Medical Images Application for X-Ray Image Classification Based on Convolution Neural Network

Budi Sugandi[1] and Galang Samudra[1]

[1] Electrical Engineering Department, Batam State Polytechnic, Batam, Indonesia
budi_sugandi@polibatam.ac.id

Abstract. The lungs are vital organs for the respiration of the human body. If it's not correctly observed, it can lead to disease in this organ. It's like pneumonia, where one child dies every 39 seconds. Indoseia ranks third after India and China in the case of tuberculosis disease. During the COVID-19 pandemic, 6.812.127 confirmed cases were reported in Indonesia. The abnormalities in the lung can be seen through the X-ray imaging. X-ray is a medical procedure that uses waves of X-ray radiation with a grayscale color type. When a doctor analyses a large number of X-ray images, it's exhausting and requires a high level of focus. Convolutional Neural Networks (CNN) are widely used today to classify images or videos. Many researchers have used CNN architecture methods to read and identify diseases from X-ray images, as well as using CNN's derivative architecture, DenseNet121, to re-train the CNN output model, which aims to improve the effectiveness of models in classifying X-ray images. This research aims to create a Graphical User Interface (GUI) that implements the CNN classification results. It can read X-rays not only one but more than two at a time, and the identification process is fast and easy to operate. Also, the accuracy of the results of the reading of X-ray images, as the final result of this research process, obtained a result of 92% accuracy. The model is integrated with the Graphical User Interface (GUI), which can perform classification against four conditions on X-ray images.

Keywords: *Lungs, X-Ray Imaging, Convolutional Neural Networks, DenseNet121.*

1 Introduction

The lungs are the vital organs for the respiration of the human body. It plays a role as an organ that exchanges carbon dioxide gas for oxygen inside the body. Although the lungs are one of the most vital organs of the body, there are still many people who aren't aware of the health of their lungs. If the respiration function of humans is disrupted, then it can be assured that it will affect them directly, and if it is not treated carefully, it can lead to disease or damage to the lungs that causes death. Microorganisms, such as bacteria, viruses, and fungi, generally cause lung diseases.

L. Lumombo et al. (eds.), *Proceedings of the 7th International Conference on Applied Engineering (ICAE 2024)*,
Advances in Engineering Research 251,
https://doi.org/10.2991/978-94-6463-620-8_18

UNICEF, a global organisation aiding children in need, warns that every 39 seconds, a child dies of pneumonia. Most of the deaths happened to children under two years, and pneumonia is one of the biggest causes of Indonesian children's deaths, as 19.000 children died in 2018 due to pneumonia [1]. On the other hand, tuberculosis is a disease caused by the bacterial pathogen Mycobacterium tuberculosis (TBC). This disease can cause death if the medicine is not consumed correctly in the 6-month range. Tuberculosis is still a global health issue, and in 2017 WHO report, there were 1.3 million deaths caused by tuberculosis and Indonesia was ranked third after India and China in tuberculosis cases [2]. At the end of 2019, acute respiratory distress disease, also known as coronavirus disease 2019 (COVID-19), was first reported to appear in Wuhan, the capital city of Hubei province, China. This disease become a global concern because of its high spreading rate and causes cases and death. The government implemented a full quarantine of the city of Wuhan (lockdown) [3][4]. In 2022, the information from the Ministry of Health of the Republic of Indonesia regarding the development of cases of people infected with the COVID-19 virus as of July 5 2023, is 6,812,127 confirmed cases, 6,642,003 cases recovered (97.5%), 161,879 cases died (2 .4%), and 8,245 active cases (0.1%) [4][5].

Those diseases can only be detected by doing some specific test or using X-ray imaging of the lungs. The X-ray imaging process shoots the X-ray beam to the body. The output is a sheet of grayscale image [5][6]. To identify the abnormality of the lungs, the specialist doctor and radiologist have to analyze the x-ray images properly so that they won't make wrong diagnoses to the patients. It needs a lot of patience and attention to analyze the images.

One of the ways to identify and detect objects in an image is by using neural networks. Convolutional neural networks (CNN) are neural network architectures based on the improvement of multi-layer perceptron (MLP) and backpropagation. CNN is widely used for object identification and detection [6][7]. CNN works by doing classification that uses image processing and can be used for labelled data using the supervised learning method. Supervised learning works if there is a dataset that can be trained and has a targeted variable as output [7][8].

Although CNN has become a very popular machine-learning method for image recognition, new research by Huang et al. provides information about performance degradation, gradient vanishing, or other problems. [8] developed a new neural network architecture known as Densely Connected Convolutional Network (DenseNet) and won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2017. This architecture skips connections and connects all layers directly each other which is referred to as dense connectivity. DenseNet uses the concatenation of input data instead of the summation of them [9].

2 Methods

2.1 X-ray Images Dataset

X-ray imaging or Rontgen, is a process of taking images from human organs using X-ray beam through the body. The result of x-ray imaging is printed on a grayscale sheet of the file. This research will consist of dataset x-ray images for artificial intelligence model training. The dataset is obtained from open resource sites such as Kaggle, which is a sharing idea and data science platform. Then, the dataset that has been obtained is a collection of X-ray images with the normal conditions and X-ray images with COVID-19, pneumonia, and tuberculosis cases.

Table 1. Dataset detail

Condition Class	Amount	Train Data	Test Data
Normal	11.995	10.795	1.200
COVID-19	5.243	4.718	525
Pneumonia	3.146	2.831	315
Tuberculosis	1.221	1.099	122
Total	**21.606**	**19.444**	**2.162**

The dataset is separated into 2 files: those are train data that will be used as a training process up to 90% of the total dataset, and test data to validate the results of the model predictions is up to 10% of the total dataset.

2.2 Preprocessing Data

Preprocessing data is a process to convert raw data into a form that is easier to understand, as shown in Fig. 1. This process will change and fix the dataset so that all images are the same size and shape, improving the X-ray image quality for the training data process. This process also adds a variety of image characteristics so the model can be more precise when making predictions.

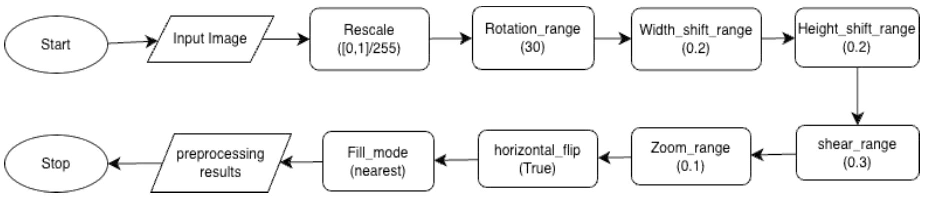


Fig. 1. Preprocessing process

In the preprocessing stages, The first step was to normalize the image pixel size to 0 and 1 by dividing each pixel value by 255, which aims to make the data easier to manage by the model. Next, the image is rotated randomly, which has been determined by 30 degrees, which means the image can be rotated a maximum of 30 degrees clockwise or counterclockwise and shift 0.2 or 20% of the width and height of the image. The shear_range will transform the shear, which makes the shape of the image tilted, this creates a stretch in the image with a tilt angle of 30 degrees. Then zoom_range will allow the image to be enlarged or reduced by 10% and rotated horizontally. And fill_mode (nearest) is used to fill the nearest empty pixels that appear in the image after the rotation or transformation process. The results of this process increase the diversity of data used for model training, so that the model can generalize to data it has never seen before. The result of preprocessing data is shown in Fig. 2.

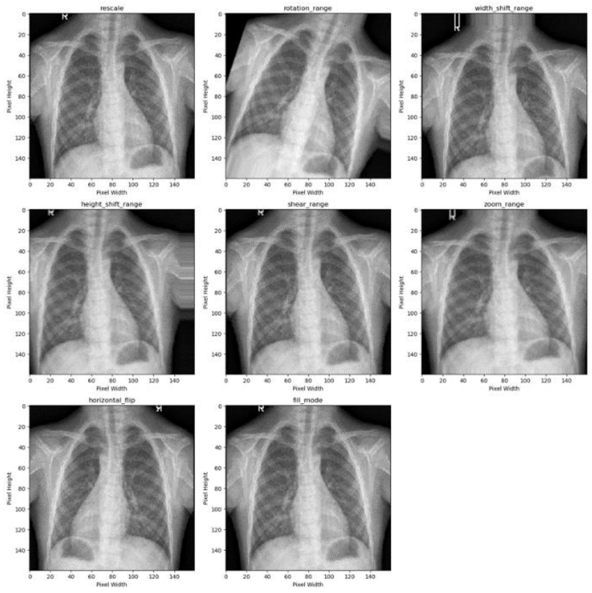


Fig. 2. Preprocessing data results

2.3 Convolutional Neural Networks (CNN)

Convolutional Neural Networks are a type of architecture of deep learning model to process 2 dimension data. Because of its capabilities, CNN is widely used in processing image data, one of the uses of which is to carry out image classification and object detection [10][11]. Convolutional Neural Networks consist of an input layer, a hidden and an output layer. These layers are divided into the convolutional layer, a pooling layer, and a fully connected layer [12].

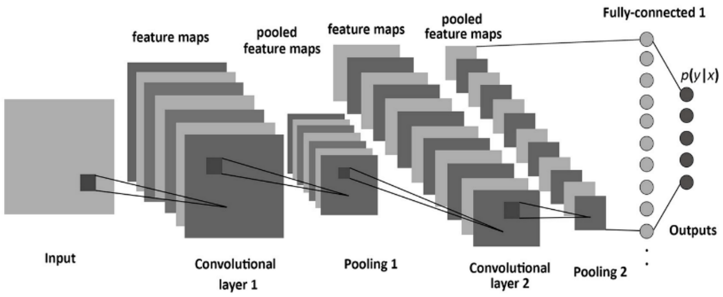


Fig. 3. Convolutional Neural Networks Architecture

The convolutional layers function extracts the image features from input with filters, which are basically a form of estimation. The results of feature extraction will be image patterns that will be identified, and weights will be produced to describe the image patterns that have been identified. This is then followed by a pooling layer, which will reduce the spatial size and can reduce the computing resources needed to process data by reducing the dimensions of the feature map (downsampling), which can speed up computation because fewer parameters are updated [12][14]. The pooling layer is done by max-pooling, namely by taking the maximum value from the previous region and reducing the number of parameters and also the complex procedure of the model to prevent overfitting. Fully connected will connect all the previous layers and utilize the activation function to make predictions[11].

2.4 DenseNet121

This paper utilize DensNet (Densely Connected Convolutional Network). It is a derivative architecture model from CNN that retrains the outputs of CNN model. DenseNet121 consists of 121 layers to classify x-ray images. This deep learning architecture utilizes less computing power and memory to process [13]. DenseNet121 used the output from a layer model to be used as input for the next neuron layers [14] [15]. DenseNet121 which connects each layer of neurons, as in the first layer is connected to the second, third and so on. This aims to empower the largest data movement between neuron layers [15][16].

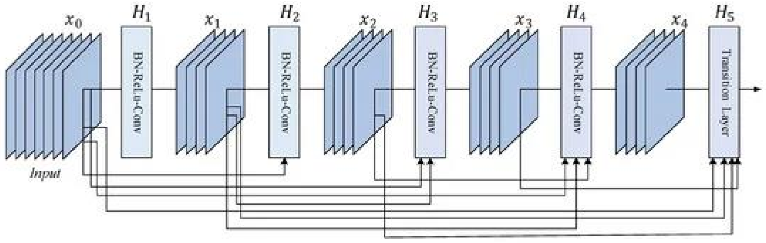


Fig. 4. DenseNet121 Architecture

Fig. 4 Shows that DenseNet architecture is construct by multiple layers that are connected to dense blocks. Each layers utilizes the data input from layers of previous model

to generate feature map and send the data to all of following layers [13]. As the X0 is an image that is passing through the convolutional network. The l^th-layer receives all of the previous feature maps ($x_0, \dots, x_{l-1}$) as input:

$$x_l = H_l([x_0, x_1, \dots, x_{l-1}]) \quad (1)$$

The equation $x_l = H_l([x_0, x_1, \dots, x_{l-1}])$ represents the consisted feature maps generated of layer (0, ..., l-1). x_l is the output of the l^th-layer, and H_l is the l^th-layer itself. Which is a composite function that comprises three consecutive operations: batch normalization (BN), rectified linear unit (ReLU), and 3 x 3 convolution (Conv) [8].

2.5 Model Training

In this phase, the neural network architecture is creating and training an artificial intelligence model utilizing the images from the dataset that has been pre-processing stage, which can make the model capable of making predictions in x-ray images.

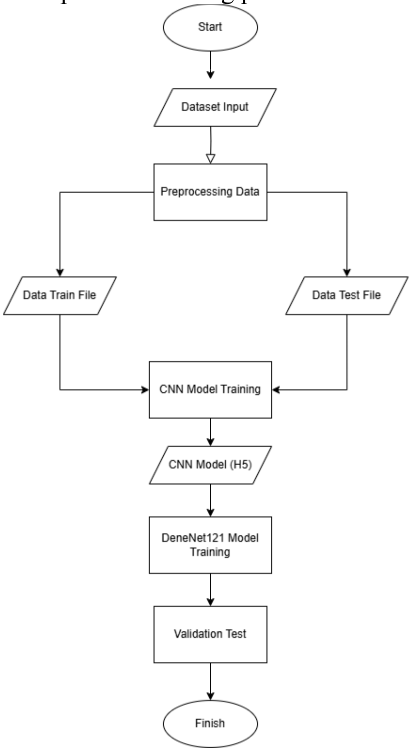


Fig. 5. Training process flowchart

The proposed system was implemented in python languange. CNN and DenseNet121 architecture were developed in TensorFlow Keras Application. Before the model is trained, the model needs to be configured such as activation function, optimizer, loss function, and the amount of epochs. And in the process on DenseNet121 training softmax activation function was used for multi-class classification. For the optimizer to optimize the weight and biases, adam optimizer was used in this research with a learning rate of 0.001. Categorical crossentropy was used for the loss function. In the process of model training, the number step of epochs for CNN training is 25 epochs and DenseNet121 is 15 epochs.

2.6 Model Validation

When the model training is finished, it's important to validation test on the model. The model's accuracy and precision needs to be measured, to ensure the model performance and the method to do that is confusion matrix. Confusion matrix is a two-dimensional matrix in which the columns represent the classifier's predicted labels and the true labels are displayed. It's used to measure the model's performance and shows the number of true or false predictions of the predicted data. The number of prediction made by model will be the variable to calculate aggragate metrics such as accuracy, precision, recall, and F1-Score. And to calculate accuracy, precision, recall, and F1-score first it needs the parameter from all of that which is False Negative (FN), False Positive (FP), True Positive (TP), and True Negative (TN) [16].

2.7 Graphical User Interface (GUI)

Graphical user interface (GUI) is an interface to interact to function of application in computer. And usually consist of button, label, toggle, to operate or execute programs. GUI isn't just processing words or numbers, but also images and videos. With GUI users can be more comfortable when using or operating applications. This purposed research is read and classify x-ray images whether the lungs is normal or not. And to make it simple to show the result from model predictions, a GUI will be used for this research. Instead of typing some codes to classify the X-ray images, it just needs to insert the photo and click the button to show the model prediction. This method will help and make it simpler when users want to classify X-ray images.

3 Result

In this section contains the result of model summary, model training and model evaluation of Convolutional Neural Networks and DenseNet121 architecture. All the x-ray images were resized to 160 x 160 x 3 for suitable input into the model. And the result will be shown on the GUI. The model is trained first using CNN architecture and the model then being retrain with DenseNet121

3.1 Model Summary

Table 2 and Table 3 show the model summary, which consists of the training process stages. Table 3. is the model summary of CNN. It's consist of layers such as convolutional layer (Conv2D), pooling layer (MaxPooling2D), and fully connected layer

(Dense). Dropout layer act to prevent overfitting for the training process. The output shape is the result the image input from the layers. The param is a parameter unit consist of weight and bias that will be learned and optimized during the model training process. Table 4 shows the summary of DenseNet121 model. It's consist of layers as the CNN architecture. denseNet121 adds padding (rows and columns) around the image input to keep the image size after convolution. And the next stage is composite function such as convolutin (Conv2D), batch normalization (BatchNormalization), and rectified linear unit (Activation). In this model the non-trainable params is a parameter in model that can't be changed or optimized during the model training process.

Table 2. CNN architecture model summary

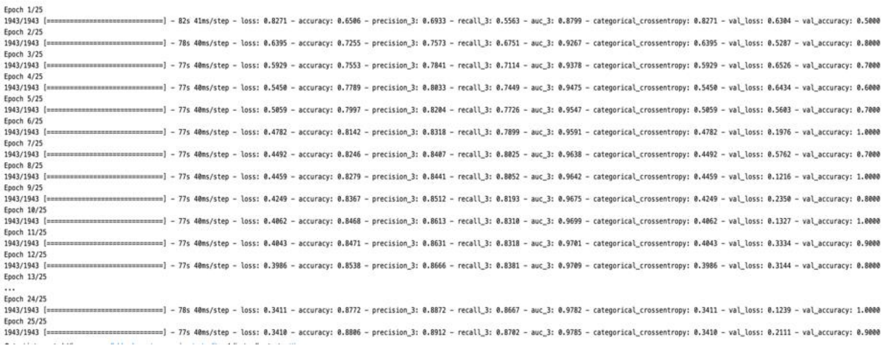
Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 158, 158, 16)	448
max_pooling2d_6 (MaxPooling2D)	(None, 79, 79, 16)	0
conv2d_7 (Conv2D)	(None, 77, 77, 32)	4640
max_pooling2d_7 (MaxPooling2D)	(None, 38, 38, 32)	0
conv2d_8 (Conv2D)	(None, 36, 36, 64)	18496
max_pooling2d_8 (MaxPooling2D)	(None, 18, 18, 64)	0
fatten_2 (Flatten)	(None, 20736)	0
dense_9 (Dense)	(None, 200)	4147400
Dropout_4 (Dropout)	(None, 200)	0
Total params: 4273488 (16.30 MB)		
Trainable params: 4273488 (16.30 MB)		
Non-trainable params: 0 (0.00 Byte)		

Table 3. DenseNet121 architecture model summary

Layer (type)	Output Shape	Param #
Input_7 (InputLayer)	(None, 160, 160, 3)	0
zero_padding2d_12 (ZeroPadding2D)	(None, 166, 166, 3)	0
conv1/conv (Conv2D)	(None, 80, 80, 64)	9408
conv1/bn (BatchNormalization)	(None, 80, 80, 64)	256
conv1/relu (Activation)	(None, 80, 80, 64)	0
zero_padding2d_13 (ZeroPadding2D)	(None, 82, 82, 64)	0
pool1 (MaxPooling2D)	(None, 40, 40, 64)	0
conv2_block1_0_bn (BatchNormalization)	(None, 40, 40, 64)	256
conv2_block1_0_relu (Activation)	(None, 40, 40, 64)	0
Total params: 7037504 (26.85 MB)		
Trainable params: 6953856 (26.53 MB)		
Non-trainable params: 83648 (326.75 KB)		

3.2 Training result

In this section the model training result is shown in fig. 6 and fig. 7.

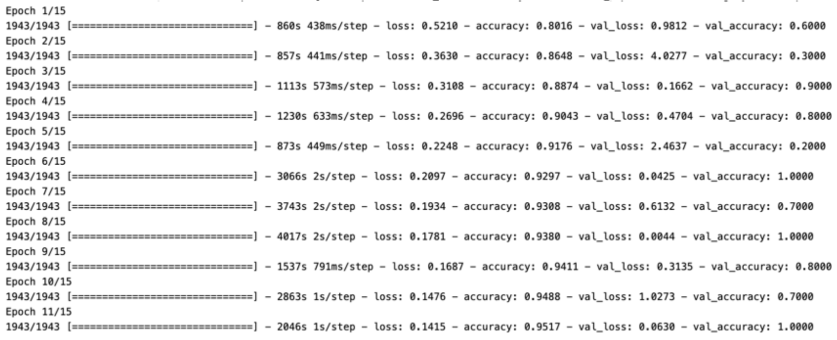


```
Epoch 1/25
1943/1943 [==============================] - 82s 41ms/step - loss: 0.8271 - accuracy: 0.6506 - precision_3: 0.6933 - recall_3: 0.5563 - auc_3: 0.8799 - categorical_crossentropy: 0.8271 - val_loss: 0.6304 - val_accuracy: 0.5000
Epoch 2/25
1943/1943 [==============================] - 78s 40ms/step - loss: 0.6395 - accuracy: 0.7255 - precision_3: 0.7573 - recall_3: 0.6751 - auc_3: 0.9267 - categorical_crossentropy: 0.6395 - val_loss: 0.5287 - val_accuracy: 0.8000
Epoch 3/25
1943/1943 [==============================] - 77s 40ms/step - loss: 0.5929 - accuracy: 0.7553 - precision_3: 0.7841 - recall_3: 0.7114 - auc_3: 0.9378 - categorical_crossentropy: 0.5929 - val_loss: 0.6526 - val_accuracy: 0.7000
Epoch 4/25
1943/1943 [==============================] - 77s 40ms/step - loss: 0.5450 - accuracy: 0.7789 - precision_3: 0.8033 - recall_3: 0.7449 - auc_3: 0.9475 - categorical_crossentropy: 0.5450 - val_loss: 0.6434 - val_accuracy: 0.6000
Epoch 5/25
1943/1943 [==============================] - 77s 40ms/step - loss: 0.5059 - accuracy: 0.7997 - precision_3: 0.8204 - recall_3: 0.7726 - auc_3: 0.9547 - categorical_crossentropy: 0.5059 - val_loss: 0.5603 - val_accuracy: 0.7000
Epoch 6/25
1943/1943 [==============================] - 77s 40ms/step - loss: 0.4782 - accuracy: 0.8142 - precision_3: 0.8318 - recall_3: 0.7899 - auc_3: 0.9591 - categorical_crossentropy: 0.4782 - val_loss: 0.1976 - val_accuracy: 1.0000
Epoch 7/25
1943/1943 [==============================] - 77s 40ms/step - loss: 0.4492 - accuracy: 0.8246 - precision_3: 0.8407 - recall_3: 0.8025 - auc_3: 0.9638 - categorical_crossentropy: 0.4492 - val_loss: 0.5762 - val_accuracy: 0.7000
Epoch 8/25
1943/1943 [==============================] - 77s 40ms/step - loss: 0.4459 - accuracy: 0.8279 - precision_3: 0.8441 - recall_3: 0.8052 - auc_3: 0.9642 - categorical_crossentropy: 0.4459 - val_loss: 0.1216 - val_accuracy: 1.0000
Epoch 9/25
1943/1943 [==============================] - 77s 40ms/step - loss: 0.4249 - accuracy: 0.8367 - precision_3: 0.8512 - recall_3: 0.8193 - auc_3: 0.9675 - categorical_crossentropy: 0.4249 - val_loss: 0.2350 - val_accuracy: 0.8000
Epoch 10/25
1943/1943 [==============================] - 77s 40ms/step - loss: 0.4062 - accuracy: 0.8468 - precision_3: 0.8613 - recall_3: 0.8310 - auc_3: 0.9699 - categorical_crossentropy: 0.4062 - val_loss: 0.1327 - val_accuracy: 1.0000
Epoch 11/25
1943/1943 [==============================] - 77s 40ms/step - loss: 0.4043 - accuracy: 0.8471 - precision_3: 0.8631 - recall_3: 0.8318 - auc_3: 0.9701 - categorical_crossentropy: 0.4043 - val_loss: 0.3334 - val_accuracy: 0.9000
Epoch 12/25
1943/1943 [==============================] - 77s 40ms/step - loss: 0.3986 - accuracy: 0.8538 - precision_3: 0.8666 - recall_3: 0.8381 - auc_3: 0.9709 - categorical_crossentropy: 0.3986 - val_loss: 0.3144 - val_accuracy: 0.8000
Epoch 13/25
...
Epoch 24/25
1943/1943 [==============================] - 78s 40ms/step - loss: 0.3411 - accuracy: 0.8772 - precision_3: 0.8872 - recall_3: 0.8667 - auc_3: 0.9782 - categorical_crossentropy: 0.3411 - val_loss: 0.1239 - val_accuracy: 1.0000
Epoch 25/25
1943/1943 [==============================] - 77s 40ms/step - loss: 0.3410 - accuracy: 0.8806 - precision_3: 0.8912 - recall_3: 0.8702 - auc_3: 0.9785 - categorical_crossentropy: 0.3410 - val_loss: 0.2111 - val_accuracy: 0.9000
```

Fig. 6. CNN training result

This is the result of CNN model training is done with 25 epochs. The model performance increased which indicates that the model continues to learn and improve its performance until 99%, and after that the training process is done. The accuracy the model gets is around 87%.

The model result from CNN training process then will get retrain in DenseNet121 architecture to increase the accuracy from the previous training.



```
Epoch 1/15
1943/1943 [==============================] - 860s 438ms/step - loss: 0.5210 - accuracy: 0.8016 - val_loss: 0.9812 - val_accuracy: 0.6000
Epoch 2/15
1943/1943 [==============================] - 857s 441ms/step - loss: 0.3630 - accuracy: 0.8648 - val_loss: 4.0277 - val_accuracy: 0.3000
Epoch 3/15
1943/1943 [==============================] - 1113s 573ms/step - loss: 0.3108 - accuracy: 0.8874 - val_loss: 0.1662 - val_accuracy: 0.9000
Epoch 4/15
1943/1943 [==============================] - 1230s 633ms/step - loss: 0.2696 - accuracy: 0.9043 - val_loss: 0.4704 - val_accuracy: 0.8000
Epoch 5/15
1943/1943 [==============================] - 873s 449ms/step - loss: 0.2248 - accuracy: 0.9176 - val_loss: 2.4637 - val_accuracy: 0.2000
Epoch 6/15
1943/1943 [==============================] - 3066s 2s/step - loss: 0.2097 - accuracy: 0.9297 - val_loss: 0.0425 - val_accuracy: 1.0000
Epoch 7/15
1943/1943 [==============================] - 3743s 2s/step - loss: 0.1934 - accuracy: 0.9308 - val_loss: 0.6132 - val_accuracy: 0.7000
Epoch 8/15
1943/1943 [==============================] - 4017s 2s/step - loss: 0.1781 - accuracy: 0.9380 - val_loss: 0.0044 - val_accuracy: 1.0000
Epoch 9/15
1943/1943 [==============================] - 1537s 791ms/step - loss: 0.1687 - accuracy: 0.9411 - val_loss: 0.3135 - val_accuracy: 0.8000
Epoch 10/15
1943/1943 [==============================] - 2863s 1s/step - loss: 0.1476 - accuracy: 0.9488 - val_loss: 1.0273 - val_accuracy: 0.7000
Epoch 11/15
1943/1943 [==============================] - 2046s 1s/step - loss: 0.1415 - accuracy: 0.9517 - val_loss: 0.0630 - val_accuracy: 1.0000
```

Fig. 7. DenseNet121 training result

The DenseNet121 training process is done with 15 epochs. And in this training the model performance is increase significantly which also indicates that the model is learing more until the training process is done. Subsequently it's necessary to analyze the progress of trained model. This can be done using train accuracy, train loss, validation accuracy, and validation loss method. And the process is using library function in python as graphical 2D images shown in fig. 8 and fig. 9.

Train Loss measures the values lost during the training process and measures how well the model makes correct predictions, so this training process exists to reduce loss so that the model can predict well. Validation Loss is a measure of the average error of the model when tested with validation data and provides an indication of how well the model can make predictions from training data to new data that it has never seen before. Train Accuracy is the level of correct predictions made by the model on the training data and provides an indication of how well the model can learn the patterns in the training data. Validation Accuracy is the level of correct predictions made by the model against validation data.

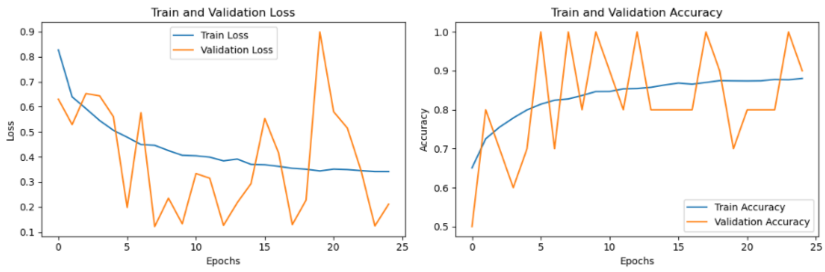


Fig. 8. Train & validation loss, train & validation accuracy of CNN model

Fig. 8 shows the changes in the loss values in the train and validation graphs from the results of the model training process using the CNN architecture. It can be seen that train loss decreases significantly as repetition cycles (epochs) increase, which shows that the model is learning and improving its performance on the training data. The validation loss graph looks very fluctuating, which indicates overfitting (the model learns the data in very detail so it cannot generalize well to new data). In the train accuracy graph, it can be seen that it increases significantly and is stable as the epochs increase. Which shows that the model has good performance in the training process. Validation accuracy appears to fluctuate and does not show consistent improvement. This also indicates the possibility of overfitting.

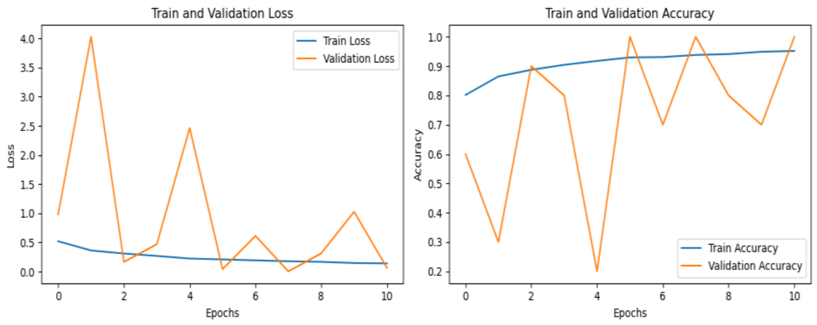


Fig. 9. Train & validation loss, train & validation accuracy

Fig. 9 shows the loss and accuracy graph from train & validation for the DenseNet121 model. In train Loss, it can be seen that it decreases gradually and consistently compared to conventional CNN architecture with each additional epochs. In validation loss the moving graph is very volatile but still shows an overall decrease, which indicates that the model can generalize the data better than the CNN architecture. Then for the train accuracy graph it increases and is stable as in the conventional CNN model. Validation accuracy shows fluctuating movements, but there is still an increase.

3.3 Model evaluation

In this model evaluation consist of confusion matrix, and the F1-Score report of the model. With the total data of 2.162 validation data to measure.

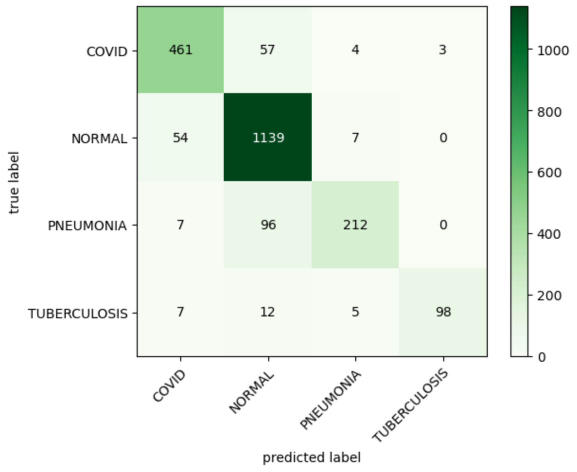


Fig. 10.CNN model confusion matrix

The confusion matrix of the CNN model in Fig. 10 can be seen from the main diagonal that the model has good performance in classifying images with normal, pneumonia, and tuberculosis classes because the number of True Positives (TP) for these classes is relatively high. And from 2162 validation data, there were 252 incorrect data predictions by the model.

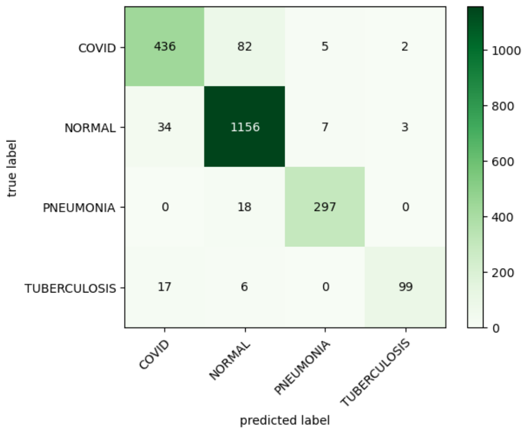


Fig. 11. DenseNet121 model confusion matrix

And in the confusion matrix of the DenseNet121 model in Figure 11, it can be seen from the main diagonal that the model has better performance. This was marked by an increase in the number of predictions for the normal and pneumonia classes, and there was no significant change in the tuberculosis class. However, in the COVID-19 class, there is a decrease in the number of predictions, because the model has difficulty identifying images with the COVID-19 class because the number of false negatives (FN) for this class is quite high. From 2162 validation data, there were 174 incorrect predictions by the model. This shows that there is an increase in prediction results by the DenseNet121 model even though there are still several prediction errors in each class provided by the model.

The parameter from confusion matrix is obtained. To further analyze the results of the two models, by doing the classification report, which can be seen in Table 4 and 5. Table 4 is a classification report from the conventional CNN model. It can be seen that the accuracy of each class is COVID-19 88%, normal 95%, pneumonia 67%, and tuberculosis 80%. for the pneumonia class, the accuracy for pneumonia is not quite good, so the accuracy of the model 88% is obtained. Table 5 is the classification report from the DenseNet121 model. It can be seen that the accuracy of each class is COVID 85%, normal 97%, pneumonia 97%, and tuberculosis 89%. So a total accuracy of 92% is obtained for this architecture. There is a significant increase in model accuracy after training using the DenseNet121 architecture.

Table 4. CNN model classification report

	Accuracy	Precision	Recall	F1-score	support
COVID	0.88	0.87	0.88	0.87	525
NORMAL	0.95	0.87	0.95	0.91	1200
PNEUMONIA	0.67	0.93	0.67	0.78	315
TUBERCULOSIS	0.80	0.97	0.80	0.88	122
accuracy	0.88				2162
Macro avg		0.91	0.83	0.86	2162
Weighted avg		0.89	0.88	0.88	2162

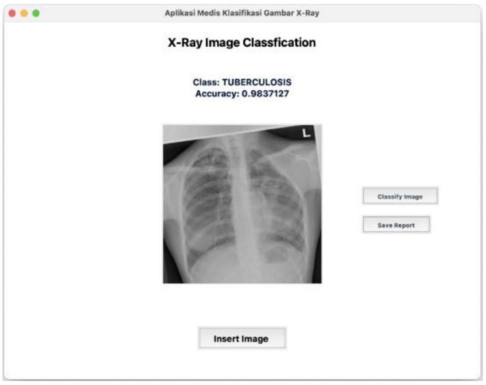
Table 5. DenseNet121 model classification report

	Accuracy	Precision	Recall	F1-score	support
COVID	0.85	0.94	0.85	0.89	525
NORMAL	0.97	0.93	0.97	0.96	1200
PNEUMONIA	0.97	0.95	0.97	0.96	315
TUBERCULOSIS	0.89	0.96	0.89	0.92	122
accuracy	0.92				2162
Macro avg		0.94	0.92	0.93	2162
Weighted avg		0.94	0.94	0.93	2162

3.4 Graphical User Inter Face (GUI) result

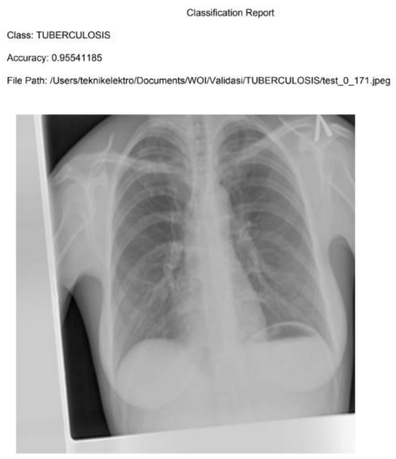
Graphical user interface (GUI) is integrated with the model from DenseNet121. This GUI provide widgets that can be operated to make a x-ray image classification. The GUI is consturcted with python and support library to program the interface. Here is some documentation on x-ray image classification test results using the integrated GUI to the model.

Fig 13 shows the GUI after the images is chosen, then the image will be displayed through the GUI display so the classify and save report button will show up. And when the results come out it can be saved into PDF report. Fig 14 is the PDF of classification report that comprise of the predictions, accuracy results and the location the data is stored.


Aplikasi Medis Klasifikasi Gambar X-Ray
X-Ray Image Classfication
L
Classify Image
Save Report
Insert Image

Class: TUBERCULOSIS
Accuracy: 0.9837127

Fig. 12.. Image classification


Classification Report

Class: TUBERCULOSIS

Accuracy: 0.95541185

File Path: /Users/teknikelektro/Documents/WOI/Validasi/TUBERCULOSIS/test_0_171.jpeg

Fig. 13. Classification result report

4 Conclusion

The use of neural networks architecture and deep learning to process and make a classification of x-ray. The model can make predictions with accuracy of 88% in CNN model and then increase to 92% using DenseNet121 model. This indicates that the use of artificial intelligence can help radiologist and specalist doctor to identifies lungs condition faster and reduce the times more. It can identifies 4 class which is normal, pneumonia, COVID-19, and tuberculosis in the thorax x-ray images. With the GUI that integrated to the program that makes the classification simple and much easier than using command line interface. And the result can be saved in pdf. With the information included which is the predictions, accuracy, and report location save file.

References

[1] "One child dies of pneumonia every 39 seconds, agencies warn," UNICEF Press Release. Accessed: Jul. 25, 2023. [Online]. Available: https://www.unicef.org/indonesia/id/siaran-pers/lembaga-kesehatan-dan-anak-memeringatkan-satu-anak-meninggal-akibat-pneumonia-setiap-39

[2] T. Dewi Kristini, R. Hamidah, F. Kesehatan Masyarakat, U. Muhammadiyah Semarang, and D. Kesehatan Provinsi Jawa Tengah, "Potensi Penularan Tuberculosis Paru pada Anggota Keluarga Penderita." [Online]. Available: https://jurnal.unimus.ac.id/index.php/jkmi,

[3] J. Prayitno, R. Admirasari, J. P. Susanto, R. Nugroho Pusat Teknologi Lingkungan BPPT, K. Puspiptek, and T. Selatan, "BIOTEKNOLOGI & BIOSAINS INDONESIA Review of Virus Inactivation Technologies for Covid-19 Pandemic Control." [Online]. Available: http://ejurnal.bppt.go.id/index.php/JBBI

[4] "INFEKSIEMERGING," Kementrian Kesehatan Republik Indonesia. Accessed: Jul. 27, 2023. [Online]. Available: https://infeksiemerging.kemkes.go.id/dashboard/covid-19

[5] F. Aulia Natasya, F. A. Natasya, P. Dokter, and F. Kedokteran, "TATALAKSANA PNEUMONIA." [Online]. Available: http://jurnalmedikahutama.com

[6] M. Saiful, L. M. Samsu, and F. Fathurrahman, "Sistem Deteksi Infeksi COVID-19 Pada Hasil X-Ray Rontgen menggunakan Algoritma Convolutional Neural Network (CNN)," Infotek : Jurnal Informatika dan Teknologi, vol. 4, no. 2, pp. 217–227, Jul. 2021, doi: 10.29408/jit.v4i2.3582.

[7] N. H. Harani, C. Prianto, and M. Hasanah, "Deteksi Objek Dan Pengenalan Karakter Plat Nomor Kendaraan Indonesia Menggunakan Metode Convolutional Neural Network (CNN) Berbasis Python," 2019.

[8] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks." [Online]. Available: https://github.com/liuzhuang13/DenseNet.

[9] R. Wang et al., "Densely connected convolutional networks for vibration based structural damage identification," Eng Struct, vol. 245, Oct. 2021, doi: 10.1016/j.engstruct.2021.112871.

[10] A. Y. W. R. S. I Wayan Suartika, "Klasifikasi Citra Menggunakan Convolutional Neural Network (Cnn) pada Caltech 101," JURNAL TEKNIK ITS, vol. 5, no. 1, pp. A65–A69, 2016.

[11] S. Sakib, A. 1#, A. Jawad, K. 2@, and H. Ahmed, "An Overview of Convolutional Neural Network: Its Architecture and Applications," 2019, doi: 10.20944/preprints201811.0546.v4.

[12] Bambang Pilu Hartato, "Penerapan Convolutional Neural Network pada Citra Rontgen Paru-Paru untuk Deteksi SARS-CoV-2," Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi), vol. 5, no. 4, pp. 747–759, Aug. 2021, doi: 10.29207/resti.v5i4.3153.

[13] S. A. Albelwi, "Deep Architecture based on DenseNet-121 Model for Weather Image Recognition," International Journal of Advanced Computer Science and Applications, vol. 13, no. 10, pp. 559–565, 2022, doi: 10.14569/IJACSA.2022.0131065.

[14] W. Wijaya Kusuma, R. Rizal Isnanto, A. Fauzi, and P. Korespondensi, "Analisis Perbandingan Model CNN VGG16 dan DenseNet121 Menggunakan Kerangka Kerja TensorFlow untuk Deteksi Jenis Hewan," Jurnal Teknik Komputer, vol. 1, no. 4, pp. 141–147, 2023, doi: 10.14710/jtk.v1i4.37009.

[15] J. Pardede and D. A. L. Putra, "Implementasi DenseNet Untuk Mengidentifikasi Kanker Kulit Melanoma," Jurnal Teknik Informatika dan Sistem Informasi, vol. 6, no. 3, Dec. 2020, doi: 10.28932/jutisi.v6i3.2814.

[16] M. Heydarian, T. E. Doyle, and R. Samavi, "MLCM: Multi-Label Confusion Matrix," IEEE Access, vol. 10, pp. 19083–19095, 2022, doi: 10.1109/ACCESS.2022.3151048.