RESEARCH ARTICLE **OPEN ACCESS**

Manuscript received November 1, 2023; revised December 30, 2023; accepted December 30, 2023; date of publication January 20, 2024
Digital Object Identifier (**DOI**): https://doi.org/10.35882/jeemi.v6i1.349

How to cite: Kania Ardhani Putri, and Wikky Fawwaz Al Maki, Enhancing Pneumonia Disease Classification using Genetic Algorithm-Tuned DCGANs and VGG-16 Integration, vol. 6, no. 1, pp. 11-22, January 2024.

Enhancing Pneumonia Disease Classification using Genetic Algorithm-Tuned DCGANs and VGG-16 Integration

Kania Ardhani Putri, and Wikky Fawwaz Al Maki
School of Computing, Telkom University, Bandung, 40257 Indonesia
Corresponding author: Wikky Fawwaz Al Maki (e-mail: wikkyfawwaz@telkomuniversity.ac.id)

ABSTRACT Diagnostic complications arise from pneumonia, characterized by lung inflammation caused by alveolar fluid accumulation, particularly in regions with limited radiologists. To tackle this issue, a new method utilizes the VGG16 methodology for categorization, bolstered by genetic algorithms. In addition, Deep Convolutional Generative Adversarial Networks (DCGANs) improve the dataset by adding fake X-rays of pneumonia. Genetic algorithms are used to optimize hyperparameters in classification tasks. In contrast, DCGANs are employed to increase data augmentation techniques, boosting models' accuracy in identifying and categorizing pneumonia cases. The study partitioned a dataset into training, testing, and validation sets for pneumonia X-ray pictures. The training of GANs entails utilizing both generators and discriminators to produce increasingly realistic pictures gradually. The genetic algorithm enhances the hyperparameter tuning process, resulting in a substantial increase in accuracy. Initially, VGG16 achieved a success rate of 89.50% and a fitness score of 87.50%. Post-optimization and DCGAN augmentation, accuracy climbed to 95.50%, and F1-Score improved to 94.75%. This study combines genetic algorithms and DCGANs to create a model that can produce genuine pneumonia X-ray pictures and enhance categorization accuracy.

INDEX TERMS Pneumonia, Deep Learning, VGG16, Genetic Algorithm, Deep Convolutional Generative Adversarial Network,

I. INTRODUCTION

A bacterial, viral, or fungal infection is usually the cause of pneumonia, commonly referred to as a lung infection. Pneumonia is an inflammatory disease affecting one or both lungs [1]. The disease irritates the lungs, especially the air sacs, which might hold liquid or discharge and cause hacking and breathing troubles [2]. Fever, coughing, and sputum production are common symptoms [3], comparable to other non-infectious respiratory disorders. A case of pneumonia may be minor or fatal. Thus, it is imperative to identify and treat pneumonia promptly to reduce the high fatality rate, particularly in children [4].

The most common infectious cause of mortality for children globally is pneumonia. Its effects were evident in 2019 when 740,180 fatalities in kids younger than five happened, making up around 14% of all deaths in that age group. Twenty-two percent of all pediatric fatalities between the ages of one and five were due to pneumonia. Fortunately, with medications such as antibiotics and antivirals, pneumonia may be controlled [5]. As a result, prompt identification of pneumonia is crucial for the efficient administration of the proper care [6].

Prevention of complications that may result in mortality largely depends on early detection and treatment of pneumonia [1]. For seasoned radiologists [1], diagnosing pneumonia from such images remains challenging due to the significant role of deep learning, mainly through artificial neural networks (JST) or convolutional neural networks (CNN) [7]. Conventional methods, such as X-ray examination, have become the standard worldwide for detecting pneumonia.

Pneumonia frequently presents unclearly on X-rays, is easily misdiagnosed, and resembles a wide range of other benign conditions [2]. Although several research studies have been conducted recently, pneumonia detection remains difficult. Islam S.'s research on compressed sensing with deep learning for identification resulted in a remarkable accuracy of

96.48%. [8]. Rachna Jain and colleagues also used VGG-16 and artificial neural networks to detect pneumonia from chest X-rays, achieving an accuracy of 92.31% [4]. Similarly, Sumit Gupta, Harsh Sharma, and colleagues employed a CNN for the 90% accuracy and thresholding approach for pneumonia discovery from chest X-beam pictures utilizing highlight extraction and arrangement [2].

Technological innovations like generative adversarial networks (GANs) have helped overcome the limitations of small sample sizes and have progressed pneumonia detection studies [7]. Previous research utilizing diverse techniques provides valuable insights for enhancing the accuracy of pneumonia diagnosis. In this regard, a novel strategy that combines genetic algorithms with GANs seeks to increase the precision and dependability of illness identification and categorization procedures.

Recent studies have explored advanced techniques, such as SVM, CNN, and deep learning, to enhance pneumonia diagnosis accuracy [8]. The challenging nature of pneumonia detection, particularly on X-rays, has prompted the investigation of novel strategies. In this context, optimizing algorithms using genetic algorithms (GA) in conjunction with generative adversarial networks (GANs) has shown promise.

Previous investigations have also employed VGG16 as a classification approach in their research. For instance, a study [9] leverages VGG16 as an attribute extractor and employs transfer learning as its primary classification technique. Furthermore, a study [10] alters and adapts the VGG16 model to yield high accuracy. In [11], the authors employed VGG16 to categorize COVID-19 pneumonia automatically.

Research spanning diverse domains, from mosquito larvae categorization and image enhancement to multi-objective evolutionary algorithms [12], demonstrates the versatility of GAs. The fusion of GAs with GANs has proven effective in generating artificial human genomes [13] and detecting COVID-19 in medical images [14]. GAs also show promise in enhancing the stability and performance of GANs [15], fine-tuning parameters for image categorization, and optimizing medical image segmentation.

Recent efforts extend the application of GAs to optimize models for X-ray image COVID-19 detection, resulting in notable improvements in accuracy [16]. These studies underscore the potential of GAs in tandem with GANs or image-processing techniques to enhance performance across diverse applications [17]. This research aims to contribute to this evolving field by introducing a novel strategy combining genetic algorithms with GANs to improve the precision and reliability of pneumonia identification and categorization. Based on the research objectives, the contributions made in this research include the following:

a. Dataset development and preprocessing using the normalization approach.
b. Development of training data on the system using the DCGAN training data model.
c. Creating a genetic algorithm hyperparameter tuning.
d. Integrating the VGG16 classification model with GA.

II. MATERIAL AND METHODS

The system of classification of pneumonia disease has stages that will be passed based on FIGURE 1. These stages start with dataset assortment, information preprocessing, making another picture from the DCGAN generator results, genetic algorithm hyperparameter tuning, and arrangement utilizing VGG16. Then, it will be evaluated with a confusion matrix to discover its precision and accuracy.

The dataset will be divided into two distinct sets: one for training the model phase and another for testing its performance. Following the training phase, the subsequent classification phase will be carried out with the help of VGG16. DCGAN [18] will be used for data preprocessing and modeling functions during the training phase. In the subsequent phase, which is the testing phase, preprocessing will apply. Additionally, the act of standardizing data and altering the dimensions of the data so that they are in alignment with the layers that are utilized in the classification process is included in the preprocessing itself.

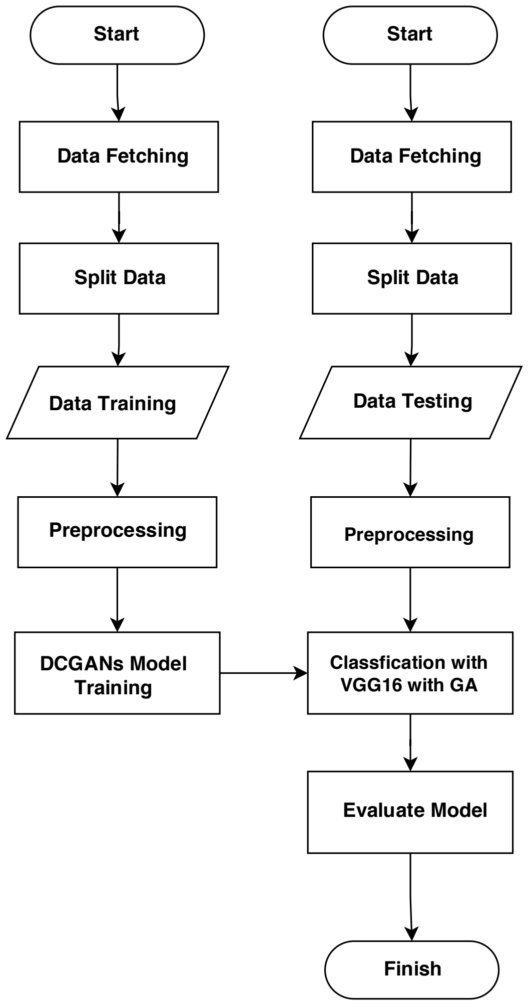


FIGURE 1. Research process system

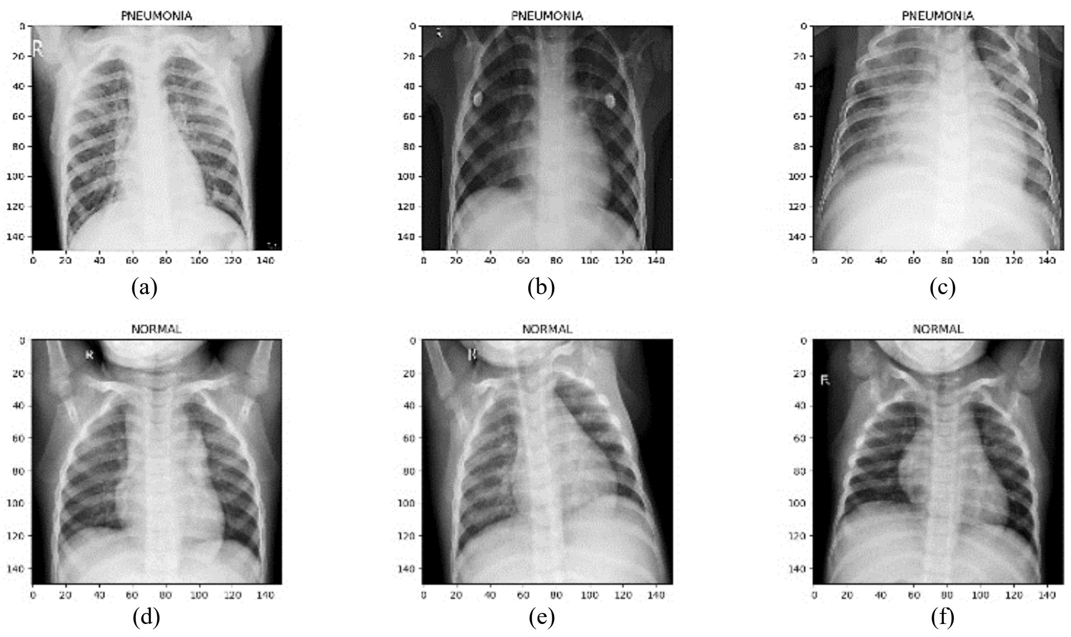


FIGURE 2. (a) Normal chest person 1, (b) Normal chest person 2, (c) Normal chest person 3, (d) Pneumonia chest person 1, (e) Pneumonia chest person 2, (f) Pneumonia chest person 3.

A. DATASET

This study utilized data collection from Kaggle, a site renowned for offering diverse datasets for research and development endeavors. This data set focuses on pneumonia-related X-ray images, as depicted in FIGURE 2. A specialized medical facility in Guangzhou, focusing on healthcare for women and children [19], has provided the dataset under a license arrangement. The classifier will partition the acquired data into multiple segments-related X-ray images, as depicted in FIGURE 2.

A specialized medical facility in Guangzhou, focusing on healthcare for women and children [19], has provided the dataset under a license arrangement. The classifier will partition the acquired data into multiple segments. Then, the training dataset comprises 3875 images indicative of infected lungs and 1341 images depicting normal lung conditions. Similarly, the testing dataset includes 390 pneumonia-related X-ray images and 234 images portraying normal lung conditions. Lastly, the validation dataset comprises 8 X-ray images representing normal lung conditions and eight images depicting pneumonia.

As a result of this segmentation, the model constructed with this dataset's help will be going through training, testing, and validation. The research on pneumonia identification using X-ray pictures is anticipated to benefit from this dataset, which was obtained under license from a reputable medical institution. It is expected that this dataset will provide authenticity and reliability.

B. PREPROCESSING

This research involves preprocessing pneumonia X-ray pictures to set up the dataset before it is engaged in the model preparation process. The initial dataset has an original image with dimensions of 150x150 pixels. We implemented a sequential application of various transformations for each X-ray image to align with the model requirements.

To reduce computational complexity and speed up the training process without sacrificing important information in the image, we resized the initial dimension of the X-ray image from 150x150 pixels to 64x64 pixels using the Resize transformation. Next, we apply the center crop transformation to the image, cropping it at its center to achieve a final size of 64x64 pixels. We apply this cropping to ensure the preservation and focus of relevant areas of the image, such as the lungs and surrounding areas.

From that point onward, tensor change changes the picture into a tensor portrayal. This transformation simplifies the image representation and facilitates the computational process. The model uses the normalized transformation to standardize the pixel values within the interval of [-1, 1]. This normalization aims to improve stability and convergence speed during model training.

The normalization approach is reported to have two primary purposes, as stated in several studies [20]. The primary goal is to provide standardized transportation methods for the same tissue type inside and among patients. The second aim is to establish a uniform interpretation of intensities across various locations within a given tissue.

C. DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS

DCGAN, a variant of GAN, incorporates convolution layers in the discriminator and generator [21]. DCGAN stands out from traditional GANs by leveraging convolutional neural networks (CNNs) instead of connected layers in its discriminator and generator, rendering it especially adept at image generation tasks [22]. FIGURE 3 depicts the architectural details of DCGAN, delineating two discernible networks: the discriminator and the generator.

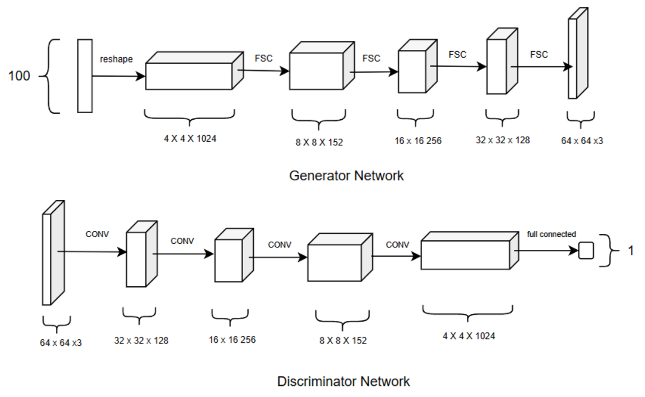


FIGURE 3. Generator and discriminator standard layer

Discriminators and generators in DCGAN inherently possess identical attributes, employing convolution layers within neural networks. FIGURE 3 depicts the architecture, where the generator distributes 100 dimensions within the network and inputs them into the convolution layer. There are four separate sequences of convolutional layers that have the potential to recover the depth of the image. Furthermore, it is imperative to note that if the final layer within the generator possesses dimensions of 64x64, it is essential for the initial layer within the discriminator to exhibit the exact dimensions of 64x64. This is because the discriminator cannot categorize or juxtapose generated images with the original photos in the dataset. Furthermore, the discriminator employs fully connected layers in its final layer.

The generator in Deep Convolutional Generative Adversarial Networks consists of many layers, as shown in .

TABLE 1. These layers collaborate to generate artificial graphics using the latent data provided. The primary convolution transpose layer transforms the latent data into a higher dimension as the initial step. This layer utilizes a 4x4 kernel with an 8-multiplier added to the number-generating feature.

To enhance stability and accelerate convergence, the model incorporates a normalization layer and ReLU activation after this layer. The repetitive process persists with the following layers, wherein each convolution transpose layer increases the spatial dimension of the picture while decreasing its feature count. This entails expanding the features by a predetermined coefficient and applying normalization and ReLU activation functions.

The last layer creates an output picture with pixel values ranging from −1 to 1, based on the desired image attributes, using the Tangent Hyperbolic (Tanh) activation function. As the training continues, this DCGAN generator uses the convolution transpose layer layers to expand and generate progressively higher-quality artificial pictures.

TABLE 1
Generator Layer Summary

Layer Name	Output Shape	Parameter
ConvTranspose2d Layer	(None, 4, 4, 512)	819200
Batch Normalization Layer	(None, 4, 4, 512)	2048
ReLU Activation Layer	(None, 4, 4, 512)	0
ConvTranspose2d Layer	(None, 8, 8, 256)	2097152
Batch Normalization Layer	(None, 8, 8, 256)	1024
ReLU Activation Layer	(None, 8, 8, 256)	0
ConvTranspose2d Layer	(None, 16, 16, 128)	524288
Batch Normalization Layer	(None, 16, 16, 128)	512
ReLU Activation Layer	(None, 16, 16, 128)	0
ConvTranspose2d Layer	(None, 32, 32, 64)	131072
Batch Normalization Layer	(None, 32, 32, 64)	256
ReLU Activation Layer	(None, 32, 32, 64)	0
ConvTranspose2d Layer	(None, 64, 64, 3)	3072
Tangen Hyperbolic Layer	(None, 64, 64, 3)	0

TABLE 1 illustrates the inclusion of 2D convolution transposition layers, which enhance sampling or augment the resolution of data representation. Following each ConvTranspose2d layer, a batch normalization layer is employed to enhance stability and accelerate the training process. Every ConvTranspose2d and batch normalization layer is then accompanied by an implemented ReLU activation layer to integrate non-linear attributes into the model. The ultimate convolution transposition layer of the generator employs a hyperbolic tangent activation function (Tanh) to generate an output inside the interval of -1 to 1, adhering to the customary methodology in generative models.

Each generator layer exhibits an identical pattern to the discriminator's and adheres to the same equation within each layer. The generator and discriminator layers can be considered reflective surfaces, where the dimensions of the generator output, such as 64x64, correspond to the dimensions of the discriminator input. Likewise, the discriminator can employ the computational formula utilized by the generator.

Furthermore, the architecture of DCGAN incorporates an additional component known as the discriminator. The discriminator's primary function is to differentiate between authentic pneumonia and artificially generated images. During the initial training phase, the discriminator receives the original pneumonia image and the image produced by the preceding generator. Subsequently, an adversarial comparison will take place. Throughout the comparison process, the generator will consistently enhance the quality of the fabricated data. Simultaneously, the discriminator will enhance its cognitive abilities in discerning the images by utilizing backpropagation. This process persists until it achieves a state of equilibrium, wherein the generator generates very persuasive data, posing a challenge for the discriminator to differentiate it. As indicated in TABLE 2, the discriminator comprises numerous convolution layers that constitute its architecture. Furthermore, the convolution layers employed are self-organizing convolution layers.

TABLE 2
Discriminator Layer Summary

Layer Name	Output Shape	Parameter
ConvTranspose2d Layer	(None, None, None, 64)	3072
ReLU Activation Layer	(None, None, None, 64)	0
ConvTranspose2d Layer	(None, None, None, 128)	131072
Batch Normalization Layer	(None, None, None, 128)	512
ReLU Activation Layer	(None, None, None, 128)	0
ConvTranspose2d Layer	(None, None, None, 256)	524288
Batch Normalization Layer	(None, None, None, 256)	1024
ReLU Activation Layer	(None, None, None, 256)	0
ConvTranspose2d Layer	(None, None, None, 512)	2097152
Batch Normalization Layer	(None, None, None, 512)	2048
ReLU Activation Layer	(None, None, None, 512)	0
ConvTranspose2d Layer	(None, None, None, 1)	8192
Sigmoid Activation Layer	(None, None, None, 1)	0

During the initial phase of training the discriminator, it is necessary to assign random weights. Arbitrary weights are vital in introducing early variability to the training model. After setting these arbitrary weights, the discriminator begins its initial training phase by employing the convolution layer. The primary function of this layer is to alter the proportions of the incoming image. After the modification of dimensions, the Rectified Linear Unit (ReLU) is implemented.

Implementing the Rectified Linear Unit (ReLU) layer requires making essential choices about constructing the neural network. The ReLU activation function operates on real numbers using the formula $\sigma(x) = max(x, 0)$. Neural networks possess a multitude of layers, which constitute their architecture. This structure includes a concealed layer with a corresponding input dimension referred to as p_0. The input dimension is obtained from the width vector function, denoted as $p = (p_0, \ldots, p(L+1))$. Another approach entails the utilization of a modified activation function that is applied to individual elements within the y vector. The shift is applied to every aspect that is not utilized in the activation of the ReLU layer. The vectors $y = (y_1, \ldots, y_r)$ and $v = (v_1, \ldots, v_r)$ belong to the set of real numbers, $\mathbb{R}_r$. The shifted activation function $\sigma_v\ (y)$ is obtained by taking the transpose of the vector $((\sigma(y_1 - v_1), \ldots, \sigma(y_r - v_r)))$ where each shifted element is converted into an accurate value in the set of real numbers, $\mathbb{R}_r$. The neural network is then represented as a function of the Eq. (2)[23]

$$f(\boldsymbol{x}) = W_{L+1} \circ \sigma_{v_L} \circ \ldots \circ W_2 \circ \sigma_{v_1} \circ W_1\, \boldsymbol{x}, \quad \boldsymbol{x} \in R \qquad (2)$$

In Eq. (2), the multilayer neural network, denoted as $f(x)$, takes an input x. Each layer in the network is represented by a weight matrix $W\ell$ and an activation function vector $V\ell$. Neurons in the $\ell - th$ layer are labeled as $P\ell$, while those in the ℓ-1st layer are labeled as $P\ell - 1$. The length of the activation function vector $V\ell$ is Pℓ. Applying the activation function sigma to elements at the middle value requires the ℓ minus t h layer's output. Specialized networks designed for regression tasks predict a single continuous value $(P\ell + 1 = 1)$. The overall expression succinctly combines the weight matrix, activation function, and input x.

During the subsequent step of this process, supplementary characteristics are included in every layer including a batch normalization layer. This stratum performs a pivotal function in augmenting the stability and speeding the convergence of the model. The quantity of characteristics increases in correlation with the profundity of the framework, and the expression employed to designate this tally is the Number Discriminator Feature (NDF).

The scale (γ) and shift (β) enable the network to acquire the most suitable scale and shift for each feature through learning. The normalization layer reduces internal covariate shifts, stabilizing and accelerating the training process. Then, the batch normalization algorithm was explained in Eq. (3) [24]. We explain the dimensions and offset of the transformation as follows.

$$y_i = \gamma \hat{x}_i + \beta \qquad (3)$$

After comprehending the method for determining the transformation using dimensions and offset values, the subsequent step entails computing a fresh value within this formula. Eq. (4) [24] calculates the specific value from standardized data, defined as normalized data $(\hat{x}_i)$.

$$\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 - \epsilon}} \qquad (4)$$

The ith data point in the dataset is denoted as x_i. representing its starting value. It has the potential to represent several types of data, such as measurements or observations that are relevant to the variable being studied. Subsequently, we employ the symbol μ. to represent the arithmetic, mean or average of all the values in the dataset. Next, we examine the variance of the dataset by dividing it by a scaled estimate of the data's dispersion. Prior to computing the normalized data value $(\hat{x}_i)$, it is necessary to obtain the variance. Eq. (5) [24] shown below specifies the computation of the variance:

$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu)^2 \qquad (5)$$

To unravel the intricacies of variance (σ^2), we embark on a journey of statistical exploration. Picture it as a quest where each data point (x_i) plays a unique role, engaging in a dance with the mean (μ). Our mission is to decipher the rhythm of these deviations by squaring their every move. This entails a grand summation, symbolized by the mighty $\sum_{i=1}^{m}$, encompassing the squared differences for each data point. Then, the moment of reckoning arrives, as we divide this collective spectacle by the total number of data points (m), signified by the mystical 1m1. These squared differences, akin to both positive and negative departures from the mean, unveil the tale of variability within the sample.

Subsequently, utilizing the variance σ^2 approach, we must calculate the mean value of the batch data, denoted as (μ), as expressed in Eq. (6) [24]. A batch input value, represented as $x_1 \sim x_m$, consists of an m sample.

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{6}$$

To utilize the variance (σ^2) technique in the provided context, it is necessary to calculate the sum of the mean value. The value will be calculated by taking the sum of all data points and dividing it by the total number of data points (m). The act of aggregating the various data points ($\sum_{i=1}^{m} x_i$) and subsequently dividing by m yields a representative value that effectively represents the central trend.

Using a convolution layer with a single output channel, the final layer creates a confidence score indicating whether the provided picture is accurate or false. The outcome of this layer is then processed by a sigmoid activation function designed to produce a scalar value confined within the range of 0 to 1. This scalar value is a decisive metric, denoting the probability or likelihood that the input image is genuine. In essence, this final layer acts as the discriminator's verdict, assigning a quantitative measure to the authenticity of the presented visual data.

Besides that, when training the DCGANs model, there will be a Loss Discriminator and Loss Generator Eq. (8)[25]. The loss functions for both main layers are constructed using the min-max function, a fundamental approach within the Generative Adversarial Networks (GANs). The min-max function will assess and record the adversarial nature of the DCGAN training process. The Generator minimizes the loss obtained from training in forming the synthetic image in this function. At the same time, the discriminator will maximize the loss obtained when validating the synthetic image.

$$\begin{aligned} &min_G max_D V(D,G) \\ &= E_{x\sim P_{data(x)}}[logD(x)]E_{x\sim P_{data(x)}}[\log\left(1-D\left(G(z)\right)\right)] \end{aligned} \tag{8}$$

$D(x)$ signifies the likelihood of discriminator networks D distinguishing x as an authentic sample. In addition, $G(z)$ refers to a sample produced by the generator network G using noise z, and $D(G(z))$ measures the likelihood of the discriminator network D correctly identifying $G(z)$ as a valid sample. The generator minimizes loss by providing synthetic data that closely mimics actual samples. The discriminator will maximize the loss when validating the images. This dynamic interplay establishes an equilibrium where the generator is driven to create realistic data that poses a substantial challenge for the discriminator. This formulation creates a balance where the generator generates realistic data that challenges the discriminator.

D. GENETIC ALGORITHM

One of the suitable algorithms for handling complicated optimization issues that are challenging to resolve using traditional techniques is the genetic algorithm [17] [21]. The heuristic approach is a strategy based on empirical criteria or intuition to discover a better solution than the already found [17]. Employing this heuristic approach effectively tackles optimization issues related to container management [20].

This research employs a genetic algorithm (GA) [15] to optimize the hyperparameters of a VGG16 model that is trained on a dataset for pneumonia detection. Consequently, identifying the best hyperparameters entails following the stages stated in FIGURE . The technique starts by initializing a population of various sets of hyperparameters, such as learning rate, maximum depth, batch size, etc., representing viable solutions.

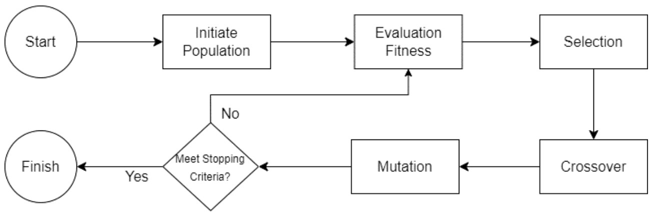


FIGURE 4. Common genetic algorithm flows

A group of individuals is formed, encompassing the variables slated for optimization called chromosomes. The population is explicitly characterized in Eq. (9) [26], where the row vector represents the solution vector in a singular iteration. The matrix X consolidates all individuals engaged in the optimization procedure.

$$pop = \begin{bmatrix} pop_1 \\ pop_2 \\ pop_3 \\ \vdots \\ pop_n \end{bmatrix} \tag{9}$$

By understanding the formula mentioned above, it is possible to determine the total population for each generation, represented by n-population. Eq. (10) [26] depicts the total population throughout the execution, with n denoting the number of iterations and m indicating the number of chromosomes.

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} & \cdots & x_{1,m} \\ x_{2,1} & x_{2,2} & x_{2,3} & \cdots & x_{2,m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & x_{n,3} & \cdots & x_{n,m} \end{bmatrix} \tag{10}$$

In addition, let us represent the percentage of the population comprised of parents who have demonstrated better fitness values by the letter P. When the percentage P is set at 30%, the fitness-based uppermost 30% of the population is deemed eligible to participate in the selection process linked to recombination or crossover. A progenitor is designated, which denotes as x_i^n, which denotes specific generation and individual, if its fitness level is among the highest 30%. Failure to meet these criteria disqualifies the individual from the selection process.

In the context of Eq. (11) [26], the total population is denoted as x^n, and the variable t represents the number of iterations. Within this population, parents are identified at a specific iteration, organized in an array denoted as $x_{iselect}^n$(t). Additionally, another array represents parents from the preceding iteration, labeled as $x^n_{iselect}\ (t-1)$. A specific notation indicates an individual's position in the population, namely $[x^n_{iselect}\ (t-1), x^{\text{in}}]$, where x^{in} refers to the individual at the nth position. This notation implies that the individual at the nth position, x^{in}, is combined with the preceding parent. This process helps keep track of individuals and their positions in the evolving population across iterations.

$$x^n_{iselect}(t) =$$
$$\begin{cases} [x^n_{iselect}(t-1), x^n_i] & if\ fitness(x^n_i)\ \epsilon\ (P \times x^n) \\ x^n_{iselect}(t-1) & if\ otherwise \end{cases} \quad (11)$$

The F1 score, computed using the validation set, quantifies the performance of our model. We employ hyperparameter tuning to assess parameter configurations and achieve optimal discrimination between pneumonia and regular patients in our model. The optimal fitness value for future use can be determined by referring to Eq. (12)[27]. The fitness value is derived from multiple training sessions on a generation and population.

$$ft = \sum_{K=1}^{M} R_s\ /\ M \quad (12)$$

Variable s represents the inclusion of $\frac{m}{k}$ in the summation term, and specific parameters are targeted for enhancement. In this scenario, R_s signifies the rule selected, and M denotes the total regulations count. The fitness value selected rules determine f sub t for each chromosome. Each chromosome undergoes evaluation against the fitness function. Only those solutions that meet the criteria of the fitness function are chosen to engage in the reproductive process, involving either crossover or mutation.

By evaluating their F1 scores, the genetic operations include finding the most intelligent people and using the uniform crossover to create a new group of possible solutions. To introduce variability within the population, we implement mutation by making random alterations to a subset of hyperparameters [28]. The mutation process is shown in FIGURE . The mutation prevents every solution in the population from falling into the local optimum of the solved issue. Offspring are the consequence of crossing and are subject to random mutation. We can randomly choose bits in binary encoding to change them from 0 to 1 or 1 to 0.

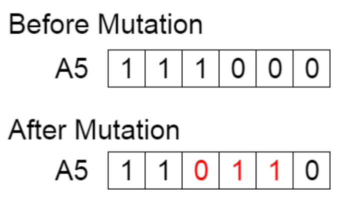


FIGURE 5. How mutation works.

The genetic algorithm incorporates the identified ideal parameters into the classification technique. The characteristics utilized are the ones of significance. The batch size controls the amount of memory utilized and determines the speed at which the model learns. Furthermore, the model utilizes other parameters to incrementally adapt to the current parameters. Epochs represent the frequency at which the model processes the complete dataset. These parameters aim to attain optimal precision and fitness value in the classification procedure.

E. VGG16 ARCHITECTURE

The VGG16 architecture will be utilized for classification to predict pneumonia versus X-rays[29]. One of the deep learning architectures frequently employed in classification is VGG16 [30]. The global average pooling, two thick layers for classification, and multiple blocks of convolutional and max-pooling layers (blocks 1 through 5) [31] make up the VGG16 architecture model, for the specific layer will be retrieved on

TABLE 3. Each convolutional block aims to extract hierarchical characteristics from the input picture by combining several convolutional and max-pooling layers [30].

After the layer Convolutional block, VGG16 has layer Global average pooling, producing an average representation of the features across the spatial data. The last two thick layers are answerable for grouping, with the layer having convolution and ReLU activation[32]. In contrast, the last layer has one sigmoid activation unit suitable for binary classification tasks. The model has 20,090,177 parameters, 65,793 trainable parameters, and 20,024,384 untrainable parameters. This diversity of parameters reflects the model's high capacity to handle complex datasets.

TABLE 3
VGG16 Architecture Layer Summary

Layer Block	Layer Name	Output Shape
	Input Layer	None, None, None, 3
	Convolution transpose layer	None, None, None, 64
Block 1	Convolution transpose layer	None, None, None, 64
	MaxPooling2D Layer	None, None, None, 64
Block 2	Convolution transpose layer	None, None, None, 128
	Convolution transpose layer	None, None, None, 128
	MaxPooling2D Layer	None, None, None, 128
Block 3	Convolution transpose layer	None, None, None, 256
	Convolution transpose layer	None, None, None, 256
	Convolution transpose layer	None, None, None, 256
	Convolution transpose layer	None, None, None, 256
	MaxPooling2D Layer	None, None, None, 256
Block 4	Convolution transpose layer	None, None, None, 512
	Convolution transpose layer	None, None, None, 512
	Convolution transpose layer	None, None, None, 512
	Convolution transpose layer	None, None, None, 512
	MaxPooling2D Layer	None, None, None, 512
Block 5	Convolution transpose layer	None, None, None, 512
	Convolution transpose layer	None, None, None, 512
	Convolution transpose layer	None, None, None, 512
	Convolution transpose layer	None, None, None, 512
	MaxPooling2D Layer	None, None, None, 512
	GlobalAveragePooling2D Layer	None, 512
	Dense Layer	None, 128
	Dense Layer	None, 1

III. RESULT

The section will display the accuracy difference between VGG16 with and without genetic algorithms and DCGAN augmentation data. The data's preprocessing is necessary before its utilization in the categorization procedure. In addition, it will demonstrate the categorization analytics and DCGAN modeling of fresh false picture data.

A. MODELING RESULT USING DCGANs

In modeling and training using DCGANs, it is essential to have both a generator and a discriminator. This is because both components are necessary ingredients. The generator will produce and mold an original picture, which will then be converted into a forged image. After the synthetic image has been produced, the discriminator will proceed with a classification. The generator will then create a new image that displays greater authenticity once the discriminator identifies the picture being examined as fake [33].

The training approach of the adversarial network model will involve integrating a random perturbation that influences the discriminator's final prediction for the produced image. The generative model includes random noise, which pertains to a vector or distribution that follows a stochastic process. The transformation process plays a crucial role in data generation [34]. The produced data is characterized by variance and an element of surprise.

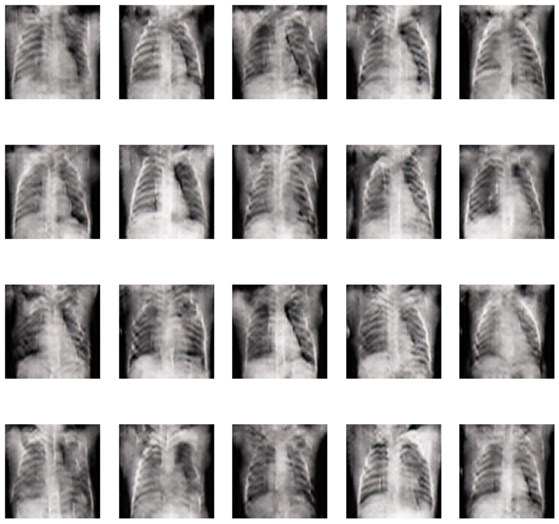


FIGURE 6. DCGANs modeling result

Comprehending the operation of the generator and identifying the suitable uproar vector to generate the required category by altering the noise vector is a simple process. Nevertheless, in this specific scenario, we utilize 100 noise vectors. The normalization layer uses a tiny batch size of 64 to train the model. The weights are assigned initially using a distribution with a median equal to the data and a deviation from the mean of 0.0002.

Both the discriminator and generator each comprise five layers in total. The generator consists of one deconvolution layer with Tanh activation on the final layer and four deconvolution levels with ReLU activation. The discriminator layer consists of four convolutional layers responsible for spatially processing the image. Following the processing of the picture in the preceding layer, the ReLU layer will function as the activator. Subsequently, the discriminator incorporates an extra layer known as the sigmoid layer.

During the training phase, the discriminator encounters both authentic and synthesized data. After several training epochs, the generator produces highly authentic pictures for further training [35]. The adequacy of the Superior DCGAN model was assessed using 100 generated images for each lung lesion category. The FIGURE 6 graphic below shows the results of modeling with DCGANs.

B. Discriminator and Generator Loss

In generative adversarial networks (GANs), a widely adopted loss function for training is the twofold cross-entropy loss. FIGURE 6 vividly illustrates this pivotal aspect of model training, providing a graphical depiction of the model training output. Here, the intricacies of generator and discriminator loss during training are visually presented, offering valuable insights into the dynamics of the training process. This graphic explicitly depicts the subtle details detected following the conclusion of 10 epochs, or 450 iterations, offering a comprehensive overview of the model's evolving performance throughout its training phase. Based on the chosen epochs, FIGURE 7 presents the training loss history derived from the observations.

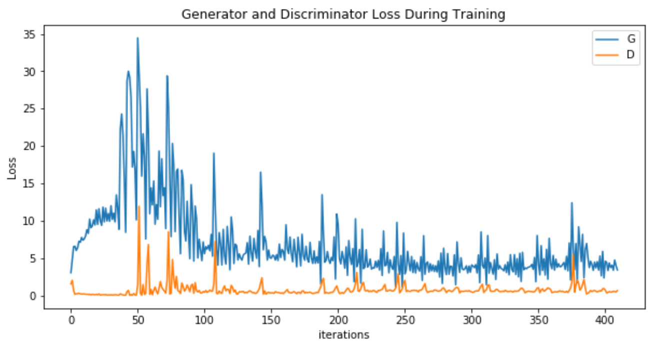


FIGURE 7. Generator and discriminator training loss history

In the first iteration, LD is relatively high (1.53) because the generator still produces unclear and unrealistic images, and the discriminator attempts to recognize the genuine image ($D(x)$) and that produced by the generator ($D(G(z))$. The resulting LG also displays a value of 3.03. $D(x)$ and $D(G(z))$ display different values, indicating that the discriminator has not been effective in distinguishing between real and fake images. As the training progresses, there is a significant decrease in LD, reaching low values such as 0.01 at epoch 1, indicating that the discriminator is getting better at distinguishing.

At the same time, the L_G increases sharply from iteration to iteration, reaching 8.42 at iteration 25. This indicates that the generator successfully improves its ability to create pictures nearer to the first picture. With a high L_G, the generator feels the squeeze to deliver an image that can "trick" the discriminator.

In addition, the values of $D(x)$ and $D(G(z))$ give an idea of the extent to which the discriminator can tell the difference between real and fake images. A high $D(x)$ value indicates the discriminator's ability to recognize the actual image, while a low $D(G(z))$ value indicates that the generator has successfully fooled the discriminator. Overall, the training results show positive progress, with the discriminator becoming more accurate and the generator becoming more effective in producing realistic images as epochs pass. Further evaluation can be done by considering other metrics, such as BAS, to validate the quality of the pictures created by the generator.

C. IMPROVING IMAGE CLASSIFICATION ACCURACY GENETIC ALGORITHM

Combining Deep Convolutional Generative Adversarial Networks (DCGANs) with Genetic Algorithms (GAs) is a promising way to improve image classification accuracy. DCGANs generate synthetic pneumonia X-ray images, enhancing the classification model's performance. Concurrently, genetic algorithms (GAs) are utilized to fine-tune parameters during classification system training.

To bolster the training of the classification model, we introduce pneumonia X-ray images as a supplement to the primary dataset. This augmentation increases the quantity of available data and enriches its diversity, ultimately enhancing the overall robustness of the model. Here, DCGAN functions as a supportive tool, effectively elevating the classification model's performance by generating and integrating additional data. The utilization of DCGAN proves instrumental in fortifying the capabilities of the classification model, ensuring a more comprehensive and effective training process.

To determine optimal parameters for the classification model, a genetic algorithm (GA) is employed. The Genetic Algorithm (GA) utilizes natural selection as inspiration to identify the optimal hyperparameter values, guided by the principle of "survival of the fittest." The specific parameters under consideration are the learning rate, epoch, and batch size.

Numerous layers make up the generator in Deep Convolutional Generative Adversarial Networks. These layers work together to construct fake visuals from the supplied latent data. First, the first convolution transpose layer is responsible for converting the latent data into a larger dimension. This layer has a 4x4 kernel with an 8-multiplier applied to the number generator feature.

During the recombination stage, the selected parents combine their parameters to produce offspring that inherit traits from each parent. Additionally, mutation introduces changes to some parameter values, preserving genetic diversity and averting premature convergence to suboptimal solutions. The outcome is a new generation comprising both surviving parents and their offspring. GA evolutionarily optimizes parameter values by retaining the best parents, significantly improving pneumonia X-ray image classification accuracy. TABLE 4 provides a detailed account of the changes in F1-Score during the discovery of the optimal parameters for classification.

TABLE 4
Fitness score changes history

	Generation 1	Generation 2	Generation 3	Generation 4
Parent 1	81.25%	62.50%	77.85%	75.00%
Parent 2	81.25%	92.55%	90.35%	75.00%
Parent 3	75.00%	90.23%	87.95%	81.25%
Parent 4	68.75%	77.32%	88.00%	82.25%
Parent 5	68.75%	87.00%	88.00%	72.75%
Parent 6	93.75%	87.00%	85.25%	72.75%
Parent 7	93.75%	94.75%	80.25%	83.85%
Parent 8	75.00%	82.50%	85.55%	80.75%

D. CLASSIFICATION USING VGG16 ARCHITECTURE

The VGG16 architecture, a convolutional neural network (CNN)-derived architecture, will be employed for data classification. To enhance the classification process, we have implemented a variety of parameters derived from the Optimization Using Genetic Algorithm. The outcomes of generating novel pictures by DCGAN will also be incorporated into the training data. To achieve optimal categorization and enhance data distribution, it is imperative to implement effective strategies.

The models will receive performance testing using the testing data, while the training data will be utilized to create the DCGAN and classification models. The DCGAN model will be added to the training set to extend it. For training, the classification model will use the more extensive dataset.

Once the model has been trained and compiled, the categories will be determined utilizing the pre-built layers. Ten epochs of training were conducted using the validation data extracted from the provided dataset to validate the model. The validation data in this dataset has been verified to ensure that it accurately represents the presence of pneumonia. The training process employs the learning rate supplied by TensorFlow to mitigate the risk of the model achieving excessive precision. In addition, the training process incorporates the Reduce LR on Plateau recall.

The determination of batch size and learning rate is predicated on the outcomes of a genetic algorithm optimization in which the optimal learning rate and batch size are chosen to optimize the model's performance. At each epoch of the training procedure, performance indicators, including loss and accuracy, were observed. The training process results indicated that the model attained a train loss of 0.26 and an accuracy rate of 89% on the train data.

Predefined test data was utilized to assess the model. The model's performance on the test data is assessed, revealing an accuracy of 84% during the evaluation process and a loss of 0.35. The obtained outcomes demonstrate that the model possesses commendable generalizability and can accurately classify novel data. This research contributes to advancing deep learning-based X-ray pneumonia disease detection systems.

By applying a genetic algorithm to optimize the hyperparameters, the optimal learning rate and batch size were determined to be 0.02 and 256, respectively, with an accuracy of 95% and an F1-Score of 94%. When comparing the manual approach with and without the optimal learning rate and batch size, it is evident that the latter achieved 89% accuracy and an 88% F1-Score for testing data, a 0.02 learning rate, and a 128-batch size, respectively. The comparison between the utilization and non-utilization of a genetic algorithm is presented in

TABLE *5*. Classification performance is significantly enhanced when the optimal hyperparameter is determined using a genetic algorithm; this suggests that parameter optimization by this technique may yield superior outcomes compared to the manual approach.

TABLE 5
Comparison Genetic Algorithm

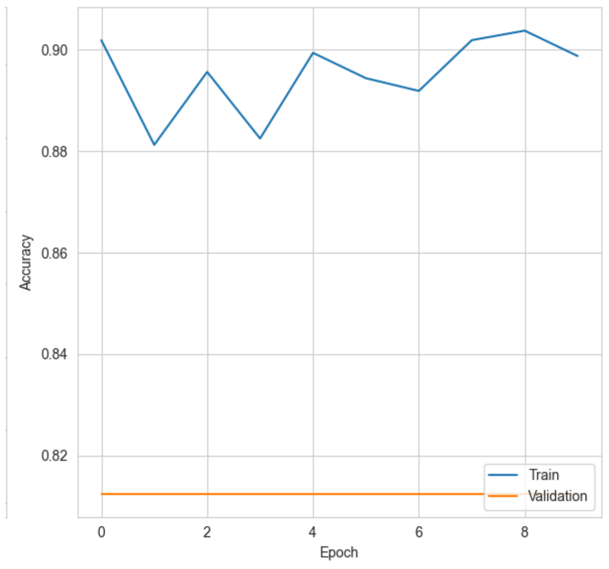
	With GA	Without GA
Accuracy	95.50%	89.50%
F1-Score	94.75%	87.50%

Furthermore, another conventional experiment assesses various learning rates and batch sizes by incorporating them into a classification model. The model yields varying outputs, with a learning rate of 0.2 and a batch size 32. These inputs result in an accuracy of 81.50% and a fitness score of 78.25%. Furthermore, an experiment was conducted utilizing a rate of learning of 0.002 and a size batch of 256, yielding a distinct outcome compared to the prior test. Using these characteristics yields an accuracy of 90.75% and a fitness score of 89.50%. Additional experiments have been conducted, and TABLE 6 has been created using these testing results. One can observe varying learning rates and batch sizes upon examining the table.

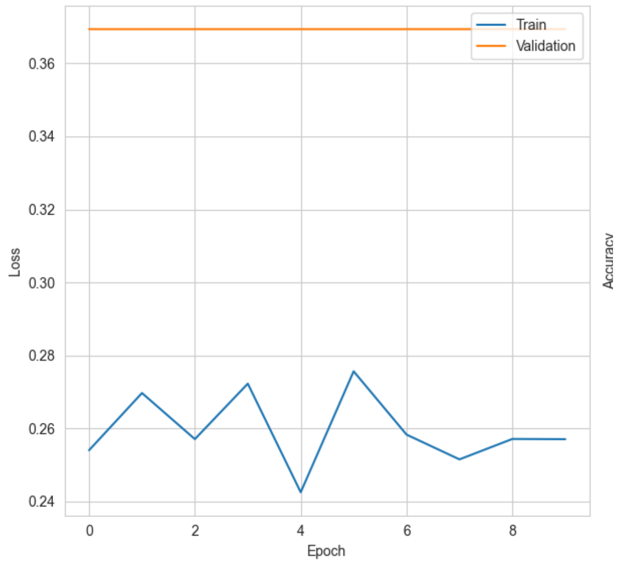
TABLE 6
Testing with different hyperparameters.

Batch Size	Learning Rate	Accuracy (%)
32	0.2	81.50
	0.02	81.75
	0.002	80.95
64	0.2	83.45
	0.02	82.85
	0.002	83.15
128	0.2	86.85
	0.02	89.50
	0.002	88.50
256	0.2	92.75
	0.02	95.50
	0.002	90.75

When training using a learning rate of 0.02 and a batch size of 128, the visible progression of loss and accuracy during model training in FIGURE 8 reveals the complex dynamics that crystallize over some time of ten epochs. During the initial epoch, the model demonstrated noteworthy performance, attaining an accuracy of 90.19% and a loss value of 0.2540. However, it is essential to note that the model's performance declined in the following epochs. Variations in the loss value and accuracy were observed from the second to sixth epochs. The loss value varied from 0.2425 to 0.2756, while the accuracy fluctuated between 88.13% and 89.44%. This indicates that the model is adjusting to the complexity of the data and may be responding to significant changes in the distribution of features.



(a)



(b)

FIGURE 8. **(a) Training accuracy evolution, (b) Training loss evolution.**

The model stabilizes in subsequent epochs, maintaining loss and accuracy values of approximately 0.2571 and 89.88%, respectively. Nevertheless, assessing this stability concerning the validation data is crucial, demonstrating a consistent accuracy of 81.25% and a loss value of 0.3694. The model experienced a decline in accuracy from 90.38% to 89.88% and a loss value of 0.2571 at the tenth and final epoch.

Thus, the complexity of the evolution of loss and accuracy reflects the dynamics of model training and provides insight into the model's response to changes and complexity in the dataset and efforts to prevent overfitting through learning rate management strategies. Further evaluation of train and test data is required to better understand the model's generalization to previously unseen data.

IV. DISCUSSION

This study's use of DCGAN (Deep Convolutional Generative Adversarial Network) and GA (Genetic Algorithm) significantly improves image classification results, especially when correctly identifying X-ray pneumonia. This study employs a hybrid approach of GA and DCGAN to enhance the classification model's performance by identifying the optimal parameters and augmenting the data. Throughout the training phase, the DCGAN model generates synthetic pneumonia X-ray images, which serve as supplementary data. In addition, the genetic algorithm will seek the most optimal hyperparameters for the classification model. The hyperparameters encompass a model's learning rate, memory capacity, and number of iterations.

DCGANs are employed to generate fake pictures to extend the classification data. While generating synthetic images, a preprocessing step is performed to resize the original image from a resolution of 150x150 pixels to 64x64 pixels. To enhance the clarity of the X-ray lung shape and minimize interference from noise, we employ a center crop technique after downsizing the original image from 150x150 to 64x64. Next, we will proceed with developing layer generators and discriminators in DCGAN. Both layers employ a convolutional layer as their primary component. The discriminator layer distinguishes between the original and synthetic images by incorporating a fully linked layer to carry out the comparison procedure. The generated image will serve as augmentation data in the training dataset.

The application of genetic algorithms (GA) in categorization yielded exceptional outcomes. During the testing phase, utilizing a Genetic Algorithm (GA) resulted in a notable improvement in accuracy and fitness match scores. Specifically, the accuracy score reached an impressive 95.50%, and the fitness match score reached 94.75% when GA was employed. In contrast, when GA was not utilized, the accuracy score was only approximately 89.50%, and the fitness match score was approximately 87.50%. This demonstrates that the implementation of genetic algorithms substantially positively impacts overall performance. Genetic Algorithm (GA) exhibits superior efficacy to conventional parameter search techniques. Nevertheless, the downside of genetic algorithms resides in the substantial computing expense associated with the exploration of hyperparameters. The procedure entails conducting experiments with several generations, each with a parent that embodies the categorization parameters. The parent with the highest performance was produced in the second generation, with a learning rate of 0.02 and a batch size 256. The researchers subsequently employed the most optimal progenitors in the process of categorization utilizing the VGG16 model, culminating in an accuracy rate of 95.50% and a fitness match score of 94.75%.

Several studies have already researched the utilization of VGG16 as a classification method. The study [9] employs VGG16 as a feature extraction method in transfer learning and achieves an accuracy of 95.07%. According to research [10], the primary classification method used is VGG16, which achieves an accuracy of 89.1%. Research [11] has utilized VGG16 as a classification technique, yielding an accuracy of 83.7%. Compared to the earlier studies, this study's model produces superior outcomes.

This research has deficiencies. These limitations arise due to the high computational expenses and time-consuming nature despite the favorable specifications of the tools. The search for genetic algorithm hyperparameters involves computationally high charges due to the model and parameters involved. The genetic algorithm undergoes multiple iterations to choose the optimal parent. Despite its high accuracy and fitness score, this drawback is indisputable due to its time-consuming nature. This research did not address the methodology for resolving the problem. Subsequent studies will explore this issue.

V. CONCLUSION

In conclusion, this research demonstrates the successful integration of Deep Convolutional Generative Adversarial Networks (DCGANs) and Genetic Algorithms (GAs) to improve the accuracy of pneumonia detection in X-ray images. The DCGAN effectively generates synthetic images, augmenting the dataset for training a classification model with VGG16 architecture. The iterative optimization process facilitated by GAs enhances the classification model's hyperparameters, resulting in a notable improvement in accuracy and F1-Score. The proposed methodology not only showcases the potential of synthetic data for training but also highlights the importance of optimizing model parameters through evolutionary algorithms, showcasing a 95.50% accuracy and 94.75% F1-Score with GAs, compared to 89.50% accuracy and 87.50% F1-Score without GAs.

This research enhances medical image analysis by providing a viable framework for utilizing generative models and optimization techniques. Integrating DCGANs and GAs addresses challenges in limited data scenarios, enhancing the robustness and accuracy of the image classification model. The substantial improvement in classification performance, as evidenced by the accuracy and F1-Score comparison, underscores the potential of this approach for enhancing the diagnostic capabilities of deep learning models in medical imaging applications.

ACKNOWLEDGMENT

We want to express our sincere gratitude to Telkom University, which has provided financial support for the completion of this research.

REFERENCES

[1] İstanbul AREL Üniversitesi, IEEE Engineering in Medicine and Biology Society, Institute of Electrical and Electronics Engineers. Turkey Section, and Institute of Electrical and Electronics Engineers, *Ayan, Enes, and Halil Murat Ünver. "Diagnosis of pneumonia from chest X-ray images using deep learning." 2019 Scientific Meeting on Electrical-Electronics & Biomedical Engineering and Computer Science (EBBT). Ieee, 2019.*

[2] Amity University and Institute of Electrical and Electronics Engineers, *Sharma, Harsh, et al. "Feature extraction and classification of chest x-ray images using cnn to detect pneumonia." 2020 10th international conference on cloud computing, data science & engineering (Confluence). IEEE, 2020.*

[3] D. R. Chandranegara, Z. Sari, M. B. Dewantoro, H. Wibowo, and W. Suharso, "Implementation of Generative Adversarial Network (GAN) Method for Pneumonia Dataset Augmentation," *Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control*, May 2023, doi: 10.22219/kinetik.v8i2.1675.

[4] R. Jain, P. Nagrath, G. Kataria, V. Sirish Kaushik, and D. Jude Hemanth, "Pneumonia detection in chest X-ray images using convolutional neural networks and transfer learning," *Measurement (Lond)*, vol. 165, Dec. 2020, doi: 10.1016/j.measurement.2020.108046.

[5] J. P. Metlay *et al.*, "Diagnosis and treatment of adults with community-acquired pneumonia," *Am J Respir Crit Care Med*, vol. 200, no. 7, pp. E45–E67, Oct. 2019, doi: 10.1164/rccm.201908-1581ST.

[6] J. A. Ramirez *et al.*, "Treatment of Community-Acquired Pneumonia in Immunocompromised Adults: A Consensus Statement Regarding Initial Strategies," *Chest*, vol. 158, no. 5. Elsevier Inc., pp. 1896–1911, Nov. 01, 2020. doi: 10.1016/j.chest.2020.05.598.

[7] D. Srivastav, A. Bajpai, and P. Srivastava, "Improved classification for pneumonia detection using transfer learning with GAN based synthetic image augmentation," in *Proceedings of the Confluence 2021: 11th International Conference on Cloud Computing, Data Science and Engineering*, Institute of Electrical and Electronics Engineers Inc., Jan. 2021, pp. 433–437. doi: 10.1109/Confluence51648.2021.9377062.

[8] S. R. Islam, S. P. Maity, A. K. Ray, and M. Mandal, "Deep learning on compressed sensing measurements in pneumonia detection," *Int J Imaging Syst Technol*, vol. 32, no. 1, pp. 41–54, Jan. 2022, doi: 10.1002/ima.22651.

[9] G. J. Chowdary, G. Suganya, M. Premalatha, and S. Ganapathy, "Impact Of Machine Learning Models In Pneumonia Diagnosis With Features Extracted From Chest X-Rays Using VGG16," 2021.

[10] Z. P. Jiang, Y. Y. Liu, Z. E. Shao, and K. W. Huang, "An improved VGG16 model for pneumonia image classification," *Applied Sciences (Switzerland)*, vol. 11, no. 23, Dec. 2021, doi: 10.3390/app112311185.

[11] M. Nishio, S. Noguchi, H. Matsuo, and T. Murakami, "Automatic classification between COVID-19 pneumonia, non-COVID-19 pneumonia, and the healthy on chest X-ray image: combination of data augmentation methods," *Sci Rep*, vol. 10, no. 1, Dec. 2020, doi: 10.1038/s41598-020-74539-2.

[12] C. He, S. Huang, R. Cheng, K. C. Tan, and Y. Jin, "Evolutionary Multiobjective Optimization Driven by Generative Adversarial Networks (GANs)," Oct. 2019, [Online]. Available: http://arxiv.org/abs/1910.04966

[13] B. Yelmen *et al.*, "Creating artificial human genomes using generative neural networks," *PLoS Genet*, vol. 17, no. 2, Feb. 2021, doi: 10.1371/JOURNAL.PGEN.1009303.

[14] M. A. A. Albadr, S. Tiun, M. Ayob, F. T. Al-Dhief, K. Omar, and F. A. Hamzah, "Optimised genetic algorithm-extreme learning machine approach for automatic COVID-19 detection," *PLoS One*, vol. 15, no. 12 December, Dec. 2020, doi: 10.1371/journal.pone.0242899.

[15] H. E. Bate, "Genetic Algorithm-based Optimization of Generative Adversarial Networks and its Applications."

[16] B. Gülmez, "A novel deep neural network model based Xception and genetic algorithm for detection of COVID-19 from X-ray images," *Ann Oper Res*, vol. 328, no. 1, pp. 617–641, Sep. 2023, doi: 10.1007/s10479-022-05151-y.

[17] J. Gu, Y. Shen, and B. Zhou, "Image Processing Using Multi-Code GAN Prior," 2019.

[18] C. Dewi, R. C. Chen, Y. T. Liu, and S. K. Tai, "Synthetic Data generation using DCGAN for improved traffic sign recognition," *Neural Comput Appl*, vol. 34, no. 24, pp. 21465–21480, Dec. 2022, doi: 10.1007/s00521-021-05982-z.

[19] M. K. D. Z. K. Kermany D.S.; Goldbaum M.; Zhang K.; Goldbaum, "Large Dataset of Labeled Optical Coherence Tomography (OCT) and Chest X-Ray Images," 2018.

[20] L. J. Isaksson *et al.*, "Effects of MRI image normalization techniques in prostate cancer radiomics," *Physica Medica*, vol. 71, pp. 7–13, Mar. 2020, doi: 10.1016/j.ejmp.2020.02.007.

[21] A. Kulkarni, D. Chong, and F. A. Batarseh, "Foundations of data imbalance and solutions for a data democracy," *Data Democracy: At the Nexus of Artificial Intelligence, Software Development, and Knowledge Engineering*, pp. 83–106, Jan. 2020, doi: 10.1016/B978-0-12-818366-3.00005-8.

[22] M. Liu and J. Yang, "Image classification of brain tumor based on channel attention mechanism," in *Journal of Physics: Conference Series*, IOP Publishing Ltd, Oct. 2021. doi: 10.1088/1742-6596/2035/1/012029.

[23] K. Eckle and J. Schmidt-Hieber, "A comparison of deep networks with ReLU activation function and linear spline-type methods," *Neural Networks*, vol. 110, pp. 232–242, Feb. 2019, doi: 10.1016/J.NEUNET.2018.11.005.

[24] H. Yang, J. Ni, J. Gao, Z. Han, and T. Luan, "A novel method for peanut variety identification and classification by Improved VGG16," *Sci Rep*, vol. 11, no. 1, Dec. 2021, doi: 10.1038/s41598-021-95240-y.

[25] Q. Wu, Y. Chen, and J. Meng, "Dcgan-based data augmentation for tomato leaf disease identification," *IEEE Access*, vol. 8, pp. 98716–98728, 2020, doi: 10.1109/ACCESS.2020.2997001.

[26] I. D. Raji, H. Bello-Salau, I. J. Umoh, A. J. Onumanyi, M. A. Adegboye, and A. T. Salawudeen, "Simple Deterministic Selection-Based Genetic Algorithm for Hyperparameter Tuning of Machine Learning Models," *Applied Sciences (Switzerland)*, vol. 12, no. 3, Feb. 2022, doi: 10.3390/app12031186.

[27] G. T. Reddy, M. P. K. Reddy, K. Lakshmanna, D. S. Rajput, R. Kaluri, and G. Srivastava, "Hybrid genetic algorithm and a fuzzy logic classifier for heart disease diagnosis," *Evol Intell*, vol. 13, no. 2, pp. 185–196, Jun. 2020, doi: 10.1007/s12065-019-00327-1.

[28] Institute of Electrical and Electronics Engineers and Manav Rachna International Institute of Research and Studies, *Proceedings of the International Conference on Machine Learning, Big Data, Cloud and Parallel Computing : trends, prespectives and prospects : COMITCON-2019 : 14th-16th February, 2019.*

[29] P. Naveen and B. Diwan, "Pre-trained VGG-16 with CNN architecture to classify X-Rays images into normal or pneumonia," in *2021 International Conference on Emerging Smart Computing and Informatics, ESCI 2021*, Institute of Electrical and Electronics Engineers Inc., Mar. 2021, pp. 102–105. doi: 10.1109/ESCI50559.2021.9396997.

[30] D. Albashish, R. Al-Sayyed, A. Abdullah, M. H. Ryalat, and N. Ahmad Almansour, "Deep CNN Model based on VGG16 for Breast Cancer Classification," in *2021 International Conference on Information Technology, ICIT 2021 - Proceedings*, Institute of Electrical and Electronics Engineers Inc., Jul. 2021, pp. 805–810. doi: 10.1109/ICIT52682.2021.9491631.

[31] A. Nasiri, A. Taheri-Garavand, and Y. D. Zhang, "Image-based deep learning automated sorting of date fruit," *Postharvest Biol Technol*, vol. 153, pp. 133–141, Jul. 2019, doi: 10.1016/j.postharvbio.2019.04.003.

[32] P. Ganakwar, "Convolutional Neural Network-VGG16 for Road Extraction from Remotely Sensed Images," *Int J Res Appl Sci Eng Technol*, vol. 8, no. 8, pp. 916–922, Aug. 2020, doi: 10.22214/ijraset.2020.30796.
[33] K. Behara, E. Bhero, and J. T. Agee, "Skin Lesion Synthesis and Classification Using an Improved DCGAN Classifier," *Diagnostics*, vol. 13, no. 16, Aug. 2023, doi: 10.3390/diagnostics13162635.
[34] M. Padala, D. Das, and S. Gujar, "Effect of Input Noise Dimension in GANs," Apr. 2020, [Online]. Available: http://arxiv.org/abs/2004.06882
[35] G. IEEE Engineering in Medicine and Biology Society. Annual International Conference (41st : 2019 : Berlin, IEEE Engineering in Medicine and Biology Society, and Institute of Electrical and Electronics Engineers, *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC) : Biomedical Engineering Ranging from Wellness to Intensive Care : 41st EMB Conference 2019 : July 23-27, Berlin.*

BIOGRAPHY OF AUTHORS

Kania Ardhani Putri was a student in Informatics Departement, School of Computing, Telkom Universit. She has been an active member of the management of Al-Fath, a religious organization on campus, for one period during his studies. She plays an active role in contributing to various social activities, character building, and other activities that support students' spiritual development. In addition, she has design and computer skills. She can be contacted at email: kaniaardhani @student.telkomuniversity.ac.id

Wikky Fawwaz Al Maki received the Dr. Eng. Degree in Integrated Science and Engineering from Ritsumeikan University, Japan, in 2009. He also received his B.Eng. in 2007. He is currently a lecturer at the School of Computing, Telkom University, Bandung, Indonesia. He is also the head of the multimedia research lab at Telkom University. His research areas are digital image processing, signal processing, stochastic systems, artificial intelligence, pattern recognition, and computer vision. He is affiliated with the Indonesia Data Scientist Society as a member. He has served as an invited reviewer for several international conferences and journals. He can be contacted at: wikkyfawwaz @telkomuniversity.ac.id