized in the evaluation of face detection algorithms, particularly in scenarios involving multiple faces. The process begins with the annotation process, where annotators manually identify and mark all visible human faces in an image using bounding boxes. These bounding boxes are rectangular and drawn tightly around each detected face. Each bounding box is defined by four primary coordinates: x (horizontal position), y (vertical position), w (width), and h (height), which together localize the face in the two-dimensional image space. Finally, the facial coordinate data generated during the annotation process is saved as a .json file. This file contains the list of all annotated bounding boxes for each image, encoded in the form of $[x, y, w, h]$ for every detected face. These ground truth annotations are essential for calculating performance metrics such as precision, recall, F1 score, and accuracy based on IoU value.

The annotation process was repeated independently by

two annotators to generate two separate sets of ground truth data. This dual-annotator strategy was essential to measure annotation consistency and to validate inter-rater agreement using the Intersection over Union (IoU) metric. Examples of annotation results from two annotators can be seen in Figure 4.

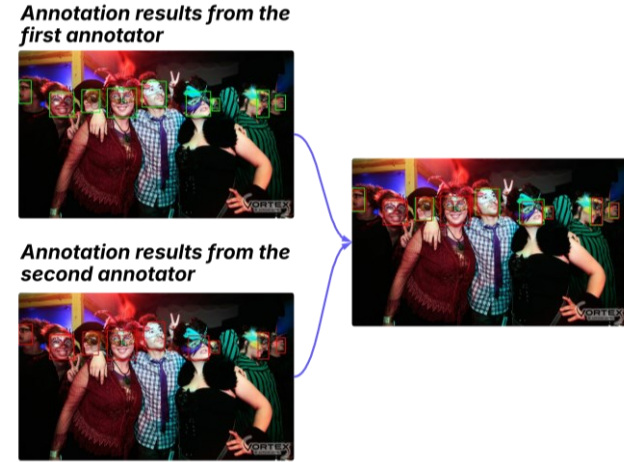


Fig. 4. Example of a visual comparison of facial bounding boxes annotated by two independent annotators.

Figure 4 presents an example of a visual comparison of facial bounding boxes annotated by two independent annotators on the same image. In the top-left image, the bounding boxes drawn by the first annotator are shown in green. Similarly, the bottom-left image illustrates the bounding boxes created by the second annotator, shown in red. The image on the right overlays both annotation sets to provide a clear visual assessment of agreement. In this combined image, the green boxes represent the first annotator, and the red boxes represent the second annotator. This comparison is crucial for validating the reliability of the ground truth data. To measure the degree of agreement, the IoU metric is calculated for each corresponding pair of bounding boxes. The average IoU value for each image and the overall IoU value are then calculated. The results of the average IoU calculation are shown in Figure 5.

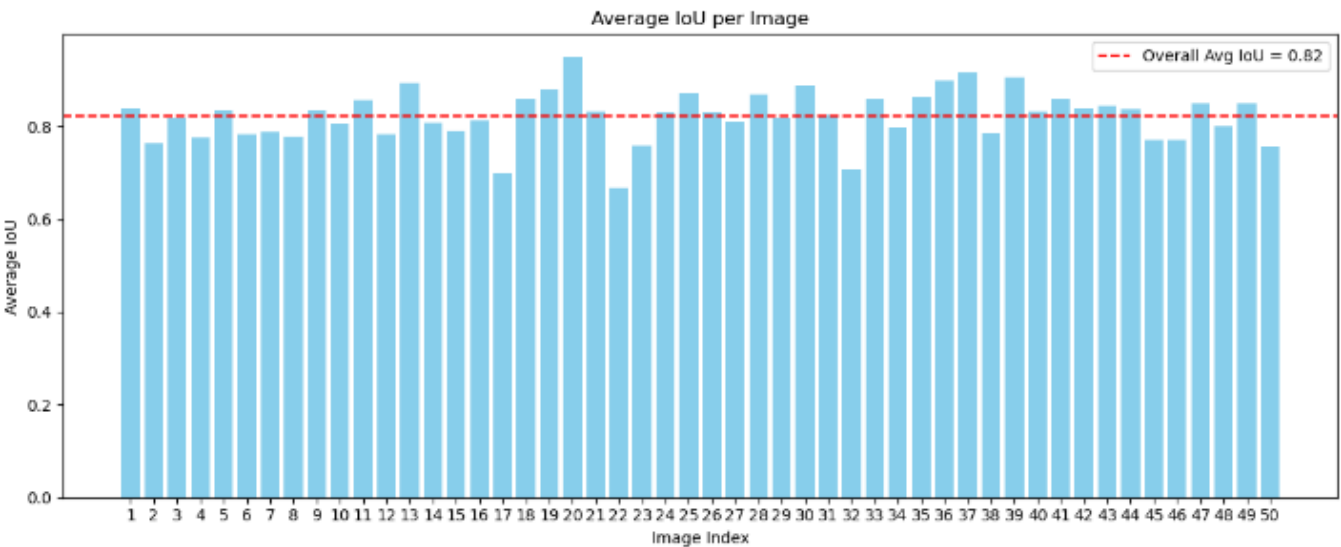


Fig. 5. The results of the average IoU calculation.



Fig. 6. The color space conversion results.

Figure 5 illustrates the inter-annotator agreement measured using the IoU metric across 50 annotated images. Each bar in the chart represents the average IoU value for a single image, comparing the bounding boxes provided by two independent annotators. The x-axis shows the image index (from 1 to 50), while the y-axis indicates the corresponding average IoU value for each image. The high average IoU score (0.82) confirms the validity of the ground truth annotations used in this study. This level of agreement aligns with commonly accepted standards for object detection benchmarks [21], [23], where an IoU of 0.5 or higher is typically considered a valid match. As such, the annotation data can be confidently used as a reference for evaluating the performance of face detection algorithms in subsequent steps of this research.

B. Images Preprocessing

Color space conversion represents a critical preprocessing step in face detection tasks, as different detection methods operate under distinct image format requirements. In this study, color conversion was applied using the OpenCV library to match the input expectations of each detection algorithm, thereby ensuring both compatibility and performance optimization. The color space conversion results are shown in Figure 6.

Based on Figure 6, it is important to note that although digital images are stored in RGB (Red-Green-Blue) color format, OpenCV reads images by default in BGR (Blue-Green-Red) format when using the `cv2.imread()` function. This is a result of OpenCV's internal conventions, and if not handled correctly, can lead to incorrect color interpretation in the detection method used. For Haarcascade and Dlib HOG method, the input images were first converted from BGR to grayscale using OpenCV's `cv2.cvtColor()` function with the flag `cv2.COLOR_BGR2GRAY`. This is particularly appropriate for Haar-like features, which are designed to detect edges and intensity contrasts rather than color (Viola & Jones, 2001). Similarly, the Histogram of Oriented Gradients (HOG) descriptor in the Dlib HOG-based detector relies solely on shape and gradient information, making grayscale images sufficient and more computationally efficient. Meanwhile, Dlib CNN and RetinaFace use images in RGB format. To match the input format expected by the frameworks of both methods, which use RGB colors, images read using OpenCV

(`cv2.imread()`) are converted from BGR to RGB using `cv2.cvtColor()` with the flag `cv2.COLOR_BGR2RGB`.

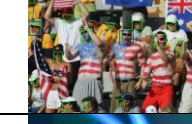
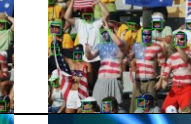
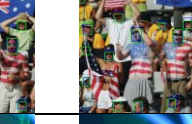
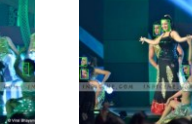
C. Multi-face Detection Results and Evaluation

This section explains the process and results of multi-face detection using the haarcascade, dlib (HOG and CNN), and RetinaFace methods on two image subsets as described in the materials and methods section. The process of multi-face detection on each image subset are described as follows.

Determining the storage directory for the input and output photos of the detection results is the first step in the process. The ground truth coordinates of the face in each image are put into the system as a `ground_truth.json` file to aid in the evaluation process. The model used is a pre-trained haarcascade that can be found in the OpenCV library as a `haarcascade_frontalface_default.xml` file. In grayscale images, this algorithm finds geometric structures and texture patterns that are typical of faces. Therefore, before the detection method is executed, each input image is first transformed from RGB to grayscale format. In order to assist texture analysis, the Dlib HOG (Histogram of Oriented Gradients) face identification procedure also starts by reading each image from the input folder and converting it to grayscale format. After scanning the grayscale image with the `hog_detector` of face detector, the HOG algorithm extracts directional gradient features in tiny cells throughout the image. The Support Vector Machine (SVM) classifier uses these features, which mimic the distinctive edge and texture patterns of human faces, to identify if a region contains a face.

In contrast to the detection process using Dlib HOG, the face detection process using Dlib CNN involves loading the `mmod_human_face_detector.dat` model and then using the `cnn_detector` method to apply it directly to the color image in order to detect faces based on the CNN architecture's bounding box results. The last step in the RetinaFace face detection process is to use the `RetinaFace.detect_faces()` method, which immediately identifies faces in the input image and provides the facial area coordinates (`facial_area`) as a bounding box. After that, the bounding box from each method's detection results is transformed into coordinates (x, y, w, and h) and saved. The results of multi-face detection in images based on various facial conditions using haarcascade, dlib (HOG and CNN), and RetinaFace are shown in Table 1.

TABLE II
EXAMPLE OF MULTI-FACE DETECTION RESULTS USING HAARCASCADE, DLIB, AND RETINAFACE ON SUBSET A.

Face Conditions	Original Image	Haarcascade	Dlib-HOG	Dlib-CNN	RetinaFace
Pose					
Scale					
Occlusion					
Expression					
Makeup					
Illumination					

The effectiveness of multi-face detection models varies significantly depending on facial conditions such as pose, scale, occlusion, expression, makeup, and illumination. Table 2 presents qualitative comparisons of multi-face detection using Haarcascade, Dlib-HOG, Dlib-CNN, and RetinaFace based on images with diverse facial challenges. Each method demonstrates unique strengths and limitations, influenced by their underlying architecture and detection strategies.

Based on Table 2, the results of multi-face detection with varying conditions show that under pose variation (i.e., rotated or non-frontal faces), Haarcascade exhibits poor performance, often failing to detect faces not directly facing the camera. This limitation is due to its reliance on frontal face features using rigid Haar-like filters and cascaded classifiers, which are highly sensitive to face alignment [2], [25]. In contrast, Dlib-HOG, which uses histogram of oriented gradients and linear SVM classifiers, performs better but still misses faces with large angular deviations. Dlib-CNN significantly outperforms the previous two, as its deep convolutional layers learn robust spatial hierarchies that can generalize to rotated faces. These results are in accordance with research by Suwarno and Kevin [9]. RetinaFace achieves the highest accuracy, detecting all faces regardless of pose, due to its use of deep residual networks and multi-task learning that jointly predicts facial landmarks and bounding boxes. This is in line with research by Deng et al. [15].

The detection of faces at varying scales is another critical challenge. Haarcascade struggles with small or distant faces because its fixed-size sliding window cannot

adaptively adjust to different scales. Dlib-HOG offers marginal improvement through scale-invariant features but is constrained by its handcrafted nature. Dlib-CNN benefits from the receptive field and depth of convolutional layers, allowing it to detect both large and small faces more effectively [9]. RetinaFace again demonstrates superior adaptability, employing feature pyramid networks (FPN) to detect multi-scale faces efficiently. This advantage is supported by findings in Zhang et al. [26], which emphasize the importance of multi-scale feature fusion in robust face detection.

When faces are partially occluded by objects such as helmets or masks, Haarcascade and Dlib-HOG both fail to detect several faces due to their reliance on visible and complete facial structures. Dlib-CNN handles occlusion more robustly by learning discriminative features even from incomplete visual information. RetinaFace is the most resilient to occlusion, likely due to its joint detection of facial landmarks, which allows the network to localize faces even when only partial regions are visible. This capability aligns with observations in [27], showing that landmark-guided detectors significantly enhance occluded face recognition.

Detection accuracy under various expressions (smiling, open mouth, etc.) also reveals the robustness of each model. Haarcascade performs inconsistently, as expressive changes distort the static features it depends on. Dlib-HOG shows moderate resilience but struggles with exaggerated expressions. Dlib-CNN maintains stable detection by virtue of deeper features learned across diverse datasets. RetinaFace surpasses the others, leveraging facial

alignment modules that reduce sensitivity to expression variation. Research by Yang et al. [21] supports this, highlighting that deep CNNs trained with large-scale, in-the-wild facial expression datasets generalize better across diverse expressions.

Heavy makeup and cosmetic alterations present another form of challenge. Haarcascade and Dlib-HOG show noticeable performance drops, missing faces with altered contrast or artificial textures. Dlib-CNN retains better accuracy as it focuses more on abstract patterns rather than raw pixel values. RetinaFace is largely unaffected, as its deep learning framework is trained on extensive datasets with high facial diversity, allowing it to generalize well to makeup and accessories. This strength echoes findings in [28], which emphasize the role of training diversity in improving real-world face detection performance.

Under poor or non-uniform lighting, Haarcascade's performance deteriorates drastically because it is sensitive to pixel intensity variations. Dlib-HOG is somewhat more resilient due to its gradient-based features, but still suffers under extreme illumination changes. Dlib-CNN exhibits significant robustness by learning illumination-invariant features through deeper layers. RetinaFace, however, stands out by performing consistently well under low light or colored lighting conditions, supported by its use of photometric normalization layers and enhanced contextual modeling. This outcome is validated by [29], who found that deep models with context-aware learning outperform traditional methods in adverse lighting scenarios.

D. Performance Comparison of Methods

Performance comparison was conducted using standard evaluation metrics: precision, recall, F1-score, and accuracy. In addition, performance evaluation is also carried out by looking at the total duration time used to perform multi-face detection on all images. These metrics provide a comprehensive understanding of each model's ability to correctly identify multiple faces under diverse real-world conditions. The evaluation was carried out on two distinct subsets of images, namely Subset A and Subset B, which were constructed from the publicly available WIDER FACE dataset using a purposive sampling strategy to ensure robustness in performance assessment. Subset A serves as the baseline for comparative evaluation. The evaluation metric values of each method tested on Subset A are shown in Table 3.

Table 3 presents the evaluation metrics on Subset A which provides a detailed comparative analysis of the used face detection methods. The Haarcascade method achieves a precision of 0.75 and a recall of 0.50, resulting in an F1-score of 0.60 and an overall accuracy of 0.43. This indicates that while the model is moderately precise in

detecting true faces, it fails to detect many actual faces (high FN = 157), likely due to its sensitivity to facial orientation, scale, and occlusion. The relatively fast processing time of 12.05 seconds for the entire subset reflects Haarcascade's computational efficiency, which is expected given its lightweight, rule-based design.

The Dlib-HOG method improves significantly in terms of precision, reaching 0.96, but its recall remains low at 0.56, yielding an F1-score of 0.70 and an accuracy of 0.54. While Dlib-HOG is highly precise (low FP = 7), it still misses a considerable number of faces (FN = 141), especially under conditions of occlusion or low resolution, as supported by the qualitative results in Table 2. The detection time of 20.07 seconds remains relatively efficient, making this method a reasonable option when real-time or resource-limited scenarios are considered. The lower recall can be attributed to the method's reliance on rigid hand-crafted HOG features, which are less effective in capturing faces with extreme variability [9].

In contrast, Dlib-CNN offers a more balanced trade-off, achieving 0.92 in both precision and F1-score, with a higher recall of 0.71 and accuracy of 0.80. This suggests that the deep convolutional architecture is more capable of generalizing across challenging facial variations, including occlusion and pose. However, this performance comes at a steep computational cost: 4514.73 seconds, or roughly 75 minutes, for the subset. This makes Dlib-CNN unsuitable for time-sensitive applications, despite its solid detection performance. The computational burden arises from the fact that Dlib-CNN was not optimized for inference speed and performs forward passes serially on the CPU, which is consistent with findings in [26].

RetinaFace demonstrates superior overall performance, achieving the highest precision (0.95), recall (0.92), F1-score (0.93), and accuracy (0.88). The number of false positives and false negatives is remarkably low (FP = 15, FN = 24), indicating excellent robustness across diverse facial conditions. Although its duration of 132.25 seconds is significantly higher than Haarcascade and Dlib-HOG, it is far more computationally efficient than Dlib-CNN, especially considering its high accuracy. RetinaFace's effectiveness stems from its deep backbone (ResNet + Feature Pyramid Network), multi-task learning for landmark localization, and extensive training on diverse facial datasets. These architectural choices enable it to detect faces accurately even under challenging conditions such as occlusion, extreme expressions, and poor lighting, as reported in [15].

Next, to validate the robustness and generalization of the evaluated face detection methods, further experiments were conducted on Subset B. The evaluation metric values of each method tested on Subset B are shown in Table 4.

TABLE III
EVALUATION METRICS OF PRECISION, RECALL, F1 SCORE, ACCURACY, AND DURATION ON SUBSET A

Methods	TP	FP	FN	Precision	Recall	F1 Score	Accuracy	Duration (s)
Haarcascade	160	54	157	0.75	0.50	0.60	0.43	12.05
Dlib-HOG	176	7	141	0.96	0.56	0.70	0.54	20.07
Dlib-CNN	226	20	91	0.92	0.71	0.80	0.67	4514.73
RetinaFace	293	15	24	0.95	0.92	0.93	0.88	132.25

TABLE IV
EVALUATION METRICS OF PRECISION, RECALL, F1 SCORE, ACCURACY, AND DURATION ON SUBSET B

Folder	Methods	TP	FP	FN	Precision	Recall	F1 Score	Accuracy	Duration (s)
First	Haarcascade	159	50	320	0.76	0.33	0.46	0.30	42.01
	Dlib-HOG	166	8	313	0.95	0.35	0.50	0.34	58.06
	Dlib-CNN	221	12	258	0.95	0.46	0.62	0.45	3806.39
	RetinaFace	374	10	105	0.97	0.78	0.85	0.76	302.71
Second	Haarcascade	228	69	381	0.77	0.37	0.50	0.33	48.96
	Dlib-HOG	224	5	385	0.98	0.37	0.53	0.36	63.61
	Dlib-CNN	307	5	302	0.98	0.50	0.67	0.50	4391.55
	RetinaFace	482	5	127	0.99	0.79	0.88	0.78	119.18
Third	Haarcascade	199	45	281	0.82	0.41	0.55	0.38	38.08
	Dlib-HOG	210	6	270	0.97	0.44	0.60	0.43	52.31
	Dlib-CNN	294	7	186	0.98	0.61	0.75	0.60	7642.93
	RetinaFace	455	6	25	0.99	0.95	0.97	0.94	122.22
Fourth	Haarcascade	279	45	257	0.86	0.52	0.65	0.48	42.13
	Dlib-HOG	271	3	265	0.99	0.51	0.67	0.50	57.91
	Dlib-CNN	333	6	203	0.98	0.62	0.76	0.61	6366.24
	RetinaFace	464	4	72	0.99	0.87	0.92	0.86	115.14
Fifth	Haarcascade	128	45	541	0.74	0.19	0.30	0.18	43.73
	Dlib-HOG	117	3	552	0.97	0.17	0.30	0.17	56.60
	Dlib-CNN	204	6	465	0.97	0.30	0.46	0.30	5207.41
	RetinaFace	439	6	230	0.99	0.66	0.79	0.65	131.80

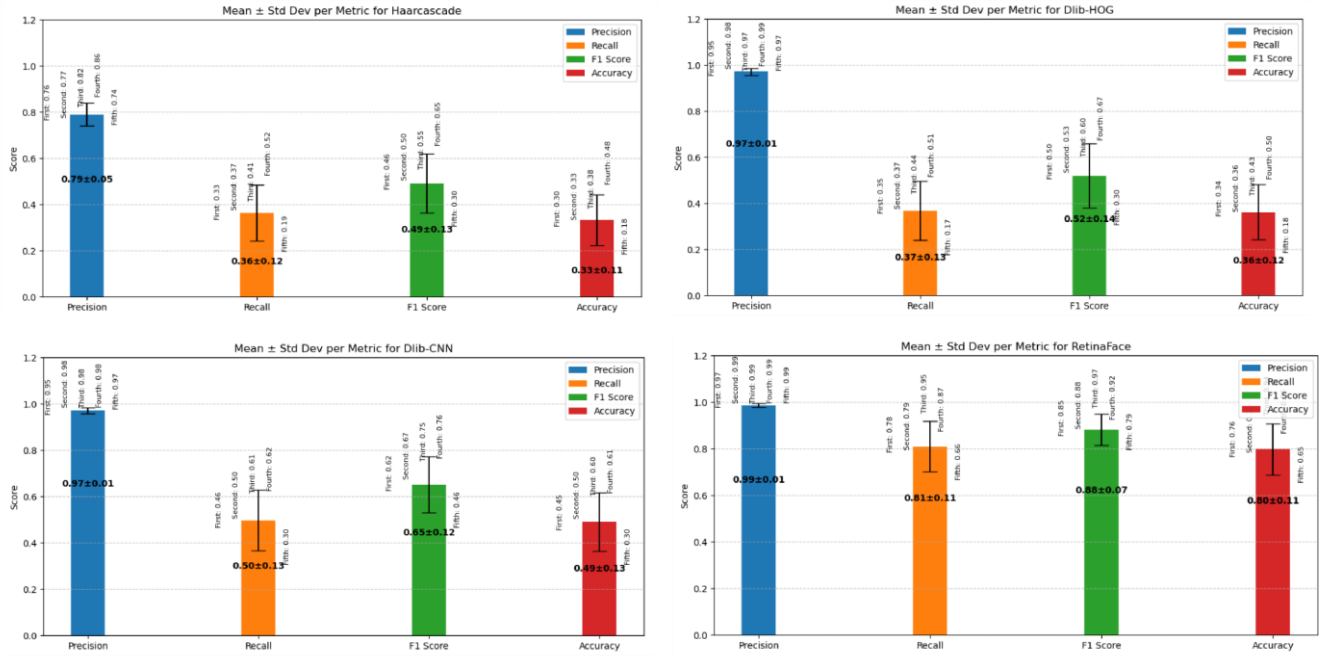


Fig. 6. The graph of the average and standard deviation value of the evaluation metrics for each method based on Subset B.

The results from Subset B confirm the trends observed in the baseline (Subset A): while Haarcascade continues to show fairly good precision, it exhibits poor recall and overall accuracy. This performance profile reinforces the discussion in the baseline, where Haarcascade was characterized with limited robustness, particularly in detecting faces under varying orientations and occlusion. The inability of the method to generalize well in Subset B validates prior conclusions about its rule-based nature and sensitivity to unconstrained imaging conditions.

Dlib-HOG maintains high precision across both subsets, consistent with its strong control over false positives. However, its recall remains notably limited in Subset B, which is aligned with the baseline discussion attributing its performance constraints to rigid, hand-crafted feature representations. This consistent drop in recall and overall detection coverage further substantiates the observation that while Dlib-HOG can confidently

detect certain facial patterns, it lacks the adaptability to detect more varied or challenging instances.

Dlib-CNN demonstrates comparable performance trends in Subset B to those in the baseline: high precision is maintained, and recall remains moderately strong, although with slight degradation. These results are consistent with the earlier characterization of Dlib-CNN as a method offering a balanced trade-off between detection quality and computational cost. The results on Subset B suggest that Dlib-CNN retains reasonable generalization capacity, although it shows some sensitivity to increased scene diversity, likely due to its serial inference and lack of contextual learning mechanisms.

RetinaFace continues to demonstrate consistently strong performance across all evaluation metrics in Subset B, reaffirming the conclusions drawn from the baseline. Its ability to maintain high recall and accuracy in more varied conditions supports the assertion that the method

generalizes well and is highly robust. The stable detection quality across both subsets reflects the advantages of RetinaFace's deep feature representation, multi-scale learning, and comprehensive training regime. These findings align with the architectural and empirical strengths highlighted in baseline experiments.

In addition to comparing the evaluation metrics of the experiments on Subset A and Subset B, it is also necessary to calculate the average value and standard deviation of the evaluation metrics values from Subset B. This is done to see the most consistent and reliable method for multi-face detection in digital images. The results of calculating the average and standard deviation of the evaluation metrics values on Subset B are shown in Figure 6.

Figure 6 shows that the comparative analysis of face detection methods using mean and standard deviation of four evaluation metrics demonstrates significant differences in both performance and consistency across Haarcascade, Dlib-HOG, Dlib-CNN, and RetinaFace. Among all methods, RetinaFace consistently achieves the highest overall performance, with a mean Precision of 0.99 ± 0.01 , Recall of 0.81 ± 0.11 , F1 Score of 0.88 ± 0.07 , and Accuracy of 0.80 ± 0.11 . These results not only indicate high effectiveness in correctly detecting multiple faces but also show a relatively stable performance across five test subsets, as reflected by the low to moderate standard deviations. This suggests that RetinaFace is a reliable and robust detector under varying image conditions such as pose, lighting, and occlusion.

In contrast, Haarcascade performs poorly, exhibiting the lowest mean Accuracy (0.33 ± 0.11) and F1 Score (0.49 ± 0.13), with standard deviations that signal poor consistency. Although simple and lightweight, its reliance on hand-crafted features and lack of learning capability make it inadequate for complex detection tasks. Dlib-HOG, while showing high Precision (0.97 ± 0.01), suffers from a notably low Recall (0.37 ± 0.13) and F1 Score (0.52 ± 0.14), indicating that it often misses faces, especially under challenging conditions. Dlib-CNN demonstrates a better balance, with F1 Score (0.65 ± 0.12) and Accuracy (0.49 ± 0.13) slightly outperforming Dlib-HOG, though its standard deviations suggest moderate variability and sensitivity to dataset diversity.

A pretrained model or face detection method's consistency and reliability can be evaluated using the standard deviation value as a benchmark. Xingkun Xu et al.'s research indicates that consistency and reliability are positively correlated with a smaller standard deviation value. According to the discussion, RetinaFace is the most reliable and consistent method for identifying multiple faces in digital images in this study.

Furthermore, to contextualize the current study and highlight its relevance, it is important to examine prior research efforts that evaluated various face detection methods across different computational setups and datasets. A summary of the comparison of the results of this study with previous studies is shown in Table 5.

TABLE V
A SUMMARY OF THE COMPARISON OF THE RESULTS OF THIS STUDY WITH PREVIOUS STUDIES

Study	Methods	Dataset	Hardware & Setup	Key Findings	Strengths	Limitations
Suwarno & Kevin [9]	Dlib (CNN, HOG), OpenCV (HAAR, DNN)	LFW, AT&T	Python, Unspecified CPU	HOG fastest (0.011s), DNN most accurate (FRR 11.69%)	Practical insight on speed vs accuracy	No RetinaFace, small dataset
Deng et al. [15]	RetinaFace	WIDER FACE, AFLW, FDDB	4× Tesla P40 GPUs (24GB)	AP 91.7% on WIDER FACE Hard; 22.3ms/image	High accuracy, 3D landmark regression	High resource dependency
Bajpai et al. [17]	GVF, Silhouette Segmentation	A.R., ORL	Pentium Core 2 Duo, 1.83 GHz	GVF: 97.88% (A.R), 31s/image; Silhouette faster	Robust frontal detection	Not suitable for still images
Edirisooriya et al. [18]	SSD, MMOD, Viola-Jones	Webcam, WIDER FACE	Intel i5-4200U, no GPU	SSD: 99.7% precision; MMOD accurate but slow (0.11 FPS)	Low-light and occlusion handling	SSD struggles with small faces
Bazarevsky et al. [19]	BlazeFace	Custom + Real-time Input	iPhone XS, TensorFlow Lite GPU	98.61% AP, 0.6ms/image (200–1000+ FPS)	Designed for mobile real-time use	Slightly lower regression accuracy
Zhang et al. [20]	FaceBoxes	WIDER FACE, FDDB	Xeon E5-2660v3 CPU, GPU	96% mAP; 20 FPS (CPU), 125 FPS (GPU)	Lightweight & fast on CPU	Needs anchor densification
Majeed et al. [11]	Haar, LBP, Dlib (HOG, CNN)	FEI, webcam images	i7-9750H CPU, GTX 1660 Ti	Dlib-CNN: 100% acc, 3.52s/image; Haar fastest (0.02s)	Comparative clarity	No RetinaFace or large-scale test
This Study	Haarcascade, Dlib (HOG, CNN), RetinaFace	Subset WIDER FACE	CPU-only (Intel Core i3, 8GB RAM)	Subset A: RetinaFace best (F1 0.93), Dlib-CNN accurate but slow, Haarcascade fastest. Subset B: RetinaFace best (for all evaluation metrics) and Haarcascade is fastest	Balanced CPU testing, 3-way comparison	Small image set, CPU-only limits generalizability

Based on Table 5, this study offers a practical evaluation of three face detection algorithms—Haarcascade, Dlib (HOG and CNN), and RetinaFace—under CPU-only conditions using a subset of the WIDER FACE dataset. Among the models tested, RetinaFace achieved the best performance (F1-score 0.93) in Subset A and best performance on all evaluation metrics in Subset B, while Dlib-CNN demonstrated high accuracy but was computationally slow, and haarcascade proved fastest with lower precision. Compared to previous research, this study emphasizes balanced CPU-level testing by comparing three methods using a small dataset. However, based on the experimental results on Subset A and Subset B, general conclusions can be drawn. In contrast, other studies such as Bazarevsky *et al.* [19] optimized BlazeFace for mobile GPUs with sub-millisecond inference time, and Deng *et al.* [15] utilized high-resource GPUs for RetinaFace, achieving state-of-the-art accuracy. Studies like Suwarno & Kevin [9] and Majeed *et al.* [11] provided broader algorithmic comparisons but lacked support for RetinaFace or large-scale evaluations. Overall, this study contributes a grounded benchmark for lightweight face detection systems in constrained environments.

IV. CONCLUSION

This study conducted a comprehensive evaluation of three multi-face detection methods—Haarcascade, Dlib (HOG and CNN), and RetinaFace—in a CPU-based environment without GPU acceleration, focusing on efficiency and accuracy under limited resource conditions. A total of 300 images from the WIDER FACE dataset, divided into Subset A (50 images) and Subset B (250 images), were used to assess detection performance across real-world conditions such as varying facial poses, scales, low illumination, partial occlusion, dynamic expressions, and the presence of makeup. The research process included data collection and annotation, image preprocessing, model implementation, and performance evaluation using metrics such as precision, recall, F1-score, accuracy, and processing time. Experimental results indicate that RetinaFace is the most accurate and consistent method, achieving a recall of 0.92, an F1-score of 0.93, and an accuracy of 0.88 on Subset A. The study further demonstrates that RetinaFace dominates all evaluation metrics on Subset B, supported by low standard deviation values, reflecting stable and reliable performance. These advantages stem from its advanced deep learning architecture (ResNet and FPN) and training on a large, diverse dataset. Dlib-CNN exhibited high detection accuracy but incurred extremely long processing times, while Dlib-HOG offered faster performance and high precision but suffered from low recall. Haarcascade, although the fastest, yielded the lowest detection accuracy and proved inadequate for complex visual conditions. By employing a low-resource hardware setup and two evaluation scenarios, this study provides valuable insights for selecting suitable face detection methods for edge devices, low-power computing systems, surveillance, and real-time detection applications with constrained resources.

REFERENCES

- [1] S. Shinde, "Face Recognition Based Attendance System," *Int J Res Appl Sci Eng Technol*, vol. 12, no. 4, pp. 3911–3918, Apr. 2024, doi: 10.22214/ijraset.2024.60784.
- [2] S. Saputra, M. Akbar, and H. Hikmatiar, "Graphical User Interface (GUI) for Face Detection Using Viola-Jones Algorithm," *Bincang Sains dan Teknologi*, vol. 4, no. 01, pp. 1–9, Jan. 2025, doi: 10.56741/bst.v4i01.768.
- [3] S. Kanagamalla, B. B. S. Krishna, R. Abishek, and S. Rajalingam, "Survey and Analysis of Real-Time Face Detection Techniques in Video Scenes: Current Advancements and Future Prospects," in *2023 Second International Conference on Advances in Computational Intelligence and Communication (ICACIC)*, IEEE, Dec. 2023, pp. 1–5. doi: 10.1109/ICACIC59454.2023.10435279.
- [4] S. R. Dubey and S. Mukherjee, "A Multi-Face Challenging Dataset for Robust Face Recognition," in *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, IEEE, Nov. 2018, pp. 168–173. doi: 10.1109/ICARCV.2018.8581283.
- [5] H.-Y. Kim, H.-J. Lee, and S.-G. Lee, "Face Representation Based on Non-Alpha Weberface and Histogram Equalization for Face Recognition Under Varying Illumination Conditions," *Journal of KIISE*, vol. 44, no. 3, pp. 295–305, Mar. 2017, doi: 10.5626/JOK.2017.44.3.295.
- [6] J. Efendi, M. I. Zul, and W. Yunanto, "Real Time Face Recognition using Eigenface and Viola-Jones Face Detector," *JOIV: International Journal on Informatics Visualization*, vol. 1, no. 1, pp. 16–22, Mar. 2017, doi: 10.30630/joiv.1.1.15.
- [7] R. Z. Ye, A. Subramanian, D. Diedrich, H. Lindroth, B. Pickering, and V. Herasevich, "Effects of Image Quality on the Accuracy Human Pose Estimation and Detection of Eye Lid Opening/Closing Using Openpose and DLib," *J Imaging*, vol. 8, no. 12, p. 330, Dec. 2022, doi: 10.3390/jimaging8120330.
- [8] D. Zhang, J. Li, and Z. Shan, "Implementation of Dlib Deep Learning Face Recognition Technology," in *2020 International Conference on Robots & Intelligent System (ICRIS)*, IEEE, Nov. 2020, pp. 88–91. doi: 10.1109/ICRIS52159.2020.00030.
- [9] S. Suwarno and K. Kevin, "Analysis of Face Recognition Algorithm: Dlib and OpenCV," *JOURNAL OF INFORMATICS AND TELECOMMUNICATION ENGINEERING*, vol. 4, no. 1, pp. 173–184, Jul. 2020, doi: 10.31289/jite.v4i1.3865.
- [10] M. P. Jacob, "Comparison of Popular Face Detection and Recognition Techniques," *International Research Journal of Modernization in Engineering Technology and Science*, vol. 3, no. 4, pp. 1304–1313, 2021, [Online]. Available: www.irjmets.com
- [11] K. Abdul Majeed *et al.*, "Face Detectors Evaluation to Select the Fastest among DLIB, HAAR Cascade, and MTCNN," *PAKISTAN JOURNAL OF EMERGING SCIENCE AND TECHNOLOGIES (PJEST)*, vol. 1, no. 1, 2021.
- [12] C. B. M. Dhanalakshmi, V. M. G. V. Bohra, and V. A. Kalsariya, "An Improved Attendance Monitoring System through Facial Recognition using RetinaFace Algorithm," in *2024 3rd International Conference for Innovation in Technology (INOCON)*, IEEE, Mar. 2024, pp. 1–5. doi: 10.1109/INOCON60754.2024.10511664.
- [13] Z. Ren, X. Liu, J. Xu, Y. Zhang, and M. Fang, "LittleFaceNet: A Small-Sized Face Recognition Method Based on RetinaFace and AdaFace," *J Imaging*, vol. 11, no. 1, p. 24, Jan. 2025, doi: 10.3390/jimaging11010024.
- [14] X. Liu, S. Zhang, J. Hu, and P. Mao, "ResRetinaFace: an efficient face detection network based on RetinaFace and residual structure," *J Electron Imaging*, vol. 33, no. 04, Jul. 2024, doi: 10.1117/1.JEI.33.4.043012.
- [15] J. Deng, J. Guo, E. Ververas, I. Kotsia, and S. Zafeiriou, "RetinaFace: Single-Shot Multi-Level Face Localisation in the Wild," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2020, pp. 5202–5211. doi: 10.1109/CVPR42600.2020.00525.
- [16] Q. Wu, X. Wang, N. Li, S. Fong, L. Zhang, and J. Yang, "Real-Time Face and Facial Landmark Joint Detection Based on End-to-End Deep Network," *IEEE Trans Instrum Meas*, vol. 74, pp. 1–13, 2025, doi: 10.1109/TIM.2025.3541698.
- [17] S. Bajpai, A. Singh, and K. V. Karthik, "An Experimental Comparison of Face Detection Algorithms," *International*

Journal of Computer Theory and Engineering, vol. 5, no. 1, pp. 47–51, 2013, doi: 10.7763/ijcte.2013.v5.644.

- [18] T. Edirisooriya and E. Jayatunga, "Comparative Study of Face Detection Methods for Robust Face Recognition Systems," in *2021 5th SLAAI International Conference on Artificial Intelligence (SLAAI-ICAI)*, IEEE, Dec. 2021, pp. 1–6. doi: 10.1109/SLAAI-ICAI54477.2021.9664689.
- [19] V. Bazarevsky, Y. Kartynnik, A. Vakunov, K. Raveendran, and M. Grundmann, "BlazeFace: Sub-millisecond Neural Face Detection on Mobile GPUs," Jul. 2019, [Online]. Available: <http://arxiv.org/abs/1907.05047>
- [20] S. Zhang, X. Zhu, Z. Lei, H. Shi, X. Wang, and S. Z. Li, "FaceBoxes: A CPU Real-time Face Detector with High Accuracy," Aug. 2017, [Online]. Available: <http://arxiv.org/abs/1708.05234>
- [21] S. Yang, P. Luo, C. C. Loy, and X. Tang, "WIDER FACE: A Face Detection Benchmark," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2016, pp. 5525–5533. doi: 10.1109/CVPR.2016.596.
- [22] L. Yang, Y. Zhang, J. Chen, S. Zhang, and D. Z. Chen, "Suggestive Annotation: A Deep Active Learning Framework for Biomedical Image Segmentation," 2017, pp. 399–407. doi: 10.1007/978-3-319-66179-7_46.
- [23] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes Challenge: A Retrospective," *Int J Comput Vis*, vol. 111, no. 1, pp. 98–136, Jan. 2015, doi: 10.1007/s11263-014-0733-5.
- [24] A. Yudhana, R. Umar, and S. Saputra, "Fish Freshness Identification Using Machine Learning: Performance Comparison of k-NN and Naïve Bayes Classifier," *Journal of Computing Science and Engineering*, vol. 16, no. 3, pp. 153–164, Sep. 2022, doi: 10.5626/JCSE.2022.16.3.153.
- [25] Y.-Q. Wang, "An Analysis of the Viola-Jones Face Detection Algorithm," *Image Processing On Line*, vol. 4, pp. 128–148, Jun. 2014, doi: 10.5201/ipol.2014.104.
- [26] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks," *IEEE Signal Process Lett*, vol. 23, no. 10, pp. 1499–1503, Oct. 2016, doi: 10.1109/LSP.2016.2603342.
- [27] M. Najibi, P. Samangouei, R. Chellappa, and L. S. Davis, "SSH: Single Stage Headless Face Detector," in *2017 IEEE International Conference on Computer Vision (ICCV)*, IEEE, Oct. 2017, pp. 4885–4894. doi: 10.1109/ICCV.2017.522.
- [28] H. Jiang and E. Learned-Miller, "Face Detection with the Faster R-CNN," in *2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*, IEEE, May 2017, pp. 650–657. doi: 10.1109/FG.2017.82.
- [29] J. Li et al., "DSFD: Dual Shot Face Detector," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2019, pp. 5055–5064. doi: 10.1109/CVPR.2019.00520.



Sabarudin Saputra is a dedicated lecturer in the Informatics Engineering Program under the Department of Information Technology at Universitas Tadulako, Palu, Indonesia. With a strong passion for advancing technology, his research primarily focuses on image processing and computer vision, areas that play a crucial role in developing innovative solutions across various industries. (email: sicoccinela@gmail.com).



Adi Jufriansah is a student in the Doctor of Physics Program under the Department of Physics at Universitas Gadjah Mada, Yogyakarta, Indonesia. With a strong passion for advancing physics, his research focuses on image/signal processing, biometrics, image analysis, pattern recognition, and mathematical model. Several grants have been obtained at national and international levels from the International Astronomical Union (IAU). (email: saompu@gmail.com).



Muhammad Amirul Mu'min is a lecturer in the Computer Science Program at Universitas Muhammadiyah Bima, Indonesia. He has a special interest in the fields of Cyber Security, Network Forensics, and Network Security which are then researched to enrich the curriculum and provide in-depth insights. (email: mhamirulmumin@gmail.com).



Muhammad Akbar is a lecturer in the Informatics Engineering Program at the Department of Information Technology, at Universitas Tadulako, Palu, Indonesia. His research interests focus on embedded systems, networking, and the Internet of Things (IoT), where he explores innovative solutions to modern technological challenges. (email: akbar.stmikhdy@gmail.com)



Deni Luvi Jayanto is a dedicated lecturer in the Informatics Engineering Program under the Department of Information Technology at Universitas Tadulako, Palu, Indonesia. He has strong passion for advancing technology, his research focuses on information system, digital security and medical informatics. (email: deniluvi@untad.ac.id).



Ayu Hernita is a lecturer in the Information System Program under the Department of Information Technology at Universitas Tadulako, Palu, Indonesia. Her research focuses on machine learning, deep learning, and data mining. She is actively involved in research and the development of artificial intelligence-based technologies applied in various fields such as education, healthcare, and big data analysis. (email: ayunitha@untad.ac.id).