LAPORAN
PROYEK AKHIR PENGENALAN POLA AKSARA JAWA

Disusun Oleh :

1. Hendra Irawan Wijaya Kusuma (32602100049)
2. Hendri Kurniawan (32602100050)
3. Hidayatul Muawanah (32602100052)
4. Hilda Putri Cahyaningrum (32602100053)

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM SULTAN AGUNG

SEMARANG

2024

A. Data Collecting & Preprocessing

1. Menghubungkan Google Colab Dengan Google Drive

```
[2]  from google.colab import drive
     drive.mount('/content/drive/')

     Mounted at /content/drive/
```

2. Mengekstrak Dataset

```
import zipfile
import os

# Path ke file ZIP dan direktori ekstraksi
zip_file_path = '/content/drive/MyDrive/dataset.zip'
extract_dir = '/content/drive/MyDrive/dataset'

# Ekstrak file ZIP
with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
    zip_ref.extractall(extract_dir)
```

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.

3. Menyiapkan Dataset

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

# Membuat direktori train dan validation
base_dir = "/content/drive/MyDrive/dataset/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")
```

Membuat direktori train dan validation. Kemudian setiap folder karakter aksara jawa akan dibagi menjadi 2 bagian, yaitu train dan validation dengan proporsi masing-masing 80% dan 20%.

4. Memproses data

```
[21] 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.
```

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().

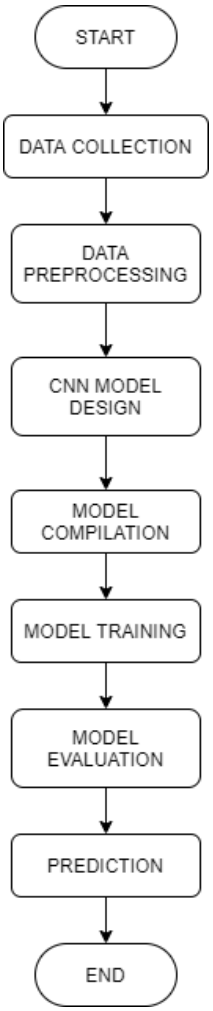
5. Menampilkan kelas-kelas dataset

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

```
{'ba': 0, 'ca': 1, 'da': 2, 'dha': 3, 'ga': 4, 'h
```

Menampilkan kelas-kelas pada dataset.

B. Flowchart



C. Penggunaan Metode Pengenalan Pola

1. Membuat Arsitektur Model CNN

```
[ ] 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),                       # Untuk menghindari terjadinya overfitting
        tf.keras.layers.Dense(512, activation = "relu"),
        tf.keras.layers.Dense(20, activation = "softmax")   # Gunakan softmax untuk klasifikasi 3 kelas atau lebih,
    ])                                                      # untuk klasifikasi dua kelas gunakan sigmoid

    model.summary()
```

Keterangan:

- Pada Convolutional layer pertama setiap satu input gambar akan menghasilkan 32 gambar baru dengan ukuran (100 x 100). Kemudian, resolusi tiap gambar akan diperkecil dengan tetap mempertahankan informasi pada gambar menggunakan MaxPoling layer yang berukuran (2, 2) dan menghasilkan ukuran output gambar sebesar (50 x 50). Proses ini juga berlaku untuk Convolutional dan MaxPoling layer.
- Output dari MaxPoling layer terakhir yang terdiri dari 128 gambar dengan ukuran (6, 6) akan diubah ke dalam bentuk array 1D (tensor 1D) dan menghasilkan output berukuran (4608). Lalu Menggunakan Dropout (0.5) untuk mengurangi overfitting.
- Output tersebut kemudian masuk ke dalam Dense layer pertama yang memiliki 512 neuron dan menghasilkan output dengan ukuran (512).
- Output dari Dense layer pertama akan diteruskan menuju Dense layer kedua yang memiliki 20 neuron sehingga akan menghasilkan output dengan ukuran (20).

```
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 diatas merupakan hasil dari model tensorflow

2. Compile Model

   Menentukan loss function, optimizer, dan metrics yang akan digunakan.

```
model.compile(                          # Gunakan categorical_crossentropy untuk klasifikasi 3 kelas atau lebih,
    loss = "categorical_crossentropy",  # untuk klasifikasi dua kelas gunakan binary_crossentropy
    optimizer = tf.optimizers.Adam(
        learning_rate = 0.0003          # Learning rate 0.0003 ( default = 0.001 ) untuk menghindari terjadinya overfitting
    ),
    metrics = ["accuracy"]
)
```

3. Membuat Callback

   - ModelCheckPoint() digunakan untuk menyimpan model setelah setiap epoch.
   - EarlyStopping() digunakan untuk menghentikan proses training lebih awal.

```
from keras.callbacks import ModelCheckpoint, EarlyStopping

checkpoint = ModelCheckpoint("model.h5", monitor = "val_accuracy", mode = "auto", save_best_only = True, verbose = 1)
earlystop = EarlyStopping(monitor = "val_accuracy", min_delta = 0, patience = 10, verbose = 1, restore_best_weights = True)
```

4. Train Model

   Untuk melatih model dapat menggunakan fungsi fit().

```
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 dibawah ini merupakan hasil terakhir train yaitu:

1. Loss: 0.0540
2. Accuracy: 0.9805
3. Val_loss: 0.1350
4. Val_accuracy:0.9623

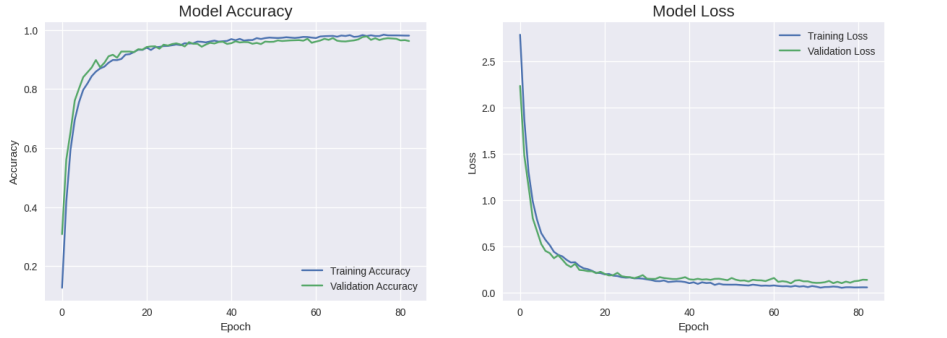
```
0.97817
6s 182ms/step - loss: 0.0557 - accuracy: 0.9812 - val_loss: 0.1045 - val_accuracy: 0.9707

TA: 0s - loss: 0.0532 - accuracy: 0.9812
0.97817
8s 187ms/step - loss: 0.0532 - accuracy: 0.9812 - val_loss: 0.1205 - val_accuracy: 0.9697

TA: 0s - loss: 0.0539 - accuracy: 0.9810
0.97817
7s 183ms/step - loss: 0.0539 - accuracy: 0.9810 - val_loss: 0.1250 - val_accuracy: 0.9643

TA: 0s - loss: 0.0547 - accuracy: 0.9806
0.97817
7s 185ms/step - loss: 0.0547 - accuracy: 0.9806 - val_loss: 0.1369 - val_accuracy: 0.9653

TA: 0s - loss: 0.0540 - accuracy: 0.9805
0.97817
best epoch: 73.
8s 189ms/step - loss: 0.0540 - accuracy: 0.9805 - val_loss: 0.1350 - val_accuracy: 0.9623
```

5. Visualisasi Data

Visualisasi data untuk menampilkan history model accuracy dan model loss selama proses training berlangsung.

```
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()
```

```
<ipython-input-27-ff10d37cf498>:9: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecated since 3.6, as they no longer correspond to the st
  plt.style.use("seaborn")
```



6. Download Model

Menyalin model terbaik yang telah disimpan dengan callback selama proses training berlangsung ke dalam Google Drive.

```
!cp /content/model.h5 "/content/gdrive/MyDrive"
```

7. Load Model

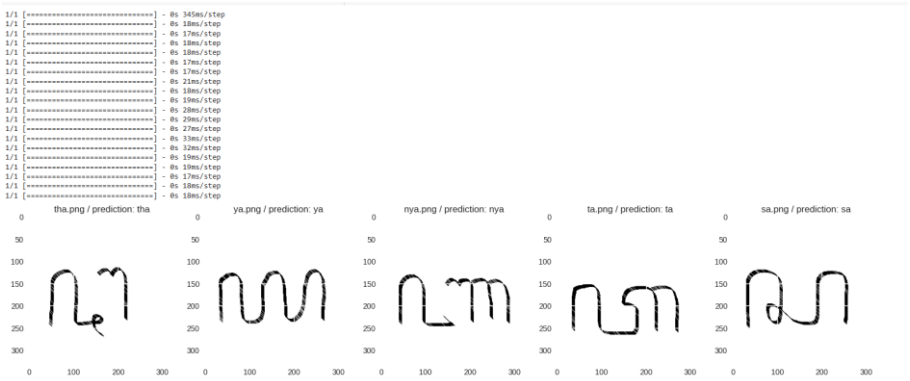
Setelah model disimpan pada Google Drive, model tersebut akan didownload dan diload untuk uji coba.

```
[ ]  from keras.models import load_model

     loaded_model = load_model("/content/model.h5")
```

8. Prediksi

Data yang digunakan untuk uji coba ini.

```
import os
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from keras.preprocessing import image
from google.colab import files

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

path = "/content/drive/MyDrive/prediction"
fig, ax = plt.subplots(nrows = 4, ncols = 5, figsize = (20, 20))

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)

    result = loaded_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 diatas merupakan prediksi yang akan digunakan untuk testing

9. Upload image untuk prediksi

Testing dari model CNN untuk prediksi aksara jawa

```
[ ]  from tensorflow.keras.preprocessing import image
     import numpy as np

     def predict_image(img_path):
         # Load dan preprocess gambar
         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)

         # Lakukan prediksi
         prediction = loaded_model.predict(img)
         predicted_class = classes[np.argmax(prediction)]
         return predicted_class
```

```
[ ]  uploaded = files.upload()

     for fn in uploaded.keys():
         img_path = fn
         result = predict_image(img_path)

         # Display the image
         img = image.load_img(img_path, color_mode="grayscale")
         plt.imshow(img, cmap='gray')
         plt.title(f"Predicted label: {result}")
         plt.axis('off')
         plt.show()

         print(f"Predicted label for {fn}: {result}")
```

Gambar dibawah ini merupakan hasil prediksinya.

Pilih File 11.jpg
- **11.jpg**(image/jpeg) - 780 bytes, last modified: 15/7/2024 - 100% done

```
Saving 11.jpg to 11.jpg
1/1 [==============================] - 0s 18ms/step
```

Predicted label: pa

```
Predicted label for 11.jpg: pa
```

Pilih File 8.jpg
- **8.jpg**(image/jpeg) - 900 bytes, last modified: 15/7/2024 - 100% done

```
Saving 8.jpg to 8 (1).jpg
1/1 [==============================] - 0s 24ms/step
```

Predicted label: ma

```
Predicted label for 8 (1).jpg: ma
```

Pilih File 8.jpg

- **8.jpg**(image/jpeg) - 913 bytes, last modified: 15/7/2024 - 100% done

```
Saving 8.jpg to 8 (2).jpg
1/1 [==============================] - 0s 36ms/step
```

Predicted label: sa

```
Predicted label for 8 (2).jpg: sa
```

LINK COLAB:

https://colab.research.google.com/drive/1IDcwcD1V3ZNjvir2ZjQ1ZaLjpN51s_H_?u sp=sharing

LINK DATASET:

https://github.com/arryaaas/Hanacaraka-Digital-Handwriting-CNN/blob/main/dataset.zip