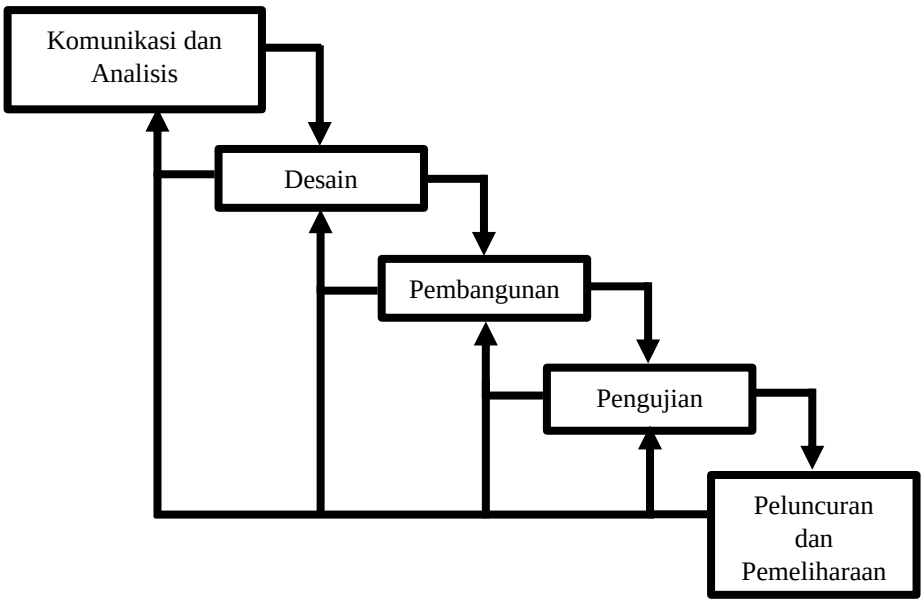
BAB III
METODE PENELITIAN

Metode penelitian yang digunakan pada penelitian ini adalah model penelitian *System Development Life Cycle*. *System Development Life Cycle* atau lebih dikenal dengan SDLC adalah proses pembuatan dan pengubahan sistem suatu model dan metodologi yang digunakan untuk mengembangkan sistem perangkat lunak (Wahid, 2020). Model SDLC pada penelitian ini menggunakan model *Waterfall*. Model *Waterfall* atau *Linear Sequential Model* adalah pendekatan pengembangan perangkat lunak yang sistematis melalui tahapan komunikasi dan analisis, desain perangkat, pembangunan perangkat, pengujian perangkat, dan peluncuran serta pemeliharaan perangkat. Setiap tahapan dapat kembali ketahapan sebelumnya jika terjadi hal yang tidak sesuai atau mengalami kesulitan pada suatu tahap tertentu. Hal ini dapat dilihat pada diagram gambar 3.1.

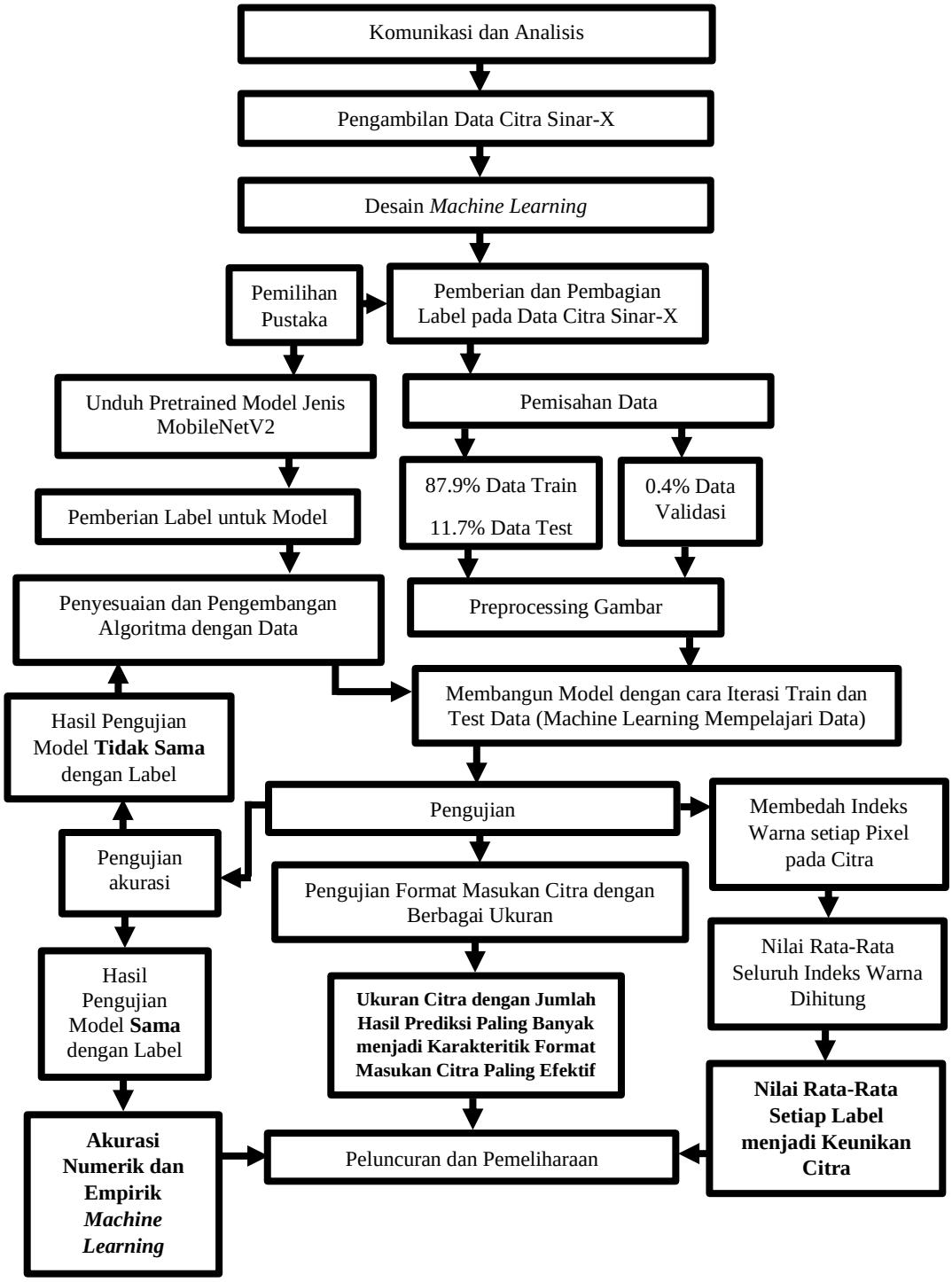


Gambar 3. 1 Diagram pendekatan *Waterfall*

3.1 Komunikasi dan Analisis

Pada gambar 3.2 dicantumkan alur dari penelitian pengembangan *Machine Learning* pengolah citra sinar-X diagnosis penyakit paru. Langkah pertama yang dilakukan adalah analisis karakteristik apa saja yang dibutuhkan oleh *Machine Learning* pengolah citra sinar-X supaya dapat mendiagnosis penyakit paru. Karakteristik yang pertama dibutuhkan adalah keunikan dari citra supaya citra

dapat diolah oleh *Machine Learning*. Setiap label atau penyakit pasti memiliki keunikan masing-masing, terutama pada *array* setiap pixel.



Gambar 3. 2 Diagram Alur Penelitian

Karakteristik kedua yang dibutuhkan adalah akurasi prediksi dari *Machine Learning* ketika mendiagnosis penyakit paru. Karakteristik ini menjadi tanda *Machine Learning* layak digunakan atau tidak. *Machine Learning* pengolah citra sinar-X diagnosis penyakit paru ini menggunakan model yang telah ada. Tetapi model sebelumnya memiliki efektifitas iterasi dan akurasi yang dapat dikembangkan lagi. Karakteristik yang terakhir adalah format masukan citra yang dapat diprediksi oleh *Machine Learning*, terutama pada ukuran efektif citra.

3.2 Alat dan Bahan

Mesin rontgen merupakan alat yang digunakan pada penelitian ini. Mesin Rontgen digunakan untuk mendapatkan data citra berupa hasil rontgen dada. Proses yang pertama kali dilakukan adalah memasukkan film putih ke dalam kaset dan diletakkan tepat di depan mesin rontgen (tempat paparan terjadi). Mesin diatur sesuai dosis dengan melewati dua langkah pengaturan. Langkah pertama mengatur intensitas paparan dengan mengatur pada tombol *kV peak* (untuk rontgen paru berada pada rentang 43 – 70 kV). Tombol *kV peak* ini berpengaruh terhadap pencahayaan pada hasil rontgen. Langkah kedua mengatur jumlah radiasi sinar-X yang dikeluarkan dengan mengatur tombol *mAs control* (untuk rontgen paru berada pada rentang 6 – 10 mAs). Tombol *mAs control* ini berpengaruh pada durasi pada saat proses rontgen berlangsung. Setelah diatur dan dilakukan proses rontgen, kaset dibawa ke ruang gelap untuk dikeluarkan kembali film yang telah digunakan, dan film melalui proses pencucian supaya hasil dapat dilihat oleh dokter maupun pasien. Sebelum hasil rontgen digunakan, hasil rontgen harus melewati proses digitalisasi.

Digitalisasi bisa dengan cara memindai hasil rontgen dengan mesin pindai. Selain itu proses digitalisasi bisa dilakukan oleh mesin rontgen secara otomatis. Mesin rontgen yang memiliki fitur tersebut bekerja seperti mesin *Computed Tomography Scan* atau yang dikenal dengan *CT-Scan*. Hasil Citra yang telah merupakan bahan utama yang digunakan pada penelitian ini. Data citra berasal dari berbagai berita dan data penelitian terdahulu mengenai dataset pendukung data primer yang diambil dari mesin rontgen. Sumber data dapat dilihat pada lampiran 4.

Selain mesin rontgen alat yang digunakan dalam penelitian adalah Bahasa Pemrograman *Python*. *Python* digunakan sebagai bahasa dasar pengembangan dan

pembangunan *Machine Learning* pada penelitian ini. *Python* dipilih karena bahasa pemrograman tingkat tinggi yang ditulis dalam Bahasa Inggris sehingga mudah dipahami oleh manusia (Zarman & Wicaksono, 2020). *Python* pertama kali dikembangkan pada tahun 1990 oleh Guido van Rossum di Stichting Mathematisch Centrum (CWI), Amsterdam. *Python* diambil dari acara televisi *Monty Python's Flying Circus* atas dasar kecintaan Guido pada acara tersebut (Budhiarto, 2018). *Python* yang digunakan adalah versi 3.11.3. Pustaka pendukung yang digunakan adalah:

1. NumPy

   NumPy adalah proyek opensource yang memungkinkan komputasi numerik dengan Python. NumPy dibuat pada tahun 2005 membangun karya awal perpustakaan Numerik dan Numarray. NumPy digunakan sebagai pemroses aljabar dari array atau matriks dari citra yang telah diubah ke tensor (Oliphant dkk., n.d.).

2. Pandas

   Pandas adalah alat analisis dan manipulasi data opensource yang cepat, kuat, fleksibel, dan mudah digunakan, dibangun menggunakan bahasa pemrograman Python (McKinney, n.d.). Pandas digunakan sebagai analisis hasil iterasi *Machine Learning* yang dicantumkan dalam informasi berbentuk tabel.

3. Matplotlib

   Matplotlib adalah pustaka komprehensif untuk membuat penggambaran statis dan animasi yang interaktif dengan menggunakan Python (Hunter, n.d.). Matplotlib digunakan sebagai analisis data yang telah mengalami iterasi, dan plot grafik untuk hasil akurasi dari iterasi *Machine Learning*.

4. Time

   Time adalah pustaka yang menyediakan fungsi ynag berhubungan dengan penggunaan waktu atau durasi. Time digunakan sebagai penghitung waktu lamanya iterasi yang dilakukan *Machine Learning*.

5. OS

   OS adalah pustaka untuk menghubungkan file dataset dengan sistem operasi yang ada pada algoritma

6. Copy

   Copy adalah pustaka yang digunakan untuk menggandakan beberapa fungsi kode.

7. Pytorch

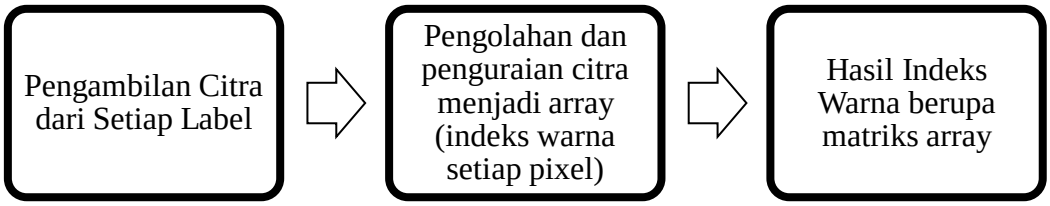
   Pytorch adalah pustaka untuk kerangka *End-to-End Machine Learning*, atau bisa diartikan sebagai kerangka untuk membuat kerangka dari awal sampai akhir pembentukan dan pembangunan *Machine Learning*. Pytorch memiliki kemampuan yang cepat dalam memproses algoritma dan memiliki tampilan yang mudah diatur dan digunakan. Pytorch digunakan sebagai pustaka utama dalam pembuatan dan perubahan algoritma pada *Machine Learning* ini.

3.3 Desain Algoritma *Machine Learning* Pengolah Citra Sinar-X dalam Diagnosis Penyakit Paru

Pada bagian ini dibahas mengenai desain dari *Machine Learning* pengolah citra sinar-X dapat bekerja dan mendapat karakteristik yang sesuai dengan apa yang dibutuhkan. Desain ini dibuat sebagai rancangan awal atau alur dari *Machine Learning* yang dikembangkan

3.3.1 Desain Algoritma *Machine Learning* untuk Mengetahui Keunikan Citra dari Setiap Label

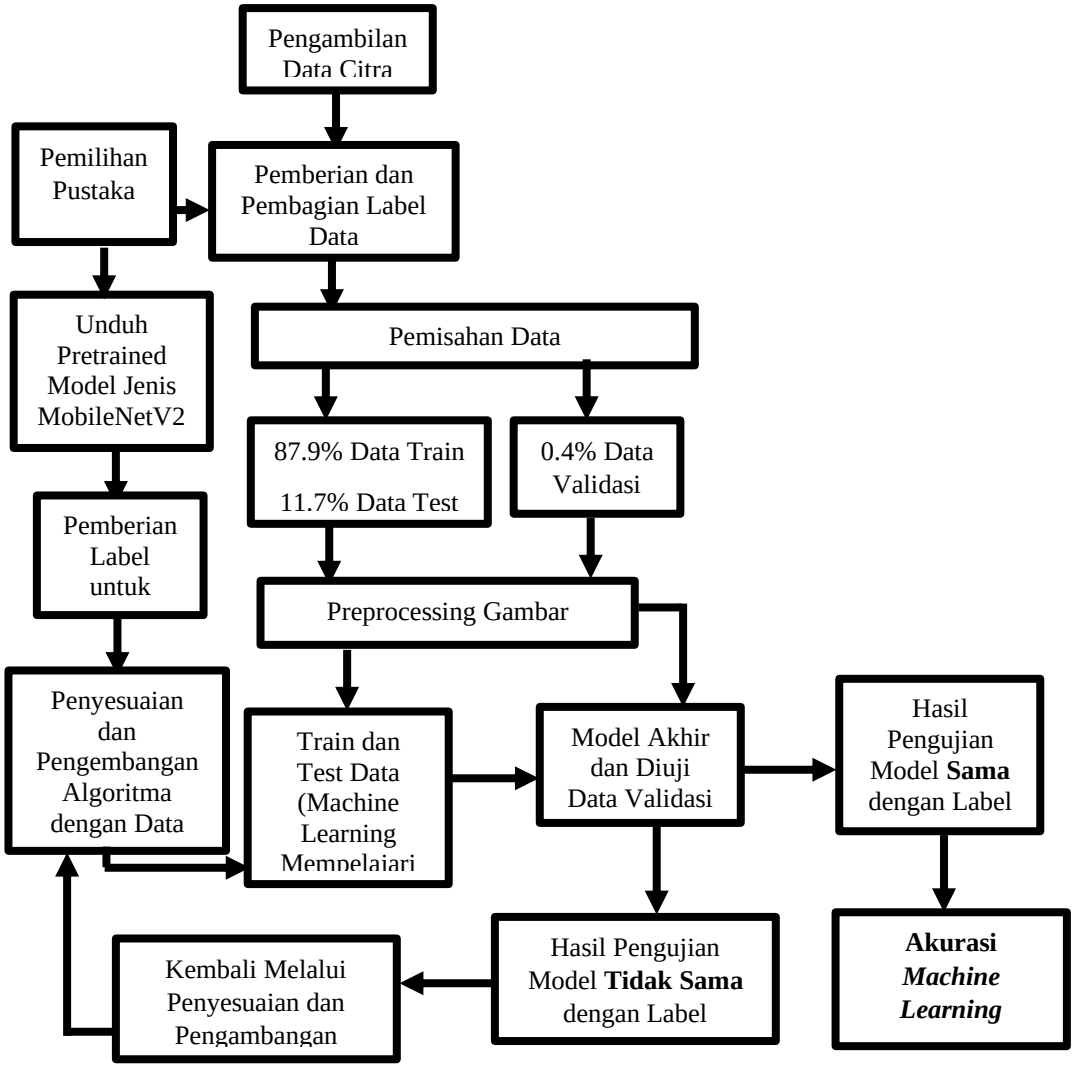
Desain algoritma untuk mengetahui keunikan citra dilakukan dari pengambilan contoh citra dari setiap label. Citra tersebut memiliki *array* masing masing setiap labelnya. *Array* tersebut akan menjadi tolak ukur *Machine Learning* supaya dapat membaca dan memprediksi penyakit paru mana yang tertera pada citra. Contoh citra yang telah dipilih akan diolah menggunakan kode untuk diurai menjadi *array* dari setiap pixel. *Array* tersebut akan muncul berupa matriks tiga dimensi yang minyimpan dtruktur data *array* yang bertipe sama, atau dikenal dikenal dengan sebutan array. Pada gambar 3.2 dicantumkan diagram algoritma untuk mengetahui keunikan citra setiap label.



Gambar 3. 3 Diagram algoritma untuk mengetahui keunikan citra setiap label

3.3.2 Desain Algoritma untuk Mengetahui Akurasi *Machine Learning* Pengolah Citra Sinar-X dalam Diagnosis Penyakit Paru

Desain algoritma untuk mengetahi akurasi *Machine Learning* yang sedang dikembangkan melalui beberapa tahap. Tahap tersebut meliputi pemilihan pustaka, pemberian label dan pemisahan data citra, *preprocessing* data, pengunduhan model yang akan dikembangkan, penyesuaian *output channel* yang digunakan pada model, iterasi dan hasil.



Gambar 3. 4 Diagram algoritma untuk mendapatkan akurasi dari *Machine Learning*

Pada gambar 3.4 dicantumkan diagram algoritma untuk mendapatkan akurasi dari *Machine Learning* yang dikembangkan. *Machine Learning* membutuhkan pustaka sebagai alat bantu pencatatannya (proses *coding*). Putaka tersebut berfungsi sebagai alat pendeteksi, pengubah dan pembaca suatu data. Pemberian label dan pemisahan data citra dilakukan untuk mengklasifikasi penyakit apa saja yang dapat diprediksi atau didiagnosis. Dan data dipisahkan sesuai dengan kebutuhannya, yaitu *training*, *test*, dan validasi atau prediksi. *Preprocessing* data citra dilakukan dengan cara mengubah ukuran, mengubah format warna, dan mengubah citra menjadi tensor (mengubah citra menjadi matriks dengan *array* setiap pixel dari citra yang diubah). Pengunduhan model merupakan tahap yang paling penting karna *Machine Learning* menggunakan sistem *Transfer Learning*. *Transfer learning* adalah mentransfer suatu kemampuan atau keahlian dari satu jenis ke jenis lainnya (Shwartz-Ziv dkk., 2022). Setelah diunduh *output channel* dari model disesuaikan dengan jumlah label yang digunakan. Terakhir model akan melakukan iterasi berupa train and test pada data yang telah dipisahkan di awal dan setelah iterasi selesai akan mendapatkan hasil berupa akurasi.

3.3.3 Desain Algoritma untuk Menentukan Format Masukan Citra Sinar-X yang dapat Diolah Oleh *Machine Learning* Pengolah Citra Sinar-X dalam Diagnosis Penyakit Paru

Desain algoritma untuk menentukan format masukan citra sinar-X supaya dapat diolah oleh *Machine Learning* dimulai dari pengambilan contoh citra dari setiap label. Setiap citra akan diubah menjadi berbagai ukuran. Citra dengan berbagai ukuran akan di prediksi oleh *Machine Learning*. Fungsi prediksi atau validasi dibuat dan digunakan ketika *Machine Learning* sudah melalui proses iterasi *train and test*.

3.4 Pembangunan *Machine Learning* Pengolah Citra Sinar-X dalam Diagnosis Penyakit Paru.

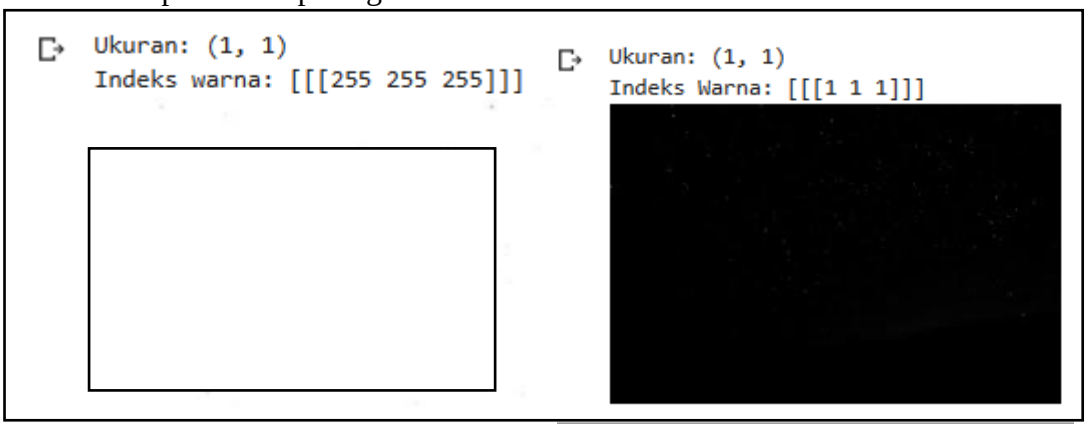
Pada bagian ini dibahas mengenai proses dari pengembangan *Machine Learning* pengolah citra sinar-X untuk mendapat karakteristik yang sesuai dengan

apa yang dibutuhkan. Pembangunan dilakukan melalui pencatatan kode di *platform Google Colab* sebagai media pencatatan. Pada bagian ini juga kode disajikan dalam bentuk *Pseudocode*.

3.4.1 Proses Mendapatkan *Array* yang Menjadi Keunikan Citra dari Setiap Label

Citra sinar-X pasti akan diubah dulu menjadi suatu data yang dapat dideteksi atau diolah oleh *Machine Learning*. Citra sinar-X dapat diubah menjadi tensor supaya dapat diolah. Merubah citra menjadi tensor adalah proses mengubah citra menjadi matriks dengan *array* setiap pixel dari citra yang diubah. Bentuk matriks ini yang dapat digunakan *Machine Learning* untuk mempelajari data.

Citra rontgen yang telah diubah menjadi tensor dapat dilihat perbedaan *array* untuk setiap label, terutama pada bagian citra paru paru. Warna gelap akan memiliki angka mendekati 1 dan semakin terang mendekati angka 255. Indeks tersebut dapat dilihat pada gambar 3.2:



Gambar 3. 5 *Array* gelap dan terang

Pseudocode dari kode proses mengurai *array* setiap pixel dicantumkan sebagai berikut:

```
from PIL IMPORT Image
IMPORT numpy as np
from numpy IMPORT asarray
SET image TO Image.open('Lokasi contoh citra yang diurai')
SET newsize TO (256, 256)
SET im TO image.resize(newsize)
```

```
SET data TO asarray(im)
SET data1 TO asarray(image)
SET image2 TO Image.fromarray(data)
OUTPUT('Ukuran:', image2.size)
OUTPUT('Array:')
OUTPUT(data)
OUTPUT("Hasil rata-rata adalah :")
OUTPUT(np.mean(data))
```

3.4.2 Proses Mendapatkan Akurasi *Machine Learning* Pengolah Citra Sinar-X dalam Diagnosis Penyakit Paru.

Proses mendapatkan akurasi *Machine Learning* meliputi pemberian label dan pemisahan data, *preprocessing* data, pengunduhan model yang akan dikembangkan, pengembangan algoritma model, iterasi.

Pseudocode dari pemilihan pustaka dapat dilihat sebagai berikut:

```
from __future__ IMPORT OUTPUT_function, division
IMPORT torch
IMPORT torch.nn as nn
IMPORT torch.optim as optim
IMPORT torchvision
IMPORT torch.backends.cudnn as cudnn
IMPORT warnings
warnings.filterwarnings('ignore')
from torch IMPORT nn
from torchvision.utils IMPORT make_grid
from torchvision IMPORT datasets, models, transforms
from torch.optim IMPORT lr_scheduler
from sklearn.metrics IMPORT classification_report, confusion_matrix
from torch.utils.data IMPORT DataLoader
from torch.autograd IMPORT Variable
```

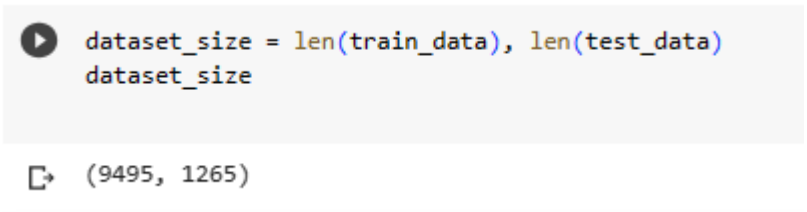
```
IMPORT numpy as np
IMPORT pandas as pd
IMPORT matplotlib.pyplot as plt
IMPORT time
IMPORT os
IMPORT copy
from collections IMPORT OrderedDict
SET cudnn.benchmark TO True
plt.ion()

SET device TO torch.device("cuda:0" IF torch.cuda.is_available() else "cpu")
"""### Load Data dan Tranform Data"""
```

1. Pemberian Label dan Pemisahan Data Citra Sinar-X

Data hasil rontgen pada lampiran 4 dikumpulkan melalui website Kaggle. Kriteria data yang dipilih memiliki sertifikasi supaya teruji keabsahannya. Data yang telah dikumpulkan dikelompokan sesuai label yang dibutuhkan, yaitu:

1. Covid19
2. Normal
3. Pneumonia
4. Tuberculosis

Setelah dikumpulkan sesuai label data dibagi menjadi 3 jenis kebutuhan, yaitu data untuk *train*, *test*, dan validasi. Data mengalami *up sampling* (penambahan data) atau pun *down sampling* (pengurangan data). Sehingga jumlah setiap data berbeda dan disesuaikan kecocokannya dengan algoritma yang akan dibuat. Jumlah data yang digunakan adalah sebagai berikut:



Gambar 3. 6 Jumlah data train dan test yang digunakan *Machine Learning*

Rincian data dari Gambar 3.6 dicantumkan sebagaii berikut:

1. Data Train 87,9 persen = 9495
    a. Covid19 = 460
    b. Normal = 4500
    c. Pneumonia = 3875
    d. Tuberculosis = 650
2. Data Test 11,7 persen = 1265
    a. Covid19 = 110
    b. Normal = 725
    c. Pneumonia = 390
    d. Tuberculosis = 40

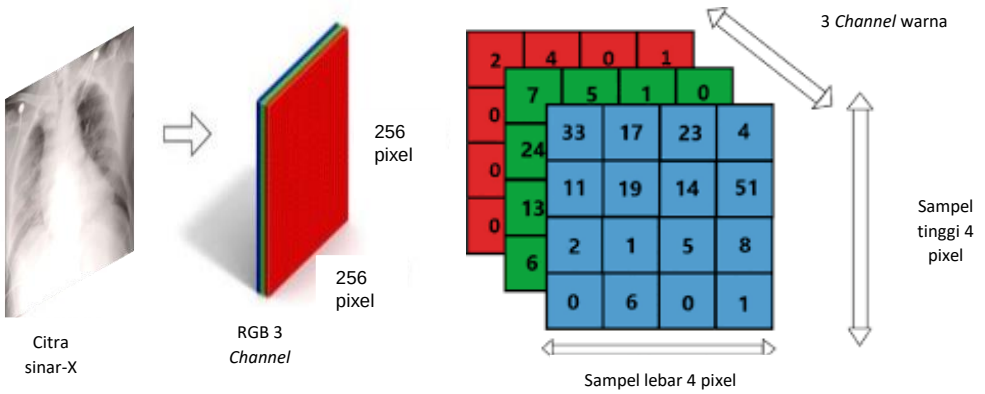
Pseudocode dari pemisahan data dan pemberian label dapat dilihat sebagai berikut:

```
DEFINE FUNCTION output_label(label):
    SET output_mapping TO {
        0: "COVID19",
        1: "NORMAL",
        2: "PNEUMONIA",
        3: "TURBERCULOSIS"
    }
    SET INPUT TO (label.item() IF type(label) EQUALS torch.Tensor else label)
    RETURN output_mapping[INPUT]
FOR images, labels IN train_loader:
    break
```

2. Preprocessing data

Data melalui tahap *preprocessing* setelah melalui tahap pemberian label dan pemisahan. Preprocessing berupa perubahan ukuran (*resize* / *rescaling*) supaya sama rata, yaitu dengan ukuran rata-rata 256 × 256 pixel. Citra yang memiliki ukuran lebih kecil akan mengalami *upscaling* (merubah citra dengan skala yang lebih besar) dan citra yang memiliki ukuran lebih besar akan mengalami *downscaling* (merubah citra dengan skala yang lebih kecil). Setelah ukurannya sama, data harus melalui tahap konversi menjadi tensor.

Normalisasi juga bisa berperan penting dalam membantu proses penajaman citra yang terlihat buram akibat *upscaling* atau *downscaling*. Karena hanya bersifat membantu, proses penajaman citra tidak seratus persen diterapkan pada citra (hanya beberapa bagian pada citra saja) dan menyesuaikan dengan jenis pustaka yang akan digunakan. Jenis normalisasi yang digunakan adalah format Red Green Blue (RGB), yang memiliki format dengan besaran [0.485, 0.456, 0.406], [0.229, 0.224, 0.225]. Besaran tersebut merupakan format standar RGB untuk pustaka *Pytorch*. Format normalisasi RGB digunakan karena mudah diaplikasikan untuk pendeteksian objek dengann warna tertentu (Kusumanto & Tompunu, 2011). Ilustrasi dari normalisasi dan perubahan gambar menjadi tensor dinyatakan pada gambar 3.7 samapai gambar 3.8.



Gambar 3. 7 Perubahan ukuran skala citra sesuai rata-rata ukuran citra dan dinormalisasi dengan format RGB (Montesinos-López dkk., 2022)

$$F_{\mu\nu} = \begin{pmatrix} 0 & B_z & -B_y & -iE_x \\ -B_z & 0 & B_x & -iE_y \\ B_y & -B_x & 0 & -iE_z \\ iE_x & iE_y & iE_z & 0 \end{pmatrix}$$

$$F_{\mu\nu} = \frac{\partial A_\nu}{\partial x_\mu} - \frac{\partial A_\mu}{\partial x_\nu}$$

Gambar 3. 8 Salah satu persamaan dan matrik yang diadaptasi untuk membentuk tensor pada citra (Branson, 2013)

Tahap *Preprocessing* ini sering juga disebut augmentasi data. Tahapan ini dapat dilihat pada *pseudocode* berikut:

```
SET Train TO transforms.Compose([
        transforms.Resize((256,256)),
        transforms.ToTensor()
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
])
SET Test TO transforms.Compose([
        transforms.Resize((256,256)),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
])
SET PATH TO "Lokasi Data"
SET train_data TO datasets.ImageFolder(os.path.join(PATH, 'train'),
                            transform=Train)
SET test_data TO datasets.ImageFolder(os.path.join(PATH,'test'),
                           transform=Test)
SET train_loader TO DataLoader(train_data, batch_size=128, shuffle=True)
SET test_loader TO DataLoader(test_data, batch_size=128)
SET class_names TO train_data.classes
OUTPUT(f'Class Category: {class_names}')
OUTPUT(f'Train Data Length: {len(train_data)}')
OUTPUT(f'Test Data Length: {len(test_data)}')
images.shape
SET ims TO make_grid(images, nrow=8)
SET inv_normalize TO transforms.Normalize(
    SET mean TO [-0.485/0.229, -0.456/0.224, -0.406/0.225],
    SET std TO [1/0.229, 1/0.224, 1/0.225]
)
SET im_inv TO inv_normalize(ims)
```

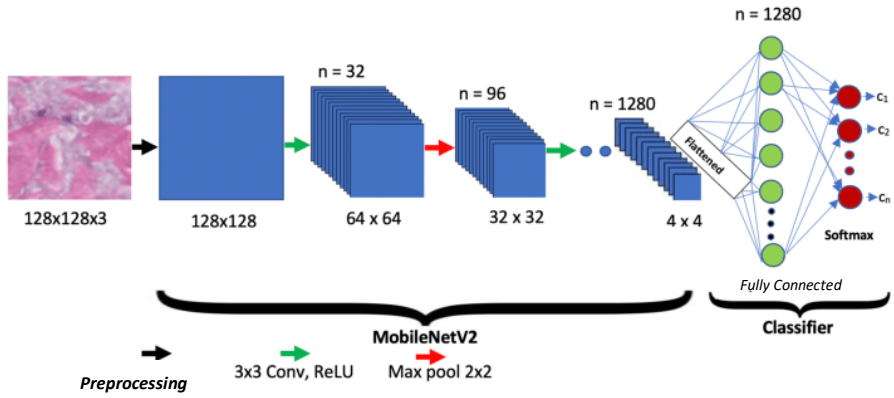
```
plt.figure(figsize=(12,8))
plt.imshow(np.transpose(im_inv.numpy(), (1, 2, 0)));
```

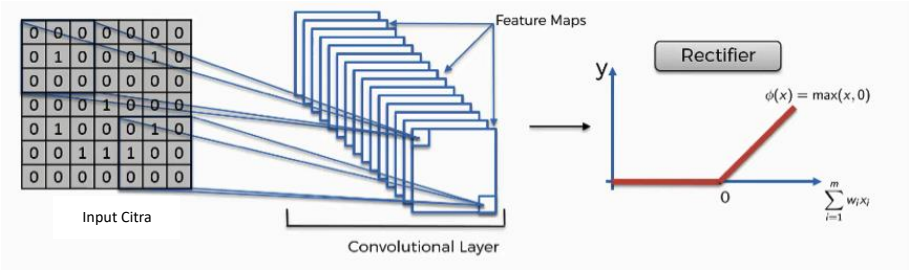
3. Pengunduhan dan Penyesuaian Algoritma Model

Algoritma *Machine Learning* dibuat dengan bantuan aplikasi atau web pencatatan yang bernama Google Colab. Goole Colab merupakan web pencatatan yang dibuat oleh google dengan basis pencatatan menggunakan Bahasa pemrograman *Python*. Dalam proses pencatatan dibutuhkan pustaka utama yang digunakan sebagai pembangun dan perubah algoritma *Machine Learning*. Pustaka utama yang digunakan sebagai pembangun *Machine Learning* adalah pustaka *Pytorch*. *Pytorch* dipilih karena memiliki waktu iterasi yang lebih singkat dibanding pustaka lain seperti *TensorFlow*, serta memiliki tampilan yang mudah dipahami dan mudah untuk diubah (Novac dkk., 2022).

Proses pembangunan model *Machine Learning* menggunakan sistem *Transfer Learning*. *Transfer learning* adalah mentransfer suatu kemampuan atau keahlian dari satu jenis ke jenis lainnya (Shwartz-Ziv dkk., 2022). *Transfer learning* dilakukan dari model yang telah ada dan dilatih sebelumnya, model ini disebut juga *Pre-trained Models*. *Pre-trained model* yang digunakan adalah *MobileNetV2*.

MobileNetV2 adalah jenis *pre-trained model* yang dikembangkan dari segi performa model dalam mengerjakan suatu tugas serta *bencmark* dengan ukuran dan data yang memiliki jangkauan luas. *MobileNetV2* memiliki arsitektur konvolusi *kernel* (matrix) 1×1 yang diklasifikasi dengan persamaan ReLU6 (klasifikasi non-linear yang cocok dugunakan dalam perhitungan presisi rendah). Lalu dilanjut dengan konvolusi *depthwise* (konvolusi yang lebih dalam untuk klasifikasi non-linear) dengan *kernel* 3×3 dan diklasifikasi kembali oleh prsamaan ReLU6. Dan terakhir dilanjut dengan konvolusi kernel 1×1 serta diklasifikasi dengan persamaan linear (Sandler dkk., 2019). *MobileNetV2* memiliki model *output channel* basis sebanyak 1000. Asitektur dari *MobileNetV2* dapat dilihat pada gambar 3.9.





Gambar 3. 9 Arsitektur dan fungsi aktivasi dari *MobileNetV2 (PradyaSin, 2019)*

Model di unduh melalui code yang dicantumkan pada pseudo code berikut:

```
DEFINE CLASS Model(nn.Module):
  DEFINE FUNCTION __init__(self, in_channels=1):
    super(Model, self).__init__()
    SET self.model TO models.mobilenet_v2(pretrained=True)
    SET self.model.classifier[1] TO nn.Linear(self.model.last_channel, banyak chanel yang diinginkan)
  DEFINE FUNCTION forward(self, x):
    RETURN self.model(x)

SET model_ft TO Model().to(device)
SET criterion TO nn.CrossEntropyLoss()
SET optimizer TO optim.SGD(model_ft.parameters(), lr=0.01, momentum=0.9)
```

4. Iterasi

Iterasi dilakukan supaya model dapat mempelajari data yang telah dipersiapkan sebelumnya. Model melalui proses iterasi dengan ketentuan iterasi sebagai berikut:

1. Sampel iterasi pada fungsi = 10 kali
2. *Learning rate* = 0.01
3. *Momentum* iterasi = 0.9
4. *Epoch* riil = 15 kali

Fungsi kode iterasi dinyatakan dalam *pseudocode* berikut:

```
SET model_ft TO Model().to(device)
SET criterion TO nn.CrossEntropyLoss()
SET optimizer TO optim.SGD(model_ft.parameters(), lr=0.01, momentum=0.9)
DEFINE FUNCTION train(num_epochs=10):
  SET train_loss TO []
  SET train_accuracy TO []
  SET val_loss TO []
  SET val_accuracy TO []
  SET predictions_list TO []
  SET labels_list TO []
  SET best_loss TO .4
  SET best_accuracy TO 0
  SET start TO time.time()
  FOR epoch IN range(num_epochs):
    OUTPUT('-' * 10)
    OUTPUT(f'Epoch: {epoch+1}/{num_epochs}')
    OUTPUT('-' * 10)

    SET train_correct TO 0
    SET val_correct TO 0
    FOR X_train, y_train IN train_loader:
      model_ft.train()
```

```
    SET X_train, y_train TO X_train.to(device), y_train.to(device)
    SET y_pred TO model_ft(X_train)
    SET loss TO criterion(y_pred, y_train)
    SET predicted TO torch.max(y_pred.data, 1)[1]
    train_correct += (predicted EQUALS y_train).sum()/dataset_size[0]
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
  train_loss.append(loss.detach().cpu().numpy())
  train_accuracy.append(train_correct.detach().cpu().numpy())
  OUTPUT(f'train_loss: {loss:.4f}\ntrain_accuracy: {train_correct:.4f}\n')

  with torch.no_grad():
    FOR X_test, y_test IN test_loader:
      model_ft.eval()
      SET X_test, y_test TO X_test.to(device), y_test.to(device)
      labels_list.append(y_test)
      SET y_val TO model_ft(X_test)
      SET predicted TO torch.max(y_val.data, 1)[1]
      predictions_list.append(predicted)
      val_correct += (predicted EQUALS y_test).sum()/dataset_size[1]

  SET loss TO criterion(y_val, y_test)
  val_loss.append(loss.detach().cpu().numpy())
  val_accuracy.append(val_correct.detach().cpu().numpy())
  OUTPUT(f'val_loss: {loss:.4f}\nval_accuracy: {val_correct:.4f}\n')

  IF best_accuracy < val_correct:
    IF best_loss > loss:
      SET best_loss TO loss
      SET best_accuracy TO val_correct
      SET best_model TO copy.deepcopy(model_ft)
```

```
  SET end TO time.time() - start
  OUTPUT('-' * 10)
  OUTPUT(f'Training time taken: {int(end/60)}min {int(end%60)}sec\n')
  OUTPUT(f'Best val_accuracy: {best_accuracy}')
  OUTPUT(f'Lowest val_loss: {best_loss}')
  RETURN best_model, [train_loss, train_accuracy, val_loss, val_accuracy],
[predictions_list, labels_list]

SET trained_model, result, preds_and_labels TO train(Banyaknya iterasi)
```

Model yang telah melalui proses iterasi akan diuji dengan data validasi yang telah melalui proses tahap persiapan juga. Ketika model tidak dapat memprediksi sesuai label yang telah diberikan di awal maka model akan mengalami pengembangan dan penyesuaian kembali. Tetapi jika hasil prediksi yang ditunjukan model sudah sesuai dengan label yang diberikan, maka model telah dapat memprediksi data baru, dan dapat diluncurkan dalam bentuk webmau pun aplikasi.

3.4.2 Proses Mendapatkan Format Masukan Citra yang dapat diolah *Machine Learning* Pengolah Citra Sinar-X dalam Diagnosis Penyakit Paru.

Citra melalui proses perubahan ukuran supaya dapat diprediksi dengan label yang tepat. Perubahan ukuran ini harus dilakukan secara efektif supaya mendapatkan akurasi yang tepat. Proses pengujian dilakukan pada setiap label dengan berbagai ukuran, supaya terlihat pada ukuran berapakah citra dapat diprediksi dengan tepat. Ukuran yang digunakan saat pengujian adalah 64×64 pixel, 128×128 pixel, 256×256 pixel, 512×512 pixel, dan 1024×1024 pixel. Citra diprediksi menggunakan Fungsi prediksi atau validasi dibangun dengan kode yang ditunjukan oleh *pseudocode* berikut:

```
IMPORT torch.nn.functional as F
from PIL IMPORT Image, ImageOps
DEFINE FUNCTION image_pred(model, image_transforms, image_path,
labels_data):
  SET image TO Image.open(image_path).convert('RGB')
```

```
  SET resize_image TO image.resize((ukuran yang akan diuji))
  SET image TO image_transforms(image).float()
  SET image TO image.unsqueeze(0).to(device)
  SET output TO model(image)
  SET probs TO F.softmax(output, dim=1)
  SET conf, classes TO torch.max(probs, 1)
  RETURN labels_data[classes.item()], round(conf.item(), 3), resize_image

from os IMPORT listdir
SET dir TO "Lokasi Data Validasi"
SET image_list TO []
FOR images IN listdir(dir):
   IF (images.endswith(".png") or images.endswith(".jpg") or
images.endswith(".jpeg")):
     image_list.append(str(dir+images))
SET fig TO plt.figure(figsize=(80,32))
SET cols, rows TO len(image_list), 1
FOR i IN range(1, cols * rows + 1):
   SET labels_pred, conf, image TO image_pred(model1, image_transforms,
image_list[i-1], labels_data)
   fig.add_subplot(rows, cols, i)
   plt.title(f'{labels_pred} {conf}')
   plt.axis('off')
   plt.imshow(image)
plt.show()
```

3.5 Pengujian dan Pengolahan Data untuk Mendapatkan Karakteristik *Machine Learning* Pengolah Citra Sinar-X dalam Diagnosis Penyakit Paru

Pada bagian ini dibahas mengenai pengolahan data untuk mendapatkan karakteristik *Machine Learning* pengolah citra sinar-X dalam diagnosis penyakit paru.

3.5.1 Pengolahan Data untuk Mendapatkan *Array* sebagai Keunikan Citra dari Setiap Label

Data yang diperolah setelah setiap contoh citra diurai menjadi *array* dari setiap pixel adalah berupa matriks tiga dimensi. Matriks inidapat dicari rata ratanya untuk mendapatkan nilai *array* secara keseluruhan dari setiap citra. Tetapi selain citra secara keseluruhan, *array* pada bagian paru paru pun diurai secara terpisah untuk melihat seperti apa keunikan citra dari setiap labelnya.

3.5.2 Pengolahan Data Hasil Iterasi untuk Mendapatkan Akurasi *Machine Learning* Pengolah Citra Sinar-X dalam Diagnosis Penyakit Paru.

Hasil iterasi akan berupa nilai akurasi dan nilai loss dari proses *Machine Learning* melakukan *train and test*. Nilai akurasi ini bersifat numerik dan dapat setiap proses iterasi dapat dilihat perubahan akurasinya. Begitu pun nilai loss pada saat *train and test*, dapat dilihat pula perubahannya dari setiap proses iterasi. Nilai akurasi juga didapat dari proses validasi data menggunakan data validasi yang telah dipisahkan di awal. Hasil validasi atau prediksi ini bersifat empirik dan dapat dibandingkan dengan akurasi numerik sebelumnya. Selain itu Nilai akurasi model *Machine Learning* sebelumnya pun dapat dibandingkan dengan *Machine Learning* yang telah mendapatkan perubahan pada *output channel*.

3.5.3 Pengolahan Data untuk Mendapatkan Format Masukan Citra yang dapat diolah Machine Learning Pengolah Citra Sinar-X dalam Diagnosis Penyakit Paru.

Data hasil prediksi dari setiap ukuran citra akan dicantumkan dalam data berupa tabel. Format ukuran citra yang paling banyak diprediksi akan menjadi ukuran paling efektif supaya *Machine Learning* dapat mengolah citra tersebut.