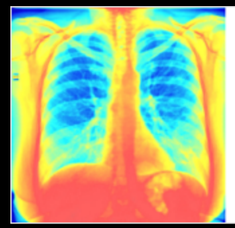
PRESENTASI

CHEST X-RAYS CLASSIFICATION (COVID,NORMAL,PNEUMONIA) USING CNN, ANN, DAN RESNET

Tugas Besar Mesin Pembelajar

Alvito Dwinovan Wibowo 140910200040
Aqiilah Cahya Ningrum 140910239008
Alief Atriadi Palguno 140910200010

SUMBER DATASET



COVID19+PNEUMONIA+NORMAL Chest X-Ray Image Dataset

Chest X-Ray Images Database

k kaggle.com

JURNAL

Automatic classification between COVID-19 pneumonia, non-COVID-19 pneumonia, and the healthy on chest X-ray image: combination of data augmentation …

[PDF] nature.com

M Nishio, S Noguchi, H Matsuo, T Murakami

Scientific reports, 2020 · nature.com

Abstract

This study aimed to develop and validate computer-aided diagnosis (CXDx) system for classification between COVID-19 pneumonia, non-COVID-19 pneumonia, and the healthy on chest X-ray (CXR) images. From two public datasets, 1248 CXR images were obtained, which included 215, 533, and 500 CXR images of COVID-19 pneumonia patients, non-COVID-19 pneumonia patients, and the healthy samples, respectively. The proposed CADx system utilized VGG16 as a pre-trained model and combination of

SHOW MORE

Save Cite Cited by 131 Related articles All 12 versions

Nishio, M., Noguchi, S., Matsuo, H. and Murakami, T., 2020. Automatic classification between COVID-19 pneumonia, non-COVID-19 pneumonia, and the healthy on chest X-ray image: combination of data augmentation methods. Scientific reports, 10(1), p.17532.

PROGRAM

- **IMPORT LIBRARY YANG DIPERLUKAN**

```
# Alvito Dwinovan Wibowo
# Program Image Detection COVID, PNEUMONIA, NORMAL

# Import Library dan Modul yang diperlukan
# Import Data Science Libraries
import numpy as np
import pandas as pd
import tensorflow as tf
from sklearn.model_selection import train_test_split
import itertools
import random

# Import visualization libraries
import os
import glob
import random
import shutil
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import pandas as pd
```

```
import tensorflow as tf
from tensorflow.keras.callbacks import EarlyStopping,ModelCheckpoint
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.models import Sequential
from tensorflow.keras import layers
from tensorflow.keras.layers import Dense, Dropout, Conv2D, MaxPooling2D, Flatten, Input

import warnings
warnings.filterwarnings('ignore')
```

PROGRAM

- **UPLOAD DATASET DAN CEK JUMLAH TIAP LABEL**

Exploratory Data Analysis

```
import os

base_dir = 'C:\\Alvito\\UNPAD\\SEMESTER 7\\Machine Learning aweu\\Tubes_ML\\Chest X-Ray'

# Menampilkan daftar file dan direktori di dalam base_dir
for file_or_dir in os.listdir(base_dir):
    print(file_or_dir)
```

```
COVID
NORMAL
PNEUMONIA
s41598-020-74539-2.pdf
Tubes ML_Alvito Dwinovan_CNN - Chest X ray.ipynb
```

```
def check_dir(dir: str = None, labels: list = None):
    for label in labels:
        num_data = len(os.listdir(os.path.join(dir, label)))
        print(f'Jumlah {label}: {num_data}')
print('Jumlah gambar pada setiap label: \n' + '='*50)
check_dir(base_dir,['COVID','NORMAL','PNEUMONIA'])
```

```
Jumlah gambar pada setiap label:
==================================================
Jumlah COVID: 1626
Jumlah NORMAL: 1802
Jumlah PNEUMONIA: 1800
```

PROGRAM

- **MEMBUAT DIREKTORI UNTUK TRAIN DAN VALIDATION**

Membuat direktori untuk train dan validation

```
def create_directories(base_dir=None, sub_dir=None, labels=None):

    # Buat direktori 'train' dan/atau 'test'
    # Jika direktori tersebut sudah ada maka akan melanjutkan dan tidak akan muncul error.
    for sub in sub_dir:
        dir_path = os.path.join(base_dir, sub)
        os.makedirs(dir_path, exist_ok=True)

    # Buat direktori berisi nama labels
    for label in labels:
        for sub in sub_dir:
            label_dir = os.path.join(base_dir, sub, label)
            os.makedirs(label_dir, exist_ok=True)

    return f'Telah berhasil membuat sub directories :{sub_dir} dan labels: {labels}'
```

[5]

```
labels = ['COVID','NORMAL','PNEUMONIA']
list_sub_dir = ['train', 'validation']

#panggil prosedur create_directories
create_directories(base_dir,  list_sub_dir, labels)
```

[6]

```
"Telah berhasil membuat sub directories :['train', 'validation'] dan labels: ['COVID', 'NORMAL', 'PNEUMONIA']"
```

PROGRAM

- MEMINDAHKAN DATA DARI DATASET KE TRAIN DAN VALIDATION

```
# Path ke folder 'train', dan 'test'
train_folder = os.path.join(base_dir,'train')
val_folder = os.path.join(base_dir,'validation')

# path ke folder 'rock', 'paper', 'scissors' (tempat file gambar)
COVID_folder = os.path.join(base_dir,'COVID')
NORMAL_folder = os.path.join(base_dir,'NORMAL')
PNEUMONIA_folder = os.path.join(base_dir,'PNEUMONIA')
```

```
import shutil
def split_data(source=None, destination=None, label=None,split_percentage=None):

    files = os.listdir(source)
    num_files = len(files)
    num_train = int(num_files * split_percentage)

    # Acak urutan file
    random.shuffle(files)

    train_files = files[:num_train]
    test_files = files[num_train:]

    # Pindahkan file ke folder tujuan
    for file in train_files:
        source_file = os.path.join(source, file)
        destination_file = os.path.join(destination[0],label)
        shutil.copy(source_file, destination_file)

    for file in test_files:
        source_file = os.path.join(source, file)
        destination_file = os.path.join(destination[1],label)
        shutil.copy(source_file, destination_file)

    return 'Telah berhasil memasukkan data'
```

PROGRAM

- **SPLIT DATA UNTUK TRAIN DAN VALIDATION**

```
# Persentase data yang akan digunakan sebagai split data
train_percentage = 0.8

#list folder train dan test
destination_folder= [train_folder,val_folder]
# Bagi data untuk masing-masing label
split_data(COVID_folder, destination_folder, 'COVID', train_percentage)
split_data(NORMAL_folder, destination_folder, 'NORMAL', train_percentage)
split_data(PNEUMONIA_folder, destination_folder,'PNEUMONIA', train_percentage)
```

```
'Telah berhasil memasukkan data'
```

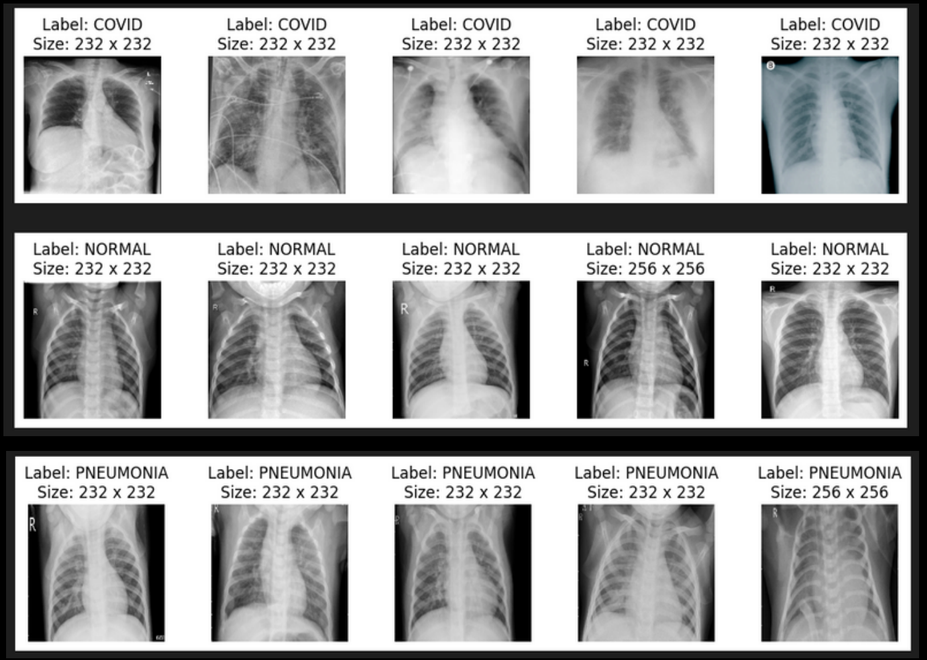
```
File Train : 4181 images
==================================================
Jumlah COVID: 1300
Jumlah NORMAL: 1441
Jumlah PNEUMONIA: 1440

File Validation : 1047 images
==================================================
Jumlah COVID: 326
Jumlah NORMAL: 361
Jumlah PNEUMONIA: 360
```

```
# Cek jumlah hasil split
train_sum = 0
validation_sum = 0

for i in os.listdir(base_dir + '/train'):
  res = base_dir + '/train/' + i
  train_sum += len(os.listdir(res))

for i in os.listdir(base_dir + '/validation'):
  res = base_dir + '/validation/' + i
  validation_sum += len(os.listdir(res))

print(f"File Train : {train_sum} images")
print(''+ '='*50)
check_dir(train_folder,['COVID','NORMAL','PNEUMONIA'])

print(f"\nFile Validation : {validation_sum} images")
print('' + '='*50)
check_dir(val_folder,['COVID','NORMAL','PNEUMONIA'])
```

PROGRAM

- **VISUALISASI DATASET**

```python
def visualize_random_data(data_dir, label, num_samples=5):
    label_dir = os.path.join(data_dir, label)
    image_files = os.listdir(label_dir)

    num_samples = min(num_samples, len(image_files))
    random_images = random.sample(image_files, num_samples)

    fig, axes = plt.subplots(1, num_samples, figsize=(10, 2), constrained_layout=True)

    # Kurangi jarak antara subplot horizontal dan vertikal
    plt.subplots_adjust(wspace=0.2, hspace=0.2)

    for j, random_image in enumerate(random_images):
        image_path = os.path.join(label_dir, random_image)

        img = mpimg.imread(image_path)
        image_size = f'Size: {img.shape[1]} x {img.shape[0]}'  # Menampilkan ukuran gambar
        title = f'Label: {label}\n{image_size}'  # Gabungkan label dan ukuran

        axes[j].imshow(img)
        axes[j].set_title(title)
        axes[j].axis('off')

    plt.show()

visualize_random_data(base_dir, label='COVID', num_samples=5)
visualize_random_data(base_dir, label='NORMAL', num_samples=5)
visualize_random_data(base_dir, label='PNEUMONIA', num_samples=5)
```



PROGRAM

- **MEMBUAT IMAGEDATAGENERATOR UNTUK AUGMENTASI**

```python
# Membuat objek ImageDataGenerator dengan augmentasi gambar
train_datagen = ImageDataGenerator(
    rescale = 1./255,
    rotation_range = 90,
    horizontal_flip = True,
    vertical_flip = True,
    shear_range = 0.2,
    fill_mode = 'nearest',
    brightness_range=[0.5, 1.5],
    width_shift_range=0.1,
    height_shift_range=0.1,)

validation_datagen = ImageDataGenerator(
    rescale = 1./225,
    rotation_range = 90,
    horizontal_flip = True,
    vertical_flip = True,
    shear_range = 0.2,
    fill_mode = 'nearest',
    brightness_range=[0.5, 1.5],
    width_shift_range=0.1,
    height_shift_range=0.1,)
```

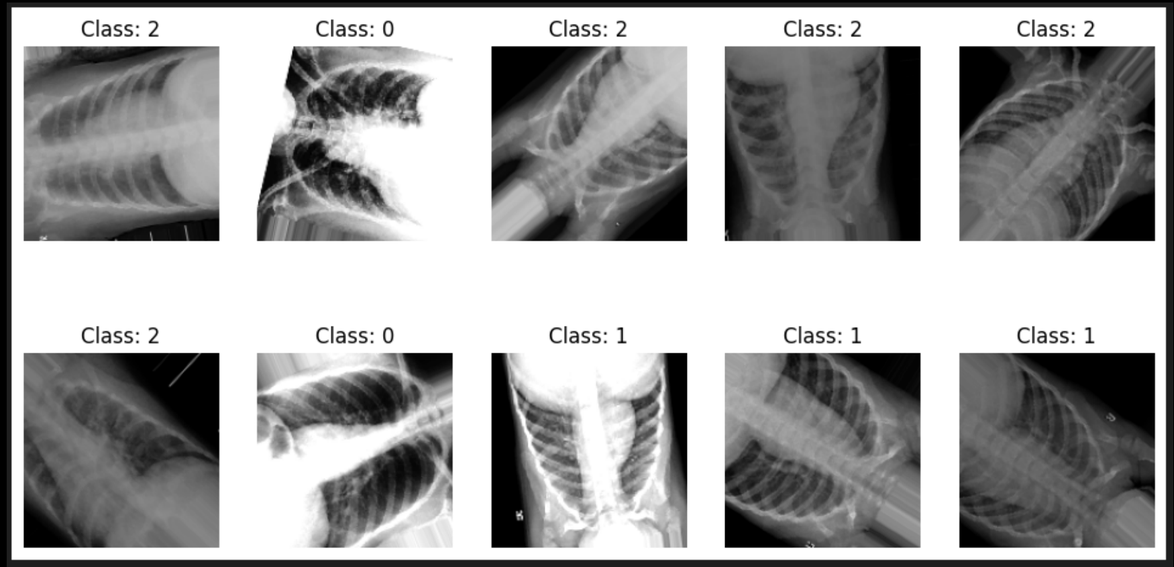
```python
# Menggunakan objek ImageDataGenerator untuk memuat data train dan validation
train_generator = train_datagen.flow_from_directory(
    base_dir + '/train',
    target_size=(150, 150),
    batch_size=4,
    class_mode='categorical'
)

validation_generator = validation_datagen.flow_from_directory(
    base_dir + '/validation',
    target_size=(150, 150),
    batch_size=4,
    class_mode='categorical'
)

class_indices = train_generator.class_indices
print(class_indices)
```

```
Found 4181 images belonging to 3 classes.
Found 1047 images belonging to 3 classes.
{'COVID': 0, 'NORMAL': 1, 'PNEUMONIA': 2}
```

PROGRAM

- **VISUALISASI AUGMENTASI GAMBAR**

```python
def display_augmented_images(data_generator, num_samples_to_display=10):
    num_rows= 2
    num_cols = int(num_samples_to_display/num_rows)

    fig, axes = plt.subplots(num_rows, num_cols, figsize=(12, 6))
    for i in range(num_rows):
        for j in range(num_cols):
            batch = data_generator.next()
            image = batch[0][0]  # Ambil gambar pertama dari batch
            label = batch[1][0]  # Ambil label pertama dari batch

            # Konversi label dalam format one-hot encoding ke kelas asli
            class_index = label.argmax()

            # Menampilkan gambar
            axes[i, j].imshow(image)
            axes[i, j].set_title(f"Class: {class_index}")
            axes[i, j].axis('off')

    plt.show()

display_augmented_images(train_generator, num_samples_to_display=10)
```



PEMBUATAN MODEL CNN

- PEMBUATAN MODEL CNN

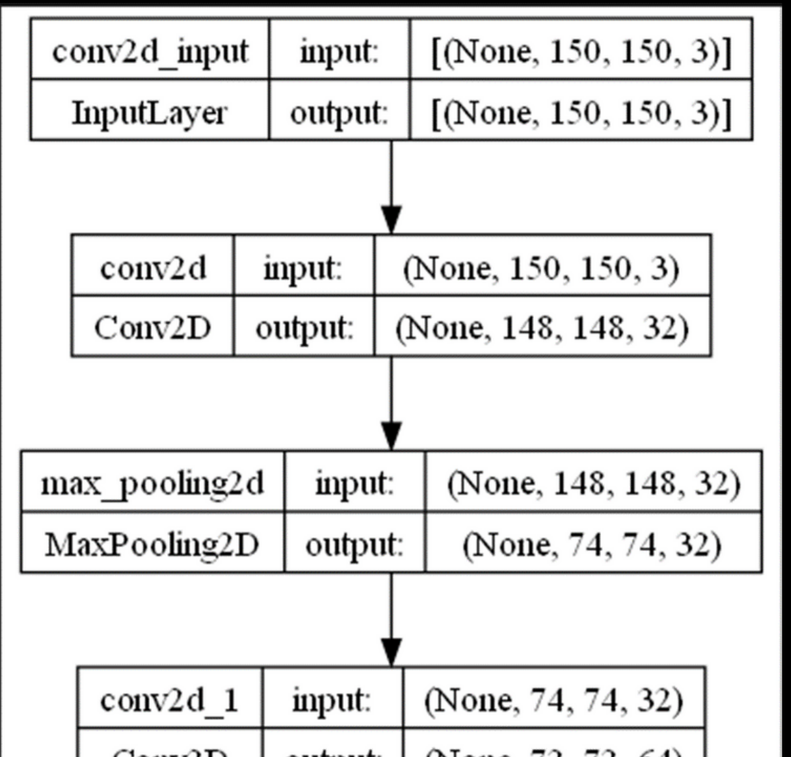
```python
import tensorflow as tf

img_height = 150
img_width = 150
input_shape = (img_height, img_width, 3)

model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=input_shape),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dropout(0.4),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(64, activation='relu'),  # Adding an extra layer for complexity
    tf.keras.layers.Dense(3, activation='softmax')  # 4 output classes
])

model.summary()
```

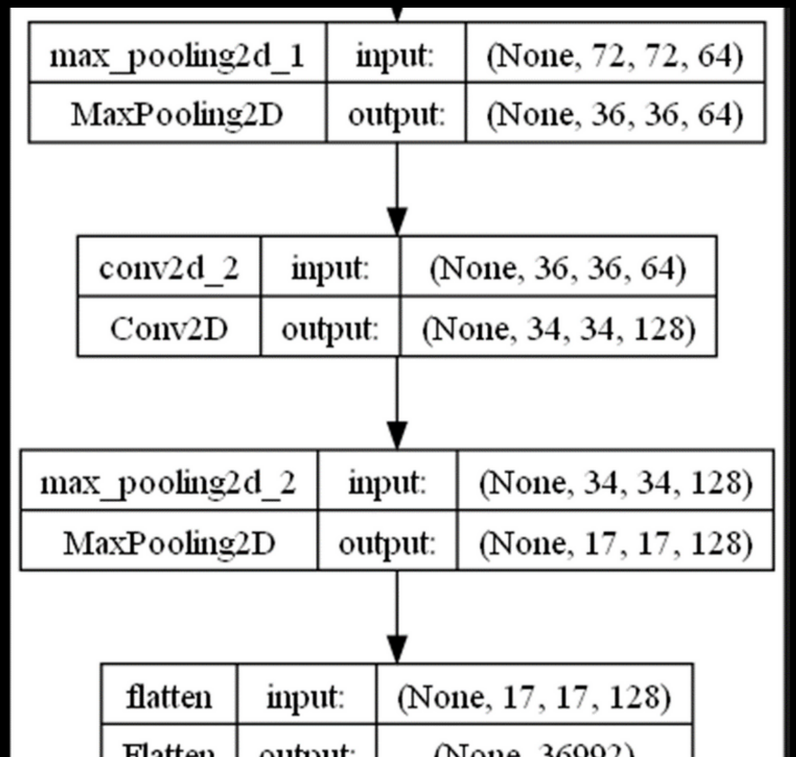
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 148, 148, 32)      896

 max_pooling2d (MaxPooling2  (None, 74, 74, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 72, 72, 64)        18496

 max_pooling2d_1 (MaxPoolin  (None, 36, 36, 64)        0
 g2D)

 conv2d_2 (Conv2D)           (None, 34, 34, 128)       73856

 max_pooling2d_2 (MaxPoolin  (None, 17, 17, 128)       0
 g2D)

 flatten (Flatten)           (None, 36992)             0

 dropout (Dropout)           (None, 36992)             0

 dense (Dense)               (None, 128)               4735104

...
Total params: 4836803 (18.45 MB)
Trainable params: 4836803 (18.45 MB)
Non-trainable params: 0 (0.00 Byte)
```

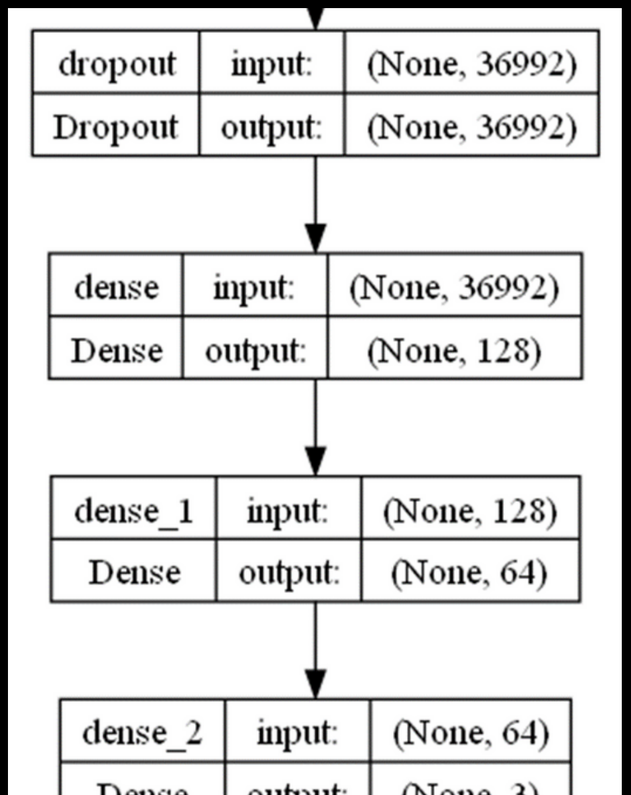
PEMBUATAN MODEL CNN

- **VISUALISASI MODEL CNN**

```
from tensorflow.keras.utils import plot_model
# Visualize the model
plot_model(model, to_file='model_plot.png', show_shapes=True, show_layer_names=True)
```







PEMBUATAN MODEL CNN

- COMPILE DAN MEMBUAT CALLBACK

```
# Compile model dengan optimizer Nadam, loss function 'categorical_crossentropy', dan metrik 'accuracy'
from tensorflow.keras.optimizers import Nadam

model.compile(loss='categorical_crossentropy',
              optimizer = Nadam(),
              metrics=['accuracy'])
```

```
# Membuat callback untuk menghentikan training apabila sudah mencapai akurasi diatas 90%

class TestCallback(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs={}):
        if(logs.get('accuracy') >= 0.90 and logs.get('val_accuracy') >= 0.90):
            print("\nAkurasi telah mencapai >=90%!")
            self.model.stop_training = True

# Definisikan callback untuk menyimpan model terbaik
checkpoint = ModelCheckpoint('model_tubesML_terbaik.h5', save_best_only=True)
callbacks = [TestCallback(), checkpoint]
```

CNN

- **TRAINING MODEL**

```python
import time
start_time = time.time()

history = model.fit(
    train_generator,
    validation_data = validation_generator,
    # validation_steps = 51, #  number_images/batch_size = 3292/64
    # steps_per_epoch = 33, #  number_images/batch_size= 2202/64
    epochs = 25,
    callbacks= callbacks,
)

end_time = time.time()
training_time = end_time - start_time
training_time_minute = training_time / 60
print(f"Total waktu training model Chest X-Ray: {training_time_minute}
```

```
Epoch 1/25
1046/1046 [==============================] - 440s 413ms/step - loss: 0.8122 - accuracy: 0.6006 - val_loss: 0.7032 - val_accuracy: 0.6791
Epoch 2/25
1046/1046 [==============================] - 386s 369ms/step - loss: 0.6177 - accuracy: 0.7388 - val_loss: 0.5630 - val_accuracy: 0.7526
Epoch 3/25
1046/1046 [==============================] - 383s 366ms/step - loss: 0.5113 - accuracy: 0.7972 - val_loss: 0.4556 - val_accuracy: 0.8290
Epoch 4/25
1046/1046 [==============================] - 390s 373ms/step - loss: 0.4565 - accuracy: 0.8244 - val_loss: 0.4252 - val_accuracy: 0.8386
Epoch 5/25
1046/1046 [==============================] - 389s 372ms/step - loss: 0.4201 - accuracy: 0.8450 - val_loss: 0.3658 - val_accuracy: 0.8586
Epoch 6/25
1046/1046 [==============================] - 410s 392ms/step - loss: 0.3868 - accuracy: 0.8582 - val_loss: 0.4377 - val_accuracy: 0.8443
Epoch 7/25
1046/1046 [==============================] - 390s 373ms/step - loss: 0.3841 - accuracy: 0.8577 - val_loss: 0.3783 - val_accuracy: 0.8720
Epoch 8/25
1046/1046 [==============================] - 395s 377ms/step - loss: 0.3563 - accuracy: 0.8744 - val_loss: 0.3229 - val_accuracy: 0.8902
Epoch 9/25
1046/1046 [==============================] - 426s 407ms/step - loss: 0.3443 - accuracy: 0.8754 - val_loss: 0.3128 - val_accuracy: 0.8911
Epoch 10/25
1046/1046 [==============================] - 424s 406ms/step - loss: 0.3393 - accuracy: 0.8847 - val_loss: 0.3074 - val_accuracy: 0.8940
Epoch 11/25
1046/1046 [==============================] - 373s 356ms/step - loss: 0.3250 - accuracy: 0.8859 - val_loss: 0.3149 - val_accuracy: 0.8892
Epoch 12/25
1046/1046 [==============================] - 359s 343ms/step - loss: 0.3211 - accuracy: 0.8862 - val_loss: 0.2979 - val_accuracy: 0.8978
Epoch 13/25
...
1046/1046 [==============================] - ETA: 0s - loss: 0.2940 - accuracy: 0.9003
Akurasi telah mencapai >=90%!
1046/1046 [==============================] - 355s 339ms/step - loss: 0.2940 - accuracy: 0.9003 - val_loss: 0.2607 - val_accuracy: 0.9160
Total waktu training model Chest X-Ray: 120.33022702534994 menit
```

CNN

- **HASIL TRAINING MODEL**

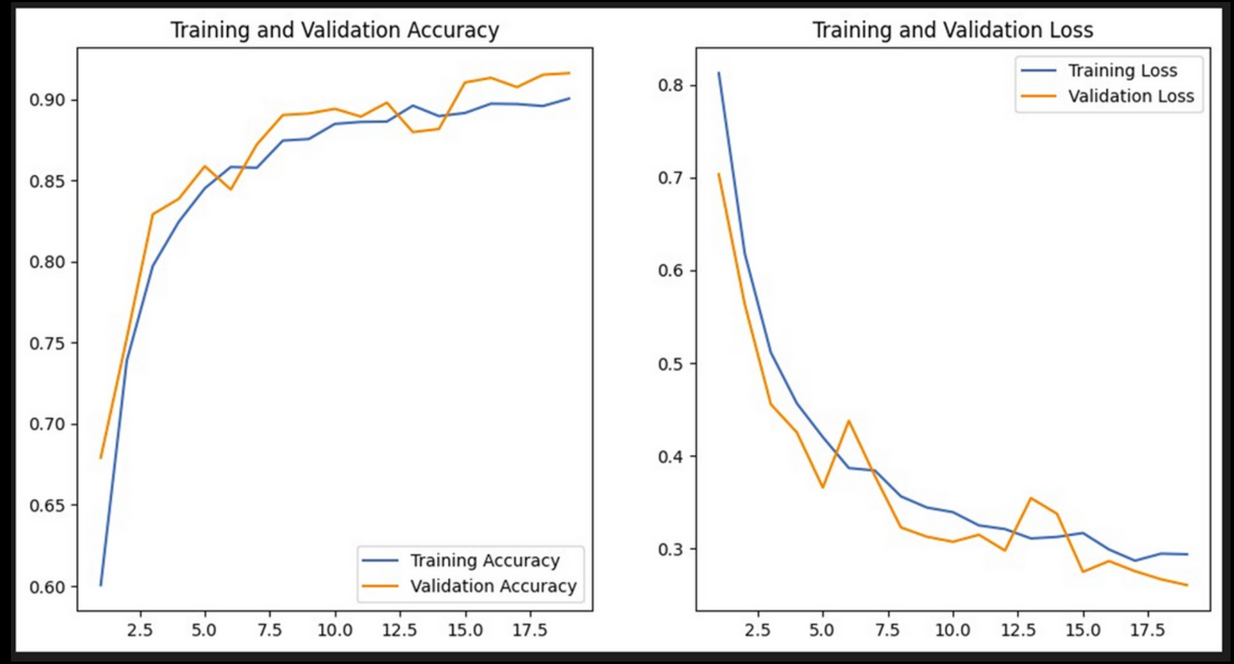
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

# Membuat dataframe
data = {
    'Epoch': range(1, len(acc) + 1),
    'Accuracy': acc,
    'Validation Accuracy': val_acc,
    'Loss': loss,
    'Validation Loss': val_loss
}

df = pd.DataFrame(data)

# Menampilkan dataframe
display(df)
```

	Epoch	Accuracy	Validation Accuracy	Loss	Validation Loss
0	1	0.600574	0.679083	0.812178	0.703197
1	2	0.738818	0.752627	0.617687	0.563041
2	3	0.797178	0.829035	0.511320	0.455583
3	4	0.824444	0.838586	0.456525	0.425226
4	5	0.845013	0.858644	0.420138	0.365769
5	6	0.858168	0.844317	0.386799	0.437743
6	7	0.857690	0.872015	0.384131	0.378332
7	8	0.874432	0.890162	0.356326	0.322896
8	9	0.875389	0.891117	0.344280	0.312781
9	10	0.884717	0.893983	0.339316	0.307377
10	11	0.885912	0.889207	0.325021	0.314945
11	12	0.886152	0.897803	0.321081	0.297935
12	13	0.895958	0.879656	0.311009	0.354438
13	14	0.889500	0.881566	0.312608	0.337618
14	15	0.891414	0.910220	0.316830	0.275008
15	16	0.897154	0.913085	0.299115	0.286617
16	17	0.896915	0.907354	0.286993	0.275728
17	18	0.895719	0.914995	0.294608	0.266980
18	19	0.900263	0.915950	0.293956	0.260706

CNN

- **EVALUASI MODEL : PLOT ACCURACY DAN VALIDATON**

```
epochs_range= range(1, len(acc) + 1)

plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```



CNN

- **EVALUASI MODEL : TEST LOSS DAN TEST ACCURACY**

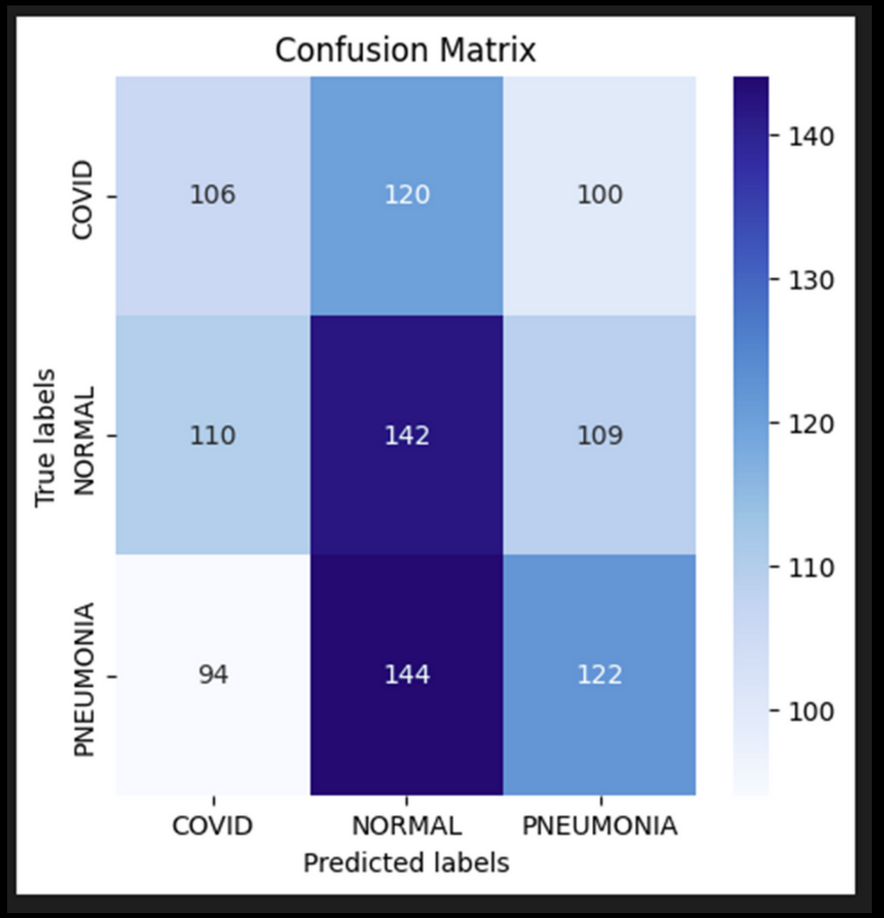
```
#evaluasi model
test_results = model.evaluate(validation_generator, verbose=0)
print(f'Test Loss     : {test_results[0]:.4f}')
print(f'Test Accuracy : {test_results[1]:.4f}')
```

```
Test Loss     : 0.2703
Test Accuracy : 0.9083
```

CNN

- **EVALUASI MODEL : CONFUSION MATRIX**

```python
from sklearn.metrics import confusion_matrix
import seaborn as sns

# Assuming you have a validation dataset and true labels
# Generate predictions for the validation dataset
predictions = model.predict(validation_generator)
predicted_classes = np.argmax(predictions, axis=1)

# Get true labels
true_classes = validation_generator.classes

# Calculate confusion matrix
conf_matrix = confusion_matrix(true_classes, predicted_classes)

# Plotting the confusion matrix as a heatmap
plt.figure(figsize=(5, 5))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
            xticklabels=class_indices, yticklabels=class_indices)
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix')
plt.show()
```



CNN

- **EVALUASI MODEL : CLASSIFICATION REPORT**

```python
# Get true labels for the validation dataset
true_labels = validation_generator.classes

# Make predictions on the validation dataset
predictions = model.predict(validation_generator)
predicted_labels = np.argmax(predictions, axis=1)

# Generate classification report
from sklearn.metrics import classification_report

class_report = classification_report(true_labels, predicted_labels)
print("Classification Report:")
print(class_report)
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.32      0.31      0.32       326
           1       0.32      0.37      0.34       361
           2       0.31      0.28      0.29       360

    accuracy                           0.32      1047
   macro avg       0.32      0.32      0.32      1047
weighted avg       0.32      0.32      0.32      1047
```

CNN

- **MENYIMPAN HASIL TRAINING MODEL**

```
from tensorflow.keras.models import load_model

# Untuk menyimpan model
model.save('model_tubesML.h5')
```

```
# Untuk memuat kembali model
model = load_model('model_tubesML_terbaik.h5')
```

CNN

- **IMPLEMENTASI MODEL**

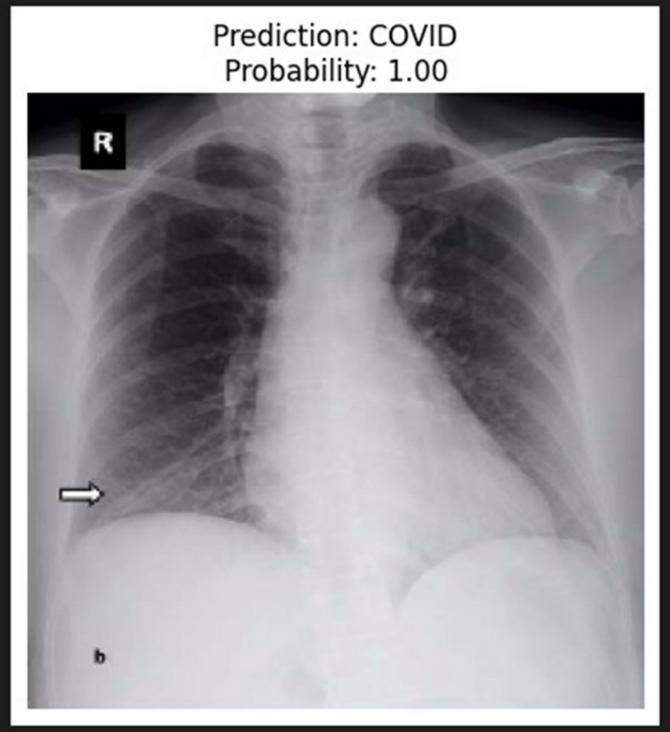
```
import tkinter as tk
from tkinter import filedialog
from tensorflow.keras.preprocessing import image
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

# Function to predict an uploaded image
def predict_uploaded_image(model):
    file_path = filedialog.askopenfilename()
    if file_path:
        predict_image(model, file_path)
        print("Image Path:", file_path)

# Function to predict an image given its path
def predict_image(model, image_path):
    labels = ['COVID', 'NORMAL', 'PNEUMONIA']

    img = image.load_img(image_path, target_size=(150, 150))
    img = image.img_to_array(img)
    img = img / 255.0

    img = np.expand_dims(img, axis=0)

    prediction = model.predict(img)
    print(prediction)
```

```
    predicted_class = labels[np.argmax(prediction)]
    predicted_probability = np.max(prediction)

    plt.figure()
    img = mpimg.imread(image_path)
    plt.imshow(img)
    plt.title(f'Prediction: {predicted_class}\nProbability: {predicted_probability:.2f}')
    plt.axis('off')
    plt.show()
    # Close the root window after displaying the image and prediction
    root.destroy()

# Create Tkinter GUI
root = tk.Tk()
root.title("Image Prediction")

# Function to trigger image upload and prediction
def on_button_click():
    predict_uploaded_image(model)  # Replace 'loaded_model' with your model variable

# Create a button to upload the image
button = tk.Button(root, text="Upload Image", command=on_button_click)
button.pack()

# Run the Tkinter main loop
root.mainloop()
```

CNN

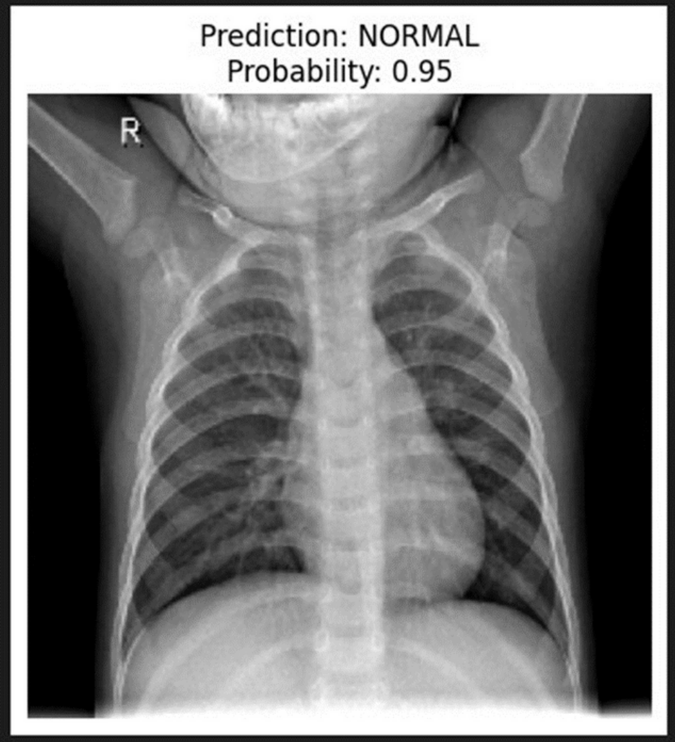
- **HASIL IMPLEMENTASI MODEL**

```
1/1 [==============================] - 0s 70ms/step
[[0.99510765 0.00139887 0.00349353]]
```



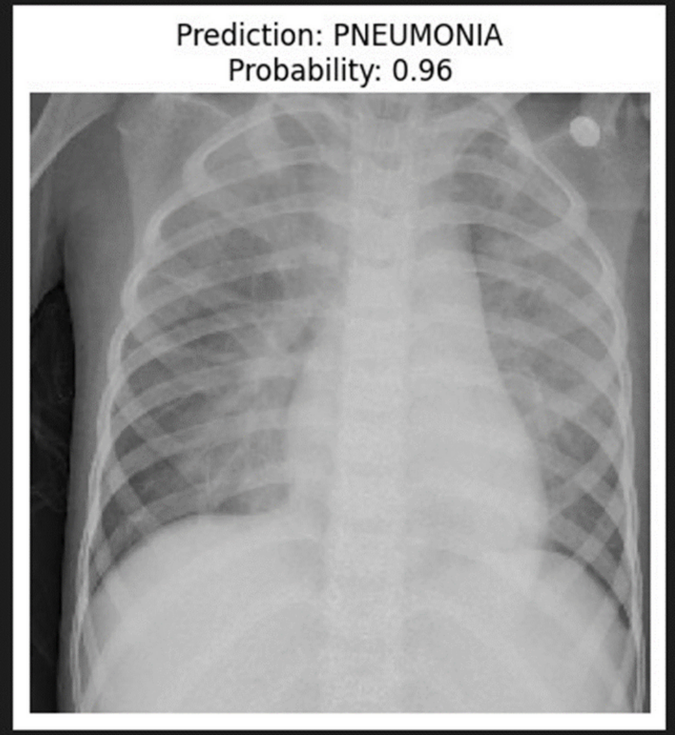
/validation/COVID/COVID_349.png

```
1/1 [==============================] - 0s 69ms/step
[[2.3959826e-04 9.5483929e-01 4.4921078e-02]]
```



/validation/NORMAL/NORMAL_691.png

```
1/1 [==============================] - 0s 105ms/step
[[6.263675e-04 3.941673e-02 9.599569e-01]]
```



/validation/PNEUMONIA/PNEUMONIA_467.png

PEMBUATAN MODEL NAS

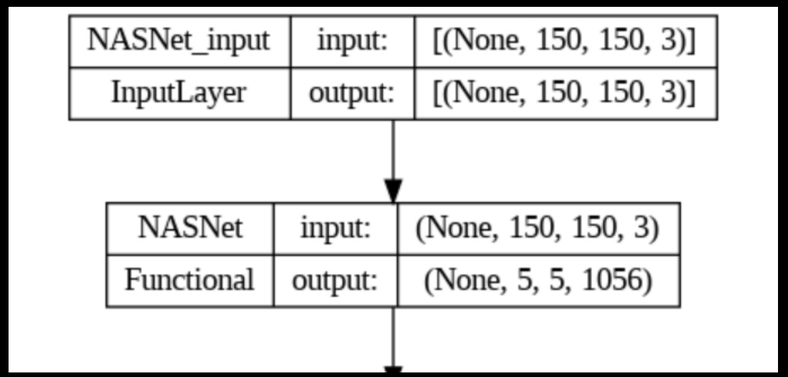
- PEMBUATAN MODEL NAS

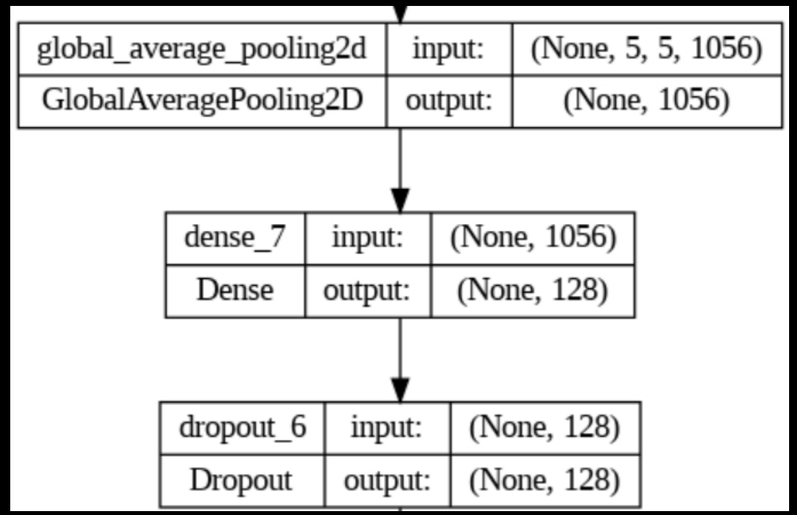
```
from keras.applications import NASNetMobile
from keras import layers
from keras import models

# Specify the input shape
img_height = 150
img_width = 150
input_shape = (img_height, img_width, 3)

# Load the NASNetMobile model pre-trained on ImageNet
base_model = NASNetMobile(input_shape=input_shape, include_top=False,
weights='imagenet')
# Freeze the layers of the pre-trained model
base_model.trainable = False

# Create a new model on top of the pre-trained model
model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.4),
    layers.Dense(64, activation='relu'),
    layers.Dense(3, activation='softmax')  # 3 output classes
])

model.summary()
```

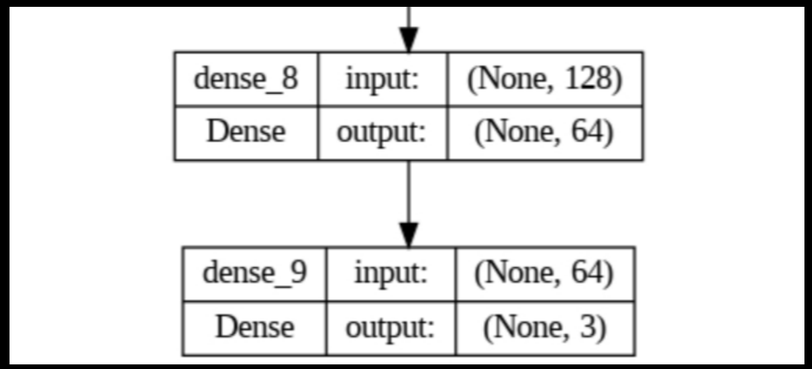
PEMBUATAN MODEL NAS

- **VISUALISASI MODEL CNN**

```python
# Save the model architecture to a file
plot_model(model, to_file='nasnet_model.png', show_shapes=True, show_layer_names=True)

# Display the model architecture image
Image(filename='nasnet_model.png')
```







PEMBUATAN MODEL NAS

- COMPILE DAN MEMBUAT CALLBACK

```
from keras.optimizers import Nadam
import tensorflow as tf

# Compile the model with Nadam optimizer, categorical crossentropy loss, and accuracy metric
model.compile(loss='categorical_crossentropy',
              optimizer=Nadam(),
              metrics=['accuracy'])

# Create a callback to stop training if accuracy is >= 90%
class TestCallback(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs={}):
        if logs.get('accuracy') >= 0.90 and logs.get('val_accuracy') >= 0.90:
            print("\nAkurasi telah mencapai >=90%!")
            self.model.stop_training = True

# Define a callback to save the best model
checkpoint = tf.keras.callbacks.ModelCheckpoint('model_terbaik.h5', save_best_only=True)
callbacks = [TestCallback(), checkpoint]
```

NAS

- **TRAINING MODEL**

```python
import time

start_time = time.time()

history = model.fit(
    train_generator,
    validation_data=validation_generator,
    epochs=25,
    callbacks=callbacks,
)

end_time = time.t
training_time = (
training_time_mi
print(f"Total wal
```

```
1105/1105 [==============================] - 345s 312ms/step - loss: 0.2891 - accuracy: 0.8937 - val_loss: 0.2851 - val_accuracy: 0.8980
Epoch 14/25
1105/1105 [==============================] - 343s 310ms/step - loss: 0.2845 - accuracy: 0.8914 - val_loss: 0.2479 - val_accuracy: 0.9033
Epoch 15/25
1105/1105 [==============================] - 341s 308ms/step - loss: 0.2855 - accuracy: 0.8937 - val_loss: 0.2913 - val_accuracy: 0.8971
Epoch 16/25
1105/1105 [==============================] - 338s 306ms/step - loss: 0.3016 - accuracy: 0.8896 - val_loss: 0.2816 - val_accuracy: 0.8883
Epoch 17/25
1105/1105 [==============================] - ETA: 0s - loss: 0.2918 - accuracy: 0.9007
Akurasi telah mencapai >=90%!
1105/1105 [==============================] - 339s 306ms/step - loss: 0.2918 - accuracy: 0.9007 - val_loss: 0.2573 - val_accuracy: 0.9050
Total waktu training model NASNetMobile: 102.7611186226209 menit
```

NAS

- **HASIL TRAINING MODEL**

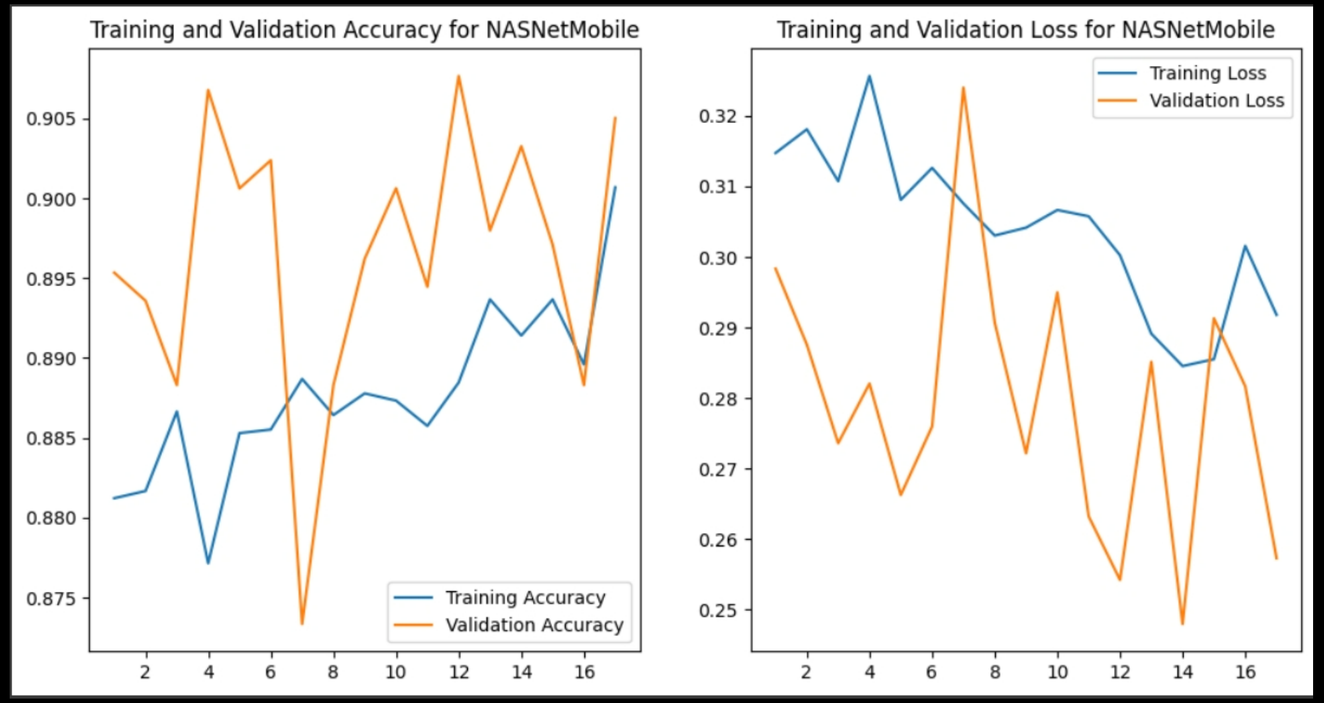
```
import pandas as pd

acc = history.history['accuracy']
val_acc =
history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

# Create a dataframe
data = {
    'Epoch': range(1, len(acc) + 1),
    'Accuracy': acc,
    'Validation Accuracy': val_acc,
    'Loss': loss,
    'Validation Loss': val_loss
}

df = pd.DataFrame(data)

# Display the dataframe
display(df)
```

	Epoch	Accuracy	Validation Accuracy	Loss	Validation Loss
0	1	0.881222	0.895339	0.314735	0.298337
1	2	0.881674	0.893580	0.318095	0.287555
2	3	0.886652	0.888303	0.310739	0.273586
3	4	0.877149	0.906772	0.325660	0.282032
4	5	0.885294	0.900616	0.308089	0.266234
5	6	0.885520	0.902375	0.312621	0.275947
6	7	0.888688	0.873351	0.307570	0.324010
7	8	0.886425	0.888303	0.303030	0.290680
8	9	0.887783	0.896218	0.304146	0.272143
9	10	0.887330	0.900616	0.306654	0.294960
10	11	0.885747	0.894459	0.305760	0.263206
11	12	0.888462	0.907652	0.300239	0.254196
12	13	0.893665	0.897977	0.289113	0.285114
13	14	0.891403	0.903254	0.284517	0.247946
14	15	0.893665	0.897098	0.285493	0.291294
15	16	0.889593	0.888303	0.301559	0.281650
16	17	0.900679	0.905013	0.291783	0.257272

NAS

- **EVALUASI MODEL : PLOT ACCURACY DAN VALIDATON**

```
import matplotlib.pyplot as plt

epochs_range = range(1, len(acc) + 1)

plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy for
NASNetMobile')
plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss for NASNetMobile')
plt.show()
```



NAS

- EVALUASI MODEL : TEST LOSS DAN TEST ACCURACY

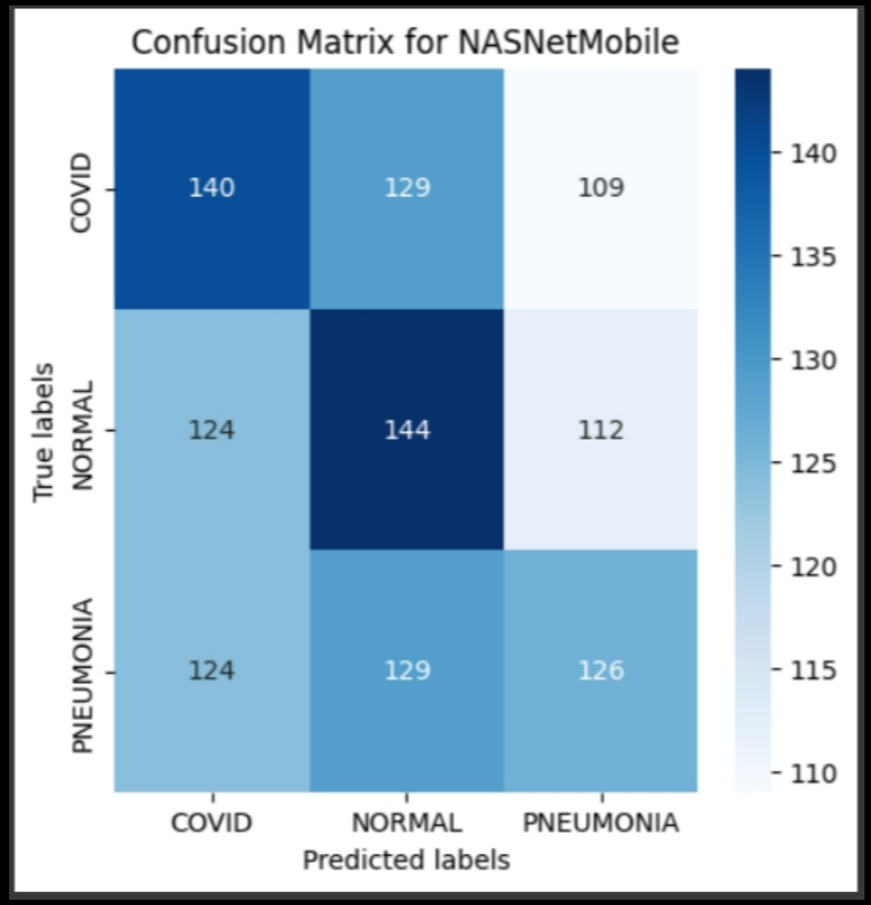
```
#evaluasi model
test_results = model.evaluate(validation_generator, verbose=0)
print(f'Test Loss      : {test_results[0]:.4f}')
print(f'Test Accuracy : {test_results[1]:.4f}')
```

```
Test Loss      : 0.2690
Test Accuracy : 0.9006
```

NAS

- **EVALUASI MODEL : CONFUSION MATRIX**

```python
from sklearn.metrics import confusion_matrix
import seaborn as sns

# Assuming you have a validation dataset and true labels
# Generate predictions for the validation dataset
predictions = model.predict(validation_generator)
predicted_classes = np.argmax(predictions, axis=1)

# Get true labels
true_classes = validation_generator.classes

# Calculate confusion matrix
conf_matrix = confusion_matrix(true_classes, predicted_classes)

# Plotting the confusion matrix as a heatmap
plt.figure(figsize=(5, 5))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
            xticklabels=class_indices, yticklabels=class_indices)
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix for NASNetMobile')
plt.show()
```



NAS

- **EVALUASI MODEL : CLASSIFICATION REPORT**

```python
# Get true labels for the validation dataset
true_labels = validation_generator.classes

# Make predictions on the validation dataset
predictions = model.predict(validation_generator)
predicted_labels = np.argmax(predictions, axis=1)

# Generate classification report
from sklearn.metrics import classification_report

class_report = classification_report(true_labels, predicted_labels)
print("Classification Report for NASNetMobile:")
print(class_report)
```

```
Classification Report for NASNetMobile:
              precision    recall  f1-score   support

           0       0.33      0.34      0.33       378
           1       0.34      0.36      0.35       380
           2       0.32      0.29      0.30       379

    accuracy                           0.33      1137
   macro avg       0.33      0.33      0.33      1137
weighted avg       0.33      0.33      0.33      1137
```

NAS

- **MENYIMPAN HASIL TRAINING MODEL**

```
from tensorflow.keras.models import load_model

# Untuk menyimpan model
model.save('model.h5')


# Untuk memuat kembali model
model = load_model('model_terbaik.h5')
```

NAS

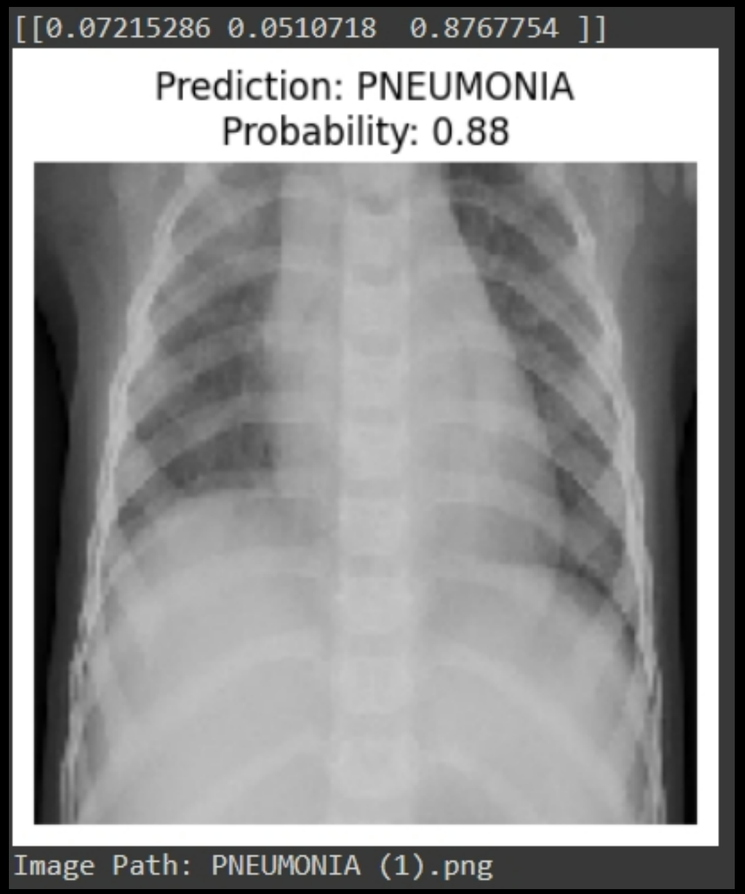
- **IMPLEMENTASI MODEL**

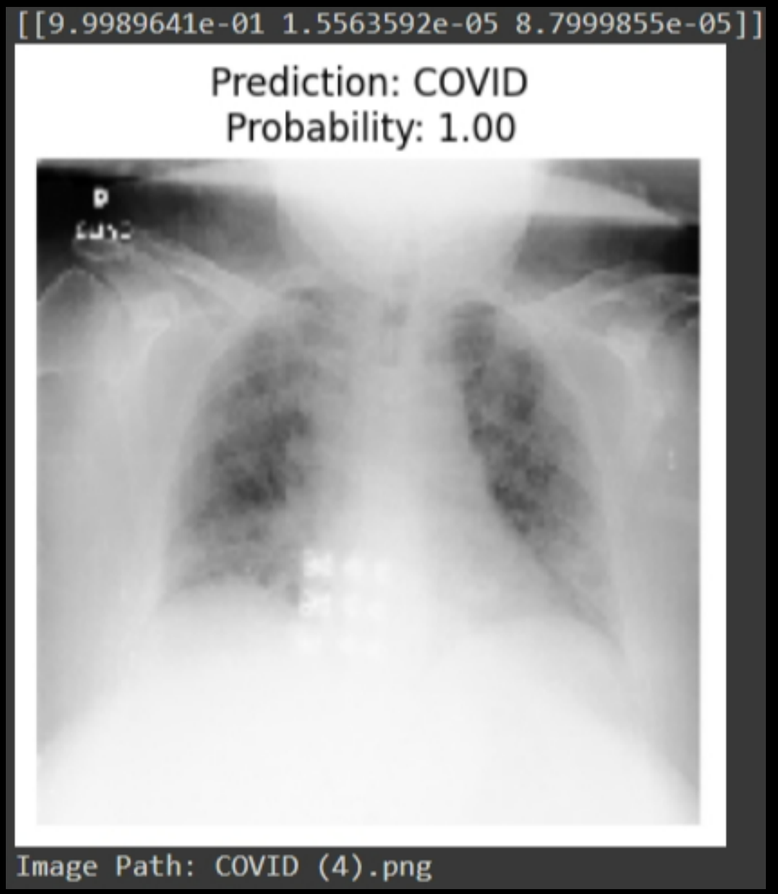
```python
from google.colab import files
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import load_model
import numpy as np

# Function to predict an image given its path
def predict_image(model, image_path):
    labels = ['COVID', 'NORMAL', 'PNEUMONIA']

    img = image.load_img(image_path, target_size=(150, 150))
    img = image.img_to_array(img)
    img = img / 255.0

    img = np.expand_dims(img, axis=0)

    prediction = model.predict(img)
    print(prediction)

    predicted_class = labels[np.argmax(prediction)]
    predicted_probability = np.max(prediction)

    # Display the uploaded image with prediction and probability
    plt.figure(figsize=(4, 4))
    plt.imshow(img[0])  # img[0] to access the image array
    plt.axis('off')  # Sembunyikan sumbu
    plt.title(f'Prediction: {predicted_class}\nProbability:
{predicted_probability:.2f}')
    plt.show()
```

```python
# Function to predict an uploaded image
def predict_uploaded_image(model):
    uploaded = files.upload()
    if uploaded:
        file_path = next(iter(uploaded))
        predict_image(model, file_path)
        print("Image Path:", file_path)

# Load your model (replace 'model_terbaik.h5' with the actual path)
model_path = 'model_terbaik.h5'
model = load_model(model_path)

# Trigger image upload and prediction
predict_uploaded_image(model)
```

NAS

- **HASIL IMPLEMENTASI MODEL**





PEMBUATAN MODEL RESNET

- PEMBUATAN MODEL RESNET

```python
from tensorflow.keras.layers import Input
from tensorflow.keras.regularizers import l2
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Activation
from tensorflow.keras.layers import AveragePooling2D
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Model

def AttentionResNet92(shape, in_channel, kernel_size, n_classes, dropout=None, regularization=0.

    input_data = Input(shape=shape)  # 32x32x3
    x = Conv2D(in_channel, kernel_size=kernel_size, padding='same')(input_data)  # 32x32x32
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = MaxPooling2D(pool_size=2, padding='same')(x)  # 16x16x32

    out_channel = in_channel * 4  # 256
    x = Residual_Unit(x, in_channel, out_channel)  # 16x16x128
    x = Attention_Block(x, skip=2)

    in_channel = out_channel  # 64
    out_channel = in_channel * 4  # 256
    x = Residual_Unit(x, in_channel, out_channel, stride=2)  # 8x8x256
    x = Attention_Block(x, skip=1)
```

```python
    in_channel = out_channel  # 256
    out_channel = in_channel * 4  # 1024
    x = Residual_Unit(x, in_channel, out_channel, stride=1)  # 4x4x1024
    x = Residual_Unit(x, in_channel, out_channel)
    x = Residual_Unit(x, in_channel, out_channel)

    x = AveragePooling2D(pool_size=4, strides=1)(x)  # 1x1x1024
    x = Flatten()(x)

    if dropout:
        x = Dropout(dropout)(x)

    output = Dense(n_classes, kernel_regularizer=l2(regularization), activation='sigmoid')(x)
    model = Model(input_data, output)

    return model
```

PEMBUATAN MODEL RESNET

- VISUALISASI MODEL RESNET

```
[116] train_data.shape

      (4234, 2)

[117] model = AttentionResNet92((64,64,1),3,3,3)

[118] optim = tf.keras.optimizers.Adam (learning_rate=1e-4)
      # optim=tf.keras.optimizers.SGD(learning_rate=0.0001, momentum=0.9, nesterov=True)
      model.compile(optimizer=optim, loss='categorical_crossentropy', metrics=['accuracy'])

      model.summary()
```

PEMBUATAN MODEL RESNET

- COMPILE DAN MEMBUAT CALLBACK

```python
# Membuat callback untuk menghentikan training apabila sudah mencapai akurasi diatas 90%

class TestCallback(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs={}):
        if(logs.get('accuracy') >= 0.90 and logs.get('val_accuracy') >= 0.90):
            print("\nAkurasi telah mencapai >=90%!")
            self.model.stop_training = True

# Definisikan callback untuk menyimpan model terbaik
checkpoint = ModelCheckpoint('model_tubesML_terbaik.h5', save_best_only=True)
callbacks = [TestCallback(), checkpoint]
```

RESNET

- **TRAINING MODEL**

```
init_time = datetime.datetime.now()

train_steps = train_gen.samples // BATCH_SIZE
valid_steps = val_gen.samples // BATCH_SIZE

early_stopping = EarlyStopping(monitor="val_loss", patience=15, mode="min")
checkpoint = ModelCheckpoint("./min_loss.h5", monitor="val_loss", verbose=1,
                             save_best_only=True, save_weights_only=True, mode="min")
learning_rate_reduction = ReduceLROnPlateau(monitor="val_loss", factor=0.1, patience=4,
                                            min_lr=1e-7, verbose=1, mode="min")
csvlogger=tf.keras.callbacks.CSVLogger('./history.csv')

history = model.fit(
    train_gen,
    validation_data=val_gen,
    batch_size=BATCH_SIZE,
    epochs=EPOCHS,
    steps_per_epoch=train_steps,
    validation_steps=valid_steps,
    callbacks=[ callbacks,
                csvlogger,
                checkpoint,
                early_stopping,
                learning_rate_reduction],
    verbose=1,
    class_weight = class_weights
    )

requared_time = datetime.datetime.now() - init_time
print(f'\nRequired time:  {str(requared_time)}\n')
```

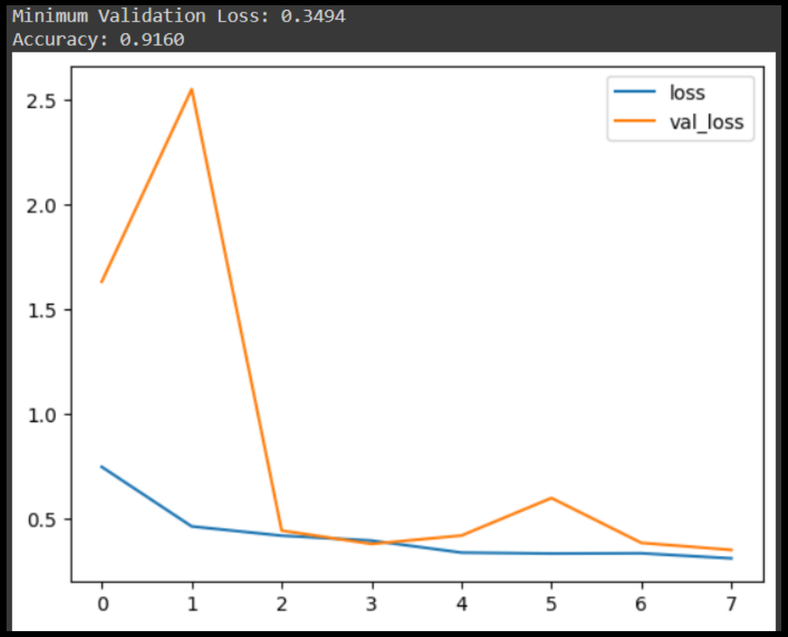
```
Epoch 5: val_loss did not improve from 0.37787
132/132 [==============================] - 501s 4s/step - loss: 0.3364 - accuracy: 0.9029 - val_loss: 0.4181 - val_accuracy: 0.8945 - lr: 1.0000e
Epoch 6/25
132/132 [==============================] - ETA: 0s - loss: 0.3318 - accuracy: 0.8986
Epoch 6: val_loss did not improve from 0.37787
132/132 [==============================] - 518s 4s/step - loss: 0.3318 - accuracy: 0.8986 - val_loss: 0.5974 - val_accuracy: 0.8457 - lr: 1.0000e
Epoch 7/25
132/132 [==============================] - ETA: 0s - loss: 0.3332 - accuracy: 0.8991
Epoch 7: val_loss did not improve from 0.37787
132/132 [==============================] - 522s 4s/step - loss: 0.3332 - accuracy: 0.8991 - val_loss: 0.3828 - val_accuracy: 0.8945 - lr: 1.0000e
Epoch 8/25
132/132 [==============================] - ETA: 0s - loss: 0.3089 - accuracy: 0.9060
Akurasi telah mencapai >=90%!

Epoch 8: val_loss improved from 0.37787 to 0.34942, saving model to ./min_loss.h5
132/132 [==============================] - 538s 4s/step - loss: 0.3089 - accuracy: 0.9060 - val_loss: 0.3494 - val_accuracy: 0.9160 - lr: 1.0000e

Required time:  1:26:37.893914
```

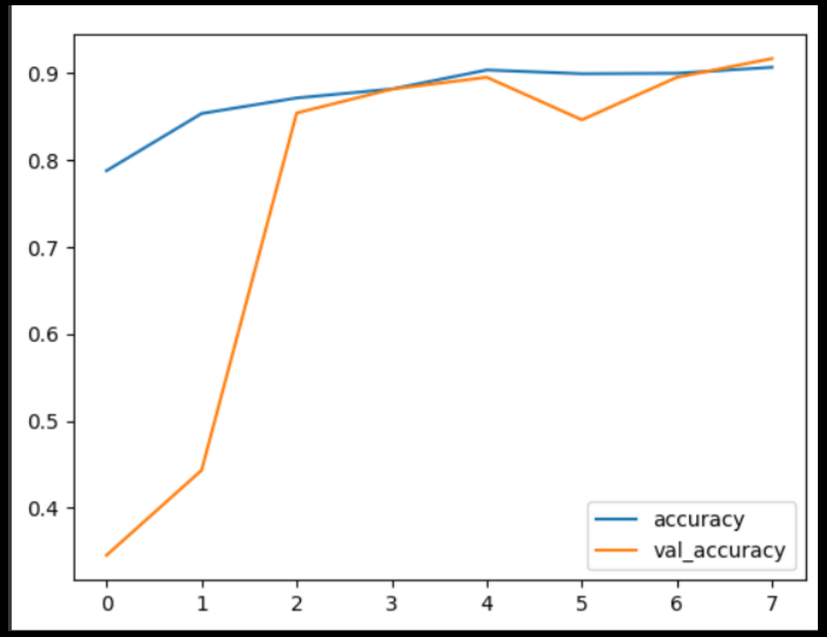
RESNET

- EVALUASI MODEL : PLOT ACCURACY DAN VALIDATON

```python
import pandas as pd
# import matplotlib.pyplot as plt

history=pd.read_csv('./history.csv')

# history_df = history.history
history.loc[0:, ['loss', 'val_loss']].plot()
plt.savefig("loss_val_loss.png")
print("Minimum Validation Loss: {:0.4f}".format(history['val_loss'].min()));

# history_df = pd.DataFrame(history.history)
history.loc[0:, ['accuracy', 'val_accuracy']].plot()
plt.savefig("acc_val_acc.png")
print("Accuracy: {:0.4f}".format(history['val_accuracy'].max()));
```

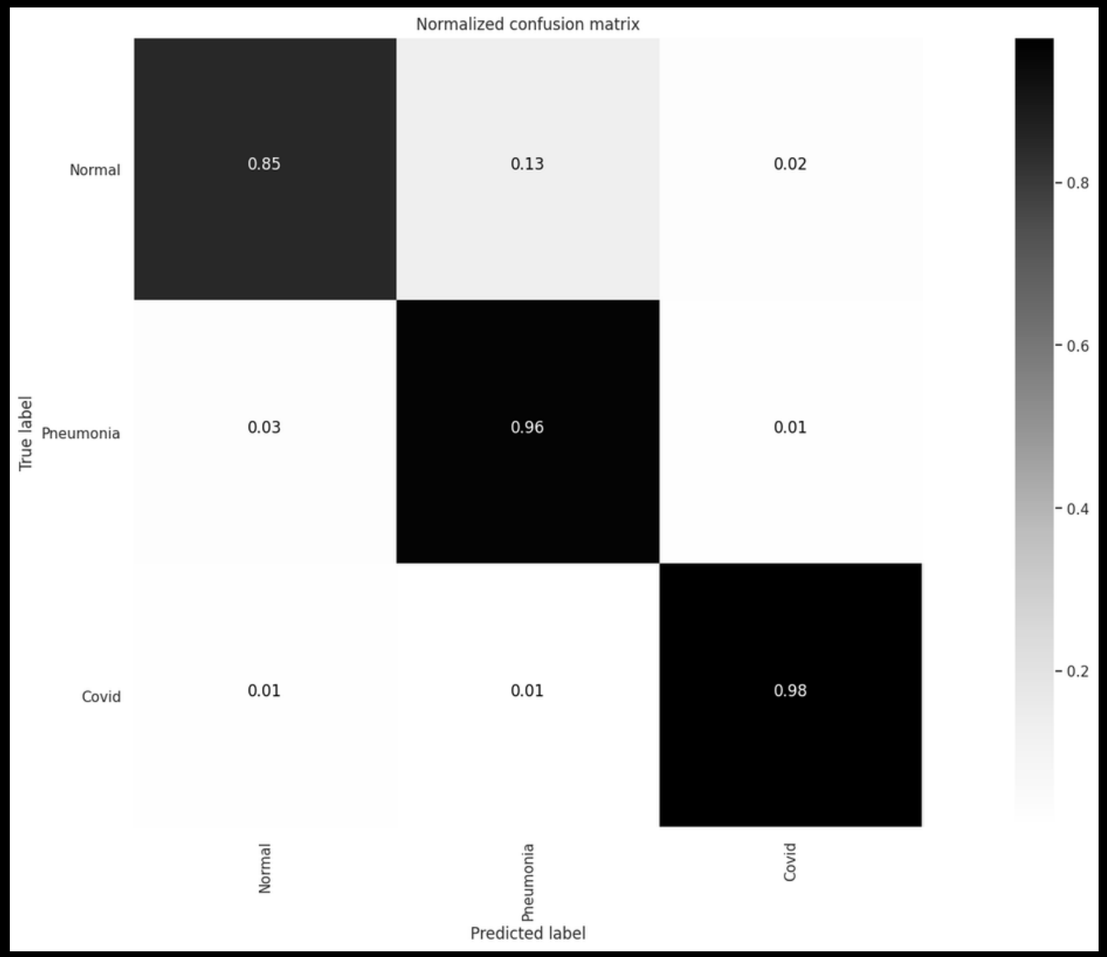
```
Minimum Validation Loss: 0.3494
Accuracy: 0.9160
```





RESNET

- **EVALUASI MODEL : CONFUSION MATRIX**



RESNET

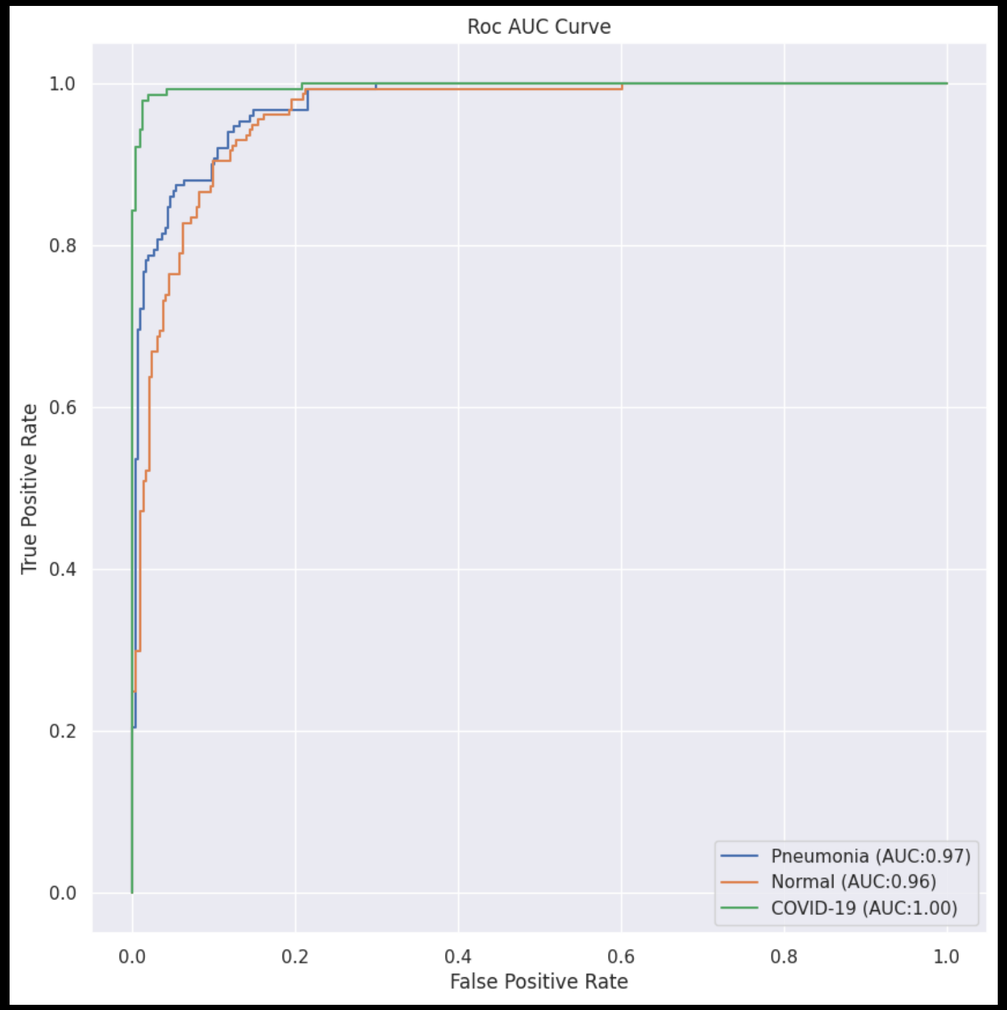
- **EVALUASI MODEL : CLASSIFICATION REPORT**

```
from sklearn.metrics import classification_report

print(classification_report(y_test, y_hat))
```

```
              precision    recall  f1-score   support

           0       0.96      0.85      0.90       151
           1       0.88      0.96      0.92       157
           2       0.96      0.98      0.97       140

    accuracy                           0.93       448
   macro avg       0.93      0.93      0.93       448
weighted avg       0.93      0.93      0.93       448
```

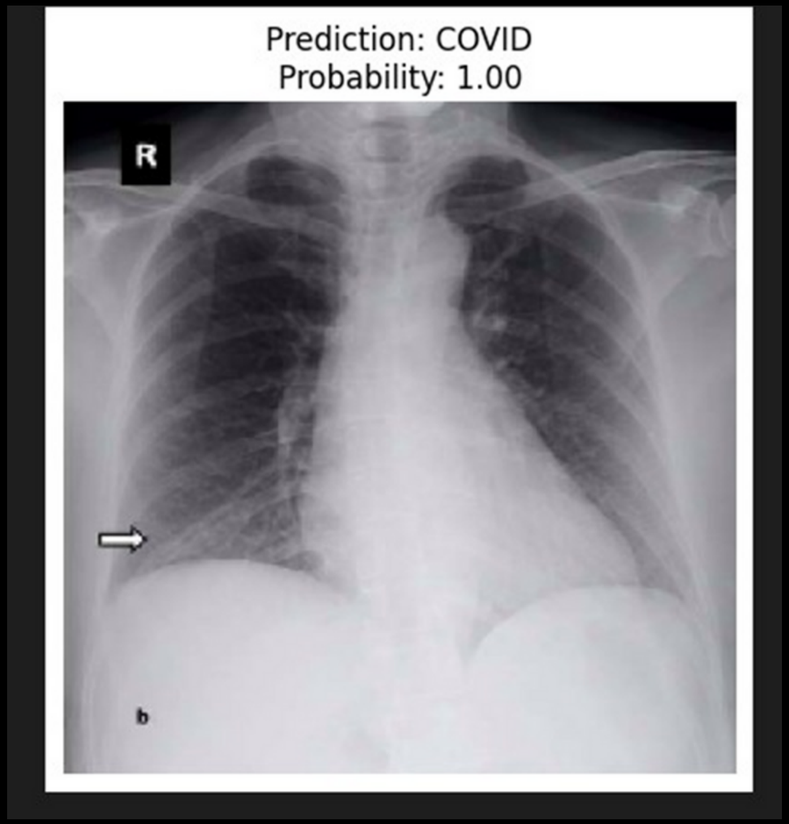
RESNET

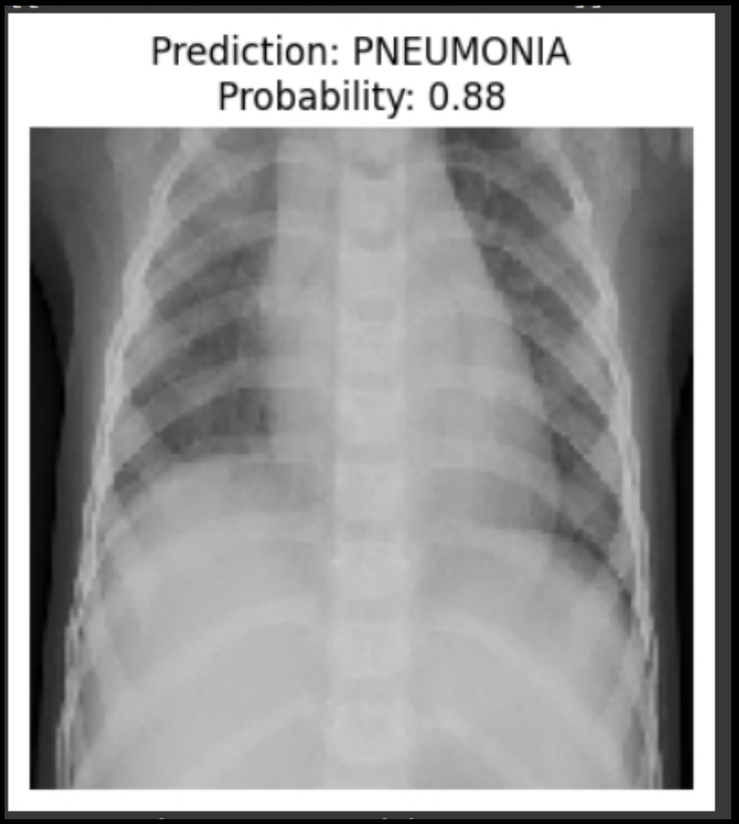
- **EVALUASI MODEL : ROC AUC CURVE**



RESNET

- **HASIL IMPLEMENTASI MODEL**





TERIMA KASIH