BAB IV
PERANCANGAN

4.1 Analisis Sistem Terdahulu

Penelitian terdahulu yang dijadikan pembanding dalam penelitian ini dari jurnal yang berjudul "Aplikasi Pengolahan Citra untuk Menentukan Tingkat Kematangan Buah Pisang dengan Menggunakan Ruang Warna Hue" oleh Mutiara Risvita B dan Ericks Rahmat Swedia. Dalam penelitian tersebut, peneliti menentukan tingkat kematangan buah pisang dengan menggunakan ruang warna hue. Warna hue yang digunakan terdiri dari warna merah, kuning, hijau, cyan, biru, dan magenta. Nilai warna hue yang didapatkan dari objek pisang yang digunakan merupakan hasil proses pencari melalui aplikasi Adobe Photoshop. Buah pisang yang dijadikan objek penelitian berupa buah pisang satuan. Hasil pemeriksaan tingkat kematangan terbagi menjadi 2 jenis yaitu buah pisang matang dan buah pisang belum matang (B & Swedia, 2017).

4.2 Spesifikasi Kebutuhan Sistem Baru

4.2.1 Spesifikasi Kebutuhan Perangkat Keras

Penelitian ini menggunakan perangkat keras yang terdiri dari prosesor, *harddisk*, dan *memory*. Spesifikasi kebutuhan perangkat keras tertera pada tabel berikut.

Tabel 4.1 Spesifikasi Kebutuhan Perangkat Keras

No	Perangkat Keras	Keterangan
1	Prosesor	AMD Ryzen 3
2	*Harddisk*	512 GB SSD
3	*Memory*	8 GB RAM

4.2.2 Spesifikasi Kebutuhan Perangkat Lunak

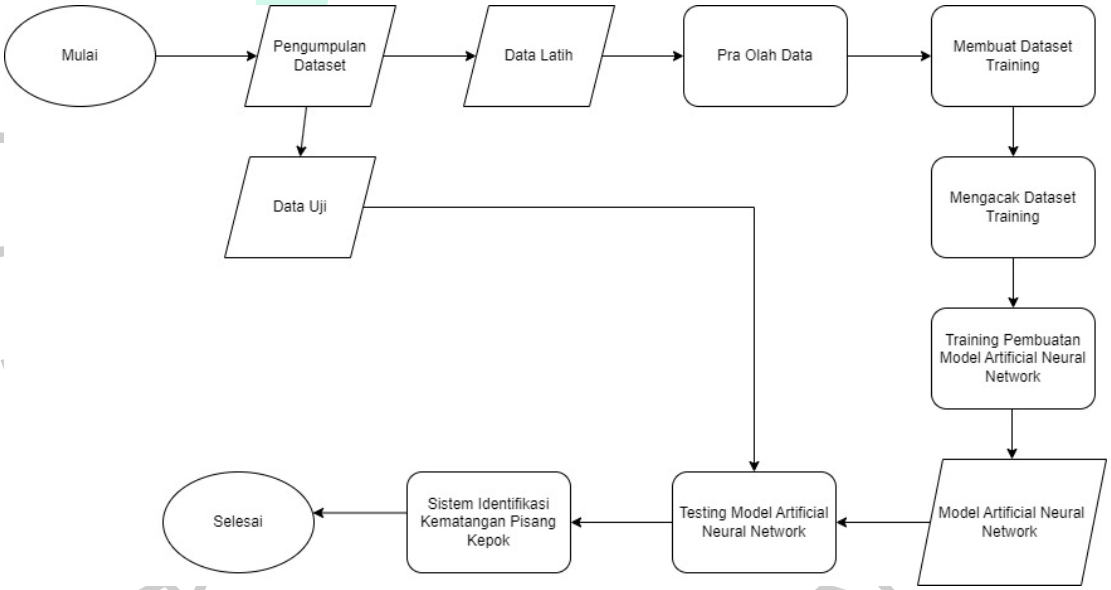
Penelitian ini menggunakan perangkat lunak yang terdiri dari sistem operasi dan aplikasi editor. Spesifikasi kebutuhan perangkat lunak yang digunakan tertera pada tabel berikut.

Tabel 4.2 Spesifikasi Kebutuhan Perangkat Lunak

No	Perangkat	Keterangan
1	Windows 11 64-bit	Sistem operasi yang digunakan.
2	Google Colab	Aplikasi untuk membangun aplikasi yang menentukan gambar pisang atau bukan pisang.
3	IDE Pycharm	Aplikasi untuk membangun aplikasi utama dalam penelitian.
4	Figma	*Platform* yang digunakan untuk desain *mockup* perancangan aplikasi.

4.3 Perancangan Sistem

Pada tahap perancangan sistem identifikasi kematangan pisang kepok yang berfungsi untuk mengenali kematangan pisang kepok berdasarkan 3 kelas yaitu pisang kepok belum matang, pisang kepok matang, dan pisang kepok terlalu matang atau busuk terdiri dari beberapa tahap. Berikut gambar dan penjelasan mengenai tahapan pembuatan aplikasi identifikasi kematangan pisang kepok.



Gambar 4.1 Tahapan Pembuatan Model *Artificial Neural Network*

4.3.1 Pengumpulan Dataset

Pengumpulan dataset merupakan langkah awal yang dilakukan oleh peneliti dalam proses pembuatan aplikasi identifikasi kematangan pisang kepok. Dataset yang dikumpulkan untuk sebagai data latih terdiri dari 300 gambar pisang kepok yang terbagi menjadi 3 kelas, yaitu 100 gambar pisang kepok belum matang, 100 gambar pisang kepok matang, dan 100 gambar pisang kepok terlalu matang atau busuk. Sedangkan dataset yang dikumpulkan untuk sebagai data uji terdiri dari 30

gambar pisang kepok yang terbagi menjadi 3 kelas, yaitu 10 gambar pisang kepok belum matang, 10 gambar pisang kepok matang, dan 10 gambar pisang kepok terlalu matang atau busuk. Jadi, peneliti mengumpulkan dataset sebanyak 330 gambar pisang kepok.

Tabel 4.3 Contoh Gambar Dataset Aplikasi Identifikasi Kematangan Pisang Kepok

No	Gambar	Keterangan
1		Contoh gambar pisang kepok belum matang yang akan dianotasi dengan label belum matang.
2		Contoh gambar pisang kepok matang yang akan dianotasi dengan label matang.

3		Contoh gambar pisang kepok terlalu matang atau busuk yang akan dianotasi dengan label terlalu matang atau busuk.

4.3.2 Pra Olah Data

Pada tahap ini peneliti melakukan beberapa proses terhadap data latih yang dikumpulkan. Berikut penjelasan proses pra olah dataset untuk pembuatan aplikasi identifikasi tingkat kematangan pisang kepok.

1. Peneliti mengubah rasio aspek setiap gambar dari ukuran 16:9 menjadi ukuran 1:1. Hal ini dilakukan agar gambar yang digunakan fokus pada objek pisang kepok.

Gambar 4.2 Contoh Gambar Pisang Kepok dengan Rasio Aspek 16:9

Gambar 4.3 Contoh Gambar Pisang Kepok dengan Rasio Aspek 1:1

2. Peneliti mengubah ukuran gambar menjadi 500 piksel x 500 piksel dengan menggunakan sebuah aplikasi.
3. Peneliti mengubah nama *file* asli menjadi a_0 sampai a_99 untuk gambar pisang kepok belum matang, b_0 sampai b_99 untuk gambar pisang kepok matang, dan c_0 sampai c_99 untuk pisang kepok terlalu matang atau busuk.
4. Peneliti mengubah nama *file* dari 300 gambar pisang kepok tersebut menjadi pisang ori (0) sampai pisang ori (299).
5. Latar dari 300 gambar pisang kepok tersebut diubah menjadi berwarna putih dan dinamakan pisang lp (0) sampai pisang lp (299). Hal ini dilakukan agar

pada saat identifikasi kematangan pisang kepok, hanya warna pada objek pisang kepok yang diproses sebagai penentuan kematangan pisang kepok.

Gambar 4.4 Contoh Gambar Pisang Kepok dengan Latar Berwarna Putih

4.3.3 Membuat Dataset *Training*

Pada tahap ini peneliti membuat *dataset inputs* dengan melakukan pengubahan dataset berupa gambar pisang kepok berlatar putih menjadi *file* npy. *File* npy berisi 300 baris dan 5 kolom yang terdiri dari nomor gambar (pada kolom 0), nilai persentase piksel cenderung berwarna merah (pada kolom 1), nilai persentase piksel cenderung berwarna hijau (pada kolom 2), nilai persentase piksel cenderung berwarna biru (pada kolom 3), dan nilai persentase piksel gelap (pada kolom 4).

Selain itu, peneliti juga membuat *dataset labels* berupa *file* npy yang berisi 300 baris dan 4 kolom. *Dataset labels* berisi angka (1,0,0) pada baris pertama sampai ke-99 yang diasumsikan bahwa gambar pisang kepok pada indeks 0 sampai indeks 99 adalah pisang kepok belum matang, angka (0,1,0) pada baris ke-100 sampai ke-199 yang diasumsikan bahwa gambar pisang kepok pada indeks 100 sampai indeks 199 adalah pisang kepok matang, dan angka (0,0,1) pada baris ke-200 sampai ke-299 yang diasumsikan bahwa gambar pisang kepok pada indeks 200 sampai indeks 299 adalah pisang kepok terlalu matang atau busuk.

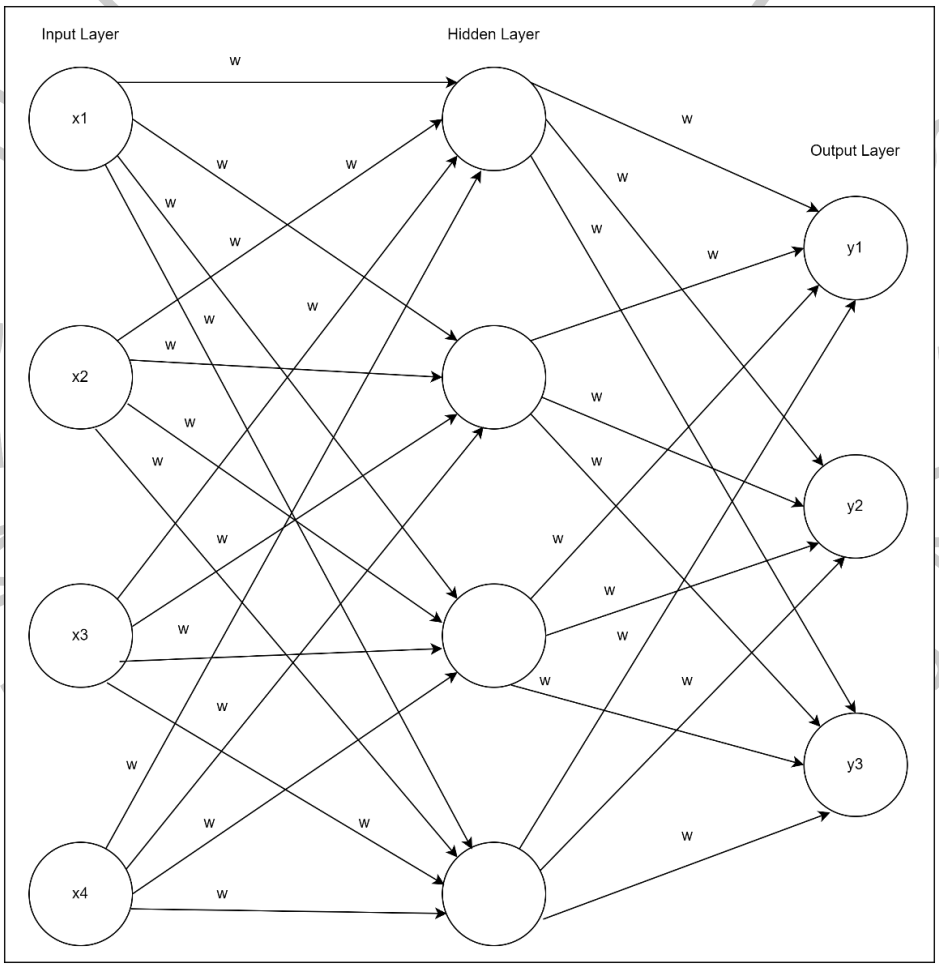
4.3.4 Mengacak Dataset *Training*

Pada tahap ini *file dataset inputs* dan *dataset labels* diacak hingga tak berurutan seperti awal posisi barisnya. Namun, setiap baris yang ada pada *dataset inputs* selalu berpasangan pada *dataset labels*. Hal ini dilakukan untuk

membuktikan bahwa model *Artificial Neural Network* yang dibuat berjalan dengan baik pada tahap identifikasi kematangan pisang kepok. Dataset yang telah diacak ini yang digunakan pada proses pelatihan model *Artificial Neural Network.*

4.3.5 Pembuatan Model dengan Algoritma *Artificial Neural Network*

Pada tahap ini, peneliti melakukan pengembangan atau membuat aplikasi identifikasi kematangan pisang kepok menggunakan algoritma *Artificial Neural Network.* Gambar di bawah ini adalah arsitektur *Artificial Neural Network* dari aplikasi yang akan dibuat.



Gambar 4.5 Arsitektur *Artificial Neural Network* dari Aplikasi Identifikasi Kematangan Pisang Kepok

Berdasarkan gambar arsitektur *Artificial Neural Network* di atas menunjukkan bahwa dalam menerapkan algoritma *Artificial Neural Network* pada aplikasi identifikasi kematangan pisang kepok membutuhkan beberapa hal berikut.

1. Pada *input layer* dibutuhkan 4 variabel yang terdiri dari nilai persentase piksel berwarna cenderung merah, nilai persentase piksel berwarna cenderung hijau, nilai persentase piksel berwarna cenderung biru, dan nilai persentase piksel gelap. Keempat variabel tersebut didapatkan dari nilai RGB gambar pisang kepok yang akan menjadi data *input* pada aplikasi. Data *input* yang digunakan merupakan hasil dari *dataset inputs* yang telah diacak.
2. *Hidden layer* ditetapkan sebanyak 1 *hidden layer* dan memiliki 4 *neuron*.
3. Pada *output layer* dibutuhkan 3 variabel untuk mempresentasikan 3 kelas yang terdiri dari pisang kepok belum matang, pisang kepok matang, dan pisang kepok terlalu matang atau busuk.

Setelah menentukan variabel dari *input layer*, *hidden layer*, dan *output layer* serta telah menyiapkan *dataset inputs* maka dapat melakukan pemodelan. Pembuatan model dengan algoritma *Artificial Neural Network* menggunakan aplikasi IDE Pycharm. Berikut merupakan implementasi kode program pada pembuatan model aplikasi identifikasi kematangan pisang kepok.

```
import numpy as np
import matplotlib.pyplot as plt
from a2_ANN import latar_putih, create_dataset
print("ANN Training & Testing")
```

Gambar 4.6 Potongan Kode Program 1

Potongan kode program pada Gambar 4.6 berfungsi untuk melakukan *import library numpy*, *library matplotlib* dan memanggil program yang digunakan untuk pra olah data.

```
nama_file_latih_input = "random_inputs_300_5.npy"
nama_file_latih_label = "random_labels_300_4.npy"
inp_row, inp_col = 1, 4
number_of_hidden_neurons = 4
epochs = 2000
act = "sigmoid"
```

Gambar 4.7 Potongan Kode Program 2

Potongan kode program pada Gambar 4.7 berfungsi untuk memanggil *dataset inputs* dan *dataset labels* yang akan dijadikan data latih pada model yang dibuat. Selain itu, berfungsi untuk menentukan nilai pada kolom berapa saja yang akan diproses, menentukan jumlah hidden neuron, menentukan epochs, dan fungsi aktivasi yang digunakan.

```
def sigmoid(a):
    hasil = 1 / (1 + np.exp(-a))
    return hasil
def ReLU(a):
    hasil = a
    e, f = a.shape
    for i in range(0, e):
        for j in range(0, f):
            if a[i,j] < 0 : hasil[i,j] = 0
            if a[i,j] >= 0: hasil[i,j] = a[i,j]
    return hasil
```

Gambar 4.8 Potongan Kode Program 3

Potongan kode program pada Gambar 4.8 berfungsi untuk menghitung dengan rumus fungsi sigmoid dan fungsi ReLU.

```
inputs = np.load(nama_file_latih_input)[:, 1:5]
inputs_nomor_gambar_asal = np.load(nama_file_latih_input)[:, 0]
labels = np.load(nama_file_latih_label)[:, 1:4]

m, n = inputs.shape; print("Data latih inputs:", m,",", n)
o, p = labels.shape; print("Data latih label:", o, "'", p)
i_n = n
o_n = p
h_n = number_of_hidden_neurons
print('Hidden neuron:', h_n)

w_i_h = np.random.uniform(-0.5, 0.5, (h_n, i_n))
w_h_o = np.random.uniform(-0.5, 0.5, (o_n, h_n))
b_i_h = np.zeros((h_n, 1))
b_h_o = np.zeros((o_n, 1))
```

Gambar 4.9 Potongan Kode Program 4

Potongan kode program pada Gambar 4.9 berfungsi untuk menghitung dengan rumus fungsi sigmoid dan fungsi ReLU.

```
#ANN TRAINING
learn_rate = 0.001
nr_correct = 0
for epoch in range(1, epochs+1):
    for inp, label in zip(inputs, labels):
        inp.shape += (1,)
        label.shape += (1,)
        # Forward propagation input -> hidden
        h_pre = b_i_h + w_i_h @ inp
        if act == "ReLU": h = ReLU(h_pre)
        if act == "sigmoid": h = sigmoid(h_pre)
        # Forward propagation hidden -> output
        o_pre = b_h_o + w_h_o @ h
        if act == "ReLU": o = ReLU(o_pre)
        if act == "sigmoid": o = sigmoid(o_pre)
        # Cost / Error calculation
        e = 1 / len(o) * np.sum((o - label) ** 2, axis=0)
        nr_correct += int(np.argmax(o) == np.argmax(label))
        # Backpropagation output -> hidden (cost function derivative)
        delta_o = o - label
        w_h_o += -learn_rate * delta_o @ np.transpose(h)
        b_h_o += -learn_rate * delta_o
```

Gambar 4.10 Potongan Kode Program 5

Potongan kode program pada Gambar 4.10 berfungsi untuk proses *training* model *Artificial Neural Network*.

```
    # Backpropagation hidden -> input (activation function derivative)
    delta_h = np.transpose(w_h_o) @ delta_o * (h * (1 - h))
    w_i_h += -learn_rate * delta_h @ np.transpose(inp)
    b_i_h += -learn_rate * delta_h
# Show accuracy for this epoch
print(f"epoch = {epoch}, " + f"Accuracy: {round((nr_correct / inputs.shape[0]) * 100, 2)}%.")
nr_correct = 0
```

Gambar 4.11 Potongan Kode Program 6

Potongan kode program pada Gambar 4.11 merupakan bagian dari proses *training* model yang berfungsi untuk menghitung tingkat akurasi dari model yang dibuat.

```
np.save("a3_b_i_h.npy", b_i_h)
np.save("a3_w_i_h.npy", w_i_h)
np.save("a3_b_h_o.npy", b_h_o)
np.save("a3_w_h_o.npy", w_h_o)
```

Gambar 4.12 Potongan Kode Program 7

Potongan kode program pada Gambar 4.12 berfungsi untuk menyimpan hasil *training* di komputer.

```
def konversi(array):
    posisi_dg_nilai_tertinggi = array.argmax()
    if posisi_dg_nilai_tertinggi == 0: label_hasil = "belum matang"
    if posisi_dg_nilai_tertinggi == 1: label_hasil = "matang"
    if posisi_dg_nilai_tertinggi == 2: label_hasil = "terlalu matang atau busuk"
    return label_hasil
```

Gambar 4.13 Potongan Kode Program 8

Potongan kode program pada Gambar 4.13 berfungsi untuk menguji model, serta menentukan label hasil identifikasi kematangan pisang kepok. Penentuan label hasil didapatkan apabila posisi nilai tertinggi pada indeks (0) atau kolom ke-1, maka pisang kepok dinyatakan belum matang, apabila posisi nilai tertinggi pada indeks (1) atau kolom ke-2, maka pisang kepok dinyatakan matang, dan apabila posisi nilai tertinggi pada indeks (2) atau kolom ke-3, maka pisang kepok dinyatakan terlalu matang atau busuk.

```
b_i_h = np.load("a3_b_i_h.npy")
w_i_h = np.load("a3_w_i_h.npy")
b_h_o = np.load("a3_b_h_o.npy")
w_h_o = np.load("a3_w_h_o.npy")
```

Gambar 4.14 Potongan Kode Program 9

Potongan kode program pada Gambar 4.14 berfungsi untuk membaca *file* hasil *training* yang telah tersimpan pada komputer.

```
while True:
    print("")
    input_file_1 = input("Masukkan nama file gambar:")
    output_file_1 = input_file_1 + " lp"
    input_file_2 = output_file_1

    #Preprocessing: Ganti latar gambar dengan piksel putih.
    latar_putih(input_file_1, output_file_1)

    dataset_outsample = create_dataset(input_file_2)
    inp = dataset_outsample[0, 1:5]
    print("Input:", inp)
    inp.shape += (1,)
```

Gambar 4.15 Potongan Kode Program 10

Potongan kode program pada Gambar 4.15 berfungsi untuk *input file* gambar yang akan diidentifikasi kematangannya (data uji), mengubah latar gambar menjadi latar putih, dan mengubah gambar latar putih menjadi dataset berupa *file* npy yang berisi nilai persentase piksel cenderung berwarna merah, nilai persentase piksel cenderung berwarna hijau, nilai persentase piksel cenderung berwarna biru, dan nilai persentase piksel gelap.

```
# Forward propagation input -> hidden
h_pre = b_i_h + w_i_h @ inp.reshape(inp_row*inp_col, 1)
if act == "ReLU": h = ReLU(h_pre)
if act == "sigmoid": h = sigmoid(h_pre)

# Forward propagation hidden -> output
o_pre = b_h_o + w_h_o @ h
if act == "ReLU": o = ReLU(o_pre)
if act == "sigmoid": o = sigmoid(o_pre)
o = sigmoid(o_pre)
output = o
print("Output:", output)
```

Gambar 4.16 Potongan Kode Program 11

Potongan kode program pada Gambar 4.16 berfungsi untuk menghitung hasil *output* berupa nilai dengan 3 indeks yang memiliki rentang nilai 0 sampai 1.

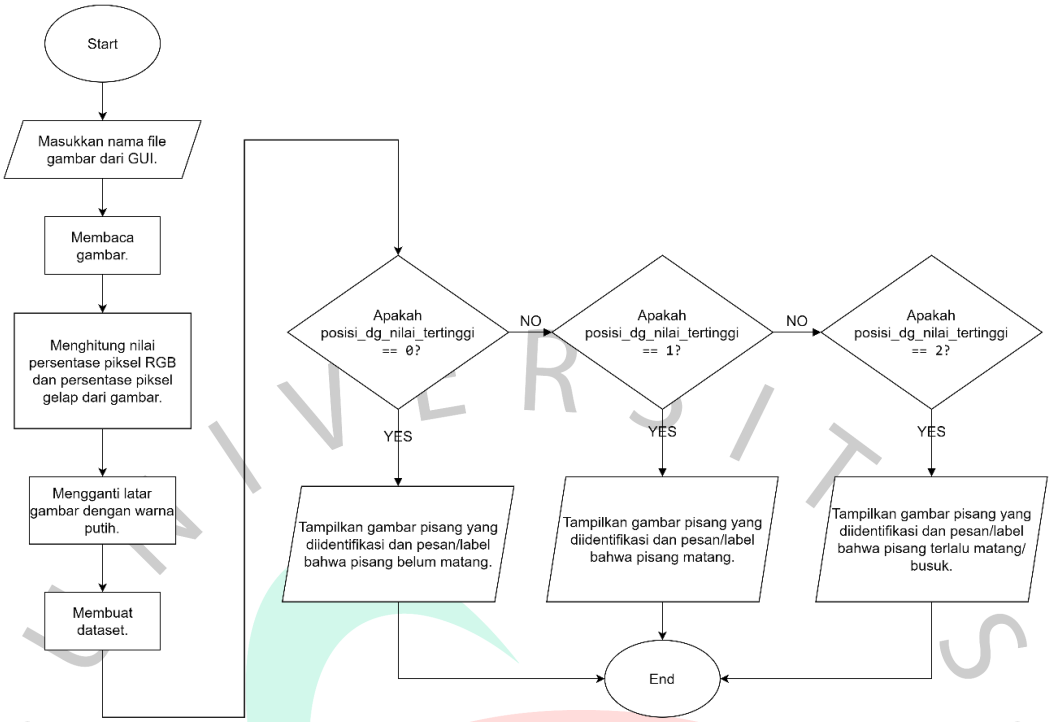
```
label_hasil = konversi(output)
print("Label hasil:", label_hasil)
gambar = plt.imread(input_file_1 + ".jpg")
plt.figure()
plt.imshow(gambar)
plt.title(f"Klasifikasi menurut model ANN: pisang " + label_hasil + ".")
plt.show()
```

Gambar 4.17 Potongan Kode Program 12

Potongan kode program pada Gambar 4.17 berfungsi untuk menentukan hasil *output* dari nilai dengan 3 indeks yang memiliki rentang nilai 0 sampai 1. Jika nilai hasil *output* pada indeks 0 lebih tinggi, maka pisang kepok dinyatakan belum matang. Jika nilai hasil *output* pada indeks 1 lebih tinggi, maka pisang kepok dinyatakan matang. Jika nilai hasil *output* pada indeks 2 lebih tinggi, maka pisang kepok dinyatakan terlalu matang atau busuk.

4.3.6 Alur Komputasi

Diagram alir digunakan untuk menggambarkan proses algoritma *Artificial Neural Network* dari mengolah gambar sampai mengidentifikasi tingkat kematangan pisang kepok. Berikut di bawah ini diagram alir aplikasi identifikasi tingkat kematangan pisang kepok.

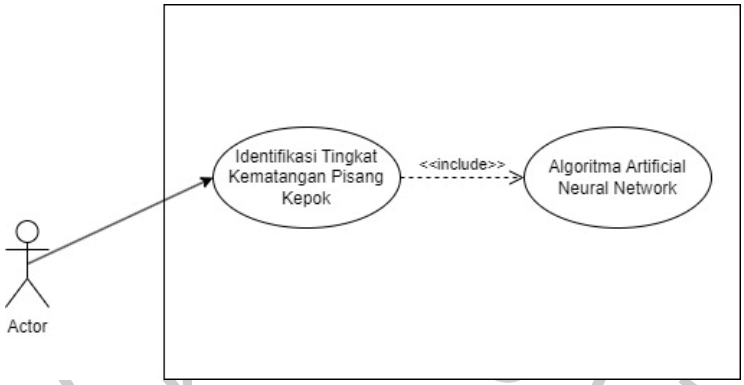


Gambar 4.18 Alur Komputasi Aplikasi Identifikasi Tingkat Kematangan Pisang Kepok

Gambar 4.18 memperlihatkan diagram alir algoritma aplikasi identifikasi tingkat kematangan pisang kepok. Berikut penjelasan dari diagram alir tersebut.

1. Pengguna memasukkan nama *file* gambar yang ingin diidentifikasi.
2. Sistem membaca gambar.
3. Sistem menghitung persentase nilai piksel RGB dan persentase nilai piksel gelap dari gambar.
4. Sistem mengganti latar gambar dengan warna putih.
5. Sistem membuat dataset yang berisi persentase piksel merah, persentase piksel hijau, persentase piksel biru, dan persentase piksel gelap.
6. Jika posisi dengan nilai tertinggi = 0, maka sistem menampilkan gambar dan pesan/label bahwa pisang kepok belum matang. Jika posisi dengan nilai tertinggi = 1, maka sistem menampilkan gambar dan pesan/label bahwa pisang kepok matang. Jika posisi dengan nilai tertinggi = 2, maka sistem menampilkan gambar dan pesan/label bahwa pisang kepok terlalu matang atau busuk.

4.3.7 *Use Case Diagram*



Gambar 4.19 *Use Case* Diagram Aplikasi Identifikasi Tingkat Kematangan Pisang Kepok

Berdasarkan Gambar 4.19, *actor* atau pengguna setelah membuka aplikasi identifikasi kematangan pisang kepok dapat melihat menu *start* dan menu keluar secara langsung. Jika pengguna memilih menu *start*, maka akan tampil halaman untuk memasukkan direktori gambar pisang kepok yang ingin diidentifikasi tingkat kematangannya. Selanjutnya, pengguna dapat memilih tombol proses untuk melihat hasil identifikasi tingkat kematangan pisang kepok. Ketika pengguna memilih menu keluar, aplikasi akan tertutup.

4.3.8 Skenario *Use Case*

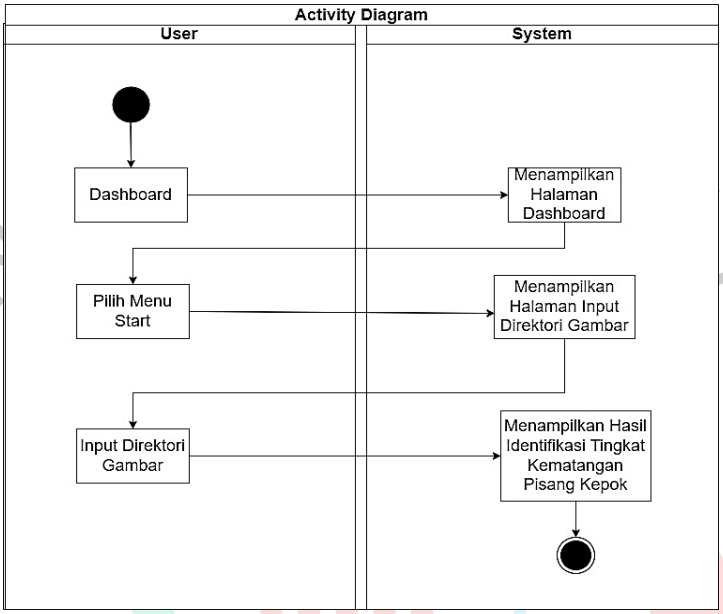
Skenario *use case* merupakan penjelasan secara rinci dari *use case* diagram. Berikut beberapa scenario *use case* yang digunakan pada aplikasi identifikasi tingkat kematangan pisang kepok.

Tabel 4.4 Skenario *Use Case* Identifikasi Tingkat Kematangan Pisang Kepok

ID	**UC1**
Nama *use case*	Identifikasi Tingkat Kematangan Pisang Kepok
Aktor	*User*
Deskripsi	Mengidentifikasi tingkat kematangan pisang kepok dengan *Artificial Neural Network*.
Pre-condition	Memilih menu *start* pada *dashboard*.
Langkah-langkah	1. *User* masuk ke halaman *dashboard*. 2. *User* memilih menu *start*. 3. *User* melakukan *input* yang berisi direktori *file* gambar yang ingin diidentifikasi. 4. *User* memilih tombol proses. 5. Gambar diproses oleh sistem dengan algoritma *Artificial Neural Network*.
Trigger	Ketika ingin memasukkan gambar yang ingin diidentifikasi dan proses identifikasi.
Post-Condition	Gambar diproses dan ditampilkan hasil identifikasi tingkat kematangannya.

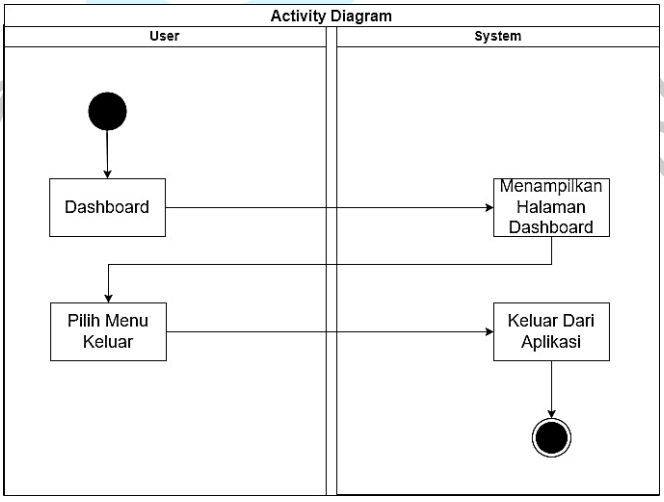
4.3.9 *Activity Diagram*

Activity diagram merupakan gambaran alur aktivitas dari sebuah sistem yang dirancang (Kurniawan & Syarifuddin, 2020). *Activity diagram* ini berisikan runtutan proses dari aplikasi yang dibuat oleh peneliti. Gambar di bawah ini adalah *activity diagram* aplikasi identifikasi tingkat kematangan pisang kepok.



Gambar 4.20 *Activity Diagram* Identifikasi Tingkat Kematangan Pisang Kepok

Gambar 4.20 adalah *activity diagram* identifikasi tingkat kematangan pisang kepok untuk memulai tahap identifikasi tingkat kematangan pisang kepok hingga langkah-langkah pengguna melakukan *input* direktori gambar dan proses sistem mengidentifikasi gambar hingga menampilkan hasil identifikasinya.

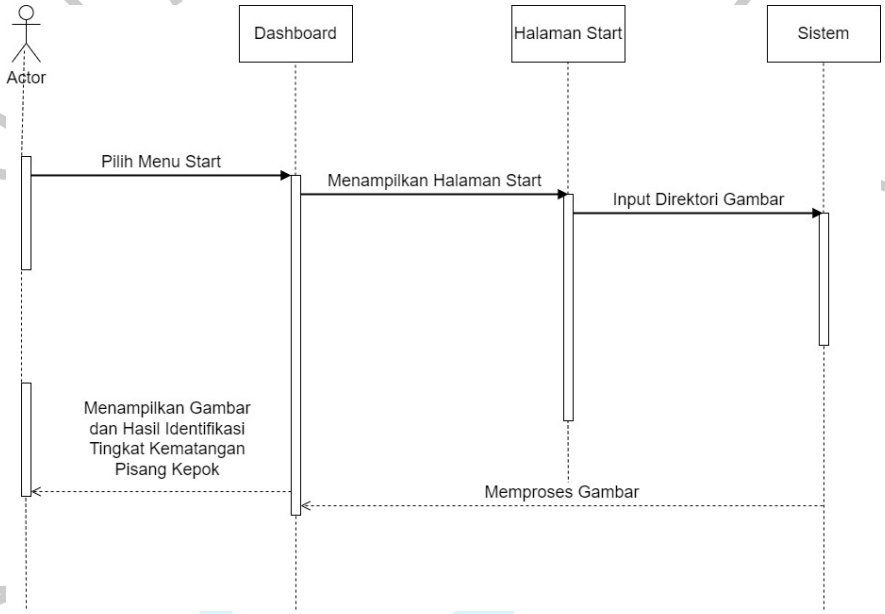


Gambar 4.21 *Activity Diagram* Keluar

Gambar 4.21 merupakan *activity diagram* keluar, yaitu langkah-langkah pengguna ketika ingin keluar dari aplikasi.

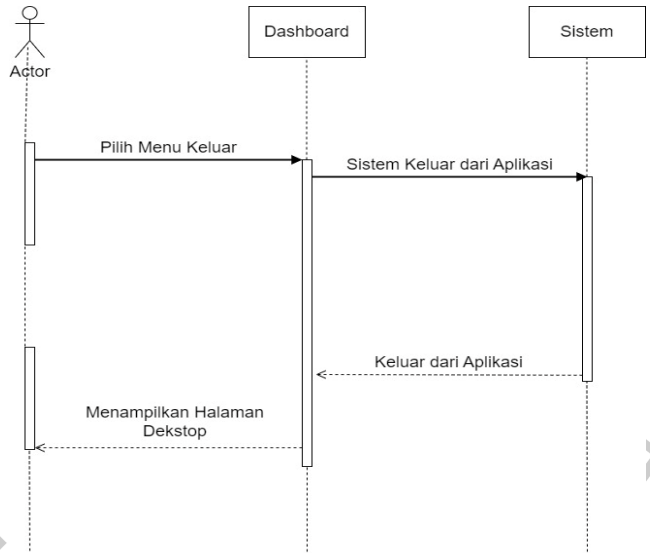
4.3.10 *Sequence Diagram*

Sequence diagram berisi penjelasan dari objek-objek yang berhubungan dengan proses berjalannya sebuah sistem. Sequence diagram digunakan untuk mengetahui urutan proses penggunaan dari aplikasi sehingga mendapatkan *output* yang diinginkan. Berikut di bawah ini gambaran *sequence diagram* dari aplikasi identifikasi tingkat kematangan pisang kepok.



Gambar 4.22 *Sequence Diagram* Identifikasi Tingkat Kematangan Pisang Kepok

Gambar 4.22 adalah *sequence diagram* identifikasi tingkat kematangan pisang kepok yang menjelaskan proses dari pilih menu *start* hingga memproses gambar pisang kepok yang ingin diidentifikasi serta menampilkan hasil identifikasi tingkat kematangan pisang kepok.



Gambar 4.23 *Sequence Diagram* Keluar

Gambar 4.23 merupakan *sequence diagram* keluar yang menjelaskan proses keluar dari aplikasi.

4.3.11 Perancangan Antar Muka

Perancangan antar muka merupakan gambaran desain aplikasi yang dibuat oleh peneliti dengan editor Figma. Berikut beberapa desain antar muka dari aplikasi identifikasi tingkat kematangan pisang kepok.

Gambar 4.24 Rancangan Tampilan *Dashboard*

Gambar 4.24 merupakan rancangan tampilan *dashboard* untuk pengguna. Pengguna dapat memilih tombol *start* jika ingin memulai tahap identifikasi tingkat kematangan pisang kepok. Jika pengguna ingin menutup aplikasi dapat memilih tombol keluar.

Gambar 4.25 Rancangan Tampilan Proses Identifikasi Tingkat Kematangan Pisang Kepok

Gambar 4.25 merupakan rancangan tampilan proses identifikasi tingkat kematangan pisang kepok untuk pengguna. Pengguna dapat mengetik *file* direktori dari gambar yang ingin diidentifikasi. Setelah itu, pengguna dapat menekan tombol proses agar aplikasi dapat melakukan proses identifikasi tingkat kematangan pisang kepok.

Gambar 4.26 Rancangan Tampilan Hasil Identifikasi Tingkat Kematangan Pisang Kepok

Gambar 4.26 merupakan rancangan tampilan hasil identifikasi tingkat kematangan pisang kepok.

4.3.12 Perancangan Pengujian

Pada perancangan aplikasi harus dilakukan tahap pengujian agar aplikasi yang dibuat dapat dipastikan hasilnya sesuai atau tidak dengan rancangan yang sudah dilakukan sebelumnya. Pada pengujian ini, peneliti menggunakan pengujian *Black Box* dan *White Box*.

4.3.12.1 Perancangan Pengujian *Black Box*

Pengujian *black box* dilakukan agar penguji mengetahui *output* yang diperoleh sudah sesuai dengan *input* (Ningrum, Suherman, Aryanti, Prasetya, & Saefudin, 2019). Berikut tabel hasil pengujian *black box*.

Tabel 4.5 Rancangan Pengujian *Black Box*

No	Fitur Aplikasi	*Input*	Hasil yang Diharapkan
1	Menu *Dashboard*	Pengguna membuka aplikasi.	Menampilkan halaman *dashboard.*
2	Menu *Start*	Pengguna memilih menu *start*.	Menampilkan halaman *start* yang berisi *input* direktori gambar.
3	Menu Proses	Pengguna memilih menu proses.	Gambar yang dimasukkan diproses dan menampilkan gambar serta hasil identifikasi tingkat kematangan pisang kepok.
4	Menu Keluar	Pengguna memilih menu keluar.	Aplikasi tertutup.

4.3.12.2 Perancangan Pengujian *White Box*

Pengujian *white box* dilakukan untuk menganalisa aplikasi hingga kode program dari aplikasi (Sie, Musdar, & Bahri, 2022). Pengujian *white box* dilakukan terhadap algoritma *Artificial Neural Network* untuk memastikan kode program berjalan dengan baik dalam mengidentifikasi tingkat kematangan pisang kepok melalui gambar.