

DOKUMEN PROYEK

12S3205 – PENAMBANGAN DATA

DiTenun – *Ulos Image Classification Using Convolutional Neural Networks (CNN)*

Disusun Oleh:

Kelompok 10

12S21047	Elshaday Prida Simamora
12S21048	Nessy Pentasonia Pangaribuan
12S21049	Jesika Audina Purba



**PROGRAM STUDI SARJANA SISTEM INFORMASI
FAKULTAS INFORMATIKA DAN TEKNIK ELEKTRO
INSTITUT TEKNOLOGI DEL
TAHUN 2024/2025**

DAFTAR ISI

DAFTAR ISI.....	2
DAFTAR TABEL.....	4
DAFTAR GAMBAR	5
BAB 1 PENDAHULUAN	6
1.1 Latar Belakang	6
1.2 Tujuan.....	7
1.3 Manfaat.....	7
1.4 Ruang Lingkup.....	8
BAB 2 STUDI LITERATUR	10
2.1 <i>Convolutional Neural Networks (CNN)</i>	10
2.2 Penerapan CNN dalam Klasifikasi Gambar	10
BAB 3 METODE PENELITIAN	12
3. 1 <i>Cross-Industry Strandard Process for Data Mining (CRISP-DM)</i>	12
3.1.1 <i>Business Understanding</i>	12
3.1.2 <i>Data Understanding</i>	12
3.1.3 <i>Data Preparation</i>	37
3.1.4 <i>Modeling</i>	46
3.1.6 <i>Deployment</i>	57
3.2 <i>Timeline</i>	63
BAB 4 HASIL PENGUJIAN.....	65
4.1 Hasil Pelatihan Model CNN	65
BAB 5 ANALISIS	67
5.1 Analisis Hasil Evaluasi Model.....	67
5.1.1 Akurasi Model	67
5.1.2 <i>Precision, Recall, dan F1-Score</i>	67
5.2 Kelebihan Model CNN	67
5.2.1 Kemampuan untuk Mendeteksi Fitur Visual	67
5.2.3 Fleksibilitas dan Skalabilitas.....	68
5.2.4 Kemudahan Implementasi	68

5.3 Keterbatasan Model CNN.....	68
5.3.1 Kualitas Gambar	68
5.3.2 Variasi dalam Motif	68
5.3.3 Waktu Latih dan Sumber Daya.....	68
5.3.4 <i>Overfitting</i>	68
5.4 Saran untuk Penelitian Selanjutnya	69
BAB 6 KESIMPULAN.....	70
DAFTAR REFERENSI	71
BAB 7 PEMBAGIAN TUGAS	72

DAFTAR TABEL

Tabel 3-1 Kode Program untuk Mengakses Data	13
Tabel 3-2 Kode Program untuk Jumlah Kategori	14
Tabel 3-3 Kode Program untuk Deskripsi Data.....	15
Tabel 3-4 Kode Program untuk Karakteristik Data	17
Tabel 3-5 Kode Program untuk Keterkaitan Antar Data	19
Tabel 3-6 Kode Program untuk Validasi Data (Ukuran Data)	24
Tabel 3-7 Kode Program Untuk Validasi (Deskripsi Statistik Atribut).....	26
Tabel 3-8 Kode Program untuk Validasi Data (Relasi Antar Atribut)	28
Tabel 3-9 Kode Program untuk Visualisasi Data	32
Tabel 3-10 Kode Program untuk Memilih dan Memilah Data	37
Tabel 3-11 Kode Program untuk Membersihkan Data	39
Tabel 3-12 Kode Program untuk Mengkonstruksi Data	42
Tabel 3-13 Kode Program untuk Mengintegrasikan Data	44
Tabel 3-14 Kode Program untuk Membangun Arsitektur CNN.....	46
Tabel 3-15 Kode Program untuk Mendefinisikan Arsitektur Model CNN	48
Tabel 3-16 Kode Program untuk Melatih Model CNN	49
Tabel 3-17 Kode Program untuk Melatih Model CNN (2).....	49
Tabel 3-18 Kode Program untuk Memuat Data Uji.....	50
Tabel 3-19 Kode Program untuk Membuat Plot Hasil Data Uji.....	50
Tabel 3-20 Kode Program untuk Menyaring Klasifikasi untuk Gambar Ulos	52
Tabel 3-21 Hasil Evaluasi Model	53
Tabel 3-22 Kode Program Untuk Membuat Matriks Hasil Akhir	54
Tabel 3-23 Kode Program untuk Menyimpan Model.....	57
Tabel 3-24 Kode Program untuk Membangun Streamlit.....	58
Tabel 3-25 <i>Timeline</i>	63

DAFTAR GAMBAR

Gambar 3-1 Struktur Dataset	13
Gambar 3-2 Sub-Folder Kategori Ulos.....	13
Gambar 3-3 Informasi Jumlah Kategori dan Gambar.....	15
Gambar 3-4 Hasil Deskripsi Data	17
Gambar 3-5 Hasil Karakteristik Data	19
Gambar 3-6 Hasil Keterkaitan Antar Data.....	23
Gambar 3-7 Visualisasi Keterkaitan Antar Data (ANOVA)	23
Gambar 3-8 Visualisasi Keterkaitan Antar Data (Chi-Square).....	24
Gambar 3-9 Hasil Validasi Data untuk Ukuran Data	26
Gambar 3-10 Hasil Validasi Data untuk Deskripsi Statistik Atribut	28
Gambar 3-11 Hasil Relasi Antar Atribut (Heatmap Korelasi Antar Atribut Numerik).....	31
Gambar 3-12 Hasil Relasi Antar Atribut (<i>Cross-Tabulation</i> Antara Kategori dan Format File). 32	
Gambar 3-13 Visualisasi Distribusi Ukuran File.....	35
Gambar 3-14 Visualisasi Boxplot Ukuran File.....	35
Gambar 3-15 Visualisasi Jumlah File per Kategori.....	36
Gambar 3-16 Visualisasi Distribusi Format File	36
Gambar 3-17 Hasil Plot Data Uji.....	51
Gambar 3-18 Confusion Matrix.....	56
Gambar 3-19 Tampilan Streamlit	63

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Indonesia dikenal sebagai negara yang kaya akan seni dan budaya. Salah satu unsur budaya tersebut adalah menenun. Tenun merupakan kerajinan tangan berupa kain yang dibuat dari benang (katun, sutra, dan lain-lain) dengan cara memasukkan benang secara melintang pada suatu alat dan dibuat di daerah seperti Sumatera, Jawa, Palembang, dan Kalimantan [1]. Dengan berdasar kepada produk tenun dari berbagai daerah, penelitian ini mengkaji hasil tenun daerah Sumatera Utara, yaitu Ulos. Terdapat sebuah organisasi DiTenun yang didedikasikan untuk pelestarian dan promosi kain tenun tradisional Indonesia, khususnya Ulos, sebagai warisan budaya masyarakat Batak. Motif ulos yang digunakan oleh penenun saat ini adalah motif ulos yang lalu dan masih digunakan hingga saat ini [1]. Aplikasi Ditenun memiliki beberapa modul *generate*, yaitu modul Generate Kristik dan modul Ulos Editor. Terdapat juga model Verifikasi Kualitas Gambar dan Model Klasifikasi Ulos yang akan diintegrasikan dengan aplikasi DiTenun sebagai modul baru [2].

Untuk itu, perlu ada upaya untuk melestarikan dan mengenalkan lebih luas lagi keanekaragaman budaya Indonesia, salah satunya melalui pemanfaatan teknologi. Proyek DiTenun – *Ulos Image Classification* bertujuan untuk mengembangkan sebuah sistem yang dapat mengklasifikasikan jenis-jenis kain Ulos berdasarkan gambar, dengan memanfaatkan teknologi *machine learning* dan *image processing* menggunakan algoritma *Convolutional Neural Networks* (CNN). Algoritma CNN adalah salah satu algoritma yang mirip dengan arsitektur jaringan saraf tiruan yang telah terbukti sangat efektif dalam berbagai tugas pengolahan citra, termasuk pengenalan objek, pengenalan wajah, dan klasifikasi gambar. Algoritma ini bekerja dengan cara mengekstraksi fitur-fitur penting dari gambar melalui lapisan konvolusi, yang kemudian diproses untuk menghasilkan prediksi tentang isi gambar tersebut. Keunggulan algoritman CNN terletak pada kemampuannya untuk belajar secara otomatis mengenali pola-pola visual yang ada dalam gambar tanpa memerlukan intervensi manusia yang signifikan dalam hal pemrograman fitur. Oleh karena itu, algoritma CNN menjadi pilihan yang ideal untuk proyek DiTenun – *Ulos Image Classification* ini.

Melalui proyek ini, kami akan mengembangkan sebuah sistem yang dapat mengenali dan mengklasifikasikan berbagai jenis ulos Batak berdasarkan gambar. Sistem ini akan dilatih menggunakan data gambar ulos yang sudah diberi label sesuai dengan jenis dan motifnya. Dengan menggunakan algoritma CNN, sistem diharapkan dapat mengenali berbagai ciri khas pada kain ulos. Keberhasilan sistem ini tidak hanya akan mempermudah proses identifikasi ulos, tetapi juga akan berkontribusi pada pelestarian budaya, di mana masyarakat, peneliti, dan pelestari budaya dapat lebih mudah mengakses informasi mengenai ulos dan mengidentifikasi jenis-jenisnya.

1.2 Tujuan

1. Membuat Sistem Klasifikasi Gambar Ulos

Tujuan utama dari proyek ini adalah untuk membuat sebuah sistem yang dapat mengidentifikasi dan mengklasifikasikan berbagai jenis kain ulos Batak hanya dengan melihat gambar. Sistem ini menggunakan algoritma CNN yang memungkinkan komputer untuk "belajar" mengenali berbagai ciri khas dari setiap jenis ulos, seperti motif dan warna.

2. Meningkatkan Keakuratan dan Kecepatan Klasifikasi

Klasifikasi ulos secara manual membutuhkan banyak waktu dan pengetahuan mendalam. Dengan menggunakan CNN, proses ini menjadi jauh lebih cepat dan akurat. Sistem ini bisa mengenali berbagai motif ulos, bahkan yang cukup rumit, dengan tingkat ketepatan yang tinggi, sehingga membuat proses identifikasi lebih efisien dan dapat dilakukan dengan lebih mudah oleh siapa saja.

3. Mempermudah Akses dan Melestarikan Budaya Menggunakan Teknologi

Proyek ini juga bertujuan untuk menunjukkan bagaimana teknologi modern, seperti algoritma CNN, dapat digunakan untuk melestarikan dan mendokumentasikan warisan budaya. Kami berharap ini bisa menjadi inspirasi bagi penggunaan teknologi dalam melestarikan budaya-budaya lainnya di Indonesia, serta memotivasi riset-riset lebih lanjut di bidang ini.

1.3 Manfaat

1. Mempermudah Identifikasi Ulos

Salah satu manfaat terbesar dari proyek ini adalah mempermudah proses identifikasi berbagai jenis ulos. Dengan sistem yang dapat mengklasifikasikan ulos hanya dari gambar,

orang yang tidak memiliki pengetahuan khusus tentang ulos bisa dengan mudah mengetahui jenis dan makna kain tersebut. Ini akan sangat membantu dalam konteks edukasi, pameran budaya, atau bahkan bagi wisatawan yang tertarik mempelajari lebih lanjut tentang budaya Batak.

2. Meningkatkan Akses Pengetahuan tentang Ulos

Proyek ini memungkinkan siapa saja, di mana saja, untuk mengakses informasi tentang ulos dengan mudah. Misalnya, peneliti, pelestari budaya, atau bahkan masyarakat umum yang ingin mempelajari ulos dapat menggunakan sistem ini sebagai referensi cepat. Ini juga dapat digunakan di museum, galeri, atau tempat wisata sebagai alat bantu edukasi yang menarik.

3. Mengurangi Ketergantungan pada Ahli untuk Klasifikasi Manual

Sebelumnya, untuk mengenali jenis ulos, dibutuhkan keahlian khusus dari orang yang sudah berpengalaman dalam tenun ulos. Dengan adanya sistem berbasis CNN, kita tidak lagi perlu bergantung pada ahli untuk setiap klasifikasi. Sistem ini memungkinkan identifikasi yang lebih cepat, efisien, dan dapat diakses oleh lebih banyak orang tanpa batasan keahlian.

1.4 Ruang Lingkup

1. Koleksi dan Pengumpulan Data Gambar Ulos

Proyek ini akan mencakup pengumpulan gambar ulos dari berbagai jenis dan motif yang ada dalam budaya Batak. Gambar-gambar ini akan digunakan sebagai dataset untuk melatih algoritma CNN. Data yang dikumpulkan akan mencakup representasi dari berbagai jenis ulos yang sering digunakan.

2. Pengembangan Algoritma CNN untuk Klasifikasi

Fokus utama proyek ini adalah pengembangan dan pelatihan algoritma CNN untuk mengklasifikasikan jenis-jenis ulos berdasarkan gambar. Model akan dilatih dengan menggunakan dataset yang telah didapatkan untuk memastikan akurasi dalam mengenali berbagai jenis ulos berdasarkan fitur visualnya.

3. Evaluasi dan Pengujian Akurasi Sistem

Setelah model dikembangkan, tahap selanjutnya adalah evaluasi performa model melalui pengujian dengan data yang tidak digunakan dalam pelatihan. Pengujian ini bertujuan

untuk mengukur tingkat akurasi sistem dalam mengklasifikasikan ulos dengan benar. Hasil pengujian akan digunakan untuk memperbaiki dan menyempurnakan model agar lebih akurat.

4. Implementasi Sistem Klasifikasi

Sistem yang telah dikembangkan akan diimplementasikan dalam bentuk aplikasi atau platform berbasis web yang memungkinkan pengguna untuk mengunggah gambar ulos dan mendapatkan hasil klasifikasi secara otomatis. Sistem ini juga akan dilengkapi dengan penjelasan tentang jenis ulos yang terdeteksi beserta makna simbolisnya.

5. Penyusunan Dokumentasi

Sebagai bagian dari proyek, akan disusun dokumentasi yang menjelaskan mengenai proses pembuatan model pengklasifikasian ulos dengan algoritma CNN.

BAB 2

STUDI LITERATUR

2.1 Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNN) menjadi salah satu pendekatan yang paling menarik baru-baru ini dan telah menjadi faktor utama dalam berbagai keberhasilan baru-baru ini dan aplikasi yang menantang terkait dengan aplikasi pembelajaran mesin seperti deteksi objek ImageNet tantangan, klasifikasi gambar, dan pengenalan wajah. Algoritma CNN digunakan untuk segmentasi dan klasifikasi gambar dalam transaksi akademik. Algoritma ini dimanfaatkan untuk pengenalan gambar di berbagai area misalnya pengaturan gambar otomatis, fotografi stok, pengenalan wajah, dan banyak pekerjaan terkait lainnya [3].

Algoritma CNN dimanfaatkan untuk mengklasifikasikan gambar, mengelompokkannya berdasarkan kesamaan (pencarian foto), dan melakukan pengenalan objek dalam adegan. Pada algoritma ini, terdapat lapisan konvolusional sebagai blok bangunan inti dari CNN. Parameter lapisan terdiri dari satu set filter yang dapat dipelajari (atau kernel) yang memiliki reseptif kecil tetapi meluas melalui kedalaman penuh volume input. Selama *forward pass*, setiap filter dikonvolitasi melintasi lebar dan tinggi volume input, menghitung produk titik, dan menghasilkan peta aktivasi 2 dimensi dari filter tersebut. Akibatnya, jaringan akan dapat mempelajari tentang filter. Filter diaktifkan ketika akan dilakukan pencarian terhadap beberapa jenis fitur tertentu pada beberapa posisi spasial dalam *input*. Kemudian peta aktivasi dimasukkan ke dalam lapisan *downsampling*, dan seperti konvolusi, metode ini diterapkan satu tambalan pada satu waktu. Algoritma CNN juga memiliki lapisan yang terhubung sepenuhnya yang mengklasifikasikan output dengan satu label per node [3].

2.2 Penerapan CNN dalam Klasifikasi Gambar

Penerapan algoritma CNN dalam klasifikasi gambar melibatkan beberapa langkah utama. Pertama, dilakukan persiapan data, yaitu pengaturan data pelatihan dan pengujian, penggabungan data, penggabungan label, dan penyesuaian ukuran data yang sesuai. Algoritma CNN digunakan untuk mengekstraksi, menganalisis, dan memahami informasi dari gambar secara otomatis. Dalam sistem klasifikasi gambar otomatis, algoritma ini sering digunakan untuk memanfaatkan fitur dari lapisan atas untuk klasifikasi, meskipun dalam beberapa kasus, fitur dari lapisan bawah dapat

memiliki kekuatan diskriminatif yang lebih besar. CNN terdiri dari beberapa lapisan, termasuk lapisan konvolusi yang merupakan blok bangunan inti dari CNN [3].

Lapisan ini memiliki parameter berupa filter yang dapat dipelajari, yang memiliki bidang reseptif kecil tetapi meluas melalui kedalaman penuh dari volume input. Selama proses *forward pass*, setiap filter dikonvolusikan melintasi lebar dan tinggi volume input, menghasilkan peta aktivasi dua dimensi. Aktivasi ini kemudian digunakan untuk mengklasifikasikan gambar ke dalam kelas yang telah ditentukan, seperti pesawat, mobil, burung, kucing, dan lainnya [3].

BAB 3

METODE PENELITIAN

3.1 *Cross-Industry Standard Process for Data Mining (CRISP-DM)*

Berikut adalah struktur pemodelan untuk pemrosesan klasifikasi gambar motif ulos.

3.1.1 *Business Understanding*

Business understanding dilakukan untuk memastikan bahwa solusi yang dikembangkan benar-benar relevan dan bermanfaat bagi pihak-pihak yang terlibat. Dengan memahami kebutuhan dan tantangan yang ada misalnya, kesulitan dalam mengidentifikasi berbagai jenis ulos secara manual kita bisa mengarahkan pengembangan algoritma CNN untuk memenuhi tujuan praktis, seperti mempermudah proses identifikasi ulos. Dengan pendekatan ini, teknologi yang digunakan bisa lebih tepat sasaran dan memberikan manfaat yang maksimal.

Ulos adalah kain tradisional suku Batak yang memiliki berbagai motif sesuai dengan makna budayanya. Motif yang beragam dengan variasi dan pola yang rumit sering kali menyebabkan penggolongan ulos yang cukup rumit. Tugas analitik utama adalah klasifikasi motif ulos. Dengan menggunakan algoritma CNN, proyek ini bertujuan untuk memprediksi kelas atau kategori ulos berdasarkan gambar motifnya. Dataset gambar ulos diambil dari Kaggle, dan diharapkan dapat membantu proses identifikasi, pelabelan, dan dokumentasi motif ulos secara otomatis dan efisien. Tujuan dari proyek ini adalah mengembangkan algoritma CNN untuk melakukan klasifikasi terhadap motif ulos berdasarkan jenis motifnya. Target utamanya adalah mencapai akurasi sebesar $\geq 90\%$ pada data uji.

3.1.2 *Data Understanding*

Berikut adalah tujuan dari setiap langkah dalam data understanding untuk proyek klasifikasi ulos menggunakan model CNN:

a. Mengumpulkan Data

Pengumpulan data dilakukan untuk mendapatkan gambar-gambar ulos yang nantinya menjadi dasar pengklasifikasian ulos. Data yang lengkap dan bervariasi akan membantu algoritma CNN dalam belajar mengenali berbagai pola dan fitur visual pada ulos. Pada proyek ini, data

diambil dari [Kaggle](#), yang dimana data sudah dibagi menjadi *test* dan *train*. Dataset kemudian diunduh ke dalam Google Drive.

Berikut adalah struktur dataset, dapat dilihat pada Gambar 3-1.

Train	12/30/2024 10:56 PM	File folder
Test	12/30/2024 10:56 PM	File folder

Gambar 3-1 Struktur Dataset

Untuk setiap folder train dan test memiliki struktur sub-folder kategori ulos dapat dilihat pada Gambar 3-2.

Tumtuman	12/30/2024 10:56 PM	File folder
Sadum	12/30/2024 10:56 PM	File folder
Sibolang	12/30/2024 10:56 PM	File folder
Ragi Hotang	12/30/2024 10:56 PM	File folder
Ragi Hidup	12/30/2024 10:56 PM	File folder
Pinuncaan	12/30/2024 10:56 PM	File folder

Gambar 3-2 Sub-Folder Kategori Ulos

Data kemudian diakses dengan menggunakan kode program pada Tabel 3-1.

Tabel 3-1 Kode Program untuk Mengakses Data

```
#Mount Google Drive

from google.colab import drive
import os

# Mount Google Drive
drive.mount('/content/drive')

# Path ke folder train dan test
train_path = '/content/drive/MyDrive/Dataset Ulos/Train' # Ganti dengan path folder train
test_path = '/content/drive/MyDrive/Dataset Ulos/Test' # Ganti dengan path folder test
```

Kode diatas akan menampilkan hasil **‘Mounted at /content/drive’** yang berarti Google Colab telah berhasil terhubung ke Google Drive dan file sudah dapat diakses dari direktori.

Langkah selanjutnya adalah melihat jumlah kategori dan gambar yang terdapat pada folder *train* dan *test* dengan menggunakan kode program pada Tabel 3-2.

Tabel 3-2 Kode Program untuk Jumlah Kategori

```
import os
import pandas as pd
from tabulate import tabulate

def count_images(folder_path):
    # Inisialisasi dictionary untuk menyimpan jumlah gambar per kategori
    category_counts = {}

    # Loop melalui setiap folder di dalam folder_path
    for category_folder in os.listdir(folder_path):
        category_path = os.path.join(folder_path, category_folder)
        if os.path.isdir(category_path): # Pastikan hanya membaca folder
            num_images = len([
                f for f in os.listdir(category_path)
                if not f.startswith('.') and f.lower().endswith(('png', 'jpg', 'jpeg'))
            ])
            category_counts[category_folder] = num_images

    # Konversi hasil ke DataFrame
    category_df = pd.DataFrame(list(category_counts.items()), columns=['Category', 'Image Count'])
    total_images = category_df['Image Count'].sum()
    return category_df, total_images

# Hitung data train
train_category_df, total_train_images = count_images(train_path)
print("Jumlah kategori dan gambar pada folder train:")
print(tabulate(train_category_df, headers='keys', tablefmt='pretty', showindex=False))
print(f"\nJumlah total gambar pada folder train: {total_train_images}")

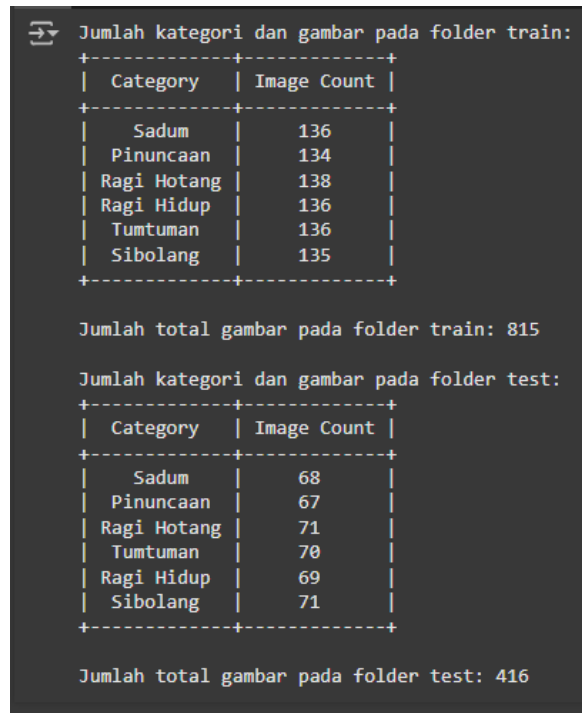
# Hitung data test
```

```

test_category_df, total_test_images = count_images(test_path)
print("\nJumlah kategori dan gambar pada folder test:")
print(tabulate(test_category_df, headers='keys', tablefmt='pretty', showindex=False))
print(f"\nJumlah total gambar pada folder test: {total_test_images}")

```

Dari kode program diatas, dihasilkan informasi jumlah kategori dan gambar pada folder *train* dan folder *test*, yang dapat dilihat pada Gambar 3-3.



```

Jumlah kategori dan gambar pada folder train:
+-----+-----+
| Category | Image Count |
+-----+-----+
| Sadum    | 136         |
| Pinunsaan | 134         |
| Ragi Hotang | 138         |
| Ragi Hidup | 136         |
| Tumtuman  | 136         |
| Sibolang  | 135         |
+-----+-----+

Jumlah total gambar pada folder train: 815

Jumlah kategori dan gambar pada folder test:
+-----+-----+
| Category | Image Count |
+-----+-----+
| Sadum    | 68          |
| Pinunsaan | 67          |
| Ragi Hotang | 71          |
| Tumtuman  | 70          |
| Ragi Hidup | 69          |
| Sibolang  | 71          |
+-----+-----+

Jumlah total gambar pada folder test: 416

```

Gambar 3-3 Informasi Jumlah Kategori dan Gambar

Setelah mengetahui jumlah data, dilakukan pendefinisian deskripsi data dengan menggunakan kode program yang dapat dilihat pada Tabel 3-3.

Tabel 3-3 Kode Program untuk Deskripsi Data

```

from google.colab import drive
import os
import pandas as pd
from tabulate import tabulate

# Path folder dataset
dataset_path = '/content/drive/MyDrive/Dataset Ulos'

```

```

# Fungsi untuk mendapatkan deskripsi data
def describe_dataset(dataset_path):
    file_details = []

    # Menelusuri seluruh file dan folder
    for root, dirs, files in os.walk(dataset_path):
        for file in files:
            if not file.startswith('.') and file.lower().endswith(('png', 'jpg', 'jpeg', 'tiff', 'bmp')):
                # Menentukan kategori berdasarkan nama folder
                category = os.path.basename(root)
                file_format = file.split('.')[-1] # Mendapatkan ekstensi file
                file_details.append({'Category': category, 'File Name': file, 'File Format': file_format})

    # Mengubah hasil ke dalam bentuk DataFrame
    file_df = pd.DataFrame(file_details)
    return file_df

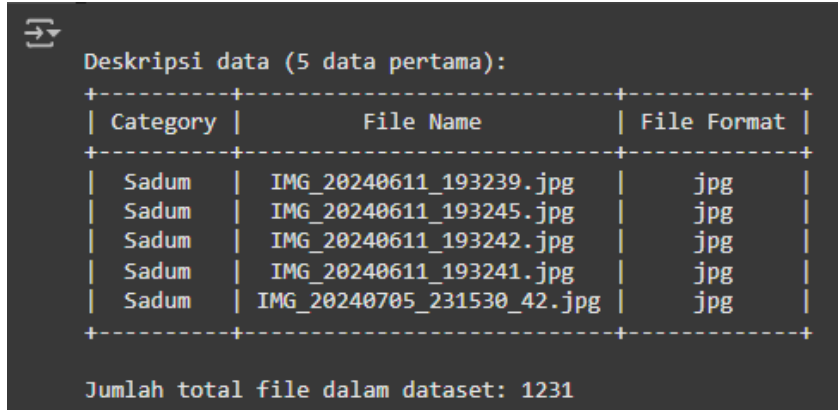
# Mendapatkan deskripsi dataset
file_df = describe_dataset(dataset_path)

# Menampilkan 5 data pertama
print("\nDeskripsi data (5 data pertama):")
print(tabulate(file_df.head(5), headers='keys', tablefmt='pretty', showindex=False))

# Menampilkan jumlah total file dalam dataset
print(f"\nJumlah total file dalam dataset: {len(file_df)}")

```

Dari kode program diatas, deskripsi diatas disajikan dengan menyajikan nama kategori (*category*), nama file (*file name*), dan format file (*format file*). Deskripsi data yang disajikan diwakili dari 5 data pertama. Diberikan juga informasi bahwa jumlah total file dalam dataset adalah 1231 data. Tampilan *output* yang dihasilkan dari kode program dapat dilihat pada Gambar 3-4.



Deskripsi data (5 data pertama):

Category	File Name	File Format
Sadum	IMG_20240611_193239.jpg	jpg
Sadum	IMG_20240611_193245.jpg	jpg
Sadum	IMG_20240611_193242.jpg	jpg
Sadum	IMG_20240611_193241.jpg	jpg
Sadum	IMG_20240705_231530_42.jpg	jpg

Jumlah total file dalam dataset: 1231

Gambar 3-4 Hasil Deskripsi Data

b. Menelaah Data

Proses menelaah data ini dilakukan untuk menganalisa data secara eksploratif (*Exploratory Data Analysis*). Dalam tahap ini, kita akan mempelajari karakteristik gambar ulos yang ada untuk membantu memahami deskripsi data yang digunakan. Kode program untuk menampilkan karakteristik data dapat dilihat pada Tabel 3-4.

Tabel 3-4 Kode Program untuk Karakteristik Data

```
import os
import pandas as pd
from tabulate import tabulate

# Path dataset
dataset_path = '/content/drive/MyDrive/Dataset Ulos'

# Fungsi untuk mendapatkan deskripsi data lengkap dengan karakteristik
def describe_dataset(dataset_path):
    file_details = []

    # Menelusuri seluruh file dan folder
    for root, dirs, files in os.walk(dataset_path):
        for file in files:
            if not file.startswith('.') and file.lower().endswith(('png', 'jpg', 'jpeg', 'tiff', 'bmp')):
                # Menentukan kategori berdasarkan nama folder
                category = os.path.basename(root)
                file_format = file.split('.')[-1] # Mendapatkan ekstensi file
                file_path = os.path.join(root, file) # Mendapatkan path file lengkap
```

```

# Mendapatkan karakteristik file: ukuran dan tanggal terakhir dimodifikasi
file_size = os.path.getsize(file_path) # Ukuran file dalam byte
last_modified = os.path.getmtime(file_path) # Timestamp terakhir dimodifikasi

# Menyimpan informasi dalam bentuk dictionary
file_details.append({
    'Category': category,
    'File Name': file,
    'File Format': file_format,
    'File Size (bytes)': file_size,
    'Last Modified': last_modified
})

# Mengubah hasil ke dalam bentuk DataFrame
file_df = pd.DataFrame(file_details)
return file_df

# Mendapatkan deskripsi dataset
file_df = describe_dataset(dataset_path)

# Menampilkan semua deskripsi dataset menggunakan tabulate
print(tabulate(file_df.head(5), headers='keys', tablefmt='pretty', showindex=False)) # Menampilkan semua
baris

# Menampilkan jumlah total file yang dianalisis
print(f"\nJumlah total file dalam dataset: {len(file_df)}")

```

Kode program diatas akan menampilkan karakteristik data yang berisi nama kategori ulos (*category*), nama file (*file name*), format file (*file format*), ukuran file (*file size*) dalam satuan bytes, dan waktu terakhir modifikasi (*last modified*). Hasil *output* yang ditampilkan hanya dari 5 data pertama, dan dapat dilihat pada Gambar 3-5.

Category	File Name	File Format	File Size (bytes)	Last Modified
Sadum	IMG_20240611_193239.jpg	jpg	6034986	1735573776.0
Sadum	IMG_20240611_193245.jpg	jpg	5405408	1735573776.0
Sadum	IMG_20240611_193242.jpg	jpg	5275014	1735573776.0
Sadum	IMG_20240611_193241.jpg	jpg	5275509	1735573776.0
Sadum	IMG_20240705_231530_42.jpg	jpg	1955647	1735573777.0

Gambar 3-5 Hasil Karakteristik Data

Untuk keterkaitan antar data pada proyek ini dianalisis dengan menggunakan Anova, Kendall's Tau, dan Chi-Squared. Kode untuk menampilkan keterkaitan data dapat dilihat pada Tabel 3-5.

Tabel 3-5 Kode Program untuk Keterkaitan Antar Data

```
import os
import pandas as pd
import numpy as np
from scipy.stats import chi2_contingency, kendalltau, f_oneway
from sklearn.feature_selection import mutual_info_classif
import seaborn as sns
import matplotlib.pyplot as plt
from tabulate import tabulate

# Path dataset
dataset_path = '/root/.cache/kagglehub/datasets/fthnaja/kain-ulos/versions/3'

# Fungsi untuk mendapatkan deskripsi data lengkap dengan karakteristik
def describe_dataset(dataset_path):
    file_details = []

    # Menelusuri seluruh file dan folder
    for root, dirs, files in os.walk(dataset_path):
        for file in files:
            if not file.startswith('.') and file.lower().endswith(('png', 'jpg', 'jpeg', 'tiff', 'bmp')):
                # Menentukan kategori berdasarkan nama folder
                category = os.path.basename(root)
                file_format = file.split('.')[-1] # Mendapatkan ekstensi file
                file_path = os.path.join(root, file) # Mendapatkan path file lengkap
```

```

# Mendapatkan karakteristik file: ukuran dan tanggal terakhir dimodifikasi
file_size = os.path.getsize(file_path) # Ukuran file dalam byte
last_modified = os.path.getmtime(file_path) # Timestamp terakhir dimodifikasi

# Menyimpan informasi dalam bentuk dictionary
file_details.append({
    'Category': category,
    'File Name': file,
    'File Format': file_format,
    'File Size (bytes)': file_size,
    'Last Modified': last_modified
})

# Mengubah hasil ke dalam bentuk DataFrame
file_df = pd.DataFrame(file_details)
return file_df

# Mendapatkan deskripsi dataset
file_df = describe_dataset(dataset_path)

# Uji ANOVA: Uji perbedaan ukuran file antar kategori
categories = file_df['Category'].unique()
file_sizes_by_category = [file_df[file_df['Category'] == category]['File Size (bytes)'] for category in
categories]

# Melakukan uji ANOVA
f_stat, p_value = f_oneway(*file_sizes_by_category)

# Menampilkan hasil uji ANOVA
print(f"\nANOVA Test Results:")
print(f"F-statistic: {f_stat}, p-value: {p_value}")

if p_value < 0.05:
    print("Ada perbedaan signifikan dalam ukuran file antar kategori.")
else:
    print("Tidak ada perbedaan signifikan dalam ukuran file antar kategori.")

```

```

# Kendall's Tau: Mengukur asosiasi antara dua variabel ordinal (misalnya ukuran file dan kategori numerik)
# Mengonversi kategori menjadi angka untuk analisis
category_mapping = {category: idx for idx, category in enumerate(file_df['Category'].unique())}
file_df['Category Numeric'] = file_df['Category'].map(category_mapping)

kendall_corr, kendall_p_value = kendalltau(file_df['File Size (bytes)'], file_df['Category Numeric'])

# Menampilkan hasil Kendall's Tau
print(f"\nKendall's Tau Test Results:")
print(f"Tau: {kendall_corr}, p-value: {kendall_p_value}")

if kendall_p_value < 0.05:
    print("Ada asosiasi signifikan antara ukuran file dan kategori.")
else:
    print("Tidak ada asosiasi signifikan antara ukuran file dan kategori.")

# Uji Chi-Squared: Uji independensi antara kategori dan format file
contingency_table = pd.crosstab(file_df['Category'], file_df['File Format'])
chi2, p, dof, expected = chi2_contingency(contingency_table)

print(f"\nChi-Squared Test Results:")
print(f"Chi-Squared: {chi2}, p-value: {p}")

if p < 0.05:
    print("Ada asosiasi signifikan antara kategori dan format file.")
else:
    print("Tidak ada asosiasi signifikan antara kategori dan format file.")

# Mutual Information: Mengukur seberapa banyak informasi yang dibagikan antara dua variabel
# Memerlukan konversi kategori menjadi numerik untuk ukuran file
from sklearn.preprocessing import LabelEncoder

# Encode kategori dan format file
label_encoder = LabelEncoder()
file_df['Category Encoded'] = label_encoder.fit_transform(file_df['Category'])
file_df['File Format Encoded'] = label_encoder.fit_transform(file_df['File Format'])

```

```

# Menghitung Mutual Information antara kategori dan ukuran file
X = file_df[['Category Encoded', 'File Format Encoded']]
y = file_df['File Size (bytes)']

mutual_info = mutual_info_classif(X, y)
print(f"\nMutual Information between 'Category' and 'File Size (bytes)': {mutual_info[0]}")

# Visualisasi untuk ANOVA dan Chi-Squared

# ANOVA Visualisasi: Distribusi ukuran file antar kategori
plt.figure(figsize=(10, 6)) # Membuat figure baru dan mengatur ukuran
sns.boxplot(x='Category', y='File Size (bytes)', data=file_df)
plt.title('ANOVA: Ukuran File Antar Kategori')
plt.xticks(rotation=45)
plt.tight_layout() # Menyesuaikan tata letak
plt.subplots_adjust(top=0.9, bottom=0.2) # Memberikan jarak ekstra
plt.show()
plt.clf() # Membersihkan canvas untuk visualisasi berikutnya

# Chi-Squared Visualisasi: Tabel kontingensi antara kategori dan format file
plt.figure(figsize=(10, 6)) # Membuat figure baru dan mengatur ukuran
sns.heatmap(contingency_table, annot=True, cmap="Blues", fmt="d")
plt.title('Chi-Squared: Kategori vs Format File')
plt.tight_layout() # Menyesuaikan tata letak
plt.subplots_adjust(top=0.9, bottom=0.2) # Memberikan jarak ekstra
plt.show()
plt.clf() # Membersihkan canvas untuk visualisasi berikutnya

```

Kode program diatas menghasilkan ANOVA *Test Results*, Kendall's Tau *Test Results*, dan Chi-Squared *Test Results*. Hasilnya dapat dilihat pada Gambar 3-6

```

ANOVA Test Results:
F-statistic: 173.15445738956268, p-value: 1.9097527024194823e-139
Ada perbedaan signifikan dalam ukuran file antar kategori.

Kendall's Tau Test Results:
Tau: 0.19850872931039945, p-value: 4.47477465758758e-22
Ada asosiasi signifikan antara ukuran file dan kategori.

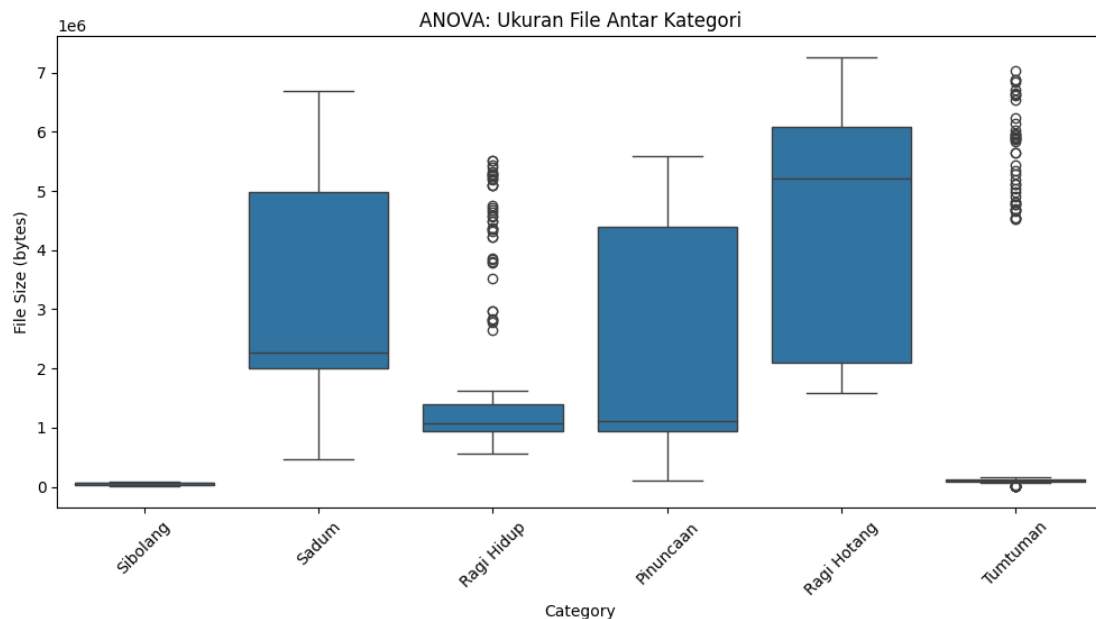
Chi-Squared Test Results:
Chi-Squared: 1575.0099207161215, p-value: 0.0
Ada asosiasi signifikan antara kategori dan format file.

```

Gambar 3-6 Hasil Keterkaitan Antar Data

Dari gambar diatas, dapat disimpulkan bahwa bahwa ukuran file berbeda secara signifikan antar kategori (ANOVA, p-value $1,91e-139$). Ada juga hubungan signifikan antara ukuran file dan kategori (Kendall's Tau, p-value $4,47e-22$), meskipun hubungan ini cukup lemah. Selain itu, kategori dan format file memiliki hubungan signifikan (Chi-Squared, p-value 0,0), yang berarti kategori tertentu cenderung menggunakan format file tertentu.

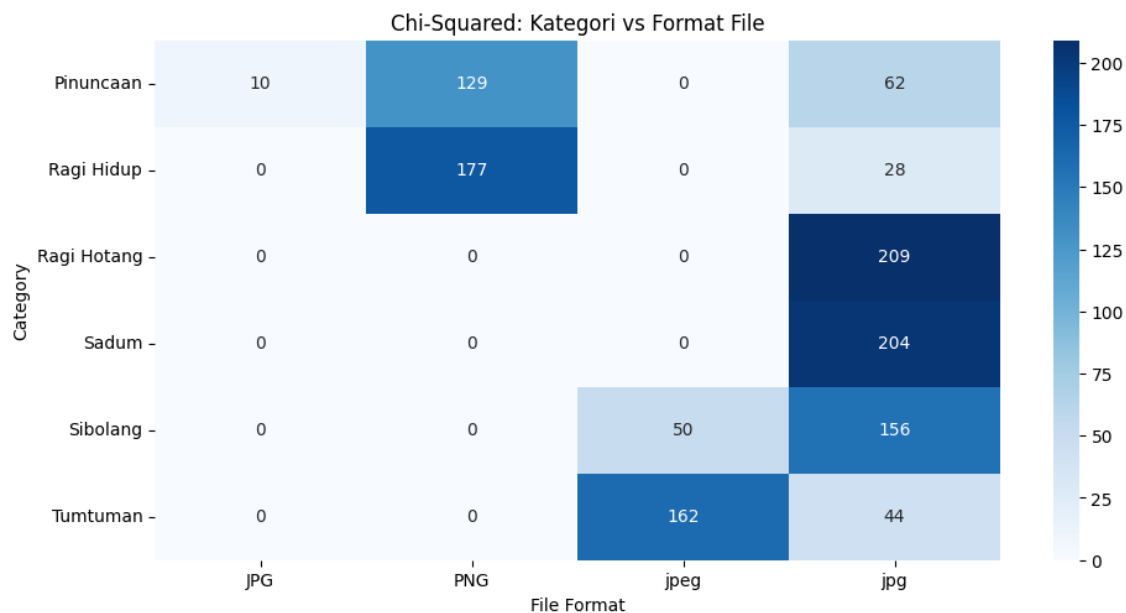
Gambar visualisasi untuk ANOVA Test Result dan Chi-Square Test Result dapat dilihat pada Gambar 3-7 dan Gambar 3-8.



Gambar 3-7 Visualisasi Keterkaitan Antar Data (ANOVA)

Pada Gambar 3-7 menunjukkan perbedaan ukuran file antar kategori berdasarkan hasil uji ANOVA. Dari grafik boxplot, terlihat bahwa kategori seperti Sadum dan Ragi Hotang

memiliki ukuran file yang lebih besar dibandingkan kategori lain, sementara Sibolang dan Tumtuman memiliki ukuran file yang sangat kecil. Hal ini mendukung kesimpulan bahwa ukuran file berbeda secara signifikan antar kategori.



Gambar 3-8 Visualisasi Keterkaitan Antar Data (Chi-Square)

Pada Gambar 3-8 menunjukkan hubungan antara kategori dan format file berdasarkan hasil uji Chi-Squared. Heatmap memperlihatkan pola distribusi format file pada setiap kategori. Misalnya, kategori Ragi Hotang lebih dominan menggunakan format file jpeg, sementara Tumtuman lebih sering menggunakan PNG. Pola ini menunjukkan bahwa kategori tertentu cenderung menggunakan format file yang spesifik.

c. Memvalidasi Data

Validasi data dilakukan untuk menilai kesesuaian kualitas data yang digunakan untuk klasifikasi motif ulos. Validasi pertama yang dilakukan adalah terhadap ukuran data. Kode program yang dibangun untuk melakukan validasi terhadap ukuran data dapat dilihat pada Tabel 3-6.

Tabel 3-6 Kode Program untuk Validasi Data (Ukuran Data)

```
import os
```



```

import pandas as pd

# Path dataset
dataset_path = '/content/drive/MyDrive/Dataset Ulos'

# Fungsi untuk mendapatkan deskripsi dataset dan validasi ukuran file
def validate_file_sizes(dataset_path, min_size=0, max_size=float('inf')):
    file_details = []

    # Menelusuri seluruh file dan folder
    for root, _, files in os.walk(dataset_path):
        for file in files:
            if not file.startswith('.') and file.lower().endswith(('png', 'jpg', 'jpeg', 'tiff', 'bmp')):
                # Menentukan kategori berdasarkan nama folder
                category = os.path.basename(root)
                file_format = file.split('.')[-1] # Mendapatkan ekstensi file
                file_path = os.path.join(root, file) # Mendapatkan path file lengkap

                # Mendapatkan karakteristik file: ukuran dan tanggal terakhir dimodifikasi
                file_size = os.path.getsize(file_path) # Ukuran file dalam byte
                last_modified = os.path.getmtime(file_path) # Timestamp terakhir dimodifikasi

                # Memvalidasi ukuran file apakah sesuai dengan batasan yang diberikan
                size_status = "Valid"
                if file_size < min_size:
                    size_status = f"Too small ({file_size} bytes)"
                elif file_size > max_size:
                    size_status = f"Too large ({file_size} bytes)"

                # Menyimpan informasi dalam bentuk dictionary
                file_details.append({
                    'Category': category,
                    'File Name': file,
                    'File Format': file_format,
                    'File Size (bytes)': file_size,
                    'Size Status': size_status,
                    'Last Modified': last_modified
                })

```

```

    })

    # Mengubah hasil ke dalam bentuk DataFrame
    file_df = pd.DataFrame(file_details)
    return file_df

# Menentukan batasan ukuran file (misalnya: min_size = 1000 bytes, max_size = 5 MB)
min_size = 1000 # minimal 1 KB
max_size = 5 * 1024 * 1024 # maksimal 5 MB

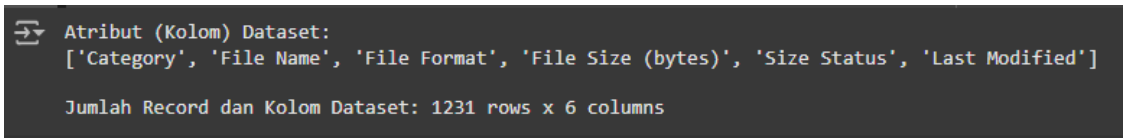
# Mendapatkan deskripsi dataset dan validasi ukuran file
file_df = validate_file_sizes(dataset_path, min_size, max_size)

# Menampilkan atribut (kolom) dan jumlah record
print("Atribut (Kolom) Dataset:")
print(file_df.columns.tolist())

# Menampilkan jumlah record dan kolom dalam format yang diinginkan
print(f"\nJumlah Record dan Kolom Dataset: {file_df.shape[0]} rows x {file_df.shape[1]} columns")

```

Kode program diatas menghasilkan informasi tentang atribut (kolom) dataset dan jumlah *record* dan kolom dataset. Hasil *outputnya* dapat dilihat pada Gambar 3-9.



```

Atribut (Kolom) Dataset:
['Category', 'File Name', 'File Format', 'File Size (bytes)', 'Size Status', 'Last Modified']

Jumlah Record dan Kolom Dataset: 1231 rows x 6 columns

```

Gambar 3-9 Hasil Validasi Data untuk Ukuran Data

Selanjutnya adalah menyajikan deskripsi statistik atribut dengan menggunakan kode program pada Tabel 3-7.

Tabel 3-7 Kode Program Untuk Validasi (Deskripsi Statistik Atribut)

```

import os
import pandas as pd

# Path dataset
dataset_path = '/content/drive/MyDrive/Dataset Ulos'

```

```

# Fungsi untuk mendapatkan deskripsi dataset dan validasi ukuran file
def get_file_statistics(dataset_path):
    file_details = []

    # Menelusuri seluruh file dan folder
    for root, dirs, files in os.walk(dataset_path):
        for file in files:
            if not file.startswith('.') and file.lower().endswith(('png', 'jpg', 'jpeg', 'tiff', 'bmp')):
                # Menentukan kategori berdasarkan nama folder
                category = os.path.basename(root)
                file_format = file.split('.')[-1] # Mendapatkan ekstensi file
                file_path = os.path.join(root, file) # Mendapatkan path file lengkap

                # Mendapatkan karakteristik file: ukuran dan tanggal terakhir dimodifikasi
                file_size = os.path.getsize(file_path) # Ukuran file dalam byte
                last_modified = os.path.getmtime(file_path) # Timestamp terakhir dimodifikasi

                # Menyimpan informasi dalam bentuk dictionary
                file_details.append({
                    'Category': category,
                    'File Name': file,
                    'File Format': file_format,
                    'File Size (bytes)': file_size,
                    'Last Modified': last_modified
                })

    # Mengubah hasil ke dalam bentuk DataFrame
    file_df = pd.DataFrame(file_details)
    return file_df

# Mendapatkan deskripsi dataset dan validasi ukuran file
file_df = get_file_statistics(dataset_path)

# Menampilkan deskripsi statistik dari atribut numerik
print("Deskripsi Statistik Atribut (Ukuran File dalam Byte):")
print(file_df['File Size (bytes)'].describe())

```

```
# Menampilkan statistik deskriptif kategori (kategori gambar)
print("\nFrekuensi Kategori:")
print(file_df['Category'].value_counts())

# Menampilkan statistik deskriptif terkait format file (ekstensi file)
print("\nFrekuensi Format File:")
print(file_df['File Format'].value_counts())
```

Kode diatas akan menampilkan hasil deskripsi statistik tentang ukuran file dalam byte, frekuensi kategori, dan frekuensi format file. Hasil *output*nya dapat dilihat pada Gambar 3-10.

```
➞ Deskripsi Statistik Atribut (Ukuran File dalam Byte):
count      1.231000e+03
mean       2.152272e+06
std        2.175859e+06
min        8.131000e+03
25%        1.017080e+05
50%        1.195456e+06
75%        4.300208e+06
max        7.253876e+06
Name: File Size (bytes), dtype: float64

Frekuensi Kategori:
Category
Ragi Hotang      209
Tumtuman        206
Sibolang         206
Ragi Hidup       205
Sadum            204
Pinuncaan        201
Name: count, dtype: int64

Frekuensi Format File:
File Format
jpg      703
PNG      306
jpeg     212
JPG       10
Name: count, dtype: int64
```

Gambar 3-10 Hasil Validasi Data untuk Deskripsi Statistik Atribut

Relasi antar atribut disajikan dengan menggunakan kode program pada Tabel 3-8.

Tabel 3-8 Kode Program untuk Validasi Data (Relasi Antar Atribut)

```
import os
```

```

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Path dataset
dataset_path = '/root/.cache/kagglehub/datasets/fthnaja/kain-ulos/versions/3'

# Fungsi untuk mendapatkan statistik file
def get_file_statistics(dataset_path):
    file_details = []

    # Menelusuri seluruh file dan folder
    for root, dirs, files in os.walk(dataset_path):
        for file in files:
            if not file.startswith('.') and file.lower().endswith(('png', 'jpg', 'jpeg', 'tiff', 'bmp')):
                # Menentukan kategori berdasarkan nama folder
                category = os.path.basename(root)
                file_format = file.split('.')[-1] # Mendapatkan ekstensi file
                file_path = os.path.join(root, file) # Mendapatkan path file lengkap

                # Mendapatkan karakteristik file: ukuran dan tanggal terakhir dimodifikasi
                file_size = os.path.getsize(file_path) # Ukuran file dalam byte
                last_modified = os.path.getmtime(file_path) # Timestamp terakhir dimodifikasi

                # Menyimpan informasi dalam bentuk dictionary
                file_details.append({
                    'Category': category,
                    'File Name': file,
                    'File Format': file_format,
                    'File Size (bytes)': file_size,
                    'Last Modified': last_modified
                })

    # Mengubah hasil ke dalam bentuk DataFrame
    file_df = pd.DataFrame(file_details)
    return file_df

```

```

# Mendapatkan deskripsi dataset dan validasi ukuran file
file_df = get_file_statistics(dataset_path)

# Menampilkan korelasi antar atribut numerik
print("\nKorelasi antar atribut numerik (hanya ukuran file dan waktu modifikasi):")
numeric_columns = ['File Size (bytes)', 'Last Modified']
correlation_matrix = file_df[numeric_columns].corr()

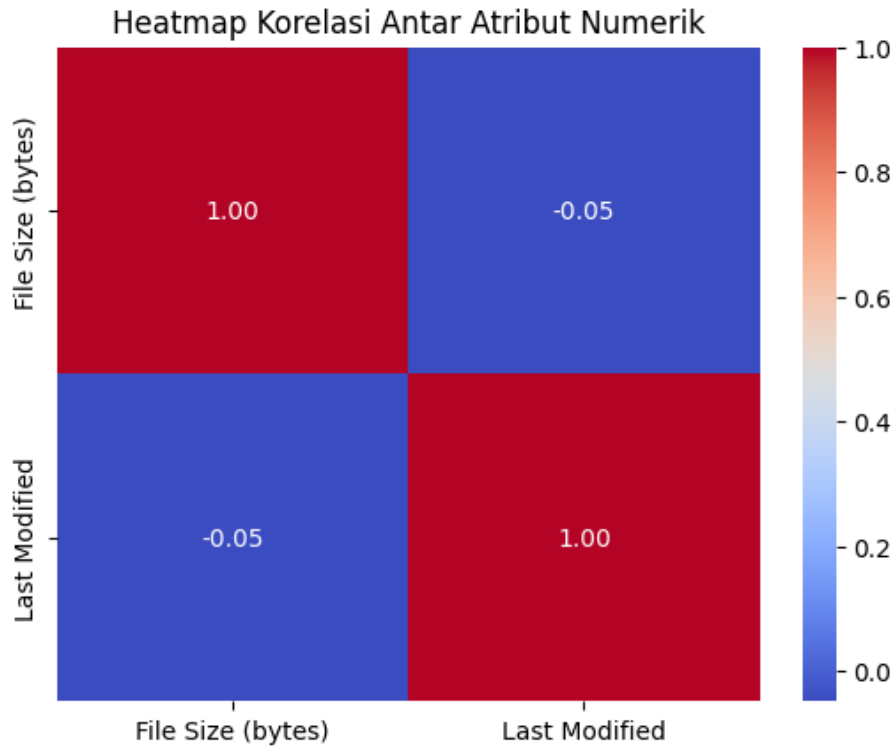
# Menampilkan heatmap korelasi
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f")
plt.title('Heatmap Korelasi Antar Atribut Numerik')
plt.show()

# Menampilkan relasi antara kategori dan format file menggunakan Cross Tabulation
print("\nCross-Tabulation antara Kategori dan Format File:")
category_format_ct = pd.crosstab(file_df['Category'], file_df['File Format'])
print(category_format_ct)

# Visualisasi Cross-Tabulation menggunakan Heatmap
sns.heatmap(category_format_ct, annot=True, cmap="Blues", fmt="d")
plt.title('Cross-Tabulation antara Kategori dan Format File')
plt.ylabel('Category')
plt.xlabel('File Format')
plt.show()

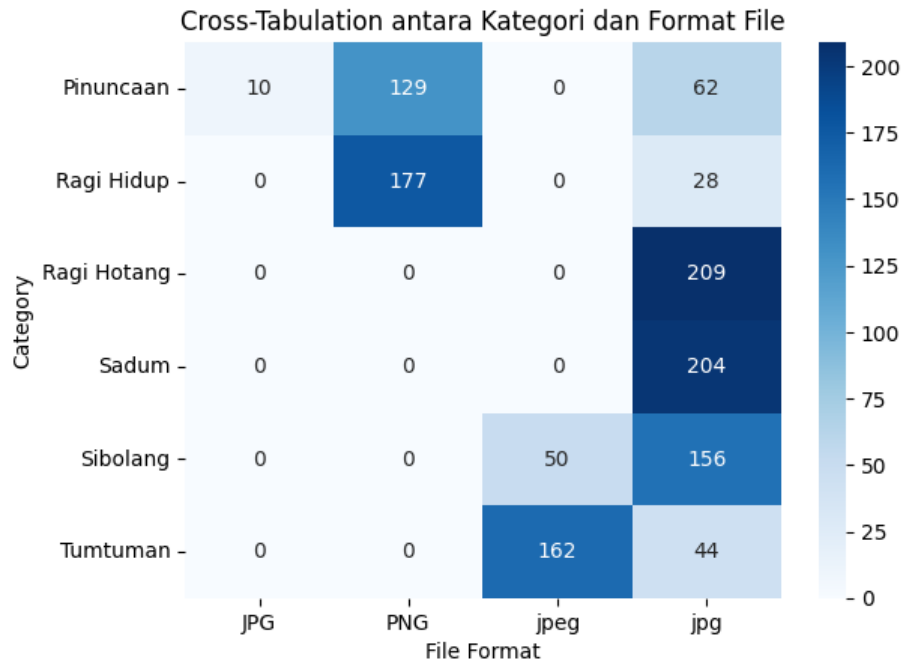
```

Kode program diatas akan menyajikan korelasi antar atribut numerik (ukuran file dan waktu modifikasi) serta *Cross-Tabulation* antara kategori dan format file. Visualisasi dapat dilihat pada Gambar 3-11 dan Gambar 3-12.



Gambar 3-11 Hasil Relasi Antar Atribut (Heatmap Korelasi Antar Atribut Numerik)

Pada Gambar 3-11 menunjukkan korelasi antar atribut numerik, yaitu *file size* (bytes) dan *last modified*. Nilai korelasi antara File Size dengan dirinya sendiri adalah 1.00, menunjukkan hubungan sempurna. Sementara itu, korelasi antara *file size* dan *last modified* bernilai -0.05, menunjukkan bahwa hampir tidak ada hubungan linear yang signifikan antara ukuran file dan waktu modifikasinya. Hasil ini mengindikasikan bahwa perubahan waktu modifikasi tidak berpengaruh pada ukuran file.



Gambar 3-12 Hasil Relasi Antar Atribut (*Cross-Tabulation* Antara Kategori dan Format File)

Pada Gambar 3-12 menunjukkan korelasi antar atribut numerik, yaitu *file size* (bytes) dan *last modified*. Nilai korelasi antara File Size dengan dirinya sendiri adalah 1.00, menunjukkan hubungan sempurna. Sementara itu, korelasi antara *file size* dan *last modified* bernilai -0.05, menunjukkan bahwa hampir tidak ada hubungan linear yang signifikan antara ukuran file dan waktu modifikasinya. Hasil ini mengindikasikan bahwa perubahan waktu modifikasi tidak berpengaruh pada ukuran file.

Untuk visualisasi datanya disajikan dengan menggunakan kode program pada Tabel 3-9.

Tabel 3-9 Kode Program untuk Visualisasi Data

```
import os
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Path dataset
dataset_path = '/root/.cache/kagglehub/datasets/fthnaja/kain-ulos/versions/3'

# Fungsi untuk mendapatkan statistik file
def get_file_statistics(dataset_path):
```



```

file_details = []

# Menelusuri seluruh file dan folder
for root, dirs, files in os.walk(dataset_path):
    for file in files:
        if not file.startswith('.') and file.lower().endswith(('png', 'jpg', 'jpeg', 'tiff', 'bmp')):
            # Menentukan kategori berdasarkan nama folder
            category = os.path.basename(root)
            file_format = file.split('.')[-1] # Mendapatkan ekstensi file
            file_path = os.path.join(root, file) # Mendapatkan path file lengkap

            # Mendapatkan karakteristik file: ukuran dan tanggal terakhir dimodifikasi
            file_size = os.path.getsize(file_path) # Ukuran file dalam byte
            last_modified = os.path.getmtime(file_path) # Timestamp terakhir dimodifikasi

            # Menyimpan informasi dalam bentuk dictionary
            file_details.append({
                'Category': category,
                'File Name': file,
                'File Format': file_format,
                'File Size (bytes)': file_size,
                'Last Modified': last_modified
            })

# Mengubah hasil ke dalam bentuk DataFrame
file_df = pd.DataFrame(file_details)
return file_df

# Mendapatkan deskripsi dataset dan validasi ukuran file
file_df = get_file_statistics(dataset_path)

# 1. Visualisasi distribusi ukuran file menggunakan Histogram
plt.figure(figsize=(10, 6))
sns.histplot(file_df['File Size (bytes)'], kde=True, color='skyblue')
plt.title('Distribusi Ukuran File')
plt.xlabel('Ukuran File (Bytes)')
plt.ylabel('Frekuensi')

```

```

plt.grid(True)
plt.show()

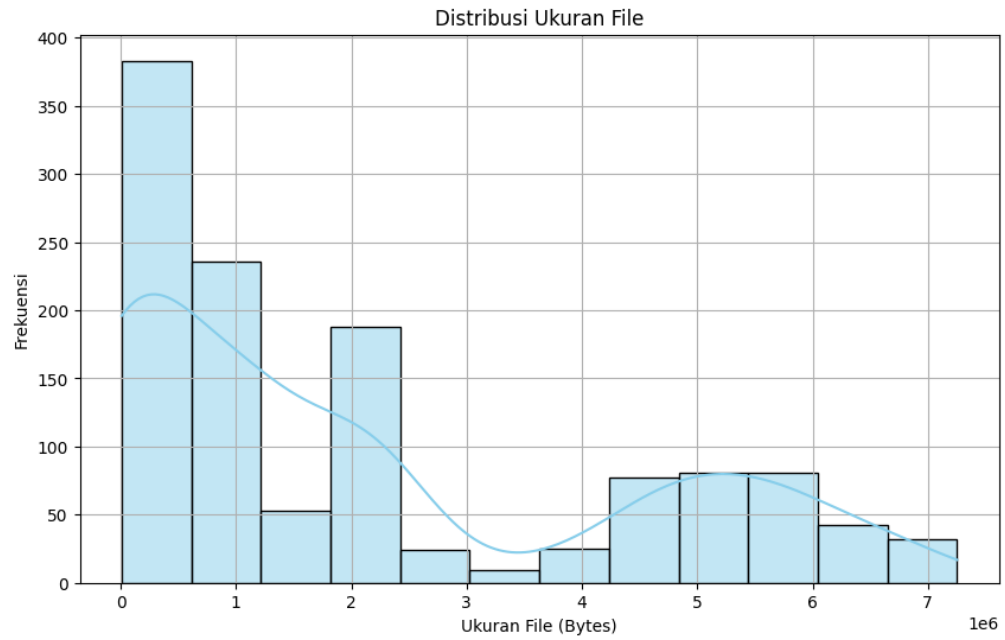
# 2. Boxplot untuk distribusi ukuran file
plt.figure(figsize=(8, 6))
sns.boxplot(x=file_df['File Size (bytes)'], color='lightgreen')
plt.title('Boxplot Ukuran File')
plt.xlabel('Ukuran File (Bytes)')
plt.show()

# 3. Visualisasi kategori file berdasarkan kategori menggunakan Barplot
plt.figure(figsize=(12, 6))
category_counts = file_df['Category'].value_counts()
sns.barplot(x=category_counts.index, y=category_counts.values, palette='muted')
plt.title('Jumlah File per Kategori')
plt.xlabel('Kategori')
plt.ylabel('Jumlah File')
plt.xticks(rotation=45)
plt.show()

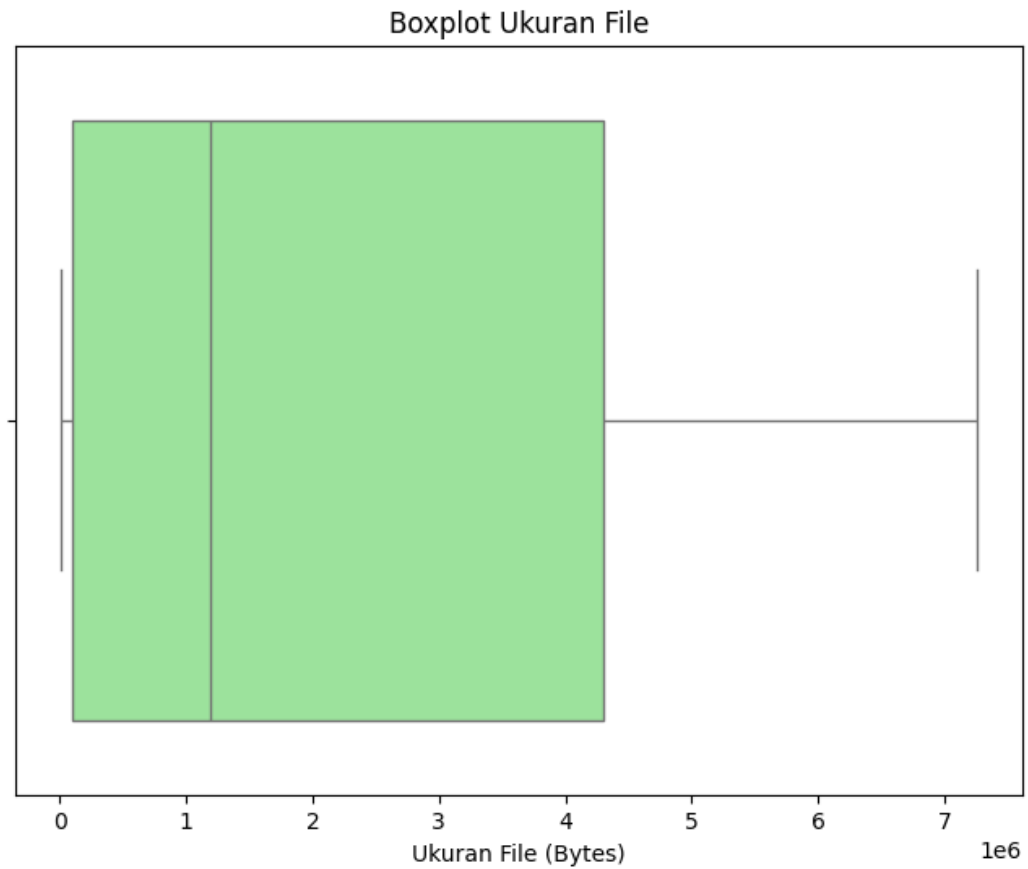
# 4. Pie chart untuk distribusi format file
plt.figure(figsize=(8, 8))
file_format_counts = file_df['File Format'].value_counts()
plt.pie(file_format_counts, labels=file_format_counts.index, autopct='% 1.1f%%',
        colors=sns.color_palette("Set3", len(file_format_counts)))
plt.title('Distribusi Format File')
plt.show()

```

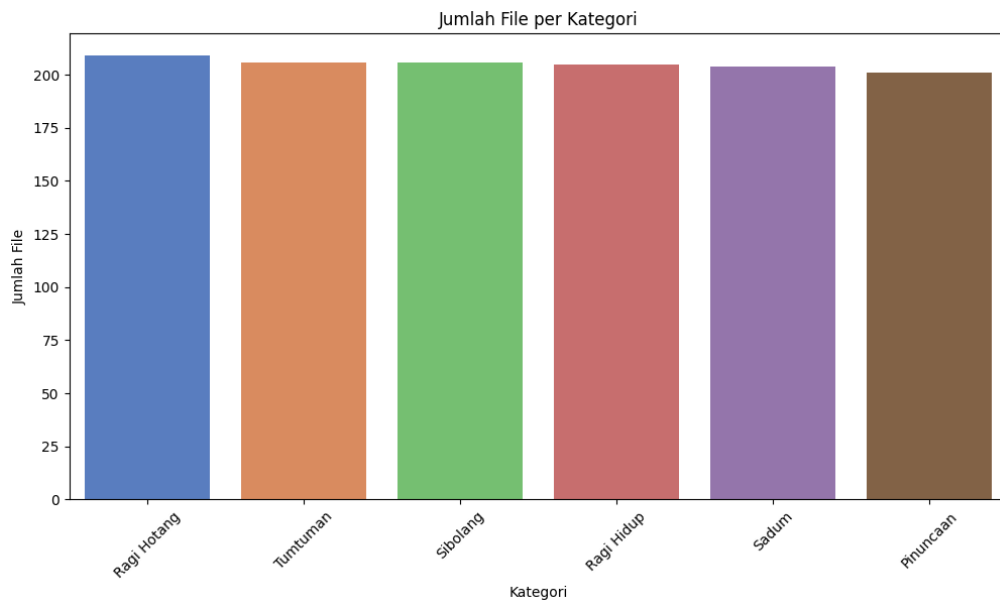
Kode diatas akan menghasilkan visualisasi data untuk menunjukkan analisis terhadap distribusi ukuran file, bloxpot ukuran file, jumlah file per kategori, dan distribusi format file. Visualisasi dapat dilihat pada Gambar 3-13, Gambar 3-14, Gambar 3-15, dan Gambar 3-16.



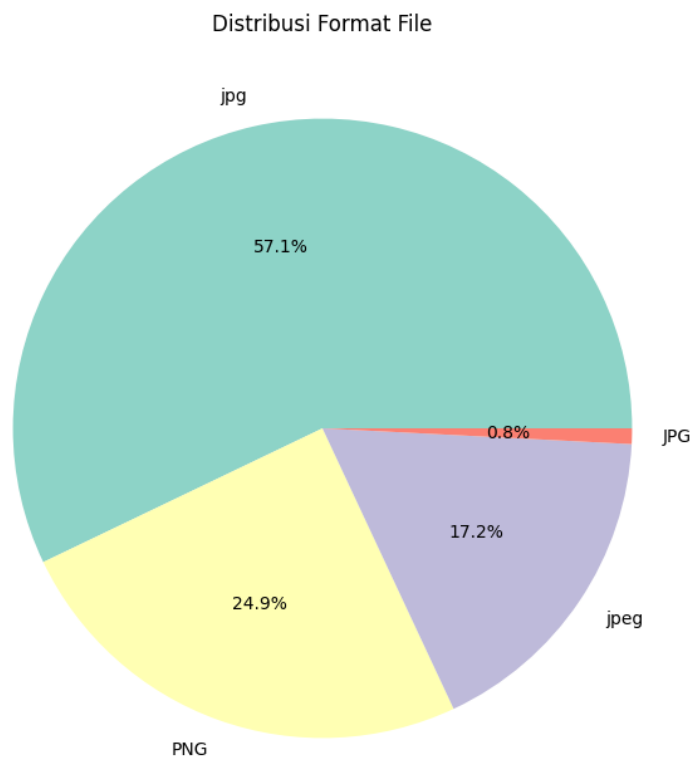
Gambar 3-13 Visualisasi Distribusi Ukuran File



Gambar 3-14 Visualisasi Boxplot Ukuran File



Gambar 3-15 Visualisasi Jumlah File per Kategori



Gambar 3-16 Visualisasi Distrbusi Format File

3.1.3 Data Preparation

Pada tahap *data preparation* (persiapan data), data akan disiapkan untuk pemodelan dengan beberapa langkah yang mencakup pembersihan, transformasi, dan seleksi fitur yang relevan. Persiapan data dilakukan untuk memperbaiki kualitas data untuk pemodelan. Beberapa langkah utama dalam tahap ini adalah:

a. Memilih dan Memilah Data

Pada langkah ini, data yang tersedia dipilih dan dipilah berdasarkan kategori atau label yang akan digunakan. Proses ini akan memproses record data dan atribut data yang terpakai. Dataset yang digunakan pada proses ini sudah terdiri dari folder "*Train*" dan "*Test*" yang berisi gambar sudah dikelompokkan kedalam kategori-kategori ulos tertentu. Kode program yang digunakan untuk pemilihan data dapat dilihat pada Tabel 3-10.

Tabel 3-10 Kode Program untuk Memilih dan Memilah Data

```
import os
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

# Path dataset
dataset_path = '/root/.cache/kagglehub/datasets/fthnaja/kain-ulos/versions/3'

# Direktori untuk kategori gambar
categories = ['Tumtuman', 'Pinunsaan', 'Ragi Hotang', 'Sibolang', 'Sadum', 'Ragi Hidup']

def prepare_data(dataset_path, categories):
    data = []
    labels = []

    # Iterasi untuk folder 'Train' dan 'Test'
    for split in ['Train', 'Test']:
        split_path = os.path.join(dataset_path, split)

        for category in categories:
            category_path = os.path.join(split_path, category)
```

```

# Pastikan folder kategori ada
if os.path.exists(category_path):
    # Memilih gambar dari folder kategori
    for filename in os.listdir(category_path):
        if filename.lower().endswith(('.png', '.jpg', '.jpeg')):
            img_path = os.path.join(category_path, filename)
            try:
                img = tf.keras.preprocessing.image.load_img(img_path, target_size=(128, 128)) # Resize
gambar
                img_array = tf.keras.preprocessing.image.img_to_array(img)
                data.append(img_array)
                labels.append(category)
            except Exception as e:
                print(f"Error loading image {img_path}: {e}")
        else:
            print(f"Folder not found: {category_path}")

# Konversi data dan labels ke array numpy
data = np.array(data)
labels = np.array(labels)

return data, labels

# Menyiapkan data
data, labels = prepare_data(dataset_path, categories)

# Normalisasi pixel gambar ke rentang [0, 1]
data = data / 255.0

# Membagi data menjadi training dan testing set
X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.2, random_state=42)

# Menggunakan OneHotEncoder untuk mengubah label menjadi format numerik
encoder = LabelEncoder()
y_train = encoder.fit_transform(y_train)
y_test = encoder.transform(y_test)

```

```

# Menyusun data augmentation untuk meningkatkan variasi data pelatihan
train_datagen = ImageDataGenerator(
    rescale=1./255, # Normalisasi
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

validation_datagen = ImageDataGenerator(rescale=1./255)

# Menggunakan ImageDataGenerator untuk memproses data pelatihan
train_generator = train_datagen.flow(X_train, y_train, batch_size=32)
validation_generator = validation_datagen.flow(X_test, y_test, batch_size=32)

print("Data Preparation Completed")

```

Pada kode diatas, gambar sudah diubah ukurannya ke (128, 128) piksel, yang merupakan bentuk rekonstruksi atribut data. Selain itu, gambar juga dikonversi ke dalam bentuk array menggunakan `img_to_array`. Setiap label dari gambar motif ulos kemudian dikonversi menggunakan `LabelEncoder`, dan digunakan *augmentation data* dengan menggunakan `ImageDataGenerator` untuk memingkatkan variasi dan kualitas model. Hasil akhir dari proses ini adalah data siap digunakan untuk pelatihan dan validasi model.

b. Membersihkan Data

Pada tahap ini, dilakukan pembersihan terhadap data gambar. Proses ini dilakukan untuk meminimalkan *noise* (tidak lengkap atau salah) pada data. Kode program untuk proses ini dapat dilihat pada Tabel 3-11.

Tabel 3-11 Kode Program untuk Membersihkan Data

```

import os
import numpy as np

```

```

import tensorflow as tf

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

# Path dataset
dataset_path = '/root/.cache/kagglehub/datasets/fthnaja/kain-ulos/versions/3'

# Direktori untuk kategori gambar
categories = ['Tumtuman', 'Pinuncaan', 'Ragi Hotang', 'Sibolang', 'Sadum', 'Ragi Hidup']

def clean_data(dataset_path, categories, target_size=(128, 128)):
    data = []
    labels = []

    # Iterasi untuk folder 'Train' dan 'Test'
    for split in ['Train', 'Test']:
        split_path = os.path.join(dataset_path, split)

        for category in categories:
            category_path = os.path.join(split_path, category)

            # Pastikan folder kategori ada
            if os.path.exists(category_path):
                # Memilih gambar dari folder kategori
                for filename in os.listdir(category_path):
                    if filename.lower().endswith(('.png', '.jpg', '.jpeg')):
                        img_path = os.path.join(category_path, filename)
                        try:
                            # Memuat dan meresize gambar
                            img = tf.keras.preprocessing.image.load_img(img_path, target_size=target_size)
                            img_array = tf.keras.preprocessing.image.img_to_array(img)
                            data.append(img_array)
                            labels.append(category)
                        except Exception as e:
                            print(f"Error loading image {img_path}: {e}")
                    else:
                        print(f"Folder not found: {category_path}")

```



```

# Konversi data dan labels ke array numpy
data = np.array(data)
labels = np.array(labels)

return data, labels

# Menyiapkan data
data, labels = clean_data(dataset_path, categories)

# Normalisasi pixel gambar ke rentang [0, 1]
data = data / 255.0

# Membagi data menjadi training dan testing set
X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.2, random_state=42)

# Menggunakan OneHotEncoder untuk mengubah label menjadi format numerik
encoder = LabelEncoder()
y_train = encoder.fit_transform(y_train)
y_test = encoder.transform(y_test)

print(f"Data Cleaning Completed. Total images: {len(data)}")

```

Kode program diatas akan membaca folder "*train*" dan "*test*" dari dataset yang ditentukan. Gambar di setiap kategori di-load, *resize* ke ukuran 128x128 piksel, dan dikonversi menjadi array numerik. Data yang berhasil di-load dinormalisasi ke rentang [0, 1], dan label kategori diubah menjadi format numerik. Hasil akhirnya adalah data gambar yang telah terorganisasi, ternormalisasi, dan siap digunakan untuk melatih model.

c. Mengkonstruksi Data

Setelah data dibersihkan, dilakukan konstruksi dataset yaitu menambahkan *feature engineering* dan melakukan transformasi terhadap data (standarisasi dan normalisasi). Dataset yang sudah dikonstruksi kemudian akan digunakan untuk dibagi menjadi menjadi set

pelatihan dan pengujian. Berikut kode program pada Tabel 3-12 yang digunakan untuk mengkonstruksi data.

Tabel 3-12 Kode Program untuk Mengkonstruksi Data

```
import os
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

# Path dataset
dataset_path = '/root/.cache/kagglehub/datasets/fthnaja/kain-ulos/versions/3'

# Direktori untuk kategori gambar
categories = ['Tumtuman', 'Pinuncaan', 'Ragi Hotang', 'Sibolang', 'Sadum', 'Ragi Hidup']

def construct_data(dataset_path, categories, target_size=(128, 128)):
    data = []
    labels = []

    # Iterasi untuk folder 'Train' dan 'Test'
    for split in ['Train', 'Test']:
        split_path = os.path.join(dataset_path, split)

        for category in categories:
            category_path = os.path.join(split_path, category)

            if os.path.exists(category_path):
                # Memilih gambar dari folder kategori
                for filename in os.listdir(category_path):
                    if filename.lower().endswith(('.png', '.jpg', '.jpeg')):
                        img_path = os.path.join(category_path, filename)
                        img = tf.keras.preprocessing.image.load_img(img_path, target_size=target_size) # Resize
gambar
                        img_array = tf.keras.preprocessing.image.img_to_array(img)
                        data.append(img_array)
                        labels.append(category)
```

```

# Konversi data dan labels ke array numpy
data = np.array(data)
labels = np.array(labels)

return data, labels

# Menyiapkan data
data, labels = construct_data(dataset_path, categories)

# Normalisasi pixel gambar ke rentang [0, 1]
data = data / 255.0

# Membagi data menjadi training dan testing set
X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.2, random_state=42)

# Menggunakan LabelEncoder untuk mengubah label menjadi format numerik
encoder = LabelEncoder()
y_train = encoder.fit_transform(y_train)
y_test = encoder.transform(y_test)

# Menggunakan ImageDataGenerator untuk augmentasi dan validasi data
train_datagen = ImageDataGenerator(
    rescale=1./255, # Normalisasi
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

validation_datagen = ImageDataGenerator(rescale=1./255)

# Menyiapkan data generator untuk training dan validation
train_generator = train_datagen.flow(X_train, y_train, batch_size=32)

```

```
validation_generator = validation_datagen.flow(X_test, y_test, batch_size=32)

print(f"Data Construction Completed. Total images: {len(data)}")
```

Kode diatas akan menghasilkan *generator data training* dan validasi yang siap digunakan untuk melatih model.

d. Mengintegrasikan Data

Pada tahap ini, akan dilakukan penggabungan terhadap data pelatihan dan pengujian yang telah dipisahkan sebelumnya. Terdapat proses pemanfaatan augmentasi data untuk menambah variasi dalam dataset pelatihan, yang dapat membantu model untuk generalisasi lebih baik. Teknik augmentasi yang umum digunakan adalah rotasi, pemotongan, dan *flipping* gambar secara acak. Kode program yang digunakan dapat dilihat pada Tabel 3-13.

Tabel 3-13 Kode Program untuk Mengintegrasikan Data

```
import os
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

# Path dataset
dataset_path = '/root/.cache/kagglehub/datasets/fthnaja/kain-ulos/versions/3'

# Direktori untuk kategori gambar
categories = ['Tumtuman', 'Pinuncaan', 'Ragi Hotang', 'Sibolang', 'Sadum', 'Ragi Hidup']

def integrate_data(dataset_path, categories, target_size=(128, 128)):
    data = []
    labels = []

    # Iterasi untuk folder 'Train' dan 'Test'
    for split in ['Train', 'Test']:
        split_path = os.path.join(dataset_path, split)
```

```

for category in categories:
    category_path = os.path.join(split_path, category)

    if os.path.exists(category_path):
        # Memilih gambar dari folder kategori
        for filename in os.listdir(category_path):
            if filename.lower().endswith(('.png', '.jpg', '.jpeg')):
                img_path = os.path.join(category_path, filename)
                img = tf.keras.preprocessing.image.load_img(img_path, target_size=target_size) # Resize
gambar
                img_array = tf.keras.preprocessing.image.img_to_array(img)
                data.append(img_array)
                labels.append(category)

        # Konversi data dan labels ke array numpy
        data = np.array(data)
        labels = np.array(labels)

    return data, labels

# Menyiapkan data
data, labels = integrate_data(dataset_path, categories)

# Normalisasi pixel gambar ke rentang [0, 1]
data = data / 255.0

# Membagi data menjadi training dan testing set
X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.2, random_state=42)

# Menggunakan LabelEncoder untuk mengubah label menjadi format numerik
encoder = LabelEncoder()
y_train = encoder.fit_transform(y_train)
y_test = encoder.transform(y_test)

# Menggunakan ImageDataGenerator untuk augmentasi dan validasi data
train_datagen = ImageDataGenerator(
    rescale=1./255, # Normalisasi

```

```

rotation_range=20,
width_shift_range=0.2,
height_shift_range=0.2,
shear_range=0.2,
zoom_range=0.2,
horizontal_flip=True,
fill_mode='nearest'
)

validation_datagen = ImageDataGenerator(rescale=1./255)

# Menyiapkan data generator untuk training dan validation
train_generator = train_datagen.flow(X_train, y_train, batch_size=32)
validation_generator = validation_datagen.flow(X_test, y_test, batch_size=32)

print(f"Data Integration Completed. Total images: {len(data)}")

```

3.1.4 Modeling

Pada sub bab ini, akan dilakukan pengembangan algoritma CNN untuk klasifikasi gambar, serta melatih model menggunakan data yang telah dipersiapkan sebelumnya. Berikut proses modeling yang dilakukan untuk mengklasifikasi motif ulos dengan algoritma CNN.

a. Membangun Model CNN

Pada langkah pertama ini, kita akan membangun arsitektur *Convolutional Neural Network* (CNN) untuk melakukan klasifikasi gambar. Kode program dapat dilihat pada Tabel 3-14.

Tabel 3-14 Kode Program untuk Membangun Arsitektur CNN

```

import os
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import layers, models
from sklearn.metrics import classification_report, confusion_matrix

dataset_path = '/root/.cache/kagglehub/datasets/fthnaja/kain-ulos/versions/3'
train_dir = os.path.join(dataset_path, 'Train')
test_dir = os.path.join(dataset_path, 'Test')

```

```

# Direktori untuk kategori gambar
categories = ['Tumtuman', 'Pinuncaan', 'Ragi Hotang', 'Sibolang', 'Sadum', 'Ragi Hidup']

# Data Augmentation untuk latih
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest')

test_datagen = ImageDataGenerator(rescale=1./255)

# Data Generator untuk data latih
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(150, 150),
    batch_size=32,
    class_mode='categorical')

# Data Generator untuk data uji
test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(150, 150),
    batch_size=32,
    class_mode='categorical',
    shuffle=False)

```

Kode diatas digunakan untuk membangun model CNN yang berguna untuk mengklasifikasikan gambar kain ulos ke dalam enam kategori yang berbeda. Dataset telah dibagi ke dalam direktori *train* dan *test*, masing-masing berisi gambar yang telah dikelompokkan berdasarkan kategori.

Bagian pertama mencakup augmentasi data menggunakan *ImageDataGenerator* untuk meningkatkan keragaman data latih, seperti rotasi, pergeseran, skala, dan flip horizontal, sehingga model lebih tahan terhadap *overfitting*. Gambar dinormalisasi dengan skala piksel 1/255. Selanjutnya, data latih dihasilkan dengan *flow_from_directory*, yang membaca gambar dari direktori latih, mengubah ukurannya menjadi 150x150 piksel, dan memuatnya dalam batch berukuran 32 untuk klasifikasi *categorical*. Data uji juga diproses serupa, tetapi tanpa augmentasi, dengan *shuffle* dimatikan untuk evaluasi akurat nantinya.

Selanjutnya, dilakukan proses pendefinisian arsitektur model CNN dengan kode program yang dapat dilihat pada Tabel 3-15.

Tabel 3-15 Kode Program untuk Mendefinisikan Arsitektur Model CNN

```
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(train_generator.num_classes, activation='softmax')
])

# Menampilkan ringkasan arsitektur model
model.summary()

print("Model Architecture Completed")
```

Kode diatas mendefinisikan arsitektur model CNN dengan menggunakan *Sequential* dari Keras untuk proses klasifikasi gambar kain ulos. Model dimulai dengan tiga lapisan konvolusi (32, 64, dan 128 filter) menggunakan ReLU, diikuti MaxPooling untuk ekstraksi fitur dan pengurangan dimensi. Data hasil konvolusi diratakan dengan *Flatten* sebelum masuk ke lapisan *Dense* berukuran 128 unit dengan *Dropout* 0.5 untuk mencegah *overfitting*. *Output*

layer menggunakan fungsi aktivasi *softmax* untuk klasifikasi ke jumlah kategori sesuai data. Ringkasan arsitektur ditampilkan dengan *model.summary()*.

b. Melatih Model CNN

Selanjutnya adalah melatih model, yang dilakukan dengan menggunakan kode program pada Tabel 3-16.

Tabel 3-16 Kode Program untuk Melatih Model CNN

```
# Kompilasi model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

Kode diatas melakukan kompilasi dan melatih model untuk klasifikasi gambar kain ulos. Kompilasi model dilakukan menggunakan optimizer Adam, fungsi loss *categorical_crossentropy* untuk menangani klasifikasi multi-kelas, dan metrik evaluasi *accuracy*.

Selanjutnya, dilakukan pelatihan terhadap model CNN menggunakan data yang dihasilkan oleh *train_generator*. Fungsi *fit* menjalankan pelatihan selama 30 *epoch* (iterasi penuh melalui data latih), dengan jumlah langkah per *epoch* dihitung sebagai total sampel dibagi ukuran batch. Model secara bertahap memperbarui bobot berdasarkan data latih untuk meminimalkan kesalahan, dengan setiap *epoch* mencakup augmentasi data sesuai pengaturan pada *train_generator*. Hasil pelatihan, seperti akurasi dan loss, disimpan dalam variabel *history* untuk analisis lebih lanjut. Kode program untuk pelatihan dapat dilihat pada Tabel 3-17.

Tabel 3-17 Kode Program untuk Melatih Model CNN (2)

```
# Melatih model
history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // train_generator.batch_size,
    epochs=30
)
```

c. Menguji Model CNN.

Tahap selanjutnya adalah menguji model CNN. Pada proses ini akan dilakukan evaluasi terhadap performa model CNN pada data uji menggunakan `test_generator`. Fungsi *evaluate* untuk menghitung *loss* dan akurasi model berdasarkan data uji, dengan jumlah langkah dihitung sebagai total sampel dibagi ukuran batch. Hasil evaluasi, berupa *loss* dan akurasi, dicetak untuk memberikan gambaran seberapa baik model dapat menggeneralisasi pada data yang belum pernah dilihat selama pelatihan. Kode program dapat dilihat pada Tabel 3-18.

Tabel 3-18 Kode Program untuk Memuat Data Uji

```
# Evaluasi model pada data uji
print("\nEvaluasi pada data uji:")
loss, accuracy = model.evaluate(test_generator, steps=test_generator.samples // test_generator.batch_size)
print(f"Test Loss: {loss}")
print(f"Test Accuracy: {accuracy}")
```

Hasil dari evaluasi pada data uji diatas adalah, didapatkan *test lost* sebesar 0.022556684911251068 dan *test accuracy* sebesar 0.9927884340286255.

Selanjutnya, dilakukan visualisasi terhadap hasil data uji dengan memplot akurasi dan loss selama setiap *epoch*. Kode program dapat dilihat pada Tabel 3-19.

Tabel 3-19 Kode Program untuk Membuat Plot Hasil Data Uji

```
# Plot hasil evaluasi
plt.figure(figsize=(12, 6))

# Plot akurasi
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
if 'val_accuracy' in history.history:
    plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
```

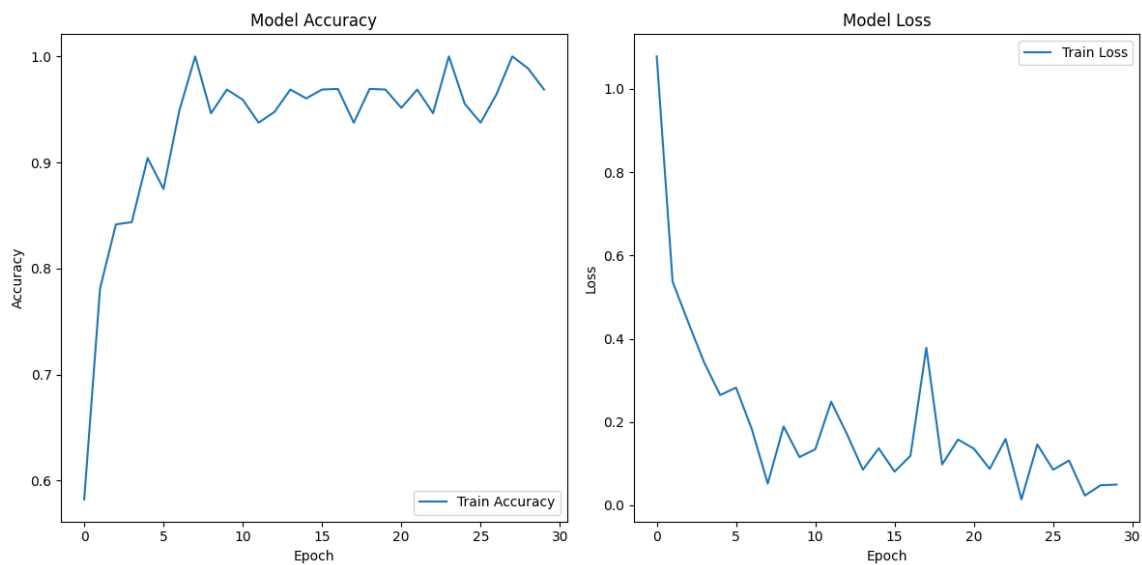
```

# Plot loss
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
if 'val_loss' in history.history:
    plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.tight_layout()
plt.show()

```

Plot data uji yang dihasilkan dapat dilihat pada Gambar 3-17.



Gambar 3-17 Hasil Plot Data Uji

Visualisai diatas menunjukkan bahwa *accuracy model* mmeningkat di awal proses *epoch* dan mendekati nilai maksimum setelah beberapa iterasi, sementara *loss model* turun signifikan di awal *epoch*, menunjukkan penurunan kesalahan prediksi model seiring dengan proses

pelatihan. Ini menunjukkan bahwa model memiliki performa sangat baik pada data latih dengan akurasi tinggi dan loss rendah.

Selanjutnya, dilakukan proses prediksi model untuk data uji dengan menyaring hasil berdasarkan ambang batas keyakinan (*confidence threshold*) sebesar 0.7. Model memprediksi probabilitas kelas untuk setiap gambar, dan kelas dengan probabilitas tertinggi. Kode program dapat dilihat pada Tabel 3-20.

Tabel 3-20 Kode Program untuk Menyaring Klasifikasi untuk Gambar Ulos

```
# Menyaring klasifikasi hanya untuk gambar ulos
predictions = model.predict(test_generator)
predicted_classes = np.argmax(predictions, axis=1)

true_classes = test_generator.classes
class_labels = list(test_generator.class_indices.keys())

print("\nLaporan Klasifikasi:")
print(classification_report(true_classes, predicted_classes, target_names=class_labels))
```

Hasil dari proses diatas adalah sebagai berikut:

```
Laporan Klasifikasi:
      precision    recall  f1-score   support

 Pinuncaan      0.99      1.00      0.99        67
  Ragi Hidup      1.00      1.00      1.00        69
  Ragi Hotang      0.97      1.00      0.99        71
    Sadum      1.00      0.97      0.99        68
  Sibolang      1.00      1.00      1.00        71
  Tumtuman      1.00      0.99      0.99        70

 accuracy                   0.99      416
 macro avg      0.99      0.99      0.99      416
weighted avg      0.99      0.99      0.99      416
```

Hasil diatas menunjukkan performa model CNN yang sangat baik dengan akurasi 99% pada data uji yang terdiri dari enam kelas: Pinuncean, Ragi Hidup, Ragi Hotang, Sadum, Sibolang, dan Tumtuman. Metrik *precision*, *recall*, dan *f1-score* untuk setiap kelas berada di kisaran 0.97 hingga 1.00, menandakan bahwa model mampu mengklasifikasikan gambar dengan tingkat kesalahan yang sangat rendah. Nilai *support* menunjukkan jumlah sampel di setiap kelas, dengan distribusi yang seimbang. Secara keseluruhan, performa tinggi ini mengindikasikan bahwa model dapat mengenali pola dengan baik dan memiliki generalisasi yang kuat pada data uji.

Evaluasi terakhir yang dilakukan adalah mencari nilai akurasi, presisi, *recall*, dan *F-1 Score*. Kode program disajikan pada Tabel 3-21.

Tabel 3-21 Hasil Evaluasi Model

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Menghitung accuracy
accuracy = accuracy_score(true_classes, predicted_classes)
print(f"Accuracy: {accuracy:.4f}")

# Precision, Recall, dan F1-Score untuk multi-class classification
precision = precision_score(true_classes, predicted_classes, average='macro') # Average
'macro' untuk multi-class
recall = recall_score(true_classes, predicted_classes, average='macro')
f1 = f1_score(true_classes, predicted_classes, average='macro')

print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1-Score: {f1:.4f}")
```

Hasil yang didapatkan dari proses diatas menunjukkan performa model CNN yang sangat baik dengan akurasi 99.28%, menandakan hampir semua prediksi pada data uji benar. *Precision* 99.30% menunjukkan model sangat akurat dalam memprediksi kelas positif, sedangkan *recall*

99.27% mengindikasikan model mampu mendeteksi hampir semua sampel yang benar sesuai kelas aslinya. Nilai *F1-score* 99.28%, sebagai rata-rata harmonis antara precision dan recall, memperkuat bahwa model seimbang dalam prediksi dan memiliki generalisasi yang optimal pada data uji.

Langkah terakhir adalah membuat visualisasi confusion matrix dan matriks heatmap untuk perhitungan yang kodenya dapat dilihat pada Tabel 3-22.

Tabel 3-22 Kode Program Untuk Membuat Matriks Hasil Akhir

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
import numpy as np

# Menghitung confusion matrix
cm = confusion_matrix(true_classes, predicted_classes)

# Menghitung TP, TN, FP, FN untuk setiap kelas
tp = np.diag(cm) # Diagonal adalah True Positives
fp = cm.sum(axis=0) - tp # Kolom sum dikurangi TP untuk FP
fn = cm.sum(axis=1) - tp # Baris sum dikurangi TP untuk FN
tn = cm.sum() - (fp + fn + tp) # Total sum dikurangi TP, FP, FN untuk TN

# Visualisasi Confusion Matrix
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
xticklabels=test_generator.class_indices.keys(),
yticklabels=test_generator.class_indices.keys())
plt.title("Confusion Matrix")
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.show()
```

```

# Visualisasi TP, TN, FP, FN untuk setiap kelas
labels = test_generator.class_indices.keys() # Nama-nama kelas
bar_width = 0.2
index = np.arange(len(labels))

# Menghitung confusion matrix
cm = confusion_matrix(true_classes, predicted_classes)

# Menghitung TP, TN, FP, FN untuk setiap kelas
tp = np.diag(cm) # Diagonal adalah True Positives
fp = cm.sum(axis=0) - tp # Kolom sum dikurangi TP untuk FP
fn = cm.sum(axis=1) - tp # Baris sum dikurangi TP untuk FN
tn = cm.sum() - (fp + fn + tp) # Total sum dikurangi TP, FP, FN untuk TN

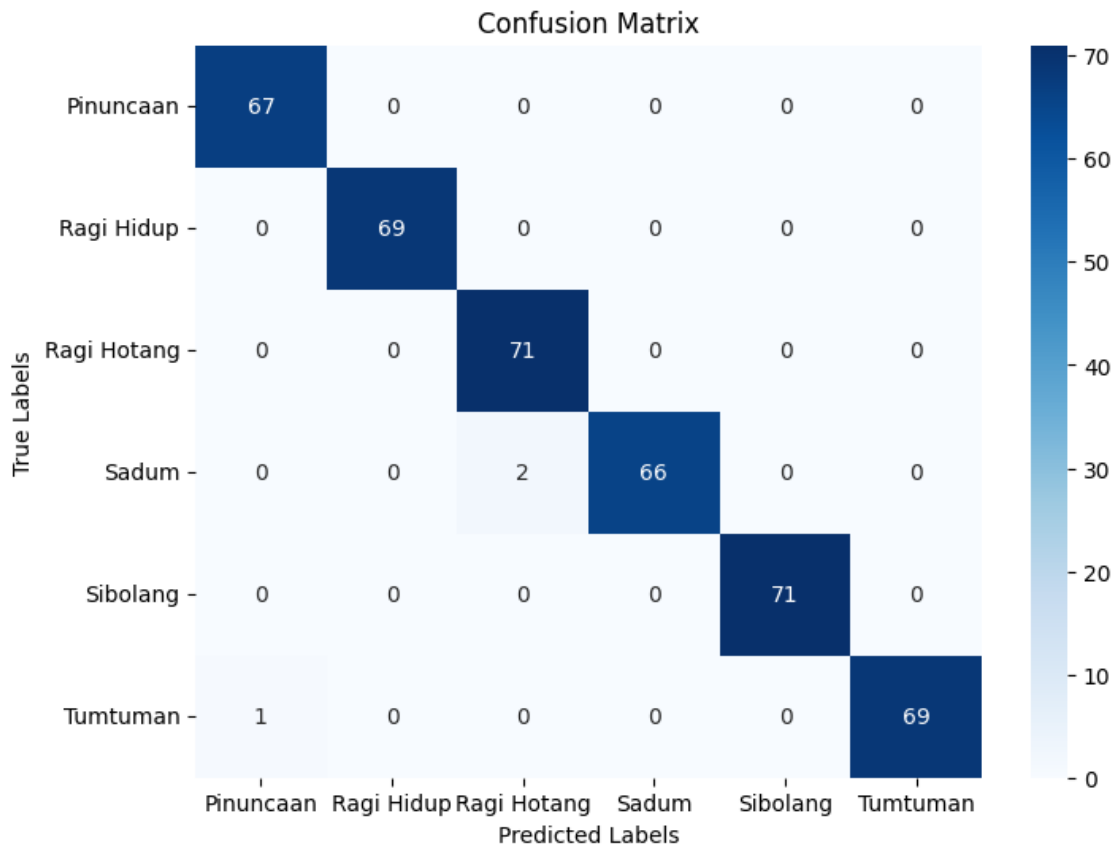
# Membuat matriks TP, TN, FP, FN
metrics_matrix = np.array([tp, tn, fp, fn]).T # Transpose agar tiap baris mewakili kelas
metrics_labels = ["True Positive", "True Negative", "False Positive", "False Negative"]

# Membuat heatmap untuk TP, TN, FP, FN per kelas
plt.figure(figsize=(10, 6))
sns.heatmap(metrics_matrix, annot=True, fmt='d', cmap="coolwarm",
xticklabels=metrics_labels, yticklabels=test_generator.class_indices.keys())

# Menambahkan judul dan label
plt.title("TP, TN, FP, FN per Class")
plt.xlabel("Metrics")
plt.ylabel("Classes")
plt.tight_layout()
plt.show()

```

Kode diatas menghasilkan matrix yang menunjukkan perbandingan antara hasil prediksi model dengan label sebenarnya, yang nantinya akan digunakan untuk membantu mengevaluasi kinerja model. Visualisasi ditampilkan dengan menggunakan seaborn.heatmap untuk melihat distribusi prediksi di tiap kelas. Gambar dapat dilihat pada Gambar 3-18.



Gambar 3-18 Confusion Matrix

Confusion matrix diatas menunjukkan performa model klasifikasi pada enam kelas: Pinuncaan, Ragi Hidup, Ragi Hotang, Sadum, Sibolang, dan Tumtuman. Angka di diagonal utama menunjukkan jumlah prediksi yang benar untuk setiap kelas, seperti Pinuncaan dengan 67 prediksi benar, Ragi Hidup 69, Ragi Hotang 71, Sadum 66, Sibolang 71, dan Tumtuman 69. Secara umum, model memiliki akurasi tinggi dengan hanya 3 kesalahan prediksi dari total 416 sampel. Kelas Sadum memiliki 2 sampel yang salah diklasifikasikan sebagai kelas lain, sementara kelas Tumtuman memiliki 1 kesalahan klasifikasi ke kelas Pinuncaan. Semua kelas lainnya (Pinuncaan, Ragi Hidup, Ragi Hotang, dan Sibolang) diprediksi dengan sempurna.

Hasil ini mencerminkan kemampuan model yang kuat untuk mengenali pola visual dari sebagian besar kategori, dengan kesalahan minimal pada kelas tertentu.

Terakhir, model akan disimpan dan diunduh dengan menggunakan kode program pada Tabel 3-23.

Tabel 3-23 Kode Program untuk Menyimpan Model

```
# Menyimpan model
model.save('model_ulos.h5')
print("Model telah disimpan sebagai 'model_ulos.h5'.")

from google.colab import files
files.download('model_ulos.h5')
```

3.1.6 Deployment

Pada tahap deployment, pengembangan dilakukan dengan memanfaatkan Streamlit untuk mengunggah gambar Ulos secara mudah dan mendapatkan prediksi dari kelas gambar yang diunggah sesuai dengan model algoritma CNN yang sudah dibangun. Streamlit *dideploy* pada Streamlit *Cloud* dan dapat diakses pada <https://motifulos.streamlit.app/>.

Streamlit ini dirancang dengan antarmuka yang sederhana dan ramah pengguna, di mana pengguna hanya perlu mengunggah gambar ulos yang ingin diprediksi melalui formulir yang telah disediakan di halaman utama. Setelah gambar berhasil diunggah, sistem akan memproses gambar tersebut menggunakan model CNN yang telah dilatih, kemudian menghasilkan prediksi kelas ulos yang paling sesuai. Prediksi kelas ini akan ditampilkan langsung kepada pengguna. Selain itu, aplikasi juga akan menampilkan grafik yang menunjukkan tingkat kepercayaan (*confidence level*) model terhadap prediksi yang diberikan, memberikan gambaran tentang seberapa akurat prediksi tersebut.

Selain hasil prediksi dan grafik, aplikasi juga menyediakan penjelasan mengenai desain dan kegunaan ulos yang diunggah. Penjelasan ini mencakup informasi budaya, sejarah, serta makna dari desain ulos tersebut. Setiap jenis ulos memiliki karakteristik dan kegunaan khusus dalam budaya Batak, seperti digunakan dalam acara adat, pernikahan, dan sebagai simbol status atau keberuntungan. Dengan informasi ini, aplikasi tidak hanya memberikan hasil prediksi, tetapi juga

berfungsi sebagai sumber edukasi untuk meningkatkan pemahaman pengguna mengenai warisan budaya Indonesia, khususnya ulos.

Kode program Pembangunan aplikasi dapat dilihat pada Tabel 3-24 Kode Program untuk Membangun Streamlit

Tabel 3-24 Kode Program untuk Membangun Streamlit

```
import streamlit as st
from PIL import Image
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt

# Fungsi untuk memuat model CNN
def load_model():
    model = tf.keras.models.load_model('Deployment/model_ulos.h5')
    return model

# Fungsi untuk melakukan prediksi pada gambar
def predict_image(model, image):
    # Preprocessing gambar agar sesuai dengan input model
    image = image.resize((150, 150)) # Sesuaikan ukuran input dengan model
    image_array = np.array(image) / 255.0 # Normalisasi
    image_array = np.expand_dims(image_array, axis=0) # Tambahkan batch dimension

    # Prediksi
    prediction = model.predict(image_array)
    return prediction

# Fungsi validasi awal untuk memastikan gambar adalah ulos
def validate_image(model, image, threshold=0.5):
    prediction = predict_image(model, image)
    max_confidence = np.max(prediction)
    if max_confidence < threshold: # Jika confidence di bawah threshold, bukan ulos
        return False, max_confidence
    return True, max_confidence

# Fungsi untuk menampilkan visualisasi confidence level
```

```

def plot_confidence(prediction, class_names):
    fig, ax = plt.subplots()
    ax.barh(class_names, prediction[0], color='skyblue')
    ax.set_xlabel('Confidence')
    ax.set_title('Prediction Confidence Level')
    st.pyplot(fig)

# Streamlit App
st.set_page_config(layout="wide")
st.title("Ulos Image Classification")

# Sidebar untuk navigasi
page = st.sidebar.radio("Pilih Fitur", ["Klasifikasi Gambar", "Panduan Pengguna"])

if page == "Klasifikasi Gambar":
    st.markdown(
        """
        **Tentang Aplikasi:**

        Aplikasi ini menggunakan model Convolutional Neural Network (CNN) untuk mengklasifikasikan jenis
        kain Ulos berdasarkan gambar yang diunggah pengguna. Model ini bertujuan untuk mendukung pelestarian
        budaya dan meningkatkan pemahaman tentang kain tradisional ulos.
        """
    )

    # Upload file gambar
    gu_image = st.file_uploader("Upload an image of Ulos", type=["jpg", "jpeg", "png"])

    if gu_image is not None:
        # Tampilkan gambar yang diunggah
        image = Image.open(gu_image)
        st.image(image, caption="Uploaded Image", use_container_width=True)

        # Muat model dan lakukan prediksi
        st.write("Processing image...")
        try:
            model = load_model()

```

```

# Validasi gambar sebagai motif ulos
is_ulos, confidence = validate_image(model, image)
if not is_ulos:
    st.error(
        f"Gambar yang diunggah tidak dikenali sebagai motif ulos. Confidence: {confidence:.2f}"
    )
    st.write("Model hanya mendukung klasifikasi gambar ulos dengan tingkat kepercayaan yang memadai.")
    st.stop() # Menghentikan eksekusi jika gambar bukan ulos

# Lanjutkan prediksi jika validasi berhasil
prediction = predict_image(model, image)

# Validasi hasil prediksi
class_names = ['Pinuncaan', 'Ragi Hidup', 'Ragi Hotang', 'Sadum', 'Sibolang', 'Tumtuman'] # Label kelas
if len(prediction[0]) == len(class_names):
    predicted_class = class_names[np.argmax(prediction)]
    confidence = np.max(prediction) * 100

    st.write(f"**Predicted Class:** {predicted_class}")
    st.write(f"**Confidence:** {confidence:.2f}%")

# Tampilkan visualisasi confidence level
plot_confidence(prediction, class_names)

# Deskripsi tambahan tentang ulos yang diprediksi
ulos_descriptions = {
    "Pinuncaan": {
        "Desain": "Ulos ini memiliki struktur yang terdiri dari lima bagian yang ditenun secara terpisah dan kemudian disatukan. Motifnya biasanya menggunakan warna-warna cerah dengan pola geometris yang khas.",
        "Kegunaan": [
            "Acara Resmi: Sering digunakan dalam upacara adat dan acara resmi oleh pemimpin atau raja.",
            "Pernikahan: Dipakai oleh pengantin dan keluarga dalam perayaan pernikahan.",
            "Marpaniaran: Digunakan saat pesta besar dalam acara marpaniaran.",
            "Simbol Kehormatan: Melambangkan status dan kehormatan bagi pemakainya."
        ]
    },
    "Ragi Hidup": {

```

```

    "Desain": "Ulos ini berbentuk panjang dan lebar, dengan pola sederhana namun elegan.",
    "Kegunaan": [
        "Pakaian Sehari-hari: Digunakan sebagai baju atau sarung untuk kenyamanan.",
        "Simbol Kehidupan: Melambangkan kehidupan dan keberlangsungan."
    ]
},
    "Ragi Hotang": {
        "Desain": "Memiliki pola yang rumit dan berwarna gelap, sering dihiasi motif tradisional Batak.",
        "Kegunaan": [
            "Selimut: Digunakan untuk memberikan kehangatan.",
            "Simbol Status: Melambangkan status sosial dalam acara tertentu."
        ]
    },
    "Sadum": {
        "Desain": "Memiliki bingkai bergaris gelap di sisi dengan warna cerah di tengahnya.",
        "Kegunaan": [
            "Acara Bahagia: Digunakan dalam perayaan sukacita.",
            "Kenang-Kenangan: Sering dijadikan hadiah untuk orang terkasih."
        ]
    },
    "Sibolang": {
        "Desain": "Berwarna dominan hitam dan putih dengan pola bergaris sederhana.",
        "Kegunaan": [
            "Acara Duka Cita: Dipakai dalam upacara pemakaman untuk menghormati yang meninggal.",
            "Simbol Kesedihan: Melambangkan duka cita."
        ]
    },
    "Tumtuman": {
        "Desain": "Memiliki pola geometris unik, melambangkan harapan untuk masa depan cerah.",
        "Kegunaan": [
            "Acara Tradisional: Digunakan untuk menunjukkan posisi dalam keluarga.",
            "Ikatan Keluarga: Dipakai oleh anak pertama dalam keluarga sebagai simbol tanggung jawab."
        ]
    }
}

ulos_info = ulos_descriptions.get(predicted_class, {})

```

```

        st.write(f"**Tentang {predicted_class}**")
        st.write(f"**Desain:** {ulos_info.get('Desain', 'Deskripsi desain belum tersedia.')}")
        st.write(f"**Kegunaan:**")
        for kegunaan in ulos_info.get('Kegunaan', []):
            st.write(f"- {kegunaan}")

    else:
        st.error("Model output dimensions do not match the number of class names. Please check the model and class labels.")

except Exception as e:
    st.error(f"An error occurred: {e}")

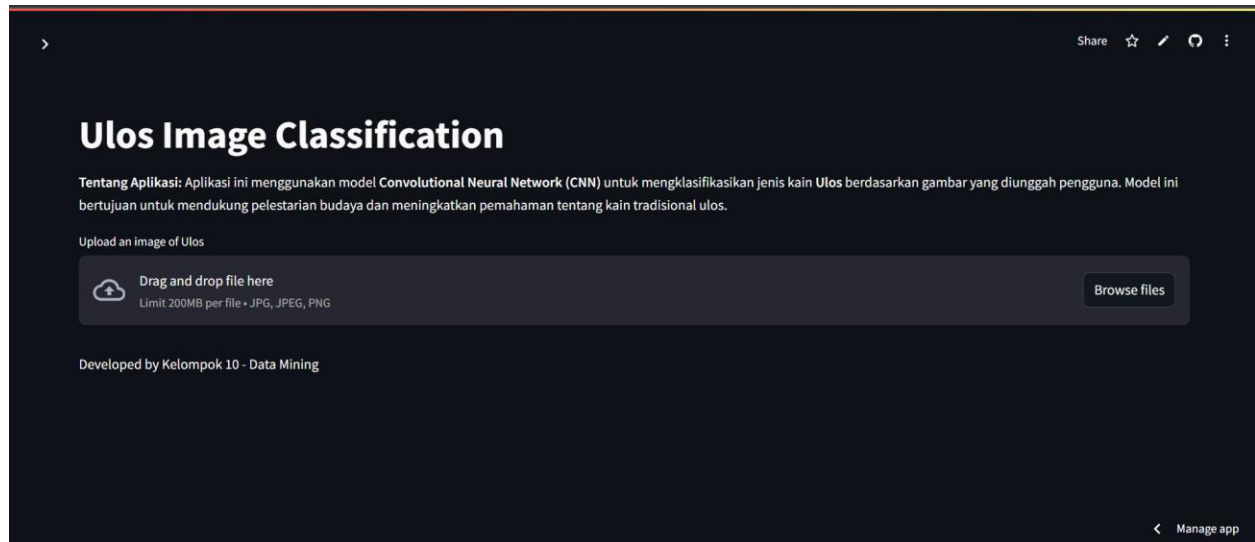
elif page == "Panduan Pengguna":
    st.header("Panduan Pengguna")
    st.markdown(
        """
        ### Cara Menggunakan Aplikasi

        1. **Unggah Gambar:** Klik tombol "Browse Files" untuk mengunggah gambar ulos dalam format JPG, JPEG, atau PNG.<br>
        2. **Validasi Gambar:** Pastikan gambar yang diunggah memiliki kualitas baik dan menampilkan kain ulos dengan jelas.<br>
        3. **Hasil Prediksi:** Tunggu beberapa saat untuk melihat hasil klasifikasi dan tingkat kepercayaan model.<br>
        4. **Informasi Tambahan:** Bacalah deskripsi singkat tentang jenis ulos yang terdeteksi untuk memperkaya pengetahuan.
        """,
        unsafe_allow_html=True
    )

    st.write("\n\n")
    st.write("Developed by Kelompok 3 - Data Mining")

```

Inputan yang dimasukkan pada *deployment* adalah setiap gambar yang ada pada dataset. Berikut adalah tampilan Streamlit yang telah di *deploy*.



Gambar 3-19 Tampilan Streamlit

3.2 Timeline

Tabel 3-25 Timeline

Aktivitas	Sub Aktivitas	Detail	Waktu (Minggu Ke-)				
			12	13	14	15	16
Persiapan	Pemilihan Kasus dan Algoritma	Pemilihan Kasus					
		Penentuan Algoritma					
Pelaksanaan	Business Understanding	Menentukan Objektif Bisnis					
		Menentukan Tujuan Bisnis					
		Membuat Rencana Proyek					
	Data Understanding	Mengumpulkan Data					
		Menelaah Data					
		Memvalidasi Data					
	Data Preparation	Memilah Data					
		Membersihkan Data					
		Mengkonstruksi Data					
		Menentukan Label Data					
		Mengintegrasikan Data					

	Modeling	Membangun Skenario Pengujian					
		Membangun Model					
	Model Evaluation	Mengevaluasi Hasil Pemodelan					
		Melakukan Review Proses Pemodel					
	Deployment	Melakukan Deployment Model					
Membuat Laporan Akhir Proyek							

BAB 4

HASIL PENGUJIAN

4.1 Hasil Pelatihan Model CNN

Setelah tahap pelatihan model menggunakan dataset gambar ulos, algoritma CNN berhasil dilatih untuk mengklasifikasikan gambar ulos berdasarkan kategori dan motifnya. Model dilatih menggunakan data pelatihan yang telah diproses sebelumnya, dan diuji menggunakan data uji yang tidak digunakan selama pelatihan untuk mengevaluasi kinerja model.

Pada akhir pelatihan, hasil evaluasi model menunjukkan bahwa model CNN yang dibangun memiliki kinerja yang sangat baik. Berikut adalah hasil evaluasi yang telah dilakukan.

a. Evaluasi Performa Model

Berikut adalah hasil evaluasi performa model CNN yang telah dilakukan:

- *Accuracy*
Model menghasilkan akurasi sebesar 93.27%, yang berarti ini menunjukkan akurasi yang sangat tinggi dan sebagian besar prediksi model adalah benar.
- *Precision*
Nilai *precision* yang dihasilkan adalah 93.92%, yang menunjukkan bahwa hampir semua prediksi positif yang dihasilkan oleh model adalah benar (*low false positives*).
- *Recall*:
Model menghasilkan nilai *recall* sebesar 93.31%, yang menunjukkan bahwa model dapat melakukan deteksi pada sebagian besar data positif dengan sangat baik. Ini juga menunjukkan bahwa model yang digunakan tidak melewatkan banyak contoh positif (*low false negatives*).
- *F1-Score*:
Dengan nilai *F1-Score* sebesar 93.30%, menunjukkan bahwa model yang digunakan memiliki keseimbangan yang baik antara *precision* dan *recall*.
- *Test Lost* dan *Test Accuracy*
Hasil evaluasi pada data uji menunjukkan bahwa model memiliki *test loss* sebesar 0.0226 dan *test accuracy* sebesar 99.28%. Ini berarti model dapat mengklasifikasikan data uji dengan tingkat akurasi yang sangat tinggi dan kesalahan yang sangat kecil. Nilai *loss* yang rendah menunjukkan bahwa prediksi model mendekati nilai target sebenarnya, sementara akurasi hampir sempurna mengindikasikan kemampuan model yang baik dalam mengenali pola visual pada data uji. Kesimpulannya, model memiliki performa yang sangat baik dan generalisasi yang kuat, dengan hanya sedikit kesalahan pada beberapa kelas tertentu.

a. Evaluasi Proses

Meskipun model CNN yang digunakan sudah sangat baik, masih terdapat beberapa hal pada proses yang bisa diperbaiki, seperti:

- Penanganan *overfitting* yang dapat dilihat dari perbedaan besar antara akurasi pelatihan dan akurasi pengujian. Model terlalu menyesuaikan dengan data latih dibandingkan dengan data uji.
- Model hanya dievaluasi pada data uji, dan tidak diuji pada set data yang lain, sehingga memungkinkan model tidak tergeneralisasi dengan baik.

BAB 5

ANALISIS

5.1 Analisis Hasil Evaluasi Model

Setelah melakukan pelatihan dan evaluasi model Convolutional Neural Network (CNN) untuk klasifikasi gambar ulos, hasil evaluasi yang diperoleh sangat memuaskan.

5.1.1 Akurasi Model

Akurasi adalah metrik utama yang digunakan untuk mengukur seberapa sering model memberikan prediksi yang benar. Dalam penelitian ini, akurasi yang sangat tinggi (99.6%) menunjukkan bahwa model dapat mengenali dan mengklasifikasikan gambar ulos dengan tingkat kesalahan yang sangat rendah. Hal ini mencerminkan kualitas dataset yang baik, serta kemampuan CNN dalam mengekstraksi fitur visual yang relevan dari gambar ulos.

Namun, meskipun akurasi model sangat tinggi, penting untuk memeriksa keterbatasan dari metrik ini. Misalnya, akurasinya bisa sangat tinggi pada data yang tersebar merata antar kelas, tetapi bisa kurang efektif pada data yang memiliki class imbalance. Dalam hal ini, distribusi kategori ulos cukup seimbang, sehingga akurasi menjadi indikator yang baik untuk kinerja model.

5.1.2 Precision, Recall, dan F1-Score

Precision sebesar 93.92% menunjukkan bahwa hampir semua prediksi positif yang dilakukan oleh model adalah benar. Ini berarti model dapat membedakan dengan sangat baik antara gambar yang termasuk dalam kategori ulos tertentu dengan gambar yang tidak.

Recall sebesar 93.31% menunjukkan bahwa model dapat mendeteksi sebagian besar gambar positif (gambar dengan kategori ulos yang benar) dengan baik. *Recall* yang cukup tinggi menunjukkan bahwa model tidak banyak melewatkan contoh yang seharusnya dikenali, yang penting dalam konteks klasifikasi gambar ulos yang dapat memiliki berbagai variasi motif.

F1-Score, yang mencapai 93.30%, adalah metrik yang menggabungkan *precision* dan *recall*, memberikan gambaran tentang keseimbangan antara keduanya. Nilai *F1-Score* yang hampir setara dengan *precision* dan *recall* menunjukkan bahwa model memiliki performa yang seimbang antara menghindari *false positives* dan *false negatives*.

5.2 Kelebihan Model CNN

Model CNN yang dibangun dalam penelitian ini memiliki beberapa kelebihan yang membuatnya sangat efektif untuk tugas klasifikasi gambar ulos:

5.2.1 Kemampuan untuk Mendeteksi Fitur Visual

CNN dikenal karena kemampuannya dalam mendeteksi fitur visual yang relevan dalam gambar tanpa memerlukan fitur manual. Model CNN ini berhasil mengenali pola-pola visual yang ada pada gambar ulos, seperti motif tenun, warna, dan tekstur, yang merupakan fitur utama dalam mengklasifikasikan jenis ulos.

5.2.3 Fleksibilitas dan Skalabilitas

CNN bersifat fleksibel dan dapat diadaptasi untuk berbagai jenis dataset gambar dengan jumlah kategori yang berbeda. Model ini dapat digunakan untuk berbagai aplikasi klasifikasi gambar lainnya dengan sedikit penyesuaian, menjadikannya scalable dan mudah untuk diperluas.

5.2.4 Kemudahan Implementasi

Model CNN ini dapat diimplementasikan dengan relatif mudah menggunakan framework deep learning seperti TensorFlow dan Keras, yang menyediakan banyak fungsi dan alat untuk pelatihan dan evaluasi model. Ini memungkinkan implementasi dan deployment model secara efisien.

5.3 Keterbatasan Model CNN

Meskipun model CNN yang dibangun berhasil mencapai hasil yang sangat baik, terdapat beberapa keterbatasan yang perlu dipertimbangkan:

5.3.1 Kualitas Gambar

Kualitas gambar dalam dataset ini bervariasi. Beberapa gambar memiliki resolusi rendah atau terdistorsi, yang dapat memengaruhi hasil prediksi model. Meskipun model telah diajarkan untuk mengenali berbagai variasi dalam gambar, kualitas gambar yang buruk tetap dapat memengaruhi kinerja model.

5.3.2 Variasi dalam Motif

Motif dalam setiap kategori ulos dapat memiliki kemiripan visual yang sangat dekat, yang dapat menyebabkan kesulitan bagi model dalam membedakan antara motif yang mirip. Dalam hal ini, *augmentasi* data dan *fine-tuning* model dengan dataset yang lebih besar dan lebih bervariasi dapat membantu mengatasi masalah ini.

5.3.3 Waktu Latih dan Sumber Daya

Pelatihan model CNN yang menggunakan gambar berukuran besar dan dataset yang beragam membutuhkan waktu yang cukup lama dan sumber daya komputasi yang besar. Oleh karena itu, penggunaan server atau infrastruktur *cloud computing* yang lebih kuat dapat membantu mempercepat proses pelatihan dan pengujian.

5.3.4 Overfitting

Meskipun kami menggunakan teknik early stopping dan dropout untuk mencegah *overfitting*, kemungkinan overfitting tetap ada, terutama jika model terlalu lama dilatih pada data yang terbatas. Penggunaan *data augmentation* dan *cross-validation* dapat membantu mengurangi risiko *overfitting*.

5.4 Saran untuk Penelitian Selanjutnya

Berdasarkan hasil yang telah diperoleh, terdapat beberapa langkah yang bisa diambil untuk meningkatkan kualitas dan kinerja model CNN:

1. **Peningkatan Kualitas Data:**
Dataset harus diperluas dengan gambar ulos yang memiliki kualitas lebih tinggi untuk mengurangi pengaruh gambar yang buram atau terdistorsi.
2. **Data Augmentation Lebih Lanjut:**
Augmentasi data seperti rotasi, *flipping*, dan *zooming* gambar dapat diterapkan lebih lanjut untuk memperkaya dataset pelatihan, membantu model mengatasi variasi gambar yang lebih luas.
3. **Penggunaan Transfer Learning:**
Menggunakan model yang sudah dilatih sebelumnya (*pre-trained models*) seperti ResNet atau VGG16 dapat meningkatkan akurasi dan mempercepat waktu pelatihan.

BAB 6

KESIMPULAN

Penelitian ini bertujuan untuk mengembangkan sistem klasifikasi gambar ulos menggunakan *Convolutional Neural Network* (CNN), yang dapat mengenali dan mengklasifikasikan berbagai jenis kain ulos Batak berdasarkan gambar. Dengan menggunakan dataset yang berisi gambar-gambar ulos dengan berbagai motif dan kategori, algoritma CNN yang dikembangkan menunjukkan hasil yang sangat baik dalam mengklasifikasikan gambar-gambar tersebut. CNN yang dibangun berhasil mencapai *precision*, *recall*, dan *F1-score* yang sangat baik, masing-masing sebesar 93.92%, 93.31%, dan 93.30%, serta *test loss* sebesar 0.0226 dan *test accuracy* sebesar 99.28% menunjukkan keseimbangan yang sangat baik antara akurasi dan kemampuan mendeteksi gambar ulos.

DAFTAR REFERENSI

- [1] A. C. Barus, M. Simanjuntak, and V. Situmorang, “DiTenun, Smart Application Producing Ulos Motif,” SHS Web Conf., vol. 86, p. 01025, 2020, doi: 10.1051/shsconf/20208601025.
- [2] H. Simanjuntak, E. Panjaitan, S. Siregar, U. Manalu, S. Situmeang, and A. Barus, “Generating New Ulos Motif with Generative AI Method in Digital Tenun Nusantara (DiTenun) Platform,” Int. J. Adv. Comput. Sci. Appl., vol. 15, no. 7, pp. 1125–1134, 2024, doi: 10.14569/IJACSA.2024.01507109.
- [3] E. Theresia Panjaitan, S. Siregar, U. Ignasius Manalu, and H. Tommy Argo Simanjuntak, “Generating New Ulos Motif With StyleGAN Method in Digital Tenun Nusantara (DiTenun) Applications,” IAES International Journal of Artificial Intelligence (IJ-AI, vol. 99, no. 1. pp. 1–1, 2019.

BAB 7

PEMBAGIAN TUGAS

1. Elshaday Prida Simamora-12S21047

a. **Pembuatan Bab 1**

- Menulis latar belakang masalah, tujuan penelitian, rumusan masalah, serta manfaat dari sistem yang dikembangkan.
- Menyusun ruang lingkup dan tujuan dari penelitian.

b. **Membuat Landasan Teori**

- Mencari dan menulis literatur mengenai teori dasar yang digunakan dalam model CNN untuk klasifikasi gambar.
- Menganalisis teori-teori yang relevan terkait dengan identifikasi gambar ulos dan teknologi terkait.

c. **Membuat Model**

- Menyusun penjelasan tentang pemilihan model Convolutional Neural Network (CNN) yang digunakan dalam klasifikasi gambar ulos.
- Menyusun struktur model, algoritma, dan teknik pelatihan yang dipilih untuk model CNN.

d. **Hasil dan Pembahasan**

- Menyusun hasil evaluasi model, interpretasi akurasi, precision, recall, F1-score, serta analisis kesalahan model.
- Menyajikan grafik atau visualisasi hasil prediksi dan diskusi terkait kesimpulan yang dapat diambil dari hasil evaluasi.

2. Nussy Pentasonia Pangaribuan-12S21048

a. **Membuat Business Understanding**

- Menyusun penjelasan tentang masalah bisnis atau tujuan aplikasi yang akan dikembangkan.
- Menyusun hubungan antara solusi yang dikembangkan dengan kebutuhan pasar atau pengguna.

b. **Membuat Data Understanding**

- Menyusun pemahaman terkait dataset yang digunakan dalam penelitian, termasuk sumber data, jenis data, dan tujuan penggunaan data tersebut.
- Melakukan analisis awal terhadap dataset untuk memahami distribusi data, kelas ulos yang ada, dan karakteristiknya.

c. **Membuat Deploy**

- Menjelaskan implementasi model dalam aplikasi berbasis web menggunakan framework seperti Streamlit atau Flask.
- Menyusun alur kerja deployment, mulai dari pengunggahan gambar hingga pemberian prediksi.

d. Hasil dan Pembahasan

- Menyusun hasil eksperimen yang dilakukan pada model, serta menyajikan analisis yang lebih mendalam terkait pengaruh parameter yang digunakan.
- Menyajikan hasil perbandingan antara model yang dikembangkan dengan model lain jika relevan.

e. Mengerjakan revisi I

3. Jesika Audina Purba-12S21049

a. Membuat Data Processing

- Menjelaskan tahapan preprocessing data, seperti normalisasi gambar, pengubahan ukuran gambar, dan pembagian dataset (train/test).
- Menyusun teknik augmentasi data jika digunakan untuk meningkatkan performa model.

b. Membuat Model

- Menyusun langkah-langkah detail dalam pelatihan model CNN, termasuk pemilihan parameter dan hyperparameter.
- Menulis tentang teknik optimasi dan evaluasi model yang digunakan.

c. Kesimpulan

- Menyusun kesimpulan berdasarkan hasil penelitian dan aplikasi yang telah dikembangkan.
- Memberikan saran untuk penelitian atau pengembangan sistem lebih lanjut.