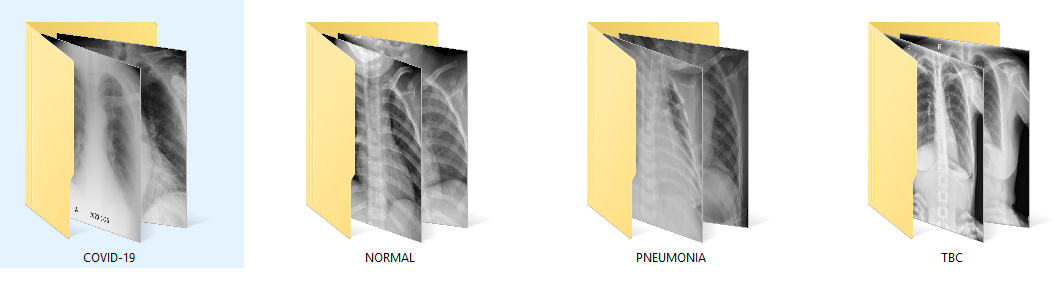
**BAB IV**

**IMPLEMENTASI DAN PENGUJIAN**

Pada penelitian ini, peneliti melakukan klasifikasi gambar hasil Chest X-Ray dengan kelas label Normal, COVID-19, Tuberculosis(TBC), dan Pneumonia dengan menggunakan *Convolutional Neural Network* (CNN) bahasa pemrograman python. Proses utama dalam pembuatan model klasifikasi metode CNN adalah perancangan klasifikasi. Prinsip dasar dari proses ini adalah dengan melakukan pelatihan pada CNN untuk menentukan model yang paling bagus dalam hal akurasi yang baik. Model yang digunakan untuk proses pelatihan adalah VGG19, ResNet50 dan InceptionV3. Dalam pelatihan CNN terdapat dataset untuk *train* (pelatihan)*, validation* (validasi)dan *test* (Pengujian). Data *train* digunakan untuk melakukan *training* atau pelatihanpada model-model yang telah ditentikan, dataset *validation* merupakan data yang digunakan untuk validasi data training, dan dataset *test* digunakan untuk menilai kinerja pengklasifikasi yang telah melalui tahapan training. Akurasi model-model tersebut dapat ditentukan dengan melakukan validasi dengan menggunakan data *test* atau data uji.

* 1. **Dataset**

Pada penelitian ini, dataset yang telah dikumpulkan dari berbagai sumber dengan total 1.000 gambar Chest X-Ray atau Foto Rontgent dada dengan format gambar PNG dan JPEG. Dataset ini dibagi menjadi 4 folder label sesuai dengan perancangan dataset yaitu kelas Normal, COVID-19, TBC dan Pneumonia dimana masing-masing kelas berisi 250 gambar. Gambar merupakan tampilan dataset yang digunakan pada penelitian ini, dataset tersebut akan dataset training dan validation. Dimana dataset validation merupuakan hasil split dari data train dengan berbagai rasio untuk pengujian model terbaik, yaitu 80% training : 20% validation, 50% training : 50% validation, dan 20% training : 80% validation.



Gambar 4. 1. Folder Subclass Dataset

* 1. **Preprocessing Dataset Convolution Neural Network (CNN)**

Proses selanjutnya setelah dataset disiapkan adalah pre-process dataset tersebut dengan beberapa teknik. Motivasi pada image pre-processing untuk memperbaiki dan meningkatkan kualitas informasi visual gambar input. Pada penelitian ini, untuk melakukan preprocessing image harus memanggil modul-modul yang akan digunakan, dan menentukan variabel global gambar seperti pada tabel dan tabel.

Tabel 4. 1. Load Modul Python

|  |
| --- |
| **import matplotlib.pyplot as plt**  **import tensorflow.keras as tfk**  **import numpy as np**  **import pandas as pd**  **import os**  **import cv2**  **from tensorflow.keras.preprocessing.image import ImageDataGenerator, array\_to\_img, img\_to\_array, load\_img**  **from tensorflow.keras.models import Sequential**  **from tensorflow.keras.layers import Flatten**  **from tensorflow.keras.layers import Dense**  **from tensorflow.keras.layers import Dropout**  **from tensorflow.keras import optimizers**  **from tensorflow.keras import backend as K**  **from tensorflow.keras.models import load\_model**  **from tensorflow.keras.applications.vgg19 import VGG19**  **from tensorflow.keras.applications.vgg19 import preprocess\_input**  **from tensorflow.keras.callbacks import ModelCheckpoint**  **from keras.utils.vis\_utils import plot\_model**  **import tensorflow as tf**  **from tensorflow.keras.applications.resnet50 import ResNet50** |

Perintah perintah pada tabel diatas merupakan perintah untuk mengimport modul-modul python yang akan digunakan, sehingga modul-modul tersbut siap digunakan. Dataset pipeline merupakan perintah untuk melakukan ekstraksi data yang berupa images/citra digital, yang berasal dari sebuah folder menjadi sebuah array yang dapat dibaca oleh tensorflow, fungsi yang digunakan adalah from tensorflow.keras.preprocessing.image import ImageDataGenerator.

Fungsi tersebut merupakan perintah untuk mengimport Image data generator sehingga kita mengenerate data image dari sebuah folder yang telah kita buat sebelumnya. pada bagian ini kita dapat menentukan generator preprocessing seperti apa yang dapat dilakukan seperti pada tabel.

Tabel 4. 2. Tabel Prepocessing Dataset

|  |
| --- |
| **img\_height = 256**  **img\_width = 256**  **channels = 3**  **batch\_size = 8**  **epochs = 20**  **train\_datagen = ImageDataGenerator(rescale=1. / 255,**  **shear\_range=0.2,**  **zoom\_range=0.2,**  **horizontal\_flip=True,**  **validation\_split=0.2)**  **train\_set = train\_datagen.flow\_from\_directory(**  **'./dataset/train',**  **target\_size = (img\_height, img\_width),**  **batch\_size = batch\_size,**  **class\_mode = 'categorical',**  **subset='training',**  **shuffle=True)**  **val\_set = train\_datagen.flow\_from\_directory(**  **'./dataset/train',**  **target\_size = (img\_height, img\_width),**  **batch\_size = batch\_size,**  **class\_mode = 'categorical',**  **subset='validation',**  **shuffle=True)** |



Gambar 4. 2. Output Prepocessing Dataset

Untuk split data, pada penelitian ini saya menggunakan rasion 80% training : 20% validation, 50% training : 50% validation, dan 20% training : 80% validation. Dataset yang telah displit menjadi data train dan data validation dengan rasio yang telah ditentukan tersebut akan digunakan pada proses training. Untuk memastikan kelas dataset sudah sesuai dengan nama folder maka dijalankan perintah berikut :

Tabel 4. 3. Pemanggilan Label Dataset

|  |
| --- |
| **num\_class = train\_set.num\_classes**  **labels = train\_set.class\_indices.keys()**  **print(labels)** |



Gambar 4. 3. Output Prepocessing Dataset

Berikut adalah tabel penejelasan teknik preprocessing dataset pada tabel 4.4 :

Tabel 4. 4. Preprocessing yang digunakan

|  |  |  |
| --- | --- | --- |
| **Argument** | **Parameter value** | **Description** |
| Rescale | 1 / 255.0 | Scale images from integers 0-255 to floats 0-1 |
| Shear range | 0.2 | Controls the angle in counterclockwise direction as radians in which our image will allowed to be sheared |
| Zoom range | 0.2 | Allows the image to be “zoomed out” or “zoomed in” |
| Horizontal flip | True | Controls when a given input is allowed to be flipped horizontally during the training process |

* 1. **Implementasi Model CNN**

Setelah melakukan preprocessing pada dataset, tahap selanjutnya adalah menentukan model yang digunakan untuk proses training. Pada penelitian ini menggunakan beberapa model dari CNN, yaitu VGG19, ResNet50, dan InceptionV3. Model tersebut digunakan sebagai model input, sedangkan model output sebagai prediksi ditentukan sendiri dari masing-masing model input tersbut seperti tabel berikut. Dikarenakan jumlah kelas pada imagenet adalah 1000 maka ini tidak cocok dengan jumlah kelas yang kita miliki yaitu hanya 4. Maka pada penelitian menggunakan beberapa layer sebagai model output.

Tabel 4. 5. Implementasi model VGG19

|  |
| --- |
| **model = VGG19(weights = "imagenet", include\_top=False, input\_shape = (img\_width, img\_height, channels), classes=num\_class)**  **for layer in model.layers[:-5]:**  **layer.trainable = False**  **headModel = model.output**  **headModel = AveragePooling2D(pool\_size=(4, 4))(headModel)**  **headModel = Flatten(name="flatten")(headModel)**  **headModel = Dense(64, activation="relu")(headModel)**  **headModel = Dropout(0.5)(headModel)**  **headModel = Dense(num\_class, activation="softmax")(headModel)**  **model = Model(inputs=model.input, outputs=headModel)** |

Tabel 4. 6. Implementasi model ResNet50

|  |
| --- |
| **model = ResNet50(weights = "imagenet", include\_top=False, input\_shape = (img\_width, img\_height, channels), classes=num\_class)**  **for layer in model.layers[:-5]:**  **layer.trainable = False**  **headModel = model.output**  **headModel = AveragePooling2D(pool\_size=(4, 4))(headModel)**  **headModel = Flatten(name="flatten")(headModel)**  **headModel = Dense(64, activation="relu")(headModel)**  **headModel = Dropout(0.5)(headModel)**  **headModel = Dense(num\_class, activation="softmax")(headModel)**  **model = Model(inputs=model.input, outputs=headModel)** |

Tabel 4. 7. Implementasi model InceptionV3

|  |
| --- |
| **model = InceptionV3(weights = "imagenet", include\_top=False, input\_shape = (img\_width, img\_height, channels), classes=num\_class)**  **for layer in model.layers[:-5]:**  **layer.trainable = False**  **x = model.output**  **x = GlobalAveragePooling2D()(x)**  **x = Dense(1024, activation='relu')(x)**  **x = BatchNormalization()(x)**  **predictions = Dense(4, activation='softmax')(x)**  **model = Model(inputs=model.inputs, outputs=predictions)** |

* 1. **Training and classification dataset**

Setelah membuat model selanjutnya adalah melakukan proses training, dimana proses training ini merupakan proses dimana machine learning bekerja sehingga algoritma yang sudah kita definisikan dapat mengingat pola dari masing masing kelas pada data yang kita train. Pada proses training penelitian ini menggunakan perulangan atau Epoch Training dengan masing-masing besaran 5, 10, dan 20 untuk setiap model. Untuk perintahnya adalah menggunakan perintah berikut.

1. VGG19

Tabel 4. 8. Training dengan Model VGG19

|  |
| --- |
|  |

1. ResNet50

Tabel 4. 9. Training dengan Model ResNet50

|  |
| --- |
|  |

1. InceptionV3

Tabel 4. 10. Training dengan Model InceptionV3

|  |
| --- |
|  |

* 1. Classification results of the proposed architectures

This subsection presents and discusses of the classification results of chest X-Ray & CT images. Before to discuss these results, let us define the two most parameters used in the state of the art of deep learning and computer vision: Train curve is calculated from the training dataset that provides an idea of how well the model is learning, while the validation curve or test curve is calculated from a hold-out validation dataset that provides an idea of how well the model is generalizing. Moreover, the training and validation loss is defined as a summation of the errors made for each example in validation or training sets. In contrast to accuracy, loss is not a percentage. To summarize, a model that generalizes well is a model that is neither overfit nor underfit, is the best model. Furthermore, the confusion matrix shows the detailed representation of what happens with images after classification (Bhandary et al., 2020).

According to the figure above, it is observed that the accuracy curve of train data is rapidly increasing from epoch 0 to epoch 6 where the accuracy is equal to 83.17% then it starts to be stable until epoch 300 where the accuracy is equal to 86.28% Same for the accuracy curve of test data with an accuracy of 84.84 for epoch 300. For the loss curve of train data is rapidly decreasing from epoch 0 to epoch 6 where the loss is 43.77% then it starts to be stable until the end of training (epoch 300) where the loss is equal to 36.56%. Same for loss curve of data test with a loss of 37.85% for epoch 300. From the confusion matrix, it is observed that the first images class (Normal), the model was able to identify 1634 images correctly in the normal class, but 95 were labelled as Pneumonia. Likewise, for the second images class (Pneumon***VGG19***

Berdasarkan gambar 5.8 *accuracy* dari *training* model mencapai 95 % dengan nilai *loss* sebesar 0.03864. Proses training disini menggunakan *learning rate* 0.001 dengan input gambar sebesar 64 x 64 piksel. Waktu pelatihan yang dibutuhkan untuk 20 *epoch* dalam menjalankan *training* model ini yaitu 2 menit. Semakin 69

Banyak *epoch* maka semakin lama juga waktu yang dibutuhkan untuk *training* model. Kemudian *accuracy* dari data *validation* mencapai 90 % dengan nilai *loss* sebesar 0.3872.

1. ***ResNet50***
2. ***InceptionV3***
   1. ***Hasil Evaluasi dan Prediksi Hasil Train dan Test***
3. ***Klasifikasi data Training***
4. ***Klasifikasi Data Testing***
5. ***Klasifikasi Data Baru***
   1. Pengujian Evaluasi dan Prediksi
6. Pengaruh Epoch