International Journal of Computational Methods and Experimental Measurements
Vol. 12, No. 2, June, 2024, pp. 107-119
Journal homepage: http://iieta.org/journals/ijcmem

An Effective Face Detection and Recognition Model Based on Improved YOLO v3 and VGG 16 Networks

Nashwan Saleh Ali[1*], Alaa Fares Alsafo[1], Hiba Dhiya Ali[2], Mustafa Sabah Taha[3]

[1] General Directorate of Education in Ninawa, Ministry of Education, Mousl 41001, Iraq
[2] Ministry of Higher Education, University of Mosul, Mosul 41002, Iraq
[3] Missan Oil Training Institute, Ministry of Oil, Al Emara 62001, Iraq

Corresponding Author Email: nashwansaleh@nan.epedu.gov.iq

https://doi.org/10.18280/ijcmem.120201

Received: 4 December 2023
Revised: 24 February 2024
Accepted: 6 March 2024
Available online: 30 June 2024

Keywords:
face detection, face recognition, VGG 16, YOLO v3, deep learning

ABSTRACT

Face detection and recognition (FRD) technology is a very useful tool that involves taking pictures of people's faces and assessing their biological characteristics to compare and match facial data recorded in databases. Owing to its numerous advantages, including noncontact functionality, time and attendance tracking, medical applications and enhanced security and surveillance, this technology is finding increased application in a variety of contexts. Considering that the face images captured by these devices are influenced by many factors, such as light, posture, and backdrop environment, the recognition rate of current face recognition models remains inadequate. This paper presents a model that combines the You Only Live Once (YOLO) v3 algorithm for face detection with VGG16 networks for efficient face recognition. The model is specifically made to handle scenarios in which people share facial traits and to recognize people in various settings with accuracy. This paper uses two different public datasets to train and test the proposed model, WIDER FACE dataset for YOLO v3 and the Labelled Faces in the Wild (LFW) dataset for the VGG 16 networks, the improved network model performed better in identification and is more robust. Furthermore, the YOLO v3 network scored a little lesser accuracy of 95.9% in face detection, while the VGG 16 network achieved an amazing 96.2% accuracy in face recognition.

1. INTRODUCTION

Face-based detection and recognition represent a significant study area in computer vision given its widespread use in security verification, real-time monitoring, target tracking, image database management, human-computer interaction, etc. Many techniques for face detection have been presented in accordance with the object detection systems, which to varying degrees, have considerably improved face detection accuracy [1, 2]. Finding and localizing faces in pictures or videos is the first stage in face recognition. It is evident that a precise detection technique can improve the functionality of the FDR model significantly, while on the other hand, an ineffective algorithm can have unfavorable effects [3]. In the era of technological renaissance, collecting images has become more convenient owing to the widespread use of mobile and smart devices. However, the computing ability (e.g., feature selection, feature extraction, and image transformation) of such devices is relatively limited [4]. Hence, it is expedient to find faster and more efficient algorithms that may be the ideal approach to addressing this issue.

Early research in the field of image-based FDR focused on the extraction of several kinds of manual features, such as directional gradient histograms [5]. The efficient sorter is then trained to identify and recognize these properties using conventional machine learning techniques. Nevertheless, these techniques usually necessitate that researchers create efficient manual characteristics and refine them separately, which may affect the detection process's overall efficacy. Deep learning DL, particularly deep CNN, has made significant progress in the areas of object identification, image semantic segmentation, and image classification over time [6, 7]. The DL method, in comparison to traditional methods, avoids the need for manual design features. This characteristic enhances the model's adaptive and generalization abilities [8]. As a result, DL-based object detection methods nowadays are extensively used in face detection and recognition.

A decent detection effect can be obtained with a face detection model based on the object detection framework region CNN (RCNN) and DL [9]. The Regional Proposal Network (RPN) and RCNN are combined in the methodology presented in the study by Najibi et al. [10] while Ren et al. [11] introduced the Faster RCNN model, which integrates face detection on many layers of feature images. To enhance the performance of face identification, Dai et al. [12] presented an enhanced framework based on the Region-based Fully Convolutional Networks (R-FCN) model which extracts feature images based on the sensitive properties of face position. YOLO [13], Single Shot Multi-box Detector (SSD) [14], RCNN, and Faster R-CNN, etc. are a few examples of

object detection algorithms based on DL that extract more depth features by designing deeper network structures. Despite this, these algorithms still achieve great levels of accuracy in the detection and classification of objects. Regrettably, these models place a heavy demand on the hardware's processing and storage capabilities and because of the issues with huge models and long detection times, these detection models cannot be widely deployed in embedded systems, and the real-time criteria are frequently unmet. At the same time, a number of circumstances cause the face in the natural landscape to vary substantially. Although it is unable to accurately describe the face, the deep level feature map that was generated from the image often provides high-level semantic information. The final layer of convolution's feature map rapidly contracts in the event of small-scale or overlapped occlusion. This poses significant hurdles for face detection tasks and contributes to the detection model's subpar performance in the aforementioned scenario.

There is a significant issue in image-based FDR systems where various face scales within one image show varying detection performance for the same detector. In addition, various challenges include variations in lighting, facial expressions, backgrounds, facial positions, angles, and distances from the camera. Various research types in image-based FDR aim to address obstacles and enhance accuracy [7, 15]. Therefore, various image-based FDR approaches have been attempting to address these challenges of different scales recently [3, 5]. They have been employing multiple network architectures as a solution to this issue. Another method involves utilizing various levels of features extracted from the network's last few layers. The detection of various facial features in normal scenarios can be achieved in practice. There is no doubt that the features used to detect a face in a 200×200-pixel image are very different from those used to detect a face in a 10×10-pixel image. YOLO v3 [16] utilizes a network structure that is similar to Feature Pyramid Network (FPN) [17] to effectively combine the features from different levels. The YOLO v3 model has achieved cutting-edge outcomes on the COCO dataset [18]; however, the face detection performance did not meet expectations. While the anchor box sizes in YOLO v3 may be appropriate for the COCO dataset, they may not be ideal for detecting faces. This is because face detection only requires locating and detecting faces, without the need to classify numerous objects like the eighty categories in the COCO dataset. Moreover, the majority existing studies commonly assumed that the FDR approaches rely on face images being properly aligned and having a similar pose to the registered images in the gallery. Yet, these assumptions do not hold true in natural settings and real-world scenarios. In real-life scenarios, FDR systems encounter various challenges like varying illumination, facial poses, expressions, backgrounds, angles, and distances from the camera. Therefore, this study proposes a solution to this problem that requires a YOLO v3-based model for face detection, and VGG16 networks for efficient face recognition especially in the case of individuals having similar facial features, and for the purpose of precisely identifying individuals in varying environments. Two different datasets – the WIDER FACE dataset for YOLO v3 and the LFW dataset for VGG16, were used in this study to build the proposed model.

The rest of this article is arranged thus: a review of the existing studies related to DL- face detection and recognition model is presented in Section 2 while the methodology used to achieve the contribution of this study is described in Section 3. Section 5 provides an in-depth analysis of the outcomes obtained by implementing the proposed methodology. Finally, the research conclusion and suggestions for future works are listed in Section 6.

2. LITERATURE REVIEW

Many of the methods used in face detection are derived from those used in object detection, which is a subset of face detection. Prior to deep learning's arrival in this domain, the majority of object detection algorithms relied on manually created features to carry out detection tasks. Scholars have had to design diverse detection models to compensate for the shortcomings of feature representations, which have proven to be insufficient in their ability. In addition, the use of sophisticated diagrams is always necessary to enhance the speed of algorithms. The expressiveness and computational efficiency of the features play a major role in the detectors' performance. The Viola-Jones [19], deformable part model (DPM), and histogram of oriented gradients (HOG) [20] are typical techniques for representative face detectors; the features in these algorithms are manually created. In deep learning, the cascade CNN [21] is widely acknowledged as the original face detector. The system performs face detection using three cascaded CNNs; overlapped bounding boxes are removed using non-maximum suppression (NMS) [22].

Zhang et al. [23] utilized a cascading structure similar to the one mentioned above; however, it goes a step further by predicting the locations of 5 landmarks (nose, eyes, and mouth corners), which helps in achieving more precise face positioning. DenseBox [24] incorporates the use of a full convolutional network (FCN) for face detection [25]. The FaceNet algorithm, proposed by Yang et al. [26] presents a two-stage face detection approach in which attribute-aware networks are used in the first stage for the generation of the response maps for different facial parts. Then, multitask CNN is used to further refine the candidate window generated in the second stage. Despite the significant improvement shown by DL-based face detection systems over traditional methods, they still face challenges when it comes to detecting small-scale and heavily occluded faces, resulting in a noticeable decrease in accuracy. Numerous research studies have been conducted, and various approaches have been proposed to address these problems in unconstrained scenarios. In their study on HR (hybrid resolution), Hu and Ramanan [27] noted the importance of incorporating both large context and scale-variant representations. They used multitask training for various scales and incredibly huge receptive fields to improve the detector's performance.

The performance of SSH [28] is enhanced when used with SSDs. The proposed method suggests utilizing various layers in VGG-net [29] to detect faces of different scales in a multi-branch approach. The authors of FDNet [30] introduced Light-Head Faster R-CNN, a method that enhances face detection performance. This approach incorporates multi-scale training and testing as well as a deformable convolutional neural network. The R-FCN model has a variation called Face R-FCN [31]. In order to tackle the difficult issue of face identification, PyramidBox [32] included a number of components, such as a context-sensitive prediction module, PyramidAnchors, and a low-level feature pyramid network; these components take context information into account. Additionally, to expand the quantity of training samples across

many scales, a technique known as data anchor sampling was introduced.

The present study departs from earlier research on face detection and recognition processes by presenting a new image-based FDR model that addresses a number of issues, including varying illumination, facial poses, expressions, backgrounds, angles, and distances from the camera. This model requires a YOLO v3-based model for face detection and VGG16 networks for effective face recognition, particularly when individuals have similar facial features. In addition, the study aims to precisely identify individuals in a variety of environments. Two distinct datasets, the WIDER FACE dataset for YOLO v3 and the LFW dataset for VGG16, were utilized in this study to construct the suggested model.

3. THE UTILIZED NETWORKS

The YOLO v3 network trained on the WIDER FACE dataset and the VGG16 model trained on the LFW dataset were used in this work for face recognition.

3.1 YOLO v3 network

Deep learning methods have recently been utilized for object detection as they leverage low-level features to create higher-level, more abstract features. DL-based methods represent data hierarchically, thereby enhancing the process of object detection. For multi-classification tasks, the deep learning-based object identification approach performs better than traditional detection algorithms in terms of resilience, accuracy, and speed [31]. Significant advancements have been made in face detection because of the continuous development of convolutional neural networks (CNNs). Modern CNN-based object detectors such as RCNN, SSD, and YOLO have played a crucial role in this progress [33]. These detectors utilize DL techniques to effectively extract image features. However, CNN must construct its own network structure and optimize the weight parameters of the network through training [28, 34].

According to Chen et al. [35], YOLO v3 is an enhanced variant of YOLO & YOLO v2. Being that it is possible to directly estimate bounding box offsets and class probabilities from whole images using a single feed-forward CNN, it is no longer necessary to generate region suggestions or sample features thanks to the YOLO v3 network. This network rather splits the supplied image into S×S-sized smaller grid cells. When an object's center lies inside the grid cell's borders, it will be detected by the grid cell. Each cell in the model predicts the location data for the B bounding boxes and calculates the object degrees that correspond to these bounding boxes [36]. It can obtain the score for each object using Eq. (1):

$$C_i^j = P_{i,j}(Object) \times IOU_{pred}^{truth} \quad (1)$$

where, C_i^j is the abjectness score, (IOU) is the intersection over union, and $P_{i,j}(Object)$ is the object function. The YOLO v3 technique, as a portion of this loss function, makes use of a binary cross entropy of the anticipated and actual object degrees, as expressed in Eq. (2) [36].

$$E_1 = \sum_{i=0}^{S2}\sum_{j=0}^{B} W_{ij}^{obj}\left[C_i^{\wedge j}\log(C_i^j) - (1 - C_i^{\wedge j})\log(1 - C_i^j)\right] \quad (2)$$

Let S2 represent the total number of grid cells. C_i^j and $C_i^{\wedge j}$ represent the predicted degrees. B represents the number of bounding squares, specifically the 4 projections: tx, ty, tw, th of the bounding box. The (CX, CY) refers to a set of grid cells located in the images' upper left end. The coordinates(bx, by), set the midpoint location of the projected final bounding boxes by starting from the image's upper left corner, as explained in Eq. (3). The computation is done thus [36].

$$\begin{aligned} b_x &= \sigma[t_x] + c_x \\ b_y &= \sigma[t_y] + c_y \end{aligned} \quad (3)$$

where, $\sigma()\ is$ a Sigmoid Function. Eq. (4) is used for the determination of the height and width of the projected bounding box.

$$\begin{aligned} b_w &= Pw^{etw} \\ b_h &= Ph^{eth} \end{aligned} \quad (4)$$

As one aspect of the loss function, the YOLO v3 network uses the square error of coordinates prediction [36] as expressed in Eq. (5).

$$\begin{aligned} E_2 = \sum_{i=0}^{s2}\sum_{j=0}^{B} W_{ij}^{obj}\big[&(\sigma(t_x)_i^j - \sigma(t_x^{\wedge})_i^j)^2 + (\sigma(t_y)_i^j \\ &- \sigma(t_y^{\wedge})_i^j)^2\big] \\ &+ \sum_{i=0}^{s2}\sum_{j=0}^{B} W_{ij}^{obj}\big[((t_w)_i^j - (t_w^{\wedge})_i^j)^2 \\ &+ ((t_h)_i^j - \sigma(t_y^{\wedge})_i^j)^2\big] \end{aligned} \quad (5)$$

A. YOLO v3 network architecture

The components of the YOLO v3 network are the residual blocks, skip connections, and up-sampling as this network is a fully convolutional network that uses a feature map and a 1×1 kernel to produce its final output. The detection kernel is (B × (5 + C)) × 1 and the kernel size is 1×1×255. YOLO v3 network operates at a speed of 30 frames per second, which significantly enhances its accuracy when compared to earlier versions [13]. Figure 1 depicts the YOLO v3 network structure.

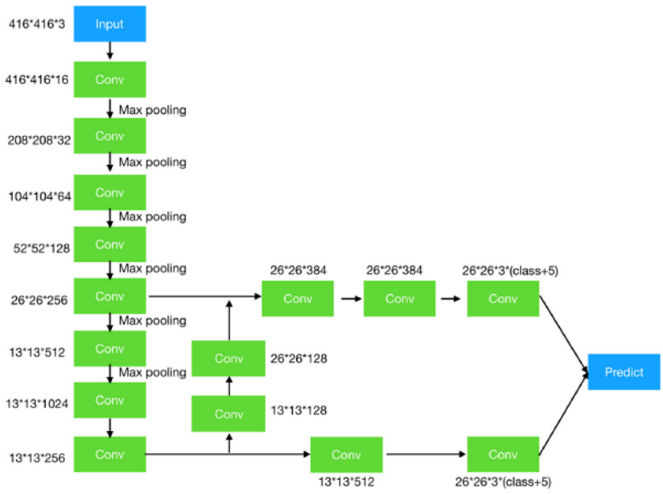

Figure 1. The YOLO v3 network structure [7]

3.2 VGG 16 network

One of the CNNs that is most frequently used for vision-related tasks in DL is the VGG 16 network (Tammina, 2019). One specific implementation of the VGG network architecture is the VGG16 model, which has 16 layers altogether because it has 3 fully linked levels and 13 convolutional layers. On the other hand, the VGG-16 network has a unique architecture. A 224×224 image is used as the standard input for the VGG-16 network. To achieve this standardization, a 224×224 section of each image in the ImageNet collection is cropped from its center. The convolutional filter used in the VGG architecture has the lowest receptive field size of all the filters used, 33. A 1×1 convolutional filter is another feature of the VGG 16 architecture that is used to execute linear modifications on the input. The ReLU, a different linear function, is then used to provide a zero output for negative inputs and a matching output for positive inputs. It has been observed that the use of Local Response Normalization (LRN) in the popular CNN AlexNet leads to longer training durations and more memory use. As a result, according to pertinent literature, LRN is not incorporated into the hidden layers of the VGG architecture. On the other hand, the VGG architecture's hidden layers use the ReLU as their activation function. To further reduce the number of parameters and dimensionality of the feature maps produced after each convolutional process, a sequence of pooling layers is added to the convolution layers. The rising number of filters—which rises gradually from 64 to 128, 256, and finally 512 in the final levels—makes pooling layers necessary. The VGG architecture, which includes the first two tiers, has a total of 4096 channels in each of its three fully linked layers.

Consider the l-th layer in which the inputs form an order 3 tensor $\boldsymbol{x}^l$ with $\boldsymbol{x}^l \in \mathbb{R}^{H^l \times W^l \times D^l}$, then, to locate any specific element in $\boldsymbol{x}^l$ requires a knowledge of the triplet index set (i^l, j^l, d^l). Note that the triplet (i^l, j^l, d^l) represents one element in $\boldsymbol{x}^l$, lying in the d^l-th channel, and at spatial location (i^l, j^l) (at the i^l-th row, and j^l-th column). VGG training requires the use of the mini-batch strategy, thereby making $\boldsymbol{x}^l$ an order 4 tensor in $\mathbb{R}^{H^l \times W^l \times D^l \times N}$ with N being the mini-batch size (taken to be $N = 1$). Furthermore, the zero-based indexing convention which provides that $0 \le i^l < H^l, 0 \le j^l < W^l$, and $0 \le d^l < D^l$, is utilized.

The input $\boldsymbol{x}^l$ is transformed in the l-th layer into an output $\boldsymbol{y}$, with $\boldsymbol{y}$ and $\boldsymbol{x}^{l+1}$ being the same object, while $\boldsymbol{y}$ serves as the next layers' input. The size of the output is taken to be $H^{l+1} \times W^{l+1} \times D^{l+1}$, and a triplet $(i^{l+1}, j^{l+1}, d^{l+1}), 0 \le i^{l+1} < H^{l+1}, 0 \le j^{l+1} < W^{l+1}$, $0 \le d^{l+1} < D^{l+1}$ is used to index an element in the output.

A. The ReLU layer

The input size in the ReLU layer remains unchanged, and the size of $\boldsymbol{x}^l$ and $\boldsymbol{y}$ is the same. Furthermore, parameter learning is not needed; hence, ReLU may be considered a truncation individually performed for each component of the input:

$$y_{i,j,d} = \max\{0, x^l_{i,j,d}\} \tag{6}$$

where, $0 \le i < H^l = H^{l+1}, 0 \le j < W^l = W^{l+1}$, and $0 \le d < D^l = D^{l+1}$.

The above relation shows that:

$$\frac{\mathrm{d}y_{i,j,d}}{\mathrm{d}x^l_{i,j,d}} = \left[\, x^l_{i,j,d} > 0 \,\right] \tag{7}$$

where, $\left[\, x^l_{i,j,d} > 0 \,\right]$ represent an indicator function which is the value 1 for positive argument and 0 for negative argument.

Therefore,

$$\left[\frac{\partial z}{\partial \boldsymbol{x}^l}\right]_{i,j,d} = \begin{cases} \left[\frac{\partial z}{\partial \boldsymbol{y}}\right]_{i,j,d} & \text{if } \boldsymbol{x}^l_{i,j,d} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

Considering that $\boldsymbol{y}$ is an alias for $\boldsymbol{x}^{l+1}$.

Theoretically, Eq. (4) is a little difficult because the function max(0,x) is not differentiable at x=0. In actuality, it is not a problem, and using ReLU is secure.

B. The pooling layer

Assume the input to the l-th layer, which is now a pooling layer, be $x^l \in \mathbb{R}^{H^l \times W^l \times D^l}$. Since the pooling operation requires no parameters (i.e., w^i is 0), there is no need for parameter learning for this layer. The design of the CNN structure specifies the spatial extent of the pooling (H×W). Since the stride equals the pooling spatial extent and H and W divide H^l and W^l, respectively, the output of pooling (y┤ or ├ x^(l+1)) will be an order 3 tensor of size $H^{l+1} \times W^{l+1} \times D^{l+1}$, with:

$$H^{l+1} = \frac{H^l}{H},\ W^{l+1} = \frac{W^l}{W},\ D^{l+1} = D^l \tag{9}$$

A pooling layer works independently on each of the $\boldsymbol{x}^l$ channels. The matrix containing $H^l \times W^l$ elements are subdivided into $H^{l+1} \times W^{l+1}$ non-overlapping sub-regions inside each channel, with each sub-region having a size of $H \times W$. Next, a subregion is translated into a single number by the pooling operator.

The two most popular forms of pooling operators are average and maximum pooling. A sub-region is mapped to its maximum value in max pooling, whereas its average value is mapped to a sub-region in average pooling.

Mathematically,

$$\text{max: } y_{i^{l+1}, j^{l+1}, d} = \max_{0 \le i < H, 0 \le j < W} x^l_{i^{l+1} \times H + i, j^{l+1} \times W + j, d}, \tag{10}$$

$$\text{average: } y_{i^{l+1}, j^{l+1}, d} = \frac{1}{HW} \sum_{0 \le i < H, 0 \le j < W} x^l_{i^{l+1} \times H + i, j^{l+1} \times W + j, d}, \tag{11}$$

where, $0 \le i^{l+1} < H^{l+1}, 0 \le j^{l+1} < W^{l+1}$, and $0 \le d < D^{l+1} = D^l$.

Since pooling is a local operator, calculating it forward is not too difficult. The focus is now on backpropagation. We can go back to the indicator matrix and consider only max pooling. All that is required to be encoded in this indicator matrix is the source of each element in y (x^l). To identify one element in the input x^l, a triplet (i^l, j^l, d^l) is needed, and another triplet $(i^{l+1}, j^{l+1}, d^{l+1})$ is needed to locate one element in y. It is determined that $x^l_{i^l, j^l, d^l}$ can only yield the pooling output $y_{i^{l+1}, j^{l+1}, d^{l+1}}$ if and only if the following criteria are satisfied:

i. They co-exist in the same channel;
ii. The (i^l, j^l) spatial entry belongs to the (i^{l+1}, j^{l+1})-th subregion;
iii. The (i^l, j^l) spatial entry is the largest in that sub-region.

These conditions can be translated as:

$$d^{l+1} = d^l \tag{12}$$

$$\left\lfloor \frac{i^l}{H} \right\rfloor = i^{l+1}, \left\lfloor \frac{j^l}{W} \right\rfloor = j^{l+1}, \tag{13}$$

$$\begin{gathered} x_{i^l, j^l, d^l} \geq y_{i+i^{l+1}\times H, j+j^{l+1}\times W, d^l}, \\ \forall 0 \leq i < H, 0 \leq j < W, \end{gathered} \tag{14}$$

where the floor function is represented as $[\cdot]$.

Eq. (11) needs to be adjusted if the stride is not $H(W)$ in the vertical (horizontal) direction.

The only (i^l, j^l, d^l) triplet that meets all of these requirements is given a $(i^{l+1}, j^{l+1}, d^{l+1})$ triplet. So, we establish an indication matrix.

$$S(\boldsymbol{x}^l) \in \mathbb{R}^{(H^{l+1}W^{l+1}D^{l+1})\times(H^lW^lD^l).} \tag{15}$$

A row in S is indicated by one triplet of indexes $(i^{l+1}, j^{l+1}, d^{l+1})$, whereas a column is indicated by (i^l, j^l, d^l). Together, these two triplets identify one element in $S(\boldsymbol{x}^l)$. If Eqs. (10) through (12) are satisfied concurrently, we set that element to 1; if not, we set it to 0. One element in $\boldsymbol{y}$ corresponds to one row of $S(\boldsymbol{x}^l)$, and one element in $\boldsymbol{x}^l$ corresponds to one column.

With this indicator matrix, it implies that:

$$\text{vec}\,(\boldsymbol{y}) = S(\boldsymbol{x}^l)\text{vec}\,(\boldsymbol{x}^l) \tag{16}$$

Meaning that,

$$\begin{gathered} \frac{\partial\,\text{vec}(\boldsymbol{y})}{\partial(\text{vec}(\boldsymbol{x}^l)^T)} = S(\boldsymbol{x}^l), \\ \frac{\partial z}{\partial(\text{vec}\,(\boldsymbol{x}^l)^T)} = \frac{\partial z}{\partial(\text{vec}\,(\boldsymbol{y})^T)} S(\boldsymbol{x}^l) \end{gathered} \tag{17}$$

Therefore,

$$\frac{\partial z}{\partial \text{vec}\,(\boldsymbol{x}^l)} = S(\boldsymbol{x}^l)^T \frac{\partial z}{\partial \text{vec}\,(\boldsymbol{y})} \tag{18}$$

$S(\boldsymbol{x}^l)$ is highly sparse. Every row has exactly one non-zero element. As a result, the computation does not require using the complete matrix. All that's left to do is locate those non-zero entries, which are limited to $H^{l+1}W^{l+1}D^{l+1}$ in $S(\boldsymbol{x}^l)$.

These equations' meanings can be understood by looking at a basic example. Now let's look at a 2×2 max pooling with stride 2. With $(i, j) = (0,0), (1,0), (0,1)$ and $(1,1)$, the first spatial subregion for a given channel d^l has four elements in the input. Assume the element at a spatial position $(0,1)$ is the largest of them. During the forward pass, the element in the $(0,0,d^l)$-th element in the output y_(0,0, *dl*) will be assigned the value indexed by $(0,1,d^l)$ in the input (i.e., $x^l_{0,1,d^l}$).

If the strides are H and W, respectively, then at most one nonzero element can be found in one column of $S(\boldsymbol{x}^l)$. Considering this example, the column of $S(\boldsymbol{x}^l)$ indexed by $(0,0,d^l), (1,0,d^l)$ and $(1,1,d^l)$ are all zero vectors. The column that corresponds to $(0,1,d^l)$ only contains one non-zero entry and the row index of this entry is determined by $(0,0,d^l)$. Therefore, the output of the backpropagation will be:

$$\left[\frac{\partial z}{\partial \text{vec}\,(\boldsymbol{x}^l)}\right]_{(0,1,d^l)} = \left[\frac{\partial z}{\partial \text{vec}\,(\boldsymbol{y})}\right]_{(0,0,d^l)} \tag{19}$$

and

$$\begin{aligned} \left[\frac{\partial z}{\partial \text{vec}\,(\boldsymbol{x}^l)}\right]_{(0,0,d^l)} &= \left[\frac{\partial z}{\partial \text{vec}\,(\boldsymbol{x}^l)}\right]_{(1,0,d^l)} \\ &= \left[\frac{\partial z}{\partial \text{vec}\,(\boldsymbol{x}^l)}\right]_{(1,1,d^l)} = 0 \end{aligned} \tag{20}$$

However, one element in the input tensor may be the largest element in many pooling sub-regions if the pooling strides are lower than H and in the vertical and horizontal directions, respectively. As a result, more than one nonzero entry can be found in a single column of $S(\boldsymbol{x}^l)$. In two pooling regions, $\begin{bmatrix}5 & 6\\ 8 & 9\end{bmatrix}$ and $\begin{bmatrix}6 & 1\\ 9 & 1\end{bmatrix}$, element 9 is the largest if a 2×2 max pooling is applied to it and the stride is 1 in both directions.

As a result, there are two nonzero entries in the column of $S(\boldsymbol{x}^l)$ that correspond to element 9 (indexed by $(2,2,d^l)$ in the input tensor, whose row indexes are $(i^{l+1}, j^{l+1}, d^{l+1}) = (1,1,d^l)$ and $(1,2,d^l)$. This example, therefore, leaves us with the relation:

$$\begin{aligned} \left[\frac{\partial z}{\partial \text{vec}\,(\boldsymbol{x}^l)}\right]_{(2,2,d^l)} &= \left[\frac{\partial z}{\partial \text{vec}\,(\boldsymbol{y})}\right]_{(1,1,d^l)} \\ &+ \left[\frac{\partial z}{\partial \text{vec}\,(\boldsymbol{y})}\right]_{(1,2,d^l)} \end{aligned} \tag{21}$$

C. The convolution layers

A convolution kernel is an order 3 tensor of size $H\times W\times D^l$, assuming that the input in the l-th layer is an order 3 tensor of size $H^l\times W^l\times D^l$. The products of the applicable components are computed in each of the D^l channels when the kernel on top of the input tensor overlaps at the spatial point (0, 0, 0). The HWD^l products are then added together to provide the convolution result. After that, the kernel is shifted to finish the convolution from L-R and from top to bottom.

A convolution layer uses a large number of convolution kernels. All of the kernels are designated as f. f, which is an order 4 tensor in $R^{(H\times W\times D^l\times D)}$, assuming that D kernels are employed and that each kernel has a spatial span of H×W. Similar to this, a particular element in the kernels can be located using the index variables $0\leq i<H$, $0\leq j<W$, $0\leq d^l<D^l$, and $0\leq d<D$. The notation w^l in the first equation represents the same object as the set of kernels f. Despite the implementation of the mini-batch approach, the kernels stay the same.

A straightforward padding approach can be used to make sure that the output and input images are of the same dimensions when necessary. If the size of the kernel is $H\times W\times D^l\times D$, and the size of the input is $H^l\times W^l\times D^l$, then, the size of the convolution result would be $(H^l-H+1)\times(W^l-W+1)\times D$. For each input channel, if $\left\lfloor\frac{H-1}{2}\right\rfloor$ rows are inserted on top of the first row and $\left\lfloor\frac{H}{2}\right\rfloor$ rows are added below the last row, and if $\left\lfloor\frac{W-1}{2}\right\rfloor$ columns is added to the left

side of the first column and $\left\lfloor \frac{W}{2} \right\rfloor$ columns is added to the right side of the last input column, then, the output size of the convolution would be $H^l \times W^l \times D$, meaning that it will have an equivalent spatial extent as the input. The floor functions are represented as $\lfloor \cdot \rfloor$. For the padded columns and rows, the elements are often set to 0, though it can assume other values.

A simple case in which the padding was not considered and the stride is 1 has been considered in this section. This leaves us with $\boldsymbol{y}$ (or $\boldsymbol{x}^{l+1}$) in $\mathbb{R}^{H^{l+1} \times W^{l+1} \times D^{l+1}}$, with $H^{l+1} = H^l - H + 1, W^{l+1} = W^l - W + 1$, and $D^{l+1} = D$.

The mathematical expression of the convolution procedure is given as:

$$y_{i^{l+1},j^{l+1},d} = \sum_{i=0}^{H} \sum_{j=0}^{W} \sum_{d^l=0}^{D^l} f_{i,j,d^l,d} \times x_{i^{l+1}+i,j^{l+1}+j,d^l} \quad (22)$$

Note that for all $0 \leq d \leq D = D^{l+1}$ and for any spatial location (i^{l+1}, j^{l+1}), this mathematical equation is repeated, thereby meeting the criterion $0 \leq i^{l+1} < \bar{H}^l - H + 1 = H^{l+1}, 0 \leq j^{l+1} < W^l - W + 1 = W^{l+1}$, where $x_{i^{l+1}+i,j^{l+1}+j,d^l}$ is the element of $\boldsymbol{x}^l$ that has been indexed by the triplet $(i^{l+1} + i, j^{l+1} + j, d^l)$.

Usually, a bias term b_d is inserted to $y_{i^{l+1},j^{l+1},d}$ but for clearer presentation, this has been omitted in this note.

4. THE PROPOSED APPROACH

To accomplish the major aim of this work, which is to build a functional model for facial detection and recognition, a thorough explanation of all the information required will be given in the next subsections. An overview of the YOLO v3 and VGG16 networks, as well as the utilized datasets, are given in this section.

4.1 Dataset description

A suitable dataset must be carefully chosen for the model's training and testing when building a model that can recognize faces. A facial recognition dataset is necessary to carry out this investigation. This is due to the fact that the intended system's goal is to distinguish and identify faces. Two distinct datasets were used for this purpose: the LFW dataset for VGG16 face identification and the WIDER FACE dataset for YOLO v3 face detection.

A. WIDER FACE dataset

This dataset was chosen to be used for training the YOLO v3 network for face detection Online and it is a comprehensive collection of data specifically designed for face-detection tasks [7]. The data was sourced cleansed manually; 32,203 images in the collection have 393,703 face-bounding box annotations. The large range of variables in position, occlusion, lighting conditions, scale, and facial expression in this dataset makes face detection quite difficult. However, the dataset is perfect for training and testing face detection algorithms because of its varied properties. The dataset contains a range of intricate face patterns, including extreme poses, small scales, and severe occlusions. Based on the degree of detection difficulties, the data from WIDER FACE was categorized as "Easy," "Medium," and "Hard." This split was made to provide a more thorough evaluation of the detector's performance. Three subsets comprise the complete database: testing (50%), validation (10%), and training (40%) using random sampling to avoid biasing the results. The dataset is mostly recognized as one of the most common and widely utilized datasets for face detection. Figure 2 depicts sample images of the WIDER FACE dataset.

Figure 2. Sample images of the WIDER FACE dataset

B. Labeled faces in the wild (LFW) dataset

The LFW dataset was used to train the VGG16-CNN for FRD [37]. The well-known dataset consists of 13,233 images of faces that were gathered from several websites. The database is made up of face images taken specially to research unrestricted facial recognition. The University of Massachusetts, Amherst built the database and currently maintains it. A total of 13,233 images of 5,749 people were successfully detected and centered by the Viola-Jones face detector. These images were then gathered from the internet. Among the individuals included in the collection, 1,680 have two or more unique images. Three types of "aligned" images and four separate sets of LFW images make up the original database. Figure 3 depicts sample images of the LFW images dataset.

Figure 3. Sample images of the LFW images dataset

4.2 Data pre-processing

The first phase of the proposed model is initiated once the compatible datasets for both the YOLO v3 and the VGG16 networks are acquired. In this paper, the stage of data pre-processing is responsible for normalization, augmentation, resizing, and transforming images into a format that is most appropriate for each of the networks after data acquisition. The YOLO v3 network is trained using the pre-processed WIDER FACE dataset, while the VGG16 network is trained using the pre-processed LFW dataset. The YOLO v3 network is

designed specifically for face detection, while the VGG16 is focused on performing face recognition. Once both networks have been trained, the performance of the suggested model and its ability to detect and recognize faces are tested using new input data, also known as testing data. The general flowchart of the suggested model is illustrated in Figure 4.

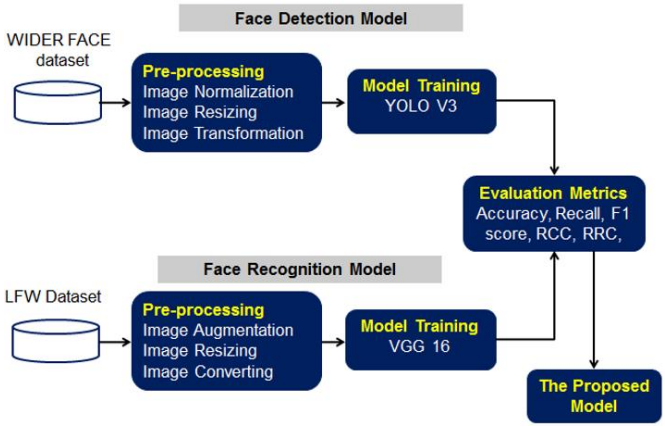

Figure 4. The general flowchart of the proposed model

A. WIDER FACE–based face detection

To improve face detection performance, the YOLO v3 network was re-trained using the WIDER Faces dataset; specifically, the data in the dataset were first normalized [7]. Data normalization is a technique that aids in the faster convergence of the model during training and reduces the likelihood of getting trapped in local optima. Data rescaling is a common practice to adjust data to a standardized scale, typically ranging between 0 and 1 or -1 and 1. This process ensures that the differences in the data ranges are preserved without any distortion. To accomplish this, the mean μ is subtracted from the pixel values in the dataset and then divided by the standard deviation σ (see Eqs. (23) and (24)).

$$p' = (p - \mu) / \sigma \tag{23}$$

where, p = the original pixel value and p' = the normalized pixel value.

The image normalization process is expressed in Eq. (24).

$$normalized\ image = \frac{(image - mean)}{standard\ deviation} \tag{24}$$

Next, the normalized images underwent transformations such as resizing [7] and converting their annotations to a format that is compatible with YOLO [35]. The first step in pre-processing the WIDER Faces dataset is to resize the images to a 224×224 pixel dimension (see Eq. (3)), as this is the optimal dimension size for training the suggested model. Subsequently, each image's annotations are transformed into the YOLO format (Algorithm 1). For every image, a distinct text file with the class label and bounding box coordinates for every annotated face must be created. Python programming and the OpenCV library were used for the pre-processing stages. The class label and normalized bounding box coordinates (x, y, w, h) in relation to the image size are included in each annotation. Samples from the WIDER FACEs Dataset are displayed in Figures 5 and 6 (without and with annotations, respectively).

Figure 5. Random samples without annotations from images folder

Figure 6. Random samples with annotations from images folder

$$image\ resizing = resize\big(image, (224, 224)\big) \tag{25}$$

Algorithm 1. Converting annotations for each image to YOLO format:

<class_label> <x_center> <y_center> <width> <height>

i. Extract the annotations for each image, which typically include bounding box coordinates (x_min, y_min, x_max, y_max), and the object's class label.
ii. Calculate the following values for each annotation:
   a) `x_center`: The x-coordinate of the bounding box center, calculated as `(x_min + x_max) / 2`.
   b) `y_center`: The y-coordinate of the bounding box center, calculated as `(y_min + y_max) / 2`.
   c) `width`: The width of the bounding box, calculated as `x_max - x_min`.
   d) `height`: The height of the bounding box, calculated as `y_max - y_min`.

iii. Normalize the calculated values relative to the image's width and height. Divide `x_center`, `width`, `y_center`, and `height` by the image's width and height, respectively. This ensures that the coordinates are represented as values between 0 and 1.

iv. Concatenate the calculated values and the class label in the YOLO format:
   a) ``<class_label> <x_center_normalized> <y_center_normalized> <width_normalized> <height_normalized>`

v. Write the YOLO-formatted annotations to a text file for each image.

B. LFW–based face recognition

This work utilized the VGG 16 network for the determination of the similarity between two comparable inputs. There is a chance for two different individuals to have similar facial features, thus, it is necessary to implement the VGG 16 to provide the highest accuracy possible in facial recognition. The LFW dataset was chosen to train the VGG16 network because this dataset comprises images of 1680 people that have two or more varying images in the dataset (the purpose becomes to identify the individuals). The LFW dataset is designed to study the problem of unconstrained face recognition in "wild" conditions, which include a range of pose, lighting, and expression variations typically encountered in everyday life.

Data from the LFW dataset must first be pre-processed before being used as input to the VGG 16 network. First, data augmentation [3] is utilized on the LFW dataset; it is a technique that is commonly used to expand the size of a dataset. It involves creating a new set of data based on the existing one, with the goal of increasing the diversity of images. This process can greatly enhance a model's ability to generalize and perform well on a variety of tasks. In this paper, several augmentation operations were applied to avoid the over-fitting when during the training process, these operations are rotation, brightness, flipping and resizing on existing images. The rotation process can be described in Eq. (26).

$$\begin{aligned} x' &= x\cos\theta - y\sin\theta \\ y' &= x\sin\theta + y\cos\theta \end{aligned} \tag{26}$$

where, θ is the angle of rotation, (x,y) are the original coordinates of a pixel, and (x',y') are its new coordinates after rotation.

Another augmentation process, flipping, can be performed on the images either horizontally, vertically, or side-wise. Eq. (27) represents the horizontal flipping:

$$f: (x, y) -> (W - x, y) \tag{27}$$

where, W represents the width of the image.

After the process of image augmentation, the images are resized to a dimension of 224×224 pixels. Moreover, the resized images were converted to greyscale images to decrease the computational complexity and to ensure that the network focuses on the essential features of the faces, rather than color information (see Eq. (28)).

$$Gray\ scale = \frac{(r + g + b)}{3} \tag{28}$$

where,

- Gray is the calculated grayscale value.
- R is the red channel value of the image.
- G is the green channel value of the image.
- B is the blue channel value of the image.

5. RESULTS AND DISCUSSION

This section is dedicated to outlining the outcomes of training the two proposed networks, YOLO v3 and VGG16. After training the networks, they were tested with previously unseen images to assess their accuracy in identifying faces. The evaluation of the models' performance relies on their ability to accurately differentiate between individuals, even when faced with various factors such as different facial expressions, poses, lighting co expression variations, and so on.

5.1 Training environment

Darknet was used to train the proposed image based -FDR model on a PC equipped with Windows 10 Home Premium Edition (64-bit), CPU Intel Core i7, 3.40 GHz, Memory 16 GB, GPU NIVDIA GeForce TITAN X 1060. The experiments utilized several Python libraries and packages, such as Scikit-Learn, Numpy, and Pandas. A batch size of 64 was used and the input image size was set to 416×416. This study made rapid use of the SGD optimizer. After being set at 0.001, the learning rate decreased exponentially every 4000 steps. To improve the model's adaptability and generalizability, different kinds of data augmentation - rotation, brightness, flipping and resizing on existing images were applied. The model was trained for 20,000 steps in total. The LFW dataset and the WIDER FACE validation dataset were used to assess the trained model. Notably, the twelve false annotations found in the WIDER FACE dataset were eliminated before starting the training.

5.2 Evaluation metrics

The suggested model's performance is evaluated by comparing its results to those of the testing datasets. There are several metrics available to evaluate the suggested model's performance, including precision, recall, F1 score, and accuracy. The phrase recall, also known as sensitivity, relates to the genuine positive value. It reflects the proportion of positively categorized inputs that were correctly classified out of all positive inputs that should have been classified positively. Precision is calculated by dividing the total number of positive outcomes by the number of true positive classifications. The F-measure is a metric that merges recall and precision into one value to provide an overall predictive performance measure for a model. It is calculated using the recall and harmonic mean of precision.

$$Recall = \frac{TP}{TP + FN} \tag{29}$$

$$Precision = \frac{TP}{TP + FP} \tag{30}$$

$$F - Measure = 2\,Precision * Recall\,Precision + Recall \tag{31}$$

$$Accuracy = \frac{TN + TP}{TN + TP + FN + FP} \quad (32)$$

True positive is designated by TP. True negative is designate. The next sub-section illustrates the achievements that have been achieved using the proposed YOLO v3 and VGG 16 networks.

5.3 YOLO v3 evaluation process

The YOLO v3 network, as mentioned earlier, was trained on the pre-processed Wider Faces dataset. This training enables the YOLO v3 network to learn the key characteristics required for identifying and recognizing faces in real-life scenarios with various challenges like varying illumination, facial poses, expressions, backgrounds, angles, and distances from the camera. The Precision-Recall (P-R) curve, the Precision Confidence curve, the Recall Confidence curve, the F1 Confidence curve, and the confusion matrix [38-41], were utilized to assess the performance of the proposed image based- FDR model. Figure 7 depicts the P-R curve for the proposed YOLO v3 network model.

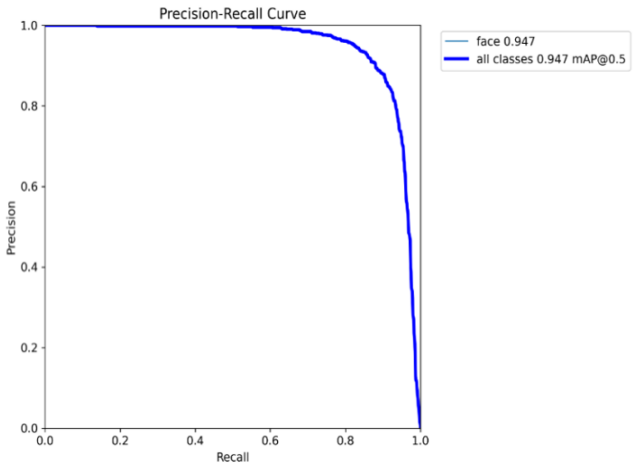

Figure 7. Precision-Recall (P-R) curve for the YOLO v3 network model

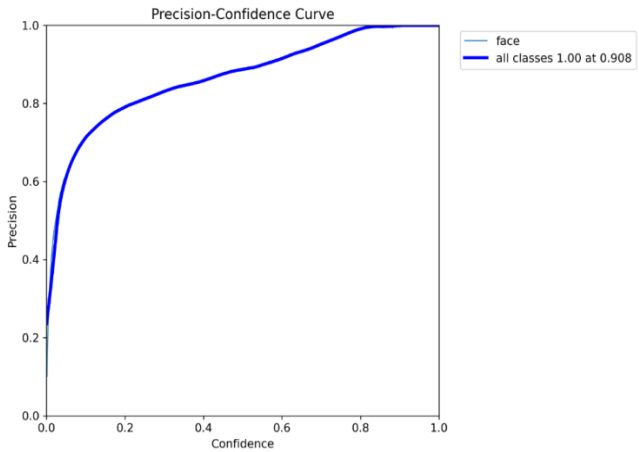

Figure 8. Precision-confidence curve metric for the proposed YOLO v3 network

The Precision-Recall (P-R) curve is a metric that quantifies the relationship between the precision and recall of the YOLO v3 network being used. Precision and recall were calculated using Eqs. (28) and (29) respectively. The performance of the YOLO v3 model was excellent as it achieved an impressive AUC score of around 0.95 which suggests its effectiveness in capturing the difference between false positive and true positives values at different threshold settings. The model also recorded impressive recall values as seen by the obtained precision-confidence curve in Figure 8.

A precision confidence curve with a value of about 1.0 indicates that the proposed YOLO v3 network has a high accuracy. The high number indicates that the model can confidently make correct predictions for facial recognition. Furthermore, it can continue to have a high accuracy rate even if the confidence threshold is changed. Figure 9 depicts the recall-confidence curve for the proposed model in this work.

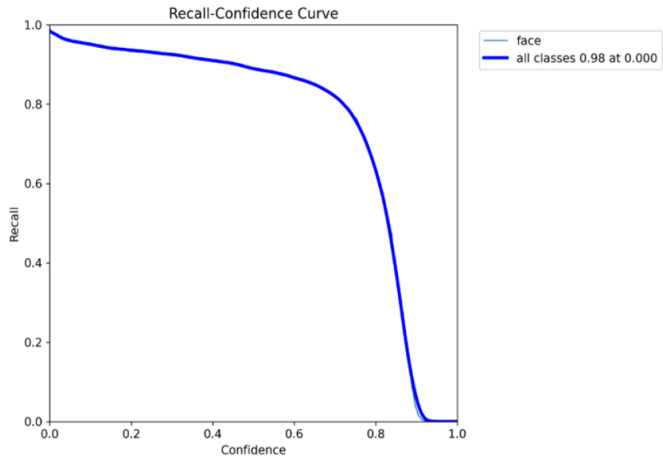

Figure 9. The Recall-confidence curve for the YOLO v3 network

Regarding the recall confidence curve, it shows that the model achieved a high value of approximately 0.98, signifying that the YOLO v3 network can identify the true positive values with a very high confidence. Figure 10 depicts the F1-confidence curve for the YOLO v3 network.

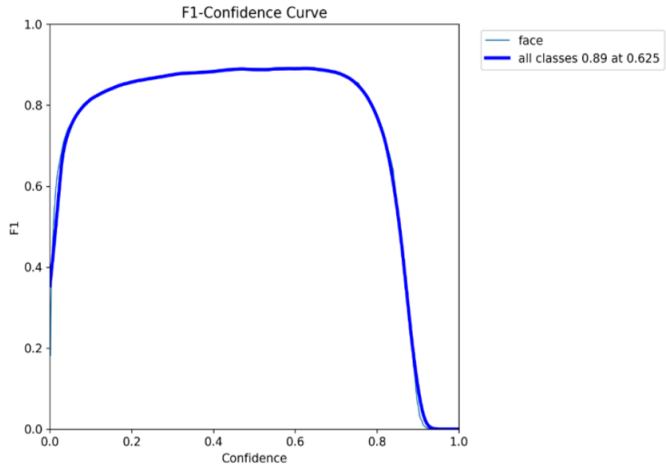

Figure 10. The F1-confidence curve for the YOLO v3 network

The F1 Confidence curve, which represents the combination of precision and recall, displayed a high score of approximately 0.89. This value points to the fact that the model exhibits a good balance between precision and recall, resulting in a consistently high F1 score across various

confidence thresholds. Figure 11 depicts the confusion matrix for the YOLO v3 network.

In addition to the aforementioned metrics, a confusion matrix was generated for the YOLO v3 network. The model demonstrates good results with a high true positive rate (TPR) of approximately 0.93 and a low FPR of around 0.07. The low FPR indicates strong models' ability to accurately identify faces and minimize errors. A visual inspection was also used to assess the performance of the proposed YOLO v3 network. In this context, the terms "validation-batch-label" and "validation-batch-pred" were used to denote the representative images, where the "validation-batch-label" image represents the ground truth labels for the faces in the validation batch. The predictions achieved with the YOLO v3 network are represented by the "validation-batch-pred" image for the same validation batch. The high degree of precision of the model was evidenced by the models' ability to correctly identify every face labelled in the "validation-batch-label" images. The ability of the model to recognize faces in difficult situations was also demonstrated, including the prediction of faces that were not present in the "validation-batch-label" images. The indication of these results is that the proposed YOLO v3 model was robust, consistent, and excellent in facial recognition. The images labeled as "validation-batch-label" and the images predicted as "validation-batch-pred" are respectively presented in Figures 12 and 13.

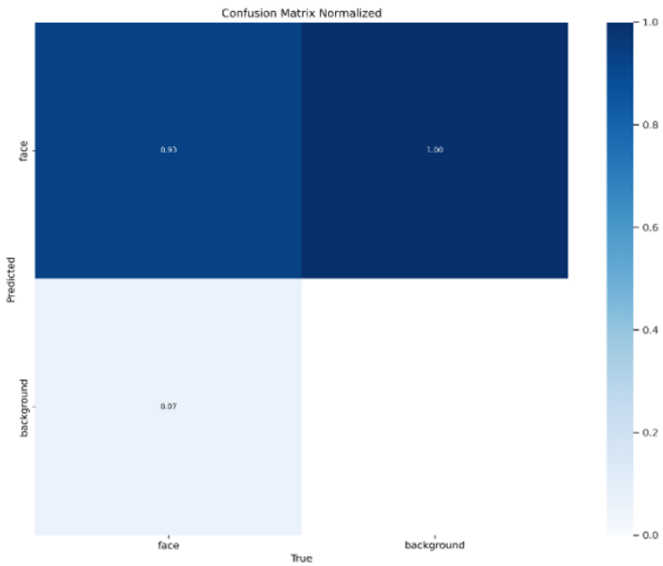

Figure 11. The confusion matrix for the YOLO v3 network

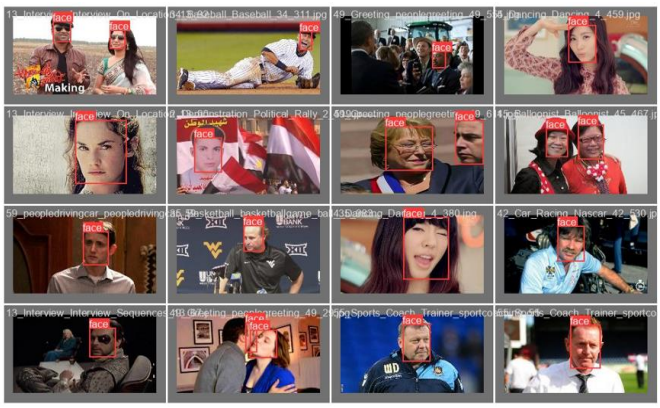

Figure 12. The validation-batch-label image

Figure 13. Validation-batch-pred images

5.4 VGG-16 process

The evaluation of the Siamese networks also provided good results that make the network suitable for face recognition. Figure 14 shows that as the number of training epochs increased, the binary cross-entropy loss decreased, reaching lower values, showing that the network effectively learned the features that distinguish each face from the other, thereby resulting in a minimal difference between the predicted outputs and the actual outputs.

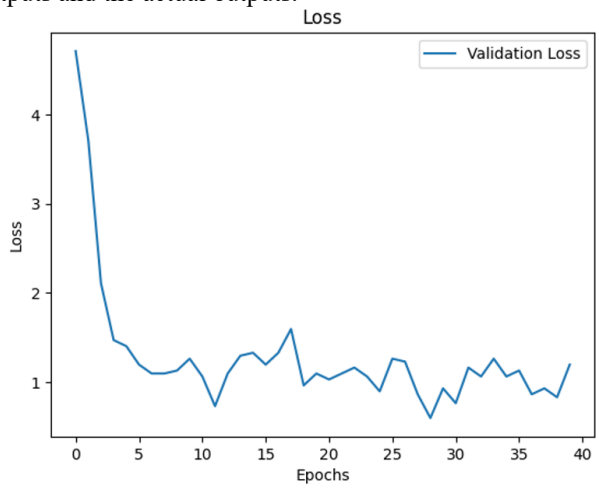

Figure 14. Binary cross entropy loss for Siamese network

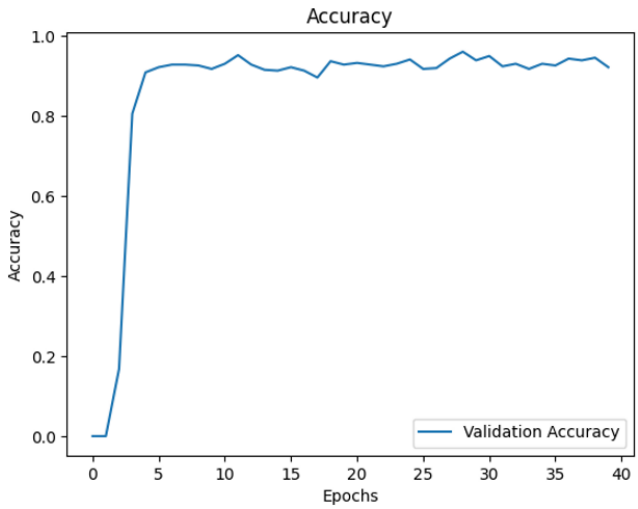

Figure 15. The VGG 16 networks' accuracy

An accuracy curve was also visualized, showing a high accuracy score of 95% approximately. This value signifies the models' ability to classify most of the input images correctly. The high accuracy result renders the network suitable for the task of logging out the employees of a company. Figure 15 depicts the VGG 16 network accuracy.

Once the training of the Siamese network was concluded, it was tested on new test images to see how well the network performed to recognize the faces in real life scenarios. When presented with 3 images of the same individual but with different facial expressions, the network was able to correctly identify the three images as the same person. This demonstrates the network's ability to accurately detect and group similar faces within the dataset. Figure 16 illustrates the test model on random samples.

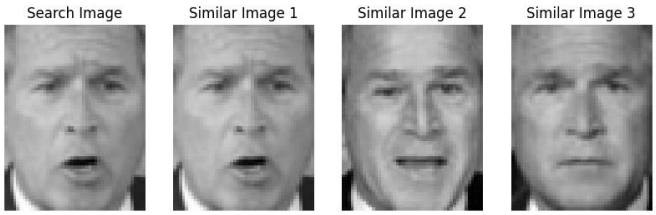

Figure 16. The test model on random samples

The results obtained in this paper demonstrate the effectiveness of the combined model that utilizes the YOLO v3 and VGG 16 networks in recognizing and identifying faces.

5.5 Comparison of face recognition performance

The proposed VGG 16 network for face recognition in this work achieved an accuracy of 96.2% on the test set, which in comparison to other studies, is a better level of accuracy, or at least comparable. For instance, in Table 1, the proposed Siamese network (96.20) scored a much higher accuracy than some methods such as DLB (88.50), CFN+APEM (87.50), L-CSSE-KSRC (92.02), and SiameseFace1 (94.80). Other methods such as weighted PCA-EFMNet (95.00) and Siamese-VGG (95.62) showed good results that are still lower in accuracy compared to the proposed Siamese network. The only method that surpassed the proposed Siamese network in accuracy is CosFace which achieved 99.73% accuracy. However, despite achieving a lower accuracy compared to CosFace, the proposed Siamese network is faster, more lightweight, and suitable for running on simple hardware.

Table 1. VGG 16 network performance in comparison to other methods

#	Method	Face Recognition ACC (%)	References
1	DLB	88.50	[42]
2	CFN+APEM	87.50	[43]
3	L-CSSE+KSRC	92.02	[44]
4	SiameseFace1	94.80	[45]
5	Weighted PCA-EFMNet	95.00	[46]
6	Siamese-VGG	95.62	[47]
	Proposed VGG 16 Network	**96.20**	

Furthermore, Table 2 demonstrates how the proposed YOLO-v3n algorithm achieved significantly better results than other algorithms in hard, medium, and easy scenarios. For instance, when YOLO-v3n scores 0.901, 0.946, 0.959 in the hard, medium, and easy scenarios respectively, FDNET achieves 0.879, 0.945, 0.959.

Table 2. Comparison of YOLO-v8n performance with other algorithms

Algorithm	Hard Scenario	Medium Scenario	Easy Scenario
Proposed YOLO-v3n	**0.901**	**0.946**	**0.959**
FDNET	0.879	0.945	0.959
SFD	0.846	0.924	0.939
CMS-RC	0.624	0.874	0.899
Multitask Cascade CNN	0.598	0.825	0.848
LDCF	0.552	0.762	0.790
Faceness Wider	0.424	0.664	0.731
Multiscale Cascade CNN	0.345	0.634	0.691
Two Stage CNN	0.323	0.618	0.681
ACF Wider	0.279	0.541	0.659

6. CONCLUSIONS

A face detection and recognition model was proposed in this study by combining the YOLO v3 and VGG 16 networks. The WIDER FACE dataset was selected for training the YOLO v3 network, where it achieved high accuracy, recall, precision, and f1 score upon testing. These results suggest the efficiency of the YOLO v3 network in FDR. On the other hand, the LFW dataset was used for training the VGG 16 network, which was able to produce low cross-entropy loss and high accuracy, illustrating its capability of identifying and distinguishing individuals in different poses, lighting conditions, and different facial expressions. Our model achieves a slightly lower accuracy of 95.9% in face detection with the YOLO v3 network, compared to a higher accuracy of 96.2% in face recognition using the VGG 16 network. Surprisingly the results indicate that our model is a top contender for image-based -FDR due to its ability to deliver superior accuracy while maintaining a compact size and efficient computation. In the future, we aim to address issues related to blurry images in low-light conditions and enhance the precision of our method. We intend to create a lightweight real-time model that succeeds in recognizing facial emotions based on landmarks. This will be achieved by utilizing different datasets within 3D CNN, 3D U-Net, and YOLOv8 frameworks.

REFERENCES

[1] Kumar, A., Kaur, A., Kumar, M. (2019). Face detection techniques: A review. Artificial Intelligence Review, 52: 927-948. https://doi.org/10.1007/s10462-018-9650-2

[2] Ahmed, A., Hasan, T., Abdullatif, F.A., Mustafa, S.T., Rahim, M.S.M. (2019). A digital signature system based on real time face recognition. In 2019 IEEE 9th International Conference on System Engineering and Technology (ICSET), Shah Alam, Malaysia, pp. 298-302. https://doi.org/10.1109/ICSEngT.2019.8906410

[3] Goyal, H., Sidana, K., Singh, C., Jain, A., Jindal, S. (2022). A real time face mask detection system using convolutional neural network. Multimedia Tools and

Applications, 81(11): 14999-15015. https://doi.org/10.1007/s11042-022-12166-x

[4] Fan, Y., Luo, Y., Chen, X. (2021). Research on face recognition technology based on improved YOLO deep convolution neural network. In Journal of Physics: Conference Series, Chongqing, China, p. 012010. https://doi.org/10.1088/1742-6596/1982/1/012010

[5] Zhu, Q., Yeh, M.C., Cheng, K.T., Avidan, S. (2006). Fast human detection using a cascade of histograms of oriented gradients. In 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), New York, NY, USA, pp. 1491-1498. https://doi.org/10.1109/CVPR.2006.119

[6] Zhou, M., Wang, Y., Tian, Z., Lian, Y., Wang, Y., Wang, B. (2018). Calibrated data simplification for energy-efficient location sensing in internet of things. IEEE Internet of Things Journal, 6(4): 6125-6133. https://doi.org/10.1109/JIOT.2018.2869671

[7] Gao, J., Yang, T. (2022). Face detection algorithm based on improved TinyYOLOv3 and attention mechanism. Computer Communications, 181: 329-337. https://doi.org/10.1016/j.comcom.2021.10.023

[8] Khan, A., Sohail, A., Zahoora, U., Qureshi, A.S. (2020). A survey of the recent architectures of deep convolutional neural networks. Artificial Intelligence Review, 53: 5455-5516. https://doi.org/10.1007/s10462-020-09825-6

[9] Wu, W., Yin, Y., Wang, X., Xu, D. (2018). Face detection with different scales based on faster R-CNN. IEEE Transactions on Cybernetics, 49(11): 4017-4028. https://doi.org/10.1109/TCYB.2018.2859482

[10] Najibi, M., Singh, B., Davis, L.S. (2019). Fa-RPN: Floating region proposals for face detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, pp. 7723-7732. https://doi.org/10.1109/CVPR.2019.00791

[11] Ren, S., He, K., Girshick, R., Sun, J. (2015). Faster r-CNN: Towards real-time object detection with region proposal networks. Advances in Neural Information Processing Systems, 28: 1-9.

[12] Dai, J., Li, Y., He, K., Sun, J. (2016). R-FCN: Object detection via region-based fully convolutional networks. Advances in Neural Information Processing Systems, 29: 1-9.

[13] Redmon, J., Divvala, S., Girshick, R., Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, pp. 779-788. https://doi.org/10.1109/CVPR.2016.91

[14] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C. (2016). SSD: Single shot multibox detector. In Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, pp. 21-37. https://doi.org/10.1007/978-3-319-46448-0_2

[15] Wen, Y., Zhang, K., Li, Z., Qiao, Y. (2016). A discriminative feature learning approach for deep face recognition. In Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, the Netherlands, pp. 499-515. https://doi.org/10.1007/978-3-319-46478-7_31

[16] Cárdenas, V.H., Cruz, N., Muñoz, S., Villanueva, J.R. (2018). Reconstruction of a kinetic k-essence Lagrangian from a modified of dark energy equation of state. The European Physical Journal C, 78: 1-9. https://doi.org/10.1140/epjc/s10052-018-6066-8

[17] Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S. (2017). Feature pyramid networks for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, pp. 2117-2125. https://doi.org/10.1109/CVPR.2017.106

[18] Chen, X., Fang, H., Lin, T.Y., Vedantam, R., Gupta, S., Dollár, P., Zitnick, C.L. (2015). Microsoft coco captions: Data collection and evaluation server. arXiv preprint arXiv:1504.00325. https://doi.org/10.48550/arXiv.1504.00325

[19] Mutneja, V., Singh, S. (2019). Modified Viola–Jones algorithm with GPU accelerated training and parallelized skin color filtering-based face detection. Journal of Real-Time Image Processing, 16(5): 1573-1593. https://doi.org/10.1007/s11554-017-0667-6

[20] Dalal, N., Triggs, B. (2005). Histograms of oriented gradients for human detection. In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, pp. 886-893. https://doi.org/10.1109/CVPR.2005.177

[21] Cai, Z., Vasconcelos, N. (2018). Cascade r-CNN: Delving into high quality object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, pp. 6154-6162. https://doi.org/10.1109/CVPR.2018.00644

[22] Neubeck, A., Van Gool, L. (2006). Efficient non-maximum suppression. In 18th International Conference on Pattern Recognition (ICPR'06), Hong Kong, China, pp. 850-855. https://doi.org/10.1109/ICPR.2006.479

[23] Zhang, K., Zhang, Z., Li, Z., Qiao, Y. (2016). Joint face detection and alignment using multitask cascaded convolutional networks. IEEE Signal Processing Letters, 23(10): 1499-1503. https://doi.org/10.1109/LSP.2016.2603342

[24] Huang, L., Yang, Y., Deng, Y., Yu, Y. (2015). Densebox: Unifying landmark localization with end to end object detection. arXiv preprint arXiv:1509.04874. https://doi.org/10.48550/arXiv.1509.04874

[25] Shelhamer, E., Long, J., Darrell, T. (2016). Fully convolutional networks for semantic segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(4): 640-651. https://doi.org/10.1109/tpami.2016.2572683

[26] Yang, S., Luo, P., Loy, C.C., Tang, X. (2017). Faceness-net: Face detection through deep facial part responses. IEEE Transactions on Pattern Analysis and Machine Intelligence, 40(8): 1845-1859. https://doi.org/10.1109/TPAMI.2017.2738644

[27] Hu, P., Ramanan, D. (2017). Finding tiny faces. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, pp. 951-959. https://doi.org/10.1109/CVPR.2017.166

[28] Najibi, M., Samangouei, P., Chellappa, R., Davis, L.S. (2017). SSH: Single stage headless face detector. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, pp. 4875-4884. https://doi.org/10.1109/ICCV.2017.522

[29] Simonyan, K., Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition.

arXiv preprint arXiv:1409.1556. https://doi.org/10.48550/arXiv.1409.1556
[30] Zhang, C., Xu, X., Tu, D. (2018). Face detection using improved faster RCNN. arXiv preprint arXiv:1802.02142. https://doi.org/10.48550/arXiv.1802.02142
[31] Wang, H., Li, Z., Ji, X., Wang, Y. (2017). Face r-CNN. arXiv preprint arXiv:1706.01061. https://doi.org/10.48550/arXiv.1706.01061
[32] Tang, X., Du, D.K., He, Z., Liu, J. (2018). Pyramidbox: A context-assisted single shot face detector. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, pp. 797-813. https://doi.org/10.1007/978-3-030-01240-3_49
[33] Sun, X., Wu, P., Hoi, S.C. (2018). Face detection using deep learning: An improved faster RCNN approach. Neurocomputing, 299: 42-50. https://doi.org/10.1016/j.neucom.2018.03.030
[34] Tammina, S. (2019). Transfer learning using VGG-16 with deep convolutional neural network for classifying images. International Journal of Scientific and Research Publications: 9(10): 143-150. https://doi.org/10.29322/IJSRP.9.10.2019.p9420
[35] Chen, W., Huang, H., Peng, S., Zhou, C., Zhang, C. (2021). YOLO-face: A real-time face detector. The Visual Computer, 37: 805-813. https://doi.org/10.1007/s00371-020-01831-7
[36] Zhao, L., Li, S. (2020). Object detection algorithm based on improved YOLOv3. Electronics, 9(3): 537. https://doi.org/10.3390/electronics9030537
[37] Learned-Miller, E., Huang, G.B., RoyChowdhury, A., Li, H., Hua, G. (2016). Labeled faces in the wild: A survey. Advances in Face Detection and Facial Image Analysis, 189-248. https://doi.org/10.1007/978-3-319-25958-1_8
[38] Wang, J., Yuan, Y., Yu, G. (2017). Face attention network: An effective face detector for the occluded faces. arXiv preprint arXiv:1711.07246. https://doi.org/10.48550/arXiv.1711.07246
[39] Valero-Carreras, D., Alcaraz, J., Landete, M. (2023). Comparing two SVM models through different metrics based on the confusion matrix. Computers & Operations Research, 152: 106131. https://doi.org/10.1016/j.cor.2022.106131
[40] Hadad, A.A.A., Khalid, H.N., Naser, Z.S., Taha, M.S. (2022). A robust color image watermarking scheme based on discrete wavelet transform domain and discrete slantlet transform technique. Ingenierie des Systemes d'Information, 27(2): 313-319. https://doi.org/10.18280/isi.270215
[41] Naveen Kumar, K.L., Dhanwad, R., Niranjan, M.S., Chaluvaraju, K.C. (2023). Area under curve method for the simultaneous quantitative estimation of pravastatin sodium and aspirin. Indian Journal of Pharmaceutical Education & Research, 57(1): 234. https://doi.org/10.5530/001954640307
[42] Chong, S.C., Teoh, A.B.J., Ong, T.S. (2017). Unconstrained face verification with a dual-layer block-based metric learning. Multimedia Tools and Applications, 76: 1703-1719. https://doi.org/10.1007/s11042-015-3120-5
[43] Xiong, C., Liu, L., Zhao, X., Yan, S., Kim, T.K. (2015). Convolutional fusion network for face verification in the wild. IEEE Transactions on Circuits and Systems for Video Technology, 26(3): 517-528. https://doi.org/10.1109/TCSVT.2015.2406191
[44] Majumdar, A., Singh, R., Vatsa, M. (2016). Face verification via class sparsity based supervised encoding. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(6): 1273-1280. https://doi.org/10.1109/TPAMI.2016.2569436
[45] Zhang, J., Jin, X., Liu, Y., Sangaiah, A.K., Wang, J. (2018). Small sample face recognition algorithm based on novel Siamese network. Journal of Information Processing Systems, 14(6): 1464-1479. https://doi.org/10.3745/JIPS.02.0101
[46] Ameur, B., Belahcene, M., Masmoudi, S., Hamida, A.B. (2018). Weighted PCA-EFMNet: A deep learning network for face verification in the wild. In 2018 4th International Conference on Advanced Technologies for Signal and Image Processing (ATSIP), Sousse, pp. 1-6. https://doi.org/10.1109/ATSIP.2018.8364460
[47] Heidari, M., Fouladi-Ghaleh, K. (2020). Using siamese networks with transfer learning for face recognition on small-samples datasets. In 2020 International Conference on Machine Vision and Image Processing (MVIP), Iran, pp. 1-4. https://doi.org/10.1109/MVIP49855.2020.9116915