

PENGENALAN POLA AKSARA JAWA



Disusun Oleh Kelompok 11 :

1. Rafif Huda Aditya 32602100109
2. Reysita Nazela Fitrah 32602100110
3. Rika Amelia 32602100112
4. Tubagus Alwasi'I 32602100121

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

A. Perencanaan

1. Pengambilan Data

Langkah pertama, mencari dataset aksara jawa di Kaggle dan unduh data tersebut pastikan dataset memiliki label yang sesuai dengan huruf aksara jawa yang terkandung. Format data berupa gambar (JPEG,PNG) aksara jawa. Kami mengambil dataset di Kaggel dan berikut link dataset : <https://www.kaggle.com/datasets/vzrengamani/hanacaraka>

2. Preprocessing Data

a. Resizing

Mengubah ukuran gambar ke dimensi yang diinginkan.

b. Grayscale Conversion

Mengonversikan gambar dari yang berwarna menjadi skala abu abu.

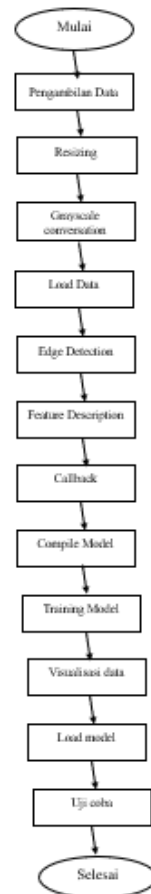
c. Load data

Memuat dataset ke dalam bahasa pemrograman Python.

3. Model

Menggunakan model Convolutional Neural Networks (CNN) untuk melatih model pada fitur yang diekstraksi.

4. Flowchart



B. Membaca Data dan Preprocessing

Berikut merupakan proses untuk membaca data dan melakukan 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/")
```

Mengekstrak dataset

```
import zipfile

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

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%.

```
├── hanacaraka
│   ├── train
│   │   ├── ha ( 80% )
│   │   ├── na ( 80% )
│   │   ├── ...
│   │   └── nga ( 80% )
│   ├── val
│   │   ├── ha ( 20% )
│   │   ├── na ( 20% )
│   │   ├── ...
│   │   └── nga ( 20% )
│   ├── ha ( 510 gambar )
│   ├── na ( 510 gambar )
│   ├── ...
│   └── nga ( 510 gambar )
```

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 karakter 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")
```

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", # Untuk kategorikal 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", # Untuk kategorikal 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.
```

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}
```

C. Penggunaan Metode, Training dan Testing

1. Penggunaan Metode

a. CNN

Menggunakan metode Convolutional Neural Networks (CNN) untuk melatih model pada fitur yang diekstraksi.

b. 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()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 100, 100, 32)	320
max_pooling2d (MaxPooling2D)	(None, 50, 50, 32)	0
conv2d_1 (Conv2D)	(None, 50, 50, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 25, 25, 64)	0
conv2d_2 (Conv2D)	(None, 25, 25, 128)	73856
max_pooling2d_2 (MaxPooling2D)	(None, 12, 12, 128)	0
conv2d_3 (Conv2D)	(None, 12, 12, 128)	147584
max_pooling2d_3 (MaxPooling2D)	(None, 6, 6, 128)	0
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)
```

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 (

- 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).

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

```
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 =
```

- ```
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])
```

[illegible]

Pada proses training, estimasi waktu yang dibutuhkan dalam satu epoch berkisar antara 4 – 5 menit. Hal ini dikarenakan proses training yang berjalan menggunakan GPU dari Google Colab.

#### b. Visualisasi Data

Membuat 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(rows = 2, cols = 2, figsize = (15, 8))

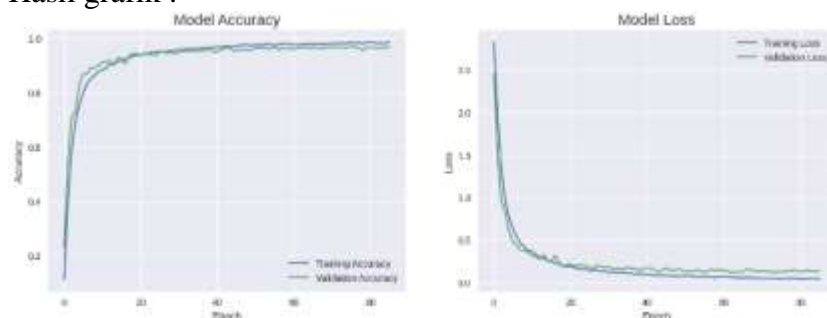
ax[0].plot(acc, label = "Training Accuracy")
ax[0].plot(val_acc, label = "Validation Accuracy")
ax[0].legend(loc = "lower right")
ax[0].set_title("Model Accuracy", fontsize = 18)
ax[0].set_xlabel("Epoch")
ax[0].set_ylabel("Accuracy")

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 = 18)
ax[1].set_xlabel("Epoch")
ax[1].set_ylabel("Loss")

plt.show()
```

Kode ini digunakan untuk memvisualisasikan performa model selama pelatihan menggunakan matplotlib. Grafik yang dihasilkan menunjukkan akurasi dan loss untuk data pelatihan dan validasi pada setiap epoch. Dengan memplot ini, kita dapat memantau bagaimana model belajar dari waktu ke waktu dan mendeteksi masalah seperti overfitting.

Hasil grafik :



#### c. Menyalin Model

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

```
!cp /content/model.h5 "/content/gdrive/My Drive/Hanacaraka"
```

#### d. Load Model

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

```
from keras.models import load_model

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

### 3. Uji Coba

#### a. Uji Coba Menggunakan Gambar

Data yang digunakan untuk uji coba ini adalah data baru yang tidak terlibat sama sekali dalam proses training.

```
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/gdrive/MyDrive/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/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)

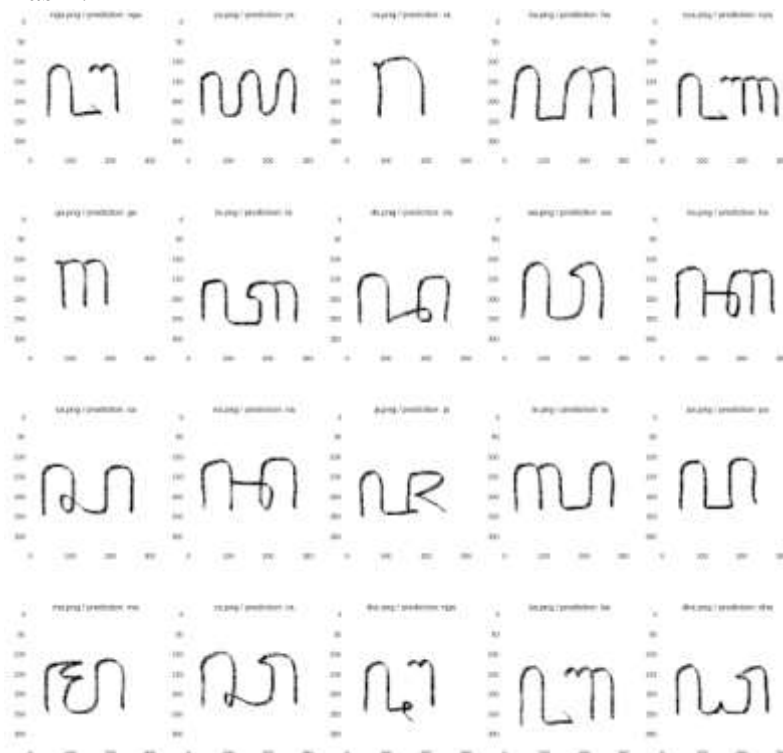
 # Gunakan variabel 'model' untuk prediksi
 result = model.predict(img)

 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)
```

Hasil :





b. Uji Coba Menggunakan Digital Handwriting

Data yang digunakan untuk uji coba ini adalah gambar digital handwriting hanacaraka.

```
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 benar)
model = load_model('/content/model.h5')

Fungsi untuk memprediksi kelas dari gambar
def classify_image(img_path):
 classes = [
 'ha', 'Ca', 'da', 'daa', 'ga', 'ha', 'ja', 'ka', 'ka', 'ka',
 'ka', 'nga', 'nga', 'pa', 'ra', 'sa', 'sa', 'tha', 'ta', '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/gdrive/MyDrive/images/ha.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()
```

Hasil :

1/1 [=====] - 0s 133ms/step  
Result: ha

Predicted: ha

