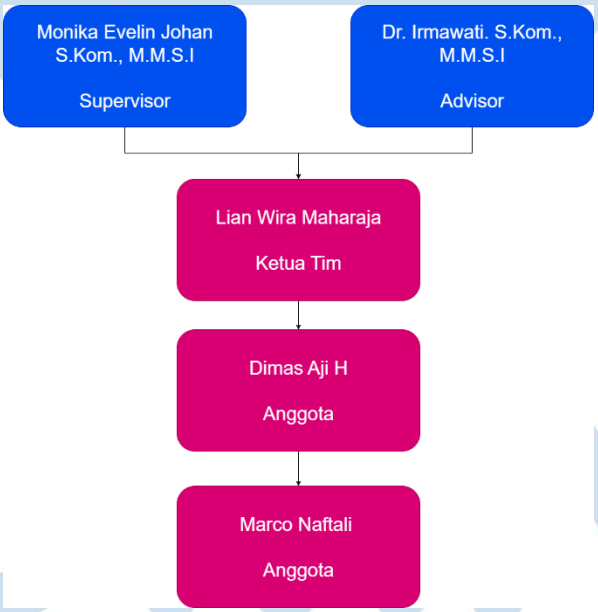
BAB III

PELAKSANAAN PRO-STEP : ROAD TO CHAMPION

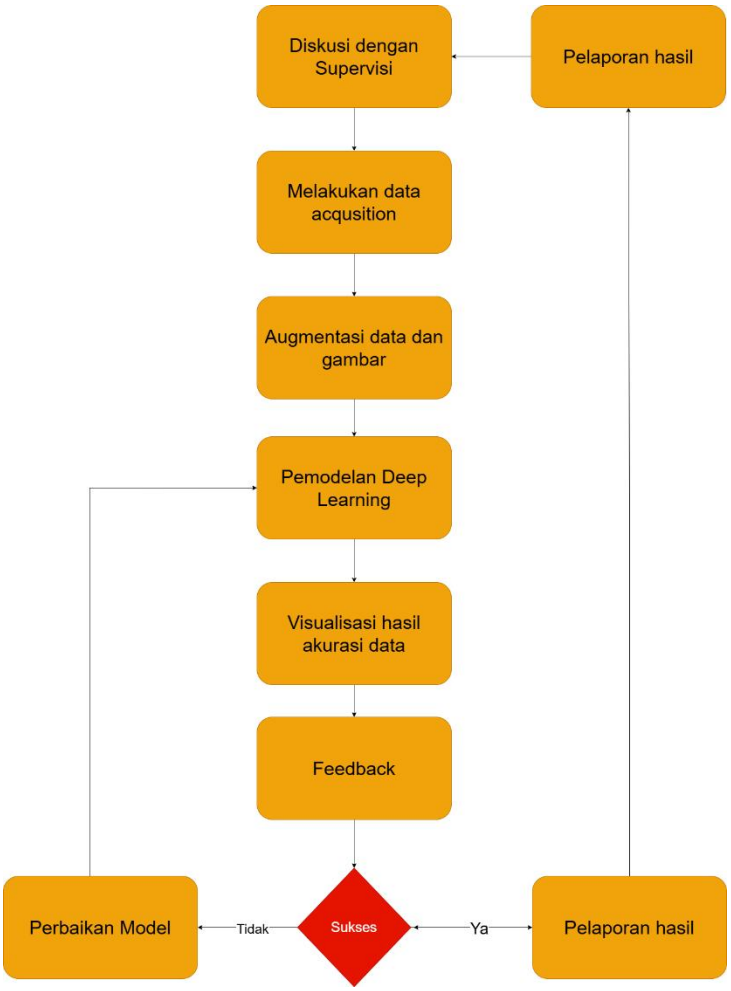
3.1. Kedudukan dan Koordinasi

Pada bagian kedudukan dan koordinasi untuk perlombaan Data Science LOGIKA UI 2025, maka dibagi menjadi 3 individu yaitu 1 ketua tim / regu dan 2 anggota. Lian Wira Maharaja sebagai ketua tim untuk perlombaan Data Science Logika UI 2025 serta ibu Monika Evelin Johan S.Kom., M.M.S.I sebagai dosen pembimbing kompetisi / Supervisor pada tim Pro Step : Road To Champion dan ibu Dr. Irmawati. S.Kom., M.M.S.I sebagai dosen Pembimbing laporan Pro Step: Road To Champion. Berikut flow kedudukan dan koordinasi Pro Step : Road To Champion.



Gambar 3. 1 Kedudukan

Penugasan Road To Champion dilakukan dengan pembagian sejumlah 3 komponen sesuai dengan isi kelompok yang mendapat tugas dan tanggung jawab pada Project Road To Champion. Pada Road To Champion, Supervisor memiliki tanggung jawab memberikan arahan berupa pembelajaran materi serta bertangggung jawab membimbing tim terhadap lomba.



Gambar 3. 2 Koordinasi Lomba

Dalam tahap koordinasi, tim membagi dalam 3 pekerjaan yang berbeda agar pekerjaan project lomba Logika UI 2025 bisa berjalan dengan baik. Tim melakukan research dan juga analisa terhadap deskripsi pekerjaan sebelum melakukan kegiatan lomba. Berikut deskripsi pekerjaan tim yang dibagi.

Tabel 3.1. Deskripsi Pekerjaan Tim (Semasa Lomba / Kompetisi)

No	Nama Tim	Deskripsi Pekerjaan
1	Lian Wira Maharaja	Melakukan optimasi data yang telah dianalisa untuk dijadikan kesimpulan dan mengumpulkan data serta melakukan data akuisisi untuk membangun data

		pipeline yang efisien
2	Marco Naftali Stevenson	Menyusun visualisasi data, serta mengembangkan dashboard untuk penyajian insight hasil analisis.
3	Dimas Aji Haritson	Mendesain dan mengimplementasikan model deep learning berbasis Convolutional Neural Network (CNN) untuk klasifikasi citra rumah adat Nusantara..

Selain itu juga, tim melakukan pekerjaan setelah kompetisi dan terbagi menjadi beberapa pekerjaan yang sudah ditentukan. Berikut pekerjaan tim setelah kompetisi LOGIKA UI 2025.

Table 3.2. Pekerjaan tim (Setelah Kompetisi)

No	Nama	Pekerjaan
1	Lian Wira	Melakukan optimasi pekerjaan tim serta optimasi hasil prediksi.
2	Marco Naftali	Melakukan komparasi model transfer learning untuk menambah insight dan melihat perbandingan dari akurasi dari hasil yang dikumpulkan
3	Dimas Aji	Melakukan implementasi hasil yang dikerjakan dengan melakukan deployment 3 hasil komparasi ke dalam prediksi dan dimasukan ke dalam CSV atau Excel

Pekerjaan setelah kompetisi dilakukan untuk mengoptimalkan hasil yang sudah dikumpul sehingga untuk berupaya menaikan hasil akurasi yang kurang memuaskan. Selain itu juga, dalam masa lomba dan masa setelah lomba, tim dibimbing oleh dua dosen pembimbing yaitu Supervisor sebagai pembimbing lapangan dan Advisor sebagai pembimbing untuk penulisan. Berikut table untuk

nama dan penjelasan pekerjaan dosen pembimbing.

Tabel 3.3. Deskripsi Dosen Pembimbing

Nama	Peran Pembimbing	Deskripsi
Dr. Irmawati. S.Kom., M.M.S.I	Advisor	Memberikan bimbingan akademik, evaluasi, serta memastikan keaslian dan relevansi proyek dengan bidang ilmu yang digeluti.
Monika Evelin Johan S.Kom., M.M.S.I	Supervisor	Memberikan arahan teknis terkait pengembangan model machine learning dan pengelolaan data serta engevaluasi efisiensi implementasi serta kinerja model dalam konteks kompetisi.

3.1.1. Pencatatan Rangkuman Mingguan Proses *PRO-STEP: Road To Champion Program*

Berisi tabel hal-hal yang penulis lakukan atau kerjakan (berisi nama proyek atau jenis pekerjaan) dalam *PRO-STEP : Road To Champion Program*.

Tabel 3.4 Detail Pekerjaan yang Dilakukan *PRO-STEP : Road to Champion Program*

No.	Minggu	Proyek	Keterangan
1	1	Mendapatkan brief lomba	Mendapatkan brief, memahami alur lomba, serta aturan penilaian yang diberikan oleh penyelenggara LOGIKA UI.

2	2	Brainstorming lomba dengan pembimbing	Diskusi bersama dosen pembimbing untuk memahami ruang lingkup lomba dan menentukan arah pengerjaan.
3	3	Penentuan tema & pembagian jobdesk	Menetapkan topik awal dan membagi peran tiap anggota tim sesuai keahlian.
4	4	Pembuatan data pipeline tahap awal	Mendesain *ImageDataGenerator Pipeline* dan melakukan koordinasi dengan supervisor.
5	5-6	Pembagian dataset lomba dari pihak panitia lomba	Kelompok melakukan pembagian untuk melakukan analisa untuk hasil lomba
6	7-8	Melakukan klasifikasi dan prediksi gambar	Lakukan klasifikasi gambar untuk mendpatkan hasil prediksi yang baik dan akurasi yang tinggi
7	9-10	Evaluasi hasil	Lakukan evaluasi hasil dan lakukan visualisasi data akurasi
8	11-12	Komparasi model	Melakukan komparasi model transfer learning untuk mendapatkan akurasi yang tinggi

3.2. Uraian Pelaksanaan Kerja Dalam *PRO-STEP : Road To Champion Program*

Selama kegiatan PRO-STEP jalur lomba, penulis mengikuti serangkaian proses kompetisi yang terdiri dari perencanaan, pengembangan model, evaluasi, hingga penyusunan laporan. Aktivitas dikerjakan melalui koordinasi tim, bimbingan pembimbing, serta supervisi teknis secara berkala.

3.2.1. Proses Pelaksanaan

Pada tahapan ini, akan menjelaskan tentang proses pelaksanaan dimulainya Pro Step Road To Champion hingga terselesaikan proses Pro Step Road To Champion. Proses pelaksanaan kegiatan Pro Step Road To Champion dibagi menjadi 3 Tahapan pendaftaran serta terdapat 4 Tahapan dalam pengerjaan project lomba. Berikut table tahapan proses pelaksanaan Pro Step Road To Champion.

Table 3.5. Tahapan Lomba

Tahapan	Tanggal	Kegiatan
1	1 September	Pendaftaran Lomba
2	15 September - 11 Oktober	Pengerjaan Lomba
3	November	Penyisihan

Tahapan lomba dilakukan sesudah mendapat bimbingan dari Dosen Supervisi dan dosen Pembimbing, serta tahapan lomba mendapat persetujuan dari pihak kampus untuk melaksanakan kegiatan terutama dari pihak *Student Development*. Berikut table tahapan pengerjaan project.

Table 3.6. Tahapan pengerjaan project

Tahapan	Waktu	Kegiatan
1	25 Agustus - 11 September	Diskusi Project
2	15 September - 11 Oktober	Pengerjaan Project
3	13 Oktober - 1 November	Komparasi model
4	2 November – 25 November	Pengerjaan setelah lomba
5	10 Desember 2025	Finalisasi

Pekerjaan dilakukan secara bertahap dan terstruktur. Ini dimulai dengan diskusi proyek pada akhir Agustus hingga pertengahan September untuk meningkatkan pemahaman, menentukan jalan, dan merumuskan kebutuhan dan tujuan proyek. Tahap selanjutnya adalah pengerjaan proyek pada pertengahan September hingga awal Oktober, yang berfokus pada proses mengembangkan dan menerapkan ide menjadi hasil yang konkret. Selanjutnya, pada periode Oktober hingga awal November dilaksakan Untuk memastikan bahwa hasil akhir siap digunakan dan sesuai dengan tujuan yang telah ditetapkan, semua pekerjaan kemudian ditutup dengan tahap finalisasi pada Desember 2025.

3.3.2 Tahapan Pengerjaan Project

3.3.2.1 Tahap 1. Diskusi Project

Pada tahapan ini, tim diberikan waktu oleh dosen pembimbing serta dosen supervisi untuk berdiskusi terkait lomba yang akan dilakukan terutama dalam pengarahan untuk kegiatan lomba. Tahap diskusi project juga dilakukan oleh tim untuk pembagian tugas serta persiapan yang harus dilakukan oleh tim sebelum masa pengerjaan project lomba.

Gambar 3. 3 Bimbingan Pertama

Tahapan diskusi project dilaksanakan oleh tim dan dosen pembimbing untuk memberikan arahan dan tujuan dari project lomba yang akan dilaksanakan, maka tim diberikan 3 pekerjaan yang berbeda, yaitu.

Jenis Pekerjaan	Nama
Data Engineering (Optimasi Transfer Learning)	Lian Wira Maharaja
Data Analyst & Visualisasi (Komparasi Transfer Learning)	Marco Naftali
Data Scientist (Implementasi Transfer Learning)	Dimas Aji

Table 3.6. Jenis pekerjaan tim

3.3.2.2 Tahap 2. Pengerjaan Project Lomba

Tahap pengerjaan project dilakukan bersama oleh kelompok sesuai

dengan job description yang telah diberikan. setelah dataset diberikan oleh pihak panitia, maka tim melakukan percobaan analisa dengan model yang berbeda. Pihak panitia LOGIKA UI memberikan dataset berupa gambar kebudayaan adat yang terbagi dalam 5 kelas yaitu Bali, Jawa, Batak, Dayak, Minangkabau. Tujuan dari panitia memberikan dataset berupa gambar adat kebudayaan adalah mendorong para peserta untuk mengasah keterampilan mereka dalam bidang *data science* khususnya *computer vision*. Pada pengerjaan project ini dilakukan untuk melakukan klasifikasi gambar adat kebudayaan dan melakukan prediksi akurasi pada klasifikasi gambar.

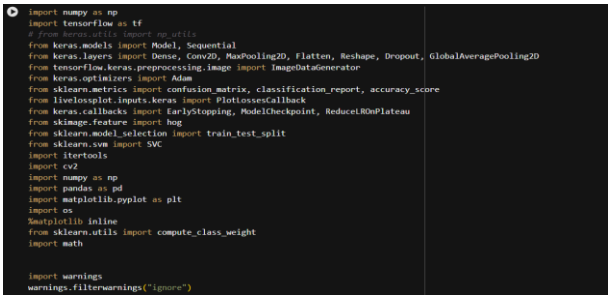
Python digunakan oleh tim sebagai alat utama dalam proses analisis dan pemodelan data karena fleksibilitas dan kelengkapan librarynya yang mendukung kebutuhan ilmu data dan visi komputer. Ini digunakan mulai dari tahap eksplorasi dataset, preprocessing gambar, hingga proses modeling dan evaluasi performa model. Beberapa library yang digunakan antara lain untuk pengolahan data, visualisasi, dan implementasi model deep learning dalam melakukan klasifikasi gambar a, b, dan c.

Draw.io juga digunakan untuk memvisualisasikan alur pengerjaan proyek secara menyeluruh. Diagram yang dibuat mencakup langkah-langkah mulai dari penerimaan dataset, preprocessing data, proses pelatihan dan evaluasi model, hingga analisis hasil akurasi yang diperoleh. Visualisasi alur ini membantu tim memahami struktur penanganan model secara sistematis dan mempermudah penyampaian proses dan hasil proyek kepada panitia dan orang lain. Berikut tahapan – tahapan dalam pengerjaan project Pro-Step : Road To Champion.

3.3.2.2. 1 Tahapan Import Data

Tahapan *import library* dilakukan untuk memanggil dan menyiapkan berbagai modul eksternal yang dibutuhkan dalam proses pengembangan proyek, sehingga fungsi-fungsi penting dapat digunakan tanpa harus membuatnya dari awal. Setiap library memiliki kegunaan spesifik, seperti

pengolahan data numerik, pembuatan model deep learning, evaluasi kinerja, visualisasi grafik, hingga pemrosesan gambar. Dengan melakukan import di awal, seluruh komponen analisis—mulai dari persiapan dataset, pelatihan model, optimasi, hingga visualisasi hasil—dapat berjalan secara efisien dan terstruktur, sehingga proses pengembangan menjadi lebih cepat, rapi, dan mudah dikelola.



Gambar 3. 4 Import Library

Untuk mendukung keseluruhan proses, mulai dari persiapan data hingga evaluasi model, berbagai modul dan library diimport pada tahap awal pengembangan model deep learning dan analisis data. Pengolahan data numerik dan manipulasi dataset dilakukan dengan library seperti NumPy dan Pandas. TensorFlow dan Keras berfungsi sebagai rangka kerja utama untuk membangun arsitektur model; mereka menyediakan class sequential, layer-layer CNN (Conv2D, MaxPooling2D, Flatten, dan Dropout), serta alat untuk meningkatkan gambar dengan ImageDataGenerator. Untuk meningkatkan kinerja dan stabilitas pelatihan model, beberapa optimizer, callback, seperti EarlyStopping, ModelCheckpoint, dan ReduceLROnPlateau, diimpor.

3.3.2.2. 2 Tahapan Spill Data

Gambar 3. 5 Spill Data

Tahapan ini digunakan untuk memanggil data yang disimpan dalam

Google Drive sehingga memakai code “from google.colab import drive”. Sedangkan untuk “drive.mount(‘/content/drive’) adalah import untuk mencari letaknya konten data yang akan dipanggil dari Google Colab. Setelah itu, variabel TRAIN_DIR, TEST_DIR, dan SAMPLE_SUB_PATH menyimpan path menuju folder dataset training, testing, dan file sampel submission. Kode os.listdir(TRAIN_DIR) kemudian digunakan untuk membaca dan menampilkan daftar nama folder yang ada di direktori training. Nama folder inilah yang kemudian dianggap sebagai kelas (label) dalam proses klasifikasi.

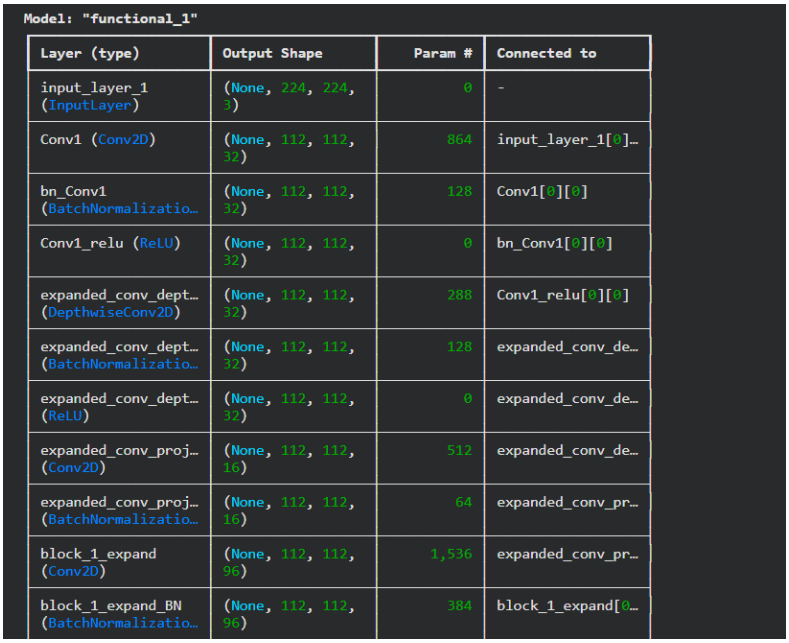
Dalam bagian berikutnya, sebuah dataframe dibangun untuk menyimpan daftar semua file gambar dan labelnya. Kode melakukan iterasi ke setiap folder kelas, lalu ekstensi file setiap gambar dicek agar hanya mengambil format file.jpg atau.png. Jalan penuh setiap file gambar dicatat ke dalam list jalan file, dan nama folder dicatat sebagai label ke dalam list label. Selanjutnya, dua daftar tersebut digabungkan menjadi sebuah dataframe df. Dataframe ini memiliki dua kolom: label untuk kelas setiap gambar dan filepath untuk lokasi file gambar. Nanti, dataframe ini akan digunakan untuk berbagai jenis pemrosesan gambar, seperti pembagian data, peningkatan, atau instruksi model.

3.3.2.2. 3 Tahapan Utility Data

Gambar 3. 6 Utility Data

Pada tahapan utility dan gambar berfungsi untuk memvisualisasikan sampel data pada setiap kategori gambar dalam direktori instruksi. Untuk memulai, program menggunakan perintah os.listdir(TRAIN_DIR) untuk mendapatkan daftar kategori dan menentukan berapa banyak gambar yang akan ditampilkan pada masing-masing kategori. Selanjutnya, program menggunakan plt.subplots() untuk membentuk grid tampilan berdasarkan jumlah kategori pada sumbu vertikal dan jumlah contoh gambar pada sumbu horizontal. Setelah itu, program melakukan iterasi pada setiap kategori, mengambil lima gambar pertama dari fold. Untuk membuat gambar lebih mudah ditemukan dalam dataset, setiap gambar diberi judul berdasarkan nama kategorinya.

3.3.2.2. 4 Set Parameter MobileVNet



Model: "functional_1"

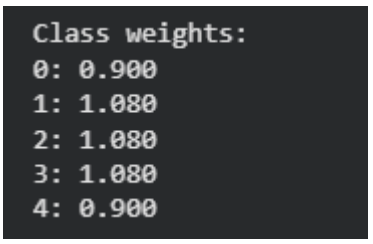
Layer (type)	Output Shape	Param #	Connected to
input_layer_1 (InputLayer)	(None, 224, 224, 3)	0	-
Conv1 (Conv2D)	(None, 112, 112, 32)	864	input_layer_1[0]…
bn_Conv1 (BatchNormalizatio…	(None, 112, 112, 32)	128	Conv1[0][0]
Conv1_relu (ReLU)	(None, 112, 112, 32)	0	bn_Conv1[0][0]
expanded_conv_dept… (DepthwiseConv2D)	(None, 112, 112, 32)	288	Conv1_relu[0][0]
expanded_conv_dept… (BatchNormalizatio…	(None, 112, 112, 32)	128	expanded_conv_de…
expanded_conv_dept… (ReLU)	(None, 112, 112, 32)	0	expanded_conv_de…
expanded_conv_proj… (Conv2D)	(None, 112, 112, 16)	512	expanded_conv_de…
expanded_conv_proj… (BatchNormalizatio…	(None, 112, 112, 16)	64	expanded_conv_pr…
block_1_expand (Conv2D)	(None, 112, 112, 96)	1,536	expanded_conv_pr…
block_1_expand_BN (BatchNormalizatio…	(None, 112, 112, 96)	384	block_1_expand[0…

Gambar 3. 7 Set Parameter

Set parameter digunakan untuk menunjukan jumlah bobot yang ada dalam dataset pada tiap layer dan kegunaan set parameter ini untuk mengatur seberapa besar kapasitas model dalam melakukan ekstraksi fitur dari gambar. Layer awal seperti Conv2D + BatchNormalization + ReLU memiliki parameter yang lebih kecil karena fokusnya pada fitur dasar seperti tepi dan tekstur sederhana. Sebaliknya, DepthwiseConv2D

dirancang dengan sangat hemat parameter untuk mengekstraksi fitur spasial tanpa membebani model, yang membuatnya lebih cepat dan efisien. Karena model mulai memperbesar dan memadatkan representasi fitur (dari channel kecil ke channel besar dan diproyeksikan kembali), jumlah parameter meningkat seiring dengan peningkatan layer dan proyeksi. Ini memastikan bahwa informasi penting tetap terjaga.

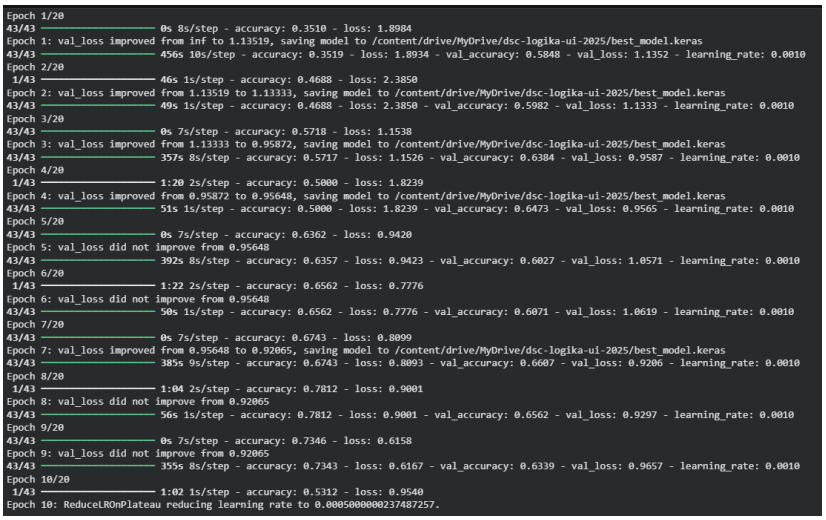
3.3.2.2. 5 Class Weight (MobileVNet)



Gambar 3. 8 Class Weight

Hasil class weight ini menunjukkan bahwa distribusi data antar kelas relatif cukup seimbang, karena nilai bobotnya tidak jauh berbeda dari 1. Kelas 1, 2, dan 3 memiliki bobot 1.080, artinya jumlah datanya sedikit lebih rendah dibanding kelas lain sehingga model diberi penekanan ekstra agar tidak mengabaikan kelas-kelas tersebut saat belajar. Sementara itu, kelas 0 dan 4 memiliki bobot 0.900, yang menandakan data pada kelas ini sedikit lebih banyak sehingga kontribusinya dikurangi agar tidak mendominasi proses training. Secara keseluruhan, pengaturan class weight ini membantu model belajar lebih adil, tetap memperhatikan semua kelas, dan mengurangi bias ke kelas yang datanya lebih banyak.

3.3.2.2. 6 Test & Training Data MobileVNet)

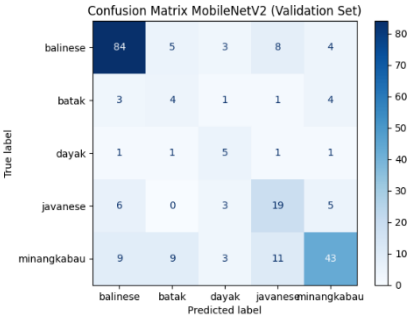


Gambar 3. 9 Training MobileVNet2

Hasil pelatihan model MobileNetV2 menunjukkan bahwa model belajar cukup cepat pada periode awal, ditandai dengan penurunan kesalahan validasi yang konsisten dari sekitar 1.13 menjadi ±0.92 dan peningkatan kesalahan validasi hingga 64–66%, menunjukkan bahwa fitur dasar gambar dipelajari dengan baik. Namun, setelah beberapa periode, kesalahan validasi mulai stagnan dan berubah-ubah, sementara akurasi pelatihan terus meningkat hingga di atas 70%, menunjukkan bahwa Mekanisme ModelCheckpoint berhasil menyimpan model terbaik saat val_loss meningkat, dan ReduceLROnPlateau menurunkan rate learning di epoch 10 karena tidak ada peningkatan yang signifikan. Ini adalah hasil dari upaya untuk menstabilkan proses belajar. Secara keseluruhan, MobileNetV2 sudah cukup baik untuk klasifikasi gambar dan stabil, tetapi masih bisa ditingkatkan dengan pengaturan kecepatan belajar, peningkatan data, atau penyempurnaan lapisan yang lebih dalam.

3.3.2.2. 7 Check Accuracy Model MobileVNet

```
8/8 ━━━━━━━━━━━━━━━━━━━━ 51s 6s/step
Validation F1-score (macro): 0.5301
Classification report MobileNetV2 (validation):
              precision    recall  f1-score   support

    balinese       0.82      0.81      0.81       104
       batak       0.21      0.31      0.25        13
       dayak       0.33      0.56      0.42         9
    javanese       0.47      0.58      0.52        33
 minangkabau       0.75      0.57      0.65        75

    accuracy                           0.66       234
   macro avg       0.52      0.56      0.53       234
weighted avg       0.70      0.66      0.67       234
```



Gambar 3. 10 Akurasi & Confusion Matrix

Hasil klasifikasi MobileNetV2 pada data validasi menunjukkan bahwa performa model tidak merata antar kelas. Dengan akurasi total 66% dan skor macro F1 sebesar 0.53, ini menunjukkan bahwa model masih mengalami kesulitan untuk mengenali secara seimbang semua kelas. Kelas Balinese dan Minangkabau memiliki performa terbaik (F1-score 0.81 dan 0.65) karena jumlah data yang lebih besar dan ciri visual yang lebih konsisten. Kelas Batak dan Dayak memiliki performa rendah (F1-score 0.25 dan 0.42) karena jumlah data yang lebih sedikit dan kemiripan visual yang lebih buruk dengan kelas lain. Kelas minoritas masih memiliki representasi yang lebih rendah, menunjukkan bahwa model cenderung "unggul" pada kelas mayoritas, dengan skor F1 berat rata-rata 0.67 yang lebih tinggi daripada rata-rata makro.

3.3.2.3 Tahap 3. Tahap Komparasi Model

Pada tahap komparasi dilakukan komparasi terhadap 3 model Transfer Learning dan 2 base model deep learning. Untuk Transfer Learning terdiri dari model *Densenet121, ResNet50, InceptionV3* sedangkan untuk base model adalah base model Deep Learning yang akan dikomparasikan adalah CNN dan ANN. Komparasi ini diperlukan untuk melihat dari segi akurasi klasifikasi, presisi, recall dan f1 score per kelas dari klasifikasi gambar serta melihat performa dari model tersebut

dan hasil prediksi gambar klasifikasi yang akan diciptakan oleh model tersebut. Berikut tahapan komparasi model Transfer Learning dan Base Model.

3.3.2.3.1. Model Densenet121

```
import tensorflow as tf
from tensorflow.keras.applications import DenseNet121
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping
import os
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import warnings
warnings.filterwarnings('ignore')
from sklearn.model_selection import train_test_split
from google.colab import drive
drive.mount('/content/drive')
```

Gambar 3. 11 Import Library Densenet121

Proses import berbagai lembaga digunakan untuk membangun, melatih, dan mengevaluasi model deep learning. Ini dilakukan pada bagian awal kode. Dengan bantuan modul keras-nya, library TensorFlow digunakan untuk menghubungkan arsitektur terlatih seperti DenseNet121, membuat model menggunakan Dense dan GlobalAveragePooling2D, dan menggunakan ImageDataGenerator untuk melakukan augmentasi gambar. Selain itu, modul optimasi seperti Adam dan callback seperti ModelCheckpoint dan EarlyStopping diimpor untuk meningkatkan kinerja pelatihan model. Sementara library lain seperti os, numpy, dan pandas digunakan untuk pengelolaan data dan manipulasi array, matplotlib dan seaborn digunakan untuk menampilkan metrik evaluasi dan grafik hasil pelatihan.

Pada bagian berikut, peringatan sistem diminimalkan dengan menggunakan warnings.filterwarnings('ignore') untuk membuat output lebih bersih. Diimpor ke scikit-learn, fungsi train_test_split membagi dataset menjadi data uji dan data latih secara terstruktur. Selain itu, modul drive Google Colab dapat digunakan untuk melakukan mounting Google

Drive dengan perintah drive.mount('/content/drive'). Ini memungkinkan pengguna untuk mengakses dataset atau menyimpan model langsung ke penyimpanan cloud. Tujuan dari proses import ini adalah untuk menyiapkan seluruh komponen yang diperlukan untuk pipeline pengolahan data dan mengembangkan model klasifikasi berbasis pembelajaran mendalam.

```
TRAIN_DIR = '/content/drive/MyDrive/klasifikasi/dsc-logika-ui-2025/Train/Train'
TEST_DIR = '/content/drive/MyDrive/klasifikasi/dsc-logika-ui-2025/Test/Test'
SAMPLE_SUB_PATH = os.path.join('/content/drive/MyDrive/klasifikasi/dsc-logika-ui-2025/densenet121.csv')

# List classes
classes = sorted(os.listdir(TRAIN_DIR))
print("Classes:", classes)

# Build dataframe
filepaths, labels = [], []
for cls in classes:
    for f in os.listdir(os.path.join(TRAIN_DIR, cls)):
        if f.endswith((".jpg", ".png")):
            filepaths.append(os.path.join(TRAIN_DIR, cls, f))
            labels.append(cls)

df = pd.DataFrame({"filepath": filepaths, "label": labels})

# Split train-val
train_df, val_df = train_test_split(df, stratify=df['label'], test_size=0.2, random_state=42)
print(len(train_df), len(val_df))
```

Gambar 3. 12 Data Frame dan Koleksi Data

Kode di atas dimulai dengan mendefinisikan direktori dataset TRAIN_DIR dan TEST_DIR untuk data latihan dan uji. Dengan menggunakan os.listdir(), program membaca semua nama kelas dalam folder data latihan dan menyimpannya dalam variabel classes yang diurutkan menurut alfabet. Proses selanjutnya adalah membuat dataframe yang berisi daftar path dan label file. Program melakukan iterasi pada seluruh file gambar dengan ekstensi.jpg atau.png untuk setiap kelas, kemudian menyimpan path dan label kelas lengkap file ke dalam dua buah list. Kedua list ini kemudian digabungkan menjadi sebuah dataframe menggunakan pandas, sehingga dataset menjadi terstruktur dan siap untuk diolah pada tahap berikutnya.

Setelah dataframe dibangun dengan sukses, dataset dibagi menjadi data latihan dan data validasi dengan menggunakan fungsi train_test_split() dari scikit-learn. Distribusi kelas pada data latih dan validasi tetap seimbang sesuai proporsi awal dengan menggunakan parameter stratify=df['label'], dan test_size=0.2 menunjukkan bahwa

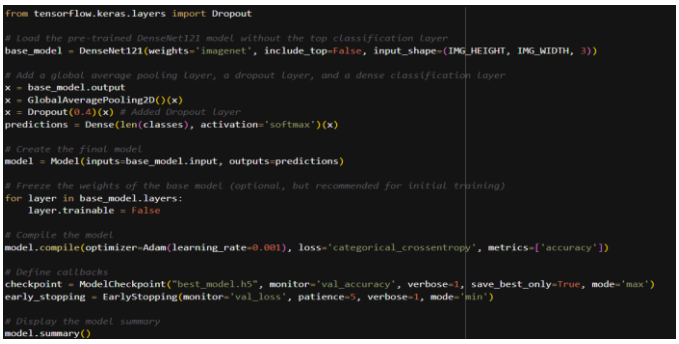
20% dari dataset akan digunakan sebagai data validasi. Random_state=42 menjamin bahwa pembagian data dapat diulang dan menghasilkan pembagian yang konsisten. Karena data latih digunakan untuk membangun model sedangkan data validasi digunakan untuk mengevaluasi kinerja model secara objektif sebelum diuji lebih lanjut, tahap ini sangat penting untuk menjaga kualitas pelatihan model.

```
IMG_HEIGHT = 224
IMG_WIDTH = 224
BATCH_SIZE = 32

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

# No augmentation for validation data, only rescaling
val_datagen = ImageDataGenerator(
    rescale=1./255
)

# Create data generators
train_generator = train_datagen.flow_from_dataframe(
    dataframe=train_df,
    x_col='filepath',
    y_col='label',
    target_size=(IMG_HEIGHT, IMG_WIDTH),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    shuffle=True
)

val_generator = val_datagen.flow_from_dataframe(
    dataframe=val_df,
    x_col='filepath',
    y_col='label',
    target_size=(IMG_HEIGHT, IMG_WIDTH),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    shuffle=False
)
```

Gambar 3. 13 Augmentasi Gambar

Proses pengembangan data digunakan dalam kode di atas untuk meningkatkan keragaman dataset dan mencegah overfitting selama pelatihan model. Dengan menggunakan ImageDataGenerator, peningkatan dilakukan dengan menambahkan beberapa perubahan, seperti normalisasi pixel yang diubah, rotasi hingga 20 derajat, zoom sebesar 20%, pergeseran sebesar 20% secara horizontal dan vertikal, dan pembalikan gambar secara horizontal. Teknik-teknik ini bertujuan untuk menghasilkan variasi baru dari gambar asli tanpa mengubah labelnya. Ini memungkinkan model untuk mengenali pola dengan lebih baik meskipun sudut, posisi, atau ukuran objek berubah. Dengan menggunakan fill_mode='nearest', area kosong yang muncul sebagai hasil dari transformasi akan diisi dengan nilai piksel terdekat.

Data validasi harus mencerminkan kondisi data asli tanpa diubah agar evaluasi performa model lebih objektif, tidak ada peningkatan apa pun yang dilakukan untuk data validasi selain rescaling. Setelah menentukan proses augmentasi, flow_from_dataframe digunakan untuk membuat generator yang dapat membaca jalan gambar dan label dari dataframe secara langsung. Generator ini mengubah ukuran gambar menjadi 224 x 224 piksel, menetapkan ukuran batch 32 piksel, dan mengubah label ke format kategorikal. Pada data pelatihan, parameter shuffle=True digunakan untuk memastikan bahwa data diacak pada setiap epoch. Di sisi lain, pada data validasi, parameter shuffle=False digunakan untuk memastikan bahwa urutan data tetap sama saat evaluasi dilakukan. Pipeline data menjadi lebih terstruktur dan siap digunakan dalam proses pelatihan model deep learning dengan cara ini.



```
from tensorflow.keras.layers import Dropout

# Load the pre-trained DenseNet121 model without the top classification layer
base_model = DenseNet121(weights='imagenet', include_top=False, input_shape=(IMG_HEIGHT, IMG_WIDTH, 3))

# Add a global average pooling layer, a dropout layer, and a dense classification layer
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dropout(0.4)(x) # Added Dropout layer
predictions = Dense(len(classes), activation='softmax')(x)

# Create the final model
model = Model(inputs=base_model.input, outputs=predictions)

# Freeze the weights of the base model (optional, but recommended for initial training)
for layer in base_model.layers:
    layer.trainable = False

# Compile the model
model.compile(optimizer=Adam(learning_rate=0.001), loss='categorical_crossentropy', metrics=['accuracy'])

# Define callbacks
checkpoint = ModelCheckpoint("best_model.h5", monitor='val_accuracy', verbose=1, save_best_only=True, mode='max')
early_stopping = EarlyStopping(monitor='val_loss', patience=5, verbose=1, mode='min')

# Display the model summary
model.summary()
```

Gambar 3. 14 Load Model

Pada kode di atas, proses penambahan dan pembuatan model dimulai dengan menggunakan arsitektur DenseNet121 sebagai dasar model dengan bobot yang dilatih dari ImageNet. Lapisan klasifikasi bawaan DenseNet121 dihapus dengan menggunakan parameter include_top=False. Ini memungkinkan model untuk disesuaikan dengan jumlah kelas yang ada dalam dataset. Setelah itu, output dari basis model diproses melalui beberapa lapisan tambahan. Ini termasuk lapisan GlobalAveragePooling2D untuk mengurangi ukuran fitur menjadi vektor satu dimensi dan lapisan Dropout dengan nilai 0.4 untuk mencegah overfitting. Untuk menghasilkan probabilitas klasifikasi, ditambahkan lapisan Dense dengan jumlah neuron yang sesuai dengan kelas dan fungsi

aktivasi softmax. Untuk membentuk model akhir yang siap dilatih, semua komponen ini digabungkan ke dalam objek Model.

Setelah model berhasil dibuat, bobot dari base model dibekukan dengan mengatur layer.trainable = False untuk mencegah perubahan selama fase pelatihan awal, sehingga proses pelatihan fokus pada lapisan baru. Model kemudian dikompilasi menggunakan optimizer Adam dengan learning rate 0.001, fungsi kerugian categorical crossentropy, dan metrik evaluasi berupa accuracy. Dua callback juga didefinisikan, yaitu ModelCheckpoint untuk menyimpan versi model terbaik berdasarkan akurasi validasi, dan EarlyStopping untuk menghentikan proses pelatihan ketika kehilangan validasi tidak menunjukkan perbaikan, sehingga mencegah pembelajaran berlebihan. Terakhir, model.summary() digunakan untuk menampilkan struktur model secara keseluruhan, termasuk jumlah parameter yang dilatih dan tidak dilatih.

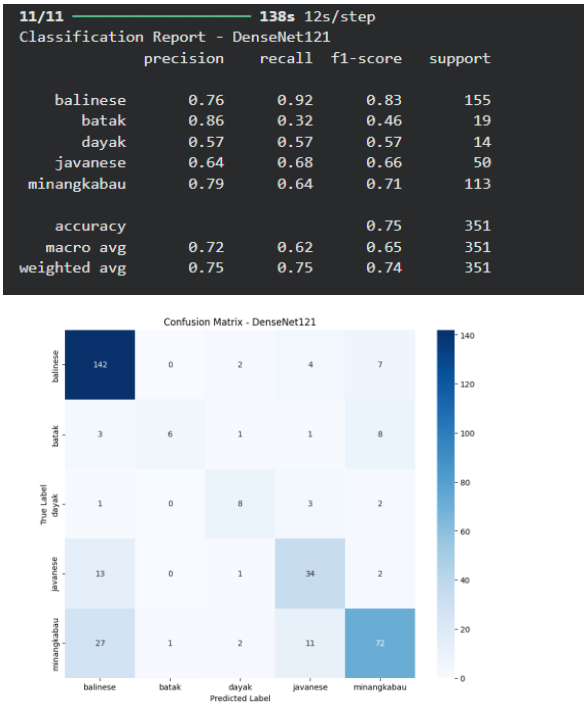
```
history_densenet = model.fit(
    train_generator,
    epochs=15, # You can adjust the number of epochs
    validation_data=val_generator,
    callbacks=[checkpoint, early_stopping],
    verbose=1
)
```

Gambar 3. 15 Load Model

Kode di atas menunjukkan proses pelatihan (training) model DenseNet121 menggunakan metode fit() pada Keras. Pada bagian ini, model dilatih menggunakan train_generator sebagai sumber data pelatihan dengan 15 epoch, yang berarti seluruh dataset akan diproses sebanyak 15 kali untuk memperbarui bobot model. Validation_data, yang merupakan val_generator dalam proses pelatihan, digunakan untuk mengevaluasi kinerja model pada data validasi setiap akhir epoch. Informasi tentang detail proses pelatihan untuk setiap waktu, seperti nilai kehilangan, ketepatan, val_kehilangan, dan val_ketepatan, ditampilkan dengan menggunakan parameter verbose=1.

Selain itu, ada dua callback dalam proses pelatihan. Mereka adalah ModelCheckpoint dan EarlyStopping, yang berfungsi melalui variabel

checkpoint dan early_stopping. ModelCheckpoint berfungsi untuk menyimpan bobot model terbaik selama pelatihan berdasarkan metrik tertentu, sehingga dapat mencegah kehilangan model optimal karena variasi dalam performa di epoch selanjutnya. Sementara itu, EarlyStopping secara otomatis menghentikan proses pelatihan jika tidak ada peningkatan dalam data.



Gambar 3. 16 Hasil Akurasi dan Confusion Matrix

Sebagai hasil dari laporan klasifikasi, model DenseNet121 dapat mencapai akurasi total sebesar 75% pada data uji. Balinese memiliki performa terbaik dengan precision 0.76, recall 0.92, dan f1-score 0.83. Sebaliknya, kelas seperti batak dan dayak memiliki recall yang lebih rendah, masing-masing 0.32 dan 0.57, menunjukkan bahwa model masih kesulitan mengenali sebagian besar sampel kelas tersebut. Peringkat rata-rata f1 macro sebesar 0,65 menunjukkan ketidakseimbangan dalam performa antar kelas, sedangkan peringkat rata-rata f1 berat sebesar 0.74 menunjukkan bahwa performa keseluruhan tetap cukup baik ketika mempertimbangkan proporsi jumlah data untuk setiap kelas.

Selain itu, distribusi prediksi model terhadap label sebenarnya

ditunjukkan oleh confusion matrix. Dengan 142 prediksi yang benar, kelas balinese kembali terlihat paling dominan, menunjukkan tingkat konsistensi model yang tinggi. Namun, beberapa kelas mengalami misklasifikasi yang signifikan, seperti javanese, yang sering dianggap sebagai balinese (13 sampel), dan minangkabau, yang juga sering dianggap sebagai balinese dan javanese. Adanya kemiripan fitur visual antara beberapa kelas atau perlunya peningkatan kualitas dan kuantitas data pelatihan ditunjukkan oleh pola misklasifikasi ini. Secara keseluruhan, visualisasi ini membantu memahami kelemahan model dan merupakan dasar untuk perbaikan.

3.3.2.3.2. Model RestNet50

```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import ResNet50V2
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.metrics import classification_report, confusion_matrix
import pandas as pd
import os
```

Gambar 3. 17 Library Restnet50

Dengan menggunakan kode di atas, Anda dapat mengimpor berbagai library yang diperlukan untuk proses pembangunan dan evaluasi model deep learning berbasis Keras dan TensorFlow. Proses ini mencakup pembuatan arsitektur model, pemrosesan gambar, dan pengaturan optimizer. Pada bagian ini, ResNet50V2 diimpor sebagai model dasar (pretrained model), dan lapisan seperti Dense, GlobalAveragePooling2D, dan Dropout digunakan untuk membentuk struktur jaringan yang lebih khusus untuk kebutuhan proyek. Selain itu, ImageDataGenerator telah disiapkan untuk melakukan augmentasi data, yang merupakan komponen yang sangat penting untuk meningkatkan kinerja model dengan meningkatkan variasi gambar pelatihan.

Selain itu, selama proses evaluasi model, library pendukung seperti matplotlib, seaborn, dan pandas digunakan untuk menampilkan dan

menganalisis data. Sementara sklearn digunakan untuk pengolahan data numerik, numpy digunakan untuk itu.Untuk menganalisis performa prediksi model, metrics menyediakan fungsi classification_report dan confusion_matrix. Terakhir, library sistem berfungsi untuk mengatur direktori dan rute file dalam sistem, yang membuat akses ke dataset dan penyimpanan hasil lebih mudah. Untuk menjalankan alur kerja pembelajaran mesin dari tahap preprocessing, pelatihan, hingga evaluasi model, semua import ini saling melengkapi.

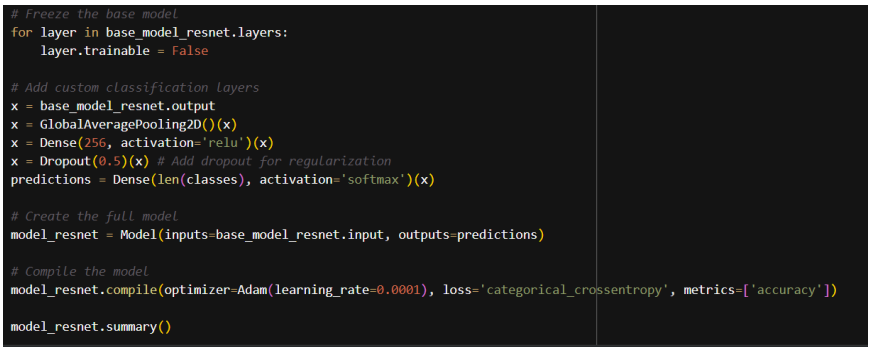
```
IMG_HEIGHT = 224
IMG_WIDTH = 224
BATCH_SIZE = 32

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

# No augmentation for validation data, only rescaling
val_datagen = ImageDataGenerator(
    rescale=1./255
)

# Create data generators
train_generator = train_datagen.flow_from_dataframe(
    dataframe=train_df,
    x_col='filepath',
    y_col='label',
    target_size=(IMG_HEIGHT, IMG_WIDTH),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    shuffle=True
)

val_generator = val_datagen.flow_from_dataframe(
    dataframe=val_df,
    x_col='filepath',
    y_col='label',
    target_size=(IMG_HEIGHT, IMG_WIDTH),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    shuffle=False
)
```

Gambar 3. 18 Augmentasi Data

Dengan menggunakan ImageDataGenerator, variasi gambar pada data pelatihan diperbesar tanpa menambah jumlah data secara manual. Memutar, memperbesar, dan menggeser gambar secara horizontal dan vertikal dapat dilakukan dengan teknik augmentasi seperti rotation_range, zoom_range, width_shift_range, dan height_shift_range. Selain itu, gambar dibalik secara horizontal (horizontal_flip=True) untuk memberi model kesempatan untuk mempelajari berbagai sudut pandang objek. Karena terbiasa dengan variasi kelas yang sama, transformasi ini membuat model lebih kuat dan mencegah overfitting. Setelah itu, gambar

secara keseluruhan dinormalisasi dengan mengubah skala=1./255 sehingga nilai piksel berada di rentang 0–1, sehingga model dapat memprosesnya dengan lebih stabil.

Agar evaluasi model tetap objektif, data validasi tidak memerlukan augmentasi. Kode hanya melakukan rescaling pada data validasi, yang berarti gambar diuji dalam kondisi asli tanpa mengubahnya. Kemudian, flow_from_dataframe membaca jalan gambar dan label langsung dari DataFrame. Kemudian, itu diubah menjadi batch berukuran 32, dan ukuran gambar diubah menjadi 224 x 224 piksel. Untuk data pelatihan, shuffle=True digunakan untuk menjaga urutan data tetap acak di setiap epoch. Di sisi lain, untuk data validasi, shuffle=False digunakan untuk menjaga konsistensi evaluasi. Skema ini membuat proses pelatihan menjadi lebih efisien dan hasil validasi menjadi lebih akurat dalam menggambarkan bagaimana model bekerja secara sebenarnya.



```
# Freeze the base model
for layer in base_model_resnet.layers:
    layer.trainable = False

# Add custom classification layers
x = base_model_resnet.output
x = GlobalAveragePooling2D()(x)
x = Dense(256, activation='relu')(x)
x = Dropout(0.5)(x) # Add dropout for regularization
predictions = Dense(len(classes), activation='softmax')(x)

# Create the full model
model_resnet = Model(inputs=base_model_resnet.input, outputs=predictions)

# Compile the model
model_resnet.compile(optimizer=Adam(learning_rate=0.0001), loss='categorical_crossentropy', metrics=['accuracy'])

model_resnet.summary()
```

Gambar 3. 19 Load Model

Kode ini menunjukkan bagaimana membuat model deep learning berbasis arsitektur ResNet, yang berfungsi sebagai feature extractor. Proses ini membekukan semua lapisan pada model dasar agar tidak dilatih kembali. Selanjutnya, layer klasifikasi yang disesuaikan ditambahkan, seperti GlobalAveragePooling2D, Dense dengan aktivasi relu, dan Dropout untuk mengurangi overfitting. Layer Dense beraktivasi softmax digunakan sebagai output sesuai jumlah kelas. Selanjutnya, seluruh bagian ini dirangkai menjadi satu model baru bernama model_resnet. Ini dikompilasi menggunakan optimizer Adam dengan rate

learning kecil (0.0001), kehilangan bertipe crossentropy categorical, dan metrik evaluasi ketepatan. Terakhir, model_resnet.summary digunakan untuk menampilkan ringkasan arsitekturnya.

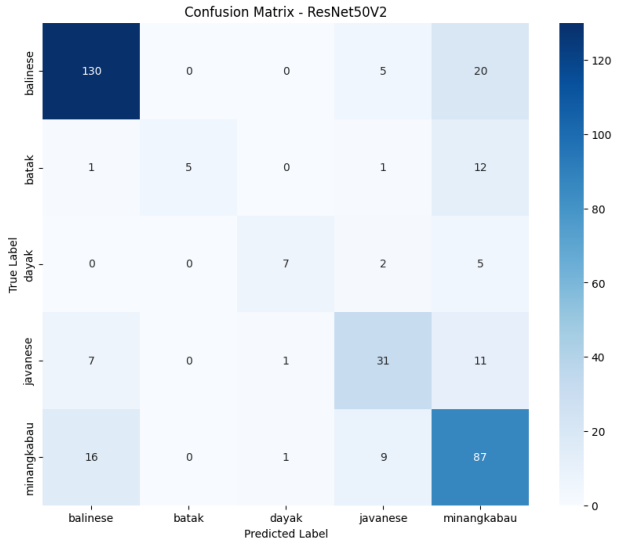
```
history_resnet = model_resnet.fit(
    train_generator,
    epochs=15,
    validation_data=val_generator,
    verbose=1
)
```

Gambar 3. 20 Training Model

Selama 15 epoch, model ResNet50 dilatih menggunakan data latih (train_generator) dan divalidasi dengan data validasi (val_generator). Pada setiap epoch, model secara bertahap belajar pola-pola penting dari data citra dengan menggunakan proses backpropagation untuk menyesuaikan bobot. Dengan menggunakan arsitektur ResNet50 dengan koneksi residual, masalah vanishing gradient dikurangi. Ini memungkinkan model untuk belajar dengan lebih stabil dan efisien meskipun memiliki kedalaman jaringan yang cukup besar.

Hasil pelatihan ditunjukkan melalui metrik seperti kehilangan dan akurasi pada data pelatihan dan validasi yang dicatat dalam objek history_resnet. Nilai kehilangan dan validasi sama-sama menurun sementara akurasi meningkat, menunjukkan bahwa model mampu mempelajari fitur dengan baik dan memiliki kemampuan generalisasi yang cukup baik terhadap data baru. Di sisi lain, perbedaan kecil antara performa pelatihan dan validasi menunjukkan bahwa mode

```
11/11 ━━━━━━━━━━━━━━━━━━━━ 135s 12s/step
Classification Report
              precision    recall  f1-score   support

    balinese       0.84      0.84      0.84       155
       batak       1.00      0.26      0.42        19
       dayak       0.78      0.50      0.61        14
    javanese       0.65      0.62      0.63        50
 minangkabau       0.64      0.77      0.70       113

    accuracy                           0.74       351
   macro avg       0.78      0.60      0.64       351
weighted avg       0.76      0.74      0.73       351
```

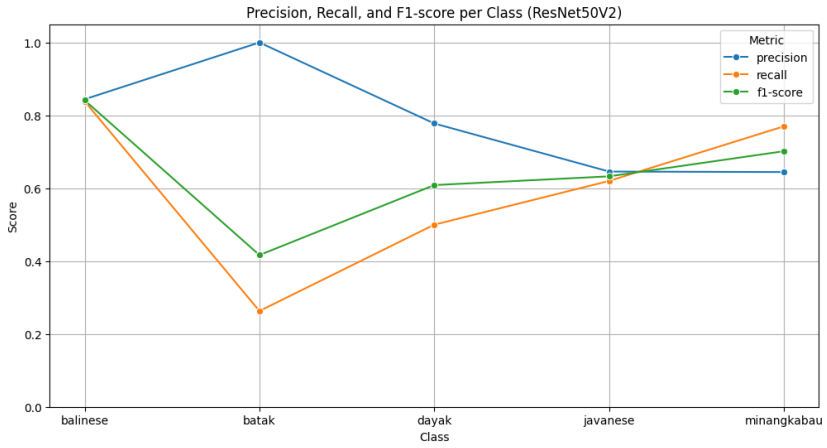


Gambar 3. 21 Hasil Akurasi & Confusion Matrix

Berdasarkan confusion matrix, model ResNet50V2 menunjukkan performa yang cukup baik pada kelas Balinese dan Minangkabau. Ini ditunjukkan oleh jumlah prediksi benar yang tinggi pada diagonal matriks: 130 prediksi benar untuk kelas Balinese dan 87 prediksi benar untuk kelas Minangkabau, menunjukkan bahwa model mampu mengenali ciri visual kedua kelas ini dengan cukup konsisten. Namun, masih ada kesalahan dalam klasifikasi kelas yang memiliki kemiripan, terutama antara Balinese–Minangkabau dan Javanese–Minangkabau. Hal ini menunjukkan adanya tumpang tindih fitur visual antara kedua budaya tersebut.

Dalam laporan klasifikasi, model memiliki akurasi keseluruhan sebesar 74%, dengan nilai nilai F1 yang diimbangi sebesar 0.73, yang menunjukkan bahwa model memiliki kinerja yang cukup stabil pada data

yang tidak seimbang. Kelas Batak memiliki tingkat keakuratan yang sangat tinggi (1.00), tetapi recall yang rendah (0.26), yang menunjukkan bahwa model sangat akurat dalam memprediksi Batak tetapi gagal mengidentifikasi banyak data Batak yang sebenarnya. Meskipun demikian, kelas Dayak dan Javanese memiliki kinerja menengah dengan keseimbangan tepat dan ingat yang cukup baik. Secara keseluruhan, hasil ini menunjukkan bahwa model sudah mampu melakukan klasifikasi dengan cukup baik; namun, untuk membuatnya lebih efisien, terutama untuk kelas dengan jumlah data yang lebih sedikit, masih diperlukan peningkatan.



Gambar 3. 22 Visualisasi Precision, Recall, F1 Score

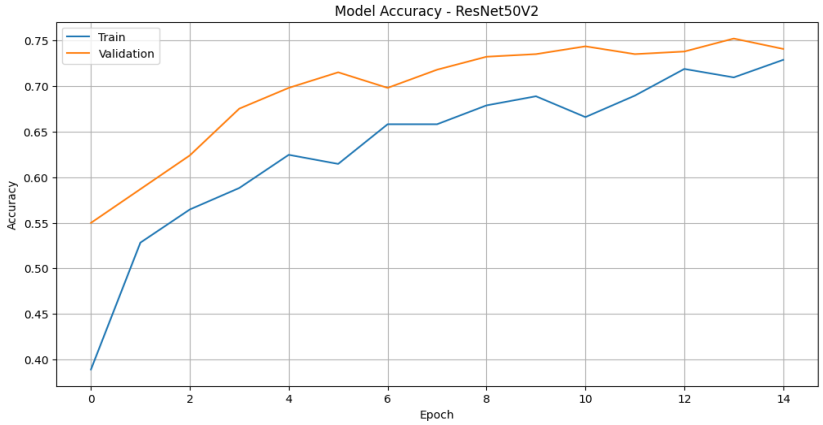
Sebuah grafik yang menunjukkan bahwa model ResNet50V2 memiliki performa yang berbeda pada setiap kelas menunjukkan ketepatan, recall, dan skor F1 untuk setiap kelas. Baik Dayak maupun Javanese memiliki nilai precision dan recall yang relatif rendah. Kelas Balinese memiliki keseimbangan yang baik antara precision dan recall (≈0.84), yang menghasilkan skor F1 yang tinggi, yang menunjukkan prediksi yang konsisten dan stabil. Kelas Batak memiliki precision yang sangat tinggi (1.00) tetapi recall yang rendah (0.26), yang menunjukkan bahwa model Batak sangat selektif dan hanya sedikit data Batak yang berhasil dikenali. Secara umum, grafik ini menunjukkan ketidakseimbangan data dan kemiripan fitur antar kelas model.

```
print(f"Overall Accuracy: {overall_accuracy:.2f}")
print(f"Overall Macro F1-score: {report['macro avg']['f1-score']:.2f}")
```

```
Overall Accuracy: 0.74
Overall Macro F1-score: 0.64
```

Gambar 3. 23 Akurasi Resnet50

Hasil evaluasi menunjukkan akurasi keseluruhan sebesar 0.74, yang menunjukkan bahwa model ResNet50V2 dapat mengklasifikasikan sekitar 74% dari data gambar secara keseluruhan dengan benar. Nilai-nilai ini menunjukkan bahwa, secara umum, model dapat mengidentifikasi pola visual pada dataset yang digunakan dengan cukup baik. Namun, akurasi tidak sepenuhnya mencerminkan kinerja model pada setiap kelas secara terpisah karena bersifat agregat dan cenderung dipengaruhi oleh kelas yang memiliki jumlah data yang lebih besar.



Gambar 3. 24 Peforma Model Resnet50

Akurasi model ResNet50V2 meningkat selama proses training dan validasi di setiap epoch. Akurasi meningkat dari sekitar 39% di awal epoch hingga hampir 73% di akhir epoch, menunjukkan bahwa model menjadi lebih mampu mendeteksi pola dari data latih. Tren umumnya tetap naik, meskipun kadang-kadang terjadi perubahan kecil di beberapa periode. Karena penyesuaian bobot tidak selalu linear, itu wajar dalam proses pembelajaran model deep learning.

Sebaliknya, akurasi validasi terus meningkat, bahkan sedikit lebih tinggi daripada akurasi training. Ini menunjukkan bahwa model tidak

hanya mampu mempelajari data latih dengan baik, tetapi juga mampu generalisasi dengan baik pada data baru. Ada kemungkinan kecil terjadi overfitting karena tidak ada perbedaan yang signifikan antara ketepatan pelatihan dan validasi. Secara keseluruhan, kinerja ini menunjukkan bahwa ResNet50V2 bekerja dengan baik untuk tugas klasifikasi gambar yang sedang dilakukan, meskipun masih ada ruang untuk meningkatkan akurasi dengan mengatur hyperparameter atau menambah data.

3.3.2.3.3. Model InceptionV3

```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import InceptionV3 # Perubahan di sini
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.metrics import classification_report, confusion_matrix
import pandas as pd
import os
```

Gambar 3. 25 Library InceptionV3

Kode ini menyiapkan seluruh "alat kerja" yang diperlukan untuk membangun dan mengevaluasi model deep learning berbasis gambar. ImageDataGenerator sangat penting untuk membaca dan melakukan preprocessing data gambar sebelum dimasukkan ke dalam model, sementara TensorFlow dan Keras digunakan sebagai dasar untuk membuat model. Arsitektur InceptionV3 menunjukkan bahwa pendekatan transfer learning dipilih, yang memanfaatkan model yang telah dilatih sebelumnya. Ini memungkinkan proses pelatihan berjalan lebih efisien dan menghasilkan hasil yang lebih optimal, terutama dalam kasus di mana jumlah data tidak terlalu banyak.

Selain itu, penggunaan alat dan lembaga ini membantu proses pengembangan dan evaluasi model secara terstruktur. Adanya Keras sebagai API tingkat tinggi membuat proses penyusunan arsitektur, pengaturan parameter pelatihan, dan evaluasi performa model seperti akurasi lebih mudah dan konsisten. Kombinasi alat ini menciptakan alur

kerja yang efisien dan terintegrasi sehingga model deep learning yang dibangun dapat belajar dengan lebih baik dan memiliki generalisasi yang lebih besar saat digunakan.

```
from google.colab import drive
from sklearn.model_selection import train_test_split # Added this line
TRAIN_DIR = '/content/drive/MyDrive/klasifikasi/dsc-logika-ui-2025/Train/Train'
TEST_DIR = '/content/drive/MyDrive/klasifikasi/dsc-logika-ui-2025/Test/Test'
SAMPLE_SUB_PATH = os.path.join('/content/drive/MyDrive/klasifikasi/dsc-logika-ui-2025/inceptionv3.csv')
classes = sorted(os.listdir(TRAIN_DIR))
print("Classes:", classes)

# Build dataframe
filepaths, labels = [], []
for cls in classes:
    for f in os.listdir(os.path.join(TRAIN_DIR, cls)):
        if f.endswith((".jpg", ".png")):
            filepaths.append(os.path.join(TRAIN_DIR, cls, f))
            labels.append(cls)

df = pd.DataFrame({"filepath": filepaths, "label": labels})

# Split train-val
train_df, val_df = train_test_split(df, stratify=df['label'], test_size=0.2, random_state=42)
print(len(train_df), len(val_df))
```

Gambar 3. 26 Data Frame InceptionV3

Sebelum masuk ke proses pelatihan model, kode ini dapat dianggap sebagai langkah awal saat "menata" data. Untuk memulai, Anda harus menghubungkan Google Drive agar Colab dapat mengakses dataset yang tersimpan di Drive. Selanjutnya, Anda harus menetapkan direktori data latihan dan tes, yang berisi folder-folder kelas. Karena nama folder diambil secara otomatis dari daftar kelas, model dapat mengikuti struktur data saat ini tanpa perlu memberikan input manual. Kode ini kemudian mengumpulkan seluruh jalan file gambar dan labelnya ke dalam sebuah DataFrame, seolah-olah kita sedang membuat daftar isi data untuk membuatnya lebih mudah diurus.

Setelah data disusun, langkah berikutnya adalah membaginya menjadi data latihan dan data validasi. Sehingga proporsi setiap kelas tetap seimbang di kedua bagian data, proses pembagian dilakukan dengan train_test_split menggunakan stratifikasi berdasarkan label. Cara berpikir dengan cermat agar model dapat belajar dengan cukup data dan diuji dengan data representatif ditunjukkan dengan pembagian dua puluh persen untuk pelatihan dan dua puluh persen untuk validasi ini. Secara keseluruhan, alur berpikir sistematis ditunjukkan oleh kode ini: memastikan data terorganisir dengan baik sebelum digunakan.

```python
from tensorflow.keras.applications import InceptionV3

# Load the pre-trained InceptionV3 model without the top classification layer
base_model_inception = InceptionV3(weights='imagenet', include_top=False, input_shape=(IMG_HEIGHT, IMG_WIDTH, 3))

# Freeze the base model layers
for layer in base_model_inception.layers:
    layer.trainable = False

# Add custom classification layers
x = base_model_inception.output
x = GlobalAveragePooling2D()(x)
x = Dense(256, activation='relu')(x)
x = Dropout(0.5)(x) # Add dropout for regularization
predictions = Dense(len(classes), activation='softmax')(x)

# Create the full model
model_inception = Model(inputs=base_model_inception.input, outputs=predictions)

# Compile the model
model_inception.compile(optimizer=Adam(learning_rate=0.0001), loss='categorical_crossentropy', metrics=['accuracy'])

model_inception.summary()
```

Gambar 3. 27 Build Model InceptionV3

Dalam model ini, InceptionV3 digunakan sebagai backbone utama dan pendekatan transfer learning digunakan. Model mengidentifikasi pola visual dasar seperti tepi, tekstur, dan bentuk dengan menggunakan informasi yang telah dipelajari dari InceptionV3 dari kumpulan data ImageNet yang besar. Karena model ingin disesuaikan dengan persyaratan klasifikasi baru, bagian atas lapisan (classifier bawaan) sengaja dihapus (include_top=False). Setelah itu, seluruh lapisan InceptionV3 dibekukan, atau dibekukan, sehingga bobotnya tidak dilatih ulang. Metode ini lebih stabil, lebih cepat, dan mengurangi risiko overfitting, terutama dalam kasus data latihan yang tidak terlalu besar.

Sebagai kepala klasifikasi baru, beberapa layer khusus ditambahkan ke output InceptionV3. Dengan menggunakan GlobalAveragePooling2D, fitur-fitur penting dari feature map diuraikan tanpa menambahkan terlalu banyak parameter. Layer Dense dengan 256 neuron dan aktivasi ReLU kemudian berfungsi untuk mempelajari kombinasi fitur yang lebih khusus untuk dataset. Agar model tidak terlalu "menghafal" data latih, dropout sebesar 0.5 ditambahkan sebagai regularisasi. Terakhir, probabilitas untuk setiap kelas dihasilkan oleh layer Dense dengan aktivasi softmax. Karena tugasnya adalah klasifikasi multikelas, model ini dikompilasi menggunakan optimizer Adam dengan rate belajar yang rendah, yang cocok untuk fine-tuning dan kehilangan crossentropy kategoris.

```
images, labels = next(val_generator)

plt.figure(figsize=(6, 6))
plt.imshow(images[0])
plt.title(f"Label: {classes[np.argmax(labels[0])]}")
plt.axis('off')
plt.show()

print(f"Shape of images batch: {images.shape}")
print(f"Shape of labels batch: {labels.shape}")
```

Label: balinese

Gambar 3. 28 Labeling Data Shape

Kode ini digunakan untuk mengambil dan mengecek satu batch data dari val_generator. Ini memungkinkan kami untuk memastikan bahwa gambar dan label telah dibaca dengan benar. Sekumpulan gambar dan label diambil oleh bar gambar, labels = next(val_generator), dan kemudian salah satu gambar pertama ditampilkan secara visual menggunakan plt.imshow(images[0]). Dengan menggunakan argmax, yang menunjukkan kelas gambar, indeks label tertinggi dapat ditampilkan pada judul gambar. Setelah itu, kode mencetak bentuk (shape) dari batch gambar dan label. Ini dilakukan untuk memastikan bahwa ukuran, seperti jumlah batch, ukuran gambar, dan jumlah kelas, sesuai dengan yang diharapkan oleh model.

Gambar tersebut menampilkan Kori Agung, gerbang utama yang megah dalam arsitektur pura Bali. Secara teknis, dalam konteks

pengajaran mesin, gambar tersebut berfungsi sebagai representasi visual dari kelas "balinese" melalui penggunaan warna putih yang dominan pada struktur batu yang padat, kain kuning-putih (wastra) yang menutupi badan gerbang, dan menara bertingkat yang menjulang ke atas. Kesuksesan sistem dalam menampilkan label "balinese" di atas gambar ini menunjukkan bahwa ground truth atau metadata dataset telah dimasukkan dengan benar ke file gambar yang tepat sebelum memasuki tahap pelatihan.

```
from sklearn.utils import class_weight

# Get the labels for the training set
train_labels = train_df['label']

# Calculate class weights
class_weights = class_weight.compute_class_weight(
    class_weight='balanced',
    classes=np.unique(train_labels),
    y=train_labels
)

# Convert to a dictionary for easier use with Keras
class_weights_dict = dict(enumerate(class_weights))

print("Class Weights:", class_weights_dict)
```

Gambar 3. 29 Class Weight InceptionV3

Untuk menyelesaikan masalah ketidakseimbangan jumlah data antar kelas, kode ini digunakan untuk menghitung dan menerapkan berat kelas pada dataset pelatihan. Untuk memulai prosedur, label kelas diambil dari dataset pelatihan (train_df['label']), yang berisi informasi kelas untuk setiap data gambar. Selain itu, library sklearn.utils.class_weight memiliki fungsi compute_class_weight dengan parameter class_weight='balanced. Dengan demikian, bobot kelas secara otomatis dihitung berdasarkan proporsi jumlah data, dengan kelas dengan jumlah data yang lebih sedikit diberikan bobot yang lebih besar daripada kelas dengan jumlah data yang lebih banyak. Metode ini memungkinkan model untuk "memperhatikan" kelas minoritas selama proses pelatihan sehingga tidak terlalu bias terhadap kelas mayoritas.

Dengan menggunakan fungsi enumerate, hasil perhitungan bobot kelas dikonversi ke dalam kosa kata. Ini membuat formatnya sesuai dan mudah digunakan untuk platform pengajaran mendalam seperti Keras atau TensorFlow. Dictionary ini menyediakan pasangan indeks kelas dan bobotnya; saat proses model berjalan, pasangan ini dapat dimasukkan ke parameter class_weight.sesuai(). Dengan menggunakan berat kelas, kesalahan prediksi pada kelas minoritas akan dihukum lebih berat dibandingkan kelas mayoritas. Hal ini mendorong model untuk belajar karakteristik yang lebih representatif untuk setiap kelas.

Classification Report:

	precision	recall	f1-score	support
balinese	0.89	0.68	0.77	155
batak	0.20	0.47	0.28	19
dayak	0.23	0.64	0.34	14
javanese	0.49	0.68	0.57	50
minangkabau	0.76	0.52	0.62	113
accuracy			0.62	351
macro avg	0.51	0.60	0.52	351
weighted avg	0.73	0.62	0.65	351

Gambar 3. 30 Hasil InceptionV3

Hasil classification report ini menunjukkan bahwa performa model berbeda-beda pada setiap kelas. Kelas balinese memiliki performa cukup baik dengan precision tinggi (0.89), yang berarti sebagian besar prediksi balinese sudah tepat, meskipun recall-nya masih sedang (0.68) sehingga masih ada data balinese yang belum berhasil terdeteksi. Kelas minangkabau juga menunjukkan performa yang relatif stabil dengan f1-score 0.62. Sementara itu, kelas javanese berada di tingkat menengah dengan keseimbangan precision dan recall yang cukup baik, walaupun belum optimal.

Sebaliknya, kelas batak dan dayak masih memiliki kinerja yang buruk, terutama pada skor precision dan f1. Ini mungkin karena jumlah data yang lebih sedikit daripada kelas lain (support kecil), sehingga model

kesulitan mempelajari ciri khasnya secara konsisten. Secara keseluruhan, akurasi model adalah 62%, dengan skor rata-rata f1 sebesar 0,65. Ini menunjukkan bahwa model cenderung lebih baik pada kelas mayoritas, tetapi masih perlu lebih banyak upaya untuk memperbaiki ketidakseimbangan data dan meningkatkan kinerja kelas minoritas.

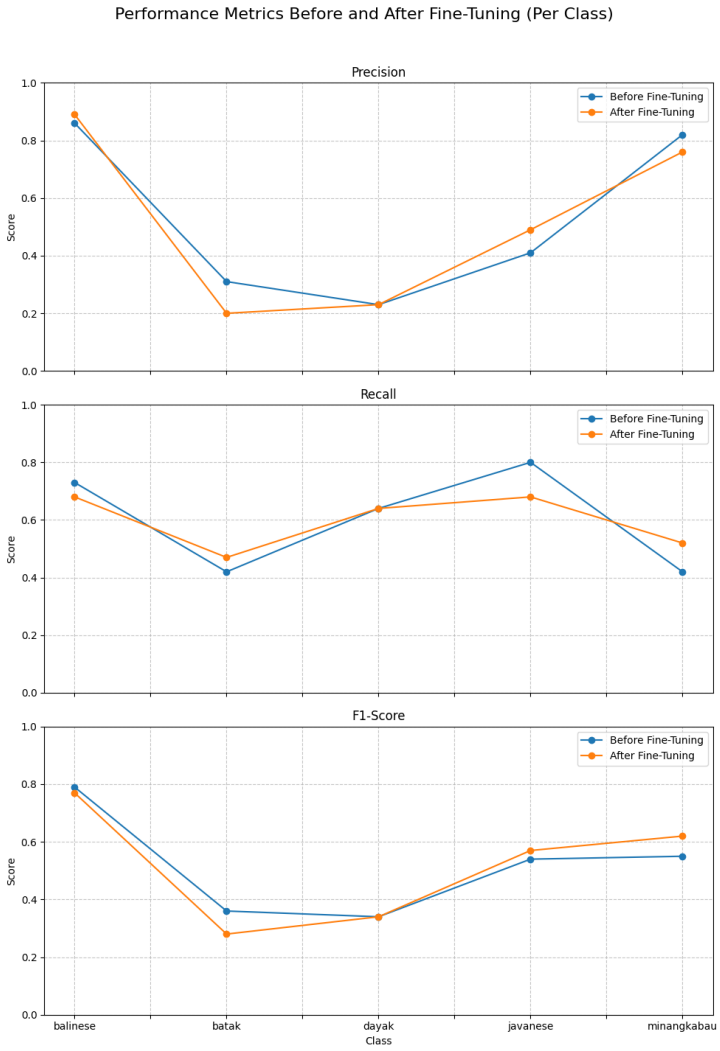
```
# It's important to re-freeze earlier layers, and only fine-tune the top layers
for layer in base_model_inception.layers[:-20]: # Unfreeze the last 20 layers
    layer.trainable = False

# Recompile the model with a lower learning rate for fine-tuning
model_inception.compile(optimizer=Adam(learning_rate=0.00001),  # Lower learning rate
                        loss='categorical_crossentropy',
                        metrics=['accuracy'])

print("Model recompiled for fine-tuning. Displaying updated summary:")
model_inception.summary()

# Continue training (fine-tuning) for an additional 10 epochs
print("\nStarting fine-tuning for 10 epochs...")
history_finetune = model_inception.fit(
    train_generator,
    epochs=10, # Additional 10 epochs for fine-tuning
    validation_data=val_generator,
    class_weight=class_weights_dict
)
```

Gambar 3. 31 Lakukan Fine Tuning

Agar lebih akurat dalam mengenali data baru, kode ini menunjukkan langkah-langkah fine-tuning pada model Inception. Pada awalnya, sebagian besar lapisan model dibuat statis secara sengaja, dan hanya dua puluh lapisan terakhir yang dibiarkan terbuka untuk penyesuaian ulang. Pola-pola umum dari model sebelumnya tetap ada, sementara bagian ujung model dipoles kembali untuk menjadi lebih spesifik dan peka terhadap detail khusus dari gambar dalam dataset, yang membuat pendekatan ini sangat efektif.

Penurunan tingkat pemahaman pada optimizer Adam juga membantu proses fine-tuning ini, yang bertujuan untuk melakukan pembaruan bobot secara lebih halus dan terkontrol. Dengan kecepatan belajar yang rendah, kita dapat menghindari perubahan bobot yang terlalu besar. Ini memastikan bahwa pengetahuan awal kita tentang model dari dataset besar sebelumnya tidak rusak. Dengan menerapkan kelas berat dan pelatihan lanjutan selama berbagai periode, model tidak hanya berkonsentrasi pada kelas dengan jumlah data yang signifikan tetapi juga belajar lebih adil tentang kelas minoritas. Kombinasi strategi ini

memungkinkan model Inception untuk melakukan generalisasi yang lebih baik dan menghasilkan kinerja yang lebih stabil dan akurat ketika digunakan untuk mengenali data gambar baru.



Gambar 3. 32 Hasil Setelah Fine Tuning

Sebuah visualisasi metrik kinerja sebelum dan sesudah fine-tuning untuk setiap kelas menunjukkan dampak proses fine-tuning pada setiap kelas adat kebudayaan. Pada metrik ketepatan, kelas Balinese dan Minangkabau menunjukkan nilai yang relatif tinggi baik sebelum maupun sesudah fine-tuning. Ini menunjukkan bahwa, dengan tingkat kesalahan false positive yang rendah, model cukup konsisten dalam memprediksi kedua kelas tersebut. Namun, kelas Batak menunjukkan

penurunan ketepatan setelah penyesuaian khusus, yang menunjukkan bahwa model menjadi lebih sering salah mengidentifikasi kelas lain sebagai Batak. Sebaliknya, kelas Dayak cenderung stabil dengan perubahan kecil, dan kelas Javanese justru menunjukkan peningkatan ketepatan setelah penyesuaian khusus, menunjukkan bahwa proses penyesuaian bobot model berhasil membantu mengidentifikasi ciri visual khas kelas tersebut.

Pada metrik recall, fine-tuning cenderung meningkatkan kemampuan model untuk mengenali data aktual dari beberapa kelas, terutama Batak dan Minangkabau, di mana nilai recall setelah fine-tuning lebih tinggi daripada sebelumnya. Ini menunjukkan bahwa model menjadi lebih peka terhadap variasi gambar pada kelas-kelas ini, sehingga mengurangi jumlah false negative. Namun, terjadi penurunan recall pada kelas Balinese dan Javanese, menunjukkan bahwa meskipun prediksi menjadi lebih akurat, masih ada beberapa data aktual yang belum dikenali dengan baik. Kelas Dayak menunjukkan nilai recall yang relatif stabil sebelum dan sesudah fine-tuning, menunjukkan bahwa karakteristik visual kelas ini sudah cukup terwakili sejak awal proses pembelajaran.

Dampak fine-tuning lebih jelas terlihat pada peningkatan performa beberapa kelas, terutama Javanese dan Minangkabau. Kenaikan F1-score pada kedua kelas tersebut menunjukkan bahwa fine-tuning mampu memperbaiki keseimbangan antara ketepatan dan kelengkapan prediksi. Sebaliknya, penurunan F1-score pada kelas Batak menunjukkan adanya trade-off antara precision dan recall y Secara keseluruhan, temuan ini menunjukkan bahwa penyesuaian meningkatkan kinerja model pada kelas tertentu berkontribusi secara positif. Namun, mereka juga menegaskan bahwa penyesuaian berbagai strategi penyesuaian, termasuk pemilihan tingkat pembelajaran, jumlah epoch, dan distribusi data, diperlukan agar peningkatan kinerja dapat lebih merata di seluruh kelas adat kebudayaan.

3.3.2.4 Tahap 4. Hasil komparasi model

Tabel 3.6. Hasil Komparasi model

No	Model	Hasil Akurasi
1	Densenet121	0.74 (74%)
2	Restnet50	0.74 (74%)
3	InceptionV3	0.62 (62%)

Hasil komparasi dari tiga model transfer learning yang digunakan, Densenet121, ResNet50, dan InceptionV3, menunjukkan bahwa kedua model tersebut menunjukkan kinerja terbaik dengan nilai akurasi yang sama, yaitu 74%. Ini menunjukkan bahwa, meskipun karakteristik visual antar kelas seperti Bali, Jawa, Batak, Dayak, dan Minangkabau memiliki kemampuan yang lebih baik untuk mengekstraksi fitur visual dari gambar adat kebudayaan.

Keunggulan dari model Densenet121 adalah arsitekturnya yang menghubungkan setiap lapisan secara langsung, yang memungkinkan aliran informasi fitur yang lebih baik dan mengurangi kehilangan informasi selama proses pelatihan. Arsitektur ini membuat Densenet121 cukup stabil dalam memahami gambar adat kebudayaan. Dengan demikian, ResNet50, yang menggunakan konsep hubungan residual, dapat mengatasi masalah vanishing gradient dan mempertahankan kinerja yang baik meskipun memiliki arsitektur yang dalam, menghasilkan akurasi yang sebanding dengan Densenet121.

Berbeda dengan dua model sebelumnya, InceptionV3 memiliki akurasi yang lebih rendah sebesar 62%. Ini menunjukkan bahwa model ini kurang efektif dalam mengidentifikasi pola tertentu pada kumpulan data yang digunakan. Ini mungkin karena cara InceptionV3 mengekstraksi fitur melalui berbagai ukuran kernel, yang mungkin tidak sesuai dengan fitur visual dataset kebudayaan konvensional. Secara

keseluruhan, hasil komparasi ini menunjukkan bahwa, pada dataset yang digunakan, Densenet121 dan ResNet50 lebih cocok untuk tugas klasifikasi gambar adat kebudayaan daripada InceptionV3.

3.3. Kendala yang Ditemukan

Selama proses pengembangan dan pengujian model klasifikasi citra rumah adat di Indonesia, terdapat beberapa kendala yang dihadapi Kendala-kendala tersebut diuraikan sebagai berikut:

a. Ketidak seimbang Dataset

Dataset yang digunakan memiliki distribusi jumlah data yang tidak merata antar kelas. Beberapa kelas memiliki jumlah citra yang jauh lebih banyak dibandingkan kelas lainnya. Kondisi ini menyebabkan model cenderung bias terhadap kelas mayoritas, sehingga memengaruhi performa klasifikasi pada kelas minoritas dan menurunkan nilai metrik evaluasi seperti *recall* dan *F1-score*.

b. Performa Model tidak stabil

Pada tahap awal proses pelatihan, beberapa model menunjukkan fluktuasi nilai *loss* dan *accuracy* yang cukup signifikan. Hal ini mengindikasikan bahwa model belum mampu melakukan konvergensi dengan baik, terutama pada arsitektur yang memiliki jumlah parameter besar serta pada penggunaan *learning rate* yang kurang optimal.

c. Waktu running yang cukup panjang

Proses eksperimen membutuhkan waktu yang cukup lama karena dilakukan pengujian terhadap beberapa arsitektur *deep learning* serta proses *hyperparameter tuning* yang memerlukan banyak iterasi. Selain itu, keterbatasan sumber daya komputasi turut memengaruhi durasi pelatihan model.

3.3.1. Solusi atas Kendala yang Ditemukan

Dalam permasalahan dan kendala terdapat solusi yang didapatkan. Pada hal ini maka diadakan beberapa solusi yang dilakukan dalam melakukan pengerjaan project dan lomba ini. Berikut solusi yang dilakukan:

a. Penerapan Data Augmentasi

Untuk mengurangi dampak ketidakseimbangan data, dilakukan teknik *data augmentation* pada kelas minoritas serta penerapan *class balancing* pada data latih. Pendekatan ini bertujuan untuk memperbaiki distribusi data antar kelas sehingga model dapat mempelajari karakteristik setiap kelas secara lebih seimbang.

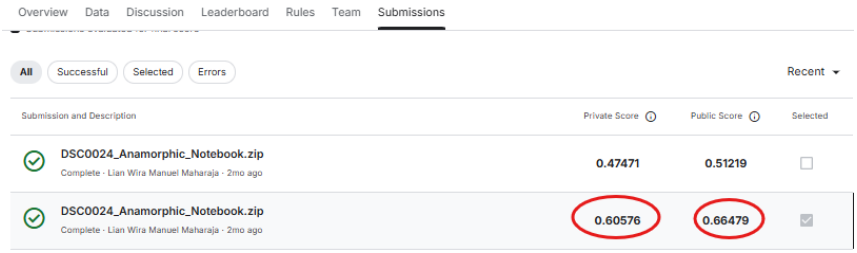
b. Pengunaan F1 Score sebagai Evaluasi

Selain menggunakan metrik akurasi, penulis menerapkan *F1-score* (khususnya *macro F1-score*) sebagai acuan evaluasi utama. Metrik ini dipilih karena mampu memberikan gambaran performa model yang lebih adil pada dataset tidak seimbang, dengan mempertimbangkan nilai *precision* dan *recall* pada setiap kelas.

3.4 Hasil Lomba/Kompetisi

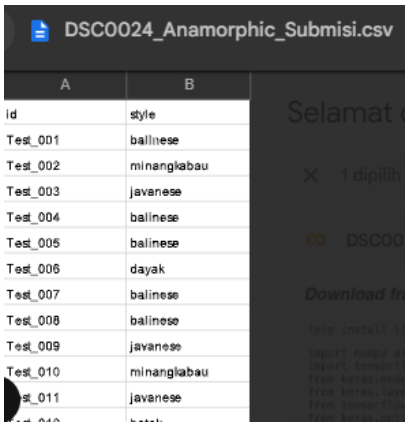
Setelah melalui seluruh tahapan penelitian yang meliputi pengumpulan data (data collecting), pra-pemrosesan data (data pre-processing), pemodelan (data modeling), pelatihan model (training), hingga evaluasi model, maka proses pengembangan model klasifikasi citra rumah adat di Indonesia telah selesai dilakukan. Model yang dikembangkan menunjukkan performa yang baik berdasarkan metrik evaluasi internal, khususnya pada nilai akurasi dan validation performance. Namun demikian, ketika dilakukan proses submission pada platform Kaggle dalam ajang Data Science Competition (DSC) LOGIKA UI 2025, performa model yang diperoleh pada sistem penilaian eksternal masih belum mencapai nilai optimal. Hal ini disebabkan oleh perbedaan skema evaluasi, distribusi data uji tersembunyi (hidden test set), serta tingkat kompleksitas data pada kompetisi tersebut.

Gambar 3.31. menunjukkan hasil leaderboard setelah model dikumpulkan pada halaman kompetisi LOGIKA UI 2025 di Kaggle.com. Berdasarkan hasil tersebut, tim peneliti memperoleh peringkat ke-60 dari total 94 kelompok peserta. Model yang dikumpulkan menghasilkan private score sebesar 0.60576 dan public score sebesar 0.66479, yang mencerminkan performa model dalam mengklasifikasikan citra rumah adat pada data uji kompetisi.



Gambar 3. 33 Pengumpulan Hasil Lomba

Proses pengumpulan hasil prediksi dilakukan dengan mengunggah file berformat CSV yang telah disusun sesuai dengan ketentuan kompetisi. File CSV tersebut terdiri dari dua kolom utama, yaitu id yang merepresentasikan nama file citra, serta style yang menunjukkan kategori hasil klasifikasi citra rumah adat. Contoh format file submission CSV ditunjukkan pada Gambar 3.32



Gambar 3. 34 Hasil Prediksi Lomba