International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 02 Issue: 10 October 2024
Page No: 2440 - 2445
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2024.0334

Real-time Performance Comparison of Face Detection Algorithms using Raspberry Pi

Hetvi Gunjan Shah[1], Vraj Bhavesh Suthar[2], Shital P. Thakkar[3], Vinay M. Thumar[4]*
[1,2]Department of Electronics and Communication, Dharmsinh Desai University, India.
[3]Associate Professor, Department of Electronics and Communication, Dharmsinh Desai University, India.
[4]Professor, Department of Electronics and Communication, Dharmsinh Desai University, India.
Emails: *shahhetvig@gmail.com[1], vrajsuthar17@gmail.com[2], shitalthakkar.ec@ddu.ac.in[3], vinay_thumar@ddu.ac.in[4]*

Abstract
This study reviewed state-of-the-art face-detection techniques like Haar cascade, Dlib HOG, MTCNN, and MediaPipe; implemented and tested them on Raspberry Pi, and evaluated their accuracy, speed, and frames per second. Overall, the research underscores the practical challenges of face detection, including varying lighting, facial expressions, occlusions, poses, scale of face, and accessories, and provides valuable insights for developers and researchers working on edge AI applications on low-cost edge devices. The study found that the MediaPipe face detection algorithm demonstrated robust performance, even with low-quality images, and showed good efficiency and resource management on the Raspberry Pi. The research emphasized the importance of considering factors like accuracy, speed, and resource efficiency in face detection on edge devices. The findings suggest that MediaPipe is a strong candidate for applications requiring efficient face detection, affordable and versatile platforms like Raspberry Pi. By focusing on the Raspberry Pi, the study offers a unique perspective on the performance of state-of-the-art face detection algorithms in real-world, resource-constrained environments, making it a significant contribution to the field of face recognition and edge computing.

Keywords: *Raspberry Pi, Face Detection Algorithms, Edge-Device.*

1. Introduction
Face detection, once a concept relegated to science fiction, has become an integral part of modern technology, seamlessly integrated into our daily lives. Have you ever wondered how your camera identifies faces while taking a selfie? This is the magic of face detection [1]. It involves teaching computers to recognize human faces in images or videos, acting as a digital detective that scrutinizes visuals pixel by pixel. It looks for facial features such as the distance between the eyes, the curve of a smile, and the placement of the nose. Once these features are identified, the computer concludes that it has detected a face. Numerous applications rely on face detection, including home security systems, criminal identification, mobile phone unlocking, and Snapchat filters [1]. The face recognition process typically involves four main stages: face detection, face alignment, feature extraction, and face classification. Among these, feature extraction is particularly crucial as it significantly impacts the overall accuracy of face recognition for a given image. Despite significant advancements in face recognition, detecting faces under extreme variations remains challenging. Literature often discusses new face detection algorithms or compares them on CPU/GPU platforms, frequently claiming applicability to real-time scenarios [2]. However, to test these claims in a practical context, we have undertaken a performance comparison of several state-of-the-art algorithms on the Raspberry Pi. The Raspberry Pi, known for its affordability, accessibility, and versatility, is widely used in various projects. This is because Raspberry Pi allows local data processing, resulting in reduced latency and dependence on cloud services; a critical

specification for devices used in real-time applications. It also helps that Raspberry Pi has multiple connectivity options for interfacing it with various sensors and devices, thus increasing its applications. The objective of this comparison is to identify the most accurate, fastest, and highest-performing face detection algorithm suitable for implementation on edge devices like the Raspberry Pi. By focusing on this compact and affordable platform, we aim to provide valuable insights for developers and researchers working on real-time, resource-constrained edge AI applications [3].

2. Challenges in Face Detection

While most face detectors perform well with large, frontal faces, real-world applications face challenges like occlusion, pose variations, and lighting changes, as shown in Figure 1. Variations in skin tone, makeup, expressions, poses, and accessories like eyeglasses affect detection accuracy. The use of masks due to COVID-19 and accessories such as sunglasses or scarves further complicates detection, particularly in surveillance. Uncontrolled environments introduce lighting, angle, and background challenges. [2] Additionally, edge devices like smartphones and CCTV cameras face performance limitations due to their constrained computational power, storage, and battery life, despite handling large volumes of facial data from selfies, video meetings, and surveillance footage. Using an efficient face detection algorithm is crucial for these devices to perform effectively under such limitations (Figure 1).

Figure 1 Challenges in Face Detection

3. Algorithms

Face detection is a computer vision technology that identifies and locates human faces within digital images or videos. It works by analyzing visual data to detect facial features, such as eyes, nose, ears, mouth, and the overall shape of the face. This process is typically the first step for various applications like face recognition, biometric identification, etc.

3.1. HAAR Cascade: Viola - Jones Algorithm

The HAAR Cascade algorithm was one of the original face detection algorithms developed in 2001 by Paula Viola and Michael Jones, hence giving it an alternate name of Viola – Jones Algorithm. It was used due to its simplicity, efficiency, and real-time performance in edge devices. It utilizes a cascade of classifiers trained with many positive and negative images, allowing it to quickly identify facial features through a combination of edge and line detection. Its ability to perform well on resource-constrained devices made it ideal for early face detection applications, particularly in environments where computational power and memory were limited [3][4].

3.2. DLib-HOG

The DLib-HOG (Histogram of Oriented Gradients) algorithm was developed in 2014 as a part of the C++ libraries. It was used for face detection due to its balance of accuracy and speed. It works by extracting gradient features from images and applying a linear SVM classifier to detect faces, making it robust against variations in lighting, pose, and scale of face. The HOG-based approach is efficient and less computationally intensive than deep learning methods, making it suitable for applications where moderate hardware resources are available. It also offers better accuracy than simpler algorithms like HAAR Cascade, particularly in detecting faces in varied orientations and conditions [5].

3.3. MTCNN

The MTCNN (Multi-task Cascaded Convolutional Networks) algorithm was developed in 2016. It was an effective method for face detection and alignment in images using deep convolutional neural networks (Figure 2a, 2b, 2c, 2d). It is normally chosen for applications that need accurate face detections at different scales and orientations, where getting a bit of time lag is of no issue. It combines three stages of neural networks, P-Net, R-Net and O-Net, that progressively refine face localization and alignment, handling challenging conditions such as varying lighting, occlusion, and complex backgrounds.

MTCNN's multi-task learning approach also enables it to perform face detection and landmark localization simultaneously, making it a popular choice for real-world applications that require a reliable and robust face detection algorithm [7][8].

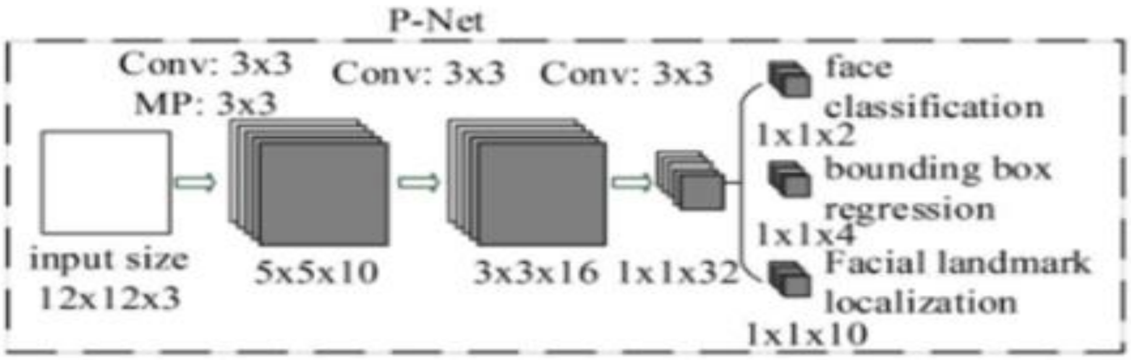

Figure 2(a) P-Net of MTCNN

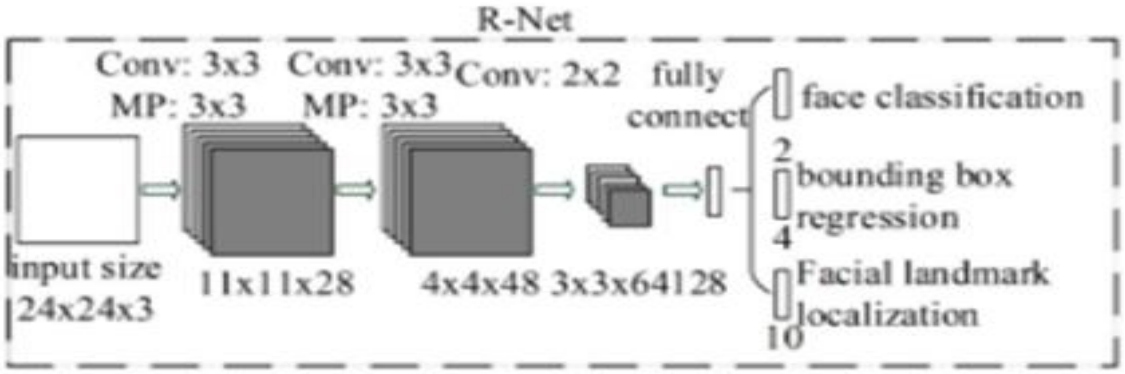

Figure 2(b) R-Net of MTCNN

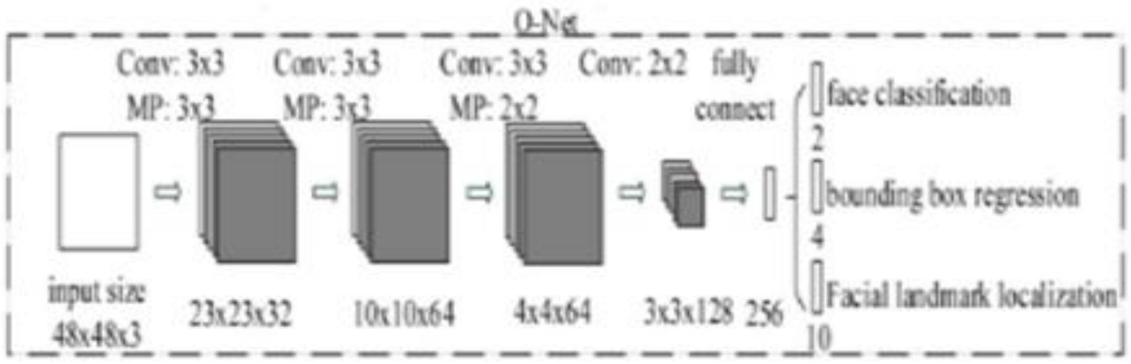

Figure 2(c) O-Net of MTCNN

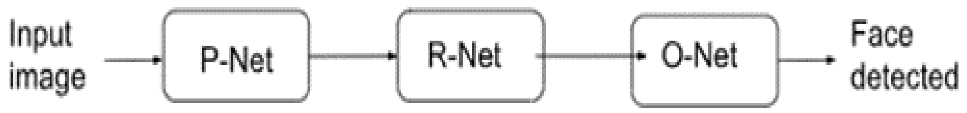

Figure 2(d) MTCNN

3.4. Media Pipe

The MediaPipe is a framework developed by Google in 2019. It provides high accuracy, cross-platform compatibility, efficient, real-time perception pipelines for various computer vision tasks, including face detection, hand tracking, and pose estimation, using machine learning techniques [9]. It is normally chosen for applications that need fast face detections on resource-constrained devices like smartphones. It also supports multitasking, allowing it to detect faces and landmarks simultaneously, which is useful for applications like augmented reality, video conferencing, and facial recognition in dynamic environments like crowded public places or outdoor surveillance [10].

4. Hardware Specifications

The Raspberry Pi 4 B features a 64-bit Quad-core Cortex SoC running at 1.5GHz with 2GB of RAM, making it suitable for various computing tasks. It offers dual-band Wi-Fi, Bluetooth 5.0, BLE for wireless connectivity and a Gigabit Ethernet port for wired connections. The board includes two USB 2.0 ports, two USB 3.0 ports, a 40-pin GPIO header for external devices, a 2-lane MIPI CSI interface for camera connections, and a 2-lane MIPI DSI for touchscreens. It also supports up to 4K resolution via two micro-HDMI ports and is powered through a 5V, 3A USB Type-C input.

5. Software Specifications

Raspberry Pi 4 B is compatible with various operating systems including Raspbian OS, Ubuntu, and other Linux distributions. It supports multiple programming languages such as Python, C/C++, Java, and Node.js, and offers a range of development tools and environments like Thony Python IDE, making it versatile for developers working on diverse applications (Figure 3 & 4).

6. Setup

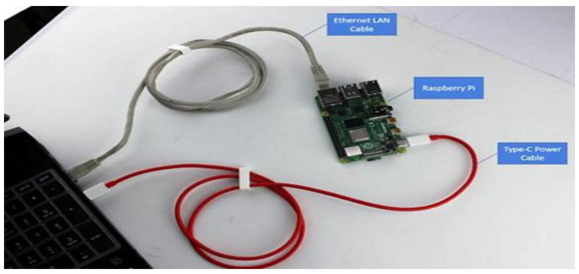

Figure 3 **Hardware Setup**

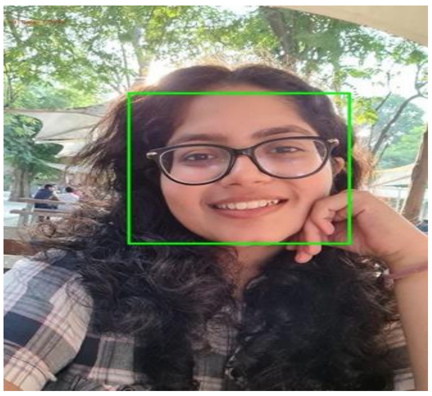

Figure 4 **Example of Successful Face Detection Using the Above Setup**

7. Dataset

We have used two datasets for this project – a standard dataset with all the images taken from the internet, and a personal dataset using casual pictures and selfies featuring the authors (Figure 5 & 6). Each of these datasets was divided into six categories based on various conditions in which a photo could be clicked. These six categories are accessories, facial expressions, lighting, occlusion, pose, and scale of face. The standard dataset contained about 120 images. So, for each of these six categories, we have 20 images for this dataset. The personal dataset contained about 60 images. So, for each of these six categories, we have 10 images for this dataset.

8. Inference Comparison Under Various Conditions

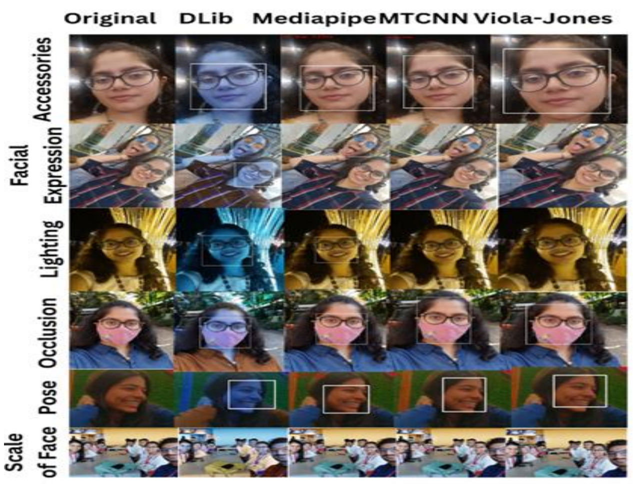

Figure 5 **Results Based Images Processed for Various Categories for All the Algorithms**

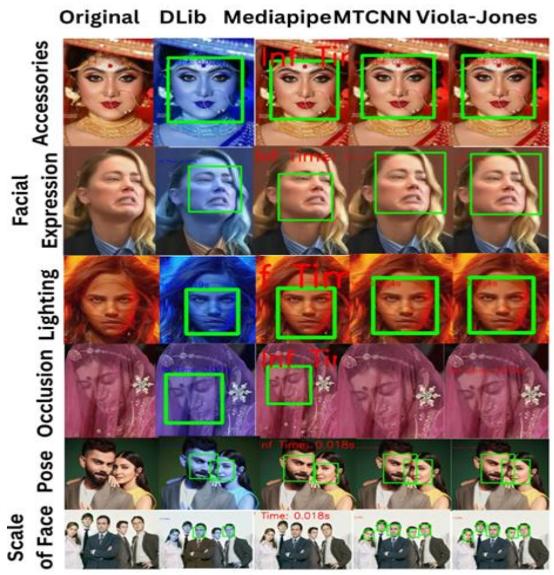

Figure 6 **Results Based Images Processed for Various Categories for All the Algorithms**

9. Results and Discussion

9.1. Results

As mentioned, face detection algorithms struggle to outline exact faces in various scenarios. The following results showcase a quantitative analysis of four algorithms concerning edge devices like Raspberry Pi. We are comparing these four algorithms based on accuracy and speed to replace the older algorithms in IoTs and edge devices with newer and more efficient algorithms (Table 1 & 2).

9.2. Discussion

MediaPipe emerged as the fastest and most accurate face detection method in our study, making it ideal for real-time applications where both speed and precision are critical. Dlib ranked second in both execution time and accuracy, offering a good balance for diverse use cases. Viola-Jones showed moderate speed but the lowest accuracy, indicating it may not be suitable for applications where high precision is required. MTCNN, while achieving high accuracy, had the slowest execution time, making it less suitable for real-time applications despite its detection capabilities.

$$\text{Avg. execution time} = \frac{maximum\ execution\ time + minimum\ execution\ time}{2}$$

$$\text{Frames per second (fps)} = \frac{1}{avg.\ execution\ time}$$

$$\text{Accuracy} = \frac{Detected\ images}{Total\ images} * 100$$

Table 1 **Algorithm's Quantitative Analysis (For a Standardized Dataset of Pictures Taken from the Internet)**

Algorithm	Average Execution Time (s)	Speed (fps)	Accuracy (%)
Dlib – HOG	3.282	0.304	64.28
MediaPipe	0.0615	16.26	71.43
MTCNN	21.606	0.046	66.43
Viola Jones	4.074	0.245	58.57

$$\text{Avg. execution time} = \frac{maximum\ execution\ time + minimum\ execution\ time}{2}$$

$$\text{Frames per second (fps)} = \frac{1}{avg.\ execution\ time}$$

$$\text{Accuracy} = \frac{Detected\ images}{Total\ images} * 100$$

Table 2 **Algorithm's Quantitative Analysis (For a Dataset of the Author's Casual Pictures)**

Algorithm	Average Execution Time (s)	Speed (fps)	Accuracy (%)
Dlib – HOG	7.0385	0.142	72.58
MediaPipe	0.1155	8.658	37
MTCNN	12.5085	0.0799	58
Viola Jones	5.0885	0.196	50

Conclusion

This project has been a comprehensive exploration into the capabilities of various image processing algorithms on a resource-constrained platform. By analysing the data and performance metrics, we can draw valuable insights into the suitability of these algorithms for real-world applications. The Raspberry Pi 4 is known for its compact size and limited computational power. In this project, we tested the resilience of image processing algorithms in such an environment. Efficiency played a critical role in the evaluation of the algorithms. MediaPipe emerged as the most efficient choice, consistently providing high Frames Per Second (FPS) while maintaining low execution times. This efficiency is pivotal for real-time image processing applications, particularly when operating under resource constraints. Each algorithm displayed unique strengths. MediaPipe excelled in real-time applications, offering rapid and reliable performance. Dlib - HOG showed robustness in scenarios with mostly frontal or slightly non-frontal faces. MTCNN demonstrated competence in detecting faces under diverse poses and lighting conditions. Viola Jones, although slightly less efficient, stood out for distinguishing faces in arbitrary images with low false positive rates and high true positives. In conclusion, real-time applications should prioritize efficiency and responsiveness, making MediaPipe a top choice. When accuracy is paramount, Dlib - HOG, MTCNN, and Viola Jones offer compelling alternatives. This project underscores the importance of making informed decisions when selecting image-processing algorithms for resource-constrained platforms. The performance data and analysis provided here offer valuable guidance for developers and researchers working with Raspberry Pi 4 and similar embedded systems while implementing image processing tasks on SoC (System on Chip).

References

[1]. M. Zamir, N. Ali, A. Naseem, et al., “Face detection & recognition from images & videos based on CNN & raspberry pit,” Computation, vol. 10, no. 9, p. 148, 2022.

[2]. Y. Feng, S. Yu, H. Peng, Y.-R. Li, and J. Zhang, “Detect faces efficiently: A survey and evaluations,” IEEE Transactions on Biometrics, Behavior, and Identity Science, vol. 4, no. 1, pp. 1–18, 2021.

[3]. I. Gupta, V. Patil, and S. Kadam, “Face detection and recognition using raspberry pi,” Dec. 2016.

[4]. M.Dahake and N. Mandaogade, “Implementation of raspberry pi for human face detection & recognition,” Tech Res Pap Compet Students, pp. 2–5, 2017.

[5]. A. Jadhav, S. Lone, S. Matey, T. Madamwar, and S. Jakhete, “Survey on face detection algorithms,” Int. J. Innov. Sci. Res. Technol, vol. 6, pp. 291–297, 2021.

[6]. K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, “Joint face detection and alignment using multitask cascaded convolutional networks,” IEEE signal processing letters, vol. 23, no. 10, pp. 1499–1503, 2016.

[7]. “Histograms of Oriented Gradients for Human Detection” by Navneet Dalal and Bill Triggs, presented at the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) in 2005.

[8]. "Rapid Object Detection using a Boosted Cascade of Simple Features" by Paul Viola and Michael Jones, presented at the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR).

[9]. “Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks" by Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao, published in 2016.

[10]. "MediaPipe: A Framework for Building Perception Pipelines" by François Bérard, Clément Henry, Irene Alvarado, Daniel Smullen, and many others, presented at the 2020 ACM Multimedia Conference.