

**TUGAS PROYEK PENGENALAN POLA**  
**LAPORAN AKHIR PROYEK**

Dosen Pengampu:

**Imam Much Ibnu Subroto, ST., M.Sc., Ph.D**



Disusun oleh:

**KELOMPOK 9**

<b>Muhammad Fikri Maulana</b>	<b>(32602100076)</b>
<b>Muhammad Hibatullah Arkaan</b>	<b>(32602100078)</b>
<b>Muhammad Iko Wardhana</b>	<b>(32602100079)</b>
<b>Muhammad Maulana Fathul Muin</b>	<b>(32602100082)</b>

**PRODI TEKNIK INFORMATIKA**  
**FAKULTAS TEKNOLOGI INDUSTRI**  
**UNIVERSITAS ISLAM SULTAN AGUNG SEMARANG**  
**2024**

## **A. Collection Data**

Dataset Aksara Jawa merupakan kumpulan gambar yang mewakili aksara-aksara Jawa. Dataset ini penting untuk melatih model machine learning yang dapat mengklasifikasikan aksara Jawa. Preprocessing data merupakan langkah penting sebelum melatih model, untuk memastikan data berkualitas baik dan siap untuk digunakan.

Berikut sumber Dataset dan Hasil Pengerjaannya :

GitHub :

- <https://github.com/vzrengamani/aksarajawa-hanacaraka>

Kaggle:

- <https://www.kaggle.com/code/mpwolke/aksara-jawa-javanese-script-images/input>

Dataset Gdrive:

- <https://drive.google.com/drive/folders/1J1fxbcAKZsZxZWu66o7BTRa36G9Hz4r?usp=sharing>

Google Colab:

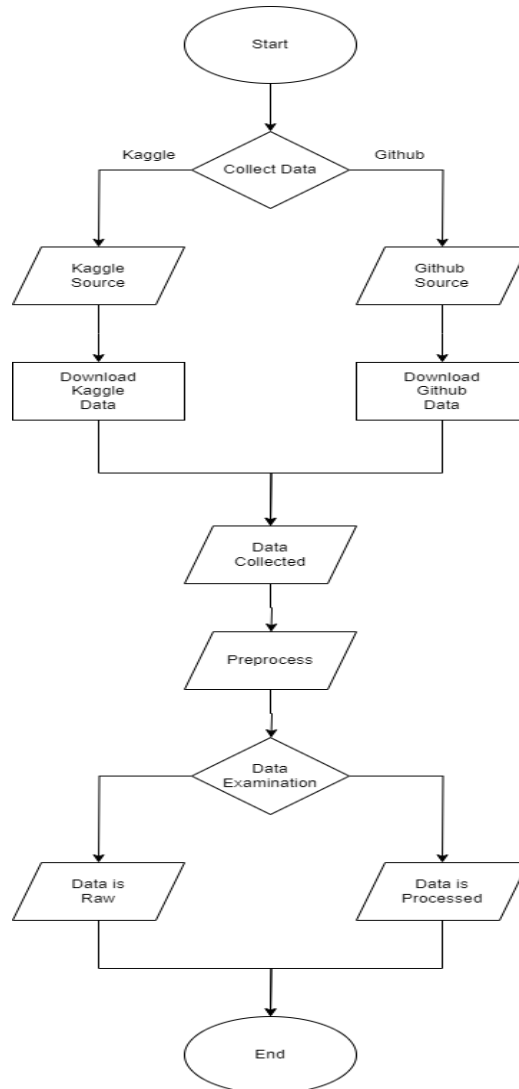
- [https://colab.research.google.com/drive/1vbcoNVOj3ig6id1VXM0udyKjQheD1Y7C?oid=117907131505526421147&usp=drive\\_link](https://colab.research.google.com/drive/1vbcoNVOj3ig6id1VXM0udyKjQheD1Y7C?oid=117907131505526421147&usp=drive_link)

Link Gdrive:

- <https://drive.google.com/drive/folders/1J1fxbcAKZsZxZWu66o7BTRa36G9Hz4r?usp=sharing>

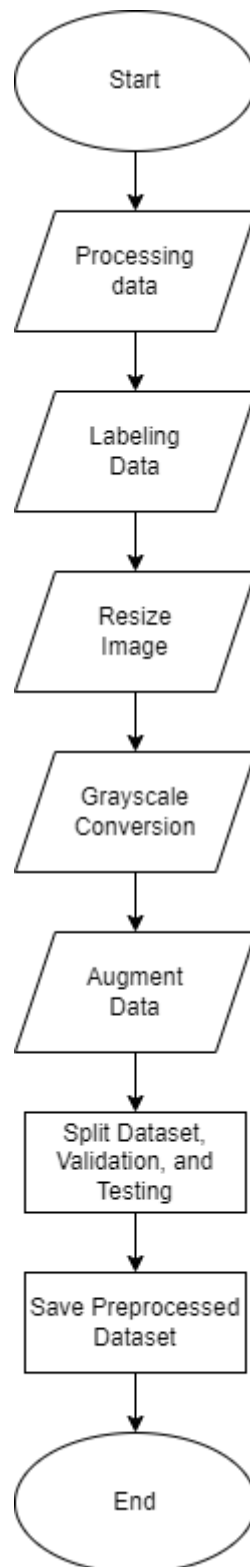
## B. Flowchart Model

### a. Flowchart Collection Data



Gambar 1. 1 Gambar Flowchart Collection Data

**b. Flowchart Preprocessing Data**



Gambar 1. 2 Gambar *Preprocessing Data*

### **C. *Preprocessing Data***

#### **a) Pelabelan Data**

Setiap gambar yang dikumpulkan dari sumber-sumber tersebut sudah dilabeli dengan benar. Label-label ini menunjukkan karakter Aksara Jawa yang ada pada setiap gambar. Langkah ini penting untuk memastikan data yang digunakan dalam pelatihan model adalah data yang terstruktur dengan baik.

#### **b) *Classification***

Data yang telah dilabeli akan digunakan untuk melatih model klasifikasi karakter Aksara Jawa. Model ini akan dapat mengenali dan mengklasifikasikan karakter-karakter Aksara Jawa dari gambar input.

#### **c) *Resizing Gambar***

Semua gambar akan diubah ukurannya menjadi dimensi yang seragam, misalnya 64x64 piksel. Ini dilakukan untuk memastikan semua input ke dalam model memiliki ukuran yang sama, yang akan membantu dalam proses pelatihan model.

#### **d) *Grayscale Conversion***

Konversi gambar ke format grayscale akan dilakukan untuk mengurangi kompleksitas data. Langkah ini juga membantu dalam meningkatkan efisiensi komputasi.

#### **e) *Augmentasi Data***

Data yang sudah di-preprocessing akan diperkaya melalui augmentasi seperti rotasi, pergeseran, dan pembesaran gambar. Ini akan membantu dalam meningkatkan variasi dan kualitas dataset.

#### **f) *Pembagian Dataset***

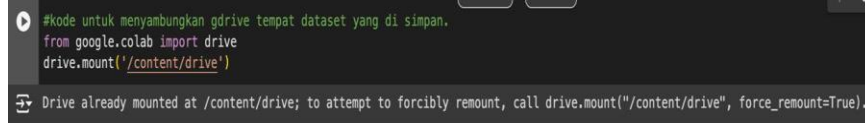
Dataset akan dibagi menjadi tiga bagian utama: set pelatihan, set validasi, dan set pengujian. Pembagian ini penting untuk mengevaluasi kinerja model dengan benar.

#### **g) *Penyimpanan Dataset***

Dataset yang sudah dipreproses akan disimpan dalam format yang sesuai untuk digunakan dalam proses pelatihan dan pengujian model.

## D. Langkah-langkah Preprocessing Data

### a) Menyiapkan Kode untuk import Dataset dari Drive



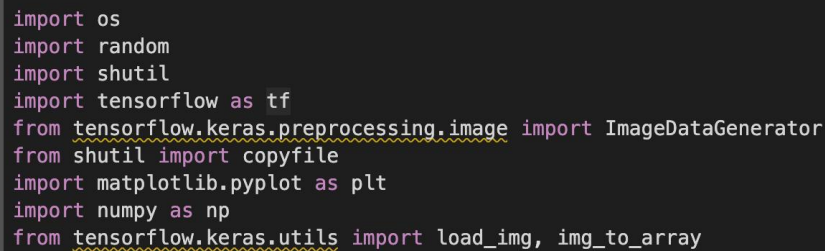
```
#kode untuk menyambungkan gdrive tempat dataset yang di simpan.  
from google.colab import drive  
drive.mount('/content/drive')
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force\_remount=True).

Gambar 1. 3 Gambar Import Dataset

Kode di atas berfungsi untuk mengimpor dataset dari Google Drive ke Google Collab

### b) Import Library



```
import os  
import random  
import shutil  
import tensorflow as tf  
from tensorflow.keras.preprocessing.image import ImageDataGenerator  
from shutil import copyfile  
import matplotlib.pyplot as plt  
import numpy as np  
from tensorflow.keras.utils import load_img, img_to_array
```

Gambar 1. 4 Gambar Import Library

Kode di atas adalah mengimpor library di Python, import library sendiri adalah proses memasukkan modul atau paket eksternal ke dalam program Anda agar Anda bisa menggunakan fungsionalitas yang ada dalam library tersebut.

### c) Mengecek dataset yang ada

```
# Define source paths
source_path = '/content/drive/MyDrive/dataset'
source_path_ba = os.path.join(source_path, 'ba')
source_path_ca = os.path.join(source_path, 'ca')
source_path_da = os.path.join(source_path, 'da')
source_path_dha = os.path.join(source_path, 'dha')
source_path_ga = os.path.join(source_path, 'ga')
source_path_ha = os.path.join(source_path, 'ha')
source_path_ja = os.path.join(source_path, 'ja')
source_path_ka = os.path.join(source_path, 'ka')
source_path_la = os.path.join(source_path, 'la')
source_path_ma = os.path.join(source_path, 'ma')
source_path_na = os.path.join(source_path, 'na')
source_path_nga = os.path.join(source_path, 'nga')
source_path_nya = os.path.join(source_path, 'nya')
source_path_pa = os.path.join(source_path, 'pa')
source_path_ra = os.path.join(source_path, 'ra')
source_path_sa = os.path.join(source_path, 'sa')
source_path_ta = os.path.join(source_path, 'ta')
source_path_tha = os.path.join(source_path, 'tha')
source_path_wa = os.path.join(source_path, 'wa')
source_path_ya = os.path.join(source_path, 'ya')

# Print number of images in each category
print(f"There are {len(os.listdir(source_path_ba))} images of ba.")
print(f"There are {len(os.listdir(source_path_ca))} images of ca.")
print(f"There are {len(os.listdir(source_path_da))} images of da.")
print(f"There are {len(os.listdir(source_path_dha))} images of dha.")
print(f"There are {len(os.listdir(source_path_ga))} images of ga.")
print(f"There are {len(os.listdir(source_path_ha))} images of ha.")
print(f"There are {len(os.listdir(source_path_ja))} images of ja.")
print(f"There are {len(os.listdir(source_path_ka))} images of ka.")
print(f"There are {len(os.listdir(source_path_la))} images of la.")
print(f"There are {len(os.listdir(source_path_ma))} images of ma.")
print(f"There are {len(os.listdir(source_path_na))} images of na.")
print(f"There are {len(os.listdir(source_path_nga))} images of nga.")
print(f"There are {len(os.listdir(source_path_nya))} images of nya.")
print(f"There are {len(os.listdir(source_path_pa))} images of pa.")
print(f"There are {len(os.listdir(source_path_ra))} images of ra.")
print(f"There are {len(os.listdir(source_path_sa))} images of sa.")
print(f"There are {len(os.listdir(source_path_ta))} images of ta.")
print(f"There are {len(os.listdir(source_path_tha))} images of tha.")
print(f"There are {len(os.listdir(source_path_wa))} images of wa.")
print(f"There are {len(os.listdir(source_path_ya))} images of ya.")
```

Gambar 1. 5 Gambar mengecek dataset

Kode diatas adalah untuk mengecek data yang ada dalam dataset, diatas mengimport data dari google drive ke google collab dengan alamat ‘/content/drive/MyDrive/dataset’

**d) Hasil pengecekan Dataset Mengecek dataset yang ada**

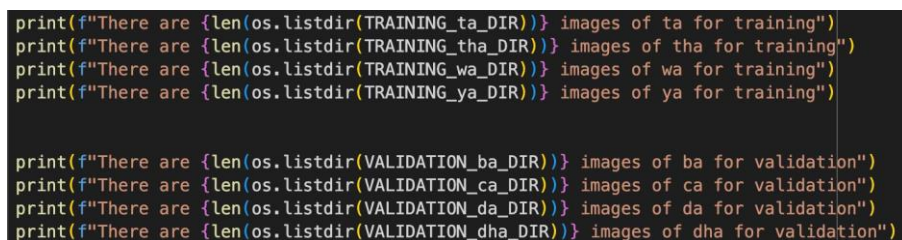


```
➡ There are 114 images of ba.  
There are 108 images of ca.  
There are 108 images of da.  
There are 108 images of dha.  
There are 108 images of ga.  
There are 102 images of ha.  
There are 108 images of ja.  
There are 108 images of ka.  
There are 108 images of la.  
There are 108 images of ma.  
There are 108 images of na.  
There are 102 images of nga.  
There are 108 images of nya.  
There are 108 images of pa.  
There are 108 images of ra.  
There are 108 images of sa.  
There are 108 images of ta.  
There are 108 images of tha.  
There are 108 images of wa.  
There are 108 images of ya.
```

Gambar 1. 6 Gambar Hasil cek Dataset

Kode di atas adalah hasil dari pengecekan data pada gambar sebelumnya, data diatas menampilkan jumlah gambar pada data Ha, Na, Ca, Ra, Ka.

**e) Penghitungan Jumlah Gambar dalam Dataset Pelatihan dan Validasi**



```
print(f"There are {len(os.listdir(TRAINING_ta_DIR))} images of ta for training")  
print(f"There are {len(os.listdir(TRAINING_tha_DIR))} images of tha for training")  
print(f"There are {len(os.listdir(TRAINING_wa_DIR))} images of wa for training")  
print(f"There are {len(os.listdir(TRAINING_ya_DIR))} images of ya for training")  
  
print(f"There are {len(os.listdir(VALIDATION_ba_DIR))} images of ba for validation")  
print(f"There are {len(os.listdir(VALIDATION_ca_DIR))} images of ca for validation")  
print(f"There are {len(os.listdir(VALIDATION_da_DIR))} images of da for validation")  
print(f"There are {len(os.listdir(VALIDATION_dha_DIR))} images of dha for validation")
```

Gambar 1. 7 Gambar Menghitung jumlah Gambar

Kode ini menampilkan jumlah gambar untuk berbagai kategori aksara (seperti ta, tha, wa, ya) dalam direktori pelatihan dan validasi. Fungsi `os.listdir()` digunakan untuk menghitung file dalam setiap direktori, memberikan gambaran distribusi data antar kategori dan set.



#### f) Struktur Direktori untuk Dataset

```
# Membuat Struktur Direktori Training dan Validation untuk Klasifikasi Gambar
root_dir = '/content/drive/MyDrive/Untitled Folder'

def create_train_val_dirs(root_path):
    os.makedirs(os.path.join(root_path, 'training'))
    os.makedirs(os.path.join(f'{root_path}/training', 'ba'))
    os.makedirs(os.path.join(f'{root_path}/training', 'ca'))
    os.makedirs(os.path.join(f'{root_path}/training', 'da'))
    os.makedirs(os.path.join(f'{root_path}/training', 'dha'))
    os.makedirs(os.path.join(f'{root_path}/training', 'ga'))
    os.makedirs(os.path.join(f'{root_path}/training', 'ha'))
    os.makedirs(os.path.join(f'{root_path}/training', 'ja'))
    os.makedirs(os.path.join(f'{root_path}/training', 'ka'))
    os.makedirs(os.path.join(f'{root_path}/training', 'la'))
    os.makedirs(os.path.join(f'{root_path}/training', 'ma'))
    os.makedirs(os.path.join(f'{root_path}/training', 'na'))
    os.makedirs(os.path.join(f'{root_path}/training', 'nga'))
    os.makedirs(os.path.join(f'{root_path}/training', 'nya'))
    os.makedirs(os.path.join(f'{root_path}/training', 'pa'))
    os.makedirs(os.path.join(f'{root_path}/training', 'ra'))
    os.makedirs(os.path.join(f'{root_path}/training', 'sa'))
    os.makedirs(os.path.join(f'{root_path}/training', 'ta'))
    os.makedirs(os.path.join(f'{root_path}/training', 'tha'))
    os.makedirs(os.path.join(f'{root_path}/training', 'wa'))
    os.makedirs(os.path.join(f'{root_path}/training', 'ya'))

    os.makedirs(os.path.join(root_path, 'validation'))
    os.makedirs(os.path.join(f'{root_path}/validation', 'ba'))
    os.makedirs(os.path.join(f'{root_path}/validation', 'ca'))
```

Gambar 1. 8 Gambar Struktur Direktori Dataset

Kode di atas membuat struktur direktori untuk dataset training dan validation dalam klasifikasi gambar menggunakan fungsi `makedirs` dari modul `os`.

#### g) Fungsi untuk Memisahkan Data Gambar menjadi Set Pelatihan dan Validasi

```
# Fungsi untuk Memisahkan Data Gambar menjadi Set Pelatihan dan Validasi
def split_data(SOURCE_DIR, TRAINING_DIR, VALIDATION_DIR, SPLIT_SIZE):

    shuffled_source = random.sample(os.listdir(SOURCE_DIR), len(os.listdir(SOURCE_DIR)))

    training_number = int(len(shuffled_source) * SPLIT_SIZE) # Menghitung jumlah gambar yang

    i = 0
    target = TRAINING_DIR

    for item in shuffled_source:
        item_source = os.path.join(SOURCE_DIR, item)
        if os.path.getsize(item_source) == 0:
            print(f'{item} is zero length, so ignoring.')
        else:
            copyfile(item_source, os.path.join(target, item))
            i += 1

    if i == training_number:
        target = VALIDATION_DIR
```

Gambar 1. 9 Gambar Fungsi Pisah Data

Kode di atas mengacak urutan data dalam direktori sumber dan memisahkannya menjadi set pelatihan dan validasi menggunakan fungsi `split_data`. Fungsi ini menggunakan `random.sample` untuk mengacak data, lalu membagi data sesuai dengan proporsi yang ditentukan oleh `SPLIT_SIZE`.

## h) Mengatur dan Memisahkan Data Gambar ke Direktori

```
# Skrip untuk Mengatur dan Memisahkan Data Gambar ke Direktori Pelatihan dan Validasi

# Jalur untuk mengakses dataset
ba_SOURCE_DIR = source_path_ba
ca_SOURCE_DIR = source_path_ca
da_SOURCE_DIR = source_path_da
dha_SOURCE_DIR = source_path_dha
ga_SOURCE_DIR = source_path_ga
ha_SOURCE_DIR = source_path_ha
ja_SOURCE_DIR = source_path_ja
ka_SOURCE_DIR = source_path_ka
la_SOURCE_DIR = source_path_la
ma_SOURCE_DIR = source_path_ma
na_SOURCE_DIR = source_path_na
nga_SOURCE_DIR = source_path_nga
nya_SOURCE_DIR = source_path_nya
pa_SOURCE_DIR = source_path_pa
ra_SOURCE_DIR = source_path_ra
sa_SOURCE_DIR = source_path_sa
ta_SOURCE_DIR = source_path_ta
tha_SOURCE_DIR = source_path_tha
wa_SOURCE_DIR = source_path_wa
ya_SOURCE_DIR = source_path_ya

TRAINING_DIR = os.path.join(root_dir, 'training')
VALIDATION_DIR = os.path.join(root_dir, 'validation')

TRAINING_ba_DIR = os.path.join(TRAINING_DIR, 'ba')
VALIDATION_ba_DIR = os.path.join(VALIDATION_DIR, 'ba')

TRAINING_ca_DIR = os.path.join(TRAINING_DIR, 'ca')
VALIDATION_ca_DIR = os.path.join(VALIDATION_DIR, 'ca')
```

Gambar 1. 10 Gambar Atur dan Pisah Data

Kode di atas mengatur jalur untuk mengakses dataset dan memisahkan data gambar ke dalam direktori pelatihan dan validasi. Setiap kelas memiliki direktori sumber yang sesuai, seperti `ba_SOURCE_DIR` untuk kelas 'ba'. Kode ini juga menentukan direktori pelatihan dan validasi untuk setiap kelas dengan menggunakan fungsi `os.path.join` untuk memastikan struktur direktori yang konsisten.

## i) Pemisahan Data untuk Pelatihan dan Validasi Model

```
TRAINING_ya_DIR = os.path.join(TRAINING_DIR, 'ya')
VALIDATION_ya_DIR = os.path.join(VALIDATION_DIR, 'ya')

# Tentukan proporsi gambar yang digunakan untuk pelatihan
split_size = 0.7

# Menjalankan fungsi di atas
split_data(ba_SOURCE_DIR, TRAINING_ba_DIR, VALIDATION_ba_DIR, split_size)
split_data(ca_SOURCE_DIR, TRAINING_ca_DIR, VALIDATION_ca_DIR, split_size)
split_data(da_SOURCE_DIR, TRAINING_da_DIR, VALIDATION_da_DIR, split_size)
split_data(dha_SOURCE_DIR, TRAINING_dha_DIR, VALIDATION_dha_DIR, split_size)
split_data(ga_SOURCE_DIR, TRAINING_ga_DIR, VALIDATION_ga_DIR, split_size)
```

Gambar 1. 11 Gambar Pisah Data Pelatihan dan Validasi Model

Kode tersebut membagi dataset menjadi set pelatihan dan validasi untuk berbagai kategori data dengan proporsi 70:30. Fungsi `split_data()` digunakan untuk memisahkan data dari direktori sumber ke direktori pelatihan dan validasi yang telah ditentukan.

#### j) Statistik Gambar Pelatihan per Kelas

```
Original fa directory has 108 images
Original sa directory has 108 images
Original ta directory has 108 images
Original tha directory has 108 images
Original wa directory has 108 images
Original ya directory has 108 images

There are 79 images of ba for training
There are 75 images of ca for training
There are 75 images of da for training
There are 75 images of dha for training
There are 75 images of ga for training
There are 71 images of ha for training
There are 75 images of ja for training
There are 75 images of ka for training
There are 75 images of la for training
```

Gambar 1. 12 Gambar Statistik Pelatihan

Gambar di atas menampilkan jumlah gambar dalam setiap direktori untuk data pelatihan berbagai kelas. Setiap direktori berisi gambar yang akan digunakan dalam proses pelatihan model. Informasi ini menunjukkan bahwa semua direktori awal memiliki 108 gambar, sementara jumlah gambar yang akan digunakan untuk pelatihan bervariasi per kelas, dengan jumlah gambar antara 71 hingga 79.

#### k) Membuat Generator Data Pelatihan dan Validasi

```
# Membuat generator data pelatihan dan validasi
"""
Argument:
- TRAINING_DIR (string): jalur direktori yang berisi gambar pelatihan
- VALIDATION_DIR (string): jalur direktori yang berisi gambar pengujian/validasi

Pengembalian:
train_generator, validation_generator - tuple yang berisi generator"""

def train_val_generators(TRAINING_DIR, VALIDATION_DIR):
    train_datagen = ImageDataGenerator(rescale=1./255,
                                       rotation_range=20,
                                       zoom_range=0.2,
                                       shear_range=0.1)

    train_generator = train_datagen.flow_from_directory(directory=TRAINING_DIR,
                                                         target_size=(64, 64),
                                                         batch_size=32,
                                                         class_mode='categorical')

    validation_datagen = ImageDataGenerator(rescale=1./255)
    validation_generator = validation_datagen.flow_from_directory(directory=VALIDATION_DIR,
                                                                  target_size=(64, 64),
                                                                  batch_size=32,
                                                                  class_mode='categorical')

    return train_generator, validation_generator
```

Gambar 1. 13 Gambar Membuat Generator Data

Kode di atas membuat generator data pelatihan dan validasi menggunakan `ImageDataGenerator` dari TensorFlow. Generator ini digunakan untuk mengelola gambar pelatihan dan validasi dengan melakukan augmentasi data seperti rescaling, rotasi, zoom, dan shear. Kode ini menghasilkan dua generator: `train_generator` untuk data pelatihan dan `validation_generator` untuk data validasi, yang siap digunakan untuk melatih model.

## 1) Definisi dan Kompilasi Model CNN

```
# Definisi dan Kompilasi Model Convolutional Neural Network (CNN) atau model cnn

def create_model():
    model = tf.keras.models.Sequential([
        tf.keras.layers.Conv2D(16, (3, 3), activation='relu', input_shape=(64, 64, 3)),
        tf.keras.layers.MaxPooling2D(2, 2),
        tf.keras.layers.Conv2D(32, (3, 3), activation='relu'),
        tf.keras.layers.MaxPooling2D(2, 2),
        tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
        tf.keras.layers.MaxPooling2D(2, 2),
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(512, activation='relu'),
        tf.keras.layers.Dense(20, activation='softmax')
    ])

    model.summary()

    from tensorflow.keras.optimizers import Adam

    model.compile(optimizer=Adam(learning_rate=0.001),
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])

    return model
```

Gambar 1. 14 Gambar Definisi dan Kompilasi Model CNN

Kode di atas mendefinisikan dan mengkompilasi model Convolutional Neural Network (CNN) menggunakan TensorFlow. Model tersebut memiliki tiga lapisan Convolutional dengan ukuran kernel 3x3 dan fungsi aktivasi ReLU, masing-masing diikuti oleh lapisan MaxPooling dengan ukuran 2x2. Model ini diakhiri dengan dua lapisan Dense, satu dengan 512 unit dan fungsi aktivasi ReLU, serta yang terakhir dengan 20 unit dan fungsi aktivasi Softmax. Model dikompilasi menggunakan optimizer Adam dengan laju pembelajaran 0.001 dan menggunakan loss function categorical\_crossentropy serta mengukur akurasi.

### m) Menampilkan Arsitektur dan Detail Parameter Model CNN

```
# Menampilkan Arsitektur dan Detail Parameter Model Convolutional Neural Network
```

```
model = create_model()
```

Model: "sequential\_5"

Layer (type)	Output Shape	Param #
conv2d_15 (Conv2D)	(None, 62, 62, 16)	448
max_pooling2d_15 (MaxPooling2D)	(None, 31, 31, 16)	0
conv2d_16 (Conv2D)	(None, 29, 29, 32)	4640
max_pooling2d_16 (MaxPooling2D)	(None, 14, 14, 32)	0
conv2d_17 (Conv2D)	(None, 12, 12, 64)	18496
max_pooling2d_17 (MaxPooling2D)	(None, 6, 6, 64)	0
flatten_5 (Flatten)	(None, 2304)	0
dense_10 (Dense)	(None, 512)	1180160
dense_11 (Dense)	(None, 20)	10260

=====  
Total params: 1214004 (4.63 MB)  
Trainable params: 1214004 (4.63 MB)  
Non-trainable params: 0 (0.00 Byte)

Gambar 1. 15 Gambar menampilkan Arsitektur dan Parameter CNN

Gambar di atas menampilkan arsitektur dan detail parameter dari model Convolutional Neural Network (CNN) yang digunakan. Model ini dibuat menggunakan fungsi `create_model()`, dan terdiri dari beberapa lapisan convolutional, max-pooling, flatten, dan dense. Tabel menunjukkan jenis lapisan, bentuk keluaran (output shape), dan jumlah parameter yang dapat dilatih (trainable params). Total parameter model adalah 1.214.004, yang semuanya dapat dilatih.

### n) Pelatihan dan Evaluasi Model CNN dengan Data Generator

```
# Pelatihan dan Evaluasi Model CNN dengan Data Generator
```

```
history = model.fit(
```

```
    train_generator,
```

```
    steps_per_epoch=len(train_generator),
```

```
    epochs=15,
```

```
    validation_data=validation_generator,
```

```
    validation_steps=len(validation_generator)
```

```
)
```

Epoch 1/15  
47/47 [=====] - 21s 432ms/step - loss: 2.9759 - accuracy: 0.0689 - val\_loss: 2.6576 - val\_accuracy: 0.1216  
Epoch 2/15  
47/47 [=====] - 18s 381ms/step - loss: 2.0298 - accuracy: 0.3703 - val\_loss: 1.2262 - val\_accuracy: 0.6535  
Epoch 3/15  
47/47 [=====] - 19s 402ms/step - loss: 1.1414 - accuracy: 0.6618 - val\_loss: 1.3410 - val\_accuracy: 0.6565  
Epoch 4/15  
47/47 [=====] - 24s 516ms/step - loss: 0.6446 - accuracy: 0.7948 - val\_loss: 0.7738 - val\_accuracy: 0.7751  
Epoch 5/15  
47/47 [=====] - 18s 374ms/step - loss: 0.4905 - accuracy: 0.8349 - val\_loss: 0.3851 - val\_accuracy: 0.8891  
Epoch 6/15  
47/47 [=====] - 19s 408ms/step - loss: 0.3703 - accuracy: 0.8797 - val\_loss: 0.2432 - val\_accuracy: 0.9255  
Epoch 7/15

Gambar 1. 16 Gambar Pelatihan dan Evaluasi CNN

Gambar di atas menunjukkan proses pelatihan dan evaluasi model CNN menggunakan data generator. Pada bagian atas gambar, terdapat potongan kode yang digunakan untuk melatih model selama 15 epoch, dengan menggunakan `train_generator` untuk data pelatihan dan `validation_generator` untuk data Svalidasi. Pada bagian bawah, ditampilkan hasil pelatihan untuk setiap epoch,

termasuk waktu per step, nilai loss, dan akurasi untuk data pelatihan serta data validasi.

#### o) Visualisasi Akurasi dan Loss Selama Pelatihan dan Validasi Model CNN



Gambar 1. 17 Gambar Visualisasi model CNN

Gambar di atas menunjukkan grafik akurasi dan loss selama proses pelatihan dan validasi model CNN. Grafik pertama menampilkan akurasi pelatihan dan validasi, di mana garis merah menunjukkan akurasi pelatihan dan garis biru menunjukkan akurasi validasi. Grafik kedua menunjukkan loss pelatihan dan validasi, dengan garis merah menunjukkan loss pelatihan dan garis biru menunjukkan loss validasi. Hasil ini membantu dalam memahami bagaimana model berperilaku selama pelatihan dan validasi, serta apakah terjadi overfitting atau underfitting.