Digital Image Processing (DIP) and Generative Adversarial Networks (GANs) Techniques for Improvement Low-Resolution Face Recognition

Dian Ade Kurnia[1*], Othman Mohd[2], Mohd Faizal Abdollah[2], Dadang Sudrajat[1], Dwi Marisa Efendi[3], Sidik Rahmatullah[3]

[1] Department of Informatics Management, STMIK IKMI Cirebon, Cirebon 45142, Indonesia
[2] Faculty of Information Communication and Technology, Universiti Teknikal Malaysia, Melaka 76100, Malaysia
[3] Information System, DCC Language and Business Technology Institute, Lampung 35111, Indonesia

Corresponding Author Email: dianade2014@gmail.com

ABSTRACT

This research addresses the challenge of improving the accuracy of face recognition in low-resolution images using Digital Image Processing (DIP) and Generative Adversarial Networks (GANs). Recent advances in facial recognition have achieved high accuracy, although predominantly for high-resolution images. Low-resolution images, common in surveillance and mobile devices, pose significant accuracy challenges. The proposed DIP+GAN method integrates image preprocessing techniques such as cropping, resizing, normalization, and filtering with GANs to enhance low-resolution images. The study leverages the Georgia Tech Face Database for experiments and employs various DIP techniques and GAN architecture. The results demonstrate improved facial recognition accuracy in low-resolution images and contribute significantly to the fields of digital image processing and artificial intelligence. This research highlights the importance of preprocessing in face recognition and the effectiveness of GANs in dealing with low-resolution images.

1. INTRODUCTION

In recent years, a variety of research initiatives have been undertaken to address the problem of improving the accuracy of facial recognition methods. Algorithmic approaches have been successfully used to achieve a significant level of optimization, resulting in an accuracy rate of over 99% [1] on the Labeled Faces in the Wild (LFW) dataset [2]. The application of deep neural network techniques has significantly contributed to increasing the effectiveness of models used in facial recognition tasks [3]. However, there are still difficulties in accurately recognizing people under various contextual differences such as posture, age, lighting conditions and facial expressions. This requires the implementation of innovative modeling approaches to effectively address these limitations [4]. The highest performance in terms of facial recognition accuracy is generally achieved with high resolution images. Empirically, high-resolution images facilitate the identification process. However, the situation changes when dealing with low resolution facial images, which tend to result in lower accuracy [5]. Low resolution in facial recognition brings its own challenges [6, 7]. There are several options when dealing with low-resolution images. This is mainly due to factors such as limited viewing angles and distances to the subject, which result in lower image quality [8].

The basic concept underlying this research is the strategy to address the challenge of low resolution in the visual representation of a face, to enable computational capabilities to recognize people based on the features contained in such images [9]. Various approaches have been tried in the context of deep learning, including high-resolution development methods, the application of convolutional methods through Convolutional Neural Networks (CNN) [10, 11], and even the newest and currently popular approach, Generative Adversarial Networks (GANs), which can generate synthetic images that resemble the originals [12].

A study has found that the main focus of face recognition is still on the feature extraction and classification phases, while the image pre-processing phase takes a back seat. However, this phase plays an important role in improving face recognition accuracy, especially in the context of low-resolution facial images [13, 14].

One of the main limitations of today's technology, particularly the problem of low-resolution face recognition, is that face recognition algorithms tend to have difficulty identifying faces that are blurry or have minimal detail, and face recognition techniques are often faced with different pose variants. Lighting conditions sometimes pose a significant challenge for face recognition, so faces that are not visible due to lack of light can affect the accuracy of face recognition. One way to solve this problem is to combine image processing techniques with deep learning techniques using GANs.

A study on GANs was applied to design a custom model that can correct blurry QR code images to make them clearer. GANs help generators generate more realistic and higher-

quality QR code images through continuous training, thereby improving the quality of QR code image recognition [15]. Other research on GANs has also explored the use of a dual parallel convolution module for image restoration and the implementation of pixel-level discriminators to identify small errors in images. Its main contribution is the ability to better capture multi-scale features in image restoration and overcome distortions of local details in images [16]. In addition, GANs are widely used in medicine, statistics, law, gaming and other object recognition fields [17-21].

Therefore, based on the background context and fundamental questions presented, we propose a research initiative that adopts a methodology called Digital Image Processing (DIP) and the application of generative adversarial networks (GANs) for low-resolution facial recognition. This initiative is abbreviated as "DIP+GAN."

The findings of this study make a significant contribution to enriching the body of knowledge, particularly in the field of digital image processing and artificial intelligence, particularly in the paradigm of deep learning techniques.

2. RELATED WORK

2.1 Super resolution (SR)

Superresolution (SR) aims to reconstruct missing high-frequency information from certain low-resolution images and restore the corresponding high-resolution images [22]. The summary of the related work shown in Table 1. Based on Table 1, previous work revealed that super-resolution techniques are used for bicubic interpolation [23]. This produces higher resolution images with minimal computational effort. Recent developments in the use of super-resolution techniques are being used to improve low-resolution face recognition in combination with Generative Adversarial Networks (GANs) techniques, with results shown in improving acupuncture in image identification of SCface datasets [24]. In 2017, the SRGAN technique was introduced, which uses a deep residual network (ResNet) with skip connections to deviate from the Mean Squared Error (MSE) approach [25]. SRGAN significantly improves image quality.

The following year, Wang et al. introduced an extension for SRGAN. They introduced Enhanced Super-Resolution Generative Adversarial Networks (ESRGAN), which improve image quality by integrating three crucial elements [26].

2.2 Face recognition (FR)

Facial recognition (FR) is widely studied by researchers. Facial recognition studies have been applied to the areas of information security, access management, biometrics, law enforcement, personal security, and synthetic photos [27-29].

Recent developments in the Convolutional Neural Network (CNN) method have solved the RF problem in the LFW dataset with an accuracy rate of over 99%.

2.3 Low-resolution face recognition (LRFR)

The LRFR study was presented in 2018 on the topic of low-resolution facial recognition using Deep Coupled Resnet (DCR) and incorporates the CNN technique by producing 96.6% accuracy for an image size of 16×16 pixels [6].

Another researcher proposed two architectures in 2019 that combine deep convolutional neural networks with CNN super-resolution [30], where this architecture is composed of 14 layers for transforming high-resolution images, and the other branch is for low-resolution face imaging The transmission of images into the common space includes a 5-layer super-resolution network connected to a 14-layer network.

Table 1. The summary of related works

Author	Problem	Methods and Evaluation Result
[11]	This study focuses on low-resolution face recognition and proposes a new model.	The study's methods use a combination of GPEN Super Resolution and FaceNet for low-resolution facial recognition. The proposed method achieved a training accuracy value of 82.8%, a validation accuracy value of 66.6%, and a testing accuracy value of 69% with data augmentation
[21]	The performance degradation of facial recognition systems.	Evaluates the performance of FaceNet on low-resolution face images compared to high-resolution face images. The analysis was carried out using the Labeled Faces in the Wild (LFW) data set. The performance evaluation results of FaceNet on the LFW dataset show an accuracy of 95.12%.
[22]	The challenges of face recognition	The method uses super-resolution techniques to improve the quality of low-resolution images and improve the recognition performance of a high-resolution facial recognition system.
[23]	The problem addressed in this study is single-image super-resolution (SISR), which aims to recover high-resolution (HR) images from low-resolution (LR) images)	The method proposed in this study is a framework called SRGAN, which consists of a generator network and a discriminator network. The results of this study show that SRGAN far outperforms all reference methods and sets a new state-of-the-art for photorealistic image super-resolution. The evaluation was carried out using both quantitative measures such as PSNR and SSIM.
[25]	The challenges and recent developments in automated facial recognition that impact forensic facial recognition	The studies highlight various techniques and challenges in matching probe images to gallery images, including the use of facial landmarks, facial aging, and near-infrared imaging. The studies also provide clues to future research directions on facial recognition in forensics.
[28]	The problem of detecting low-resolution probe face images	The approach leverages Deep Convolutional Neural Networks (DCNNs) and demonstrates superior detection accuracy compared to existing state-of-the-art methods, especially on very low-resolution probe images. In addition, the authors evaluate the effectiveness of this method on extremely low-resolution probe images.
[29]	Several research gaps and improvement areas for GANs. One of the gaps is the limited work on applying GANs in other areas such as audio, music, etc	Offer insights into the current status of GANs research, their applications as well as research gaps and potential for improvement. It also highlights the importance of model variation and how GANs can generate synthetic realistic data through unsupervised learning

3. METHODS

3.1 Datasets

This study uses unstructured datasets in the form of images, particularly facial images. Datasets from public data, namely the Georgia Tech Face Database with the URL address http://www.anefian.com/research/face_reco.htm.The capacity is 128 MB and contains images of 50 subjects taken in two or three sessions between 06/01/99 and 11/15/99 at the Center for Signal and Image Processing at the Georgia Institute of Technology.

All individuals in the database are represented by 15 color JPEG images with cluttered backgrounds captured at a resolution of 640×480 pixels. Up to 50 people could be seen in the pictures taken. Each one has 15 different face poses. The number of image data is therefore 750 images. This means that the population of this data is 750 data images.

3.2 Proposed methods

The techniques proposed in this study can be performed in the phases of facial image input, image preprocessing, feature extraction, GANs techniques and result classification, as shown in Figure 1.

Based on Figure 1, some of the main reasons for proposing the technique are to consider the adaptability of the lr, the use of lost features, and the training effectiveness of the GANs themselves. The ability of the RMSprop optimization algorithm to adaptively modify the lr for every parameter to facilitate faster convergence led to its selection. The difference between the actual class label and the model's predictions is measured using the binary cross-entropy loss function. The training algorithm used allows the generator and discriminator to be updated simultaneously via their respective loss gradients. This technique allows the generator to produce images that are harder for discriminators to distinguish, which helps the two models correct each other during training.

3.3 Image processing techniques

Preprocessing data images using cropping, resizing, normalization, and filtering techniques. First, cropping is a technique for reducing the size of an image by cropping the image at predetermined coordinates in an image area. The cropping process creates a cropping object from an image or part of an image of a specific size. Second, resizing is a process of changing the size of an object larger or smaller. For images, however, resizing means increasing or decreasing the height or width of an image. There are two resizing methods, namely scaling and cropping. To resize an image, the scaling method uses the interpolation function.

The third, normalization technique in this experiment uses the Histogram Equalization technique. The histogram represents the statistical likelihood of the distribution of each gray level within a digital image. The basic concept of histogram equalization is to menstruate the histogram, resulting in a larger pixel difference, or in other words the image information becomes stronger so that the eye can capture the image information conveyed. The formula that can be used to calculate the histogram equalization is:

$$K_o = round\left(\frac{C_i.(2^k-1)}{w.h}\right) \qquad (1)$$

where, Ci is the cumulative distribution of the grayscale value to -i of the original image, round is the rounding function to the nearest number, Ko is the greyness value of the equalization histogram, w is the width of the image and $\boldsymbol{h}$ is the height of the image. Fourth, filtering is a fundamental technique used in image processing to enhance or modify images by removing noise or unwanted features from the image.
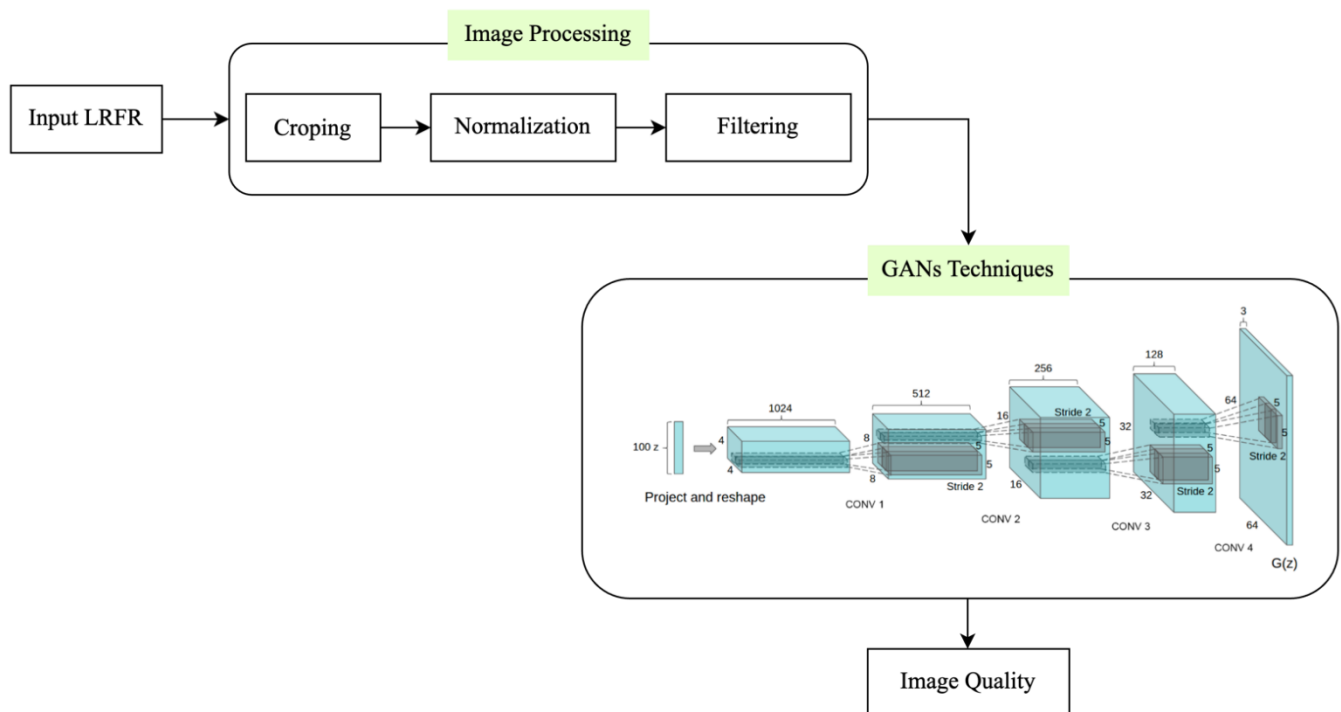

Figure 1. The Techniques of proposed methods

The filtering techniques in this research use medium methods. The process involves applying a mathematical operation to each pixel of the image to produce a new image with improved visual quality or better suitability for further processing.

In addition to the above techniques, there are various more advanced image enhancement techniques that can improve image quality and prepare training data for deep learning models. These techniques include Super-Resolution GANs (SR-GANs), noise reduction, histogram equalization, Contrast Limited Adaptive Histogram Equalization (CLAHE), Canny, Smoothing, Sharpening, and so on.

3.4 Working of Generative Adversarial Networks (GANs)

The first step is to first sample batches of random vectors from a Gaussian distribution and then generate synthetic images using a generator model. Since the generator is not yet trained at this point, the generated images have no similarity to the actual input data distribution. Stacks of real images from the input data distribution are then fed into a discriminator model along with the synthetic images produced by the generator. This is intended to train the discriminator to distinguish between authentic and synthetic images.

The batch of images generated by the generator then goes through the discriminator again after the discriminator training is complete. In this case, no authentic images are included in the input. The discriminator provides probability values as output. These output probabilities are then compared to the expected probability of one for the generator's output. An error is then calculated and propagated back via the generator. This error is instrumental in updating the generator model weights. This iterative process mentioned above continues until the synthetic images produced by the generator are very similar to those obtained from the actual input data distribution. Shown in Figure 2.
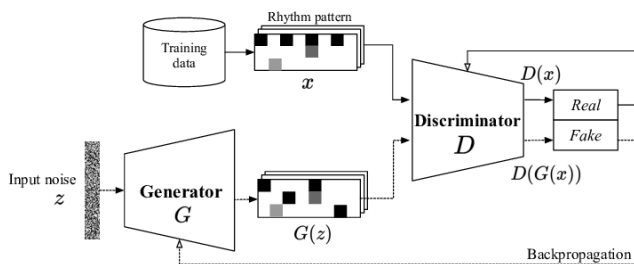

Figure 2. The Working of Generative Adversarial Networks (GANs) [31]

4. EXPERIMENT, RESULT AND DISCUSSION

4.1 Experiment of Image Processing with Cropping

The image cropping formula in image processing is a relatively simple process that removes or crops certain parts of an image to create a new, smaller image. This technique is often used to remove unwanted parts of an image and focus on a specific object or area. The mathematical formula for image cropping can be expressed as follows:

$$cropped\_image = original\_image[y1:y2, x1:x2] \qquad (2)$$

In this formula, "original_image" represents the image to be

cropped, while "cropped_image" represents the resulting image that has undergone the cropping operation. The coordinates "x1" and "y1" indicate the position of the upper left corner of the area to be cropped, while the coordinates "x2" and "y2" indicate the position of the lower right corner of the area to be cropped.

Table 2. The algorithm of cropping in image processing

Algorithm of Cropping in Image Processing
1. Load the image.
2. Define the top-left and bottom-right corners of the crop region.
3. Determine the pixel coordinates of the corners.
4. Determine the size of the crop region, either in pixels or as a percentage of the original image size.
5. Calculate the dimensions of the cropped image.
6. Create a new image with the cropped dimensions.
7. Loop through the rows and columns of the cropped image.
8. Calculate the corresponding pixel coordinates in the original image.
9. Copy the pixel values from the original image to the cropped image.
10. Display the cropped image.

Based on Table 2, it is implemented into the python programming language so that the results can be obtained as shown in Figure 3. The image is read from the drive and contains the following information: image size 640 px × 480 px, Joint Photographic Experts Group (JPEG) file format, and RGB image mode. Next, determine the coordinate points of the parts to be trimmed at the top left and bottom right. The result of the cropping can be seen in Figure 4. The information obtained from the image is an image of size 241×281 pixels, an image without format and an image in RGB mode.
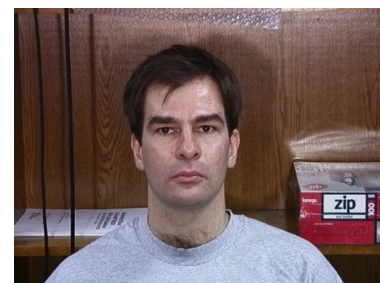

Figure 3. Original image (size 640 px × 480 px)

Figure 4. Image cropping result (size 241 px × 281 px)

4.2 Experiment of image processing with resizing

The resizing algorithm, a well-known method in image processing, allows changing the dimensions of an image while maintaining its aspect ratio. This algorithm is often used to

either reduce or increase the size of an image in Table 3.

Based on pseudocode in Table 3, it is translatable to the Python programming language by reducing 10% of the original face image, which originally had dimensions of 241×181 pixels in RGB mode. Figure 4 into dimensions measuring dimensions of 28 pixels × 21 pixels with RGB models Figure 5. Therefore, referring to the previous literature, it is stated that the image of the face is of low resolution as it has dimensions below 32 pixels.

Table 3. The image processing resizing algorithm using downscaling methods

Algorithm of Resizing in Image Processing Using Downscale Methods

1. Define a function called downscale_image that takes in the following arguments:
image: the original image
scale_factor: the factor by which the image needs to be downscaled
2. Get the dimensions of the original image and calculate the new dimensions based on the scale_factor:
new_width = original_width / scale_factor
new_height = original_height / scale_factor
3. Create a new image with the new dimensions:
Create an empty image with dimensions (new_width, new_height)
4. Loop over each pixel in the new image:
Calculate the corresponding position in the original image based on the scale_factor:
orig_x = pixel_x * scale_factor
orig_y = pixel_y * scale_factor
Get the color of the corresponding pixel in the original image:
color = image[orig_x, orig_y]
Set the color of the pixel in the new image to the color obtained in the previous step:
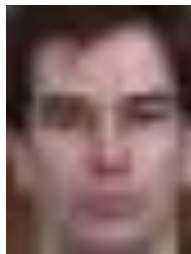new_image[pixel_x, pixel_y] = color
5. Return the new image.

Figure 5. Resizing image downscale methods (28 px×21 px)

4.3 Experiment of image processing with normalization

The next stage of image processing is normalization. Normalization in image processing involves adjusting the intensity values of an image to a desired scale or range. This process is often used to improve the contrast, brightness and visibility of an image. The normalization process involves rescaling pixel values so that they fall within a certain range, typically between 0 and 255 for 8-bit images. This can be done using various techniques including linear scaling, histogram equalization, and contrast stretching. Histogram equalization is a technique that redistributes the pixel values in an image to improve its contrast. The intensity histogram of the image is transformed to have a flat distribution. This makes the dark pixels darker and the light pixels brighter, resulting in an image with improved contrast and visibility, as shown in Table 4:

Table 4. The algorithm of normalization in image processing using histogram equalization

Algorithm of Normalization in Image Processing Using Histogram Equalization

1. Input an image
2. Calculate the histogram of the image (i.e., the frequency distribution of the intensity values of the pixels in the image)
3. Calculate the cumulative distribution function (CDF) of the histogram
4. Normalize the CDF so that it has the same range as the intensity values in the image
5. Apply the normalized CDF to each pixel in the image to get the new intensity value for that pixel
6. Output the new image with the equalized histogram

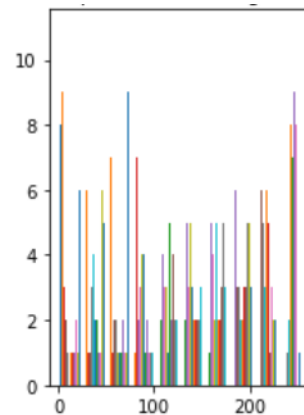**Figure 6.** Result of normalization image

Figure 7. Result of histogram normalization image

Based on Figure 6 and Figure 7 the results of applying the histogram equalization technique are shown, which aims to increase the contrast of the image by flattening the pixel intensity distribution. Visually, the increase in contrast is particularly noticeable in areas of low intensity (low contrast) that were previously less noticeable. This technique has made it possible to highlight facial features such as the contours of the nose, eyes and mouth more clearly. However, on low-resolution images, this normalization process also introduces side effects in the form of noise and visual artifacts that make facial texture appear rougher. The resulting pixel intensity distribution results in greater sharpness, but can eliminate fine details that are essential for natural image quality.

The Histogram in Figure 7 shows a more even distribution of pixel intensity, with intensity values now ranging from 0 to 255. This indicates that the overall contrast of the image has been improved and the available dynamic range has been maximized. However, sharp peaks in the histogram indicate a significant increase in intensity in certain areas, which may result in excessive amplification.

4.4 Experiment of Generative Adversarial Networks (GANs)

4.4.1 Import the libraries

Some libraries imported and required in this experiment are Tensorflow, Keras, Numpy, Matplotlib, OpenCV, TQDM and other libraries deemed necessary. The coding begins with the integration of TensorFlow, a comprehensive and flexible deep learning framework, along with Keras, a resource that provides a more intuitive platform for building and training neural networks. Matplotlib was used to create educational visualizations, OpenCV was utilized for sophisticated image processing, and the NumPy library was also imported for its effective numerical processing capabilities. To further enhance the flexibility of data processing, standard Python modules like "os" for operating system interaction and "re" for regular expressions were added. Additions like "tqdm" for interactive progress bars and Keras utilities like "img_to_array" for image conversion and "plot_model" for model architecture visualization make it easier to explore and evaluate machine learning models.

4.4.2 Load the datasets

The algorithm that is used to process a group of images stored in a directory is listed in Table 5.

Table 5. The algorithm of load the datasets

Algorithm Load the Datasets
Input: directory_path (default = '../input/low-resolution')
Output: Sorted list of files based on alphanumeric order
1.   DEFINE convert_text AS a lambda function:
1.1.   IF text IS numeric:
RETURN the integer value of text
1.2.   ELSE:
RETURN the lowercase version of text
2.   DEFINE key_extractor AS a lambda function:
2.1. SPLIT each key (string) into segments, separating numbers from text
2.2. CONVERT each segment using the convert_text function
2.3. RETURN the list of converted segments as the sorting key
3.   LIST image_files = list of files in directory_path
4.   SORT image_files using key_extractor AS sorting key
5.   FOR each image_name IN image_files:
5.1.   DISPLAY progress using tqdm
5.2.   IF image_name IS 's45_15.jpg':
EXIT the loop
5.3.   ELSE:
CONTINUE processing (specific action not defined)
6.   RETURN image_files

Based on Table 5, the sorted_alphanumeric function is defined to sort data alphanumeric. This function uses a lambda conversion expression that converts text to an integer if it is a digit, or to lowercase if it is not. This ensures that the sorting is done taking into account the numeric value of the numbers contained in the string. Directories containing low-resolution images are specified as paths. A list of filenames is retrieved from the directory using os.listdir and then sorted using the defined sorted_alphanumeric function. The iteration is performed on the sorted file list using tqdm, which is normally used to display a progress bar in the loop. If the file name in the loop matches "s45_15.jpg", the loop stops. This indicates that the process only goes up to that specific file, perhaps as part of a validation or test. If the file name does not match, the

image is read using cv2.imread, which is then converted from BGR to RGB color space, since OpenCV reads images in BGR format by default. The image is then resized to the size specified by the SIZE constant. The image is normalized by subtracting each pixel with a value of 127.5 and then dividing by the same value. This normalization step changes the range of pixel values from [0, 255] to [-1, 1], which is a common practice in image processing before feeding into a neural network. The image is then converted to a float data type. It is then appended to a list or array, preparing it for further processing, perhaps as part of a data batch used by the machine learning model.

The result of load the datasets in the Figure 8:

Figure 8. The result of load the datasets

4.4.3 Visualization current images

This algorithm defines a function plot_images for visualizing images. This function accepts a parameter sqr that specifies the number of images to display in a square grid, as shown Table 6.

Table 6. The algorithm of visualization current images

Algorithm of Visualization Current Images
Input: grid_size (default = 5)
Output: Grid of sample images displayed
1.   Setting figure size to (15, 15)
2.   Setting title of the figure to "Sample of Real Images" with font size 35
3.   For each index idx from 0 to (grid_size * grid_size - 1) do:
3.1.   Setting up a subplot at position (grid_size, grid_size, idx + 1)
3.2.   Show the image at index idx scaled by 0.5 and shifted by 0.5
3.3.   Remove x-axis ticks
3.4.   Remove y-axis ticks
4.   End For
5.   Call Show_Images with grid_size = 15

Based on Table 6, it is known that this function begins by specifying the size of the visualization figure using plt.figure with size (15,15), which will create a plot area with the specified size. The title of the figure is set via plt.title with the title "Real Images" and a font size of 35.

Figure 9. The sample result of visualization current images

Based on Figure 9 the function is iterated with a for loop that executes sqr by sqr, effectively creating a square grid for placing the image captions. The position of the subplot to be displayed in the grid is determined by plt. Subplot in every iteration. The ith image from the img array is then displayed by calling plt. imshow. This is done by adding 0.5 to the pixel values that have already been normalized to the value range [-1.1]. [0.1] was normalized. Reach. The plt.xticks and plt.yticks functions with empty list arguments are used to remove the label ticks from the x and y axes on each subplot to display an image without labels on these axes. After the function definition, the plot_images function is called with the argument 15, which indicates that this function plots the images in the 15×15 grid, thus displaying a total of 225 images if available in the img array.

4.4.4 Define batch size

The algorithm in Table 7 illustrates the dataset creation process in TensorFlow, a popular machine learning framework.

Table 7. The algorithm of define of batch size

Algorithm Define of Batch Size
Input: _image (list of images), bsize (def= 32)
Output: Dataset divided into batches
1.   SET bsize TO 32
2.   CONVERT _image list to a NumPy array and store in variable `image_array`
3.   CREATE a TensorFlow Dataset using `image_array` with from_tensor_slices method and store in variable `dataset`
4.   DIVIDE `dataset` into batches of size bsize using the batch method
5.   RETURN `dataset`

Base on Table 7, the variable batch_size is initialized with a value of 32, which defines the number of samples per batch to be processed in one iteration during model training. The dataset was created from an array of img that was converted to a tensor using NumPy, through the function call np.array(img). NumPy is a widely used library in scientific computing that provides support for large and efficient multidimensional arrays. The tf.data.Dataset.from_tensor_slices function is called with the image tensor as an argument. This function is tasked with creating a Dataset object in TensorFlow that allows iteration through the image tensor in smaller chunks or "slices". The.batch(batch_size) method is applied to the created Dataset object. This method groups batch_size consecutive elements of the dataset into a single batch, facilitating more efficient parallel processing during model training.

4.4.5 Build generator network

This generator model is often used in the context of Generative Adversarial Networks (GANs), where it plays a role in generating new data that is similar to the original data distribution. The algorithm reflects the best practices of generative modeling in Table 8 and integrates modern techniques into the network architecture for effective and robust learning.

The output of the generator network can be characterized as a multi-layer artificial neural network designed to perform complex machine learning tasks. According to Table 8, the model consists of the following layers: convolutional layer (Cnv2D), activation function (LeakyReLU), batch normalization (BatchNormalization), dense layer (Dns) and convolutional transposition layer (Cnv2DTrans).

Table 8. Model summary of generator network architecture

Layer (Type)	Output Shape	Parameter #
g_dns (Dns)	(None, 49152)	4915200
g_rshp (Dns)	(None, 128, 128, 3)	0
g_cnv2d (Cnv2D)	(None, 128, 128, 128)	6144
g_cnv2d_1 (Cnv2D)	(None, 64, 64, 128)	262144
g_bnorm (BatchNormalization)	(None, 64, 64, 128)	512
g_lky_re_lu (LeakyReLu)	(None, 64,64,128)	0
g_cnv2d_2 (Cnv2D)	(None, 64, 64, 256)	524288
g_cnv2d_3 (Cnv2D)	(None, 32, 32, 256)	1848576
g_bnorm_1 (BatchNormalization)	(None, 32, 32, 256)	1824
g_lky_re_lu_1 (LeakyRelu)	(None, 32, 32, 256)	0
g_cnv2d_trans (Cnv2Dtrans)	(None, 32, 32, 512)	2807152
g_cnv2d_4 (Cnv2D)	(None, 16, 16, 512)	4194384
g_lky_re_lu_2 (LeakyReLu)	(None, 16, 16, 512)	0
g_cnv2d_trans_1 (Cnv2Dtrans)	(None, 16, 16, 512)	4194384
g_cnv2d_trans_2 (Cnv2Dtranspose)	(None, 32, 32, 512)	4194384
G_bnorm_2 (BatchNormalization)	(None, 32, 32, 512)	2848
g_lky_re_lu_3 (LeakyReLu)	(None, 16, 16, 512)	0
g_cnv2d_trans_3 (Cnv2Dtrans)	(None, 32, 32, 256)	2897152
g_cnv2d_trans_4 (Cnv2Dtrans)	(None, 64, 64, 256)	1848570
g_bnorm_3 (BatchNormalization)	(None, 64, 64, 256)	2848
g_cnv2d_trans_5 (Cnv2Dtrans)	(None, 128, 128, 128)	524288
g_cnv2d_trans_6 (Cnv2Dtrans)	(None, 128, 128, 128)	262144
g_bnorm_4 (BatchNormalization)	(None, 128, 128, 128)	512
g_cnv2d_trans_7 (Cnv2Dtrans)	(None, 128, 128, 3)	6147
G_Total Parameters	**25,379,843**	
G_Trainable G_Parameters	**25,377,283**	
G_Non-trainable G_Parameters	**2,568**	

Base on of Figure 8, it can be described that in the first layer there is a dense layer with a very large number of parameters, which means that a significant number of neurons are required to capture high-level features. The layer is then reshaped to ensure that the input data conforms to the dimensions required for the next convolutional layer. After applying filters to extract spatial features from the data, the convolution layer (Cnv2D) is followed by a batch normalization layer, which aims to increase training stability and convergence speed. The network can learn more complex relationships between input and output data by introducing nonlinearity into the model using the LeakyReLU activation function. The convolutional transposition layer (Cnv2DTranspose) is used to increase the dimensionality of the learned representation. It is often used in architectures such as autoencoders to decompress data to the original dimensions after the compression process.

The overall architecture suggests that this model could be designed for tasks related to image processing such as image segmentation or image generation, where the convolutional transposition layer is usually instrumental in forming an output that has the same dimension as the original input.

The model consists of millions of trainable parameters, indicating that it is very deep and may require large data sets and significant computing power for training. The presence of untrainable layers indicates the use of fixed parameters during the training process, which may be related to layers such as BatchNormalization, which require parameters for normalization that are not updated during backpropagation.

This architectural structure reflects a complex and well-structured design that has the ability to learn feature representations at different levels and produce detailed outputs from the given inputs.

4.4.6 Build discriminator network

Base on Table 9, the discriminator architecture is defined in the GAN context, which is designed to identify and distinguish between the original data and the data generated by the generator. This architecture is typical for two-class classification tasks and incorporates good design principles such as batch normalization and appropriate weight initialization to support an efficient training process.

Specifically, this architecture features sequentially arranged convolution layers (Conv2D), followed by batch normalization and an intercalated nonlinear activation function (LeakyReLU). The structure is shown in Table 9.

Base on Table 9, it can be described that in the convolutional layer, it can be seen that the feature size (feature map) gradually decreases, from 64x64 to 4x4, indicating that the network compresses spatial information into a more abstract and dense representation. This reduction in spatial dimension is accompanied by an increase in the number of filters from 128 to 512. This is usually done to compensate for the reduction in spatial resolution by increasing the depth of features that can encode more complex information at a higher level. Batch normalization, applied after each convolution layer, aims to stabilize and accelerate the learning process by eliminating the internal covariate shift problem, i.e. h. the shift in the input distribution of the layers during the training phase is reduced.

LeakyReLU is a variation of the ReLU activation function that allows small gradients when units are inactive (negative input values) to overcome the dead neuron problem that can occur with traditional ReLU. After a sequence of convolution, batch normalization and LeakyReLU layers, the architecture ends with a flattening layer that converts two-dimensional features into one-dimensional vectors, preparing them for integration into a dense or fully connected layer. The final dense layer shows only one entity, meaning the architecture is designed for binary regression or classification tasks.

4.4.7 Defining plot image generated by generator before training

The generator model in Table 10 is intended to learn from the distribution of the training data and generate new data that is similar to the original data. This approach is common in GAN networks. This algorithm is specifically used to visualize the results of a single image generation instance and provides visual insights into the performance and capabilities of the generator model.

Table 9. Model summary of discriminator architecture

Layer (Type)	Output Shape	Parameter #
d_cnv2_d5 (Cnv2D)	(None, 64,64,128)	6144
d_bnorm_5 (BatchNormalization)	(None, 64,64,128)	512
d_Lky_re_lu_4 (LeakyReLu)	(None, 64,64,128)	0
D_cnv2_d6 (cnv2D)	(None, 32,32,128)	262144
d_bnorm_6 (BatchNormalization)	(None, 32,32,128)	512
d_Lky_re_lu_5 (LeakyReLu)	(None, 32,32,128)	0
d_cnv2_d7 (Cnv2D)	(None, 16,16,256)	524288
d_bnrom_7 (BatchNormalization)	(None, 16,16,256)	1024
d_Lky_re_lu_6 (LeakyReLu)	(None, 16,16,256)	0
d_cnv2_d8 (Cnv2D)	(None, 8,8,256)	0
d_bnorm_8 (BatchNormalization)	(None, 16,16,256)	1024
d_Lky_re_lu_7 (LeakyReLu)	(None, 8,8,256)	0
d_cnv2_d9 (Cnv2D)	(None, 4,4,512)	2097152
d_Lky_re_lu_8 (LeakyReLu)	None, 4,4,512)	0
d_flatten (Flatten)	(None 8192)	0
d_dense_1 (Dense)	(None, 1)	0
D_Total Parameters	3,494,569	
D- Trainable Parameters	3,948,033	
D_Non-trainable Parameters	1,536	

Table 10. Defining plot image generated

| Algorithm of Defining the Plot Image Generated |
|---|
| Input: generator_model |
| Output: Display the generated image |
| |
| 1. GENERATE random_noise: |
| 1.1 DEFINE mean AS -1 |
| 1.2 DEFINE standard_deviation AS 1 |
| 1.3 DEFINE shape AS (1, 100) |
| 1.4 CREATE random_noise FROM normal distribution with mean and standard deviation, and with the specified shape |
| |
| 2. GENERATE generated_image BY passing random_noise TO generator_model |
| |
| 3. DISPLAY the first generated image: |
| 3.1 EXTRACT the first image FROM generated_image array |
| 3.2 USE Imshow TO display the image |
| 3.3 CALL plt.show() TO display the image |
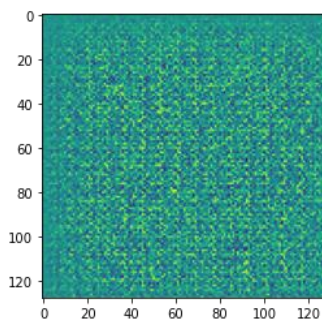
Figure 10. Plot image generated by generator

Based on Figure 10, a generative process is performed to generate images using the deep learning model. First, the noise vector is initialized using a normal distribution. This vector has a dimension of 1×100 and the values are drawn from a normal distribution with mean -1 and standard deviation 1. The noise vector is then fed as input to the generator function. Since this function is a pre-trained model, it can generate image data from the noise vector. To retrieve the first image from the generation stack, the image tensor, the output of the generator function is indexed. Then the image is visualized using the Matplotlib library (symbolized as plt) using the imshow function. This function displays the image generated by the model generator. Finally, plt.show() is called to display the generated chart to the user, as shown in Figure 10.

4.4.8 Defining loss function and optimizer

A decay rate of 1e-8, a clip value of 1.0, and an lr of 0.0001 are all set for the RMSprop optimization algorithm. Because RMSprop can adaptively change the lr for each parameter, it is a popular optimization technique in neural networks that helps accelerate convergence. The binary cross-entropy loss function is configured to work with logits, meaning it does not go through sigmoid activation first. It measures the discrepancy between the actual class label and the prediction

generated by the model. The generator loss function is calculated assuming that all outputs generated by the generator (fake_output) are positive class samples (1). The binary cross-entropy function is then used to calculate the loss by comparing these fake outputs with the label output.

The discriminator loss function is calculated with two components: fake_loss and real_loss. The fake loss is calculated by comparing the generator's fake output with a label of zero, indicating that the sample is fake. The actual loss is calculated by comparing the actual output with the label one, which indicates that the sample is real. The total loss for the discriminator is the sum of the false loss and the actual loss.

4.4.9 Defining the training model

The algorithm is a standard training iteration in GAN, where the generator and discriminator are simultaneously updated via the calculated gradients of their respective loss functions. This algorithm allows both models to correct each other in the training process, with the aim of the generator producing images that are increasingly difficult for the discriminator to distinguish. This method underlies effective GAN training and is the basis for adversarial learning.

4.4.10 Training and model performance

The algorithm in Table 11 consists of a set of two functions used to train and visualize the performance results of the Generative Adversarial Network (GAN) model.

Based on Table 11, it can be interpreted that the model achieved a final generative loss of 1.4768 and a final discriminative loss of 0.7807 in the first experiment with 50 epochs and a learning rate (lr) of 0.0001. The training time was 7:00 p.m. (time unit not specified) with a final accuracy of 70.62%.

In the second experiment, the number of epochs was increased to 60 at the same lr, which resulted in a slightly higher final generator loss (1.7887) and a lower final discriminator loss (0.6954) compared to the first experiment. The accuracy increased to 72.12%, showing improved performance despite the increased generator loss. In the third experiment, the number of epochs was reset to 50, but with a lower lr (0.00001), resulting in a lower loss of the final generator (0.6797) but a much higher loss of the final discriminator (1.6408) compared to the two previous experiments. The accuracy drops drastically to 50.92%, indicating that a reduction in the lr is unfavorable to the performance of this model.

The conclusion that can be drawn from this table is that adjusting the lr and number of epochs has a significant impact on the training results. A higher lr with more epochs (in the second experiment) seems to improve the accuracy of the model more effectively. Meanwhile, reducing the lr (in the third experiment) negatively affected the accuracy of the model, although the generator loss decreased, which may indicate that the model did not evolve enough during training to generalize well to unseen data. This highlights the importance of balancing the lr and number of epochs to achieve optimal performance in machine learning models.

Table 11. The result of model training

No. of Datasets	No. of Sample	Epoch	Parameter Optimization			Training Process			
			lr	Clip Value	Decay	Last GenerativLoass	Last Discriminative Loss	Time	Accuracy
750	674	50	0.0001	1.00	1,00E-08	1.4768	0.7807	19.00	70.62%
750	**674**	**60**	**0.0001**	**1.00**	**1,00E-08**	**1.7887**	**0.6954**	**19.00**	**72.12%**
750	674	50	0.00001	1.0	1,00E-08	0.6797	1.6408	19.00	50.92%

4.4.11 Model visualization

To evaluate and understand the dynamics between components in a Generative Adversarial Network (GAN), a series of experiments were conducted to observe the changes in metrics related to generator and discriminator losses as well as discriminator accuracy during the training phase. The following three graphs (Figures 11-13) show the measured results of the experiments conducted over different epochs. Through these visualizations, it is possible to interpret how the adjustment and adjustment of the model parameters affects the efficiency and effectiveness of the training process, as well as the balance between the generator's ability to generate convincing data and the discriminator's ability to distinguish between them, to judge original samples and samples generated by the generator. These metrics serve as a basis for improving and refining the GAN architecture and training strategy to improve the model's performance in generating samples that not only outsmart the discriminator but also maintain high quality and consistent variability.
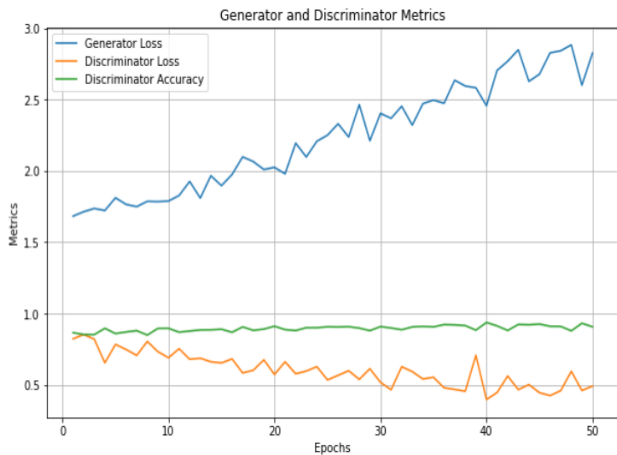

Figure 11. Comparison of Generator loss, discriminator loss, and discriminator accuracy and result of visualization model during GAN training with **50 epoch and LR 0.0001**
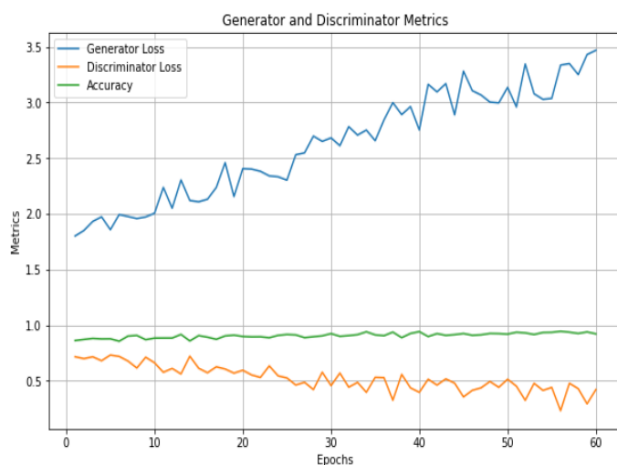

Figure 12. Comparison of generator loss, discriminator loss, and discriminator accuracy and result of visualization model During GAN training with **60 Epoch and LR 0.0001**
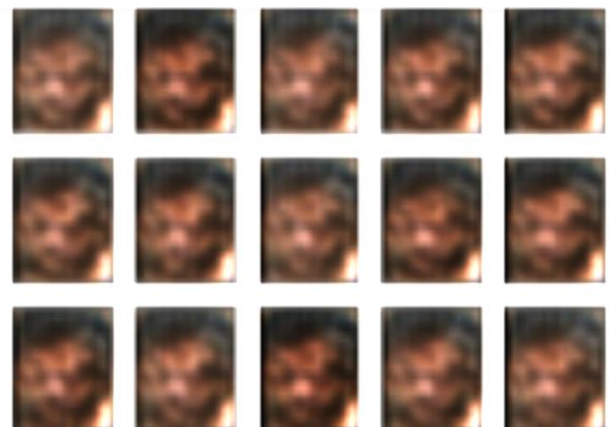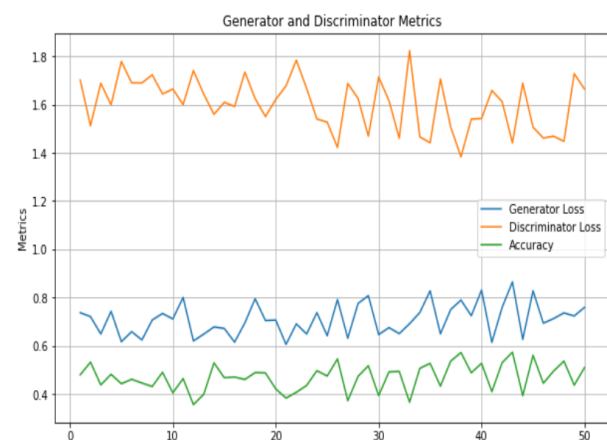

Figure 13. Comparison of generator loss, discriminator loss, and discriminator accuracy and result of visualization model during GAN training with **50 Epoch and LR 0.00001**

The Figure 11 shows the progression of the metrics for the two components within the Generative Adversarial Networks (GANs) over the course of 50 training epochs. The two components are the generator and the discriminator, each of which performs different tasks and is characterized by two metrics: loss and accuracy. On the vertical axis, the chart displays the metric values, while the horzontal axis indicates the number of epochs. The training timeline shows that the generator loss tends to increase from the beginning to the end of training, while the discriminator loss gradually decreases. Accordingly, the discriminator accuracy appears relatively stable and high throughout the training process.

An increase in generator loss indicates that over time the generator is finding it increasingly difficult to fool the discriminator with its output, which could indicate that the discriminator is getting better at distinguishing between the real data and the fake ones generated by the generator distinguish data.

Reducing the discriminator's loss means increasing its ability to identify the data generated by the generator. However, if the discriminator loss drops too much, there is a risk that the discriminator will become too powerful compared to the generator, which could hinder the co-learning process.

High and stable discriminator accuracy indicates that the discriminator is consistently good at classifying real and fake data. However, if the accuracy is too high, an overfitting phenomenon may occur, where the discriminator is overfitted to the training data set and may not perform well on data that has not yet been seen.

Figure 12 shows the dynamics of the interaction between generators and discriminators in the Generative Adversarial Network (GAN) over 60 training epochs. Metrics evaluated include generator loss, discriminator loss, and discriminator accuracy. From the graphical observations, it can be seen that the generator loss value varies, but generally increases gradually as the number of epochs increases. This indicates that it is becoming increasingly difficult for the generator to produce samples good enough to defeat the discriminator. In contrast, the discriminator loss decreases very little over time, indicating that the discriminator is becoming more and more effective in distinguishing between the original data and the data generated by the generator. This can be interpreted as an improvement in the performance of the discriminator. The accuracy of the discriminator, however, remained relatively constant and showed no significant increasing or decreasing trend. This stable and high accuracy may reflect that the discriminator can maintain its performance in classifying real and fake samples with a constant success rate.

The Figure 13, presented shows the performance of the two key components in a Generative Adversarial Network (GAN), namely the generator and the discriminator, as measured by loss and accuracy metrics, over 50 training epochs.

On the vertical axis, the chart measures the metrics associated with the performance of both components, while the horizontal axis reflects the number of training epochs. Based on the graph, the generator loss has a high variability with relatively stable values, but tends to increase slightly as the training process progresses. This high variability may indicate that the generator continues to have difficulty generating data compelling enough to defeat the discriminator.

Regarding the discriminator loss, the graph shows a gradual decrease, indicating an increase in the efficiency of the discriminator in identifying the samples generated by generator as incorrect. This is reinforced by the accuracy graph,

which shows a gradual and steady increase during training, suggesting that the discriminator is getting better and better at performing correct classification between the original data and the data generated by the generator.

The pattern emerging from this graph may indicate that the discriminator is in a favorable position compared to the generator, as the discriminator shows a consistent improvement in performance while the generator does not show a significant improvement in performance. However, in ideal practice, both components should simultaneously improve performance, indicating the presence of a healthy adversarial dynamic where the generator becomes better at producing convincing samples and the discriminator becomes better at detecting the authenticity of the samples.

4.5 Discussion

According to the study's results, face recognition accuracy in low-resolution images can be improved by up to 72-12% when combining Digital Image Processing (DIP) and Generative Adversarial Networks (GANs). Although these results are very promising, some aspects require further discussion to understand model generation and its performance on larger, more complex real-world datasets.

Model generalization shows good performance for the dataset used, but there are several factors associated with generalization, namely data variability and overfitting. The dataset used in this study may have limited variations in poses, lighting, facial expressions, and backgrounds. To evaluate the generalization capabilities of the model, tests must be performed on data sets that have higher variability and reflect more diverse real-world conditions. The model may be at risk of overfitting if it performs well on training data but poorly on test data. Testing a larger, more diverse additional data set can help find and resolve overfitting issues.

5. CONCLUSION

Based on the results and discussions conducted, as well as the stages of the research method, it can be concluded that there is an improvement in face recognition accuracy in low resolution images using Digital Image Processing (DIP) techniques and Generative Adversarial Networks (GANs). The proposed DIP+GAN method successfully shows an accuracy improvement of face detection on low resolution images of 72.12% with a number of epochs of 60 from a sample dataset of 674 face image data. It makes an important contribution to the development of digital image processing and artificial intelligence in the context of facial recognition.

The innovative aspects of this research include combining Digital Image Processing (DIP) and Generative Adversarial Networks (GANs) techniques to improve the accuracy of face recognition on low-resolution images, demonstrating that combining traditional methods and deep learning Methods successful is achieving better results. This study also highlights the importance of setting parameters such as lr and number of epochs to achieve optimal model training results.

The results of this study may have implications for future studies, so we hope that the research on GAN architecture will become more diverse and sophisticated to improve the quality of low-resolution image processing. Larger and more diverse aspects of testing datasets need to be an important focus to evaluate the effectiveness of the technique in different

conditions and environments. Integration with other image processing methods can improve accuracy under more complex conditions such as low light conditions or variations in facial pose.

The results of this study may have implications for future studies, so we hope that the research on GAN architecture will become more diverse and sophisticated to improve the quality of low-resolution image processing. Larger and more diverse aspects of testing datasets need to be an important focus to evaluate the effectiveness of the technique in different conditions and environments. Integration with other image processing methods can improve accuracy under more complex conditions such as low light conditions or variations in facial pose.

Some recommendations that may be relevant for future research, based on research on low-resolution facial recognition using DIPs and GANs, include exploring more diverse and sophisticated GAN architectures to improve the quality of low-resolution image processing. Conducting tests on larger and more diverse data sets to evaluate the effectiveness of the technique in different conditions and environments. Integrating facial recognition techniques with other image processing methods to improve accuracy under more complex conditions such as low lighting or different facial poses.

ACKNOWLEDGMENT

REFERENCES

[1] Wen, G., Chen, H., Cai, D., He, X. (2018). Improving face recognition with domain adaptation. Neurocomputing, 287: 45-51. https://doi.org/10.1016/j.neucom.2018.01.079

[2] Zhang, N., Deng, W. (2016). Labeled Faces in the Wild: A database for studying face recognition in unconstrained environments. In 2016 International Conference on Biometrics (ICB), Halmstad, Sweden, pp. 1-11. https://doi.org/10.1109/ICB.2016.7550057

[3] Robert, B., Brown, E.B. (2017). Face recognition in real-world surveillance videos with deep learning method. In 2017 2nd International Conference on Image, Vision and Computing (ICIVC), Chengdu, China, pp. 239-243. https://doi.org/10.1109/ICIVC.2017.7984553

[4] Guo, G., Zhang, N. (2019). A survey on deep learning based face recognition. Computer Vision and Image Understanding, 189: 102805. https://doi.org/10.1016/j.cviu.2019.102805

[5] Li, S., Liu, Z., Wu, D., Huo, H., Wang, H., Zhang, K. (2022). Low-resolution face recognition based on feature-mapping face hallucination. Computers and Electrical Engineering, 101: 108136. https://doi.org/10.1016/j.compeleceng.2022.108136

[6] Lu, Z., Jiang, X., Kot, A. (2018). Deep coupled ResNet for low-resolution face recognition. IEEE Signal Processing Letters, 25(4): 526-530. https://doi.org/10.1109/LSP.2018.2810121

[7] Cheng, Z., Zhu, X., Gong, S. (2020). Face re-identification challenge: Are face recognition models good enough? Pattern Recognition, 107: 107422. https://doi.org/10.1016/j.patcog.2020.107422

[8] Li, P., Prieto, L., Mery, D., Flynn, P. (2018). Face recognition in low quality images: A survey. arXiv preprint arXiv:1805.11519. http://arxiv.org/abs/1805.11519

[9] Herrmann, C., Willersinn, D., Beyerer, J. (2016). Low-resolution convolutional neural networks for video face recognition. In 2016 13th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Colorado Springs, CO, USA, pp. 221-227. https://doi.org/10.1109/AVSS.2016.7738017

[10] Shi, J., Liu, T., Chen, N., Liu, J., Dou, Y., Zhao, Y. (2021). Low resolution and multi-pose face recognition based on residual network. In IEEE Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Chongqing, China, pp. 1587-1593. https://doi.org/10.1109/IAEAC50856.2021.9390821

[11] Kurnia, D. A., Setiawan, A., Amalia, D. R., Arifin, R. W., Setiyadi, D., (2021). Image processing identifacation for indonesian cake cuisine using CNN classification technique. Journal of Physics: Conference Series, 1783: 012047. https://doi.org/10.1088/1742-6596/1783/1/012047

[12] Zebua, K.S., Kartowisastro, I.H., Kusuma, G.P. (2023). Low resolution face recognition using combination of gpen super resolution and facenet. Journal of Theoretical and Applied Information Technology, 101(12): 4991-5000.

[13] Oloyede, M.O., Hancke, G.P., Myburgh, H.C. (2020). A review on face recognition systems: Recent approaches and challenges. Multimedia Tools and Applications, 79(37-38): 27891-27922. https://doi.org/10.1007/s11042-020-09261-2

[14] Kurnia, D.A., Mohd, O., Abdollah, F., Sudrajat, D., Wijaya, Y.A. (2021). Face recognition techniques : A systematic literature review (research trends, datasets, and methods). Journal of Theoretical and Applied Information Technology, 99(21): 5217-5231.

[15] Dong, H., Liu, H., Li, M., Ren, F., Xie, F. (2024). An algorithm for the recognition of motion-blurred QR codes based on generative adversarial networks and attention mechanisms. International Journal of Computational Intelligence Systems, 17(1): 83. https://doi.org/10.1007/s44196-024-00450-7

[16] Ren, H., Sun, K., Zhao, F., Zhu, X. (2024). Dunhuang murals image restoration method based on generative adversarial network. Heritage Science, 12(1): 1-20. https://doi.org/10.1186/s40494-024-01159-8

[17] Usman Akbar, M., Larsson, M., Blystad, I., Eklund, A. (2024). Brain tumor segmentation using synthetic MR images - A comparison of GANs and diffusion models. Scientific Data, 11(1): 1–17. https://doi.org/10.1038/s41597-024-03073-x

[18] Kim, J., Lim, M.H., Kim, K., Yoon, H.J. (2024). Continual learning framework for a multicenter study with an application to electrocardiogram. BMC Medical Informatics and Decision Making, 24(1): 1-13. https://doi.org/10.1186/s12911-024-02464-9

[19] Murgas, B., Stickel, J., Ghosh, S. (2024). Generative adversarial network (GAN) enabled statistically equivalent virtual microstructures (SEVM) for modeling

cold spray formed bimodal polycrystals. npj Computational Materials, 10(1): 1-14. https://doi.org/10.1038/s41524-024-01219-4

[20] Yin, Y., Yuan, Z., Tanvir, I. M., Bao, X. (2024). Electronic medical records imputation by temporal generative adversarial network. BioData Mining, 17(1): 1-22. https://doi.org/10.1186/s13040-024-00372-2

[21] Yamada, F.M., Batagelo, H.C., Gois, J.P., Takahashi, H. (2024). Generative approaches for solving tangram puzzles. Discover Artificial Intelligence, 4(1): 12. https://doi.org/10.1007/s44163-024-00107-6

[22] Hao, X., Hongfeng, L., Jun, L., Nian, C. (2021). Survey on deep learning based image super-resolution. Computer Engineering and Application, 57(24): 51-60. https://doi.org/10.3778/j.issn.1002-8331.2105-0418

[23] Golla, M.R., Sharma, P. (2019). Performance evaluation of FaceNet on low resolution face images. In Communication, Networks and Computing: First International Conference, CNC 2018, Gwalior, India, pp. 317-325. https://doi.org/10.1007/978-981-13-2372-0_28

[24] Ullah, M., Hamza, A., Taj, I.A., Tahir, M. (2021). Low resolution face recognition using enhanced SRGAN generated images. In 2021 16th International Conference on Emerging Technologies (ICET), Islamabad, Pakistan, pp. 1-6. https://doi.org/10.1109/ICET54505.2021.9689885

[25] Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A. (2017). Photo-realistic single image super-resolution using a generative adversarial network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, pp. 105-114. https://doi.org/10.1109/CVPR.2017.19

[26] Wang, X.T., Yu, K., Wu, S.X., Gu, J.J., Liu, Y.H., Dong, C., Qiao, Y., Loy, C.C. (2018). Esrgan: Enhanced super-resolution generative adversarial networks. In Proceedings of the European conference on computer vision (ECCV) workshops, pp. 63-79. https://doi.org/10.1007/978-3-030-11021-5_5

[27] Jain, A. K., Klare, B., Park, U. (2011). Face recognition: Some challenges in forensics. In 2011 IEEE International Conference on Automatic Face and Gesture Recognition Workshops (FG 2011), Santa Barbara, CA, USA, pp. 726-733. https://doi.org/10.1109/FG.2011.5771338

[28] Lohiya, R., Shah, P. (2015). Face recognition techniques: A survey for forensic applications. International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), 4(4): 1-8.

[29] Bah, S.M., Ming, F. (2020). An improved face recognition algorithm and its application in attendance management system. Array, 5: 100014. https://doi.org/10.1016/j.array.2019.100014

[30] Zangeneh, E., Rahmati, M., Mohsenzadeh, Y. (2020). Low resolution face recognition using a two-branch deep convolutional neural network architecture. Expert Systems with Applications, 139: 112854. https://doi.org/10.1016/j.eswa.2019.112854

[31] Luo, G., He, G., Jiang, Z., Luo, C. (2023). Attention-based mechanism and adversarial autoencoder for underwater image enhancement. Applied Sciences, 13(17): 9956. https://doi.org/10.3390/app13179956