BAB III

METODOLOGI PENELITIAN

3.1 Waktu Penelitian

Waktu penelitian ini dimulai pada bulan Mei sampai dengan bulan Juli 2023.

3.2 Tempat penelitian

Tempat penelitian ini berlokasi di Institut Informatika dan Bisnis (IIB) Darmajaya, Jl. ZA. Pagar Alam No.93, Gedong Meneng, Kec. Rajabasa, Kota Bandar Lampung, Lampung, 35141.

3.3 Alat Penelitian

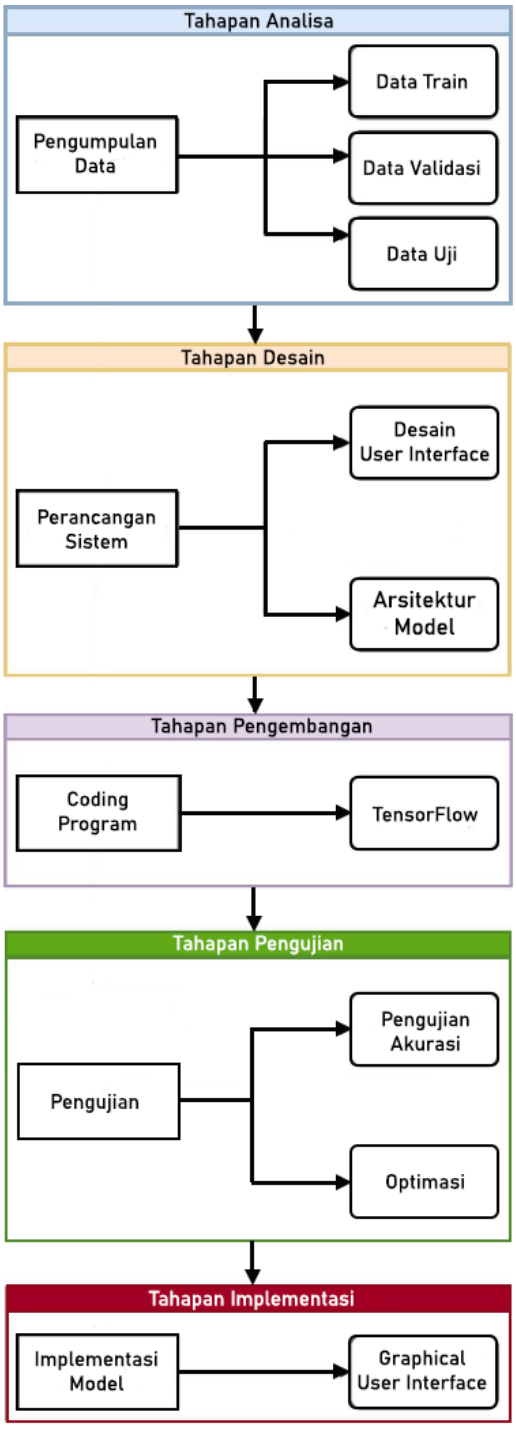
Perangkat lunak dan perangkat keras yang digunakan dalam penelitian ini untuk mengumpulkan, mengolah, menganalisa, dan menyajikan data yaitu:

1. Perangkat Lunak *(Software)*:
   a. Sistem Operasi: Windows 11 Home
   b. Integrated Development Environment (IDE): Jupyter Notebook, Anaconda
   c. Framework: TensorFlow

2. Perangkat Keras *(Hardware)*:
   a. Processor: 11th Gen Intel Core i5-11400H
   b. Graphics Processing Unit (GPU): Nvidia GeForce RTX 3050
   c. Random Access Memory (RAM): 8 GB
   d. Penyimpanan: 512 GB SSD

3.4 Metode Penelitian

Untuk menerapkan Metode *Convolutional Neural Network (CNN)*, terdapat beberapa tahapan yang akan dilakukan (Arsal et al., 2020):



Gambar 3.1 Tahapan penerapan Metode *Convolutional Neural Network (CNN)*

Berikut adalah langkah-langkah umum untuk menerapkan *Convolutional Neural Network (CNN)* dalam pengembangan model *deep learning*:

1. Tahapan Analisa. Langkah pertama adalah mempersiapkan data gambar yang akan digunakan untuk pelatihan model CNN. Ini melibatkan pengumpulan, pengolahan, dan pemisahan data menjadi set pelatihan (data train), set validasi (data validasi), dan set pengujian (data uji). Data juga perlu diaugmentasi, seperti dengan rotasi ataupun menambahkan kecerahan gambar untuk meningkatkan keragaman data dan meningkatkan kemampuan generalisasi pada model.
2. Tahapan Desain. Setelah persiapan data, langkah berikutnya adalah perancangan desain *User Interface (UI)* yang akan digunakan pada *Graphical User Interface (GUI)* dan dengan merancang arsitektur model.
    a. Desain *User Interface* melibatkan perancangan desain antarmuka pengguna berupa tampilan halaman, tombol, teks, dan fitur lainnya yang akan diterapkan pada *Grapichal User Interface (GUI)*.
    b. Perancangan arsitektur model melibatkan pemilihan jumlah lapisan, tipe lapisan (konvolusi, pooling, dll.), ukuran kernel, fungsi aktivasi, serta pengaturan parameter seperti jumlah filter, ukuran batch, dan lain-lain.
3. Tahapan Pengembangan. Pada tahap ini, pembuatan kode dilakukan menggunakan Bahasa Python. Pada tahap ini pengembangan model dilakukan menggunakan *framework* TensorFlow yang digunakan untuk membangun model dan melatih model. Arsitektur model yang telah didesain diterapkan pada kode. Setelah arsitektur model didefinisikan, langkah berikutnya adalah melatih model dengan menggunakan set pelatihan. Proses pelatihan melibatkan pemberian data gambar pada model, penghitungan *loss*, optimisasi parameter menggunakan *optimizer*, dan pembaruan bobot model. Pelatihan berlangsung selama beberapa *epoch* untuk meningkatkan performa model secara bertahap. Setelah pelatihan, langkah selanjutnya adalah melakukan validasi model menggunakan set validasi yang sebelumnya telah dipersiapkan. Validasi dilakukan untuk mengukur kinerja model secara objektif, seperti akurasi. Jika model tidak memenuhi kriteria performa yang diharapkan, maka arsitektur

model atau parameter dapat disesuaikan dan proses pelatihan dapat diulang. Setelah validasi, langkah berikutnya adalah menguji model pada set pengujian yang telah dipersiapkan sebelumnya.

4. Tahapan pengujian. Setelah model dinyatakan memuaskan, langkah terakhir adalah melakukan pengujian pada data yang belum pernah dilihat sebelumnya untuk menguji kemampuan model dalam memprediksi. Setelah itu, jika model masih belum mencapai performa yang diharapkan, dapat dilakukan optimisasi pada arsitektur model. Jika model telah terbukti efektif, maka model dapat digunakan pada sistem keamanan.
5. Tahapan Implementasi. Pada tahap ini, model yang telah dikembangkan dan dioptimasi diterapkan pada sistem keamanan menggunakan *Graphical User Interface (GUI)*.

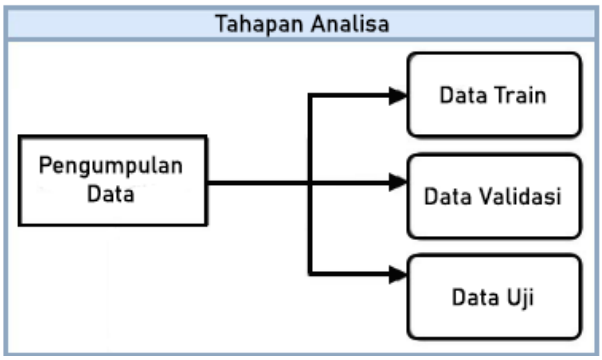
3.5 Jadwal Penelitian

Waktu penelitian ini akan dilakukan selama 3 bulan, yaitu bulan Mei sampai dengan bulan Juli 2023 dengan detail aktivitas yang dapat dilihat pada tabel di bawah ini:

No.	Kegiatan	Mei				Juni				Juli			
		1	2	3	4	1	2	3	4	1	2	3	4
1	Tahapan Analisa	■	■										
2	Tahapan Desain		■	■									
3	Tahapan Pengembangan				■	■	■	■	■				
4	Tahapan Pengujian							■	■	■	■	■	
5	Tahapan Evaluasi											■	■

Tabel 3.1 Jadwal Penelitian

3.6 Tahapan Analisa



Gambar 3.2 Tahapan Analisa

Pada Tahapan Analisa, data yang digunakan berupa dataset *Labeled Faces in the Wild (LFW)* yang dipublikasikan oleh University of Massachusetts Amherst yang merupakan salah satu *research university* terletak di Amherst, Massachusetts, Amerika Serikat. LFW dapat diakses melalui: http://vis-www.cs.umass.edu/lfw/.

Gambar 3.3 Logo University of Massachusetts

Labeled Faces in the Wild (LFW) merupakan database foto wajah yang dirancang untuk mempelajari masalah pengenalan wajah. Kumpulan data ini berisi lebih dari 13.000 gambar wajah yang dikumpulkan dari web. Setiap wajah telah diberi label sesuai dengan nama orang yang digambarkan. Pada dataset ini terdapat 1680 orang dalam foto memiliki dua atau lebih foto yang berbeda dalam dataset.

3.6.1 Pemilihan Sampel Gambar dari Dataset

Untuk teknik pemilihan sampel gambar pada dataset, label dipilih berdasarkan banyaknya jumlah gambar pada label. *Convolutional Neural Network (CNN)* membutuhkan banyak gambar pada pelatihan. Karena itu, pemilihan label diambil berdasarkan jumlah gambar pada label yang memiliki lebih dari 70 gambar. Pada penelitian ini, saya juga menambahkan beberapa gambar sebanyak 524 gambar dari beberapa orang berbeda yang nantinya akan digunakan untuk pelatihan dan pengujian model. Proses pemilihan ini menghasilkan 11 label terpilih dengan total 1726 sampel gambar dengan gambar yang sudah diubah menjadi *grayscale* dan ukurannya yang sudah diubah yaitu 50x50 piksel.

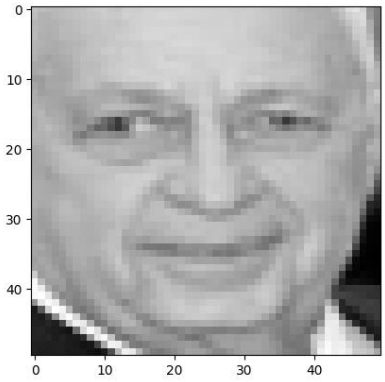
```
[##########] (77 samples)        label : Ariel_Sharon
[##########] (236 samples)       label : Colin_Powell
[##########] (121 samples)       label : Donald_Rumsfeld
[##########] (530 samples)       label : George_W_Bush
[##########] (109 samples)       label : Gerhard_Schroeder
[##########] (71 samples)        label : Hugo_Chavez
[##########] (280 samples)       label : Husain_Ahmad_Faiq
[##########] (80 samples)        label : Muhammad_Saifuddin_Mahfudz
[##########] (150 samples)       label : Najib_Abdullah_Muqsith
[##########] (97 samples)        label : Pak_Hary_Sabita
[##########] (144 samples)       label : Tony_Blair

                number of samples : 1804
```

Gambar 3.4 Sampel yang akan digunakan untuk pelatihan model

3.6.2 Augmentasi dan Balancing Data

Augmentasi data merupakan proses untuk meningkatkan jumlah dan variasi data yang ada dalam satu set data dengan cara membuat modifikasi pada data yang ada. Tujuan utama dari augmentasi data adalah untuk memperluas variasi data yang tersedia, meningkatkan kualitas model, dan mengurangi *overfitting*.

Pada proses ini, 1 sampel gambar diaugmentasikan sehingga setiap 1 gambar pada label menghasilkan 50 sampel gambar baru dengan variasi berbeda. Contoh di bawah ini merupakan hasil augmentasi data dari images[0].



Gambar 3.5 Contoh gambar yang digunakan untuk augmentasi data

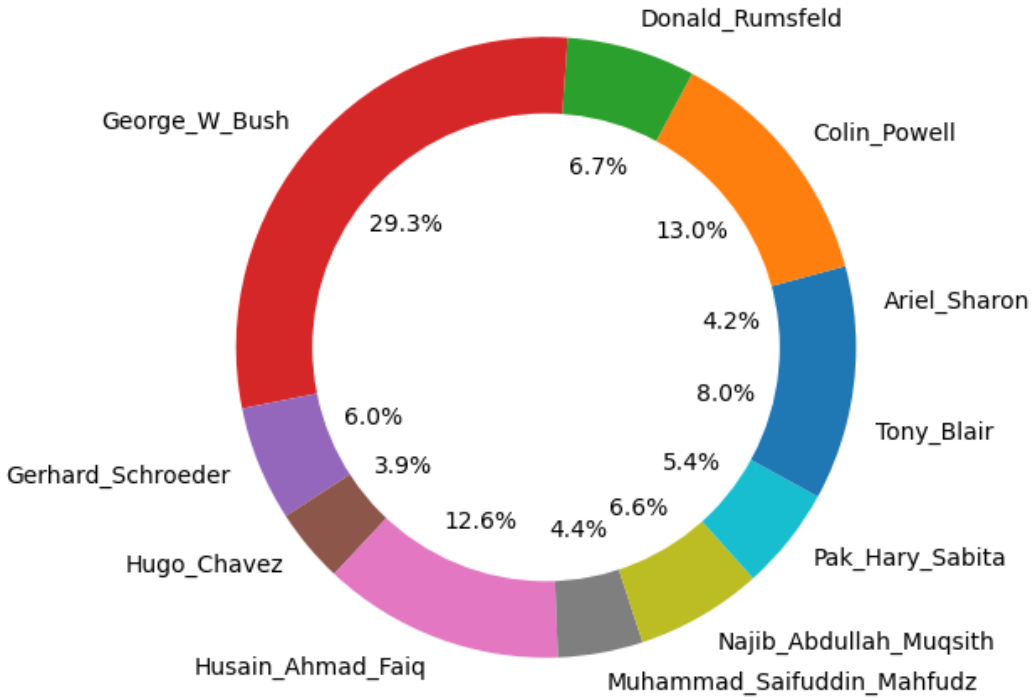
				
Rotasi 3 derajat	Rotasi -3 derajat	Rotasi 5 derajat	Rotasi -5 derajat	Rotasi 7 derajat
				
Rotasi -7 derajat	Rotasi 10 derajat	Rotasi -10 derajat	Rotasi 12 derajat	Rotasi -12 derajat

				
Rotasi 15 derajat	Rotasi -15 derajat	Rotasi 20 derajat	Rotasi -20 derajat	Translasi 2 piksel
				
Translasi -2 piksel	Translasi 3 piksel	Translasi -3 piksel	Translasi 4 piksel	Translasi -4 piksel
				
Translasi 6 piksel	Translasi -6 piksel	Translasi 8 piksel	Translasi -8 piksel	Translasi 10 piksel
				
Translasi -10 piksel	Translasi vertikal 2 piksel	Translasi vertikal -2 piksel	Translasi vertikal 3 piksel	Translasi vertikal -3 piksel

				
Translasi vertikal 4 piksel	Translasi vertikal -4 piksel	Translasi vertikal 6 piksel	Translasi vertikal -6 piksel	Translasi vertikal 8 piksel
				
Translasi vertikal -8 piksel	Translasi vertikal 10 piksel	Translasi vertikal -10 piksel	Kecerahan 10 piksel	Kecerahan -10 piksel
				
Kecerahan 15 piksel	Kecerahan -15 piksel	Kecerahan 20 piksel	Kecerahan -20 piksel	Kecerahan 30 piksel
				
Kecerahan -30 piksel	Kecerahan 40 piksel	Kecerahan -40 piksel	Kecerahan 45 piksel	Kecerahan -45 piksel

Tabel 3.2 Hasil augmentasi data dari images[0]

Pada proses augmentasi data ini, 1726 sampel gambar menghasilkan 86300 sampel gambar baru, sehingga total keseluruhan gambar yang akan digunakan pada proses pelatihan model sebanyak 88026 sampel gambar.



92004 →

```
('Ariel_Sharon', 3876)
('Colin_Powell', 11934)
('Donald_Rumsfeld', 6171)
('George_W_Bush', 26928)
('Gerhard_Schroeder', 5508)
('Hugo_Chavez', 3570)
('Husain_Ahmad_Faiq', 11577)
('Muhammad_Saifuddin_Mahfudz', 4080)
('Najib_Abdullah_Muqsith', 6069)
('Pak_Hary_Sabita', 4947)
('Tony_Blair', 7344)
```

Gambar 3.6 Persentase jumlah sampel gambar pada label

Setelah augmentasi data dilakukan, proses selanjutnya yaitu balancing data dengan menyamaratakan jumlah sampel gambar pada tiap label. Tujuannya untuk menghindari terjadinya *overfitting* yang menyebabkan masalah pada generalisasi saat mendeteksi wajah. Proses *balancing data* ini dilakukan dengan mengambil sebanyak 3500 sampel gambar pada tiap label, hingga diperoleh total keseluruhan sampel gambar dari 11 label yaitu sebanyak 38500 sampel gambar.

3.6.3 Label Encoding

Label encoding adalah sebuah proses untuk mengubah data kategorikal yang bersifat *string* menjadi numerik agar lebih mudah dimengerti oleh sistem. Pada penelitian ini, *label encoding* menggunakan teknik *One-Hot Encoding*. *One-Hot encoding* adalah sebuah proses yang bertujuan untuk mengubah data kategorikal *integer* menjadi *boolean*, di mana setiap data kategori unik akan dikembangkan menjadi parameter baru. Ada 11 label yang akan digunakan pada proses ini.

```
number of class : 11
['Ariel_Sharon' 'Colin_Powell' 'Donald_Rumsfeld' 'George_W_Bush'
 'Gerhard_Schroeder' 'Hugo_Chavez' 'Husain_Ahmad_Faiq'
 'Muhammad_Saifuddin_Mahfudz' 'Najib_Abdullah_Muqsith' 'Pak_Hary_Sabita'
 'Tony_Blair']
```

Kemudian, label tersebut diubah menjadi *array*.

```
[ 0  0  0 ... 10 10 10]
```

Setelah itu, label diubah menjadi *one-hot encoding* seperti berikut.

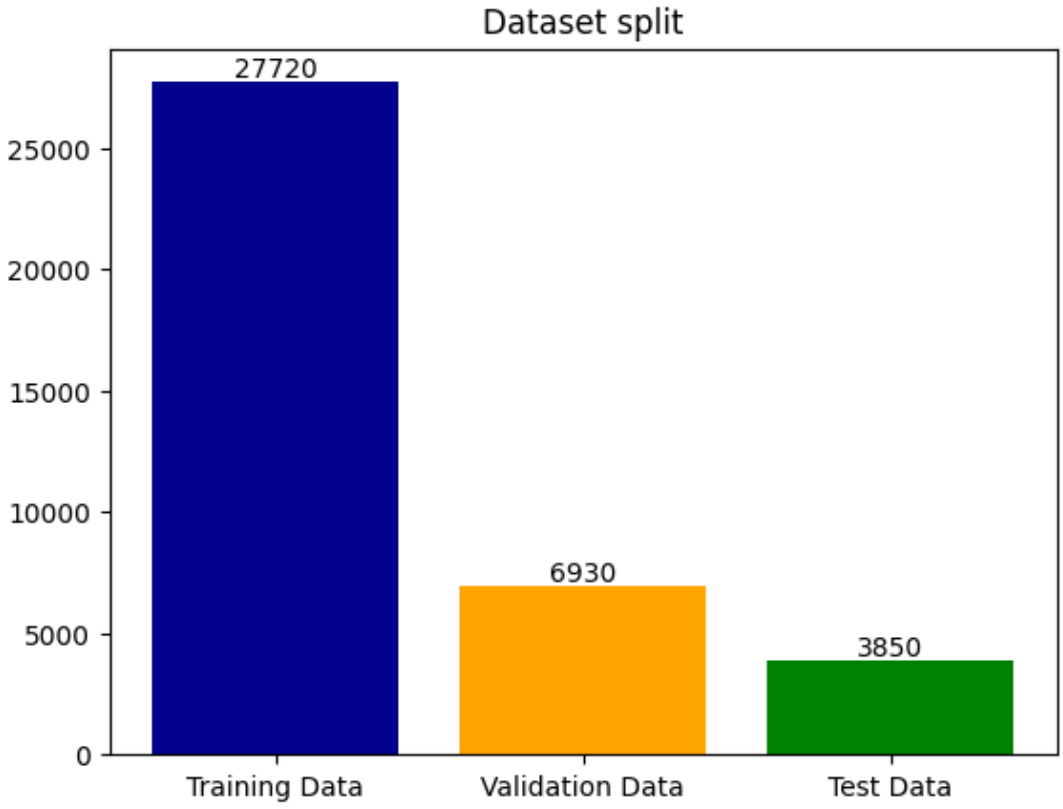
```
[[1. 0. 0. ... 0. 0. 0.]
 [1. 0. 0. ... 0. 0. 0.]
 [1. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 1.]
 [0. 0. 0. ... 0. 0. 1.]
 [0. 0. 0. ... 0. 0. 1.]]
```

Gambar 3.7 *One-Hot Encoding*

3.6.4 Pembagian Dataset

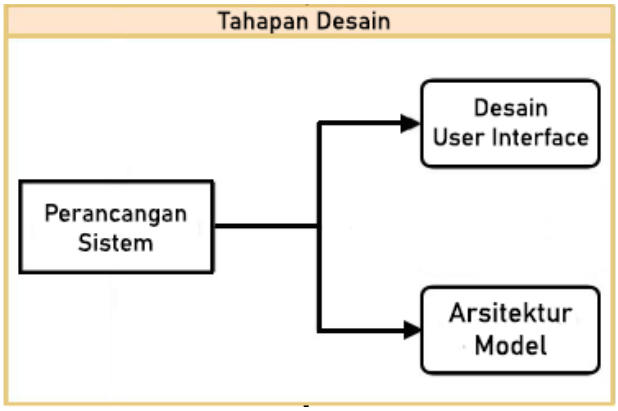
Pembagian dataset ini dilakukan dengan membagi dataset menjadi 3 bagian, yaitu: *Data train* (pelatihan), *data validation* (validasi), dan *data test* (pengujian). Ketiganya merupakan bagian dari proses pengembangan dan evaluasi model *machine learning*.

Pembagian ini penting untuk memastikan model tidak hanya berkinerja baik pada data yang digunakan untuk melatihnya *(data train)* tetapi juga mampu beradaptasi dengan data baru yang belum pernah dilihat sebelumnya *(data test)*. Pada penelitian ini, pembagian dataset dari 38500 sampel gambar yang telah diolah dilakukan dengan persentase data train dan validasi 90% (34650), dan data test: 10% (3850 gambar).



Gambar 3.8 Diagram pembagian dataset

3.7 Tahapan Desain



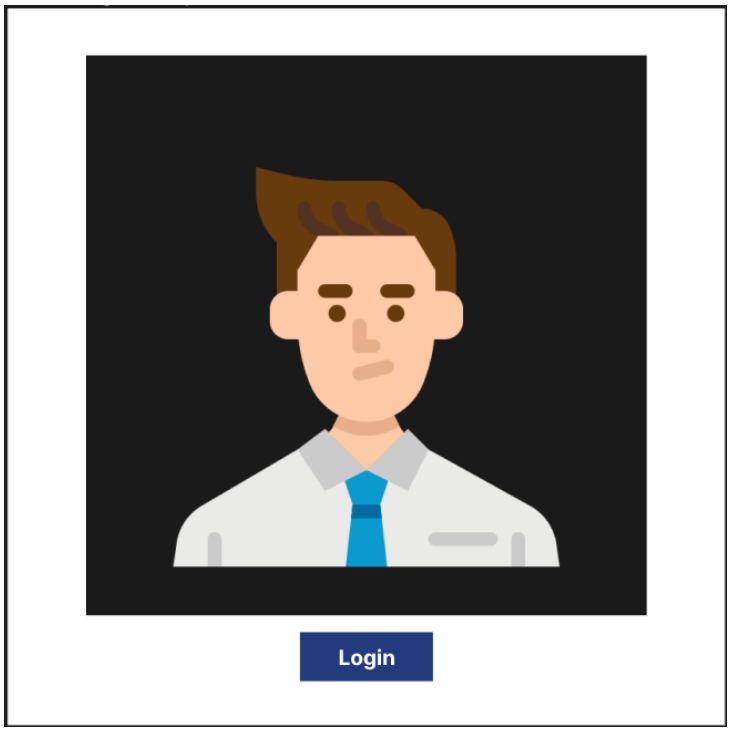
Gambar 3.9 Tahapan Desain

Tahapan Desain ini dilakukan untuk merancang sistem yang akan digunakan untuk membangun model. Tahap ini terdiri dari perancangan desain *User Interface* dan arsitektur model yang akan digunakan.

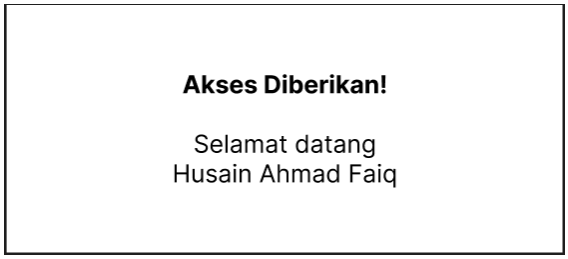
3.7.1 Desain User Interface

Pada *Graphical User Interface (GUI)* akan menggunakan 3 halaman, yaitu: Halaman Utama yang menjadi halaman awal *Graphical User Interface (GUI)*, Halaman Pengenalan Wajah yang digunakan untuk mendeteksi wajah, dan Halaman Tentang yang berisi tentang projek yang telah dikembangkan. Berikut adalah desain halaman yang akan diterapkan pada *Graphical User Interface (GUI)*:

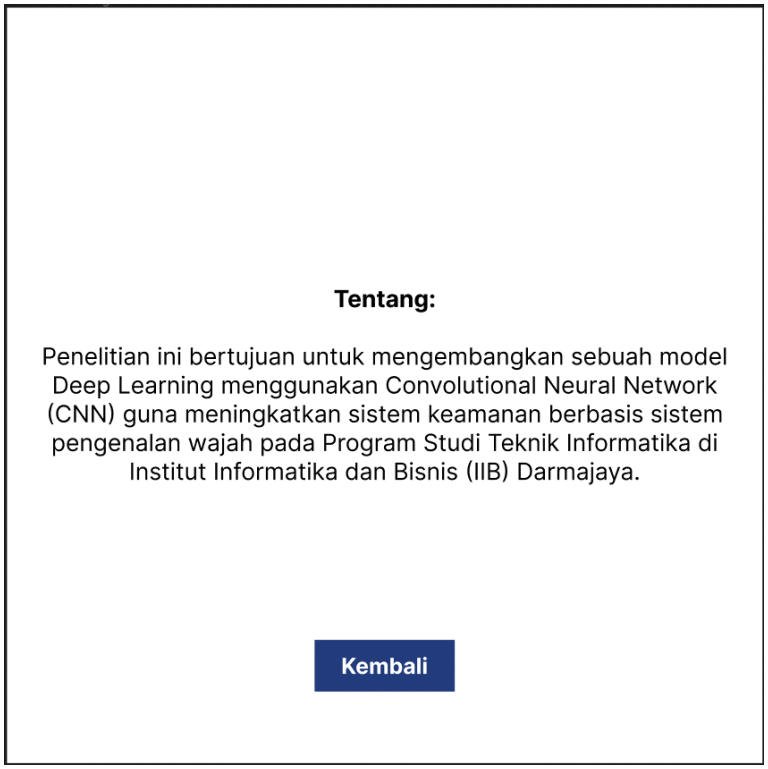
Gambar 3.10 Halaman Utama



Gambar 3.11 Halaman Pengidentifikasi Wajah

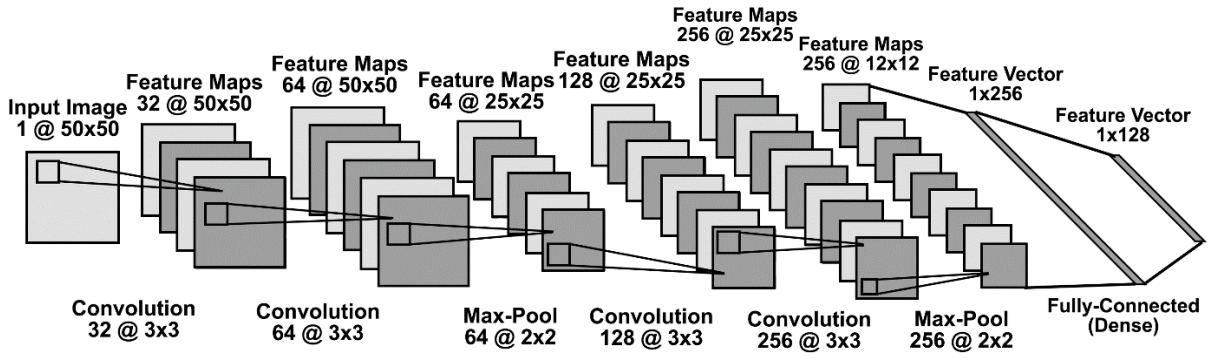


Gambar 3.12 *Alert Dialog*



Gambar 3.13 Halaman Tentang

3.7.2 Arsitektur Model



Gambar 3.14 Arsitektur Model

Pada model *Convolutional Neural Network (CNN)* ini, model menggunakan 8 lapisan *(layer)*, yaitu:

a. *Convolution layer 1*. Lapisan konvolusi ini mengekstrak gambar input dengan ukuran 50x50 piksel menggunakan 32 kernel dengan ukuran 3x3 piksel.
b. *Convolution layer 2*. Lapisan konvolusi ini mengekstrak gambar input dengan ukuran 50x50 piksel menggunakan 64 kernel dengan ukuran 3x3 piksel.
c. *Max-Pooling 1*. Lapisan ini melakukan *max-pooling* dengan mengambil nilai maksimum dari *feature maps* yang telah diekstrak oleh lapisan konvolusi. Tujuannya adalah untuk mengurangi kompleksitas komputasi dan jumlah parameter dalam model, sambil tetap mempertahankan fitur-fitur yang paling dominan. Proses *max-pooling* ini menggunakan kernel dengan ukuran 2x2 piksel hingga menghasilkan 64 *feature maps* dengan ukuran 25x25 piksel.
d. *Convolution layer 3*. Lapisan konvolusi ini mengekstrak *feature maps* yang telah dilakukan *max-pooling* sebelumnya dengan ukuran 25x25 piksel menggunakan 128 kernel dengan ukuran 3x3 piksel.
e. *Convolution layer 4*. Lapisan konvolusi ini juga mengekstrak *feature maps* yang telah dilakukan *max-pooling* sebelumnya dengan ukuran 25x25 piksel menggunakan 256 kernel dengan ukuran 3x3 piksel.

f. *Max-Pooling 2*. Lapisan ini melakukan *max-pooling* dengan mengambil nilai maksimum dari *feature maps* yang telah diekstrak oleh lapisan konvolusi. Proses *max-pooling* ini menggunakan kernel dengan ukuran 2x2 piksel hingga menghasilkan 256 *feature maps* dengan ukuran 12x12 piksel.
g. *Fully-Connected (Dense) Layer 1*. Lapisan ini digunakan untuk menggabungkan fitur-fitur yang telah diekstrak dari *layer* sebelumnya dan menghubungkan setiap fitur tersebut dengan setiap unit di *dense layer*. Ini memungkinkan model untuk memahami kombinasi kompleks dari fitur-fitur tersebut dan membantu dalam mempelajari pola-pola yang lebih abstrak dalam data. Lapisan ini menggunakan kernel dengan ukuran 1x 256 piksel.
h. *Fully-Connected (Dense) Layer 2*. Sama seperti lapisan *dense* sebelumnya, lapisan ini digunakan untuk membantu model dalam mempelajari pola-pola yang abstrak dalam data. Lapisan ini menggunakan kernel dengan ukuran 1x 128 piksel.

Berikut adalah kode Python yang digunakan untuk membangun model.

```
def cnn_model(input_shape):
    model = Sequential()
    model.add(Conv2D(32,
                     (3, 3),
                     name="Conv_1",
                     padding="same",
                     activation="relu",
                     input_shape=input_shape))
     model.add(Conv2D(64,
                     (3, 3),
                     name="Conv_2",
                     padding="same",
                     activation="relu",
```

```
                    input_shape=input_shape))
    model.add(MaxPool2D(pool_size=(2, 2)))
    model.add(Conv2D(128,
                     (3, 3),
                     name="Conv_3",
                     padding="same",
                     activation="relu"))
    model.add(Conv2D(256,
                     (3, 3),
                     name="Conv_4",
                     padding="same",
                     activation="relu"))
    model.add(MaxPool2D(pool_size=(2, 2)))
    model.add(Flatten())
    model.add(Dense(256,
                    name="Dense_1",
                    activation="relu"))
    model.add(Dense(32,
                    name="Dense_2",
                    activation="relu"))
    model.add(Dense(len(labels)))
    model.add(Activation("softmax"))
    model.summary()
    optimizer = keras.optimizers.Adam()
    loss = keras.losses.CategoricalCrossentropy()
```

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

Jumlah parameter dalam lapisan adalah jumlah elemen yang "dapat dipelajari" oleh model *machine learning* pada lapisan itu. Dari arsitektur model yang telah dirancang pada tahapan sebelumnya, berikut adalah jumlah parameter yang dihasilkan yang akan digunakan untuk pelatihan model. (Vasudev, 2019)

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Conv_1 (Conv2D)             (None, 50, 50, 32)        320

 Conv_2 (Conv2D)             (None, 50, 50, 64)        18496

 max_pooling2d (MaxPooling2D  (None, 25, 25, 64)       0
 )

 Conv_3 (Conv2D)             (None, 25, 25, 128)       73856

 Conv_4 (Conv2D)             (None, 25, 25, 256)       295168

 max_pooling2d_1 (MaxPooling  (None, 12, 12, 256)      0
 2D)

 flatten (Flatten)           (None, 36864)             0

 Dense_1 (Dense)             (None, 256)               9437440

 Dense_2 (Dense)             (None, 32)                8224

 dense (Dense)               (None, 11)                363

 activation (Activation)     (None, 11)                0

=================================================================
Total params: 9,833,867
Trainable params: 9,833,867
Non-trainable params: 0
```

Gambar 3.15 *Model Summary*

Berikut adalah perhitungan jumlah parameter dari setiap layer yang digunakan.

a. *Convolution Layer 1*. Rumus untuk menghitung jumlah parameter dalam lapisan konvolusi yaitu: ((m * n * d) + 1) * k),
   m = lebar filter;
   n = tinggi filter;
   d = jumlah filter di lapisan sebelumnya;
   k = jumlah filter di lapisan saat ini.
   Jadi, jumlah parameter pada lapisan ini adalah ((3 x 3 x 1) + 1) x 32) = 320.
b. *Convolution Layer 2*. Parameter pada lapisan ini adalah ((3 x 3 x 32) + 1) x 64) = 18496.
c. *Max- Pooling 1*. Lapisan ini tidak memiliki parameter yang dapat dipelajari. Jadi jumlah parameter pada lapisan ini adalah 0.
d. *Convolution Layer 3*. Parameter pada lapisan ini adalah ((3 x 3 x 64) + 1) x 128) = 73856.
e. *Convolution Layer 4*. Parameter pada lapisan ini adalah ((3 x 3 x 128) + 1) x 256) = 295168.
f. *Max- Pooling 2*. Parameter pada lapisan ini adalah 0.
g. *Flatten*. Fungsi *flatten* digunakan untuk mengubah output dari lapisan konvolusi sebelumnya menjadi satu vektor (array 1D) yang akan menjadi input untuk lapisan-lapisan *Fully-Connected (Dense) Layer*. Untuk menghitung jumlah input yang akan digunakan pada dense layer yaitu:
   m * n * k,
   m = lebar filter lapisan sebelumnya;
   n = tinggi filter lapisan sebelumnya;
   k = jumlah filter lapisan sebelumnya.
   Jadi, jumlah input yang akan digunakan pada *dense layer* adalah 12 x 12 x 256 = 36864.
h. *Fully-Connected (Dense) Layer 1*. Rumus untuk menghitung jumlah parameter dalam lapisan *dense* yaitu: ((c * p) + (1 * c)),
   c = jumlah filter pada lapisan saat ini;
   p = jumlah neuron di lapisan sebelumnya.

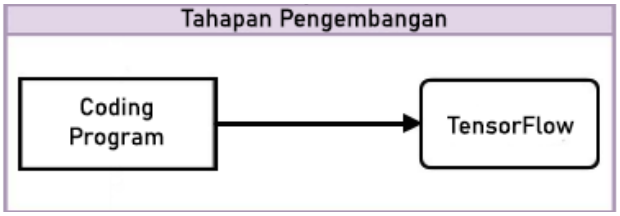
Jadi jumlah parameter pada lapisan ini adalah ((256 x 36864) + (1 x 256)) = 9437440.

i. *Fully-Connected (Dense) Layer 2*. Jumlah parameter pada lapisan ini adalah ((32 x 256) + (1 x 32)) = 8224.

j. *Fully-Connected (Dense) Layer 3*. Jumlah parameter pada lapisan ini adalah ((11 x 32) + (1 x 11)) = 363.

Total parameter dari keseluruhan lapisan adalah 320 + 18496 + 73856 + 295168 + 9437440 + 8224 + 363 = 9833867 parameter.

3.8 Tahapan Pengembangan



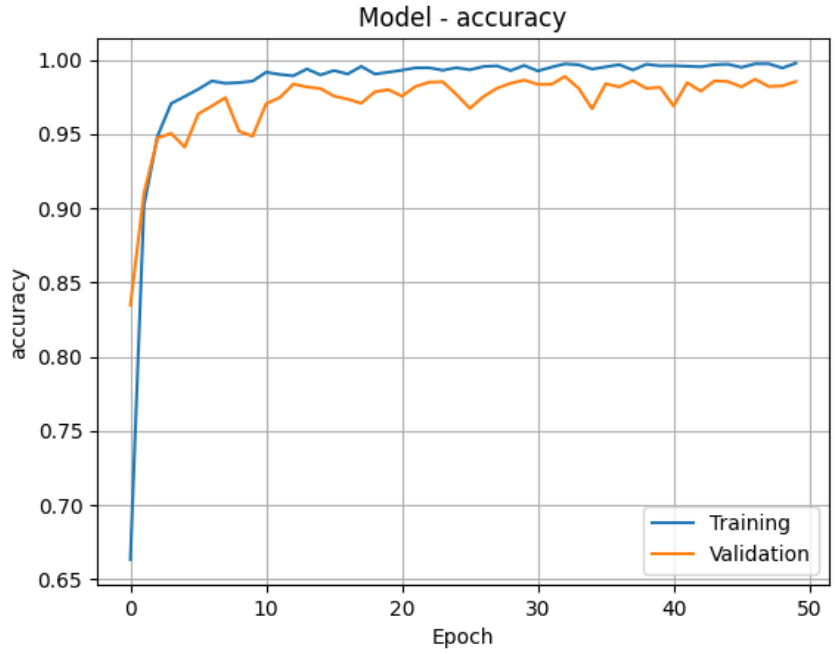
Gambar 3.16 Tahapan Pengembangan

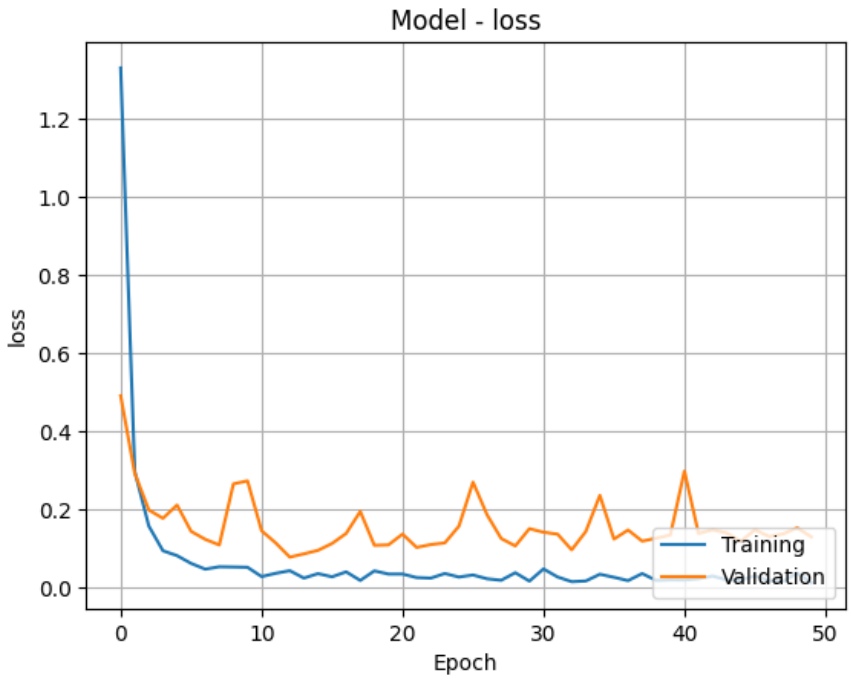
Setelah melalui tahap persiapan data, *pre-processing data*, pembagian dataset, dan perancangan arsitektur, langkah selanjutnya yaitu Tahapan Pengembangan model dengan melakukan proses pelatihan pada model. Model dikembangkan menggunakan sebuah *framework* sumber terbuka *(open-source)*, TensorFlow. Berikut adalah kode yang digunakan untuk melakukan proses pelatihan pada model.

```
input_shape = x_train[0].shape
EPOCHS = 50
BATCH_SIZE = 32
model = cnn_model(input_shape)
```

```
history = model.fit(x_train,
                    y_train,
                    epochs=EPOCHS,
                    batch_size=BATCH_SIZE,
                    shuffle=True,
                    validation_split=0.2)
```

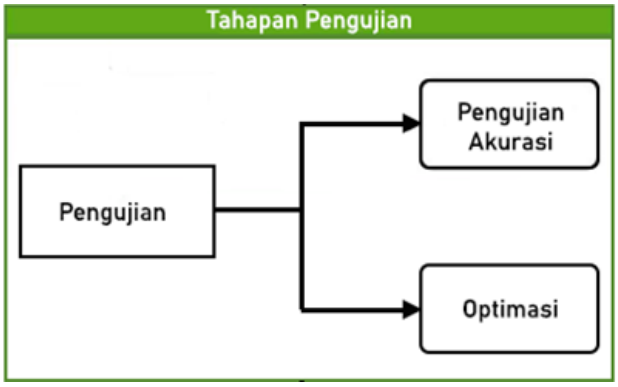
Berikut adalah metriks yang dihasilkan selama pelatihan model dengan 50 *epoch* (jumlah iterasi) dan 32 *batch size* (jumlah sampel yang digunakan pada setiap satu iterasi).





Gambar 3.17 *Accuracy* dan *Loss* pada pelatihan model

3.9 Tahapan Pengujian



Gambar 3.18 Tahapan Pengujian

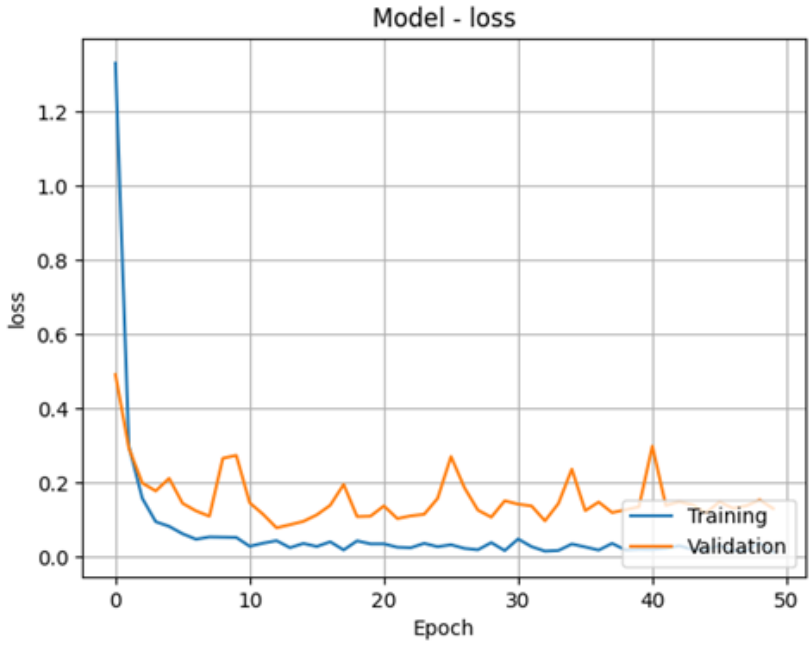
Tahap akhir dalam pengembangan model *deep learning* yaitu tahap pengujian, dimana dalam tahap ini, model diuji untuk melakukan prediksi pada data yang belum pernah dilihat sebelumnya dengan menggunakan dataset test yang telah dibagi pada proses pembagian dataset. Pada model ini, hasil metrik *Accuracy* dan *Loss* yang diperoleh yaitu:

```
model_test.evaluate(x_test, y_test)
```

```
121/121 [==============================] - 1s 10ms/step - loss: 0.1759 - accuracy: 0.9748
[0.17585498094558716, 0.9748051762580872]
```

Gambar 3.19 Hasil Pengujian Model

Model masih memiliki nilai *loss* yang tinggi. Artinya, model memiliki kesalahan yang tinggi pada data pengujian dan kinerjanya memungkinkan memberikan hasil prediksi yang tidak memuaskan. Untuk itu, perlu adanya diagnosa lebih lanjut untuk memperbaiki performa model.



Gambar 3.20 Kurva nilai *loss* dari hasil *training* pada model

Jika dilihat dari kurva pelatihan dan validasi model pada tahapan sebelumnya, kurva pada data validasi terlihat fluktuatif dan memiliki jarak pada kurva *training* dan *validation*. Jadi dapat disimpulkan bahwa model mengalami *overfitting* dimana nilai *validation loss* naik melebihi nilai *training loss*.

3.9.1 Generalisasi Model

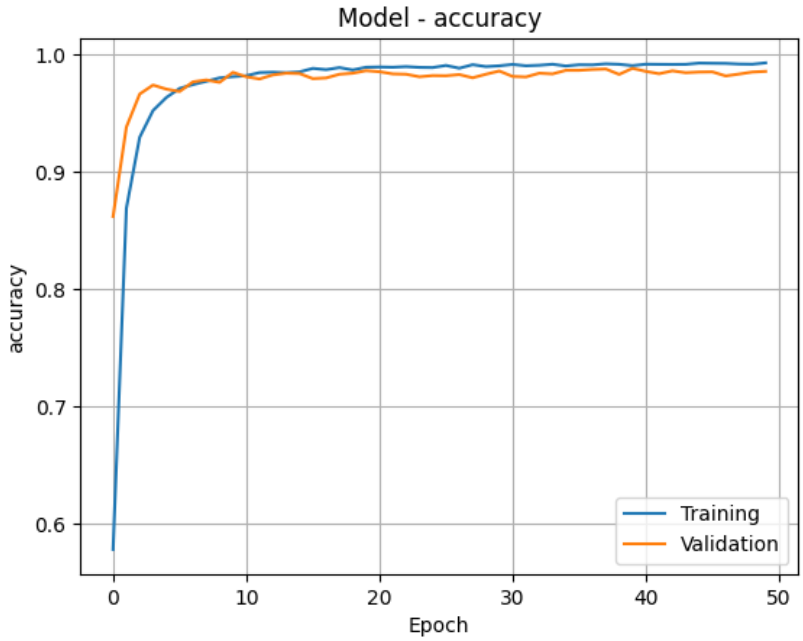
Proses optimasi model digunakan untuk meningkatkan kemampuan model dalam generalisasi untuk mengeluarkan prediksi yang akurat dan relevan pada data yang belum pernah dilihat sebelumnya. Tujuannya untuk mengembangkan model yang dapat belajar pola dan hubungan dari data pelatihan *(data training)* dan kemudian menerapkan pengetahuan yang diperoleh pada data uji *(data test)* yang belum pernah dilihat oleh model sebelumnya.

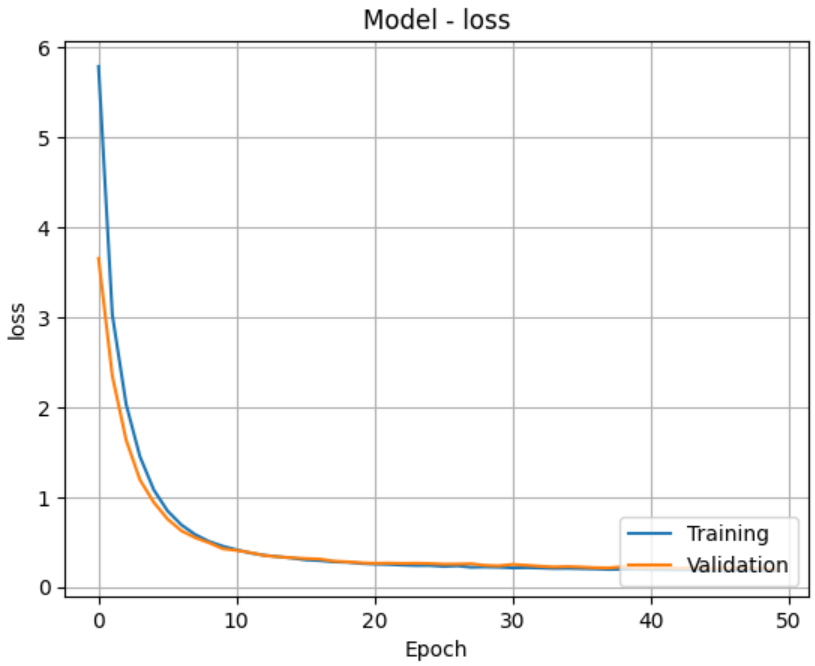
```
def cnn_model(input_shape):
    model = Sequential()
    model.add(Conv2D(32,
                     (3, 3),
                     name="Conv_1",
                     padding="same",
                     activation="relu",
                     input_shape=input_shape,
                     kernel_regularizer=regularizers.l2(1e-02)))
    model.add(Conv2D(64,
                     (3, 3),
                     name="Conv_2",
                     padding="same",
                     activation="relu",
                     input_shape=input_shape,
                     kernel_regularizer=regularizers.l2(1e-02)))
    model.add(MaxPool2D(pool_size=(2, 2)))
```

```
    model.add(Conv2D(128,
                     (3, 3),
                     name="Conv_3",
                     padding="same",
                     activation="relu",
                     kernel_regularizer=regularizers.l2(1e-02)))
    model.add(Conv2D(256,
                     (3, 3),
                     name="Conv_4",
                     padding="same",
                     activation="relu",
                     kernel_regularizer=regularizers.l2(1e-02)))
    model.add(MaxPool2D(pool_size=(2, 2)))
    model.add(Flatten())
    model.add(Dense(256,
                    name="Dense_1",
                    activation="relu",
                    kernel_regularizer=regularizers.l2(1e-02)))
    model.add(Dropout(0.1))
    model.add(Dense(32,
                    name="Dense_2",
                    activation="relu",
                    kernel_regularizer=regularizers.l2(1e-02)))
    model.add(Dropout(0.1))
```

```
    model.add(Dense(len(labels)))
    model.add(Activation("softmax"))
    model.summary()
    optimizer = keras.optimizers.Adam(learning_rate=1e-04)
    loss = keras.losses.CategoricalCrossentropy()
    model.compile(optimizer=optimizer,
                  loss=loss,
                  metrics=['accuracy'])
    return model
```

Pada model ini, dilakukan *L2 Regularization* dengan tingkat regularisasi sebesar 0,01 pada setiap lapisan, menambahkan lapisan *Dropout* pada lapisan *Fully-Connected (dense) layer* dengan probabilitas 0.1 (10%), dan menambahkan *learning rate* pada *optimizer* sebanyak 0.0001. Berikut adalah kurva dari hasil pelatihan yang telah diregularisasi.






```
model_test = load_model("model-cnn-facerecognition.h5")
model_test.evaluate(x_test, y_test)
```

```
121/121 [==============================] - 1s 9ms/step - loss: 0.2041 - accuracy: 0.9855
[0.20405049622058868, 0.9854545593261719]
```

Gambar 3.21 Hasil pembelajaran model setelah regularisasi

Hasil dari generalisasi sudah cukup baik jika dilihat dari kurva pelatihan dan validasi. Namun, hasil nilai loss masih cukup tinggi. Untuk itu, digunakan *hyperparameter tuning* untuk mendapatkan nilai *hyperparameter* terbaik agar model dapat bekerja secara optimal.

3.9.2 Hyperparameter Tuning

Hyperparameter tuning pada penelitian ini dilakukan untuk mencari val_loss terbaik dan nilai yang digunakan pada *regularizers* dan lapisan *dropout* dengan 35 *trails*. Berikut adalah hasil trail terbaik dan 10 *trails* terbaik dari 35 *trails* pada *hyperparameter tuning*.

```
Trial 35 Complete [00h 05m 44s]
val_loss: 0.36584359407424927

Best val_loss So Far: 0.06142796576023102
Total elapsed time: 00h 28m 35s
INFO:tensorflow:Oracle triggered exit
```

Gambar 3.22 Hasil dari *trail* terbaik pada *hyperparameter tuning*

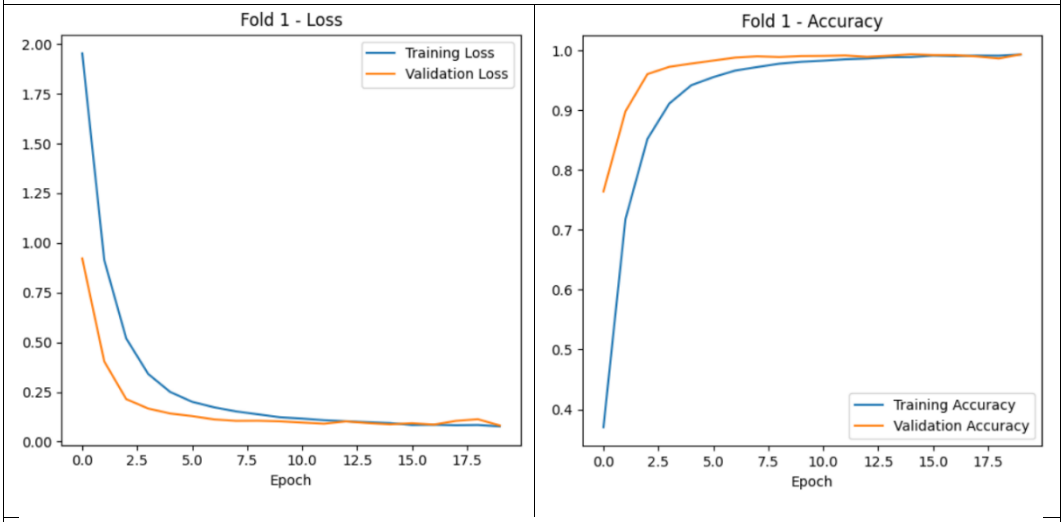
Trial 16 summary Hyperparameters: l2_regularization: 0.0001 dropout_rate_1: 0.25 dropout_rate_2: 0.2 Score: 0.06142796576023102	Trial 05 summary Hyperparameters: l2_regularization: 0.0001 dropout_rate_1: 0.1 dropout_rate_2: 0.25 Score: 0.0650591179728508
Trial 2 summary Hyperparameters: l2_regularization: 0.0001 dropout_rate_1: 0.2 dropout_rate_2: 0.1 Score: 0.06523200869560242	Trial 17 summary Hyperparameters: l2_regularization: 0.0001 dropout_rate_1: 0.300000000 dropout_rate_2: 0.35 Score: 0.06697758287191391
Trial 07 summary Hyperparameters: l2_regularization: 0.0001 dropout_rate_1: 0.2 dropout_rate_2: 0.450000000 Score: 0.07686296105384827	Trial 31 summary Hyperparameters: l2_regularization: 0.0001 dropout_rate_1: 0.25 dropout_rate_2: 0.35 Score: 0.07993989437818527
Trial 25 summary Hyperparameters: l2_regularization: 0.0001 dropout_rate_1: 0.30000000 dropout_rate_2: 0.5 Score: 0.0808633491396904	Trial 21 summary Hyperparameters: l2_regularization: 0.001 dropout_rate_1: 0.2 dropout_rate_2: 0.1 Score: 0.0899420827627182
Trial 06 summary Hyperparameters: l2_regularization: 0.0001 dropout_rate_1: 0.25 dropout_rate_2: 0.15000000 Score: 0.09216005355119705	Trial 14 summary Hyperparameters: l2_regularization: 0.001 dropout_rate_1: 0.30000000 dropout_rate_2: 0.15000000 Score: 0.09475275874137878

Tabel 3.3 10 *Trails* terbaik pada *hyperparameter tuning*

3.9.3 Cross-Validation

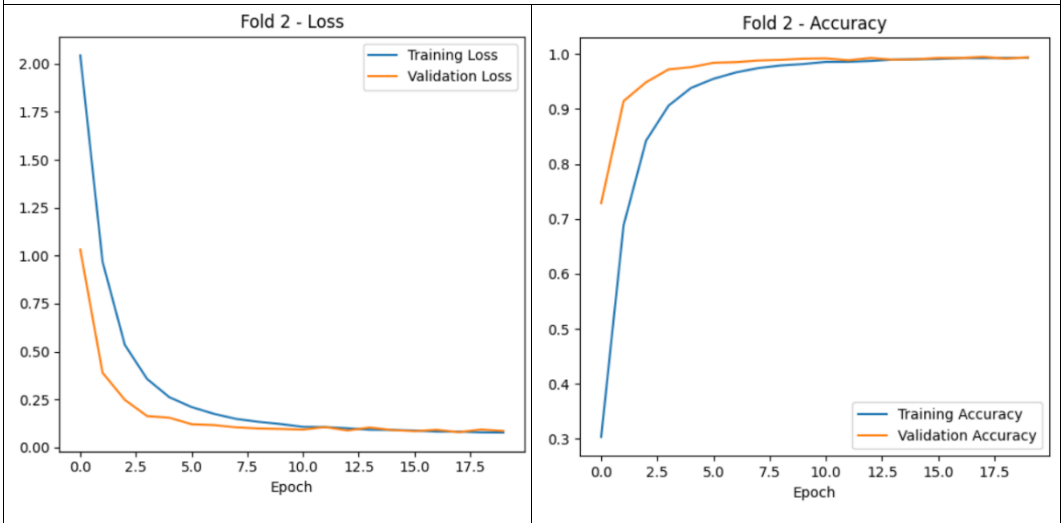
Metode *Cross-Validation* pada penelitian ini dilakukan menggunakan 5-Fold dengan masing-masing data validasi yang digunakan diambil dari 20% data training. Berikut adalah hasil 5 model dari 5-Fold yang dihasilkan.

Fold 1



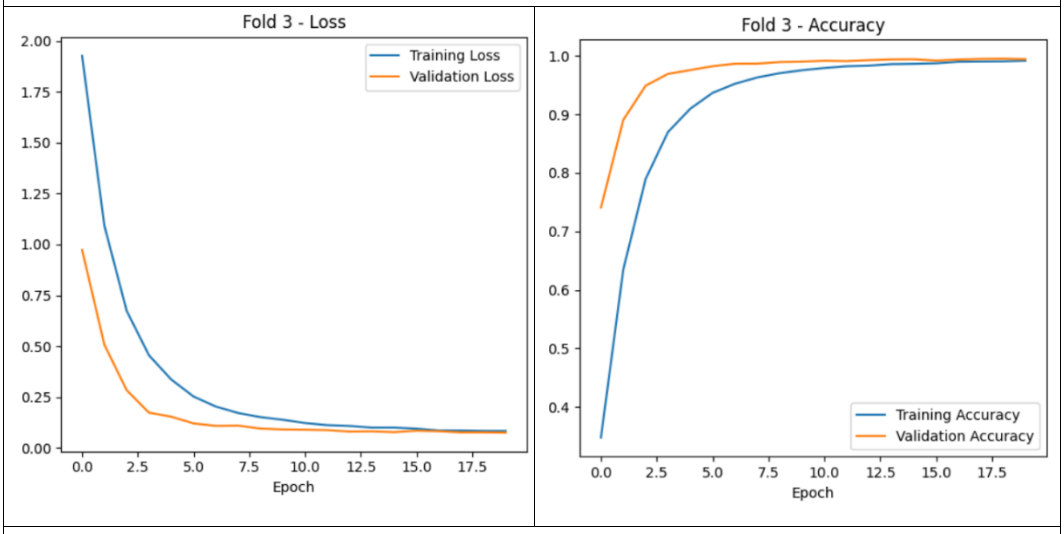

```
Fold 1 - Loss: 0.0951789990067482, Accuracy: 0.9927272796630859
```

Fold 2



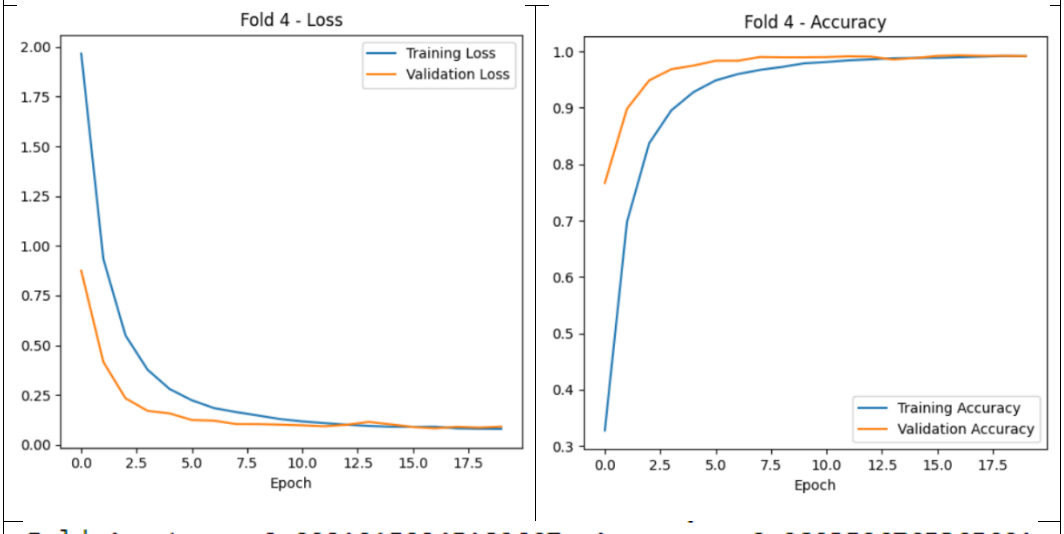

```
Fold 2 - Loss: 0.09402814507484436, Accuracy: 0.9927272796630859
```

Fold 3

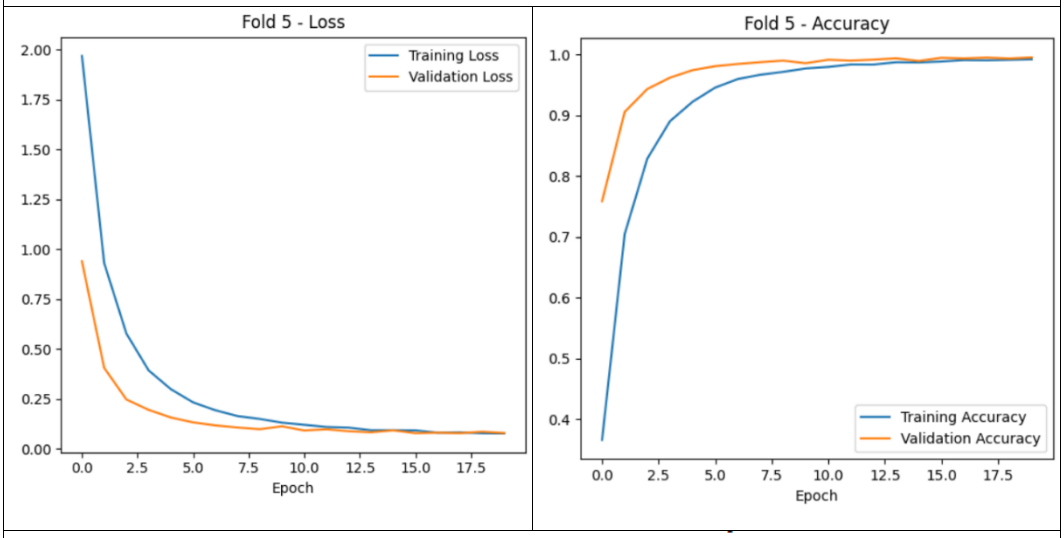



```
Fold 3 - Loss: 0.0824093222618103, Accuracy: 0.9942857027053833
```

Fold 4




```
Fold 4 - Loss: 0.09219150245189667, Accuracy: 0.9893506765365601
```

Fold 5

Fold 5 - Loss: 0.09251967072486877, Accuracy: 0.991428554058075

Tabel 3.4 Hasil 5-Fold dari *Cross-Validation*

Untuk pengambilan model terbaik dilakukan dengan mengambil nilai tertinggi dari rata-rata akurasi dari kelima model yang dihasilkan, sehingga terpilih Fold 3 sebagai model terbaik.

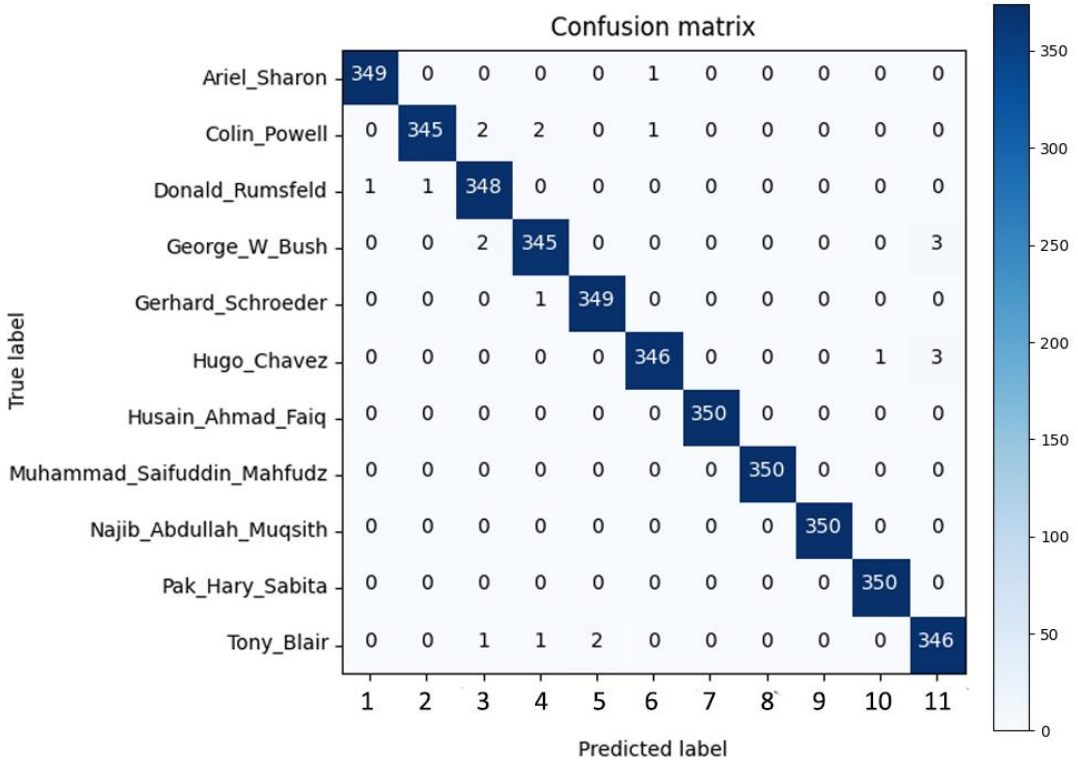
```
Fold 3 - Loss: 0.0824093222618103, Accuracy: 0.9942857027053833
```

Gambar 3.23 Hasil model terbaik dari metode *Cross-Validation*

3.9.4 Perhitungan Akurasi Menggunakan Confusion Matrix

Confusion matrix adalah suatu metode evaluasi performa model klasifikasi yang menyajikan hasil prediksi model dalam bentuk tabel, membandingkan prediksi dengan nilai sebenarnya. Berikut adalah rumus *confusion matrix* untuk menghitung akurasi:

$$\text{Akurasi} = \frac{\text{Jumlah prediksi benar}}{\text{Jumlah keseluruhan data}}$$



Gambar 3.24 *Confusion Matrix*

Keterangan:

1 = Ariel Sharon	5 = Gerhard Schroeder	9 = N Abdullah Muqsith
2 = Colin Powell	6 = Hugo Chavez	10 = Pak Hary Sabita
3 = Donald Rumsfeld	7 = Husain Ahmad Faiq	11 = Tony Blair
4 = George W Bush	8 = M Saifuddin Mahfudz	

Perhitungan nilai akurasi yang dihasilkan:

$$\text{Akurasi} = \frac{\text{Jumlah prediksi benar}}{\text{Jumlah keseluruhan data}}$$

$$\text{Akurasi} = \frac{3828}{3850}$$

Akurasi = 0.9942857

Akurasi = 99%