J-COSINE (Journal of Computer Science and Informatics Engineering)
Vol. 8, No. 2, December 2024 E-ISSN:2541-0806
Accredited Sinta-4 by RISTEKDIKTI Decree No. 79/E/KPT/2023 P-ISSN:2540-8895

Boosting Time Efficiency in Helmeted Face Recognition with Transfer Learning and k-NN

Dena Indiarto Permadi[1]*, Ida Bagus Ketut Widiartha[1], Santi Ika Murpratiwi[1], Tran Thi Thanh Thuy[2]
[1]Dept Informatics Engineering, University of Mataram
Jl. Majapahit 62, Mataram, Lombok NTB, INDONESIA

[2] Hanoi School of Business and Management
Hanoi, Vietnam
Email: denapermadi24@mhs.unram.ac.id, *:* [widi, santiika]@unram.ac.id, thuyttt@hsb.edu.vn

Corresponding Author

Abstract **This research aims to optimize the retraining process of face recognition systems for motor vehicle parking areas using Transfer Learning and k-nearest Neighbors (k-NN). The system enhances efficiency by retraining only on new classes, identified by checking the encoding file, allowing it to skip pre-existing classes, thus significantly improving retraining speed. Face data from 100 individuals (2725 images) was collected and augmented, with the VGG16 architecture used to generate face embeddings through Transfer Learning. Achieving 95.62% accuracy, with a precision, recall, and F1-score of 0.96 at k=1, the system demonstrates high accuracy and speed in identifying registered faces while effectively rejecting unknown ones. These optimizations greatly increase the feasibility of face recognition in parking security systems, making them more responsive to new data. Future research should focus on increasing dataset diversity and exploring more complex conditions, such as varied lighting, to enhance system performance and robustness further.**

Key words**: Face Recognition, Transfer Learning, k-Nearest Neighbors, Boosting re-training speed, Parking Security System**

I. INTRODUCTION

Nowadays, the development of the world of technology is growing very rapidly. Technology has been widely applied to daily activities, including parking lots. Usually, to protect vehicles from theft, many vehicle owners install additional security. One is adding a lock to the vehicle's disk, such as a locking padlock. Several people have conducted research to improve facial recognition security to prevent theft, as each human face has unique characteristics [1]. Therefore, face recognition systems have become increasingly popular as a way to improve security and productivity. By using a face recognition system, the security of the vehicle parking lot will be improved, and the occurrence of unwanted things will be reduced.

In 2023-2024, vehicle loss in parking lots is still a significant problem in Indonesia. For example, cases of lost vehicles in official parking lots, such as shopping centers and other public places, show that many customers have difficulty getting compensation from parking managers, even though they have complied with existing parking rule [2]. This makes using facial recognition systems very important for improving security in parking lots.

Each face has unique characteristics that need to be recognized. This identification can be used to unlock a safe containing goods. One of the human parts that has unique characteristics is the face. Faces can be used to recognize a person, such as attendance needs, population data collection, and security systems using facial recognition systems. This is because faces have different characteristics, which make security systems difficult to penetrate, such as lighting, skin color, haircuts, glasses, and different facial positions when looking down, turning their heads, or looking up [3]. Recognizing the face of a motorcycle rider using a helmet is challenging, as the area of the face that the system can realize is more limited. This reduces the amount of visual information available, making it easier for the system to identify accurately. A more robust method is needed to handle such cases so that face recognition remains effective even when most of the face is covered by a helmet. Since faces have their characteristics, using face recognition systems in parking lots will effectively solve the problem of lost vehicles in parking lots.

One of the main challenges in applying facial recognition to parking security systems is recognizing individuals wearing helmets. Helmets obscure significant portions of the face, making it difficult for traditional systems to identify individuals accurately. Although several methods have been proposed to address this issue, such as using advanced CNN architectures or fine-tuning models to handle occlusion, these solutions often come with increased computational costs. A more efficient solution is needed to balance accuracy and computational performance.

When designing a face recognition system, the thing that must be considered is that we need to train a computer to recognize a face so that the system can work properly. The ability of computers is different from that of humans to recognize one human face from another easily. As a result, face recognition techniques become complicated. To identify and compare the similarity of two faces, computers need training and test face data [4]. When the system can already work, there is often a problem that when new face data is entered into the system, the system will carry out a retraining process, which can be burdensome and time-consuming for the system to work correctly.

An effective retraining process is very important in face recognition systems, especially when new face data must be added to the system. The effectiveness of retraining affects the overall speed and performance of the system. If new data is added every time, the system has to retrain from scratch. This can take a long time and burden computing resources, ultimately reducing efficiency and productivity. Therefore, a smarter and more efficient retraining mechanism is needed, which involves only new data without having to repeat the entire process from the beginning. With an effective retraining approach, the face recognition system will be more responsive, quickly adapt to data changes, and be able to maintain optimal performance without sacrificing system time or capacity. This is especially important for security applications that require fast and accurate responses, such as parking lot surveillance, where fast face detection and identification can improve vehicle security.

This research focuses on optimizing the retraining process by combining transfer learning with k-NN, an approach chosen to balance accuracy and computational efficiency. Unlike SVM or deeper CNNs, k-NN requires minimal parameter tuning. It offers a straightforward mechanism for incremental updates, making it ideal for dynamic environments where new data is frequently added. Through a pre-trained VGG16 model, transfer learning ensures robust feature extraction even with limited training data. This combination effectively reduces retraining time while maintaining high accuracy.

II. Literature Review

A. *Related Research*

This research draws from several other studies that emphasize the difficulties in facial recognition efficiency, especially when retraining big datasets and improving accuracy for security reasons. Building on past investigations, this research aims to improve the accuracy of facial recognition in a parking system and optimize the retraining process.

As in the research that has been done by [5], in this study, the results of training on face data were found to be less efficient. During the retraining process, the system will retrain new data along with data that has been previously trained. This makes the retraining process, especially for retraining systems with large classes or data, long, so it will burden the performance of the system. Therefore, this research aims to optimize the retraining process to make it more efficient so that it does not take time and burden the system later.

Research [6] uses the eigenface algorithm for face recognition systems in vehicle parking lots. The results of the system training data will be stored in the database, which will then be used as a medium to recognize the faces of motor vehicle owners. In the test data results, motor vehicle owners can only enter when the face recognition rate is above 70%. The results of this study indicate that the eigenface algorithm can be an effective solution to improve the security and efficiency of the parking system by ensuring that only authorized vehicle owners can access the parking area.

Research [7] In this study, the EfficientNet-B0 architecture with transfer learning and fine-tuning methods was proposed for facial expression recognition models. The experiment was conducted on seven pre-trained Deep Convolutional Neural Network (CNN) architectures. The CK+ and JAFFE, facial expression datasets, were used, with a frontal profile view and input image size of 224 x 224. The recommended methods demonstrated high expression recognition accuracy; EfficientNet-B0 achieved 99.30% testing accuracy on CK+, and VGG-19 achieved 100.0% testing accuracy on JAFFE.

Research [8] whose test results show that the LBPH (Local Binary Pattern Histogram) method can be used to detect faces. With parameters Neighbors = 9, Gridx. Gridy = 8x8, radius = 1, buffer body size 180, and distance 20-25 cm; this method shows an accuracy of 95.56 and a computation time of 2.35 seconds.

Based on what has been described above, the authors researched optimizing the retraining process and increasing the accuracy of the face recognition system using the k-Nearest Neighbors (k-NN) method in the parking lot at the University of Mataram. This research aims to improve the security and efficiency of parking management by utilizing facial recognition technology.

B. *Supporting Theory*

The following are general theories that serve as foundational support for this research:

B.*1. Digital Image Processing*

Digital image processing is a field of science that studies image processing methods; the image in question is a still image (photograph) or a moving image (such as a recorded video). "Digital" refers to digital image or image processing using a digital computer. RGB stands for Red, Green, and Blue, three primary colors usually used as a reference for other colors. Using the RGB base, we can convert colors into numeric codes that make them appear universally identical. Computers already package color information into the same color model, making RGB color processing easy [9].

B.*2. Face Recognition*

Face Recognition is a technology that can identify faces by determining the face's location, size, and features while ignoring the background image and then identifying the face. Many variables influence this facial recognition, including source images, processed images, extracted images, and individual profile data. In addition, it is necessary to have sensing tools such as camera sensors and techniques to identify whether the images captured by the webcam are of human faces and to identify profile information corresponding to those facial images. Facial recognition is used by various organizations, including civilian, police, and military, to ensure that a person can be identified and to control physical access. Facial recognition is important for various purposes, such as surveillance systems, automatic tagging, and human-robot interaction [10]. With the advancement of technology, this facial recognition system is expected to become increasingly accurate and efficient in various conditions, enhancing security and convenience in various everyday applications.

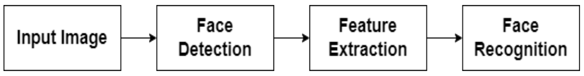


Fig. 1. Face Recognition Block Diagram [10]

Figure 1 shows the block diagram of the face recognition method, which consists of three parts: face detection, feature extraction, and face recognition [10].

B.*3. Transfer Learning*

Transfer learning is a technique in machine learning that utilizes a pre-trained model to complete a task, then applies it to another task related to the initial task. This technique is very useful when the available dataset is limited. In the transfer learning process, the existing model is used as a starting point by freezing some of the convolutional layers at the beginning of the model. Only the final layers are adjusted and retrained to fit the new task, such as classification. This approach allows for utilizing features learned from previous data and adapting them for new problems, thereby saving time and training resources [11].

B.*4. KNN (K-Nearest Neighbor) Method*

The basic principle of the k-Nearest Neighbors (kNN) algorithm for classifying objects is that similar objects should be located sufficiently close to each other. Geometric distances such as Euclidean are usually used to calculate distance. However, with the correctly chosen metrics, such as Manhattan or cosine, it is possible to achieve optimal classification results and accuracy [12]. The selection of the appropriate distance metric can significantly impact the performance of the kNN model, making it essential to consider the characteristics of the data when determining the most suitable metric.

The steps to calculate using the k-NN method are as follows [13]:

- Determine the parameter K (the number of nearest neighbors).
- Calculate each object's squared Euclidean distance (query instance) against the provided sample data (new data).
- Then, sort the objects into groups based on the smallest Euclidean distance
- Gather the category Y, which is intended for nearest neighbor classification
- The value of the computed query instance can be predicted using the most frequent nearest neighbor category.

The K.N.N. algorithm has several advantages, including its robustness against training data that contains much noise and its effectiveness when the training data is large. However, k-NN also has some weaknesses. One of them is the need to determine the value of parameter K (the number of nearest neighbors) and the ambiguity in determining the type of distance and attributes that should be used to achieve the best results. In addition, k-NN has high computational costs because it requires distance calculations from each query instance to all training samples [13].

B.*5. OpenCV and Python*

The Open Computer Vision (OpenCV) library is an open-source project that focuses on image processing. This means that computers must have capabilities comparable to human visual processing. Many basic computer vision algorithms are available in OpenCV. In addition, OpenCV offers an object detection module that utilizes computer vision techniques.

Python is an interpreted, interactive, and object-oriented high-level programming language that can be used on almost all platforms, including Mac, Linux, and Windows. Its easy-to-understand syntax and ability to combine ready-to-use modules make it an efficient programming language with high-level data structures [14].

B.*6. Normalization*

Normalization is a procedure used to prepare data to meet user needs and support subsequent procedures with better results. The normalization process is crucial to ensure the input has the same value, meaning images are sized with the same pixels [15]. This also helps accelerate convergence during the model training process, as different features will not disproportionately influence the model parameters.

B.*7. Image Processing*

Image processing, also known as image processing, is a rapidly growing field widely used in various sciences and engineering. It relies heavily on visual perception. The data entered and information generated are images, so image processing processes existing images to produce better images with expected information [16].

III. RESEARCH METHODOLOGY

The following is the research method plan that will be carried out.

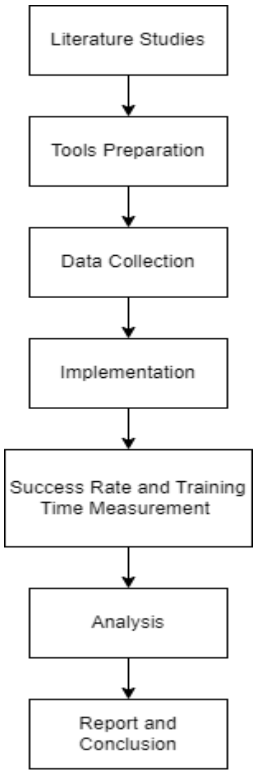


Fig. 2. Reseach Methods

Figure 2 shows that the research methods to be used in this journal consist of literature study, system requirements, data collection, implementation, analysis, reporting, and conclusions.

A. *Literature Studies*

At this stage, the author is searching for various references and theories as a foundation for preparing this journal. Several theories can support this research, namely the understanding of machine learning such as image processing, face recognition, OpenCV, and Python as media and programming language, the k-NN method, which is used as the method in the face recognition system, and other research sources or journals that can support the preparation of this research journal.

B. *Tools Preparation*

This stage will identify the hardware and software devices used in this research. The hardware used in this research is an Asus T.U.F. Gaming FX505DD laptop with an AMD Ryzen 5 3550H processor, 16GB of RAM, a 1TB HDD, and a 256GB SSD., which is utilized for testing the facial recognition system. The software used is VS Code.

C. *Data Collection*

The author collected data to serve as test data for the face recognition system. This process involved collecting facial images of 100 people, with a total of 2,725 image files. The data was obtained by photographing each respondent wearing a helmet and capturing some data via video. Some images were taken against a background containing many objects to simulate realistic parking lot conditions, while others featured a plain wall background. In addition, the dataset also includes a variety of lighting conditions (such as bright sunlight, dim light, and indoor lighting) and slightly different face angles to increase diversity. This comprehensive test data is essential for evaluating the system's performance and accuracy, ensuring the system can correctly recognize faces in various real-life scenarios. By using a diverse and representative data set, the authors could better analyse the system's robustness and identify areas for improvement, especially in handling real-world complexities.

D. *Implementation*

At this stage, the author implements the facial recognition system designed using previously collected data. The first step is to convert the image from BGR/RGB to grayscale. Once the image is changed to grayscale, the system will detect the faces present using haarcascade_frontalface_default.xml and then crop it according to the detected face in the image. After that, the cropped face image will undergo augmentation processes of four types for each class to increase the training data and strengthen the model by creating new variations of the existing images. The augmented data will then be divided into training data and testing data for the face recognition system, and subsequently, this data will be stored in a database. In this process, optimization will also be carried out on the re-training process to enhance the efficiency and performance of the facial recognition system. What is done is create a folder named new_data to store the data that will be processed temporarily, and then the data classes in that folder will undergo augmentation processing. After the augmentation process, the next step is to divide the class data into train and test folders. Once the data is divided, all data in the new_data folder will be deleted. Then, the data in the train and test folders will undergo the process of storing face encodings, which will be saved in a file in JSON format. After the encoding storage, the data in the train and test folders will also be deleted. The two folders are intended to store images for processing temporarily, and once completed, they will be deleted. The aim is to ensure that only newly entered data will be processed during the re-training encoding while avoiding reprocessing already completed data. In addition, to optimize the re-training encoding process, a check is performed on the JSON file to see if the data to be processed is already present in the JSON file. When a data class is found to already exist in the encoding JSON file, that data will not be processed again, ensuring that only new data is processed. With these processes, it will be possible to optimize the performance of the re-training encoding process that will be carried out by the system later. However, some aspects require further clarification for methodological completeness, specifically the choice of parameters and model architecture. For instance, the k=1 setting in the k-Nearest Neighbors (k-NN) classifier is used here, which implies that only the closest neighbor is considered for classification. This choice can be advantageous in facial recognition tasks where a strict one-to-one match is essential, although it could also be sensitive to outliers. Additionally, using the VGG16 architecture suggests leveraging pre-trained, deep feature extraction layers well-suited for detailed image representation. VGG16's layered depth, although computationally intensive, has been found effective in capturing the hierarchical structure of facial features, which is crucial for high-accuracy recognition. The flowchart for checking and storing classes in the encodings file is shown in the figure below.

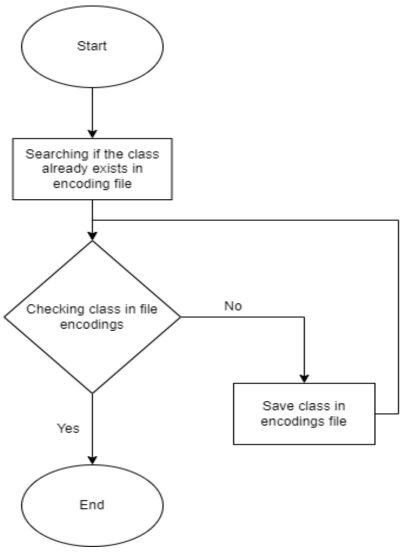

Start
Searching if the class already exists in encoding file
Checking class in file encodings
No
Save class in encodings file
Yes
End

Fig. 3. Flowchart of Checking and Storing Classes In Encodings

E. *Success Rate and Training Time Measurement*

In this stage, the author measures the success rate results and the time required during the training process on the dataset. The measurement is done by comparing the training process using all data and some data. The results of the two methods will be analyzed to determine the difference in performance in terms of accuracy and efficiency of the time required.

F. *Analysis*

After the previous process, the author analyzes the facial recognition system's results to determine whether the system's performance meets expectations and adheres to the established standards. This evaluation involves comparing the system's predicted results with the actual labelled data and calculating evaluation metrics such as accuracy, precision, recall, and F1-score. If the results are unsatisfactory, the author will adjust or refine the model, parameters, or techniques until the system achieves the desired performance.

G. *Report and Conclusion*

The final stage is to draw a comprehensive conclusion from the research analysis. This result includes the findings of the facial recognition system research and covers F1 scores, accuracy, precision, and recall. The conclusion also details the system's performance and discusses the advantages and disadvantages of the developed facial recognition system. Thus, the results will provide an overview of the success of the system that has been tested and what can be done to improve the system's performance in the future.

IV. Results and Discussion

Several facial data classes have been collected in the research stage, totaling 100 classes, examples of which can be seen in the image below.

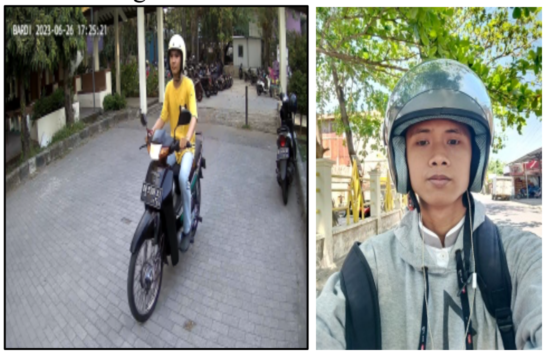


Fig. 4. Example of Face Dataset

The first step is preprocessing and augmentation, which increases the training data for each class before proceeding to the next process.

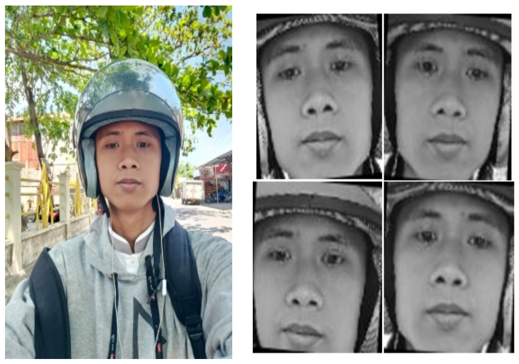


Fig. 5. Example of Dataset Augmentation Image

The dataset image to be taken will be just the face part, which will then be converted to grayscale, cropped, and resized to 100x100 pixels for consistency. Then, several augmentation processes will be carried out on the facial images, such as horizontal flipping, rotation, brightness adjustment, and Gaussian blur.

The next step is to split or divide the collected face class data into a folder named test and train, with a ratio of 30% for the test folder and 70% for the train folder. The comparison aims to provide a good balance in supplying sufficient data for the model to learn from while leaving enough data to evaluate how the model performs with new data. It should be seen in the example in the picture below.

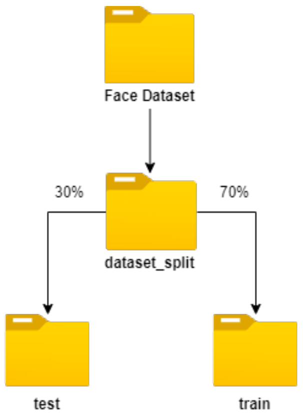
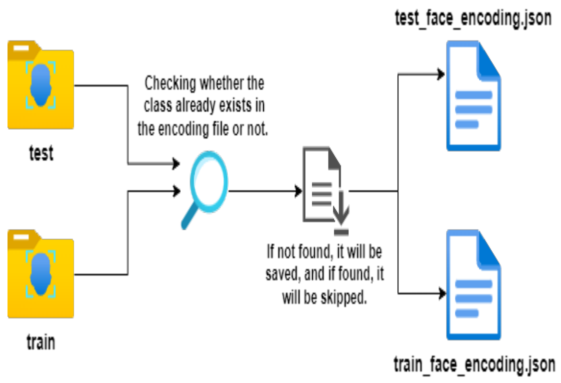


Fig. 6. Example of Folder Split Dataset

Then, the data will be divided into train and test, and a checking process will be performed for each class in the test and train folders. If a class is found in the encoding file, it will not be processed. If a class is not found, it will be saved in the encoding file. Once the data-splitting process is complete, the classes in the "Face Dataset" folder will be deleted.


test_face_encoding.json
Checking whether the class already exists in the encoding file or not.
test
train
If not found, it will be saved, and if found, it will be skipped.
train_face_encoding.json
Fig. 7. Class Search Process in Encoding Files

As can be seen in Figure 6, the checking process aims to save time in the train encoding process. If the class is found, the existing class will not be processed. The system will continue to check the next class until the class is found not to exist in the encoding file. The class will then be saved into the file. So, with this process, the system no longer needs to store class data that has been previously stored in the encoding file.

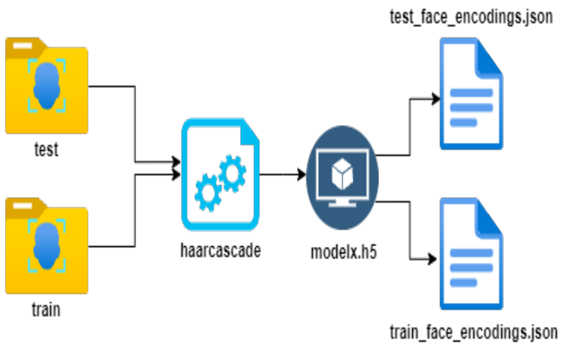


Fig. 8. Encoding Process of Test and Train Folders

After the checking process is complete, it can be seen in Figure 7 that there is an encoding process of the face data contained in the test and train folders. In this process, haarcascade_frontalface_default.xml is used to detect faces in the datasets contained in the test and train folders. After the face is successfully detected, the pre-trained model with the name model.h5 obtained from VGG16 will store the face in the folder to encode the face. This model will output a numerical representation (embedding) of the face recognition system's face data. In this process, a check is also made on the JSON file to see whether the data has previously been encoded or not. If the data has already been encoded, then the data will not be encoded again, and data that has not been encoded will be processed. This aims to make the process efficient and fast so that it does not take long. Encoding face data that has been successfully created will then be stored in a file in .json format with the file names test_face_encodings.json and train_face_encodings.json. We will later use this file for the system testing process. After the train encoding process is complete, the class data in the train and test folders will be deleted automatically.

Here are the results of the runtime experiments on the class data that have been done:

TABEL I. RUNTIME TESTING WITH 75 CLASSES

No	Tested Classes	Runtime
1.	75 Class	15 Minute 2 Second
2.	Add 1 Class	1 Minute 16 Second

The table for the project runtime of 99 classes can be seen in the table below:

TABEL II. RUNTIME TESTING WITH 99 CLASSES

No	Tested Classes	Runtime
1.	99 Class	19 Minute 58 Second
2.	Add 1 Class	1 Minute 38 Second

The table for the project runtime of 100 classes can be seen in the table below:

TABEL III. RUNTIME TESTING WITH 100 CLASSES

No	Tested Classes	Runtime
1.	100 Class	20 Minute 4 Second

When adding 1 class for the retrain encoding process, the runtime process is relatively the same, about 1 minute less. This result indicates that the retraining encoding process runs optimally and speeds up the retraining process compared to having to retrain all classes simultaneously, which is very time-consuming. Then, it can be seen in Table III, with 100 classes, that the resulting runtime is slightly faster when totaled with Table II, with 99 classes and one additional class. This is caused by the time when calling the transfer learning model, where when training 100 classes at once, the transfer learning model will only be called once, while if training twice, 99 classes and 1 class, the model will be called twice. This is why there is a slight difference in the runtime process in Table II and Table III. After seeing the results of the runtime experiment of the retrain encoding process, the retrain encoding optimization process was successfully carried out with faster results than the retrain encoding process that retrains all data from the beginning again.

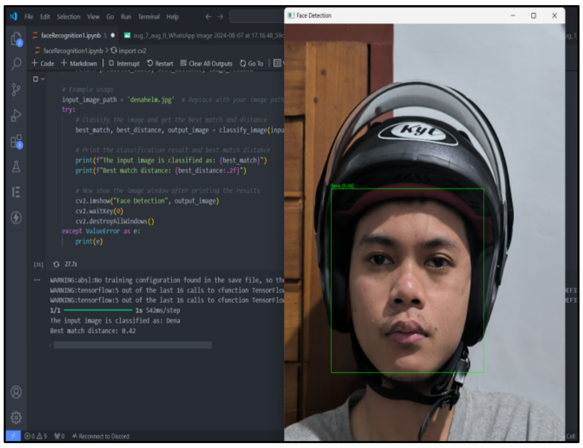
The following process is classification to see the system's performance, which was designed using the k-NN method with a value of k = 1. It can be seen in the following table:

TABEL IV. CLASSIFICATION RESULTS

No	Metric	Value
1.	Accuracy	95.62%
2.	Precision	0.96
3.	F1 Score	0.96
4.	Recall	0.96
5.	Total Predictions	708
6.	Correct Predictions	677

Table IV shows that the classification results with the k-NN method with a value of k = 1 that have been carried out provide excellent results for a face recognition system with a total accuracy of 95.62%, precision of 0.96, F1 score of 0.96, and recall of 0.96. In the table, it can also be seen that the total predictions made were 708, with the number of correct predictions being 677. This data confirms that the system performs excellently in recognizing faces with a relatively low error rate, showing the consistency and reliability of the system in various test conditions. This success reflects the effectiveness of the methods and parameters applied in the classification process.

The last stage is to test using an input image to determine whether the system recognizes or detects the person's face by the facial data entered into the encoding data. Before face detection is carried out, the system will convert the input image to grayscale, and then face detection will be carried out using haarcascade_frontalface_default.xml on the input image. Then, after the face is found, it will be displayed as in the following image:

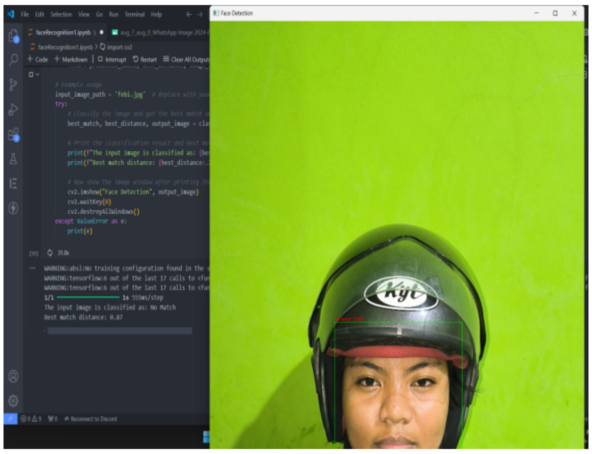

```
# Example usage
input_image_path = 'denahelm.jpg'  # Replace with your image path
try:
    # Classify the image and get the best match and distance
    best_match, best_distance, output_image = classify_image(inpu

    # Print the classification result and best match distance
    print(f"The input image is classified as: {best_match}")
    print(f"Best match distance: {best_distance:.2f}")

    # Now show the image window after printing the results
    cv2.imshow("Face Detection", output_image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
except ValueError as e:
    print(e)
```
```
WARNING:absl:No training configuration found in the save file, so th
WARNING:tensorflow:5 out of the last 16 calls to <function TensorFlo
WARNING:tensorflow:5 out of the last 16 calls to <function TensorFlo
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 542ms/step
The input image is classified as: Dena
Best match distance: 0.42
```
File Edit Selection View Go Run Terminal Help
faceRecognition1.ipynb
aug_7_aug_0_WhatsApp Image 2024-08-07 at 17.16.48_59
Code Markdown Interrupt Restart Clear All Outputs Go To
Face Detection

Fig. 9. Face Detection Testing 1

It can be seen in Figure 8 that the detection results find a face that matches the face in the encoding data, which shows that the photo is detected as a person named Dena with a distance value of 0.42. The trial conducted with the image shows that the system correctly recognizes the face according to its name.


```
# Example usage
input_image_path = 'febi.jpg'  # Replace with you
try:
    # Classify the image and get the best match a
    best_match, best_distance, output_image = cla

    # Print the classification result and best ma
    print(f"The input image is classified as: {be
    print(f"Best match distance: {best_distance:.

    # Now show the image window after printing th
    cv2.imshow("Face Detection", output_image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
except ValueError as e:
    print(e)
```
```
WARNING:absl:No training configuration found in the s
WARNING:tensorflow:6 out of the last 17 calls to <fun
WARNING:tensorflow:6 out of the last 17 calls to <fun
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 555ms/step
The input image is classified as: No Match
Best match distance: 0.87
```
File Edit Selection View Go Run Terminal Help
faceRecognition1.ipynb
Code Markdown Interrupt Restart Clear All Outputs
Face Detection

Fig. 10. Face Detection Testing 2

As can be seen in Figure 9, the detection results show that the system does not find a face that matches the face in the encoding data. This is by the desired results during the test process by trying to test the system to detect the faces of people whose facial data has not been stored. This success indicates that the facial recognition system has detected faces accurately and does not misidentify unknown faces. Thus, the system demonstrated its ability to distinguish between registered and unregistered faces effectively. The system has also successfully shown quite good results when conducting experiments using face images with helmets under certain conditions, such as lack of lighting. This is because, in the augmentation process, all facial data that has been collected has been applied to the brightness adjustment process, which will adjust/increase the respective lighting in the image so that the results of the face recognition process run well.

After the very satisfactory results of the system, there were also some minor problems, such as the system would occasionally detect people who had never been registered with the system before if the distance from the person's face was somewhat similar to the distance of the face contained in the encoding file. Although rare, the system sometimes struggles to distinguish between similar and registered faces, significantly when lighting conditions or viewing angles change. Such occurrences sometimes indicate weaknesses in face recognition and deserve further attention in future system development.

V. Conclusion and Recommendations

Based on the results of the analysis, optimizing the re-train process of the face recognition system that has been designed can work as expected. Namely, it can efficiently perform the re-train encoding process so that the re-train process no longer needs to re-train all facial data from the beginning. The results are also fast, re-training data as much as from 100 machines only takes 20 minutes and 4 seconds, while adding 1 class only takes an average of less than 2 minutes. This impacts the re-train process time to be faster and more efficient while maintaining high accuracy using the k-NN method k=1 value with an accuracy result of 95.62%. This system also successfully recognizes face data stored in the database and can recognize new face data that has never been registered in the database before.

Suggestions for further research should be better in finding datasets for faces by considering the quality of facial images and increasing the number of datasets so that the face recognition system is more effective to use with extensive data. Future research should also pay attention to several conditions when adding data in different lighting conditions, angles, or occlusion variations and examine more deeply the distance threshold of facial data so that later the system does not adjust the distance of faces that have never been stored before and are instead recognized by the system because of the distance.

References

[1] L. Pamboro and E. Setiawan, "Sistem Pengaman Kendaraan Roda Dua dengan Pengenalan Wajah menggunakan Principal Component Analisis," J. Pengemb. Teknol. Inf. dan Ilmu Komput., vol. 7, no. 5, pp. 2500–2506, 2023, [Online]. Available: https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/12735

[2] G. Satria and A. Ferdian, "Kendaraan Hilang di Area Parkir Resmi, Konsumen Bisa Tuntut Ganti Rugi," Kompas.com, 2023. https://otomotif.kompas.com/read/2023/02/14/160100015/kendaraan-hilang-di-area-parkir-resmi-konsumen-bisa-tuntut-ganti-rugi (accessed Aug. 28, 2024).

[3] G. A. Rama, F. Fauziah, and N. Nurhayati, "Perancangan Sistem Keamanan Brangkas Menggunakan Pengenalan Wajah Berbasis Android," J. Media Inform. Budidarma, vol. 4, no. 3, p. 635, 2020, doi: 10.30865/mib.v4i3.2149.

[4] D. Yulianti, I. Triastomoro, and S. Sa'idah, "Identifikasi Pengenalan Wajah Untuk Sistem Presensi Menggunakan Metode Knn (K-Nearest Neighbor)," J. Tek. Inf. dan Komput., vol. 5, no. 1, pp. 1–10, 2022, doi: 10.37600/tekinkom.v5i1.477.

[5] P. P. Kusuma, I. Bagus, and K. Widiartha, "Face Recognition on Motorcycle Rider Wearing Helmet Using Transfer Learning" in MIMSE, Mataram, 2023, pp. 1–9.

[6] N. M. Raharja, M. A. Fathansyah, and A. N. N. Chamim, "Vehicle Parking Security System with Face Recognition Detection Based on Eigenface Algorithm," J. Robot. Control, vol. 3, no. 1, pp. 78–85, 2022, doi: 10.18196/jrc.v3i1.12681.

[7] I. N. Alam, "Metode Transfer Learning Pada Deep Convolutional Neural Network (DCNN) untuk Pengenalan Ekspresi Wajah," ResearchGate, no. October, pp. 13–14, 2022.

[8] A. Fauzan, L. Novamizanti, and Y. N. Fuadah, "Perancangan Sistem Deteksi Wajah Untuk Presensi Kehadiran Menggunakan Metode LBPH (Local Binary Pattern Histogram) Berbasis Android," e-Proceeding Eng., vol. 5, no. 3, pp. 5403–5413, 2018.

[9] S. Ratna, "Pengolahan citra digital dan histogram dengan phyton dan text editor phycharm," Technologia, vol. 11, no. 3, pp. 181–186, 2020.

[10] D. Suprianto, R. N. Hasanah, and others, "Sistem Pengenalan Wajah Secara Real-Time dengan Adaboost, Eigenface PCA & MySQL," J. EECCIS, vol. 7, no. 2, pp. 179–184, 2014.

[11] A. E. Putra, M. F. Naufal, and V. R. Prasetyo, "Klasifikasi Jenis Rempah Menggunakan Convolutional Neural Network dan Transfer Learning," J. Edukasi dan Penelit. Inform., vol. 9, no. 1, p. 12, 2023, doi: 10.26418/jp.v9i1.58186.

[12] Anissa Ollivia Cahya Pratiwi, "Klasifikasi Jenis Anggur Berdasarkan Bentuk Daun Menggunakan Convolutional Neural Network Dan K-Nearest Neighbor," J. Ilm. Tek. Inform. dan Komun., vol. 3, no. 2, pp. 201–224, 2023, doi: 10.55606/juitik.v3i2.535.

[13] Danar Putra Pamungkas, "Ekstraksi Citra menggunakan Metode GLCM dan KNN untuk Indentifikasi Jenis Anggrek (Orchidaceae)," Innov. Res. Informatics, vol. 1, pp. 51–56, 2019.

[14] T. C. A.-S. Zulkhaidi, E. Maria, and Y. Yulianto, "Pengenalan Pola Bentuk Wajah dengan OpenCV," J. Rekayasa Teknol. Inf., vol. 3, no. 2, p. 181, 2020, doi: 10.30872/jurti.v3i2.4033.

[15] D. Satria Yudha Kartika and H. Maulana, "Preprosesing dan normalisasi pada dataset kupu-kupu untuk ekstraksi fitur warna, bentuk dan tekstur," J. Comput. Electron. Telecommun., vol. 1, no. 2, pp. 1–8, 2021, doi: 10.52435/complete.v1i2.76.

[16] T. K. SUWARDI, "Normalisasi Kemiringan Pada Citra Ktp Dengan Metode Momen," *J. Sains Dan Komput.*, no. x, 2019, [Online]. Available: http://journal.ukrim.ac.id/index.php/JIF/article/download/118/93

.