# Laporan Proyek Mini: Klasifikasi Aksara Jawa Tulisan Tangan Menggunakan CNN

# Pendahuluan

Proyek ini bertujuan untuk membangun sebuah model *deep learning* yang mampu mengenali dan mengklasifikasikan gambar aksara Jawa tulisan tangan. Kami menggunakan arsitektur Convolutional Neural Network (CNN) yang dibangun dari awal (from scratch) dengan framework TensorFlow/Keras. Dataset yang digunakan adalah kumpulan gambar aksara Jawa yang telah dipisahkan ke dalam folder sesuai dengan kelasnya. Laporan ini akan menjabarkan seluruh proses, mulai dari persiapan data, perancangan model, pelatihan, hingga evaluasi performa.

# Setup Awal Librari

## Selecting a Template (Heading 2)

*Penjelasan*: Langkah pertama adalah menyiapkan lingkungan kerja. Kami mengimpor semua library yang diperlukan dan mendefinisikan parameter-parameter utama dalam sebuah class Konfigurasi agar mudah diubah untuk eksperimen.

**import** tensorflow **as** tf

**from** tensorflow.keras **import** layers, models

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

**import** os

**from** sklearn.metrics **import** confusion\_matrix, ConfusionMatrixDisplay

**from** tensorflow.keras.callbacks **import** EarlyStopping, ReduceLROnPlateau

**from** tensorflow.keras.preprocessing **import** image

print('library berhasil di import')

library berhasil di import

## Konfig-konfig Awal Project

**class** Konfigurasi:

DATA\_PATH **=** "/kaggle/input/hanacaraka"

*# pengaturan gambar*

IMAGE\_SIZE **=** (64, 64)

BATCH\_SIZE **=** 32

*# jumlah epoch*

EPOCHS **=** 100

config **=** Konfigurasi()

## Memuat dan Mempersiapkan Dataset

*Penjelasan*: Kami menggunakan fungsi image\_dataset\_from\_directory dari Keras untuk memuat data. Fungsi ini sangat efisien karena secara otomatis membaca gambar dari sub-folder dan menetapkan nama folder sebagai label kelas. Kami juga membagi data menjadi 80% untuk pelatihan dan 20% untuk validasi.

*# menggunakan 80% data untuk training*

train\_ds **=** tf**.**keras**.**utils**.**image\_dataset\_from\_directory(

config**.**DATA\_PATH,

validation\_split**=**0.2,

subset**=**"training",

seed**=**123,

image\_size**=**config**.**IMAGE\_SIZE,

batch\_size**=**config**.**BATCH\_SIZE,

color\_mode**=**'grayscale'

)

*# menggunakan 20% data untuk validasi*

val\_ds **=** tf**.**keras**.**utils**.**image\_dataset\_from\_directory(

config**.**DATA\_PATH,

validation\_split**=**0.2,

subset**=**"validation",

seed**=**123,

image\_size**=**config**.**IMAGE\_SIZE,

batch\_size**=**config**.**BATCH\_SIZE,

color\_mode**=**'grayscale'

)

*# mendapatkan nama class dataset*

class\_names **=** train\_ds**.**class\_names

NUM\_CLASSES **=** len(class\_names)

print(f"Ditemukan {NUM\_CLASSES} kelas (aksara):")

print(class\_names)

Found 1562 files belonging to 20 classes.

Using 1250 files for training.

Found 1562 files belonging to 20 classes.

Using 312 files for validation.

Ditemukan 20 kelas (aksara):

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

## Visualisasi Data & Ekspolorasi

*Penjelasan*: Sebelum melatih model, penting untuk melihat seperti apa data yang kita gunakan. Sel ini akan menampilkan beberapa contoh gambar dari dataset pelatihan beserta labelnya.

# visualisasi beberapa contoh gambar

plt**.**figure(figsize**=**(10, 10))

**for** images, labels **in** train\_ds**.**take(1): *# Ambil satu batch pertama*

**for** i **in** range(9): *# Tampilkan 9 gambar pertama dari batch*

ax **=** plt**.**subplot(3, 3, i **+** 1)

*# Reshape dari (64, 64, 1) menjadi (64, 64) untuk ditampilkan*

plt**.**imshow(np**.**squeeze(images[i]**.**numpy()**.**astype("uint8")), cmap**=**'gray')

plt**.**title(class\_names[labels[i]])

plt**.**axis("off")

plt**.**show()

## Optimasi Performa Dataset

*Penjelasan*: Untuk mempercepat proses training, kami menggunakan metode .cache() dan .prefetch(). .cache() akan menyimpan data di memori setelah pertama kali dimuat, dan .prefetch() akan menyiapkan batch data berikutnya saat GPU sedang memproses batch saat ini.

AUTOTUNE **=** tf**.**data**.**AUTOTUNE

Identify applicable funding agency here. If none, delete this text box.

train\_ds **=** train\_ds**.**cache()**.**shuffle(1000)**.**prefetch(buffer\_size**=**AUTOTUNE)

val\_ds **=** val\_ds**.**cache()**.**prefetch(buffer\_size**=**AUTOTUNE)

print('optimasi selesai')

## Merancang Arsitektur CNN

*Penjelasan*: Kami membangun model CNN dari awal menggunakan Keras Sequential API. Arsitekturnya terdiri dari beberapa blok konvolusi untuk ekstraksi fitur dan lapisan Dense untuk klasifikasi. Kami juga menambahkan data augmentation untuk membuat model lebih robust dan Dropout untuk mencegah overfitting.

model **=** models**.**Sequential([

*# Lapisan Augmentasi & Normalisasi*

layers**.**Input(shape**=**(config**.**IMAGE\_SIZE[0], config**.**IMAGE\_SIZE[1], 1)),

layers**.**RandomFlip("horizontal"),

layers**.**RandomRotation(0.1),

layers**.**RandomZoom(0.1),

layers**.**Rescaling(1.**/**255),

*# Blok Konvolusi 1*

layers**.**Conv2D(64, (3, 3), padding**=**'same'),

layers**.**LeakyReLU(alpha**=**0.2), *# GANTI 'relu' dengan LeakyReLU*

layers**.**MaxPooling2D((2, 2)),

layers**.**BatchNormalization(),

*# Blok Konvolusi 2*

layers**.**Conv2D(128, (3, 3), padding**=**'same'),

layers**.**LeakyReLU(alpha**=**0.2), *# GANTI 'relu' dengan LeakyReLU*

layers**.**MaxPooling2D((2, 2)),

layers**.**BatchNormalization(),

*# Blok Konvolusi 3*

layers**.**Conv2D(256, (3, 3), padding**=**'same'),

layers**.**LeakyReLU(alpha**=**0.2), *# GANTI 'relu' dengan LeakyReLU*

layers**.**MaxPooling2D((2, 2)),

*# Lapisan Klasifikasi*

layers**.**Flatten(),

layers**.**Dense(256),

layers**.**LeakyReLU(alpha**=**0.2), *# GANTI 'relu' dengan LeakyReLU*

layers**.**Dropout(0.5),

layers**.**Dense(NUM\_CLASSES) *# Output logit (tanpa aktivasi)*

])

*# Kompilasi ulang dengan learning rate rendah*

optimizer\_kustom **=** tf**.**keras**.**optimizers**.**Adam(learning\_rate**=**1e-4)

model**.**compile(

optimizer**=**optimizer\_kustom,

loss**=**tf**.**keras**.**losses**.**SparseCategoricalCrossentropy(from\_logits**=True**),

metrics**=**['accuracy']

)

model**.**summary()

## Melatih model CNN

*Penjelasan*: Ini adalah tahap implementasi utama di mana kami melatih model menggunakan data training dan mengevaluasinya pada setiap epoch menggunakan data validasi. Proses ini akan memakan waktu tergantung pada hardware yang digunakan.

print("Memulai proses pelatihan model...")

early\_stopping **=** EarlyStopping(

monitor**=**'val\_loss',

patience**=**15,

restore\_best\_weights**=True**

)

reduce\_lr **=** ReduceLROnPlateau(

monitor**=**'val\_loss',

factor**=**0.2, *# Kurangi LR sebesar 80% (1 - 0.2)*

patience**=**5,

min\_lr**=**1e-6 *# Learning rate minimum*

)

history **=** model**.**fit(

train\_ds,

validation\_data**=**val\_ds,

epochs**=**config**.**EPOCHS,

callbacks**=**[early\_stopping]

)

print("Pelatihan model selesai.")

## Evaluasi Hasil

*Penjelasan*: Setelah training selesai, kami memvisualisasikan grafik akurasi dan loss untuk data training dan validasi. Grafik ini sangat penting untuk menganalisis apakah model mengalami *overfitting* atau *underfitting*.

# ambil data

acc **=** history**.**history['accuracy']

val\_acc **=** history**.**history['val\_accuracy']

loss **=** history**.**history['loss']

val\_loss **=** history**.**history['val\_loss']

epochs\_range **=** range(len(acc))

*# plot*

plt**.**figure(figsize**=**(15, 6))

plt**.**subplot(1, 2, 1)

plt**.**plot(epochs\_range, acc, label**=**'Training Accuracy')

plt**.**plot(epochs\_range, val\_acc, label**=**'Validation Accuracy')

plt**.**legend(loc**=**'lower right')

plt**.**title('Training and Validation Accuracy')

plt**.**subplot(1, 2, 2)

plt**.**plot(epochs\_range, loss, label**=**'Training Loss')

plt**.**plot(epochs\_range, val\_loss, label**=**'Validation Loss')

plt**.**legend(loc**=**'upper right')

plt**.**title('Traning and validation loss')

plt**.**show()

## Analisis Performa dengan Confusion Matrix

*Penjelasan*: Untuk memahami performa model lebih dalam, kami menggunakan Confusion Matrix. Matriks ini menunjukkan kelas mana saja yang sering salah diklasifikasikan oleh model.

In [133]:

*# --- Membuat Prediksi pada Data Validasi ---*

y\_pred\_probs **=** model**.**predict(val\_ds)

y\_pred **=** np**.**argmax(y\_pred\_probs, axis**=**1)

*# --- Mendapatkan Label Asli ---*

y\_true **=** np**.**concatenate([y **for** x, y **in** val\_ds], axis**=**0)

*# --- Menghitung dan Menampilkan Confusion Matrix ---*

cm **=** confusion\_matrix(y\_true, y\_pred)

disp **=** ConfusionMatrixDisplay(confusion\_matrix**=**cm, display\_labels**=**class\_names)

fig, ax **=** plt**.**subplots(figsize**=**(12, 12))

disp**.**plot(ax**=**ax, cmap**=**plt**.**cm**.**Blues, xticks\_rotation**=**'vertical')

plt**.**title("Confusion Matrix")

plt**.**show()

**def** prediksi\_gambar\_tunggal(img\_path):

**try**:

*# Memuat gambar menggunakan Keras*

img **=** tf**.**keras**.**utils**.**load\_img(

img\_path,

target\_size**=**config**.**IMAGE\_SIZE,

color\_mode**=**'grayscale'

)

*# Mengubah gambar menjadi array NumPy*

img\_array **=** tf**.**keras**.**utils**.**img\_to\_array(img)

*# Menambahkan dimensi batch (menjadi 1, 64, 64, 1)*

img\_array **=** tf**.**expand\_dims(img\_array, 0)

*# Melakukan prediksi*

predictions **=** model**.**predict(img\_array)

*# Mendapatkan kelas dengan probabilitas tertinggi*

predicted\_class\_index **=** np**.**argmax(predictions[0])

predicted\_class\_name **=** class\_names[predicted\_class\_index]

confidence **=** 100 **\*** np**.**max(predictions[0])

*# Menampilkan hasil*

plt**.**figure(figsize**=**(6, 6))

plt**.**imshow(np**.**squeeze(img\_array[0]), cmap**=**'gray')

plt**.**title(f"Prediksi: {predicted\_class\_name}\nKeyakinan: {confidence:.2f}%")

plt**.**axis("off")

plt**.**show()

**except** FileNotFoundError:

print(f"ERROR: File tidak ditemukan di '{img\_path}'.")

print("Pastikan path yang Anda masukkan sudah benar.")

**except** Exception **as** e:

print(f"Terjadi error: {e}")

*# --- CONTOH PENGGUNAAN ---*

*# Ganti path di bawah ini dengan path ke salah satu gambar dari dataset Anda*

*# Contoh path di Kaggle: '/kaggle/input/hanacaraka/ha/ha\_1.png'*

PATH\_GAMBAR\_UJI **=** "/kaggle/input/hanacaraka/la/4693403521314974166\_base64\_27.png"