LAPORAN PENGENALAN POLA
AKSARA JAWA

Disusun oleh kelompok 12 :

1. wahid ivan saputra (32602100123)
2. zulham prabandanu (32602100126)
3. andhi rohman (32602100134

PROGRAM STUDI TEKINIK INFORMATIKA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM SULTAN AGUNG

SEMARANG

2023

BAB I

Pengambilan Data

A. perencanaan pengabilan data

Data yang kita ambil berdasarkan pada web kaggle, disana terdapat data data tulisan tangan hanacaraka yang di upload oleh seseorang, oleh karena itu kita menggunakan data yang sudah ada daripada harus membuat ulang data tesebut. Untuk data tersbut terdapat pada link berikut https://github.com/arryaaas/Hanacaraka-Digital-Handwriting-CNN Tahap tahapan pada preprocessing:

1. Pengumpulan Data

   Mengumpulkan dataset huruf yang akan digunakan untuk pelatihan dan pengujian model. Dataset bisa berasal dari berbagai sumber seperti gambar tulisan tangan, cetakan komputer, atau dataset standar seperti MNIST.

2. Desizing data

   Mengubah ukuran gambar ke dimensi yang seragam dan menormalkan nilai pixel:

   - **Resizing:** Mengubah ukuran gambar agar memiliki dimensi yang konsisten, misalnya 28x28 piksel.

3. Konversi ke Grayscale

   Mengubah gambar berwarna menjadi gambar hitam-putih (grayscale) untuk mengurangi kompleksitas dan ukuran data:

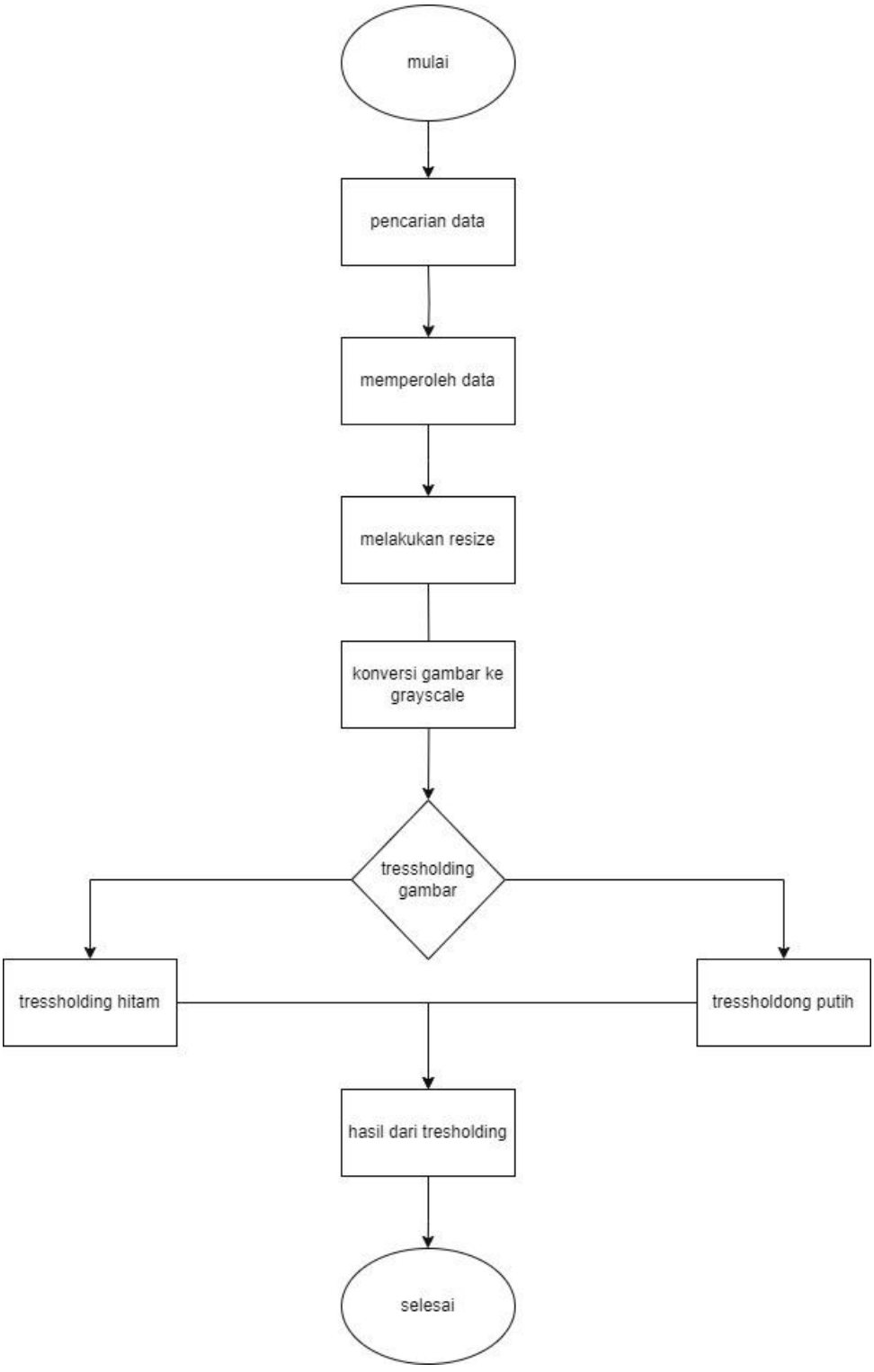
   - **Grayscale Conversion:** Mengubah gambar berwarna menjadi grayscale.

4. Tressholding

   Mengubah gambar grayscale menjadi gambar biner (hitam-putih) untuk menyoroti huruf dan mengurangi noise:

   - **Thresholding:** Menentukan ambang batas untuk mengubah piksel menjadi hitam atau putih.

B. flowchart pemrosesan data dan preprocessing



Gambar 1. 1 Flowchart

BAB II
Membaca Data dan Preprocessing

A. Membaca Data dan Preprocessing

Berikut merupakan proses untuk membaca data dan melakukkan preprocessing :

1. Ekstrak dataset

   Dataset hanacaraka terdiri dari 20 folder yaitu ha, na, ca, ra, ka, da, ta,sa, wa, la, pa, dha, ja, ya, nya, ma, ga, ba, tha, dan nga yang masing- masing folder terdiri dari 510 gambar. Total gambar secara keseluruhan pada dataset adalah 10.200 gambar

   Menghubungkan google colab dan google drive

```
from google.colab import drive

drive.mount("/content/gdrive/")
```

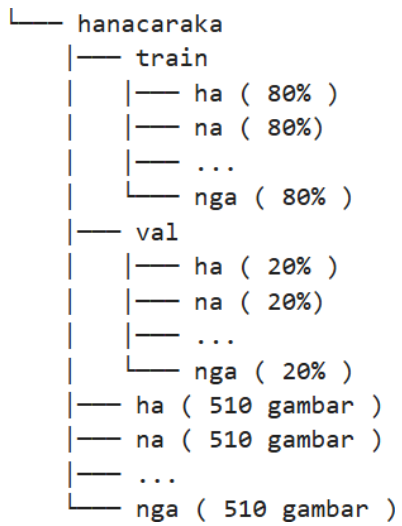
Gambar 1. 2 load dataset pada gdrive

   Mengekstrak file

```
import zipfile

filename = "/content/gdrive/My Drive/dataset.zip"
zip_ref = zipfile.ZipFile(filename)
zip_ref.extractall()
zip_ref.close()
```

Gambar 1. 3 imort dataset pada gdrive

2. Pembagian Set Pelatihan: Membagi dataset menjadi set pelatihan dan validasi. Kemudian setiap folder karakter aksara jawa akan dibagi menjadi 2 bagian, yaitu train dan validation dengan proporsi masing-masing 80% dan 20%.



Gambar 1. 4 struktur folder

3. Pada bagian ini adalah Memproses data sebelum di load menggunakan ImageDataGenerator(). ImageDataGenerator() dapat melakukan preprocessing, pelabelan sampel otomatis, dan augmentasi gambar Kemudian load data ke dalam memori dengan fungsi flow_from_directory().

```
import os
import shutil
from sklearn.model_selection import train_test_split

# Membuat direktori train dan validation
base_dir = "/content/hanacaraka"
train_dir = os.path.join(base_dir, "train")
val_dir = os.path.join(base_dir, "val")

os.mkdir(train_dir)
os.mkdir(val_dir)

# Inisialisasi 20 folder karekter aksara jawa
ha_dir = os.path.join(base_dir, "ha")
na_dir = os.path.join(base_dir, "na")
ca_dir = os.path.join(base_dir, "ca")
ra_dir = os.path.join(base_dir, "ra")
ka_dir = os.path.join(base_dir, "ka")
da_dir = os.path.join(base_dir, "da")
ta_dir = os.path.join(base_dir, "ta")
sa_dir = os.path.join(base_dir, "sa")
wa_dir = os.path.join(base_dir, "wa")
la_dir = os.path.join(base_dir, "la")
pa_dir = os.path.join(base_dir, "pa")
dha_dir = os.path.join(base_dir, "dha")
ja_dir = os.path.join(base_dir, "ja")
ya_dir = os.path.join(base_dir, "ya")
nya_dir = os.path.join(base_dir, "nya")
ma_dir = os.path.join(base_dir, "ma")
ga_dir = os.path.join(base_dir, "ga")
ba_dir = os.path.join(base_dir, "ba")
tha_dir = os.path.join(base_dir, "tha")
nga_dir = os.path.join(base_dir, "nga")
```

Gambar 1. 5 load dataset

4. Menampilkan kelas-kelas pada dataset. Urutan kelas ini nantinya akan dijadikan acuan dalam membuat array classes yang digunakan dalam proses uji coba.

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(
    rescale = 1./255,
    rotation_range = 20,
    shear_range = 0.2,
    zoom_range = 0.2,
    fill_mode = "nearest"
)

validation_datagen = ImageDataGenerator(
    rescale = 1./255,
    rotation_range = 20,
    shear_range = 0.2,
    zoom_range = 0.2,
    fill_mode = "nearest"
)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size = (100, 100),
    batch_size = 32,
    class_mode = "categorical",    # Gunakan categorical untuk klasifikasi 3 kelas atau lebih,
    color_mode = "grayscale"        # untuk klasifikasi dua kelas gunakan binary
)

validation_generator = validation_datagen.flow_from_directory(
    val_dir,
    target_size = (100, 100),
    batch_size = 32,
    class_mode = "categorical",    # Gunakan categorical untuk klasifikasi 3 kelas atau lebih
    color_mode = "grayscale"        # untuk klasifikasi dua kelas gunakan binary
)
```

```
Found 8160 images belonging to 20 classes.
Found 2040 images belonging to 20 classes.
```

Gambar 1. 6 menampilkan kelas dataset

5. Menampilkan mapping atau peta dari nama kelas Hanacaraka

```
print(train_generator.class_indices)
```

```
{'ba': 0, 'ca': 1, 'da': 2, 'dha': 3, 'ga': 4, 'ha': 5, 'ja': 6, 'ka': 7, 'la': 8, 'ma': 9, 'na': 10, 'nga':
11, 'nya': 12, 'pa': 13, 'ra': 14, 'sa': 15, 'ta': 16, 'tha': 17, 'wa': 18, 'ya': 19}
```

Gambar 1. 7 mapping data

BAB III

Membaca Data dan Preprocessing

A. Penggunaan Metode, Training dan Testing

1. Pemilihan Metode

   Pada tugas ini memutuskan untuk memilih metode CNN. Convolutional Neural Network (CNN) adalah algoritma deep learning yang digunakan untuk memproses inputan data gambar, menentukan kepentingan (bobot dan bias yang dapat dipelajari) ke berbagai aspek dalam gambar dan berfungsi untuk membedakan objek satu dengan objek lainnya. Keunggulan dari metode ini antara lain :

   - Memberikan hasil yang lebih representatif dalam bentuk 3D.
   - Mampu menerima dan memproses banyak input sekaligus dalam satu waktu.
   - Penggunaannya lebih luas dan bisa diaplikasikan dalam berbagai bidang.
   - Lebih hemat daya.

   Serta kekurangannya yang dimilikinnya yaitu :

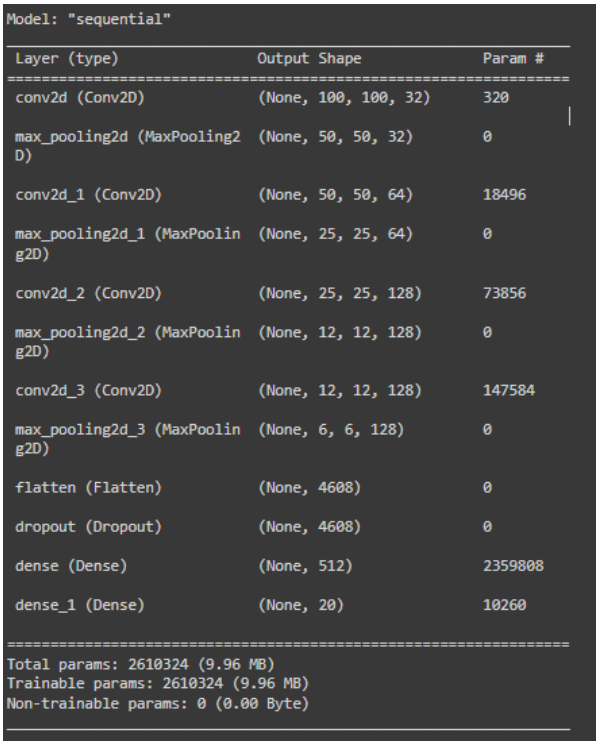
   - Biaya operasional yang mahal.
   - Bila ada ketidakstabilan *input*, maka kinerja CNN bisa menurun.

2. Proses pengerjaan

```
import tensorflow as tf

model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), padding = "same", activation = "relu", input_shape = (100, 100, 1)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(64, (3, 3), padding = "same", activation = "relu"),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(128, (3, 3), padding = "same", activation = "relu"),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(128, (3, 3), padding = "same", activation = "relu"),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(512, activation = "relu"),
    tf.keras.layers.Dense(20, activation = "softmax")
])

model.summary()
```

Gambar 3. 1 inisialisasi project

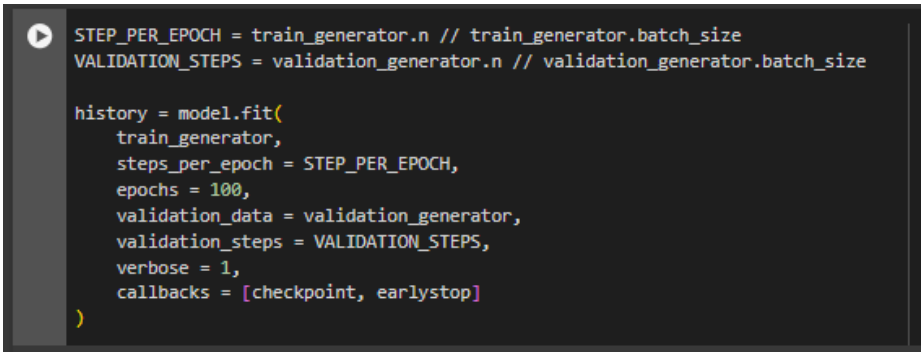
Pada gamabr 3.1 menampilkan inisialisasi proyek yang akan di gunakan menggunakan google collab, dimana penggunan tesorflow sebagai library penunjang pengerjaan.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 100, 100, 32)      320

 max_pooling2d (MaxPooling2  (None, 50, 50, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 50, 50, 64)        18496

 max_pooling2d_1 (MaxPoolin  (None, 25, 25, 64)        0
 g2D)

 conv2d_2 (Conv2D)           (None, 25, 25, 128)       73856

 max_pooling2d_2 (MaxPoolin  (None, 12, 12, 128)       0
 g2D)

 conv2d_3 (Conv2D)           (None, 12, 12, 128)       147584

 max_pooling2d_3 (MaxPoolin  (None, 6, 6, 128)         0
 g2D)

 flatten (Flatten)           (None, 4608)              0

 dropout (Dropout)           (None, 4608)              0

 dense (Dense)               (None, 512)               2359808

 dense_1 (Dense)             (None, 20)                10260

=================================================================
Total params: 2610324 (9.96 MB)
Trainable params: 2610324 (9.96 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Gambar 3. 2 output

Pada gambar 3.2 menampilkan output dari gambar 3.1 dimana terdapat perubahan ukuran gambar ata *resizing* data, dengan adanya memperkecil ukuran ini tetap tidak menngurangi informasi dari setiap gambar, dari output tersebut gambar di perkecil sampai ukuran 50x50.



```
STEP_PER_EPOCH = train_generator.n // train_generator.batch_size
VALIDATION_STEPS = validation_generator.n // validation_generator.batch_size

history = model.fit(
    train_generator,
    steps_per_epoch = STEP_PER_EPOCH,
    epochs = 100,
    validation_data = validation_generator,
    validation_steps = VALIDATION_STEPS,
    verbose = 1,
    callbacks = [checkpoint, earlystop]
)
```

Gambar 3. 3 memulai training

Pada gambar 3.3 menampilkan kode proses untuk training dataset yang sudah ada.

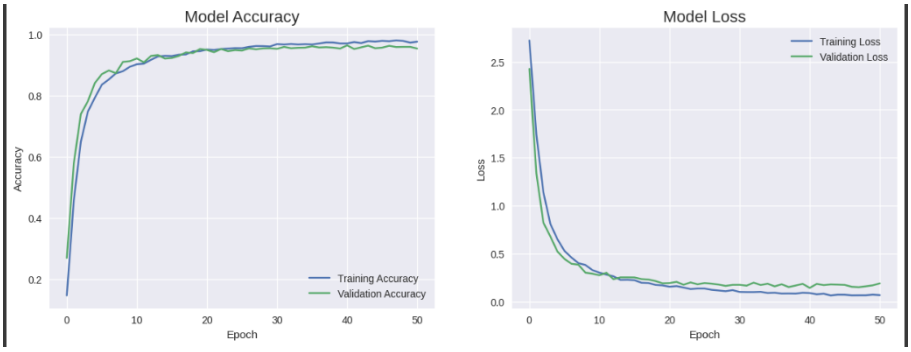
```
Epoch 1/100
255/255 [==============================] - ETA: 0s - loss: 2.7224 - accuracy: 0.1479
Epoch 1: val_accuracy improved from -inf to 0.26984, saving model to model.h5
255/255 [==============================] - 243s 948ms/step - loss: 2.7224 - accuracy: 0.1479 - val_loss: 2.4262 - val_accuracy: 0.2698
Epoch 2/100
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file via `
  saving_api.save_model(
255/255 [==============================] - ETA: 0s - loss: 1.7434 - accuracy: 0.4574
Epoch 2: val_accuracy improved from 0.26984 to 0.57837, saving model to model.h5
255/255 [==============================] - 226s 885ms/step - loss: 1.7434 - accuracy: 0.4574 - val_loss: 1.3378 - val_accuracy: 0.5784
Epoch 3/100
255/255 [==============================] - ETA: 0s - loss: 1.1408 - accuracy: 0.6484
Epoch 3: val_accuracy improved from 0.57837 to 0.73909, saving model to model.h5
255/255 [==============================] - 230s 901ms/step - loss: 1.1408 - accuracy: 0.6484 - val_loss: 0.8245 - val_accuracy: 0.7391
Epoch 4/100
255/255 [==============================] - ETA: 0s - loss: 0.8105 - accuracy: 0.7468
Epoch 4: val_accuracy improved from 0.73909 to 0.78075, saving model to model.h5
255/255 [==============================] - 233s 916ms/step - loss: 0.8105 - accuracy: 0.7468 - val_loss: 0.6798 - val_accuracy: 0.7808
Epoch 5/100
255/255 [==============================] - ETA: 0s - loss: 0.6525 - accuracy: 0.7924
Epoch 5: val_accuracy improved from 0.78075 to 0.84028, saving model to model.h5
255/255 [==============================] - 236s 925ms/step - loss: 0.6525 - accuracy: 0.7924 - val_loss: 0.5243 - val_accuracy: 0.8403
Epoch 6/100
255/255 [==============================] - ETA: 0s - loss: 0.5311 - accuracy: 0.8349
Epoch 6: val_accuracy improved from 0.84028 to 0.86954, saving model to model.h5
255/255 [==============================] - 232s 909ms/step - loss: 0.5311 - accuracy: 0.8349 - val_loss: 0.4471 - val_accuracy: 0.8695
Epoch 7/100
255/255 [==============================] - ETA: 0s - loss: 0.4612 - accuracy: 0.8522
Epoch 7: val_accuracy improved from 0.86954 to 0.88145, saving model to model.h5
255/255 [==============================] - 232s 909ms/step - loss: 0.4612 - accuracy: 0.8522 - val_loss: 0.3964 - val_accuracy: 0.8814
Epoch 8/100
255/255 [==============================] - ETA: 0s - loss: 0.4029 - accuracy: 0.8718
Epoch 8: val_accuracy did not improve from 0.88145
255/255 [==============================] - 231s 905ms/step - loss: 0.4029 - accuracy: 0.8718 - val_loss: 0.3863 - val_accuracy: 0.8725
Epoch 9/100
255/255 [==============================] - ETA: 0s - loss: 0.3832 - accuracy: 0.8789
Epoch 9: val_accuracy improved from 0.88145 to 0.90923, saving model to model.h5
255/255 [==============================] - 231s 906ms/step - loss: 0.3832 - accuracy: 0.8789 - val_loss: 0.3034 - val_accuracy: 0.9092
Epoch 10/100
255/255 [==============================] - ETA: 0s - loss: 0.3300 - accuracy: 0.8935
Epoch 10: val_accuracy improved from 0.90923 to 0.91171, saving model to model.h5
255/255 [==============================] - 229s 899ms/step - loss: 0.3300 - accuracy: 0.8935 - val_loss: 0.2926 - val_accuracy: 0.9117
Epoch 11/100
255/255 [==============================] - ETA: 0s - loss: 0.3029 - accuracy: 0.9017
Epoch 11: val_accuracy improved from 0.91171 to 0.92063, saving model to model.h5
255/255 [==============================] - 233s 913ms/step - loss: 0.3029 - accuracy: 0.9017 - val_loss: 0.2767 - val_accuracy: 0.9206
Epoch 12/100
255/255 [==============================] - ETA: 0s - loss: 0.2829 - accuracy: 0.9037
Epoch 12: val_accuracy did not improve from 0.92063
255/255 [==============================] - 232s 910ms/step - loss: 0.2829 - accuracy: 0.9037 - val_loss: 0.3017 - val_accuracy: 0.9077
```

Gambar 3. 4 Hasil Dari Training

```python
import matplotlib.pyplot as plt

acc = history.history["accuracy"]
val_acc = history.history["val_accuracy"]

loss = history.history["loss"]
val_loss = history.history["val_loss"]

plt.style.use("seaborn")
fig, ax = plt.subplots(nrows = 1, ncols = 2, figsize = (15, 5))

ax[0].plot(acc, label = "Training Accuracy")
ax[0].plot(val_acc, label = "Validation Accuracy")
ax[0].legend(loc = "lower right")
ax[0].set_title("Model Accuracy", fontsize = 16)
ax[0].set_xlabel("Epoch")
ax[0].set_ylabel("Accuracy")

ax[1].plot(loss, label = "Training Loss")
ax[1].plot(val_loss, label = "Validation Loss")
ax[1].legend(loc = "upper right")
ax[1].set_title("Model Loss", fontsize = 16)
ax[1].set_xlabel("Epoch")
ax[1].set_ylabel("Loss")

plt.show()
```

Gambar 3. 5 visualisasi data

Pada gambar 3.5 menampilkan kode untuk train visualisasi dataset yang digunakan dengan menggunakan library matplotlib, hal tersebut bertujuan untuk mengetahui n history model accuracy dan model loss selama proses training berlangsung. Output dari kode tersebut akan berupa sebuah grafik yang menampilkan keakurasian saat proses training.



Gambar 3. 6 output grafik training

Pada gambar 3.6 menampilkan dari output dari kode yang ditampilkan pada gambar 3.5, grafik diatas menunjukan tingkat dari proses training dan proses validasi dataset. Hasil dari proses training diatas menunjukan angka yang cukup signifikan dari validasi, hal tersebut menunjukan bahwa ke akuratan dari proses terhadap dataset yang tersedia terbilang cukup baik.

```
[ ]  from keras.models import load_model

     load_model = load_model("/content/Hanacaraka-Digital-Handwriting-CNN-main/model.h5")
```

Gambar 3. 7 melakukan load gambar

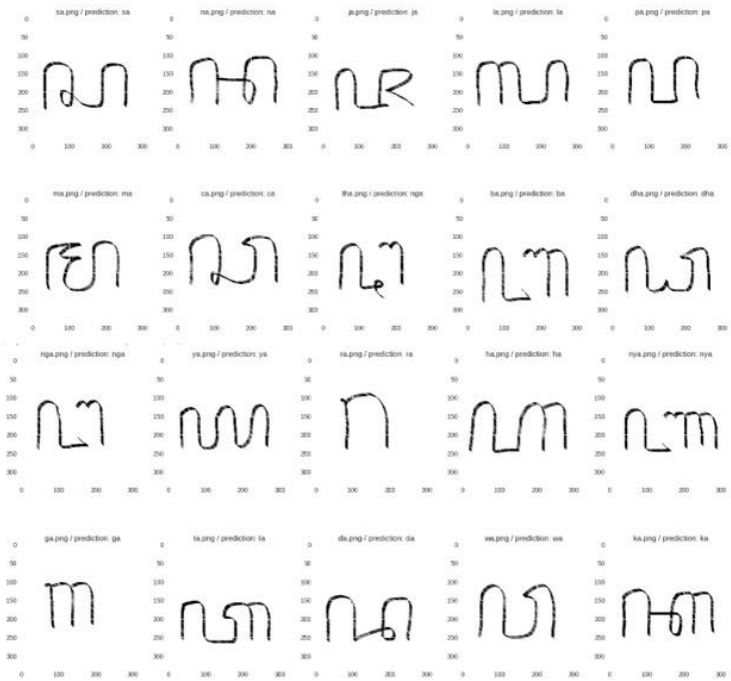
Pada gambar 3.7 menampilkan kode yang berfungsi untuk melakukan load data.

```python
import os
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from keras.preprocessing import image

classes = [
    "ba", "ca", "da", "dha", "ga", "ha", "ja", "ka", "la", "ma",
    "na", "nga", "nya", "pa", "ra", "sa", "ta", "tha", "wa", "ya"
]

path = "/content/Hanacaraka-Digital-Handwriting-CNN-main/prediction"
fig, ax = plt.subplots(nrows = 4, ncols = 5, figsize = (20, 20))
# Muat model dengan nama variabel yang berbeda untuk menghindari penimpaan fungsi
model = load_model("/content/Hanacaraka-Digital-Handwriting-CNN-main/model.h5")

x = 0
for y, img_name in enumerate(os.listdir(path)):
    img_path = "{}{}{}".format(path, "/", img_name)
    img = image.load_img(img_path, color_mode = "grayscale", target_size = (100, 100))

    img = image.img_to_array(img)
    img = np.expand_dims(img, axis = 0)

     # Gunakan variabel 'model' untuk prediksi
    result = model.predict(img)

    img_preview = mpimg.imread(img_path)

    if (y > 1) and (y % 5 == 0):
        x += 1

    ax[x, (y % 5)].set_title("{} / prediction: {}".format(img_name, classes[np.argmax(result)]))
    ax[x, (y % 5)].imshow(img_preview)
```

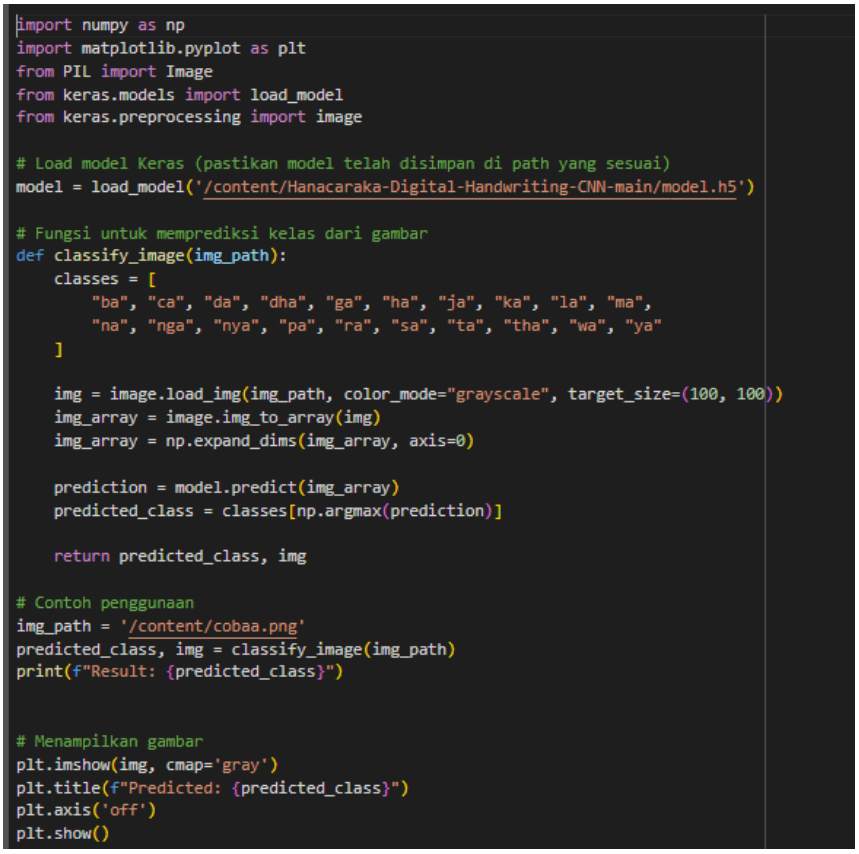
Gambar 3. 8 penulisan kode uji coba

Pada gambar 3.8 menampilkan kode pengujian terhadap data baru yang data tersebut tidak diikutsertakan pada pengujian sebelumnya.

Output



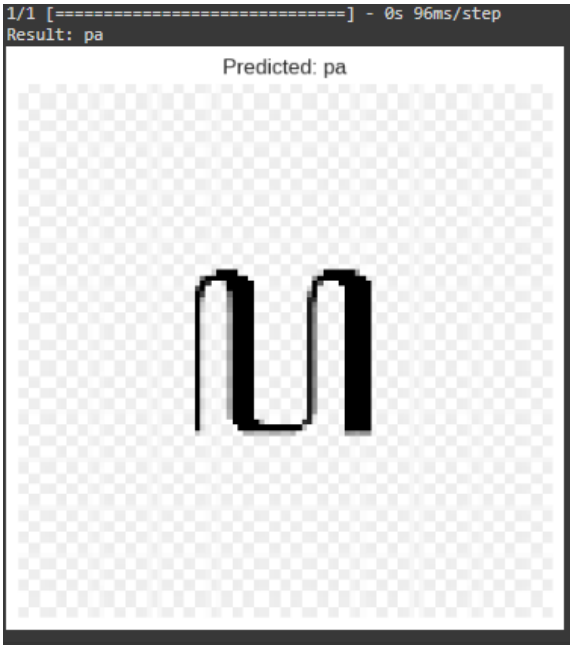
Gambar 3. 9 ouput pengujian



```python
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
from keras.models import load_model
from keras.preprocessing import image

# Load model Keras (pastikan model telah disimpan di path yang sesuai)
model = load_model('/content/Hanacaraka-Digital-Handwriting-CNN-main/model.h5')

# Fungsi untuk memprediksi kelas dari gambar
def classify_image(img_path):
    classes = [
        "ba", "ca", "da", "dha", "ga", "ha", "ja", "ka", "la", "ma",
        "na", "nga", "nya", "pa", "ra", "sa", "ta", "tha", "wa", "ya"
    ]

    img = image.load_img(img_path, color_mode="grayscale", target_size=(100, 100))
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)

    prediction = model.predict(img_array)
    predicted_class = classes[np.argmax(prediction)]

    return predicted_class, img

# Contoh penggunaan
img_path = '/content/cobaa.png'
predicted_class, img = classify_image(img_path)
print(f"Result: {predicted_class}")


# Menampilkan gambar
plt.imshow(img, cmap='gray')
plt.title(f"Predicted: {predicted_class}")
plt.axis('off')
plt.show()
```

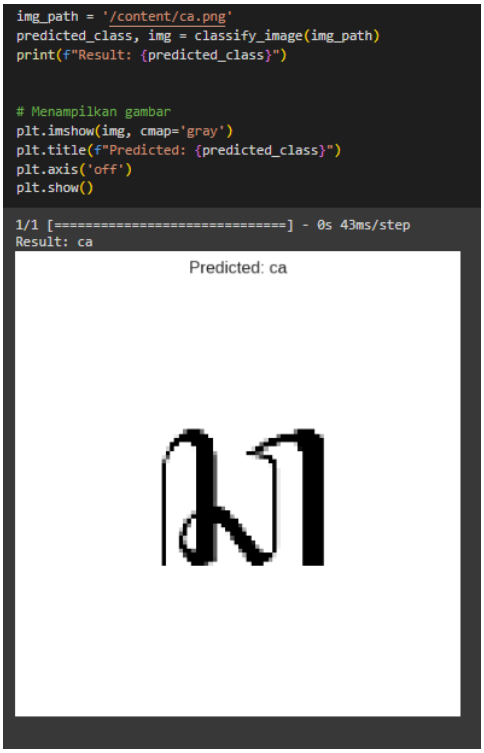
Gambar 3. 10 testing pengujian

Pada gambar 3.9 menampilkan kode yang berfungsi untuk melakukan proses pengetesan. Proses pengetesan tersebut mennggunakan sample berupa huruf aksara jawa digital yanng sudah kita persiapkan.



Gambar 3. 11 output hasil

Pada gambar 3.10 merupakan hasil output dari kode yang dituliskan pada gambar 3.9. dapat dilihat sistem dapat menebak dan mengenali pada data uji yang kita berikan. Hal tersebut membuktikan bahwa pengujian berhasil dengan baik.



Gambar 3. 12hasil lain dari pengujian

Pada gambar 3.11 menampilkan hasil lain tes pengujian yang dilakukan.