LAPORAN UJIAN AKHIR SEMESTER PENGOLAHAN CITRA DIGITAL

PENINGKATAN KUALITAS CITRA
UNTUK DETEKSI TUBERKULOSIS PADA PARU-PARU

DISUSUN OLEH:

Riva Dian Ardiansyah	22031554043
Faiz Dwi Febriansyah	22031554023
Daffa Fazly Rashidan	22031554006

PROGRAM STUDI S1 SAINS DATA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS NEGERI SURABAYA

2024

BAB I
PENDAHULUAN

1.1 Latar Belakang

Tuberkulosis (TB) adalah masalah kesehatan global dan deteksi dini sangat penting karena penyakit infeksi yang menyerang sebagian besar populasi dunia. Penyakit ini umumnya disebabkan oleh bakteri yang dikenal sebagai Mycobacterium tuberculosis dan sebagian besar menyerang paru-paru manusia. TBC menyebar melalui udara dari semua orang atau di mana saja. Melalui batuk, bersin, meludah yang dirasakan oleh pasien, bakteri TB menyebar secara luas ke udara. Setiap tahun, sepertiga dari populasi dunia telah terkena bakteri Mycobacterium TB pada tingkat satu persen dari populasi dengan infeksi baru. Bakteri TB telah menyebabkan umur yang pendek bagi penduduk dunia[1]. Oleh karena itu, tindakan proaktif terhadap penyakit TB merupakan tindakan yang paling tepat untuk mengurangi kecepatan penyebarannya. Metode deteksi saat ini memiliki keterbatasan, sementara radiografi paru-paru menawarkan potensi yang belum sepenuhnya dimanfaatkan karena interpretasinya memerlukan keahlian khusus dan seringkali subjektif. Proyek ini bertujuan untuk mengatasi tantangan tersebut dengan meningkatkan kualitas citra radiografi paru-paru dan mengembangkan metode deteksi TB yang lebih baik, sehingga dapat meningkatkan efisiensi diagnosis dan pengobatan TB.

1.2 Tujuan dan Manfaat

1. Meningkatkan kualitas citra paru-paru yang akan digunakan dalam deteksi tuberkulosis.
2. Menggunakan suatu algoritma atau metode yang efisien dan efektif untuk deteksi tuberkulosis pada citra paru-paru.
3. Meningkatkan akurasi dan kecepatan diagnosis tuberkulosis, sehingga dapat memberikan pengobatan yang tepat waktu dan efektif bagi pasien.

BAB II
METODE

2.1 Peningkatan Citra

Teknik peningkatan citra memperbaiki gambar yang diberikan, membuat fitur yang diinginkan lebih mudah dilihat oleh sistem visual manusia atau lebih mudah ditemukan oleh sistem analisis citra otomatis. Teknik ini memungkinkan pengamat melihat detail pada gambar yang mungkin tidak dapat mereka lihat secara langsung pada gambar aslinya[2]. Misalnya, tingginya tingkat kebisingan dalam gambar, ketidaksesuaian antara rentang dinamis data dan tampilan, atau kurangnya kontras. Peningkatan gambar pada dasarnya adalah pemetaan atau transformasi dari satu gambar ke gambar lainnya. Transformasi ini tidak selalu satu-ke-satu, sehingga dua gambar yang berbeda dapat berubah menjadi gambar yang sama atau serupa setelah ditambahkan. Filter linier adalah alat penting saat pemrosesan sinyal dan gambar berkembang. Adanya beberapa properti yang diinginkan dan kesederhanaan matematisnya membuatnya mudah untuk dirancang dan digunakan. Selain itu, filter linier tidak berfungsi dengan baik dalam banyak gambar dan filter linear tidak berfungsi dengan noise yang tidak aditif dan dengan masalah sistem yang mengalami nonlinieritas atau statistik non-Gaussian. Maka, solusi nonlinier dihasilkan oleh berbagai kriteria, termasuk kriteria entropi maksimum. Filter linier cenderung memburamkan bagian tepi dalam program pemrosesan gambar[3]. Selain itu, berbagai kriteria seperti kriteria entropi maksimum menghasilkan solusi non-linier. Dalam pemrosesan gambar, filter linier mengaburkan tepian, tidak menghilangkan derau impulsif secara efektif, dan tidak bekerja dengan baik di hadapan derau yang bergantung pada sinyal.

2.2 Histogram Ekualisasi

Setiap tingkat abu-abu memiliki peluang yang sama untuk terjadi karena teknik histogram equalization (HE) bertujuan untuk mendistribusikan tingkat abu-abu dalam suatu gambar. Dengan HE, kecerahan dan kontras gambar gelap diubah dan gambar dengan kontras rendah dan warna gelap untuk meningkatkan kualitas gambar. Untuk gambar yang gelap, histogram akan condong ke arah ujung bawah skala abu-abu, dan informasi gambar akan diperas ke ujung gelap histogram. Untuk membuat histogram lebih merata, tingkat abu-abu dapat didistribusikan kembali ke ujung gelap, yang membuat gambar menjadi jernih[3].

2.3 Koreksi Gamma

Koreksi gamma digunakan sebagai metode nonlinier untuk penyandian dan penyandian ulang pencahayaan agar dapat diadopsi dengan persepsi visual manusia. Non-linearitas pencahayaan yang dihasilkan oleh perangkat pencitraan sering kali dapat dijelaskan dengan operasi titik-bijaksana sederhana yang disebut koreksi gamma[4]. Koreksi gamma mengontrol kecerahan gambar secara keseluruhan. Pengetahuan tentang gamma penting untuk mencoba menyempurnakan gambar. Tingkat kontras gambar akan ditingkatkan dengan menemukan nilai gamma yang tepat. Metode ini secara otomatis menghitung nilai gamma dari histogram kumulatif gambar.

2.4 Listing Code

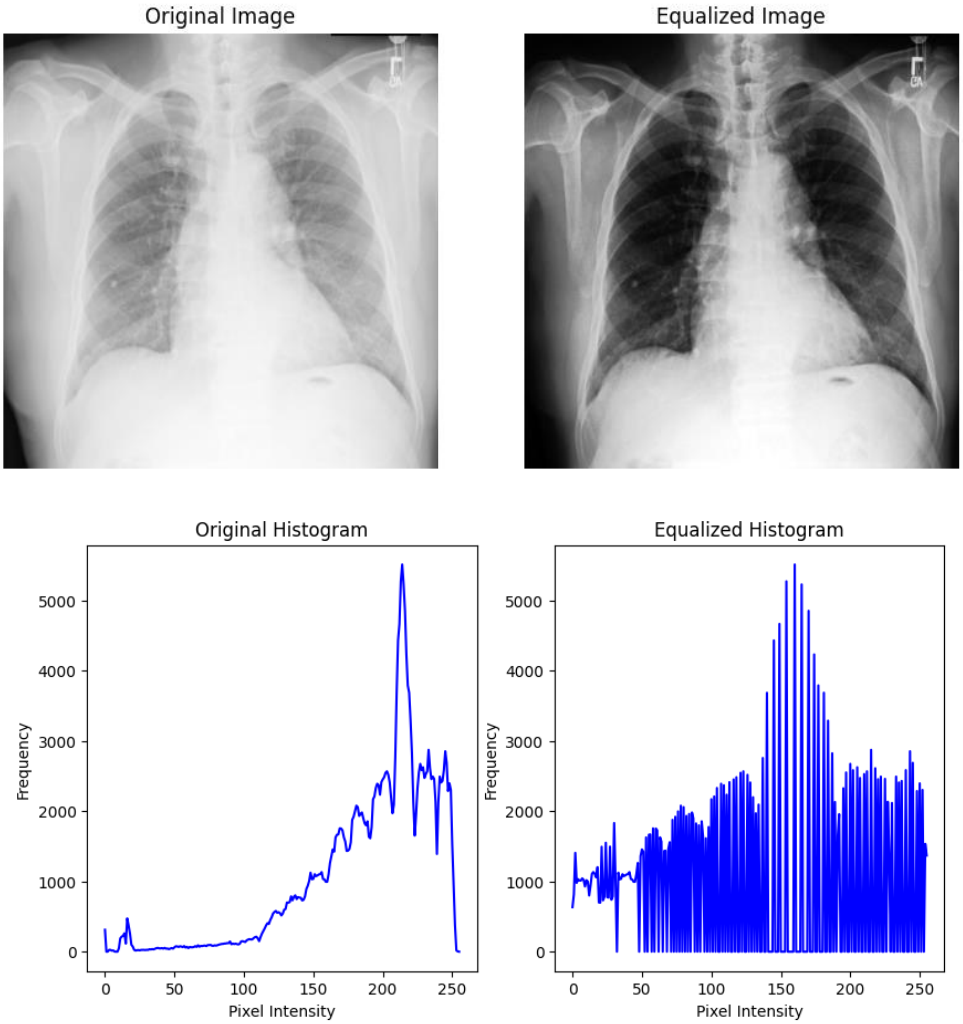
Preprocessing gambar

Histogram Ekualisasi

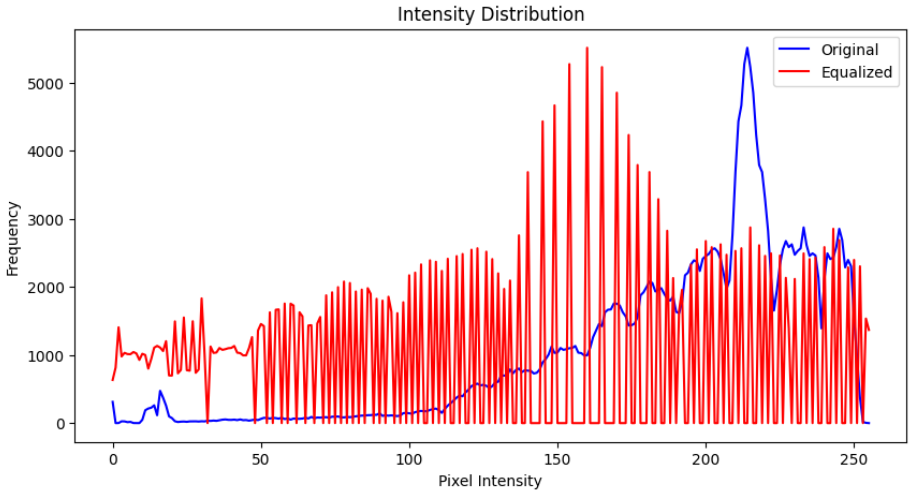
```
# Import library yang diperlukan
import cv2
import matplotlib.pyplot as plt
import numpy as np

# Fungsi untuk melakukan equalization histogram
def histogram_equalization(image):
    # Konversi gambar berwarna (BGR) ke grayscale
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Melakukan equalization histogram pada gambar grayscale
    equalized_image = cv2.equalizeHist(gray_image)

    return equalized_image

# Fungsi untuk menghitung kontras gambar
def kalkulasi_kontras(image):
    # Mengembalikan standar deviasi dari intensitas piksel gambar, yang digunakan
sebagai ukuran kontras
    return image.std()

# Fungsi untuk menghitung distribusi intensitas gambar
def kalkulasi_intensity_distribusi(image):
    # Menghitung histogram intensitas piksel gambar
    histogram, _ = np.histogram(image.flatten(), bins=256, range=[0, 256])
    return histogram

# Membaca gambar dari file
image = cv2.imread('testprepro-normal.png')

# Menerapkan equalization histogram pada gambar
equalized_image = histogram_equalization(image)

# Menghitung histogram gambar asli
hist_original = cv2.calcHist([image], [0], None, [256], [0,256])

# Menghitung histogram gambar yang telah diequalisasi
hist_equalized = cv2.calcHist([equalized_image], [0], None, [256], [0,256])

# Menampilkan gambar asli dan hasil equalization histogram
```

```
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)  # Membuat subplot pertama
plt.title('Original Image')  # Menambahkan judul
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))  # Menampilkan gambar asli dalam format RGB
plt.axis('off')  # Menghilangkan axis

plt.subplot(1, 2, 2)  # Membuat subplot kedua
plt.title('Equalized Image')  # Menambahkan judul
plt.imshow(equalized_image, cmap='gray')  # Menampilkan gambar equalized dalam format grayscale
plt.axis('off')  # Menghilangkan axis

plt.show()  # Menampilkan plot

# Menampilkan histogram gambar asli dan equalized
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)  # Membuat subplot pertama
plt.title('Original Histogram')  # Menambahkan judul
plt.plot(hist_original, color='blue')  # Menampilkan histogram gambar asli dengan warna biru
plt.xlabel('Pixel Intensity')  # Menambahkan label sumbu x
plt.ylabel('Frequency')  # Menambahkan label sumbu y

plt.subplot(1, 2, 2)  # Membuat subplot kedua
plt.title('Equalized Histogram')  # Menambahkan judul
plt.plot(hist_equalized, color='blue')  # Menampilkan histogram gambar equalized dengan warna biru
plt.xlabel('Pixel Intensity')  # Menambahkan label sumbu x
plt.ylabel('Frequency')  # Menambahkan label sumbu y

plt.show()  # Menampilkan plot

# Menghitung kontras sebelum dan sesudah equalization histogram
kontras_asli = kalkulasi_kontras(cv2.cvtColor(image, cv2.COLOR_BGR2GRAY))
kontras_equalized = kalkulasi_kontras(equalized_image)

# Menampilkan kontras sebelum dan sesudah equalization histogram
print("Kontras sebelum equalization histogram:", kontras_asli)
print("Kontras setelah equalization histogram:", kontras_equalized)

# Menghitung distribusi intensitas sebelum dan sesudah equalization histogram
distribusi_asli = kalkulasi_intensity_distribusi(cv2.cvtColor(image, cv2.COLOR_BGR2GRAY))
distribusi_equalized = kalkulasi_intensity_distribusi(equalized_image)

# Menampilkan distribusi intensitas
plt.figure(figsize=(10, 5))
```

```
# Menampilkan distribusi intensitas gambar asli
plt.plot(distribusi_asli, color='blue', label='Original')

# Menampilkan distribusi intensitas gambar equalized
plt.plot(distribusi_equalized, color='red', label='Equalized')

plt.title('Intensity Distribution')  # Menambahkan judul
plt.xlabel('Pixel Intensity')  # Menambahkan label sumbu x
plt.ylabel('Frequency')  # Menambahkan label sumbu y
plt.legend()  # Menambahkan legenda untuk membedakan antara original dan equalized

plt.show()  # Menampilkan plot
```




```
Kontras sebelum equalization histogram: 42.70728630316623
Kontras setelah equalization histogram: 74.0295870571604
```

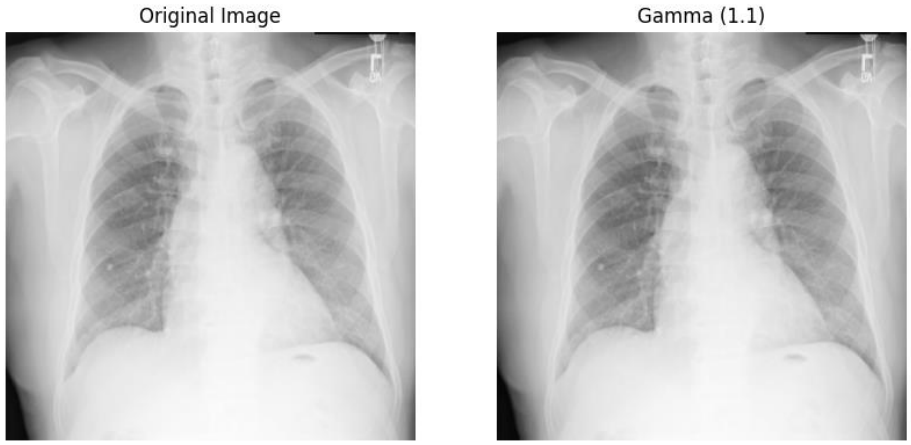


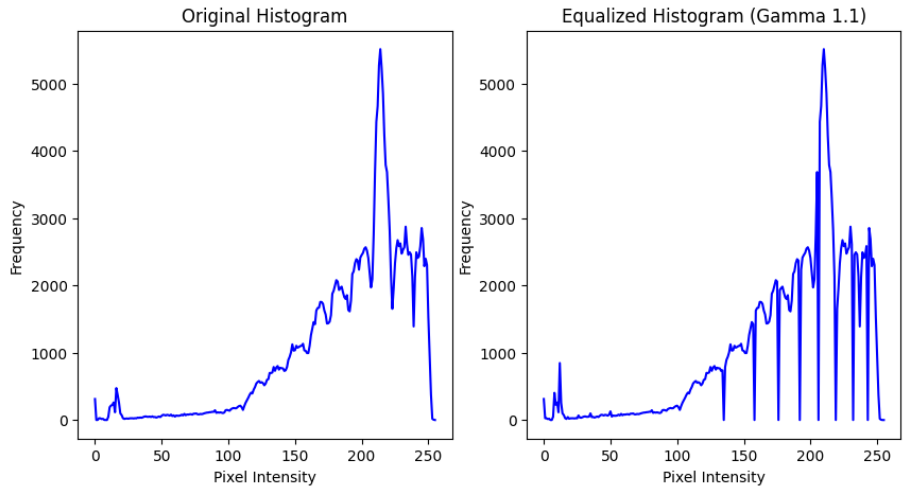
Gamma Correction

```
# Import library yang diperlukan
import cv2
import matplotlib.pyplot as plt
import numpy as np

# Fungsi untuk menghitung kontras
def kalkulasi_kontras(image):
    # Mengembalikan standar deviasi dari intensitas piksel gambar, yang digunakan sebagai ukuran kontras
    return image.std()

# Fungsi untuk menghitung distribusi intensitas
def kalkulasi_intensity_distribusi(image):
    # Menghitung histogram intensitas piksel gambar
    histogram, _ = np.histogram(image.flatten(), bins=256, range=[0, 256])
    return histogram

# Fungsi untuk koreksi gamma
def gamma_correction(image, gamma):
    # Terapkan koreksi gamma dengan membagi intensitas piksel dengan 255, menerapkan operasi gamma, dan mengalikannya kembali dengan 255
    adjusted_image = np.power(image / 255.0, gamma) * 255.0
    return np.uint8(adjusted_image)

# Baca gambar
image = cv2.imread('testprepro-normal.png')

# Terapkan koreksi gamma dengan nilai gamma 1.1
gamma = 1.1
equalized_image = gamma_correction(image, gamma)

# Hitung histogram gambar asli
hist_original = cv2.calcHist([image], [0], None, [256], [0, 256])
```

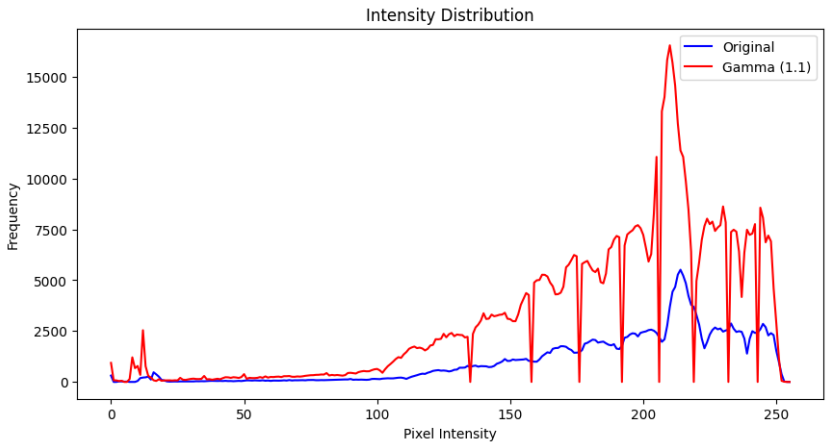
```
# Hitung histogram gambar yang telah dikoreksi gamma
hist_equalized = cv2.calcHist([equalized_image], [0], None, [256], [0, 256])

# Tampilkan gambar asli dan hasil koreksi gamma
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)  # Membuat subplot pertama
plt.title('Original Image')  # Menambahkan judul
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))  # Menampilkan gambar asli dalam format RGB
plt.axis('off')  # Menghilangkan axis

plt.subplot(1, 2, 2)  # Membuat subplot kedua
plt.title('Gamma (1.1)')  # Menambahkan judul
plt.imshow(equalized_image, cmap='gray')  # Menampilkan gambar hasil koreksi gamma dalam format grayscale
plt.axis('off')  # Menghilangkan axis

plt.show()  # Menampilkan plot

# Tampilkan histogram gambar asli dan hasil koreksi gamma
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)  # Membuat subplot pertama
plt.title('Original Histogram')  # Menambahkan judul
plt.plot(hist_original, color='blue')  # Menampilkan histogram gambar asli dengan warna biru
plt.xlabel('Pixel Intensity')  # Menambahkan label sumbu x
plt.ylabel('Frequency')  # Menambahkan label sumbu y

plt.subplot(1, 2, 2)  # Membuat subplot kedua
plt.title('Equalized Histogram (Gamma 1.1)')  # Menambahkan judul
plt.plot(hist_equalized, color='blue')  # Menampilkan histogram gambar hasil koreksi gamma dengan warna biru
plt.xlabel('Pixel Intensity')  # Menambahkan label sumbu x
plt.ylabel('Frequency')  # Menambahkan label sumbu y

plt.show()  # Menampilkan plot

# Perhitungan kontras sebelum dan sesudah koreksi gamma
kontras_asli = kalkulasi_kontras(cv2.cvtColor(image, cv2.COLOR_BGR2GRAY))
kontras_equalized = kalkulasi_kontras(equalized_image)

# Menampilkan kontras sebelum dan sesudah koreksi gamma
print("Kontras sebelum koreksi gamma:", kontras_asli)
print("Kontras setelah koreksi gamma:", kontras_equalized)

# Distribusi intensitas sebelum dan sesudah koreksi gamma
```

```
distribusi_asli = kalkulasi_intensity_distribusi(cv2.cvtColor(image,
cv2.COLOR_BGR2GRAY))
distribusi_equalized = kalkulasi_intensity_distribusi(equalized_image)

# Plot distribusi intensitas
plt.figure(figsize=(10, 5))

# Menampilkan distribusi intensitas gambar asli
plt.plot(distribusi_asli, color='blue', label='Original')

# Menampilkan distribusi intensitas gambar hasil koreksi gamma
plt.plot(distribusi_equalized, color='red', label='Gamma (1.1)')

plt.title('Intensity Distribution')  # Menambahkan judul
plt.xlabel('Pixel Intensity')  # Menambahkan label sumbu x
plt.ylabel('Frequency')  # Menambahkan label sumbu y
plt.legend()  # Menambahkan legenda untuk membedakan antara original dan hasil koreksi
gamma

plt.show()  # Menampilkan plot
```






```
Kontras sebelum koreksi gamma: 42.70728630316623
Kontras setelah koreksi gamma: 44.5890011835097
```



Sharpening (penajaman)

```
# Import library yang diperlukan
import cv2
import matplotlib.pyplot as plt
import numpy as np

# Fungsi untuk menghitung kontras
def kalkulasi_kontras(image):
    # Mengembalikan standar deviasi dari intensitas piksel gambar, yang digunakan sebagai ukuran kontras
    return image.std()

# Fungsi untuk menghitung distribusi intensitas
def kalkulasi_intensity_distribusi(image):
    # Menghitung histogram intensitas piksel gambar
    histogram, _ = np.histogram(image.flatten(), bins=256, range=[0, 256])
    return histogram

# Fungsi untuk filter sharpening
def sharpening_filter(image):
    # Definisikan kernel untuk sharpening
    kernel = np.array([[0, -1, 0],
                       [-1, 5, -1],
                       [0, -1, 0]])

    # Terapkan filter sharpening menggunakan kernel
    sharpened_image = cv2.filter2D(image, -1, kernel)

    return sharpened_image

# Baca gambar dari file
image = cv2.imread('testprepro-normal.png')

# Terapkan filter sharpening pada gambar
sharpened_image = sharpening_filter(image)

# Hitung histogram gambar asli
```

```
hist_original = cv2.calcHist([image], [0], None, [256], [0, 256])

# Hitung histogram gambar yang telah di-sharpen
hist_sharpened = cv2.calcHist([sharpened_image], [0], None, [256], [0, 256])

# Tampilkan gambar asli dan hasil sharpening
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)  # Membuat subplot pertama
plt.title('Original Image')  # Menambahkan judul
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))  # Menampilkan gambar asli dalam format RGB
plt.axis('off')  # Menghilangkan axis

plt.subplot(1, 2, 2)  # Membuat subplot kedua
plt.title('Sharpened Image')  # Menambahkan judul
plt.imshow(cv2.cvtColor(sharpened_image, cv2.COLOR_BGR2RGB))  # Menampilkan gambar hasil sharpening dalam format RGB
plt.axis('off')  # Menghilangkan axis

plt.show()  # Menampilkan plot

# Tampilkan histogram gambar asli dan hasil sharpening
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)  # Membuat subplot pertama
plt.title('Original Histogram')  # Menambahkan judul
plt.plot(hist_original, color='blue')  # Menampilkan histogram gambar asli dengan warna biru
plt.xlabel('Pixel Intensity')  # Menambahkan label sumbu x
plt.ylabel('Frequency')  # Menambahkan label sumbu y

plt.subplot(1, 2, 2)  # Membuat subplot kedua
plt.title('Sharpened Histogram')  # Menambahkan judul
plt.plot(hist_sharpened, color='blue')  # Menampilkan histogram gambar hasil sharpening dengan warna biru
plt.xlabel('Pixel Intensity')  # Menambahkan label sumbu x
plt.ylabel('Frequency')  # Menambahkan label sumbu y

plt.show()  # Menampilkan plot

# Perhitungan kontras sebelum dan sesudah sharpening
kontras_asli = kalkulasi_kontras(cv2.cvtColor(image, cv2.COLOR_BGR2GRAY))
kontras_sharpened = kalkulasi_kontras(cv2.cvtColor(sharpened_image, cv2.COLOR_BGR2GRAY))

# Menampilkan kontras sebelum dan sesudah sharpening
print("Kontras sebelum sharpening:", kontras_asli)
print("Kontras setelah sharpening:", kontras_sharpened)
```

```
# Distribusi intensitas sebelum dan sesudah sharpening
distribusi_asli = kalkulasi_intensity_distribusi(cv2.cvtColor(image,
cv2.COLOR_BGR2GRAY))
distribusi_sharpened = kalkulasi_intensity_distribusi(cv2.cvtColor(sharpened_image,
cv2.COLOR_BGR2GRAY))

# Plot distribusi intensitas
plt.figure(figsize=(10, 5))

# Menampilkan distribusi intensitas gambar asli
plt.plot(distribusi_asli, color='blue', label='Original')

# Menampilkan distribusi intensitas gambar hasil sharpening
plt.plot(distribusi_sharpened, color='red', label='Sharpened')

plt.title('Intensity Distribution')  # Menambahkan judul
plt.xlabel('Pixel Intensity')  # Menambahkan label sumbu x
plt.ylabel('Frequency')  # Menambahkan label sumbu y
plt.legend()  # Menambahkan legenda untuk membedakan antara original dan hasil
sharpening

plt.show()  # Menampilkan plot
```

```
Kontras sebelum sharpening: 42.70728630316623
Kontras setelah sharpening: 43.28519166535115
```

Menggabungkan Metode Preprocessing

```
# Import library yang diperlukan
import cv2
import os
import numpy as np

# Fungsi untuk equalisasi histogram
def histogram_equalization(image):
    # Konversi gambar ke ruang warna grayscale
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Equalisasi histogram pada gambar grayscale
    equalized_image = cv2.equalizeHist(gray_image)

    return equalized_image

# Fungsi untuk koreksi gamma
```

```
def gamma_correction(image, gamma):
    # Terapkan koreksi gamma pada gambar
    adjusted_image = np.power(image / 255.0, gamma) * 255.0
    return np.uint8(adjusted_image)

# Fungsi utama untuk memproses gambar dalam folder
def main():
    # Path ke folder input dan output gambar
    input_folder = 'C:/Users/Daffa fazly
r/OneDrive/Documents/S4/pcd/projek/Tuberculosis'
    output_folder = 'C:/Users/Daffa fazly
r/OneDrive/Documents/S4/pcd/projek/preproses_gambar_tb'

    # Membaca semua nama file dalam folder input
    image_files = os.listdir(input_folder)

    # Loop melalui setiap file gambar dalam folder
    for image_file in image_files:
        # Path lengkap ke file gambar
        image_path = os.path.join(input_folder, image_file)

        # Baca gambar dari file
        image = cv2.imread(image_path)

        # Cek apakah gambar berhasil dibaca
        if image is None:
            print(f"Error reading {image_file}, skipping.")
            continue

        # Terapkan histogram equalization pada gambar
        equalized_image = histogram_equalization(image)

        # Terapkan gamma correction pada gambar hasil histogram equalization
        gamma = 0.89  # Contoh nilai gamma
        gamma_corrected_image = gamma_correction(equalized_image, gamma)

        # Simpan gambar hasil ke folder output
        output_path = os.path.join(output_folder, 'gamma_equalized_' + image_file)
        cv2.imwrite(output_path, gamma_corrected_image)

        print(f"Gamma corrected and equalized {image_file} saved as {output_path}")

# Jalankan fungsi utama jika script ini dieksekusi sebagai program utama
if __name__ == "__main__":
    main()
```

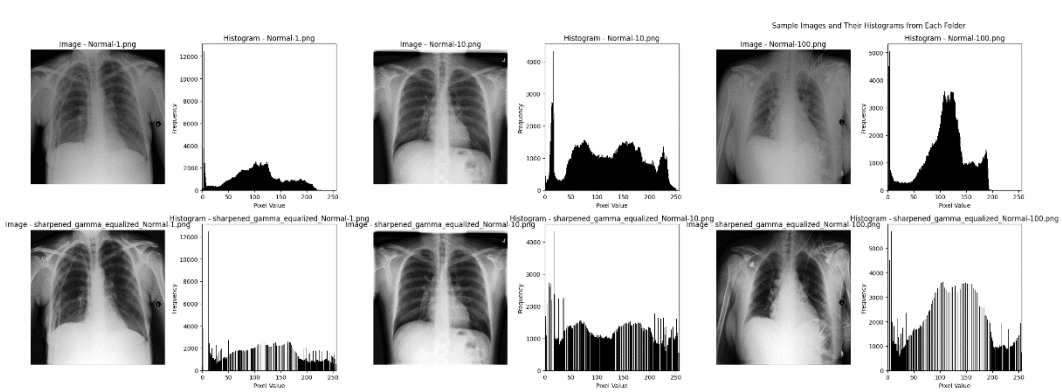
Visualisasi Hasil Preprocessing

```
# Import library yang diperlukan
import os
```

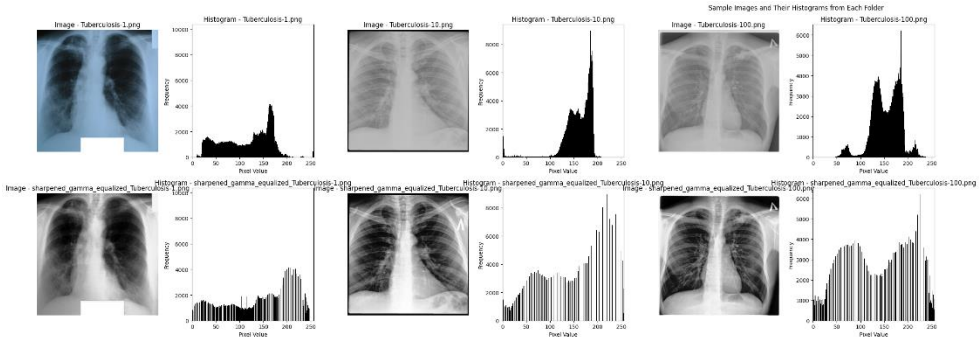
```
import matplotlib.pyplot as plt
import cv2

# Atur ukuran figure untuk menampilkan gambar
plt.figure(figsize=(40, 20))

# Tentukan folder yang akan digunakan
folders = [
    r'C:/Users/Daffa fazly r/OneDrive/Documents/S4/pcd/projek/normal',
    r'C:/Users/Daffa fazly r/OneDrive/Documents/S4/pcd/projek/preproses_gambar_normal'
]

# Iterasi melalui setiap folder dan menampilkan 3 gambar pertama dari masing-masing folder
for j, folder in enumerate(folders):
    files = os.listdir(folder)
    for i, file in enumerate(files[:3]):  # Mengambil 3 file pertama
        image_path = os.path.join(folder, file)
        img = cv2.imread(image_path)
        img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)  # Konversi dari BGR ke RGB
        gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)  # Konversi ke grayscale

        # Plot gambar
        ax = plt.subplot(2, 5, j * 5 + i + 1)  # Membuat subplot
        ax.title.set_text(f"Image - {file}")
        plt.imshow(img_rgb)
        plt.axis('off')  # Sembunyikan sumbu

# Menampilkan judul utama dan menyesuaikan layout
plt.suptitle("Sample Images from Each Folder")
plt.tight_layout()
plt.show()
```

```
# Import library yang diperlukan
import os
import matplotlib.pyplot as plt
import cv2
```

```python
# Atur ukuran figure untuk menampilkan gambar
plt.figure(figsize=(40, 9))

# Tentukan folder yang akan digunakan
folders = [
    r'C:/Users/Daffa fazly r/OneDrive/Documents/S4/pcd/projek/normal',
    r'C:/Users/Daffa fazly r/OneDrive/Documents/S4/pcd/projek/preproses_gambar_normal'
]

# Iterasi melalui setiap folder dan menampilkan 3 gambar pertama dari masing-masing folder
for j, folder in enumerate(folders):
    files = os.listdir(folder)
    for i, file in enumerate(files[:3]):  # Mengambil 3 file pertama
        image_path = os.path.join(folder, file)
        img = cv2.imread(image_path)
        img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)  # Konversi dari BGR ke RGB
        gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)  # Konversi ke grayscale

        # Plot gambar
        ax = plt.subplot(2, 10, j * 10 + i * 2 + 1)  # Membuat subplot untuk gambar
        ax.title.set_text(f"Image - {file}")
        plt.imshow(img_rgb)
        plt.axis('off')  # Sembunyikan sumbu

        # Plot histogram
        ax = plt.subplot(2, 10, j * 10 + i * 2 + 2)  # Membuat subplot untuk histogram
        ax.title.set_text(f"Histogram - {file}")
        plt.hist(gray_img.ravel(), bins=256, range=[0, 256], color='black')
        plt.xlim([0, 256])
        plt.xlabel('Pixel Value')
        plt.ylabel('Frequency')

# Menampilkan judul utama dan menyesuaikan layout
plt.suptitle("Sample Images and Their Histograms from Each Folder")
plt.tight_layout()
plt.show()
```



```python
# Import library yang diperlukan
import os
import matplotlib.pyplot as plt
import cv2

# Atur ukuran figure untuk menampilkan gambar
plt.figure(figsize=(40, 9))

# Tentukan folder yang akan digunakan
folders = [
    r'C:/Users/Daffa fazly r/OneDrive/Documents/S4/pcd/projek/Tuberculosis',
    r'C:/Users/Daffa fazly r/OneDrive/Documents/S4/pcd/projek/preproses_gambar_tb'
]

# Iterasi melalui setiap folder dan menampilkan 3 gambar pertama dari masing-masing folder
for j, folder in enumerate(folders):
    files = os.listdir(folder)
    for i, file in enumerate(files[:3]):  # Mengambil 3 file pertama
        image_path = os.path.join(folder, file)
        img = cv2.imread(image_path)
        img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)  # Konversi dari BGR ke RGB
        gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)  # Konversi ke grayscale

        # Plot gambar
        ax = plt.subplot(2, 10, j * 10 + i * 2 + 1)  # Membuat subplot untuk gambar
        ax.title.set_text(f"Image - {file}")
        plt.imshow(img_rgb)
        plt.axis('off')  # Sembunyikan sumbu

        # Plot histogram
        ax = plt.subplot(2, 10, j * 10 + i * 2 + 2)  # Membuat subplot untuk histogram
        ax.title.set_text(f"Histogram - {file}")
        plt.hist(gray_img.ravel(), bins=256, range=[0, 256], color='black')
        plt.xlim([0, 256])
        plt.xlabel('Pixel Value')
        plt.ylabel('Frequency')

# Menampilkan judul utama dan menyesuaikan layout
plt.suptitle("Sample Images and Their Histograms from Each Folder")
plt.tight_layout()
plt.show()
```



Build

```
import numpy  as np
import pandas as pd
from skimage.transform import resize
from skimage.io import imread
import matplotlib.pyplot as plt
import os
```

Inisialisasi Flatten dan Target

```
# Definisikan daftar kategori gambar
Categories = ["preproses_gambar_normal", "preproses_gambar_tb"]

# Inisialisasi daftar kosong untuk menyimpan data gambar yang sudah di-flatten
flat_data_arr = []

# Inisialisasi daftar kosong untuk menyimpan label/target dari data gambar
target_arr = []
```

Load File Gambar

```
datadir = "C:/Users/Daffa fazly r/OneDrive/Documents/S4/pcd/projek"

import os
from skimage.io import imread
from skimage.transform import resize

# Loop melalui setiap kategori dalam daftar Categories
for category in Categories:
    print(f"Loading category: {category}")

    # Gabungkan jalur direktori data dengan nama kategori
    path = os.path.join(datadir, category)

    # Loop melalui setiap gambar dalam direktori kategori
    for img in os.listdir(path):
        # Baca gambar menggunakan skimage.io.imread
        img_array = imread(os.path.join(path, img))

        # Ubah ukuran gambar menjadi 150x150 piksel dengan 3 channel warna (RGB)
        img_resized = resize(img_array, (150, 150, 3))
```

```
            # Flatten gambar yang telah diubah ukurannya dan tambahkan ke flat_data_arr
            flat_data_arr.append(img_resized.flatten())

            # Tambahkan indeks kategori sebagai label ke target_arr
            target_arr.append(Categories.index(category))

        # Cetak pesan setelah selesai memuat kategori
        print(f"Loaded category: {category}")
```

```
Loading category: preproses_gambar_normal
Loaded category: preproses_gambar_normal
Loading category: preproses_gambar_tb
Loaded category: preproses_gambar_tb
```

Mengkonversi Data Gambar Menjadi Array

```
# Konversi daftar data gambar yang telah di-flatten menjadi array numpy
flat_data = np.array(flat_data_arr)

# Konversi daftar label/target menjadi array numpy
target = np.array(target_arr)
```

Membuat DataFrame dari data gambar yang telah diubah menjadi array

```
# Buat DataFrame dari data gambar yang telah di-flatten (flat_data)
df = pd.DataFrame(flat_data)

# Tambahkan kolom "Target" ke DataFrame, yang berisi label/target untuk setiap gambar
df["Target"] = target
```

```
df.head()
```

	0	1	2	3	4	5	6	\
0	0.043254	0.043254	0.043254	0.047327	0.047327	0.047327	0.047059	
1	0.969315	0.969315	0.969315	0.933949	0.933949	0.933949	0.897341	
2	0.007604	0.007604	0.007604	0.011378	0.011378	0.011378	0.011836	
3	0.999642	0.999642	0.999642	0.998294	0.998294	0.998294	0.995814	
4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	

	7	8	9	...	67491	67492	67493	67494	\
0	0.047059	0.047059	0.047059	...	0.065072	0.065072	0.065072	0.065103	
1	0.897341	0.897341	0.870849	...	0.031008	0.031008	0.031008	0.089056	
2	0.011836	0.011836	0.011836	...	0.046618	0.046618	0.046618	0.061719	
3	0.995814	0.995814	0.993645	...	0.062592	0.062592	0.062592	0.105573	
4	0.000000	0.000000	0.000000	...	0.705379	0.705379	0.705379	0.884770	

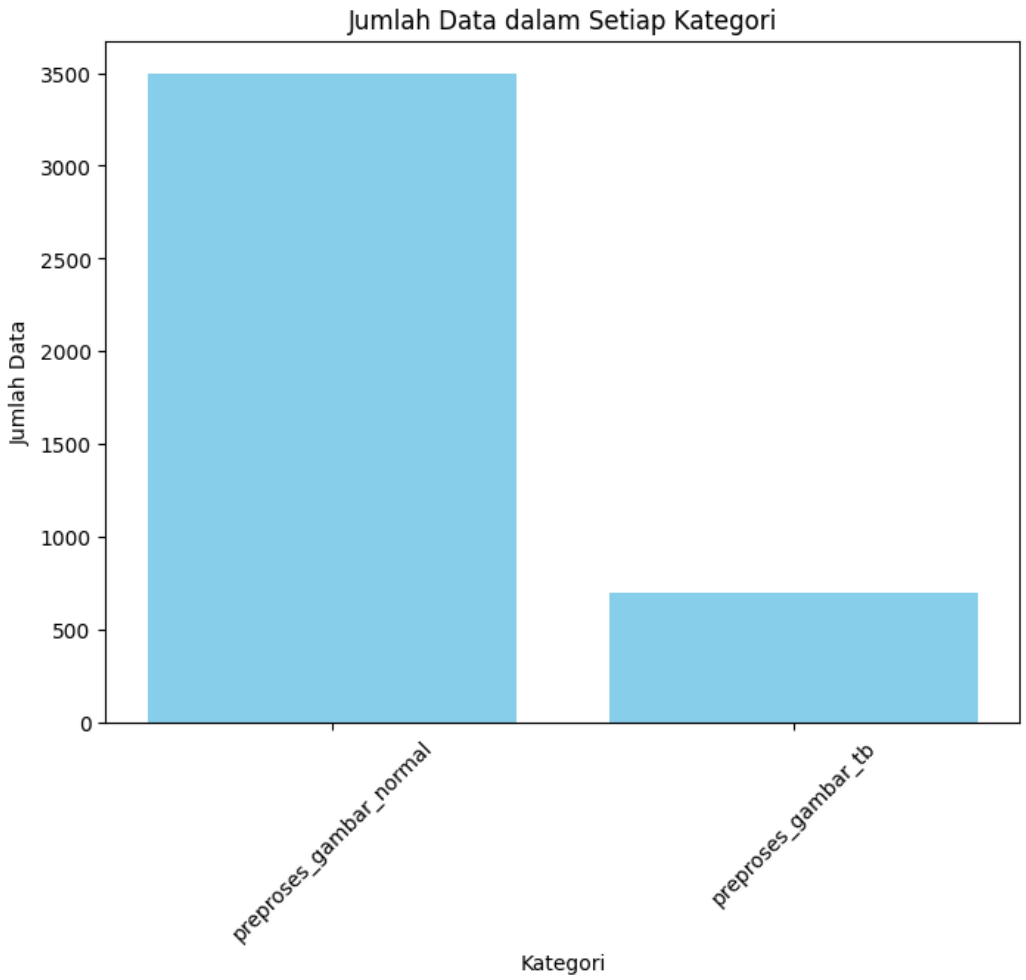
	67495	67496	67497	67498	67499	Target
0	0.065103	0.065103	0.064841	0.064841	0.064841	0
1	0.089056	0.089056	0.331749	0.331749	0.331749	0
2	0.061719	0.061719	0.062760	0.062760	0.062760	0
3	0.105573	0.105573	0.163265	0.163265	0.163265	0

```
4  0.884770  0.884770  0.959035  0.959035  0.959035       0

[5 rows x 67501 columns]
```

Inisialisasi X dan y

```
# X berisi fitur-fitur (kolom-kolom) dari DataFrame df kecuali kolom 'Target'
X = df.drop('Target', axis=1)

# y berisi hanya kolom 'Target'
y = df[['Target']]
```

```
category_counts = df['Target'].value_counts()

# Plot visualisasi jumlah data
plt.figure(figsize=(8, 6))
plt.bar(Categories, category_counts, color='skyblue')
plt.title('Jumlah Data dalam Setiap Kategori')
plt.xlabel('Kategori')
plt.ylabel('Jumlah Data')
plt.xticks(rotation=45)
plt.show()
```




```
X.head()
```

```
      0         1         2         3         4         5         6       \
0  0.043254  0.043254  0.043254  0.047327  0.047327  0.047327  0.047059
```

```
1  0.969315  0.969315  0.969315  0.933949  0.933949  0.933949  0.897341
2  0.007604  0.007604  0.007604  0.011378  0.011378  0.011378  0.011836
3  0.999642  0.999642  0.999642  0.998294  0.998294  0.998294  0.995814
4  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000

       7         8         9      ...     67490     67491     67492     67493  \
0  0.047059  0.047059  0.047059   ...   0.065071  0.065072  0.065072  0.065072
1  0.897341  0.897341  0.870849   ...   0.027455  0.031008  0.031008  0.031008
2  0.011836  0.011836  0.011836   ...   0.045454  0.046618  0.046618  0.046618
3  0.995814  0.995814  0.993645   ...   0.054914  0.062592  0.062592  0.062592
4  0.000000  0.000000  0.000000   ...   0.566504  0.705379  0.705379  0.705379

       67494     67495     67496     67497     67498     67499
0  0.065103  0.065103  0.065103  0.064841  0.064841  0.064841
1  0.089056  0.089056  0.089056  0.331749  0.331749  0.331749
2  0.061719  0.061719  0.061719  0.062760  0.062760  0.062760
3  0.105573  0.105573  0.105573  0.163265  0.163265  0.163265
4  0.884770  0.884770  0.884770  0.959035  0.959035  0.959035

[5 rows x 67500 columns]
```

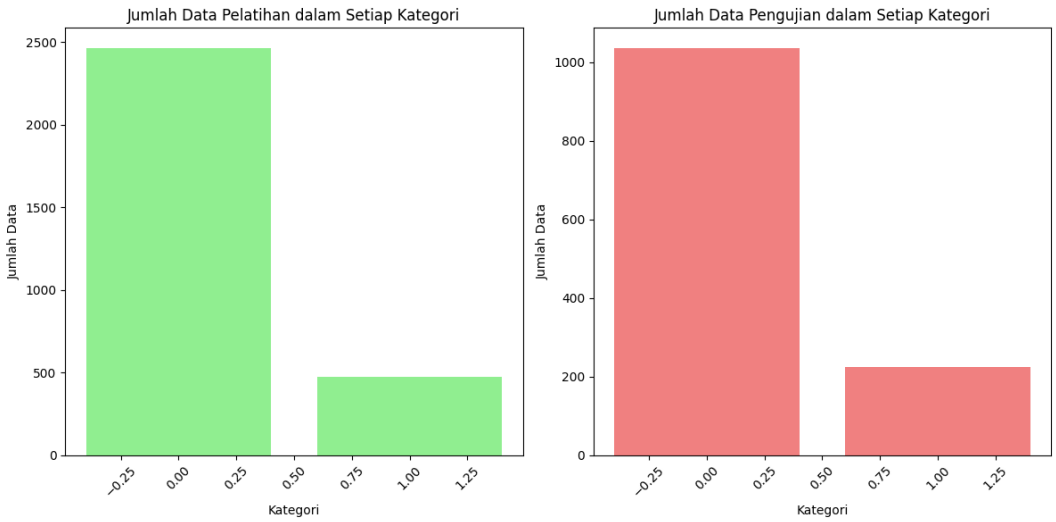
```
y.head()
```

```
   Target
0       0
1       0
2       0
3       0
4       0
```

Spliting data

```
from sklearn import svm

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=123)

# Menghitung jumlah data dalam setiap kategori setelah pemotongan
train_category_counts = y_train['Target'].value_counts()
test_category_counts = y_test['Target'].value_counts()

# Mendapatkan kategori unik dari target
Categories = y_train['Target'].unique()

# Plot visualisasi jumlah data untuk set data pelatihan dan pengujian
if not train_category_counts.empty and not test_category_counts.empty:
    plt.figure(figsize=(12, 6))

    # Plot untuk set data pelatihan
```

```
    plt.subplot(1, 2, 1)
    plt.bar(Categories, train_category_counts, color='lightgreen')
    plt.title('Jumlah Data Pelatihan dalam Setiap Kategori')
    plt.xlabel('Kategori')
    plt.ylabel('Jumlah Data')
    plt.xticks(rotation=45)

    # Plot untuk set data pengujian
    plt.subplot(1, 2, 2)
    plt.bar(Categories, test_category_counts, color='lightcoral')
    plt.title('Jumlah Data Pengujian dalam Setiap Kategori')
    plt.xlabel('Kategori')
    plt.ylabel('Jumlah Data')
    plt.xticks(rotation=45)

    plt.tight_layout()
    plt.show()
else:
    print("Tidak ada data untuk plotting.")
```




```
# Check the shape of the training and testing sets
print("Train images shape:", X_train.shape)
print("Test images shape:", X_test.shape)
print("Train labels shape:", y_train.shape)
print("Test labels shape:", y_test.shape)
```

```
Train images shape: (2940, 67500)
Test images shape: (1260, 67500)
Train labels shape: (2940, 1)
Test labels shape: (1260, 1)
```

SVM

```
from sklearn.metrics import classification_report

svc = svm.SVC(kernel="poly", probability=True)
svc.fit(X_train, y_train)
```

```python
# Cetak akurasi model
accuracy = svc.score(X_test, y_test)
print(f"SUPPORT VECTOR MACHINE ACCURACY: {accuracy}")

# Prediksi menggunakan model yang telah dilatih
y_pred = svc.predict(X_test)

# Buat dan cetak classification report
report = classification_report(y_test, y_pred, target_names=["Normal",
"Tuberculosis"])
print(report)
```

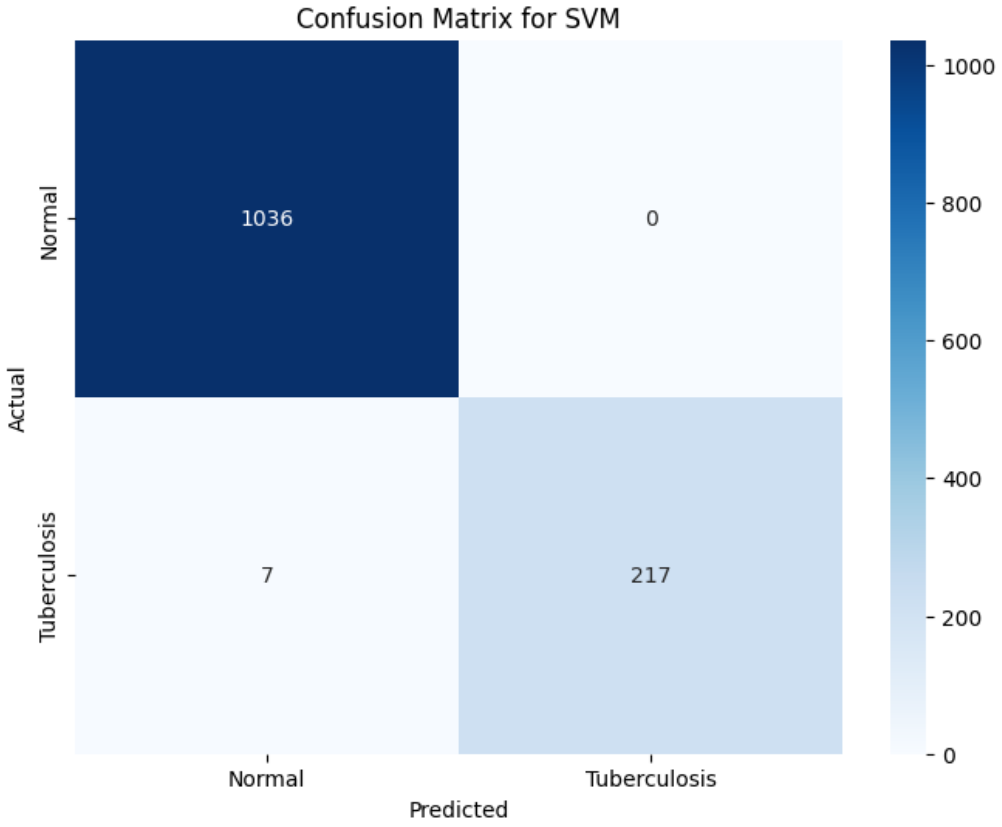
```
C:\Users\Daffa fazly
r\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache
\local-packages\Python310\site-packages\sklearn\utils\validation.py:1143:
DataConversionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

```
SUPPORT VECTOR MACHINE ACCURACY: 0.9944444444444445
              precision    recall  f1-score   support

      Normal       0.99      1.00      1.00      1036
Tuberculosis       1.00      0.97      0.98       224

    accuracy                           0.99      1260
   macro avg       1.00      0.98      0.99      1260
weighted avg       0.99      0.99      0.99      1260
```

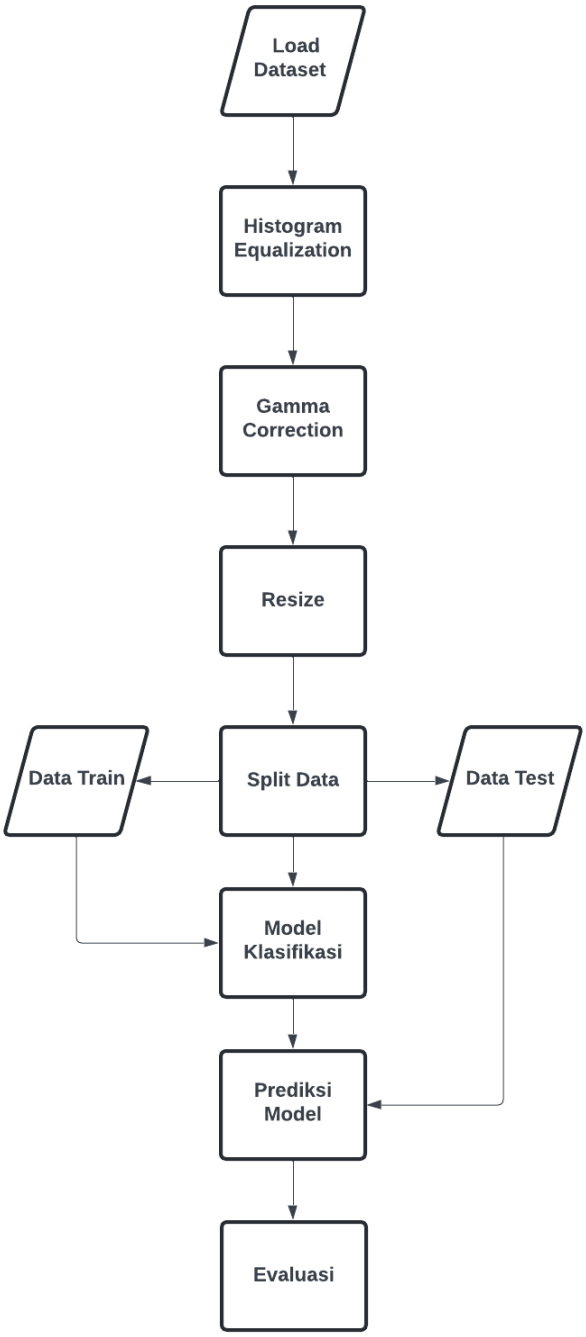
```
svc.predict(X_test)
```

```
array([0, 0, 1, ..., 0, 0, 0])
```

```python
from sklearn.metrics import confusion_matrix

conf_matrix_svm = confusion_matrix(y_test, y_pred)

import matplotlib.pyplot as plt
import seaborn as sns

# Membuat heatmap dari matriks kebingungan
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix_svm, annot=True, cmap='Blues', fmt='d', xticklabels=["Normal",
"Tuberculosis"], yticklabels=["Normal", "Tuberculosis"])
plt.title('Confusion Matrix for SVM')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```


Confusion Matrix for SVM
Normal
Tuberculosis
Actual
1036
0
7
217
Normal
Tuberculosis
Predicted
1000
800
600
400
200
0

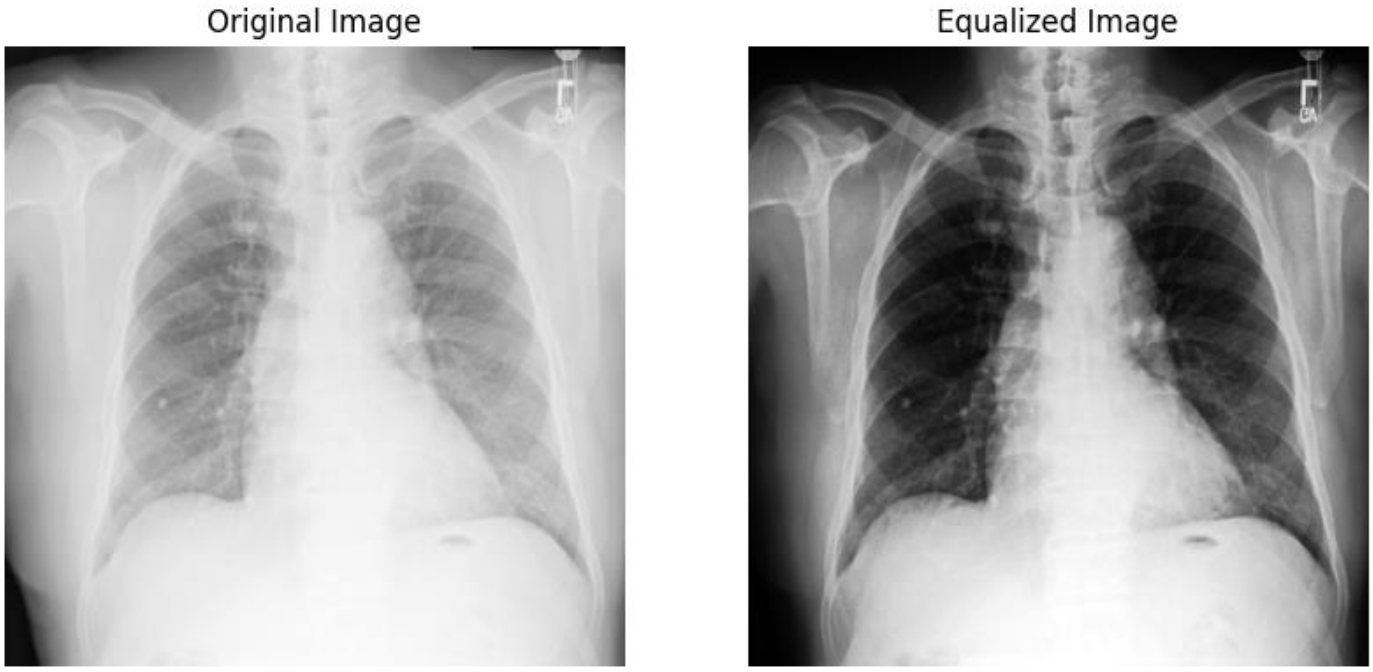
1. Flowchart


Load Dataset
Histogram Equalization
Gamma Correction
Resize
Split Data
Data Train
Data Test
Model Klasifikasi
Prediksi Model
Evaluasi

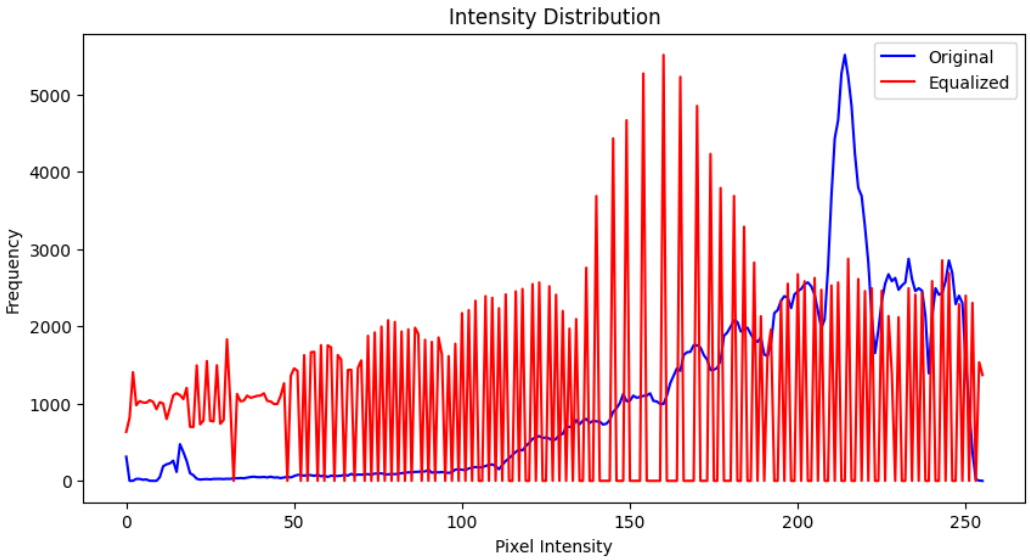
BAB III

HASIL & PEMBAHASAN

3.1 Hasil & Pembahasan Histogram Ekualisasi



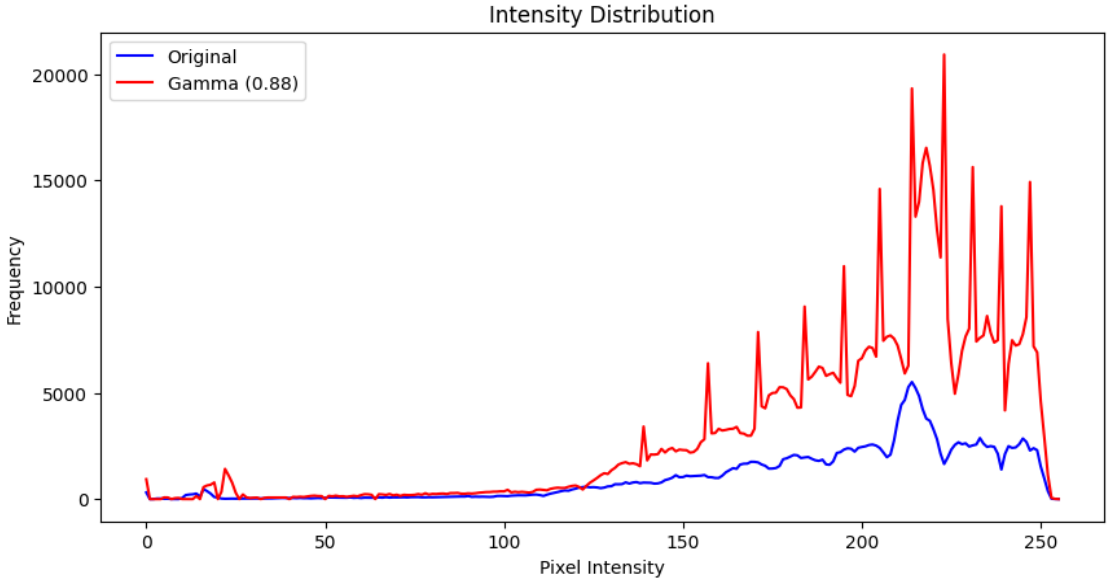
Hasil ekualisasi gambar menunjukan kontras yang bertambah, Kontras sebelum ekualisasi histogram adalah 42.70728630316623 dan Kontras setelah ekualisasi histogram adalah 74.0295870571604, gambar X-ray menunjukan semakin tajam. Namun sebenarnya dengan kita menambahkan kontras pada paru-paru bisa saja menghilangkan identitas Tuberkulosisnya, namun tujuan kita adalah dengan memperjelas anatomi paru paru (antara sekeleton dan organ). Distribusi intensitas piksel tersebar lebih merata di seluruh rentang intensitas. Hal ini menunjukkan bahwa equalized histogram telah berhasil mendistribusikan ulang intensitas piksel, sehingga meningkatkan kontras gambar menyelurh. Berikut adalah grafik distribusi intensitas kontras dengan ekualisasi histogram:



Sumbu X: Menunjukkan nilai intensitas piksel, mulai dari 0 (paling gelap) hingga 255 (paling terang).
Sumbu Y: Menunjukkan frekuensi piksel dengan intensitas tertentu. Semakin tinggi batang histogram, semakin banyak piksel dengan intensitas tersebut.

3.2 Hasil dan Pembahasan Gamma Correction

Hasil Ekualisasi Kontras sebelum ekualisasi histogram adalah 42.70728630316623, Kontras setelah ekualisasi histogram adalah 40.18302847622345. Dengan memanipulasi nilai gamma akan menujukan perubahan intensitas saturasi kurang dari 1 akan menjadi lebih gelap (semakin mendekati 0), dan lebih dari 1 akan menjadi lebih terang (semakin mendekati 255). Berikut adalah Histogram distribusi intensitas saturasinya:



Sumbu X: Menunjukkan nilai intensitas piksel, mulai dari 0 (paling gelap) hingga 255 (paling terang).
Sumbu Y: Menunjukkan frekuensi piksel dengan intensitas tertentu. Semakin tinggi batang histogram, semakin banyak piksel dengan intensitas tersebut.

Hasil Akurasi yang didapat akan terjadi perubahan dengan mengubah nilai Gamma dan juga mengelimininasi Pre-Processing data menggunakan Sharpening, Alasan utama kami mengevaluasi Output setelah sharpening dan dengan tanpa sharpening adalah karena hasil akurasi menunjukan penurunan sebesar

0.8 % (dengan sharpening = 97.3%) dibandingkan dengan dataset tanpa pre-processing (tanpa sharpening = 98.1%). Berikut hasil evaluasi kami dengan memanipulasi nilai gamma:

No	Nilai Gamma	Akurasi	Keterangan
1	TB&Normal= 1.4	97.3	Terjadi Penurunan
2	TB& Normal= 0.80	97.6	Terjadi Penurunan
3	TB&Normal= 0.84	98.0	Terjadi Penurunan
4.	TB&Normal= 0.88	98.0	Terjadi Penurunan
5.	TB&Normal= 0.90	98.0	Terjadi Penurunan
6.	TB&Normal= 0.98	98.2	Terjadi Peningkatan
7.	TB= 0.89, Normal= 1.1	99.5	Terjadi peningkatan Optimal

Rata-Rata Running Time TB&Normal sama: ± 15 Menit

Rata-Rata Running Time TB&Normal Berbeda: ± 7 Menit

BAB IV
PENUTUP

4.1 Kesimpulan

Dengan meningkatkan kualitas citra gambar paru-paru dan mengembangkan metode deteksi tuberkulosis yang lebih baik terdapat beberapa komparasi kualitas gambar dari ketiga metode yang digunakan yaitu histogram ekualisasi dan koreksi gamma terdapat beberapa hasil akurasi yang turun dan naik, pada tahap pre-processing dari proses pengukuran parameter menggunakan koreksi gamma mulai dari 0.80-1.4 terjadi peningkatan akurasi pada nilai gamma 0.98 sebesar 98.2 dan terjadi penurunan akurasi pada nilai gamma 1.4 sebesar 97.3 dengan rata-rata running time sekitar 15 menit. Merubah nilai gamma yang berbeda dataset memberikan output yang lebih optimal, terjadi peningkatan akurasi pada nilai gamma dataset TB = 0.89 dan Normal = 1.1 sebesar 99.5, dengan rata-rata running time 7 menit. Sedangkan tanpa pre-preprocessing hasil akurasi sebesar 98.1. Ini telah menyelesaikan permasalahan dimana saat melakukan pemaparan projek hasil akurasi terjadi peningkatan akurasi pada gambar paru-paru penderita tuberkulosis.

DAFTAR PUSTAKA

[1] W. Y. Nyein Naing and Z. Z. Htike, "Advances in Automatic Tuberculosis Detection in Chest X-Ray Images," *Signal Image Process*, vol. 5, no. 6, pp. 41–53, Dec. 2014, doi: 10.5121/sipij.2014.5604.

[2] P. Bhardwaj and A. Kaur, "Impact of image enhancement methods on lung disease diagnosis using x-ray images," *International Journal of Information Technology (Singapore)*, vol. 15, no. 7, pp. 3521–3526, Oct. 2023, doi: 10.1007/s41870-023-01409-1.

[3] T. Rahman *et al.*, "Exploring the effect of image enhancement techniques on COVID-19 detection using chest X-ray images," *Comput Biol Med*, vol. 132, May 2021, doi: 10.1016/j.compbiomed.2021.104319.

[4] S. Rahman, M. M. Rahman, M. Abdullah-Al-Wadud, G. D. Al-Quaderi, and M. Shoyaib, "An adaptive gamma correction for image enhancement," *EURASIP J Image Video Process*, vol. 2016, no. 1, Dec. 2016, doi: 10.1186/s13640-016-0138-1.