Bulletin of Electrical Engineering and Informatics
Vol. 13, No. 2, April 2024, pp. 1228~1237
ISSN: 2302-9285, DOI: 10.11591/eei.v13i2.5424

Mathematics for 2D face recognition from real time image data set using deep learning techniques

Ambika G. N.[1], Yeresime Suresh[2]
[1]Department of Computer Science and Engineering, BMS Institute of Technology and Management, Bengaluru and Visvesveraya Technological University, Belagavi, India
[2]Department of Computer Science and Engineering, Ballari Institute of Technology and Management, Ballari and Visvesveraya Technological University, Belagavi, India

Article Info

Article history:

Received Dec 7, 2022
Revised Oct 4, 2023
Accepted Oct 17, 2023

Keywords:

Computer vision
Dataset
Deep learning
Face detection
Haar features
OpenCV
Rasterized images

ABSTRACT

The recognition of human faces poses a complex challenge within the domains of computer vision and artificial intelligence. Emotions play a pivotal role in human interaction, serving as a primary means of communication. This manuscript aims to develop a robust recommendation system capable of identifying individual faces from rasterized images, encompassing features such as eyes, nose, cheeks, lips, forehead, and chin. Human faces exhibit a wide array of emotions, with some emotions, including anger, sadness, happiness, surprise, fear, disgust, and neutrality, being universally recognizable. To achieve this objective, deep learning techniques are leveraged to detect objects containing human faces. Every human face exhibits common characteristics known as Haar features, which are employed to extract feature values from images containing multiple elements. The process is executed through three distinct stages, starting with the initial image and involving calculations. Real-time images from popular social media platforms like Facebook are employed as the dataset for this endeavor. The utilization of deep learning techniques offers superior results, owing to their computational demands and intricate design when compared to classical computer vision methods using OpenCV. The implementation of deep learning is carried out using PyTorch, further enhancing the precision and efficiency of face recognition.

Corresponding Author:

Ambika G. N.
Department of Computer Science and Engineering
BMS Institute of Technology and Management, Bengaluru, and Visvesveraya Technological University
Belagavi, India
Email: ambikagn@bmsit.in

1. INTRODUCTION

The major idea is to propose a method that recognizes and detects human faces in given pictures or videos. Facial recognition is the ability to automatically detect a face. It has the potential to identify different parts of the embodiment depending on the presence of facial characteristics. It's always easy for a person to find a face in a collection of images and differentiate images correctly. However, a computer should be properly trained so that when a real-time dataset is provided, the system should be able to identify the face of a person as well as additional characteristics such as eyes, nose, mouth, cheeks, lips, forehead, and chin.

The key concept here is to build a system that can distinguish faces from non-facial elements [1]. Deep learning adapts to new images, assuming they are similar to the data it was trained on. Computer vision is an area of artificial intelligence that enables computers and systems to derive meaningful information from

binary images, videos, and other visual inputs [2]. The dataset is taken as a set of real-time images. But what is an image? An image is a collection of arrays representing different numerical values in terms of red, green, and blue (RGB) which is commonly used in computer vision. The values for these colors range from 0 to 255, where a higher value represents more intensity or brightness. Images are stored in multidimensional arrays. There are two main types of images: raster-based images and vector images. For our research, we are considering raster-based images.

There are different file formats for images such as jpg, GIF, PNG, TIFF, RAW [3], and PSD. The structures present in an image and the final output are the detected picture and a selection of the face [4]. The competitive aspect of this research article is to develop a system that can identify faces under various lighting conditions. Using convolutional neural networks (CNN), we predict the faces from different images in the dataset. Predicting information in the image is challenging, and we first need to train a CNN to correctly classify the images.

Many researchers have explored various methods for analyzing the emotions of images; outcomes of machine learning techniques are significant. Among several machine learning techniques given in Table 1, methods that depend on deep learning produce the best efficiency for face recognition from the images in the photographs.

Table 1. Literary review

Authors	Title of paper	Techniques used for face recognition	Dataset	Result (%)
Turk and Pentland [5]	Eigenfaces for recognition	Principal component analysis	400	79.65
Tomasi *et al.* [6]	Ive got my virtual eye on you: remote proctors and academic integrity	Wavelet transform	100	90
Yang *et al.* [7]	Large scale identity deduplication using face recognition based on facial feature points	Active shape model	100	89
Hasan *et al.* [8]	Human face detection techniques: a comprehensive review and future research directions	Support vector machines	200	91
Sparks [9]	The brainstem control of saccadic eye movements	CNN	200	93.7

2. MATERIALS AND METHOD

CNN model [10], [11] to be trained to predict whether an image has a triangle or any other information. How would you tell a computer about the shape present in the image If the image is shifted? How would a neural network be able to predict this? CNN have many filters, which are used to scan an image and obtain a feature map. Output is obtained by implementing the different concepts of CNN, as shown in Figure 1. CNN consists of various operations: i) convolutions, ii) feature detectors, iii) padding, iv) stride, v) activation layer, vi) pooling, vii) fully connected, and viii) Softmax function [12].

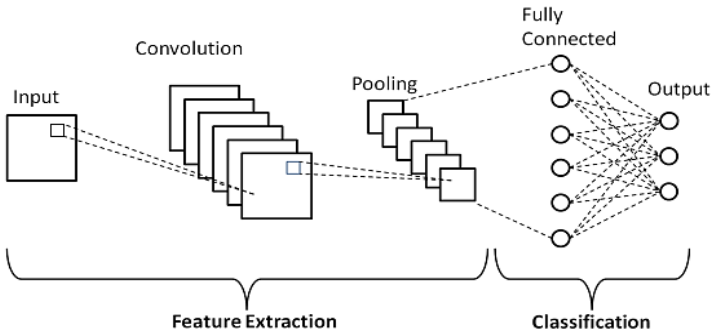

Figure 1. Block diagram of CNN

2.1. Convolution operation and feature detector

A scientific term to explain the method of combining two functions to obtain a third outcome is known as a feature map, as depicted in Figure 2. Convolution, also known as filter or kernel, is in a matrix form applied to an input image [13], [14].

Image * Kernel = Feature Map

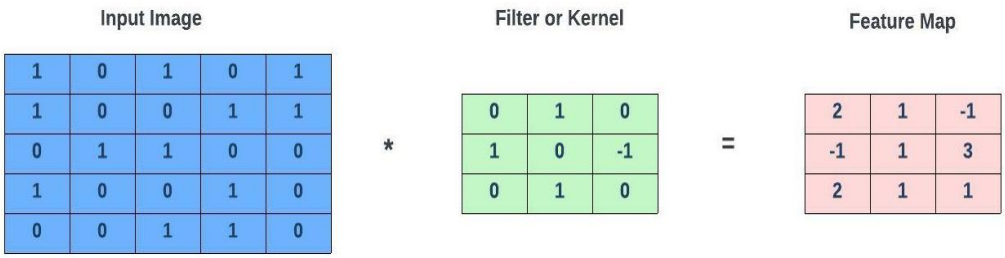

Figure 2. An example for a convolution operation

The convolution operation is used to detect the features in images; feature maps are also known as feature detectors, which can see many features in the image, as shown in Figure 3. A combination of these features is used to classify the image correctly.

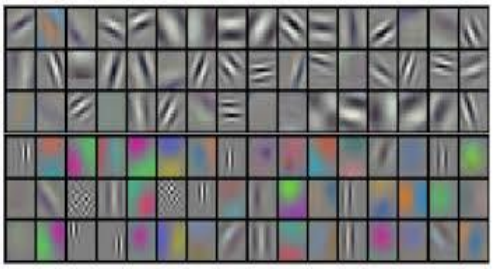

Figure 3. Different features obtained from a convolution operation

2.2. Padding

It allows us to manipulate the feature map size. Conv filters produce an output more minor than the input; we have taken a 5x5 image and padded it with 0's in all dimensions: left, correct, bottom, and top corners. Padding helps pass the conv filters multiple times, as shown in Figure 4.

$$Feature\ Map\ Size(m) = n - f + 1 \quad (1)$$

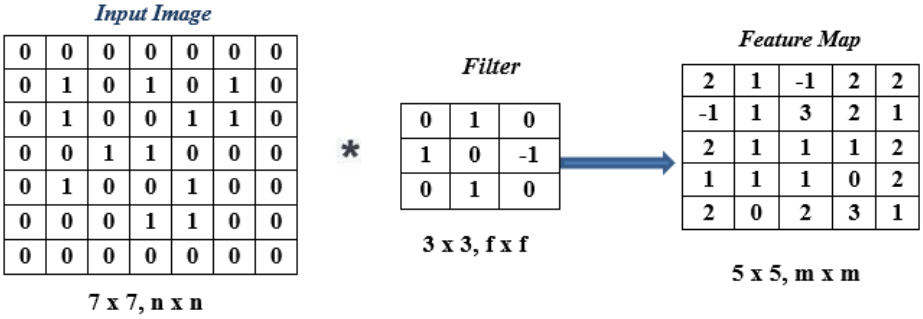

Figure 4. An example of padding operation

2.3. Stride

Stride defines how many steps the convolution window moves across the input image. Examples are illustrated for a stride of 1 and a stride of 2. An example for a stride operation is defined. A larger stride produces a smaller feature map output, and a larger stride has less overlap. Stride is used to control the size of the feature map. A larger stride has less overlap. We can calculate the feature map size using (2):

$$(n \times n) * (f \times f) = \left(\frac{n+2p-f}{s} + 1\right) \times \left(\frac{n+2p-f}{s} + 1\right) \quad (2)$$

Where $n \times n$ is input image size, $f \times f$ is filter size, s denotes stride value and p denotes padding

2.4. Activation layer

The purpose of the activation function is to enable the learning of complex patterns in our data, introduce nonlinearity to our network, and allow a nonlinear decision boundary via nonlinear combinations of the weight and inputs. There are several activation functions we can use in our CNN. Rectified linear units (ReLU) have become the activation function of choice for CNNs. ReLU function helps to train CNN. Simple computation (fast to qualify) does not saturate. The ReLu operation changes all negative values to 0 and leaves all positive values alone.

2.5. Pooling layer

Pooling is the process where we reduce the size of dimensionality of a feature map, allowing us to decrease the size of parameters in our network while retaining essential features. Pooling is also known as Subsampling or downsampling. In the below operation, A 2x2 kernel is used in the first block of 2x2; a maximum value of 123 is selected, and subsequent values in the output are 253, 187, and 165 in the production.

Pooling makes our CNN model more invariant to minor transformations and distortions in our images. Pooling helps to maximize the translation invariance. Pooling reduces the output size by sub-sampling the filter response without losing information [15]. A significant stride in the pooling layer leads to high information loss. A stride of 2 and a kernel size 2x2 for the pooling layer were effective in practice.

2.6. Fully connected layer

It means all the nodes in one layer are connected to the outputs of the subsequent layer. Considers 3D data output of the previous layer and flattens it into a single vector used for input in the next layer. It is also known as dense layer. A fully connected layer compiles the data/outputs extracted from previous layers to produce outcomes, easy to learn nonlinear combinations of these features.

3. SYSTEM DESIGN

3.1. Why convolutional neural network works well on images?

Standard neural networks don't have convolution filter inputs; for images, every pixel will be its input; therefore, a small image that's 28 X 28 would have 784 input nodes for our first layer. The first step here is how to train a CNN. Conv filters learn what the feature detectors learn, and our typical early layer of CNN learns low-level features (like edges or lines) specified in Figure 5. Mid-layers learn simple patterns, whereas high-level layers learn more structured, complex ways [16]–[19]. During training process, the following are the steps to be followed:

- Initialize random weights values for our trainable parameters.
- Forward propagate an image or batch of images through our network.
- Calculate the total error (get some output through the random values).
- We use a back propagation to update our gradients (weights) via a gradient descent.
- Propagate more images (or batch) and update weights until all images in the dataset have been propagated (one epoch).
- Repeat a few more epochs (i.e, passing all image batches through our network) until our loss reaches satisfactory values.

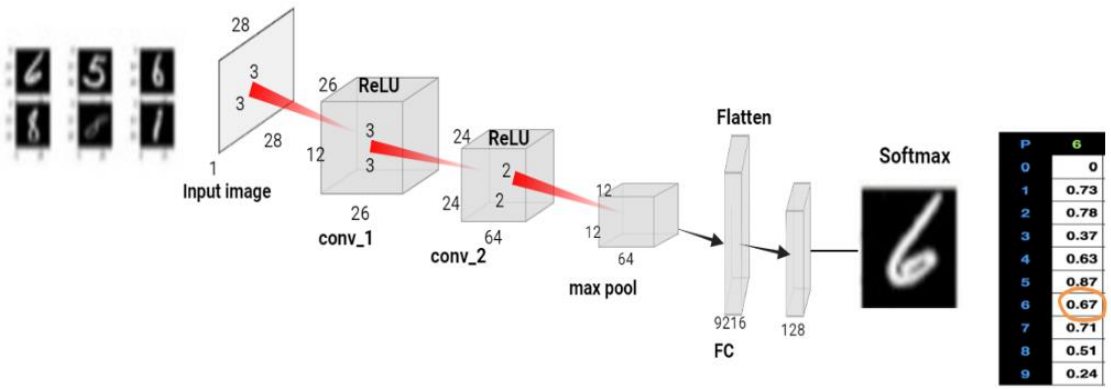

Figure 5. Training process of CNN

This is how we generate random weights for one input image; we get the output of arbitrary values for several images, as shown in Figure 6. If all the values generated are correct, we need to figure out how we correct our results using the CNN. We create a loss function and use the back propagation method to do this.

P	6	5	6	8	8	1
0	0	0.9	0.83	0.21	0.19	0.62
1	0.73	0.8	0.89	0.7	0.92	0.07
2	0.78	0.88	0.19	0.39	0.08	0.74
3	0.37	0.56	0.07	0.64	0.64	0.9
4	0.63	0.25	0.79	0.94	0.52	0.55
5	0.87	0.65	0.57	0.63	0.97	0.04
6	0.67	0.05	0.45	0.51	0.87	0.51
7	0.71	0.66	0.13	0.59	0.86	0.89
8	0.51	0.88	0.59	0.01	0.37	0.63
9	0.24	0.52	0.79	0.15	0.63	0.78

Figure 6. Result of random values during the training process in CNN for six images

3.2. Loss function

The loss function is used for quantifying the loss, how bad the probabilities we predicted, and we need to quantify the degree of our prediction. Entropy loss is used for quantifying the loss; it uses two distributions,

$$L = -y.\log(\hat{y}) \tag{3}$$

Here y is the ground truth vector, ^y is the predicted distribution, and ' .' is the inner product. There are also other loss functions that exist. Loss function functions are also called cost functions. For binary classification problems, we use binary cross entropy loss. For regression, we often use the mean square error (MSE).

$$\text{MSE} = (\text{Target} - \text{Predicted})^2 \tag{4}$$

Other loss functions used are L1, L2, hinge loss, and mean absolute error (MAE). If there are errors in the values, we will update our weights using the back propagation technique to minimize the loss.

3.3. Back propagation

Back propagation is very important in training the neural networks, and this process is used to know how much to change/update the gradients to reduce the overall loss, as depicted in Figure 7. The Figure 7 shows the operation of back propagation and the formulas we use in the same [20]. Using the loss value, backpropagation can tell us now, for the next iteration, how much we should increase or decrease the weights to reduce the overall loss in the network. By forward propagating input data, we can use backpropagation to lower the importance and the loss. But this tunes weights for that particular input or batch or inputs. We improve generalization (ability to make good predictions on unseen data by using all data in our training dataset.

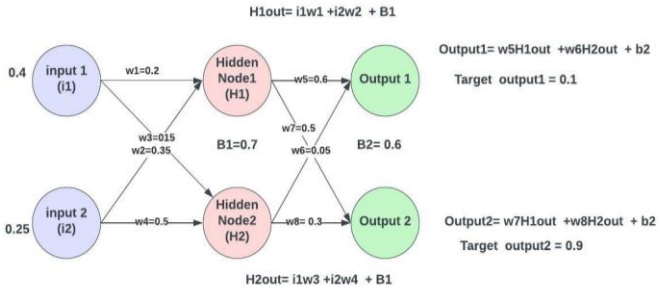

Figure 7. An operation of back propagation technique

3.4. Gradient descent

In the back propagation process, we update the individual weights or gradients given by wx+b. The main is to find the correct value of consequences where the loss is the lowest. Gradient descent is The method of achieving this goal (i.e., updating all weights to lower the total loss). It's the point at which we find the optimal weights such that failure is near the lowest. Gradients are the derivative of a function; they tell us the rate of change of one variable for another variable.

$$Gradient = \frac{dE}{dw}$$

Where E is the error or loss and w is the weights.

A positive gradient means loss increases if weight increases, and a negative gradient means loss decreases. At point A, moving right increases our weight and decreases our loss negatively; at point B, moving right increases our weight and increases our loss positively. Therefore, the negative of our gradient tells us the direction in which we are moving. The point at which a gradient is zero means that small changes to the left or right don't change the loss. In training neural networks, this is good and bad; at point C, minor changes to the left or right don't change the loss. Minimal changes to the left or right don't change the loss, and the network gets stuck during training. This is called getting stuck in a local minima. We will use the mini batch gradient descent method, which combines both ways. It takes a batch of data points (images) and forward propagates all, then updates the gradients. This leads to faster training and convergence to the global minima.

3.5. Optimization

This is a more advanced gradient descent method that allows us to find the lowest weights, and a few advanced optimization techniques are used. What are the problems we need to deal with standard stochastic gradient descent? These include choosing an appropriate learning rate (LR), deciding on learning rate schedules, and using the same learning rate for all parameter updates (as in the case of sparse data). Stochastic gradient descent [21] is susceptible to getting trapped in local minima or saddle points (where one dimension slopes up and the other slopes down).

Several other algorithms have been developed to solve these problems, including extensions to the stochastic gradient descent method, such as momentum and nesterov's acceleration. Several optimizers, including Adagrad, Adadelta, Adam, RMSprop, AdaMax, and Nadam. have been introduced. We have used the Adam optimizer [22]; Adam's adaptive moment estimation is a method that computes adaptive learning rates for each parameter and stores an exponentially decaying average of past gradients, similar to momentum. Adam is quite effective.

4. IMPLEMENTATION

4.1. Dataset

Dataset considered for implementation is real-time dataset. Images are collected from the social media, different restaurants and several showrooms [23]. Below is the displayed mathematical model for face recognition. We will create a collection of facial images in the database, labeled as y1, y2, y3, … yn. N classes are created from these sets, and each class corresponds to a registered person. We define a vector comprising K values for each image.

$$\mu = (\mu_1, \mu_2, \mu_3, \cdots, \mu_k)^T \quad (5)$$

Let T represent the transpose operator. We define a distance function, denoted as d(μ2, μs), for the feature vector μ that corresponds to the farthest distance in the input form and fit in to class XL, for each image.

$$d(\mu_1, \mu_s) > d(\mu_j, \mu_s) \neq j, 1, j = 0,1, \cdots, L-1 \quad (6)$$

In the context of the provided distance function, class XL must exceed a precomputed threshold value, represented as d(μ1, μs) > τc. The face recognition algorithm takes an image as input and produces a sequence of face frame coordinates as output. There may be one face frame, zero face frames, or several face frames in this sequence [24].

The mathematical model for determining the integral image involves determining the pixel values of the face as (7):

$$I_i(y, z) = \sum_{i=1}^{y.z} i(y', z') \quad (7)$$

The sum of the intensities of the pixels within the black areas is as follows: let I_i(y, z) represent the value of the ith element in the integral image with coordinates (y, z), and let (y', z') represent the brightness of the pixel in the image under consideration at coordinates (y', z').

$$s_i = \sum_{i=0}^{v.y} r_i, k - \sum_{i=0}^{z.y} r_i, f \tag{8}$$

The equation above is employed for computing the total brightness of the pixels. With a threshold value of τ_c, the classifier's expression is as (9):

$$w = \begin{cases} 1, & f_i > \tau_c; \\ -1, & f_i < \tau_c; \end{cases} \tag{9}$$

A set of weights, denoted as wi for 1 < i ≤ n, corresponds to each sample. The best strong classifier is computed using a fixed number of weak classifiers, and the equation for the strong classifier is as (10), (11):

$$w = \begin{cases} 1, & \sum_{c=1}^{c} a_c w_c \geq \frac{1}{2} \sum_{c=1}^{c} a_c ; \\ -0, & \sum_{c=1}^{c} a_c \, w_c \, f_i < \frac{1}{2} \sum_{c=1}^{c} a_c ; \end{cases} \tag{10}$$

$$a_c = \log \frac{1}{\beta_c} \tag{11}$$

Where w_c represents the weak classifier, and a_c and β are the weight coefficients associated with the weak classifier. Here, c stands for the current number, and C=(1,....,C) represents the set of weak classifiers. The goal of this iterative technique is to build a reliable classifier. The following describes the images that show the object both before and after illumination:

$$q1(y,z) = \frac{\delta_{0,j}(y,z)}{\delta_{1,j}(y,z)} + 1 \tag{12}$$

Where j is the number of current value of the sequence y, z, $\delta_{b,j}(y,z)$ the brightness value of the pixel of the array $I_{b,j}$. b is the identifier of the pixel array, with b taking values in the set (0,1).

The scientific study involves the analysis of image dispersion after applying brightness. The dispersion value is determined by calculating the sum of squared differences between the pixel values in the modified image and the mean pixel value across the width and height arrays.
The dispersion value is calculated as:

$$D(I_{0,j} I_{1,j)} = \frac{1}{WH} \sum_{y=0}^{W-1} \sum_{z=0}^{H-1} \left(q_j(y,z) - q_j\right)^2 \tag{13}$$

In image processing, this formula describes the calculation of dispersion value between two images ($I(0.j)$ *and* $I(1,j)$, $q_j(y,z)$ denotes the pixel value at coordinates (y,z) for the image parameter j. q_j is the mean pixel value along the y-axis for a specific image parameter j. random variable q_j which is calculated by (14):

$$q_j = \frac{1}{WH} \sum_{y=0}^{W-1} \sum_{z=0}^{H-1} (q_j\,(y,z)) \tag{14}$$

5. RESULTS AND DISCUSSION

Colored RGB images are captured from a camera. The images are depicted in Figure 8 and represent real-time images of my friends. These images serve as input for face recognition, which is implemented using TensorFlow and Keras libraries. To extract facial features, we employed Haar cascade features [25], including various types such as: i) edge characteristics; ii) linear characteristics; iii) center characteristics; and iv) diagonal characteristics. The face recognition model and the images used for face recognition are subsequently employed for emotion detection of individuals using deep learning techniques, as illustrated in Figures 9 and 10.

Figure 8. Input images for face detection

Figure 9. Face recognition images using deep learning techniques

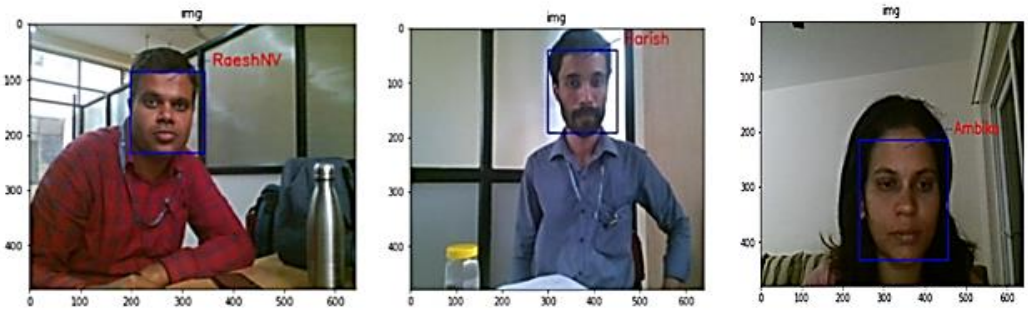

Figure 10. Faces are detected through webcam

5.1. Measuring the accuracy of face recognition for an image

Several images, either in groups or individually, are uploaded into the system to check their accuracy. A person should have appeared in the photos multiple times. When all the images are tested in the proposed system, the data is computed in the confusion matrix, as shown in Table 2, to calculate the system's efficiency. The accuracy of the proposed system is then computed as Table 2.

Table 2. Confusion matrix form face recognition

People	No of Images	True
Person 1	5	5
Person 2	5	5
Person 3	5	4

6. CONCLUSION

In this research article, face detection and face recognition in videos are achieved through deep learning techniques. The complete process of the face detection system begins with data training using the CNN approach, followed by face recognition, which is elaborated upon. This article employs Tensorflow and Keras to test the model on various RGB images. Additionally, the system's performance is assessed for both sets of images and single images captured via a webcam, as well as for video inputs. In video processing, the system effectively extracts faces of individuals. The proposed system exhibits a high level of accuracy, achieving a recognition rate of 94% after training with a substantial number of face images. However, several factors can impact the model's accuracyp. In scenarios with insufficient light intensity or other factors affecting image clarity, accuracy tends to decrease compared to situations with higher light intensity. Furthermore, the classifier plays a pivotal role in the recognition process, and superior results are obtained when the model is trained with a large dataset of images. The generated faces can be utilized as inputs in various applications such as emotion recognition, theft identification, defense applications, and more. Deep learning techniques consistently outperform OpenCV functions in terms of accuracy.

REFERENCES

[1] Y. Ge, R. Zhang, X. Wang, X. Tang, and P. Luo, "DeepFashion2: A Versatile Benchmark for Detection, Pose Estimation, Segmentation and Re-Identification of Clothing Images," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019, pp. 5332–5340, doi: 10.1109/CVPR.2019.00548.

[2] J. Kim and L. Wei, "Performance Analysis of Machine Learning-based Face Detection Algorithms in Face Image Transmission over AWGN and Fading Channels," in *2021 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, Aug. 2021, pp. 1–5, doi: 10.1109/BMSB53066.2021.9547181.

[3] P. Mahapattanakul, "From Human Vision to Computer Vision — A Brief History (Part2/4)," *Medium*. 2019. [Online]. Available: https://becominghuman.ai/from-human-vision-to-computer-vision-convolutional-neural-network-part3-4-24b55ffa7045

[4] M. S. KALAS, "Real Time Face Detection and Tracking Using OpenCV," in *international journal of soft computing and Artificial Intelligence*, 2014, pp. 137–140.

[5] M. Turk and A. Pentland, "Eigenfaces for Recognition," *Journal of Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, Jan. 1991, doi: 10.1162/jocn.1991.3.1.71.

[6] L. F. Tomasi, V. L. Figiel, and M. Widener, "Ive Got My Virtual Eye On You: Remote Proctors And Academic Integrity," *Contemporary Issues in Education Research (CIER)*, vol. 2, no. 1, pp. 31–36, Jan. 2011, doi: 10.19030/cier.v2i1.1103.

[7] X. Yang, G. Su, J. Chen, N. Su, and X. Ren, "Large Scale Identity Deduplication Using Face Recognition Based on Facial Feature Points," in *Chinese Conference on Biometric Recognition*, Heidelberg: Springer Link, 2011, pp. 25–32, doi: 10.1007/978-3-642-25449-9_4.

[8] M. K. Hasan, M. S. Ahsan, Abdullah-Al-Mamun, 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] D. L. Sparks, "The brainstem control of saccadic eye movements," *Nature Reviews Neuroscience*, vol. 3, no. 12, pp. 952–964, Dec. 2002, doi: 10.1038/nrn986.

[10] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, 2001, vol. 1, doi: 10.1109/CVPR.2001.990517.

[11] N. Mahajan and H. Mahajan, "Emotion Detection Algorithm," *International Journal of Electrical and Electronics Research*, vol. 2, no. 2, pp. 56–60, 2014.

[12] I. Paliy, "Face detection using Haar-like features cascade and convolutional neural network," in *TCSET 2008 - Modern Problems of Radio Engineering, Telecommunications and Computer Science - Proceedings of the International Conference*, 2008, pp. 375–377.

[13] Y. Sun, X. Wang, and X. Tang, "Deeply learned face representations are sparse, selective, and robust," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015, pp. 2892–2900, doi: 10.1109/CVPR.2015.7298907.

[14] H. Ahamed, I. Alam, and M. M. Islam, "HOG-CNN Based Real Time Face Recognition," in *2018 International Conference on Advancement in Electrical and Electronic Engineering (ICAEEE)*, Nov. 2018, pp. 1–4, doi: 10.1109/ICAEEE.2018.8642989.

[15] F. Huang, X. Zhang, Z. Zhao, J. Xu, and Z. Li, "Image–text sentiment analysis via deep multimodal attentive fusion," *Knowledge-Based Systems*, vol. 167, pp. 26–37, Mar. 2019, doi: 10.1016/j.knosys.2019.01.019.

[16] B. Abirami, T. S. Subashini, and V. Mahavaishnavi, "Gender and age prediction from real time facial images using CNN," in *Materials Today: Proceedings*, 2020, vol. 33, pp. 4708–4712, doi: 10.1016/j.matpr.2020.08.350.
[17] S. Khan, A. Akram, and N. Usman, "Real Time Automatic Attendance System for Face Recognition Using Face API and OpenCV," *Wireless Personal Communications*, vol. 113, no. 1, pp. 469–480, Jul. 2020, doi: 10.1007/s11277-020-07224-2.
[18] M. Khan, S. Chakraborty, R. Astya, and S. Khepra, "Face Detection and Recognition Using OpenCV," in *2019 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, Oct. 2019, pp. 116–119, doi: 10.1109/ICCCIS48478.2019.8974493.
[19] Pranav. K B and Manikandan. J, "Design and Evaluation of a Real-Time Face Recognition System using Convolutional Neural Networks," in *Procedia Computer Science*, 2020, vol. 171, pp. 1651–1659, doi: 10.1016/j.procs.2020.04.177.
[20] S. Tikoo and N. Malik, "Detection of Face using Viola Jones and Recognition using Back Propagation Neural Network," *arXiv preprint arXiv:1701.08257*, 2017.
[21] T. S. Akheel, V. U. Shree, and S. A. Mastani, "Stochastic gradient descent linear collaborative discriminant regression classification based face recognition," *Evolutionary Intelligence*, vol. 15, no. 3, pp. 1729–1743, Sep. 2022, doi: 10.1007/s12065-021-00585-y.
[22] O. C. Oguine, K. J. Oguine, H. I. Bisallah, and D. Ofuani, "Hybrid facial expression recognition (FER2013) modelfor real-time emotion classification and prediction," *BOHR International Journal of Internet of things, Artificial Intelligence and Machine Learning*, vol. 1, no. 1, pp. 56–64, 2022, doi: 10.54646/bijiam.2022.09.
[23] T. Zheng, W. Deng, and J. Hu, "Cross-Age LFW: A Database for Studying Cross-Age Face Recognition in Unconstrained Environments," Beijing University of Posts and Telecommunications, 2018.
[24] A. G. N., B. P. Singh, B. Sah, and D. Tiwari, "Air Quality Index Prediction using Linear Regression," *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 8, no. 2, pp. 4247–4252, 2019, doi: 10.35940/ijrte.B2437.078219.
[25] Z. Arya and V. Tiwari, "Automatic Face Recognition and Detection Using OpenCV, Haar Cascade and Recognizer for Frontal Face," *International Journal of Engineering Research and Applications www.ijera.com*, vol. 10, no. 6, pp. 13–19, 2020, doi: 10.9790/9622-1006051319.

BIOGRAPHIES OF AUTHORS

Ambika G. N. received the Engineer degree in Computer Science and Engineering from R. L. Jalappa Institute of Technology in 2008 affiliated to VTU. Completed Master of Technology in 2012 from Nitte Meenakshi Institute of Technology Affiliated to VTU, currently pursuing Ph.D. in the area of Computer Vision. Areas of interests are artificial intelligence, robotics, and computer vision. She has published various research articles in various national and international journals. She can be contacted at email: ambikagn@bmsit.in.

Suresh Yeresime received Ph.D. in 2015 from National Institute of Technology, Rourkela, India. Currently he is working as Professor, Department of Computer Science and Engineering, Ballari Institute of Technology and Management. His areas of interest include: artificial intelligence, machine learning, and data mining. He has published various research articles in reputed conferences and journals. He can be contacted at email: suresh.vec04@gmail.com.