BIS INFORMATION TECHNOLOGY AND COMPUTER SCIENCE

Volume 1 2024 | DOI: 10.31603/bistycs.117
5th Borobudur International Symposium on Science and Technology (BIS-STE) 2023

UNIMMA PRESS
UNIVERSITAS MUHAMMADIYAH MAGELANG

Attendance face detection on mobile device using particle swarm optimization and linear discriminant analysis

Iskandar[1*], M A Sobarnas[1], U T Abdurrahman[1] and N Wuryani[1]

[1] Teknik Informatika, Sekolah Tinggi Teknologi Muhammadiyah Cileungsi, Bogor, Indonesia
[*]Corresponding author email: iskandar@sttmcileungsi.ac.id

Abstract

Artificial intelligence-based facial recognition has been utilized extensively in automation systems, however processing facial recognition properly is necessary to achieve high accuracy with reasonable computing time. On the model and mobile application side, there were two different kinds of experiments run in this study. Feature selection in modeling is done by the use of the Linear Discriminant Analysis method, K-Nearest Neighbor classification, and Particle Swarm Optimization. The ORL database, which contains 255 facial photographs from various persons, was the source of the dataset for this study. The K-Nearest Neighbor method model combined with Particle Swarm Optimization with Confusion Matrix results in the largest accuracy improvement in the model test, producing values of accuracy 97.77%, precision 97.22%, recall 97.61%, and k-fold 97.3, The average value has increased by three points compared to earlier testing without Particle Swarm Optimization. It results in an inference time of 4 to 5 seconds on Android 7 test devices and a half-second inference time on Android 11 test devices when tested on actual devices to measure the response speed of objects identified by mobile devices for facial recognition.

Keywords

Artificial intelligence, Face detection, Mobile device

Published:
October 20, 2024

Introduction

Face recognition powered by artificial intelligence (AI) [1] is becoming more and more widespread because of computer technology and interpreter languages like Python. using Python 3.8, which includes built-in tensor flow and face recognition functionality.

Faces are processed via computer digitization, which seeks to be automated for detecting reasons. Faces are processed by taking photographs of a specific size, cropping them, and then encoding them using 128-D Encodings [2]. Then facial landmark detection with dLib is performed. The facial training data in this dLib is used to estimate the location of coordinates 68 (x, y) that are mapped to the obtained facial data. Imutils performs a face orientation process alignment by default with the user facing forward.

The K-Nearest Neighbor method [3] is the most basic method with a learning algorithm foundation for classifying data, and it compares a new data object to the stored training data that it is most comparable to (neighbor). The KNN approach will be employed in research and for the feature selection process in face processing utilizing the Particle Swarm Optimization (PSO) method because it is also extremely successful on very huge data [4].

For machines to recognize and identify a specific item, which is an object that is captured in the form of an image, machine learning [5] imitates how the human brain functions. The Mobile FaceNet library, which can present a very effective CNN model specifically created for high-precision real-time face verification on mobile devices, was first developed by a company with the name Watchdata Inc. located in Beijing. It is used in its implementation using Android mobile based TensorFlow Lite and the Mobile FaceNet library in the process of real-time face recognition on Android applications. With a model that is only 4.0 MB in size, Mobile FaceNet delivers very good speed performance and high accuracy. The precision attained is quite comparable to heavier models like FaceNet [6].

As a basis for researchers in conducting this research, the following are related previous studies that have something to do with this research, namely research on facial recognition optimization with Linear Discriminant Analysis (LDA), K-Nearest Neighbor (K-NN) combined with Particle Swarm Optimization (PSO). produces an accuracy value of 71.67% [7].

Method

Research flowchart

In this study, numerous steps were completed, including literature review, dataset construction, system design, system testing, and documenting and reporting if everything went well (Figure 1). If an error occurred, the system was repaired again until it was finished.

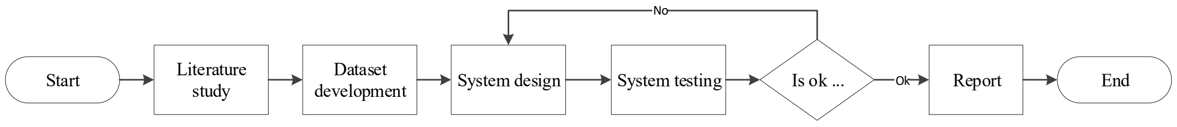

Figure 1. Research flowchart

Collection of datasets

In this study, the dataset used facial data taken directly by the backend application then cropped (height and width) to a size of 108x108 and stored in the database. The dataset consists of 255 facial images taken by students where each student is automatically taken with 51 facial images in various positions. Files are stored in the database in jpg format with a gray level of 256 pixels [8].

According to the explanation in Figure 2, the backend program retrieves sample training data for students' faces. The system performs this task automatically, retrieving up to 51 data points, which are subsequently stored in the database. The facial data training process is then carried out for all registered facial data, where the system will determine the height (a) and width (b) of the face shape. Once facial data training is finished for all registered facial data, the results will be saved in the form of a .CLF file.

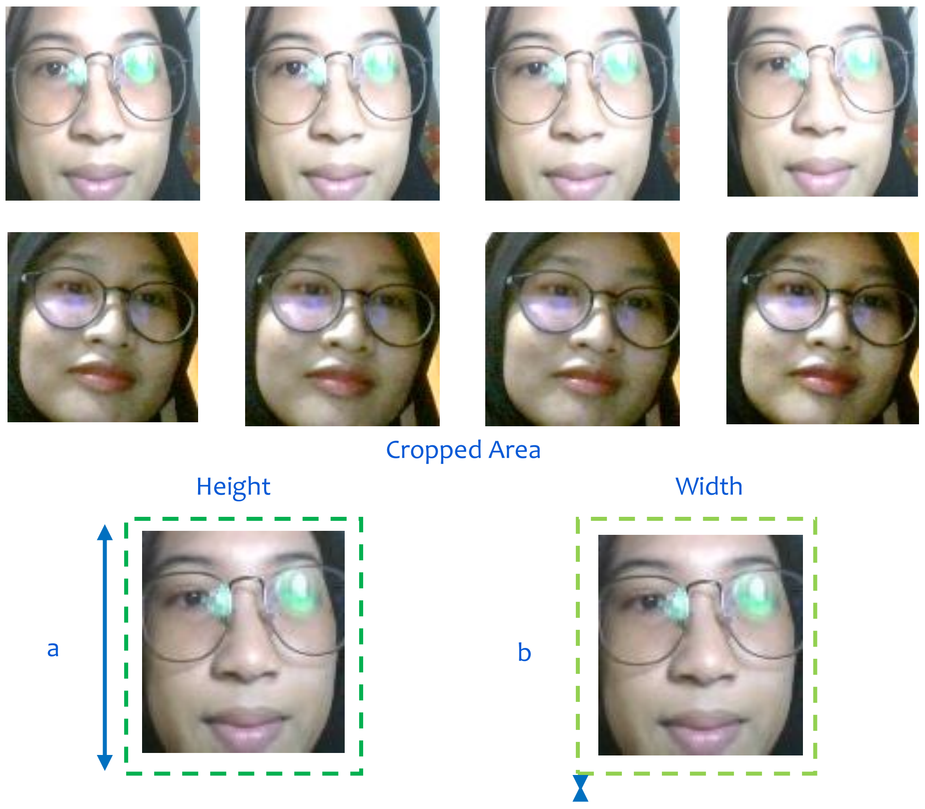

Figure 2. Example datasets

System design

Two training and testing methods are shown in Figure 3, followed by preprocessing, feature extraction, and feature selection. The outcome of the LDA feature selection is converted to a KNN feature and saved in the database as the subsequent phase. Reload the features and indexes from the LDA extracted database to do testing. To achieve accuracy, precision, recall, and 1 k-fold, the classification procedure is finally carried out using KNN.

Preprocessing

The preprocessing procedure includes splitting, scaling, softening, and normalizing the image, among other operations. The data is separated into numerous portions with the goal of using the split data as training and test data. Every feature in the image has a wide range of values, therefore image resizing, image reduction, and image normalization are all done to change the resolution, dimensions, and range of values. Every procedure is carried out by the system.

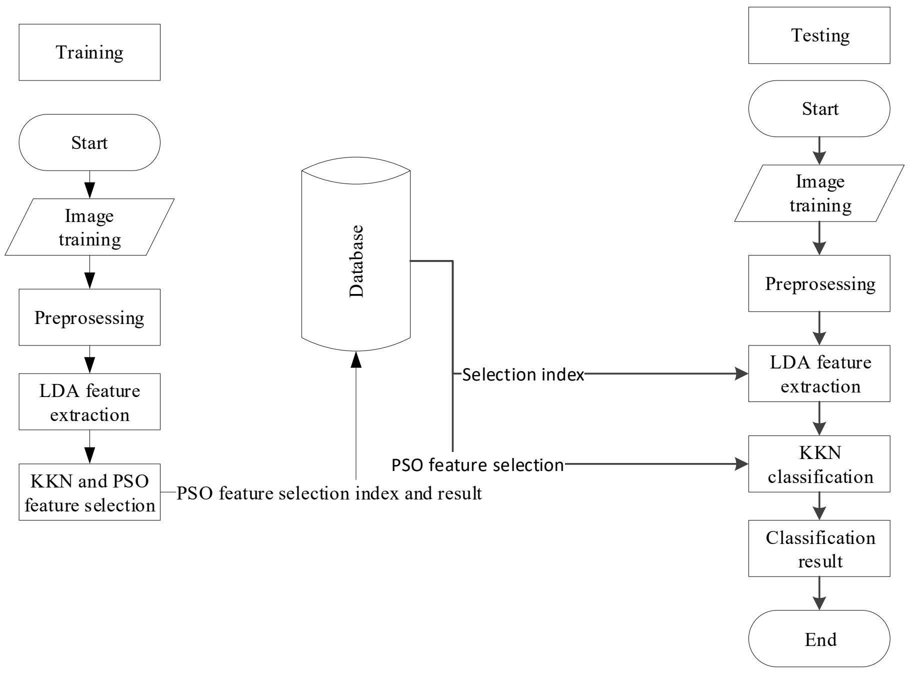

Figure 3. System design [7]

LDA feature extraction

The use of the Linear Discriminant Analysis (LDA) [9] method is a method with good discriminative properties for estimating a subspace linearly [7]. This method is a method for performing feature extraction which works based on the analysis of a distribution matrix to find the optimal projection. So, this LDA method maximizes the spread between classes and minimizes the spread within the face class. Differences between classes are represented by scatter between classes (Sb) and differences within classes are represented by scatters within class (Sw). The stages of the feature extraction process using the LDA method:

1. Convert a two-dimensional matrix to a one-dimensional matrix or convert it to row and column vectors.
2. The training data is grouped into several class matrices (X_i).
3. Calculating the average value (mean) of each class (μi) with Eq. (1), if the data is in the form of a row vector, then the calculation of the mean is calculated based on the row, so then the mean dimension is equal to one dimension one with training data not dataset dimensions.

$$\mu_i = \frac{1}{N_i} \sum_{X \in w_i} x \tag{1}$$

4. Calculating average of all classes (μ) with Eq. (2) average:

$$Mean: \mu_i = \frac{1}{x_i + \cdots + x_c} \sum_{X \in w_i} x \tag{2}$$

5. Calculating scatter matrix:

$$S_b = \sum_{i=1}^{c} N_i \,(\mu_i - \mu)^T (\mu_i - \mu) \tag{3}$$

$$S_w = \sum_{i=i}^{c} \quad \sum_{j=1}^{Ni} \quad ((\mu_j - \mu_i)^T \, (\mu_j - \mu_i) \tag{4}$$

6. Calculating the value of the covariance matrix (C) with Eq. (5)

$$C = S_b * (S_w)^{-1} \tag{5}$$

7. Calculating the eigenvalues (λ) and eigenvectors (v) with Eq. (6)

$$C_v = \lambda \, v \tag{6}$$

8. Projecting the original image with eigenvectors selected from the largest to the smallest eigenvalues as many as n-1 eigenvectors, where n is the number of classes with Eq. (7)

$$V = V^T \, X^i \tag{7}$$

Feature selection with particle swarm optimization

Particle Swarm Optimization [10] is one of the algorithms utilized that is motivated by how birds find food. Assuming how birds find food in a given area, this algorithm demonstrates how the optimal approach for birds to find food is to follow the nearest bird.

Algorithm for optimizing particle swarms

An algorithm known as particle swarm optimization operates on the presumption that a population of particles, each of which represents a point in the issue to be addressed, exists. When a particle crosses the search field, this procedure is where the best adjustments are made—both to the particle itself (local best) and to the position of all particles worldwide (global best). Regarding the PSO algorithm's steps [11] as below:

1. The particle's initial starting velocity
2. Calculate the particle's fitness value using the Global Best (Gbest) and Local Best (Pbest) values.
3. Determine the Pbest value before and after the iteration, if the fitness value of the calculated particle is greater than the previous Pbest, then it is made the newest Pbest
4. Calculate the Pbest value before and after the iteration; if the calculated particle's fitness value is higher than the previous Pbest, it becomes the newest Pbest.
5. Determine the Gbest value by finding the highest fitness value from the Pbest value.
6. Update the velocity which aims to determine the direction of movement of a particle that is in a certain population according to Eq. (8), then determine the inertial weight with Eq. (9).

$$V_j^i = WV_j^i + C_1.rand_1 \, x \left(Pbest_j - X_j^i\right) + C_2.rand_2 \, x \left(Gbest_j - X_j^i\right) \tag{8}$$

$$W = Wmax - \frac{W_{max} - W_{min}}{iter \; max} \; x \; iter \tag{9}$$

7. Update the values for each particle using an Eq. (10 that resembles a sigmoid). In this algorithm, the feature that is chosen is represented by a binary digit with the value 1, and the feature that is not chosen is represented by a value of 0.

$$ig(V_{i,j}^t) = \frac{1}{1+e^{-v_{t,j}^t}} \tag{10}$$

The velocity change value is compared to random values using the sigmoid value that was acquired. The most recent sigmoid position is valued 1 if the sigmoid value is bigger than the random value, and vice versa if the random value is tiny, as in Eq. (11).

$$X_{i,j}^{t+1} = \begin{cases} 1, & if\ Sigmoid > rand\ (0,1) \\ 0, & Otherwise \end{cases} \tag{11}$$

8. Up until the requirements are satisfied or converge at locations b, c, d, e, and f.

The PSO algorithm's iteration process will come to an end if the condition reaches its maximum value, the iteration fails to find a solution, it fails to converge, the normalized population radius value approaches zero, and the value of the function object graph approaches zero, according to the iteration.

K-nearest neighbor (KNN) method

Use the K-Nearest Neighbor (KNN) Method [12][13] is one of the approaches for generating decisions used in the classification of objects based on learning information that is closest to the item. This KNN method's objective is to categorize new objects using their attributes and training data samples, which are subsequently saved in vector form from a data classification feature. With the Euclidean distance formula, determine the class's shortest distance. The KNN method's steps are:

1. Count the number of closest neighbors, or the variable K.
2. Using the formula equation (12), calculate the separation between the photographs under test and the images in the database as follows:

   euclidean distance $d(x,y) = \sqrt{\sum_{i=1}^{n} \quad (X_i - Y_i)^2}$ (12)
3. Groups all items according to which group has the shortest Euclidean distance.
4. Gather categories from the next neighbor's classification, Y.
5. The value of a query instance can be included in forecasting.

Testing approach

The testing phase is the time to determine whether the system is functioning properly or not and to identify system flaws when a testing error arises. The accuracy, recall, and precision numbers are calculated during the testing phase using the confusion matrix method. The confusion matrix can be calculated using the equation below:

$$Accuracy = \frac{data\ overall\ from\ all\ targets}{data\ overall} \tag{13}$$

$$Precision = \frac{The\ volume\ of\ data\ for\ one\ class\ that\ is\ targeted}{The\ overall\ sum\ is\ appropriate} \tag{14}$$

$$Recall = \frac{The\ volume\ of\ data\ for\ one\ class\ that\ is\ targeted}{the\ volume\ of\ data\ for\ a\ single\ class} \tag{15}$$

$$F\ 1\ Score = \frac{2\ x\ precision\ value\ x\ recall\ value}{presision + recall} \tag{16}$$

Results and Discussion

Device specifications for implementation

Use the following device required specifications when implementing machine learning optimization tests on the KNN, LDA, and PSO algorithms:

1. Hardware
   a. Intel Core^{TM} i3-6006U CPU @2.00GHz 1.99 GHz
   b. RAM 4.0 Ghz
   c. 64-bit Operating System
   d. HDD 500 Ghz
2. Software
   a. Windows 8.1 Pro
   b. Python Environment (.env)
   c. Python 3.8
   d. Numpy
   e. Matplotlib
   f. Sklearn
   g. Keras
   h. cmat2scores
   i. Pandas

Results of the LDA feature extraction test using the KNN from earlier studies

Previous studies testing the LDA feature extraction with KNN produced values for accuracy, precision, and recall, as shown in Table 1. The accuracy, precision, and recall results at the value of k = 1 produce the maximum test value and faster calculation time.

Table 1. Based on studies, the outcomes of LDA and KNN testing [7]

k	Accuracy (%)	Precision (%)	Recall (%)	Calculating time (s)
1	70.00	71.42	70.00	0.2233
3	61.67	64.40	61.67	0.2268
5	62.50	57.43	62.50	0.2400

Use of the PSO algorithm in testing

Employing the PSO algorithm for feature selection testing to ascertain the particles are initialized in this function, which fills the particle position values with beginning positions that have been established, as illustration testing used

```
<A_Rec> E:\A_Rec\Nearst\mchlearning.py
Result : 'model'
        Best score : 0.9666666666666667
        Best parameter :
                'n_estimators'       : 85
                'max_depth'          : 11
                'min_samples_split'  : 3
```

```
'min_samples_leaf'   :10
Random seed: 440001612.
```

The results of the optimization with the Particle Swarm Optimization (PSO) algorithm are as follows:

1. Evaluation time : 17.304224 [99.86 %]
2. Optimization time : 0.02424 [0.14 %]
3. Iteration time : 17.3284 [2.89 iter/sec]

Testing the KNN and KNN+PSO algorithms

In testing the accuracy, precision, recall and k-fold score F1 by not using the Particle Swarm Optimization (PSO) algorithm in Figure 4 and by testing the K-NN + PSO algorithm as below:

```
<A_Rec> E_Rec\Nearst> measure.py
                Accuracy     :    0.9466666666666667
                Precision    :    0.9441269841269841
                Recall       :    0.9420289855072465
                F1           :    0.9419191919191919
```

The results of the confusion matrix test after using the PSO algorithm with an accuracy value of 94.66%, precision 94.41%, recall 94.20%, k-fold score F1 94.19.

```
<A_Rec> E_Rec\KNN-CI> measure.py
                Accuracy     :    0.9777777777777777
                Precision    :    0.9722222222222222
                Recall       :    0.9761904761904763
                F1           :    0.9731615673644659
```

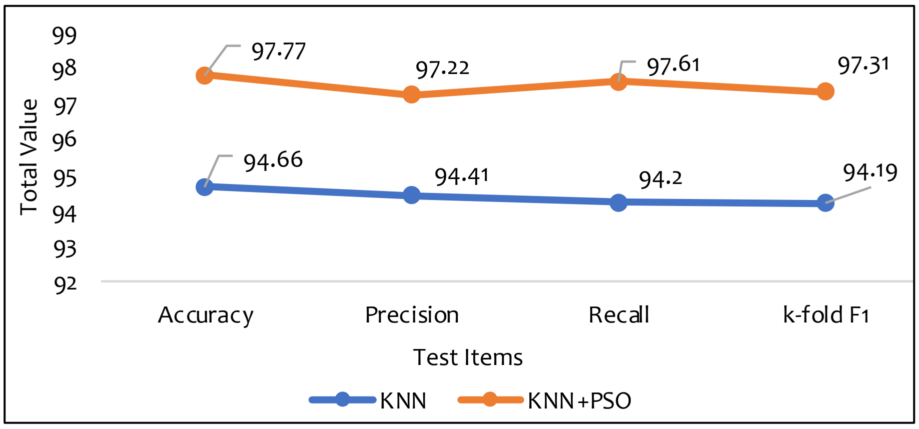

Figure 4. Algorithm optimization comparison graph on KNN and K-NN+PSO

The results of the confusion matrix test after using the PSO algorithm with an accuracy value of 97.77%, precision 97.22%, recall 97.61%, k-fold score F1 97.31. For different points in testing the K-NN algorithm with K-NN + PSO as shown in Figure 4. This result might be from the effect of PSO tendency to cluster solutions to the most likely optimum place although it might be not a global optimum. Also, it might be from the PSO parallel execution character.

Figure 4 illustrates the differences in the optimization of the testing algorithms for KNN and K-NN+PSO with respect to testing accuracy, precision, recall, and k-fold, which are 3.11, 2.81, 3.41, and 3.12, respectively.

Testing front-end applications (mobile devices)

The following are the outcomes of testing with mobile device applications as in Table 2 about testing on actual devices employing presence applications that employ machine learning techniques using facial recognition.

Table 2. Testing result of mobile devices

No	Modul Items	Feedback	Inference time (sec)	Device Name	Status
1.	Attendance Process with Face Recognition	The device has the ability to recognize student faces and take attendance.	4 ~ 5 sec	Samsung J7 Core (Android 7), ram 1Ghz	accepted
2.			1 ~ 2 sec	Oppo A16 (Android 11), ram 2 Ghz	accepted
3.	A screen presence indicator	Data about attendance can be shown on the gadget.	± 1 sec	Samsung J7 Core (Android 7), ram 1Ghz	accepted
4.			± 1 sec	Oppo A16 (Android 11), ram 2 Ghz.	accepted

Conclusion

The following is the outcome of combining K-Nearest Neighbor (K-NN) and Particle Swarm Optimization (PSO), which is based on study results on optimizing machine learning-based facial recognition with the Linear Discriminant Analysis (LDA) method: 1) In the confusion matrix test, the accuracy, precision, recall, and k-flod F1 values for linear discriminant analysis and K-NN with Particle Swarm Optimization (PSO) were 97.77%, 97.22%, and 97.61%, respectively. 2) With the difference in testing, the optimization of the algorithm for KNN and K-NN+PSO is increased, with the following results: accuracy 3.11, precision 2.81, recall 3.41, and k-fold 3.12. Results from tests using two different types of mobile devices for machine learning-based attendance systems with facial recognition, specifically: 1) Functionality Samsung J7 Core (Android 7), 2) Facial recognition inference method for attendance: 4–5 sec. Display: + 1 second for attendance. 4) OPPO A 16 (Android 11) performance: 5) Facial recognition inference method for attendance: 1–2 sec. Display attendance plus one second. As this field of research is increasing steadily, we will add more variance and options to see if they will add a better result on the problem of face and biometric recognition in the future.

Acknowledgement

We thank STT Muhammadiyah Cileungsi and LPPMPK for facilitating research activities and publishing articles.

References

[1] Widodo Budiharto and D. Suhartono, *Artificial Intelegence Konsep dan Penerapannya*. Jogjakarta: Andi, 2015.

[2] J. Y. Ng, F. Yang, and L. S. Davis, "Exploiting Local Features from Deep Networks for Image Retrieval," *Comput. Vis. Found.*, 2015.

[3] M. S. Sarma, Y. Srinivas, M. Abhiram, and L. Ullala, "Insider Threat Detection with Face Recognition And KNN User Classification," *2017 IEEE Int. Conf. Cloud Comput. Emerg. Mark. Insid.*, 2017, doi: 10.1109/CCEM.2017.16.

[4] G. Hermosilla *et al.*, "Particle Swarm Optimization for the Fusion of Thermal and Visible Descriptors in Face Recognition Systems," *IEEE Access*, vol. 6, 2018, doi: 10.1109/ACCESS.2018.2850281.

[5] Y. Xin *et al.*, "Machine Learning and Deep Learning Methods for Cybersecurity," *IEEE Access*, vol. 6, pp. 35365–35381, 2018, doi: 10.1109/ACCESS.2018.2836950.

[6] J. Xiao, G. Jiang, and H. Liu, "A Lightweight Face Recognition Model based on MobileFaceNet for Limited Computation Environment," *EAI Endorsed Trans.*, vol. 7, no. 27, pp. 1–9, 2022.

[7] M. H. Ramdani, I. G. P. S. Wijaya, and R. Dwiyansaputra, "Optimalisasi Pengenalan Wajah Berbasis Linear Discriminant Analysis Dan K-Nearest Neighbor Menggunakan Particle Swarm Optimization," *J. Teknol. Informasi, Komput. dan Apl.*, vol. 4, no. 1, pp. 40–51, 2022.

[8] Siti Khotimatul Wildah, S. Agustiani, Ali Mustopa, Nanik Wuryani, Hendri Mahmud Nawawi, and Rizky Ade Safitri, "Pengenalan Wajah Menggunakan Pembelajaran Mesin Berdasarkan Ekstraksi Fitur Pada Gambar Wajah Berkualitas Rendah," *INFOTECH J. Inform. Teknol.*, vol. 2, no. 2, 2021, doi: 10.37373/infotech.v2i2.189.

[9] N. Cintisa, E. Suhartono, and S. Aulia, "Pengenalan Ekspresi Pada Raut Wajah Pada Keselamatan Berkendara Menggunakan Principal Component Analysis (PCA) Dan Linear Discriminant Analysis (LDA)," *e-Proceeding Eng.*, vol. 6, no. 3, pp. 10292–10300, 2019.

[10] Y. Liu, G. Wang, H. Chen, H. Dong, X. Zhu, and S. Wang, "An Improved Particle Swarm Optimization for Feature Selection," *J. Bionic Eng.*, vol. 8, no. 2, pp. 191–200, 2011, doi: 10.1016/S1672-6529(11)60020-6.

[11] S. B. Sakri, N. Binti, A. Rashid, and Z. M. Zain, "Particle Swarm Optimization Feature Selection for Breast Cancer Recurrence Prediction," *IEEE Access*, vol. 6, pp. 29637–29647, 2018, doi: 10.1109/ACCESS.2018.2843443.

[12] X. Fu, J. Lu, and X. Zhang, "Intelligent In-vehicle Safety and Security Monitoring System with Face Recognition," *2019 IEEE Int. Conf. Comput. Sci. Eng. IEEE Int. Conf. Embed. Ubiquitous Comput.*, pp. 225–229, 2019, doi: 10.1109/CSE/EUC.2019.00050.

[13] M. A. Abuznied and A. Mahmood, "Enhanced Human Face Recognition Using LBPH Descriptor , Multi-KNN , and Back- Propagation Neural Network," *IEEE Access*, vol. 3536, no. c, pp. 1–11, 2018, doi: 10.1109/ACCESS.2018.2825310.