Available online at website: https://jurnal.iaii.or.id/index.php/RESTI

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 9 No. 6 (2025) 1405 - 1416 e-ISSN: 2580-0760

RadReader: An Enhanced AlexNet-Based GUI Application for Pneumonia Prediction in Thoracic X-Ray Images

Giri Wahyu Wiriasto[1], Ahdiat Aunul Hipzi[2] I Made Budi Suksmadana[3], Misbahuddin[4], Putu Aditya Wiguna[5], Indira Puteri Kinasih[6]

[1,2,3,4]Departement of Electrical Engineering, University of Mataram, Mataram, Indonesia
[5]Departement of Medical and Healthy Science, University of Mataram, Mataram City, Indonesia
[6]Faculty of Science, Universiti Brunei Darussalam, Bandar Sri Begawan City, Brunei Darussalam
[6]Departement of Math Tarbiyah, Universitas Islam Negeri Mataram, Mataram City, Indonesia

[1]giriwahyuwiriasto@unram.ac.id, [2]ahdiats542@email.com, [3]imbudisuks@unram.ac.id, [4]misbahuddin@unram.ac.id, [5]aditya.ku2004@gmail.com, [6]indiraputeri@ubd.ac.bn

Abstract

Recent advancements in radiology applications have led to user-friendly interfaces, improving pneumonia diagnosis by accurately differentiating between viral and bacterial pneumonia from thoracic X-rays. This approach enhances diagnostic precision and efficiency while offering intuitive real-time interaction for radiologists. This study aims to achieve two objectives: (i) developing a desktop-based radiology reader application, and (ii) modifying the alexNet architecture for classifying pneumonia based on thoracic X-ray datasets with the output encompassing pneumonia and normal cases. The desktop application assists radiologists in efficient image analysis and is developed using python–Tkinter. Integrate enhanced of AlexNet models which has been modified to better differentiate. The modified alexNet includes changes like adding max pooling in specific blocks and adjusting hidden layer neuron count. The dataset consists of 7442 images, with 4484 positive pneumonia and 2958 normal images obtained from the Mendeley websites. The enhanced alexNet (EAM) model achieves impressive results: 95.36% accuracy, 95.34% precision, 95.28% recall, and 95.31% F1-score for classifying bacterial pneumonia.

Keywords: *Enhanced-alexNet (EAM), Pneumonia classification, Radilogy application Reader, thoracic X-ray*

How to Cite: G. W. Wiriasto, A. A. Hipzi, I. M. B. Suksmadana, Misbahuddin, I. puteri Kinasih, and P. A. Wiguna, "RadReader: An Enhanced AlexNet-Based GUI Application for Pneumonia Prediction in Thoracic X-Ray Images", *J. RESTI (Rekayasa Sist. Teknol. Inf.)* , vol. 9, no. 6, pp. 1405 - 1416, Dec. 2025.
Permalink/DOI: https://doi.org/10.29207/resti.v9i6.7023

Received: July 30, 2025
Accepted: December 18, 2025
Available Online: December 28, 2025

This is an open-access article under the CC BY 4.0 License
Published by Ikatan Ahli Informatika Indonesia

1. Introduction

Pneumonia can be described as inflammation within the lung tissue resulting from infection agents, whether viral or bacterial [1]. Radiology, especially thoracic X-ray imaging, has become a crucial tool in the diagnosis and management of pneumonia [2]. Thoracic with detailed visualization of the lungs, X-rays enable clinicians to identify abnormal findings and evaluate how widely the lungs are affected. However, interpreting thoracic X-rays for pneumonia can be challenging, as the manifestations of the disease can vary [3], and subtle features can be easily overlooked, making this research important to conduct. With significant advancements in machine learning (ML), Applying machine learning methods can identify pneumonia with strong accuracy performance, including the use of deep learning algorithms [4]. Machine Learning is a technology that enables machines to learn and has a very broad scope, divided into several subfields that handle different learning tasks [5]. Deploying deep learning methods within ML for radiology could fundamentally reshape how clinical practice utilizes medical imaging [6]. As a type of deep learning architecture, CNNs are built to learn hierarchical patterns by extracting features automatically, making them a strong choice for pneumonia detection and classification on lung radiographs [7]. CNNs can assist radiologists in

interpreting complex images, reducing diagnostic errors, and speeding up the diagnosis process [8] using an automated approach to detect pneumonia from thoracic X-ray images [9]. The scope of this research includes the development and implementation of the RadReader application, a radiology reader system equipped with an enhanced AlexNet model for pneumonia classification by means of thoracic radiographic images. This work centers on leveraging deep learning techniques, particularly the AlexNet model, which has been modified to improve accuracy in detecting pneumonia. This application is designed to assist healthcare professionals in automatically analyzing thoracic X-ray images, aiming to expedite the diagnosis process and reduce the risk of interpretative errors. The scope of the research also includes evaluating the performance of the model in various clinical scenarios and analyzing how this application can be applied in everyday radiology practice. There are approximately 2,685 articles from 2012 to 2020 discussing medical record data (thoracic, neuroradiology, musculoskeletal, etc.), 251 of which discuss thoracic data, and only 103 articles specifically address thoracic X-ray image data. Furthermore, only 3 articles explore the use of the CNN-AlexNet model [2].

Due to the limited research using the AlexNet model, This work investigates how well the AlexNet model performs was evaluated by modifying the architecture of AlexNet [10], referred to as enhanced AlexNet, with the goal of achieving better results, which was then integrated into the developed interface application. The AlexNet model was selected as the reference for improving the created CNN model due to the limited studies applying the AlexNet model for classification in the medical field, especially related to radiology. The improved AlexNet model represents a significant advancement in applying CNN for pneumonia classification through thoracic X-ray images. This holds promise for enhancing diagnostic accuracy and expediting the diagnosis process. Additionally, it is highly important to develop interface applications for radiologists in their work. The main contribution of this research is the development of the RadReader application, specifically designed to assist radiologists in classifying pneumonia through thoracic X-ray images. This application utilizes a modified AlexNet model (enhanced AlexNet) to improve accuracy in detecting pneumonia. By leveraging the enhanced architecture, RadReader aims to provide more precise and faster analysis of X-ray images, thereby expediting the diagnosis process and reducing the risk of interpretative errors by healthcare professionals.

Several radiology applications have been developed on smartphones, utilizing standards such as DICOM (Digital Imaging and Communication in Medicine) [11]. These developments also support the creation of standardized reporting formats for radiologists [12] and include formalized assessment and evaluation systems [13]. Concurrently, the Imaging AI in Practice (IAIP) demonstrations were created to demonstrate how AI tools can create results, embed them within the workflow, and deliver them across the radiology pathway under simulated clinical conditions. These initiatives reinforce the need for semantic standards, interoperable systems, and orchestration profiles to support the widespread implementation of AI in clinical settings [14]. Building on these foundations, this study introduces RadReader: An Enhanced AlexNet-Based GUI Application for Pneumonia Prediction in Thoracic X-Ray Images. Unlike mobile-based solutions, RadReader was developed as a desktop-based interface using the Python Tkinter library, designed specifically to assist radiologists in managing thoracic X-ray datasets. This combination of optimized deep learning architecture with a practical, RadReader's potential as a decision-support tool in radiological practice.

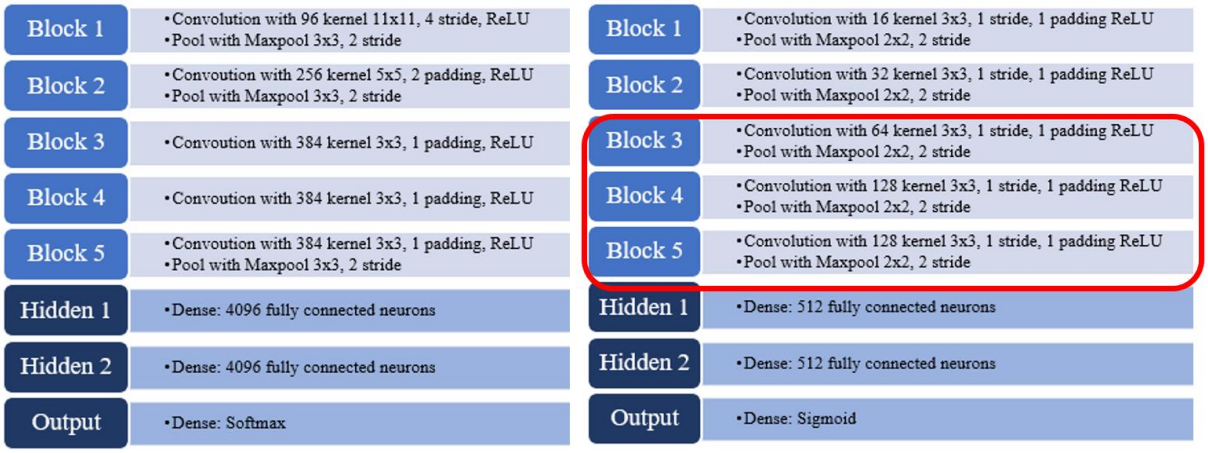
2. Methods

Designing a system for testing CNN architecture models involves several stages, starting from inputting digital image data, data visualization, data preprocessing, data augmentation, creating the CNN model, visualizing, training, visualizing training results, saving the model, loading the model, and testing the CNN model. In this research, the creation of a CNN model is conducted, where the architecture of the CNN model is based on the architecture of AlexNet, which is an evolution of the previous CNN model, LeNet-5 [15]. The study noted that the AlexNet architecture contains eight learned layers, made up of five convolutional layers and three fully connected layers [16].

The enhanced alexNet (EAM) model architecture developed The approach adopted in this study consists of two phases: extracting features and performing classification. Figure 1(a,b) illustrates a comparison between the AlexNet architecture and EAM model, where the red-highlighted sections indicate the parts of the architecture that have been modified along with their parameters. In AlexNet, the specific input image size is 224x224 pixels with RGB or 3 channels [17], while in the EAM, the specific input size is 200x200 pixels with Grayscale or 1 channel. In the first convolutional block, AlexNet first processes the input using 96 kernels of size 11×11, stride 4, and ReLU activation, then applies max pooling with a 3×3 window and stride 2.

The EAM model differs in its first block by using 16 kernels (3×3), stride 1, zero padding 1, and ReLU, followed by 2×2 max pooling with stride 2 and zero padding that preserves the spatial dimensions. In the second convolutional block, AlexNet uses 256 kernels of size 5×5 with stride 4, padding 2, and ReLU, then performs 3×3 max pooling with stride 2. EAM's second block uses 32 kernels (3×3) with stride 1, padding 1, and ReLU, followed by 2×2 max pooling with stride 2

under zero padding that preserves the spatial dimensions. In the subsequent blocks, AlexNet does not apply pooling to the third and fourth convolutions. However, EAM model still applies max pooling in the same manner as in the previous convolutional blocks, with modifications to convolution parameters. In this study, we used two classes, namely lung X-ray images with Pneumonia and Normal classes. In addition to integrating the enhanced AlexNet model for thoracic X-ray image classification, we also explored the capabilities of Tkinter to design an interface. Tkinter enabled the development of essential GUI components such as image upload, classification control buttons, and result display panels, which allow seamless interaction between the user and the underlying AlexNet-based model.



(a) (b)

Figure 1. (a). AlexNet architecture and (b) Enhanced-alexNet proposed

The hidden layers are modified in terms of the number of neurons. AlexNet employs 4096 neurons, while the EAM model uses 512 neurons. The output is also altered: AlexNet uses softmax activation [18], whereas the created EAM uses the sigmoid function.

2.1. EAM Network Architecture

EAM model, in the feature extraction phase, each convolution uses a 3x3 filter matrix with a stride of one step, one unit of padding, and ReLU activation. Subsequently, the end output obtained from the convolution layers is fed into a flatten layer, where the feature map undergoes reshaping.

This transforms the multidimensional array or matrix into a vector shape, which is then passed through the classification process using a Multi-Layer Perceptron (MLP) configured with a predefined neuron count for its hidden layers. Image classes are classified based on the neuron values across the hidden layers using the sigmoid activation function. In the feature extraction part as shown in Figure 2, there are five blocks of convolution layers, where each block consists of a convolutional layer and a pooling layer. The number of channels/kernels in each convolutional block is sequentially 16, 32, 64, 128, and 128. ReLU activation is applied to each convolution, as it has a simple calculation that speeds up training, reduces epochs, and uses fewer samples [19]. The ReLU activation function is expressed as Equation 1.

$$\sigma(x) = max(0, x) \quad (1)$$

Max pooling is adopted in the pooling layer to generate a compact output matrix of size $N_{out} \times N_{out}$ from an $N_{in} \times N_{in}$ input. This downsampling is implemented by splitting the $N_{in} \times N_{in}$ input square into N_{out}^2 pooling regions $(P_{i,j})$ [20], as described in Equation 2.

$$P_{(i,j)} \subset \{1,2,....,N_{in}\}^2 \text{ for each } (i,j) \in \{1,2,n,N_{in}\}^2;$$

$$Output_{i,j} = \frac{\max}{(k,l) \in P_{i,j}} Input_{k,l} \quad (2)$$

In the classification phase as shown in *F*igure 3, there are three hidden layers with a sequential number of neurons: 512, 512, and 1. Dropout Regularization is applied to each hidden layer, which is a technique to randomly activate some neurons in a hidden layer while deactivating others. This is done to prevent the model from overfitting during training [21]. The output uses the sigmoid function because there are only two categories [22]. Overall the architecture of the EAM model, along with the spatial dimensions, types, and parameters of each layer, can be observed in Table 1.

Table 1 provides the details of the EAM model convolution layers from Figures 2 and 3. It's evident that the generated EAM model has a total of 3,719,681 parameters, with an increased number of filters in each layer. The comparison of parameter counts in convolution blocks indicates that the fifth convolution block has significantly more parameters than the other convolution blocks, suggesting its ability to capture more complex features. The consistent use of 3x3 filter sizes across all layers helps the model detect more

common features. Based on Figure 2, this architecture is a CNN-type feature extractor that processes a 200×200×1 grayscale input image through several repeating blocks of 3×3 convolution followed by a ReLU activation, and then 2×2 max-pooling for downsampling. Each pooling stage progressively reduces the spatial resolution, while the number of feature channels increases to capture more complex patterns. With this scheme, the network extracts features from low-level cues to higher-level representations (structural patterns), producing a final feature map which can then be forwarded to a fully connected classification component.

Figure 2 represents the feature extraction stage from chest X-ray images, where the input image is processed through a series of convolution and pooling layers to produce a feature map with smaller spatial dimensions but richer informational content. Figure 3 illustrates the classification stage after the CNN features have been obtained: the feature map is flattened into a 1D vector, fed into a 512-neuron hidden layer, and finally passed to a sigmoid-activated output layer to generate predictions for the Normal versus Pneumonia classes. Thus, the second figure is not a separate architecture but the classifier head of the CNN, completing an end-to-end pipeline from feature extraction to class prediction.

Table 1. The Enhanced Alexnet (EAM) Model Layer Convolution

Layer	Type	Channel	Filter	Stride	Padding	Output	Pamars
Input	Data layer	1	-	-	-	200x200	-
Convolution Layer	Convolution+ReLU	16	3x3	1	1	200x200	160
Pooling layer	Max_pooling	16	2x2	2	1	100x100	-
Convolution Layer	Convolution+ReLU	32	3x3	1	1	100x100	4.640
Pooling layer	Max_pooling	32	2x2	2	1	50x50	-
Convolution Layer	Convolution+ReLU	64	3x3	1	1	50x50	18.496
Pooling layer	Max_pooling	64	2x2	2	1	25x25	-
Convolution Layer	Convolution+ReLU	128	3x3	1	1	25x25	73.856
Pooling layer	Max_pooling	128	2x2	2	1	13x13	-
Convolution Layer	Convolution+ReLU	128	3x3	1	1	13x13	147.584
Pooling layer	Max_pooling	128	2x2	2	1	7x7	-
Flatten	Dropout	-	0,2	-	-	12544	-
Fully Connected layer	Dense + ReLU + Dropout	512	0,2	-	-	512	32.117.76
Fully Connected layer	Dense + ReLU + Dropout	512	0,2	-	-	1	262.656
Fully Connected layer	Dense Sigmoid	1	-	-	-	1	513
Total Parameter							3.719.681
Trained Parameter							3.719.681
Untrained Parameter							-

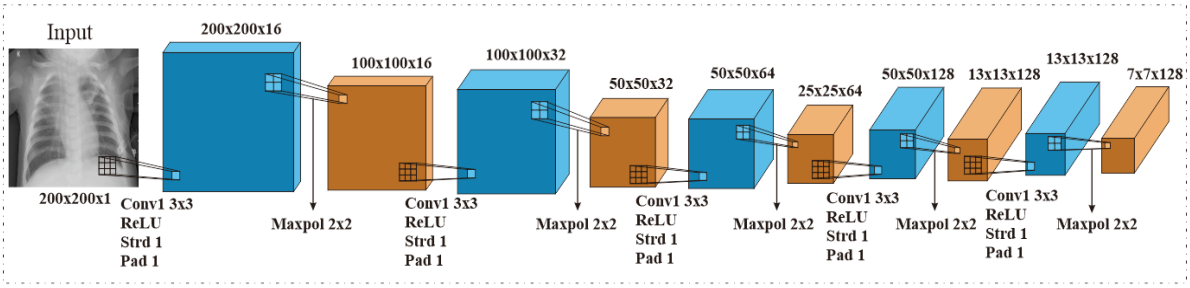


Figure 2. EAM Feature Extraction Architecture

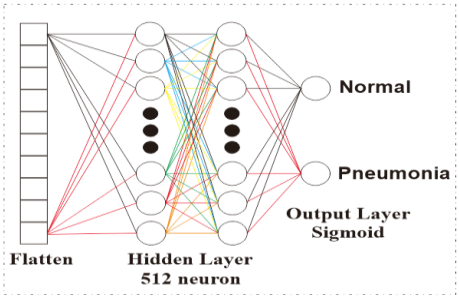


Figure 3. EAM Classification

3.2. Dataset and Evaluation Methods

In this research, secondary data are utilized in the form of a two-dimensional image dataset. The dataset utilized originates from two sources obtained from the Mendeley websites. The initial dataset includes antero-posterior lung X-ray images obtained from a retrospective cohort of children between one and five years old treated at Guangzhou's Women and Children's Medical Center [23]. The second dataset is a unified repository of fifteen other datasets collected by four institutions: the Indian Institute of Science, PES University, M.S Ramaiah Institute of Technology, and Concordia University [24].

The utilized dataset is a combination of two datasets, comprising a total of 7,442 lung X-ray images. Among these, 4,484 images fall under the category of Pneumonia, while 2,958 images belong to the Normal

category. The image sizes within the dataset vary, ranging from 710 x 430 pixels to 2720 x 2460 pixels. This dataset is divided into two parts: the training data and the testing data. The training data consists of 7,242 images, while the testing data comprises 200 images. The training data is further divided into two subsets: the actual training data and the validation data, with an 8:2 ratio.

Data preprocessing is carried out on all images in the dataset to enhance image quality by manipulating or replacing missing values and balancing images in an imbalanced dataset [25]. Three preprocessing techniques applied in this study are grayscale conversion, rescaling, and resizing. Data augmentation provides the simplest solution to increase the dataset size by creating variations in the used dataset. For instance, labels for handwritten characters should not change with minor location shifts, rotations, slight shearing, intensity changes, line thickness alterations and size variations [26].

There are five methods used to create variations in the dataset as follows: a) Rotation Range: This involves randomly rotating image data. In this study, the image rotation range is 30°; b) Zoom Range: This transformation involves randomly enlarging the size of image shapes. The zoom range used in this study is 20% of the original image data; c) Width Shift Range: This process involves randomly shifting the original image data horizontally. The shift applied in this study is 10%; d) Height Shift Range: This entails randomly shifting the original image data vertically, either upwards or downwards. The shift used in this study is 10%; e) Horizontal Flip: This transformation involves flipping image data horizontally at random. Preprocessing is a stage of image data processing where these augmentation techniques are applied to create variations in the dataset, improving its diversity and quality.

In the training phase of the EAM model, the training dataset consisting of 3,624 pneumonia case images and 2,240 normal case images is used to train the EAM model. To train this EAM model, the loss function used is Binary Crossentropy, and the optimizer is the *Adam Optimizer*. Other predetermined parameters for the training process include a Learning Rate of '0.001', batch size of 32, Max Epoch of 50, and *'patience'* set to '2'. Additionally, several methods are applied to assist in the training process include tensorflow [27], as follows:

In the adjustment of the learning rate value, the *'ReduceLROnPlateau'* function is employed to decrease the learning rate value during the training process, aiming to prevent overfitting. The parameters configured for this method are outlined in Table 2.

Table 2. Hyperparameter Function Reducelronplateau'

Monitor	*Patience*	*Verbose*	*Factor*	*min_lr*
val_loss	2	1	0,5	0,00001

Based on the set parameters, this function will decrease the learning rate value from its previous value, based on the reference of the *'val_loss'* or validation loss specified in the *'Monitor'* parameter. The learning rate adjustment occurs when the validation loss obtained for the next '2' epochs does not show any improvement compared to the previous validation loss. This behavior is controlled by the *'patience'* parameter value provided. Notifications regarding these changes can be displayed by setting the *'verbose'* parameter to a value of '1'. The next learning rate value to be used can be determined using the *'factor'* parameter with Equation 3.

$$lr_i = lr_{i-1} \times factor \quad (3)$$

From the equation provided with an initial *learning rate (lr)*, $lr_1 = 0.001$, the subsequent learning rate values can be calculated as follows: $lr_2 = 0.0005$, $lr_3 = 0.00025$, $lr_4 = 0.000125$, $lr_5 = 0.0000625$, $lr_6 = 0.00003125$, and $lr_7 = 0.000015625$. The parameter *'min_lr'* is set to a value of *'0.000001'* to establish the minimum limit for the learning rate change, which will not go below '0.000001'.

In the Early stopping stage, the training process stops based on several preconfigured parameters. These parameters include *'monitor'* to specify the value used as a reference for stopping, *'patience'* to determine the number of epochs, *'verbose'* to enable notification of the early stopping action, and *'restore best weights'* to restore the best weights based on the best reference value obtained. The parameter settings can be observed in Table 3.

The reference value used for stopping the training process is the *'val_loss'* value specified in the *'monitor'* parameter. The *'patience'* parameter is set to '5' to halt the training process if the *'val_loss'* value does not improve for five epochs. The *'verbose'* parameter is set to '1' to provide notifications when *'EarlyStopping'* function is executed. The *'restore best weights'* parameter is set to *'True'* to restore the best weights obtained during the training process for storage.

Table 3. Hyperparameter Function *'Earlystopping'*

Monitor	*Patience*	*Verbose*	*restore best weights*
val_loss	5	1	*True*

In the Checkpoin stage, the 'ModelCheckpoint' function has five parameters to control how the trained model will be saved. The settings for each parameter can be seen in Table 4.

Table 4. Hyperparameter Function *'ModelCheckpoint'*

Name	Monitor	Mode	Ver-bose	Save best only
Xray_model.h5	val_loss	Min	1	True

In Table 4, the five preconfigured parameters will save the trained model with the name *'Xray_model.h5'* in hierarchical data format (HDF) or *'.h5'* format specified in the *'name'* parameter. The *'mode'* parameter

set to *'min'* is used when the *'save best only'* parameter is activated 'True'. It will save the training model with the smallest reference value *'val_loss'* specified in the *'monitor'* parameter.

Each channel used during the convolution of an input has different weights and captures distinct input features as well. Furthermore, for the convolutional process within each block, the procedure remains the same, guided by a general equation based on the input, hyperparameters, parameters, and activation functions predetermined for each block in Table 5 [28]:

Table 5. Variable of Equation

Variable	Description
H	Kernel height with order row-i
W	Kernel width with order column-j;
D	Dimension or kernel value until dimension to-d;
H^l	Input height with order row-i^l;
W^l	Input width with order column-j^l;
D^l	Dimension or atau input channel with dimension to-d^l
D^{l+1}	Dimension or input channel convolution result;
d^l	Dimension order or channel convolution;
X^l	input value
f	The weight value in the kernel
H^{l+1}	High yield convolution order row-i^{l+1};
W^{l+1}	Width yield convolution order column-j^{l+1};
b_d	bias value from channel to-d;
$Y_{H^{l+1},W^{l+1},D^{l+1}}$	Feature map from convolution result ;
j	column order from featur map;
i	row order from featur map;

$$y_{(i^{(l+1)},j^{(l+1)},d)} = \sum_{i=0}^{H}\sum_{j=0}^{W}\sum_{d^i=0}^{D^l} f_{(i,j,d^l,d)} \times x^l_{(i^{(l+1)}+i,j^{(l+1)}+j,d^l)} + b_d \quad (4)$$

Equation 4 expresses the general multi-channel convolution operation in a convolutional layer, i.e., how each output feature-map element $Y_{H_{l+1},W_{l+1},D_{l+1}}$ is computed from the input X_l. For each output channel d(with $0 \le d < D = D_{l+1}$), the output value is obtained by sliding an $H \times W$ kernel over a local region of the input and performing element-wise multiplication between the kernel weights f and the corresponding input patch across spatial indices (i, j), then summing the results over all input channels (d_l), and finally adding the bias term b_d. This equation provides the general convolution formulation used in each convolutional block before being simplified to the single-channel case in Equation 5.

$$y_{(i^{(l+1)},j^{(l+1)},d)} = \sum_{i=0}^{H}\sum_{j=0}^{W} f_{(i,j,d)} \times x_{(i^{(l+1)}+i,j^{(l+1)}+j)} + b_d \quad (5)$$

Then, the result of Equation 5 is passed through the ReLU function using Equation 6:

$$Y_{(i,j,d)} = max\{0; y_{(i^{(l+1)},j^{(l+1)},d)}\} \quad (6)$$

With $0 \le i^{l+1} \le H^l + 1 = H^{l+1}$, $0 \le j^{l+1} \le W^l + 1 = W^{l+1}$, $0 \le d < D = D^{l+1}$ hence the result of the feature map from the convolutional layer is $Y_{H^{l+1},W^{l+1},D^{l+1}}$ where H^{l+1} for the height of the feature map, W^{l+1} for the width of the *featur map*, dan D^{l+1} for the dimension or number of channels in the feature map. Here is an explanation of the convolution process for each block in the EAM model (Table1) :

Convolutional layer *'conv_1'* serves as the first convolutional operation, tasked with extracting features from the input image of size 200x200 pixels with 1 channel (grayscale). In this *'conv_1'* layer, there are 16 kernels with a size of 3x3, a stride of '1', and zero-padding of '1'.

The ReLU activation function is applied, causing the output of *'conv_1'* to fall within the range of '0' to the maximum value. Here, the maximum value represents the largest value obtained after applying the input values to the ReLU function. This configuration is based on the parameters and hyperparameters that were previously determined in conv_1, where H=W=3, $H^l = W^l = 200$, and D= D^{l+1}=16, with $0 \le i < H = \{0, 1, 2\}$, $0 \le j < W = \{0, 1, 2\}$, and $0 \le d < D = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$. With $0 \le i^{l+1} \le H^l + 1 = H^{l+1}$, $0 \le j^{l+1} \le W^l + 1 = W^{l+1}$, $0 \le d < D = D^{l+1}$. Specifically for the conv_1 layer, because the input only has one channel (D^l=1), Hence, the parameter and hyperparameter values are inserted into Equation 5 to obtain the convolution calculation result, as shown in Equation 7:

$$y_{i^{l+1},\,j^{l+1},d} = \sum_{i=0}^{3}\sum_{j=0}^{3} f_{i,j,d} \times x_{i^{l+1}+i,\,j^{l+1}+j} + b_d \quad (7)$$

then the convolution result will be activated using the ReLU activation function, as shown in Equation 8:

$$Y_{i,j,d} = max\{0; y_{i^{l+1},\,j^{l+1},d}\} \quad (8)$$

The feature maps resulting from the convolution process in *'conv_1'* become the input for the Maxpooling layer '*Maxpool_1'*. '*Maxpool_1'* employs *maxpooling*, which involves selecting the highest value from the input image in the pooling layer. '*Maxpool_1'* is configured with specific hyperparameters: a 2x2 size, a stride of '2', and padding of '1'. This layer receives input with a size of 200x200 and 16 feature maps, which, after pooling, generates 16 feature maps each with a size of 100x100.

Convolutional Layer *'conv_2'* represents the second convolution process aimed at extracting features from the input image obtained after the 100x100-pixel *max-pooling* of *'Maxpool_1'* with 16 feature maps. In this *'conv_2'* layer, there are 32 kernels with a size of 3x3, a stride of '1', and zero-padding of '1'.

The ReLU activation function is applied, resulting in the output of *'conv_2'* falling within the range of '0' to the maximum value. Here, the maximum value represents the largest value from the input values passed through the ReLU function. This configuration is based on the parameters and hyperparameters that were previously determined in *'conv_2'*, where H==W=3, $H^l = W^l = 100$, $D^l = 16$ and D= D^{l+1}=32, with $0 \le i < H = \{0, 1, 2\}$, $0 \le j < W = \{0, 1, 2\}$, dan $0 \le d < D = \{0, 1, 2, ..., 31\}$. The parameter and hyperparameter

values are inserted into Equation 4 to obtain the convolution calculation result, as shown in Equation 9.

$$:y_{i^{l+1},\,j^{l+1},d} = \sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{d^i=0}^{16} f_{i,j,d^l,d} \times x^{l}_{i^{l+1}+i,\,j^{l+1}+j,d^l} + b_d \quad (9)$$

Then, the convolution result will be activated using the ReLU activation function, as shown in Equation 10.

$$Y_{i,j,d} = max\{0; y_{i^{l+1},\,j^{l+1},d}\} \quad (10)$$

The feature maps resulting from the convolution process in *'conv_2'* become the input for the *maxpooling* layer *'Maxpool_2'*. *'Maxpool_2'* employs *maxpooling*, which involves selecting the highest value from the input image in the pooling layer. *'Maxpool_2'* is configured with specific hyperparameters: a 2x2 size, a stride of '2', and padding of '1'. This layer receives input with a size of 100x100 and 32 feature maps, which, after pooling, generates 32 feature maps each with a size of 50x50.

Convolutional layer *'conv_3'* represents the third convolutional process, intended to extract features from the input image received after the 50x50-pixel *maxpooling* of *'Maxpool_2'* with 32 feature maps. In *'conv_3'*, there are 64 kernels with a size of 3x3, a stride of '1', and zero-padding of '1'.

ReLU activation is applied, which means that the output of *'conv_3'* falls within the range of '0' to the maximum value. This implies that all negative values (values < 0) are changed to '0'. This configuration is based on the parameters and hyperparameters that were previously determined in *'conv_2'*, where H==W=3, $H^l = W^l = 50$, $D^l = 32$ and $D= D^{l+1}=64$, with $0 \le i < H=\{0, 1, 2\}$, $0 \le j < W=\{0, 1, 2\}$, and $0 \le d < D = \{0, 1, 2, ..., 63\}$.

The parameter and hyperparameter values are inserted into Equation 4 to obtain the convolution calculation result, as shown in Equation 11.

$$y_{i^{l+1},\,j^{l+1},d} = \sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{d^i=0}^{32} f_{i,j,d^l,d} \times x^{l}_{i^{l+1}+i,\,j^{l+1}+j,d^l} + b_d \quad (11)$$

Then, the convolution result will be activated using the ReLU, as shown in Equation 12.

$$Y_{i,j,d} = max\{0; y_{i^{l+1},\,j^{l+1},d}\} \quad (12)$$

In the *'Maxpool_3'* layer, *maxpooling* is applied, involving the selection of the highest value from the input image in the pooling layer. *'Maxpool_3'* is configured with specific hyperparameters: a 2x2 size, a stride of '2', and padding of '1'. This layer receives input from the convolution result of *'conv_3'* with a size of 50x50 and 64 feature maps. It generates an output of 64 feature maps, each with a size of 25x25.

The Convolutional Layer *'conv_4'* serves as the fourth convolutional process aimed at extracting input-image feature representations received after the 25x25-pixel *maxpooling* of *'Maxpool_3'* with 64 feature maps. In this *'conv_4'* layer, there are 128 kernels with a size of 3x3, a stride of '1', and zero-padding of '1'.

After convolution, the resulting feature map is passed through a ReLU, causing the output of *'conv_4'* to fall within the range of '0' to the maximum value. This configuration is based on the parameters and hyperparameters that were previously determined in *'conv_4'*, where H==W=3, $H^l = W^l = 25$, $D^l = 64$ and $D= D^{l+1}=128$, with $0 \le i < H=\{0, 1, 2\}$, $0 \le j < W=\{0, 1, 2\}$, and $0 \le d < D = \{0, 1, 2, ..., 127\}$. The parameter and hyperparameter values are inserted into equation (4) to obtain the convolution calculation result, as shown in Equation 13.

$$y_{i^{l+1},\,j^{l+1},d} = \sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{d^i=0}^{64} f_{i,j,d^l,d} \times x^{l}_{i^{l+1}+i,\,j^{l+1}+j,d^l} + b_d \quad (13)$$

Then the convolution result will be activated using the ReLU activation function, as shown in Equation 14.

$$Y_{i,j,d} = max\{0; y_{i^{l+1},\,j^{l+1},d}\} \quad (14)$$

In the *'Maxpool_4'* layer, *maxpooling* is also applied, with hyperparameters set as a spatial size of 2x2, a stride of '2', and padding of '1'. This layer receives input from the convolutional result of *'conv_4'* with a size of 25x25 and 128 feature maps. It produces an output of 128 feature maps, each with a size of 13x13.

The Convolutional layer *'conv_5'* represents the fifth convolution process, utilizing the same number of kernels as *'conv_4'*, which is 128 kernels. *'Conv_5'* is characterized by hyperparameters: a 3x3 size, a stride of '1', zero-padding of '1', and ReLU activation. This layer receives input images from the 13x13-pixel *maxpooling* result of *'Maxpool_4'* with 128 feature maps, generating an output of 128 feature maps each with a size of 13x13 pixels.

Based on the given parameters and hyperparameters, where H==W=3, $H^l = W^l = 13$, $D^l = 128$ dan $D= D^{l+1}=128$, with $0 \le i < H=\{0, 1, 2\}$, $0 \le j < W=\{0, 1, 2\}$, and $0 \le d < D = \{0, 1, 2, ..., 127\}$. The parameter and hyperparameter values are input into Equation 4 to obtain the convolution calculation result, as shown in Equation 15.

$$y_{i^{l+1},\,j^{l+1},d} = \sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{d^i=0}^{128} f_{i,j,d^l,d} \times x^{l}_{i^{l+1}+i,\,j^{l+1}+j,d^l} + b_d \quad (15)$$

Then, the convolution result will be activated using the ReLU, as stated in Equation 16.

$$Y_{i,j,d} = max\{0; y_{i^{l+1},\,j^{l+1},d}\} \quad (16)$$

The feature maps resulting from the convolution process in *'conv_5'* become the input for the Maxpooling layer *'Maxpool_5'*. *'Maxpool_5'* employs Max pooling, which consists of taking the highest value from the input image in the pooling layer. *'Maxpool_5'* is configured with specific hyperparameters: a 2x2 size,

a stride of '2', and padding of '1'. This layer receives an input of size 13x13 with 128 feature maps, which, after pooling, generates 128 feature maps each with a size of 7x7.

The outputs produced by the convolution operation used in each block are in the form of feature map weights, which can be seen in Table 6 by taking a sample of kernel weights in each convolution block.

3. Results and Discussions

Table 6. The Outcome of the Convolution Process in a Set of Feature Map Weights, Obtained by Sampling One Kernel Weight from Each Convolution Block

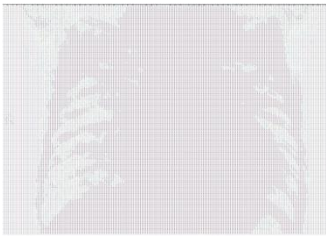
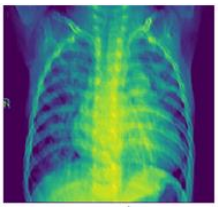
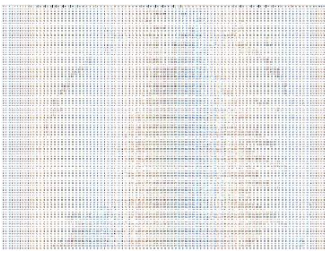
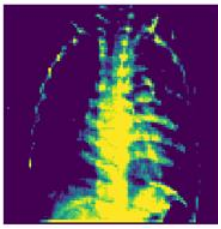
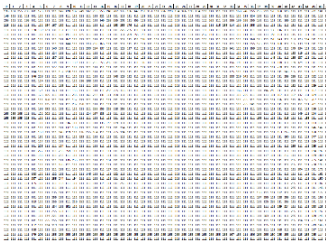
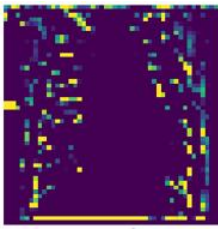
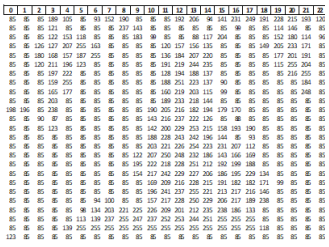
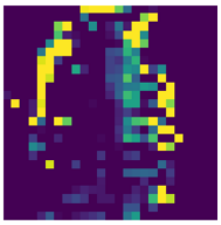
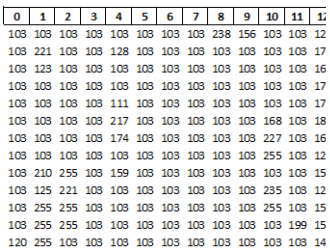
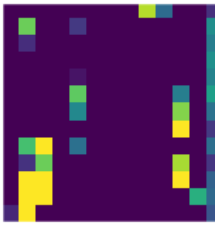
Block *(a)*	Layer name *(b)*	Channel/kernel *(c)*	Output data *(d)*	Channel color reference *(e)*	Output with color reference *(f)*
1	Conv_1	0,1542 ; 0,0027; -0,0216 0,1330 ; -0,0170; -0,1173 0,1209 ; 0,1539; 0,0287			
2	Conv_2	0,0411 ; -0,0636; 0,1127 0,1115 ; 0,1257; -0,0766 0,1417 ; 0,0818; 0,0928			
3	Conv_3	-0,1252; -0,0692; 3,26E-05 -0,0294; -0,0449; -0,0172 -0,0174; -0,0487; -0,0043			
4	Conv_4	0,0434 ; 0,1226; 0,0918 0,0185 ; 0,0555; 0,0627 0,0774 ; 0,0544; 0,0635			
5	Conv_5	-0,0033; 0,0642; 0,0739 -0,0355; -0,0484; -0,0326 0,0568 ; -0,0265; 0,0071			

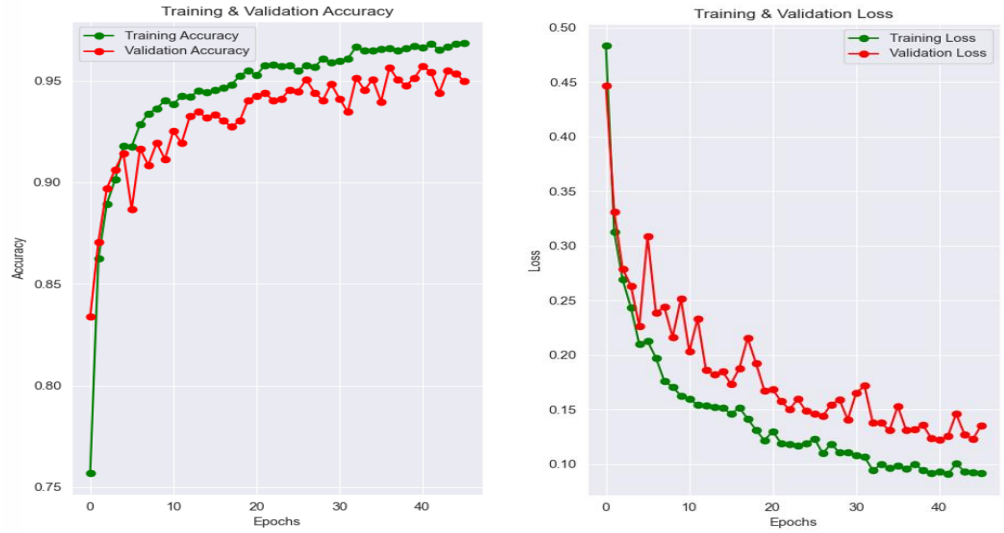
Table 6 displays the convolution results by taking a single convolution sample from each block. These convolution results take the form of feature map weights that have been activated using the ReLU function.

The weight values in these feature maps range from '0' to 254, where the weight value becomes '0' if there are no characteristics in the feature map that match the ex-

tracted kernel characteristics. Table 6 *(a)* shows the sequence of convolution blocks, with each block containing convolution layers named according to Table 6*(b)*. Table 6*(c)* presents the weight values of the channels/kernels taken as samples. Table 6*(d)* displays the outcome obtained from the convolution step in the form of feature maps. Furthermore, Table 6*(e)* represents the color form of weight values in the used kernels. Finally, Table 6*(f)* illustrates the color representation of weight values in the output feature maps found in Table 6*(d)*.

3.1. Training result and validation

The results of the enhanced EAM training are illustrated through plots of accuracy (Figure 4(a)) and loss (Figure 4(b)).



(a) (b)

Figure 4. (a). Comparison between the training accuracy and the validation accuracy values; (b) Comparison between the training loss and the validation loss values

Figure 4(a) depicts the progression of training and validation accuracy over 50 epochs. Both curves rise steadily and converge at approximately 95%, suggesting that the EAM learned effectively with no substantial gap between training and validation performance. The strong overlap of these curves also indicates good generalization to unseen data, supporting reliable pneumonia classification from thoracic X-ray images. In Figure 4(b), the training and validation loss curves show a gradual decline, with training loss settling near 0.09 and validation loss near 0.12 by the end of training. The small final difference between the losses (0.0291) further indicates minimal overfitting, implying that the model captured meaningful features rather than merely memorizing the training set. Overall, the consistent convergence of accuracy and loss highlights the contribution of the EAM adjustments—such as reducing kernel counts and adding pooling layers—to improved classification performance. When the training process is complete, the model is saved and then validated on the validation dataset to assess whether it performs well and meets the desired level of optimality. The results appear in Table 7. From the testing results using validation data, the values of precision, recall, F1-score, MCC, along with overall accuracy can be calculated using Equations 17 to 21.

$$Precision = \frac{TP}{TP+FP} \quad (17)$$

$$Recall = \frac{TP}{TP+FN} \quad (18)$$

$$F1-score = \frac{2\times TP}{2\times TP+FN+FP} \quad (19)$$

$$MCC = \frac{TP\times TN-FP\times FN}{\sqrt{(TP+FP)\times(TP+FN)\times(TN+FP)\times(TN+FN)}} \quad (20)$$

$$Accuracy = \frac{TP+TN}{TP+TN+FN+FP}; \quad (21)$$

However, the training process stopped at epoch 46 and saved the best weights obtained at epoch 41 with a learning rate of 1.5625e-5. The training phase achieved 96.64% accuracy on the training set and 95.72% on the validation set. Regarding the loss values, the training process resulted in a training loss of 0.0929 and a validation loss of 0.1220. Figure. 4(a-b) indicate that EAM did not experience overfitting during the training process. Overfitting occurs when a model does not generalize well to unseen test data relative to its performance on the training set. It is commonly assessed by comparing the training and validation curves; in this case, the accuracy and loss values for both sets remain relatively close. By the end of training, the difference in accuracy is only 0.92%, and the difference in loss is only 0.0291.

Table 8 summarizes the test results on 1,378 validation images. Precision was 95.15% for the normal class and 95.53% for the pneumonia class; recall was 94.53% and 96.03%, respectively; and F1-scores were 94.84% (normal) and 95.78% (pneumonia).

Table 7. Confusion Matrix

		Actual	
		Normal	Pneumonia
Predict	Normal	588	34
	Pneumonia	30	726
Total		612	760

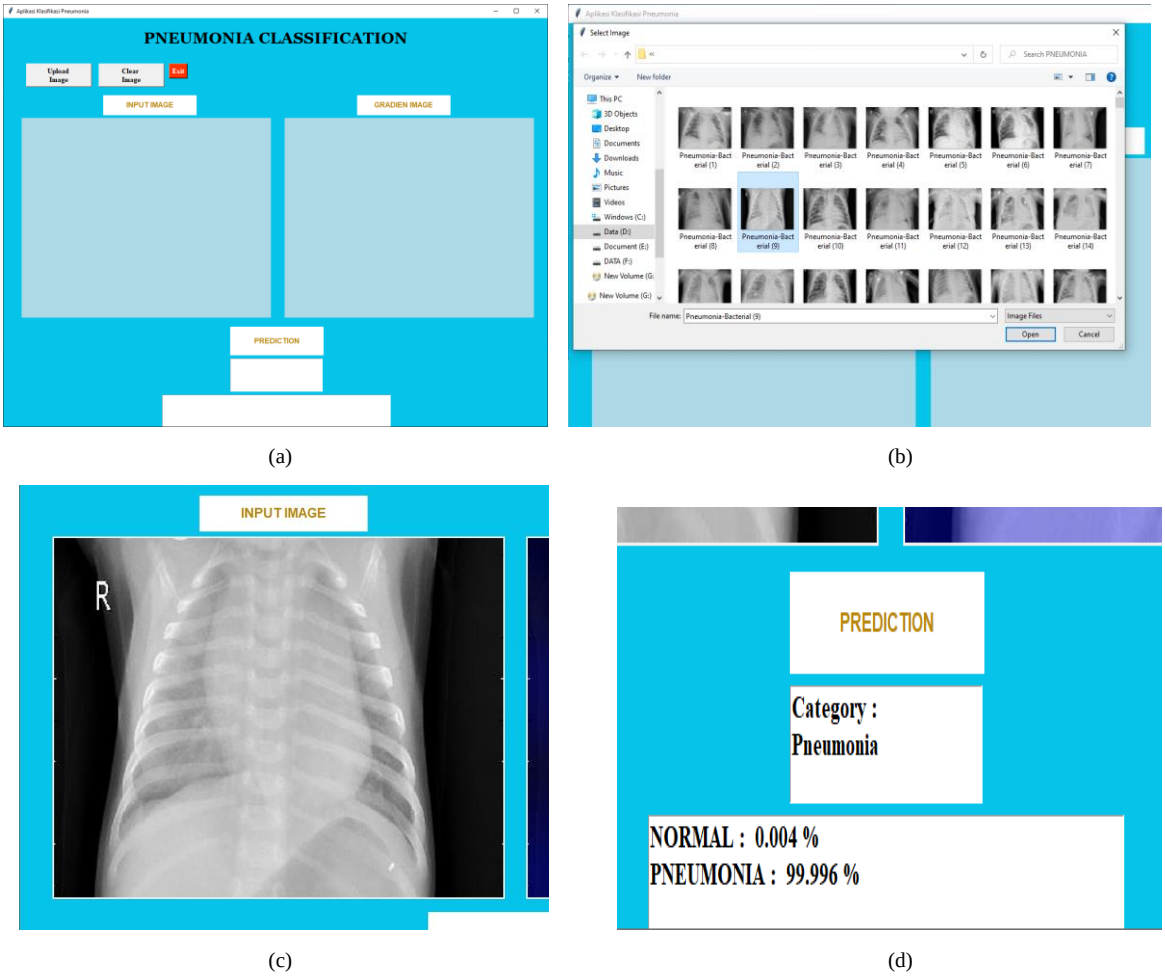
An MCC of 0.91 indicates strong agreement between predictions and true labels in distinguishing normal and pneumonia classes. The model also achieved 95.36% accuracy, and all evaluation metrics exceeded 90%, demonstrating excellent classification performance. The slight difference of 0.36% in accuracy between training validation and model testing using the validation data indicates that the model is working optimally without overfitting. With these outlined achievements, the model can be considered a contribution to the development of ML systems for pneumonia classification using the EAM model.

Table 8. Calculation Results For Precision, Recall, F1-Score, and Accuracy

Class	Result				
	Precision (%)	Recall (%)	F1-score (%)	MCC	Accuracy (%)
Normal	95,15	4,53	94,84	0,91	95,36
Pneumonia	95,53	96,03	95,78		

3.2. Application

Implementation of EAM Model in Radiology Reader Application. The saved model, after training, is tested again using a Graphical User Interface (GUI) desktop to assess whether the stored model can be implemented in the form of an application. The desktop GUI interface is depicted in Figure 5(a-d).



(a) (b)

(c) (d)

Figure 5. Displays the GUI application interface

In Figure 5(a-d), the images represent features present in the application. Figure 5(a) Rad-reader GUI Apps home page; Figure 5(b) depicts the *'Upload Image'* feature, allowing users to select an image from their directory and input it into the application for classification. (c) The *'Input Image'* section displays the uploaded image within the application. Figure 5 (d), the prediction feature is divided into two parts. The first part displays the predicted category, while the second part indicates the prediction confidence level for each

class in the form of percentages. The RadReader application was tested using the black-box testing technique, where we evaluated whether the user interface functioned according to the intended actions. This included opening or loading image data, then pressing the prediction button to process the test data. The model then worked until the classification result appeared on the right panel of the RadReader GUI apps. The research results indicate that the EAM model is highly effective and superior when compared to the model without enhancement. The EAM model is capable of achieving high accuracy in training and validation while maintaining low loss values. EAM can be successfully implemented in the developed new form desktop-based user interface application, namely as the Radiology Application Reader, or *RadReader*. The classification results obtained from this model can serve as a diagnostic reference for doctors, assisting in patient evaluation and providing reliable confirmation of the presence of pneumonia in lung X-ray images.

4. Conclusions

The results obtained in this study include: (i) The research results demonstrate that the Enhanced-AlexNet (EAM) model can be implemented for use in the desktop-based user interface application known as Radiology Application Reader, which has been developed. The classification results can serve as a diagnostic reference for doctors, aiding in patient evaluation and confirming the presence of pneumonia. (ii) The classification results using the EAM model with validation data, consisting of 1,378 images, yielded a precision of 95.15%, Recall of 94.53%, and F1-score of 94.84% for the 'normal' class. For the 'positive-pneumonia' class, Precision was 95.53%, Recall was 96.03%, and F1-score was 95.78%. MCC was measured at 0.91, and the model's Accuracy was 95.36%. The best-performing EAM model was obtained during the training process at epoch 41, achieving a training accuracy of 96.64%, a validating accuracy of 95.72%, a training loss of 0.0929, and a validating loss of 0.1220. The results achieved by this EAM are better 2% compared to the non-enhanced AlexNet, where only an accuracy of 89.11% and a loss of 0.261 were obtained. With such high accuracy values and low loss values, the EAM model can be confidently stated to be highly capable of classifying pneumonia in lung X-ray images. Future work, the current classification should be extended by adding new classes to distinguish between viral and bacterial pneumonia. This multi-class classification will provide more clinically relevant insights, enabling doctors not only to confirm the presence of pneumonia but also to understand its etiology, thereby supporting faster and more targeted treatment decisions.

Acknowledgments

This work was supported by– Universitas Mataram. The data used is open to the public and can be found on Mendeley. The dataset is provided at, https://data.mendeley.com/datasets/9xkhgts2s6/3.

References

[1] D. Lokida, H. Farida, R. Triasih, Y. Mardian, H. Kosasih, A.M. Naysilla, et al., “Epidemiology of community-acquired pneumonia among hospitalised children inIndonesia: a multi-centre, prospective study,” BMJ Open, vol. 12, no. 6, p. e057957, June 2022, doi: 10.1136/bmjopen-2021-057957.

[2] L. Wang, H. Wang, Y. Huang, B. Yan, Z. Chang, Z. Liu, et al., “Trends in the application of deep learning networks in medical image analysis: Evolution between 2012 and 2020,” European Journal of Radiology, vol. 146, article no. 110069, January 2022, doi: 10.1016/j.ejrad.2021.110069.

[3] R. Kundu, R. Das, Z. W. Geem, G. T. Han, and R. Sarkar, “Pneumonia detection in chest X-ray images using an ensemble of deep learning models,” PLoS ONE, vol. 16, no. 9, p. e0256630, September 2021, doi: 10.1371/journal.pone.0256630.

[4] Y. Li, Z. Zhang, C. Dai, Q. Dong, and S. Badrigilan, "Accuracy of Deep Learning for Automated Detection of Pneumonia Using Chest X-Ray Images: A Systematic Review and Meta-Analysis," Computers in Biology and Medicine, vol. 123, article no. 103898, August 2020, doi: 10.1016/j.compbiomed.2020.103898.

[5] S. Shalev-Shwartz and S. Ben-David, Understanding Machine Learning: From Theory to Algorithms,1st ed., Cambridge University Press, 2014, doi: 10.1017/cbo9781107298019.

[6] G. Choy, O. Khalilzadeh, M. Michalski, S. Do, A. E. Samir, O. S. Pianykh, et al., “Current Applications and Future Impact of Machine Learning in Radiology,”Radiology, vol. 288, no. 2, pp. 318–328, August 2018, doi: 10.1148/radiol.2018171820.

[7] S. Samir, E. Emary, K. El-Sayed, and H. Onsi, “Optimization of a Pre-Trained AlexNet Model forDetecting and Localizing Image Forgeries,” Information, vol. 11, no. 5, article no. 275, May 2020, doi: 10.3390/info11050275.

[8] F. Zhang, “Application of machine learning in CT images and X-rays of COVID-19 pneumonia,” Medicine, vol. 100, no. 36, article no. e26855, September 2021, doi: 10.1097/md.0000000000026855.

[9] R. Alsharif, Y. Al-Issa, A. M. Alqudah, I. A. Qasmieh, W. A. Mustafa, and H. Alquran, “PneumoniaNet: Automated Detection and Classification of Pediatric Pneumonia Using Chest X-ray Images and CNN Approach,” Electronics, vol. 10, no. 23, article no. 2949, November 2021, doi: 10.3390/electronics10232949.

[10] X. Zhang, S. Lu, S. H. Wang, X. Yu, S. J. Wang, L. Yao, Y. Pan, and Y. D. Zhang, "Diagnosis of Covid-19 Pneumonia Via a Novel Deep Learning Architecture," Journal of computer science and technology, vol. 37, no. 2, pp. 330-343, March 2022, doi: 10.1007/s11390-020-0679-8.

[11] H. Al-Hasani, H. Abboudi, T. Ninan, B. Shaygi, and C. Roobottom, “Smartphone Applications for the Radiologist,” OJRad, vol. 03, no. 04, pp. 231–237, November 2013, doi: 10.4236/ojrad.2013.34037.

[12] S. A. R. Moezzi, A. Ghaedi, M. Rahmanian, S. Z. Mousavi, and A. Sami, “Application of deep learning in generating structured radiology reports: a transformer-based technique,” Journal of Digital Imaging, vol. 36, no. 1, pp. 80-90, August 2023, doi: 10.1007/s10278-022-00692-x.

[13] W. Tanguay, P. Acar, B. Fine, M. Abdolell, B. Gong, A. Cadrin-Chenevert, et al., “Assessment of Radiology Artificial Intelligence Software: A Validation and Evaluation Framework,” Can Assoc Radiol J, vol. 74, no. 2, pp. 326–333, May 2023, doi: 10.1177/08465371221135760.

[14] W. F. Wiggins, K. Magudia, T. M. S. Schmidt, S. D. O’Connor, C. D. Carr, M. D. Kohli, et al., “Imaging AI in Practice: A Demonstration of Future Workflow Using Integration Standards," Radiology: Artificial Intelligence, vol. 3, no. 6, articel no. E210097, October 2021, doi: 10.1177/08465371221135760.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,”

Communications of the ACM, vol. 60, no. 6, pp. 84-90, May 2017, doi: 10.1145/3065386.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," Commun. ACM, vol. 60, no. 6, pp. 84–90, May 2017, doi: 10.1145/3065386.

[17] A. Krizhevsky, "One weird trick for parallelizing convolutional neural networks," arXiv, article no. 1404.5997, April 2014.

[18] H. C. Chen, A. M. Widodo, A. Wisnujati, M. Rahaman, J. C. W. Lin, L. Chen, et al., "AlexNet Convolutional Neural Network for Disease Detection and Classification of Tomato Leaf," Electronics, vol. 11, no. 6, article no. 951, March 2022, doi: 10.3390/electronics11060951.

[19] J. Si, S. L. Harris, and E. Yfantis, "A Dynamic ReLU on Neural Network," in 2018 IEEE 13th DallasCircuits and Systems Conference (DCAS), Dallas, TX: IEEE, pp. 1-6, November 2018, doi: 10.1109/dcas.2018.8620116.

[20] B. Graham, "Fractional Max-Pooling," arXiv, article no. 1412.0671v4, May 2015.

[21] A. Kumar M. and A. Chakrapani, "Classification of ECG signal using FFT based improved Alexnetclassifier," PLoS ONE, vol. 17, no. 9, article no. e0274225, September 2022, doi: 10.1371/journal.pone.0274225.

[22] H. Moujahid, B. Cherradi, O. E. Gannour, L. Bahatti, O. Terrada, and S. Hamida, "Convolutional Neural Network Based Classification of Patients with Pneumonia using X-ray Lung Images," Adv. sci.technol. eng. syst. j., vol. 5, no. 5, pp. 167–175, September 2020, doi: 10.25046/aj050522.

[23] D. Kermany, K. Zhang, and M. Goldbaum, "Labeled optical coherence tomography (oct) and chest x-ray images for classification," Mendeley data, vol. 2, no. 2, article no. 651, January 2018, doi: 10.17632/rscbjbr9sj.2

[24] U. Sait, K.V. L. Gokul, S. P. Prajapati, R. Bhaumik, T. Kumar, S. Shivakumar, et al. "Curated dataset for COVID-19 posterior-anterior chest radiography images (X-Rays)," Mendeley Data, v1, September 2020, doi: 10.17632/9xkhgts2s6.1.

[25] S. B. AlSumairi and M. M. B. Ismail, "X-ray image based pneumonia classification using convolutional neural networks," TIPCV, vol. 6, no. 20, pp. 54–67, August 2020, doi: 10.19101/tipcv.2020.618050.

[26] A. Antoniou, A. Storkey and H. Edwards, "Augmenting Image Classifiers Using Data Augmentation Generative Adversarial Networks," Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, vol. 3, no. 27, pp. 594-603, September 2018, doi: 10.1007/978-3-030-01424-7_5

[27] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, et al. "TensorFlow: Large-scale machine learning on heterogeneous Distributed systems," arXiv, article no. 1603.04467v2, March 2016.

[28] J. Wu, "Introduction to convolutional neural networks," National Key Lab for Novel Software Technology, China: Nanjing University, vol. 5, no. 23, article no. 495, May 2017.

[29] Z. P. Jiang, Y. Y. Liu, Z. E. Shao, and K. W. Huang, "An Improved VGG16 Model for Pneumonia Image Classification," Applied Sciences, vol. 11, no. 23, article no. 11185, November 2021, doi: 10.3390/app112311185.