

**DOKUMEN CD - 4**



**KLASIFIKASI JENIS BATIK  
MENGUNAKAN MACHINE LEARNING  
BERBASIS APLIKASI**

Oleh :


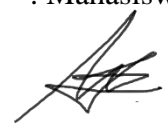


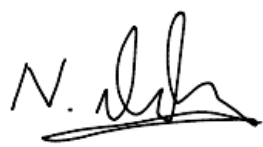
Aulia Chusnyriani Sani Z	/ 1101194043
I Gusti Ngurah Rejski A. P	/ 1101190017
Nada Fauzia Reviana	/ 1101194198
Rahmawati Hidayah	/ 1101194070

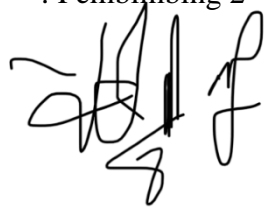

**PRODI S1 TEKNIK TELEKOMUNIKASI  
FAKULTAS TEKNIK ELEKTRO  
UNIVERSITAS TELKOM  
BANDUNG  
2023**

## Dokumentasi Produk Capstone Design

### Lembar Pengesahan Dokumen

Judul Capstone Design : Klasifikasi Jenis Batik Menggunakan  
 Machine Learning Berbasis Aplikasi  
 Jenis Dokumen : Implementasi  
 Nomor Dokumen : FTE-CD-4  
 Nomor Revisi : 1(sesuaikan dengan jumlah revisi)  
 Tanggal Pengesahan : 18/06/2023  
 Fakultas : Fakultas Teknik Elektro  
 Program Studi : S1 Teknik Telekomunikasi  
 Jumlah Halaman : 35

Data Pemeriksaan dan Persetujuan			
Ditulis Oleh	Nama : Aulia Chusnyriani Sani Zulkarnaen	Jabatan : Mahasiswa	
	NIM : 1101194043	Tanda Tangan	
	Nama : I Gusti Ngurah Rejski A. P	Jabatan : Mahasiswa	
	NIM : 1101190017	Tanda Tangan	
	Nama : Nada Fauzia Reviana	Jabatan : Mahasiswa	
	NIM : 1101194198	Tanda Tangan	
	Nama : Rahmawati Hidayah	Jabatan : Mahasiswa	
	NIM : 1101194070	Tanda Tangan	
Disetujui Oleh	Nama : Nur Ibrahim S.T, M.T. Tanggal :	Jabatan : Pembimbing 1 Tanda Tangan	

	Nama : R Yunendah Nur Fuadah, S.T, M.T. Tanggal :	Jabatan : Pembimbing 2 Tanda Tangan 
	Nama : Nor Kumalasari Caecar Pratiwi S.T, M. T Tanggal :	Jabatan : Pembimbing 3 Tanda Tangan 

**Timeline Revisi Dokumen**

Versi, Tanggal	Revisi	Perbaikan dilakukan	yang	Halaman Revisi

## DAFTAR ISI

<b>1</b>	<b>Pengantar .....</b>	<b>7</b>
<i>1.1</i>	<i>Ringkasan Isi Dokumen .....</i>	<i>7</i>
<i>1.2</i>	<i>Tujuan Penulisan dan Aplikasi .....</i>	<i>7</i>
<i>1.3</i>	<i>Referensi .....</i>	<i>7</i>
<i>1.4</i>	<i>Daftar Singkatan .....</i>	<i>8</i>
<b>2</b>	<b>Implementasi Sistem.....</b>	<b>8</b>
<i>2.1</i>	<i>Machine Learning .....</i>	<i>8</i>
2.1.1	Cara Kerja Machine Learning .....	8
2.1.2	Implementasi .....	13
2.1.3	Pengujian .....	18
<i>2.2</i>	<i>Mobile Application BatiQu .....</i>	<i>21</i>
2.2.1	Cara Kerja Mobile Application .....	21
2.2.2	Implementasi .....	22
2.2.3	Pengujian .....	23
2.2.4	Analisis Pengerjaan Implementasi Sistem .....	23
<b>3</b>	<b>Analisis Pengerjaan Implementasi Sistem.....</b>	<b>24</b>
<b>4</b>	<b>Hasil Akhir .....</b>	<b>25</b>
<i>4.1</i>	<i>Machine Learning .....</i>	<i>25</i>
4.1.1	Tampilan Tabel Confusion Matrix dari Program Klasifikasi.....	25
4.1.2	Hasil Classification Report dari Program Klasifikasi .....	26
<i>4.2</i>	<i>Mobile Application .....</i>	<i>27</i>
4.2.1	Tampilan AntarMuka Kamera dan Galeri Aplikasi.....	27
<b>5</b>	<b>Lampiran .....</b>	<b>28</b>
<i>5.1</i>	<i>Lampiran 1 .....</i>	<i>28</i>
<i>5.2</i>	<i>Lampiran 2 .....</i>	<i>31</i>

## DAFTAR GAMBAR

<b>Gambar 2.1.1.1</b> gambar Batik Tambal (a) Batik Parang (b) Batik Nitik (c) Batik Megamendung (d) Batik Kawung (e) Batik Ceplok (f) .....	9
<b>Gambar 2.1.1.2</b> Flowchart Pre-processing dengan Augmentasi data.....	10
<b>Gambar 2.1.1.3</b> Flowchart ekstraksi fitur dengan model Resnet152 V2.....	11
<b>Gambar 2.1.1.4</b> Flowchart alur proses klasifikasi .....	12
<b>Gambar 2.1.2.1</b> Program Pre-processing menggunakan Augmentasi Data .....	13
<b>Gambar 2.1.2.2</b> Program Augmentasi data dengan data Testing .....	14
<b>Gambar 2.1.2.3</b> Visualisasi data dengan teknik augmentasi Horizontal Flip.....	14
<b>Gambar 2.1.2.4</b> Visualisasi data dengan teknik augmentasi Brighthness .....	15
<b>Gambar 2.1.2.5</b> Program ekstraksi fitur dengan model ResNet152 V2 .....	15
<b>Gambar 2.1.2.6</b> Program untuk mengatur optimizer dan learning rate .....	16
<b>Gambar 2.1.2.7</b> Program untuk proses pelatihan dan evaluasi model.....	17
<b>Gambar 2.1.2.8</b> Program untuk visualisasi performa model dari proses training. ....	18
<b>Gambar 2.1.3.1</b> Hasil dari Visualisasi Augmentasi data dengan Horizontal .....	19
<b>Gambar 2.1.3.2</b> Hasil dari model ekstraksi fitur dengan ResNet152 V2 .....	19
<b>Gambar 2.1.3.3</b> Hasil rangkuman model ekstraksi fitur dengan ResNet152 V2..	20
<b>Gambar 2.1.3.4</b> Hasil program untuk proses pelatihan dan evaluasi model .....	21
<b>Gambar 2.1.3.5</b> Hasil visualisasi dari akurasi dan loss yang telah dilatih.....	21
<b>Gambar 2.2.1.1</b> Flowchart dari cara kerja Mobile Application.....	22
<b>Gambar 4.2.1</b> Tampilan AntarMuka Kamera dan Galeri Aplikasi.....	27
<b>Gambar 4.2.2</b> Tampilan AntarMuka About Us pada Aplikasi .....	28

## DAFTAR TABEL

<b>Tabel 1.4.1</b> Daftar Singkatan.....	8
<b>Tabel 2.2.1</b> Rencana pengerjaan Capstone Design .....	24
<b>Tabel 2.2.2</b> Hasil implementasi pengerjaan fase 2 Capstone Design.....	24
<b>Tabel 4.1.1</b> Confusion Matrix dengan metode ResNet152 V2 .....	25

## 1 Pengantar

### 1.1 Ringkasan Isi Dokumen

Dokumen *Capstone Design-4* dirancang menjadi beberapa bagian, yaitu pengantar, implementasi sistem, analisis pengerjaan implementasi sistem, hasil akhir, dan Lampiran. Dalam Pengantar terdapat bagian ringkasan isi dokumen, tujuan adanya penulisan dan aplikasi, referensi dan daftar singkatan. Pada bagian implementasi sistem yang berisikan mengenai cara kerja dari tiap-tiap subsistem, kemudian terdapat implementasi yang tujuannya untuk penempatan *source code* sebagai penunjang yang berhubungan dengan cara kerja sub-sistem dari dokumen ini, selanjutnya ada pengujian dimana hasil implementasi dan bagaimana hasil pengujiannya digunakan untuk mengukur kinerja dari sistem tersebut. Terdapat analisis pengerjaan implementasi sistem, hasil akhir dari pengerjaan dokumen *Capstone Design-4* dengan nama dari *Mobile Application* adalah “BatiQu” yang didukung menggunakan *software Android Studio*. Serta yang terakhir terdapat lampiran terkait rujukan yang dapat ditambahkan sebagai informasi pendukung.

### 1.2 Tujuan Penulisan dan Aplikasi

1. Penulisan dokumen *Capstone Design-4* ini ditujukan untuk memenuhi kewajiban pada kelas Proposal Tugas Akhir.
2. Merancang, menganalisis dan mengimplementasikan sistem kerja dari klasifikasi jenis Batik menggunakan aplikasi *software mobile apps (android)*.
3. Melakukan klasifikasi jenis batik dengan metode CNN.
4. Membantu masyarakat agar lebih banyak mengenal jenis ragam Batik Indonesia.
5. Melakukan analisis hasil sehingga dapat diidentifikasi berdasarkan motif dan jenis Batik Indonesia serta melakukan analisis pada accuracy dan loss berdasarkan parameter metode yang digunakan dalam *Machine Learning*.

### 1.3 Referensi

- [1] M. Afif, A. Fawwaz, K. N. Ramadhani, and F. Sthevanie, “Klasifikasi Ras pada Kucing menggunakan Algoritma Convolutional Neural Network(CNN),” *J. Tugas Akhir Fak. Inform.*, vol. 8, no. 1, pp. 715–730, (2020).



## 1.4 Daftar Singkatan

Singkatan	Arti
CNN	<i>Convolutional Neural Network</i>
ML	<i>Machine Learning</i>
XML	<i>eXtensible Markup Language</i>
IDE	<i>Integrated Development Environment</i>
UI	<i>User Interface</i>

**Tabel 1.4.1** Daftar Singkatan

## 2 Implementasi Sistem

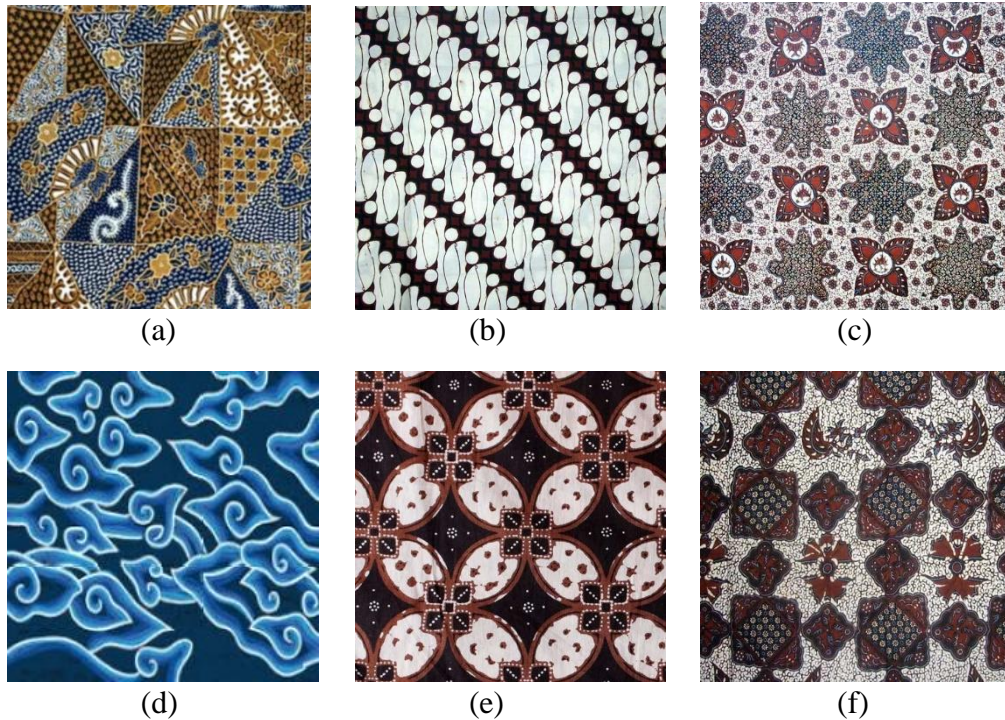
Implementasi sistem merupakan tahapan atau prosedur yang dilakukan untuk dapat menyelesaikan perancangan sistem yang telah dibuat dan akan dilakukan pengujian. Dimana dalam implementasi sistem juga disiapkan sistem yang akan dijalankan. Tujuan Implementasi Sistem ini agar peneliti dapat menyelesaikan desain sistem yang telah dirancang dalam dokumen sebelumnya serta mendokumentasikan prosedur yang diperlukan. Memastikan serta memperhitungkan bahwa sistem dapat beroperasi dengan baik serta memenuhi parameter yang sudah disediakan.

### 2.1 Machine Learning

#### 2.1.1 Cara Kerja Machine Learning

##### 2.1.1.1 Pengumpulan data

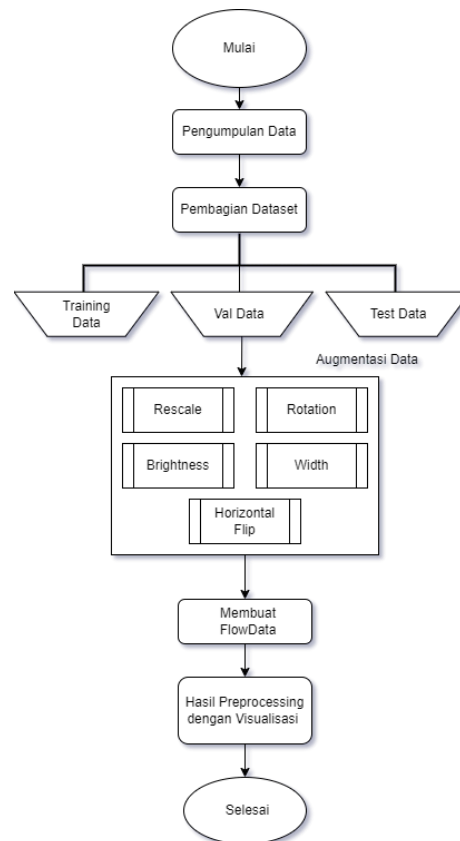
Data yang dikumpulkan adalah gambar batik dengan jumlah 660 gambar yang dikelompokkan menjadi 6 kelas yaitu batik kawung, batik ceplok, batik tambal, batik megamendung, batik parang dan batik nitik. Setelah data dibagi menjadi beberapa kelompok kemudian data dibagi lagi menjadi data training, data testing dan juga data validasi. Tujuan pengelompokkan data menjadi 3 folder untuk dilakukan pelabelan disetiap datanya. Pada data training berjumlah 480 gambar, data validasi berjumlah 120 data dan untuk data testing berjumlah 60 gambar.



**Gambar 2.1.1.1** gambar Batik Tambal (a) Batik Parang (b) Batik Nitik (c) Batik Megamendung (d) Batik Kawung (e) Batik Ceplok (f)

#### 2.1.1.2 *Pre-processing*

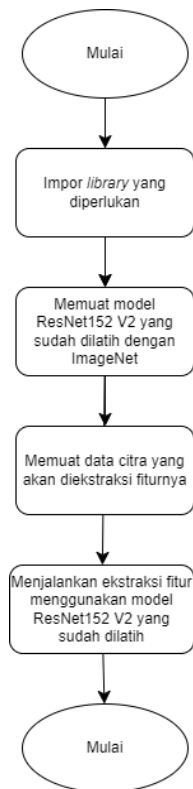
Setelah proses dari pengumpulan data, tahapan selanjutnya adalah adanya *pre-processing*. *Pre-processing* adalah tahapan awal pengolahan data gambar pada *Convolutional Neural Network* (CNN). Dimana teknik *pre-processing* digunakan untuk mempersiapkan data input sebelum memasuki proses bagian input model CNN. Tujuan dari *pre-processing* adalah untuk meningkatkan kualitas gambar maupun kinerja model dengan meminimalkan informasi yang tidak valid dan mempertahankan informasi penting yang dapat dilanjutkan kedalam proses selanjutnya. Augmentasi data merupakan salah satu metode untuk dapat mengurangi *overfitting* dengan meningkatkan ukuran citra dataset dan akurasi dengan tujuan meminimumkan data agar komputasi lebih cepat. Dengan augmentasi data juga dapat memodifikasi dan memperbanyak citra agar menciptakan variasi sampel data baru sehingga kualitas dan keragaman data lebih banyak sebelum diinputkan kedalam model. Dibawah ini akan ditampilkan dari proses atau *flowchart pre-processing* pada gambar 2.1.1.2.



**Gambar 2.1.1.2** Flowchart Pre-processing dengan Augmentasi data

### 2.1.1.3 Feature extraction

Ekstraksi fitur adalah proses untuk mengambil fitur atau karakteristik dari data yang dimasukkan (dapat berupa teks atau gambar). Fitur atau karakteristik yang diambil berupa sifat-sifat khusus dari data tersebut yang dapat membedakan antara data satu dengan data yang lainnya seperti tekstur, warna, bentuk, dan pola. Secara umum proses ekstraksi fitur pada CNN langsung dimulai dengan memuat model arsitektur yang digunakan. Hal ini disebabkan karena pada CNN itu ekstraksi fiturnya menjadi satu dengan metode klasifikasinya. Kemudian, dataset yang akan diuji dimuat di dalam model tersebut untuk diekstraksi fiturnya dengan model arsitektur yang digunakan. Pada aplikasi BatiQu ini, ekstraksi fiturnya menggunakan basis arsitektur ResNet152 V2 yang sudah dilatih dengan menggunakan dataset ImageNet. Penggunaan bobot *ImageNet* ini dapat membantu dalam proses ekstraksi fitur dan dapat meningkatkan performa dari sistem klasifikasi BatiQu.



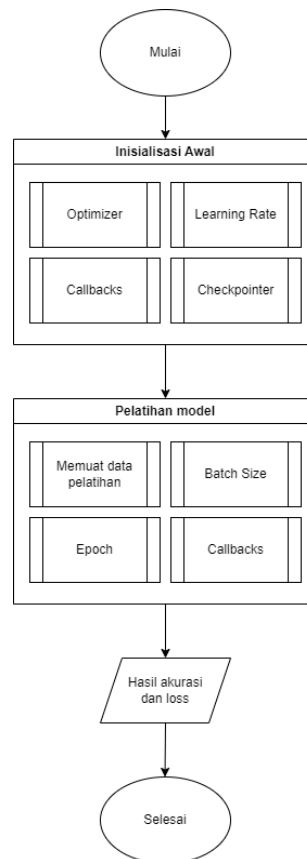
**Gambar 2.1.1.3** Flowchart ekstraksi fitur dengan model Resnet152 V2

Seperti yang terlihat pada flowchart pada gambar 2.1.1.3, *flowchart* tersebut menjelaskan tentang alur program dalam melakukan ekstraksi fitur dengan model ResNet152 V2.

#### 2.1.1.4 Classification

Klasifikasi pada metode *Convolutional Neural Network* atau yang biasa disingkat menjadi CNN memiliki suatu pengertian sebagai proses pengelompokan dan identifikasi suatu citra atau objek gambar ke dalam kelas tertentu. Klasifikasi menjadi bentuk pembelajaran dalam metode CNN agar menempatkan kelas-kelas berdasarkan fitur yang telah dipelajari setelah melewati pelatihan atau proses *training*. Dalam hal ini, terdapat dua klasifikasi dari arsitektur yang berbeda untuk dibandingkan yaitu arsitektur MobileNet V1 dan Resnet152 V2. Arsitektur MobileNet V1 dan Resnet152 V2 memiliki beberapa kesamaan hyperparameter dan juga parameter yang akan digunakan seperti *Optimizer*, *Learning Rate*, *Callbacks*, *Batch Size*, *Epochs*. Nilai hyperparameter serta parameter dapat memengaruhi

output model dari proses pelatihan programnya.



**Gambar 2.1.1.4** Flowchart alur proses klasifikasi

Pada flowchart alur proses klasifikasi diawali dengan inisialisasi variabel yang diperlukan. Tahap awal dimulai dengan menambahkan *optimizer*, *learning rate*, *callbacks*, dan *checkpoint*. Dalam tahapan ini, *optimizer* digunakan untuk mengoptimalkan bobot model agar *loss* yang dihasilkan antara nilai *output* dengan hasil nilai dari neuron dapat diminimalisir saat proses *training*. Sedangkan, *learning rate* merupakan hyperparameter untuk mengatur nilai perubahan dari bobot model selama proses pelatihan. Selanjutnya, terdapat parameter *callbacks* yang berfungsi untuk memanggil fungsi *checkpoint*. *Checkpoint* sendiri memiliki peran untuk menyimpan model dengan performa terbaik.

Selanjutnya adalah mengatur pelatihan dari model yang telah dibuat. Tahapan pertama adalah memuat data yang akan digunakan untuk melakukan proses *training*, *validation*, dan *testing*. Setelah itu, dilakukan pengaturan pada *batch size* dan *epoch* yang berfungsi untuk menginterpretasikan jumlah data pelatihan dalam satu *batch* yang dapat memengaruhi tingkat akurasi. Sedangkan, *epoch* merupakan jumlah pemrosesan model untuk dari seluruh dataset yang telah



dilatih oleh program.

### 2.1.2 Implementasi

Implementasi pada Klasifikasi Batik dilakukan untuk membuat aplikasi yang mampu melakukan prediksi citra gambar menggunakan model *Machine Learning* (ML) dan mengirimkan citra ke BatiQu. Adapun implementasi pada *pre-processing* sebagai berikut :

```
#memanggil seluruh citra dan menampungnya ke dalam array yang sudah
#disediakan, serta merize sesuai kebutuhan
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   brightness_range=[0.1, 1.5],
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   width_shift_range=0.2,
                                   horizontal_flip=True)

test_datagen = ImageDataGenerator(rescale=1./255)

train_generator =
train_datagen.flow_from_directory('/content/drive/MyDrive/Khusus ALC/UJI
COBA TA/DATASET BATIK MANUAL/Data Train',
                                target_size=(300, 300),
                                batch_size= 32,
                                shuffle= False,
                                class_mode= 'categorical')

validation_generator =
test_datagen.flow_from_directory('/content/drive/MyDrive/Khusus ALC/UJI
COBA TA/DATASET BATIK MANUAL/Data Val',
                                target_size=(300, 300),
                                batch_size= 32,
                                class_mode= 'categorical')
```

**Gambar 2.1.2.1** Program *Pre-processing* menggunakan Augmentasi Data

Pada tabel 2.1.1.2.1 menjelaskan mengenai implementasi dari *pre-processing* menggunakan '*ImageDataGenerator*', proses ini digunakan untuk memisahkan data yang digunakan untuk training data ataupun validasi data. Proses untuk menguji test pada *pre-processing* sebagai berikut :

```
test_datagen = ImageDataGenerator(rescale = 1./255) #Image normalization.

test_set = test_datagen.flow_from_directory('/content/drive/MyDrive/Khusus
ALC/UIJ COBA TA/DATASET BATIK MANUAL/Data Testing',
                                         target_size = (300, 300),
                                         batch_size = 32,
                                         class_mode = 'categorical')
```

**Gambar 2.1.2.2** Program Augmentasi data dengan data *Testing*

Implementasi untuk *pre-processing* ini, dilanjutkan dengan memvisualisasikan hasil *pre-processing* hanya dengan dua teknik yang digunakan. Berikut Visualisasi data yang akan ditampilkan :

```
from tensorflow import keras
image_path = '/content/drive/MyDrive/Khusus ALC/UIJ COBA
TA/DATASET BATIK MANUAL/Data Train/Parang/46.jpg'

img = keras.preprocessing.image.load_img(image_path, target_size= (300,
300))
img_tensor = keras.preprocessing.image.img_to_array(img)
img_tensor = np.expand_dims(img_tensor, axis=0)

#Gunakan ImageDataGenerator untuk memutar gambar
datagen = ImageDataGenerator(horizontal_flip=True)

#Creates our batch of one image
pic = datagen.flow(img_tensor, batch_size= 32)
plt.figure(figsize=(10,5))

#Plots semua figur
for i in range(1,4):
    plt.subplot(1, 3, i)
    batch = pic.next()
    image_ = batch[0].astype('uint8')
    plt.imshow(image_)
plt.show()
```

**Gambar 2.1.2.3** Visualisasi data dengan teknik augmentasi *Horizontal Flip*

Setelah hasil dari *pre-processing* keluar, dilanjutkan dengan melakukan visualisasi data dari augmentasi data yang kemudian proses dilanjutkan ke tahap *Feature Extraction* dan juga Klasifikasi.

```

#Uses ImageDataGenerator to flip the images
datagen = ImageDataGenerator(brightness_range=[0.1, 1.5])
#Creates our batch of one image
pic = datagen.flow(img_tensor, batch_size= 32)
plt.figure(figsize=(10,5))

#Plots our figures
for i in range(1, 4):
    plt.subplot(1, 3, i)
    batch = pic.next()
    image_ = batch[0].astype('uint8')
    plt.imshow(image_)
plt.show()

```

**Gambar 2.1.2.4** Visualisasi data dengan teknik augmentasi *Brighness*

Tahapan berikutnya adalah *Feature extraction*, berikut merupakan implementasi pada *Feature extraction*:

```

model = tf.keras.Sequential([ ResNet152V2(
                                include_top=False,
                                weights='imagenet',
                                input_shape=(300, 300, 3)),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(6, activation='softmax')
])

```

**Gambar 2.1.2.5** Program ekstraksi fitur dengan model ResNet152 V2

Tahapan ekstraksi fitur diimplementasikan dalam bentuk program seperti yang terlihat pada gambar 2.1.2.5. Pada program tersebut, digunakan ekstraksi fitur dengan model ResNet152 V2 yang sudah dilatih dengan dataset imagenet dengan menambahkan tulisan '*weights="imagenet"*'. Pada awal program, dituliskan '*model = tf.keras.Sequential*' yang berfungsi untuk membuat objek model *sequential* sehingga *layer-layer* dapat ditambahkan ke dalam model secara berurutan. Kemudian di dalam fungsi ResNet152V2 ditambahkan argumen *include\_top=False* karena penelitian ini berfokus pada bagian ekstraksi fitur dari model ini dan tidak memerlukan program untuk menyertakan *layer output* terakhir. Selanjutnya ditambahkan juga dimensi dari *input* citra yang digunakan dengan mencantumkan '*input\_shape=(300, 300, 3)*', lalu disambung dengan menambahkan program untuk *layer*. Pada program *layer*, digunakan *layer flatten* untuk mengubah output dari model ResNet152 V2 menjadi vektor satu dimensi, kemudian



dilanjutkan dengan ‘*layer dense*’ dengan 128 unit dan aktivasi ‘ReLU’ agar representasi yang dihasilkan dari proses fiturnya menjadi lebih kompleks. Yang terakhir ditambahkan ‘*layer dense*’ dengan 6 unit karena disesuaikan dengan jumlah kelas klasifikasinya dan disertai dengan fungsi aktivasi ‘*softmax*’ untuk menghasilkan probabilitas prediksi kelas sesuai dengan hasil ekstraksi fitur yang dimasukkan ke model. Tahapan terakhir untuk proses ini adalah Klasifikasi, berikut merupakan implementasi Klasifikasi :

```
#setting optimizer
filepath='model.h5'
opt = tf.keras.optimizers.legacy.Adam(learning_rate=0.0001 #learning rate
biasanya antara 0.001 - 0.1. jika terlalu besar akan terjadi overshooting
model.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['acc'])

checkpointer = ModelCheckpoint(filepath,
                               monitor = 'val_acc',
                               verbose=1,
                               save_best_only=True,
                               mode='max')
callbacks_list = [checkpointer]
```

**Gambar 2.1.2.6** Program untuk mengatur optimizer dan learning rate

Pada table 2.1.2.6 merupakan tahapan klasifikasi untuk mendefinisikan bahwa ‘*optimizer*’ yang digunakan yaitu Adam dan learning rate yang diatur sebesar 0.0001. ‘Optimizer’ Adam sangat umum digunakan dalam metode CNN karena dapat memberikan hasil yang lebih baik. Untuk pemilihan nilai learning rate berada dalam rentang 0 sampai 1 [1]. Learning rate dapat mempengaruhi kecepatan

dari hasil pelatihan model.

```

model_history = []
history = model.fit(
    train_generator,
    validation_data = validation_generator,
    batch_size= 32,
    epochs= 200,
    callbacks= [checkpointer]
)

# Evaluasi model pada data pelatihan
train_score = model.evaluate(train_generator)
print('Train Score: ', train_score)

val_score = model.evaluate(validation_generator)
print('Val Score: ', val_score)

test_score = model.evaluate(test_set)
print('Test Score: ', test_score)

```

**Gambar 2.1.2.7** Program untuk proses pelatihan dan evaluasi model.

Setelah mengatur nilai pada bagian optimizer dan learning rate, pada tabel 2.1.2.7 merupakan program untuk melakukan pelatihan dan evaluasi model. Program tersebut telah memproses data *training* dan data *validation* yang akan dilatih dengan ‘train\_generator’ dan dievaluasi dengan ‘validation\_generator’. Selanjutnya, terdapat batch\_size yang terdiri dari 32 sampel. Selain itu, parameter epoch sebesar 200 juga diperlukan untuk mencapai tingkat akurasi terbaik. Nilai epoch menentukan hasil pembelajaran model dalam memahami pola dari dataset yang telah dilatih. Apabila kedua hyperparameter tersebut telah disesuaikan, maka diperlukan evaluasi kinerja model pada data *training*, data *validation*, dan data *testing*. Evaluasi dari ketiga dataset tersebut akan ditampilkan dengan code ‘print’ untuk menghasilkan skor atau performa model. Hal ini mengacu pada data yang

telah dilatih dapat memahami dan memprediksi dengan akurasi yang baik.

```
print(history.history.keys())
# summarize history for accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()

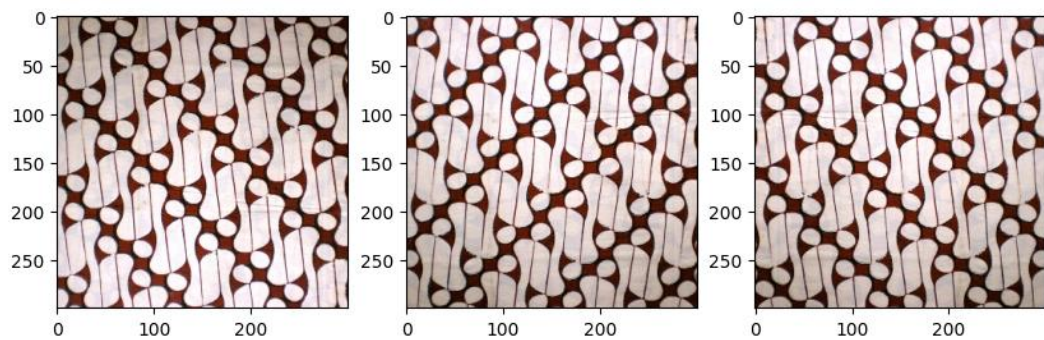
# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```

**Gambar 2.1.2.8** Program untuk visualisasi performa model dari proses training.

Setelah melakukan pelatihan dan evaluasi kinerja model, program pada Tabel 2.1.2.8 dapat membantu menampilkan perubahan akurasi dan *loss* model selama proses *training*. Hasil visualisasi ini dapat dianalisis dan dipahami lebih dalam mengenai performa yang telah didapatkan.

### 2.1.3 Pengujian

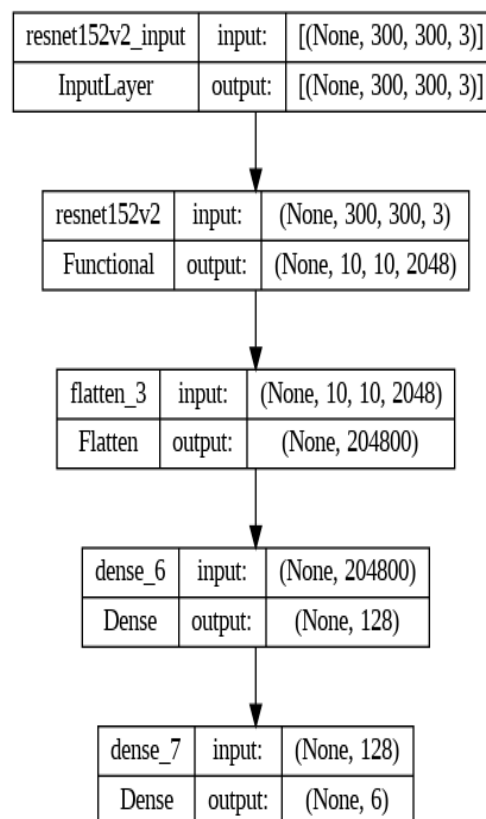
Pengujian pada prosedur ini dilakukan dengan cara berbagai tahapan. Untuk tahapan yang pertama pengujian akan dilakukan pada augmentasi data. Dimana augmentasi data dipastikan bisa digunakan atau proses dapat berjalan sesuai dengan inputan sebelumnya. Berikut merupakan hasil dari pengujian augmentasi data bahwa augmentasi data dapat berjalan :



**Gambar 2.1.3.1** Hasil dari Visualisasi Augmentasi data dengan Horizontal

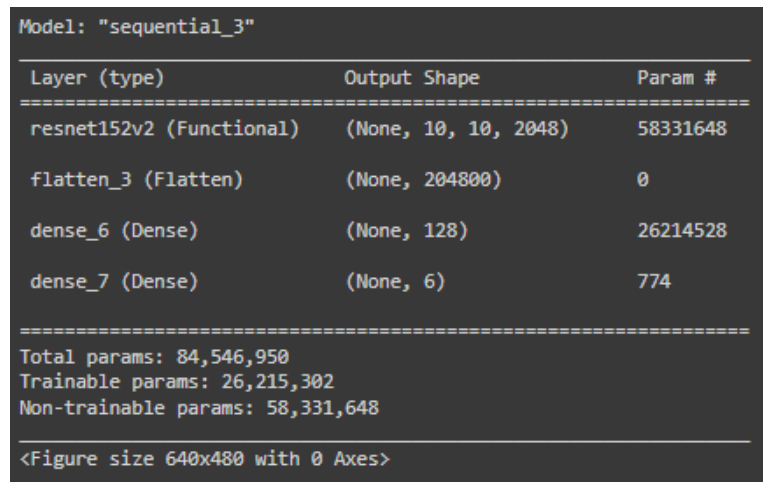
Pada gambar 2.1.3.1 dapat dijelaskan bahwa *pre-processing* dengan pemanfaatan augmentasi data yang berupa *horizontal flip*, hasil yang didapatkan berupa dataset yang dibalik secara *horizontal* ke arah kanan dan kiri dengan sudut 180 derajat.

Tahapan selanjutnya, untuk pengujian bagian ekstraksi fitur. Pada pengujian ini diawali dengan memastikan agar model yang sudah diimplementasikan ke dalam program sudah berjalan dengan baik. Berikut adalah hasil dari model ekstraksi fitur yang sudah dibuat.



**Gambar 2.1.3.2** Hasil dari model ekstraksi fitur dengan ResNet152 V2

Dari gambar 2.1.3.2 sudah terlihat bahwa *output* yang dihasilkan sudah sesuai dengan *input shape* citra dan parameter lainnya yang disertakan dalam program seperti jumlah unit pada *layer dense*. Kemudian pengujian ekstraksi fitur dilanjutkan dengan menampilkan rangkuman dari model yang dibuat dengan menggunakan program `model.summary()`. Berikut adalah hasil rangkuman dari model yang dibuat.



```

Model: "sequential_3"
_____
Layer (type)                 Output Shape              Param #
-----
resnet152v2 (Functional)     (None, 10, 10, 2048)     58331648
flatten_3 (Flatten)          (None, 204800)            0
dense_6 (Dense)               (None, 128)               26214528
dense_7 (Dense)              (None, 6)                  774
_____
Total params: 84,546,950
Trainable params: 26,215,302
Non-trainable params: 58,331,648
<Figure size 640x480 with 0 Axes>

```

**Gambar 2.1.3.3** Hasil rangkuman model ekstraksi fitur dengan ResNet152 V2

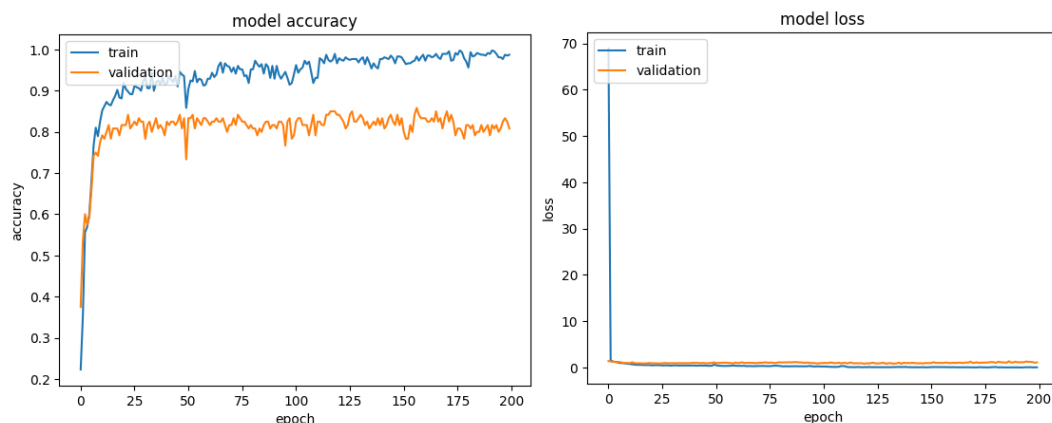
Pada gambar 2.1.3.3 terlihat bahwa hasil dari model dan *layer* yang ditampilkan sesuai dengan bagan model yang sebelumnya sudah ditampilkan pada gambar 2.1.3.2. Namun selain menampilkan model dan *layer*, pada gambar tersebut juga ditampilkan nilai *params* yang menunjukkan jumlah parameter dalam model tersebut. Dari total nilai *params* yang ditampilkan, di bawahnya dibagi lagi menjadi *trainable params* dan *non-trainable params*. *Trainable params* menunjukkan jumlah parameter yang dapat dilatih dalam model dan nilainya akan diperbarui selama proses pelatihan. Sedangkan *non-trainable params* menunjukkan jumlah parameter yang tidak dapat dilatih dalam model tersebut yang mencakup parameter dari layer ResNet152 V2 yang sudah dilatih dengan bobot pre-trained dari ImageNet.

Tahapan selanjutnya, untuk pengujian bagian Klasifikasi.

```
Epoch 199/200
15/15 [=====] - ETA: 0s - loss: 0.0486 - accuracy: 0.9854
Epoch 199: val_accuracy did not improve from 0.85833
15/15 [=====] - 18s 1s/step - loss: 0.0486 - accuracy: 0.9854 - val_loss: 1.0336 - val_accuracy: 0.8250
Epoch 200/200
15/15 [=====] - ETA: 0s - loss: 0.0313 - accuracy: 0.9875
Epoch 200: val_accuracy did not improve from 0.85833
15/15 [=====] - 18s 1s/step - loss: 0.0313 - accuracy: 0.9875 - val_loss: 1.1188 - val_accuracy: 0.8083
15/15 [=====] - 17s 1s/step - loss: 0.0470 - accuracy: 0.9896
Train Score: [0.04701627418398857, 0.9895833134651184]
4/4 [=====] - 2s 490ms/step - loss: 1.1188 - accuracy: 0.8083
Val Score: [1.1188181638717651, 0.8083333373069763]
2/2 [=====] - 2s 639ms/step - loss: 1.1940 - accuracy: 0.8167
Test Score: [1.1939619779586792, 0.8166666626930237]
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

**Gambar 2.1.3.4** Hasil program untuk proses pelatihan dan evaluasi model

Pada gambar 2.1.3.4 dapat terlihat bahwa hasil selama proses *training* dan evaluasi model memberikan gambaran dari setiap skor performa data *train*, data *validation*, dan data *testing*. Tingkat akurasi dan performa model dapat diprediksi melalui hasil dari program ini serta dapat menguji kemampuan model dalam memahami sejauh mana pembelajaran telah dilatih.



**Gambar 2.1.3.5** Hasil visualisasi dari akurasi dan loss yang telah dilatih

Pada gambar 2.1.3.5 menunjukkan visualisasi dari akurasi dan *loss* yang telah melalui proses *training* atau pelatihan. Terdapat grafik akurasi dan grafik *loss* yang memiliki dua kurva untuk merepresentasikan data *training* dan juga data *validation*. Grafik yang ditampilkan dapat dianalisis untuk mengevaluasi performa model yang dihasilkan.

## 2.2 Mobile Application BatiQu

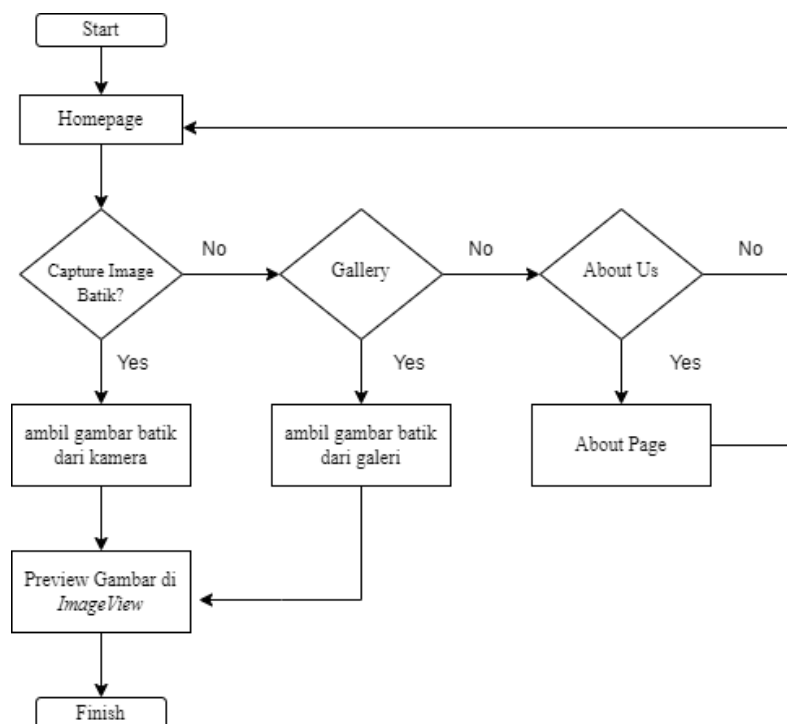
### 2.2.1 Cara Kerja Mobile Application

Aplikasi Mobile berfungsi sebagai menampilkan hasil klasifikasi jenis batik kepada pengguna. Tampilan pada aplikasi sendiri terdiri dari beberapa elemen seperti tombol, gambar dan teks. Tampilan elemen yang digunakan ditulis dalam format XML (*eXtensible Markup Language*). Fungsionalitas dari setiap elemen



menggunakan bahasa pemrograman Kotlin yang dirancang menggunakan Android Studio IDE (*Integrated Development Environment*).

Setiap tombol pada halaman awal aplikasi memiliki fungsi masing masing, seperti contoh pada tombol Mulai yang berarti aplikasi tersebut dapat berjalan jika menekan tombol tersebut. Pada tampilan klasifikasi gambar, pengguna dapat mencoba klasifikasi jenis batik dengan tombol kamera yang berfungsi untuk memotret gambar batik yang dimiliki atau tombol galeri jika ingin *upload* gambar melalui galeri. Elemen – elemen dari pembuatan aplikasi ini dipersiapkan pada proses pembuatan UI (*User Interface*) dengan menggunakan aplikasi Figma dan Canva dan tiap elemen diekspor dengan format .png. Elemen yang telah di ekspor kemudian dimasukkan ke dalam Android Studio sebagai *resource* penampilan *Image View* pada tiap halaman aplikasi. Cara kerja *Mobile Application* seperti gambar di bawah ini



**Gambar 2.2.1.1** Flowchart dari cara kerja *Mobile Application*

## 2.2.2 Implementasi

### 2.2.2.1 Kamera dan Galeri

Kamera pada aplikasi ini berfungsi sebagai pengambilan gambar batik melalui kamera *handphone* user untuk selanjutnya di proses menggunakan *Machine Learning*. Setelah mengambil gambar dari kamera, user bisa melihat hasil gambar

nya melalui *Image View* dan jika merasa gambar itu kurang sesuai bisa mengambil gambar ulang.

Fitur galeri pada aplikasi ini berfungsi sebagai pengambilan gambar batik melalui galeri *handphone* untuk selanjutnya di proses menggunakan Machine Learning. User bisa menggunakan gambar batik yang tersedia di masing – masing galeri *handphone* user untuk di masukkan ke dalam *Image View*.

#### 2.2.2.2 About Us

Fitur About Us menampilkan fungsi singkat dari aplikasi ini serta data diri dari pembuat aplikasi ini sebagai bentuk *copyright* dari pembuat aplikasi.

### 2.2.3 Pengujian

#### 2.2.3.1 Kamera dan Galeri

Pengujian dilakukan dengan melihat apakah gambar yang diambil baik dari kamera atau dari galeri dapat terlihat di *Image View* dan gambar yang di *Image View* sesuai dengan yang diambil

#### 2.2.3.2 About Us

Pengujian pada fitur ini dengan melihat dari tampilan antarmuka yang dibuat dari file XML apakah dapat terlihat.

### 2.2.4 Analisis Pengerjaan Implementasi Sistem

Implementasi cara kerja sub sistem ini adalah gambar dapat di ambil dari kamera atau galeri dan gambar tersebut dapat dilihat di *Image View* dan sesuai dengan gambar yang diambil baik dari kamera maupun dari galeri.



### 3 Analisis Pengerjaan Implementasi Sistem

Pada dokumen proposal *Capstone Design* sebelumnya sudah dilampirkan tabel rencana pengerjaan *capstone* seperti yang terlihat pada tabel 3.1 berikut ini:

Kegiatan	Progress	2022				2023						
		September	Oktober	November	Desember	Januari	Februari	Maret	April	Mei	Juni	Juli
Fase1												
Studi Literature	100%											
Pemilihan Arsitektur	100%											
Pengambilan Data	80%											
Perancangan Spesifikasi & Verifikasi	100%											
Fase2												
Perancangan Program Sistem	0%											
Uji Coba Sistem Klasifikasi	0%											
Pembuatan Aplikasi	0%											
Uji Coba Aplikasi	0%											
Analisa dan Penulisan	0%											
Sidang Tugas Akhir	0%											

**Tabel 2.2.1** Rencana pengerjaan *Capstone Design*

Dari tabel tersebut terlihat bahwa sebelumnya fase 1 sudah selesai yang ditandai dengan warna merah dan ditutup dengan mengunggah dokumen proposal *Capstone Design*. Saat ini pengerjaan sudah masuk ke dalam fase 2 yang ditandai dengan warna kuning. Selanjutnya akan ditampilkan tabel hasil implementasi pengerjaan *Capstone Design* sebagai berikut:

Kegiatan	Progress	2022		2023							
		November	Desember	Januari	Februari	Maret	April	Mei	Juni	Juli	
Fase 1											
Pengambilan Data	100%										
Fase 2											
Perancangan Program Sistem	100%										
Uji Coba Sistem Klasifikasi	85%										
Pembuatan Aplikasi	50%										
Uji Coba Aplikasi	50%										
Analisa dan Penulisan	50%										
Sidang Tugas Akhir	0%										

**Tabel 2.2.2** Hasil implementasi pengerjaan fase 2 *Capstone Design*

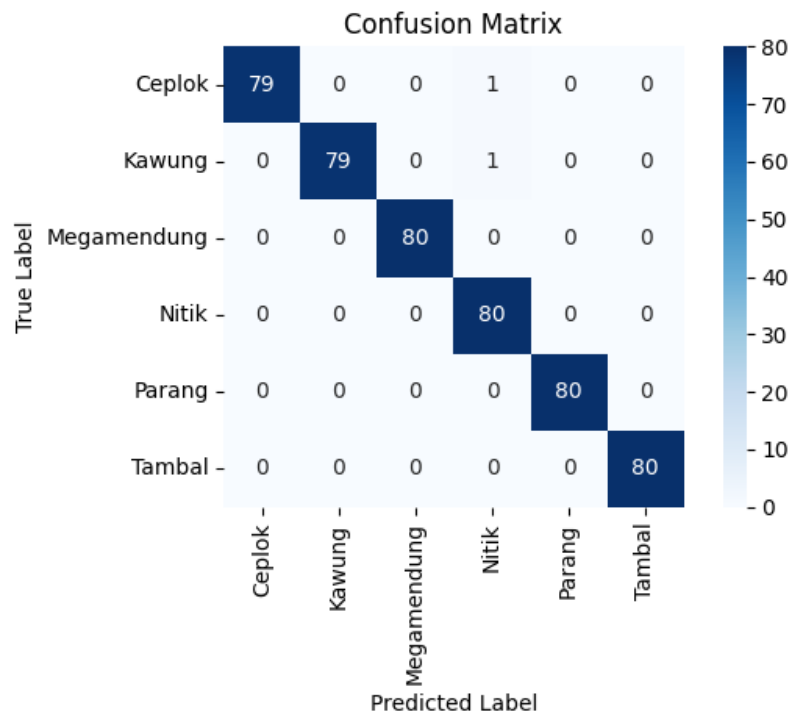
Pada tabel 3.2 terlihat progress pengerjaan dari *Capstone Design* yang tercatat hingga bulan Juni 2023. Pada tabel 3.2 terlihat cukup banyak perbedaan dibandingkan dengan tabel 3.1 yang sebelumnya sudah ditampilkan. Pada bagian uji coba sistem klasifikasi yang awalnya direncanakan selesai bulan februari harus ditambah hingga bulan Juni karena terdapat kesulitan dalam memperoleh hasil klasifikasi yang baik dari beberapa metode klasifikasi yang sudah dicoba. Selain itu sumber dataset dengan kualitas baik yang terbatas juga cukup mempersulit pengerjaan sehingga pengumpulan dataset baru bisa diselesaikan pada bulan Maret

2023. Penambahan waktu dari uji coba sistem klasifikasi ini berdampak pada penambahan waktu juga pada pembuatan aplikasi dan uji coba aplikasi.

## 4 Hasil Akhir

### 4.1 Machine Learning

#### 4.1.1 Tampilan Tabel Confusion Matrix dari Program Klasifikasi



**Tabel 4.1.1** *Confusion Matrix* dengan metode ResNet152 V2

Pada gambar 4.1.1 merupakan tampilan table dari hasil confusion matrix yang menggunakan metode Resnet152 V2. *Confusion Matrix* merupakan parameter yang digunakan untuk menggambarkan dan mengevaluasi kualitas dan performa dari model. Hasil diatas dapat dijelaskan bahwa *confusion matrix* dapat melakukan pembelajaran model yang sudah di-*inputkan* sebelumnya dengan cukup baik yang menandakan pembelajaran model dapat dilakukan untuk pengklasifikasi.

#### 4.1.2 Hasil Classification Report dari Program Klasifikasi

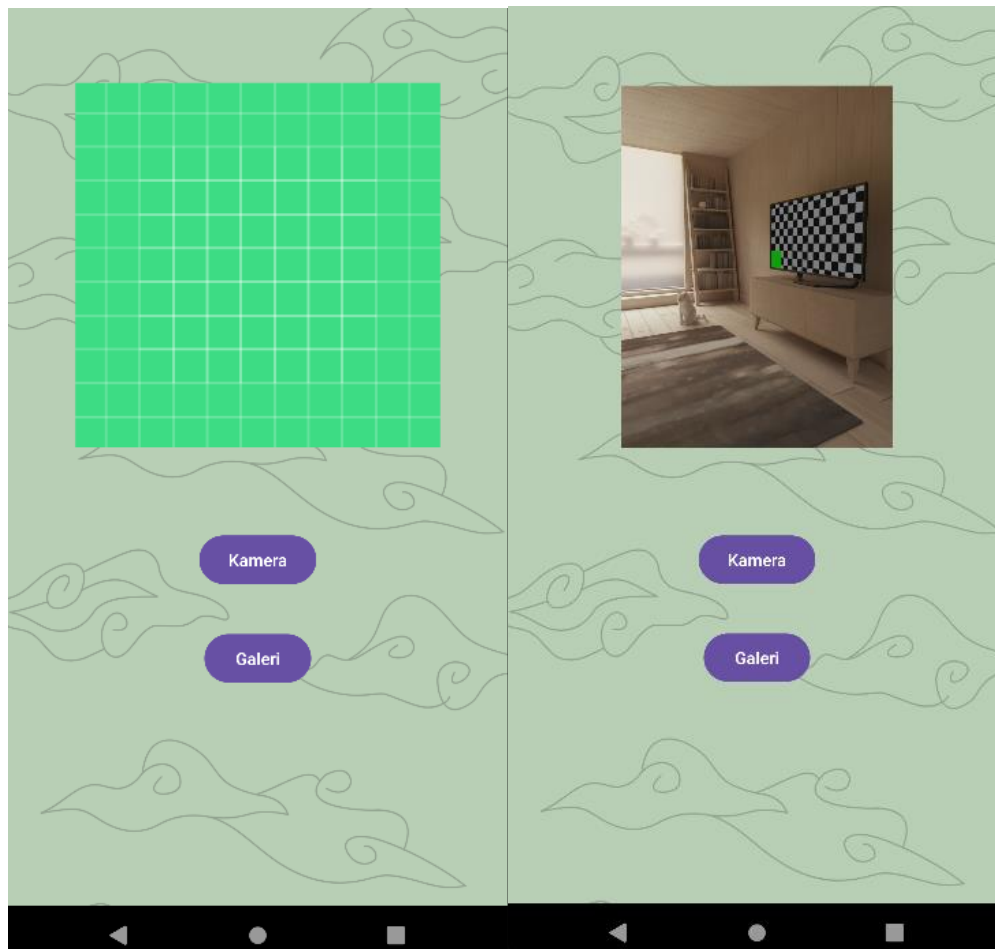
	precision	recall	f1-score	support
Ceplok	0.99	1.00	0.99	80
Kawung	1.00	0.97	0.99	80
Megamendung	1.00	1.00	1.00	80
Nitik	0.96	0.97	0.97	80
Parang	0.99	0.99	0.99	80
Tambal	0.99	0.99	0.99	80
accuracy			0.99	480
macro avg	0.99	0.99	0.99	480
weighted avg	0.99	0.99	0.99	480

**Gambar 4.1.2.** *Classification Report* dari metode ResNet152 V2

Penjelasan gambar 4.1.2 diatas masih sangat bersangkutan dengan Confusion Matrix, dimana dengan adanya konteks diatas dapat menganalisis hasil presisi, *recall* dan *F1-Score*. Report Klasifikasi berhasil dijalankan berdasarkan perhitungan dari hasil prediksi model yang telah dilatih sebelumnya. Presisi merupakan parameter untuk mengukur akurasi atau nilai ketepatan model yang telah dibuat. Presisi memberikan hasil berbentuk persen dengan mengetahui prediksi yang benar. Dapat dikatakan juga bahwa Report Klasifikasi bisa berjalan sesuai dengan pelatihan model. *Recall* digunakan untuk mengidentifikasi nilai yang dimana semakin tinggi nilai *recall* maka menghasilkan nilai tinggi pada nilai prediksi yang terdeteksi. *F1-Score* sendiri merupakan nilai rata-rata dari presisi dan juga *recall*.

## 4.2 Mobile Application

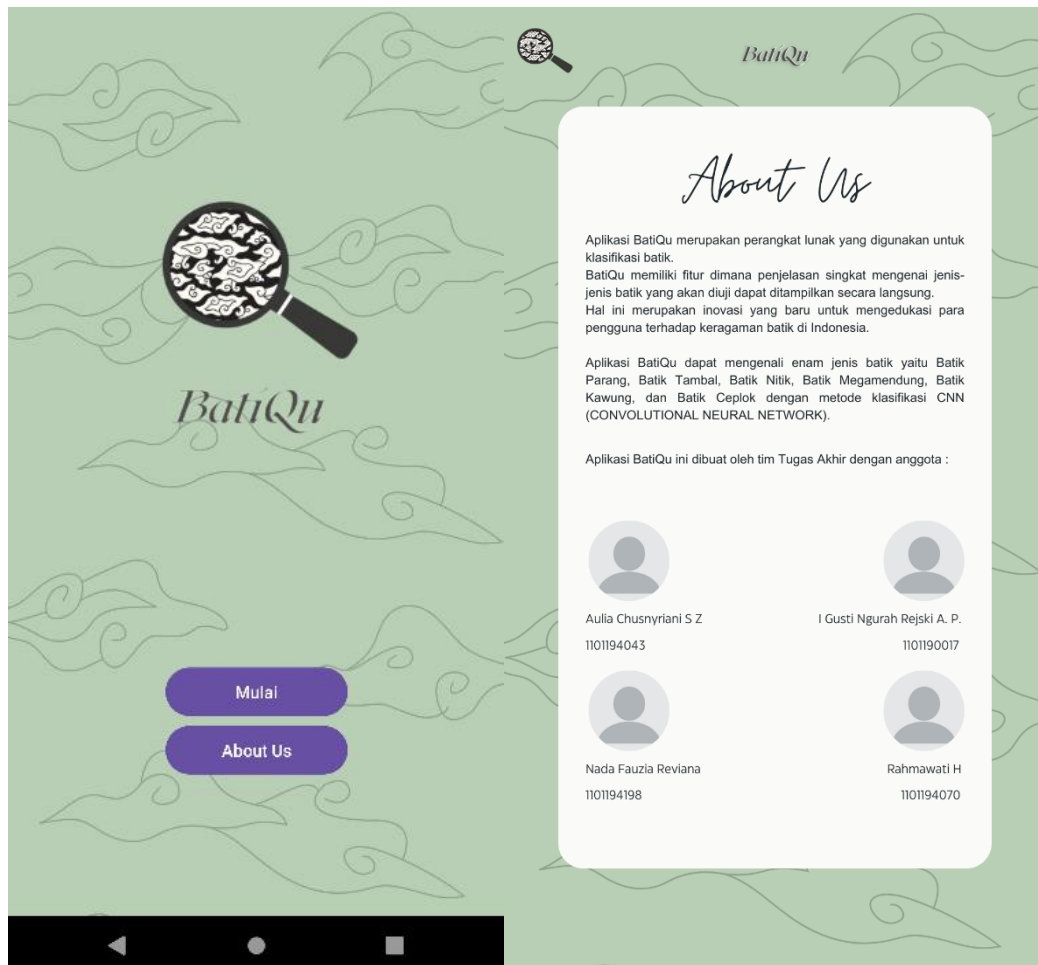
### 4.2.1 Tampilan AntarMuka Kamera dan Galeri Aplikasi



**Gambar 4.2.1** Tampilan AntarMuka Kamera dan Galeri Aplikasi

Pada gambar di atas merupakan tampilan dari fitur Utama aplikasi yaitu fitur kamera dan galeri. Untuk menggunakan fitur ini, user bisa menggunakan tombol kamera atau galeri sebagai pengambilan gambar batik. Setelah berhasil mengambil gambar, user dapat melihat hasil gambar tersebut di Image View yang ada di atas.

#### 4.2.2 Tampilan AntarMuka About Us pada Aplikasi



**Gambar 4.2.2** Tampilan AntarMuka About Us pada Aplikasi

Pada gambar di atas merupakan tampilan dari fitur tambahan aplikasi yaitu fitur *about us*. User bisa melihat fitur ini berada di halaman depan aplikasi dan untuk mencoba fitur ini cukup menekan tombol About Us dan akan muncul halaman khusus yang berisikan informasi aplikasi singkat beserta data diri pembuat aplikasi.

## 5 Lampiran

### 5.1 Lampiran 1

Source Code *activity\_main.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
```

```

android:layout_height="match_parent"
android:background="@drawable/background"
tools:context=".MainActivity">

<ImageView
    android:id="@+id/imageView3"
    android:layout_width="350dp"
    android:layout_height="350dp"
    android:layout_marginTop="100dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:srcCompat="@drawable/batik3" />

<Button
    android:id="@+id/start"
    android:layout_width="150dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="90dp"
    android:text="Mulai"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/imageView3" />

<Button
    android:id="@+id/about"
    android:layout_width="150dp"
    android:layout_height="wrap_content"
    android:text="About"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/start" />

</androidx.constraintlayout.widget.ConstraintLayout>

```

Source Code activity\_camera.xml

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/background"
    tools:context=".Camera">
    <ImageView

```



```

android:id="@+id/imageView"
android:layout_width="300dp"
android:layout_height="300dp"
android:layout_marginTop="20dp"
android:layout_marginBottom="17dp"
app:layout_constraintBottom_toTopOf="@id/button"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent"
app:srcCompat="@drawable/ic_launcher_background"
/>

```

#### <Button

```

android:id="@+id/button"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginBottom="210dp"
android:text="Kamera"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@id/imageView"
tools:layout_editor_absoluteY="334dp"
/>

```

#### <Button

```

android:id="@+id/gallery"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginBottom="180dp"
android:text="Galeri"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@id/imageView"
app:layout_constraintVertical_bias="1.0"
/>

```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

#### Source Code activity\_about.xml

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:background="@drawable/about"
tools:context=".About">

```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

## 5.2 Lampiran 2

### Source Code MainActivity.kt

```
package com.example.klasifikasibatik

import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Button

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        val button = findViewById<Button>(R.id.start)
        val button2 = findViewById<Button>(R.id.about)

        button.setOnClickListener {
            startActivity(Intent(this@MainActivity, Camera::class.java))
            finish()
        }
        button2.setOnClickListener {
            startActivity(Intent(this@MainActivity, About::class.java))
            finish()
        }
    }
}
```

### Source Code Camera.kt

```
package com.example.klasifikasibatik

import android.Manifest
import android.content.Intent
import android.content.pm.PackageManager
import android.graphics.Bitmap
import android.net.Uri
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.provider.MediaStore
import android.widget.Button
import android.widget.ImageView
import androidx.core.app.ActivityCompat

class Camera : AppCompatActivity() {
```



```

lateinit var button: Button
private val pickImage = 100
private var imageUri: Uri? = null

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_camera)
    val button = findViewById<Button>(R.id.button)
    val button2 = findViewById<Button>(R.id.gallery)

    button2.setOnClickListener {
        val gallery = Intent(Intent.ACTION_PICK,
            MediaStore.Images.Media.INTERNAL_CONTENT_URI)
        startActivityForResult(gallery, pickImage)
    }

    button.isEnabled = false

    if (ActivityCompat.checkSelfPermission(
        this,
        Manifest.permission.CAMERA
    ) != PackageManager.PERMISSION_GRANTED) {
        ActivityCompat.requestPermissions(this,
            arrayOf(Manifest.permission.CAMERA), 111)
    } else {
        button.isEnabled = true
        button.setOnClickListener {
            var i = Intent(MediaStore.ACTION_IMAGE_CAPTURE)
            startActivityForResult(i, 101)
        }
    }

    override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
        val imageView = findViewById<ImageView>(R.id.imageView)
        super.onActivityResult(requestCode, resultCode, data)
        if(requestCode==101){
            var pic = data?.getParcelableExtra<Bitmap>("data")
            imageView.setImageBitmap(pic)
        }
        if (resultCode == RESULT_OK && requestCode == pickImage) {
            imageUri = data?.data
            imageView.setImageURI(imageUri)
        }
    }
}

```

```

override fun onRequestPermissionsResult(
    requestCode: Int,
    permissions: Array<out String>,
    grantResults: IntArray
) {
    val button = findViewById<Button>(R.id.button)
    super.onRequestPermissionsResult(requestCode, permissions,
grantResults)
    if(requestCode == 111 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED)
    {
        button.isEnabled = true
    }
}

```

*Source Code About.kt*

```

package com.example.klasifikasibatik

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle

class About : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_about)
    }
}

```

### 5.3 Lampiran 3

*Source Code AndroidManifest.xml*

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission
android:name="android.permission.READ_EXTERNAL_STORAGE" />

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"

```

```

        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportRtl="true"
        android:theme="@style/Theme.KlasifikasiBatik"
        tools:targetApi="31">
        <activity
            android:name=".About"
            android:exported="false" />
        <activity
            android:name=".Camera"
            android:exported="false" />
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>

```

#### 5.4 Lampiran 4

Tabel rincian biaya keperluan *Capstone Design*

No.	Keperluan	Jumlah	Biaya	Total
1	Google Colab Premium	1	Rp 170.0000	Rp 170.0000
2	Box Kayu	1	Rp 450.000	Rp 450.000
3	LED 12V 6 Mata	6	Rp 5.000	Rp 30.000
4	Kabel tembaga kecil merah+hita	1	Rp 20.000	Rp 20.000
5	Saklar plastik kecil	1	Rp 3.000	Rp 3.000
Total :				Rp 673.000