International Journal on Advanced Science Engineering Information Technology

Vol.12 (2022) No. 3
ISSN: 2088-5334

Face Recognition Application Based on Convolutional Neural Network for Searching Someone's Photo on External Storage

I Putu Arya Dharmaadi [a,*], Deden Witarsyah [b], I Putu Agung Bayupati [a], Gusti Made Arya Sasmita [a]

[a] *Department of Information Technology, Udayana University, Badung, Bali, 80361, Indonesia*
[b] *Department of Information System, Faculty of Industrial Engineering, Telkom University, Bandung, 40257, Indonesia*
Corresponding author: [*]aryadharmaadi@unud.ac.id

***Abstract*—Digital photos are often defined as personal archives collected long ago and are stored on a large enough storage media such as an external hard disk or flash disk. Problems arise when someone wants to find photos of themselves or others in tons of photo collections. Searching manually, such as opening a photo file or folder one by one, will certainly be very troublesome. Based on these problems, this study designed an application for searching certain photos based on the similarity of the inserted face photo. This application is built for computer or laptop devices, which was developed by using the Python programming language and *Dlib* module that applied the *face recognition* method through the combination of *Convolutional Neural Network* (CNN), *FaceNet Embedding*, and *Triplet Loss* for matching faces. The recognition scheme starts from *face detection*, *face alignment*, *face encoding*, and *face classification* stage. Our application is very handy to run in looking for particular face images on external storage compared to prior studies. We have done experimental research, demonstrating that the application can find almost all image files the user is looking for. In addition to the result in the form of an application, this study contributes to exploring the performance of the Dlib module, in terms of precision and recall rate, which could not recognize non-frontal face images well. We encourage other researchers to address this limitation in further studies.**

***Keywords*— Photo searching; convolutional neural network; face recognition; python; dlib.**

Manuscript received 13 Apr. 2020; revised 7 Apr. 2021; accepted 20 May 2021. Date of publication 30 Jun. 2022.

I. INTRODUCTION

Nowadays, the development of digital technology is growing rapidly so that digital data can be generated and stored, including digital photos. Digital photos technology has increased rapidly in the form of camera technologies that have sprung up, such as a pocket, DSLR, smartphone, and laptop cameras [1]. Technology makes it easy for everyone to take photos of every important moment with others [2]. Because the number and size are quite large, all digital photos are generally moved and stored on external storage media such as hard disk or flash disk.

Problems arise when someone wants to find photos of themselves or others' photos in a ton of photo collections [3]. Searching manually, such as opening a photo file or folder one by one, will certainly be very troublesome. Generally, file management systems or photo management applications on a computer or personal computer (PC) only provide limited search features, such as searching by name, size, file type, and date for the file creation [4]. Finding certain photos with this feature doesn't help much because photo file names are usually sequential. In this way, we need an application that can search for certain photos based on the inserted face photo. Some recent smartphones already have similar applications that can group photos based on the faces detected by the application. For example, an iPhone type smartphone has a 'Photos' application that can be used to search for facial photos. However, on a laptop or PC, this type of application is very rarely found. On the other hand, the inability of smartphones in reading hard disks or flash drives where millions of digital photos are usually long stored makes these photos have to be transferred first to a smartphone's internal memory to be read. The process of moving photos is certainly very annoying and time-consuming.

Based on these problems, we designed a photo search application that runs on a laptop or a PC device with the ability to read files on an internal hard disk or an external hard disk, or a flash drive. In order to do particular tasks such as understanding images and finding the right people's faces, the application needs a special method or algorithm that can

analyze pictures, distinguish a face from other objects, and recognize the faces that are sought. After conducted a literature review, we found that the face recognition method is the technique that is being researched by researchers for detecting the face and classifying it based on its feature extraction [5], [6].

The face recognition method is not completely new, but it has been around for a long time, around 1966 [7]. However, currently, this technology is growing rapidly because it is supported by drastically increasing computer capabilities. By using face recognition technology, we can verify (one-to-one matching) someone's face or identify (one-to-many matching) unknown faces on an image [8]. Three main modules make the face recognition system work properly. They are: face detector module for localizing faces in images; facial landmark detector module for aligning the faces to normalized canonical coordinate; and face recognition module for extracting face features and matching it [9]. Based on the review result of face recognition methods and techniques presented by Lal *et al.* [10], we found that the neural network method later combined with multi-stage image convolution layers is an effective facial recognition technique [11].

Therefore, in this research, we built an application applying a face recognition method named *FaceNet Embedding* approach that utilizes the concept of convolutional neural network and the mathematical formulation of *triplet loss*. This approach has been widely used by global technology companies such as Google and Facebook because of its excellent accuracy, close to 98% [12]. By utilizing this method, the built applications can provide accurate results with fast processing time.

Considering its vital role in the system being built, a study on face recognition needs to be discussed more deeply, especially on how it works. Face recognition is one branch of knowledge in computer vision that provides knowledge to computers to distinguish someone's face from others [13]. For decades ago, researchers have been trying to develop various techniques and tools to make computers able to recognize a person's face automatically. Based on the research results from Beham and Roomi [14], there are broadly three main approaches in building face recognition systems, namely appearance-based, feature-based, and soft computing-based. Among those approaches, the soft-computing approach, on artificial neural network concept, is the most popular approach widely used and developed by academics due to its impressive performance.

An artificial neural network is a technique that adapts the human brain to work and learn [15]. In order to give better results on image processing, the model is combined with multi-stage image convolution layers so that the combination concept is called a convolutional neural network (CNN) [16]. Besides reducing the number of input parameters into the neural network, the application of convolution series aims to extract the main features of an image [17]. As a result, CNN can learn the best parameters or weights from large datasets of images during the training stage to produce the expected results [18].

The workflow of the face recognition process based on CNN is described in figure 1. In the first step, the image will be analyzed using the *histogram of oriented gradients* (HOG) method to determine the face [19]. This step produces the pixel coordinates of a bounding box for each face. Second, every face found is explored further to determine the specific points or landmarks of the face, such as nose, eyes, mouth, etc. [20]. Based on the face landmarks, we crop and align the face so that the mouth and eyes are centered well in the face photo [21]. The output of the face alignment stage is the cropped face image with a fixed size, i.e., 96x96. The next step, face encoding, aims to extract the face to be a compact and discriminative feature vector called an embedding value [22]. The face encoding step utilizes CNN and applies *FaceNet embedding* and *triplet loss function* during the training so that the faces of the same person generate a similar embedding value, and vice versa [23]. In the last step, using a classification algorithm such as a support vector machine (SVM) to process the embedding value, we can distinguish someone's face from the others' [24].

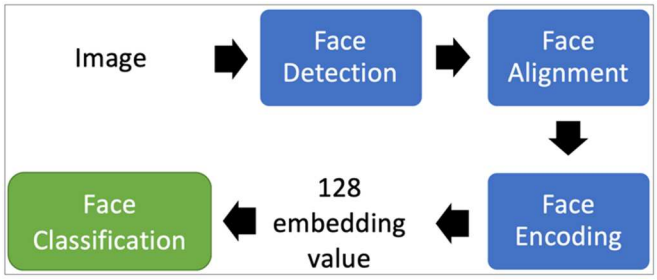

Fig. 1 Face recognition workflow

To produce the CNN model with the best parameters, we had to train it with very large datasets of face images that will be done in dozens of hours in a supercomputer. Luckily, some researchers had conducted the training process and shared their best models to compare the models' performance. One of the popular models is the *Dlib* library, written in *C++* and *Python* language, developed by [25]. Using the library, we can easily implement the face recognition workflow without creating and training the model from scratch because Dlib has provided it. This library utilizes the *ResNet-34* network developed by He *et al.* [26] as the network model with some modification, such as a few layers removed. The model has been trained with several datasets, such as the face scrub dataset, the VGG dataset, and others, with a total of 3 million face images so that we can just use it right away [27]. Currently, the library has released a 19.17 version and is still developing.

II. Materials and Method

We needed to do several processes to build a face recognition application properly. These stages were aimed to collect information and propose the best design fitting the requirements.

A. System Overview

According to the problems described in the introduction, we developed a desktop application that can find specific images in a hard drive based on the entered face photo. In order to fit into the *Dlib* library and run-on various desktop machines, this application was built in Python language. Thus, the computers that would run this application have to install a Python environment first. The two main tasks of the application were scanning all images in a drive and save the

results in the database; and searching for specific faces based on the scanning results.

To support the scanning images task, we built several functions, such as *choose_directory*, *scan_images*, *face_encoding*, and *save_the_result*. The *choose_directory* function works by asking the user to pick a folder that it wants to scan. This function utilized the API of the OS file manager and OS device manager provided by the operating system. The results, an array of various files stored in the folder, will be received by the *scan_images* function. This function will only sort files in the form of images in *bmp*, *png*, and *jpg* formats. Next, by utilizing the *Dlib* library, the pictures will be analyzed to find all human faces. The processes will produce the face location and the embedding values for every face that is successfully identified in images. By using the *SQLite3* library, the data output will be saved in the database. For more detail, see figure 2.

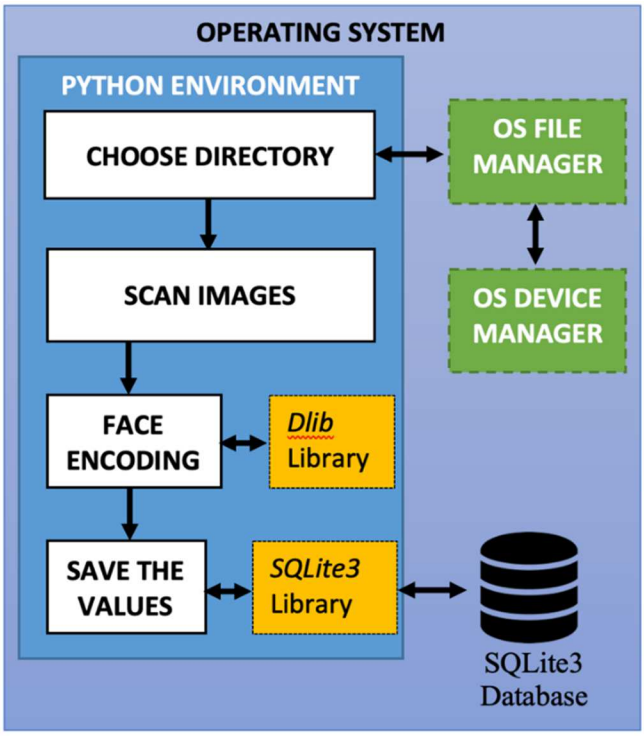

Fig. 2 System overview of scanning images

The next task is searching for specific faces. It is similar to the previous task, which is detecting dan extracting the image containing human faces. The difference is that the resulting embedding value is not stored, but it is compared with all embedding values saved before. If it is matched, the images that have those embedding values will be shown to the user. For more detail, see figure 3.

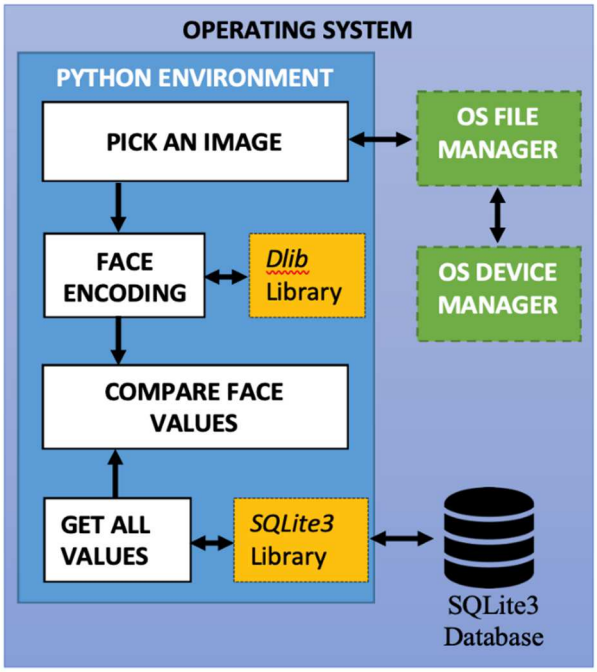

Fig. 3 System overview of searching for specific faces

B. Data Collection

To produce a system that can search for photos of one's face, digital images are needed as the images collection that will be targeted for face search. The images will be read and extracted through their main characteristics to produce embedding information stored in the database. Thus, in this stage, we gather some public images.

In this case, we used photos of football players from the Italian club, Juventus, which can be downloaded from sports news websites with various formats, such as *.bmp*, *.jpg*, and *.png*. The resolution and file size to be used are not limited. First, we downloaded profile pictures of top Juventus players as follows.

TABLE I
PEOPLE FACE BEING SEARCHED IN IMAGE COLLECTION

Number	Person's name	Initial	Photo
1	Ronaldo	R	
2	Dybala	D	
3	Pjanic	P	
4	Chiellini	C	
5	Buffon	B	

Next, we needed to download images containing the faces of the people above with various poses. Each downloaded image was further analyzed manually in terms of the person's identity in the picture. The result of the image collection that has been downloaded and analyzed is displayed below.

TABLE II
IMAGE COLLECTION

	Filename	Size	The face of				
			R	D	P	C	B
1	dybala-is-clapping.jpg	48 KB	-	✓	-	-	-
2	bernardeschi-is-shocked.jpg	70 KB	-	-	-	-	-
3	dybala-and-ronaldo.jpg	58 KB	✓	✓	-	-	-
4	pjanic-and-ronaldo.jpg	52 KB	✓	-	✓	-	-
5	dybala-try-to-hold-the-ball.jpg	48 KB	-	✓	-	-	-
6	ronaldo-is-heading.jpg	46 KB	✓	-	-	-	-
7	ronaldo-and-bonucci-hold-mandzukic.jpg	49 KB	✓	-	-	-	-
8	dybala-celebration.jpg	40 KB	-	✓	-	-	-
9	ronaldo-and-bonucci-give-support.jpg	60 KB	✓	-	-	-	-
10	chiellini.jpg	56 KB	-	-	-	✓	-
	TOTAL		5	4	1	1	0

C. Requirement Analysis

When a user wants to find someone's photos in a huge image collection, the problems that appear when a user wants to find someone's photos in a huge image collection will be solved by designing a face image search feature. This feature needs three steps as follows.

- The user defines the directory of image collection set as a searching target location.
- The user runs the training process that aims to identify all image files in the directory, generate the embedding value of each image, and save the database's values.
- The user inputs someone's face that wants to be automatically found in the image collection.

For more detail, please see the figure below.

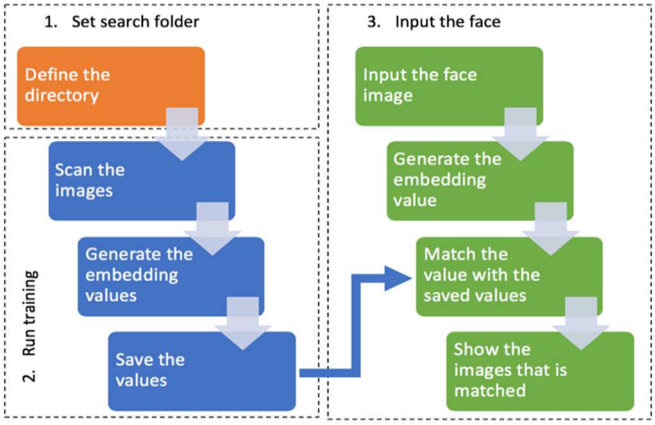

Fig. 4 Our proposed application workflow

When the user inputs someone's face photo, this feature will extract the photo using Dlib's CNN model to get its embedding value. The classification algorithm then matches this value with the embedding values that have been stored in the database. If matched, the photos are what we are looking for. The photos that we found will be displayed in the form of a list view to ease the user in seeing in detail of file's metadata, such as file name, size, created date, and file location. The open button is also provided for opening the image in the default image viewer application.

Generally, the image collection is saved in external storage such as memory card, flash disk, and external hard disk. Therefore, the application should run on a desktop computer that can read data from external sources properly. Because it involves image processing methods that need high computation resources, it requires a computer with a satisfying specification. Therefore, the requirements of the application are as follows.

- Functional needs
  a. Search directory settings menu
  b. Run the training process menu.
  c. The face photo input menu for a specific photo the user is looking for.
  d. The search results page in the form of a detailed list
  e. Menu to open files from search results
- Operational needs
  a. Open-source software
  b. Runs well on *Windows*, *Mac*, or *Linux*
  c. Minimum RAM of 4GB
  d. *Intel i5 processor* or equivalent

D. Application Design

Based on the results of the needs analysis, the application user interface design is made as follows.

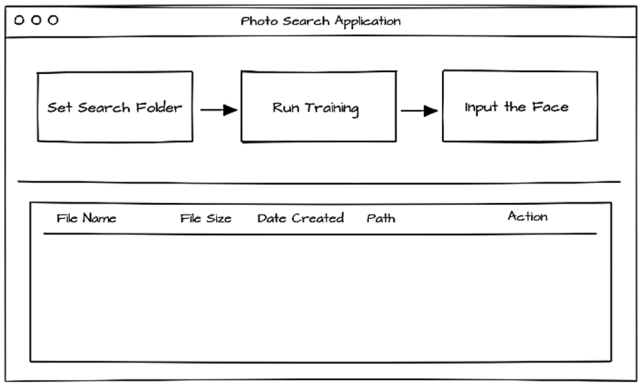

Fig. 5 Application user interface design

Face photo search application consists of 3 buttons, namely:

1) 'Set Search Folder' to choose which directory to target as a photo search location.

2) Run Training' button to run the training process and generally takes a longer processing time, depending on the number of image files in the search location folder and subfolder.

3) 'Input the Face' button to enter the sample face image that the user wants to search. Next, after entering a face image, the application will automatically search for and display search results in the list view.

E. *Database Design*

To simplify data processing, this research utilized a Database Management System (DBMS) called *SQLite* that has been integrated with the Python platform. A database is needed to store information on the training process results in the form of the embedding value of each image file in the target directory. Thus, the database schema design used by the application is described as follows.

When the user runs the 'Training' process, the application will list all files with image type in the target directory. The obtained data, such as file name, size, file location, and file creation date, are stored in the *image* table. The *image_id* attribute will be created uniquely and automatically generated by the DBMS with the *auto_increment* facility provided. In each image, a face detection process will be carried out to find the location of the facial coordinates in an image. The location of this coordinate is stored with the *face_location* attribute in the *face* table. Then proceed with the face encoding process, which aims to produce an embedding value from each face. This value is stored in the *embedding_val* attribute in the *face* table. When completing the embedding process, the timestamp is stored in the *created_at* attribute, and the *image_id* attribute is written with the *id* of the image file being processed. Similar to the *image_id* attribute in the *image* table, the *face_id* attribute will also be made uniquely and automatically generated by the DBMS with the *auto_increment* facility provided. Table relationships are made one-to-many because one picture can contain more than one person's face.

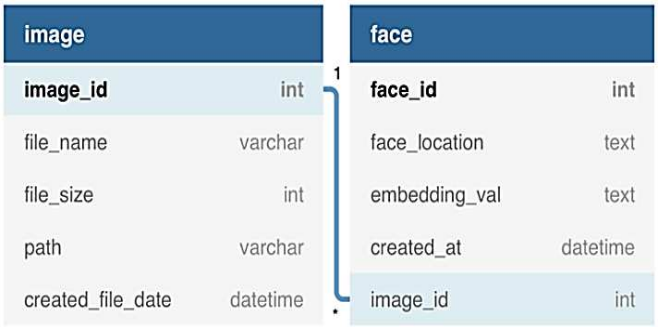

Fig. 6 Database schema

III. Result and Discussion

Based on the design formulated in the previous section, the final result of the face photo search application is explained as follows.

A. *Application Development Results*

When the face photo search application is launched, the main page displays 3 main buttons, namely *'Set Search Folder'*, *'Run Training'*, and *'Input the Face'*. At the bottom of the page, the search result's table is still empty because the search has not been performed.

When the 'Set Search Folder' button is clicked, a dialog page will appear to select which directory or folder to target the face photo search location. After selecting the folder, it will return to the main application page. Next, select the 'Run Training' button to run the training process for several minutes, or even hours, depending on how many image files are found in the target location directory. After completing the training process, select the 'Input the Face' button to enter the face image file to search.

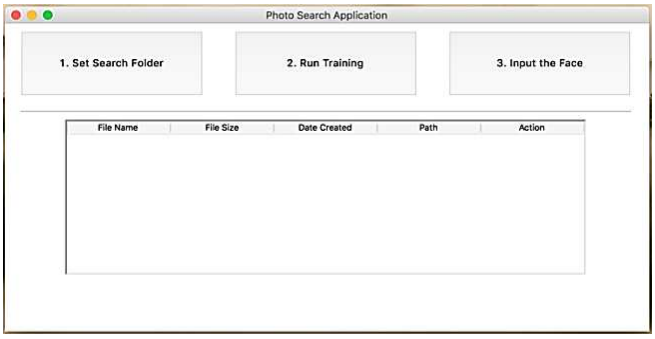

Fig. 7 The main interface of the application

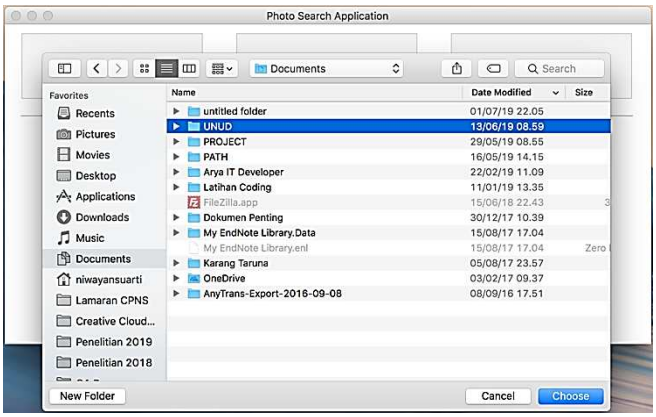

Fig. 8 The search target directory selection dialogue page

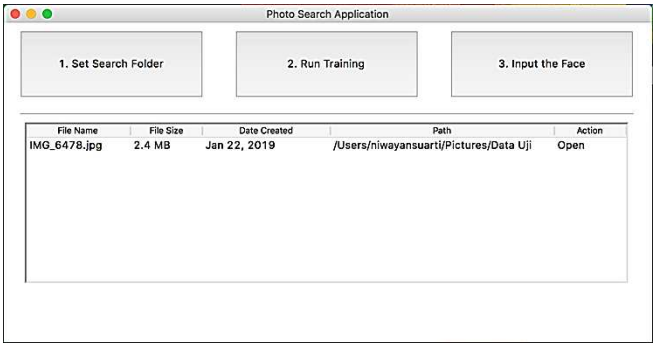

Fig. 9. Search result page

When the user enters a face image to be searched through the 'Input the Face' button, the application will search for that specific face in the target directory and display its search results in the search result table.

B. *Precision* and Recall Testing

To find out how well the performance of the face photo search application, several performance testing were performed, namely precision and recall testing. Testing precision and completeness (recall) were done by marking all photos containing certain faces (such as Mr. X's face) in a directory and then running the application to find photos of those faces. Precision rate (*P rate*) is the ratio between the number of someone's face correctly recognized and the number of someone's is recognized, while recall rate (*R rate*) is the ratio between the number of someone's face correctly recognized and the number of someone's face occurs [28].

This testing tested the image collection of football players, which were downloaded from a sports news website (Table I and Table II). The test results are displayed as follows.

TABLE III
SEARCHING RESULTS

No	Found faces					
	R	D	P	C	B	
1	-	✓*(true)*	-	-	-	
2	-	-	-	-	-	
3	✓*(true)*	✓*(true)*	-	-	-	
4	-	-	✓*(true)*	-	-	
5	-	-	-	✓*(false)*	-	
6	✓*(true)*	-	-	-	-	
7	-	-	-	-	-	
8	-	✓*(true)*	-	-	-	
9	✓*(true)*	-	-	-	-	
10	-	-	-	✓*(true)*	-	
Total	3	3	1	2	0	Average
P Rate	3/3 (100%)	3/3 (100%)	1/1 (100%)	½ (50%)	1 (100%)	90%
R Rate	3/5 (60%)	¾ (75%)	1/1 (100%)	1/1 (100%)	1 (100%)	87%

According to the testing results examined to 5 people's faces, the average precision rate is 90%, and the recall rate is 87%. This means that the proposed application with the *Dlib* library can perform well. Several images that were failed to find by the application is *pjanic-and-ronaldo.jpg*, *ronaldo-and-bonucci-hold-mandzukic.jpg*, and *dybala-try-to-hold-the-ball.jpg,* which contain difficult face poses to detect by *Dlib* library.

Fig. 10 Photo pjanic-and-ronaldo.jpg

Figure 10 and 11 show that Ronaldo's faces were failed to be detected by the application, but other persons such as Pjanic was successfully recognized. This condition has happened because Ronaldo's faces are not facing the camera so that the Dlib model had difficulty recognizing half the face. The face recognition system has the best accuracy when dealing with frontal faces (a face facing the camera). However, this is currently being explored to create a precision face recognition model that can recognize any face in various poses [29], [30].

Fig. 11 Ronaldo's faces that fail to be detected by the application.

IV. CONCLUSION

Based on the demonstrated experiments, we created a face searching application that runs on a desktop computer. It has an acceptable precision and recall rate considering the limitation of the face recognition model. For future studies, based on the problems we analyzed above, other researchers should develop face recognition models with new approaches that can deal with any face with different scales, poses, occlusion, expression, make-up, and illumination.

ACKNOWLEDGMENT

This research was funded by the Institute for Research and Community Services (LPPM), Udayana University, through Study Program Excellent Research (PUPS).

REFERENCES

[1] Z. Wei *et al.*, "AutoPrivacy: Automatic privacy protection and tagging suggestion for mobile social photo," *Comput. Secur.*, vol. 76, pp. 341–353, 2018, doi: https://doi.org/10.1016/j.cose.2017.12.002.

[2] Y. Lei, Y. Chen, L. Iida, B. Chen, H. Su, and W. H. Hsu, "Photo Search by Face Positions and Facial Attributes on Touch Devices," in *Proceedings of the 19th ACM international conference on Multimedia*, 2011, pp. 651–654.

[3] D. Wang, C. Otto, and A. K. Jain, "Face Search at Scale," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1122–1136, 2017, doi: 10.1109/TPAMI.2016.2582166.

[4] H. Kang and B. Shneideman, "Visualization Methods for Personal Photo Collections : Browsing and Searching in the PhotoFinder," in *IEEE International Conference on Multimedia and Expo (ICME)*, 2000, vol. 03, pp. 1539–1542.

[5] M. Wang and W. Deng, "Deep face recognition: A survey," *Neurocomputing*, vol. 429, pp. 215–244, 2021, doi: https://doi.org/10.1016/j.neucom.2020.10.081.

[6] G. Guo and N. Zhang, "A survey on deep learning based face recognition," *Comput. Vis. Image Underst.*, vol. 189, p. 102805, 2019, doi: 10.1016/j.cviu.2019.102805.

[7] I. Masi, Y. Wu, T. Hassner, and P. Natarajan, "Deep Face Recognition: A Survey," *Proc. - 31st Conf. Graph. Patterns Images, SIBGRAPI 2018*, pp. 471–478, 2018, doi: 10.1109/SIBGRAPI.2018.00067.

[8] R. Jafri and H. R. Arabnia, "A survey of face recognition techniques," *J. Inf. Process. Syst.*, vol. 5, no. 2, pp. 41–68, 2009.

[9] M. Wang and W. Deng, "Deep Face Recognition: A Survey," *CoRR*, vol. abs/1804.0, 2018, doi: 10.1109/SIBGRAPI.2018.00067.

[10] M. Lal, K. Kumar, R. H. Arain, A. Maitlo, S. A. Ruk, and H. Shaikh, "Study of Face Recognition Techniques: A survey," *Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 6, pp. 42–49, 2018, doi: 10.14569/IJACSA.2018.090606.

[11] Q. Wang and G. Guo, "Benchmarking deep learning techniques for face recognition," *J. Vis. Commun. Image Represent.*, vol. 65, p. 102663, 2019, doi: 10.1016/j.jvcir.2019.102663.

[12] Y. Taigman, M. A. Ranzato, T. Aviv, and M. Park, "DeepFace : Closing the Gap to Human-Level Performance in Face Verification," 2014.

[13] S. Z. Li and A. K. Jail, *Handbook of Face Recognition*. London: Springer, 2011.
[14] M. P. Beham and S. M. M. Roomi, "A review of face recognition methods," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 27, no. 4, pp. 13560051–135600535, 2013, doi: 10.1142/S0218001413560053.
[15] S. Haykin, *Neural Networks and Learning Machines Third Edition*, vol. 3. New Jersey: Pearson Education, 2009.
[16] A. Elmahmudi and H. Ugail, "Deep face recognition using imperfect facial data," *Futur. Gener. Comput. Syst.*, vol. 99, pp. 213–225, 2019, doi: 10.1016/j.future.2019.04.025.
[17] H. Habibi, A. Elnaz, J. Heravi, P. Application, and T. Detection, *Guide to Convolutional Neural Networks*. Springer, 2017.
[18] Y. Cai, Y. Lei, M. Yang, Z. You, and S. Shan, "A fast and robust 3D face recognition approach based on deeply learned face representation," *Neurocomputing*, vol. 363, pp. 375–397, 2019, doi: 10.1016/j.neucom.2019.07.047.
[19] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," 2005.
[20] L. Boussaad and A. Boucetta, "Deep-learning based descriptors in application to aging problem in face recognition," *J. King Saud Univ. Comput. Inf. Sci.*, 2020, doi: 10.1016/j.jksuci.2020.10.002.
[21] A. Geitgey, "Machine Learning is Fun! Part 4: Modern Face Recognition with Deep Learning," *Medium.com*, 2016.
[22] D. S. Trigueros, L. Meng, and M. Hartnett, "Face Recognition : From Traditional to Deep Learning Methods," *arXiv e-prints*, p. arXiv:1811.00116, 2018.
[23] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet : A Unified Embedding for Face Recognition and Clustering," 2015.
[24] L. Shi, X. Wang, and Y. Shen, "Research on 3D face recognition method based on LBP and SVM," *Optik (Stuttg).*, vol. 220, p. 165157, 2020, doi: https://doi.org/10.1016/j.ijleo.2020.165157.
[25] D. E. King, "Dlib-ml : A Machine Learning Toolkit," *J. Mach. Learn. Res.*, vol. 10, pp. 1755–1758, 2009.
[26] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *CoRR*, vol. abs/1512.0, 2015.
[27] D. E. King, "dlib-models," *Github*, 2018.
[28] M. Junker, R. Hoch, and A. Dengel, "On the Evaluation of Document Analysis Components by Recall, Precision, and Accuracy," *Proc. Fifth Int. Conf. Doc. Anal. Recognition. ICDAR '99*, pp. 713–716, 1999.
[29] M. Taskiran, N. Kahraman, and C. E. Erdem, "Face recognition: Past, present and future (a review)," *Digit. Signal Process. A Rev. J.*, vol. 106, p. 102809, 2020, doi: 10.1016/j.dsp.2020.102809.
[30] L. Zhou, W. Li, Y. Du, B. Lei, and S. Liang, "Adaptive illumination-invariant face recognition via local nonlinear multi-layer contrast feature," *J. Vis. Commun. Image Represent.*, vol. 64, p. 102641, 2019, doi: 10.1016/j.jvcir.2019.102641.