

DISKUSI PROYEK
Pengenalan Pola Aksara Jawa
Mata Kuliah Pengenalan Pola

Dosen Matakuliah :

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

Nama Anggota Kelompok :

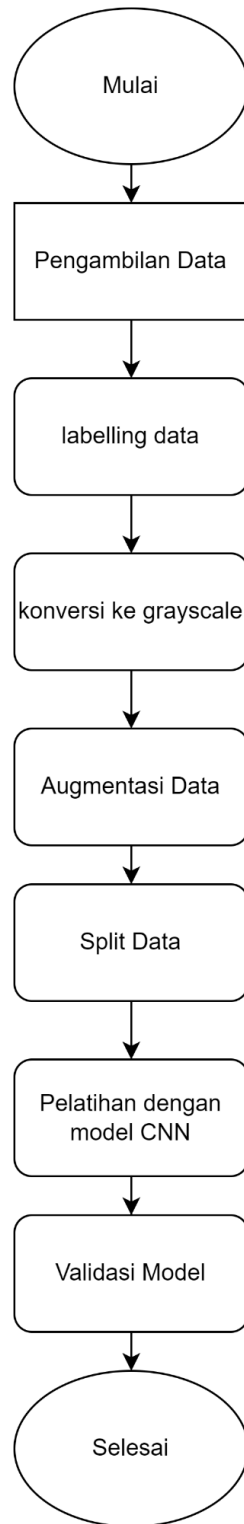
- | | |
|---------------------------------|---------------|
| 1. Muhammad Nafid Zanis | (32602100083) |
| 2. Muhammad Wahyu Syaiful Anaam | (32602100089) |
| 3. Raehan Supriantono Putri | (32602100108) |
| 4. Octabriana Anggun R. | (32602100107) |

1. Perencanaan Pengambilan data

Pengambilan dataset Aksara Jawa mengambil dari situs Kaggle.com <https://www.kaggle.com/datasets/phiard/aksara-jawa> dengan mencari keyword “Aksara Jawa” di menu dataset. Bentuk format gambar berupa .JPG, .JPEG, atau .PNG

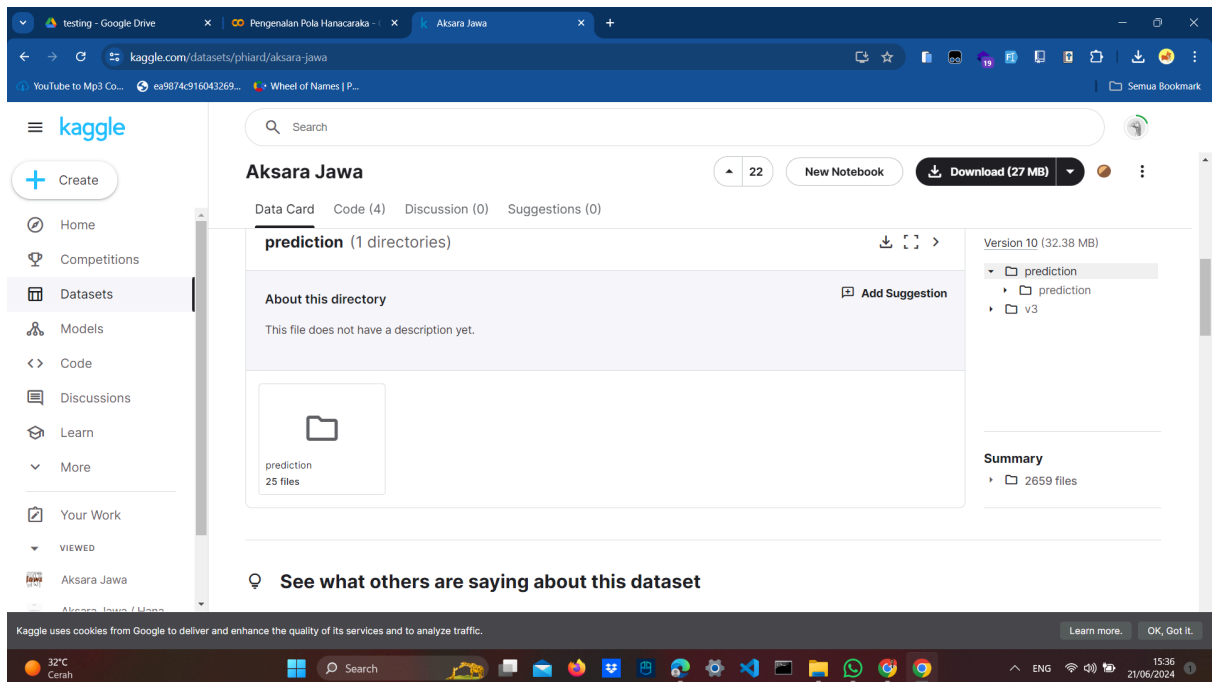
2. Flowchart pemrosesan data

Flowchart merupakan diagram alir yang menampilkan langkah-langkah dan keputusan untuk melakukan sebuah proses dari suatu program. Rangkaian flowchart harus urut dari atas ke bawah. Pemrosesan data pada pengenalan pola Aksara Jawa dimulai dari mengolah *raw data* menjadi data yang siap digunakan, karena kami menggunakan algoritma CNN, untuk dapat menggunakan CNN maka raw data harus diubah sesuai dengan format yang tepat. Gambar dibawah merupakan *flowchart* pengenalan pola Aksara Jawa menggunakan metode CNN (Convolutional Neural Network).



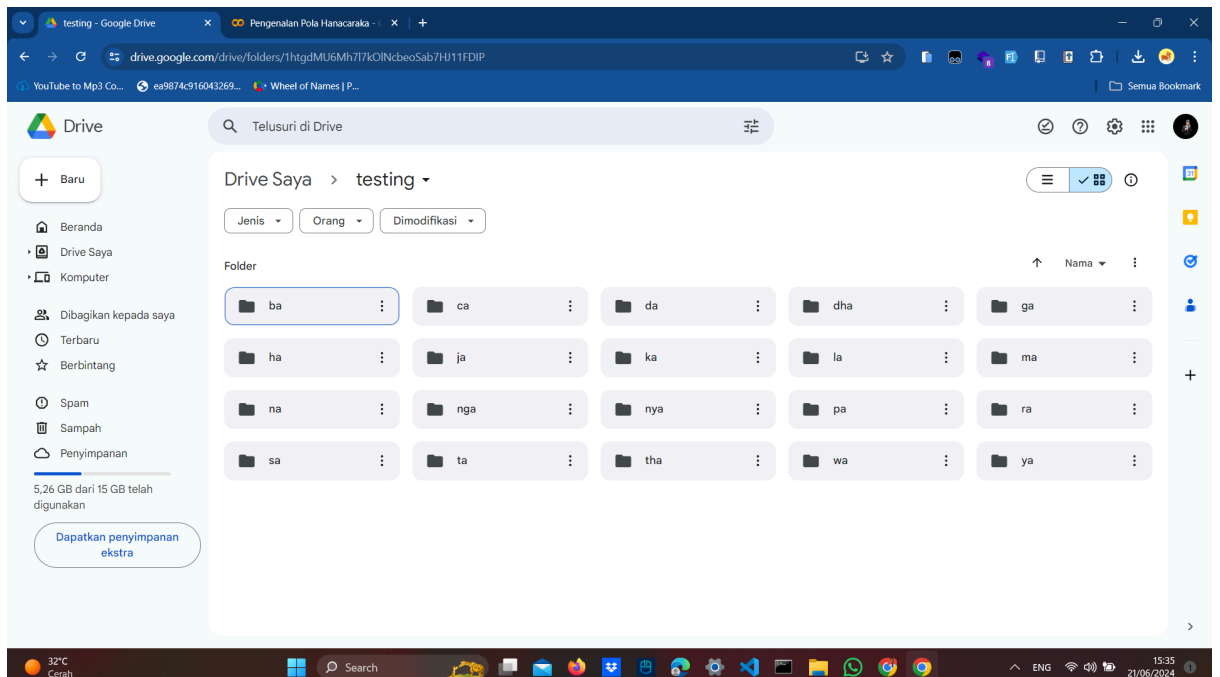
Gambar 1.1 Flowchart

1. Pengambilan Data Input



Gambar 1.2 Mencari dataset


Pada gambar 1.2 dilakukan pencarian dataset Aksara Jawa melalui situs Kaggle.com, datacard yang kami dapat bisa diperoleh di <https://www.kaggle.com/datasets/phiard/aksara-jawa>



Gambar 1.3 Mengunduh dataset

Pada gambar 1.3 setelah mencari dataset di Kaggle.com selanjutnya mengunduh datasetnya untuk disimpan di repositori google drive agar memudahkan Google Collaboratory untuk mencari sumbernya

```
from google.colab import drive, files
drive.mount('/content/gdrive/')
```

 Mounted at /content/gdrive/

```
import os
import cv2
import numpy as np
import matplotlib.pyplot as plt

images = []
labels = []
label_to_images = {}

# direktori dataset
data_dir =  '/content/gdrive/MyDrive/testing'
```

Gambar 1.4 Mengambil dataset dari Google Drive

Pada gambar 1.4 merupakan proses untuk menyambungkan google collaboratory ke google drive, penyambungan ini untuk mempermudah google collaboratory membaca dataset yang sudah diunduh dan di upload ke google drive. kita perlu mengimpor beberapa library diantaranya library 'os' untuk berinteraksi dengan sistem operasi seperti manipulasi file dan direktori, library 'cv2' untuk pemrosesan gambar, library 'numpy' untuk operasi numerik dengan array, dan library 'matplotlib.pyplot' untuk membuat plot dan visualisasi data.

2. Labelling Data

Input

```
# Periksa apakah label_dir adalah direktori
if os.path.isdir(label_dir):
    label_images = []
    for image_file in os.listdir(label_dir):
        image_path = os.path.join(label_dir, image_file)
        # Periksa apakah file adalah file gambar yang didukung
        if os.path.isfile(image_path) and image_file.lower().endswith(('.png', '.jpg', '.jpeg')):
            image = cv2.imread(image_path)
            if image is not None:
                image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
                images.append(image_rgb)
                labels.append(label)
                label_images.append(image_rgb)
            else:
                print(f"Warning: Gambar {image_path} tidak bisa dimuat.")
        label_to_images[label] = label_images

total_images = len(images)
total_unique_labels = len(label_to_images)

print("Total Gambar: ", total_images)
print("Total Label: ", total_unique_labels)

# Periksa apakah ada label unik sebelum plotting
if total_unique_labels > 0:
    fig, axes = plt.subplots(1, total_unique_labels, figsize=(15, 5))

    # Jika hanya ada satu label unik, axes adalah satu AxesSubplot, bukan array
    if total_unique_labels == 1:
        axes = [axes]

    # Plot images
    for i, (label, label_images) in enumerate(label_to_images.items()):
        axes[i].imshow(label_images[0])
        axes[i].set_title(label)
        axes[i].axis('off')

    plt.tight_layout()
    plt.show()
else:
    print("Tidak ada label unik yang ditemukan.")
```

Gambar 1.5 proses labelling data

Pada gambar 1.5 dilakukan proses memeriksa gambar dalam bentuk PNG, JPG, atau JPEG untuk dimasukkan ke labelling data, jika format diluar dari ketentuan maka akan muncul tulisan di output “Gambar {image} tidak bisa dimuat”.

Output



Gambar 1.6 output proses labelling data

Pada gambar 1.6 yaitu output dari hasil labelling data, dari total 1299 data gambar dalam bentuk JPG, PNG, atau JPEG yang telah diperiksa dan diproses terdapat 20 total label sesuai dengan total huruf aksara jawa yaitu 20 data label.

3. Preprocessing

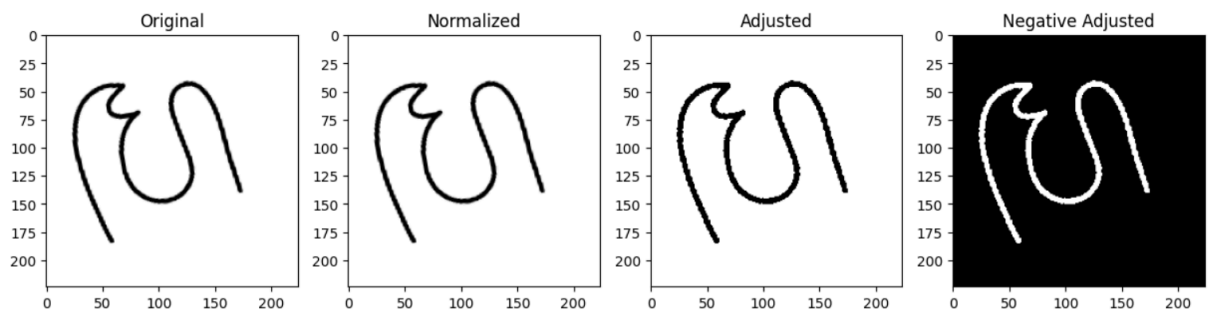
Input

```
def preprocess_image(image):  
    # Normalized image  
    normalized_image = image.astype(np.float32) / 255.0  
  
    threshold = 0.87  
  
    # Mengganti nilai di atas threshold menjadi putih dan di bawah threshold menjadi hitam  
    adjusted_image_array = np.where(normalized_image > threshold, 1.0, 0.0)  
  
    # Membuat gambar negatif  
    negative_adjusted_image = 1.0 - adjusted_image_array  
  
    # Pastikan nilai negatif terkclip untuk menghindari masalah dengan imshow  
    negative_adjusted_image = np.clip(negative_adjusted_image, 0.0, 1.0)  
  
    # Konversi kembali gambar negatif ke uint8  
    negative_adjusted_image = (negative_adjusted_image * 255).astype(np.uint8)  
  
    return normalized_image, adjusted_image_array, negative_adjusted_image  
  
# Memilih satu contoh citra dari data yang diproses  
sample_image_index = 600 # ganti dengan indeks citra yang ingin anda gunakan  
sample_image_data = (images[sample_image_index], *preprocess_image(images[sample_image_index]))  
  
# Menampilkan contoh citra pada seluruh tahapan pra pemrosesan  
fig, axes = plt.subplots(1, 4, figsize=(15, 5))  
axes[0].imshow(cv2.cvtColor(sample_image_data[0], cv2.COLOR_BGR2RGB))  
axes[0].set_title('Original')  
axes[1].imshow(sample_image_data[1], cmap='gray')  
axes[1].set_title('Normalized')  
axes[2].imshow(sample_image_data[2], cmap='gray')  
axes[2].set_title('Adjusted')  
axes[3].imshow(sample_image_data[3], cmap='gray')  
axes[3].set_title('Negative Adjusted')  
plt.show()
```

Gambar 1.7 pemrosesan citra

Pada gambar 1.7 dilakukan tahap normalisasi citra dengan membagi nilai piksel dengan 255.0 dalam rentang [0,1]. Selanjutnya dilakukan thresholding dengan metode binerisasi yang mengubah citra menjadi hitam putih

Output



Gambar 1.8 Output pemrosesan citra

Pada gambar 1.8 merupakan output pemrosesan citra, dimulai dari kiri original merupakan citra asli sebelum diproses, selanjutnya normalized yaitu citra setelah dinormalisasi, kemudian adjusted yaitu citra setelah thresholding, dan yang paling kanan Negative Adjusted yaitu citra setelah membuat negatif yang kemudian dikonversi kembali ke format 'uint8'.

Input

```
# Mendapatkan jumlah total gambar dan label unik
total_images = len(images)
total_unique_labels = len(np.unique(labels))

print("Total Gambar: ", total_images)
print("Total Label: ", total_unique_labels)

# Menampilkan contoh gambar dari setiap label
fig, axes = plt.subplots(nrows=1, ncols=total_unique_labels, figsize=(15, 5))

for i, (label, label_images) in enumerate(label_to_images.items()):
    ax = axes[i]
    if len(label_images) > 50:
        ax.imshow(label_images[50], cmap='gray')
        ax.set_title(label)
    else:
        ax.set_title(f"{label}\n(Not enough images)")
        ax.axis('off')

plt.tight_layout()
plt.show()
```

Gambar 1.9 pengambilan gambar

Output



Gambar 1.10 Sampel gambar setelah dinormalisasi

4. Pelatihan Model CNN

```
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers, models
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
```

Gambar 1.11 Import Library

Pada gambar 1.11 sebelum menggunakan algoritma CNN, perlu mengimpor beberapa library diantaranya 'tensorflow' untuk kecerdasan buaatannya, library 'tensorflow.keras import layers, models' untuk pelatihan neural network, library 'sklearn.metrics import confusion matrix' yang digunakan untuk analisis data setelah modelling.

```
# Fungsi membangun model
def build_model(input_shape, num_classes):
    model = models.Sequential([
        layers.Conv2D(32, (3, 3), activation='relu', input_shape=input_shape),
        layers.MaxPooling2D((2, 2)),
        layers.Conv2D(64, (3, 3), activation='relu'),
        layers.MaxPooling2D((2, 2)),
        layers.Conv2D(64, (3, 3), activation='relu'),
        layers.Flatten(),
        layers.Dense(64, activation='relu'),
        layers.Dense(num_classes, activation='softmax')
    ])
    model.compile(optimizer='adam',
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])
    return model

# ukuran gambar yang akan digunakan
input_shape = images[0].shape

# jumlah kelas
num_classes = len(label_to_images)

# membangun model
model = build_model(input_shape, num_classes)

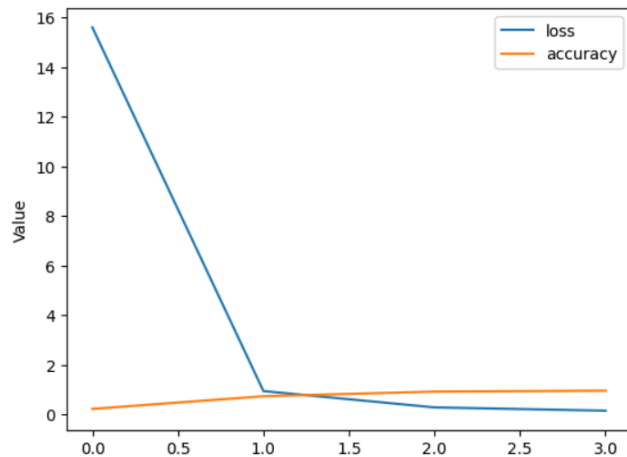
# mengkonversi label menjadi bilangan bulat
label_to_index = {label: i for i, label in enumerate(label_to_images)}
labels_index = np.array([label_to_index[label] for label in labels])

# melatih model
history = model.fit(np.array(images), labels_index, epochs=4, batch_size=32, validation_split=0.2)

# menghitung akurasi
accuracy = history.history['accuracy'][-1]
print("Training accuracy: ", accuracy)
```

Gambar 1.12 Modelling CNN


```
Epoch 1/4  
33/33 [=====] - 38s 1s/step - loss: 15.5902 - accuracy: 0.2185 - val_loss: 7.0718 - val_accuracy: 0.0000e+00  
Epoch 2/4  
33/33 [=====] - 33s 1s/step - loss: 0.9355 - accuracy: 0.7286 - val_loss: 10.5861 - val_accuracy: 0.0000e+00  
Epoch 3/4  
33/33 [=====] - 35s 1s/step - loss: 0.2784 - accuracy: 0.9153 - val_loss: 16.9456 - val_accuracy: 0.0000e+00  
Epoch 4/4  
33/33 [=====] - 32s 987ms/step - loss: 0.1453 - accuracy: 0.9509 - val_loss: 24.7985 - val_accuracy: 0.0000e+00  
Training accuracy: 0.9509143233299255
```



Gambar 1.13 Output Pelatihan

Pada gambar 1.13 menunjukkan hasil evaluasi dan pelatihan model menggunakan grafik matplotlib dengan menunjukkan jumlah epochs sebanyak 4. Data menunjukkan nilai akurasi dari epoch 1 sampai dengan epoch 4 mengalami peningkatan pada epoch 4 dengan akurasi 0.9509 yang artinya semakin tinggi nilai akurasi, semakin baik kinerja model dalam melakukan prediksi.

```

# menonaktifkan warning untuk pembagian oleh nol
warnings.filterwarnings("ignore", category=RuntimeWarning)

# predict labels for the validation data
predicted_labels = model.predict(np.array(images))
# convert predicted labels to class indices
predicted_indices = np.argmax(predicted_labels, axis=1)
# compute confusion matrix
conf_matrix = confusion_matrix(labels_index, predicted_indices)
# calculate classification metrics
true_positives = np.diag(conf_matrix)
false_positives = np.sum(conf_matrix, axis=0) - true_positives
false_negatives = np.sum(conf_matrix, axis=1) - true_positives
# calculate precision, recall, and f1-score
precision = true_positives / (true_positives + false_positives)
recall = true_positives / (true_positives + false_negatives)

# handle division by zero
precision = np.nan_to_num(precision)
recall = np.nan_to_num(recall)

# calculate f1-score
f1_score = 2 * (precision * recall) / (precision + recall)

# handle division by zero in f1-score calculation
f1_score = np.nan_to_num(f1_score, nan=0)

# average metrics across classes
avg_precision = np.mean(precision)
avg_recall = np.mean(recall)
avg_f1_score = np.mean(f1_score)

# calculate accuracy
accuracy = accuracy_score(labels_index, predicted_indices)

```

Gambar 1.14 Source code evaluasi model

```

41/41 [=====] - 12s 286ms/step
Classification Report:
Accuracy:  0.7821401077752117
Average precision:  0.6457013170322086
Average Recall:  0.7824725274725275
Average F1-Score:  0.7025896641927802

```

Gambar 1.15 Output evaluasi model

Pada gambar 1.15 menunjukkan hasil prediksi dan evaluasi model. Data menunjukkan akurasi model memprediksi sebanyak 0.7821401077752117 atau 78.21%, Average precision 0.6457013170322086 atau 64.57% , Average recall

sebanyak 0.7824725274725275 atau 78.25%, dan Average F1-Score sebanyak 0.7025896641927802 atau 70.26%

```
# memprediksi label untuk data validasi
# memprediksi probabilitas untuk data validasi
predicted_probabilities = model.predict(np.array(images))

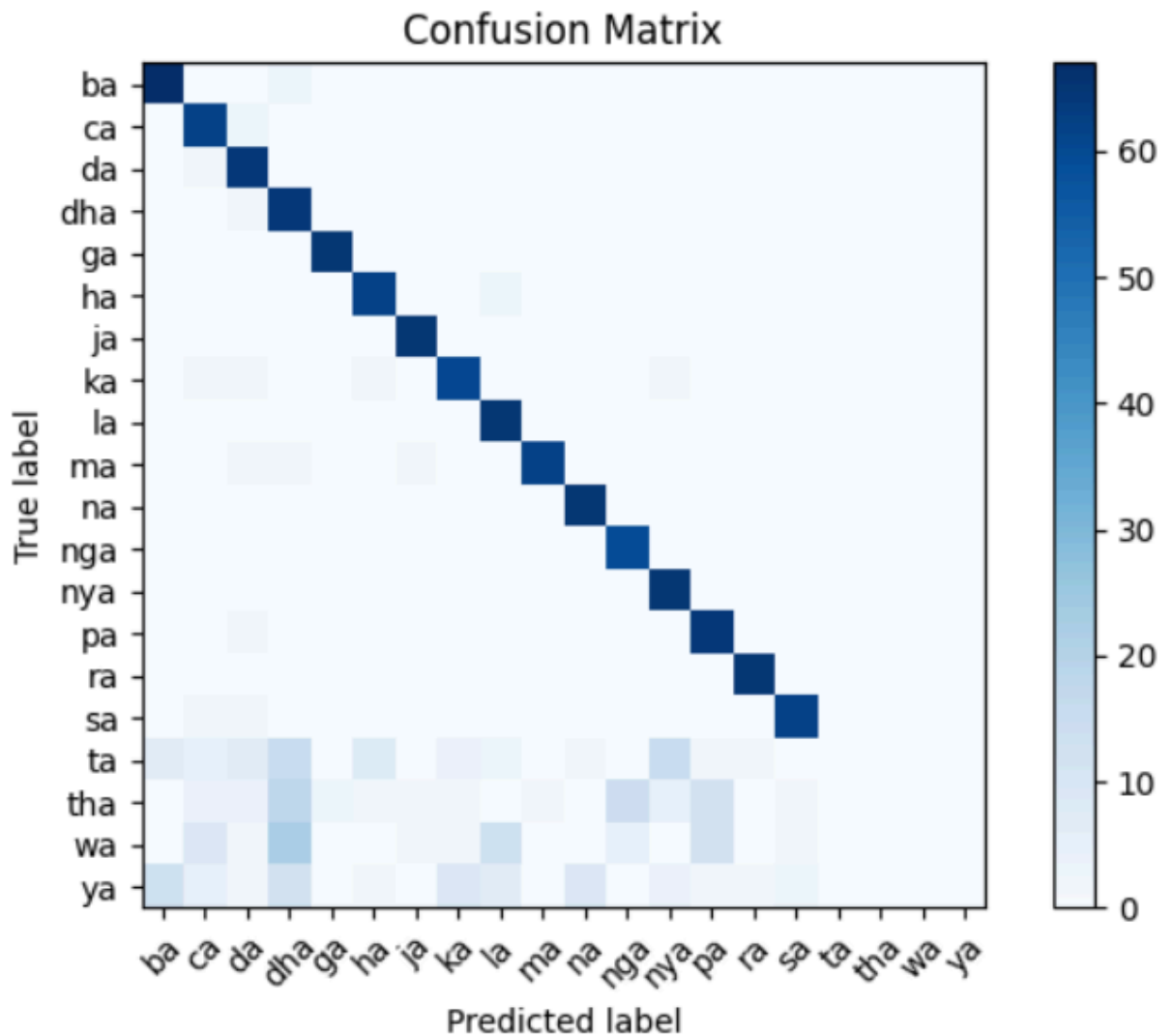
# mengambil label dengan probabilitas tertinggi sebagai prediksi
predicted_labels = np.argmax(predicted_probabilities, axis=1)

# membuat confusion matrix
cm = confusion_matrix(labels_index, predicted_labels)

# menampilkan confusion matrix
print("Confusion Matrix: ")
print(cm)

# Plot confusion matrix
plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)
plt.title('Confusion Matrix')
plt.colorbar()
tick_marks = np.arange(len(label_to_index))
plt.xticks(tick_marks, label_to_index, rotation=45)
plt.yticks(tick_marks, label_to_index)
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.tight_layout()
plt.show()
```

Gambar 1.16 source code prediksi validasi label data



Gambar 1.17 Output prediksi validasi label data

Pada gambar 1.17 menunjukkan hasil prediksi validasi dari label data menunjukkan semakin gelap warna biru menunjukkan nilai yang lebih tinggi dalam confusion matrix yang berarti banyak prediksi yang lebih benar. Jika warna biru lebih terang menunjukkan nilai rendah dalam confusion matrix artinya terdapat banyak kesalahan prediksi sedikit prediksi yang benar untuk kombinasi tertentu dari label.

```

import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

# fungsi mengonversi label kedalam bentuk numerik
def convert_labels_to_numeric(labels):
    unique_labels = np.unique(labels)
    label_to_index = {label: i for i, label in enumerate(unique_labels)}
    numeric_labels = [label_to_index[label] for label in labels]
    return np.array(numeric_labels)

# melatih model dengan subset dataset tertentu
def train_model_with_subset(images, labels, subset_size, model):
    # ambil subset dataset
    num_samples = int(len(images) * subset_size)
    subset_images = images[:num_samples]
    subset_labels = labels[:num_samples]

    # bagi data menjadi data pelatihan dan data validasi
    x_train, x_val, y_train, y_val = train_test_split(subset_images, subset_labels, test_size=0.2, random_state=42)

    # latih model
    history = model.fit(x_train, y_train, epochs=4, batch_size=32, validation_data=(x_val, y_val))

    # ambil akurasi dari history
    accuracy = history.history['accuracy'][-1]
    return accuracy

# fungsi membagi dataset menjadi subset
def divide_dataset(images, labels, num_subsets):
    subset_sizes = np.linspace(0.1, 1.0, num_subsets) # mulai dari 10% hingga 100% dataset
    accuracies = []

    # lakukan pelatihan utk setiap subset
    for subset_size in subset_sizes:
        # lakukan pelatihan dan evaluasi
        accuracy = train_model_with_subset(images, labels, subset_size, model)
        accuracies.append(accuracy)

    return subset_sizes, accuracies

# contoh penggunaan
num_subsets = 10 # dataset menjadi 10 subset

# memuat gambar ke dalam numpy arrays
images_np = np.array(images)
labels_np = np.array(labels)

# konversi label ke dalam bentuk numerik dan dimuat kedalam numpy arrays
numeric_labels = convert_labels_to_numeric(labels_np)

subset_sizes, accuracies = divide_dataset(images_np, numeric_labels, num_subsets)

# plotting hasil
plt.plot(subset_sizes, accuracies, marker='o')
plt.title('Pengaruh ukuran dataset terhadap kinerja model')
plt.xlabel('ukuran dataset sebagai pecahan dari dataset penuh')
plt.ylabel('akurasi model')
plt.grid(True)
plt.show()

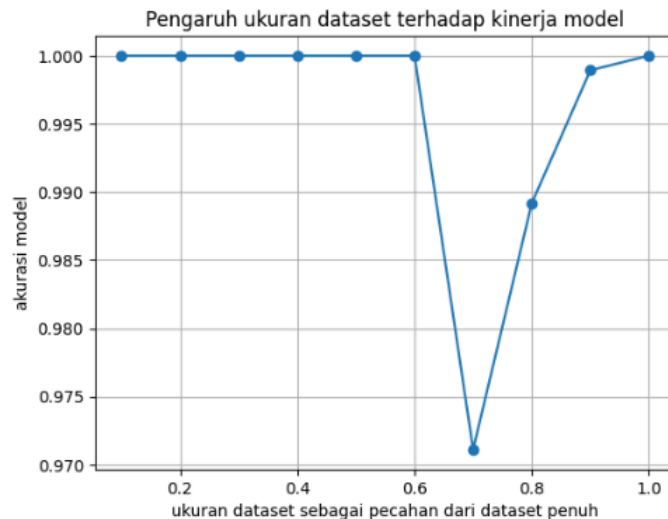
```

Gambar 1.18 Source code ukuran dan akurasi model

```

Epoch 3/4
26/26 [=====] - 26s 991ms/step - loss: 0.0429 - accuracy: 0.9880 - val_loss: 0.2920 - val_accuracy: 0.9183
Epoch 4/4
26/26 [=====] - 27s 1s/step - loss: 0.0376 - accuracy: 0.9892 - val_loss: 0.3140 - val_accuracy: 0.9279
Epoch 1/4
30/30 [=====] - 29s 959ms/step - loss: 1.0587 - accuracy: 0.8203 - val_loss: 0.2913 - val_accuracy: 0.9145
Epoch 2/4
30/30 [=====] - 28s 922ms/step - loss: 0.0844 - accuracy: 0.9754 - val_loss: 0.3015 - val_accuracy: 0.9145
Epoch 3/4
30/30 [=====] - 29s 966ms/step - loss: 0.0236 - accuracy: 0.9925 - val_loss: 0.2464 - val_accuracy: 0.9444
Epoch 4/4
30/30 [=====] - 31s 1s/step - loss: 0.0057 - accuracy: 0.9989 - val_loss: 0.2410 - val_accuracy: 0.9402
Epoch 1/4
33/33 [=====] - 33s 1s/step - loss: 0.7297 - accuracy: 0.8874 - val_loss: 0.3101 - val_accuracy: 0.9192
Epoch 2/4
33/33 [=====] - 36s 1s/step - loss: 0.0582 - accuracy: 0.9856 - val_loss: 0.2475 - val_accuracy: 0.9346
Epoch 3/4
33/33 [=====] - 35s 1s/step - loss: 0.0117 - accuracy: 0.9981 - val_loss: 0.2560 - val_accuracy: 0.9308
Epoch 4/4
33/33 [=====] - 32s 988ms/step - loss: 0.0029 - accuracy: 1.0000 - val_loss: 0.2529 - val_accuracy: 0.9346

```



Gambar 1.19 Output ukuran dan akurasi model

Pada gambar 1.19 menunjukkan hasil prediksi ukuran dan akurasi model. Pada sumbu X merupakan ukuran dataset sebagai pecahan dari dataset penuh yang digunakan dalam pelatihan model mulai dari 0.1 (10%) hingga 1.0 (100%). Pada sumbu Y menunjukkan akurasi model setelah dilatih dengan subset dataset yang bersangkutan dengan nilai akurasi berkisar dari 0.970 hingga 1.000. Dari ukuran subset 10% hingga 60% akurasi model tetap sangat tinggi dan mendekati 1.000, hal ini menunjukkan bahwa model mampu belajar dengan sangat baik bahkan dengan subset dataset yang relatif kecil.

Ada penurunan drastis dalam akurasi ketika ukuran subset mencapai sekitar 60% dari dataset penuh yaitu akurasi turun ke sekitar 0.970, ini mungkin menunjukkan ada variabilitas dalam data subset atau problem lain seperti overfitting dan underfitting pada subset. Setelah penurunan tajam, akurasi kembali meningkat disaat ukuran subset mencapai 80% hingga 100%. Akurasi akhirnya mendekati nilai awal sekitar 1.000 pada ukuran subset 100%

