IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 30 June 2022, accepted 10 July 2022, date of publication 8 August 2022, date of current version 12 August 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3194915

SURVEY

Hardware Accelerators for Real-Time Face Recognition: A Survey

ASMA BAOBAID, MAHMOUD MERIBOUT, (Senior Member, IEEE), VARUN KUMAR TIWARI, AND JUAN PABLO PENA

Department of Electrical Engineering and Computer Science, College of Engineering, Khalifa University, Abu Dhabi, United Arab Emirates

Corresponding author: Asma Baobaid (asma.baobaid@outlook.com)

This work was supported by Khalifa University under Grant CIRA-2020-86.

ABSTRACT Real-time face recognition has been of great interest in the last decade due to its wide and varied critical applications which include biometrics, security in public places, and identification in login systems. This has encouraged researchers to design fast and accurate embedded and portable systems that are capable of detecting and recognizing a large number of faces at almost a video frame rate. Due to the increasing volume of reference faces, traditional general-purpose computing engines such as the ones based on Intel's Pentium processors have shown not to be adequate and various dedicated hardware accelerators based on either Graphical Processing Units (GPU), Field Programmable Gate Arrays (FPGA), Application Specific Integrated Circuits (ASIC), or even multi-core Central Processing Units (CPU) have emerged. Earlier published review papers on face detection/recognition have discussed face detection and face recognition algorithms enhancement that improve the detection accuracy. Nevertheless, none of them have reviewed the hardware accelerators used for this application. Accordingly, this paper aims to provide a comprehensive review of the most recent face recognition algorithms and associated embedded hardware systems targeting real-time performance. A detailed comparison between neural network and non-neural network-based algorithms in terms of accuracy and processing time is provided. Discussions on their suitability to be implemented into parallel hardware architectures such as Single Instruction Multiple Thread (SIMT) or Single Instruction Multiple Data (SIMD) is also discussed.

INDEX TERMS Face recognition, face detection, FPGA, GPU, multicore CPU, neural network algorithms, non-neural network algorithms.

I. INTRODUCTION

The use of biometric features, such as the face, fingerprint, voice, hand geometry, and retina eye, which are based on human's unique biological, physical and behavioral characteristics has been widely used for identification and recognition tasks [1]. These features have the advantage over other computer security tools, such as passwords, confidential codes, or handheld tokens in being more secure [2]. Among all the aforementioned biometric features, human faces remain one of the most effective features used for human identification since the associated systems can be contactless and can handle several humans simultaneously.

The associate editor coordinating the review of this manuscript and approving it for publication was Zahid Akhtar.

This has led researchers to suggest various embedded face recognition systems, which were driven by the great success of low-power edge devices as well as memory devices in terms of computation power and latency/bandwidth respectively. These systems nowadays deployed, as indispensable systems, at various premises, such as airports, healthcare facilities, bus stations, social media websites, and so many other places, for criminal identification, security reasons, and user authentication [3]. Some of the most important features of a face recognition system are the number of faces it can recognize under different illumination and pose, the corresponding accuracy, the ability to be scalable to handle more reference images, the computation time, and power consumption. Almost all existing systems perform well when test images are captured under similar conditions as the training

images. However, problems such as poses, facial expression, illumination changes, hairstyle, and cosmetics changes are still unsolved [4]. Therefore, with the rapid growth of technology and artificial intelligence, advanced hardware accelerators based on either GPU, FPGA, ASIC, or multicore CPUs are continuing to emerge to yield impressive performances at low cost [5]. This has contributed to promote parallelizable face recognition hardware algorithms which can be grouped into neural network (e.g. DNN and CNN) or statistical-based algorithms.

The main contributions of this paper are:

1. A comprehensive critical review of most recent face recognition algorithms targeting real-time and portable applications (i.e. either AI-based or not), along with their respective performance.

2. Present the commonly associated hardware accelerators and compare their performance.

3. Suggest some solutions which may improve the performance of the existing systems.

Several other review papers on face detection and recognition were recently published. For instance, in [6] the authors described the development stages of face recognition along with the related developed techniques in each stage. The authors compared the accuracy performance of these techniques in the Labeled Faces in the Wild (LFW) dataset and summarized the performance of different datasets used in face recognition systems. In [7], the authors categorized the common face recognition algorithms into appearance-based, feature-based and soft computing-based. In the paper, existing studies previously done on the topic were used to compare and discuss the performance of these approaches. The paper concluded that feature-based face recognition algorithms achieved the highest recognition accuracy compared to appearance-based and soft computing-based algorithms, with a normalized rate of 95%. In a very recent review paper [8], a wide variety of face detection algorithms are divided into feature-based and image-based approaches. The authors provided a detailed description along with a comparative evaluation of all the algorithms among the two categories. The authors concluded that feature-based approaches are highly preferred for real-time detection, whereas image-based approaches achieve better performance for gray-scale images. Additionally, among the sub-areas, neural network algorithms were capable of achieving very high performance. Similarly, in another recent survey [5], authors compared local approaches; which describe the image using some main features, holistic approaches; which use the complete face features, and hybrid approaches that combine both. The comparison between techniques that fall under each category was in terms of accuracy, robustness, complexity, and discrimination. It was concluded that local approaches outperform the other two approaches in terms of discrimination, accuracy, and complexity.

Hence, all of the above mentioned review papers mainly presented the algorithms used for face detection and recognition, along with their corresponding accuracy; but without tackling the dedicated hardware accelerators that achieve real-time performance.

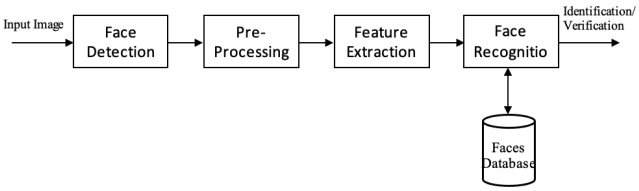

FIGURE 1. A typical face recognition system.

II. FACE RECOGNITION SYSTEM

Face recognition systems typically consist of three main stages; face detection, feature extraction, and face recognition as shown in Fig.1. The detection stage verifies the presence of a face in an image or a video. If the face is present, it locates the face region and the extent of each face by generating bounding rectangular or elliptical boxes around only the faces in any image. After the face is detected, the image is preprocessed to reduce the processing time and facilitate the face recognition process. Despite causing some information losses, the preprocessing step helps to speed up and enhance the accuracy of the recognition phase by eliminating all kinds of non-relevant details. The selection of the preprocessing techniques depends on the type of application required. One preprocessing technique is normalization, where face images of different scales are transformed to the same scale [9]. Face alignment is another technique that aims to identify the facial landmark of the face (nose, eyes, mouth, etc.) and attempt to align the position of those facial landmarks by rotation, transition, or cropping. Additionally, one of the easiest preprocessing techniques is changing the brightness of the image [10].

The next step is feature extraction, where the main features of the face images detected in the first step are extracted. In feature extraction, the face is represented with a set of feature vectors that describe the features of the face image (e.g. mouth, nose, and eyes) with their geometry distribution [5]. The most common available feature extraction are Local Binary Pattern (LBP) [11], Principle Component Analysis (PCA) [12], [13] Independent Component Analysis (ICA) [14], Linear Discriminant Analysis (LDA) [15], Eigenface [13] and Gabor filter.

In the last step of the process, the face recognition step, the features extracted from the image during the previous step are compared with known faces in a specific dataset. Face recognition can operate in two different modes: face verification and face identification. Face verification is when the input face is compared with other face images to identify whether or not the input face corresponds to the claimed identity. Whereas, face identification compares a face image with other face images to find the most likely person in the database that corresponds to the input face. Machine learning algorithms like Convolution Neural Network (CNN) are most preferred in recent time as they can efficiently address

this task, following a consistent and comprehensive learning procedure [9], [10].

In order to fairly assess the performance of different face detection algorithms, databases comprising images collected in a controlled environment are usually used by researchers. This includes for instance LFW dataset which comprises 12,233 images of unique 5,749 persons of different images and which was mostly used in the literature. It was also useful for hardware researchers to fine-tune the algorithms parameters such as the data types and number of bits to be optimally assigned to the variables, without altering too much the system accuracy.

III. HARDWARE PLATFORMS FOR FACE DETECTION AND RECOGNITION

Hardware devices targeting embedded applications (e.g. edge and IoT devices) have improved over the years in both their form factor and the number of operations per second per Watt they can handle. This development gave the the designers the opportunity to select from a wider spectrum of processors the most suitable hardware accelerator for their systems. The most commonly known hardware platforms are CPU-based architectures (Multi-Core CPUs or GPUs), and field-programmable gate arrays (FPGAs).

A. FIELD-PROGRAMMABLE GATE ARRAYS (FPGAs)

FPGA can be defined as a programmable semiconductor device that comprises an array of typically tens of thousands of Configurable Logic Blocks (CLB) and hundreds of floating-point Digital Signal Processing (DSP) blocks which are interconnected via a programmable interconnection network [16]. Unlike other devices like CPUs and GPUs, FPGA does not embed a structured chip or a hardwired CPU. This allows them to avoid the long latency process caused by the repeated fetch and decode operations for every single instruction, that can usually hold not more than two operands. Indeed, a stream of multi-operations and multi-operands functions can be executed at once in one or several pipeline stages. This would provide high throughput and low power consumption, but probably at a slower clock rate than multi-core CPU and GPU devices. Yet, FPGAs long development time and challenging programming codes are their main drawbacks [17].

Two major companies, namely Altera and Xilinx, emerged to successfully provide different FPGA chips and associated software development tools [17]. Nevertheless, FPGAs are still not adequate to host DNN algorithms because of their relatively low memory storage capacity to store all the weights. This requires extensive external memory, which significantly increases the algorithm latency. Altera was recently acquired by Intel, which may indicate the future direction of the next generation of multi-core CPU to incorporate reprogrammable devices to accelerate machine vision and image processing tasks. This is the case of the recent Xilinx Zynq Ultrascale + MPSoC FPGA, which is one of the largest ICs comprising millions of logic cells and which includes a quad-core 1.5 GHz 64-bits Cortex A-3 processor, and a dual-core Arm Cortext R5F making it suitable to host Xilinx's deep learning processing unit (DPU), created for researchers working on hardwired machine learning algorithms [18]. The chip which supports several CNNs such as VGG, ResNet, GoogLeNet, YOLO, SSD, MobileNet, and FPN is supported by an associated software development kit which allows performing pruning and quantization to satisfy low latency (DECENT), mapping the neural network to the DPU instructions (DNCC), and handling resource allocation and DPU scheduling (N2Cube). Very recently, in the Arm TechCon exhibition, the chip was demonstrated to successfully perform traffic light detection [19]. When running CNN tasks, it achieves 14 images/second/watt, which outperforms the Tesla K40 GPU (4 images/second/watt). Also, for object tracking tasks, it reaches 60 Frames Per Second (fps) in a live 1080p video stream.

B. FIXED ARCHITECTURE DEVICES

1) MULTI-CORE CENTRAL PROCESSING UNIT (CPU)

The CPU is designed in such a way that it reduces the program's latency by including different levels of memory caches within the same chip, while executing simultaneously different tasks such as data transfer, branch predictions, and arithmetic/floating-point operations [20]. The original design of the CPU had only one single-core or a single central processing unit. However, to increase the performance of the CPU, manufacturers added additional cores to the CPU, and this processor is referred to as a multi-core processor. These latest CPUs are usually optimized for single/multi-threads operations, and they typically include single instruction multiple data (SIMD) vector instructions, such as SSE, SSE, and AVX vector extensions, which typically operate from 64 to up to 1024 bit data elements. Several low power multi-core CPUs were designed for real-time image processing applications targeting edge or IoT devices. This include low latency multi-core RISC-V CPU (8 CPU), DNN hardware accelerators, and the IoT-based GPA9 processor provided by Greenwave-technologies Company. The GPA9 processor features a low power for a high throughput (0.33 mW/GOP), and it can yield up to 150.8 GOPS at a maximal frequency of 400 MHz to manipulate 8 to 64-bit fixed and floating-point operands. The processor also features an 8-bit CPI (Camera Parallel Interface), and it also includes security modules such as AES128/256 cryptography that allows the device to be uniquely and securely identified. The processor is dedicated for processing small-sized images (160 x 160 image), where the MobileNet V1 network can run in just 12ms for a power consumption of 806 mW/frame/s [17].

Using this processor, it was shown that the processor can perform face recognition, with 1 second latency, using only 1.5 Mb of weights with 16-bit fixed point integer computation. Another multi-core 16 nm technology processor, namely the Myrad X VPU, from Intel Corporation, was also recently released to target AI-based real-time

image processing applications. The processor which operates at 700 MHz, includes a powerful neural compute engine for DNN acceleration. The processor also comprises 16 x 128-bit VLIW vector processors to yield 1 TOPs at a maximal 2.5 W power consumption [19].

2) GRAPHICS PROCESSING UNIT (GPU)

GPUs [21] have been widely used and specially designed for different applications like games, video processing, and image processing. They consist of many processing cores, and they are used to perform fast matrix calculations in parallel [17]. In their latest generations, GPUs have high processing speed and they are cost-effective.

The basic design of a modern-day GPU consists of a set of streaming multiprocessors (SMs). Each SM is composed of several Streaming Processors (SP), which in turn comprise an ALU that executes integer and floating-point arithmetic and logic operations. Each SM contains a shared L1 cache and all SMs can intercommunicate using a shared L2 cache and the off-chip memory (GDDR). To enhance the GPU overall performance, the shared L1 memory is composed of 32 banks (one bank per SP), so that all threads in a warp can access different memory banks in parallel. Thus, GPUs are based on single instruction multiple thread (SIMT) architectures and are designed to maximize the amount of parallel processing in the graphics pipeline. Unlike SIMD architecture, where the processing engines typically execute the same instruction, SIMT architecture allows different threads to more readily follow divergent execution paths through a given thread program. NVIDIA's Jetson AGX Xavier processor is one of the most recent GPU processors targeting embedded machine vision and AI applications. It comprises 64 Tensor cores, in addition to an 8-core ARM v8.2 64-bit CPU. The processor can be configured to consume 10W, 15W, or 30W based on the target performance [22].

The Ampere 100 Tensor core is another very recent GPU which was used for real-time embedded image processing systems [23]. It can be dynamically partitioned into up to 7 GPU instances to achieve up to 9.1 TFLOPS FP64 or 19.5 TFLOPS FP32, or up to 1248 TOPS INT9 Tensor core. However, the processor consumes more than 100 W.

IV. FACE DETECTION/RECOGNITION ALGORITHMS TARGETING REAL-TIME AND EMBEDDED APPLICATIONS

A. NON-NEURAL NETWORK ALGORITHMS

1) PRINCIPLE COMPONENT ANALYSIS (PCA)

Principle Component Analysis (PCA) is the most widely used statistical approach for data dimensionality reduction. It is used for feature extraction, image recognition, classification, and image compression [12]. This technique is also known as Eigenspace Projection or Karhunen-Loeve Transformation. In 1991, Turk and Pentland of MIT Media Laboratory introduced PCA for face recognition [6]. PCA involves several steps that transforms a set of correlated variables into a small

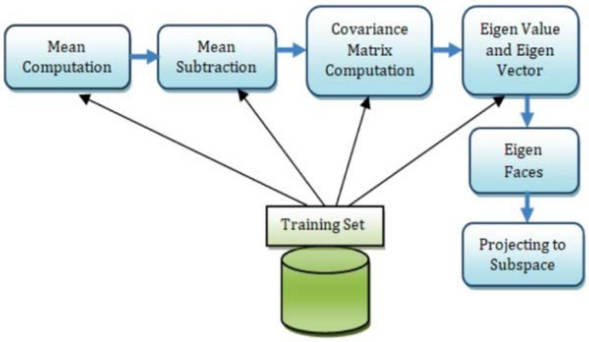

FIGURE 2. PCA based face recognition [94].

number of uncorrelated variables called principal components. These transformations are done to achieve the main goal of PCA, which is transforming the two-dimensional image into a one-dimensional feature vector in subspace [24]. Fig.2 shows the steps to find the principal components. These steps can be summarized as follow [25]:

a. Collect x_i of an n-dimensional data set x, where $i = 1, 2, 3 \ldots m$

b. Normalize face vectors by calculating the mean m_x of all images in the database & subtract it from each data point $x_i - m_x$.

c. Covariance matrix computation: $C = (x_i - m_x)(x_i - m_x)^T$.

d. Eigen Value and Eigen Vector Computation of matrix C.

e. Order the eigenvalues and corresponding eigenvectors in descending order.

f. Select the first $d \leq n$ eigenvectors that will generate a dataset representing the whole training set.

g. By similarity measure, each projected test image is compared to every projected training image to output the closest training image to the test image.

The main advantages of the PCA algorithm are that it is less sensitive to noise, has higher efficiency as it operates in a space of smaller dimensions, and it requires less memory space and capacity than AI-based algorithms [26]. However, one of its drawbacks is that the training dataset has to be large enough, around thousands, for the results to be meaningful [6].

All the above steps, from *a* to *f*, are done offline, during the training phase and hence do not require hardware accelerators. On the other hand, Step *g* is required to be performed online for each input image. It consists of projecting the input image into the selected principal components (PCs) after data normalization. This high computation demand, which requires dense matrix multiplications, has led several researchers to suggest PCA-based parallel implementations of face detection/recognition using either FPGA or GPU processors [27], [28]. Overall, it was demonstrated that FPGAs offer the best performance (in terms of combined power consumption and throughput) over GPUs, at the expense of higher design efforts and a reduced image database. Some researchers have tackled some

of these issues by using high level synthesis tools HLS) to implement the Eigenvalue value decomposition (EVD) of singular value decomposition (SVD) which remains an important module of PCA algorithms [29]–[31]. Recently, in [32], a complete PCA implementation on Xilinx's Virtex7 was done for a small matrix dimension. Hence, two blocks of memory were considered to store the rows and columns elements of the matrices respectively to perform one row-column multiplication, which required high resources within the FPGA. Experimental results have shown that for small-sized matrices FPGA-based implementation is faster and yields lower power consumption than CPU and GPU-based implementations.

For face recognition applications, an embedded face recognition system terminal scheme based on PCA was proposed in [33]. The system used Xilinx's Zynq-7000 FPGA, which has the advantage of co-hosting an ARM9 CPU core for sequential processing and 7.7 million programmable logic cells to implement datapaths and glue logic. A recognition throughput of 9960 faces/second, with a high recognition rate of 95.8%, was achieved using a system clock frequency of 100 MHz. However, only eight people out of 40 subjects from the ORL database were considered during testing, which is relatively low. In [34] a face recognition system based on PCA Eigenfaces for recognition, and the adaboost algorithm, which is based on the Haar transform, for detection was proposed. The two algorithms are implemented to process the data in parallel on NVIDIA's GeForce GTX 770 GPU (1536-Core). Input images are down-sampled to a block size of 24 x 24 pixels to yield a processing time of around 350 ms and 312 ms for face detection and recognition, respectively, for 700 x 580 images. The host mapped memory was used as a shared memory to avoid transferring images from the host to the GPU module whenever they were needed by a thread. Each GPU block is allocated to one down-sampled image to simultaneously search for faces within the associated image. The detected face is then forwarded to the face recognition module without waiting for all faces to be detected. The two algorithms were also implemented into a 3.1 GHz Intel Xenon CPU (4-Core) to evaluate the GPU hardware. The face detection was achieved within a throughput of 109 frames/s on the GPU platform, while its execution on the CPU platform was at least 5 times slower. In [35] an attendance management system based on the eigenfaces algorithm that uses PCA for both face detection and recognition was developed on Raspberry PI 2 multicores CPU to achieve 88% accuracy. However, it was not mentioned how many persons could be accommodated in the system. Similarly, in [36] Eigenfaces for feature extraction, PCA for decreasing the size of the image and Euclidean distance classifier for recognition were used to implement a face recognition system. The proposed method achieved 82.5% as the best recognition rate when tested on the ORL database, which consists of 40 distinct subjects with 10 different images for each. The proposed system was implemented on Raspberry Pi 3 Model B+ (Quad-Core) where it achieved a processing time of 0.206 seconds which is much lower compared to other implementations that are i7 PC-based. Nevertheless, the system has several limitations like the restricted pose of a person, restricted expressions, and very bounded illumination conditions. Additionally, the suggested Raspberry Pi board has a limited memory space to host a reasonable amount of training data and to process good resolution images.

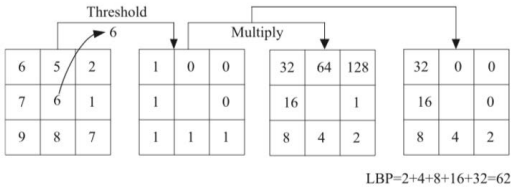

FIGURE 3. The process of LBP technique [39].

In summary, the high computation requirements of the PCA inferring phase, which consist mainly of deep matrix multiplication and addition, make this algorithm not suitable to handle large number of faces using multi-cores CPUs in the lower range of the spectrum. Nevertheless, IoT based face recognition systems using PCA algorithm are quite feasible.

2) LOCAL BINARY PATTERN (LBP)

The Local Binary Pattern (LBP) is one of the powerful low level image processing methods used to describe the texture and shape of a digital image and has been successfully used for face detection and recognition applications [37]. It extracts the structure of each pixel by comparing its intensity with the one corresponding to the surrounded pixels. Its basic principle is to divide the input facial images into blocks of n x n pixels (with usually n = 3) [24]. The center pixel of each of these blocks is then used as a threshold or a reference for thresholding when compared with the surrounding pixels to produce a binary code. For instance, if the neighbor's pixel value is greater than the center pixel value, the neighbor's value will be set as '1', otherwise, it will be set as '0' [5], [24]. Hence, for the case of n = 3, the LBP algorithm would generate an 8-digit binary code, which is obtained by multiplying the thresholded values by weights given by powers of 2. The partial results are then aggregated with the obtained results in a clockwise direction [38]. Mathematically, for n = 3, the LBP operator, $LBP_{(P,R)}$, can be defined as follows:

$$\sum_{i=0}^{8} 2^p * s(p_i - p_c) \tag{1}$$

where,

$$s(x) = \begin{cases} 1, & x \geq 0. \\ 0, & x < 0. \end{cases} \tag{2}$$

where p is the number of sampling pixels, R is the radius, p_c is the center pixel, and p_i is the neighbor's pixel. Fig.3 below

shows an example of the computation of the binary code corresponding to a 3 x 3 block. In terms of computation complexity, the multiplication operations can be substituted with left-shift registers, requiring a total of 8 shift registers of 2 to 8-bit width and an 8 operands adder to yield an 8-bits result. This can be easily implemented into FPGA to yield one clock cycle latency. It can also be optimized to run in few clock cycles by the exploring the instruction level parallelism (ILP) which is available in multi-core CPUs and GPUs as well. The next step of the LBP algorithm is to build a histogram and compute the Chi-distance which can be defined as follows:

$$\chi^2(S, M) = \sum_{i}^{len} \frac{(S_i - M_i)^2}{S_i + M_i} \tag{3}$$

where *len* is the length of the feature vector, S_i the sample and M_i the model image in the respective bin [37]. Hence, the Chi-distance measures the similarity between two LBP images, where the lower the value, the higher similarity between these two images. However, the recurrent division featured in Equation 2, can cause significant latency using multi-cores CPU or GPU and is hardware costly on FPGA. Hence, a hardware-friendly approximation such as division by a constant factor, preferably which is a multiple of 2 is worth investigating. It can also be noted that the LBP algorithm consists of low-level image processing tasks where neighboring operations are executed to compute the binary code (i.e. equations 1 and 2) and an intermediate level image processing to compute the histogram. Thus, these tasks are highly parallelizable at a fine-grain level and can be efficiently hosted into FPGA, multicore-CPUs, or GPU processors. This has motivated researchers to suggest some LBP-based hardware accelerators for face detection/recognition. Specially, that this algorithm is invariant to both luminosity and rotations and intrinsically features local information such as curve edges, spots, and corners.

For instance, in [39], a real-time low-memory multi-face detection system using LBP was implemented on an Altera Cyclone IV FPGA chip. A Naive Bayes classifier to recognize candidate faces was used to, achieve 96.14% accuracy with a throughput of 60 FPS. In [40], a face recognition system based on LBP and its variants (i.e. Rotation LBP (RILBP) and Pyramid of LBP (PLBP)) were also implemented. The system was tested on three frontal face datasets; Database of Faces (TDF) with 447 images of 27 subjects; Caltech Faces 1999 (CF1999) with 40 subjects, a combination of TDF and CF1999; and Labeled Faces in the Wild (LFW) with 12233 images of 5749 people. The system, which considered 6 x 7 blocks, obtained a very low recognition rate on the first two datasets and a high recognition rate of 90.95% for the LFW dataset. The low accuracy was mainly caused by the skewness and blurriness, which have made the matching between face images harder because of loss of texture information. The algorithms were implemented on a general-purpose computer, and no execution time analysis was provided. Like other methods, most LBP-based face recognition algorithms were tested within a controlled environment. This includes the work done in [41], where a face verification and identification method under various illumination variation conditions using a combination of LBP, VanderLugt correlator, and DoG filtering was introduced. Accuracies of 98.4% and 94.6% could be achieved on YaleB and YaleB extended datasets, respectively. The two sets contain images of 10 and 38 individuals, respectively, under different lighting conditions. However, the proposed algorithm was tested with the pose invariance of each person using a general-purpose computer, with again no indication of the execution time. Similarly, in [42], LBP and K-NN were used together for face detection and recognition and achieved an accuracy of 99.26% on the CMU-PIE database of 750000 images of 337 people, and a lower recognition rate on the LFW dataset. In [43] a variant of the LBP technique that is more discernment and less sensitive to noise, named Local Ternary Patterns (LTP) was proposed. This technique could yield a high recognition accuracy on three datasets; CMU-PIE, Yale B, and O2FN datasets. The accuracy was increased even further in [44], where the Relaxed LTP technique was introduced. This technique demonstrated higher recognition results than the original LTP method on the same three datasets. Similarly, in [45], an improvement in the accuracy is achieved by introducing Enhanced Local Ternary Pattern (ELTP), which has also surpassed the original LBP and LTP techniques. Likewise, a fast parallelized face recognition based on LBP that obtained an accuracy of approximately 85% was suggested in [37]. In [46] a new method using LBP combined with advanced image processing techniques such as contrast adjustment and image blending to improve the accuracy of face recognition and face detection is presented. Without the use of these techniques, the system obtained face recognition accuracy of 89.3% on a dataset of 400 faces. However, the use of advanced image processing techniques improved the recognition accuracy to 99% on a dataset of around 760 faces and improved the detection accuracy to 95%.

With regard to LBP-based hardware accelerators, a GPU-based parallelized approach of LBP for face recognition using AMD's Radeon HD 7650 GPU processor was described in [45] using the OpenCL programming language. The parallel implementation consists of allocating to each GPU core of the processor a sub-block of the input image, which is adequate since this avoids using inter-block communications which is very time consuming. The system, which was tested on large images (4096 x 4096 pixels), could achieve a good performance of 74% to 79% on the ORL dataset, while outperforming the CPU platform (i.e. Intel core i3), in terms of execution time, by a factor of 251. In another work [47], Haar cascade and LBP algorithms were used for face detection and face recognition respectively using an edge server device, namely the Intel Xeon E5 (8-core CPU) processor operating at 2.4 GHz and a

smartphone (Cortex-A7 Dual-Core CPU), operating at 1.7 GHz. The results showed that the computation time corresponding to face detection depends on the number of faces in the image and the image resolution, but not on the image size. Similarly, the accuracy of face recognition is also affected by the number of faces in the image. However, in both cases, the Intel multi-core CPU process was faster than the smartphone because of its enhanced hardware resources. Indeed, the face detection on the CPU was achieved within 0.33 seconds for 1344 x 1521 images, which is quite fast but does not satisfy real-time constraints. In [37], results based on comparison between the Intel core i3 CPU at 2.67GHz clock and the AMD 6500 GPU showed that GPU processor parallel implementation resulted in approximately 50 times faster processing than the CPU. Similarly, in [45] experimental results performed using AMD Radeon HD 7650 GPU and Intel core i3 CPU for 4096 × 4096 image showed that GPU achieved 374x speed up compared to the CPU. In [48], AMD 6500 GPU was compared against Dual-core CPU and achieved 30x speed up for 1024 x 1024 image size.

In conclusion, similarly to PCA, LBP algorithm was used for either face detection or recognition. It features low-level computation tasks, making it suitable to be hosted in highly parallel architectures. It is also suitable for applications that do not require large databases, where IoT or edge processors can be suffice.

3) SCALE-INVARIANT FEATURE TRANSFORM (SIFT)

Scale-Invariant Feature Transform (SIFT) has proven to be a powerful technique for object detection and recognition, and recently it has been used for face recognition [49]. SIFT features are extracted from reference images and stored in a database [50]. The SIFT algorithm consists of four main stages: scale-space extrema detection, key-point localization, orientation assignment, and key-point description [49]–[51]. The scale-space extrema detection consists of searching to all the scale and image locations. This is done by implementing the Difference-of-Gaussian (DoG) function, which identifies the potential interest points, which are called key-points in the SIFT framework. The convolution of a variable-scale Gaussian $G(x,y,\sigma)$, with the input image, $I(x,y)$ produces the scale space function which can be defined as follows:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{4}$$

where,

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \tag{5}$$

In the key-point location phase, for each key-point candidate location, a model is fitted to compute information about the location and scale. These key-points are chosen based on the measures of their stability. In the orientation assignment phase, each key-point is assigned to one or more orientations according to local image gradient directions. Finally, the key-point descriptors are computed by measuring the local

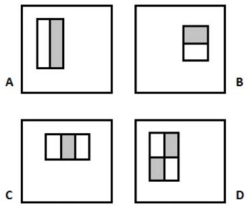

FIGURE 4. Haar features in viola-Jones [38].

image gradients at the selected scale around each key-point. The local gradients are transformed into a representation that allows the change of local shape distortion and illumination significantly. Hence, all the above phases correspond to low-level image processing tasks which are very well suitable to be implemented into FPGA and GPU platforms to handle fine grain-level parallelism.

In [52], the SIFT algorithm achieved an accuracy of 86% for a data set containing 639 faces which was split into 312 training and 327 test tracks. Yet, the system did not perform well when there was illumination, rotation, or blurriness in the image. In a similar work, [51] have also used the same algorithm along with Speed Up Robust Features (SURF) [53]. Two novel detector-descriptor variants were proposed; (1) SURF detector-SIFT descriptor and (2) SIFT detector-SURF descriptor. Experiments on two LFW and Face94 benchmarks resulted in 78% to 90% recognition accuracy. In [54] an extension to SIFT algorithm was suggested. The authors introduced a combination of dense SIFT and Fisher Vector to achieve a higher recognition accuracy. The system used Viola-Jones for face detection and achieved an accuracy of 93%, which is higher than other implementations using SIFT algorithm alone.

With regard to SIFT-based hardware accelerators for face detection/recognition not many implementations were suggested in the literature, probably due to its relatively higher complexity compared to PCA and LBP algorithms for instance. One can list the work done recently in [55], where each step of the SIFT algorithm was effectively parallelized using NVIDIA's GTX 480 GPU for feature extraction. Shared and texture memories were explored for multithreading to avoid pitfalls which can commonly occur in the GPU's runtime, such as frequent data transfers from the GPU memory to CPU memory, which is very costly. Also, separate horizontal and vertical convolution procedures were simultaneously run to minimize the number of memory and arithmetic operations and make them increase linearly with the kernel size. Compared to Intel's quad-core CPU implementation, the GPU implementation was significantly better by at least 3 times the speed-up to achieve real-time performance for HD video frames (e.g. 1920 x 1080 image frames). With the emergence of high-performance edge processors including GPU and FPGAs, further works on parallel SIFT algorithms into IoT or edge processors is worth to investigate. Furthermore, overcoming the algorithm limitations to deal with illumination, rotation, and blurriness is quite possible such as

applying Fisher vectors after SIFT feature extraction on the detected faces, where 93.1% accuracy on LFW dataset could be achieved in [56].

4) VIOLA-JONES

Viola-Jones algorithm [57], [58] was introduced in 2001 by Paul Viola and Michael Jones [57]. It is one of the most famous techniques for face detection as it can be used for real-time applications and can yield a high training rate [59]. It consists of three major phases: First, instead of pixels, the Viola-Jones algorithm uses rectangular features, which as shown in Fig.4 can be of four different types [59], [60].

The Haar features in Viola-Jones are applied to images, then the sum of pixel values under the bright region is subtracted from the sum of all pixel values under the dark region [59], [60]. However, summing entire image pixels and then subtracting them to get a single value is not efficient, especially for real-time applications. This Haar features calculation has to be done all over the image, and this is approximately around 160000+ features per image [61]. Therefore, AdaBoost is used as the second step to reduce the redundant features. AdaBoost is a machine learning algorithm, that features accurate and fast detection [60]. It is used for feature selection and training classifiers and to identify both relevant and irrelevant features [59]. It assigns '1', if a feature is likely to be a nose or eyes, otherwise, it attributes the value of '0'. It then merges a group of weak classification functions to develop a stronger classifier [61]. Computations are further reduced by cascading during the last and final stage of the algorithm. The cascade classifier efficiently combines many of the features together [61].

With regard to parallel hardware implementations, in [62] a GPU-based face detection hardware accelerator, based on NVIDIA's K40 GPU, for the Viola-Jones algorithm was implemented using the OpenCV library. 20 x 20 search windows were used as detection parameters for live video frames at 24 fps. The paper reveals the capability of the GPU to improve the system performance, by speeding up the execution time by up to x22 over the CPU, which was based on an Inten Xeon core processor operating at 2.4 GHz for 640 x 480 input images. Haar-based face detection using a Raspberry PI hardware platform was recently suggested in [63]. The results indicate a detection accuracy of up to 80% for an image dataset of around 20 faces. It was noticed that the detection accuracy decreases for cases such as wearing glasses or masks, which is not the case with CNN-based algorithms. No information was provided regarding the throughput of the system. The Viola-Jones algorithm which uses Haar features is the default algorithm in the OpenCV library for face detection. In [64] a GPU implementation of this algorithm was suggested using the Geforce920M hardware platform to achieve a total execution time of less than 100 ms. To reduce the access to the slow global memory, the authors suggested using the cached texture memory, the content of which does not change during the kernel execution and while accessing the cached shared memory. It can be concluded that despite featuring high parallelism and a low-level image processing computation model, not too many hardware implementations were suggested for the the Viola-Jones algorithm for IoT/edge processors, which can be indeed very promising.

5) OTHER NON-AI TECHNIQUES FOR FACE DETECTION AND RECOGNITION

In addition to the above algorithms, other techniques were also successfully suggested for face detection and face recognition. This includes the Gabor Filter [65], Gaussian Face, Eigen Distance [66], Linear Discriminant Analysis (LDA), and Kernel Fisher Analysis [67]. For instance, in [68] a face recognition system using a subset of the orthogonal Gabor filter for feature extraction was proposed. The output features are then compressed using Linear Discriminant Analysis (LDA). The system has been tested using 2D datasets; CASIA dataset of 123 subjects, ORL, and Cropped Yale B of 39 subjects, and achieved an average recognition rate of 98.9% which is quite impressive. The low-level image processing featured in Gabor filters has motivated researchers to build dedicated embedded accelerators face detection/recognition. For instance, in [16] an embedded door access control system based on face recognition and using a Xilinx FPGA platform was implemented. The system used a hybrid feature extraction technique which is based on extracting local features from eyes, nose, and mouth using Gabor Filter. About 40 different scales and orientations were applied on the images to determine the Gabor representations. Experiments were done on three datasets; Face 94 of 153 persons, FEL of 100 persons, and ORL of 40 persons. The system achieved high recognition accuracy around 100% on the three regions of the face (eye,nose and mouth) using the Face 94 dataset. Therefore, the results reveal that high recognition accuracy can be obtained when facial images are taken with front pose and minimally changed expressions. In another work, in [56], a face verification system implementation was carried out using Gaussian Face and it obtained a surpassed recognition rate on the LFW dataset. In [69] authors used a combination of Radon Transform to crop facial areas, Gabor filter, LDA for dimensionality reduction, and AdaBoost for face recognition. The system achieved a high accuracy rate on three 3D datasets.

B. NEURAL NETWORK ALGORITHMS FOR FACE DETECTION AND RECOGNITION

An Artificial neural network (ANN) consists of neurons ''nodes'' that are interconnected to each other according to some pre-trained weights. It comprises an input, hidden, and output layers to implement any kind of non-linear relationship, the complexity of which depends on the number of sublayers in the hidden layer nodes as shown in Fig.5. Each node sums the activation values it receives from the previous node and adjusts them based on its transfer function [70]. ANN was successfully used for face detection and recognition, but not both together. Its performance is enhanced further using

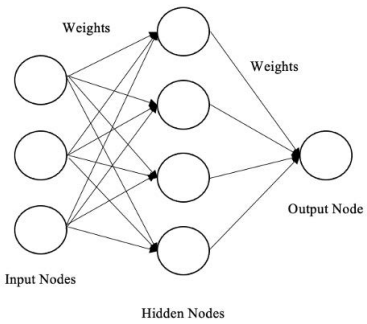

FIGURE 5. Neural network structure.

Convolutional Neural Network (CNN) and Deep Neural Network (DNN) models which simultaneously perform preprocessing, face detection, feature extraction, and classification. The CNN/DNN accuracy can be increased by increasing the number of hidden layers [71].

1) CONVOLUTIONAL NEURAL NETWORK (CNN) FOR FACE DETECTION AND RECOGNITION

A Convolutional Neural Networks (CNNs) [72], [73], is a common form of DNN composed of several CONV layers [73]. In [3], a parallelized CNN algorithm that simultaneously detects and recognize faces in a classroom was demonstrated to achieve higher recognition accuracy on the LFW dataset, of 97.7%, compared to [74]. The proposed system was used for a smart classroom students' attendance using an edge processor (i.e. Nvidia 1800 ti GPU) to detect simultaneously up to 35 faces simultaneously and recognizes 33 out of them. 5 convolution blocks using masks from (11 x 11) to (3 x 3), followed by a ReLU rectification layer and Max pooling, and 3 fully connected layers were used. A similar implementation was done in [75] and achieved recognition accuracy of 97%. However, the system was tested on a small dataset of six persons with 40 pictures per person. In [76] a new unsupervised technique called Local Binary Pattern Network (LBPNet), which is a deep network based on the LBP descriptor was introduced. The idea is attractive as it suggests substituting the time-consuming convolution layers with off-the-shelf computer vision descriptors. This technique has the same topology as CNN and it extracts a hierarchical representation of data. The network is divided into two parts; (i) a deep network for feature extraction and (ii) a regular network for classification. It is composed of two layers that are based on LBP and PCA techniques, respectively. Experiments using public benchmarks (i.e., FERET and LFW) showed that the LBPNet approach achieved high recognition accuracy. In [75], a face recognition system implementation using CNN on Raspberry Pi Model B, could yield a latency of 529ms to detect and recognize a face in an image with 97% accuracy. The Viola-Jones algorithm from the Open CV library was used for face detection on an 8-bit gray scale image, while CNN, implemented in Python, was used for classification. The illumination dependency was reduced by using the LBP algorithm, which generates a 46 x 46 pixels for the CNN algorithm. Even though the database, which consisted of six people for a total of 40 pictures, was relatively small, the system potentially paves the way toward IoT devices for CNN-based face detection. These works also highlight the advantage of combining non-AI with AI algorithms to reduce the computation complexity, while keeping the accuracy high. Very recently, in [77], an FPGA-based implementation for face detection using the MobileNet CNN algorithm was suggested to target an IoT platform. The system, which is based on Intel Aria10Gx 900 and Stratix 10 GX1100 FPGAs, consists of processing high quality images, locally on the edge device and and offloading to the cloud lower quality images, such as images in a crowd to the cloud. A systolic array architecture, which is the natural implementation of the CNN algorithm on FPGA, using matrix tiling and fixed-point precision was used. The advantage of using MobileNet is that it has fewer computation requirements than other standard CNN algorithms as it uses depth-wise separable convolution (DSC). For local processing, the Mobilenet CNN models are stored in off chip memory which are then transferred to the on-chip memory during the runtime. Experimental results show that an impressive throughput of 298 fps and 763.5 fps were achieved on Arria 10 and Stratix 10 FPGAs ICs respectively, for face classification using 97% and 74% of their DSP resources respectively. This high performance was due to the usage of depthwise separable convolution (DSC) and quantized weights with low precision, the fixed point representation. However, the authors did not mention the number of faces that can be processed by the system. In [78], a Single Shot Detector (SSD) implementation which is one type of CNN based face detection algorithms, achieved a throughput of 40 fps and 110 fps on CPU and GPU processors, respectively, using the WIDER Face database [79]. The model memory requirement was about 7.3 MB.

In summary, while CNNs were demonstrated to yield high accuracy, their high computation and memory requirements (e.g. over a billion operations and hundreds of megabytes to store parameters are required for each input image) made it difficult to run on hardware accelerators. While the ideas of combining them with non AI-vision algorithms and to use separable convolutions [80] (e.g. DCS network) were demonstrated to be effective, the emergence of powerful edge processors would make CNN-based IoT and edge devices for face recognition quite possible.

2) MTCNN

The Multi-Task Cascaded Convolutional Neural Network (MTCNN) is a convolution network developed by the Chinese Academy of Sciences [81]. Its architecture consists of three stages of convolutional networks that predict the facial landmarks to detect the faces in an image [2]. The first stage is a proposal network that will predict potential face positions and bound rectangular or elliptical boxes around the predicted locations. This may lead to a large number

of false detections. Therefore, in the second stage, more complex CNN is used, where it refines the predictions from the first stage and eliminates most of the non-face boxes. Similarly, in the last stage more powerful CNN is used and further improvement to the detected faces is done. where five facial landmarks are generated [82]. MTCNN which has been successfully used for face detection in ([2], [83], [84], [85]), features an advantage over FaceNet is that it can simultaneously detect more than one face in an image and feed them to a recognition system. On the other hand, FaceNet can detect only one face at a time [2]. However, one of its major disadvantages is that it cannot cope with face rotations [86]. Nevertheless, several researchers revealed its superior accuracy over Viola-Jones [87]. In [84] a face detection and recognition system was implemented on various hardware platforms that include multicore CPUs Raspberry PI-3+, and NVIDIA's Jetson nano GPU. Two face detection algorithms: Haar feature based and MTCNN were tested and compared. In over 30% of images, the Haar algorithm detected a face, where in fact it is not a face. Whereas, it is only 4% when MTCNN is used. This test concluded that MTCNN provided better results and confidence of detected faces with an accuracy of over 97% with a processing time of 0.75 s and 3.08 s on the Jeston nano and Raspberry PI-3 hardware platforms, respectively. This is longer than FaceNet, where 0.16s and 0.88s were achieved on Jetson nano and Raspberry 3+ platforms respectively. Similarly, in [83], a joint face detection and facial expression recognition system using MTCNN was developed. However, the validation accuracy of facial expression recognition is low, and therefore, further improvement is required to improve the use of the MTCNN algorithm for face recognition. In [86], it was revealed that Viola-Jones is less accurate than MTCNN, MobileNet-SSD and HOG (Histogram of Oriented Gradient) [88] algorithms, where it achieved an accuracy of down to 18.05%. However, it is attractive in the case of simple images since it can achieve real-time performance on mobile devices without the need for hardware accelerators. They also recommend using MobileNet-SSD, since it yields higher accuracy, for real-time face detection. Its implementation on the Jetson Tx1 platform could yield an accuracy of up to 90% and a throughput of 10 FPS using the AFW dataset, which has 205 images with 468 faces in total [89].

Techniques that use sliding windows with handcrafted features such as Haar and HOG methods require less hardware than DNN and CNN algorithms but they are not scalable to handle large number of faces and struggle to perform well in complex scenes such as low/high illumination, face occlusion, and poor image quality.

3) FaceNet

FaceNet is a face recognition system developed by Google [84]. The system as described in detail in [90], extracts high-quality features from the face and predicts a 128-element vector representation called a face embedding. Face embeddings are then mapped to generate a compact Euclidean space, where L2 distances are calculated to measure face similarity [84], [90]. Face recognition becomes then a K-NN classification problem, while face clustering is either k-means or agglomerative clustering. The main advantage of this algorithm is its low memory requirements where only 128 bytes are needed per face. In addition, it requires only a minimal alignment (i.e. a tight crop around the face area. However, its disadvantage is that it can only recognize one face at a time, which requires using a face detection algorithm in case several faces are expected in an image. In [90], a face recognition system based on FaceNet was demonstrated to achieve an impressive accuracy of 99.63% on the LFW dataset. However, the suitability of the algorithm to run on IoT/edge devices was not investigated and was recommended a major future goal by the authors. In [2] a face recognition network cascaded to a face detection network is presented to run on an edge device (x86 platform). The detection is based on the MTCNN algorithm, and the FaceNet algorithm is used to recognize the detected faces. The system could yield a high accuracy on the LFW dataset of 99.03%, slightly higher than the Facenet-only algorithm which yielded an accuracy of 98.71%. The system latency was 57 ms, which was much slower than the cloud Microsoft azure which yielded 893 ms latency. Similarly, in [84] a cascaded detection and recognition network was developed. For face detection, two different algorithms; Haar feature-based and MTCNN were tested, and FaceNet was used for recognition. The proposed systems were implemented on the Rasberry Pi 3B+ and Jetson Nano. The corresponding results showed that the GPU surpass the raspberry pi in terms of processing time at the expense of higher cost. Similarly, using MTCNN for detection and FaceNet for recognition, a performance comparison of the proposed face recognition interface using cloud based (i.e. GTX 1080 (2560-cores), RTX 2080Ti (4352-cores), RTX 2070 (2304-cores) and RTX8000 (4608-cores)) and edge based parallel hardware architectures Jetson Nano (128-cores), Jetson TX2 (256-cores), Jetson Xavier NX (384-cores) and jetson Xavier AGX (512-cores) was conducted in [85]. The corresponding results demonstrated that cloud devices show a largely reduced execution time (2x to 7.7x on average), but the latency time is much higher than edge-based devices. The results also indicate that TensorRT optimization yielded consistently the fastest execution time over TFLite when it runs on edge or cloud devices, but at the expense of 40% larger energy consumption. It was also concluded that FaceNet-TFLite represents the best tradeoff power consumption-memory usage.

V. DISCUSSIONS AND RECOMMENDATIONS

Table 1 and Table 2 summarizes all the aforementioned implementations of face detection and face recognition systems. To the best of our knowledge these tables cover the most recent notable works in this area.

Hence, many research works were successfully conducted in the recent past and they can be categorized into; non-neural network-based algorithms and neural network-based

TABLE 1. Summary of face detection systems.

Algorithm	Ref.	Database	Platform	Accuracy	Performance (Frames/ second)	Comment
DNN/SSD	[91]	FDDB	Raspberry Pi 3B (cortex A53 CPU) + Intel's Movidius NCS2 for neural netowrk	93.25%	11.253 FPS	Modified SSD with MobileNet instead of VGG16
DNN/MTCNN	[84]	Own dataset	1. Raspberry Pi 3B+ 2. Jetson Nano 3.PC+GTX1060		Processing time (seconds): 1. 3.08 (0.32FPS) 2. 0.75 (1.32 FPS) 3. 0.13 (7.6 FPS)	3059 images for 12 different persons
	[85]		1. Cloud-based GPU 1.1 GTX 1080 1.2 RTX 2070 1.3 RTX 8000 1.4 RTX 2080 Ti (Ubuntu 18.04) 1.5 RTX 2080 Ti (Ubuntu 16.04) 2. Edge-Based Devices 2.1 Jetson Nano 2.2 Jetson TX2 2.3 Jetson Xavier NX 2.4 Jetson Xavier AGX		1. Cloud-based GPU 1.1 0.13 (7.62FPS) 1.2 0.11 (9 FPS) 1.3 0.26 (3.81 FPS) 1.4 0.16 = 6.2 FPS 1.5 0.11= 9 FPS 2. Edge-Based Devices 2.1 0.17 (5.8 FPS) 2.2 0.68 (1.5 FPS) 2.3 0.69 (1.5 FPS) 2.4 0.47 (2 FPS)	
Haar-Cascade classifier	[47]	Frontal Face	1. Edge server: Intel Xeon Process (8-core) 2.Smartphone: Cortex-A7 (Dual-core)		Detection time: For 1 to 8 faces 1. Edge server: 0.01(99FPS) to 0.1(10 FPS) seconds approx. 2. Smartphone: 1.7 (0.6) to 6.8 (0.1) seconds approx. Detection time: For different image resolution. 1. Edge: 0.323 (3FPS) seconds (1344 x 1521 Pixels) and 0.152 (6.5FPS) seconds (896 x 592)	
	[84]	Own dataset	1. Raspberry Pi 3B+ 2. Jetson Nano 3.PC+GTX1060		Processing time (seconds): 1. 0.92 (1 FPS) 2. 0.31 (3.2 FPS) 3. 0.06 (16.5FPS)	3059 images for 12 different persons
Local Binary Pattern (LBP)	[39]	MUCT	Altera Cyclone IV DE2-115 FPGA	96.14%	60 FPS	1502 face images
	[46]	Own dataset	1. Edge server: Intel Xeon Process 2.Smartphone: Cortex	95%		Dataset of 226 faces

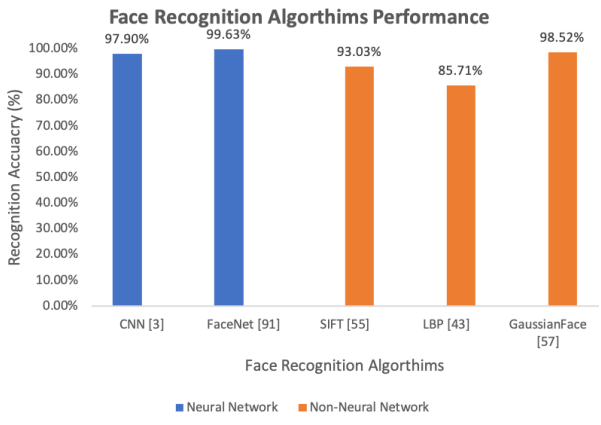

FIGURE 6. Face recognition algorithms performance on LFW dataset.

algorithms. To achieve efficient and acceptable results, the algorithms have to be tested on a good training dataset. In section 3, different algorithms that were tested on different datasets are discussed. However, it is not easy to conduct a fair review comparing different algorithms and their performance if they were not tested on the same dataset. Therefore, to make this review as fair as possible, a comparison between some non-neural network algorithms (SIFT, LBP, Gaussian Face) and neural network algorithms (CNN, FaceNet) that were tested on the LFW dataset is shown in Fig.6. The LFW dataset contains 13233 images of 5749 subjects with a variety of poses, lighting, expressions, age, and other parameters. 1680 subjects of LFW have at least two images each and the rest have one image. Most of these images are colored, and some are in grayscale. As shown, GaussianFace [56], a non-neural network algorithm, achieved a high recognition accuracy result of 98.52%, and as claimed in [56] that for the first time human-level performance in face verification on LFW was surpassed. However, in [90] FaceNet a neural network algorithm was able to achieve a higher recognition accuracy of 99.63% on LFW surpassing all others. The main difference between non-neural network and neural network algorithms is that non-neural network algorithms which are also referred as shallow learning, are based on features decided by humans. That is, after the results have been generated, the strength of the features is decided [3]. Whereas, a neural network which is based on deep learning, is an automated feature

TABLE 2. Summary of face recognition systems.

Algorithm	Ref.	Database	Platform	Accuracy	Performance	Comment
DNN/CNN	[3]	LFW	Nvidia 1080 ti GPU	97.9%		- Parallel algorithm - Challenges: increasing beard, glasses, tiled face.
	[91]	FDDB	Raspberry Pi 3B (cortex A53 CPU) + Intel's Movidius NCS2 for neural network	93 %	7 FPS for detect & recognize	
	[74]	LFW	GAP8 SoC (GAPuino board)	93%		
	[75]	Own dataset	Raspberry Pi, model B	97%	- 2FPS - Detect+recognize= 529 +/- 64ms (2FPS)	6 persons with 40 pictures per person.
	[81]		1. 2.60GHz GPU 2. Nvidia Titan Black GPU	95%	1. 16FPS 2. 99FPS	
	[76]	1. FERET 2. LFW	16 CPU cores	1. 97.8% 2. 94.04%		- LBPNet
DNN/FaceNet	[2]	LFW	X86 general purpose processor (C-RAN)	97% on LFW (MTCNN for detection+ Face Net for recognition)	57ms edge latency (17FPS)	
	[92]	CUFace	NVDIA Jetson TX2	90.29%	5-10 FPS Recognition with tracking: 0.14s	Recognize 8 faces in real-time. Dataset generated by capturing people standing in front of the camera.
	[84]	Own dataset	1. Raspberry Pi 3B+ 2. Jetson Nano 3.PC+GTX1060	97.7%	Processing time (seconds): 1. 0.88 (1FPS) 2. 0.16 (6FPS) 3. 0.02 (50FPS)	3059 images for 12 different persons
	[90]	1. LFW 2. YouTube faces DB		1. 99.63% 2. 95.12%		
	[85]		1. Cloud-based GPU 1.1 GTX 1080 1.2 RTX 2070 1.3 RTX 8000 1.4 RTX 2080 Ti (1) 1.5 RTX 2080 Ti (2) 2. Edge-Based Devices 2.1 Jetson Nano 2.2 Jetson TX2 2.3 Jetson Xavier NX 2.4 Jetson Xavier AGX		1. Cloud-based GPU 1.1 0.09 (11FPS) 1.2 0.05 (20FPS) 1.3 0.07 (14FPS) 1.4 0.08 (12FPS) 1.5 0.05 (20FPS) 2. Edge-Based Devices 2.1 0.86 (1FPS) 2.2 0.49 (2FPS) 2.3 0.47 (2FPS) 2.4 0.27 (4FPS)	cloud devices show a largely reduced execution time (2x to 7.7x on average)
DNN/MTCNN	[83]	FER2013		60.7%		
SIFT algorithm for feature extraction	[52]	TV series Buffy the vampire slayer		86 %		- Parallel algorithm - Do not perform well for rotation, blurriness and change of illumination in the image.
	[51]	1. LFW (200 images of LFW + 40 from Face-94 2. Face94(400 images + 40 Unknown Images from LFW)		78 to 90 %		- SIFT + SURF
	[54]	LFW		93.03%		- Dense SIFT+ Fisher Vector Encoding - 12233 images for 5749 people
	[47]	Frontal Face	1. Edge server: Intel Xeon Process (8-core) 2.Smartphone: Cortex-A7 (Dual-core)		Total Processing time: For 1 to 8 faces 1. Edge server: 0.1 (10FPS) to 0.8 seconds (1FPS) 2. Smartphone: 2 (0.5 FPS) to 9 seconds (0.1FPS)	
	[93]	MUCT	Raspberry Pi-3 board (Quad-Core)		Detection + recognition: 6.8 (1.5 FPS) to 7.8 seconds (0.1FPS)	- used Haar cascade for face detection and LPPH for recognition - 10 individuals with 15 images each - Haar is claimed to be fastest for detection
	[40]	1. CF1999 2. TDF 3. CF: TDF + CF1999 4. LFW		1. 13.03% 2. 5% 3. 8.64% 4. 90.95%		1. 447 imager of 27 unique people. 2. 40 subjects 4. 13233 images
	[41]	1. YaleB 2. YaleB Extended		1. 98.4% 2. 94.6%		LBP+ VLC (Vinder Lugt Correlator) + DoG (difference of Gaussian) - 10persons in YaleB and 38 persons in YaleB Extended with 64 different illuminations for each.
	[42]	1.CMU PIE 2.LFW	Intel Core i5-2430M CPU	1. 99.26% 2. 85.71%		1. 40,000 images for around 68 different people 2. 13,233 web-collected images for 5749 different identities

TABLE 2. ***(Continued.)*** **Summary of face recognition systems.**

Local Binary Pattern (LBP) for feature extraction[45]	[44]	1. CMU-PIE 2. Extended YaleB 3. O2FN		1. 95.75% 2. 98.71% 3. 98.4%		**Relaxed LBP** 1. 40000 facial images of 68 subjects 2. contains 38 subjects 3. contains 2000 face images for 50 subjects
	[45]	ORL	1. AMD Radeon HD 7650 GPU 2. Intel Core i3 CPU	93%	GPU (AMD Radeon HD 7650M) outperforms CPU based face recognition system using LBP in terms of speed and accuracy.	400 images corresponding to 40 individuals
	[48]	ORL	1. AMD 6500 GPU 2. Core 2 Duo	1. GPU: 74%-79% 2. CPU: 74%-79%	GPU (512x512 image size) feature Ext: 36.3ms (27FPS) CPU (512x512 image size) feature Ext: 545.6 ms (1.8FPS)	400 images corresponding to 40 individuals
	[37]	ORL	1. Intel Core i3 CPU 2. AMD 6500 GPU	1. CPU recognition accuracy: 85.4% 2.GPU recognition accuracy: 85.3%	- GPU parallel implementation is about 50 times faster than the counterpart on CPU. -GPU is slightly lower accuracy than CPU implementation because of lack of support of double precision FP.	400 images corresponding to 40 individuals
	[46]	1. Dataset I: No image blending 2. Dataset III		1. 90.49% 2. 99%		**Improved LBP** 1. around 400 faces 2. around 760 faces
PCA	[33]	ORL	Black Gold AX515 (Cyclone IV) FPGA	95.8%	9960 FPS	Only 8 people were tested.
	[94]	1. The Yale Face Database 2. Self-created dataset of 40 images	AMD Radeon Graphics Processor (GPU)		0.2683 seconds (4FPS)	
PCA/Eigen faces	[35]		Raspberry Pi-2 board	88% for detection + recognition		Low computation demanding
	[34]	1. Yale 2. Biold Face	1. GeForce GTX 770 GPU 2. intel ® Xenon(R) CPU		Detection: 109 frames/s Recognition for (650x400): GPU: 127ms (8FPS) CPU: 279 ms (4FPS)	1. 5760 images of 10 different individuals 2. 1521 images of 23 different person
	[36]	ORL	Raspberry Pi 3 Model B+ (Quad-Core)	82.5%	0.206 seconds (5FPS)	400 images corresponding to 40 individuals
Gabor Filter (parallelized)	[68]	1. CASIA 2. ORL 3. Cropped YaleB		1. 99.5% 2. 97% 3. 99.16%		- Gabor filters to extract features and LDA. - using 25 orthogonal filters. 1. 123 persons, with 30 images each. 2. 40 subjects, with 10 images each. 3. Total of 2535 images of 39 subjects.
	[16]	1. Faces 94 2. FEL 3. ORL	FPGA (Xilinx system Generator)	1. 70%- 97% 2. 97.77%-100% 3. 77%-87% (Based on the region)		- Gabor filters + Three segment facial region. 1. total of 153 individuals 2. 100 persons 3. 400 images corresponds to 40 people
GaussianFace	[56]	LFW		98.52%		
Radon Transform + Symbolic LDA + AdaBoost Learning	[69]	Bosphorus, CASIA, Texas		99.65 %		4096 images corresponding to 123 subjects of the three datasets.
Eigen Distance	[66]		Intel Pentium @2.4GHz	92%		
Gabor Kernel Fisher Analysis (KFA)	[67]	1. Yale B 2. CMU-PIE		1. 99.1% 2. 99.6%		

extraction, whereby it will automatically select the best features to predict the output. This difference makes neural network algorithms highly suited for accurate face recognition. Another strength is that neural network or CNN-based algorithms rarely require the use of preprocessing techniques and feature extraction. For example in [90], only a minimal alignment which is a tight crop around the face was required for the image before the recognition process. Unlike other algorithms, CNN based algorithms are known to have high scalability and they still achieve high recognition accuracy on large-scale datasets. Parkhi *et al.* [95] achieved a recognition accuracy of 97.3% using CNN on a dataset of 2.6M images of 2.6K people. However, one drawback of a neural network is that it requires a large number of samples to train the system [96].

To achieve a high-performing face detection and recognition system, a suitable hardware accelerator has to be selected for the system. Similarly, a fair review cannot be conducted to conclude which hardware accelerator has the best performance for face detection and recognition systems as other factors can affect the hardware accelerator performance like the used algorithm, dataset, number of faces, and image resolution. For the detection task, three hardware accelerators; namely the Raspberry Pi 3B+, Jetson Nano GPU, and GTX1060 GPU were used to host two different algorithms [84]. The Raspberry Pi in the system

did not exceed 1 FPS and the other two processors have outperformed both algorithms. However, in [91] the use of hybrid accelerators has decreased the detection time and improved the Raspberry Pi processing time performance. In the recognition process, the raspberry pi has an average of 1.5 FPS processing rate when neural network algorithms are used [75], [84]. However, the average value is exceeded in [91] and [36]. In [91] the Raspberry Pi is used along with Intel Movidius which improved the processing time performance. Whereas, in [36] a 5 FPS processing rate was achieved using the non-neural network PCA algorithm. Accordingly, the Raspberry Pi accelerator could be more suitable for non-neural network algorithms than neural network algorithms when it's used alone.

In comparing the fixed hardware accelerators, GPU and multi core-CPU, it is proved in [47] that there is a proportional relationship between the number of cores and processing rate. Therefore, GPU's performance surpasses CPUs' due to their large number of cores and due to the fact that these algorithms require fine-grain parallelism. Papers [34], [37], [45], and [48] have compared the performance of both accelerators and all concluded that a GPU outperformed multi-core CPUs in terms of accuracy and processing time, since the GPU provides fine-grain parallelism compared to multi-core CPUs which features coarser grain parallelism. Papers [84] and [85] have compared the processing time for cloud-based and edge-based GPUs in detection and recognition systems. As known, cloud-based GPUs provide a much higher processing rate than edge-based GPUs. However, cloud computing is associated with problems such as high response latency, bandwidth limitation, higher risk of information leakage, and privacy exposure. This makes cloud-based GPUs unsuitable to be used for time-sensitive applications.

It can also be concluded that FPGAs are very attractive in terms of power and execution time. However, they cannot completely substitute GPUs and multi-core CPUs. A heterogenous system-on-chip that includes these three technologies seems to be a reasonable way forward. For instance, GPUs are easier to program for complex tasks, especially when the power consumption is not an issue and an FPGA core can accelerate critical datapath intensive tasks. This requires a user-friendly hardware-software co-design framework to seamlessly develop hardware algorithms for face detection and recognition.

VI. CONCLUSION

Face detection and face recognition systems are considered to be challenging and are among the most studied problems in the computer vision field. These systems are commonly used in many real-world applications, such as security, surveillance, and user authentication. A lot of research works on designing new associated algorithms were successfully conducted in the recent past, and they can be typically categorized into non-neural network algorithms and neural network algorithms. This paper discusses the recent implementations of face recognition systems and their performance. A comparative study between face recognition algorithms and platforms in terms of accuracy and processing time was conducted. We can conclude that neural network-based algorithms outperform non-neural network algorithms in terms of accuracy, and GPUs outperform other hardware accelerators in face detection and recognition systems. It can also be concluded that while GPU-based edge or IoT platforms yield very high performance in terms of processing throughput, FPGA-based platforms yield lower power consumption and even higher throughput in the case of a relatively small database of faces. This suggests probably embedding both technologies in the same chip and working on designing a suitable hardware-software development tools that allow the user to partition the algorithm into either technology in a user-friendly manner.

REFERENCES

[1] S. S. Harakannanavar, P. C. Renukamurthy, and K. B. Raja, "Comprehensive study of biometric authentication systems, challenges and future trends," *Int. J. Adv. Netw. Appl.*, vol. 10, no. 4, pp. 3958–3968, 2019, doi: 10.35444/IJANA.2019.10048.

[2] Y. Pan, X. Peng, X. Lin, and M. Xia, "Prototype development of face and speaker recognitions based on edge computing," in *Proc. Int. Symp. Netw., Comput. Commun. (ISNCC)*, Oct. 2020, pp. 1–5, doi: 10.1109/ISNCC49221.2020.9297243.

[3] M. Z. Khan, S. Harous, S. U. Hassan, M. U. G. Khan, R. Iqbal, and S. Mumtaz, "Deep unified model for face recognition based on convolution neural network and edge computing," *IEEE Access*, vol. 7, pp. 72622–72633, 2019, doi: 10.1109/ACCESS.2019.2918275.

[4] N. H. Barnouti, S. S. M. Al-Dabbagh, and W. E. Matti, "Face recognition: A literature review," *Int. J. Appl. Inf. Syst.*, vol. 11, no. 4, pp. 21–31, Sep. 2016, doi: 10.5120/IJAIS2016451597.

[5] Y. Kortli, M. Jridi, A. A. Falou, and M. Atri, "Face recognition systems: A survey," *Sensors*, vol. 20, no. 2, p. 342, Jan. 2020, doi: 10.3390/S20020342.

[6] L. Li, X. Mu, S. Li, and H. Peng, "A review of face recognition technology," *IEEE Access*, vol. 8, pp. 139110–139120, 2020, doi: 10.1109/ACCESS.2020.3011028.

[7] M. P. Beham and S. M. M. Roomi, "A review of face recognition methods," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 27, no. 4, pp. 1–35, 2013, doi: 10.1142/S0218001413560053.

[8] M. K. Hasan, M. S. Ahsan, S. H. S. Newaz, and G. M. Lee, "Human face detection techniques: A comprehensive review and future research directions," *Electronics*, vol. 10, no. 19, p. 2354, Sep. 2021, doi: 10.3390/electronics10192354.

[9] M. O. Oloyede, G. P. Hancke, and H. C. Myburgh, "A review on face recognition systems: Recent approaches and challenges," *Multimedia Tools Appl.*, vol. 79, nos. 37–38, pp. 27891–27922, Oct. 2020, doi: 10.1007/S11042-020-09261-2.

[10] N. H. Barnouti, "Improve face recognition rate using different image preprocessing techniques," *Amer. J. Eng. Res.*, vol. 5, no. 4, pp. 46–53, 2016.

[11] M. A. Rahim, M. S. Azam, N. Hossain, and M. R. Islam, "Face recognition using local binary patterns (LBP)," *Global J. Comput. Sci. Technol.*, vol. 13, no. 4, pp. 1–9, May 2013.

[12] M. Amin, M. Yunus, R. Zainuddin, and N. Abdullah, "A survey: Linear and nonlinear PCA based face recognition techniques," *Int. Arab J. Inf. Technol.*, vol. 13, no. 6, pp. 1–10, Nov. 2013.

[13] A. A. Tamimi, O. N. Al-Allaf, and M. A. Alia, "Eigen faces and principle component analysis for face recognition systems: A comparative study," *Int. J. Comput. Technol.*, vol. 14, no. 4, pp. 5650–5660, Feb. 2015, doi: 10.24297/ijct.v14i4.1967.

[14] Z. Abdi, "Face recognition using independent component analysis algorithm," *Int. J. Comput. Appl.*, vol. 126, no. 3, pp. 34–38, Sep. 2015, doi: 10.5120/IJCA2015906016.

[15] H. J. Seo and P. Milanfar, "Face verification using the LARK representation," *IEEE Trans. Inf. Forensics Security*, vol. 6, no. 4, pp. 1275–1286, Dec. 2011, doi: 10.1109/TIFS.2011.2159205.

[16] Q. Al-Shebani, "Embedded door access control systems based on face recognition," M.S. thesis, Univ. Wollongong, 2014.

[17] A. HajiRassouliha, A. J. Taberner, M. P. Nash, and P. M. F. Nielsen, "Suitability of recent hardware accelerators (DSPs, FPGAs, and GPUs) for computer vision and image processing algorithms," *Signal Process., Image Commun.*, vol. 68, pp. 101–119, Oct. 2018, doi: 10.1016/j.image.2018.07.007.

[18] *An End-to-End Open Source Machine Learning Platform*. Accessed: May 12, 2022. [Online]. Available: https://www.tensorflow.org

[19] *Edge AI Inference Rugged System With 6th Gen Intel Core i5-6442EQ QC and Two Intel MA2485 VPUs*. Accessed: May 12, 2022. [Online]. Available: https://anewtech.net/sites/default/files/pdf/Anewtech-AI-inference-system-AD-AIR-200.pdf

[20] D. Castaño-Díez, D. Moser, A. Schoenegger, S. Pruggnaller, and A. S. Frangakis, "Performance evaluation of image processing algorithms on the GPU," *J. Struct. Biol.*, vol. 164, no. 1, pp. 153–160, Oct. 2008, doi: 10.1016/j.jsb.2008.07.006.

[21] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips, "GPU computing," *Proc. IEEE*, vol. 96, no. 5, pp. 879–899, May 2008, doi: 10.1109/JPROC.2008.917757.

[22] *TensrotFlow*. Accessed: May 20, 2022. [Online]. Available: https://ngc.nvidia.com/catalog/containers/nvidia

[23] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE Access*, vol. 8, pp. 58443–58469, 2020, doi: 10.1109/ACCESS.2020.2983149.

[24] M. Esmaeel and M. Pasandi, "Face, age and gender recognition using local descriptors," M.S. thesis, Univ. Ottawa, Ottawa, ON, Canada, 2014, pp. 1–98.

[25] Ö. Toygar and A. Acan, "Face recognition using PCA, LDA and ICA approaches on colored images," *IU-J. Elect. Electron. Eng.*, vol. 3, no. 1, pp. 735–743–743, 2012.

[26] M. Ü. Çarçkçı and F. Özen, "A face recognition system based on Eigenfaces method," *Proc. Technol.*, vol. 1, pp. 118–123, Jan. 2012, doi: 10.1016/j.protcy.2012.02.023.

[27] M. U. Torun, O. Yilmaz, and A. N. Akansu, "FPGA, GPU, and CPU implementations of Jacobi algorithm for eigenanalysis," *J. Parallel Distrib. Comput.*, vol. 96, pp. 172–180, Oct. 2016, doi: 10.1016/j.jpdc.2016.05.014.

[28] S. Kasap and S. Redif, "Novel field-programmable gate array architecture for computing the eigenvalue decomposition of para-hermitian polynomial matrices," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 3, pp. 522–536, Mar. 2014, doi: 10.1109/TVLSI.2013.2248069.

[29] X. Wang and J. Zambreno, "An FPGA implementation of the Hestenes-Jacobi algorithm for singular value decomposition," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops*, May 2014, pp. 220–227, doi: 10.1109/IPDPSW.2014.29.

[30] S. Zhang, X. Tian, C. Xiong, J. Tian, and D. Ming, "Fast implementation for the singular value and eigenvalue decomposition based on FPGA," *Chin. J. Electron.*, vol. 26, no. 1, pp. 132–136, Jan. 2017, doi: 10.1049/CJE.2016.06.033.

[31] Y. Ma and D. Wang, "Accelerating SVD computation on FPGAs for DSP systems," in *Proc. IEEE 13th Int. Conf. Signal Process. (ICSP)*, Nov. 2016, pp. 487–490, doi: 10.1109/ICSP.2016.7877882.

[32] U. A. Korat and A. Alimohammad, "A reconfigurable hardware architecture for principal component analysis," *Circuits, Syst., Signal Process.*, vol. 38, no. 5, pp. 2097–2113, May 2019, doi: 10.1007/S00034-018-0953-Y.

[33] Z. Wen, M. Wu, and Y. Xie, "Design and implementation of face recognition system based on XilinxZynq-7000," in *Proc. IEEE Int. Conf. Signal, Inf. Data Process. (ICSIDP)*, Dec. 2019, pp. 2019–2022, doi: 10.1109/ICSIDP47821.2019.9173429.

[34] S. Bhutekar and A. Manjaramkar, "Parallel face detection and recognition on GPU," *Int. J. Comput. Sci. Inf. Technol.*, vol. 5, no. 2, pp. 2013–2018, 2014.

[35] C. Gaddam, N. V. K. Ramesh, and H. Dhanekula, "Face recognition based attendance management system with raspberry pi 2 using Eigen faces algorithm," *ARPN J. Eng. Appl. Sci.*, vol. 11, no. 13, pp. 8107–8112, 2016.

[36] I. M. Mohammed, M. Z. N. Al-Dabagh, M. I. Ahmad, and M. N. M. Isa, "Face recognition using PCA implemented on raspberry Pi," in *Proc. 11th Nat. Tech. Seminar Unmanned Syst. Technol.*, vol. 666, 2021, pp. 873–889, doi: 10.1007/978-981-15-5281-6_63.

[37] Z. Guo, J. Han, and J. Chen, "Fast face recognition on GPU," in *Proc. 6th IEEE Int. Conf. Softw. Eng. Service Sci. (ICSESS)*, Sep. 2015, pp. 783–786, doi: 10.1109/ICSESS.2015.7339173.

[38] J. Sun, G. Fan, L. Yu, and X. Wu, "Concave-convex local binary features for automatic target recognition in infrared imagery," *EURASIP J. Image Video Process.*, vol. 2014, no. 1, pp. 1–10, Dec. 2014, doi: 10.1186/1687-5281-2014-23.

[39] K.-Y. Chou and Y.-P. Chen, "Real-time and low-memory multi-faces detection system design with naive Bayes classifier implemented on FPGA," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 11, pp. 4380–4389, Nov. 2020, doi: 10.1109/TCSVT.2019.2955926.

[40] P. Khoi, L. Huu, and V. Hoai, "Face retrieval based on local binary pattern and its variants: A comprehensive study," *Int. J. Adv. Comput. Sci. Appl.*, vol. 7, no. 6, pp. 249–258, 2016, doi: 10.14569/IJACSA.2016.070632.

[41] T. Napoléon and A. Alfalou, "Local binary patterns preprocessing for face identification/verification using the VanderLugt correlator," *Proc. SPIE*, vol. 9094, May 2014, Art. no. 909408, doi: 10.1117/12.2051267.

[42] I. K. Beli and C. Guo, "Enhancing face identification using local binary patterns and K-nearest neighbors," *J. Imag.*, vol. 3, no. 3, p. 37, Sep. 2017, doi: 10.3390/jimaging3030037.

[43] X. Tan and B. Triggs, "Enhanced local texture feature sets for face recognition under difficult lighting conditions," *IEEE Trans. Image Process.*, vol. 19, no. 6, pp. 1635–1650, Jun. 2010, doi: 10.1109/TIP.2010.2042645.

[44] J. Ren, X. Jiang, and J. Yuan, "Relaxed local ternary pattern for face recognition," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2013, pp. 3680–3684.

[45] B. Vinith, M. K. Akhila, N. Naik, and G. N. Rathna, "GPU accelerated face recognition system with enhanced local ternary patterns using OpenCL," in *Proc. Int. Conf. Digit. Image Comput., Techn. Appl. (DICTA)*, Nov. 2015, pp. 1–7, doi: 10.1109/DICTA.2015.7371263.

[46] S. M. Bah and F. Ming, "An improved face recognition algorithm and its application in attendance management system," *Array*, vol. 5, Mar. 2020, Art. no. 100014, doi: 10.1016/j.array.2019.100014.

[47] N. Muslim and S. Islam, "Face recognition in the edge cloud," in *Proc. Int. Conf. Imag., Signal Process. Commun. (ICISPC)*, 2017, pp. 5–9, doi: 10.1145/3132300.3132310.

[48] C. Y. N. Dwith and G. N. Rathna, "Parallel implementation of LBP based face recognition on GPU using OpenCL," in *Proc. 13th Int. Conf. Parallel Distrib. Comput., Appl. Technol.*, Dec. 2012, pp. 755–760, doi: 10.1109/PDCAT.2012.107.

[49] C. Geng and X. Jiang, "SIFT features for face recognition," in *Proc. 2nd IEEE Int. Conf. Comput. Sci. Inf. Technol.*, Aug. 2009, pp. 598–602, doi: 10.1109/ICCSIT.2009.5234877.

[50] V. L. Kumar, B. Spandana, B. Lawanya, M. D. Khaleel, M. Pavani, and B. Aishwarya, "Attendance system by face recognition using sift algorithm and SMS reporting to parents mobile phone," *Int. J. Sci. Develop. Res.*, vol. 2, no. 4, pp. 645–646, 2017.

[51] V. A, D. Hebbar, V. S. Shekhar, K. N. B. Murthy, and S. Natarajan, "Two novel detector-descriptor based approaches for face recognition using SIFT and SURF," *Proc. Comput. Sci.*, vol. 70, pp. 185–197, Jan. 2015, doi: 10.1016/j.procs.2015.10.070.

[52] R. G. Cinbis, J. Verbeek, and C. Schmid, "Unsupervised metric learning for face identification in TV video," in *Proc. Int. Conf. Comput. Vis.*, Nov. 2011, pp. 1559–1566, doi: 10.1109/ICCV.2011.6126415.

[53] G. Du, F. Su, and A. Cai, "Face recognition using SURF features," *Proc. SPIE*, vol. 7496, Feb. 2009, Art. no. 749628, doi: 10.1117/12.832636.

[54] K. Simonyan, O. Parkhi, A. Vedaldi, and A. Zisserman, "Fisher vector faces in the wild," in *Proc. Brit. Mach. Vis. Conf.*, 2013, pp. 1–13, doi: 10.5244/C.27.8.

[55] H. Fassold and J. Rosner, "A real-time GPU implementation of the SIFT algorithm for large-scale video analysis tasks," *Proc. SPIE*, vol. 9400, Apr. 2015, Art. no. 940007, doi: 10.1117/12.2083201.

[56] C. Lu and X. Tang, "Surpassing human-level face verification performance on LFW with GaussianFace," in *Proc. 29th AAAI Conf. Artif. Intell.*, vol. 5, Apr. 2014, pp. 3811–3819.

[57] K. Vikram and S. Padmavathi, "Facial parts detection using Viola Jones algorithm," in *Proc. 4th Int. Conf. Adv. Comput. Commun. Syst. (ICACCS)*, Jan. 2017, pp. 2015–2018, doi: 10.1109/ICACCS.2017.8014636.

[58] Y.-Q. Wang, "An analysis of the Viola-Jones face detection algorithm," *Image Process. Line*, vol. 4, pp. 128–148, Jun. 2014, doi: 10.5201/IPOL.2014.104.

[59] T. Paul, U. A. Shammi, M. U. Ahmed, R. Rahman, S. Kobashi, and M. A. R. Ahad, "A study on face detection using Viola-Jones algorithm in various backgrounds, angles and distances," *Int. J. Biomed. Soft Comput. Hum. Sci., Off. J. Biomed. Fuzzy Syst. Assoc.*, vol. 23, no. 1, pp. 27–36, 2018, doi: 10.24466/IJBSCHS.23.1_27.

[60] A. R. Bhatia, N. M. Patel, and N. C. Chauhan, "Parallel implementation of face detection algorithm on GPU," in *Proc. 2nd Int. Conf. Next Gener. Comput. Technol. (NGCT)*, Oct. 2016, pp. 674–677, doi: 10.1109/NGCT.2016.7877497.
[61] N. T. Deshpande and S. Ravishankar, "Face detection and recognition using Viola-Jones algorithm and fusion of LDA and ANN," *IOSR J. Comput. Eng.*, vol. 18, no. 6, pp. 1–6, 2016.
[62] A. W. Y. Wai, S. M. Tahir, and Y. C. Chang, "GPU acceleration of real time Viola-Jones face detection," in *Proc. IEEE Int. Conf. Control Syst., Comput. Eng. (ICCSCE)*, Nov. 2015, pp. 183–188, doi: 10.1109/ICCSCE.2015.7482181.
[63] R. A. Asmara, M. Ridwan, and G. Budiprasetyo, "Haar cascade and convolutional neural network face detection in client-side for cloud computing face recognition," in *Proc. Int. Conf. Electr. Inf. Technol. (IEIT)*, Sep. 2021, pp. 1–5, doi: 10.1109/IEIT53149.2021.9587388.
[64] H. B. Fredj, S. Sghair, and C. Souani, "An efficient parallel implementation of face detection system using CUDA," in *Proc. 5th Int. Conf. Adv. Technol. Signal Image Process. (ATSIP)*, Sep. 2020, pp. 1–6, doi: 10.1109/ATSIP49331.2020.9231723.
[65] K. Okajima, "Two-dimensional Gabor-type receptive field as derived by mutual information maximization," *Neural Netw.*, vol. 11, no. 3, pp. 441–447, 1998, doi: 10.1016/S0893-6080(98)00007-0.
[66] V. Pande, K. Elleithy, and L. Almazaydeh, "Parallel processing for multi face detection and recognition," in *Proc. 9th Conf. Comput. Robot Vis.*, May 2012, pp. 290–297, doi: 10.1109/CRV.2012.45.
[67] Y. K. Cheong, V. V. Yap, and H. Nisar, "A robust face recognition algorithm under varying illumination using adaptive retina modeling," in *Proc. AIP Conf.*, 2013, pp. 155–164, doi: 10.1063/1.4825007.
[68] S. F. Hafez, M. M. Selim, and H. H. Zayed, "2D face recognition system based on selected Gabor filters and linear discriminant analysis LDA," *Int. J. Comput. Sci. Issues*, vol. 12, no. 1, p. 33, 2015.
[69] P. S. Hiremath and M. Hiremath, "Depth and intensity Gabor features based 3D face recognition using symbolic LDA and AdaBoost," *Int. J. Image, Graph. Signal Process.*, vol. 6, no. 1, pp. 32–39, Nov. 2013, doi: 10.5815/IJIGSP.2014.01.05.
[70] M. M. Kasar, D. Bhattacharyya, and T.-H. Kim, "Face recognition using neural network: A review," *Int. J. Secur. Appl.*, vol. 10, no. 3, pp. 81–100, Mar. 2016, doi: 10.14257/IJSIA.2016.10.3.08.
[71] P. Kalaiarasi and P. E. Rani, "Review on neural networks for face recognition," *Int. J. Sci. Technol. Res.*, vol. 8, no. 10, pp. 2995–3003, 2019.
[72] G. Hu, Y. Yang, D. Yi, J. Kittler, W. Christmas, S. Z. Li, and T. Hospedales, "When face recognition meets with deep learning: An evaluation of convolutional neural networks for face recognition," in *Proc. IEEE Int. Conf. Comput. Vis. Workshop (ICCVW)*, Dec. 2015, pp. 384–392, doi: 10.1109/ICCVW.2015.58.
[73] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017, doi: 10.1109/JPROC.2017.2761740.
[74] M. Zemlyanikin, A. Smorkalov, T. Khanova, A. Petrovicheva, and G. Serebryakov, "512 KiB RAM is enough! Live camera face recognition DNN on MCU," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop (ICCVW)*, Oct. 2019, pp. 2493–2500, doi: 10.1109/ICCVW.2019.00305.
[75] O. Dürr, Y. Pauchard, D. Browarnik, R. Axthelm, and M. Loeser, "Deep learning on a raspberry Pi for real time face recognition," in *Proc. Eurographics*, Jan. 2015, pp. 1–5, doi: 10.2312/EGP.20151036.
[76] M. Xi, L. Chen, D. Polajnar, and W. Tong, "Local binary pattern network: A deep learning approach for face recognition," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2016, pp. 3224–3228.
[77] X. Liu, J. Yang, C. Zou, Q. Chen, X. Yan, Y. Chen, and C. Cai, "Collaborative edge computing with FPGA-based CNN accelerators for energy-efficient and time-aware face tracking system," *IEEE Trans. Computat. Social Syst.*, vol. 9, no. 1, pp. 252–266, Feb. 2022, doi: 10.1109/TCSS.2021.3059318.
[78] B. Ye, Y. Shi, H. Li, L. Li, and S. Tong, "Face SSD: A real-time face detector based on SSD," in *Proc. 40th Chin. Control Conf. (CCC)*, Jul. 2021, pp. 8445–8450, doi: 10.23919/CCC52363.2021.9550294.
[79] S. Yang, P. Luo, C. C. Loy, and X. Tang, "WIDER FACE: A face detection benchmark," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 5525–5533, doi: 10.1109/CVPR.2016.596.
[80] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.
[81] 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.
[82] M. Ma and J. Wang, "Multi-view face detection and landmark localization based on MTCNN," in *Proc. Chin. Autom. Congr. (CAC)*, Nov. 2018, pp. 4200–4205, doi: 10.1109/CAC.2018.8623535.
[83] J. Xiang and G. Zhu, "Joint face detection and facial expression recognition with MTCNN," in *Proc. 4th Int. Conf. Inf. Sci. Control Eng. (ICISCE)*, Jul. 2017, pp. 424–427, doi: 10.1109/ICISCE.2017.95.
[84] T. Lindner, D. Wyrwal, M. Bialek, and P. Nowak, "Face recognition system based on a single-board computer," in *Proc. Int. Conf. Mech. Syst. Mater. (MSM)*, Jul. 2020, pp. 1–6, doi: 10.1109/MSM49833.2020.9201668.
[85] A. Koubaa, A. Ammar, A. Kanhouch, and Y. AlHabashi, "Cloud versus edge deployment strategies of real-time face recognition inference," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 1, pp. 143–160, Jan. 2022, doi: 10.1109/TNSE.2021.3055835.
[86] Y. Guo and B. C. Wunsche, "Comparison of face detection algorithms on mobile devices," in *Proc. 35th Int. Conf. Image Vis. Comput. New Zealand (IVCNZ)*, Nov. 2020, pp. 1–6, doi: 10.1109/IVCNZ51579.2020.9290542.
[87] S. Sarkar, V. M. Patel, and R. Chellappa, "Deep feature-based face detection on mobile devices," in *Proc. IEEE Int. Conf. Identity, Secur. Behav. Anal. (ISBA)*, Feb. 2016, pp. 1–8, doi: 10.1109/ISBA.2016.7477230.
[88] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2005, vol. 1, no. 1, pp. 886–893, doi: 10.1109/CVPR.2005.177.
[89] X. Zhu and D. Ramanan, "Face detection, pose estimation, and landmark localization in the wild," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 2879–2886, doi: 10.1109/CVPR.2012.6248014.
[90] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 815–823, doi: 10.1109/CVPR.2015.7298682.
[91] Y. Xie, L. Ding, A. Zhou, and G. Chen, "An optimized face recognition for edge computing," in *Proc. IEEE 13th Int. Conf. ASIC (ASICON)*, Oct. 2019, pp. 1–4, doi: 10.1109/ASICON47005.2019.8983596.
[92] S. Saypadith and S. Aramvith, "Real-time multiple face recognition using deep learning on embedded GPU system," in *Proc. Asia–Pacific Signal Inf. Process. Assoc. Annu. Summit Conf. (APSIPA ASC)*, Nov. 2018, pp. 1318–1324, doi: 10.23919/APSIPA.2018.8659751.
[93] C. T. J. Prentice and G. Karakonstantis, "Smart office system with face detection at the edge," *Proc. IEEE SmartWorld, Ubiquitous Intell. Comput., Adv. Trusted Comput., Scalable Comput. Commun., Cloud Big Data Comput., Internet People Smart City Innov. (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, 2018, pp. 88–93, doi: 10.1109/SmartWorld.2018.00050.
[94] S. Sapna, R. Anjali, and S. N. Kamath, "Performance analysis of parallel implementation of PCA-based face recognition using OpenCL," in *Proc. 4th Int. Conf. Recent Trends Electron., Inf., Commun. Technol. (RTEICT)*, May 2019, pp. 877–881, doi: 10.1109/RTEICT46194.2019.9016732.
[95] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," in *Proc. Brit. Mach. Vis. Conf.*, vol. 3, 2015, pp. 41.1–41.12, doi: 10.5244/C.29.41.
[96] J. Wang and Z. Li, "Research on face recognition based on CNN," *IOP Conf. Ser., Earth Environ. Sci.*, vol. 170, Jul. 2018, Art. no. 032110, doi: 10.1088/1755-1315/170/3/032110.

ASMA BAOBAID received the B.Sc. degree in electrical engineering from Khalifa University, UAE, in 2020, where she is currently pursuing the M.Sc. degree in electrical and computer engineering.

MAHMOUD MERIBOUT (Senior Member, IEEE) received the B.Eng. and Ph.D. degrees in electronics engineering from the University of Technology of Compiegne, Compiegne, France, in 1985 and January 1995, respectively. He was at Nippon Telegraph and Telephone Corporation, Tokyo, Japan, and NEC Corporation, Tokyo, from 1995 to 2000, where he was involved in several projects related to embedded systems design. In 1998, he received the NTT Best Award for his research and development efforts in the areas of embedded systems design and imaging systems. In 2008, he joined the Electrical Engineering Department, Khalifa University, where he is currently a Full Professor. His current research interests include embedded systems, imaging systems (THz, MPI, optical, electrical, and magnetic tomography), instrumentation, and multiphase flow metering.

VARUN KUMAR TIWARI received the Ph.D. degree from the Visvesvaraya National Institute of Technology, in collaboration with the Indian Institute of Science, Bangalore, India, in 2020, in the area of the IoT and machine learning. He was with Wipro Technologies, Bangalore, from 2019 to 2021, where he worked on multiple cloud and edge-based machine learning projects. He is currently a Research Associate with the Computer Science Department, Khalifa University, UAE. His research interests include distributed systems, sensor systems, and machine learning.

JUAN PABLO PENA received the B.Sc. degree in mechatronics engineering from the National Autonomous University of Mexico, in 2019. Currently, he is pursuing the M.Sc. degree in electrical and computing engineering with Khalifa University, UAE. He has participated in multiple internships in Japan, Canada, and Saudi Arabia, where he worked alongside international teams on creative product development in the areas of VR/AR simulation, visual computing, and machine learning.